Android 应用界面布局实验

一、 实验目的

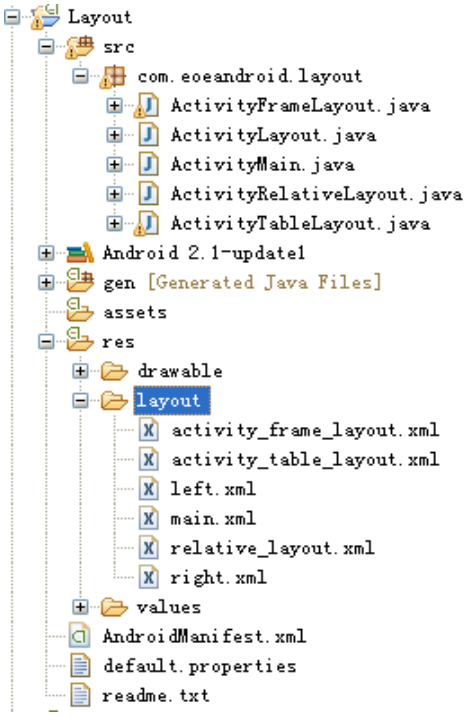
1.了解 FrameLayout 布局
2.了解 LinearLayout 布局
3.了解 RelateLayout 布局
4.了解 TableLayout 布局
5.了解 UI 事件处理方式
6.了解 Intent 的使用

二、 实验条件

- ✓ IBM-PC 兼容机
- ✓ Windows、Ubuntu11.04 或其他兼容的 Linux 操作系统
- ✓ JDK（建议安装 JDK8 及其以上版本）、Android Studio 或 Eclipse with ADT
- ✓ INTEL ATOM 平板

三、 Demo

若使用 Eclipse 开发工具，可以通过
File --> Import --->General --->Existing Projects into Workspace
将 Layout 工程文件夹导入 Eclipse 中。
在 Eclipse 里可以看到如下的工程文件



