AngularJS & Ionic Framework

Hybride App-Entwicklung mit JavaScript und HTML5

Bearbeitet von
Bengt Weiße

1. Auflage 2016. Buch. 345 S. Hardcover
ISBN 978 3 446 44671 7
Format (B x L): 18 x 24,7 cm
Gewicht: 742 g

HANSER

Leseprobe

Bengt Weiße

AngularJS & Ionic Framework

Hybride App-Entwicklung mit JavaScript und HTML5

ISBN (Buch): 978-3-446-44671-7

ISBN (E-Book): 978-3-446-44807-0

Weitere Informationen oder Bestellungen unter

http://www.hanser-fachbuch.de/978-3-446-44671-7

sowie im Buchhandel.

Inhalt

Vorwort

Dies ist die erste Auflage meines Buches *AngularJS und das Ionic Framework*. Als Entwickler war ich bereits bei einem der führenden Anbieter von eCommerce Software tätig. Als technischer Leiter bei der FLYACTS GmbH bin ich für die Entwicklung hybrider Apps verantwortlich und kam so unweigerlich zu AngularJS und dem Ionic Framework.

Das Buch richtet sich vor allem an Webentwickler, die in das Thema AngularJS oder Ionic einsteigen wollen oder bereits damit arbeiten. Für Webagenturen kann sich aus diesem Bereich ein weiteres Standbein entwickeln, da sie mit bereits bekannten Mitteln mobile Apps erstellen können. Natürlich soll auch jeder Interessierte seinen Wissensdurst mit diesem Werk stillen. Des Weiteren kann es als Begleitwerk zu schulischen oder hochschulischen Veranstaltungen dienen.

Alle beschriebenen Funktionalitäten der Technologien beziehen sich dabei auf die Versionen 1.0 und 1.1 des Ionic Frameworks und 1.6.3 und 1.6.4 der Ionic CLI. In Version 1.0 kommt AngularJS 1.3.13 und in 1.1 AngularJS 1.4.3 zum Einsatz.

Die Technologien selbst entwickeln sich äußerst schnell weiter, sodass Funktionen in der Zwischenzeit in aktuelleren Versionen hinzugekommen oder weggefallen sein können.

Die Kapitel beinhalten zahlreiche praktische Beispiele, weshalb bei der Umsetzung auf die oben genannten Versionen Rücksicht genommen werden sollte.

Als Entwickler von hybriden und mobilen Apps konnte ich die Entwicklung des Ionic Frameworks hautnah und tagtäglich miterleben. Deshalb setzt dieses Buch nicht nur auf die Vermittlung theoretischer Fakten, sondern führt den Leser zielstrebig zur Umsetzung einer eigenen Anwendung.

Zusätzlich gehe ich in Form von Hinweisen oder Tipps auf viele Probleme und Knackpunkte ein, die es bei der Programmierung mit AngularJS und Ionic gibt, und zeige, wenn möglich, Lösungen auf.

https://github.com/AngularIonic

Hier stehen alle Quelltexte aus dem Buch zur Verfügung:

- die Listings: *https://github.com/AngularIonic/Buch*
- die komplette Beispielanwendung: *https://github.com/AngularIonic/MusikMax*

An dieser Stelle möchte ich mich ganz besonders bei Anja Heumann für die Erstellung der schönen Grafiken und für ihre Korrekturarbeit bedanken.

Ein weiteres Dankeschön geht an das Ionic-Team selbst, insbesondere an Mike Hartington. Sie hatten stets ein offenes Ohr für meine Fragen.

Jena, Januar 2016

Bengt Weiße

3 Apache Cordova und Adobe PhoneGap

3.1 Einführung

Eine nativ entwickelte mobile Anwendung zeichnet sich durch den weitreichenden Zugriff auf die Funktionalitäten des Gerätes aus. Diesen erhält sie entweder durch die Entwicklung mit der vom Betriebssystem vorgeschriebenen Programmiersprache oder durch den Einsatz der systemspezifischen Tools und Baukästen. Unter Android werden diese zwei Komponenten durch die Programmiersprache Java und das Android SDK (Software Development Kit) repräsentiert. Dem gegenüber stehen die Programmiersprachen Objective-C und Swift sowie die XCode IDE (Integrated Development Environment) für das Betriebssystem iOS von Apple. Aufgrund der verschiedenen Architekturen und Technologien müssen native Apps für jede Plattform entwickelt werden. Dadurch kann sich der Entwickler auch bei den vorgegebenen UI-Elementen des Betriebssystems bedienen. Darunter fallen beispielsweise Checkboxen, Listen, Tooltipps oder das Action-Sheet auf iOS.

