Android API Demos 2.3 学习笔记

作者	snowdream
时间	2011 年 08 月 16 日

*谨以此书献给所有和我一样热爱**Android**的*
Coder!

前言

由于 Android 从诞生到现在并不是很久，与之有关的学习资料也不是很多。因此对于学习 Android 的人来说，Android SDK 附带的 API Demos 无疑是最好的学习资料。

本书作者试图通过自身学习实践，不断总结，记录笔记，来熟悉和掌握 Android 平台开发相关基础知识，并为后来者学习 Android 提供参考。

作　者

2011 年 8 月

目录

第 1 章　导言

1.1 搭建 Android 开发环境

本书主要介绍在 Ubuntu 11.04 + JDK 7 环境下，如何搭建 Android 开发环境。如果您需要在 Windows 下搭建 Android 开发环境，请参考网络相关内容。

1.1.1 搭建 JDK 开发环境

第一步：下载 JDK 7 压缩包

```
wget -c http://download.oracle.com/otn-pub/java/jdk/7/jdk-7-linux-i586.tar.gz
```

(注：如果下载不下来，建议使用迅雷下载，然后拷贝到 Linux 系统上。)

第二步：解压安装

```
sudo tar zxvf ./jdk-7-linux-i586.tar.gz  -C /usr/lib/jvm
cd /usr/lib/jvm
sudo mv jdk1.7.0/ java-7-sun
```

第三步：修改环境变量

```
vim ~/.bashrc
```

在该文件末尾添加以下内容：

```
export JAVA_HOME=/usr/lib/jvm/java-7-sun
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
```

保存退出，输入以下命令使之立即生效。

```
source  ~/.bashrc
```

第四步：配置默认 JDK 版本

由于 Ubuntu 中可能会有默认的 JDK，如 OpenJDK。为了使默认使用的是我们安装的 JDK 7，还要进行以下操作。

执行代码:

```
sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/java-7-sun/bin/java 300
sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/java-7-sun/bin/javac 300
```

执行代码

```
sudo update-alternatives --config java
```

系统会列出各种 JDK 版本，如下所示：

```
snowdream@snowdream:~$ sudo update-alternatives --config java
有 3 个候选项可用于替换 java (提供 /usr/bin/java)。
  选择       路径                            优先级  状态
------------------------------------------------------------
* 0            /usr/lib/jvm/java-6-openjdk/jre/bin/java   1061      自动模式
```

```
 1          /usr/lib/jvm/java-6-openjdk/jre/bin/java   1061      手动模式
 2          /usr/lib/jvm/java-6-sun/jre/bin/java       63        手动模式
 3          /usr/lib/jvm/java-7-sun/bin/java           300       手动模式

要维持当前值[*]请按回车键，或者键入选择的编号：3
update-alternatives: 使用 /usr/lib/jvm/java-7-sun/bin/java 来提供 /usr/bin/java (java)，于手动模式 中。
```

第四步：测试

在终端中输入 java -version，测试 JDK 环境是否安装成功。

```
snowdream@snowdream:~$ java -version
java version "1.7.0"
Java(TM) SE Runtime Environment (build 1.7.0-b147)
Java HotSpot(TM) Server VM (build 21.0-b17, mixed mode)
```

1.1.2 下载并安装 Eclipse

第一步：下载并安装 Eclipse （官方网站下载：http://www.eclipse.org/downloads/ ）

根据实际情况，推荐安装以下版本：

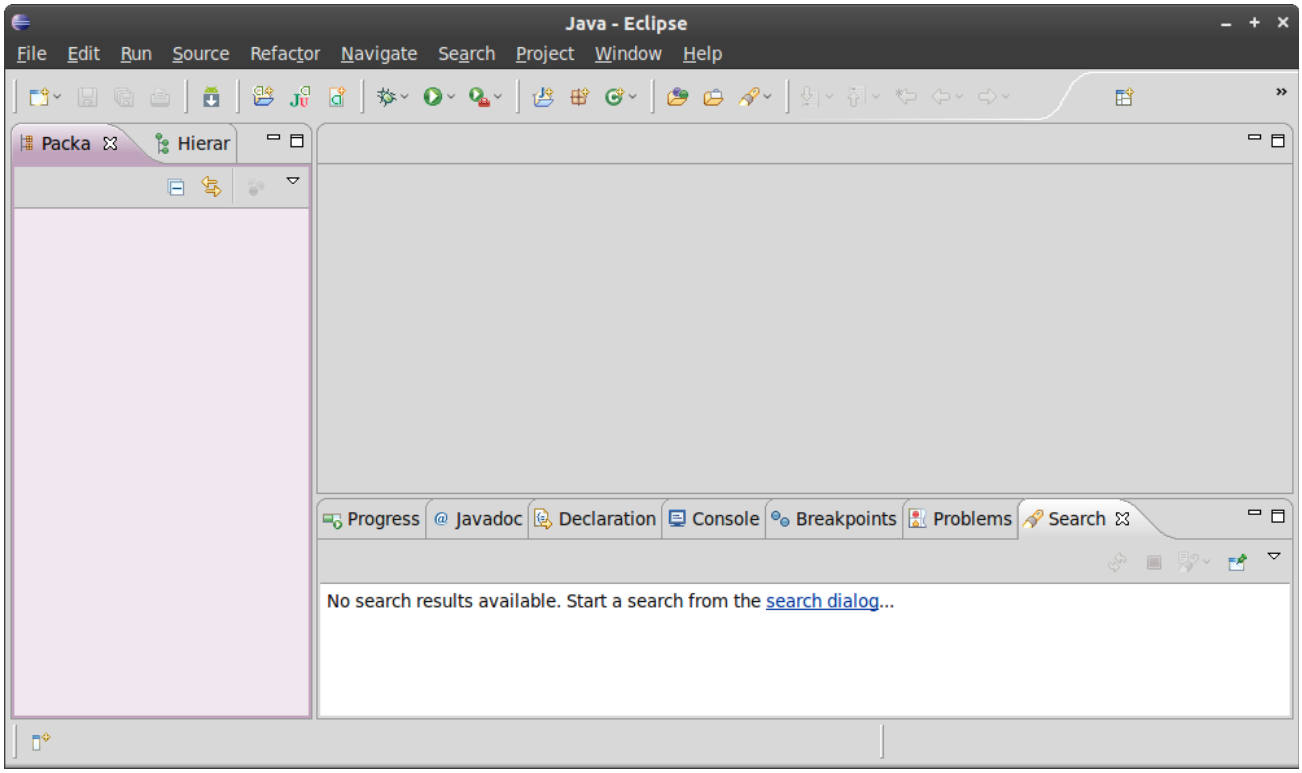
Eclipse IDE for Java and Report Developers, 250 MB

（Ubuntu11.04 32 位系统请直接通过以下命令下载并安装 Eclipse ）

```
wget -c
http://mirror.bjtu.edu.cn/eclipse/technology/epp/downloads/release/indigo/R/eclipse-reporting-indigo-linux-gtk.tar.gz
tar zxvf eclipse-reporting-indigo-linux-gtk.tar.gz
```

第二步：测试

进入 Eclispe 安装目录，双击 Eclipse 可执行程序，如果依次出现以下画面，则 Eclipse



安装成功。

1.1.3 下载 Android SDK 以及搭建 Android 开发环境

第一步：下载并安装 Android SDK

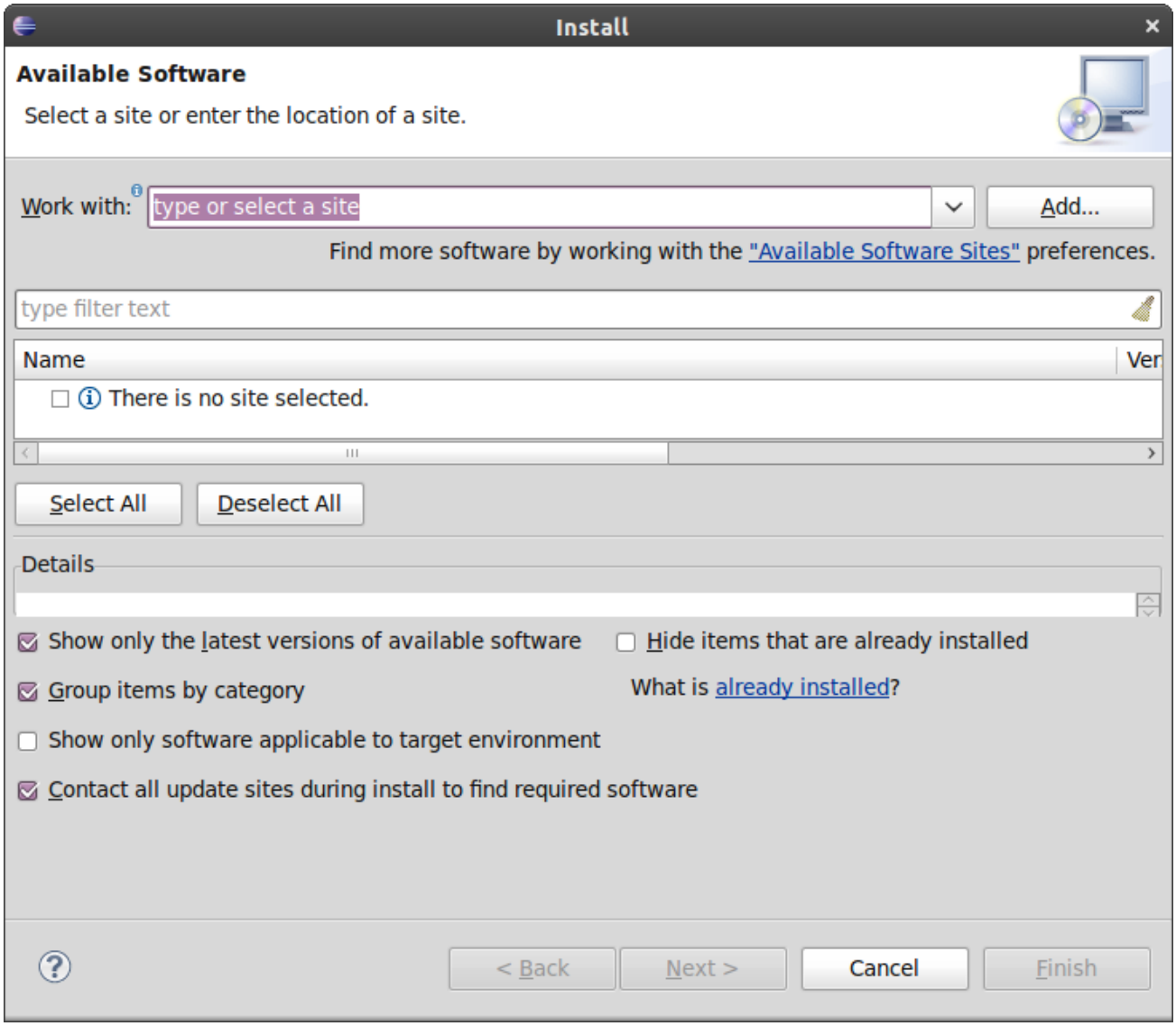
```
wget -c http://dl.google.com/android/android-sdk_r12-linux_x86.tgz

tar zxvf android-sdk_r12-linux_x86.tgz
```

第二步：在线安装 Eclipse 插件 ADT

启动 Eclipse，然后依次选择菜单：Help > Install New Software....

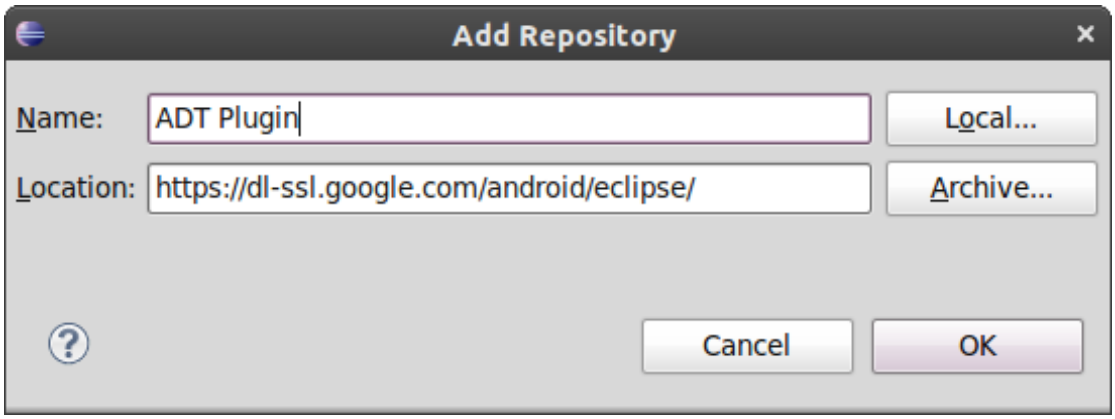
在窗口右上角点击 Add 按钮



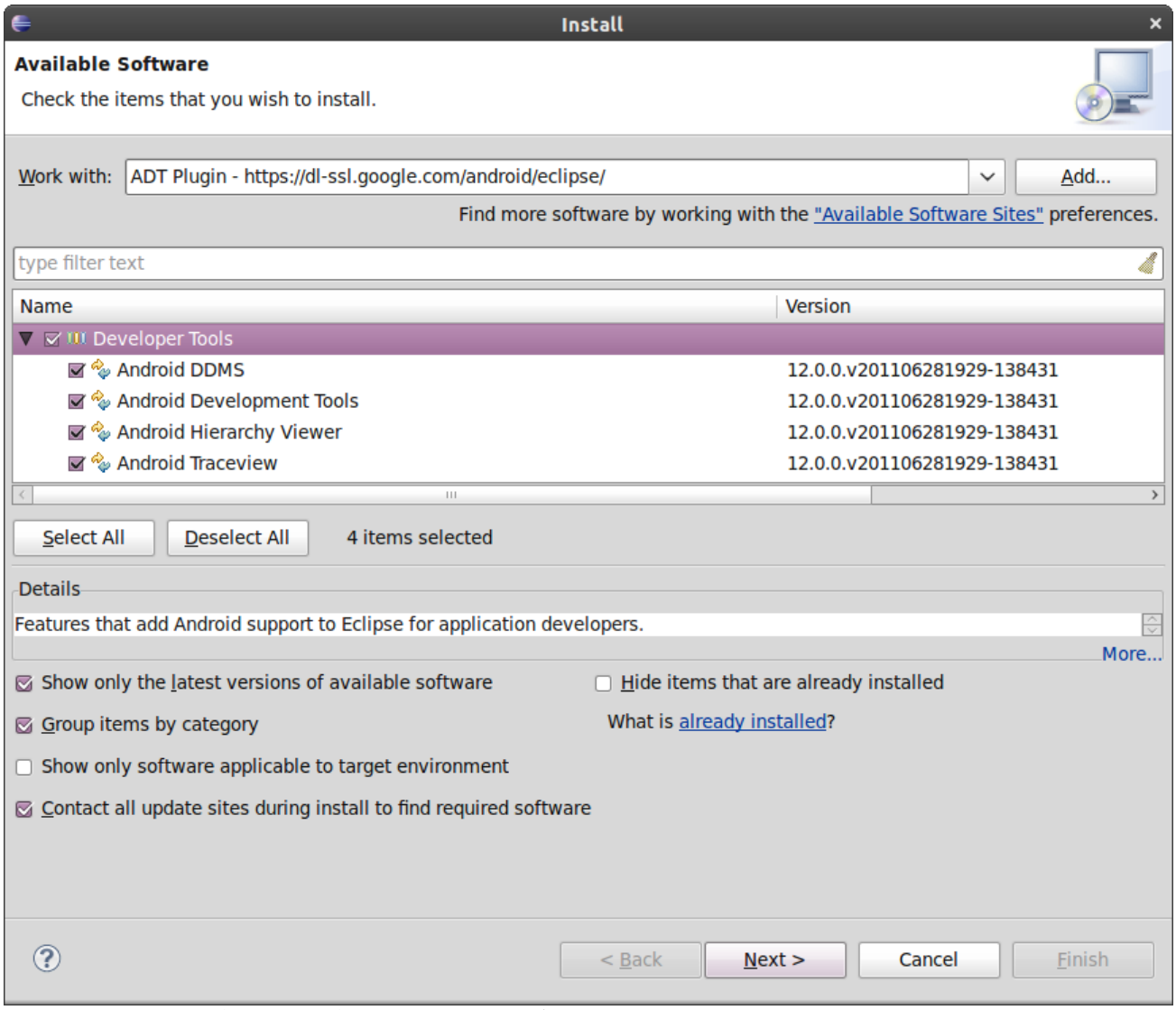
在添加源地址窗口中，在 Name 字段后面填写"ADT Plugin"，而在下面的 Location 字段后面填写以下地址：

```
https://dl-ssl.google.com/android/eclipse/
```