实例操作演示

1.此布局程序运行后的主界面如图 1.1 所示
2.单击第一个按钮后出现的一个地图的界面，如图 1.2 所示

图 1.1

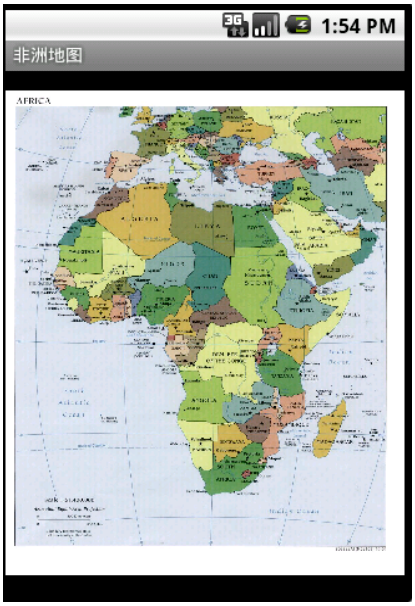


图 1.2

3.单击第二个按钮后出现的界面，如图 1.3 所示。
4.单击第三个按钮后出现的界面，如图 1.4 所示。
5.单击第四个按钮后出现的界面，如图 1.5 所示。

图 1.3

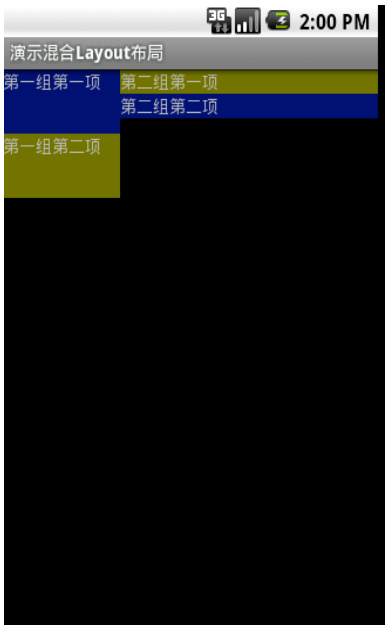


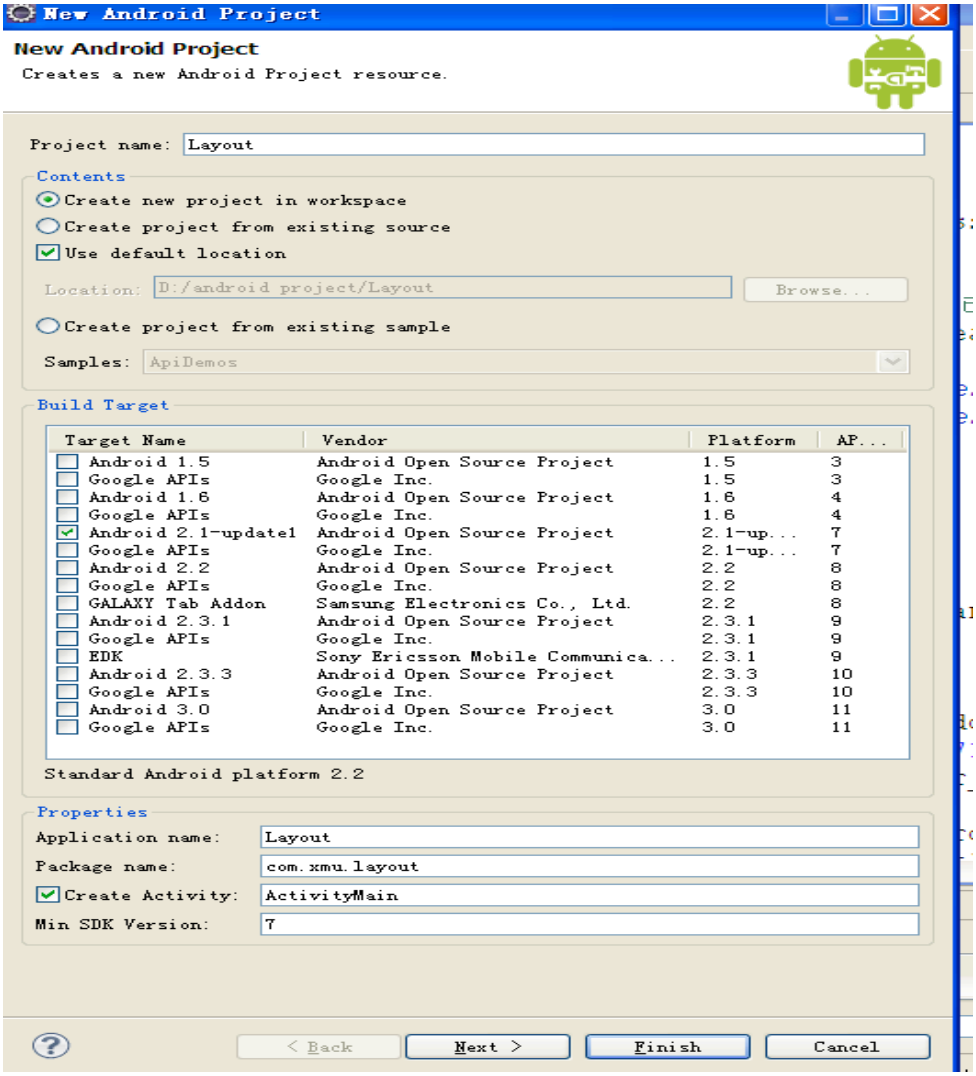
图 1.4

图 1.5

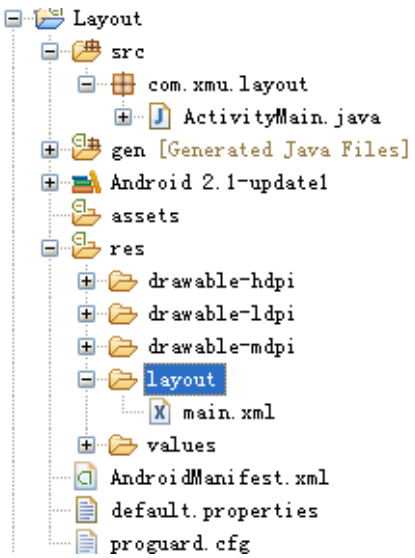
四、 实验步骤

1. 创建工程

我们开始动手写这一个程序。
首先，先创建一个工程 Layout，可以通过 File ----> Android Project 创建新的工程。



点击 Finish.后就可以看到如下目录结构



首先在 com.xmu.layout 包里新增四个 Activity（即添加四个类，这些类继承于 android.app.Activity)。
ActivityLayout.java
ActivityFrameLayout.java
ActivityRelativeLayout.java
ActivityTableLayout.java

