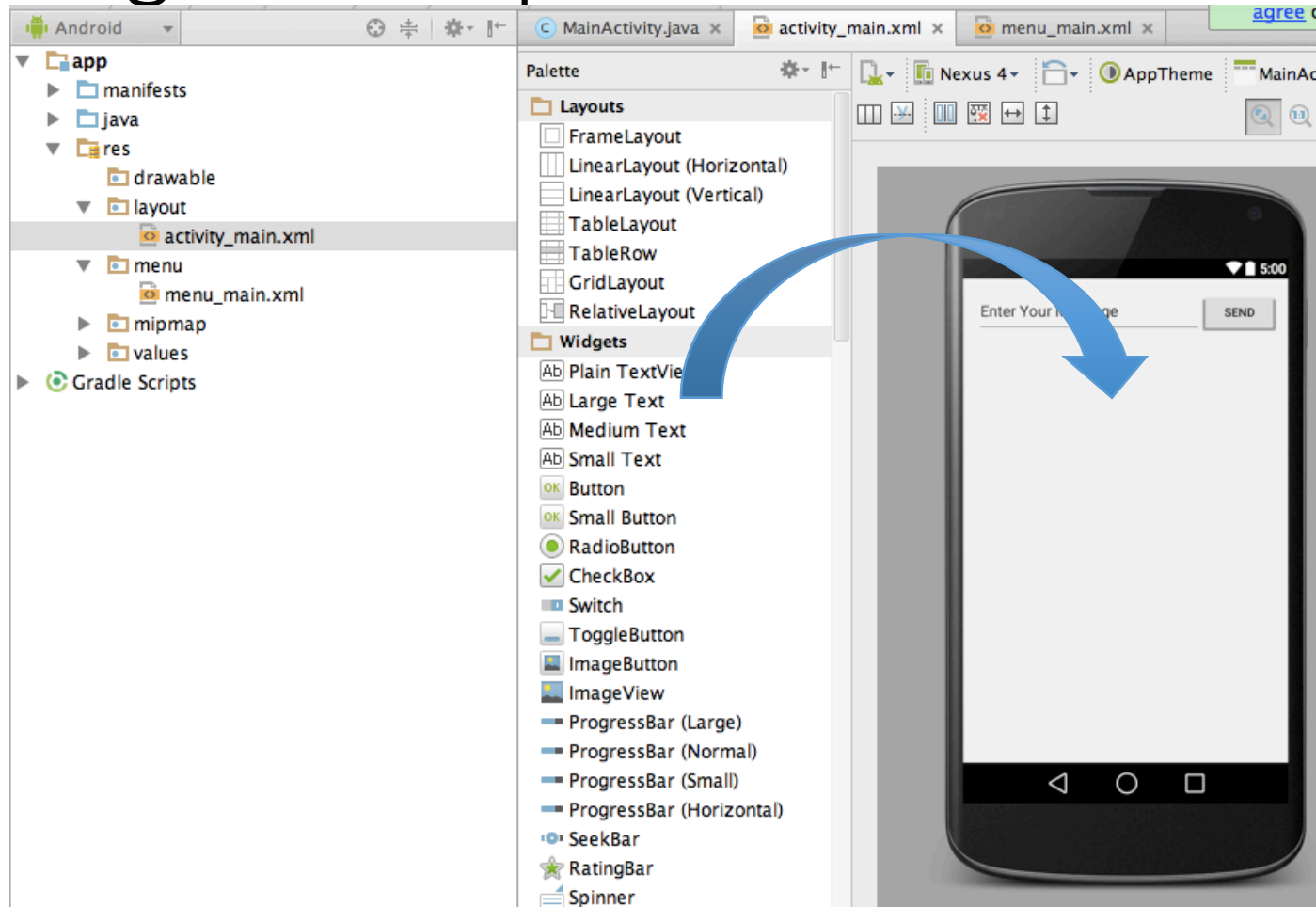


Basic User Interface: Intent and Action Bar

Basic Layout

- Different Layout Styles (View Group):
 - Relative layout (default), linear layout, Table layout...
- Hierarchy of View
 - View Group (contains views, invisible)
 - View (visible: button, text ...)

