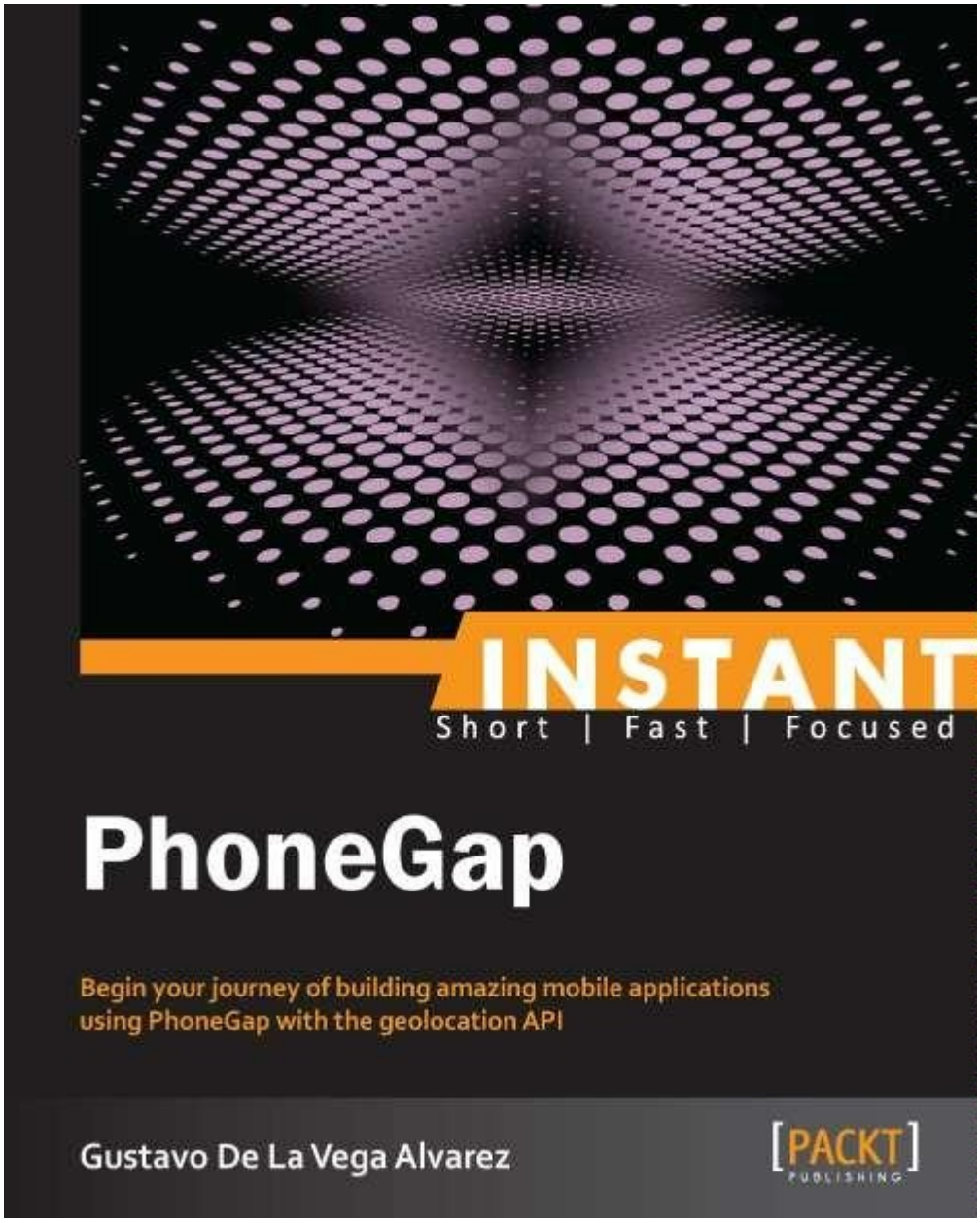

INSTANT
Short | Fast | Focused
PhoneGap
Begin your journey of building amazing mobile applications using PhoneGap with the geolocation API
Gustavo De La Vega Alvarez
[PACKT]
PUBLISHING

Table of Contents

Instant PhoneGap

Instant PhoneGap

First published: December 2013

Production Reference: 1271213

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham B3 2PB, UK.

ISBN 978-1-78216-869-0

www.packtpub.com

Credits

Author

Gustavo De La Vega Alvarez

Reviewers

Evan Mullins

Zainul Setyo Pamungkas

Acquisition Editors

Usha Iyer

Anthony Alburquerque

Harsha Bharwani

Commissioning Editor

Amit Ghodake

Technical Editor

Monica John

Copy Editor

Alfida Paiva

Project Coordinator

Joel Goveya

Proofreader

Stephen Copestake

Graphics

Yuvraj Mannari

Production Coordinator

Saiprasad Kadam

Cover Work

Saiprasad Kadam

Cover Image

Yuvraj Mannari

About the Author

Gustavo De La Vega Alvarez's IT history started when he was an eight-year-old kid. His father gave him a personal computer that worked with cassettes and 5 ¼ " diskettes. He played Buck Rogers in the 21st century. Now, we are in the 21st.

He started programming with an Apple II in the third grade and continued with the Logo programming language. Now, he has his own company, called NativApps that develops mobile apps for third parties using cross-platform languages such as HTML5, CSS, and JavaScript, tools such as PhoneGap, and innovative technologies such as augmented reality.

His education was divided between Business Administration and Computer Science at Universidad de los Andes in Bogotá, Colombia. He started to teach Down Syndrome kids in 1999 on topics such as how to use the PC, word processors, and MSN messenger. His first job as an IT engineer was developing hotel-booking software for a company where his aunt had a job, and another billing software for a law firm. In the early 2000s, he was working for General Motors Colmotores and left the job in 2003 to establish his first company, ProActive Consulting. This company started to work with mobile software and ERPs such as PeopleSoft and SAP. In 2009, he decided to start exploring HTML5 and related tools to develop web apps. Finally, in the same year, he was introduced to PhoneGap since that moment, he has been a PhoneGap believer and a passionate cross-platform developer.

First, I would like to thank God for everything. Thanks go to all the people who helped me in producing this book. Guys, without you, this book would not have ever been done. Thanks to Manuel Cantillo Pareja for his help with the code. Thanks to my whole family, especially to my wife Olga Lucia and my daughter Laura Maria, for supporting me to make my dreams come true. And yes, thanks to my aunties too. Finally, a huge thanks to Juana Gutierrez and Carolina Escobar, who taught me the way to enjoy life day by day. Girls, you rock!

About the Reviewers

Evan Mullins was always interested in both design and technology. He studied Digital Media and earned his BFA at the University of Georgia. While attending college, he also studied Computer Science, Animation, and New Media. He has been doing web design since 2003, picking up HTML, CSS, JavaScript, and even PHP. He has been building interactive sites since 2004 and is mostly a self-taught programmer. He loves the cross-section of art and technology in which we find web design.

Helping out at Indie Music PR firm, Team Clermont, and then Cartoon Network New Media during school, he knew for sure that the Internet was the place for him. During post graduation, he jumped on board helping a slew of start-ups and small businesses with websites until he found himself as a creative developer at Ogilvy. Currently, he is the Interactive Director at the Atlanta Branding and Marketing Agency Brand Fever. Along the way, he learned about client relations, best practices and art direction, and was up to date with the current technologies. He now keeps himself busy by designing and developing interesting things online.

Since 2004, Evan has also maintained circlecube studio as a freelance web studio, and even a playground for open source experiments, examples, and to simply share tips he learns along the way. The blog's content centers around interactive design principles and technologies. He gives back by sharing what he learns online as well as at conferences.

Evan is happily married and a proud father of 4. He enjoys spending time away from work with his family, his church, and unplugging to be outside camping and/or playing music.

Zainul Setyo Pamungkas is a web developer who is interested in mobile development. He has been involved in web development for four years, primarily working with PHP and sometimes with Java. After playing around with Java for Android development for some time, he switched to start mobile development using web technology.

For the past three years, he was a freelance web developer. Currently, he is working for an Indonesian company named Digital Semesta as a mobile developer and service integrator. He works with web technologies including PHP, Laravel, HTML5, jQuery Mobile, RESTful API, and SOAP web services. Mobile development is done using web technologies and the PhoneGap framework.

I'd like to thank God for giving me everything as I can't mention it one by one. I'd also like to thank my parents who have always supported me.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at <service@packtpub.com> for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access

I would like to dedicate this book to to my wife Olga Lucia and my daughter Laura for the support with every project. Mom and Dad, thanks for life; and yes, thanks to my aunties too.

--Gustavo De La Vega Alvarez

Chapter 1. Instant PhoneGap

Welcome to PhoneGap. This book has been specially created to provide you with all the information that you need to set up your development environment and build your first functional mobile app for Android and iOS using PhoneGap. You will learn the basics of PhoneGap, get started with PhoneGap setup, build your first mobile app using geolocation features, and discover some tips and tricks for using PhoneGap

This book contains the following sections:

So, what is PhoneGap? explains what PhoneGap actually is, what you can do with it, and why it's so great.

Installation explains how to download PhoneGap and then set it up so that you can use it as soon as possible.

Quick start – creating the My Places app explains how to develop your first app using PhoneGap with geolocation features. Follow the steps to develop the app, which will be the basis of most of your work in PhoneGap.

Top features you need to know about teaches you how to perform tasks with the most important features of PhoneGap using HTML5, CSS, JavaScript, geolocation features, and XML files. By the end of this section, you will be able to develop your first full functional app using geolocation that reads an XML file with the information about specific places and shows these places on the map.

People and places you should get to know provides you with many useful links to the project page and forums, as well as a number of helpful articles, tutorials, blogs, and the Twitter feeds of PhoneGap super-contributors.

So, what is PhoneGap?

For us, PhoneGap is the coolest framework to build cross-platform mobile apps using HTML5, CSS, and JavaScript. PhoneGap supports iOS, Android, Windows phone, BlackBerry, and WebOS before Version 3.0. PhoneGap 3.0 supports iOS, Android, and Windows.

PhoneGap allows us to build cross-platform mobile apps, avoiding the use of native languages such as Java, C#, and others. It creates standard APIs that allow developers to access native mobile OS features such as camera and GPS, using JavaScript.

