

Experiment Report

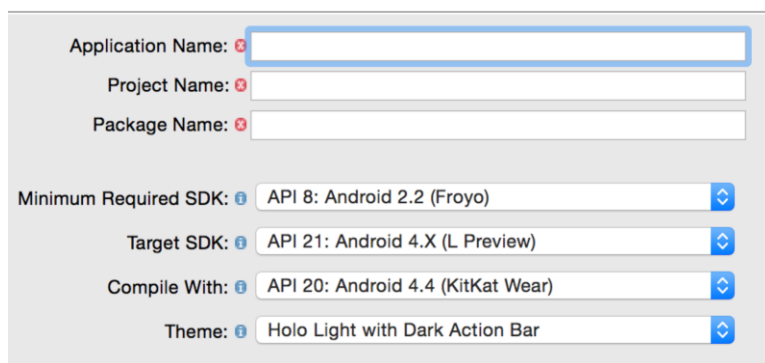
Name	张三	Student ID	3120000419
Exp. Title	MyFirstApp	Exp. Date	2016-11-28

一、Basic Principles (原理简述)

在本次实验中，主要是配置好实验环境，编写第一个 Android App，学习安卓 App 编程的基础知识，包括用户界面和处理用户输入。

配置实验环境详情请看第二部分，不在原理简述中描述。

1、用 Eclipse 创建一个新的 Android Application Project



输入适当的 Application Name，然后在下面的 SDK 选项中，最低要求，目标 sdk，和编译选项均选为 API19，切记千万不能把 Compile With 选的比 Target SDK 高，比如说前面两个选择 API19，第三个选 API20，会导致 res/layout 文件下是空，而没有自动生成的 activity_main.xml 以及对应的 Graphical Layout（这也是由于 Mac 的 Eclipse ADT 版本过高，一开始下载下来的只有 Compile With : API20 导致的，手动下载 API19 即可解决问题，具体看第二部分）

2、运行 APP

具体生成的 project 说明可以具体参照

<http://developer.android.com/training/basics/firstapp/running-app.html>

这里面摘录比较重要的一些参数（目录）说明：

Before you run your app, you should be aware of a few directories and files in the Android project:

`AndroidManifest.xml`

The [manifest file](#) describes the fundamental characteristics of the app and defines each of its components. You'll learn about various declarations in this file as you read more training classes.

One of the most important elements your manifest should include is the `<uses-sdk>` element. This declares your app's compatibility with different Android versions using the `android:minSdkVersion` and `android:targetSdkVersion` attributes. For your first app, it should look like this:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19" />
    ...
</manifest>
```

You should always set the `android:targetSdkVersion` as high as possible and test your app on the corresponding platform version. For more information, read [Supporting Different Platform Versions](#).

`src/`

Directory for your app's main source files. By default, it includes an [Activity](#) class that runs when your app is launched using the app icon.

`res/`

Contains several sub-directories for [app resources](#). Here are just a few:

`drawable-hdpi/`

Directory for drawable objects (such as bitmaps) that are designed for high-density (hdpi) screens. Other drawable directories contain assets designed for other screen densities.

`layout/`

Directory for files that define your app's user interface.

`values/`

Directory for other various XML files that contain a collection of resources, such as string and color definitions.

When you build and run the default Android app, the default [Activity](#) class starts and loads a layout file that says "Hello World." The result is nothing exciting, but it's important that you understand how to run your app before you start developing.

具体运行过程可以看实验步骤（第二部分）

3、建立简单的用户界面

具体用户界面说明可以具体参照

<http://developer.android.com/training/basics/firstapp/building-ui.html>

这里面摘录比较重要的一些参数（目录）说明：

Android provides an XML vocabulary that corresponds to the subclasses of `View` and `ViewGroup` so you can define your UI in XML using a hierarchy of UI elements.

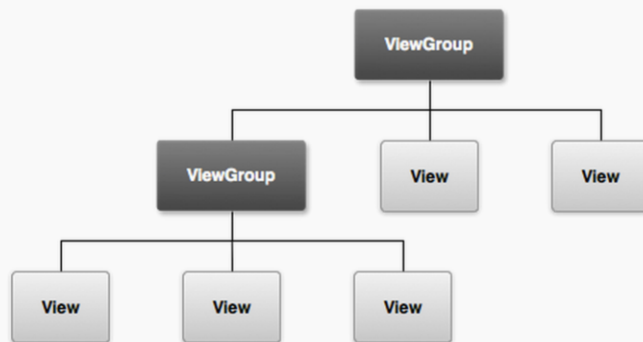


Figure 1. Illustration of how `ViewGroup` objects form branches in the layout and contain other `View` objects.

在 `res/layout/` 目录下，通过修改 `activity_main.xml` 代码，来修改 `<RelativeLayout>` 为 `<LinearLayout>`，并加入 `android:orientation` 修改为水平，随后在 `LinearLayout` 下加入一个 `<EditText>`和`<Button>`，并且修改 String Resource，注意要求要把 `EditText` 的参数设置为：

