Horst Klier

»Web trifft App – mit PhoneGAP zur nativen App mit Webtechniken

Der mobile Markt boomt. Webdesigner und -entwickler mit einem Blick für die Zukunft haben sich bereits mit den Spezialitäten mobiler Endgeräte befasst. Doch damit ist noch nicht das Ende erreicht. Dank PhoneGAP ist der Sprung zur nativen App für iPhone und Co. nicht mehr weit.

```
<!DOCTYPE HTML>
<html>

<head>
<meta id="vp" name="viewport" content="" />
<script type="text/javascript" charset="utf-8" src="phonegap-1.0.0.js">
<script>
function onLoad() {
  document.addEventListener("deviceready", onDeviceReady, false);
  document.addEventListener("resume", doResumeApp, false);
  document.addEventListener("pause", doPauseApp, false);
}
function onDeviceReady() {
    Init();
}
</script>
```

Abbildung 1: Typischer Beginn einer PhoneGAP-index.html. Über das eingebundene PhoneGAP-Javascript greift man auf die API zu.

Jedes moderne Smartphone hat heute einen integrierten Browser. Entwickler können diesen auch einfach innerhalb Ihrer Apps benutzen, indem sie einen Teil des Bildschirms als WebView (so nennt es sich bei Android, bei iOS UIWebView) definieren. In diesem Browserfenster kann dann normales HTML mit Javascript angezeigt werden. Die Idee von PhoneGAP ist nun, diesen WebView auf den ganzen Bildschirm auszudehnen und die App dann komplett mit Webtechniken zu entwickeln. Zusätzlich wird eine API (Programmierschnittstelle) per Javascript bereitgestellt, damit man auf die vollen Funktionen des Telefons zugreifen kann. Fotos mit der Kamera, Abfrage des Adressbuchs, Auslesen der Bewegungssensoren und vieles mehr sind so direkt möglich.

Im Grunde funktioniert es so: Man legt einen Ordner www an und speichert darin eine index.html sowie alle anderen benötigten Dateien. PhoneGAP packt das alles in eine App, die beim Start die index.html anzeigt. Per Javascript hat der Entwickler dann die Kontrolle über den Ablauf – allerdings eben auch nur per Javascript. Websites, die mit dynamisch am Server erzeugten Seiten arbeiten, können nicht genauso am Smartphone laufen. Moderne Seiten, die auch im Web bereits die Inhalte per Javascript laden, sind dagegen kaum ein Problem. Damit eignet sich PhoneGAP wunderbar immer dann, wenn eine Web-App zur mobilen App werden soll.

Dabei ist es zum Teil sogar leichter als im

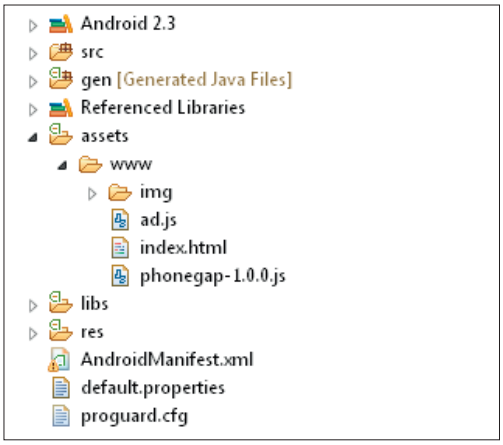

Abbildung 2: Struktur eines PhoneGAP-Projektes in Eclipse. Die eigentliche App steckt im Verzeichnis „www“.

DER AUTOR

Horst Klier entwickelt als Internet-Flüsterer Internet-Projekte und Handy-Apps.

