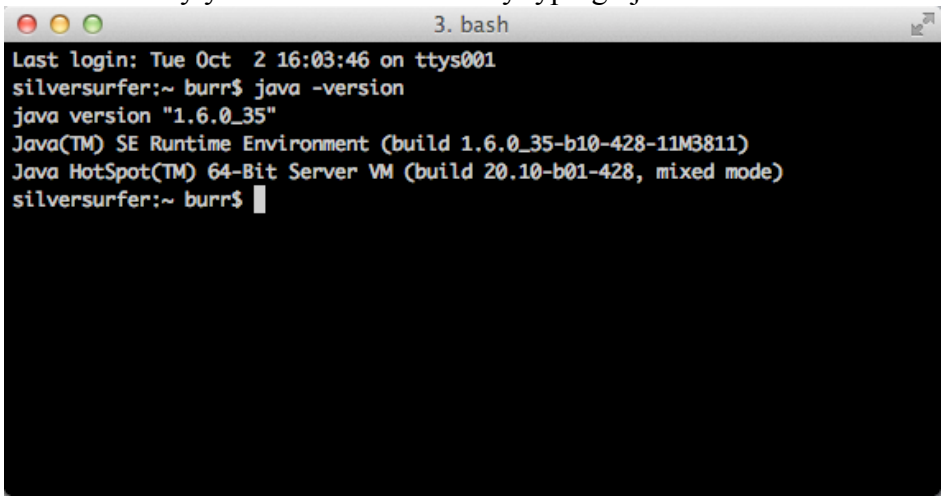
Apache Cordova/Phonegap + Android Development in JBoss Developer Studio 5

Note: Android is not yet a tested plug-in in JBDS 5, it will be for JBDS 6

By the end of this tutorial, you will have:
a) Setup your development environment
b) Created your first Phonegap/Apache Cordova-based Android Application
c) Run the app on the Android emulator
d) Run the app on a real Android device (if you have one handy)
e) Leverage the Apache Cordova/Phonegap API for native device functionality
f) Connected your app "to the cloud" with a REST endpoint

0) First make sure that you have a JDK installed on your machine. This document was created and tested with Java SE 6 - JDK
http://www.oracle.com/technetwork/java/javase/downloads/index.html

You can verify your Java installation by typing "java -version" at a command prompt.

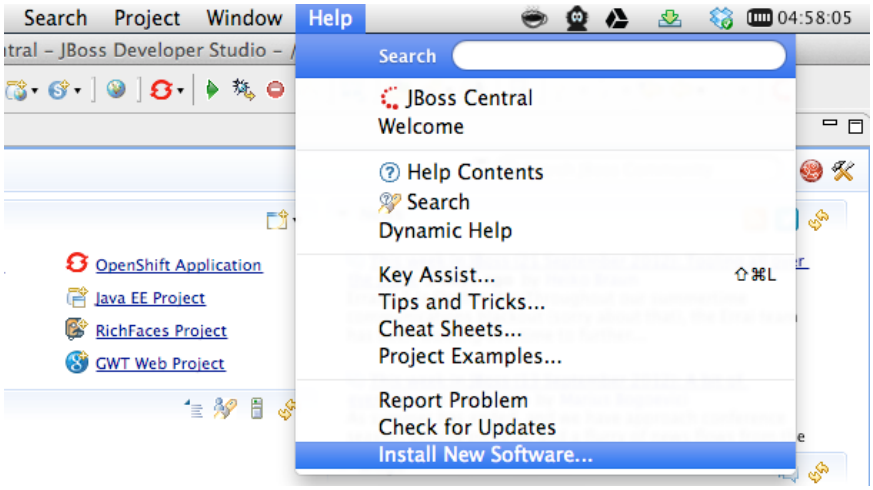


1) Download & Install JBDS
Note: Mac is not required for Android development but it is for iOS (iPod, iPhone or iPad) development.
https://devstudio.jboss.com/earlyaccess/

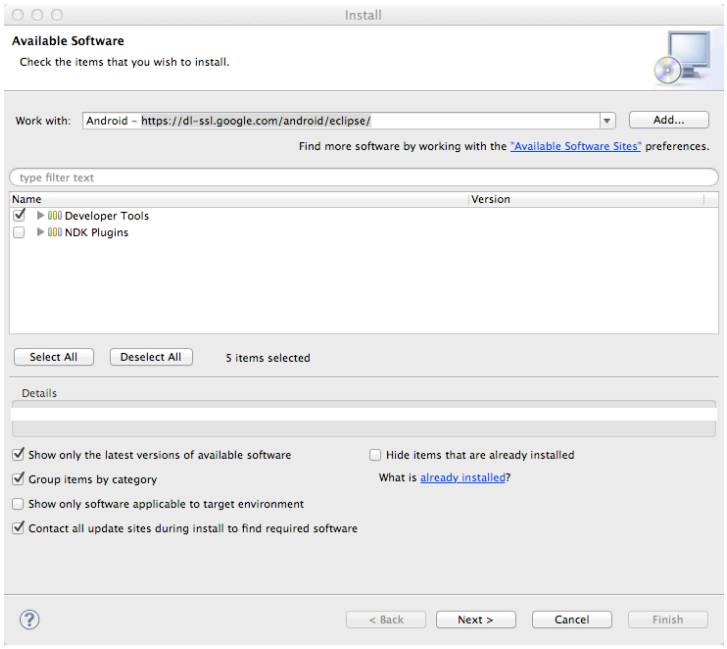
2) Get the Android SDK
http://developer.android.com/sdk/index.html
Download, unzip into a directory that you will remember. I have dropped my Android SDK into /Users/burr/android-sdks

Now install the Android (ADT) Plug-Ins into JBDS/Eclipse

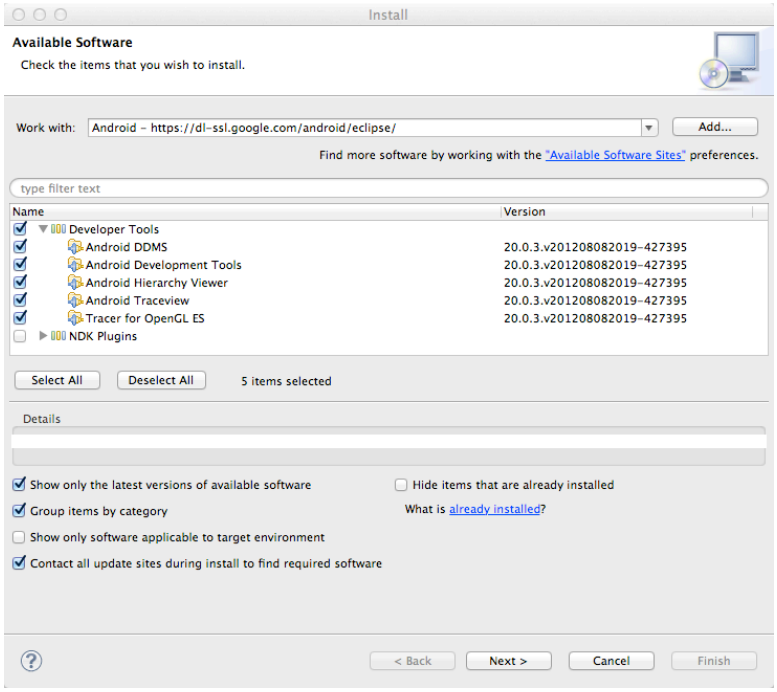
3) Help – Install New Software



https://dl-ssl.google.com/android/eclipse/



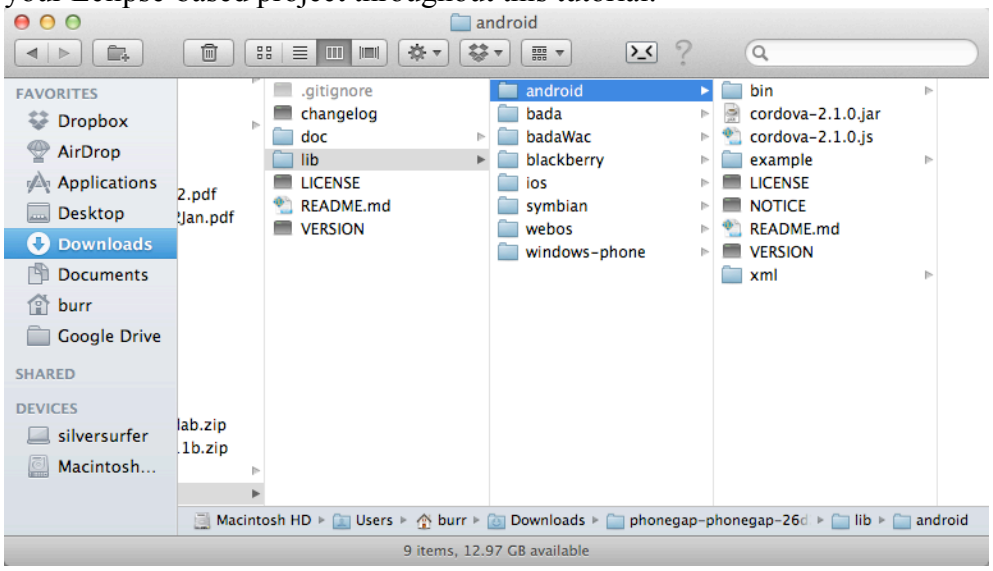
Check Developer Tools



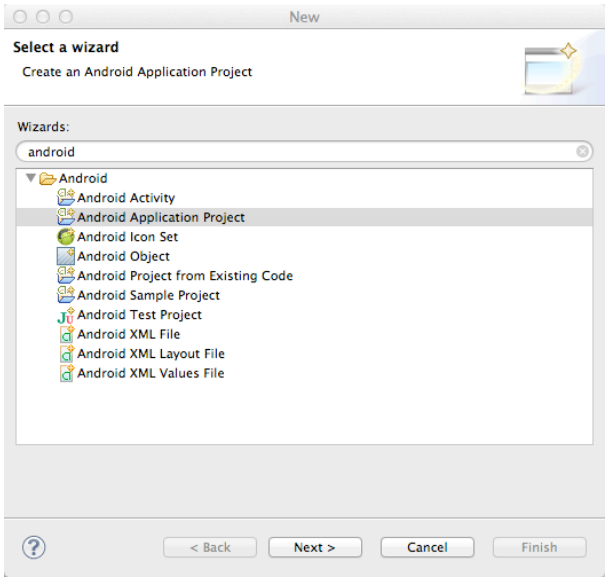
You will be prompted to restart Eclipse, please do so.
When JBDS/Eclipse restarts you should notice at least two new icons in your toolbar -

the leftmost one is the Android SDK Manager. Click on the SDK Manager to see if it launches correctly, make sure that your Android OS version, for your real-world device is listed as **"**Installed**"**. This will be covered again later in this tutorial.

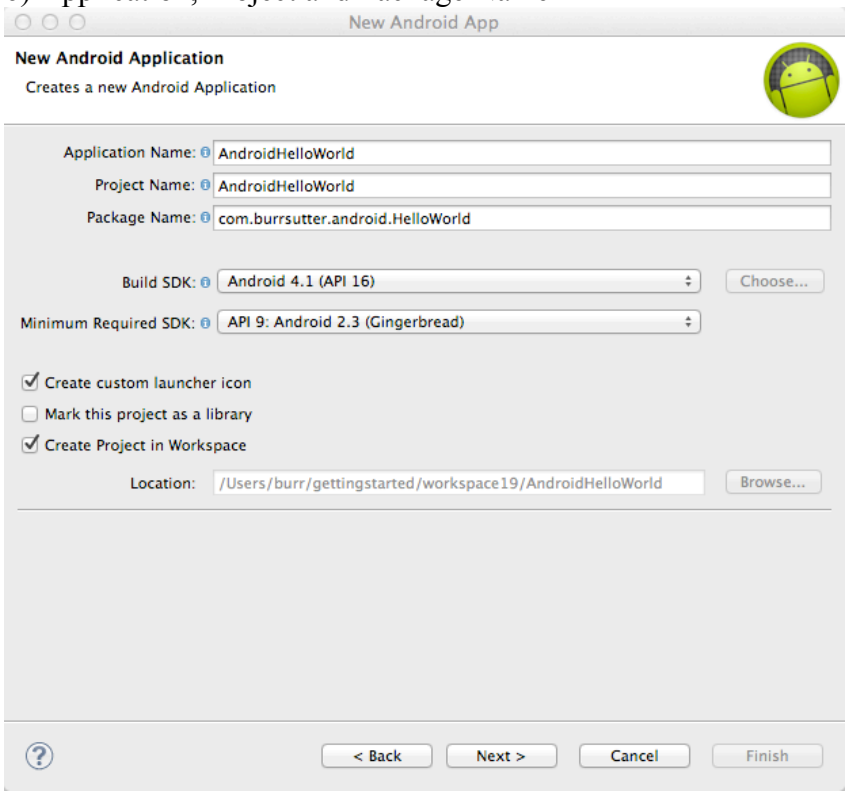
4) Download and Unzip PhoneGap
http://phonegap.com/download
Make sure to remember where you unzipped it - you will be pulling several files into your Eclipse-based project throughout this tutorial.



5) Create a new Android Project: File – New – Other (or Cntrl/Command N)



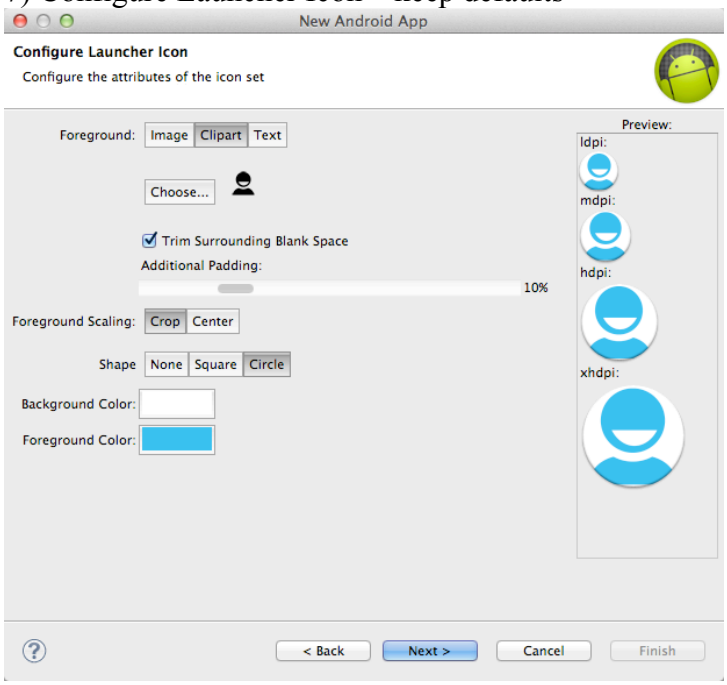
6) Application, Project and Package Name



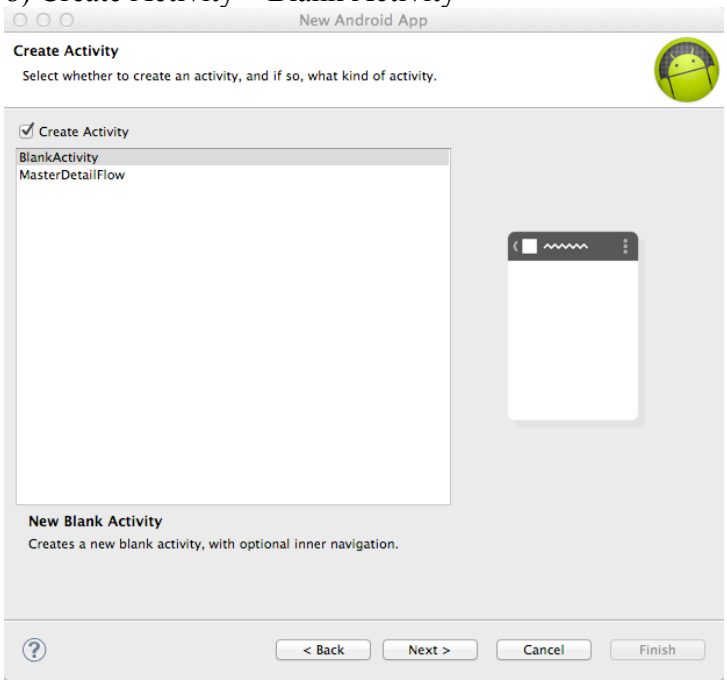
Note: Build SDK and Minimum Required SDK should be set based on your actual Android phone - the physical device. In addition, these drop boxes are populated based on the Android SDK Manager - "installed" options. If your real-world device Android OS is not listed here, simply take the defaults for now. It is possible to switch these

settings back to your real target platform, however, it might be best for you to start your project over again - just for the learning experience.

