

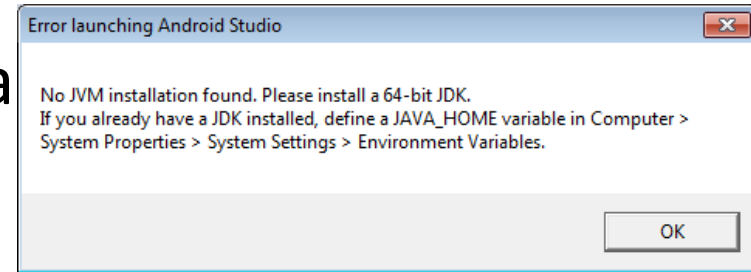
Android Developing - Environment Setup

Android Studio

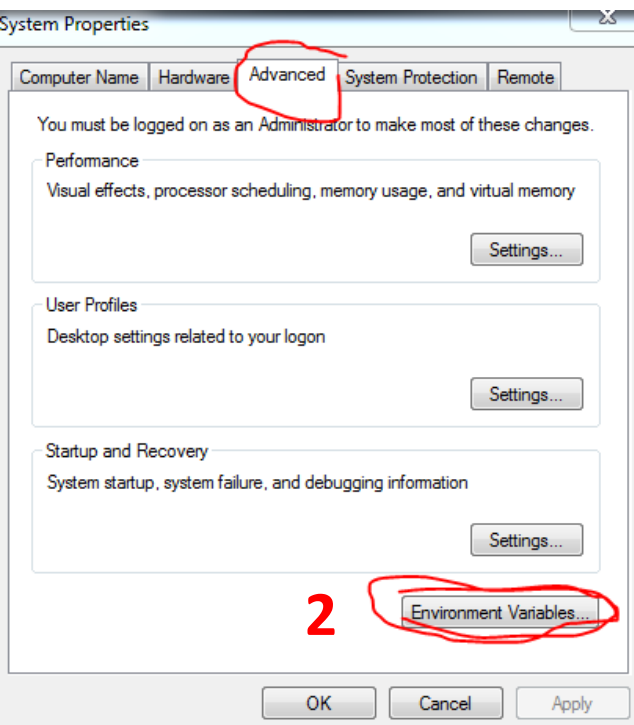
- Download:
<http://developer.android.com/sdk/index.html>
- Requirement:
 - Java JDK 7 or higher version
 - 2 GB memory
 - Windows / Mac OS X (10.8.5)

Error – Environment Variable

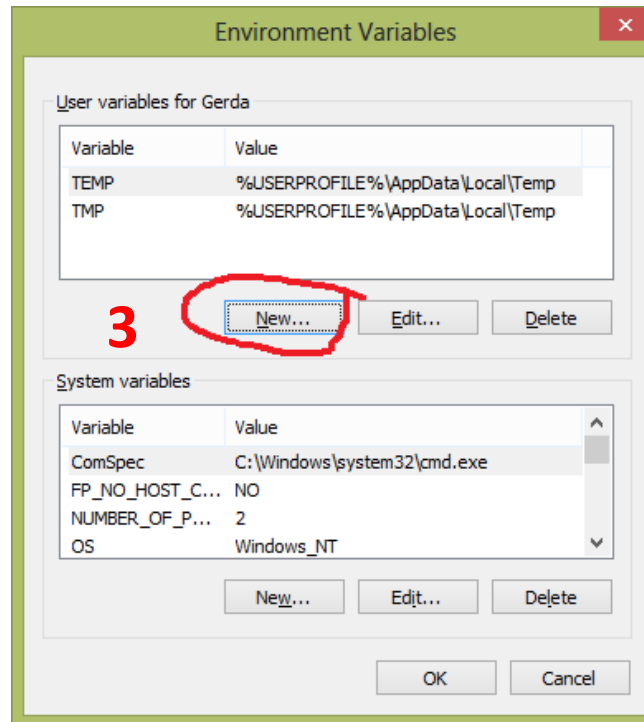
- Find the path of the installed Java JDK and add it as a system environment variable



