

Android Development: Building Your App

Al-Kandari, AbdulMuhsin

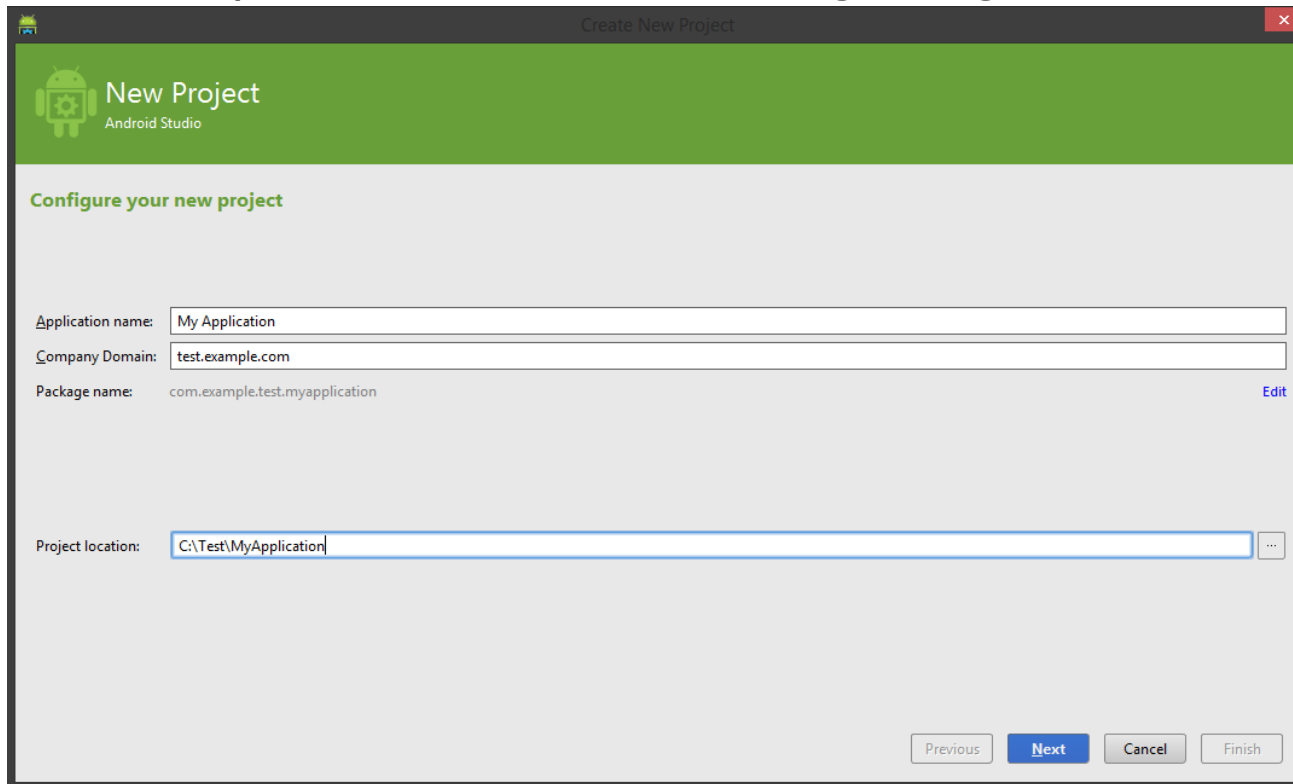
Blessing, James

Setup: Getting the Right Tools for the Job

- ▶ Choose the IDE or Text Editor that you would like to develop android with. (We will be using Android Studio for these workshops. Skip to the last step if you are using Android Studio.)
- ▶ Check to see if the IDE you have chosen already has the android SDK, if not then download the SDK.
<http://developer.android.com/sdk/installing/index.html?pkg=adt>
- ▶ For Eclipse users make sure to install the ADT Plugin.
- ▶ Download the latest SDK tools and platforms using the SDK Manager.

Starting Off: Creating Your Project

- ▶ When you start Android Studio you are prompted to choose a pre-existing project or to create a new project.
- ▶ If you are not presented with a prompt go to File → New Project.
- ▶ You will be presented with the following dialog box.



