Chapter 4

Using PhoneGap Build

In this chapter, we cover:

- Beginning with PhoneGap Build
- Defining the structure of the application to upload
- Creating builds for different mobile platforms
- Loading private keys
- Installing the builds on the emulator
- Downloading apps directly to the device
- Debugging the apps

Beginning with PhoneGap Build

In Chapter 3, we saw that developing PhoneGap applications for multiple mobile platforms is a critical task. We need to install and configure the Software Development Kit (SDK), tools, and simulators for each platform. Because each mobile platform is based on different environments, the technique of accessing the hardware and using it in the applications is also quite different. PhoneGap Build relieves us from all this overhead.

PhoneGap Build is a cloud build service that makes the task of building PhoneGap applications in the cloud quite easy. What we are supposed to do is just create the Web content and upload it to the Build service. Thereafter, it is the task of PhoneGap Build to develop the code that supports different mobile platforms, including Android, Windows Phone, BlackBerry, iOS, Symbian, and webOS. It also means that we, as developers, just need to develop a single application (in any platform of our choice), and when that application is loaded to the Build service, PhoneGap Build will generate applications for the other platforms for us.

For using PhoneGap Build, we first need to create an account on the PhoneGap Build Web site. So, visit the URL `http://build.phonegap.com` and sign up. When we sign in to the PhoneGap Build site, we are provided with two options to upload our application: we can upload to the Build service either an open-source public app or a private app. A private app, as the name suggests, is the app that is kept on our local disk drive and is meant to be accessed and used solely

PhoneGap Build: Developing Cross Platform Mobile Applications in the Cloud
B.M. Harwani
International Standard Book Number-13: 978-1-4665-8974-2 (Hardback)
© 2014 by Taylor & Francis Group, LLC

by us and no one else. A public app means the application is created through the public Github account and is available for public use. An application can be as simple as a single `index.html` file or a combination of several files. If the application is a combination of several files, it can be compressed into a zip file. The `index.html` file or the compressed zip file can then be uploaded to the PhoneGap Build service.

We prefer to upload a public app that is available on a Github account into the PhoneGap Build service. That is, we will create a Git repository, push our app files into it, and then pull it into the Build service. Refer to Appendix B to create a public repository on Github. We can copy the Git repository into the box shown in Figure 4.1 to upload it to the PhoneGap Build service. But before that, let us connect PhoneGap Build with our Github account. Why?

The benefit of connecting our Github account with PhoneGap Build is that any updating done in the Github repository can be directly applied to the app that is uploaded in the PhoneGap Build service. So, to connect the PhoneGap Build account with the Github account, let us click the `Connect your Github account` link that appears below the box.

Upon clicking the `Connect your Github account` link, the `Edit account` page opens as shown in Figure 4.2. We find a link `Connect a GitHub ID` inside the `Linked`

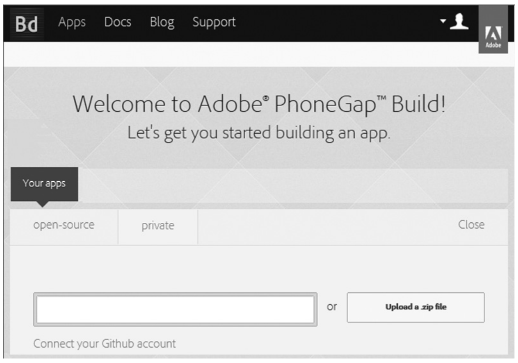
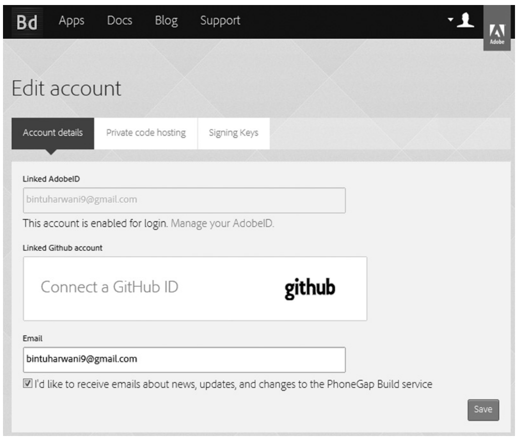

Bd
Apps
Docs
Blog
Support
Adobe
Welcome to Adobe® PhoneGap™ Build!
Let's get you started building an app.
Your apps
open-source
private
Close
or
Upload a .zip file
Connect your Github account

Figure 4.1 The first page that appears upon loading the PhoneGap Build site.



Edit account

Linked AdobeID
bintuharwani9@gmail.com
This account is enabled for login. Manage your AdobeID.

Linked Github account

Email
bintuharwani9@gmail.com

☑ I'd like to receive emails about news, updates, and changes to the PhoneGap Build service

Figure 4.2 The Edit account page showing the link to connect with the Github account.

