Dialog

一、 实验目的

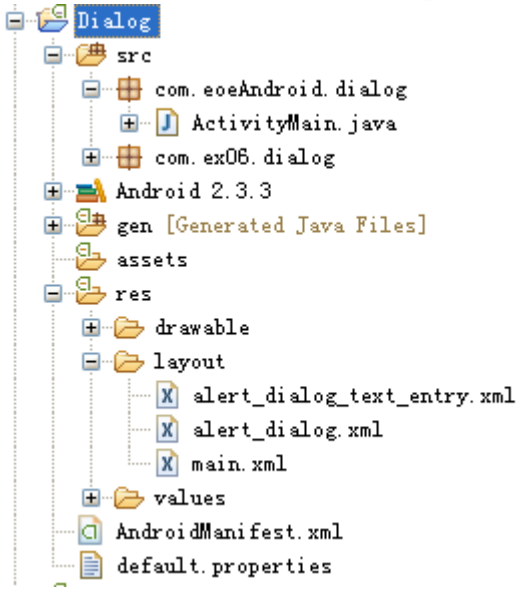
了解 Dialog 的使用
了解资源文件的使用

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

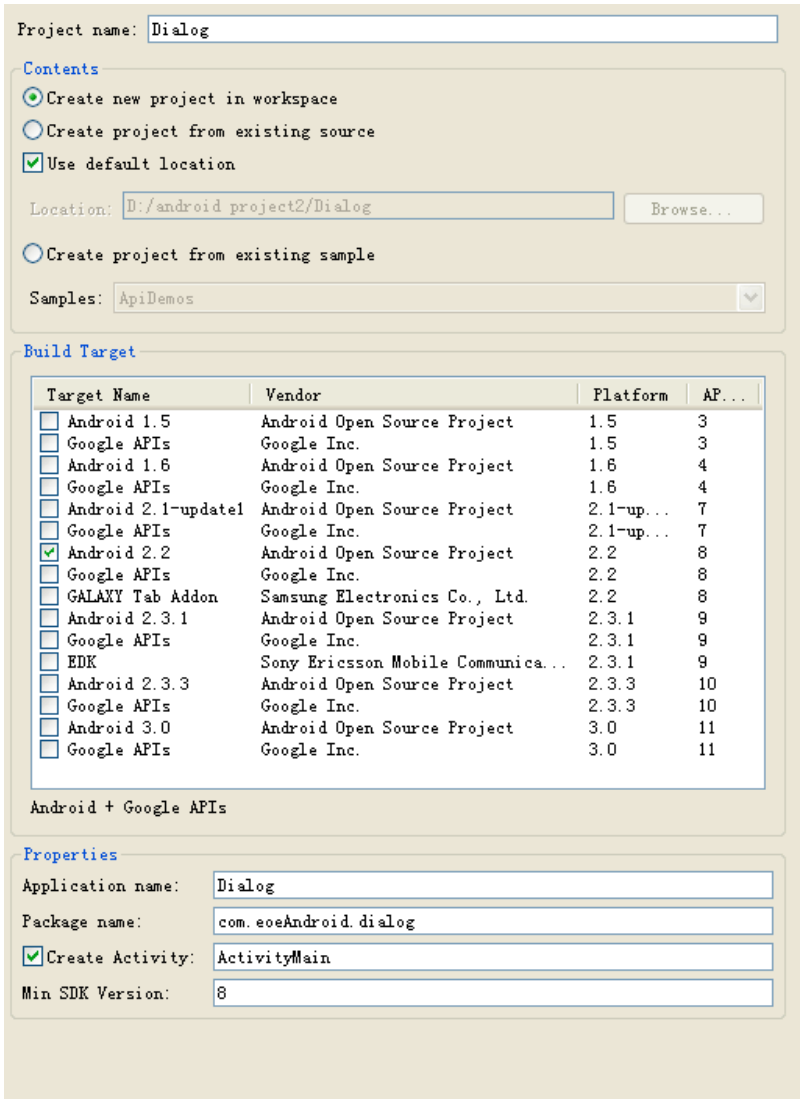
若使用 Eclipse 开发工具，可以通过
File --> Import --->General --->Existing Projects into Workspace
然后将 Dialog 工程文件夹导入 Eclipse 中。