Drag and Drop

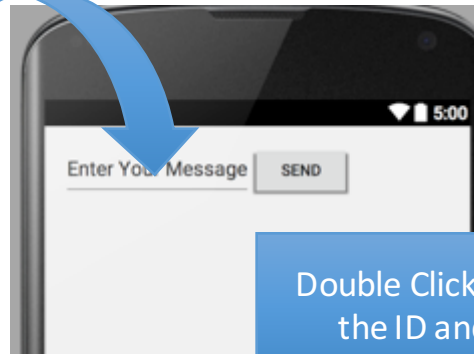
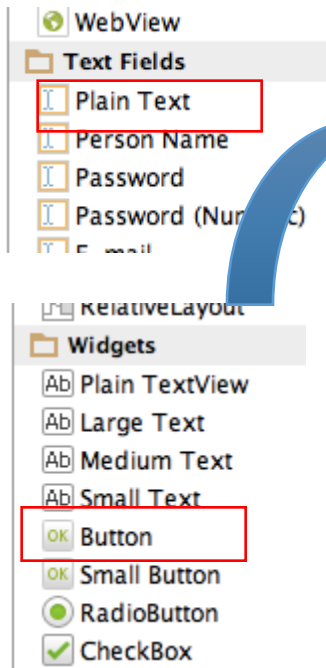


Change the Layout Style

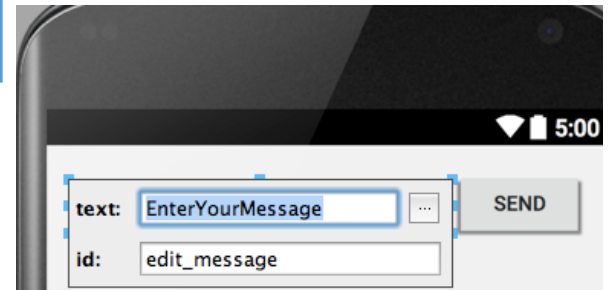
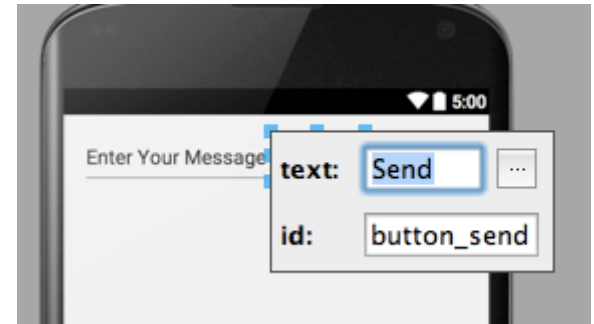
- Relative layout -> linear layout
 - Cannot drag and drop to change the root layout
 - Modify it in XML code



Add EditText and Button

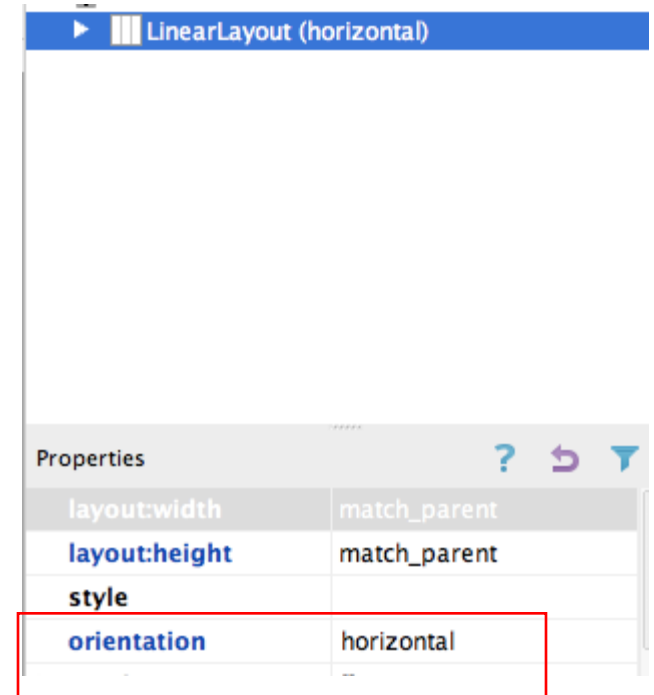


Double Click on Views To Change the ID and Text of the View



Modify the Properties of Views and View Group

- Select a view in the hierarchy in your right hand side
 - E.g., Change the orientation to horizontal



Properties Used in Our Example

- Orientation of LinearLayout: horizontal
- Height and Width: wrap_content
- Weight (priority) of Each Views:
 - Edit_message: weight=1
 - Button: weight=0

The image displays two screenshots of an IDE's Component Tree and Properties panels, illustrating the configuration of a LinearLayout.