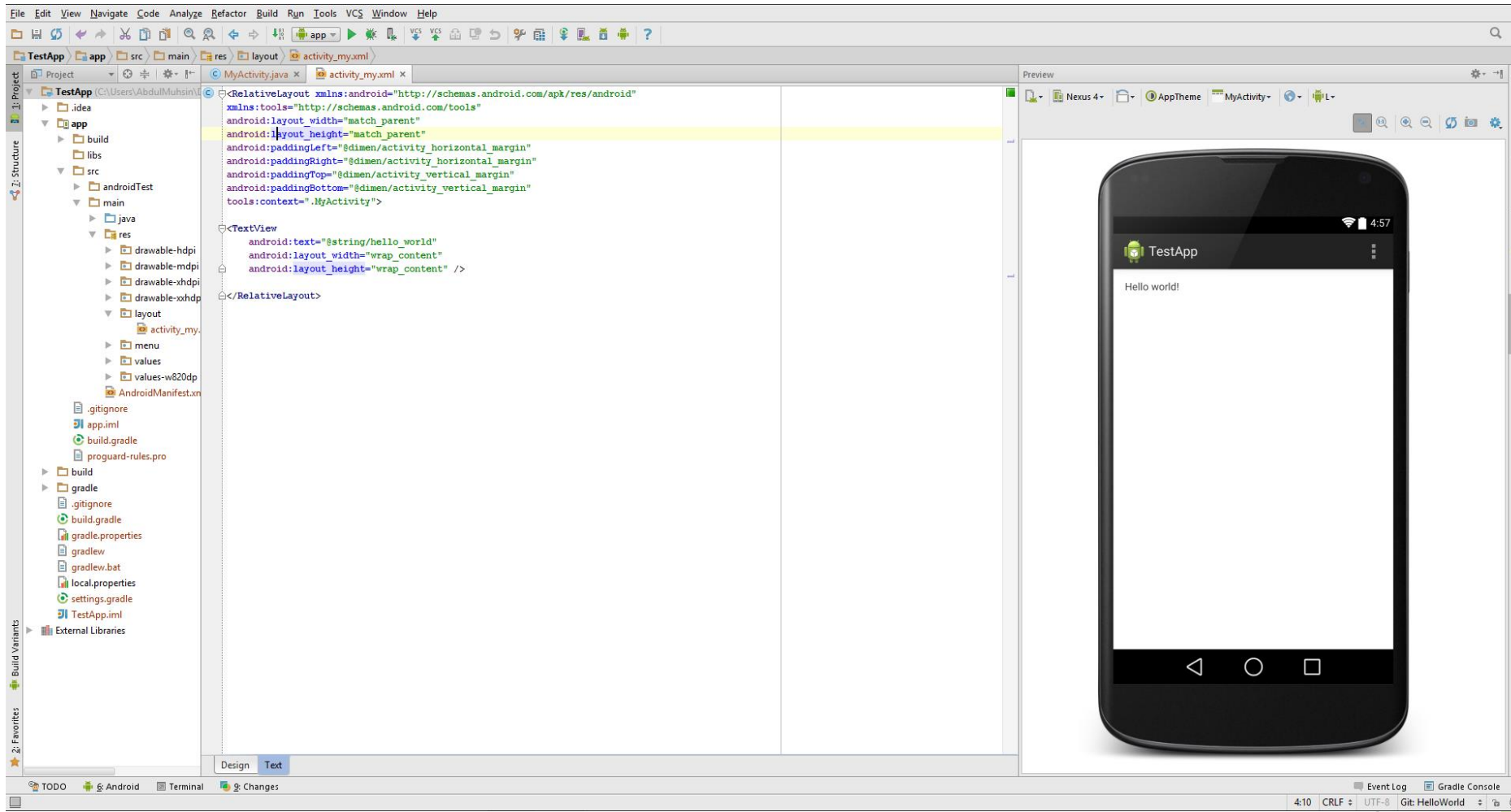
Continued:

- ▶ Choose the Application name, henceforth referred to as *app_name* in any code. Also choose the company domain and project location that suits you best.
- ▶ Choose the minimum SDK that you would like your app to support (remember that this is a give and take choice, as newer SDKs will present more features but will allow for less users and vice-versa.)
- ▶ You are then presented with a choice of activities to add to your project.
- ▶ For the purpose of this lesson we shall choose Blank Activity
- ▶ Another dialog, again choose the activity name and be aware that changing the name of your activity will change the layout name. (This will be referred to as *activity_name*.)