Abbildung 3: Blutdruck-Erfassung als Web-App im Internet-Explorer (www.blutdruckdaten.de) und auf dem iPhone

Web. Muss man sich dort nämlich evtl. noch mit uralten Browsern quälen, die noch unterstützt werden wollen, sind die mobilen Varianten dank WebKit richtig fortschrittlich. Apple hat hier auf dem iPhone vorgelegt, dass es eine wahre Freude ist. CSS-Animationen laufen dank Hardwarebeschleunigung butterweich und als Webentwickler kann man aus dem Vollen schöpfen, ohne Gedanken an updateunwillige Internet-Explorer-Nutzer zu verschwenden. Allerdings sollte man sich davor hüten, nun ungebremst jede App für jede Plattform anzubieten. Jedes System hat seine Eigenheiten, und wenn es nur unterschiedliche Icon-Größen sind. Ohne genaue Kenntnis und entsprechende Anpassung wird einem nicht der erhoffte Erfolg beschieden sein. So sollte man z. B. die bei Apple so hervorragenden CSS-Animationen bei Android besser sein lassen. Solches „eye candy“ wird bei Google sehr stiefmütterlich behandelt. Überhaupt gibt es schlicht Unterschiede im Bedienkonzept zwischen den Systemen. Android-Geräte haben einen Menü- und einen Zurück-Knopf, das iPhone nicht. Das hat eine fest definierte Bildschirmgröße und -auflösung (bzw. durch das Retina-Display zwei verschiedene Auflösungen), bei Android gibt es Hunderte.

In der Praxis

Je nach Zielsystem arbeitet man in der typischen Entwicklungsumgebung. Wer iPhone-Anwendungen erstellen will, benötigt zwangsweise einen Mac und nutzt dort typischerweise xCode. Android-Anwendungen werden dagegen meist mit Eclipse als IDE (Integrated Develop Environment – Integrierte Entwicklungsumgebung) erstellt. Den Umgang damit und alles, was dazugehört, nimmt PhoneGAP einem nicht ab. Allerdings gibt es gute „Getting started“-Anleitungen, sodass man keine Berührungsängste haben sollte. Wer Apples Plattformen ansprechen will, kommt um eine Registrierung als Entwickler bei Apple nicht herum und muss sich auch mit den entsprechenden Methoden zur Signierung und Auslieferung der Apps befassen. Ohne den iTunes App Store kommen die Apps sonst nicht zum Benutzer. Bei Android ist das etwas leichter, aber auch da werden die Apps signiert, wenn sie in

PhoneGap

PhoneGAP ist Open Source und steht unter der MIT-Lizenz. Am 29.07. erschien nach langer Entwicklungszeit Version 1.0.0, wobei es bereits seit Langem produktiv eingesetzt werden kann.
» www.phonegap.com

Begriffe
Wie das bei neuen Dingen meist so ist, gibt es verschiedene Begriffe, die das Gleiche meinen, aber auch gleiche Begriffe für verschiedene Dinge. So gibt es iOS-Entwickler, die eine App als „hybrid" bezeichnen, wenn sie sowohl am iPad als auch am iPhone läuft. Die eigentlich dafür vorgesehene Bezeichnung von Apple ist „universal". Hier im Text meint „hybrid" Apps auf verschiedenen Plattformen, wobei mit PhoneGAP auch universale Apps möglich sind.
Ebenso uneindeutig ist der Begriff Web-App. Damit kann eine Anwendung im Browser auf normalen PC gemeint sein, die natürlich auch auf einem mobilen Browser aufgerufen werden kann. Apple hatte bei der Einführung des iPhone ursprünglich nur Web-Apps vorgesehen. Dementsprechend kann man solche mobilen Web-Apps auf iOS-Geräten auch als Symbol ablegen und für den Anwender besteht kein großer Unterschied zu einer nativen App. Eine PhoneGAP-App unterscheidet sich im Grunde von einer solchen mobilen Web-App nur wenig, ist aber trotzdem eine native App.

den offiziellen Market sollen. Generell tut sich leichter, wer die Zielplattform kennt, und sei es nur aus Nutzersicht. So gibt es auch den Service PhoneGAP Build, dem man nur eine ZIP-Datei mit dem www-Verzeichnis übergibt und auf Wunsch für alle Plattformen fertige Apps erhält, aber wirklich sinnvoll ist das auch nur, wenn man genügend Geräte zum Testen hat.