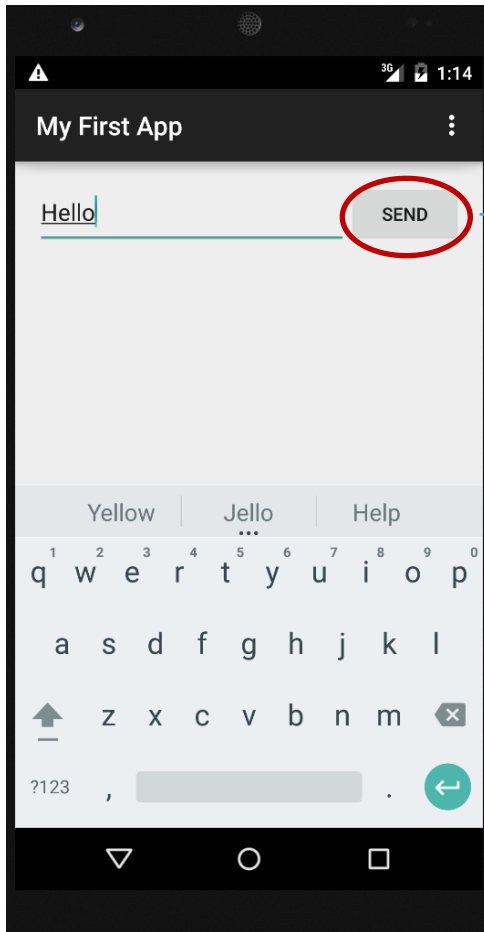
Left Screenshot (LinearLayout Properties):

- Component Tree: Device Screen > LinearLayout (horizontal) > editMessage (EditText)
- Properties:
 - layout:width: wrap_content
 - layout:height: wrap_content
 - layout:gravity: []
 - layout:margin: []
 - layout:weight: 1

Right Screenshot (Button Properties):

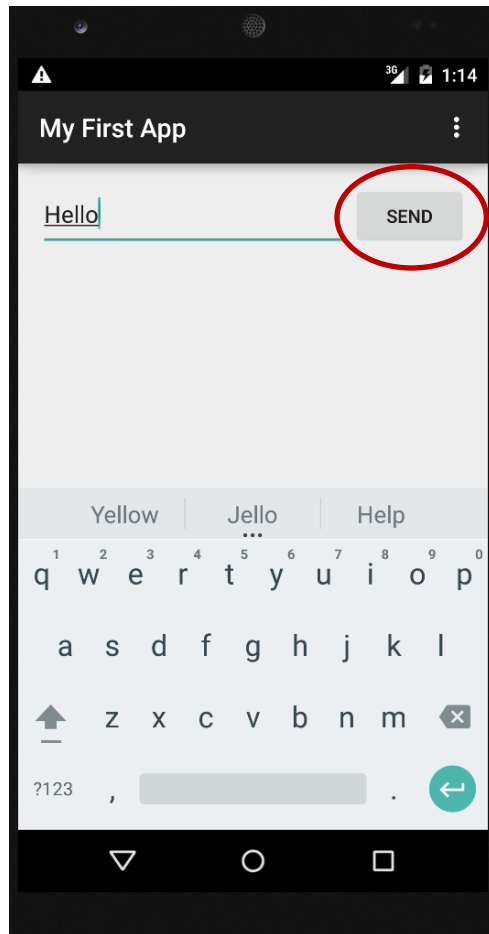
- Component Tree: Device Screen > LinearLayout (horizontal) > button - "Send"
- Properties:
 - layout:width: wrap_content
 - layout:height: wrap_content
 - layout:gravity: []
 - layout:margin: []
 - layout:weight: 0

