Getting started with PhoneGap in Eclipse for Android

by Andrew Trice

Adobe

Content

Created
14 March 2012

Page tools

Requirements

Prerequisite knowledge
While not required, some knowledge of HTML, JavaScript, CSS, XML, and Eclipse will help you make the most of this article.

Additional required other products
Eclipse Classic
- http://www.eclipse.org/downloads/

Android SDK
- http://developer.android.com/sdk/index.html

PhoneGap
- http://phonegap.com/download

User level
All

Note: Be sure to change the version number of the Cordova JavaScript file to match the version of PhoneGap/Cordova that you are using.

Eclipse is an open source integrated development environment (IDE) that supports many technologies, but this article is focused on its support of Java, the native language for Android applications. Android is Google's open source mobile operating system. Android is the operating system for many smartphone and tablet devices, including the Samsung Galaxy line of phones and tablets, the Amazon Kindle Fire tablet, and the Barnes and Noble Nook tablet, as well as many other devices from numerous manufacturers. PhoneGap is an open source application platform that enables you to create natively-installed mobile applications using HTML and JavaScript.

Setting up Eclipse

The first step in setting up your development environment for PhoneGap applications on Android is to download and install the Eclipse IDE.

Android development with PhoneGap can be done in Windows, OS X, or Linux. There are many different installation packages for Eclipse. While PhoneGap may work with other package configurations, the Eclipse Classic package is recommended and already includes tools that you need to get started and be productive with PhoneGap application development.

1. Visit the Eclipse downloads page to download the Eclipse Classic package for your operating system. The Eclipse download will be an archive containing the development environment.
2. Extract the archive to your local hard disk and remember its location.
3. Once extracted, you can launch Eclipse by double-clicking the Eclipse application, without any additional setup steps.

Setting up Android Tools

After you have downloaded and set up Eclipse, you will need to configure your environment to use Google's Android development tools. There are two steps to this process. First, you download and install the Android SDK. Second, you install the ADT plugin for Eclipse.

Download and configure the Android SDK

The first step in configuring Android tools on your system is to download the Android SDK.

1. Visit the Android SDK site to download the appropriate build for your operating system.
2. Extract the downloaded archive to your local hard drive and remember its location.

Configure the ADT Plugin for Eclipse

Next, you need to set up the ADT (Android Development Tools) plugin for Eclipse. The ADT plugin must be installed through the Eclipse Install New Software wizard.

1. Start Eclipse.
2. Follow the download instructions for the ADT plugin, available at the Android developer SDK page for Eclipse. These steps will guide you through the installation of the ADT plugin.
3. Restart Eclipse.

Once you've installed the ADT plugin and restarted Eclipse, you need to configure it to use reference the Android SDK that you have already downloaded to your local file system.

4. Follow the instructions on the Android developer SDK page for configuring Eclipse to set the appropriate Android SDK location in the ADT plugin.

Downloading and installing PhoneGap

The next step is to download and set up PhoneGap.

1. Visit the PhoneGap download page and click the orange Download link to begin the download process.
2. Extract the archive to your local file system for use later.

You are now ready to create your first PhoneGap project for Android within Eclipse.

Note: The steps that follow are for PhoneGap 1.5, but the process should be applicable or similar for all versions of PhoneGap.

Creating the project in Eclipse

Follow these steps to create a new Android project in Eclipse:

1. Choose New > Android Project (see Figure 1).

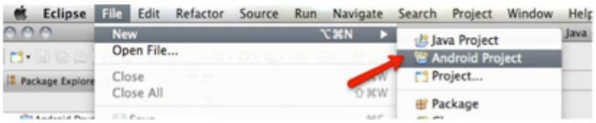


Figure 1. Creating a new Android project.

After you create a new, standard Android project you will update that project to use PhoneGap.

2. In the New Android Project dialog box, type a project name and select Create New Project In Workspace (see Figure 2).
3. Click Next.

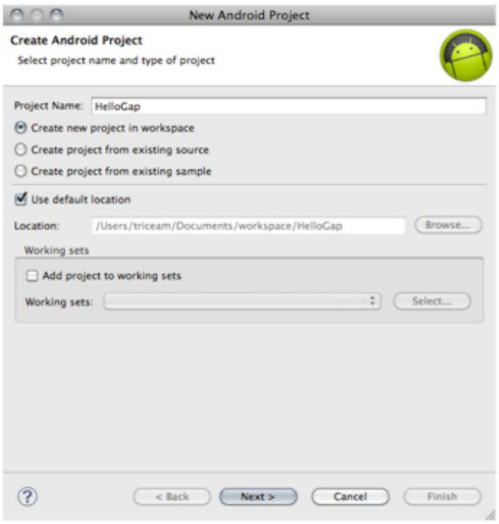


Figure 2. The New Android Project dialog box.

