1 Entwickeln mit PhoneGap

1.1 Das Cross-Plattform-Problem

Bevor ich mit Ihnen in die praktische Entwicklung einsteige, möchte ich Ihnen von einem kurzen Gespräch berichten, das sich vor einiger Zeit in der IT-Abteilung eines mittelständischen Unternehmens abspielte.

Geschäftsführer: »Wir brauchen eine App.«

IT-Leiter: »Wow, ja gerne, welche Plattform?«

Geschäftsführer: »Wieso Plattform – mobil!«

IT-Leiter: »Ja, aber: Windows Phone, iOS, Android, BlackBerry, oder ...?«

Geschäftsführer: »Na alle! Oder nicht?«

IT-Leiter: »O.K. – Ich glaube, da haben wir ein Problem!«

Sie sehen, Anspruch und Wirklichkeit können sehr stark auseinanderliegen. Welcher IT-Leiter oder Entwickler möchte schon für eine einzelne App mehrere Entwicklungsprojekte für diverse mobile Plattformen auf die Beine stellen? Und welcher Geschäftsführer möchte das bezahlen?

Für dieses Problem bietet PhoneGap eine Lösung.

1.2 Wozu braucht man PhoneGap?

Die Zeiten von Apples App Store als Marktführer sind vorbei. Android hat als Plattform aufgeholt und der Kampf um Platz 3 im App-Markt ist noch nicht entschieden. Eines ist dabei aber sicher: Wer mobile Anwendungen nicht für eine abgegrenzte Kundschaft mit Mobilgeräten des gleichen Betriebssystems entwickelt, der hat mit einer Vielzahl von Zielplattformen zu tun. Er muss seine Apps so entwickeln, dass Kunden möglichst vieler Plattformen etwas damit anfangen können.

Für jede interessante Zielplattform eine proprietäre App zu entwickeln, kommt in der Regel nicht in Frage. Zu groß ist der Aufwand für mehrere parallele Entwicklungen in sehr unterschiedlichen Sprachen (siehe Tab. 1–1). Insbesondere für

Einzelentwickler und kleinere Firmen ist dies keine Option. Und selbst große Unternehmen möchten sich kaum mehrere Entwickler-Teams für ein einzelnes Produkt leisten.

Plattform	Entwicklungssprache
iOS	Objective-C
Android	Java (Dalvik VM)
BlackBerry	Java/C++
Windows Phone	C#, VB.NET, JavaScript
Nokia	C++, Flash

Tab. 1–1 *Einige wichtige Smartphone-Systeme und ihre Programmiersprachen*

Hier kommen die sogenannten *Web-Apps* ins Spiel. Dies sind mobile Anwendungen, die auf Basis von HTML5, CSS3 und JavaScript erstellt werden und auf allen Smartphones laufen – im jeweiligen Webbrowser. Leider bietet sich diese Lösung nur bedingt an. Web-Apps werden Sie nämlich in keinen virtuellen Store der Smartphone-Anbieter einstellen können, da dort ja nur *native Apps* für die jeweilige Plattform angeboten und verkauft werden können, z.B. iOS-Apps im Apple Store. Die Kunden suchen aber hauptsächlich in den App Stores nach Anwendungen für ihre Smartphones.

PhoneGap bietet hier eine Lösung: Sie erstellen mittels HTML5, CSS und JavaScript eine Web-App und nutzen PhoneGap dann als *Packetierer*, um die Web-App für die jeweilige Zielplattform in eine native App umzuwandeln. Nach dem Verpacken in eine iOS-Anwendung erhalten Sie z.B. eine ipa-Datei, die Sie an Apple schicken können. Ähnlich funktionert es auf allen anderen Plattformen.

Mittlerweile müssen Sie bei diesem Vorgehen nicht mal mehr auf den Hardwarezugriff (Kamera, GPS, andere Sensoren) auf den Smartphones verzichten. PhoneGap unterstützt Sie dabei durch eine eigene API.

Damit Sie ein Gefühl dafür bekommen, mit welchen Mitteln all dies umgesetzt wird, folgt nun ein Überblick über die Architektur von PhoneGap.

