How to Build and Android App with an Action Bar

ECE 480 Team 1

Nicholas Huff

3/28/14

Contents

Introduction

Over 50% of all adult Americans own a smart phone[1], and most of these people own phones that operate with the Android operating system. There are countless smartphone apps available, yet newer and more efficient apps with greater functionality are coming out all the time. As the spread of Android becomes more prolific, the demand for applications to extend the usability of these Android enabled devices becomes greater.

There are several ways to create an Android App, but the most flexible way to create an app is to use the Eclipse IDE and program in Java. Eclipse has the added feature of the ADT Plugin, giving it additional tools to program Android Apps effectively. This tutorial will walk the reader through building a basic Android app that includes buttons along the action bar. Other tutorials that go into greater detail can be found at:

http://developer.android.com/index.html

Objective

The objective of this tutorial is to help readers to design a basic Android App to lay a foundation for apps with greater complexity. This tutorial includes information on how to setup your system to begin programming with Eclipse, create a basic first page, and how to add buttons to your action bar.

How to Setup Programming Environment

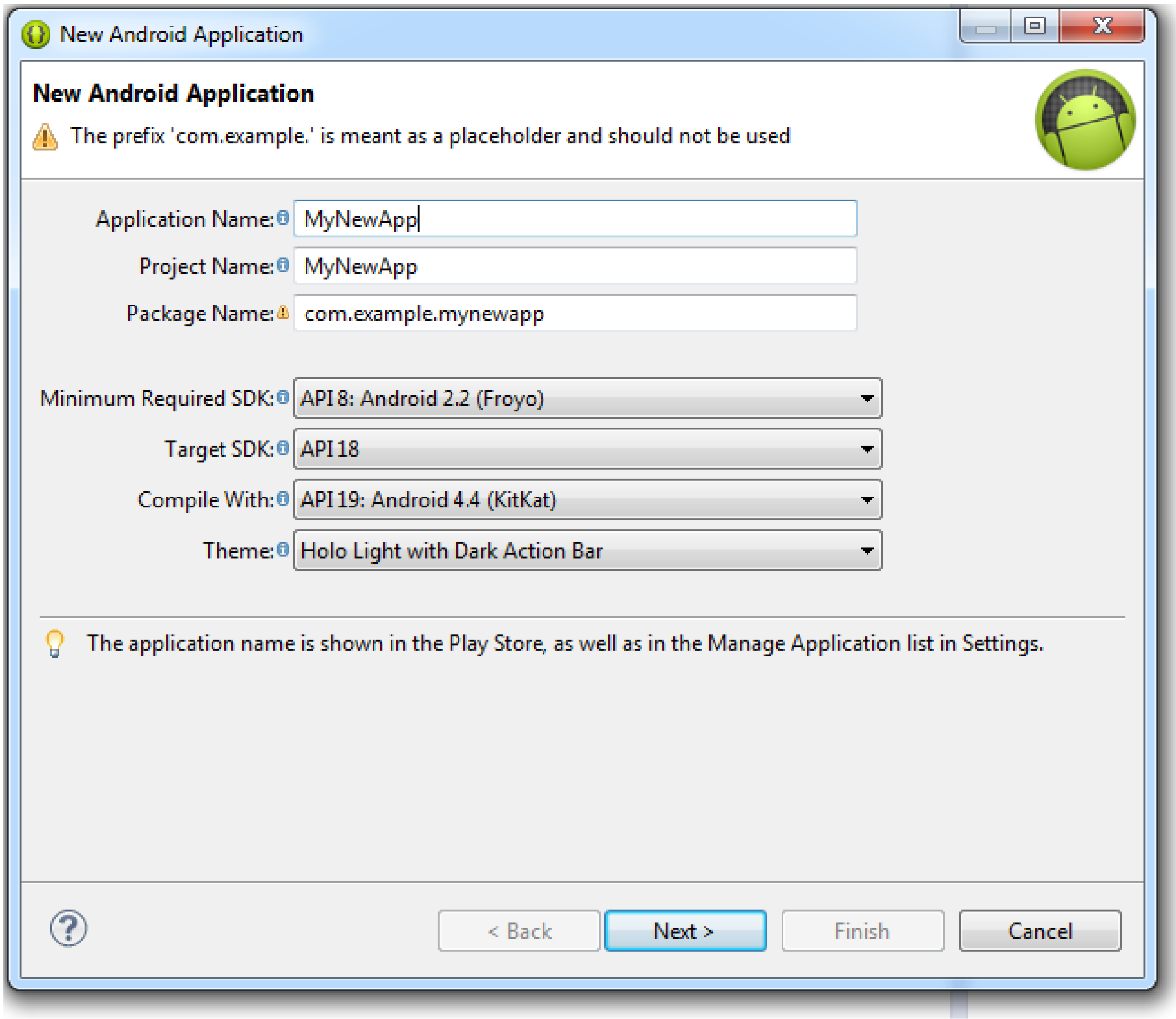
Your first step will be to download/install Eclipse at:
https://www.eclipse.org/downloads/packages/eclipse-standard-432/keplersr2