Wäre es nun möglich, eine App mit einfachen Webtechnologien, wie JavaScript, HTML und CSS, zu programmieren, müsste diese Anwendung trotzdem in die für das Betriebssystem passende Form gebracht werden. Ein System basierend auf JavaScript-, HTML- und CSS-Dateien ist dafür nicht ausreichend.

An diesem Punkt setzen Cordova und Adobe PhoneGap an. Mit beiden Systemen lassen sich Webanwendungen einfach und schnell in eine App verwandeln. Mithilfe einer Code-Basis können dadurch gleich mehrere Plattformen bedient werden.

Aufgaben von Apache Cordova/Adobe PhoneGap:

- Aufbau und Strukturierung der Anwendung
- Bereitstellung von Schnittstellen für die Verwendung nativer Funktionen (z. B. Benachrichtigungen, Kamera, Dateisystem)
- Finales Erzeugen der App für die spezifischen Plattformen (z. B. .apk-Dateien für Android, .ipa-Dateien für iOS)
- Bereitstellung von Tools zum Testen und Emulieren

Beide genannten Systeme benötigen die Entwicklungsumgebungen der zu unterstützenden Plattform, damit sie aus dem Quellcode eine mobile Anwendung bauen können. Dies wird

hauptsächlich bei Apps für iOS zu einer Hürde, da die Entwicklungsumgebung XCode nur für Mac OS zur Verfügung steht. Aus diesem Grund werden weitere Erklärungen und Beispiele anhand von Android dargelegt.

Die angesprochenen Schnittstellen sind begrenzt und bieten nur grundlegende Zugriffe auf das Gerät und dessen Sensoren. Aus diesem Grund gibt es eine Vielzahl von Plugins, welche die Funktionspalette erweitern. Falls unter den vorhandenen Plugins eine gewünschte Funktionalität nicht verfügbar ist, können auch eigene Erweiterungen entwickelt und veröffentlicht werden. Diese müssen jedoch zwingend mit den systemspezifischen Programmiersprachen entwickelt werden. Dennoch ist der Aufwand deutlich geringer, als zwei komplette Anwendungsprojekte zu warten.

Ein großer Unterschied zwischen Cordova und PhoneGap existiert derzeit nicht. Im Grunde basiert PhoneGap auf Cordova, behält sich aber vor, die nativen Schnittstellen bzw. APIs zu erweitern. Bisher wird jedoch der Cordova-Programmcode fast eins zu eins übernommen, was auch anhand der Versionierung beider Projekte ersichtlich wird. Der Grund dafür lässt sich durch die gemeinsamen Wurzeln erahnen.

3.2 Entstehung und Entwicklung

Die Idee für ein System zur Erstellung von hybriden Anwendungen für mobile Systeme kam bereits im Sommer 2008 auf – nur knapp ein dreiviertel Jahr nachdem das erste iPhone auf dem deutschen Markt eingeführt wurde. Nach dem iPhoneDevCamp 2008 in San Francisco (einer iPhone-Entwickler-Konferenz) entstand die Vision, ein System zu entwickeln, welches native Funktionalitäten von mobilen Geräten Webanwendungen zur Verfügung stellt. Das Ziel war es, Anwendungen nur mit Webtechnologien umzusetzen. Auf der genannten Konferenz gelang es bereits, einer Web-App die Standortbestimmung des iPhones zur Verfügung zu stellen. Unter den Teilnehmern waren unter anderem die PhoneGap-Gründer Rob Ellis und André Charland. Letzterer war zu diesem Zeitpunkt Geschäftsführer des von ihm 1998 gegründeten Unternehmens Nitobi, welches dieses Vorhaben unter dem Produkttitel PhoneGap umsetzte. Schon im April 2009 wurde Nitobis PhoneGap mit dem People's Choice Award auf der Web 2.0 Expo ausgezeichnet.