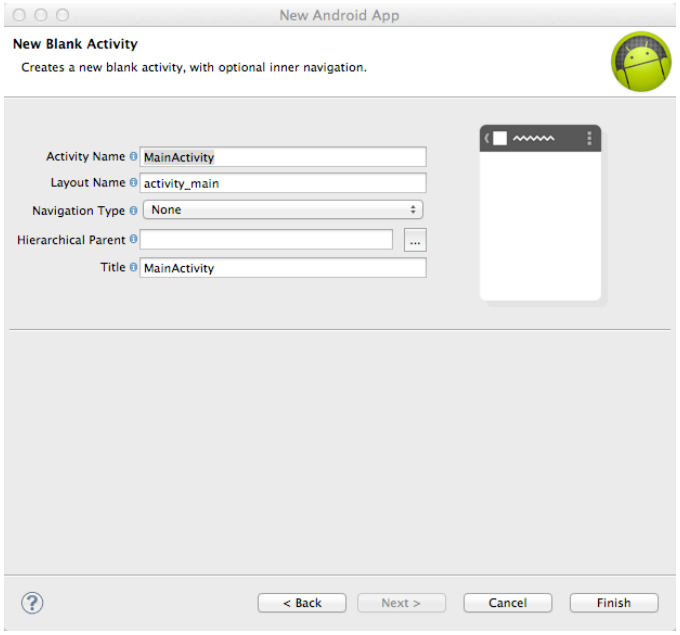
7) Configure Launcher Icon – keep defaults



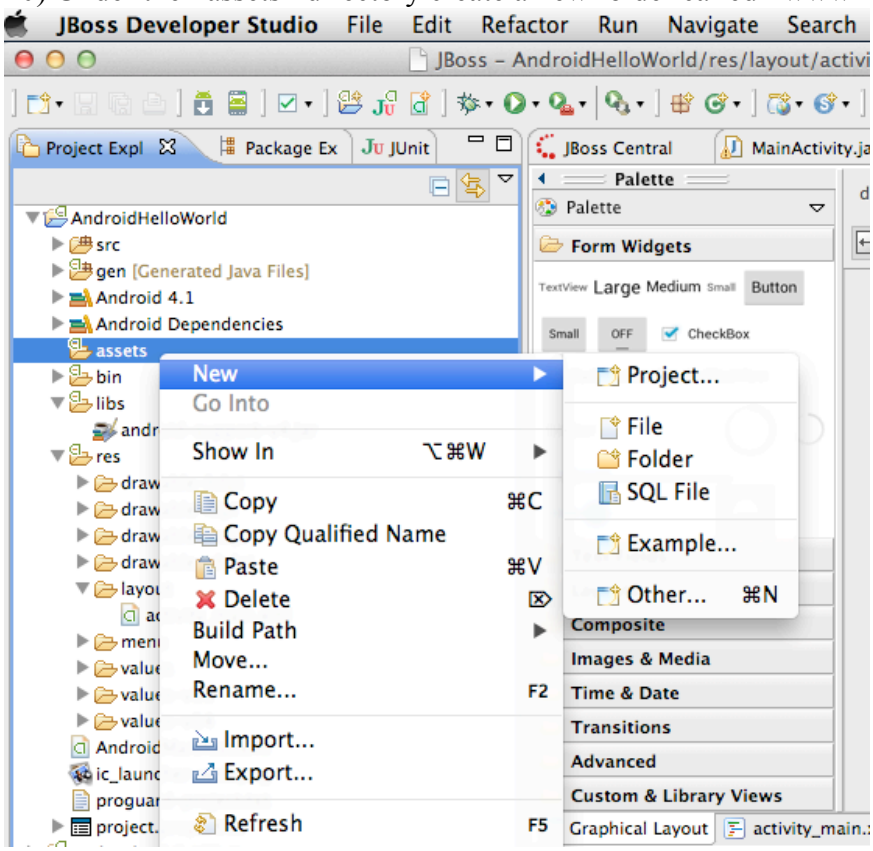
8) Create Activity – Blank Activity

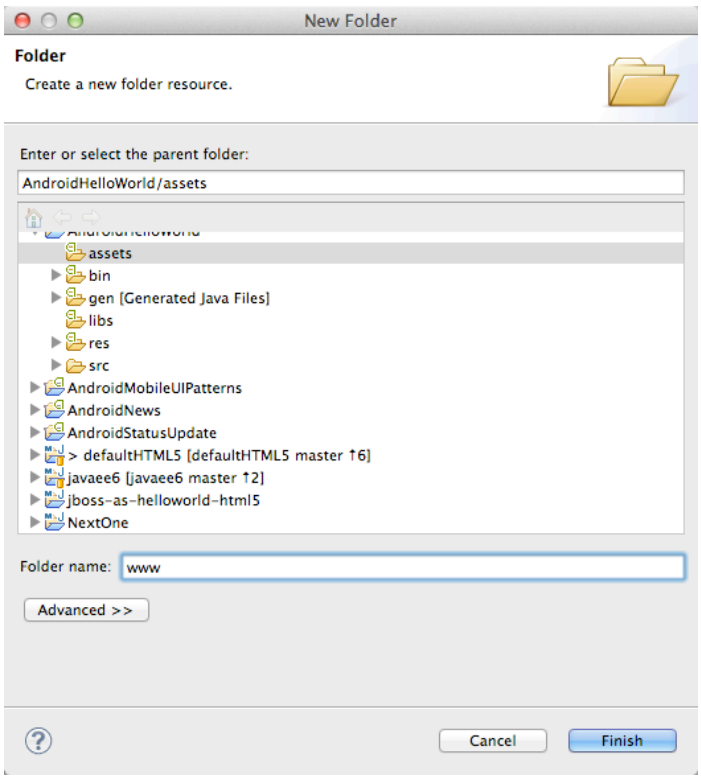


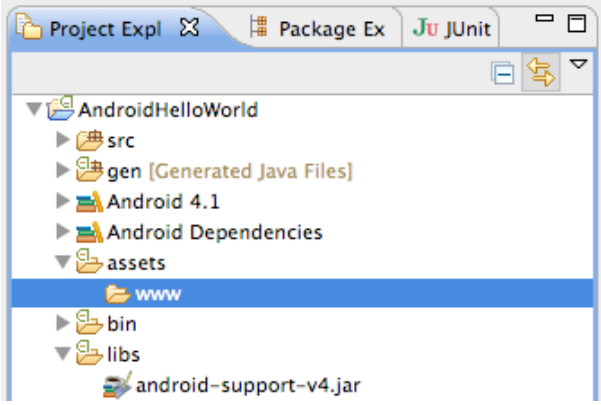
9) Defaults for the new Activity – just hit Finish



10) Under the "assets" directory create a new folder called "www"

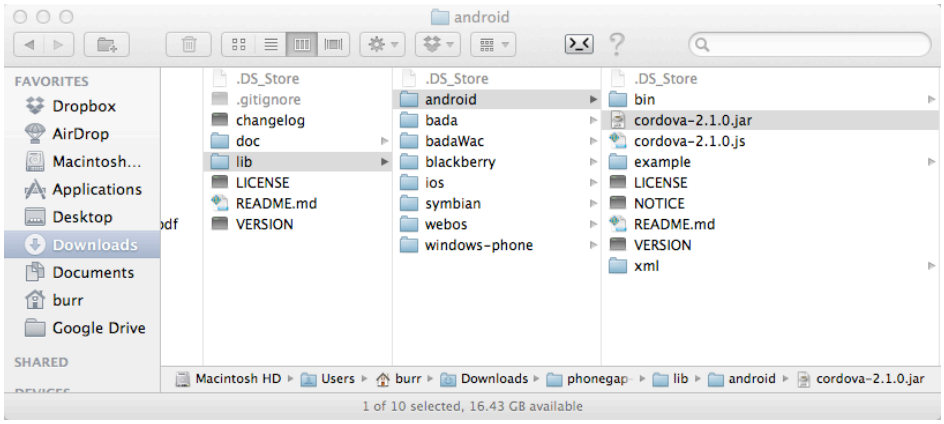


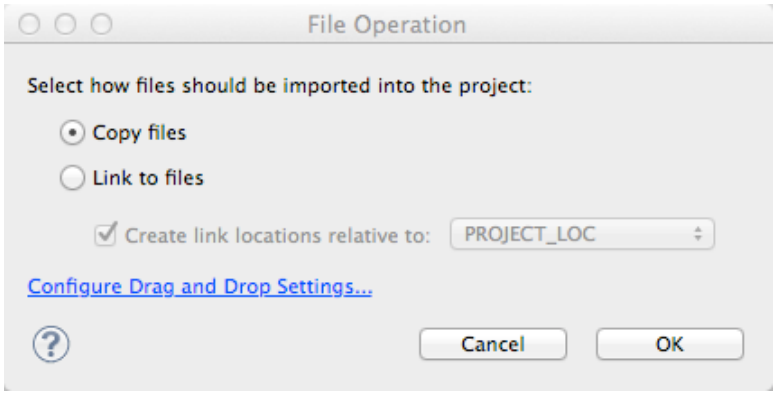


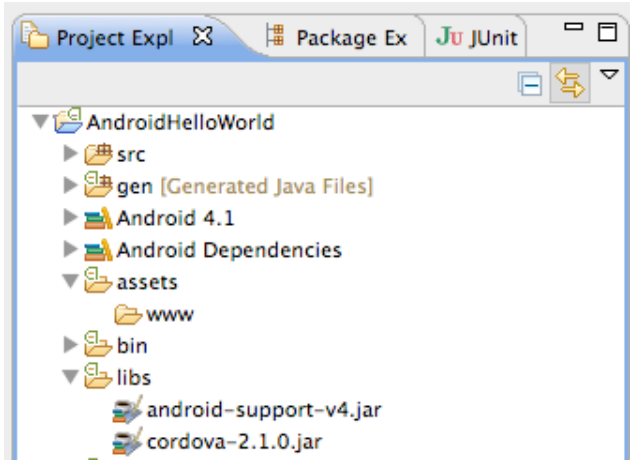


Think of **"**www**"** as the equivalent of **"**webapp**"** in a Maven project.

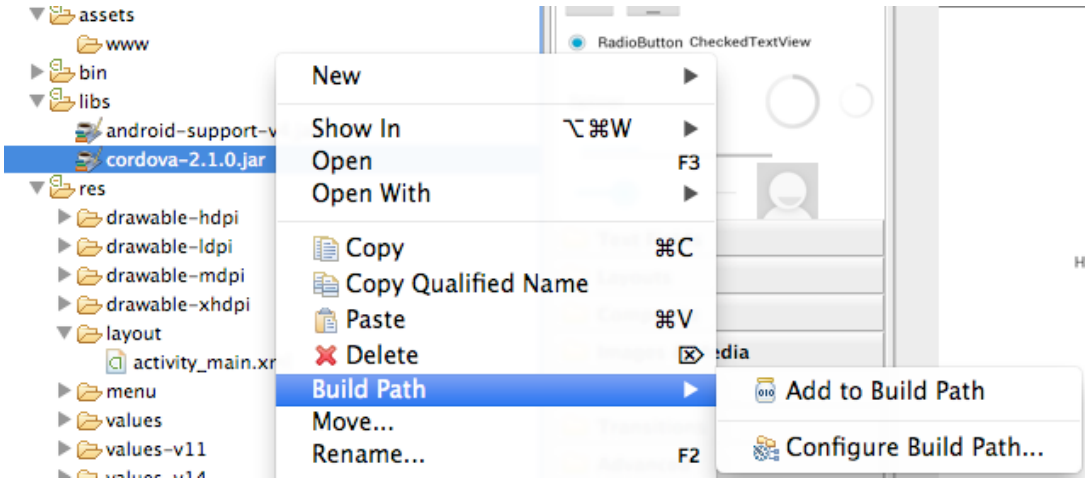
11) Add Cordova-2.1.0.jar to under libs – you can just drag & drop into Eclipse/JBDS
And when prompted, Copy files and select OK



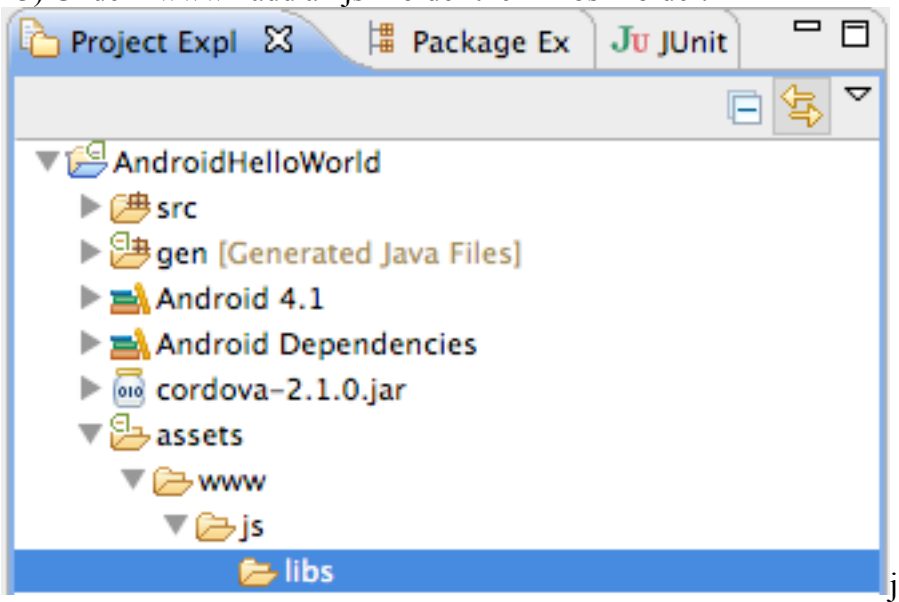




12) Right-click on cordova-2.1.0.jar and select Build Path**...**Add to Build Path



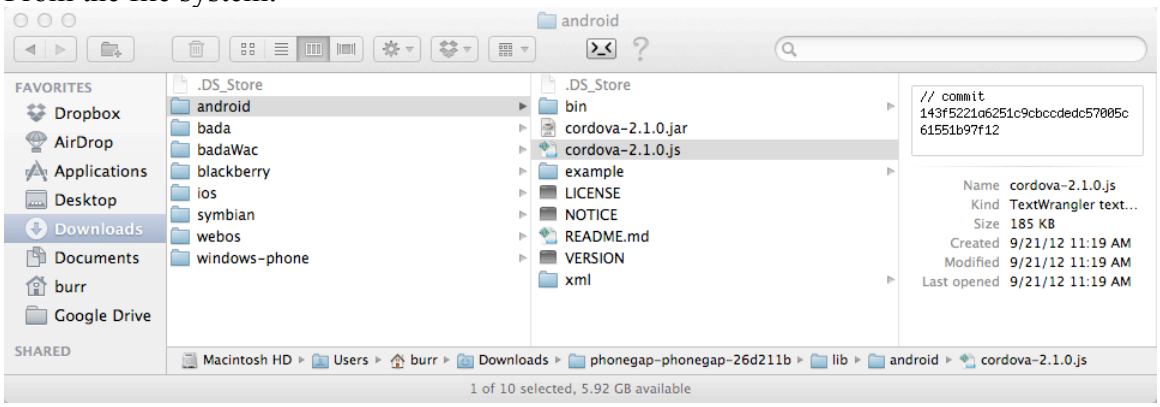
13) Under "www" add a "js" folder then "libs" folder.

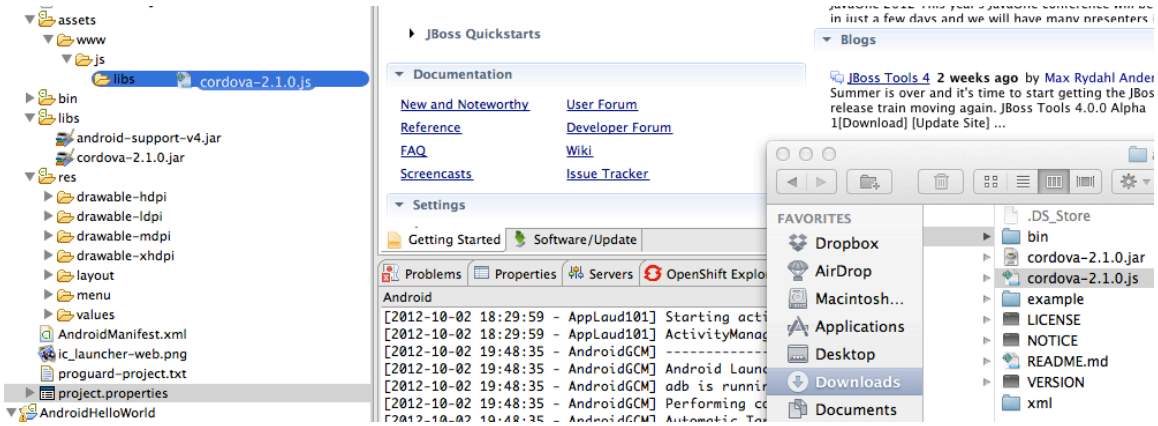


js

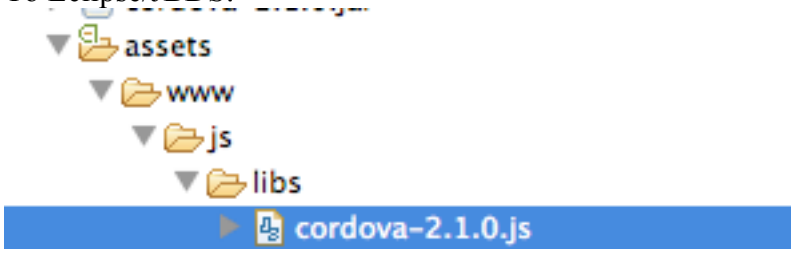
So you can add cordova-2.1.0.js to libs by drag & drop

From the file system:

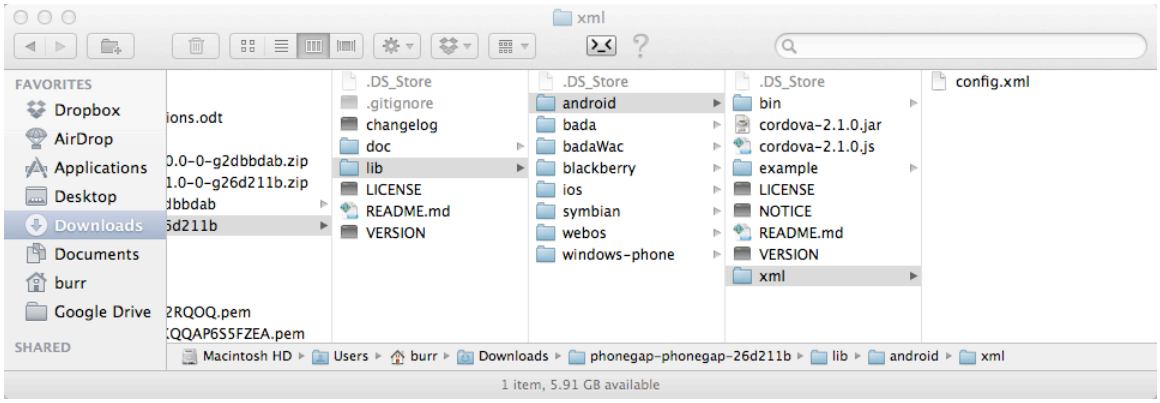


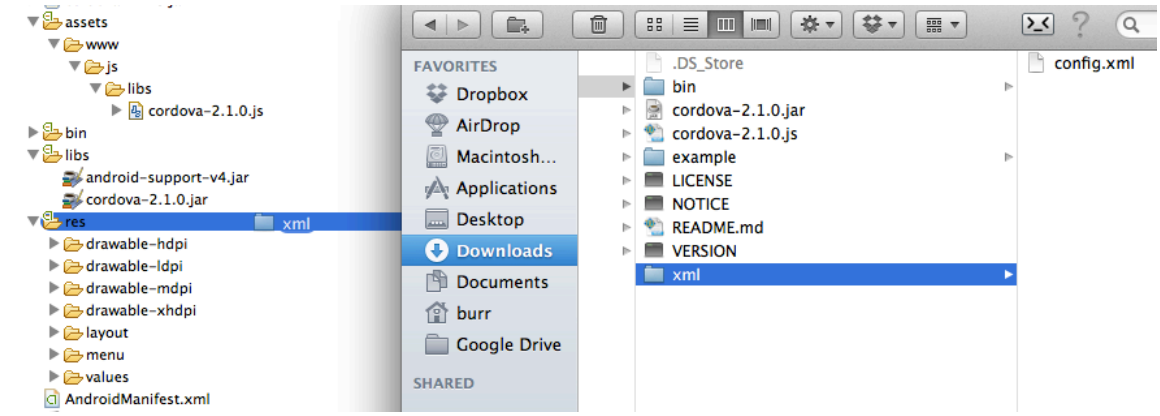


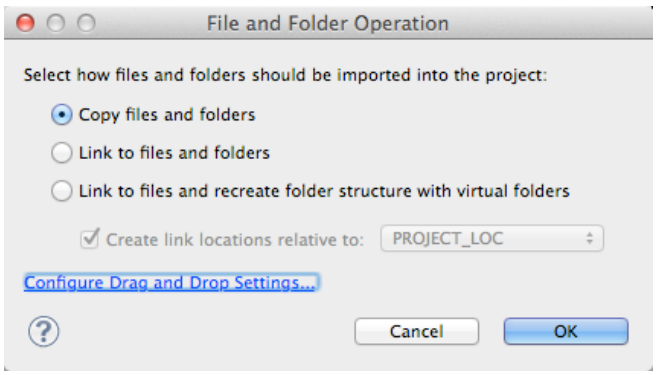
To Eclipse/JBDS:



