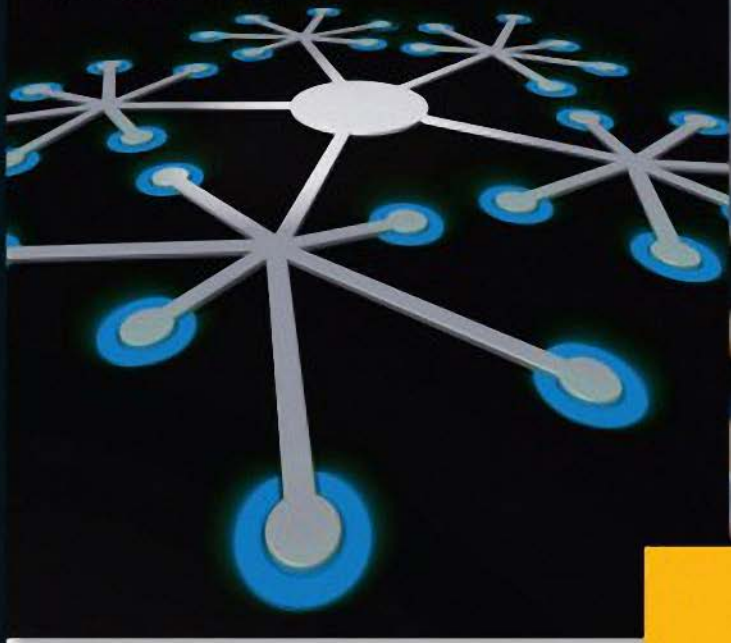
Michael Kamleitner



```
<body>
    <h1>Welcome to our tab!</h1>
    <? if ($signed_request["page"]["liked"] ==
    <p>Danke, dass du Fan geworden bist! Der C
    <? } else { ?>
    <p>Bitte <b>werde Fan unserer Seite</b>, u
    auf „Gefällt mir"!</p>
    <? } ?>
```

Facebook-Programmierung

Entwicklung von Social Apps & Websites

Aktuell zu Timeline und Open Graph

- Grundlagen und Konzeption von Facebook-Anwendungen
- Autorisierung, Graph API, FQL, Facebook JavaScript SDK
- Externe Websites anbinden, Social Plugins, Credits, Mobile Web, Internationalisierung

DVD ROM Alle Programmierbeispiele aus dem Buch, Tools zur Facebook-Programmierung

Galileo Computing

Michael Kamleitner

Facebook-Programmierung

Entwicklung von Social Apps & Websites

Galileo Press

Liebe Leserin, lieber Leser,

die offene Architektur von Facebook bietet viele Optionen der Individualisierung der eigenen Unternehmens-Präsenz sowie die Möglichkeit, eigene Webanwendungen oder ganze Websites in die Plattform zu integrieren. Timeline-Profile und Open Graph eröffnen Anwendern völlig neue Wege zum gemeinsamen Entdecken von digitalen Inhalten – ein neues Paradigma, das Content-Anbieter wie Spotify oder Netflix bereits erfolgreich zur Traffic-Steigerung nutzen. Für Webentwickler stellt sich die Facebook-Plattform als spannendes, neues Betätigungsfeld dar, für dessen Nutzung neben PHP- und JavaScript-Know-How auch die Kenntnis der zahlreichen Facebook-Schnittstellen erforderlich ist.

Michael Kamleitner von der Agentur »Die Socialisten« verfügt über jahrelange Erfahrung in der Entwicklung von Facebook-Anwendungen. Er führt Sie Schritt für Schritt in die Konzepte der Facebook-Anwendungs-Entwicklung mit vielen Praxisbeispielen ein. Von grundlegenden Aufgaben wie z.B. der Nutzung des Facebook-Kontos zum Login in der eigenen Web-Anwendung bis hin zu brandaktuellen Themen wie der Nutzung des neuen Open Graph deckt dieses Buch dabei die meisten Bereiche der Facebook-Plattform ab.

Um die Qualität unserer Bücher zu gewährleisten, stellen wir stets hohe Ansprüche an Autoren und Lektorat. Falls Sie dennoch Anmerkungen und Vorschläge zu diesem Buch formulieren möchten, so freue ich mich über Ihre Rückmeldung.

Viel Vergnügen beim Lesen!

Ihr Stephan Mattescheck
Lektorat Galileo Computing

Stephan.Mattescheck@galileo-press.de
www.galileocomputing.de
Galileo Press · Rheinwerkallee 4 · 53227 Bonn

Gewidmet meinen Eltern,
Franz und Elisabeth

Auf einen Blick

Der Name Galileo Press geht auf den italienischen Mathematiker und Philosophen Galileo Galilei (1564 – 1642) zurück. Er gilt als Gründungsfigur der neuzeitlichen Wissenschaft und wurde berühmt als Verfechter des modernen, heliozentrischen Weltbilds. Legendär ist sein Ausspruch *Eppur si muove* (Und sie bewegt sich doch). Das Emblem von Galileo Press ist der Jupiter, umkreist von den vier Galileischen Monden. Galilei entdeckte die nach ihm benannten Monde 1610.

Lektorat Stephan Matteschеck
Korrektorat Alexandra Müller, Olfen
Einbandgestaltung Barbara Thoben
Titelbild Hauptbild: © Stefan Rajewski – Fotolia.com; Button: © Already – Fotolia.com; Mann: Jetta Produktions/Getty Images
Typografie und Layout Vera Brauner
Herstellung Steffi Ehrentraut
Satz SatzPro, Krefeld
Druck und Bindung Beltz Druckpartner, Hemsbach

Gerne stehen wir Ihnen mit Rat und Tat zur Seite

stephan.matteschеck@galileo-press.de
bei Fragen und Anmerkungen zum Inhalt des Buches

service@galileo-press.de
für versandkostenfreie Bestellungen und Reklamationen

britta.behrens@galileo-press.de
für Rezensions- und Schulungsexemplare

Dieses Buch wurde gesetzt aus der TheAntiquaB (9,35/13,7 pt) in FrameMaker.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN 978-3-8362-1843-6

1. Auflage 2012

Inhalt

Vorwort

Über zweitausend Paar Hände applaudieren, als Mark Zuckerberg, Gründer und CEO von Facebook Inc., am 22. September 2011 auf der Entwicklerkonferenz »f8« (ausgesprochen »eff eight«, obwohl »fate«, zu Deutsch »Schicksal« oder »Fügung«, nicht unpassend wäre) im San Francisco Design Center die strategische Ausrichtung der Facebook-Plattform für das nächste Jahr präsentiert. Über zweitausend Besucher sind gekommen. Die meisten von ihnen sind Software-Entwickler, die teilweise bereits seit 2007, als Facebook sein soziales Netzwerk erstmals für Drittthersteller von Applikationen und Anbieter von Inhalten geöffnet hat, ihren kreativen Schwerpunkt und damit auch ihr wirtschaftliches Geschick (»fate«) zu einem beträchtlichen Teil in die Hände von Zuckerberg und seinem Plattform-Team gelegt haben. Damals von vielen Online-Afficionados und Webentwicklern noch aufgrund der drohenden, hohen Abhängigkeit von einem Plattform-Hersteller mitleidig belächelt, sieht die Welt im Social Web des Jahres 2011 anders aus.

Mit über einer Milliarde aktiver Benutzer weltweit führt heute kein Weg mehr an Facebook und der Facebook-Plattform vorbei. Kaum eine neue Webapplikation, die ihren Benutzern nicht die Möglichkeit bietet, sich über ihr Facebook-Konto einzuloggen. Kaum eine Online-Marketing-Kampagne, die nicht von Maßnahmen auf Facebook flankiert wird, elementar auf dem sozialen Netzwerk basiert oder überhaupt ausschließlich in diesem Kanal stattfindet. Und ganz bestimmt keine Content-Anbieter, die nicht zumindest mittels integriertem LIKE-Button versuchen, ihre Inhalte in der riesigen Masse der Facebook-Benutzer effizienter oder gar »viral« zu verbreiten.

Für Software-Entwickler – sei es die freiberufliche One-WoMan-Show, das ambitionierte Startup oder die große Digital-Agentur – gehören Kenntnisse der Facebook-Plattform damit zunehmend zum Standardrepertoire, das Kunden heute zu Recht fordern. War die Facebook-Plattform in den ersten Jahren noch durch mehr oder weniger proprietäre Technologien wie FBML für Webentwickler oft schwer zugänglich, so ist der Einstieg in die Anwendungsentwicklung durch konsequentes Setzen auf offene Standards wie OAuth und HTTP-REST heute wesentlich leichter. Und dennoch: Erfolgreiche Facebook-Anwendungen sind mehr als in einen iFrame gepresste Webseiten, eine Spezialisierung und umfassendere Beschäftigung mit den von Facebook angebotenen Schnittstellen lohnt sich also. Zudem ändert und erweitert Facebook getreu dem Firmenmotto »Move fast and break things« die bereitgestellten Schnittstellen in vergleichsweise kurzen Zeitabständen – sich bloß »nebenbei« mit der Facebook-Plattform auseinanderzusetzen erscheint also fast unmöglich.

Um keine Missverständnisse aufkommen zu lassen: Software-Entwickler, die sich auf den schnelllebigen Bereich des Social Webs spezialisieren möchten, müssen auch künftig die Augen offen halten. Wenngleich die Facebook-Plattform heute und in der nahen Zukunft hier eine dominante Rolle spielen wird, lohnt es sich, schon frühzeitig mit neuen Plattformen wie etwa Google+ zu experimentieren. Eine Pluralität der genutzten Plattformen sorgt letztlich durch Konkurrenz für beschleunigte Innovation – davon profitieren sowohl Benutzer als auch Dritthersteller von Anwendungen.

Mit diesem Buch möchte ich Ihnen, lieber Leser, also die Anwendungsentwicklung auf der Facebook-Plattform in ihren verschiedenen Formen schmackhaft machen. Sie sollen dabei nicht nur das notwendige technische Rüstzeug zur Nutzung der zahlreichen Programmierschnittstellen vermittelt bekommen. Darüber hinaus soll Ihnen dieses Buch auch den Einstieg in das *Social Application Design* erleichtern. Es geht dabei um Antworten auf die grundlegende Frage, wie eine herkömmliche Webanwendung im Kontext sozialer Netzwerke ihr volles Potenzial ausschöpfen kann. In diesem Sinne: Ich hoffe, mit diesem Buch einen Beitrag zu neuen, noch »sozialeren« Webanwendungen leisten zu können, und wünsche Ihnen: »Happy coding!«

Michael Kamleitner
San Francisco

Zur Benutzung dieses Buches

Entwicklern, die mit diesem Buch ihre ersten Schritte auf der Facebook-Plattform tun, empfehle ich, die Kapitel 2 bis 7 in der vorgesehenen Reihenfolge durchzuarbeiten, da hier die wesentlichsten Eckpfeiler der Facebook-Plattform erklärt werden. Dies ist auch aufgrund der starken Abhängigkeiten zwischen den einzelnen Schnittstellen der Plattform sinnvoll.

Kapitel 1, »Einstieg in die Anwendungsentwicklung auf der Facebook-Plattform«, gibt einen allgemeinen Überblick über die Möglichkeiten, die die Facebook-Plattform dem Webanwendungsentwickler bietet. Hier finden Sie auch Hinweise zur Hosting-Umgebung, die Sie zur Ausführung von Facebook-Anwendungen bereitstellen sollten. In einem kurzen Tutorial wird erklärt, wie Sie Ihre eigene Hosting-Umgebung am lokalen PC einrichten können. Danach geht es auch schon an die erste eigene Applikation – Hello Facebook!

In Kapitel 2, »Authentifikation und Autorisierung«, wird gezeigt, wie Sie die Benutzer Ihrer Anwendung anhand ihres Facebook-Kontos identifizieren können. Das Berechtigungssystem von Facebook gibt Ihnen dabei umfangreiche Zugriffsmöglichkeiten auf die Profilinformationen Ihrer Benutzer.

Kapitel 3, »Die Graph API«, und Kapitel 5, »Die Facebook Query Language (FQL)«, behandeln die beiden zentralen Schnittstellen zum Zugriff auf die in Facebook gespeicherten Daten in großer Ausführlichkeit – diese Kapitel sollen auch als Nachschlage-Referenz dienen und müssen nicht unbedingt vollständig durchgelesen werden.

Kapitel 4, »Das JavaScript SDK«, widmet sich der clientseitigen Integration von Facebook in Ihre Anwendung. Sie lernen, wie Sie Standarddialoge von Facebook verwenden und Ihre Anwendungen nahtlos in das Application Canvas von Facebook integrieren können.

In Kapitel 6, »Social Plugins«, werden die zahlreichen Facebook-Widgets vorgestellt, die Sie in Ihre Webseiten und -anwendungen integrieren können. Vom allseits bekannten LIKE-Button über mächtige Möglichkeiten der Personalisierung bis hin zum Live-Chat hat Facebook einiges zu bieten.

Die Umsetzung mobiler, webbasierter Facebook-Anwendungen wird in Kapitel 7, »Mobile Webanwendungen auf Facebook«, beschrieben.

Kapitel 8, »Open Graph«, deckt die jüngste Plattform-Erweiterung von Facebook, den neuen *Open Graph*, ab. Lernen Sie, wie Sie beliebige Benutzeraktionen auf dem Timeline-Profil und im Ticker veröffentlichen können.

In Kapitel 9, »Facebook Credits«, erfahren Sie, wie Sie mit dem Mikropayment-Service *Facebook Credits* Zahlungen von den Benutzern Ihrer Anwendung entgegennehmen können.

Die abschließenden Anhänge behandeln besondere Themen wie etwa das Application Hosting mit dem Cloud-Provider Heroku, die Verwendung des offiziellen PHP SDKs oder die Internationalisierung von Facebook-Anwendungen.

Um den Einstieg in die Entwicklung von Facebook-Anwendungen so praxisnah wie möglich zu gestalten, finden Sie in diesem Buch zahlreiche Code-Beispiele zu den behandelten Inhalten. Diese Code-Beispiele sind auch auf der DVD enthalten, die diesem Buch beigelegt ist. Zum einfacheren Auffinden beginnen die Dateinamen dabei immer mit der Nummer des zugehörigen Kapitels.

Webseite zum Buch

Die Webseite zum Buch finden Sie unter der URL *http://book.socialisten.at*. Eine Liste mit weiterführenden Links zum Thema Facebook-Entwicklung hält Sie immer auf dem laufenden Stand. Zudem finden Sie hier etwaige Aktualisierungen der Code- Beispiele.

Kapitel 1
Einstieg in die Anwendungsentwicklung auf der Facebook-Plattform

Dieses Kapitel soll Ihnen einen Überblick über die kurze, aber rasante Geschichte von Facebook geben und Ihnen zeigen, welche Arten von Anwendungen Sie auf der Plattform heute realisieren können. Sie erfahren, welche Voraussetzungen Sie im Hinblick auf Programmiersprachen und Hosting erfüllen müssen, und werden Ihre erste eigene Facebook-Applikation entwickeln.

Im Sommer 2007 öffnete Facebook anlässlich der seither jährlich stattfindenden Facebook-Entwicklerkonferenz »f8« das damals in den USA bereits beliebte soziale Netzwerk für Möglichkeiten zur Entwicklung von Applikationen für Dritthersteller. Facebook war damals weit von der dominierenden Rolle im Social Web entfernt, die es mit knapp einer Milliarde Benutzern heute einnimmt. Die länger am Markt etablierten Konkurrenten MySpace oder im deutschsprachigen Business-Bereich auch XING (früher OpenBC) waren die Platzhirsche am Markt der sozialen Netzwerke, darüber hinaus wurden nischen-spezifische Services wie Twitter (Kurznachrichten-Service), Flickr (Photo Sharing) und Last.fm (Musik) im Zuge des Hypes um »Web 2.0« beinahe täglich gelauncht (und oft auch ebenso schnell wieder eingestellt).

1.1 Überblick über die Facebook-Plattform

Mit der ersten Veröffentlichung der »Facebook-Plattform« 2007 war es nun plötzlich unabhängigen Entwicklern aus aller Welt möglich, individuelle eigene Anwendungen und Inhalte innerhalb des sozialen Netzwerks Facebook.com zu entwickeln, auszuführen und – in vielen erfolgreichen Fällen – auch zu amortisieren. Die sogenannten *Canvas-Apps* der Dritthersteller wurden dabei in einen von Facebook bereitgestellten »Rahmen« (bestehend aus Kopfbereich, Fußbereich und Seitenleiste) integriert. Aus Sicht des Anwenders ist diese Form des Ineinandergreifens von Anwendung und Plattform so transparent, dass Funktionalitäten von Facebook und solche der externen Anwendungen wie aus einem Guss wirken.

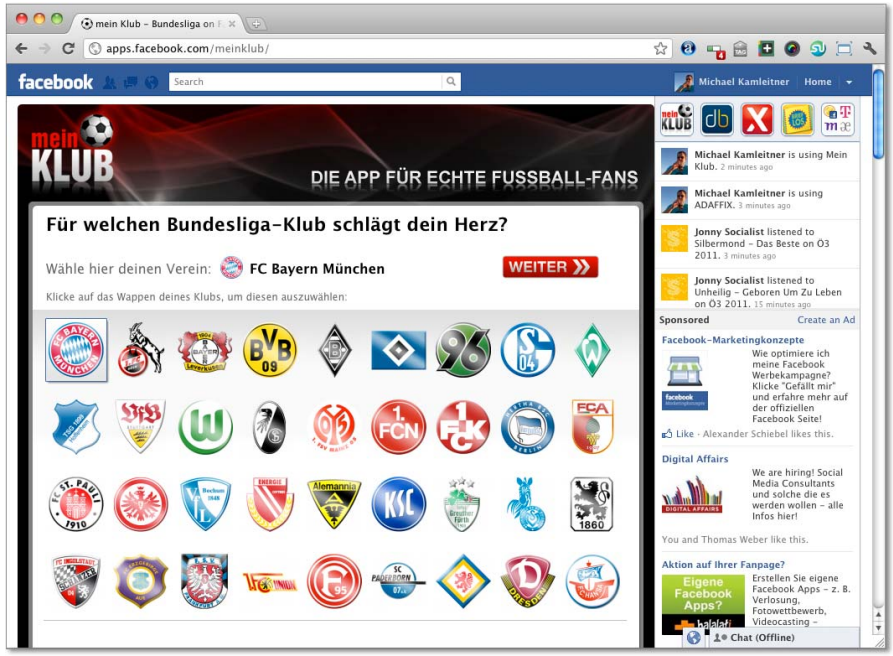


Abbildung 1.1 Canvas-App »mein Klub« auf Facebook.com

Aus damaliger Sicht war dieser Schritt eine bahnbrechende Neuerung, ein Schritt zur Offenheit, den andere Betreiber von Webservices bis dato nicht gewagt hatten. Zwar war es seit der Ära des »Web 2.0« bereits durchaus üblich, Drittherstellern das Andocken an den eigenen Service mit *Programmierschnittstellen (APIs – Application Programming Interfaces)* zu ermöglichen, jedoch liefen diese angekoppelten Anwendungen immer außerhalb der Kernplattform, etwa auf einer vom Dritthersteller betriebenen Website und Domain oder als lokal am PC installierte Software. Beispiele für erfolgreiche APIs der »Web 2.0«-Ära sind der Fotoservice Flickr oder der Kurznachrichten-Service Twitter.

Software-Plattformen im Social Web

Der Begriff der *Software-Plattform* ist mit Bedeutungen und Interpretationen reichlich überladen. Es lohnt sich also, diese genauer zu definieren. Marc Andreessen, Gründer von Netscape und Ning.com, definiert Plattform etwa »als ein System oder Software-Produkt, das von vom eigentlichen Hersteller unabhängigen Entwicklern mittels Programmierschnittstellen angepasst und erweitert werden kann«. Etwas einfacher gesagt: *»If you can program it, then it's a platform, if you can't, then it's not«*. Andreessen, der selbst bedeutend an der Entstehung des WWW beteiligt war, bietet darüber hinaus eine gute Kategorisierung von webbasierten Software-Plattformen an. Anhand der Kategorisierung ist ersichtlich, welche Arten von Anwendun-

gen ein Entwickler auf einer Plattform umsetzen kann und welche Voraussetzungen er dafür erfüllen muss:

- *Level 1 – Access API:* Webplattformen, die eine Access API besitzen, erlauben Entwicklern den Zugriff (*Access*) auf Daten innerhalb der Plattform. So können etwa über die Twitter-API (ein typisches Beispiel für eine Level-1-API) Kurznachrichten eines Benutzers gelesen und geschrieben werden. Anwendungen, die auf einer Level-1-API aufbauen, sind allerdings in keiner Weise in die Benutzeroberfläche der Kernplattform eingebunden, sondern werden vielmehr auf einer eigenen Webseite oder auch als native Software-Anwendung (etwa als Twitter-Client für Windows) ausgeführt.
- *Level 2 – Plugin API:* Plattform-Anbieter, die Plugin APIs betreiben, gehen einen bedeutenden Schritt weiter: Sie gestatten es externen Anwendungsentwicklern, ihre Anwendungen in das eigene Produkt, die eigene Benutzeroberfläche, zu integrieren. Die bekannteste und heute erfolgreichste Level-2-API ist ohne Zweifel Facebook selbst. Durch die nahtlose Integration von externen Anwendungen mit dem eigentlichen Kernprodukt profitieren Anwendungsentwickler zusätzlich bei der Verbreitung und dem Marketing ihrer Anwendungen.
- *Level 3 – Runtime Environment:* Entwickler, die Level-1- oder Level-2-APIs nutzen, sind für den reibungslosen Betrieb ihrer Anwendungen selbst verantwortlich – sie müssen die passende Serverinfrastruktur bereitstellen, diese betreuen und bei wachsenden Benutzerzahlen auch skalieren. Der externe Anwendungscode wird niemals auf der Infrastruktur des Plattform-Anbieters ausgeführt. Stellt eine webbasierte Software-Plattform hingegen eine *Runtime Environment* (Laufzeitumgebung) zur Verfügung, bedeutet dies, dass Entwickler ihren Anwendungscode auf die Server der Plattform hochladen können, um ihn beim Plattform-Betreiber auszuführen. Der Plattform-Anbieter nimmt damit die technische und finanzielle Bürde, die die Bereitstellung der Infrastruktur darstellt, von den Schultern des Anwendungsentwicklers. Die derzeit bekannteste Laufzeitumgebung ist wohl App Engine von Google, die das Hosting und die Ausführung von Python-, Java- und Go-Anwendungen erlaubt. Facebook hat auf diesem Gebiet noch kein Angebot für Entwickler – eine Partnerschaft mit dem auf Cloud-Hosting von Ruby-on-Rails-Applikationen spezialisierten Betreiber Heroku (*http://www.heroku.com*) legt nahe, dass Facebook im Bereich Hosting vorerst nicht selbst auf den Markt drängen wird, sondern vielmehr mit Partnern zusammenarbeiten möchte.

Neben dem Betrieb von klassischen Canvas-Anwendungen auf Facebook.com bietet die Facebook-Plattform heute zahlreiche weitere Möglichkeiten für Dritthersteller, ihre Anwendungen mit der Plattform zu integrieren:

1. *Canvas-Applikationen:* Die ursprünglichen Canvas-Anwendungen sind auch heute noch verfügbar und erfreuen sich nach wie vor großer Beliebtheit unter Ent-

wicklern und Anwendern. Wie das Wort *Canvas* suggeriert, stellt Facebook hierbei eine leere »Leinwand« innerhalb von Facebook.com bereit, in die Anwendungsentwickler ihre Applikationen integrieren können. Technisch betrachtet, werden Canvas-Applikationen mittlerweile ausschließlich auf Basis von herkömmlichen iFrames realisiert. Dies war nicht immer so – zum Start der Facebook-Plattform 2007 wurden Facebook-Anwendungen noch nicht iFrame-basiert realisiert. Vielmehr wurden von Drittherstellern bereitgestellte Applikationsinhalte transparent durch die als Proxy agierenden Server von Facebook geschleust, dort aufbereitet, in die HTML-Struktur von Facebook.com integriert und schließlich an den Browser des Benutzers gesendet. Diese sogenannten *FBML-Anwendungen* (*Facebook Markup Language* – eine Facebook-spezifische Erweiterung von HTML) hatten gravierende Nachteile: Einerseits waren sie aufgrund des zusätzlichen »Umwegs« über die Facebook-Server meist aus Sicht des Benutzers recht langsam. Andererseits konnten Entwickler aus Sicherheitsgründen nicht alle üblichen Webtechnologien zur Entwicklung ihrer Anwendungen nutzen: AJAX-Aufrufe funktionierten nur über den Umweg der Facebook-Server und waren damit langsam, Flash-Elemente konnten nur eingeschränkt genutzt werden, und JavaScript wurde aus Sicherheitsgründen auf einen Facebook-spezifischen Dialekt namens *FBJS* (*Facebook JavaScript*) limitiert. Mit diesen Einschränkungen war es nicht einfach, zeitgemäß bedienbare Anwendungen im Stil der »Web 2.0«-Ära zu schreiben. Schon bald lieferte Facebook die iFrame-Unterstützung für Canvas-Apps nach.

Abbildung 1.2 Die Canvas-App »Adaffix« auf Facebook.com. Hervorgehoben ist die 760 Pixel breite »Leinwand«, die als iFrame realisiert ist.

Ein iFrame ist eine unabhängige »Webseite innerhalb einer Webseite« – damit können Entwickler in Sachen Webtechnologien aus dem Vollen schöpfen. Lediglich Applikationen, die als Tab/Reiter einer Page ausgeführt wurden, waren bis Februar 2011 auf die nachteilige FBML-Implementierung beschränkt – seitdem unterstützt Facebook allerdings auch in diesem Bereich iFrames, womit FBML-Anwendungen endgültig der Vergangenheit angehören. Die Leinwand von Canvas-Applikationen ist grundsätzlich auf eine Breite von 760 Pixeln bei beliebiger Höhe fixiert. Facebook bietet seit Kurzem auch die Möglichkeit, Canvas-Anwendungen, die breiter als 760 Pixel sind, zu realisieren. Diese sogenannten *Fluid-Canvas-Anwendungen* nutzen vor allem größere Bildschirme besser aus und sind z. B. für grafisch aufwendige Spiele geeignet.

2. *Canvas-Applikationen in Seiten-Tabs/Reitern:* Seit geraumer Zeit bietet Facebook die Möglichkeit, eigene Unternehmens- oder Markenseiten mit individuellen Reitern oder *Tabs* zu erweitern. Tabs/Reiter waren ursprünglich auch auf herkömmlichen Benutzerprofilen verfügbar – diese Möglichkeit hat Facebook allerdings Ende 2010 ersatzlos gestrichen. Der Name *Tabs* erklärt sich daraus, dass diese Applikationen früher visuell als Registerkarten am oberen Rand des Profils dargestellt wurden – seit der Umstellung auf Timeline-Profile für Pages werden Tabs/Reiter als grafische Buttons unter dem Cover-Foto der jeweiligen Page dargestellt.

Abbildung 1.3 Canvas-Applikation im Seiten-Tab »Austrian Airlines«. Hervorgehoben ist die 810 Pixel breite Leinwand, die als iFrame realisiert ist.

Seiten sind ähnlich aufgebaut wie Benutzerprofile, repräsentieren aber keine Personen, sondern ein Unternehmen, eine Marke oder eine Organisation. Tabs/Reiter sind eine hervorragende Möglichkeit, um die Facebook-Präsenz einer Firma zu individualisieren und um eigene Anwendungs- und Informationsangebote zu erweitern.

Beispiele sind etwa der Filialfinder eines Handelsunternehmens, die Darstellung der Firmenhistorie eines traditionsreichen Medienkonzerns oder auch die Durchführung der meisten Facebook-Gewinnspiele, die in Tabs der Markenseiten abgewickelt werden.

Auch Tabs/Reiter stellen eine – mit 810 Pixeln etwas breitere – Leinwand zur Verfügung, auf der individuelle Anwendungen ausgeführt werden können. Während Canvas-Applikationen schon länger iFrame-basiert umgesetzt werden, hat Facebook die iFrame-Unterstützung für Seiten-Tabs/Reiter erst im Februar 2011 verfügbar gemacht. Zuvor wurden auch Seiten-Tabs/Reiter über den Umweg der Facebook-Server ausgeliefert.

3. *Native, mobile Applikationen:* Facebook bietet offizielle *Software Development Kits* (SDKs) für die mobilen Anwendungsplattformen Apple iOS (iPhone und iPad) und Android an. Diese SDKs erlauben es Anwendungsentwicklern, Facebook-Funktionalitäten in ihre mobilen Apps zu integrieren. Unter *nativen Anwendungen* versteht man dabei speziell für die jeweilige Plattform kompilierte Anwendungen im Binärformat, also unter Android etwa Java-basierte und auf iOS Objective-C-basierte Anwendungen. Mobile Webanwendungen hingegen profitieren nicht von diesen SDKs – das macht aber nichts, da sie wie herkömmliche Webanwendungen mit Facebook verknüpft werden können!

Abbildung 1.4 iPhone-Applikation »Shazam« mit Facebook-Integration

4. *Webapplikationen:* Mit der Einführung der Technologie *Facebook Connect* im Jahr 2008 wurde die Facebook-Plattform um die Möglichkeit erweitert, Facebook-spezifische Funktionalitäten auf unabhängigen Websites und Webapplikationen von Drittherstellern zu verwenden. Damit müssen Facebook-Anwendungen nicht mehr unbedingt integriert auf der eigentlichen Plattform unter Facebook.com laufen, sondern können wahlweise auf einer eigenen Website, unter eigener Domain und im eigenständigen Look & Feel betrieben werden.

Abbildung 1.5 Webapplikation »mixcloud«. Hervorgehoben sind der Button zum Login per Facebook und der Autorisierungsdialog der Anwendung.

Ergänzend sei erwähnt, dass Facebook Anwendungsentwicklern auch für Canvas-Applikationen auf Facebook.com alle Freiheiten bezüglich der grafischen Gestaltung ihrer Anwendung lässt. Um eine möglichst nahtlose Integration mit dem »Rahmen« von Facebook.com zu erzielen, empfiehlt es sich dabei allerdings, gewisse von Facebook gesetzte Bedienstandards und grafische Auszeichnungen (etwa die farbliche Darstellung von Buttons in Dunkelblau) zu übernehmen. Einschränkend wirkt außerdem die Fixierung auf 810 bzw. 760 Pixel Breite.

5. Eine wichtige Eigenschaft von Facebook Connect ist die Möglichkeit, die eigene Facebook-Identität, also das eigene Facebook-Konto, auf externen Seiten und Anwendungen zu benutzen. So können Benutzer ihr Facebook-Konto verwenden, um sich mit einem Klick bei Webapplikationen anderer Hersteller anzumelden, ohne eine langwierige Registrierungsprozedur zu durchlaufen. Andererseits können bestimmte soziale Integrationspunkte wie etwa Wall-Postings oder Applika-

tionseinladungen auch auf diesen externen Websites verwendet werden. 2010 wurde Facebook Connect durch die umfassendere Facebook Graph API abgelöst, die gemeinsam mit der JavaScript SDK den Funktionsumfang des bisherigen Facebook Connect abdeckt. Dennoch wird der Begriff Facebook Connect auch heute noch oft verwendet, wenn es darum geht, eigenständige Websites und -anwendungen mit Facebook zu verknüpfen.

6. *Open Graph Protocol und Social Plugins:* Zusätzlich bietet Facebook mit dem ebenfalls 2010 eingeführten *Open Graph Protocol* die Möglichkeit, externe Webinhalte in den *Social Graph* von Facebook zu integrieren. Open Graph ist im Wesentlichen ein Set aus Metadaten, das Betreiber von contentlastigen Websites auf ihren einzelnen Seiten nutzen können, um Suchmaschinen, aber eben auch Facebook das sinnhafte Erfassen von Inhalten zu erleichtern. *Social Plugins* wie der bekannte LIKE-Button oder die RECOMMENDED-Box bieten dabei auch Seitenbetreibern ohne Programmierkenntnisse die Möglichkeit, eigene Webangebote in die Facebook-Plattform zu integrieren.

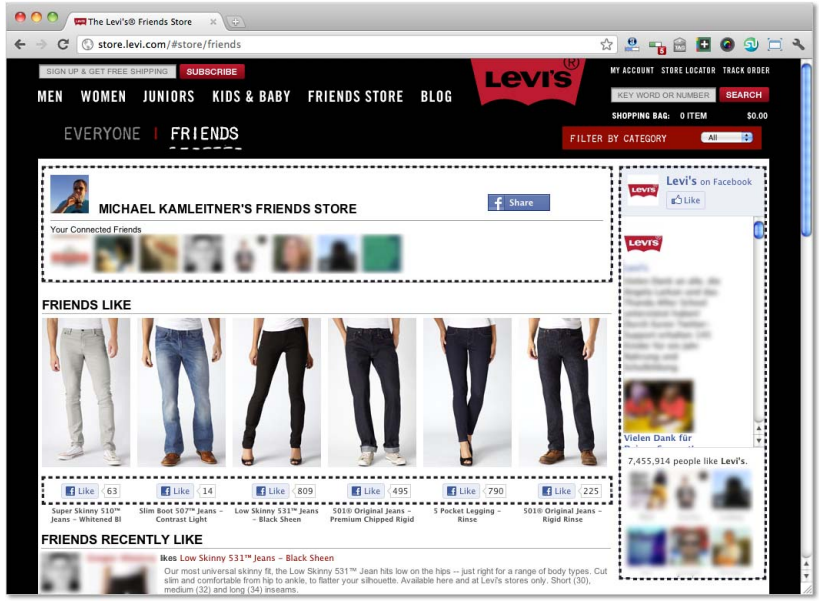


Abbildung 1.6 Webapplikation »Levis Friends Store«.
Hervorgehoben sind die eingebundenen Social Plugins.

Metadaten

Metadaten sind einfache Datenfelder, die im HTML-Code einer Seite hinterlegt werden, für den Benutzer aber nicht sichtbar sind. Metadaten liefern üblicherweise zusätzliche, maschinenlesbare Informationen zur Seite, auf der sie integriert sind.

Beispiele wären etwa der Autor eines Artikels auf der Webseite eines Nachrichten-Magazins oder die Adresse eines Restaurants auf der Webseite eines Travel Guides. Metadaten sind schon in der HTML-Spezifikation vorgesehen und können darüber hinaus anwendungsspezifisch erweitert werden – eine solche Erweiterung stellt das Open Graph Protocol dar.

Social Graph – der soziale Graph

Die Entwicklung von Software im Social Web im Allgemeinen und auf Facebook im Speziellen ist geprägt vom Gedanken, Anwendungen anhand des *sozialen Graphen* eines Benutzers zu personalisieren. Doch was ist nun eigentlich dieser soziale Graph?

Ursprünglich beschreibt der aus der Mathematik stammende Begriff *Graph* nichts anderes als das Netz an Freunden oder Kontakten, in dem ein Benutzer eines sozialen Netzwerks eingegliedert ist. Die Knoten repräsentieren dabei Personen, die Kanten zwischen den Knoten repräsentieren die Beziehungen zwischen den Personen – ursprünglich also beidseitige Freundschaftsbeziehungen. Der soziale Graph auf Facebook geht aber mittlerweile über das reine Freundschaftsgeflecht zwischen Benutzern hinaus – auch Firmen, Marken und Organisationen können mit ihren Präsenzen Teil des Social Graph werden. Die Kante zwischen einer Person und einer Firma ist dabei üblicherweise einseitig gerichtet und sagt aus, dass diese Person ein positive Beziehung zu der Firma ausdrücken möchte, indem sie auf der entsprechenden Seite auf Gefällt mir geklickt hat und damit ein »Fan« des Unternehmens geworden ist. Auch Gruppen, denen ein Facebook-Benutzer beitritt, können im sozialen Graphen dargestellt werden – die entsprechende Kante drückt dabei die Zugehörigkeit eines Benutzers zu einer Gruppe aus. Seit September 2011 können Personen auf Facebook darüber hinaus die öffentlichen Statusnachrichten einer anderen Person »abonnieren« – damit ist der soziale Graph auch um einseitig gerichtete Kanten zwischen Personen ergänzt worden. Das einseitige »Abonnieren« von Personen entspricht dem von Twitter schon länger bekannten Mechanismus des »Folgens«.

Warum ist das Konzept des sozialen Graphen nun für Sie als Anwendungsentwickler so wichtig? Nun, als Anwendungsentwickler haben Sie – mit der Zustimmung des Benutzers – umfangreiche Möglichkeiten, den sozialen Graphen Ihrer Benutzer zu lesen und zu bearbeiten. Ihre Anwendung »weiß« also, mit welchen Personen ein Benutzer unserer Applikation befreundet ist, für welche Firmen und Organisationen das Herz des Benutzers schlägt oder welchen Hobbys er oder sie nachgeht. Diese Informationen werden in erfolgreichen sozialen Webapplikationen verwendet, um die Applikation und ihre Inhalte für den einzelnen Benutzer zu personalisieren.

Ein gutes Beispiel wäre ein Fotowettbewerb, der als Facebook-Applikation umgesetzt werden soll. Wenn ein neuer Benutzer namens Jürgen erstmals bei dem Fotowettbewerb mitmacht, kann unsere Applikation im Hintergrund prüfen, ob etwa Freunde von Jürgen bereits vorher Fotos zum Wettbewerb eingereicht haben. In diesem Fall wäre es empfehlenswert, Jürgen einen entsprechenden Hinweis – »Fotos deiner Freunde Max, Peter und 3 weitere« – anzuzeigen. Jürgen wird anhand der Profilbilder sofort erkennen, dass seine Freunde bereits am Fotowettbewerb teilnehmen – er wird vermutlich weiterklicken, um die eingereichten Fotos seiner Freunde zu betrachten, wird diese eventuell auch kommentieren oder bewerten und wird ganz bestimmt auch motiviert sein, selbst in den Wettbewerb einzusteigen.

Zusammenfassend ist der soziale Graph also ein wichtiges Werkzeug, das Anwendungsentwickler zur Personalisierung von Inhalten und zur Förderung von Interaktion nutzen können.

Wichtige Links:

- *http://developers.facebook.com/docs/* – Einstiegsseite der offiziellen Plattform-Dokumentation von Facebook

1.2 Der Development Stack

Da alle von Facebook bereitgestellten Programmierschnittstellen grundsätzlich auf offenen Standards wie OAuth und HTTP-REST basieren, kann prinzipiell jeder zur Entwicklung von Webapplikationen geeignete Development Stack auch zur Entwicklung von Facebook-Anwendungen genutzt werden.

Development Stack

Unter *Development Stack* versteht man dabei den »Stapel« an Software, der notwendig ist, um eine Webapplikation zu entwickeln und zu betreiben. Dazu zählen also das Betriebssystem, mit dem der oder die Server betrieben werden (Linux, Windows ...), die Software für Webserver (meist Apache) und die Datenbank (MySQL, PostgreSQL ...) sowie die serverseitig eingesetzte Programmiersprachen (PHP, Ruby ...). Kommen zusätzlich zur Kernsprache weitere Applikations-Frameworks (CakePHP, ZendEngine, Ruby on Rails ...) oder clientseitige Bibliotheken (etwa JQuery, Scriptaculous ...) zum Einsatz, zählen diese natürlich ebenfalls zum Development Stack.

Während Facebook selbst *offizielle SDKs* für PHP, JavaScript, iOS und Android bereitstellt, gibt es für zahlreiche beliebte Programmiersprachen Bibliotheken und SDKs mit ähnlichem Funktionsumfang. Während also Facebook selbst als »plattform-

agnostisch« zu bezeichnen ist, genießen PHP- und JavaScript-Entwickler den großen Vorteil, von Facebook selbst immer mit den jeweils aktuellsten Updates versorgt zu werden – bei manch anderer Sprache müssen sie hier etwas länger warten oder als Entwickler selbst Hand anlegen. Da PHP aktuell nach wie vor die am weitesten verbreitete Programmiersprache in der Welt der Webapplikationen ist und darüber hinaus Anhänger von anderen Development Stacks oft zumindest grundlegende PHP-Kenntnisse besitzen, habe ich beschlossen, die Programmierbeispiele in diesem Buch größtenteils in PHP auszuführen. Dem Einsteiger, der sich noch nicht auf einen Development Stack festgelegt hat, empfehle ich ebenfalls, zumindest die ersten Schritte mit Facebook in PHP zu machen, da hier auch der im Web verfügbare Fundus an How-tos, Beispiel-Code und FAQs am größten ist. Außerdem werden die meisten Programmierbeispiele in diesem Buch nicht auf dem offiziellen PHP SDK basieren. Während das SDK den täglichen Umgang mit Facebook und vor allem der Graph API durch Abstraktion grundlegender Methoden wesentlich erleichtert, ist es vor allem für Einsteiger wichtig, die dahinterliegenden Mechanismen zu verstehen und auf einem niedrigeren Level selbst umzusetzen. Natürlich wird aber auch dem PHP SDK genügend Raum gewidmet.

Wichtige Links:

- *http://developers.facebook.com/docs/sdks/* – Überblick über die offiziellen Facebook-SDKs für JavaScript, PHP, iOS und Android
- *https://github.com/mmangino/facebooker2*, *https://github.com/nov/fb_graph* – zwei populäre Implementierungen der Graph API für Ruby on Rails
- *https://github.com/facebook/python-sdk/* – populäre Implementierung der Graph API für Python
- *http://code.google.com/p/facebook-java-api/* – populäre, leider aber nicht mehr aktuelle Implementierung der Graph API für Java

1.3 Hosting

Für Entwickler, die damit beginnen möchten, eigene Facebook-Anwendungen zu erstellen, ist es wesentlich, zu wissen, dass Facebook selbst grundsätzlich keine Hosting-Möglichkeiten für ihre Anwendungen zur Verfügung stellt. Als Applikationsentwickler sind Sie also selbst für die Bereitstellung von geeigneter Hard- und Software, Monitoring, Backup und Skalierung bei Benutzerwachstum zuständig.

Bei der Wahl einer geeigneten Hosting-Lösung für die eigene Facebook-Anwendung erlegt Facebook den Entwicklern seiner Plattform wenige Einschränkungen oder Anforderungen auf: Grundsätzlich ist jeder mietbare Webspace mit PHP-Unterstützung zum Hosting einfacher Facebook-Anwendungen geeignet. Ob der Webspace

dabei unter Windows oder einem Linux-basierten Betriebssystem betrieben wird, ist dabei ebenso unerheblich wie die eingesetzte Webserver-Software.

1.3.1 Leistungsfähigkeit und Skalierung

Wie bei jeder anderen Webapplikation auch müssen die Überlegungen zur idealen Wahl der Hosting-Umgebung vor allem auch auf die erwarteten Nutzerzahlen und Nutzungscharakteristika Rücksicht nehmen. Während eine Applikation mit wenigen hundert Benutzern, die darüber hinaus selten gleichzeitig auf die Applikation zugreifen werden, problemlos in einer kostengünstigen Shared-Hosting-Umgebung laufen kann, empfiehlt sich für größere Anwendungen rasch der Einsatz eines virtuellen oder physischen eigenen Servers (*VServer* oder *Dedicated Server*).

Agenturen oder Applikationshersteller, die planen, eine größere Anzahl an voneinander unabhängigen Facebook-Applikationen zu entwickeln, sei jedenfalls zum Einsatz eines dedizierten Servers geraten, also einer Maschine, die ausschließlich für die Applikationen der Agentur zur Verfügung steht und nicht mit anderen Kunden des Hosting-Providers geteilt werden muss. Für Applikationsentwicklungen, die eine Benutzerzahl in Millionenhöhe anstreben, sollte darüber hinaus ein skalierbares Serverkonzept entwickelt werden. Hierbei können etwa mehrere Server unterschiedliche Aufgaben übernehmen, z. B. wenn eigene Maschinen als Datenbankserver, Frontend-Webserver und Load-Balancer eingesetzt werden. Steigen die Benutzerzahlen an, kann so durch Hinzufügen weiterer Maschinen die maximale Kapazität gesteigert werden. Selbstverständlich sind auch Cloud-Hosting-Lösungen wie etwa der Elastic Cloud Service EC2 von Amazon oder App Engine von Google für das Hosting großer Facebook-Anwendungen geeignet.

Planung und Umsetzung solcher Hosting-Lösungen würden den Rahmen dieses Buches sprengen und erfolgen darüber hinaus im Wesentlichen analog zu herkömmlichen Webanwendungen.

Application Hosting auf Heroku

Seit September 2011 bietet Heroku (*http://www.heroku.com*) Cloud-Hosting von Facebook-Anwendungen in Partnerschaft mit Facebook an – direkt in die Developer-Benutzeroberfläche von Facebook integriert. Heroku unterstützt das Hosting von Facebook-Anwendungen, die in den Sprachen PHP, Node.js, Python oder Ruby geschrieben sind, und bietet zahlreiche Erweiterungen wie etwa PostgreSQL-Datenbanken, SMS-Gateways, Video-Encoding und mehr. Je nach Auslastung und Anzahl der möglichen parallelen Zugriffe auf die Anwendung, Größe der Datenbank sowie der Anzahl der Erweiterungen wird der monatliche Preis der Dienstleistung berechnet. Während das Hosting von populären Anwendungen dabei schnell mehrere hundert US-Dollar pro Monat kosten kann, ist der Minimalbetrieb sogar kostenfrei.

Für Neulinge im Cloud-Hosting mag die Verwendung von Heroku ungewohnt erscheinen – so wird der Code der eigenen Anwendung etwa nicht per FTP oder SCP auf den Server hochgeladen, sondern ausschließlich mittels der Software-Versionierung *Git*. Das Hosting mit Heroku wird in Anhang A, »Cloud-Hosting von Facebook-Anwendungen mit Heroku«, detailliert erklärt.

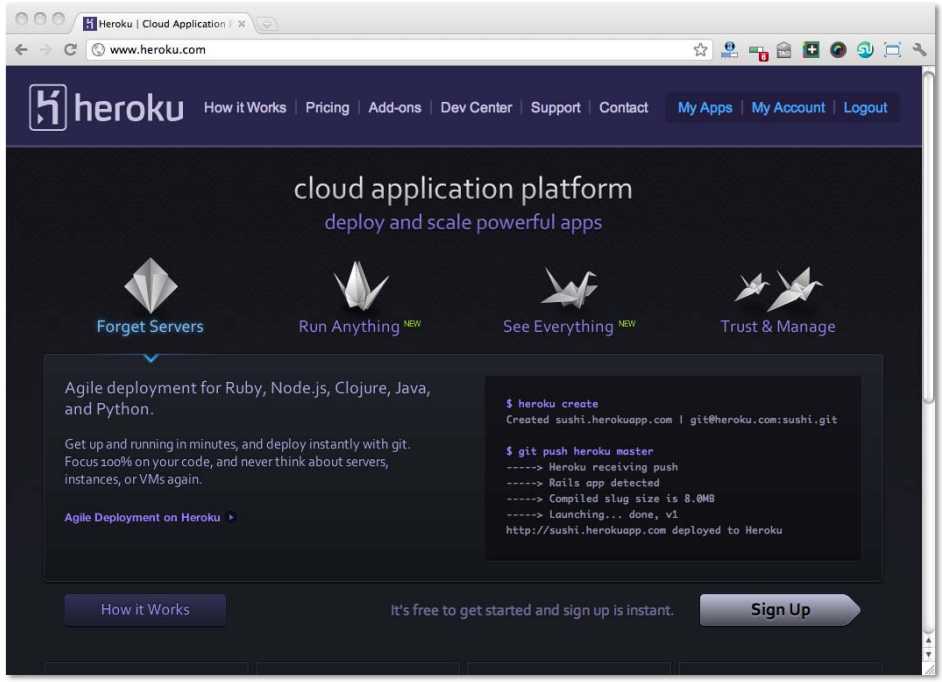


Abbildung 1.7 Cloud-Hosting von Facebook-Anwendungen mit Heroku.com

Als Grundlage für dieses Buch und als Empfehlung für Ihren Start in die Welt der Facebook-Anwendungsentwicklung gehe ich von einer üblichen Hosting-Umgebung auf einem virtuellen oder dedizierten Linux-basierten Serversystem aus. Als Webserver kommt hier üblicherweise Apache 2 zum Einsatz, MySQL übernimmt die Rolle des Datenbankservers. Eine aktuelle Version von PHP (5.x und höher) ist daneben die einzige Anforderung an unseren Development Stack.

1.3.2 SSL-Unterstützung

Eine kleine, aber umso wesentlichere Anforderung an das Hosting von Facebook-Applikationen gibt es aber doch: Facebook verlangt seit 1. Oktober 2011, dass alle Facebook-Anwendungen von Drittherstellern wahlweise auch über das SSL-Protokoll (*Secure Socket Layer*) ausgeliefert werden müssen.

Warum Verschlüsselung mit SSL?

Der Hintergrund für diese Anforderung ist nachvollziehbar: Informationen, die zwischen Browser und Webserver über das unverschlüsselte HTTP-Protokoll (*Hypertext Transfer Protocol*) übertragen werden, können grundsätzlich von Angreifern sehr einfach abgefangen werden. So können etwa Angreifer, die im selben lokalen WiFi-Netzwerk wie ihr Opfer surfen, auf einfachstem Wege Kennwörter und sonstige übertragenen Daten einer Websession mitlesen. So schlug 2010 etwa die Veröffentlichung des Firefox-Plugins *Firesheep* hohe Wellen, das es auch Laien ermöglichte, Kennwörter von Facebook, Twitter etc. von anderen »Mitsurfern« zu erspähen. Nicht zuletzt seit Firesheep hat sich auch herumgesprochen, wie man diese frappante Sicherheitslücke stopfen kann: Mit Verwendung des SSL-Protokolls werden Daten zwischen Browser und Webserver verschlüsselt übertragen, was das einfache »Mitlesen« von Kennwörtern oder Cookies effektiv unterbindet. War diese am Protokollkürzel *https* erkennbare Übertragungsart bisher vor allem dem E-Commerce und Tele-Banking-Bereich vorbehalten, bieten mittlerweile auch die meisten populären Social-Networking-Anbieter wie Twitter und Facebook dieses Protokoll zumindest wahlweise an. Da aber jede Sicherheitsmaßnahme nur so wirksam ist wie das schwächste Glied der jeweiligen Applikationskette, ist Facebook dazu übergegangen, auch seine Applikationsentwickler zur Unterstützung von SSL zu verpflichten. Während dies gerade für den Hobby-Entwickler eine zusätzliche Hürde darstellt, ist es doch aus Sicht der Benutzersicherheit ein begrüßenswerter Schritt.

Was bedeutet die Pflicht zur SSL-Unterstützung nun konkret für Sie als Entwickler von Facebook-Anwendungen? Nun, die Domain und der Webserver, unter dem Ihre Facebook-Anwendung bzw. der jeweilige Canvas-Inhalt abgerufen wird, müssen über ein gültiges SSL-Zertifikat verfügen. Wurde die Applikation etwa bisher über *http://apps.mycompany.com* abgerufen, muss dies nun auch über *https://apps.mycompany.com* möglich sein. Bei der Wahl eines Hosting-Angebots müssen Sie also unbedingt darauf achten, dass der jeweilige Anbieter die Installation eines SSL-Zertifikats unterstützt. Entwickler, die sich nicht näher mit der Installation des Zertifikats am Webserver auseinandersetzen möchten, wählen idealerweise ein Hosting-Angebot, das SSL-Zertifikate inklusive Installation anbietet. Hobby-Entwickler, die die jährlichen Kosten eines SSL-Zertifikats von rund 100 US-Dollar pro Domain scheuen, können zumindest für Testzwecke auch ein selbst erzeugtes Zertifikat verwenden, müssen dann aber mit entsprechenden Sicherheitswarnungen im Browser des Benutzers leben.

Wenn Sie als Agentur oder Hersteller planen, eine größere Anzahl an Facebook-Anwendungen zu entwickeln und auf einem Serversystem zu betreiben, empfiehlt sich hinsichtlich des Einsatzes von SSL eine der folgenden Vorgehensweisen:

1. Erwerben Sie ein SSL-Zertifikat für den von Ihnen gewählten Hostnamen, unter dem künftig alle Facebook-Applikationen laufen werden, z. B. *apps.mycompany.com*. Erstellen Sie im Root-Verzeichnis Ihres Webservers (auf üblichen Apache-Installationen etwa */var/www/*) Unterverzeichnisse für jede Applikation, die Sie betreiben möchten. Diese sind dann unter den URLs *https://apps.mycompany.com/app1*, *https://apps.mycompany.com/app2* etc. abrufbar. Der Vorteil dieser Strategie: Es muss nur ein SSL-Zertifikat für die Domain erworben und installiert werden – damit ist diese Vorgehensweise vor allem für Einsteiger zu empfehlen.
2. Erwerben Sie ein sogenanntes *Wildcard-Zertifikat* für Ihre Domain **.mycompany.com*. Ein Wildcard-Zertifikat ist zwar etwas teurer als ein herkömmliches Zertifikat, erlaubt aber die Verschlüsselung von beliebig vielen Sub-Domains, etwa *https://app1.mycompany.com* oder *https://app2.mycompany.com*. Das SSL-Protokoll erfordert leider, dass Sie gleichzeitig eine eigene IP-Adresse pro eingesetzter Sub-Domain bereitstellen. Die meisten Hosting-Angebote erlauben die Verwendung von mehreren IP-Adressen nur zu zusätzlichen Kosten, darüber hinaus ist diese Konfiguration komplizierter einzurichten. Der Vorteil dieser Vorgehensweise liegt in der saubereren Trennung unterschiedlicher Applikationen am Webserver, etwa in Form von separierten Log-Dateien und Dokumentenverzeichnissen.
3. Planen Sie als Agentur, Anwendungen für Ihre Kunden zu entwickeln, kann es wünschenswert sein, Ihre eigene Domain vor dem Endbenutzer zu verbergen. In diesem Fall müssen Ihre Kunden eine Sub-Domain mit passendem SSL-Zertifikat bereitstellen, etwa *https://apps.myclient1.com* und *https://apps.myclient2.com*. Während diese Vorgehensweise eine auch nach außen hin saubere Trennung Ihrer Kundenapplikationen erlaubt, ist sie auch die kostspieligste und in der Einrichtung komplexeste, da Ihre Kunden dazu auch Anpassungen an ihrer DNS-Konfiguration vornehmen müssen. Um etwa die Domain *apps.myclient1.com* auf ihren eigenen Application Server verweisen zu lassen, muss die IT-Abteilung Ihres Kunden die entsprechende Einstellung in der DNS-Konfiguration vornehmen (DNS: *Domain Name System*).

   Achtung: Besonders wichtig ist die Verwendung von separaten Domains oder Sub-Domains pro Applikation dann, wenn Sie planen, Facebook Social Plugins einzusetzen. Ein Beispiel dafür ist etwa die RECOMMENDED-BOX, die Benutzern personalisierte Empfehlungen zu Inhalten Ihrer Applikation oder Website liefert. Da speziell dieses Social Plugin immer nur auf Domain-Ebene funktioniert und etwaige Unterverzeichnisse in der Appikations-URL ignoriert, würden die Empfehlungen von verschiedenen Applikationen vermischt angezeigt werden, wenn Sie sich für die einfachste Vorgehensweise entscheiden!

Hosting-Umgebung für Einsteiger

Für die Programmierbeispiele in diesem Buch gehe ich von einer einfachen Hosting-Umgebung aus, wie sie die meisten kostengünstigen Miet-Webspaces bieten. Neben dem Webserver (üblicherweise Apache 2) benötigen Sie eine MySQL-Datenbank (mindestens Version 5.x) und die Programmiersprache PHP (mindestens Version 5.2) als aktiviertes Apache-Modul. Um Anwendungen zu entwickeln, müssen Sie darüber hinaus eine beliebige Domain oder Sub-Domain (etwa *apps.mycompany.com*) mitsamt dem bereits erwähnten SSL-Zertifikat auf Ihrem Webserver einrichten. Wenn Sie Ihren Development Stack nicht selbst konfigurieren möchten, empfiehlt es sich, Domain, Webspace und Zertifikat als Paket von einem Hosting-Anbieter zu erwerben – in diesem Fall sollte die Konfiguration bereits vom Anbieter vorgenommen werden. Webspace mit den beschriebenen Eigenschaften können Sie heute bereits für fünf bis zehn EUR pro Monat mieten, etwa beim deutschen Anbieter Hetzner (*http://www.hetzner.de/hosting/produkte_webspace/level4*) oder dem US-Hoster Go Daddy (*http://www.godaddy.com/hosting/web-hosting.aspx?ci=9009*).

Alle Dateien Ihrer Anwendung, also den PHP-Code, Bilder, JavaScript- und CSS-Dateien, laden Sie üblicherweise per FTP- oder SCP in das Root-Verzeichnisses Ihres Webservers. Ein beliebtes, kostenloses Tool zur Datenübertragung ist z. B. FileZilla (*http://filezilla-project.org/*). Für alle folgenden Beispiele gilt die Annahme, dass sich das Root-Verzeichnis des Webservers in */var/www* befindet.

Der Zugriff auf die MySQL-Datenbank erfolgt meistens über ein Web Interface (phpMyAdmin o. Ä.), das entweder Sie selbst oder Ihr Provider einrichten. Alternativ können Sie die Datenbank auch über die Kommandozeile oder die MySQL GUI Tools (*http://dev.mysql.com/downloads/gui-tools/5.0.html*) bedienen. Zum Zugriff auf die Datenbank ist eine Reihe an Parametern notwendig, für die die folgenden üblichen Standardwerte angenommen werden können:

Hostname und Port des Datenbankservers: `localhost:3306`
Benutzername: `myapp`
Kennwort: `mypassword`
Datenbankname: `mydatabase`

Die notwendigen Schritte zur Inbetriebnahme von Webspace und Datenbank können je nach Hosting-Anbieter sehr unterschiedlich sein und daher in diesem Buch nicht detailliert berücksichtigt werden. Wenden Sie sich bei Problemen an den Support Ihres Providers, oder ziehen Sie dessen Online-Dokumentation zu Rate.

Lokaler Webserver mit XAMPP

Alternativ können Sie die meisten Programmierbeispiele im Buch auch mit einer lokal installierten Hosting-Umgebung am eigenen Rechner ausprobieren. Ausnahmen bestehen aber z. B. in der Verwendung von Open Graph und Social Plugins, da diese Technologien voraussetzen, dass die Server von Facebook selbst Zugriff auf

Ihren Anwendungsserver haben. Eine einfache lokale Hosting-Umgebung kann etwa mit dem Open-Source-Paket *XAMPP* heruntergeladen und auf Windows, Mac OS und Linux installiert werden. XAMPP beinhaltet den Webserver Apache 2, die Datenbank MySQL und natürlich die Programmiersprache PHP. Nach der Installation werden Datenbank und Webserver wahlweise automatisch als Systemservice geladen oder manuell über das Control Panel von XAMPP gestartet. Letzteres ist sinnvoll, um die Ressourcen des eigenen Rechners zu schonen, wenn Sie gerade keine Anwendungen entwickeln.

Wichtig für die Einrichtung der lokalen Entwicklungsumgebung ist die Konfiguration einer Domain oder Sub-Domain, die auf den eigenen Rechner verweist. Üblicherweise wird dazu keine echte Domain registriert, sondern eine beliebig gewählte Domain in der lokalen Hosts-Datei angelegt. Die Hosts-Datei finden Sie unter Windows in *c:\windows\system32\drivers\etc\hosts*, unter Mac OS und Linux in */etc/hosts*. In dieser Datei müssen Sie folgenden Eintrag am Dateiende ergänzen:

```
127.0.0.1    apps.mycompany.com
```

Dieser Eintrag bewirkt, dass alle Zugriffe auf den Hostnamen `apps.mycompany.com` auf den lokalen Rechner, der durch die sogenannte *Loopback-Adresse* `127.0.0.1` repräsentiert wird, umgeleitet werden. Wenn Sie also z. B. den Hostnamen in Ihrem Browser aufrufen, wird der lokal installierte Webserver antworten – perfekt für die lokale Anwendungsentwicklung!

Bei der Verwendung von XAMPP laden Sie die Dateien Ihrer Anwendung (PHP-Code, Bilder etc.) direkt in das lokale Root-Verzeichnis des Webservers, das sich unter Windows üblicherweise in *c:\xampp\htdocs*, unter Mac OS in */Applications/xampp/htdocs* befindet. Auch hier kann der Datenbankzugriff über phpMyAdmin, die MySQL Client Tools oder den Kommandozeilen-Client erfolgen, der sich unter Windows in *c:\xampp\mysql\bin\mysql* und unter Mac OS in */Applications/xampp/mysql/bin/mysql* befindet.

Wichtige Links:

- *http://www.digicert.com/help/* – Tool zum Testen des eigenen SSL-Zertifikats auf korrekte Installation
- *http://www.whichssl.com/comparisons/price.html* – Preisvergleich zwischen Anbietern von SSL-Zertifikaten
- *http://www.debian-administration.org/articles/349/* – How-to SSL-Zertifikat am Apache-Webserver installieren
- *http://www.debian-administration.org/articles/284* – How-to SSL-Zertifikate selbst erzeugen
- *http://www.apachefriends.org/en/xampp.html* – XAMPP

1.4 Setup der ersten eigenen Applikation

Um Ihre erste eigene Facebook-Applikation zu erstellen, müssen Sie zuerst die Facebook-Entwickler-App aufrufen und autorisieren. Die oft auch schlicht *Developer* genannte Anwendung finden Sie unter der URL *http://www.facebook.com/developer*. Beim ersten Aufruf muss die Entwickler-App – wie jede andere Facebook-Anwendung auch – erst vom Benutzer autorisiert werden. Der dazu angezeigte Standarddialog muss vom Benutzer für jede neue Applikation durchlaufen werden. Der Dialog zeigt detailliert an, auf welche Profilinformationen die Anwendung nach der Autorisierung zugreifen darf. Im Falle der Entwickler-App sind dies lediglich die Basisinformationen eines Profils.

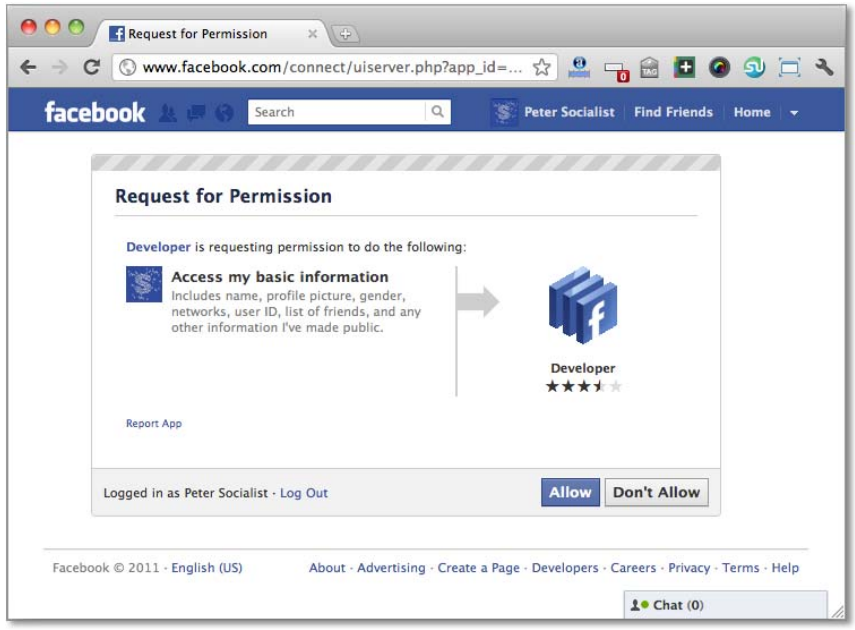


Abbildung 1.8 Autorisierung der Facebook-Entwickler-App

Nach der Autorisierung der Entwickler-App teilt sich der Bildschirm in die – momentan noch leere – Liste Ihrer eigenen Anwendungen sowie einige nützliche Direkt-Links zur Dokumentation der Facebook-Plattform.

Mit dem Klick auf CREATE NEW APP geht es los: Im ersten Schritt legen Sie den frei wählbaren, nicht notwendigerweise eindeutigen Namen der Facebook-Applikation fest. Bevor Sie die erste eigene Anwendung anlegen können, müssen Sie aber noch die allgemeinen Geschäftsbedingungen und die Platform Policies von Facebook akzeptieren und einen einfachen Captcha-Test bestehen, der das automatisierte Anlegen von Anwendungen durch Skripte verhindern soll – dann ist Ihre erste Facebook-Applikation auch schon angelegt!

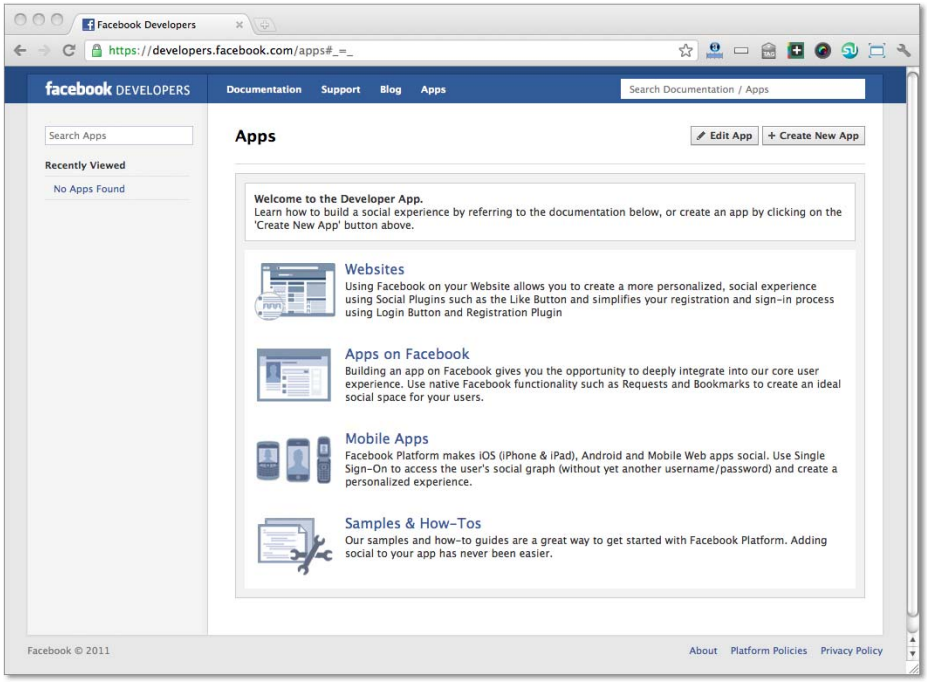


Abbildung 1.9 Startbildschirm der Facebook-Entwickler-App

Abbildung 1.10 Dialog zum Anlegen einer neuen Facebook-Anwendung

Unter Umständen erhalten Sie nach Bestätigung des Captchas folgende Fehlermeldung: *Your account must be verified before you can take this action. Please verify your account by adding your mobile phone or credit card.*

Was bedeutet das? Um den Missbrauch von Facebook einzuschränken, etwa indem Anwendungen die Privatsphäre oder den Datenschutz der Benutzer verletzten, verlangt Facebook, dass alle Facebook-Entwickler mit ihrer Kreditkarte oder einem Mobiltelefon identifiziert sind. In der Fehlermeldung werden Links zu beiden Verifikationsmöglichkeiten angeboten. Während die Verifikation per Mobiltelefon lediglich die Eingabe eines per SMS an die eigene Nummer gesendeten Bestätigungscodes erfordert, müssen Sie für die Freischaltung per Kreditkarte eine gültige Kreditkarten-

nummer eingeben. Wenn Sie dies nicht ohnehin zur Buchung von Werbung auf Facebook tun, empfiehlt sich die einfachere Verifikation per SMS. Nach der erfolgreichen Freischaltung Ihres Kontos kann es dann mit der ersten eigenen Anwendung aber wirklich losgehen!

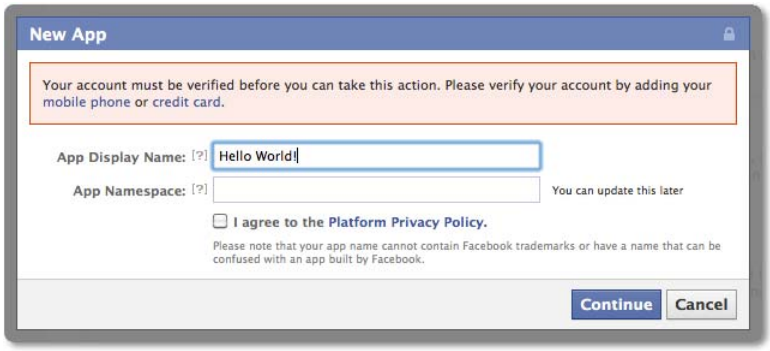


Abbildung 1.11 Fehlermeldung beim Anlegen einer Facebook-Anwendung

App Display Name und App Namespace

Wie bereits erwähnt, kann der *App Display Name* oder *Anwendungsname* prinzipiell frei gewählt werden. Facebook verwendet den Anwendungsnamen, um die Applikation gegenüber dem Benutzer erkennbar zu machen – etwa bei der Autorisierung der Applikation, bei Wall-Postings oder Einladungen zur Anwendung.

Der Anwendungsname muss nicht eindeutig sein, kann Sonderzeichen und Leerzeichen enthalten und darf eine Länge von 50 Zeichen nicht überschreiten. *Achtung:* Manche reservierten Wörter wie *Wall* oder *Facebook* können nicht Teil des Anwendungsnamens sein.

Der *App Namespace* ist ein Facebook-weit eindeutiger Bezeichner, der z. B. genutzt wird, um eine Applikation im Browser nach dem URL-Schema *http://apps.facebook.com/AppNamespace* aufzurufen. Der Namespace darf ausschließlich Buchstaben und keine Ziffern, Leer- oder Sonderzeichen enthalten und darf eine Länge von 20 Zeichen nicht überschreiten. Bestimmte Arten von Facebook-Anwendungen benötigen keinen Namespace, etwa iOS-/Android-Apps oder externe Websites, die lediglich Social Plugins wie den Like-Button einbinden.

Im folgenden Dialog können alle relevanten Eigenschaften der Facebook-Anwendungen zentral festgelegt werden. Dabei spielt es keine Rolle, ob es sich um eine Canvas-Anwendung, eine Tab-Anwendung, eine externe Webapplikation oder eine native mobile Anwendung für iOS oder Android handelt – alle Einstellungen sind hier gebündelt zu finden.

1.4.1 Einstellungen »Basic«

Der Abschnitt SETTINGS gliedert sich in die Sektionen BASIC, AUTH DIALOG und ADVANCED. Übrigens: Da Facebook gerade die Entwickler-App derzeit nur teilweise in die deutsche Sprache übersetzt hat, bezieht sich dieses Buch durchgehend auf die englischsprachige Benutzeroberfläche. Für Entwickler ist es sinnvoll, Ihre Standardsprache auf ENGLISCH zu setzen, da auch die meisten Blog-Beiträge und News sich auf die englischen Bezeichnungen beziehen.

Unter BASIC werden zuerst wichtige Eckdaten der Anwendung, wie der soeben gewählte freie APP DISPLAY NAME, die plattformweit eindeutige, numerische APP ID und das sogenannte APP SECRET, zusammengefasst. Das App Secret dient Facebook dazu, sicherzustellen, von welcher Applikation ein Zugriff auf die Facebook-Plattform tatsächlich erfolgt ist. Da das Secret nur Ihnen als Applikationsentwickler (und Facebook selbst) bekannt ist, verhindert es den Missbrauch der eigenen Applikation durch etwaige Angreifer.

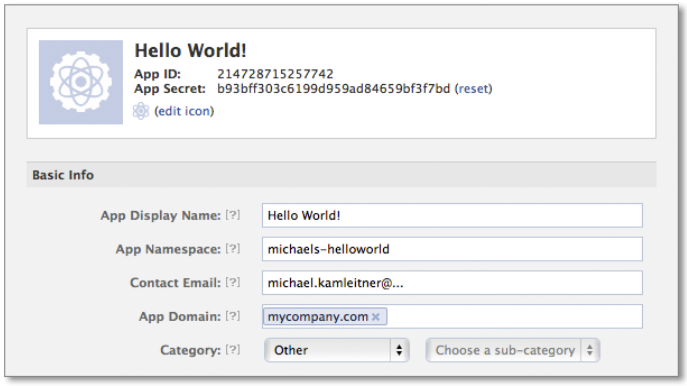


Abbildung 1.12 Basisinformationen einer Facebook-Anwendung

Numerische IDs von Applikationen, Benutzern, Seiten und mehr

Facebook verwendet ein durchgehendes Identifikationssystem für alle auf der Plattform gespeicherten Objekte, wie etwa Benutzer, Seiten, Gruppen, Postings und auch Applikationen. Jedes Objekt wird über eine – über alle Objekttypen hinweg – eindeutige numerische Identifikationsnummer adressiert.

Achtung: Aufgrund der großen Anzahl an Objekten, die auf der Facebook-Plattform gespeichert werden, können die Identifikationsnummern sehr groß sein. Zur Speicherung – etwa von Benutzer-IDs in der Datenbank Ihrer Applikation – muss dabei ein mindestens 20-stelliger Big-Integer-Wert vorgesehen werden. In MySQL-Datenbanken werden Facebook-IDs also üblicherweise folgendermaßen gespeichert:

```
USER_ID BIGINT(20) NOT NULL
```

Hinweis: Möchten Sie die Benutzer-ID Ihres eigenen Facebook-Kontos herausfinden? Nichts leichter als das, steuern Sie über den Kopfbereich von Facebook unter Profile Ihr persönliches Profil an. Die daraufhin geladene URL der Form *http://www.facebook.com/profile.php?id=1234* gibt Aufschluss über Ihre Benutzer-ID, die im GET-Parameter `?id=` codiert ist. Vielleicht haben Sie Ihre Profil-URL aber bereits individualisiert und sind etwa unter *http://www.facebook.com/michael.kamleitner* zu finden. In diesem Fall klicken Sie doch einfach auf Ihr eigenes Profilbild! Sie gelangen auf eine Seite mit einer URL der Form *http://www.facebook.com/photo.php?fbid=...&set=a .483848100863.268103.609190863&type=1&theater* – auch hier finden Sie Ihre Benutzer-ID codiert im Parameter `set` am Ende der URL wieder. Auf diesem Weg können Sie übrigens auch die IDs von anderen Benutzern und Seiten herausfinden.

Allgemeine Informationen:

Unter Basic Info finden Sie folgende Einstellungen:

- Applikationslogo und -icon: Diese beiden grafischen Elemente erleichtern Benutzern das Wiedererkennen einer Anwendung. Als Logo sollte ein 75 × 75 Pixel, als Icon ein 16 × 16 Pixel großes Bild in einem der üblichen Grafikformate (PNG, GIF, JPG) hochgeladen werden.
- App Display Name: Der bereits beim Anlegen gewählte Name der Applikation. Unter diesem Namen wird die Applikation gegenüber dem Benutzer innerhalb von Facebook präsentiert, etwa in Dialogen zur Installation der Anwendung oder in Wall-Postings. Da die allgemeine Suchfunktion von Facebook mitunter auch Applikationen zu Tage fördert, lohnt es sich, den Namen der Applikation auch in dieser Hinsicht zu optimieren. Für unsere erste eigene Anwendung wählen Sie `Hello World!` als Titel.
- App Namespace: Der eindeutige Bezeichner einer Anwendung. Im Wesentlichen ist der Namespace mit dem unter *http://facebook.com/username* frei wählbaren Benutzernamen zu vergleichen. Der wichtigste Zweck des App Namespace ist die Adressierung von Canvas-Anwendungen unter der URL *http://apps.facebook.com/AppNamespace*. Für unsere Hello-World-Anwendung wählen Sie den Namespace `ihrname-helloworld` oder ähnlich.
- Contact Email: Die primäre Kontaktadresse des Anwendungsentwicklers. An diese E-Mail-Adresse wird einerseits Benutzer-Feedback geleitet, andererseits dient sie dem Facebook-Plattform-Team als Kontaktadresse, wenn es etwa Probleme mit der Anwendung geben sollte. Es ist daher ratsam, hier eine E-Mail-Adresse anzugeben, die vom verantwortlichen Entwickler oder seinem Team regelmäßig abgerufen wird.

- App Domain: Facebook-Anwendungen, die als externe Website oder -applikation realisiert werden, müssen über die Domain der externen Website mit der Facebook-Plattform verknüpft werden. Soll die Anwendung etwa auf *http://apps.mycompany.com* laufen, würde die Angabe `apps.mycompany.com` die entsprechende Verknüpfung sicherstellen. Soll die Anwendung unter mehreren Sub-Domains, etwa *http://moreapps.mycomapany.com* etc., laufen, empfiehlt sich eine Wildcard-Angabe mittels `mycompany.com` (allerdings ohne »*.« o. Ä.).
- Category: Diese Auswahl erlaubt eine Kategorisierung Ihrer Anwendung. Die vorgenommene Kategorisierung wird dem Benutzer etwa auf dem Applikationsprofil der Anwendung angezeigt, hat ansonsten aber kaum Bedeutung, seitdem Facebook das eigene Applikationsverzeichnis eingestellt hat.

Cloud Services:

An dieser Stelle können Sie die bereits erwähnten Cloud-Hosting-Services von Heroku und künftig auch anderen Hosting-Partner von Facebook aktivieren.

Anwendungsintegration:

Die folgenden vier Abschnitte unter Select how your app integrates with Facebook widmen sich den unterschiedlichen Arten von Plattform-Anwendungen, wie sie in Abschnitt 1.1, »Überblick über die Facebook-Plattform«, vorgestellt wurden:

Unter Website finden Sie Optionen, die für den OAuth-Ablauf auf einer Anwendung wesentlich sind.

- Site URL: Die primäre URL, unter der die Anwendung abrufbar ist. Facebook leitet den Benutzer im OAuth-Ablauf mittels HTTP-Redirect von Facebook.com auf die URL der Anwendung um. Aus Sicherheitsgründen erfolgen Redirects nur auf die URL, die unter der Einstellung Site URL angegeben wurde. In unserer Beispielanwendung geben Sie für Site URL den Wert `http://apps.mycompany.com` ein, da dies die URL ist, unter der unser Anwendungsserver adressiert wird.

Unter App on Facebook finden Sie Einstellungen für Applikationen, die als Canvas-Anwendung oder Tab/Reiter auf Facebook ausgeführt werden sollen.

- Canvas URL: Jene URL, die den App-iFrame einer Canvas-Anwendung ausliefert, also etwa *http://apps.mycompany.com/helloworld/*. Wichtig ist dabei, dass die angegebene URL mit einem Schrägstrich oder einem dynamischen HTTP-Parameter der Form `?param=...` enden muss. URLs, die etwa mit einem PHP-Skript enden, z. B. *http://apps.mycpompany.com/helloworld.php*, werden nicht unterstützt, aus diesem Grund müssen Sie für die erste Anwendung ein Unterverzeichnis namens *helloworld* erstellen und dort die Datei *index.php* hinterlegen.

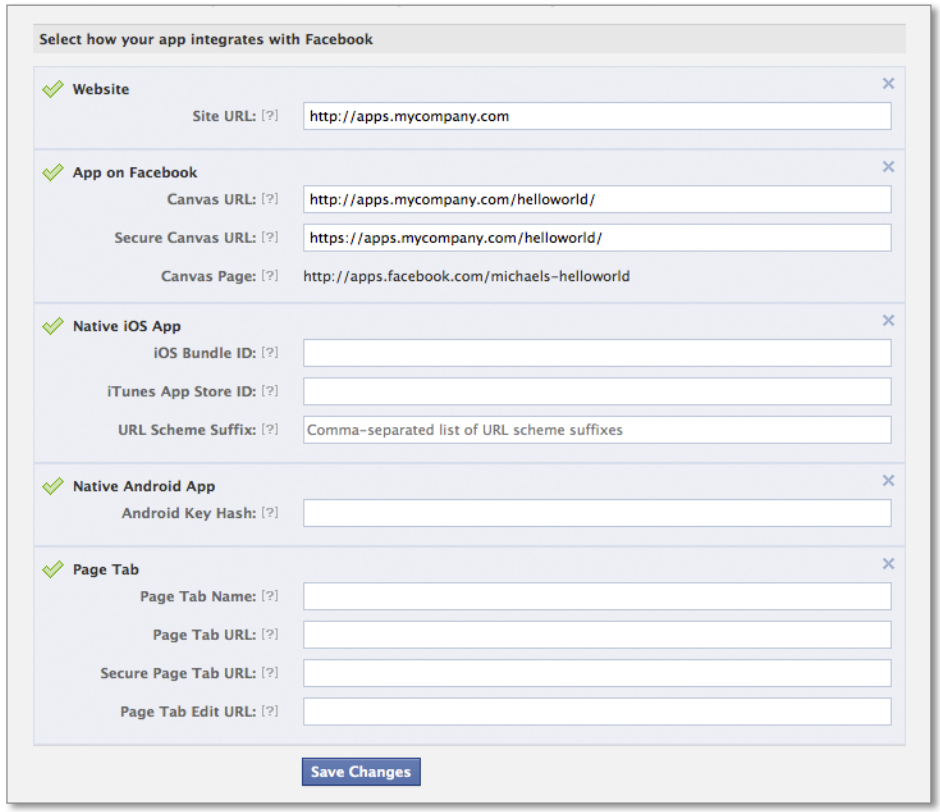


Abbildung 1.13 Einstellungsdialog zur Anwendungsintegration

- Secure Canvas URL: Jene URL, die den App-iFrame einer Canvas-Anwendung bei Ansteuerung über das verschlüsselte HTTPS-Protokoll ausliefert. Meist entspricht die Secure Canvas URL der Canvas URL, lediglich das verwendete Protokoll wird zu *https*. In unserem Fall also: *https://apps.mycompany.com/helloworld/*.

Unter Page Tab finden Sie Einstellungen für Applikationen, die als Tab-Anwendung auf Facebook ausgeführt werden sollen. Für Ihre erste eigene Anwendung verzichten Sie vorerst auf den Betrieb als Tab/Reiter.

- Page Tab Name: Der frei wählbare Name, der als Beschriftung des Tabs verwendet werden soll. Seitenadministratoren können in den Seiteneinstellungen später eine eigene Tab-Beschriftung definieren.
- Page Tab URL: Jene URL, die den App-iFrame einer Tab-Anwendung ausliefert.

- Secure Canvas URL: Jene URL, die den App-iFrame einer Tab-Anwendung bei Ansteuerung über das verschlüsselte HTTPS-Protokoll ausliefert.
- Page Tab Edit URL: Wenn Seitenadministratoren in den Seiteneinstellungen auf den Settings-Link einer Tab-Applikation klicken, werden sie zu dieser URL weitergeleitet. Üblicherweise wird es sich hierbei um einen seitenspezifischen Einstellungsdialog handeln, der z. B. als Unterseite auf *http://apps.facebook.com/AppNamespace/settings/SeitenID* bereitgestellt wird.
- Page Tab Width: Erlaubt bis 31. März 2012 den Betrieb eines Tabs/Reiters im alten – schmaleren – Format mit 520 Pixeln Breite.

Unter Native iOS App finden sich Einstellungen für Applikationen, die als native Anwendungen auf iOS-Geräten ausgeführt werden sollen.

- iOS Bundle ID: Die interne Bundle-ID einer iOS-Applikation wird aus Sicherheitsgründen bei Callbacks durch die Facebook-Plattform überprüft und muss daher hier eingetragen werden.
- iTunes App Store ID: Die Anwendungs-ID im iTunes App Store wird verwendet, um die Anwendung in iTunes direkt von Facebook.com zu verlinken.
- URL scheme suffix: Diese Einstellung erlaubt es, eine Applikations-ID in mehreren Anwendungen zu verwenden.

Unter Native Android App finden Sie Einstellungen für Applikationen, die als native Anwendungen auf Android-Geräten ausgeführt werden sollen.

- Android Key Hash: Der interne Key Hash einer Android-Applikation wird aus Sicherheitsgründen bei Callbacks durch die Facebook-Plattform überprüft und muss daher hier eingetragen werden.

1.4.2 Einstellungen zum »Auth Dialog«

Im Abschnitt Auth Dialog können Sie als Anwendungsentwickler detaillierte Einstellungen zum Autorisierungsdialog Ihrer Applikation vornehmen. Wie bereits erwähnt, müssen Benutzer diesen Dialog durchlaufen, bevor eine Anwendung auf die Profildaten des Benutzers zugreifen kann. Diesen Vorgang bezeichnet man meist auch als *Installation* oder *Autorisierung* der Anwendung durch den Benutzer. Für Ihre erste Anwendung, die keine Benutzerautorisierung vorsieht, müssen Sie in diesem Abschnitt vorerst keine Einstellungen vornehmen. Die Erklärung der einzelnen Optionen folgt in Kapitel 2, »Authentifikation und Autorisierung«.

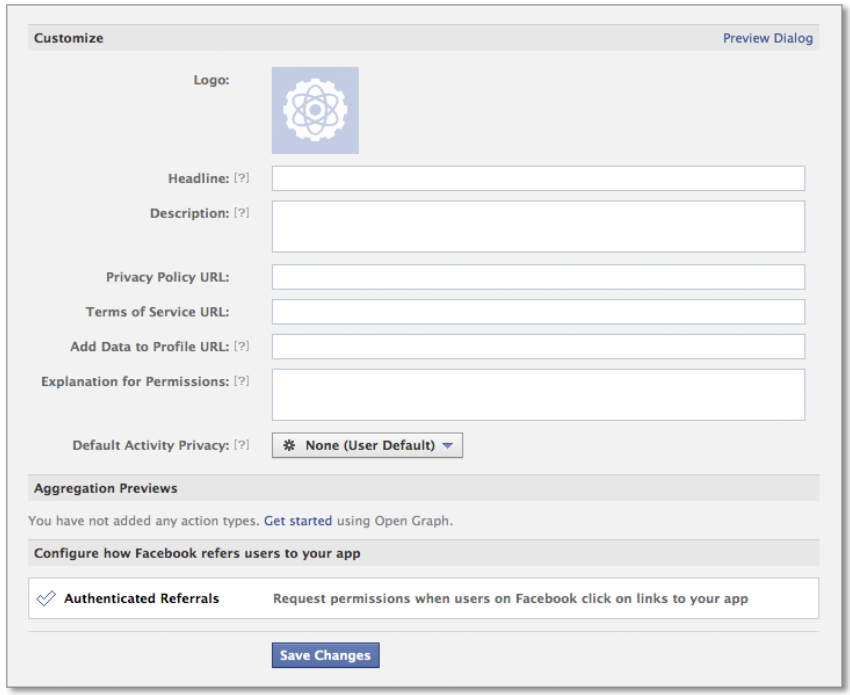


Abbildung 1.14 Einstellungen zum Autorisierungsdialog Ihrer Anwendung

1.4.3 Einstellungen »Advanced«

Die Einstellungen der folgenden »fortgeschrittenen« Abschnitte müssen Sie für Ihre erste eigene Facebook-Anwendung noch nicht berücksichtigen.

Authentifikation

Im Abschnitt AUTHENTICATION finden Sie verschiedene Einstellungen, die die Authentifikation und Autorisierung von Anwendungen durch den Benutzer betreffen.

- APP TYPE: Hier wird eingestellt, ob es sich bei der Anwendung um eine Webapplikation oder eine native mobile oder Desktop-Anwendung handelt. Die Einstellung hat derzeit keine bekannten Auswirkungen.
- DEAUTHORIZE CALLBACK: Diese URL wird von Facebook im Hintergrund aufgerufen, wenn ein Benutzer zu einem späteren Zeitpunkt eine Anwendung über seine Privatsphäre-Einstellungen deinstalliert. Dies kann hilfreich oder gar notwendig sein, um etwa am Applications Server den bestehenden Benutzerdatensatz zu löschen oder als gelöscht zu markieren – ohne Callback würde Ihre Anwendung keine Kenntnis von der Deinstallation durch den Benutzer erhalten. Um die Privatsphäre eines Benutzers zu wahren, ist dies aber meist nötig.

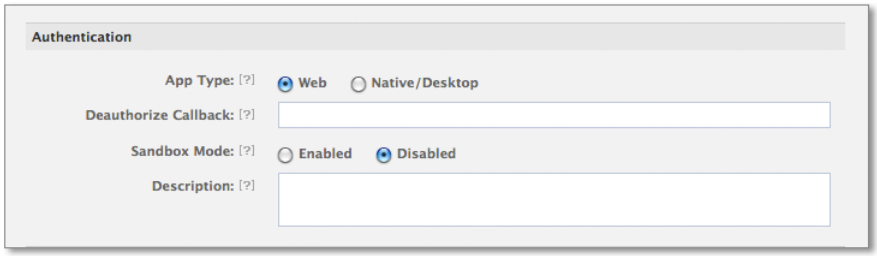


Abbildung 1.15 Einstellungen zur Authentifikation einer Anwendung

Eine Facebook-Anwendung deinstallieren

Haben Sie sich auch schon mal gefragt, wie man als Benutzer eine bereits installierte Facebook-Anwendung wieder deinstalliert? Durch die häufigen Änderungen an den Dialogen zu Konto- und Privatsphäre-Einstellungen, macht es Facebook seinen Benutzern nicht gerade leicht, die entsprechenden Dialoge zu finden. Gerade für Anwendungsentwickler ist es aber oft wichtig, eine in Entwicklung befindliche Anwendung wieder zu deinstallieren, um etwa den Installationsprozess testen zu können. Und so geht's:

- Im Kopfbereich von Facebook.com klicken Sie rechts oben auf den Pfeil neben Home.
- Wählen Sie im aufklappenden Menü Privacy Settings.
- Klicken Sie nun im Abschnitt Apps and Websites auf Edit Settings.
- Klicken Sie als Nächstes im Abschnitt Apps you use wieder auf Edit Settings.
- Sie bekommen nun eine Liste all jener Facebook-Applikationen angezeigt, die Sie in der Vergangenheit autorisiert haben. Über den mit x markierten Link können Sie eine Anwendung ganz einfach entfernen – die Anwendung hat dann über die Facebook-Schnittstellen keinen Zugriff mehr auf Ihre Profildaten.
- Sie können übrigens mit einem Klick auf Edit für jede Anwendung festlegen, auf welche Ihrer Profilinformationen die Anwendung genau zugreifen kann.

- Sandbox Mode: Diese Einstellung ist besonders wichtig: Um eine Anwendung während der Entwicklungs- und Testphase vor der Öffentlichkeit verborgen zu halten, muss der sogenannte Sandbox Mode aktiviert werden. Die Anwendung ist dann nur für unter Roles eingetragene Entwickler, Administratoren und Tester sichtbar und benutzbar.
- Description: In diesem Eingabefeld sollten Sie ein kurze, erklärende Beschreibung der Anwendung hinterlegen. Diese wird von Facebook etwa in Newsfeed-Stories angezeigt.

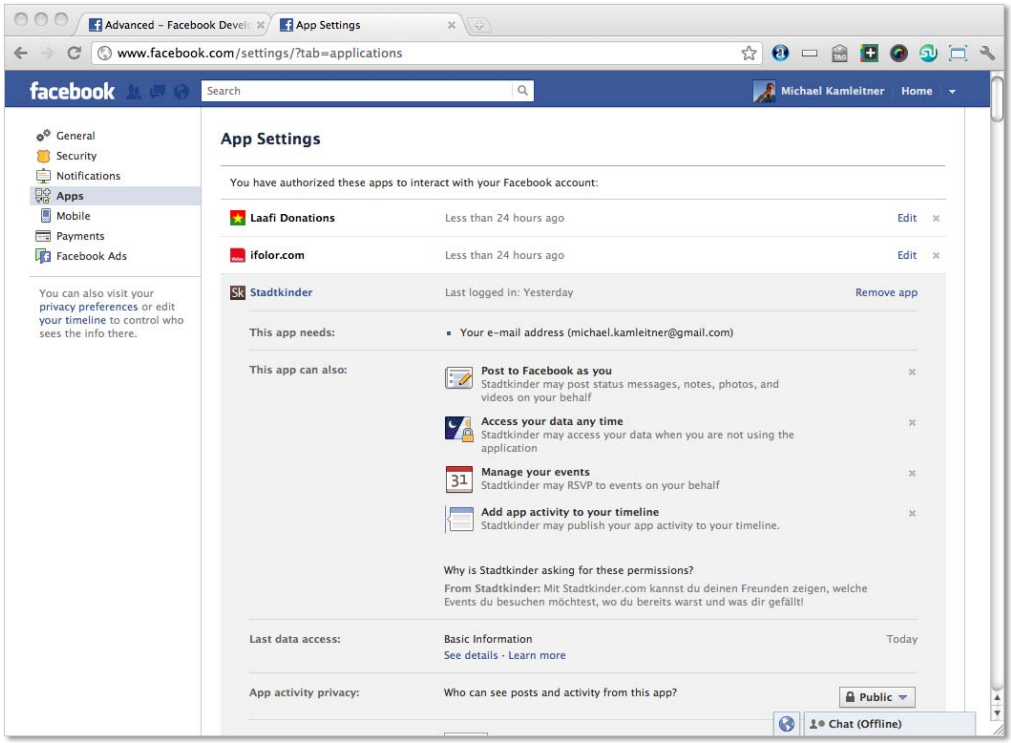


Abbildung 1.16 Gut versteckt: die Deinstallation von Facebook-Anwendungen

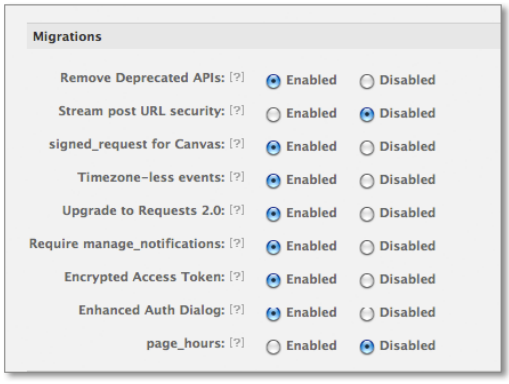


Abbildung 1.17 Migrationseinstellungen erlauben das Vorabtesten neuer oder geänderter Plattform-Features.

Migrationen

Facebook führt laufend Erweiterungen an den Schnittstellen seiner Plattform durch. Um Entwicklern die Vorbereitung auf kommende Plattform-Änderungen zu erleichtern, können diese meist im Vorhinein mit den Migrationseinstellungen für

bestimmte Anwendungen aktiviert werden. So können Sie als Entwickler Ihre Anwendungen auf Neuerungen vorbereiten und testen. Die jeweils verfügbaren Migrationseinstellungen werden laufend erweitert und – nach dem Veröffentlichen des entsprechenden Features – wieder entfernt. Deshalb möchte ich hier von einer detaillierten Auflistung absehen und wichtige Optionen stattdessen später in den jeweiligen Kapiteln erläutern.

Security

Hier können Betreiber von Anwendungen Vorkehrungen zur Erhöhung der Sicherheit im Betrieb vornehmen.

- Server Whitelist: Üblicherweise gestattet Facebook API-Aufrufe im Namen einer Anwendung von jeder beliebigen IP-Adresse, solange die Aufrufe nur mit einem passenden Access Token »unterzeichnet« werden. Um die Sicherheit zu erhöhen, kann die Quelle von API-Aufrufen in dieser Einstellung auf eine durch Beistriche separierte Liste von IP-Adressen eingeschränkt werden. Für maximale Sicherheit können Sie hier die IP-Adresse(n) Ihres Anwendungsserver(s) eintragen.
- Update Settings IP Whitelist: Mit dieser Einstellung kann eingeschränkt werden, von welchen IP-Adressen über den Facebook Developer Änderungen an den Einstellungen einer Anwendung vorgenommen werden können. Für maximale Sicherheit können Sie hier etwa die IP-Adresse Ihres Arbeitsplatzes eintragen.
- Update Notification Email: Gerade in größeren Entwicklerteams kann es sinnvoll sein, eine Benachrichtigungs-E-Mail zu erhalten, wenn ein Mitglied des Teams Änderungen an den Anwendungseinstellungen vorgenommen hat.

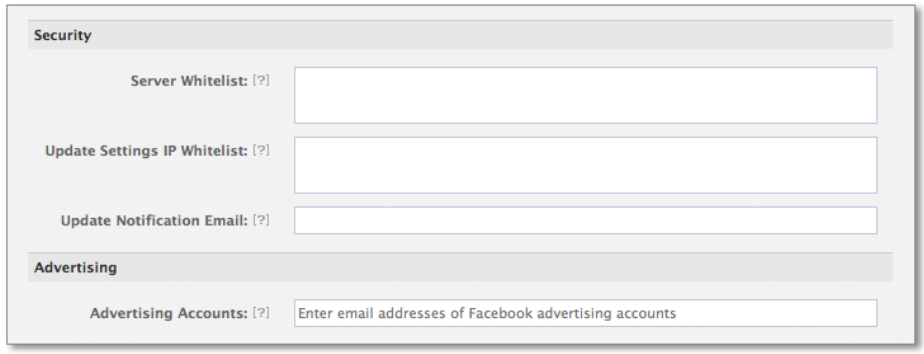


Abbildung 1.18 Sicherheits- und Advertising-Einstellungen

Advertising

Hier können die E-Mail-Adressen von Facebook-Advertising-Konten festgelegt werden, die mit der Anwendung verknüpft werden sollen.

Canvas Settings

In diesem Abschnitt können Sie Einstellungen zum Canvas, also der als iFrame realisierten »Leinwand«, die Ihrer Anwendung auf Facebook.com zur Verfügung steht, vornehmen.

- CANVAS WIDTH: Diese Einstellung betrifft Canvas-Anwendungen, die nicht als Tabs/Reiter integriert sind und deshalb normalerweise auf eine feste Breite von 760 Pixeln festgelegt sind. Mit der Option FLUID ist es jedoch möglich, den auf größeren Bildschirmen darüber hinaus zur Verfügung stehenden Platz zu nutzen. Während die auf 760 Pixel Breite fixierte Leinwand immer zentriert dargestellt wird, beginnt die Leinwand bei FLUID am linken Rand des Browser-Fensters und endet dynamisch 256 Pixel vor dem rechten Rand (dieser Bereich ist für die Seitenleiste von Facebook reserviert).
- CANVAS HEIGHT: Während die Höhe des Canvas-iFrames üblicherweise per FLUID dynamisch abhängig vom Content des iFrames vergrößert und verkleinert wird, kann mit der Einstellung SETTABLE und mittels JavaScript-SDK-Methode `FB.Canvas.setSize` die Höhe des iFrames manuell verändert werden (siehe Abschnitt 4.3, »Canvas-Methoden des SDKs«).
- SOCIAL DISCOVERY: Ist diese Option aktiviert, wird ein Newsfeed-Eintrag auf der Pinnwand jedes neuen Benutzers gepostet, der darauf hinweist, dass dieser Benutzer begonnen hat, die Anwendungen zu nutzen (*Michael is using Hello World! now*).
- BOOKMARK URL: Benutzer können Bookmarks zu ihren Lieblingsanwendungen speichern. Ein Klick auf das Bookmark führt den Benutzer normalerweise zur Canvas-URL der Anwendung (*http://apps.facebook.com/AppNamespace*). Mit dieser Option kann jedoch eine individuelle URL angegeben werden, auf die Benutzer, die über das Bookmark in die Anwendung einsteigen, geleitet werden sollen.

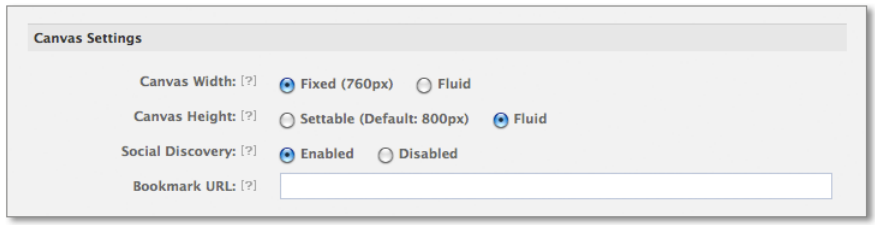


Abbildung 1.19 Einstellungen zum Application Canvas einer Anwendung

1.4.4 Benutzerrollen

Unter dem Menüpunkt ROLES können Sie als Anwendungsentwickler festlegen, welchen Benutzern oder Benutzergruppen Sie erweiterten Entwicklerzugriff auf die

jeweilige Facebook-Anwendung gewähren möchten. Dabei sind vier unterschiedliche Stufen der Zugriffsberechtigung möglich:

- ADMINISTRATOREN: Hier eingetragene Benutzer dürfen alle Einstellungen der Anwendung ändern, die Applikation im Sandbox-Modus benutzen und vor allem weiteren Benutzern erweiterte Zugriffsrechte unter ROLES verschaffen.
- DEVELOPER: Hier eingetragene Benutzer dürfen Einstellungen der Anwendung ändern und die Applikation im Sandbox-Modus verwenden. Sie dürfen anderen Benutzern aber keine Developer-Rechte erteilen oder entziehen.
- TESTER: Hier eingetragene Benutzer dürfen ausschließlich die Applikation im Sandbox-Modus benutzen, aber keine Einstellungen der Anwendung verändern.
- INSIGHTS: Hier eingetragene Benutzer haben ausschließlich Zugriff auf die von Facebook bereitgestellten Nutzungsstatistiken der Applikation (auch Administratoren und Developer haben selbstverständlich Zugriff auf diese Statistiken).

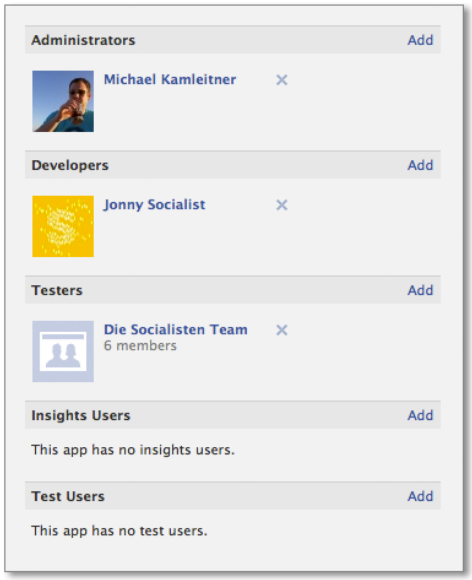


Abbildung 1.20 Management der Entwicklerberechtigungen

Management von mehrköpfigen Entwicklerteams und Testern

Gerade in Agenturen mit größeren Entwicklerteams und einer großen Anzahl an entwickelten Applikationen kann das Management der erweiterten Entwicklerzugriffsrechte mühsam und fehleranfällig sein – um zu gewährleisten, dass alle Teammitglieder jederzeit Zugriff auf alle im Unternehmen entwickelten Applikationen haben, muss darauf geachtet werden, jedes Teammitglied bei jeder App zumindest als »Developer« einzutragen. Verlässt ein Entwickler das Unternehmen oder wird

ein neuer Mitarbeiter eingestellt, müssen alle vorhandenen Applikationen entsprechend angepasst werden.

Glücklicherweise bietet Facebook seit August 2011 die Möglichkeit an, erweiterte Developer-Zugriffsrechte auch auf Basis von gewöhnlichen Facebook-Gruppen zu vergeben. Es empfiehlt sich also, für das eigene Unternehmen unter *http://www.facebook.com/groups* eine Gruppe für das Entwicklerteam anzulegen. Fortan genügt es, bei jeder neu erstellten Applikation einfach die Gruppe als »Developer« einzutragen. Mitarbeiter können dann zentral in dieser Gruppe ein- oder ausgetragen werden und haben automatisch Zugriff auf alle Applikationen. Daneben kann die Gruppe auch zur effizienten internen Kommunikation genutzt werden!

Ein ähnliches Problem kann sich ergeben, wenn mehrere Mitarbeiter eines Kunden eine in der Entwicklung befindliche Applikation im Sandbox-Modus testen möchten. Auch hier kann es sich lohnen, für jeden Kunden eine eigene Gruppe anzulegen und dieser die Berechtigung »Tester« zu erteilen. Übrigens: um als »Tester« eine Applikation in der Sandbox zu testen, muss der Account des jeweiligen Benutzers glücklicherweise nicht wie beschrieben per SMS oder Kreditkarte verifiziert sein.

Wichtige Links:

- *http://www.facebook.com/developer* – die Facebook-Developer-Applikation

1.4.5 Open Graph

Das ursprünglich 2010 eingeführte Konzept des *Open Graph* ermöglichte es Webentwicklern, eigene Webseiten auf einfache Weise in den sozialen Graphen von Facebook zu integrieren. Wesentlichstes Erscheinungsmerkmal des Open Graph war damals wie heute der LIKE-Button – ein iFrame oder JavaScript-basiertes Social Plugin, das auf beliebigen Webseiten eingebunden werden kann. Klickt ein Benutzer auf LIKE, wird automatisch ein Newsfeed-Posting auf der Pinnwand des Benutzers veröffentlicht, das sich auf die einbindende Webseite bezieht. Facebook geht dabei davon aus, dass jedes Objekt, das mittels LIKE in den sozialen Graphen eingebunden werden kann, auf einer eigenen, eindeutigen Webseite bzw. URL repräsentiert wird. Auf dieser Seite können Sie mittels Open Graph Meta Tags Angaben zum Inhalt des Objekts machen, also etwa Attribute wie Titel, Beschreibungstext oder Thumbnail-Grafik definieren.

Im September 2011 wurde das Open-Graph-Konzept bedeutend erweitert, weswegen ich es an dieser Stelle bei dieser kurzen Übersicht belassen und später in Kapitel 8, »Open Graph«, näher auf die damit verbundenen Konzepte eingehen werde.

Abbildung 1.21 Open Graph in Aktion – Webseite mit »Like«-Button und das daraus resultierende Wall-Posting

1.4.6 Credits

Möchten Sie in Ihrer Anwendung das von Facebook angebotene Micropayment-System *Facebook Credits* nutzen, müssen Sie in diesem Abschnitt die entsprechenden Einstellungen und Ihre Bankverbindung konfigurieren. Facebooks Micropayment-Lösung ist Kapitel 9, »Facebook Credits«, gewidmet.

1.4.7 Insights

Dieser Menüpunkt in der Entwickler-App führt zu ausführlichen Nutzungsstatistiken der Applikation. Hier können Sie auswerten, wie viele Benutzer Ihre Anwendung in einem beliebigen Zeitabschnitt verwendet haben, wie viele Wall-Postings erzeugt wurden, wie viele Benutzer die Anwendung wieder deinstalliert haben und vieles mehr.

1.5 Hello Facebook!

Für Ihre erste eigene Anwendung genügt es, wie in Abschnitt 1.4.1, »Einstellungen ›Basic‹« beschrieben, lediglich den Abschnitt Basic korrekt auszufüllen. Danach erstellen Sie im Root-Verzeichnis Ihres Webservers ein Unterverzeichnis namens *helloworld*, also etwa in */var/www/helloworld*. In diesem Verzeichnis legen Sie die Datei *index.php* mit folgendem Inhalt an:

```
<!DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
```

```
    <title>Hello Facebook!</title>
</head>
<body>
    <h1>Hello Facebook!</h1>
    <p>Willkommen bei meiner ersten Facebook-Anwendung!</p>
</body>
</html>
```

Listing 1.1 Die Datei »index.php« von Hello Facebook!

Wie Sie sehen, enthält die Datei *index.php* gewöhnlichen HTML-Code. Der im `<html>`-Element angeführte XML-Namespace `xmlns:fb` deutet als Einziges darauf hin, dass Sie es hier mit dem Rumpf einer Facebook-Anwendung zu tun haben könnten. Wenn Sie im Browser nun über die URL *http://apps.mycompany.com/helloworld* direkt auf Ihren Anwendungsserver zugreifen, führt das, wie zu erwarten, zu folgendem Resultat:

Abbildung 1.22 Hello Facebook! – direkt über den Application Server aufgerufen

Tipp: Testen Sie Ihre SSL-Unterstüzung!

Dies ist eine gute Gelegenheit, die SSL-Unterstützung Ihres Webspaces zu testen! Rufen Sie dazu die Anwendung mit dem Protokollkürzel *https* auf: *https://apps.mycompany.com/helloworld*. Je nach Browser wird ein erfolgreicher Zugriff per HTTPS mit einem Schloss-Symbol o. Ä. dargestellt – ein nicht vorhandenes oder fehlerhaftes SSL-Zertifikat führt hingegen zu einer Fehlermeldung.

Doch wie können Sie Hello Facebook! nun als Canvas-Anwendung in Facebook integrieren? Die Antwort ist einfach: Sie haben die Integration bereits mit der Angabe des App Namespace und der Canvas-URL vorgenommen! Um die Anwendung innerhalb von Facebook aufzurufen, steuern Sie die URL *http://apps.facebook.com/michaels-helloworld* im Browser an (*michaels-helloworld* ist dabei der individuell gewählte, eindeutige App Namespace):

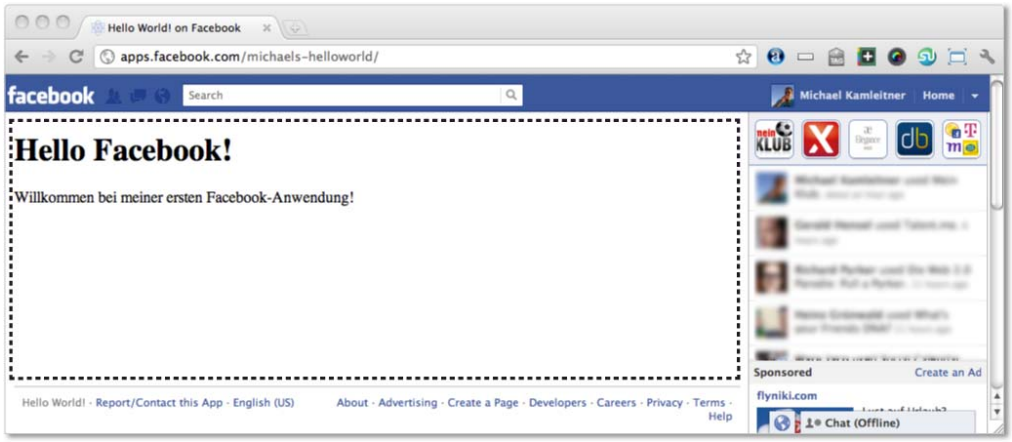


Abbildung 1.23 Hello Facebook! als Canvas-Applikation auf Facebook. Hervorgehoben ist der Bereich des iFrames, der vom Anwendungsserver bereitgestellt wird.

Gratulation, Ihre erste – zugegebenermaßen einfache und wenig »soziale« – Facebook-Anwendung ist damit bereits online!

Canvas-Anwendungen auf Facebook.com

Da es für das Verständnis der Integration von Canvas-Anwendungen auf Facebook elementar ist, sei hier Schritt für Schritt erklärt, wie die oben abgebildete Hello-Facebook!-Applikation funktioniert:

- Beim Öffnen von *http://apps.facebook.com/michaels-helloworld* sendet der Browser des Benutzers eine gewöhnliche HTTP-Anfrage an Facebook.com ab.
- Facebook.com antwortet mit einer Seite, die lediglich den Kopf- und Fußbereich und die Seitenleiste, im Content-Bereich allerdings nur ein `<iframe>`-Element enthält.
- Anhand des App Namespaces *michaels-helloworld* erkennt Facebook.com, welche Facebook-Anwendung für den konkreten Aufruf zuständig ist. Die in den Einstellungen dieser Applikation festgelegte CANVAS URL wird als `src`-Attribut im iFrame gesetzt, etwa: `<iframe src="http://apps.mycompany.com/helloworld"/>`.
- Der Browser des Benutzers setzt einen zweiten Request an `http://apps.mycompany.com/michaels-helloworld` ab.
- Der Anwendungsserver antwortet mit dem Inhalt von *index.php* – gewöhnlichem HTML-Code, der vom Browser anschließend im Bereich des iFrames angezeigt wird.

1.5.1 Hello Facebook! als Tab-Anwendung

Nachdem Sie Ihre erste eigene Anwendung als klassische Canvas-Anwendung ausgeführt haben, fragen Sie sich nun vielleicht, wie Sie diese Anwendung in den Tab/Reiter Ihrer bestehenden Page einbinden können. Dazu müssen Sie zuerst in den Anwendungseinstellungen unter BASIC den Abschnitt PAGE TAB ausfüllen:

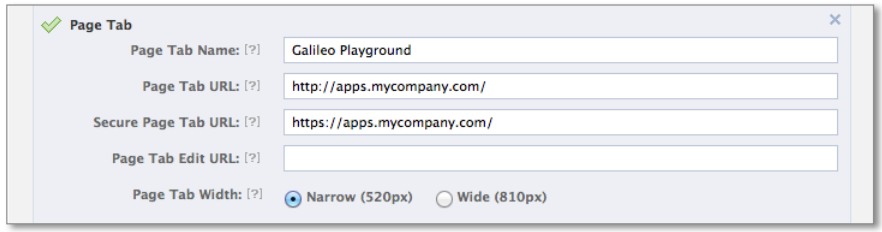


Abbildung 1.24 Einstellungen für Tab-Anwendungen

Neben dem Titel, unter dem das Tab eingebunden werden soll, ist hier vor allem die Angabe der PAGE TAB URL und ihres SSL-Pendants wichtig: Von dieser URL wird Facebook später den Inhalt des Tabs in den bereitgestellten iFrame laden. Für dieses einfache Beispiel können Sie hier dieselbe URL angeben wie weiter oben bei CANVAS URL – damit wird bewirkt, dass im Tab der gleiche Inhalt angezeigt wird wie beim Aufruf der Anwendung über *http://apps.facebook.com /AppNamespace*.

Anlegen einer Testseite

Wenn Sie noch keine Seite auf Facebook angelegt haben, können Sie dies für Testzwecke unter *http://www.facebook.com/pages* sehr einfach tun. Wählen Sie dazu eine beliebige Kategorie, und geben Sie den Namen Ihrer Testseite an. Nach dem Anlegen können Sie in den Seiteneinstellungen, die Sie unter EDIT PAGE finden, im Abschnitt MANAGE PERMISSIONS einstellen, dass die Seite nicht öffentlich, sondern nur für Seitenadministratoren sichtbar sein soll. Aktivieren Sie dazu die Option UNPUBLISH PAGE (ONLY ADMINS CAN SEE THIS PAGE).

Um die Tab-Anwendung nun auf Ihrer Testpage einzubinden, ist derzeit leider die manuelle Eingabe einer bestimmte URL notwendig – Facebook hat die entsprechende Funktion nach der Einstellung der Anwendungsprofile im Februar 2012 leider noch nicht im Benutzer-Interface von Facebook.com nachgezogen. Die aufzurufende URL lautet:

```
http://www.facebook.com/add.php?api_key=AnwendungsID&pages=1
```

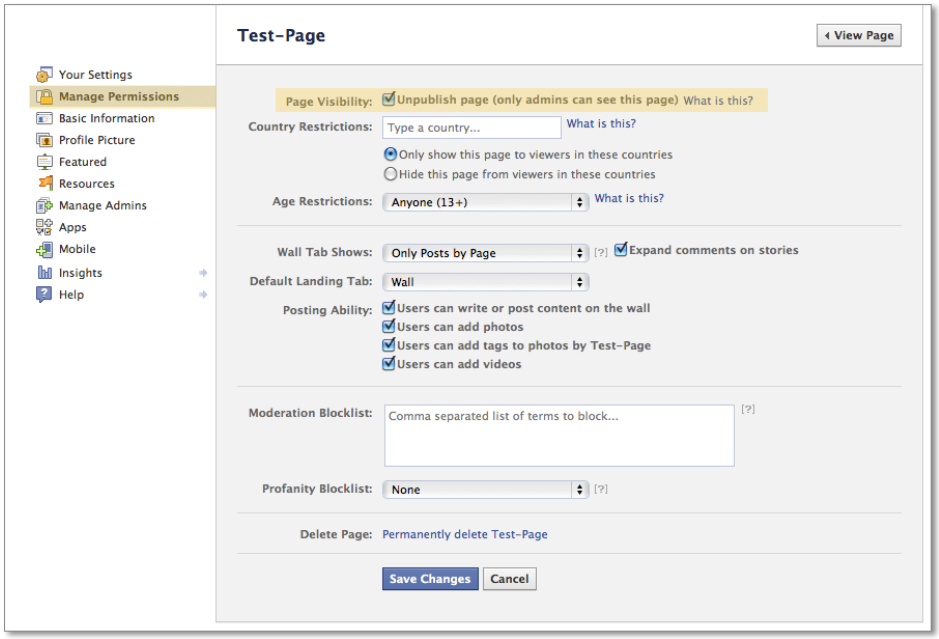


Abbildung 1.25 Anlegen einer nicht-öffentlichen Testseite

Facebook zeigt daraufhin einen Dialog an, der es dem Benutzer erlaubt, das Anwendungs-Tab zu einer beliebigen Seite hinzuzufügen, für die der Benutzer Administrationsrechte besitzt:

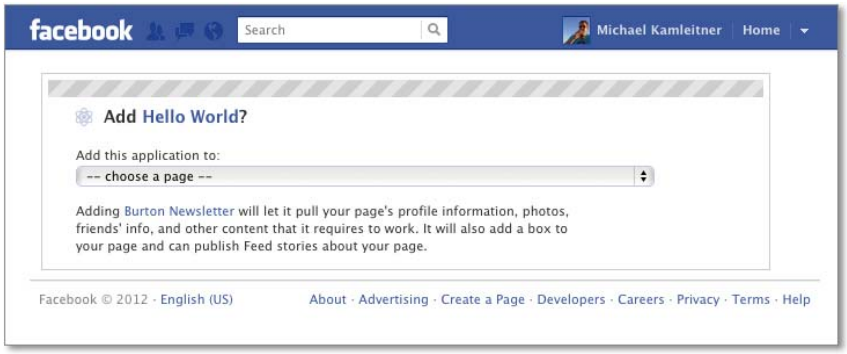


Abbildung 1.26 Hinzufügen eines Anwendungs-Tabs zu einer bestehenden Seite

Da die Select-Box mit der Liste der administrierten Seiten bei einer großen Zahl an Seiten aufgrund der fehlenden alphabetischen Sortierung schlecht bedienbar ist, kann beim URL-Aufruf die numerische ID der Zielseite auch gleich mit angegeben werden:

```
http://www.facebook.com/add.php?api_key=AnwendungsID&pages=1&page=SeitenID
```

Nach Bestätigung des Dialogs wird der Benutzer automatisch auf die Seite weitergeleitet – das neue Tab ist dann auch schon unter dem in der Einstellung PAGE TAB NAME gewählten Namen als Button unter dem Cover-Foto zu finden!

Abbildung 1.27 Testpage mit der als Tab/Reiter eingebundenen Hello-Facebook!-Anwendung

1.6 Ressourcen und Tools für Entwickler

1.6.1 Offizielle Dokumentation

Facebook stellt die vollständige und aktuelle Dokumentation zu allen Bereichen der Entwickler-Plattform online zur Verfügung. Die Dokumentation dient Entwicklern als technische Referenz, die oft mit ausführlichen Beispielen den Einsatz der jeweiligen Schnittstellen erläutert. Darüber hinaus wird in grundlegenden Artikeln auch eine Anleitung zur inhaltlichen Konzeption von Facebook-Anwendungen in Form von Best-Practice-Beispielen geboten.

1.6.2 Developer-Blog und Live-Status

Das Developer-Blog gehört zur absoluten Pflichtlektüre jedes Facebook-Developers – Sie sollten das Blog unbedingt regelmäßig besuchen oder besser noch per RSS oder E-Mail abonnieren! Hier erfahren Sie oft schon Monate vorher von geplanten Änderungen und Erweiterungen der Facebook-Plattform. Größere Änderungen auf der Plattform, die die Funktionsfähigkeit von bestehenden Applikationen beeinträchtigen, werden von Facebook immer mindestens 90 Tage zuvor angekündigt, um vor allem Entwicklern mit zahlreichen bestehenden Applikationen ausreichend Zeit zu geben, ihre Anwendungen auf den aktuellen Stand zu bringen.

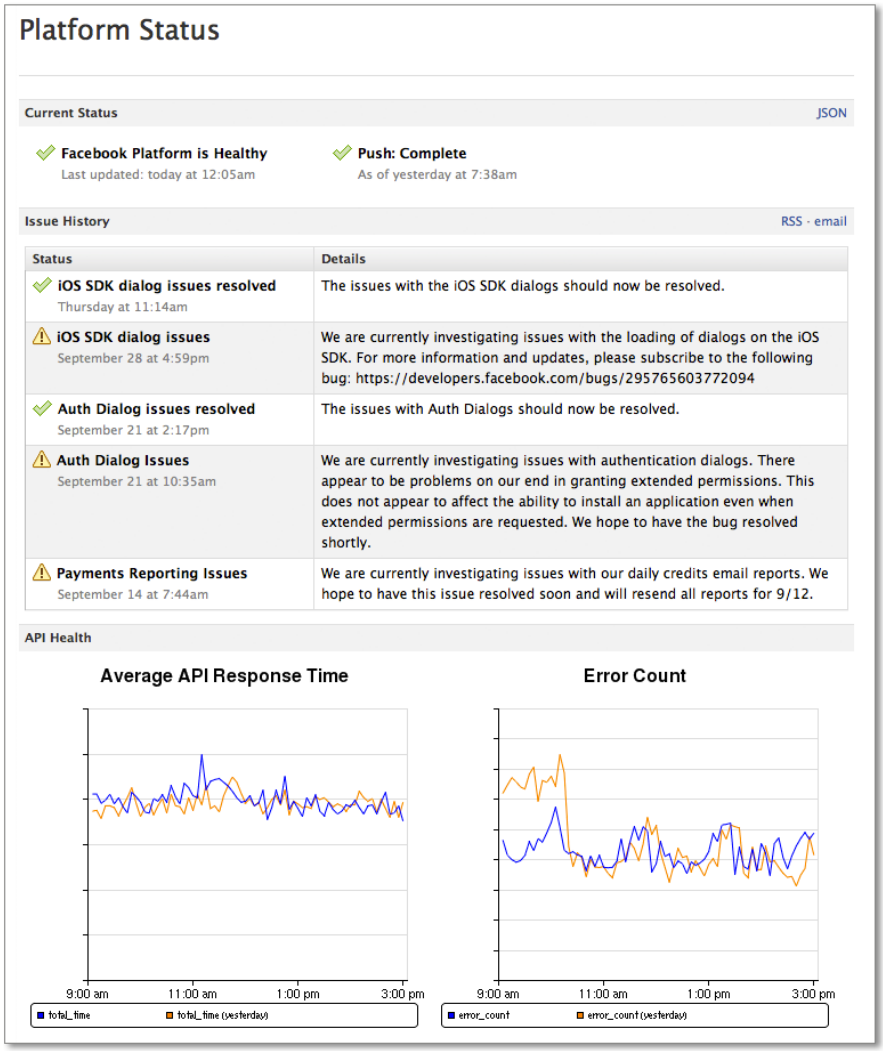


Abbildung 1.28 Live-Status der Facebook-Plattform

Auf der Live-Status-Seite werden hingegen Probleme – teilweise auf Ebene einzelner API-Methoden – und Downtimes der gesamten Plattform oder ihrer Teile vermerkt, ebenso wie der Zeitpunkt, ab dem diese Probleme wieder behoben sein sollten. Die durchschnittlichen API-Antwortzeiten und die Anzahl der Fehlermeldungen ermöglichen es darüber hinaus auch, schon vor einer offiziellen Bestätigung etwaige Plattform-Probleme nachzuvollziehen.

Wichtig: Für Facebook-Anwendungsentwickler empfiehlt es sich, sowohl Blog als auch Live-Status per E-Mail oder RSS-Reader zu abonnieren!

1.6.3 Developer-Forum

Das über lange Jahre betriebene offizielle Developer-Forum von Facebook wurde 2011 im Rahmen einer Partnerschaft mit dem Q&A-Service Stackoverflow.com abgelöst. Zwar sind die alten Beiträge noch unter *http://forum.developers.facebook.net/* archiviert, da sich viele der Beiträge aber auf mittlerweile veraltete Schnittstellen und APIs beziehen, ist Stackoverflow der geeignetste Ort, um Rat von anderen Entwicklern einzuholen. Anders als klassische Foren organisiert Stackoverflow unterschiedliche Themengebiete nicht nach Subforen, sondern nach Tags (Graph API, Social Plugins, Open Graph ...) und belohnt aktive Community-Mitglieder mit einem Punktesystem.

1.6.4 Developer-Tools

Auf dieser Seite verlinkt Facebook alle offiziellen Tools, die das Leben eines Facebook-Anwendungsentwicklers leichter machen. Mit dem *Access Token Tool* können Sie benutzerbezogene Access Tokens zum Testen einer Applikation erstellen. Im *Changelog* werden Änderungen und Korrekturen an verschiedensten Teilen der Plattform chronologisch festgehalten. Der *Open Graph Debugger* (auch bekannt als *Linter*) hilft beim Testen der eigenen Open Graph Tags, während die *Test User API* das Erstellen von Benutzer-Accounts zum Testen von Anwendungen erlaubt.

Für Einsteiger besonders interessant sind die folgenden beiden Tools: Der *Graph API Explorer* erlaubt das spielerische Erforschen der Graph API, ihrer Objekte und Verbindungen. Mit wenigen Klicks bekommen Sie einen guten Überblick darüber, mit welchen API-Aufrufen Sie welche Informationen über Benutzer, Fotos, Pages etc. abfragen können. Auch schreibende Zugriffe, etwa zum Posten auf der Pinnwand eines Benutzers, können mit dem Explorer ausprobiert werden. Da der Explorer auch das erweiterte Berechtigungssystem der Facebook-Autorisierung unterstützt, ist das Tool auch ein geeigneter Weg, um herauszufinden, welche Berechtigungen Sie benötigen, um bestimmte Informationen (etwa die E-Mail-Adresse des Benutzers) zu erfragen oder zu veröffentlichen.

Während sich der Explorer ganz der Graph API widmet, dient die *JavaScript Test Console* dem Erforschen des JavaScript SDKs. Über ein Eingabefeld können beliebige JavaScript-Kommandos, etwa zum Erzeugen von Wall-Postings, zum Verschicken von Anwendungseinladungen oder auch zur Applikationsautorisierung, ausprobiert werden. Eine umfangreiche Bibliothek an Beispielen erleichtert Ihnen hierbei den Einstieg und kann per Copy & Paste sogar als Basis für eigene Anwendungen dienen.

1.6.5 Bug Tracker

Früher oder später werden Sie wie jeder ambitionierte Facebook-Anwendungsentwickler an einen Punkt gelangen, an dem weder das Studieren der Dokumentation

oder ausführliche Experimente noch das Wühlen im Developer-Forum zur Lösung eines bestimmten Problems führen. Speziell wenn die Ergebnisse eines API-Aufrufs von dem laut Dokumentation oder früherer Erfahrung zu erwartenden Ergebnis abweichen, liegt der Verdacht nahe, einem Plattform-Bug auf die Spur gekommen zu sein. In diesem Fall gehört es zum guten Ton, den offiziellen Bug Tracker nach eventuell bereits bestehenden Bug Reports zum gleichen Thema zu durchsuchen. Werden Sie dort nicht fündig, lohnt es sich in jedem Fall, einen eigenen Bug Report zu schreiben. Fast immer erhalten Sie so rasch Bestätigung von anderen Entwicklern, die mit dem gleichen Problem zu kämpfen haben, und auch das Team der Facebook-Plattform antwortet üblicherweise sehr rasch. Um diesem die Arbeit zu erleichtern, ist es wichtig, den Fehler detailliert und, wenn möglich, reproduzierbar zu beschreiben, am besten mit einer live abrufbaren URL zur Demonstration des Problems. Durch das Abonnieren des eigenen Bug Reports werden Sie außerdem per Benachrichtigungs-E-Mail auf dem Laufenden gehalten und erfahren so rasch, wenn ein Bug behoben wurde.

Wichtige Links:

- *http://developers.facebook.com/tools/* – Überblick über die von Facebook bereitgestellten Developer-Tools
- *http://facebook.overflow.com* – das neue, in Partnerschaft mit Stackoverflow geführte Developer-Forum
- *http://forum.developers.facebook.net/* – das mittlerweile eingestellte Developer-Forum von Facebook
- *http://developers.facebook.com/blog/*, *http://developers.facebook.com/live_status* – das Facebook-Developer-Blog und der Live-Status der Plattform
- *http://bugs.developers.facebook.net/* – der Facebook Bug Tracker

1.7 Terms of Service und Plattform-Guidelines

Obwohl unabhängige Entwickler keinen formalen Vertrag mit Facebook eingehen müssen, um eine Anwendung auf der Facebook-Plattform betreiben zu können, gibt es doch ein ausführliches Regelwerk, das der Entwickler beim Erstellen einer Facebook-Applikation akzeptieren muss. Daneben stellt Facebook noch eine Reihe weiterer Regelwerke bereit, die verschiedene Aspekte der Plattform regeln:

- *Terms of Service:* Dieses Regelwerk ist für jeden Facebook-Benutzer relevant und legt fest, unter welchen Bedingungen Facebook verwendet werden darf. So sind etwa Benutzer dazu aufgefordert, nur ein Profil pro Person anzulegen und dieses auch mit dem realen Namen zu führen.

- *Facebook Platform Policy*: Dieses Dokument beschreibt ausführlich, was externe Anwendungsentwickler bei der Integration mit Facebook zu beachten haben. Geregelt wird etwa, welche Daten Anwendungsentwickler von Benutzern beziehen dürfen und ob diese zwischengespeichert werden dürfen. Es finden sich auch allgemeine Empfehlungen dazu, wie eine Anwendung zu gestalten ist, um von Benutzern nicht potenziell als störend oder spamartig wahrgenommen zu werden.
- *Advertising Guidelines:* Hier werden Regeln festgelegt, die bei der Nutzung von Facebook-Werbemitteln wie etwa den Facebook Ads oder Sponsored Stories beachtet werden müssen.
- *Promotions Guidelines:* Wer Facebook oder Facebook-Anwendungen zur Durchführung von Marketing-Kampagnen, Gewinnspielen oder sonstigen Werbeaktionen nutzen möchte, findet hier alle zu beachtenden Richtlinien. So ist es etwa explizit untersagt, Gewinnspiele ausschließlich auf der Pinnwand einer Page zu platzieren – dazu fordert Facebook hierfür die explizite Einrichtung einer Applikation bzw. eines Tabs/Reiters (eine Regel, an die sich leider auch heute noch zahlreiche große Firmen und Marken nicht halten möchten).
- *Payment Terms:* Wer in Facebook-Anwendungen elektronische Zahlungsmittel wie die von Facebook selbst angebotenen Facebook Credits oder ein Payment-System eines Drittanbieters verwenden möchte, wird in diesem Dokument fündig. *Achtung:* Für Spiele, die auf der Facebook-Plattform umgesetzt werden, gilt seit Juli 2011 die Pflicht, ausschließlich die hauseigenen Facebook Credits als Zahlungsmittel anzunehmen.

Auch wenn die teilweise recht ausführlich und nüchtern geschriebenen Dokumente abschreckend wirken mögen – es lohnt sich jedenfalls, sie ausführlich zu studieren. Obwohl Facebook wohl kaum die personellen Ressourcen aufbringen können wird, die notwendig wären, um die enorme Anzahl an Applikation auf Konformität mit den Richtlinien zu prüfen, hat das Unternehmen doch mehrere Mechanismen zur Überwachung der Plattform geschaffen. So werden Applikationen, die etwa in Relation zur Zahl der Benutzer auffällig viele API-Zugriffe – etwa zum Verfassen von Wall-Postings oder zum Veröffentlichen von Markierungen auf Fotos – absetzen, automatisch gemeldet und anschließend geprüft. Auch können Facebook-Benutzer etwa bei Wall-Postings, die durch eine Anwendung generiert wurden, etwaigen Missbrauch melden. All dies führt dazu, dass fragwürdige Anwendungen und grobe Verstöße gegen die Platform Policies tatsächlich immer wieder erkannt und von Facebook geahndet werden. Letzteres bedeutet im Regelfall eine Sperrung der Applikation oder der Seite. Fühlen Sie sich als Entwickler der Anwendung zu Unrecht bestraft, bleibt nur der oft langsame Weg per E-Mail, um eine Klärung herbeizuführen – bis dahin bleibt Ihre Applikation aber gesperrt.

Wichtige Links:

- *http://www.facebook.com/terms.php* – Terms of Service für alle Facebook-Benutzer
- *http://developers.facebook.com/policy/* – Platform Policy für Anwendungsentwickler
- *http://www.facebook.com/ad_guidelines.php* – Advertising Guidelines für die Nutzung der Facebook-Werbemittel
- *http://www.facebook.com/promotions_guidelines.php* – Promotions Guidelines regeln die Durchführung von Marketing-Kampagnen und Gewinnspielen auf Facebook.
- *http://www.facebook.com/payments_terms* – Payment Terms regeln die Durchführung von Zahlungen mit Facebook Credits und anderen Zahlungsmitteln innerhalb von Facebook-Applikationen.
- *http://developers.facebook.com/docs/guides/policy/policy_checklist/* – übersichtliche Checkliste für Anwendungsentwickler

Kapitel 2
Authentifikation und Autorisierung

Dieses Kapitel beschäftigt sich mit einer der wichtigsten Grundlagen der Facebook-Anwendungsentwicklung – der Authentifikation und Autorisierung von Applikationen durch den Benutzer. In diesem Kapitel erfahren Sie, wie Sie das Autorisierungskonzept von Facebook nutzen, um Ihre Benutzer zu identifizieren und Zugriff auf ihre Profildaten zu erhalten.

Bisher haben Sie gelernt, dass Canvas-Applikationen auf der Facebook-Plattform als iFrame innerhalb von Facebook.com ausgeführt werden und dabei im Wesentlichen wie herkömmliche, von Facebook unabhängige Webseiten oder -applikationen funktionieren. Nachdem Sie eine erste einfache Applikation erstellt haben, drängt sich Ihnen als ambitioniertem Entwickler jetzt wahrscheinlich eine Frage auf: Wie kann man innerhalb der Anwendungslogik herausfinden, welcher Facebook-Benutzer meine Applikation gerade ausführt? Wie können Entwickler auf Profildaten des Benutzers, etwa seinen sozialen Graphen, zugreifen? Oder wie können sie gar im Namen des Benutzers Interaktionen mit der Facebook-Plattform anstoßen? Die Antworten auf diese Fragen finden Sie im Authentifikations- und Autorisierungskonzept von Facebook.

Authentifikation – Sag mir, wer du bist!

Was versteht man eigentlich unter *Authentifikation*? Nun, Anwendungsentwickler auf der Facebook-Plattform sehen sich vor zwei Herausforderungen gestellt:

- Um die meisten herkömmlichen Webanwendungen zu nutzen, ist eine Anmeldung mit einem eigenen Benutzerkonto notwendig. Dieser Login stellt also die *Authentizität* des Benutzers gegenüber der Anwendung sicher. Auf der Facebook-Plattform werden unsere Anwendungen per iFrame in den »Rahmen« von Facebook.com integriert. Ein eigenes Benutzerkonto innerhalb der Anwendung zu erstellen wäre zwar technisch möglich, würde aber dem Grundgedanken der Facebook-Plattform zuwiderlaufen. Immerhin ist einer der größten Vorteile der Plattform ja gerade, nicht für jede Anwendung ein neues Konto erstellen zu müssen. Der Authentifikationsmechanismus von Facebook ermöglicht es Anwendungsentwicklern, Benutzer anhand ihrer eindeutigen numerischen Benutzer-ID zu identi-

fizieren, und ersetzt damit die Notwendigkeit für eigene Login-Mechanismen in der Anwendung.

- Umgekehrt ist es oft wünschenswert, bei bestimmten Aktionen, die ein Benutzer in unserer Anwendung durchführen kann, entsprechende Interaktionen auf Facebook.com selbst anzustoßen. Dies kann z. B. das Veröffentlichen eines Wall-Postings sein, wenn ein Benutzer in einem Geschicklichkeitsspiel einen neuen Highscore erzielt. Facebook ermöglicht solche Interaktionen und bietet mit seinem Autorisierungskonzept die Möglichkeit für Anwendungen, im Namen ihrer Benutzer auf Facebook.com zu agieren.

2.1 OAuth 2.0

Facebook baut dabei auf dem weit verbreiteten OAuth-2.0-Standard auf – einem Identifikationsprotokoll, das auch von anderen sozialen Netzwerken und Webservices wie etwa Google, Twitter oder Foursquare genutzt wird. Die Unterstützung des OAuth-Standards hat den großen Vorteil, dass Sie sich als Entwickler nur einmal mit den Grundkonzepten des OAuth-Identifikationsablaufs vertraut machen müssen und die so gewonnenen Kenntnisse dann sehr einfach auf verschiedenen Plattformen anwenden können.