Nach zwei Jahren kontinuierlicher Entwicklung zeichnete sich 2011 der nächste große Schritt für PhoneGap ab. Adobe bekundete Interesse an Nitobis Hauptprodukt und kaufte noch im selben Jahr das gesamte Unternehmen, um es der Apache Software Foundation zu spenden. Dort entsteht aus PhoneGap das Open-Source-Projekt Apache Cordova. Der ursprüngliche Name lautete Apache Callback. Dieser wurde jedoch verworfen, da Callback in der Softwareentwicklung ein zu generischer und gebräuchlicher Begriff ist. Parallel dazu existierte ein eigenes Projekt bei Adobe. Dabei wurde die alte Bezeichnung PhoneGap übernommen und es entstand Adobe PhoneGap. Bis heute bestehen beide Projekte, wobei PhoneGap auf Cordova basiert. Adobe behält sich jedoch vor, Veränderungen am Quellcode vorzunehmen, eigene Erweiterungen zu implementieren oder eigene Services aufbauend auf dem eigenen Produkt anzubieten. Jedoch halten sich die Änderungen derzeit in Grenzen, was die gleiche Versionierung beider Systeme erkennen lässt. Im April 2015 erschien sowohl von

Apache Cordova und als auch von Adobe PhoneGap die Version 5.0.0. Beide Systeme werden stetig weiterentwickelt. Besonders hervorzuheben ist die schnelle Reaktionszeit bei Anregungen oder auftretenden Problemen.

Bei den Lösungen von Adobe und Apache handelt es sich um sogenannte CLIs. CLI ist die Abkürzung für Command-Line-Interface, was zu Deutsch so viel wie Kommandozeilenschnittstelle bedeutet. Das heißt, dass die Steuerung und Nutzung über ein Eingabeaufforderungsfenster unter Windows bzw. einem Terminal unter Linux und Mac OS geschieht. PhoneGap bietet als Alternative einen eigenen Service namens PhoneGap Build an. Dort kann der Programmcode einer Anwendung hinterlegt werden, woraufhin der Service die finale Anwendung für die verschiedenen Plattformen baut.

3.3 Installation

Nach der kurzen Einführung in das Thema Cordova bzw. PhoneGap und einem kleinen Einblick in die Entstehungsgeschichte dieser Systeme soll dieser Abschnitt Einblick in die Einrichtung geben. Aufgrund der Verbreitung und Nutzung werden die Inhalte im Verlaufe des Buches anhand des Windows Betriebssystems 8.1 und das Cordova CLI 5.0.0 erläutert. Der Ablauf, die Voraussetzungen und die Installationsschritte sind für das PhoneGap CLI 5.0.0 gleich. Zusätzlich wird ein kurzer Überblick über das Setup auf Linux und Mac OS-Systemen gegeben.

Ein CLI muss, wie jede andere Anwendung, in einer Programmiersprache geschrieben sein und vom System interpretiert werden können, damit sie ausgeführt werden kann. Das Cordova CLI setzt auf die noch recht neue plattformübergreifende Laufzeitumgebung NodeJS. Mit NodeJS lassen sich Server- und Netzwerkanwendungen mit JavaScript realisieren. Da Cordova ein Open-Source-Projekt ist, werden auch nur Technologien benutzt, deren Quellcode öffentlich zur Verfügung steht. Als Erstes muss NodeJS auf dem Rechner installiert werden, falls es noch nicht vorhanden ist.

Im einfachsten Fall bietet NodeJS für das vorliegende Betriebssystem bereits einen Installer, der alles Nötige selbst einrichtet. Für Windows findet sich im Download-Bereich der NodeJS-Webseite bereits ein solcher Installer (*https://nodejs.org/download/* – NODEJS 0.12.4). Nach dem Herunterladen der .msi-Datei kann diese ausgeführt werden. Daraufhin startet der Installations-Assistent, wobei dort alle vorgeschlagenen Standardeinstellungen beibehalten werden können. Ist die Installation abgeschlossen, wird der Rechner neu gestartet. Neben NodeJS wird auch das Paketverwaltungssystem NPM (Node Package Manager) installiert, welches dazu dient, zusätzliche NodeJS-Module und -Anwendungen zu installieren, zu deinstallieren oder zu aktualisieren. Um zu überprüfen, ob die Installation funktioniert hat, kann eine neue Eingabeaufforderung geöffnet und folgende Befehle können nacheinander eingegeben werden:

```
node -v
```

(sollte die installierte NodeJS-Version zurückgeben, z. B. v0.12.4)

```
npm -v
```

(sollte die installierte NPM-Version ausgeben, z. B. 2.10.1)

Damit Cordova später die Apps in das passende Format bringen kann, muss die für die mobile Plattform entsprechende Entwicklungsumgebung installiert sein. Im Falle von Android handelt es sich um das Android SDK. Dieses kann im Entwicklerportal von Google heruntergeladen werden (*https://developer.android.com/sdk/index.html#Other* v24.2).