四、 实验步骤

1. 创建工程

首先，建立一个 Dialog 的 android 工程，设置如下



ActivityMain.java

在这个实验中，我们将学习 dialog 的使用，这个实验包括四个对话框，第一个是包含 ok 和 cancel 按键的对话框，第二个是包含三个按键的对话框，第三个是包含输入框的对话框，第四个对话框显示一个进度条。修改 ActivityMain.java 文件内容如下：

```
public class ActivityMain extends Activity {
    private static final int DIALOG1 = 1;
    private static final int DIALOG2 = 2;
    private static final int DIALOG4 = 4;
    private static final int DIALOG3 = 3;



    /****
     * 生成一个显示信息的对话框，显示的内容布局可以从 dialogView 里获取
     *
```

```
     * @param dialogView
     * @param titleResourcesId
     */
    public    void showDialogInfo( String title) {
        LayoutInflater inflater = LayoutInflater.from(this);
        final View textEntryView = inflater.inflate(
                R.layout.alert_dialog_text_entry, null);
        Context context = this;
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setTitle(title);
        builder.setView(textEntryView);
        builder.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                    }
                });
        builder.setCancelable(false);
        builder.create();
        builder.show();
        context = null;
        builder = null;
    }

    public void showProcessDialog(){
        final ProgressDialog dialog = new ProgressDialog(this);
        dialog.setTitle("正在下载歌曲");
        dialog.setMessage("请稍候……");
        dialog.show();
        new Thread(new Runnable(){
            @Override
            public void run() {
                try {
                    Thread.sleep(3*1000);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                dialog.dismiss();
            }}).start();
    }


    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
```

```
        case DIALOG1:
            return buildDialog1(ActivityMain.this);
        case DIALOG2:
            return buildDialog2(ActivityMain.this);
        case DIALOG3:
            return buildDialog3(ActivityMain.this);
        case DIALOG4:
            return buildDialog4(ActivityMain.this);
        }
        return null;
    }

    protected void onPrepareDialog(int id, Dialog dialog) {
        if (id == DIALOG1) {
            setTitle("测试");
        }
    }

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

        Button button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG1);
            }
        });

        Button buttons2 = (Button) findViewById(R.id.buttons2);
        buttons2.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialogInfo("hello world");
            }
        });

        Button button3 = (Button) findViewById(R.id.button3);
        button3.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG3);
            }
        });

        Button button4 = (Button) findViewById(R.id.button4);
```

```
        button4.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                //showDialog(DIALOG4);
                showProcessDialog();
            }
        });
    }

    private Dialog buildDialog1(Context context) {
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setIcon(R.drawable.alert_dialog_icon);
        builder.setTitle(R.string.alert_dialog_two_buttons_title);
        builder.setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的确定按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的取消按钮");
                    }
                });
        return builder.create();

    }

    private Dialog buildDialog2(Context context) {
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setIcon(R.drawable.alert_dialog_icon);
        builder.setTitle(R.string.alert_dialog_two_buttons_msg);
        builder.setMessage(R.string.alert_dialog_two_buttons2_msg);
        builder.setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的确定按钮");
                    }
                });
        builder.setNeutralButton(R.string.alert_dialog_something,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的进入详细按钮");
                    }
                });
```

```
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {

                        setTitle("点击了对话框上的取消按钮");
                    }
                });
        return builder.create();
    }

    private Dialog buildDialog3(Context context) {
        LayoutInflater inflater = LayoutInflater.from(this);
        final View textEntryView = inflater.inflate(
                R.layout.alert_dialog_text_entry, null);
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setIcon(R.drawable.alert_dialog_icon);
        builder.setTitle(R.string.alert_dialog_text_entry);
        builder.setView(textEntryView);
        builder.setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的确定按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的取消按钮");
                    }
                });
        return builder.create();
    }

    private Dialog buildDialog4(Context context) {
        ProgressDialog dialog = new ProgressDialog(context);
        dialog.setTitle("正在下载歌曲");
        dialog.setMessage("请稍候……");
        return dialog;
    }
}
```