Current State

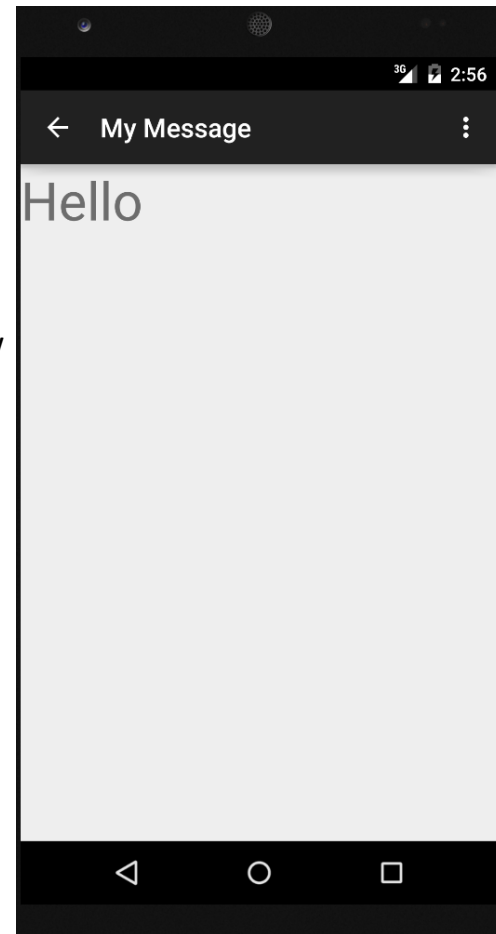


Nothing Happened When You Click the Button

Using Intent to Switch to Another Activity



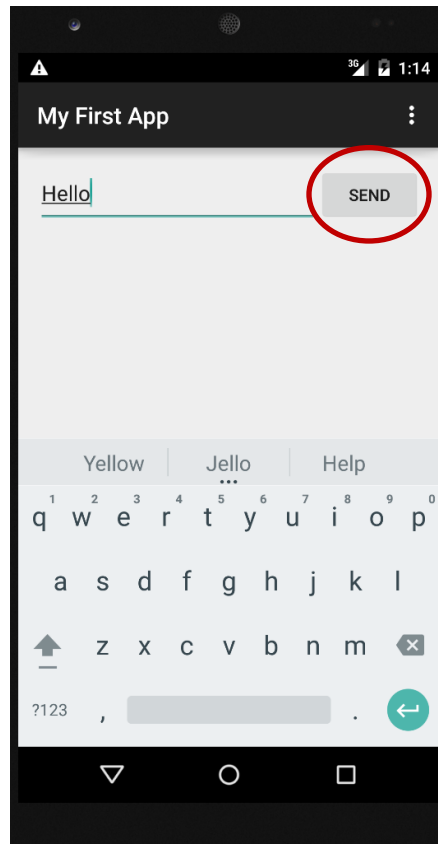
Open another activity to show the message



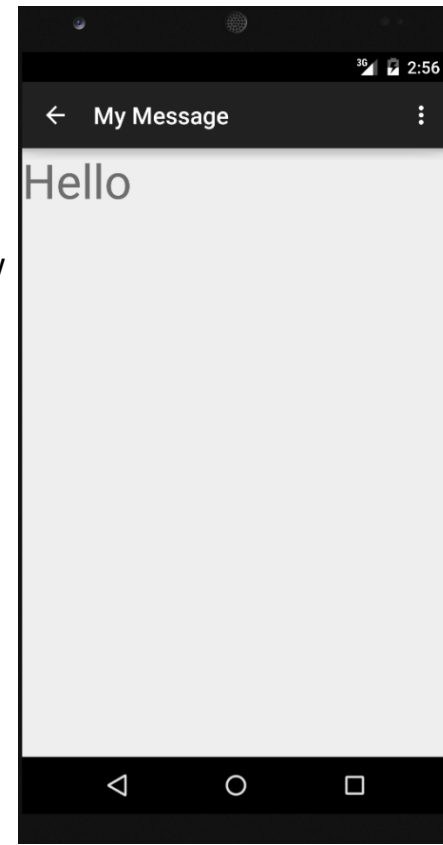
What is Intent?

- An intent is a description of an action to be performed
- Intent to **do something** or **go to somewhere**
 - Open Browser, Camera, ...
 - Go (switch) to another activity
- Using **Bundle** to carry data
- You can find the actions in the following link
<http://developer.android.com/reference/android/content/Intent.html>