Die Grundidee von OAuth: Benutzer, die bereits auf einer Webseite A registriert sind, dort also ein Benutzerkonto besitzen, können dieses Benutzerkonto auf anderen unabhängigen Webseiten B, C, D ... zum Login verwenden. Genauer gesagt: Der Benutzer verwendet seine auf Webseite A bereits bekannte Identität, um auf die Webseiten B, C, D ... Zugriff zu erhalten. Sie ahnen es schon – Webseite A ist in unserem Fall Facebook.com, B, C, D sind unsere Applikationen! Der grundlegende Ablauf einer Authentifikation per OAuth ist folgender:

- Ein Benutzer surft auf Webseite B, auf der er kein Benutzerkonto besitzt. Er findet einen Login-Button, der mit Mit Facebook anmelden oder ähnlich beschriftet ist.
- Ein Klick auf diesen Login-Button löst eine Weiterleitung – weg von Webseite B – auf eine spezielle Seite von Facebook.com aus.
- Diese spezielle Seite wird *OAuth-Dialog* genannt. Im Wesentlichen stellt Facebook.com dabei dem Benutzer die Frage: »Möchtest du dein Konto auf Facebook.com für den Zugriff auf Webseite B benutzen?«
- Klickt der Benutzer auf Ja, leitet Facebook.com den Benutzer wieder zurück zu Webseite B. Gemeinsam mit der Weiterleitung wird ein temporäres Token übergeben, das der Anwendung auf Webseite B erlaubt, im Namen des Benutzers zu handeln.

Wichtig: Ein Grundprinzip von OAuth ist, dass die »konsumierende Seite«, also Webseite B, C, D ..., niemals die vollständigen Login-Daten, also vor allem nicht das Passwort des Benutzers auf Webseite A, zu Gesicht bekommt. Stattdessen wird lediglich ein Token ausgetauscht, das Webseite B, C, D ... autorisiert, im Namen des Benutzers zu handeln. Dies ist einerseits praktisch, da ein Ändern des Kennworts auf Webseite A nicht mehr zur Folge hat, dass bereits erteilte Autorisierungen von Webseite B, C, D ... aufgrund eines veralteten Kennworts nicht mehr funktionierten. Andererseits erhöht dies natürlich die Sicherheit für den einzelnen Benutzer enorm, da das Kennwort nicht mehr im Klartext weitergegeben werden muss.

OAuth wurde im Zuge der »Web 2.0«-Ära entwickelt und ist heute die Basis der meisten Webapplikationen, die eine API zum Zugriff von außen bieten: Twitter, Foursquare, Google und natürlich Facebook selbst – sie alle basieren auf dem OAuth-Standard.

Wichtiger Link:

- *http://tools.ietf.org/pdf/draft-ietf-oauth-v2-12.pdf* – OAuth-2.0-Spezifikation

Die Facebook-Plattform unterstützt zwei unterschiedliche Modelle des OAuth-Ablaufs: Der *serverseitige* Ablauf kommt immer dann zum Einsatz, wenn Zugriffe auf die Facebook-Plattform im Namen eines Benutzers in der Anwendungslogik, also im serverseitigen Code, erfolgen sollen. Der *clientseitige* Ablauf kommt hingegen dann zum Einsatz, wenn der Zugriff auf die API im Browser per JavaScript SDK oder auch auf einer nativen mobilen Applikation auf iOS oder Android erfolgen soll.

2.2 Serverseitige Authentifikation

Bei der Authentifikation wird einerseits sichergestellt, dass der Benutzer mit einem gültigen Konto bei Facebook angemeldet ist (*Authentifikation*), zum anderen wird vom Benutzer die Erlaubnis zum Ausführen der Facebook-Applikation eingeholt (*Autorisierung*). Dies ist wichtig, weil Anwendungen, die vom Benutzer autorisiert wurden, Zugriff auf bestimmte Profilinformationen des Benutzers haben – um die Privatsphäre des Benutzers zu wahren, darf dies nur nach erfolgreicher Autorisierung möglich sein. Authentifikation und Autorisierung erfolgen im Facebook-Konzept zur gleichen Zeit.

2.2.1 Weiterleitung zum Facebook-OAuth-Dialog

Authentifikation und Autorisierung werden gestartet, indem die Anwendung den Benutzer per *Redirect zum OAuth-Dialog* von Facebook weiterleitet. Dies kann etwa ausgelöst werden, wenn der Benutzer innerhalb Ihrer Anwendung auf einen Login- oder Facebook-Connect-Button klickt.

Dem Redirect müssen im GET-Parameter `client_id` die Applikations-ID und im Parameter `redirect_uri` eine Redirect-URL mitgegeben werden. Die Redirect-URL ist dabei jene, an die Facebook den Browser nach erfolgter oder auch nach abgebrochener Authentifikation zurückleitet. Und so sieht die URL zum OAuth-Dialog aus:

```
https://www.facebook.com/dialog/oauth?client_id=YOUR_APP_ID
&redirect_uri=YOUR_URL
```

Achtung: Die Redirect-URL muss innerhalb jener Domain liegen, die der Entwickler als SITE URL in den Anwendungseinstellungen festgelegt hat! Facebook überprüft dies aus Sicherheitsgründen und bricht bei abweichender Domain den Authentifikationsvorgang mit einer Fehlermeldung ab.

Um den OAuth-Dialog auszuprobieren, erstellen Sie nun im Root-Verzeichnis Ihres Webservers ein neues Verzeichnis */var/www/login* und legen dort die Datei *index.php* mit folgendem Inhalt an:

```
<?
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
define('SITE_URL', 'http://apps.mycompany.com');
$login_url = "https://www.facebook.com/dialog/oauth?client_id=".APP_ID.
"&redirect_uri=".SITE_URL."/login/callback.php";
?>
<DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
   Sie sind nicht angemeldet!
   <a href="<?= $login_url ?>">Login mit Facebook</a>
</body>
</html>
```

Listing 2.1 Anzeige eines Login-Links, der auf den OAuth-Dialog von Facebook verweist

Setzen Sie die Konstanten `APP_ID`, `APP_SECRET` und `SITE_URL` dabei auf jene Werte, die Sie im Developer für Ihre Hello-World!-Anwendung festgelegt haben. Wie Sie sehen, bildet der Code vorerst nur die URL des OAuth-Dialogs in `$login_url` nach dem beschriebenen Schema. Im Browser sieht das so aus:

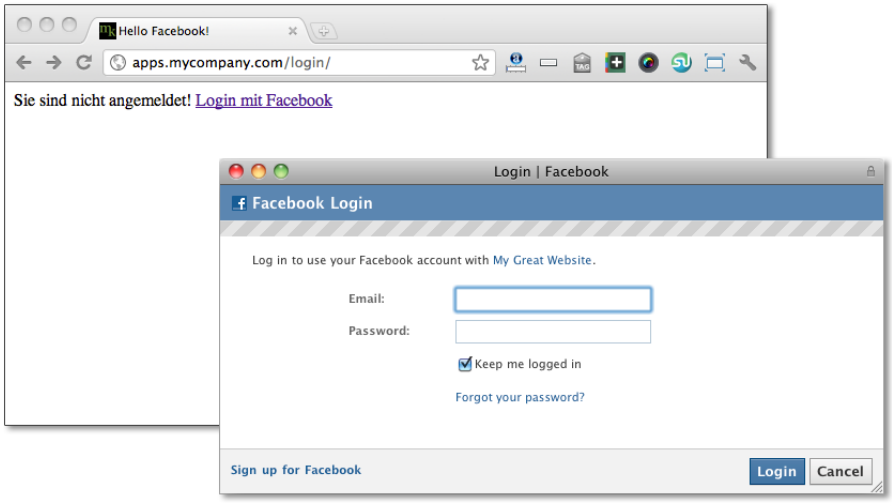


Abbildung 2.1 OAuth-Dialog – Aufforderung zum Login mit dem Facebook-Konto

Beim Aufruf des Dialogs überprüft Facebook anhand des Login-Cookies von Facebook.com, ob der Benutzer in der aktuellen Browser-Session bereits auf Facebook eingeloggt ist. Ist dies nicht der Fall, zeigt der OAuth-Dialog zuerst eine Facebook-Login-Maske an, in der sich der Benutzer mit seinem Facebook-Konto anmelden muss.

Nach erfolgtem Login, oder wenn der Benutzer bereits zum Zeitpunkt des Dialogaufrufs bei Facebook angemeldet war, erscheint der Facebook-OAuth-Dialog zur Authentifikation und Autorisierung von Anwendungen:

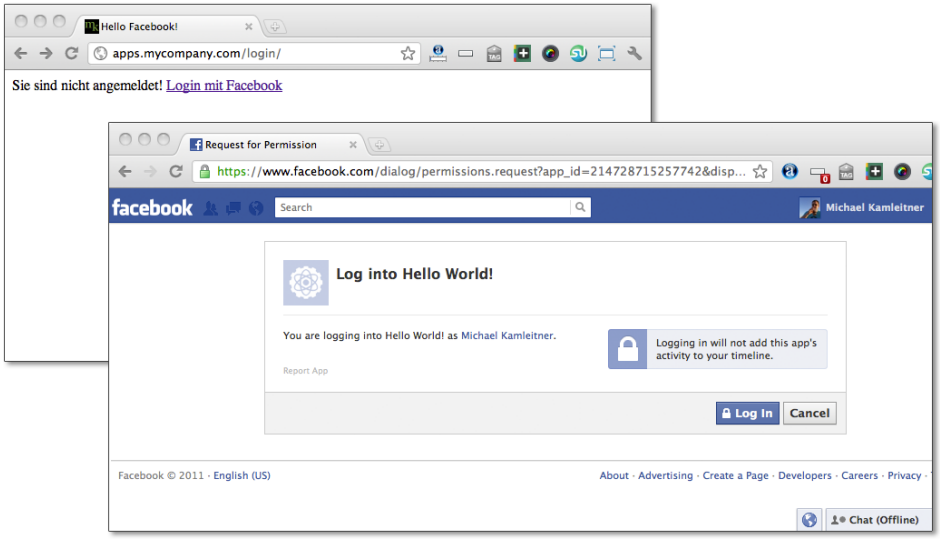


Abbildung 2.2 OAuth-Dialog – Aufforderung zur Autorisierung der Anwendung

In diesem Dialog wird dem Benutzer angezeigt, welche Anwendung er in Begriff ist zu autorisieren. Die Anwendung wird dabei mit Titel (»Log into Hello World!«) und Logo visualisiert. Im linken Teil des Dialogs wird dem Benutzer detailliert aufgelistet, auf welche Profildaten die Anwendung bei erfolgreicher Autorisierung zugreifen darf – standardmäßig sind dies nur Basisinformationen wie Benutzer-ID, Name, Geschlecht, Profilbild und – besonders wichtig für den Anwendungsentwickler – die Liste der Facebook-Freunde des Benutzers. Für umfassendere Zugriffsrechte müssen Anwendungen erweiterte Zugriffsrechte (*Extended Permissions*) vom Benutzer einholen. Dazu müssen Sie einen entsprechend vorbereiteten Parameter namens `scope` an den OAuth-Dialog übergeben:

```
https://www.facebook.com/dialog/oauth?client_id=YOUR_APP_ID
&redirect_uri=YOUR_URL&scope=email,user_photos
```

Der so aufgerufene OAuth-Dialog holt vom Benutzer zusätzlich zu den Basisinformationen die Berechtigungen zum Zugriff auf die E-Mail-Adresse und die Facebook-Fotos und -Alben des Benutzers ein:

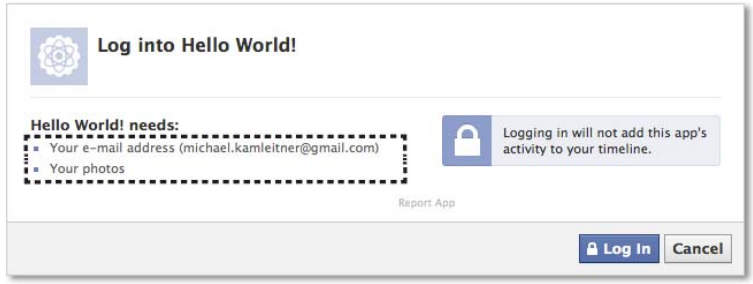


Abbildung 2.3 Der OAuth-Dialog – Aufforderung zur Autorisierung der Anwendung. Hervorgehoben sind zusätzlich geforderte Berechtigungen zum Zugriff auf die E-Mail-Adresse und Fotos des Benutzers.

2.2.2 Der Callback am Anwendungsserver

Unabhängig davon, ob der Benutzer im Facebook-OAuth-Dialog auf LOG IN oder CANCEL klickt, passiert nun Folgendes: Facebook leitet den Browser des Benutzers mittels HTTP-Redirect zu jener URL, die beim Aufruf des Dialogs im Parameter `redirect_uri` übergeben wurde. In unserem Fall wird also *http://apps.mycompany.com/login/callback.php* aufgerufen.

Hat der Benutzer mit CANCEL die Autorisierung der Anwendung abgebrochen, erfolgt der Redirect nach folgendem Schema:

```
http://YOUR_URL?error_reason=user_denied&error=access_denied
&error_description=The+user+denied+your+request
```

Die von Facebook gesetzten HTTP-GET-Parameter `error_reason`, `error` und `error_description` geben dabei Auskunft, warum die Autorisierung fehlgeschlagen ist. Um dies zu testen, legen Sie im Verzeichnis `/var/www/login` Ihres Webservers die Datei *callback.php* mit folgendem Inhalt an:

```
<pre><? print_r($_REQUEST); ?></pre>
```

Listing 2.2 Der Facebook-OAuth-Dialog leitet bei erfolgreichem oder abgebrochenem Dialog zur Datei »callback.php« auf unserem Anwendungsserver weiter.

Rufen Sie nun *http://apps.mycompany.com/login* erneut auf, klicken Sie auf Log In, und brechen Sie den OAuth-Dialog anschließend mit Cancel ab. Sie erhalten folgende Ausgabe im Browser:

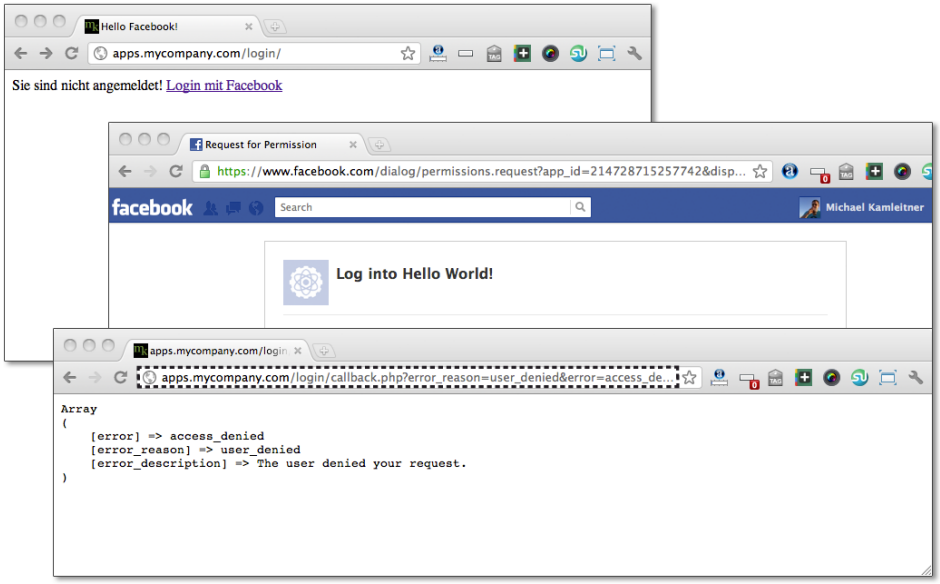


Abbildung 2.4 OAuth-Dialog und anschließende Weiterleitung nach Abbrechen des Dialogs. Hervorgehoben sind die von Facebook übergebenen GET-Parameter.

Ist der Benutzer mit der angefragten Autorisierung hingegen einverstanden und klickt im OAuth-Dialog auf Log In, geschieht Folgendes: Facebook betrachtet die Anwendung »Hello World!« von nun an als für diesen Benutzer autorisiert oder »installiert« – der OAuth-Dialog muss also nicht mehr durchlaufen werden, wenn ein Benutzer später die Anwendung erneut aufruft – und zwar so lange, bis der Benutzer die Anwendung entweder deinstalliert oder Sie als Entwickler zusätzliche Berechtigungen mittels `scope`-Parameter einfordern. In der Folge leitet Facebook den Browser des Benutzers mittels HTTP-Redirect weiter zur `redirect_uri`, wobei anstelle der Fehlerparameter nun ein GET-Parameter namens `code` gesetzt wird, der einen sogenann-

ten *Autorisierungscode* enthält. Probieren Sie es aus, und klicken Sie in Hello World! auf Log In:

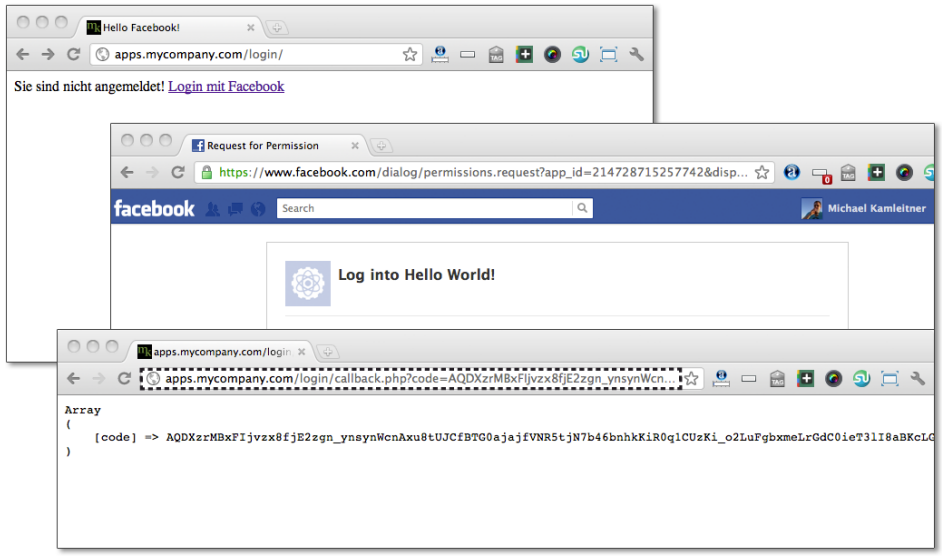


Abbildung 2.5 OAuth-Dialog und anschließende Weiterleitung nach erfolgreicher Autorisierung. Hervorgehoben sind die von Facebook übergebenen GET-Parameter.

Achtung: Der Autorisierungscode ermöglicht es Ihnen noch nicht, innerhalb von Facebook.com im Namen des Benutzers zu handeln! Vielmehr muss der Autorisierungscode in einem zweiten Schritt in ein sogenanntes *Access Token* umgewandelt werden.

2.2.3 Bezug des Access Tokens

Dazu müssen Sie den Autorisierungscode gemeinsam mit der Applikations-ID und dem App Secret folgendermaßen an die Graph API von Facebook übermitteln:

```
https://graph.facebook.com/oauth/access_token?
client_id=YOUR_APP_ID&redirect_uri=YOUR_URL&
client_secret=YOUR_APP_SECRET&code=THE_CODE_FROM_ABOVE
```

Hinweis: Der Abruf des Access Tokens erfolgt dabei nicht als HTTP-Redirect, sondern als normaler GET-Zugriff, etwa über die PHP-Funktion `file_get_contents()` oder `curl()`.

Sind alle Parameter korrekt übergeben worden, antwortet Facebook mit einem gültigen Access Token, das es der Anwendung fortan erlaubt, im Namen des Benutzers auf Facebook zuzugreifen:

Abbildung 2.6 Das Access Token wird für den Benutzer ausgestellt.

Neben dem eigentlichen Token enthält die Antwort in `expires` auch die Dauer der Gültigkeit des Tokens in Sekunden.

Erweitern Sie nun die Datei *callback.php* um den Code, der notwendig ist, um das Access Token zu beziehen. Danach verwenden Sie das Access Token, um über die Graph API Informationen über den eigenen Benutzer zu lesen:

```
<?
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
define('SITE_URL', 'http://apps.mycompany.com');
?>
<DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
<? if (!empty($_REQUEST["code"])) {
   $url = "https://graph.facebook.com/oauth/access_token".
"?client_id=".APP_ID.
"&redirect_uri=".urlencode(SITE_URL."/login/callback.php").
"&client_secret=".APP_SECRET."&code=".$_REQUEST["code"];

   $access_token = curl($url);
   $access_token =
substr($access_token,0,strpos($access_token,"&"));
?>
   <h1>Erfolgreich autorisiert!</h1>
   <p>Dein Access Token: <?= $access_token ?></p>
   <?
   $url = "https://graph.facebook.com/me?".$access_token;
   $me = curl($url);
   ?>
```

```
   <pre><? print_r(json_decode($me)); ?></pre>
<? } else { ?>
   <h1>Fehler beim Autorisieren!</h1>
   <pre><? print_r($_REQUEST); ?></pre>
<? } ?>
</body>
</html>

<?
function curl($url, $postargs=false, $method=null) {
   $ch = curl_init($url);
   if ($postargs !== false) {
      curl_setopt ($ch, CURLOPT_POST, true);
      curl_setopt ($ch, CURLOPT_POSTFIELDS, $postargs);
   }
   if ($method)
     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, strtoupper($method));
   curl_setopt($ch, CURLOPT_VERBOSE, 0);
   curl_setopt($ch, CURLOPT_HEADER, 0);
   curl_setopt($ch, CURLOPT_FOLLOWLOCATION,1);
   curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
   $response = curl_exec($ch);
   $responseInfo=curl_getinfo($ch);
   curl_close($ch);
   if( intval( $responseInfo['http_code'] ) == 200 )
       return $response;
   else
      return false;
}
?>
```

Listing 2.3 Die Datei »callback.php« mit Abruf des Access Tokens

Hinweise zum Code:

- *callback.php* prüft zuerst, ob der GET-Parameter `code` gesetzt ist. Wenn ja, bedeutet das, dass der OAuth-Dialog erfolgreich bestätigt wurde. Ist `code` nicht gesetzt, bedeutet dies, dass der Benutzer den OAuth-Dialog abgebrochen hat.
- Mit dem `code` kann anschließend – gemeinsam mit Applikations-ID, -Secret und Redirect URI – eine Anfrage an die Graph API gestellt werden. Diese Anfrage ist *kein Redirect*, sondern ein normaler Request über HTTP GET! Wichtig ist dabei, dass der Parameter `redirect_uri` die gleiche URL enthält wie beim vorangegangenen Redirect zum OAuth-Dialog!

- Das Ergebnis dieses GET-Requests wird in `$access_token` gespeichert und enthält neben dem eigentlich Token auch die `expires`-Angabe. Das Ergebnis wird mit `strpos` getrennt und enthält nur das Access Token mit dem Präfix `access_token=`.
- In der Folge wird ein weiterer HTTP-GET-Request an Facebook gesendet: Der Graph-API-Pfad `/me` liefert Informationen zum eigenen Benutzerobjekt und wird mit dem Access Token »signiert«.
- Die Graph API liefert ihre Ergebnisse üblicherweise in JSON-Codierung. Zur übersichtlichen Ausgabe in PHP wird daher zuvor die PHP-Funktion `json_decode()` verwendet.

Im Code-Beispiel werden alle HTTP-GET-Requests mit einer Hilfsfunktion namens `curl()` realisiert. Die Bibliothek *Curl* erlaubt die einfache Umsetzung von HTTP-Zugriffen aus PHP und ist in den meisten PHP-Installationen verfügbar. Die alternative Verwendung der Funktion `file_get_contests($url)` würde auf manchen Hosting-Umgebungen zu Problemen führen, da hier oft der Zugriff auf externe URLs deaktiviert ist. Da die Hilfsfunktion `curl()` in den folgenden Code-Beispielen immer wieder verwendet wird, kann sie in eine Hilfsbibliothek *tools.php* mit folgendem Inhalt ausgelagert werden:

```
<?
//
// tools.php - eine Sammlung an nützlichen Funktionen für das Buch
// Facebook-Awendungsentwicklung - Michael Kamleitner
//
function curl($url, $postargs=false, $method=null) {
   $ch = curl_init($url);
   if ($postargs !== false) {
      curl_setopt ($ch, CURLOPT_POST, true);
      curl_setopt ($ch, CURLOPT_POSTFIELDS, $postargs);
   }
   if ($method)
     curl_setopt($ch, CURLOPT_CUSTOMREQUEST, strtoupper($method));
   curl_setopt($ch, CURLOPT_VERBOSE, 0);
   curl_setopt($ch, CURLOPT_HEADER, 0);
   curl_setopt($ch, CURLOPT_FOLLOWLOCATION,1);
   curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
   $response = curl_exec($ch);
   $responseInfo=curl_getinfo($ch);
   curl_close($ch);
   if( intval( $responseInfo['http_code'] ) == 200 )
       return $response;
   else
```

```
        return false;
}
?>
```

Listing 2.4 Die Hilfsbibliothek »tools.php«

In den folgenden Code-Beispielen wird die Hilfsbibliothek fortan immer mit folgender Zeile eingebunden:

```
include_once('tools.php');
```

Und so sieht das Ergebnis im Browser aus, nachdem der OAuth-Dialog mit Log in bestätigt wurde:

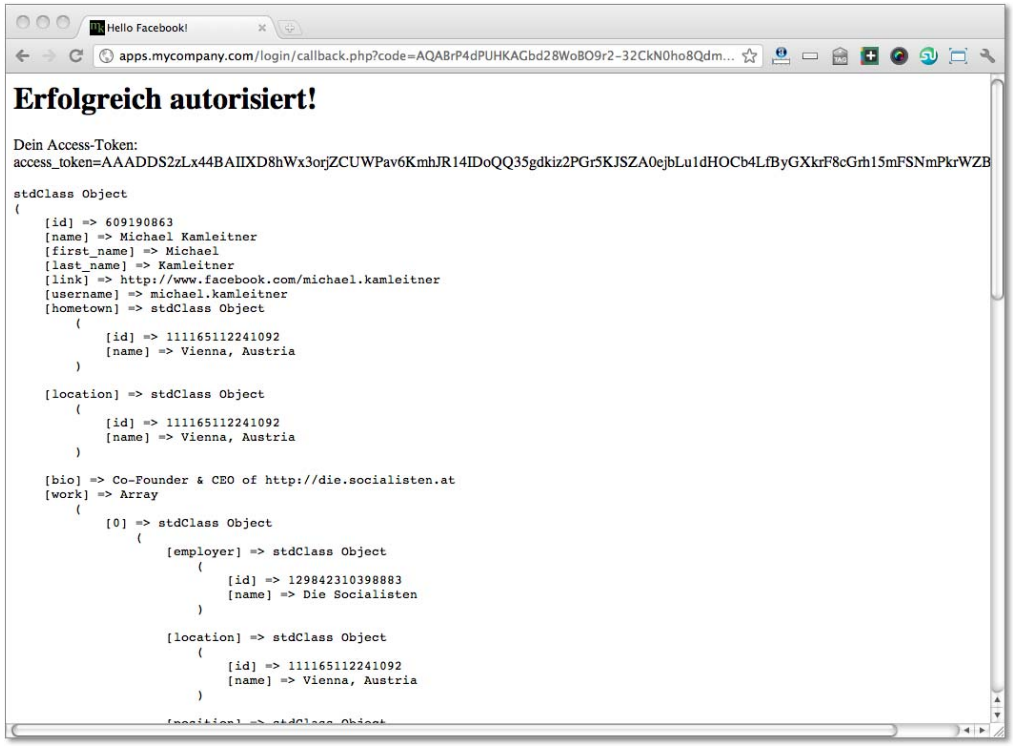


Abbildung 2.7 Erfolgreich bezogenes Access Token und anschließender Zugriff auf den Graph-API-Pfad »/me«

Wie Sie anhand der Testausgabe von *callback.php* erkennen können, hat Facebook auf die Anfrage ein gültiges Access Token an Ihre Anwendung übermittelt. Damit konnten Sie anschließend erfolgreich das eigene Benutzerobjekt auslesen und sich im Browser anzeigen lassen. Der Benutzer kann somit in unserer Anwendung als authentifiziert gelten, die Facebook-Anwendung ist erfolgreich autorisiert worden.

2.2.4 Erweiterung der Authentifikation um Session-Management

In einem nächsten Schritt ist es sinnvoll, das bezogene Access Token in der Session-Variablen des Benutzers zu speichern. So kann es bei späteren Zugriffen auf die Graph API jederzeit einfach wiederverwendet werden.

Session-Management in PHP-Anwendungen

Das *Session-Management* ist eine Grundfunktion aller Programmiersprachen, die zur Entwicklung von Webanwendungen geeignet sind. Sie lösen ein grundlegendes Problem: Ein Benutzer ruft bei der Verwendung einer Webanwendung üblicherweise mehrere HTTP-GET-Requests nacheinander auf – etwa wenn er sich von Seite zu Seite durchklickt. Jeder GET-Request wird von einem eigenständigen Prozess des Webservers beantwortet, was zur Folge hat, dass diese Prozesse nichts von ihren Vorgängern oder Nachfolgern »wissen«. Es gibt also per se keine Möglichkeit, Daten zwischen den einzelnen GET-Requests und den Prozessen, die sie beantworten, gemeinsam zu nutzen. Oft ist aber gerade das notwendig, etwa um in einem mehrseitigen Formular auf die Eingaben einer vorangegangenen Seite zugreifen zu können.

Um dieses Problem zu lösen, implementiert PHP einen Mechanismus zum sogenannten *Session-Management*. Dieser Mechanismus erlaubt es, beliebige Daten in der reservierten Variablen `$_SESSION` zu speichern und so von Request zu Request zu übergeben. Technisch wird das Session-Management üblicherweise mit Cookies auf Clientseite und mit temporären Session-Dateien auf Serverseite gelöst. Durch das Cookie, das der Browser bei jedem Request mitsendet, kann der Webserver die einzelnen Requests einem Benutzer bzw. dessen Session zuordnen. Die in `$_SESSION` gespeicherten Daten werden dabei zwischen den Requests in einer temporären Datei am Server gespeichert und jedem weiteren Request wieder automatisch zur Verfügung gestellt.

Der Zugriff und die Steuerung der Session erfolgen über folgende Methoden und Variablen:

- `session_start()` – Diese Methode startet explizit eine Session in einem PHP-Skript. Aktuelle PHP-Versionen rufen diese Methode implizit automatisch am Beginn der Ausführung des Skripts auf, weshalb ein expliziter Aufruf meist nicht notwendig ist.
- `session_destroy()`– Diese Methode zerstört die aktuelle Session und die in ihr gespeicherten Daten.
- `$_SESSION` – Diese Variable enthält – meistens als mehrdimensionales Array – alle Daten, die in der Session gespeichert sind. Lesender und schreibender Zugriff erfolgen so, wie von PHP gewohnt: `$_SESSION["Config"]["user_id"] = 47110815;`

Mehr über das Session-Management von PHP erfahren Sie unter *http://www.php.net/manual/en/book.session.php*.

Für die Implementierung eines einfachen Session-Managements erweitern Sie zuerst die Datei `/var/www/login/callback.php` folgendermaßen:

```
<?
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
define('SITE_URL', 'http://apps.mycompany.com');

if (!empty($_REQUEST["code"])) {
   $url = "https://graph.facebook.com/oauth/access_token".
"?client_id=".APP_ID.
"&redirect_uri=".urlencode(SITE_URL."/login/callback.php").
"&client_secret=".APP_SECRET.
"&code=".$_REQUEST["code"];

   $access_token = curl($url);
   $access_token = substr($access_token,0,strpos($access_token,"&"));
   $url = "https://graph.facebook.com/me?".$access_token;
   $me = json_decode(curl($url));
   session_start();
   $_SESSION["access_token"] = $access_token;
   $_SESSION["user"] = $me;
   header("Location: ./index.php");
   ?>
<? } else { ?>
   <DOCTYPE html>
   <html lang="de-de"
         xmlns:fb="http://www.facebook.com/2008/fbml">
   <head>
      <meta charset="utf-8">
      <title>Hello Facebook!</title>
   </head>
   <body>
      <h1>Fehler beim Autorisieren!</h1>
      <pre><? print_r($_REQUEST); ?></pre>
   </body>
   </html>
<? } ?>
```

Listing 2.5 Um das Session-Management erweiterte Datei »callback.php«

Hinweise zum Code:

- Der grundlegende Ablauf bleibt unverändert – nach der erfolgreichen Prüfung, ob der Parameter `code` übergeben wurde, wird das Access Token für den Benutzer bezogen. Danach wird das Access Token verwendet, um mehr über den angemeldeten Benutzer zu erfahren und in `$me` zu speichern.
- Danach wird mit `session_start()`eine explizite Session gestartet.
- In der Session-Variablen werden anschließend das Access Token (`$_SESSION["access_token"] = $access_token;`) und das komplette User-Objekt (`$_SESSION["me"] = $me;`) gespeichert. Alle in der Session-Variablen gespeicherten Daten werden dank Session-Management in künftigen Requests dieses Benutzers verfügbar sein!
- Danach wird der Benutzer mittels `header("Location: ./index.php");` zur Startseite Ihrer Anwendung weitergeleitet.

Das Skript */var/www/login/index.php* muss nun ebenfalls auf das Session-Management vorbereitet werden:

```
<?
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
define('SITE_URL', 'http://apps.mycompany.com');
$login_url = "http://www.facebook.com/dialog/oauth".
"?client_id=".APP_ID.
"&redirect_uri=".SITE_URL."/login/callback.php";
session_start();
?>
<DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
<? if (@$_SESSION["access_token"]) { ?>
   Sie sind angemeldet als <?= @$_SESSION["me"]->name ?>!
   <a href="logout.php">Logout</a>.
   <pre><? print_r($_SESSION); ?></pre>
<? } else { ?>
   Sie sind nicht angemeldet!
   <a href="<?= $login_url ?>">Login mit Facebook</a>.
   <pre><? print_r($_SESSION); ?></pre>
```

```
<? } ?>
</body>
</html>
```

Listing 2.6 Um das Session-Management erweiterte Datei »index.php«

Hinweise zum Code:

- Mit `session_start()` wird zu Beginn des Skripts explizit eine Session begonnen (da dies in aktuellen PHP-Installationen meist implizit erfolgt, können Sie diese Zeile normalerweise weglassen – ein expliziter Start der Session schadet aber auch nicht).
- Danach wird geprüft, ob die Session-Variable `$_SESSION["access_token"]` existiert – wenn ja, hat sich der Benutzer offenbar bereits authentifiziert und kann somit als angemeldet betrachtet werden! Wenn nicht, wird dem Benutzer der bekannte Login-Link zum Facebook-OAuth-Dialog angeboten.
- Wenn der Benutzer angemeldet ist, wird ebenfalls der in der Session-Variablen unter `$_SESSION["name"]` gespeicherte Name des Benutzers ausgegeben.

Um dem Benutzer auch das Abmelden zu ermöglichen, legen Sie eine Datei */var/www/login/logout.php* mit folgendem Inhalt an:

```
<?
session_start();
session_unset();
header("Location: ./index.php");
?>
```

Listing 2.7 Löschen der Session-Variablen mit der Datei »logout.php«

Hinweise zum Code:

- Zum Abmelden des Benutzers müssen Sie nach dem obligatorischen `session_start()` lediglich dafür sorgen, dass mittels `session_unset()` alle in der Session gespeicherten Daten gelöscht werden.
- Danach wird der Benutzer auf die Startseite unserer Anwendung zurückgeleitet – da dort die Session-Daten nicht mehr verfügbar sind, erscheint wieder der Login-Link zum OAuth-Dialog.

Das Ergebnis Ihrer Bemühungen ist eine einfache Webanwendung, an der sich Benutzer mit ihrem Facebook-Konto an- und abmelden können:

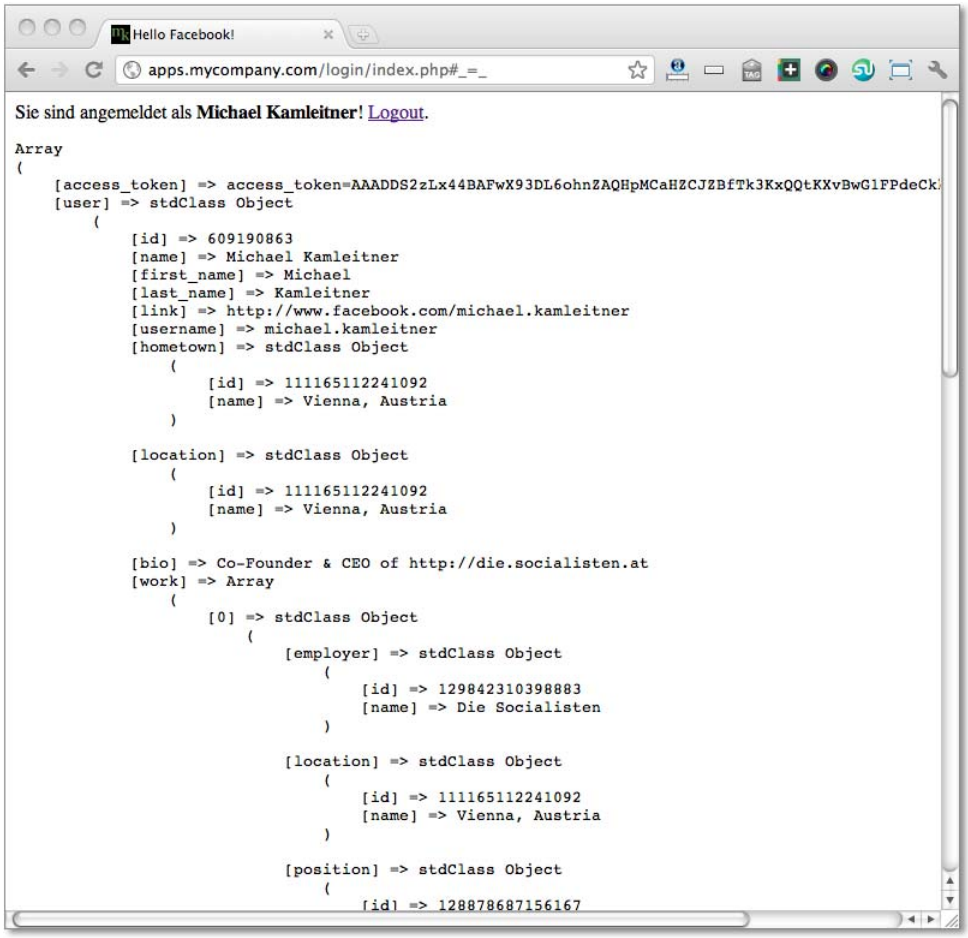


Abbildung 2.8 Die um das Session-Management erweiterte Anwendung

Sicherheit im OAuth-Ablauf

Um die Verwendung von OAuth sicher zu gestalten, hat Facebook mehrere Prüfmechanismen implementiert:

- *Prüfung der Site URL:* Es ist wichtig, dass die im Parameter `redirect_uri` angegebene Callback-URL innerhalb der gleichen Domain liegt, die als Site URL in den Einstellungen der Anwendung angegeben wurde. Stimmen die Domains nicht überein, bricht Facebook den OAuth-Dialog mit einer entsprechenden Fehlermeldung ab. Dies ist wichtig, da die öffentlich bekannte Applikations-ID ansonsten missbraucht werden könnte, um den Benutzer zum OAuth-Dialog einer fremden Anwendung zu lenken.

- *Prüfung des App Secrets:* Das App Secret muss beim Beziehen des Access Tokens gemeinsam mit der Applikations-ID und dem Autorisierungscode an Facebook übergeben werden. Da dieser Request serverseitig und nicht etwa per Redirect im Browser des Benutzers erfolgt, bleibt das App Secret für den Benutzer unsichtbar – und damit geheim.
- *Verwendung von SSL:* Zugriffe auf die Autorisierungsfunktionen der Graph API dürfen ausschließlich über das verschlüsselte HTTPS-Protokoll erfolgen.

2.3 Clientseitige Authentifikation

Manchmal ist es nicht notwendig, das Access Token des Benutzers im serverseitigen Code der Anwendung zu kennen, sondern es genügt ein clientseitiger Zugriff mittels JavaScript im Browser. Dies kann sinnvoll sein, wenn Zugriffe auf die Graph API lediglich im Browser erfolgen sollen, aber keine serverseitige Anwendungslogik vorgesehen ist.

Der bereits bekannte Redirect zum OAuth-Dialog von Facebook läuft dabei prinzipiell gleich ab, jedoch wird mit dem zusätzlichen Parameter `response_type=token` angegeben, dass die clientseitige Authentifikation genutzt werden soll:

```
https://www.facebook.com/dialog/oauth?client_id=YOUR_APP_ID
&redirect_uri=YOUR_URL&scope=...&response_type=token
```

Bestätigt der Benutzer den OAuth-Dialog, leitet Facebook den Browser auf die in `redirect_uri` angegebene URL weiter. Diesmal wird aber nicht der Autorisierungscode im GET-Parameter `code` übergeben, sondern gleich das Access Token (und dessen Gültigkeitsdauer) in einem sogenannten *URI-Fragment*, also jenem Teil einer URL, der nach dem Hash-Symbol (»#«) folgt:

```
http://YOUR_URL#access_token=166942940015970%7C2.sa0
&expires_in=64090
```

Der Inhalt des URI-Fragments kann ausschließlich per JavaScript im Browser ausgelesen und weiterverwendet werden, etwa um Aufrufe der Graph API per JavaScript SDK mit dem Access Token zu unterzeichnen.

Die Sicherheit des OAuth-Ablaufs wird auch bei der clientseitigen Authentifikation gewährleistet, indem die `redirect_uri` auf Übereinstimmung mit der im Developer angegebenen SITE URL geprüft wird.

Um die clientseitige Authentifikation auszuprobieren, legen Sie auf Ihrem Webserver eine neue Datei unter */var/www/login/index.client.php* an:

```
<?
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
define('SITE_URL', 'http://apps.mycompany.com');
$login_url = "http://www.facebook.com/dialog/oauth".
"?client_id=".APP_ID.
"&response_type=token".
"&redirect_uri=".SITE_URL."/login/index.client.php";
?>
<DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
   <script type="text/javascript">
   if (window.location.hash.length == 0) {
      document.write('Sie sind nicht angemeldet!
      <a href="<?= $login_url ?>">Login mit Facebook</a>.');
   } else {
      var accessToken = window.location.hash.substring(1);
      var path = "https://graph.facebook.com/me?";
      var queryParams = [accessToken, 'callback=displayUser'];
      var query = queryParams.join('&');
      var url = path + query;
      var script = document.createElement('script');
      script.src = url;
      document.body.appendChild(script);
   }
   function displayUser(user) {
      var p = document.createElement('p');
      p.innerHTML = 'Sie sind angemeldet als
                     <b>'+user.name+'</b>!';
      document.body.appendChild(p);
   }
   </script>
</body>
</html>
```

Listing 2.8 Clientseitige, JavaScript-basierte Authentifikation

Hinweise zum Code:

- Anders als die bisher genutzte serverseitige Authentifikation wird für die clientseitige Authentifikation kein Callback-Skript benötigt – die Logik wird ja vollständig im JavaScript-Code von *index.client.php* realisiert.
- Per JavaScript wird nach dem Aufrufen der Seite per `if (window.location.hash.length == 0)` geprüft, ob das URI-Fragment nach dem Hash-Zeichen bereits gesetzt ist. Wenn nicht, ist der Benutzer nicht angemeldet, und es wird mit `document.write` ein entsprechender Login-Link ausgegeben, der wie üblich auf den Facebook-OAuth-Dialog verweist.
- Der OAuth-Dialog wird wie bisher aufgerufen, jedoch wird mittels `response_type=token` die clientseitige Authentifikation aktiviert. Als `redirect_uri` wird nicht das Callback-Skript angegeben, sondern das Index-Skript selbst.
- Nach dem Bestätigen des OAuth-Dialogs leitet Facebook den Benutzer zurück zum Index-Skript, dabei wird das Access Token als URI-Fragment übergeben.
- Wird im JavaScript-Code nach dem Neuladen der Index-Seite festgestellt, dass das URI-Fragment gesetzt ist, wird in der Variablen `url` ein Graph-API-Zugriff der Form *https://graph.facebook.com/me?access_token=...* vorbereitet und mittels `<script>`-Tag geladen. In der Callback-Funktion `displayUser` wird anschließend auf das Ergebnis des Graph-API-Requests zugegriffen und der Name des authentifizierten Benutzers ausgegeben.

2.4 Signed Request

Die bisherigen Programmierbeispiele sind meist von einer einfachen Anwendung ausgegangen, die außerhalb der Umgebung von Facebook.com, also auf einer externen Webseite, aufgerufen wurde. Für Canvas- und Tab-Anwendungen sieht Facebook aber eine Besonderheit vor – den *Signed Request*. Mit diesem Mechanismus übergibt Facebook allen Anwendungen, die innerhalb von Facebook.com aufgerufen werden, bestimmte Informationen zum aktuellen Benutzer. Um dem Signed Request auf die Spur zu kommen, kann das Skript *index.php* aus Abschnitt 1.5, »Hello Facebook!«, weiterverwendet und um folgende Änderungen ergänzt werden:

```
<!DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
   <h1>Hello Facebook!</h1>
```

```
    <p>Willkommen bei meiner ersten Facebook-Anwendung!</p>
    <pre><? print_r($_REQUEST); ?></pre>
</body>
</html>
```

Listing 2.9 Auf der Spur des Signed Requests

Die Funktion `print_r($_REQUEST)` tut dabei nichts anderes, als alle etwaigen GET- oder POST-Parameter, die an das Skript übergeben wurden, auszugeben.

Wenn Sie die so modifizierte *index.php* über *http://apps.facebook.com/NAMESPACE* oder das vorhin angelegte Tab aufrufen, offenbart sich der GET-Parameter `signed_request`, den Facebook beim Abrufen des iFrames über die in den Anwendungseinstellungen definierte CANVAS URL unserem Anwendungsserver automatisch hinzufügt:

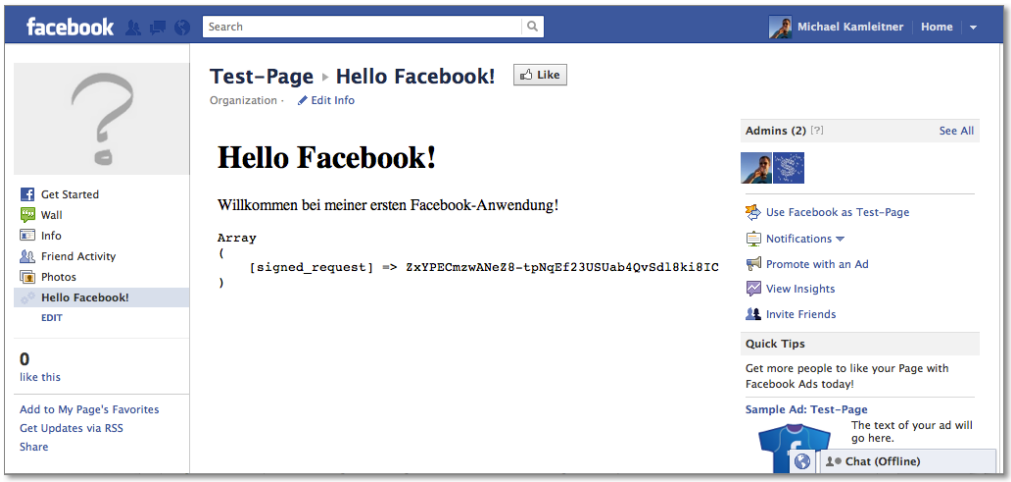


Abbildung 2.9 Der von Facebook an den iFrame der Anwendung übergebene Parameter »signed_request«

Der `signed_request`-Parameter besteht aus einer durch einen Punkt (».«) verbundenen HMAC-SHA-256-codierten Signatur und einem base64url-codierten JSON-Objekt:

```
vlXgu64BQGFSQrYOZcJBZASMvYvTHu9GQOYM9rjPSso.
eyJhbGdvcmlOaGOiOiJITUFDLVNIQTI1NiIsIjAiOiJwYXlsb2FkInO
```

Die HMAC-Signatur wird von Facebook mit dem App Secret verschlüsselt, womit Sie sicherstellen können, dass Facebook tatsächlich der Absender des Requests ist. Facebook stellt eine praktische Funktion `parse_signed_request()` zum Entschlüsseln des `signed_request`-Parameters bereit, die in die Datei *index.php* eingebaut wird:

```
<?
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
?>
<!DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
   <h1>Hello Facebook!</h1>
   <p>Willkommen bei meiner ersten Facebook-Anwendung!</p>
   <pre><? print_r($_REQUEST); ?></pre>
   <pre><? print_r(
   parse_signed_request($_REQUEST["signed_request"], APP_SECRET));
   ?>
</body>
</html>

<?
function parse_signed_request($signed_request, $secret) {
  list($encoded_sig, $payload) = explode('.', $signed_request, 2);

  // decode the data
  $sig = base64_url_decode($encoded_sig);
  $data = json_decode(base64_url_decode($payload), true);

  if (strtoupper($data['algorithm']) !== 'HMAC-SHA256') {
    error_log('Unknown algorithm. Expected HMAC-SHA256');
    return null;
  }

  // check sig
  $expected_sig = hash_hmac('sha256', $payload, $secret,
                            $raw = true);
  if ($sig !== $expected_sig) {
    error_log('Bad Signed JSON signature!');
    return null;
  }
  return $data;
}
```

```
function base64_url_decode($input) {
  return base64_decode(strtr($input, '-_', '+/'));
}
?>
```

Listing 2.10 Entschlüsseln des Signed Requests mit dem App Secret

Das Entschlüsseln des Signed Requests ergibt ein Array mit zahlreichen Informationen zum aufrufenden Benutzer:

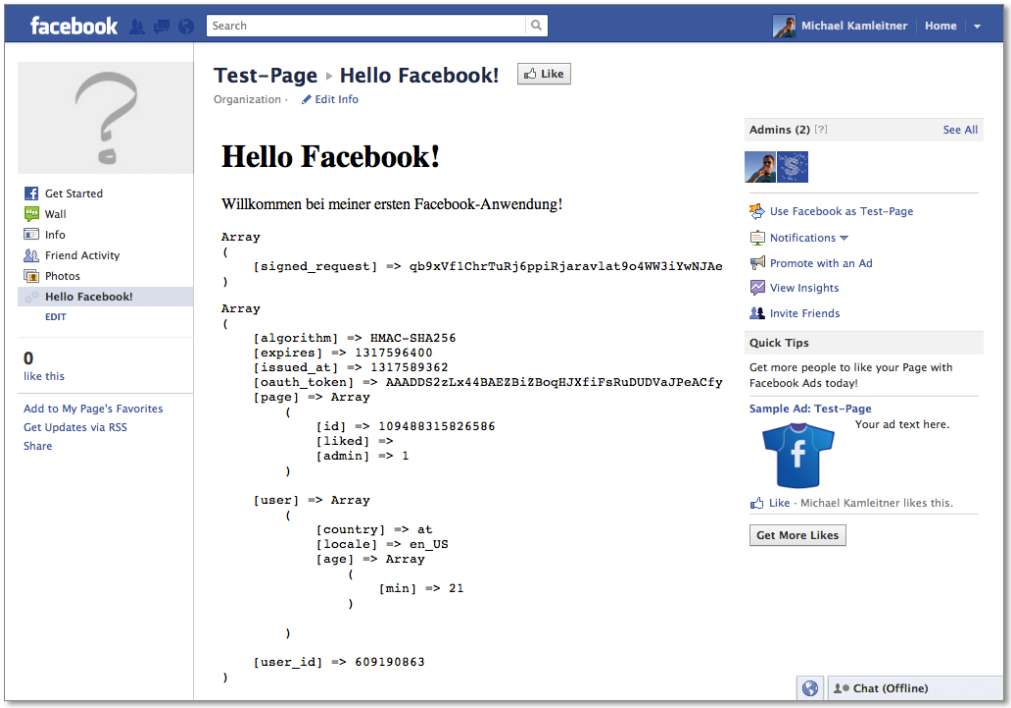


Abbildung 2.10 Der entschlüsselte Signed-Request-Parameter

Variable	Feld	Beschreibung
`algorithm`	–	Codierungsschema, mit dem der Signed Request verschlüsselt wurde – immer HMAC-SHA256
`issued_at`	–	Zeitstempel, an dem der Signed Request übergeben wurde

Tabelle 2.1 Informationen, die der Signed Request bereitstellt

Variable	Feld	Beschreibung
`oauth_token`	–	Access Token des aktuellen Benutzers – nur gesetzt, wenn der Benutzer die Anwendung bereits autorisiert hat!
`expires`	–	Zeitstempel, bis zu dem das Access Token gültig ist – nur gesetzt, wenn der Benutzer die Anwendung bereits autorisiert hat!
`page`	–	Informationen über die Page, innerhalb der die Anwendung aufgerufen wurde – nur gesetzt bei Zugriff als Tab-Anwendung
`page`	`id`	die numerische ID der Page
`page`	`liked`	1, wenn der Benutzer Fan der Page ist – nur gesetzt, wenn das Canvas innerhalb eines Tabs/Reiters aufgerufen wurde
`page`	`admin`	1, wenn der Benutzer Administrator der Page ist – nur gesetzt, wenn das Canvas innerhalb eines Tabs/Reiters aufgerufen wurde
`user`	–	Informationen über den aktuellen Benutzer
`user`	`country`	Herkunftsland des Benutzers
`user`	`locale`	Spracheinstellung des Benutzers
`user`	`age - min`	Mindestalter des Benutzers
`user_id`	–	numerische ID des Bentuzers – nur gesetzt, wenn der Benutzer die Anwendung bereits autorisiert hat!

Tabelle 2.1 Informationen, die der Signed Request bereitstellt (Forts.)

Hinweis: Besonders interessant ist, dass für Benutzer, die die Anwendung bereits autorisiert haben, im Parameter `oauth_token` ein gültiges Access Token und in der Struktur `user` verschiedene Informationen über das Benutzerkonto – insbesondere die Benutzer-ID – übergeben werden. Damit muss der Anwender nach dem Laden der Canvas-Anwendung nicht mehr den Umweg über den Aufruf des OAuth-Dialogs nehmen, um authentifiziert zu werden!

Die Informationen, die Sie als Anwendungsentwickler durch den Signed Request erhalten, können in vielfacher Weise genutzt werden – ein klassisches Beispiel ist die Implementierung eines Fan-Gates.

2.4.1 Implementierung eines Fan-Gates

Unter einem *Fan-Gate* versteht man eine als Tab in ein Seitenprofil integrierte Anwendung, die Benutzern unterschiedliche Inhalte zeigt – je nachdem, ob der Benutzer bereits Fan der Seite ist oder nicht. Beispiele sind etwa der Teilnahmelink zu einem Gewinnspiel oder die Darstellung eines Coupon-Codes für Fans der Seite. Fan-Gates sind damit ein beliebtes und effektives Mittel zur Fan-Generierung für Seitenbetreiber, Unternehmen und Marken.

Da Tab-Anwendungen oft nur statische Inhalte anzeigen und meist keine Autorisierung durch den Benutzer erfordern (immerhin sollen alle Benutzer, die das Tab aufrufen, Inhalte sehen, nicht bloß jene, die bereit sind, die Anwendung zu installieren), ist die effektivste Möglichkeit, ein Fan-Gate zu realisieren, die Nutzung des Signed Requests. Über den darin enthaltenen Wert `["page"][ "liked"]` können Sie nach Entschlüsselung des Signed Requests entscheiden, welche Inhalte angezeigt werden sollen. Das folgende Programmierbeispiel legen Sie unter */var/www/fangate/index.php* an. Fügen Sie dazu auch in der Datei *tools.php* die Funktionen `parse_signed_request()` und `base64_url_decode()` zum Entschlüsseln des Signed Requests hinzu – diese stehen fortan auch in folgenden Beispielen durch Einbinden dieser Datei zur Verfügung:

```
<?
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************************');
$signed_request =
parse_signed_request($_REQUEST["signed_request"], APP_SECRET);
?>
<!DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook!</title>
</head>
<body>
   <h1>Welcome to our tab!</h1>
   <? if ($signed_request["page"]["liked"] == 1) { ?>
      <p>Danke, dass du Fan geworden bist! Der Coupon-Code lautet:
      <b>FACEBOOK2012</b>!
```

```
    <? } else { ?>
      <p>Bitte <b>werde Fan unserer Seite</b>, um einen Coupon-Code
      für ein kostenloses Exemplar dieses Buches zu erhalten! Klicke
      dazu oben auf "Gefällt mir"!</p>
    <? } ?>
</body>
</html>
```

Listing 2.11 Implementierung eines einfachen Fan-Gates

Bevor Sie die Tab-Anwendung über Ihre Seite aufrufen, vergessen Sie nicht, in den Anwendungseinstellungen die PAGE TAB URL an die neue URL *http://apps.mycompany.com/fangate/* anzupassen! Danach können Sie Ihr Fan-Gate auf der Page testen. Je nachdem, ob Sie bereits auf GEFÄLLT MIR geklickt haben oder nicht, werden unterschiedliche Inhalte dargestellt. Praktisch dabei: Klickt ein Benutzer, der noch nicht Fan ist, auf den LIKE-Button, lädt Facebook die gesamte Page – damit wird auch der Inhalt der Tab-Anwendung für Fans angezeigt.

2.5 Deautorisierung von Anwendungen

Wie bereits in Abschnitt 1.4.3, »Einstellungen »Advanced«, erwähnt, ist es für Sie als Anwendungsentwickler wichtig, zu erfahren, wenn ein Benutzer eine Anwendung deinstalliert und damit die Autorisierung der Anwendung aufhebt. Damit verliert die Anwendung die Möglichkeit, auf die Informationen des Benutzers zuzugreifen. Dies erledigt Facebook automatisch – dabei verliert auch ein eventuell noch existierendes und in Verwendung befindliches Access Token des Benutzers an Gültigkeit (dies ist vor allem bei einem permanent gültigen Access Token, das mit der Berechtigung `offline_access` erlangt wurde, relevant). Auf der Seite der Anwendung müssen Sie als Entwickler dafür Sorge tragen, dass alle die Privatsphäre eines Benutzers betreffenden Anwendungsdaten aus Ihrer Datenbank entfernt werden. In der Praxis müssen Sie dazu meist den Benutzerdatensatz sowie alle vom Benutzer kreierten Daten löschen.

Um von der Deinstallation unserer Anwendung durch einen Benutzer zu erfahren, muss die DEAUTHORIZATION CALLBACK URL in den Einstellungen der Anwendung unter ADVANCED angegeben werden. Facebook ruft im Hintergrund diese URL jedes Mal auf, wenn ein Benutzer Ihrer Anwendung die Autorisierung entzieht. Der Aufruf dieser URL erfolgt mit einem `signed_request`-Parameter, der, wie vorhin beschrieben, decodiert werden kann, um die Benutzer-ID des jeweiligen Benutzers zu ermitteln.

2.6 Erweiterte Zugriffsrechte (»Extended Permissions«)

Anwendungsentwickler können mithilfe des `scope`-Parameters beim Aufruf des OAuth-Dialogs genau festlegen, welche Berechtigungen eine Anwendung bei der Autorisierung vom Benutzer einholt.

2.6.1 Basisberechtigungen

Fordert der Anwendungsentwickler keine zusätzlichen Berechtigungen vom Benutzer ein, hat die Anwendung nach erfolgter Autorisierung Zugriff auf folgende Basisprofildaten des Benutzers:

- Vor- und Nachname
- Profilfoto
- Liste der Freunde des Benutzers
- Heimatstadt und aktueller Aufenthaltsort
- Biographie
- Interessen

2.6.2 Berechtigungen für Profildaten von Benutzern und Freunden

Diese Berechtigungen regeln, welche Profilinformationen eine Anwendung entweder vom autorisierenden Benutzer oder dessen Freunden auslesen darf – die meisten Berechtigungen müssen also jeweils getrennt eingeholt werden. Der Anwendungsentwickler kann hierbei detailliert festlegen, welche Berechtigungen ein Benutzer zur erfolgreichen Autorisierung der Anwendung genehmigen muss. Derzeit gilt dabei »alles oder nichts« – ein Benutzer kann nicht selektiv entscheiden, welche Berechtigungen er einer bestimmten Anwendung gestatten möchte und welche nicht. Eine Überarbeitung des Autorisierungsdialogs, die dem Benutzer hier mehr Freiheiten einräumt, wurde aber von Facebook bereits angekündigt.

Benutzerberechtigung	Freunde-Berechtigung	Beschreibung
`user_about_me`	`friends_about_me`	Zugriff auf die Profilsektion Über mich über das Attribut `about`
`user_activities`	`friends_activities`	Zugriff auf die unter Aktivitäten verknüpften Objekte vom Typ `activities`

Tabelle 2.2 Berechtigungen für den Zugriff auf Profildaten des Benutzers und seiner Freunde

Benutzerberechtigung	Freunde-Berechtigung	Beschreibung
user_birthday	friends_birthday	Zugriff auf das Geburtsdatum über das Attribut birthday_date
user_checkins	friends_checkins	Lesezugriff auf Checkins des Benutzers
user_education_history	friends_education_history	Zugriff auf die Sektion Ausbildung über das Attribut education
user_events	friends_events	Zugriff auf die Veranstaltungen, die ein Benutzer besucht, verknüpfte Objekte vom Typ events
user_groups	friends_groups	Zugriff auf die Gruppen, in denen ein Benutzer Mitglied ist, verknüpfte Objekte vom Typ groups
user_hometown	friends_hometown	Zugriff auf die Heimatstadt eines Benutzers über das Attribut home_town
user_interests	friends_interests	Zugriff auf die Interessen eines Benutzers als verknüpfte Objekte vom Typ interests
user_likes	friends_likes	Zugriff auf die Liste der Seiten, auf denen der Benutzer auf Gefällt mir geklickt hat, als verknüpfte Objekte vom Typ likes
user_location	friends_location	Zugriff auf die Aktuelle Stadt eines Benutzers über das Attribut education
user_notes	friends_notes	Zugriff auf die Notizen eines Benutzers als verknüpfte Objekte vom Typ notes
user_online_presence	friends_online_presence	Zugriff auf den Online-/Offline-Status des Benutzers

Tabelle 2.2 Berechtigungen für den Zugriff auf Profildaten des Benutzers und seiner Freunde (Forts.)

Benutzerberechtigung	Freunde-Berechtigung	Beschreibung
user_photos	friends_photos	Zugriff auf die Fotos, die ein Benutzer hochgeladen hat oder in denen er markiert wurde
user_relationships	friends_relationships	Zugriff auf die Familienmitglieder eines Benutzers sowie seinen Beziehungsstatus
user_relationship_details	friends_relationship_details	Zugriff auf die Beziehungspräferenzen eines Benutzers
user_religion_politics	friends_religion_politics	Zugriff auf die religiösen und politischen Anschauungen eines Benutzers
user_status	friends_status	Zugriff auf die neueste Statusnachricht des Benutzers
user_videos	friends_videos	Zugriff auf die Videos, die ein Benutzer hochgeladen hat oder in denen er getagged wurde
user_website	friends_website	Zugriff auf die URL der Website eines Benutzers
user_work_history	friends_work_history	Zugriff auf den beruflichen Lebenslauf eines Benutzers über das Attribut work
email	–	Zugriff auf die E-Mail-Adresse des Benutzers

Tabelle 2.2 Berechtigungen für den Zugriff auf Profildaten des Benutzers und seiner Freunde (Forts.)

2.6.3 Erweiterte Berechtigungen

Neben den Berechtigungen für den Zugriff auf Profildaten von Benutzern und seiner Freunde stellt Facebook noch eine Palette an erweiterten Berechtigungen zur Verfügung. Diese regeln auch, welche »Schreibzugriffe« auf Facebook.com die autorisierte Anwendung im Namen des Benutzers durchführen darf.

Benutzerberechtigung	Beschreibung
read_friendlists	Zugriff auf die individuellen Freundeslisten des Benutzers (nicht zu verwechseln mit der Liste aller Freunde, die auch mit der Basisberechtigung verfügbar ist)
read_insights	Zugriff auf die Nutzungsstatistiken von Seiten, Anwendungen und Domains, die der Benutzer administriert
read_mailbox	Lesezugriff auf das Facebook-Nachrichten-Postfach des Benutzers
read_requests	Zugriff auf die Freundschaftsanfragen des Benutzers
read_stream	Lesezugriff auf den Newsfeed des Benutzers
xmpp_login	Login-Möglichkeit in den Facebook-Chat
ads_management	Verwaltung von Facebook Ads im Namen des Benutzers
create_event	Berechtigung zum Veröffentlichen von Veranstaltungen
manage_friendlists	Anlegen und Bearbeiten von Freundeslisten
manage_notifications	Berechtigung zum Lesen von Benachrichtigungen sowie zum Markieren von Benachrichtigungen als »gelesen«
manage_pages	Berechtigung, für die vom Benutzer administrierten Seiten Access-Tokens zu beziehen. Mit diesen Seiten-Access-Tokens kann im Namen der Seite auf Facebook veröffentlicht werden.
offline_access	Diese Berechigung bewirkt, dass das Access Token des Benutzers kein Ablaufdatum hat, sondern unbegrenzt gültig ist. Ein solches Token kann also auch verwendet werden, wenn der Benutzer die Anwendung in diesem Moment gar nicht nutzt – dies kann vor allem für wiederkehrende Hintergrundprozesse erforderlich sein.
publish_checkins	Berechtigung zur Veröffentlichung von Checkins
publish_stream	Berechtigung zur Veröffentlichung von Wall-Postings, Status-Updates und Kommentaren auf der Pinnwand des Benutzers. Hinweis: Diese Berechtigung ermöglicht das Veröffentlichen von Inhalten mithilfe eines Anwendungs-Access-Tokens.

Tabelle 2.3 Erweiterte Berechtigungen

Benutzerberechtigung	Beschreibung
`rsvp_event`	Berechtigung zum Veröffentlichen von Veranstaltungszusagen oder -absagen.
`publish_actions`	Berechtigung zum Veröffentlichen von Open-Graph-Aktionen und Objekten vom Typ `score` und `achievement`

Tabelle 2.3 Erweiterte Berechtigungen (Forts.)

Tipp: Je mehr Berechtigungen eine Anwendung vom Benutzer einfordert, umso höher ist die Wahrscheinlichkeit, dass der Benutzer die Autorisierung verweigert. Als Grundregel empfiehlt sich daher: nur das absolute Minimum an Berechtigungen einfordern! Oft macht es auch Sinn, erweiterte Berechtigungen erst zu einem späteren Zeitpunkt in der Benutzung der Anwendung einzuholen, nämlich erst dann, wenn diese tatsächlich benötigt werden.

Best-Practice-Beispiel

Geplant ist die Umsetzung eines Facebook-basierten Fotowettbewerbs. Teilnehmer sollen dabei die Möglichkeit haben, Fotos entweder von ihrer lokalen Festplatte hochzuladen oder aus einem vorhandenen Facebook-Album zu übernehmen. Gemäß den Plattform-Richtlinien von Facebook soll die Benachrichtigung der Gewinner per E-Mail erfolgen. Beim ersten Aufruf der Anwendung wird dem Benutzer auf einem Willkommensbildschirm der Fotowettbewerb vorgestellt – hierzu ist noch gar keine Anwendungsautorisierung notwendig, da es zu diesem Zeitpunkt noch nicht relevant ist, welcher Benutzer angemeldet ist. Die Identifikation des Benutzers durch seine Benutzer-ID kann also erst zu dem Zeitpunkt erfolgen, zu dem sich der Benutzer zur Teilnahme am Wettbewerb entscheidet und auf Mitmachen klickt. An dieser Stelle wird der Facebook-OAuth-Dialog mit dem Parameter `scope=email` aufgerufen – der Benutzer gibt also mit der Autorisierung der Anwendung Zugriff auf seine Basisprofilinformationen (Benutzer-ID, Name) sowie seine EMail-Adresse, die zur späteren Benachrichtigung der Gewinner benötigt wird. Beim nun angezeigten »Mitmachen«-Formular hat der Benutzer die Möglichkeit, ein Foto von seinem PC mittels Formular-Upload hochzuladen – in diesem Fall werden keine weiteren Berechtigungen benötigt. Klickt der Benutzer hingegen auf Foto von Facebook übernehmen, wird der OAuth-Dialog erneut mit dem Parameter `scope=user_photos` aufgerufen. Der Benutzer wird also aufgefordert, zusätzlich zu der bereits erteilten Berechtigung zum Zugriff auf die E-Mail-Adresse nun auch den Zugriff auf seine Facebook-Fotos und -Alben zu gewähren. Nach erfolgreicher Autorisierung können die Fotos des Benutzers per Graph API ausgelesen werden, und es kann ein entsprechender Auswahldialog angezeigt werden.

Wichtiger Link:

- *http://developers.facebook.com/docs/reference/api/permissions/* – Referenz aller verfügbaren erweiterten Zugriffsrechte

2.7 Access Tokens für Anwendungen und Seiten

Die bisher verwendeten Access Tokens wurden jeweils bezogen, indem ein Benutzer eine Anwendung und die damit geforderten Zugriffsrechte autorisiert hat. Die so erzeugten Access Tokens waren somit immer mit genau einem Benutzer – genauer: einer Anwendung – verknüpft. Als Anwendungsentwickler können Sie mit so einem Access Token Aktionen im Namen des Benutzers auf der autorisierten Anwendung durchführen (also z. B. ein Wall-Posting veröffentlichen oder ein Foto hochladen). Neben den benutzerbezogenen kennt die Facebook-Plattform aber noch zwei weitere Arten von Access Tokens.

2.7.1 Access Tokens für Anwendungen

Bestimmte Vorgänge auf der Facebook-Plattform, wie z. B. das Beziehen von Echtzeit-Updates, das Einlesen von Nutzungsstatistiken oder das Installieren eines Reiters auf einer Seite, benötigen ein Access Token, das nicht abhängig von einem bestimmten Benutzer, sondern auf die Applikation selbst ausgestellt ist. Um ein Anwendungs-Access-Token zu beziehen, genügt ein GET-Zugriff auf folgende URL der API:

```
https://www.facebook.com/oauth?client_id=YOUR_APP_ID
&redirect_uri=YOUR_URL&grant_type=client_credentials
```

Anders als bei den bisherigen Aufrufen wird durch den Parameter `grant_type=client_credentials` bewirkt, dass ein Anwendungs-Access-Token erzeugt wird. Das Ergebnis dieses Aufrufs enthält das Access Token mitsamt dem Parameternamen `access_token`:

```
access_token=214728715257742|UGP1R6ba_mO1ACO3Da6hLSAuWUO
```

Das Anwendungs-Access-Token ist leicht von einem benutzerbezogenen Token zu unterscheiden: Es ist deutlich kürzer und enthält im ersten Teil vor dem Pipe-Symbol die numerische ID der Applikation (»`214728715257742`«).

Das folgende Beispiel demonstriert das Beziehen eines Anwendungs-Access-Tokens in PHP:

```
<?
include_once("tools.php");
define('APP_ID', '214728715257742');
define('APP_SECRET', '********************');
define('SITE_URL', 'http://apps.mycompany.com');

print get_app_accesstoken(APP_ID, APP_SECRET);

function get_app_accesstoken($app_id, $app_secret) {
   $url =
   "https://graph.facebook.com/oauth/access_token?".
   "client_id=".$app_id.
   "&client_secret=".$app_secret.
   "&grant_type=client_credentials";

   $access_token = curl($url);
   $access_token =
      substr($access_token,strpos($access_token,"=")+1);
   return $access_token;
}
?>
```

Listing 2.12 Beziehen eines Anwendungs-Access-Tokens in PHP

Da die Funktion `get_app_accesstoken()` in späteren Beispielen noch öfter zum Einsatz kommen wird, übernehmen Sie bitte den Code in die Bibliothek *tools.php*!

2.7.2 Access Tokens für Seiten

Ähnlich wie für Anwendungen gibt es auch für Seiten spezielle Access Tokens, um Aktionen im Namen der Seite – etwa Wall-Postings oder Foto-Uploads – durchzuführen. Um ein Seiten-Access-Token zu beziehen, ist es notwendig, dass einer der Seitenadministratoren die Anwendung mit der Zugriffsberechtigung `manage_pages` autorisiert. Dazu wird wie üblich der Parameter `scope` beim Aufrufen des OAuth-Dialogs gesetzt:

```
https://www.facebook.com/dialog/oauth?client_id=YOUR_APP_ID
&redirect_uri=YOUR_URL&scope=manage_pages
```

Der Benutzer wird im OAuth-Dialog zur Autorisierung des entsprechenden Zugriffsrechts aufgefordert:

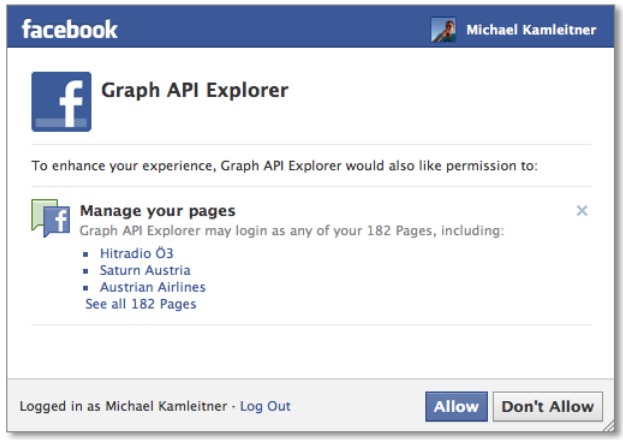


Abbildung 2.11 Autorisierung einer Anwendung mit der Berechtigung »manage_pages«

Hinweis: Facebook erlaubt es derzeit nicht, `manage_pages` für einzelne Seiten zu autorisieren. Der Benutzer muss also immer die Berechtigung zum Verwalten *aller Seiten* erteilen.

Damit erhält die Anwendung ein normales, benutzerbezogenes Token, mit dem allein noch nicht im Namen der Seite gehandelt werden kann. Vielmehr dient dieses Token dazu, am Graph-API-Endpunkt `https://graph.facebook.com/me/accounts` eine Liste aller vom Benutzer administrierten Seiten zu beziehen. Das Ergebnis dieses API-Aufrufs sieht folgendermaßen aus:

```
{
  "data": [
    {
      "name": "Brieflos Aufreißzone",
      "access_token": "...",
      "category": "Company",
      "id": "172940206066630"
    },
    {
      "name": "Die Socialisten",
      "access_token": "...",
      "category": "Internet/software",
      "id": "129842310398883"
    },
    {
      "name": "Easy Drivers",
      "access_token": "...",
```

```
    "category": "Product/service",
    "id": "142652962425026"
  },
  ...
```

Listing 2.13 Rückgabewert des API-Aufrufs »/me/accounts«

Das Ergebnis-Array enthält alle administrierten Seiten mit ihrem Namen, ihrer numerischen ID und vor allem einem individuellen Access Token. Mit diesem Access Token kann die Anwendung fortan im Namen der Seite Wall-Postings verfassen, Fotos hochladen etc.

Wichtig: Das Seiten-Access-Token ist, anders als das Anwendungs-Access-Token, intern mit dem Benutzer-Access-Token des Benutzers verknüpft, der die Berechtigung `manage_pages` autorisiert hat. Darüber hinaus »erbt« das Seiten-Access-Token auch einige der in Abschnitt 2.6.3, »Erweiterte Berechtigungen«, beschriebenen erweiterten Berechtigungen. Möchte eine Anwendung etwa im Namen einer Seite Wall-Postings auf der Pinnwand der Seite veröffentlichen, wird ein Seiten-Access-Token benötigt, das mit `manage_pages` von einem Benutzer bezogen werden muss, der darüber hinaus auch die Berechtigung `publish_stream` erteilt hat.

Anders als Benutzer-Access-Tokens bleiben Seiten-Access-Tokens langfristig gültig. Lediglich – und damit unterscheiden sie sich von den Anwendungs-Tokens – wenn der autorisierende Benutzer sein Facebook-Kennwort ändert, verlieren sie ihre Gültigkeit.

Kapitel 3
Die Graph API

In diesem Kapitel lernen Sie Facebooks umfangreichste und wichtigste Schnittstelle, die Graph API, kennen. Die Graph API erlaubt lesenden und schreibenden Zugriff auf die meisten Objekte und Verbindungen, die den sozialen Graphen von Facebook ausmachen – Personen, Fotos, Seiten, Postings und vieles mehr.

Wie in Abschnitt 1.1, »Überblick über die Facebook-Plattform«, beschrieben, lassen sich alle Objekte der Facebook-Plattform mit ihren viel- und wechselseitigen Verbindungen im sogenannten *sozialen Graphen* abbilden oder visualisieren. Der soziale Graph besteht aus verschiedenen Knoten wie etwa Personen, Seiten bzw. Pages, Veranstaltungen, Fotos, Gruppen und vielen mehr, die auf verschiedene Arten über Kanten miteinander verknüpft sind. So sind Personen untereinander etwa mit Freundschafts- (beidseitig) oder auch Abonnementbeziehungen (einseitig) verbunden, während die Verbindung zwischen Personen und Seiten mittels eines Klicks auf GEFÄLLT MIR entsteht (obwohl Facebook selbst im Zusammenhang mit Personen, die mit einer Seite verknüpft sind, schon länger nicht mehr von »Fans« spricht, hat sich das »Fanwerden« vor allem im deutschen Sprachgebrauch verankert). Fotos sind mit ihrem Besitzer verknüpft, aber auch mit anderen Personen, die mit auf dem Foto markiert wurden. Veranstaltungen sind mit vielen Personen, den Teilnehmern, verknüpft etc.

Die Graph API gibt Entwicklern Zugriff auf dieses komplexe Objektmodell und bietet Methoden zum Lese- und Schreibzugriff auf große Teile des sozialen Graphen. Damit ist die Graph API die bei weitem wichtigste Schnittstelle, um Daten aus Facebook zu beziehen oder Daten auf der Plattform zu speichern.

Wichtiger Link:

- *http://developers.facebook.com/docs/reference/api/* – Die offizielle Dokumentation zur Graph API beschreibt alle Objekte, ihre Verknüpfungen und Eigenschaften ausführlich.

3.1 Grundlagen der Graph API

Die Graph API wurde von Facebook als HTTP-REST-Schnittstelle realisiert. Das bedeutet, dass Dritthersteller von Plattform-Anwendungen Lese- und Schreibzugriffe auf den sozialen Graphen von Facebook über einfache HTTP-Aufrufe von `https://graph.facebook.com` realisieren können. Um die Graph API zu benutzen, sind Sie also nicht auf die Bereitstellung eines SDKs für Ihre eigene Entwicklungsplattform angewiesen – schließlich bietet jeder zeitgemäße Development Stack Methoden, um HTTP-Zugriffe durchzuführen, etwa über die weit verbreitete cURL-Bibliothek.

HTTP-REST-Schnittstellen

REST steht für *Representational State Transfer* und beschreibt ein *zustandsloses* (im Englischen: *stateless*) Kommunikationsprotokoll zwischen Client und Server. Als *zustandslos* wird die Kommunikation deswegen bezeichnet, weil jede einzelne Anfrage eines Clients unabhängig von etwaigen vorherigen oder nachfolgenden Anfragen vom Server beantwortet wird. Dabei können sowohl Lese- als auch Schreibzugriffe (Hinzufügen, Bearbeiten oder Löschen) über eine REST-Schnittstelle abgebildet werden.

Da REST-Schnittstellen meistens auf dem einfachen HTTP-Protokoll basieren, sind sie eine einfacher umzusetzende Alternative zu komplexeren Protokollen wie CORBA, RPC (Remote Procedure Call) oder SOAP. REST-Schnittstellen sind in gewissem Sinne auch als »plattformagnostisch« zu bezeichnen, da das HTTP-Protokoll als Webstandard auf praktisch jeder Entwicklungsplattform implementiert ist. Die Anbieter von APIs müssen somit keine Rücksicht auf die Heterogenität der Ziel-Plattformen nehmen, auf denen die APIs verwendet werden sollen.

HTTP-REST-Schnittstellen haben sich seit dem Aufkommen von »Web 2.0«-Services und ihren zahlreichen APIs weitgehend etabliert und stellen heute die Grundlage von vielen öffentlich verfügbaren APIs dar.

Wichtiger Link:

- *http://rest.elkstein.org/* – eine Einführung in die Grundlagen von HTTP-REST-basierten Programmierschnittstellen

Wie in Abschnitt 1.4.1, »Einstellungen ›Basic‹«, beschrieben, werden alle Objekte (also Personen, Seiten, Fotos etc.) auf der Facebook-Plattform bzw. im Social Graph unter einer eindeutigen numerischen Identifikationsnummer (ID) gespeichert. Diese IDs stellen auch die Basis für die Adressierung von Objekten über die Graph API dar. Durch einfaches Anhängen einer numerischen ID an die Basis-URL der Graph API (`https://graph.facebook.com`) können Sie so Informationen von unterschiedlichsten Objekten über GET-Zugriffe abfragen:

```
https://graph.facebook.com/ObjektID
```

Die folgenden Beispiel-URLs können Sie probehalber auch einfach in Ihrem Webbrowser öffnen:

- `https://graph.facebook.com/609190863` – Liefert ein Personenobjekt vom Typ `user` (»Michael Kamleitner«).
- `https://graph.facebook.com/113515098748988` – Liefert ein Veranstaltungsobjekt vom Typ `page` (»f8 2011«).
- `https://graph.facebook.com/19292868552` – Liefert ein Seitenobjekt (»Facebook-Plattform«).
- `https://graph.facebook.com/195466193802264` – Liefert ein Gruppenobjekt vom Typ `group` (»Facebook Developers«).
- `https://graph.facebook.com/127675550671337` – Liefert ein Applikationsobjekt vom Typ `application` (»Last.fm Scrobbler«).
- `https://graph.facebook.com/10150318315203553` – Liefert ein Foto vom Typ `photo` (»f8-Logo«).
- `https://graph.facebook.com/244720248918422` – Liefert ein Fotoalbum vom Typ `album` (»Winter im Museumsquartier«).

Wenn Sie die angegebenen URLs im Browser öffnen, werden Sie sehen, dass die Graph API je nach Objekttyp mit unterschiedlich umfangreichen Ergebnisdatensätzen antwortet. Für mein persönliches Personenobjekt unter `https://graph.facebook.com/609190863` erhalten Sie etwa folgendes Ergebnis:

```
{
   "id": "609190863",
   "name": "Michael Kamleitner",
   "first_name": "Michael",
   "last_name": "Kamleitner",
   "link": "http://www.facebook.com/michael.kamleitner",
   "username": "michael.kamleitner",
   "gender": "male",
   "locale": "en_US"
}
```

Listing 3.1 Der Abruf eines Personenobjekts liefert diesen Rückgabewert.

Die Ergebnisse der Graph API werden immer in Form eines assoziativen Objekts in JSON-Notation zurückgeliefert. Die früher unterstützte alternative Ausgabe im XML-Format, die durch das Anhängen des GET-Parameters `?format=xml` aktiviert werden konnte, wird in der aktuellen Graph API nicht mehr unterstützt.

JSON-Notation

JSON (*JavaScript Object Notation*) ist ein einfacher, leicht zu implementierender Standard zum textbasierten Datenaustausch und damit eine Alternative zu XML- oder CSV-Dateien, die mit dem Aufkommen von AJAX-basierten JavaScript-Anwendungen im Web 2.0 an Bedeutung gewonnen hat. JSON-Notation bedeutet, dass Datenstrukturen und assoziative Arrays in der von JavaScript gewohnten Art und Weise codiert werden. Der große Vorteil: Eine AJAX-basierte JavaScript-Anwendung kann Resultate eines HTTP-Requests so direkt weiterverwenden und kommt auf diese Weise ohne aufwendiges Parsen von XML-Daten aus.

Für PHP-Entwickler ist der Umgang mit JSON-Notation denkbar einfach: Über die beiden Methoden `json_decode()` und `json_encode()` können Datenstrukturen einfach in beide Richtungen zwischen PHP-eigenen Objekten und JSON konvertiert werden:

```
<?
$php_array = array("id" => 609190863,
                   "name" => array(
                             "first_name" => "Michael",
                             "last_name" => "Kamleitner"));
print "JSON-Notation:\n".json_encode($php_array)."\n\n";
print "PHP:\n";
print_r(json_decode(json_encode($php_array)));
?>
```

Im Beispiel wird ein einfaches assoziatives Array als PHP-Datenstruktur angelegt und anschließend mittels `json_encode($php_array)` in JSON-Notation ausgegeben. Dieses Resultat kann mittels `json_decode()` wiederum zurück in ein assoziatives Array verwandelt werden. Die Ausgabe dieses Beispiels sieht folgendermaßen aus:

```
JSON-Notation:
{"id":609190863,"name":{"first_name":"Michael","last_name":"Kamleitner"}}

PHP:
stdClass Object
(
    [id] => 609190863
    [name] => stdClass Object
    (
        [first_name] => Michael
        [last_name] => Kamleitner
    )
)
```

Wichtige Links:

- *http://en.wikipedia.org/wiki/JSON* – Wikipedia-Eintrag zu JSON
- *http://www.json.org/* – Einführung in JSON
- *http://php.net/manual/en/book.json.php* – Informationen zum Umgang mit JSON in PHP

3.1.1 Zugriff mit eindeutigen Identifikationsnamen

Bestimmte Objekte im sozialen Graphen von Facebook können neben der eindeutigen Identifikationsnummer auch über einen eindeutigen Namen identifiziert werden. Dies gilt vor allem für Personen- und Seitenobjekte, für die Benutzer bzw. Seitenadministratoren einen frei wählbaren Namen festlegen können. Die Graph API kann diese Objekte zusätzlich zur numerischen ID auch über ihren eindeutigen Kurznamen adressieren:

- `https://graph.facebook.com/michael.kamleitner` entspricht dem Zugriff über `https://graph.facebook.com/609190863` (Personenobjekt).
- `https://graph.facebook.com/diesocialisten` entspricht dem Zugriff über `https://graph.facebook.com/129842310398883` (Seitenobjekt).
- `https://graph.facebook.com/meinklub` entspricht dem Zugriff über `https://graph.facebook.com/32788395891` (Applikationsobjekt).

Festlegen von Benutzer-, Applikations-, Gruppen- und Seitennamen

Individuelle Benutzer- oder Seitennamen (oft auch als *Vanity-URLs* bezeichnet, da sie die Erkennbarkeit und Einprägsamkeit von Facebook-Profil-URLs deutlich erhöhen) können vom jeweiligen Benutzer oder Seitenadministrator unter *http://www.facebook.com/username* selbst festgelegt werden. Benutzer- und Seitennamen dürfen nur aus Groß- und Kleinbuchstaben, Ziffern und Punkten bestehen und müssen global eindeutig sein. Nachdem der Benutzer- oder Seitenname festgelegt wurde, ist das jeweilige Profil unter der Kurz-URL *http://www.facebook.com/GewaehlterName* abrufbar.

Achtung: Die Wahl von Benutzer- und Seitenname kann nur einmalig erfolgen! Ein einmal festgelegter Name kann vom Benutzer oder Seitenadministrator nicht mehr geändert und rückgängig gemacht werden! Der Benutzer- oder Seitenname sollte auch nach SEO-Maßgaben gewählt werden, da Facebook-Profile und -Seiten auch von Google und anderen Suchmaschinen indiziert werden.

Hinweis: Auch für angelegte Gruppen können Gruppenadministratoren einen Kurznamen festlegen. Diese Einstellung erfolgt am Gruppenprofil unter Edit Group, wo

mit der Option Email Address eine E-Mail Adresse festgelegt werden kann, unter der in die Gruppe gepostet werden kann. Wird etwa die Gruppen-E-Mail-Adresse *diesocialisten@groups.facebook.com* gewählt, ist das Gruppenprofil fortan auch unter der Kurz-URL *http://www.facebook.com/groups/diesocialisten* abrufbar. Aber Vorsicht: Der Gruppenname erlaubt keine global eindeutige Identifikation und ist daher nicht zur Adressierung über die Graph API zu gebrauchen!

Ein weiterer Sonderfall sind Applikationen. Wie in Abschnitt 1.4.1, »Einstellungen ›Basic‹«, beschrieben, kann für jede Applikation ein eindeutiger App Namespace vergeben werden. Dieser Namespace kann gleichzeitig mit der Adressierung eines Applikationsobjekts über die Graph API verwendet werden.

Darüber hinaus wurde bis Februar 2012 jede Applikation auf einer eigenen Profilleite unter einer URL der Form *http://facebook.com/apps/application.php?id=ApplikationsID* präsentiert. Diese Applikationsprofile sind wie herkömmliche Seiten aufgebaut – sie bieten etwa eine Pinnwand, auf der Fans und Benutzer der Anwendung kommunizieren –, können aber keine Vanity-URL besitzen. Auch andere von Seiten bekannte Features wie Reiter, Fotos, Videos und Notizen können auf Applikationsprofilen genutzt werden. Im Februar 2012 wurden die Profilseiten für Anwendungen von Facebook eingestellt – Entwickler werden stattdessen angehalten, eine übliche Profilseite für die Community ihrer Anwendung anzulegen.

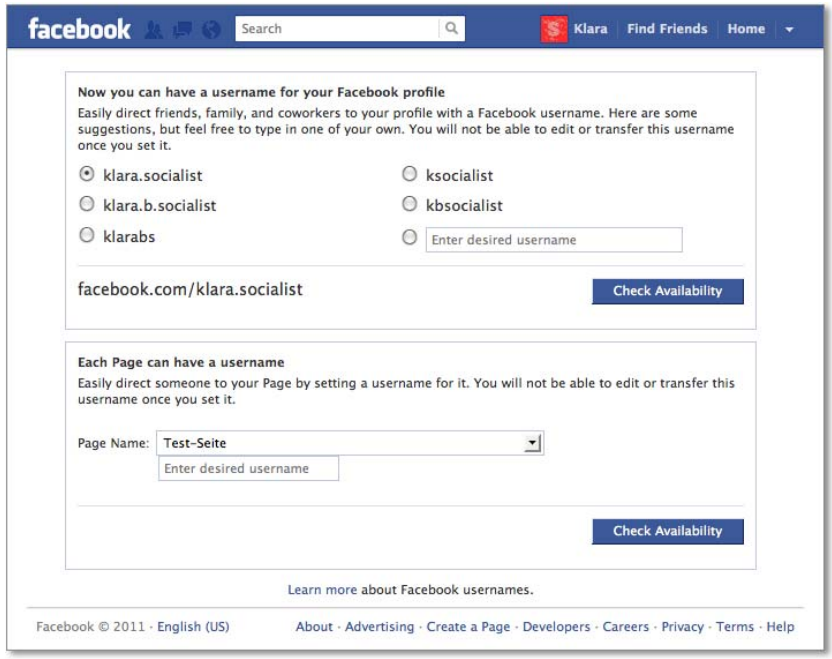


Abbildung 3.1 Festlegen von Benutzer- oder Seitennamen unter *http://www.facebook.com/username*

3.1.2 Authentifizierter Zugriff auf die Graph API

Wie die vorangegangenen Beispiele bereits gezeigt haben, können viele Lesezugriffe auf die Graph API anonym und ohne vorherige Authentifikation erfolgen, die so angezeigten Daten sind also öffentlich zugänglich. Bei bestimmten Objekttypen wie etwa Seiten (die per Definition immer öffentlich sichtbar sind) oder öffentlichen Veranstaltungen liegt dies auch in der Natur dieser Objekte. Bei anderen Objekttypen, wie z. B. Gruppen, Fotos, aber natürlich vor allem Personenobjekten selbst, ist die Situation nicht ganz so einfach. Hier hängt es von den jeweiligen Privatsphäre-Einstellungen, die ein bestimmter Benutzer getroffen hat, ab, welche Informationen die Graph API bei einem nicht authentifizierten Lesezugriff preisgibt. Schreibzugriffe müssen naheliegenderweise immer authentifiziert erfolgen, da alle erstellten Inhalte im sozialen Graphen einem Benutzer zugeordnet werden müssen.

Zum authentifizierten Zugriff auf die Graph API ist lediglich ein Access Token notwendig, das, wie in Kapitel 2, »Authentifikation und Autorisierung«, beschrieben, auf Benutzer-, Seiten- oder Applikationsebene erworben werden kann.

Um einen authentifizierten Zugriff auf die Graph API durchzuführen, müssen Sie nur das Access Token als HTTP-Parameter an die URL hängen:

```
https://graph.facebook.com/ObjektId?access_token=...
```

Achtung: Über Access Token authentifizierte Zugriffe auf die Graph API müssen aus Sicherheitsgründen immer SSL-verschlüsselt, also über das HTTPS-Protokoll, erfolgen!

Mit dem Access Token entscheidet sich auch, auf welche Objektdaten ein Entwickler tatsächlich zugreifen kann – wird das Access Token ohne besondere, erweiterte Zugriffsrechte (siehe auch Abschnitt 2.6, »Erweiterte Zugriffsrechte (»Extended Permissions«)«) ausgestellt, kann etwa nicht auf die E-Mail-Adresse oder die Fotos eines Benutzerobjekts zugegriffen werden. Mit anderen Worten: Die Facebook-Plattform verwaltet alle ausgegebenen Access Tokens und speichert dazu ab, welcher Benutzer damit welche Applikation autorisiert hat und welche spezifische Kombination von Zugriffsrechten er dabei erteilt hat. Die folgenden Abschnitte werden detailliert auf alle verfügbaren Objekttypen, ihre Datenfelder und verknüpften Objekte eingehen und dabei jeweils erläutern, welche Berechtigungen Sie für den Zugriff benötigen.

3.1.3 Objektverknüpfungen im sozialen Graphen

Es liegt in der Natur des sozialen Graphen, dass die darin gespeicherten Objekte oder Knoten nicht isoliert voneinander existieren, sondern über bestimmte Beziehungen oder Kanten miteinander verknüpft sind. Die möglichen Verknüpfungen leiten sich dabei vom jeweiligen Objekttyp ab. So können z. B. Personenobjekte mit anderen

Personenobjekten verknüpft sein (Freundschaft oder Abonnement), Personen mit Seiten (Fans bzw. GEFÄLLT MIR), Fotoalben mit Fotos, Fotos mit Markierungen (*Tags*) etc.

Die Graph API bildet diese Objektverknüpfungen in der URL-Struktur ab. Um zu einem bestimmten Graph-Objekt die verknüpften Objekte eines bestimmten Objekttyps herauszufinden, genügt es, den Namen des gewünschten Objekttyps an die URL zur Adressierung anzuhängen:

```
https://graph.facebook.com/ObjektID/ObjektVerknüpfungsTyp
```

Die folgenden Beispiele zeigen, wie Sie die verknüpften Objekte eines Benutzers über die Graph API abfragen können. Viele dieser Zugriffe erfordern ein Access Token – deshalb können Sie diese Beispiel-URLs nicht ohne Weiteres in Ihrem Browser ausprobieren. Auf *http://developers.facebook.com/docs/reference/api/* finden Sie jedoch zahlreiche weitere Beispiele, die bereits mit einem gültigen Access Token verknüpft sind:

- `https://graph.facebook.com/609190863/?access_token=...` – das Personenobjekt selbst
- `https://graph.facebook.com/609190863/feed?access_token=...` – die Pinnwand-Einträge der Person
- `https://graph.facebook.com/609190863/photos?access_token=...` – die Fotos, die von der Person hochgeladen wurden
- `https://graph.facebook.com/609190863/likes?access_token=...` – Seiten, auf denen die Person auf GEFÄLLT MIR geklickt hat
- `https://graph.facebook.com/609190863/checkins?access_token=...` – Checkins der Person
- `https://graph.facebook.com/609190863/events?access_token=...` – Veranstaltungen, die von der Person besucht werden
- `https://graph.facebook.com/609190863/groups?access_token=...` – Gruppen, bei denen die Person Mitglied ist

Wenn Sie wissen möchten, welche verknüpften Objekttypen ein bestimmtes Objekt aufweist, können Sie dies durch Anhängen des Parameters `metadata=1` an die URL zur Adressierung des Objekts herausfinden:

```
https://graph.facebook.com/609190863?metadata=1
```

Das zurückgegebene JSON-Array enthält im Element `metadata` alle verfügbaren `connections` eines Objekts – praktischerweise werden die Verbindungen gleich als URL geliefert, die Sie zur weiteren Verarbeitung direkt abrufen können:

```
{
   "id": "609190863",
   "name": "Michael Kamleitner",
   "first_name": "Michael",
   "last_name": "Kamleitner",
   "link": "http://www.facebook.com/michael.kamleitner",
   "username": "michael.kamleitner",
   "gender": "male",
   "locale": "en_US",
   "metadata": {
      "connections": {
         "home": "https://graph.facebook.com/609190863/home",
         "feed": "https://graph.facebook.com/609190863/feed",
         "links": "https://graph.facebook.com/609190863/links",
         "posts": "https://graph.facebook.com/609190863/posts",
         ...
```

Listing 3.2 Rückgabewerte mit dem Parameter »metadata=1«

Tipp: Erfolgt der Zugriff auf die Graph API mit einem Access Token, das an einen Benutzer geknüpft ist, kann anstelle der numerischen ID oder des eindeutigen Benutzernamens das Schlüsselwort `me` zur Adressierung des eigenen Benutzerobjekts verwendet werden. Die folgenden Aufrufe sind somit gleichbedeutend:

```
https://graph.facebook.com/me
https://graph.facebook.com/609190863
https://graph.facebook.com/michael.kamleitner
```

3.1.4 Selektion in der Graph API

Üblicherweise liefern Aufrufe der Graph API die meisten – gemäß den Zugriffsrechten – verfügbaren Datenfelder des Objekts im Ergebnis zurück. Um etwa Bandbreite bei der Datenübertragung zwischen Facebook und dem Anwendungsclient oder -server zu sparen oder Datenfelder, die nicht standardmäßig ausgelesen werden, abzufragen, ist es möglich, die zurückgelieferten Felder mit einer kommaseparierten Liste im Parameter `fields` zu selektieren:

```
https://graph.facebook.com/ObjektID/?fields=FeldName,FeldName...
```

- `https://graph.facebook.com/609190863?fields=id,name` – Liefert ausschließlich die numerische ID und den Namen des Personenobjekts.
- `https://graph.facebook.com/129842310398883?fields=id,name,likes` – Liefert ausschließlich die numerische ID, den Namen und die Anzahl der GEFÄLLT MIR eines Seitenobjekts zurück.

Während die eingesparte Bandbreite bei der Selektion der zurückgelieferten Felder meist recht gering sein dürfte, verspricht die nächste Methode, die Anzahl der notwendigen Zugriffe auf die Graph API – und damit die Laufzeit Ihres Codes – bedeutend zu reduzieren. Verwenden Sie zur Adressierung der gesuchten Graph-Objekte anstelle des gewohnten URL-Segments den Parameter `ids`, können Sie mehrere gesuchte Objekt-IDs kommasepariert übergeben:

```
https://graph.facebook.com/?ids=ObjektID,ObjektID,...
```

- `https://graph.facebook.com/?ids=3415743,335700009,500065899` – Liefert drei unterschiedliche Personenobjekte.
- `https://graph.facebook.com/?ids=diesocialisten,facebook,platform` – Liefert die Objektdaten der drei Seiten »Die Socialisten«, »Facebook« und »Facebook Platform«.

Eine besondere Form der Adressierung von Graph-Objekten ergibt sich bezüglich sogenannter Open-Graph-Objekte, die in Kapitel 7, »Social Plugins«, näher beleuchtet werden. So viel sei vorweggenommen: Open Graph erlaubt es Entwicklern, beliebige Content-Objekte, die auf Webseiten außerhalb von Facebook.com repräsentiert werden, in den sozialen Graphen von Facebook zu integrieren. Die URLs zu diesen Webseiten ersetzen dabei die numerische ID zur eindeutigen Adressierung dieser Objekte. Konsequenterweise können Sie mit dem Parameter `ids` also auch Open-Graph-Objekte mit Ihrer URL abfragen:

```
https://graph.facebook.com/?ids=http://url.to-open-graph.com/...
```

- `https://graph.facebook.com/?ids=http://die.socialisten.at/2011/02/facebook-sponsored-stories-jetzt-in-osterreich/` – Liefert Informationen zu einem Blog-Artikel, der als Open-Graph-Objekt in Facebook integriert wurde.

3.1.5 Selektion von Datumsfeldern

Datenfelder, die Datumsinformationen enthalten, also z. B. der Zeitstempel, an dem ein bestimmtes Wall-Posting erstellt oder geändert wurde, werden von der Graph API normalerweise im ISO-8601-Format zurückgeliefert. Das ISO-8601-Format stellt das Datum in klar lesbarer Form dar. Der folgende Aufruf der Graph API liefert die Wall-Postings einer Seite und deren Erstellungsdatum im ISO-8601-Format:

```
https://graph.facebook.com/platform/feed?
fields=id,name,created_time&access_token=...
```

Das Ergebnis sieht folgendermaßen aus:

```
{
   "data": [
   {
      "id": "19292868552_237589062971265",
      "name": "Facebook Preferred Developer Consultant...",
      "created_time": "2011-11-17T18:00:16+0000"
   },
   {
      "id": "19292868552_239717212756395",
      "name": "Announcing the Fall Submission Round...",
      "created_time": "2011-11-10T23:13:20+0000"
   },
   ...
}
```

Listing 3.3 Rückgabe von Datumsfeldern im ISO-8601-Format

Um Datumsfelder in einem anderen, frei wählbaren Format zu beziehen, können Sie Anfragen an die Graph API mit dem Parameter `date_format` versehen. Das gewünschte Datumsformat wird dabei in Form eines Format-Strings bestimmt, wie er auch in der PHP-Funktion `date()` verwendet wird.

- `https://graph.facebook.com/platform/feed?fields=id,name,created_time&date_format=Y-m-d&access_token=...` – Liefert Datumsfelder im Format `Y-m-d` (Jahr-Monat-Tag, z. B. »2011-11-22«) zurück.
- `https://graph.facebook.com/platform/feed?fields=id,name,created_time&date_format=U&access_token=...` – Liefert Datumsfelder gemäß dem UNIX-Standard als Anzahl der Sekunden seit 1. Januar 1970 00:00:00 zurück (z. B. »1321552816«).

Wichtige Links:

- *http://php.net/manual/en/function.date.php* – PHP-Manual zur `date()`-Funktion
- *http://en.wikipedia.org/wiki/ISO_8601* – Wikipedia-Eintrag zum ISO-8601-Format

Wichtig: Die Graph API liefert Datumsfelder automatisch immer bezogen auf die Zeitzone des abfragenden Benutzers aus!

3.1.6 Zugriff auf Profilbilder

Profilbilder von Personen, Seiten und anderen Objekten können wie gewohnt über die Adressierung des Objekts abgerufen werden und werden im Datenfeld `picture` als URL zum Binärbild zurückgeliefert. Dabei ist zu beachten, dass die Graph API etwa bei

Personenobjekten das Feld `picture` *nicht standardmäßig ausliest* – aus diesem Grund muss `picture`, wie in Abschnitt 3.1.4, »Selektion in der Graph API«, beschrieben, explizit selektiert werden:

```
https://graph.facebook.com/ObjektID/?fields=id,name,picture
```

Diese Selektion liefert etwa für ein Personenobjekt folgendes Resultat zurück:

```
{
   "id": "609190863",
   "name": "Michael Kamleitner",
   "picture": "http://profile.ak.fbcdn.net/".
              "hprofile-ak-snc4/273867_609190863_8020272_q.jpg"
}
```

Das Feld `picture` enthält die direkt abrufbare URL zum Profilbild der Personen und kann etwa direkt in das `src`-Attribut eines `<img>`-Tags geschrieben werden.

Eine alternative Methode zur Einbindung von Profilbildern hilft Entwicklern einerseits, zusätzliche Aufrufe der Graph API einzusparen, und ist andererseits flexibler im Umgang mit der Bildgröße von Profilbildern. Ist die Adressierung eines Graph-Objekts über eine numerische ID oder einen eindeutigen Namen möglich, kann die Graph-URL durch Anhängen des Pfades `/picture` direkt in `<img>`-Tags verwendet werden:

```
<img src=https://graph.facebook.com/609190863/picture/>
```

Der Parameter `type` erlaubt dabei, das Profilbild in drei unterschiedlichen Größen zu beziehen:

- `https://graph.facebook.com/me/picture/?type=square` – Liefert ein exakt 50 × 50 Pixel großes Thumbnail des Profilbildes zurück.
- `https://graph.facebook.com/me/picture/?type=normal` – Liefert das Profilbild in 100 Pixeln Breite und variabler Höhe zurück.
- `https://graph.facebook.com/me/picture/?type=large` – Liefert das Profilbild in 200 Pixeln Breite und variabler Höhe zurück.

3.1.7 Paging in der Graph API

Viele Zugriffe auf die Graph API liefern eine große Anzahl an Ergebnisdatensätzen zurück, etwa der Zugriff auf die Wall-Postings einer Seite oder die mit Gefällt mir verknüpften Seiten einer Person. Mit den Parametern `limit` und `offset` können Ent-

wickler durch umfangreiche Ergebnisse blättern, indem sie die Anzahl der zurückgelieferten Objekte und das erste zurückgelieferte Objekt festlegen:

```
https://graph.facebook.com/ObjektID/ObjektVerknüpfungsTyp/
?offset=n&limit=10
```


- `https://graph.facebook.com/me/likes?limit=3&offset=0&access_token=` ... – Liefert die ersten drei mit GEFÄLLT MIR verknüpften Seiten einer Person zurück.
- `https://graph.facebook.com/me/likes?limit=3&offset=0&access_token=` ... – Liefert die nächsten drei mit GEFÄLLT MIR verknüpften Seiten einer Person zurück.

Die Ergebnisdatensätze der Graph API enthalten praktischerweise im Element `paging` die Links, um im Ergebnis vor- (`next`) oder zurückzublättern (`previous`). Das folgende Code-Beispiel blättert etwa die Freunde des eigenen Benutzerobjekts in Zehnerschritten durch:

```
<?
$url =
  "https://graph.facebook.com/me/friends".
  "?limit=10&offset=0&access_token=...";
while ($url) {
  $friends = json_decode(curl($url));
  foreach ($friends->data as $f) {
      print $f->name."\n";
   }
   $url = (isset($friends->paging->next) ?
          $friends->paging->next : null);
}?>
```

Listing 3.4 Durchblättern der Freunde eines Personenobjekts mittels Paging

3.1.8 Veröffentlichen und Löschen von Objekten mit der Graph API

Bestimmte Objekttypen können mit der Graph API nicht nur ausgelesen, sondern – ein mit entsprechenden Zugriffsberechtigungen ausgestattetes Access Token vorausgesetzt – auch im sozialen Graphen von Facebook veröffentlicht werden. Für diese Objekttypen ist also auch ein Schreibzugriff möglich – genauer gesagt, geht es dabei um das *Erstellen neuer Objekte*, das Bearbeiten vorhandener Objekte ist derzeit grundsätzlich nicht vorgesehen.

Während die bisher behandelten Lesezugriffe mit GET-Requests durchgeführt wurden, sieht die Graph API für das Veröffentlichen (*Publishing*) von Objekten POST-Requests mittels HTTP vor. Die POST-Requests erfolgen dabei immer auf jenen API-Endpunkt, der ansonsten zum Auslesen verwendet werden würde:

- *https://graph.facebook.com/me/feed* – Veröffentlichen eines Wall-Postings auf der Pinnwand des eigenen Benutzers
- *https://graph.facebook.com/me/photos* – Veröffentlichen eines Fotos im eigenen Benutzerprofil
- *https://graph.facebook.com/me/events* – Veröffentlichen einer Veranstaltung im eigenen Benutzerprofil

Als POST-Parameter müssen Sie dabei immer ein gültiges Access Token und die jeweils zur Veröffentlichung benötigten Inhalte übergeben – bei einem Wall-Posting also den Text, der gepostet werden soll, bei einem Foto die Binärdaten des Bildes etc.

Auf UNIX- und Mac OS-Rechnern, auf denen meist das Kommandozeilen-Tool `curl` verfügbar ist, können Sie die Veröffentlichung mittels POST-Request auf der Kommandozeile ausprobieren:

```
curl -F 'access_token=...' \
     -F 'message=Das ist ein Test-Wall-Posting!' \
     https://graph.facebook.com/me/feed
```

Neben dem Access Token muss zum Veröffentlichen eines Wall-Postings also lediglich der gewünschte Text im Parameter `message` an die Graph-URL übergeben werden, die die eigene Pinnwand adressiert (`/me/feed`).

Mit der in Abschnitt 2.2.3, »Bezug des Access Tokens«, eingeführten PHP-Funktion `curl()` sieht der entsprechende POST-Request auf die Graph API folgendermaßen aus:

```
<?
$params = array(
                "access_token" => "...",
                "message" => "Das ist ein Test-Wall-Posting!"
                );
$url = "https://graph.facebook.com/me/feed";
curl ($url, $params, "POST");
?>
```

Listing 3.5 Veröffentlichen eines Wall-Postings mit der Graph API

Die Graph API unterstützt derzeit die Veröffentlichung folgender Objekttypen:

- Wall-Postings (`/ProfilID/feed`)
- Kommentare (`/ObjektID/comments`)
- Gefällt mir (`/ObjektID/likes`) – Wird lediglich für bestimmte Objekttypen, wie etwa Fotos, Wall-Postings, *nicht aber für Seiten* unterstützt.

- Notizen (`/ProfiID/notes`)
- Links (`/ProfilID/links`)
- Veranstaltungen (`/ProfilID/events`)
- Veranstaltungsteilnahme (`/EventID/attending`, `/EventID/maybe`, `/EventID/declined`)
- Fotoalben (`/ProfilID/albums`)
- Fotos (`/ProfilID/photos` oder `/AlbumID/photos`) – Fotos können in ein bestimmtes Album oder unabhängig von Alben veröffentlicht werden. Wenn kein spezielles Album angegeben wird, werden Fotos automatisch in einem Album mit dem Titel der Anwendung abgelegt.
- Checkins (`/ProfilID/checkins`)

Die später folgenden detaillierten Erläuterungen zu den unterschiedlichen Objekttypen enthalten jeweils auch Hinweise zum Veröffentlichen dieser Objekte.

Neben der Veröffentlichung von Objekten kann die Graph API auch zum Löschen von bestimmten Objekttypen genutzt werden. Dies geschieht mit der – im Vergleich zu GET und POST weniger bekannten – HTTP-Methode DELETE, die über die Kommandozeile folgendermaßen abgesetzt werden kann:

```
curl -X DELETE
     https://graph.facebook.com/ObjektID?access_token=...
```

Die Objekt-ID enthält dabei die numerische ID eines Objekts, für dessen Objekttypen die Graph API Löschzugriffe unterstützt. Dies sind vor allem Kommentare, Fotos, Fotoalben, Links und Notizen.

Da unsere `curl()`-Funktion wie viele andere gängige Implementierungen von `curl` DELETE-Zugriffe nicht explizit unterstützt, bietet Facebook zusätzlich die gleiche Funktionalität über GET-Requests an. Dabei müssen Sie an die aufgerufene URL lediglich den Parameter `method=delete` anhängen:

```
https://graph.facebook.com/ObjektID/?
method=delete&access_token=...
```

3.1.9 Der Graph API Explorer

Egal, welche Entwicklungsumgebung oder Programmiersprache Sie nutzen – mit dem bisher vermittelten Wissen können Sie sofort beginnen, HTTP-Anfragen an die Graph API zu stellen. Dank Facebook müssen Sie dazu aber nicht Editor und Code bemühen! Stattdessen können Sie Ihre ersten Schritte mit der Graph API mithilfe des *Graph API Explorers* tun. Dieses praktische Tool finden Sie unter *http://developers.facebook.com/tools/explorer*. Es erlaubt, beliebige Zugriffe auf die Graph API mit einem einfachen Web Interface auszuprobieren.

Entwickler können dabei beliebige Zugriffs-URLs an die Graph API senden. Dabei können sowohl GET-Zugriffe als auch POST-Zugriffe mit beliebigen Parametern getestet werden. Im linken Bereich wird das JSON-Resultat des API-Zugriffs angezeigt, während rechts daneben die verknüpften Objekttypen per Mausklick zur Verfügung stehen und die einzelnen Datenfelder des Ergebnisses erklärt werden.

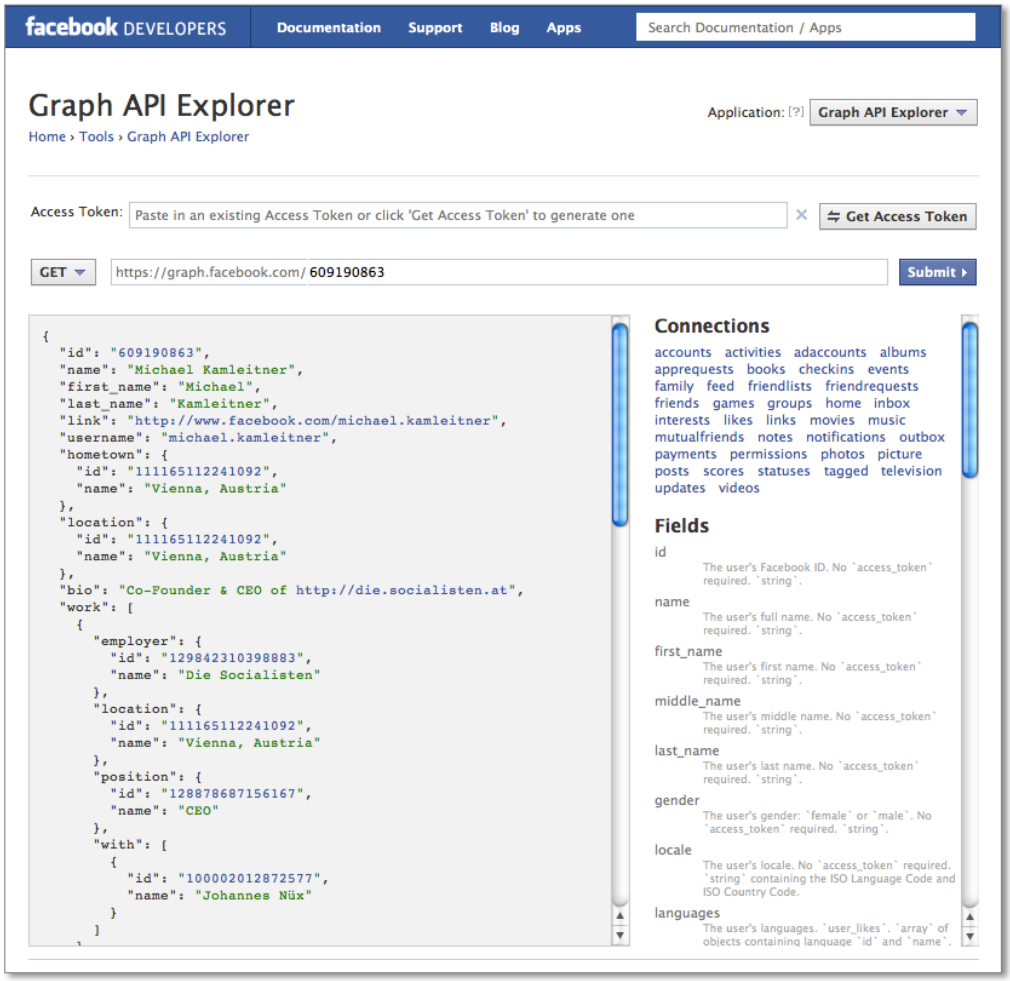


Abbildung 3.2 Der Graph API Explorer ermöglicht das einfache Erforschen der Objekte im sozialen Graphen.

Über das Menü APPLICATION können Entwickler einfach Access Tokens von bestehenden Anwendungen übernehmen. Dies ist besonders praktisch, da die Resultate von API-Zugriffen wesentlich vom verwendeten Access Token abhängig sind. Apro-

pos: Mit GET ACCESS TOKEN können Sie sich ein individuelles Access Token ausstellen lassen. In einem detaillierten Dialog können Sie dabei festlegen, welche Zugriffsrechte auf die Daten des Benutzers (USER DATA PERMISSIONS) und seiner Freunde (FRIENDS DATA PERMISSIONS) Sie gewähren möchten und welche erweiterten Zugriffsrechte (EXTENDED PERMISSIONS) Ihr Access Token abdecken soll:

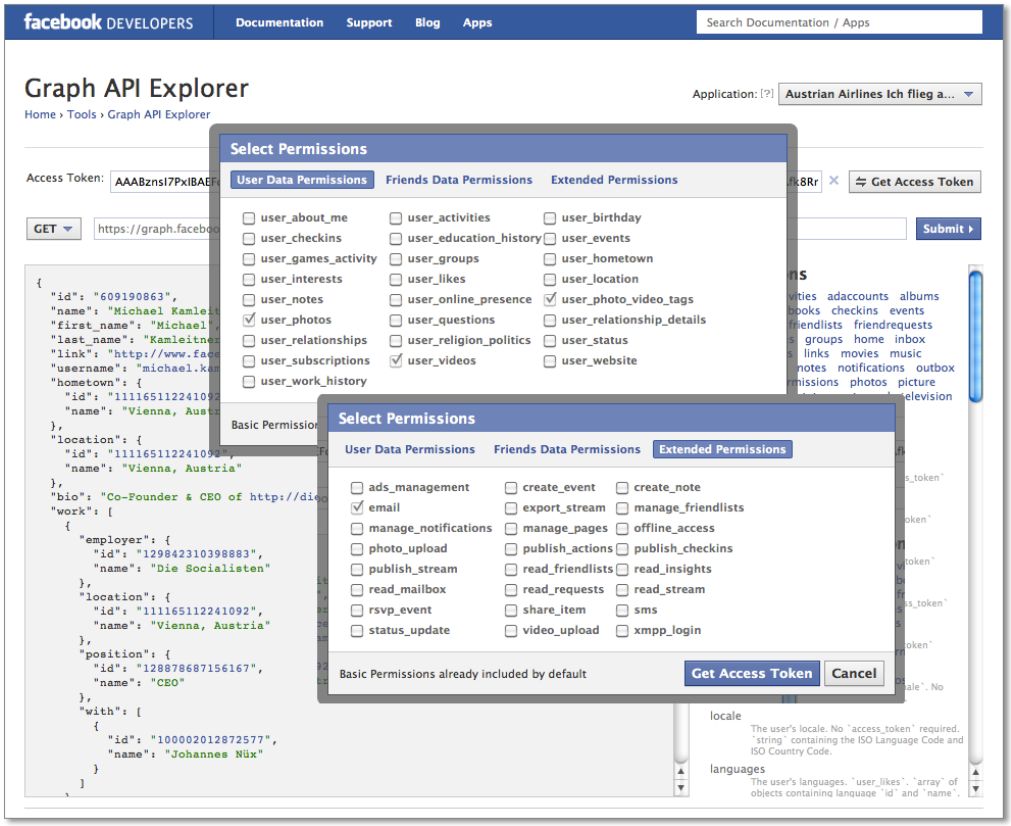


Abbildung 3.3 Erzeugen Sie sich mit dem Graph API Explorer ein individuelles Access Token, um zu erkunden, welche Zugriffsrechte notwendig sind, um bestimmte Daten auszulesen.

Der Graph API Explorer ist ein äußerst praktisches Hilfsmittel zum Kennenlernen von Facebooks sozialem Graphen, aber auch ein unverzichtbares Werkzeugbei der täglichen Arbeit als Facebook-Plattform-Entwickler. Nehmen Sie sich die Zeit, und erforschen Sie das Objektmodell von Facebook!

Wichtiger Link:

- *https://developers.facebook.com/tools/explorer* – der Graph API Explorer

3.2 Objekttypen und Verknüpfungen der Graph API

Die folgenden Abschnitte behandeln die unterschiedlichen Objekttypen, die im sozialen Graphen der Facebook-Plattform abgebildet werden und auf die die Graph API Zugriff gewährt. Die einzelnen Objekttypen sind durch die Datenfelder, die sie aufweisen, ebenso charakterisiert wie durch die jeweils möglichen Objektverknüpfungen und die Operationen, die die Graph API dem Entwickler gestattet.

Hinweis: Facebook erweitert die Graph API laufend – einerseits werden bestehende Objekte um Funktionalitäten erweitert, andererseits werden neue Objekttypen und -verknüpfungen ergänzt, wenn neue Features für Benutzer verfügbar gemacht werden. Daher stellt dieses Kapitel eine umfangreiche Momentaufnahme der Graph API dar, die zum Zeitpunkt der Veröffentlichung dieses Buches vermutlich nicht mehr vollständig sein wird. Für Anwendungsentwickler ist daher die regelmäßige Lektüre der offiziellen Dokumentation der Graph API Pflicht. Die meisten Änderungen werden auch frühzeitig im Developer-Blog angekündigt.

3.2.1 Das User-Objekt

Jeder auf Facebook registrierte Benutzer-Account, also jede Person, die im sozialen Graphen von Facebook abgebildet ist, wird durch ein Objekt vom Typ `user` repräsentiert. User-Objekte können durch die eindeutige, numerische ID, den optional gewählten Benutzernamen oder das Schlüsselwort `me` adressiert werden:

- `https://graph.facebook.com/609190863`
- `https://graph.facebook.com/michael.kamleitner`
- `https://graph.facebook.com/me`
  benötigt die Authentifizierung über Access Tokens des Benutzers

Felder des User-Objekts

User-Objekte stellen eine Vielzahl an Datenfeldern oder Attributen bereit, die abhängig von der Verwendung eines Access Tokens abgefragt werden können:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID des Benutzers	kein Access Token notwendig	`string`

Tabelle 3.1 Die Felder des User-Objekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`name`	vollständiger Name des Benutzers	kein Access Token notwendig	`string`
`first_name`	Vorname des Benutzers	kein Access Token notwendig	`string`
`middle_name`	mittlerer Name des Benutzers	kein Access Token notwendig	`string`
`last_name`	Nachname des Benutzers	kein Access Token notwendig	`string`
`gender`	Geschlecht des Benutzers	kein Access Token notwendig	`male` oder `female`
`locale`	Lokalisierungsinformationen des Benutzers	kein Access Token notwendig	`string` enthält den ISO-Sprach- und Landescode, z. B. `de_DE`.
`languages`	Sprachen, die der Benutzer spricht	`user_likes`	Array aus Objekten mit Sprach-`id` und `-name`
`link`	URL zum Facebook-Benutzerprofil	kein Access Token notwendig	`url`
`username`	der frei gewählte, eindeutige Benutzername	kein Access Token notwendig	`string`
`third_party_id`	eine anonymisierte, eindeutige Benutzerkennung	Benötigt ein beliebiges Access Token.	`string`
`timezone`	Zeitzone des Benutzers als Differenz zu UTC	nur verfügbar mit Access Token des aktuellen Benutzers	`number`

Tabelle 3.1 Die Felder des User-Objekts (Forts.)

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
updated_time	Zeitstempel der letzten Änderung am Profil. Berücksichtigt keine Änderungen an languages, link, timezone, verified, interested_in, favorite_athletes, favorite_teams, video_upload_limits.	Benötigt ein beliebiges Access Token.	string mit einem ISO-8601-Zeitstempel
verified	Gibt an, ob der Benutzer sein Konto mit Kreditkarte oder Mobiltelefon verifiziert hat.	Benötigt ein beliebiges Access Token.	true oder false
bio	Biographie des Benutzers	user_about_me oder friends_about_me	string
birthday	Geburtstag des Benutzers	user_about_me oder friends_about_me	string im Format MM/DD/YYY
education	Informationen zur Ausbildung des Benutzers	user_education_history oder friends_education_history	Array aus Objekten mit den Feldern year, type, school, year, degree, concenttraion, classes und with

Tabelle 3.1 Die Felder des User-Objekts (Forts.)

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
email	E-Mail-Adresse des Benutzers	Benötigt ein Access Token mit email-Berechtigung des aktuellen Benutzers. Das Auslesen der E-Mail-Adressen der Freunde des aktuellen Benutzers ist grundsätzlich nicht möglich!	string
hometown	Heimatstadt des Benutzers	user_hometown oder friends_hometown	Objekt, bestehend aus name und id
interested_in	geschlechtliche Interessen des Benutzers	user_relationship_details oder friends_relationship_details	Array, bestehend aus strings
location	aktuelle Stadt des Benutzers	user_location oder friends_location	Objekt, bestehend aus name und id
political	politische Ansichten des Benutzers	user_religion_politics oder friends_religion_politics	string
favorite_athletes	Lieblingsathleten des Benutzers	user_likes oder friends_likes	Array aus Objekten mit id und name
favorite_teams	Lieblingssportteams des Benutzers	user_likes oder friends_likes	Array aus Objekten mit id und name

Tabelle 3.1 Die Felder des User-Objekts (Forts.)

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
quotes	Lieblingszitate des Benutzers	user_about_me oder friends_about_me	string
relationship_status	Beziehungsstatus des Benutzers	user_relationships oder friends_relationships	string, z. B. Single, In a relationship, It's complicated etc.
religion	Religionszugehörigkeit des Benutzers	user_religion_politics oder friends_religion_politics	string
significant_other	(Ehe-)Partner des Benutzers	user_relationship_details oder friends_relationship_details	Objekt, bestehend aus name und id
video_upload_limits	maximale Länge und Dateigröße von Videos, die der Benutzer hochladen darf	nur verfügbar mit Access Token des aktuellen Benutzers	Objekt, bestehend aus length und size
website	URL der persönlichen Website des Benutzers	user_website oder friends_website	url
work	Informationen zur Karriere des Benutzers	user_work_history oder friends_work_history	Array aus Objekten mit den Feldern employer, location, position, start_date und end_date

Tabelle 3.1 Die Felder des User-Objekts (Forts.)

Hinweis: Immer wenn in der Spalte BENÖTIGTE ZUGRIFFSRECHTE zwei unterschiedliche Berechtigungen user_... oder friends_... angeführt sind, bedeutet dies: Die

Berechtigung `user_...` ist notwendig, um die entsprechende Information für den aktuellen Benutzer, mit dessen Access Token die Abfrage durchgeführt wird, zu erhalten. Die Berechtigung `friends_...` hingegen wird benötigt, um die entsprechende Information von Freunden des aktuellen Benutzers, mit dessen Access Token die Abfrage durchgeführt wird, zu erhalten. Fehlt die Angabe `friends_...`, bedeutet dies, dass dieses Feld ausschließlich für den aktuellen Benutzer, nicht aber dessen Freunde abgefragt werden kann.

Detaillierte Ausführungen zum Konzept der unterschiedlichen Berechtigungen für Profildaten von Benutzern und dessen Freunden finden Sie in Abschnitt 2.6.2, »Berechtigungen für Profildaten von Benutzern und Freunden«.

Verknüpfungen des User-Objekts

Objekte vom Typ `User` können in der Graph API mit einer Vielzahl an anderen Objekttypen verknüpft werden. Die Adressierung erfolgt dabei wie gewohnt nach dem Schema *https://graph.facebook.com/ObjektID/ObjektTyp*.

Die folgende Tabelle zeigt alle möglichen Objektverknüpfungen und – analog zu den bereits beschriebenen Datenfeldern – die Berechtigung, die zum Zugriff notwendig ist.

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`accounts`	Applikationen und Seiten, die der Benutzer administriert oder selbst erstellt hat	Benötigt ein Access Token mit `manage_pages`-Berechtigung.	Array aus Objekten mit den Feldern `id`, `name`, `access_token`, und `category`
`achievements`	in Spielen erreichte Leistungen des Benutzers	`user_games_activity` oder `friends_games_activity`	Array aus Objekten vom Typ `achievement`
`activities`	Aktivitäten des Benutzers	`user_activities` oder `friends_activities`	Array aus Objekten mit den Feldern `id`, `name`, `created_time`, und `category`
`albums`	Fotoalben des Benutzers	`user_photos` oder `friends_photos`	Array aus Objekten vom Typ `album`

Tabelle 3.2 Die Verknüpfungen des User-Objekts

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
apprequests	unbeantwortete Applikationseinladungen des Benutzers	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ apprequest
books	Bücher, die ein Benutzer in seinem Profil angegeben hat	user_likes oder friends_likes	Array aus Objekten mit den Feldern id, name created_time, und category
checkins	Orte, an denen ein Benutzer eingecheckt hat	user_checkins oder friends_checkins	Array aus Objekten vom Typ checkin
events	Veranstaltungen, die ein Benutzer besucht	user_events oder friends_events	Array aus Objekten mit den Feldern id, name start_time, end_time, location und rsvp_status
family	Familienbeziehungen des Benutzers	user_relationships oder friends_relationships	Array aus Objekten mit den Feldern id, name und relationship
feed	Pinnwand oder Wall des Benutzers (enthält Postings von Freunden)	read_stream	Array aus Objekten vom Typ post
friendlists	Freundeslisten des Benutzers	read_friendlists	Array aus Objekten mit den Feldern id und name

Tabelle 3.2 Die Verknüpfungen des User-Objekts (Forts.)

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`friendrequests`	die unbeantworteten Freundschaftsanfragen des Benutzers	`user_friendlists`	Array aus Objekten mit den Feldern `to`, `from`, `message`, `created_time` und `unread`
`friends`	die bestätigten Freunde des Benutzers	Benötigt ein beliebiges Access Token.	Array aus Objekten mit den Feldern `id` und `name`
`games`	Spiele, die ein Benutzer in seinem Profil angegeben hat	`user_likes`	Array aus Objekten mit den Feldern `id`, `name`, `created_time` und `category`
`groups`	Gruppen, in denen der Benutzer Mitglied ist	`user_groups` oder `friends_groups`	Array aus Objekten mit den Feldern `id`, `name`, `version`, `administrator` und `bookmark_order`
`home`	Newsfeed des Benutzers	`read_stream`	Array aus Objekten vom Typ `post`
`inbox`	Konversationen der Facebook-Mailbox des Benutzers	`read_mailbox`	Array aus Objekten vom Typ `thread`
`interests`	Interessen, die der Benutzer in seinem Profil angegeben hat	`user_interests` oder `friends_interests`	Array aus Objekten mit den Feldern `id`, `name`, `created_time` und `category`
`likes`	Seiten, auf denen ein Benutzer auf GEFÄLLT MIR geklickt hat	`user_likes` oder `friends_likes`	Array aus Objekten mit den Feldern `id`, `name`, `created_time` und `category`

Tabelle 3.2 Die Verknüpfungen des User-Objekts (Forts.)

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
links	Links, die ein Benutzer auf sein Profil gepostet hat	read_stream	Array aus Objekten vom Typ link
movies	Filme, die ein Benutzer in seinem Profil angegeben hat	user_likes oder friends_likes	Array aus Objekten mit den Feldern id, name, created_time und category
music	Musik, die ein Benutzer in seinem Profil gepostet hat	user_likes oder friends_likes	Array aus Objekten mit den Feldern id, name, created_time und category
mutualfriends	gemeinsame Freunde eines Benutzers mit einem zweiten Benutzer	Benötigt ein beliebiges Access Token und die ID des Benutzers, mit dem die Freunde verglichen werden sollen.	Array aus Objekten mit den Feldern id und name
notes	Notizen des Benutzers	read_stream	Array aus Objekten vom Typ note
notifications	Benachrichtigungen des Benutzers	manage_notifications	Array aus Objekten mit den Feldern id, from, to, created_time, updated_time, link, title, unread und application
outbox	Nachrichten im Postausgang der Facebook-Mailbox des Benutzers	read_mailbox	Array aus Objekten vom Typ message

Tabelle 3.2 Die Verknüpfungen des User-Objekts (Forts.)

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
payments	Facebook-Credits-Transaktionen, die ein Benutzer in der aktuellen Anwendung durchgeführt hat	Benötigt ein Anwendungs-Access-Token.	Array aus Objekten vom Typ order
permissions	Zugriffsberechtigungen, die der Benutzer der aktuellen Anwendung erteilt hat	keine	Array mit den einzelnen Zugriffsberechtigungen und boolean-Flag
photos	Fotos, auf denen der Benutzer markiert ist	user_photo_video_tags oder friends_photo_video_tags	Array aus Objekten vom Typ photo
picture	Profilbild des Benutzers	keine	HTTP-302-Redirect auf die Bilddatei
pokes	Pokes des Benutzers	read_mailbox	Array aus Objekten mit den Feldern to, from, created_time und type
posts	eigene Postings des Benutzers (wie feed, aber ohne Postings von Freunden)	Benötigt ein beliebiges Access Token oder read_stream, um nicht-öffentliche Postings zu erhalten.	Array aus Objekten vom Typ post
questions	Fragen, die der Benutzer gepostet hat	user_questions	Array aus Objekten vom Typ question

Tabelle 3.2 Die Verknüpfungen des User-Objekts (Forts.)

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
scores	Punktestände des Benutzers in Spielen	user_games_activity oder friends_games_activity	Array aus Objekten mit den Feldern user, application, score und type
statuses	Status-Updates des Benutzers (wie posts, aber ohne Fotos, Videos etc.)	read_stream	Array aus Objekten vom Typ status message
tagged	Posts, in denen der Benutzer markiert wurde	read_stream	Array aus Objekten mit den Feldern id, from, to, picture, link, name, caption, description, icon, actions, type, application, created_time und updated_time
television	TV-Shows, die ein Benutzer in seinem Profil gepostet hat	user_likes oder friends_likes	Array aus Objekten mit den Feldern id, name, category und created_time
updates	Updates in der Facebook-Mailbox des Benutzers	read_mailbox	Array aus Objekten vom Typ message
videos	Videos, in denen der Benutzer markiert wurde	user_videos oder friends_videos	Array aus Objekten vom Typ video

Tabelle 3.2 Die Verknüpfungen des User-Objekts (Forts.)

Hinweis: Auch hier gilt: Die Berechtigungen der Form user_... sind notwendig, um die entsprechende Verknüpfungen für den aktuellen Benutzer zu erhalten. Die Berechtigungen der Form friends_... hingegen werden benötigt, um die entsprechende Verknüpfungen von Freunden des aktuellen Benutzers zu erhalten.

Obwohl die Graph API keine Möglichkeiten zum Erstellen von neuen Benutzerobjekten bietet (dies muss immer durch eine reale Person über das Web Frontend von Facebook.com erfolgen), können dafür die meisten verknüpften Objekte mit Schreibzugriff veröffentlicht werden.

Veröffentlichen von Alben

Um ein neues Fotoalbum auf einem Benutzerprofil zu veröffentlichen, genügt es, den API-Endpunkt `https://graph.facebook.com/BenutzerID/albums` mit einem HTTP-POST-Zugriff anzusprechen. Dabei stehen folgende POST-Parameter zur Verfügung, die die Eigenschaften des Albums festlegen:

Parameter	Beschreibung	Typ	Pflichtfeld
`name`	Name des Albums	`string`	ja
`message`	Beschreibung des Albums	`string`	nein
`privacy`	Privatsphäre-Einstellungen des Albums	ein JSON-codiertes Objekt, das die Privatsphäre-Einstellungen detailliert regelt	nein

Tabelle 3.3 Parameter zum Veröffentlichen von Fotoalben

Das `privacy`-Objekt kann dabei folgende Felder enthalten:

- `value` – Die primäre Privatsphäre-Einstellung – kann auf `EVERYONE` (Album ist vollkommen öffentlich), `CUSTOM` (benutzerdefinierte Einstellungen), `ALL_FRIENDS` (Album ist für die Freunde des Erstellers sichtbar), `NETWORKS_FRIENDS` (Album ist für Freunde und Netzwerke des Benutzers sichtbar), `FRIENDS_OF_FRIENDS` (Album ist für Freunde und deren Freunde sichtbar) und `SELF` (Album ist nur für den Benutzer selbst sichtbar) gesetzt werden.
- `friends` – Wurde zuvor bei `value` die Einstellung `CUSTOM` gewählt, kann hier festgelegt werden, welche Benutzer das Album sehen können. Mögliche Werte sind `EVERYONE`, `SOME_FRIENDS`, `ALL_FRIENDS`, `NETWORKS_FRIENDS`, `FRIENDS_OF_FRIENDS` und `NO_FRIENDS` (nur für Netzwerke sichtbar).
- `networks` – Wurde zuvor bei `value` die Einstellung `CUSTOM` gewählt, kann hier eine kommaseparierte Liste an Netzwerk-IDs angegeben werden, die das Album sehen kann. Der Wert 1 bedeutet, dass alle Netzwerke auf das Album zugreifen können.
- `allow` – Wurde zuvor bei `friends` die Einstellung `SOME_FRIENDS` gewählt, können hier mit einer kommaseparierten Liste die IDs jener Benutzer angegeben werden, die Zugriff auf das Album erhalten sollen.

- `deny` – Wurde zuvor bei `friends` die Einstellung `SOME_FRIENDS` gewählt, können hier mit einer kommaseparierten Liste die IDs jener Benutzer angegeben werden, die keinen Zugriff auf das Album erhalten sollen.

Hinweis: Zum Veröffentlichen eines Albums sind – anders als zum Hochladen von Fotos – keine erweiterten Berechtigungen notwendig!

War das Veröffentlichen des Albums erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des neuen Albums.