It is really important to understand the difference between PhoneGap and Apache Cordova. Nitobi, a Canadian company, created PhoneGap a few

years ago. In 2011, it was acquired by Adobe. Later in 2011, Adobe/Nitobi donated the code to Apache Foundation as a project called Cordova. PhoneGap is a distribution of Cordova. For a detailed explanation about the difference, please visit the blog: http://phonegap.com/2012/03/19/phonegap-cordova-and-what's-in-a-name/.

As Cordova is an open source project, every developer in the world can contribute to Cordova by creating new APIs and writing tutorials and documents through the Apache Cordova Project. Please visit the site cordova.apache.org for more information.

PhoneGap created a service called **PhoneGap Build** that allows us to compile our applications in the cloud. This service allows us to avoid the SDK's maintenance and gives us other possibilities for building amazing mobile apps.

The mobile app market is evolving quickly. We, as developers, need to be updated about new native OSes, platforms, smartphones, and more. PhoneGap as a framework is the best choice to develop numerous mobile operating systems.

Using PhoneGap with tools such as jQuery Mobile, and with the help of this book, you will be able to build amazing mobile apps. Step-by-step, we will discover the way to build cross-platform apps using PhoneGap.

Installation

First, we need to set up our system to work properly with PhoneGap. To do that, we will follow various steps and take other actions to ensure the success of our project.

Step 1 – system requirements

We can develop for different platforms using SDKs and PhoneGap installation. But, in this book, we will include the easiest way to set up your system to develop with PhoneGap, and use PhoneGap Build to compile your mobile app.

We can download the latest version available from http://phonegap.com/install/.

If you want to set up your system using SDKs, then go to:

http://docs.PhoneGap.com/en/2.3.0/guide_getting-started_index.md.html.

Also, I recommend that you install the following elements before we start development using PhoneGap:

Operating systems

We can use any of the following operating systems of our preference:

- Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Ubuntu Linux and Lucid Lynx):

  The following are the requirements for a Linux operating system:

  - GNU C Library (glibc) 2.7 or later is required
  - On Ubuntu Linux, Version 8.04 or later is required
  - 64-bit distributions must be capable of running 32-bit applications

IDE or code editor

We can use any of the following IDEs according to our preference:

- Aptana Estudio
- Eclipse
- Xcode

- Notepad++ (Only on MS Windows)
- Text Mate
- TextWrangler

Step 2 – create your directory structure

You have to execute the following steps to create your directory:

1. From the ZIP file that you downloaded, go to the `Android` folder and then find the `www` folder.
2. Inside the `www` folder, create a directory with the name of the app. For this section, we call it `HelloWorld`.
3. In the `HelloWorld` directory, create the following directories, as shown in the following screenshot:

4. Move the `cordova.js` file to the `js` folder, and move the `index.html` file inside the `HelloWorld` root directory.

Step 3 – create your config.xml and index.html files

Depending on your choice of IDE or code editor, you will create the following files:

- Inside the `HelloWorld` directory, create the `config.xml` file. As an example, we show the minimum code we need to build our `HelloWorld` mobile app. The code is as follows:

```
<?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.helloworld"
    versionCode="10"
        version   = "1.0">

    <!-- versionCode is optional and Android
only -->

    <name>Hello World</name>
```

```
    <description>
        Hello World
    </description>

    <author href="http://nativapps.com"
email="support@nativapps.com">d
        NativApps
    </author>
  <feature
name=http://api.PhoneGap.com/1.0/notification/>
</widget>
```

> **Tip**
>
> **Downloading the example code**
>
> You can download the example code files for all Packt Publishing books you have purchased from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

You need to use an API to create a notification (alert), which can be done using the following line of code:

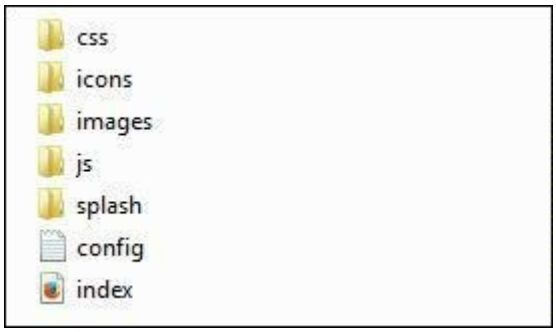
```
<feature
name=http://api.PhoneGap.com/1.0/notification/>
```

- Inside the `HelloWorld` directory, move the `index.html` file. Please ensure that your `index.hml` file has the following lines of code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World App</title>

    <script src="js/PhoneGap.js"></script>
    <script type="text/javascript"
charset="utf-8">

    // Initialization

    document.addEventListener("deviceready",
onDeviceReady, false);

    // We are ready

    function onDeviceReady() {
        // Empty
    }
```

```
        // Alerts dismissed
        function alertDismissed() {
            // empty
        }

        // Show Hello World Alert

        function showAlert() {
            navigator.notification.alert(
                'Your environment to develop is
OK!',  // message
                alertDismissed,         //
callback
                'Hello World App',     // title
                'OK'                  //
buttonName
            );
        }