Example 1 – Carry Your Message From one Activity to Another



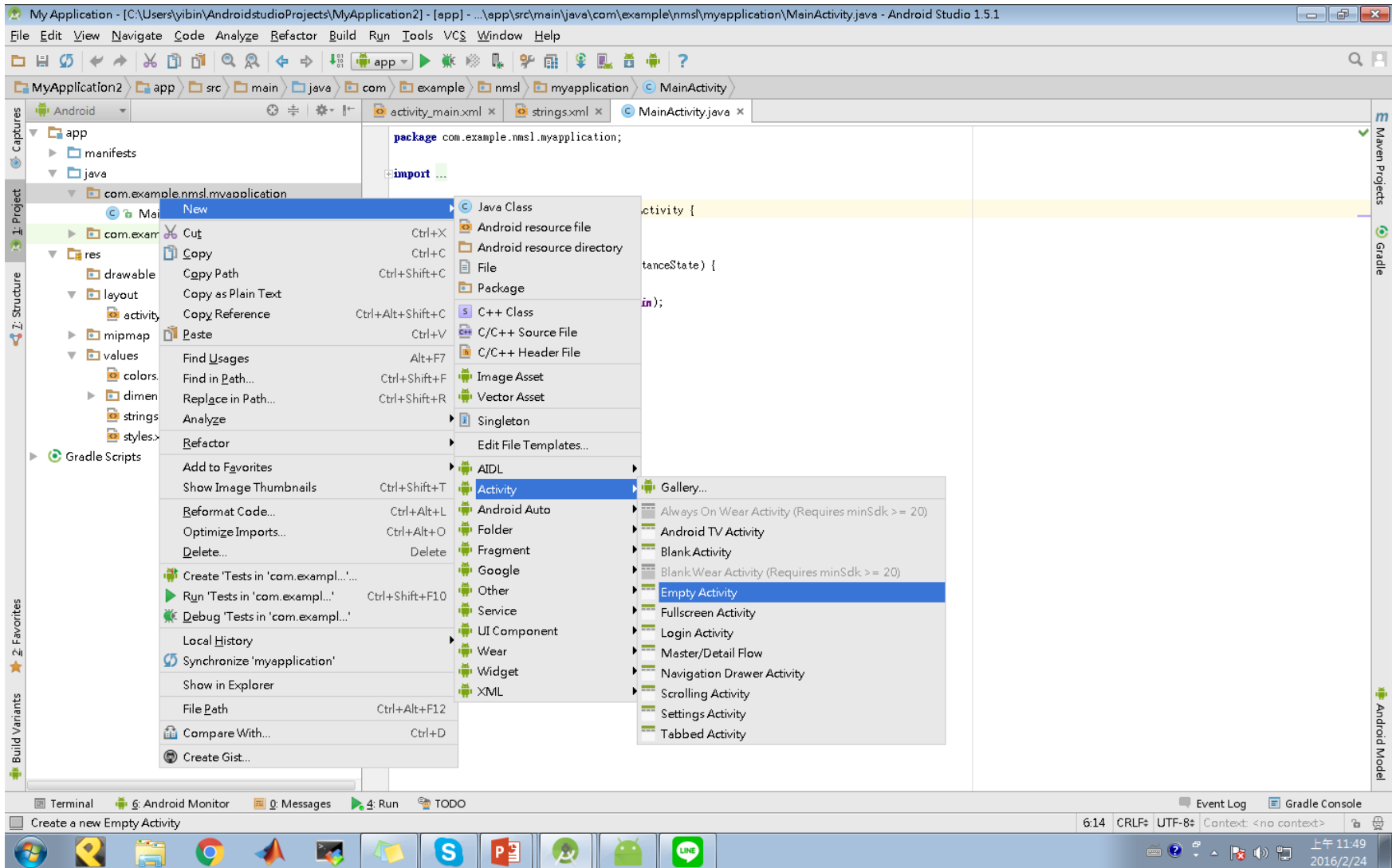
Open another activity to show the message



Step 1:

- **Create second activity**
- Create a function which is triggered once you click your button
- Describe your intent in the function
- Get your message from your TextEdit Field
- Put your message into a Bundle
- Perform your intent which carries with the Bundle

Create the Second Activity



Create the Second Activity

New Android Activity

Customize the Activity

Creates a new empty activity

Activity Name:

Generate Layout File

Layout Name:

Launcher Activity

Package name:

Empty Activity

The name of the activity class to create

Previous Next Cancel Finish

Step 2:

- Create second activity
- Create a function which is triggered once you click your button
- Describe your intent in the function
- Get your message from your TextEdit Field
- Put your message into a Bundle
- Perform your intent which carries with the Bundle
- Get the message in the second activity

OnClick Listener

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

```
import android.content.Intent;
```

```
...
```

```
public void sendMessage(View view) {
```

```
    // Do something in response to button
```

```
}
```


Step 3:

- Create second activity
- Create a function which is triggered once you click your button
- Describe your intent in the function
- Get your message from the TextEdit Field
- Put your message into a Bundle
- Perform your intent which carries with the Bundle
- Get the message in the second activity

Describe the Intent and Get the Message

```
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();
    ...
}
```

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

Second Activity

Current Activity

Id of Your Message View

Step 4:

- Create second activity
- Create a function which is triggered once you click your button
- Describe your intent in the function
- Get your message from your TextEdit Field
- Put your message into a Bundle
- Perform your intent which carries with the Bundle
- Get the message in the second activity

Using Bundle to Carry the Message and Perform the Intent

```
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();  
    Bundle bundle = new Bundle();  
    bundle.putString(EXTRA_MESSAGE, message);  
    intent.putExtras(bundle);  
    startActivity(intent);  
}
```

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

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

Step 5:

- Create second activity
- Create a function which is triggered once you click your button
- Describe your intent in the function
- Get your message from your TextEdit Field
- Put your message into a Bundle
- Perform your intent which carries with the Bundle
- **Get the message in the second activity**