Veröffentlichen und Lesen von Applikationsanfragen

`apprequest`-Objekte nehmen eine Sonderstellung ein – sie treten lediglich in Verknüpfung mit einem Objekt vom Typ `user` auf und repräsentieren Applikationseinladungen, die ein Benutzer an einen Freund versendet hat (das Versenden von Applikationsanfragen wird detailliert in Abschnitt 4.4.5, »Requests-Dialog«, behandelt). Oft kann es in Anwendungen notwendig sein, die Applikationseinladungen, die ein Benutzer erhalten hat, auszulesen – etwa um dem Benutzer mehrere unbeantwortete Einladungen in einem Menü anzuzeigen. Dazu dient ein GET-Zugriff auf den API-Endpunkt `https://graph.facebook.com/apprequests`, der mit dem Access Token des Benutzers signiert ist, dessen unbeantwortete Einladungen ausgelesen werden sollen. Das Ergebnis dieses Zugriffs ist ein Array aus Objekten mit folgenden Feldern:

Name	Beschreibung	Typ
`id`	Objekt-ID der Applikationsanfrage	`string`
`application`	Applikation, für die die Einladung versendet wurde	Objekt mit den Feldern `id`, `name`, `canvas_name` und `namespace`
`to`	Benutzer, an den die Einladung versendet wurde	Objekt mit den Feldern `id` und `name`
`from`	Benutzer, der die Einladung versendet hat	Objekt mit den Feldern `id` und `name`
`message`	Beschreibungstext der Einladung	`string`
`created_time`	Zeitstempel der Erstellung der Einladung	UNIX-Zeitstempel

Tabelle 3.4 Felder beim Auslesen von Applikationsanfragen

Name	Beschreibung	Typ
type	Typ der Einladung – enthält immer apprequest	string
data	Daten, die beim Versenden der Einladung übergeben wurden	string

Tabelle 3.4 Felder beim Auslesen von Applikationsanfragen (Forts.)

Applikationsanfragen werden normalerweise durch einen Benutzerdialog ausgelöst, wie in Abschnitt 4.4.5, »Requests-Dialog«, beschrieben. Die Graph API bietet allerdings auch eine Möglichkeit, apprequest-Objekte per API-Zugriff zu erzeugen.

Achtung: Eingeladene Benutzer erhalten für auf diese Weise erzeugte Applikationsanfragen *keine Benachrichtigung* in der Kopfzeile von Facebook.com, sondern können diese lediglich manuell über http://www.facebook.com/reqs.php abrufen. Aus diesem Grund ist das Erstellen von Applikationsanfragen per API eher unüblich.

Möchten Sie dieser Einschränkung zum Trotz dennoch eine Anfrage per API erzeugen, genügt ein POST-Zugriff auf den Endpunkt https://graph.facebook.com/BenutzerID/apprequests. Als Benutzer-ID übergeben Sie dabei die numerische ID oder den eindeutigen Namen des *einzuladenden Benutzers*. Der Zugriff muss mit dem Anwendungs-Access-Token anstelle des Benutzer-Access-Tokens signiert werden und enthält im Body folgende Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
message	Beschreibungstext der Einladung	string	ja
data	Daten, die beim Versenden der Einladung übergeben werden sollen	string	nein

Tabelle 3.5 Felder zum Veröffentlichen von Applikationsanfragen

War das Erzeugen der Applikationsanfrage erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID der Anfrage und im Feld to die Benutzer-ID des Empfängers.

Hat ein Benutzer eine eingehende Einladung angenommen, sollte die Anwendung die Applikationsanfrage anschließend löschen. Dies wird nicht von Facebook selbst übernommen, sondern ist die Aufgabe des Applikationsentwicklers. Es genügt hierzu, einen DELETE-Zugriff auf den API-Endpunkt auf eine URL der Form https://graph.facebook.com/RequestID_BenutzerID durchzuführen. Der Parameter wird dabei aus der ID der Anfrage und der Benutzer ID – verknüpft mit einem Unterstrich

– gebildet. Löschzugriffe auf Applikationsanfragen müssen mit dem Anwendungs- oder Benutzer-Access-Token signiert sein.

Veröffentlichen von Checkins

Seit 2010 bietet Facebook seinen Benutzern die Möglichkeit, ihren aktuellen geographischen Standort mittels *Checkin* bekannt zu geben. Die `checkin`-Objekte, die diese Standortbestimmungen repräsentieren, können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/checkins` veröffentlicht werden.

Das zur Signatur verwendete Access Token muss vom eincheckenden Benutzer mit der Zugriffsberechtigung `publish_checkins` autorisiert worden sein. Folgende Parameter spezifizieren den Checkin näher:

Parameter	Beschreibung	Typ	Pflichtfeld
`place`	Objekt-ID des Platzes bzw. der Seite, auf der eingecheckt werden soll	`string`	ja
`coordinates`	die geographische Position des Benutzers, der eingecheckt hat	Array mit den Feldern `latitude` und `longitude`	ja
`Tags`	Liste von Freunden, die gemeinsam mit dem Benutzer eingecheckt haben	Benutzer-IDs als kommaseparierte Liste	nein
`description`	Beschreibungstext	`string`	nein
`link`	externer Link zum Checkin	`string`	nein
`Picture`	externer Link zu einem Bild des Checkins	`string`	nein

Tabelle 3.6 Felder zum Veröffentlichen von Checkins

Beim Veröffentlichen von Checkins sollten Sie Folgendes beachten:

- Mit dem Start des Checkin-Features 2010 hat Facebook Orte anders als herkömmliche Seiten behandelt. Mittlerweile jedoch werden Orte als normale Seiten repräsentiert.
- Der aktuelle Aufenthaltsort des Benutzers muss als Koordinatenpaar übergeben werden. Weicht dieser stark vom Standort der Seite bzw. des Ortes ab, wird der Checkin nicht veröffentlicht.

War das Erzeugen des `checkin`-Objekts erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Checkins.

Veröffentlichen von Veranstaltungen

Um eine Veranstaltung vom Objekttyp `event` auf einem Benutzerprofil zu veröffentlichen, müssen Sie einen POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/events` durchführen. Das zur Signatur verwendete Access Token muss vom Benutzer mit der Zugriffsberechtigung `create_event` autorisiert worden sein.

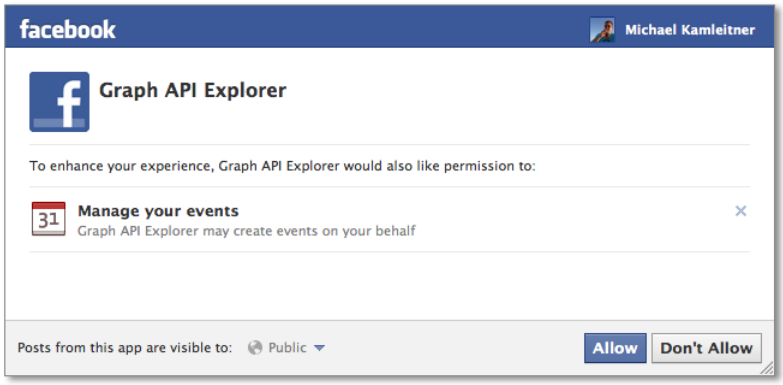


Abbildung 3.4 Autorisierung einer Anwendung mit der Berechtigung »create_events«

Folgende Parameter spezifizieren die Veranstaltung näher:

Parameter	Beschreibung	Typ	Pflichtfeld
`name`	Name der Veranstaltung	`string`	ja
`start_time`	UNIX-Zeitstempel des Beginns der Veranstaltung	`UNIX timestamp`	ja
`end_time`	UNIX-Zeitstempel des Endes der Veranstaltung	`UNIX timestamp`	nein
`message`	Beschreibung der Veranstaltung	`string`	nein
`location`	Ort der Veranstaltung	`string`	nein
`privacy_type`	Privatsphäre-Einstellung – `OPEN`, `CLOSED` oder `SECRET`	`string`	nein

Tabelle 3.7 Felder zum Veröffentlichen von Veranstaltungen

Die Privatsphäre-Einstellungen im Parameter `privacy_type` bieten folgende Möglichkeiten:

- `OPEN` – Die Veranstaltung ist offen, jeder Benutzer kann teilnehmen.
- `CLOSED` – Die Veranstaltung ist öffentlich sichtbar, Benutzer können aber nur mit Einladung teilnehmen.
- `SECRET` – Die Veranstaltung ist versteckt, Benutzer können nur mit Einladung teilnehmen.

War das Erzeugen der Veranstaltung erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID der Veranstaltung.

Veranstaltungen, die auf einem Benutzerprofil veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/EventID` wieder gelöscht werden.

Veröffentlichen von Freundeslisten

Facebook bietet jedem Benutzer die Möglichkeit, Freunde in Freundeslisten zu organisieren. Um Freundeslisten über die Graph API einlesen zu können, muss das verwendete Access Token mit der Zugriffsberechtigung `manage_friendlists` autorisiert werden. Das Veröffentlichen von Freundeslisten benötigt hingegen die Berechtigung `manage_friendlists`.

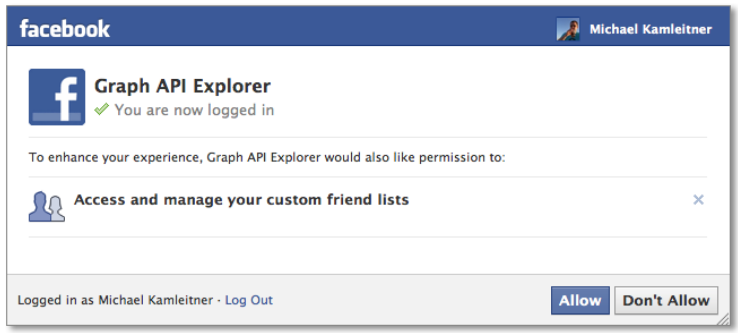


Abbildung 3.5 Autorisierung einer Anwendung mit den Berechtigungen »manage_friendlists« und »read_friendlists«

Um Freundeslisten vom Objekttyp `friendlist` zu erstellen, muss ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/friendlists` durchgeführt werden. Mit dem Parameter `name` wird die Bezeichnung der Freundesliste definiert:

Parameter	Beschreibung	Typ	Pflichtfeld
name	Name der Freundesliste	string	ja

Tabelle 3.8 Felder zum Veröffentlichen von Freundeslisten

War das Erzeugen der Freundesliste erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID der Freundesliste.

Veröffentlichen von Links

Anwendungen können im Namen eines Benutzers Objekte vom Typ link auf der Pinnwand des Benutzers veröffentlichen. Wie auch zum Veröffentlichen von Wall-Posts ist dazu ein Access Token mit der erweiterten Berechtigung publish_stream notwendig.

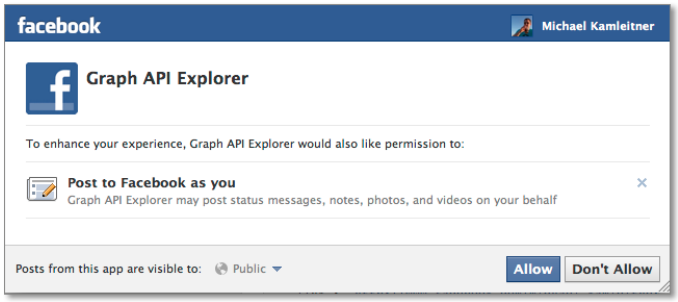


Abbildung 3.6 Autorisierung einer Anwendung mit der Berechtigung »publish_stream«

Zum Veröffentlichen muss ein POST-Zugriff auf den Endpunkt https://graph.facebook.com/me/links mit folgenden Parametern gesendet werden:

Parameter	Beschreibung	Typ	Pflichtfeld
link	URL des zu veröffentlichenden Links	string	ja
message	Beschreibungstext des zu veröffentlichenden Links	string	nein

Tabelle 3.9 Felder zum Veröffentlichen von Links

Hinweis: Das Veröffentlichen von Objekten des Typs link auf der Pinnwand von Freunden des aktuellen Benutzers ist *derzeit nicht möglich*! Die Graph API ist in diesem Aspekt inkonsistent, immerhin ermöglicht die Berechtigung publish_stream sehr wohl die Veröffentlichung von normalen Wall-Postings oder Wall-Postings, die einen Link enthalten.

War das Erzeugen des Links erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Links.

Links, die auf dem Benutzerprofil veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/BenutzerID_LinkID` wieder gelöscht werden. Die ID wird dabei aus *Benutzer- und Link-ID zusammengesetzt*!

Veröffentlichen von Notizen

Ähnlich wie das Veröffentlichen von Links erfordert auch das Veröffentlichen von Objekten des Typs `notes` die Zugriffsberechtigung `publish_stream`. Auch Notizen können ausschließlich auf der Pinnwand des aktuellen Benutzers – und nicht etwa auf den Pinnwänden von dessen Freunden – veröffentlicht werden. Der API-Endpunkt `https://graph.facebook.com/me/notes` nimmt folgende Parameter bei POST-Zugriffen entgegen:

Parameter	Beschreibung	Typ	Pflichtfeld
`subject`	Betreff der zu veröffentlichenden Notiz	`string`	ja
`message`	Beschreibungstext des zu veröffentlichenden Links	`string`	ja

Tabelle 3.10 Felder zum Veröffentlichen von Notizen

Hinweis: der `message`-Parameter darf dabei einfache HTML-Auszeichnungen wie `<b>`, `<i>`, `<u>`, `<blockquote>`, `<ul>` und `<li>` enthalten.

War das Erzeugen der Notiz erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID der Notiz.

Notizen, die auf dem Benutzerprofil veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/NotizID` wieder gelöscht werden.

Veröffentlichen von Fotos

Sie können Fotos auf der Pinnwand des Benutzers oder auf Pinnwänden von Freunden mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/photos` veröffentlichen. Dazu ist ein Benutzer-Access-Token, das mit der erweiterten Zugriffsberechtigung `publish_stream` signiert ist, notwendig. Folgende Parameter werden dabei übergeben:

Parameter	Beschreibung	Typ	Pflichtfeld
source	das zu veröffentlichende Foto als binäre Zeichenkette	multipart/ form-data	ja
message	Beschreibungstext des zu veröffentlichenden Fotos	string	ja

Tabelle 3.11 Felder zum Veröffentlichen von Fotos

Hinweis: Das Codieren des Fotos als binäre Zeichenkette wird dabei üblicherweise von der `curl()`-Funktion übernommen.

War das Veröffentlichen des Fotos erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Fotos.

Beispiel: Veröffentlichen von Fotos auf der Pinnwand

Probieren Sie nun folgendes Code-Beispiel aus, das ein einfaches Upload-Formular für Fotos bereitstellt, die auf der Pinnwand des aktuellen Benutzers veröffentlicht werden sollen:

```
<?
include_once("tools.php");

if (isset($_FILES["source"]["tmp_name"])) {
   $url = "https://graph.facebook.com/me/photos";
   $params = array(
      "access_token" => "...",
      "message" => "Testfoto: ".$_FILES["source"]["name"],
      "source" => "@".$_FILES["source"]["tmp_name"]
   );

   $result = json_decode(curl($url, $params, "POST"));
   print "<pre>";
   print_r($result);
   print "</pre>";
} else { ?>
   <form method="post" enctype="multipart/form-data" action="">
   Image-File: <input type="file" name="source"/><br/>
   <input type="submit"/>
   </form>
<? } ?>
```

Listing 3.6 Code zum Upload und Veröffentlichen von Fotos

Wie funktioniert dieses Beispiel?

- Zu Beginn wird dem Benutzer ein HTML-Formular präsentiert, das lediglich ein `<input>`-Feld zum Hochladen der Bilddatei enthält. Wichtig ist dabei, dass im `<form>`-Element das Formular-Encoding mit dem Attribut `enctype="multipart/form-data"` eingestellt wird – ansonsten würde der Upload nicht funktionieren.
- Anhand des gesetzten PHP-Arrays `$_FILES["source"]` erkennt das Skript, dass ein Bild im Formular hochgeladen wurde. Im Folgenden werden die zum Veröffentlichen des Fotos notwendigen Parameter im Array `$params` zusammengestellt – die Bildunterschrift `message` enthält dabei den Dateinamen des hochgeladenen Fotos, in `source` wird der temporäre Pfad zur hochgeladenen Bilddatei angegeben. Das vorangestellte `@`-Symbol bewirkt dabei, dass `curl()` eigenständig den Inhalt der Bilddatei ausliest und in korrekter Codierung an die Graph API übergibt.
- Die Veröffentlichung erfolgt anschließend durch einen POST-Zugriff auf den Endpunkt `https://graph.facebook.com/me/photos`. Das Ergebnis enthält die numerische ID des veröffentlichten Fotos.
- Wichtig: Das zur Veröffentlichung verwendete Access Token muss dabei die erweiterte Berechtigung `publish_stream` bereitstellen.

Veröffentlichen von Videos

Videos können auf der Pinnwand des Benutzers oder auf Pinnwänden von Freunden mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/videos` veröffentlicht werden. Dazu ist ein Benutzer-Access-Token, das mit der erweiterten Zugriffsberechtigung `publish_stream` signiert ist, notwendig. Übergeben Sie dabei folgende Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`source`	das zu veröffentlichende Video als binäre Zeichenkette	`multipart/form-data`	ja
`title`	Titel des zu veröffentlichenden Videos	`string`	nein
`description`	Beschreibungstext des zu veröffentlichenden Videos	`string`	nein

Tabelle 3.12 Felder zum Veröffentlichen von Videos

Hinweis: Das Codieren des Videos als binäre Zeichenkette und das Veröffentlichen von Videos werden genauer in Abschnitt 3.2.8, »Das Videoobjekt«, erläutert.

Veröffentlichen und Lesen von Wall-Postings

Wall-Postings können auf der Pinnwand des Benutzers oder auf den Pinnwänden von Freunden mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/feed` erzeugt werden. Dazu ist ein Benutzer-Access-Token notwendig, das vom Benutzer mit der erweiterten Berechtigung `publish_stream` autorisiert wurde. Die API-Methode unterstützt folgende Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	die Benutzernachricht des Wall-Postings	`string`	ja
`link`	der Link des Wall-Postings	`string`	nein
`picture`	URL zu einem Thumbnail-Bild	`string`	nein
`name`	der Ankertext, der für `link` angezeigt wird	`string`	nein
`caption`	die Überschrift des Wall-Postings	`string`	nein
`description`	der Beschreibungstext des Wall-Postings	`string`	nein
`actions`	Action-Links des Wall-Postings	Array aus Objekten mit den Feldern `name` und `link`	nein
`privacy`	Privatsphäre-Einstellungen des Wall-Postings	`string`	nein
`object_attachment`	Verknüpfung eines existierenden Fotoobjekts mit dessen numerischer ID. Der Benutzer muss der Besitzer des Fotos sein, und das Foto darf nicht selbst ein Attachment einer Nachricht sein.	`string`	nein

Tabelle 3.13 Felder zum Veröffentlichen von Wall-Postings

Die einzelnen möglichen Komponenten eines Wall-Postings sollen in folgendem Beispiel mit dem Graph API Explorer demonstriert werden:

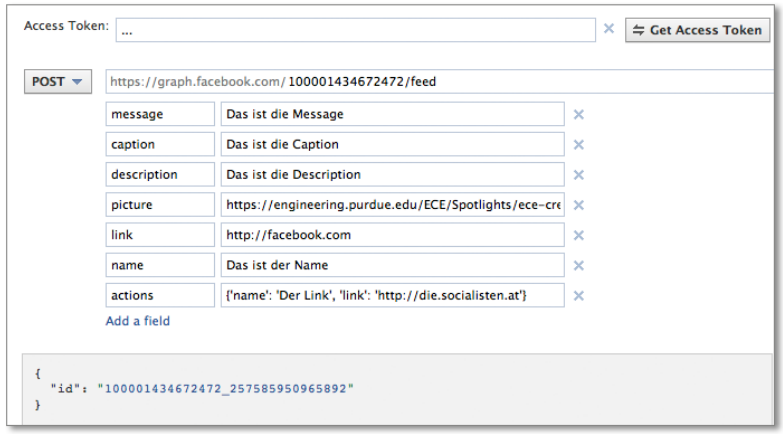


Abbildung 3.7 Verwendung des Graph API Explorers zum Veröffentlichen eines Wall-Postings

Das Beispiel nutzt die meisten möglichen Parameter:

- In `message` wird die benutzerspezifische Nachricht gesetzt, die über dem Wall-Posting in Zusammenhang mit dem Profilbild und Namen des sendenden Benutzers angezeigt wird (»Das ist die Message«).
- In `caption` wird die erste Zeile des Posting-Textes definiert (»Das ist die Caption«). Facebook stellt diese optisch kaum unterscheidbar vom restlichen Posting-Text dar.
- Der restliche Posting-Text wird in `description` definiert (»Das ist die Description«).
- Die in `link` angegebene URL wird über der Caption als klickbarer Hyperlink angezeigt. Wird `name` ebenfalls übergeben, wird anstelle der URL der Inhalt von `name` als Klicktext verwendet.
- Das Thumbnail, auf das mit der in `picture` übergebenen URL verwiesen wird, erscheint links neben dem Wall-Posting-Text.
- Der als JSON-Array in `actions` übergebene Action-Link wird unter dem Wall-Posting neben dem KOMMENTIEREN-Link angezeigt (»Der Link«) und mit der ebenfalls übergebenen URL verlinkt.

Abbildung 3.8 Darstellung des veröffentlichten Wall-Postings im Newsfeed

War das Veröffentlichen des Postings erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Postings.

Postings, die auf einem Benutzerprofil veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/BenutzerID_PostingID` wieder gelöscht werden. Die ID wird dabei aus *Benutzer- und Posting-ID zusammengesetzt*!

Veröffentlichen von Status-Updates

Status-Updates sind aus der Sicht der Graph API eine besondere Form des Wall-Postings, das ausschließlich eine Benutzernachricht enthält. Analog zum Veröffentlichen von Wall-Postings ist daher ein Benutzer-Access-Token mit der Berechtigung `publish_stream` notwendig. Das Veröffentlichen des Status-Updates erfolgt mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/feed` und akzeptiert lediglich einen Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	die Benutzernachricht des Status-Updates	`string`	ja

Tabelle 3.14 Felder zum Veröffentlichen von Status-Updates

Status-Updates können auf der Pinnwand des Benutzers und auf den Pinnwänden seiner Freunde veröffentlicht werden.

War das Veröffentlichen des Status-Updates erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Status-Updates.

Status-Updates, die auf einem Benutzerprofil veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/BenutzerID_PostingID` wieder gelöscht werden. Die ID wird dabei aus *Benutzer- und Posting-ID zusammengesetzt*!

Veröffentlichen und Lesen von Punkteständen

Der Objekttyp `scores` speichert Punktestände, die ein Facebook-Benutzer in einem Facebook-Spiel erzielt hat. Um Punktestände von Benutzern zu lesen, ist der API-Endpunkt `https://graph.facebook.com/BenutzerID/scores` notwendig. Welche Punktestände hierbei abrufbar sind, hängt vom verwendeten Access Token ab:

- Ein Access Token ohne erweiterte Berechtigungen liefert ausschließlich die Punktestände des aktuellen Benutzers in der aktuellen Anwendung zurück.

- Ein Access Token mit der erweiterten Berechtigung `user_games_activity` liefert alle Punktestände des aktuellen Benutzers aus allen Anwendungen.
- Ein Access Token mit der erweiterten Berechtigung `friends_games_activity` liefert alle Punktestände der Freunde des aktuellen Benutzers aus allen Anwendungen.

Der API-Aufruf liefert ein JSON-Array aus Objekten mit folgenden Feldern:

Name	Beschreibung	Typ
`user`	Benutzer, der den Punktestand erreicht hat	Objekt mit den Feldern `id` und `name`
`score`	numerischer Wert des Punktestandes	`integer`
`application`	Anwendung, in der der Punktestand erreicht wurde	Objekt mit den Feldern `id` und `name`
`type`	Typ des Punktestandes, derzeit immer `score`	`string`

Tabelle 3.15 Felder zum Veröffentlichen von Punkteständen

Um Punktestände zu veröffentlichen, muss der Benutzer die Anwendung mit der erweiterten Berechtigung `publish_actions` autorisieren. Dann genügt das Anwendungs-Access-Token, um Punktestände mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/scores` zu veröffentlichen. Übergeben Sie dazu folgende Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`score`	numerischer Wert des Punktestandes	`integer`	ja

Tabelle 3.16 Felder zum Veröffentlichen von Punkteständen

War die Veröffentlichung des Punktestandes erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Der aktuelle Punktestand des Benutzers kann mit einem DELETE-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/scores` gelöscht werden. Auch hierzu muss der Benutzer die Anwendung mit der erweiterten Berechtigung `publish_actions` autorisiert haben, und es muss ein Anwendungs-Access-Token verwendet werden. Der Löschaufruf erfordert keine Parameter und liefert `true` zurück, wenn er erfolgreich durchgeführt wurde.

Veröffentlichen und Lesen von Leistungen

Neben Punkteständen können Entwickler von Facebook-Spielen bestimmte vom Spieler erreichte »Leistungen« im sozialen Graphen speichern (z. B. »Level 2 erreicht«). Leistungen werden mittels GET-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/achievements` ausgelesen, wobei das Ergebnis auch hier vom verwendeten Access Token abhängig ist:

- Ein Access Token ohne erweiterte Berechtigungen liefert ausschließlich die Leistungen des aktuellen Benutzers in der aktuellen Anwendung zurück.
- Ein Access Token mit der erweiterten Berechtigung `user_games_activity` liefert alle Leistungen des aktuellen Benutzers aus allen Anwendungen oder Spielen.
- Ein Access Token mit der erweiterten Berechtigung `friends_games_activity` liefert alle Leistungen der Freunde des aktuellen Benutzers aus allen Anwendungen oder Spielen.

Der API-Aufruf liefert ein JSON-Array aus Objekten mit folgenden Feldern:

Name	Beschreibung	Typ
`id`	numerische ID des Objekts vom Typ `achievement`	`string`
`from`	Benutzer, der die Leistung erreicht hat	Objekt mit den Feldern `id` und `name`
`start_time`	Zeitstempel vom Beginn der Leistung	Zeitstempel im ISO-8601-Format
`end_time`	Zeitstempel vom Ende der Leistung	Zeitstempel im ISO-8601-Format
`publish_time`	Zeitstempel der Veröffentlichung der Leistung	Zeitstempel im ISO-8601-Format
`application`	Anwendung, in der die Leistung erreicht wurde	Objekt mit den Feldern `id` und `name`
`achievement`	die erreichte Leistung	Objekt mit den Feldern `id`, `url`, `type` und `title`
`likes`	Anzahl der Gefällt-mir-Klicks	`number`
`comments`	Anzahl der Kommentare	`number`

Tabelle 3.17 Felder von Leistungen

Ein Beispiel einer Leistung aus dem Spiel »CityVille« sieht etwa folgendermaßen aus:

```
{
  "data": [
    {
      "id": "2336911821985",
      "from": {
        "id": "122555...",
        "name": "Michaela W..."
      },
      "start_time": "2011-11-25T08:37:59+0000",
      "end_time": "2011-11-25T08:37:59+0000",
      "publish_time": "2011-11-25T08:37:59+0000",
      "application": {
        "id": "291549705119",
        "name": "CityVille"
      },
      "achievement": {
        "id": "10150319361197660",
        "url": "http://fb-client-0.cityville.zynga.com/
           fbAchievement.php?id=77",
        "type": "game.achievement",
        "title": "Level 77 Badge"
      },
      "likes": {
        "count": 0
      },
      "comments": {
        "count": 0
      }
    },
    ...
```

Listing 3.7 JSON-Array eines Leistungsobjekts

Analog zu Punkteständen muss ein Benutzer eine Anwendung mit der erweiterten Berechtigung `publish_actions` autorisieren, damit Sie unter Verwendung des Anwendungs-Access-Tokens Leistungen veröffentlichen können. Dies können Sie über einen POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/achievements` tun. Als Parameter muss dabei lediglich die URL übergeben werden, unter der die Leistung als Open-Graph-Objekt repräsentiert wird:

Parameter	Beschreibung	Typ	Pflichtfeld
achievement	eindeutige URL zum Open-Graph-Objekt, das die Leistung repräsentiert	string	ja

Tabelle 3.18 Felder zum Veröffentlichen erreichter Leistungen

Ist das Veröffentlichen der Leistung erfolgreich durchgeführt werden, enthält das Ergebnis des Aufrufs `true`.

Um eine Leistung zu löschen, führen Sie einen DELETE-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/achievements` aus. Als Parameter übergeben Sie wiederum die URL zum Open-Graph-Objekt der Leistung:

Parameter	Beschreibung	Typ	Pflichtfeld
achievement	eindeutige URL zum Open-Graph-Objekt, das die Leistung repräsentiert	string	ja

Tabelle 3.19 Felder zum Löschen erreichter Leistungen

Ist das Löschen der Leistung erfolgreich durchgeführt worden, enthält das Ergebnis des Aufrufs `true`.

Lesen der Freunde

Ein beliebiges, vom Benutzer autorisiertes Access Token erlaubt das Auslesen aller Freunde des *aktuellen Benutzers* mit einem GET-Zugriff auf den Endpunkt `https://graph.facebook.com/me/friends`. Das Ergebnis enthält ein Array von Objekten mit den Feldern `name` und `id` der Freunde.

Hinweis: Die vollständige Freundesliste kann immer nur für den eigenen, aktuellen Benutzer ausgelesen werden!

Um zu prüfen, ob ein Benutzer mit einem bestimmten anderen Benutzer befreundet ist, kann der Endpunkt `https://graph.facebook.com/me/friends/BenutzerID` verwendet werden. Sind die Benutzer befreundet, wird ein Array mit den Feldern `name` und `id` des Freundes zurückgegeben. Sind die Benutzer nicht befreundet, ist das Ergebnis leer.

Hinweis: Es können alle Freundschaftsbeziehungen geprüft werden, die zumindest einen Freund des aktuellen Benutzers einschließen!

Zusätzlich bietet die Graph API die Möglichkeit, gemeinsame Freunde von zwei Benutzern zu ermitteln. Dazu ist ein GET-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/mutualfriends/BenutzerID` ausreichend. Das Ergebnis ist ein Array an Objekten mit den Feldern `id` und `name`.

Lesen der Freundschaftsanfragen

Um die ausstehenden Freundschaftsanfragen des aktuellen Benutzers auszulesen, muss ein GET-Zugriff auf den API-Endpunkt `https://graph.facebook.com/me/friendrequests` unter Verwendung eines Access Tokens mit der erweiterten Berechtigung `read_requests` durchgeführt werden.

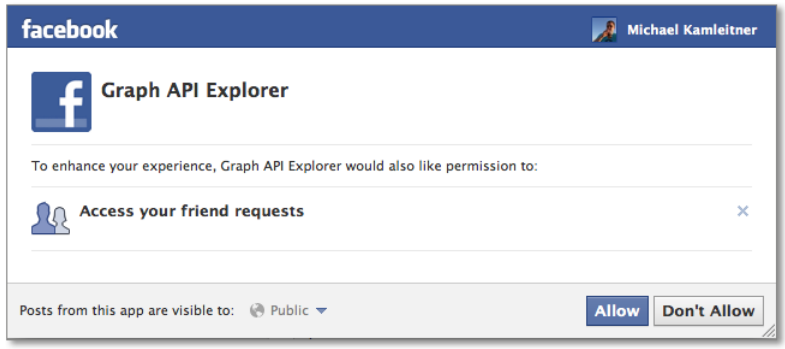


Abbildung 3.9 Autorisierung einer Anwendung mit der Berechtigung »read_requests«

Das Ergebnis dieses Aufrufs ist ein Array an Objekten mit folgenden Feldern:

Feldname	Beschreibung	Typ
`from`	Name und ID des anfragenden Benutzers	Objekt mit den Feldern `id` und `name`
`to`	Name und ID das angefragten Benutzers	Objekt mit den Feldern `id` und `name`
`created_time`	Zeitstempel vom Zeitpunkt der Erstellung der Anfrage	Zeitstempel im ISO-8601-Format
`message`	Nachricht des Benutzers, der die Anfrage gesendet hat	`string`
`Unread`	Hat der angefragte Benutzer die Anfrage gelesen?	`boolean`

Tabelle 3.20 Felder von Freundschaftsanfragen

Lesen von Benachrichtigungen

Benachrichtigungen werden von Facebook verwendet, um Benutzer auf Aktivitäten in Anwendungen hinzuweisen. Benachrichtigungen werden als rot hervorgehobene Zahl in der Kopfzeile von Facebook dargestellt:

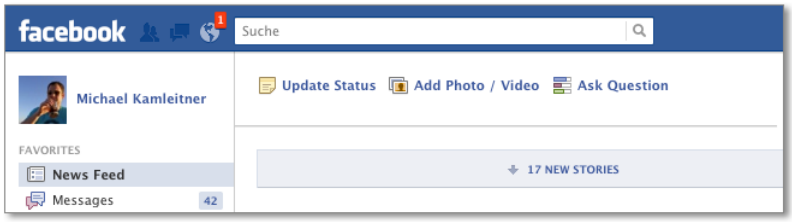


Abbildung 3.10 Anzeige neuer Benachrichtigungen in der Kopfzeile von Facebook

Wichtig: Benachrichtigungen können nicht unmittelbar durch eine Anwendung erzeugt werden! Wenn ein Benutzer seine Freunde über den Dialog `FB.ui` (siehe Abschnitt 4.4.5, »Requests-Dialog«) zu einer Anwendung einlädt, werden diese aber normalerweise von Facebook als Benachrichtigung angezeigt. Ausnahmen stellen in rascher Abfolge abgesetzte Einladungen oder Einladungen zu Anwendungen, die Facebook als spamkritisch betrachtet, dar – diese werden meist nicht als Benachrichtigung beim eingeladenen Benutzer angezeigt.

Um die Benachrichtigungen eines Benutzers lesen zu können, benötigen Sie ein Access Token, das mit der erweiterten Berechtigung `manage_notifications` signiert wurde.

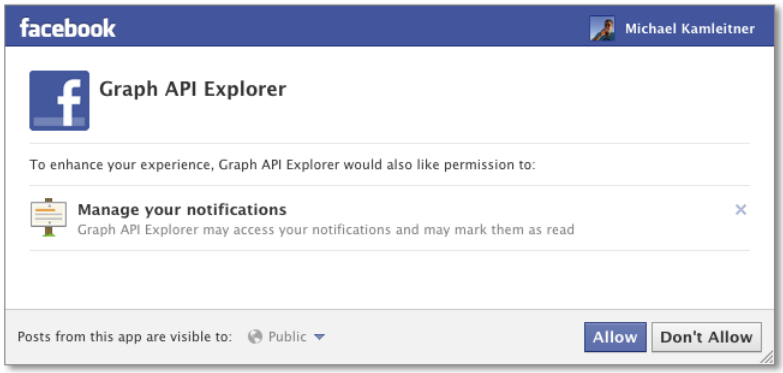


Abbildung 3.11 Autorisierung einer Anwendung mit der Berechtigung »manage_notifications«

Den entsprechenden API-Endpunkt finden Sie unter `https://graph.facebook.com/me/notifications`. Über den GET-Parameter `include_read`, der die Werte 0 und 1 annehmen darf, können Sie dabei steuern, ob bereits gelesene Benachrichtigungen

im Ergebnis enthalten sein sollen. Das Ergebnis ist ein Array aus Objekten, die folgende Felder enthalten:

Feldname	Beschreibung	Typ
`id`	numerische ID der Benachrichtigung	`string`
`from`	Name und ID des Benutzers, der die Benachrichtigung gesendet hat	Objekt mit den Feldern `id` und `name`
`to`	Name und ID des Benutzers, der die Benachrichtigung empfangen hat	Objekt mit den Feldern `id` und `name`
`created_time`	Zeitstempel, an dem die Benachrichtigung erzeugt wurde	Zeitstempel im ISO-8601-Format
`updated_time`	Zeitstempel, an dem die Benachrichtigung zuletzt aktualisiert wurde	Zeitstempel im ISO-8601-Format
`title`	Titel der Benachrichtigung	`string`
`link`	Link zu dem Objekt, auf das sich die Benachrichtigung bezieht	`string`
`application`	Name und ID der Applikation, durch die die Benachrichtigung versendet wurde	Objekt mit den Feldern `id` und `name`
`unread`	Kennzeichnet, ob eine Benachrichtigung bereits gelesen wurde (1) oder nicht (0).	`boolean`

Tabelle 3.21 Felder von Benachrichtigungen

Über einen POST-Zugriff können darüber hinaus auch Benachrichtigungen als gelesen markiert werden. Der API-Endpunkt dazu lautet `https://graph.facebook.com/BenachrichtigungID/?unread=0` und erfordert ebenfalls die Berechtigung `manage_notifications`.

Lesen von »Gefällt mir« (»Likes«)

Um auszulesen, bei welchen Seiten – oder allgemeiner ausgedrückt: Objekten im sozialen Graph – ein Benutzer auf Gefällt mir geklickt hat, wird der API-Endpunkt `https://graph.facebook.com/BenutzerID/likes` genutzt. Um die »Likes« des aktuel-

len Benutzers zu lesen, ist dabei die Berechtigung `user_likes` notwendig. Analog dazu ermöglicht die Berechtigung `friends_likes` das Auslesen der »Likes« der Freunde des aktuellen Benutzers.

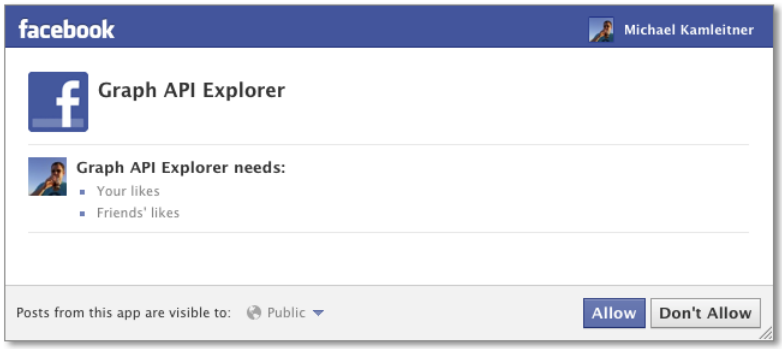


Abbildung 3.12 Autorisierung einer Anwendung mit den Berechtigungen »user_likes« und »friends_likes«

Das Ergebnis dieser API-Methode enthält ein Array mit Objekten, die aus folgenden Feldern bestehen:

Feldname	Beschreibung	Typ
`id`	numerische Objekt-ID der Seite	`string`
`name`	Name der Seite	`string`
`category`	Kategorie der Seite (z. B. »Company«, »Product/Service« etc.)	`string`
`created_time`	Zeitstempel des Zeitpunktes, zu dem auf GEFÄLLT MIR geklickt wurde	Zeitstempel im ISO-8601-Format

Tabelle 3.22 Felder von »Likes«

Um zu prüfen, ob ein Benutzer auf einer bestimmten, gesuchten Seite auf GEFÄLLT MIR geklickt hat, kann der Endpunkt `https://graph.facebook.com/me/likes/ObjektID` verwendet werden. Hat der Benutzer auf dieser Seite auf LIKE geklickt, wird ein Array mit den Feldern `name`, `category`, `created_time` und `id` der Seite zurückgegeben. Ist dies nicht der Fall, ist das Ergebnis leer.

Deautorisieren von Anwendungen und Zugriffsrechten

Benutzer haben unter *http://www.facebook.com/settings/?tab=privacy&section=apps* die Möglichkeit, installierte Anwendungen ganz zu deautorisieren oder verge-

bene Zugriffsrechte anzupassen und wieder zu entziehen. Dieselbe Möglichkeit bietet der API-Endpunkt `https://graph.facebook.com/BenutzerID/permissions`, der mit einem DELETE-Zugriff und folgenden Parametern aufgerufen werden kann:

Parameter	Beschreibung	Typ	Pflichtfeld
`permission`	Die Berechtigung, die der Anwendung entzogen werden soll (z. B. `publish_stream`). Bleibt der Parameter leer, wird die ganz Anwendung deautorisiert.	`string`	nein

Tabelle 3.23 Felder zum Löschen von Zugriffsrechten

War die Deautorisierung oder der Entzug des Zugriffsrechtes erfolgreich, enthält das Ergebnis des Aufrufs `true`.

3.2.2 Das Seitenobjekt

Jede auf Facebook erstellte Unternehmens-, Produkt- oder Organisationsseite wird im sozialen Graphen durch ein Seitenobjekt vom Typ `page` repräsentiert. Orte (*Places*), die früher als eigenständiger Objekttyp behandelt wurden, werden mittlerweile ebenfalls als Seitenobjekte dargestellt, die zusätzliche Informationen über die geographische Position des Subjekts der Seite enthalten. Zusätzlich zu den Seiten, die auf Facebook selbst existieren, besteht mit dem Open Graph (siehe Kapitel 6, »Social Plugins«) auch die Möglichkeit, jede externe Webseite als Seitenobjekt im sozialen Graphen abzubilden.

Felder des Seitenobjekts

Die Felder von Seitenobjekten gelten grundsätzlich als öffentlich und sind daher auch ohne Access Token zu beziehen. Von dieser Regel gibt es zwei Ausnahmen:

- Seiten, die vom Ersteller etwa in der Testphase noch nicht veröffentlicht wurden (»*unpublish*«), sind lediglich mit dem Access Token des Erstellers oder eines anderen Seitenadministrators über die Graph API abrufbar.
- Seiten, deren Verfügbarkeit vom Seitenersteller geographisch oder hinsichtlich des Mindestalters eingeschränkt wurden, sind lediglich mit dem Access Token eines Benutzers abrufbar, der diesen Einschränkungen genügt.

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Seite	kein Access Token notwendig	`string`
`name`	Name der Seite	kein Access Token notwendig	`string`
`link`	URL zum Facebook-Seitenprofil	kein Access Token notwendig	`url`
`category`	Seitentyp	kein Access Token notwendig	`string`, z. B. »Company«, »Product«...
`checkins`	Anzahl der Benutzer, die an dem Ort der Seite eingecheckt haben	kein Access Token notwendig	`number`
`description`	Beschreibung der Seite	kein Access Token notwendig	`string`
`likes`	Anzahl der Benutzer, die auf GEFÄLLT MIR geklickt haben	kein Access Token notwendig	`number`
`location`	Adresse der Seite, wenn vorhanden	kein Access Token notwendig	Array aus Objekten mit den Feldern `street`, `city`, `country`, `zip`, `latitude` und `longitude`
`phone`	Telefonnummer der Seite, wenn vorhanden	kein Access Token notwendig	`string`
`picture`	Profilbild der Seite	kein Access Token notwendig	`url`
`type`	Typ des Seitenobjekts	kein Access Token notwendig	`string`, z. B. »Page«, »Application«...

Tabelle 3.24 Felder des Seitenobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
username	frei gewählter, eindeutiger Name der Seite	kein Access Token notwendig	string
website	externe Webseite	kein Access Token notwendig	url
access_token	ein Seiten-Access-Token	Access Token eines Seitenadministrators mit der Berechtigung manage_pages	string

Tabelle 3.24 Felder des Seitenobjekts (Forts.)

Aufgrund der vielen unterschiedlichen Seitentypen können sich auch die jeweils verfügbaren Felder und Verknüpfungen unterscheiden. Die Aufstellung zeigt daher jene Felder, die bei den meisten Seitentypen gleich sind. Abhängig vom Seitentyp im Feld type enthalten Seitenobjekte weitere Datenfelder:

- Seiten vom Typ page enthalten folgende Felder: location, parking, public_transit, hours, talking_about_count
- Seiten vom Typ application enthalten folgende Felder: canvas_name, app_space, icon_url, logo_url, company, daily_active_users, weekly_active_users, monthly_active_users

Verknüpfungen des Seitenobjekts

Objekte vom Typ page können in der Graph API mit den in der folgenden Tabelle aufgeführten Objekttypen verknüpft werden. Das für einige Zugriffe notwendige Seiten-Access-Token kann, wie in Abschnitt 2.7.2, »Access Tokens für Seiten«, beschrieben, von einem Benutzer mit der Berechtigung manage_pages bezogen werden.

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
feed	die Pinnwand der Seite	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ post
picture	Profilbild der Seite	keine	HTTP-302-Redirect auf die Bilddatei

Tabelle 3.25 Verknüpfungen des Seitenobjekts

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`settings`	Einstellungen der Seite	Access Token der Seite	Array aus Objekten mit den Feldern `setting` und `value`
`tagged`	Fotos, Videos und Postings, in denen die Seite markiert wurde	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `post`, `video` oder `photo`
`links`	Links, die von der Seite gepostet wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `link`
`photos`	Fotos, die von der Seite gepostet wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `photo`
`groups`	Gruppen, in denen die Seite Mitglied ist	Benötigt ein beliebiges Access Token.	Array aus Objekten mit den Feldern `id`, `version`, `name` und `unread`
`albums`	Fotoalben, die von der Seite angelegt wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `album`
`statuses`	Status-Updates der Seite (wie `posts` Fotos, Videos etc.)	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `status message`
`videos`	Videos, die von der Seite hochgeladen wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `videos`
`notes`	Notizen, die von der Seite angelegt wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `note`

Tabelle 3.25 Verknüpfungen des Seitenobjekts (Forts.)

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
posts	Postings der Seite (wie feed, aber ohne Postings von Benutzern oder anderen Seiten)	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ post
questions	Fragen, die von der Seite gepostet wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ question
events	Veranstaltungen, die von der Seite angelegt wurden	Benötigt ein beliebiges Access Token.	Array aus Objekten mit den Feldern id, name start_time, end_time, location und rsvp_status
checkins	Checkins auf dieser Seite vom aktuellen Benutzer und seinen Freunden	user_checkins oder friends_checkins	Array aus Objekten vom Typ checkin
admins	Liste der Seitenadministratoren	Access Token der Seite	Array aus Objekten mit den Feldern id und name
blocked	Liste der von der Seite geblockten Benutzer	Access Token der Seite	Array aus Objekten mit den Feldern id und name
tabs	Reiter, die auf der Seite verwendet werden	Access Token der Seite	Array aus Objekten mit den Feldern id, name link, application, custom_name, is_permanent, position und is_not_connection_landing_tab

Tabelle 3.25 Verknüpfungen des Seitenobjekts (Forts.)

Hinweis: Um die im Folgenden beschriebenen verknüpften Objekte von Seitenobjekten veröffentlichen zu können, muss grundsätzlich ein wie in Abschnitt 2.7.2, »Access Tokens für Seiten«, beschriebenes Seiten-Access-Token verwendet werden. Zusätzlich erfordern die meisten Veröffentlichungen die erweiterte Berechtigung publish_stream für das Benutzer-Access-Token.

Wichtig: Wird beim Veröffentlichen von mit Seiten verknüpften Objekten versehentlich ein Benutzer-Access-Token verwendet, wird Facebook in vielen Fällen keine Fehlermeldung ausgeben, sondern das *Objekt am Benutzerprofil des aktuellen Benutzers* veröffentlichen!

Veröffentlichen von Veranstaltungen

Um eine Veranstaltung vom Objekttyp `event` auf einer Seite zu veröffentlichen, müssen Sie einen POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/SeitenID/events` durchführen. Das zur Signatur verwendete Access Token muss vom Benutzer mit der Zugriffsberechtigung `create_event` autorisiert worden sein.

Folgende Parameter spezifizieren die Veranstaltung näher:

Parameter	Beschreibung	Typ	Pflichtfeld
`name`	Name der Veranstaltung	`string`	ja
`start_time`	UNIX-Zeitstempel des Beginns der Veranstaltung	UNIX-Zeitstempel	ja
`end_time`	UNIX-Zeitstempel des Endes der Veranstaltung	UNIX-Zeitstempel	nein
`description`	Beschreibung der Veranstaltung	`string`	nein
`location`	Ort der Veranstaltung	`string`	nein
`privacy_type`	Privatsphäre-Einstellung – `OPEN`, `CLOSED` oder `SECRET`	`string`	nein

Tabelle 3.26 Felder zum Veröffentlichen von Veranstaltungen

Die Privatsphäre-Einstellungen im Parameter `privacy_type` erlauben folgende Einstellungen:

- `OPEN` – Die Veranstaltung ist offen, jeder Benutzer kann teilnehmen.
- `CLOSED` – Die Veranstaltung ist öffentlich sichtbar, Benutzer können aber nur mit Einladung teilnehmen.
- `SECRET` – Die Veranstaltung ist versteckt, Benutzer können nur mit Einladung teilnehmen.

War das Erzeugen der Veranstaltung erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID der Veranstaltung.

Veranstaltungen, die auf einer Seite veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/EventID` wieder gelöscht werden.

Veröffentlichen von Links

Anwendungen können im Namen einer Seite Objekte vom Typ `link` auf der Pinnwand der Seite veröffentlichen. Wie auch zum Veröffentlichen von Wall-Posts ist dazu ein Seiten-Access-Token mit der erweiterten Berechtigung `publish_stream` notwendig.

Zum Veröffentlichen muss ein POST-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID/links` mit folgenden Parametern gesendet werden:

Parameter	Beschreibung	Typ	Pflichtfeld
`link`	URL des zu veröffentlichenden Links	`string`	ja
`message`	Beschreibungstext des zu veröffentlichenden Links	`string`	nein

Tabelle 3.27 Felder zum Veröffentlichen von »Likes«

War das Erzeugen des Links erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Links.

Links, die auf einer Seite veröffentlicht wurden, können Sie mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID_LinkID` wieder löschen. Die ID wird dabei aus *Seiten- und Link-ID zusammengesetzt*!

Veröffentlichen von Notizen

Ähnlich wie das Veröffentlichen von Links erfordert auch das Veröffentlichen von Objekten des Typs `notes` auf einer Seite ein Seiten-Access-Token mit der erweiterten Zugriffsberechtigung `publish_stream`. Der API-Endpunkt `https://graph.facebook.com/SeitenID/notes` nimmt folgende Parameter bei POST-Zugriffen entgegen:

Parameter	Beschreibung	Typ	Pflichtfeld
subject	Betreff der zu veröffentlichenden Notiz	string	ja
message	Beschreibungstext des zu veröffentlichenden Links	string	ja

Tabelle 3.28 Felder zum Veröffentlichen von Links

Hinweis: Der message-Parameter darf dabei einfache HTML-Auszeichnungen wie <b>, <i>, <u>, <blockquote>, <ul> und <li> enthalten.

War das Erzeugen der Notiz erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID der Notiz.

Notizen, die auf einer Seite veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt https://graph.facebook.com/NotizenID wieder gelöscht werden.

Veröffentlichen von Fotos

Fotos können auf der Pinnwand einer Seite mit einem POST-Zugriff auf den API-Endpunkt https://graph.facebook.com/SeitenID/photos veröffentlicht werden. Dazu ist ein Seiten-Access-Token, das mit der erweiterten Zugriffsberechtigung publish_stream signiert ist, notwendig. Folgende Parameter können Sie dabei übergeben:

Parameter	Beschreibung	Typ	Pflichtfeld
source	das zu veröffentlichende Foto als binäre Zeichenkette	multipart/form-data	ja
message	Beschreibungstext des zu veröffentlichenden Fotos	string	nein

Tabelle 3.29 Felder zum Veröffentlichen von Fotos

Hinweis: Das Codieren des Fotos als binäre Zeichenkette wird dabei üblicherweise von curl oder dem PHP SDK übernommen.

War das Veröffentlichen des Fotos erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID des Fotos.

Veröffentlichen von Videos

Videos können auf der Pinnwand einer Seite mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/SeitenID/videos` veröffentlicht werden. Dazu ist ein Seiten-Access-Token, das mit der erweiterten Zugriffsberechtigung `publish_stream` signiert ist, notwendig. Folgende Parameter werden dabei übergeben:

Parameter	Beschreibung	Typ	Pflichtfeld
`source`	das zu veröffentlichende Video als binäre Zeichenkette	`multipart/form-data`	ja
`title`	Titel des zu veröffentlichenden Videos	`string`	nein
`description`	Beschreibungstext des zu veröffentlichenden Videos	`string`	nein

Tabelle 3.30 Felder zum Veröffentlichen von Videos

Hinweis: Das Codieren des Videos als binäre Zeichenkette wird dabei üblicherweise von `curl` oder dem PHP SDK übernommen.

War das Veröffentlichen des Videos erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Videos.

Veröffentlichen von Wall-Postings

Wall-Postings können auf der Pinnwand einer Seite mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/SeitenID/feed` erzeugt werden. Dazu ist ein Seiten-Access-Token notwendig, das vom Benutzer mit der erweiterten Berechtigung `publish_stream` autorisiert wurde. Die API-Methode unterstützt folgende Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	die Benutzernachricht des Wall-Postings	`string`	ja
`link`	der Link des Wall-Postings	`string`	nein
`picture`	URL zu einem Thumbnail-Bild	`string`	nein

Tabelle 3.31 Felder zum Veröffentlichen von Wall-Postings

Parameter	Beschreibung	Typ	Pflichtfeld
`name`	der Ankertext, der für `link` angezeigt wird	`string`	nein
`caption`	die Überschrift des Wall-Postings	`string`	nein
`description`	der Beschreibungstext des Wall-Postings	`string`	nein
`actions`	Action-Links des Wall-Postings	Array aus Objekten mit den Feldern `name` und `link`	nein
`privacy`	Privatsphäre-Einstellungen des Wall-Postings	`string`	nein

Tabelle 3.31 Felder zum Veröffentlichen von Wall-Postings (Forts.)

Die einzelnen Parameter des Wall-Postings funktionieren analog, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben.

War das Veröffentlichen des Postings erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Postings.

Postings, die auf einer Seite veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID_PostingID` wieder gelöscht werden. Die ID wird dabei aus *Seiten- und Posting-ID zusammengesetzt*!

Veröffentlichen von Status-Updates

Status-Updates werden als Sonderform des Wall-Postings analog zum Benutzerobjekt behandelt. Zum Veröffentlichen von Status-Updates ist daher ebenfalls ein Seiten-Access-Token mit der Berechtigung `publish_stream` notwendig. Das Veröffentlichen des Status-Updates erfolgt mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/feed` und akzeptiert lediglich einen Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	die Nachricht des Status-Updates	`string`	ja

Tabelle 3.32 Felder zum Veröffentlichen von Status-Updates

War das Veröffentlichen des Status-Updates erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Status-Updates.

Status-Updates, die auf einer Seite veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID_PostingID` wieder gelöscht werden. Die ID wird dabei aus *Seiten- und Posting-ID zusammengesetzt*!

Ändern von Seiteneinstellungen

Facebook bietet Anwendungsentwicklern die Möglichkeit, einige – leider nur wenige – Seiteneinstellungen durch API-Zugriffe zu ändern. Dies erfordert ein beliebiges Seiten-Access-Token und wird durch einen POST-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID/Settings` mit folgenden Parametern durchgeführt:

Parameter	Beschreibung	Typ	Pflichtfeld
`setting`	Name der Einstellung, die geändert werden soll	`string`	ja
`value`	`true` oder `false`	`boolean`	ja

Tabelle 3.33 Felder zum Ändern von Seiteneinstellungen

Folgende Einstellungen können im Parameter `setting` gewählt werden:

- `USERS_CAN_POST` – Gibt an, ob Benutzer grundsätzlich auf der Pinnwand der Seite posten dürfen.
- `USERS_CAN_POST_PHOTOS` – Gibt an, ob Benutzer Fotos auf der Pinnwand der Seite posten dürfen.
- `USERS_CAN_TAG_PHOTOS` – Gibt an, ob Benutzer die Seite in Fotos markieren dürfen.
- `USERS_CAN_POST_VIDEOS` – Gibt an, ob Benutzer Videos auf der Pinnwand der Seite posten dürfen.

Die entsprechenden Einstellungen können zur Kontrolle in Facebook unter der URL *https://www.facebook.com/pages/edit/?id=SeitenID* abgerufen werden.

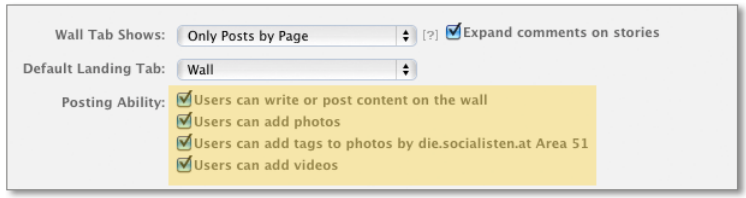


Abbildung 3.13 Der Dialog zur Seiteneinstellung, der mit den im Objekt »settings« änderbaren Einstellungen korrespondiert

Ändern von Reitern/Tabs

Wie in Abschnitt 1.5.1, »Hello Facebook! als Tab-Anwendung«, beschrieben, können Facebook-Canvas-Anwendungen in Reiter/Tabs von Seiten integriert werden. Die Graph API bietet praktischerweise Methoden, um die installierten Reiter einer Seite auszulesen, Reiter hinzuzufügen, zu ändern oder zu löschen. Generell ist dazu lediglich das Seiten-Access-Token notwendig.

Ein GET-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID/tabs` liefert alle aktuell installierten Reiter als JSON-Array mit folgenden Feldern zurück:

Name	Beschreibung	Typ
`id`	Identifikation des Reiters, zusammengesetzt aus Seiten- und Anwendungs-ID	`string`
`name`	der Default-Name des Reiters	`string`
`link`	Direkt-URL zum Reiter	`string`
`application`	Anwendung, die den Reiter bereitstellt	Array mit den Feldern `id`, `name`, `canvas_name` und `name_space`
`custom_name`	der vom Seitenadministrator optional festgelegte, benutzerdefinierte Name des Reiters	`string`
`is_permanent`	Zeigt an, ob der Reiter permanent installiert ist. `true` ausschließlich für die Standard-Tabs von Facebook (Wall, Info, Insights).	`boolean`
`position`	Reihenfolge der Reiter	`integer`
`is_non_connection_landingtab`	Zeigt an, ob der Reiter als »Landing-Tab« für Benutzer, die noch nicht auf Gefällt mir geklickt haben, verwendet wird.	`boolean`

Tabelle 3.34 Lesen von Tabs/Reitern einer Seite

Das folgende Beispiel zeigt einen Ausschnitt des Ergebnisses für eine typische Facebook-Seite. Neben dem Standard-Tab »Wall« (erkennbar an `is_permanent=true`) findet sich ein von der Anwendung »Gewinnspiel« bereitgestellter Reiter, der darüber hinaus als Landing-Tab konfiguriert ist (`is_non_connection_landingtab=true`):

```
{
  "data": [
    {
      "id": "116784881738026/tabs/wall",
      "name": "Wall",
      "link": "http://www.facebook.com/pages/diesocialistenat-Area-51/
       116784881738026?sk=wall",
      "is_permanent": true,
      "position": 0,
      "is_non_connection_landing_tab": true,
      "type": "profile_tab"
    },
    ...
    {
      "id": "116784881738026/tabs/app_111600682255979",
      "name": "Gewinnspiel",
      "link": "http://www.facebook.com/pages/diesocialistenat-Area-51/
       116784881738026?sk=app_111600682255979",
      "application": {
        "name": " Gewinnspiel",
        "canvas_name": "gewinnen-mit-dev-a",
        "namespace": "gewinnen-mit-dev-a",
        "id": "111600682255979"
      },
      "custom_name": "Jetzt gewinnen!",
      "position": 3,
      "type": "profile_tab"
      "is_non_connection_landingtab": true
    },
    ...
```


Listing 3.8 Rückgabewert beim Auslesen von Tabs/Reitern einer Seite

Landing Tabs

Bis März 2012 konnten Betreiber einer Facebook-Seite in den Seiteneinstellungen festlegen, welches Tab einem Besucher, der noch nicht Fan der Seite war, standardmäßig angezeigt werden sollte. Diese als *Default Landing Tab* bezeichnete Funktion steht allerdings seit der Umstellung auf die Timeline nicht mehr zur Verfügung. Obwohl nicht auszuschließen ist, dass Facebook diese beliebte Funktion später wieder einführen wird, landen Besucher der Seite also vorerst immer auf der Timeline der Seite.

Neben der Möglichkeit, alle Reiter einer Seite abzufragen, können Sie mit dem Aufruf `https://graph.facebook.com/SeitenID/tabs/TabName` auch einen Reiter gezielt abrufen. Als `TabName` können Sie dabei den Bezeichner eines Facebook-Standard-Tabs (`wall`, `events`, `friendactivity`, `info`, `page_insights`, `notes`, `photos` , oder `reviews`) oder die numerische ID der bereitstellenden Anwendung übergeben. Damit können Anwendungen leicht überprüfen, ob ihr Reiter auf einer bestimmten Seite bereits installiert ist: Ist dies nicht der Fall, ist das Ergebnis des Aufrufs ein leeres Array. Verwenden Sie hierbei anstelle des Seiten- ein Anwendungs-Access-Token, können Sie auch die Installation des eigenen Anwendungsreiters auf jeder beliebigen Seite prüfen – ein Benutzer- oder Seiten-Access-Token ist dazu nicht notwendig.

Verfügen Sie über ein Seiten-Access-Token, können Sie mittels Graph API auch eine beliebige Anwendung auf die Seiten installieren, für die der autorisierende Benutzer Administrationsrechte besitzt. Dazu müssen Sie einen POST-Zugriff auf `https://graph.facebook.com/SeitenID/tabs` mit folgendem Parameter senden:

Parameter	Beschreibung	Typ	Pflichtfeld
`app_id`	ID der Anwendung, deren Reiter installiert werden soll	`string`	ja

Tabelle 3.35 Felder zum Installieren eines Tabs/Reiters

War das Installieren des Reiters erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Mithilfe des Seiten-Access-Tokens ist es auch möglich, manche Einstellungen bereits installierter Reiter zu manipulieren. Dazu ist ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/SeitenID/tabs/TabID` mit folgenden Parametern notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`position`	Reihenfolge, in der die Reiter angezeigt werden. Facebook-Standard-Tabs werden immer vorgereiht, die Sortierung der Anwendungs-Tabs beginnt bei 0.	`integer`	nein
`custom_name`	der vom Seitenadministrator optional festgelegte, benutzerdefinierte Name des Reiters	`string`	nein

Tabelle 3.36 Felder zum Ändern von Tabs/Reitern einer Seite

Parameter	Beschreibung	Typ	Pflichtfeld
is_non_connection_landing_tab	Definiert den Reiter als »Landing-Tab« für Benutzer, die noch nicht auf Gefällt mir geklickt haben.	boolean	nein

Tabelle 3.36 Felder zum Ändern von Tabs/Reitern einer Seite (Forts.)

War das Ändern der Reiter-Einstellungen erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Abschließend kann mit einem Seiten-Access-Token auch jeder installierte Reiter wieder von einer Seite entfernt werden. Dazu muss ein DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID/tabs/TabID` gesendet werden.

War das Löschen des Reiters erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Verwalten blockierter Benutzer

Facebook erlaubt es Seitenadministratoren, im Missbrauchsfall einzelne Benutzer vom Zugriff auf ihre Seite zu blockieren. Blockierte Benutzer können die Inhalte einer Seite zwar lesen, jedoch nicht mehr an der Konversation auf der Pinnwand teilnehmen. Neben dem Facebook User Interface kann dies auch über die Graph API bewerkstelligt werden. Auch für diese Zugriffe ist immer ein Seiten-Access-Token für die verwalteten Seiten notwendig.

Um die Liste der aktuell blockierten Benutzer auszulesen, kann ein GET-Zugriff auf den Endpunkt `https://graph.facebook.com/SeitenID/blocked` verwendet werden. Das Ergebnis ist ein Array aller blockierten Benutzer, dessen Objekte folgende Felder enthalten:

Name	Beschreibung	Typ
id	numerische ID des blockierten Benutzers	string
name	Name des blockierten Benutzers	string

Tabelle 3.37 Felder beim Lesen blockierter Benutzer

Mit einer Anfrage an `https://graph.facebook.com/SeitenID/blocked/BenutzerID` können Sie dies auch für einen bestimmten Benutzer prüfen. Ist der Benutzer blockiert, enthält das Ergebnis ein Objekt mit den Feldern `id` und `name` des Benutzers.

Zum Blockieren eines Benutzers dient der POST-Zugriff auf `https://graph.facebook.com/SeitenID/blocked`, wobei der zu blockierende Benutzer als Parameter übergeben werden muss:

Parameter	Beschreibung	Typ	Pflichtfeld
uid	numerische ID des Benutzers, der blockiert werden soll	string	ja

Tabelle 3.38 Felder zum Blockieren von Benutzern

War das Blockieren des Benutzers erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Um die Blockierung eines Benutzers wieder aufzuheben, genügt ein DELETE-Zugriff auf den API-Endpunkt `https://graph.facebook.com/SeitenID/blocked/BenutzerID`.

War das Aufheben der Blockierung erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Prüfen von Administratoren

Mit einem gültigen Seiten-Access-Token kann geprüft werden, ob ein bestimmter Benutzer Administrator einer Seite ist. Der GET-Zugriff lautet `https://graph.facebook.com/SeitenID/admins/BenutzerID`. Ist der Benutzer nicht Administrator, ist das Ergebnis ein leeres Array. Anderenfalls wird ein Objekt mit folgenden Feldern zurückgeliefert:

Name	Beschreibung	Typ
id	numerische ID des geprüften Benutzers	string
name	Name des geprüften Benutzers	string

Tabelle 3.39 Lesen von Administratoren einer Seite

3.2.3 Das Gruppenobjekt

Facebook-Gruppen werden im sozialen Graphen in Objekten vom Typ `group` gespeichert. Gruppen können in Verbindung von Seiten und Benutzern auftreten und daher unter folgenden API-Endpunkten mittels GET abgefragt werden:

```
https://graph.facebook.com/BenutzerID/groups
https://graph.facebook.com/SeitenID/groups
https://graph.facebook.com/GruppenID
```

Je nach Privatsphäre-Einstellung der Gruppe ist dabei ein Access Token notwendig:

- Öffentliche Gruppen mit der Privatsphäre-Einstellung `OPEN` können mit einem beliebigen gültigen Access Token eingelesen werden.

- Die nicht-öffentlichen Gruppen (Privatsphäre-Einstellung `SECRET` oder `CLOSED`) eines Benutzers können mit der erweiterten Berechtigung `user_groups` eingelesen werden.
- Die nicht-öffentlichen Gruppen der Freunde eines Benutzers können mit der erweiterten Berechtigung `friends_groups` eingelesen werden.

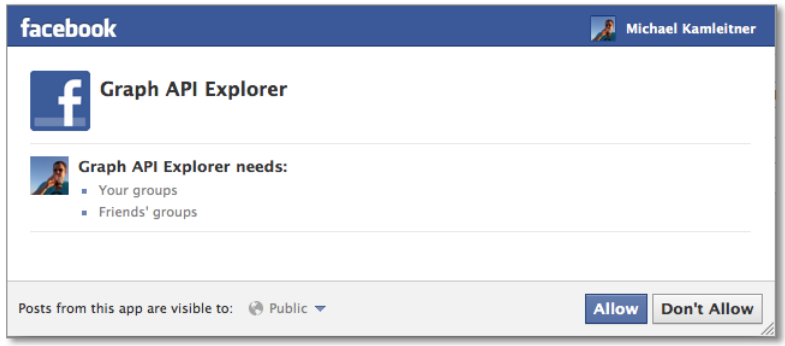


Abbildung 3.14 Autorisierung einer Anwendung mit den Berechtigungen »user_groups« und »friends_groups«

Felder des Gruppenobjekts

Objekte vom Typ `group` weisen folgende Felder auf:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Gruppe	beliebiges Access Token oder `user_groups` oder `friends_groups`	`string`
`version`	Kennzeichen, ob die Gruppe vor Oktober 2010 (Wert `0`) oder danach (Wert `1`) erstellt wurde	beliebiges Access Token oder `user_groups` oder `friends_groups`	`integer`
`icon`	URL zum Icon der Gruppe	beliebiges Access Token oder `user_groups` oder `friends_groups`	`string`
`owner`	Ersteller der Gruppe	beliebiges Access Token oder `user_groups` oder `friends_groups`	Objekt mit den Feldern `id` und `name`

Tabelle 3.40 Felder des Gruppenobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
name	Name der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	string
description	Beschreibung der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	string
link	URL zur Website der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	string
privacy	Privatsphäre-Einstellung der Gruppe (OPEN, CLOSED oder SECRET)	beliebiges Access Token oder user_groups oder friends_groups	string
updated_time	Zeitstempel der letzten Modifikation der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	string mit einem ISO-8601-Zeitstempel

Tabelle 3.40 Felder des Gruppenobjekts (Forts.)

Verknüpfungen des Gruppenobjekts

Objekte vom Typ group können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
feed	die Pinnwand der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	Array aus Objekten vom Typ post
members	Mitglieder der Gruppe (liefert nur die ersten 500 Mitglieder)	beliebiges Access Token oder user_groups oder friends_groups	Array aus Objekten mit den Feldern id, name und administrator
picture	Profilbild der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	HTTP-302-Redirect auf die Bilddatei

Tabelle 3.41 Verknüpfungen des Gruppenobjekts

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
docs	Dokumente der Gruppe	beliebiges Access Token oder user_groups oder friends_groups	Array aus Objekten mit den Feldern id, from, subject, message, icon, updated_time, revision, can_edit und can_delete

Tabelle 3.41 Verknüpfungen des Gruppenobjekts (Forts.)

Veröffentlichen von Links

Anwendungen können Objekte vom Typ link auf der Pinnwand der Gruppe veröffentlichen. Wie auch zum Veröffentlichen von Wall-Posts ist dazu ein Access Token mit der erweiterten Berechtigung publish_stream notwendig.

Zum Veröffentlichen muss ein POST-Zugriff auf den Endpunkt https://graph.facebook.com/GruppenID/feed mit folgenden Parametern gesendet werden:

Parameter	Beschreibung	Typ	Pflichtfeld
link	URL des zu veröffentlichenden Links	string	ja
message	Beschreibungstext des zu veröffentlichenden Links	string	nein

Tabelle 3.42 Felder zum Veröffentlichen von Links

War das Erzeugen des Links erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID des Links.

Links, die in einer Gruppe veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt https://graph.facebook.com/GruppenID_LinkID wieder gelöscht werden. Die ID wird dabei aus *Gruppen- und Link-ID zusammengesetzt*!

Veröffentlichen von Wall-Postings

Wall-Postings können auf der Pinnwand einer Gruppe mit einem POST-Zugriff auf den API-Endpunkt https://graph.facebook.com/GruppenID/feed erzeugt werden. Dazu ist ein Access Token notwendig, das vom Benutzer mit der erweiterten Berechtigung publish_stream autorisiert wurde. Die API-Methode unterstützt folgende Parameter:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	die Benutzernachricht des Wall-Postings	`string`	ja
`link`	der Link des Wall-Postings	`string`	nein
`picture`	URL zu einem Thumbnail-Bild	`string`	nein
`name`	der Ankertext, der für `link` angezeigt wird	`string`	nein
`caption`	die Überschrift des Wall-Postings	`string`	nein
`description`	der Beschreibungstext des Wall-Postings	`string`	nein
`actions`	Action-Links des Wall-Postings	Array aus Objekten mit den Feldern `name` und `link`	nein
`privacy`	Privatsphäre-Einstellungen des Wall-Postings	`string`	nein

Tabelle 3.43 Felder zum Veröffentlichen von Wall-Postings

Die einzelnen Parameter des Wall-Postings funktionieren analog, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben.

War das Veröffentlichen des Postings erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Postings.

Postings, die in einer Gruppe veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/GruppenID_PostingID` wieder gelöscht werden. Die ID wird dabei aus *Gruppen- und Posting-ID zusammengesetzt*!

Veröffentlichen von Status-Updates

Status-Updates werden als Sonderform des Wall-Postings analog zum Benutzer behandelt. Zum Veröffentlichen von Status-Updates ist daher ebenfalls ein Access Token mit der Berechtigung `publish_stream` notwendig. Das Veröffentlichen des Status-Updates erfolgt mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/GruppenID/feed`, der lediglich einen Parameter akzeptiert:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	die Nachricht des Status-Updates	`string`	ja

Tabelle 3.44 Felder zum Veröffentlichen von Status-Updates

War das Veröffentlichen des Status-Updates erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Status-Updates.

Status-Updates, die in einer Gruppe veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/GruppenID_PostingID` wieder gelöscht werden. Die ID wird dabei aus *Gruppen- und Posting-ID zusammengesetzt*!

3.2.4 Das Albumobjekt

Fotos werden auf Facebook-Benutzer- und -Seitenprofilen in Fotoalben organisiert und im sozialen Graphen als Objekt vom Typ `album` repräsentiert.

Felder des Albumobjekts

Fotoalben, die öffentlich sind, können mit jedem beliebigen Access Token abgefragt werden. Um auf nicht-öffentliche Alben zuzugreifen, benötigen Sie die Berechtigung `user_photos`, wenn das Album dem aktuellen Benutzer gehört, oder `friends_photos`, wenn das Album einem Freund des aktuellen Benutzers gehört.

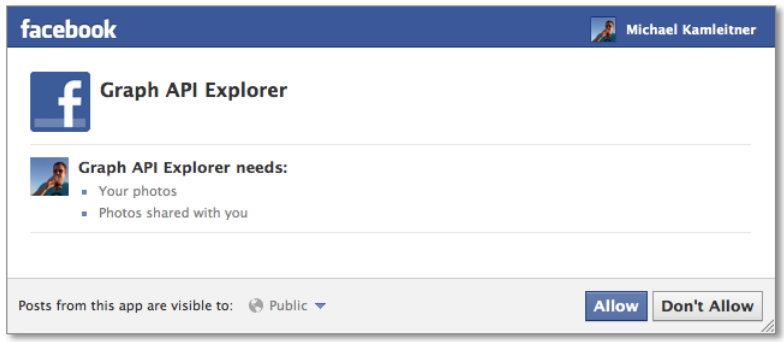


Abbildung 3.15 Autorisierung einer Anwendung mit den Berechtigungen »user_photos« und »friends_photos«

Alben können mit Benutzer-, Seiten- und Anwendungsobjekten verknüpft sein. Dementsprechend können Sie die Alben dieser Objekte mit folgenden API-Endpunkten mittels GET-Zugriff abfragen:

```
https://graph.facebook.com/BenutzerID/albums
https://graph.facebook.com/SeitenID/albums
https://graph.facebook.com/AnwendungsID/albums
```


Das Ergebnis ist jeweils ein Array von JSON-Objekten mit folgenden Feldern:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Seite	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	`string`
`from`	Objekt-ID des Erstellers (Benutzer, Applikation oder Seite)	Benötigt ein beliebiges Access Token.	Array mit den Feldern `id` und `name`
`name`	Titel des Albums	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	`string`
`description`	Beschreibung des Albums	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	`string`
`location`	Ort des Albums	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	`string`
`link`	Link zum Album auf Facebook	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	`url`
`cover_photo`	ID des Cover-Fotos	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	`string`

Tabelle 3.45 Felder des Albumobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
privacy	Privatsphäre-Einstellungen des Albums	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	string
count	Anzahl der Fotos in diesem Album	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	number
type	Typ des Albums	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	string, z. B. profile, mobile, wall, normal oder album
created_time	Zeitstempel der Erstellung des Albums	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	string mit einem ISO-8601-Zeitstempel
updated_time	Zeitstempel der letzten Modifikation des Albums	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	string mit einem ISO-8601-Zeitstempel

Tabelle 3.45 Felder des Albumobjekts (Forts.)

Verknüpfungen des Albumobjekts

Objekte vom Typ album können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
photos	Fotos, die zum Album gehören	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	Array aus Objekten vom Typ photo
likes	Benutzer, die bei diesem Album auf Gefällt mir geklickt haben	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	Array mit den Feldern id und name

Tabelle 3.46 Verknüpfungen des Albumobjekts

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
comments	Kommentare, die in diesem Album hinterlassen wurden	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	Array mit den Feldern id, from, message, created_time und name
picture	das Cover-Foto des Albums	Benötigt ein beliebiges Access Token oder user_photos oder friends_photos.	HTTP-302-Redirect auf die Bilddatei

Tabelle 3.46 Verknüpfungen des Albumobjekts (Forts.)

Veröffentlichen von Fotos

Fotos können – ähnlich wie auf die Pinnwand eines Benutzers oder einer Seite – direkt in ein bestehendes Album hochgeladen werden. Der dazu notwendige POST-Zugriff auf den Endpunkt `https://graph.facebook.com/AlbumID/photos` muss mit einem Access Token signiert werden, das die erweiterte Berechtigung `publish_stream` besitzt. Die folgenden Parameter werden dabei übergeben:

Parameter	Beschreibung	Typ	Pflichtfeld
source	das zu veröffentlichende Foto als binäre Zeichenkette	multipart/form-data	ja
message	Beschreibungstext des zu veröffentlichenden Fotos	string	nein

Tabelle 3.47 Felder zum Veröffentlichen von Fotos

War das Veröffentlichen des Fotos erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Fotos.

Veröffentlichen von Kommentaren

Kommentare zu Fotoalben können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/AlbumID/comments` veröffentlicht werden. Dazu sind ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
message	der Kommentartext	string	ja

Tabelle 3.48 Felder zum Veröffentlichen von Kommentaren

War das Veröffentlichen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID des Kommentars.

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um ein Fotoalbum mit Gefällt mir im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt https://graph.facebook.com/AlbumID/likes. Das verwendete Access Token muss die erweiterte Berechtigung publish_stream beinhalten.

Um das Gefällt mir des Albums zu löschen, muss ein DELETE-Zugriff an den Endpunkt https://graph.facebook.com/AlbumID/likes gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung publish_stream beinhalten.

War das Veröffentlichen oder Löschen des Gefällt mir erfolgreich, enthält das Ergebnis des Aufrufs true.

3.2.5 Das Anwendungsobjekt

Facebook-Anwendungen werden im sozialen Graphen durch Objekte vom Typ application repräsentiert. Anwendungen können mithilfe der Graph API lediglich ausgelesen, nicht aber verändert oder erstellt werden. Der Zugriff auf eine bestimmte Anwendung erfolgt über folgenden API-Endpunkt:

```
https://graph.facebook.com/AnwendungsID
```

Da die meisten Felder des Anwendungsobjekts öffentlich sichtbare Informationen enthalten, ist zum Zugriff kein Access Token notwendig. Für einige Ausnahmen sowie die meisten der verknüpften Objekttypen ist allerdings ein Anwendungs-Access-Token wie in Abschnitt 2.7.1, »Access Tokens für Anwendungen«, notwendig.

Felder des Anwendungsobjekts

Anwendungsobjekte besitzen folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
id	numerische Objekt-ID der Anwendung	keine	string
name	Name der Anwendung	keine	string
description	Beschreibung der Anwendung	keine	string
canvas_name	Canvas URL der Anwendung	keine	string
category	Kategorie der Anwendung	keine	string
subcategory	Unterkategorie der Anwendung	keine	string
company	Hersteller der Anwendung	keine	string
icon_url	URL zum Icon der Anwendung	keine	string
logo_url	URL zum Logo der Anwendung	keine	string
link	URL zur Anwendung	keine	string
daily_active_users	aktive Benutzer während des letzten Tages	keine	integer
weekly_active_users	aktive Benutzer während der letzten Woche	keine	integer
monthly_active_users	aktive Benutzer während des letzten Monats	keine	string
migrations	Migrationseinstellungen der Anwendung	Anwendungs-Access-Token	string
namespace	gleichbedeutend mit canvas_name	–	string
restrictions	demographische Einschränkungen der Anwendung	keine	JSON-Objekt mit den Feldern type, location, age und age_distr

Tabelle 3.49 Felder des Anwendungsobjekts

Das Ende der Anwendungsprofilseiten

Bis vor Kurzem wurde jede angelegte Facebook-Anwendung automatisch auf einer eigenständigen Anwendungsprofilseite dargestellt. Diese Anwendungsprofile funktionieren weitestgehend wie herkömmliche Seiten – sie besitzen eine Pinnwand, erlauben das Veröffentlichen von Fotoalben und erlauben es dem Anwendungshersteller, eine Community für die Benutzer der App zu führen. Da Anwendungen oftmals ohnehin in bereits bestehende Facebook-Seiten integriert werden, war das automatisch erstellte Anwendungsprofil für viele Entwickler ärgerlich: So landen etwa viele Benutzer versehentlich am Anwendungsprofil statt auf der eigentlichen Markenpräsenz eines Unternehmens. Darüber hinaus wurden Anwendungsprofilseiten auch von Facebook stiefmütterlich behandelt – die Profile hinkten den herkömmlichen Facebook-Seiten im Funktionsumfang immer etwas hinterher.

Wie im Dezember 2011 angekündigt wurde, wird Facebook die Unterstützung von Anwendungsprofilseiten mit dem 1. März 2012 einstellen. Entwickler werden die Möglichkeit haben, vorhandene Anwendungsprofile in herkömmliche Seiten umzuwandeln, neu angelegte Anwendungen werden von vornherein ohne Profilseite erstellt. Neben der Migrationsproblematik für bestehende Anwendungsprofile werden auch einige der Verknüpfungen im sozialen Graphen verschwinden – im folgenden Abschnitt werden die betroffenen Verknüpfungen daher nur kurz aufgelistet und nicht im Detail beschrieben.

Verknüpfungen des Anwendungsobjekts

Objekte vom Typ `application` können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`accounts`	Testkonten, die für die Anwendung angelegt wurden	Anwendungs-Access-Token	Array aus Objekten mit den Feldern `id`, `access_token` und `login_url`
`albums`	Fotoalben der Anwendung – *eingestellt ab 1. März 2012*	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `albums`
`banned`	Benutzer, die von der Anwendung blockiert wurden	Anwendungs-Access-Token	Array aus Objekten mit den Feldern `id` und `name`

Tabelle 3.50 Verknüpfungen des Anwendungsobjekts

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`feed`	Pinnwand der Anwendung – *eingestellt ab 1. März 2012*	Benötigt ein beliebiges Access Token bzw. `read_stream` für nicht-öffentliche Postings.	Array aus Objekten vom Typ `post`
`insights`	Nutzungsstatistiken der Anwendung	Anwendungs-Access-Token oder `read_stream`	Array aus Objekten vom Typ `insight`
`links`	veröffentlichte Links der Anwendung – *eingestellt ab 1. März 2012*	Benötigt ein beliebiges Access Token bzw. `read_stream` für nicht-öffentliche Postings.	Array aus Objekten vom Typ `link`
`payments`	Facebook-Credits-Bestellungen, die innerhalb der Anwendung getätigt wurden	Anwendungs-Access-Token	Array aus Objekten vom Typ `order`
`picture`	das Profilbild der Anwendung	keine	HTTP-302-Redirect auf die Bilddatei
`posts`	veröffentlichte Wall-Postings der Anwendung – *eingestellt ab 1. März 2012*	Benötigt ein beliebiges Access Token bzw. `read_stream` für nicht-öffentliche Postings.	Array aus Objekten vom Typ `post`
`reviews`	Reviews der Anwendung – *eingestellt ab 1. März 2012*	keine	Array aus Objekten vom Typ `review`
`roles`	Entwicklerrollen der Anwendung	Anwendungs-Access-Token	Array aus Objekten mit den Feldern `app_id`, `user` und `role`

Tabelle 3.50 Verknüpfungen des Anwendungsobjekts (Forts.)

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`static-resources`	statische Ressourcen, die von der Anwendung per JavaScript SDK registriert wurden	Anwendungs-Access-Token	Array aus Objekten mit den Feldern `useage_stats` und `flushed_resources`
`statuses`	veröffentlichte Status-Updates der Anwendung – *eingestellt ab 1. März 2012*	Benötigt ein beliebiges Access Token bzw. `read_stream` für nicht-öffentliche Postings.	Array aus Objekten vom Typ `message`
`tagged`	Fotos und Videos, in denen die Anwendung markiert wurde – *eingestellt ab 1. März 2012*	Benötigt ein beliebiges Access Token.	Array aus Objekten vom Typ `post`, `photo` oder `video`
`subscriptions`	Echtzeit-Abonnements der Anwendung	Anwendungs-Access-Token	Array aus Objekten vom Typ `subscription`
`translations`	zur Internationalisierung bereitgestellte Zeichenketten der Anwendung	Anwendungs-Access-Token	Array aus Objekten mit den Feldern `native_string` und `description`
`scores`	Punktestände des aktuellen Benutzers und seiner Freunde	Benötigt ein beliebiges Benutzer-Access-Token.	Array aus Objekten mit den Feldern `user`, `score`, `application` und `type`
`videos`	Videos, die von der Anwendung veröffentlicht wurden – *eingestellt ab 1. März 2012*	Access Token mit `read_stream`	Array aus Objekten vom Typ `video`

Tabelle 3.50 Verknüpfungen des Anwendungsobjekts (Forts.)

3.2.6 Das Domain-Objekt

Website-Domains, die etwa über die Verwendung von Social Plugins in den sozialen Graphen von Facebook integriert werden, werden in der Graph API in Objekten von

Typ `domain` gespeichert. Damit werden Domain-Namen eindeutigen Objekt-IDs auf der Facebook-Plattform zugeordnet.

Da diese Informationen öffentlich einsehbar sind, ist zum Zugriff kein Access Token notwendig. Das Lesen von Domain-Objekten kann auf zwei Arten erfolgen:

- Unter dem API-Endpunkt `https://graph.facebook.com/?domain=www.facebook.com` kann eine einzelne Domain nach ihrem Namen abgefragt werden. `https://graph.facebook.com/?domains=www.facebook.com,www.mycompany.com` erlaubt das gleichzeitige Abfragen mehrerer Domains.
- `https://graph.facebook.com/DomainID` ermöglicht den umgekehrten Weg – das Ermitteln des Domain-Namens zu einer bereits bekannten Domain-ID.

Felder des Domain-Objekts

Domain-Objekte besitzen folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Domain	keine	`string`
`name`	Domain-Name	keine	`string`

Tabelle 3.51 Felder des Domain-Objekts

3.2.7 Das Fotoobjekt

Jedes Foto im sozialen Graphen von Facebook wird in einem Objekt vom Typ `photo` gespeichert. Abhängig vom verwendeten Access Token und den Privatsphäre-Einstellungen des Fotos gestalten sich die Zugriffsmöglichkeiten:

- Ein beliebiges Access Token ist ausreichend, um ein Bild zu lesen, das mit der Privatsphäre-Einstellung `public` veröffentlicht wurde.
- Ein Benutzer-Access-Token mit der erweiterten Berechtigung `user_photos` ist notwendig, um nicht-öffentliche Fotos zu lesen, die der aktuelle Benutzer hochgeladen hat oder in denen er markiert wurde.
- Ein Benutzer-Access-Token mit der erweiterten Berechtigung `friends_photos` ist notwendig, um nicht-öffentliche Fotos zu lesen, die Freunde des aktuellen Benutzers hochgeladen haben oder in denen sie markiert wurden.
- Um Fotos auf der eigenen Pinnwand, auf der Pinnwand von Freunden oder in einem eigenen Album zu veröffentlichen, ist ein Access Token mit der erweiterten Berechtigung `publish_stream` notwendig.

Die Fotos der jeweils verknüpften Objekte können Sie unter folgenden API-Endpunkten über einen GET-Zugriff abfragen:

```
https://graph.facebook.com/BenutzerID/photos
https://graph.facebook.com/SeitenID/photos
...
```

Alternativ kann der Zugriff auf ein bestimmtes Foto direkt erfolgen:

```
https://graph.facebook.com/FotoID
```

Felder des Fotoobjekts

Fotoobjekte besitzen folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID des Fotos	Benötigt ein generisches Access Token, `user_photos` oder `friends_photos`.	`string`
`from`	Objekt-ID des Erstellers (Benutzer, Applikation oder Seite)	Benötigt ein generisches Access Token, `user_photos` oder `friends_photos`.	Array mit den Feldern `id` und `name`
`tags`	Benutzer, die auf einem Foto markiert wurden, und deren Positionen	Benötigt ein generisches Access Token, `user_photos` oder `friends_photos`.	Array an Objekten mit den Feldern `id`, `created_time`, `x`, `y` (Pixel-Koordinaten) und `name`
`name`	Titel des Fotos	Benötigt ein generisches Access Token, `user_photos` oder `friends_photos`.	`string`
`icon`	Icon-URL zur Darstellung des Fotos im Newsfeed	Benötigt ein generisches Access Token, `user_photos` oder `friends_photos`.	`string`
`picture`	URL zum Thumbnail des Fotos	Benötigt ein generisches Access Token, `user_photos` oder `friends_photos`.	`string`

Tabelle 3.52 Felder des Fotoobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
source	URL zum Foto in voller Größe	Benötigt ein generisches Access Token, user_photos oder friends_photos.	string
height	Höhe des Fotos in Pixeln	Benötigt ein generisches Access Token, user_photos oder friends_photos.	number
width	Breite des Fotos in Pixeln	Benötigt ein generisches Access Token, user_photos oder friends_photos.	number
images	vier Größenvarianten des Fotos (Thumbnail, zwei Preview-Größen, volle Größe)	Benötigt ein generisches Access Token, user_photos oder friends_photos.	Array an Objekten mit den Feldern height, width und source
link	Facebook-URL zum Foto	Benötigt ein generisches Access Token, user_photos oder friends_photos.	string
created_time	Zeitstempel, an dem das Foto erzeugt wurde	Benötigt ein generisches Access Token, user_photos oder friends_photos.	Zeitstempel im ISO-8601-Format
updated_time	Zeitstempel, an dem das Foto zuletzt aktualisiert wurde	Benötigt ein generisches Access Token, user_photos oder friends_photos.	Zeitstempel im ISO-8601-Format
position	die Reihenfolge des Fotos im Album	Benötigt ein generisches Access Token, user_photos oder friends_photos.	number

Tabelle 3.52 Felder des Fotoobjekts (Forts.)

Hinweis: Die Verwendung eines »generischen Access Tokens« bedeutet, dass das notwendige Token vom jeweiligen Mutterobjekt und den Privatsphäre-Einstellungen abhängig ist.

Verknüpfungen des Fotoobjekts

Objekte vom Typ `photo` können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`tags`	Benutzer, die in dem Foto markiert wurden	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	Array aus Objekten mit den Feldern `id` und `name`
`likes`	Benutzer, die bei diesem Foto auf GEFÄLLT MIR geklickt haben	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	Array aus Objekten mit den Feldern `id` und `name`
`comments`	Kommentare, die auf diesem Foto hinterlassen wurden	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	Array mit den Feldern `id`, `from`, `message`, `created_time` und `name`
`picture`	Album-Preview des Fotos	Benötigt ein beliebiges Access Token oder `user_photos` oder `friends_photos`.	HTTP-302-Redirect auf die Bilddatei

Tabelle 3.53 Verknüpfungen des Fotoobjekts

Veröffentlichen von Kommentaren

Kommentare zu Fotoalben können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/FotoID/comments` veröffentlicht werden. Dazu sind ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	der Kommentartext	`string`	ja

Tabelle 3.54 Felder zum Veröffentlichen von Kommentaren

War das Veröffentlichen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Kommentars.

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um ein Foto mit GEFÄLLT MIR im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt `https://graph.facebook.com/FotoID/likes`. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

Um das GEFÄLLT MIR des Fotos zu löschen, muss ein DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/FotoID/likes` gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Veröffentlichen oder Löschen des GEFÄLLT MIR erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Veröffentlichen und Lesen von Markierungen

Die Markierungen eines Fotos können Sie unter dem Endpunkt `https://graph.facebook.com/FotoID/tags` auslesen. Dazu ist analog zum Foto selbst ein passendes Access Token notwendig. Zum Hinzufügen von Tags sind hingegen immer die erweiterten Berechtigungen `publish_stream` und `user_photos` notwendig. Der entsprechende POST-Zugriff zum Markieren eines Benutzers kann auf zwei Arten erfolgen:

```
https://graph.facebook.com/FotoID/tags/BenutzerID
https://graph.facebook.com/FotoID/tags?to=BenutzerID
```

Zum Setzen der Markierung werden folgende Parameter übergeben:

Parameter	Beschreibung	Typ	Pflichtfeld
`to`	ID des zu markierenden Benutzers	`string`	nein
`x`	X-Koordinate der Markierung als prozentueller Abstand vom linken Rand	`number`	ja
`y`	Y-Koordinate der Markierung als prozentueller Abstand vom oberen Rand	`number`	ja

Tabelle 3.55 Felder zum Veröffentlichen von Markierungen

War das Setzen der Markierung erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Existierende Markierungen eines Benutzers können durch einen erneuten Aufruf des beschriebenen API-Endpunktes mittels POST aktualisiert werden. Ein Löschen von Markierungen über die Graph API ist derzeit allerdings nicht vorgesehen.

Hinweis: Die Nutzungsbedingungen zur Facebook-Plattform verlangen, dass Fotomarkierungen nur auf echten Fotos, auf denen die markierten Benutzer tatsächlich abgebildet sind, verwendet werden dürfen. Benutzer, die in ihren Privatsphäre-Einstellungen die entsprechende Einstellung aktiviert haben, können weder über die API noch über Facebook.com in Fotos markiert werden bzw. müssen die Markierung manuell bestätigen. Es können *maximal 50 Personen* auf einem Foto markiert werden.

Obwohl Seiten über Facebook.com in Fotos markiert werden können, wurde diese Funktionalität bisher nicht in der Graph API umgesetzt.

3.2.8 Das Videoobjekt

Der soziale Graph bildet Videos-Clips im Objekttyp `video` ab. Videos können in Verknüpfung mit Objekten vom Typ `user`, `application` und `page` gespeichert werden. Es stehen folgende GET-Zugriffe zum Einlesen von Videos zur Verfügung:

```
https://graph.facebook.com/BenutzerID/videos
https://graph.facebook.com/SeitenID/videos
https://graph.facebook.com/AnwendungsID/videos
```

Alternativ kann der Zugriff auf ein bestimmtes Video direkt erfolgen:

```
https://graph.facebook.com/VideoID
```

Während Videos von Seitenobjekten immer öffentlich und daher mit jedem beliebigen Access Token abrufbar sind, muss für den Zugriff auf die eigenen Videos des Benutzers die erweiterte Berechtigung `user_videos`, für Videos von Freunden die Berechtigung `friends_videos` vorhanden sein.

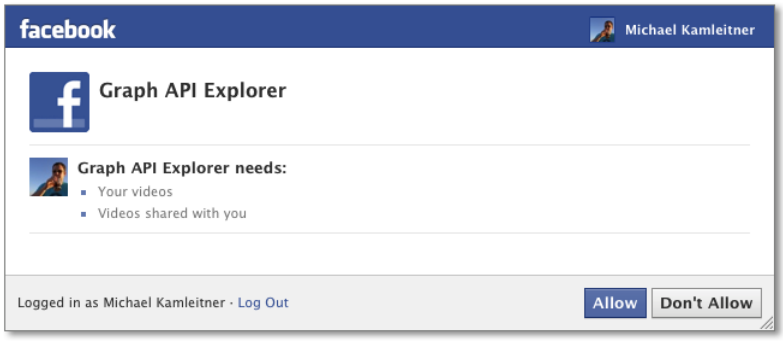


Abbildung 3.16 Autorisierung einer Anwendung mit den Berechtigungen »user_videos« und »friends_videos«

Felder des Videoobjekts

Videoobjekte besitzen folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
id	numerische Objekt-ID des Videos	user_videos oder friends_videos	string
from	der Benutzer oder die Seite, der bzw. die das Video erstellt hat	user_videos oder friends_videos	Objekt mit den Feldern id und name
tags	Benutzer, die in diesem Video markiert wurden	user_videos oder friends_videos	Array aus Objekten mit den Feldern id und name
name	Titel des Videos	user_videos oder friends_videos	string
description	Beschreibungstext des Videos	user_videos oder friends_videos	string
picture	URL zum Thumbnail-Bild des Videos	user_videos oder friends_videos	string
embed_html	HTML-Embed-Code zum Integrieren des Videos auf Drittseiten	user_videos oder friends_videos	string
icon	Icon-URL zur Darstellung des Videos im Newsfeed	user_videos oder friends_videos	string
source	URL zur Videodatei im FLV-Format	user_videos oder friends_videos	string
created_time	Zeitstempel, an dem das Video erzeugt wurde	user_videos oder friends_videos	Zeitstempel im ISO-8601-Format
updated_time	Zeitstempel, an dem das Video zuletzt aktualisiert wurde	user_videos oder friends_videos	Zeitstempel im ISO-8601-Format

Tabelle 3.56 Felder des Videoobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
comments	Kommentare zum Video	user_videos oder friends_videos	Array aus Objekten mit den Feldern id, from, message, created_time und name
format	zwei Größenvarianten des Videos (klein und groß)	user_videos oder friends_videos	Array aus Objekten mit den Feldern filter, embed_html, height, width und picture

Tabelle 3.56 Felder des Videoobjekts (Forts.)

Wichtiger Link:

- *http://developers.facebook.com/docs/reference/api/video/* – Die offizielle Dokumentation zum Videoobjekt enthält unter anderem eine vollständige Auflistung der unterstützten Videoformate.

Verknüpfungen des Videoobjekts

Objekte vom Typ video können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
likes	Benutzer, die bei diesem Video auf Gefällt mir geklickt haben	user_videos oder friends_videos	Array aus Objekten mit den Feldern id und name
comments	Kommentare, die auf diesem Video hinterlassen wurden	user_videos oder friends_videos	Array mit den Feldern id, from, message, created_time, likes und name
picture	Album-Preview des Videos	user_videos oder friends_videos	HTTP-302-Redirect auf die Bilddatei

Tabelle 3.57 Verknüpfungen des Videoobjekts

Veröffentlichen von Kommentaren

Kommentare zu Fotoalben können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/VideoID/comments` veröffentlicht werden. Dazu sind ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	der Kommentartext	`string`	ja

Tabelle 3.58 Felder zum Veröffentlichen von Kommentaren

War das Veröffentlichen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Kommentars.

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um ein Video mit GEFÄLLT MIR im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt `https://graph.facebook.com/VideoID/likes`. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

Um das GEFÄLLT MIR des Videos zu löschen, muss ein DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/VideoID/likes` gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Veröffentlichen oder Löschen des GEFÄLLT MIR erfolgreich, enthält das Ergebnis des Aufrufs `true`.

3.2.9 Das Veranstaltungsobjekt

Veranstaltungen werden im sozialen Graphen als Objekte vom Typ `event` gespeichert. Verknüpfungen von Veranstaltungen mit Objekten der Typen `user`, `page` und `application` können unter folgenden API-Endpunkten über einen GET-Zugriff abgefragt und per POST-Zugriff veröffentlicht werden:

```
https://graph.facebook.com/SeitenID/events
https://graph.facebook.com/BenutzerID/events
https://graph.facebook.com/AnwendungsID/events
```

Auch bei Veranstaltungen entscheidet das verwendete Access Token, welche Operationen auf das Objekt durchgeführt werden dürfen:

- Ein beliebiges Access Token ist ausreichend, um Veranstaltungen zu lesen, die mit Privatsphäre-Einstellungen `OPEN` veröffentlicht wurden.

- Ein Benutzer-Access-Token mit der erweiterten Berechtigung `user_events` ist notwendig, um nicht-öffentliche Veranstaltungen zu lesen.
- Ein Benutzer-Access-Token mit der erweiterten Berechtigung `friends_events` ist notwendig, um nicht-öffentliche Veranstaltungen von Freunden des aktuellen Benutzers zu lesen.
- Um den Teilnahmestatus eines Benutzers zu einer Veranstaltung zu manipulieren, ist die erweiterte Berechtigung `rsvp_event` notwendig.
- Um eine neue Veranstaltung auf dem Benutzerprofil zu veröffentlichen, ist die erweiterte Berechtigung `create_event` notwendig.
- Um eine neue Veranstaltung auf einer Seite oder einem Anwendungsprofil zu veröffentlichen, muss zusätzlich ein Seiten-Access-Token verwendet werden.
- Um Freunde zu einer neuen Veranstaltung einzuladen, ist die erweiterte Berechtigung `create_event` notwendig.

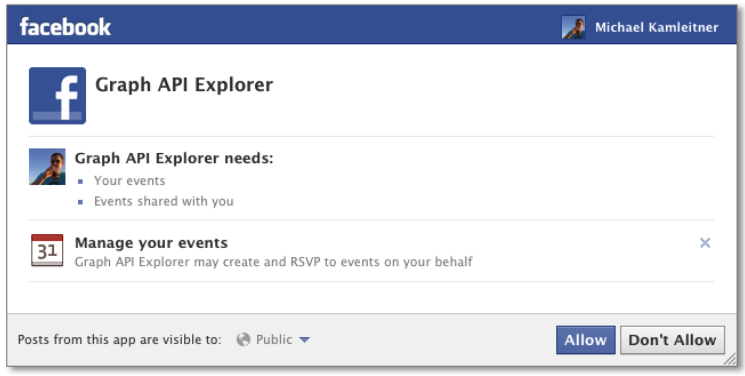


Abbildung 3.17 Autorisierung einer Anwendung mit den Berechtigungen »user_events«, »friends_events«, »create_events« und »rsvp_events«

Felder des Veranstaltungsobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische ID der Veranstaltung	generisches Access Token	`string`
`name`	Name der Veranstaltung	generisches Access Token	`string`
`description`	Beschreibung der Veranstaltung	generisches Access Token	`string`

Tabelle 3.59 Felder des Veranstaltungsobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
owner	Ersteller der Veranstaltung	generisches Access Token	Objekt mit den Feldern id und name
start_time	Zeitstempel des Beginns der Veranstaltung	generisches Access Token	Zeitstempel im ISO-8601-Format
end_time	Zeitstempel des Endes der Veranstaltung	generisches Access Token	Zeitstempel im ISO-8601-Format
location	Ort der Veranstaltung	generisches Access Token	string
venue	Ort der Veranstaltung, wenn dieses mit einem Seitenobjekt verknüpft ist	generisches Access Token	Array mit den Feldern street, city, state, zip, country, latitude und longitude
updated_time	Zeitstempel der letzten Änderung an der Veranstaltung	generisches Access Token	Zeitstempel im ISO-8601-Format
privacy_type	Privatsphäre-Einstellung – OPEN, CLOSED oder SECRET	generisches Access Token	string, Werte OPEN, CLOSED, SECRET

Tabelle 3.59 Felder des Veranstaltungsobjekts (Forts.)

Verknüpfungen des Veranstaltungsobjekts

Objekte vom Typ video können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
feed	die Pinnwand der Veranstaltung	generisches Access Token	Array aus Objekten des Typs post
noreply	Benutzer, die noch nicht auf eine Einladung geantwortet haben	generisches Access Token	Array mit den Feldern id, name und rsvp_status
invited	Benutzer, die zur Veranstaltung eingeladen wurden	generisches Access Token	Array mit den Feldern id, name und rsvp_status

Tabelle 3.60 Verknüpfungen des Veranstaltungsobjekts

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
attending	Benutzer, die an der Veranstaltung teilnehmen	generisches Access Token	Array mit den Feldern id, name und rsvp_status
maybe	Benutzer, die vielleicht an der Veranstaltung teilnehmen	generisches Access Token	Array mit den Feldern id, name und rsvp_status
declined	Benutzer, die nicht an der Veranstaltung teilnehmen	generisches Access Token	Array mit den Feldern id, name und rsvp_status
picture	URL zum Thumbnail der Veranstaltung	generisches Access Token	string

Tabelle 3.60 Verknüpfungen des Veranstaltungsobjekts (Forts.)

Veröffentlichen von Links, Wall-Postings und Status-Updates

Da Veranstaltungen eine eigene Pinnwand besitzten, die ähnlich wie die Pinnwand von Seiten oder Benutzern funktioniert, gelten die Regeln zum Veröffentlichen von Pinnwand-Einträgen – also Links, Posts und Status-Updates – analog. In allen Fällen ist ein Access Token mit der erweiterten Berechtigung `publish_stream` notwendig.

Zum Veröffentlichen muss ein POST-Zugriff auf den Endpunkt `https://graph.facebook.com/VeranstaltungsID/links` gesendet werden, die Parameter müssen Sie dabei so setzen, wie in den vorangegangenen Abschnitten beschrieben.

War das Erzeugen des Links, Posts oder Status-Updates erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des neuen Pinnwand-Eintrags.

Pinnwand-Einträge, die auf einer Veranstaltungspinnwand veröffentlicht wurden, können mit einem HTTP-DELETE-Zugriff auf folgende Endpunkte wieder gelöscht werden:

```
https://graph.facebook.com/VeranstaltungsID_LinkID
https://graph.facebook.com/VeranstaltungsID_PostID
https://graph.facebook.com/VeranstaltungsID_StatusID
```

Die ID wird dabei aus der Veranstaltungs- und der jeweiligen Post-/Link- oder Status-ID zusammengesetzt!

Veröffentlichen der Teilnahme an Veranstaltungen

Facebook-Benutzer können entscheiden, ob sie einer Veranstaltung sicherlich beiwohnen möchten (`attending`), vielleicht kommen werden (`maybe`) oder absagen (`declined`).

3.2.10 Das Post-Objekt

Posts – oder *Postings* – sind ein zentrales Kommunikationselement auf der Facebook-Plattform. Objekte, die eine Pinnwand (Verknüpfung `feed`) besitzen, sind etwa `page`, `user`, `application` und `group`. Diese Pinnwände setzen sich aus Objekten vom Typ `post` zusammen, die von Benutzern oder Seiten veröffentlicht wurden. Die neuesten veröffentlichten Postings können je nach verknüpftem Objekt unter folgenden API-Endpunkten abgefragt werden:

```
https://graph.facebook.com/SeitenID/feed
https://graph.facebook.com/BenutzerID/feed
https://graph.facebook.com/GruppenID/feed
https://graph.facebook.com/AnwendungsID/feed
```

Ein einzelnes Wall-Posting kann direkt mit der Post-ID abgefragt werden:

```
https://graph.facebook.com/PostID
```

Felder des Post-Objekts

Je nach verwendetem Access Token liefert die Graph API für Post-Objekte unterschiedliche Felder zurück. Während ein beliebiges Access Token nur die öffentlich zugänglichen Felder offenbart, können mit der erweiterten Berechtigung `read_stream` weitere Informationen ausgelesen werden.

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische ID des Posts	beliebiges Access Token	`string`
`From`	Ersteller des Posts (Benutzer oder Seite)	beliebiges Access Token	Objekt mit den Feldern `id` und `name`
`to`	Benutzer oder Seiten, die in dem Post markiert wurden	beliebiges Access Token	Array aus Objekten mit den Feldern `id` und `name`

Tabelle 3.61 Felder des Post-Objekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
message	der Nachrichtentext des Posts	beliebiges Access Token	string
message_tags	Detailinformationen zu den Benutzern oder Seiten, die in dem Post markiert wurden	beliebiges Access Token	Array aus Objekten, deren numerischer Index die Position der Markierung angibt. Die Objekte enthalten die Felder id, offset, length und name.
picture	URL zum Bild, das zum Post hinzugefügt wurde	beliebiges Access Token	string
link	Link, der zum Post hinzugefügt wurde	beliebiges Access Token	string
name	Ankertext des Links	beliebiges Access Token	string
caption	Überschrift, die zum Post hinzugefügt wurde	beliebiges Access Token	string
description	Beschreibungstext, der zum Post hinzugefügt wurde	beliebiges Access Token	string
source	URL zu einem eingebetteten Flash-Video oder einer Videodatei	beliebiges Access Token	string
properties	zusätzliche Attribute des Posts	beliebiges Access Token	Array aus Objekten mit den Feldern id, href und name
icon	Icon-URL zur Darstellung des Posts im Newsfeed	beliebiges Access Token	string
actions	Liste der Action-Links, die unter dem Post angezeigt werden	beliebiges Access Token	Array aus Objekten mit den Feldern name und link

Tabelle 3.61 Felder des Post-Objekts (Forts.)

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`privacy`	Privatsphäre-Einstellungen des Posts	Access Token mit `read_stream`	ein JSON-codiertes Objekt, das die Privatsphäre-Einstellungen detailliert regelt
`type`	Typ des Posts (`link`, `status`, `photo`, `video`)	beliebiges Access Token	`string`
`likes`	GEFÄLLT MIR des Posts	beliebiges Access Token	Objekt, das in `count` die Zahl der »Likes« und im Array `data` Objekte mit den Feldern `id` und `name` enthält
`place`	Ort, der mit dem Post verknüpft ist	Access Token mit `read_stream`	Objekt mit den Feldern `id`, `name` und `location`. `location` enthält die Koordinaten und die Adresse des Ortes.
`story`	Enthält den Text einer Feed-Story, die automatisch beim Veröffentlichen des Posts von Facebook erstellt wurde. Zum Beispiel: *Michael was tagged in Sabines photo.*	Access Token mit `read_stream`	`string`
`story_tags`	Objekte, die in der Feed-Story markiert wurden	Access Token mit `read_stream`	Array aus Objekten, deren numerischer Index die Position der Markierung angibt. Die Objekte enthalten die Felder `id`, `offset`, `length` und `name`.

Tabelle 3.61 Felder des Post-Objekts (Forts.)

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
comments	Kommentare zum Post	Access Token mit read_stream	Objekt, das in count die Zahl der Kommentare und im Array data Objekte mit den Feldern id, from, message und created_time enthält
object_id	numerische Objekt-ID eines hochgeladenen Fotos oder Videos	Access Token mit read_stream	string
application	Anwendung, über die das Post veröffentlicht wurde	Access Token mit read_stream	Objekt mit den Feldern id, href und name
created_time	der Zeitstempel der Erstellung des Posts	Access Token mit read_stream	string mit einem ISO-8601-Zeitstempel
updated_time	der Zeitstempel des letzten Kommentars auf das Post	Access Token mit read_stream	string mit einem ISO-8601-Zeitstempel
targeting	Länder- und Sprachenbeschränkung des Posts	Access Token mit manage_pages	Objekt mit den Listen country, city, region und locale

Tabelle 3.61 Felder des Post-Objekts (Forts.)

Das privacy-Objekt kann dabei folgende Felder enthalten:

- value – Die primäre Privatsphäre-Einstellung, kann auf EVERYONE (Post ist vollkommen öffentlich), CUSTOM (benutzerdefinierte Einstellungen), ALL_FRIENDS (Post ist für die Freunde des Erstellers sichtbar), NETWORKS_FRIENDS (Post ist für Freunde und Netzwerke des Benutzers sichtbar), FRIENDS_OF_FRIENDS (Post ist für Freunde und deren Freunde sichtbar) und SELF (Post ist nur für den Benutzer selbst sichtbar) gesetzt werden.
- friends – Wurde zuvor bei value die Einstellung CUSTOM gewählt, kann hier festgelegt werden, welche Benutzer das Post sehen können. Mögliche Werte sind EVERYONE, SOME_FRIENDS, ALL_FRIENDS, NETWORKS_FRIENDS, FRIENDS_OF_FRIENDS und NO_FRIENDS (nur für Netzwerke sichtbar).

- `networks` – Wurde zuvor bei `value` die Einstellung `CUSTOM` gewählt, kann hier eine kommaseparierte Liste an Netzwerk-IDs angegeben werden, die das Post sehen kann. Der Wert 1 bedeutet, dass alle Netzwerke auf das Post zugreifen können.
- `allow` – Wurde zuvor bei `friends` die Einstellung `SOME_FRIENDS` gewählt, können hier mit einer kommaseparierten Liste die IDs jener Benutzer angegeben werden, die Zugriff auf das Post erhalten sollen.
- `deny` – Wurde zuvor bei `friends` die Einstellung `SOME_FRIENDS` gewählt, können hier mit einer kommaseparierten Liste die IDs jener Benutzer angegeben werden, die keinen Zugriff auf das Post erhalten sollen.

Hinweis: Die Privatsphäre-Einstellungen für Posts gelten nur, wenn diese auf der *eigenen Pinnwand* des aktuellen Benutzers veröffentlicht werden. Wird hingegen ein Post auf der Pinnwand einer Seite oder eines Freundes veröffentlicht, gelten automatisch die dort maßgeblichen Privatsphäre-Einstellungen!

Das Veröffentlichen von Posts erfolgt analog zur Beschreibung in Abschnitt 3.2.1, »Das User-Objekt«.

Löschen von Post-Objekten

Um ein veröffentlichtes Post zu löschen, muss ein DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/PostID` gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Löschen des Posts erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Verknüpfungen des Post-Objekts

Objekte vom Typ `post` können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`likes`	Benutzer, die bei diesem Post auf Gefällt mir geklickt haben	`user_notes` oder `friends_notes`	Array aus Objekten mit den Feldern `id` und `name`
`comments`	Kommentare, die zu diesem Post hinterlassen wurden	`user_notes` oder `friends_notes`	Array mit den Feldern `id`, `from`, `message`, `created_time`, `likes` und `name`

Tabelle 3.62 Verknüpfungen des Post-Objekts

Veröffentlichen und Löschen von Kommentaren

Kommentare zu Posts können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/PostID/comments` veröffentlicht werden. Dazu sind ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	der Kommentartext	`string`	ja

Tabelle 3.63 Felder zum Veröffentlichen von Kommentaren

Um einen veröffentlichten Kommentar zu löschen, senden Sie einen DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/KommentarID`. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Veröffentlichen oder Löschen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Kommentars.

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um ein Post mit Gefällt mir im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt `https://graph.facebook.com/PostID/likes`. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

Um das Gefällt mir des Posts wieder zu löschen, muss ein DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/PostID/likes` gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Veröffentlichen oder Löschen des Gefällt mir erfolgreich, enthält das Ergebnis des Aufrufs `true`.

3.2.11 Das Linkobjekt

Links sind eine besondere Form von Pinnwand-Postings, die sich dadurch auszeichnen, dass sie einen Link zu einer anderen Webseite enthalten. Objekte vom Typ `link` bilden diese Postings im sozialen Graphen ab und können verknüpft mit `page`, `user` und `application` auftreten. Während öffentlich sichtbar gepostete Links mit einem beliebigen Access Token gelesen werden können, ist die erweiterte Berechtigung `read_stream` zum Lesen von nicht-öffentlichen Links des aktuellen Benutzers notwendig.

Veröffentlichte Links können je nach verknüpftem Objekttyp unter folgenden API-Endpunkten abgefragt werden:

```
https://graph.facebook.com/SeitenID/links
https://graph.facebook.com/BenutzerID/links
https://graph.facebook.com/AnwendungsID/links
```

Ein einzelner Link kann direkt mit der Link-ID abgefragt werden:

```
https://graph.facebook.com/LinkID
```

Felder des Linkobjekts

Während öffentlich gepostete Links mit einem beliebigen Access Token gelesen werden können, erlaubt die erweiterte Berechtigung `read_stream` auch das Lesen der eigenen, nicht-öffentlichen Links.

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische ID des Links	beliebiges Access Token oder `read_stream`	`string`
`from`	ID des Benutzers, der den Link gepostet hat	beliebiges Access Token oder `read_stream`	Objekt mit den Feldern `id` und `name`
`link`	URL des geteilten Links	beliebiges Access Token oder `read_stream`	`string`
`name`	Linktext bzw. Titel der geteilten URL	beliebiges Access Token oder `read_stream`	`string`
`comments`	Kommentare, die zu diesem Link veröffentlicht wurden	`read_stream`	Array aus Objekten mit den Feldern `id`, `from`, `message` und `created_time`
`description`	Beschreibuntgstext des geteilten Links	beliebiges Access Token oder `read_stream`	`string`

Tabelle 3.64 Felder des Linkobjekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`icon`	URL zum Icon des Links	beliebiges Access Token oder `read_stream`	`string`
`picture`	URL zum Thumbnail-Bild des Links	beliebiges Access Token oder `read_stream`	`string`
`message`	vom teilenden Benutzer verfasste Nachricht zum Link	beliebiges Access Token oder `read_stream`	`string`
`created_time`	der Zeitstempel der Erstellung des Links	beliebiges Access Token oder `read_stream`	`string` mit einem ISO-8601-Zeitstempel
`type`	Typ des Objekts (immer `link`)	beliebiges Access Token oder `read_stream`	`string`

Tabelle 3.64 Felder des Linkobjekts (Forts.)

Verknüpfungen des Linkobjekts

Objekte vom Typ `link` können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`likes`	Benutzer, die bei diesem Link auf Gefällt mir geklickt haben	`read_stream`	Array aus Objekten mit den Feldern `id` und `name`
`comments`	Kommentare, die zu diesem Link hinterlassen wurden	`read_stream`	Array mit den Feldern `id`, `from`, `message`, `created_time`, `likes` und `name`

Tabelle 3.65 Verknüpfungen des Linkobjekts

Veröffentlichen von Kommentaren

Kommentare zu Links können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/LinkID/comments` veröffentlicht werden. Dazu sind ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	der Kommentartext	`string`	ja

Tabelle 3.66 Felder zum Veröffentlichen von Kommentaren

War das Veröffentlichen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Kommentars.

3.2.12 Das Fragenobjekt

Seit 2010 bietet Facebook Benutzern die Möglichkeit, Fragen an Freunde in Form von Multiple-Choice-Umfragen auf der Pinnwand zu stellen. Diese Fragen werden in Objekten vom Typ `question` im sozialen Graphen dargestellt. Der API-Endpunkt zum Auslesen von Fragen lautet `https://graph.facebook.com/BenutzerID/questions` und muss mit einem GET-Zugriff aufgerufen werden.

Um die Fragen des aktuellen Benutzers auszulesen, ist die erweiterte Berechtigung `user_questions` notwendig, während `friends_questions` Zugriff auf die Fragen der Freunde des Benutzers gibt. Ein Veröffentlichen von neuen Fragen ist in der Graph API derzeit nicht vorgesehen.

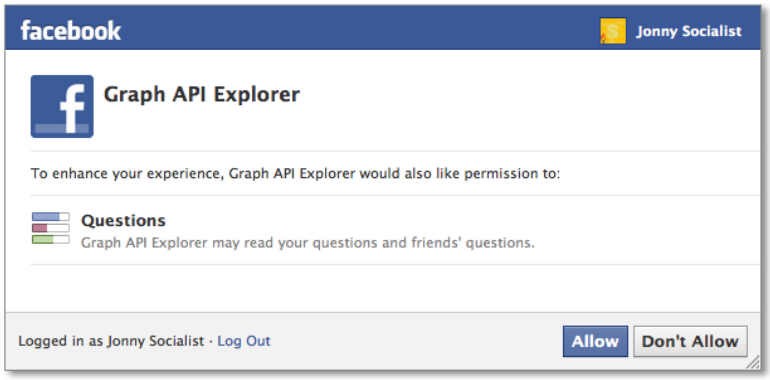


Abbildung 3.18 Autorisierung einer Anwendung mit den Berechtigungen »user_questions« und »friends_questions«

Felder des Fragenobjekts

Aufrufe an den API-Endpunkt liefern Objekte mit folgenden Feldern zurück:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Frage	`user_questions` oder `friends_questions`	`string`
`from`	Benutzer, der die Frage gestellt hat	`user_questions` oder `friends_questions`	Objekt mit den Feldern `id` und `name`
`question`	Text mit der Fragestellung	`user_questions` oder `friends_questions`	`string`
`created_time`	der Zeitstempel der Erstellung der Frage	`user_questions` oder `friends_questions`	`string` mit einem ISO-8601-Zeitstempel
`updated_time`	der Zeitstempel der letzten Änderung an der Frage	`user_questions` oder `friends_questions`	`string` mit einem ISO-8601-Zeitstempel
`options`	Liste der Fragenoptionen	`user_questions` oder `friends_questions`	Array von Objekten des Typs `questionoption`

Tabelle 3.67 Felder des Fragenobjekts

Verknüpfungen des Fragenobjekts

Objekte vom Typ `question` können im sozialen Graphen ausschließlich mit Objekten vom Typ `questionoption` verknüpft werden.

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`Options`	Fragenoptionen der Frage	`user_questions` oder `friends_questions`	Array von Objekten des Typs `questionoption`

Tabelle 3.68 Verknüpfungen des Fragenobjekts

3.2.13 Das Fragen-Optionsobjekt

Objekte vom Typ `questionsoptions` enthalten die Fragenoptionen der Umfrage und können entweder über den bereits genannten API-Endpunkt `https://graph.facebook.com/BenutzerID/questions` für alle Fragen oder unter `https://graph.facebook.com/FragenID/options` für eine bestimmte Frage abgerufen werden.

Felder des Fragen-Optionsobjekts

Fragen-Optionsobjekte enthalten folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Fragenoption	`user_questions` oder `friends_questions`	`string`
`from`	Benutzer, der die Anwortoption eingegeben hat	`user_questions` oder `friends_questions`	Objekt mit den Feldern `id` und `name`
`votes`	Anzahl der Stimmen	`user_questions` oder `friends_questions`	`integer`
`object`	optionale Seite, die mit der Fragenoption verknüpft wurde	`user_questions` oder `friends_questions`	Objekt mit den Feldern `id`, `name` und `category`
`created_time`	der Zeitstempel der Erstellung der Frage	`user_questions` oder `friends_questions`	`string` mit einem ISO-8601-Zeitstempel

Tabelle 3.69 Felder des Fragen-Optionsobjekts

Verknüpfungen des Fragen-Optionsobjekts

Die einzelnen Fragen-Optionsobjekte verraten über das Feld `votes` nicht nur die Anzahl der Stimmen, die einzelnen Stimmen können über den API-Endpunkt `https://graph.facebook.com/FragenoptionsID/votes` auch detailliert abgefragt werden.

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`votes`	Benutzer, die für die Fragenoption gestimmt haben	`user_questions` oder `friends_questions`	Array aus Objekten mit den Feldern `id` und `name`

Tabelle 3.70 Verknüpfungen des Fragen-Optionsobjekts

3.2.14 Das Checkin-Objekt

Besucht ein Facebook-Benutzer einen Ort, der durch ein `page`-Objekt im sozialen Graphen repräsentiert wird, hat er mit der mobilen Facebook-Anwendung am Smartphone die Möglichkeit, den aktuellen Aufenthaltsort per *Checkin* seinen Freunden mitzuteilen. Diese Interaktion wird im Graphen im Objekttyp `checkin` gespeichert. Checkins können unter folgenden Voraussetzungen über die Graph API gelesen werden:

- Das Lesen der Checkins des aktuellen Benutzers erfordert die erweiterte Berechtigung `user_checkins`.
- Das Lesen der Checkins der Freunde des aktuellen Benutzers erfordert die erweiterte Berechtigung `friends_checkins`.

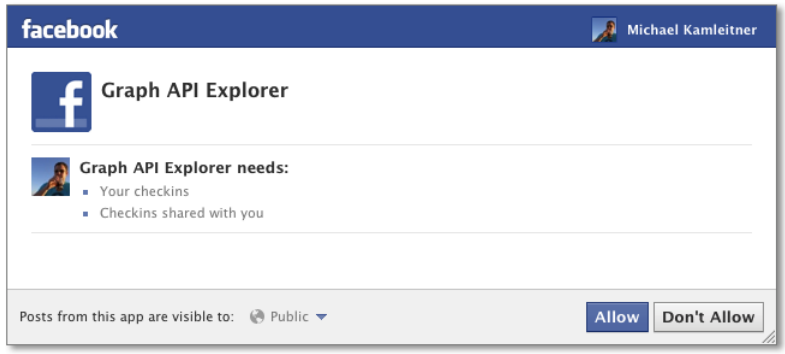


Abbildung 3.19 Autorisierung einer Anwendung mit den Berechtigungen »user_checkins« und »friends_checkins«

Felder des Checkin-Objekts

Checkins können mittels GET-Zugriff auf den API-Endpunkt `https://graph.facebook.com/BenutzerID/checkins` eingelesen werden. Das Ergebnis ist ein JSON-Array an Objekten mit folgenden Feldern:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID des Checkins	`user_checkins` oder `friends_checkins`	`string`
`from`	Benutzer, der den Checkin veröffentlicht hat	`user_checkins` oder `friends_checkins`	Objekt mit den Feldern `id` und `name`

Tabelle 3.71 Felder des Checkin-Objekts

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
tags	andere Benutzer, die der Ersteller im Check-in markiert hat	user_checkins oder friends_checkins	Array von Objekten mit den Feldern id und name
place	Ort, an dem der Benutzer eingecheckt hat	user_checkins oder friends_checkins	Objekt mit den Feldern id, name und location. location enthält die Koordinaten und die Adresse des Ortes.
application	Anwendung, mit der der Checkin veröffentlicht wurde	user_checkins oder friends_checkins	Objekt mit den Feldern id und name
created_time	Zeitstempel der Erstellung des Checkins	user_checkins oder friends_checkins	string mit einem ISO-8601-Zeitstempel
likes	Anzahl der Gefällt mir des Checkins	user_checkins oder friends_checkins	Array von Objekten mit den Feldern id und name
message	Nachricht, die der Benutzer zum Checkin verfasst hat	user_checkins oder friends_checkins	string
comments	Kommentare zum Checkin	user_checkins oder friends_checkins	Array von Objekten mit den Feldern id, from, message und created_time
Type	Typ des Objekts – immer checkin	user_checkins oder friends_checkins	string

Tabelle 3.71 Felder des Checkin-Objekts (Forts.)

Verknüpfungen des Checkin-Objekts

Objekte vom Typ checkin können im sozialen Graphen mit folgenden Objekttypen verknüpft werden:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
comments	Kommentare zum Checkin	user_checkins oder friends_checkins	Array aus Objekten mit den Feldern id, from, message und created_time
likes	Gefällt mir zum Checkin	user_checkins oder friends_checkins	Array aus Objekten mit den Feldern id und name

Tabelle 3.72 Verknüpfungen des Checkin-Objekts

Veröffentlichen von Kommentaren

Kommentare zu Checkins können mit einem POST-Zugriff auf den API-Endpunkt https://graph.facebook.com/CheckinID/comments veröffentlicht werden. Dazu sind ein Access Token mit der erweiterten Berechtigung publish_stream und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
message	der Kommentartext	string	ja

Tabelle 3.73 Felder zum Veröffentlichen von Kommentaren

War das Veröffentlichen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld id die Objekt-ID des Kommentars.

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um einen Checkin mit Gefällt mir im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt https://graph.facebook.com/CheckinID/likes. Das verwendete Access Token muss die erweiterte Berechtigung publish_stream besitzen.

Um das Gefällt mir des Checkins zu löschen, muss ein DELETE-Zugriff an den Endpunkt https://graph.facebook.com/CheckinID/likes gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung publish_stream beinhalten.

War das Veröffentlichen oder Löschen des Gefällt mir erfolgreich, enthält das Ergebnis des Aufrufs true.

3.2.15 Das Kommentarobjekt

Kommentare werden im sozialen Graphen von Facebook in Objekten vom Typ `comment` gespeichert. Kommentare können mit einer Vielzahl an Objekttypen verknüpft sein: `album`, `link`, `notes`, `photo`, `post`, `status message` und `video`. Dementsprechend können die Kommentare dieser Objekte mit folgenden API-Endpunkten mittels GET-Zugriff abgefragt werden:

```
https://graph.facebook.com/AlbumID/comments
https://graph.facebook.com/LinkID/comments
https://graph.facebook.com/PostID/comments
...
```

Felder des Kommentarobjekts

Das Ergebnis ist jeweils ein Array von JSON-Objekten mit folgenden Feldern:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID des Kommentars	Benötigt ein generisches Access Token.	`string`
`from`	Objekt-ID des Erstellers (Benutzer oder Seite)	Benötigt ein generisches Access Token.	Array mit den Feldern `id` und `name`
`message`	der Kommentartext	Benötigt ein generisches Access Token.	`string`
`created_time`	der Zeitstempel der Erstellung des Kommentars	Benötigt ein generisches Access Token.	`string` mit einem ISO-8601-Zeitstempel
`likes`	Anzahl der Gefällt-mir-Klicks des Kommentars	Benötigt ein generisches Access Token.	`integer`
`user_likes`	Zeigt, ob der aktuelle Benutzer den Kommenar mit Gefällt mir ausgezeichnet hat.	Benötigt ein generisches Access Token.	`boolean`
`type`	der Typ des Objekts (immer `comment`)	Benötigt ein generisches Access Token.	`string`

Tabelle 3.74 Felder des Kommentarobjekts

Hinweis: Die Verwendung eines »generischen Access Tokens« bedeutet, dass das notwendige Token vom jeweiligen Mutterobjekt abhängig ist.

Um einen Kommentar zu veröffentlichen, muss ein POST-Zugriff auf den entsprechenden Endpunkt verwendet werden. Dazu sind ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgender Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	der Kommentartext	`string`	ja

Tabelle 3.75 Felder zum Veröffentlichen von Kommentaren

Das Löschen eines Kommentars erfolgt mittels DELETE-Zugriff auf den API-Endpunkt `https://graph.facebook.com/KommentarID`. Auch dazu ist ein Access Token mit der erweiterten Berechtigung `publish_stream` notwendig.

War das Veröffentlichen oder Löschen des Kommentars erfolgreich, enthält das Ergebnis des Aufrufs `true`.

Verknüpfungen des Kommentarobjekts

Objekte vom Typ `comment` können im sozialen Graphen lediglich mit dem Objekttyp `like` verknüpft werden.

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`likes`	Gefällt mir des Kommentars	generisches Access Token	Array aus Objekten mit den Feldern `id`, `from`, `message` und `created_time`

Tabelle 3.76 Verknüpfungen des Kommentarobjekts

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um einen Kommentar mit Gefällt mir im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt `https://graph.facebook.com/KommentarID/likes`. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` besitzen.

Um das Gefällt mir des Kommentars zu löschen, muss ein DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/KommentarID/likes` gesendet werden. Das

verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Veröffentlichen oder Löschen des Kommentars erfolgreich, enthält das Ergebnis des Aufrufs `true`.

3.2.16 Das Notizenobjekt

Unter *Notizen* versteht man im sozialen Graphen von Facebook längere Texte, die ähnlich einem Blog-Posting auf einer Seite oder einem Benutzerprofil veröffentlicht werden. Die veröffentlichten Notizen einer Seite oder eines Benutzers können über folgende API-Endpunkte gelesen werden:

```
https://graph.facebook.com/BenutzerID/notes
https://graph.facebook.com/SeitenID/notes
```

Alternativ kann der Zugriff auf eine bestimmte Notiz direkt erfolgen:

```
https://graph.facebook.com/NotizenID
```

Das verwendete Access Token bestimmt die möglichen Zugriffe auf Notizen:

- Ein beliebiges Access-Token ist ausreichend, um Notizen zu lesen, die mit der Privatsphäre-Einstellung `public` veröffentlicht wurden.
- Ein Benutzer-Access-Token mit der erweiterten Berechtigung `user_notes` ist notwendig, um nicht-öffentliche Notizen zu lesen, die vom aktuellen Benutzer erstellt wurden.
- Ein Benutzer-Access-Token mit der erweiterten Berechtigung `friends_notes` ist notwendig, um nicht-öffentliche Notizen zu lesen, die Freunde des aktuellen Benutzers erstellt haben.

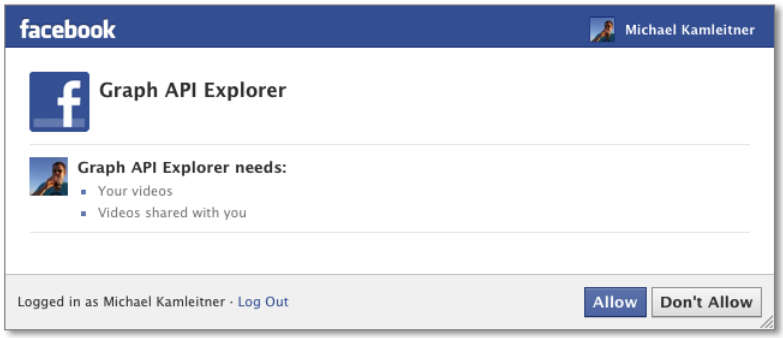


Abbildung 3.20 Autorisierung einer Anwendung mit den Berechtigungen »user_videos« und »friends_videos«

Felder des Notizenobjekts

Objekte vom Typ `note` enthalten folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Notiz	`user_notes` oder `friends_notes`	`string`
`from`	Objekt-ID des Erstellers (Benutzer oder Seite)	`user_notes` oder `friends_notes`	Array mit den Feldern `id` und `name`
`subject`	Betreff der Notiz	`user_notes` oder `friends_notes`	`string`
`message`	der Text der Notiz	`user_notes` oder `friends_notes`	`string` mit HTML-Unterstützung
`created_time`	der Zeitstempel der Erstellung des Notiz	`user_notes` oder `friends_notes`	`string` mit einem ISO-8601-Zeitstempel
`updated_time`	der Zeitstempel der letzten Änderung an der Notiz	`user_notes` oder `friends_notes`	`string` mit einem ISO-8601-Zeitstempel
`comments`	Kommentare zur Notiz	`user_notes` oder `friends_notes`	Array aus Objekten mit den Feldern `id`, `from`, `message`, `created_time` und `name`
`icon`	Icon-URL zur Darstellung der Notiz im Newsfeed	`user_notes` oder `friends_notes`	`string`

Tabelle 3.77 Felder des Notizenobjekts

Verknüpfungen des Notizenobjekts

Objekte vom Typ `note` können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`likes`	Benutzer, die bei dieser Notiz auf GEFÄLLT MIR geklickt haben	`user_notes` oder `friends_notes`	Array aus Objekten mit den Feldern `id` und `name`
`comments`	Kommentare, die auf dieser Notiz hinterlassen wurden	`user_notes` oder `friends_notes`	Array mit den Feldern `id`, `from`, `message`, `created_time`, `likes` und `name`

Tabelle 3.78 Verknüpfungen des Notizenobjekts

Veröffentlichen von Kommentaren

Kommentare zu Notizen können mit einem POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/NotizID/comments` veröffentlicht werden. Dazu ist ein Access Token mit der erweiterten Berechtigung `publish_stream` und folgendem Parameter notwendig:

Parameter	Beschreibung	Typ	Pflichtfeld
`message`	der Kommentartext	`string`	ja

Tabelle 3.79 Felder zum Veröffentlichen von Kommentaren

War das Veröffentlichen des Kommentars erfolgreich, enthält das JSON-Ergebnis im Feld `id` die Objekt-ID des Kommentars.

Veröffentlichen und Löschen von »Gefällt mir« (»Likes«)

Um eine Notiz mit GEFÄLLT MIR im Namen des aktuellen Benutzers auszuzeichnen, genügt ein POST-Zugriff ohne weitere Parameter auf den Endpunkt `https://graph.facebook.com/NotizID/likes`. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

Um das GEFÄLLT MIR des Videos zu löschen, muss ein DELETE-Zugriff an den Endpunkt `https://graph.facebook.com/NotizID/likes` gesendet werden. Das verwendete Access Token muss die erweiterte Berechtigung `publish_stream` beinhalten.

War das Veröffentlichen oder Löschen des GEFÄLLT MIR erfolgreich, enthält das Ergebnis des Aufrufs `true`.

3.2.17 Das Review-Objekt

Facebook erlaubt es Benutzern, Anwendungen in Reviews zu bewerten, die im Graphen in Objekten vom Typ `review` gespeichert werden. Da diese Bewertungen immer öffentlich sind, können die Reviews einer Anwendung über den API-Endpunkt `https://graph.facebook.com/AnwendungsID/reviews` ohne Access Token ausgelesen werden. Das Veröffentlichen von Reviews wird von der Graph API derzeit nicht unterstützt.

Felder des Review-Objekts

Objekte vom Typ `review` enthalten folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID des Reviews	keine	`string`
`from`	Benutzer, der die Review erstellt hat	keine	Objekt mit den Feldern `id` und `name`
`to`	die Anwendung, auf die sich die Review bezieht	keine	Objekt mit den Feldern `id` und `name`
`message`	Review-Text	keine	`string`
`rating`	Bewertung des Reviews von 1 bis 5	keine	`Integer`
`created_time`	der Zeitstempel der Erstellung des Reviews	keine	`string` mit einem ISO-8601-Zeitstempel

Tabelle 3.80 Felder des Review-Objekts

3.2.18 Das Abonnementobjekt

Wie in Abschnitt 3.5, »Echtzeit-Zugriff auf die Graph API«, behandelt wird, bietet die Graph API Entwicklern die Möglichkeit, über Änderungen bestimmter Objekttypen und -felder in Echtzeit benachrichtigt zu werden. Diese Benachrichtigungen müssen

von einer Anwendung explizit abonniert werden. Die Abonnementinformationen werden dabei in Objekten vom Typ `subscription` gespeichert.

Felder des Abonnementobjekts

Der API-Endpunkt `https://graph.facebook.com/AnwendungsID/subscriptions` liefert mittels GET-Zugriff unter Verwendung eines Anwendungs-Access-Tokens die aktuell aktiven Abonnements der Anwendung mit folgenden Feldern zurück:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`object`	Objekttyp, auf den sich das Abonnement bezieht – derzeit nur `user`, `page` oder `permission`	Anwendungs-Access-Token	`string`
`fields`	Felder, deren Änderungen abonniert wurden	Anwendungs-Access-Token	`string` mit kommaseparierter Liste der Feldnamen
`callback_url`	Endpunkt der Anwendung, an die Facebook Benachrichtigungen ausliefert	Anwendungs-Access-Token	`string`

Tabelle 3.81 Felder des Abonnementobjekts

Detaillierte Hinweise zur Verwendung von Echtzeit-Abonnements finden Sie in Abschnitt 3.5, »Echtzeit-Zugriff auf die Graph API«.

3.2.19 Das Konversationenobjekt

Facebook bietet mit der Graph API auch die Möglichkeit, lesend auf das Facebook-Postfach des Benutzers zugreifen zu können. Dazu ist immer die erweiterte Berechtigung `read_mailbox` notwendig. Das Facebook-Postfach ist in Objekte vom Typ `thread` (»Konversationen«) strukturiert, die dann die eigentlichen Nachrichten vom Typ `message` enthalten.