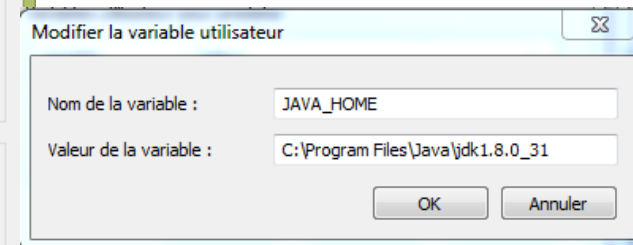
1



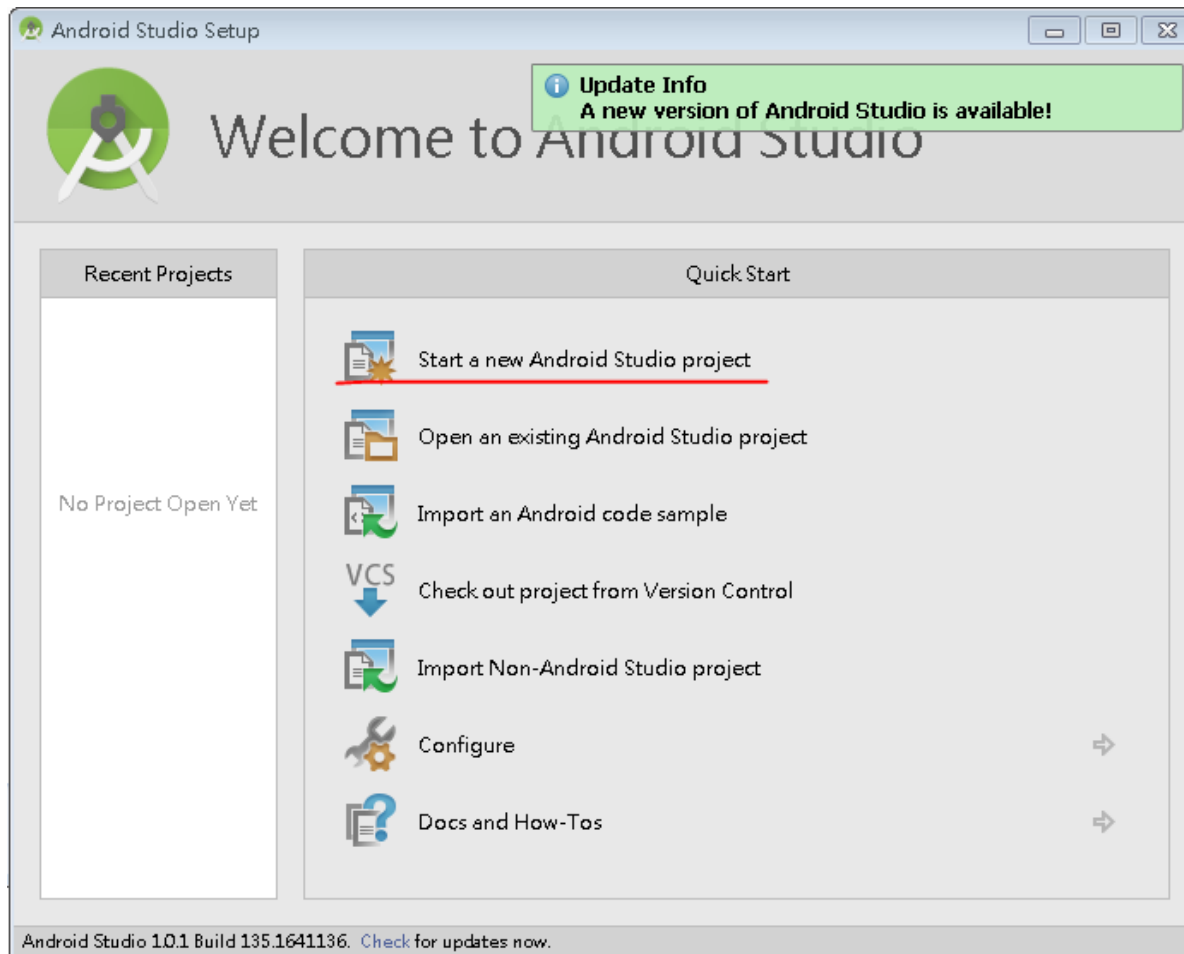
2



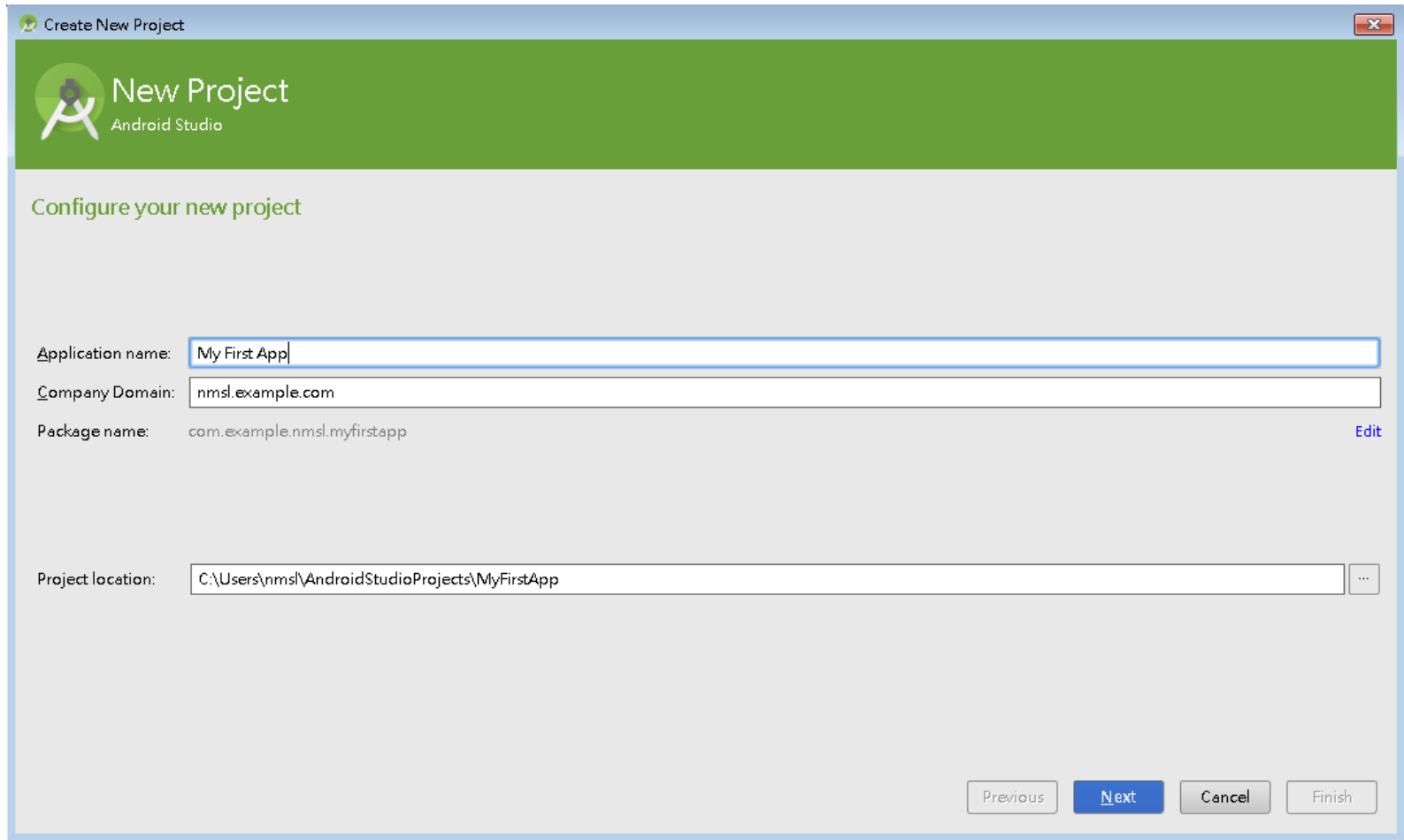
4



Create Your First Android Project



Your Project Name



Create New Project

New Project
Android Studio

Configure your new project

Application name:

Company Domain:

Package name: [Edit](#)

Project location: ...