Nach der Installation öffnet sich der Android SDK Manager, mit dessen Hilfe weitere Tools und SDK-Versionen installiert werden können. Hier sollten die SDKs der zu unterstützenden Android-Versionen installiert sein. Da für das Bauen von iOS-Apps immer XCode gebraucht wird, kann dies nur auf einem Computer mit Mac OS geschehen. Die Einrichtung auf diesem Betriebssystem erfolgt nach dem gleichen Prinzip wie bei einem Windows-Rechner. Der einzige und komfortable Unterschied ist, dass XCode standardmäßig auf Mac OS-Geräten vorinstalliert ist.

Als letzte Voraussetzung wird das Git CLI benötigt. Bei Git handelt es sich um ein freies und performantes Datei-Versionierungssystem. Dieses erfreut sich momentan immer größerer Beliebtheit – viele Unternehmen lösen alte Strukturen auf und setzen auf dieses neue System. Daraus sind auch bekannte Online-Dienste wie GitHub, GitLab oder Bitbucket entstanden. GitHub wird vor allem von Open-Source-Projekten benutzt. Unter diesen befinden sich auch Apache und das Cordova-Projekt. Über die offizielle Git-Webseite lassen sich auch hier Installer für verschiedenste Betriebssysteme herunterladen (*https://git-scm.com/download* v1.9.5). Aktuell wird die Version 1.9.5 verwendet. Git bringt eine eigene Eingabeaufforderung mit – die Git-Bash. Sie bringt einige Tools und Konsolenbefehle mit, die so auf Windows nicht verfügbar wären. Wer nicht zwei verschiedene Eingabeaufforderungen nutzen will, kann während der Installation eine Einstellung wählen, sodass auch die Windows-Eingabeaufforderung alle zusätzlichen Funktionen erhält. Über das Git CLI kann Cordova eigene Beispielprojekte oder Module und Komponenten versionsabhängig herunterladen und verwenden.

Nun sind alle Vorbereitungen für die eigentliche Cordova-Installation getroffen. Als finalen Schritt wird eine Eingabeaufforderung benötigt. Dort kann nun der Installationsbefehl für Cordova eingegeben werden:

```
npm install -g cordova
```

Unter Mac OS und Linux-Betriebssystemen muss der Befehl mit root-Rechten ausgeführt werden, weshalb der Befehl wie folgt aussieht:

```
sudo npm install -g cordova
```

Nun sollte das Cordova CLI global auf dem Rechner verfügbar sein. Zur Kontrolle kann auch hier die installierte Version durch einen Befehl in der Eingabeaufforderung überprüft werden:

```
cordova -v
```

(sollte die installierte Version ausgeben, z. B. 5.0.0)

Um Adobes PhoneGap zu installieren, muss in den eben vorgestellten Kommandos `cordova` durch `phonegap` ersetzt werden. Damit ist die Installation des Cordova CLI abgeschlossen und die Entwicklungsumgebung ist bereit, ein neues Cordova-Project anzulegen.

Anleitung für Eilige

Voraussetzung:

- NodeJS und NPM (zur Überprüfung: `node -v` bzw. `npm -v`)
- Android SDK (benötigt das Java SDK)
- Git

Installation:

- `npm install -g cordova` (zur Überprüfung: `cordova -v`)

3.4 Umgang mit der CLI

3.4.1 Das erste eigene Projekt

Cordova versucht, den Umgang mit dem CLI möglichst simpel zu gestalten. Dabei soll der Entwickler so wenig wie möglich über die zu unterstützenden Systeme wissen müssen. Deshalb bietet Cordova für alle grundlegenden Konfigurationen und möglichen Anpassungen einfache und aussagekräftige Befehle für die Kommandozeile an. Das beginnt bereits bei der initialen Erstellung eines Projekts und zieht sich bis zum finalen Bauen der Anwendungsdateien hindurch.

Zuerst sollte ein geeigneter Platz auf der Festplatte gefunden werden, an dem das Projekt generiert und gespeichert werden kann. Ist dieser gefunden, generiert der nachfolgende Befehl ein neues Cordova-Projekt:

```
cordova create cordovaProjekt com.beispiel.hallo MeineErsteApp
```

Der Abschnitt `cordova create` ruft die Projekterzeugungs-Funktion des Cordova CLI auf. Danach folgt der Ordnername, in dem die erzeugten Dateien und Verzeichnisse abgelegt werden sollen. Dieser muss nicht existieren, sondern wird im Prozess angelegt.