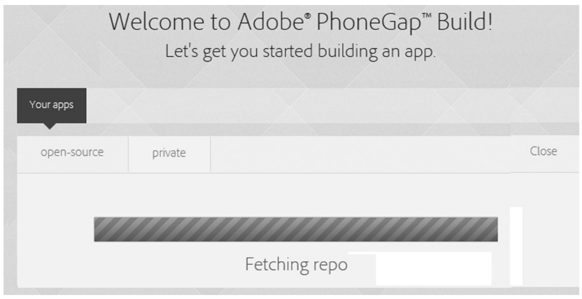

Welcome to Adobe® PhoneGap™ Build!
Let's get you started building an app.
Your apps
open-source
private
Close
Fetching repo

Figure 4.3 The Github repository is being fetched into the PhoneGap Build service.

`Github account` box. It is this link that is used for connecting a PhoneGap Build account with a Github account.

Upon clicking the link `Connect a GitHub ID`, a dialog will open asking whether we want to authorize `PhoneGap:Build` to read our public information, update our user profile, and update our private and public repositories. Click the `Authorize app` button to enable our PhoneGap Build account to access and modify our Github repositories.

When a connection with the Github account is established, our Github ID will appear in the `Linked Github account` box of the `Edit account` page, which confirms that our PhoneGap Build and Github accounts are successfully linked.

Now, we can enter our Github repository URI in the box (see Figure 4.1). On entering our Git repository Uniform Resource Identifier (URI), `https://github.com/bmharwani/PhoneGapApps`, in the box, the PhoneGap Build service will begin fetching the repository from the Github account, as shown in Figure 4.3.

Defining the Structure of the Application to Upload

A PhoneGap application may be either a simple `index.html` file or a combination of several files, including JavaScript, Cascading Style Sheets (CSS), and media files, if required by the application. For example, the application that we have uploaded to the PhoneGap Build service above comprises the following three files:

- `index.html`—To display Web content.
- `config.xml`—To specify application configuration settings.
- `icon.png`—Image file to represent an application's icon.

In addition to the files above, we can always include JavaScript, CSS, video, and audio files in an application.

The application that we have uploaded on PhoneGap Build is the simple welcome application that we created in earlier chapters. The application prompts for a username, and when the user clicks the `Submit` button after entering a name, a welcome message is displayed on the screen. The `index.html` file is shown in Listing 4.1 for reference.

Listing 4.1 Code Written in the `index.html` File

```
<!DOCTYPE HTML>
<html>
   <head>
   <title>Welcome Message App</title>
   <script type = "text/javascript" charset = "utf-8" src = "cordova-2.2.0.js">
   </script>
   <script type = "text/javascript">
   function onBodyLoad() {
      document.addEventListener("deviceready", PhonegapLoaded, false);
   }
   function PhonegapLoaded(){
   document.getElementById("submitButton").addEventListener("click", dispMessage);
   }
   function dispMessage(){
      var nameOfUser = document.getElementById("name").value;
      document.getElementById("welcomemsg").innerHTML = "Welcome "+nameOfUser + " !";
   }
   </script>
   </head>
   <body onload = "onBodyLoad()">
      <form>
         Enter your name <input type = "text" id = "name"><br/>
         <input type = "button" id = "submitButton" value = "Submit">
         <div id = "welcomemsg"></div>
      </form>
   </body>
</html>
```

One thing to remember is that we do not have to supply a copy of the PhoneGap/Cordova JavaScript with the application that we upload on the Build service, because it is automatically included by PhoneGap Build. But we do have to include its reference in the documents.

The configuration file, `config.xml`, is used to store details of our application. That is, information like application name, its description, author info, e-mail, application icon, and other settings is supplied in the form of a `config.xml` file. A sample `config.xml` file that we have used in our application appears as shown in Listing 4.2.

Listing 4.2 Code Written in the `config.xml` File

```
<?xml version = "1.0" encoding = "UTF-8"?>
<widget xmlns = "http://www.w3.org/ns/widgets"
xmlns:gap = "http://phonegap.com/ns/1.0"
id = "com.phonegap.pgwelcomemsg"
version = "1.0.0">
<name>Welcome App</name>
<description>
App that displays welcome message to the user.
</description>
<author href = "http://bmharwani.com" email = "bmharwani@yahoo.com">
Bintu Harwani
</author>
<feature name = "http://api.phonegap.com/1.0/device"/>
<preference name = "phonegap-version" value = "2.2.0"/>
<preference name = "orientation" value = "default"/>
<preference name = "target-device" value = "universal"/>
<preference name = "fullscreen" value = "false"/>
<icon src = "icon.png"/>
</widget>
```

The meanings of different elements used in `config.xml` are briefly described in Table 4.1.