并在 res/layout 文件夹下新增 5 个 xml 布局文件。（注意，在创建 xml 文件的时候，文件名不可以包含大写字母）

activity_frame_layout.xml
activity_table_layout.xml
Left.xml
relative_layout.xml
Right.xml

然后，还需要往 AndroidMainfest.xml 文件里添加这四个 activity 的的描述，代码如下：

```
<activity android:name=".ActivityLayout"
    android:label="演示混合Layout布局">
</activity>
<activity android:name=".ActivityRelativeLayout"
    android:label="演示RelativeLayout布局">
</activity>
<activity android:name=".ActivityFrameLayout"
    android:label="非洲地图">
</activity>
<activity android:name=".ActivityTableLayout"
    android:label="TableLayout布局">
</activity>
```

这些代码要添加在 application 节点下面，可以不必考虑位置关系。

这样的话，整个工程算是初步建立完毕，接下来就是要往里面添加一些代码，以使得整个工程连贯起来。

下面先来看看 LinearLayout 布局。

2. LinearLayout

ActivityMain 的布局由 main.xml 文件描述，它是系统自动创建的布局文件。在这个 activity 中我们说明 LinearLayout 的使用。

Main.xml

然后再 res/layout 文件夹下可以看到一个 main.xml,这是系统自动创建的布局文件。
修改该 layout 文件，定义一些 button，代码如下

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent">
<Button
android:id="@+id/button0"
android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="非洲草原地图：FrameLayout的使用"
/>
<Button
android:id="@+id/button1"
android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="个性化表单:RelativeLayout的使用"
/>
<Button android:id="@+id/button2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="LinearLayout和RelativeLayout互助使用" />
<Button
android:id="@+id/button3"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="整齐的表单：TableLayout的使用" />
</LinearLayout>
```

代码解释：

这是一个典型的 LinearLayout 布局的使用。LinearLayout 是一种 Android 中最常用的布局之一，它将自己包含的子元素按照一个方向进行排列，方向有两种，水平或者竖直。这个方向可以通过设置 android:orientation="vertical"或者 android:orientation="horizontal"来实现，所有的元素排列都是一个接着一个的。如果是竖直排列，那么 LinearLayout 的元素就一个接着一个的从上到下竖直排列，例如，在 ActivityMain 的视图中，就是这样竖直的一个一个接着排列的。如果是水平排列，那么就是 LinearLayout 里面的子元素从左到右一个一个的进行排列的。

- Android:id 定义组件的 id，也就是名字，在应用程序当中，我们通过 id 就可以访问到所定义的元素。
- Android:layout_width="fill_parent" 表示布局可以在 x 轴方向填充父容器的空间，类似的 Android:layout_height="fill_parent"表示布局可以在 y 轴方向填充父容器的空间。
- Android:layout_height="wrap_content" 表示 ImageView 元素的长度只需要将该 widget 包裹起来即可，并不需要填充父容器。

关于 LinearLayout 的更多相关信息，可以查看
docs/reference/android/widget/LinearLayout.html
docs/reference/android/view/ViewGroup.LayoutParams.html

ActivityMain.java

在创建项目时，ActivityMain.java 已自动生成，它的代码如下

```
public class ActivityMain extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    }
}
```

为了使这个 activity 中的 button 能响应用户的点击操作，我们需给该类添加一些成员变量。这些成员将用来引用界面上的元素。

```
    OnClickListener listener0 = null;
    OnClickListener listener1 = null;
    OnClickListener listener2 = null;
    OnClickListener listener3 = null;
    Button button0;
    Button button1;
    Button button2;
    Button button3;
```

接着，给界面上的 button 添加事件，主要是在 onCreate 函数里添加处理代码。最后代码如下。

```
public class ActivityMain extends Activity {
    OnClickListener listener0 = null;
    OnClickListener listener1 = null;
    OnClickListener listener2 = null;
    OnClickListener listener3 = null;
    Button button0;
    Button button1;
    Button button2;
    Button button3;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
         /*创建OnClickListener对象*/
        listener0 = new OnClickListener() {
            public void onClick(View v) {
                Intent intent0 = new Intent(ActivityMain.this,
                        ActivityFrameLayout.class);
                setTitle("FrameLayout");
                startActivity(intent0);
            }
        };
        listener1 = new OnClickListener() {
            public void onClick(View v) {
                Intent intent1 = new Intent(ActivityMain.this,
                        ActivityRelativeLayout.class);
                startActivity(intent1);
            }
        };
        listener2 = new OnClickListener() {
            public void onClick(View v) {
                setTitle("这是在ActivityLayout");
                Intent intent2 = new Intent(ActivityMain.this,
                        ActivityLayout.class);
                startActivity(intent2);