然后点击 OK。



在可选软件窗口，点击 Select All 全部安装，然后点击 Next。



在下一个窗口，你会看到一系列即将被下载的工具，点击 Next。

阅读并且接受软件协议，然后点击 Finish。

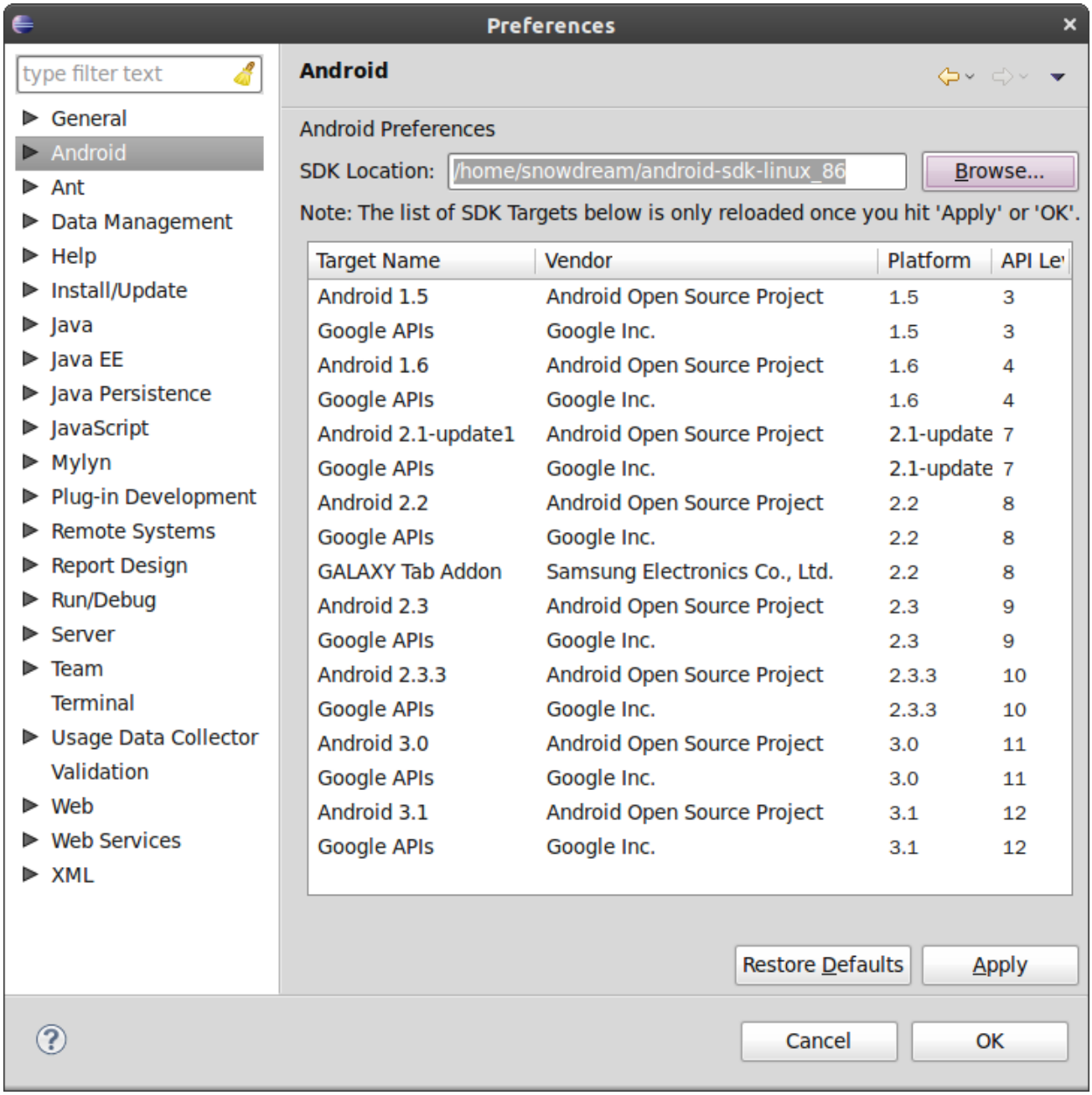
第三步：配置 Eclipse 插件 ADT

启动 Eclipse，然后依次选择菜单：Window > Preferences...

在左边的面板上选择 Android 选项，如下所示：

点击 Browse... 并且定位到你的 Android SDK 目录,例如 /home/snowdream/android-sdk-linux_86

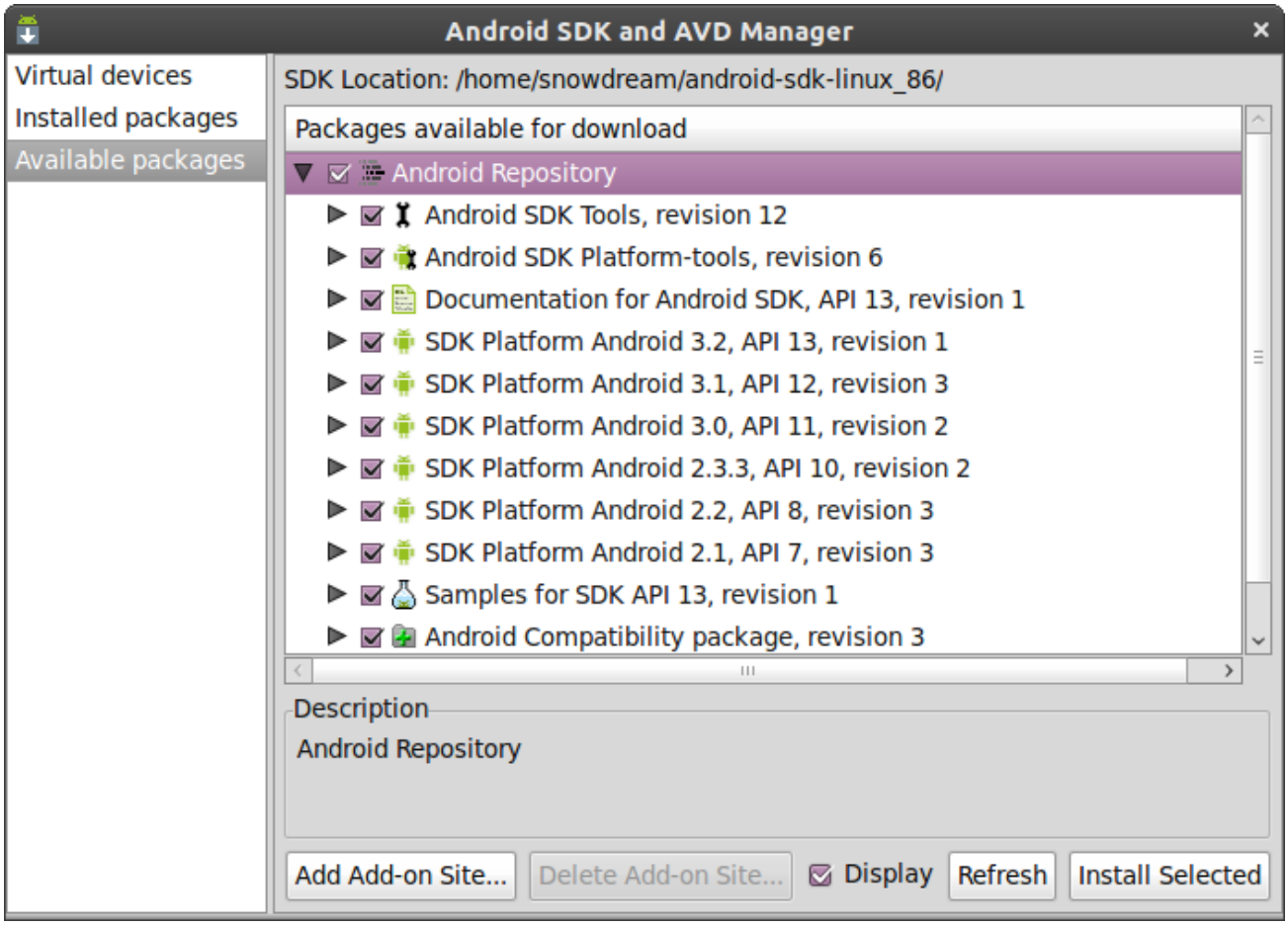
先点击 Apply, 然后点击 OK。



第四步：添加 Android SDK 组件

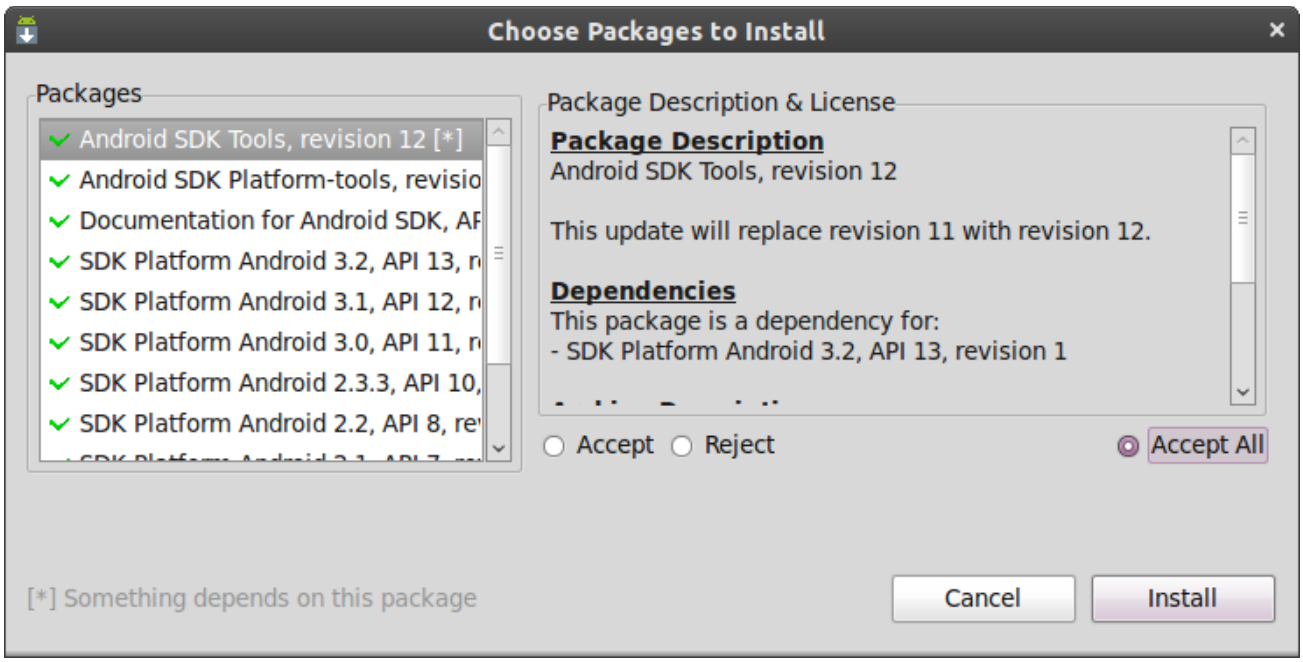
启动 Eclipse，然后依次选择菜单：Window > Android SDK and AVD Manager

在左侧面板上选择 Available Packages，这将会在右侧显示 SDK 源中所有可以进行下载安装的组件。



根据需求，选择你所需要安装的组件，然后点击 Install Selected。在接下来弹出的阅读协议窗口中，选择 Accept All，然后点击 Install。这些组件将会安装到您的 Android SDK 安装目录。