If the `config.xml` file is not used in an application, PhoneGap Build will build the application using the default settings.

Table 4.1 A Brief Description of the Elements Used in `config.xml`

Element	*Description*
`<widget>`	The `<widget>` element must be the root of our XML document. It confirms that we are following the World Wide Web Consortium (W3C) specification. The following two attributes are usually used with the `<widget>` element: • **ID**—Used to identify our application uniquely. To support all mobile platforms, the ID is assigned in a reverse-domain name format: `com.company/category.appname`. • **Version**—Defines the version of our application. It comprises three numbers that represent the major, minor, and patch numbers of our application.
`<name>`	Represents the name of the application.
`<description>`	Represents a small description of our application.
`<author>`	Specifies the author name and customer support contacts.
`<feature>`	Used to specify the features we want to use in our application. For example, the feature statement used in the `config.xml` file above indicates that we want to access the device application programming interface (API).
`<preference>`	Comprises the `name` and `value` pair and is used for defining different properties of the application. There can be zero or more `<preference>` elements in a `config.xml` file. If no `<preference>` element is used in the `config.xml` file, PhoneGap Build will apply default properties to our application. A few of the preference elements are explained below: • The `phonegap-version` preference ensures that our application is built with the specific version of PhoneGap. The statement used in the `config.xml` above will build our application with PhoneGap version 2.2.0. • The `orientation` preference is for enabling `landscape` and `portrait` orientations of our device. Possible values of this element are `default`, `landscape`, or `portrait`, where `default` value means that both `landscape` and `portrait` orientations are enabled. • The `target-device` preference is for informing the target device of our application. Possible values are `handset`, `tablet`, or `universal`, where the `universal` value means that our application is compatible with all devices.

(Continued)

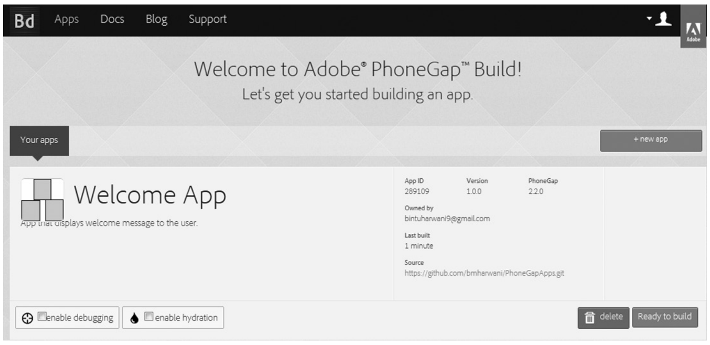
Table 4.1 A Brief Description of the Elements Used in `config.xml` (Continued)

Element	*Description*
	• The `fullscreen` preference determines whether to hide the status bar at the top of the screen. Possible values are `true` and `false`, where the `true` value will hide the status bar. The default value is `false`. • The `<icon>` element is for defining the icon image of the application. If no icon image is specified, the PhoneGap logo is considered our application's icon. The icon file must be named `icon.png` because each platform by default uses this filename for its icon. The icon image file must reside in the root of our application folder. We can also define different icons for each platform and screen type. The following statement defines an icon for the `mdpi` screen of an Android platform: `<icon src = "icons/android/mdpi.png" gap:platform = "android" gap:density = "mdpi"/>` Similarly, we can define icons for `ldpi`, `hdpi`, and `xhdpi` displays too.

Note: You might be wondering why I have referenced PhoneGap version 2.2.0 in the `config.xml` file above when the recent PhoneGap version available is 2.3.0. The reason is that PhoneGap Build at the time of this writing supports up to 2.2.0 version only. The currently supported PhoneGap versions are 1.1.0, 1.2.0, 1.3.0, 1.4.1, 1.5.0, 1.6.1, 1.7.0, 1.8.1, 1.9.0, 2.0.0, 2.1.0, and 2.2.0. If we specify an unsupported version number, the app will not be built by the PhoneGap Build service.

Creating Builds for Different Mobile Platforms

When the application is retrieved in PhoneGap Build, it will appear under the `Your apps` section along with its icon, title, and other information, as shown in Figure 4.4.



Welcome to Adobe® PhoneGap™ Build!
Let's get you started building an app.

Welcome App
App that displays welcome message to the user.

App ID	Version	PhoneGap
289109	1.0.0	2.2.0

Owned by
bintuharwani9@gmail.com

Last built
1 minute

Source
https://github.com/bmharwani/PhoneGapApps.git

Figure 4.4 The application is retrieved and visible in PhoneGap Build.