            }
        };
        listener3 = new OnClickListener() {
            public void onClick(View v) {
                setTitle("TableLayout");
                Intent intent3 = new Intent(ActivityMain.this,
```

```
                    ActivityTableLayout.class);
            startActivity(intent3);

        }
    };
    setContentView(R.layout.main);
    /*从layout中找出button对象，并注册事件监听对象*/
    button0 = (Button) findViewById(R.id.button0);
    button0.setOnClickListener(listener0);
    button1 = (Button) findViewById(R.id.button1);
    button1.setOnClickListener(listener1);
    button2 = (Button) findViewById(R.id.button2);
    button2.setOnClickListener(listener2);
    button3 = (Button) findViewById(R.id.button3);
    button3.setOnClickListener(listener3);
  }
}
```

代码解释：

- setContentView(R.layout.main);负责将当前的 Activity 与 main.xml 布局文件关联。
- findViewById()得到 4 个 Button 的引用，并且给 Button 设置单击监听器。Button 是最常用的组件之一。它就是一个按钮，一般需要设置监听器来处理单击事件。
- 每一个监听器都是 跳转到一个新的 Activity.
- Intent 用于协助完成各个组件之间的通信。它负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android 则根据此 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。代码中所用的 Intent 的构造函数为 public Intent (Context packageContext, Class<?> cls)，这个类型的构造函数将为指定的组件创建一个 Intent 对象，该 Intent 的意图为执行这个类对象。

此时运行后的结果如下。

注：也可以在 main.xml 文件中描述 button 的 onclick 事件处理。

3. FrameLayout

在 ActivityFrameLayout 中使用 FrameLayout，首先修改 activity_frame_layout.xml 文件的代码如下。

activity_frame_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout android:id="@+id/left"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <ImageView android:id="@+id/photo" android:src="@drawable/bg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"  />
</FrameLayout>
```

代码解释：

- FrameLayout

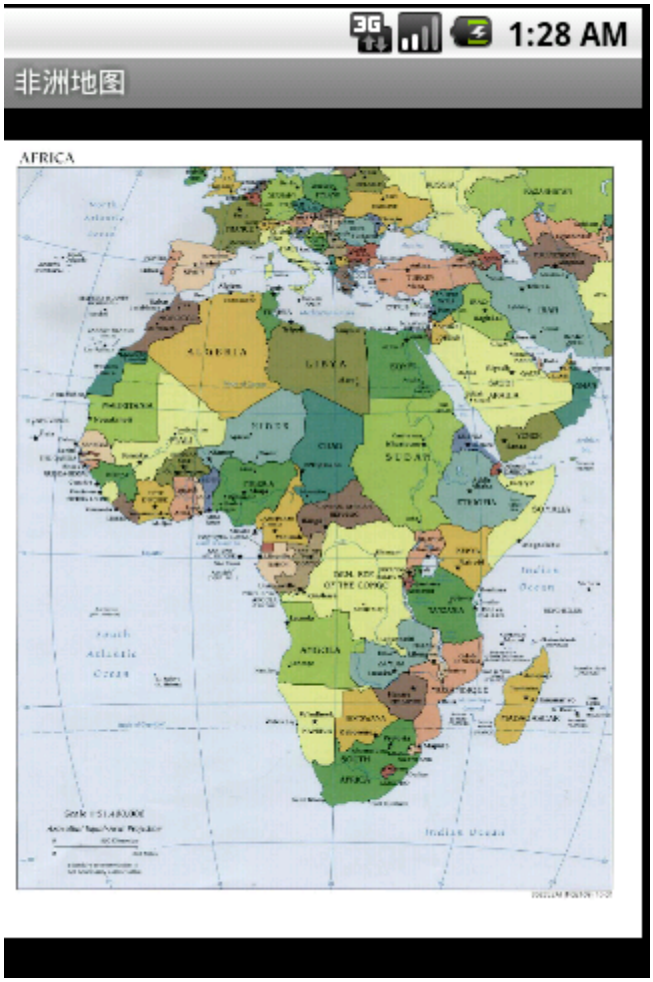
一个 FrameLayout 对象就好比一块在屏幕上提前预定好的空白区域，然后可以填充一些元素到里面，比方说一张图片等。需要注意的是，所有的元素都被放置在 FrameLayout 区域最左上的区域。而且无法为这些元素指定一个确切的位置。如果一个 FrameLayout 里边有多个子元素，那么后边的子元素的显示会重叠在前一个元素上。

