大话企业级 Android 开发·第五部分

本教程说明及版权声明

- 《大话企业级 Android 开发》是国士工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国士工作室所有。
- 本教程是由国士工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国士工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国士工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方博客 http://www.cnblogs.com/guoshiandroid/留言或者直接与国士工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国士工作室官方博客定期更新，请访问国士工作室博客 http://www.cnblogs.com/guoshiandroid/获取更多更新内容。

关于国士工作室

我们(国士工作室)是一支专注于 Android 平台企业级应用开发的技术团队，对娱乐多媒体应用有着深刻的理解及研发能力，致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持，为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持，服务端可以采用 Java EE，也可以采用轻量级流行的 LAMP 技术体系。目前，研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件，并在持续升级中。

目前，我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作，发展迅速，渴望有志之士的加入，和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗，为移动互联网和智能手机时代贡献力量！

联系我们

电话:15711060468
Email:guoshiandroid@gmail.com
博客: http://www.cnblogs.com/guoshiandroid/

1 Android 程序设计的骨架--MVC

小安:博士,在您的指导下,通过学习了一个 HelloWorld 程序,了解了 Android 的基本运行情况，以及 Android 的工程结构。但是在学习的过程中也产生了一些疑问,比如为什么 Android 的文件结构要设置的这么复杂呢？都放在 src 文件夹下多好？何必要分的那么细？

大致：这应该是个初学者都会问到的问题。尽管前面已经给你详细的介绍了每个文件夹的作用，可是如果你没有实际开发经验，还是不能很好的体会它们存在的意义。下面就来告诉你这样设计的初衷，可能听完后你的吸收效果并不会很好，但是希望你以后学习 Android 还是以本章的思想为出发点，慢慢理解ＭＶＣ这个经典的设计模式!

MVC 设计模式简介：

MVC 本来是存在于 Desktop 程序中的,M 是指数据模型,V 是指用户界面,C 则是控制器。使用 MVC 的目的是将 M 和 V 的代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据你可以分别用柱状图、饼图来表示。C 存在的目的则是确保 M 和 V 的同步，一旦 M 改变，V 应该同步更新。

模型－视图－控制器（MVC）是 Xerox PARC 在八十年代为编程语言 Smalltalk－80 发明的一种软件设计模式，至今已被广泛使用。最近几年被推荐为 Sun 公司 J2EE 平台的设计模式，并且受到越来越多的使用 ColdFusion 和 PHP 的开发者的欢迎。模型－视图－控制器模式是一个有用的工具箱，它有很多好处，但也有一些缺点。

1.1MVC 如何工作

MVC 是一个设计模式，它强制性的将应用程序的输入、处理和输出分开。使用 MVC 应用程序被分成三个核心部件：模型、视图、控制器，它们各自处理自己的任务。

视图

视图是用户看到并与之交互的界面。对老式的 Web 应用程序来说，视图就是由 HTML 元素组成的界面，在新式的 Web 应用程序中，HTML 依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括 Adobe Flash 和像 XHTML、XML/XSL、WML 等一些标识语言和 Web services.

如何处理应用程序的界面变得越来越有挑战性。MVC 一个大的好处是它能为你的应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

模型

模型表示企业数据和业务规则。在 MVC 的三个部件中，模型拥有最多的处理任务。例如它可能用像 EJBs 和 ColdFusion Components 这样的构件对象来处理数据库。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的冗余，增加了代码的复用性。

控制器

控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击 Web 页面中的超链接和发送 HTML 表单时，控制器(例如:servlet)本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后确定用哪个视图来显示模型处理返回的数据。

现在我们总结 MVC 的处理过程，首先控制器接收用户的请求，并决定应该调用哪个模型来进行处理，然后模型用业务逻辑来处理用户的请求并返回数据，最后控制器用相应的视图格式化模型返回的数据，并通过表示层呈现给用户。

1.2为什么要使用 MVC

以前的大部分应用程序(非 Android 应用)都是用像 ASP、PHP，或者 CFML 这样的过程化(自 PHP5.0 版本后已全面支持面向对象模型)语言来创建的。它们将像数据库查询语句这样的数据层代码和像 HTML 这样的表示层代码混在一起。经验比较丰富的开发者会将数据从表示层分离开来，但这通常不是很容易做到的，它需要精心的计划和不断的尝试。MVC 从根本上强制性的将它们分开。尽管构造 MVC 应用程序需要一些额外的工作，但是它给我们带来的好处是无庸质疑的。

首先，最重要的一点是多个视图能共享一个模型。现在需要用越来越多的方式来访问你的应用程序，对此，其中一个解决之道就是使用 MVC，那么无论你的用户想要使用 XML 布局、直接使用代码编写界面或者通过 WebView 控件使用 HTML，用一个模型就能处理它们。由于已经将数据和业务规则从表示层分开，所以可以最大化的重用代码。

由于模型返回的数据没有进行格式化，所以同样的构件能被不同界面使用。模型也有状

态管理和数据持久性处理的功能。

因为模型是自包含的，并且与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。如果想把数据库从 MySQL 移植到 Oracle，或者改变基于 RDBMS 数据源到 LDAP，只需改变其模型即可。一旦正确的实现了模型，不管其数据来自数据库或是 LDAP 服务器，视图将会正确的显示它们。关于这点我们目前不用考虑，Android 中集成的 SQLite 数据库已经能很好的解决我们的需求。由于运用 MVC 的应用程序的三个部件是相互独立，改变其中一个不会影响其它两个，所以依据这种设计思想你能构造良好的松偶合的构件。

对我们来说，控制器也提供了一个好处，就是可以使用控制器来联接不同的模型和视图去完成用户的需求，这样控制器可以为构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择模型进行处理，然后选择视图将处理结果显示给用户。

MVC 的优点：

◆低耦合性。视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动 MVC 的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

◆高重用性和可适用性。随着技术的不断进步，现在需要用越来越多的方式来访问应用程序。MVC 模式允许你使用各种不同样式的视图来访问同一个服务器端的代码。它包括任何 WEB（HTTP）浏览器或者无线浏览器（WAP），比如，用户可以通过电脑也可通过手机来订购某样产品，虽然订购的方式不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。例如，很多数据可能用 HTML 来表示，但是也有可能用 WAP 来表示，而这些表示所需要的仅仅是改变视图层的实现方式，而控制层和模型层无需做任何改变。

◆较低的生命周期成本。MVC 使降低开发和维护用户接口的技术含量成为可能。

◆快速的部署。使用 MVC 模式使开发时间得到相当大的缩减，它使程序员（Java 开发人员）集中精力于业务逻辑，界面程序员（HTML 和 JSP 开发人员）集中精力于表现形式上。