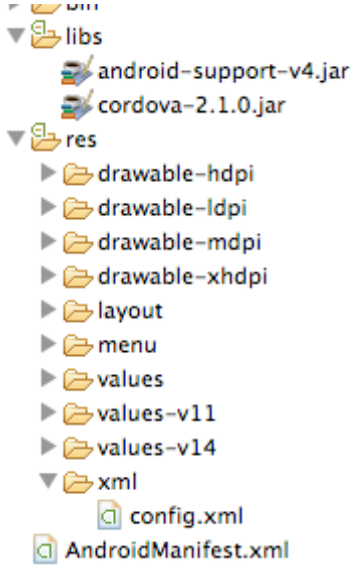
14) Also, bring in the XML folder with its config.xml into **"**res**"**
From:



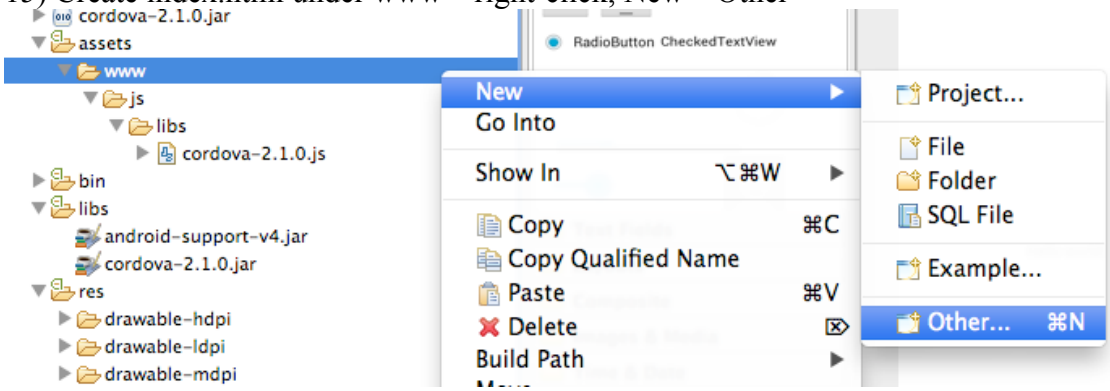




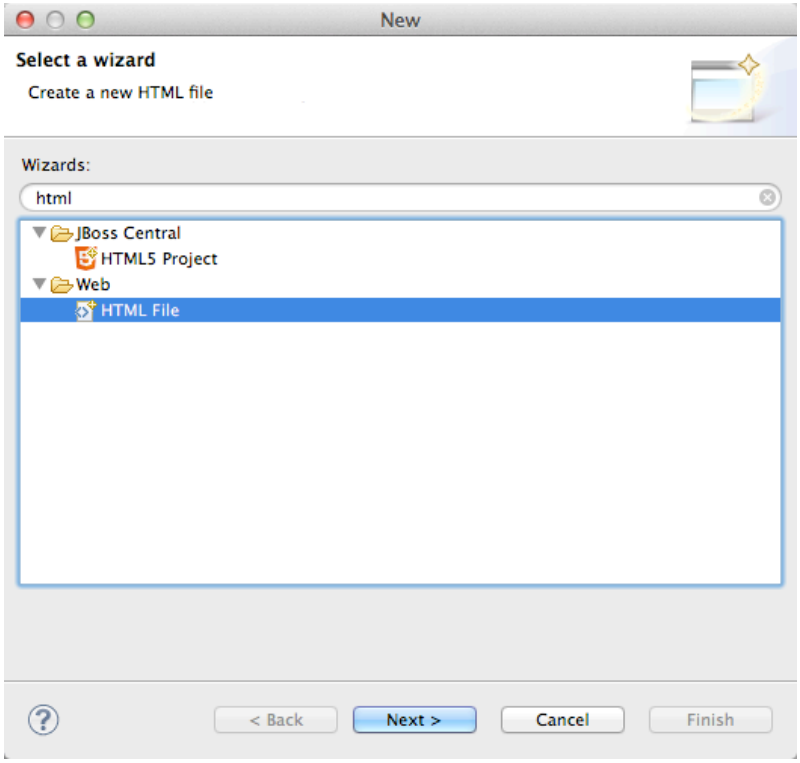
To:



15) Create index.html under www – right-click, New – Other

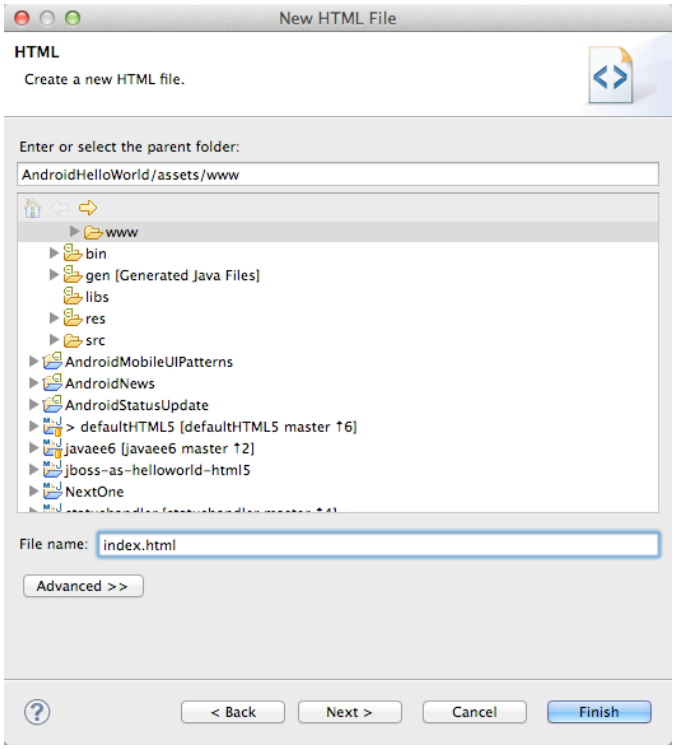


Select "HTML File"



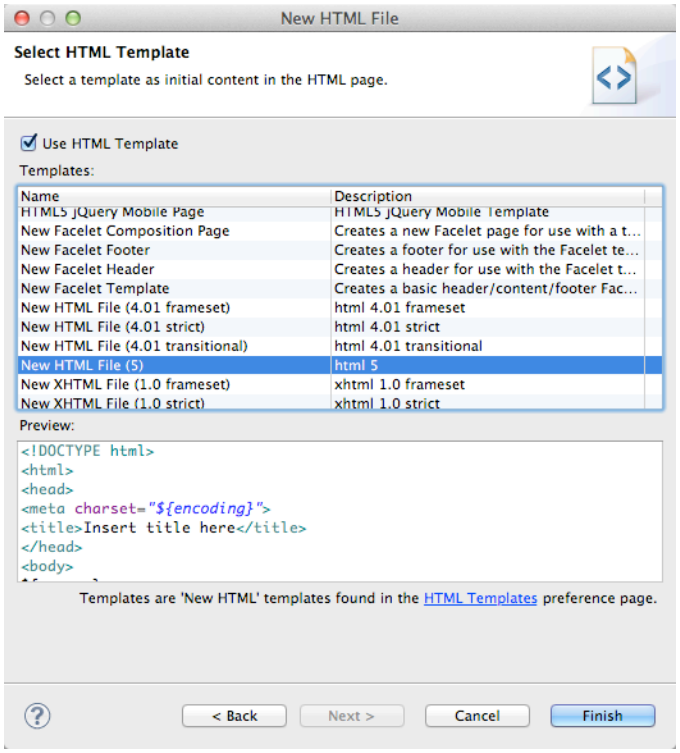
Select Next

Name it “index.html”



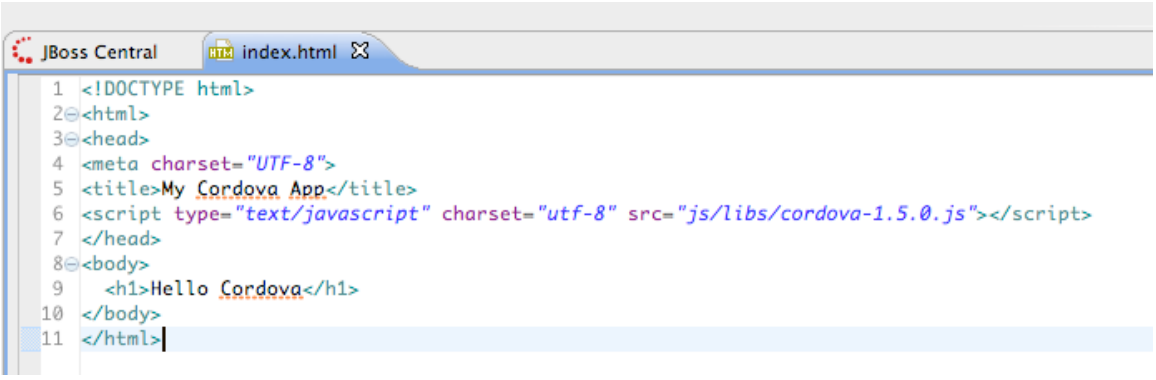
Select Next

Select the HTML 5 Template

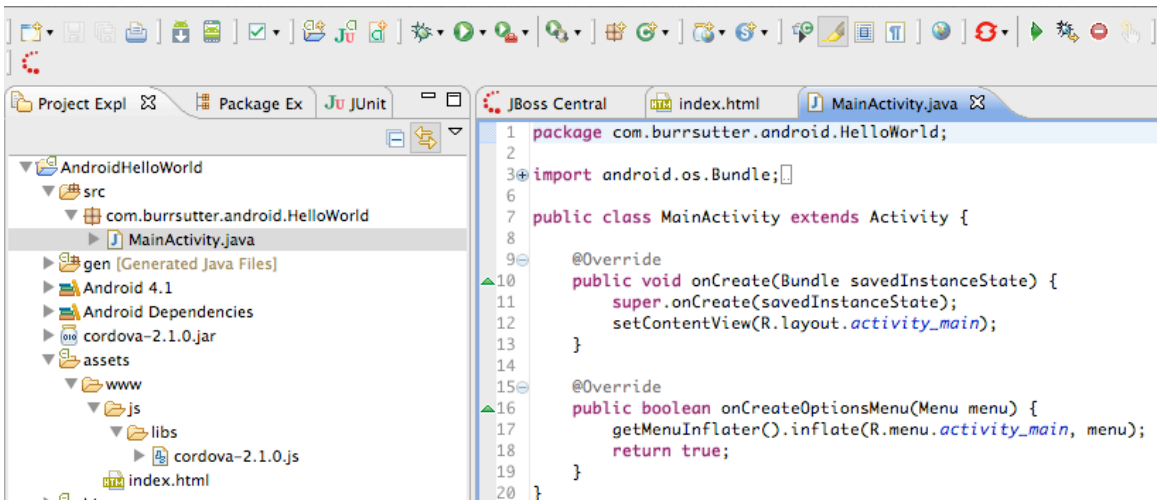


Select Finish

16) Add a reference to cordova-2.1.0.js into the HEAD section of the HTML
<script type="text/javascript" charset="utf-8" src="js/libs/cordova-2.1.0.js"></script>
and a hello message in the BODY
<h1>Hello Cordova</h1>

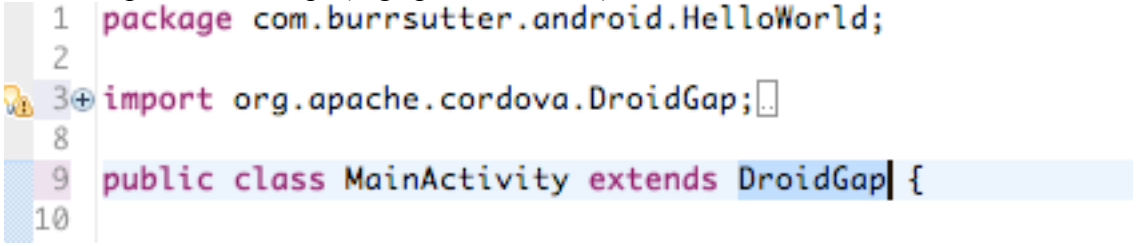


17) Edit MainActivity.java



18) Change **"**extends Activity**"** to be **"**extends DroidGap**"**
This will initially give you a ton of red in the editor as DroidGap has not yet been imported – click on the red X beside the public class MainActivity line and you will be prompted to import DroidGap

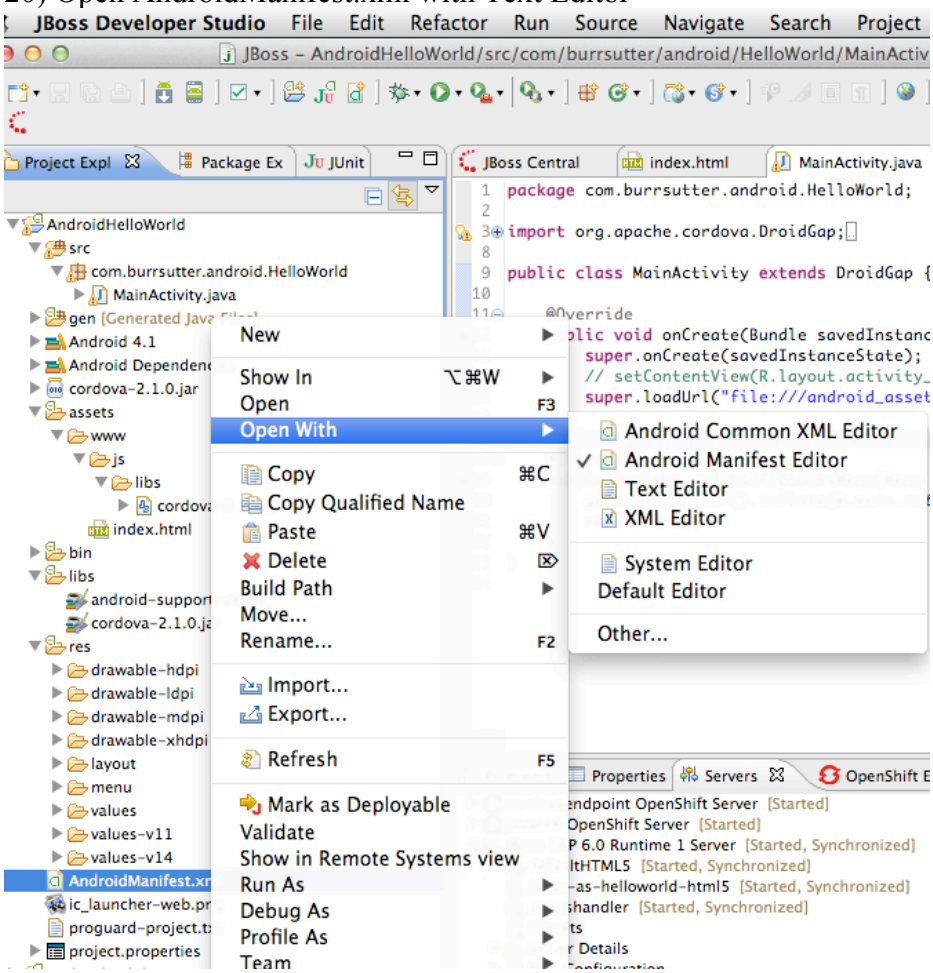
Select Import **'**DroidGap**'** (org.apache.cordova)



19) Comment out **"**setContentView(**...**" and enter the following line:
super.loadUrl("file:///android_asset/www/index.html");

```
JBoss Central | index.html | MainActivity.java

1  package com.burrsutter.android.HelloWorld;
2
3  import org.apache.cordova.DroidGap;
8
9  public class MainActivity extends DroidGap {
10
11     @Override
12     public void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         // setContentView(R.layout.activity_main);
15         super.loadUrl("file:///android_asset/www/index.html");
16     }
17
18     @Override
19     public boolean onCreateOptionsMenu(Menu menu) {
20         getMenuInflater().inflate(R.menu.activity_main, menu);
21         return true;
22     }
23 }
24
```