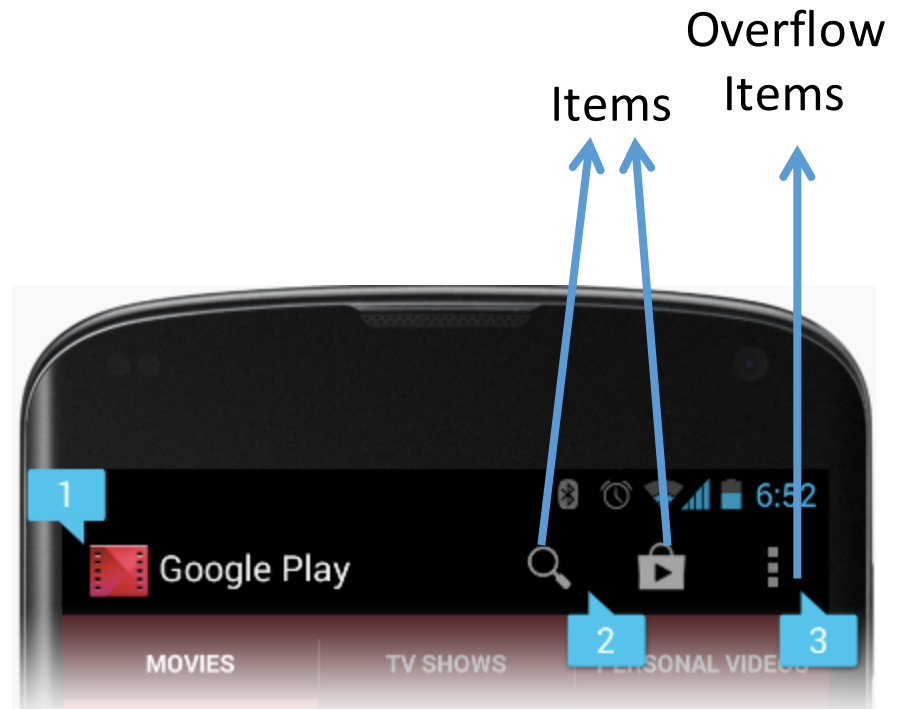
Receive the Intent and Get the Message

- Edit DisplayMessageActivity.java (the second activity)
 - Get the message from the intent
 - Create a textview to show the message