1.3 Die Architektur von PhoneGap

Vereinfacht kann man PhoneGaps Arbeitsweise so erklären: Es ruft den Webbrowser einer Smartphone-Plattform auf und lässt die zuvor entwickelte Web-App darin ablaufen. Dabei unterdrückt PhoneGap alles, was an den Browser erinnert, also den Browserrahmen und alle Bedienelemente wie Internetadresszeile oder Menüs. Auf den ersten Blick haben wir es also nur mit einem Browser im Vollbildmodus zu tun, darin verpackt die mobile Applikation.

Damit Sie aber auch Zugriff auf die Hardware der Zielplattform bekommen, wird in den anzuzeigenden HTML-Seiten eine JavaScript-Bibliothek geladen. Der

Browser fängt dabei gewisse Events ab, die dann in native Aufrufe der jeweiligen Plattform überführt werden. Damit ist auch der Zugriff auf die Hardware möglich.

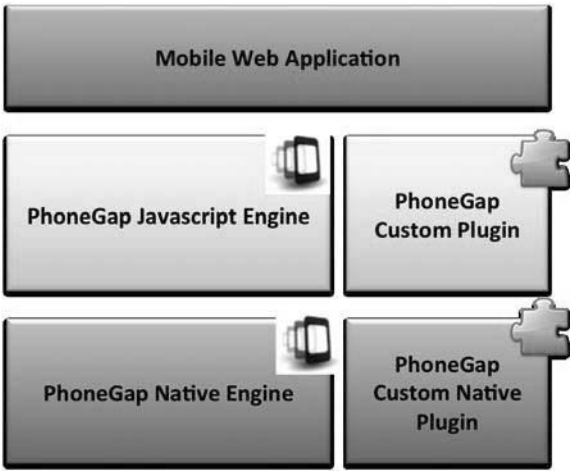

Abb. 1–1 *Die Architektur von PhoneGap*

Etwas genauer beschrieben, verhält es sich wie folgt: PhoneGap besteht aus zwei wesentlichen Bausteinen. Der erste Baustein ist eine Bibliothek namens cordova.js (die *JavaScript Engine)*, mit der Ihre HTML/CSS/JavaScript-Webapplikation arbeitet. Diese Bibliothek wird in der Webapplikation eingebunden und stellt ihr eine API für plattformunabhängige Zugriffe auf Smartphones zur Verfügung. Die JavaScript-Bibliothek interagiert dann mit dem zweiten Baustein von PhoneGap, der sogenannten *Native Engine*. Diese Engine existiert in verschiedenen Implementierungen, jeweils eine pro mobiler Zielplattform, und ermöglicht die Abfragen an die Sensoren für das jeweilige Betriebssystem.

Wenn die vorhandenen Zugriffe auf die Hardware per Native Engine nicht ausreichen, gibt es die Möglichkeit, sie durch eigene *Plugins* zu erweitern. Mit diesem Konzept können zusätzliche Zugriffe auf die Hardware entwickelt werden. Dabei gilt für jedes Plugin: Es wird sowohl eine JavaScript-Datei als auch eine plattformabhängige Klasse benötigt. Die JavaScript-Datei stellt den Aufruf der Funktionalität in PhoneGap dar. Damit das jeweilige Smartphone-System diesen Aufruf umsetzen kann, wird noch eine Implementierung des Plugins in der Native Engine benötigt. Daher muss der JavaScript-Layer nur einmal geschrieben werden. Der native Layer in den zu unterstützenden Plattformen ist aber jeweils separat. Wie einfach die Nutzung von Plugins ist, erfahren Sie in Kapitel 7.

Nachdem nun die Architektur und das Wirken von PhoneGap erklärt sind, möchte ich noch auf den Doppelnamen eingehen.

1.4 PhoneGap oder Apache Cordova?

Wenn Sie vor diesem Buch schon etwas über PhoneGap gelesen haben, werden Sie wahrscheinlich auch der Bezeichnung »Apache Cordova« begegnet sein.

Man könnte etwas vereinfacht sagen, dass Apache Cordova der neue Name von PhoneGap ist. Und man könnte hinzufügen, dass er sich bisher nicht wirklich durchgesetzt hat, denn in Artikeln und Konferenzvorträgen – und auch auf Buchtiteln – liest man nach wie vor: »PhoneGap«.

Wie es zu diesem Nebeneinander von »PhoneGap« und »Apache Cordova« kam und warum wir damit weiterhin leben müssen, möchte ich im Folgenden erläutern. Lassen Sie mich dazu kurz die Geschichte dieser Software erzählen.