    </script>
  </head>
  <body>
    <p><a href="#" onclick="showAlert();
return false;">Start Hello World App</a></p>
  </body>
</html>
```

- The following screenshot shows the result:

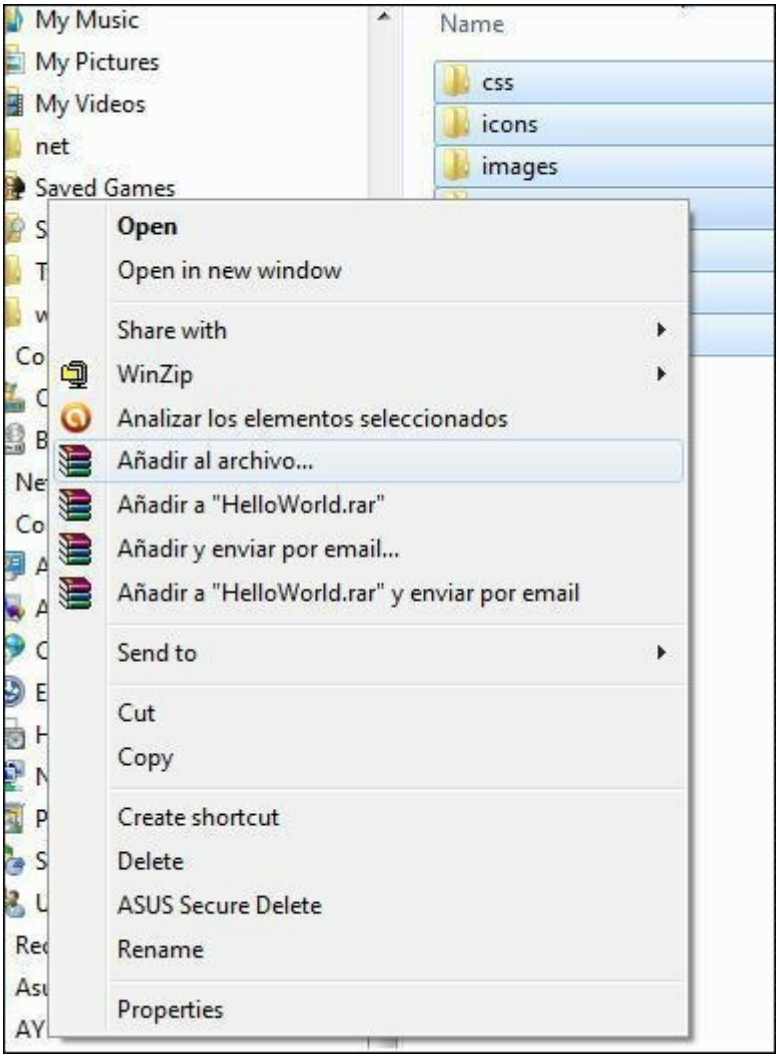


Step 4 – package your code

Now, we need to package our files and compile them using PhoneGap Build to test our development environment:

1. Select all the files in the `HelloWorld` directory, as shown in the following screenshot:

2. Create a ZIP file by right-clicking on the selected files:



3. If necessary, rename your file to `HelloWorld.zip`.

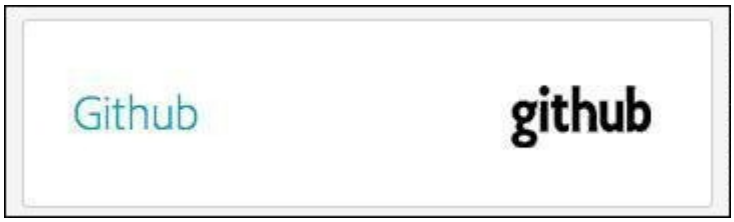
Step 5 – create different OS apps through PhoneGap's Build feature

The following steps need to be executed to create different OS apps through PhoneGap's Build feature:

1. Go to http://build.PhoneGap.com/.
2. Click on **Register**.
3. Choose **Free Plan**.

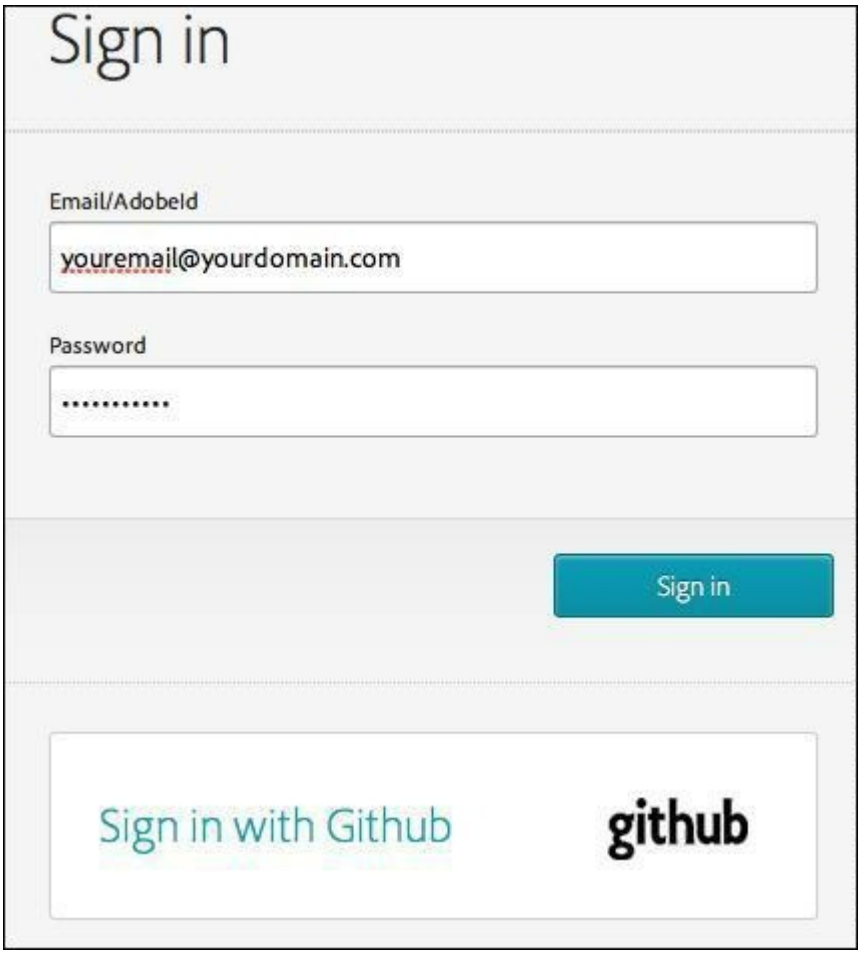
   Before you go to the next step, we need to be sure that we have our GitHub credentials; alternatively, go to https://GitHub.com/ and sign up for free.

4. Click on the **GitHub** option:

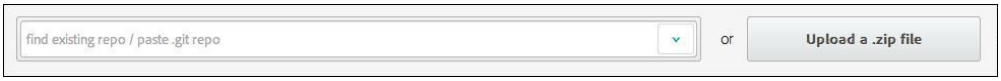


5. Click on **Authorize app**:

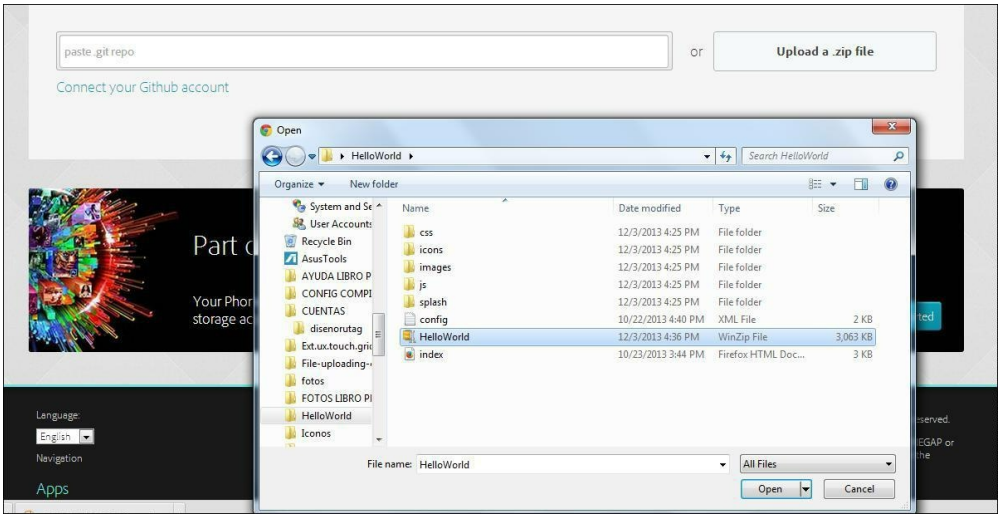
6. Click on **Sign in with GitHub**:



7. Agree with the terms and click on **Complete my registration**.
8. Then click on **Upload a .zip file**:

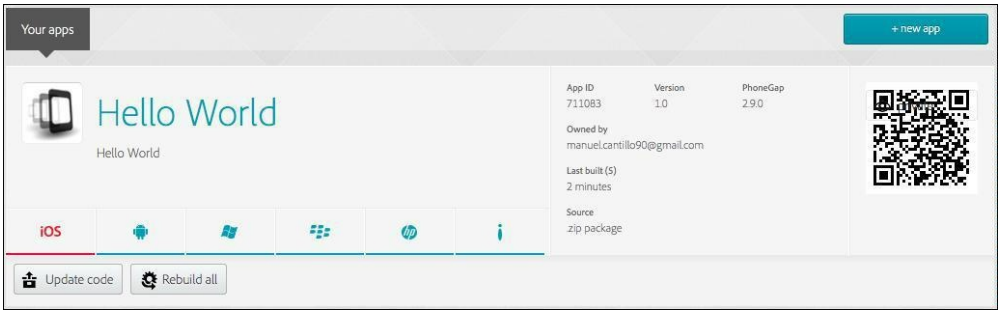


9. Select the `HelloWorld.zip` file and click on **Open**:



10. Click on **Ready to build**:

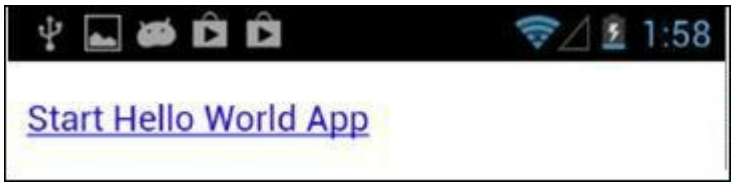
PhoneGap Build completes the process. And as a result, we will get to see the following screenshot:



Note

iOS will be in red because we need to include our iOS key for a successful build. To add our key, please click on the **iOS** icon and add the key.

11. Download the file to test on the device. We will use the `HelloWorld-debug.apk` Android file and install it on the device. Open the app, and you will get to see the following screenshot:



12. Click on the **Start Hello World App** link. If you get to see the following screenshot, then your test was successful:


1:58
Start Hello World App
Hello World App
Your development environment is OK!
OK

Quick start – creating the My Places app

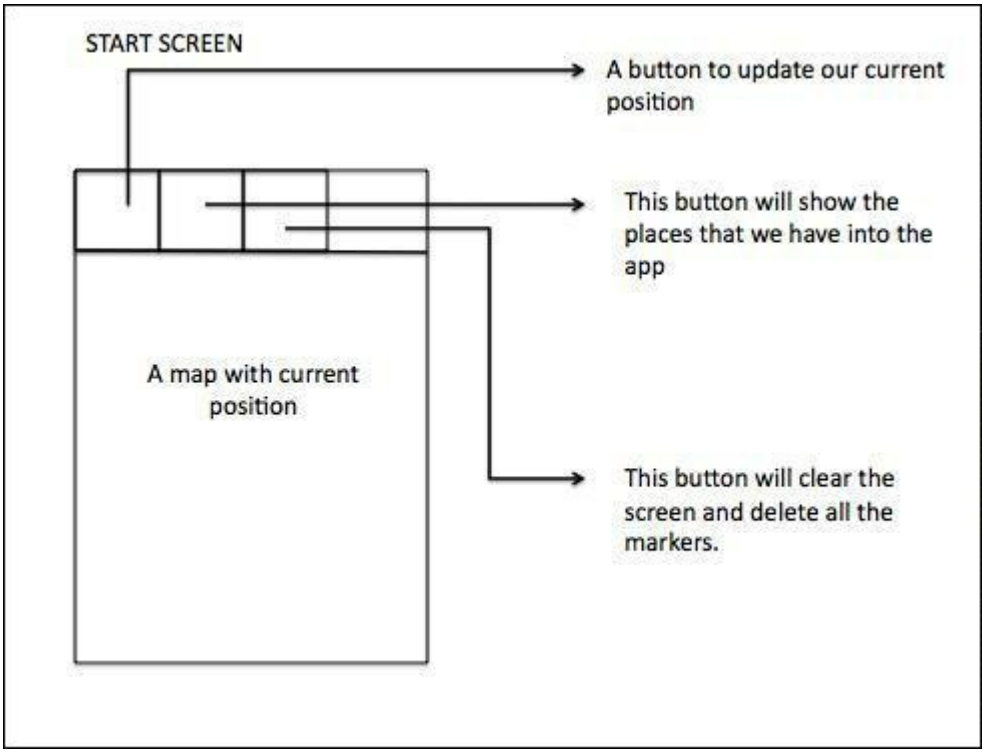
In this section, we start to build our first cross-platform app using PhoneGap. One of the most interesting features in the mobile world is geolocation. The app "My Places" will have the following features:

- **Splash screen**: This is used to start your app. It is always important that the user notices that we are working for them. Until our app starts, we need to show a fancy screen; we call it the splash screen.
- **User's current position**: Once you open the app, it shows you your current position and uses a function called watch position to refresh it after every *n* seconds. This is the main feature of our app. Once the app starts, it will show the user's position.
- **Position of the destination**: We will build an XML file with the position of our restaurants or places. The places are meaningful points for us, such as our home, favorite pizza restaurant, and our workplace. We will define an XML file with all the information, such as our name, short description, places, and their geolocation coordinates (longitude, latitude, and altitude).
- **Route**: This is used to show the route from your current position to the place that you choose using Google Maps API. In the app, you choose the place that you want to go, and the app will show you the way using Google Maps and the Google API.
- **Search bar**: This is used to search for information about the places that you have on your app. If we decide to include a lot of places, we will need a search bar to find what we are looking for in our app.
- **About your place**: This includes a short description, the geolocation coordinates, a picture, and some information about it.

Top features you need to know about

In this section, we will define the app's start screen and the other views of the app. We will keep in mind the rule "No more than three clicks from any option of the app" to design our app and build the views.

The first action when we want to build an app is to make a drawing of the screens and features that the app will have. For us, it is the most important exercise. With that, we can visualize the whole app and its features. The next screenshot shows our result:



After that, we will build the views. In a cross-platform app, the view means the CSS files, which define the styles and buttons' positions on our app.

Building the view's elements

Before we build all the views, we will need to build the elements that contain the views, for example, buttons and icons.

Buttons:

- **Current location**: It is the button that shows the main screen and updates the current location of the user. The following is the screenshot that represents the current location button:

- **My Places**: This button represents the places that the app has; for example, we are food lovers and decide that the points that we will show are restaurants and fast food places. That's the reason to choose this button with a knife and a fork to indicate our places. The following is a screenshot of the My Places button:

- **Clear button**: This button deletes all the points from the map view. The following is the screenshot of this button:

Icons:

- **Places icon**: We will use the following icon to identify our places:

- **Location icon**: We will use the following icon to identify our current position:

The views – find out how to build the app's views using CSS and HTML5

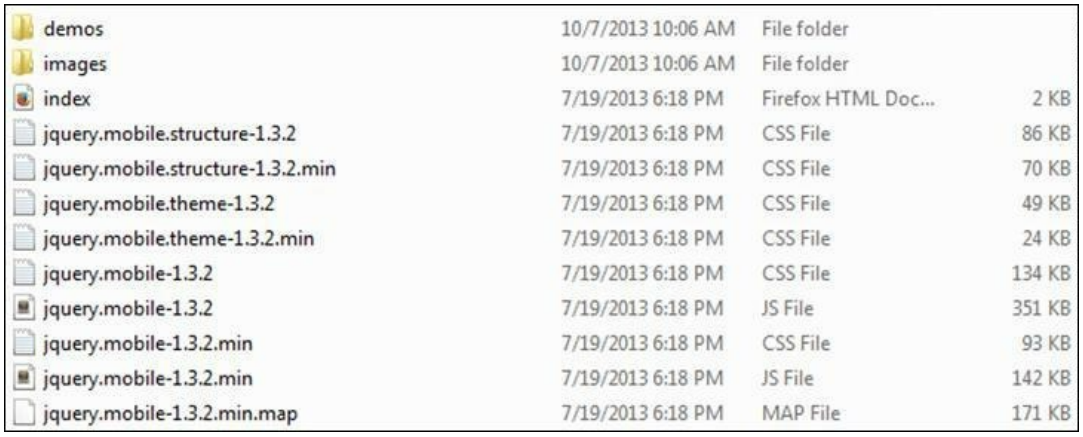
In this section, we will learn how to build the User Interface (UI) using the jQuery Mobile framework. This HTML5 framework gives us a variety of components to build the UI in an easy way.

Step 1 – download jQuery mobile

Go to http://jquerymobile.com/download/ and download the last compatible ZIP. In this case, the minimum version is 1.3.2. We can use a later version if it's available.

Step 2 – unzip the file

After downloading the file, proceed to extract the files. We will find a structure that shows all folders. It looks like the following screenshot:



We will use complete files on an advanced project but, for this book, we use only a few files. The files that we need to copy into our project's JS folder are as follows:

- `jquery.mobile-1.3.2.js`
- `jquery-1.9.1.js` (if we cannot find the file, we will download it from http://code.jquery.com/jquery-1.9.1.js)

From the `images` folder, copy all the files into our project's `images` folder. In our project, we will copy the following files into the CSS folder:

- `jquery.mobile.structure-1.3.2.css`
- `jquery.mobile.theme-1.3.2.css`

Note

We need to open the `jquery.mobile.theme-1.3.2.css` file and change all URLs from `images/` to `../images/`.

Once the preceding changes are done, our `index.html` file will look like the following snippet:

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Page Title</title>
    <meta name="viewport"
content="width=devicewidth,initialscale=1">

    <link
rel="stylesheet"href="css/jquery.mobile.structure1.3.2.css" />
    <link rel="stylesheet"
href="css/jquery.mobile.theme-1.3.2.css"/>

    <script src="js/jquery-1.9.1.js"></script>
    <script
src="js/jquery.mobile-1.3.2.js"></script>
</head>
<body>
    ...content goes here...
</body>
</html>
```