4. Select the Android 2.2 build target, and click Next (see Figure 3).

Note: Choosing the Android 2.2 build target will configure the compiler to target the Android 2.2 SDK, and will ensure that your PhoneGap application will work on devices running Android 2.2 and newer versions of the operating system.

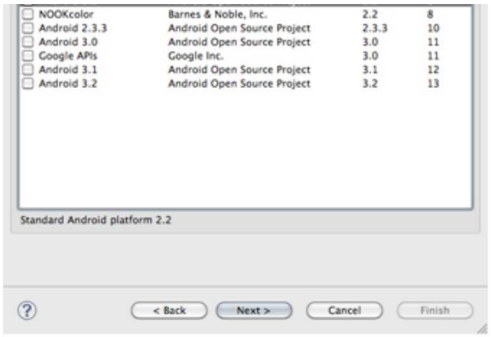


Figure 3. Selecting a build target

5. On the Application Info screen, type a package name for your main Android application (see Figure 4). This should be a namespace that logically represents your package structure; for example, **com.yourcompany.yourproject**.
6. Click Finish.

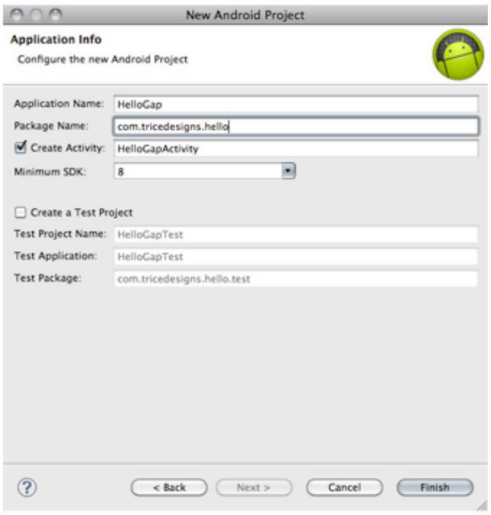


Figure 4. Specifying a package name.

Configure the project to use PhoneGap

At this point, Eclipse has created an empty Android project. However, it has not yet been configured to use PhoneGap. You'll do that next.

1. Create an assets/www directory and a libs directory inside of the new Android project. All of the HTML and JavaScript for your PhoneGap application interface will reside within the assets/www folder (see Figure 5).

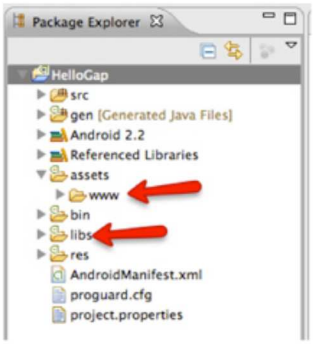
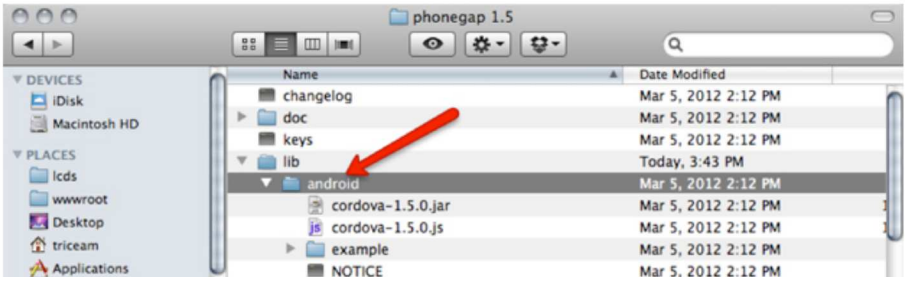


Figure 5. New project directories.

2. To copy the required files for PhoneGap into the project, first locate the directory where you downloaded PhoneGap, and navigate to the lib/android subdirectory (see Figure 6).



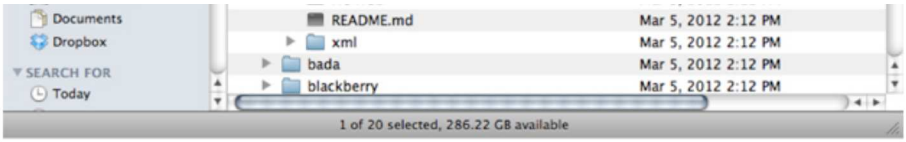


Figure 6. The PhoneGap lib/android directory.