Select the API Level

Create New Project

New Project
Android Studio

Select the form factors your app will run on

Different platforms require separate SDKs

Phone and Tablet

Minimum SDK: API 19: Android 4.4 (KitKat)

Lower API levels target more devices, but have fewer features available. By targeting API 19 and later, your app will run on approximately 24.5% of the devices that are active on the Google Play Store. [Help me choose.](#)

TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

Wear

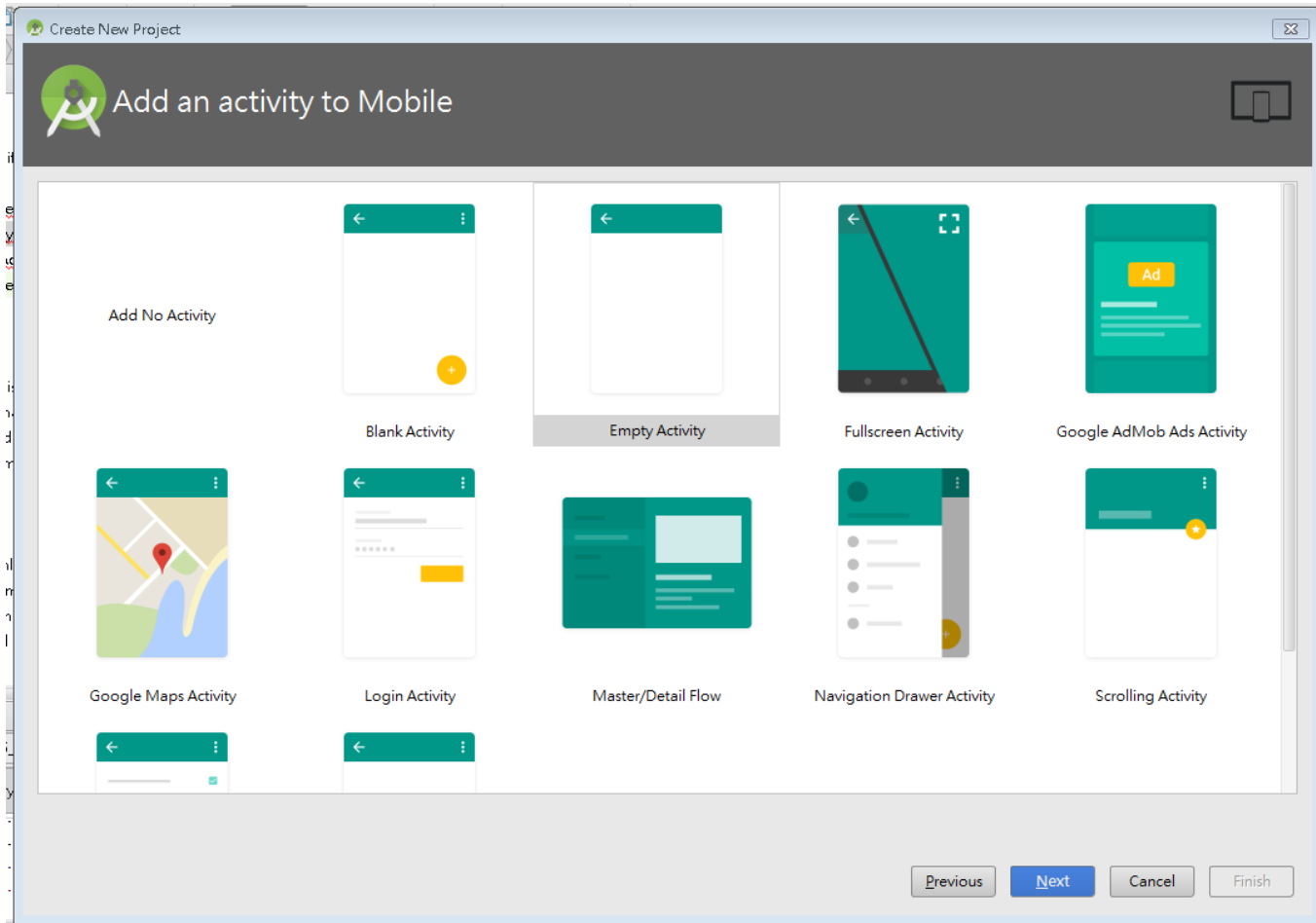
Minimum SDK: API 21: Android 5.0 (Lollipop)

Glass (Not Installed)

Minimum SDK:

Previous Next Cancel Finish


Empty Activity



Your Activity Name

Create New Project

Choose options for your new file



Creates a new blank activity with an action bar.

Activity Name:

Layout Name:

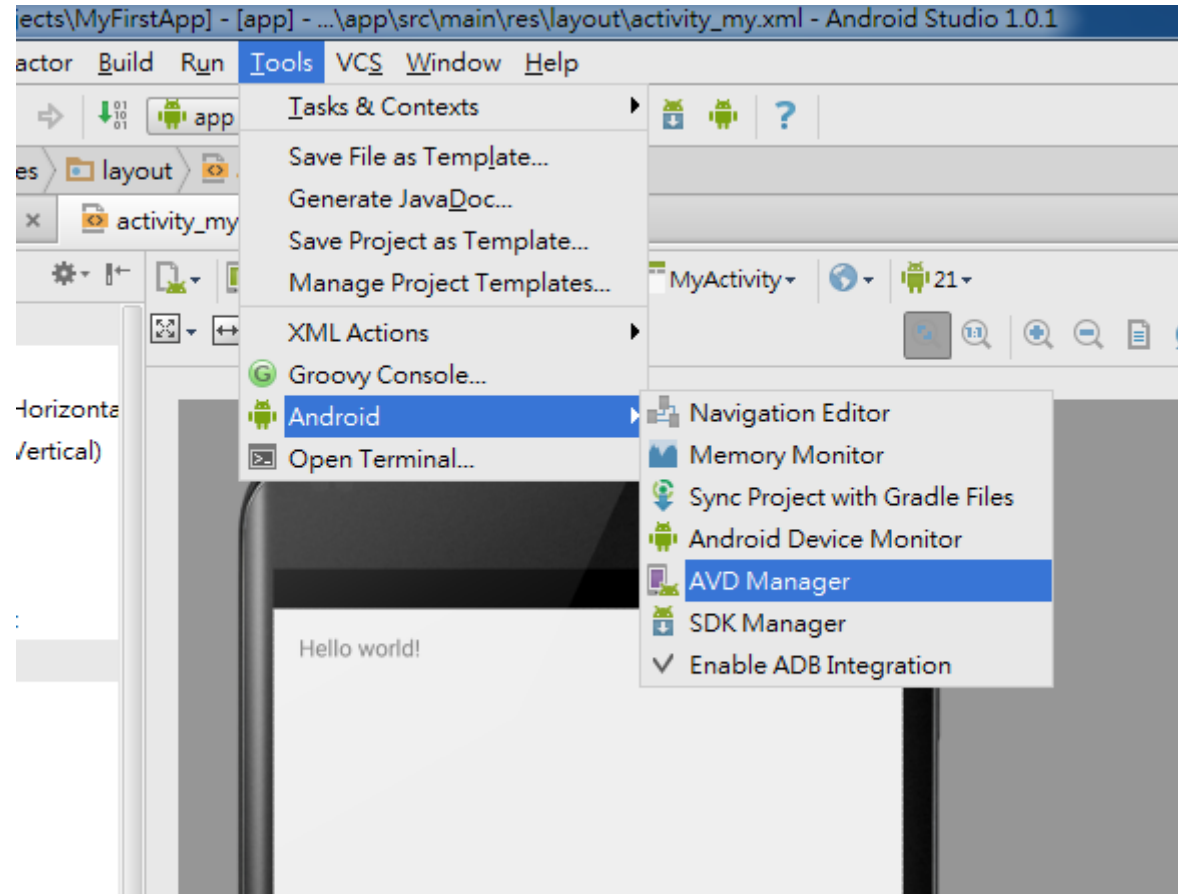
Title:

Menu Resource Name:

Blank Activity

The name of the activity class to create

Create Your Emulator



Cannot See AVD Manager?

- If you cannot see the option of AVD Manager, please change the permission of your android studio folder

Modify Your Permission to Launch AVD Manager

The screenshot shows a Windows Explorer window with the address bar set to '電腦 > 本機磁碟 (C:) > Program Files'. The main pane displays the 'Android' folder, last modified on 2015/2/18 下午 0... The folder is highlighted, and three dialog boxes are overlaid on top of it.

Dialog 1: Android - 內容
This dialog shows the '安全性' (Security) tab. The object name is 'C:\Program Files\Android'. The group or user name is 'Administrators (smonion-PCUsers)'. The permissions list includes '完全控制' (Full Control), '修改' (Change), '讀取和執行' (Read & Execute), '列出資料夾內容' (List Folder Contents), '讀取' (Read), and '寫入' (Write). A link at the bottom says '深入了解存取控制及權限'.

Dialog 2: Android 的進階安全性設定
This dialog shows the '權限' (Permissions) tab. The object name is 'C:\Program Files\Android'. The permissions list includes '允許' (Allow) for 'TrustedInstaller', 'SYSTEM', 'Administrators (smonion-PCUsers)', 'Users (smonion-PCUsers)', and 'CREATOR OWNER'. A link at the bottom says '管理權限項目'.

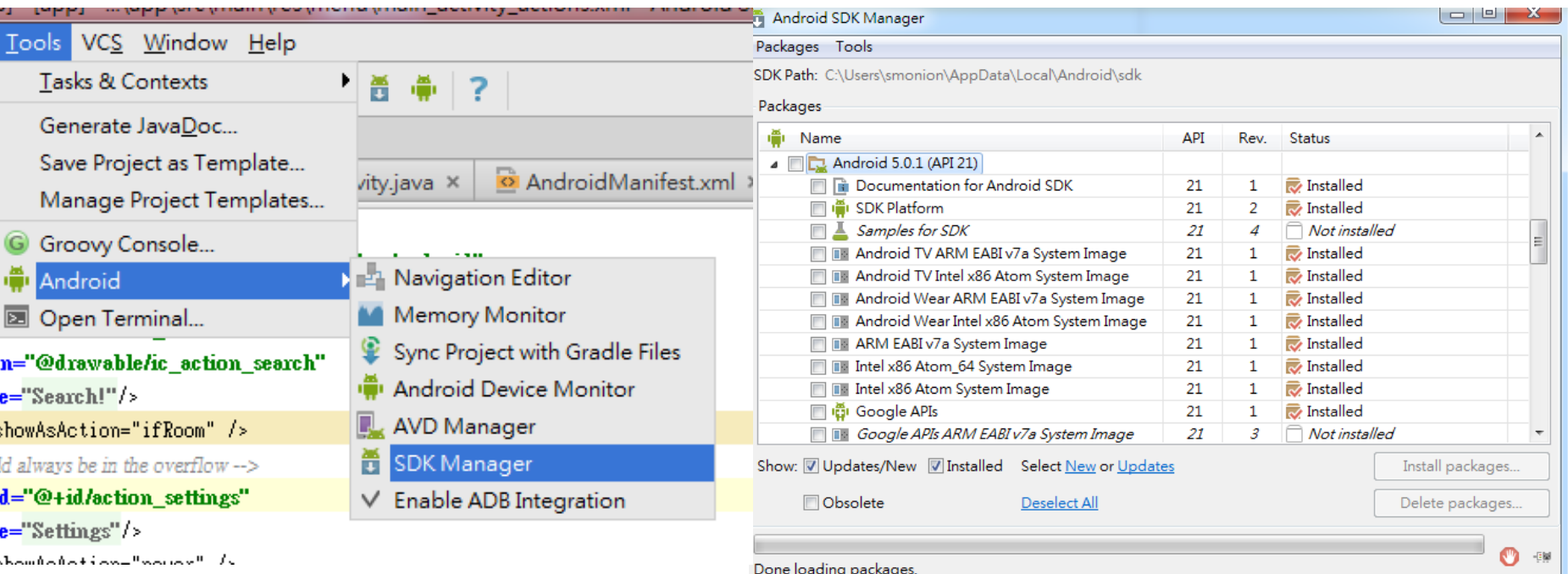
Dialog 3: Android 的權限項目
This dialog shows the '物件' (Object) tab. The name is 'Users (smonion-PCUsers)'. The scope is '這個資料夾、子資料夾及檔案'. The permissions list includes '完全控制' (Full Control), '周遊資料夾/執行檔案' (Traverse Folders/Execute Files), '列出資料夾/讀取資料' (List Folder Contents), '讀取屬性' (Read Attributes), '讀取擴充屬性' (Read Extended Attributes), '建立檔案/寫入資料' (Create Files/Write Data), '建立資料夾/附加資料' (Create Folders/Append Data), and '寫入屬性' (Write Attributes). A link at the bottom says '管理權限'.