- Android:layout_width="fill_parent"和 Android:layout_height="fill_parent"表示 FrameLayout 布局可以在 x 轴方向和 y 轴方向填充父容器的空间。
- Android:layout_width="wrap_content" 和 Android:layout_height="wrap_content" 表示 ImageView 元素的长和宽只需要将 bg.jpg 包裹起来即可，并不需要填充父容器。

在 ActivityFrameLayout.java 文件添加 onCreate 事件重载的代码，将 Activity 与 activity_frame_layout.xml 的布局关联起来。

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);;
    setContentView(R.layout.activity_frame_layout);
}
```

运行程序后，单击第一个按钮，会看到



4. RelativeLayout

在 relative_layout.xml 里添加如下代码。

relative_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Demonstrates using a relative layout to create a form -->
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:background="@drawable/blue" android:padding="10dip">
    <TextView android:id="@+id/label" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="请输入用户名: " />
```

```
    <!--
        这个EditText放置在上边id为label的TextView的下边
    -->
    <EditText android:id="@+id/entry" android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label" />
    <!--
        取消按钮和容器的右边齐平，并且设置左边的边距为10dip
    -->
    <Button android:id="@+id/cancel" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip" android:text="取消" />
    <!--
        确定按钮在取消按钮的左侧，并且和取消按钮的高度齐平
    -->
    <Button android:id="@+id/ok" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/cancel"
        android:layout_alignTop="@id/cancel" android:text="确定" />
</RelativeLayout>
```

代码解释：

- RelativeLayout

  从类的名字可以猜测出，这是一个相对布局类，即首先RelativeLayout是一个容器，它里边的元素，如Button按钮等的位置是按照相对位置来计算的，例如，有两个Button按钮都布局在一个relativeLayout里边，我们可以定义第二个Button在第一个Button的上边或者是右边。但到底第二个Button在什么位置呢，它还是依赖于第一个Button的位置。需要注意的是，出于性能上的考虑，对于相对布局的精确位置的计算只会执行一次，所以，如果一个可视化组件B依赖于A,那么必须让A出现在B的前边。

- Android:id定义组件的id。
- Android:layout_width定义组件的宽度。Android:layout_height定义了组件的高度。
- Android:background="@drawable/blue"定义组件的背景，在这里系统会解析到"@drawable/blue"的值为#770000ff,这是颜色的RGB码。
- Android:padding="10dip",dip的意思是依赖于设备的像素，是描述区域大小的一种单位。和html类似，在android中也有padding和margin的概念。Padding表示填充，margin表示边距。
- Android:layout_below="@id/label",按照字面的意思比较容易理解，就是将当前组件放置于id为label的组件的下方。这就是经典的相对布局。这种布局就是的好处就是不用关注很多细节，而且它的适配性很强，在不同大小的屏幕或者手机设备上都可以使用。所以，我们在android的布局中，推荐这种相对布局，而不推荐按照像素精确布局。
- Android:layout_alignParentRight="true",这个也可以从字面上来理解，就是和父容器的右边齐平，这也就是相对布局。