3. Copy cordova-1.5.0.js to the assets/www directory within your Android project.
4. Copy cordova-1.5.0.jar to the libs directory within your Android project.
5. Copy the xml directory into the res directory within your Android project (see Figure 7).

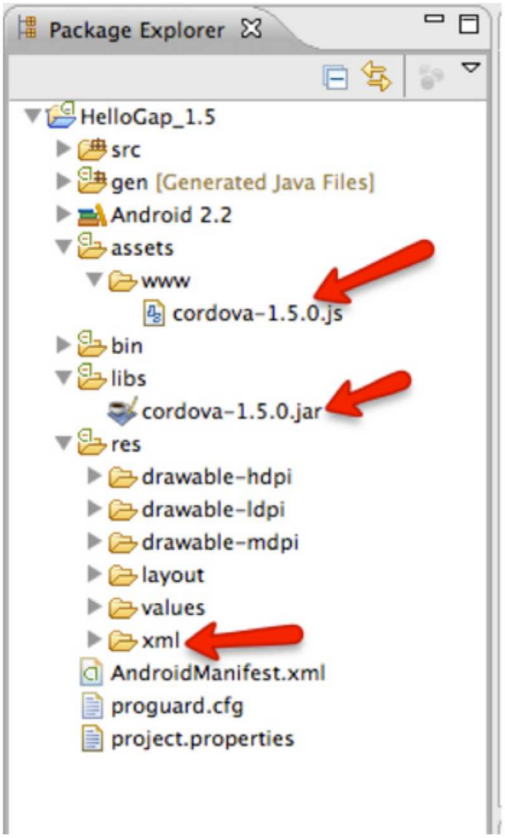


Figure 7. Copied resources.

6. Next, create a file named index.html in the assets/www folder. This file will be used as the main entry point for your PhoneGap application's interface.
7. In index.html, add the following HTML code to act as a starting point for your user interface development:

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>PhoneGap</title>
  <script type="text/javascript" charset="utf-8" src="cordova-1.5.0.js"></scrip
 </head>
 <body>
    <h1>Hello PhoneGap</h1>
 </body>
</html>
```

8. You will need to add the cordova-1.5.0.jar library to the build path for the Android project. Right-click cordova-1.5.0.jar and select Build Path > Add To Build Path (see Figure 8).

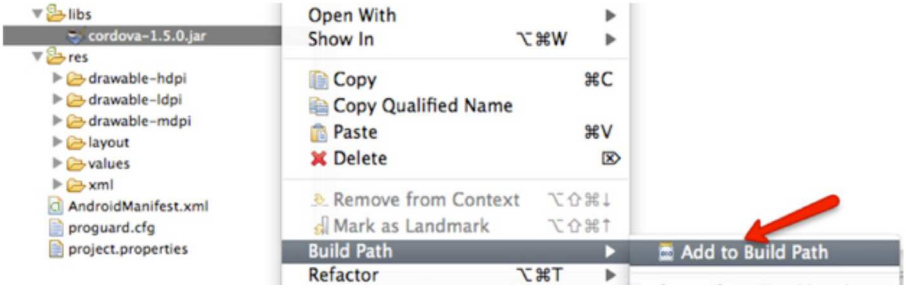


Figure 8. Adding cordova-1.5.0.jar to the build path.

Update the Activity class

Now you are ready to update the Android project to start using PhoneGap.

1. Open your main application Activity file. This file will have the same name as your project, followed by the word "Activity". It will be located under the src folder in the project package that you specified earlier in this process.

For my project, which I named HelloGap, the main Android Activity file is named HelloGapActivity.java, and is located in the package com.tricedesigns.hello, which I specified in the New Android Project dialog box.

2. In the main Activity class, add an import statement for `org.apache.cordova.DroidGap`:

```
import org.apache.cordova.DroidGap;
```

3. Change the base class from `Activity` to `DroidGap` ; this is in the class definition following the word `extends` :

```
public class HelloGapActivity extends DroidGap {
```

4. Replace the call to `setContentView()` with a reference to load the PhoneGap interface from the local assets/www/index.html file, which you created earlier (see Figure 9).