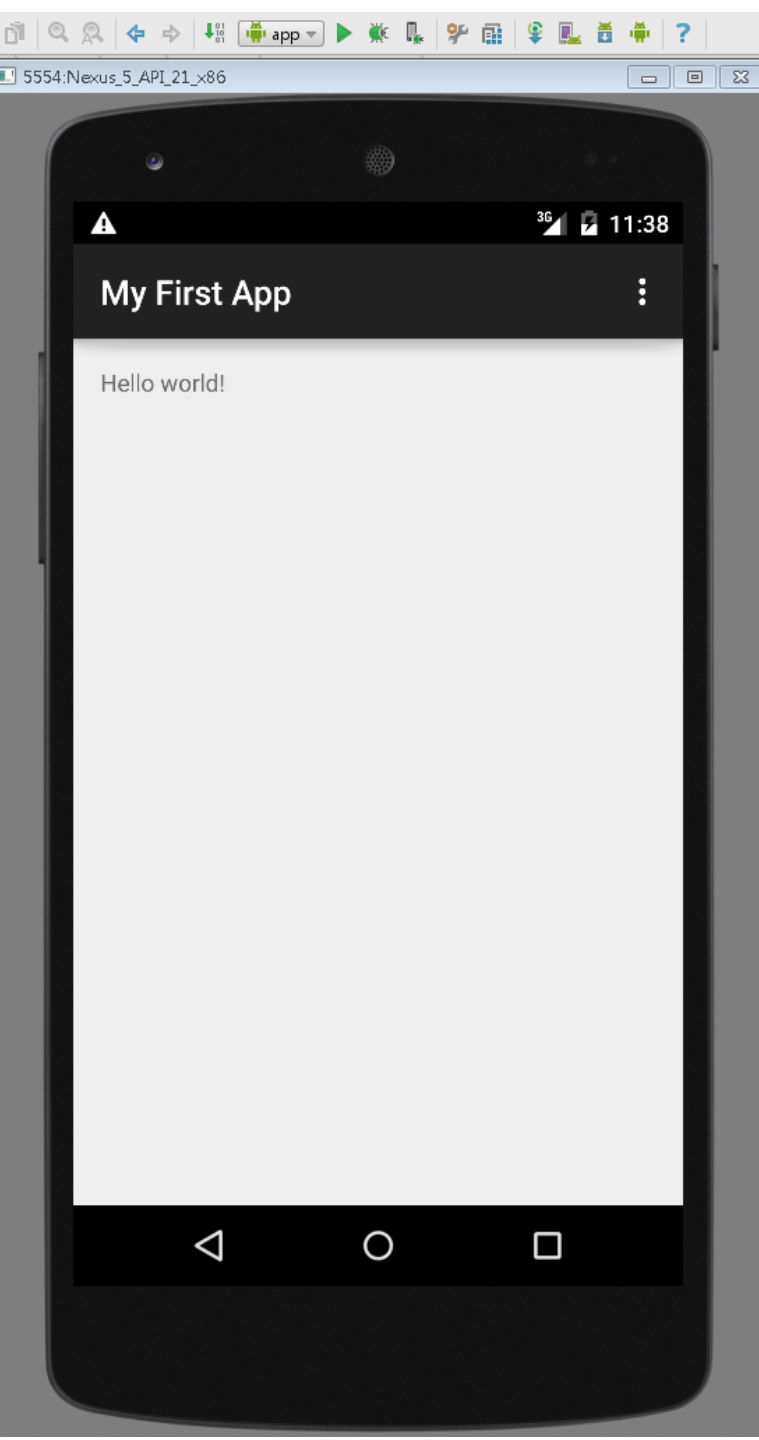
Launch the Default Emulator

- Please run the default Nexus 5 Emulator using AVD Manager
- If you would like to create your own emulator, you need to update your SDK packages first.

SDK Manager

- Update your SDK package using SDK Manager





- Compile and run your project and you can see the message on your virtual Nexus 5!

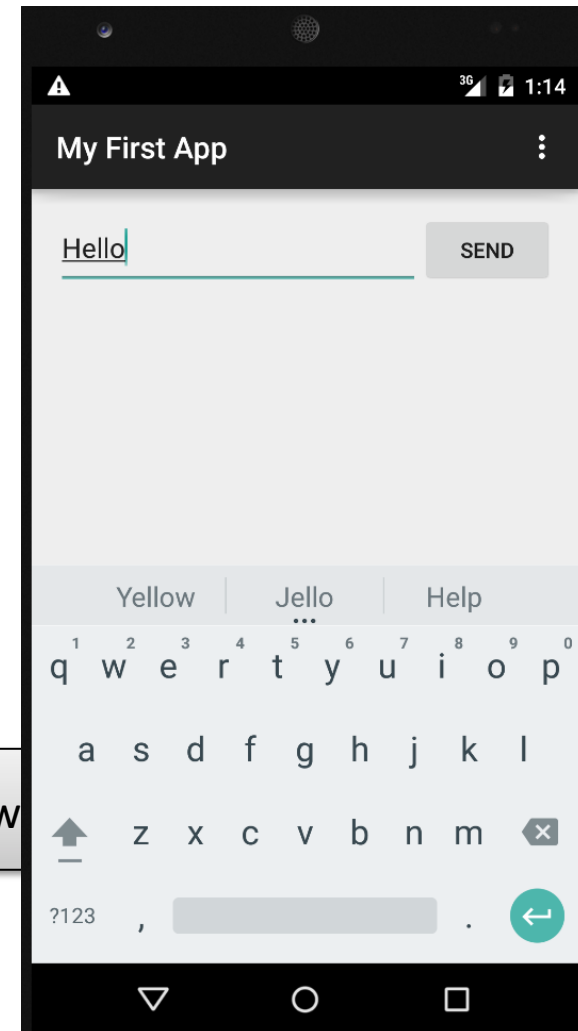
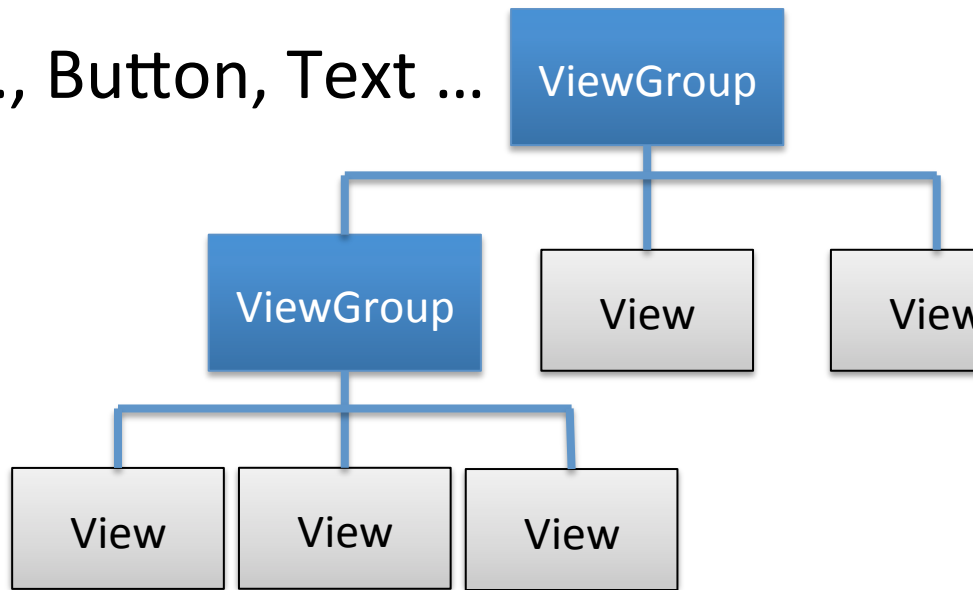
Android Developing - Your First App