2. 修改布局文件

在 res/layout 添加如下文件。

alert_dialog_text_entry.xml
alert_dialog.xml

对布局文件进行修改。alert_dialog_text_entry.xml 的内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:orientation="vertical">
<TextView android:id="@+id/username_view"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" android:layout_marginLeft="20dip"
    android:layout_marginRight="20dip" android:text="用户名"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<EditText android:id="@+id/username_edit"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" android:layout_marginLeft="20dip"
    android:layout_marginRight="20dip" android:capitalize="none"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<TextView android:id="@+id/password_view"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" android:layout_marginLeft="20dip"
    android:layout_marginRight="20dip" android:text="密码"
    android:textAppearance="?android:attr/textAppearanceMedium" />
<EditText android:id="@+id/password_edit"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" android:layout_marginLeft="20dip"
    android:layout_marginRight="20dip" android:capitalize="none"
    android:password="true"
    android:textAppearance="?android:attr/textAppearanceMedium" />
</LinearLayout>
```

alert_dialog.xml 修改如下：

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/screen" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:orientation="vertical">
<LinearLayout android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:orientation="vertical">
<Button android:id="@+id/button1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="两个 button 的对话框 1" />
<Button android:id="@+id/buttons2"
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content" android:text="三个 button 的对话框 2" />
<Button android:id="@+id/button3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="可以进行输入的对话框" />
<Button android:id="@+id/button4"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" android:text="进度框" />
</LinearLayout>
</ScrollView>
```

3. icon 资源

Icon 是种图像资源，在 Android 项目中，图像资源放在 res/drawable 目录中，另外，也可以根据手机屏幕的分辨率将不同分辨率的图像资源文件分别放在 res\drawable-hdpi（高分辨率）、res\drawable-ldpi（低分辨率）和 res\drawable-mdpi（中分辨率）目录下。本实验将 icon 图片文件 alert_dialog_icon.png 放在 res/drawable 目录下，说明无论是哪种分辨率，都将使用这个图片。

4. 字符串资源

字符串可以放在代码也可以放在资源文件 string.xml 中，通常将字符串资源放在资源文件中有利于增加项目的可维护性，同时也有利于国际化的实现。本实验的 string.xml 文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ActivityMain</string>
    <string name="app_name">对话框</string>
    <string name="alert_dialog_two_buttons_title">
        这是一个提示框。 点击取消后可以返回。
    </string>
    <string name="alert_dialog_two_buttons_msg">标题部分，可以自定义</string>
    <string name="alert_dialog_two_buttons2_msg">
```

对话框虽然在我们的程序当中不是必备的，但是用好对话框能对我们编写的应用增色不少。采用对话框可以大大的增强应用的友好性。在这个用户至上的时代，如果你的应用没有了用户，也就没有多大意义的。所以我们要学好对话框，并在适当的地方灵活运用。比较常用的场景是，用户登录、网络正在下载，下载成功或失败的提示。

```
    </string>

    <string name="alert_dialog_ok">确定</string>

    <string name="alert_dialog_hide">隐藏</string>

    <string name="alert_dialog_something">进入详细</string>

    <string name="alert_dialog_cancel">取消</string>

    <string name="alert_dialog_text_entry">请输入</string>

</resources>
```

5. Dialog

Dialog 类，是一切对话框的基类。需要注意的是，Dialog 类虽然可以在界面上显示，但是并非继承于习惯用的 View 类，而是直接从 java.lang.Object 开始构造出的。类似于 Activity,Dialog 也是有生命周期的，它的生命周期由 Activity 来维护。Activity 负责生成，保存，恢复它。在生命周期的每个阶段都有一些回调函数供系统反向调用。

在 Activity 当中可以主动调用的函数为：

1. showDialog(int id)，负责显示标识为 id 的 dialog。这个函数如果调用后，系统将方向调用 Dialog 的回调函数 onCreateDialog(int id)。
2. dismissDialog(int id),使标识为 id 的 dialog 在界面当中消失。

Dialog 有两个比较常见的回调函数，onCreateDialog(int id)和 onPrepareDialog(int id, Dialog dialog).当在 Activity 当中调用了 showDialog(int id)后，如果这个 Dialog 是第一次生成，系统将反向调用 Dialog 的回调函数 onCreateDialog(int id)，然后再调用 onPrepareDialog(int id,Dialog dialog)；如果这个 Dialog 已经生成，只不过还没有显示出来，那么将不会回调 onCreateDialog(int id),而是直接回调 onPrepareDialog(int id, Dialog dialog)方法。

onPrepareDialog(int id, Dialog dialog)方法提供了这样的一套机制，即当 Dialog 生成但是没有显示出来的时候，使得有机会在显示前对 Dialog 做一些修改，如对 Dialog 标题进行修改等。

了解了 Dialog 的一些基本知识后，来看 ActivityMain.java 文件当中 onCreate()方法里边的实现代码：