◆可维护性。分离视图层和业务逻辑层也使得 WEB 应用更易于维护和修改。

◆有利于软件工程化管理。由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化管理程序代码。

MVC 的缺点：

MVC 的缺点是由于它没有明确的定义，所以完全理解 MVC 并不是很容易。使用 MVC 需要精心的计划，由于它的内部原理比较复杂，所以需要花费一些时间去思考。

你将不得不花费相当可观的时间去考虑如何将 MVC 运用到你的应用程序，同时由于模型和视图要严格的分离，这样也给调试应用程序到来了一定的困难。每个构件在使用之前都需要经过彻底的测试。一旦你的构件经过了测试，你就可以毫无顾忌的重用它们了。

根据开发者经验，由于开发者将一个应用程序分成了三个部件，所以使用 MVC 同时也意味着你将要管理比以前更多的文件，这一点是显而易见的。这样好像我们的工作量增加了，但是请记住这比起它所能带给我们的好处是不值一提的。

MVC 并不适合小型甚至中等规模的应用程序，花费大量时间将 MVC 应用到规模并不是很大的应用程序通常会得不偿失。

MVC 设计模式是一个很好创建软件的途径，它所提倡的一些原则，像内容和显示互相分离可能比较好理解。但是如果你要隔离模型、视图和控制器的构件，你可能需要重新思考你的应用程序，尤其是应用程序的构架方面。如果你肯接受 MVC，并且有能力应付它所带来的

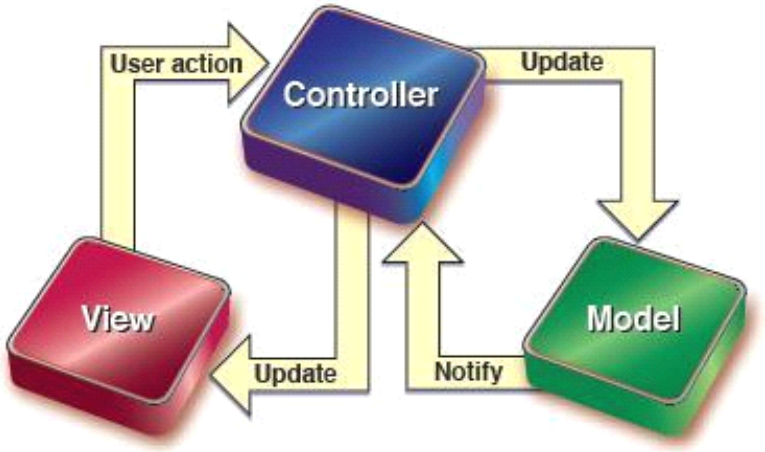
额外的工作和复杂性，MVC 将会使你的软件在健壮性、代码重用和结构方面上一个新的台阶。

大致：好了，这就是 MVC 的大概，怎么样，听的？
小安：妈呀，完全受不鸟了…这个 MVC 太复杂了。可是，讲了这么多到底他是怎样在 Android 中体现的呢？
大致：恩，问的好，下面我就结合前面的 MVC 的知识，解释一下它与 Android 的关系。

1.3Android 与 MVC

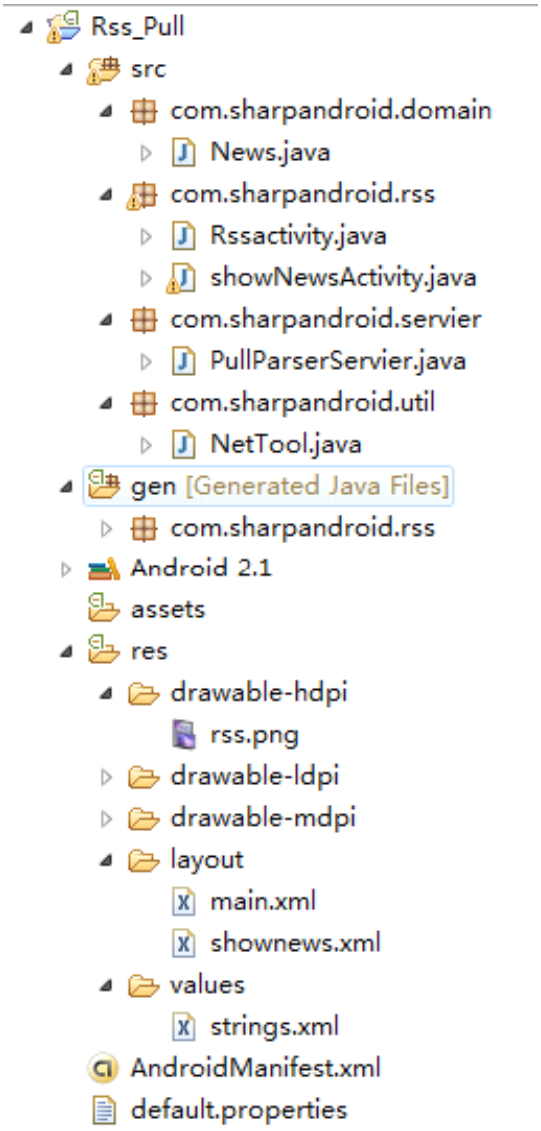
前面我们已经阐述了 MVC 的概念，以上知识，如果有 JavaEE 开发经验的话，可能理解起来会轻松一点。那么，Android 具体又是怎样与 MVC 框架结合呢？在思考这个问题之前，我们是不是真的已经理解什么是 MVC 模式呢？

很多学过 JavaEE 的开发人员，可能会说我已经很了解 Struts、Spring、Hibernate 等框架，可是在学习过程中是否想过这样一个问题：为什么要学习框架？框架到底给我带来了什么？同理，Android 程序也是如此，虽然 Android 平台没有 Struts 这样的框架，可是如果我们明白了 Struts 为 JavaEE 开发带来了什么样的改变，就好像采用 MVC 设计模式开发为 Android 开发带来了什么？



MVC 关系图

这里以我们后面将要学习的一个示例（RSS）为模板进行解释，
首先：观察软件的整个项目的结构，如下图：



该项目 MVC 分层：

M（Model）模型层

com.sharpandroid.domain:实体模型层，存放在程序中调用到的实体类。

com.sharpandroid.service:业务模型层，存放在程序中调用到的业务逻辑。

V（View）显示层

Android 很好的将显示层抽离，并放入 res/目录中以 XML 的形式体现。虽然对于控件属性修改可以通过代码完成，但还是推荐将控件的属性在 XML 设置为佳，遇到动态修改的内容再采用硬编码的方式。这样增加了程序的可读性，也有利于软件后期的维护。

main.xml、shownews.xml:布局文件

strings.xml：存放常量

drawable：存放使用的图片文件

C（Control）控制层

Control 是 Acticity 的天职，你只要告诉 Acticity 做什么，而至于怎么做，那是模型层的事。

试想一下，如果不用 MVC 模式，可能我们的代码会看起来相对的简洁，而且我们一样可以完成应用，甚至将尽量多的代码都涌入 Activity 文件，那样应用的层次会更加简洁。可是，结合 MVC 的设计模式思考一下，除了带来相对简洁的编程（注意，我们这里说的是相对），还会带来什么？