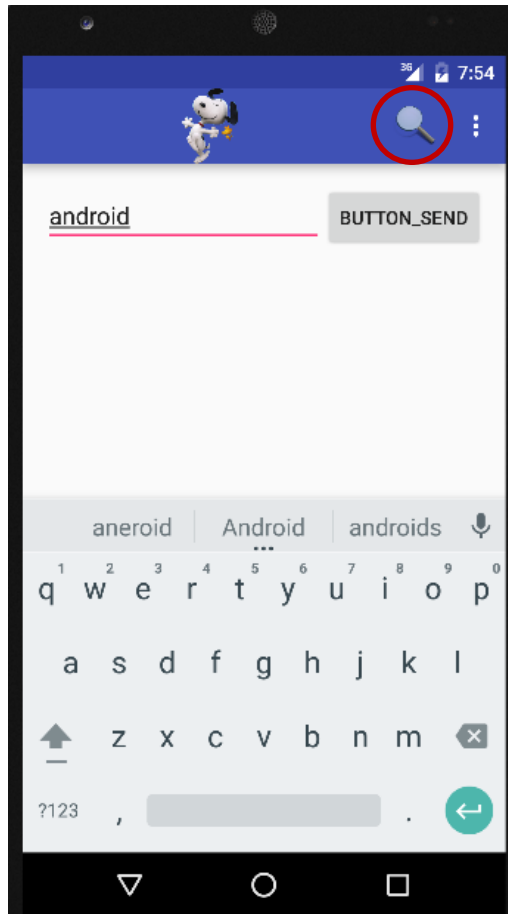
```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    Intent intent = getIntent();  
    Bundle bundle = intent.getExtras();  
    String message = bundle.getString(MyActivity.EXTRA_MESSAGE);  
    TextView textView = new TextView(this);  
    textView.setTextSize(40);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

Action Bar

- Action bar shows users where you are
- Make important actions be easily accessible
- Includes
 - Application icon
 - Items
 - Overflow items



Example 2 – Search by Google

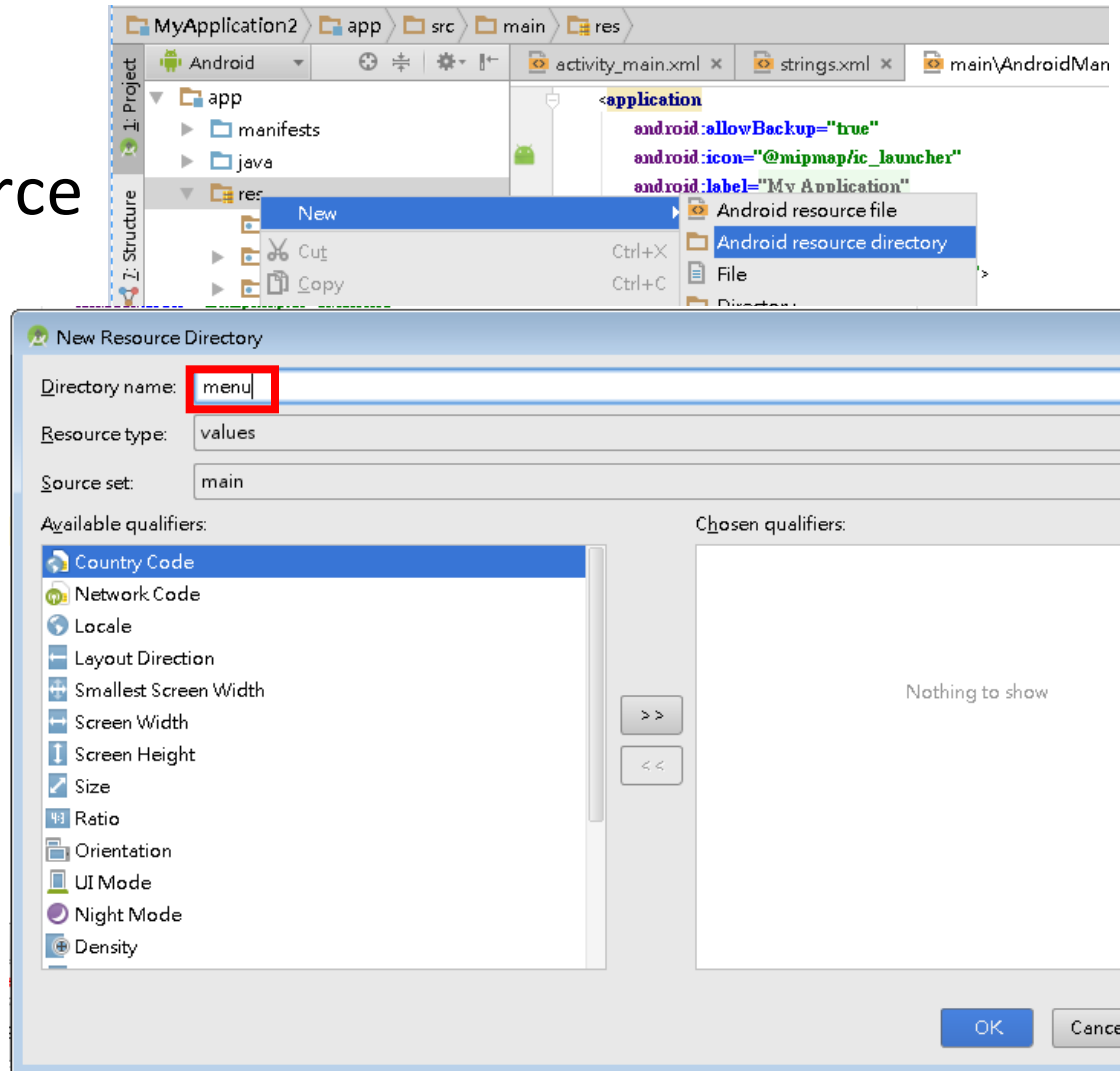


Search the words by Google



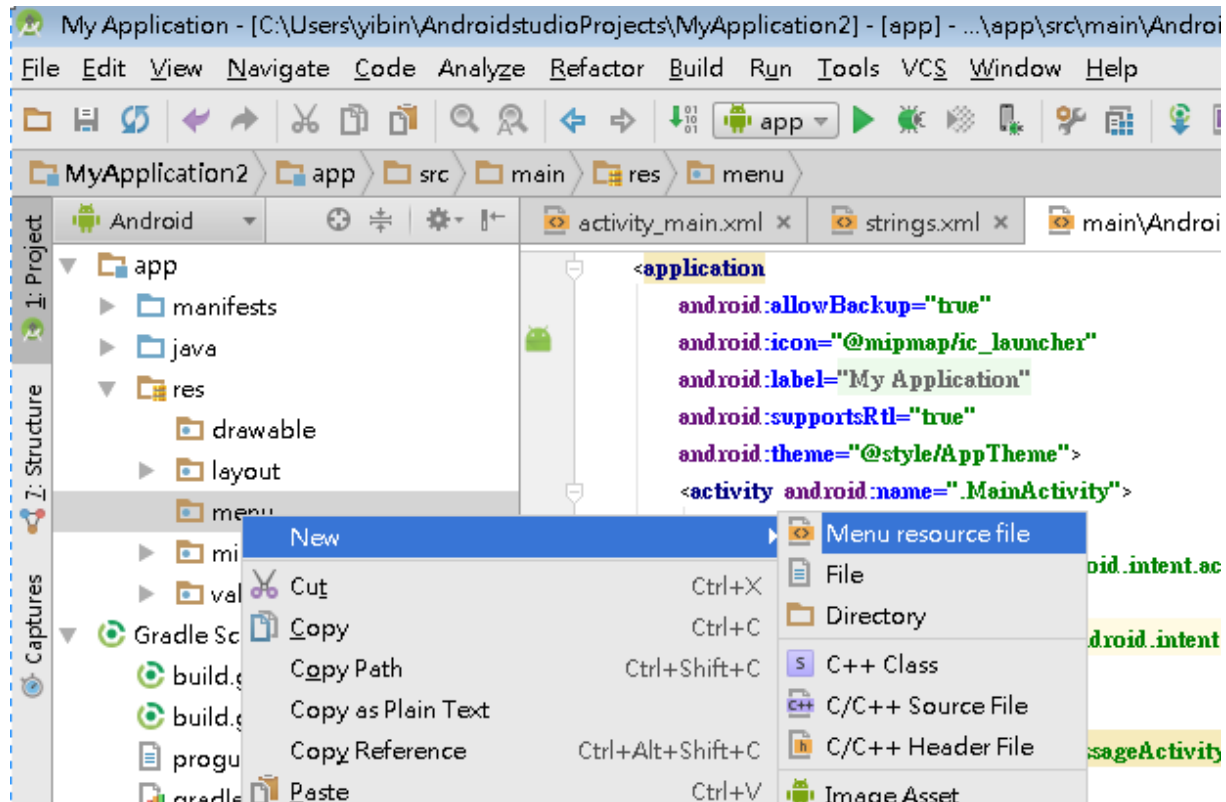
Create a Menu and Add Items

- app/res → New
→ Android resource directory
- Directory name: menu



Create a Menu and Add Items

- app/res/menu → New → Menu resource file (menu_my.xml)

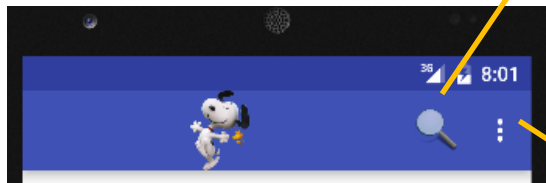


Create a Menu and Add Items

- Edit `app/res/menu/menu_my.xml`