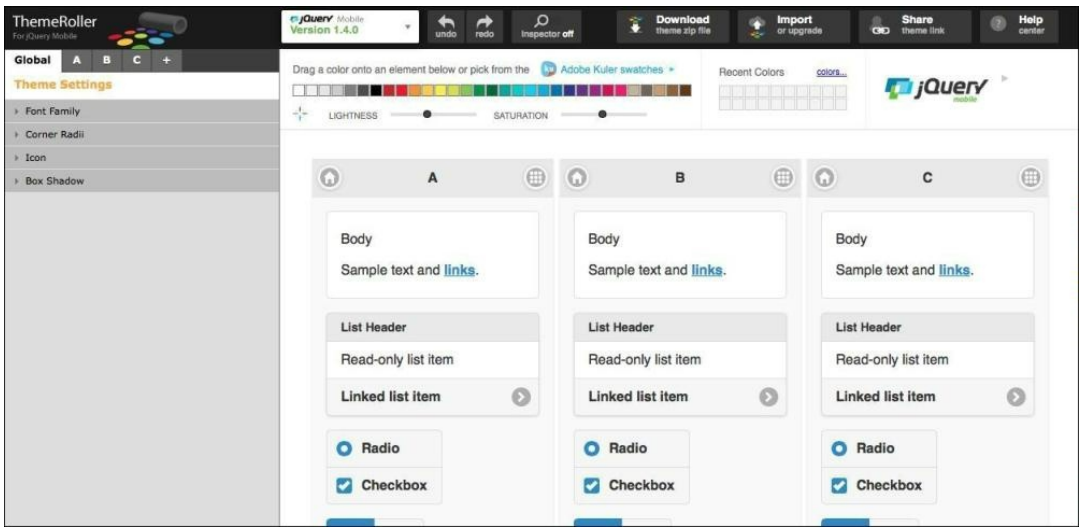
Now, we build our first view using the multipage technique provided by jQuery Mobile. We will have only one `index.html` file with the pages (views) of our project. The page has three sections: header, content, and footer. Once we include these sections in our `index.html` file, our code will look like the following snippet:

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
    <meta name="viewport" content="width=device
width,initialscale=1">

    <link
rel="stylesheet"href="css/jquery.mobile.structure1.3.2.css" />
    <link rel="stylesheet"
href="css/jquery.mobile.theme-1.3.2.css"/>

    <script src="js/jquery-1.9.1.js"></script>
    <script
src="js/jquery.mobile-1.3.2.js"></script>
</head>
<body>
<div data-role="page">
    <div data-role="header">
        <h1>Page Title</h1>
    </div><!-- /header -->
    <div data-role="content">
        <p>Page content goes here.</p>
    </div><!-- /content -->
    <div data-role="footer">
        <h4>Page Footer</h4>
    </div><!-- /footer -->
</div><!-- /page -->
</body>
```

```
</html>
```

Step 3 – customize the CSS

To customize our app and build it using our own style, we use the jQuery Mobile online tool called ThemeRoller. With this tool, we build our own style. This tool is available at http://jquerymobile.com/themeroller/. The following is the screenshot of the screen when we open the URL:

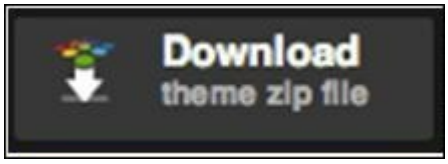


On the left side, we will find some tabs with different themes that we can change according to our style preferences. The sections that you can change are as follows:

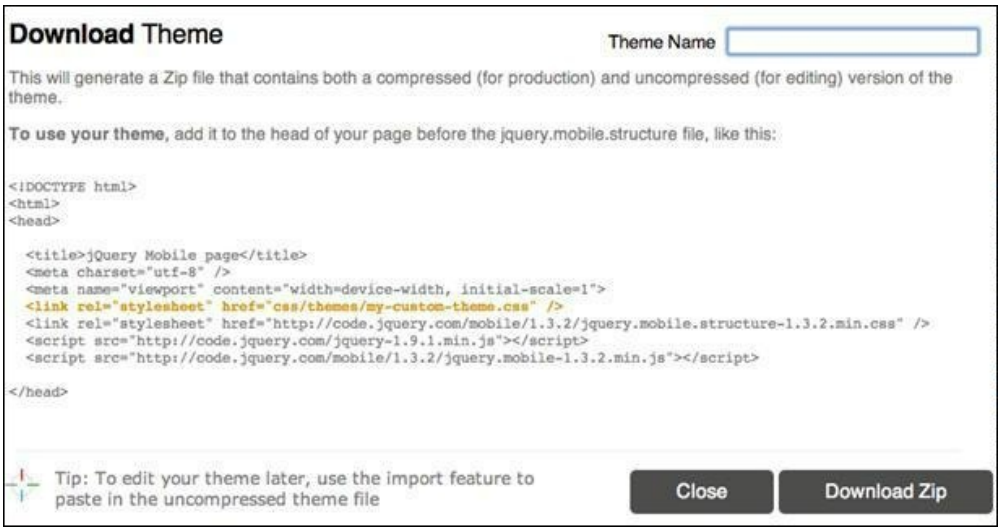
- **Header/Footer bar**
- **Content Body**
- **Button: Normal**
- **Button: Hover**
- **Button: Pressed**

We can change the options and attributes that we want depending on our design requirements. We are free to make any changes. Once we define the style, we will perform the following steps to put the new style on our project:

1. We will download the theme by clicking on **Download theme zip file**.



2. We will use the `ourtheme` name for the file and click on the **Download Zip** button:



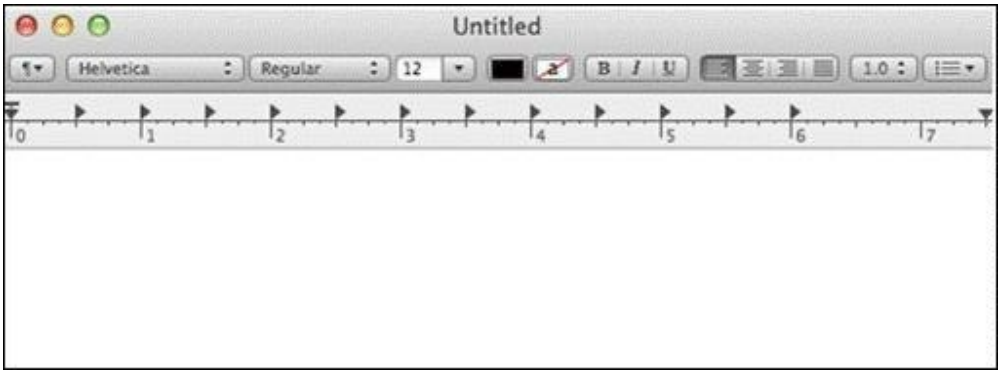
3. Unzip the file and save it in our `CSS` folder.
4. Open the file and change the URLs from `images/` to `../images/`.
5. We will change our `index.html` file by adding the following line:

```
<link rel="stylesheet" href="css/rutag.css" />
```

Step 4 – finishing the CSS

In this book, we do not cover CSS in detail. To finish our project using the correct CSS, perform the following steps:

1. Open a text editor program:



2. Save the new file as `rutag.css`.

> **Note**
>
> Don't forget to change the file type to **All Files**.

3. Copy-and-paste the `rutag.css` code from the `CSS` folder in the code bundle.
4. Repeat from step 1 using the name `my.css` and the `my.css` code from in the `CSS` folder in the code bundle.
5. Finally, we will include the following line into our `index.html` file:

```
<link rel="stylesheet" href="css/my.css" />
```

The following is the screenshot of our views:

JavaScript – find out the way to develop the events and actions of your app

In order to develop the events and actions of your app, you need to follow the following steps:

Step 1 – Build the XML file

To build your app, we need to define our places and the data that we want to keep from our point of view. To do that, we need to create an XML file using the standard form. First, we define the name of the data that we will keep on our app. We need to provide the following information:

- **Name**: The way we know the place
- **Address**: The way to get there

- **Description**: More information about the place as we don't want just the name
- **Latitude**: Coordinates
- **Longitude**: Coordinates
- **Altitude**: Coordinates
- **Phone**: Just a number to call
- **Website**: Just for fun

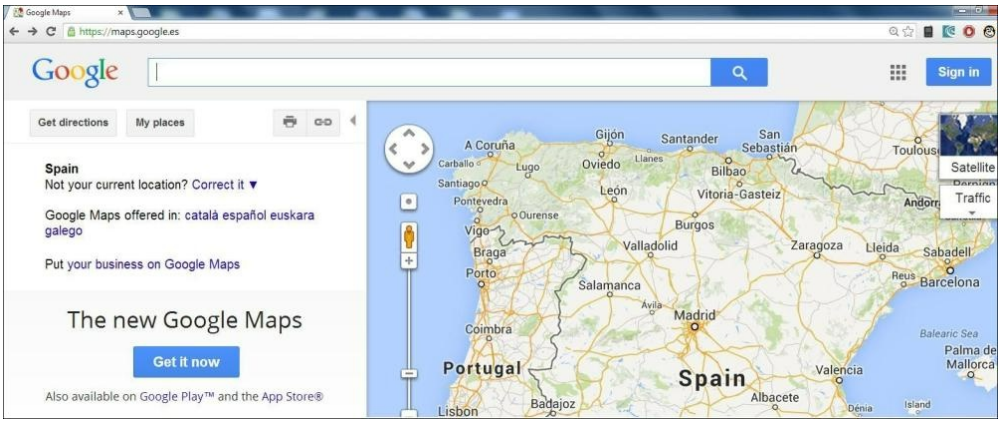
Second, we create our tags using a standard XML quotation:

```
<?xml version="1.0" encoding="UTF-8"?>
<points>
<point>
<name></name>
<address></address>
<description></description>
<latitude></latitude>
<longitude></longitude>
<altitude></altitude>
<phone></phone>
<website></website>
</point>
<points>
```

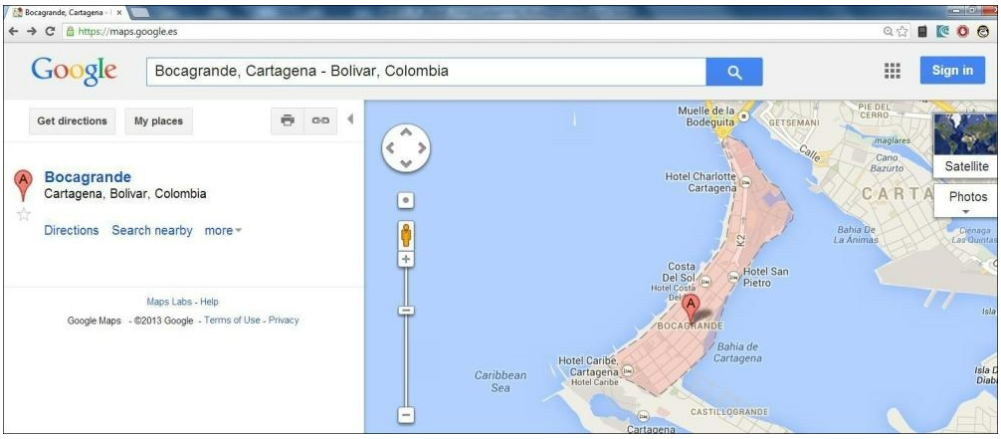
Step 2 – define places

Our XML file is not complete without information; our next step is to fill our XML file with the correct info (for this book, we will use some restaurants in town). We start with Author's Restaurant (not a real name).

1. Get the coordinates and then open Google Maps using http://maps.google.com on your browser:



2. We find our restaurant on the map. Enter `Bocagrande, Cartagena - Bolivar, Colombia` as it is in the Google search field, and hit the *Enter* key:



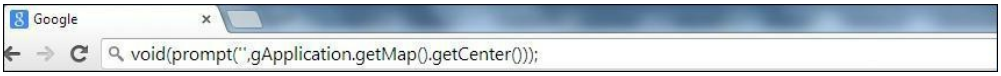
Note

Cartagena is an amazing and beautiful place in the Caribbean; if you decide to take a vacation, come and visit our city.

3. We choose the place with the mouse, right-click on it and select `Center map here`:

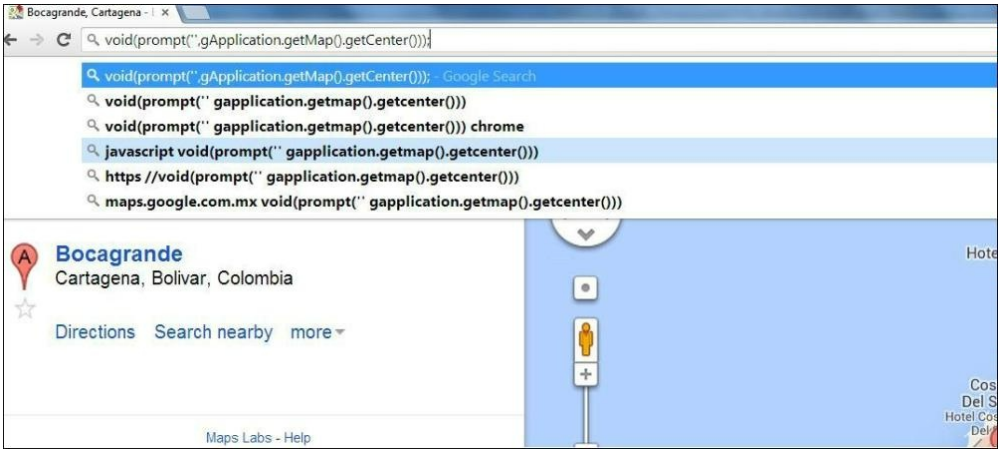
4. Write the following command in the URL field:

```
javascript:void(prompt('',gApplication.getMap().getCenter()));:
```



Note

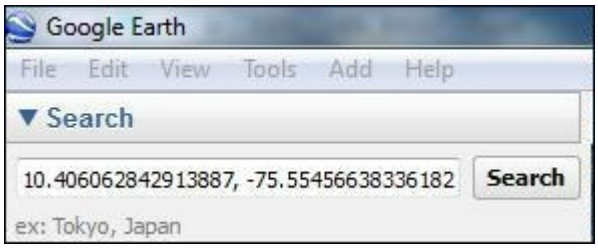
We need to be careful because the browser will delete the word `javascript:`. If that happens, we have to rewrite the word.



5. We get the latitude and longitude:

6. We take note of the numbers (`10.406062842913887, -75.55456638336182`) and open Google Earth, as we need to know the altitude:

7. Enter the latitude and longitude in the search box and click on **Search**:



8. We take the altitude number from the right-hand corner at the bottom of the page:

After that, we are ready to complete our first point with the rest of the information. Our first XML point looks like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<points>
<point>
<name>Author's Restaurant</name>
<address> First street around the corner</address>
<description>An amazing place to eat. Summer the
whole year and from the window we enjoy and
amazing Caribbean sea view</description>
<latitude>10.406062842913887</latitude>
<longitude>-75.55456638336182</longitude>
<altitude>11</altitude>
<phone>+5756535530</phone>
<website>www.packtpub.com</website>
</point>
<points>
```

We repeat steps from step 1 to step 8 to create our second point. After that, we get our two-point XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<points>
<point>
<name>Author's Restaurant</name>
<address> First street around the corner</address>
<description>An amazing place to eat. Summer the
whole year and from the window we enjoy and
amazing Caribbean sea view</description>
<latitude>10.406062842913887</latitude>
<longitude>-75.55456638336182</longitude>
```

```
<altitude>10</altitude>
<phone>+5756535530</phone>
<website>www.packtpub.com</website>
</point>
<point>
<-- Same as above… !-->
</point>
<points>
```

Geolocation – using PhoneGap features to improve an app's functionality, write once use everywhere

To implement geolocation in our app, we will use the API that PhoneGap provides us as part of a bunch of methods to use through our app. To build this, we need to create a JS file called `myplaces.js`. Create the file once using a text editor program such as Sublime Text or Notepad. Then we will perform the following steps:

Step 1 – define global variables

We include the following variables and functions in our file:

```
var map;
var latitud;
var longitud;
var xmlDoc = loadXml("puntos.xml");
var marker;
var markersArray = [];

function loadXml(xmlUrl) {
  var xmlhttp;
  if (window.XMLHttpRequest) {
        xmlhttp = new XMLHttpRequest();
  } else {
    xmlhttp = new
ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.open("GET", xmlUrl, false);
  xmlhttp.send();
  xmlDoc = xmlhttp.responseXML;
  return xmlDoc;
}
```

With this code, we build our first function called `loadXml` that just loads the information into the XML file and then in to the smart phone memory.

Step 2 – get current position

We will build the following functions that we need to show our current position in Google Maps:

- `getCurrentPosition`: This function receives an object with the latitude and longitude. These values are set to two variables defined as global variables with the same name. This function receives three parameters, which are as follows:
    - A function to manage the current latitude and longitude
    - A function to manage errors if they occur when the device is trying to get the position
    - Options to configure
- `maximumAge`: This is the time required to keep our position on the cache in milliseconds
- `timeout`: This is the maximum time required to wait for an answer from the `getCurrentPosition` function in milliseconds
- `enableHighAccuracy`: This provides a more accurate location

The values of these functions can be set to either `True` or `False`.

The following is the code for the preceding functions:

```
getCurrentPosition(coordinates, errors, {
  maximumAge : 3000,
  timeout : 15000,
  enableHighAccuracy : true
}
```

The following is the code for the `coordinates` and `errors` functions:

```
function coordinates(position) {
  latitud = position.coords.latitude; /*
saving latitude */
  longitud = position.coords.longitude; /*
saving longitude*/
  loadMap();
}// end function


function errors(err) {
  /* Managing errors */
  if (err.code == 0) {
    alert("Oops! Something is wrong");
  }
  if (err.code == 1) {
    alert("Oops! Please accept share your
position with us.");
  }
  if (err.code == 2) {
```