20) Open AndroidManifest.xml with Text Editor

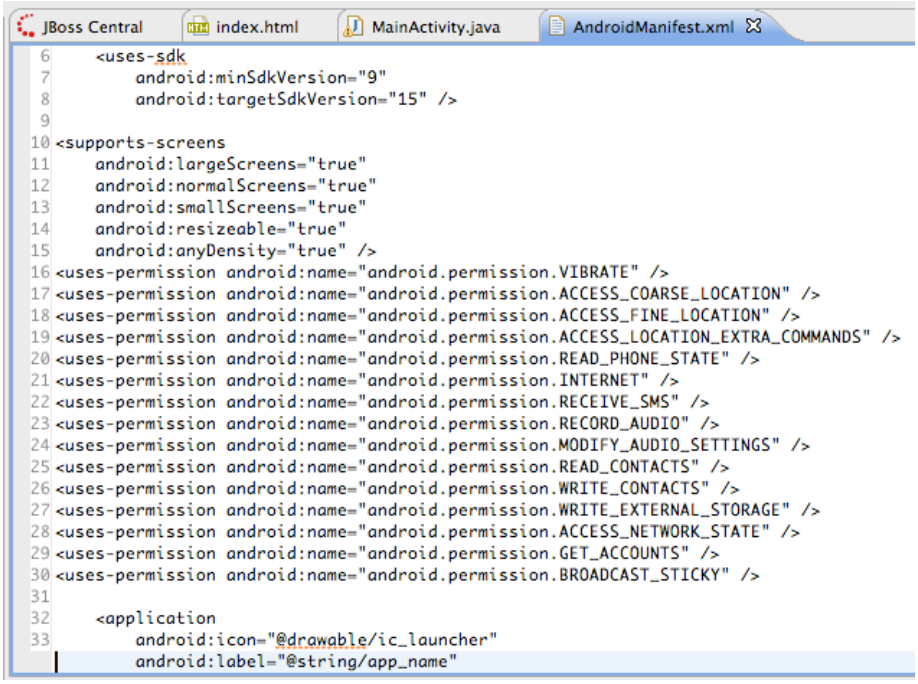


21) Between the **"**uses-sdk**"** and **"**application**"** tags insert the following:

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

And then the following:
<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_COMMANDS" />
<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" />
```

Your current **“**hello world**”** style application does not yet require any of these permissions but it may**…**eventually.

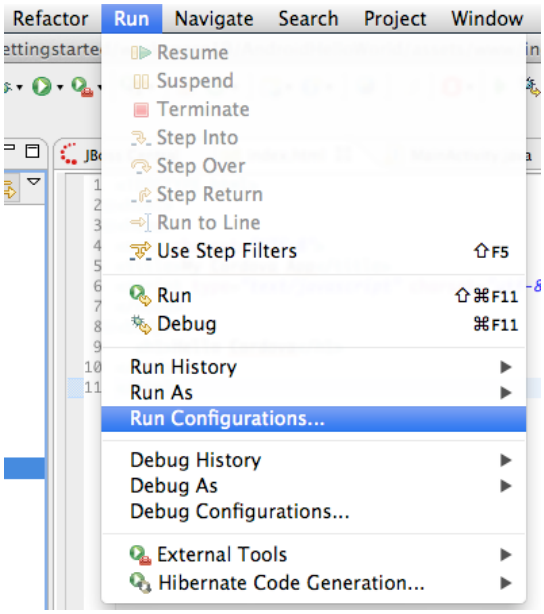
The result should look like the following screenshot



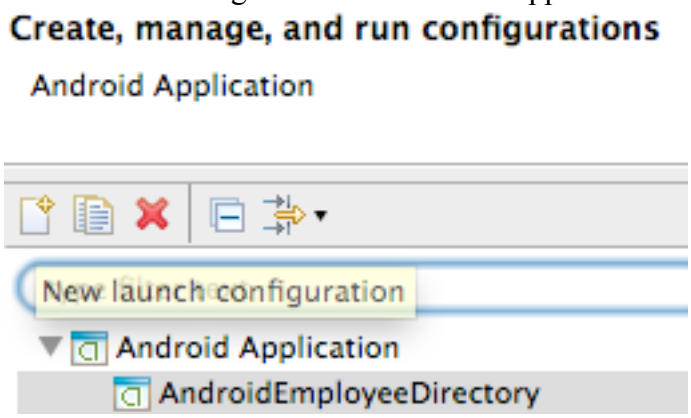
22) Locate the <activity> tag and add the following attribute:

```
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main"
            android:configChanges="orientation|keyboardHidden"  >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

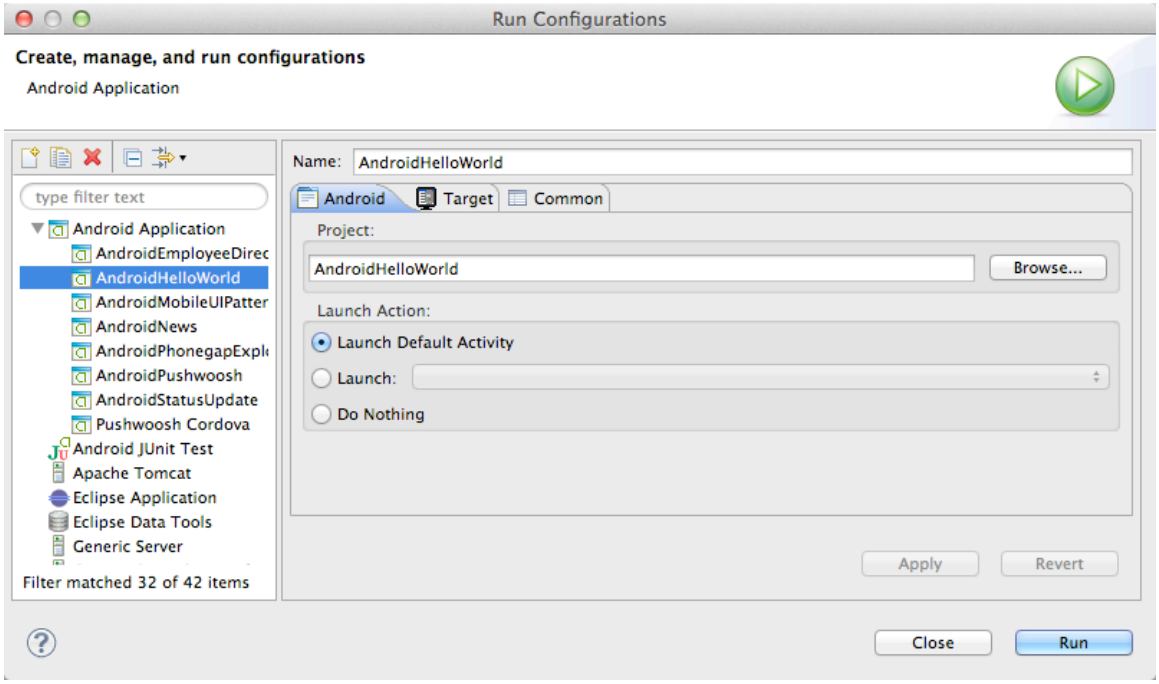
23) Under the JBDS/Eclipse Run menu, select Run Configurations**...**



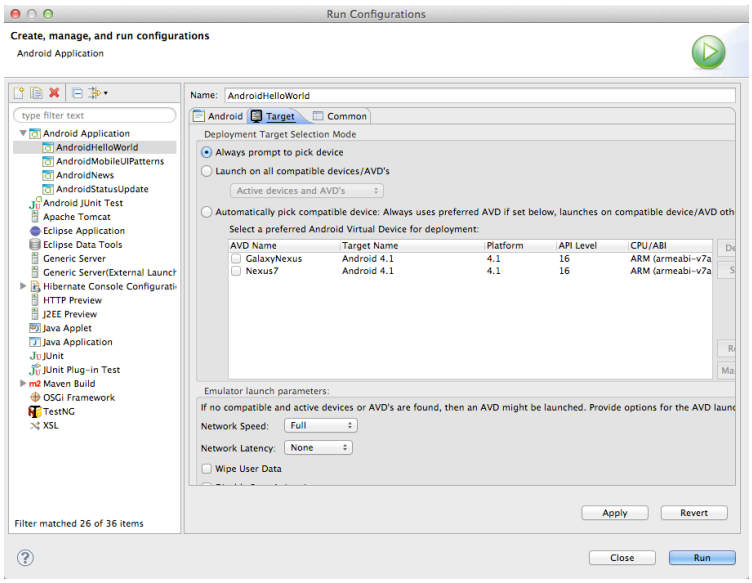
Add a New Configuration for Android Application – AndroidHelloWorld



Select the AndroidHelloWorld project using the Browse button



And on the Target tab, select **"**Always prompt to pick device**"**



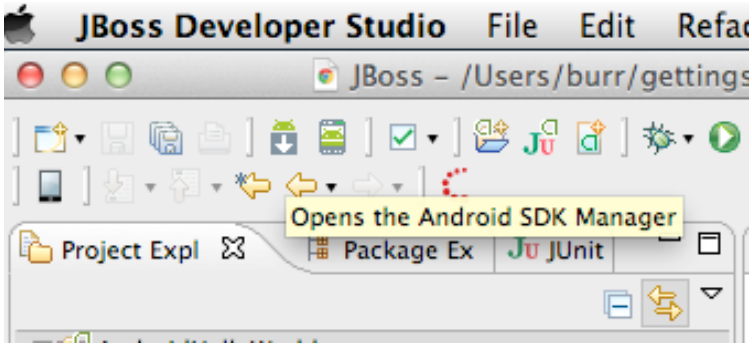
And select Apply

This setup will make it easier for you to switch back and forth between the Android Emulator (AVDs) and a real device plugged in via USB.
Close the Run Configurations Dialog

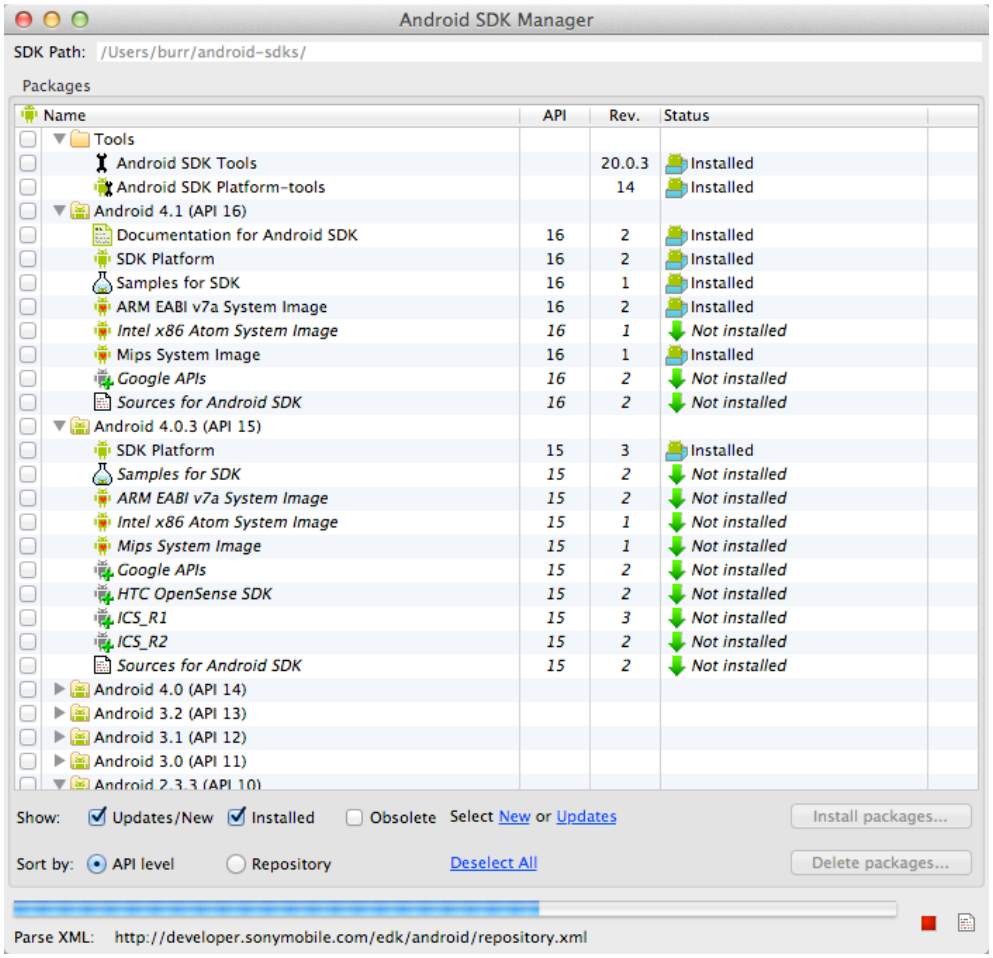
24) iOS and Android both have their own unique ways of setting up to run the simulator/emulator as well as on real devices. In the case of Android, you need to first visit the SDK Manager and install the SDK versions that you wish to work with.

Note: If you originally created the project to require Android 4 and then it may (normally does) complain when you try to deploy it to your Android 2.x real-world phone in your pocket.

There is an icon in the main toolbar to launch the Android SDK Manager – it is the leftmost of the two Android icons.

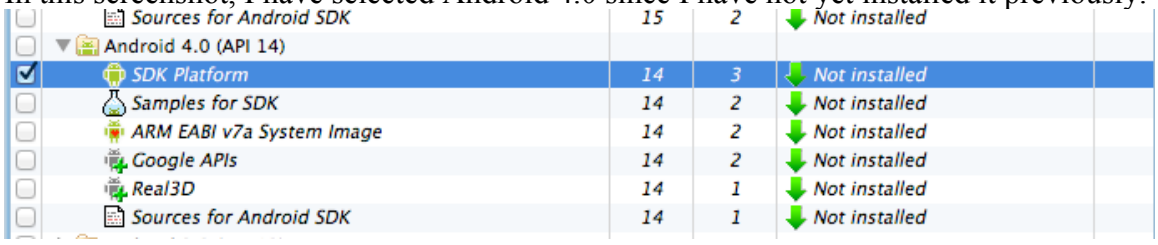


Android SDK Manager

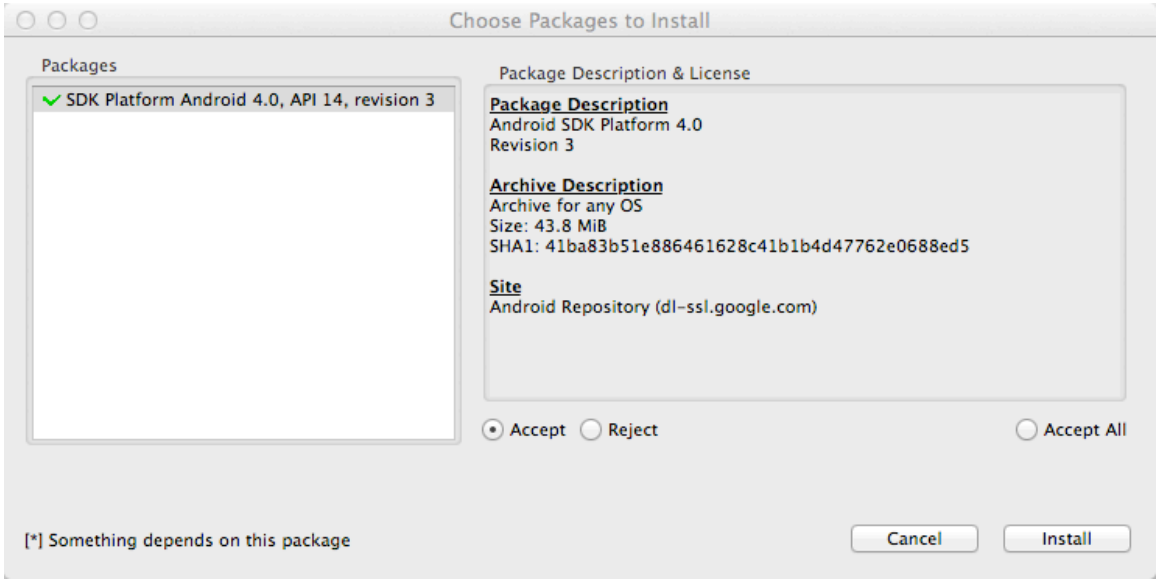


Install Android 2.3.3 (API 10), 4.0.3 (API 15) and 4.1 (API 16) – Cordova works as far as back as Android 2.2 – but our focus is on 2.3, 4.0 and 4.1 (yes, we skipped 3.x altogether).

25) Check the box in the left-most column and select Install 1 package**...** button that becomes enabled.
In this screenshot, I have selected Android 4.0 since I have not yet installed it previously.