Die Firma Nitobi wurde 1998 in Vancouver, Kanada, gegründet. An ihrem Anfang standen rund 15 Personen, die sehr früh die Mächtigkeit von JavaScript erkannten und schon vor dem iPhone an Ideen arbeiteten, um das mobile Web voranzubringen. Ihr Ziel war es, die Einfachheit der Webapplikationsentwicklung mit mobilen Geräten zu verbinden. Das Projekt, das sie dazu ins Leben riefen, nannten sie *PhoneGap*. Am Anfang tat PhoneGap nicht mehr, als Webseiten, die in einem lokalen Ordner lagen, auf mobilen Geräten anzuzeigen. Damit konnten Webentwickler plötzlich einfache Apps erstellen (z.B. Eventkalender, Produktinfos, Anleitungen).

Nachdem PhoneGap als sogenannter Packager gestartet war, fehlte aber noch ein wichtiges Feature: der *Zugriff auf die Hardware* der Geräte. Es musste möglich sein, eine Kamera zu nutzen oder auch Audiodaten aufzuzeichnen. Damit wurde eine Domäne betreten, die bisher nativ geschriebene Programme vorbehalten war. Mit dieser Anforderung entstand der zweite Teil von PhoneGap – der Hardwarezugriff mittels API.

Der Wunsch nach immer mehr Erweiterungen mündete schließlich in einer *Plugin*-Struktur. Diese gibt dem Entwickler die Möglichkeit, selber Funktionen von nativen Plattformen hinzuzufügen. Mehr dazu werden Sie in Kapitel 7 zum Einsatz von Plugins erfahren.

Von dem ersten Prototyp, der auf dem iPhoneDevCamp in San Francisco entwickelt wurde, bis zur Version 1.0 vergingen nicht einmal drei Jahre. Dabei verfolgte Nitobi immer das Ziel, PhoneGap als Open-Source-Software zu entwickeln. Hersteller wie BlackBerry oder Microsoft beteiligten sich und stellten Arbeitskraft für das Projekt zur Verfügung. Wie schnell sich das Projekt bis zur Version 1.0 entwickelte, zeigt die folgende Tabelle:

Datum	Event
August 2008	iPhoneDevCamp, San Francisco
15.10.2008	Unterstützung von Android
23.10.2008	Unterstützung von BlackBerry
25.02.2009	PhoneGap 0.6.0, erste stabile API-Version
April 2009	Gewinner der Web 2.0 Expo LaunchPad
02.04.2009	PhoneGap 0.7.2, Android und iPhone API sind gleich.
21.08.2009	Unterstützung von Windows Mobile
16.09.2009	Unterstützung des Nokia WRT (S60)
13.10.2009	PhoneGap 0.8.0+, wird von Apple für den App Store freigegeben.
November 2009	Auf der Konferenz InfoWorld wird PhoneGap als »top emerging enterprise technology« in der Kategorie "cross-platform mobile app development" ausgezeichnet.
18.11.2009	Sony Ericsson integriert PhoneGap in das »WebSDK«.
12.07.2009	Support für Palm hinzugefügt
19.07.2010	Symbian fügt PhoneGap den Web Extensions der Symbian^3-Plattform hinzu.
12.07.2011	PhoneGap 1.0.0rc1 freigegeben
29.07.2011	PhoneGap-Tag in Portland, Oregon, Start-Party der Version 1.0

Tab. 1–2 *Die Geschichte von PhoneGap bis zur Version 1.0*

Am 3. Oktober 2011 sorgte dann ein Blogpost von Nitobi für Aufregung in der Open-Source-Community, hier ein Auszug:

»...Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap Open Source HTML5 Mobile App Platform, Accelerates Adobe's HTML5 and Web Standards Strategy...«

Adobe hatte also Nitobi gekauft, vermutlich mit dem Ziel, es als Ersatz für Flash in seinen Produkten zu verwenden. Mit diesem Kauf kam sofort die Befürchtung auf, dass PhoneGap keine Open-Source-Zukunft mehr haben würde. Adobe stellte aber schnell klar, dass PhoneGap weiterhin Open Source bleiben solle und übergab den Code an die Apache Software Foundation. Die Foundation überführte PhoneGap dann in ein Incubator-Projekt und änderte den Namen zunächst in Apache Callback und mit der Version 1.4 dann in *Apache Cordova*. Heute ist Apache Cordova ein sogenanntes Top Level Project der Apache Foundation.