```
android:layout_weight="1"
android:layout_width="0dp"
```

关于一些参数设置说明：

About these attributes:

`android:id`

This provides a unique identifier for the view, which you can use to reference the object from your app code, such as to read and manipulate the object (you'll see this in the next lesson).

The at sign (`@`) is required when you're referring to any resource object from XML. It is followed by the resource type (`id` in this case), a slash, then the resource name (`edit_message`).

The plus sign (`+`) before the resource type is needed only when you're defining a resource ID for the first time. When you compile the app, the SDK tools use the ID name to create a new resource ID in your project's `gen/R.java` file that refers to the `EditText` element. Once the resource ID is declared once this way, other references to the ID do not need the plus sign. Using the plus sign is necessary only when specifying a new resource ID and not needed for concrete resources such as strings or layouts. See the sidebar for more information about resource objects.

`android:layout_width` and `android:layout_height`

Instead of using specific sizes for the width and height, the `"wrap_content"` value specifies that the view should be only as big as needed to fit the contents of the view. If you were to instead use `"match_parent"`, then the `EditText` element would fill the screen, because it would match the size of the parent `LinearLayout`. For more information, see the [Layouts](#) guide.

`android:hint`

This is a default string to display when the text field is empty. Instead of using a hard-coded string as the value, the `@string/edit_message` value refers to a string resource defined in a separate file. Because this refers to a concrete resource (not just an identifier), it does not need the plus sign. However, because you haven't defined the string resource yet, you'll see a compiler error at first. You'll fix this in the next section by defining the string.

Note: This string resource has the same name as the element ID: `edit_message`. However, references to resources are always scoped by the resource type (such as `id` or `string`), so using the same name does not cause collisions.

设置完毕之后，运行 App

4、建立一个新的 Activity，对应按下 Send 按钮给出回复

（一）设置 Button 被按下时候给的动作： `android:onClick="sendMessage"`

具体说明：

The `android:onClick` attribute's value, `"sendMessage"`, is the name of a method in your activity that the system calls when the user clicks the button.

Open the `MainActivity` class (located in the project's `src/` directory) and add the corresponding method:

```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    // Do something in response to button
}
```

In order for the system to match this method to the method name given to `android:onClick`, the signature must be exactly as shown. Specifically, the method must:

- Be public
- Have a void return value
- Have a `View` as the only parameter (this will be the `View` that was clicked)

Next, you'll fill in this method to read the contents of the text field and deliver that text to another activity.

（二）建立 Intent

An `Intent` is an object that provides runtime binding between separate components (such as two activities). The `Intent` represents an app's "intent to do something." You can use intents for a wide variety of tasks, but most often they're used to start another activity.

An intent not only allows you to start another activity, but it can carry a bundle of data to the activity as well. Inside the `sendMessage()` method, use `findViewById()` to get the `EditText` element and add its text value to the intent:

（三）建立一个新的 Activity

To start an activity, call `startActivity()` and pass it your `Intent`. The system receives this call and starts an instance of the `Activity` specified by the `Intent`.

用 Eclipse 建立一个新的 Activity，名字为 `DisplayMessageActivity`

在 `src/DisplayMessageActivity.java` 目录下，一些参数说明：

Open the `DisplayMessageActivity.java` file. If you used Eclipse to create this activity:

- The class already includes an implementation of the required `onCreate()` method. You will update the implementation of this method later.
- There's also an implementation of the `onCreateOptionsMenu()` method, but you won't need it for this app so you can remove it.
- There's also an implementation of `onOptionsItemSelected()` which handles the behavior for the action bar's Up behavior. Keep this one the way it is.
- There's also a `PlaceholderFragment` class that extends `Fragment`. You will not need this class in the final version of this activity.

Fragments decompose application functionality and UI into reusable modules. For more information on fragments, see the [Fragments API Guide](#). The final version of this activity does not use fragments.

Note: Your activity may look different if you did not use the latest version of the ADT plugin. Make sure you install the latest version of the [ADT plugin](#) to complete this tutorial.

（四）给第二个 Activity 加入一个标题：

修改 `strings.xml` 加入 `<string name="title_activity_display_message">My Message</string>`

（五）由于使用的是 Eclipse，所以不需要修改 `AndroidManifest.xml`，因为已经自动加入了我们新建的第二个 activity 了

（六）接收 Intent

Every `Activity` is invoked by an `Intent`, regardless of how the user navigated there. You can get the `Intent` that started your activity by calling `getIntent()` and retrieve the data contained within it.

In the `DisplayMessageActivity` class's `onCreate()` method, get the intent and extract the message delivered by `MainActivity`:

（七）显示第二个 Activity 的 Message

To show the message on the screen, create a `TextView` widget and set the text using `setText()`. Then add the `TextView` as the root view of the activity's layout by passing it to `setContentView()`.

最后运行 APP.

二、Step-by-Step Procedure (实验步骤)

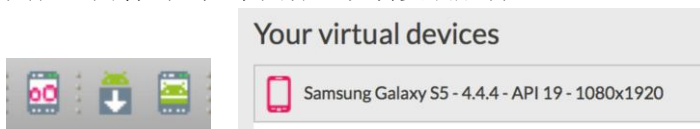
实验环境： Mac OS X 10.10.1
MacBook Pro (Retina, 13-inch, Late 2013)
Processor 2.6 GHz Intel Core i5
Memory 8 GB 1600 MHz DDR3

(一) 这次实验我实在自己的笔记本上面做的，所以没有按照老师给的 win 下的教程来配置环境，具体配置环境的过程如下：

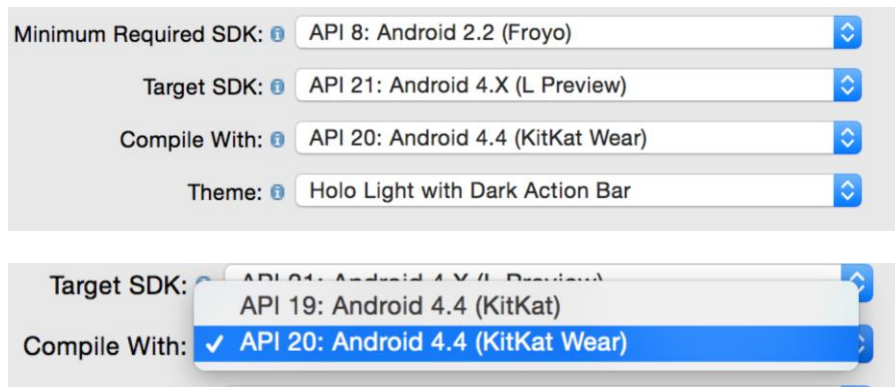
- 1、 下载最新版的 jdk，并安装，打开 terminal 来检验是否成功安装：