1. 这样做带来最明显的缺点就是过分的耦合。试想一下，在设计初期，没有遵循 MVC 进行严格的分层，而在开发中，当需要对一个方法或者一个布局进行更改时，由于层与层之间的过分耦合，那么你将面对的是“牵一发而动全身”的修改过程。如果基于 MVC，我们只要修改相应层，就达到了我们的目的。
2. 难以分工。在不使用 MVC 情况下，程序员要为如何设计 UI 头疼不已，浪费大量的精力，而不能将重点放在核心代码的编写上，降低开发效率。如果遵循 MVC，就可以很好的将视图层交给美工处理，程序员可以更好去关心核心代码的编写，不用再被繁琐的布局所困扰。虽然现在 Android 的布局并不是十分复杂，可是随着 Android 的发展，这必然是一个趋势。
3. 不宜维护性。在不使用 MVC 情况下，即使能顺利将其开发完成。在开发过程中用户可能对某一模块不满意，需要修改或者去除，有时需要添加新的模块，这样的事情，对于没有使用 MVC 设计模式的程序，将会是多么头疼的事情。
4. Android 系统专门提供了 res/values/目录下的诸如 strings.xml、colors.xml 类型的文件，可以将我们的常量值写入 xml 文件中，方便调用。这样不仅节省资源，还便于对资源的管理。如果某变量需要修改，可以直接对 strings.xml 文件进行修改。否则，我们需要对整个应用中所有用到该变量的代码进行修改。

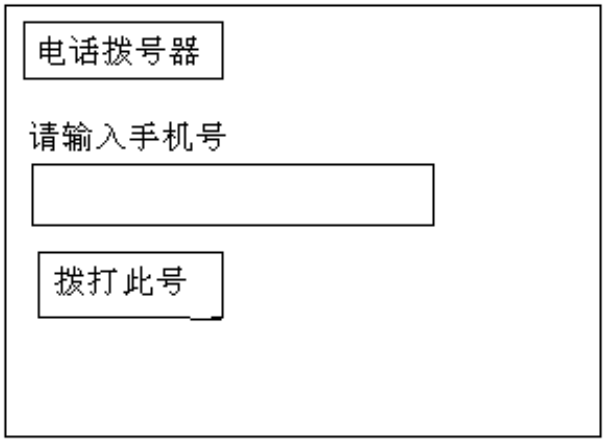
大致：怎么样，到现在是不是对 MVC 设计模式有一个大概的了解呢？
小安：恩，听你这么一讲，总算是知道 MVC 的重要性了，看来我还需要多做一些应用，不断的提高自己才行。

2 电话拨号器和短信发送器示例

大致：我们在前面已经完成了第一个示例 Android 项目，并且学习了项目的目录结构，对 Androi 开发的大体流程有了一个整体的印象和了解，接下来我们再做两个简单但是非常实用的小应用，这两个应用所需要的知识并不多，但可以熟悉 Android 的开发流程，体验 Android 开发的乐趣所在。开发自己的拨打电话及发送短信的应用。
小安： Android 手机上已经带有电话拨号器和消息发送器了，为什么我们还要学习自己开发呢？
大致：那我给你假设一种应用场景，你看看是否需要自己开发电话拨号及短信发送应用？假设我们已经开发了一个 CRM（客户关系管理）系统，在系统中存放了很多客户的联系方式。系统中可以查询客户的联系方式，那么要给客户拨号该如何做呢？我们只有将号码记录下来，再到系统自带的拨号器中拨打。但如果我们在 CRM 中加入拨号功能，点击用户号码就直接调用 Android 系统的拨号服务来进行拨号，岂不是更加高效？
小安：是啊，博士。这样太棒了，非常期待，让我们开始吧。

2.1 电话拨号器

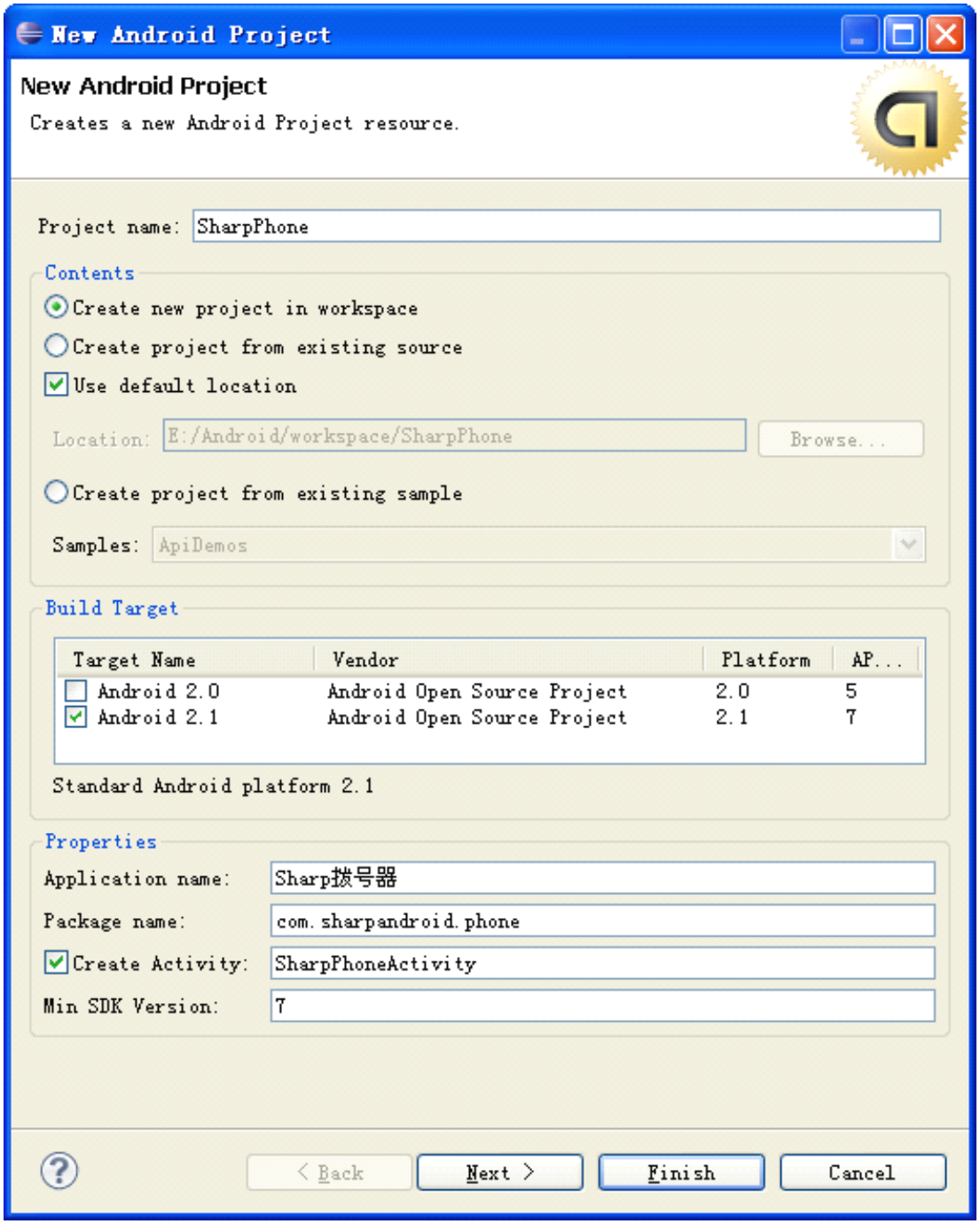
接下来我们实现一个简单的手机拨号器，其设计界面如下。



开发步骤如下：

1. 创建项目

项目名称为“phone”。在 Eclipse 中依次点击【File】→【New】→【Android Project】开始创建项目，创建时各项属性的填写如下图所示。