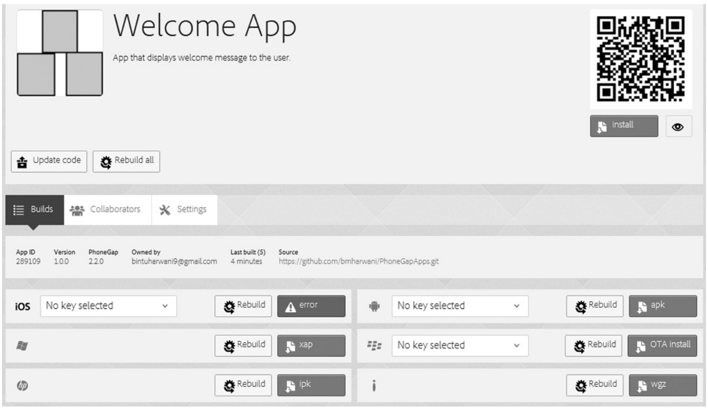
Figure 4.4 shows the three buttons `+new app`, `delete`, and `Ready to build`, along with two checkboxes, `enable debugging` and `enable hydration`. The usage of the buttons and checkboxes is given below:

- `enable hydration`—Enables hydration in our application. Hydration is a tool that not only improves an application's compilation time but also enables us to update the application that is installed on a device. That is, whenever we upload a new build of the application, the user of the application is notified about the same upon restarting the application. If the user wants to run the new code, the hydrated app will automatically fetch and run the new build. At the time of this writing, the hydration feature is available only for iOS and Android platforms and supports PhoneGap versions 1.8.1, 1.9.0, and 2.0.0. The procedure of installing hydrated applications on the device is the same as for nonhydrated applications. To disable hydration, select an application by clicking its title or icon. From the detail page, open the `Settings` panel. You will see a checkbox `enable hydration`; just uncheck it and click the `Save` button.
- `enable debugging`—Check this checkbox to enable debugging in an application. This feature helps in modifying the application in real time, that is, when it is installed on a device. Small modifications and fixing bugs in an application can be interactively done in this debug mode. Debugging is explained at the end of this chapter.
- `+new apps`—Found at the top right of the page, this button is used to upload a new application. It opens a dialog to enter information for the new application. Enter the title of the application, provide its source code (either a single `index.html` file, .zip file, or a Git repository), and after selecting the desired settings, click the `Create` button to upload it to the PhoneGap Build service. The application's title, description, and icon will be fetched from the `config.xml` file (if it is supplied with the application files).
- `Delete`—This button will delete the selected application.
- `Ready to build`—This button, when clicked, initiates PhoneGap Build to begin the building process—creating apps for different mobile platforms. If the key is not configured for the iOS platform, PhoneGap Build will display an error while creating a build for the iOS application.

Note: For iOS applications, it is essential to have a developer account to get the certificate. More specifically, the iOS application needs to be signed by a developer certificate and a provisioning profile. The provisioning profile is linked to an Apple developer account. To test the application on an iOS device, the device has to be registered with this provisioning profile.

Upon selecting the `Ready to build` button, the builds of all the mobile platforms except the iOS platform will be made as shown in Figure 4.5. The text `No key selected` with iOS, Android, and BlackBerry platforms indicates that a private key has to be uploaded for these three platforms to generate their `release` builds. That is, until a private key is uploaded for these three platforms, PhoneGap Build will generate `debug` builds for them. Recall from Chapter 3 that `debug` builds are meant for testing purposes and cannot be installed on the devices. It also means that to develop `release` builds of these three mobile platforms, we need to load their respective private keys on the PhoneGap Build site.

We can press the `Update code` button to update our application from the Git repository and rebuild the application. On making any changes in the application, whether in terms of code or in adding or removing of keys, we can use the `Rebuild` button that appears on the right of


Welcome App

App that displays welcome message to the user.

App ID	Version	PhoneGap	Owned by	Last built (S)	Source
289109	1.0.0	2.2.0	bintuharwani9@gmail.com	4 minutes	https://github.com/bmharwani/PhoneGapApps.git

Figure 4.5 The builds for five successfully created mobile platforms.

each platform to develop their respective build. We can also use the `Rebuild all` button to develop the builds (cross-platform apps) for all the platforms: Android, BlackBerry, iOS, Windows Phone, Symbian, and webOS.