```
TroySmileNow:~ SmileNow$ javac -version
javac 1.8.0_20
TroySmileNow:~ SmileNow$ java -version
java version "1.8.0_20"
Java(TM) SE Runtime Environment (build 1.8.0_20-b26)
Java HotSpot(TM) 64-Bit Server VM (build 25.20-b23, mixed mode)
TroySmileNow:~ SmileNow$
```

- 2、 安装 Eclipse 和 Genymotion，并且下载安装 ADT Plugin for Eclipse 23.0.4、Genymotion 下下载一个 API19 的模拟器，安装完毕之后，打开 Eclipse 和 Genymotion，如果能在工具栏上面看到这几个图标，说明安装成功：



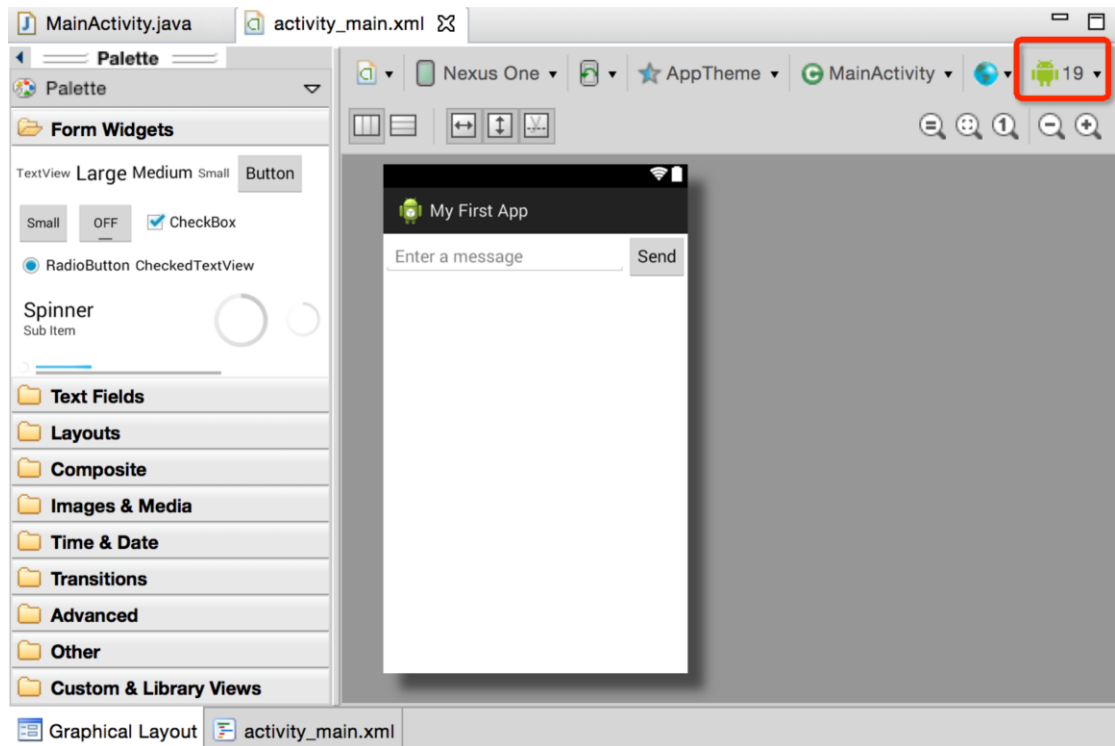
由于下载的 ADT 是新版的，Compile with 选项只有 API20 Android 4.4(KitKat Wear)的，所以必须手动下载 API19 的 SDK（经老师指导），不然后面实验代码输入完毕之后，在 res/layout/activity_main.xml 会无法加载出 Graphical Layout