点击【Finish】创建完成。

Android 平台在设计的时候已经吸收了很多先进的设计理念，如 Web 开发的经典模式——MVC 模式，按照 MVC 模式，可以在设计应用时将界面、控制层、业务层分开。首先我们完成界面，Android 中的界面完全可以采用布局文件进行描述。由于默认情况下 ADT 创建的 Activity 类中所关联的布局文件就是自动生成的 main.xml，因此在这里我们可以直接修改 main.xml 文件来完成界面设计。

2. 编写 strings.xml 文件

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

```
<resources>
    <string name="hello">Hello World, SharpPhoneActivity!</string>
    <string name="app_name">Sharp拨号器</string>
    <string name="mobile">请输入手机号</string>
    <string name="button">拨打此号</string>
</resources>
```

此时可以自行查看 R.java 文件，在其 string 静态内部类中已多出两条常量。

3. 编写 main.xml 文件

编写 main.xml 界面布局文件。打开 main.xml 文件，编辑文件，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"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/mobile"
    />
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button"
    />
</LinearLayout>
```

说明：

EditText 视图组件相当于 HTML 中的 input 控件，用以接受输入。

Button 视图组件则是接受点击事件。

二者与 TextView 视图组件每行显示一个，共显示三行。

点击 Eclipse 中的【Layout】选项卡，预览界面。如下图。

可以看出，图中的显示与我们的预期有一定的差距。这是由于 ADT 仍然有些 Bug，只需留意即可。不过在模拟手机上运行时会出现正常的效果。

4. 处理按钮点击事件

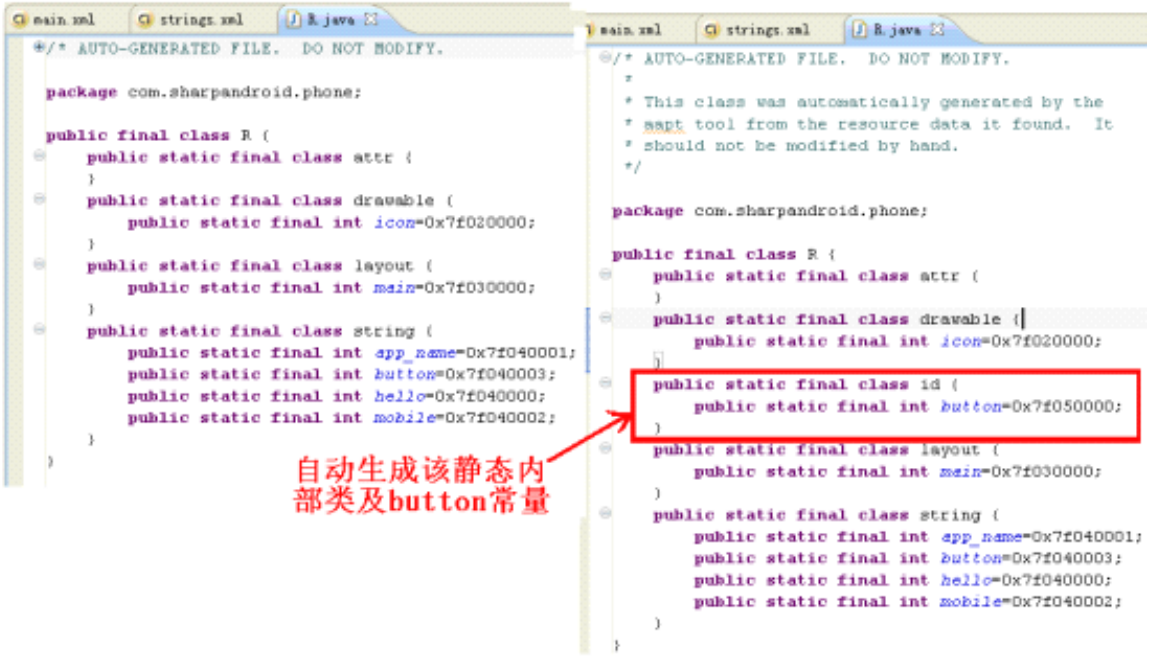
为了能够在点击“拨打此号”按钮之后能够真正的拨打电话，我们需要为该按钮添加一个点击事件监听器，具体方法如下。为了能够在代码中获取到该按钮，需要在 main.xml 文件中为该按钮添加一个唯一标识符，也就是指定它的 id 属性。

将 main.xml 文件中<Button>元素作如下修改，

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button"
    android:id="@+id/button"
    />
```

id 属性指定该视图组件的唯一标识符，供程序或其他视图组件引用。

"@+id/button"的含义是在R.java文件中的id这个静态内部类添加一条常量名为button。如果id这个静态内部类不存在，则会先生成它。在保存main.xml文件修改前后查看R.java文件对比如下图。



在拨打电话时需要获取在EditText中输入的电话号码，因此也应为EditText添加上id属性。添加id属性语法与上一致，只需要“/”后面更换变量名，修改后代码如下：

```
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/phoneno"
    />
```

接下来进入SharpPhoneActivity.java文件，前面已经说过，当SharpPhoneActivity对象实例化之后会调用onCreate()方法，完成初始化操作，因此我们可以将为按钮添加处理点击事件的初始化代码写入onCreate()方法中去。

SharpPhoneActivity.java具体代码如下:

```
package com.sharpandroid.phone;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

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

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                EditText phonenoText = (EditText)
findViewById(R.id.phoneno);
                String phoneno = phonenoText.getText().toString();//获取
手机号
                if((phoneno!=null )&& (!"".equals(phoneno.trim()))){//判
断手机号是否为空
                    //数据都是通过Uri进行包装的，创建一个拨号意图
                    Intent intent = new
Intent(Intent.ACTION_CALL,Uri.parse("tel:"+phoneno));
                    startActivity(intent);
                }
            }
        });
    }
}
```

说明：

- View android.app.Activity.findViewById(int id)
  该方法继承自其父类Activity，作用是根据指定的id寻找对应的View。在该程序中，R.id.button是R.java中id静态内部类中的button常量，其对应main.xml文件中定义的Button元素。Button是View的子类，因此该方法也可以返回Button，只是需要进行强制类型转换。
- void android.view.View.setOnClickListener(OnClickListener l)
  为Button注册一个回调方法，即当该Button被点击时会调用其参数 l，参数 l 中的onClick()方法在点击Button后会被调用。l 的类型是View类的内部接口，创建该类型的实例，可以创建一个实现了该接口的类，再实例化。当然简便的做法是采用匿名内部类的方法来实现，并且在Android中建议我们使用匿名内部类，这样有助于提高应用的性能。因此参数为

```
new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        }
},
```