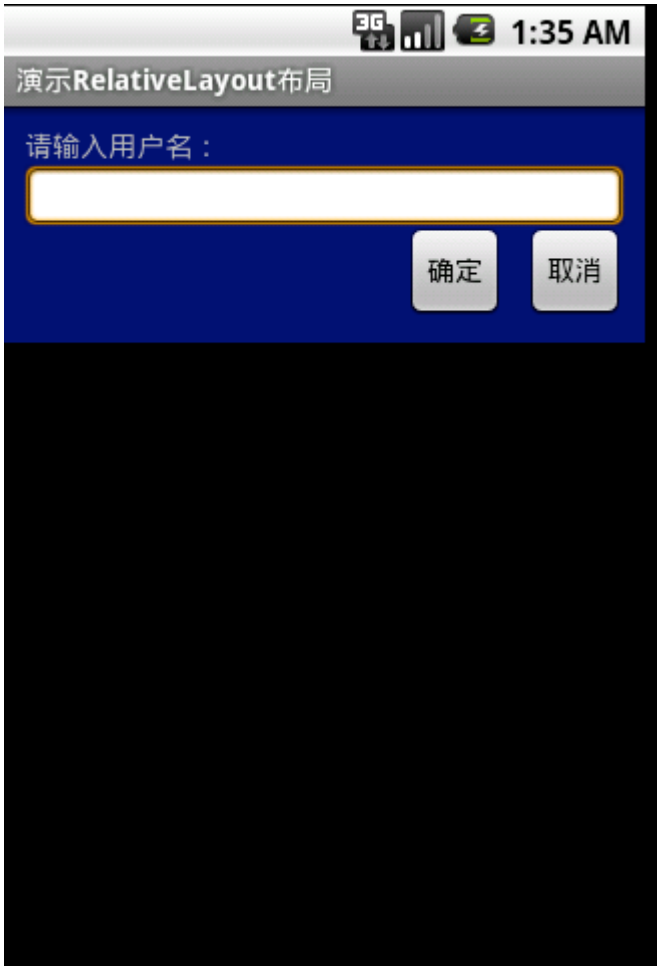
- Android:layout_marginLeft="10dip",设置id为Cancel的Button的左边距为10dip.
- Android:layout_toLeftOf="@id/cancel",设置此组件在id为Cancel的组件的左边。
- Android: layout_alignTop="@id/cancel"，设置此组件和id为Cancel组件的高度齐平。

在 ActivityRelativeLayout.java 文件里添加代码， 重载 onCreate 事件，将 Activity 与 relative_layout.xml 的布局关联起来。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.relative_layout);

}
```

运行后，单击第二个按钮，可以看到如下。



5. TableLayout

在 activity_table_layout.xml 添加如下代码

activity_table_layout.xml

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView android:text="用户名:" android:textStyle="bold"
            android:gravity="right" android:padding="3dip" />
        <EditText android:id="@+id/username" android:padding="3dip"
            android:scrollHorizontally="true" />
    </TableRow>
    <TableRow>
        <TextView android:text="登录密码:" android:textStyle="bold"
            android:gravity="right" android:padding="3dip" />
        <EditText android:id="@+id/password" android:password="true"
            android:padding="3dip" android:scrollHorizontally="true" />
    </TableRow>
    <TableRow android:gravity="right">
        <Button android:id="@+id/cancel"
            android:text="取消" />
        <Button android:id="@+id/login"
            android:text="登录" />
    </TableRow>
</TableLayout>
```

代码解释：

- TableLayout

  从字面上了解 TableLayout 是一种表格式的布局。这种布局会把包含的元素以行和列的形式进行排列。表格的列数为每一行的最大列数。当然表格里边的单元格可以为空。
- TableLayout 标签定义了一个表格布局。
- TableRow 标签定义了表格布局里边的一行。每一行里边可以自由的加入一些组件，比如上边我们主要添加了按钮组件和文本框组件。

在 ActivityTableLayout.java 文件里添加代码， 重载 onCreate 事件，将 Activity 与 activity_table_layout.xml 的布局关联起来。

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);;
        setContentView(R.layout.activity_table_layout);

    }
```

运行后，单击最后一个按钮，可以看到如下。

五、 附加实验

1. 试着修改各种 Layout 的属性，例如 `orientation`，`layout_width`，`layout_height` 等属性的值，看看实际的布局效果有何改变
2. UI 的事件处理是程序与用户交互的一种重要手段，本实验用到了 button 的 onclick 事件处理，它也可以在 main.xml 文件中描述，请参考 docs/reference/android/widget/Button.html，用 button 的 android:onClick 属性，修改 button 的 OnClick 事件处理。

六、 参考资料

docs/guide/topics/ui/layout-objects.html
docs/guide/topics/ui/declaring-layout.html

docs/guide/topics/ui/index.html
docs/reference/android/widget/LinearLayout.html
docs/reference/android/view/ViewGroup.LayoutParams.html
docs/guide/topics/intents/intents-filters.html
docs/reference/android/content/Intent.htm
docs/guide/topics/resources/layout-resource.html

七、 实验报告要求

实验报告中要包含以下几个部分：

1、实验目的
2、实验条件
3、实验原理
4、实验步骤分析
5、实验结果与总结
6、实验思考题

实验步骤要详细，关键步骤要有截图，运行结果也要有截图。