Android Activity

- Interact with users
- Visual User interface
- Hierarchy of **views**
- One or several activities in an application

Hierarchy of View

- View Group:
 - Invisible view container
 - How the child views are laid out
- View:
 - Visible
 - E.g., Button, Text ...



Activity Life Cycle

- Activities are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity
- States:
 - Active / running: activity in the foreground
 - Pause: An activity has lost focus but is still visible
 - Stopped: It's no longer visible but still retains all state member information
 - Finish / Kill

Android Manifest

- The components used in an android application should be declared
 - Activity
 - permission
 - Intent
 - ...

What is Intent

- Intent is like an event sent by user
- Intent should
 - Specify the receiving component
 - Or have an intent-filter to allow your android system to know this intent

Manifest Example

<manifest xmlns:android=<http://schemas.android.com/apk/res/android>

package="com.example.nmsl.myfirstapp" >

The Package Name

<application →

Describe Your Application

android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >

<activity →

Activity Component

android:name=".MainActivity"
android:label="@string/app_name" >

<intent-filter →

Intent Filter to receive launch
intent

<action android:name="android.intent.action.MAIN" />

Create()

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

Start()

</intent-filter>

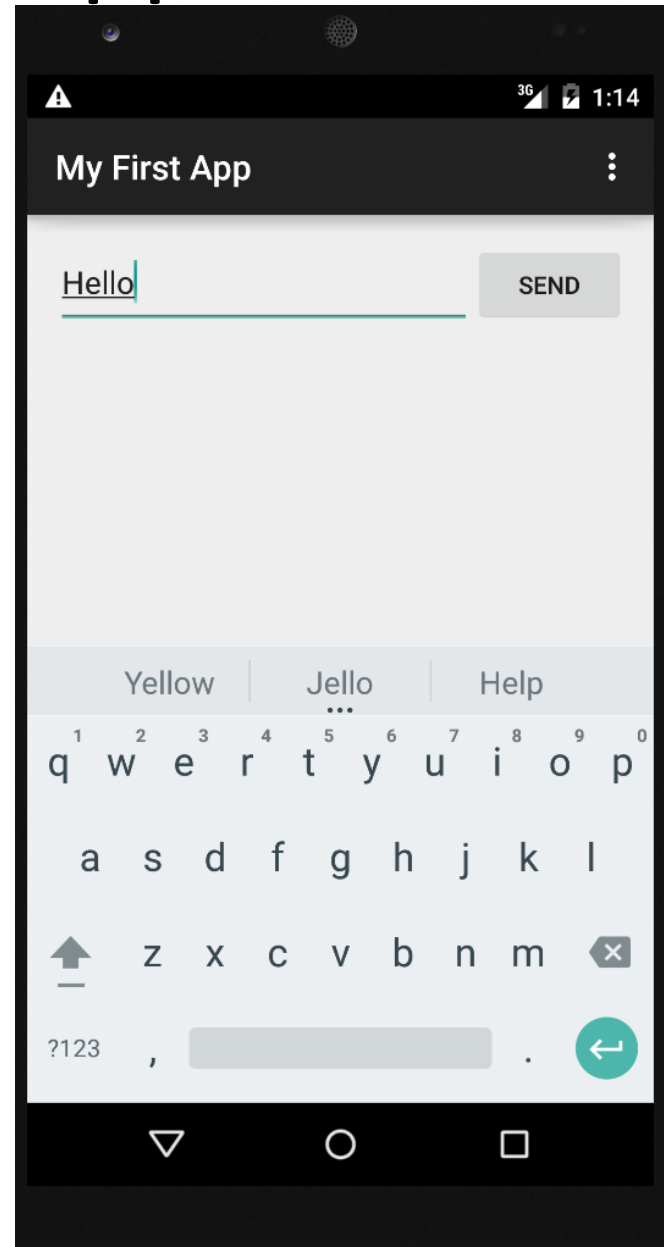
</activity>

</application>

</manifest>

Your First App

- Edit Text
- Button
 - Listener
 - Send Message
 - Create Second Activity



Steps

- Create a linear layout
- Add your view objects into the layout
- Create the resources used in the view objects
- Create the second activity
- Create the function to do interaction while we push the button

Linear Layout

- Edit your activity_my.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
  xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"  
  android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_ma  
  android:paddingRight="@dimen/activity_horizontal_margin"  
  android:paddingTop="@dimen/activity_vertical_margin"  
  android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity"  
  android:orientation="horizontal">
```

```
<EditText android:id="@+id/edit_message"  
  android:layout_weight="1"  
  android:layout_width="0dp"  
  android:layout_height="wrap_content"  
  android:hint="@string/edit_message" />
```

The default value of weight of each view is 0

Missing the String