当用户点击“拨打此号”按钮时会交给上面对象的onClick方法来处理。并且将被点击的组件传入，在本例中就是R.id.button这个按钮。

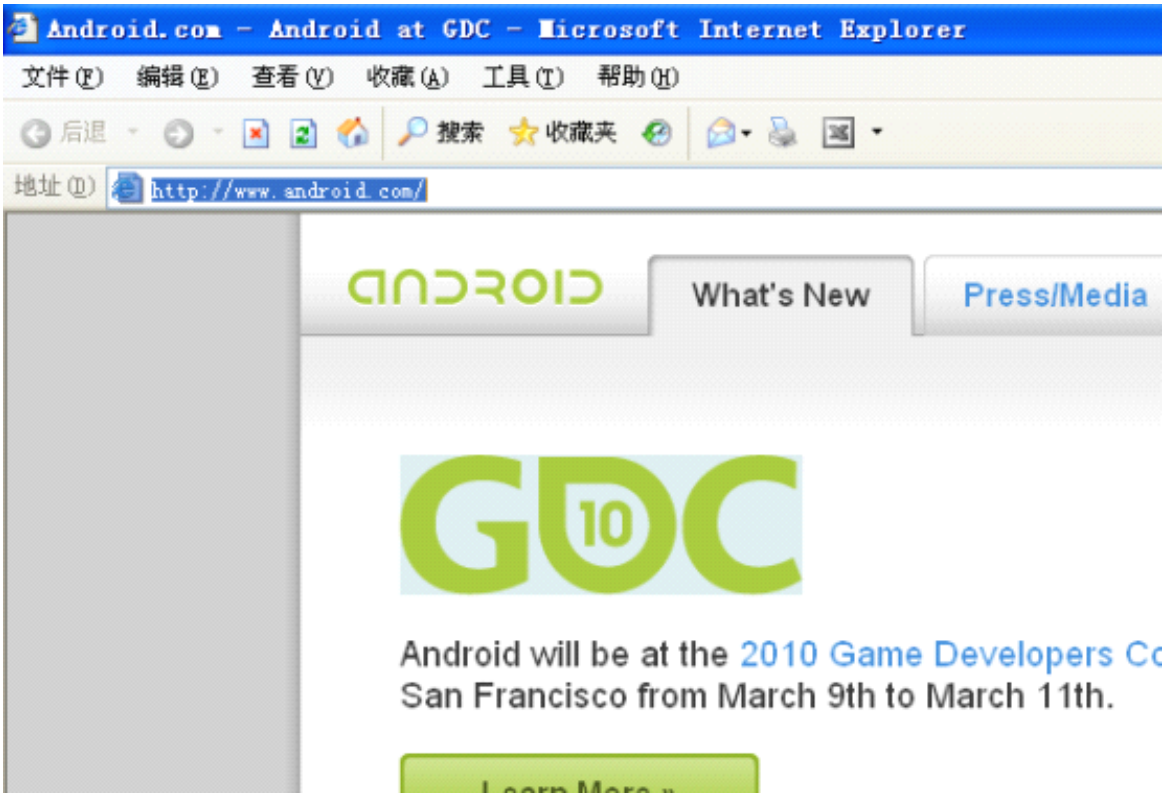
- EditText mobileEdit = (EditText)findViewById(R.id.mobile);
  获取文本框。建议在变量命名时加上Edit作为后缀，使得当我们看到该变量时能够知道其为一个EditText类型变量。
- String mobile = mobileText.getText().toString();
  得到该文本框中输入内容的String值。

- Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:"+phoneno));

我们所做的电话拨号并不需要自己实现拨打电话的功能，而是调用系统的拨号服务，由拨号服务来帮我们拨打电话。在介绍如何调用系统拨号服务之前，我们先来了解下在Windows系统下的一个与其类似的操作。

在windows系统下点击【开始】菜单，再点击【运行】，如图所示

在【打开】元素后面输入“http://www.android.com/”这个网址，点击确定，系统会自动的调用默认浏览器，并浏览该网址。如下图：



在Android操作系统中，也会有类似的功能，当我们把一个电话号码，加上“tel:”前缀传给系统之后，操作系统会自动调用拨号服务来拨打该号码。

创建一个拨打电话的意图，并将电话号码传给该意图。参数Intent.ACTION_CALL代表拨号动作名称，表示将会调用电话拨号器。参数Uri.parse("tel:"+phoneno)，将电话号码前加上“tel:”前缀之后，用Uri的parse方法将这个字符串包装成一个Uri对象作为参数传入，形成一个Uri地址，如同上面windows例子中“http://www.android.com/”告诉Windows操作系统调用浏览器后访问的网址一样，告诉Android系统调用电话拨号器之后要拨打的号码。切记“tel:”前缀不能写错。

- startActivity(intent);
  发送意图。这种意图只能激活Activity。该方法继承自其父类Activity，并不需要我们去实现，在此知道调用该方法可以产生的效果即可。

5. 申请拨号权限

由于我们拨打电话是使用手机系统中带的拨号器，因此要向 Android 平台申请使用手机拨号功能的权限。如前面所述，我们需要在功能清单文件 AndroidManifest.xml 中申请权限，详细如下：

AndroidManifest.xml

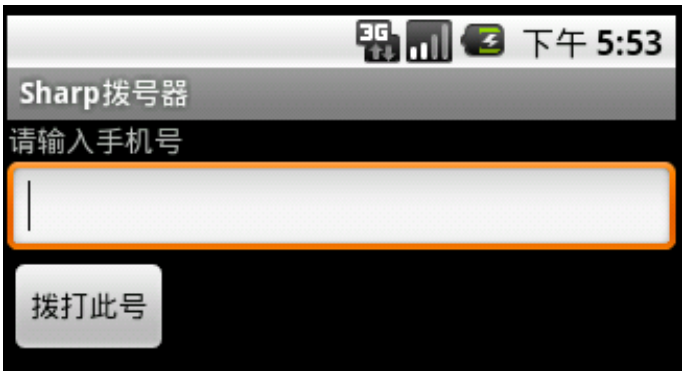
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.sharpandroid.phone"
      android:versionCode="1"
      android:versionName="1.0">
```