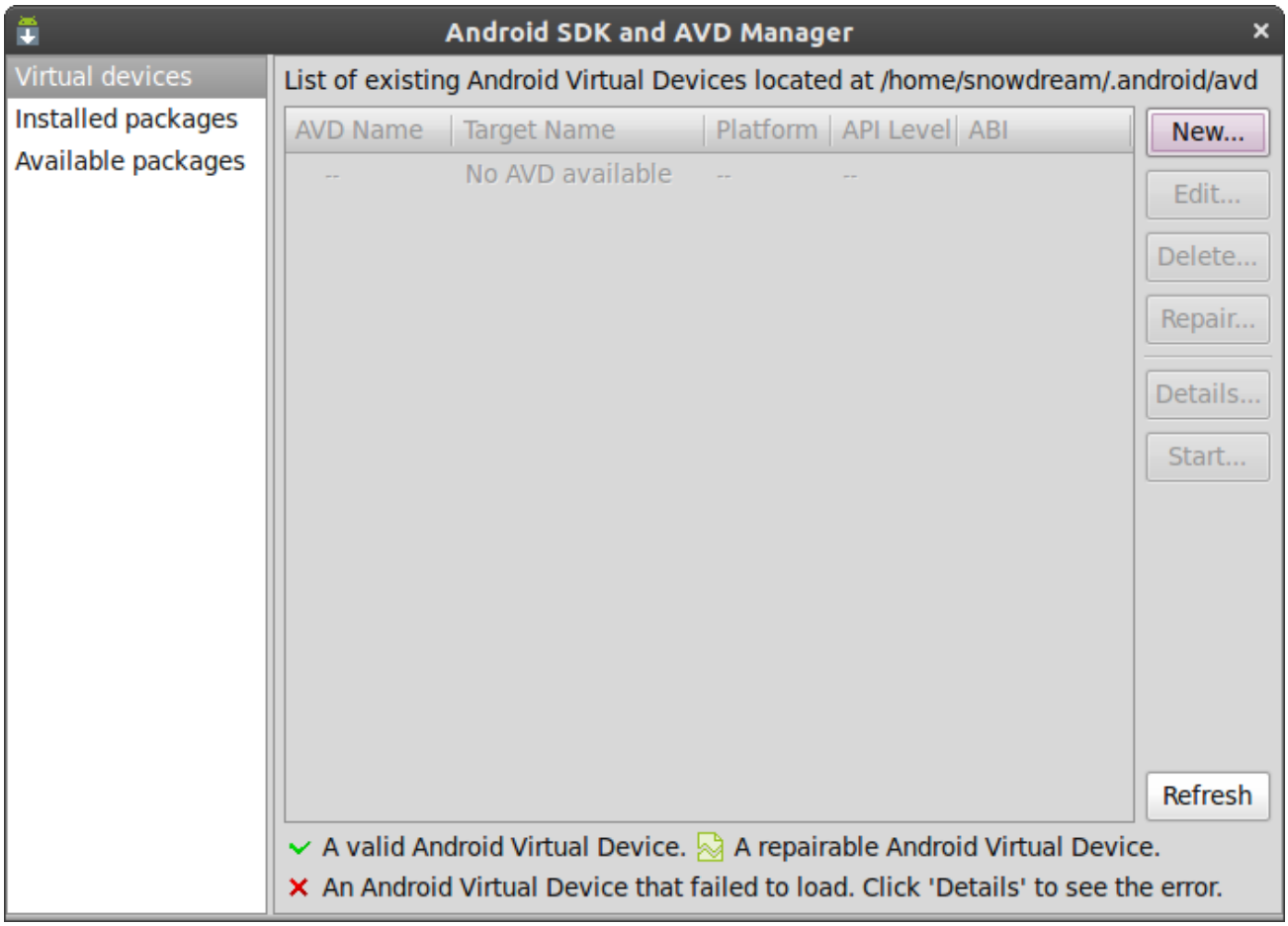
注：这一步可能需要花费数小时，具体时间和您的网络环境密切相关，请耐心等候。



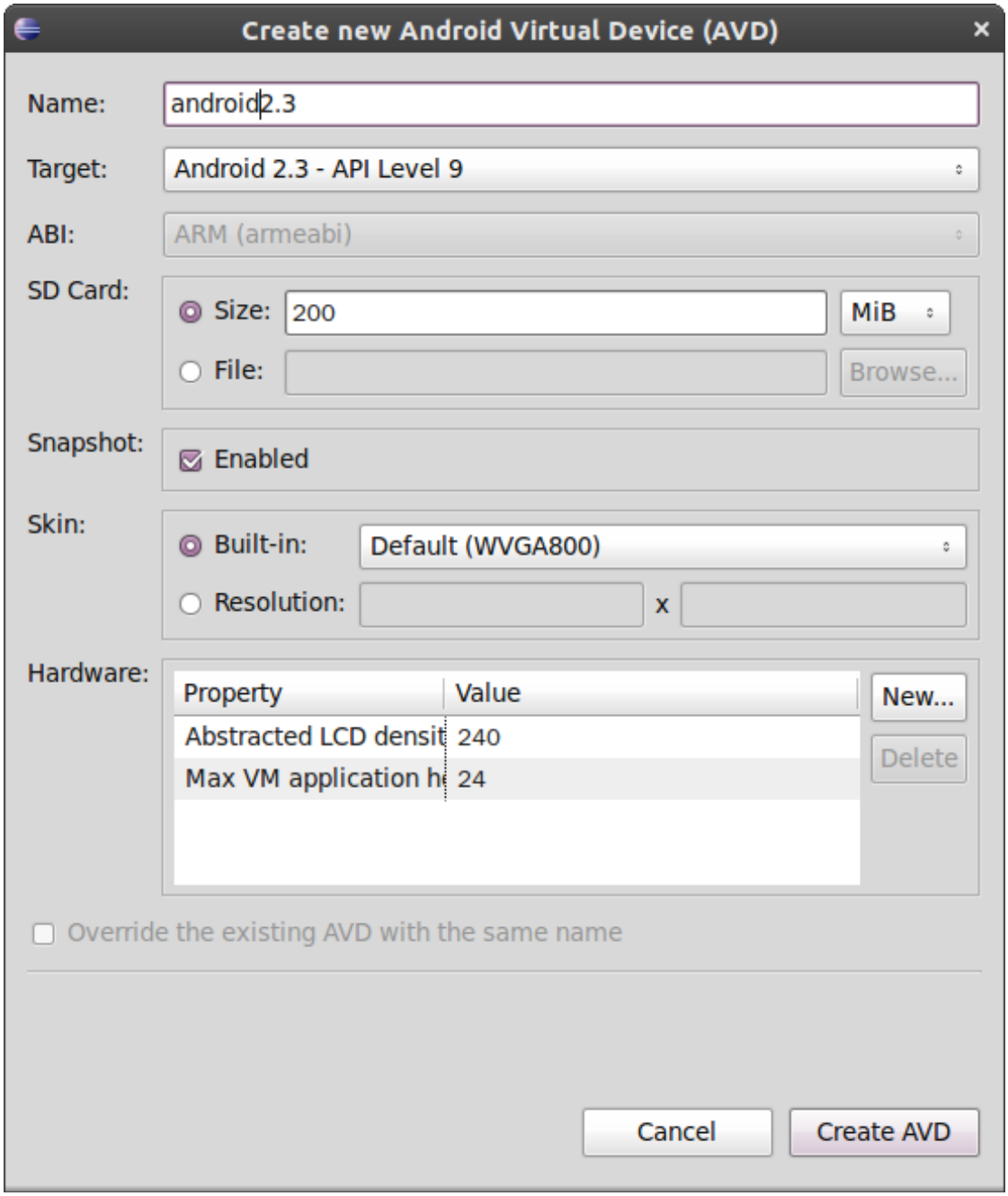
安装完成后，根据提示，需要重新启动 Eclipse 才能应用更新。

1.1.4 创建 Android 虚拟设备 AVD

启动 Eclipse，然后依次选择菜单：Window > Android SDK and AVD Manager



在左侧面板上选择 Virtual Devices，然后在右上角点击 New... 新建 AVD 设备，如下所示：



注明：

Name：填写 AVD 名称，例如 android2.3

Target：根据常用的 SDK 版本进行选择，例如，Android 2.3-API Level 9

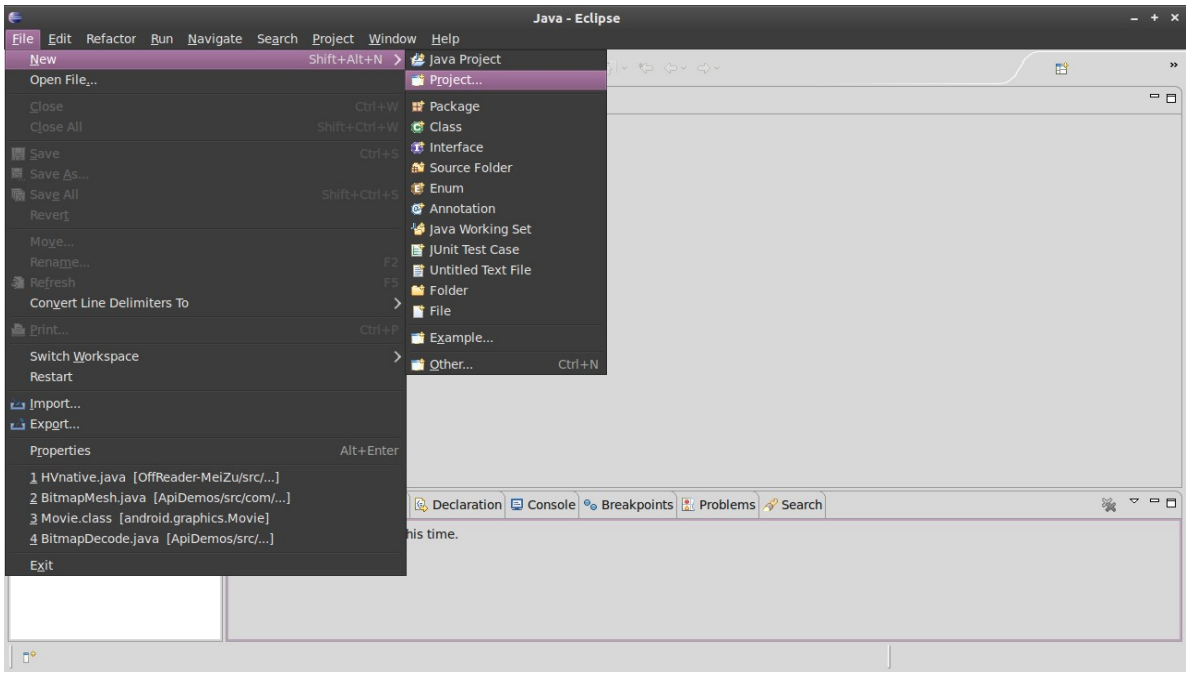
Size：虚拟 sd 卡容量大小，根据实际需求设置，例如 200MiB

Built-in：选择 AVD 的皮肤，这里保持默认选项

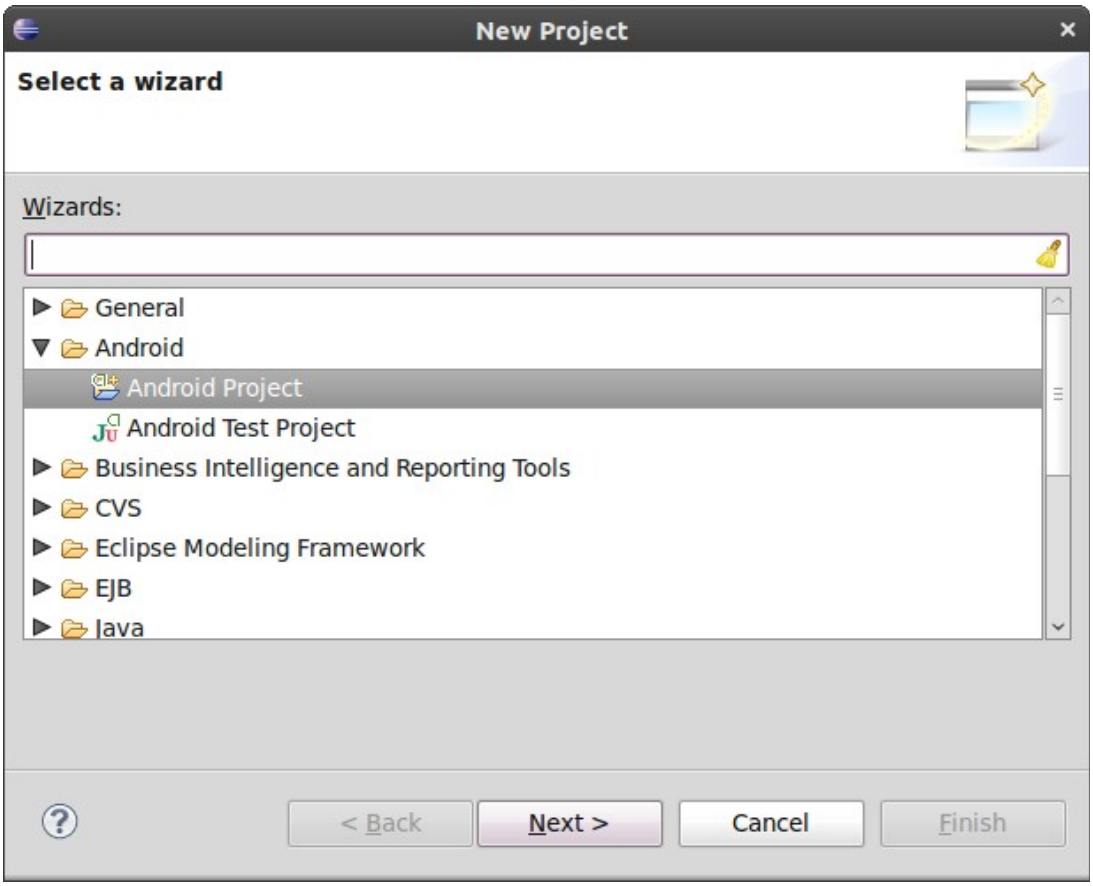
1.2 创建第一个 **Android** 项目（**Hello World!**）

第一步：根据新建项目向导创建项目

启动 Eclipse, 选择"File"--"New"--"Project",打开新建项目向导。

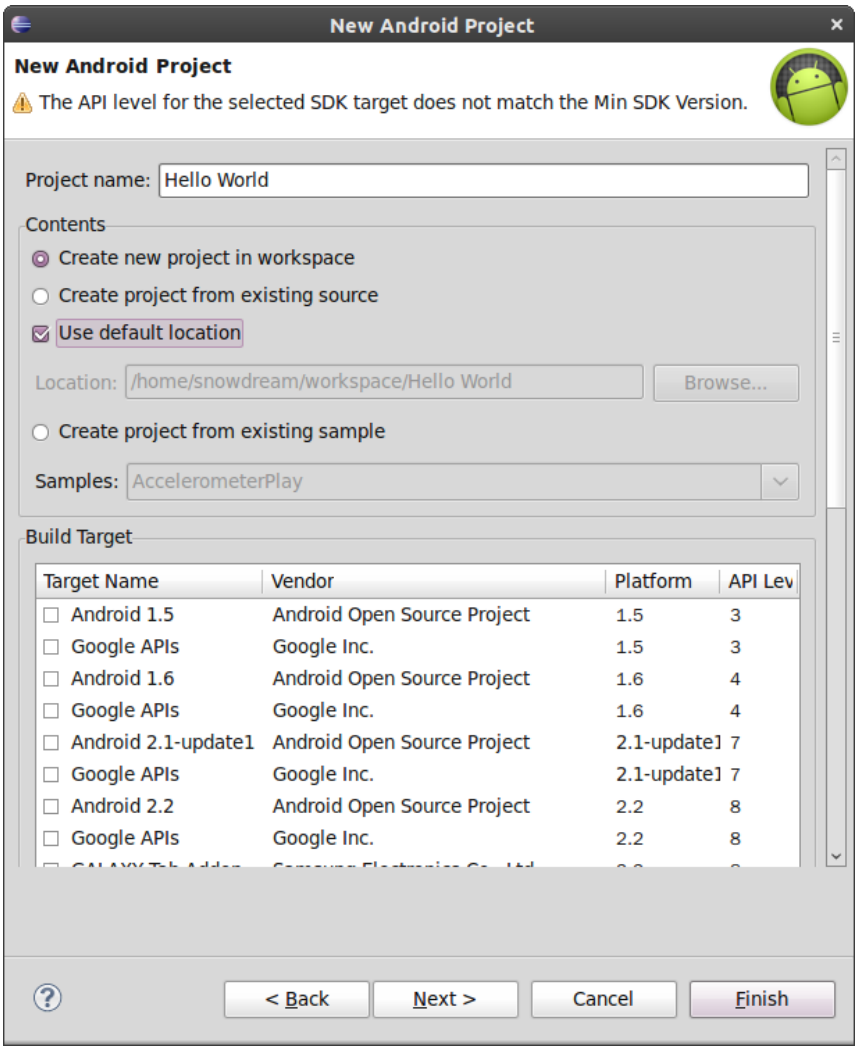


展开"Android"项，选择"Android Project"，单击"Next"按钮继续创建。



在"Project name:"字段后填写项目名称"Hello World"。

注：默认在 Eclipse 工作目录下以项目名称创建一个新文件夹作为该项目的主文件夹，如果您需要自定义项目主文件夹，需要先点击掉"Use default location"选项，然后在下面的"location"字段后面填写自定义路径。



把右边的滚动条往下拉，在"Build Target"下面选择您编译需要使用的 SDK 版本，这里我们选择版本"Android 2.3"。其他字段填写说明如下：

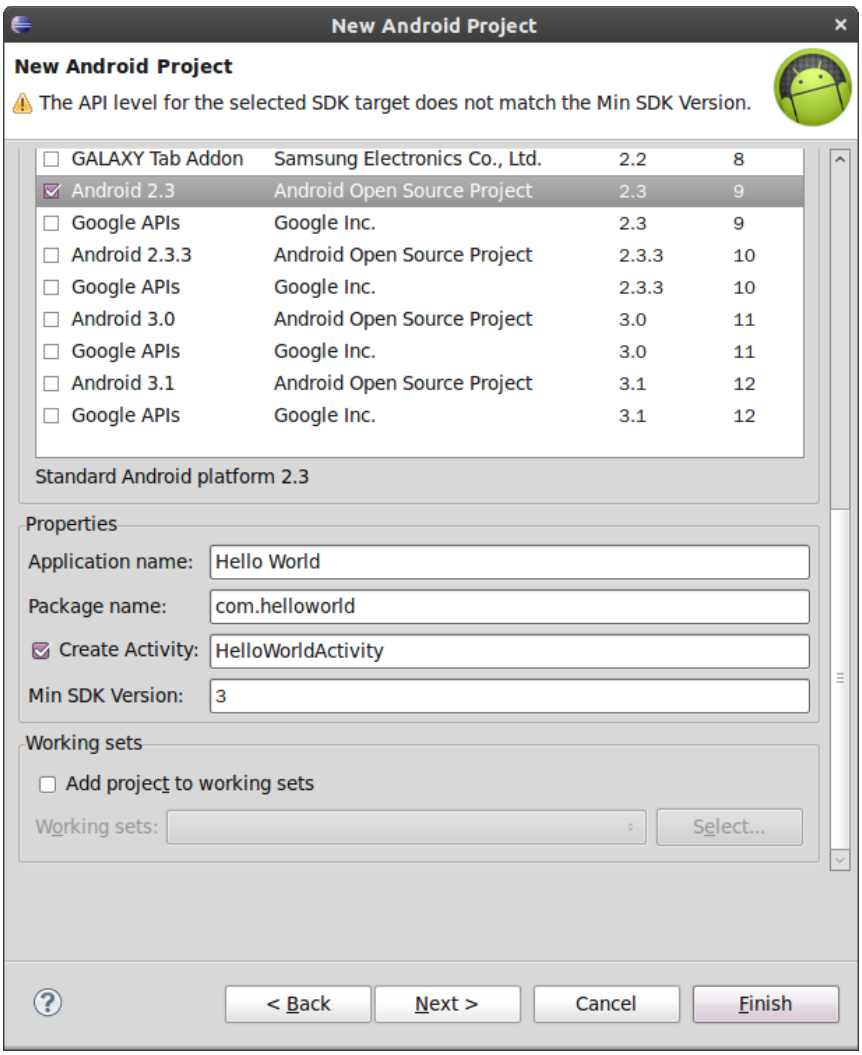
Application name： Hello World　　//程序名称

Package name: com.helloworld　　//软件包名称

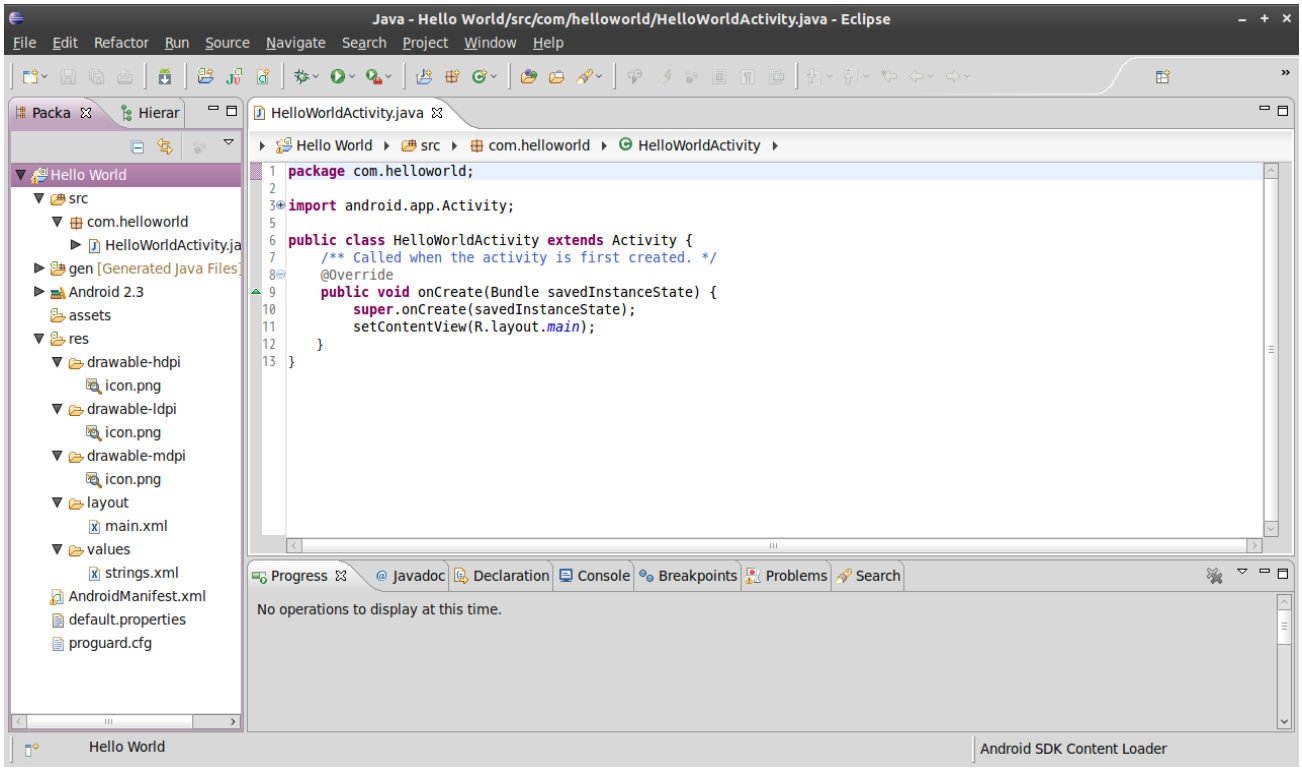
Create Activity: HelloWorldActivity　//Android 项目主 Activity 名称