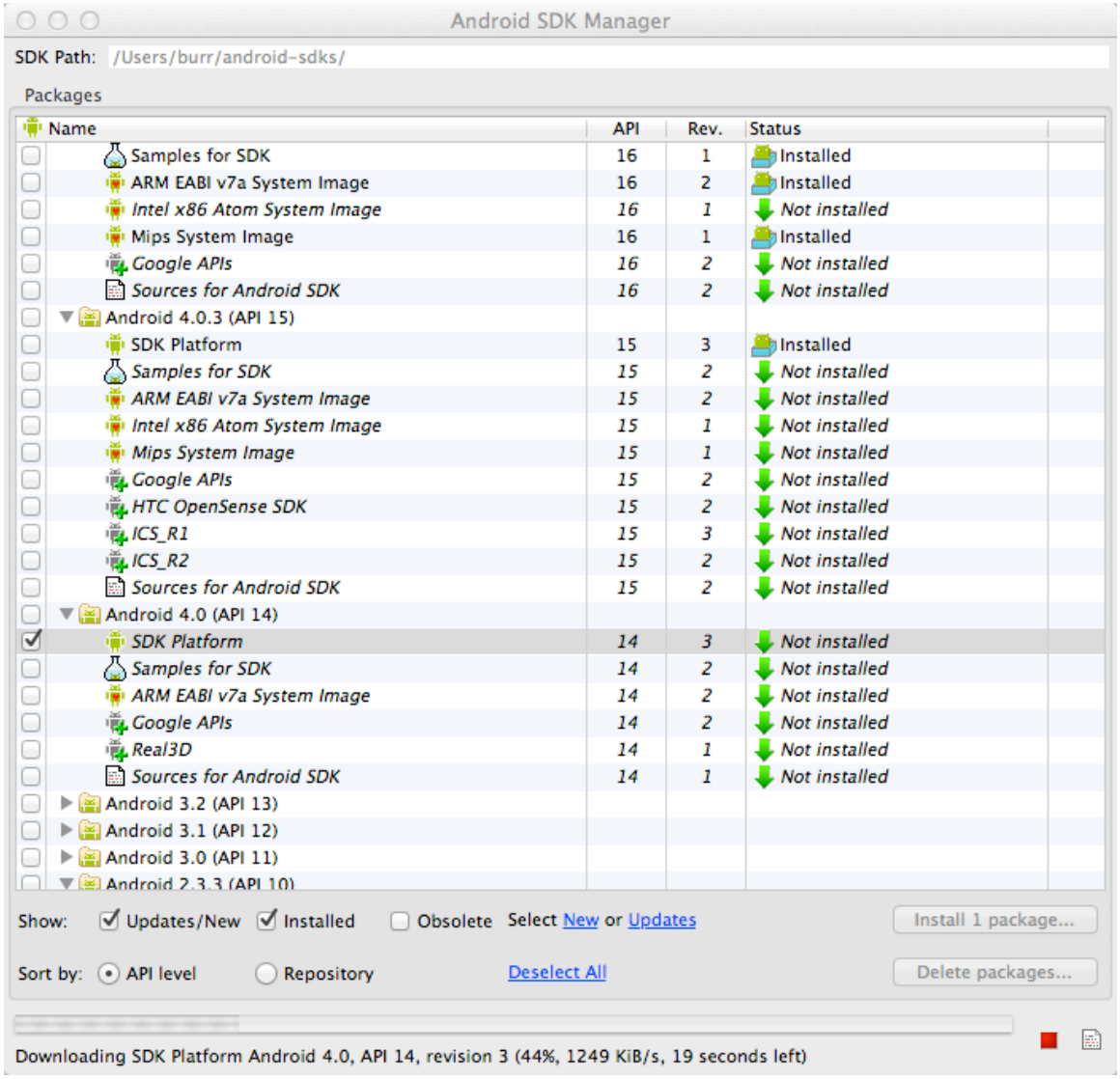


Select Accept on the Choose packages

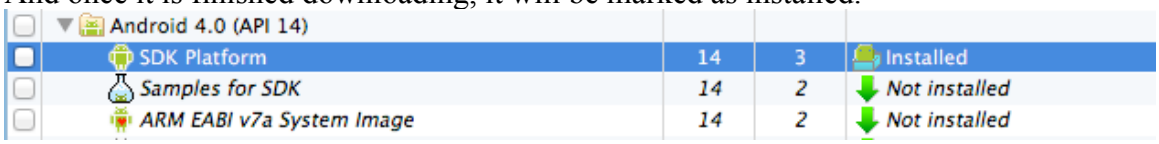


And select Install

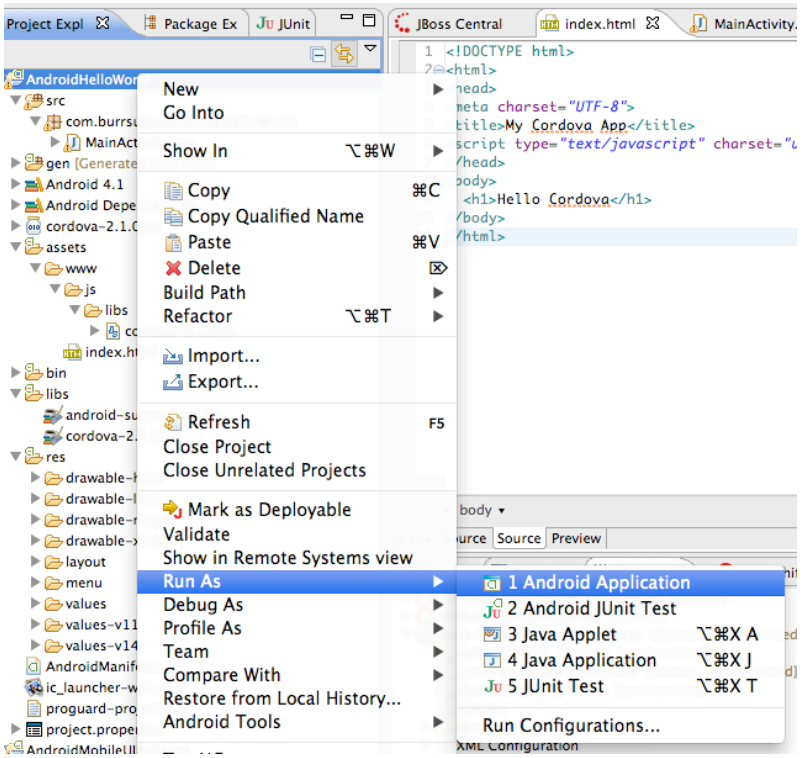
Wait for the download, there is a progress meter at the bottom of the Android SDK Manager dialog.



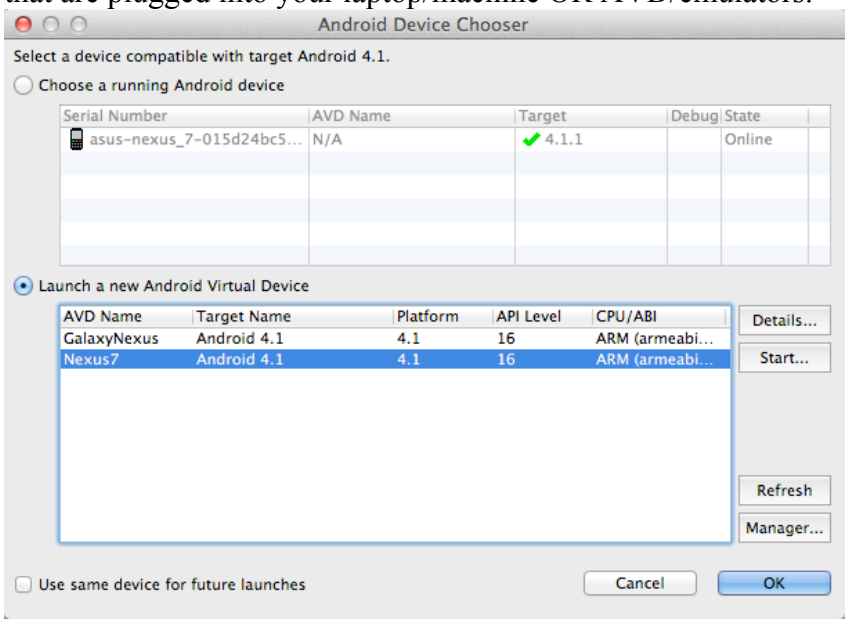
And once it is finished downloading, it will be marked as installed.



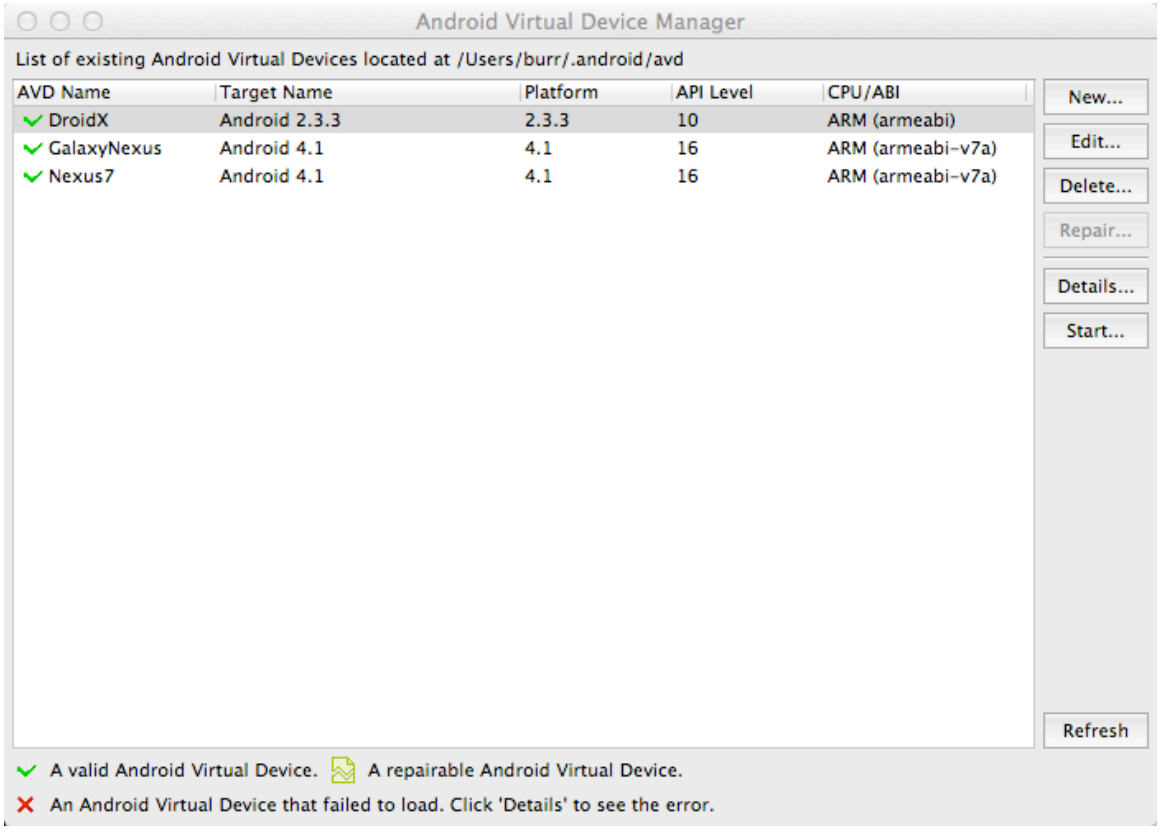
26) Now back in the main Eclipse/JBDS, right-click on the project and select Run As -> Android Application



27) Android Device Chooser dialog allows you to select between any real world devices that are plugged into your laptop/machine OR AVD/emulators.

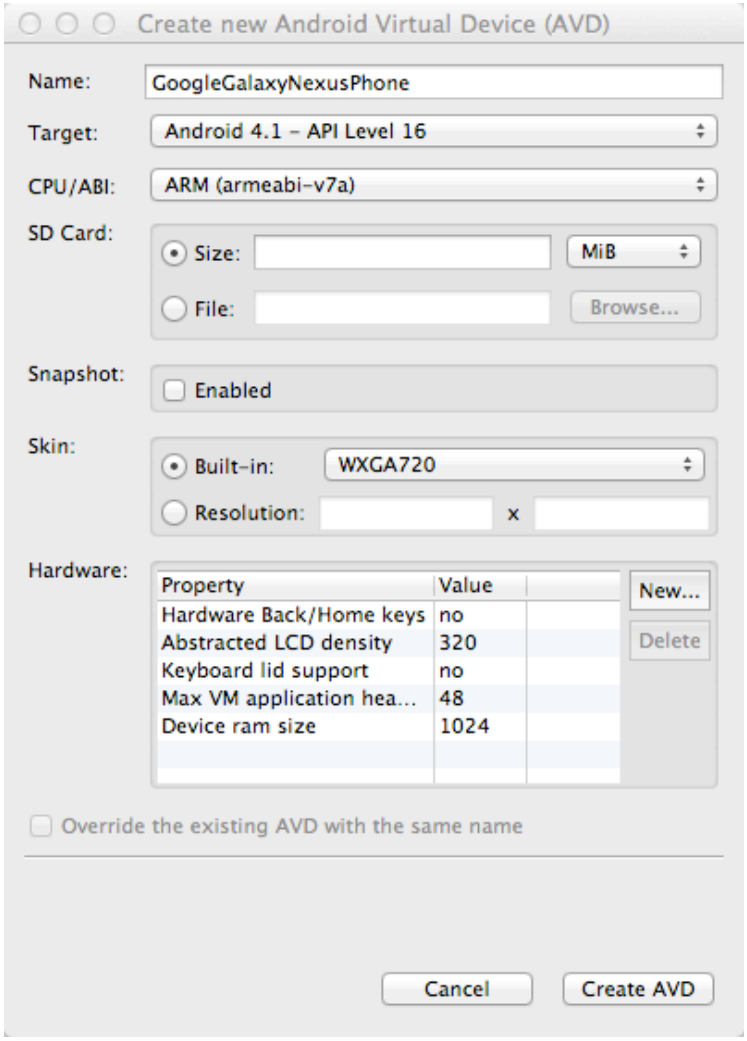


In your case, this screen is mostly likely blank – no real devices nor any AVDs. Let's setup a new AVD – Select the Manager**...** button in the lower right hand corner. This brings up the Android Virtual Device Manager dialog



28) Select the New**…** button
The Name is any string that you wish to provide – make it something descriptive – since I have a Google Galaxy Nexus Phone**…**"GoogleGalaxyNexusPhone"
Target: This drop-down is populated by the Installed SDKs via the SDK Manager
CPU: ARM
SD Card: <blank>
Skin: WXGA720 (or custom resolution)
Property:
Hardware Back/Home keys = no
Abstracted LCD Density = 320
Keyboard lid support = no
Max VM application heap size = 48
Device ram size = 1024

Tip: Use Google with something like **"**Galaxy Nexus AVD Settings**"** to find out what others have tried for these properties. This is one challenge with the Android world – lots and lots of devices with various properties. Some common/popular Android phone configurations can be found in this article:
http://mobile.tutsplus.com/tutorials/android/common-android-virtual-device-configurations/



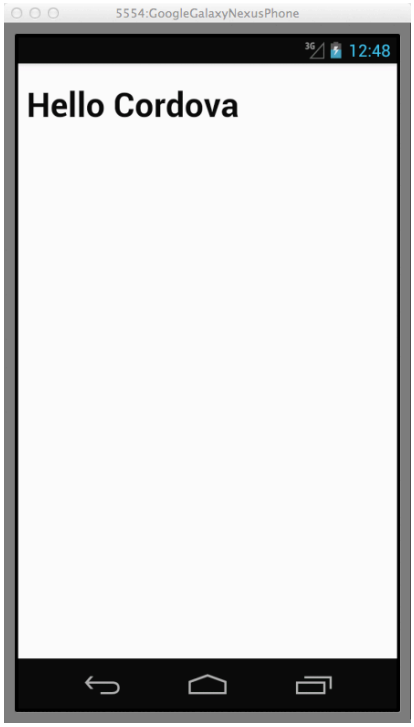
Select Create AVD

Close Android SDK Manager and return to Android Device Chooser

29) Select Refresh to see the newly added AVD – make sure Launch a new Android Virtual Device is selected, highlight the AVD you wish to launch and select OK**...**

The ″default″ end-user experience is typically displayed in the emulator. You will need to manually bypass the lock screen. On iOS, this happens ″auto-magically″ but in Android land, you normally have to manually disengage the lock screen, just click down on the padlock icon and pull your mouse to the right-side of the screen.

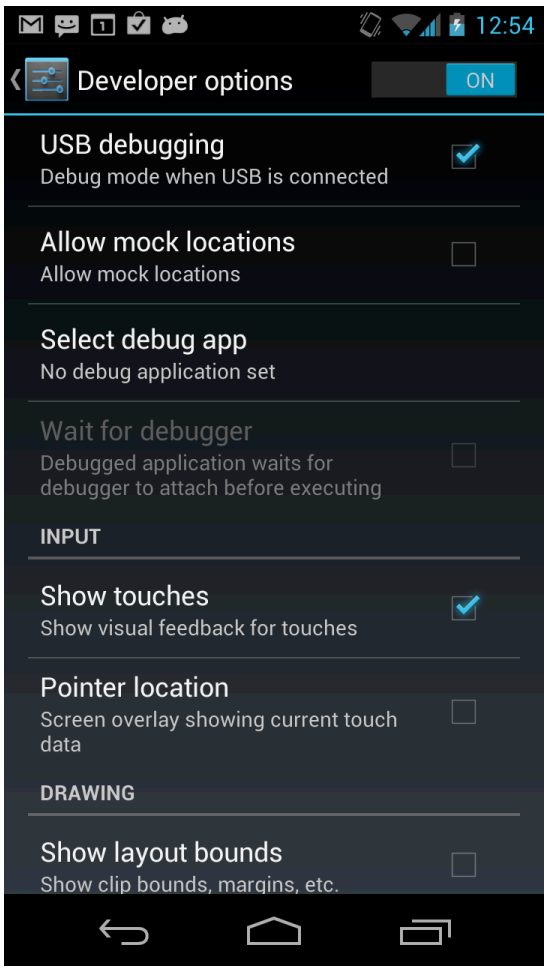
Once the Android OS has fully loaded, it will then run your application. This takes a while, on a slower computer likely several minutes.



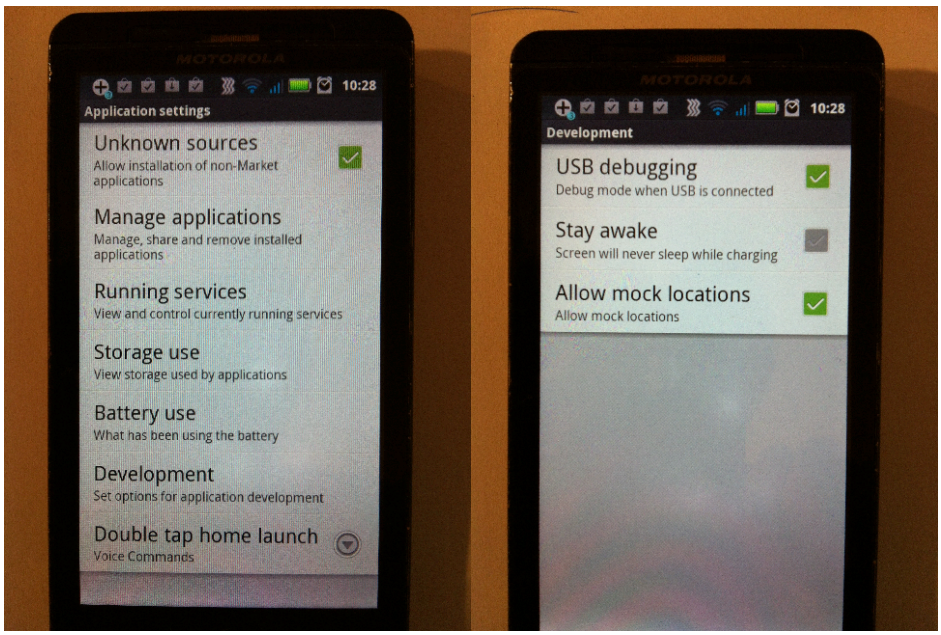
You may find that deployment to a real device is actually faster than the emulator, plus, you can leave the emulator running which helps with the turnaround time during testing.