```
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".SharpPhoneActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    </manifest>
```

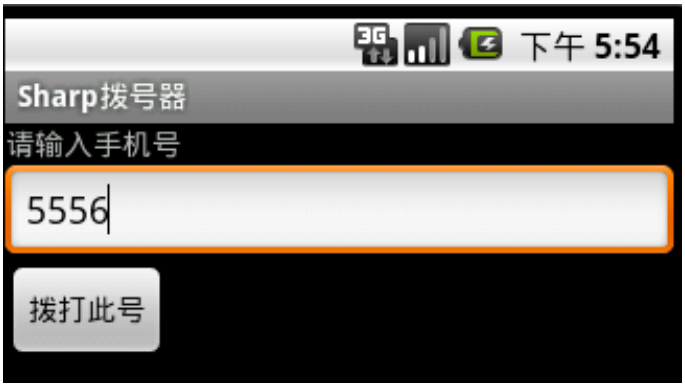
其中<uses-permission android:name="android.permission.CALL_PHONE"/>就是向系统申请手机拨号权限。

6. 运行程序

运行程序的方式将不再赘述，没有掌握的话可以翻阅前面的内容。将改程序发布到 sharp 模拟器上，运行效果如下：



为了能够测试出拨打的效果，因此启动另一个模拟手机——listen，如前所述，listen 的号码为其端口号“5556”，在文本框中输入“5556”，点击【拨打此号】按钮。



完成点击之后 5554：sharp 的界面如下，

被拨打的 AVD——5556：listen 中的界面如下图：

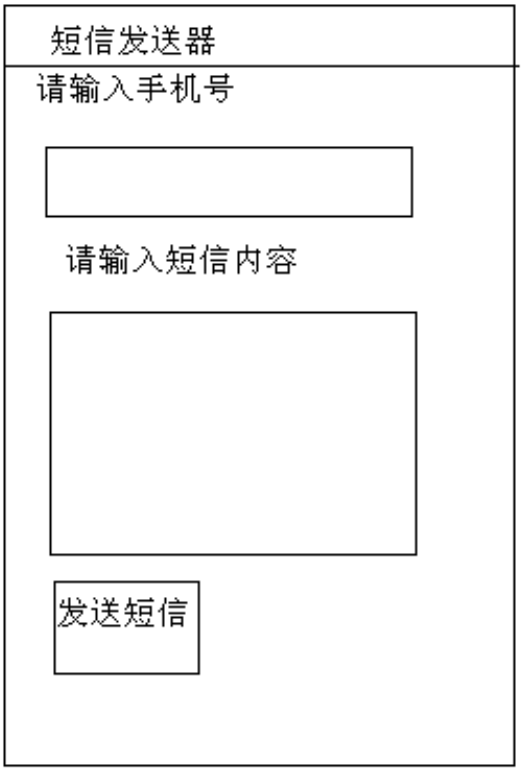
可以点击其  键或向左拖动屏幕中  图标拒接，也可以点击  键或者向右拖动品目中的  图标接听。

至此，电话拨号器开发完成并运行通过。

2.2短信发送器

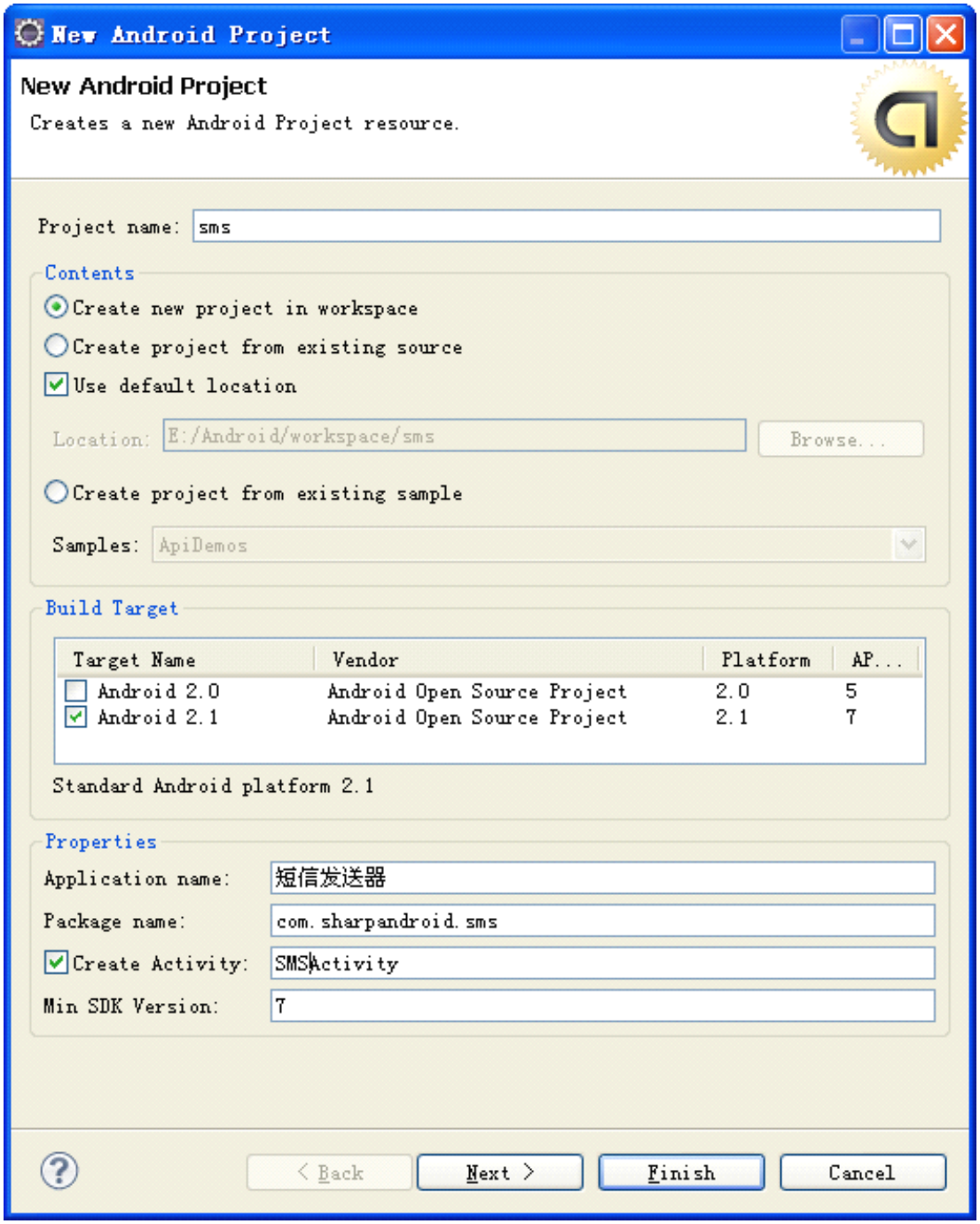
功能设计

前面我们实现了手机拨号器，下面我们实现另一个简单的应用——短信发送器。我们在第一个文本框中输入电话号码，第二个文本框中输入要发送的短信内容，之后点击“发送短信”按钮，就可以把输入的短信内容发给上面文本框中输入的这个号码了。其界面设计示意图如下：



1. 创建项目

创建项目时窗口内容如下图所示：



2. **编写 strings.xml：**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SMSActivity!</string>
    <string name="app_name">短信发送器</string>
    <string name="mobile">请输入手机号</string>
    <string name="content">请输入短信内容</string>
    <string name="button">发送短信</string>
</resources>
```