在 Window->Android SDK Manager 下手动勾选 API19，然后安装：

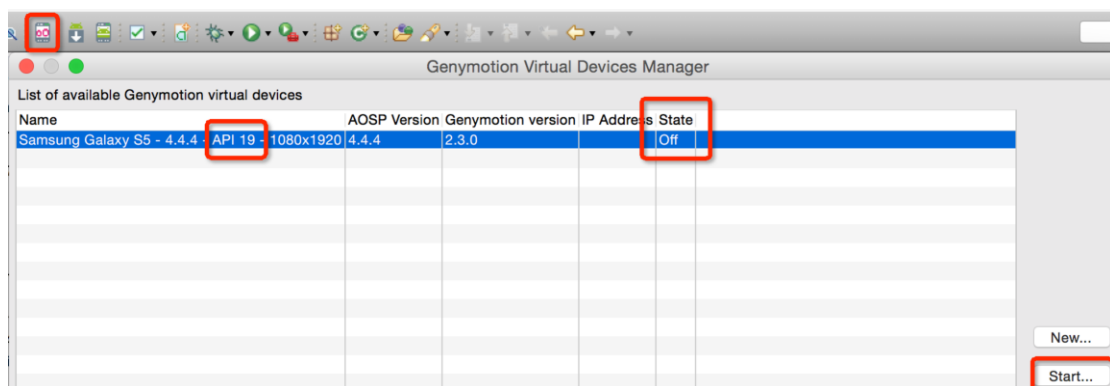


最后在新建的 Android Applicant 的 Graphical Layout 的右上角勾选 API19，就不会出现 Error 了，能够顺利编译了：

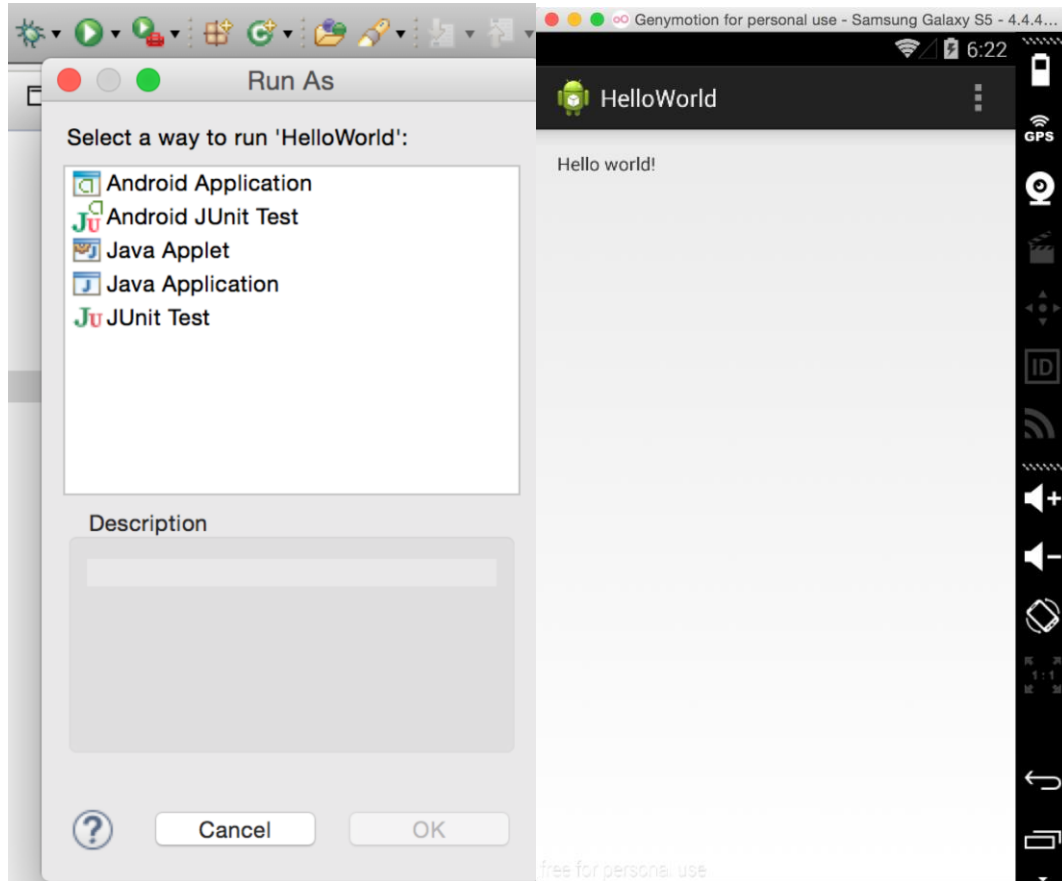


（二）运行 APP

新建工程之后，在 Eclipse 下点击 Genymotion 选项，然后开机



然后点击运行按钮，选择 Android Application，之后切换到 Genymotion 的模拟器，发现是可以成功运行的！



（三）建立简单的用户界面

修改/res/layout/activity_main.xml 代码为:

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

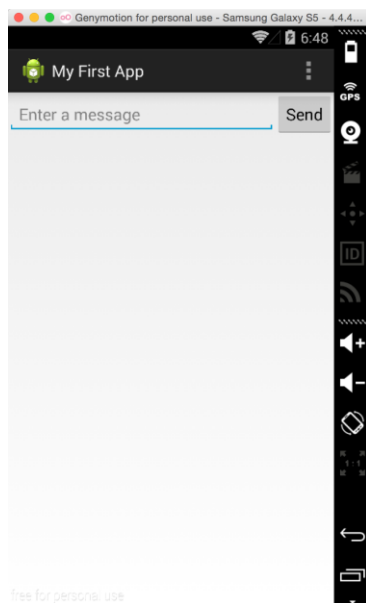
同时修改 res/values/strings.xml 代码为:


```
<?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>
    <string name="action_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="title_activity_display_message">My Message</string>
    <string name="hello_world">Hello world!</string>

</resources>
```

然后运行 App



能够正确地输入，同时可以按下 Send 这个 Button

（四）建立一个新的 Activity，对应按下 Send 按钮给出回复

1、修改/res/layout/activity_main.xml:

```
<Button

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

加入 onClick 动作

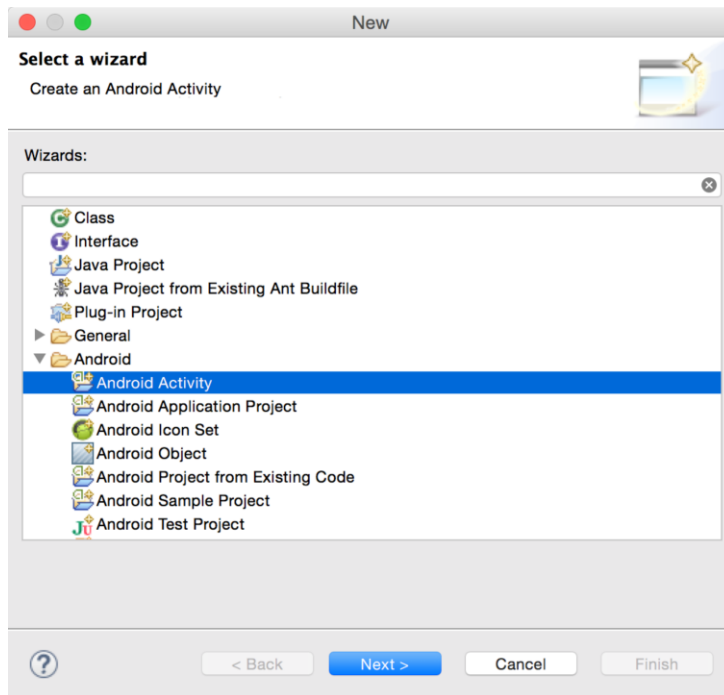
2、建立 Intent

在 src/com.example.myfirstapp/MainActivity.java 中加入 sendMessage 函数，加入 Intent:

```
public void sendMessage(View view) {
    // Do something in response to button
    Intent intent = new Intent(this, DisplayMessageActivity.class);
```

3、建立 DisplayMessageActivity

用 Eclipse 自带的 wizard 来新建一个新的 Activity:



然后修改 `src/com.example.myfirstapp/ DisplayMessageActivity.java` 代码如下:

```
1 package com.example.myfirstapp;
2
3 import android.app.Activity;
4
5
6
7
8
9 public class DisplayMessageActivity extends Activity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         // setContentView(R.layout.activity_display_message);
15         // Get the message from the intent
16         Intent intent = getIntent();
17         String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
18
19         // Create the text view
20         TextView textView = new TextView(this);
21         textView.setTextSize(40);
22         textView.setText(message);
23
24         // Set the text view as the activity layout
25         setContentView(textView);
26     }
27
28     @Override
29     public boolean onOptionsItemSelected(MenuItem item) {
30         // Handle action bar item clicks here. The action bar will
31         // automatically handle clicks on the Home/Up button, so long
32         // as you specify a parent activity in AndroidManifest.xml.
33         int id = item.getItemId();
34         if (id == R.id.action_settings) {
35             return true;
36         }
37         return super.onOptionsItemSelected(item);
38     }
39
40 }
```

4、给第二个 Activity 加入一个标题:

修改 `strings.xml` 加入 `<string name="title_activity_display_message">My Message</string>`

5、由于使用的是 Eclipse，所以不需要修改 `AndroidManifest.xml`，因为已经自动加入了我们新建的第二个 activity 了



6、接收 Intent

修改 src/com.example.myfirstapp/DisplayMessageActivity.java 的 onCreate 函数, 加入这两行代码:

```
9 public class DisplayMessageActivity extends Activity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         //setContentView(R.layout.activity_display_message);
15         // Get the message from the intent
16         Intent intent = getIntent();
17         String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
18     }
19 }
```

7、显示第二个 Activity 的 Message

修改 src/com.example.myfirstapp/DisplayMessageActivity.java 的 onCreate 函数, 加入 TextView, 最后代码如下:

```
9 public class DisplayMessageActivity extends Activity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         //setContentView(R.layout.activity_display_message);
15         // Get the message from the intent
16         Intent intent = getIntent();
17         String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
18
19         // Create the text view
20         TextView textView = new TextView(this);
21         textView.setTextSize(40);
22         textView.setText(message);
23
24         // Set the text view as the activity layout
25         setContentView(textView);
26     }
27 }
```

最后运行 APP 即可。

三、Results and Analysis (结果与分析)

- (一) 从第二部分的实验步骤来看，配置环境正确，并且能成功运行 Android Application
- (二) 运行 HelloWorld APP 正确！
- (三) 能够顺利运行 My First App，可以在 EditText 栏输入，同时可以按下 Button 键，实验正确！
- (四) 最后运行 APP，输入一些字符串，然后按下 Send 按钮，跳转到第二个 Activity，显示刚刚输入的信息，实验正确！