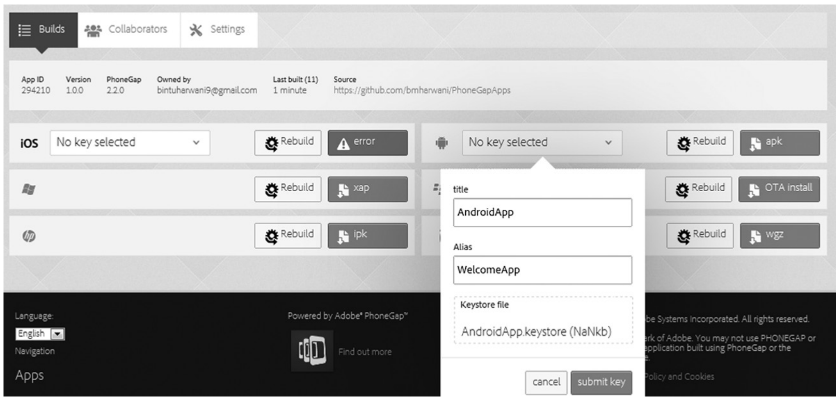
PhoneGap Build provides buttons to download individual builds of mobile platforms. To download the build of any platform, we can click the button on the right of the `Rebuild` button that is associated with each platform. For example, to download the build of Windows Phone 7, we can click the button with the caption `xap` found on the right of the `Rebuild` button in the Windows row. Similarly, clicking the button with the caption `apk` in the Android row will download the build for the Android platform.

Loading Private Keys

One thing is for sure: the `apk` file of the Android platform that we will download from PhoneGap Build cannot be installed on a device. Why?

The reason is quite simple. Because we have not loaded the private key for the Android platform, the build version developed by PhoneGap Build will be a `debug` version and not a `release` version. So, let us load the private key for the Android platform. In Chapter 3, we learned to create a private keystore using `keytool`. We will use the same `AndroidApp.keystore` that we developed in Chapter 3 for building our Android app. So, click the `No key selected` text found in the Android row. A small dialog will pop up prompting us to enter the title, alias, and filename of keystore. Let us assign `AndroidApp` and `WelcomeApp` as the `Title` and `Alias` for the keystore and browse, and select the `AndroidApp.keystore` file on our local disk drive (see Figure 4.6). Finally, click the `submit key` button to upload the specified keystore on the PhoneGap Build service.

Upon clicking the `submit key` button, the key in the `AndroidApp.keystore` will be uploaded for the Android application. The title of the keystore along with a lock symbol will appear in the Android row that indicates that the key is successfully uploaded. Next, we rebuild the Android application by clicking the `Rebuild` button available in the Android row. But, we will not be able to do so, because the key needs to be unlocked while building the application. To unlock the key, click the lock symbol that appears adjacent to the keystore title. A dialog will



App ID 294210 | Version 1.0.0 | PhoneGap 2.2.0 | Owned by bintuharwani9@gmail.com | Last built (11) 1 minute | Source https://github.com/bmharwani/PhoneGapApps

title: AndroidApp
Alias: WelcomeApp
Keystore file: AndroidApp.keystore (NaNkb)

Figure 4.6 The loading key for the Android platform.

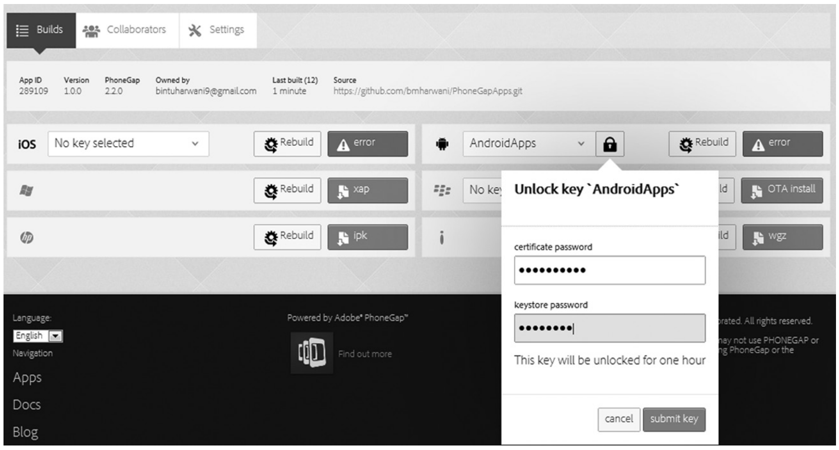
pop up asking for the certificate and keystore password that we used while creating the keystore through `keytool`. Enter the two passwords correctly, followed by clicking the `submit key` button to unlock the key (see Figure 4.7).

For safety purposes, PhoneGap Build will keep the keys unlocked for an hour. After an hour, the keys will be automatically locked again. While the keys are unlocked, we can click the `Rebuild` button (found on the right side of the lock symbol) to build the `release` version of our Android app.

A question that may occur to you is: Do we have to upload keys for every individual app uploaded on PhoneGap Build? Can we not upload the keys once and use the same key for all the apps?

Of course, yes! The steps to upload the Android private keys that can be universally applied to all Android apps are as follows:

- Click your account ID link at the bottom of the page.
- Click the `Signing keys` tab found at the top in the `Edit account` page that opens up.
- Click the `add a key` button that appears below the Android logo.