Min SDK Version: 3　　//向下兼容的最低 Android 版本，对应”Build Target“下面的”API Level“

如下图所示：

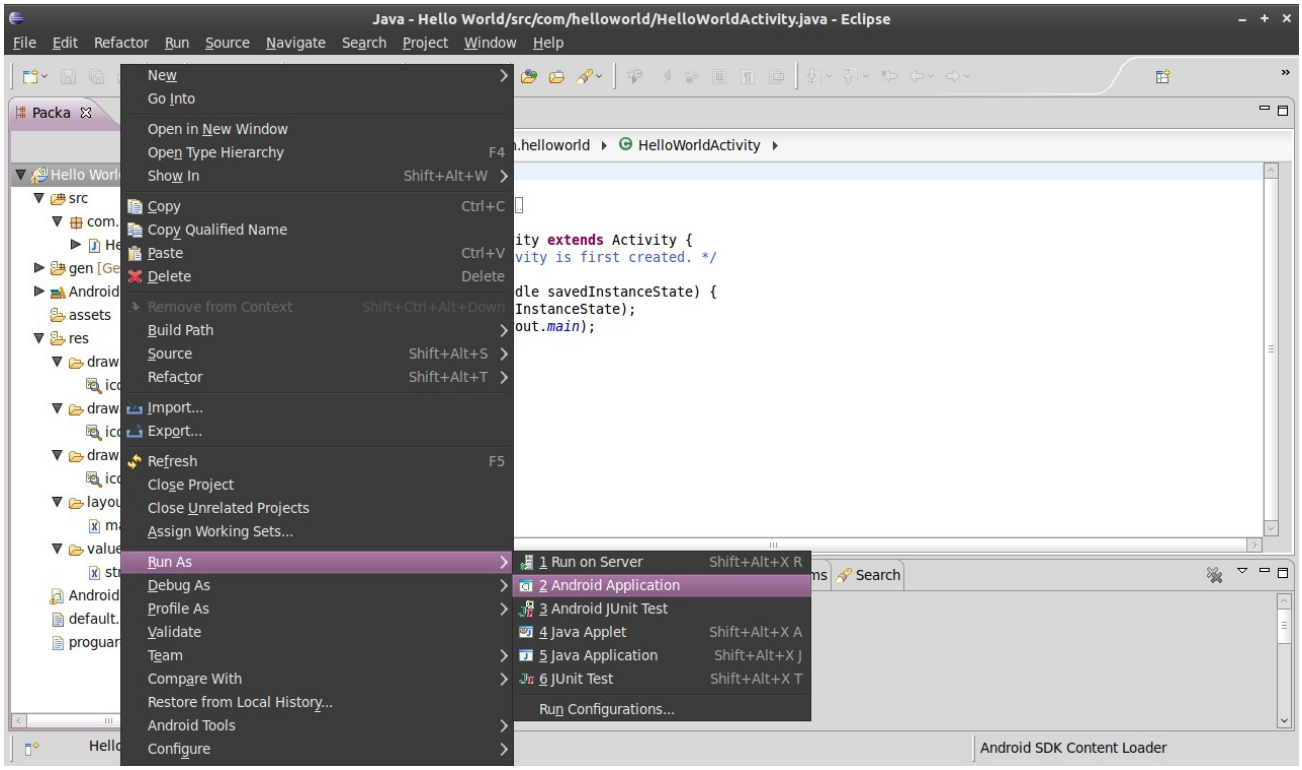


最后单击"Finish"按钮，项目创建完成。



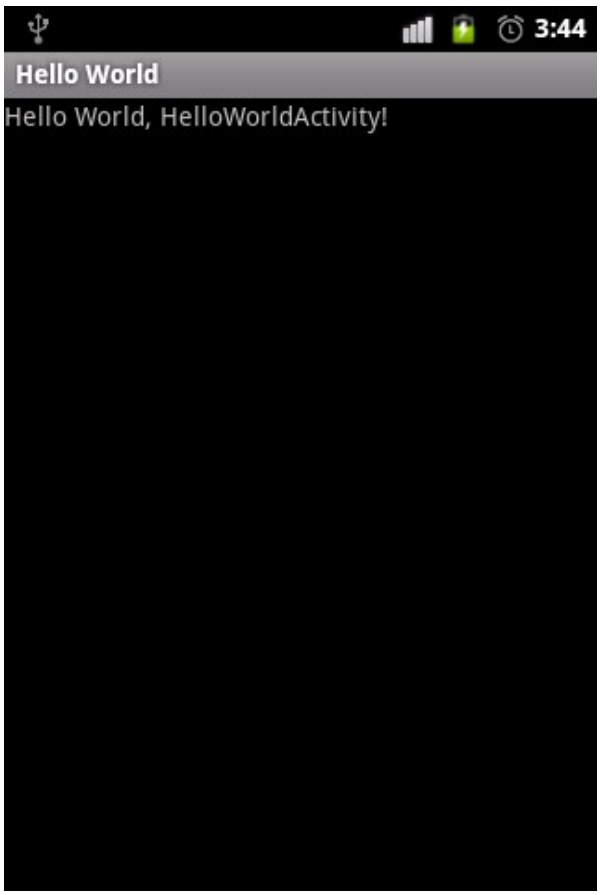
第二步：运行 Android 项目程序

在 Eclipse 左侧"Package Explorer"窗口，右键点击刚刚创建好的"Hello World"项目文件夹，在功能菜单上选择"Run As--Android Application"功能。如下图所示：



如果已经创建 AVD 虚拟设备，则会自动启动模拟器。否则，请参考"1.1.4　创建 Android 虚拟设备 AVD"章节先创建一个 AVD 虚拟设备。如果您拥有 Android 手机，也可以不用创建该设备，直接使用手机运行调试 Android 程序。

运行效果：



1.3 Android 应用程序架构

Android 应用程序以可以分为下三种类型：

1、前端 Activity（Foreground Activities）；

通俗一点讲 Activity 可以理解为一个界面容器，里面装着各种各样的 UI 组件。例如，上面例子中“Hello World” 显示界面。

2、后台服务（Background Services）；

系统服务（System Service）、系统 Broadcast（广播信息）与 Receiver（广播信息）接收器）等都属于后台服务。它们在后台运行时，并不会对于前端 Activity 的显示造成影响。例如，音乐播放放到后台时，并不影响其他界面操作响应。

3、间隔执行 Activity（Intermittent Activities）；

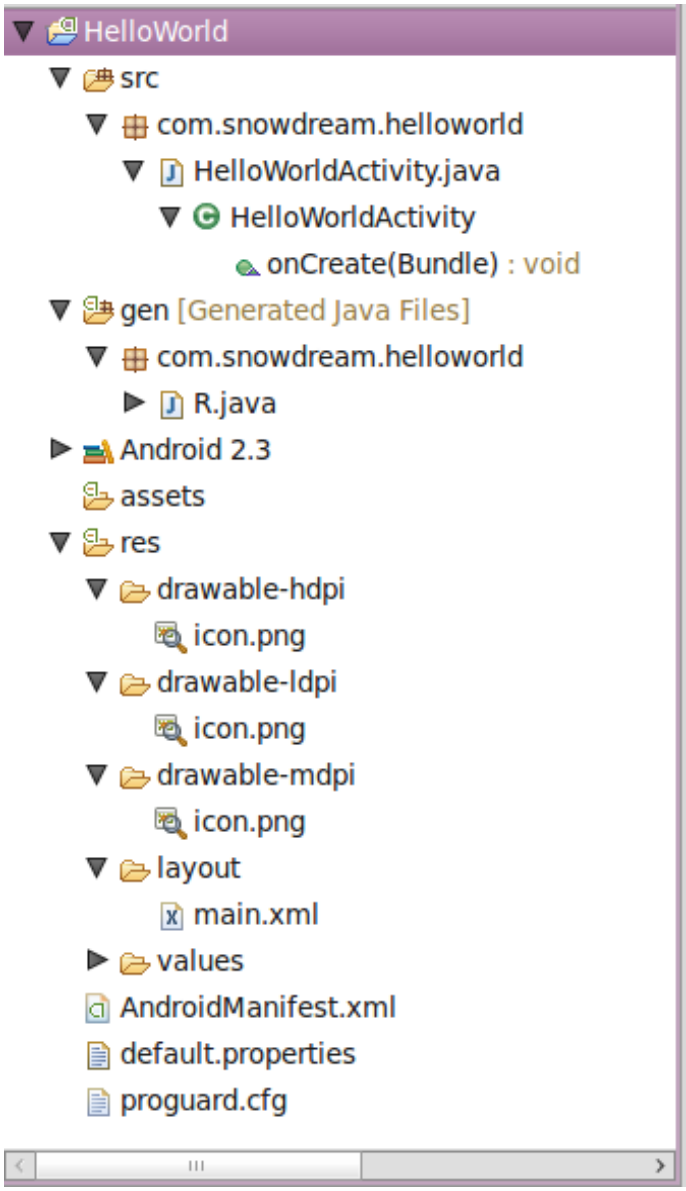
包括进程（Threading）、Notification Manager 等都属于这一类。

这里我们以 Hello World 这个简单的应用程序为例，简述一下 Android 应用程序的架构。如右图所示：

src/ java 源代码存放目录

gen/ 自动生成目录

gen 目录中存放所有由 Android 开发工具自动生成的文件。目录中最重要的就是 R.java 文件。 这个文件由 Android 开发工具自动产生的。Android 开发工具会自动根据你放入 res 目录的 xml 界面文件、图标与常量，同步更新修改 R.java 文件。正因为 R.java 文件是由开发工具自动生成的，所以我们应避免手工修改 R.java。R.java 在应用中起到了字典的作用，它包含了界面、图标、常量等各种资源的 id，通过 R.java，应用可以很方便地找到对应资源。另外编绎器也会检查 R.java 列表中的资源是否被使用到，没有被使用到的资源不会编绎进软件中，这样可以减少应用在手机占用的空间。



res/　资源(Resource)目录

在这个目录中我们可以存放应用使用到的各种资源，如 xml 界面文件，图标或常量

res/drawable 专门存放图标文件

res/layout 专门存放 xml 界面文件，xml 界面文件和 HTML 文件一样，主要用于用户界面显示

res/values 专门存放应用使用到的各种常量，作用和 struts 中的国际化资源文件一样。

AndroidManifest.xml 功能清单文件

这个文件列出了应用程序所提供的功能，在这个文件中，你可以指定应用程序使用到的服务(如电话服务、互联网服务、短信服务、GPS 服务等等)。另外当你新添加一个 Activity 的时候，也需要在这个文件中进行相应配置，只有配置好后，才能调用此 Activity。

default.properties 系统默认信息，一般是不需要修改此文件

proguard.cfg　proguard 代码混淆工具配置文件，可能需要修改修改此文件

从 SDK2.3 开始我们可以看到在 android-sdk-windows\tools\下面多了一个 proguard 文件夹。proguard 是一个 java 代码混淆的工具，通过 proguard，别人即使反编译你的 apk 包，也只会看到一些让人很难看懂的代码，从而达到保护代码的作用。

第 2 章　Text

2.1　Linkify

Android 实现 TextView 中文本链接的方式有很多种。

总结起来大概有 4 种：

1、通过 android:autoLink 属性来实现对 TextView 中文本相应类型的链接进行自动识别。

例如：android:autoLink = all 可以自动识别 TextView 文本中的网络地址，邮件地址，电话号码，地图位置等，并进行链接。

android:autoLink 所有支持的链接属性取值如下：

常量	值	描述
none	0x00	不进行自动识别 (默认).
web	0x01	自动识别网络地址
email	0x02	自动识别邮件地址
phone	0x04	自动识别电话号码
map	0x08	自动识别地图位置
all	0x0f	自动识别以上四种链接属性 (相当于 web|email|phone|map).

注：可以通过“|”符号连接多个属性值来支持多种类型的链接自动识别。例如，

android:autoLink = web|email|phone 支持对网络地址，邮件地址，电话号码的自动识别，并进行链接。

这是在 XML 文件中进行属性设置来识别链接的方式，还有一种在 Java 代码中进行属性设置的方式，同样可以实现类似功能。例如 TextView 对象 mTextView1，我们可以通过 mTextView1.setAutoLinkMask(int mask)来实现对 TextView 中文本相应类型的链接进行自动识别。其中 mask 所有取值如下：

常量		
int	ALL	自动识别邮件地址，网络地址，地图位置和电话号码
int	EMAIL_ADDRESSES	自动识别邮件地址
int	MAP_ADDRESSES	自动识别地图位置
int	PHONE_NUMBERS	自动识别电话号码
int	WEB_URLS	自动识别网络地址

注：使用时请在常量前面加上 Linkify.字样，例如：
mTextView1.setAutoLinkMask(Linkify.ALL)

2、将含有 HTML 链接标记的文本写在 Android 资源文件中，如 string.xml，然后在 Java 代码中直接引用。

3、通过 Html 类的 fromHtml（String source）方法来对含有 HTML 链接标记的文本进行格式化处理。

4、通过 Spannable 或继承它的类，如 SpannableString 来格式化部分字符串。关于 SpannableString 的详细用法，请参考：
http://blog.csdn.net/yang_hui1986527/article/details/6776629

注：默认情况下，第 2，3，4 种方法可以显示链接，但是无法响应用户的点击输入。如果需

要激活该响应，需要调用 TextView 对象的以下方法:
setMovementMethod(LinkMovementMethod.getInstance())

下面我们进行实例代码解析：

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="link_text_auto"><b>text1:</b> This is some text.  In
    this text are some things that are actionable.  For instance,
    you can click on http://www.google.com and it will launch the
    web browser.  You can click on google.com too.  And, if you
    click on (415) 555-1212 it should dial the phone.
   </string>
   <string name="link_text_manual"><b>text2:</b> This is some other
    text, with a <a href="http://www.google.com">link</a> specified
    via an &lt;a&gt; tag.  Use a \"tel:\" URL
    to <a href="tel:4155551212">dial a phone number</a>.
   </string>
</resources>
```

res-layout-link.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="vertical"
          android:layout_width="match_parent"
          android:layout_height="wrap_content">

 <!-- 四个 TextView 控件, 每个控件都显示包含链接的文本。 -->

 <!-- text1 自动识别文本链接，例如 URL 网络地址和电话号码等。 -->
 <TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/text1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:autoLink="all"
        android:text="@string/link_text_auto"
        />

 <!-- text2 使用包含用<a>等显式 HTML 标记来指定链接的文本资源。 -->
 <TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/text2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/link_text_manual"
        />

 <!-- text3 在 Java 代码中使用 HTML 类来构造包含链接的文本。 -->
 <TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/text3"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        />

 <!-- text4 在 Java 代码中不使用 HTML 类来构造包含链接的文本。  -->
 <TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/text4"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />

</LinearLayout>
```

src-com.example.android.apis.text-Link.java

```
package com.example.android.apis.text;

import com.example.android.apis.R;

import android.app.Activity;
import android.graphics.Typeface;
import android.os.Bundle;
import android.text.Html;
import android.text.SpannableString;
import android.text.Spanned;
import android.text.method.LinkMovementMethod;
import android.text.style.StyleSpan;
import android.text.style.URLSpan;
import android.widget.TextView;

public class Link extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      //super.onCreate(savedInstanceState)是调用父类的 onCreate 构造函数
      //savedInstanceState 是保存当前 Activity 的状态信息
      super.onCreate(savedInstanceState);

       //将 link 布局文件渲染出一个 View 对象，并作为 Activity 的默认 View
       setContentView(R.layout.link);

       // text1 通过 android:autoLink 属性自动识别文本中的链接，例如 URL 网络地址和电话号码等。
        // 不需要任何 java 代码来使之起作用。

       // text2 含有由<a>等 HTML 标记指定的文本链接。默认情况下，这些链接可以显示但不会响应用户输入。
       //要想这些链接响应用户的点击输入，你需要调用 TextView 的 setMovementMethod()方法 。

       TextView t2 = (TextView) findViewById(R.id.text2);
       t2.setMovementMethod(LinkMovementMethod.getInstance());
```

```
        // text3 显示在 java 代码中通过 HTML 类来创建包含文本链接的文本，而不是从文本资源中创建。
        //请注意，对于一个固定长度文本，最好像上面的例子一样，从文本资源中创建。
        // 这个例子仅仅说明您怎样去显示来自动态来源（例如，网络）的文本。

        TextView t3 = (TextView) findViewById(R.id.text3);
        t3.setText(
            Html.fromHtml(
                "<b>text3:</b>  Text with a " +
                "<a href=\"http://www.google.com\">link</a> " +
                "created in the Java source code using HTML."));
        t3.setMovementMethod(LinkMovementMethod.getInstance());

        // text4 举例说明完全不通过 HTML 标记来构建一个包含链接的格式化文本。
        // 对于固定长度的文本，你最好使用 string 资源文本（即在 string.xml 中指定），而不是硬编码值（即在 java 代码中指定）。

        //构建一个 SpannableString
        SpannableString ss = new SpannableString(
            "text4: Click here to dial the phone.");

        //设置粗体
        ss.setSpan(new StyleSpan(Typeface.BOLD), 0, 6,
                Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
        //设置电话号码的链接
        ss.setSpan(new URLSpan("tel:4155551212"), 13, 17,
                Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

        TextView t4 = (TextView) findViewById(R.id.text4);
        t4.setText(ss);
        t4.setMovementMethod(LinkMovementMethod.getInstance());
    }
}
```

知识点 1：android:id="@+id/text2"表示为相应对象新增一个 id 名（text1），方便在 Java 代码中引用该对象。引用方法为：R.id.id 名，如下所示：

TextView t2 = (TextView) findViewById(R.id.text2);

知识点 2：android:layout_width 和 android:layout_height

这两个是控件的布局属性，可以取值 FILL_PARENT，MATCH_PARENT，WRAP_CONTENT。其中 FILL_PARENT 和 MATCH_PARENT 代表该控件宽/高与 parent 相同，而 WRAP_CONTENT 代表该控件宽/高随着本身的内容而调整。

注：android2.2 以前我们使用 FILL_PARENT。从 android2.2 开始，FILL_PARENT 被弃用，改用 MATCH_PARENT。

知识点 3：android:orientation

一般用作 LinearLayout 线性布局的属性。android:orientation="vertical" 表示垂直布局；android:orientation="horizontal" 表示水平布局 。

2.2 LogTextBox

Android 中对于 Button 控件的监听方法主要有两种：

1、设置监听器

通过设置监听器来监听用户对于按钮的点击响应。当用户点击该按钮时，便会触发监听器，并执行监听器中 onClick 方法内部定义的指定动作。

```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // 点击触发时需要执行的动作
    }
});
```

2、自定义监听方法

首先需要在 Layout 布局文件中为该按钮添加属性（android:onClick="selfDestruct" ）。其中 selfDestruct 为自定义监听方法名称，后面需要用到。

```
<Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="@string/self_destruct"
        android:onClick="selfDestruct" />
```

接着，在 Activity 中实现自定义监听方法：

```
public void selfDestruct(View view) {
    // 点击触发时需要执行的动作
}
```