```
    alert("Oops! We can't get your current
position.");
  }
  if (err.code == 3) {
    alert("Oops! Timeout!");
  }
}// end errors
```

- `LocateMe`: This function checks whether the device supports the PhoneGap geolocalization API or not. If the device supports the API, then this function calls a method to obtain the current position. If the device doesn't support geolocalization API, then we will use the current position's function, `getCurrrentPosition`, explained previously. The complete code for the `LocateMe` function is as follows:

```
function locateMe() {
  if (navigator.geolocation) { /* The browser
have geolocalization */

 navigator.geolocation.getCurrentPosition(coordinates,
errors, {
      maximumAge : 3000,
      timeout : 15000,
      enableHighAccuracy : true
    });
  } else {
    alert('Oops! Your browser doesn't support
geolocalization');
  }
}
```

Step 3 – showing the current position using Google Maps

To do this, we use a function that we called `loadMap`. This function is responsible for loading a map using Google Maps API with the current position. Also, we use a JavaScript object called `myOptions` with three properties, `zoom` to define the zoom, `center` to define where the maps will be centered, and `mapTypeId` to indicate the map style (that is Satellite).

Following the code, we find two lines: the first one initializes the `actualHeight` variable according to the value that is returned by the `getActualContentHeight` function. This function returns the height that the map needs to have according to the screen size. In the second line, we change the height of the div "map canvas" through the jQuery Mobile method's CSS. In the following code file, we set the variable map with the `google.maps.Map` object that receives the parameters, the HTML element, and the `myOptions` object. We use an additional event to resize the map when the screen size changes.

Now, we create a new marker object `google.maps.Marker` with four properties: position for the place of the marker, map that is the global

variable, icon to define the image that we want to use as a marker, and the tooltip with the property title.

We have two functions: the first one is `createMarkers` that allows us to read the XML file with the points uploaded in memory and, after that, puts the markers in the map with our second function `addMarker` that receives four parameters (the global map variable, point name, address and phone, and the `google.maps.LatLng` object with the point coordinates) to build the marker. We add a `click` event to show all the information about the place.

Finally, we use the `removePoints` function that clears the maps, thereby deleting the points loaded.

This is the code for the JS file:

```
function loadMap() {
  var latlon = new google.maps.LatLng(latitud,
longitud);
var myOptions = {
    zoom : 17,
    center : latlon,
mapTypeId : google.maps.MapTypeId.ROADMAP
  };
  map = new
google.maps.Map(document.getElementById('map_canvas'),
myOptions);

    var mapDiv =
document.getElementById('map_canvas');
    google.maps.event.addDomListener(mapDiv,
'resize', function(){

      google.maps.event.trigger(map,'resize');
    });
  var coorMarker = new
google.maps.LatLng(latitud, longitud);

   marker = new google.maps.Marker({/* Create the
marker */

    position : coorMarker,
    map : map,
    icon : 'images/green-dot.png',
    title : "Where am I?"
  });

}
```

Step 4 – edit the HTML file

To implement the preceding functions, we need to add some code lines in our HTML file. We need to add the script line to include the new JS file, and

add a complete JavaScript in the HTML head tag to execute the `locateMe` function. Now that the device is ready, another PhoneGap function calls `watchPosition` to update the position when the user is moving.

The following are the lines to add to our HTML file:

```
<script src="js/myplaces.js"></script>
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?
sensor=true"></script>
<script type="text/javascript">
  document.addEventListener("deviceready",
onDeviceReady, false);
  var watchID = null;
  function onDeviceReady() {
    locateMe();
    var options = {
      enableHighAccuracy : true
    };
    watchID =
navigator.geolocation.watchPosition(onSuccess,
onError,options);
  }
  // onSuccess Geolocation
  //
  function onSuccess(position) {

    var latlng = new
google.maps.LatLng(position.coords.latitude,
    position.coords.longitude);
    marker.setPosition(latlng);

  }
  function onError(error) {
    alert('code: ' + error.code + '\n' +
'message: ' + error.message + '\n');
  }
</script>
</head>
<body>
  <div id="basic_map" data-theme="a"
data-role="page">
    <div  data-role="header" class="hea"
      data-position="fixed"
data-tap-toggle="false" data-fullscreen="false">
        <div class="ui-block-a settings3">
          <div align="left">
            <table id="SearchTable">
            <tr>
                <td>
                <a onclick="locateMe()" >
                <img  alt=""
src="images/01miubicacion.png">
                </a>
```

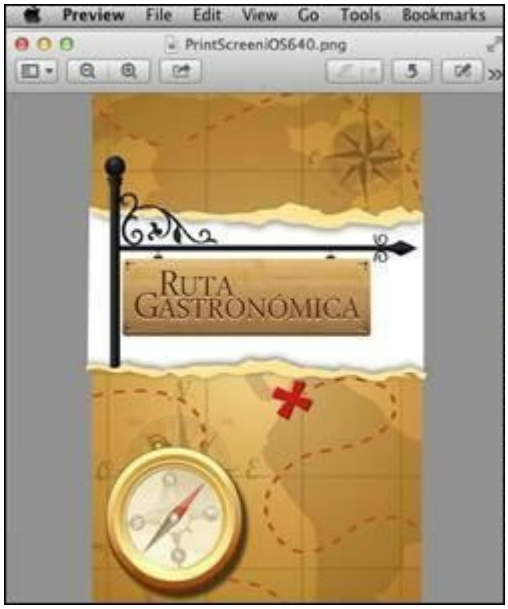
```
                </td>
                <td>
                <a onclick="showMyPoints()">
                <img  alt=""
src="images/02restaurant.png">
                </a>
                </td>
                <td>
                <a onclick="removePoints()"><img
alt="" src="images/05info.png">
                </a>
                  </td>
              </tr>            </table>
          </div>
        </div>
    </div>
    <!-- /header -->
      <div data-role="content"
id="map_canvas"></div>