For real devices (and sometimes the AVD), you are likely to have to manually unlock.

30) Running on a real device running Android 4.x, on the phone/tablet, go to Settings -> Developer Options and make sure USB debugging is toggled on. I also like Show touches to be toggled on.

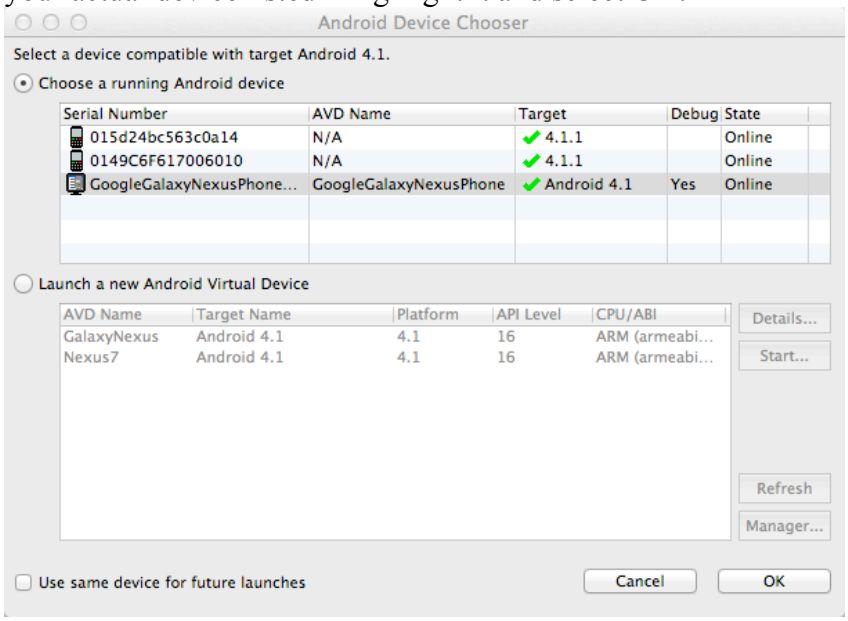


For Android 2.x, hit the physical Settings/Menu button on the phone, then select Applications. Check Unknown sources - Allow installation of non-Market applications. This will allow you and your development buddies to email .apk files around - installing the app from an email attachment.
Then Select the Development option, check on USB debugging and you may find that Stay awake is also helpful if you are frequently editing, saving, redeploying your application.



Many Android 2.x devices do not have the ability to make a screenshot - so I took these photos with my iPhone test device!

31) Now on the Android Device Chooser (Run As Android Application), you should see your actual device listed – highlight it and select OK.

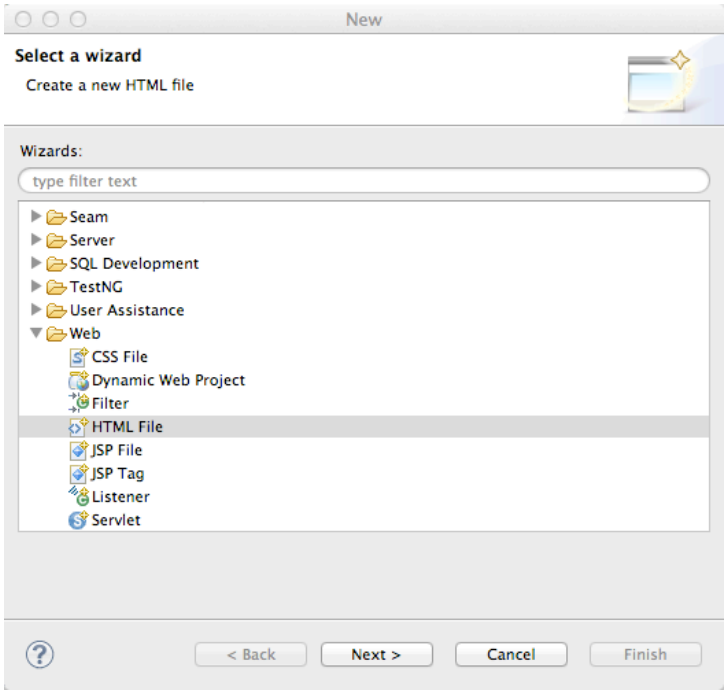


If you are prompted for something called **"**LogCat**"** do say yes. It is a valuable tool for understanding what might be going wrong/right with your deployment to the real device.

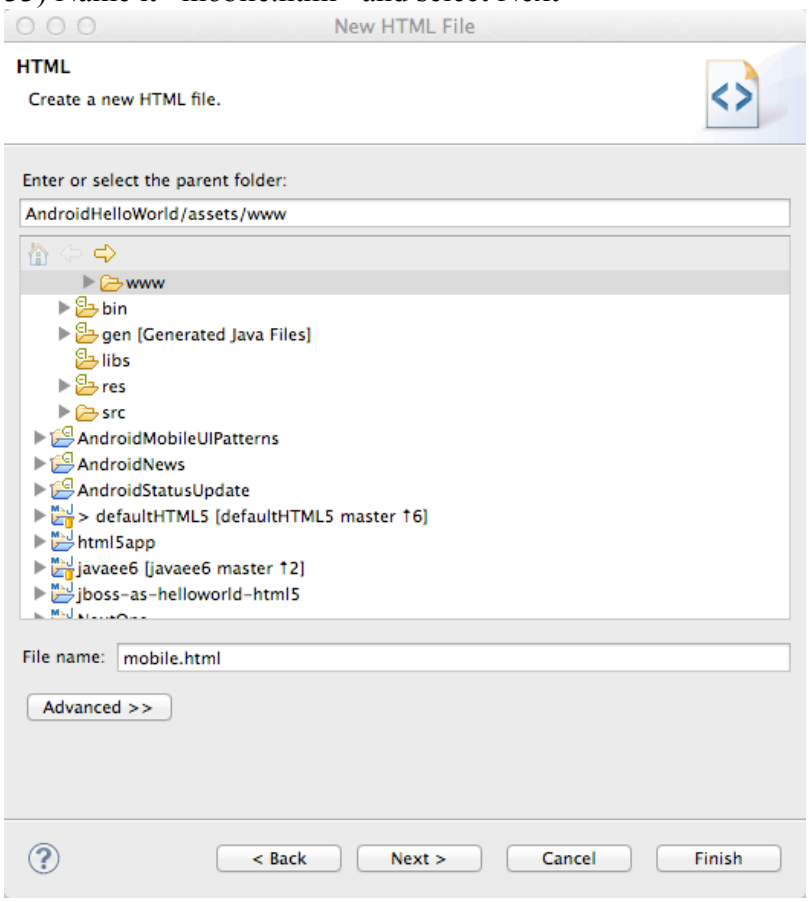
And that's Hello World

Note: I have seen the run operation fail at times, simply try again. In one case, the LogCat message indicated that I needed to restart Eclipse just to get things running and after a quick restart (File - Restart) it was back to normal.

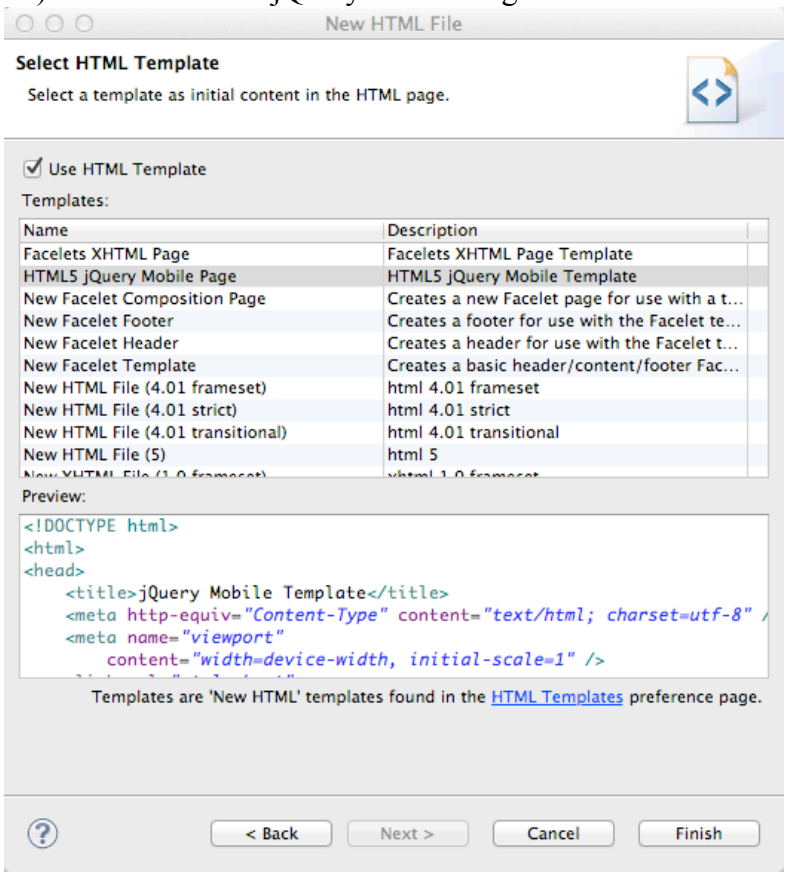
32) Now let's make it more interesting by adding some jQuery Mobile functionality. Right-click on "www" and select New Other.



33) Name it **"**mobile.html**"** and select Next

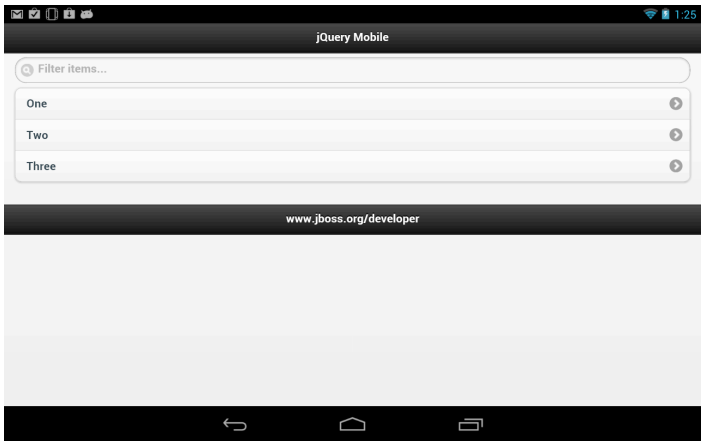


34) Select "HTML5 jQuery Mobile Page" and select Finish



35) Open MainActivity.java – change the reference to index.html to mobile.html
super.loadUrl("file:///android_asset/www/mobile.html");

Save and Run

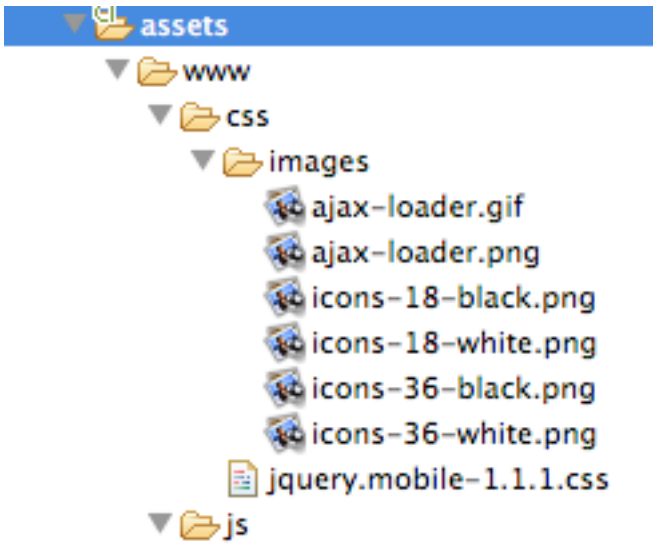


Running on Galaxy Nexus 7 (Asus/Google) – Landscape mode
Note: This file contains references to remotely located jQuery & jQuery Mobile - it is recommended that you bring those files locally into the project for a Phonegap/Cordova style application.

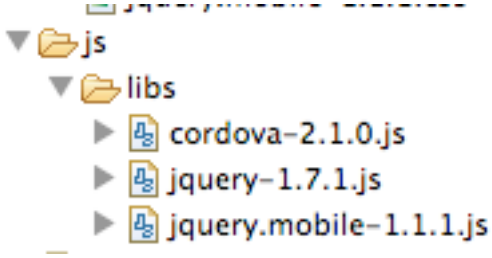
36) Edit mobile.html by pulling in copies of jQuery
Change the external links for jQuery & jQuery Mobile into "local" ones by downloading the necessary files, loading them into your project and change the references. For a mobile app, you want the majority of your resources, especially static ones bundled in the app.

Download jQuery Mobile from http://jquerymobile.com/download/ and unzip
Download the uncompressed/development jQuery http://jquery.com/download/

Create a "css" directory underneath "www" - drag & drop the "jquery.mobile-1.1.1.css" and the "images" into "css" as shown in the following screenshot.



In your "js/libs" folder, add "jquery.mobile-1.1.1.js" and "jquery-1.7.1.js".



Note: It is normally recommended to use the .min versions of the JS libs, however, Eclipse often complains about those files as it cannot parse them. This can even become a bigger problem when Eclipse refuses to let you deploy (Run As) your app. Plus, these files are being bundled in your app for deployment the device so size is less important.

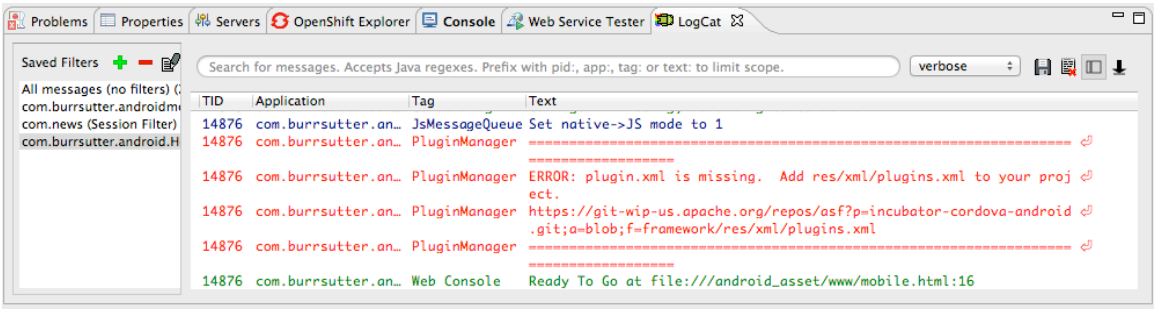
Now, change the references in the mobile.html file to point to the correct location and the correct file names.

```
<link rel="stylesheet" href="css/jquery.mobile-1.1.1.css" />
<script type="text/javascript" src="js/libs/jquery-1.7.1.js"></script>
<script type="text/javascript"
        src="js/libs/jquery.mobile-1.1.1.js"></script>
```

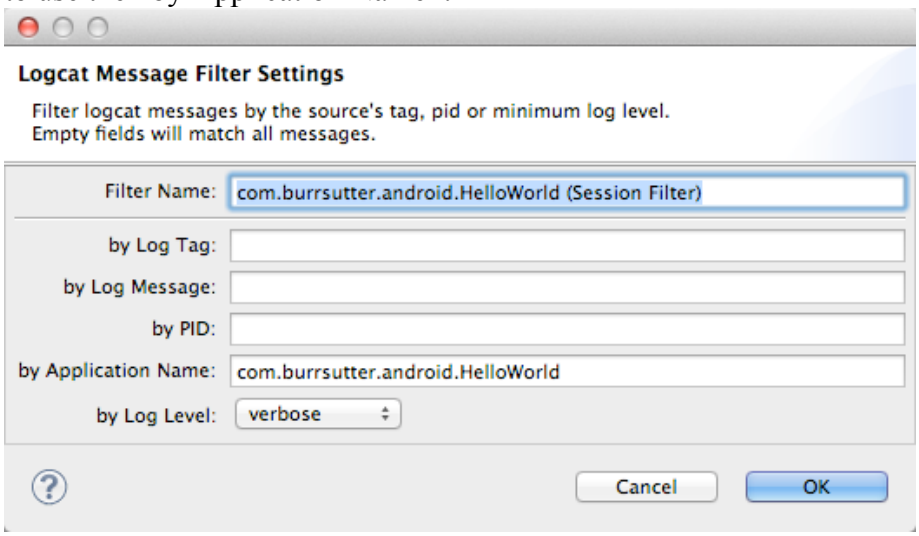
Also, change the alert(“Ready To Go”); to console.log

```
<script type="text/javascript">
      $(document).on("pageinit", "#page1", function(event){
            console.log("Ready To Go");
      });
</script>
```

And in the LogCat tab you should see the console.log output.



You may need to setup a filter by clicking on the green plus sign + and the easiest filter is to use the “by Application Name”.



If you receive a “[ERROR] Error initializing Cordova: Class not found” message then it is likely you forgot to add the xml/config.xml directory & file under “res” in the project.

Note: It is much easier to perform your JavaScript debugging via Chrome, Safari or FireFox than it is using console.log and LogCat here but you can at least see that your application attempted to load the JS/HTML files if nothing else works. Weinre, a remote web inspector debugger is covered in another document.

Also, make sure the device is awake and not locked, sitting on the home screen before attempting to Run As Android Application – it tends to work more often in that scenario. In Developer options (on the device), there is an option to Stay awake.

37) Add some Cordova Magic - Geolocation (where is the device on the planet)
First, add the reference to cordova.js before the script tag for on("pageinit"…
<script type="text/javascript" charset="utf-8" src="js/libs/cordova-2.1.0.js"></script>

```
<link rel="stylesheet"
    href="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css" />
<script type="text/javascript"
    src="http://code.jquery.com/jquery-1.7.1.min.js"></script>
<script type="text/javascript"
    src="http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js"></script>

<script type="text/javascript" charset="utf-8" src="js/libs/cordova-2.1.0.js"></script>

<script type="text/javascript">
    $(document).on("pageinit", "#page1", function(event){
        console.log("Ready To Go");
    }); // on pageinit
```

Next, below the $(document).on("pageinit"… block add an eventlistener for "deviceready" - you only want your JS to begin running after Phonegap/Apache Cordova have established the environment.

