Android in 90 Minutes

Geoff Matrangola geoff@matrangola.com

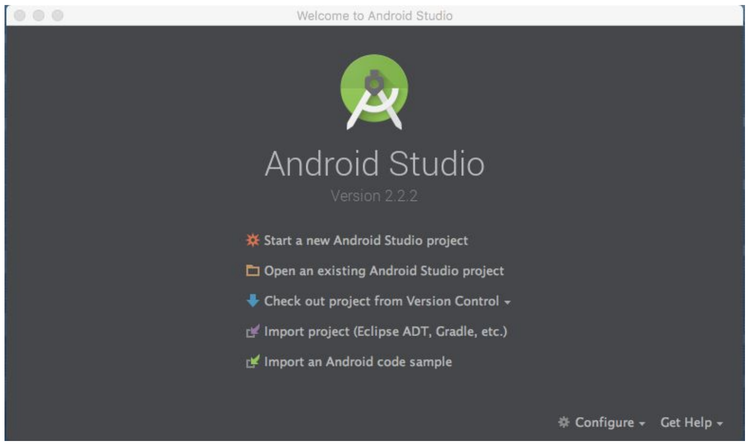
Download the App

Download and install: https://developer.android.com/studio

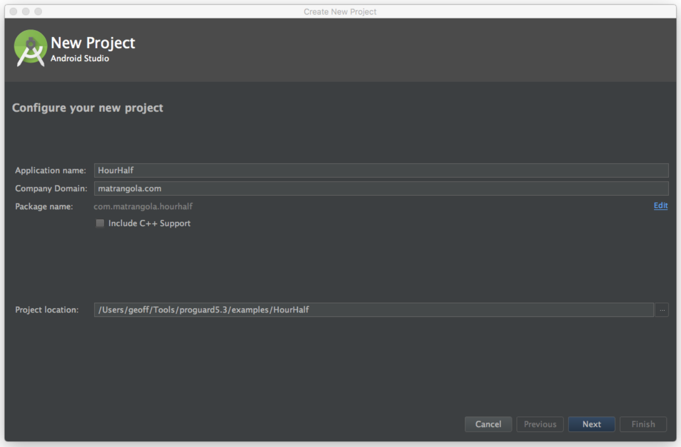
Explore: https://developer.android.com/studio/intro/index.html

Intro	5 Minutes
Android Studio Tour, Virtual Device, Create Project	5 Minutes
Lab 1 - Create Project	10 Minutes
Beyond Hello World - Label and Button, findViewById, onClickListener	5 Minutes
Lab 2 - Beyond Hello World	10 Minutes
Download Weather - Add commons-io dependency, Create IntentService, download JSON string	20 Minutes
Lab 3 - Call IOUtil.toString(URL)	10 Minutes
Send and Receive Broadcast Intent	20 Minutes
Lab 4 - Send and Receive Broadcast Intent	10 Minutes

Start a new Android Studio Project


Welcome to Android Studio
Android Studio
Version 2.2.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

Choose Metadata



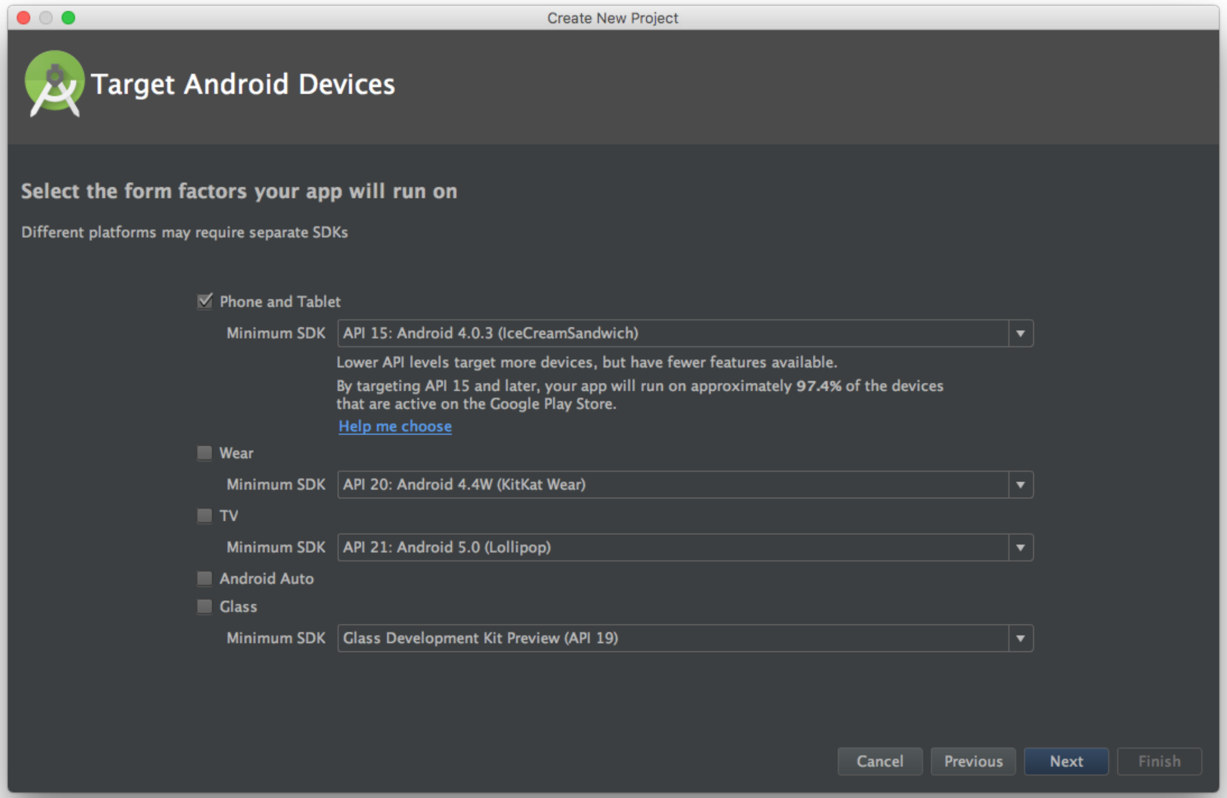
Configure your new project

Application name: HourHalf
Company Domain: matrangola.com
Package name: com.matrangola.hourhalf Edit
Include C++ Support

Project location: /Users/geoff/Tools/proguard5.3/examples/HourHalf

1. Application name: HourHalf
2. Company Domain: yourdomain.com
3. Package location: *You Decide*

Target Android Devices



Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet
Minimum SDK API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.
Help me choose

Wear
Minimum SDK API 20: Android 4.4W (KitKat Wear)

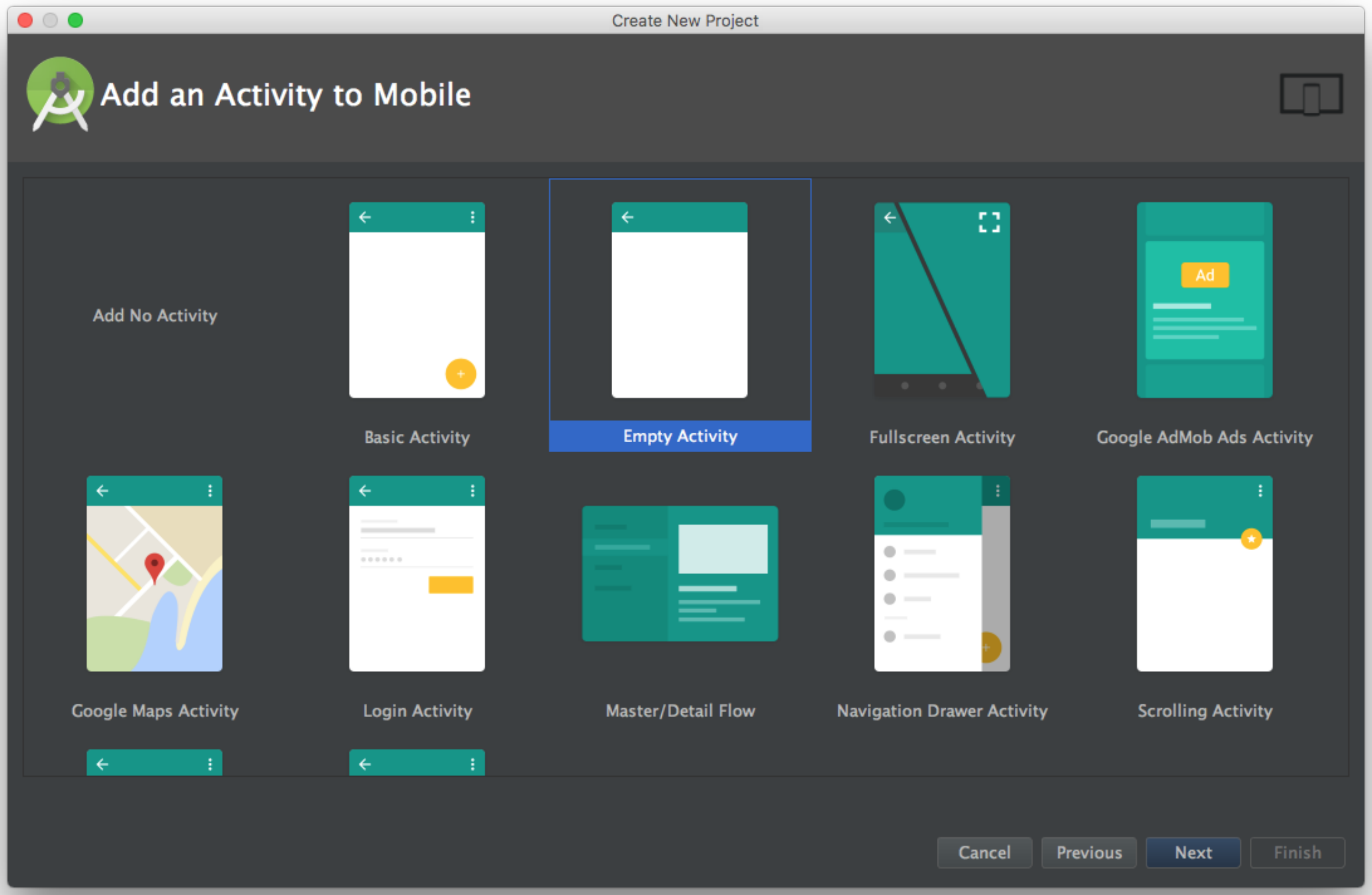
TV
Minimum SDK API 21: Android 5.0 (Lollipop)