3. 编写 main.xml 文件：

```
<?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"
    >
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/mobile"
        />
     <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/mobile"
        />
     <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/content"
        />
     <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:minLines="3"
        android:id="@+id/content"
        />
      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button"
        android:id="@+id/button"
        />
</LinearLayout>
```

里面的一些视图组件和属性我们在前面已经介绍过了，唯一值得注意的是由于短信的内容可能较多，一行可能显示不下，因此我们可以通过设置 minLines 属性来设置视图的最小行数，我们这里设为“3”。运行之后可以比较两个文本框的高度来体验该属性的效果。

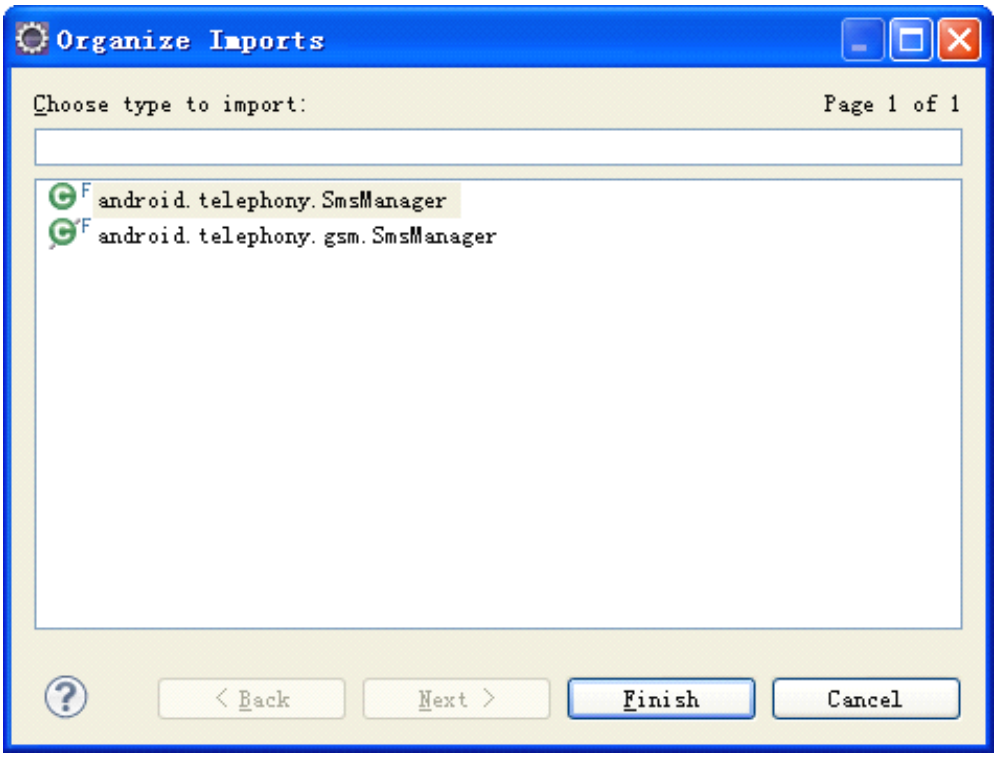
4. 编写 SMSActivity.java 类：

```
package com.sharpandroid.sms;

import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class SMSActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //获取按钮
        Button button = (Button)this.findViewById(R.id.button);
        //为按钮设置监听事件
        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                //获取手机号码文本框
                EditText mobileText =
                                (EditText)findViewById(R.id.mobile);
                //获取短信内容文本框
                EditText contentText =
                                (EditText)findViewById(R.id.content);
                //获取手机号
                String mobile = mobileText.getText().toString();
                //获取短信内容
                String content = contentText.getText().toString();
                //获取系统默认的短信管理器，此处导包时应注意
                //应导入android.telephony.SmsManager;
                //若导入 android.telephony.gsm.SmsManager包，
                //则SmsManger显示过时
                SmsManager smsManager = SmsManager.getDefault();
```

```
                //如果短信超过70个中文,将短信拆分成几条短信，以List形式逐条存放
                List<String> texts = smsManager.divideMessage(content);
                //迭代数组，逐条发送短信
                for(String text : texts){
                    //发送短信
                    smsManager.sendTextMessage(mobile, null, text, null,
null);
                }
                //添加一条发送结果提示
                Toast.makeText(SMSActivity.this, R.string.success,
Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

说明：

- `SmsManager smsManager = SmsManager.getDefault();`
  该行代码的作用是获取一个短信管理器SmsManger。使用该类需要导入系统jar包。导包时应注意导入的包是否正确，在Eclipse中按下【Ctrl+Shift+O】快捷键可以自动导入需要的jar包，但对该类导包时会出现如下选择界面。

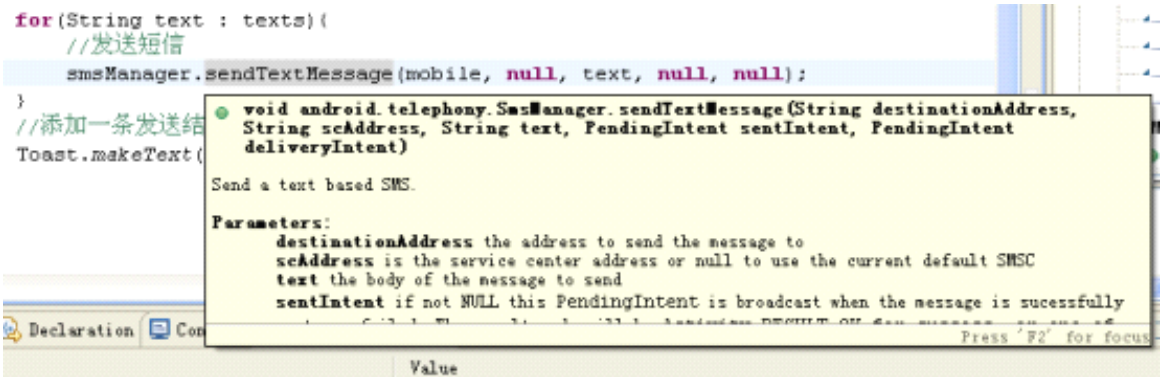


在此，我们选择“android.telephony.SmsManger”包。点击该行即可，之后点击【Finish】。导入成功。

若导入的是“android.telephony.gsm.SmsManger”包，则在代码中SmsManger类被Eclipse标记为已经过时，即类被一条横线划过，代码效果如下：