Source Code Control: Committing Changes Through Android Studio

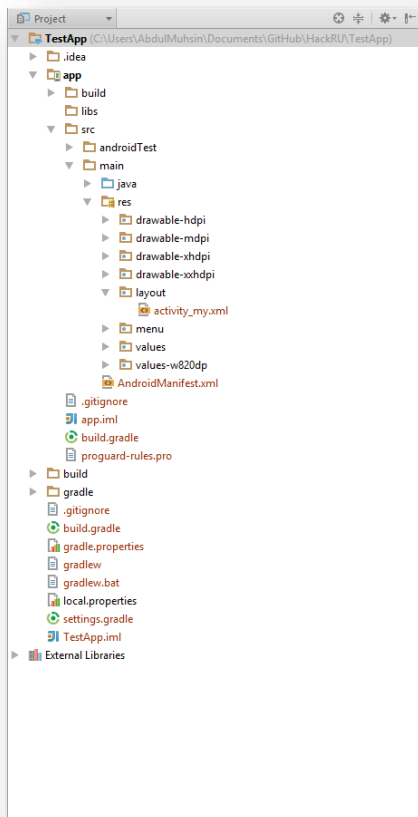
- ▶ Android Studio offers an easy method to connect to a version control service (Github for example) this feature can prove beneficial and should be taken advantage of.
- ▶ In Android Studio, on the menu bar select
VCS → Enable Version Control Integration → Git
VCS → Git → Add
- ▶ This will allow you to update your project to the latest version, and it will allow you to commit and push your changes to a branch of choice on your repo.
VCS → Update Project → OK
VCS → Commit Changes → Commit
- ▶ To change the branch you are working on you can:
VCS → Git → Branches

“What You See Is What You Get” (WYSIWYG)