Android Auto

Glass
Minimum SDK Glass Development Kit Preview (API 19)

1. Select Phone and Tablet
2. Minimum SDK API 15: ANdroid 4.0.3
3. Explore Help me Choose - Android Dashboards
4. Support Library

Add an Activity to Mobile

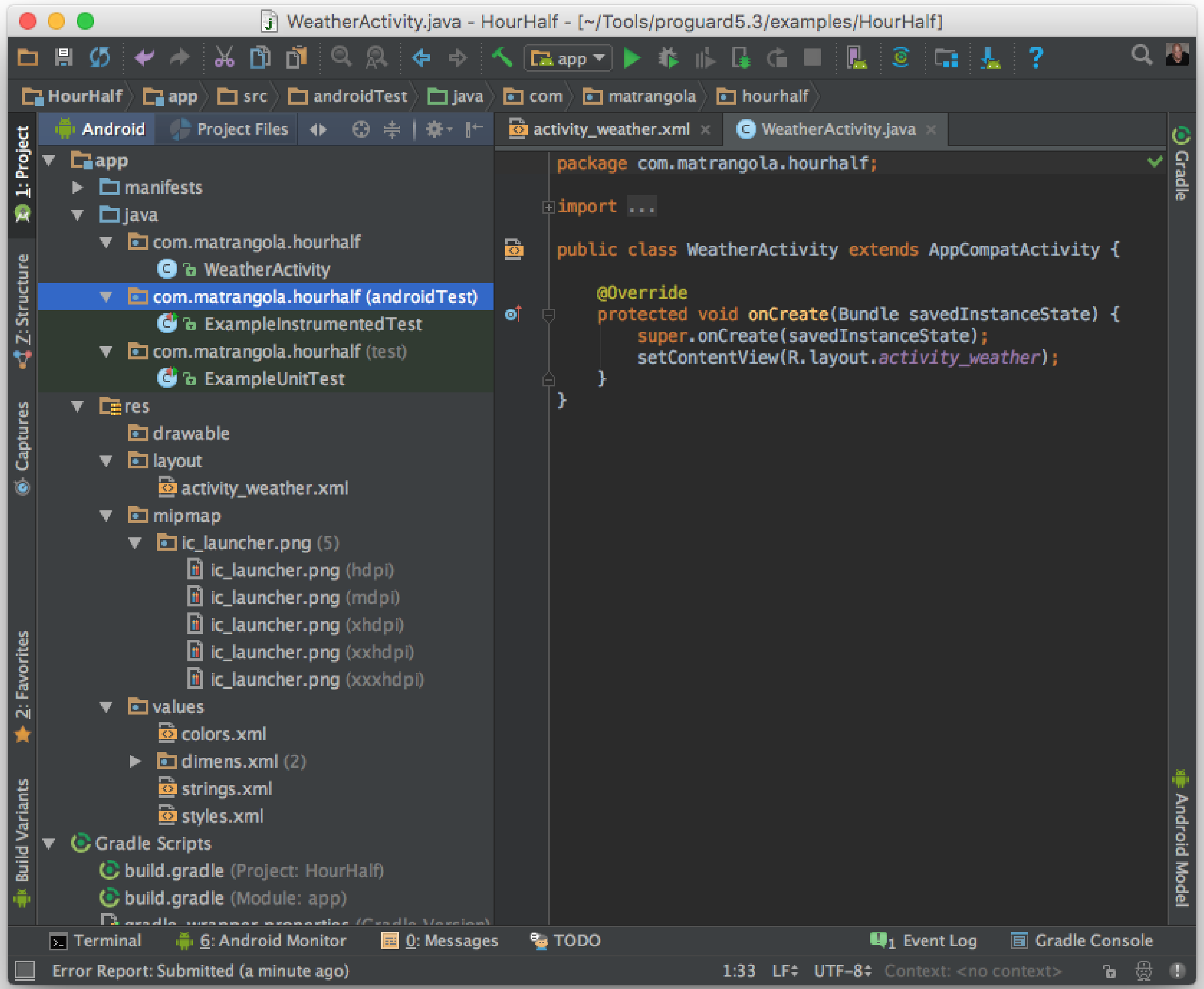


1. Empty Activity
2. Next

Customize the Activity

1. Activity Name: WeatherActivity
2. Layout Name: activity_weather
3. Generate Layout File
4. Backwards Compatibility (AppCompat)