```
SmsManager smsManager = SmsManager.getDefault();
```

- `List<String> texts = smsManager.divideMessage(content);`
  SmsManger的divideMessage方法的作用是将传入的短信内容进行拆分。由于每条短信的字数会受到限制，因此我们在发送短信之前应将文本拆分，确保每条短信的字数不超过最大限制。拆分后的字符串放入List之中。
- `smsManager.sendTextMessage(mobile, null, text, null, null);`
  SmsManger的sendTextMessage方法的作用是发送短信。下面介绍其参数含义。
  在Eclipse中可以通过如下方式查看方法的参数、返回值的类型及意义等信息。将鼠标指向需要查看的方法，Eclipse就会自动的弹出如下的提示。



destinationAddress ：要发送的目标手机号码

scAddress ：短信中心的号码，可以置空

text：要发送的短信内容

sentIntent ：当手机将短信发出后有两种状态用户可能会比较关心，就是短信是否已经成功发出以及对方是否已经收到。当我们通过调用该方法发送短信时，是通过手机硬件来调用手机运营商的网络来完成发送功能，运营商会将发送的状态返回给手机，手机接到状态信号后会以意图的方式进行广播，如果程序中注册了该广播接收者就可以接受该意图。该参数就是用来接受短信是否发送成功的意图。如果不想获得，可以置空。

deliveryIntent： 与第四个参数类似，用来接收对方是否已经收到短信这个意图。如果不想获得，可以置空。

- `Toast.makeText(SMSActivity.this, R.string.success, Toast.LENGTH_LONG).show();`
  该行代码的作用是当消息发送之后，弹出一个提示信息，告诉用户已经发送过了。

makeText()方法是创建一个Toast通知，参数介绍：

context：当前应用的上下文，封装了当前应用的环境等等信息，而SMSActivity这个类是Context的子类。因此可以将SMSActivity这个对象传入，如果Toast在SMSActivity类中，可以直接使用“this”来代表一个Content对象，但该行代码位于匿名内部类中，要访问外部类的对象，可以采用SMSActivity.this的方式。

resId ：在该通知上要显示的内容，可以直接写入一个字符串，也可以通过R.string.xxx来引用strings.xml文件中定义的文字。本例就是引用string.xml中定义的success这条文字记录。

duration：设置显示的时间长短，可以输入数字0或1，也可以调用Toast类中已经定义的静态常量LENGTH_SHORT或者LENGTH_LONG。

Show()方法则是将该Toast显示。如果想让该Toast内容显示，切记要调用show()方法，好多初学者经常遇到明明已经创建了Toast但在界面上却显示不到的问题，这就是忘记调用show()方法。

5. 申请发送短信权限

发送短信需要向系统申请发送短信权限，功能清单文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.sharpandroid.sms"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".SMSActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
```

```
    <uses-sdk android:minSdkVersion="7" />
    <uses-permission android:name="android.permission.SEND_SMS"/>
</manifest>
```

- `<uses-permission android:name="android.permission.SEND_SMS"/>`

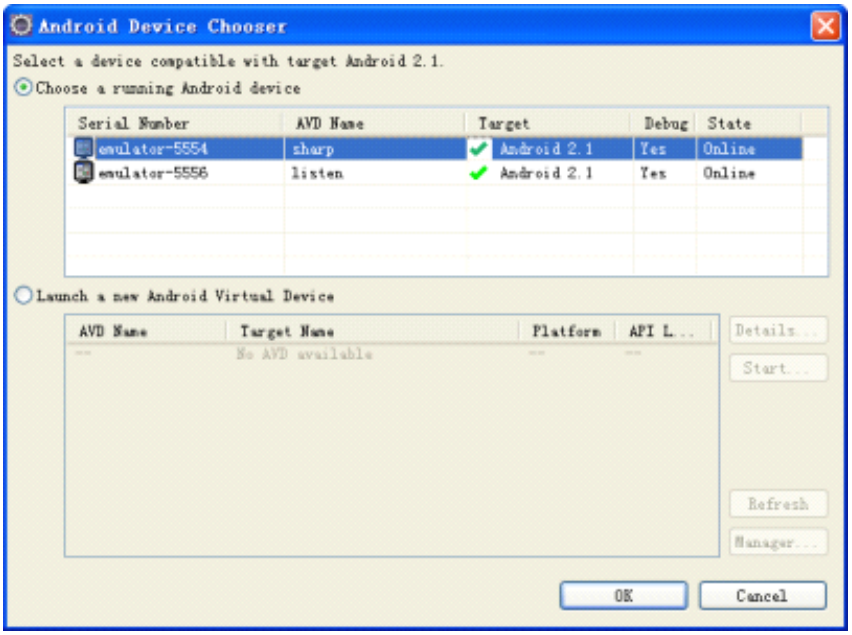
  该行代码就是申请短信发送权限

6. 运行程序

为了测试出发送短信的效果，如同电话拨号器的测试一样，我们仍需要再启动另一个AVD——listen。

运行程序。由于我们开启了两个 AVD，因此需要选择我们的目标 AVD，选择界面如下图。

我们双击 sharp 所在的列，或者选中之后点击【OK】，则开始将程序发送到 sharp 上面。

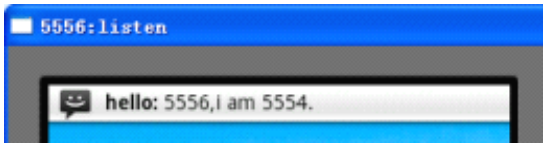


此时，sharp的界面如下图：

我们在界面上输入信息如下图所示：

点击【发送短信】按钮，sharp的屏幕下方出现Toast提示，如下图。

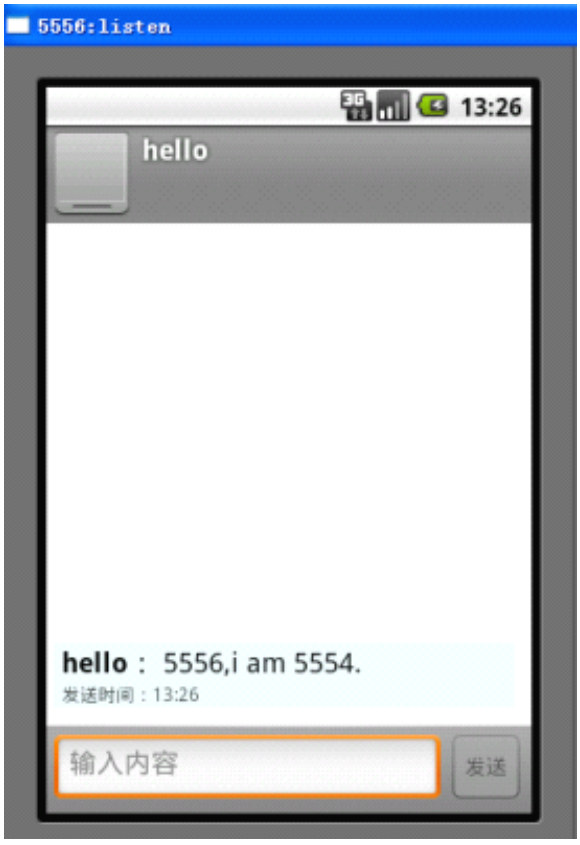
我们再来观察listen模拟器，在其屏幕上方的“状态栏”出现有未读短信的提示，如下图。



点击桌面上的 

图标，打开短信管理器，界面如下图。

点击未读短信，显示如下图。



至此，短信发送器开发完成。