Wichtig: Die Graph API bietet ausschließlich *Lesezugriff* auf die Postfächer des *aktuellen Benutzers*!

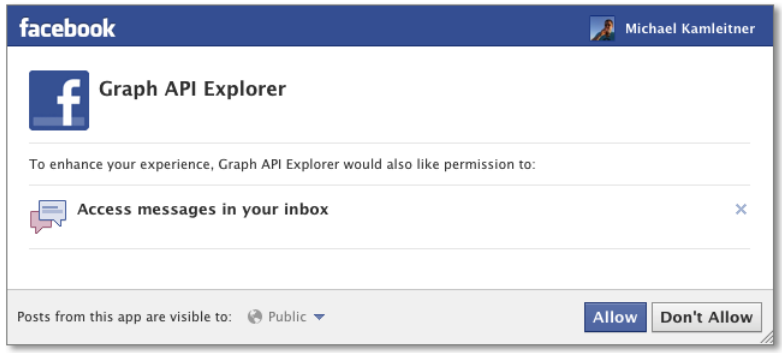


Abbildung 3.21 Autorisierung einer Anwendung mit der Berechtigung »read_mailbox«

Felder des Konversationenobjekts

Alle Konversationen des Benutzerpostfachs können am API-Endpunkt `https://graph.facebook.com/inbox` abgefragt werden, einzelne Konversationen mittels `https://graph.facebook.com/ThreadID`. Beide Methoden enthalten im Ergebnis Arrays mit folgenden Feldern:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Konversation	`read_mailbox`	`string`
`snippet`	kurzer Ausschnitt aus der Konversation zur Voransicht	`read_mailbox`	`string`
`updated_time`	der Zeitstempel der letzten Änderung an der Konversation	`read_mailbox`	`string` mit einem ISO-8601-Zeitstempel
`message_count`	Anzahl der Nachrichten in dieser Konversation	`read_mailbox`	`integer`
`unread_count`	Anzahl der ungelesenen Nachrichten in dieser Konversation	`read_mailbox`	`integer`

Tabelle 3.82 Felder des Konversationenobjekts

Verknüpfungen des Konversationenobjekts

Objekte vom Typ `thread` können mit folgenden Objekttypen verknüpft sein:

Objekttyp	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`Tags`	Markierungen in der Konversation	`read_mailbox`	Array aus Objekten mit dem Feld `name`
`participants`	Teilnehmer der Konversation	`read_mailbox`	Array aus Objekten mit den Feldern `id`, `email` und `name`
`former_participants`	Teilnehmer, die die Konversation verlassen haben (`unsubscribe`)	`read_mailbox`	Array aus Objekten mit den Feldern `id`, `email` und `name`
`senders`	Teilnehmer, die eine Nachricht in der Konversation veröffentlicht haben	`read_mailbox`	Array aus Objekten mit den Feldern `id`, `email` und `name`
`messages`	Nachrichten in dieser Konversation	`read_mailbox`	Array aus Objekten vom Typ `message`

Tabelle 3.83 Verknüpfungen des Konversationenobjekts

3.2.20 Das Nachrichtenobjekt

Einzelne Nachrichten werden im sozialen Graphen von Facebook als Objekte vom Typ `message` abgebildet. Zum Zugriff auf einzelne Nachrichten gelten die in Abschnitt 3.2.12, »Das Fragen-Objekt«, genannten Voraussetzungen, also ein Access Token mit `read_mailbox`-Berechtigung.

Zur Adressierung aller Nachrichten einer Konversation dient der API-Endpunkt `https:///graph.facebook.com/KonversationsID`, während eine einzelne Nachricht auch mit `https://graph.facebook.com/NachrichtenID` abgerufen werden kann. Die ID der einzelnen Nachricht setzt sich dabei aus der Konversations-ID, gefolgt von einem Unterstrich und einer fortlaufenden Nummer, zusammen.

Felder des Nachrichtenobjekts

Objekte vom Typ `message` enthalten folgende Felder:

Feldname	Beschreibung	Benötigte Zugriffsrechte	Ergebnis
`id`	numerische Objekt-ID der Nachricht	`read_mailbox`	`string`
`created_time`	der Zeitstempel der Erstellung der Nachricht	`read_mailbox`	`string` mit einem ISO-8601-Zeitstempel
`from`	Absender der Nachricht	–	Objekt mit den Feldern `id` und `name`
`to`	Liste der Empfänger der Nachricht	–	Array aus Objekten mit den Feldern `id` und `name`
`Message`	der Nachrichtentext	–	`string`

Tabelle 3.84 Felder des Nachrichtenobjekts

3.3 Performance und Caching der Graph API

Grundsätzlich ist es möglich, die Graph API immer dann abzufragen, wenn Informationen über ein bestimmtes Graph-Objekt benötigt werden. In der Praxis ist so ein Vorgehen allerdings meistens nicht besonders performant, müssen doch die gleichen Daten immer wieder abgefragt werden. Daher empfiehlt es sich, bestimmte Inhalte der Graph API lokal, also am Server, auf dem Ihre Anwendung läuft, zwischenzuspeichern (*Caching*).

Beispiel: Benutzerdatenbank ohne Caching

Fast jede Anwendung auf der Facebook-Plattform benötigt einfache Datenstrukturen und Methoden zum Verwalten der Applikationsbenutzer. Üblicherweise wird dazu eine Tabelle in der lokalen Datenbank des Webservers genutzt. Diese Tabelle `users` könnte im einfachsten Fall etwa so aussehen:

```
mysql> create table users (
   id bigint(20) not null primary key,
   created datetime default null
);
```

Der Entwickler, der diese Benutzertabelle entworfen hat, war vielleicht von folgendem Gedanken geleitet: »Letztlich müssen wir, um die Benutzer unserer Anwendung zu identifizieren, nur die numerische Facebook-Benutzer-ID speichern. Praktischerweise können Sie die `id` auch als Primärschlüssel verwenden, da sie im sozialen Graphen eindeutig ist. Das Feld `created` verwenden wir, um festzuhalten, wann ein Benutzerdatensatz hinzugefügt worden ist. Alles Weitere, also den Vor- und Nachnamen des Benutzers, sein Profilbild etc., können Sie zur Laufzeit ja mithilfe der Graph API ermitteln.«

Während dieser hypothetische Entwickler grundsätzlich nicht Unrecht hat, würde er doch bei der Entwicklung seiner Anwendungslogik rasch erkennen, dass dieser Ansatz nicht praktikabel ist. Angenommen, bei der zu entwickelnden Anwendung handelt es sich um eine Videoplattform, auf der Benutzer unterschiedliche Video-Clips bewerten und kommentieren können. Hat ein Video etwa Kommentare von zehn unterschiedlichen Benutzern gesammelt, wären zum Anzeigen der Kommentare bis zu zehn Zugriffe auf die Graph API notwendig, um die Namen der kommentierenden Personen zu ermitteln – der Aufruf eines Videos würde damit allein schon für die Zugriffe auf die API etwa zehn Sekunden benötigen. Selbst wenn die in Abschnitt 3.1.4, »Selektion in der Graph API«, vorgestellte Methode zum gebündelten Abfragen mehrerer Objekte in einem Request verwendet werden würde, wären die entstehenden Verzögerungen immer noch zu groß, um ein flüssiges Laden der Videoseiten zu ermöglichen.

Zugriffe auf die Graph API werden – zumindest aus europäischer Sicht – von Facebooks Servern in durchschnittlich einer Sekunde Laufzeit beantwortet. Addiert man dazu die Laufzeit der eigenen Anwendungslogik (also Ihr eigener PHP-Code, etwaige Datenbankzugriffe etc.) und die Laufzeit des HTTP-Requests zwischen Client und Server, ist damit die Antwortzeit von ein bis zwei Sekunden, die man zu Recht heute von modernen Webapplikationen erwarten darf, kaum noch zu erzielen (und dabei ist von der Ladezeit von verknüpften CSS-, JavaScript- oder Bilddateien noch gar nicht die Rede).

Beispiel: Benutzerdatenbank mit Caching

Entsetzt von der schlechten Performance seiner Anwendung, entschließt sich unser hypothetischer Entwickler zur Implementierung eines lokalen Caching-Mechanismus. Er erweitert die Datenbanktabelle `users` dabei folgendermaßen:

```
mysql> create table users (
  id bigint(20) not null primary key,
  created datetime default null,
  modified datetime default null,
  first_name varchar(255) default null,
  last_name varchar(255) default null,
  gender varchar(32) default null
);
```

Die Idee: Beim Anlegen eines Benutzerobjekts werden weitere Daten wie der Vor- und Nachname und das Geschlecht des Benutzers über einen Zugriff auf die Graph API ermittelt und in der Tabelle abgelegt (`https://graph.facebook.com/BenutzerID`). Beim Anzeigen eines Videos mitsamt seiner Kommentare können die Namen der kommentierenden Personen nun direkt aus der lokalen Datenbank abgefragt werden, im Normalfall ist also kein langsamer Zugriff auf die Graph API mehr notwendig.

Einen kleinen Haken hat die Sache jedoch: Ändert eine Person zu einem späteren Zeitpunkt z. B. ihren Namen oder ihre E-Mail-Adresse, wird die Anwendung davon keine Notiz nehmen, sondern unverändert die alten Daten darstellen. Während dies vielleicht bei mancher Marketing-Kampagne, die nur über wenige Wochen laufen soll, ein akzeptabler Nachteil wäre, sollten Entwickler sich doch grundsätzlich überlegen, wie Daten aus der Graph API regelmäßig aktualisiert werden können.

Die Lösung lässt sich bereits am Datenbankfeld `modified` erahnen: Dieses Feld wird genutzt, um jenen Zeitstempel zu speichern, an dem die Daten der Graph API zuletzt aktualisiert wurden. Bei jedem Zugriff auf das Benutzerobjekt wird geprüft, ob der Zeitpunkt `modified` länger als z. B. 24 Stunden zurückliegt. Ist dies der Fall, werden die Daten über die Graph API aktualisiert, und es wird dafür gesorgt, dass `modified` auf den aktuellen Zeitpunkt gesetzt wird. Die im folgenden Beispiel ausformulierte Funktion `getUser()` kapselt diese Funktionalität in eine leicht wiederverwendbare Methode:

```
<?
// Herstellung der Verbindung zur Datenbank
mysql_connect("localhost", "benutzer", "passwort");
mysql_select_db("datenbankname");

function createUser($id) {
   $url = "https://graph.facebook.com/".$id;
   $user = json_decode(curl($url));
```

```
   mysql_query("insert into users set ".
               "id=".$id.",".
               "first_name='".$user->first_name."',".
               "last_name='".$user->last_name."',".
               "gender='".$user->gender."',".
               "created=now(),".
               "modified=now()");
   return $user;
}

function updateUser($id) {
   $url = "https://graph.facebook.com/".$id;
   $user = json_decode(curl($url));
   mysql_query("insert into users set ".
               "first_name='".$user->first_name."',".
               "last_name='".$user->last_name."',".
               "gender='".$user->gender."',".
               "modified=now()".
               "where id=".$id);
   return $user;
}

function getUser($id) {
   // Einlesen des Benutzerobjekts aus der lokalen DB
   $res = mysql_query("select * from users where id=".$id);
   if ($user = mysql_fetch_object($res)) {
      // Benutzer älter als 24h - Update per Graph API
      if (date("U") - strtotime($user->modified) > 24*60*60) {
         return updateUser($id);
      } else {
         return $user;
       }
   } else {
      // Benutzer nicht in DB gefunden - Anlegen per Graph API
      return createUser($id);
   }
}
print_r(getUser(609190863));
?>
```

Listing 3.9 Benutzerverwaltung mit Caching der Zugriffe auf die Graph API

3.4 Batch-Zugriff auf die Graph API

Wie im vorangegangenen Abschnitt festgestellt, stellen Zugriffe auf die Graph API einen performancekritischen Vorgang dar, den Sie am besten so oft wie möglich durch Caching vermeiden sollten. In Abschnitt 3.1.4, »Selektion in der Graph API«, haben Sie erfahren, dass durch einfache Verknüpfung von numerischen IDs oder Benutzernamen mehrere Objekte des sozialen Graphen auf einen Streich abgefragt werden können. Diese Technik hat ihre Grenzen – so kann sie etwa nicht dazu dienen, Objektverknüpfungen von mehreren Objekten (z. B. die Kommentare zu mehreren Fotos) gleichzeitig abzufragen. Auch Zugriffe mit sich unterscheidenden Parametern (etwa für Paging, Feldselektion etc.) oder gar Schreib- oder Löschvorgänge können so nicht performant abgebildet werden.

Glücklicherweise bietet die Graph API für solche komplexeren Zugriffe eine flexible Batch-API, die es ermöglicht, beliebige Arten von Zugriffen in einem einzigen, performanceschonenden HTTP-Request zu bündeln.

Wichtig: Obwohl Batch-Zugriffe maximale Flexibilität in der Art der gebündelten Zugriffe bieten, sind sie doch auf *maximal 50 API-Zugriffe* pro Batch-Request begrenzt!

Wichtiger Link:

- *https://developers.facebook.com/docs/reference/api/batch/* – die offizielle Dokumentation zu Batch-Zugriffen auf die Graph API

3.4.1 Einfache Batch-Zugriffe

Batch-Zugriffe werden realisiert, indem alle gewünschten Aufrufe der Graph API in einem JSON-Array mit der verwendeten Methode (Parameter `method`, GET, POST oder DELETE) und dem gewünschten Endpunkt der Graph API (Parameter `relative_url`) spezifiziert und mittels POST-Request im Parameter `batch` an `https://graph.facebook.com` gesendet werden.

Wollen Sie z. B. die Abfrage von zwei Benutzerobjekten in einem Batch-Zugriff bündeln, können Sie dies mittels `curl` auf der Kommandozeile folgendermaßen tun:

```
curl \
   -F 'access_token=...' \
   -F 'batch=[ \
         {"method": "GET", \
          "relative_url": "609190863?fields=id,first_name"}, \
         {"method": "GET", \
          "relative_url": "633616172?fields=id,first_name"} ]'\
https://graph.facebook.com
```

Analog zur Anfrage wird auch das Ergebnis von Facebook in ein JSON-Array verpackt, das pro gestellter API-Anfrage den HTTP-Ergebnis-Code `code`, die zurückgelieferten HTTP-Header `headers` und das eigentliche Resultat im Element `body` enthält. Für das obige Beispiel sieht das Ergebnis in JSON-Notation also so aus:

```
[
   { "code" : 200,
     "headers": [
        { "name" : "Content-Type",
          "value" : "text\/javascript; charset=UTF-8"
        } ],
     "body" : "{\"id\":\"609190863\",\"first_name\":\"Michael\"}"
   },
   { "code" : 200,
     "headers" : [
        { "name" : "Content-Type",
          "value" : "text\/javascript; charset=UTF-8"
        } ],
      "body" : "{\"id\":\"633616172\",\"first_name\":\"Billie\"}"
   }
]
```

Listing 3.10 Ergebnis-Array eines Batch-Zugriffs auf die Graph API

Es lohnt sich, das Ergebnis genauer unter die Lupe zu nehmen:

- Neben dem HTTP-Header `Content-Type` werden meist noch zahlreiche weitere Header (z. B. `Cache-Control`, `ETag`, `Connection` ...) ausgeliefert, die hier aus Gründen der Übersichtlichkeit nicht angeführt wurden.
- Die beiden eigentlichen Resultate der API-Zugriffe finden sich in den jeweiligen `body`-Elementen. Die Ergebnisse sind wiederum als *String-codiertes JSON-Array* codiert – dies muss bei der weiteren Verarbeitung beachtet werden!

Übrigens: Batch-Zugriffe können leider noch nicht über den Graph API Explorer getestet werden! Zum Testen bietet sich daher entweder der Aufruf von `curl` auf der Kommandozeile oder der in Abschnitt 3.4.3, »Kombination von Schreib- und Lesezugriffen«, vorgestellte PHP-Code an.

Wichtig: Bei der Nutzung von Batch-Zugriffen muss *immer ein gültiges Access Token angegeben werden,* auch wenn die gebündelten API-Zugriffe wie im obigen Beispiel selbst kein Access Token erfordern würden!

3.4.2 Verwendung von unterschiedlichen Access Tokens

Apropos Access Token: In manchen Fällen kann es erforderlich sein, die gebündelten API-Zugriffe mit jeweils unterschiedlichen Access Tokens zu signieren. Ein Beispiel könnte eine Anwendung sein, die den Benutzern einen täglichen Newsletter mit interessanten Wall-Postings ihrer Freunde zustellen möchte. Um täglich die aktuellsten Freundschaftsbeziehungen verwenden zu können, sollte also vor dem Versenden des Newsletters die Liste der Freunde Ihrer Benutzer aktualisiert werden. Dazu muss je Benutzer eine Anfrage an `https://graph.facebook.com/BenutzerID/friends` gestellt werden, wozu allerdings von jedem Benutzer ein individuelles Access Token verwendet werden muss (die Freundschaftsbeziehungen im sozialen Graphen sind standardmäßig ohne Access Token nicht öffentlich abzufragen). Der entsprechende `curl`-Aufruf müsste so aussehen:

```
curl \
   -F 'access_token=...' \
   -F 'batch=[ \
         {"method": "GET", \
          "relative_url": "609190863/friends?access_token=A.."}, \
         {"method": "GET", \
          "relative_url": "633616172/friends?access_token=B.."}]'\
https://graph.facebook.com
```

Beachten Sie folgende Details bei der Verwendung unterschiedlicher Access Tokens:

- Wie im Beispiel zu sehen ist, wird ein allgemeines, sogenanntes *Top-Level Access Token*, definiert, darüber hinaus je ein weiteres Token pro API-Zugriff im GET-Parameter `?access_token=`.
- Das Top-Level Access Token dient als Fallback, wenn einzelne API-Zugriffe ohne eigenes Access Token erfolgen.
- Auch wenn alle API-Zugriffe über ein individuelles Token verfügen, verlangt Facebook dennoch auch die Angabe des Top-Level Access Tokens.

3.4.3 Kombination von Schreib- und Lesezugriffen

Wie eingangs erwähnt, können Batch-Zugriffe auf die Graph API auch unterschiedliche Aufrufe zum Lesen und Schreiben von Objekten im sozialen Graphen enthalten. Dazu muss unter `method` lediglich die passende HTTP-Methode (POST oder DELETE) angegeben werden. Bei POST-Zugriffen müssen darüber hinaus im Parameter `body` all jene POST-Parameter übergeben werden, die zum Veröffentlichen des jeweiligen Objekttyps notwendig sind. Die Parameter werden dabei wie bei einer üblichen HTTP-GET-Query URL-codiert (`parameter1=wert&parameter2=wert&...`). Ein mit `curl`

realisiertes Beispiel, das zuerst ein Wall-Posting veröffentlicht, und anschließend das neueste, eben gepostete Element von der Pinnwand des Benutzers ausliest, sieht so aus:


```
curl \
   -F 'access_token=...' \
   -F 'batch=[ \
         {"method": "POST", \
          "relative_url": "me/feed", \
          "body": "message=Mein Wall-Posting&link=http://..."} \
         {"method": "GET", \
          "relative_url": " me/feed?limit=1"} ]'\
https://graph.facebook.com
```

Das Beispiel zeigt sehr schön die unterschiedlichen Zugriffsmethoden der Graph API: Beide Zugriffe beziehen sich auf den gleichen API-Endpunkt (`relative_url`), jedoch wird einmal mittels POST ein Schreibzugriff (Wall-Posting veröffentlichen) und einmal mittels GET ein Lesezugriff (Wall-Postings auslesen) realisiert. Beim Veröffentlichen wird dazu im Parameter `body` übergeben, welchen Textinhalt (`message`) und welchen Link (`link`) das Wall-Posting enthalten soll.

Der folgende PHP-Code zeigt, wie der eben mit `curl` demonstrierte Batch-Zugriff realisiert werden kann:

```
<?
$batch = array(
   array( "method" => "POST",
          "relative_url" => "me/feed",
          "body" => "message=Mein Wall-Posting&http://..."),
   array( "method" => "GET",
          "relative_url" => "me/feed?limit=1"),
);
$url = "https://graph.facebook.com";
$params = array(
   "access_token" => "...",
   "batch" => json_encode($batch)
);
print_r(json_decode(curl($url, $params)));
?>
```

Listing 3.11 Batch-Zugriff auf die Graph API mittels PHP

Der PHP-Code ist analog zum Kommandozeilen-Aufruf von `curl` aufgebaut:

- Im Array `$batch` werden die gewünschten API-Aufrufe mit Methode (`method`), API-Endpunkt (`relative_url`) und gegebenenfalls zu übergebenden POST-Parametern (`body`) als Query-String aufgebaut.
- Der eigentliche Batch-Zugriff erfolgt als normaler POST-Aufruf der Graph API mit der bereits bekannten Helferfunktion `curl()` – dieser werden das Access Token und das mit `json_encode()` JSON-codierte Array `$batch` übergeben.

Das Ergebnis präsentiert sich in folgendem PHP-Array:

```
Array
(
   [0] => stdClass Object
      (
         [code] => 200
         [headers] => Array
            (
               [0] => stdClass Object
                  (
                     [name] => Content-Type
                     [value] => text/javascript; charset=UTF-8
                  )
            )
         [body] => {"id":"609190863_10150498217110864"}
      )
   [1] => stdClass Object
      (
         [code] => 200
         [headers] => Array
            (
               [0] => stdClass Object
                  (
                     [name] => Content-Type
                     [value] => text/javascript; charset=UTF-8
                  )
               )
         [body] => {"data":[{"id":"609190863_278579792178519","from":{"name":
            "Michael Kamleitner","id":"609190863"},"message":
            "Mein Wall-Posting","link":"http:\/\/...","created_time":
            "2011-11-26T14:32:14+0000","updated_time":"2011-11-26T16:41:
            09+0000"...
      )
)
```

Listing 3.12 Ergebnis eines Batch-Zugriffs auf die Graph API

3.4.4 Abhängigkeiten zwischen API-Zugriffen und Verschachtelung

Das in Abschnitt 3.4.3, »Kombination von Schreib- und Lesezugriffen«, dargestellte Beispiel zum Veröffentlichen und Auslesen eines Wall-Postings hat einen kleinen, aber folgenschweren Haken: Werden mehrere API-Zugriffe mittels Batch gebündelt, gibt es keine Garantie dafür, in welcher Reihenfolge die einzelnen Zugriffe ausgeführt werden – mitunter wird Facebook sie sogar parallel durchführen. In unserem Beispiel könnte es also passieren, dass der zweite Aufruf zum Lesen der Wall schon vor dem Veröffentlich des Postings ausgeführt wird, das Ergebnis damit also nicht das neu veröffentliche Posting enthält. Um dieses Problem zu umgehen, unterstützt die Graph API die sogenannte *JSONPath-Syntax*, die es ermöglicht, innerhalb einer JSON-Datenstruktur bestimmte JSON-Objekte zu referenzieren. Sehen Sie sich z. B. folgenden Aufruf der Graph API mittels `curl` an:

```
curl \
   -F 'access_token=...' \
   -F 'batch=[{ "method": "GET", \
                "name" : "get-friends", \
                "relative_url": "me/friends?limit=5", \
              }, \
              { "method": "GET", \
                "relative_url": "?ids={result=get-friends:$.data.*.id}" \
              }]' \
      https://graph.facebook.com/
```

Listing 3.13 Batch-Zugriff auf die Graph API mit JSONPath-Abhängigkeiten

Der erste API-Zugriff in diesem Batch-Request liest über den Endpunkt `/me/friends` fünf Freunde des aktuellen Benutzers ein. Dank des Attributs `name` kann auf das Ergebnis dieses Aufrufs fortan über den Namen `get-friends` zugegriffen werden. Im zweiten API-Zugriff werden die Benutzer-IDs aus dem Ergebnis des ersten Zugriffs an den Endpunkt `/?ids=...` übergeben, um die detaillierten User-Objekte der Freunde zu ermitteln. Der JSONPath-Ausdruck `{result=get-friends:$.data.*.id}` verweist dabei auf das `get-friends` benannte Ergebnisobjekt und extrahiert alle `id`-Felder aus dem Array `data`. Das Backend von Facebook erkennt die Abhängigkeit zwischen den beiden Zugriffen und sorgt dafür, dass die beiden Requests nacheinander ausgeführt werden.

Wichtiger Link:

- *http://code.google.com/p/jsonpath/* – Dokumentation zu JSONPath

3.5 Echtzeit-Zugriff auf die Graph API

Eine weitere Besonderheit der Graph API stellt die Möglichkeit dar, sich als Anwendungsentwickler über Datenänderungen von bestimmten Objekttypen im sozialen Graphen in Echtzeit benachrichtigen zu lassen. Anwendungen müssen bestimmte Felder (z. B. den Namen eines Benutzers) damit nicht mehr laufend und selbstständig mittels Zugriff auf die Graph API aktualisieren, sondern erhalten von Facebook automatisch eine Benachrichtigung, wenn sich diese Objekte ändern.

Um Echtzeit-Benachrichtigungen zu erhalten, muss eine Anwendung einerseits in einem »Abonnement« festlegen, welche Felder und Verknüpfungen überwacht werden sollen. Andererseits muss die Anwendung einen Endpunkt, also eine URL, bereitstellen, an die Facebook die Updates melden soll. Bei der Umsetzung von Echtzeit-Benachrichtigungen setzt Facebook dabei weitestgehend auf ein Protokoll mit dem eigenwilligen Namen *PubSubHubBub*.

Wichtig: Echtzeit-Updates werden nur für jene Benutzer oder Seiten unterstützt, die die abonnierende Facebook-Anwendung selbst installiert haben. Dabei macht es aber keinen Unterschied, ob die Installation vor oder nach Einrichten des Abonnements erfolgte. Dies verdeutlicht noch einmal, dass Echtzeit-Updates unabhängig von den Benutzer-Access-Tokens funktionieren und automatisch für jeden autorisierten Benutzer gültig sind.

Die Graph API unterstützt Echtzeit-Updates derzeit für die folgenden Objekttypen:

Objekttyp	Unterstützte Felder
`user`	alle Felder außer `verified`, verknüpfte Objekte vom Typ `feed`, `friends`, `activities`, `interests`, `music`, `books`, `movies`, `television`, `likes`, `checkins`
`page`	keine Felder, ausschließlich verknüpfte Objekte vom Typ `feed`, `picture`, `tagged`, `checkins`
`permission`	alle Zugriffsrechte, die der Anwendung gewährt oder entzogen werden

Tabelle 3.85 Unterstützte Objekttypen für Echtzeit-Updates

Wichtige Links:

- *http://developers.facebook.com/docs/reference/api/realtime/* – die offizielle Dokumentation zu Echtzeit-Zugriffen auf die Graph API
- *http://code.google.com/p/pubsubhubbub/* – die Spezifikation des PubSubHubBub-Protokolls

3.5.1 Verwaltung von Echtzeit-Abonnements

Die Verwaltung von Echtzeit-Abonnements wird über den API-Endpunkt `https://graph.facebook.com/ApplikationsID/subscriptions` durchgeführt. Zugriffe auf diese URL müssen immer mit einem Anwendungs-Access-Token signiert werden, das bezogen werden kann, wie in Abschnitt 2.7.1, »Access Tokens für Anwendungen«, dargestellt.

Hinzufügen oder Bearbeiten von Abonnements

Abonnements werden über einen POST-Zugriff auf den `/subscriptions`-Endpunkt erstellt oder bearbeitet. Da jede Anwendung pro Objekttyp nur ein Abonnement besitzen kann, wird jeder weitere POST-Zugriff automatisch von Facebook als Update des vorhandenen Abonnements interpretiert. Folgende Parameter müssen dabei übergeben werden:

Parameter	Beschreibung	Typ	Pflichtfeld
`object`	Objekttyp, für den das Abonnement erstellt werden soll. Derzeit nur `user` oder `permission`	`string`	ja
`fields`	kommaseparierte Liste von Feldern oder Verknüpfungen, deren Änderungen abonniert werden sollen	`string`	ja
`callback_url`	URL, an die Facebook Benachrichtigung bei Änderung der abonnierten Objekte senden soll	`url`	ja
`verify_token`	Prüf-Token für den Abonnementvorgang	`string`	ja

Tabelle 3.86 Felder zum Hinzufügen oder Bearbeiten von Echtzeit-Updates

Beim Hinzufügen oder Bearbeiten von Abonnements sind folgende Hinweise nützlich:

- Der in `object` angegebene Bezeichner gibt an, für welche Objekttypen Änderungen abonniert werden sollen. *Achtung:* Sowohl für Benutzer- als auch für Seitenobjekte muss hier der Wer `user` übergeben werden – diese Objekttypen werden also immer gemeinsam abonniert!
- Die in `fields` angegebenen abonnierten Felder müssen abhängig vom Objekttyp gewählt werden.
- `callback_url` ist jene URL, die Facebook künftig in Echtzeit von Änderungen an den abonnierten Objekttypen benachrichtigen soll. Auf dieser URL müssen Anwendungsentwickler ein Skript bereitstellen, das einerseits den Abonnement-

vorgang bestätigt und andererseits die Benachrichtigungen verarbeitet – doch dazu später mehr.

- Der `verify_token` wird dazu benutzt, im Abonnementvorgang sicherzustellen, dass eine Abonnementanfrage tatsächlich von den Servern von Facebook angenommen wurde.

Auslesen aller aktiven Abonnements

Das Auslesen aller aktiven Abonnements der Anwendung ist mit einem GET-Zugriff auf den `/subscriptions`-Endpunkt möglich. Das zurückgelieferte Array enthält für jedes Abonnement die Felder `object`, `callback_url` und `fields`, deren Bedeutung analog zur vorangegangenen Tabelle ist. Das Feld `active` enthält vorerst immer `true`, da Abonnements derzeit nur gelöscht, nicht aber deaktiviert werden können.

Das folgende Code-Beispiel zeigt zuerst, wie ein Anwendungs-Access-Token bezogen werden kann. Anschließend wird das Token zum Erstellen eines Abonnements auf das Feld `name` von Benutzer- und Seitenobjekten verwendet. Zur abschließenden Kontrolle werden alle existierenden Abonnements der Anwendung ausgegeben:

```
<?
include_once("tools.php");
define('APP_ID', '214728715257742'); '
define('APP_SECRET', '***************');
define('SITE_URL', 'http://apps.mycompany.at');

// Beziehen des Anwendungs-Access-Tokens
$access_token = get_app_accesstoken(APP_ID, APP_SECRET);

// Hinzufügen des Abonnements
$url = "https://graph.facebook.com/".APP_ID."/subscriptions";
curl($url,array("object" => "user",
   "fields" => "name",
   "access_token" => $access_token,
   "verify_token" => "thisisoursecret",
   "callback_url" => SITE_URL.'/ callback-realtime.php'));

// Auslesen aller existierender Abonnements
$url = "https://graph.facebook.com/".
       APP_ID."/subscriptions?access_token=".$access_token;
$subscriptions = json_decode(curl($url));
print "Alle Abonnements:\n";
print_r ($subscriptions);
?>
```

Listing 3.14 Hinzufügen und Auslesen von Echzeit-Abonnements

Und so sieht die Array-Darstellung des erfolgreich hinzugefügten Abonnements aus:

```
Alle Abonnements:
stdClass Object
(
   [data] => Array
      (
         [0] => stdClass Object
         (
            [object] => user
            [callback_url] =>
              http://apps.mycompany.com/callback-realtime.php
            [fields] => Array
               (
                 [0] => name
               )
            [active] => 1
         )
      )
)
```

Tabelle 3.87 Rückgabewert des hinzugefügten Echtzeit-Abonnements

Löschen aller aktiven Abonnements

Wie bereits erwähnt, werden Änderungen an Abonnements durchgeführt, indem der API-Endpunkt einfach erneut über einen POST-Zugriff für denselben Objekttyp aufgerufen wird. Abgesehen davon, unterstützt die Echtzeit-API lediglich eine weitere Operation, nämlich das Löschen aller aktiven Abonnements einer Anwendung auf einen Schlag. Dazu genügt ein DELETE-Zugriff auf den Endpunkt `https://graph.facebook.com/ApplikationsID/subscriptions`, wie der folgende Beispiel-Code zeigt:

```
<?
include_once("tools.php");
define('APP_ID', '214728715257742');
define('APP_SECRET', '*************');
define('SITE_URL', 'http://apps.mycompany.com');

$access_token = get_app_accesstoken(APP_ID, APP_SECRET);

// Löschen aller aktiven Abonnements
$url = "https://graph.facebook.com/".
       APP_ID."/subscriptions?access_token=".$access_token;
curl($url, false, "DELETE");
```

```
// Kontrolle: Auslesen aller aktiven Abonnements
$url = "https://graph.facebook.com/".
       APP_ID."/subscriptions?access_token=".$access_token;
$subscriptions = json_decode(curl($url));

print "Alle Abonnements:\n\n";
print_r ($subscriptions);
?>
```

Listing 3.15 Löschen aller aktiven Echtzeit-Abonnements

Der auf das Löschen folgende, kontrollierende GET-Zugriff zum Auslesen der bestehenden Abonnements gibt als Ergebnis ein leeres Array zurück.

3.5.2 Entgegennehmen von Echzeit-Updates

Wie bereits aus den Parametern zum Hinzufügen oder Bearbeiten von Abonnements ersichtlich war, müssen Anwendungsentwickler eine Callback-URL bereitstellen, an die die Backend-Server von Facebook Änderungen an abonnierten Objekttypen melden können. Diese Callback-URL ist gleichzeitig dafür verantwortlich, ein hinzugefügtes Abonnement zu bestätigen. Dabei muss folgender Ablauf eingehalten werden:

1. Die Anwendung fügt ein Echtzeit-Abonnement hinzu, indem sie einen POST-Zugriff auf `https://graph.facebook.com/AppID/subscriptions` absetzt. Dabei werden auch die `callback_url` und das `verify_token` an Facebook übergeben.
2. Facebook prüft das Abonnement unmittelbar danach, indem die Callback-URL vom Facebook-Backend aufgerufen wird. Dabei werden die Parameter `hub_mode`, `hub_challenge` und `hub_verify_token` übergeben.
3. Die Anwendung nimmt den Aufruf auf `callback_url` entgegen und kontrolliert als Sicherheitsmaßnahme, ob der Parameter `hub_verify_token` jenem String entspricht, der beim POST-Zugriff zum Hinzufügen des Abonnements an Facebook übergeben wurde.
4. Stimmt `hub_verify_token` mit `verify_token` überein, ist gewährleistet, dass tatsächlich ein Backend-Server von Facebook das Abonnement bestätigen möchte. Das Skript unter `callback_url` antwortet in diesem Fall durch Ausgabe des Inhalts vom Parameter `hub.challenge` und bestätigt damit das Abonnement.
5. Von nun an setzt Facebook jedes Mal, wenn ein abonnierter Objekttyp verändert wird, einen POST-Zugriff auf `callback_url` ab. Die Anwendung muss diesen Zugriff entgegennehmen und verarbeiten.

Die POST-Zugriffe auf `callback_url` erfolgen mit dem Content-Type `application/json` und enthalten im Body ein JSON-codiertes Objekt, das Informationen über die geänderten Objekte enthält. In PHP muss der Body-Inhalt des Zugriffs mit `file_get_contents('php://input')` von der Standardeingabe gelesen und decodiert werden.

Das folgende Code-Beispiel zeigt die Behandlung der Abonnementbestätigung und der eingehenden Benachrichtigungen und muss analog zu den vorangegangenen Beispielen am Webserver in der Datei `callback-realtime.php` abgelegt werden:

```
<?
include_once("tools.php");
define('APP_ID', '214728715257742');
define('APP_SECRET', ******************');
define('SITE_URL', 'http://apps.diesocialisten.at');

if (@$_REQUEST["hub_mode"] == "subscribe" &&
    @$_REQUEST["hub_verify_token"] == "thisisoursecret") {

   // Da die Parameter hub_... gesetzt sind, handelt es sich
   // um einen Zugriff zur Abobestätigung. Dazu antwortet
   // das Skript mit dem Inhalt von hub_challenge:

   print @$_REQUEST["hub_challenge"];
   exit;
} else {
   // Andernfalls handelt es sich beim Aufruf um eine
   // Benachrichtigung über eine Objektänderung.

   // Einlesen des Request-Bodys von der Standardeingabe
   $buffer = (file_get_contents('php://input'));
   $buffer = json_decode($buffer);

   // Logging in Textdatei
   $log = fopen("realtime.log","a");
   fputs($log, print_r($buffer,true));
   fclose($log);

   // An dieser Stelle sollten die Echtzeit-Updates weiter-
   // verarbeitet werden ...
}
?>
```

Listing 3.16 Callback-Skript zur Bestätigung von Abonnements und Entgegennahme von Echtzeit-Updates

Wichtig: Nicht zuletzt zur Wahrung der Privatsphäre der Benutzer enthalten die Echtzeit-Updates lediglich einen Hinweis darauf, *welche Felder* geändert wurden, nicht aber den *Inhalt der Felder* selbst. Das bedeutet, dass die abonnierende Applikation nach Erhalt der Benachrichtigung eigenständig für ein Update der Daten sorgen muss!

Da das Callback-Skript nur von Facebook aufgerufen wird und somit schwierig im Browser oder auf der Kommandozeile zu testen ist, empfiehlt es sich, während der Entwicklung das decodierte JSON-Objekt von der Standardeingabe in eine Textdatei zu schreiben. Eine Benachrichtigung über die Änderung des Namens sieht dabei decodiert etwa so aus:

```
{
   "object": "user",
   "entry":
   [
      {
         "uid": ...,
         "changed_fields":
         [
            "name"
         ],
         "time": 232323
      },
      {
         "uid": ...,
         "changed_fields":
         [
            "name"
        ],
        "time": 232325
     }
   ]
}
```

Listing 3.17 JSON-Array zur Benachrichtigung über ein Echtzeit-Update

Die Bedeutung der einzelnen Elemente sei kurz erklärt:

- Facebook bündelt Echtzeit-Updates, wenn möglich, und sendet sie in einem ungefähren Abstand von fünf Sekunden an den Anwendungsserver. Das oben dargestellte Beispiel enthält demnach Benachrichtigungen über zwei unterschiedliche User-Objekte, bei denen der Name des Benutzers geändert wurde.

- Im Element `changed_fields` wird angegeben, welche Felder des Objekts sich geändert haben.
- `time` enthält den Zeitstempel der Änderung als UNIX-Zeitstempel.

Wie vorhin bemerkt, liefert die Echtzeit-Benachrichtigung lediglich den Hinweis, dass der Name geändert wurde, nicht aber den neuen Namen selbst. Dieser muss anschließend von der Anwendung mit einem üblichen Zugriff auf die Graph API ermittelt werden.

Kapitel 4
Das JavaScript SDK

Egal, ob Canvas-Anwendung, Tab/Reiter auf einem Seitenprofil oder externe Webseite – fast immer werden Facebook-Anwendungen im Browser mittels JavaScript in die Plattform integriert. Das JavaScript SDK ermöglicht diese Integration und stellt Methoden zum Autorisieren von Anwendungen, zum Verfassen von Wall-Postings oder zum Einladen von Freunden bereit. Dieses Kapitel gibt Ihnen einen vollständigen Überblick über den Einsatz der SDK-Methoden.

Mit dem Facebook JavaScript SDK können Entwickler serverseitige Plattform-Schnittstellen wie die alte REST-API und die moderne Graph API clientseitig, also aus dem Browser des Benutzers, ansprechen. Darüber hinaus stellt Facebook über das SDK zahlreiche Dialoge zur Verfügung, die Facebook-Funktionalitäten in Plattform-Anwendungen bereitstellen – etwa das Verfassen von Wall-Postings oder das Versenden von Anwendungsanfragen an Freunde. Auch die Darstellung der in Kapitel 6, »Social Plugins«, beschriebenen Social Plugins wird normalerweise vom JavaScript SDK übernommen.

4.1 Laden des JavaScript SDKs

Das JavaScript SDK wird üblicherweise auf jeder HTML-Seite der Anwendung geladen. Dabei macht es keinen Unterschied, ob eine Canvas-Anwendung auf Facebook, eine Tab/Reiter-Anwendung oder eine Anwendung auf einer externen Website entwickelt wird. Das Laden des SDKs sollte asynchron durchgeführt werden – damit verzögert der Ladevorgang nicht den Aufbau der eigentlichen Webseite. Der folgende Beispiel-Code lädt das SDK:

```
<div id="fb-root"></div>
<script>
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
                   // Anwendungs-ID
      channelUrl : '//apps.mycompany.com/channel.php',
```

```
                  // Channel-Datei
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
    // Weitere Initialisierungen hier vornehmen ...
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
</script>
```

Listing 4.1 Asynchrones Laden des Facebook JavaScript SDKs

Beim Laden des SDKs ist Folgendes zu beachten:

- Mit `window.fbAsyncInit` wird eine Funktion definiert, die erst nach dem vollständigen Laden des SDKs automatisch aufgerufen wird. Zu diesem Zeitpunkt steht das SDK in Form des Objekts `FB` zur Verfügung. Individueller Initialisierungscode, der ebenfalls auf der Verfügbarkeit des SDKs aufbaut, sollte innerhalb dieser Funktion nach dem Initialisieren des SDKs mittels `FB.init()` eingefügt werden.
- Weiter unten im Code wird das asynchrone Laden des SDKs von `connect.facebook.net/en_US/all.js` initiiert.
- Der Parameter `channelUrl` verweist auf eine Hilfsdatei, die die reibungslose Kommunikation zwischen der Domain der Anwendung und Facebook.com ermöglicht. Die Hilfsdatei muss unter jener Domain abgerufen werden, die auch im Facebook Developer als SITE URL angegeben wurde.
- Der Parameter `status` gibt ab, ob das SDK den Login-Status des aktuellen Benutzers prüfen soll. Dies ist praktisch, um sofort erkennen zu können, ob der aktuelle Benutzer die Facebook-Anwendung bereits autorisiert hat.
- Der Parameter `cookie` gibt an, ob der serverseitige Zugriff auf die Session mittels Cookies aktiviert werden soll.
- Der Parameter `oauth` gibt an, dass die Authentifizierung mit OAuth 2.0 erfolgen soll. Seit Oktober 2011 ist dies vorgeschrieben.
- Der Parameter `xfbml` gibt an, ob das SDK XFBML-Tags, wie etwa die Social Plugins, automatisch parsen und die entsprechenden UI-Elemente ersetzen soll.

- Das Laden des SDKs muss immer in Verbindung mit einer Anwendungs-ID erfolgen – diese wird im Parameter `appId` festgelegt.

Hinweis: Da der HTML-Quellcode Ihrer Webseiten für jeden Benutzer einsehbar ist, kommt das Laden des SDKs immer ohne Bereitstellung des App Secrets aus!

4.1.1 Hello World mit dem Facebook JavaScript SDK

In einem ersten einfachen Beispiel soll nun das SDK geladen und zum Parsen eines XFBML-Elements genutzt werden. XFBML-Elemente sind Erweiterungen der HTML-Syntax, die Facebook-spezifische Funktionen bereitstellen. Der folgende Code wird im Root-Verzeichnis des Webservers in der Datei *5-jssdk-helloworld.php* abgelegt:

```
<DOCTYPE html>
<html lang="de-de" xmlns=http://www.w3.org/1999/xhtml
      xmlns:fb="http://ogp.me/ns/fb#"> <head>
   <meta charset="utf-8">
   <title>Hello Facebook JS SDK!</title>
</head>
<body>
<h1>Hello <fb:name uid="609190863" useyou="false"/>!</h1>
<div id="fb-root"></div>
<script>
   window.fbAsyncInit = function() {
   FB.init({
      appId      : '214728715257742',
                   // Anwendungs-ID
      channelUrl : '//apps.mycompany.com/channel.php',
                   // Channel-Datei
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
```

```
</script>
</body>
</html>
```

Listing 4.2 Hello World mit dem Facebook JavaScript SDK

Neben dem bereits bekannten Skript zum asynchronen Laden des SDKs enthält das Beispiel lediglich den üblichen HTML-Rumpf. Die einzige auffällige Zeile findet sich hier:

```
<h1>Hello <fb:name uid="609190863" useyou="false"/>!</h1>
```

Das Element `<fb:name>` ist ein sogenanntes *XFBML-Tag*, das zur Laufzeit vom JavaScript SDK durch passende Inhalte – in diesem Fall dem Namen des Benutzers mit der ID `609190863` – ersetzt wird. Dementsprechend sieht die Ausgabe im Browser folgendermaßen aus:

Abbildung 4.1 Ausgabe des JavaScript-basierten Hello-World-Beispiels

Was genau ist hier passiert?

- Die Seite *jssdk-helloworld.php* wird im Browser aufgerufen. Zu diesem Zeitpunkt wird das Tag `<fb:name>` nicht angezeigt, da es vom Browser nicht interpretiert werden kann und daher ignoriert wird.
- Nach dem Laden des Grundgerüsts der Seite wird das asynchrone Laden des JavaScript SDKs initiiert. Wichtig ist dabei, dass der Parameter `xfbml` auf `true` gesetzt wird – nur dann werden XFBML-Tags vom SDK beachtet.
- Ist das SDK vollständig geladen, werden automatisch alle XFBML-Tags des Dokuments gesucht und durch passende Inhalte (hier: »Michael Kamleitner«) ersetzt.

Wenn Sie ein Debugging-Tool wie Firebug oder den in Chrome integrierten Inspector verwenden, können Sie den HTML-Code, wie er sich nach dem Laden der Webseite präsentiert, leicht untersuchen:

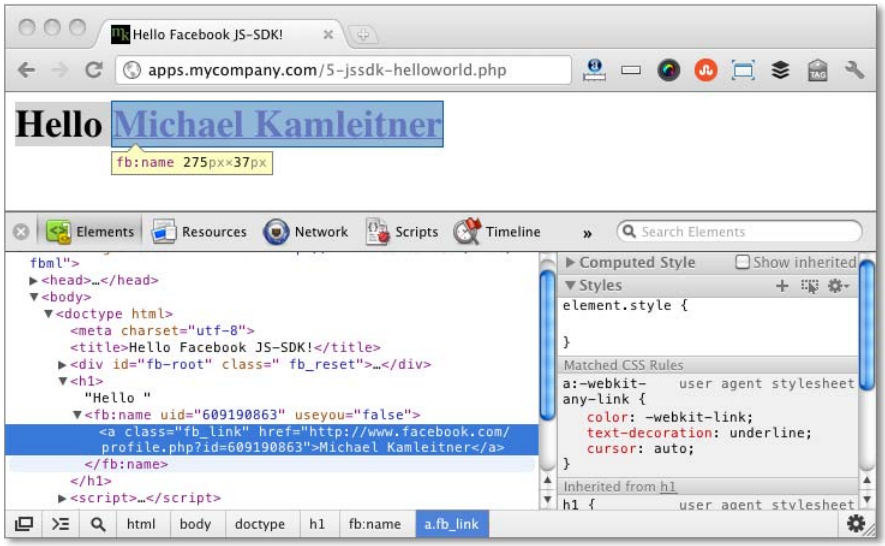


Abbildung 4.2 Quellcode von Hello World nach dem Laden des SDKs

Es zeigt sich, dass das proprietäre XFBML-Tag `<fb:name>` vom SDK mit üblichem HTML-Code ergänzt wurde:

```
<fb:name uid="609190863" useyou="false">
   <a class="fb_link"
      href="http://www.facebook.com/profile.php?id=609190863">
      Michael Kamleitner</a>
</fb:name>
```

Hinweis: Bei der Verwendung von XFBML ist die Erweiterung des XML-Namespaces im `<html>`-Element mit dem Attribut `xmlns:fb="http://ogp.me/ns/fb#"` besonders wichtig. Ohne diese Erweiterung werden XFBML-Elemente in manchen Versionen des Internet Explorers nicht angezeigt.

Tools zum Debugging von JavaScript- und HTML-Code

Die in den folgenden Abschnitten vorgestellten JavaScript-Methoden sind mit den »Bordmitteln« der meisten Browser nur schwierig zu testen und zu debuggen. Viele Methoden liefern JSON-Objekte als Rückgabewert, die z. B. nicht einfach mittels `alert()` ausgegeben werden können.

Um sich das Leben zu erleichtern, empfehle ich Ihnen daher, für die Entwicklung von JavaScript-Code einen Browser mit integriertem JavaScript-Debugger zu verwenden. Dabei kann z. B. Google Chrome zum Einsatz kommen – dieser Browser enthält standardmäßig einen mächtigen HTML- & CSS-Inspector sowie eine JavaScript-Konsole.

Fans des Browsers Firefox können dazu einfach auf die Erweiterung Firebug zurückgreifen.

In beiden Fällen steht eine Konsole zur Verfügung, auf die Entwickler über einfache JavaScript-Befehle auch komplexe JSON-Objekte ausgeben können:

```
console.debug(json_objekt);
```

Wichtige Links:

- *http://getfirebug.com/* – Firebug-Erweiterung für Firefox
- *https://www.google.com/chrome* – Google Chrome Browser

4.1.2 Die Channel-Datei

Besonders wichtig beim Laden des SDKs ist der Parameter `channelUrl`. Mit diesem Parameter wird auf eine Channel-Datei verwiesen, die die reibungslose Kommunikation zwischen der Domain Ihrer Anwendung (`apps.mycompany.com`) und Facebook (`facebook.com`) in allen Browsern ermöglicht. Obwohl der Parameter optional ist, empfiehlt sich die korrekte Verwendung, da ansonsten Social Plugins in manchen Browsern gar nicht funktionieren bzw. längere Ladezeiten entstehen können.

Die Channel-Datei enthält lediglich eine Zeile, die in einem `<script>`-Element auf das JavaScript SDK verweist:

```
<script src="//connect.facebook.net/en_US/all.js"></script>
```

Wichtig ist dabei, dass sowohl das Protokoll (HTTP oder HTTPS) als auch die verwendete Lokalisierung mit jener URL übereinstimmen, die beim asynchronen Laden des SDKs verwendet wurde. Darüber hinaus muss die Channel-Datei unter derselben URL abgerufen werden, die in den Einstellungen der Applikation unter Site URL angegeben wurde.

Da die Channel-Datei bei jedem Initialisieren des SDKs aufgerufen wird, ist es wichtig, die Datei so lange wie möglich clientseitig zu cachen. Legen Sie die Channel-Datei am Webserver unter *channel.php* ab, und fügen Sie folgenden Code ein:

```
<?php
 $cache_expire = 60*60*24*365; // = 1 Jahr
 header("Pragma: public");
 header("Cache-Control: max-age=".$cache_expire);
 header('Expires: ' .
        gmdate('D, d M Y H:i:s', time()+$cache_expire) . ' GMT');
 ?>
 <script src="//connect.facebook.net/en_US/all.js"></script>
```

Durch das Setzen der entsprechenden HTTP-Header wird so eine Gültigkeitsdauer des Caches für ein Jahr definiert – dies verhindert, dass der Browser die Channel-Datei bei jedem Seitenaufruf erneut lädt.

Hinweis: Für erste Experimente mit dem JavaScript SDK können Sie durchaus auf das Einrichten einer Channel-Datei verzichten. Für Applikationen im Live-Betrieb sollten Sie jedoch unbedingt auf eine korrekt gecachte Channel-Datei achten.

4.1.3 Lokalisierung des SDKs

Wie die Benutzeroberfläche von Facebook.com selbst steht auch das JavaScript SDK von Facebook in zahlreichen Sprachen zur Verfügung. Beim Laden des SDKs wird die gewünschte Lokalisierung (*Locale*) dabei als Teil der URL zur JavaScript-Datei angegeben: So lädt der folgende Code das SDK im Lokalisierungsschema Englisch (US):

```
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
```

Um das SDK in deutscher Lokalisierung zu laden, muss lediglich der Pfad zur JavaScript-Datei modifiziert werden:

```
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/de_DE /all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
```

Möchten Sie eine mehrsprachige Anwendung entwickeln, ist es ratsam, jeweils jenes SDK zu laden, das der Lokalisierungseinstellung des aktuellen Benutzers entspricht – so werden alle über das SDK angesteuerten Dialoge automatisch in der Sprache des Benutzers angezeigt. Die Lokalisierungseinstellung des aktuellen Benutzers lässt sich in Canvas- und Tab-Anwendungen über den Signed Request ermitteln (siehe Abschnitt 2.4, »Signed Request«). Weitere Informationen zur Internationalisierung Ihrer Anwendung finden Sie in Anhang B, »Internationalisierung von Facebook-Anwendungen«.

Eine Liste aller verfügbaren Lokalisierungen stellt Facebook in einer XML-Datei unter *https://www.facebook.com/translations/FacebookLocales.xml* zur Verfügung.

4.2 Kernmethoden des SDKs

4.2.1 FB.init

Wie bereits gezeigt, wird die Methode `FB.init` nach dem Laden des SDKs zum Initialisieren des SDKs aufgerufen:

```
FB.init (options);
```

Das JSON-Array `options` kann dabei folgende Einstellungen zur Steuerung der Initialisierung enthalten:

Name	Typ	Beschreibung
`appId`	`string`	numerische ID der Anwendung
`cookie`	`boolean`	Legt fest, ob der serverseitige Zugriff auf die Session mittels Cookies ermöglicht werden soll (`true`).
`logging`	`boolean`	Aktivierung des Loggings (`true`)
`status`	`boolean`	Legt fest, ob das SDK den Login-Status des Benutzers prüfen soll (`true`).
`xfbml`	`boolean`	Legt fest, ob das SDK XFML-Tags und Social Plugins zur Laufzeit parsen soll (`true`).
`channelUrl`	`string`	URL zur Channel-Datei
`authResponse`	Objekt	Gibt ein Objekt an, das die Antwort auf eine Autorisierungsanfrage enthält. Diese Information kann alternativ über die Methode `getAuthresponse` abgerufen werden.
`oauth`	`boolean`	Legt fest, ob die Autorisierung mit OAuth 2.0 erfolgen soll (`true`, seit Oktober 2011 vorgeschrieben).
`frictionless-Requests`	`boolean`	Legt fest, ob *Frictionless Sharing* aktiviert werden soll (`true`) – siehe auch Abschnitt 4.4.5, »Requests-Dialog«.

Tabelle 4.1 Optionen der Funktion »FB.init«

4.2.2 FB.api

Die Methode `FB.api` erlaubt es Entwicklern, beliebige Anfragen an die Graph API clientseitig per JavaScript durchzuführen. Dabei werden nicht nur Lesezugriffe

mittels GET unterstützt, sondern auch das Veröffentlichen mittels POST und das Löschen mittels DELETE. `FB.api` unterstützt dabei folgende Parameter:

```
FB.api (path, method, params, callback);
```


Name	Typ	Beschreibung	Pflichtfeld
`path`	`string`	API-Endpunkt, der aufgerufen werden soll	ja
`method`	`string`	verwendete HTTP-Methode – `GET` (Default), `POST` oder `DELETE`	nein
`params`	Objekt	JSON-Objekt mit den zu verwendenden HTTP-Parametern	nein
`callback`	Funktion	Callback-Funktion, die nach der Ausführung aufgerufen werden soll	nein

Tabelle 4.2 Optionen der Funktion »FB.api«

Das folgende Code-Beispiel nutzt die Methode `FB.api`, um einen einfachen GET-Zugriff auf den API-Endpunkt `https://graph.facebook.com/diesocialisten` durchzuführen:

```
<DOCTYPE html>
<html lang="de-de" xmlns=http://www.w3.org/1999/xhtml
      xmlns:fb="http://ogp.me/ns/fb#">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook JS SDK!</title>
</head>
<body>
<a href="#" onclick="fbApi('/diesocialisten');">
   Testing: FB.api('/diesocialisten')</a>

<div id="fb-root"></div>
<script>
function fbApi(path) {
   FB.api(path, function(response) {
      console.debug(response);
   });
}
```

```
window.fbAsyncInit = function() {
   FB.init({
      ...
   });
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
</script>
</body>
</html>
```

Listing 4.3 Einfacher GET-Zugriff auf die Graph API mittels »FB.api«

Ein Anklicken des Links `Testing: ...` löst den Aufruf der Methode `FB.api` aus und gibt das Ergebnis auf der JavaScript-Konsole des Browsers aus:

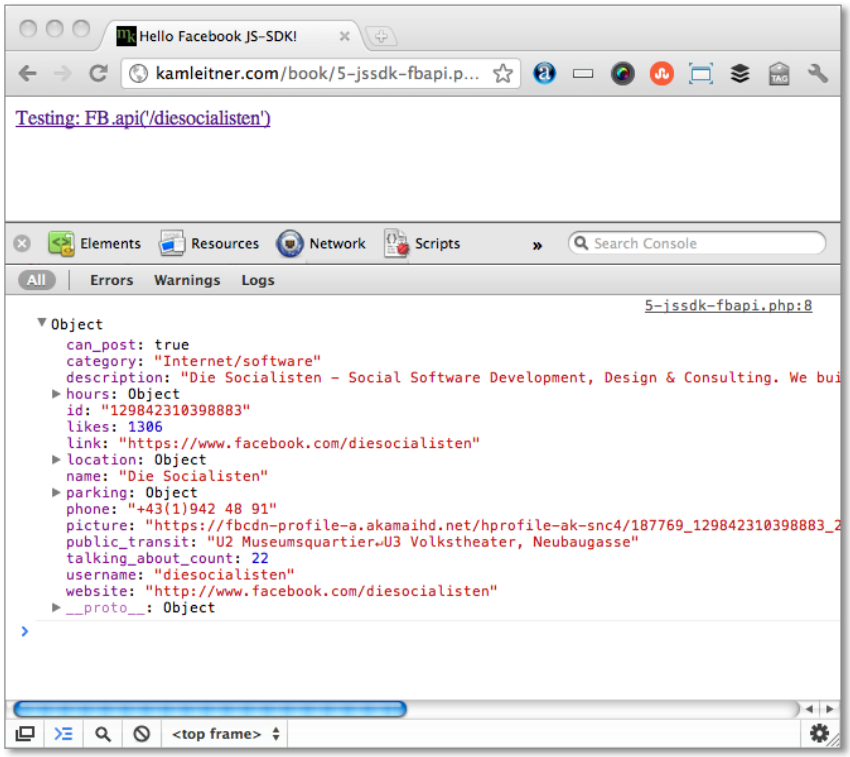


Abbildung 4.3 Ausgabe des FB.api-Aufrufs in der JavaScript-Konsole des Browsers

Und so funktioniert dieser Beispiel-Code:

- Das angezeigte HTML-Dokument zeigt lediglich einen Link zum Testaufruf der weiter unten definierten Funktion `fbApi()` an – dieser Funktion wird der abzufragende API-Endpunkt im Parameter `path` übergeben.
- Die Funktion `fbApi()` ruft ihrerseits die SDK-Funktion `FB.api` auf, wobei der API-Endpunkt im ersten Parameter übergeben wird. Im zweiten Parameter wird inline die Callback-Funktion definiert.
- Die Callback-Funktion erhält das Ergebnis des API-Aufrufs als JSON-Objekt im Parameter `response`. Um das JSON-Objekt einfach am Bildschirm auszugeben, wird die browserspezifische Methode `console.debug()` genutzt. *Achtung:* `console.debug()` steht nur in bestimmten Browsern, etwa Google Chrome oder Firefox mit installiertem Firebug-Plugin, zur Verfügung!
- Da es sich beim abgefragten Objekt um eine öffentliche Seite handelt, ist der Zugriff ohne Access Token möglich.
- *Hinweis:* Der bereits aus vorangegangenen Beispielen bekannte Code zum Initialisieren des SDKs mittels `FB.init()` wurde aus Platzgründen im obigen Beispiel nicht ausgeführt.

Neben GET-Zugriffen lassen sich auch POST-Zugriffe über `FB.api` durchführen. Das folgende Beispiel veröffentlicht ein Wall-Posting auf der Pinnwand des aktuellen Benutzers:

```
var body = Das ist ein Test-Wall-Posting';
FB.api('/me/feed', 'post', { message: body }, function(response) {
   if (!response || response.error) {
      alert('Ein Fehler ist aufgetreten!');
   } else {
      alert('Post ID: ' + response.id);
   }
});
```

Listing 4.4 Veröffentlichen eines Wall-Postings mit »FB.api«

Um ein Objekt im sozialen Graphen zu veröffentlichen, sind dabei die gleichen erweiterten Zugriffsberechtigungen notwendig wie beim in Kapitel 3, »Die Graph API«, beschriebenen serverbasierten Zugriff auf die Graph API. Zum Veröffentlichen von Wall-Postings ist also analog ein Access Token mit der Berechtigung `publish_stream` erforderlich. Sie müssen das Token allerdings nicht explizit an `FB.api()` übergeben, da dies automatisch vom JavaScript SDK erledigt wird.

Um das soeben veröffentlichte Wall-Posting wieder zu löschen, genügt ein DELETE-Zugriff mittels `FB.api()`:

```
var postId = '609190863_1234567890';
FB.api(postId, 'delete', function(response) {
   if (!response || response.error) {
      alert('Ein Fehler ist aufgetreten!');
   } else {
      alert('Das Post wurde gelöscht!');
   }
});
```

Listing 4.5 Löschen eines Wall-Postings mit »FB.api«

Auch dieser Zugriff erfordert einen autorisierten Benutzer mit der Berechtigung `publish_stream`.

4.2.3 FB.login

Die Funktion `FB.login()` kann aufgerufen werden, um den aktuellen Benutzer zur Autorisierung der Applikation aufzufordern. Weil dabei ein Browser-Popup mit dem OAuth-Dialog geöffnet wird, ist es wichtig, diese Methode nur nach einem Mausklick des Benutzers auf einen Link oder Button aufzurufen. Anderenfalls würde das Pop-up von den meisten üblichen Browsern geblockt werden. Der Aufruf der Funktion erfolgt nach diesem Schema:

```
FB.login(handler, options);
```

Name	Typ	Beschreibung
`handler`	`function`	Callback-Funktion, die nach dem Schließen des OAuth-Dialogs aufgerufen wird
`options`	`array`	Array mit zusätzlichen Optionen zum OAuth-Dialog. Die einzige aktuelle verfügbare Option heißt `scope` und enthält eine kommaseparierte Liste an zusätzlichen Berechtigungen, die der Benutzer bei der Autorisierung freigeben soll.

Tabelle 4.3 Parameter der Funktion »FB.login«

Der Callback-Handler von `FB.login()` wird mit einem JSON-Objekt `response` mit folgendem Aufbau aufgerufen:

```
{
   status: "",          // Aktueller Status der Session
   authResponse: {      // Informationen zur aktuellen Session
```

```
      userID: "",        // ID des aktuellen Benutzers
      signedRequest: "", // Aktueller Signed Request
      expiresIn: "",     // Zeitstempel, an dem die Session abläuft
      accessToken: "",   // Access Token des aktuellen Benutzers
  }
}
```


Listing 4.6 Response-Objekt des Callback-Handlers von »FB.login«

Das folgende Code-Beispiel öffnet den Autorisierungsdialog und fordert den Benutzer auf, die erweiterte Berechtigung `email` zu erteilen:

```
<a href="#" onclick="fbLogin();">Login</a>
<script>
var fbLogin = function() {
   FB.login(function(response) {
      console.debug(response);
      if (response.authResponse) {
         console.log('Login erfolgreich!');
      } else {
         console.log('Login abgebrochen!');
      }
   }, {scope: 'email'});
};
...
</script>
```

Listing 4.7 Öffnen des OAuth-Autorisierungsdialogs

Hinweis: Hat der aktuelle Benutzer die Applikation bereits zu einem früheren Zeitpunkt autorisiert, öffnet das JavaScript SDK das Popup-Fenster mit dem OAuth-Dialog nur kurz und schließt es daraufhin automatisch wieder. Der Callback-Handler wird wie gewohnt mit dem `response`-Objekt aufgerufen.

4.2.4 FB.logout

Die Funktion `FB.logout()` meldet den aktuellen Benutzer von der eigenen Anwendung und von Facebook ab. Dabei verliert auch das aktuelle Access Token an Gültigkeit, außer es wurde mit der erweiterten Berechtigung `publish_stream` ausgestellt.

Der Aufruf der Funktion erfolgt nach diesem Schema:

```
FB.logout(handler);
```

Name	Typ	Beschreibung
handler	function	Callback-Funktion, die nach dem Abmelden aufgerufen wird

Tabelle 4.4 Parameter der Funktion »FB.logout«

Das folgende Code-Beispiel meldet den aktuellen Benutzer von der Anwendung und von Facebook ab:

```
<a href="#" onclick="fbLogout();">Logout</a>
<script>
var fbLogout = function() {
   FB.logout(function(response) {
      console.debug(response);
   });
};
...
</script>
```

Listing 4.8 Abmelden des Benutzers mit »FB.logout«

4.2.5 FB.getLoginStatus

`FB.getLoginStatus()` erlaubt es dem Entwickler, den aktuellen Login-Status des Benutzers zu ermitteln. Die Funktion wird typischerweise nach dem Laden der Seite aufgerufen, um etwa je nach Login-Status zu entscheiden, welche Bedienelemente dem Benutzer angezeigt werden sollen.

Hinweis: Beim Aufruf von `FB.getLoginStatus()` wird unter Umständen ein HTTP-Request an Facebook.com ausgelöst, um den Login-Status zu ermitteln. Wurde `FB.getLoginStatus()` in der aktuellen Browser-Session bereits aufgerufen und der Session-Status mittels Cookie gespeichert, entfällt dieser HTTP-Request in Zukunft.

Der Aufruf der Funktion erfolgt nach diesem Schema:

```
FB.getLoginStatus(handler);
```

Name	Typ	Beschreibung
handler	function	Callback-Funktion, die nach dem Ermitteln des Login-Status aufgerufen wird

Tabelle 4.5 Parameter der Funktion »FB.getLoginStatus«

Der Callback-Handler von `FB.getLoginStatus()` wird mit einem JSON-Objekt `response` mit folgendem Aufbau aufgerufen:

```
{
   status: "",           // Aktueller Status der Session
   authResponse: {       // Informationen zur aktuellen Session
      userID: "",        // ID des aktuellen Benutzers
      signedRequest: "", // Aktueller Signed Request
      expiresIn: "",     // Zeitstempel, an dem die Session abläuft
      accessToken: "",   // Access Token des aktuellen Benutzers
  }
}
```

Listing 4.9 Response-Objekt des Callback-Handlers von »FB.getLoginStatus«

Das Feld `status` kann dabei folgende Werte annehmen:

- `connected` – Der Benutzer ist bei Facebook angemeldet und hat die Applikation bereits autorisiert.
- `not_authorized` – Der Benutzer ist bei Facebook angemeldet, hat die Applikation aber noch nicht autorisiert.
- `connected` – Der Benutzer ist nicht bei Facebook angemeldet.

4.2.6 FB.getAuthResponse

`FB.getAuthResponse()` ist vergleichbar mit der Funktion `FB.getLoginStatus()` und ermittelt wie diese den Login-Status des Benutzers. Im Unterschied zu `FB.getLoginStatus()` wird sie jedoch synchron ausgeführt. Das bedeutet, dass JavaScript die nächste Zeile im Code erst ausführt, wenn `FB.getAuthResponse()` durchgeführt wurde. Das Ergebnis wird auch nicht an einen Callback-Handler, sondern als Ergebnis der Funktion selbst zurückgeliefert:

```
var response = FB.getAuthResponse();
```

Das Ergebnisobjekt `response` enthält im Unterschied zu `FB.getLoginStatus()` nur das Objekt `authResponse`, nicht aber das `status`-Feld:

```
{
   authResponse: {       // Informationen zur aktuellen Session
      userID: "",        // ID des aktuellen Benutzers
      signedRequest: "", // Aktueller Signed Request
      expiresIn: "",     // Zeitstempel, an dem die Session abläuft
      accessToken: "",   // Access Token des aktuellen Benutzers
  }
}
```

Listing 4.10 Response-Objekt des Callback-Handlers von »FB.getAuthResponse«

Hinweis: Die Funktion `FB.getAuthResponse()`wird vor allem in jenen Teilen der Anwendung eingesetzt, in denen bereits vorausgesetzt werden kann, dass der Benutzer die Applikation autorisiert hat. Niemals sollte `FB.getAuthResponse()` hingegen beim Laden der Seite aufgerufen werden!

Achtung: Die Funktion `FB.getSession()` liefert wie `FB.getAuthResponse()`den Login-Status des Benutzers, wurde jedoch im Juli 2011 mit der Umstellung auf OAuth 2.0 eingestellt.

4.2.7 FB.XFBML.parse

Die Funktion `FB.XFBML.parse()` ermöglicht es, das Parsen von XFBML-Elementen manuell auszulösen. Bei diesem Prozess durchsucht das SDK das HTML-Dokument (oder einen Ausschnitt daraus) nach allen bekannten XFBML-Tags und ersetzt diese durch die passenden Facebook-spezifischen Inhalte. Üblicherweise wird dies bereits beim Initialisieren des SDKs erledigt, wenn `FB.init()` mit der Option `xfbml:true` aufgerufen wurde. Oftmals sollen aber Inhalte, die zu einem Zeitpunkt nach dem ursprünglichen Laden der Seite mittels AJAX vom Server nachgeladen werden, ebenfalls XFBML-Elemente enthalten. In diesen Fällen ist ein manueller Aufruf von `FB.XFBML.parse()` notwendig.

Um das vollständige HTML-Dokument zu parsen, ist folgender Aufruf notwendig:

```
FB.XFBML.parse();
```

Um nur ein bestimmtes Unterelement des HTML-Dokuments zu parsen (etwa ein Element, dessen Inhalte per AJAX geladen wurden), genügt es, eine Referenz zum Element als Parameter zu übergeben.

```
FB.XFBML.parse(element);
FB.XFBML.parse(document.getElementById('element-id'));
```

Name	Typ	Beschreibung
`Element`	`DOM-Element`	Wurzelelement im HTML-Dokument, ab dem geparst werden soll. Default: `<body>`.
`handler`	`function`	Callback-Funktion, die nach dem Schließen des OAuth-Dialogs aufgerufen wird

Tabelle 4.6 Parameter der Funktion »FB.XFBML.parse«

Der Parser von `FB.XFBML.parse()` verarbeitet folgende XFBML-Elemente:

```
fb:activity, fb:add-profile-tab,
fb:bookmark, fb:comments, fb:friendpile, fb:like, fb:like-box,
fb:live-stream, fb:login-button, fb:recommendations, fb:serverFbml,
fb:profile-pic, fb:name, fb:user-status
```


4.3 Canvas-Methoden des SDKs

Die unter `FB.Canvas` gesammelten Methoden dienen der Kommunikation zwischen der Canvas-Applikation und der darüberliegenden Facebook-Plattform. Mit diesen Funktionen können z. B. Informationen über den Parent-Frame (jenes HTML-Dokument, in dem der iFrame der Applikation eingebettet ist) ermittelt oder auch das Verhalten des iFrames beeinflusst werden.

4.3.1 FB.Canvas.getPageInfo

Die Funktion `FB.Canvas.getPageInfo()` liefert ein JavaScript-Objekt mit Informationen über die Dimension des Parent-Frames, also jenes Application Canvas, in dem die eigene Anwendung mittels iFrame integriert ist. Die Funktion wird mit einem Callback-Handler als Parameter aufgerufen:

```
FB.Canvas.getPageInfo (handler);
```

Name	Typ	Beschreibung
`handler`	`function`	Callback-Funktion, die nach dem Ermitteln der Dimensionen des Parent-Frames aufgerufen wird

Tabelle 4.7 Parameter der Funktion »FB.Canvas.getPageInfo«

Der folgende Beispiel-Code ruft `FB.Canvas.getPageInfo()` auf und registriert einen Callback-Handler, der anschließend das zurückgegebene Objekt auf der Konsole ausgibt:

```
FB.Canvas.getPageInfo( function(info) {
   console.debug(info);
});
```

Das Objekt `info` enthält dabei folgende Felder:

```
{
   clientHeight: ...,   // Höhe des Parent-Frames in Pixeln
   clientWidth: ...,    // Breite des Parent-Frames in Pixeln
```

```
    offsetLeft: ...,     // Pixel zwischen linkem Rand des Parent-
                         // Frames und linkem Rand des App-iFrames
    offsetTop: ...,      // Pixel zwischen oberem Rand des Parent-
                         // Frames und oberem Rand des App-iFrames
    scrollLeft: ...,     // Anzahl der Pixel, um die der Inhalt
                         // des App-iFrames nach links verschoben ist
    scrollTop: ...       // Anzahl der Pixel, um die der Inhalt
                         // des App-iFrames nach oben verschoben ist
}
```

Listing 4.11 Response-Objekt des Callback-Handlers von »FB.Canvas.getPageInfo«

4.3.2 FB.Canvas.scrollTo

Die Funktion `FB.Canvas.scrollTo()` ermöglicht es dem Anwendungsentwickler, den sichtbaren Ausschnitt (*Viewport*) des Parent-Frames auf eine bestimmte Position zu setzen. Damit kann aus dem iFrame der Anwendung heraus ein Scrollen des Application Canvas erwirkt werden. Dies ist praktisch, um etwa beim Laden einer neuen Seite innerhalb des iFrames den Parent-Frame wieder ganz nach oben zu bewegen.

Hinweis: `FB.Canvas.scrollTo()` kann nur aufgerufen werden, wenn die Höhe des Canvas in den Anwendungseinstellungen auf SETTABLE (DEFAULT: 800PX) gesetzt wurde!

Aufgerufen wird die Funktion folgendermaßen:

```
FB.Canvas.scrollTo(x,y);
```

Name	Typ	Beschreibung
x	int	horizontale Scroll-Position in Pixeln
y	int	vertikale Scroll-Position in Pixeln

Tabelle 4.8 Parameter der Funktion »FB.Canvas.scrollTo«

4.3.3 FB.Canvas.setSize

Ein wiederkehrendes Problem von Canvas-Applikationen ist die oft dynamische Dimension des App-iFrames. Werden z. B. nach dem ursprünglichen Laden des iFrames Inhalte mittels AJAX hinzugefügt (etwa durch Ausklappen einer Textbox), wächst die Höhe des iFrame-Inhalts. Ohne das Zutun des Entwicklers würde der Parent-Frame von der veränderten Dimension des iFrame-Inhalts keine Notiz nehmen – das Ergebnis sind unschöne Scroll-Balken im iFrame. Die Funktion `FB.Canvas.`

`setSize()` schafft Abhilfe, indem sie das Anpassen der iFrame-Größe zur Laufzeit erlaubt. Ohne weitere Parameter aufgerufen, sorgt das SDK dabei selbstständig für eine Dimensionierung, die gerade groß genug für die Inhalte des iFrames ist:

```
FB.Canvas.setSize();
```


Alternativ kann beim Aufruf auch die gewünschte Dimension als Objekt übergeben werden:

```
FB.Canvas.setSize( { width: 760, height: 800 });
```

Name	Typ	Beschreibung
`options`	Objekt	Objekt mit den Feldern `width` und `height`

Tabelle 4.9 Parameter der Funktion »FB.Canvas.setSize«

Hinweis: `FB.Canvas.setSize()` kann nur aufgerufen werden, wenn die Höhe des Canvas in den Anwendungseinstellungen auf SETTABLE (DEFAULT: 800PX) gesetzt wurde!

4.3.4 FB.Canvas.setAutoGrow

Die Funktion `FB.Canvas.setSize()` hat einen entscheidenden Nachteil: Der Entwickler muss sie, jedes Mal, wenn sich der Inhalt des iFrames so ändert, dass eine Redimensionierung notwendig ist, eigenständig aufrufen. Mit der Funktion `FB.Canvas.setAutoGrow()` geht es auch einfacher: Diese Methode konfiguriert einen JavaScript-Timer, der im Abstand von wenigen Millisekunden die Größe des Anwendungsinhalts prüft und die Größe des iFrames erhöht, wenn der Inhalt gewachsen ist (umgekehrt wird der iFrame allerdings nicht verkleinert, was normalerweise aber kein Problem darstellt, da in diesem Fall keine störenden Scroll-Balken erscheinen). `FB.Canvas.setAutoGrow()` muss also nur einmalig beim Laden der Applikation aufgerufen werden und kümmert sich fortan selbstständig um die Dimensionierung des iFrames:

```
FB.Canvas.setAutoGrow();
```

Alternativ kann der Timer beim Aufruf von `FB.Canvas.setAutoGrow()` mittels der Variablen `mode` ein- oder ausgeschaltet und über `interval` auch zeitlich spezifiziert werden:

```
FB.Canvas.setAutoGrow ( mode, interval);
```

Name	Typ	Beschreibung
mode	boolean	Aktivierung bzw. Deaktivierung von FB.Canvas.setAutoGrow()
interval	integer	Dauer des Intervals, in dem die Größe des iFrames geprüft wird, in Millisekunden

Tabelle 4.10 Parameter der Funktion »FB.Canvas.setAutoGrow«

Hinweis: FB.Canvas.setAutoGrow() kann nur aufgerufen werden, wenn die Höhe des Canvas in den Anwendungseinstellungen auf Settable (Default: 800px) gesetzt wurde!

Achtung: FB.Canvas.setSize() ersetzt mit dem 1. Januar 2012 die bisher ebenfalls verfügbare Funktion FB.CanvasAutoResize().

Das folgende Code-Beispiel zeigt eine typische Einbindung des JavaScript SDKs, wie sie etwa für Seiten-Tabs oder Canvas-Applikationen geeignet ist. Unmittelbar nach dem Initialisieren der App erfolgen zwei SDK-Aufrufe:

- FB.Canvas.setAutoGrow() sorgt dafür, dass der iFrame, in dem die Anwendung geladen wird, automatisch mit dem Inhalt der Anwendung mitwächst.
- FB.Canvas.scrollTo(0,0) sorgt dafür, dass die integrierende Canvas-Seite bei jedem neuen Seitenaufruf innerhalb des iFrames in die linke obere Ecke gescrollt wird. Damit wird sichergestellt, dass der Browser auch beim Laden einer Anwendungsseite mit weniger Inhalt als der vorangegangenen immer den Seitenbeginn darstellt.

```
<div id="fb-root"></div>
<script>
window.fbAsyncInit = function() {
   FB.init({
      appId      : '214728715257742',
                   // Anwendungs-ID
      channelUrl : '//apps.mycompany.com/channel.php',
                   // Channel-Datei
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
   });
```

```
   FB.Canvas.setAutoGrow();
   FB.Canvas.scrollTo(0, 0);
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
</script>
```

Listing 4.12 Laden des Facebook JavaScript SDKs und Aufrufen von »FB.Canvas.setAutoGrow« und »FB.Canvas.scrollTo«

4.3.5 FB.Canvas.setUrlHandler

Die Funktion `FB.Canvas.setUrlHandler()` erlaubt die Verarbeitung von bestimmten anwendungsbezogenen Klick-Events auf der Canvas-Seite, in der der Anwendungs-iFrame eingebunden ist. Normalerweise würden diese Events ein Neuladen des Application Canvas (*http://apps.facebook.com/...*) bewirken – mit `FB.Canvas.setUrlHandler()` können Entwickler diese Events abfangen und selbst verarbeiten, wenn der Benutzer sich gerade ohnehin am Application Canvas befindet.

Hinweis: Die Verwendung von `FB.Canvas.setUrlHandler()` ändert also nicht die grundlegende Funktionsweise einer Anwendung, sondern sorgt lediglich für eine reibungslosere Bedienung, da das vollständige Neuladen von Application Canvas und iFrame entfallen kann.

Folgende Events werden von `FB.Canvas.setUrlHandler()` verarbeitet:

- Klicks auf eine Ticker-Story der aktuellen Anwendung
- Klicks auf ein Bookmark der aktuellen Anwendung
- Klicks auf eine Anwendungsanfrage der aktuellen Anfrage
- Klicks auf eine Benachrichtigung über eine Anwendungsanfrage der aktuellen Anfrage

In allen Fällen wird anstelle des Neuladens des Application Canvas der Callback-Handler aufgerufen, der mit `FB.Canvas.setUrlHandler()` registriert wurde:

```
FB.Canvas.setUrlHandler(handler)
```

Name	Typ	Beschreibung
handler	function	Callback-Funktion, die beim Klick-Ereignis aufgerufen werden soll

Tabelle 4.11 Parameter der Funktion »FB.Canvas.setUrlHandler«

Der Callback-Handler nimmt dabei ein JSON-Objekt im Parameter `data` entgegen, das lediglich ein Feld namens `path` enthält:

```
{
   path: '/relative/path/to/app'
}
```

`path` repräsentiert dabei den relativen Pfad zum Application Canvas, der normalerweise geladen worden wäre. Wäre z. B. beim Klick auf ein Bookmark die URL *http://apps.facebook.com/myapp/home* geladen worden, enthielte `path` den Wert `/home`.

Das folgende Code-Beispiel zeigt die Registrierung eines Callback-Handlers, der nichts anderes tut, als die in `path` übergebene Seite im Anwendungs-iFrame mittels `document.location.href...` zu laden.

```
FB.Canvas.setUrlHandler( function(data) {
   //console.debug(data);
   document.location.href = data.path;
});
```

Listing 4.13 Registrieren eines Callback-Handlers

4.3.6 FB.Canvas.setDoneLoading

Die Funktion `FB.Canvas.setDoneLoading()` erlaubt die Registrierung eines JavaScript-Timers, der die Ladezeit des Anwendungs-iFrames vom Eintreffen der ersten Bytes der Request-Antwort am Browser bis zum Zeitpunkt des Aufrufs von `FB.Canvas.setDoneLoading()` misst:

```
FB.Canvas.setDoneLoading(handler);
```

Name	Typ	Beschreibung
handler	function	Callback-Funktion, die nach dem vollständigen Laden der Seite aufgerufen werden soll

Tabelle 4.12 Parameter der Funktion »FB.Canvas.setDoneLoading«

Das an den Callback-Handler übergebene JSON-Objekt `result` enthält dabei lediglich ein Feld `time_delta_ms` mit der Ladezeit in Millisekunden:

```
{
   time_delta_ms: 1343
}
```


Das folgende Code-Beispiel registriert einen Callback-Handler und gibt die Ladedauer der Seite mittels `alert()` aus:

```
FB.Canvas.setDoneLoading( function(result) {
   console.debug(result);
   alert(result.time_delta_ms);
});
```

Listing 4.14 Registrieren eines Callback-Handlers, der die Ladedauer der Seite ausgibt

Hinweis: Ein angenehmer Nebeneffekt bei der Verwendung von `FB.Canvas.setDoneLoading()` ist, dass Facebook, ausgelöst durch die Aufrufe der SDK-Funktion, beginnt, die Kennzahl PAGE LOAD TIME in den Insights der Anwendung zu protokollieren. Da diese Zahlen einen guten Hinweis auf die Lade-Performance der Anwendung geben, lohnt es sich also, `FB.Canvas.setDoneLoading()` generell bei jedem Neuladen aufzurufen, auch wenn das Ergebnis ansonsten nicht verarbeitet wird.

4.3.7 FB.Canvas.stopTimer

In bestimmten Fällen kann es sinnvoll sein, den Timer, den die Funktion `FB.Canvas.setDoneLoading()` zur Messung der Ladezeit nutzt, zu stoppen. Dies ist immer dann notwendig, wenn eine Benutzereingabe – etwa eine Willkommensnachricht – erwartet wird, die den Ladevorgang der Seite künstlich verzögert. Die Funktion `FB.Canvas.stopTimer()` kann genutzt werden, um den Timer vorübergehend abzuhalten:

```
FB.Canvas.stopTimer(handler);
```

Name	Typ	Beschreibung
`handler`	`function`	Callback-Funktion, die nach dem Stoppen des Timers aufgerufen werden soll

Tabelle 4.13 Parameter der Funktion »FB.Canvas.stopTimer«

Das an den Callback-Handler übergebene JSON-Objekt `result` enthält dabei wieder ein Feld `time_delta_ms` mit der bisherigen Ladzeit in Millisekunden:

```
{
   time_delta_ms: 1343
}
```

4.3.8 FB.Canvas.startTimer

Ein mit `FB.Canvas.stopTimer()` angehaltener Timer kann mit der Funktion `FB.Canvas.startTimer()` wieder gestartet werden. Dies ist etwa sinnvoll, wenn ein vom Benutzer erwartetes Ereignis eingetreten ist, wenn der Benutzer also z. B. die Willkommensnachricht per Klick geschlossen hat.

```
FB.Canvas.startTimer();
```

Der Aufruf von `FB.Canvas.startTimer()` erfolgt ohne die Angabe von Parametern.

4.3.9 FB.Canvas.Prefetcher.addStaticResource

Anwendungen, die auf einem Seiten-Tab oder Application Canvas ausgeführt werden, können statische Ressourcen (CSS-, JavaScript- oder Flash-Dateien) der Anwendung über den Prefetching-Mechanismus des SDKs im Vorhinein laden. Dies verbessert die Performance vor allem in jenen Fällen, in denen die statischen Objekte nicht im HTML-Dokument referenziert werden, sondern dynamisch per JavaScript nachgeladen werden. Auch in Situationen, in denen große statische Ressourcen erst beim zweiten aufeinanderfolgenden Seitenaufruf im Anwendungs-iFrame geladen werden, kann Prefetching die Performance verbessern.

Um eine statische Ressource im Prefetcher zu registrieren, genügt der Aufruf der Methode `FB.Canvas.Prefetcher.addStaticResource()` mit der URL zum Objekt:

```
FB.Canvas.Prefetcher.addStaticResource
('http://apps.mycompany.com/flash/large.swf');
```

Damit der Prefetching-Mechanismus des SDKs funktioniert, müssen die statischen Ressourcen mit folgenden HTTP-Headern ausgeliefert werden:

- Der `Content-Type` muss einem der folgenden entsprechen: `application/x-shockwave-flash`, `application/x-compress`, `text/css`, `application/javascript` und `application/x-javascript`.
- Die `Content-Length` muss korrekt gesetzt werden, da das SDK eigenständig nur die größten Ressourcen mittels Prefetching lädt.

- Ist `Cache-Control` auf `no-cache`, `no-store` oder `private` gesetzt, ignoriert der Prefetching-Mechanismus die Ressource.
- Ist `Vary` gesetzt, ignoriert der Prefetching-Mechanismus die Ressource.
- Ist nur einer der beiden Header `Expires` und `Last-Modified` gesetzt oder liegen die Werte zu nahe an der aktuellen Uhrzeit, ignoriert der Prefetching-Mechanismus die Ressource.

4.3.10 FB.Canvas.Prefetcher.setCollectionMode

Normalerweise läuft der Prefetching-Mechanismus des SDKs in einem automatischen Modus. Dabei versucht das SDK eigenständig, zum Prefetching geeignete Ressourcen zu erkennen. Entwickler können auch in diesem Modus zusätzlich mittels `FB.Canvas.Prefetcher.addStaticResource()` eigene Objekte zum Prefetching registrieren. Dabei kann es allerdings vorkommen, dass das SDK eigenständig entscheidet, welche Ressourcen am sinnvollsten vorgeladen werden können.

Im manuellen Modus liegt die Kontrolle hingegen vollständig bei Ihnen – nur mittels `FB.Canvas.Prefetcher.addStaticResource()` registrierte Ressourcen werden beim Prefetching berücksichtigt. Der Modus, in dem Prefetching ausgeführt wird, kann über folgende Aufrufe der Methode `FB.Canvas.Prefetcher.setCollectionMode()` definiert werden:

```
FB.Canvas.Prefetcher.setCollectionMode
(FB.Canvas.Prefetcher.COLLECT_AUTOMATIC)
FB.Canvas.Prefetcher.setCollectionMode
(FB.Canvas.Prefetcher.COLLECT_MANUAL)
```

4.4 Dialoge mit FB.ui

Das JavaScript SDK von Facebook stellt mit der Funktion `FB.ui()` eine mächtige Hilfsfunktion bereit, die es Entwicklern erlaubt, verschiedene generische Dialoge von Facebook aufzurufen. Die Dialoge werden dabei im gewohnten CSS-Styling von Facebook dargestellt, was vor allem innerhalb von Canvas-Anwendungen für eine konsistente Benutzerführung sorgt. `FB.ui()` unterstützt folgende Arten von Dialogen:

- Dialoge zum Veröffentlichen von Pinnwand-Einträgen
- Dialoge zum Senden einer Freundschaftsanfrage
- Dialoge zum Durchführen von Zahlungstransaktionen mittels Facebook Credits
- Dialoge zum Versenden von Applikationsanfragen (*Invites*)
- Dialoge zum Teilen von Links

Der Aufruf von `FB.ui()` erfolgt dabei immer nach folgendem Muster:

```
FB.ui (options, handler);
```

Name	Typ	Beschreibung
`options`	Objekt	JSON-Objekt, dessen Felder den Dialogtyp (`method`), das Erscheinungsbild (`display`) und die Funktionsweise der Dialoge konfigurieren. Je nach Dialogtyp können andere Optionen gesetzt werden.
`handler`	`function`	Callback-Funktion, die nach dem Stoppen des Timers aufgerufen werden soll

Tabelle 4.14 Parameter der Funktion »FB.ui«

Die verfügbaren Dialoge werden in den folgenden Abschnitten anhand von ausführlichen Code-Beispielen dargestellt.

Grundlegende FB.ui-Optionen

Während die meisten verfügbaren Konfigurationsoptionen von `FB.ui()` vom aufgerufenen Dialogtyp abhängen, stehen einige spezielle Optionen immer zur Verfügung:

- `method` gibt an, welcher Dialogtyp aufgerufen werden soll. Beispielwerte für `method` sind `feed` (Wall-Posting), `pay` (Transaktion mit Facebook Credits) oder `apprequests` (Anwendungsanfragen).
- `display` legt die Darstellungsweise fest, mit der Dialoge aufgerufen werden. Gültige Werte sind `page` (Anzeige auf einer eigenen, neu geladenen Webseite), `popup` (Darstellung als Browser-Popup), `iframe` (Darstellung als iFrame, der über die aktuelle Seite geladen wird), `touch` (Darstellung auf einem Smartphone) und `wap` (Anzeige auf einem älteren Mobiltelefon, wird nicht mehr unterstützt). Für den Aufruf per JavaScript kann die explizite Angabe von `display` meist unterbleiben – das SDK wählt dann automatisch die sinnvollste Darstellungsform. Eine Besonderheit ist dabei, dass das SDK aus Sicherheitsgründen Dialoge immer in einem Browser-Popup öffnet, wenn der Benutzer die aktuelle Anwendung noch nicht autorisiert hat, während autorisierte Benutzer die elegantere iFrame-Variante zu sehen bekommen.

Dialoge mit URL-Aufruf öffnen

Die in den folgenden Abschnitten beschriebenen Dialoge lassen sich grundsätzlich auf zwei Arten aufrufen:

- Aufruf über die SDK-Methode `FB.ui()` – der Aufruf per JavaScript erlaubt eine möglichst nahtlose Integration der Facebook-Dialoge in die eigene Anwendung, egal, ob es sich um eine Canvas-Anwendung oder eine Anwendung unter eigener Domain handelt.
- Direkter Aufruf über eine URL – alle Dialoge können alternativ durch den Aufruf der URL *http://facebook.com/dialog/DialogTyp...* geöffnet werden. `DialogTyp` gibt dabei den gewünschten Dialog an, der geöffnet werden soll (`feed`, `friends` etc.). Die URL kann dabei serverseitig mittels Redirect oder clientseitig als normaler Link geladen werden. Die gewünschten Dialogoptionen werden als HTTP-GET-Parameter an die Dialog-URL übergeben. Dabei ist zu beachten, dass hierbei zumindest die Parameter `app_id` und `redirect_uri` gesetzt werden müssen. `app_id` stellt die notwendige Verbindung zur eigenen Anwendung her – dies würde beim Aufruf mittels JavaScript automatisch passieren. `redirect_uri` ist jene URL, auf die der Benutzer nach Beenden des Dialogs von Facebook weitergeleitet wird. *Wichtig:* Die URL muss dabei innerhalb der unter App Domain in den Anwendungseinstellungen angegebenen Domain liegen! Wird einer der beiden Parameter gar nicht oder ungültig gesetzt, öffnet Facebook den Dialog mit einer Fehlermeldung.

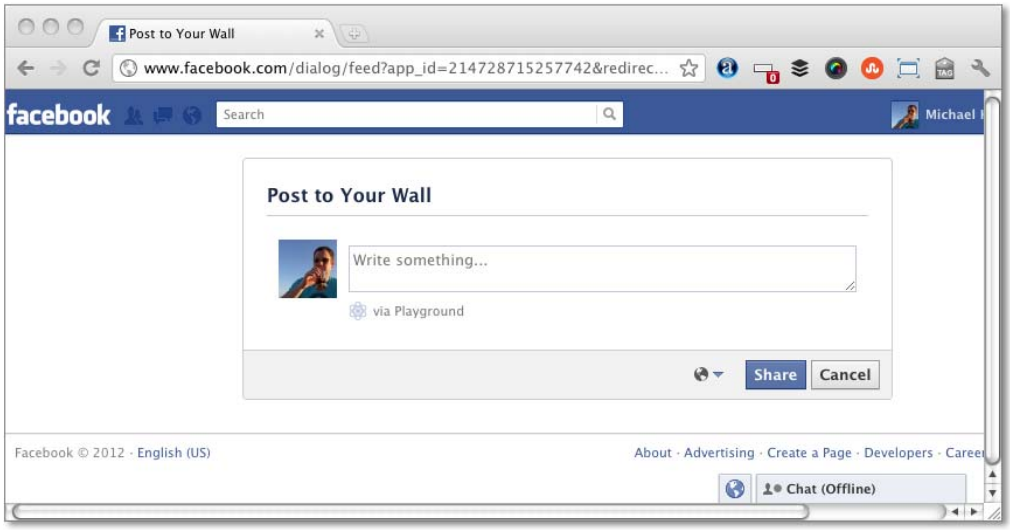


Abbildung 4.4 Dialoge, die mittels URL-Aufruf geöffnet wurden, werden innerhalb von Facebook.com dargestellt.

4.4.1 Feed-Dialog – Veröffentlichen von Pinnwand-Einträgen

Der wohl wichtigste Dialogtyp, den `FB.ui()` für Anwendungsentwickler zugänglich macht, ist jener zum Veröffentlichen von Pinnwand-Beiträgen bzw. Wall-Postings. Um den Feed-Dialog aufzurufen, muss `FB.ui()` der Wert `feed` für den Parameter `method` im Optionsobjekt übergeben werden. Ohne die Übergabe weiterer Parameter öffnet `FB.ui()` daraufhin ein einfaches Pop-up, in dem der Benutzer eine Statusnachricht veröffentlichen kann:

```
var options = {
      method: 'feed',
   };
FB.ui(options);
```

Der geöffnete Dialog sieht folgendermaßen aus:

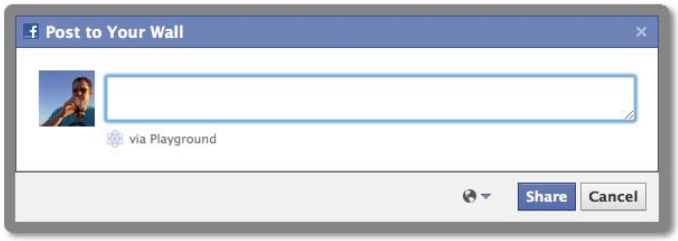


Abbildung 4.5 Einfacher Feed-Dialog ohne weitere Optionsparameter

Hinweis: Bis zum Juli 2011 war es möglich, mit der Option `message` die Nachricht, die vom Benutzer im Feed-Dialog veröffentlicht wird, vorzudefinieren. Diese Möglichkeit ist nicht mehr vorhanden, um zu fördern, dass Benutzer der Anwendung tatsächlich eigene Nachrichten verfassen.

Hinweis: Zum Aufruf des Feed-Dialogs sind keine erweiterten Berechtigungen notwendig! Insbesondere die Berechtigung `publish_stream` wird nicht benötigt, da das Veröffentlichen von Pinnwand-Einträgen nicht automatisch erfolgt, sondern immer die manuelle Zustimmung des Benutzers erfordert. Mehr noch: Zum Aufruf des Feed-Dialogs ist es nicht einmal notwendig, dass der Benutzer die aktuelle Anwendung bereits autorisiert hat!

Feed-Dialog mit URL-Aufruf öffnen

Um den Feed-Dialog mittels Redirect oder Direkt-Link zu öffnen, muss die URL `http://www.facebook.com/dialog/feed` im Browser des Benutzers geladen werden. Die GET-Parameter enthalten dabei neben den Pflichtfeldern `app_id` und `redirect_uri` die gewünschten Optionsparameter, die analog zur SDK-Funktion `FB.ui()` benannt sind.

Der folgende URL-Aufruf öffnet den Feed-Dialog und teilt die URL *http://developers.facebook.com* auf der Pinnwand des aktuellen Benutzers:

```
http://www.facebook.com/dialog/feed
    ?app_id=123050457758183
    &redirect_uri=http://apps.mycompany.com/
    &link=http://developers.facebook.com
    &name=Facebook+Developers
```

`FB.ui()` ist aber zu wesentlich mehr in der Lage, als dem Benutzer eine bloße Aufforderung zum Verfassen eines Status-Updates anzuzeigen. Das Optionsobjekt unterstützt eine Vielzahl an Parametern, die im Wesentlichen den Möglichkeiten der Graph API zum Veröffentlichen von Pinnwand-Einträgen entsprechen:

Name	Typ	Beschreibung
`app_id`	`int`	numerische ID der Anwendung – nur beim direkten Aufruf des Dialogs per URL
`redirect_uri`	`string`	URL, zu der der Browser nach Beenden des Dialogs weitergeleitet wird – nur beim direkten Aufruf des Dialogs per URL
`display`	`string`	Anzeigemodus für den Dialog. Beim Aufruf über das JavaScript SDK wird dieser Parameter üblicherweise nicht verwendet.
`from`	`int`	Numerische ID des Absenders. Wird dieser Parameter nicht angegeben, ist der Absender der aktuelle Benutzer. Andere gültige Werte für diesen Parameter sind die IDs von Seiten, die der Benutzer administriert.
`to`	`int`	Numerische ID des Benutzers oder der Seite, auf dessen bzw. deren Pinnwand gepostet werden soll. Per Default wird hier der Wert von `from`, also meist die ID des aktuellen Benutzers und damit dessen Pinnwand, verwendet.
`message`	`string`	Definition der Benutzernachricht, die veröffentlicht werden soll. *Seit Juli 2011 eingestellt.*
`link`	`string`	URL einer Webseite, die als Link zum Wall-Posting hinzugefügt werden soll

Tabelle 4.15 Das Optionsobjekt zum Aufrufen des Feed-Dialogs mit »FB.ui«

Name	Typ	Beschreibung
picture	string	URL zu einem Bild von mindestens 50 x 50 Pixeln Größe, das zum Wall-Posting hinzugefügt werden soll
source	string	URL zu einer Flash-Datei oder einem Film, der zum Wall-Posting hinzugefügt werden soll
name	string	Linktext, mit dem der Link beschriftet werden soll
caption	string	Text, der unter dem Link dargestellt werden soll
description	string	längerer Beschreibungstext, der unter caption dargestellt werden soll
properties	Objekt	JSON-Array, das beliebig viele Feldpaare mit den Namen text und href enthält. Diese werden als Liste unter dem Beschreibungstext angezeigt und können weiterführende Links oder Aktionen zum Wall-Posting enthalten.
actions	string	JSON-Objekt, das maximal ein Feldpaar mit den Namen name und link enthält. Dieses wird unter dem Wall-Posting, neben den Links für Comment und Like, angezeigt und kann ebenfalls einen weiterführenden Link enthalten.
ref	string	Ein frei wählbarer Kategorie-String, der in Facebook Insights genutzt wird, um unterschiedliche Posting-Typen zu unterscheiden

Tabelle 4.15 Das Optionsobjekt zum Aufrufen des Feed-Dialogs mit »FB.ui« (Forts.)

Das folgende Code-Beispiel demonstriert einige der Optionsparameter von FB.ui():

```
<DOCTYPE html>
<html lang="de-de" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:fb="http://ogp.me/ns/fb#">
<body>

Caption:
<input type="text" id="caption" value="Das ist der Caption-Text"/><br/>

Name:  <input type="text" id="name"
value="Das ist der Linktext"/><br/>
```

```
Picture: <input type="text" id="picture"
value="http://fbrell.com/f8.jpg"/><br/>
Link: <input type="text" id="link"
value="https://developers.facebook.com/..."/><br/>

Description: <textarea id="description">Das ist der Description-Text.
 Er darf ruhig etwas länger sein, wird jedoch von Facebook nach ca.
 270 Zeichen abgeschnitten.
 Der überschüssige Text verbirgt sich hinter einem mit "Read More" betitelten
 Link und erscheint, wenn dieser Link vom Benutzer angeklickt wird.
 Normalerweise sind 270 Zeichen aber völlig ausreichend, um den geteilten Link
 genauer zu beschreiben.</textarea></br>
<input type="button" value="Post!" onclick="fbUiDialog();"/>

<div id="fb-root"></div>
<script>
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
                   // Anwendungs-ID
      channelUrl : '//apps.mycompany.com/channel.php',
                    // Channel-Datei
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
};
var fbUiDialog = function() {
   var options = {
       method: 'feed',
       link: document.getElementById('link').value,
       picture: document.getElementById('picture').value,
       name: document.getElementById('name').value,
       caption: document.getElementById('caption').value,
       description: document.getElementById('description').value,
   };
   FB.ui(options, function(result) {
       console.debug(result);
   });
};
```

Listing 4.15 Aufruf des Feed-Dialogs mit einigen der gebräuchlichsten Optionen

Über ein HTML-Formular können die gewünschten Werte für Link, Bild, Beschreibung etc. eingegeben und anschließend mit einem Klick auf den Button POST! an den Dialog übergeben werden:

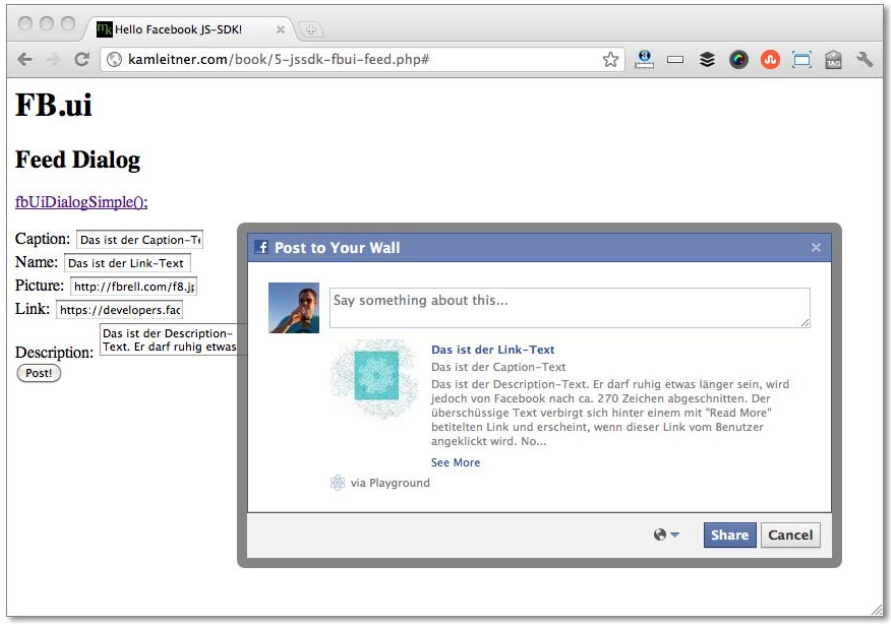


Abbildung 4.6 Aufruf des Feed-Dialogs mit einigen der gebräuchlichsten Optionen

Der Callback-Handler des Feed-Dialogs

Nachdem der Benutzer auf SHARE oder CANCEL geklickt hat, übergibt das JavaScript SDK an den im zweiten Parameter optional registrierten Callback-Handler:

```
FB.ui(options, function(result) {
      console.debug(result);
   });
```

Der Rückgabewert `result` ist dabei leer, wenn der Benutzer den Dialog abgebrochen hat, und enthält im Feld `post_id` die ID des erzeugten Wall-Postings, wenn der Dialog mit SHARE bestätigt wurde:

```
{
   post_id: "609190863_300494266653372"
}
```

Die ID des Wall-Postings kann zur weiteren Verarbeitung in der Datenbank der Anwendung gespeichert werden. Das auf der Pinnwand des Benutzers erzeugte Wall-Posting wird dabei folgendermaßen dargestellt:

Abbildung 4.7 Pinnwand-Darstellung des per Feed-Dialog erzeugten Wall-Postings

Veröffentlichen von Wall-Postings mit Properties und Actions

Die Parameter `actions` und `properties` bedürfen näherer Erklärung. `properties` sind weiterführende Links, die in Listenform unter dem Beschreibungstext des Wall-Postings angezeigt werden. Entwickler können eine beliebige Anzahl an `properties` in Form eines JSON-Arrays, das aus Feldpaaren mit den Feldern `text` (Linktext) und `href` (URL) besteht, verwenden.

Im Parameter `actions` kann *maximal ein sogenannter Action-Link* übergeben werden. Auch der Action-Link ist ein weiterführender Link, wird jedoch im Unterschied zu `properties` unter den Wall-Postings, neben den Links COMMENT und LIKE, angezeigt. Das SDK erwartet im Parameter `actions` ein Array mit den Feldern `name` (Linktext) und `link` (URL).

Der folgende Beispiel-Code öffnet den Feed-Dialog mit den gesetzten Parametern `actions` und `properties`:

```
var options = {
    method: 'feed',
    link: document.getElementById('link').value,
    picture: document.getElementById('picture').value,
    name: document.getElementById('name').value,
    caption: document.getElementById('caption').value,
    properties: [ { text: 'Link #1',
                    href: 'http://www.facebook.com' },
                  { text: 'Link #2',
                    href: 'http://developers.facebook.com'
                ],
    actions: { name: 'Action #1',
               link: 'http://www.facebook.com' },
};
FB.ui(options);
```

Listing 4.16 Aufruf des Feed-Dialogs mit gesetzten Parametern »actions« und »properties«

Der Feed-Dialog, der dem Benutzer nach diesem Aufruf präsentiert wird, zeigt den gesetzten Parameter `properties` bereits in der Vorschau, während der in `actions` gesetzte Action-Link für den Benutzer bis zur Veröffentlichung unsichtbar bleibt:

Abbildung 4.8 Feed-Dialog mit gesetzten Parametern »actions« und »properties«

Erst am veröffentlichten Wall-Posting zeigt sich die Auswirkung von `actions`:

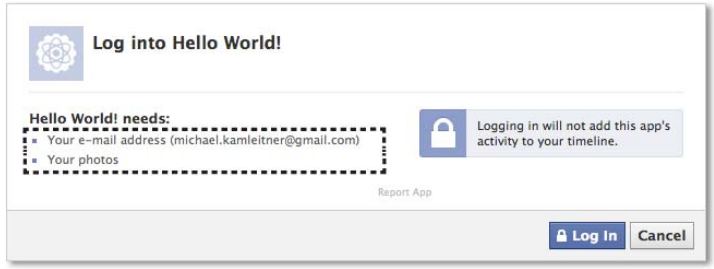


Abbildung 4.9 Wall-Posting mit »properties« und Action-Link (hervorgehoben)

4.4.2 Friends-Dialog – Versenden von Freundschaftsanfragen

Eine seltener genutzte Spielart von `FB.ui()` ist der Friends-Dialog. Er bietet Entwicklern die Möglichkeit, Benutzer der Anwendung zum Versenden einer Freundschaftsanfrage an einen anderen Benutzer aufzufordern.

Hinweis: Auch hier gilt: Das Versenden der Freundschaftsanfrage erfolgt nicht automatisch, sondern erfordert die explizite Bestätigung des Benutzers der Anwendung. Facebook fordert Plattform-Entwickler auf, den Friends-Dialog nur zu verwenden, um Benutzer, die sich im realen Leben tatsächlich kennen, zu verbinden. Die Nutzung des Friend-Dialogs zum wahllosen Versenden von Freundschaftsanfragen wird hingegen explizit ausgeschlossen!

Um den Friends-Dialog aufzurufen, muss die Option `method` auf den Wert `friends` gesetzt und in der Option `id` die numerische ID des Benutzers übergeben werden, dessen Freundschaft angefragt werden soll:

```
var options = {
      method: 'friends',
      id: 609190863
   };
   FB.ui(options, function(result) {
      console.debug(result);
   });
```

Listing 4.17 Aufrufen des Friends-Dialogs mit »FB.ui«

Der auf diese Weise aufgerufene Friends-Dialog wird dabei folgendermaßen dargestellt:

Abbildung 4.10 Friends-Dialog zum Versenden einer Freundschaftsanfrage

Darüber hinaus unterstützt der Friends-Dialog nur die Standardoptionen von `FB.ui()`:

Name	Typ	Beschreibung
`app_id`	`int`	numerische ID der Anwendung – nur beim direkten Aufruf des Dialogs per URL
`redirect_uri`	`string`	URL, zu der der Browser nach Beenden des Dialogs weitergeleitet wird – nur beim direkten Aufruf des Dialogs per URL
`display`	`string`	Anzeigemodus für den Dialog. Beim Aufruf über das JavaScript SDK wird dieser Parameter üblicherweise nicht verwendet.
`id`	`int`	numerische ID des Benutzers, dem eine Freundschaftsanfrage geschickt werden soll

Tabelle 4.16 Das Optionsobjekt zum Aufruf des Friends-Dialogs mit »FB.ui«

Ist der aktuelle Benutzer der Anwendung bereits mit dem Benutzer, dem eine Freundschaftsanfrage geschickt werden soll, befreundet, wird dies im Dialog folgendermaßen angezeigt:

Abbildung 4.11 Aufruf des Friends-Dialogs für einen bereits bestätigten Freund

Friends-Dialog mit URL-Aufruf öffnen

Um den Friends-Dialog mittels Redirect oder Direkt-Link zu öffnen, muss die URL *http://www.facebook.com/dialog/friends* im Browser des Benutzers geladen werden. Neben den GET-Parametern `app_id` und `redirect_uri` muss dabei lediglich `id` übergeben werden.

Der folgende URL-Aufruf öffnet den Feed-Dialog und teilt die URL *http://developers.facebook.com* auf der Pinnwand des aktuellen Benutzers:

```
http://www.facebook.com/dialog/friends
   ?app_id=123050457758183
   &redirect_uri=http://apps.mycompany.com/
   &id=609190863
```

Der Callback-Handler des Friends-Dialogs

Nachdem der Benutzer den Friends-Dialog bestätigt oder abgebrochen hat, wird der optionale Callback-Handler aufgerufen. Der Rückgabewert `result` enthält dabei das Boolean-Feld `action`, das den Wert 1 hat, wenn der Dialog tatsächlich bestätigt wurde:

```
{
   action: 1
}
```

4.4.3 OAuth-Dialog

Der bereits in Kapitel 2, »Authentifikation und Autorisierung«, ausführlich behandelte OAuth-Dialog unterscheidet sich grundsätzlich von den mittels `FB.ui()` aufgerufenen Dialogen. Der OAuth-Dialog kann nicht über `FB.ui()`, sondern lediglich über die SDK-Methode `FB.login()` (und natürlich über einen direktem URL-Aufruf) aufgerufen werden. Der Aufruf über `FB.login()` wurde bereits in Abschnitt 4.2.3, »FB.login«, der Aufruf per URL in Kapitel 2, »Authentifikation und Autorisierung«, detailliert beschrieben. An dieser Stelle möchte ich daher nur zwei einfache Beispiele anführen,

die zeigen, wie derselbe OAuth-Dialog über das JavaScript SDK und Direkt-URL aufgerufen werden kann.

Der SDK-Aufruf erfolgt dabei wie beschrieben über die Funktion `FB.login()`. Beim Aufruf der Funktion wird ein Callback-Handler registriert, und das Optionsobjekt enthält im Feld `scope` die notwendigen Berechtigungen:


```
FB.login(function(response) {
   console.debug(response);
   if (response.authResponse) {
      console.log('Login erfolgreich!');
   } else {
      console.log('Login abgebrochen!');
   }
}, {scope: 'email'});
```

Listing 4.18 Aufruf des OAuth-Dialogs mit »FB.login«

Der gleiche OAuth-Dialog kann auch durch Laden folgender URL erzeugt werden:

```
https://www.facebook.com/dialog/oauth/
   ?client_id=214728715257742
   &redirect_uri=http://apps.mycompany.com/
   &response_type=token
   &scope=email
```

Achtung: Anders als bei den übrigen Dialogen wird die Anwendungs-ID beim OAuth-Dialog im Parameter `client_id` anstelle von `app_id` übergeben!

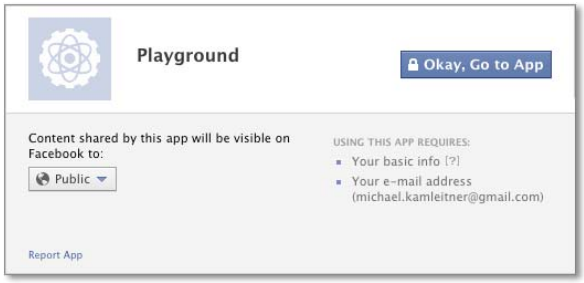


Abbildung 4.12 OAuth-Dialog zum Autorisieren einer Anwendung

4.4.4 Payment-Dialog

Seit 2010 bietet Facebook auf seiner Anwendungsplattform ein Mikro-Zahlungsmittel namens *Facebook Credits* an. Dieses Zahlungsmittel ermöglicht es Anwendungsentwicklern, kleinere Zahlungen für überwiegend virtuelle Güter von ihren Benutzern entgegenzunehmen. Eine detaillierte Darstellung von Facebook Credits

finden Sie in Kapitel 9, »Facebook Credits«, während ich in diesem Abschnitt lediglich die zugehörigen SDK-Funktionen beschreibe.

Wichtig: Bei der Abwicklung von Zahlungen mittels Facebook Credits und dem Payment-Dialog müssen Sie folgende Punkte beachten:

- Damit eine Facebook-Anwendung Facebook Credits benutzen kann, muss der entsprechende Abschnitt CREDITS in den Einstellungen der Anwendung vollständig und korrekt konfiguriert werden.
- Der Payment-Dialog steht derzeit ausschließlich in Facebook-Canvas-Anwendungen, also in Seiten-Tabs und Anwendungen unter der URL *http://apps.facebook.com/...*, zur Verfügung. Seit Kurzem werden Credits auch in mobilen Anwendungen unterstützt. Die Durchführung von Zahlungen auf Anwendungen, die auf einer externen Website laufen, wird hingegen derzeit explizit nicht unterstützt.

Der Payment-Dialog dient der Abwicklung von Transaktionen zwischen Benutzer und Applikation. Der Payment-Dialog ist von Facebook standardisiert und zeigt dem Benutzer (= Käufer) genaue Informationen über das zum Verkauf stehende Produkt an. Der Benutzer hat die Möglichkeit, das Produkt zu kaufen oder – falls notwendig – seinen Facebook-Credits-Kontostand aufzuladen.

Um den Payment-Dialog zu öffnen, wird `FB.ui()` mit dem Wert `pay` für die Option `method` aufgerufen:

```
var order_info = 'produkt-1';
var obj = {
            method: 'pay',
            order_info: order_info,
            purchase_type: 'item',
            dev_purchase_params: {'oscif': true}
          };
FB.ui(obj, function (data) {
   console.debug(data);
});
```

Listing 4.19 Aufruf des Payment-Dialogs mit »FB.ui«

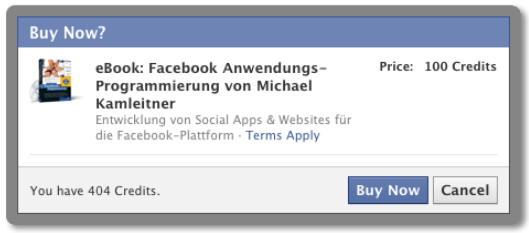


Abbildung 4.13 Der Payment-Dialog fordert den Benutzer zum Kauf eines Produkts auf.

Folgendes ist hier passiert:

1. Das JavaScript SDK bekommt von `FB.ui()` das Kommando zum Öffnen des Payment-Dialogs. Aus Sicherheitsgründen werden die Informationen über das zum Verkauf stehende Produkt jedoch nicht im JavaScript-Aufruf definiert – dies wäre für etwaige Angreifer leicht clientseitig zu fälschen!
2. Um die Produktinformationen anzeigen zu können, löst die `FB.ui()` zuerst einen AJAX-Aufruf auf Facebook.com aus. Facebook ruft daraufhin jene URL am Anwendungsserver auf, die der Anwendungsentwickler in den Einstellungen unter Credits Callback URL angegeben hat.
3. Dieser Credits Callback liefert Facebook nun – serverseitig – die Produktinformationen, also Preis, Name des Produkts, Thumbnail-Grafik etc. Daneben werden auch Daten zum Credits-Kontostand des aktuellen Benutzers übergeben, um prüfen zu können, ob der Benutzer ausreichend Credits für das zum Verkauf stehende Produkt besitzt oder ob er sein Konto zuvor aufladen muss.
4. Facebook liefert die Produktinformationen an `FB.ui()` zurück, woraufhin die Funktion nun den Payment-Dialog mitsamt Produktinformationen anzeigen kann.

Die genaue Funktionsweise des Credits-Callback-Handlers wird in Kapitel 9, »Facebook Credits«, näher beschrieben.

Beim Aufruf von `FB.ui()` legen einige weitere Parameter fest, wie der Payment-Dialog konfiguriert werden soll:

- `order_info` ist eine Zeichenkette oder ein JSON-Objekt, das das Produkt, das zum Verkauf steht, identifiziert. Im einfachsten Fall entspricht `order_info` der anwendungsinternen Produkt-ID.
- `purchase_type` legt den Typ des verkauften Produkts fest und muss derzeit immer auf `item` gesetzt (oder ganz weggelassen) werden. Damit wird festgelegt, dass die Transaktion einen einmaligen Verkauf eines bestimmten Produkts darstellt. Eine naheliegende – wenngleich von Facebook bisher nicht angekündigte – Erweiterung von Facebook Credits wäre etwa die Durchführung von wiederkehrenden Zahlungen bzw. Abonnements.
- `dev_purchase_params` ist ein optionales JSON-Objekt, über das weitere Einstellungen zum Payment-Dialog festgelegt werden können.

Name	Typ	Beschreibung
app_id	int	numerische ID der Anwendung – nur beim direkten Aufruf des Dialogs per URL
redirect_uri	string	URL, zu der der Browser nach Beenden des Dialogs weitergeleitet wird – nur beim direkten Aufruf des Dialogs per URL
display	string	Anzeigemodus für den Dialog. Beim Aufruf über das JavaScript SDK wird dieser Parameter üblicherweise nicht verwendet.
order_info	string oder Objekt	Merkmal zur Identifikation des zum Verkauf stehenden Produkts, etwa eine anwendungsinterne Produkt-ID
credits_purchase	boolean	Legt fest, ob der Payment-Dialog zum Einkaufen von Credits (true) oder zum Kauf von Produkten (false) verwendet werden soll.
purchase_type	string	Typ der Transaktion, derzeit ist ausschließlich der Wert item zulässig.
dev_purchas_params	Objekt	Objekt mit weiterführenden Feldern zur Konfiguration des Payment-Dialogs. Mit dem Feld oscif kann festgelegt werden, ob der Dialog die Umrechnung von Credits in Echtwährung anzeigt (false) oder nicht (true). Das Feld shortcut kann zur

Tabelle 4.17 Das Optionsobjekt zum Aufrufen des Payment-Dialogs mit »FB.ui«

Wenn der Benutzer im Payment-Dialog auf Buy Now klickt, gilt der Kauf als abgeschlossen. Im Hintergrund wird dazu wiederum der serverseitige Credits-Callback-Handler der Anwendung von Facebook aufgerufen. Ist die Transaktion erfolgreich abgeschlossen, übergibt FB.ui() den Rückgabewert im Objekt data folgendermaßen an den Callback-Handler:

```
{
   status: 'settled',
   order_id: 9006133015289
}
```

Das Feld status enthält den Wert settled, wenn die Transaktion erfolgreich durchgeführt wurde. Gleichzeitig wird in order_id die Facebook-ID der Transaktion zurückge-

liefert – diese sollte unbedingt in der Datenbank der Anwendung abgespeichert werden, um die Nachvollziehbarkeit der Transaktion zu gewährleisten.

Wird der Payment-Dialog abgebrochen, enthält das Rückgabeobjekt die Felder `error_code` und `error_message`, die über den Grund des Abbruchs Auskunft geben.

Payment-Dialog mit URL-Aufruf öffnen

Um den Payment-Dialog mittels Redirect oder Direkt-Link zu öffnen, muss die URL *http://www.facebook.com/dialog/pay* im Browser des Benutzers geladen werden. Neben den GET-Parametern `app_id` und `redirect_uri` werden die Parameter zur Konfiguration des Dialogs analog zur Methode `FB.ui()` übergeben.

Der folgende URL-Aufruf öffnet den Feed-Dialog und bietet das mit der ID `produkt-1` identifizierte Produkt zum Kauf an:

```
https://www.facebook.com/dialog/pay
    ?app_id=214728715257742
    &redirect_uri=http://apps.mycompany.com
    &credits_purchase=false
    &order_info=produkt-1
```

Payment-Dialog zum Kauf von Credits

Wird der Payment-Dialog mit der Option `credits_purchase=true` aufgerufen, wird im Dialog kein Produkt der Anwendung zum Kauf angeboten (ein eventuell gleichzeitig gesetzter Parameter `order_info` wird dabei auch ignoriert), sondern der Benutzer wird dazu aufgefordert, sein Credits-Guthaben mittels Kreditkarte, PayPal oder mit einem anderen unterstützten Zahlungsmittel aufzuladen. Der Payment-Dialog präsentiert sich in diesem Nutzungsszenario folgendermaßen:

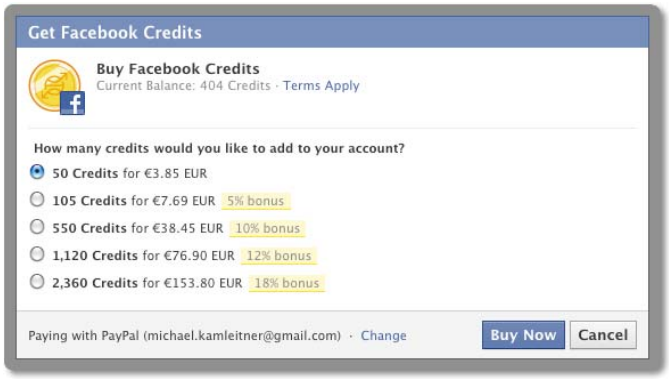


Abbildung 4.14 Payment-Dialog mit auf »true« gesetzter Option »credits_purchase«

4.4.5 Requests-Dialog

Der Requests-Dialog ermöglicht Benutzern das Versenden von Anwendungsanfragen an einen oder mehrere Freunde. Eine Anwendungsanfrage stellt dabei üblicherweise eine soziale Interaktion innerhalb einer bestimmten Anwendungsdomäne dar. In einem Multiplayer-Spiel wäre eine typische Anwendungsanfrage etwa die Einladung eines Freundes zu einem Spiel. In einem Fotowettbewerb würde ein Teilnehmer seine Freunde über eine Anwendungsanfrage zum Abgeben ihrer Stimme auffordern etc.

Um den Requests-Dialog mit `FB.ui()` zu öffnen, muss der Parameter `method` auf den Wert `apprequests` gesetzt werden. Gleichzeitig muss im Parameter `message` eine individuelle Anfragenachricht übergeben werden, die beim Empfänger der Anfrage angezeigt wird. Hierfür eignen sich Texte, die einen Hinweis auf die Art der Anfrage geben. Möchte ein Benutzer etwa Freunde zur Stimmabgabe in einem Fotowettbewerb auffordern, wäre ein geeigneter Text »Hilf mit und schenk mir deine Stimme beim Fotowettbewerb von ...«.

```
var options = {
      method: 'apprequests',
      message: 'Einladung zu meiner App'
   };
FB.ui(options, function(result) {
   console.debug(result);
});
```

Listing 4.20 Aufrufen des Requests-Dialogs mit »FB.ui«

Abbildung 4.15 Requests-Dialog zum Versenden von Anwendungsanfragen

Beim Aufruf von `FB.ui()` wird der Dialog geöffnet, und der Benutzer wird aufgefordert, mindestens einen und maximal 50 Empfänger aus der Liste seiner Freunde auszuwählen.

Hinweis: In den Browsern Internet Explorer 6.0 den 7.0 gestattet der Requests-Dialog lediglich die Auswahl von maximal 25 Freunden.

Ein Kick auf ANFRAGE SENDEN versendet die Anwendungsanfrage schließlich. Der anschließende Aufruf des optional registrierten Callback-Handlers liefert folgendes JSON-Objekt in `result` zurück:

```
{
   request: 354690814548024,
    to: [ 100001434672472,
          100001903705011 ]
}
```

Das zurückgelieferte Objekt enthält in `request` die Facebook-ID der versandten Anfrage, und im Array `to` die IDs jener Benutzer, an die die Anfrage versendet wurde.

Das neue Format Requests 2.0 Efficient

Der in diesem Buch beschriebene Aufbau des Rückgabeobjekts, das an den Callback-Handler des Requests-Dialogs übergeben wird, entspricht dem im Jahr 2011 eingeführten Format *Requests 2.0 Efficient*. Dieses neue, effizientere Rückgabeformat konnte während der Arbeiten an diesem Buch in den Migrationseinstellungen noch manuell deaktiviert werden. Da der vollständige Umstieg auf *Requests 2.0 Efficient* von Facebook jedoch für Januar 2012 angekündigt ist, wird auf eine Darstellung des alten Formats an dieser Stelle verzichtet.

Hinweis: Das Versenden von Anwendungsanfragen erfordert keine erweiterten Berechtigungen, sondern kann tatsächlich auch von Benutzern ausgeführt werden, die die aktuelle Anwendung noch gar nicht autorisiert haben. Der Requests-Dialog versendet niemals automatisch Anfragen, sondern erwartet immer die explizite Bestätigung durch den Benutzer (eine Ausnahme bilden Anfragen, die als *Frictionless Requests* versendet werden – mehr dazu erfahren Sie weiter unten in diesem Abschnitt).

Requests-Dialog mit URL-Aufruf öffnen

Um den Requests-Dialog mittels Redirect oder Direkt-Link zu öffnen, muss die URL *http://www.facebook.com/dialog/apprequests* im Browser des Benutzers geladen werden. Neben den GET-Parametern `app_id` und `redirect_uri` muss dabei mindestens der Parameter `message` übergeben werden.

Der folgende URL-Aufruf öffnet den Feed-Dialog und bietet das mit der ID `produkt-1` identifizierte Produkt zum Kauf an:

```
http://www.facebook.com/dialog/apprequests
    ?app_id=214728715257742
    &redirect_uri=http://apps.mycompany.com
    &message=Einladung+zu+meiner+App
```

Eingehende Anwendungsanfragen werden beim Empfänger an verschiedenen Stellen von Facebook.com angezeigt:

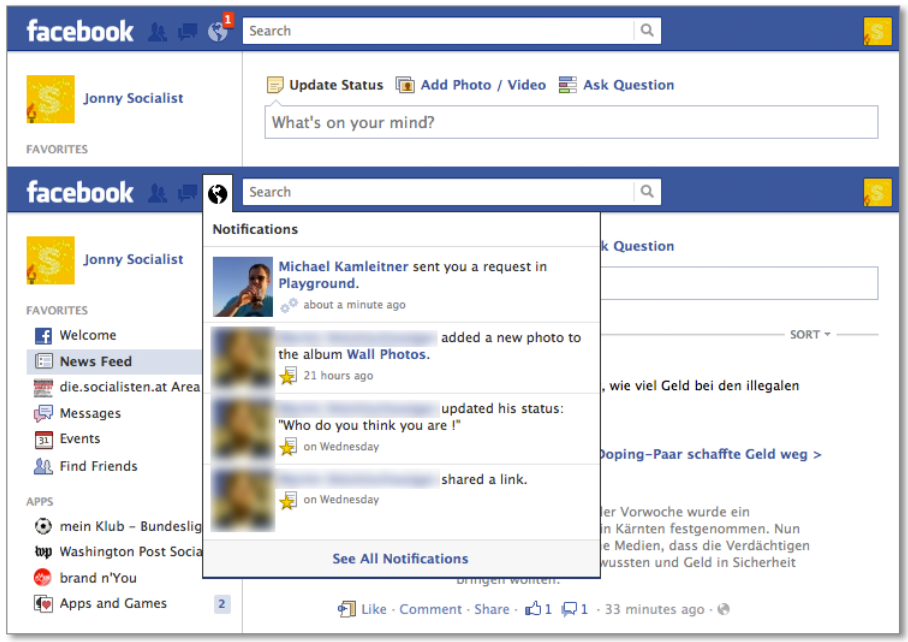


Abbildung 4.16 Anzeige von Anwendungsanfragen in den Benachrichtigungen

- Alle eingehenden Anwendungsanfragen werden einmalig im *Benachrichtigungsbereich* im Header von Facebook.com angezeigt. Dabei sorgt die rot hinterlegte Anzahl der ungelesenen Benachrichtigungen für zusätzliche Aufmerksamkeit. Anfragen im Benachrichtigungsbereich werden auch bei Benutzern dargestellt, die die Anwendung noch nicht installiert haben.
- Facebook gruppiert jene Anwendungen, die ein Benutzer zuletzt oder besonders häufig benutzt hat, im *Bookmark-Bereich* der Startseite von Facebook.com. Die Anzahl der ungelesenen Anfragen einer Anwendung wird neben dem Titel der Anwendung hellblau hinterlegt angezeigt. Diese Anzeige erfolgt nur, wenn der Empfänger die Anwendung bereits installiert hat und diese aufgrund des individuellen Nutzungsverhaltens tatsächlich im Bookmark-Bereich angezeigt wird.

- Ist ein Empfänger der Anfrage im Moment des Versendens gerade auf Facebook.com online, wird ein entsprechender Hinweis für einige Sekunden in der linken unteren Ecke des Browsers eingeblendet.
- Darüber hinaus bietet Facebook.com eine Übersichtsseite für Benachrichtigungen (*https://www.facebook.com/notifications*) bzw. auch speziell für Anwendungsanfragen (*https://www.facebook.com/reqs.php*) – diese Übersichtsseiten werden von Benutzern jedoch nur selten aufgerufen.

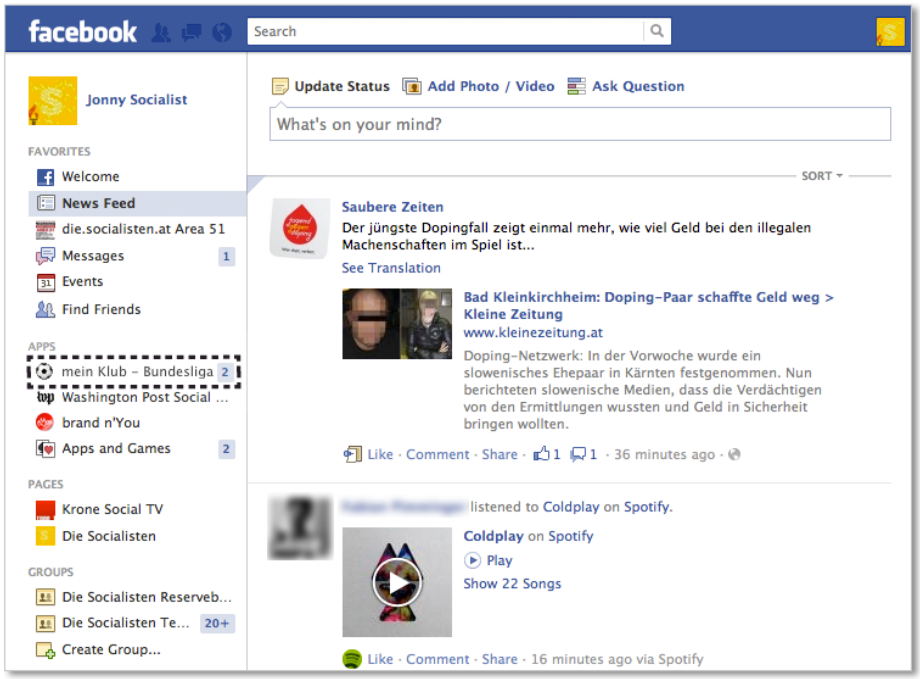


Abbildung 4.17 Anzeige von Anwendungsbenachrichtigungen im Bookmark-Bereich beliebter Anwendungen

Mit folgenden Optionsparametern kann der Requests-Dialog detaillierter konfiguriert werden:

Name	Typ	Beschreibung
`app_id`	`int`	numerische ID der Anwendung – nur beim direkten Aufruf des Dialogs per URL
`redirect_uri`	`string`	URL, zu der der Browser nach Beenden des Dialogs weitergeleitet wird – nur beim direkten Aufruf des Dialogs per URL

Tabelle 4.18 Das Optionsobjekt zum Aufrufen des Requests-Dialogs mit »FB.ui«

Name	Typ	Beschreibung
`display`	`string`	Anzeigemodus für den Dialog. Beim Aufruf über das JavaScript SDK wird dieser Parameter üblicherweise nicht verwendet.
`message`	`string`	Anfragenachricht, die erklärt, zu welcher Anwendung eingeladen wird
`title`	`string`	Titel, der in der Kopfzeile des Requests-Dialogs angezeigt wird. Ohne die Angabe dieses Parameters lautet der Text in der Kopfzeile SELECT FRIENDS FOR APPNAME.
`to`	`string`	Kommaseparierte Liste an Benutzer-IDs, an die die Einladung versendet werden soll. Ist dieser Parameter gesetzt, wird die Auswahl der Empfänger im Dialog ausgeblendet.
`exclude_ids`	`string`	Kommaseparierte Liste an Benutzer-IDs, die nicht in der Liste zur Auswahl der Empfänger enthalten sein sollen. Dieser Parameter ist praktisch, um beim Versenden von allgemeinen Einladungen zu einer Anwendung eine bestimmte Gruppe an Benutzern auszunehmen.
`max_recipients`	`int`	Mit dieser Option kann festgelegt werden, wie viele Freunde ein Benutzer höchstens als Empfänger auswählen kann. Der Maximalwert wird von Facebook jedoch auf 50 (bzw. 25 bei Internet Explorer 6.0 und 7.0) festgelegt.
`data`	`string`	ein optionales Datenfeld, das Informationen zum Tracking der Anfrage enthalten kann
`filters`	Array	Während der Requests-Dialog per Default die Auswahl aller Freunde zulässt (`['all']`), kann die Liste der auswählbaren Freunde mit dieser Option auf Applikationsbenutzer (`['app_users']`) oder Freunde, die die Applikation noch nicht installiert haben (`[app_non_users']`), eingeschränkt werden. Darüber hinaus können mit der Angabe eines JSON-Objekts Freundesgruppen definiert werden, nach denen der Benutzer seine Freunde filtern kann. Das JSON-Objekt muss dabei ein Array von Feldpaaren ^ und `id` sein: `[{ name: 'Beste Freunde', user_ids: [ 532617296,633616172] }]`

Tabelle 4.18 Das Optionsobjekt zum Aufrufen des Requests-Dialogs mit »FB.ui« (Forts.)

Werden mit dem Parameter `to` einer oder mehrere Empfänger vorausgewählt, stellt das SDK den Dialog ohne die Auswahlliste für Empfänger dar. Unter To: werden die Namen der vorselektierten Freunde angezeigt:

Abbildung 4.18 Requests-Dialog mit zwei vorausgewählten Empfängern

Das folgende Code-Beispiel demonstriert die Definition einer Benutzergruppe mit dem Parameter `filters`:

```
var options = {
      method: 'apprequests',
      title: 'Freunde einladen ohne Vorauswahl',
      message: 'Einladung zu meiner App',
      filters: [{ name: 'Beste Freunde',
                  user_ids: [532617296,633616172] }]
};
FB.ui(options, function(result) {
   console.debug(result);
});
```

Listing 4.21 Aufruf des Requests-Dialogs mit einer definierten Benutzergruppe

Der Requests-Dialog wird mit dieser Konfiguration folgendermaßen angezeigt:

Abbildung 4.19 Requests-Dialog mit gesetztem »filters«-Parameter

Frictionless Requests

Frictionless Requests sind eine relativ neue Funktion des JavaScript SDKs, die es Benutzern erlaubt, wiederkehrende Anwendungsanfragen ohne manuelle Bestätigung über den Requests-Dialog zu versenden. Der Anfragevorgang verläuft damit aus Sicht des Benutzers reibungsloser (*frictionless*), was eine höhere Interaktionsrate ermöglichen soll.

Frictionless Requests fügen eine zusätzliche Checkbox im Requests-Dialog hinzu, die standardmäßig angekreuzt ist, und den Benutzer zur Aktivierung der Funktion auffordert:

Don't ask again before sending requests to Jonny from this app.