Explore the Template App

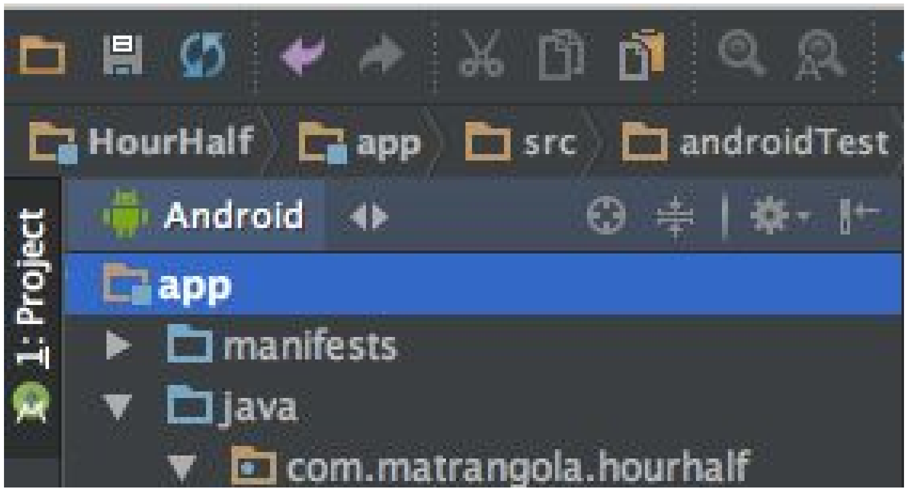


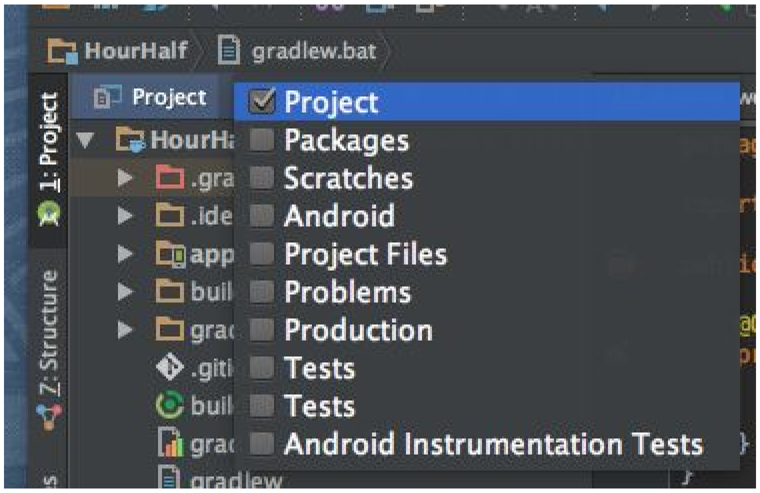
```java
package com.matrangola.hourhalf;

import ...

public class WeatherActivity extends AppCompatActivity {

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

Error Report: Submitted (a minute ago)

1. Initial View
   a. WeatherActivity.java - Activity Class
   b. actiivty_wehater.xml - default layout
   c. File Explorer

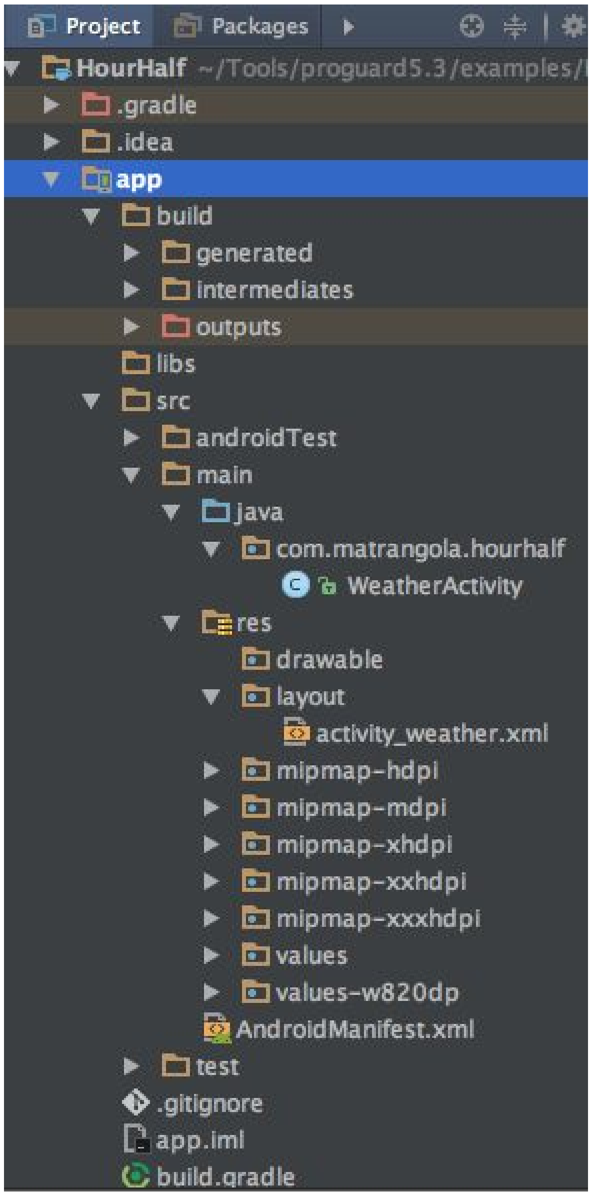
Switch to Project View



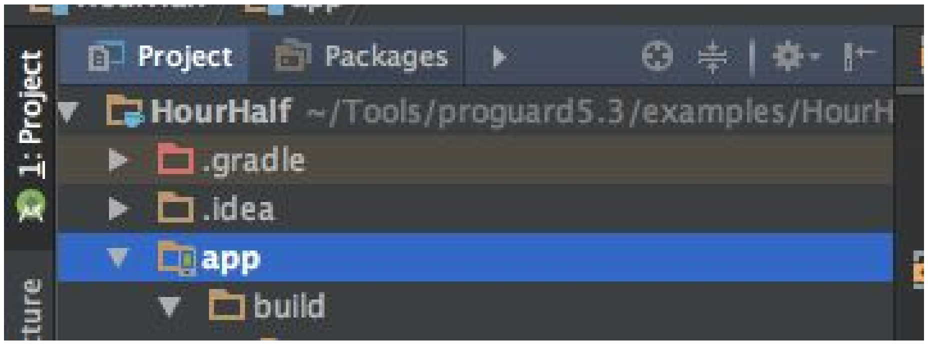


1. Explorer Layout
   a. Android - Default - Follows app file hierarchy layout
   b. Project - Preferred - Follows full file hierarchy

Source and Layout Directory Structure



```
Project   Packages
▼ HourHalf ~/Tools/proguard5.3/examples/
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▼ build
      ▶ generated
      ▶ intermediates
      ▶ outputs
      libs
    ▼ src
      ▶ androidTest
      ▼ main
        ▼ java
          ▼ com.matrangola.hourhalf
              WeatherActivity
        ▼ res
            drawable
          ▼ layout
              activity_weather.xml
          ▶ mipmap-hdpi
          ▶ mipmap-mdpi
          ▶ mipmap-xhdpi
          ▶ mipmap-xxhdpi
          ▶ mipmap-xxxhdpi
          ▶ values
          ▶ values-w820dp
          AndroidManifest.xml
      ▶ test
    .gitignore
    app.iml
    build.gradle
```