```
<Button  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
  android:text="@string/button_send"/>
```

The width and height can just contain the view

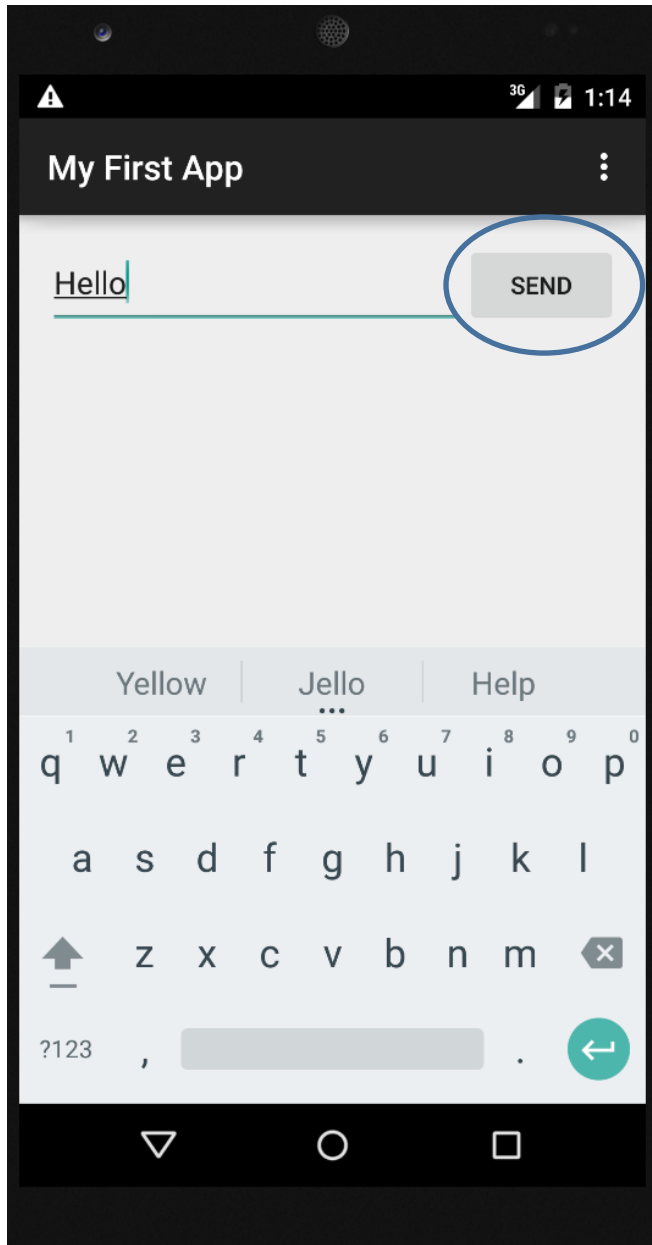
Missing the String

```
</LinearLayout>
```


Add String Resources

- Edit string.xml

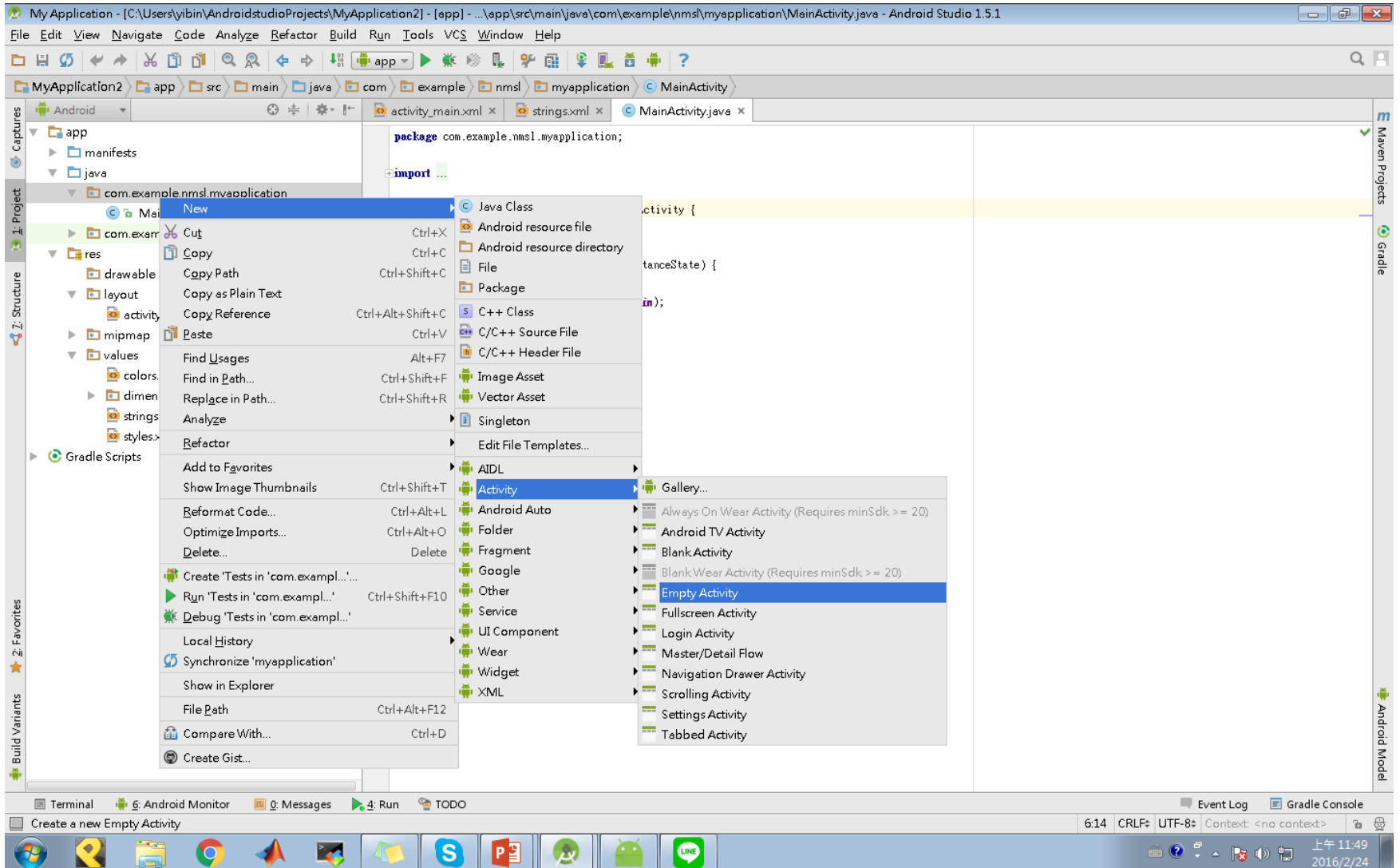
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">My First App</string>
  <string name="edit_message">Enter a message</string>
  <string name="button_send">Send</string>>
</resources>
```