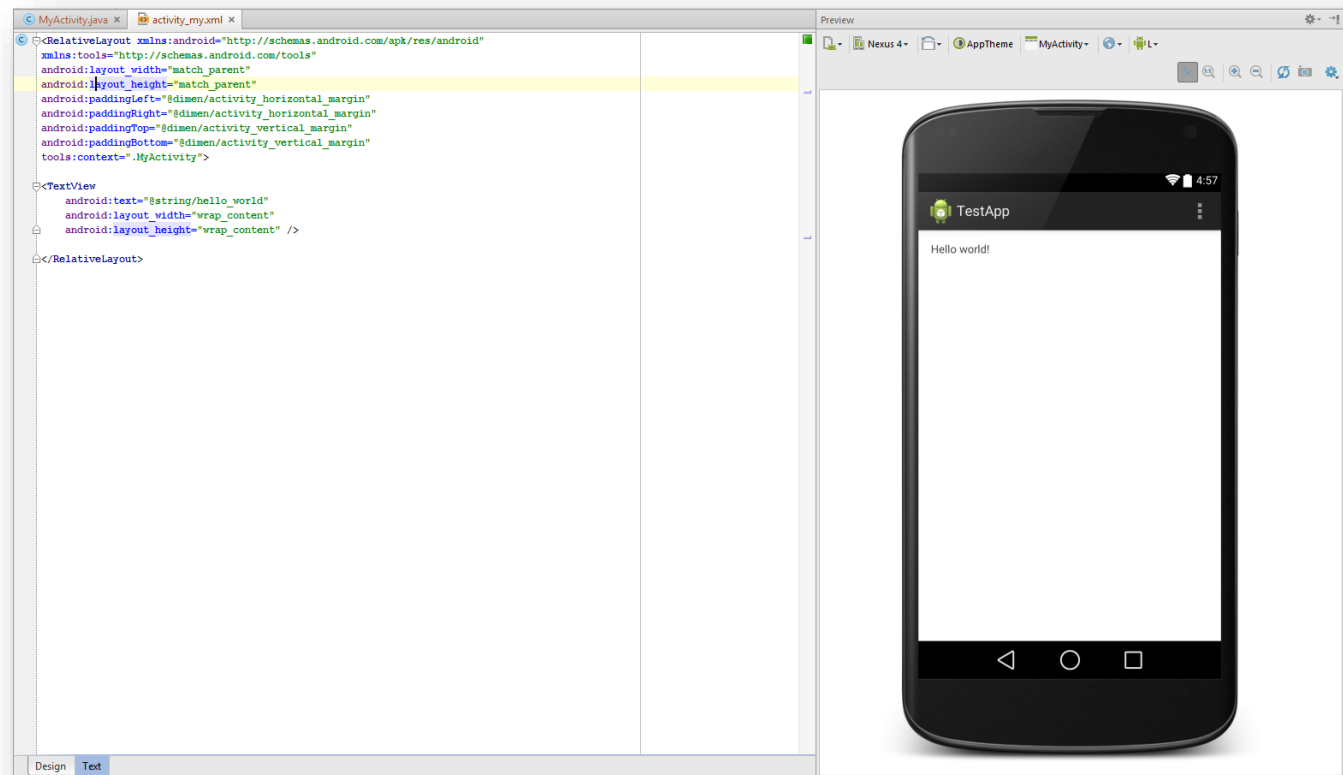


Parts of The Screen

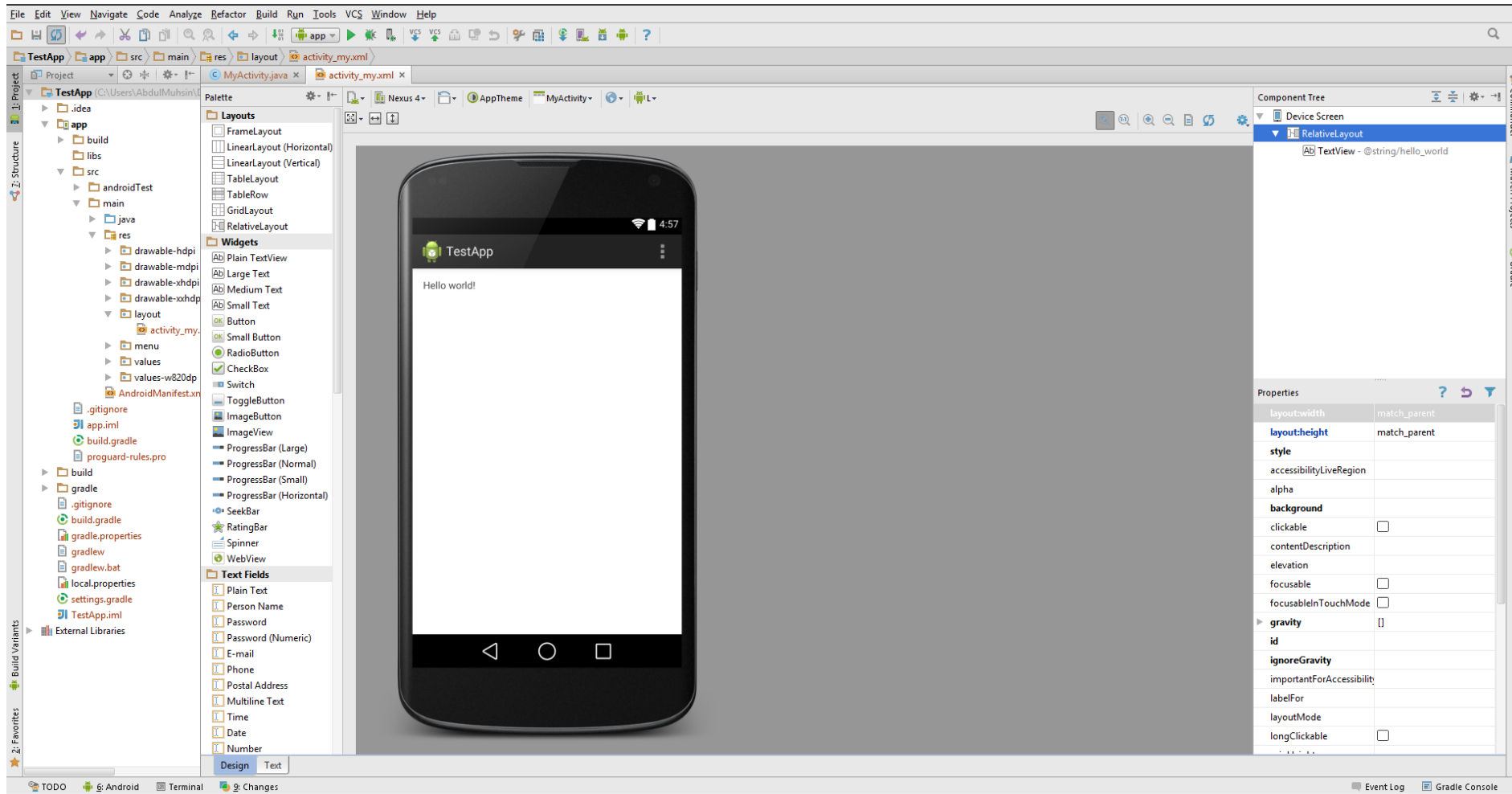
File Manager



Editor



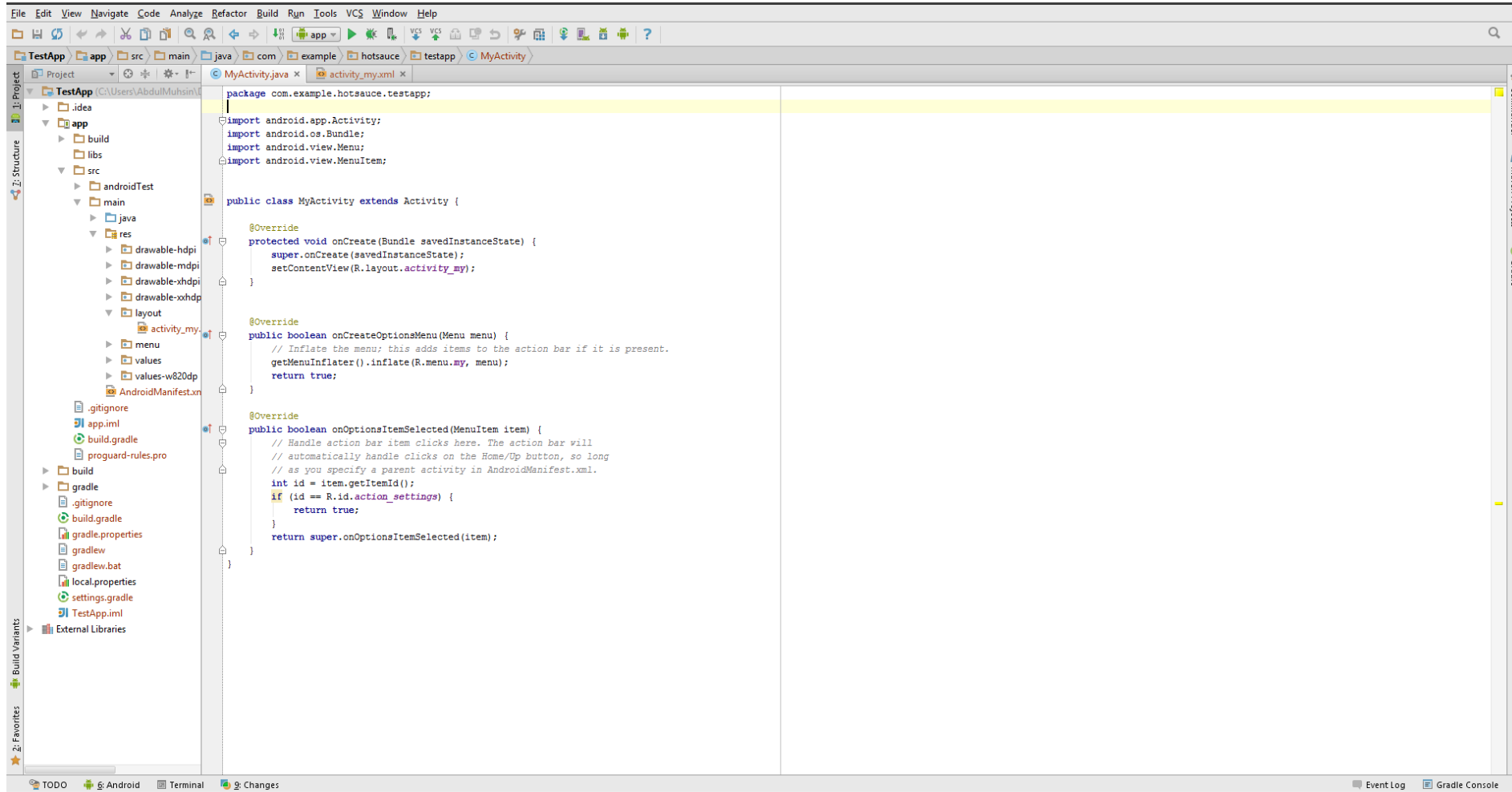
Design View of XML File



XML File: Importance & Tips

- ▶ The XML file displayed in the previous slides manipulates the UI, familiarize yourself with it because you'll be spending a lot of time perfecting your design.
- ▶ As displayed above XML files can be edited through the Design View or the Text View.
- ▶ In order to familiarize yourself with the different XML objects, tags and attributes that you can add in android, it's recommended to start out with the design view.
- ▶ The Text View allows for a higher level of control and is considered more advanced