```
1: Project
Project   Packages
▼ HourHalf ~/Tools/proguard5.3/examples/HourH
  ▶ .gradle
  ▶ .idea
  ▼ app
    ▼ build
```

1. Directory Structure is based on Build Tool, Gradle
2. Gradle uses the Convention over Configuration
3. Gradle uses Maven Convention
4. Can have multiple Apps in a single Project.
5. Project can contain Android Apps, Android Wear Apps, Android Libraries, Java Library Source, Build Configuration, Tests, C++ Native (JNI) libraries, etc.
6. Use the Target icon to find your file's location in the tree view.
7. Java Source lives under Project/app/src/main/java/com/company/package/ClassName.java

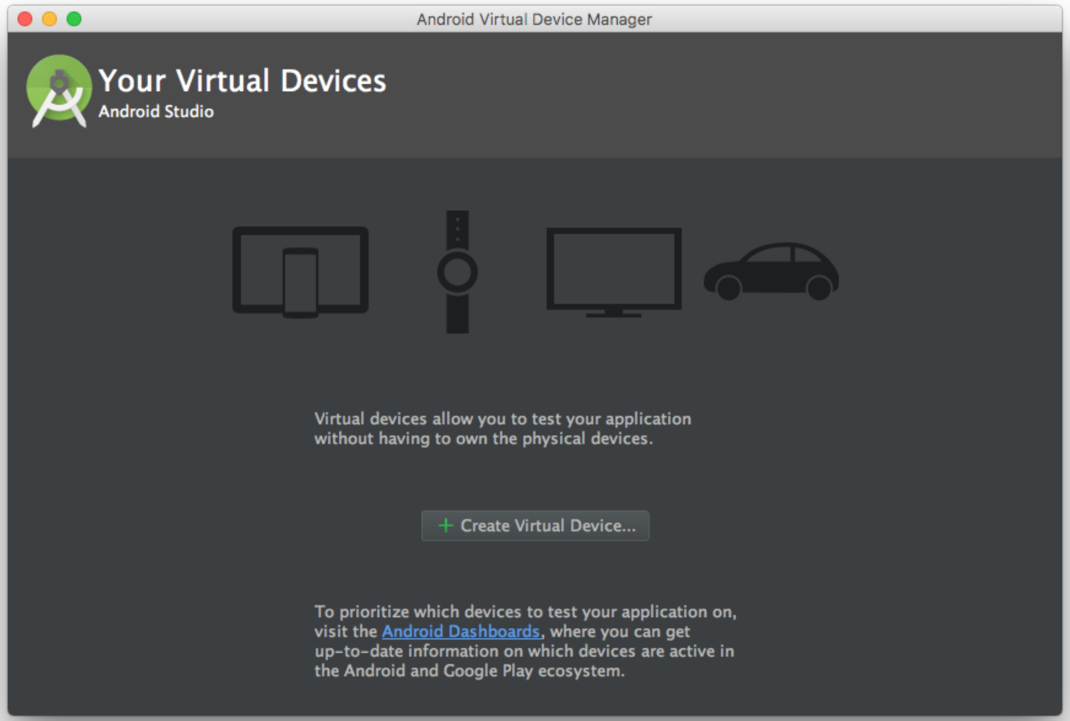
8. Layout is in Project/app/src/main/res/layout/activity_name.xml

Android Virtual Device

Reference: https://developer.android.com/studio/run/managing-avds.html

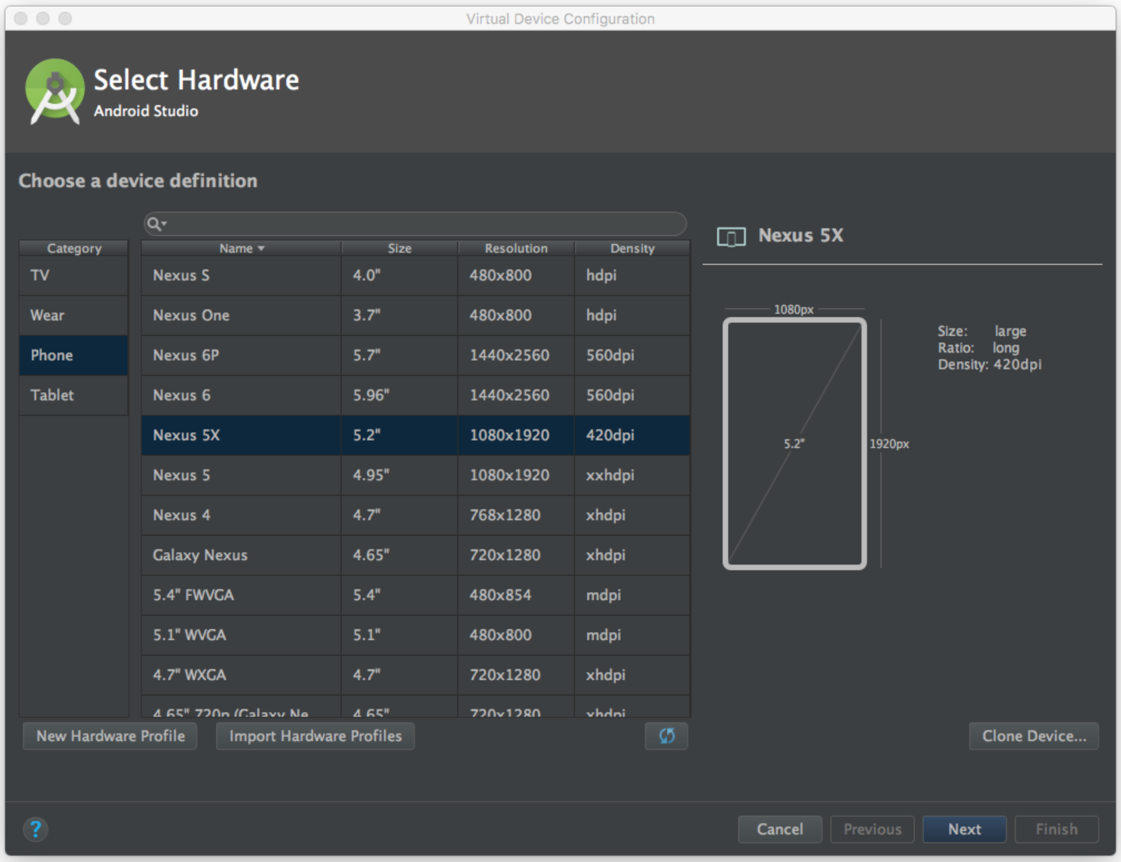
- Create Virtual Device for different OS versions, Screen Sizes, etc
- Helps with Automated Unit Testing
- Use to test GPS
- Huge improvements over last year
- Faster, More Reliable
- Still need to test on real device
- Laptop battery consumption

Create Virtual Device



Virtual devices allow you to test your application without having to own the physical devices.

To prioritize which devices to test your application on, visit the Android Dashboards, where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.



Choose a device definition

Category	Name	Size	Resolution	Density
TV	Nexus S	4.0"	480x800	hdpi