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

Then add the onDeviceReady function that receives this event.

```
function onDeviceReady() {
  console.log("Device Ready To Go");
  console.log("Asking for geo location");
  navigator.geolocation.getCurrentPosition(onGeoSuccess, onGeoError);
} // onDeviceReady
```

The getCurrentPosition() call above has two arguments - what function to call when things go well and what function to call when things go poorly. Add those two functions into your overall <script> block.

```
<script type="text/javascript">
    $(document).on("pageinit", "#page1", function(event){
        console.log("Ready To Go");
    }); // on pageinit

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

    function onDeviceReady() {
        console.log("Device Ready To Go");
        console.log("Asking for geo location");
        navigator.geolocation.getCurrentPosition(onGeoSuccess, onGeoError);
    } // onDeviceReady
```

The onGeoSuccess function receives a position object that can be peeled apart and displayed on screen.

```
function onGeoSuccess(position) {
   var element = document.getElementById('geolocation');
   element.innerHTML =
    'Latitude: '  + position.coords.latitude          + '<br />' +
    'Longitude: ' + position.coords.longitude         + '<br />' +
    'Altitude: '  + position.coords.altitude          + '<br />' +
    'Accuracy: '  + position.coords.accuracy          + '<br />' +
    'Alt Accuracy: ' + position.coords.altitudeAccuracy + '<br />' +
    'Heading: '   + position.coords.heading           + '<br />' +
    'Speed: '     + position.coords.speed             + '<br />' +
    'Timestamp: ' + position.timestamp                + '<br />';
 }
```

The onGeoError function receives the error message, if there is one.

```
function onGeoError(error) {
       alert('code: '    + error.code    + '\n' +
             'message: ' + error.message + '\n');
}
```

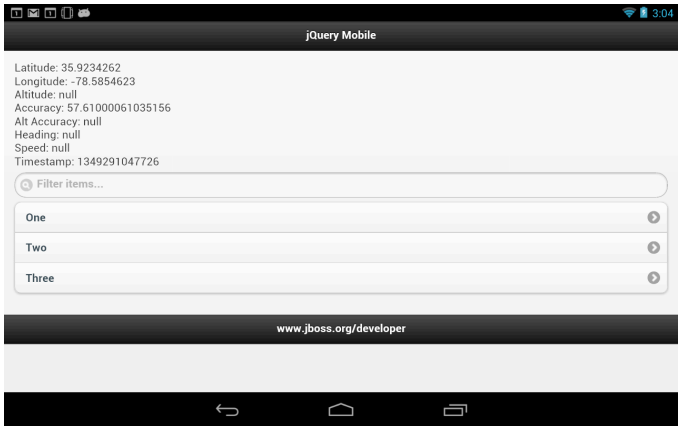
```
    // onSuccess Geolocation
    //
    function onGeoSuccess(position) {
        var element = document.getElementById('geolocation');
        element.innerHTML = 'Latitude: '          + position.coords.latitude          + '<br />' +
                            'Longitude: '         + position.coords.longitude         + '<br />' +
                            'Altitude: '          + position.coords.altitude          + '<br />' +
                            'Accuracy: '          + position.coords.accuracy          + '<br />' +
                            'Alt Accuracy: '      + position.coords.altitudeAccuracy  + '<br />' +
                            'Heading: '           + position.coords.heading           + '<br />' +
                            'Speed: '             + position.coords.speed             + '<br />' +
                            'Timestamp: '         + position.timestamp                + '<br />';
    }

    // onError Callback receives a PositionError object
    //
    function onGeoError(error) {
        alert('code: '    + error.code    + '\n' +
              'message: ' + error.message + '\n');
    }
```

Finally, add the HTML tag needed display the results goes in the data-role=”content” section.

```
<div id="geolocation">Finding geolocation...</div><p>
```

```
<body>
    <div data-role="page" id="page1">
        <div data-role="header">
            <h1>jQuery Mobile</h1>
        </div>
        <div data-role="content">
            <div id="geolocation">Finding geolocation...</div><p>
            <ul id="listOfItems" data-role="listview" data-inset="true"
                data-filter="true">
                <li><a href="">One</a></li>
                <li><a href="">Two</a></li>
                <li><a href="">Three</a></li>
            </ul>
        </div>
        <div data-role="footer">
            <h4>www.jboss.org/developer</h4>
        </div>
    </div>
</body>
```

The result:



For more information on how Apache Cordova/Phonegap addresses Geolocation check out the API docs at:

http://docs.phonegap.com/en/2.1.0/cordova_geolocation_geolocation.md.html#Geolocation

38) Accelerometer

Inside the onDeviceReady function, add the block of code that establishes the "watcher" for the device's accelerometer.

```
var options = {};
options.frequency = 1000;
console.log("Hitting Accelerometer");

var accelerationWatch =
   navigator.accelerometer.watchAcceleration(
     updateAccelerationUI, function(ex) {
       console.log("accel fail (" + ex.name + ": " + ex.message + ")");
   }, options);
```

The 1000 represents milliseconds, in this case, update my callback function every 1 second.

Next add the function for updateAccelertationUI

```
// called when Accelerometer detects a change
   function updateAccelerationUI(a) {
      document.getElementById('my.x').innerHTML = a.x;
      document.getElementById('my.y').innerHTML = a.y;
      document.getElementById('my.z').innerHTML = a.z;
   } // updateAccelerationUI
```

Screenshot:

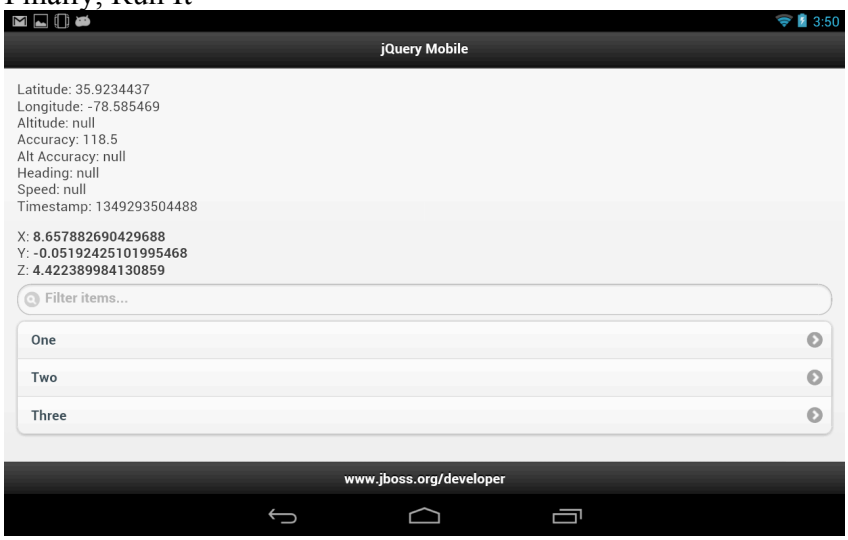
```
    function onDeviceReady() {
        console.log("Device Ready To Go");
        console.log("Asking for geo location");
        navigator.geolocation.getCurrentPosition(onGeoSuccess, onGeoError);

        var options = {};
        options.frequency = 1000;
        console.log("Hitting Accelerometer");

        var accelerationWatch = navigator.accelerometer.watchAcceleration(
                updateAccelerationUI, function(ex) {
                    console.log("accel fail (" + ex.name + ": " + ex.message + ")");
                }, options);

    } // onDeviceReady

    // called when Accelerometer detects a change
    function updateAccelerationUI(a) {
        document.getElementById('my.x').innerHTML = a.x;
        document.getElementById('my.y').innerHTML = a.y;
        document.getElementById('my.z').innerHTML = a.z;
    } // updateAccelerationUI
```

Now add the HTML elements to display the X, Y and Z in the content section

```
      <div>X: <b id="my.x"></b> </div>
      <div>Y: <b id="my.y"></b> </div>
      <div>Z: <b id="my.z"></b> </div>
      </p>
```

```
<body>
    <div data-role="page" id="page1">
        <div data-role="header">
            <h1>jQuery Mobile</h1>
        </div>
        <div data-role="content">
            <div id="geolocation">Finding geolocation...</div><p>

            <div>X: <b id="my.x"></b> </div>
            <div>Y: <b id="my.y"></b> </div>
            <div>Z: <b id="my.z"></b> </div>
            </p>

            <ul id="listOfItems" data-role="listview" data-inset="true"
                data-filter="true">
                <li><a href="">One</a></li>
                <li><a href="">Two</a></li>
                <li><a href="">Three</a></li>
```

Finally, Run It



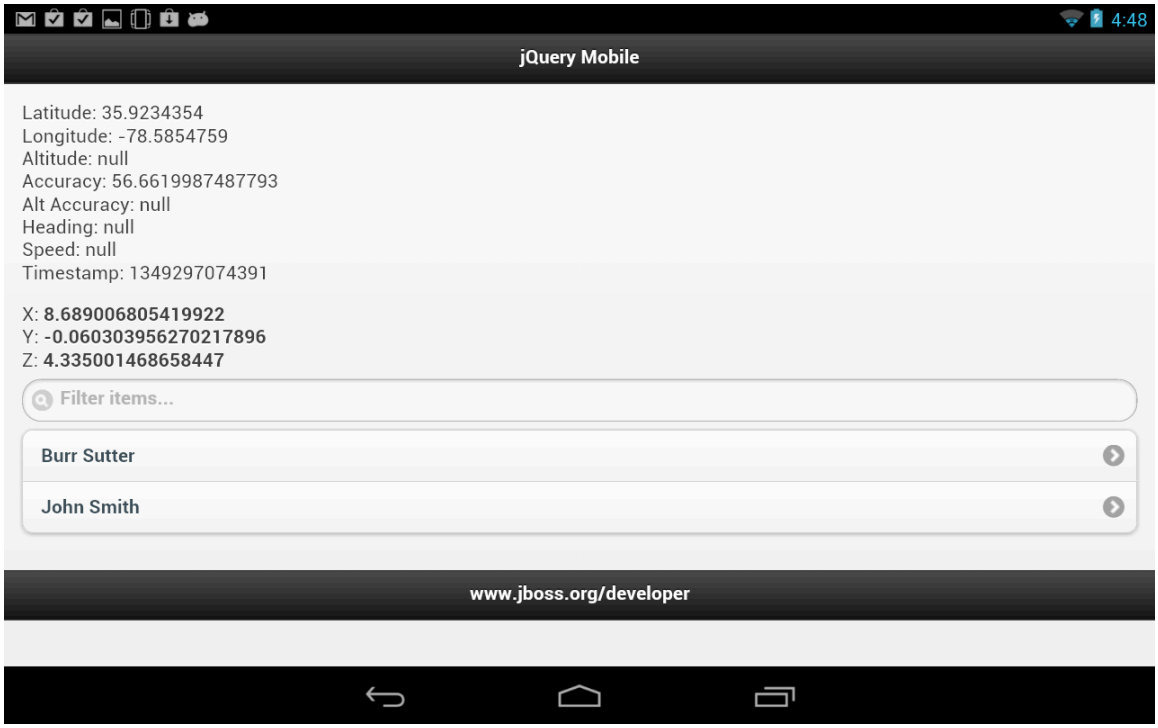
39) Data & REST

In the on pageinit function add the block of jQuery code to retrieve data from a rest endpoint and load it into the UL called listOfItems. This UL was part of the original template so it should still be in your HTML body.

```
$.getJSON("http://cordovaendpoint-burrsutter.rhcloud.com/rest/members",
function(members) {
    // console.log("returned are " + members);
    var listOfMembers = $("#listOfItems");
    listOfMembers.empty();
    $.each(members, function(index, member) {
           // console.log(member.name);
          listOfMembers.append(
            "<li><a href='#'>" + member.name + "</a>");
    });
    listOfMembers.listview("refresh");
});
```

The getJSON call is accessing the network therefore the Android-based app needs to have this permission setup in AndroidMainfest.xml. That was already handled back in step 20. In addition, you must add the URL to the Cordova whitelist. The easiest solution is as follows:
Open res/xml/config.xml -
And update access origin to equal **"*"** - this is primarily useful for development for an app you are deploying to real end-users via the Google Play Store or Apple App Store, you will wish to be more specific.

```
<cordova>
    <!--
    access elements control the Android whitelist.
    Domains are assumed blocked unless set otherwise
     -->

    <access origin="http://127.0.0.1*"/> <!-- allow local pages -->

    <!-- <access origin="https://example.com" /> allow any secure requests to example.com -->
    <!-- <access origin="https://example.com" subdomains="true" /> such as above, but including
    <access origin=".*"/>
```



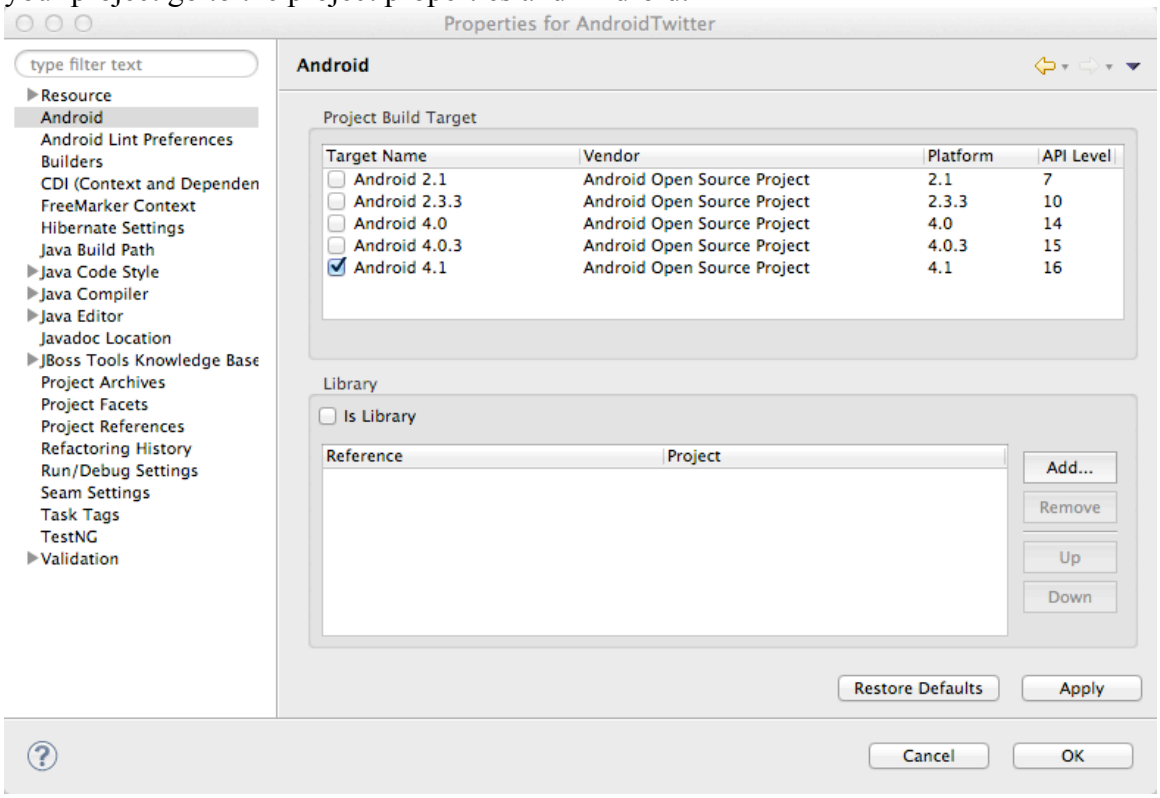
And if you wish to be more adventurous, wrap this logic in a check for Wifi vs 3G vs no connection and make a determination as to how to display a message to the end-user.

```
var networkState = navigator.network.connection.type;
```

http://docs.phonegap.com/en/2.1.0/cordova_connection_connection.md.html#Connection

This ends the tutorial portion of this document, what follows are some general tips & tricks that I have learned while working with PhoneGap/Apache Cordova + Android.

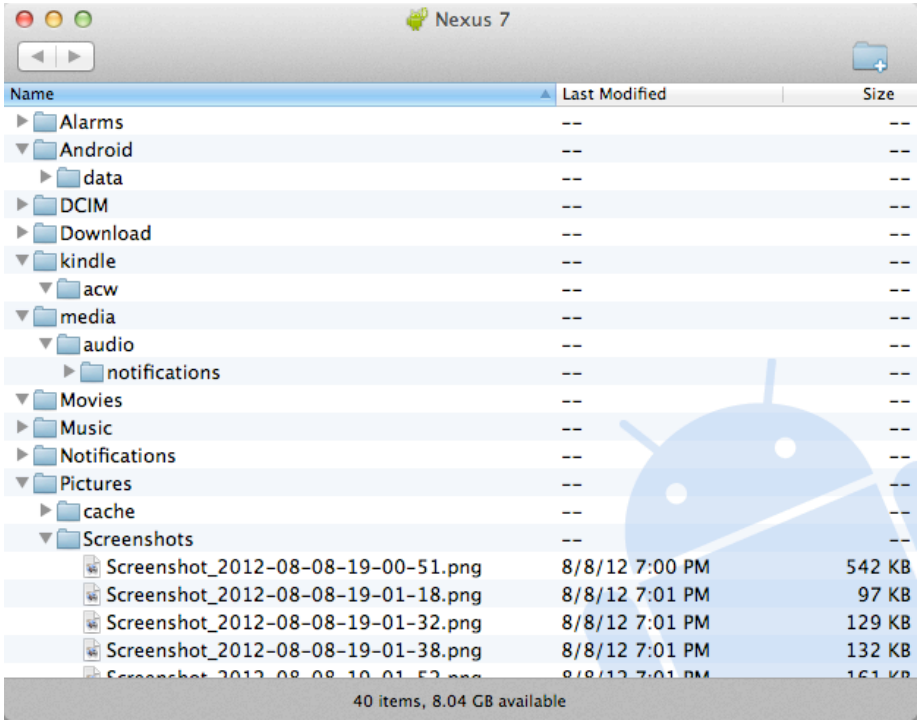
Tips & Tricks:
1) Target Android Version: In case you wish to switch the Android version targets for your project go to the project properties and Android.