Als Nächstes folgt die Projekt-ID mit `com.beispiel.hallo`. Diese hat eine grundlegende Bedeutung für die spätere Veröffentlichung der App. Jede Anwendung hat eine eindeutige ID, die in den App-Stores nur einmal vorkommen darf. Wird die ID einer veröffentlichten App später geändert, werten die App-Stores diese als neue Applikation. Ein neuer App-Store-Eintrag wird angelegt und im schlimmsten Fall sind dann beide Versionen der App für die Endnutzer verfügbar. Daher sollte spätestens vor der Veröffentlichung überprüft werden, ob eine sinnvolle und möglichst spezifische ID hinterlegt wurde.

Der letzte Abschnitt des Befehls `MeineErsteApp` stellt den späteren Anzeigenamen dar. Dieser erscheint nach dem Installieren auf einem mobilen Endgerät in der Liste der installierten Apps.

3.4.2 Die Projektstruktur

Hat das Erstellen geklappt, sollte nun ein Ordner `cordovaProjekt` existieren und folgende Verzeichnisse und Dateien beinhalten (Bild 3.1).

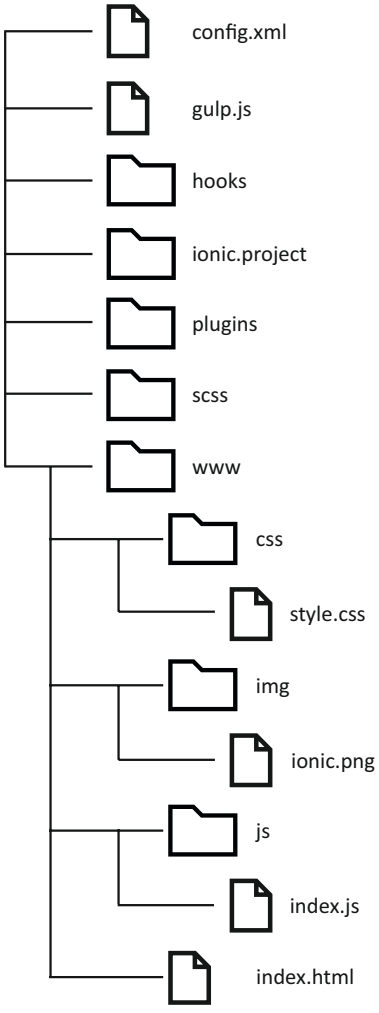

Bild 3.1
Ordnerstruktur des Beispielprojektes

Die wichtigsten Dateien sind die `config.xml` und die Inhalte des www-Ordners. Wie der Name schon vermuten lässt, beinhaltet die `config.xml` (Listing 3.1) alle wichtigen Einstellungen für das Projekt. Sie ist notwendig, um das Projekt auszuführen und die Anwen-

dungsdateien bauen zu können. Der Inhalt lässt sich aber bedenkenlos ansehen und später, wenn notwendig, auch ändern.

Listing 3.1 Beispiel einer config.xml

```
<?xml version='1.0' encoding='utf-8'?>
<widgetid="com.beispiel.hallo" version="0.0.1" xmlns=http://www.w3.org/ns/widgets
 xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>HelloWorld</name>
  <description>
    A sample Apache Cordova application that responds to the deviceready event.
  </description>
  <author email="dev@cordova.apache.org" href="http://cordova.io">
    Apache Cordova Team
  </author>
  <content src="index.html" />
  <plugin name="cordova-plugin-whitelist" version="1" />
  <access origin="*" />
  <allow-intent href="http://*/*" />
  <allow-intent href="https://*/*" />
  <allow-intent href="tel:*" />
  <allow-intent href="sms:*" />
  <allow-intent href="mailto:*" />
  <allow-intent href="geo:*" />
  <platform name="android">
    <allow-intent href="market:*" />
  </platform>
  <platform name="ios">
    <allow-intent href="itms:*" />
    <allow-intent href="itms-apps:*" />
  </platform>
</widget>
```

Hier findet sich auch die gewählte App-ID im `widget`-Abschnitt wieder. Sie kann wie der App-Name im `name`-Block geändert werden. Zusätzlich können eine Beschreibung und Informationen über den Autor hinterlegt werden.