```
<menu
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  tools:context=".MyActivity">
  <item
    android:id="@+id/search"
    android:icon="@drawable/ic_action_search"
    app:showAsAction="ifRoom"
    android:title="search_title"/>
  <item android:id="@+id/action_settings"
    android:title="action_settings"
    android:orderInCategory="100"
    app:showAsAction="never"/>
</menu>
```

Items: Title, Icon, ShowAsAction

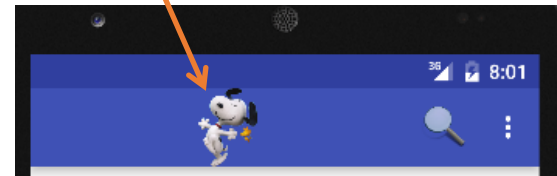


Show Your Action Bar

- `getMenuInflater().inflate(R.menu.menu_my, menu);`
 - Show your `menu_my.xml`


```
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getSupportActionBar().setIcon(R.drawable.ic_nmsl);  
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
    getMenuInflater().inflate(R.menu.menu_my, menu);  
    return true;  
}
```

Show the application icon or not



Handling Clicks on Actions

- When you click an action, the Android system calls your activity's `onOptionsItemSelected()`

```
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.search) {  
        googleIt();  Create this function later  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

Example 2 – Search by Google

- Add “**googleIt**” function in MyActivity.java

```
public void googleIt() {  
    // Do something in response to button  
    EditText editText = (EditText) findViewById(R.id.edit_message);  
    String message = editText.getText().toString();  
    String url = "http://www.google.com/search?q="+message;  
    Intent i = new Intent(Intent.ACTION_VIEW);  
    i.setData(Uri.parse(url));  
    startActivity(i);  
}
```

The url for searching
the message by google

Q&A