注： 1、该自定义方法必须是 Public 类型；

2、该自定义方法必须并且只能接受一个参数 View。

下面我们进行实例代码解析：

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="log_text_box_1_do_nothing_text">Do nothing</string>
    <string name="log_text_box_1_add_text">Add</string>
</resources>
```

res-layout-log_text_box_1.xml

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

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- android:onClick="selfDestruct" 监听方法二需要添加此属性，其中 selfDestruct 为自定义监听方法名称-->
```

```
    <Button
        android:id="@+id/add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/log_text_box_1_add_text"
        />

      <!--自定义控件 LogTextBox ， 继承自 TextView -->
    <com.example.android.apis.text.LogTextBox
        android:id="@+id/text"
        android:background="@drawable/box"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="1"
        android:scrollbars="vertical"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/log_text_box_1_do_nothing_text"/>

</LinearLayout>
```

src-com.example.android.apis.text-LogTextBox.java

```
package com.example.android.apis.text;

import android.widget.TextView;
import android.content.Context;
import android.text.method.ScrollingMovementMethod;
import android.text.method.MovementMethod;
import android.text.Editable;
import android.util.AttributeSet;

/**
 * 这是一个可以编辑并且默认可以滚动的 TextView 控件。
 * 类似缺少光标的 EditText 控件。
 *
 * <p>
 * <b>XML attributes</b>
 * <p>
 * See
 * {@link android.R.styleable#TextView TextView Attributes},
 * {@link android.R.styleable#View View Attributes}
 */
public class LogTextBox extends TextView {
    public LogTextBox(Context context) {
        this(context, null);
    }

    public LogTextBox(Context context, AttributeSet attrs) {
        this(context, attrs, android.R.attr.textViewStyle);
```

```
    }

    public LogTextBox(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    @Override
    protected MovementMethod getDefaultMovementMethod() {
        return ScrollingMovementMethod.getInstance();
    }

    @Override
    public Editable getText() {
        return (Editable) super.getText();
    }

    @Override
    public void setText(CharSequence text, BufferType type) {
        super.setText(text, BufferType.EDITABLE);
    }
}
```

src-com.example.android.apis.text-LogTextBox1.java

```
package com.example.android.apis.text;

import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;


public class LogTextBox1 extends Activity {

    private LogTextBox mText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.log_text_box_1);

        mText = (LogTextBox) findViewById(R.id.text);

        //对 Button 的监听方法一：设置监听器
        Button addButton = (Button) findViewById(R.id.add);
        addButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                mText.append("This is a test\n");
```

```
        } });


    }

    //对 Button 的监听方法二：自定义监听方法，需要设置 android:onClick 属性
    //这个方法必须是 Public 类型的，并且只能接受 view 作为唯一参数
    public void selfDestruct(View view) {
        mText.append("This is a test\n");
    }
}
```

知识点 1：*android:text="@string/log_text_box_1_add_text"*

在 android 中，建议将所有固定字符串资源放在 res/values/string.xml 文件中，方便统一管理。在 layout 布局文件等 xml 类型的文件中引用字符串资源时，通过*"@string/*字符串资源名称"格式进行引用。而在 Java 代码中需要引用字符串资源时，则通过格式 "R.string.字符串资源名"进行引用，具体引用方法如下所示：

```
//通过 string 字符串资源获得 CharSequence
CharSequence cs = getText(R.string.log_text_box_1_add_text);

//获取字符串资源，并且显示在 TextView 控件上
mText = (TextView) findViewById(R.id.text);
mText.setText(R.string.log_text_box_1_add_text);
```

知识点 2：*android:scrollbars="vertical"*

该属性定义控件在滚动时是否显示滚动条。该属性可以同时取多个值，但必须用"|"隔开。例如：横向纵向都支持滚动条显示（*android:scrollbars="horizontal | vertical "*）

常量	值	描述
none	0x00000000	不显示滚动条
horizontal	0x00000100	仅仅现实横向滚动条
vertical	0x00000200	仅仅现实纵向滚动条

知识点 3：android:layout_weight="1"

layout_weight 用于给一个线性布局中的诸多视图的重要度赋值。所有的视图都有一个 layout_weight 值，默认为零，意思是需要显示多大的视图就占据多大的屏幕空间。若赋一个高于零的值，则将父视图中的可用空间分割，分割大小具体取决于每一个视图的 layout_weight 值以及该值在当前屏幕布局的整体 layout_weight 值和在其它视图屏幕布局的 layout_weight 值中所占的比率而定。

举个例子：比如说我们在水平方向上有一个文本标签和两个文本编辑元素。该文本标签并无指定 layout_weight 值，所以它将占据需要提供的最少空间。如果两个文本编辑元素每一个的 layout_weight 值都设置为 1，则两者平分在父视图布局剩余的宽度(因为我们声明这两者的重要度相等)。如果两个文本编辑元素其中第一个的 layout_weight 值设置为 1，而第二个的设置为 2，则剩余空间的三分之一分给第一个，三分之而分给第二个(数值越大，重要度越高)。

关于 layout_weight 更完整的解释，请参考以下文章：
http://blog.csdn.net/jincf2011/article/details/6598256

注：值得注意的是，在水平布局中设置 layout_weight 的时候，必须这样进行设置 android:layout_width="0dip"。同理，在垂直布局中设置 layout_weight 时，也必须要做相应设置 android:layout_height="0dip"。

2.3 Marquee

在 TextView 及其子类控件中，当文本内容太长，超过控件长度时，默认情况下，无法完全显示文本内容。此时，通过在 xml 布局文件中设置控件的 android:ellipsize 属性，可以将无法显示的部分用省略号表示，并放在文本的起始，中间或者结束位置；还可以跑马灯的方式来显示文本（即文本控件获得焦点时，文本会进行滚动显示）。具体设置方法如下所示：

1、默认不处理

```
android:singleLine="true"
android:ellipsize="none"
```

2、省略号放在起始

```
android:singleLine="true"
android:ellipsize="start"
```

3、省略号放在中间

```
android:singleLine="true"
android:ellipsize="middle"
```

4、省略号放在结束

```
android:singleLine="true"
android:ellipsize="end"
```

5、跑马灯效果

```
android:focusable="true"
android:focusableInTouchMode="true"
android:singleLine="true"
android:ellipsize="marquee"
android:marqueeRepeatLimit="marquee_forever"
```

注：1、 android:singleLine="true" 表示单行显示。

2、在设置跑马灯效果时候，最好加上 android:focusable="true"和 android:focusableInTouchMode="true" ，分别表示可以获得焦点，和在触摸模式下可以获得焦点。

3、android:marqueeRepeatLimit 表示跑马灯效果重复显示的次数，只能取值 marquee_forever 和正整数。取值 marquee_forever 时，表示跑马灯效果一直重复显示。

下面我们进行实例代码解析：

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="marquee_default">This use the default marquee animation limit of 3</string>
    <string name="marquee_once">This will run the marquee animation once</string>
    <string name="marquee_forever">This will run the marquee animation forever</string>
</resources>
```

res-layout-marquee.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!-- 默认跑马灯效果 -->
    <Button
        android:layout_width="150dip"
        android:layout_height="wrap_content"
        android:text="@string/marquee_default"
        android:singleLine="true"
        android:ellipsize="marquee"/>

    <!-- 跑马灯效果，重复播放一次 -->
    <Button
        android:layout_width="150dip"
        android:layout_height="wrap_content"
        android:text="@string/marquee_once"
        android:singleLine="true"
        android:ellipsize="marquee"
        android:marqueeRepeatLimit="1"/>

     <!-- 跑马灯效果，一直重复播放 -->
    <Button
        android:layout_width="150dip"
        android:layout_height="wrap_content"
        android:text="@string/marquee_forever"
        android:singleLine="true"
        android:ellipsize="marquee"
        android:marqueeRepeatLimit="marquee_forever"/>

</LinearLayout>
```

src-com.example.android.apis.text-Marquee.java

```java
package com.example.android.apis.text;

import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;

public class Marquee extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //将 marquee 布局文件渲染出一个 View 对象，并作为 Activity 的默认 View
        setContentView(R.layout.marquee);
    }
}
```

第 3 章　Views

3.1　Buttons

Buttons 示例介绍了定义 Button，ToggleButton 的基本方法。从示例布局文件来看，主要是在线性 LinearLayout 中定义三个 Button，其中第一个为正常的 Button，而第二个通过 style 属性定义了一个小的 Button，第三个为 ToggleButton。

这里我们主要介绍下 ToggleButton。这是一种具有选中和未选中两种状态的按钮，类似开关按钮。通过“android:textOn”属性来设置选中状态下按钮上显示的文本，而相应的，android:textOff”则是用来设置未选中状态下按钮上显示的文本。具体设置参考如下：

```
android:textOn="开"
android:textOff="关"
```

ToggleButton 可以根据按钮状态的不同，来执行不同的响应动作。

下面主要介绍 ToggleButton 常用的两种监听方法：点击监听和状态改变监听

1、点击监听 OnClickListener

ToggleButton 的点击监听和普通 Button 的点击监听差不多，唯一不同的时，在响应点击时，ToggleButton 会根据点击后状态的不同，来执行不同的响应动作。

```
//声明 ToggleButton 对象
 private ToggleButton mtoggleBtn = null;

 //通过 findViewById 获得 ToggleButton
mtoggleBtn = (ToggleButton)findViewById(R.id.button_toggle);

//点击监听
mtoggleBtn.setOnClickListener(new ToggleButton.OnClickListener() {

        public void onClick(View v) {
            // TODO 点击按键时触发响应
            if(mtoggleBtn.isChecked()){
            //当按键被按下，处于选中状态时，执行此处定义的动作
            }
            else{
            //当按键被未被按下，处于未选中状态时，执行此处定义的动作
            }
        }
    });
```

2、状态改变监听　OnCheckedChangeListener

当 ToggleButton 的状态发生改变时，即状态从选中到未选中，或者从未选中到选中时，都会触发状态改变监听事件。而在响应时，ToggleButton 同样会根据改变后状态的不同，来执行不同的响应动作。普通 Button 不进行状态区分，也就没有状态改变监听事件。

```
        //声明 ToggleButton 对象
        private ToggleButton mtoggleBtn = null;

        //通过 findViewById 获得 ToggleButton
      mtoggleBtn = (ToggleButton)findViewById(R.id.button_toggle);

      //状态改变监听
      mtoggleBtn.setOnCheckedChangeListener(new ToggleButton.OnCheckedChangeListener()
{

                  public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                        // TODO 状态改变时触发响应
                        if(isChecked){
                        //当按键被按下，处于选中状态时，执行此处定义的动作
                        }
                        else{
                        //当按键被未被按下，处于未选中状态时，执行此处定义的动作
                        }
                  }
          });
```

下面我们进行实例代码解析：

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="buttons_1_normal">Normal</string>
   <string name="buttons_1_small">Small</string>
   <string name="buttons_1_toggle">Toggle</string>
</resources>
```

res-layout-buttons_1.xml

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

<!--很多按钮，可能需要滑动，所以需要放在 ScrollView 控件内部-->
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <LinearLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:orientation="vertical">

      <!-- 正常大小按钮 -->
      <Button android:id="@+id/button_normal"
         android:text="@string/buttons_1_normal"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />

      <!-- 小按钮 -->
```

```
    <Button android:id="@+id/button_small"
        style="?android:attr/buttonStyleSmall"
        android:text="@string/buttons_1_small"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <!-- 触发按钮，通常有两种状态，代表开和关
            android:textOff  按钮未选中时，显示该属性定义的文本
            android:textOn   按钮被选中时，显示该属性定义的文本 -->
    <ToggleButton android:id="@+id/button_toggle"
        android:text="@string/buttons_1_toggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

  </LinearLayout>

</ScrollView>
```

src-com.example.android.apis.view-Buttons1.java

```
package com.example.android.apis.view;


import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.ToggleButton;



public class Buttons1 extends Activity {
  //声明 ToggleButton 对象
    private ToggleButton mtoggleBtn = null;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.buttons_1);

    //通过 findViewById 获得 ToggleButton
    mtoggleBtn = (ToggleButton)findViewById(R.id.button_toggle);

    //点击监听
    mtoggleBtn.setOnClickListener(new ToggleButton.OnClickListener() {

                public void onClick(View v) {
                    // TODO 点击按键时触发响应
                    if(mtoggleBtn.isChecked()){
                    //当按键被按下，处于选中状态时，执行此处定义的动作
                    }
```

```
                else{
                //当按键被未被按下，处于未选中状态时，执行此处定义的动作
                }
            }
        });

    //状态改变监听
    mtoggleBtn.setOnCheckedChangeListener(new ToggleButton.OnCheckedChangeListener()
{

    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                // TODO 状态改变时触发响应
                if(isChecked){
                //当按键被按下，处于选中状态时，执行此处定义的动作
                }
                else{
                //当按键被未被按下，处于未选中状态时，执行此处定义的动作
                }
            }
    });
  }
}
```

知识点 1：style="?android:attr/buttonStyleSmall"

第二个按钮添加了此属性，表示该 Button 采用 android 系统内置的针对小按钮的样式 buttonStyleSmall。从帮助文档中，我们可以看出，系统内置属性的调用格式有两种，分别为 "@[+][package:]type:name" 和 "?[package:][type:]name"。 以样式 buttonStyleSmall 为例，套用两种格式分别为：style="@+android:attr/buttonStyleSmall" 和 style="?android:attr/buttonStyleSmall"。Button 的其他样式参考如下：

int	buttonStyle	正常按钮样式
int	buttonStyleInset	插入 EditText 的一种 Button 样式
int	buttonStyleSmall	小按钮样式
int	buttonStyleToggle	ToggleButton 样式

3.2 ImageButton

ImageButton 示例介绍了定义 ImageButton 的基本方法。从示例布局文件来看，主要是在线性 LinearLayout 中定义三个 ImageButton。不同的是，这三个按钮通过 android:src 属性分别引用三张来自 android 系统内置的图片作为按钮图标。

下面我们以 ImageButton 为例，简单介绍如何引用 android 系统内置图标资源。

1、在 java 代码中引用

在 java 代码中，我们通过“android.R.drawable.图标名称”格式来引用系统图标资源。具体参考如下：

```
    private ImageButton mImageButton = null;

        //通过 findViewById 获得 ImageButton
    mImageButton = (ImageButton)findViewById(R.id.myImageButton01);

    //引用 android 内置图标，作为 ImageButton 的按钮图标
```

```
mImageButton.setImageResource(android.R.drawable.sym_action_call);
```

2、在 xml 布局文件中

在布局文件中，我们通常按照"@android:drawable/图标名称" 格式来引用资源。具体参考如下：

```
<ImageButton
android:id="@+id/myImageButton01"
android:layout_width="100dip"
android:layout_height="50dip"
android:src="@android:drawable/sym_action_call" />
```

注：我们可以在 Android SDK 目录下找到这些系统内置图标资源，具体位置在对应 Android 版本的资源目录下。以 android 2.3 为例，这些图标在 android-sdk-linux_86/platforms/android-9/data/res/drawable-hdpi 目录下。

另外，除了引用 android 系统内置图标之外，我们也可以引用自定义图标。具体操作如下：

1、为了表示按钮不同状态（例如：被聚焦，被点击等），我们可以为每种状态定义一张图片。首先，我们准备三张类似的图片，放在 drawable 目录下。

butt  butt  d but  ed

2、在 drawable 目录下新建一个 xml 文件 btn_star.xml，通过"selector"来定义正常状态，聚焦状态下以及按下状态下分别显示什么图标。

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:state_pressed="true"
     android:drawable="@drawable/button_pressed" /> <!-- pressed -->
  <item android:state_focused="true"
     android:drawable="@drawable/button_focused" /> <!-- focused -->
  <item android:drawable="@drawable/button_normal" /> <!-- default -->
</selector>
```

3、以在 xml 布局文件中引用为例，在引用该自定义图标时，图标名称为定义在 drawable 下的 xml 文件名称（不包括 xml 后缀）。例如，上面我们定义了 btn_star.xml，引用时，可以这样引用："@android:drawable/btn_star"。

```
    <ImageButton
    android:id="@+id/myImageButton04"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@android:drawable/btn_star" />
```

下面我们进行实例代码解析：

res-layout-image_button_1.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

   <!-- ImageButton，引用 android 系统内置图标 sym_action_call（拨打电话）作为按钮图标 -->
  <ImageButton
    android:id="@+id/myImageButton01"
    android:layout_width="100dip"
    android:layout_height="50dip"
    android:src="@android:drawable/sym_action_call" />

   <!-- ImageButton，引用 android 系统内置图标 sym_action_chat（聊天）作为按钮图标 -->
    <ImageButton
    android:id="@+id/myImageButton02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@android:drawable/sym_action_chat" />

   <!-- ImageButton，引用 android 系统内置图标 sym_action_email（邮件）作为按钮图标 -->
    <ImageButton
    android:id="@+id/myImageButton03"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@android:drawable/sym_action_email" />
```

```
    <!-- ImageButton，引用自定义图标作为按钮图标 -->
      <ImageButton
      android:id="@+id/myImageButton04"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@android:drawable/btn_star" />
</LinearLayout>
```

src-com.example.android.apis.view-ImageButton1.java

```
package com.example.android.apis.view;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageButton;


import com.example.android.apis.R;


public class ImageButton1 extends Activity {
    private ImageButton mImageButton = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.image_button_1);

        //通过 findViewById 获得 ImageButton
        mImageButton = (ImageButton)findViewById(R.id.myImageButton01);

        //引用 android 内置图标，作为 ImageButton 的按钮图标
        mImageButton.setImageResource(android.R.drawable.sym_action_call);

    }
}
```

3.3 Visibility

Visibility 示例以 TextView 为例介绍了 View 的三种可见性以及如何设置 View 的可见性。这些可见性的设置方法同样适用于 View 以及其他继承自 View 的子类对象。从示例布局文件来看，主要分为两部分，一部分为一个线性垂直布局，包含三个不同背景色的 TextVew 对象；另一部分，为一个线性水平布局，包含三个 Button 对象。

View 的可见性主要分为三种， VISIBLE（可见）、 INVISIBLE（不可见）、GONE（彻底隐藏）。这三种可见性的区别在于：

1、VISIBLE（可见）代表该 View 对象显示可见；

2、INVISIBLE（不可见）代表该 View 对象隐藏不可见，但是仍然占据着 Layout 布局中的位置；

3、GONE（彻底隐藏）代表该 View 对象彻底隐藏不可见，并且不占据 Layout 布局中的位置；

下面我们以 TextView 为例，简单介绍 View 的三种可见性以及相应的设置方法。

1、在 Java 代码中设置可见性

在 Java 代码中，我们通过函数 public void setVisibility (int visibility)来设置 View 的可见性，其中参数 visibility 取值范围如下：View.*VISIBLE*、View.*INVISIBLE*、以及 View.*GONE*。具体参考如下：

```
private View mVictim = null;

// 通过 findViewById 获得一个待改变可见性的 View 对象
mVictim = findViewById(R.id.victim);

//设置 mVictim 彻底隐藏
mVictim.setVisibility(View.GONE);
```