App ID 289109 | Version 1.0.0 | PhoneGap 2.2.0 | Owned by bintuharwani9@gmail.com | Last built (12) 1 minute | Source https://github.com/bmharwani/PhoneGapApps.git

AndroidApps

Unlock key `AndroidApps`
certificate password: ••••••••••
keystore password: ••••••••
This key will be unlocked for one hour

Figure 4.7 Unlocking the Android key to prepare for its build.

- Enter the `Title` and `Alias` of the private keystore and browse and select the keystore file, `AndroidApp.keystore`, that contains the private key for the Android apps. Click the `submit key` button (see Figure 4.8).

Upon clicking the `submit key` button, the private key for the Android apps will be stored. The private keystore filename, `AndroidApp`, appears below the Android logo to confirm that key is successfully loaded on PhoneGap Build. As stated earlier, the key will be locked by default; the lock symbol (see Figure 4.9) informs us that the key is in a locked state and needs to be unlocked before developing the Android build. The trash icon on the right of the lock symbol is meant for deleting the uploaded keystore.

Unlocking of the key can be done for individual apps, as we saw earlier or in this `Edit account` page to unlock the key for all the Android apps. To unlock the key for all Android apps, let us click the lock symbol. A dialog will appear asking for certificate and keystore passwords, similar to the one we saw in Figure 4.7. Upon entering the correct certificate and keystore passwords, the key will be unlocked.

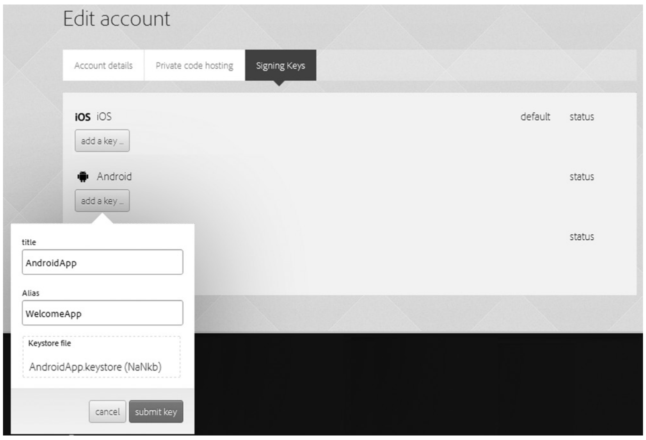

title: AndroidApp
Alias: WelcomeApp
Keystore file: AndroidApp.keystore (NaNkb)

Figure 4.8 Assigning a private key for the Android apps.

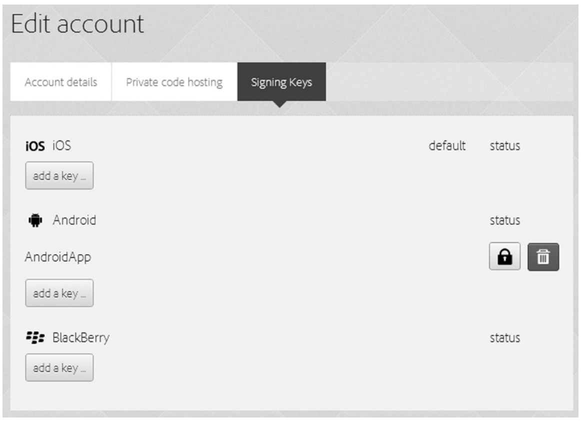
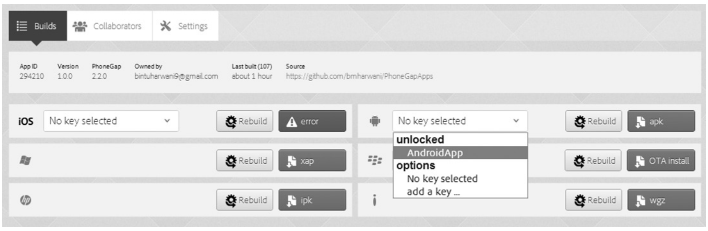


Figure 4.9 Keystore is uploaded and the keys are in a locked state.

PhoneGap Build: Developing Cross Platform Mobile Applications in the Cloud
B.M. Harwani
International Standard Book Number-13: 978-1-4665-8974-2 (Hardback)
© 2014 by Taylor & Francis Group, LLC



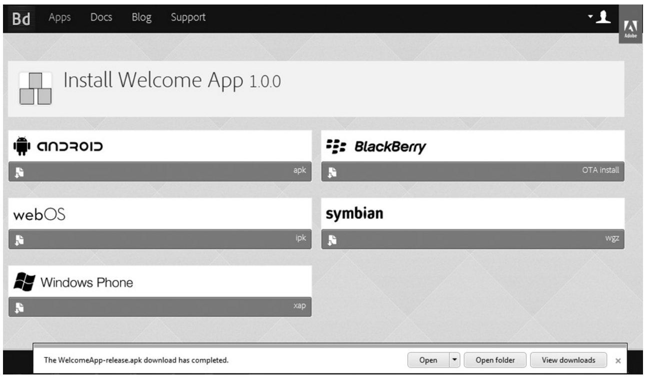
App ID	Version	PhoneGap	Owned by	Last built (107)	Source
294210	1.0.0	2.2.0	bintuharwani9@gmail.com	about 1 hour	https://github.com/bmharwani/PhoneGapApps

Figure 4.10 Applying the earlier stored keys to the Android app.