Next, download/install the Android SDK at
http://developer.android.com/training/basics/firstapp/index.html

Next, download/install the ADT plugin for Eclipse:
http://developer.android.com/tools/sdk/eclipse-adt.html

Once you have completed these tasks, it's time to open up Eclipse and begin building your first app.

1. Click **New** then Android Application in the toolbar.
2. Under the next window type out “My New App” in the application name. The SDK is simply the Android version you are going to be running your app on. The “Theme” of your app pertains to the color scheme of your action bar and activity pages. For now, keep it as the Holo Light package then click next.



New Android Application

New Android Application

⚠ The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name: MyNewApp

Project Name: MyNewApp

Package Name: com.example.mynewapp

Minimum Required SDK: API 8: Android 2.2 (Froyo)

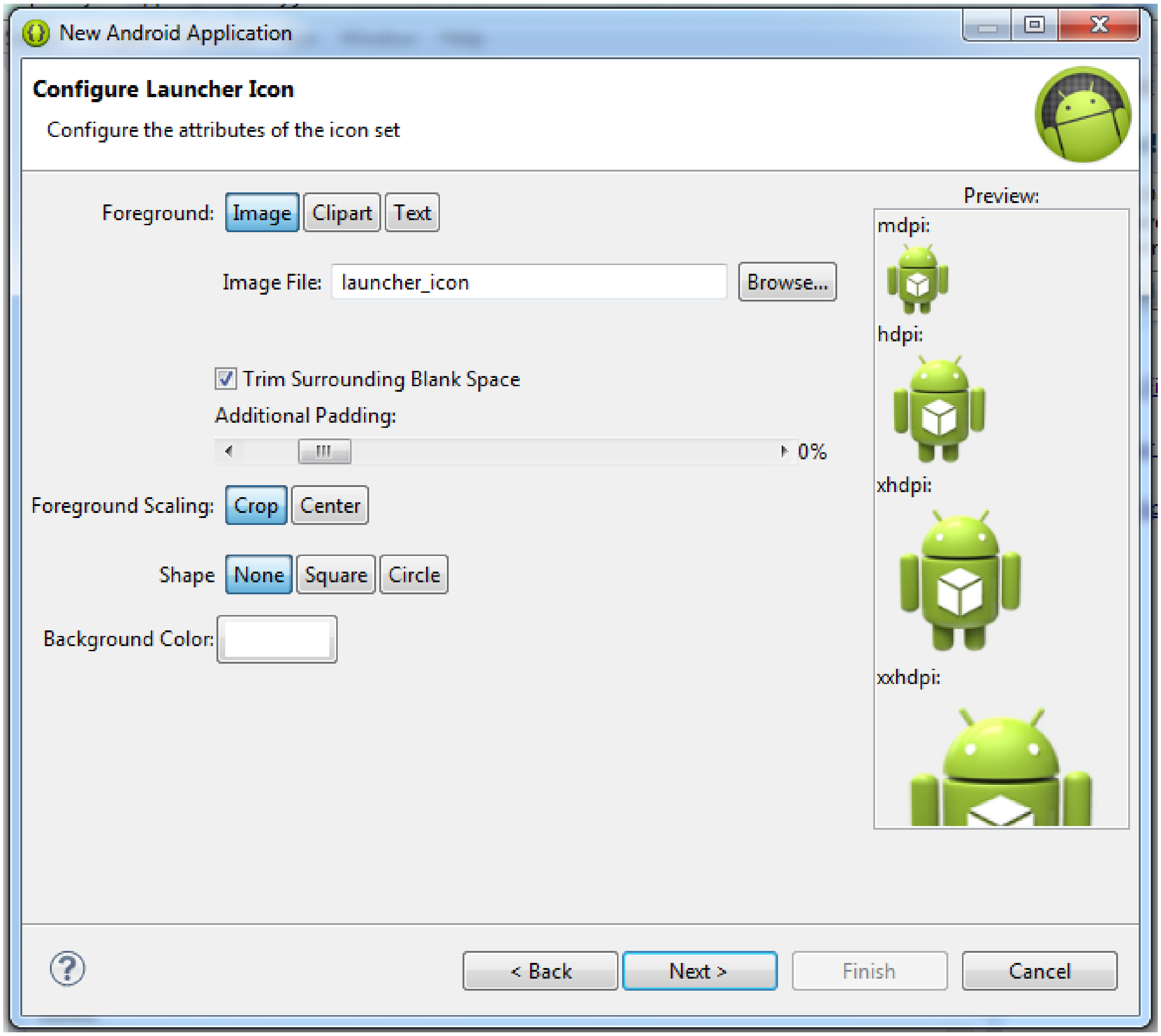
Target SDK: API 18

Compile With: API 19: Android 4.4 (KitKat)