Wear	Nexus One	3.7"	480x800	hdpi
Phone	Nexus 6P	5.7"	1440x2560	560dpi
Tablet	Nexus 6	5.96"	1440x2560	560dpi
	Nexus 5X	5.2"	1080x1920	420dpi
	Nexus 5	4.95"	1080x1920	xxhdpi
	Nexus 4	4.7"	768x1280	xhdpi
	Galaxy Nexus	4.65"	720x1280	xhdpi
	5.4" FWVGA	5.4"	480x854	mdpi
	5.1" WVGA	5.1"	480x800	mdpi
	4.7" WXGA	4.7"	720x1280	xhdpi
	4.65" 720p (Galaxy Ne	4.65"	720x1280	xhdpi

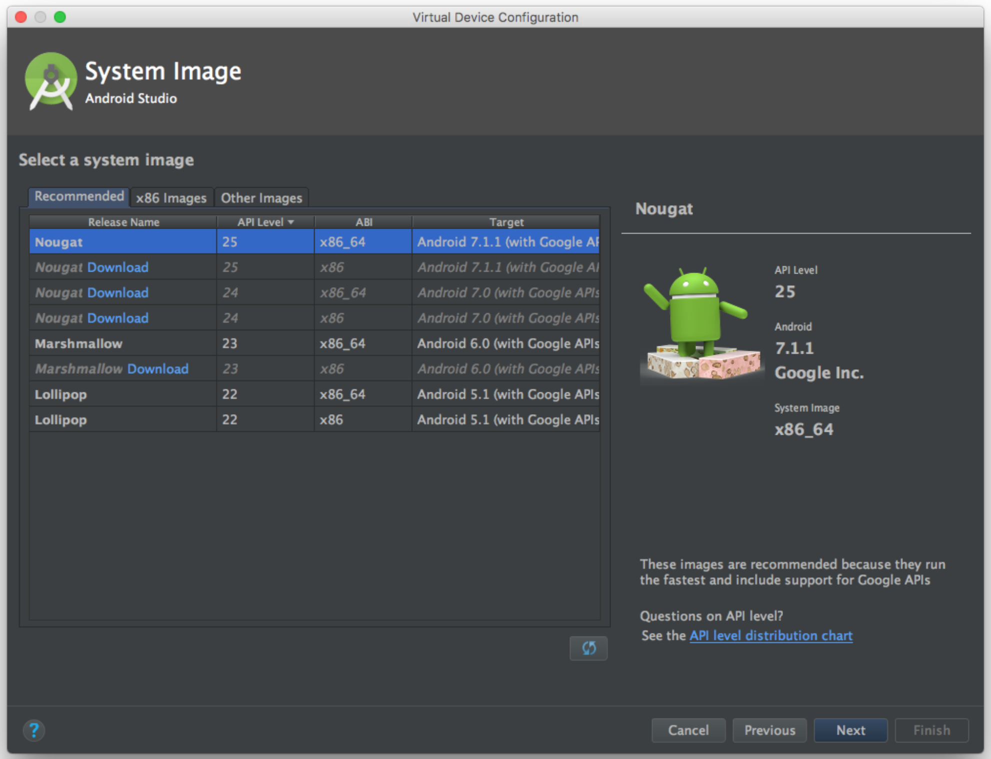
Nexus 5X

Size: large
Ratio: long
Density: 420dpi

For Development-flow chose modest options with latest API

Choose Phone -> Nexus 5X

Download Before Class, or use latest API available for your architecture.



Select a system image

Release Name	API Level	ABI	Target
Nougat	25	x86_64	Android 7.1.1 (with Google AI
Nougat Download	25	x86	Android 7.1.1 (with Google AI
Nougat Download	24	x86_64	Android 7.0 (with Google APIs
Nougat Download	24	x86	Android 7.0 (with Google APIs
Marshmallow	23	x86_64	Android 6.0 (with Google APIs
Marshmallow Download	23	x86	Android 6.0 (with Google APIs
Lollipop	22	x86_64	Android 5.1 (with Google APIs
Lollipop	22	x86	Android 5.1 (with Google APIs

Nougat

API Level
25

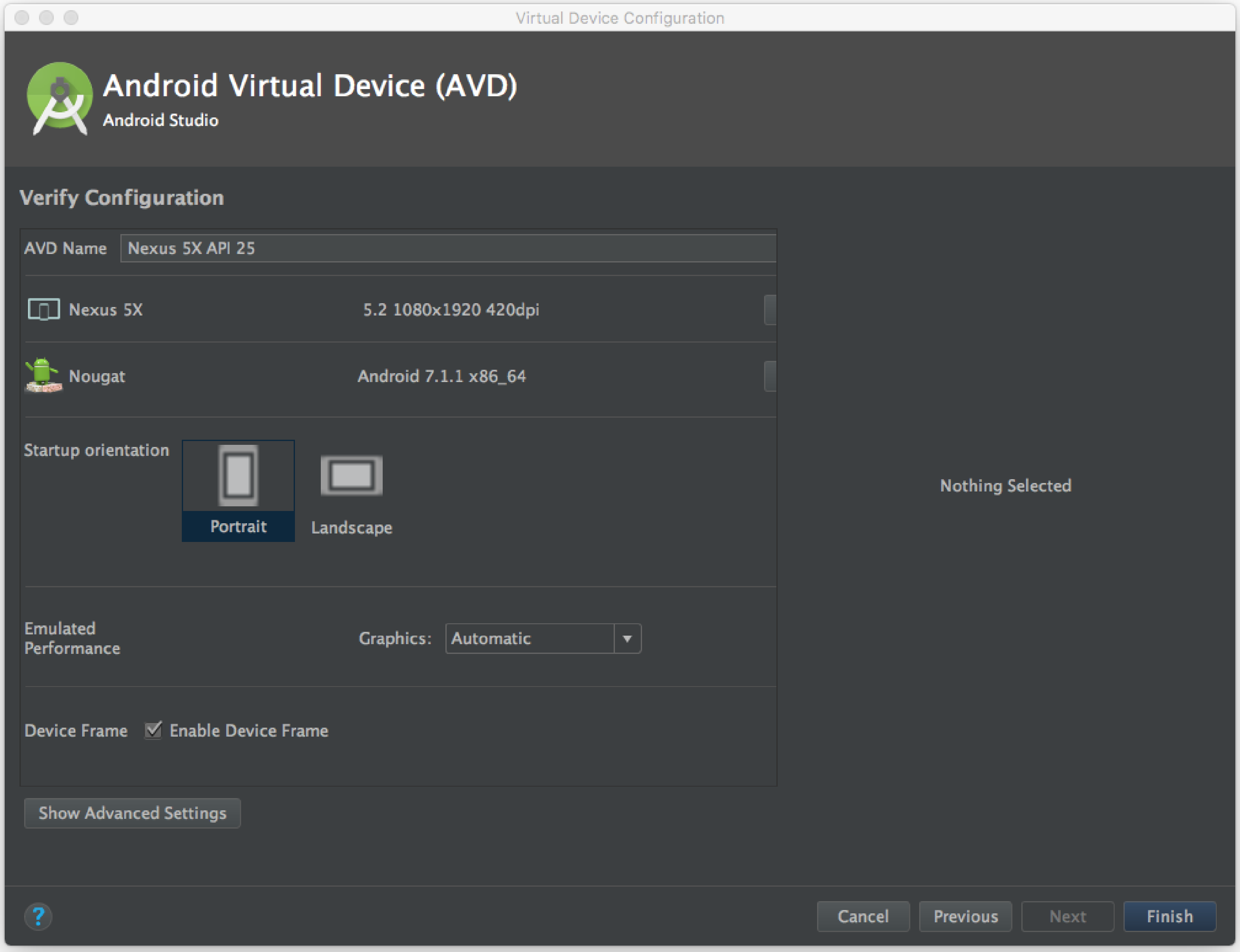
Android
7.1.1
Google Inc.

System Image
x86_64

These images are recommended because they run the fastest and include support for Google APIs

Questions on API level?
See the API level distribution chart

Android Virtual Device (AVD)



Verify Configuration

AVD Name: Nexus 5X API 25

Nexus 5X — 5.2 1080x1920 420dpi

Nougat — Android 7.1.1 x86_64