```
super.loadUrl("file:///android_asset/www/index.html");
```

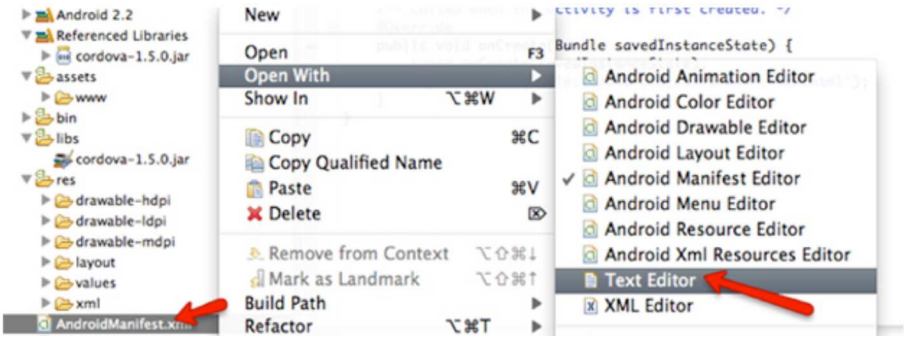
Note: In PhoneGap projects, you can reference files located in the assets directory with a URL reference file:///android_asset, followed by the path name to the file. The file:///android_asset URI maps to the assets directory.

Figure 9. Updates to the main Activity class.

Configure the project metadata

You have now configured the files within your Android project to use PhoneGap. The last step is to configure the project metadata to enable PhoneGap to run.

1. Begin by opening the AndroidManifest.xml file in your project root. Use the Eclipse text editor by right-clicking the AndroidManifest.xml file and selecting Open With > Text Editor (see Figure 10).



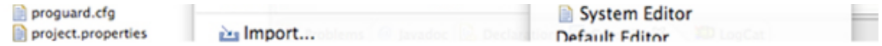


Figure 10. Opening AndroidManifest.xml.

2. In AndroidManifest.xml, add the following `supports-screen` XML node as a child of the root `manifest` node:

```
<supports-screens
    android:largeScreens="true"
    android:normalScreens="true"
    android:smallScreens="true"
    android:resizeable="true"
    android:anyDensity="true"
    />
```

The `supports-screen` XML node identifies the screen sizes that are supported by your application. You can change screen and form factor support by altering the contents of this entry. To read more about `<supports-screens>`, visit the Android developer topic on the supports-screen element.

Next, you need to configure permissions for the PhoneGap application.

3. Copy the following `<uses-permission>` XML nodes and paste them as children of the root `<manifest>` node in the AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMAND"
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
```

The `<uses-permission>` XML values identify the features that you want to be enabled for your application. The lines above enable all permissions required for all features of PhoneGap to function. After you have built your application, you may want to remove any permissions that you are not actually using; this will remove security warnings during application installation. To read more about Android permissions and the `<uses-permission>` element, visit the Android developer topic on the uses-permission element..

After you have configured application permissions, you need to modify the existing `<activity>` node.

4. Locate the `<activity>` node, which is a child of the `<application>` XML node. Add the following attribute to the `<activity>` node:

```
android:configChanges="orientation|keyboardHidden"
```

5. Next, you need to create a second `<activity>` node for the `org.apache.cordova.DroidGap` class. Add the following `<activity>` node as a sibling of the existing `<activity>` XML node:

```
<activity
    android:name="org.apache.cordova.DroidGap"
    android:label="@string/app_name"
    android:configChanges="orientation|keyboardHidden">
    <intent-filter></intent-filter>
</activity>
```

At this point, your project is configured to run as a PhoneGap project for Android. If you run into any issues, verify your configuration against the example provided at the PhoneGap getting started site for Android.

Running the application

To launch your PhoneGap application in the Android emulator, right-click the project root, and select Run As > Android Application (see Figure 11).

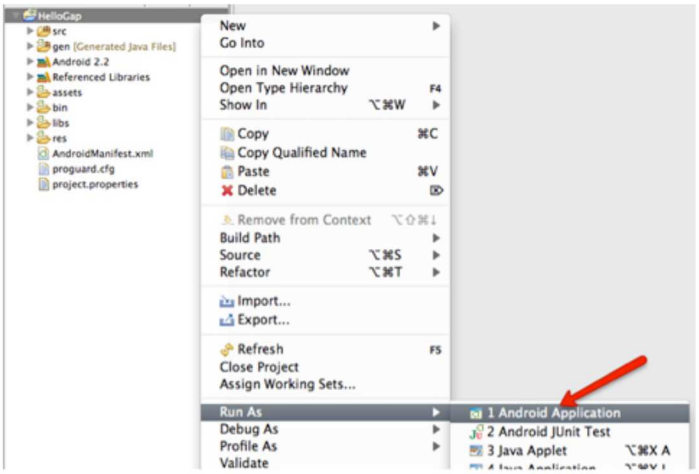


Figure 11. Launching the Android application.

If you don't have any Android virtual devices set up, you will be prompted to configure one. To learn more about configuring Android emulator virtual devices, visit the Android developer guide for devices.