Die Dateien des www-Verzeichnisses stellen die eigentlichen Inhalte der App dar. Hier liegen alle HTML- und JavaScript-Dateien, mögliche zusätzliche Bibliotheken, Styling-Dateien (z. B. CSS) und statische Dateien, wie App-Logos, Splashscreens und andere Grafiken, die innerhalb der Anwendung benötigt werden. Die Struktur der Unterordner ist nur ein Vorschlag und kann nach Belieben variiert werden. Wichtig ist nur, dass der Startpunkt der App – in diesem Fall die `index.html` – erhalten bleibt. Aber selbst diese Basisdatei kann in der `config.xml` im Abschnitt `content` verändert werden. Die aktuelle `index.html` sollte ungefähr wie folgt aufgebaut sein (Listing 3.2):

Listing 3.2 Beispiel einer index.html

```
<html>
  <head>
    <meta http-equiv="Content-Security-Policy" content="default-src 'self' data:
gap: https://ssl.gstatic.com 'unsafe-eval'; style-src 'self' 'unsafe-inline';
 media-src *">
    <meta name="format-detection" content="telephone=no">
    <meta name="msapplication-tap-highlight" content="no">
    <meta name="viewport" content="user-scalable=no, initial-scale=1,
```

```
maximum-scale=1, minimum-scale=1, width=device-width">
    <link rel="stylesheet" type="text/css" href="css/index.css">
    <title>Hello World</title>
  </head>
  <body>
    <div class="app">
      <h1>Apache Cordova</h1>
      <div id="deviceready" class="blink">
        <p class="event listening">Connecting to Device</p>
        <p class="event received">Device is Ready</p>
      </div>
    </div>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="js/index.js"></script>
  </body>
</html>
```

In dieser HTML-Datei werden alle eigenen Quellen eingebunden – im Kopfbereich die .css-Datei und im `body` die `index.js` sowie, bei allen Cordova-Projekten notwendig, die `cordova.js`. Durch diese Datei erhält die gebaute Anwendung erst die Schnittstellen zum Gerät.

Im Verzeichnis `platforms` werden später die verschiedenen Quellen für die zu unterstützende Plattformen hinterlegt. In dem aktuellen Zustand des Projekts ist noch keine Plattform angegeben.

Die Ordner `hooks` und `plugins` sind in diesem Abschnitt eher unwichtig, sollen jedoch nicht unerwähnt bleiben. Das englische Wort Hook bedeutet auf Deutsch so viel wie Haken bzw. Aufhänger. In der Programmierung ist vielleicht der Begriff Einhängepunkte treffender. Cordova bietet nämlich während des Bauprozesses verschiedene Punkte an, in denen sich der Entwickler mit eigenen NodeJS-Skripten einhaken kann. Plugins sind dagegen wie eingangs erwähnt Erweiterungen, um zusätzliche native Funktionen des mobilen Endgeräts nutzen zu können.

Somit ist der grundlegende Aufbau eines Cordova-Projekts erläutert. Im Anschluss kann nun das Beispielprojekt im Android-Kontext ausgeführt werden.

3.4.3 Eine Plattform hinzufügen

Für die Generierung einer hybriden Anwendung auf Basis von Cordova ist es notwendig, die Plattformen und Betriebssystemarten, auf denen die App erscheinen soll, anzugeben. Je nach Plattform müssen zusätzliche Quellen abgerufen und eingerichtet werden. Darüber hinaus wird nur das installiert, was auch benötigt wird. Nutzt man beispielsweise später ein externes Plugin, richtet Cordova dieses nur für die angegebenen Plattformen ein. Um das Beispielprojekt für Android zu registrieren, genügt folgender Befehl:

```
cordova platform add android
```

Daraufhin werden die nötigen Inhalte über NPM heruntergeladen und installiert. Als Ergebnis sollte nun im Verzeichnis `platforms` ein Unterordner `android` erscheinen. Nach diesem Prinzip können weitere Plattformen hinzugefügt werden. Zur Erinnerung: iOS ist davon jedoch ausgeschlossen, da iOS-Apps nur auf Mac OS erzeugt werden können.

Durch die folgende Konsoleneingabe wird eine Plattform wieder entfernt:

```
cordova platform remove android
```

3.4.4 Die Anwendung ausführen

Im einfachsten Fall ist die App wie eine Webanwendung aufgebaut und kann über die `index.html` in einem installierten Browser aufgerufen werden. Dabei kann jedoch nicht auf die nativen Schnittstellen zurückgegriffen werden.