Theme: Holo Light with Dark Action Bar

The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

3. Click next.
4. Select your App’s icon by clicking browse under foreground. You can also change the background color for your app here.



New Android Application

Configure Launcher Icon

Configure the attributes of the icon set

Foreground: Image

Image File: launcher_icon

☑ Trim Surrounding Blank Space

Additional Padding: 0%

Foreground Scaling: Crop

Shape: None

Background Color:

5. Click next then finish.

Congratulations! You now have the most basic of basic apps.

“Hello World” First App

We can begin running the app from here, but before that, you should know of a few main directories and pages that form the App.

AndroidManifest.xml

The AndroidManifest file organizes your basic characteristics of the app and defines the background components that make up your app. Think of it as your “front page” defining the important aspects of your app.

src/

This is your directory for source files that your app will be using. Your app is divided into “Activities” that are written here. Think of activities as new “windows” for each instance of your app.

/res

Has several sub-directories that hold different app resources.

drawable-hdpi/

Stores your icons, images and other uploaded drawings or images to be used throughout your app.

layout/

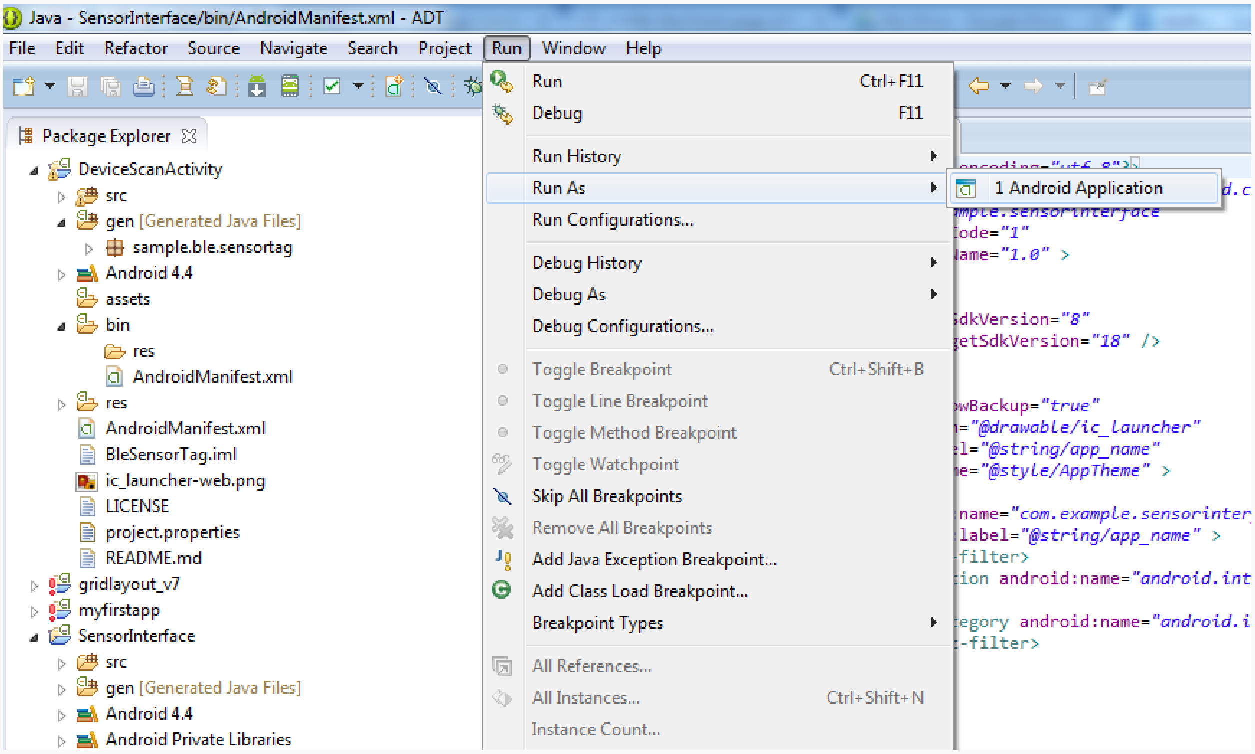
Files that describe how your app’s front end are to be laid out are stored here.

values/

Instead of declaring string or item values at the top of every page, you store them here for easy access to variable change.