Wichtig: Die Aktivierung von Frictionless Requests wird immer nur für jene Empfänger aktiviert, an die der Benutzer die Anwendungsanfrage versendet! Wird zu einem späteren Zeitpunkt ein Requests-Dialog mittels `FB.ui()` angestoßen, bei dem für alle im Parameter `to` gesetzten Empfänger Frictionless Requests bereits aktiviert wurden, *öffnet das SDK den Dialog nicht*, sondern versendet die Anwendungsanfrage ohne weitere Bestätigung.

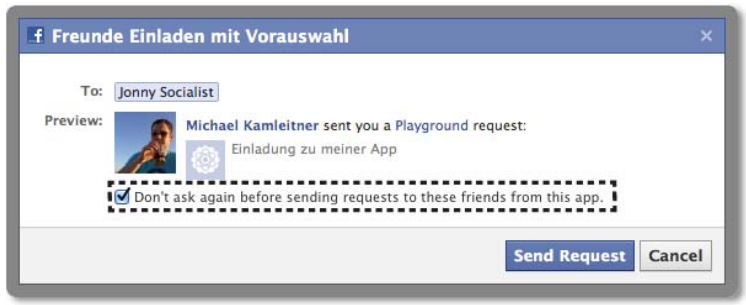


Abbildung 4.20 Requests-Dialog mit aktiviertem »Frictionless Sharing«

Wichtig: Frictionless Requests müssen beim Initialisieren des SDKs explizit aktiviert werden. Die entsprechende Checkbox im Requests-Dialog wird nur angezeigt, wenn das Optionsfeld `frictionlessRequests` beim Aufruf von `FB.init()` auf `true` gesetzt ist:

```
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
      channelUrl : '//apps.mycompany.com/channel.php',
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true, // XFBML parsen
```

```
      frictionlessRequests : true,
});
};
```

Listing 4.22 Initialisieren des SDKs mit aktiviertem »Frictionless Sharing«

Verarbeiten von Anwendungsanfragen

Akzeptiert ein Empfänger eine eingehende Anwendungsanfrage, indem er die Benachrichtigung anklickt oder die Anfrage mittels Klick auf ACCEPT bestätigt, leitet Facebook den Browser des Benutzers auf die Canvas-URL der Anwendung weiter, wobei die ID der Anfrage im HTTP-GET-Parameter `request_ids` übergeben wird:

```
https://apps.facebook.com/APPNAMESPACE/?request_ids=RequestID
```

Wurden von derselben Anwendung mehrere Anfragen an einen Benutzer versendet, werden die IDs aller Anfragen als kommaseparierte Liste übergeben:

```
https://apps.facebook.com/APPNAMESPACE/
?request_ids=RequestID-1,Request-ID-2,...
```

Wichtig: Handelt es sich bei der Anfrage um eine allgemeine Einladung zur Applikation, muss der Entwickler der Anwendung normalerweise nichts weiter tun – die Weiterleitung auf die Canvas-URL lädt üblicherweise die Startseite der Anwendung. Für jede weitere Verarbeitung von Anwendungsanfragen ist jedoch der Entwickler der Anwendung allein verantwortlich!

Serverseitig können die Anwendungsanfragen anhand der Anfrage-IDs über die Graph API abgefragt werden. Da für diese Zugriffe ein Anwendungs-Access-Token verwendet werden darf, ist es hierzu nicht notwendig, dass der Empfänger der Anfrage die Applikation bereits autorisiert hat! Das folgende Code-Beispiel sollte beim Aufruf der Startseite der Applikation (`https://apps.facebook.com/APPNAMESPACE/`) ausgeführt werden:

```
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', 'b93bff303c6199d959ad84659bf3f7bd');
define('SITE_URL', 'http://apps.mycompany.com');

// Verarbeiten der eingehenden Anwendungsanfragen
// Anhand der übergebenen Anfrage-IDs ist zu erkennen, ob
   ein Benutzer über eine Anfrage zur Anwendung gelangt ist
if (isset($_REQUEST["request_ids"])) {
   // Trennen der Anfrage-IDs, falls mehrere vorhanden
   print "<pre>";
```

```
    $request_ids = split(",", $_REQUEST["request_ids"]);
    print "Request-IDs:<br/><br/>";
    print_r($request_ids);

    foreach ($request_ids as $id) {
       print "<br/>ID ".$id."<br/><br/>"
       $request = json_decode(curl("https://graph.facebook.com/".
          $id."?access_token=".
          get_app_accesstoken(APP_ID, APP_SECRET))));
       }
       print "</pre>";
}
```

Listing 4.23 Verarbeiten eingehender Anwendungsanfragen

Was ist bei der serverseitigen Verarbeitung von eingehenden Anfragen zu beachten?

- Facebook übergibt die ID oder die IDs der Anfragen im GET-Parameter `request_ids`.
- Für jede übergebene Request-ID können die genauen Informationen zur Anfrage mit der Graph API abgefragt werden. Der API-Endpunkt ist dabei die Anfrage-ID selbst (`https://graph.facebook.com/AnfrageID`) und kann mit dem Anwendungs-Access-Token abgefragt werden, das mit der Hilfsfunktion `get_app_accesstoken()` erzeugt wird.

Die einzelnen Anfrageobjekte enthalten zahlreiche Informationen über die Anfrage, den Sender und die Anwendung, über die die Anfrage versendet wurde:

```
{
   // ID der Anfrage
   id: "354690814548024",
   // Anwendung, über die die Anfrage versendet wurde
   application: {
      name: 'Playground',
      canvas_name: 'galileo-playground',
      namespace: 'galileo-playground',
      id: 214728715257742
   },
   // Benutzer, der die Anfrage versendet hat
   from: {
   name: 'Michael Kamleitner',
   id: 609190863
   }
```

```
   // Zeitstempel & Nachricht der Anfrage
   message: 'Einladung zu meiner App',
   created_time: '2012-01-05T18:54:00+0000'
}
```

Listing 4.24 JSON-Array eines Anfrageobjekts

Damit lassen sich schon gezielte Einlademechanismen realisieren: Soll eine Spieleanwendung etwa ermöglichen, dass ein Benutzer einen Freund zu einem Spiel einlädt, genügt es, beim Empfänger der Anfrage die Benutzer-ID des Absenders zu ermitteln und das Spiel zu eröffnen.

In manchen Fällen kann es aber notwendig sein, zusätzliche Daten mit einer Anwendungsanfrage zu verknüpfen. Ein Benutzer, der an einem Fotowettbewerb mit mehreren Einreichungen teilnimmt, sollte etwa in der Lage sein, seine Freunde zur Stimmabgabe zu einem bestimmten Foto einzuladen. Dazu eignet sich der Aufruf des Requests-Dialogs mit dem optionalen Feld `data`, das zusätzliche Tracking-Daten mit der Anfrage verknüpft:

```
var options = {
      method: 'apprequests',
      message: 'Einladung zu meiner App',
      data: { photo-id: 4711 }
   };
FB.ui(options);
```

`data` kann dabei einen einfachen String (etwa die ID des Fotos, zu dem eingeladen werden soll) oder auch ein komplexeres JSON-Objekt enthalten. Wird die Anfrage für den Empfänger anhand ihrer ID abgefragt, enthält das von der Graph API zurückgelieferte Objekt auch diese Tracking-Daten:

```
{
   id: "354690814548024",
   application: {
      name: 'Playground',
      canvas_name: 'galileo-playground',
      namespace: 'galileo-playground',
      id: 214728715257742
   },
   from: {
   name: 'Michael Kamleitner',
   id: 609190863
   }
```

```
    // Mit der Anfrage verknüpfte Tracking-Daten
    data: '{"photo-id":4711}',
    message: 'Einladung zu meiner App',
    created_time: '2012-01-05T18:54:00+0000'
}
```

Listing 4.25 JSON-Array eines Anfrageobjekts mit »data«-Feld

Mit der zusätzlichen Information im Feld `data` kann die Anwendung nun dafür sorgen, dass der Benutzer zum gewünschten Foto weitergeleitet wird.

Löschen von Anwendungsanfragen

Eine oft missverstandene Frage ist die Behandlung von Anwendungsanfragen, nachdem diese vom Empfänger akzeptiert wurden und der Empfänger damit auf der Applikation gelandet ist. Facebook entfernt angenommene Anwendungsanfragen nicht selbstständig, sondern erwartet vom Anwendungsentwickler, dass dieser für das Löschen einer bereits akzeptierten Anfrage sorgt. Dies ist vor allem wichtig, um dem eingeladenen Benutzer Verwirrung zu ersparen, da auch bereits akzeptierte Anfragen noch im Dropdown-Menü der Benachrichtigungen angezeigt werden.

Anwendungsanfragen können mit einer DELETE-Anfrage an die Graph API gelöscht werden. Der Endpunkt wird dabei aus der Request-ID und der Benutzer-ID des Empfängers, verbunden mit einem Unterstrich, gebildet:

```
https://graph.facebook.com/RequestID_BenutzerID
```

Darüber hinaus ist zum Löschen der Anwendungsanfrage ein Access Token des Empfängers notwendig – mit der PHP-Hilfsfunktion `curl()` würde der Aufruf also etwa so aussehen:

```
curl('https://graph.facebook.com/354690814548024_100001903705011',
false, 'DELETE');
```

4.4.6 »Send«-Dialog

Der SEND-Dialog ermöglicht es Benutzern, Inhalte und Links als Nachricht an bestimmte Freunde, Facebook-Gruppen oder per E-Mail zu versenden, anstatt sie öffentlich auf ihrer Pinnwand zu teilen. Damit hat der SEND-Dialog eine ähnliche Funktion wie das neuere Social Plugin *Send-Button*, das in Abschnitt 6.3, »Der ›Send‹-Button«, beschrieben wird. An Benutzer versendete Nachrichten werden in der Facebook-Mailbox der Empfänger und als Benachrichtigung in der Kopfzeile von Facebook.com angezeigt.

Facebook empfiehlt, den SEND-Dialog ausschließlich für persönliche Kommunikation zwischen zwei Benutzern (oder in einer Facebook-Gruppe) zu nutzen, nicht jedoch für anwendungsbezogene Anfragen oder Einladungen zu Anwendungen – diese Einsatzmöglichkeiten sollen dem Requests-Dialog vorbehalten bleiben.

Um den SEND-Dialog aufzurufen, muss `FB.ui()` mit dem Wert `send` in der Option `method` aufgerufen werden. Zusätzlich wird mit dem Parameter `link` die URL einer Webseite angegeben, die mit der Nachricht versendet werden soll:

```
var options = {
      method: 'send',
      link: 'http://die.socialisten.at/'
   };
FB.ui(options);
```

Hinweis: Der SEND-Dialog liefert keinen Rückgabewert zurück, daher kann die Definition eines Callback-Handlers entfallen.

Der SEND-Dialog wird folgendermaßen dargestellt:

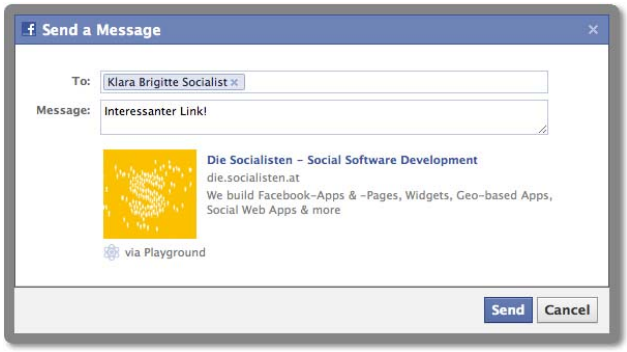


Abbildung 4.21 »Send«-Dialog zum Versenden privater Nachrichten

Wichtig: Auch wenn die offizielle Dokumentation dies nicht explizit darstellt, dürfte die Angabe einer URL mittels `link` verpflichtend sein. Obwohl der SEND-Dialog auch ohne eine angehängte URL zu funktionieren scheint, kommen die auf diese Weise versendeten Nachrichten nicht beim Empfänger an!

»Send«-Dialog mit URL-Aufruf öffnen

Um den SEND-Dialog mittels Redirect oder Direkt-Link zu öffnen, muss die URL *http://www.facebook.com/dialog/send* im Browser des Benutzers geladen werden. Neben den GET-Parametern `app_id` und `redirect_uri` muss dabei mindestens der Parameter `link` übergeben werden.

Der folgende URL-Aufruf öffnet den SEND-Dialog:

```
http://www.facebook.com/dialog/send
    ?app_id=214728715257742
    &redirect_uri=http://apps.mycompany.com
    &link=http://die.socialisten.at
```

Name	Typ	Beschreibung
app_id	int	numerische ID der Anwendung – nur beim direkten Aufruf des Dialogs per URL
redirect_uri	string	URL, zu der der Browser nach Beenden des Dialogs weitergeleitet wird – nur beim direkten Aufruf des Dialogs per URL
display	string	Anzeigemodus für den Dialog. Beim Aufruf über das JavaScript SDK wird dieser Parameter üblicherweise nicht verwendet.
to	string	Kommaseparierte Liste an Benutzer-IDs, an die die Nachricht versendet werden soll. Neben den so vorausgewählten Empfängern kann der Benutzer weitere Freunde, Gruppen oder E-Mail-Adressen eingeben.
link	string	URL zur Webseite, die mit der Nachricht versendet werden soll
picture	string	URL zu einer Thumbnail-Grafik, die mit der Nachricht versendet werden soll. Wird diese Option nicht angegeben, versucht Facebook automatisch, ein Thumbnail von der unter link angegebenen Webseite zu extrahieren.
name	string	Linktext der URL. Wird diese Option nicht angegeben, versucht Facebook automatisch, ein Thumbnail von der unter link angegebenen Webseite zu extrahieren.
description	string	Beschreibungstext der URL. Wird diese Option nicht angegeben, versucht Facebook automatisch, ein Thumbnail von der unter link angegebenen Webseite zu extrahieren.

Tabelle 4.19 Das Optionsobjekt zum Aufruf des »Send«-Dialogs mit »FB.ui«

4.4.7 Tab-Dialog

Der Tab-Dialog bietet Administratoren von Facebook-Seiten eine einfache Möglichkeit, eine bestehende Anwendung als Tab auf ihren Seiten zu integrieren. Während dies üblicherweise den Aufruf einer speziellen URL wie in Abschnitt 1.5.1, »Hello Facebook! als Tab-Anwendung«, erfordert, bietet der Tab-Dialog also eine wesentlich benutzerfreundlichere Methode.

Um den Tab-Dialog aufzurufen, müssen Sie `FB.ui()` mit dem Wert `pagetab` in der Option `method` aufrufen. Die Angabe einer Anwendungs-ID entfällt, da diese automatisch jener ID entspricht, mit der das SDK in `FB.init()` initialisiert wurde:

```
var options = {
      method: 'pagetab'
   };
FB.ui(options);
```

Hinweis: Der Tab-Dialog liefert keinen Rückgabewert zurück, daher kann die Definition eines Callback-Handlers entfallen.

Tab-Dialog mit URL-Aufruf öffnen

Um den Tab-Dialog mittels Redirect oder Direkt-Link zu öffnen, muss die URL *http://www.facebook.com/dialog/pagetab* im Browser des Benutzers geladen werden. Dabei müssen die GET-Parameter `app_id` und `redirect_uri` übergeben werden.

Der folgende URL-Aufruf öffnet den Tab-Dialog:

```
http://www.facebook.com/dialog/pagetab
   ?app_id=214728715257742
   &redirect_uri=http://apps.mycompany.com
```

Der Tab-Dialog wird folgendermaßen dargestellt:

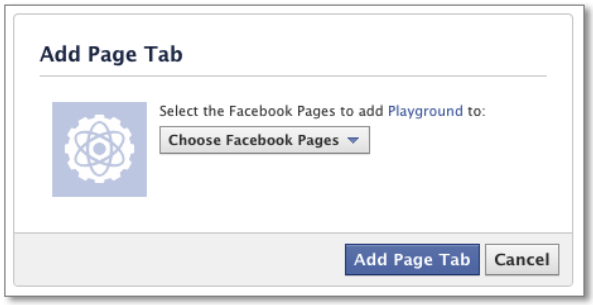


Abbildung 4.22 Der Tab-Dialog zum Installieren von Anwendungs-Tabs auf Seiten

4.5 Laden von Social Plugins

Das JavaScript SDK ist üblicherweise auch für das Laden und Anzeigen der in Kapitel 6, »Social Plugins«, beschriebenen Social Plugins zuständig. Während diese Plugins grundsätzlich auch ohne SDK, über herkömmliche HTML-iFrames, eingebunden werden können, werden doch einige neuere Parameter nur bei der Einbindung per SDK unterstützt.

Der Vorgang zur Anzeige von Social Plugins ist dabei der gleiche wie im Hello-World!-Beispiel in Abschnitt 4.1.1, »Hello World mit dem Facebook JavaScript SDK«, beschrieben: Nach dem Laden der Seite werden alle im HTML-Dokument gefundenen XFBML-Tags automatisch durch die eigentlichen Inhalte der Plugins ersetzt.

Entwickler, die Wert auf syntaktisch valide HTML-Dokumente legen, scheiterten lange an der Invalidität der XFBML-Tags. Als Lösung bietet Facebook daher seit einiger Zeit die alternative Einbindung von Social Plugins als HTML5-kompatible Tags an. Das Parsen und Ersetzen der HTML5-Tags durch die Plugin-Inhalte funktioniert dabei genauso wie in der XFBML-Variante.

Das folgende Beispiel demonstriert die drei möglichen Varianten zur Einbindung von Social Plugins:

```
<DOCTYPE html>
<html lang="de-de" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:fb="http://ogp.me/ns/fb#">
<head>
   <meta charset="utf-8">
   <title>Hello Facebook JS SDK!</title>
</head>
<body>
<p>Like-Button per iFrame (ohne SDK):</p>
<iframe src="//www.facebook.com/plugins/like.php?href=http%3A%2F%2F
  www.facebook.com&amp;send=false&amp;layout=standard&amp;width=450&amp;
  show_faces=true&amp;action=like&amp;colorscheme=light&amp;font&amp;
  height=80" scrolling="no" frameborder="0" style="border:none;
  overflow:hidden; width:450px; height:80px;" allowTransparency="true">
</iframe>

<p>Like-Button per XFBML-Tag (erfordert SDK):</p>

<fb:like href="http://www.facebook.com" send="false" width="450" show_faces=
"true"></fb:like>

<p>Like-Button per HTML5-Tag (erfordert SDK):</p>
```

```
<div class="fb-like" data-href="http://www.facebook.com" data-send=
"false" data-width="450" data-show-faces="true"></div>

<div id="fb-root"></div>
<script>
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
                   // Anwendungs-ID
      channelUrl : '//apps.mycompany.com/channel.php',
                   // Channel-Datei
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
</script>
</body>
</html>
```

Listing 4.26 Drei Varianten der Einbindung von Social Plugins

Der Beispiel-Code erzeugt drei Varianten des bekanntesten Social Plugins, dem LIKE-Button. Die drei Varianten werden dabei folgendermaßen erzeugt:

- Der erste LIKE-Button wird mittels iFrame-Element eingebunden. Diese Variante funktioniert als einzige auch ohne eingebundenes JavaScript SDK.
- Der zweite LIKE-Button wird als XFBML-Tag `<fb:like>` eingebunden. Das Tag wird zur Laufzeit des SDKs durch einen iFrame ersetzt.
- Der dritte LIKE-Button wird als HTML5-Tag `<div class="fb-like">` eingebunden. Das SDK durchsucht das HTML-Dokument zur Laufzeit nach allen Elementen, die die Klasse `fb-like` besitzen, und ersetzt diese durch iFrame-Elemente.

Die Einbindung per iFrame sollte nur in Ausnahmefällen zum Einsatz kommen, da sie im Vergleich zu XFBML- und HTML5 weniger Optionen zur Anpassung des Plugins bietet und beim Einsatz mehrerer Plugins auf einer Seite weniger performant ist. Ein

mögliches Szenario für die iFrame-Lösung könnte etwa die Erweiterung einer bestehenden Website um Social Plugins sein, bei der sich das Hinzufügen des JavaScript SDKs als schwierig erweist.

Die XFBML- und HTML5-Lösungen sind derzeit hinsichtlich ihres Funktionsumfangs gleichgestellt – um zukunftssicheren, validen Code zu erzeugen, empfiehlt sich aber schon heute die konsequente Verwendung der HTML5-Variante.

Detaillierte Informationen zur Verwendung der einzelnen Social Plugins finden Sie in Kapitel 6, »Social Plugins«.

Abbildung 4.23 Drei Varianten der Einbindung von Social Plugins

4.6 Event-Handling

Das JavaScript SDK bietet ein einfaches Event-Modell, mit dessen Hilfe Anwendungsentwickler Kenntnis von bestimmten Vorgängen (*Events*) erhalten und gegebenenfalls eigenen Code ausführen können. Dazu müssen die gewünschten Events lediglich »abonniert« und mit einem eigenen Event-Handler verknüpft werden. Der Event-Handler enthält dabei jenen Code, der bei Auslöser des Events ausgeführt werden soll.

4.6.1 FB.Event.subscribe

Die SDK-Funktion `FB.Event.subscribe()` dient dem »Abonnieren« von Events und wird folgendermaßen aufgerufen:

```
FB.Event.subscribe('event_name, handler);
```

Name	Typ	Beschreibung
event_name	string	Name des zu abonnierenden Events
handler	function	Event-Handler als JavaScript-Funktion

Tabelle 4.20 Parameter der Funktion »FB.Event.subscribe«

Während der erste Parameter den Namen des gewünschten Events als String enthält, verweist der zweite Parameter auf den Funktionsnamen des Event-Handlers. Um etwa das Event `edge.create` zu abonnieren, ist folgender Code geeignet:

```
<script>
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
      channelUrl : '//apps.mycompany.com/channel.php',
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
    // Events abonnieren ...
    FB.Event.subscribe('edge.create', function(response) {
      alert('Event: edge.create, response: '+response);
    });
};
...
```

Listing 4.27 Beispiel-Code zum Abonnieren eines Events

Im Code-Beispiel wird das SDK geladen und mit `FB.init()` initialisiert. Danach wird das Event `edge.create` abonniert und mit einem Event-Handler versehen. In dieser inline definierten Funktion werden über `alert()` die Auslösung des Events und der in `response` übergebene Antwortparameter ausgegeben.

Alternativ kann die Angabe des Event-Handlers auch als benannte Funktion erfolgen:

```
FB.Event.subscribe('edge.create', edgeCreateHandler);

var edgeCreateHandler = function(response) {
   alert('Event: edge.create, response: '+response);
};
```

Listing 4.28 Benannte Funktion zum Abonnieren eines Events

Hinweis: Die Funktion `FB.Event.subscribe()` kann mehrfach für dasselbe Event aufgerufen werden – somit ist die Definition von mehreren Event-Handlern für dasselbe Event möglich!

Wichtig: Das Abonnieren von Events muss nach dem Laden des JavaScript SDKs erfolgen, da die erforderliche Funktion `FB.Event.subscribe()` davor nicht verfügbar ist!

4.6.2 FB.Event.unsubscribe

Ist ein bestehendes Event-Abonnement nicht mehr gewünscht, kann es mittels `FB.Event.unsubscribe()` wieder entfernt werden. Als Parameter müssen wieder der Event-Name und der Handler angegeben werden:

```
FB.Event.unsubscribe('event_name, handler);
```

Name	Typ	Beschreibung
`event_name`	`string`	Name des zu löschenden Event-Abonnements
`handler`	`function`	Event-Handler als JavaScript-Funktion

Tabelle 4.21 Parameter der Funktion »FB.Event.unsubscribe«

Da `FB.Event.unsubscribe()`die namentliche Angabe des zu löschenden Event-Handlers erfordert, funktioniert dies nur, wenn der Event-Handler beim Aufruf von `FB.Event.subscribe()`als benannte Funktion definiert wurde:

```
FB.Event.subscribe('edge.create', edgeCreateHandler);

var edgeCreateHandler = function(response) {
   alert('Event: edge.create, response: '+response);
};
FB.Event.unsubscribe('edge.create', edgeCreateHandler);
```

4.6.3 Verfügbare Events

Event auth.login

Dieses Event wird ausgelöst, wenn sich der Benutzer mit seinem Facebook-Konto an der Applikation anmeldet. Das `response`-Objekt dieses Events ist ein JSON-Objekt mit folgendem Aufbau:

```
{
   status: "",           // Aktueller Status der Session
   authResponse: {       // Informationen zur aktuellen Session
      userID: "",        // ID des aktuellen Benutzers
      signedRequest: "", // Aktueller Signed Request
      expiresIn: "",     // Zeitstempel, an dem die Session abläuft
      accessToken: "",   // Access Token des aktuellen Benutzers
  }
}
```

Listing 4.29 Response-Objekt des Events »auth.login«

Event auth.authResponseChange

Dieses Event wird ausgelöst, wenn sich der `authResponse` ändert, also wenn ein Benutzer an- oder abgemeldet oder die Session aktualisiert wird. Das `response`-Objekt ist genauso aufgebaut wie bei `auth.login`.

Event auth.statusChange

Dieses Event wird ausgelöst, wenn sich der Login-Status ändert. Das `response`-Objekt ist genauso aufgebaut wie bei `auth.login`.

Event auth.logout

Dieses Event wird ausgelöst, wenn sich der aktuelle Benutzer von Facebook abgemeldet hat. Das `response`-Objekt ist folgendermaßen aufgebaut:

```
{
   status: "",           // Aktueller Status der Session
}
```

Event auth.prompt

Dieses Event wird ausgelöst, wenn ein Benutzer aufgefordert wird, sich an Facebook anzumelden, nachdem er einen LIKE-Button angeklickt hat. Das `response`-Objekt enthält die URL des LIKE-Buttons, für den das Event ausgelöst wurde.

```
"http://facebook.com/diesocialisten"
```

Event xfbml.render

Dieses Event wird ausgelöst, wenn ein Aufruf der SDK-Funktion `FB.XFBML.parse()` abgeschlossen ist. Diese Funktion durchsucht das HTML-Dokument nach XFBML-Tags und ersetzt sie durch die entsprechenden Inhalte.

Event edge.create

Dieses Event wird ausgelöst, wenn ein an Facebook angemeldeter Benutzer einen LIKE-Button angeklickt hat und damit eine Verbindung (*edge* = Kante) zwischen Benutzer und Seite im sozialen Graphen erstellt wurde. Das `response`-Objekt enthält die URL des LIKE-Buttons, für den das Event ausgelöst wurde.

```
"http://facebook.com/diesocialisten"
```

Event edge.remove

Dieses Event wird ausgelöst, wenn ein an Facebook angemeldeter Benutzer einen bereits angeklickten LIKE-Button noch einmal angeklickt hat und damit eine Verbindung zwischen Benutzer und Seite im sozialen Graphen entfernt wurde (*Unlike*). Das `response`-Objekt enthält die URL des LIKE-Buttons, für den das Event ausgelöst wurde.

```
"http://facebook.com/diesocialisten"
```

Event message.send

Dieses Event wird ausgelöst, wenn ein Benutzer eine private Nachricht über einen per Social Plugin eingebundenen SEND-Button (`<fb:send>`) veröffentlicht hat. Das `response`-Objekt enthält die URL, die versendet wurde, als String.

```
"http://developers.facebook.com"
```

Event comment.create

Dieses Event wird ausgelöst, wenn ein Benutzer einen Kommentar in einer per Social Plugin eingebundenen Kommentarbox (`<fb:comments>`) veröffentlicht hat. Das `response`-Objekt ist folgendermaßen aufgebaut:

```
{
   href: "",           // URL, auf der das Kommentar-Plugin
                       // eingebunden wurde
   commentID: "",      // Die ID des neuen Kommentars
}
```

Listing 4.30 Response-Objekt des Events »comment.create«

Event comment.remove

Dieses Event wird ausgelöst, wenn ein Benutzer einen Kommentar in einer per Social Plugin eingebundenen Kommentarbox (`<fb:comments>`) gelöscht hat. Das `response`-Objekt ist folgendermaßen aufgebaut:

```
{
   href: "",           // URL, auf der das Kommentar-Plugin
                       // eingebunden wurde
   commentID: "",      // Die ID des gelöschten Kommentars
}
```

Listing 4.31 Response-Objekt des Events »comment.remove«

Kapitel 5
Die Facebook Query Language (FQL)

In diesem Kapitel erhalten Sie einen Überblick über die Abfragesprache Facebook Query Language (FQL) und die verfügbaren FQL-Tabellen.

Die Facebook Query Language ist eine an die weithin bekannte Datenbank-Abfragesprache SQL (Structured Query Language) angelehnte Schnittstelle, die ähnlich wie die Graph API Zugriff auf die Objekte des sozialen Graphen ermöglicht. Als Vorgänger der Graph API sind Zugriffe mittels FQL heute nur mehr in besonderen Fällen notwendig und empfehlenswert.

Die *Facebook Query Language* (FQL) wurde von Facebook seit dem Start der Anwendungsplattform im Jahr 2007 als alternative Möglichkeit zum direkten Zugriff auf Objektdaten der Plattform positioniert. Bevor 2010 die Graph API umfassenden Zugriff auf die Objekte des sozialen Graphen erlaubte, mussten Entwickler häufig zwischen der damaligen HTTP-REST-API und FQL wechseln, um die gewünschten Daten abfragen zu können. Beide Technologien erlaubten leider jeweils nur Zugriff auf einen Teil der Objektdaten.

Mit der Graph API hat Facebook große Anstrengungen unternommen, die Abbildung des sozialen Graphen in einer einzigen API zu vereinheitlichen. Heute können *fast alle Anwendungsfälle* mit der Graph API abgebildet werden, lediglich einige wenige Spezialfälle erfordern noch den Einsatz von FQL.

FQL orientiert sich an der bekannten Datenbank-Abfragesprache SQL (*Structured Query Language*) und folgt demselben Aufbau wie diese gebräuchliche Sprache:

```
SELECT Feldname,... FROM Tabellenname WHERE Bedingung,...
```

FQL-Abfragen sehen SQL-Abfragen zwar ähnlich, sind jedoch in vielen Punkten wesentlich unflexibler als herkömmliche Datenbank-Queries:

- Die `WHERE`-Klausel darf nur Bedingungen für Felder enthalten, die mindestens ein von Facebook explizit indiziertes Feld besitzen.
- Die `FROM`-Klausel einer FQL-Abfrage kann immer nur eine Tabelle enthalten – Verknüpfungen mehrerer Tabellen mittels Inner oder Outer Join sind grundsätzlich nicht vorgesehen.

- Verschachtelte Abfragen (*Sub Selects*) sind in der `WHERE`-Klausel prinzipiell mit dem Schlüsselwort `IN` möglich, können aber, anders als bei SQL, nicht Variablen der äußeren Abfrage referenzieren.
- FQL kennt keine Möglichkeit zur Gruppierung von Ergebnissen ähnlich der `GROUPBY`-Klausel von SQL.
- Mathematische und logische Operationen sind in FQL grundsätzlich vorhanden, in der offiziellen Dokumentation leider aber nur unzureichend beschrieben.
- FQL unterstützt die Klauseln `ORDERBY` zum Sortieren und `LIMIT` zum Festlegen der Anzahl der Ergebniszeilen.
- FQL ermöglicht *ausschließlich Lesezugriffe*: `INSERT`-, `UPDATE`- oder `DELETE`-Befehle sind nicht mittels FQL möglich und können nur wie gewohnt über die Graph API durchgeführt werden.
- Ähnlich der Adressierung von Objekten in der Graph API dient das Schlüsselwort `me()` in FQL dazu, den aktuellen Benutzer anhand des verwendeten Access Tokens zu identifizieren.
- Die Verwendung eindeutiger Benutzer- oder Seitennamen wird – anders als beim Adressieren von Objekten in der Graph API – nicht von FQL unterstützt.

5.1 FQL-Zugriffe über die Graph API

Mit der angekündigten Einstellung der HTTP-REST-API müssen Entwickler FQL-Abfragen nunmehr über die Graph API durchführen. Dazu ist ein GET-Zugriff auf den speziellen API-Endpunkt `/fql` notwendig, dem neben dem optionalen Access Token die FQL-Abfrage im Parameter `q` übergeben wird:

```
https://graph.facebook.com/fql?q=SELECT+uid2+FROM+friend+WHERE+uid1=me()&
access_token=...
```

Wichtig: Bei der Übergabe der FQL-Abfrage müssen Sie unbedingt auf eine korrekte URL-Codierung der Query achten. In PHP kann dies am einfachsten mit der Funktion `urlencode()` erfolgen.

Parameter	Beschreibung	Typ	Pflichtfeld
`q`	Auszuführende FQL-Abfrage. Die Abfrage muss dabei URL-codiert übergeben werden.	`string`	ja
`access_token`	Access Token zur Durchführung der Abfrage	`string`	nein

Tabelle 5.1 Parameter zum Ausführen von FQL über die Graph API

Zum Testen von FQL-Abfragen kann das in Abschnitt 3.1.9, »Der Graph API Explorer«, vorgestellte Tool verwendet werden. Dabei kann die FQL-Abfrage ohne weitere URL-Codierung im Parameter `q` an den API-Endpunkt angehängt werden, da die Codierung automatisch vom Graph API Explorer übernommen wird:

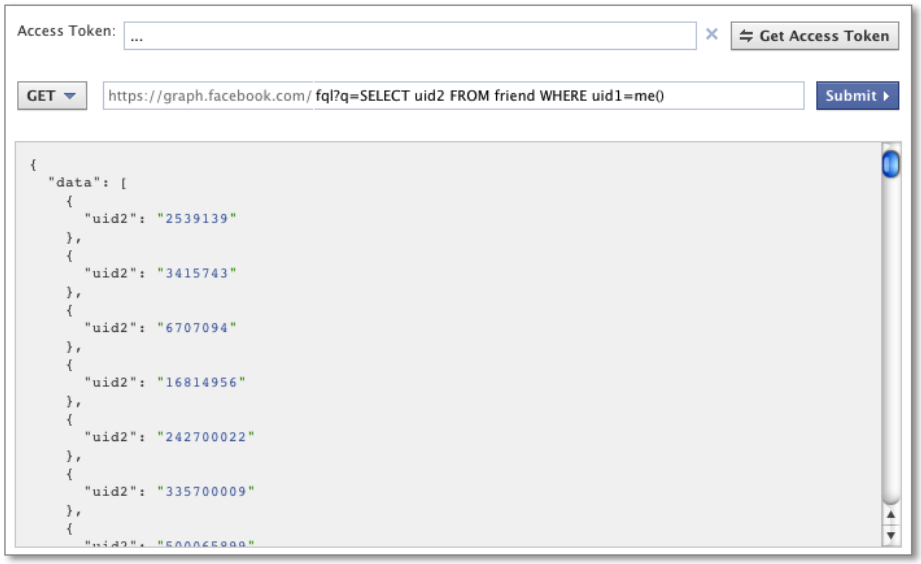


Abbildung 5.1 Durchführen einer FQL-Abfrage über den Graph API Explorer

Das Ergebnis von FQL-Abfragen wird – wie von der Graph API gewohnt – in JSON-Codierung zurückgegeben.

Das folgende Beispiel zeigt die Durchführung einer einfachen FQL-Abfrage über die Graph API. Der Code muss am Webserver unter dem Dateinamen *6-fql-graphapi.php* abgelegt werden:

```
<?
include_once("tools.php");
define('APP_ID', '214728715257742');
define('APP_SECRET', '...');
define('SITE_URL', 'http://apps.mycompany.com');

$login_url = "http://www.facebook.com/dialog/oauth?client_id=".
   APP_ID."&redirect_uri=".SITE_URL."/6-fql-graphapi.php";

if (empty($_REQUEST["code"])) {
   header("Location: ".$login_url);
   exit;
```

```
} else {
   $url =
      "https://graph.facebook.com/oauth/access_token?client_id=".
      APP_ID."&redirect_uri=".urlencode(SITE_URL.
             "/6-fql-graphapi.php")."&client_secret=".APP_SECRET.
             "&code=".$_REQUEST["code"];

      $access_token = curl($url);
      $access_token =
         substr($access_token,0,strpos($access_token,"&"));
      $fql = "SELECT uid,name,pic FROM user WHERE uid=me()";
      $url = "https://graph.facebook.com/fql?q=".
             urlencode($fql)."&".$access_token;
      $result = json_decode(curl($url));
      print "FQL-Query: ".$fql."<br/><br/>";
      print "FQL-Ergebnis:<br/><pre>";
      print_r($result);
}
?>
```

Listing 5.1 Ausführung einer FQL-Abfrage über die Graph API

Und so funktioniert dieses Beispiel:

- Da für viele FQL-Zugriffe analog zur Graph API ein Access Token erforderlich ist, wird im ersten Schritt geprüft, ob der Parameter `code` gesetzt ist – beim ersten Aufruf des Beispiels ist dies nicht der Fall, weswegen auf den OAuth-Dialog von Facebook umgeleitet wird. Nach erfolgreicher Autorisierung erfolgt der Callback mit gesetztem Parameter `code` auf das Skript.
- Mittels `code` kann nun das Access Token wie gewohnt bezogen werden.
- Die durchzuführende FQL-Abfrage wird in der Variablen `$fql` vorbereitet und an den API-Endpunkt `/fql?q=...` übergeben.
- Dabei muss die FQL-Abfrage im Parameter `q` mittels der PHP-Funktion `urlencode()` umgewandelt werden.
- Das Ergebnis der FQL-Abfrage wird wie üblich als JSON-Array zurückgegeben und kann mit der Funktion `json_decode()` in eine PHP-Struktur umgewandelt werden.

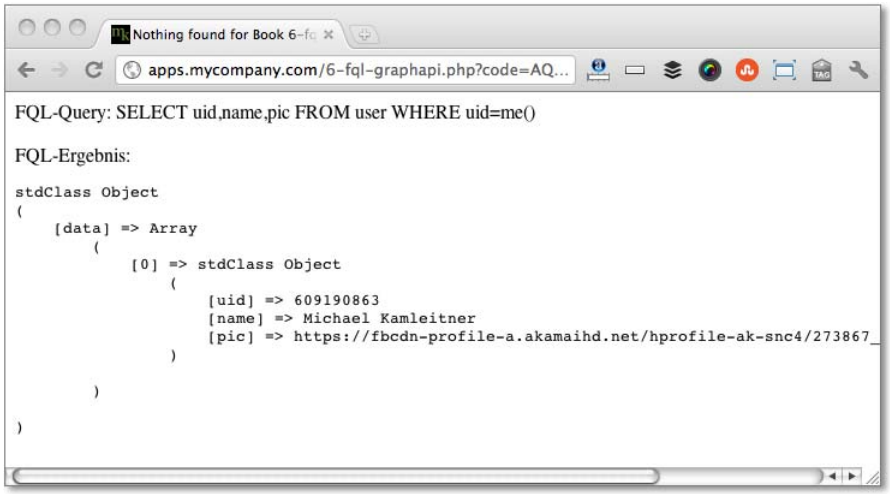


Abbildung 5.2 Anzeige des Ergebnisses einer einfachen FQL-Abfrage

5.2 FQL-Tabellen

Der folgende Abschnitt gibt einen vollständigen Überblick über die verfügbaren FQL-Tabellen und deren Felder. Eine Besonderheit kommt dabei der Feldeigenschaft INDIZIERT zu, denn die `WHERE`-Bedingungen einer FQL-Abfrage muss immer mindestens ein indiziertes Feld enthalten! Wenn Sie die Angabe eines indizierten Feldes vergessen, liefert die Abfrage eine entsprechende Fehlermeldung zurück.

Eine weitere Besonderheit von FQL betrifft Datums- bzw. Zeitfelder – diese liefern – anders als in der Graph API – keinen Zeitstempel im ISO-8601-Format, sondern die unter UNIX-Betriebssystemen übliche Notation als Differenz zwischen dem 1. Januar 1970, 00:00:00 Uhr, in Sekunden zurück.

Da Facebook, ähnlich wie bei der Graph API, laufend Anpassungen und Erweiterungen an den FQL-Tabellen vornimmt, ist ein Blick auf die offizielle Online-Dokumentation jedenfalls empfehlenswert.

Wichtiger Link:

- *https://developers.facebook.com/docs/reference/fql/* – offizielle Dokumentation zu den FQL-Tabellen

5.2.1 Albumtabelle

Die Tabelle `album` entspricht dem in Abschnitt 3.2.4, »Das Albumobjekt«, beschriebenen Objekt der Graph API. Hinsichtlich der Zugriffsberechtigungen gelten analog die gleichen Anforderungen an das Access Token wie beim Zugriff über die Graph API.

Beispiel – Abfrage der eigenen Alben des aktuellen Benutzers:

```
SELECT aid, object_id, name FROM album WHERE owner=me()
```

Beispiel – Abfrage eines bestimmten Albums anhand seiner ID:

```
SELECT description, link, name FROM album
WHERE object_id=10150397398555864
```

Feldname	Beschreibung	Typ	Indiziert
aid	Interne Facebook-ID des Albums – die Verwendung dieser ID stammt aus der Zeit vor Einführung der Graph API.	string	ja
object_id	Objekt-ID des Albums – entspricht jener ID, über die das Objekt üblicherweise mit der Graph API adressiert werden kann.	int	ja
owner	Besitzer des Albums (Benutzer oder Seite)	int	ja
cover_pid	Interne Facebook-ID des Cover-Fotos – die Verwendung dieser ID stammt aus der Zeit vor Einführung der Graph API.	string	nein
cover_object_id	Objekt-ID des Cover-Fotos – entspricht jener ID, über die das Objekt üblicherweise mit der Graph API adressiert werden kann.	int	nein
name	Titel des Albums	string	nein
created	Zeitstempel der Erstellung des Albums	UNIX-Zeitstempel	nein
modified	Zeitstempel der letzten Modifikation des Albums	UNIX-Zeitstempel	nein
modified_major	Zeitstempel der letzten signifikanten Modifikation des Albums (z. B. das Hochladen eines neuen Fotos)	UNIX-Zeitstempel	nein
description	Beschreibung des Albums	string	nein
location	Ort des Albums	string	nein

Tabelle 5.2 Felder der Albumtabelle

Feldname	Beschreibung	Typ	Indiziert
size	Anzahl der Fotos und Videos in diesem Album	int	nein
link	URL zum Album	string	nein
visible	Privatsphäre-Einstellung des Albums (friends, friends-of-friends, networks, everyone, custom)	string	nein
edit_link	URL zur Bearbeitungsseite des Albums (aufrufbar nur durch den Besitzer des Albums)	string	nein
type	Typ des Albums (profile, mobile, wall, normal)	string	nein
can_upload	Zeigt an, ob der aktuelle Benutzer Fotos in das Album hochladen kann.	boolean	nein
photo_count	Anzahl der Fotos in diesem Album	integer	nein
video_count	Anzahl der Videos in diesem Album	integer	nein

Tabelle 5.2 Felder der Albumtabelle (Forts.)

5.2.2 Application-Tabelle

Die Tabelle `application` entspricht dem in Abschnitt 3.2.5, »Das Anwendungsobjekt«, beschriebenen Graph-API-Objekt. Da mittels FQL ausschließlich öffentliche Informationen über Anwendungen ermittelt werden können, erfordert der Zugriff kein Access Token.

Beispiel – Auslesen einer bestimmten Anwendung über die bekannte CANVAS URL:

```
SELECT app_id, api_key, canvas_name FROM application
WHERE canvas_name='meinklub'
```

Feldname	Beschreibung	Typ	Indiziert
app_id	Objekt-ID der Anwendung	int	ja
api_key	öffentlicher API-Key der Anwendung	string	ja

Tabelle 5.3 Felder der Application-Tabelle

Feldname	Beschreibung	Typ	Indiziert
canvas_name	Entspricht der Option Canvas URL in den Einstellungen der Anwendung.	string	ja
display_name	Name der Anwendung	string	nein
icon_url	URL zum Icon der Anwendung	string	nein
logo_url	URL zum Logo der Anwendung	string	nein
company_name	Name des Herstellers der Anwendung	string	nein
developers	Liste der Benutzer, die als Entwickler der Anwendung eingetragen sind. Dieses Feld wird von Facebook mittlerweile immer leer zurückgeliefert, da die Entwickler der Anwendung nicht mehr veröffentlicht werden!	string	nein
description	Beschreibung der Anwendung	string	nein
daily_active_users	Anzahl der aktiven Benutzer am letzten Tag	integer	nein
weekly_active_users	Anzahl der aktiven Benutzer in der letzten Woche	integer	nein
monthly_active_users	Anzahl der aktiven Benutzer im letzten Monat	integer	nein
category	Kategorie der Anwendung	string	nein
subcategory	Unterkategorie der Anwendung	string	nein
is_facebook_app	Zeigt an, ob die Anwendung von Facebook selbst oder einem Drittthersteller entwickelt wurde.	boolean	nein
restriction_info	demographische Einschränkungen der Anwendung	JSON-Objekt	nein

Tabelle 5.3 Felder der Application-Tabelle (Forts.)

5.2.3 Apprequest-Tabelle

Applikationsanfragen werden, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben, in `apprequest`-Objekten im sozialen Graphen von Facebook gespeichert. Diese Objekte können ebenfalls über die FQL-Tabelle `apprequest` gelesen werden. Zum

Lesen der Applikationsanfragen eines Benutzers ist immer das Access Token dieser Person erforderlich.

Beispiel – Lesen aller Anfragen einer Anwendung an den aktuellen Benutzer:

```
SELECT recipient_uid, request_id, app_id FROM apprequest
WHERE recipient_uid = me() AND app_id = ...
```


Hinweis: Da immer nur die Applikationsanfragen der eigenen Anwendung ausgelesen werden dürfen, müssen die Felder `recipient_uid` und `app_id` immer gemeinsam in der `WHERE`-Klausel abgefragt werden!

Beispiel – Lesen einer bestimmten Applikationsanfrage:

```
SELECT request_id, app_id FROM apprequest
WHERE request_id = ...
```

Feldname	Beschreibung	Typ	Indiziert
`request_id`	Objekt-ID der Applikationsanfrage	`int`	ja
`app_id`	Objekt-ID der Anwendung, von der die Anfrage erzeugt wurde	`int`	ja (nur gemeinsam mit `recipient_uid`)
`recipient_uid`	ID des Benutzers, an den die Anfrage verschickt wurde	`int`	ja (nur gemeinsam mit `app_id`)
`sender_uid`	ID des Benutzers, der die Anfrage verschickt hat	`int`	nein
`message`	Nachricht, mit der die Anfrage verschickt wurde	`string`	nein
`data`	optional mit der Anfrage verknüpftes JSON-Objekt	`string`	nein
`created_time`	Zeitstempel der Anfrage	UNIX-Zeitstempel	nein

Tabelle 5.4 Felder der Apprequest-Tabelle

5.2.4 Checkin-Tabelle

Die Tabelle `checkin` entspricht dem in Abschnitt 3.2.14, »Das Checkin-Objekt«, beschriebenen Objekt der Graph API. Hinsichtlich der Zugriffsberechtigungen gelten analog die gleichen Anforderungen an das Access Token wie beim Zugriff über die Graph API.

Beispiel – Lesen der neuesten Checkins einer bestimmten Seite bzw. eines bestimmten Ortes:

```
SELECT checkin_id, author_uid, message, timestamp
FROM checkin WHERE page_id = 163483520335945
ORDER BY timestamp DESC
```

Beispiel – Lesen der neuesten Checkins des aktuellen Benutzers:

```
SELECT checkin_id, page_id, message FROM checkin
WHERE author_uid= me()
ORDER BY timestamp DESC
```

Feldname	Beschreibung	Typ	Indiziert
`checkin_id`	Objekt-ID des Checkins	`int`	ja
`author_uid`	ID des Benutzers, der den Checkin veröffentlicht hat	`int`	ja
`page_id`	ID der Seite, die den Ort des Checkins repräsentiert	`int`	ja
`app_id`	ID der Anwendung, über die der Checkin veröffentlicht wurde	`int`	nein
`post_id`	ID des Postings, das mit dem Checkin veröffentlicht wurde	`int`	nein
`coords`	Koordinaten des Checkins	Array mit den Feldern `latitude` und `longitude`	nein
`timestamp`	Zeitstempel des Checkins	UNIX-Zeitstempel	nein

Tabelle 5.5 Felder der Checkin-Tabelle

Feldname	Beschreibung	Typ	Indiziert
tagged_uids	weitere Benutzer, die in diesem Checkin markiert wurden	Array, bestehend aus Benutzer-IDs	nein
message	optionale Nachricht zum Checkin	string	nein

Tabelle 5.5 Felder der Checkin-Tabelle (Forts.)

5.2.5 Comment-Tabellen

Die Tabelle comment entspricht grundsätzlich dem in Abschnitt 3.2.15, »Das Kommentarobjekt«, beschriebenen Objekt der Graph API. Hinsichtlich der Zugriffsberechtigungen gelten analog die gleichen Anforderungen an das Access Token wie beim Zugriff über die Graph API.

Beispiel – Lesen der Kommentare zu einem Wall-Posting:

```
SELECT post_id, post_fbid, fromid, object_id, text, time
FROM comment WHERE object_id = 177372615695106
```

Die Tabelle comments erlaubt auch den Zugriff auf Kommentare, die über das Social Plugin fb:comments veröffentlicht wurden. fb:comments erlaubt die Einbindung eines standardisierten Kommentarmoduls auf beliebigen Webseiten und -apps und wird in Abschnitt 6.9, »Die Kommentarbox«, ausführlich beschrieben. Das Kommentarmodul wird grundsätzlich über die URL der Webseite, auf der es eingesetzt wird, identifiziert. Um die Kommentare eines bestimmten Kommentarmoduls einzulesen, muss daher mithilfe der Tabelle link_stat und einer Unterabfrage die Objekt-ID der Webseite, auf der das Kommentarmodul eingebunden wurde, herausgefunden werden:

```
SELECT post_fbid, fromid, object_id, text, time
FROM comment WHERE object_id in
 (SELECT comments_fbid FROM link_stat
  WHERE url =
  'http://developers.facebook.com/docs/reference/fql/comment/')
```

Feldname	Beschreibung	Typ	Indiziert
xid	externe ID zur Identifikation des Kommentarmoduls	string	ja

Tabelle 5.6 Felder der Comment-Tabelle

Feldname	Beschreibung	Typ	Indiziert
object_id	ID des Objekts (Foto, Wall-Posting etc.), das kommentiert wurde	string	ja
post_id	ID des Kommentars	string	ja
fromid	ID des Benutzers, der den Kommentar veröffentlicht hat	int	nein
time	Zeitstempel des Kommentars	UNIX-Zeitstempel	nein
text	Text des Kommentars	string	nein
id	eindeutige Kommentar-ID innerhalb einer xid	int	nein
username	Name des Benutzers, wenn dieser nicht über einen Facebook-Account kommentiert hat (das Kommentar-Plugin unterstützt Benutzer-Logins via Yahoo!, AOL und Hotmail)	string	nein
reply_xid	Ziel-ID für ein Wall-Posting, das vom Benutzer veröffentlicht wurde	string	nein
post_fbid	Objekt-ID des Kommentars	string	nein
app_id	ID der Anwendung, die mit dem Kommentar verknüpft ist	int	nein
likes	Anzahl der Gefällt mir des Kommentars	int	nein
comments	weitere Kommentare, die als Antwort zu diesem Kommentar veröffentlicht wurden	Array an Objekten mit den Feldern id, from, message und created_time	nein
user_likes	Zeigt, ob der aktuelle Benutzer bei diesem Kommentar auf Gefällt mir geklickt hat.	boolean	nein
is_private	Zeigt, ob der Kommentar privat ist.	boolean	nein

Tabelle 5.6 Felder der Comment-Tabelle (Forts.)

Hinweis: In früheren Versionen des Kommentarmoduls wurde die Verknüpfung zwischen einbindender Webseite und Modul nicht durch die URL, sondern durch die Anwendungs-ID in Kombination mit einem manuell gesetzten Parameter `xid` hergestellt. Die Felder der Tabelle `comments` enthalten diese `xid` aus Kompatibilitätsgründen nach wie vor.

Comments-Info-Tabelle

In diesem Zusammenhang muss aus Gründen der Vollständigkeit auch die Tabelle `comments_info` erwähnt werden, aus der die mittlerweile eingestellten `xid`-basierten Kommentarmodule einer bestimmten Anwendungs-ID abgefragt werden können:

Feldname	Beschreibung	Typ	Indiziert
`app_id`	ID der Anwendung	`string`	ja
`xid`	externe ID zur Identifikation des Kommentarmoduls	`string`	nein
`count`	Anzahl der Kommentare	`int`	nein
`updated_time`	Zeitstempel der letzten Modifikation des Albums	UNIX-Zeitstempel	nein

Tabelle 5.7 Felder der Comments-Info-Tabelle

5.2.6 Connection-Tabelle

Die Tabelle `connection` enthält alle Verbindungen des aktuellen Benutzers (Freundschaft) zu anderen Benutzern und Seiten (Fan). Um diese Verbindungen einzulesen, ist ein Access Token mit der erweiterten Berechtigung `read_stream` notwendig. Die Tabelle `connection` ist immer nur für den aktuellen Benutzer verfügbar und muss daher folgendermaßen abgefragt werden:

```
SELECT target_id,target_type FROM connection WHERE source_id=me()
```

Feldname	Beschreibung	Typ	Indiziert
`source_id`	ID des Benutzers, von dem die Verbindung ausgeht	`int`	ja
`target_id`	ID des Benutzers oder der Seite, mit der der Benutzer verknüpft ist	`int`	nein

Tabelle 5.8 Felder der Connection-Tabelle

Feldname	Beschreibung	Typ	Indiziert
target_type	Zeigt an, ob das verknüpfte Objekt ein Benutzer (user) oder eine Seite (page) ist.	string	nein
is_following	Zeigt an, ob der Benutzer, von dem die Verknüpfung ausgeht, Pinnwand-Beiträge des verknüpften Benutzers oder der verknüpften Seite ausgeblendet hat (false) oder nicht (true).	boolean	nein

Tabelle 5.8 Felder der Connection-Tabelle (Forts.)

5.2.7 Cookies-Tabelle

Laut offizieller Dokumentation enthält die Tabelle cookies eine Liste aller aktuell gesetzten Browser-Cookies des aktuellen Benutzers. Zum Zugriff wird ein Access Token benötigt, womit lediglich die Cookies des eigenen Benutzers innerhalb der Anwendung, für die das Token ausgestellt wurde, abgefragt werden können:

```
SELECT uid, name, value, expires, path
FROM cookies WHERE uid = me()
```

Hinweis: Die Tabelle cookies hat während der Arbeiten an diesem Buch ausschließlich leere Ergebnisse zurückgeliefert, was nahelegt, dass diese Funktion nur in Verbindung mit der mittlerweile eingestellten HTTP-REST-API funktioniert hat. In den aktuellen Versionen des PHP SDKs und seit der Einführung von OAuth 2.0 dürfte dieser Tabelle daher keine Bedeutung mehr zukommen.

Feldname	Beschreibung	Typ	Indiziert
uid	ID des Benutzer, dem das Cookie zugeordnet ist	int	ja
name	Name des Cookies	string	nein
value	Inhalt des Cookies	string	nein
expires	Zeitstempel, zu dem die Gültigkeit des Cookies ausläuft	boolean	nein
path	URL-Pfad relativ zur Canvas URL, mit der das Cookie verknüpft werden soll	string	nein

Tabelle 5.9 Felder der Cookies-Tabelle

5.2.8 Developer-Tabelle

Die Tabelle `developer` enthält alle Anwendungen eines Benutzers, bei denen dieser als Administrator, Entwickler, Tester oder Insights-Benutzer eingetragen ist. Der Zugriff ist nur auf die Anwendungen des aktuellen Benutzers möglich und erfordert ein beliebiges Access Token:


```
SELECT application_id FROM developer WHERE developer_id = me()
```

Feldname	Beschreibung	Typ	Indiziert
developer_id	ID des Benutzers, der der Anwendung als Administrator, Entwickler, Tester oder Insight-Benutzer zugeordnet ist	int	ja
application_id	ID der Anwendung	int	nein

Tabelle 5.10 Felder der Developer-Tabelle

5.2.9 Domain-Tabellen

Die Tabelle `domain` entspricht grundsätzlich dem in Abschnitt 3.2.6, »Das Domain-Objekt«, beschriebenen Objekt der Graph API und erlaubt das Auslesen der Zuordnung von Domain-Namen zu ihren Objekt-IDs im sozialen Graphen.

Beispiel – Auslesen der Objekt-ID einer des Namens nach bekannten Domain

```
SELECT domain_id FROM domain
WHERE domain_name='www.facebook.com'
```

Beispiel – Auslesen des Domain-Namens einer der ID nach bekannten Domain:

```
SELECT domain_name FROM domain
WHERE domain_id= 369296215699
```

Feldname	Beschreibung	Typ	Indiziert
domain_id	Objekt-ID der Domin	int	ja
domain_name	Domain-Name	string	ja

Tabelle 5.11 Felder der Domain-Tabelle

Domain-Administrator-Tabelle

Neben der `domain`-Tabelle wird in der Tabelle `domain_admin` gespeichert, ob eine Domain von einem Facebook-Benutzer beansprucht (*claim*) wurde. Dies geschieht üblicherweise über Facebook Insights, das Statistik-Tool von Facebook, um Einblick in Nutzungsstatistiken einer Domain zu erhalten. Domains können von Benutzern, Seiten und Anwendungen bzw. deren Administratoren beansprucht werden.

Feldname	Beschreibung	Typ	Indiziert
`owner_id`	ID des beanspruchenden Benutzers bzw. der beanspruchenden Seite oder Anwendung	`int`	ja
`domain_id`	Objekt-ID der Domain	`int`	ja

Tabelle 5.12 Felder der Domain-Administrator-Tabelle

Zum Beanspruchen einer Domain ist es notwendig, durch das Setzen eines speziellen Open Graph Tags `fb:admins` auf der eigenen Webseite die Inhaberschaft der Domain nachzuweisen. Ab diesem Zeitpunkt wird die erfolgreiche Beanspruchung der Domain auch in der FQL-Tabelle `domain_admins` repräsentiert.

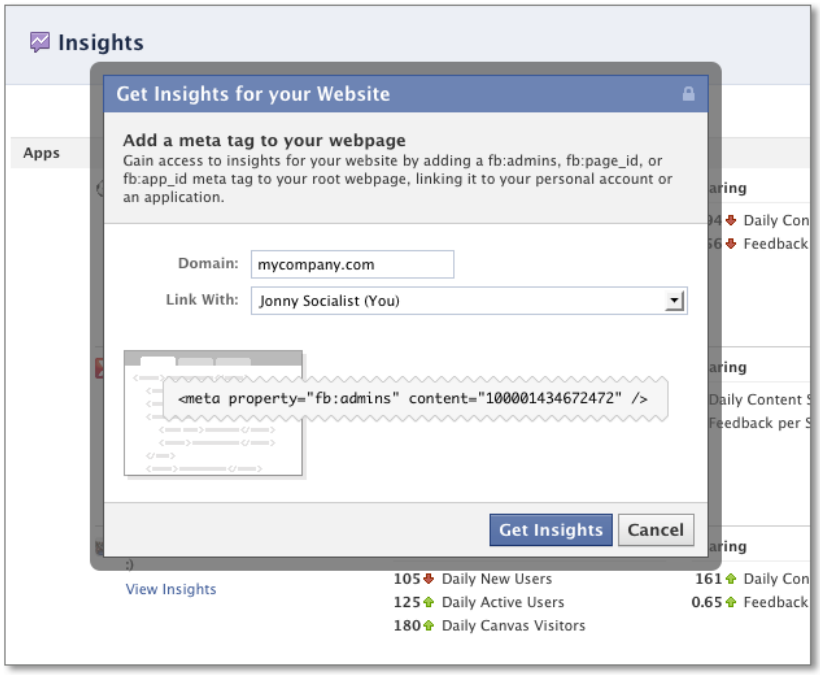


Abbildung 5.3 Beanspruchen einer Domain in Facebook Insights

5.2.10 Event-Tabellen

Die Tabelle `event` entspricht grundsätzlich dem in Abschnitt 3.2.9, »Das Veranstaltungsobjekt«, beschriebenen Objekt der Graph API. Hinsichtlich der Zugriffsberechtigungen gelten analog die gleichen Anforderungen an das Access Token wie beim Zugriff über die Graph API.

Beispiel – Auslesen eines bestimmten Events:

```
SELECT name, venue, location, start_time FROM event
WHERE eid = 209798352393506
```

Feldname	Beschreibung	Typ	Indiziert
`eid`	numerische ID der Veranstaltung	`int`	ja
`name`	Name der Veranstaltung	`string`	nein
`tagline`	der Untertitel bzw. die Zusammenfassung der Veranstaltung	`string`	nein
`nid`	numerische ID des Benutzernetzwerks, zu dem die Veranstaltung gehört	`int`	nein
`pic_small`	URL zum kleinen Profilbild der Veranstaltung (maximal 50 x 150 Pixel)	`string`	nein
`pic_big`	URL zum großen Profilbild der Veranstaltung (maximal 200 x 600 Pixel)	`string`	nein
`pic_square`	URL zum quadratischen Profilbild der Veranstaltung (50 x 50 Pixel)	`string`	nein
`pic`	URL zum mittelgroßen Profilbild der Veranstaltung (maximal 100 x 300 Pixel)	`string`	nein
`host`	Name des Veranstalters	`string`	nein
`description`	Beschreibung der Veranstaltung	`string`	nein

Tabelle 5.13 Felder der Event-Tabelle

Feldname	Beschreibung	Typ	Indiziert
event_type	Kategorie der Veranstaltung (diese Event-Einstellung wird von Facebook nicht mehr bereitgestellt und steht nur bei historischen Events zur Verfügung)	string	nein
event_subtype	Unterkategorie der Veranstaltung (diese Event-Einstellung wird von Facebook nicht mehr bereitgestellt und steht nur bei historischen Events zur Verfügung)	string	nein
start_time	Zeitstempel des Beginns der Veranstaltung	UNIX-Zeitstempel	nein
end_time	Zeitstempel des Endes der Veranstaltung	UNIX-Zeitstempel	nein
creator	Benutzer- oder Seiten-ID des Erstellers der Veranstaltung	int	nein
update_time	Zeitstempel der letzten Aktualisierung der Veranstaltung	UNIX-Zeitstempel	nein
venue	Ort der Veranstaltung	Array mit den Feldern street, city, state, zip, country, latitude und longitude	nein
location	Name des Ortes der Veranstaltung	string	nein
privacy	Privatsphäre-Einstellung der Veranstaltung (OPEN, CLOSED oder SECRET)	string	nein
hide_guest_list	Zeigt an, ob die Gästeliste versteckt werden soll.	boolean	nein
can_invite_friends	Zeigt an, ob der Benutzer, der die FQL-Anfrage stellt, Freunde zu der Veranstaltung einladen darf.	boolean	nein

Tabelle 5.13 Felder der Event-Tabelle (Forts.)

Event-Member-Tabelle

Die Tabelle `event_member` enthält die Liste der Teilnehmer einer Veranstaltung. Je nach Art des Zugriffs ist ein Access Token analog, wie in Abschnitt 3.2.9, »Das Veranstaltungsobjekt«, beschrieben, notwendig.

Beispiel – Lesen aller Teilnehmer einer bestimmten Veranstaltung:

```
SELECT eid, uid, rsvp_status FROM event_member
WHERE eid = 209798352393506;
```

Beispiel – Lesen aller Veranstaltungen des aktuellen Benutzers:

```
SELECT eid, uid, rsvp_status FROM event_member WHERE uid=me()
```

Feldname	Beschreibung	Typ	Indiziert
`eid`	numerische ID der Veranstaltung	`int`	ja
`uid`	Benutzer-ID des Teilnehmers	`int`	nein
`rsvp_status`	Teilnahmestatus des Benutzers (`attending`, `unsure`, `declined` oder `not_replied`)	`string`	nein
`start_time`	UNIX-Zeitstempel des Beginns der Veranstaltung	UNIX-Zeitstempel	nein

Tabelle 5.14 Felder der Event-Member-Tabelle

5.2.11 Family-Tabelle

Die FQL-Tabelle `family` erlaubt den Zugriff auf die Familienmitglieder eines Benutzers. Die Tabelle entspricht der `family`-Verknüpfung des User-Objekts, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben. Das verwendete Access Token muss zum Lesen der Familienmitglieder die Berechtigung `user_relationships` (aktueller Benutzer) bzw. `friends_relationships` (Freunde des aktuellen Benutzers) enthalten.

Beispiel – Lesen aller Familienmitglieder des aktuellen Benutzers:

```
SELECT uid, name, birthday, relationship FROM family
WHERE profile_id = me()
```

Feldname	Beschreibung	Typ	Indiziert
`profile_id`	numerische ID des abgefragten Benutzers	`int`	ja
`uid`	Numerische ID des verknüpften Familienmitglieds. Nur gesetzt, wenn dieses die Verknüpfung bestätigt hat.	`int`	nein
`name`	Name des Familienmitglieds	`string`	nein
`birthday`	Geburtstag des Familienmitglieds. Nur gesetzt, wenn dieses die Verknüpfung *nicht* bestätigt hat.	`string`	nein
`relationship`	Art der Familienbeziehung (`sister`, `brother` etc.)	`string`	nein

Tabelle 5.15 Felder der Family-Tabelle

5.2.12 Friend-Tabelle

Die FQL-Tabelle `friend` bildet die Freundschaftsbeziehungen im sozialen Graphen von Facebook ab und entspricht damit der `friends`-Beziehung, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben.

Beispiel: Lesen der vollständigen Freundesliste des aktuellen Benutzers:

```
SELECT uid2 FROM friend WHERE uid1 = me()
```

Dabei können die Spalten `uid1` und `uid2` beliebig vertauscht werden, da die FQL-Tabellen die Freundschaftsbeziehungen bidirektional abbilden.

Hinweis: Auch hier gilt – die vollständige Liste an Freunden kann nur für den eigenen, aktuellen Benutzer ausgelesen werden, nicht aber für die Freunde des aktuellen Benutzers!

Beispiel: Über FQL kann auch die Freundschaftsbeziehung zweier Benutzer geprüft werden:

```
SELECT uid1, uid2 FROM friend
WHERE uid1 = 532617296 AND uid2 = 633616172
```

Sind die beiden Benutzer nicht befreundet, wird als Ergebnis ein leeres Array zurückgeliefert.

Hinweis: Die Freundschaftsbeziehung kann für alle Freunde des aktuellen Benutzers geprüft werden!

Feldname	Beschreibung	Typ	Indiziert
uid1	numerische ID des ersten Benutzers in der Freundschaftsbeziehung	int	ja
uid2	numerische ID des zweiten Benutzers in der Freundschaftsbeziehung	int	ja

Tabelle 5.16 Felder der Friend-Tabelle

5.2.13 Friend-Request-Tabelle

Die FQL-Tabelle `friend_request` bildet alle ausstehenden Freundschaftsanfragen des aktuellen Benutzers ab. Sie entspricht der `friendrequests`-Verbindung, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben, und erfordert wie diese ein Access Token mit der erweiterten Berechtigung `read_requests`.

Beispiel – Lesen aller ausstehenden eingehenden Freundschaftsanfragen des aktuellen Benutzers:

```
SELECT uid_from, time, message
FROM friend_request WHERE uid_to = me()
```

Beispiel – Lesen einer bestimmten ausgehenden Freundschaftsanfrage des aktuellen Benutzers:

```
SELECT time, message FROM friend_request
WHERE uid_from = me() AND uid_to = 633616172
```

Hinweis: Das Auslesen aller ausgehenden Freundschaftsanfragen des aktuellen Benutzers wird von FQL nicht unterstützt, da das Feld `uid_from` nicht indiziert ist, sondern nur in Kombination mit `uid_to` abgefragt werden kann!

Feldname	Beschreibung	Typ	Indiziert
uid_to	numerische ID des Benutzers, an den die Anfrage gesendet wurde	int	ja
uid_from	numerische ID des Benutzers, der die Anfrage gesendet hat	int	nein (Abfrage nur in Kombination mit uid_from möglich)
time	UNIX-Zeitstempel des Beginns der Veranstaltung	UNIX-Zeitstempel	nein
message	persönliche Nachricht zur Anfrage	string	nein
unread	Zeigt an, ob die Anfrage bereits gelesen wurde. Wird nur zurückgeliefert, wenn uid_to dem aktuellen Benutzer entspricht.	boolean	nein

Tabelle 5.17 Felder der Friend-Request-Tabelle

5.2.14 Friendlist-Tabellen

Die FQL-Tabelle `friendlist` enthält die Freundeslisten des aktuellen Benutzers und entspricht damit der `friendlists`-Verbindung, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben. Zum Lesen der Freundeslisten ist ein Access Token mit der erweiterten Berechtigung `read_friendlists` notwendig.

Beispiel – Lesen der Freundeslisten des aktuellen Benutzers:

```
SELECT flid, owner, name FROM friendlist WHERE owner=me()
```

Feldname	Beschreibung	Typ	Indiziert
owner	numerische ID des Benutzers, der die Freundesliste besitzt	int	ja
flid	numerische ID der Freundesliste	int	nein
name	Name der Freundesliste	string	nein

Tabelle 5.18 Felder der Friendlist-Tabelle

Friendlist-Member-Tabelle

Die Mitglieder einer Freundesliste können aus der Tabelle `friendlist_member` ausgelesen werden. Auch hierzu ist die Berechtigung `read_friendlists` notwendig.

Beispiel – Lesen der Mitglieder einer bestimmten Freundesliste:

```
SELECT uid, flid FROM friendlist_member
WHERE flid = 10150456156070864
```

Beispiel – Lesen aller Mitglieder aus allen Freundeslisten des aktuellen Benutzers unter Verwendung einer verschachtelten Abfrage:

```
SELECT uid, flid FROM friendlist_member
WHERE flid IN (SELECT flid FROM friendlist WHERE owner=me())
```

Feldname	Beschreibung	Typ	Indiziert
`flid`	numerische ID der Freundesliste	`int`	ja
`uid`	numerische ID des Benutzers, der Mitglied der Liste ist	`int`	nein

Tabelle 5.19 Felder der Friendlist-Member-Tabelle

5.2.15 Group-Tabellen

Die FQL-Tabelle `group` bildet Gruppenobjekte, wie in Abschnitt 3.2.3, »Das Gruppenobjekt«, beschrieben, ab. Während öffentliche Gruppen mit einem beliebigen Access Token ausgelesen werden können, sind für private Gruppen die `user_groups`- bzw. `friends_groups`-Berechtigung und das Access Token eines Gruppenmitglieds notwendig.

Beispiel – Lesen einer öffentlichen Gruppe:

```
SELECT gid, name, description, pic_small
FROM group WHERE gid = 146797922030397
```

Hinweis: Die Gruppenfunktion von Facebook wurde im Oktober 2010 grundlegend verändert. Felder, die in der folgenden Übersicht mit dem Vermerk unbenutzt ab Version 1 versehen sind, werden für Gruppen, die nach Oktober 2010 erstellt wurden, nicht mehr verwendet.

Feldname	Beschreibung	Typ	Indiziert
gid	numerische ID der Gruppe	int	ja
name	Name der Gruppe	string	nein
nid	numerische ID des Benutzernetzwerks, zu dem die Gruppe gehört	int	nein
pic_small	URL zum kleinen Profilbild der Gruppe (maximal 50 x 150 Pixel)	string	nein
pic_big	URL zum großen Profilbild der Gruppe (maximal 200 x 600 Pixel)	string	nein
pic	URL zum mittelgroßen Profilbild der Gruppe (maximal 100 x 300 Pixel)	string	nein
description	Beschreibung der Gruppe (unbenutzt ab Version 1)	string	nein
group_type	Kategorie der Gruppe (unbenutzt ab Version 1)	string	nein
group_subtype	Unterkategorie der Gruppe	string	nein
recent_news	Neuigkeiten-Feld der Gruppe	string	nein
creator	ID des Benutzers, der die Gruppe angelegt hat	int	nein
update_time	UNIX-Zeitstempel des letzten Updates der Gruppe	UNIX-Zeitstempel	nein
office	Ort/Büro der Gruppe (unbenutzt ab Version 1)	string	nein
website	Website der Gruppe (unbenutzt ab Version 1)	string	nein
venue	geographischer Ort der Gruppe (unbenutzt ab Version 1)	Objekt mit den Feldern street, city, state, country, zip, latitude und longitude	nein

Tabelle 5.20 Felder der Group-Tabelle

Feldname	Beschreibung	Typ	Indiziert
icon	URL zum Icon der Gruppe (16 x 16 Pixel)	string	nein
icon34	URL zum Icon der Gruppe (34 x 34 Pixel)	string	nein
icon68	URL zum Icon der Gruppe (68 x 68 Pixel)	string	nein
email	E-Mail-Adresse, die genutzt werden kann, um in die Gruppe zu posten	string	nein
version	Kennzeichen, ob die Gruppe vor Oktober 2010 (Wert 0) oder danach (Wert 1) erstellt wurde	int	nein

Tabelle 5.20 Felder der Group-Tabelle (Forts.)

Group-Member-Tabelle

Um die Mitglieder einer Gruppe zu ermitteln, kann die FQL-Tabelle `group_member` ausgelesen werden. Auch hier gilt: Die Mitglieder einer öffentlichen Gruppe können mit einem beliebigen Access Token, die Mitglieder einer privaten Gruppe mit einem Access Token mit `user_groups`- bzw. `friends_groups`-Berechtigung eines Mitglieds ausgelesen werden.

Beispiel – Lesen aller Mitglieder einer bestimmten Gruppe:

```
SELECT gid, uid FROMgroup_member
WHERE gid = 146797922030397
```

Beispiel – Lesen aller Mitgliedschaften des aktuellen Benutzers:

```
SELECT gid, uid FROM group_member WHERE uid=me()
```

Feldname	Beschreibung	Typ	Indiziert
uid	numerische ID des Mitglieds	int	ja
gid	numerische ID der Gruppe	int	ja
administrator	Zeigt an, ob das Mitglied Gruppenadministrator ist.	boolean	nein

Tabelle 5.21 Felder der Group-Member-Tabelle

Feldname	Beschreibung	Typ	Indiziert
positions	Berechtigungsstufe des Mitglieds – OWNER, ADMIN oder OFFICER	string	nein
unread	Anzahl der ungelesenen Nachrichten in der Gruppe (nur verfügbar, wenn das Feld uid in der WHERE-Klausel abgefragt wird)	int	nein
bookmark_order	Position der Gruppe in den Bookmarks des Benutzers (nur verfügbar, wenn das Feld uid in der WHERE-Klausel abgefragt wird)	int	nein

Tabelle 5.21 Felder der Group-Member-Tabelle (Forts.)

5.2.16 Insights-Tabelle

Facebook bietet zahlreiche Nutzungsstatistiken zu Objekten vom Typ page und application. Abgesehen von einigen öffentlich einsehbaren Metriken, ist zum Zugriff ein Anwendungs-Access-Token (bei anwendungsbezogenen Statistiken) oder ein Access Token mit der read_insights-Berechtigung (Anwendungs-, Seiten- oder domainbezogene Statistiken notwendig) eines Benutzers mit Administrationsrechten notwendig.

Beispiel – Abfrage der Metrik application_installed_users für den Zeitraum 1. Dezember bis 31. Dezember 2011:

```
SELECT metric, value FROM insights
WHERE object_id= 32788395891
AND metric='application_installed_users'
AND end_time=end_time_date('2011-12-31')
AND period=period('month')
```

Feldname	Beschreibung	Typ	Indiziert
object_id	numerische ID der Seite, Anwendung oder Domain	int	ja
metric	Bezeichnung der Metrik	string	ja
end_time	UNIX-Zeitstempel, der das Ende des Betrachtungszeitraums festlegt	UNIX-Zeitstempel	ja

Tabelle 5.22 Felder der Insights-Tabelle

Feldname	Beschreibung	Typ	Indiziert
`period`	Betrachtungszeitraum in Sekunden (1 Tag = 86.400, 1 Woche = 604.800, 1 Monat = 2.592.000, 0 = Lifetime)	`int`	ja
`value`	Wert der abgefragten Metrik im spezifizierten Zeitraum	`string`	nein

Tabelle 5.22 Felder der Insights-Tabelle (Forts.)

- Beim Abfragen der `insights`-Tabelle ist zu beachten, dass in der `WHERE`-Klausel immer die Felder `object_id`, `metric`, `period` und `end_time` vorkommen müssen.
- Es wird also immer der Wert einer bestimmten Metrik für ein bestimmtes Objekt (Anwendung, Seite oder Domain) in einem bestimmten Betrachtungszeitraum abgefragt.
- Der Betrachtungszeitraum ergibt sich aus der Angabe `end_time`, die, als UNIX-Zeitstempel codiert, das Ende des Zeitraums festlegt. `period` legt dabei fest, wie lang der Zeitraum – ausgehend von seinem Endpunkt – gewählt werden soll.
- `period` kann entweder in Sekunden angegeben werden oder mit der Hilfsfunktion `period()`, der der gewünschte Zeitraum als gut lesbarer String übergeben wird: `period('day')`, `period('week')`, `period('month')`oder `period('lifetime')`. *Hinweis:* Die verfügbaren Zeitperioden unterscheiden sich von Metrik zu Metrik.
- `end_time` kann entweder als UNIX-Zeitstempel (Anzahl der Sekunden seit 1. Januar 1970, 00:00 Uhr) angegeben werden oder über die Hilfsfunktion `end_time_date()`, der ein Datumsstring in der Form »YYYY-MM-DD« übergeben werden kann.

Facebook bietet mittlerweile über hundert Metriken für Anwendungen, Seiten oder Domains an. Aufzählung und Beschreibung der einzelnen Zahlen würden den Rahmen dieses Buches sprengen, deshalb verweise ich Sie an dieser Stelle auf die offizielle Dokumentation.

Wichtiger Link:

- *https://developers.facebook.com/docs/reference/fql/insights/* – offizielle Dokumentation zur FQL-Tabelle `insights`

5.2.17 »Like«-Tabelle

Wall-Postings, Videos, Fotos, Fotoalben, Notizen und Links, bei denen ein Benutzer auf Gefällt mir geklickt hat, werden in der FQL-Tabelle `like` abgebildet. Die Tabelle entspricht damit der `likes`-Verknüpfung der genannten Objekte in der Graph API.

Hinweis: Gefällt mir von Facebook-Seiten oder anderen Open-Graph-Objekten werden nicht in der FQL-Tabelle `like`, sondern in `page_fan` abgebildet!

Beispiel – Lesen aller Gefällt mir eines bestimmten Fotoalbums:

```
SELECT user_id FROM like WHERE object_id=10150409321060864
```

Beispiel – Lesen aller Gefällt mir des aktuellen Benutzers. Diese Art des Zugriffs erfordert die erweiterte Berechtigung `read_stream`!

```
SELECT user_id, object_id, object_type, post_id
FROM like WHERE user_id=me()
```

Feldname	Beschreibung	Typ	Indiziert
`object_id`	numerische ID des Objekts (Video, Foto, Link, Notiz, Album etc.)	`int`	ja
`post_id`	ID des zum Gefällt mir gehörenden Wall-Postings	`int`	ja
`user_id`	ID des Benutzers, der auf Gefällt mir geklickt hat	`int`	ja
`object_type`	Typ des Objekts (`photo`, `album`, `event`, `group`, `note`, `link`, `video`, `application`, `status`, `check-in`, `review`, `comment`, `post`)	`string`	nein

Tabelle 5.23 Felder der »Like«-Tabelle

5.2.18 Linktabelle

Die FQL-Tabelle `link` enthält die von einem Benutzer auf der Pinnwand veröffentlichten Links und entspricht damit dem in Abschnitt 3.2.11, »Das Linkobjekt«, beschriebenen Graph-Objekt. Analog dazu ist ein beliebiges Access Token zum Lesen von öffentlichen bzw. die `read_stream`-Berechtigung zum Lesen von privaten Links notwendig.

Beispiel – Lesen aller Links einer Facebook-Seite:

```
SELECT link_id, owner, title, summary, url, image_urls
FROM link WHERE owner=129842310398883
```

Beispiel – Lesen eines bestimmten Links:

```
SELECT link_id, owner, title, summary, url, image_urls
FROM link WHERE link_id=336134509732399
```

Feldname	Beschreibung	Typ	Indiziert
link_id	numerische ID des Links	int	ja
owner	ID des Benutzers, der Seite oder der Anwendung, die den Link veröffentlicht hat	int	ja
owner_comment	vom teilenden Benutzer bzw. der Seite/Anwendung verfasste Nachricht zum Link	string	nein
created_time	UNIX-Zeitstempel der Veröffentlichung des Links	UNIX-Zeitstempel	nein
title	Linktext bzw. Titel der geteilten URL	string	nein
summary	Beschreibungstext des Links	string	nein
url	URL des geteilten Links	string	nein
picture	URL zum Thumbnail-Bild des Links	string	nein
image_urls	URLs zu weiteren Bildern, die mit dem Artikel verknüpft sind	Array	nein

Tabelle 5.24 Felder der Linktabelle

5.2.19 Link-Stat-Tabelle

Der Tabelle link_stat kommt besondere Bedeutung zu, da die in ihr abgebildeten Daten derzeit nicht über die Graph API, sondern ausschließlich per FQL zugänglich sind. link_stat enthält Informationen zu sozialen Interaktionen, die auf verschiedenen Webseiten bzw. URLs vonstattengehen. Die abgefragten Webseiten können z. B. Fan-Seiten auf Facebook sein, aber auch jede externe Webseite. die mittels Social Plugins und Open Graph oder auch durch einfaches Sharing Teil des sozialen Graphen der Facebook-Plattform geworden sind.

Beispiel – Lesen der Linkstatistiken einer externen Seite:

```
SELECT url, normalized_url, share_count, like_count, comment_count,
total_count FROM link_stat WHERE url='facebook.com/diesocialisten'
```

Beispiel – gleichzeitiges Lesen der Linkstatistiken von mehreren externen Seiten mittels IN-Operator:

```
SELECT url, normalized_url, share_count, like_count, comment_count,
total_count FROM link_stat
WHERE url in ('http://die.socialisten.at/2011/10/last-fm-scrobbler-for-face-
book-open-graph-timeline/', 'http://die.socialisten.at/2011/08/facebook-fluid-
canvas-layout-f-apps-games/')
```

Beispiel – Lesen der Linkstatistiken einer Facebook-Seite:

```
SELECT url, normalized_url, share_count, like_count, comment_count,
total_count FROM link_stat
WHERE url='facebook.com/diesocialisten'
```

Feldname	Beschreibung	Typ	Indiziert
url	URL, auf die sich die Linkstatistiken beziehen	string	ja
normalized_url	normalisierte URL – vereinheitlicht unterschiedliche Schreibweisen, z. B. abschließende Schrägstriche etc.)	string	nein
share_count	Gibt an, wie oft diese URL auf Facebook geteilt wurde.	int	nein
like_count	Gibt an, wie oft Benutzer Gefällt mir auf dieser URL angeklickt haben.	int	nein
comment_count	Anzahl der Kommentare, die zu dieser URL auf Facebook veröffentlicht wurden	int	nein
total_count	Summe aus share_count, like_count und comment_count	int	nein
click_count	Gibt an, wie oft Benutzer auf Facebook auf die geteilte URL geklickt haben.	int	nein

Tabelle 5.25 Felder der Link-Stat-Tabelle

Feldname	Beschreibung	Typ	Indiziert
comments_fbid	Ist auf der URL das Facebook-Kommentar-Plugin integriert, enthält dieses Feld die ID, mit der die Kommentare aus der Tabelle comment gelesen werden können.	int	nein
commentsbox_count	Ist auf der URL das Facebook-Kommentar-Plugin integriert, enthält dieses Feld die Anzahl der veröffentlichten Kommentare.	int	nein

Tabelle 5.25 Felder der Link-Stat-Tabelle (Forts.)

5.2.20 Mailbox-Folder-Tabelle

Die FQL-Tabelle mailbox_folder bildet die unterschiedlichen Ordner des Facebook-Nachrichten-Postfachs des aktuellen Benutzers ab. In der Graph API werden diese durch die Verknüpfungen inbox, outbox und updates des user-Objekts repräsentiert (siehe Abschnitt 3.2.1, »Das User-Objekt«). Die Nachrichten-Ordner können nur für den aktuellen Benutzer gelesen werden und erfordern ein Access Token mit der read_mailbox-Berechtigung.

Beispiel – Lesen aller Ordner des aktuellen Benutzers:

```
SELECT name, unread_count, total_count
FROM mailbox_folder
WHERE viewer_id = me()
```

Feldname	Beschreibung	Typ	Indiziert
folder_id	Numerische ID des Ordners. Diese ID ist für jeden Benutzer gleich und kann die Werte 0 (Inbox), 1 (Outbox) oder 4 (Updates) annehmen.	int	ja
viewer_id	ID des Benutzers, dessen Nachrichten-Postfach abgefragt wird	int	ja
name	Name des Ordners (Inbox)	string	nein
unread_count	Anzahl der ungelesenen Nachrichten	int	nein
total_count	Anzahl der Nachrichten insgesamt	int	nein

Tabelle 5.26 Felder der Mailbox-Folder-Tabelle

5.2.21 Message-Tabelle

Die FQL-Tabelle `message` enthält die einzelnen Nachrichten im Postfach des aktuellen Benutzers. Die Nachrichten können nur für den aktuellen Benutzer gelesen werden und erfordern ein Access Token mit der `read_mailbox`-Berechtigung. Die Tabelle entspricht dem in Abschnitt 3.2.20, »Das Nachrichtenobjekt«, beschriebenen Graph-Objekt `message`.

Beispiel – Lesen aller Nachrichten eines Threads:

```
SELECT message_id, body FROM message
WHERE thread_id=10150421384539627
```

Beispiel – Lesen einer bestimmten Nachricht:

```
SELECT message_id, thread_id, author_id, body, created_time
FROM message WHERE message_id='10150421384539627_0'
```

Feldname	Beschreibung	Typ	Indiziert
`message_id`	ID der Nachricht	`string`	ja
`thread_id`	ID des Threads	`int`	ja
`author_id`	Benutzer-ID des Senders der Nachricht	`int`	nein
`body`	Inhalt der Nachricht	`string`	nein
`created_time`	UNIX-Zeitstempel der Erstellung der Nachricht	UNIX-Zeitstempel	nein
`attachment`	Anhang der Nachricht	Array	nein
`viewer_id`	Benutzer-ID des Empfängers der Nachricht bzw. Besitzers des abgefragten Postfachs	`int`	nein

Tabelle 5.27 Felder der Message-Tabelle

5.2.22 Note-Tabelle

Die FQL-Tabelle `note` enthält auf der Pinnwand veröffentlichte Notizen und entspricht dem in Abschnitt 3.2.16, »Das Notizenobjekt«, beschriebenen Graph-Objekt. Während öffentlich von Seiten gepostete Notizen mit einem beliebigen Access Token gelesen werden können, ist für von Benutzern veröffentlichte Notizen die Berechtigung `user_notes` bzw. `friends_notes` notwendig.

Beispiel – Lesen aller Notizen des aktuellen Benutzers:

```
SELECT uid, note_id, title, content_html, content,
created_time, FROM note WHERE uid=me()
```

Beispiel – Lesen einer bestimmten Notiz:

```
SELECT uid, note_id, title, content_html, content,
created_time, FROM note WHERE note_id=122788341354
```


Feldname	Beschreibung	Typ	Indiziert
uid_id	Benutzer-ID des Erstellers der Notiz	int	ja
note_id	Objekt-ID der Notiz	int	ja
created_time	UNIX-Zeitstempel der Erstellung der Notiz	UNIX-Zeit-stempel	nein
updated_time	UNIX-Zeitstempel des letzten Updates der Notiz	UNIX-Zeit-stempel	nein
content	Inhalt der Notiz als Text	string	nein
content_html	Inhalt der Notiz als HTML-Text	string	nein
title	Titel der Notiz	string	nein

Tabelle 5.28 Felder der Note-Tabelle

5.2.23 Notification-Tabelle

Die FQL-Tabelle `notification` enthält die Benachrichtigungen des aktuellen Benutzers und entspricht damit der `notifications`-Verknüpfung des `user`-Objekts, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben. Das Lesen der Benachrichtigungen ist nur für den aktuellen Benutzer möglich und erfordert die erweiterte Berechtigung `manage_notifications`.

Beispiel – Abfrage der ungelesenen Benachrichtigungen des aktuellen Benutzers:

```
SELECT notification_id, sender_id, title_html, body_html, href
FROM notification
WHERE recipient_id=me() AND is_unread = 1 AND is_hidden = 0
```

Beispiel – Abfrage einer bestimmten Benachrichtigung – da die `notification_id` allein nicht eindeutig ist, muss die Abfrage gemeinsam mit der `recipient_id` erfolgen:

```
SELECT notification_id, sender_id, title_html, body_html, href
FROM notification
WHERE notification_id= 156574449 and recipient_id=me()
```

Feldname	Beschreibung	Typ	Indiziert
notification_id	ID der Benachrichtigung. Diese ID ist allein nicht eindeutig und muss daher immer gemeinsam mit dem Feld recipient_id abgefragt werden.	int	nein
sender_id	Benutzer-ID des Absenders	int	nein
recipient_id	Benutzer-ID des Empfängers, also des aktuellen Benutzers	int	ja
created_time	UNIX-Zeitstempel der Erstellung der Benachrichtigung	UNIX-Zeitstempel	nein
updated_time	UNIX-Zeitstempel der Erstellung der Benachrichtigung bzw. des Zeitpunktes, zu dem sie vom Empfänger versteckt oder sichtbar gemacht wurde	UNIX-Zeitstempel	nein
title_html	Benachrichtungstext als HTML-Text	string	nein
title_text	Benachrichtungstext	string	nein
body_html	zusätzlicher Inhalt als HTML-Text	string	nein
body_text	zusätzlicher Inhalt als Text	string	nein
href	URL, die mit der Benachrichtigung verknüpft ist	string	nein
app_id	ID der Anwendung, über die die Benachrichtigung erzeugt wurde	string	nein
is_unread	Zeigt an, ob die Benachrichtigung bereits gelesen wurde.	boolean	nein
is_hidden	Zeigt an, ob die Benachrichtigung vom Empfänger versteckt wurde.	boolean	nein
object_id	Objekt-ID der Benachrichtigung	int	nein

Tabelle 5.29 Felder der Notification-Tabelle

Feldname	Beschreibung	Typ	Indiziert
object_type	Typ des Objekts, auf das sich die Nachricht bezieht (photo, friend, stream ...)	string	nein
icon_url	URL zum Icon der Benachrichtigung (16 x 16 Pixel)	string	nein

Tabelle 5.29 Felder der Notification-Tabelle (Forts.)

5.2.24 Object-URL-Tabelle

Die FQL-Tabelle object_url bildet die Zuordnung von externen URLs zu Objekten im sozialen Graphen von Facebook ab und ist damit äquivalent zum in Abschnitt 3.2.6, »Das Domain-Objekt«, beschriebenen Objekt. Der Zugriff kann ohne Access Token erfolgen.

Beispiel – Abfrage der Objekt-ID anhand einer externen URL:

```
SELECT url, id, type, site FROM object_url
WHERE url = 'http://die.socialisten.at/2011/10/last-fm-scrobbler-for-facebook-open-graph-timeline/'
```

Beispiel – Abfrage der Objektinformationen anhand einer bekannten ID:

```
SELECT url, id, type, site FROM object_url
WHERE id = 230200637040140
```

Feldname	Beschreibung	Typ	Indiziert
url	externe URL des Objekts	string	ja
id	Objekt-ID des Objekts	int	ja
type	Typ des Objekts (page oder andere Open-Graph-Typen)	string	nein
site	normalisierte Domain der externen URL	string	nein

Tabelle 5.30 Felder der Object-URL-Tabelle

5.2.25 Page-Tabellen

Die FQL-Tabelle page bildet Seitenobjekte des sozialen Graphen ab, wie in Abschnitt 3.2.2, »Das Seitenobjekt«, beschrieben. Öffentliche Seiten können ohne Access Token, nicht-öffentliche oder geographisch eingeschränkt verfügbare Seiten mit dem Access Token eines berechtigten Benutzers eingelesen werden.

Beispiel – Lesen der Anzahl der Fans einer der ID nach bekannten Seite:

```
SELECT username, name, fan_count FROM page
WHERE page_id = 129842310398883
```

Beispiel – Lesen der Anzahl der Fans einer dem eindeutigen Namen nach bekannten Seite:

```
SELECT name, fan_count FROM page
WHERE username = 'diesocialisten'
```

Feldname	Beschreibung	Typ	Indiziert
page_id	Objekt-ID der Seite	int	ja
name	Name der Seite	string	ja
username	eindeutiger Kurzname der Seite	string	ja
description	Beschreibung der Seite	string	nein
categories	Seitenkategorien	Array	nein
is_community_page	Zeigt an, ob eine Seite eine Community-Seite ist.	boolean	nein
pic_small	URL zum kleinen Profilbild der Seite (maximal 50 x 150 Pixel)	string	nein
pic_big	URL zum großen Profilbild der Seite (maximal 200 x 600 Pixel)	string	nein
pic_square	URL zum quadratischen Profilbild der Seite (50 x 50 Pixel)	string	nein
pic	URL zum mittelgroßen Profilbild der Seite (maximal 100 x 300 Pixel)	string	nein
pic_large	URL zum größten Profilbild der Seite (maximal 396 x 1.188 Pixel)	string	nein

Tabelle 5.31 Felder der Page-Tabelle

Feldname	Beschreibung	Typ	Indiziert
page_url	URL zum Profil der Seite	string	nein
fan_count	Anzahl der Fans der Seite	int	nein
type	Typ der Seite	string	nein
website	URL zur Website der Seite	string	nein
has_added_app	Zeigt an, ob die Seite die Anwendung, von der die FQL-Abfrage gestellt wurde, als Reiter installiert hat.	boolean	nein
general_info	allgemeine Informationen zur Seite	string	nein
can_post	Zeigt an, ob der aktuelle Benutzer auf der Pinnwand der Seite posten darf.	boolean	nein
checkins	Anzahl der Checkins (nur für Seiten vom Typ PLACE)	int	nein
founded	nur für Seiten vom Typ COMPANY	string	nein
company_overview	nur für Seiten vom Typ COMPANY	string	nein
mission	nur für Seiten vom Typ COMPANY	string	nein
products	nur für Seiten vom Typ COMPANY	string	nein
location	nur für Seiten vom Typ PLACE	Array	nein
parking	nur für Seiten vom Typ BUSINESS oder PLACE	Array	nein
hours	nur für Seiten vom Typ BUSINESS oder PLACE	Array	nein
pharma_safety_info	nur für Seiten vom Typ PHARMACEUTICAL	string	nein
public_transit	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	string	nein
attire	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	string	nein

Tabelle 5.31 Felder der Page-Tabelle (Forts.)

Feldname	Beschreibung	Typ	Indiziert
payment_options	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	string	nein
culinary_team	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	string	nein
general_manager	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	string	nein
price_range	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	string	nein
restaurant_services	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	Array	nein
restaurant_specialities	nur für Seiten vom Typ RESTAURANT oder NIGHTLIFE	Array	nein
phone	Telefonnumer	string	nein
release_date	nur für Seiten vom Typ MOVIE	string	nein
genre	nur für Seiten vom Typ MOVIE	string	nein
starring	nur für Seiten vom Typ MOVIE	string	nein
screenplay_by	nur für Seiten vom Typ MOVIE	string	nein
directed_by	nur für Seiten vom Typ MOVIE	string	nein
produced_by	nur für Seiten vom Typ MOVIE	string	nein
studio	nur für Seiten vom Typ MOVIE	string	nein
awards	nur für Seiten vom Typ MOVIE	string	nein
plot_ouline	nur für Seiten vom Typ MOVIE	string	nein
season	nur für Seiten vom Typ TV SHOW	string	nein
network	nur für Seiten vom Typ TV SHOW	string	nein
schedule	nur für Seiten vom Typ TV SHOW	string	nein
written_by	nur für Seiten vom Typ TV SHOW	string	nein
band_members	nur für Seiten vom Typ MUSICIAN/BAND	string	nein

Tabelle 5.31 Felder der Page-Tabelle (Forts.)

Feldname	Beschreibung	Typ	Indiziert
hometown	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
current_location	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
record_label	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
booking_agent	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
press_contact	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
artists_we_like	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
influences	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
band_interests	Nur für Seiten vom Typ MUSICIAN/BAND	string	nein
bio	nur für Seiten vom Typ MUSICIAN/BAND	string	nein
affiliation	nur für Seiten vom Typ PEOPLE	string	nein
birthday	nur für Seiten vom Typ PEOPLE	string	nein
personal_info	nur für Seiten vom Typ PEOPLE	string	nein
personal_interests	nur für Seiten vom Typ PEOPLE	string	nein
built	nur für Seiten vom Typ CARS	string	nein
features	nur für Seiten vom Typ CARS	string	nein
mpg	nur für Seiten vom Typ CARS	string	nein

Tabelle 5.31 Felder der Page-Tabelle (Forts.)

Page-Admin-Tabelle

Die FQL-Tabelle page_admin enthält alle Benutzer, die Administrationsrechte für eine Seite besitzen, und entspricht damit der in Abschnitt 3.2.2, »Das Seitenobjekt«,

beschriebenen admins-Verknüpfung. Die Berechtigung manage_pages ist notwendig, um die administrierten Seiten des aktuellen Benutzers auszulesen.

Beispiel – Lesen aller Seiten, für die der aktuelle Benutzer Administrationsrechte besitzt:

```
SELECT page_id, type from page_admin WHERE uid=me()
```

Beispiel – Lesen aller Administratoren einer bestimmten Seite:

```
SELECT uid, type from page_admin WHERE page_id=129842310398883
```

Feldname	Beschreibung	Typ	Indiziert
uid	Benutzer-ID des Administrators	int	ja
page_ud	ID der Seite	int	ja
type	Typ der Seite	string	nein

Tabelle 5.32 Felder der Page-Admin-Tabelle

Page-Blocked-User-Tabelle

Die FQL-Tabelle page_blocked_users enthält alle durch Administratoren gesperrten Benutzer einer Seite und entspricht damit der in Abschnitt 3.2.2, »Das Seitenobjekt«, beschriebenen blocked-Verknüpfung. Die Berechtigung manage_pages ist notwendig, um die gesperrten Benutzer einer Seite auszulesen.

Beispiel – Lesen aller gesperrten Benutzer einer bestimmten Seite:

```
SELECT uid FROM page_blocked_user WHERE page_id = 129842310398883
```

Feldname	Beschreibung	Typ	Indiziert
page_id	ID der Seite	int	ja
uid	ID des gesperrten Benutzers	int	nein

Tabelle 5.33 Felder der Page-Blocked-User-Tabelle

Page-Fan-Tabelle

Die FQL-Tabelle page_fans bildet die Gefällt-mir-Beziehung zwischen Benutzern und Seiten ab und entspricht damit der in Abschnitt 3.2.1, »Das User-Objekt«, beschriebenen likes-Verknüpfung. Die Abfrage erfolgt bei öffentlichen Beziehungen

mit einem beliebigen Access Token, bei nicht-öffentlichen GEFÄLLT MIR mit der erweiterten Berechtigung `user_likes` bzw. `friend_likes`.

Hinweis: Da das einzige indizierte Feld der Tabelle `page_fans` die Spalte `uid` ist, kann sie lediglich dazu benutzt werden, alle oder eine bestimmte GEFÄLLT-MIR-Beziehung eines bestimmten Benutzers auszulesen. Ein Auslesen aller GEFÄLLT-MIR-Beziehungen einer einzelnen Seite ist hingegen nicht möglich!

Beispiel – Lesen aller GEFÄLLT-MIR-Beziehungen des aktuellen Benutzers:

```
SELECT page_id, profile_section FROM page_fan WHERE uid=me()
```

Beispiel – Prüfen, ob ein bestimmter Benutzer Fan einer bestimmten Seite ist:

```
SELECT page_id, profile_section FROM page_fan
WHERE page_id=129842310398883 AND uid=633616172
```

Feldname	Beschreibung	Typ	Indiziert
`uid`	Benutzer-ID	`int`	ja
`page_ID`	ID der Seite	`int`	nein
`type`	Typ der Seite	`string`	nein
`profile_section`	Gibt an, in welchem Profilabschnitt des Benutzers die Seite angezeigt wird (`music`, `activities` ...).	`string`	nein
`created_time`	UNIX-Zeitstempel der Herstellung der GEFÄLLT-MIR-Beziehung	UNIX-Zeitstempel	nein

Tabelle 5.34 Felder der Page-Fan-Tabelle

5.2.26 Permission-Tabelle

Die FQL-Tabelle `permission` bildet die Berechtigungen ab, die ein Benutzer der aktuellen Anwendung erteilt hat. Damit entspricht sie der in Abschnitt 3.2.1, »Das User-Objekt«, beschriebenen `permissions`-Verknüpfung. Erteilte Berechtigungen können immer nur für die eigene Anwendung und den aktuellen Benutzer ermittelt werden. Dafür ist ein Access Token des Benutzers oder der Anwendung notwendig.

Beispiel – Prüfen der Berechtigungen `manage_pages` und `publish_stream` für den aktuellen Benutzer:

```
SELECT manage_pages, publish_stream FROM permissions
WHERE uid = me()
```

Hinweis: Die Tabelle `permission` besitzt eine virtuelle Spalte je erteilter Berechtigung. Die Spalten sind dabei analog, wie in Abschnitt 2.6, »Erweiterte Zugriffsrechte (»Extended Permissions«)«, beschrieben, benannt und werden daher an dieser Stelle nicht vollständig wiederholt.

Feldname	Beschreibung	Typ	Indiziert
`uid`	Benutzer-ID	`int`	ja
`user_relationship`	Zeigt an, ob die Berechtigung erteilt wurde.	`boolean`	nein
`user_groups`	Zeigt an, ob die Berechtigung erteilt wurde.	`boolean`	nein
`user_events`	Zeigt an, ob die Berechtigung erteilt wurde.	`boolean`	nein
...	...	...	...
`PERMISSION_NAME`	Jede von der Facebook-Plattform unterstützte Berechtigung wird durch eine entsprechende FQL-Spalte repräsentiert.	`boolean`	nein

Tabelle 5.35 Felder der Permission-Tabelle

5.2.27 Fototabellen

In der FQL-Tabelle `photo` werden alle auf der Facebook-Plattform gespeicherten Fotos abgebildet. Das entsprechende Graph-Objekt wurde in Abschnitt 3.2.7, »Das Fotoobjekt«, beschrieben. Zum Zugriff auf die öffentlichen Fotos einer Seite ist ein beliebiges Access Token ausreichend, während private Fotos von Benutzern ein Token mit der Berechtigung `user_photos` oder `friends_photos` erfordern.

Beispiel – Lesen aller Fotos eines bestimmten Albums:

```
SELECT pid, src FROM photo WHERE aid='20531316728_324257'
```

Beispiel – Lesen eines bestimmten Fotos:

```
SELECT pid, src FROM photo WHERE pid='20531316728_7844072'
```

Feldname	Beschreibung	Typ	Indiziert
pid	ID des Fotos	string	ja
aid	ID des Albums, in dem das Foto gespeichert ist	string	ja
owner	Benutzer- oder Seiten-ID des Erstellers des Fotos	int	nein
src_small	URL zu Thumbnail des Fotos (maximal 75 x 225 Pixel)	string	nein
src_small_width	Breite des Thumbnails in Pixeln	int	nein
src_small_height	Höhe des Thumbnails in Pixeln	int	nein
src_big	URL zum Foto in höchster verfügbarer Auflösung (maximal 720 x 720 Pixel, ab 1. März 2012: maximal 960 x 960 Pixel)	string	nein
src_big_width	Breite des voll aufgelösten Fotos	int	nein
src_big_height	Höhe des voll aufgelösten Fotos	int	nein
src	URL zur Album-Ansicht des Fotos (maximal 130 x 130 Pixel)	string	nein
src_width	Breite des Fotos in der Album-Ansicht	int	nein
src_width	Höhe des Fotos in der Album-Ansicht	int	nein
link	URL zum Foto	string	nein
caption	Beschriftung des Fotos	string	nein
created	UNIX-Zeitstempel der Veröffentlichung des Fotos	UNIX-Zeitstempel	nein
modified	UNIX-Zeitstempel der letzten Änderung des Fotos	UNIX-Zeitstempel	nein
position	Position des Fotos im Album	int	nein
object_id	Objekt-ID des Fotos	int	ja

Tabelle 5.36 Felder der Fototabelle

Feldname	Beschreibung	Typ	Indiziert
album_object_id	Objekt-ID des Albums, in dem das Foto gespeichert ist	int	ja
images	Enthält alle URLs sowie Breiten- und Höhenangaben zu allen verfügbaren Auflösungen des Fotos.	Array	nein

Tabelle 5.36 Felder der Fototabelle (Forts.)

Photo-Tag-Tabelle

Die FQL-Tabelle `photo_tag` enthält Fotomarkierungen (*Tags*) und entspricht damit der `tags`-Verknüpfung des `photo`-Objekts, wie in Abschnitt 3.2.7, »Das Fotoobjekt«, beschrieben. Zum Lesen von Fotomarkierungen ist die Berechtigung `user_photos` bzw. `friends_photos` erforderlich.

Beispiel – Lesen aller Fotomarkierungen des aktuellen Benutzers:

```
SELECT pid, object_id FROM photo_tag WHERE subject = me()
```

Beispiel – Lesen aller auf einem bestimmten Foto markierten Benutzer:

```
SELECT pid, object_id, subject FROM photo_tag
WHERE pid = 3129746340803180882
```

Neben Fotomarkierungen wird `photo_tag` auch genutzt, um die Fotos, die mit einer Gruppe oder einem Event verknüpft sind, auszulesen. Dazu muss als `subject` lediglich die ID der Gruppe oder des Events abgefragt werden.

Beispiel – Lesen aller Fotos, die mit einer bestimmten Gruppe verknüpft sind:

```
SELECT pid, object_id FROM photo_tag
WHERE subject = 269627173053176
```

Feldname	Beschreibung	Typ	Indiziert
pid	ID des Fotos	string	ja
subject	ID des markierten Benutzers bzw. der Veranstaltung oder Gruppe	int	ja
object_id	Objekt-ID des Fotos	int	nein
text	Beschriftung der Markierung	string	nein

Tabelle 5.37 Felder der Photo-Tag-Tabelle

Feldname	Beschreibung	Typ	Indiziert
xcoord	X-Koordinate der Markierung als prozentueller Abstand vom linken Rand	float	nein
ycoord	Y-Koordinate der Markierung als prozentueller Abstand vom linken Rand	float	nein

Tabelle 5.37 Felder der Photo-Tag-Tabelle (Forts.)

5.2.28 Place-Tabelle

Die FQL-Tabelle `place` bildet `page`-Objekte im sozialen Graphen ab, die eine geographische Verortung über die Adresse erlauben und somit als »Ort« für Checkins zur Verfügung stehen. Da Orte im sozialen Graphen mittlerweile immer als Seitenobjekt gespeichert werden, existiert für sie kein eigener Objekttyp.

Beispiel – Lesen der Detailinformationen eines Ortes:

```
SELECT name, description, geometry,
latitude, longitude, checkin_count
FROM place WHERE page_id = 163483520335945
```

Feldname	Beschreibung	Typ	Indiziert
page_id	ID des Ortes bzw. der Seite	int	ja
name	Name des Ortes	string	nein
description	Beschreibung des Ortes	string	nein
geometry	Array, das in den Feldern `type` und `coordinates` die geographische Verortung des Ortes enthält	Array	nein
latitude	Breitengrad	float	nein
longitude	Längengrad	float	nein
checkin_count	Anzahl der Checkins	int	nein
display_subtext	Kurzinfo zum Ort (enthält den Typ des Ortes und die Anzahl der Checkins)	string	nein

Tabelle 5.38 Felder der Place-Tabelle

5.2.29 Privacy-Tabellen

Mithilfe der FQL-Tabelle `privacy` können die Privatsphäre-Einstellungen ermittelt werden, mit denen der aktuelle Benutzer Videos, Fotos, Alben, Notizen oder Links veröffentlicht hat. Die Privatsphäre-Einstellungen dieser Objekte können nur für den aktuellen Benutzer ermittelt werden, dessen Access Token beim Zugriff notwendig ist.

Beispiel – Lesen der Privatsphäre-Einstellungen eines bestimmten Objekts:

```
SELECT value, description FROM privacy
WHERE id = 10150146071791729
```

Feldname	Beschreibung	Typ	Indiziert
`id`	Objekt-ID des Videos, Fotos, Albums, der Notiz oder des Links	`int`	ja
`object_id`	Alias für die Spalte `id`	`int`	ja
`value`	Privatsphäre-Einstellung – `EVERYONE`, `CUSTOM`, `ALL_FRIENDS`, `NETWORKS_FRIENDS` oder `FRIENDS_OF_FRIENDS`	`string`	nein
`description`	Textbeschreibung der Privatsphäre-Einstellungen	`string`	nein
`allow`	kommaseparierte Liste an numerischen IDs von Benutzern oder Freundeslisten, denen Zugriff auf das Objekt gewährt wurde	`string`	nein
`deny`	kommaseparierte Liste an numerischen IDs von Benutzern oder Freundeslisten, denen Zugriff auf das Objekt verwehrt wurde	`string`	nein
`owner_id`	Benutzer-ID des Besitzers des Objekts	`int`	nein
`networks`	ID des Benutzernetzwerks, das Zugriff auf das Objekt hat	`int`	nein
`friends`	Zeigt an, welchen Freunden Zugriff auf das Objekt gewährt wurde – `EVERYONE`, `NETWORKS_FRIENDS`, `FRIENDS_OF_FRIENDS`, `ALL_FRIENDS`, `SOME_FRIENDS`, `SELF`, `NO_FRIENDS`.	`string`	nein

Tabelle 5.39 Felder der Privacy-Tabelle

Privacy-Setting-Tabelle

Die FQL-Tabelle `privacy_setting` gibt Auskunft über die Standard-Privatsphäre-Einstellungen, die der aktuelle Benutzer für die eigenen Anwendung gesetzt hat. Diese Einstellungen können nur für den aktuellen Benutzer ermittelt werden, ein entsprechendes Access Token ist beim Zugriff notwendig.

Hinweis: Derzeit kennt Facebook nur eine anwendungsbezogene Einstellung namens `default_stream_privacy`. Diese regelt die Privatsphäre-Einstellungen aller Veröffentlichungen auf der Pinnwand des Benutzers, die durch die Anwendung vorgenommen werden.

Beispiel – Lesen der Standard-Privatsphäre-Einstellungen für die aktuelle Anwendung:

```
SELECT name, value, description, allow, deny, networks, friends
FROM privacy_setting WHERE name = 'default_stream_privacy'
```

Feldname	Beschreibung	Typ	Indiziert
`Name`	Name der Privatsphäre-Einstellungen – derzeit immer `default_stream_privacy`	`string`	ja
`value`	Standard-Privatsphäre-Einstellung – `EVERYONE`, `CUSTOM`, `ALL_FRIENDS`, `NETWORKS_FRIENDS` oder `FRIENDS_OF_FRIENDS`	`string`	nein
`description`	Textbeschreibung der Standard-Privatsphäre-Einstellungen	`string`	nein
`allow`	kommaseparierte Liste an numerischen IDs von Benutzern oder Freundeslisten, denen Zugriff auf Objekte gewährt wird	`string`	nein
`deny`	kommaseparierte Liste an numerischen IDs von Benutzern oder Freundeslisten, denen Zugriff auf Objekte verwehrt wird	`string`	nein

Tabelle 5.40 Felder der Privacy-Setting-Tabelle

Feldname	Beschreibung	Typ	Indiziert
networks	ID des Benutzernetzwerks, das Zugriff auf Objekte hat	int	nein
friends	Zeigt an, welchen Freunden Zugriff auf veröffentlichte Objekte gewährt wird: EVERYONE, NETWORKS_FRIENDS, FRIENDS_OF_FRIENDS, ALL_FRIENDS, SOME_FRIENDS, SELF, NO_FRIENDS	string	nein

Tabelle 5.40 Felder der Privacy-Setting-Tabelle (Forts.)

5.2.30 Profile-Tabelle

Die FQL-Tabelle `profile` bildet Profildaten von unterschiedlichen Objekten wie Benutzern, Gruppen, Seiten, Veranstaltungen und Anwendungen zusammengefasst in einer Tabelle ab. Damit entspricht sie keinem eigenen Objekttypen im Graphen von Facebook, sondern nimmt eine Art übergeordnete Rolle ein. Bei der Abfrage stellen sich je nach Objekttyp folgende Voraussetzungen an das verwendete Access Token:

- Benutzerobjekte können ohne Access Token abgefragt werden, mit Ausnahme des Feldes `url`, das ein beliebiges Token erfordert.
- Öffentliche Gruppen können ohne Access Token, private Gruppen mit dem Token eines Mitglieds abgefragt werden.
- Für die Abfrage von Seiten ist kein Access Token notwendig.
- Für die Abfrage von Anwendungen ist kein Access Token notwendig.
- Öffentliche Veranstaltungen können ohne Access Token, private Veranstaltungen mit dem Token eines eingeladenen Benutzers abgefragt werden.

Beispiel – Abfrage des Profils einer Seite anhand ihres eindeutigen Namens:

```
SELECT id, name, url, pic FROM profile
WHERE username = 'diesocialisten'
```

Feldname	Beschreibung	Typ	Indiziert
id	Objekt-ID von Benutzer, Seite, Anwendung, Veranstaltung oder Gruppe	int	ja
can_post	Zeigt an, ob der aktuelle Benutzer auf der Pinnwand des abgefragten Objekts posten darf (nur unter Verwendung eines Access Tokens gesetzt).	boolean	nein
name	Name des abgefragten Objekts	string	nein
pic_small	URL zum kleinen Profilbild des Objekts (maximal 50 x 150 Pixel)	string	nein
pic_big	URL zum großen Profilbild des Objekts (maximal 200 x 600 Pixel)	string	nein
pic_square	URL zum quadratischen Profilbild des Objekts (50 x 50 Pixel)	string	nein
pic	URL zum mittelgroßen Profilbild des Objekts (maximal 100 x 300 Pixel)	string	nein
pic_crop	URL zum großen quadratischen Profilbild des Objekts (mindestens 100 x 100 Pixel)	string	nein
type	Typ des Objekts (user, group, application, event oder page)	string	nein

Tabelle 5.41 Felder der Profile-Tabelle

5.2.31 Question-Tabellen

In der FQL-Tabelle question werden auf Pinnwänden veröffentlichte Fragen gespeichert. Die Tabelle entspricht damit dem in Abschnitt 3.2.12, »Das Fragenobjekt«, beschriebenen Objekt des sozialen Graphen. Der Zugriff auf Fragen erfordert in jedem Fall ein Access Token mit der Berechtigung user_questions bzw. friends_questions.

Beispiel – Lesen aller gestellten Fragen des aktuellen Benutzers:

```
SELECT id, question FROM question WHERE owner=me()
```

Feldname	Beschreibung	Typ	Indiziert
id	ID der Frage	int	ja
owner	ID des Benutzers, der die Frage veröffentlicht hat	int	ja
question	Text der Frage	string	nein
created_time	UNIX-Zeitstempel der Erstellung der Frage	UNIX-Zeitstempel	nein
updated_time	UNIX-Zeitstempel des letzten Updates der Frage	UNIX-Zeitstempel	nein

Tabelle 5.42 Felder der Question-Tabelle

Question-Option-Tabelle

Während `question` nur den Fragetext enthält, müssen die einzelnen Antwortoptionen aus der FQL-Tabelle `question_option` abgefragt werden. Auch hierzu ist ein Access Token mit der Berechtigung `user_questions` bzw. `friends_questions` erforderlich.

Beispiel – Lesen aller Antwortoptionen einer bestimmten Frage:

```
SELECT id, name FROM question_option
WHERE question_id = 326069260742300
```

Feldname	Beschreibung	Typ	Indiziert
id	ID der Antwortoption	int	ja
question_id	ID der zugehörigen Frage	int	ja
name	Text der Antwortoption	string	nein
votes	Anzahl der Stimmen	int	nein
object_id	optionale ID einer mit der Antwortoption verknüpften Seite	int	nein
owner	ID des Benutzers, der die Antwortoption veröffentlicht hat	int	nein
created_time	UNIX-Zeitstempel der Erstellung der Antwortoption	UNIX-Zeitstempel	nein

Tabelle 5.43 Felder der Question-Option-Tabelle

Question-Option-Votes-Tabelle

Die einzelnen Stimmen, die eine Antwortoption in einer Frage erhalten hat, können über die Tabelle `question_option_votes` abgefragt werden. Auch hierzu ist ein Access Token mit der Berechtigung `user_questions` bzw. `friends_questions` erforderlich.

Beispiel – Lesen aller Stimmen einer bestimmten Antwortoption:

```
SELECT voter_id FROM question_option_votes
WHERE option_id = 201037246642437
```

Feldname	Beschreibung	Typ	Indiziert
`option_id`	ID der Antwortoption	`int`	ja
`voter_id`	ID des Benutzers, der für diese Option gestimmt hat	`int`	nein

Tabelle 5.44 Felder der Question-Option-Votes-Tabelle

5.2.32 Review-Tabelle

Die FQL-Tabelle `review` speichert Benutzer-Reviews zu Plattform-Anwendungen und entspricht dem in Abschnitt 3.2.17, »Das Review-Objekt«, beschriebenen Objekttyp. Da Reviews öffentlich sind, ist zum Zugriff kein Access Token notwendig.

Beispiel – Lesen aller Reviews zu einer bestimmten Anwendung:

```
SELECT review_id, message, rating FROM review
WHERE reviewee_id = 32788395891
```

Feldname	Beschreibung	Typ	Indiziert
`reviewee_id`	ID der bewerteten Anwendung	`int`	ja
`reviewer_id`	ID des Benutzers, der die Bewertung veröffentlicht hat	`int`	ja
`message`	Text der Bewertung	`string`	nein
`created_time`	UNIX-Zeitstempel der Erstellung der Bewertung	UNIX-Zeit-stempel	nein
`rating`	numerische Bewertung	`int`	nein

Tabelle 5.45 Felder der Review-Tabelle

5.2.33 Standard-Friend-Info-Tabelle

Die FQL-Tabelle `standard_friend_info` kann verwendet werden, um die Freundeslisten von Benutzern der aktuellen Anwendung abzufragen. Dazu ist kein Benutzer-Access-Token notwendig, sondern das entsprechenden Access Token der Anwendung. *Hinweis:* Da die Abfrage somit unabhängig von einem aktiv angemeldeten Benutzer erfolgt, ist die Information aus dieser Tabelle nur für den internen Gebrauch durch die Anwendung, nicht aber zur Anzeige in der Applikation gedacht.

Beispiel – Lesen aller Freunde eines bestimmten Anwendungsbenutzers:

```
SELECT uid2 FROM standard_friend_info WHERE uid1 = 633616172
```

Feldname	Beschreibung	Typ	Indiziert
`uid1`	ID des ersten abgefragten Anwendungsbenutzers	`int`	ja
`uid2`	ID des zweiten abgefragten Anwendungsbenutzers	`int`	ja

Tabelle 5.46 Felder der Standard-Friend-Info-Tabelle

5.2.34 Standard-User-Info-Tabelle

Ähnlich wie `standard_friend_info` kann die FQL-Tabelle `standard_user_info` verwendet werden, um Informationen über die Benutzer, die eine Anwendung bereits installiert haben, mit dem Anwendungs-Access-Token abzufragen. Auch hier gilt: Informationen aus dieser Tabelle sind nur für den internen Gebrauch der Anwendung und nicht zur Anzeige in der Applikation gedacht.

Beispiel – Lesen der Benutzerinfos eines bestimmten Anwendungsbenutzers:

```
SELECT name, username, sex, locale FROM standard_user_info
WHERE uid = 609190863
```

Feldname	Beschreibung	Typ	Indiziert
`uid`	ID des abgefragten Anwendungsbenutzers	`int`	ja
`name`	voller Name des Benutzers	`string`	ja
`username`	eindeutiger Benutzername	`string`	ja

Tabelle 5.47 Felder der Standard-User-Info-Tabelle

Feldname	Beschreibung	Typ	Indiziert
third_party_id	eine anonymisierte, eindeutige Benutzerkennung	string	ja
first_name	Vorname	string	nein
last_name	Nachname	string	nein
locale	Lokalisierungsinformationen des Benutzers	string enthält den ISO-Sprach- und Landescode, z. B. de_DE	nein
affiliations	Netzwerk-Zugehörigkeit des Benutzers	Array	nein
profile_url	URL zum Profil des Benutzers	string	nein
timezone	Zeitzone des Benutzers als Differenz zu UTC	string	nein
birthday	Geburtstag	string	nein
sex	Geschlecht	string	nein
proxied_email	Proxy-E-Mail-Adresse	string	nein
current_location	aktuelle Heimatstadt	string	nein
allowed_restrictions	kommaseparierte Liste der demographischen Zugriffseinschränkungen, die der Benutzer erfüllt – derzeit nur alcohol	string	nein

Tabelle 5.47 Felder der Standard-User-Info-Tabelle (Forts.)

5.2.35 Status-Tabelle

Die FQL-Tabelle `status` speichert auf der Pinnwand veröffentlichte Status-Updates von Benutzern und entspricht damit der `statuses`-Verknüpfung des in Abschnitt 3.2.1, »Das User-Objekt«, beschriebenen Objekts. Zum Lesen dieser Tabelle ist die erweiterte Berechtigung `read_stream` notwendig.

Beispiel – Lesen der Status-Updates des aktuellen Benutzers:

```
SELECT status_id, message FROM status WHERE uid=me()
```

Beispiel – Lesen eines bestimmten Status-Updates:

```
SELECT status_id, message FROM status
WHERE status_id = 10150453764222297
```

Feldname	Beschreibung	Typ	Indiziert
uid	ID des abgefragten Benutzers	int	ja
status_id	ID des Status-Updates	int	ja
time	UNIX-Zeitstempel der Veröffentlichung des Updates	UNIX-Zeitstempel	nein
source	ID der Anwendung, über die das Update veröffentlicht wurde	int	nein
message	Text des Status-Updates	string	nein

Tabelle 5.48 Felder der Status-Tabelle

5.2.36 Stream-Tabellen

Die FQL-Tabelle `stream` repräsentiert die veröffentlichten Einträge einer Pinnwand und entspricht damit der `feed`-Verknüpfung der Objekttypen `user`, `page`, `application`, `event` und `group`. Zum Lesen der `stream`-Tabelle ist ein Access Token mit `read_stream`-Berechtigung notwendig. Um auch das Statistik-Feld `impressions` von der Pinnwand einer Seite zu lesen, ist zusätzlich die `read_insights`-Berechtigung notwendig.

Hinweis: Abfragen auf die Tabelle `stream` liefern grundsätzlich nur maximal 50 Postings und Postings der letzten 30 Tage. Durch Aufnahme des Feldes `created_time` in die `WHERE`-Klausel können aber beliebige Zeiträume abgefragt werden.

Beispiel – Abfrage der Pinnwand-Einträge des aktuellen Benutzers:

```
SELECT post_id, actor_id, target_id, message
FROM stream WHERE source_id = me()
```

Feldname	Beschreibung	Typ	Indiziert
post_id	ID des Pinnwand-Postings	string	ja
viewer_id	ID des aktuellen Benutzers, der die Abfrage stellt	int	nein

Tabelle 5.49 Felder der Stream-Tabelle

Feldname	Beschreibung	Typ	Indiziert
app_id	ID der Anwendung, über die das Posting veröffentlicht wurde	int	nein
source_id	ID des Benutzers, der Seite, Anwendung, Veranstaltung oder Gruppe, deren Pinnwand abgefragt wird	int	ja
updated_time	UNIX-Zeitstempel des letzten Updates des Postings durch einen Kommentar	UNIX-Zeitstempel	nein
created_time	UNIX-Zeitstempel der Veröffentlichung des Postings	UNIX-Zeitstempel	nein
filter_key	Filterkriterium, mit dem die Pinnwand abgefragt werden soll (siehe Stream-Filter-Tabelle)	string	ja
attribution	Name der Anwendung, über die das Posting veröffentlicht wurde	string	nein
actor_id	ID des Benutzers, der Seite oder Anwendung, die das Posting veröffentlicht hat	int	nein
target_id	ID des Benutzers, der Seite, Gruppe, Anwendung oder Veranstaltung, an die das Posting gerichtet ist	int	nein
message	Text des Postings	string	nein
app_data	optional mit dem Posting verknüpfte, zusätzliche Daten	Array	nein
action_links	Enthält Text und URL der Action-Links.	Array	nein
attachment	Anhang zum Posting	Array	nein
comments	Enthält die Kommentare zum Posting als Objekt mit den Feldern can_remove (zeigt an, ob der aktuelle Benutzer Kommentare löschen kann), can_post (zeigt an, ob der aktuelle Benutzer Kommentare posten kann) und comment_list (Array mit den Kommentaren).	Array	nein

Tabelle 5.49 Felder der Stream-Tabelle (Forts.)

Feldname	Beschreibung	Typ	Indiziert
impressions	Gibt an, wie oft das Posting anderen Benutzern angezeigt wurde (benötigt read_insights-Berechtigung).	int	nein
likes	Enthält die Gefällt-mir-Einträge zum Posting als Objekt mit den Feldern href (URL zu einer Seite, die alle Gefällt mir anzeigt), count (Anzahl der Gefällt mir), sample (Array mit den Gefällt mir), friends (Array von Freunden des aktuellen Benutzers, die Gefällt mir angeklickt haben), user_likes (zeigt an, ob der aktuelle Benutzer auf Gefällt mir geklickt hat), can_like (zeigt an, ob bei dem Posting auf Gefällt mir geklickt werden kann).	Array	nein
privacy	Privatsphäre-Einstellungen des Postings	Array	nein
permalink	Link zum Posting	string	nein
xid	Pinnwände von Live-Stream-Plugins enthalten in diesem Feld die mit der Box verknüpfte xid.	string	nein
tagged_ids	IDs der Benutzer, Seiten, Veranstaltungen etc., die im Posting markiert wurden	Array	nein
message_tags	Array, das die Markierungen von Benutzern, Seiten, Veranstaltungen etc. mit ihrer genauen Position im Text angibt – name (Name der Markierung), offset (Anzahl der Zeichen ab Beginn), length (Länge der Markierung)	Array	nein
description	Text von implizit generierten Pinnwand-Postings	string	nein
description_tags	Array, das die Markierungen von Benutzern, Seiten, Veranstaltungen etc. mit ihrer genauen Position in implizit generierten Postings angibt – name (Name der Markierung), offset (Anzahl der Zeichen ab Beginn), length (Länge der Markierung)	Array	nein

Tabelle 5.49 Felder der Stream-Tabelle (Forts.)

Stream-Filter-Tabelle

Die `stream_filter`-Tabelle enthält alle Filtermöglichkeiten, die einem bestimmten Benutzer im Frontend von Facebook.com zur Verfügung stehen. Diese Filter können mit einem beliebigen Access Token, aber nur für den aktuellen Benutzer ausgelesen werden.

Beispiel – Lesen aller Filtermöglichkeiten des aktuellen Benutzers:

```
SELECT filter_key,name,type
FROM stream_filter WHERE uid=me()
```

Das Ergebnis dieser Abfrage sieht etwa so aus:

```
{
  "data": [
    {
      "filter_key": "nf",
      "name": "News Feed",
      "type": "newsfeed",
    },
    {
      "filter_key": "fl_10150456156070864",
      "name": "Familie",
      "type": "friendlist",
    },
    {
      "filter_key": "fl_10150387764450864",
      "name": "Arbeitskollegen",
      "type": "friendlist",
    },
    ...
```

Listing 5.2 JSON-Array mit Ergebnissen der Stream-Filter-Tabelle

Feldname	Beschreibung	Typ	Indiziert
`uid`	ID des Benutzers, dessen Filter abgefragt werden	`int`	ja
`filter_key`	ID des Filter	`string`	ja
`name`	Name des Filters, z. B. »News Feed« oder der Name einer Freundesliste	`string`	nein

Tabelle 5.50 Felder der Stream-Filter-Tabelle

Feldname	Beschreibung	Typ	Indiziert
rank	Reihenfolge des Filters im Frontend von Facebook.com	int	nein
icon_url	URL zum Icon des Filters. Entspricht bei Anwendungsfiltern dem Icon der Anwendung.	string	nein
is_visible	Zeigt an, ob der Filter auf der Startseite von Facebook.com sichtbar ist.	boolean	nein
type	Typ des Filters (application, newsfeed, friendlist, network, public_profiles)	string	nein
value	numerische ID des Filtertyps	int	nein

Tabelle 5.50 Felder der Stream-Filter-Tabelle (Forts.)

Beispiel – In Kombination mit der stream-Tabelle und deren filter_key-Spalte kann so etwa der Newsfeed (die Facebook-Startseite) eines bestimmten Benutzers ausgelesen werden:

```
SELECT post_id, actor_id, target_id, message
FROM stream WHERE filter_key = 'nf'
```

Beispiel – Auslesen aller Pinnwand-Einträge von Seiten, dessen Fan der aktuelle Benutzer ist (entspricht dem Newsfeed, eingeschränkt auf Seiten):

```
SELECT post_id, actor_id, target_id, message
FROM stream WHERE filter_key = 'pp'
```

Stream-Tag-Tabelle

Die FQL-Tabelle stream_tag bildet ab, wenn Benutzer oder Seiten in Wall-Postings markiert wurden. Während Markierungen in öffentlichen Postings mit einem beliebigen Access Token gelesen werden können, ist für private Postings die Berechtigung read_stream notwendig.

Beispiel – Lesen aller Postings, in denen der aktuelle Benutzer markiert wurde:

```
SELECT post_id,actor_id FROM stream_tag
WHERE target_id=me()
```

Beispiel – Lesen aller Postings, in denen der aktuelle Benutzer andere Seiten oder Benutzer markiert hat:

```
SELECT post_id,actor_id FROM stream_tag
WHERE actor_id=me()
```


Feldname	Beschreibung	Typ	Indiziert
post_id	ID des Postings	int	ja
actor_id	ID des Benutzers, der das Posting veröffentlicht hat	int	ja
target_id	ID des Benutzers oder der Seite, die im Posting markiert wurde	int	ja

Tabelle 5.51 Felder der Stream-Tag-Tabelle

5.2.37 Thread-Tabelle

Die FQL-Tabelle `thread` enthält die Konversationen im Postfach des aktuellen Benutzers. Die Konversationen können nur für den aktuellen Benutzer gelesen werden und erfordern ein Access Token mit der `read_mailbox`-Berechtigung. Die Tabelle entspricht dem in Abschnitt 3.2.19, »Das Konversationenobjekt«, beschriebenen Graph-Objekt `thread`.

Beispiel – Lesen aller Konversationen des Inbox-Ordners:

```
SELECT thread_id, subject, recipients FROM thread
WHERE folder_id = 0
```

Feldname	Beschreibung	Typ	Indiziert
thread_id	ID der Konversation	int	ja
folder_id	ID des Nachrichten-Ordners – 0 (Inbox), 1 (Outbox) oder 4 (Updates)	int	ja
subject	Betreff der Konversation	string	nein
recipients	Benutzer-IDs der Empfänger der Konversation	Array	nein

Tabelle 5.52 Felder der Thread-Tabelle

Feldname	Beschreibung	Typ	Indiziert
updated_time	UNIX-Zeitstempel des letzten Updates der Konversation	UNIX-Zeit-stempel	nein
parent_ message_id	Wenn diese Konversation von einer Nachricht abstammt, enthält dieses Feld die ID der Nachricht, ansonsten 0.	string	nein
parent_ thread_id	Wenn diese Konversation von einer anderen Konversation abstammt, enthält dieses Feld die ID der Konversation, ansonsten 0.	int	nein
message_count	Anzahl der Nachrichten in dieser Konversation	int	nein
snippet	kurze Textvorschau auf die neueste Nachricht der Konversation	string	nein
snippet_ author	Benutzer-ID des Erstellers der neuesten Nachricht	int	nein
object_id	ID des Objekts, das die Nachricht versandt hat. 0, wenn die Nachricht nicht von einem Objekt stammt.	int	nein
unread	Anzahl der ungelesenen Nachrichten in der Konversation	int	nein
viewer_id	Benutzer-ID des Besitzers der Konversation bzw. Besitzers des abgefragten Postfachs	int	nein

Tabelle 5.52 Felder der Thread-Tabelle (Forts.)

5.2.38 Translation-Tabelle

Die FQL-Tabelle `translation` enthält die Zeichenketten-Übersetzungen zur Internationalisierung der aktuellen Facebook-Anwendung. Sie kann mit dem Anwendungs-Access-Token gelesen werden.

Beispiel – Lesen aller Übersetzungen der aktuellen Anwendung für die Lokalisierung `de_DE`:

```
SELECT native_hash, pre_hash_string, native_string, description,
translation FROM translation WHERE locale = 'de_DE'
```

Feldname	Beschreibung	Typ	Indiziert
locale	Lokalisierung, die abgefragt werden soll	string	ja
native_hash	Interne Hash-Abbildung der übersetzten Zeichenkette. Der Hash wird mit dem Tiger128,3-Algorithmus aus dem Feld pre_hash_string errechnet.	string	ja
native_string	Die ursprüngliche Zeichenkette in der primären Sprache der Anwendung	string	nein
description	Beschreibungstext zur Zeichenkette. Diese Beschreibung wird vom Entwickler vergeben und soll das Zuordnen einer Zeichenkette erleichtern.	string	nein
translation	Übersetzung der in native_string enthaltenen Zeichenkette in der abgefragten locale	string	nein
approval_status	Zeigt an, ob die Übersetzung bestätigt wurde – auto-approved, approved oder unapproved.	string	nein
pre_hash_string	zusammengesetztes Feld aus native_string, gefolgt von drei Doppelpunkten, dem Feld description und einem abschließenden Doppelpunkt	string	nein
best_string	Enthält die übersetzte Zeichenkette, die Benutzern in der entsprechenden locale angezeigt wird.	string	nein

Tabelle 5.53 Felder der Translation-Tabelle

5.2.39 Unified-Message-Tabellen

Hinweis: Facebook bereitet derzeit eine Überarbeitung des Nachrichten-Postfachs unter dem Titel *Unified Messaging* vor. Die Tabellen unified_message, unified_thread, unified_thread_action und unified_thread_count werden künftig den Zugriff auf das neue Postfach erlauben. Zum aktuellen Zeitpunkt sind diese FQL-Tabellen aber nur für Benutzer zugänglich, die als Developer in der aktuellen Anwendung registriert sind. Bis zum öffentlichen Roll-out von Unified Messaging dürfen daher ausschließlich die bereits behandelten Tabellen message und thread zum Zugriff auf das Postfach des aktuellen Benutzers verwendet werden!

Der Zugriff auf die Tabellen des neuen Unified Messaging wird ausschließlich für den aktuellen Benutzer möglich sein und erfordert die Berechtigung `read_mailbox`.

Unified-Thread-Tabelle

Die FQL-Tabelle `unified_thread` erlaubt den Zugriff auf die Konversationen im Unified-Messaging-Postfach.

Feldname	Beschreibung	Typ	Indiziert
`action_id`	Numerische Versions-ID der Konversation – diese ID spiegelt den Status einer Konversation zu einem bestimmten Zeitpunkt wider und wird laufend erhöht.	`string`	nein
`archived`	Zeigt an, ob die Konversation archiviert wurde.	`boolean`	ja
`can_reply`	Zeigt an, ob der aktuelle Benutzer in dieser Konversation antworten kann.	`boolean`	nein
`folder`	Name des Ordners (`inbox`, `other` oder `spam`)	`string`	nein
`former_participants`	Enthält eine Liste aller Benutzer (Felder `name`, `id`, `email`), die die Konversation verlassen haben.	Array	nein
`has_attachments`	Zeigt an, ob die Konversation einen Anhang enthält.	`boolean`	nein
`is_subscribed`	Zeigt an, ob der aktuelle Benutzer die Konversation abonniert hat (nur enthalten bei Konversationen mit mehreren Teilnehmern).	`boolean`	nein
`last_visible_add_action_id`	Versions-ID, die einer der folgenden Aktionen entspricht: `new message in thread`, `new thread participant`, `participant left the thread`	`string`	nein
`name`	Name der Konversation	`string`	nein
`num_messages`	Anzahl der Nachrichten in dieser Konversation	`int`	nein
`num_unread`	Anzahl der ungelesenen Nachrichten in dieser Konversation	`int`	nein

Tabelle 5.54 Felder der Unified-Thread-Tabelle

Feldname	Beschreibung	Typ	Indiziert
object_participants	Liste der Konversationsteilnehmer, die eine Seite, eine Gruppe oder eine Veranstaltung sind	Array	nein
participants	Liste der Konversationsteilnehmer, die Benutzer sind (Felder name, email, id)	Array	nein
senders	Liste der Konversationsteilnehmer, die Benutzer sind und eine Nachricht in der Konversation gesendet haben (Felder name, email, id)	Array	nein
single_recipient	Bei Konversationen mit nur zwei Teilnehmern enthält dieses Feld die ID des anderen Teilnehmers.	int	ja
snippet	Kurztext zur Voransicht	string	nein
snippet_sender	Ersteller der Nachricht, die im Feld snippet angezeigt wird (Felder name, email, id)	Array	nein
thread_id	numerische ID der Konversation	string	ja
thread_participants	Liste der Konversationsteilnehmer, die Benutzer sind (Felder name, email, id)	Array	nein
timestamp	UNIX-Zeitstempel des letzten Updates der Konversation	UNIX-Zeitstempel	ja
unread	Zeigt an, ob die Konversation ungelesene Nachrichten enthält.	boolean	ja

Tabelle 5.54 Felder der Unified-Thread-Tabelle (Forts.)