Open another activity to show the message



Create the Second Activity

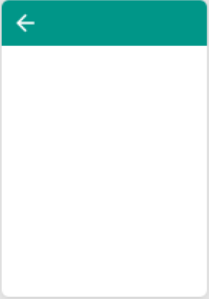


Create the Second Activity

New Android Activity

Customize the Activity

Creates a new empty activity

 Empty Activity

Activity Name:

Generate Layout File

Layout Name:

Launcher Activity

Package name:

The name of the activity class to create

Starting Another Activity

- Link your button with a function to do something
- Edit activity_m.xml

```
<Button
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="@string/button_send"
```

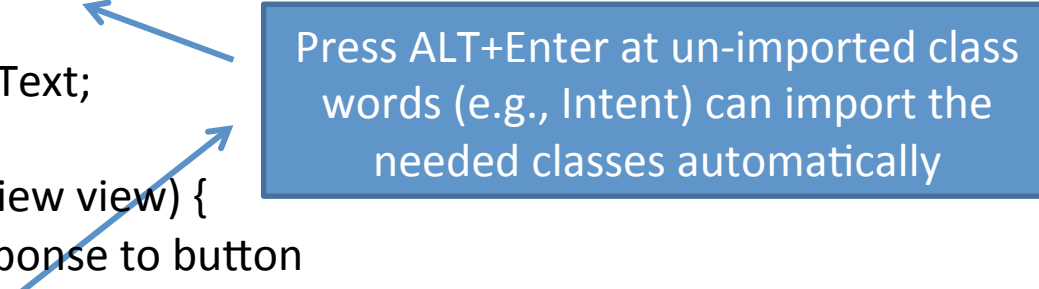
```
    android:onClick="sendMessage" />
```

The name of your function

- Edit MyActivity.java to add the function

Create the function

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
...
public void sendMessage(View view) {
    // Do something in response to button
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```



Press ALT+Enter at un-imported class words (e.g., Intent) can import the needed classes automatically

Create a Unique Key

```
public void sendMessage(View view) {  
    // Do something in response to button  
    Intent intent = new Intent(this, DisplayMessageActivity.class);  
    EditText editText = (EditText) findViewById(R.id.edit_message);  
    String message = editText.getText().toString();  
    intent.putExtra(EXTRA_MESSAGE, message);  
    startActivity(intent);  
}
```

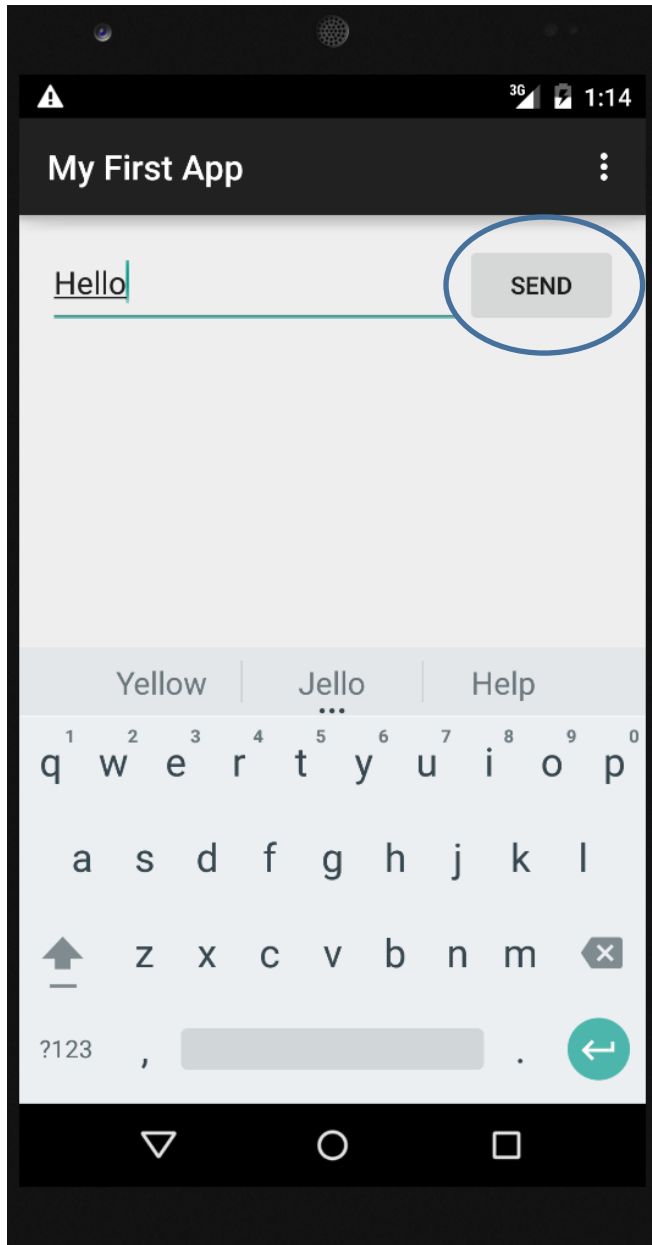
```
public class MainActivity extends AppCompatActivity {  
    public final static String EXTRA_MESSAGE = "com.mycompany.myfirstapp.MESSAGE";  
    ...  
}
```

Create a unique key for the message put by the intent. We then get the message by this key in the second activity (next page)

Receive the Intent

- Edit DisplayMessageActivity.java
 - Get the message from the intent
 - Create a textview to show the message

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Intent intent = getIntent();  
    String message = intent.getStringExtra(MyActivity.EXTRA_MESSAGE);  
    TextView textView = new TextView(this);  
    textView.setTextSize(40);  
    textView.setText(message);  
    setContentView(textView);  
}
```

Open another activity to show the message