```
    Button button1 = (Button) findViewById(R.id.button1);
    button1.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            showDialog(DIALOG1);
        }
        });
```

代码解释：

- findViewById 方法通过组件的 id 返回这个组件的引用。
- setOnClickListener 方法为 button1 设置了一个单击侦听器。
- onClick（）为单击 Button 后的回调函数。
- showDialog()为 Activity 里边的函数。负责将 id 为 DIALOG1 的 Dialog 显示出来。

执行 showDialog(DIALOG1)代码后，系统执行回调函数 onCreateDialog(),实现代码如下所示：

```
    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case DIALOG1:
            return buildDialog1(ActivityMain.this);
        case DIALOG2:
            return buildDialog2(ActivityMain.this);
        case DIALOG3:
            return buildDialog3(ActivityMain.this);
        case DIALOG4:
            return buildDialog4(ActivityMain.this);
        }
        return null;
    }
```

代码解释：

- 针对不同的 Dialog 的 id，生成不同的 Dialog.
- buildDialog1 函数生成第一个要显示的 Dialog.

接下来看一下 buildDialog1()函数，其实现代码如下所示：

```
    private Dialog buildDialog1(Context context) {
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setIcon(R.drawable.alert_dialog_icon);
        builder.setTitle(R.string.alert_dialog_two_buttons_title);
        builder.setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的确定按钮");
                    }
                });
        builder.setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        setTitle("点击了对话框上的取消按钮");
                    }
                });
        return builder.create();
    }
```

代码解释：

- AlertDialog.Builder builder = new AlertDialog.Builder(context)语句首先生成一个 AlertDialog.

Builder 的对象，这样就可以开始构造 AlertDialog.

- builder.setIcon(R.drawable.*alert_dialog_icon*)语句给 AlertDialog 预设置一个图片。这里预设的是 Alert_dialog_icon.png 这张图片。
- builder.setTitle(R.string.alert_dialog_two_buttons_title)语句给 AlertDialog 预设一个标题，标题的字符串内容在 value 目录的 string.xml 文件里定义。
- setPositiveButton（）这个方法设置确定按钮的一些属性。第一个参数为按钮上显示出阿里的内容。第二个参数为 DialogInterface.OnClickListener()监听器对象，这个监听器和前边学习的 Button 的单击监听器类似。
- onClick（）方法为监听器中的回调方法，即当单击 Dialog 的按钮时，系统回调这个方法。一般将对话框处理的逻辑写到回调函数里边。
- 预设好所有关于 Dialog 的属性后，执行 builder.create()后生成一个配置好的 Dialog.
- AlertDialog 与 AlertDialog.Builder

AlertDialog 是 Dialog 的一个直接子类，AlertDialog 也是 Android 系统当中的最常用的对话框之一。一个 AlertDialog 可以有两个 Button 或者 3 个 Button,可以对一个 AlertDialog 设置 title 和 message。不能直接通过 AlertDialog 的构造函数来生成一个 AlertDialog.一般生成 AlertDialog 的时候都是通过它的一个内部静态类 AlertDialog.Builder 来构造。

五、 附加实验

试着将代码中用到的字符串放到资源文件 string.xml 中，在代码中用 R.string.text_id 实现对字符串的引用。

六、 参考资料

docs/reference/android/app/AlertDialog.html
docs/guide/topics/resources/index.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。