Eclipse will automatically start an Android emulator instance (if one is not already running), deploy your application to the emulator, and launch the application (see Figure 12).

Figure 12. The application in the Android emulator.

After you get your application running in the Android emulator, you'll want to test it out on a physical device. I strongly recommend that you always test your applications on a physical device before deploying the application into production environments. Physical devices always have different computing abilities and form factors than emulators, and device testing can uncover issues that may not have been detected in the emulator environment.

Follow these steps to launch your application on a physical Android device:

1. Make sure the device is connected to your computer via USB.
2. Choose Run > Run Configurations (see Figure 13).

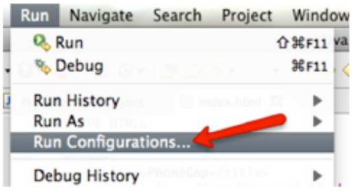


Figure 13. Updating run configurations.

3. Select your application under Android Application on the left side of the Run Configurations dialog box.
4. Click the Target tab, and then select Manual as the Deployment Target Selection Mode.
5. When you are ready to launch your application, click Run (see Figure 14).

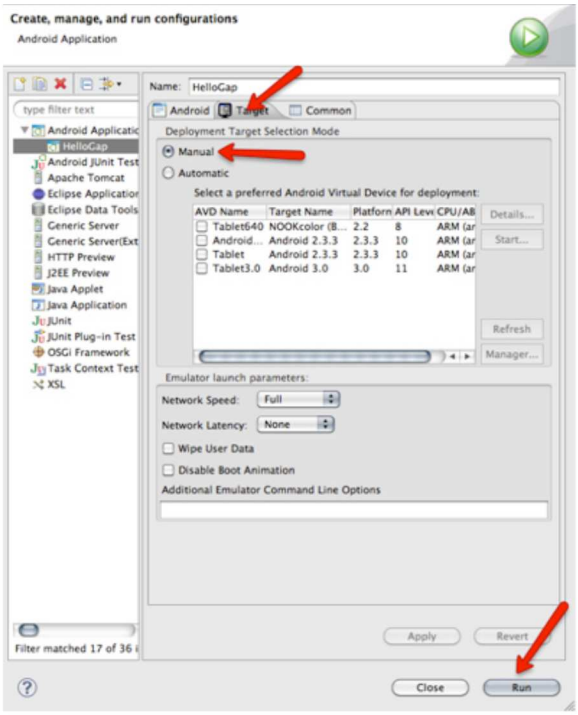


Figure 14. Preparing to run the application on a device.

In the Android Device Chooser dialog box, you can select either an emulator or a connected Android device. All connected Android devices will be displayed in this list.

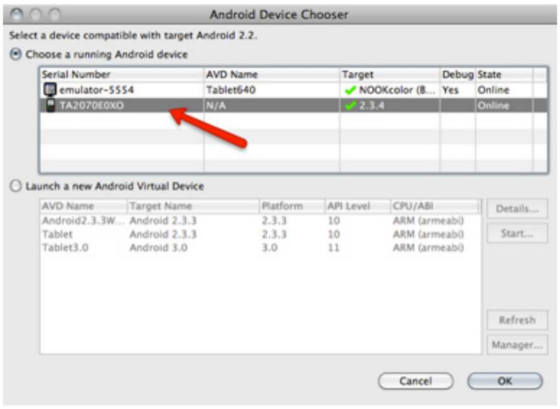


Figure 15. Choosing an Android device.

6. Select the device that you want to use (see Figure 15), and click OK.

Your PhoneGap application will be installed and launched on the device.

Where to go from here

If you've made it this far, you are ready to start building real applications for Android using PhoneGap. Next, you may want to read Extending PhoneGap with native plugins for Android.

Remember, PhoneGap applications are built using HTML, CSS, and JavaScript for the user interface. This enables you to create great looking applications with ease using traditional web development skills. To learn more about PhoneGap, check out the PhoneGap wiki, join the PhoneGap Google Group, or explore the PhoneGap documentation.

More Like This

Using CSS3 transitions: A comprehensive guide
The pursuit of simplicity
An Overview of Brackets' Code Architecture
Backbone.js Wine Cellar tutorial – Part 1: Getting started
JavaScript design patterns – Part 1: Singleton, composite, and façade
Using the Geolocation API

Unit test JavaScript applications with Jasmine

Backbone.js Wine Cellar tutorial – Part 2: CRUD

Build a Hangman game with HTML5 Canvas, JavaScript, and CSS – Part 1: Creating the interface

Introduction to web typography and @font-face