2、在 xml 布局文件中

在 xml 布局文件中，我们通过设置“android:visibility”属性来设置 View 可见性。该属性取值范围如下所示：

常量	值	描述
visible	0	屏幕可见，默认取值。
invisible	1	屏幕上显示，但是在 Layout 布局上会占据相应位置。
gone	2	彻底隐藏不显示，并且不会在 Layout 布局上占据任何位置。

具体设置参考如下：

```
<TextView android:id="@+id/victim"
  android:background="@drawable/green"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:visibility="visible"
   android:text="@string/visibility_1_view_2"/>
```

下面我们进行实例代码解析：

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="visibility_1_view_1">View A</string>
   <string name="visibility_1_view_2">View B</string>
   <string name="visibility_1_view_3">View C</string>
   <string name="visibility_1_vis">Vis</string>
   <string name="visibility_1_invis">Invis</string>
   <string name="visibility_1_gone">Gone</string>
</resources>
```

res-layout-visibility_1.xml

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

<!--改变 View 可见性的实例演示，请参考相应的 Java 代码。 -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

     <!--一个线性垂直布局，包含三个 TextView 对象 -->
   <LinearLayout
    android:orientation="vertical"
    android:background="@drawable/box"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

       <!--一个背景色为红色的 TextView 对象 -->
    <TextView
       android:background="@drawable/red"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:visibility="visible"
       android:text="@string/visibility_1_view_1"/>

       <!--一个背景色为绿色的 TextView 对象 -->
    <TextView android:id="@+id/victim"
       android:background="@drawable/green"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
```

```
        android:visibility="visible"
        android:text="@string/visibility_1_view_2"/>

        <!--一个背景色为蓝色的 TextView 对象 -->
    <TextView
        android:background="@drawable/blue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="visible"
        android:text="@string/visibility_1_view_3"/>

    </LinearLayout>

     <!--一个线性水平布局，包含三个 Button 对象 -->
    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">

      <Button android:id="@+id/vis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/visibility_1_vis"/>

      <Button android:id="@+id/invis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/visibility_1_invis"/>

      <Button android:id="@+id/gone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/visibility_1_gone"/>

    </LinearLayout>
</LinearLayout>
```

src-com.example.android.apis.view-Visibility1.java

```
package com.example.android.apis.view;


import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;


/**
 * 设置一个 View 对象可见，不可见，彻底消失的实例演示
```

```
 *
 */
public class Visibility1 extends Activity {

    private View mVictim = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.visibility_1);

        // 通过 findViewById 获得一个待改变可见性的 View 对象
        mVictim = findViewById(R.id.victim);

        // 通过 findViewById 获得三个 Button 对象
        Button visibleButton = (Button) findViewById(R.id.vis);
        Button invisibleButton = (Button) findViewById(R.id.invis);
        Button goneButton = (Button) findViewById(R.id.gone);

        // 给每个 Button 添加点击监听器
        visibleButton.setOnClickListener(mVisibleListener);
        invisibleButton.setOnClickListener(mInvisibleListener);
        goneButton.setOnClickListener(mGoneListener);
    }


    OnClickListener mVisibleListener = new OnClickListener() {
        public void onClick(View v) {
                //设置 mVictim 可见
            mVictim.setVisibility(View.VISIBLE);
        }
    };

    OnClickListener mInvisibleListener = new OnClickListener() {
        public void onClick(View v) {
                //设置 mVictim 不可见
            mVictim.setVisibility(View.INVISIBLE);
        }
    };

    OnClickListener mGoneListener = new OnClickListener() {
        public void onClick(View v) {
                //设置 mVictim 彻底隐藏
            mVictim.setVisibility(View.GONE);
        }
    };
}
```

知识点 1:android:background="@drawable/red"

该属性用于设置 View 的背景，取值可以是背景图片或者背景颜色，可以是系统内置的，也可以是自定义的。其中系统内置图片的引用上一节已经讲过，这里不再赘述。这里我们简单介绍如何定义和引用自定义颜色。

1、打开工程下 res-value-color.xml 文件，如果没有，请新建一个，文件内容参考如下：
其中 name 字段为颜色资源名称，#开头的数字为具体颜色值。

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

<resources>
   <drawable name="red">#7f00</drawable>
   <drawable name="blue">#770000ff</drawable>
   <drawable name="green">#7700ff00</drawable>
     <drawable name="yellow">#77ffff00</drawable>

     <drawable name="screen_background_black">#ff000000</drawable>
   <drawable name="translucent_background">#e0000000</drawable>
   <drawable name="transparent_background">#00000000</drawable>

   <color name="solid_red">#f00</color>
   <color name="solid_blue">#0000ff</color>
   <color name="solid_green">#f0f0</color>
   <color name="solid_yellow">#ffffff00</color>

</resources>
```

2、对于 drawable 和 color 两种标签的颜色资源，引用方式稍有不同，具体参考如下：
对于 color 颜色资源的引用：

```
    <TextView
       android:background="@color/solid_blue"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:visibility="visible"
         android:text="@string/visibility_1_view_1"/>
```

对于 drawable 颜色资源的引用：

```
    <TextView
       android:background="@drawable/red"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:visibility="visible"
         android:text="@string/visibility_1_view_1"/>
```

3.4 WebView

WebView 是一个专门用来显示网页的 View 子类。它使用 WebKit 渲染引擎来显示网页，并且支持包括前进，后退，放大，缩小，文本搜索等多种功能。

WebView 有一个辅助类叫 WebSettings，它管理 WebView 的设置状态。该类的对象可以通过 WebView.getSettings()方法获得。下面我们介绍几个常用的 WebSettings 设置：

```
//得到 WebView 对象
WebView mWebView = (WebView) findViewById(R.id.wv1);

//通过 WebView 得到 WebSettings 对象
WebSettings mWebSettings = mWebView.getSettings();

//设置支持 Javascript 的参数
mWebSettings.setJavaScriptEnabled(true);

//设置可以支持缩放
mWebSettings.setSupportZoom(true);

//设置出现缩放工具
mWebSettings.setBuiltInZoomControls(true);

//设置默认缩放方式尺寸是 far
mWebSettings.setDefaultZoom(WebSettings.ZoomDensity.FAR);

//设置允许访问文件数据
mWebSettings.setAllowFileAccess(true);

//设置是否保存密码
mWebSettings.setSavePassword(true);

//设置网页默认编码
mWebSettings.setDefaultTextEncodingName(encoding);
```

有关 WebSettings 更多的设置选项，请参考 Android SDK 的帮助文档。

下面我们结合实例，简单介绍下通过 WebView 访问网页的三种常用方法：loadUrl、loadData 以及 loadDataWithBaseURL。

1、**public void** loadUrl(String url)

loadUrl 方法很简单，只需要提供一个参数，即 url 地址，就可以进行自动加载。该方法适用于加载网页、网页图片、本地网页以及本地图片。具体参考如下：

```
        WebView wv = (WebView) findViewById(R.id.wv1);
        wv.requestFocus(); //请求焦点
        wv.getSettings().setJavaScriptEnabled(true); //设置是否支持 JavaScript
        wv.getSettings().setSupportZoom(true); //设置是否支持缩放
        wv.getSettings().setBuiltInZoomControls(true); //设置是否显示内建缩放工具
        wv.getSettings().setSavePassword(true); //设置是否保存密码
        wv.loadUrl("http://www.baidu.com/");        //加载在线网页
        wv.loadUrl("http://www.google.com/logos/2011/Googles_13th_Birthday-2011-hp.jpg"); //
    加载在线图片
        wv.loadUrl("file:///mnt/sdcard/Google.html"); //加载本地网页(SD 卡根目录下)
         wv.loadUrl("file:///mnt/sdcard/Googles_13th_Birthday-2011-hp.jpg");    //加载本地图片
    (SD 卡根目录下)
```

注：　1、后面四句包含“loadUrl”的语句分别对应加载在线网页/在线图片/本地网页/本地图片四中情况，每种情况下只取其一。

2、对于 sdcard 中的本地网页和图片等资源，一般 url 地址以“file:///mnt/sdcard/”开头；

3、对于 android 项目中 assets 目录下的本地网页和图片等资源， 一般 url 地址以“ file:///android_asset/ "开头；

2、**public void** loadData(String data, String mimeType, String encoding)

loadData 将指定的 data 加载到 WebView 中。由于本身机制的原因，该方法不能加载来自网络的内容。其中，data 内的 html 代码中若含有以下四种字符'#', '%', '\', '?'，则应该依次替换为%23, %25, %27, %3f 。 参数 mimeType，即 *MIME Type*，也就是该资源的媒体类型，可以取值 text/html, image/jpeg 等。参数 encoding 为网页编码，可以取值 utf-8, base64 等。具体参考如下：

```
        final String mimeType = "text/html";   // image/jpeg  etc
         final String encoding = "utf-8";   //base64  etc
         String data = null;

        WebView wv = (WebView) findViewById(R.id.wv7);
        data = "<a href ='http://www.baidu.com/'>百度</a>";  //加载本地网页 Html 代码
        data = "loadData 方法加载本地图片<img src ='file:///mnt/sdcard/Googles_13th_Birthday-
    2011-hp.jpg' />";   //加载本地图片(SD 卡根目录下)
        wv.requestFocus();
        wv.getSettings().setDefaultTextEncodingName(encoding);  //对于 data 中含有中文字符的，
    必须加上这个设置，否则会乱码。
         wv.loadData(data, mimeType, encoding);
```

注：1、中间两个 data 字段分别对应加载本地网页代码和本地图片的情况；
2、经过实际操作，发现前面文本超级链接的 html 代码可以正常显示。点击超级链接，调用外部 web 浏览器打开链接。
3、经过实际操作，发现后面的本地图片其实加载不上来，无法显示。
4、loadData 方法中，如果 data 含有中文字符，则需要调用 setDefaultTextEncodingName 方法来设置 WebView 的 text 指定编码。详情请点击阅读：《关于 WebView 的 loadData 方法》

3、**public void** loadDataWithBaseURL(String baseUrl, String data,
String mimeType, String encoding, String historyUrl)
loadDataWithBaseURL 方法也是将指定的 data 加载到 WebView 中。由于本身机制的原因，该方法不能加载来自网络的内容。参数 mimeType，即 MIME Type，也就是该资源的媒体类型，可以取值 text/html, image/jpeg 等。参数 encoding 为网页编码，可以取值 utf-8, base64 等。参数 baseUrl 为基础目录，data 中的文件路径可以是相对于基础目录的相对目录。例如：文件 **file:///mnt/sdcard/Googles_13th_Birthday-2011-hp.jpg**，baseUrl 为 **file:///mnt/sdcard/**，那么 data 中就可以直接引用图片 Googles_13th_Birthday-2011-hp.jpg 了。参数 historyUrl 用作历史记录的字段，可以设置为 null。具体参考如下：

```
        final String mimeType = "text/html";   // image/jpeg  etc
        final String encoding = "utf-8";   //base64  etc

        String baseUrl = null;
        String data = null;

       WebView wv = (WebView) findViewById(R.id.wv9);
       baseUrl = "file:///mnt/sdcard/";   //网页基础目录
       data = "<a href='Google.html'>谷歌</a>";  //加载本地网页 Html 代码(SD 卡根目录下)
       data = "<img src ='Googles_13th_Birthday-2011-hp.jpg' />";  //：加载本地图片(SD 卡根目录下)
       wv.requestFocus();
        wv.loadDataWithBaseURL(baseUrl, data, mimeType, encoding, null);
```

注：中间两个 data 字段分别对应加载本地网页和本地图片的情况。经过测试，网页和本地图片都能正常加载。

下面我们进行实例代码解析：

res-layout-webview_1.xml

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

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">

   <!--一个线性垂直布局，包含十个 WebView 对象 -->
   <LinearLayout
      android:orientation="vertical"
      android:layout_width="match_parent"
      android:layout_height="wrap_content">

      <WebView android:id="@+id/wv1"
         android:layout_height="wrap_content"
         android:layout_width="match_parent"
```

```
        />

    <WebView android:id="@+id/wv2"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv3"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv4"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv5"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv6"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv7"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv8"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv9"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />

    <WebView android:id="@+id/wv10"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        />
  </LinearLayout>

</ScrollView>
```

src-com.example.android.apis.view-WebView1.java

```
package com.example.android.apis.view;

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

import com.example.android.apis.R;


/**
 *<pre>WebView 组件(mWebView)有一个辅助类叫 WebSettings，它管理 WebView 的设置状态。
 *该对象可以通过 WebView.getSettings()方法获得。

 *得到 WebSettings 对象，设置支持 Javascript 的参数
 *mWebView.getSettings().setJavaScriptEnabled(true);

 *设置可以支持缩放
 *mWebView.getSettings().setSupportZoom(true);

 *设置默认缩放方式尺寸是 far
 *mWebView.getSettings().setDefaultZoom(WebSettings.ZoomDensity.FAR);

 *设置允许访问文件数据
 *mWebView.getSettings().setAllowFileAccess(true);

 *设置是否保存密码
 *mWebView.getSettings().setSavePassword(true);

 *设置网页默认编码
 *mWebView.getSettings().setDefaultTextEncodingName(encoding);

 *设置出现缩放工具
 *mWebView.getSettings().setBuiltInZoomControls(true);</pre>
 *
 */
public class WebView1 extends Activity {

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        setContentView(R.layout.webview_1);

        final String mimeType = "text/html";   // image/jpeg  etc
        final String encoding = "utf-8";   //base64  etc

        WebView wv = null;
        String baseUrl = null;
        String data = null;
```

```
        //加载在线网页
        wv = (WebView) findViewById(R.id.wv1);
        wv.requestFocus(); //请求焦点
        wv.getSettings().setJavaScriptEnabled(true); //设置是否支持 JavaScript
        wv.getSettings().setSupportZoom(true); //设置是否支持缩放
        wv.getSettings().setBuiltInZoomControls(true); //设置是否显示内建缩放工具
        wv.getSettings().setSavePassword(true); //设置是否保存密码
        wv.loadUrl("http://www.baidu.com/");

        //加载在线图片
        wv = (WebView) findViewById(R.id.wv2);
        wv.requestFocus();
        wv.loadUrl("http://www.google.com/logos/2011/Googles_13th_Birthday-2011-hp.jpg");

        //加载本地网页(SD 卡根目录下)
        wv = (WebView) findViewById(R.id.wv3);
        wv.requestFocus();
        wv.loadUrl("file:///mnt/sdcard/Google.html");

        //加载本地图片(SD 卡根目录下)
        wv = (WebView) findViewById(R.id.wv4);
        wv.requestFocus();
        wv.loadUrl("file:///mnt/sdcard/Googles_13th_Birthday-2011-hp.jpg");

        //加载在线网页失败，把网页地址当作字符串处理
        wv = (WebView) findViewById(R.id.wv5);
        data = "http://www.baidu.com/";
        wv.requestFocus();
        wv.loadData(data, mimeType, encoding);

        //加载在线图片失败，只显示一个默认图片进行替换
        wv = (WebView) findViewById(R.id.wv6);
        data = "http://www.google.com/logos/2011/Googles_13th_Birthday-2011-hp.jpg";
        wv.requestFocus();
        wv.loadData(data, mimeType, encoding);

        //加载本地网页 Html 代码，可以显示超级链接。点击，调用外部浏览器打开该链接。
        wv = (WebView) findViewById(R.id.wv7);
        data = "<a href ='http://www.baidu.com/'>百度</a>";
        wv.requestFocus();
        wv.getSettings().setDefaultTextEncodingName(encoding);  //对于 data 中含有中文字符的，
必须加上这个设置，否则会乱码。
        wv.loadData(data, mimeType, encoding);

        //加载本地图片(SD 卡根目录下)
        wv = (WebView) findViewById(R.id.wv8);
        data = "loadData 方法加载本地图片<img src ='file:///mnt/sdcard/Googles_13th_Birthday-
2011-hp.jpg' />";
        wv.requestFocus();
        wv.getSettings().setDefaultTextEncodingName(encoding); //对于 data 中含有中文字符的，
必须加上这个设置，否则会乱码。
        wv.loadData(data, mimeType, encoding);
```

```
        //loadDataWithBaseURL 方法：加载本地网页 Html 代码(SD 卡根目录下)，可以显示超级链接。点击直接显示本地网页内容。
        wv = (WebView) findViewById(R.id.wv9);
        baseUrl = "file:///mnt/sdcard/";  //网页基础目录
        data = "<a href='Google.html'>谷歌</a>";  //相对路径，相对基础目录而言
        wv.requestFocus();
        wv.loadDataWithBaseURL(baseUrl, data, mimeType, encoding, null);

        //loadDataWithBaseURL 方法：加载本地图片(SD 卡根目录下)，默认直接显示该图片
        wv = (WebView) findViewById(R.id.wv10);
        baseUrl = "file:///mnt/sdcard/";  //网页基础目录
        data = "<img src ='Googles_13th_Birthday-2011-hp.jpg' />";  //相对路径，相对基础目录而言
        wv.requestFocus();
        wv.loadDataWithBaseURL(baseUrl, data, mimeType, encoding, null);
    }
}
```

关于 WebView 更详细的用法介绍，请点击阅读：《超好的 webview 学习资料》

3.5 Date Widgets

Date Widgets 大致可以分为两类，一类是弹出对话框类型的控件，包括 DatePickerDialog（日期选择对话框控件）和 TimePickerDialog（时间选择对话框控件）；另一类就是内嵌类型的控件，包括 DatePicker（日期选择窗口控件）和 TimePicker（时间选择窗口控件）。这些控件广泛应用在需要选择和记录时间信息的场合，例如：谷歌日历。

下面我们简单介绍下这四种控件的调用方法：

1、DatePickerDialog

```
// 日期和时间相关定义数据
private int mYear;
private int mMonth;
private int mDay;
private int mHour;
private int mMinute;

// 通过 java 类 Calendar 获得系统当前时间数据信息
final Calendar c = Calendar.getInstance();
mYear = c.get(Calendar.YEAR);
mMonth = c.get(Calendar.MONTH);
mDay = c.get(Calendar.DAY_OF_MONTH);
mHour = c.get(Calendar.HOUR_OF_DAY);
mMinute = c.get(Calendar.MINUTE);

// 创建日期选择对话框
DatePickerDialog mDatePickerDialog = new DatePickerDialog(this,
                    new DatePickerDialog.OnDateSetListener() {

            public void onDateSet(DatePicker view, int year, int monthOfYear,
                                                    int dayOfMonth) {
                                            mYear = year;
                                            mMonth = monthOfYear;
                                            mDay = dayOfMonth;
                                            /*这里放更新日期的方法*/
                                    }
                            }, mYear, mMonth, mDay);
mDatePickerDialog.show();
```