Adobe arbeitet allerdings weiterhin am Projekt mit eigenen Entwicklern mit und gibt neuen Code an das Cordova-Projekt. Gleichzeitig verwendet Adobe den alten Namen PhoneGap für eigene Aktivitäten weiter. So trägt Adobes Cloud-basierter Buildservice den Namen »PhoneGap Build«. Außerdem betreibt Adobe weiterhin die Website *www.phonegap.com* und bietet dort Dokumentation, Beispiele und das Framework selbst an. Wenn Sie PhoneGap dort herunterladen, erhalten Sie ein Apache-Cordova-Paket – oder wenn Sie es so nennen wollen: eine Distribution – mit allen Plattformen. Auf der Apache-Seite dagegen, muss man die Plattformen einzeln herunterladen.

Sie können sich also zur besseren Unterscheidung merken, dass Apache Cordova die Codebasis des Projekts ist und PhoneGap eine Distribution von Adobe.

In diesem Buch wird im Folgenden hauptsächlich der populärere Name »PhoneGap« verwendet und nicht die in der Regel genauso korrekte Bezeichnung »Apache Cordova«.

Aber Achtung: Manchmal ist »PhoneGap« wirklich die einzig korrekte Wahl, nämlich dann, wenn es um die Adobe-eigenen Aktivitäten geht, z.B. bei »PhoneGap Build«. Und manchmal muss es wiederum eindeutig »Cordova« heißen – etwa bei einigen Dateien und Verzeichnissen (`cordova-2.0.0.js`, `Cordova.plist`) und API-Aufrufen (`device.cordova`), denen Sie auch im Verlauf dieses Buches begegnen werden.

Mit dem Namenswechsel bzw. dem Nebeneinander der beiden Namen haben das Projekt und die Entwickler sicherlich noch einige Zeit zu kämpfen. Wichtiger als diese Unklarheit ist aber: Das Projekt ist weiterhin Open Source und wird es unter der Federführung der Apache Software Foundation auch bleiben.

Abb. 1–2 *Die Apache-Cordova-Webseite*

Abb. 1–3 *Die PhoneGap-Website*

1.5 Zusammenfassung

Jetzt kennen Sie die Geschichte von PhoneGap bzw. Apache Cordova sowie deren Aufbau und Arbeitsweise. Sie haben gelernt, dass Web-Apps eine Möglichkeit sind, um eine mobile Anwendung auf verschiedenen Zielplattformen laufen zu lassen. Sie wissen jetzt aber auch, dass es Web-Apps beim Verkauf über App Stores und beim Hardwarezugriff nicht mit nativen Apps aufnehmen können. Dafür ist PhoneGap bzw. Apache Cordova als Brückentechnologie gedacht. Wir halten also fest:

PhoneGap ist ...

... eine Software, die es ermöglicht, HTML-Code nebst JavaScript auf verschiedenen Smartphone-Plattformen als native App auszuführen. Zudem kann man per JavaScript auf Gerätefeatures zugreifen.

Aber:

PhoneGap ist nicht ...

... dazu gedacht, dass es das UI der Zielplattform widerspiegelt. Es hat also nicht das Look-and-Feel des jeweiligen Betriebssystems. Es ist auch zu beachten, dass durch das Ausführen auf dem lokalen Browser nicht die gleiche Performance wie bei einer nativen App zu erwarten ist.

Sie wissen nun, dass PhoneGap/Apache Cordova keine Oberflächen der Plattformen nachahmen will. Daher ist Ihnen dort die Wahl überlassen, selber eine Lösung zu entwickeln oder ein UI-HTML-Framework zu nutzen. Eine kleine Auswahl der bekanntesten Vertreter werde ich Ihnen in Abschnitt 2.3.1 an die Hand geben (jQuery Mobile, Sencha Touch, iUi, BBUi)

Jetzt wünsche Ich Ihnen viel Spaß beim Entwickeln Ihrer App-Ideen mit PhoneGap/Apache Cordova. Im nächsten Kapitel gibt es zunächst einen Überblick über die verschiedenen Phasen der App-Entwicklung. Im Anschluss beginnen wir in Kapitel 3 mit der praktischen Arbeit und widmen uns der Installation und Konfiguration der Entwicklungstools.