Beispiel – Lesen aller Konversationen mit dem Tag inbox mithilfe der Funktion has_tags():

```
SELECT subject, snippet FROM unified_thread
WHERE has_tags('inbox')
```

Beispiel – Lesen aller Konversationen aus dem Ordner inbox:

```
SELECT subject, snippet FROM unified_thread
WHERE folder='inbox'
```

Beispiel – Lesen aller Konversationen, die ein bestimmtes Schlüsselwort enthalten – dies kann mit der Hilfsfunktion `contains()` erreicht werden:

```
SELECT subject, snippet FROM unified_thread
WHERE contains('schlüsselwort')
```

Unified-Thread-Count-Tabelle

Die FQL-Tabelle `unified_thread_count` gibt Auskunft über die Anzahl der Konversationen in den Ordnern des Unified-Messaging-Postfachs.

Beispiel – Abfrage der Anzahl der Konversationen in allen Ordnern:

```
SELECT folder, unread_count, unseen_count, total_threads
FROM unified_thread_count WHERE 1
```

Feldname	Beschreibung	Typ	Indiziert
`folder`	Name des Ordners (`inbox`, `other` oder `spam`)	`string`	ja
`unread_count`	Anzahl der ungelesenen Konversationen im abgefragten Ordner	`int`	ja
`unseen_count`	Anzahl der noch nie angezeigten Konversationen im abgefragten Ordner	`int`	ja
`last_action_id`	die höchste `action_id` im abgefragten Ordner	`int`	ja
`total_threads`	Gesamtanzahl der Konversationen im abgefragten Ordner	`int`	ja

Tabelle 5.55 Felder der Unified-Thread-Count-Tabelle

Unified-Thread-Action-Tabelle

Die FQL-Tabelle `unified_thread_action` bildet die Aktionen ADD PEOPLE und LEAVE CONVERSATION im neuen Unified-Messaging-Postfach ab.

Beispiel – Lesen aller Aktionen einer bestimmten Konversation:

```
SELECT actor, type, users FROM unified_thread_action
WHERE thread_id='t_id.2O4485922964776'
```

Feldname	Beschreibung	Typ	Indiziert
action_id	Numerische Versions-ID der Konversation – diese ID spiegelt den Status einer Konversation zu einem bestimmten Zeitpunkt wider und wird laufend erhöht.	string	nein
actor	Benutzer, der die Aktion durchgeführt hat (Felder name, id, email)	Array	nein
thread_id	ID der Konversation, in der die Aktion durchgeführt wurde	string	ja
timestamp	UNIX-Zeitstempel der Aktion	UNIX-Zeitstempel	nein
type	Zeigt an, ob ein Benutzer zur Konversation hinzugefügt (1) wurde oder ob ein Benutzer die Konversation verlassen hat (2).	int	nein
users	Liste der Benutzer, für die die Aktion durchgeführt wurde (Felder name, id, email)	Array	nein

Tabelle 5.56 Felder der Unified-Thread-Action-Tabelle

Unified-Message-Tabelle

Die FQL-Tabelle unified_messages enthält alle Nachrichten des neuen Unified-Messaging-Postfachs.

Beispiel – Lesen aller Nachrichten einer bestimmten Konversation:

```
SELECT subject, body FROM unified_message
WHERE thread_id='t_id.204485922964776'
```

Feldname	Beschreibung	Typ	Indiziert
message_id	ID der Nachricht	string	ja
thread_id	ID der Konversation	string	ja
subject	Betreff der Nachricht	string	nein
body	Text der Nachricht	string	nein

Tabelle 5.57 Felder der Unified-Message-Tabelle

Feldname	Beschreibung	Typ	Indiziert
unread	Zeigt an, ob die Nachricht noch nicht gelesen wurde.	boolean	ja
action_id	Numerische Versions-ID der Konversation – diese ID spiegelt den Status einer Konversation zu einem bestimmten Zeitpunkt wider und wird laufend erhöht.	string	nein
timestamp	UNIX-Zeitstempel der Veröffentlichung der Nachricht	UNIX-Zeitstempel	ja
tags	Tags der Nachricht	Array	nein
sender	Absender der Nachricht (Felder name, id, email)	Array	nein
recipients	Liste der Empfänger der Nachricht (Felder name, id, email)	Array	nein
object_sender	Absender der Nachricht, wenn dieser eine Seite, Veranstaltung oder Gruppe ist (Felder object_address_type und id)	Array	nein
html_body	Text der Nachricht als HTML-String	string	nein
attachments	Liste der Anhänge der Nachricht	Array	nein
attachments_map	Verknüpfung der Anhänge zu Objekten mit weiterführenden Informationen (Felder id, type, mime_type, filename, file_size)	Array	nein
shares	Liste der geteilten Objekte (Links, Videos, Fotos ...)	Array	nein
shares_map	Verknüpfung der gesharten Objekte mit weiterführenden Informationen (share_id, type, href, title, summary, image)	Array	nein

Tabelle 5.57 Felder der Unified-Message-Tabelle (Forts.)

5.2.40 URL-Like-Tabelle

Die FQL-Tabelle url_like enthält alle externen, per Open Graph integrierten Seiten, auf denen ein Benutzer den GEFÄLLT-MIR-Button angeklickt hat. Zum Lesen der Tabelle ist die erweiterte Berechtigung user_likes bzw. friends_likes notwendig.

Beispiel – Lesen aller Gefällt mir eines bestimmten Benutzers:

```
SELECT url FROM url_like WHERE user_id = 532617296
```

Feldname	Beschreibung	Typ	Indiziert
user_id	ID des Benutzers, der auf Gefällt mir geklickt hat	int	ja
url	URL der externen Seite, auf der der Gefällt-mir-Button angeklickt wurde	string	nein

Tabelle 5.58 Felder der URL-Like-Tabelle

5.2.41 User-Tabelle

Die FQL-Tabelle user enthält Benutzerobjekte, wie in Abschnitt 3.2.1, »Das User-Objekt«, beschrieben. Zur Abfrage der einzelnen Felder sind jeweils jene Access Tokens und erweiterten Berechtigungen notwendig, die auch beim Zugriff über die Graph API notwendig sind.

Beispiel – Lesen der Benutzerinformationen des aktuellen Benutzers:

```
SELECT uid, name, username, pic FROM user WHERE uid=me()
```

Feldname	Beschreibung	Typ	Indiziert
uid	ID des Benutzers	int	ja
username	eindeutiger Benutzername	string	ja
first_name	Vorname	string	nein
middle_name	zweiter Vorname	string	nein
last_name	Nachname	string	nein
name	vollständiger Name	string	ja
pic_small	URL zum kleinen Profilbild des Benutzers (maximal 50 x 150 Pixel)	string	nein
pic_big	URL zum großen Profilbild des Benutzers (maximal 200 x 600 Pixel)	string	nein

Tabelle 5.59 Felder der User-Tabelle

Feldname	Beschreibung	Typ	Indiziert
pic_square	URL zum quadratischen Profilbild des Benutzers (50 x 50 Pixel)	string	nein
pic	URL zum mittelgroßen Profilbild des Benutzers (maximal 100 x 300 Pixel)	string	nein
affiliations	Netzwerke, denen der Benutzer angehört	Array	nein
profile_update_time	UNIX-Zeitstempel der letzten Aktualisierung des Profils. 0, wenn diese länger als drei Tage zurückliegt.	UNIX-Zeitstempel	nein
timezone	Zeitzone des Benutzers als Differenz zu UTC	int	nein
religion	Religionszugehörigkeit	string	nein
birthday	Geburtstag in lokalisiertem Format	string	nein
birthday_date	Geburtstag im Format MM/DD/YYY	string	nein
sex	Geschlecht	string	nein
hometown_location	Heimatstadt	Array	nein
meeting_sex	Geschlechter, die der Benutzer kennenlernen möchte	Array	nein
meeting_for	Gründe, aus denen der Benutzer andere Benutzer kennenlernen möchte	Array	nein
relationship_status	Beziehungsstatus	string	nein
significant_other_id	Benutzer-ID des Partners	int	nein
political	politische Einstellung	string	nein
current_location	aktueller Aufenthaltsort	Array	nein
activities	Aktivitäten	string	nein
interests	Interessen	string	nein

Tabelle 5.59 Felder der User-Tabelle (Forts.)

Feldname	Beschreibung	Typ	Indiziert
is_app_user	Zeigt an, ob der Benutzer die aktuelle Anwendung installiert hat.	boolean	nein
music	Lieblingsmusik	string	nein
tv	Lieblings-TV-Show	string	nein
movies	Lieblingsfilme	string	nein
books	Lieblingsbücher	string	nein
quotes	Lieblingszitate	string	nein
about_me	Kurz-Bio des Benutzers	string	nein
hs_info	Highschool (eingestellt)	Array	nein
education_history	Ausbildung (eingestellt)	Array	nein
work_history	Arbeitserfahrung (eingestellt)	Array	nein
notes_count	Anzahl der veröffentlichten Notizen	int	nein
wall_count	Anzahl der veröffentlichten Pinnwand-Einträge	int	nein
status	aktueller Status	string	nein
has_added_app	eingestellt, entspricht dem Feld is_app_user	boolean	nein
online_presence	Status im Facebook-Chat (active, idle, offline, error)	string	nein
locale	Lokalisierungseinstellung des Benutzers	string	nein
proxied_email	Enthält die Proxy-E-Mail-Adresse des Benutzers, wenn diese bei der Erteilung der Berechtigung email bereitgestellt wurde.	string	nein
profile_url	URL zum Profil des Benutzers	string	nein
email_hashes	Liste der bestätigten E-Mail-Adressen als Hashes (eingestellt) mit Facebook-Icon als Overlay	Array	nein

Tabelle 5.59 Felder der User-Tabelle (Forts.)

Feldname	Beschreibung	Typ	Indiziert
pic_small_with_logo	URL zum kleinen Profilbild des Benutzers (maximal 50 x 150 Pixel) mit Facebook-Icon als Overlay	string	nein
pic_big_with_logo	URL zum großen Profilbild des Benutzers (maximal 200 x 600 Pixel) mit Facebook-Icon als Overlay	string	nein
pic_square_with_logo	URL zum quadratischen Profilbild des Benutzers (50 x 50 Pixel) mit Facebook-Icon als Overlay	string	nein
pic_with_logo	URL zum mittelgroßen Profilbild des Benutzers (maximal 100 x 300 Pixel) mit Facebook-Icon als Overlay	string	nein
allowed_restrictions	kommaseparierte Liste der demographischen Zugriffseinschränkungen, die der Benutzer erfüllt – derzeit nur alcohol	string	nein
verified	Zeigt an, ob das Konto des Benutzers per Handy oder Kreditkarte verifiziert wurde.	boolean	nein
profile_blurb	Text unter dem Profilbild des Benutzers	string	nein
family	Liste der unmittelbaren Verwandten des Benutzers (Felder relationship, uid, name, birthday)	Array	nein
website	Website des Benutzers	string	nein
is_blocked	Zeigt an, ob der abgefragte Benutzer gegenüber dem aktuellen Benutzer geblockt ist.	boolean	nein
contact_email	primäre E-Mail-Adresse des Benutzers	string	nein
email	primäre E-Mail-Adresse oder Proxy-E-Mail-Adresse des Benutzers	string	nein
third_party_id	eine anonymisierte, eindeutige Benutzerkennung	string	ja
name_format	Angabe zur Formatierung des vollen Namens, z. B. »{first} {last}«	string	nein

Tabelle 5.59 Felder der User-Tabelle (Forts.)

Feldname	Beschreibung	Typ	Indiziert
video_upload_limits	maximale Länge und Dateigröße von Video-Uploads (Felder length und size)	Array	nein
hames	Lieblingsspiel	string	nein
is_minor	Zeigt an, ob der Benutzer minderjährig ist.	boolean	nein
work	Liste der Arbeitserfahrung (Felder employer, location, position, start_date, end_date)	Array	nein
education	Liste der Ausbildungen (Felder name, id, type, year, degree, concentration, classes)	Array	nein
sports	Liste der ausgeübten Sportaren (Felder id, name)	Array	nein
favorite_athletes	Liste der Lieblingsathleten (Felder id, name) – eingestellt	Array	nein
favorite_teams	Liste der Lieblingsteams (Felder id, name) – eingestellt	Array	nein
inspirational_people	Liste der inspirierenden Personen (Felder id, name)	Array	nein
languages	Liste der gesprochenen Sprachen (Felder id, name)	Array	nein
likes_count	Anzahl der Seiten, auf denen der Benutzer auf Gefällt mir geklickt hat	int	nein
mutual_friend_count	Anzahl der Freunde, die der abgefragte Benutzer mit dem aktuellen Benutzer gemeinsam hat	int	nein
can_post	Zeigt an, ob der aktuelle Benutzer auf der Pinnwand des abgefragten Benutzers posten kann.	boolean	nein

Tabelle 5.59 Felder der User-Tabelle (Forts.)

5.2.42 Videotabellen

Die FQL-Tabelle video bildet Videoobjekte, wie in Abschnitt 3.2.8, »Das Videoobjekt«, beschrieben, ab. Während öffentliche Videos mit einem beliebigen Access Token

abgefragt werden können, ist für private Videos ein Access Token mit der erweiterten Berechtigung `user_videos` bzw. `friends_videos` notwendig.

Beispiel – Lesen aller Videos des aktuellen Benutzers:

```
SELECT vid, owner, title, description, thumbnail_link, embed_html,
updated_time, created_time
FROM video WHERE owner=me()
```

Feldname	Beschreibung	Typ	Indiziert
`vid`	ID des Videos	`int`	ja
`owner`	Benutzer-ID des Besitzers des Videos	`int`	ja
`album_id`	ID des zugehörigen Albums	`int`	ja
`title`	Titel des Videos	`string`	nein
`description`	Beschreibung des Videos	`string`	nein
`link`	URL zum Video	`string`	nein
`thumbnail_link`	URL zum Thumbnail-Bild des Videos	`string`	nein
`embed_html`	HTML-Code zum Einbinden des Videos in externe Seiten	`string`	nein
`created_time`	UNIX-Zeitstempel der Erstellung des Videos	UNIX-Zeitstempel	nein
`updated_time`	UNIX-Zeitstempel des letzten Updates des Videos	UNIX-Zeitstempel	nein
`length`	Dauer des Videos in Sekunden	`int`	nein
`src`	URL zur Quelldatei des Videos in normaler Auflösung	`string`	nein
`src_hq`	URL zur Quelldatei des Videos in hoher Auflösung	`string`	nein

Tabelle 5.60 Felder der Videotabelle

Video-Tag-Tabelle

Die FQL-Tabelle `video_tag` enthält Videomarkierungen (*Tags*) von Benutzern und entspricht damit dem `tags`-Feld des `video`-Objekts, wie in Abschnitt 3.2.8, »Das Videoobjekt«, beschrieben. Abgesehen von den öffentlichen Videos einer Seite, ist zum Lesen von Videomarkierungen die Berechtigung `user_videos` bzw. `friends_videos` erforderlich.

Beispiel – Lesen aller Videomarkierungen des aktuellen Benutzers:

```
SELECT vid FROM video_tag WHERE subject = me()
```

Beispiel – Lesen aller auf einem bestimmten Video markierten Benutzer:

```
SELECT subject FROM video_tag WHERE vid=10150309485965864
```

Neben Videomarkierungen wird `video_tag` auch genutzt, um die Videos, die mit einer Gruppe oder einem Event verknüpft sind, auszulesen. Dazu muss als `subject` lediglich die ID der Gruppe oder des Events abgefragt werden.

Beispiel – Lesen aller Videos, die mit einer bestimmten Gruppe verknüpft sind:

```
SELECT vid FROM video_tag
WHERE subject = 269627173053176
```

Feldname	Beschreibung	Typ	Indiziert
`vid`	ID des Videos	`int`	ja
`subject`	ID des markierten Benutzers, der Veranstaltung oder Gruppe	`int`	ja
`created_time`	UNIX-Zeitstempel der Erstellung der Videomarkierung	UNIX-Zeitstempel	nein
`updated_time`	UNIX-Zeitstempel des letzten Updates der Videomarkierung	UNIX-Zeitstempel	nein

Tabelle 5.61 Felder der Video-Tag-Tabelle

Kapitel 6
Social Plugins

Mit den Social Plugins bietet Facebook ein leistungsstarkes Set an Widgets, mit denen Entwickler und Content-Anbieter auf einfachstem Weg Webseiten mit Facebook integrieren und soziale Funktionen in ihren Webanwendungen hinzufügen können.

Mit der Veröffentlichung von *Social Plugins*, insbesondere dem mittlerweile allgegenwärtigen LIKE-Button, hat Facebook im Jahr 2010 begonnen, den sozialen Graphen über die Grenzen von Facebook.com hinaus zu erweitern. Social Plugins sind *Widgets*, also kleine HTML-Komponenten, die es ermöglichen, Inhalte und Funktionalitäten von der Plattform des Widget-Anbieters (also Facebook) auf den Webseiten Dritter einzubinden. Widgets sind seit der »Web 2.0«-Ära gut bekannt und zählen seitdem zum Standardrepertoire der meisten Webanwendungen und -services. Die Social Plugins von Facebook gehen dabei allerdings einen Schritt weiter als die Widgets der meisten anderen Anbieter, da sie neben der Bereitstellung von Inhalten und Funktionalitäten auch die Integration mit der eigenen »Objektdatenbank«, also dem sozialen Graphen von Facebook, ermöglichen. Diese Integration mit dem sozialen Graphen beruht wesentlich auf dem *Open Graph Protocol*, das in Abschnitt 8.1, »Open Graph Protocol«, näher beschrieben wird.

Neben der Einsatzmöglichkeit auf externen Webseiten sind Social Plugins aber auch für Canvas-Anwendungen auf Facebook selbst sehr nützlich: Sie können etwa dazu dienen, eine Facebook-Anwendung rasch um Kommentarfunktionen, Sharing-Möglichkeiten oder gar einen Live-Chat zu erweitern. Technisch gesehen, macht es jedenfalls keinen Unterschied, ob Social Plugins auf der eigenen Website oder innerhalb des Canvas zum Einsatz kommen.

6.1 Laden von Social Plugins

Wie bereits in Abschnitt 4.5, »Laden von Social Plugins«, beschrieben wurde, können Social Plugins auf drei Arten eingebunden werden:

- Social Plugins werden im einfachsten Fall durch den Einbau eines *HTML-iFrames* in der eigenen Webseite integriert. Im iFrame wird das gewünschte Plugin mit den notwendigen Konfigurationseinstellungen geladen. Der Vorteil dieser Methode:

Der Betreiber der Webseite muss lediglich eine Zeile HTML-Code einfügen, tieferes technisches Wissen oder Modifikationen der eigenen Webseite sind nicht notwendig.

```
// Einbindung als iFrame
<iframe src="//www.facebook.com/plugins/like.php?
href=http%3A%2F%2Fdie.socialisten.at
&amp;send=false
&amp;layout=standard
&amp;width=450
&amp;show_faces=true
&amp;action=like
&amp;colorscheme=light"
scrolling="no" frameborder="0"
allowTransparency="true">
</iframe>
```

Listing 6.1 Laden des »Like«-Buttons als iFrame

- Die derzeit vermutlich am häufigsten genutzte Methode zur Integration von Social Plugins ist das Laden über XFBML-Tags. Dabei werden spezielle Tags zum Einfügen der Social Plugins verwendet, etwa das Tag `<fb:like>` zum Einbinden des Like-Buttons. Um die XFBML-Tags mit den eigentlichen Widget-Inhalten zu füllen, ist es notwendig, das Facebook JavaScript SDK zu laden und `FB.init()` mit der Option `xfbml=true` zu initialisieren. Dies bewirkt, dass das SDK automatisch alle XFBML-Tags im HTML-Dokument durch die jeweiligen Inhalte von Facebook ersetzt. Im Vergleich zur iFrame-Methode haben XFBML-Tags den Vorteil der dynamischen Größenanpassung (die Größe von iFrames ist hingegen statisch) und eines etwas größeren Funktionsumfangs der Plugins. Speziell beim Laden einer größeren Anzahl an Plugins auf derselben Seite hat die XFBML-Variante durch ihre asynchrone Ausführung auch einen kleinen Performance-Vorteil. Im Gegensatz zur iFrame-Methode muss der Betreiber der Webseite aber für die korrekte Einbindung des SDKs sorgen.

```
// Einbindung als XFBML-Tag
<fb:like href="http://die.socialisten.at"
send="false"
width="450"
show_faces="true">
</fb:like>
```

Listing 6.2 Laden des »Like«-Buttons als XFBML-Tag

- Da XFBML-Tags eine Facebook-spezifische Erweiterung der HTML-Syntax darstellen, verhindert ihre Nutzung, dass die eigene Webseite HTML-konform codiert werden kann. Abhilfe schafft die dritte Methode zur Einbindung von Social Plugins mittels HTML5-konformer Tags. Dabei werden meist herkömmliche `<div>`-Container genutzt, die durch Definition bestimmter Klassen als Social Plugins ausgewiesen werden. So werden LIKE-Buttons etwa mit `<div class="fb-like">` ausgezeichnet. `data-...`-Attribute dienen dabei der weiteren Konfiguration der Plugins. *Hinweis:* Während die Parameter in XFBML-Tags meist Unterstriche enthalten, werden diese in den HTML5-Varianten durch Bindestriche ersetzt! Auch bei dieser Methode sorgt erst das eingebundene JavaScript SDK dafür, dass die HTML5-Tags durch die entsprechenden Plugin-Inhalte ersetzt werden.

```
// Einbindung als HTML5-Tag
<div class="fb-like"
  data-href="http://die.socialisten.at"
  data-send="false"
  data-width="450"
  data-show-faces="true">
</div>
```

Listing 6.3 Laden des »Like«-Buttons als HTML5-Tag

Hinweis: Auch bei Nutzung von XFBML- und HTML5-Tags kommen letztlich iFrames zur Anzeige der Plugin-Inhalte zum Einsatz! Die iFrames werden dabei allerdings nicht fest im HTML-Dokument eingebaut, sondern vom SDK zur Laufzeit eingefügt!

Künftig empfiehlt es sich, Social Plugins immer mit der HTML5-Methode einzubinden. Diese Methode wird von Facebook selbst bevorzugt und bietet damit den größten Funktionsumfang, ist am performantesten und ermöglicht darüber hinaus die Erstellung von validem HTML5-Code. Nur in Ausnahmefällen, etwa wenn die Einbindung des JavaScript SDKs aus technischen Gründen unmöglich ist, sollte auf die iFrame-Methode zurückgegriffen werden.

In den Code-Beispielen der folgenden Abschnitte wird grundsätzlich die HTML5-Methode eingesetzt, bei den meisten Beispielen finden Sie zum Vergleich aber auch Code für die iFrame- und XFBML-Varianten.

6.2 Der »Like«-Button

Der LIKE-Button ist das am weitesten verbreitete Social Plugin. Er bietet Besuchern einer beliebigen Webseite die Möglichkeit, die Seite und ihre Inhalte mit nur einem einzigen Klick auf der eigenen Facebook-Pinnwand mit den eigenen Freunden zu teilen. Da der LIKE-Button damit ein mächtiges Werkzeug zur sozialen Distribution von

Content ist, wird er mittlerweile auf Millionen Websites genutzt – kaum ein Anbieter von News, Video, Musik oder E-Commerce verzichtet heute auf die Möglichkeiten des LIKE-Buttons.

Das folgende Code-Beispiel zeigt die Einbindung eines LIKE-Buttons mit der HTML5-Methode:

```
// Einbindung als HTML5-Tag
<div class="fb-like"
     data-href="http://www.facebook.com/"
     data-send="true"
     data-width="450"
     data-show-faces="true">
</div>
```

Listing 6.4 Laden des »Like«-Buttons als HTML5-Tag

Das HTML5-Tag gibt im Parameter `href` an, welche URL auf der Pinnwand des Benutzers geteilt werden soll. Die weiteren Parameter konfigurieren die im Plugin angezeigten Inhalte. Zur Laufzeit wird dieses HTML5-Tag vom JavaScript SDK durch einen iFrame ersetzt, der den eigentlichen Plugin-Inhalt von Facebook lädt. Der eigentliche LIKE-Button wird dabei optional ergänzt um eine zusätzliche SENDEN-Funktion, die Anzahl der Benutzer, die auf den Button geklickt haben, und einen Auszug jener Freunde des Benutzers, die ebenfalls schon mit diesem LIKE-Button interagiert haben.

Abbildung 6.1 Darstellung des »Like«-Buttons in seiner bekannten Form

Der »href«-Parameter in Social Plugins

Beim LIKE-Button und auch den meisten anderen Plugins kommt dem Parameter `href` besondere Bedeutung zu. Mit diesem Parameter wird bestimmt, auf welche URL sich das Social Plugin bezieht. Beim LIKE-Button legt `href` also fest, welche Webseite auf der Pinnwand der Benutzer, die den Button anklicken, geteilt werden soll. Die in `href` angegebene URL hat also eine identifizierende Funktion, da jede mit LIKE-Button ausgestattete Seite auch durch ein Objekt im sozialen Graphen von Facebook repräsentiert wird.

Es ist also von großer Wichtigkeit, `href` konsistent und sinnvoll zu setzen. Folgende Problemfelder sind dabei besonders zu beachten:

- HTTP vs. HTTPS – Facebook unterscheidet URLs nach dem jeweils verwendeten Protokoll. Die URLs *https://apps.mycompany.com* und *http://apps.mycompany.com* werden also von Facebook als unterschiedliche Objekte im sozialen Graphen abgebildet. Bei Webseiten, auf denen HTTPS eingesetzt werden soll, empfiehlt es sich daher, den `href`-Parameter immer mit der HTTPS-URL zu belegen, auch wenn ein Benutzer gerade mittels HTTP-Protokoll auf die Seite zugegriffen hat. Verwendet man stattdessen etwa beide Protokolle wechselweise in Verbindung mit dem Like-Button, hat dies zur Folge, dass Facebook zwei unterschiedliche Objekte im sozialen Graphen anlegt und folglich auch die Anzahl der Interaktionen mit dem Button auf diese beiden Objekte aufteilt.
- Für dynamische GET-Parameter in URLs gilt Ähnliches – auch die Verwendung von GET-Parametern führt dazu, dass Facebook unterschiedliche Objekte für die gleiche Webseite anlegt. So werden etwa die URLs *http://apps.mycompany.com/?view=all* und *http:/apps.mycompany.com/* als getrennte Objekte behandelt. Es empfielht sich also, den `href`-Parameter immer normalisiert, also ohne GET-Parameter, zu setzen.
- Permalinks vs. veränderliche URLs – in vielen Webseiten werden sprechende URLs zur besseren Lesbarkeit und Suchmaschinenoptimierung verwendet. Die URL zu einem Artikel könnte also etwa *http://apps.mycompany.com/articles/Erster-Artikel* lauten. Meistens werden diese URLs automatisch aus dem Titel des jeweiligen Inhalts abgeleitet. Doch was passiert, wenn sich der Titel etwa von »Erster Artikel« auf »Mein erster Artikel« ändert? Meistens wird dabei auch die URL auf *http://apps.mycompany.com/articles/Mein-Erster-Artikel* angepasst. Dies hätte wiederum zur Folge, dass der Like-Button, der diese URL mittels `href` referenziert, ein weiteres, getrenntes Objekt im sozialen Graphen von Facebook erzeugen würde. Eine mögliche Lösung wäre die Verwendung von unveränderlichen, ID-basierten URLs im `href`-Parameter. Würde die URL den Artikel anhand seiner numerischen ID (»1«) adressieren (*http://apps.mycompany.com/articles/1*), wäre eine spätere Änderung des Titels unproblematisch. Im Browser bzw. gegenüber dem Benutzer könnten dabei weiterhin die besser lesbaren, sprechenden URLs verwendet werden.

Hinweis: Die XFBML- & HTML5-Varianten der Social Plugins erlauben es grundsätzlich, den Parameter `href` nicht explizit anzugeben. In diesem Fall ermittelt das SDK automatisch die aktuelle URL im Browser, auf der das Plugin integriert wurde. Obwohl dies auf den ersten Blick praktisch erscheinen mag, rate ich Ihnen von dieser Vorgehensweise aufgrund der erwähnten Problemfelder jedoch ganz klar ab.

Der gleiche Like-Button kann auch in XFBML-Notation integriert werden – der wesentliche Unterschied ist dabei, dass das proprietäre Tag `<fb:like>` mit den ebenfalls proprietären Attributen `send`, `show_faces` etc. verwendet wird. Obwohl dieser

Code also aus Sicht des Benutzers zum gleichen Ergebnis führt, hat er doch den Nachteil, nicht valider HTML-Syntax zu entsprechen:

```
// Einbindung als XFBML-Tag
<fb:like href="https://www.facebook.com"
         send="true"
         width="450"
         show_faces="true">
</fb:like>
```

Listing 6.5 Laden des »Like«-Buttons als XFBML-Tag

Der Vollständigkeit halber sei hier auch der Code abgebildet, der notwendig ist, um den LIKE-Button als iFrame auf der eigenen Webseite zu integrieren:

```
// Einbindung als iFrame
<iframe src="//www.facebook.com/plugins/like.php?
        href=https%3A%2F%2Fwww.facebook.com
             &amp;send=false
             &amp;layout=standard
             &amp;width=450
             &amp;show_faces=true
             &amp;action=like
             &amp;colorscheme=light
             &amp;font&amp;height=80
             &amp;appId=144485858997536"
scrolling="no" frameborder="0"
style="border:none;overflow:hidden;width:450px;height:80px;"
allowTransparency="true"></iframe>
```

Listing 6.6 Laden des »Like«-Buttons als iFrame

Wichtig: Die iFrame-Variante des LIKE-Buttons unterstützt nicht die Möglichkeit, einen SENDEN-Button mit dem Parameter `send` im Plugin anzuzeigen!

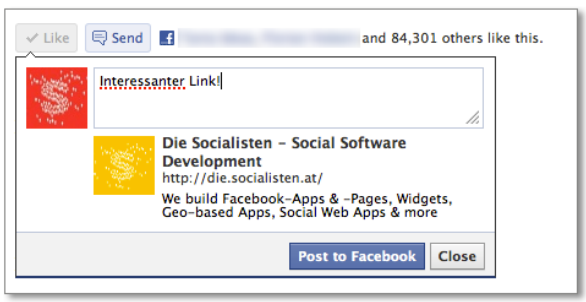


Abbildung 6.2 Hinzufügen eines persönlichen Kommentars zum geteilten Link

Klickt ein Benutzer auf den LIKE-Button, passiert Folgendes:

- Ist der Benutzer in der aktuellen Browser-Session nicht bei Facebook angemeldet, öffnet sich ein entsprechender Login-Dialog – der LIKE-Button funktioniert nur für angemeldete Benutzer!
- Für angemeldete Benutzer öffnet sich ein Fenster, das die Eingabe eines zusätzlichen, persönlichen Kommentars ermöglicht. *Hinweis:* Dieses Kommentarfeld wird bei LIKE-Buttons, die mittels iFrame eingebunden wurden, nicht unterstützt. Für HTML5- & XFBML-Tags ist das Kommentarfeld verpflichtend und kann vom Entwickler auch nicht deaktiviert werden.
- Unabhängig davon, ob der Benutzer einen persönlichen Kommentar eingibt oder nicht, veröffentlicht Facebook anschließend den geteilten Inhalt auf der Pinnwand des Benutzers.
- Dabei werden Linktext, Thumbnail und Beschreibungstext automatisch von Facebook aus der in `href` angegebenen Webseite extrahiert. Hierbei werden entweder die üblichen HTML-Tags `<title>`, `<meta>` etc. verwendet oder – wenn vorhanden – die Facebook-spezifischen Open Graph Tags. Diesen in Abschnitt 8.1, »Open Graph Protocol«, detaillierter beschriebenen Meta-Tags kommt dabei eine große Bedeutung zu. Sie geben dem Content-Anbieter umfangreiche Möglichkeiten, festzulegen, wie geteilte Inhalte auf Facebook dargestellt werden sollen.

Abbildung 6.3 So sieht das Wall-Posting aus, das durch Anklicken des »Like«-Buttons auf der Pinnwand des Benutzers veröffentlicht wird.

Was bedeutet eigentlich die Zahl am »Like«-Button?

Meist wird davon ausgegangen, dass die Zahl, die Facebook am LIKE-Button zu einer bestimmten URL darstellt, bedeutet, dass eine entsprechende Anzahl von Benutzern den LIKE-Button bereits angeklickt hat. Das ist allerdings nicht korrekt! Vielmehr stellt die angezeigte Zahl die Summe an Interaktionen dar, die in Bezug zur referenzierten URL erfolgt sind. Die Zahl enthält somit:

- Anzahl der Benutzer, die den LIKE-Button angeklickt haben (mehrmaliges LIKE & UN-LIKE wird dabei selbstverständlich nicht doppelt gewertet)

- Anzahl, wie oft die URL von Benutzern auf Facebook geteilt wurde (etwa über manuelles Wall-Posting oder die SHARE-Funktion, also unabhängig vom LIKE-Button)
- Anzahl der Kommentare und »Likes«, die auf Wall-Postings, die die URL geteilt haben, veröffentlicht wurden
- Anzahl der privaten Nachrichten, die die URL enthalten haben

Wichtig ist, festzuhalten, dass die dargestellte Zahl am LIKE-Button relativ einfach durch einzelne Benutzer zu manipulieren ist, etwa indem sie die URL öfter teilen und zahlreiche Kommentare dazu veröffentlichen. Dies bedeutet leider auch, dass der LIKE-Button bzw. die darauf dargestellte Zahl kein besonders gut geeigneter Mechanismus zur Durchführung von Votings und Wettbewerben ist.

Der LIKE-Button unterstützt zahlreiche Layout-Formate, die mit folgenden Parametern konfiguriert werden können:

- `layout` – Während das Default-Layout (`standard`) mehr Platz für die Anzeige der Namen von Personen, die ebenfalls mit dem Button interagiert haben, beansprucht, erlauben die Formate `button_count` bzw. `box_count` ein platzsparendes horizontales bzw. vertikales Format, das sich auf die Anzeige der Anzahl der Interaktionen beschränkt. Die minimale Breite in diesem Layout beträgt 225 Pixel bzw. zusätzliche 40 Pixel, wenn als Aktion RECOMMEND gewählt ist, und zusätzliche 60 Pixel, wenn der optionale SEND-Button angezeigt werden soll. Die minimale Höhe beträgt 35 Pixel. *Hinweis:* Die hier angegebenen Minimalgrößen des LIKE-Buttons sind grundsätzlich mit großer Vorsicht zu verwenden – die tatsächliche Größe des Buttons hängt vor allem auch von der aktuell verwendeten Sprache ab, so ist etwa das deutsche GEFÄLLT MIR wesentlich breiter als das ursprüngliche »Like«. Entwickler und Screen-Designer müssen dies beim Layout der eigenen Seiten und Anwendungen beachten. Die tatsächlich angezeigte Sprache leitet sich dabei von der Lokalisierung ab, mit der das SDK ursprünglich geladen wurde. Nur bei der Einbindung als iFrame kann mit dem GET-Parameter `locale=de_DE` die Lokalisierung manuell ausgewählt werden.
- Nur im `standard`-Layout kann darüber hinaus mit dem Parameter `show_faces` festgelegt werden, ob die Profilbilder von Personen, die ebenfalls mit dem Button interagiert haben, angezeigt werden sollen. Dies kann die Klickraten eines LIKE-Buttons entscheidend verbessern, vor allem wenn Freunde des Benutzers bereits auf den Button geklickt haben. Die minimale Höhe beträgt 80 Pixel.
- Der optionale SEND-Button kann mit dem Parameter `send` aktiviert werden. Analog zum SEND-Dialog des JavaScript SDKs erlaubt er das gezielte Versenden an mehrere Freunde, Facebook-Gruppen oder E-Mail-Adressen.

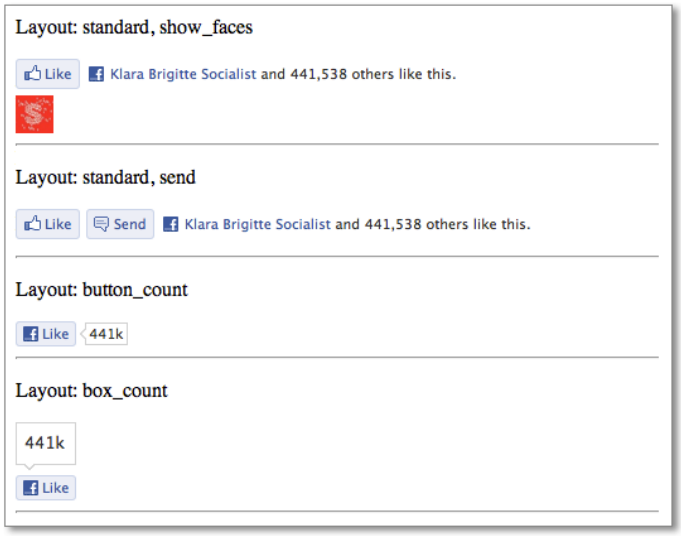


Abbildung 6.4 Unterschiedliche Layout-Formate des »Like«-Buttons

Die folgende Tabelle zeigt eine Übersicht über alle verfügbaren Einstellungsmöglichkeiten des LIKE-Buttons:

Name	Typ	Beschreibung
`href`	`string`	URL der Webseite, auf die sich der LIKE-Button beziehen soll bzw. die auf der Pinnwand des Benutzers geteilt werden soll. Besonders sollten dabei die erwähnten Problemfelder beachtet werden!
`send`	`boolean`	Gibt an, ob der optionale SEND-Button angezeigt werden soll (nicht unterstützt bei Einbindung als iFrame).
`layout`	`string`	`standard` (Default), `box_count` oder `button_count`
`show_faces`	`boolean`	Gibt an, ob unter dem LIKE-Button die Profilbilder anderer Benutzer, die bereits mit dem Button interagiert haben, angezeigt werden sollen (nur im Layout `standard` unterstützt).
`width`	`int`	Breite des Plugins
`action`	`string`	Gibt an, welches Verb im LIKE-Button angezeigt werden soll. Derzeit nur wählbar zwischen `like` und `recommend`.

Tabelle 6.1 Parameter des »Like«-Buttons

Name	Typ	Beschreibung
font	string	Schriftart, in der der Like-Button angezeigt werden soll. Wählbar sind derzeit arial, lucida grande, segoe ui, tahoma, trebuchet ms, verdana
colorscheme	string	Das Farbschema, in dem der Like-Button angezeigt wird, kann entweder mit light (Default-Einstellung) oder dark eingestellt werden.
ref	string	Ein maximal 50 Zeichen langes Feld mit Tracking-Daten. Wenn ref gesetzt ist, ergänzt Facebook jeden Klick auf die in href angegebene URL mit folgenden GET-Parametern: fb_ref enthält den in ref angegebenen Wert. fb_source gibt Auskunft darüber, wo der Link auf Facebook angeklickt wurde (home, profile, search, other).

Tabelle 6.1 Parameter des »Like«-Buttons (Forts.)

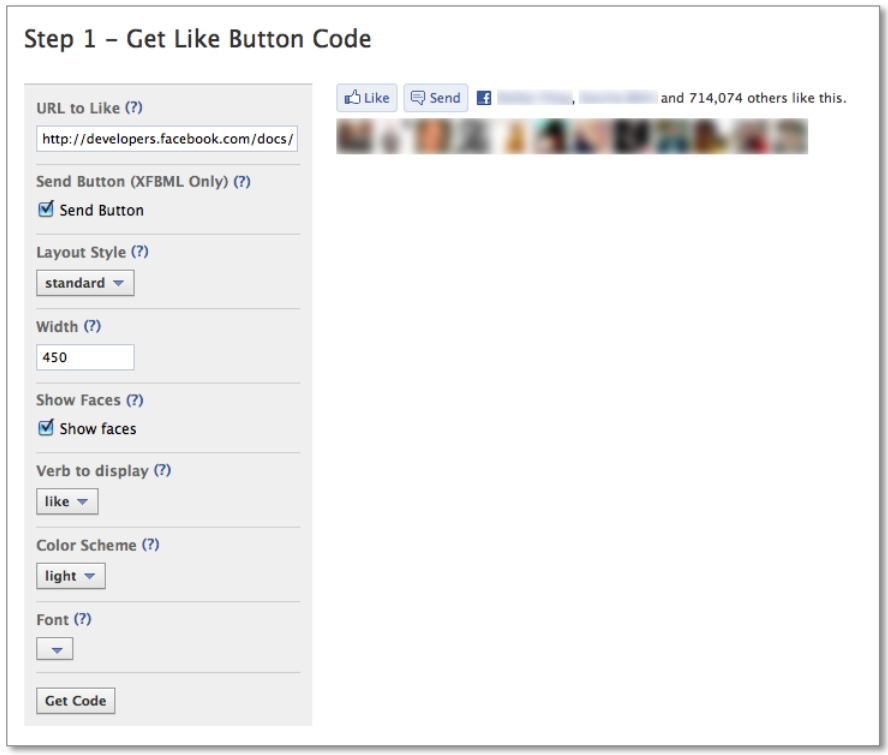


Abbildung 6.5 Der Plugin-Konfigurator des »Like«-Buttons

Hinweis: Die notwendigen Konfigurationsparameter können für die iFrame-, XFBML- und HTML5-Methode einfach mit dem von Facebook bereitgestellten Konfigurator erzeugt werden. Der Konfigurator steht für alle Social Plugins zur Verfügung und ermöglicht ein einfaches Experimentieren mit den verschiedenen Optionen.

Wichtiger Link:

- *https://developers.facebook.com/docs/reference/plugins/like/* – Plugin-Konfigurator und Dokumentation zum LIKE-Button

Event-Handler für den »Like«-Button

Entwickler haben mit dem JavaScript SDK die Möglichkeit, durch Abonnement der Events `edge.create` und `edge.remove` festzustellen, wenn ein Benutzer auf einen LIKE-Button geklickt bzw. durch wiederholtes Klicken das »Like« wieder entfernt hat.

```
FB.Event.subscribe('edge.create', function(response) {
   console.debug('edge.create');
   console.debug(response);
});
FB.Event.subscribe('edge.remove', function(response) {
   console.debug('edge.remove');
   console.debug(response);
});
```

Der entsprechende Event-Handler erhält im Parameter `response` dabei den Wert des Parameters `href` – also die URL, auf die sich der LIKE-Button bezieht. Mit dieser Information ist es möglich, etwa mittels AJAX einen serverseitigen Prozess anzustoßen, der die Anzahl der »Likes« der angeklickten URL mittels Graph API abfragt und etwa in der Datenbank der eigenen Anwendung aktualisiert. Dies ist praktisch, wenn Objekte in der Datenbank (etwa die Fotos in einem Fotowettbewerb) z. B. nach Anzahl der »Likes« sortiert ausgegeben werden sollen.

Achtung: Die auf diese Art ermittelte Anzahl der »Likes« kann dennoch von der am LIKE-Button angezeigten Zahl abweichen! Einerseits kann es sein, dass Benutzer ihr »Like« über ihr Profil auf Facebook.com zurückziehen, was natürlich kein Event auf-seiten der Anwendung auslöst. Zum anderen fließen wie erwähnt auch die Anzahl der »Shares«, Kommentare etc. in die am LIKE-Button angezeigte Zahl.

6.3 Der »Send«-Button

Mit dem SEND-Button steht die Möglichkeit des Versands privater Nachrichten an mehrere Benutzer, Facebook-Gruppen und E-Mail-Adressen, wie vom LIKE-Button mit dem Parameter `send` bekannt, auch als separates Plugin zur Verfügung. Der

SEND-Button entspricht in der Funktionalität dem in Abschnitt 4.4.6, »›Send‹-Dialog«, beschriebenen SDK-Dialog.

Das folgende Code-Beispiel demonstriert die Einbindung des SEND-Buttons als HTML5- bzw. XFBML-kompatiblen Code:

```
// Einbindung als HTML5-Tag
<div class="fb-send"
     data-href="https:// developers.facebook.com">
</div>

// Einbindung als XFBML-Tag
<fb:send href="https://www.facebook.com/developers"></fb:send>
```

Listing 6.7 Einbindung des »Send«-Buttons als HTML5- und XFBML-Tag

Hinweis: Der SEND-Button steht nicht in einer iFrame-Version zur Verfügung, sondern muss mittels XFBML oder HTML5 eingebunden werden!

Das JavaScript SDK ersetzt dieses XFBML- bzw. HTML5-Tag nach dem Laden der Seite durch einen iFrame, in dem der SEND-Button von Facebook geladen und folgendermaßen angezeigt wird:

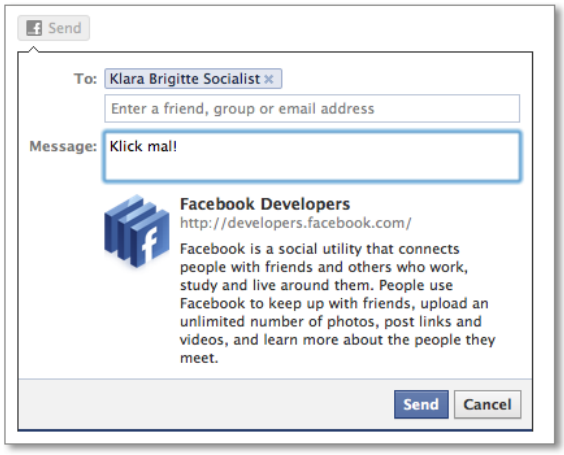


Abbildung 6.6 »Send«-Button zum Versenden privater Nachrichten

Der Parameter `href` stellt auch beim SEND-Button den Bezug zu jener Webseite her, die als Anhang der Nachricht versendet werden soll. Obwohl, anders als beim LIKE-Button, der Wert von `href` hier weniger sensibel bezüglich unterschiedlicher Schreibweisen, GET-Parameter etc. ist, lohnt es sich dennoch, auch beim SEND-Button eine

normalisierte URL zu verwenden: Die Anzahl der Nachrichten, die mit einer bestimmten URL versendet wurden, wird nämlich ebenfalls zu jener Zahl addiert, die am LIKE-Button dargestellt wird.

Neben `href` kann der SEND-Button mit folgenden Parametern konfiguriert werden:

Name	Typ	Beschreibung
`href`	`string`	URL der Webseite, die als Anhang der Nachricht versendet werden soll. Die Wahl von `href` sollte analog zum LIKE-Button erfolgen.
`font`	`string`	Schriftart, in der der SEND-Button angezeigt werden soll. Wählbar sind derzeit `arial`, `lucida grande`, `segoe ui`, `tahoma`, `trebuchet ms`, `verdana`.
`colorscheme`	`string`	Das Farbschema, in dem der SEND-Button angezeigt wird, kann entweder mit `light` (Default-Einstellung) oder `dark` eingestellt werden.
`ref`	`string`	Ein maximal 50 Zeichen langes Feld mit Tracking-Daten. Wenn `ref` gesetzt ist, ergänzt Facebook jeden Klick auf die in `href` angegebene URL mit folgenden GET-Parametern: `fb_ref` enthält den in `ref` angegebenen Wert. `fb_source` gibt Auskunft darüber, wo der Link in der Nachricht angeklickt wurde (`message`, `group`, `email`).

Tabelle 6.2 Parameter des »Send«-Buttons

Event-Handler für den »Send«-Button

Entwickler haben mit dem JavaScript SDK die Möglichkeit, durch Abonnement des Events `message.send` festzustellen, wenn ein Benutzer eine Nachricht mittels SEND-Button versendet hat.

```
FB.Event.subscribe('message.send', function(response) {
   console.debug('message.send');
   console.debug(response);
});
```

Der entsprechende Event-Handler erhält im Parameter `response` dabei den Wert des Parameters `href` – also die URL, auf die sich der SEND-Button bezieht.

6.4 Der »Subscribe«-Button

Der SEND-Button ermöglicht es, die öffentlichen Pinnwand-Einträge eines Facebook-Benutzers auf einfache Art zu abonnieren. Anders als eine Freundschaftsbeziehung zwischen zwei Facebook-Benutzern muss ein Abonnement von der abonnierten Person nicht bestätigt werden, es handelt sich also um eine unidirektionale Verknüpfung zweiter Benutzer. Der SUBSCRIBE-Button ist etwa zur Integration auf der eigenen Homepage oder dem eigenen Blog gedacht. Ein wiederholtes Klicken auf den SUBSCRIBE-Button entfernt ein zuvor aktiviertes Abonnement wieder.

Beim Laden des SUBSCRIBE-Buttons muss der `href`-Parameter auf die URL eines Facebook-Benutzerprofils verweisen (der SUBSCRIBE-Button unterstützt also keine Seiten- oder Anwendungsprofile). Der folgende Code zeigt die Integration des SUBSCRIBE-Buttons als HTML5-Tag:

```
// Einbindung als HTLM5-Tag
<div class="fb-subscribe"
     data-href="https://www.facebook.com/zuck"
     data-show-faces="true"
     data-width="450">
</div>
```

Listing 6.8 Einbindung des »Subscribe«-Buttons als HTML5-Tag

Die Einbindung als XFBML-Tag (`<fb:subscribe>`) oder iFrame (*http://www.facebook.com/plugins/subscribe.php?...)* erfolgt analog zum LIKE-Button und wird an dieser Stelle daher nicht ausführlich wiederholt.

Die Möglichkeiten zur Konfiguration des Layouts des SUBSCRIBE-Buttons entsprechen weitestgehend denen des LIKE-Buttons:

- `layout` – Während das Default-Layout (`standard`) mehr Platz für die Anzeige der Namen von Personen, die den verknüpften Benutzer ebenfalls abonniert haben, beansprucht, erlauben die Formate `button_count` bzw. `box_count` ein platzsparendes horizontales bzw. vertikales Format, das sich auf die Anzeige der Anzahl der Abonnements beschränkt. Bezüglich der minimalen horizontalen und vertikalen Maße gelten die gleichen Angaben wie beim LIKE-Button.
- Nur im `standard`-Layout kann darüber hinaus mit dem Parameter `show_faces` festgelegt werden, ob die Profilbilder von Personen, die den verknüpften Benutzer ebenfalls abonniert haben, angezeigt werden sollen.

Die unterschiedlichen Layout-Optionen des SUBSCRIBE-Buttons werden folgendermaßen dargestellt:

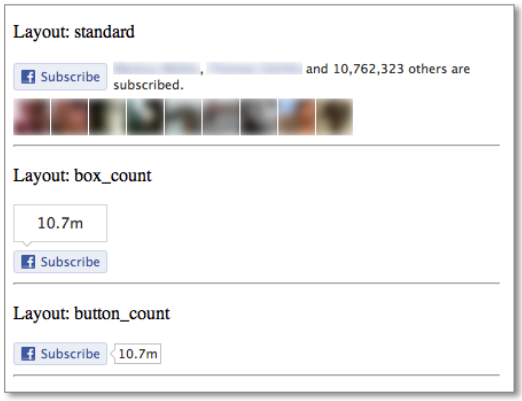


Abbildung 6.7 »Subscribe«-Button in unterschiedlichen Darstellungsformen

Alle weiteren Konfigurationsoptionen finden Sie in folgender Tabelle:

Name	Typ	Beschreibung
`href`	`string`	URL des Facebook-Benutzerprofils, das abonniert werden soll
`layout`	`string`	`standard` (Default), `box_count` oder `button_count`
`show_faces`	`boolean`	Gibt an, ob unter dem SUBSCRIBE-Button die Profilbilder anderer Benutzer, die den verknüpften Benutzer bereits abonniert haben, angezeigt werden sollen (nur im Layout `standard` unterstützt).
`width`	`int`	Breite des Plugins
`font`	`string`	Schriftart, in der der LIKE-Button angezeigt werden soll. Wählbar sind derzeit `arial`, `lucida grande`, `segoe ui`, `tahoma`, `trebuchet ms`, `verdana`.
`colorscheme`	`string`	Das Farbschema, in dem der LIKE-Button angezeigt wird, kann entweder mit `light` (Default-Einstellung) oder `dark` eingestellt werden.

Tabelle 6.3 Parameter des »Subscribe«-Buttons

6.5 Die »Like«-Box

Die LIKE-Box ermöglicht es Betreibern von Facebook-Seiten, ihr Seitenprofil auf attraktive Art in externen Webseiten zu integrieren, um so auch dort den Aufbau neuer Fans zu fördern. Das Plugin ermöglicht dabei die optionale Anzeige verschie-

dener Komponenten, etwa die neuesten Postings von der Pinnwand der Seite oder die Profilbilder einiger Fans der Seite.

Die Einbindung der LIKE-Box erfolgt über folgendes HTML5-Tag:

```
// Einbindung als HTML5-Tag
<div class="fb-like-box"
     data-href="http://www.facebook.com/diesocialisten"
     data-width="292"
     data-show-faces="true"
     data-stream="true"
     data-header="true">
</div>
```

Listing 6.9 Einbindung der »Like«-Box als HTML5-Tag

Der Parameter `href` muss dabei auf die URL einer Facebook-Seite oder -Anwendung verweisen. Beim Laden der Seite wird das HTML5-Tag vom JavaScript SDK durch einen iFrame ersetzt, der die eigentlichen Inhalte der LIKE-Box enthält.

Die LIKE-Box kann analog als XFBML-Tag oder iFrame eingebunden werden:

```
// Einbindung als XFBML-Tag
<fb:like-box href="http://www.facebook.com/diesocialisten"
             width="292" show_faces="true" stream="true"
             header="true">
</fb:like-box>

// Einbindung als iFrame
<iframe src="//www.facebook.com/plugins/likebox.php
   ?href=http%3A%2F%2Fwww.facebook.com%2Fdiesocialisten
   &amp;width=292
   &amp;height=590
   &amp;colorscheme=light
   &amp;show_faces=true
   &amp;border_color
   &amp;stream=true
   &amp;header=true
   &amp;appId=144485858997536"
scrolling="no" frameborder="0"
style="border:none; overflow:hidden; width:292px; height:590px;"
allowTransparency="true">
</iframe>
```

Listing 6.10 Einbindung der »Like«-Box als XFBML-Tag und iFrame

Die LIKE-Box unterstützt folgende Layout-Konfigurationen:

- Mit dem Parameter `show_faces` wird festgelegt, ob am unteren Rand der LIKE-Box Profilbilder von Fans der Seite angezeigt werden sollen.
- Der Parameter `stream` legt fest, ob die LIKE-Box die neuesten Pinnwand-Einträge der Seite enthalten soll.
- Der Parameter `header` legt fest, ob die LIKE-Box mit einer Kopfzeile im Facebook-Stil angezeigt werden soll.
- Über die Parameter `width` und `height` kann die genaue Größe der LIKE-Box festgelegt werden. Die Default-Breite beträgt 300 Pixel, die Default-Höhe hängt davon ab, ob Stream, Profilbilder und Header angezeigt werden (556 Pixel) oder nicht (63 Pixel).

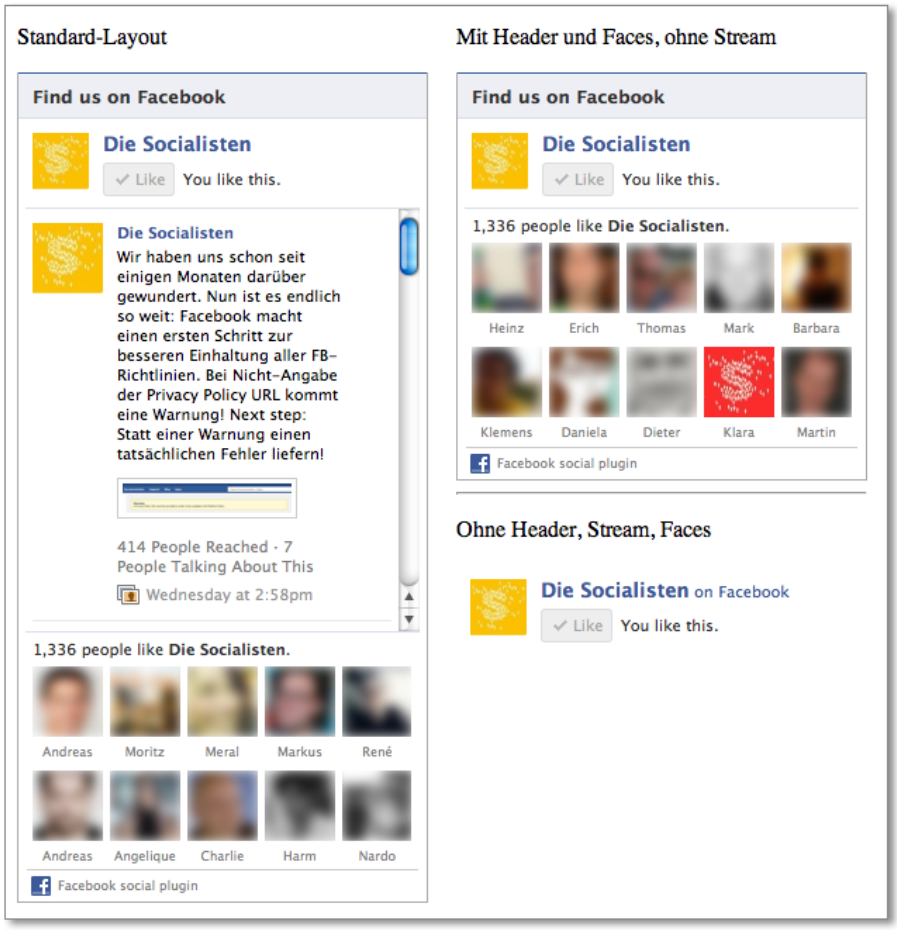


Abbildung 6.8 Die »Like«-Box in unterschiedlichen Layout-Konfigurationen

Alle weiteren Konfigurationsoptionen der LIKE-Box finden Sie in folgender Tabelle:

Name	Typ	Beschreibung
`href`	`string`	URL des Facebook-Seitenprofils, für das die LIKE-Box angezeigt werden soll
`show_faces`	`boolean`	Gibt an, dass am unteren Rand der LIKE-Box die Profilbilder von Fans der Seite angezeigt werden sollen. Die Anzahl der angezeigten Profilbilder wird dynamisch an die Höhe des Plugins angepasst.
`width`	`int`	Breite des Plugins
`height`	`int`	Höhe des Plugins
`font`	`string`	Schriftart, in der der LIKE-Button angezeigt werden soll. Wählbar sind derzeit `arial`, `lucida grande`, `segoe ui`, `tahoma`, `trebuchet ms`, `verdana`.
`colorscheme`	`string`	Das Farbschema, in dem die LIKE-Box angezeigt wird, kann entweder mit `light` (Default-Einstellung) oder `dark` eingestellt werden.
`stream`	`boolean`	Legt fest, ob die neuesten Pinnwand-Einträge der Seite angezeigt werden sollen.
`header`	`boolean`	Legt fest, ob die Kopfzeile der LIKE-Box angezeigt werden soll.
`border_color`	`string`	Farbe des Rahmens in hexadezimaler Notation (z. B. `#FF0000`). Wird nur berücksichtigt, wenn entweder `stream` oder `header` auf `true` gesetzt sind.
`force_wall`	`boolean`	Legt fest, ob bei Seiten die auch einen Ort darstellen, Pinnwand-Einträge anstelle der Checkins angezeigt werden sollen (Default: `false`).

Tabelle 6.4 Parameter der »Like«-Box

Event-Handler für die »Like«-Box

Die bereits beim LIKE-Button verwendeten Events `edge.create` und `edge.remove` funktionieren analog auch bei der LIKE-Box und werden ausgelöst, wenn der in der Box integrierte LIKE-Button angeklickt wird.

```
FB.Event.subscribe('edge.create', function(response) {
   console.debug('edge.create');
   console.debug(response);
});
FB.Event.subscribe('edge.remove', function(response) {
   console.debug('edge.remove');
   console.debug(response);
});
```

Der entsprechende Event-Handler erhält im Parameter `response` dabei den Wert des Parameters `href` – also die URL der Facebook-Seite, auf die die LIKE-Box referenziert.

6.6 Der Activity Feed

Das Social Plugin *Activity Feed* realisiert ein Widget, das verschiedene Interaktionen von Benutzern mit dem Content einer Website aggregiert darstellt. Dabei wird etwa angezeigt, wenn Benutzer einen LIKE-Button angeklickt haben, einen Link geteilt haben oder einen Kommentar mittels Kommentar-Plugin veröffentlicht haben. Der Activity Feed ist ein mächtiges Werkzeug zur Personalisierung einer Website, wird doch sein Inhalt automatisch an den Benutzer angepasst, der gerade in der aktuellen Browser-Session bei Facebook angemeldet ist. Dies bedeutet, dass Interaktionen, die von Freunden des Benutzers durchgeführt wurden, priorisiert angezeigt werden. Dies funktioniert auch, noch bevor der Benutzer die eigene Facebook-Anwendung autorisiert oder sich auf der eigenen Website angemeldet hat! Ist der betrachtende Benutzer hingegen nicht bei Facebook angemeldet, werden Interaktionen von beliebigen Benutzern angezeigt.

Das HTML5-Tag zur Einbindung des Activity Feeds sieht folgendermaßen aus:

```
// Einbindung als HTML5-Tag
<div class="fb-activity"
     data-site="die.socialisten.at"
     data-width="300"
     data-height="380"
     data-header="true"
     data-recommendations="false">
</div>
```

Listing 6.11 Einbindung des Activity Feeds als HTML5-Tag

Besondere Bedeutung kommt dem Parameter `site` zu, der angibt, von welcher Domain die im Plugin angezeigten Interaktionen aggregiert werden sollen.

Hinweis: Da der Activity Feed normalerweise alle Interaktionen einer Domain anzeigt, ist es problematisch, mehrere separate Facebook-Anwendungen auf einer Domain in verschiedenen Unterverzeichnissen zu betreiben (etwa *http://apps.mycompany.com/app1*, *http://apps.mycompany.com/app2* etc.). Ein derartiges Setup erfordert eine Einschränkung des Activity Feeds mit dem Parameter `filter`, um zu verhindern, dass Aktionen mehrerer Anwendungen vermischt werden.

Nachdem das SDK das HTML5-Tag des Activity Feeds durch den eigentlichen iFrame ersetzt hat, wird das Widget folgendermaßen dargestellt:

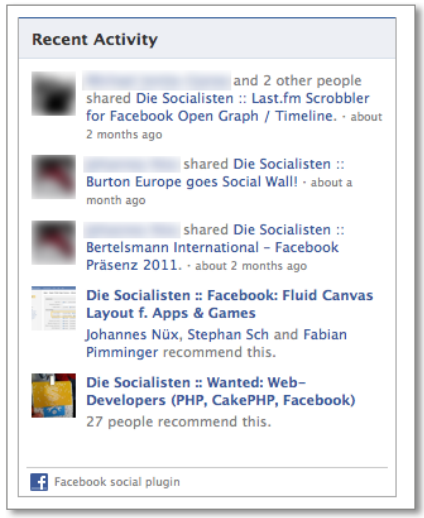


Abbildung 6.9 Der Activity Feed aggregiert Interaktionen einer Domain.

Der Activity Feed kann analog als XFBML-Tag (`<fb:activity>`) oder iFrame (*http://www.facebook.com/plugins/activity.php*?...) eingebunden werden:

```
// Einbindung als XFBML-Tag
<fb:activity site="die.socialisten.at"
             width="300"
             height="380"
             header="true" r
             ecommendations="false">
</fb:activity>

// Einbindung als iFrame
<iframe src="//www.facebook.com/plugins/activity.php
   ?site=die.socialisten.at
```

```
    &amp;width=300
    &amp;height=380
    &amp;header=true
    &amp;colorscheme=light
    &amp;linktarget=_blank
    &amp;border_color
    &amp;font
    &amp;recommendations=false
    &amp;appId=144485858997536"
    scrolling="no" frameborder="0"
    style="border:none; overflow:hidden; width:300px; height:380px;"
    allowTransparency="true">
</iframe>
```

Listing 6.12 Einbindung des Activity Feeds als XFBML-Tag und iFrame

Die folgende Tabelle zeigt eine Übersicht über alle verfügbaren Einstellungsmöglichkeiten des Activity Feeds:

Name	Typ	Beschreibung
`site`	`string`	Kommaseparierte Liste von Domains, von denen Interaktionen (Shares, Likes, Comments ...) aggregiert werden sollen. In der XFBML-/HTML5-Variante per Default die Domain der aktuellen Webseite, die das Plugin integriert.
`width`	`int`	Breite des Plugins (Default: 300 Pixel)
`height`	`int`	Höhe des Plugins (Default: 300 Pixel)
`font`	`string`	Schriftart, in der der Activity Feed angezeigt werden soll. Wählbar sind derzeit `arial`, `lucida grande`, `segoe ui`, `tahoma`, `trebuchet ms`, `verdana`.
`colorscheme`	`string`	Das Farbschema, in dem der Activity Feed angezeigt wird, kann entweder auf `light` (Default-Einstellung) oder `dark` eingestellt werden.
`header`	`boolean`	Legt fest, ob die Kopfzeile des Activity Feeds angezeigt werden soll.
`border_color`	`string`	Farbe des Rahmens in hexadezimaler Notation (z. B. `#FF0000`)

Tabelle 6.5 Parameter des Activity Feeds

Name	Typ	Beschreibung
recommendation	boolean	Legt fest, ob auch Empfehlungen im Feed angezeigt werden sollen.
filter	string	Mit diesem Parameter ist es möglich, die angezeigten Interaktionen auf URL-Ebene zu filtern. Wird etwa in site die Domain apps.mycompany.com und in filter der Wert /app1 angegeben, zeigt der Activity Feed nur Interaktionen auf URLs des Schemas apps.mycompany.com/app1/* an. Dies ist äußerst praktisch, wenn mehrere logisch getrennte Applikationen auf einer Domain in Unterverzeichnissen betrieben werden.
linktarget	string	Klickt ein Benutzer auf einen Eintrag im Activity Feed, wird per Default ein neues Browser-Fenster geöffnet (_blank). Mit den Werten _top und _parent kann dies zumindest für Inhalte, die nicht auf Facebook.com liegen, verhindert werden.
max_age	int	Erlaubt die Einschränkung der Einträge, die im Activity Feed angezeigt werden, nach ihrem maximalen Alter. Die Angabe erfolgt in Tagen.
ref	string	Ein maximal 50 Zeichen langes Feld mit Tracking-Daten. Wenn ref gesetzt ist, ergänzt Facebook jeden Klick auf einen Eintrag im Activity Feed mit dem GET-Parameter fb_ref, der den Wert von ref enthält.

Tabelle 6.5 Parameter des Activity Feeds (Forts.)

6.7 Die Recommendations-Box

Ähnlich dem Plugin Activity Feed erlaubt die Recommendations-Box eine sofortige Personalisierung von Inhalten für den in der Browser-Session an Facebook angemeldeten Benutzer. Anders als der Activity Feed werden hierbei aber nur Empfehlungen von Inhalten (*Recommendations*) berücksichtigt.

Die Domain, von der Empfehlungen angezeigt werden sollen, wird bei der Recommendations-Box ebenfalls über den Parameter site konfiguriert. Anders als der Activity Feed erlaubt dieses Plugin jedoch keine Filterung nach URL-Bestandteilen. Deshalb sollten Sie den Betrieb von mehreren separaten Anwendungen unter der gleichen Domain bei gleichzeitiger Verwendung der Recommendations-Box auf jeden Fall vermeiden, um vermischte Inhalte zu verhindern.

Das HTML5-Tag zur Einbindung der Recommendations-Box sieht folgendermaßen aus:

```
// Einbindung als HTML5-Tag
<div class="fb-recommendations"
     data-site="www.stadtkinder.com"
     data-width="300"
     data-height="400"
     data-header="true">
</div>
```

Listing 6.13 Einbindung der Recommendations-Box als HTML5-Tag

Analog kann die Einbindung mittels XFBML-Tag (`<fb:recommenations>`) oder iFrame (*http://www.facebook.com/plugins/recommendations.php*?...) erfolgen:

```
// Einbindung als XFBML-Tag
<fb:recommendations site="www.stadtkinder.com"
                    width="300" height="400"
                    header="true"></fb:recommendations>

// Einbindung als iFrame
<iframe src="//www.facebook.com/plugins/recommendations.php
   ?site=www.stadtkinder.com
   &amp;width=300
   &amp;height=400
   &amp;header=true
   &amp;colorscheme=light
   &amp;linktarget=_blank
   &amp;border_color
   &amp;font
   &amp;appId=144485858997536"
scrolling="no" frameborder="0"
style="border:none; overflow:hidden; width:300px;
height:400px;" allowTransparency="true">
</iframe>
```

Listing 6.14 Einbindung der Recommendations-Box als XFBML-Tag und iFrame

Nachdem das SDK das HTML5- bzw. XFBML-Tag durch den entsprechenden iFrame ersetzt hat, wird das Plugin folgendermaßen im Browser dargestellt:

Abbildung 6.10 Anzeige des Recommendations-Box-Plugins

Die folgende Tabelle zeigt eine Übersicht über alle verfügbaren Einstellungsmöglichkeiten der Recommendations-Box:

Name	Typ	Beschreibung
`site`	`string`	Kommaseparierte Liste von Domains, von denen Empfehlungen aggregiert werden sollen. In der XFBML-/HTML5-Variante per Default die Domain der aktuellen Webseite, die das Plugin integriert.
`width`	`int`	Breite des Plugins (Default: 300 Pixel)
`height`	`int`	Höhe des Plugins (Default: 300 Pixel)
`font`	`string`	Schriftart, in der die Recommendations-Box angezeigt werden soll. Wählbar sind derzeit `arial`, `lucida grande`, `segoe ui`, `tahoma`, `trebuchet ms`, `verdana`.
`colorscheme`	`string`	Das Farbschema, in dem die Recommendations-Box angezeigt wird, kann entweder auf `light` (Default-Einstellung) oder `dark` eingestellt werden.
`header`	`boolean`	Legt fest, ob die Kopfzeile der Recommendations-Box angezeigt werden soll.
`border_color`	`string`	Farbe des Rahmens in hexadezimaler Notation (z. B. `#FF0000`)

Tabelle 6.6 Parameter der Recommendations-Box

Name	Typ	Beschreibung
linktarget	string	Klickt ein Benutzer auf einen Eintrag der Recommendations-Box, wird per Default ein neues Browser-Fenster geöffnet (_blank). Mit den Werten _top und _parent kann dies zumindest für Inhalte, die nicht auf Facebook.com liegen, verhindert werden.
max_age	int	Erlaubt die Einschränkung der Einträge, die in der Recommendations-Box angezeigt werden, nach ihrem maximalen Alter. Die Angabe erfolgt in Tagen.
ref	string	Ein maximal 50 Zeichen langes Feld mit Tracking-Daten. Wenn ref gesetzt ist, ergänzt Facebook jeden Klick auf einen Eintrag in der Recommendations-Box mit dem GET-Parameter fb_ref, der den Wert von ref enthält.

Tabelle 6.6 Parameter der Recommendations-Box (Forts.)

6.8 Der Live Stream

Das Live-Stream-Plugin ermöglicht Entwicklern, ihre Webseiten und Facebook-Anwendungen auf einfachste Weise um einen Live-Chat zu erweitern. Angenehmer Nebeneffekt ist dabei, dass Textbeiträge im Live-Chat per Default auch automatisch auf der Pinnwand der Teilnehmer veröffentlicht werden, was für eine weitere Verbreitung der eigenen Anwendung sorgen kann.

Zur Einbindung des Plugins Live Stream per HTML5-Tag muss im Parameter event_app_id die numerische ID der eigenen Facebook-Anwendung angegeben werden. Während andere Social Plugins also auch unabhängig von einer Facebook-Anwendung eingesetzt werden können, gilt dies nicht für den Live Stream. Er wird anhand dieser ID eindeutig erkannt, womit seitens Facebook verhindert wird, dass Nachrichten mit anderen Implementationen des Plugins vermischt werden. Darüber hinaus kann im Parameter xid ein zusätzlicher Identifikationsstring übergeben werden. Damit ist es möglich, innerhalb einer Anwendung unterschiedliche, voneinander getrennte Live-Chats einzubinden.

```
// Einbindung als HTML5-Tag
<div class="fb-live-stream"
     data-event-app-id="179099518812387"
     data-xid="home"
```

```
     data-width="400" data-height="500"
     data-always-post-to-friends="false">
</div>
```

Listing 6.15 Einbindung des Live Streams als HTML5-Tag

Analog kann die Einbindung mittels XFBML-Tag (`<fb:live-stream event_app_id>`) erfolgen:

```
// Einbindung als XFBML-Tag
<fb:live-stream event_app_id="179099518812387"
                width="400" height="500"
                xid="home"
                always_post_to_friends="false">
</fb:live-stream>
```

Listing 6.16 Einbindung des Live Streams als XFBML-Tag

Hinweis: Der Live Stream steht nicht in einer iFrame-Version bereit, sondern muss mittels XFBML oder HTML5 eingebunden werden!

Nachdem das SDK das XFBML- bzw. HTML5-Tag durch den entsprechenden iFrame ersetzt hat, wird der Live Stream folgendermaßen dargestellt:

Abbildung 6.11 Realisierung der Chat-Funktion mittels Live-Stream-Plugin

Mit folgenden Einstellungen kann das Live-Stream-Plugin detailliert konfiguriert werden:

Name	Typ	Beschreibung
`event_app_id`	`int`	numerische ID der Anwendung, mit der der Live Stream verknüpft werden soll
`sid`	`string`	Identifikationsstring – dieser kann gesetzt werden, um unterschiedliche Live-Stream-Plugins innerhalb der gleichen Anwendung zu nutzen.
`width`	`int`	Breite des Plugins (Default: 400 Pixel)
`height`	`int`	Höhe des Plugins (Default: 500 Pixel)
`always_post_to_friends`	`boolean`	Ist dieses Feld auf `true` gesetzt, werden alle Chat-Beiträge von Benutzern zwingend auch auf deren Pinnwänden veröffentlicht – die Checkbox zur Deaktivierung dieser Funktion wird dabei ausgeblendet.
`via_url`	`string`	Unter den auf der Pinnwand veröffentlichten Chat-Beiträgen wird normalerweise das Applikationsprofil verlinkt. Mit diesem Parameter kann eine alternative URL für diesen Link angegeben werden.

Tabelle 6.7 Parameter des Live Streams

6.9 Die Kommentarbox

Mit der Kommentarbox steht Entwicklern ein mächtiges Plugin zur Realisierung von Benutzerkommentaren zur Verfügung, das mit einer einzigen Code-Zeile in eigene Webseiten und Anwendungen integriert werden kann. Neben Facebook-Benutzern steht die Kommentarbox auch Benutzern von Yahoo!, AOL und Hotmail zur Verfügung – der Login zu diesen Services kann mittels Pop-up direkt aus der Kommentarbox heraus angestoßen werden. Das Plugin selbst bietet eine chronologische Ansicht der Kommentare mit einstufig verschachtelten Antwort-Posts. Ähnlich der Kommentarfunktion auf der Facebook-Pinnwand findet sich auch hier die Möglichkeit, einzelne Beiträge mit Gefällt mir auszuzeichnen. Daraus und aus der Anzahl der Kommentare ermittelt Facebook die Top-Kommentatoren einer Webseite, die mit einem kleinen Sternchen herausgehoben werden. Überhaupt ist die Darstellung der teilnehmenden Personen im Vergleich zu herkömmlichen Kommentarfunktionen auf Webseiten sehr gut gelöst: So sieht man neben dem realen Namen und dem Profilbild der Benutzer auch gleich deren Beruf und Tätigkeit – dies hilft Lesern, Kommentare besser nach ihrer Relevanz einzuordnen.

Abbildung 6.12 Die Kommentarbox

Ein weiterer Vorteil der Kommentarbox ist die tiefe Integration mit Facebook.com. So werden etwa Kommentare per Default auch auf der Pinnwand des kommentierenden Benutzers veröffentlicht. Antwortet ein anderer Benutzer auf den eigenen Beitrag, wird dies als Benachrichtigung in der Kopfzeile von Facebook angezeigt.

Abbildung 6.13 Benachrichtigung über eine neue Antwort in der Kommentarbox

Das HTML5-Tag zur Einbindung der Kommentarbox sieht so aus:

```
// Einbindung als HTML5-Tag
<div class="fb-comments"
     data-href="http://apps.mycompany.com"
     data-num-posts="10"
     data-width="470">
</div>
```

Listing 6.17 Einbindung der Kommentarbox als HTML5-Tag

Der Parameter `href` sorgt dabei wie üblich für die eindeutige Identifizierung der Kommentarbox, womit es z. B. möglich ist, auf jeder Beitragsseite eines Blogs ein eigenständiges Kommentarmodul zu realisieren.

Die alternative Einbindung per XFBML-Tag wird ebenfalls unterstützt:

```
// Einbindung als XFBML-Tag
<fb:comments href="http://apps.mycompany.com"
              num_posts="10"
              width="470">
</fb:comments>
```

Listing 6.18 Einbindung der Kommentarbox als XFBML-Tag

Hinweis: Die Kommentarbox steht nicht in einer iFrame-Version zur Verfügung, sondern muss mittels XFBML oder HTML5 eingebunden werden!

Facebook stellt ein weiteres spezielles XFBML/HTML5-Tag bereit, das es ermöglicht, die Anzahl der veröffentlichten Kommentare pro Kommentarbox anzuzeigen. Dies ist z. B. praktisch für Inhaltsverzeichnisse oder Übersichtsseiten von Web Logs, wo die Anzahl der Kommentare je Unterseite dargestellt werden soll.

```
// Einbindung als XFBML-Tag
<fb:comments-count
    href="http://apps.mycompany.com"/></fb:comments-count>

// Einbindung als HTML5-Tag
<span class="fb-comments-count"
      href="http://apps.mycompany.com"></span>
```

Listing 6.19 Einbindung des Kommentarzählers als HTML5- und XFBML-Tag

Die folgende Tabelle zeigt alle möglichen Einstellungen der Kommentarbox:

Name	Typ	Beschreibung
`href`	`string`	URL der Webseite, auf die sich die Kommentarbox beziehen soll. Durch Angabe verschiedener eindeutiger URLs können unterschiedliche, voneinander getrennte Kommentarboxen auf einer Webseite eingesetzt werden.
`width`	`int`	Breite der Kommentarbox in Pixeln. Die minimale empfohlene Breite beträgt 400 Pixel.
`colorscheme`	`string`	Das Farbschema, in dem die Kommentarbox angezeigt wird, kann entweder auf `light` (Default-Einstellung) oder `dark` gesetzt werden.
`num_posts`	`int`	Anzahl der Kommentare, die beim Laden des Plugins angezeigt werden soll. Mit dem Link SHOW MORE können weitere Kommentare angezeigt werden. Dabei ist zu beachten, dass die Höhe des Plugins dynamisch angepasst wird.
`mobile`	`boolean`	Die Kommentarbox wird auf mobilen Endgeräten automatisch in einer optimierten Version angezeigt. Dabei wird der `width`-Parameter ignoriert und stattdessen eine dynamische Breite von 100 % verwendet. Mit dem Parameter `mobile` kann dieses Verhalten explizit ein- oder ausgeschaltet werden.

Tabelle 6.8 Parameter der Kommentarbox

Graph-API-Zugriff auf die Kommentarbox

In bestimmten Fällen möchten Entwickler alle Kommentare, die über die Kommentarbox veröffentlicht wurden, in der eigenen Anwendung bzw. Datenbank weiterverarbeiten. Dies kann etwa zur besseren Suchmaschinenoptimierung sinnvoll sein: Da die Kommentarbox wie alle anderen Social Plugins von Facebook als iFrame realisiert wird, berücksichtigen alle aktuellen Suchmaschinen die Texte der veröffentlichten Kommentare nicht für das Ranking der integrierenden Seite. Ein Lösungsansatz ist, die Kommentare in regelmäßigen Abständen in der eigenen Datenbank zu speichern und als statischen Text in die eigenen Webseiten zu integrieren. Suchmaschinen können so den Kommentartext indizieren, während normalen Besuchern der Seite per JavaScript anstelle des statischen Textes das Kommentar-Plugin angezeigt wird.

Glücklicherweise unterstützt die Graph API das Auslesen der Kommentare über folgenden API-Endpunkt:

```
https://graph.facebook.com/ids=URL
```

Als URL wird dabei die Webseite angegeben, auf der die Kommentarbox integriert wurde bzw. die im `href`-Parameter beim Laden des Plugins angegeben wurde. Die Abfrage der API liefert eine Datenstruktur mit allen Kommentartexten und Informationen zum Absender:


```
{
   "http://apps.mycompany.com": {
      "data": [
         {
            "id": "10150520597013491_20626197",
            "from": {
               "name": "Michael Kamleitner",
               "id": "609190863"
            },
            "message": "Das ist ein Testkommentar.",
            "created_time": "2012-01-08T15:21:44+0000",
            "comments": {
               "data": [
                  {
                     "id": "10150520597918491",
                     "from": {
                        "name": "Klara Brigitte Socialist",
                        "id": "100001903705011"
                     },
                     "message": "Gut so!",
                     "created_time": "2012-01-08T15:21:57+0000"
                  }
               ],
               "count": 1
            }
         },
         ...
```

Listing 6.20 JSON-Array über die Graph API ausgelesener Kommentare

Event-Handler für die Kommentarbox

Anstatt neue Kommentare in regelmäßigen Abständen abzufragen, können Entwickler mit dem JavaScript SDK die Events `comment.create` und `comment.remove`

abonnieren. Diese werden ausgelöst, wenn ein Kommentar hinzugefügt oder gelöscht wird:

```
FB.Event.subscribe('comment.create', function(response) {
   console.debug('comment.create');
   console.debug(response);
});
FB.Event.subscribe('comment.remove', function(response) {
   console.debug('comment.remove');
   console.debug(response);
});
```

Der an den Callback-Handler übergebene `response`-Wert enthält im Feld `href` die URL, mit der die Kommentarbox verknüpft ist, und in `commentID` die numerische ID des veröffentlichten bzw. gelöschten Kommentars. Mit diesen Informationen kann etwa mittels AJAX ein Prozess aufgerufen werden, der über die Graph API den eigentlichen Inhalt des Kommentars ausliest und in der eigenen Datenbank speichert.

```
{
   href: 'http://apps.mycompany.com',
   commentID: 2331111232132
}
```

Die Moderationsfunktion der Kommentarbox

Betreiber beliebter Webseiten und großer Communities kennen das Problem – früher oder später kommt es zum Missbrauch der Kommentarfunktion. Obwohl Facebook mit der Verwendung des echten Namens der Benutzer hier sehr grundlegend gegensteuert, kann es natürlich auch hier den Bedarf nach Moderation von Kommentaren geben.

Um die Kommentarbox von Facebook zu moderieren, ist es zuerst notwendig, jene Benutzer festzulegen, die eine Moderationsberechtigung erhalten sollen. Dies geschieht über die Angabe des speziellen Open Graph Tags `<fb:admins>` auf jener Webseite, auf der das Kommentar-Plugin eingebunden werden soll:

```
<head>
   <meta property="fb:admins"
         content="michael.kamleitner,100001434672472"/>
   ...
</head>
```

Im Attribut `content` wird dabei eine kommaseparierte Liste von Benutzer-IDs oder Benutzernamen der Moderatoren angegeben. Facebook liest diesen Wert aus und bestimmt daraufhin, ob der aktuelle Benutzer Administrationsberechtigungen hat oder nicht. Alternativ kann mit `<fb:app_id>` auch die ID einer mit der Kommentarbox verknüpften Anwendung angegeben werden. Facebook erteilt dann automatisch allen in der Anwendung eingetragenen Entwicklern Moderationsrechte in der Kommentarbox.


```
<head>
   <meta property="fb:app_id" content="214728715257742"/>
   ...
</head>
```

Benutzer mit Moderationsberechtigung finden anschließend am oberen Rand der Kommentarbox einen mit MODERATORENANSICHT beschrifteten Link, der folgendes Interface zur Moderation der Kommentare öffnet:

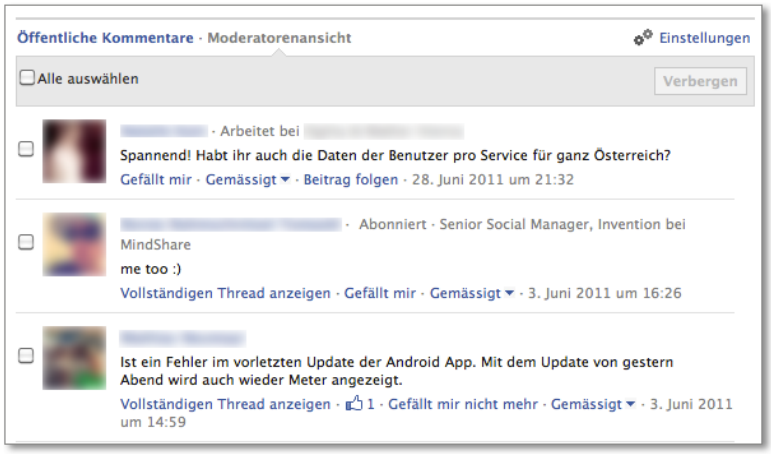


Abbildung 6.14 Moderationsansicht der Kommentarbox

Hier können Administratoren einzelne Kommentare verbergen, Benutzer von der Nutzung der Kommentarfunktion ausschließen oder besonders informative Kommentare aufwerten (»boosten« – damit werden diese Kommentare weiter oben im Stream angezeigt).

Darüber hinaus haben Administratoren der Kommentarbox Zugriff auf einen zusätzlichen Einstellungsdialog, der folgende Möglichkeiten bietet:

- Aktivierung und Deaktivierung des Moderationsmodus, der vor der Veröffentlichung jedes neuen Kommentars eine Bestätigung durch einen Administrator erfordert

- Aktivierung und Deaktivierung der Login-Möglichkeit für AOL-, Yahoo!- und Hotmail-Benutzer
- Aktivierung und Deaktivierung der automatischen Korrektur häufig auftretender Grammatikfehler
- Anlegen und Verwalten einer Liste für Kommentare gesperrter Wörter (*Blacklisting*) oder Nutzung der von Facebook geführten Liste
- Aktivierung und Deaktivierung der Anzeige von zusätzlichen Benutzerinformationen, wenn mehr als fünf Kommentare veröffentlicht wurden
- Anzeige aller mittels `<fb:admin>` und `<fb:app_id>` definierten Administratoren sowie die Möglichkeit, weitere Administratoren hinzuzufügen

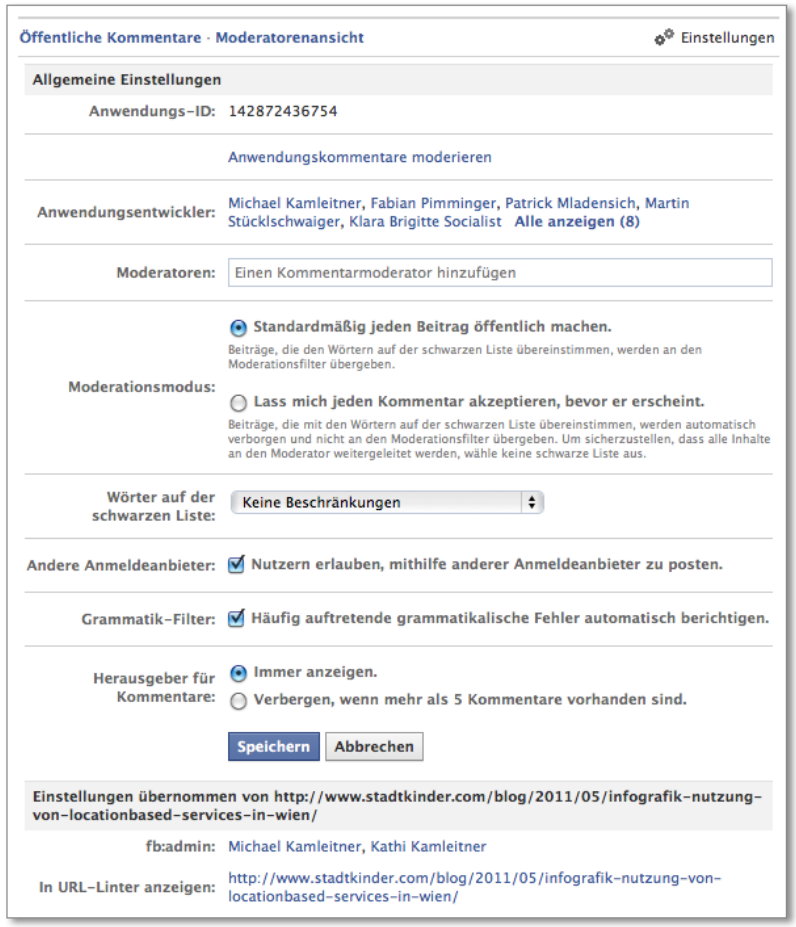


Abbildung 6.15 Zusätzliche Einstellungen der Kommentarbox

Hinweis: Alle Veränderungen im Einstellungsdialog wirken automatisch für alle Implementierungen der Kommentarbox unter der gleichen Domain!

6.10 Der »Log-In«-Button

Der Log-In-Button ist ein einfaches Plugin, das verwendet werden kann, um Benutzer zur Autorisierung der eigenen Anwendung aufzufordern. Der eingeblendete Button tut nichts anderes, als die bereits bekannte SDK-Methode `FB.login()` aufzurufen. Autorisiert der Benutzer die Anwendung, wird anschließend die Webseite, auf der das Plugin integriert wurde, neu geladen. Ein besonderes Merkmal des Log-In-Buttons ist die Möglichkeit, eine Leiste mit den Profilbildern jener Freunde einzublenden, die die Anwendung bereits installiert haben. Dies führt üblicherweise zu einer besseren Konversionsrate bei neuen Benutzern.

Abbildung 6.16 »Log-In«-Button mit angezeigter Profilbild-Leiste

Die Einbindung des Log-In-Buttons erfolgt mit folgendem HTML5-Tag:

```
// Einbindung als HTML5-Tag
<div class="fb-login-button"
     data-show-faces="true"
     data-width="200"
     data-max-rows="1">
</div>
```

Listing 6.21 Einbindung des »Log-In«-Buttons als HTML5-Tag

Hinweis: Die ID der Anwendung, für die der Login-Prozess ausgeführt werden soll, kann nicht beim Laden des Log-In-Buttons festgelegt werden, sondern wird automatisch von jener Anwendungs-ID übernommen, die beim Initialisieren des SDKs mit `FB.init()` angegeben wurde.

Alternativ kann auch dieses XFBML-Tag benutzt werden:

```
// Einbindung als XFBML-Tag
<fb:login-button show-faces="true"
                 width="200" max-rows="1">
</fb:login-button>
```

Listing 6.22 Einbindung des »Log-In«-Buttons als XFBML-Tag

Hinweis: Die Kommentarbox steht nicht in einer iFrame-Version zur Verfügung, sondern muss mittels XFBML oder HTML5 eingebunden werden!

Die folgende Tabelle zeigt alle möglichen Einstellungen des Log-In-Buttons:

Name	Typ	Beschreibung
`show-faces`	`boolean`	Legt fest, ob unter dem Log-In-Button eine Leiste mit den Profilbildern von Freunden, die die Anwendung bereits installiert haben, angezeigt werden soll.
`width`	`int`	Breite des Plugins (Default: 200 Pixel)
`max-rows`	`int`	Legt fest, in wie vielen Zeilen Profilbilder dargestellt werden sollen (Default: 1).
`scope`	`string`	Kommaseparierte Liste an Berechtigungen, die bei der Autorisierung der Anwendung abgefragt werden sollen

Tabelle 6.9 Parameter des »Log-In«-Buttons

6.11 Die Facepile-Box

Die *Facepile-Box* blendet eine Profilbild-Leiste von Freunden oder Benutzern ein, die bereits Fan einer Seite sind oder bereits eine bestimmte Anwendung installiert haben. Auch die Facepile-Box dient in erster Linie dazu, neuen Besuchern anzuzeigen, welcher ihrer Freunde bereits mit der Seite oder Anwendung verknüpft ist.

- Wird im Parameter `href` die URL zu einer Facebook-Seite angegeben, zeigt Facepile die Profilbilder von Freunden des aktuellen Benutzers an, die bereits Fan der Seite sind. Ist noch kein Freund des aktuellen Benutzers Fan der Seite, werden Profilbilder von anderen, unbekannten Benutzern und die Gesamtzahl der Fans angezeigt.
- Wird der Parameter `href` nicht angegeben, zeigt Facepile die Profilbilder von Freunden des aktuellen Benutzers an, die die aktuelle Anwendung bereits installiert haben. Hat noch kein Freund des aktuellen Benutzers die Anwendung installiert, werden keine Profilbilder angezeigt.

Hinweis: Die ID der Anwendung, für die der Facepile-Box-Prozess ausgeführt werden soll, kann nicht beim Laden des Plugins festgelegt werden, sondern wird automatisch von jener Anwendungs-ID übernommen, die beim Initialisieren des SDKs mit `FB.init()` angegeben wurde.

Abbildung 6.17 Facepile-Boxen für Seite und Anwendung

Das HTML5-Tag zur Einbindung von Facepile sieht folgendermaßen aus:

```
// Einbindung als HTML5-Tag
<div class="fb-facepile"
     data-href="http://apps.mycompany.com"
     data-width="200" data-max-rows="1">
</div>
```

Listing 6.23 Einbindung der Facepile-Box als HTML5-Tag

Alternativ unterstützt Facepile auch die Einbindung per XFBML-Tag oder iFrame:

```
// Einbindung als XFBML-Tag
<fb:facepile href="http://apps.mycompany.com"
             width="200" max_rows="1"></fb:facepile>

// Einbindung als iFrame
<iframe src="//www.facebook.com/plugins/facepile.php
   ?href=http%3A%2F%2Fapps.mycompany.com
   &amp;size=small&amp;width=200
   &amp;max_rows=1
   &amp;colorscheme=light
   &amp;appId=144485858997536"
scrolling="no" frameborder="0" style="border:none; overflow:hidden;
width:200px;" allowTransparency="true">
</iframe>
```

Listing 6.24 Einbindung der Facepile-Box als XFBML-Tag und iFrame

Die folgende Tabelle zeigt alle möglichen Einstellungen der Facepile-Box:

Name	Typ	Beschreibung
`href`	`string`	URL der Facebook-Seite, deren Fans in der Facepile-Box dargestellt werden sollen (Angabe alternativ zu `app_id`)
`app_id`	`int`	ID der Anwendung, deren Benutzer in der Facepile-Box dargestellt werden sollen. Angabe alternativ zu `href` – nur bei Einbindung mittels iFrame zulässig – XFBML- und HTML5-Variante übernehmen die Anwendungs-ID automatisch von `FB.init()`.
`width`	`int`	Breite der Facepile-Box in Pixeln (Default: 200 Pixel)
`colorscheme`	`string`	Das Farbschema, in dem die Kommentarbox angezeigt wird, kann entweder auf `light` (Default-Einstellung) oder `dark` eingestellt werden.
`max-rows`	`int`	Legt fest, in wie vielen Zeilen Profilbilder dargestellt werden sollen (Default: 1).
`size`	`string`	Größe, in der Profilbilder dargestellt werden sollen (`small` oder `large`)

Tabelle 6.10 Parameter der Facepile-Box

6.12 Die Registrierungsbox

Die *Registrierungsbox* erlaubt es Entwicklern, einen möglichst einfachen Registrierungsprozess auf externen Websites oder Anwendungen zu ermöglichen. Die Grundidee ist dabei, das Formular zur Registrierung so weit wie möglich mit Daten aus dem Facebook-Konto des Benutzers im Vorhinein auszufüllen. Diese Daten werden anschließend in Form eines speziellen Signed Requests (siehe Abschnitt 2.4, »Signed Request«) an die Website übergeben, die die Registrierungsbox eingebaut hat. Benutzern, die kein Facebook-Konto besitzen oder dieses nicht für die Registrierung verwenden wollen, bietet die Registrierungsbox eine Möglichkeit, das Formular auf herkömmlichem Weg auszufüllen. Darüber hinaus erlaubt das Plugin dem Entwickler, detailliert festzulegen, welche Informationen ein Benutzer bei der Registrierung bereitstellen soll.

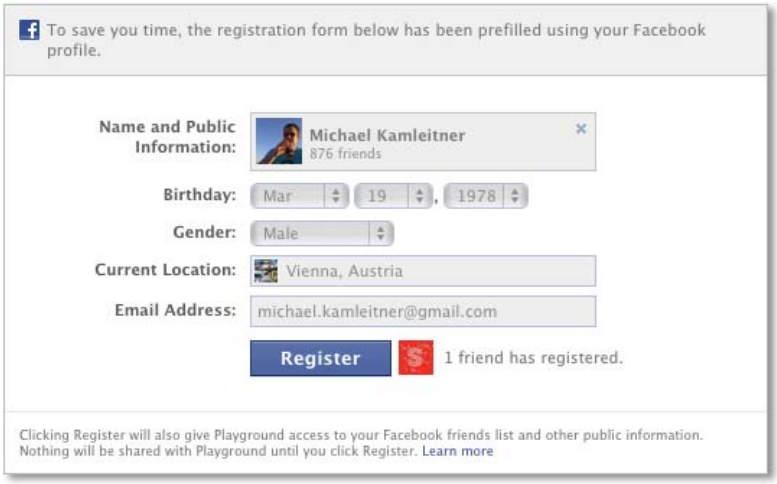


Abbildung 6.18 Die Registrierungsbox füllt das Registrierungsformular automatisch mit den Benutzerinformationen des Facebook-Kontos.

Hinweis: Der Begriff *Registrierung* bedeutet in Zusammenhang mit diesem Plugin, dass Besucher zur Autorisierung bzw. Installation der dahinterliegenden Facebook-Anwendung aufgefordert werden. Da die Registrierungsbox aber keinen Parameter `scope` kennt, können dabei leider keine erweiterten Berechtigungen vom Benutzer eingefordert werden. Stattdessen erteilt der Anwender automatisch Zugriff auf jene Profildaten, die für das Registrierungsformular vorgesehen sind, also etwa Geburtsdatum und aktuellen Aufenthaltsort. Etwas inkonsequent ist die Registrierungsbox dabei insofern, als dass sie nicht den gewohnten OAuth-Dialog zur Bestätigung nutzt. Nach dem Klick auf REGISTER ist die Anwendung sofort autorisiert, dem Benutzer wird im daraufhin folgenden Bestätigungsfenster allerdings die Möglichkeit gegeben, die Registrierung rückgängig zu machen.

Abbildung 6.19 Bestätigung der Registrierung/Autorisierung der Anwendung

Die Registrierungsbox kann mit folgenden HTML5- bzw. XFBML-Tags auf der eigenen Webseite eingebunden werden:

```
// Einbindung als HTML5-Tag
<div class="fb-registration"
     data-fields="name,birthday,gender,location,email"
     data-redirect-uri=https://dev.diesocialisten.at/book/7
     socialplugins-registration.php"http://apps.mycompany.com"
     data-only="false"
     width="530">
</div>

// Einbindung als XFBML-Tag
<div class="fb-registration"
     data-fields="name,birthday,gender,location,email"
     data-redirect-uri=http://apps.mycompany.com
     data-only="false"
     width="530">
</div>
```

Listing 6.25 Einbindung der Registrierungsbox als HTML5- und XFBML-Tag

Hinweis: Die ID der Anwendung, für die die Registrierung angeboten werden soll, kann bei den XFBML- und HTML5-Varianten nicht beim Laden des Plugins festgelegt werden, sondern wird automatisch von jener Anwendungs-ID übernommen, die beim Initialisieren des SDKs mit `FB.init()` angegeben wurde. Im Gegensatz dazu muss bei der ebenfalls unterstützten Einbindung als iFrame die ID der Anwendung im Parameter `client_id` übergeben werden:

```
// Einbindung als iFrame
<iframe src=https://www.facebook.com/plugins/registration.php
            ?client_id=214728715257742
            &redirect_uri=https%3A%2F%2Fapps.mycompany.com
            &fields=name,birthday,gender,location,email
scrolling="auto"
frameborder="no"
style="border:none"
allowTransparency="true"
width="350" height="350">
</iframe>
```

Listing 6.26 Einbindung der Registrierungsbox als iFrame

Mit folgenden Parametern kann die Registrierungsbox detaillierter konfiguriert werden:

Name	Typ	Beschreibung
`client_id`	`int`	Numerische ID der Anwendung, für die der Registrierungsprozess durchgeführt werden soll. Angabe nur bei Einbindung mittels iFrame zulässig – XFBML- und HTML5-Variante übernehmen die Anwendungs-ID automatisch von `FB.init()`.
`width`	`int`	Breite des Plugins. Bei einer Breite unter 520 Pixeln wird das Plugin in einer speziellen, schmaleren Variante angezeigt.
`redirect_uri`	`string`	URL, die nach dem Durchführen der Registrierung aufgerufen wird und an die die Registrierungsdaten in Form eines Signed Requests übergeben werden
`fields`	`string`	Legt fest, welche Felder im Registrierungsformular abgefragt werden sollen. Angabe entweder als kommaseparierte Liste mit Standardfeldern oder als JSON-Objekt mit individuellen Felddefinitionen.
`fb_only`	`boolean`	Legt fest, ob sich Benutzer nur mittels Facebook-Konto oder auch auf herkömmlichem Weg registrieren können (Default: `false`).
`fb_register`	`boolean`	Legt fest, ob sich Benutzer, die noch keinen Facebook-Account besitzen, direkt innerhalb des Plugins für Facebook registrieren können (Default: `false`).
`border_color`	`string`	Farbe des Rahmens in hexadezimaler Notation (z. B. `#FF0000`)
`target`	`string`	Browser-Fenster bzw. Frameset, in dem nach Abschluss des Registrierungsprozesses die `redirect_uri` aufgerufen werden soll. `_top` (Default) öffnet die Seite im obersten Frameset des Browser-Fensters, `_set` im darüberliegenden Frameset und `_self` im iFrame des Plugins.

Tabelle 6.11 Parameter der Registrierungsbox

Lesen der Registrierungsdaten aus dem Signed Request

Nach dem erfolgreichen Abschluss der Registrierung leitet das Plugin den Browser an die im Parameter `redirect_uri` angegebene URL weiter. Im GET-Parameter `signed_request` werden dabei die Registrierungsdaten übergeben, die, wie in Abschnitt 2.4, »Signed Request«, erklärt, mit dem App Secret decodiert werden können. Das resultierende Array enthält neben dem OAuth-Token und der Facebook-Benutzer-ID auch alle im Formular übergebenen Informationen:

```
Array
(
    [algorithm] => HMAC-SHA256
    [expires] => 1326308400
    [issued_at] => 1326304716
    [oauth_token] => AAADDS2zLx44BAEpoT1DUbZBZCgjPVH...
    [registration] => Array
        (
            [name] => Michael Kamleitner
            [birthday] => 03/19/1978
            [gender] => male
            [location] => Array
                (
                    [name] => Vienna, Austria
                    [id] => 111165112241092
                )
            [email] => michael.kamleitner@...
        )
    [registration_metadata] => Array
        (
            [fields] => name,birthday,gender,location,email
        )
    [user] => Array
        (
            [country] => at
            [locale] => en_US
        )
    [user_id] => 609190863
)
```

Listing 6.27 Entschlüsselter Signed Request der Registrierungsbox

- Die Felder `algorithm`, `expires`, `issued_at`, `oauth_token`, `user_id` und das Array `user` werden dabei ebenso wie beim Signed Request auf Canvas-Seiten verwendet.
- Das Array `registration_metadata` listet alle Felder, die im Registrierungsformular angezeigt wurden.
- Das Array `registration` enthält die eigentlichen Registrierungsdaten in separaten Feldern deren Schlüssel den in `fields` verwendeten Feldnamen entspricht.

Das folgende Code-Beispiel zeigt die Einbindung der Registrierungsbox sowie die Decodierung des Signed Requests nach Absenden des Formulars:

```
<?
include_once('tools.php');
define('APP_ID', '214728715257742');
define('APP_SECRET', 'b93bff303c6199d959ad8465...');
define('SITE_URL', 'http://apps.mycompany.com');
?><DOCTYPE html>
<html lang="de-de" xmlns=http://www.w3.org/1999/xhtml
      xmlns:fb="http://ogp.me/ns/fb#">
<head>
    <meta charset="utf-8">
    <title>Hello Facebook JS SDK!</title>
</head>
<body>
<h2>Registration</h2>
<?
if (@$_REQUEST["signed_request"]) {
   $signed_request =
   parse_signed_request($_REQUEST["signed_request"], APP_SECRET);
   print "<pre>"; print_r($signed_request); print "</pre>";

   // Hier können die Registrierungsdaten verarbeitet werden ...
} else { ?>
   <fb:registration fields="name,birthday,gender,location,email"
                    redirect-uri=http://apps.mycompany.com
                    fb-only="false"
                    fb-register="true"
                    width="530">
   </fb:registration>
<? } ?>
<div id="fb-root"></div>
<script>
```

```
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
      channelUrl : '//apps.mycompany.com/channel.php',
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
</script>
</body>
</html>
```

Listing 6.28 Anzeige der Registrierungsbox und Entschlüsseln des Signed Requests

Anmerkungen zum Beispiel-Code:

- Die meisten Bestandteile des Beispiel-Codes sind mittlerweile gut bekannt: Das JavaScript SDK wird initialisiert und mit `xfbml=true` zum Parsen von XFBML-Tags konfiguriert.
- Ist der GET-Parameter `$_REQUEST['signed_request']` nicht gesetzt, bedeutet dies, dass das Registrierungsformular noch nicht abgeschickt wurde. In diesem Fall wird das Formular mit dem XFBML-Tag `<fb:registration>` angezeigt.
- Andernfalls können der Signed Request mit der in Abschnitt 2.4, »Signed Request«, eingeführten Hilfsfunktion `parse_signed_request()` decodiert und die Registrierungsdaten weiterverarbeitet werden.

Anpassung des Registrierungsformulars

Die Registrierungsbox unterstützt standardmäßig folgende Felder, die durch Angabe als kommaseparierte Liste im Parameter `fields` beim Laden des Plugins festgelegt werden können:

Bezeichnung	Beschreibung	Default
name	vollständiger Name des Benutzers	ja
first_name	Vorname	nein
last_name	Nachname	nein
birthday	Geburtsdatum im Format MM/DD/YYYY	ja
email	E-Mail-Adresse des Benutzers	ja
gender	Geschlecht (male , female oder leer)	ja
location	Objekt mit den Feldern name und id des aktuellen Aufenthaltsortes	ja
password	Feld zur Eingabe eines Passworts für die registrierende Anwendung (unabhängig vom Kennwort des Facebook-Kontos)	nein
captcha	Bildschutz gegen automatische Registrierungen	nein

Tabelle 6.12 Standardfelder der Registrierungsbox

Möchte ein Entwickler zusätzliche, individuelle Felder im Registrierungsformular verwenden, kann dies über die Angabe eines JSON-Objekts im Parameter `fields` erfolgen. `fields` muss dabei ein Array von Objekten enthalten, die in den folgenden Feldern die Eigenschaften des Formulars näher spezifizieren:

Feld	Beschreibung
name	Bezeichnung des Feldes, die als name-Attribut im HTML-Formular und als Bezeichner im Signed Request benutzt wird
description	Feldbeschreibung, die dem Benutzer neben dem Feld im <label> angezeigt wird
type	Datentyp des Feldes – zulässige Typen sind text, date, select, checkbox, typeahead und hidden.
view	Gibt an, ob das Feld nur angezeigt werden soll, wenn sich der Benutzer mittels Facebook-Konto registriert (prefilled) oder nur, wenn der Benutzer auf herkömmlichem Weg das Formular ausfüllt (not_prefilled). Ohne Angabe dieses Parameters wird das Feld in beiden Fällen angezeigt.

Tabelle 6.13 Eigenschaften individueller Felder der Registrierungsbox

Feld	Beschreibung
options	Für Felder vom Typ select wird in diesem Parameter als Key-Value-Paar ein Array an Auswahloptionen angegeben.
categories	Für Felder vom Typ typeahead, die ein automatisch vervollständigendes Eingabefeld auf Basis von Open-Graph-Protocol-Objekttypen realisieren. Das hier angegebene Array legt fest, welche Objekttypen im Feld unterstützt werden.
default	Enthält für Felder vom Typ checkbox den Default-Zustand (checked) und für Felder vom Typ select den Schlüssel der vorselektierten Option.

Tabelle 6.13 Eigenschaften individueller Felder der Registrierungsbox (Forts.)

Das folgende Beispiel zeigt ein JSON-Objekt zur individuellen Konfiguration der Felder in der Registrierungsbox. Auffallend ist dabei, dass die von Facebook bereitgestellten Standardfelder (name, email, location ...) ohne nähere Definition verwendet werden:

```
[
 {'name':'name'},
 {'name':'email'},
 {'name':'location'},
 {'name':'gender'},
 {'name':'birthday'},
 {'name':'password'},
 {'name':'like', 'description':'Do you like this plugin?',
                'type':'checkbox',  'default':'checked'},
 {'name':'phone', 'description':'Phone Number',
                 'type':'text'},
 {'name':'anniversary', 'description':'Anniversary',
                       'type':'date'},
 {'name':'captain', 'description':'Best Captain',
                   'type':'select',
                   'options':{'P':' Picard','K':'Kirk'}},
 {'name':'force', 'description':'Which side?',
                 'type':'select',
                 'options':{'jedi':'Jedi','sith':'Sith'},
                 'default':'sith'},
```

```
 {'name':'live', 'description':'Best Place to Live',
                 'type':'typeahead',
                 'categories':['city','country']},
 {'name':'captcha'}
]
```

Listing 6.29 JSON-Objekt zur Konfiguration der Felder der Registrierungsbox

Kapitel 7
Mobile Webanwendungen auf Facebook

In diesem Kapitel erfahren Sie, wie Sie Ihre webbasierten Facebook-Anwendungen fit fürs Smartphone machen und Ihrer Applikation so zu noch mehr Verbreitung verhelfen.

Dank der Verfügbarkeit von nativen Clients für die beliebtesten mobilen Plattformen wie iOS und Android ist die mobile Nutzung von Facebook im Vormarsch – über 350 Millionen Benutzer greifen pro Monat über ein mobiles Endgerät auf Facebook zu. Neben den nativen SDKs bietet die Facebook-Plattform seit Kurzem auch die Möglichkeit, webbasierte Anwendungen für mobile Endgeräte zu optimieren.

Schon bald nach der Einführung der zweiten Generation von Apples iPhone, die erstmals die Möglichkeit bot, mittels App Store Anwendungen von Drittherstellern auf einfachstem Weg auf einem Smartphone zu installieren, veröffentlichte Facebook einen offizielle, native Client-Software für das Gerät. *Native Client-Software* bezeichnet dabei ein Programm, das speziell für eine bestimmte Hardware-Plattform entwickelt und im Binärformat kompiliert wurde – der beliebte Facebook-Client für das iPhone war also nur auf dieser Plattform lauffähig und musste später für die Android-Plattform von Grund auf neu entwickelt werden.

Grundsätzlich können *native Anwendungen* auf mobilen Plattformen sehr einfach an die Facebook-Plattform andocken – durch Nutzung von OAuth und der HTTP-REST-basierten Graph API sind etwa die Autorisierung einer mobilen Anwendung mittels Facebook-Konto und der Zugriff auf die Profilinformationen leicht zu realisieren. Vor allem aus Benutzersicht ist dabei störend, dass die bekannten Facebook-Dialoge zur Autorisierung, zum Verfassen von Wall-Postings etc. nur durch Laden in einem eingebetteten Browser-Fenster (etwa mittels `UIWebView`-Klasse auf iOS bzw. der `WebView`-Klasse auf Android) angezeigt werden können. Diese Art der Benutzerführung ist meist ungewohnt für den Benutzer, lädt mitunter langsamer und entspricht generell nicht dem Look & Feel der jeweiligen mobilen Plattform. Daher hat Facebook schon sehr bald *offizielle SDKs für iOS und Android* bereitgestellt, die Entwickler von nativen Anwendungen einfach in eigene Projekte einbinden können. Die genannten SDKs bieten ähnlich wie das JavaScript SDK Methoden für verschiedene Aspekte der Facebook-Plattform:

- Zugriff auf die *Graph API und FQL-Tabellen* (über den Graph-API-Endpunkt `/fql`)
- *Autorisieren von Benutzern und Anwendungen* um den Login mittels Facebook-Konto auch auf mobilen, nativen Anwendungen zu ermöglichen
- Aufruf von *Facebook-Dialogen* – derzeit werden nur der Feed- und der OAuth-Dialog (Android, iOS) bzw. zusätzlich der Requests-Dialog (iOS) unterstützt.

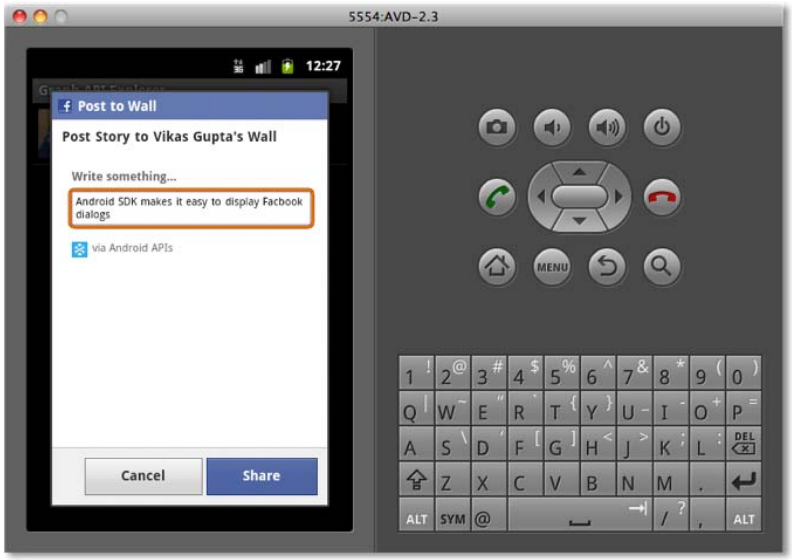


Abbildung 7.1 Dieser Feed-Dialog wurde über das Android SDK geöffnet (Ansicht im Soft-Emulator).

Wichtige Links:

- *http://developers.facebook.com/docs/mobile/ios/build/* – Tutorial zur Entwicklung von nativen iOS-Anwendungen
- *http://developers.facebook.com/docs/reference/iossdk/* – offizielle Dokumentation zum iOS SDK
- *http://developers.facebook.com/docs/mobile/android/build/* – Tutorial zur Entwicklung von nativen Android-Anwendungen
- *http://developers.facebook.com/docs/reference/androidsdk/* – offizielle Dokumentation zum Android SDK

Da die Entwicklung von nativen mobilen Anwendungen je nach Ziel-Plattform ein sehr individuelles, umfassendes Themengebiet ist, möchte ich es für dieses Buch bei dieser kurzen Zusammenfassung der Möglichkeiten der nativen SDKs belassen. Viel spannender – jedenfalls aus der Sicht des Webentwicklers – ist hingegen die im November neu veröffentlichte Unterstützung der Facebook-Plattform für *mobile Webanwendungen*, die ich im folgenden Abschnitt näher beschreiben möchte.

7.1 Einrichten der Mobile Web URL

Obwohl Canvas-Anwendungen auf der Facebook-Plattform schon seit geraumer Zeit auf offenen, prinzipiell auch mobilen Technologien wie HTML, CSS, JavaScript und vor allem der Einbindung mittels iFrame setzen, war der Betrieb von Canvas-Anwendungen bisher nur auf Desktop-Browsern möglich. Mobile Anwender wurden im besten Fall auf die am Smartphone nur schlecht bedienbare Desktop-Version (*http://apps.facebook.com/AppNamespace...*) weitergeleitet, im schlechtesten Fall wurde nur eine Fehlermeldung angezeigt.

Ähnlich sieht es leider bis heute mit Seiten-Tabs/Reitern aus – öffnet ein Benutzer mittels iOS- oder Android-Client ein Seitenprofil, werden am Desktop verfügbare Tabs schlichtweg nicht angezeigt. Lediglich die Standard-Tabs Wall, Info und Photos sind auf dem mobilen Endgerät verfügbar. Für die kommerzielle Nutzung von Facebook-Seiten ist das natürlich problematisch, da Promotionen, Gewinnspiele und sonstige individuelle Inhalte, die normalerweise als Tabs abgebildet werden, nicht in die mobile Präsenz integriert werden können.

Zumindest für Canvas-Applikationen bietet die Facebook-Plattform nun seit November 2011 deutlich bessere Integrationsmöglichkeiten auf mobilen Endgeräten mit iOS-Betriebssystem und bei Zugriffen über die mobile Facebook-Version auf *http://m.facebook.com*. Dazu müssen Sie als Entwickler eine spezielle URL bereitstellen, über die Facebook eine für mobile Endgeräte optimierte Version Ihres Canvas-Inhalts zur Verfügung stellt. Diese tragen Sie unter Mobile Web URL in den Einstellungen einer bestehenden oder neuen Anwendung ein. Bitte beachten Sie dabei, dass die URL wie auch die Canvas URL innerhalb der unter App Domain angegebenen Domain (also etwa `mycompany.com`) liegen muss:

Abbildung 7.2 Einstellen der mobilen Web-URL in den Anwendungseinstellungen

Diese Einstellung bewirkt, dass Facebook bei Zugriffen auf die Canvas-Anwendung über *http://apps.facebook.com/AppNamespace* automatisch entscheidet, von welcher URL der Content vom Anwendungsserver bezogen werden soll: Bei Desktop-Browsern wird automatisch die Canvas URL bzw. bei Zugriffen über SSL die Secure Canvas URL verwendet, bei Zugriffen über iOS-Geräte oder die mobile Facebook-Version auf *http://m.facebook.com* wird der Anwendungsinhalt automatisch über die Mobile Web URL bezogen. Als Entwickler sind Sie dafür zuständig, unter beiden URLs passende Inhalte auszuliefern.

Hinweis: Facebook bietet derzeit keine explizite Option zum Setzen einer Secure Mobile Web URL, Sie sollten also das HTTPS-Protokoll durchgängig verwenden, um auch Ihren mobilen Benutzern sicheres Browsen zu ermöglichen!

Tragen Sie zum Testen der mobilen Fähigkeiten Ihrer Anwendung *http://apps.mycompany.com/mobile/* als Mobile Web URL ein, und legen Sie im Root-Verzeichnis Ihres Webservers ein Unterverzeichnis namens `mobile` an. In diesem Verzeichnis erstellen Sie nun eine Datei namens `index.php` mit folgendem Inhalt:

```
<DOCTYPE html>
<html lang="de-de" xmlns="http://www.w3.org/1999/xhtml"
      xmlns:fb="http://ogp.me/ns/fb#">
<head>
   <meta charset="utf-8"/>
   <meta name="viewport"
         content="width=device-width, initial-scale=1">
   <title>Mobile Facebook</title>
</head>
<body>
   <h1>Mobile Web-Apps auf Facebook</h1>
   <div id="fb-root"></div>

<script>
window.fbAsyncInit = function() {
    FB.init({
      appId      : '214728715257742',
                   // Anwendungs-ID
      channelUrl : '//apps.mycompany.com/channel.php',
                   // Channel-Datei
      status     : true, // Login-Status prüfen
      cookie     : true, // Cookies aktivieren
      oauth      : true, // OAuth 2.0 aktivieren
      xfbml      : true  // XFBML parsen
    });
};
// Asynchrones Laden des SDKs
(function(d){
 var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
}(document));
```

```
</script>
</body>
</html>
```

Listing 7.1 Grundgerüst einer mobilen Facebook-Webanwendung

Das Hinterlegen einer eigenen MOBILE WEB URL bewirkt, dass Facebook beim Aufrufen des Application Canvas automatisch erkennt, mit welcher Art von Endgerät auf die Applikation zugegriffen wird, und entsprechend den passenden Inhalt ausliefern kann:

- Erfolgt der Zugriff über einen Desktop-Browser, liefert Facebook automatisch den Inhalt der unter CANVAS URL angegebenen Webseite aus.
- Wird hingegen von einem unterstützten Smartphone auf die Anwendung zugegriffen, erfolgt die Auslieferung der unter MOBILE WEB URL angegebenen Webseite. Als Entwickler müssen Sie dafür Sorge tragen, dass die hier ausgelieferten Inhalte an mobile Endgeräte angepasst sind – Facebook kann Ihnen die Erstellung spezieller mobiler Webseiten also nicht abnehmen.

Der Aufruf des Application Canvas kann dabei auf unterschiedliche Arten erfolgen, so können Benutzer etwa die URL der Anwendung gemäß dem Schema *http://apps.facebook.com/AppNamespace* direkt aufrufen oder etwa aus der nativen iOS-Facebook-Anwendung über den Link eines Wall-Postings auf das Application Canvas gelangen.

Abbildung 7.3 Anzeige des mobilen Application Canvas bei Direkt-Aufruf (links) und Aufruf aus der Facebook-iOS-Anwendung (rechts)

Je nach Art des Aufrufs und je nach Endgerät kommt es dabei zu unterschiedlichen Darstellungen: Bei einem direkten Aufruf der Canvas-URL erfolgt üblicherweise ein Redirect auf *http://m.facebook.com/AppNamespace* – die Anwendung wird dabei also in einem neuen, leeren Browser-Tab geladen. Wird die Canvas-URL hingegen aus der nativen Facebook-iOS-Anwendung geladen, erfolgt die Darstellung innerhalb der nativen Applikation, gut erkennbar am blauen Facebook-Header, der dabei eingeblendet bleibt. Diese Unterschiede sind im Wesentlichen jedoch zu vernachlässigen – wichtig ist, dass bei allen mobilen Zugriffen auf die Anwendung der Inhalt von der Mobile Web URL bezogen wird.

7.2 Mobile Nutzung von Facebook-Dialogen

Da Sie in Ihrer mobilen Facebook-Anwendung wie gewohnt das JavaScript SDK geladen haben, können Sie sofort mit der Nutzung einiger Facebook-Dialoge, wie in Abschnitt 4.4, »Dialoge mit FB.ui«, beschrieben, beginnen. Das SKD erkennt dabei automatisch, dass es auf einem mobilen Endgerät geladen wurde, und passt die Dialoge dementsprechend an. Zum Testen der Dialoge am Smartphone können Sie einfach die JavaScript-Beispiele aus Abschnitt 4.4 in die Datei *mobile/index.php* einfügen:

```
<h1>Mobile Web-Apps auf Facebook</h1>
<p>
  <a href="#" onclick="fbInvite();">fbInvite</a> |
  <a href="#" onclick="fbLogin()">fbLogin</a> |
  <a href="#" onclick="fbUiDialog()">fbUiDialog</a> |
</p>

<script>
var fbLogin = function() {
   FB.login(function(response) {
   }, {scope: 'email'});
};
// Feed-Dialog
var fbUiDialog = function() {
   var options = {
       method: 'feed',
       link: 'http://www.orf.at',
       picture: 'http://orf.at/mojo/1_1/storyserver/...',
       name: 'ORF.at',
       caption: 'News-Portal des österreichischen Rundfunks'
   };
   FB.ui(options, function(result) {
```

```
   });
};
// Requests-Dialog
function fbInvite() {
  var options = {
       method: 'apprequests',
       message: 'Einladung zu meiner mobilen App',
   };
   FB.ui(options, function(result) {
   });
}
</script>
```

Listing 7.2 JavaScript-Code zum Testen der mobilen Dialoge

Rufen Sie nun Ihre mobile Anwendung erneut auf, und testen Sie die unterschiedlichen Dialoge! Vielleicht haben Sie bemerkt, dass Facebook die Dialoge beim ersten Aufruf jeweils in einem neuen Browser-Tab geladen hat – dabei kommt die bereits bekannte Methode zum Laden von Dialogen mittels Redirect auf eine URL der Form *https://www.facebook.com/dialog/DialogTyp* zum Einsatz. Ein Klick zum Bestätigen oder Abbrechen des Dialogs schließt das Tab und führt den Benutzer wieder zum Tab der eigentlichen Anwendung zurück. Dabei wird auch der optional definierte Callback-Handler der Funktion `FB.ui()` aufgerufen – damit können Sie wie gewohnt auf das Ergebnis des Dialogs reagieren.

Bei jedem weiteren Aufruf des Dialogs greift das JavaScript SDK auf eine gecachte Version des Dialogs zurück und verzichtet somit auf das Öffnen eines neuen Browser-Tabs – dies bewirkt, dass Dialoge auch am mobilen Endgerät sehr schnell geöffnet werden und so eine angenehme Benutzerführung möglich machen. Wenn Sie dieses Caching zu Testzwecken deaktivieren möchten, können Sie dies beim Aufruf von `FB.init()` mit der Option `useCachedDialogs:false` bewirken.

Hinweis: Die SEND- und Friends-Dialoge zum Versenden von privaten Nachrichten bzw. zum Hinzufügen von Freunden werden aktuell leider nicht auf mobilen Endgeräten unterstützt – ein entsprechender Aufruf wird vom JavaScript SDK mit einer Fehlermeldung quittiert.

Besondere Beachtung verdient der Requests-Dialog zum Einladen von Freunden zu einer Anwendung:

- Der `message`-Parameter darf zur mobilen Nutzung maximal 60 Zeichen lang sein.
- Über den Parameter `suggestions` kann `FB.ui()` ein Array an Benutzer-IDs übergeben werden – die Auswahlliste im Dialog wird dabei nach diesen Benutzern gefiltert. `suggestions` wird in der Desktop-Version des Dialogs ignoriert.

Abbildung 7.4 Die mobilen Dialoge zum Verfassen von Wall-Postings, zum Einladen von Freunden und zum Autorisieren der Anwendung

- Die Parameter `filters`, `exclude_ids`, `max_recipients` und `data` werden dafür noch nicht in der mobilen Version des Dialogs unterstützt.
- Eingehende Anwendungsanfragen werden auch in der mobilen Version von Facebook sowie in der nativen Applikation als Benachrichtigung im Header der Website/App angezeigt.
- Darüber hinaus werden Anwendungsanfragen bei Nutzern der aktuellen Facebook-Anwendung ab iOS 4.x *auch als Push-Notification* an das Smartphone verschickt.

Auch der Payment-Dialog zur Zahlung mit Facebook Credits steht auf mobilen Endgeräten zur Verfügung – allerdings nur, wenn die entsprechende Anwendung *nicht über die native iOS-Applikation* von Facebook aufgerufen wird! Der JavaScript-Code zum Aufruf des Dialogs kann dabei unverändert aus Abschnitt 4.4, »Dialoge mit FB.ui«, übernommen werden:

```
var order_info = 'produkt-1';
var obj = {
   method: 'pay',
   order_info: order_info,
   purchase_type: 'item',
   dev_purchase_params: {'oscif': false} };
FB.ui(obj, function (data) {
});
```

Listing 7.3 Aufruf des Payment-Dialogs

Hinweis: Der Dialog zur Anzeige der »Offer Wall« wird derzeit noch nicht auf mobilen Endgeräten unterstützt.

Abbildung 7.5 Zahlung mit Facebook Credits am Smartphone

7.3 Social Plugins im mobilen Web

Grundsätzlich funktionieren alle in Kapitel 6, »Social Plugins«, beschriebenen Social Plugins auch auf mobilen Endgeräten, sofern diese – wie alle aktuellen iOS- und Android-Modelle – ausreichende JavaScript-Unterstützung mitbringen.

Hinweis: Vergessen Sie bei der Nutzung von Social Plugins auf mobilen Webseiten nicht, dass zur Darstellung dieser Widgets mindestens ein zusätzlicher HTTP-Request durchgeführt werden muss. Da in mobilen Nutzungsszenarien oft nur eingeschränkte Bandbreite verfügbar ist, sollten Sie daher Social Plugins in jedem Fall sparsam einsetzen, also etwa höchstens ein oder zwei Plugins pro Webseite.

Abbildung 7.6 Recommendations- und Kommentarbox auf dem mobilen Endgerät

Plugins, die vor allem der Darstellung und Verlinkung von Content dienen, also etwa die Recommendations-Box oder der Activity Feed, können mit einer entsprechenden Angabe der maximalen Breite im Parameter `width` also auch in mobilen Webseiten eingesetzt werden. Besonders geeignet ist dabei aufgrund seiner geringen Dimension und flexiblen Konfigurationsmöglichkeiten der LIKE-Button. Weniger gut geeignet zur Darstellung auf mobilen Endgeräten sind hingegen interaktive Plugins wie die Registrierungsbox oder der Live Stream – von ihrer Nutzung auf mobilen Endgeräten sollten Sie lieber absehen.

Dies würde auch für die relativ komplexe Kommentarbox gelten, allerdings stellt Facebook für dieses Plugin seit Kurzem eine optimierte Darstellung für mobile Endgeräte bereit. Die mobile Version wird dabei automatisch angezeigt, wenn der Zugriff auf die integrierende Webseite über ein unterstütztes Smartphone erfolgt. Darüber hinaus haben Sie als Entwickler die Möglichkeit, über den Boolean-Parameter `mobile` explizit festzulegen, ob die mobile oder die Desktop-Variante angezeigt werden soll.

Kapitel 8
Open Graph

Ursprünglich wenig mehr als ein Set an Metadaten, das die Integration des »Like«-Buttons ermöglichte, bietet Open Graph heute flexible Möglichkeiten, um die eigene soziale Anwendung in Objekten, Aktionen und Aggregationsansichten zu modellieren und mit Facebook zu verknüpfen.

Das Konzept *Open Graph* existiert bereits seit 2010, als Facebook erstmals die Möglichkeit für Dritthersteller schuf, eigene, auf externen Webseiten gespeicherte Informationen und Objekte in den sozialen Graphen von Facebook zu importieren. Die damals noch *Open Graph Protocol* genannte Technologie wurde aus Sicht der Benutzer vor allem in Form des beliebten LIKE-Buttons wahrgenommen, der bald auf zahlreichen Blogs, Nachrichten- und Content-Websites integriert wurde. Im Herbst 2011 hat Facebook das dahinterliegende Konzept wesentlich über den Funktionsumfang des LIKE-Buttons hinaus erweitert und gleichzeitig in *Open Graph* umbenannt. Die nunmehr stark mit den gleichzeitig angekündigten Timeline-Benutzerprofilen integrierte Technologie ermöglicht Anwendungsentwicklern eine weitaus flexiblere Integration der eigenen Anwendungsdomäne in das Datenmodell von Facebook. Da das bisherige Open Graph Protocol dabei allerdings als Untermenge von *Open Graph* seine Relevanz behalten hat, ist der folgende erste Abschnitt dieses Kapitels diesem Protokoll gewidmet.

8.1 Open Graph Protocol

Ursprünglich wurde das Konzept von Open Graph 2010 unter dem Namen *Open Graph Protocol* veröffentlicht. Die Idee dahinter: Betreiber von Websites sollten eine einfache Möglichkeit erhalten, um ihre Inhalte in den sozialen Graphen von Facebook zu integrieren. Die Knoten des Graphen waren bis dahin ausschließlich Objekte, die auf der Facebook-Plattform abgebildet waren: Personen, Seiten bzw. Unternehmen, Marken, Organisationen, Orte. Die Kanten im Graphen bildeten die Beziehungen zwischen diesen Objekten ab: Freundschaftsbeziehungen zwischen Personen, Fan-Beziehungen zwischen Personen und Seiten etc.

Mit dem Open Graph Protocol konnte nun jede beliebige Webseite im Internet, die ein bestimmtes Objekt oder einen bestimmten Inhalt repräsentiert, ein Knoten im sozialen Graph werden. Zur besseren Integration können Betreiber von Websites ihre Inhalte dazu mit speziellen Metadaten, den Open Graph Tags, auszeichnen. Facebook nutzt diese Tags, um die auf der Webseite dargestellten Objekte oder Inhalte besser kategorisieren und interpretieren zu können. So werden etwa Angaben über den Titel des Objekts, den Objekttyp und die grafische Repräsentation des Objekts in den Open Graph Tags gespeichert.

Doch wie entsteht nun die Kante, die Beziehung zwischen diesen neuen, im Web beheimateten Knoten und Facebook-Benutzern? Die Antwort: Zeitgleich mit dem Open Graph Protocol hat Facebook den in Abschnitt 6.2, »Der ›Like‹-Button«, beschriebenen LIKE-Button und einige andere Social Plugins veröffentlicht. Diese Plugins wurden ursprünglich per iFrame, später per JavaScript in die eigene, externe Website eingebunden. Klickt ein bereits auf Facebook.com angemeldeter Benutzer auf den Button, wird automatisch die Verbindung zwischen Benutzer und Open-Graph-Objekt hergestellt und gespeichert. Zeitgleich wird dieser Vorgang mit einem entsprechenden Wall-Posting auf der Pinnwand des Benutzers veröffentlicht.

Open-Graph-Objekte

Die grundlegende Idee des Open Graph Protocols ist, dass unterschiedliche Instanzen von Objekten auf eigenen, unter eindeutiger URL adressierten Webseiten abgelegt werden. Vereinfacht gesagt: Jedes Objekt wird auf einer eigenen Webseite dargestellt.

Für einen Restaurant-Guide ist etwa »Lokalität« ein geeigneter Objekttyp, die einzelnen Instanzen wären jeweils auf eigenen Webseiten zu finden. Etwa der »Gasthof zur Goldenen Gans« unter *http://www.myguide.com/zur-goldenen-gans* oder das »Hotel Weiße Lilie« unter *http://www.myguide.om/hotel-weisse-lilie*.

Ein weiteres Beispiel lässt sich anhand der bekannten Internet Movie Database (IMDB) demonstrieren. Jeder in der Datenbank gespeicherte Film ist unter einer eigenen URL abrufbar, etwa »Real Steel« unter *http://www.imdb.com/title/tt0433035/* oder »The Social Network« unter *http://www.imdb.com/title/tt1285016/*.

Auf jeder einzelnen Webseite werden nun die Open Graph Tags des dargestellten Objekts gesetzt. Für »The Social Network« sieht dies etwa so aus:

```
<meta property="og:url" content="http://www.imdb.com/title/tt1285016/" />
<meta property="og:title" content="The Social Network (2010)"/>
<meta property="og:type" content="video.movie"/>
<meta property="og:image"
      content="http://ia.media-imdb.com/images/M/SY140_.jpg"/>
<meta property="og:site_name" content="IMDb"/>
```

Hinweis: Die Speicherung von Objekten im Open Graph ist von Facebook ausschließlich zur Abbildung von real existierenden Objekten gedacht – also etwa Personen, Unternehmen, Lokalitäten und vor allem auch Medieninhalte wie Filme, Bücher, Artikel und Songs.

Hinweis: Obwohl Facebook das Open Graph Protocol von Beginn an als offenen Webstandard publiziert hat, werden die zugrunde liegenden Standards für Metadaten bisher leider von keinem anderen bedeutenden Anbieter von Webservices oder Webapplikationen implementiert. Konkurrenten haben vielmehr eigene Alternativen zum Open Graph Protocol veröffentlicht, etwa Google und Bing mit Schema.org (*http://schema.org*). Bedenkt man Facebooks enormen Marktanteil in den sozialen Netzwerken und die weite Verbreitung des LIKE-Buttons, hat dies der Akzeptanz des Open Graph Protocols offenbar nicht sonderlich geschadet.

Wichtige Links:

- *http://developers.facebook.com/docs/opengraph/* – offizielle Dokumentation zum Open Graph Protocol
- *http://developers.facebook.com/tools/debug* – der URL Debugger oder *Linter* zum Testen von Open Graph Tags
- *http://ogp.me/* – Spezifikation des Open Graph Protocols als offener Webstandard

8.1.1 Zusammenspiel zwischen Open Graph Tags, »Like«-Button und Facebook

Im Folgenden wird der notwendige Ablauf beschrieben, der durchlaufen werden muss, um auf einer mit Open Graph Tags ausgezeichneten Webseite enthaltene Objekte in den sozialen Graphen von Facebook zu integrieren:

- Voraussetzung einer erfolgreichen Open-Graph-Integration ist, dass alle Webseiten der eigenen Site mit *passenden Open Graph Tags ausgestattet* werden. Jede Webseite sollte dabei genau ein Objekt (also ein Restaurant im Restaurant-Guide, einen Film in der Filmdatenbank etc.) repräsentieren. Die Tags können Sie als Entwickler mit dem Facebook URL Debugger überprüfen.
- Als Nächstes müssen Sie auf jeder Webseite den LIKE-Button integrieren – ob die Integration per iFrame, XFBML oder HTML5 gewählt wird, spielt dabei keine Rolle. Die einzelnen LIKE-Buttons werden mit dem `href`-Parameter an die URL des jeweiligen Objekts geknüpft. Damit sind die Vorbereitungen abgeschlossen – es besteht aber noch keine Verknüpfung zwischen den eigenen Objekten und dem sozialen Graphen von Facebook.