The key uploaded in the `Edit account` page, whether in a locked or unlocked state, can be applied to all the Android apps that are available in the current PhoneGap Build account. For example, to apply the key to our current `Welcome App`, click the application and click the `No key selected` text in the Android row. A drop-down list will appear showing the uploaded key, `AndroidApp`. Above the `AndroidApp` key, the text `unlocked` appears to declare that the key is currently in an unlocked state (see Figure 4.10). Whether in a locked or unlocked state, the `AndroidApp` key, when selected, will be applied to our `Welcome App` application. Unlock the key if it is in a locked state, and click the `Rebuild` button to create the `release` build of the Android app.

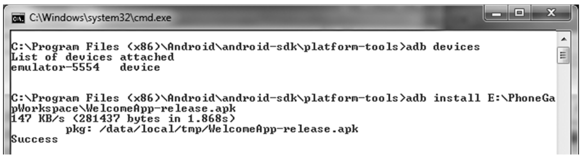
Installing the Builds on the Emulator

When we feel that all updates are applied to the application and are ready to be released and distributed, click the `Install` button found on the right side of the application's title. Upon clicking the `Install` button, the builds of all the platforms will be rebuild and appear as shown in Figure 4.11. The icon below each platform title, the one with a document icon and a down arrow, represents a download link. We can download the build of any platform by clicking the download link of the respective platform. Upon clicking the download link of the Android platform, its build file, `WelcomeApp-release.apk`, will be downloaded onto our computer. The term `release` with the downloaded filename confirms that the build is a `release` version and can be installed on the devices and is ready to distribute.



The WelcomeApp-release.apk download has completed.

Figure 4.11 Saving the Android build on a local machine.



```
C:\Program Files (x86)\Android\android-sdk\platform-tools>adb devices
List of devices attached
emulator-5554   device

C:\Program Files (x86)\Android\android-sdk\platform-tools>adb install E:\PhoneGa
pWorkspace\WelcomeApp-release.apk
147 KB/s (281437 bytes in 1.868s)
        pkg: /data/local/tmp/WelcomeApp-release.apk
Success
```

Figure 4.12 Installing the Android build, which was retrieved from the PhoneGap Build service on the Android emulator.

Let us check the downloaded Android build on an Android emulator. To start an Android emulator, launch Eclipse integrated development environment (IDE), open the Android Virtual Device (AVD) Manager by clicking the `Window->Android Virtual Device Manager` option, select the `PhoneGap AVD` that we created in Chapter 1, and click the `Start` button. We will make use of the `ADB` to install the downloaded Android build, `WelcomeApp-release.apk`, upon the running AVD instance. So, open the command prompt and navigate to the folder where Android Debug Bridge (ADB) exists, that is, the `C:\Program Files (x86)\Android\android-sdk\platform-tools` folder, and run the `adb devices` command to confirm if the AVD is running (see Figure 4.12). Assuming the downloaded Android build `WelcomeApp-release.apk` is available in the `E:\PhoneGapWorkspace` folder, execute the following statement to install the Android build on the running AVD instance:

```
C:\Program Files (x86)\Android\android-sdk\platform-tools> adb install
E:\PhoneGapWorkspace\WelcomeApp-release.apk
```

The message `Success` that appears upon executing the `adb install` command confirms that the Android build is successfully installed on the currently running AVD instance. `WelcomeApp` appears among the Apps list in the AVD; we can see its icon in Figure 4.13a. Upon clicking the icon, the application runs successfully. It asks for a name, and when the user clicks the `Submit` button after entering a name, a welcome message appears on the screen, as shown in Figure 4.13b.

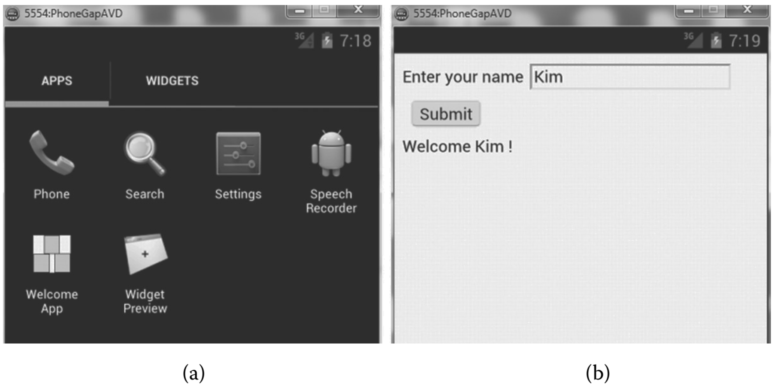


Figure 4.13 (a) The installed application appears among the Apps list. (b) The output on the screen upon running the application in the emulator.