And also change your AndroidManifest.xml

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="15" />
```

2) Android File Transfer (for Mac Users) – allows you to grab files from the device when plugged in via USB



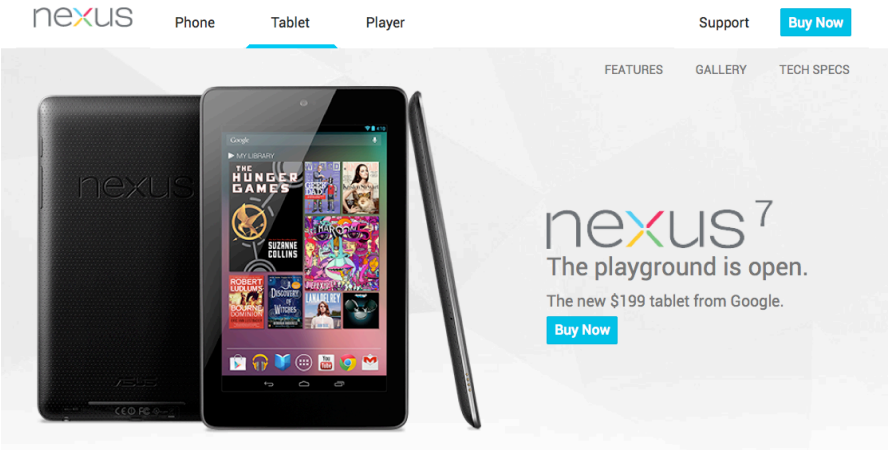
http://www.android.com/filetransfer/

3) Phonegap Plugins
https://github.com/phonegap/phonegap-plugins

ChildBrowser is one of the most popular – it allows you to render a webpage, inside of your application

4) REST Endpoints – you need to "open" access to the specific Internet domains in res/xml/config.xml
<access origin=".*"/>

5) Google Nexus 7 is $199 – great inexpensive testing device



Biggest downside**…**it uses Android 4.1, which is not representative of the overall market – the majority of Android users are still on 2.2 or 2.3.x – and on lower powered CPUs/RAM combinations.

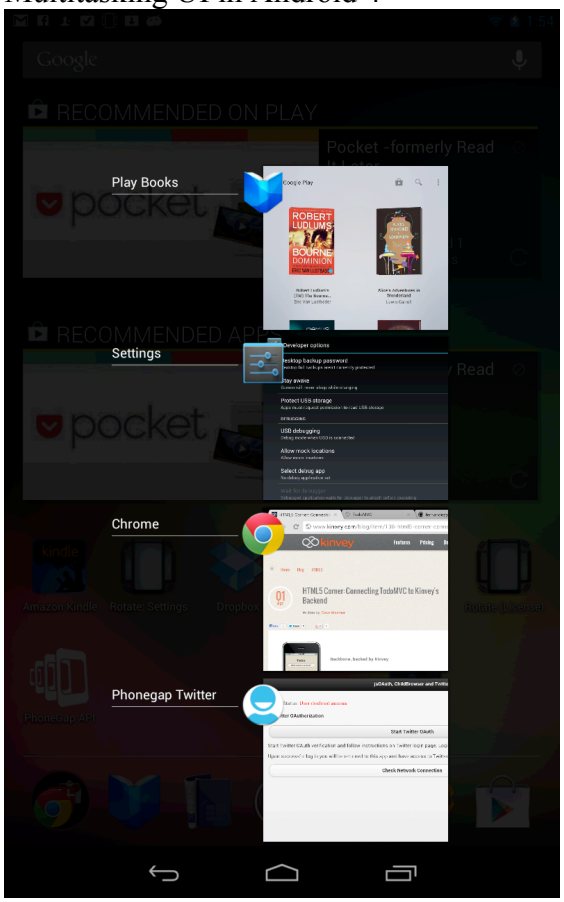
6) Change MainActivity title/label on the installed application – res/values/strings.xml

index.html | JBoss Central | config.xml | strings.xml | strings.xml | 3

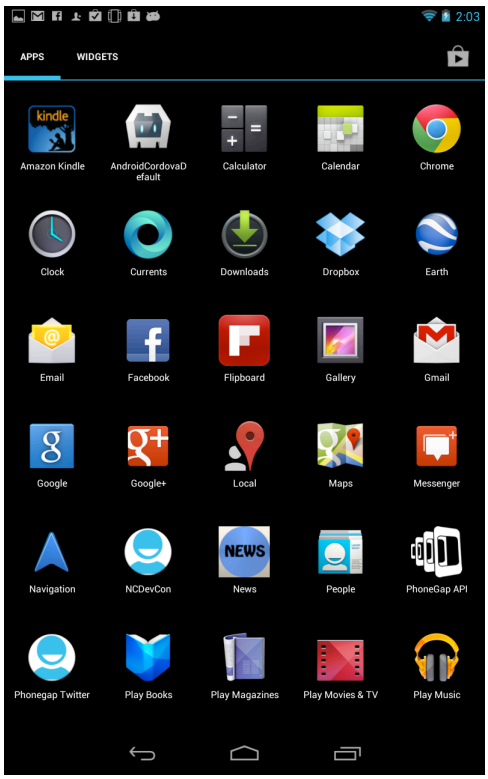
```
1  <resources>
2
3      <string name="app_name">AndroidTwitter</string>
4      <string name="hello_world">Hello world!</string>
5      <string name="menu_settings">Settings</string>
6      <string name="title_activity_main">Phonegap Twitter</string>
7
8  </resources>
```

This will change the application name on the launch icon as well as when multi-tasking.

Multitasking UI in Android 4



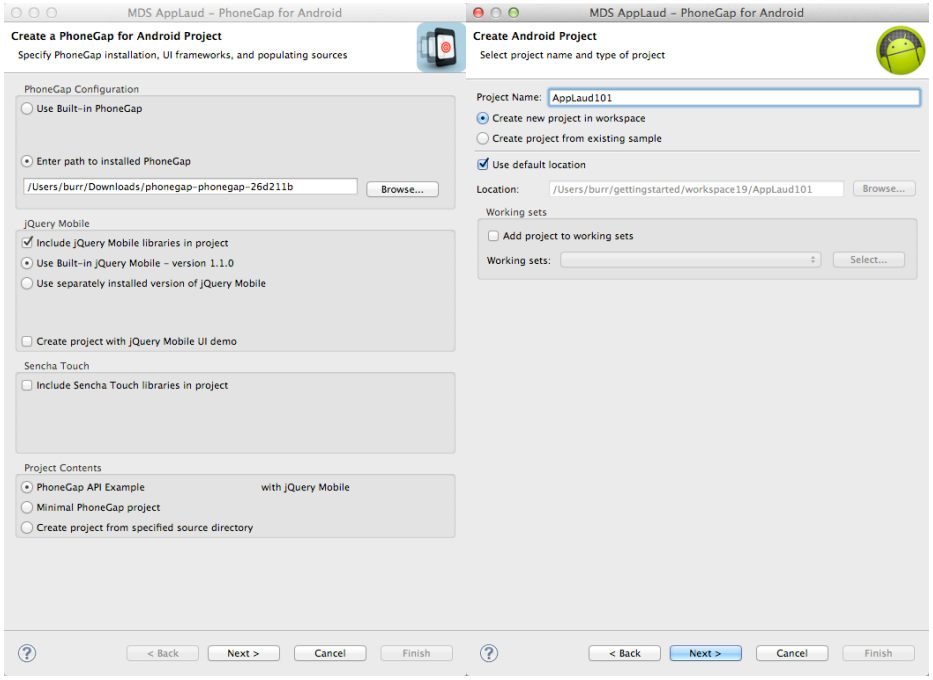
Application Launcher in Android 4

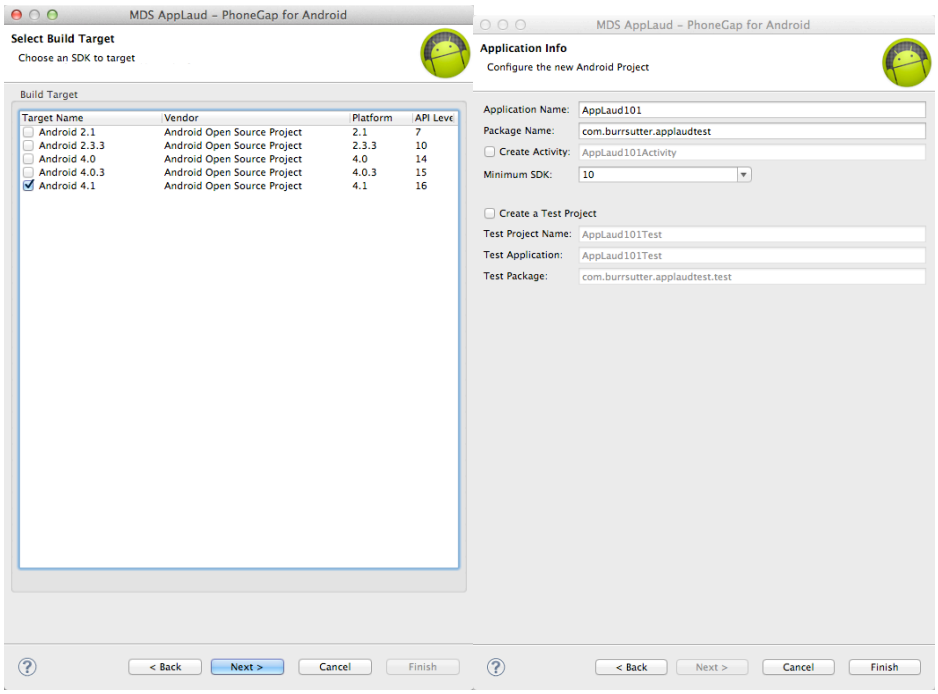


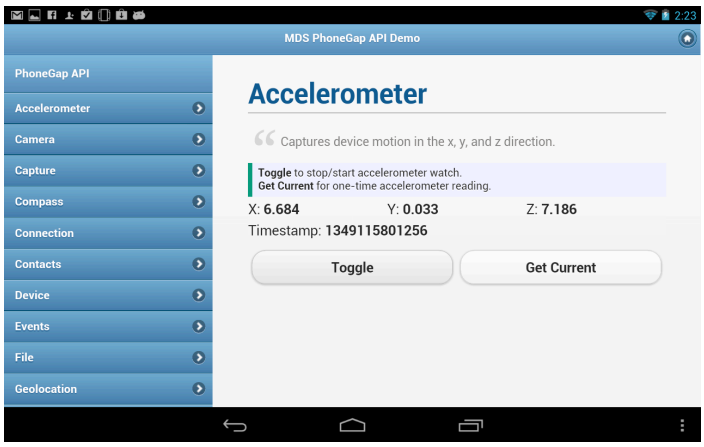
7) Phonegap API Explorer – to test your device and phonegap**…**install
https://play.google.com/store/apps/details?id=org.coenraets.phonegapexplorer&hl=en

8) Screenshot on Nexus 7 is to press the power button and down volume button, simultaneously, holding for approximately 1 to 2 seconds.

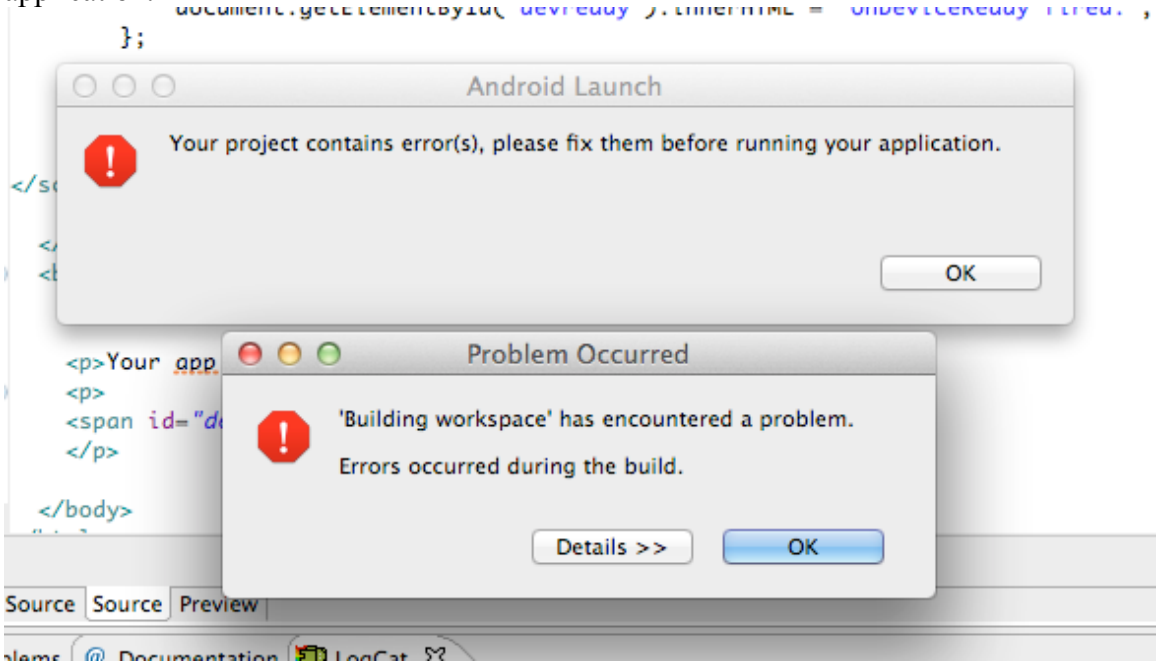
9) Faster Getting Started Experience with MDS AppLaud Eclipse Plug-In
http://www.mobiledevelopersolutions.com/home/start
Note: We are not recommending this to customers – but it is useful for personal **"**ramp-up**"**.

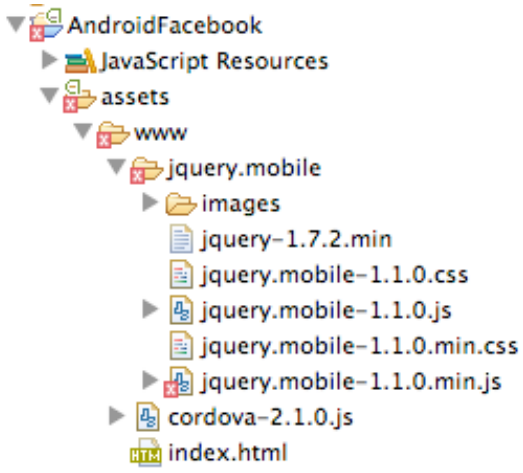

MDS AppLaud – PhoneGap for Android
Create a PhoneGap for Android Project
Specify PhoneGap installation, UI frameworks, and populating sources
PhoneGap Configuration
Use Built-in PhoneGap
Enter path to installed PhoneGap
/Users/burr/Downloads/phonegap-phonegap-26d211b
Browse...
jQuery Mobile
Include jQuery Mobile libraries in project
Use Built-in jQuery Mobile – version 1.1.0
Use separately installed version of jQuery Mobile
Create project with jQuery Mobile UI demo
Sencha Touch
Include Sencha Touch libraries in project
Project Contents
PhoneGap API Example
with jQuery Mobile
Minimal PhoneGap project
Create project from specified source directory
< Back
Next >
Cancel
Finish
MDS AppLaud – PhoneGap for Android
Create Android Project
Select project name and type of project
Project Name:
AppLaud101
Create new project in workspace
Create project from existing sample
Use default location
Location:
/Users/burr/gettingstarted/workspace19/AppLaud101
Browse...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Cancel
Finish


MDS AppLaud – PhoneGap for Android
Select Build Target
Choose an SDK to target
Build Target
Target Name
Vendor
Platform
API Leve
Android 2.1
Android Open Source Project
2.1
7
Android 2.3.3
Android Open Source Project
2.3.3
10
Android 4.0
Android Open Source Project
4.0
14
Android 4.0.3
Android Open Source Project
4.0.3
15
Android 4.1
Android Open Source Project
4.1
16
< Back
Next >
Cancel
Finish
MDS AppLaud – PhoneGap for Android
Application Info
Configure the new Android Project
Application Name:
AppLaud101
Package Name:
com.burrsutter.applaudtest
Create Activity:
AppLaud101Activity
Minimum SDK:
10
Create a Test Project
Test Project Name:
AppLaud101Test
Test Application:
AppLaud101Test
Test Package:
com.burrsutter.applaudtest.test
< Back
Next >
Cancel
Finish

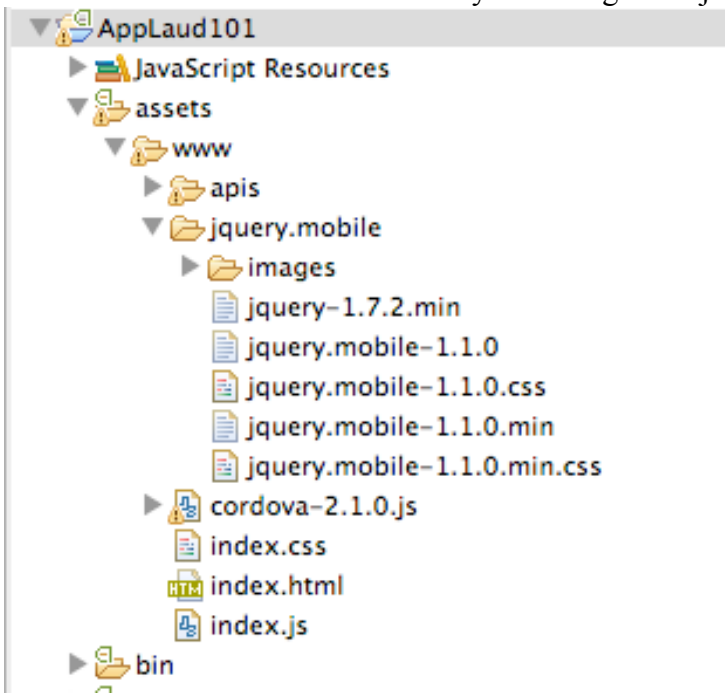


10) Eclipse complains about various JS libs – and in some cases won't let you deploy the application.





Workaround: Rename the libraries by removing their .js extensions



Caution: This may mean that Eclipse will not recognize them as JavaScript files and not provide the correct editor. Another option is to use the full or non-minified versions of the JavaScript library that you are interested in. Since these files should be bundled in your Cordova-based app's distribution, there is no significant network download penalty to downloading the file at runtime.

11) You really wish to build apps for the iPhone, but you are too damn cheap to buy a Mac – use https://build.phonegap.com/ - it takes care of the cross-platform app creation for you – iOS, Android, Blackberry 5/6/7, webOS, Symbian and Windows Phone.

12) Debugging**...**on device - Weinre