View of Java Activity File



The screenshot displays an IDE interface with a project structure on the left and a Java file editor in the center. The project structure shows a hierarchy starting from 'TestApp' down to 'MyActivity'. The Java file 'MyActivity.java' is open, showing the following code:

```
package com.example.hotsauce.testapp;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MyActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.my, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

The IDE interface includes a menu bar at the top with options like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The bottom status bar shows tabs for TODO, Android, Terminal, and Changes, along with Event Log and Gradle Console.

Java Activity File: Importance & Tips

- ▶ The Java Activity file displayed in the previous slide is used to manipulate the activity's XML file.
- ▶ Although java has the ability to add UI elements to the XML file from the Activity, one should abstain from doing so as it creates extra runtime overhead.
- ▶ Remember each Java Activity File can be used for navigation to other activities using an *Intent* which will be discussed later in this presentation.
- ▶ An Activity can also give life to a fragment using an inflater, but can take life from a fragment using a deflator.
- ▶ Also as the MVP architecture states the model has no connection to the view without the presenter and the Activity can be considered the Presenter.

View of Android Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.hotsauce.testapp" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="TestApp"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MyActivity"
            android:label="TestApp" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Android Manifest File: Importance & Tips

- ▶ The android manifest file is the application. This file holds the references to all of the activities in the application.
- ▶ All <activity> tags have to be included in this file within the application node, otherwise the application will be unaware of the activity's existence.
- ▶ The manifest is also where you declare the activity's label and icon.



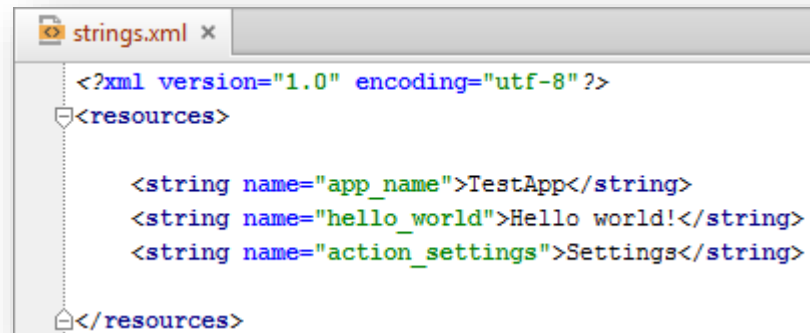
- ▶ Note: Within the manifest file is where the main (starting) activity is declared.

Resources: What Are They?

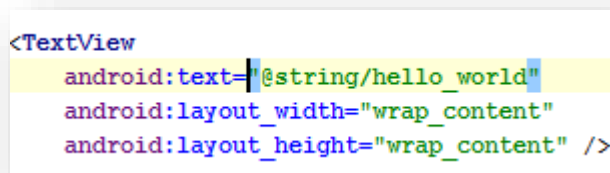
- ▶ Resources (values) are an element of the XML part of android, they act as variables for the XML files and can be accessed using the following format : *@resourceFileName/resourceName*
- ▶ *Note: Common resource types are dimensions, strings, and styles*
- ▶ Example:



The current resource files



The current string resources



The resource call in the name_activity.xml file

Resources: Why use them?

- ▶ Using resources is better than hard coding values because:
 - ▶ It allows for the change of all occurrences of the resource in question at once, instead of having to look for every occurrence in the different files (this is especially useful on larger projects)
 - ▶ It allows for the support of multiple languages as a separate strings.xml file can be used for each supported language
 - ▶ It creates an easily accessible area to look at all of your values, no matter what type they are.
 - ▶ It is considered to be a part of the best practices in android development.
- ▶ Note: *The creation of new resource files is possible and is supported by the android SDK. An example of this may be a color.xml file that holds your different color choices.*

Summary: What We Have Covered

- ▶ The creation of a new android project
- ▶ Linking our project to a VCS (Version Control System)
- ▶ The WISYWYG Editor
- ▶ The different files in the Android project, their uses and their importance
- ▶ The importance of resource files.