PhoneGap Build

Developing Cross Platform Mobile Applications in the Cloud

B.M. Harwani

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20130919

International Standard Book Number-13: 978-1-4665-8974-2 (Hardback)

Library of Congress Cataloging-in-Publication Data

Harwani, B. M.
PhoneGap build : developing cross platform mobile applications in the cloud / Bintu Harwani.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4665-8974-2 (hardback)
1. Mobile computing. 2. PhoneGap (Application development environment) 3. Application software--Development. 4. Cloud computing. 5. Cross-platform software development. I. Title.

QA76.59.H367 2013
004.67'82--dc23 2013037623

Contents

PhoneGap Build: Developing Cross Platform Mobile Applications in the Cloud
B.M. Harwani
International Standard Book Number-13: 978-1-4665-8974-2 (Hardback)
© 2014 by Taylor & Francis Group, LLC

Preface

PhoneGap is a standards-based, open-source development framework for building fast, easy, cross-platform mobile apps with HTML5, CSS3, and JavaScript for iPhone/iPad, Android, Windows Phone 8, Palm, Symbian, BlackBerry, and more. Once written, the PhoneGap application can be deployed to any mobile device without losing the features of a native app. The PhoneGap applications are able to interact with mobile device hardware, such as the accelerometer or global positioning system (GPS). In other words, PhoneGap provides access to a device's accelerometer, compass, and geolocation capabilities. In addition, you can access device contacts, the local file system, camera, and media on multiple platforms without writing a single line of code.

This book is for intermediate to advanced users. It is comprehensive and covers each topic in detail. The book teaches techniques to configure environments for different mobile platforms. It describes the use of HTML5, CSS3, and JavaScript to develop apps that run on devices of different mobile operating systems. It exploits the features provided by PhoneGap and PhoneGap Build to develop cross-platform mobile applications for the cloud.

The book begins with explaining the differences among the existing mobile platforms, the different types of browsers they support, and the programming languages and integrated development environment (IDE) required to develop apps for each of them. It then describes how PhoneGap makes the task of developing cross-platform mobile apps easier. Initially beginning with small applications, the text gradually moves toward discussing advanced concepts, exploiting different application programming interfaces (APIs) and methods in creating real-life practical applications. By the time you finish the book, you will have learned how to develop feature-rich mobile applications that can run on the cloud to support different platforms.

This book is designed to teach you how to

- Use and configure development environments for mobile platforms, including iOS, Android, BlackBerry, Windows phone, webOS, and Symbian
- Exploit PhoneGap features to develop cross-platform mobile applications
- Use PhoneGap Build to develop mobile apps in the cloud
- Exploit geolocation, compass, accelerometer, contacts, camera, media, and capture APIs and use them in real-life practical mobile applications
- Use the `Database` object to save, fetch, and maintain information for future use
- Use PhoneGap with Sencha Touch
- Use PhoneGap with jQuery Mobile

PhoneGap Build: Developing Cross Platform Mobile Applications in the Cloud
B.M. Harwani
International Standard Book Number-13: 978-1-4665-8974-2 (Hardback)
© 2014 by Taylor & Francis Group, LLC

This book will be beneficial for developers and instructors who want to learn or teach mobile programming. For practical implementation, the book also explains using back-end databases for storing and fetching information. In short, it is a very useful reference book.

About the Author

B.M. Harwani is the founder and owner of Microchip Computer Education (MCE), based in Ajmer, India, which provides computer education in all programming and Web developing platforms. He graduated with a B.E. in computer engineering from the University of Pune, and also has a C Level (master's diploma in computer technology) from DOEACC, Government of India. Having been involved in the teaching field for more than 19 years, Harwani has developed an art of explaining even the most complicated topics in a straightforward and easily understandable fashion. He has written several books that include *Foundation Joomla!* (friendsofED, 2009); *jQuery Recipes: A Problem-Solution Approach* (Apress, 2010); *Beginning Web Development for Smartphones: Developing Web Applications with PHP, MSQL, and jQTouch* (CreateSpace, 2010); *Core Data iOS Essentials* (Packt Publishing, 2011); *Introduction to Python Programming and Developing GUI Applications with PyQT* (Cengage Learning PTR, 2011); *Android Programming Unleashed* (Sams Publishing, 2012); and *The Android Tablet Developer's Cookbook* (Developer's Library) (Addison-Wesley Professional, 2013). To learn more, visit his blog at: http://bmharwani.com/blog.