Now it’s time to begin running your application on a real device. Before transferring the app, you need to make sure that USB debugging is enabled. To do this on most devices go to **Settings** then **Applications**, then to **Development** then check“USB Debugger”. If you are running Android 4.0 or higher, it’s in **Settings,** then to **Developer Options**. Also be aware that selecting USB Debugger enables you to install apps outside of the app store... so be careful! Malicious apps could be installed if you always have this box checked, so make sure to uncheck it once you are done working on your app.

To begin installing your app onto your phone, select **Run** from the menu, **Run As**, then select **Android Application,** then **Ok**.



```
encoding="utf-8"?>
                                d.c
mple.sensorinterface
Code="1"
Name="1.0" >

SdkVersion="8"
getSdkVersion="18" />

owBackup="true"
n="@drawable/ic_launcher"
el="@string/app_name"
me="@style/AppTheme" >

name="com.example.sensorinter
label="@string/app_name" >
-filter>
ion android:name="android.int

egory android:name="android.i
-filter>
```

Click Ok on the next window, and you should now have an app installed! Once clicking on the application icon, your app should open up a page saying “Hello World”. You can edit the text of the “Hello World” by editing the string.xml file found under the /res/values location. Simply clicking on the Hello_World string, then edit, you can change the value.We can begin adding the action bar now.

Adding the Action Bar

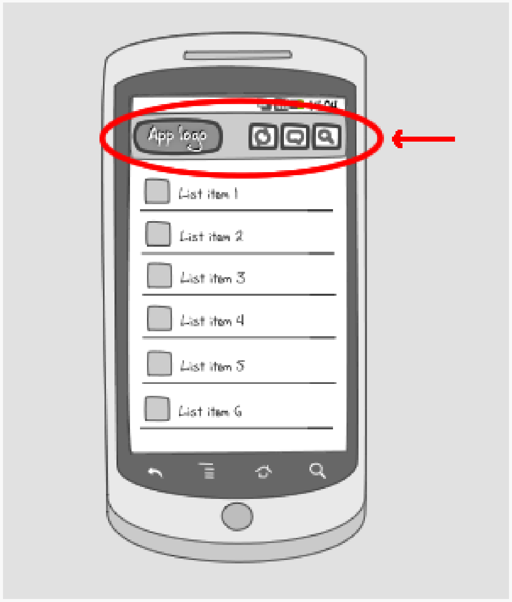


Image Courtisy of http://www.androidpatterns.com/

For details on how to add an action bar to Android versions below 3.0, visit:

http://developer.android.com/training

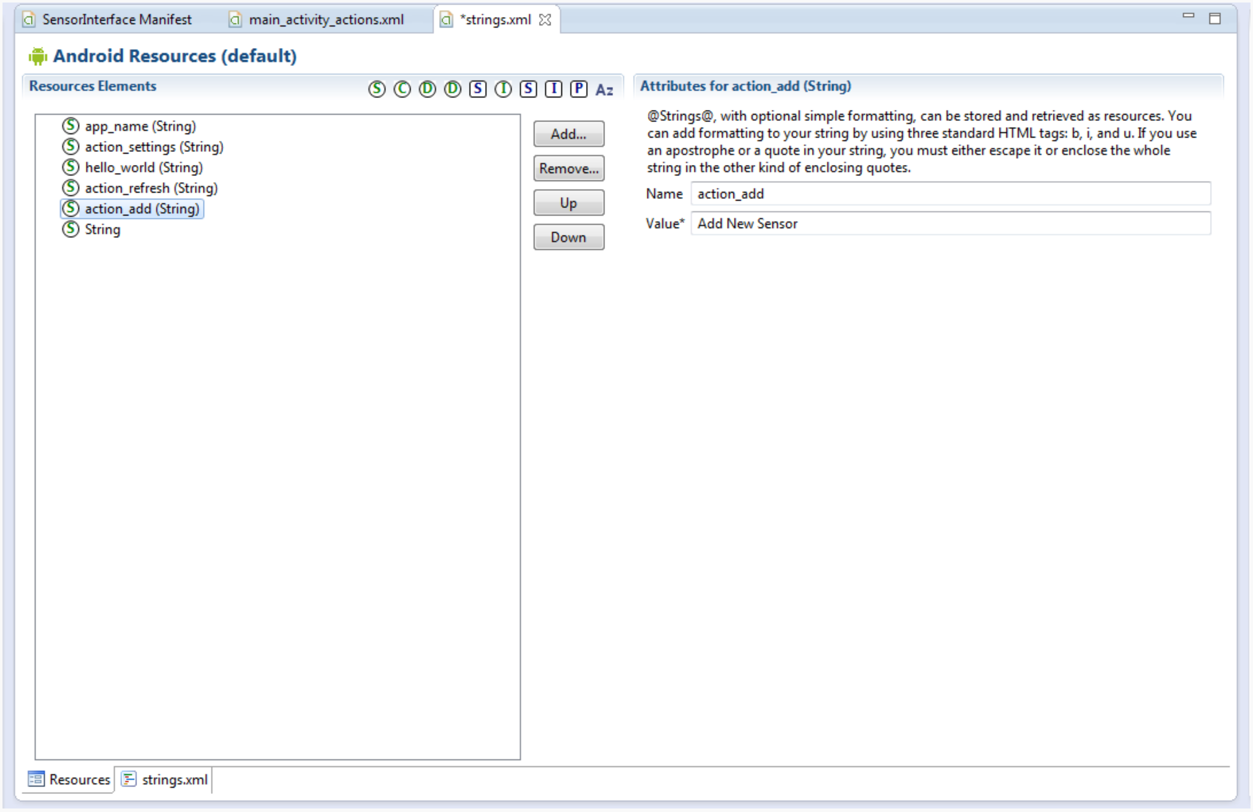
The action bar allows you add consistent functionality to all parts of your android app. You can add buttons to your action bar so that wherever a person is in your app, they can still use the action bar's main functions. You could add a home button, settings button, or whatever else you deem fit for your app. There are hundreds of stock Android buttons to use, as well as the ability to design your own. To download the Action Bar Icon Pack and many other addons for your app, visit:

https://developer.android.com/design/downloads/index.html\

Now we can begin adding buttons to your action bar. First, create a new source file by going to **File** then **New Source.** Name your new file "main_activity_actions.xml" and save it in the res/menu/ folder. Copy and paste these lines of code into your new document to add two new buttons to your action bar:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/action_sync"
          android:icon="@drawable/ic_action_refresh"
          android:title="@string/action_refresh"
          android:showAsAction="ifRoom" />
    <!-- Settings, should always be in the overflow -->
    <item android:id="@+id/action_new"
          android:icon="@drawable/ic_action_new"
          android:title="@string/action_add"
          android:showAsAction="ifRoom" />
</menu>
```

android:id specifies the unique resource of your button. android:icon is the file name of the icon you want to use, android:title is the title of the resource you are using and the android:showAsAction=”ifRoom” removes the buttons if there is not enough room in the action bar. This will add two buttons to your action bar, but it’s not quite done yet! You next need to go to /res/values/strings.xml to add your resources.Select **Add** then under the new window **Strings.**



- app_name (String)
- action_settings (String)
- hello_world (String)
- action_refresh (String)
- action_add (String)
- String

Attributes for action_add (String)

@Strings@, with optional simple formatting, can be stored and retrieved as resources. You can add formatting to your string by using three standard HTML tags: b, i, and u. If you use an apostrophe or a quote in your string, you must either escape it or enclose the whole string in the other kind of enclosing quotes.

Name action_add

Value* Add New Sensor

From here, name your string “action_add”, and a small description as a value. This will help later once your program becomes more complex. Do these same steps to add you action_refresh button.

Next we need to upload the files we are going to be using to the right directories. After finding the icons you want, move them to the /res/drawable folders. The hdpi-xxhdpi labels are about screen resolution. Different phones will use different images based on their screen resolution, so place 5 copies of your button image to these five drawable locations. Make sure their name matches with the android:icon, or the button won’t load. From here, you can load the app back on to your phone. And that’s it! You now have an action bar with clickable icons.

Conclution

By the end of this tutorial, the reader should be able to create an Android app under the Eclipse IDE environment. This app should display text, and the user should be able to alter this text to his or her liking. The app also includes two buttons from the Android Icon Library that the user can click on. Adding functionality to these buttons is beyond the scope of this tutorial, go to http://developer.android.com/index.html for greater detail

References

[1]Kerr, Dara. "Over Half of American Adults Own Smartphones." *CBSNews*. CBS Interactive, 6 June 2013. Web. 28 Mar. 2014. <http://www.cbsnews.com/news/over-half-of-american-adults-own-smartphones/>.

[2]"Action Bar." *Android Interaction Design Patterns*. N.p., n.d. Web. 28 Mar. 2014. <http://www.androidpatterns.com/uap_pattern/action-bar>.