Bild 3.2
Ausführung der Anwendung im Browser

In der Mitte des geöffneten Browserfensters sollte nun obiges Bild (Bild 3.2) erscheinen. Dabei sollte der Text unter Apache Cordova beachtet werden. *Connecting to device* heißt so viel wie *Verbinde mit Gerät*. Da auf keine native Schnittstelle bzw. API zugegriffen werden kann, erhält die Anwendung auch keinerlei Informationen über den aktuellen Gerätestatus. Aus diesem Grund gibt es weitere Möglichkeiten, während der Entwicklung die Anwendung zu testen. Die erste ist sehr schnell und flexibel. Die Anwendung wird auf dem Computer gestartet und auch für mobile Geräte im gleichen Netzwerk zur Verfügung gestellt. Das Ganze geschieht über ein Cordova CLI-Kommando:

```
cordova serve android
```

Nach dem Ausführen des Befehls kann im Browser folgende Adresse aufgerufen werden:

```
http://localhost:8000/
```

Nun erscheint eine Übersicht an Informationen über die App und eine Liste von Plattformen. Android ist die einzige anklickbare Option. Wird der Link geöffnet, erscheint die schon bekannte App-Oberfläche. Sollten nun Hinweisfenster erscheinen, werden diese über *Abbrechen* geschlossen. Mit dem Unterschied zur vorher erklärten Variante wird nun *Device is ready* (das Gerät ist bereit) angezeigt (Bild 3.3).

Bild 3.3
Ausführung der Anwendung im nativen Kontext

Befindet sich ein mobiles Gerät im gleichen Netzwerk wie der Rechner – im einfachsten Fall sind sie im gleichen WLAN –, kann über die interne IP-Adresse des Rechners auch vom Smartphone oder Tablet auf die Anwendung zugegriffen werden. Als Beispiel kann im Browser des Android-Telefons die IP-Adresse mit dem Port 8000 eingegeben werden:

```
http://192.168.178.33:8000
```

Die letzte Methode, die App vor dem eigentlichen Bauen auszuführen und zu testen, benutzt sogenannte Emulatoren (Bild 3.4). Diese sind meist bei den bereits installierten plattformabhängigen Tools dabei. Dadurch ist es möglich, Funktionen einer bestimmten Betriebssystemversion oder eines bestimmten Gerätetyps nachzuahmen und die App in diesem Kontext auf einem Computer auszuführen. Die Einrichtung kann jedoch ein wenig kompliziert sein. An dieser Stelle wird nicht näher darauf eingegangen, da diese Informationen für das weitere Verständnis des Buches nicht notwendig sind. Letztendlich wird auch diese Cordova-Funktion über einen kleinen Befehl gestartet:

```
cordova emulate android
```

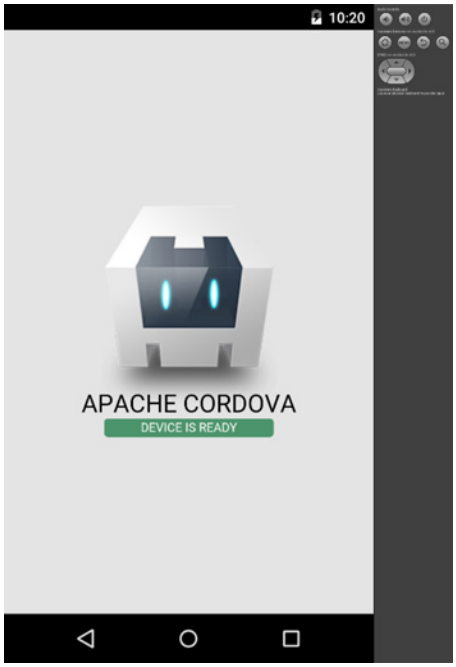

Bild 3.4
Ausführung der Anwendung im Android Emulator

3.4.5 Plugins verwalten

Im Normalfall lebt eine App von der Interaktivität mit dem mobilen Gerät – sei es die Benutzung der Kamera oder die Verknüpfung mit sozialen Netzwerken wie Facebook. Dazu reichen die von Cordova standardmäßig angebotenen Schnittstellen nicht aus. Jedoch gibt es

aufgrund des freien Quellcodes und eines eigenen Plugin-Systems eine Vielzahl von Erweiterungen. Ein Cordova-Plugin wird in der Regel in der für die Plattformen vorgeschriebenen Programmiersprache geschrieben. Dazu kommt eine JavaScript-Datei, welche die Brücke zwischen nativem Code und späterer Webanwendung darstellt. Diese muss nach einem gewissen Schema aufgebaut sein, damit Cordova das Plugin erkennt und verwenden kann. Ein Plugin kann mit einfachen Mitteln öffentlich anderen Nutzern zugänglich gemacht werden. Auf die Erstellung und Programmierung eines eigenen Plugins wird an dieser Stelle nicht genauer eingegangen.