Abbildung 8.1 Integration des »Like«-Buttons auf der externen Webseite des Objekts

- Diese Verbindung wird hergestellt, sobald das erste Mal *ein Besucher auf einen Like-Button auf einer der Objektseiten klickt*. In diesem Moment wird Facebook auf das neue Objekt »aufmerksam« und schickt seinen Web Crawler aus, der die Open Graph Tags von der Webseite des Objekts liest. *Hinweis:* Die Inhalte der Open Graph Tags werden dabei bis zu 24 Stunden im Cache von Facebook gespeichert und erst bei späteren Klicks auf den LIKE-Button wieder aktualisiert. Zur Entwicklung leistet Ihnen daher der URL Debugger oder *URL Linter* wertvolle Dienste. Dieses Tool hilft beim Testen der eigenen Open Graph Tags und erzwingt ein erneutes Einlesen der Tags durch den Facebook-Crawler.
- Anschließend erzeugt Facebook ein neues *Knotenobjekt im sozialen Graphen* von Facebook.com. Das Objekt wird mit der Objekt-URL und den dort gefundenen Metadaten verknüpft.
- Das neu erzeugte Objekt wird nun durch eine »Like«-Beziehung mit dem Benutzer, der den LIKE-Button angeklickt hat, verbunden. Diese Beziehung kann später im Facebook-Profil des Benutzers angezeigt werden.
- Außerdem veröffentlicht Facebook *automatisch ein Posting auf der Pinnwand* des Benutzers, das anzeigt, dass der Benutzer den LIKE-Button dieses Objekts angeklickt hat. Dabei werden Titel, Beschreibung und Thumbnail aus den Open Graph Tags zur Darstellung des Wall-Postings herangezogen. Außerdem wird das Posting mit der externen URL des Objekts verlinkt.

Abbildung 8.2 Facebook erzeugt beim Klick auf den »Like«-Button automatisch ein Wall-Posting auf der Pinnwand des Benutzers.

Besonders bemerkenswert ist die Tatsache, dass Facebook für jedes externe Objekt, für das mindestens einmal der LIKE-Button geklickt wurde, eine interne Repräsenta-

tion im eigenen sozialen Graphen erstellt. Dieses interne Objekt entspricht dabei weitestgehend einem normalen Seitenprofil, wie es ansonsten für Fan-Pages verwendet wird. Wie später gezeigt wird, besitzen diese »virtuellen Seitenobjekte« sogar eine Pinnwand, die auf herkömmlichem Weg über das Benutzer-Interface von Facebook.com oder auch über die Graph API erreichbar ist.

8.1.2 Die Tags des Open Graph Protocols

Die folgende Tabelle gibt einen Überblick über alle Metadaten, die das Open Graph Protocol unterstützt. Als »Pflichtfeld« markierte Felder sollten Sie dabei angeben, weil Facebook andernfalls versucht, diese Daten durch die Inhalte herkömmlicher HTML-Tags zu ersetzen.

Tag	Beschreibung	Pflichtfeld
`og:title`	Titel des Objekts, das durch die Webseite repräsentiert wird. Der Titel sollte lediglich den gebräuchlichen Namen des Objekts enthalten und auf Zusätze wie eine nähere Beschreibung oder einen Website-Namen, wie sie aus SEO-Gründen gerne im `<title>`-Tag verwendet werden, verzichten. Wird `og:title` nicht angegeben, extrahiert Facebook den Objekttitel aus dem `<title>`-Tag.	ja
`og:description`	Ein- bis zweizeiliger Beschreibungstext des repräsentierten Objekts. Wird `og:description` nicht angegeben, extrahiert Facebook den Objekttitel aus dem `<meta name="description">`-Tag oder aus dem `<body>` der Seite.	nein
`og:type`	Typ des repräsentierten Objekts – eine Liste der unterstützten Objekttypen finden Sie weiter unten.	ja
`og:url`	URL des repräsentierten Objekts, also üblicherweise die URL der Webseite, auf der die Open Graph Tags eingebaut werden. Sie sollten für `og:url` die »kanonische« URL zu dem Objekt wählen, d. h., dynamische GET-Parameter und veränderliche URL-Bestandteile (etwa für die Sprache) sollten vermieden werden.	ja

Tabelle 8.1 Die verfügbaren Tags des Open-Graph-Protokolls

Tag	Beschreibung	Pflichtfeld
`og:locale`	Angabe über die Lokalisierung des repräsentierten Objekts, z. B. `de_DE`	ja
`og:site_name`	Name oder Bezeichnung der Website, auf der das repräsentierte Objekt zu finden ist	nein
`og:image`	URL zu einem Bild, das das repräsentierte Objekt zeigt. Das Bild muss mindestens 50 x 50 Pixel groß sein, im Format JPEG, PNG oder GIF vorliegen und ein maximales Seitenverhältnis von 3:1 aufweisen (besonders breite Bilder werden also ignoriert!). Geben Sie `og:image` nicht an, versucht Facebook automatisch, ein Bild aus dem Inhalt des HTML-Dokuments zu extrahieren – dies führt allerdings meist zu nicht sehr befriedigenden Ergebnissen.	ja
`fb:app_id`	Numerische ID einer Facebook-Anwendung, die zur Administration des Open-Graph-Objekts verwendet werden soll. Mit dieser Anwendung können Sie etwa auf die Pinnwand von Open Graph zugreifen.	nein
`fb:admins`	kommaseparierte Liste an Benutzer-IDs oder Benutzernamen, die das Open-Graph-Objekt administrieren dürfen	nein
`og:upc`	Ermöglicht die Angabe eines UPC-Codes (*Universal Product Code*). Dieses Feld ist ausschließlich für Objekttypen der Kategorie »Produkte & Unterhaltung« gedacht.	nein
`og:isbn`	Ermöglicht die Angabe eines ISBN-Codes (*International Standard Book Number*). Dieses Feld ist ausschließlich für Objekte vom Typ `book` gedacht.	nein

Tabelle 8.1 Die verfügbaren Tags des Open-Graph-Protokolls (Forts.)

Die im Tag `og:type` zulässigen Objekttypen werden in der folgenden Tabelle dargestellt. *Hinweis:* Mit der Erweiterung von Open Graph im Jahr 2012 fällt die Einschränkung auf vorgegebene Objekttypen gänzlich weg. Entwickler haben dann die Möglichkeit, eigene Objekttypen beliebig zu definieren.

Art	Mögliche Werte für og:type
Aktivitäten	activity, sport
Unternehmen	bar, company, cafe, hotel, restaurant
Gruppen	cause, sports_league, sports_team
Organisationen	band, government, non_profit, school, university
Personen	actor, athlete, author, director, musician, poltician, public_figure
Orte	city, country, landmark, state_province
Produkte & Unterhaltung	album, book, drink, food, game, product, song, movie, tv_show
Websites	blog, website, article

Tabelle 8.2 Zulässige Objekttypen im Open-Graph-Protokoll

Je nach Objekttyp werden Objekte, mit denen ein Benutzer durch den Klick auf den LIKE-Button verknüpft ist, auf dem Facebook-Profil des Benutzers angezeigt, etwa in den Abschnitten für MUSIK, FILME oder INTERESSEN.

Die Typen `website` und `blog` sind zur Identifikation ganzer Websites gedacht und sollten damit nur auf der Homepage einer Website eingesetzt werden. `article` dient hingegen der Auszeichnung einzelner Artikel, Unterseiten etc. etwa auf Blogs.

Für Objekte, die einen Ort oder ein Unternehmen repräsentieren, können Sie in den folgenden Open Graph Tags zusätzliche Informationen speichern:

Tag	Beschreibung	Pflichtfeld
og:latitude	Breitengrad	nein
og:longitude	Längengrad	nein
og:street-adress	Adresse	nein
og:locality	Stadt	nein
og:region	Region/Bundesland	nein
og:postal-code	Postleitzahl	nein
og:country-name	Land	nein

Tabelle 8.3 Open Graph Tags für Unternehmen und Orte

Tag	Beschreibung	Pflichtfeld
og:email	E-Mail-Adresse	nein
og:phone_number	Telefonnummer	nein
og:fax_number	Faxnummer	nein

Tabelle 8.3 Open Graph Tags für Unternehmen und Orte (Forts.)

Das folgende Code-Beispiel zeigt vollständige Metadaten für das »Restaurant Zur Goldenen Gans« einer fiktiven Gastronomie-Website »MyGuide.com«:

```
<head>
   <meta charset="utf-8"/>
   <title>MyGuide.com - Gasthof zur Goldenen Gans</title>
   <meta property="og:url"
         content="http://www.myguide.com/zur-goldenen-gans" />
   <meta property="og:title" content="Gasthof zur Goldenen Gans"/>
   <meta property="og:description" content=
"Im Gasthof zur Goldenen Gans erwarten Sie ganztägig warme Küche und beste Haus-
mannskost!"/>
   <meta property="og:type" content="restaurant"/>
   <meta property="og:image"
         content="http://myguide.com/img/goldene-gans-logo.jpg"/>
   <meta property="og:site_name" content="MyGuide.com"/>
   <meta property="fb:app_id" content="214728715257742"/>
   <meta property="fb:admins" content="michael.kamleitner"/>
   <meta property="og:latitude" content="33.29802,56.601563"/>
   <meta property="og:longitude" content="40.756896"/>
   <meta property="og:street-address"
         content="Planegger Straße 31"/>
   <meta property="og:locality" content="München"/>
   <meta property="og:region" content="Bayern"/>
   <meta property="og:postal-code" content="81241"/>
   <meta property="og:country-name" content="Deutschland"/>
</head>
```

Listing 8.1 Vollständige Open Graph Tags für ein Restaurant-Objekt

8.1.3 Einbindung von Video-, Flash- und Audiodaten

Neben der Unterstützung von Thumbnail-Grafiken zur grafischen Repräsentation des abgebildeten Objekts unterstützt Facebook mit dem Open Graph Protocol auch

die Einbindung von multimedialen Audio-, Flash- und Videodaten. Diese werden in den durch den LIKE-Button erzeugten Wall-Postings als direkt abspielbare Inhalte auf der Pinnwand des Benutzers angezeigt und erhöhen somit die Klickraten von anderen Benutzern.

Video- und Flash-Daten können Sie mit folgenden Tags einbinden:

Tag	Beschreibung	Pflichtfeld
`og:video`	URL zum Video- oder Flash-Content	ja
`og:video:secure_url`	sichere URL zum Video- oder Flash-Content mit HTTPS-Protokoll – für Benutzer, die Browsen per HTTPS aktiviert haben. Alternativ kann auch in `og:video` bereits eine HTTPS-URL angegeben werden.	nein
`og:video:type`	MIME-Typ, z. B. `application/x-shockwave-flash` für Flash-Content	ja
`og:video:width`	Breite des Flash- oder Video-Contents in Pixeln	nein
`og:video:height`	Höhe des Flash- oder Video-Contents in Pixeln	nein

Tabelle 8.4 Open Graph Tags zur Einbindung von Flash- und Video-Content

Das folgende Beispiel bindet einen YouTube-Clip als Flash-Video in die Open Graph Tags ein:

```
<meta property="og:video:type"
      content="application/x-shockwave-flash"/>
<meta property="og:video"
      content="https://www.youtube.com/v/1G4isv_Fylg?version=3"/>
<meta property="og:video:width" content="300"/>
<meta property="og:video:height" content="180"/>
```

Listing 8.2 Einbindung eines YouTube-Clips in die Open Graph Tags

Das beim Klick auf den LIKE-Button erzeugte Wall-Posting enthält das direkt abspielbare YouTube-Video:

Abbildung 8.3 Wall-Posting mit direkt abspielbarem Flash-Content

Neben Flash-Content können auch Videos im MP4-Format eingebunden werden. Dazu müssen Sie den `og:video:type` auf den MIME-Typ `video/mp4` setzen. Dabei ist auch eine Angabe von mehreren Videos in einem Open-Graph-Objekt zulässig. Facebook entscheidet bei der Anzeige des Wall-Postings dann je nach Endgerät, welches Content-Format angezeigt wird. Während auf Desktop-Browsern grundsätzlich dem Flash-Content der Vorzug gegeben wird, werden auf iOS-Geräten wie dem iPhone oder iPad die MP4-Inhalte angezeigt.

Audiodaten können derzeit ausschließlich in Form von MP3-Dateien eingebunden werden, wozu Ihnen folgendes Set an Meta-Tags zur Verfügung steht:

Tag	Beschreibung	Pflichtfeld
`og:audio`	URL zum Audio-Content	ja
`og:audio:title`	Titel des Audio-Contents	nein
`og:audio:artist`	Interpret/Künstler	nein
`og:audio:album`	Album	nein
`og:audio:type`	MIME-Typ des Audio-Contents, derzeit immer `application/mp3`	ja

Tabelle 8.5 Open Graph Tags zur Einbindung von Audiodaten

Das folgende Beispiel zeigt die Einbindung eines MP3-Clips in das Open-Graph-Objekt:

```
<meta property="og:audio"
      content="http://mycompany.com/clip.mp3" />
<meta property="og:audio:title" content="Song-Titel" />
```

```
<meta property="og:audio:artist" content="Künstler" />
<meta property="og:audio:album" content="Album-Titel" />
<meta property="og:audio:type" content="application/mp3" />
```

Listing 8.3 Einbindung eines Audio-Clips in die Open Graph Tags

Und so wird der MP3-Clip im Wall-Posting angezeigt – auch hier können Benutzer den Audio-Clip direkt in der Pinnwand abspielen, wobei der hierbei verwendete Flash-basierte Player automatisch von Facebook bereitgestellt wird:

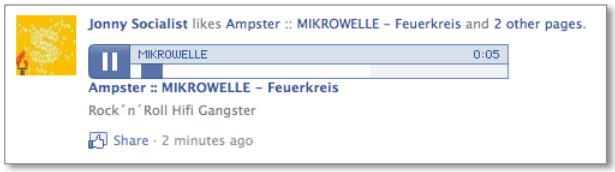


Abbildung 8.4 Wall-Posting mit direkt abspielbarem Audio-Content

8.1.4 Testen von Open Graph Tags mit dem URL Debugger

Wie bereits erwähnt, speichert Facebook die Inhalte von Open Graph Tags bis zu 24 Stunden in einem eigenen Cache zwischen. Änderungen an der Implementierung von Open Graph Tags wären damit schwierig zu testen – gäbe es nicht ein von Facebook bereitgestelltes Tool – den URL Debugger (oder *Linter*, wie er früher auch oft genannt wurde). Der URL Debugger (*http://developers.facebook.com/tools/debug)* liest die Open Graph Tags einer beliebigen Webseite ein und stellt die darin gefundenen Inhalte so dar, wie sie später auch von Facebook interpretiert werden. Dabei werden auch hilfreiche Angaben zu etwaigen Problemen und fehlenden Tags gemacht. Angenehmer Nebeneffekt: Jeder Aufruf des URL Debuggers setzt den Cache für die abgefragte Webseite zurück! Damit ist der URL Debugger ein unverzichtbares Tool für Ihre Arbeit mit dem Open Graph Protocol!

Sie können den URL Debugger auch benutzen, um den Cache der Open Graph Tags für eine größere Anzahl an Webseiten zurückzusetzen. Dazu genügt ein PHP-Skript, das die einzelnen Seiten etwa mittels `curl()` durch den Debugger schickt. Eine Möglichkeit, den Cache für alle Webseiten einer Domain auf einen Streich zu löschen, gibt es hingegen leider nicht.

```
<?
include("tools.php");
curl("http://developers.facebook.com/tools/lint/?url=".
     "http://myguide.com/seite-1");
curl("http://developers.facebook.com/tools/lint/?url=".
     "http://myguide.com/seite-1");
```

```
curl("http://developers.facebook.com/tools/lint/?url=".
     "http://myguide.com/seite-1");
...
?>
```

Listing 8.4 Löschen des Caches für mehrere Open-Graph-Objekte

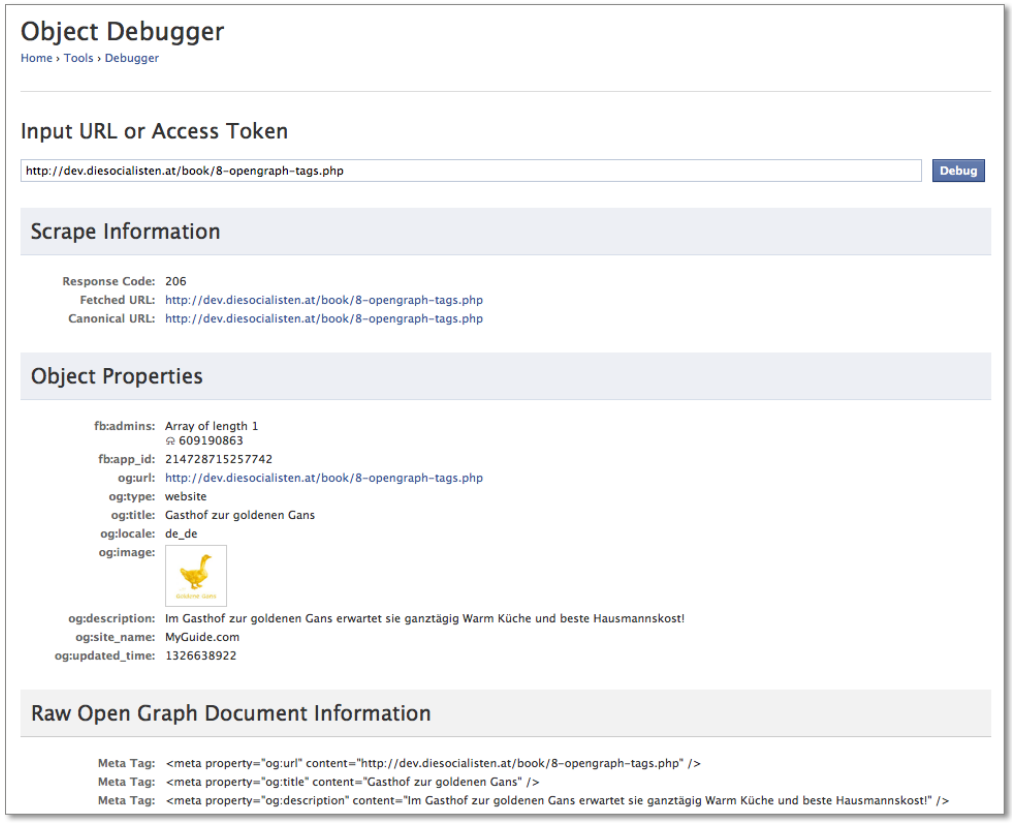


Abbildung 8.5 Der URL Debugger prüft die Open Graph Tags einer beliebigen Webseite.

8.1.5 Administrieren und Veröffentlichen auf Open-Graph-Seiten

Wie eingangs beschrieben, legt Facebook für jede mit Open Graph Tags versehene Seite beim ersten Klick auf den LIKE-Button ein internes Objekt im sozialen Graphen an. Dieses Objekt entspricht in weiten Teilen einem Seitenobjekt vom Typ `page` und besitzt wie diese auch eine eigene Pinnwand. Benutzern, die mittels `fb:admins` als Administratoren des Open-Graph-Objekts ausgewiesen wurden oder die eingetragene Entwickler der in `fb:app_id` angegebenen Anwendung sind, zeigt Facebook neben dem LIKE-Button einen speziellen Link ADMIN PAGE an:

Abbildung 8.6 Der »Admin Page«-Link (hervorgehoben) für Administratoren des Open-Graph-Objekts

Mit einem Klick auf ADMIN PAGE öffnen Sie die »virtuelle« Administrationsseite des Open-Graph-Objekts auf Facebook.com:

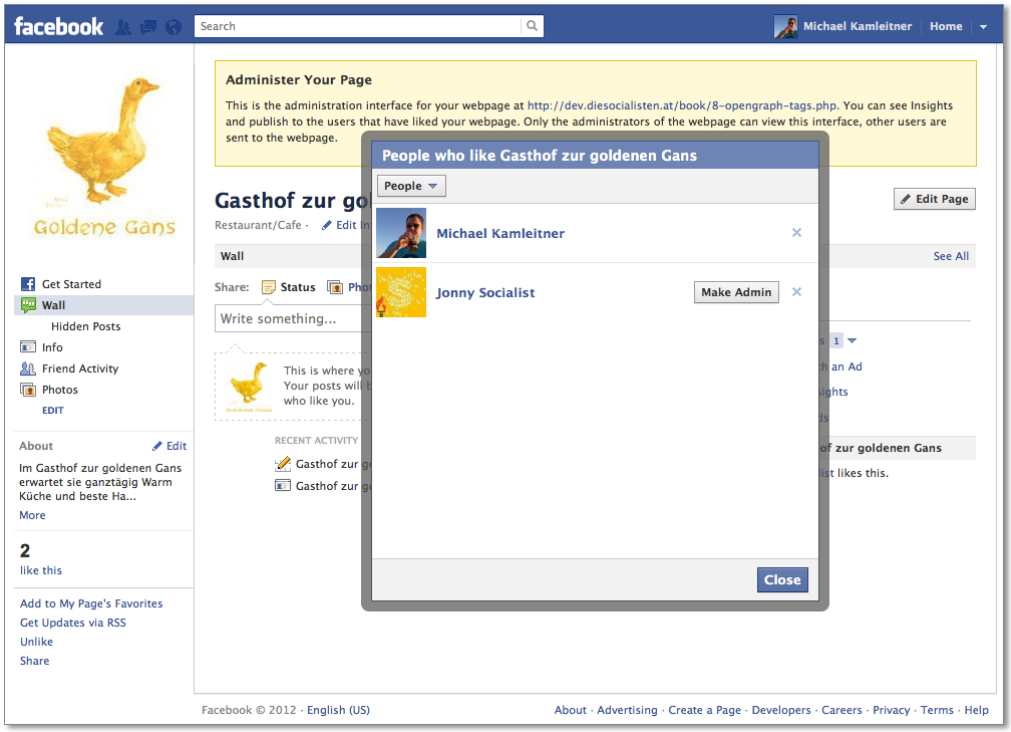


Abbildung 8.7 Die Administrationsseite des Seitenobjekts

Wichtig: Diese »virtuelle« Seite ist ausschließlich für Administratoren des Open-Graph-Objekts sichtbar! Besucht ein normaler Benutzer die URL dieser Seite (z. B. *http://www.facebook.com/pages/Gasthof-zur-goldenen-Gans/304059166297447*), wird er automatisch auf die Webseite des Objekts weitergeleitet, also jene URL, die im Open Graph Tag `og:url` angegeben wurde! Diese Tatsache ist zu Beginn oft verwirrend, obwohl Facebook einen entsprechenden Hinweis einblendet:

Abbildung 8.8 Hinweis auf die Sichtbarkeit der virtuellen Administrationsseite

Aus Administratorensicht gleicht die Administrationsseite weitestgehend einem herkömmlichen Seitenprofil – so können Sie etwa jene Personen, die beim entsprechenden Open-Graph-Objekt auf den LIKE-Button geklickt haben, auflisten (Klick auf LIKE THIS) oder über die Pinnwand Wall-Postings veröffentlichen. Solche Postings erscheinen bei all jenen Benutzern im Newsfeed, die »Fan« des Open-Graph-Objekts geworden sind (dabei muss man allerdings erwähnen, dass Facebook die Relevanz und damit die Sichtbarkeit solcher Wall-Postings im Vergleich zu Postings von herkömmlichen Seiten einschränkt). Dennoch: Bei Open-Graph-Objekten mit einer großen Anzahl an verknüpften Benutzern können Sie über die Administrationsseite einen wertvollen Rückkanal zu den Fans eröffnen!

Dieser Rückkanal kann auch über die Graph API genutzt werden. Dazu ist es notwendig, dass Sie zuvor über das Tag `fb:app_id` eine bestimmte Facebook-Anwendung mit den Open-Graph-Objekten verknüpft haben. Diese Anwendung kann nun mit einem Anwendungs-Access-Token Wall-Postings auf der Pinnwand veröffentlichen, wie das folgende Code-Beispiel zeigt:

```
<?
include_once("tools.php");
define('APP_ID', '214728715257742');
define('APP_SECRET', 'b93bff303c6199d9...');
define('SITE_URL', 'http://myguide.com');

$app_access_token = get_app_accesstoken(APP_ID, APP_SECRET);
$url = "https://graph.facebook.com/feed";
$params = array(
   "access_token" => $app_access_token,
   "message" => "Wall-Posting auf der Objektseite",
   "id" => "http://www.myguide.com/zur-goldenen-gans"
);
$result = curl ($url, $params, "POST");
?>
```

Listing 8.5 Veröffentlichen von Wall-Postings auf der Administrationsseite

Beim Veröffentlichen von Wall-Postings mittels Graph API müssen Sie Folgendes beachten:

- Mit der Hilfsfunktion `get_app_accesstoken()` ermitteln Sie zuerst das Anwendungs-Access-Token.
- Der API-Endpunkt zum Veröffentlichen lautet immer `https://graph.facebook.com/feed`.
- Im Parameter `message` übergeben Sie analog zu herkömmlichen Wall-Postings die Nachricht – andere Parameter für `link`, `picture`, `caption` etc. können Sie ebenfalls wie gewohnt verwenden.
- Das Open-Graph-Objekt, auf dessen Administrationsseite veröffentlicht werden soll, definieren Sie im Parameter `id` – dazu geben Sie einfach die URL der Objektseite, wie im Parameter `og:url` angegeben, an.

Anstatt in `id` die vollständige URL zum Objekt anzugeben, können Sie auch die numerische Objekt-ID des Open-Graph-Objekts verwenden. Doch wie kann diese ermittelt werden? Dazu genügt es, einen GET-Aufruf des API-Endpunktes nach folgendem Schema durchzuführen:

```
https://graph.facebook.com?id=OpenGraphObjektUrl
```

Also z. B.:

```
https://graph.facebook.com?id=http://www.myguide.com/zur-goldenen-gans
```

Das zurückgelieferte JSON-Objekt gibt Auskunft über die interne Speicherung des Open-Graph-Objekts im sozialen Graphen von Facebook, wobei das Feld `id` die interne numerische ID des Objekts enthält:

```
{
  "id": "303072659739080",
  "name": "Gasthof zur Goldenen Gans",
  "picture": "http://profile.ak.fbcdn.net/hprofile-ak-snc4/373478_
  303072659739080_298557113_s.jpg",
  "link": "http://www.myguide.com/zur-goldenen-gans",
  "likes": 1,
  "app_id": 214728715257742,
  "category": "Restaurant/cafe",
  "description": "Im Gasthof zur Goldenen Gans erwarten Sie ganztägig warme
  Küche und beste Hausmannskost!",
  "can_post": true,
  "type": "page"
}
```

Listing 8.6 JSON-Objekt mit der internen Repräsentation eines Open-Graph-Objekts

8.2 Open Graph

Auf der Entwicklerkonferenz f8 im September 2011 hat Facebook das Konzept des Open Graph Protocols gänzlich neu definiert. Bis dahin war das Protokoll vor allem darauf ausgerichtet, bestimmte Objekttypen mittels LIKE-Button in den sozialen Graphen von Facebook aufzunehmen. Die dabei unterstützten Objekte mussten in ein bestimmtes, von Facebook vorgegebenes Schema passen. Dieses Schema war vor allem auf Unternehmen und Organisationen, Personen und Medieninhalte (Artikel, Musik, Video etc.) ausgerichtet. Nur Objekte, die diesem Schema entsprachen, konnten sinnvoll als Knoten Teil des sozialen Graphen von Facebook werden. Noch weniger Optionen hatten Entwickler bei der Wahl der Kanten (*Edges*), die Open-Graph-Objekte mit Benutzern verknüpften: Mit dem LIKE-Button stand lediglich ein Kantentyp zur Verfügung. Eine »Like«-Kante hat dabei lediglich die Bedeutung, dass ein Benutzer mittels Klick auf GEFÄLLT MIR in eine positive, aber nicht näher spezifizierbare Beziehung zu dem Objekt getreten ist.

Mit dem neuen Open Graph beseitigt Facebook diese Einschränkungen nun grundlegend! Open Graph gibt Entwicklern erstmals die Möglichkeit, beliebige Objekttypen abseits der bekannten Typen zu definieren und damit jeden beliebigen Inhalt der eigenen Anwendung im sozialen Graphen zu repräsentieren. Der Objekttyp wird dabei ähnlich einer Datenbanktabelle in Eigenschaften bzw. Feldern und Verknüpfungen zu anderen Objekttypen modelliert. Dabei werden die bereits bekannten Open Graph Tags weiterverwendet und um anwendungsspezifische Tags ergänzt.

Doch damit nicht genug! Genauso wie Objekttypen können Entwickler Open Graph nutzen, um beliebige Aktionen als Kantentyp im sozialen Graphen darzustellen. Ein Benutzer kann also ein Objekt nicht nur mittels GEFÄLLT MIR positiv auszeichnen, sondern in beliebigen »Aktionen« mit dem Objekt interagieren. So können Benutzer ein Musikstück vom Typ »Song« etwa mit der Aktion »Listen« anhören, ein Objekt vom Typ »Kochrezept« mit der Aktion »Kochen« zubereiten oder sich einen »Artikel« auf einer Nachrichten-Website mit der Aktion »Lesen« durchlesen. Anders ausgedrückt, bietet Ihnen Open Graph die Möglichkeit, nahezu beliebige Aktionen, die der Benutzer mit einem Objekt in Ihrer Anwendung durchführt, auch als Aktion in Facebook abzubilden.

Hinweis: Individuelle Aktionen von Anwendungen müssen im Rahmen eines Prüfungsprozesses (*Approval Process*) von Facebook geprüft und explizit freigeschaltet werden. Diese Prüfung soll dem Missbrauch des neuen Open Graph, etwa durch irreleitende Aktionsbezeichnungen oder unverhältnismäßig zahlreiche Veröffentlichungen, entgegenwirken.

8.2.1 Ticker und Timeline-Profil

Gemeinsam mit Open Graph hat Facebook zwei Neuerungen im Benutzer-Interface von Facebook.com angekündigt, die in starkem Zusammenhang mit den neuen Möglichkeiten zur Modellierung von Objekten und Aktionen stehen:

- Der *Ticker* ist ein neuer, echtzeitbasierter Nachrichten-Feed, der am rechten Rand des Facebook-Interfaces angezeigt wird. Er soll den Newsfeed (die individuelle Startseite jedes Facebook-Benutzers) gleichzeitig entlasten und ergänzen. Einerseits werden elementare Aktionen wie ein »Like« oder ein Kommentar künftig in den Ticker verlagert, andererseits dient der Ticker primär auch zur Anzeige aller Aktionen, die über Open Graph veröffentlicht werden. Im Ticker laufen diese Aktionen in Echtzeit an den Augen des Benutzers vorbei, während Einträgen, denen Facebook mehr Relevanz zuordnet, wie etwa von Freunden hochgeladene Fotos, Status-Updates oder geteilte Inhalte mit mehreren Kommentaren, der klassische Newsfeed vorbehalten bleibt. Zeigt der Benutzer mit der Maus auf einen Ticker-Eintrag, blendet Facebook in einem aufklappenden Fenster weitere Informationen zum Eintrag ein, etwa in Bezug auf die Aktion veröffentlichte Kommentare oder die Thumbnail-Grafik des verknüpften Objekts oder Inhalts. Open-Graph-Aktionen werden darüber hinaus mit einem kleinen Button ergänzt, der den betrachtenden Benutzer dazu einlädt, mit dem geteilten Open-Graph-Objekt in Interaktion zu treten. Hört z. B. ein Freund gerade einen Song mit dem Open-Graph-fähigen Musikservice Spotify, genügt ein Klick auf den PLAY-Button, um den gleichen Song selbst abzuspielen.
- Die weitaus auffallendste Neuerung der f8 betrifft die Benutzerprofile. Unter dem Namen *Timeline* (im Deutschen *Chronik*) hat Facebook die Profile der Benutzer grundlegend umgestaltet – anstelle der alten Pinnwand werden Wall-Postings, Status-Updates, Fotos, geteilte Inhalte aus dem Web und mehr nun in einer zweispaltigen, chronologischen Ansicht dargestellt. Die Grundidee ist dabei, dass Benutzer die Timeline nutzen können, um ihr Leben – online wie offline – umfassend und in grafisch anspruchsvoller Art und Weise darzustellen. Ein besonderes Gimmick ist dabei neben dem großformatigen Cover-Foto ein *Infinite Scrolling* genannter Mechanismus, der – gelangt ein Benutzer beim Betrachten des Profils an den unteren Bildschirmrand – dafür sorgt, dass automatisch älterer Content des Profils nachgeladen wird. So kann die »Chronik« eines Benutzers auf Facebook vom aktuellsten Ereignis bis zurück zur Geburt (oder zumindest zum Zeitpunkt des ersten Postings) der Person zurückverfolgt werden. Noch ist unklar, wie Timeline von den Facebook-Benutzern aufgenommen werden wird – es steht aber fest, dass Facebook mit Timeline die gravierendste Änderung an der eigenen Plattform seit Einführung des Newsfeeds vorgenommen hat.

Abbildung 8.9 Der neue Echtzeit-Ticker stellt elementare Interaktionen und veröffentlichte Open-Graph-Aktionen dar.

8.2.2 Das Timeline-Profil im Überblick

Neben der augenscheinlichen optischen Umgestaltung von Timeline bringen die neuen Facebook-Benutzerprofile auch viele Änderungen mit sich, die sich erst auf den zweiten Blick erschließen.

Die Timeline ist im Wesentlichen ein chronologischer Nachrichtenstrom – ähnlich der bisherigen Pinnwand, nun aber in zweispaltigem Layout. Die neuesten Postings werden dabei wie gewohnt zuoberst dargestellt. Der chronologische Charakter wird durch einen zentriert dargestellten »Zeitstrahl« repräsentiert, an dem die Einträge als Sprechblasen mit Knotenpunkten andocken. Eine Datumsselektion am rechten oberen Rand erlaubt es Betrachtern, die Timeline auf bestimmte Zeiträume einzuschränken.

Neben Status-Updates, Fotos, Videos und sonstigen Inhalten, die direkt auf Facebook.com erzeugt werden, können Timeline-Einträge auch individuelle Aktionen enthalten, die über Open-Graph-fähige Anwendungen veröffentlicht wurden. Am oberen Rand der rechten Spalte finden sich Boxen, in denen die letzten Aktionen in häufig benutzten Open-Graph-Anwendungen des Benutzers zusammengefasst werden.

Hinweis: Facebook gibt dem Benutzer derzeit keine Kontrolle darüber, welche Anwendungen hier dargestellt werden, sondern bestimmt dies anhand des Nutzungsverhaltens des Benutzers.

Neben den chronologischen Einträgen wird die Timeline aber auch immer wieder durch besondere Boxen unterbrochen, die Content und Open-Graph-Interaktionen des Benutzers über einen bestimmten Zeitraum aggregiert darstellen. Anwendungsentwickler definieren dabei, welche Daten der Anwendung in diesen sogenannten *Aggregationen* dargestellt werden sollen. Für eine Musikapplikation könnten dies z. B. die vom Benutzer am häufigsten gehörten Songs sein, eine geobasierte Checkin-Applikation wie Foursquare könnte die zuletzt besuchten Orte eines Benutzers darstellen.

Hinweis: Während die Daten und Inhalte von Aggregationen vom Anwendungsentwickler weitestgehend flexibel definiert und gruppiert werden können, werden das Layout und die grafische Darstellung allein von Facebook bestimmt.

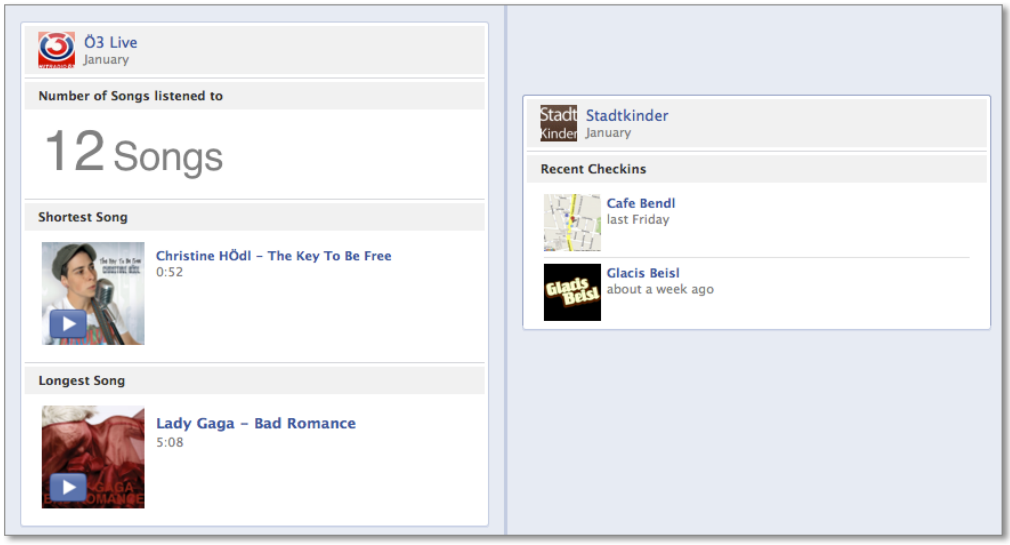


Abbildung 8.10 Aggregationen von zwei Open-Graph-Anwendungen

Unter dem großformatigen Cover-Foto der Timeline finden sich vier plus acht – durch Klick auf den Pfeil ausklappbare – Schaltflächen, die sogenannte *Timeline-Views* öffnen. Timeline-Views sind anwendungsbezogene Ausschnitte der Timeline eines Benutzers, zeigen also nur Einträge an, die über eine bestimmte Open-Graph-Anwendung veröffentlicht worden sind.

Auch Timeline-Views sind chronologisch sortiert und lassen sich einfach auf bestimmte Zeiträume einschränken. Ganz oben werden dabei im Abschnitt ALL TIME Aggregationen über den gesamten Nutzungszeitraum einer Anwendung angezeigt

(in einer Musikanwendung also z. B. die beliebtesten Songs oder Interpreten des Benutzers). Benutzer können beliebig festlegen, welche Anwendungen in den vier plus acht Schaltflächen abgebildet werden.

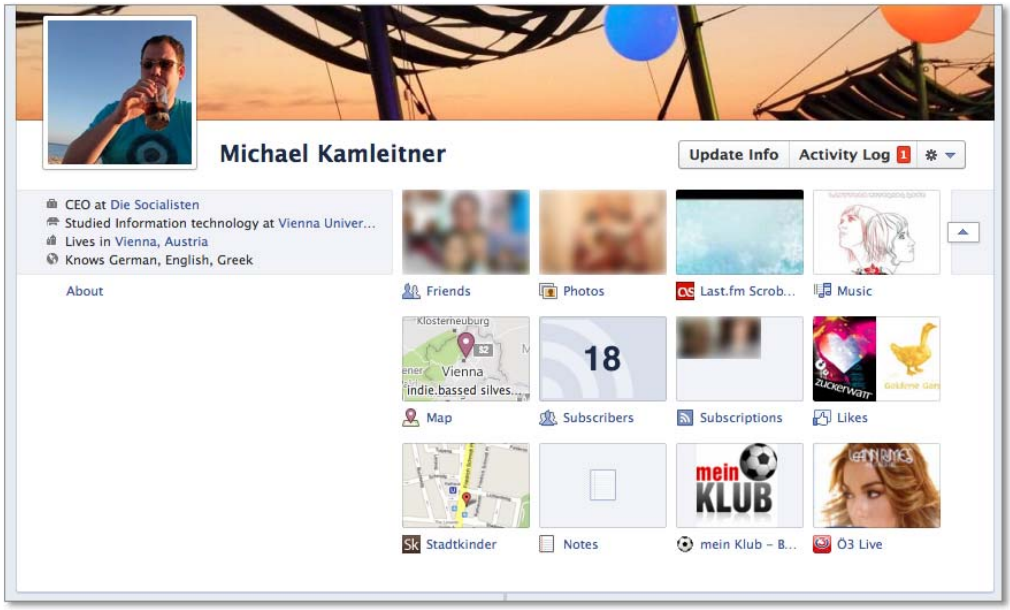


Abbildung 8.11 Die acht Schaltflächen führen zu den Timeline-Views für verschiedene Anwendungen.

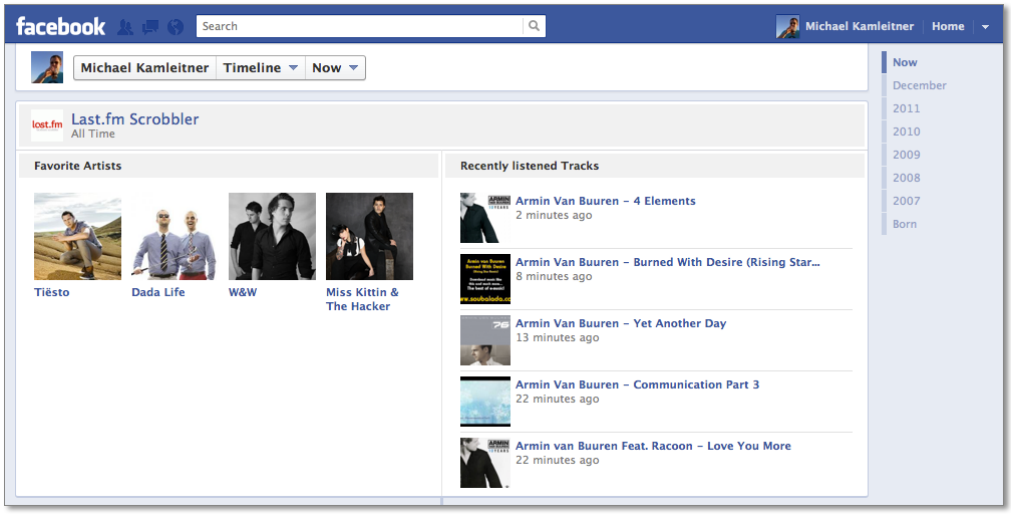


Abbildung 8.12 Die Timeline-View einer Musikanwendung filtert die Timeline nach Einträgen dieser Applikation. Ganz oben: der Bereich »All time«.

Ein weiteres neues Element des Timeline-Profils ist das *Activity Log* – dieses zeigt eine gemeinsame Listendarstellung aller über Open-Graph-Anwendungen veröffentlichten Aktionen sowie aller Facebook-eigenen Aktivitäten an – also etwa Kommentare, per GEFÄLLT MIR geteilte Links und neue Freundschaften. Hier können Benutzer auf einfache Weise veröffentlichte Aktionen auffinden und wieder von ihrem Timeline-Profil entfernen. *Wichtig:* Das Activity Log ist immer nur für das eigene Benutzerkonto sichtbar!

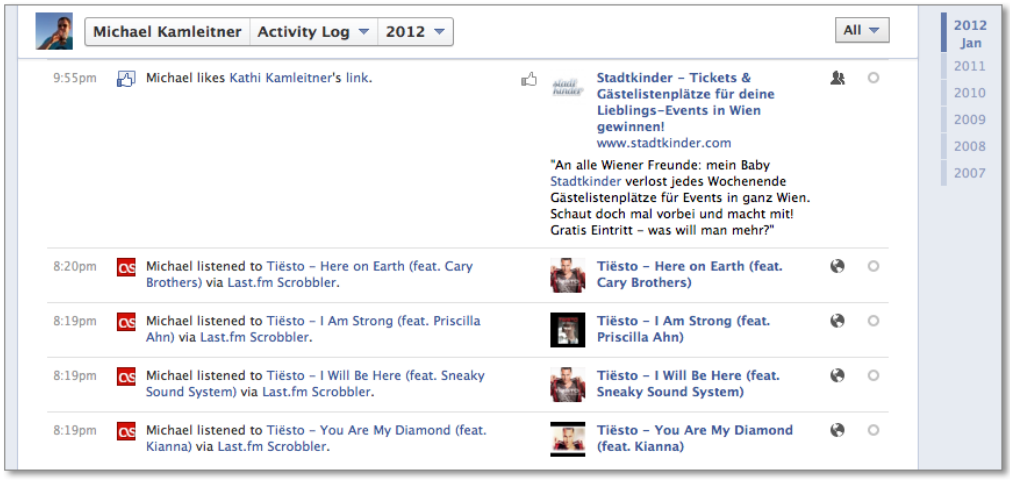


Abbildung 8.13 Das Activity Log eines Benutzers

Nachdem Anwendungsentwickler die neue Timeline schon zur f8 manuell aktivieren konnten, hat Facebook schließlich im Dezember 2011 mit dem weltweiten Roll-out der Timeline begonnen. Während der Arbeit an diesem Buch war die Timeline aber noch nicht flächendeckend verfügbar. Facebook hat jedenfalls angekündigt, dass die Umstellung auf Timeline-Profile letztlich für alle Benutzer verpflichtend ist. Obwohl dies wie bei den meisten größeren Änderungen an der Plattform kontrovers diskutiert werden wird, darf man davon ausgehen, dass die Timeline in den nächsten Jahren das Aussehen und die Funktonalität von Facebook wesentlich bestimmen wird.

Die Timeline wird nach der weltweiten Veröffentlichung auch auf mobilen Endgeräten zur Verfügung stehen.

Wichtige Links:

- *https://www.facebook.com/about/timeline* – Die offizielle Vorstellung der neuen Timeline gibt einen guten Überblick über die neuen Benutzerprofile.
- *http://techcrunch.com/2011/09/22/how-to-enable-facebook-timeline/* – How-to, das zeigt, wie die Timeline vor dem offiziellen Roll-out aktiviert werden kann

Abbildung 8.14 Timeline – das neue Facebook-Benutzerprofil. Hervorgehoben sind die neuen Integrationsmöglichkeiten für Open-Graph-Anwendungen: Reports, Aggregationen und Timeline-Views.

Zur f8 hat Facebook gemeinsam mit ausgewählten Partnern einige exemplarische Open-Graph-Anwendungen präsentiert. Diese Showcases vermitteln einen guten Eindruck der neuen Möglichkeiten. Meine Empfehlung: Verschaffen Sie sich anhand dieser Best-Practice-Beispiele einen Überblick über die Möglichkeiten von Open Graph, bevor Sie mit der Konzeption und Entwicklung eigener Anwendungen beginnen!

- Der On-Demand-Musikservice Spotify (*http://www.spotify.com/*) nutzt den Open Graph, um alle über den Service konsumierten Musikstücke als Open-Graph-Aktion zu veröffentlichen. Damit wird es für Benutzer einfacher, neue Musik von Freunden zu entdecken oder gar gemeinsam Musik zu hören. Die eigenen musikalischen Vorlieben werden dabei aggregiert am Timeline-Profil der Benutzer veröffentlicht. *Hinweis:* Spotify ist außerhalb der USA bisher nur in wenigen Ländern verfügbar, anders als Österreich und die Schweiz gehört Deutschland nicht zu diesen Ländern.
- Mit der Canvas-Applikation Washington Post Social Reader (*http://apps.facebook.com/wpsocialreader/*) können Online-Leser der Zeitung gelesene Artikel in Ticker und Timeline veröffentlichen. Auch hier sollen Benutzern neue Möglichkeiten zur gemeinsamen Nutzung und zum Entdecken von Content geboten werden.
- Der amerikanische DVD-Miet- und On-Demand-Service Netflix (*http://apps.facebook.com/netflix_social/*) veröffentlicht Videos und Filme, die Benutzer sich anschauen, als Open-Graph-Aktionen und funktioniert damit ähnlich wie Spotify. Netflix ist nur in den USA verfügbar.

Hinweis: Der neue Open Graph wird während der Entstehung dieses Buches von Facebook noch im Beta-Status betrieben. Mit Ausnahme privilegierter Partnerunternehmen (wie etwa Spotify, der Washington Post oder Netflix) sind damit alle auf Open Graph basierenden Anwendungen derzeit nur für die als Entwickler oder Tester der Anwendung eingetragenen Benutzer verfügbar. Da vor allem alle anwendungsspezifischen Open-Graph-Aktionen nach individueller Prüfung von Facebook freigeschaltet werden müssen, sind Aktionen von Anwendungen im Sandbox-Modus auch nach der Veröffentlichung auf dem Timeline-Profil eines Benutzers nur für Entwickler und Tester sichtbar. Der öffentliche Launch von Open Graph, das Freischalten von Open-Graph-Aktionen und damit eine flächendeckende Verfügbarkeit von Open-Graph-Anwendungen ist von Facebook ohne Angabe eines genauen Datums für Anfang 2012 angekündigt. Indirekt hat dies auch mit dem Ausrollen der neuen Timeline-Profile zu tun, das im Dezember 2011 bereits in einigen Ländern angelaufen ist.

8.2.3 Definieren von Objekten

Als *Objekte* werden im Open Graph all jene Dinge bezeichnet, mit denen ein Benutzer durch eine Interaktion in Verbindung treten kann. Die Modellierung der eigenen

Objekte stellt den ersten Schritt zur Erstellung einer eigenen Open-Graph-Anwendung dar und sollte daher zu Beginn erfolgen.

Die neuen Open-Graph-Funktionen basieren auf einer Erweiterung des bereits vom Open Graph Protocol bekannten Objektmodells. Anders als bisher können Entwickler nun aber beliebige Objekttypen selbst modellieren. Dazu stellt Facebook in der Developer-Anwendung (*http://developers.facebook.com/apps*) im Abschnitt Open Graph die entsprechenden Tools bereit.

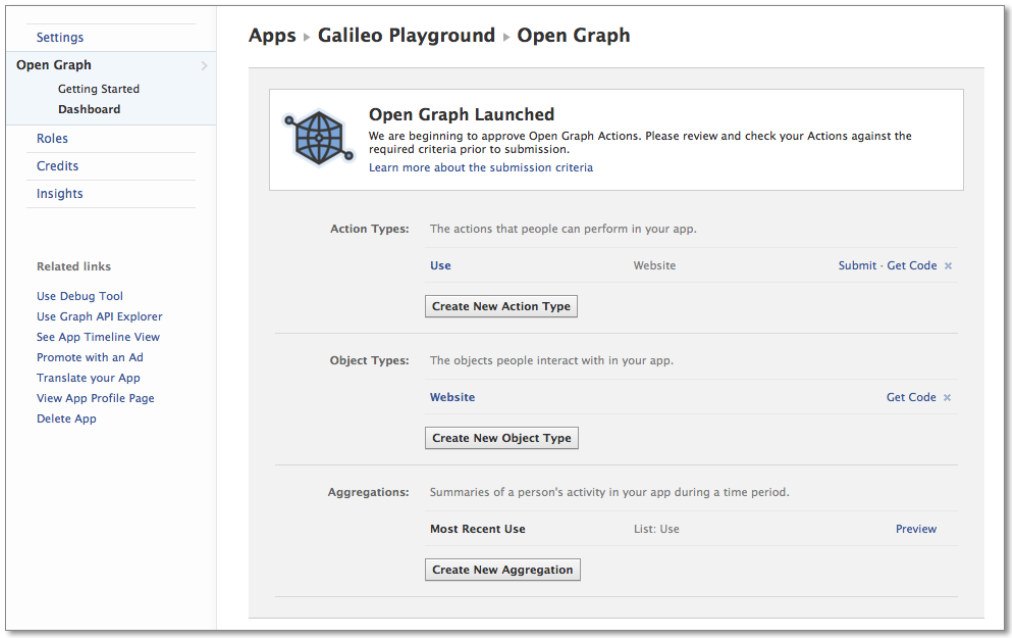


Abbildung 8.15 Das »Open Graph Dashboard« in der Developer-Anwendung

Wichtig: Wie schon im Open Graph Protocol und bei der Implementierung des Like-Buttons gilt auch hier: Individuelle Open-Graph-Objekte müssen auf externen Webseiten unter eindeutiger URL dargestellt werden.

Die Open-Graph-Cocktail-Bar

Um die Funktionen von Open Graph in der Praxis kennenzulernen, werden Sie im Laufe dieses Kapitels eine durchgehende Beispielanwendung erstellen. Stellen Sie sich vor, Sie betreiben eine erfolgreiche Website namens *http://digitalbartender.com*, die sich ganz dem Thema »Cocktails« widmet. Aktuell besteht Ihre Website vor allem aus unzähligen Cocktail-Rezepten und -Empfehlungen – die Besucher der Website haben aber, abgesehen von einer Kommentarfunktion, noch keine Möglichkeiten zur Interaktion.

Mit der Veröffentlichung von Open Graph möchten Sie das jedoch ändern! Künftig soll *http://digitalbartender.com* die erste Anlaufstelle für Cocktail-Liebhaber auf der ganzen Welt werden. Benutzer sollen die Möglichkeit erhalten, ihre liebsten Drink-Rezepte auf Facebook zu teilen und ihren Freunden mitzuteilen, wenn sie gerade einen Cocktail mixen oder trinken – vielleicht hat ja gerade jemand Lust, einen Drink zu teilen! Intuitiv vermuten Sie, dass der flexible Open Graph für die Modellierung Ihrer Objekte (»Cocktail«) und Aktionen (»Trinken«, »Mixen« ...) gut geeignet ist – im folgenden Abschnitt werden Sie dies nun unter Beweis stellen!

Im Anwendungs-Developer sehen Sie, dass Facebook einen Standardobjekttyp namens Website erstellt hat – ignorieren Sie diesen Objekttyp, und klicken Sie stattdessen auf Create New Object Type, um Ihren eigenen Objekttyp zu erstellen!

Da die Drink-Rezepte das zentrale Element Ihrer Website darstellen, geben Sie als Objektnamen »Cocktails« ein. Der Objektname darf maximal 20 Zeichen lang sein und darf aus Buchstaben, Ziffern und Leerzeichen bestehen. In der internen Codierung werden Leerzeichen dabei allerdings zu Unterstrichen umgewandelt. *Tipp:* Objektnamen sollten immer in der Einzahl formuliert werden!

Im Einstellungsdialog des Objekts sehen Sie, dass Facebook bereits einige verpflichtende Merkmale des Objekts (Object Properties) vordefiniert hat.

- Title: Enthält den Titel des Objekts, also in Ihrem Fall den Namen des jeweiligen Cocktails.
- URL: die URL auf der Webseite *http://digitalbartender.com*, unter der das Objekt repräsentiert wird
- Image: URL zum Thumbnail des Objekts – genauer gesagt, wird durch die eckigen Klammern `[]` angedeutet, dass Facebook hier optional auch mehrere Bilder als Array akzeptiert.
- Description: Beschreibungstext des Objekts

Diese verpflichtenden Objektattribute entsprechen den vom Open Graph Protocol bekannten Metadaten-Feldern `og:title`, `og:url`, `og:description` und `og:image`.

Neben den Attributen finden Sie jeweils ein Eingabefeld für Beispieldaten. Diese werden von Facebook genutzt, um weiter unten in der News Feed Preview eine Vorschau von der Darstellung des Objekts im Newsfeed zu erstellen, wie sie angezeigt wird, wenn ein Benutzer auf Gefällt mir klickt. Geben Sie für Ihr Cocktail-Objekt Beispieldaten für Titel, Beschreibung und Thumbnail ein!

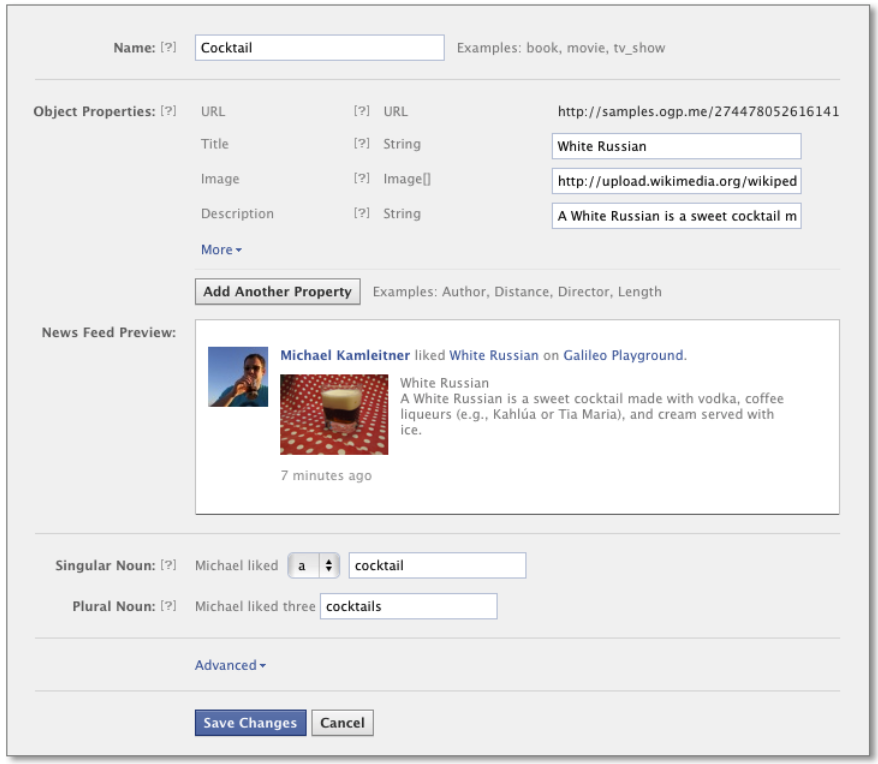


Abbildung 8.16 Anlegen des Cocktail-Objekts im Open Graph

Unter MORE finden sich einige weitere Standardattribute:

- LOCALE: Enthält die Lokalisierung bzw. Sprache, in der das Objekt auf der Webseite repräsentiert wird (z. B. `de_DE`).
- VIDEO: URL zu einem oder mehreren Videos des Objekts
- AUDIO: URL zu einem oder mehreren Audio-Clips des Objekts
- SITE NAME: der Name der Website, also etwa *Digital Bartender*
- DETERMINER: der zu verwendende Artikel in englischer Sprache
- LAST UPDATE TIME: Zeitstempel der letzten Aktualisierung des Objekts als UNIX-Timestamp

Unter SINGULAR NOUN und PLURAL NOUN können Sie die korrekte Ein- und Mehrzahl sowie den Artikel Ihres Objekttyps angeben. Dies ist wichtig, da Facebook auf der Timeline mitunter Interaktionen mit mehreren Objektinstanzen zu einem Eintrag aggregiert. Ein Beispiel wäre etwa die Anzeige MICHAEL LIKED FOUR COCKTAILS, wenn ein Benutzer den LIKE-Button von vier Cocktail-Rezepten angeklickt hat.

Hinter dem Link Advanced verbirgt sich die Einstellung Open Graph Type – hier wird der codierte Objekttyp festgelegt, der anschließend im Open-Graph-Tag `og:type` auf allen Webseiten gesetzt werden muss, die ein Cocktail-Rezept repräsentieren.

```
<meta property="og:type" content="digitalbartender:cocktail" />
```

Der Objekttyp setzt sich aus dem App Namespace der Anwendung (`digitalbartender`), gefolgt von einem Doppelpunkt und dem Objektnamen in Kleinschreibung (`cocktail`), zusammen, etwaige Leerzeichen werden durch Unterstriche ersetzt.

Abbildung 8.17 Die Definition des Objekttyps entspricht üblicherweise dem Objektnamen in Kleinschreibung.

Hinweis: Die Definition von Open-Graph-Objekten und -Aktionen kann prinzipiell mehrsprachig erfolgen und internationalisiert werden. Es empfiehlt sich, alle Angaben zuerst in der primären Sprache der Anwendung vorzunehmen und erst an die Übersetzung zu gehen, wenn das Objektmodell abgeschlossen ist. Da das Open Graph Dashboard selbst noch nicht von Facebook übersetzt wurde, beschränken sich die Beispiele dieses Kapitels auf die englische Sprache.

Nach dem Speichern des Objekttyps landen Sie wieder im Open Graph Dashboard. Dort können Sie unter Object Types Ihr neues Objekt Cocktail bereits sehen. Mit Get Code liefert Facebook ein aus Ihren Beispieldaten generiertes Set an Open Graph Tags:

```
<head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb#
digitalbartender: http://ogp.me/ns/fb/digitalbartender #">
  <meta property="fb:app_id" content="214728715257742" />
  <meta property="og:type"
        content="digitalbartender:cocktail" />
  <meta property="og:url" content="Put Your Own URL Here" />
  <meta property="og:title" content="White Russian" />
  <meta property="og:description" content="A White Russian is a sweet
     cocktail made with vodka, coffee liqueurs (e.g., Kahlúa or
     Tia Maria), and cream served with ice" />
  <meta property="og:image"
        content="http://wikimedia.org/.../White_Russian.jpg" />
```

Listing 8.7 Open Graph Tags für ein Objekt vom Typ »Cocktail«

Wie Sie sehen, entspricht die Notation fast vollständig dem bereits bekannten Open Graph Protocol! Lediglich das `prefix`-Attribut im `<head>`-Element enthält einen zusätzlichen Verweis auf die eigene Anwendung (*http://ogp.me/ns/fb/digitalbartender #*), und das Feld `og:type` enthält anstelle der von Facebook vorgegebenen Objekttypen den aus App Namespace und Objektnamen zusammengesetzten Typ `digitalbartender:cocktail`.

Wichtig: Der Inhalt des Attributs `og:url` wird von Facebook im Beispiel-Code nur mit einem Platzhalter belegt – das Feld muss die URL der Webseite enthalten, auf der die jeweilige Instanz des Objekts repräsentiert wird. Wie beim Open Graph Protocol sollten Sie bei dieser URL darauf achten, eine eindeutige, permanente Adresse zum Objekt zu verwenden:

```
<meta property="og:url"
      content="http://digitalbartender/cocktails/whiterussian" />
```

Um Ihre Website *http://digitalbartender.com* nun auf die Open-Graph-Integration vorzubereiten, müssen Sie auf allen Webseiten, die ein Cocktail-Rezept repräsentieren, dieses Set an Metadaten entsprechend einbauen. Da Ihre Website Cocktail-Rezepte sicherlich in einer Datenbank speichert, genügt es, das entsprechende Template für die Cocktail-Unterseiten einmalig entsprechend anzupassen. Meistens ist es sinnvoll, bei dieser Gelegenheit auch gleich den LIKE-Button in die Webseiten zu integrieren – die Interaktion des GEFÄLLT MIR zwischen Benutzer und Open-Graph-Objekt wird nämlich auch im neuen Open Graph exklusiv mit diesem Social Plugin hergestellt.

Anschließend können Sie den Debugger von Facebook (*https://developers.facebook.com/tools/debug*) nutzen, um die korrekte Implementierung der Tags zu prüfen – der Debugger prüft dabei nicht nur die bereits vom Open Graph Protocol bekannten Objekttypen und -Attribute, sondern berücksichtigt auch die individuellen Objekttypen.

Bevor wir nun das COCKTAIL-Objekt weiter verfeinern, legen Sie einen weiteren Objekttyp namens INGREDIENT an – Objekte diesen Typs werden Sie verwenden, um abzubilden, aus welchen Zutaten die einzelnen Cocktail-Rezepte zubereitet werden. Diese Beziehung – vergleichbar mit einer m:n-Beziehung aus der relationalen Datenbankwelt (»ein Cocktail besteht aus mehreren Zutaten, Zutaten sind Bestandteil mehrerer Cocktails«) – kann auch im Objektmodell von Open Graph abgebildet werden.

Hinweis: Damit die Abbildung von Zutaten im Open Graph funktioniert, müssen Sie gewährleisten, dass auch alle Zutaten jeweils auf einer eigenen Webseite mit entsprechenden Meta-Tags abrufbar sind.

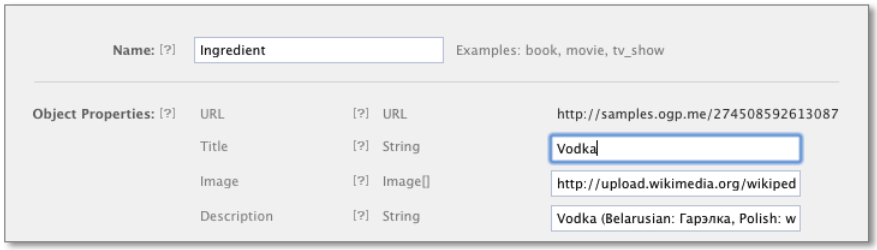


Abbildung 8.18 Der Open-Graph-Objekttyp »Ingredient«

Wie schon im Open Graph Protocol müssen auch alle Instanzen Ihrer eigenen Objekttypen auf einer eigenen, unter eindeutiger URL adressierten Webseite dargestellt werden. Diese Webseite enthält die Open Graph Tags, die das Objekt näher beschreiben. Facebook liest die Open Graph Tags automatisiert von dieser Webseite ein, wenn ein Benutzer erstmals in Interaktion mit dem Objekt tritt (also einen mit dem Objekt verbundenen LIKE-Button anklickt oder eine individuelle Open-Graph-Aktion wie »Trinken«, »Mixen« etc. veröffentlicht).

Der folgende Beispiel-Code zeigt die minimal notwendigen Tags für die Zutat »Vodka«:

```
<meta property="fb:app_id" content="214728715257742" />
<meta property="og:type"
      content="digitalbartender:ingredient" />
<meta property="og:url"
      content="http://digitalbartender.com/ingredients/vodka" />
<meta property="og:title" content="Vodka" />
<meta property="og:description" content="Vodka (Belarusian:,
     Polish: wódka, Russian:  ) is a distilled beverage. It is
   composed primarily of water and ethanol with traces of impurities and
   flavorings. Vodka is made by the distillation of fermented
   substances such as grains, potatoes, or sometimes fruits and/or Sugar." />
<meta property="og:image"
      content="http://wikimedia.org/.../Vodka_Sobieski.jpg" />
```

Listing 8.8 Open Graph Tags für ein Objekt vom Typ »Ingredient«

Neben den bereits bekannten Tags für Titel, Thumbnail-Grafik und Beschreibungstext ist hier vor allem der Objekttyp zu beachten, der im Attribut `og:type` den Wert `digitalbartender:ingredient` enthält. Im Attribut `og:url` muss wie üblich die eindeutige URL zu jener Webseite angegeben werden, die das Objekt repräsentiert, also für die Zutat »Vodka« z. B. *http://digitalbartender.com/ingredients/vodka*.

Legen Sie für die folgenden Beispiele einige Webseiten mit Zutaten auf Ihrem Server ab, etwa für »Vodka«, »Kahlua« und »Cream«, den Zutaten unseres Beispielcocktails »White Russian«:

http://digitalbartender.com/ingredients/vodka
http://digitalbartender.com/ingredients/kahlua
http://digitalbartender.com/ingredients/cream

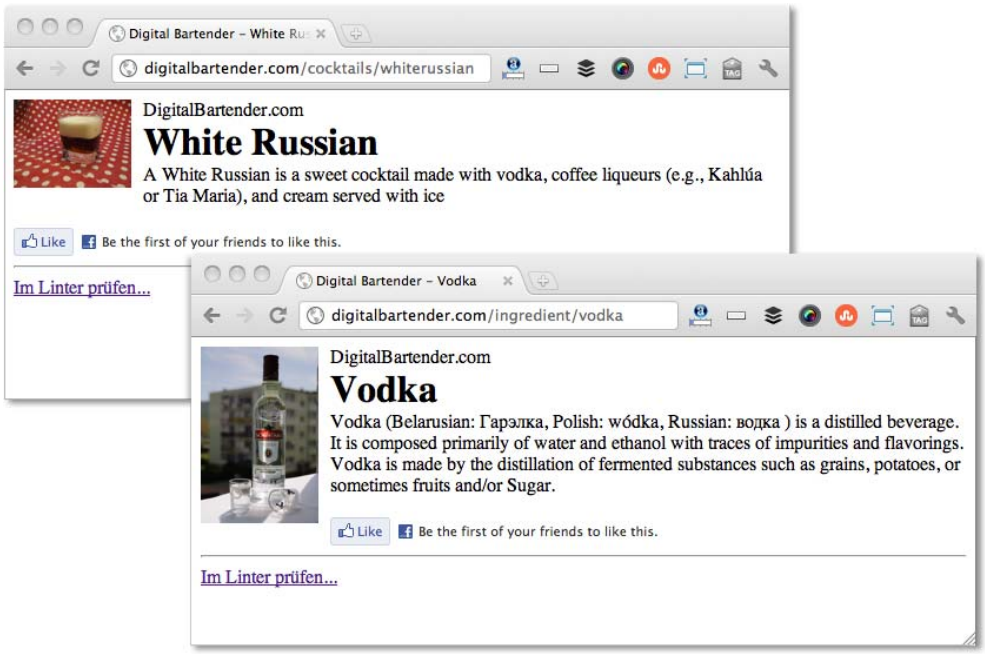


Abbildung 8.19 Beispielseiten für Objekte vom Typ »Cocktail« und »Ingredient«

Individuelle Objektattribute

Wechseln Sie nun zurück zu den Objekteinstellungen des COCKTAIL-Objekts – im nächsten Schritt werden Sie diesen Objekttyp mit weiteren, individuellen Attributen versehen. Mit der Schaltfläche ADD ANOTHER PROPERTY können Sie neue Attribute zum Objekt hinzufügen. Dabei haben Sie die Wahl zwischen folgenden Datentypen:

- BOOLEAN für Ja-/Nein-Felder
- DATETIME für Zeitstempel (Angabe jeweils als UNIX-Timestamp oder im ISO-8601-Format)
- FLOAT für Gleitkommazahlen
- INTEGER für ganzzahlige Werte
- STRING für beliebige Zeichenketten
- AUDIO für die Verknüpfung von Audiodaten

- Video für die Verknüpfung von Videodaten
- ContactInfo für die Verknüpfung einer in Adresse, Postleitzahl, Stadt etc. strukturierten Kontaktadresse
- GeoPoint für geographische Informationen, bestehend aus Längengrad, Breitengrad und Seehöhe
- Profile für die Verknüpfung eines Benutzerprofils
- Image für die Verknüpfung von Bildern
- Quantity für die Angabe von Maßzahlen in verschiedenen Einheiten (Meter, Kilogramm etc.)

Neben dem Datentyp können Sie für jedes Feld wiederum Beispieldaten eingeben, die in der Newsfeed-Vorschau angezeigt werden. Über das zweite Dropdown-Menü können Sie einstellen, ob es sich um ein Pflichtfeld handelt (Is Required), ob das Feld mehrfach angegeben werden darf (Is Array) und ob das Feld in Timeline-Einträgen angezeigt werden soll (Hide from News Feed).

Experimentieren Sie mit individuellen Erweiterungen Ihres Cocktail-Objekts! Sollen etwa Benutzer in Ihrer Anwendung die Möglichkeit haben, eigene Cocktail-Rezepte zu veröffentlichen, wäre es praktisch, den Ersteller des Rezepts im Objekt abzubilden. Dazu fügen Sie ein Feld Creator vom Typ Profile hinzu. Als Beispielwert tragen Sie die Benutzer-ID Ihres eigenen Facebook-Kontos ein. Eine weitere Ergänzung könnte das Feld Created vom Typ DateTime sein, das den Zeitstempel der Veröffentlichung des Rezepts enthält.

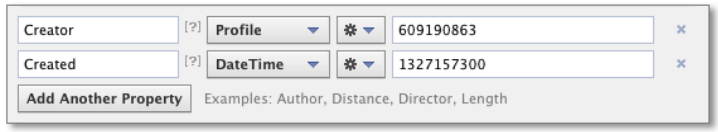


Abbildung 8.20 Hinzufügen individueller Objektattribute

Nachdem Sie einige Attribute hinzugefügt haben, wechseln Sie wieder zum Dashboard und lassen sich mittels Get Code ein neues Set an Beispiel-Tags erzeugen:

```
<meta property="digitalbartender:creator" content="609190863" />
<meta property="digitalbartender:created"
      content="2012-01-21T06:48:20-08:00" />
```

Wie Sie sehen, setzen sich die Attributnamen der ergänzten Felder – ähnlich wie bereits der Objekttyp – wiederum aus dem App Namespace, gefolgt von einem Doppelpunkt, und dem Feldnamen zusammen (`digitalbartender:created`). Dies dient auch der eindeutigen Unterscheidung von Feldnamen unterschiedlicher Applikationen.

Verknüpfungen zwischen Objekttypen

Besonders interessante Möglichkeiten ergeben sich durch die Verknüpfung eines Attributs mit einem anderen Open-Graph-Objekttyp. Damit kann z. B. die bereits erwähnte Beziehung zwischen »Cocktails« und deren Zutaten (»Ingredients«) hergestellt werden. Beim Hinzufügen eines neuen Attributs kann dazu als Datentyp neben den bereits aufgezählten Standarddatentypen auch ein anderer Objekttyp Ihrer Anwendung ausgewählt werden.

Um Cocktail-Rezepte strukturiert speichern zu können, erweitern Sie nun den Objekttyp COCKTAIL um ein Attribut INGREDIENTS. Als Datentyp wählen Sie nun INGREDIENT, womit der zweite Objekttyp, den Sie zuvor angelegt haben, referenziert wird. Da fast alle Cocktails aus mehreren Zutaten zubereitet werden, wählen Sie im zweiten Dropdown IS ARRAY.

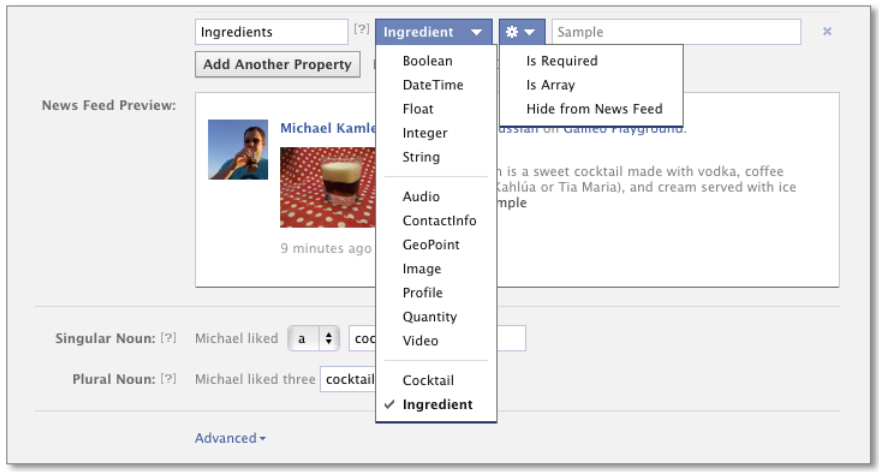


Abbildung 8.21 Erstellen des Attributs »Ingredients«, das ein Array an Objekten vom Typ »Ingredient« enthält

Nach dem Speichern des neuen Attributs gehen Sie nun daran, die Metadaten auf der Webseite des Cocktails »White Russian« um die Zutaten zu ergänzen, die man zur Zubereitung dieses Cocktails benötigt. Dazu fügen Sie folgenden Code ein:

```
<meta property="digitalbartender:ingredients"
      content="http://digitalbartender/ingredients/vodka" />
<meta property="digitalbartender:ingredients"
      content="http://digitalbartender/ingredients/kahlua" />
<meta property="digitalbartender:ingredients"
      content="http://digitalbartender/ingredients/cream" />
```

Listing 8.9 Open Graph Tags zur Definition mehrerer Zutaten eines Objekts vom Typ »Cocktail«

Da Sie das Feld INGREDIENTS als Array definiert haben, ist die Angabe mehrerer Zutaten zulässig – dabei müssen Sie lediglich das entsprechende Open Graph Tag für jede Zutat separat angeben. Der Inhalt des Attributs ist jeweils jene URL, unter der das Open-Graph-Objekt der Zutat abrufbar ist.

Nach jeder Änderung der Open Graph Tags empfiehlt es sich, mit dem URL Debugger zu kontrollieren, ob die Notation der Tags syntaktisch und inhaltlich richtig ist:

Abbildung 8.22 Kontrollieren der Open Graph Tags mit dem URL Debugger

Wie Sie sehen, wurde das Feld `digitalbartender:ingredients` korrekt als Array mit drei verknüpften Objekten vom Typ INGREDIENT erkannt – Ihre Objektverknüpfung zwischen »Cocktail« und »Zutaten« funktioniert also bestens!

8.2.4 Definieren von Aktionen

Nachdem Sie eigene Open-Graph-Objekttypen definiert und sich auch mit der Erweiterung um individuelle Attribute und Objektverknüpfungen beschäftigt haben, gehen Sie nun an die Definition der Aktionen (*Actions*), mit denen die Benutzer Ihrer Website mit den Objekten in Interaktion treten können. Anders ausgedrückt: Aktionen sind die Kanten im sozialen Graphen, die Objekte mit Benutzern verbinden. Für DigitalBartender.com werden Sie als Erstes eine Aktion definieren, die verwendet werden soll, wenn Ihre Benutzer ein Cocktail-Rezept in die Tat umsetzen und den Drink »mixen«.

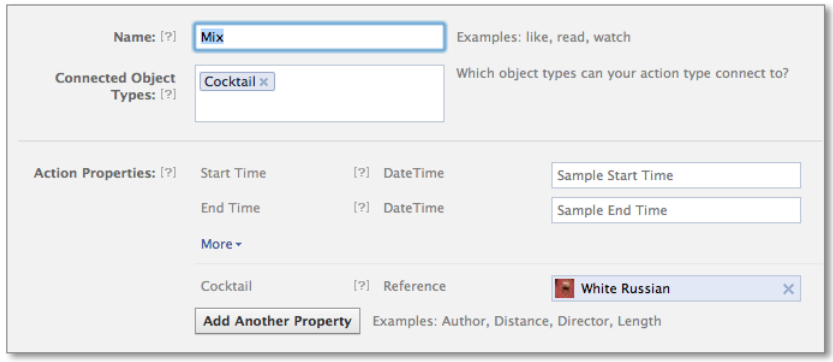


Abbildung 8.23 Festlegen der verknüpften Objekttypen und Attribute der Open-Graph-Aktion »Mix«

Nach dem Klick auf den Button CREATE NEW ACTION TYPE im Open Graph Dashboard legen Sie zuerst das Verb fest, dass die Aktion beschreibt – in Ihrem Fall also MIX. Im folgenden Einstellungsdialog spezifizieren Sie die Aktion näher:

- NAME: der Name der Aktion, meistens das Verb, das die Aktion beschreibt (MIX)
- CONNECTED OBJECT TYPES: Jede veröffentlichte Open-Graph-Aktion muss mit mindestens einem Open-Graph-Objekt verknüpft werden. In diesem Feld geben Sie an, welche Objekttypen für die Aktion MIX zulässig sind. Die Aktion MIX soll verwendet werden, wenn ein Benutzer einen Cocktail zubereitet – wählen Sie hier also den eigenen Objekttyp COCKTAIL aus (das automatisch vervollständigende Eingabefeld hilft dabei, aus den zulässigen Objekttypen zu wählen). In manchen Anwendungen kann die Angabe mehrerer Objekttypen sinnvoll sein – denken Sie etwa an die Aktion »Lesen«, die sich z. B. auf Objekte vom Typ »Buch«, »Magazin« oder »Zeitung« beziehen könnte.
- ACTION PROPERTIES: Ähnlich wie Objekte können auch Aktionen bei der Veröffentlichung mit zusätzlichen Attributen versehen werden. Neben einigen unveränderlichen Standardattributen hat Facebook aufgrund Ihrer Angabe des Objekttyps COCKTAIL in der Einstellung CONNECTED OBJECT TYPES hier automatisch ein entsprechendes Attribut COCKTAIL eingefügt. Dieses Attribut referenziert den Cocktail, der in der Aktion MIX jeweils veröffentlicht werden soll.

Neben den verknüpften Objekten bietet Facebook folgende ACTION PROPERTIES standardmäßig an:

- START TIME und END TIME: Zeitstempel im UNIX-Timestamp- oder ISO-8601-Format, an dem die veröffentlichte Aktion beginnt und endet. Mithilfe dieser Angaben entscheidet Facebook, ob die veröffentlichte Aktion in Gegenwarts- oder Vergangenheitsform angezeigt wird. Die meisten Anwendungen setzen diese Attribute nicht explizit, sondern gehen von der Annahme aus, dass Aktionen in

Echtzeit erfolgen, also zu dem Zeitpunkt, zu dem sie vom Benutzer angestoßen werden. Die Dauer der Aktion wird dabei mit einem von Facebook nicht genau spezifizierten Default-Wert von wenigen Minuten festgelegt.

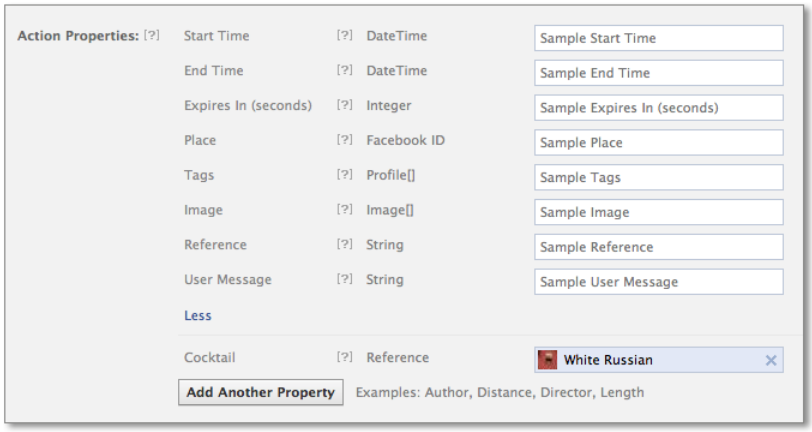


Abbildung 8.24 Die »Action Properties« geben weitere Auskunft über eine veröffentlichte Aktion.

- EXPIRES IN: Alternativ zu den Zeitstempeln kann die Dauer der Aktion in diesem Attribut in Sekunden angegeben werden. Auch diese Information wird von Facebook zur Wahl von Gegenwarts- oder Vergangenheitsform genutzt.
- PLACE: Dieses Attribut ermöglicht die optionale Verknüpfung eines Ortes mit der Aktion. Dies erlaubt Entwicklern, Open-Graph-Aktionen einen geographischen Aspekt zu verleihen. Die Angabe des Attributs erfolgt als numerische ID eines Seitenprofils mit Ortsangabe (PLACE). Ein klassischer Anwendungsfall für die Verwendung von PLACE wäre eine Checkin-Anwendung wie Foursquare.
- TAGS: Mit diesem Attribut können weitere Facebook-Benutzer in der veröffentlichten Aktion markiert werden. Die Angabe erfolgt als kommaseparierte Liste an Benutzer-IDs. *Hinweis:* Dieses Attribut wird aktuell noch nicht von Facebook unterstützt!
- IMAGE: Mit diesem Attribut kann ein aktionsspezifisches Bild mit der veröffentlichten Aktion verknüpft werden. Dieses Bild wird von Facebook im entsprechenden Timeline-Eintrag angezeigt. Die meisten Anwendungen verwenden dieses Attribut jedoch nicht, da Facebook in diesem Fall automatisch auf das im verknüpften Objekt angegebene Bild zurückgreift. »Mixt« ein Benutzer Ihrer Anwendung also einen Cocktail, würde in der veröffentlichten Aktion also automatisch das Thumbnail des Cocktail-Objekts angezeigt werden. Bietet Ihre Anwendung hingegen Benutzern die Möglichkeit, eigene Fotos des gemixten Getränks hochzuladen, könnte diesem Bild aktionsspezfisich (also nur für diesen Benutzer) der Vorrang gegeben werden.

- Reference: Dieses Feld erlaubt die Angabe eines beliebigen Referenzwertes, der später bei der statistischen Auswertung der Open-Graph-Nutzung Ihrer Anwendung in den Insights zum Tracking genutzt werden kann.
- User Message: Mit diesem Attribut können veröffentlichte Aktionen um eine vom Benutzer eingegebene Nachricht ergänzt werden. Vielleicht möchten Sie Ihren Benutzern die Möglichkeit geben, gemixte Cocktails mit einer kurzen Bemerkung zu kommentieren – diesen Kommentar können Sie im Feld User Message hinterlegen.

Auch die Action Properties können um individuelle Attribute ergänzt werden – dabei stehen die gleichen Datentypen und Einstellungsmöglichkeiten wie bei Objekten zur Verfügung.

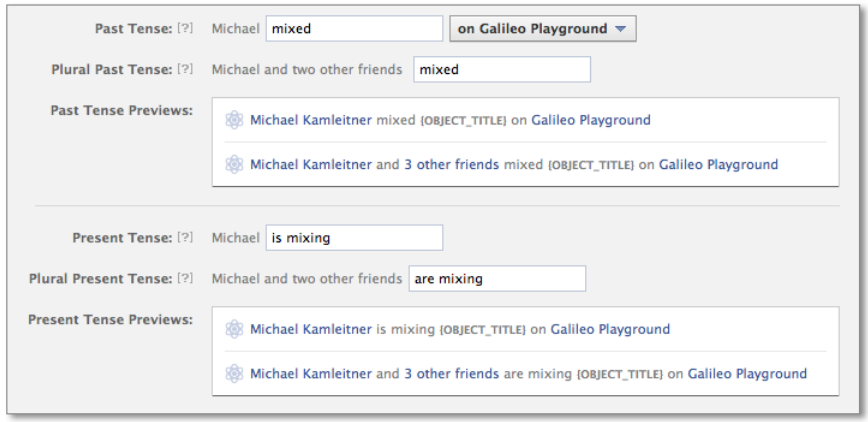


Abbildung 8.25 Das Verb der Aktion muss in Einzahl und Mehrzahl sowie in Gegenwarts- und Vergangenheitsform angegeben werden.

Spracheinstellungen und Internationalisierung

Die folgenden Einstellungen regeln die sprachliche Darstellung der veröffentlichten Aktionen in Newsfeed- und Ticker-Einträgen. Facebook unterscheidet hier automatisch, ob die Aktion in diesem Moment stattfindet (Ticker) oder schon einige Zeit zurückliegt (Newsfeed, Timeline), und wählt dementsprechend die Gegenwarts- oder Vergangenheitsform. Darüber hinaus kann es vorkommen, dass Facebook Open-Graph-Aktionen, die von mehreren Freunden unabhängig voneinander veröffentlicht wurden, zu einem gemeinsamen Eintrag zusammenfasst – daher muss das Verb der Aktion jeweils in passender Einzahl und Mehrzahl angegeben werden:

- Past Tense und Plural Past Tense legen die Vergangenheitsform des Verbs in Einzahl und Mehrzahl fest (mixed).
- Present Tense und Plural Present Tense legen die Gegenwartsform des Verbs in Einzahl und Mehrzahl fest (is mixing, are mixing).

Facebook erleichtert die sprachlichen Einstellungen durch Vorschauanzeigen und automatische Vorgaben in englischer Sprache.

Auch Aktionen können mit den Lokalisierungs-Tools von Facebook in verschiedene Sprachen übersetzt und angezeigt werden. Wie bei Objekten sollten Sie aber auch hier zuerst in englischer Sprache arbeiten und die Übersetzung erst vornehmen, wenn Ihre Aktionen fertig definiert sind.

Darstellung von Open-Graph-Aktionen

Im folgenden Abschnitt der Aktionseinstellungen kann festgelegt werden, wie eine einzelne Aktion am Timeline-Profil und der Timeline-View der Anwendung angezeigt werden soll. Die Einstellung ATTACHMENT LAYOUT erlaubt dabei die Wahl von drei unterschiedlichen Darstellungen. Die Default-Einstellung ITEM gleicht den Wall-Postings, die durch Anklicken des LIKE-Buttons erzeugt werden – neben dem Thumbnail-Bild wird dabei lediglich der Objekttitel dargestellt. Als Entwickler können Sie darüber hinaus im Eingabefeld CAPTION festlegen, welche Informationen unter dem Objekttitel angezeigt werden sollen.

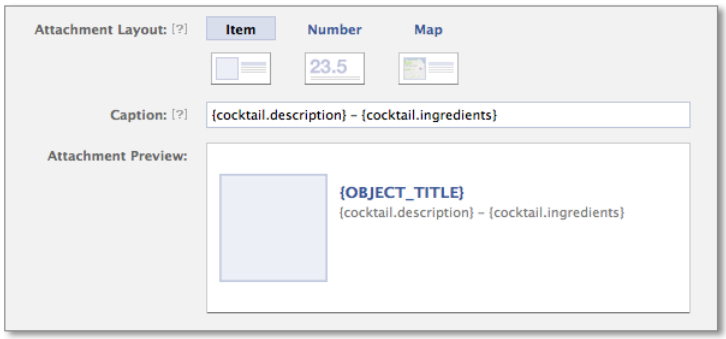


Abbildung 8.26 Einstellungen zur Darstellung von Open-Graph-Aktionen

Dabei können Sie mit einer speziellen Syntax auch Attribute des Objekts oder der Aktion als Platzhalter oder Variablen mit einbeziehen. Um einen Platzhalter einzufügen, müssen Sie ihn zwischen geschwungenen Klammern `{}` anführen. Facebook ist dabei wieder mit einer Auto-Vervollständigung behilflich.

Um etwa Attribute des Objekts, auf das sich die Aktion bezieht, in die CAPTION mit aufzunehmen, genügt eine Angabe der Form `{Objektname.Attributname}`. Der Platzhalter `{cocktail.description}` würde also den Beschreibungstext des gemixten Cocktails anzeigen, `{cocktail.ingredients}` eine automatisch erzeugte Liste der Zutaten. Doch auch die Attribute der Aktion selbst können Teil des Timeline-Eintrags werden. Die Angabe erfolgt dabei in der Form `{Attributname}` – also ohne explizite Angabe des Namens der Aktion. Mit `{start_time}` könnten Sie etwa den Zeitpunkt der Veröffentlichung der Aktion in die CAPTION integrieren.

Die Platzhalter können mit beliebigen fixen Zeichenketten zu einem vollständigen Satz kombiniert werden:

```
On {start_time}, mixed with {cocktail.ingredients}
```

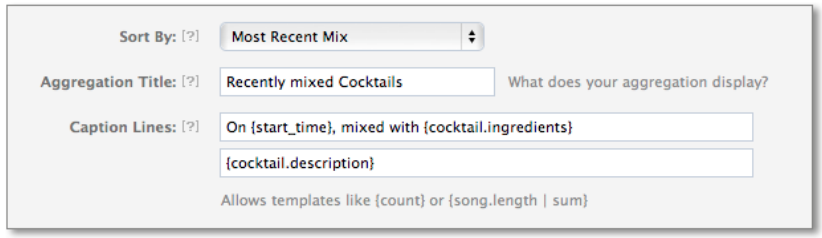


Abbildung 8.27 Kombination von Platzhaltern und festen Zeichenketten in der »Caption« einer Aktion

Hinweis: Wie Objekte und Aktionen können auch Aggregationen mit den Internationalisierungs-Tools von Facebook übersetzt werden!

Mit der Einstellung NUMBER steht eine alternative Darstellung von Aktionen zur Verfügung, die einen einzelnen numerischen Wert in den Vordergrund rückt. So könnten Sie etwa neben MIX eine weitere Aktion namens RATE definieren, die von Ihren Benutzern verwendet werden kann, um Cocktail-Rezepte zu bewerten – z. B. mit einem bis fünf Sternen. Das Benutzer-Interface können Sie am einfachsten mit einem Dropdown realisieren – wählt der Benutzer eine Option aus dem Dropdown, wird automatisch die entsprechende Open-Graph-Aktion publiziert.

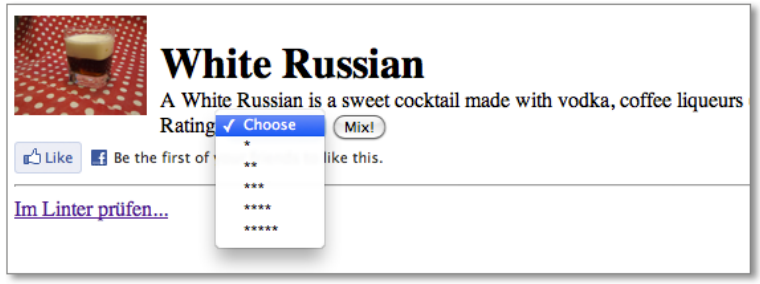


Abbildung 8.28 Benutzer-Interface zum Bewerten von Cocktail-Rezepten

Die Open-Graph-Aktion RATE müsste dazu ein zusätzliches Integer-Attribut namens RATING erhalten, in dem Sie die individuelle Bewertung eines Benutzers speichern könnten. Das referenzierte Objekt ist wieder vom Typ COCKTAIL.

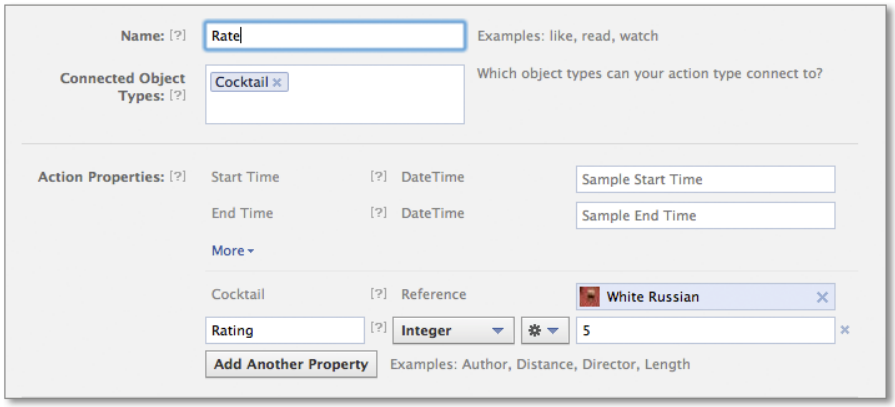


Abbildung 8.29 Definition der Open-Graph-Aktion »Rate«

Um die Aktion RATE ansprechend in der Timeline darzustellen, bietet sich das ATTACHMENT LAYOUT mit der Einstellung NUMBER an – das Rating kann dabei in großer Schriftgröße gut sichtbar angezeigt werden. Dazu muss in das Feld NUMBER lediglich der entsprechende Platzhalter `{rating}` eingegeben werden. Die `Caption` kann auch hier aus festem Text und weiteren Platzhaltern, die sich auf das Objekt oder die Aktion beziehen, zusammengebaut werden.

Abbildung 8.30 »Attachment-Layout« und »Caption« der Aktion »Rate«

Zu guter Letzt können Open-Graph-Aktionen mit geographischem Bezug mit der Option MAP auch auf einer Landkarte dargestellt werden. Voraussetzung dafür ist, dass entweder das von der Aktion referenzierte Objekt (etwa ein »Checkin« bei einer »Location«) oder die Aktion selbst ein individuelles Attribut oder ein Array vom Typ GEOPOINT bereitstellen. Dieser Datentyp enthält strukturierte Felder für Längen- und Breitengrad (sowie optional der Seehöhe), womit die entsprechende Aktion oder das durch die Aktion referenzierte Objekt auf einer Landkarte markiert werden kann.

```
<meta property="namespace:feldname:latitude" content="1" />
<meta property="namespace:feldname:longitude" content="2" />
<meta property="namespace:feldname:altitude" content="3" />
```

Listing 8.10 Definition eines Feldes vom Typ »GeoPoint« als Open Graph Tags eines Objekts

Folgende Einstellungsmöglichkeiten des Layouts MAP sind möglich:

- Mit HIGHLIGHTED POINTS kann ein Feld oder Array vom Typ GEOPOINT angegeben werden. Die so referenzierten Punkte werden mit Markern auf der Landkarte hervorgehoben. Um z. B. einen Checkin auf einem Objekt vom Typ LOCATION mit dem GEOPOINT-Datenfeld `latlng` darzustellen, könnten Sie hier etwa einen einzelnen Datenpunkt mit dem Platzhalter `{location.latlng}` angeben.
- Zusätzlich können Sie im Feld ROUTE ein Array vom Typ GEOPOINT angeben. Die Punkte des Arrays werden auf der Karte als durchgehende Linie dargestellt. Ein Beispiel für die Anwendung dieser Aktionseinstellung wäre das Veröffentlichen einer per Smartphone und GPS aufgezeichneten Laufstrecke.

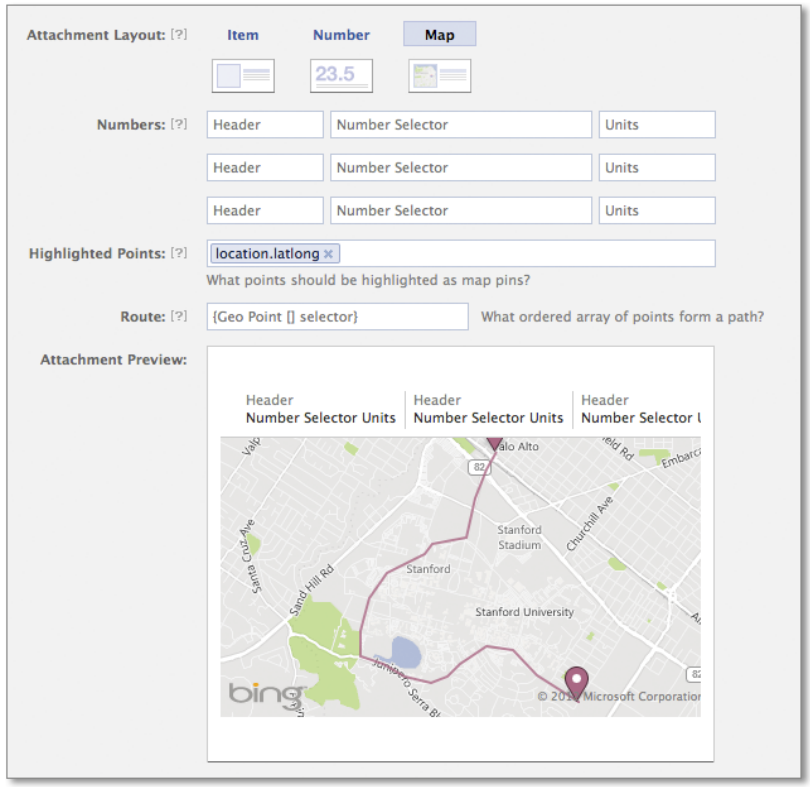


Abbildung 8.31 »Attachment Layout« mit der Einstellung »Map«

- Die Einstellungsfelder unter der Bezeichnung NUMBER erlauben die Einblendung von maximal drei zusätzlichen numerischen Attributen. Bezieht sich die veröffentlichte Route auf eine Laufstrecke, könnten diese Werte etwa die Dauer, die zurückgelegte Distanz und die durchschnittliche Geschwindigkeit sein. Sie können jedes dieser Attribute dabei mit seiner Bezeichnung (AVG. SPEED), dem Wert (9.6) sowie einer Maßeinheit (KM/H) definieren. Der Wert wird dabei ein Platzhalter für ein Attribut der veröffentlichten Aktion sein, etwa `{avgspeed}`.

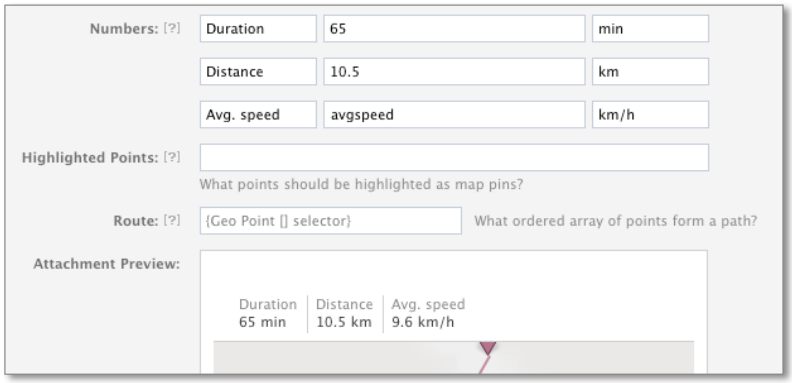


Abbildung 8.32 Zusätzliche Angaben einer als Map veröffentlichten Aktion

8.2.5 Definieren von Aggregationen

Aggregationen sind zusammenfassende Darstellungen mehrerer Aktionen, die von einem Benutzer in einer bestimmten Anwendung veröffentlicht wurden. Facebook zeigt Aggregationen von häufig verwendeten Open-Graph-Anwendungen automatisch in der Timeline des Benutzers an, um Einträge innerhalb eines bestimmten Zeitraums gesammelt darzustellen.

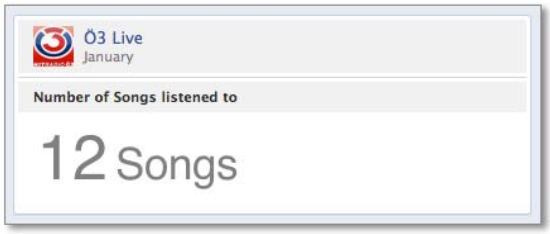


Abbildung 8.33 Aggregation einer Musikanwendung, die die Anzahl der abgespielten Songs innerhalb eines Monats anzeigt

Aggregationen werden derzeit sowohl im Timeline-Profil eines Benutzers dargestellt als auch auf den Timeline-Views, die ja letztlich nur einen anwendungsspezifischen Ausschnitt der gesamten Timeline darstellen.

Hinweis: Wenn Sie beginnen, mit Aggregationen zu experimentieren, dauert es meist einige Stunden, bis Facebook diese auch tatsächlich auf Ihrer Timeline anzeigt. Im Wesentlichen kommt es dabei darauf an, wie viele Open-Graph-Aktionen über Ihr Konto veröffentlicht werden – je regelmäßiger und häufiger das Nutzungsverhalten einer bestimmten Anwendung ist, desto wahrscheinlicher ist es, dass Facebook die zugehörigen Aggregationen anzeigt.

Stellt eine Anwendung verschiedene Aggregationen bereit, werden diese von Facebook in einem sogenannten *Report* zusammengefasst. Auch für Reports gilt, dass sich die Aggregationen, aus denen sie sich zusammensetzen, immer auf einen bestimmten Zeitraum – meist ein Monat – beziehen. Facebook entscheidet auch hier selbstständig aufgrund des Datenmaterials, das durch die vom Benutzer veröffentlichten Aktionen bereitsteht, welche Reports und Aggregationen an welchem Punkt der Timeline angezeigt werden.

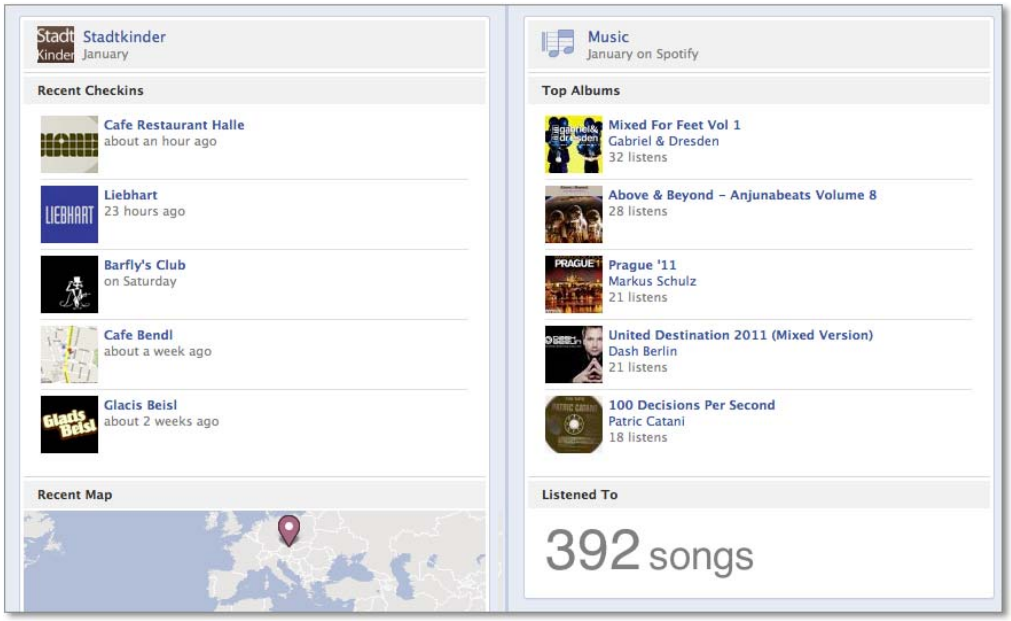


Abbildung 8.34 Timeline-Darstellung von zwei »Reports« unterschiedlicher Anwendungen. In der linken Spalte finden sich die Aggregationen »Recent Checkins« und »Recent Map« der Anwendung »Stadtkinder« zu einem gemeinsamen Report vereint.

Hinweis: Wird eine Aggregation bereits auf der Timeline oder der Timeline-View Ihres eigenen Benutzers angezeigt, wirken sich Änderungen, die Sie im Open Graph Dashboard vornehmen, meist sofort auf die Anzeige aus. Dies erleichtert das Modifizieren bestehender Aggregationen ungemein!

Erstellen von Aggregationen

Vom Open Graph Dashboard gelangen Sie mit dem Button CREATE NEW AGGREGATION in den Einstellungsdialog Ihrer ersten Aggregation. Die obersten Einstellungen DATA TO DISPLAY und ACTION TYPE legen grundlegend fest, auf welche Open-Graph-Daten sich die Aggregation beziehen soll. Dabei haben Sie die Wahl zwischen zwei Herangehensweisen:

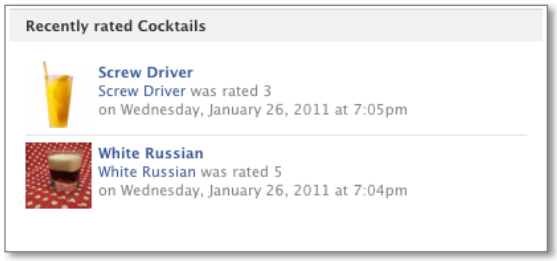


Abbildung 8.35 Die aktionsbezogene Aggregation »Recently rated Cocktails«

- Die Aggregation kann sich auf veröffentlichte Aktionen, also etwa RATE, beziehen. Dazu wählen Sie im Feld DATA TO DISPLAY einfach den gewünschten Aktionstyp aus. Aggregationen, die sich auf Aktionen beziehen, können auch Attribute der einzelnen veröffentlichten Aktion anzeigen. Ein Beispiel für eine *aktionsbezogene Aggregation* wäre RECENTLY RATED COCKTAILS, die die zuletzt vom Benutzer veröffentlichten Cocktail-Bewertungen anzeigen könnte. Dabei könnten etwa auch die jeweilige Bewertung (WAS RATED 3) und das Datum, an dem diese veröffentlicht wurde, angezeigt werden.

Abbildung 8.36 Die objektbezogene Aggregation »Regularly mixed Cocktails«

- Aggregation können sich alternativ auf einen Objekttyp und eine zugehörige Aktion beziehen, also etwa COCKTAILS und MIX. Eine solche Aggregation wird also aus allen Objekten gebildet, mit denen der Benutzer über die gewählte Aktion in Interaktion getreten ist. Dabei führt Facebook eine Gruppierung der veröffentlichten Aktionen auf Objektebene durch – vergleichbar mit einer `group by`-Klausel in SQL. Damit kann z. B. über den Platzhalter `{count}` die Anzahl der mit diesem

Objekt veröffentlichten Aktionen angezeigt werden. Ein Beispiel für eine *objektbezogene Aggregation* wäre Regularly mixed Cocktails, die die am meisten gemixten Cocktails des Benutzers, mitsamt der jeweiligen Anzahl der Zubereitungen (Mixed 3 times), anzeigen könnte.

Abbildung 8.37 Einstellung des »Layout Styles« einer Aggregation

Mit der Einstellung Layout Style können Sie festlegen, in welcher Form die aggregierten Daten dargestellt werden sollen:

- List erzeugt eine listenartige Darstellung der aggregierten Daten. Dabei werden je nach Typ der Aggregation entweder Objekte oder Aktionen mit Thumbnail, Objekttitel und zwei frei definierbaren Textzeilen dargestellt.
- Gallery funktioniert ähnlich der Listendarstellung, rückt durch eine größere Darstellung allerdings das Objektbild in den Vordergrund.
- Table ermöglicht die tabellarische Darstellung von aggregierten Objekten und Aktionen. Sie können dabei neben der Titelspalte bis zu drei weitere Spalten mit Spaltenüberschrift und -inhalt frei definieren.
- Map ist für die Aggregation von Objekten und Aktionen mit geographischen Metainformationen gedacht. Dabei kann gewählt werden, ob die Landkarte eine oder mehrere Aktionen anzeigen soll (Single Actions oder Multiple Actions). Wie bei der Darstellung von Aktionen können auch hier mit Highlighted Points und Route entweder einzelne Punkte oder zusammenhängende Wegstrecken dargestellt werden. Mit den Einstellungen unter Number können zusätzliche Informationen über die veröffentlichten Datenpunkte angezeigt werden.
- Number erlaubt die hervorgehobene Darstellung einer bestimmten, aussagekräftigen Zahl (z. B. Number of mixed Cocktails). Wie bei Aktionen können auch hier die Maßeinheit und eine freie Textzeile definiert werden.

Alle Layout-Typen sind grundsätzlich sowohl für aktions- als auch für objektbezogene Aggregationen verfügbar. Die detaillierte Konfiguration der Darstellung erfolgt ähnlich wie bei den Aktionen über die Felder Aggregation Title (Überschrift bzw. Name der Aggregation) und Caption Lines (ein bis zwei frei definierbare Textzeilen).

Mit der Option Sort By kann festgelegt werden, nach welchem Kriterium die Aggregation sortiert werden soll. Die Standardeinstellung Most Recent sortiert die aggregierten Aktionen rückwärts nach ihrem Erstellungsdatum, während Favorite jene

Objekte zuerst anzeigt, mit denen die meisten verknüpften Aktionen veröffentlicht wurden. Damit ist FAVORITE auch nur für objektbezogene Aggregationen sinnvoll und verfügbar! Über die Einstellung CUSTOM können Sie darüber hinaus nach beliebigen Objekt- oder Aktionsattributen sortieren.

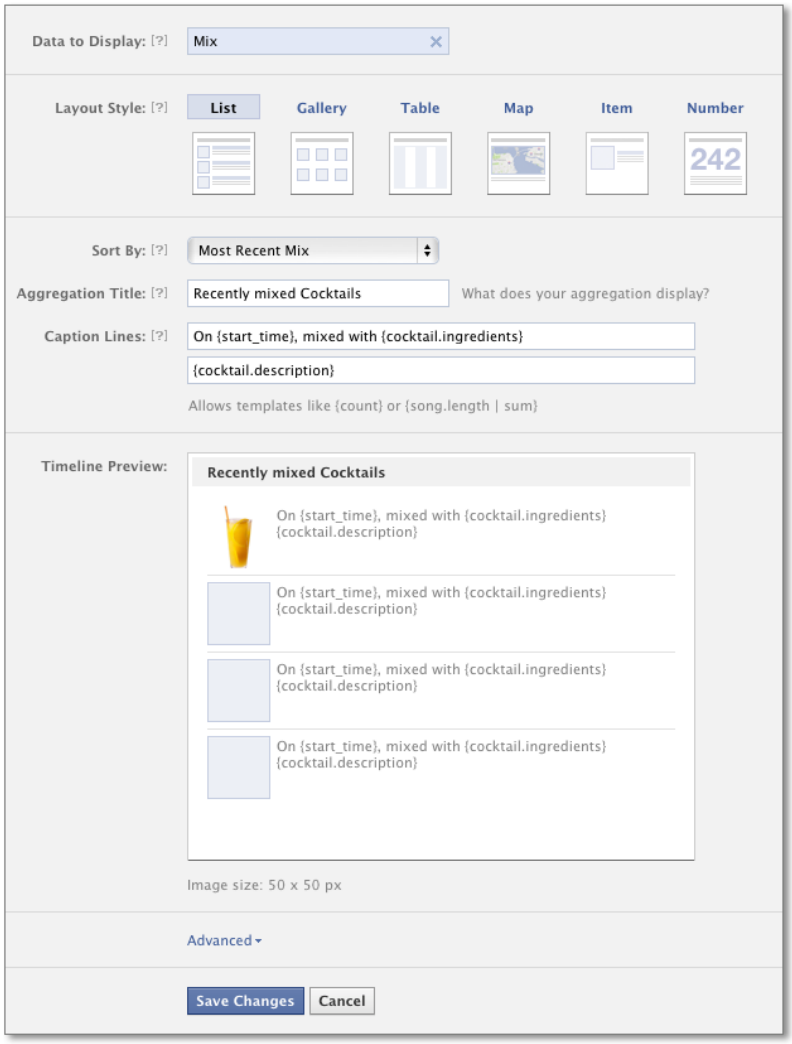


Abbildung 8.38 Erstellen der Aggregation »Recently mixed Cocktails«

Hinter der Einstellung ADVANCED verbergen sich zwei weitere interessante Möglichkeiten, die Sie zur Anpassung Ihrer Aggregationen nutzen können:

- Mit ADD FILTER können Sie flexible Kriterien definieren, die ein Objekt oder eine Aktion erfüllen muss, um Teil der Datenbasis für die Aggregation zu werden. Ein Filter ist vergleichbar mit der `WHERE`-Klausel in SQL und bewirkt, dass nur jene

Aktionen und Objekte berücksichtigt werden, die eine oder mehrere bestimmte Bedingungen erfüllen. ADD FILTER erlaubt die Prüfung eines Objekt- oder Aktionsattributs mit den Operatoren <, >, ==, !=, >= und <= und einem fest zu definierenden Vergleichswert. Der Vergleichswert kann als numerischer Wert oder als Zeichenkette angegeben werden.

- GROUP BY ist eine nicht manuell veränderbare Einstellung, die verdeutlicht, nach welchem Objekttyp objektbezogene Aggregationen gruppiert werden. Damit ist diese Einstellung mit der `group by`-Klausel von SQL vergleichbar.

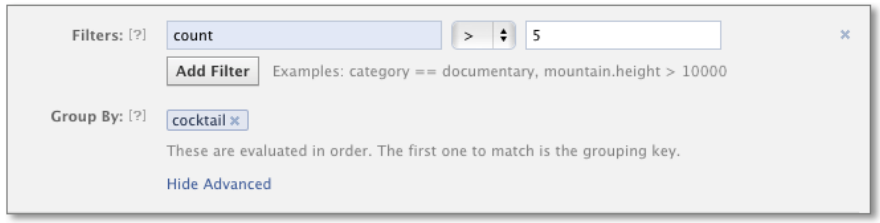


Abbildung 8.39 Die abgebildete Einstellung der Aggregation »Regularly mixed Cocktails« berücksichtigt ausschließlich Cocktails, die mehr als fünfmal zubereitet wurden. Unter »Group By« wird angezeigt, dass die Aggregation nach Cocktail-Objekten gruppiert wird.

Zwei Beispiel-Aggregationen für DigitalBartender.com

Versuchen Sie nun, die eingangs erwähnten Beispiel-Aggregationen für Ihre Website DigitalBartender.com umzusetzen! Beginnen Sie mit der Aggregation RECENTLY RATED COCKTAILS. Diese soll die zuletzt bewerteten Cocktails auf der Timeline des Benutzers anzeigen, sie muss also als aktionsbezogene Aggregation angelegt werden, indem Sie in DATA TO DISPLAY den Aktionstyp MIX auswählen. Als Layout der Aggregation ist LIST gut geeignet, da es mehrere Cocktails übersichtlich anzeigt – alternativ wären aber GALLERY und TABLE ebenfalls geeignet. Als Sortierung kann die Standardeinstellung MOST RECENT RATE belassen werden – damit werden die neuesten Bewertungen zuerst angezeigt. Neben dem AGGREGATION TITLE müssen Sie nun nur mehr passende Beschriftungszeilen (CAPTION LINES) definieren, etwa:

```
{cocktail.title} was rated {rating}
on {start_time}
```

Mit dieser Einstellung enthält jeder Eintrag in der Aggregationsliste einen Verweis auf den Cocktail (`{rating.title}`) und die Höhe der Bewertung (`{rating}`) sowie den Zeitpunkt, zu dem die Bewertung veröffentlicht wurde (`{start_time}`). Die Einstellung für ADD FILTER könnte optional genutzt werden, um etwa nur Bewertungen ab einem gewissen Minium anzuzeigen, während GROUP BY leer bleibt, da es sich um eine aktionsbezogene Aggregation handelt.

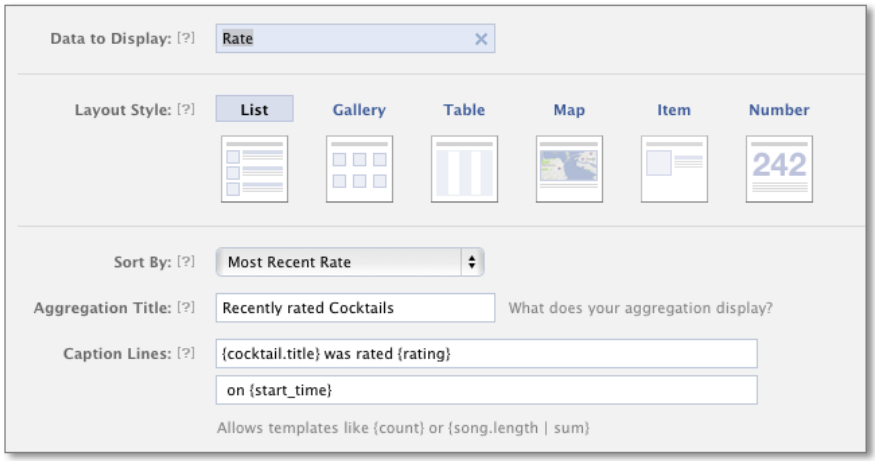


Abbildung 8.40 Einstellungen der aktionsbezogenen Aggregation »Recently rated Cocktails«

Als Nächstes definieren Sie die Aggregation REGULARLY MIXED COCKTAILS. Diese soll die am häufigsten zubereiteten Cocktails eines Benutzers in ansprechender Form präsentieren. Da hierbei eine Aktion (MIX) anhand eines Objekts (COCKTAIL) gruppiert werden soll, handelt es sich um eine objektbezogene Aggregation. Erstellen Sie diese, indem Sie unter DATA TO DISPLAY den zu gruppierenden Objekttyp COCKTAIL und unter ACTION TYPE die Aktion MIX eingeben. Als Layout-Typ eignet sich GALLERY besonders gut für diese Aggregation, da die dargestellten Cocktails die absoluten Highlights des Benutzers darstellen und somit eine größere Bilddarstellung verdienen. Als Sortierung kann hier die Standardeinstellung FAVORITE COCKTAIL dienen, die automatisch jene Cocktails vorreiht, die am häufigsten zubereitet wurden. Alternativ können Sie aber auch mit der Einstellung CUSTOM nach dem Platzhalter COUNT sortieren lassen – das Ergebnis ist dasselbe. Neben dem AGGREGATION TITLE können Sie nun noch die frei wählbaren Textzeilen (CAPTION LINES) definieren:

```
{cocktail.title}
Mixed {count} times
```

Hier bietet es sich an, anzuzeigen, wie oft der jeweilige Cocktail im Zeitraum der Aggregation zubereitet wurde (`{count}`).

Hinweis: Wie bereits erwähnt, wählt Facebook den Betrachtungsseitraum einer Aggregation selbstständig. Aggregationen am Timeline-Profil fassen meist die Daten eines Monats zusammen, während sich am Beginn der Timeline-View einer Anwendung immer eine Aggregation über den gesamten »Lebenszyklus« der Anwendung findet. Sie als Entwickler haben jedenfalls keine Möglichkeit, explizit anzugeben, auf

welches Zeitfenster sich eine Aggregation bezieht, was gerade aufgrund der ansonsten zahlreichen Parallelen zu SQL anfangs verwirrend wirken mag.

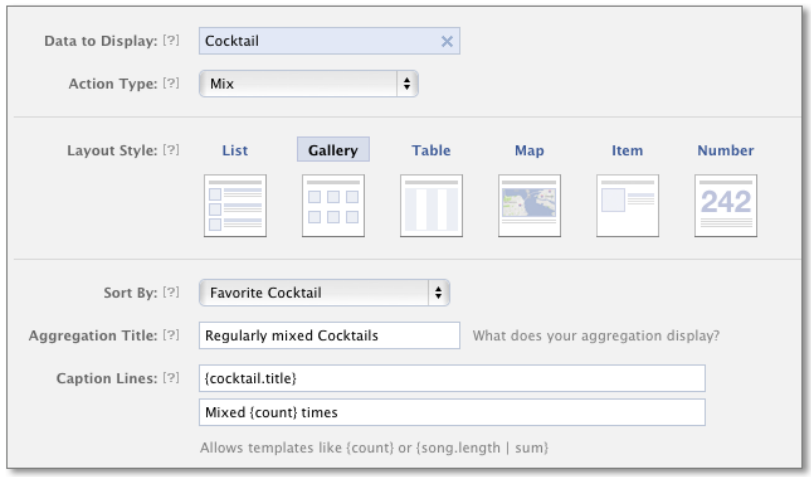


Abbildung 8.41 Einstellungen der objektbezogenen Aggregation »Regularly mixed Cocktails« mit individueller Sortierung

Vorschau auf Aggregationen mit Testdaten

Das Open Graph Dashboard bietet eine praktische Vorschaufunktion, um Aggregationen zu testen, ohne bereits tatsächlich Aktionen zu veröffentlichen. Klicken Sie dazu auf den Link PREVIEW neben der Aggregation!

Damit die Vorschaufunktion genutzt werden kann, müssen Sie unter PREVIEW OBJECTS zuerst für ein vollständiges Set an Testdaten sorgen. Legen Sie dazu zuerst einige Zutaten vom Objekttyp INGREDIENT an, etwa die bereits bekannten Zutaten des »White Russian« – Vodka, Kahlua und Cream. Zur besseren Visualisierung der Vorschau sollten Sie auch URLs zu Thumbnail-Grafiken angeben. Im zweiten Schritt erzeugen Sie ein Preview-Objekt vom Typ COCKTAIL – auch hier geben Sie neben Titel und Beschreibung die URL zum Thumbnail an. Darüber hinaus können Sie hier auch einfach die Zutaten im Array INGREDIENTS hinzufügen (die Auto-Vervollständigung bezieht sich dabei auf alle Preview-Objekte, die Sie bereits erzeugt haben).

Nachdem Sie die Objekte angelegt und verknüpft haben, können Sie unter PREVIEW ACTIONS einige Testaktionen hinterlegen. Da sich das Formular automatisch auf die Aktion der Aggregation – MIX – bezieht, müssen Sie darüber hinaus lediglich den gewünschten Cocktail mittels automatischer Vervollständigung auswählen – alle weiteren Attribute der Aktion können Sie leer bzw. auf den Default-Werten belassen.

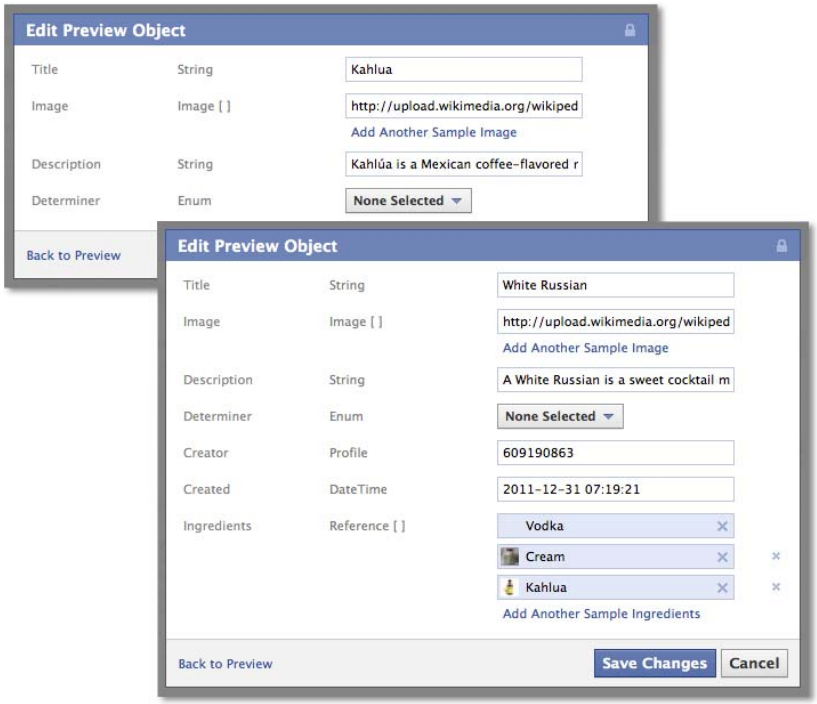


Abbildung 8.42 Eingabe von Vorschauobjekten des Typs »Ingredient« und »Cocktail«

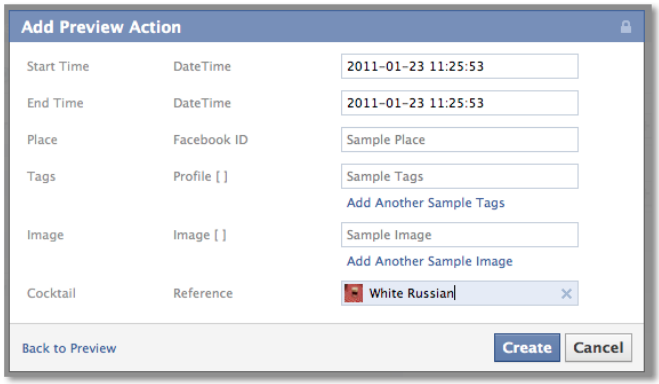


Abbildung 8.43 Eingabe einer Vorschauaktion vom Typ »Mix«

Nachdem Sie einige Preview-Objekte und -Aktionen erfasst haben, wird mit der Einstellung PREVIEW FOR AUTH DIALOG PREVIEW USER die entsprechende Vorschau im Dialog auf Basis der eingegebenen Testdaten angezeigt. Die Preview-Funktion ist damit ein praktisches Werkzeug, um neue Aggregationen vorab zu testen. Alternativ können mit der Einstellung PREVIEW FOR MYSELF aber auch Previews mit den tatsächlich veröffentlichten Aktionen des aktuellen Benutzers angezeigt werden.

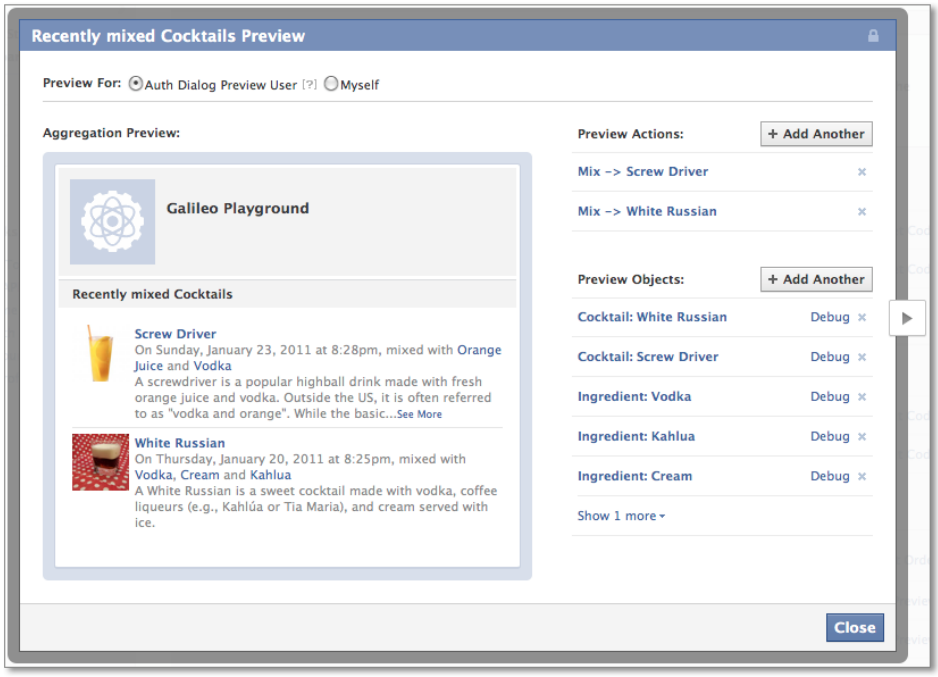


Abbildung 8.44 Vorschau der Aggregation »Recently mixed Cocktails«

Formatieren von Platzhaltern

Mit der Unterstützung von Platzhaltern erlaubt Open Graph die Anzeige von dynamischen Inhalten auf Objekt- oder Aktionsebene in der Darstellung von Aktionen und Aggregationen.

Um ein Attribut eines Objekts darzustellen, wird ein Platzhalter in folgender Notation verwendet:

```
{Objektname.Attributname}
```

Um in der Darstellung von Aktionen und aktionsbezogenen Aggregationen auf die Attribute der Aktion zu referenzieren, genügt die Angabe des Attributnamens im Platzhalter:

```
{Attributname}
```

Der Platzhalter `{start_time}` würde also z. B. den Zeitstempel der Veröffentlichung einer Aktion anzeigen. Attribute, die wiederum einen anderen Objekttyp referenzieren, können mit einem verschachtelten Platzhalter adressiert werden:

```
{Attributname-Objektrefrenz.Attributname}
```

Die Titelfelder der INGREDIENTS unseres Objekttyps COCKTAIL können so etwa mit `{ingredients.title}` referenziert werden.

Besonders numerische und Datumsattribute bedürfen oft einer speziellen Formatierung, um dem Benutzer in angenehm lesbarer Form angezeigt werden zu können. Das Platzhalter-System von Open Graph bietet dazu spezielle Formatierungen (*Formatter*), die mit dem |-Symbol (*Pipe*) an einen Attributnamen angehängt werden können:

```
{Attributname | Formatierung}
```


Hinweis: Open Graph erlaubt die Verkettung von bis zu drei Formatierungen pro Platzhalter!

```
{Attributname | Formatierung1 | Formatierung2 | Formatierung3}
```

Wenn Sie z. B. den Platzhalter `{start_time | date("Y")}` verwenden, wird lediglich die Jahreszahl des Zeitstempels ausgegeben. Die Formatierung `date` kann dabei mit allen Formatierungen aufgerufen werden, die auch in der PHP-Funktion `date()` zur Verfügung stehen:

```
{ Attributname | date} => Wed, 10 Aug 2011 12:07:12
{ Attributname | date("M d, Y")} => Aug 10, 2011
{ Attributname | date("F d, Y")} => August 10, 2011
```

Zusätzlich stehen mit `fb_absolute` und `fb_relative` zwei Formatierungsmöglichkeiten bereit, die den auf Facebook.com üblichen Zeitangaben entsprechen:

```
{ Attributname | date("fb_absolute")} => July 24
{ Attributname | date("fb_relative")} => One week ago
```

Die Formatierung `duration` ist praktisch, um Attribute, die Zeitangaben in Sekunden enthalten, ansprechend in der Form `Stunden:Minuten:Sekunden` darzustellen.

```
{Attributname | duration} => 1:46 (106 Sekunden)
{Attributname | duration} => 1:00:40 (3.640 Sekunden)
```

Die Formatierung `currency` rundet ganzzahlige- oder Float-Werte auf zwei Dezimalstellen. Ein etwaiges Währungssymbol wird dabei jedoch nicht ausgegeben, sondern muss vom Entwickler hinzugefügt werden:

```
{Attributname | currency} => 99.90
```

Mit der Formatierung `sprintf` können darüber hinaus beliebige Formatierungsstrings der PHP-Funktion `sprintf()` auf numerische Open-Graph-Attribute angewandt werden.

```
{Attribut | sprintf("%.2f")} => 1.5
```

Besonders praktisch ist die Formatierung `pluralize` – mit ihr können Sie die Textausgabe abhängig vom Inhalt eines numerischen Attributs formatieren. Dies ist meistens notwendig, um die korrekte Anzeige von Einzahl und Mehrzahl zu gewährleisten, z. B. »1 gehörter Song« vs. »3 gehörte Songs«). `pluralize` wird nach folgendem Schema verwendet:

```
{Attribut | pluralize("Text-1", "Text-2", "Text-3 value")}
```

`Text-1` wird dabei angezeigt, wenn der Wert von `Attribut` 0 ist, `Text-2`, wenn der Wert genau 1 beträgt, und `Text-3` bei allen größeren Werten. Der spezielle Platzhalter `value` fügt dabei den eigentlichen Wert von `Attribut` ein. Um etwa das obige Problem zu lösen, wäre folgender Platzhalter geeignet:

```
{Attribut |
 pluralize("Kein Song gehört",
           "1 Song gehört",
           "value Songs gehört")}
```

Neben den Formatierungshilfen stellen Open-Graph-Platzhalter auch einige mathematische Operationen bereit, die auf numerische Attribute von Aggregationsdaten angewendet werden können. Die Operationen `max`, `min` und `sum` funktionieren ähnlich wie die von SQL bekannten Gruppierungsfunktionen und liefern den höchsten oder kleinsten Wert bzw. die Summe der Werte einer Liste an Attributen.

```
{Attribut | max}
{Attribut | min}
{Attribut | sum}
```

8.2.6 Veröffentlichen von Aktionen

Nachdem Sie nun einige Objekte, Aktionen und Aggregationen definiert und mit der Vorschau und Testdaten getestet haben, ist es höchste Zeit, Ihre ersten Open-Graph-Aktionen auch tatsächlich auf Facebook zu veröffentlichen! Glücklicherweise funktioniert das Veröffentlichen von Aktionen nach dem gleichen Schema wie alle anderen Schreibzugriffe mit der Graph API, nämlich mit einem POST-Zugriff auf einen API-Endpunkt nach folgendem Schema:

```
https://graph.facebook.com/me/AnwendungsNamespace:Aktionstyp
```

Die URL des Endpunkts setzt sich also aus dem Schlüsselwort `me` (referenziert den aktuellen Benutzer) und der mit Namespace und Aktionstyp eindeutig identifizierten Aktion zusammen. Als Parameter übergeben Sie dabei das Access Token des aktuellen Benutzers, das unbedingt die erweiterte Berechtigung `publish_actions` erlauben muss. Als weitere Parameter werden all jene Attribute akzeptiert, die Sie zuvor im Open Graph Dashboard zur jeweiligen Aktion definiert haben.

Im Dashboard finden Sie neben jeder Ihrer Aktionen unter GET CODE einen beispielhaften `curl`-Aufruf der Graph API zur Veröffentlichung der jeweiligen Aktion. Für ihre MIX-Aktion lautet das Beispiel etwa:


```
curl -F 'access_token=...' \
     -F 'cocktail=http://samples.ogp.me/274478052616141' \
        'https://graph.facebook.com/me/digitalbartender:mix'
```

Wie Sie in der Aktionsdefinition von MIX sehen können, erfordert diese Aktion lediglich ein Attribut – im Parameter `cocktail` muss die URL zu jenem Open-Graph-Objekt vom Typ COCKTAIL angegeben werden, auf das sich die Aktion beziehen soll. Sie müssen die Beispiel-URL also durch eine tatsächlich verfügbare URL austauschen:

```
curl
  -F 'access_token=...' \
  -F 'cocktail=http://digitalbartender/cocktails/whiterussian' \
     'https://graph.facebook.com/me/digitalbartender:mix'
```

Den fertigen `curl`-Aufruf können Sie nun auf der Kommandozeile Ihres Rechners ausführen. Das Ergebnis des API-Aufrufs ist ein Array, das im Feld `id` die numerische ID der erfolgreich veröffentlichten Aktion enthält:

```
{"id":"10150641341595864"}
```

Sobald Sie die Aktion veröffentlichen, erscheint auch der passende Eintrag in Ihrem Ticker und auf Ihrer Timeline. *Hinweis:* Solange die Aktion nicht von Facebook im Überprüfungsprozess freigegeben wurde, sind veröffentlichte Einträge nur für Sie und alle anderen eingetragenen Entwickler und Tester der Anwendung sichtbar!

Nachdem das Veröffentlichen auf der Kommandozeile geklappt hat, können Sie im nächsten Schritt versuchen, die Veröffentlichung mit folgendem PHP-Code aus Ihrer Anwendung heraus durchzuführen:

```
include_once("tools.php");
$params = array(
   "access_token" => "...",
   "cocktail" => "http://digitalbartender/cocktails/whiterussian"
);
```

```
$url = "https://graph.facebook.com/me/digitalbartender:mix ";
$response = curl ($url, $params);
print_r(json_decode($response));
```

Listing 8.11 PHP-Code zum Veröffentlichen einer Open-Graph-Aktion

Greifen Sie dabei wieder auf die bewährte Hilfsfunktion `curl()` zurück, und übergeben Sie als Parameter das Access Token und in `cocktail` die URL zum Cocktail, der in der Aktion veröffentlicht werden soll.