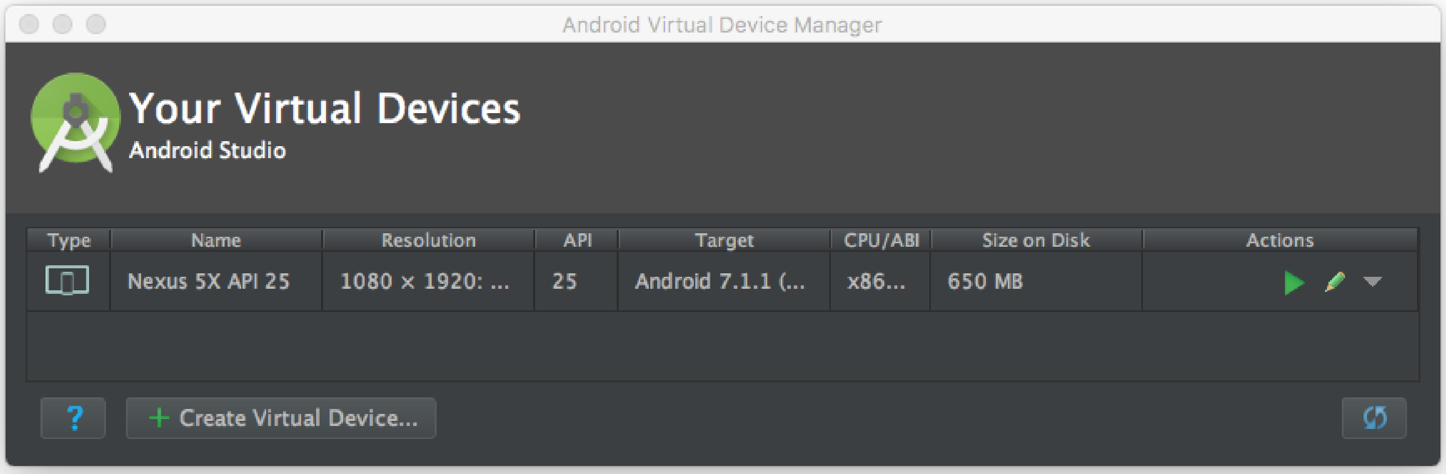
Startup orientation: [✓ Portrait] Landscape

Emulated Performance — Graphics: Automatic

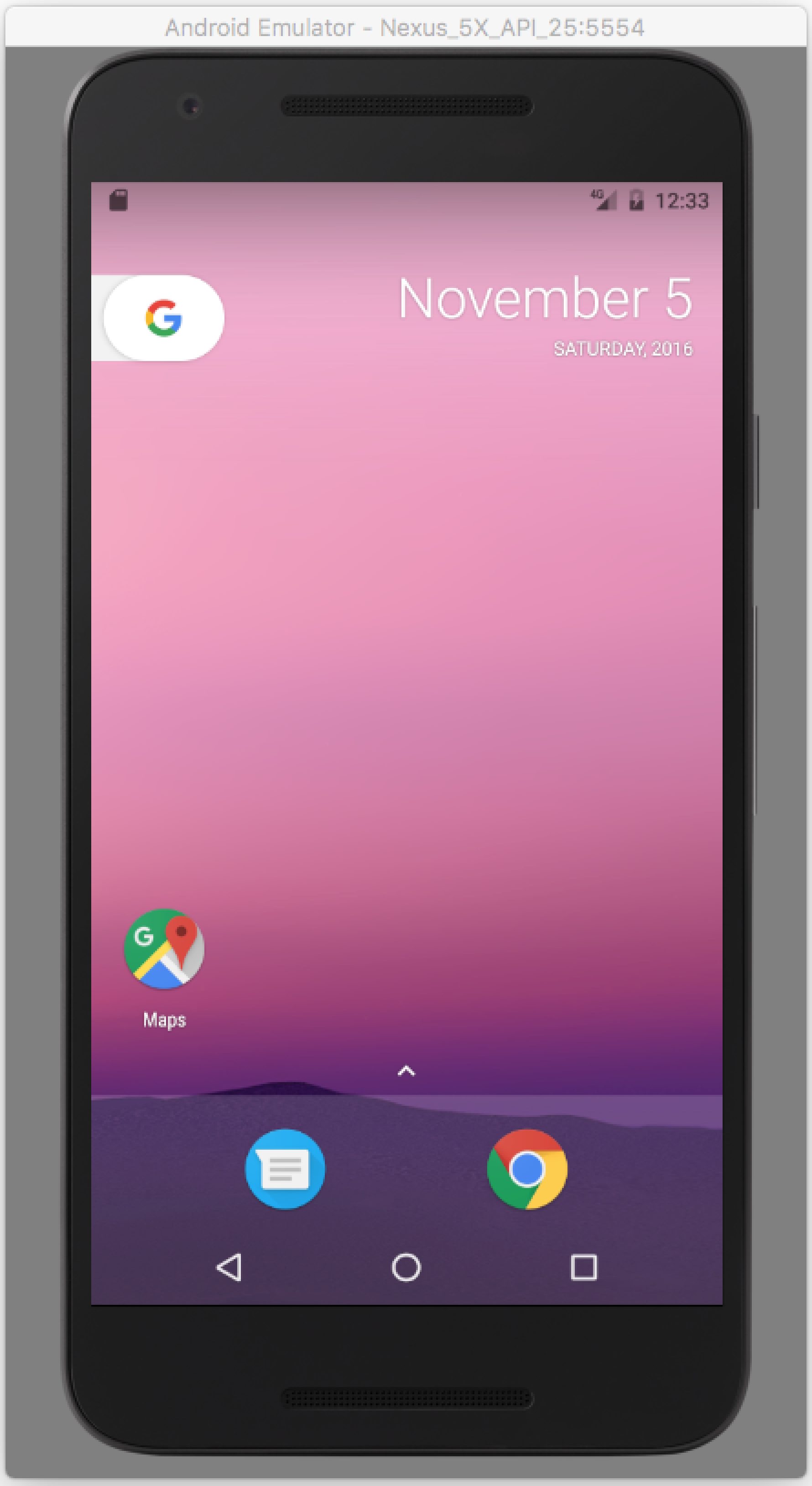
Device Frame: ✓ Enable Device Frame

Leave Defaults and Finish

Start the Virtual Device

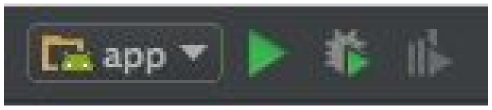


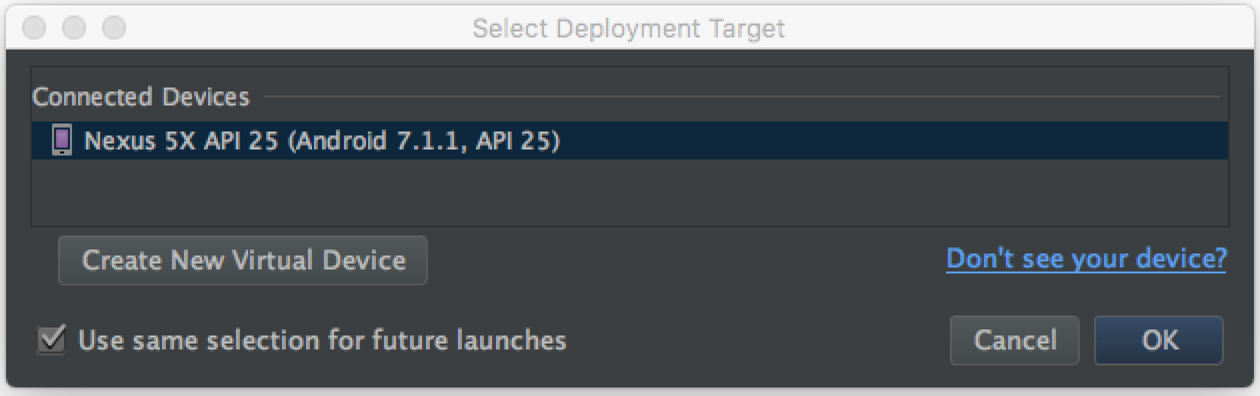
Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 25	1080 × 1920: ...	25	Android 7.1.1 (...	x86...	650 MB	



Leave Running

Start the app

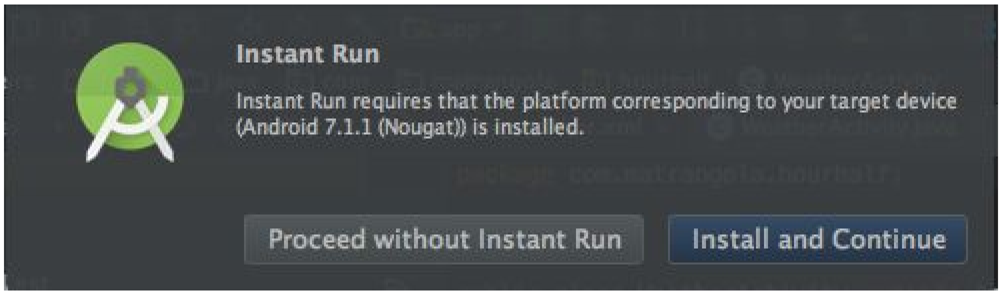



Select Deployment Target

Connected Devices

Nexus 5X API 25 (Android 7.1.1, API 25)

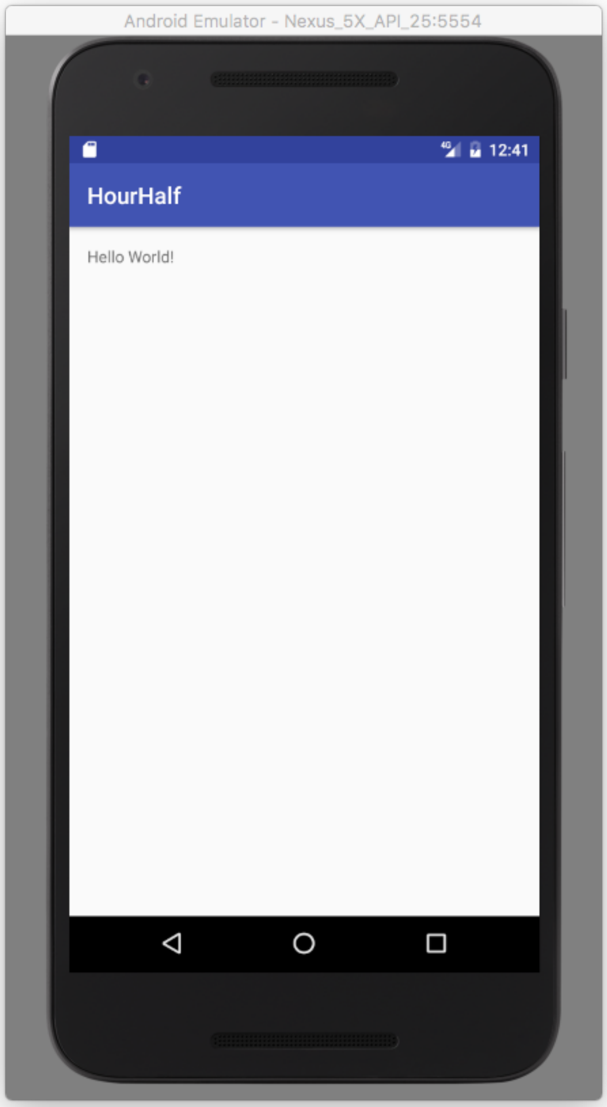
Create New Virtual Device
Don't see your device?
Use same selection for future launches
Cancel
OK


Instant Run

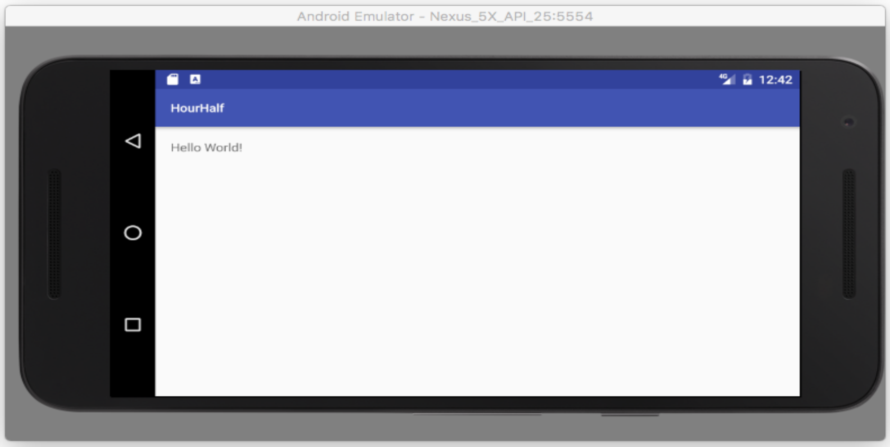
Instant Run requires that the platform corresponding to your target device (Android 7.1.1 (Nougat)) is installed.

1. Select Development Target
    a. Choose Nexus 5X that you just created
    b. Confirm Use same selection for future launches is checked
    c. To save Lab time, you can proceed without **Instant Run** if prompted

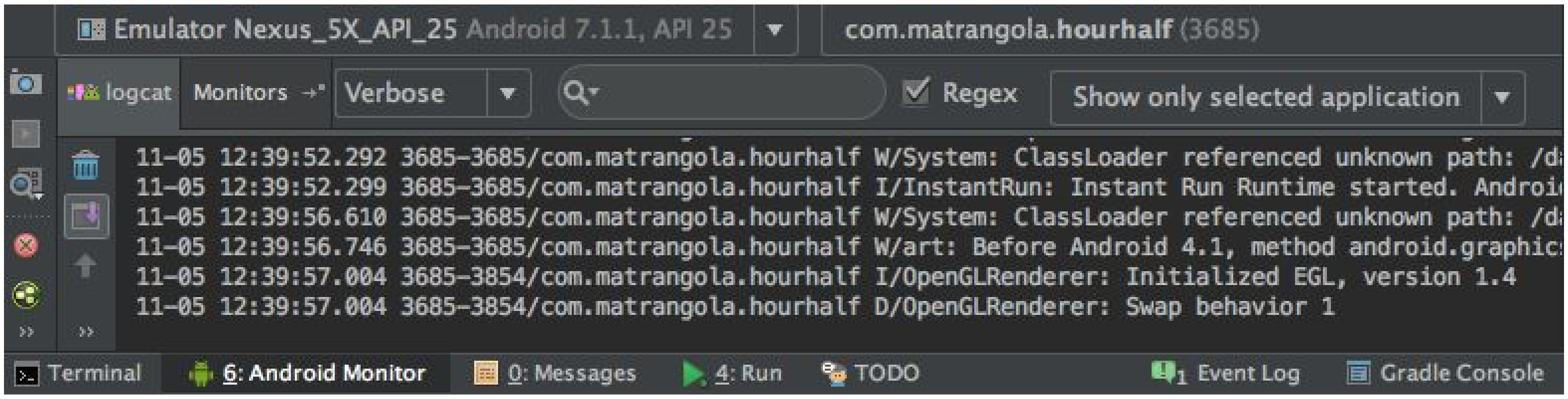
Hello World



Hello World!



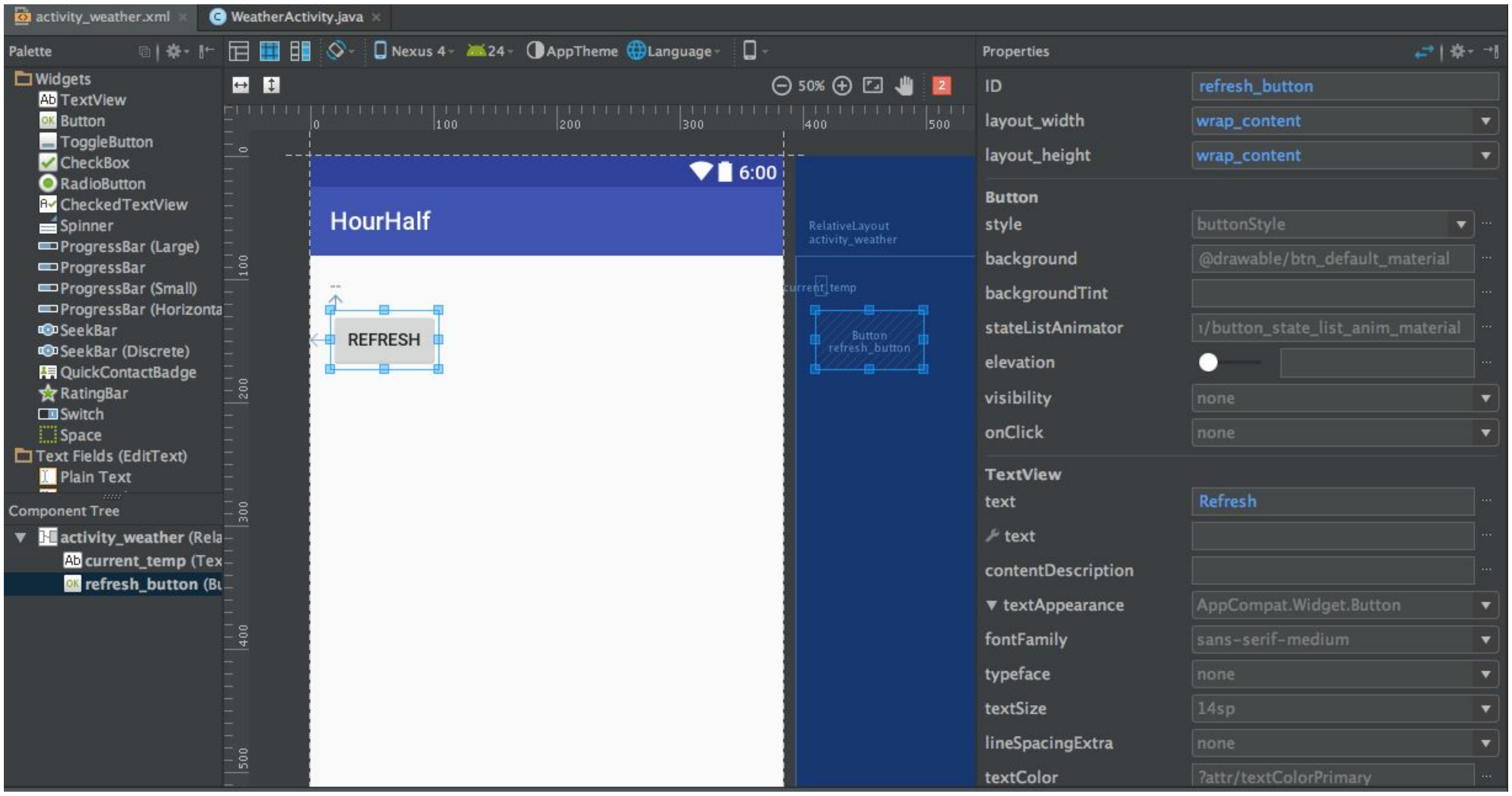
Hello World!


Emulator Nexus_5X_API_25 Android 7.1.1, API 25
com.matrangola.hourhalf (3685)
logcat
Monitors
Verbose
Regex
Show only selected application
Terminal
6: Android Monitor
0: Messages
4: Run
TODO
Event Log
Gradle Console