  </div>
  <!-- /page -->

  </body>

  </html>
```

Splash screen – finding out a fancy way to say hello to the app's users

In this chapter, we will create a splash screen for "My Places":

Step 1 – define the splash screen's image

Today, we have a lot of screen sizes and resolutions. It is really important to create a splash screen to show to our users when the app starts. This splash screen is necessary because all the apps take one or two seconds to load, and this screen will give useful information to our users. A screenshot of the splash screen of an app is as follows:



Step 2 – build resolutions and sizes

Depending upon the platform we use, we will have to build a splash screen with different resolutions.

The following are the different resolutions available for iOS:

- 320 x 480
- 640 x 960
- 640 x 1136
- 768 x 1004
- 1024 x 748
- 1536 x 2008
- 2048 x 1496

The following are the different resolutions available for Android:

- 320 x 426
- 320 x 470
- 480 x 640
- 720 x 960

Note

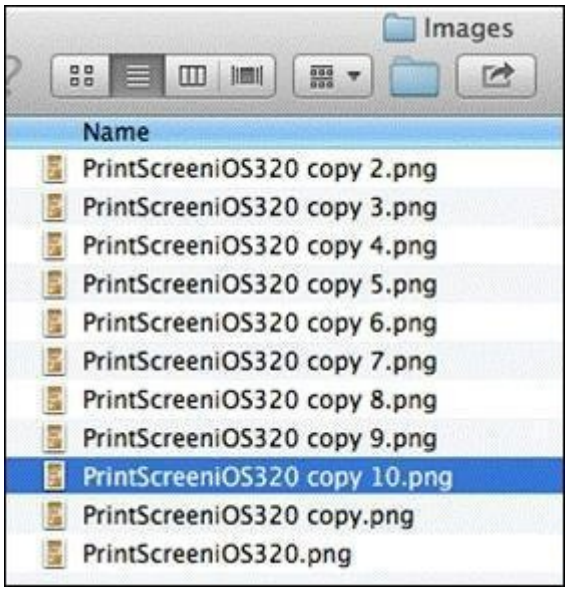
We will check for some resolution upgrades. Probably, you will find more resolutions.

The process to create the splash screens is simple. First of all, you need to choose the image and design it using any of the programs that work for

creating images. My recommendation is to create the first image using the highest resolution possible, and then perform the following steps:

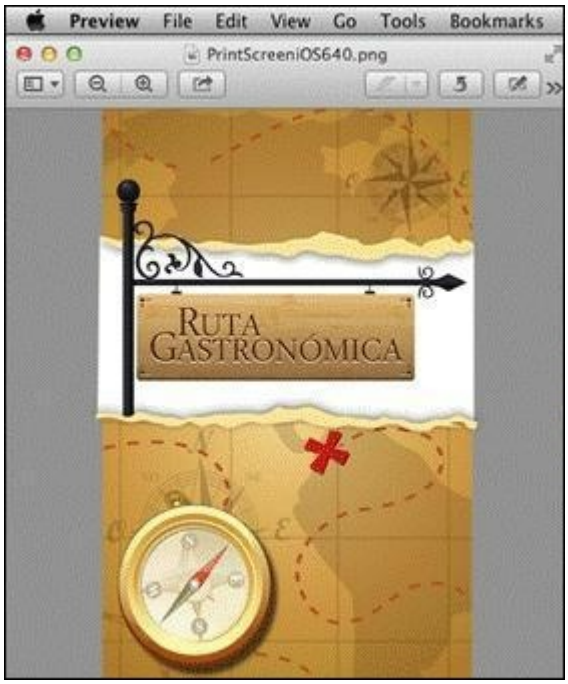
For Mac:

1. Make as many copies as you need for the resolutions of splash screens:



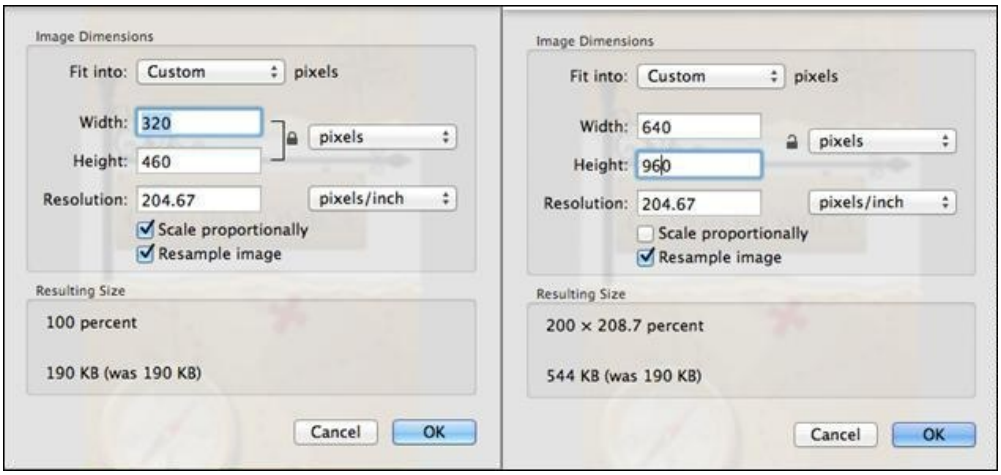
2. Rename the file with a name that is related to the resolution of the image (`PrintScreeniOS640.png`):

3. Open the image by double-clicking on it. As a result, we get the following screenshot:

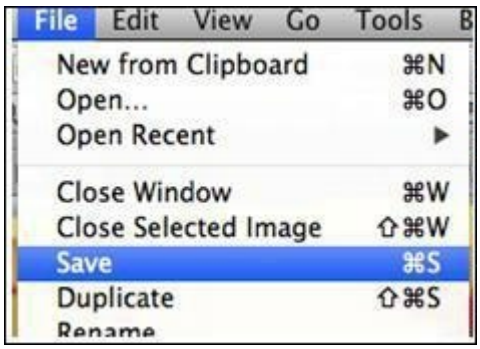


4. Go to **Tools** | **Adjust Size**:

5. In the resulting screen, we uncheck **Scale proportionally** and enter the new resolution size; for example, we choose `320` x `460`:



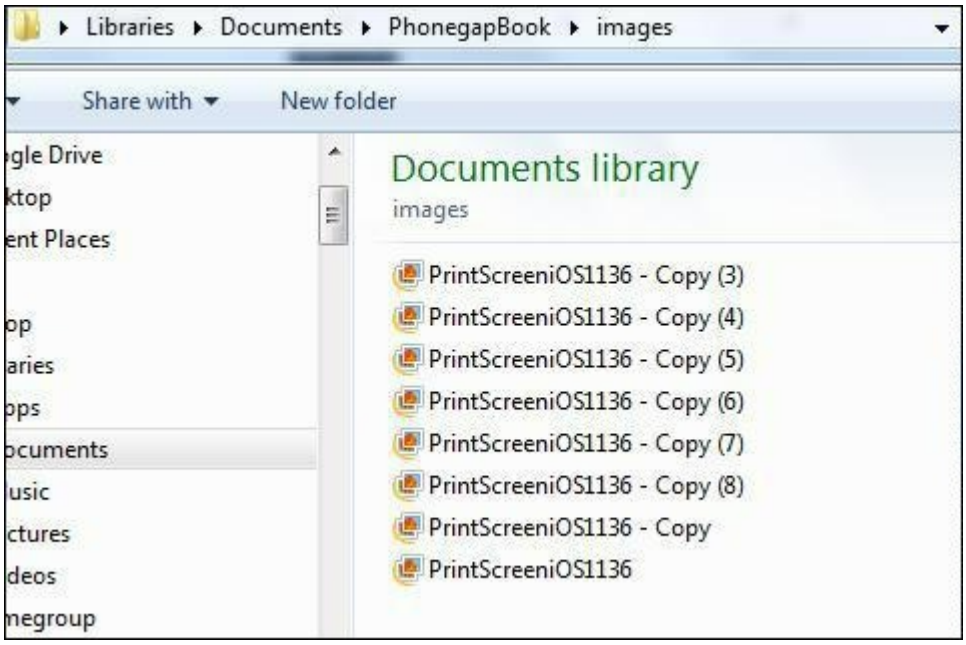
6. Then click on **OK**.
7. Navigate to **File** | **Save** and save the file:



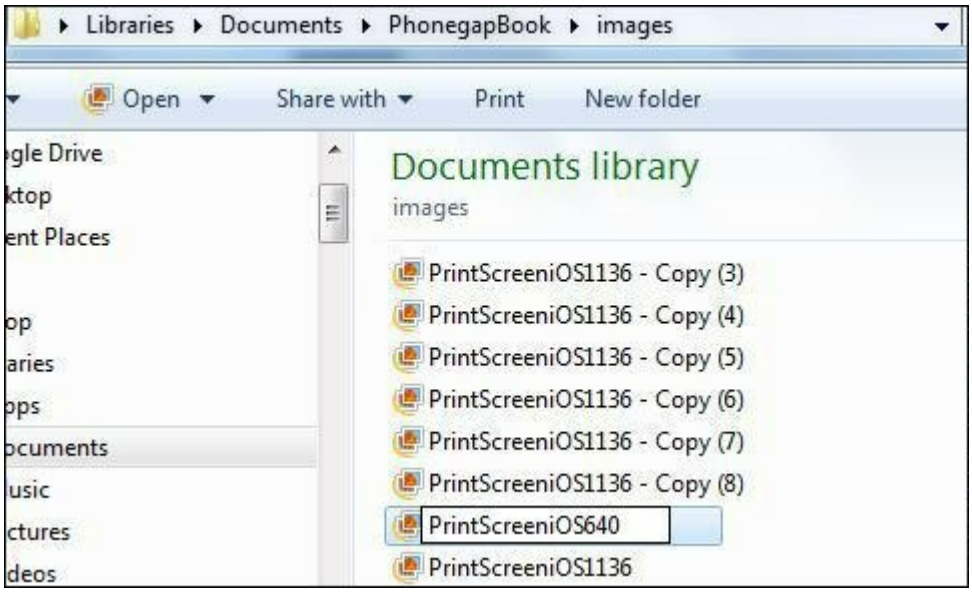
Starting from step 2, we again repeat the steps for as many times as the number of resolutions we need to build.

For Windows:

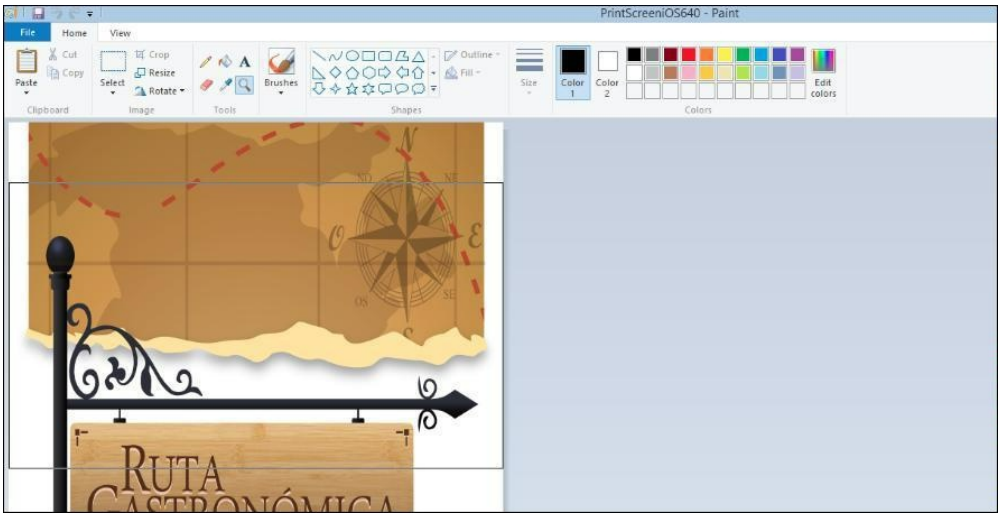
1. Make as many copies as you need for the resolutions of splash screens:



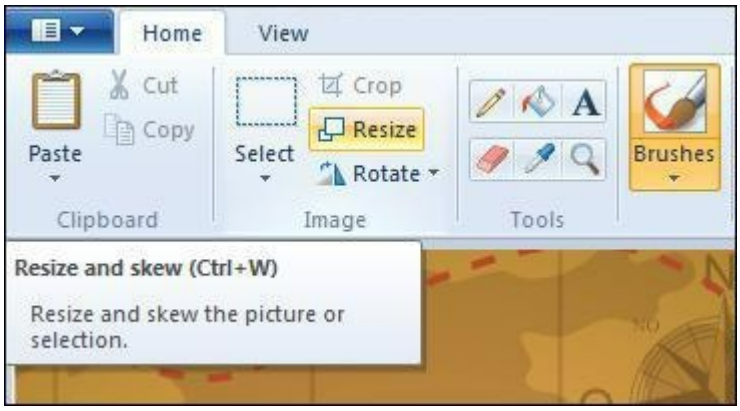
2. Rename the file with a name related to the resolution of the image (`PrintScreeniOS640.png`):



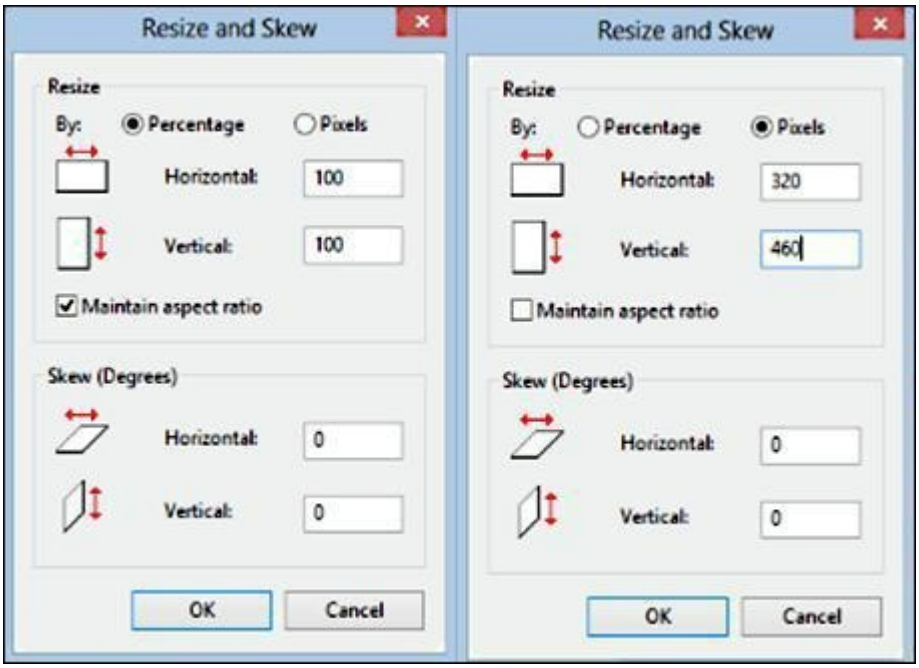
3. We right-click on the image and go to **Open With** | **Paint**. As a result, we get the following screenshot:



4. Click on **Resize**:



5. As shown in the following screenshot, we select **Pixels**, uncheck **Maintain aspect ratio**, and enter the new resolution size. We choose `320` **x** `460`:



6. Click on **OK**.
7. Click on **Save**:

Starting from step 2, we again repeat the steps for as many times as the number of resolutions we need to build.

Step 3 – code the script

Now, we need to implement the necessary code to show or hide the splash screen. To do that, we need to make some changes in our `config.xml` file. Before that, our file looks like the following snippet:

```
<?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.helloworld"
    versionCode="10"
        version   = "1.0">

    <!-- versionCode is optional and Android only
-->

    <name>Hello World</name>
```

```
    <description>
        Hello World
    </description>

    <author href="http://nativapps.com"
email="support@nativapps.com">d
        NativApps
    </author>
    <feature
name=http://api.PhoneGap.com/1.0/notification/>
</widget>
```

The code to implement the splash screen on an Android device is as follows:

```
<gap:splash src="splash/android/ldpi.png"
gap:platform="android" gap:density="ldpi" />
<gap:splash src="splash/android/mdpi.png"
gap:platform="android" gap:density="mdpi" />
<gap:splash src="splash/android/hdpi.png"
gap:platform="android" gap:density="hdpi" />
<gap:splash src="splash/android/xhdpi.png"
gap:platform="android" gap:density="xhdpi" />
```

We include the preceding code before the `widget` tag. After this inclusion, our `config.xml` file looks like the following snippet:

```
<?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.helloworld"
    versionCode="10"
        version   = "1.0">

    <name>My Places</name>

    <description>My Places</description>

    <author href="http://nativapps.com"
email="support@nativapps.com">d
        NativApps
    </author>
    <feature
name=http://api.PhoneGap.com/1.0/notification/>

<gap:splash src="splash/android/ldpi.png"
gap:platform="android" gap:density="ldpi" />
<gap:splash src="splash/android/mdpi.png"
gap:platform="android" gap:density="mdpi" />
<gap:splash src="splash/android/hdpi.png"
gap:platform="android" gap:density="hdpi" />
```

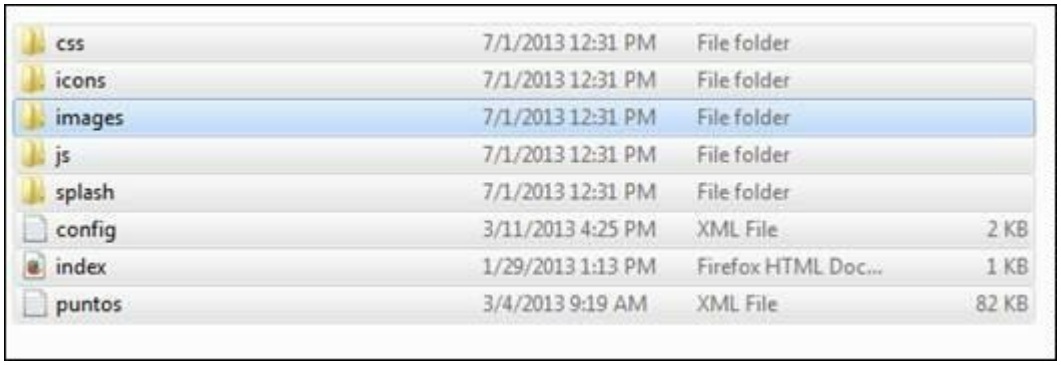
```
<gap:splash src="splash/android/xhdpi.png"
gap:platform="android" gap:density="xhdpi" />

</widget>
```

Test and enjoy – compiling your app in different platforms using Build

In order to compile the app in different platforms, we need to follow the following steps:

Step 1 – validate the app's package

We need to verify that our files are like the following screenshot. We need to pay special attention to our `config.xml` file:



Step 2 – log in and build the app using PhoneGap's build feature

To log in and build the app using PhoneGap's build feature, we need to execute the following steps:

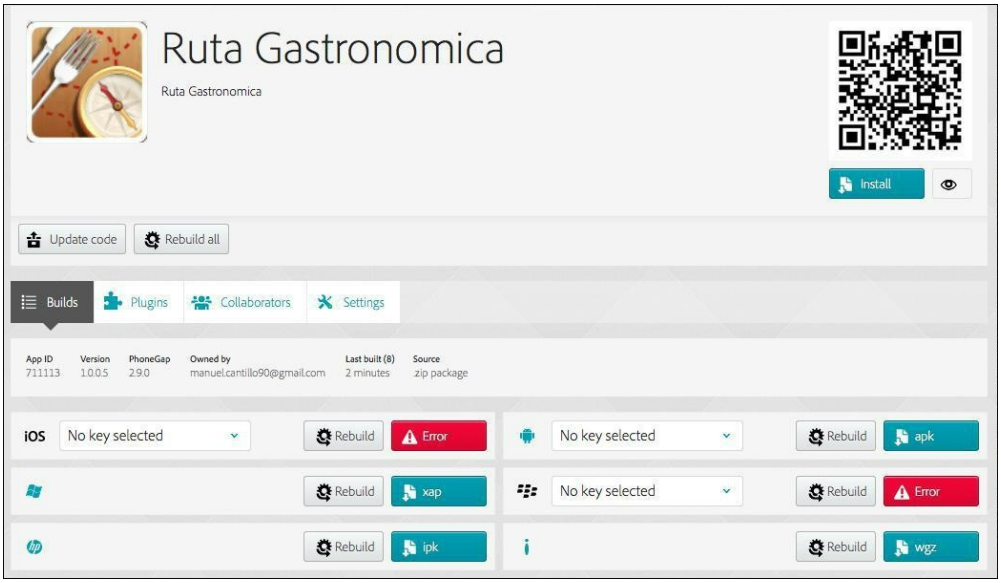
1. Now, we log in to https://build.PhoneGap.com; if you are not a user yet, follow the steps mentioned in the *Installation* section and create the user.

   Remember that we created the PhoneGap user to build our `Hello World` app.

2. We create the ZIP file of our project.
3. Upload the ZIP file using the **Update code** option:

4. Select the file and click on the **Upload** button:

5. We have to wait for a few minutes until PhoneGap creates the job. After PhoneGap has finished creating the job, the screen looks as follows:



Step 3 – download the file and install the app on an Android device

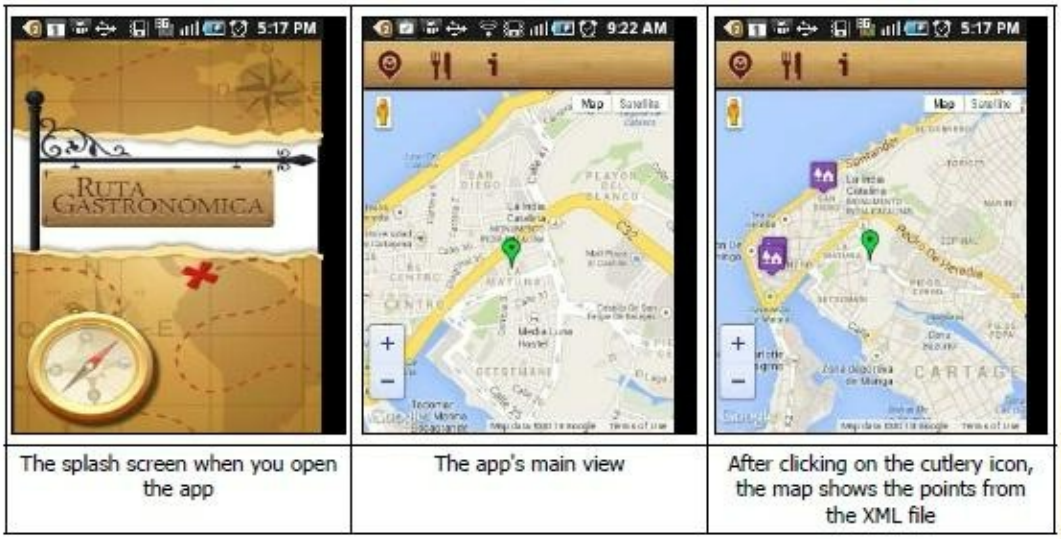
We need to execute the following steps to download and install the app on an Android phone:

1. Select the **.apk** button.
2. Transfer the file between your computer and your Android device using a USB connection.
3. Install the app in the device by clicking on the file.

> **Note**
>
> We can even install the file by scanning the QR Code on the screen with the smartphone using any available app for scanning.

Step 4 – test

Click on the icon on the screen and start the app. We will get some screens like the following screenshots:



The splash screen when you open the app	The app's main view	After clicking on the cutlery icon, the map shows the points from the XML file

More features you need to know about

We will want to take our app to the next level. Some of the most important PhoneGap features are as follows:

- **Push notifications**: If we share our app with our family and friends, we will probably want to send them information about the places or about app updates. For this, we will use the push notifications plugins. You can check this link to implement this feature: https://GitHub.com/PhoneGap-build/PushPlugin.
- **Accelerometer**: If we are thinking of building the "How to get there function", we will use the accelerometer feature. You can check this link to implement this feature: http://docs.PhoneGap.com/en/2.0.0/cordova_accelerometer_accelerometer.md.html.
- **Compass**: We will need the compass feature to implement "How to get there function" too, so please review this link to implement this feature: http://docs.PhoneGap.com/en/2.0.0/cordova_compass_compass.md.html#Compass.

People and places you should get to know

Finally, to keep yourself updated about PhoneGap and other cross-platform developments, you should visit the following:

Official sites

- http://cordova.apache.org
- www.PhoneGap.com
- http://build.PhoneGap.com

Articles and tutorials

- http://PhoneGap.com/blog/tag/tutorial/
- http://devgirl.org
- http://PhoneGap.com/blog/PhoneGap-build/

Twitter accounts

- `@elcabedlv`
- `@PhoneGap`
- `@brianleroux`
- `@davejohnson`
- `@stevesgill`
- `@apperyio`
- `@maxkatz`

Tools

- http://appery.io
- https://steroids.appgyver.com
- http://jquerymobile.com
- http://www.sencha.com

Community

- http://PhoneGap.com/community/