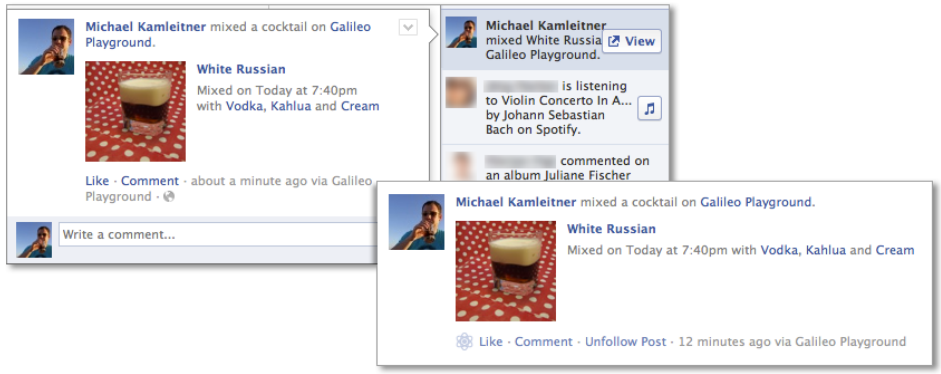


Abbildung 8.45 Ticker- und Timeline-Posting zu einer veröffentlichten Open-Graph-Aktion vom Typ »Mix«

Wenn Sie das JavaScript SDK in Ihrer Anwendung einsetzen, können Sie Open-Graph-Aktionen auch direkt aus dem Browser durch einen Aufruf der SDK-Methode `FB.api()` veröffentlichen:

```
FB.api('/galileo-playground:mix',
       'post',
       { cocktail:'http://digitalbartender/cocktails/whiterussian' });
```

Auffallend ist hierbei, dass die Angabe des Access Tokens entfallen kann, da diese vom SDK automatisch zum Aufruf hinzugefügt ist, wenn der aktuelle Benutzer die Anwendung bereits autorisiert hat.

Wann sollten Open-Graph-Aktionen veröffentlicht werden?

Für Entwickler, die bereits einige Erfahrung mit dem Open Graph Protocol und dem dazugehörenden LIKE-Button gesammelt haben, mag das Veröffentlichen von individuellen Open-Graph-Aktionen ungewohnt erscheinen. Anders als beim LIKE-Button stellt Facebook für individuelle Aktionen kein Interface-Element und keinen Button zur Verfügung, der die Veröffentlichung auslöst.

Im Gegenteil, im neuen Open Graph gilt: Gestaltung und Implementierung von Benutzer-Interface-Elementen obliegen allein dem Anwendungsentwickler! Die Veröffentlichung der passenden Open-Graph-Aktionen erfolgt im »Hintergrund« und wird vom Benutzer – anders als ein Klick auf den LIKE-Button oder ein Popup zum Verfassen eines Wall-Postings – unter Umständen gar nicht wahrgenommen. Facebook fasst diesen Paradigmenwechsel unter dem Stichwort *Frictionless Sharing* zusammen und meint damit, dass das Teilen von Informationen, genauer gesagt, das Veröffentlichen von Interaktionen, möglichst ohne zusätzliches Eingreifen des Benutzers erfolgen soll.

Als Anwendungsentwickler sollten Sie Open-Graph-Aktionen immer dann veröffentlichen, wenn ein Benutzer die entsprechende Aktion in Ihrer Anwendung tatsächlich durchführt – gibt der Benutzer also eine Bewertung für ein Cocktail-Rezept ab, veröffentlichen Sie im Hintergrund die entsprechende Open-Graph-Aktion RATE etc. Der Zugriff auf die Graph API kann dabei entweder serverseitig oder auch direkt am Client über das JavaScript SDK erfolgen.

Beim POST-Zugriff zum Veröffentlichen von Open-Graph-Aktionen können Sie neben den individuellen Aktionsattributen wahlweise auch die Standardattribute, die Facebook bei jedem Aktionstyp zulässt, definieren. Wenn Sie diese Attribute nicht explizit setzen, verwendet Facebook den aktuellen Zeitpunkt als `start_time` und lässt die restlichen Attribute leer.

Attributname	Beschreibung	Format
`start_time`	Startzeitpunkt der Aktion	Zeitstempel im UNIX-Timestamp- oder ISO-8601-Format
`end_time`	Endzeitpunkt der Aktion	Zeitstempel im UNIX-Timestamp- oder ISO-8601-Format
`expires_in`	Dauer der Aktion in Sekunden ab `start_time` – dieses Attribut kann alternativ zu `end_time` angegeben werden.	`integer`
`place`	numerische ID eines Page-Profils mit Geoinformationen (PLACE)	`integer`

Tabelle 8.6 Standardattribute beim Veröffentlichen von Aktionen

Attributname	Beschreibung	Format
tags	Andere Benutzer, die in der Aktion markiert werden sollen. Dieses Attribut wird derzeit noch nicht von Facebook unterstützt.	kommaseparierte Liste an Benutzer-IDs
ref	Tracking-Parameter, der in Facebook Insights zur Analyse der veröffentlichten Aktionen dient	string
scrape	Wenn dieses Attribut auf true gesetzt wird, veranlasst dies Facebook, die Open Graph Tags der mit der veröffentlichten Aktion verknüpften Objekte vor der Veröffentlichung erneut einzulesen. Dies sollte nur passieren, wenn sich der Inhalt der Tags auch tatsächlich verändert hat.	boolean

Tabelle 8.6 Standardattribute beim Veröffentlichen von Aktionen (Forts.)

Auslesen von veröffentlichten Aktionen

Die beim Veröffentlichen von der Graph API zurückgegebene eindeutige, numerische ID der veröffentlichten Aktion kann verwendet werden, um die Attribute und sonstigen Details der Aktion wieder auszulesen. Dazu müssen Sie die ID der Aktion lediglich über die Graph API abfragen:

```
https://graph.facebook.com/AktionsID
```

Das Ergebnis erhält alle Informationen über den veröffentlichenden Benutzer, die Applikation, die zur Veröffentlichung verwendet wurde, etwaige »Likes« oder Kommentare, die zur Aktion veröffentlicht wurden, und natürlich die Attribute der Aktion selbst:

```
{
  "id": "10150641341595864",
  "from": {
    "name": "Michael Kamleitner",
    "id": "609190863"
  },
  "start_time": "2012-01-27T18:40:02+0000",
  "end_time": "2012-01-27T18:40:02+0000",
  "publish_time": "2012-01-27T18:40:02+0000",
  "application": {
    "name": "Galileo Playground",
    "canvas_name": "galileo-playground",
    "namespace": "galileo-playground",
```

```
    "id": "214728715257742"
  },
  "data": {
    "cocktail": {
      "id": "10150530192697860",
      "url": "http://digitalbartender.com/cocktails/whiterussian",
      "type": "galileo-playground:cocktail",
      "title": "White Russian"
    }
  },
  "type": "open_graph_action",
  "likes": {
    "count": 0,
    "can_like": true,
    "user_likes": false
  },
  "comments": {
    "count": 0,
    "can_comment": true
  }
}
```

Listing 8.12 Veröffentlichte Open-Graph-Aktionen können mit all ihren Attributen und Informationen zu Benutzer und Applikation ausgelesen werden.

Der Prüfungsprozess für Open-Graph-Aktionen

Mit der Veröffentlichung des neuen Open Graph hat Facebook auch bekannt gegeben, dass individuelle, von Anwendungsentwicklern definierte Open-Graph-Aktionen in einer manuellen Prüfung (*Approval Process*) von Facebook freigegeben werden müssen.

Was auf den ersten Blick wie eine unnötige, zeitlich schwer einzuschätzende Hürde erscheinen mag, ist tatsächlich gut begründet: Mit der Erteilung der erweiterten Berechtigung `publish_actions` gibt der Benutzer der Anwendung umfangreiche Befugnisse zum automatischen Veröffentlichen von Aktionen. Da Entwickler von der Veröffentlichung dieser Aktionen in Ticker und Timeline-Profilen zu Recht erhöhte Aufmerksamkeit in den Freundeskreisen ihrer Benutzer erwarten, ist es aus Sicht des Plattform-Betreibers Facebook wichtig, Missbrauch zu verhindern. Missbrauch könnte etwa in Form von – absichtlich oder unabsichtlich – missverständlich bzw. falsch benannten Aktionen erfolgen. Stellen Sie sich etwa vor, Sie verwenden ein Open-Graph-Videoportal, und die Anwendung würde in Ihrem Namen jedes betrachtete Video mit der Aktion »empfiehlt« veröffentlichen, obwohl Sie das Video nur betrachtet, nicht aber aktiv bewertet haben (ein besser passendes Aktionsverb wäre

in diesem Fall »betrachtet«). Noch schlimmer wäre der Fall, in dem eine Anwendung ohne Ihr Wissen oder ganz ohne eine von Ihnen ausgelöste Interaktion Open-Graph-Aktionen veröffentlichen würde. Im Leitfaden zum Prüfungsprozess nennt Facebook folgende problematische Nutzung von Aktionen:

- automatisches Veröffentlichen von Aktionen durch einen Timer, also durch Ablaufen einer bestimmten Zeitspanne
- aufeinanderfolgendes Veröffentlichen mehrerer Aktionen aufgrund lediglich einer Benutzerinteraktion
- Verknüpfung von Aktionen mit semantisch unpassenden Objekten
- Verstöße gegen die Facebook Platform Policies, etwa durch »nicht angebrachte« oder anstößige Aktionsverben
- nachträgliches Ändern von Titel, Thumbnail oder anderen Metadaten der durch Aktionen referenzierten Objekte
- schlechte sprachliche oder grammatikalische Darstellung
- Kombinationen von Objektnamen oder Aktionsnamen mit Adjektiven

Als Entwickler sollten Sie Ihre Open-Graph-Anwendung ausführlich und in allen Sprachen testen, bevor Sie den Prüfungsprozess beginnen. Solange Open-Graph-Aktionen nicht von Facebook freigeschaltet wurden, sind die veröffentlichten Aktionen, Timeline-Views, Timeline-Einträge, Aggregationen und Reports ausschließlich für eingetragene Entwickler und Tester der Anwendung sichtbar! Es lohnt sich üblicherweise, in dieser Phase bereits externe Tester zur Prüfung der Anwendung einzuladen!

Sobald Sie die Entwicklung und das Testen der Anwendung erfolgreich abgeschlossen haben, können Sie im Open Graph Dashboard jede von Ihnen definierte Aktion mit dem Link Submit zur Prüfung anmelden. Im darauffolgenden Dialog gibt Ihnen Facebook einen Überblick über die sprachlichen Einstellungen der Aktion, die referenzierten Objekttypen sowie eine Vorschau auf die Aggregationen, die auf die Aktion Bezug nehmen. Kontrollieren Sie diese noch einmal gründlich, bevor Sie im nächsten Schritt im Feld Provide Usage Instructions genaue Angaben machen, wie genau und bei welchen User-Interaktionen die Veröffentlichung der Aktion ausgelöst wird. Formulieren Sie dies so, dass das Facebook-Team die Angaben leicht nachvollziehen kann!

Nach dem Abschicken des Formulars müssen Sie als Entwickler Geduld beweisen – Facebook meldet die erfolgreiche Prüfung per E-Mail und Benachrichtigung. Im Falle einer Ablehnung finden Sie im Open Graph Dashboard eine Begründung für den negativen Ausgang des Prüfungsprozesses.

Da der Prüfungsprozess für Open-Graph-Aktionen zum Zeitpunkt der Entstehung dieses Buches noch nicht von Facebook gestartet wurde, kann an dieser Stelle leider

noch keine Einschätzung darüber gegeben werden, wie lange die Prüfung üblicherweise dauert. Als Entwickler einer Anwendung ist es für Sie aus diesem Grund derzeit noch schwierig, genaue Angaben zum Launch einer Anwendung zu machen. Es bleibt zu hoffen, dass erste Praxiserfahrungen mit Open-Graph-Anwendungen eine bessere Einschätzung des Prüfungsprozesses erlauben.

Wichtige Links:

- *https://developers.facebook.com/docs/opengraph/opengraph-approval/* – offizielle Hinweise zum Prüfungsprozess von Open-Graph-Aktionen
- *https://developers.facebook.com/docs/opengraph/checklist/* – Die Open-Graph-Checklist fasst zusammen, was Sie als Entwickler bei der Verwendung von eigenen Open-Graph-Aktionen beachten sollten.

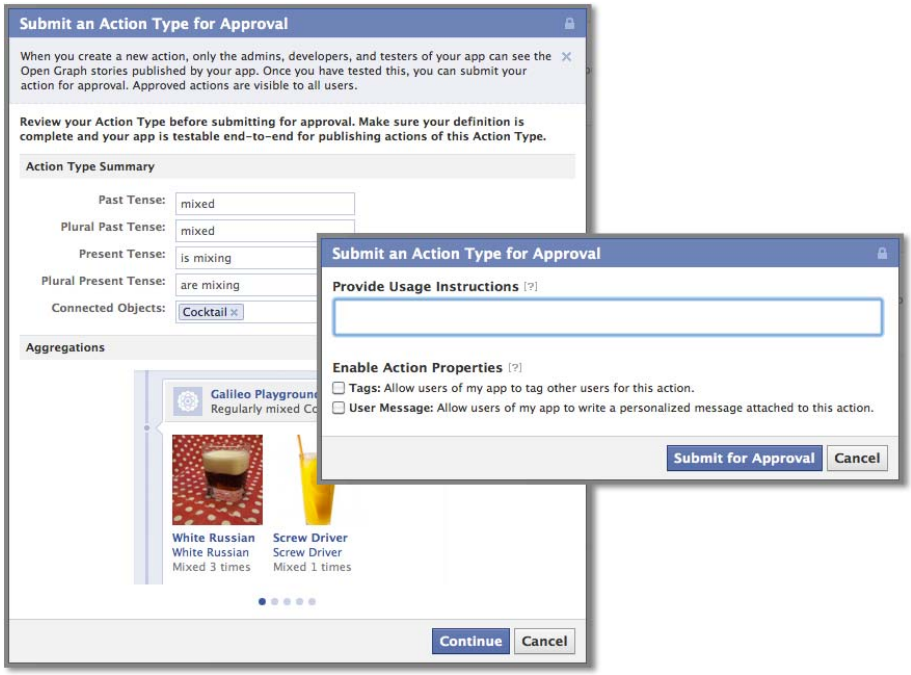


Abbildung 8.46 Alle individuellen Open-Graph-Aktionen müssen von Facebook in einem manuellen Prüfungsprozess freigeschaltet werden.

Löschen und Aktualisieren von veröffentlichten Aktionen

Die beim Veröffentlichen von der Graph API zurückgegebene eindeutige, numerische ID der veröffentlichten Aktion kann verwendet werden, um die Aktion zu einem späteren Zeitpunkt zu löschen oder zu aktualisieren.

Hinweis: Als Entwickler tragen Sie die Verantwortung, alle von Benutzern in Ihrer Anwendung durchgeführten und mittels Open Graph veröffentlichten Aktionen

gegebenenfalls auf Facebook zu löschen oder zu aktualisieren, wenn der Benutzer die ursprüngliche Benutzerinterraktion ändert oder zurücknimmt. Ändert ein Benutzer von DigitalBartender.com die Bewertung eines Cocktails etwa von fünf auf drei Sterne, sollten Sie diese Änderung auch im Open Graph speichern.

Zum Aktualisieren einer Aktion genügt ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/AktionsID`. Als Parameter können Sie dabei alle Attribute genauso angeben wie beim erstmaligen Veröffentlichen – Facebook speichert die Änderungen dann in der veröffentlichten Aktion. Um etwa das Attribut `expires_in` zu aktualisieren, ist folgender Code notwendig:

```
include_once("tools.php");
$params = array(
   "access_token" => "...",
   "expires_in" => "300"
);
$url = "https://graph.facebook.com/10150641341595864";
$response = curl ($url, $params, "POST");
```

Analog dazu genügt ein DELETE-Zugriff auf den Endpunkt *h*`https://graph.facebook.com/AktionsID`, um die veröffentlichte Aktion wieder zu löschen:

```
include_once("tools.php");
$params = array(
   "access_token" => "...",
);
$url = "https://graph.facebook.com/10150641341595864";
$response = curl ($url, $params, "DELETE");
```

Listing 8.13 PHP-Code zum Löschen einer veröffentlichten Open-Graph-Aktion

Veröffentlichen von Kommentaren und »Likes« zu Aktionen

Die Graph API erlaubt auch das Veröffentlichen von Kommentaren und »Likes« im Namen des aktuellen Benutzers. Um ein »Like« zu veröffentlichen, genügt ein POST-Zugriff auf den Endpunkt `https://graph.facebook.com/AktionsID/likes`:

```
include_once("tools.php");
$params = array(
   "access_token" => "...",
);
$url = "https://graph.facebook.com/10150641341595864/likes";
$response = curl ($url, $params, "POST");
```

Listing 8.14 PHP-Code zum Veröffentlichen eines »Likes« zu einer Open-Graph-Aktion

Kommentare können mit einem POST-Zugriff auf den Endpunkt `https://graph.facebook.com/AktionsID/comments` veröffentlicht werden – dabei muss der Text des Kommentars im Parameter `message` übergeben werden:

```
include_once("tools.php");
$params = array(
   "access_token" => "...",
   "message" => "Das ist der Kommentartext ... "
);
$url = "https://graph.facebook.com/10150641341595864/comments";
$response = curl ($url, $params, "POST");
```

Listing 8.15 PHP-Code zum Veröffentlichen eines Kommentars zu einer Open-Graph-Aktion

8.2.7 Der neue OAuth-Dialog

Im Zuge des neuen Open Graph hat Facebook auch die optische Darstellung des OAuth-Dialogs zum Autorisieren von Anwendungen grundlegend überarbeitet. Der neue OAuth-Dialog muss derzeit für Anwendungen manuell mit der Migrationseinstellung ENHANCED AUTH DIALOG aktiviert werden, wird aber im Frühjahr 2012 auch für alle bestehenden Facebook-Anwendungen aktiviert.

Im Vergleich zum bisherigen OAuth-Dialog ist ein Grundprinzip gleich geblieben: Alle Basisberechtigungen sowie die Berechtigungen zum Zugriff auf Profildaten des aktuellen Benutzers und seiner Freunde (siehe auch Abschnitt 2.6, »Erweiterte Zugriffsrechte (›Extended Permissions‹)«) werden in einem ersten Schritt abgefragt und stellen die unveränderliche Voraussetzung zur Installation der Anwendung dar. Alle erweiterten Berechtigungen, wie etwa der Zugriff auf die E-Mail-Adresse oder die Berechtigung zum `offline_access`, werden nach der Autorisierung in einem zweiten Schritt abgefragt und können vom Benutzer optional deaktiviert werden – mit zwei Ausnahmen: Die besonders wichtigen erweiterten Berechtigungen `email` und `publish_actions` zum Veröffentlichen von Open-Graph-Aktionen sind ebenfalls Teil des ersten Autorisierungsschritts. Damit ist es Anwendungsentwicklern von Open-Graph-Applikationen möglich, sicherzustellen, dass alle Anwender auch tatsächlich die Berechtigung zum Veröffentlichen von Aktionen erteilen.

Die Einstellungen zum neuen OAuth-Dialog werden im AUTH DIALOG des Abschnitts SETTINGS in den Einstellungen der Anwendung zusammengefasst:

- In den Feldern HEADLINE und DESCRIPTION können Sie Texte zur Begrüßung und Beschreibung Ihrer Anwendung hinterlegen. Diese werden im ersten Schritt des Autorisierungsdialogs angezeigt und sollten dem Anwender ein gutes Bild vom Zweck Ihrer Anwendung vermitteln.

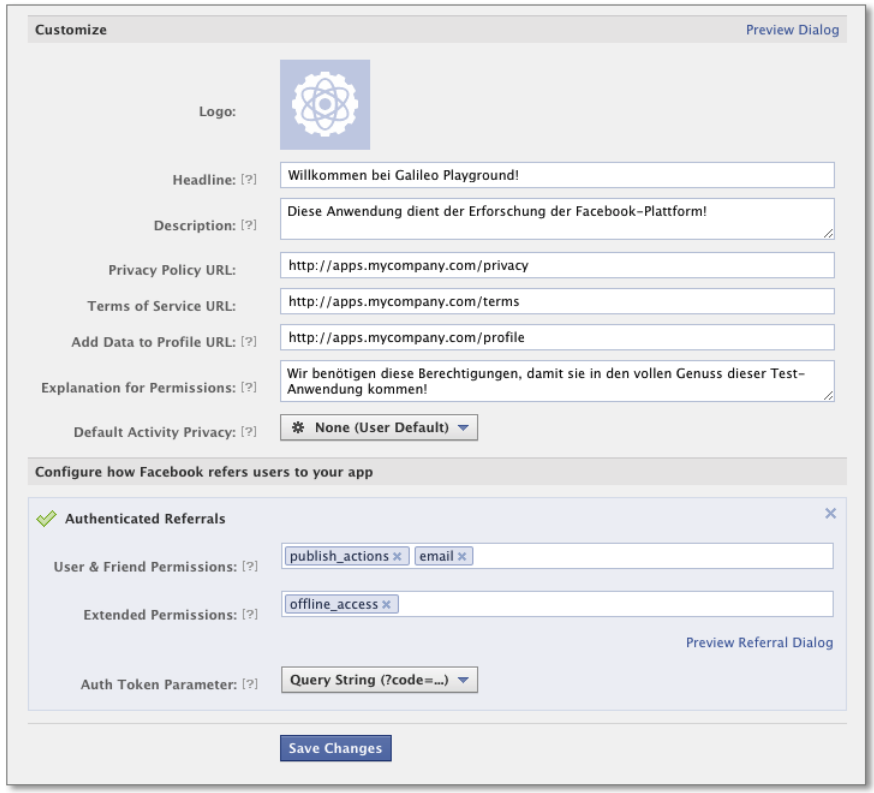


Abbildung 8.47 Einstellungen zum neuen Autorisierungsdialog

- Die Felder PRIVACY POLICY URL und TERMS OF SERVICE URL sollten auf Informationsseiten Ihrer Anwendung verweisen, die ein Statement zur Privatsphäre und zu Ihren Nutzungsbedingungen enthalten. Diese beiden URLs werden in der Fußzeile des Autorisierungsdialogs verlinkt.
- Unter ADD DATA TO PROFILE URL können Sie eine Seite Ihrer Anwendung angeben, auf der Benutzer erfahren, wie sie beginnen können, Open-Graph-Aktionen zu veröffentlichen.
- Der Text im Feld EXPLANATION FOR PERMISSIONS wird im zweiten Schritt des Autorisierungsdialogs, bei der Abfrage der erweiterten Berechtigungen, angezeigt. Mit diesem Text sollten Sie Ihren Anwendern erklären, wozu die Anwendung diese Berechtigungen benötigt.
- DEFAULT ACTIVITY PRIVACY legt fest, mit welchen Privatsphäre-Einstellungen Open-Graph-Aktionen normalerweise veröffentlicht werden. Der Benutzer kann diese Einstellung im ersten Schritt des OAuth-Dialogs abändern, es empfiehlt sich aber, mit NONE (USER DEFAULT) die Default-Einstellung des Benutzers zu übernehmen.

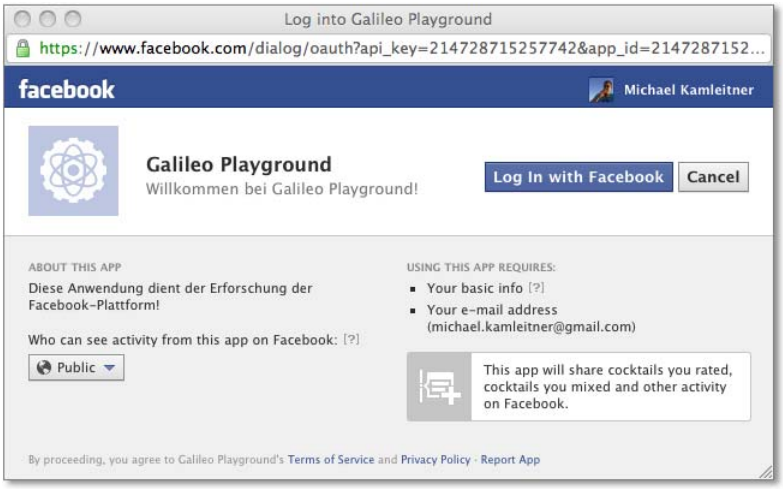


Abbildung 8.48 Der erste Schritt des neuen Autorisierungsdialogs fragt die Rechte für »email«, »publish_actions« und den Zugriff auf Profildaten des Benutzers und seiner Freunde ab.

Hinweis: Die Vergabe der Berechtigung `publish_actions` wird im ersten Schritt des Autorisierungsdialogs durch Anzeige des grauen Plus-Symbols dargestellt (THIS APP WILL SHARE ...). Dabei werden die im Open Graph Dashboard definierten Objekte und Aktionen exemplarisch aufgelistet, um dem Benutzer einen besseren Eindruck von den zu erwartenden Veröffentlichungen zu geben. Haben Sie noch keine Objekte, Aktionen oder Aggregationen definiert, ignoriert der OAuth-Dialog die Berechtigung `publish_actions` vollständig.

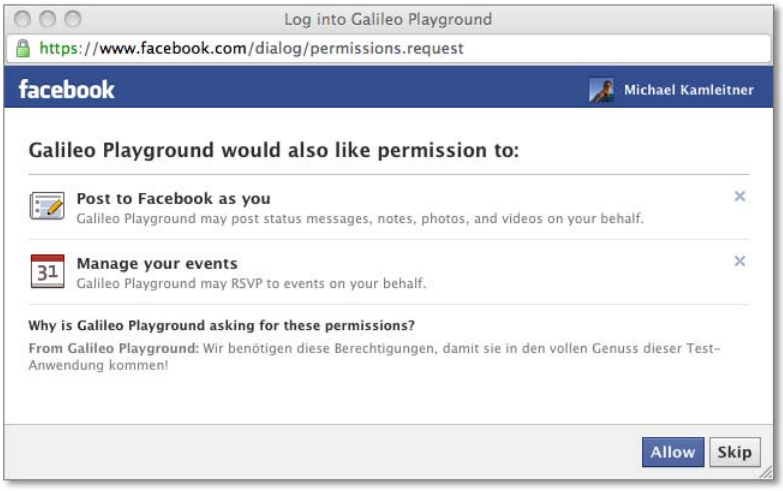


Abbildung 8.49 Der zweite Schritt des Autorisierungsdialogs fragt alle restlichen erweiterten Berechtigungen ab und erlaubt dem Benutzer, diese wahlweise zu deaktivieren.

Authenticated Referrals

Mit der Einstellung Authenticated Referrals bietet Ihnen Facebook eine Möglichkeit zu erzwingen, dass alle Zugriffe auf das Application Canvas über die URL *http://apps.facebook.com/NameSpace* nur für autorisierte Benutzer möglich sind. Anwender, die Ihre Anwendung noch nicht installiert haben, werden von Facebook automatisch auf den OAuth-Dialog weitergeleitet. Diese Einstellung kann Ihnen helfen, die Benutzerbasis Ihrer Anwendung rasch zu vergrößern, hat aber den Nachteil, dass Anwender, die nicht bereit sind, Ihre Applikation zu autorisieren, gänzlich ausgesperrt bleiben. In den Einstellungen des Autorisierungsdialogs können Sie für die Einstellung Authenticated Referrals noch folgende Konfigurationen vornehmen:

- User & Friend-Permissions – Geben Sie hier die Berechtigungen zum Zugriff auf die Profildaten des Benutzers und seiner Freunde an, die Sie von Ihren Benutzern auf jeden Fall verlangen möchten. Auch `email` und `publish_actions` müssen hier angeführt werden, da sie im ersten Schritt des Dialogs abgefragt werden.
- Extended Permissions – Geben Sie hier alle erweiterten Berechtigungen an, die Sie von Ihren Benutzern fordern.
- Auth Token Parameter – Wählen Sie hier aus, ob Sie das Access Token als URL-Fragment (etwa zur weiteren Verwendung mittels JavaScript) erhalten möchten oder ob Facebook im Parameter `code` ein Code-Wort übergeben soll, mit dem Sie das eigentliche Access Token serverseitig beziehen können (siehe Abschnitt 2.3, »Clientseitige Authentifikation«).

Achtung: Auch wenn die Einstellung Authenticated Referrals aktiviert ist, müssen Sie beachten, dass Zugriffe direkt auf die Canvas URL (z. B. *http://apps.mycompany.com*) Ihrer Anwendung erfolgen können – hier kann Facebook keine Autorisierung erzwingen, Sie müssen daher – etwa per JavaScript – selbst für einen Redirect auf das Application Canvas sorgen!

Hinweis: Die Einstellung Authenticated Referrals wirkt sich nicht auf die Darstellung von Canvas-Applikationen aus, die als Tab/Reiter in ein Seitenprofil integriert wurden – diese werden weiterhin auch für Benutzer angezeigt, die die Anwendung nicht autorisiert haben!

8.2.8 Vordefinierte Objekt- und Aktionstypen

Mit der Veröffentlichung des neuen Open Graphs hat Facebook Anwendungsentwicklern auch ein Set an vordefinierten (*Built in*) Objekt- und Aktionstypen zur Verfügung gestellt. Die vordefinierten Objekte und Aktionen decken vor allem die wichtigen Anwendungsbereiche News, Musik und Video ab.

Hinweis: Obwohl die vordefinierten Objekte und Aktionen inhaltlich vorgegeben sind, verlangt Facebook auch von Anwendungen, die sie benutzen möchten, das erfolgreiche Durchlaufen des manuellen Prüfungsprozesses. Dieser wird genauso wie für individuelle Aktionen im Open Graph Dashboard mit dem Link SUBMIT gestartet.

Vordefinierte Aktionen und Objekte werden wie gewohnt im Open Graph Dashboard hinzugefügt – wird dabei als Objekt- bzw. Aktionsname ein vordefinierter Name eingegeben, verwendet Facebook automatisch die vordefinierten Objekte bzw. Aktionen.

Eine Besonderheit von vordefinierten Aktionen findet sich in der Anzeige auf Timeline-Profilen und Newsfeeds: Facebook aggregiert hier alle veröffentlichten Aktionen aus den Bereichen Musik, News und Video in Boxen, egal, über welche Applikation sie veröffentlicht wurden.

Wichtige Links:

- *https://developers.facebook.com/docs/opengraph/objects/builtin/*
  offizielle Dokumentation der vordefinierten Objekttypen
- *https://developers.facebook.com/docs/opengraph/actions/builtin/*
  offizielle Dokumentation der vordefinierten Aktionstypen

Das »Article«-Objekt und die Aktion »Read«

Der Objekttyp Article ist für die Kennzeichnung von Webseiten gedacht, die Textinhalte wie etwa einen Blog-Beitrag, einen Nachrichten-Artikel oder sonstige Texte enthalten. Die Verwendung des Typs Article ermöglicht die Veröffentlichung der vom Benutzer gelesenen Artikel mit der vordefinierten Aktion READ.

Objekte vom Typ Article müssen im Attribut `og:type` den Wert `article` enthalten und können neben den üblichen Open Graph Tags für Titel, Beschreibungstext, Thumbnail etc. die folgenden spezifischen Attribute enthalten:

Attributname	Beschreibung
`article:published_time`	Zeitstempel im ISO-8601-Format, der den Zeitpunkt der ersten Veröffentlichung des Artikels enthält
`article:modified_time`	Zeitstempel im ISO-8601-Format, der den Zeitpunkt der letzten Aktualisierung des Artikels enthält

Tabelle 8.7 Individuelle Open-Graph-Attribute des Objekttyps »Article«

Attributname	Beschreibung
article:author	URL, die auf ein Open-Graph-Objekt des Typs profile verweist, das ein Profil des Autors enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
article:tag	Enthält ein einzelnes Schlagwort zum Artikel. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
article:section	Enthält die Content-Kategorie des Artikels (z. B. »Sport«, »Nachrichten« etc.)
article:expiration_time	Zeitstempel im ISO-8601-Format, der optional den Zeitpunkt angibt, zu dem ein Artikel seine Relevanz verliert. Facebook hört ab diesem Zeitpunkt auf, den Artikel in Ticker, Timeline etc. hervorzuheben.

Tabelle 8.7 Individuelle Open-Graph-Attribute des Objekttyps »Article« (Forts.)

Der folgende Beispiel-Code zeigt die Open Graph Tags eines Article-Objekts (auf die Darstellung der üblichen Attribute für Thumbnail, Beschreibung etc. wurde aus Gründen der Übersichtlichkeit verzichtet):

```
<meta property="og:title"
      content="W. Basketball: Trojans trounced"/>
<meta property="og:type" content="article"/>
<meta property="og:url"
      content="http://www.stanforddaily/article/3434534"/>
<meta property="article:published_time" content="2010-01-24" />
<meta property="article:author"
      content="http://www.stanforddaily.com/author/ivynguyen/" />
<meta property="article:tag" content="Jeanette Pohlen" />
<meta property="article:tag"
      content="Stanford women's basketball" />
<meta property="article:tag" content="USC" />
<meta property="article:section" content="Sports">
<meta property="article:expiration_time" content="2010-01-26">
```

Listing 8.16 Beispiel eines »Article«-Objekts

Das Article-Objekt ist besonders in Kombination mit der vordefinierten Aktion READ interessant. Diese Aktion sollte immer dann veröffentlicht werden, wenn ein Benutzer einen Artikel gelesen, also die entsprechende Webseite im Browser aufgerufen

hat. Die von Facebook unterstützten Open Graph Showcases führen die Veröffentlichung automatisch, also ohne Zutun des Benutzers, im Hintergrund aus, es wäre aber auch denkbar, dies an eine explizite Benutzeraktion zu knüpfen.

Zum Veröffentlichen der READ-Aktion ist ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/me/news.reads` notwendig, wobei im Parameter `article` die URL des Artikels übergeben werden muss:

```
curl
  -F 'access_token=...' \
  -F 'article=http://www.stanforddaily/article/3434534' \
     'https://graph.facebook.com/me/news.reads'
```


READ-Aktionen, die von Benutzern über verschiedene Anwendungen veröffentlicht werden, aggregiert Facebook im Newsfeed und in der Box NEWS am Timeline-Profil des Benutzers.

Abbildung 8.50 Anzeige von aggregierten »Read«-Aktionen im Newsfeed

Das Videoobjekt und die Aktion »Watches«

Mit dem Videoobjekt stellt Facebook mehrere spezifische Objekttypen für Open-Graph-Objekte bereit, die Bewegtbilddaten enthalten. Dies können etwa Filme, TV-Shows oder Serien-Episoden sein.

Videoobjekte müssen dementsprechend im Attribut `og:type` den Wert `video.movie`, `video.episode`, `video.tv_show` oder `video.other` enthalten und können neben den üblichen Open Graph Tags für Titel, Beschreibungstext, Thumbnail etc. folgende spezifische Attribute besitzen:

Attributname	Beschreibung
`video:release_date`	Zeitstempel im ISO-8601-Format, der den Zeitpunkt der ersten Veröffentlichung des Videos enthält
`video:duration`	Spielzeit des Videos in Sekunden

Tabelle 8.8 Individuelle Open-Graph-Attribute des Objekttyps »Video«

Attributname	Beschreibung
video:actor	URL, die auf ein Open-Graph-Objekt des Typs profile verweist, das ein Profil des Schauspielers/Darstellers enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
video:director	URL, die auf ein Open-Graph-Objekt des Typs profile verweist, das ein Profil des Regisseurs enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
video:writer	URL, die auf ein Open-Graph-Objekt des Typs profile verweist, das ein Profil des Autors enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
video:tag	Enthält ein einzelnes Schlagwort zum Video. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
video:upc	UPC-Code des Videos
video:series	Bei Objekten vom Typ video.episode kann hier die URL eines Open-Graph-Objekts vom Typ tv_show angegeben werden. Dieses Objekt sollte Informationen zur Serie enthalten, zu der das Video gehört.

Tabelle 8.8 Individuelle Open-Graph-Attribute des Objekttyps »Video« (Forts.)

Der folgende Beispiel-Code zeigt die Open Graph Tags eines Videoobjekts (auf die Darstellung der üblichen Attribute für Thumbnail, Beschreibung etc. wurde aus Gründen der Übersichtlichkeit verzichtet):

```
<meta property="og:title" content="The Frog"/>
<meta property="og:type" content="video.movie"/>
<meta property="og:url" content="http://www.imdb.com/movie/345"/>
<meta property="video:release_date" content="1998-04-23" />
<meta property="video:duration" content="1560" />
<meta property="video:actor"
      content="http://www.imdb.com/name/nm0000632/" />
<meta property="video:actor"
      content="http://www.imdb.com/name/nm0000506/" />
<meta property="video:director"
      content="http://www.imdb.com/name/nm0000452/" />
```

```
<meta property="video:writer"
      content="http://www.imdb.com/name/nm0000234/" /
<meta property="video:writer"
      content="http://www.imdb.com/name/nm0000564/" />
<meta property="video:tag"  content="Stand Up Comedian" />
<meta property="video:tag"  content="Friendship" />
<meta property="video:tag"  content="New York Yankees" />
<meta property="video:series"
      content="http://www.imdb.com/name/nm0000123/">
```

Listing 8.17 Beispiel eines Videoobjekts

Ähnlich wie READ kann die vordefinierte Aktion WATCHES benutzt werden, um zu veröffentlichen, wenn ein Benutzer sich ein Video anschaut.

Zum Veröffentlichen der WATCHES-Aktion ist ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/me/videos.watches` notwendig, wobei im Parameter `movie` die URL des angesehenen Videos übergeben werden muss:

```
curl
  -F 'access_token=...' \
  -F 'movie=http://www.imdb.com/movie/345' \
     'https://graph.facebook.com/me/videos.watches'
```

WATCHES-Aktionen, die von Benutzern über verschiedene Anwendungen veröffentlicht werden, aggregiert Facebook im Newsfeed und in der Box VIDEOS am Timeline-Profil des Benutzers.

Abbildung 8.51 Anzeige von aggregierten »Watches«-Aktionen am Timeline-Profil

Die Musikobjekte Song, Album, Playlist und Radiostation und die Aktionen »Listen« und »Create«

Facebook stellt für den Anwendungsbereich »Musik« gleich mehrere vordefinierte Objekt- und Aktionstypen bereit, um einzelne Musikstücke, Musikalben, Playlisten und Radiostationen im Open Graph abzubilden.

Einzelne »Songs« werden mit einem `og:type` des Wertes `music.song` ausgezeichnet und können folgende zusätzliche Attribute enthalten:

Attributname	Beschreibung
`music:musician`	URL, die auf ein Open-Graph-Objekt des Typs `profile` verweist, das ein Profil des Interpreten/Künstlers enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:album`	URL, die auf ein Open-Graph-Objekt des Typs `music:album` verweist, das ein Profil des Albums enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:album:disc`	Gibt an, auf welcher CD eines Albums sich das Stück befindet (Default: 1).
`music:album:track`	Gibt die Nummer des Stücks innerhalb eines Albums an.
`music:duration`	Spielzeit des Musikstücks in Sekunden

Tabelle 8.9 Individuelle Open-Graph-Attribute des Objekttyps »Songs«

Der folgende Beispiel-Code zeigt die Open Graph Tags eines Song-Objekts (auf die Darstellung der üblichen Attribute für Thumbnail, Beschreibung etc. wurde aus Gründen der Übersichtlichkeit verzichtet):

```
<meta property="og:title" content="Bad Moon Rising"/>
<meta property="og:type" content="music.song"/>
<meta property="music:musician"
      content="http://www.amazon.com/Creedence-Clearwater-Revival/e/
               B000APTGX2/digital/ref=ntt_mp3_rdr">
<meta property="music:album"
      content="http://www.amazon.com/Creedence-Clearwater-Revival/e/
               B000APTGX2/digital/ref=ntt_mp3_rdr">
<meta property="music:album:track" content="4">
<meta property="music:duration" content="139">
```

Listing 8.18 Beispiel eines Song-Objekts

Alben werden mit einem `og:type` des Wertes `music.album` gekennzeichnet und können folgende zusätzliche Attribute enthalten:

Attributname	Beschreibung
`music:song`	URL, die auf ein Open-Graph-Objekt des Typs `music:song` verweist, das ein Profil eines Songs enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:musician`	URL, die auf ein Open-Graph-Objekt des Typs `profile` verweist, das ein Profil des Album-Interpreten/-Künstlers enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:release_date`	Zeitstempel im ISO-8601-Format, der den Zeitpunkt der ersten Veröffentlichung des Albums enthält
`music:song:disc`	Gibt an, auf welcher CD eines Albums sich das Stück befindet (Default: 1). Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:song:track`	Gibt die Nummer des Stücks innerhalb eines Albums an. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.

Tabelle 8.10 Individuelle Open-Graph-Attribute des Objekttyps »Album«

Der folgende Beispiel-Code zeigt die Open Graph Tags eines Albumobjekts (auf die Darstellung der üblichen Attribute für Thumbnail, Beschreibung etc. wurde aus Gründen der Übersichtlichkeit verzichtet):

```
<meta property="og:title" content="Chronicle: 20 Greatest Hits"/>
<meta property="og:type" content="music.album"/>
<meta property="music:release_date" content="1998-04-23">
<meta property="music:musician"
      content="http://www.amazon.com/Creedence-Clearwater-Revival/e/
               B000APTGX2/digital/ref=ntt_mp3_rdr ">
<meta property="music:song"
      content="http://www.amazon.com/gp/product/B000UBLXG4/">
<meta property="music:song:track" content="1">
<meta property="music:song"
      content="http://www.amazon.com/gp/product/B000UBLXG4/">
<meta property="music:song:track" content="2">
...
```

Listing 8.19 Beispiel eines Albumobjekts

»Playlist«-Objekte dienen der Zusammenfassung einer beliebigen Anzahl an Tracks. Sie werden mit einem `og:type` des Wertes `music.playlist` gekennzeichnet und können folgende zusätzliche Attribute enthalten:

Attributname	Beschreibung
`music:song`	URL, die auf ein Open-Graph-Objekt des Typs `music:song` verweist, das ein Profil eines Songs enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:song:track`	Gibt die Nummer des Stücks innerhalb der Playlist an. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`music:creator`	URL, die auf ein Open-Graph-Objekt des Typs `profile` verweist, das ein Profil des Erstellers der Playlist enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.

Tabelle 8.11 Individuelle Open-Graph-Attribute des Objekttyps »Playlist«

Mit dem `og:type` des Wertes `music.radio_station` steht ein eigener Objekttyp zur Abbildung von Radiosendern mit folgenden erweiterten Attributen bereit:

Attributname	Beschreibung
`music:creator`	URL, die auf ein Open-Graph-Objekt des Typs `profile` verweist, das ein Profil des Erstellers des Radiosender-Objekts enthält. Dieses Attribut ist ein Array und kann daher mehrfach angegeben werden.
`og:audio`	URL zum Live-Audio-Stream des Radiosenders

Tabelle 8.12 Individuelle Open-Graph-Attribute des Objekttyps »Radiostation«

Ähnlich wie READ und WATCHES können angehörte Musikstücke mit der vordefinierten Aktion LISTENS am Timeline-Profil eines Benutzers veröffentlicht werden.

Zum Veröffentlichen der LISTENS-Aktion genügt ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/me/music.listens`, wobei im Parameter `song` die URL des gehörten Songs übergeben werden muss:

```
curl
  -F 'access_token=...' \
  -F 'song=http://www.amazon.com/gp/product/B000UBLXG4/' \
     'https://graph.facebook.com/me/music.listens'
```

LISTENS-Aktionen, die von Benutzern über verschiedene Anwendungen veröffentlicht werden, aggregiert Facebook im Newsfeed und in der Box MUSIC am Timeline-Profil des Benutzers.

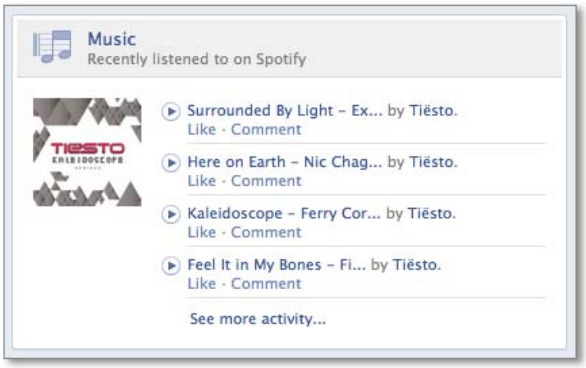


Abbildung 8.52 Anzeige von aggregierten »Listens«-Aktionen am Timeline-Profil

Der zweite vordefinierte Aktionstyp für den Anwendungsbereich »Musik« erlaubt die Veröffentlichung von Playlisten, die ein Benutzer angelegt hat. Dazu ist ein POST-Zugriff auf den API-Endpunkt `https://graph.facebook.com/me/music.playlists` notwendig, wobei im Parameter `playlist` die URL der veröffentlichten Playlist übergeben werden muss:

```
curl
  -F 'access_token=...' \
  -F 'playlist=http://www.amazon.com/playlist/...' \
     'https://graph.facebook.com/me/music.playlists
```

Hinweis: Aus lizenzrechtlichen Gründen behandelt Facebook die vordefinierten Aktionstypen LISTENS und PLAYLISTS besonders restriktiv. So war das Veröffentlichen von solchen Aktionen während der Entstehung dieses Buches nicht möglich – selbst für eingetragene Entwickler und Tester der eigenen Anwendung. Auch wenn es hierzu keine offiziellen Angaben seitens Facebook gibt, ist doch zu vermuten, dass für die Nutzung dieser Aktionen eine gesonderte Prüfung durch Facebook vorgesehen ist.

8.2.9 Social Plugins für Open Graph

Gemeinsam mit dem neuen Open Graph hat Facebook einige neue Plugins eingeführt, die speziell auf die Eigenschaften des Open Graphs abzielen. *Add-to-Timeline* und *Recommendations-Bar* ermöglichen Ihnen eine besonders einfache Integration von Open-Graph-Funktionen in bestehende Websites. Vor allem die Recommendations-Bar ist für die Betreiber von News-Portalen und Blogs interessant, da sie das

einfache Veröffentlichen der vordefinierten Open-Graph-Aktion READ in Bezug auf Artikel und Textbeiträge erlaubt.

Darüber hinaus wurden die bestehenden Plugins Activity Feed, Recommendations-Box und Facepile um die Möglichkeit erweitert, Open-Graph-Aktionen anzuzeigen und zu aggregieren. Haben diese Plugins schon im Open Graph Protocol wertvolle Möglichkeiten zur *Instant Personalization*, also dem benutzerabhängigen Anzeigen von passendem Content, geboten, können sie mit der Unterstützung von individuellen Open-Graph-Aktionen nun noch besser ihr Potenzial ausschöpfen.

Die Add-to-Timeline-Box

Die ADD-TO-TIMELINE-Box funktioniert ähnlich wie das in Abschnitt 6.10, »Der ›Log-In‹-Button«, beschriebene Plugin. Es erlaubt Benutzern einer Website, die auf ihr eingebundene Facebook-Applikation auf einfache Art und Weise mit der erweiterten Berechtigung `publish_actions` zu autorisieren. Ein Klick auf den Button öffnet den gewohnten OAuth-Autorisierungsdialog von Facebook. Nach Autorisierung der Anwendung können Sie sofort beginnen, Open-Graph-Aktionen im Namen des Benutzers – etwa über das JavaScript SDK – zu veröffentlichen.

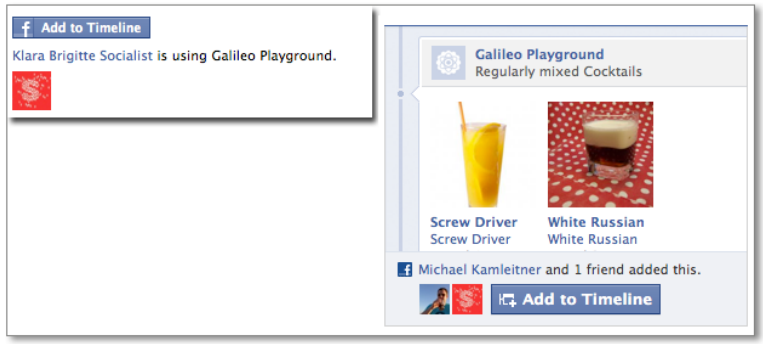


Abbildung 8.53 Das Add-to-Timline-Plugin in Button- (links) und Box-Ansicht (rechts)

Die Einbindung der ADD-TO-TIMELINE-Box kann mittels HTML5, XFBML und iFrame erfolgen:

```
// Einbindung als HTML5-Tag
<div class="fb-add-to-timeline" data-show-faces="true"></div>

// Einbindung als XFBML-Tag
<fb:add-to-timeline show-faces="true"></fb:add-to-timeline>

// Einbindung als iFrame
<iframe src="www.facebook.com/plugins/add_to_timeline.php
   ?show-faces=true&amp;
```

```
mode=box&amp;
appId=APP_ID"
scrolling="no" frameborder="0"
style="border:none; overflow:hidden;"
allowTransparency="true"></iframe>
```

Listing 8.20 Einbindung der Add-to-Timeline-Box

Die ADD-TO-TIMELINE-Box bezieht sich automatisch auf jene Anwendung, mit der das JavaScript SDK mittels `FB.init()` initialisiert wurde, und kann mit folgenden Parametern detaillierter konfiguriert werden:

Name	Typ	Beschreibung
`show-faces`	`boolean`	Legt fest, ob unter dem Plugin die Profilbilder von Freunden angezeigt werden sollen, die die Anwendung bereits autorisiert haben.
`mode`	`string`	Der Standard-Anzeigemodus `box` zeigt eine Vorschau der auf der Timeline veröffentlichten Aktionen, während im Modus `button` lediglich der Button zum Autorisieren der Anwendung angezeigt wird.

Tabelle 8.13 Weitere Parameter des Add-to-Timeline-Plugins

Die Recommendations-Bar

Die *Recommendations-Bar* ist ein besonders praktisches Social Plugin für Betreiber von Nachrichten-Portalen und Blogs, kombiniert sie doch die Funktionalitäten von LIKE-Button und Recommendations-Box mit der Fähigkeit, die Open-Graph-Aktion READ zu veröffentlichen. Damit ist sie das ideale Tool, wenn Sie die Leser Ihrer Website zur Interaktion oder zum Weiterklicken ermutigen möchten!

Die Recommendations-Bar wird beim Laden der Seite vom JavaScript SDK am unteren linken oder rechten Rand der einbindenden Website eingeblendet. Zu Beginn wird dabei lediglich ein kleiner LIKE-Button angezeigt.

Wenn der Benutzer die Seite bis zu einem bestimmten Punkt im Dokument nach unten gescrollt ist und gleichzeitig eine bestimmte Zeitspanne verstrichen ist (das genaue Verhalten können Sie als Entwickler konfigurieren), expandiert das SDK den LIKE-Button zur vollen Recommendations-Bar. Diese zeigt Empfehlungen weiterer Artikel auf der aktuellen Website an, wobei Artikel, die von Freunden oder anderen Benutzern mit dem LIKE-Button ausgezeichnet wurden, bevorzugt werden.

Abbildung 8.54 Website mit eingebundener Recommendations-Bar – der zuerst klein dargestellte »Like«-Button wird nach Verstreichen einer bestimmten Zeitspanne zur Recommendations-Bar expandiert.

Doch das interessanteste Feature der Recommendations-Bar verbirgt sich hinter der Schaltfläche READING IS OFF – ein Klick auf diesen Button öffnet den OAuth-Dialog zur Autorisierung der mit der Recommendations-Bar verknüpften Facebook-Anwendung, wobei ähnlich wie bei der ADD-TO-TIMELINE-BOX auch gleich die `publish_actions`-Berechtigung vom Benutzer eingefordert wird (erkennbar an dem Hinweis THIS APP WILL SHARE ARTICLES YOU READ AND OTHER ACTIVITY ON FACEBOOK).

Abbildung 8.55 Beim Klick auf »Reading is Off« öffnet die Recommendations-Bar den OAuth-Autorisierungsdialog.

Hinweis: Zum Zeitpunkt der Entstehung dieses Buches war die Recommendations-Bar noch nicht öffentlich verfügbar, sondern wurde nur bei Besuchern angezeigt, die als Entwickler oder Tester der im JavaScript SDK verknüpften Facebook-Anwendung eingetragen waren! Auch wenn dies noch nicht endgültig feststeht, wird eine Prüfung von Applikationen, die die Recommendations-Bar nutzen möchten, durch Facebook höchstwahrscheinlich notwendig sein.

Bestätigt der Benutzer die Autorisierung der Anwendung, werden alle auf der Website künftig gelesenen Artikel mit der Open-Graph-Aktion READ automatisch auf dem Timeline-Profil des Benutzers veröffentlicht. Mit einem weiteren Klick auf die nunmehr angezeigte Schaltfläche READING IS ON kann der Benutzer dies fallweise aber auch wieder deaktivieren.

Abbildung 8.56 Darstellung der durch die Recommendations-Bar veröffentlichten »Read«-Aktionen im Activity Log des Benutzers

Da die verwendete Open-Graph-Aktion READ bzw. `news.reads` zu den vordefinierten Aktionstypen von Facebook gehört, muss sie im Open Graph Dashboard nicht extra aktiviert werden. Beachten Sie, dass sich `news.reads` ausschließlich auf Objekte vom Typ `article` anwenden lässt, Sie müssen also dafür sorgen, dass die einzelnen Artikel-Seiten Ihrer Website das entsprechende `og:type`-Tag aufweisen. Darüber hinaus sollten Sie natürlich auch die Open Graph Tags für Titel, Thumbnail, Beschreibungstext etc. optimieren, um eine ideale Darstellung in Ticker und Timeline zu gewährleisten.

Abbildung 8.57 Darstellung einer durch die Recommendations-Bar veröffentlichten »Read«-Aktion im Ticker

Die Recommendations-Bar kann derzeit ausschließlich mittels XFBML- oder HTML5-Tag eingebunden werden. Die Anwendung, auf die sich das Plugin bei der Autorisierung bezieht, ist dabei immer automatisch jene bei der Initialisierung des SDKs mittels FB.init() angegebene.

```
// Einbindung als HTML5-Tag
<div class="fb-recommendations-bar"
  data-href="http://die.socialisten.at/2012/01/graphinspector/"
  data-read-time="10"
  data-site="die.socialisten.at">
</div>

// Einbindung als XFBML-Tag
<fb:recommendations-bar
  href="http://die.socialisten.at/2012/01/graphinspector/"
  read_time="10"
  site="die.socialisten.at">
</fb:recommendations-bar>
```

Listing 8.21 Einbindung der Recommendations-Bar

Die folgende Tabelle zeigt alle weiteren Parameter, mit der die Recommendations-Bar konfiguriert werden kann:

Name	Typ	Beschreibung
`site`	`string`	Kommaseparierte Liste von Domains, auf die sich das Plugin beziehen soll. Der Wert dieses Attributs muss mit der Einstellung App Domain in der Developer-App übereinstimmen.
`href`	`string`	URL der Webseite, auf die sich die Recommendations-Box und deren Like-Button und Read-Aktion beziehen sollen.
`action`	`string`	Gibt an, welches Verb im Like-Button der Recommendations-Bar angezeigt werden soll. Derzeit nur wählbar zwischen `like` und `recommend`.
`trigger`	`string`	Event, das nach dem Verstreichen der in `read_time` angegebenen Zeit zum Ausklappen der Recommendations-Bar führt

Tabelle 8.14 Parameter der Recommendations-Bar

Name	Typ	Beschreibung
trigger (Forts.)		onvisible – Event tritt ein, wenn der Benutzer über die Position des XFBML-/HTML5-Tags der Recommendations-Bar hinausscrollt. X% – Event tritt ein, wenn der Benutzer über mindestens x Prozent der Gesamthöhe der Webseite hinausscrollt. manual – Das Aufklappen der Recommendations-Bar erfolgt manuell durch Aufrufen der SDK-Methode FB.XFBML.RecommendationsBar.markRead(href);.
read_time	integer	Sekunden, die verstreichen müssen, bis die Recommendations-Bar aufklappt (Default: 30, Minimum 10). Unabhängig von der verstrichenen Zeit muss außerdem das in trigger angegebene Event eingetreten sein!
side	string	Gibt an, ob die Recommendations-Bar am unteren linken (left) oder unteren rechten (right, Default) Rand angezeigt werden soll.
num_recommendations	int	Anzahl der Artikel, die als Empfehlung angezeigt werden (Default: 2)
ref	string	Ein maximal 50 Zeichen langes Feld mit Tracking-Daten. Wenn ref gesetzt ist, ergänzt Facebook jeden Klick auf einen über die Recommendations-Bar veröffentlichten Eintrag mit folgenden GET-Parametern: fb_ref, der den Wert von ref enthält. fb_source gibt Auskunft darüber, wo der Link auf Facebook angeklickt wurde (home, profile, search, other).

Tabelle 8.14 Parameter der Recommendations-Bar (Forts.)

Der Activity Feed

Das in Abschnitt 6.6, »Der Activity Feed«, beschriebene Plugin wurde im Zuge der Veröffentlichung des neuen Open Graphs um Parameter zur Unterstützung von Aktionen erweitert. Während das Plugin ursprünglich alle Interaktionen aggregierte, die auf der im Parameter site definierten Website durchgeführt wurden, können Entwickler nun mit den Parametern app_id und action genauer definieren, welche Interaktionen bzw. Open-Graph-Aktionen in dem Plugin dargestellt werden sollen.

Name	Typ	Beschreibung
app_id	int	Wenn dieser Parameter gesetzt ist, werden alle Aktionen – von Facebook vordefinierte ebenso wie individuelle – angezeigt, die über die mit der ID angegebenen Anwendung veröffentlicht wurden. Dieser Parameter kann nur bei der Einbindung als iFrame angegeben werden, da die Anwendungs-ID ansonsten automatisch vom JavaScript SDK übernommen wird.
actions	string	In diesem Parameter können Sie in einer kommaseparierten Liste all jene Open-Graph-Aktionen angeben, die im Plugin angezeigt werden sollen. Dabei können sowohl von Facebook vordefinierte als auch individuelle Aktionen angegeben werden.

Tabelle 8.15 Zusätzliche Parameter des Activity Feeds zur Unterstützung des Open Graph

Alle weiteren Parameter des Plugins bleiben in ihrer Funktion unverändert und sind in Abschnitt 6.6, »Der Activity Feed«, nachzulesen. Unverändert ist auch, dass Sie den Activity Feed mittels iFrame, XFBML- oder HTML5-Tag in Ihre Webseiten integrieren können.

Das folgende Beispiel zeigt, wie Sie das Plugin für Ihre Webseite DigitalBartender.com so konfigurieren können, dass ausschließlich die veröffentlichten Open-Graph-Aktionen vom Typ Mix und Rate im Plugin angezeigt werden:

```
<div class="fb-activity"
     data-site="digitalbartender.com"
     data-action="digitalbartender:mix,digitalbartender:rate"
     data-width="300"
     data-height="300"
     data-header="true"
     data-recommendations="false">
</div>
```

Listing 8.22 Einbindung des Activity Feeds mit Unterstützung von Open-Graph-Aktionen

Hinweis: Während der Entstehung dieses Buches haben die neuen Parameter des Activity Feeds nur teilweise bzw. nicht immer funktioniert – so wurde der Parameter `actions` fallweise gänzlich ignoriert. Bis zum öffentlichen Start von Open Graph sollten diese Probleme jedoch von Facebook behoben worden sein.

Die Recommendations-Box

Auch das in Abschnitt 6.7, »Die Recommendations-Box«, beschriebene Plugin wurde um Open-Graph-Unterstützung erweitert und erlaubt nun die Filterung der angezeigten Interaktionen auf Open-Graph-Aktionen:

Name	Typ	Beschreibung
app_id	int	Wenn dieser Parameter gesetzt ist, werden alle Aktionen – von Facebook vordefinierte ebenso wie individuelle – angezeigt, die über die mit der ID angegebenen Anwendung veröffentlicht wurden. Dieser Parameter kann nur bei der Einbindung als iFrame angegeben werden, da die Anwendungs-ID ansonsten automatisch vom JavaScript SDK übernommen wird.
actions	string	In diesem Parameter können Sie in einer kommaseparierten Liste all jene Open-Graph-Aktionen angeben, die im Plugin angezeigt werden sollen. Dabei können sowohl von Facebook vordefinierte als auch individuelle Aktionen angegeben werden.

Tabelle 8.16 Zusätzliche Parameter der Recommendations-Box zur Unterstützung des Open Graph

Alle weiteren Parameter des Plugins bleiben in ihrer Funktion unverändert und sind in Abschnitt 6.7 nachzulesen. Die Recommendations-Box kann mittels iFrame, XFBML- oder HTML5-Tag in Ihre Webseiten integriert werden.

Das folgende Code-Beispiel zeigt, wie Sie die Recommendations-Box konfigurieren können, um ausschließlich Aktionen vom Typ RATE und MIX anzuzeigen:

```
<div class="fb-recommendations"
     data-site="digitalbartender.com"
     data-action="digitalbartender:mix,digitalbartender:rate"
     data-width="300"
     data-height="300"
     data-header="true">
</div>
```

Listing 8.23 Einbindung der Recommendations-Box mit Unterstützung von Open-Graph-Aktionen

Hinweis: Während der Entstehung dieses Buches haben die neuen Parameter der Recommendations-Box nur teilweise bzw. nicht immer funktioniert – so wurde der

Parameter `actions` fallweise gänzlich ignoriert. Bis zum öffentlichen Start von Open Graph sollten diese Probleme jedoch von Facebook behoben worden sein.

Die Facepile-Box

Auch das in Abschnitt 6.11, »Die Facepile-Box«, beschriebene Plugin kann nun benutzt werden, um die Profilbilder von Freunden anzuzeigen, die eine bestimmte Open-Graph-Aktion mit einem bestimmten Objekt veröffentlicht haben.

Name	Typ	Beschreibung
`href`	`string`	URL des Open-Graph-Objekts, auf das sich die angezeigten Aktionen beziehen müssen
`actions`	`string`	In diesem Parameter können Sie in einer kommaseparierten Liste all jene Open-Graph-Aktionen angeben, die im Plugin angezeigt werden sollen. Dabei können sowohl von Facebook vordefinierte als auch individuelle Aktionen angegeben werden.

Tabelle 8.17 Zusätzliche Parameter der Facepile-Box zur Unterstützung des Open Graph

Alle weiteren Parameter des Plugins bleiben in ihrer Funktion unverändert und sind in Abschnitt 6.11 nachzulesen. Die Facepile-Box kann mittels iFrame, XFBML- oder HTML5-Tag in Ihre Webseiten integriert werden.

Die folgende Einbindung der Facepile-Box zeigt jene Freunde des aktuellen Benutzers an, die mit einem bestimmten Objekt, nämlich dem Cocktail »White Russian«, durch die Aktion Mix in Beziehung getreten sind:

```
<div class="fb-facepile"
    data-href="http://digitalbartender.com/cocktails/whiterussian"
    data-action="digitalbartender:mix"
    data-width="300"
    data-max-rows="1"
</div >
```

Listing 8.24 Einbindung der Facepile-Box mit Unterstützung von Open-Graph-Aktionen

Hinweis: Während der Entstehung dieses Buches haben die neuen Parameter der Facepile-Box nur teilweise bzw. nicht immer funktioniert – so wurde der Parameter `actions` fallweise gänzlich ignoriert. Bis zum öffentlichen Start von Open Graph sollten diese Probleme jedoch von Facebook behoben worden sein.

Kapitel 9
Facebook Credits

Mit Facebook Credits bietet die Facebook-Plattform bereits seit 2011 ein praktisches Zahlungsmittel im Micropayment-Bereich. Facebook Credits werden bereits heute häufig zum Kauf von virtuellen Gütern in Spielen oder zur Bezahlung von Content eingesetzt. In diesem Kapitel erfahren Sie, wie Sie Facebook Credits in Ihrer Anwendung einsetzen.

Mit der Einführung des Mikropayment-Zahlungsmittels *Facebook Credits* bietet Facebook-Anwendungsentwicklern und Benutzern eine einfache Möglichkeit zur Abwicklung von Bezahl-Transaktionen beim Kauf von digitalen Produkten oder virtuellen Gütern an. Benutzer müssen dabei ihre Kreditkarten- oder sonstigen Zahlungsdaten nicht an den Entwickler oder Betreiber einer Anwendung übermitteln, sondern unterhalten stattdessen ein anwendungsübergreifendes Credits-Konto auf Facebook.com, das sie mit einer Vielzahl an herkömmlichen Zahlungsmitteln aufladen können. In Deutschland unterstützt Facebook dabei etwa Click & Buy, PaySafe Card, Sofortüberweisungen, Kreditkarten, PayPal sowie das Aufladen über Mehrwert-SMS. International unterstützt Facebook über 80 Zahlungsmethoden, womit auch ein gewaltiger Vorteil für Sie als Entwickler offenbar wird: Anstatt die gängigen Zahlungsmittel jedes nationalen Zielmarktes separat zu unterstützen, müssen Sie lediglich einmalig Facebook Credits in Ihrer Anwendung implementieren und können damit Ihre Anwendung auf einem weltweiten Markt anbieten.

Für den Service der Zahlungsabwicklung behält Facebook derzeit 30 % der Transaktionssumme als Entgelt ein – die Höhe dieser Gebühr ist mit jener in Apples iTunes Store vergleichbar und legt nahe, Facebook Credits vor allem für digitale Content-Produkte und virtuelle Güter einzusetzen und nicht etwa im klassischen E-Commerce. Erfolgreiche Beispiele für den Einsatz von Facebook Credits sind etwa der Verkauf von digitalen Inhalten (E-Books, Musik, Video) oder als digitales Zahlungsmittel im Social Gaming (so können etwa in den erfolgreichen Spielen der FarmVille-Reihe zusätzliche Features und Levels nur durch Bezahlung per Facebook Credits freigeschaltet werden).

Beim Aufladen des Credits-Kontos wird ein Euro zum Zeitpunkt der Entstehung dieses Buches mit einem Wechselkurs von circa 1:13,15 in Facebook Credits umgerechnet – für einen Euro erhält der Benutzer also 13.15 Credits, wobei die Mindestbezugs-

menge allerdings 50 Credits zum Preis von EUR 3,80 beträgt. Facebook stellt den Euro-Wert von Credits in allen Bezahl-Dialogen dar, sodass Benutzer zu jedem Zeitpunkt einschätzen können, wie teuer eine Transaktion tatsächlich ist.

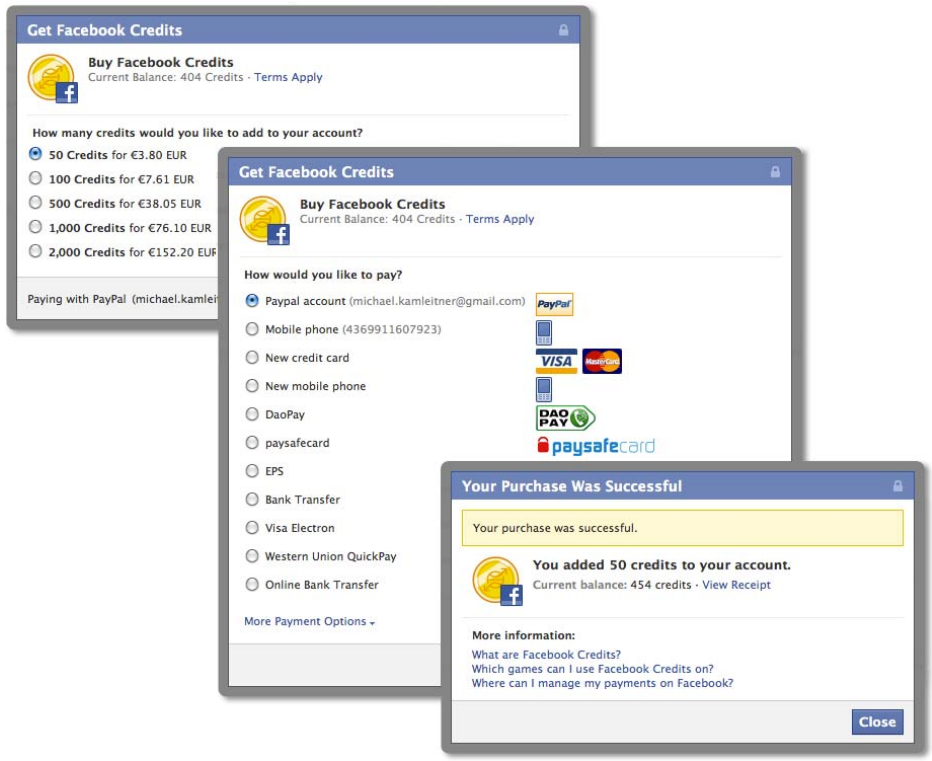


Abbildung 9.1 Aufladen des Facebook Credits-Kontos

Wichtige Links:

- *https://developers.facebook.com/docs/credits/* – offizielle Dokumentation zu Facebook Credits
- *https://developers.facebook.com/docs/credits/reports/* – Informationen zu den Zahlungsreports an Betreiber von Credits-Anwendungen sowie die unterstützen Länder
- *https://www.facebook.com/help/?faq=203680236341574* – Liste der unterstützten nationalen Zahlungsmittel zum Aufladen des Credits-Kontos

9.1 Einrichten von Facebook Credits

Um in Ihrer Anwendung Zahlungen mit Facebook Credits entgegennehmen zu können, müssen Sie zuvor einige Einstellungen im Abschnitt Credits der Anwendungs-

einstellungen vornehmen. Facebook verlangt dazu das Anlegen eines detaillierten Unternehmensprofils des Betreibers der Anwendung – neben der Anschrift und Rechtsform des Unternehmens und einer Kontaktperson müssen Sie hier vor allem angeben, auf welche Art Ihnen Facebook das Credits-Guthaben Ihrer Anwendung ausbezahlen soll. Unterstützt werden dabei Banküberweisungen und Überweisungen per PayPal.

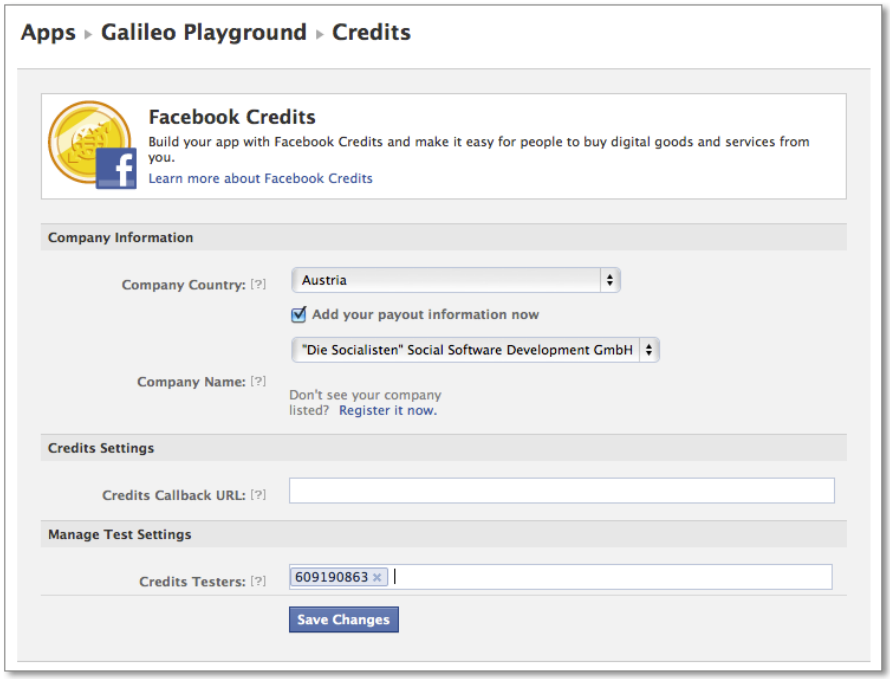


Abbildung 9.2 Einrichten des Unternehmensprofils zur Nutzung von Facebook Credits

Hinweis: Als Betreiber einer Anwendung mit Credits-Unterstützung erhalten Sie einen täglichen E-Mail-Report über alle eingegangenen Zahlungen in Form einer tabellarischen Textdatei. Alle Details zum Zahlungsreport finden Sie unter *https://developers.facebook.com/docs/credits/reports/*.

Achtung: In der EU ist der Verkauf von virtuellen Gütern und digitalen Inhalten prinzipiell umsatzsteuerpflichtig – wenn Sie als Anwendungsbetreiber Zahlungen mit Facebook Credits entgegennehmen, sind Sie dabei selbst für die Abführung der Umsatzsteuer zuständig.

Nachdem Sie Ihr Unternehmensprofil vollständig ausgefüllt haben, können Sie es in den Anwendungseinstellungen unter Company Name auswählen. Darüber hinaus können Sie unter Credits Testers beliebig viele Benutzer-IDs als Tester eintragen – diesen Benutzern werden beim Durchführen von Zahlungen in Ihrer Anwendung keine Credits abgezogen, sie können die Anwendung also kostenfrei ausprobieren.

9.2 Der Payment-Dialog

Wie in Abschnitt 4.4.4, »Payment-Dialog«, beschrieben, stellt das JavaScript SDK einen Dialog zur Durchführung von Zahlungen mittels Facebook Credits zur Verfügung. Dieser Dialog ist die einzige Möglichkeit, Zahlungen mit Credits entgegenzunehmen!

Der Payment-Dialog wird durch einen Aufruf der Methode `FB.ui()` geöffnet:

```
var order_info = 'produkt-1';
var obj = {
            method: 'pay',
            order_info: order_info,
            purchase_type: 'item',
            dev_purchase_params: {'oscif': true}
          };
FB.ui(obj, function (data) {
   console.debug(data);
});
```

Listing 9.1 Aufruf des Payment-Dialogs mit »FB.ui«

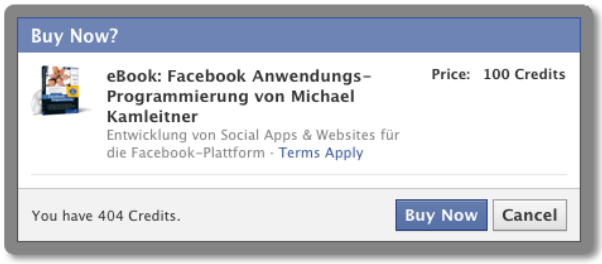


Abbildung 9.3 Der Payment-Dialog fordert den Benutzer zum Kauf eines Produkts auf.

`FB.ui()` wird mit dem Wert `pay` im Parameter `method` angewiesen, den Facebook-Credits-Zahlungsdialog zu öffnen. Wie Sie sehen, enthält der Parameter `order_info` den einzigen Hinweis darauf, welches Produkt eigentlich gekauft werden soll (in unserem Fall das Produkt mit der ID `produkt-1`). Die Anzeige des Zahlungsdialogs, der Details zum Produkt (Titel, Beschreibung, Thumbnail) und Preis enthält, kann also nur erfolgen, wenn die entsprechenden Informationen von der Callback-URL des Anwendungsservers bezogen werden können (siehe dazu Abschnitt 9.3, »Die Credits-Callback-URL«). Weitere Hinweise zu den möglichen Aufrufen des Zahlungsdialogs finden Sie in Abschnitt 4.4.4, »Payment-Dialog«.

9.3 Die Credits-Callback-URL

Bei der Zahlungsabwicklung mit Facebook Credits müssen Sie als Entwickler unter einer speziellen URL ein Callback-Skript bereitstellen, das von Facebook an bestimmten Punkten der Transaktion aufgerufen wird. Facebook übergibt dabei etwa die Bezeichnung des gekauften Produkts und erwartet dafür im Gegenzug detaillierte Informationen über Titel, Beschreibung und vor allem Preis des Produkts. Der Grund, warum diese Informationen nicht etwa direkt beim Aufruf des Dialogs mittels `FB.ui()` übergeben werden, ist, dass diese Informationen clientseitig leicht fälschbar wären. Durch das wechselweise Aufrufen der serverseitigen Callback-URL sind Credits-Transaktionen fälschungssicher.

Die Callback-URL wird in den Anwendungseinstellungen im Abschnitt Credits im Feld Credits Callback URL definiert. Die Callback-URL wird niemals direkt vom Browser des Benutzers aufgerufen, sondern immer nur von Facebooks Backend-Servern. Dennoch empfiehlt es sich, die Callback-URL von einer mit SSL verschlüsselten Adresse auszuliefern.

9.3.1 Aufruf der Callback-URL zum Abrufen der Produktdetails

Wenn ein Zahlungsdialog mittels `FB.ui()` geöffnet werden soll, löst das JavaScript SDK noch vor dem Anzeigen des Dialogs folgenden Prozess aus:

1. Das JavaScript SDK im Browser des Benutzers setzt einen AJAX-Request an das Backend von Facebook ab. Dieser Request enthält Angaben zum Benutzer, zur aktuellen Facebook-Anwendung und zu der im Parameter `order_info` an die Methode `FB.ui` übergebenen Produktidentifikation.
2. Facebooks Backend übergibt diese Anfrage wiederum an einen Aufruf der Callback-URL der Anwendung.
3. Die Anwendung nimmt den Aufruf unter der Callback-URL entgegen. Mithilfe der `order_info` wird erkannt, welches Produkt gekauft werden soll – der Anwendungsserver kann also dementsprechend mit Detailinformationen und Preis des Produkts antworten.
4. Facebooks Backend nimmt die Antwort der Callback-URL entgegen und gibt sie als Response auf die AJAX-Anfrage des JavaScript SDKs zurück. Das SDK bzw. `FB.ui` kann mit dieser Antwort nun den vollständigen Zahlungsdialog anzeigen.

Facebook ruft die Callback-URL dabei mit folgenden POST-Parametern auf:

```
Array
(
    [signed_request] => 5t-j4eVDpLNVP3TIg2xPYCxrGk_wnjdv...
    [buyer] => 609190863
```

```
    [receiver] => 609190863
    [order_id] => 310074429044862
    [order_info] => "produkt-1"
    [test_mode] => 1
    [method] => payments_get_items
)
```

Listing 9.2 POST-Parameter beim Aufruf der Callback-URL

Beachten Sie zuerst den Parameter `method`: Der Wert `payments_get_items` zeigt an, dass die Callback-URL zum *Abrufen der Produktdetails* zur Anzeige im Zahlungsdialog aufgerufen wurde. Der Parameter `test_mode` ist nur auf 1 gesetzt, wenn der Aufruf von einem unter Credits Testers eingetragenen Benutzer erfolgte.

Hinweis: Lassen Sie sich von den Feldern `buyer` und `receiver` nicht irritieren – Credits können derzeit nur zur Zahlung von Benutzern an Anwendungen verwendet werden – eine Zahlung zwischen zwei Benutzern ist hingegen nicht möglich.

Während auch die weiteren Parameter bereits Aufschluss über den Benutzer (`buyer`) und das angefragte Produkt (`order_info`) geben, untersuchen sie zuerst den übergebenen Signed Request (`signed_request`). Dieser kann wie gewohnt mit dem App Secret entschlüsselt werden und offenbart danach folgende Informationen:

```
Array
(
    [algorithm] => HMAC-SHA256
    [credits] => Array
        (
            [buyer] => 609190863
            [receiver] => 609190863
            [order_id] => 295289927194588
            [order_info] => "produkt-1"
            [test_mode] => 1
        )
    [expires] => 1328392800
    [issued_at] => 1328388682
    [oauth_token] =
> AAADDS2zLx44BALWPhCyXfZAbwOint3VVD2JpF94NpnZChoeZA7vMcSiNqRHW1nLkP8AnOiCeAen
rriZA6QTiDSk5WZAA4egfHaHwlz1SZBpQZDZD
    [user] => Array
        (
            [country] => at
            [locale] => en_US
            [age] => Array
```

```
                (
                    [min] => 21
                )
        )
    [user_id] => 609190863
)
```

Listing 9.3 Entschlüsselter Signed Request beim ersten Aufruf der Callback-URL

Unter dem Schlüssel `credits` enthält auch der Signed Request alle Informationen zur angefragten Transaktion. Der Vorteil des Signed Requests: Anders als die POST-Parameter können die Informationen im Signed Request nicht von einem etwaigen Angreifer gefälscht werden, da das App Secret, das nur Facebook und Ihnen als Anwendungsentwickler bekannt ist, zur Verschlüsselung verwendet wird.

Nach dem Entschlüsseln des Signed Requests muss das Skript hinter der Callback-URL die Detailinformationen und den Preis des im Parameter `order_info` angefragten Produkts ermitteln (etwa aus einer Produktdatenbank) und als JSON-codiertes Ergebnis zurückliefern:

```
{
   "method" : "payments_get_items",
   "content" : [
      { "item_id" : "produkt-1",
        "price" : 1000,
        "title" : "E-Book: Facebook-Anwendungsprogrammierung...",
        "description" : "Entwicklung von Social Apps & Websites
                         f\u00fcr die Facebook-Plattform",
        "image_url" : "http://cover.galileo-press.de/123.png",
        "product_url" : http://www.galileocomputing.de/ID-2991
     } ]
}
```

Listing 9.4 Antwort der Callback-URL in JSON-Notation

Wichtig: Das zurückgelieferte JSON-Array muss im Feld `method` wiederum den Wert `payments_get_items` enthalten.

Die Produktinformationen werden im Array `content` bereitgestellt. Obwohl die Übergabe als Array erfolgt, werden derzeit nur Transaktionen unterstützt, in denen ein einzelnes Produkt gekauft werden soll. Die Produktinformationen setzen sich aus folgenden Feldern zusammen:

Feldname	Beschreibung	Datentyp
`title`	Name des Produkts	`string` (max. 50 Zeichen)
`description`	Produktbeschreibung	`string` (max. 175 Zeichen)
`image_url`	URL zum Thumbnail des Produkts	`string`
`product_url`	URL zu weiteren Produktinformationen	`string`
`price`	Preis des Produkts in Credits	`integer` (>0)
`item_id`	anwendungsspezifische Produkt-ID (etwa Katalognummer)	`string`

Tabelle 9.1 Aufbau der Produkt-Detailinformationen, die von der Callback-URL zurückgeliefert werden

Wie Sie sehen, vergibt Facebook bereits beim Öffnen des Zahlungsdialogs eine numerische ID für die folgende Transaktion – diese wird im Signed Request im Feld `order_id` übergeben. Dies mag ungewöhnlich wirken, kann es doch vorkommen, dass der Benutzer die Transaktion gleich nach dem Öffnen wieder abbricht. Für Sie als Entwickler bietet es jedoch die Chance, alle begonnenen Transaktionen lückenlos in Ihrer Datenbank aufzuzeichnen. Sie sollten daher schon beim Aufruf der Callback-URL mit der Methode `payments_get_items` einen Datenbankeintrag für die begonnene Transaktion anlegen.

Den vollständigen PHP-Code des Skripts hinter der Callback-URL finden Sie am Ende dieses Abschnitts!

9.3.2 Aufruf der Callback-URL zum Bestätigen der Transaktion

Nun ist wieder der Benutzer am Zug – nachdem er im Zahlungsdialog die Produktinformationen geprüft hat, kann er den Kauf mit einem Klick auf Confirm bestätigen. Dabei löst das JavaScript SDK folgenden Prozess aus:

1. Das JavaScript SDK sendet vom Browser des Benutzers einen AJAX-Request an das Backend von Facebook. Die über die `order_id` identifizierte Transaktion wird dabei bestätigt und platziert – die Credits werden dem Benutzer noch nicht abgezogen, sondern nur »reserviert«.
2. Das Backend von Facebook ruft wiederum die Callback-URL der Anwendung auf, diesmal mit der Methode `payments_status_update`. Die Callback-URL erkennt daran, dass die Transaktion vom Benutzer bestätigt wurde. Die Anwendung muss das gekaufte Produkt nun »ausliefern«, indem sie etwa in der eigenen Datenbank einen entsprechenden Vermerk speichert.

3. Das Ergebnis der Callback-URL teilt dem Backend von Facebook im Feld `status` mit, dass die Transaktion ausgeführt (`settled`) wurde. Facebook zieht nun die vorher reservierten Credits vom Konto des Benutzers ab und schreibt sie dem Betreiber der Anwendung gut.
4. Das Backend meldet die erfolgreiche Durchführung der Transaktion an das JavaScript SDK, das eine entsprechende Erfoglsmeldung im Browser des Benutzers anzeigt.

Abbildung 9.4 Bestätigung der erfolgreich durchgeführten Transaktion

Beim zweiten Aufruf der Callback-URL wird im POST-Parameter `method` der Wert `payments_status_update` übergeben – anhand dieses Werts können Sie erkennen, dass Facebook die Callback-URL aufgerufen hat, um die Transaktion zu bestätigen. Darüber hinaus werden die ID der Transaktion, der `status` und die Produkt-Detailinformationen als JSON-Objekt im Feld `order_details` übergeben:

```
Array
(
    [signed_request] => SvO4GFhC24oLG8w4EC8Gg5...
    [order_details] => ...
    [status] => placed
    [order_id] => 301678476548809
    [test_mode] => 1
    [method] => payments_status_update
)
```

Listing 9.5 POST-Parameter beim zweiten Aufruf der Callback-URL

Auch der zweite Aufruf der Callback-URL enthält alle relevanten Daten im Signed Request:

```
Array
(
    [algorithm] => HMAC-SHA256
```

```
    [credits] => Array
        (
            [order_details] => {"order_
            id":267426806663880,"buyer":609190863,"app":214728715257742,"recei
            ver":609190863,"amount":1000,"update_time":1328441808,"time_
            placed":1328441805,"data":"","items":[{"item_id":"produkt-1",
            "title":"E-Book: Facebook-Anwendungsprogrammierung von Michael
            Kamleitner","description":"Entwicklung von Social Apps &
            Websites f\u00fcr die Facebook-Plattform","image_url":
            "http:\/\/cover.galileo-press.de\/9783836218436.png",
            "product_url":"http:\/\/www.galileocomputing.de\/katalog\/
            buecher\/titel\/gp\/titelID-2991","price":1000,"data":""}],
            "status":"placed"}
            [status] => placed
            [order_id] => 267426806663880
            [test_mode] => 1
        )
    [expires] => 1328446800
    ...
    [user_id] => 609190863
)
```

Listing 9.6 Entschlüsselter Signed Request beim zweiten Aufruf der Callback-URL

Besonders wichtig ist das Feld `status`, das mit dem Wert `placed` zu erkennen gibt, dass der Benutzer die Zahlung bestätigt hat und die Credits von Facebook reserviert wurden. Mit der übergebenen `order_id` kann Ihre Anwendung den Zusammenhang zur Transaktion herstellen und den Status gegebenenfalls in der Datenbank der Anwendung aktualisieren.

Die Callback-URL beantwortet den zweiten Aufruf wiederum mit einem JSON-Objekt, das im Objekt `content` die ID der Transaktion und deren Status bestätigt. Wurde die Transaktion erfolgreich durchgeführt, enthält `status` den Wert `settled`, konnte die Anwendung aus irgendeinem Grund die Auslieferung des bezahlten Produkts nicht vornehmen, antwortet sie mit dem Wert `cancelled`. In diesem Fall wird Facebook die reservierten Credits des Benutzers wieder freigeben und nicht von seinem Konto abbuchen.

```
{ "method" : "payments_status_update",
  "content" : { "status" : "settled",
                "order_id" : 301678476548809 }
}
```

Listing 9.7 Antwort der Callback-URL in JSON-Notation

Hinweis: Bis März 2012 erfolgt nach dem zweiten Aufruf der Callback-URL noch ein dritter Bestätigungs-Request – erkennbar an der Methode `payments_status_update` und dem `status` des Wertes `settled`. Dieser Aufruf hat keine Bedeutung mehr und kann somit von der Anwendung ignoriert werden.

9.3.3 Beispiel-Code zur Bedienung der Callback-URL

Das folgende Code-Beispiel bildet die vollständige Behandlung der beiden Aufrufe der Callback-URL in PHP ab.

```
<?
define('APP_ID', '214728715257742');
define('APP_SECRET', 'b93bff303c619...');
include("tools.php");

$signed_request =
parse_signed_request(@$_REQUEST["signed_request"], APP_SECRET);
// Prüfung, ob der Signed Request und die Methode vorhanden sind
if (isset($_REQUEST["method"]) &&
    isset($signed_request["credits"])) {
      $data = array();
      $data["method"] = $_REQUEST["method"];
      $payload  = $signed_request["credits"];
      $order_id = $payload["order_id"];

      // Hier erfolgt die Behandlung des zweiten Aufrufs
      if ($data["method"] == "payments_status_update") {
         if ($payload["status"] == "placed") {
        // Die Zahlung wurde vom Benutzer bestätigt.
               // Hier würde die Auslieferung des Produkts
               // erfolgen ...
               $next_state = "settled";
               $data["content"]["status"] = $next_state;
               // Aktualiseren Sie den Status der Transaktion
               // in Ihrer Datenbank!

         } else if( $payload["status"] == "settled" ){
               // Der veraltete dritte Aufruf kann ignoriert
               // werden ...
               @die;
         }
         $data["content"]["order_id"] = $order_id;
```

```
        // Hier erfolgt die Behandlung des ersten Aufrufs
        } else if ($data["method"] == "payments_get_items") {

           // Einlesen der order_info unter Beachtung der JSON-
           // Notation (Entfernen der Hochkommata)
           $order_info = str_replace('"','',$payload["order_info"]);

           $item = array();
           if ($order_info == "produkt-1") {
              $item["item_id"]     = $order_info;
              $item["price"]       = 1000;
              $item["title"]       = "E-Book: Facebook-Anwendung ...";
              $item["description"] = "Entwicklung Social Apps ...";
              $item["image_url"]   =
                  "http://cover.galileo-press.de/9783836218436.png";
              $item["product_url"] =
                  "http://www.galileocomputing.de/ID-2991";
           } elseif ($order_info == "produkt-2") {
              ...
           }
           if ($item) {
              $data["content"] = array($item);
              // Speichern Sie die Transaktion hier in Ihrer Datenbank!
           }
        }
        echo json_encode($data);
} else {
        die;
}
?>
```

Listing 9.8 PHP-Code zur Bedienung der Callback-URL

Hinweis: Schreiben und Testen des Codes zur Bedienung der Callback-URL können sich äußerst schwierig gestalten, da er üblicherweise nur im Hintergrund von Facebooks Backend-Servern aufgerufen wird. Ein manuelles Aufrufen scheitert meist an der Bereitstellung der Informationen im Signed Request. Eine mögliche Lösung ist das Schreiben einer Log-Datei. Dazu können die PHP-Funktionen `ob_start()`, `ob_get_contents()` und `ob_end_clean()` zur Pufferung der Ausgabe des Skripts genutzt werden:

```
<?
ob_start();

// Behandlung der Callback-URL wie beschrieben
...

$buffer = ob_get_contents();
ob_end_clean();
file_put_contents("credits.log", $buffer."\n", FILE_APPEND);
?>
```

Listing 9.9 Pufferung der Ausgabe und Schreiben einer Log-Datei

In der Datei `credits.log` werden damit alle Aufrufe der Callback-URL mit-protokolliert – eine große Hilfe beim Testen des Codes.

9.3.4 Zugriff auf Transaktionen mit der Graph API

Die Graph API bietet die Möglichkeit, Detailinformationen über durchgeführte Transaktionen auch im Nachhinein abzurufen. Dazu muss lediglich ein GET-Zugriff auf folgenden API-Endpunkt durchgeführt werden:

```
https://graph.facebook.com/OrderID
```

Dieser Zugriff muss mit dem Anwendungs-Access-Token signiert werden! Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau:

```
{
   "id": "",
   "from": {
      "name": "",
      "id": ""
   },
   "to": {
      "name": "",
      "id": ""
   },
   "amount": ,
   "status": "",
   "application": {
      "name": "",
      "id": ""
   },
```

```
   "country": "",
   "created_time": "",
   "updated_time": ""
}
```

Listing 9.10 JSON-Objekt zur Darstellung eines Transaktionsobjekts

Neben einer einzelnen Transaktion kann mit dem Anwendungs-Access-Token auch eine Liste aller Transaktionen oder aller Transaktionen eines bestimmten Benutzers abgefragt werden. Dazu dienen folgende API-Endpunkte:

```
https://graph.facebook.com/AnwendungsID/payments
https://graph.facebook.com/BenutzerID/payments
```

Mit dem Parameter `status` können dabei auch Transaktionen mit einem bestimmten Status gefiltert werden (`settled`, `placed` ...), mit den Feldern `since` und `until` kann, wie bei der Graph API üblich, eine Einschränkung auf einen bestimmten Zeitraum erfolgen.

9.4 Facebook Credits und Offers

Mit *Offers* bietet Facebook Benutzern eine Möglichkeit, durch das Konsumieren oder Komplettieren von Werbeangeboten zu verdienen. So werden Benutzer etwa für das Betrachten eines Werbevideos oder das Ausfüllen einer Umfrage mit einer Gutschrift von Facebook Credits belohnt. Wenn Sie als Unternehmen eigene Offers anbieten möchten, müssen Sie dies in Zusammenarbeit mit dem Facebook-Partner *http://www.trialpay.com/* tun – eine detaillierte Beschreibung eigener Offers würde den Rahmen dieses Buches sprengen.

Möchten Sie hingegen Ihren Benutzern die Möglichkeit bieten, Offers von anderen Unternehmen zu komplettieren, um den eigenen Credits-Kontostand aufzubessern, genügt ein einfacher Aufruf des Zahlungsdialogs mit dem speziellen Parameter `action=earn_credits`:

```
var obj = {
    method: 'pay',
    action: 'earn_credits'
  };
FB.ui(obj, js_callback);
```

Listing 9.11 Aufruf der »Offer Wall« mit »FB.ui«

Dieser Aufruf öffnet anstelle des Zahlungsdialogs die sogenannte *Offer Wall* – eine je nach Nationalität des Benutzers angepasste Übersicht über alle verfügbaren Offers von anderen Unternehmen:

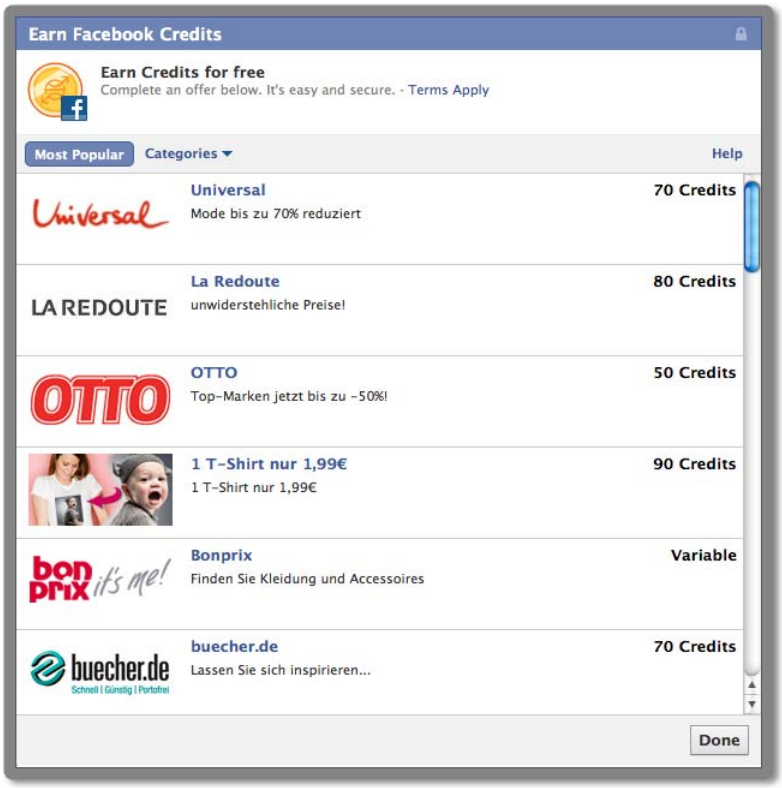


Abbildung 9.5 Anzeige aller verfügbaren Offers in der »Offer Wall«

Nach Vervollständigung eines Offers wird dem Benutzer automatisch die entsprechende Summe an Facebook Credits gutgeschrieben – als Anwendungsentwickler müssen Sie sich also nicht weiter um Offers kümmern. Alternativ können Sie Offers auch nutzen, um Benutzern anstelle von Facebook Credits anwendungsspezifische Gutschriften zukommen zu lassen – dies ist praktisch, wenn Sie etwa eine eigene virtuelle Währung in Ihrer Applikation verwenden. Eine genaue Beschreibung dieser *In-App Currency Offers* finden Sie unter *https://developers.facebook.com/docs/credits/offers/*.

Anhang A
Cloud-Hosting von Facebook-Anwendungen mit Heroku

Mit dem Cloud-Hosting-Service Heroku steht Facebook-Anwendungsentwicklern eine mächtige und innovative Möglichkeit zum Betrieb eigener Anwendungen zur Verfügung. Der Start in die Welt des Cloud-Hostings ist dabei sogar kostenlos! In diesem Kapitel erfahren Sie, wie Sie Ihre erste Facebook-Anwendung auf Heroku einrichten können.

Viele Entwickler, die mit dem Schreiben von Webanwendungen beginnen möchten, müssen noch vor dem Schreiben der ersten Code-Zeile zuerst für eine geeignete Hosting-Umgebung für die eigene Applikation sorgen. Dabei müssen zahlreiche Software-Komponenten, wie etwa der Webserver, die Datenbank-Software und die verwendete Skriptsprache, ausgewählt und aufeinander abgestimmt werden. Zwar bieten viele herkömmliche Hosting-Provider bereits günstige Einstiegspakete mit vorinstallierter Software an, oft reichen die Ressourcen solcher meist mit anderen Kunden geteilten Webserver allerdings nicht für den Betrieb von vielgenutzten Webanwendungen aus. Der Umzug der eigenen Anwendung auf ein leistungsstärkeres Hosting-Paket ist wiederum aufwendig und bringt oft neue Probleme mit sich, etwa die regelmäßige Wartung und Aktualisierung der Server-Software oder das Durchführen von Backups.

Mit dem Aufkommen von Cloud-Hosting-Anbietern wie etwa Amazon Web Services oder Google App Engine bieten sich Webentwicklern mittlerweile aber einfach in Betrieb zu nehmende, kostengünstige und doch flexible Alternativen zum Betrieb des eigenen Servers an. Beim Cloud-Hosting stellt der Anbieter bestimmte Services wie etwa Webserver, Datenbank etc. bereit und sorgt für deren Konfiguration und laufende Aktualisierung. Bei der Wahl eines Cloud-Hosting-Anbieters müssen Sie als Entwickler genau beachten, ob die von Ihnen verwendete Programmiersprache, die gewünschte Datenbank und sonstige Services auch tatsächlich vom Provider unterstützt werden – anders als beim eigenen Server gibt es im Cloud-Hosting nämlich normalerweise keine Möglichkeit, eigene Software zu installieren.

Heroku (*http://www.heroku.com*) ist ein beliebter Cloud-Hosting-Provider, der Anwendungen in den Programmiersprachen PHP, Node.js, Python und Java unter-

stützt. Zusätzlich bietet Heroku zahlreiche Add-ons wie etwa Unterstützung für PostgreSQL-Datenbanken, Mail-Versand und zahlreiche andere externe Services.

Im September 2011 hat Facebook die Hosting-Services von Heroku in den Facebook Developer integriert. Damit können Facebook-Anwendungsentwickler mit wenigen Mausklicks Facebook-Anwendungen auf der Infrastruktur von Heroku betreiben. In diesem Kapitel erfahren Sie, wie Sie als Facebook-Entwickler die ersten Schritte mit Heroku machen.

Anwendungs-Hosting mit Heroku – die Kosten

Die gute Nachricht zuerst: Der Basis-Funktionsumfang von Heroku kann während der Entwicklung kostenfrei genutzt werden – um Ihre erste Anwendung in der Cloud zu hosten, müssen Sie dazu nicht einmal Ihre Kreditkartendaten angeben.

Für produktive Anwendungen berechnen sich die monatlichen Kosten von Heroku nach der Anzahl der gewünschten gleichzeitigen Zugriffe. Sogenannte *Dynos* übernehmen dabei die Durchführung der einzelnen Zugriffe von Benutzern (*Web Dynos*) und Hintergrundprozessen (*Worker Dynos*). Für eine Anwendung, die fünf parallele Zugriffe durch Clients unterstützen soll, müssen Sie mit etwa 180 US-Dollar monatlichen Kosten rechnen. Zusätzliche Add-ons wie Datenbanken, Mail-Versand und Anbindungen an externe Services werden gesondert verrechnet.

Für eine komplexe Anwendung können dabei nicht unerhebliche Kosten entstehen, die meist über den reinen Mietkosten eines eigenen Servers liegen. Allerdings darf nicht außer Acht gelassen werden, dass eine auf Heroku betriebene Anwendung in der laufenden Wartung deutlich günstiger und leichter skalierbar ist. Heroku bietet unter *http://www.heroku.com/pricing* ein Tool zur Berechnung der monatlichen Kosten an.

Wichtige Links:

- *http://www.heroku.com* – Homepage des Cloud-Hosting Providers
- *http://devcenter.heroku.com/articles/facebook* – Heroku-Tutorial für Facebook-Entwickler
- *https://developers.facebook.com/blog/post/558/* – Veröffentlichung der Partnerschaft zwischen Facebook und Heroku

A.1 Anlegen der Heroku-Testapplikation

Um Ihre erste Anwendung auf der Hosting-Umgebung von Heroku einzurichten, legen Sie im Facebook Developer zuerst eine neue Anwendung an. Wählen Sie dazu einen passenden App Display Name (etwa Hello Heroku!) und einen verfügbaren

App Namespace (z. B. galileo-heroku). Nehmen Sie vorerst keine weiteren Einstellungen im Einstellungsabschnitt Basic vor, sondern klicken Sie unter Cloud Services auf den Button Get Started:

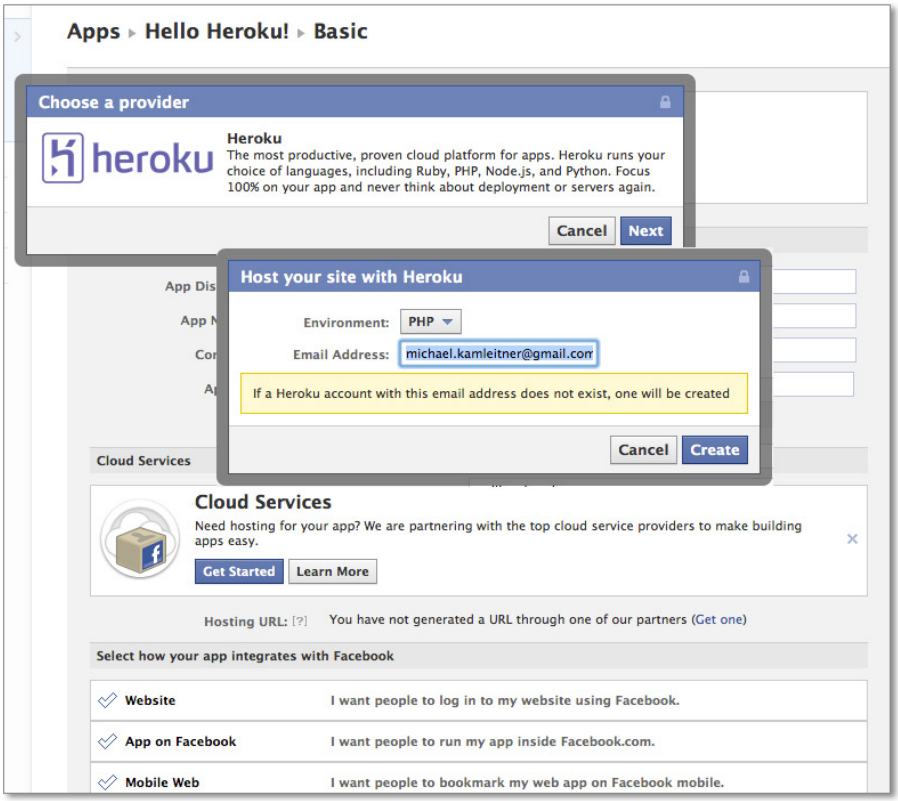


Abbildung A.1 Einrichten der Cloud-Hosting-Umgebung von Heroku

Wählen Sie im nun folgenden Dialog PHP als Environment, und geben Sie Ihre persönliche E-Mail-Adresse oder die Verteiler-Adresse Ihres Entwicklerteams ein. Heroku legt automatisch einen Account unter dieser E-Mail-Adresse an und sendet Ihnen eine E-Mail zur Bestätigung. Bei der Bestätigung dieser Mail legen Sie außerdem Ihr Heroku-Kennwort fest. Dieses benötigen Sie später, um Anwendungscode auf Ihre Heroku-Instanz laden zu können.

Gleichzeitig hat Heroku bereits die neue Anwendung auf seiner Infrastruktur angelegt – diese steht sofort unter einer zufällig generierten Sub-Domain, z. B. *https://quiet-autumn-6610.herokuapp.com/*, zur Verfügung. Dieser Hostname wurde dabei automatisch als Website im Anwendungs-Developer eingetragen. Geben Sie diese URL zusätzlich auch als Canvas URL (mit dem Protokoll HTTP) und Secure Canvas URL (mit dem Protokoll HTTPS) an:

Abbildung A.2 Die neu angelegte Heroku-Anwendung steht unter einer zufällig generierten Sub-Domain von herokuapp.com bereit.

Sie ahnen es bereits: Heroku nimmt Entwicklern die Bereitstellung eines gültigen SSL-Zertifikats ab – zumindest wenn Sie bereit sind, Ihre Anwendung unter der Sub-Domain von herokuapp.com zu betreiben. Selbstverständlich können Sie Heroku-Anwendungen aber auch unter Ihrem eigenen Domain-Namen nutzen!

Wenn Sie die neue Anwendung nun unter der URL *http://apps.facebook.com/galileo-heroku* aufrufen, werden Sie automatisch zum OAuth-Autorisierungsdialog der Applikation weitergeleitet. Dies geschieht, weil der Beispiel-Code, den Heroku auf jeder neuen Anwendung hinterlegt, eine zwingende Autorisierung erfordert. Doch wie können Sie diesen Code lesen oder bearbeiten, wie erhalten Sie als Entwickler Zugriff auf den Inhalt von *quiet-autumn-6610.herokuapp.com*? Anders als bei herkömmlichem Web Space stellt Heroku weder FTP-, SCP- noch Shell-Zugang bereit, sondern erfordert die Nutzung der Versionskontrolle Git.

A.2 Bearbeiten des Anwendungscodes mit Git

Git ist ein beliebtes Software-Tool zur Versionskontrolle, das ursprünglich vor allem im Open-Source-Bereich eingesetzt wurde, heute aber auch weite Verbreitung bei Webentwicklern genießt. Um Git zu nutzen, müssen Sie die Software erst auf Ihrer lokalen Maschine installieren. Obwohl auch grafische Git-Clients zur Verfügung stehen, werden die folgenden Schritte mit dem Kommandozeilen-Tool, das unter allen Plattformen gleich funktioniert, demonstriert. Zusätzlich zu Git installieren Sie nun noch den »Heroku Toolbelt«, eine Sammlung lokaler Tools zum Management Ihrer Heroku-Anwendung, die für die wichtigsten Betriebssysteme zur Verfügung steht.

Wichtige Links:

- *http://git-scm.com/* – Download der Software Git für zahlreiche Plattformen (Windows, Mac OS, Linux ...)
- *http://toolbelt.herokuapp.com/* – der Heroku Toolbelt

Bevor Sie mit Git arbeiten können, müssen Sie zuerst die SSH-Verbindung zwischen Ihrem lokalen Computer und Heroku einrichten. Tun Sie dies, indem Sie den Befehl `heroku login` auf der Kommandozeile ausführen:

```
$ heroku login
Enter your Heroku credentials.
Email: michael.kamleitner@gmail.com
Password:
Could not find an existing public key.
Would you like to generate one? [Yn]
Generating new SSH public key.
Uploading ssh public key /Users/michael/.ssh/id_rsa.pub
```

Listing A.1 Einrichten der SSH-Verbindung mit Heroku

Geben Sie die E-Mail-Adresse und das Kennwort Ihres Heroku-Kontos ein. Anschließend generiert das Tool einen lokalen SSH-Schlüssel bzw. verwendet einen bereits vorhandenen. Der SSH-Schlüssel wird automatisch auf die Heroku-Server hochgeladen und gewährleistet fortan die Autorisierung Ihrer lokalen Maschine.

Im nächsten Schritt können Sie mit Git eine lokale Kopie des Beispiel-Codes beziehen (man spricht dabei in Git vom *Klonen* des Repositorys), die Heroku automatisch beim Anlegen der neuen Applikation in einem Git-Repository hinterlegt hat:

```
$ git clone git@heroku.com:quiet-autumn-6610.git -o heroku
Cloning into quiet-autumn-6610...
remote: Counting objects: 139, done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 139 (delta 64), reused 135 (delta 62)
Receiving objects: 100% (139/139), 73.38 KiB | 82 KiB/s, done.
Resolving deltas: 100% (64/64), done.
```

Listing A.2 Anlegen eines lokalen Klons des Anwendungscodes

Die URL des Git-Repositorys Ihrer Anwendung (*git@heroku.com:quiet-autumn-6610.git*) ergibt sich dabei immer aus der zufällig für Ihre Anwendung generierten Sub-Domain (*quiet-autumn-6610.herokuapp.com*), kann aber auch unter *https://api.heroku.com/myapps* ermittelt werden – in Ihrer Heroku-Account-Verwaltung finden Sie all Ihre Heroku-Anwendungen und deren Eigenschaften.

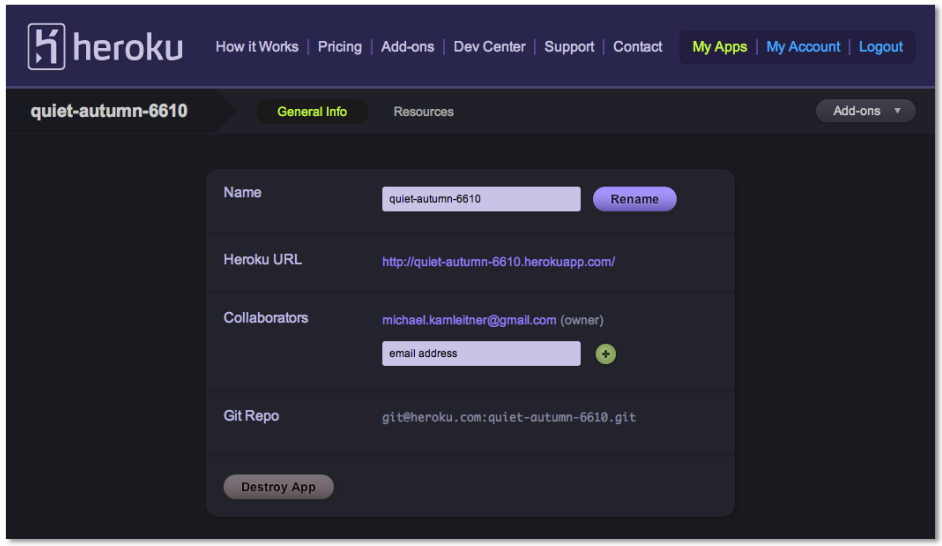


Abbildung A.3 Account- und Applikationsverwaltung auf Heroku.com

Nach dem Klonen des Repositorys finden Sie auf Ihrer lokalen Maschine ein Verzeichnis namens *quiet-autumn-6610*, das den Beispiel-Code Ihrer ersten Anwendung enthält:

```
$ cd quiet-autumn-6610/
$ ls
AppInfo.php  FBUtils.php  Readme.md    close.php    images
index.php    stylesheets  utils.php
```

Listing A.3 Lokale Kopie des Git-Repositorys Ihrer Heroku-Anwendung

Um Ihre erste eigene Heroku-Anwendung hochzuladen, können Sie den vorhandenen Beispiel-Code mit folgendem Kommando löschen:

```
$ git rm -r *
rm 'AppInfo.php'
rm 'FBUtils.php'
...
```

Danach legen Sie einfach eine neue Datei namens *index.php* mit folgendem Inhalt an:

```
<!DOCTYPE html>
<html lang="de-de" xmlns:fb="http://www.facebook.com/2008/fbml">
<head>
   <meta charset="utf-8">
```

```
   <title>Hello Facebook and Heroku!</title>
</head>
<body>
   <h1>Hello Facebook!</h1>
   <p>Willkommen bei meiner ersten Facebook-Anwendung auf Heroku!</p>
</body>
</html>
```

Listing A.4 Code Ihrer ersten Heroku-Anwendung

Mit dem Kommando `git add index.php` fügen Sie die neue Datei Ihrem lokalen Repository hinzu, um die Änderungen anschließend mit `git commit` zu bestätigen:

```
$ git add index.php
$ git commit -am "Meine erste Heroku-App"
[master b27ceb7] Meine erste Heroku-App
 1 files changed, 12 insertions(+), 0 deletions(-)
 create mode 100644 index.php
```

Listing A.5 Hinzufügen der Datei »index.php« und Bestätigen der Änderungen

Zum Abschluss müssen Sie Ihr lokales Git-Repository, das nun anstelle des Heroku-Beispiel-Codes Ihre eigene *index.php* enthält, auf die Server von Heroku übertragen. Dies geschieht mit dem Kommando `git push`:

```
$ git push heroku master
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 644 bytes, done.
Total 5 (delta 0), reused 0 (delta 0)
-----> Heroku receiving push
-----> PHP app detected
-----> Discovering process types
       Procfile declares types -> (none)
       Default types for PHP   -> web
-----> Compiled slug size is 21.5MB
-----> Launching... done, v5
       http://quiet-autumn-6610.herokuapp.com deployed to Heroku

To git@heroku.com:quiet-autumn-6610.git
   ec75edf..b27ceb7  master -> master
```

Listing A.6 Übertragen des lokalen Repositorys auf die Server von Heroku

Fertig! Ihre erste Facebook-Anwendung liegt nun auf Ihrer Heroku-Instanz bereit und kann im Browser aufgerufen werden!

A.3 Aufrufen der Heroku-Testapplikation

Um die Heroku-Anwendung aufzurufen, können Sie entweder direkt die CANVAS URL Ihrer Anwendung (*https://quiet-autumn-6610.herokuapp.com/*) oder das Application Canvas über die URL *https://apps.facebook.com/galileo-heroku/* aufrufen. Wie erwartet wird der Inhalt der Datei *index.php* ausgegeben:

Abbildung A.4 Application Canvas Ihrer ersten Heroku-Anwendung

Mit diesem kleinen Tutorial haben Sie einen Überblick über die Anwendungsentwicklung mit Heroku erhalten. Obwohl das Management des Quellcodes der Anwendung mittels Git auf den ersten Blick kompliziert erscheinen mag, bietet dieses Tool zur Versionskontrolle gerade bei der gleichzeitigen Arbeit mehrerer Entwickler in Teams wesentliche Vorteile. Die Anwendungsentwicklung mit Heroku ist es jedenfalls wert, ausprobiert zu werden – bevor Sie aber die Entscheidung fällen, ein größeres Projekt auf Heroku umzusetzen, sollten Sie ausführlich mit den angebotenen Zusatzservices wie etwa der integrierten Datenbank experimentieren – nur so können Sie abschätzen, ob Sie die geplanten Funktionen auf Heroku auch tatsächlich umsetzen können.

Anhang B
Internationalisierung von Facebook-Anwendungen

Die Benutzeroberfläche von Facebook ist derzeit in über 70 Sprachen verfügbar. Mit dem Translations-Tool stellt Facebook Anwendungsentwicklern umfangreiche Möglichkeiten zur Internationalisierung der eigenen Anwendung zur Verfügung. Erfahren Sie in diesem Kapitel, wie Sie Ihre Facebook-Anwendungen mehrsprachig umsetzen können.

Ein starkes Argument, Anwendungen für die Facebook-Plattform zu entwickeln, ist die enorme Anzahl an weltweiten Benutzern. Facebook selbst sah sich bald der Herausforderung gegenübergestellt, die Benutzeroberfläche von Facebook.com möglichst rasch in möglichst vielen Sprachen anzubieten. Man ist dabei einen eher ungewöhnlichen Weg gegangen: Mit dem Launch des *Translations-Tools* konnten Benutzer selbst an der Übersetzung in ihre Sprache mitarbeiten. Neue Übersetzungen wurden von der Community bewertet und bei Erreichen eines gewissen qualitativen Mindeststandards veröffentlicht. Der Plan ist aufgegangen – Facebook war schneller und in mehr Sprachen verfügbar als andere Services mit vergleichbarer Komplexität. Glücklicherweise steht Facebook Translations auch Ihnen als Entwickler zur Übersetzung Ihrer eigenen Anwendungen zur Verfügung – Sie müssen dabei allerdings nicht unbedingt auf die Mitarbeit Ihrer Anwender zurückgreifen, sondern können die Übersetzung selbstverständlich auch selbst vornehmen.

B.1 Unterstützung von Locales und JavaScript SDK

Facebook unterstützt eine große Zahl an *Locales*, also länderspezifischen Lokalisierungseinstellungen. Diese Locales werden dem ISO-Schema gemäß in der Form `ll_CC` codiert, wobei `ll` die Sprache und `CC` das Land repräsentiert (z. B. `de_DE` für Deutsch (Deutschland) oder `en_US` für US-Englisch). Eine vollständige Liste der unterstützten Locales wird von Facebook in einer XML-Datei unter *https://www.facebook.com/translations/FacebookLocales.xml* bereitgestellt.

Die gewünschte Locale kann beim Laden des JavaScript SDKs als Teil der URL ausgewählt werden:

```
var js,id = 'facebook-jssdk'; if (d.getElementById(id)) {return;}
 js = d.createElement('script'); js.id = id; js.async = true;
 js.src = "//connect.facebook.net/en_US/all.js";
 d.getElementsByTagName('head')[0].appendChild(js);
```

Besonders hilfreich bei Canvas-Applikationen ist dabei der Signed Request, da dieser immer die Locale des aktuellen Benutzers – also die Spracheinstellung des Benutzers auf Facebook.com – enthält. Damit kann ein dynamisches Laden des SDKs mit der jeweils richtigen Locale umgesetzt werden:

```
js.src = "//connect.facebook.net/<?= $signed_request["user"]["locale"]
?>/all.js";
```

Das Laden des JavaScript SDKs mit der gewünschten Locale bewirkt, dass alle Dialoge (OAuth-Dialog, Wall-Postings etc.) automatisch in der passenden Sprache angezeigt werden.

B.2 Internationalisierung von Anwendungstexten

Das Translations-Tool dient der Übersetzung von Texten, die an verschiedenen Stellen des Benutzer-Interfaces der Anwendung angezeigt werden. Dabei werden vor allem folgende Arten von Texten unterstützt:

- Alle Texte, die in den Anwendungseinstellungen auf Facebook.com eingegeben werden – dazu zählen etwa der Applikationsname, die Beschreibung der Applikation, Erklärungstexte des Autorisierungsdialogs, der Name des Tabs/Reiters sowie alle Texte, die mit Open-Graph-Objekten, -Aktionen und -Aggregationen zusammenhängen.
- Alle Texte, die vom Anwendungsserver im HTML-Code des Application Canvas angezeigt werden. Diese Texte müssen vom Entwickler allerdings zuvor mit speziellen XFBML-Tags für die Übersetzung durch Translations vorbereitet werden.

Für manch andere Art von Texten stellt Translations jedoch keine Möglichkeit zur Übersetzung bereit. In diesen Fällen müssen Anwendungsentwickler selbst für eine passende Internationalisierung anhand der Locale des Benutzers sorgen:

- Texte, die in der Datenbank der Anwendung gespeichert werden, also etwa Produktinformationen, Artikel etc.
- Bilder und Grafiken, die Texte enthalten, etwa Navigationselemente oder Menüstrukturen der Anwendung
- Texte, die in E-Mails an Benutzer versandt werden

Damit Translations überhaupt Texte im HTML-Code Ihrer Anwendung erkennen und zur Übersetzung bereitstellen kann, müssen Sie diese mit speziellen XFBML-Tags auszeichnen. Das Tag `<fb:intl>` markiert dabei eine zu übersetzende Textstelle, wobei im Attribut `desc` optional eine kurze Beschreibung des Textes angegeben werden kann:

```
<fb:intl desc="App-Headline">Welcome to my App</fb:intl>
```

Der Text »Welcome to my App« steht damit automatisch zur Übersetzung im Translations-Tool bereit. Oft enthalten Texte Platzhalter zur dynamischen Darstellung von Informationen, die nicht mit übersetzt werden können. Mit dem Tag `<fb:intl-token>` können diese Platzhalter von der Übersetzung ausgenommen werden:

```
<fb:intl desc="Ask to tate a cocktail">
Do you want to rate <fb:intl-token name="cocktail-title">White Russian
</fb:intl-token> now?
</fb:intl>
```

Im Übersetzungs-Tool wird diese Textstelle folgendermaßen angezeigt:

```
Do you want to rate {cocktail-title} now?
```

Der Übersetzer muss dabei den in geschwungenen Klammern angezeigten Platzhalter in die Übersetzung übernehmen, kann aber die Reihenfolge der Worte beliebig anpassen:

```
Möchtest du {cocktail-title} jetzt bewerten?
```

Einen Sonderfall stellt die Übersetzung von HTML-Attributen dar – so wird etwa die Beschriftung eines SUBMIT-Buttons üblicherweise im `value`-Attribut des `<input>`-Tags definiert. Um dem Translations-Tool solche Texte zugänglich zu machen, kann das Tag `<fb:tag-attribute>` verwendet werden. Dabei wird der Wert von `value` aus dem `<input>`-Tag extrahiert und gesondert festgelegt:

```
<input type="submit">
  <fb:tag-attribute name="value">
    <fb:intl desc="Submit Rating">Submit Rating</fb:intl>
  </fb:tag-attribute>
</input>
```

Weitere Attribute erlauben das lokalisierte Anzeigen von Datumsinformationen (`<fb:date>`), das Setzen des Titels des Browser-Fensters (`<fb:window-title>`) und das Übersetzen ganzer HTML-Tags (`<fb:tag>`, `<fb:tag-body>`). Detaillierte Informationen dazu finden Sie unter *https://developers.facebook.com/docs/internationalization/*.

Hinweis: Die eigentliche Anzeige der übersetzten Texte wird zur Laufzeit vom JavaScript SDK erledigt. Es durchsucht das HTML-Dokument nach allen `<fb:intl>`-Tags und tauscht deren Inhalte gemäß der Locale aus, die beim Initialisieren des SDKs definiert wurde.

Um auf das Translations-Tool für Ihre Anwendung zuzugreifen, klicken Sie auf den Link Translate Your App in der linken Spalte der Anwendungseinstellungen.

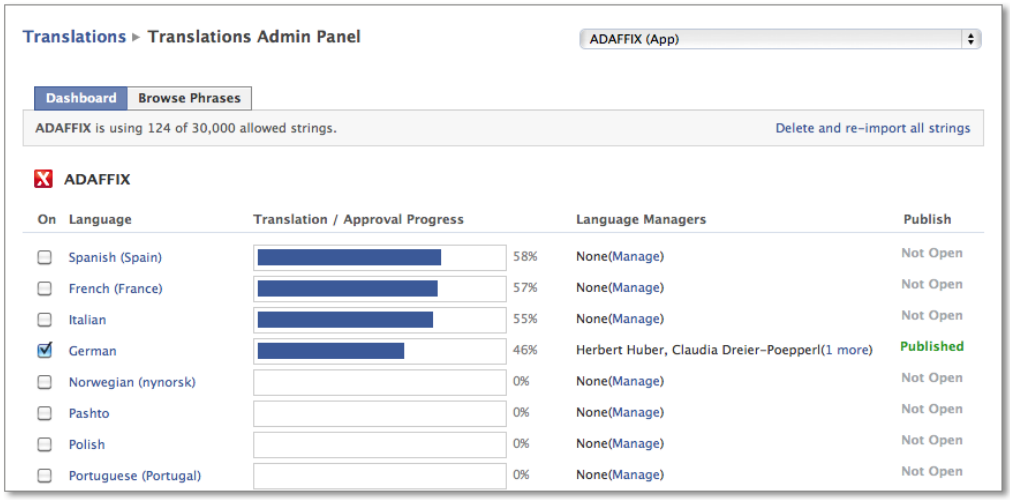


Abbildung B.1 Das »Translations«-Dashboard einer Anwendung

Das Dashboard des Translations-Tools zeigt alle verfügbaren Locales tabellarisch an. Sprachen, die mit On markiert sind, können von beliebigen Benutzern der Anwendung übersetzt werden – möchten Sie als Entwickler die Übersetzung lieber allein vornehmen, sollten Sie diese Checkbox also deaktiviert lassen. Unter Approval Progress wird der Fortschritt der Übersetzungen grafisch dargestellt, und in der Spalte Publish werden jene Sprachen grün hervorgehoben, die bereits der Öffentlichkeit zur Verfügung gestellt werden. Unter Language Managers können Sie für jede

Sprache eine Reihe an Benutzern festlegen, die bereits übersetzte Textstellen ändern, löschen oder freigeben können – dies ist vor allem bei der Community-unterstützten Übersetzung hilfreich.

Ein Klick auf die jeweilige Sprache oder BROWSE PHRASES lädt das eigentliche Übersetzungs-Tool. Hier werden alle Texte, die Translations entweder aus den Anwendungseinstellungen übernommen oder dank der Markierung durch das Tag `<fb:intl>` am Application Canvas erkannt hat, aufgelistet.

Die linke Spalte zeigt dabei die Texte in der Originalsprache an, in der rechten können diese in die Zielsprache übersetzt werden. Dabei hilft das Tool mit der Übernahme von Platzhaltern (etwa `{cocktail-title}`) per Mausklick und einer Suchfunktion. Die einzelnen Texte können mit einem Klick auf APPROVE bestätigt werden. Aber Achtung: Erst wenn Sie die gesamte Sprache im Dashboard mittels PUBLISH veröffentlichen, wird die Übersetzung auch tatsächlich beim Benutzer angezeigt!

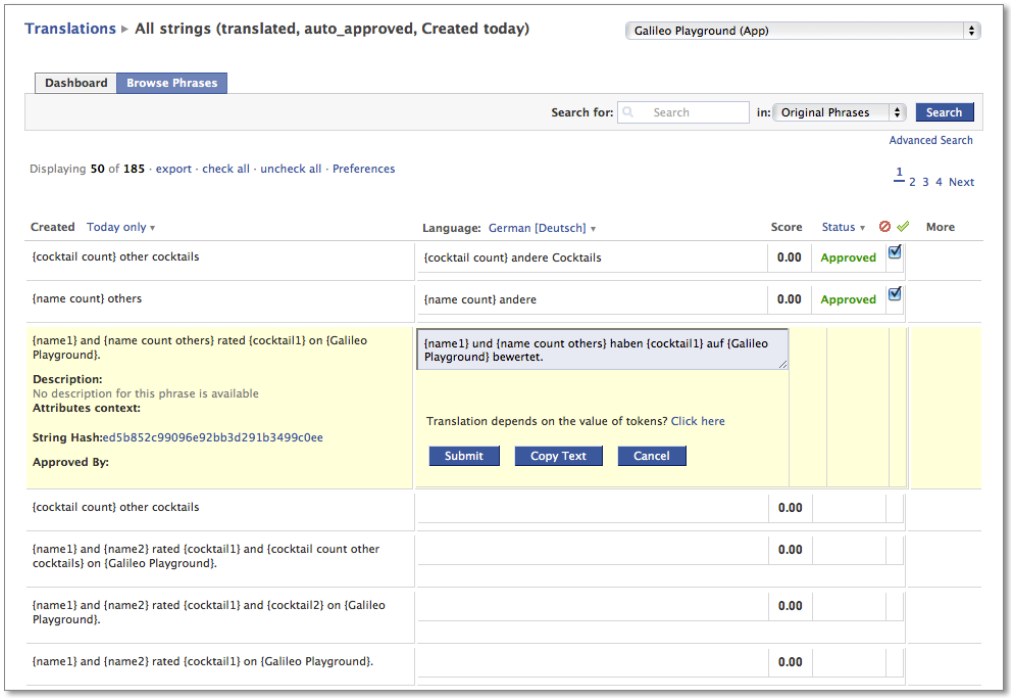


Abbildung B.2 Das »Translations«-Tool dient der Übersetzung von Anwendungstexten.

Anhang C
Das offizielle Facebook PHP SDK

Facebook bietet für PHP-Programmierer traditionell ein offizielles SDK an. Diese Programmbibliothek vereinfacht den Zugriff auf die Graph API und andere Schnittstellen der Plattform und dient darüber hinaus als Referenz für SDKs in anderen Programmiersprachen. In diesem Kapitel erfahren Sie, wie Sie das offizielle SDK in Ihren eigenen Anwendungen nutzen können.

Das offizielle PHP SDK erleichtert das Leben von Facebook-Anwendungsentwicklern, die sich für die Programmiersprache PHP entschieden haben, ungemein: Vorgefertigte Methoden zur Autorisierung von Anwendungen und zum Zugriff auf die Graph API helfen, Fehler zu vermeiden und rascher zu Ergebnissen zu kommen. Vielleicht fragen Sie sich nun, warum das PHP SDK erst in einem Anhang zu diesem Buch vorgestellt wird! Nun, es liegt in der Natur der Sache, dass der Einsatz des SDKs die Anwendungsentwicklung auf einer höheren Abstraktionsebene ermöglicht. Anstatt sich mit HTTP-Requests mittels `curl()` zu beschäftigen, werden z. B. Zugriffe auf die Graph API direkt vom SDK ausgeführt. Durch den vorläufigen Verzicht auf das SDK konnten Sie wahrscheinlich ein tieferes Verständnis der zugrunde liegenden Mechanismen erlangen. Außerdem macht es der bisher im Buch gewählte »Low-Level«-Ansatz Entwicklern auf anderen Development Stacks leichter, die Beispiele in »ihrer« Programmiersprache umzusetzen. Für die produktive Entwicklung eigener Facebook-Anwendungen sollten Sie künftig aber jedenfalls auf das PHP SDK (bzw. seine inoffiziellen Pendants auf anderen Stacks) zurückgreifen!

Dieses Kapitel möchte Ihnen nur einen Überblick über die Verwendung des PHP SDKs geben – alle Details zu den verfügbaren Methoden finden Sie in der Dokumentation unter *https://developers.facebook.com/docs/reference/php/*.

C.1 Download und Installation des SDKs

Die jeweils neueste Version des SDKs (zum Zeitpunkt der Entstehung dieses Buches war Version 3.1.1 aktuell) kann kostenlos vom Github-Repository unter *https://github.com/facebook/php-sdk* heruntergeladen werden. Ambitionierte Entwickler können hier auch Fehlerbehebungen und Erweiterungen des SDKs ins Repository

hochladen – das Facebook-Plattform-Team nimmt immer wieder Verbesserungen aus der Community in das offizielle SDK auf!

Extrahieren Sie die heruntergeladene ZIP-Datei, und laden Sie die Dateien im Unterordner *src* auf Ihren Webserver in ein neues Verzeichnis namens *sdk*. Das SDK besteht derzeit aus nur drei Dateien (*base_facebook.php*, *facebook.php* und *fb_ca_chain_bundle.txt*).

Zum Einbinden des SDKs in Ihre eigene Anwendung ist folgender Code notwendig, den Sie am besten in eine neue Datei *index.php* im Root-Verzeichnis Ihres Webservers kopieren:

```
<?
include_once("sdk/facebook.php");
$config = array();
$config['appId'] = '214728715257742';
$config['secret'] = 'b93bff...';
$config['fileUpload'] = false;

$facebook = new Facebook($config);
?>
```

Listing C.1 Einbinden und Initialisieren des PHP SDKs

Beim Initialisieren des SDK-Objekts `Facebook` übergeben Sie dabei die Anwendungs-ID sowie das App Secret. Dieses wird von allen SDK-Methoden danach automatisch verwendet.

Hinweis: Das offizielle PHP SDK wird regelmäßig von Facebook aktualisiert. Um ein reibungsloses Funktionieren des SDKs zu gewährleisten, sollten Sie Ihre Anwendungen zeitnah nach der Veröffentlichung einer neuen Version des SDKs aktualisieren!

C.2 Autorisierung der Anwendung

Hat ein Benutzer Ihre Facebook-Anwendung bereits autorisiert, können Sie mit der SDK-Methode `getUser()` die Benutzer-ID des Anwenders ermitteln:

```
$uid = $facebook->getUser();
```

Hat der Benutzer die Anwendung noch nicht autorisiert, liefert `getUser()` anstelle der Benutzer-ID 0 zurück. In diesem Fall können Sie den Benutzer zum OAuth-Dialog weiterleiten, wo er aufgefordert wird, Ihre Anwendung zu bestätigen. Anstatt wie bisher die URL zum OAuth-Dialog selbst zu konstruieren, hilft Ihnen hierbei die SDK-Methode `getLoginUrl()`. Das Array, das Sie an diese Methode übergeben, enthält im

Parameter `scope` die gewünschten erweiterten Berechtigungen und im Parameter `rediret_uri` die Webseite, auf die Facebook den Benutzer nach erfolgter Autorisierung weiterleiten soll:

```
$params = array(
   'scope' => 'email',
   'redirect_uri' => 'https://apps.mycompany.com/index.php'
  );
$loginUrl = $facebook->getLoginUrl($params);
```

Das folgende Code-Beispiel zeigt, wie Sie mit `getUser()` erkennen können, ob der aktuelle Benutzer Ihre Anwendung autorisiert hat – und falls nicht, wie Sie ihn zum OAuth-Dialog weiterleiten:

```
<?
include_once("sdk/facebook.php");
$config = array();
$config['appId'] = '214728715257742';
$config['secret'] = 'b93bff303c6199d959ad84659bf3f7bd';

$facebook = new Facebook($config);
// Ist der Benutzer bereits identifzierbar (=App autorisiert)
if (!($uid = $facebook->getUser())) {
  // Nein - Weiterleitung zum OAuth-Dialog.
  $params = array(
    'scope' => 'email',
    'redirect_uri' => 'https://apps.mycompany.com/index.php'
  );
  $loginUrl = $facebook->getLoginUrl($params);
  header("Location: ".$loginUrl);
} else {
  // Ja, Benutzer ist über seine ID identifzierbar
  print "Hello User #".$uid;
}
?>
```

Listing C.2 Identifikation des Benutzers und Autorisierung der Anwendung

Hinweis: Das PHP SDK kümmert sich dabei auch eigenständig um das Session-Management. Es ist also nicht mehr erforderlich, Access Token oder Benutzer-ID selbst in einer Session-Variablen abzulegen!

Diese Methoden des SDKs machen den Umgang mit der Benutzerautorisierung noch einfacher:

- `getAccessToken()` liefert das Access Token des aktuellen Benutzers zurück.
- `getSignedRequest()` liefert den mit dem App Secret entschlüsselten Inhalt des Signed Requests als PHP-Array zurück.
- `getLogoutUrl()` liefert eine URL zurück, die bei einem Redirect oder Klick den aktuellen Benutzer von Facebook abmeldet.

C.3 Zugriff auf die Graph API

Der Zugriff auf die Graph API erfolgt mit der SDK-Methode `api()`. Sie können der Methode folgende Parameter übergeben:

```
$result = $facebook->api($path, $method, $params);
```

Zum Ausführen eines GET-Zugriffs genügt es dabei, den gewünschten API-Endpunkt (ohne die vorangestellte Domain *https://graph.facebook.com*) im ersten Parameter zu übergeben. So lesen Sie etwa das Benutzerobjekt des aktuellen Anwenders aus:

```
$me = $facebook->api("/me");
```

Das Ergebnis enthält die Ausgabe der Graph API als PHP-Array – ein manuelles Decodieren des JSON-Outputs ist daher nicht mehr notwendig:

```
Array
(
    [id] => 609190863
    [name] => Michael Kamleitner
    [first_name] => Michael
    [last_name] => Kamleitner
    [link] => http://www.facebook.com/michael.kamleitner
    [username] => michael.kamleitner
    [hometown] => Array
        (
            [id] => 111165112241092
            [name] => Vienna, Austria
        )
    [location] => Array
        (
            [id] => 111165112241092
```

```
            [name] => Vienna, Austria
        )
    ...
```

Listing C.3 Rückgabewert der Methode »api()«

Ein POST-Zugriff zum Veröffentlichen von Informationen wird umgesetzt, in dem Sie als zweiten Parameter die Zugriffsmethode `POST` und im dritten Parameter ein Array mit allen benötigten Parametern übergeben. Das Access Token des aktuellen Benutzers wird vom SDK automatisch hinzugefügt. Zum Veröffentlichen eines Wall-Postings genügt also folgender Aufruf:

```
$result = $facebook->api("/me/feed",
                         "POST",
                         array("message" => "Testing"));
```

Das SDK gibt als Ergebnis ein PHP-Array mit der ID des veröffentlichten Wall-Postings zurück.

Auch der Upload von Binärdaten, etwa zum Veröffentlichen eines Fotos, kann mit dem PHP SDK einfach umgesetzt werden. Genauso wie `curl()` unterstützt die Methode `api()` dabei den Verweis auf eine Datei im File-System über ein vorangestelltes `@`-Symbol und kümmert sich selbstständig um die Codierung der binären Daten. Zuvor müssen Datei-Uploads allerdings mit der SDK-Methode `setFileUploadSupport()` aktiviert werden:

```
$facebook->setFileUploadSupport(true);
$result =
  $facebook->api("/me/photos",
                 "POST",
                 array("source" => "@./img/orangejuice.jpg",
                       "message" => "Orange Juice"));
```

Um veröffentlichte Inhalte wieder zu löschen, rufen Sie die Funktion `api()` mit der Methode DELETE und dem API-Pfad des zu löschenden Objekts auf:

```
$facebook->api("/10150665257130864", "DELETE");
```

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z