PhoneGap als auch Cordova besitzen eigene Webseiten, auf denen nach Plugins gesucht werden kann. Ganz einfach funktioniert das Suchen aber auch über die CLIs:

```
cordova plugin search bar
```

Dieser Befehl spricht die Plugin-Funktion des Cordova-CLI an und startet eine Suche nach Schlagwörtern. In diesem Beispiel werden Erweiterungen nach dem Wort *bar* durchsucht. Als Ergebnis werden eine Liste von Plugin-Namen und eine Beschreibung ausgegeben. Über den angegebenen Namen kann ein Plugin installiert werden, z. B.:

```
cordova plugin add cordova-plugin-statusbar
```

Danach liegt das Plugin im Projektverzeichnis unter `plugins`. Eine Erweiterung lässt sich äquivalent zum Befehl von Plattformen wieder entfernen:

```
cordova plugin remove cordova-plugin-statusbar
```

Sollte im Entwicklungsprozess eine Plattform hinzukommen, müssen bereits installierte Plugins entfernt und nochmals installiert werden, da die Quellen für diese Plattformen sonst fehlen.

Ist eine Erweiterung noch nicht bei Cordova registriert, kann diese über ihr Git-Repository installiert werden. Dazu wird einfach im Installationsbefehl der Plugin-Name mit der URL zum entsprechenden Repository ausgetauscht:

```
cordova plugin add https://github.com/apache/cordova-plugin-file.git
```

Zur Verwendung des individuellen Plugins gibt es auf der Plugin-Seite oder – falls vorhanden – in der Anleitungsdatei im Plugin-Ordner des Projekts weitere Informationen.

3.4.6 Die erste App bauen

Nachdem alle Grundlagen zum Entwickeln mit dem Cordova CLI vermittelt wurden, fehlt nur noch der letzte und wichtigste Schritt auf dem Weg zur eigenen App – das Bauen. Dabei werden für jede Plattform spezifische Anwendungspakete gebaut. Unter Android entsteht im Bauprozess die .apk-Datei.

```
cordova build android
```

Die Ausgaben des Befehls geben Aufschluss über die einzelnen Schritte des Prozesses und im Erfolgsfall zusätzlich den Pfad zur erstellten App. In Fall des Beispielprojekts sollte dieser dem folgenden ab dem Projektverzeichnis entsprechen:

```
cordovaProjekt\platforms\android\build\outputs\apk\android-debug.apk
```

Nun kann die Datei testweise auf ein Android-Gerät übertragen, installiert und ausgeführt werden.

3.4.7 Zusammenfassung

Anlegen eines Projektes

Angabe von Projektverzeichnis, App-ID und -Namen:

```
cordova create cordovaProjekt com.beispiel.hallo MeineErsteApp
```

Aufbau des Projektes

- **config.xml**: Konfigurationsdatei
- **hooks**: eigene Skripte für den Bauprozess (z. B. für die Minimierung und/oder Paketierung einzelner App-Komponenten)
- **platforms**: Quellen für die unterstützten Plattformen, um die Anwendung zu generieren
- **plugins**: zusätzlich eingebundene Erweiterungen der nativen Schnittstelle
- **www**: eigentliches Verzeichnis der Anwendung
  - **index.html**: Einstiegspunkt der App, der alle benötigten Quellen (u. a. cordova.js) einbindet
  - **css**: für das Styling relevante Dateien
  - **img**: Ordner für benötigte Bilder
  - **js**: JavaScript-Quellen

Hinzufügen und Entfernen von Plattformen

Angabe der Plattformnamen (z. B. Android, iOS):

```
cordova platform add android [ios, ...]
cordova platform remove android
```

Ausführen und Testen der Anwendung

- Ausführen der Anwendung im Browser durch Öffnen der `index.html`
- Ausführen und Testen mit nativen Funktionen: `cordova serve android`

Index

Symbole

V

W

X