```
11-05 12:39:52.292 3685-3685/com.matrangola.hourhalf W/System: ClassLoader referenced unknown path: /da
11-05 12:39:52.299 3685-3685/com.matrangola.hourhalf I/InstantRun: Instant Run Runtime started. Androi
11-05 12:39:56.610 3685-3685/com.matrangola.hourhalf W/System: ClassLoader referenced unknown path: /da
11-05 12:39:56.746 3685-3685/com.matrangola.hourhalf W/art: Before Android 4.1, method android.graphic
11-05 12:39:57.004 3685-3854/com.matrangola.hourhalf I/OpenGLRenderer: Initialized EGL, version 1.4
11-05 12:39:57.004 3685-3854/com.matrangola.hourhalf D/OpenGLRenderer: Swap behavior 1
```

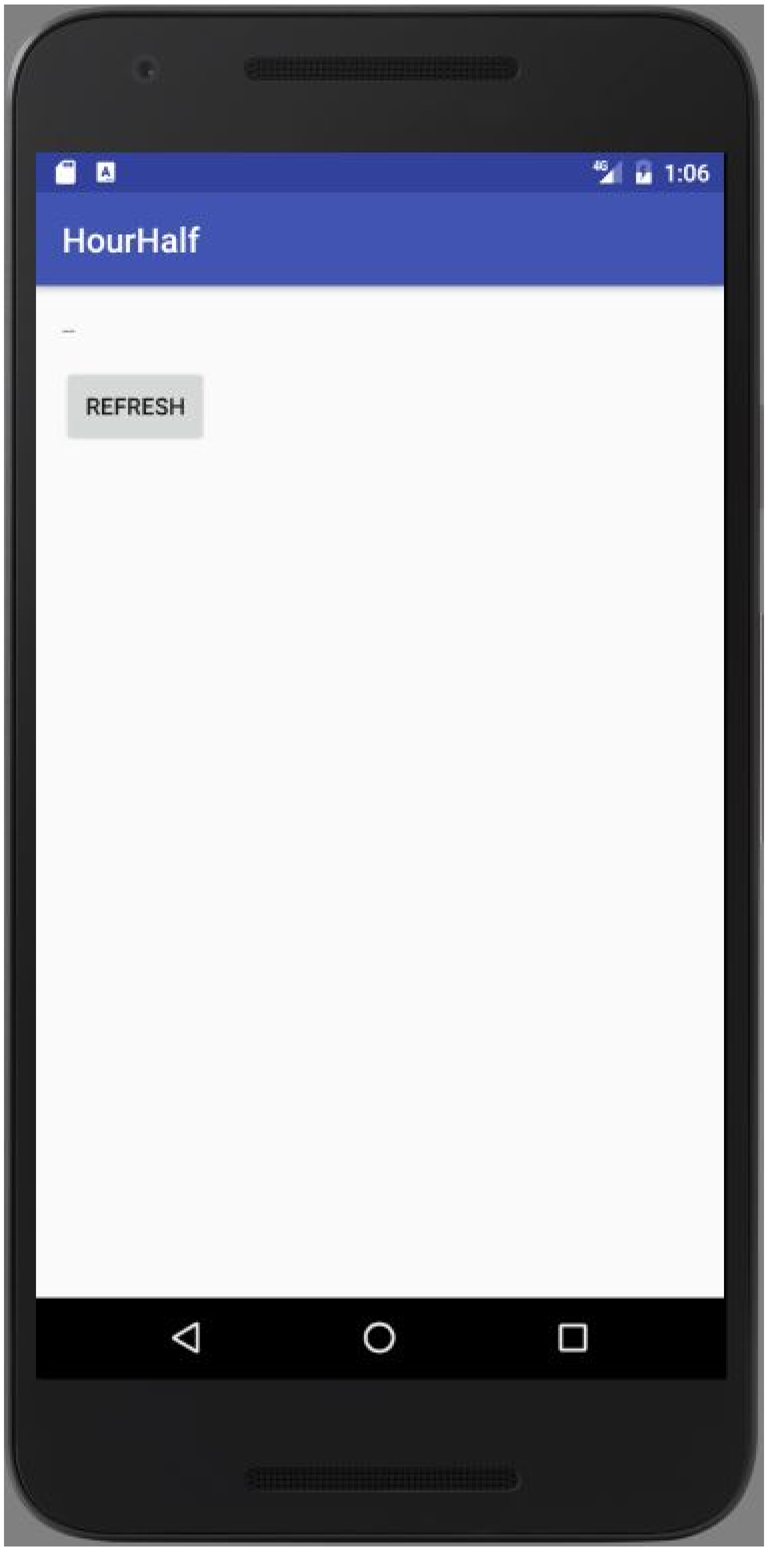
1. App Running
2. No need to stop it between runs. Just leave the Emulator running and app on screen. It will save you time in launches.
3. Console output goes to Android Monitor

Beyond Hello World



ID	refresh_button
layout_width	wrap_content
layout_height	wrap_content
Button	
style	buttonStyle
background	@drawable/btn_default_material
backgroundTint	
stateListAnimator	ı/button_state_list_anim_material
elevation	
visibility	none
onClick	none
TextView	
text	Refresh
text	
contentDescription	
textAppearance	AppCompat.Widget.Button
fontFamily	sans-serif-medium
typeface	none
textSize	14sp
lineSpacingExtra	none
textColor	?attr/textColorPrimary

1. Open activity_weather.xml
2. Click on “Hello World!” label and change Properties
   a. ID = current_temp
   b. Text = “--”
3. Add Refresh Button and change Properties
   a. ID = refresh_button
   b. Text = Refresh



Run the App

Hook up the UI Elements in the Activity Class

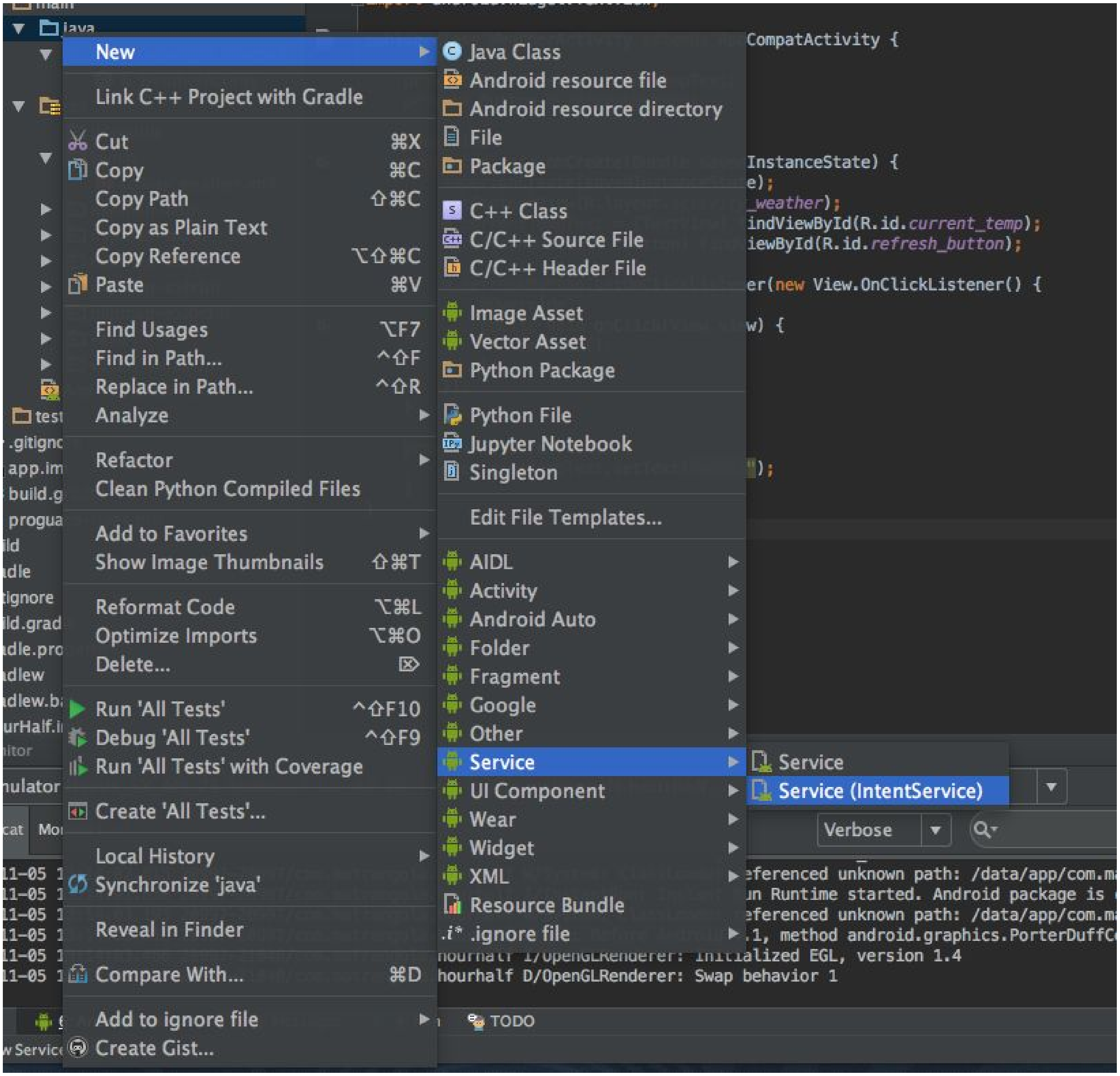
```
package com.matrangola.hourhalf;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class WeatherActivity extends AppCompatActivity {
    private TextView m_currentTempText;
    private Button m_refreshButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_weather);
        m_currentTempText = (TextView) findViewById(R.id.current_temp);
        m_refreshButton = (Button) findViewById(R.id.refresh_button);