Praxis-Beispiel: Hybride Blutdruck-Erfassung in der Cloud

Trotz dieser Einschränkungen eignet sich PhoneGAP hervorragend für hybride Apps, also Apps, die nahezu überall laufen, in dem Fall im Web und als App auf den verschiedenen Geräten. Als Beispiel sei hier der Dienst BlutdruckDaten genannt. Dieser bietet eine relativ einfache Möglichkeit, seine Blutdruckwerte in einer zentralen Datenbank zu speichern und als Diagramm anzuzeigen. Die Datenbank läuft dabei auf einem Server. Die Eingabe, Verwaltung und Anzeige der Daten ist im Web eine klassische HTML- und Javascript-Lösung in einer einzigen HTML-Datei. Um daraus eine App zu machen, bestand das größte Problem in der Umstellung des Benutzerinterface auf die kleineren Bildschirme und in der Bedienung per Touch statt Maus und Tastatur. Die eigentliche Programmlogik konnte komplett beibehalten werden und auch die Routinen zur Ausgabe des Diagramms sind identisch. Seit dafür ein HTML-Canvas genutzt wird, wurde das sogar deutlich einfacher, weil in der App keine Spezialanpassungen für den Internet-Explorer nötig sind. Im Ergebnis existiert die Anwendung nun im Web und als App auf den beiden wichtigsten Plattformen iPhone und Android. Dabei sind die Apps nicht nur eine Notlösung, was zahlreiche 5-Sterne-Bewertungen und ein „Sehr gut" des Android-Magazins beweisen.

Fazit

PhoneGAP ermöglicht native Apps mit Webtechniken. Damit erschließen sich Webentwickler den Zugang zur Welt der Apps. Die wirklich großen Erfolge feiert PhoneGAP im Bereich der hybriden Entwicklung als Zugang zur Cloud. Es ist schlicht teuer und zeitintensiv, verschiedene Programmstände für verschiedene Plattformen zu pflegen. Wer eine Cloud-Anwendung mit Web-Interface betreibt, der erhält mit PhoneGAP den Schlüssel zur mobilen Welt. ¶

Alternativen

appcelerator®

Titanium Mobile von Appcelerator ermöglicht die Entwicklung nativer Apps für Android und iPhone. Entwickelt wird mit Javascript, wobei man Zugriff auf die nativen Elemente des jeweiligen Geräts hat. Wenn man so will, eine Art Zwischenlösung. Man ist nicht vom Browser des Geräts abhängig, kann aber trotzdem mit Javascript entwickeln. Die Apps sind dadurch auf mehreren Plattformen lauffähig. Im Moment sind das iOS und Android.
» *www.appcelerator.com*

Corona® SDK

Corona SDK von Ansca ist optimiert auf die Entwicklung von Spielen. Programmiert wird in Lua, das ist eine sehr einfach zu erlernende Scriptsprache. Viele fertige Bibliotheken machen erste Erfolge leicht. Eine Physik-Engine ermöglicht Spiele wie Angry Birds ohne viel Aufwand. Rund um den Entwicklungsprozess gibt es zahlreiche Tools zur Unterstützung, auch von Drittanbietern. Eine gewisse Nähe zu Flash ist vorhanden, weswegen ehemalige Flash-Entwickler eine der Zielgruppen für die Umgebung sind. Umsetzungen von Flash-Spielen als Apps sollen sehr einfach möglich sein. Die fertigen Apps laufen dann unter iOS oder Android. Corona SDK ist allerdings kostenpflichtig.
» *www.anscamobile.com*

Plattformen

PhoneGAP unterstützt:
» iOS (iPhone & Co.)
» Android
» Windows Phone 7
» BlackBerry
» webOS
» symbian
» bada (Samsung)