注：　创建 DatePickerDialog 时，5 个参数以及它们对应的含义如下：

context: 对话框的上下文环境 context

callBack: 对话框监听器 DatePickerDialog.OnDateSetListener

Year: 对话框创建时显示的初始年份

MonthOfYear:对话框创建时显示的初始月份

DayOfMonth:对话框创建时显示的初始日期

2、TimePickerDialog

```
// 日期和时间相关定义数据
private int mYear;
private int mMonth;
private int mDay;
private int mHour;
private int mMinute;

// 通过 java 类 Calendar 获得系统当前时间数据信息
final Calendar c = Calendar.getInstance();
mYear = c.get(Calendar.YEAR);
mMonth = c.get(Calendar.MONTH);
mDay = c.get(Calendar.DAY_OF_MONTH);
mHour = c.get(Calendar.HOUR_OF_DAY);
mMinute = c.get(Calendar.MINUTE);

// 创建时间选择对话框
TimePickerDialog mTimePickerDialog =  new TimePickerDialog(this,
                    new TimePickerDialog.OnTimeSetListener() {

            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                                    mHour = hourOfDay;
                                    mMinute = minute;
                                    /*这里放更新时间的方法*/
                                }
                    }, mHour, mMinute,true);
        mTimePickerDialog.show();
```

注：　创建 TimePickerDialog 时，5 个参数以及它们对应的含义如下：
context: 对话框的上下文环境 context
callBack: 对话框监听器 TimePickerDialog.OnTimeSetListener
hourOfDay: 对话框创建时显示的初始小时
minute:对话框创建时显示的初始分钟
is24HourView: 是否显示 24 小时制视图。true 则显示，否则显示 12 小时制视图

3、DatePicker

```
// 日期和时间相关定义数据
private int mYear;
private int mMonth;
private int mDay;
private int mHour;
private int mMinute;
```

```
// 通过 java 类 Calendar 获得系统当前时间数据信息
final Calendar c = Calendar.getInstance();
mYear = c.get(Calendar.YEAR);
mMonth = c.get(Calendar.MONTH);
mDay = c.get(Calendar.DAY_OF_MONTH);
mHour = c.get(Calendar.HOUR_OF_DAY);
mMinute = c.get(Calendar.MINUTE);

// 通过 findViewById 方法获得一个 DatePicker 对象
DatePicker datePicker = (DatePicker) findViewById(R.id.datePicker);
datePicker.init(mYear, mMonth, mDay, new DatePicker.OnDateChangedListener(){

                public void onDateChanged(DatePicker view, int year, int monthOfYear,
                                int dayOfMonth) {
                        mYear = year;
                        mMonth = monthOfYear;
                        mDay = dayOfMonth;
                /*这里放更新日期的方法*/
                }
});
```

注：　创建 DatePicker 时，需要注意的是，使用的是方法是 init()，这与其他三个控件创建方法不一样，切忌混淆。4 个参数以及它们对应的含义如下：

Year:　DatePicker 创建时显示的初始年份
MonthOfYear:　DatePicker 创建时显示的初始月份
DayOfMonth:　DatePicker 创建时显示的初始日期
callBack:　监听器　DatePicker.OnDateChangedListener

4、TimePicker

```
//  日期和时间相关定义数据
private int mYear;
private int mMonth;
private int mDay;
private int mHour;
private int mMinute;

// 通过 java 类 Calendar 获得系统当前时间数据信息
final Calendar c = Calendar.getInstance();
mYear = c.get(Calendar.YEAR);
mMonth = c.get(Calendar.MONTH);
mDay = c.get(Calendar.DAY_OF_MONTH);
mHour = c.get(Calendar.HOUR_OF_DAY);
mMinute = c.get(Calendar.MINUTE);
```

```
    // 通过 findViewById 方法获得一个 TimePicker 对象
    TimePicker timePicker = (TimePicker) findViewById(R.id.timePicker);
    timePicker.setIs24HourView(true); //参数 is24HourView，如果是 true，则为 24 小时制，否则，则为 12 小时制。
    timePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {

                public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
                    mHour = hourOfDay;
                    mMinute = minute;
                    /*这里放更新时间的方法*/
                }
    });
```

下面我们进行实例代码解析：

实例一：DateWidgets1

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="date_widgets_example_dateDisplay_text"></string>
   <string name="date_widgets_example_pickTime_text">change the time</string>
   <string name="date_widgets_example_pickDate_text">change the date</string>
</resources>
```

res-layout-date_widgets_example_1.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="vertical">

     <!-- 日期和时间显示区 -->
   <LinearLayout android:layout_width="wrap_content"
         android:layout_height="wrap_content">
      <TextView android:id="@+id/dateDisplay"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/date_widgets_example_dateDisplay_text"/>
   </LinearLayout>

     <!-- 按钮一，点击弹出日期选择对话框 -->
   <Button android:id="@+id/pickDate"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/date_widgets_example_pickDate_text"/>

     <!-- 按钮二，点击弹出时间选择对话框 -->
   <Button android:id="@+id/pickTime"
         android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="@string/date_widgets_example_pickTime_text"/>


</LinearLayout>
```

src-com.example.android.apis.view-DateWidgets1.java

```
package com.example.android.apis.view;

import com.example.android.apis.R;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.view.View;

import java.util.Calendar;

public class DateWidgets1 extends Activity {

    // 日期和时间显示区域
    private TextView mDateDisplay;

    // 日期和时间相关定义数据
    private int mYear;
    private int mMonth;
    private int mDay;
    private int mHour;
    private int mMinute;

    // 对话框标示，用于创建对话框时进行区分
    static final int TIME_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.date_widgets_example_1);

        // 通过 findViewById 方法获得一个 TextView 对象
        mDateDisplay = (TextView) findViewById(R.id.dateDisplay);

        // 通过 findViewById 方法获得一个 Button 对象：pickDate，并设置监听器。
        Button pickDate = (Button) findViewById(R.id.pickDate);
        pickDate.setOnClickListener(new View.OnClickListener() {
```

```
                public void onClick(View v) {
                        showDialog(DATE_DIALOG_ID); // 点击按钮时，触发响应，创建或者显示日期选择对话框
                }
        });

        // 通过 findViewById 方法获得一个 Button 对象：pickTime，并设置监听器。
        Button pickTime = (Button) findViewById(R.id.pickTime);
        pickTime.setOnClickListener(new View.OnClickListener() {

                public void onClick(View v) {
                        showDialog(TIME_DIALOG_ID); // 点击按钮时，触发响应，创建或者显示时间选择对话框
                }
        });

        // 通过 java 类 Calendar 获得系统当前时间数据信息，并更新显示在 TextView 上
        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);
        mHour = c.get(Calendar.HOUR_OF_DAY);
        mMinute = c.get(Calendar.MINUTE);
        updateDisplay();
    }

    // 根据对话框标识创建对话框
    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
        case TIME_DIALOG_ID:
                // 创建时间选择对话框 最后一个参数 is24HourView，如果是 true，则为 24 小时制，否则，则为 12 小时制。
                return new TimePickerDialog(this, mTimeSetListener, mHour, mMinute,
                                true);
        case DATE_DIALOG_ID:
                // 创建日期选择对话框
                return new DatePickerDialog(this, mDateSetListener, mYear, mMonth,
                                mDay);
        }
        return null;
    }

    // 根据对话框标识更新对话框
    @Override
    protected void onPrepareDialog(int id, Dialog dialog) {
        switch (id) {
        case TIME_DIALOG_ID:
                // 更新时间选择对话框信息
                ((TimePickerDialog) dialog).updateTime(mHour, mMinute);
                break;
```

```
            case DATE_DIALOG_ID:
                // 更新日期选择对话框信息
                ((DatePickerDialog) dialog).updateDate(mYear, mMonth, mDay);
                break;
            }
    }

    // 更新日期和时间显示区的信息
    private void updateDisplay() {
            mDateDisplay.setText(new StringBuilder()
            // 由于月份是按照从 0 到 11 进行计算，因此显示的时候加上 1，进行转换。
            .append(mMonth + 1).append("-").append(mDay).append("-")
            .append(mYear).append(" ").append(pad(mHour)).append(":")
            .append(pad(mMinute)));
    }

    // 理论上可以修改系统时间，但由于系统限制，缺乏 root 权限，实际上这个函数并不能真正去修改系统时间。
    private void updateSystemTime() {
            final Calendar c = Calendar.getInstance();
            // 由于月份是按照从 0 到 11 进行计算，因此设置的时候减去 1，进行转换。
            c.set(mYear, mMonth - 1, mDay, mHour, mMinute);
    }

    // 日期设置监听器，当改变日期时，更新时间显示信息
    private DatePickerDialog.OnDateSetListener mDateSetListener = new DatePickerDialog.OnDateSetListener() {

            public void onDateSet(DatePicker view, int year, int monthOfYear,
                            int dayOfMonth) {
                    mYear = year;
                    mMonth = monthOfYear;
                    mDay = dayOfMonth;
                    updateSystemTime();
                    updateDisplay();
            }
    };

    // 时间设置监听器，当改变时间时，更新时间显示信息
    private TimePickerDialog.OnTimeSetListener mTimeSetListener = new TimePickerDialog.OnTimeSetListener() {

            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                    mHour = hourOfDay;
                    mMinute = minute;
                    updateSystemTime();
                    updateDisplay();
            }
    };

    // 当小时或者分钟为个位数字时，前面加一个 0
    private static String pad(int c) {
```

```
            if (c >= 10)
                return String.valueOf(c);
            else
                return "0" + String.valueOf(c);
    }
}
```

实例二：DateWidgets2

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="date_widgets_example_dateDisplay_text"></string>
   <string name="date_widgets_example_pickTime_text">change the time</string>
   <string name="date_widgets_example_pickDate_text">change the date</string>
</resources>
```

res-layout-date_widgets_example_2.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="vertical">

   <!-- 内嵌日期选择控件 -->
   <DatePicker android:id="@+id/datePicker"
      android:layout_height="wrap_content"
      android:layout_width="match_parent"/>

    <!-- 内嵌时间选择控件 -->
   <TimePicker android:id="@+id/timePicker"
      android:layout_height="wrap_content"
      android:layout_width="match_parent"/>

      <!-- 日期和时间显示区 -->
   <TextView android:id="@+id/dateDisplay"
         android:layout_height="wrap_content"
         android:layout_width="match_parent"
         android:paddingLeft="4dip"
         android:text="@string/date_widgets_example_dateDisplay_text"/>
</LinearLayout>
```

src-com.example.android.apis.view-DateWidgets2.java

```
package com.example.android.apis.view;

import java.util.Calendar;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.DatePicker;

import com.example.android.apis.R;

public class DateWidgets2 extends Activity {

    // 日期和时间显示区域
    private TextView mTimeDisplay;

    // 日期和时间相关定义数据
    private int mYear;
    private int mMonth;
    private int mDay;
    private int mHour;
    private int mMinute;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.date_widgets_example_2);

        // 通过 findViewById 方法获得一个 TextView 对象，用作日期和时间显示区
        mTimeDisplay = (TextView) findViewById(R.id.dateDisplay);

        // 通过 java 类 Calendar 获得系统当前时间数据信息，并更新显示在 TextView 上
        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);
        mHour = c.get(Calendar.HOUR_OF_DAY);
        mMinute = c.get(Calendar.MINUTE);
        updateDisplay();

        // 通过 findViewById 方法获得一个 DatePicker 对象
        DatePicker datePicker = (DatePicker) findViewById(R.id.datePicker);
        datePicker.init(mYear, mMonth, mDay, new DatePicker.OnDateChangedListener(){

            public void onDateChanged(DatePicker view, int year, int monthOfYear,
                    int dayOfMonth) {
                mYear = year;
                mMonth = monthOfYear;
                mDay = dayOfMonth;
                updateSystemTime();
                updateDisplay();
            }
        });
```

```
        // 通过 findViewById 方法获得一个 TimePicker 对象
        TimePicker timePicker = (TimePicker) findViewById(R.id.timePicker);
        timePicker.setIs24HourView(true); //参数 is24HourView，如果是 true，则为 24 小时制，否则，则为 12 小时制。
        timePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener()
{

            public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
                mHour = hourOfDay;
                mMinute = minute;
                updateSystemTime();
                updateDisplay();
            }
        });

    }

    // 更新日期和时间显示区的信息
    private void updateDisplay() {
        mTimeDisplay.setText(new StringBuilder()
        // 由于月份是按照从 0 到 11 进行计算，因此显示的时候加上 1，进行转换。
        .append(mMonth + 1).append("-").append(mDay).append("-")
        .append(mYear).append(" ").append(pad(mHour)).append(":")
        .append(pad(mMinute)));
    }



    // 理论上可以修改系统时间，但由于系统限制，缺乏 root 权限，实际上这个函数并不能真正去修改系统时间。
    private void updateSystemTime() {
        final Calendar c = Calendar.getInstance();
        // 由于月份是按照从 0 到 11 进行计算，因此设置的时候减去 1，进行转换。
        c.set(mYear, mMonth - 1, mDay, mHour, mMinute);
    }

    // 当小时或者分钟为个位数字时，前面加一个 0
    private static String pad(int c) {
        if (c >= 10)
            return String.valueOf(c);
        else
            return "0" + String.valueOf(c);
    }

}
```

知识点 1：如何修改系统时间？

最初我以为 android 下修改系统时间是一件很容易的事情，但是后来，我发现我错了。为了系统安全起见，一般情况下，android 是禁止应用程序修改系统时间的。不过，通过应用程序来修改系统时间也不是不可能的。如果你对此有兴趣的话，请点击阅读：《Android 中如何修改系统时间》

知识点 2：Android Dialog 之 showDialog(***int id***)

关于 android 下 Dialog 创建以及使用等，以下文章讲的非常详细，建议点击阅读：《android 之 Dialog》

3.6 Rating Bar

RatingBar 是基于 SeekBar 和 ProgressBar 的扩展，用星型来显示等级评定。用户可以通过触屏点击或者轨迹球左右移动来进行星型等级评定。RatingBar 有三种风格:

- ratingBarStyle 默认风格
- ratingBarStyleSmall 小风格
- ratingBarStyleIndicator 大风格

其中，默认风格的 RatingBar 是我们通常使用的可以交互的，而后面两种不能进行进行交互，只能作为指示牌。设置 RatingBar 样式的方法是在 xml 布局文件中 RatingBar 控件内设置 style：

```
style="?android:attr/ratingBarStyle"
style="?android:attr/ratingBarStyleSmall"
style="?android:attr/ratingBarStyleIndicator"
```

下面我们以 RatingBar 为例，简单介绍如何创建三种不同样式的 RatingBar。

1、 ratingBarStyle 默认风格

```
<RatingBar android:id="@+id/ratingbar1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="3"
    android:rating="2.5"
    android:stepSize="0.5" />
```

注：android:numStars="3" 属性用来设置星星总个数，必须是一个整形值，例如 100 。

android:rating="2.5" 属性用来设置初始星星评分星星数目，必须是浮点类型，例如 1.2。

android:stepSize="0.5" 属性用来设置步长，即一次增加或者减少的星星数目是这个数字的整数倍。必须是浮点类型，例如 0.5。

2、ratingBarStyleSmall 小风格

```
<RatingBar android:id="@+id/small_ratingbar"
    style="?android:attr/ratingBarStyleSmall"
    android:layout_marginLeft="5dip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical" />
```

3、ratingBarStyleIndicator 大风格

```
<RatingBar android:id="@+id/indicator_ratingbar"
    style="?android:attr/ratingBarStyleIndicator"
    android:layout_marginLeft="5dip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical" />
```

RatingBar 主要通过设置监听器 OnRatingBarChangeListener 来响应用户点击的。具体使用方法如下：

```
// 通过 findViewById 方法获得 RatingBar 对象
RatingBar mRatingBar = (RatingBar) findViewById(R.id.ratingbar);
mRatingBar.setOnRatingBarChangeListener(mRatingBarChangeListener);

//创建 RatingBar 监听器
RatingBar.OnRatingBarChangeListener mRatingBarChangeListener = new
RatingBar.OnRatingBarChangeListener() {

        public void onRatingChanged(RatingBar ratingBar, float rating,
                        boolean fromUser) {
                // TODO 执行用户点击 RatingBar 后的响应动作
                final int mNumStars = ratingBar.getNumStars();  //获取 RatingBar 总星星数
                float mrating = rating;         //用户点击评定后的评分（高亮的星星个数）
                if(fromUser){
                        // TODO
                }
                else{
                        // TODO
                }
        }
 };
```

注：　创建 RatingBar 监听器时，需要注意的是，3 个参数以及它们对应的含义如下：

ratingBar:　由于多个 RatingBar 可以同时指定同一个 RatingBar 监听器 。该参数就是当前触发 RatingBar 监听器的那一个 RatingBar 对象。

rating: 当前评级分数。取值范围从 0 到 RatingBar 的总星星数。

fromTouch: 如果触发监听器的是来自用户触屏点击或轨迹球左右移动，则为 true。

下面我们进行实例代码解析：

res-layout-ratingbar_1.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:paddingLeft="10dip"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

     <!-- 一个星星数为 3，评分 2.5 个星星的默认 RatingBar -->
   <RatingBar android:id="@+id/ratingbar1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:numStars="3"
      android:rating="2.5"
      android:stepSize="0.5" />

     <!-- 一个星星数为 5，评分 2.5 个星星的默认 RatingBar 默认步长 0.5-->
   <RatingBar android:id="@+id/ratingbar2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:numStars="5"
        android:rating="2.25"
        android:stepSize="0.5" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip">

        <TextView android:id="@+id/rating"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