        m_refreshButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                refresh();
            }
        });
    }

    private void refresh() {
        m_currentTempText.setText("nice!");
    }
}
```

Rotate

FTW?

The Activity Class is recreated every time the context changes. Orientation change is a context change.

Add IntentService to Get the Weather Data



```
hourhalf I/OpenGLRenderer: Initialized EGL, version 1.4
hourhalf D/OpenGLRenderer: Swap behavior 1
```

```java
package com.matrangola.hourhalf;

import android.app.IntentService;
import android.content.Intent;
import android.content.Context;
import android.util.Log;

/**
 * An {@link IntentService} subclass for handling asynchronous task requests in
 * a service on a separate handler thread.
 * <p>
 * Get the weather data.
 */
public class WeatherIntentService extends IntentService {
    private static final String TAG = "WeatherService";
    private static final String ACTION_GET_TEMP = "com.matrangola.hourhalf.action.GET_TEMP";

    public WeatherIntentService() {
        super("WeatherIntentService");
    }

    /**
     * Starts this service to perform action Foo with the given parameters. If
     * the service is already performing a task this action will be queued.
     *
     * @see IntentService
     */
    public static void startActionGetWeather(Context context) {
        Intent intent = new Intent(context, WeatherIntentService.class);
        intent.setAction(ACTION_GET_TEMP);
        context.startService(intent);
    }


    @Override
    protected void onHandleIntent(Intent intent) {
        if (intent != null) {
            final String action = intent.getAction();
            Log.d(TAG, "onHandleIntent" + action);
            handleAction();
        }
    }

    /**
     * Handle action Foo in the provided background thread with the provided
     * parameters.
     */
    private void handleAction() {

    }

}
```

Get the Weather Data as a JSON stream

URL:
http://api.openweathermap.org/data/2.5/weather?zip=94040,us&units=imperial&appid=4d36b5f1fce463fe1647b8b9711bf707

Output:

```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 741,
      "main": "Fog",
      "description": "fog",
      "icon": "50d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.45,
    "pressure": 1019,
    "humidity": 93,
    "temp_min": 280.15,
    "temp_max": 285.15
  },
  "visibility": 16093,
  "wind": {
    "speed": 0.75,
    "deg": 31.003
  },
  "clouds": {
    "all": 20
  },
  "dt": 1478357760,
  "sys": {
    "type": 1,
    "id": 451,
    "message": 0.1867,
    "country": "US",
    "sunrise": 1478356708,
    "sunset": 1478394297
  },
  "id": 5375480,
  "name": "Mountain View",
  "cod": 200
}
```

Gradle Build System Dependencies

In order to download and parse the Weather Data we need to add Apache Commons IO Util and GSON to our project. This is done in the app/build.gradle file.

Add compile group: 'commons-io', name: 'commons-io', version: '2.5' in the dependencies section.

Click “Sync Now” so that commons.io will be available to your code.

```
apply plugin: 'com.android.application'

android {
   compileSdkVersion 24
   buildToolsVersion "23.0.2"
   defaultConfig {
       applicationId "com.matrangola.hourhalf"
       minSdkVersion 15
       targetSdkVersion 24
       versionCode 1
       versionName "1.0"
       testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
   }
   buildTypes {
       release {
           minifyEnabled false
           proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
       }
   }
}

dependencies {
   compile fileTree(dir: 'libs', include: ['*.jar'])
   androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
       exclude group: 'com.android.support', module: 'support-annotations'
   })
   compile 'com.android.support:appcompat-v7:24.2.1'
   compile group: 'commons-io', name: 'commons-io', version: '2.5'
   testCompile 'junit:junit:4.12'
}
```

Android Manifest

The Manifest tells the OS how to manage your app. Declares Name, Activities, Services, Permissions etc. Activities and Services are added automatically by the IDE. You have to add permissions by hand.

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

   <uses-permission android:name="android.permission.INTERNET"/>

   <application
       android:allowBackup="true"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:supportsRtl="true"
```

```
        android:theme="@style/AppTheme">
        <activity android:name=".WeatherActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <service
            android:name=".WeatherIntentService"
            android:exported="false"/>
    </application>

</manifest>
```

Call the Intent Service

Refresh Button send Intent
Use the WeatherIntentService helper static method.

```
private void refresh() {
    WeatherIntentService.startActionGetWeather(this);
}
```

Run and press button.

Send Temperature Using Broadcast Intent

In handleAction()

```
 // Send Intent back to the Activity for display
        Intent tempResponseIntent = new Intent(TEMP_RESPONSE);
        tempResponseIntent.putExtra(TEMP_EXTRA, temp);
        sendBroadcast(tempResponseIntent);
```

Complete WeatherIntentService Code

```
package com.matrangola.hourhalf;

import android.app.IntentService;
import android.content.Intent;
import android.content.Context;
import android.content.IntentFilter;
import android.util.JsonReader;
import android.util.Log;

import org.apache.commons.io.IOUtils;
import org.json.JSONException;
```

```java
import org.json.JSONObject;

import java.io.IOException;
import java.io.StringReader;
import java.net.URL;

/**
* An {@link IntentService} subclass for handling asynchronous task requests in
* a service on a separate handler thread.
* <p>
* Get the weather data.
*/
public class WeatherIntentService extends IntentService {
   private static final String TAG = "WeatherService";
   private static final String ACTION_GET_TEMP = "com.matrangola.hourhalf.action.GET_TEMP";
   public static final String TEMP_ACTION = "com.matrangola.hourshalf.action.TEMP_RECEIVED";
   public static final String TEMP_RESPONSE = "com.matrangola.hourfalf.response.TEMP_RESPONSE";
   public static final String TEMP_EXTRA = "com.matrangola.hourhalf.extra.TEMP";

   public WeatherIntentService() {
       super("WeatherIntentService");
   }

   /**
    * Starts this service to perform action Foo with the given parameters. If
    * the service is already performing a task this action will be queued.
    *
    * @see IntentService
    */
   public static void startActionGetWeather(Context context) {
       Intent intent = new Intent(context, WeatherIntentService.class);
       intent.setAction(ACTION_GET_TEMP);
       context.startService(intent);
   }


   @Override
   protected void onHandleIntent(Intent intent) {
       if (intent != null) {
           final String action = intent.getAction();
           Log.d(TAG, "onHandleIntent" + action);
           handleAction();
       }
   }

   /**
    * Handle action Foo in the provided background thread with the provided
    * parameters.
    */
   private void handleAction() {
       String url = "http://api.openweathermap.org/data/2.5/weather?zip=94040,us&units=imperial&appid=4d36b5f1fce463fe1647b8b9711bf707";
       String json = "empty";
       try {
           json = IOUtils.toString(new URL(url), "UTF-8");
           Log.d(TAG, "json = " + json);
           JSONObject jsonWeather = new JSONObject(json);
           JSONObject main = jsonWeather.getJSONObject("main");
           double temp = main.getDouble("temp");
           Log.d(TAG, "temp = " + temp);
           // Send Intent back to the Activity for display
```

```
            Intent tempResponseIntent = new Intent(TEMP_RESPONSE);
            tempResponseIntent.putExtra(TEMP_EXTRA, temp);
            sendBroadcast(tempResponseIntent);
        } catch (IOException e) {
            Log.e(TAG, "Error getting JSON weather data ", e);
        } catch (JSONException e) {
            Log.e(TAG, "Error parsing JSON string " + json, e);
        }
    }
}
```

Receive BroadcastIntent in Activity

onResume() is called before the Activity is displayed. onPause() is called when the Activity is put in background (cannot receive input). In onResume(), create an inner-anonymous BroadcastReceiver object and register it to receive the TEMP_RESPONSE BroadcastIntent. In onPause(), release the registration.

Final WeatherActivity Code

```
package com.matrangola.hourhalf;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class WeatherActivity extends AppCompatActivity {

    private TextView m_currentTempText;
    private Button m_refreshButton;
    private BroadcastReceiver m_receiver;
    private String TAG = "WeatherActivity";
    private static final String DEGREE  = "\u00b0";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_weather);
        m_currentTempText = (TextView) findViewById(R.id.current_temp);
        m_refreshButton = (Button) findViewById(R.id.refresh_button);

        m_refreshButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                refresh();
            }
        });
    }
    @Override
    protected void onResume() {
        super.onResume();
        m_receiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                double tempExtra = intent.getDoubleExtra(WeatherIntentService.TEMP_EXTRA, 0);
                Log.d(TAG, "onReceive: " + tempExtra);
                m_currentTempText.setText(tempExtra + DEGREE +  "F");
            }
        };
        registerReceiver(m_receiver, new IntentFilter(WeatherIntentService.TEMP_RESPONSE));
    }

    @Override
    protected void onPause() {
        super.onPause();
        unregisterReceiver(m_receiver);
    }

    private void refresh() {
        WeatherIntentService.startActionGetWeather(this);
    }
}
```