                <!-- 一个 stlye 为 ratingBarStyleSmall 的 RatingBar，不支持交互 -->
        <RatingBar android:id="@+id/small_ratingbar"
            style="?android:attr/ratingBarStyleSmall"
            android:layout_marginLeft="5dip"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical" />

    </LinearLayout>

                <!-- 一个 stlye 为 indicator_ratingbar 的 RatingBar，不支持交互 -->
        <RatingBar android:id="@+id/indicator_ratingbar"
            style="?android:attr/ratingBarStyleIndicator"
            android:layout_marginLeft="5dip"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical" />

</LinearLayout>
```

src-com.example.android.apis.view-RatingBar1.java

```
package com.example.android.apis.view;

import android.app.Activity;
import android.os.Bundle;
import android.widget.RatingBar;
import android.widget.TextView;

import com.example.android.apis.R;

/**
 * 演示如何使用 rating bar
 */
public class RatingBar1 extends Activity implements RatingBar.OnRatingBarChangeListener {
    RatingBar mSmallRatingBar;
    RatingBar mIndicatorRatingBar;
    TextView mRatingText;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.ratingbar_1);

        // 通过 findViewById 方法获得一个 TextView 对象
        mRatingText = (TextView) findViewById(R.id.rating);

        // 通过 findViewById 方法获得两个用作指示牌的 RatingBar 对象
        mIndicatorRatingBar = (RatingBar) findViewById(R.id.indicator_ratingbar);
        mSmallRatingBar = (RatingBar) findViewById(R.id.small_ratingbar);

        // 通过 findViewById 方法获得两个可以交互的 RatingBar 对象，并设置监听器
        ((RatingBar)findViewById(R.id.ratingbar1)).setOnRatingBarChangeListener(this);
        ((RatingBar)findViewById(R.id.ratingbar2)).setOnRatingBarChangeListener(this);
    }

    public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromTouch) {
        final int numStars = ratingBar.getNumStars();
        //显示当前星星总数多少，被选中高亮的星星数多少
        mRatingText.setText(
                getString(R.string.ratingbar_rating) + " " + rating + "/" + numStars);

        //当用户点击上面可以交互的 RatingBar，触发监听器，在这里更新两个作为指示牌
作用的 RatingBar 对象的三个参数
        //更新两个作为指示牌作用的 RatingBar 对象的星星总数
        if (mIndicatorRatingBar.getNumStars() != numStars) {
            mIndicatorRatingBar.setNumStars(numStars);
            mSmallRatingBar.setNumStars(numStars);
        }

        //更新两个作为指示牌作用的 RatingBar 对象的星星数（被选中高亮）
        if (mIndicatorRatingBar.getRating() != rating) {
            mIndicatorRatingBar.setRating(rating);
            mSmallRatingBar.setRating(rating);
        }

        //更新两个作为指示牌作用的 RatingBar 对象的步长值
        final float ratingBarStepSize = ratingBar.getStepSize();
        if (mIndicatorRatingBar.getStepSize() != ratingBarStepSize) {
            mIndicatorRatingBar.setStepSize(ratingBarStepSize);
            mSmallRatingBar.setStepSize(ratingBarStepSize);
        }
    }

}
```

知识点 1：android:layout_marginLeft="5dip"

该属性表示该空间离父控件的左边距离为 5 个 dip。其他 android:layout_marginRight 等属性的含义与之类似。 关于 dip 等 android 单位的详细介绍，请点击阅读：《Android 系统

Dimension 和间距参数详解》和《Android 单位区别与转化》

知识点 2：android:layout_gravity="center_vertical"

该属性表示相对于该控件的父控件而言，该控件在父控件中的什么位置。还有一个属性 android:gravity 则表示对于该控件而言，控件内部的文本显示在该控件的什么位置。两个属性容易混淆，切记进行区分。关于两个属性取值含义以及两者的区别等更详细的信息，请点击阅读：《android:layout_gravity 和 android:gravity 的区别》

知识点 3：如果默认的 RatingBar 不能满足您的需求，您不想用星星，而想用其他的自定义图片的话，请点击阅读：《 给 RatingBar 改图片》

3.7 Seek Bar

SeekBar 是基于 ProgressBar 的扩展，可以理解为添加了滑动条的 ProgressBar。用户可以左右移动滑动条或者左右移动轨迹球来设置当前的进度值。最好不要在 SeekBar 左边或者右边放置一个可以聚焦的控件。

SeekBar 控件最经典的应用是在播放器中用于显示/改变播放进度的进度条。下面是一个简单的 SeekBar 控件：

```
<SeekBar android:id="@+id/seek"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="100"
    android:progress="50"
    android:secondaryProgress="75" />
```

注：android:max="100" 属性表示该 SeekBar 的最大值为 100，即取值范围为 0~100
android:progress="50" 属性表示当前初始的进度值为 50
android:secondaryProgress="75" 属性表示当前初始的第二进度值为 75
在播放器的进度条中， android:progress 常用来表示当前播放的进度，而 android:secondaryProgress 则常用来表示当前音频/视频文件缓冲的进度。

SeekBar 主要通过设置监听器 OnSeekBarChangeListener 来响应用户点击的。具体使用方法如下：

```
// 通过 findViewById 方法获得 RatingBar 对象
SeekBar mSeekBar = (SeekBar) findViewById(R.id.seekbar);

 //创建 SeekBar 监听器
mSeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

            public void onStopTrackingTouch(SeekBar seekBar) {
                // TODO 用户停止对 seekbar 进行触屏操作时触发响应

            }

            public void onStartTrackingTouch(SeekBar seekBar) {
                // TODO 用户开始对 seekbar 进行触屏操作时触发响应

            }

            public void onProgressChanged(SeekBar seekBar, int progress,
                    boolean fromUser) {
                // TODO 用户移动滑动条，导致进度值改变时，触发响应

            }
        });
```

注：　创建 SeekBar 监听器时，需要注意的是，3 个参数以及它们对应的含义如下：

seekBar:　由于多个 SeekBar 可以同时指定同一个 SeekBar 监听器 。该参数就是当前触发 SeekBar 监听器的那一个 SeekBar 对象。

progress: 当前进度值。取值范围从 0 到 android:max 属性所设定的最大值。

fromUser: 如果触发监听器的是来自用户触屏点击或轨迹球左右移动，则为 true。

下面我们进行实例代码解析：

res-layout-seekbar_1.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

     <!-- 一个 SeekBar 对象-->
     <!-- android:max="100" 表示 seekbar 最大值是 100，即取值范围 0~100-->
     <!-- android:progress="50" 表示 seekbar 当前进度取值是 50-->
     <!-- android:secondaryProgress="75"  表示 seekbar 当前第二进度取值是 75-->
    <SeekBar android:id="@+id/seek"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:max="100"
      android:progress="50"
      android:secondaryProgress="75" />

     <!-- 一个 TextView 对象，用于显示当前的 progress 进度值 -->
    <TextView android:id="@+id/progress"
     android:layout_width="match_parent"
      android:layout_height="wrap_content" />

     <!-- 一个 TextView 对象，用于显示用户当前是否在对 Seek Bar 进行触屏操作 -->
    <TextView android:id="@+id/tracking"
     android:layout_width="match_parent"
      android:layout_height="wrap_content" />
</LinearLayout>
```

src-com.example.android.apis.view-SeekBar1.java

```
package com.example.android.apis.view;

import android.app.Activity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;

import com.example.android.apis.R;


/**
 * 演示如何使用 seek bar
 */
```

```
public class SeekBar1 extends Activity implements SeekBar.OnSeekBarChangeListener {

    SeekBar mSeekBar;
    TextView mProgressText;
    TextView mTrackingText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.seekbar_1);

        // 通过 findViewById 方法获得一个 SeekBar 对象和两个用于提示作用的 TextView
        mSeekBar = (SeekBar)findViewById(R.id.seek);
        mSeekBar.setOnSeekBarChangeListener(this); //设置 Seek Bar 监听器
        mProgressText = (TextView)findViewById(R.id.progress);
        mTrackingText = (TextView)findViewById(R.id.tracking);
    }

    //当用户用手移动滑动条，改变进度值时，触发该响应。progress 表示当前进度值。
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromTouch) {
        mProgressText.setText(progress + " " +
                getString(R.string.seekbar_from_touch) + "=" + fromTouch);
    }

    //用户开始对 SeekBar 进行触屏操作时触发响应
    public void onStartTrackingTouch(SeekBar seekBar) {
        mTrackingText.setText(getString(R.string.seekbar_tracking_on));
    }

    //用户停止对 SeekBar 进行触屏操作时触发响应
    public void onStopTrackingTouch(SeekBar seekBar) {
        mTrackingText.setText(getString(R.string.seekbar_tracking_off));
    }
}
```

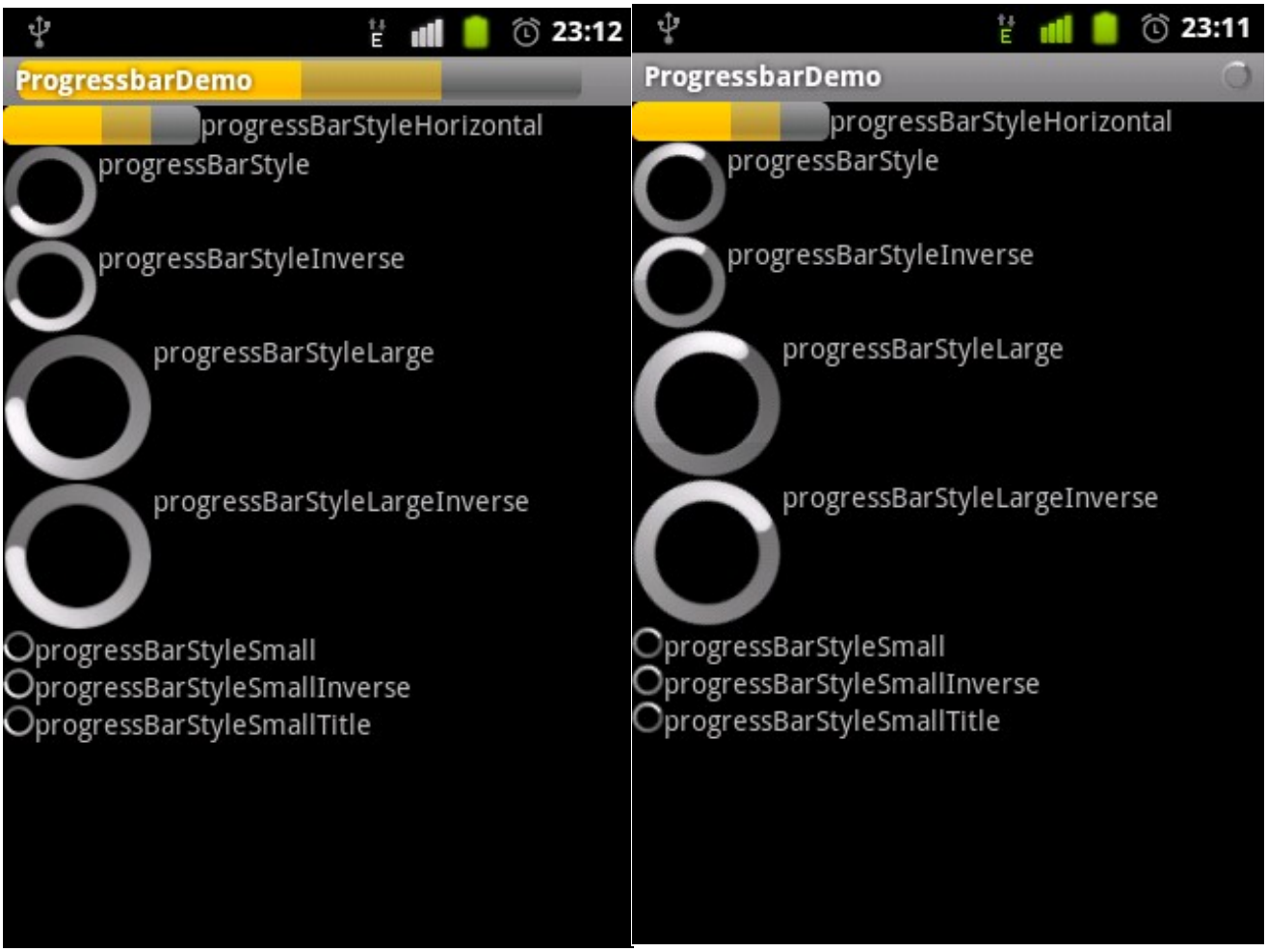
知识点 1：关于如何自定义 SeekBar 的颜色，大小，图片等进阶知识，请点击阅读《SeekBar 自定义(颜色,大小,图片)》

3.8 Progress Bar

进度条 ProgressBar 可以用来显示某项操作（比如下载文件）的当前进度。ProgressBar 主要有两种模式：可以随时确定当前进度值的模式（progressBarStyleHorizontal 风格），无法确定当前进度值的模式（其他 7 种风格）。另外，ProgressBar 还有 8 种系统内置的风格：

- progressBarStyle 默认风格
- progressBarStyleHorizontal 水平进度条风格
- progressBarStyleInverse 反色中等大小风格
- progressBarStyleLarge 超大风格
- progressBarStyleLargeInverse 反色超大风格
- progressBarStyleSmall 超小风格
- progressBarStyleSmallInverse 反色超小风格
- progressBarStyleSmallTitle 标题小风格

这些风格的 ProgressBar 预览效果如下所示：



以上几种风格的设置方法是在 xml 布局文件中 ProgressBar 控件内设置 style（默认风格 progressBarStyle 可以不标明）：

```
style="?android:attr/progressBarStyle"
style="?android:attr/progressBarStyleHorizontal"
style="?android:attr/progressBarStyleInverse"
style="?android:attr/progressBarStyleLarge"
style="?android:attr/progressBarStyleLargeInverse"
style="?android:attr/progressBarStyleSmall"
style="?android:attr/progressBarStyleSmallInverse"
style="?android:attr/progressBarStyleSmallTitle"
```

ProgressBar 不仅可以放在主界面中，而且还能放在标题栏中，甚至还可以放在对话框中。这里我们为了便于理解，按照 ProgressBar 的模式和用途分成六种情况进行介绍。

1、主界面中的水平进度条

```
<ProgressBar android:id="@+android:id/progressBarStyleHorizontal"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="120dip"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="50"
        android:secondaryProgress="75" />
```

注：android:max="100" 属性表示该 ProgressBar 的最大值为 100，即取值范围为 0~100
android:progress="50" 属性表示当前初始的进度值为 50
android:secondaryProgress="75" 属性表示当前初始的第二进度值为 75

2、主界面中的环形进度条

```
<ProgressBar android:id="@+android:id/progressBarStyle"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

注：style 可以取值除了 progressBarStyleHorizontal 之外的另外 7 个风格取值。

3、标题栏中的水平进度条

```
//请求窗口特色风格，这里设置成明确的进度风格
requestWindowFeature(Window.FEATURE_PROGRESS);
setContentView(R.layout.main);

//设置标题栏上 ProgressBar 的主要进度值和第二进度值
//标题栏上 ProgressBar 的取值范围为 0~10000
setProgress(5000);
setSecondaryProgress(7500);

//显示标题上的进度条(确定进度值)
setProgressBarVisibility(true);
```

注：1、requestWindowFeature(Window.FEATURE_PROGRESS);一句必须放在 setContentView(R.layout.main);之前，否则程序报错。
2、setProgressBarVisibility(true);用来设置进度条是否可见。setProgress(5000); 和 setSecondaryProgress(7500); 分别用来设置进度条的主要进度值和第二进度值。 这几句必须放在 setContentView(R.layout.main);之后，否则程序报错。

4、标题栏中的环形进度条

```
//请求窗口特色风格，这里设置成不明确的进度风格
```

```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
setContentView(R.layout.main);

//显示标题上的进度条(不确定进度值)
 setProgressBarIndeterminateVisibility(true);
```

注：1、 requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS); 一句必须放在 setContentView(R.layout.main);之前，否则程序报错。

2、 setProgressBarIndeterminateVisibility(true);用来设置进度条是否可见。这一句必须放在 setContentView(R.layout.main);之后，否则程序报错。

5、进度对话框中的水平进度条

```
 //创建 ProgressDialog
final ProgressDialog mProgressDialog = new ProgressDialog(this);

 //设置 ProgressDialog 对象类型为环形
mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);

//设置 ProgressDialog 的图标
mProgressDialog.setIcon(R.drawable.icon);

//设置 ProgressDialog 的标题
mProgressDialog.setTitle("ProgressDialog");

//设置 ProgressDialog 的消息
mProgressDialog.setMessage("Loading...");

//设置 ProgressDialog 中进度条最大值 100，即取值范围为 0~100
mProgressDialog.setMax(100);

//设置 ProgressDialog 中进度条当前初始主要进度值为 50
mProgressDialog.setProgress(0);

//设置 ProgressDialog 中进度条当前初始第二进度值为 75
mProgressDialog.setSecondaryProgress(0);

//设置 ProgressDialog 的当前进度值是否不确定
mProgressDialog.setIndeterminate(false);

//设置 ProgressDialog 是否可以通过回退键取消
mProgressDialog.setCancelable(true);

//设置 ProgressDialog 的确定按钮以及监听器
mProgressDialog.setButton( DialogInterface.BUTTON_POSITIVE,"Ok", new
DialogInterface.OnClickListener(){

                public void onClick(DialogInterface dialog, int which) {
                        // TODO 执行 Button_OK 按钮点击后的响应动作
                        mProgressDialog.dismiss();
                }

});
```

```
        //设置 ProgressDialog 的取消按钮以及监听器
        mProgressDialog.setButton(DialogInterface.BUTTON_NEGATIVE,"Cancel", new
DialogInterface.OnClickListener(){

                        public void onClick(DialogInterface dialog, int which) {
                                // TODO 执行 Button_CANCEL 按钮点击后的响应动作
                                mProgressDialog.dismiss();
                        }

        });

        //显示 ProgressDialog
        mProgressDialog.show();
```

注：直接调用 setProgress 和 setSecondaryProgress 无法真正改变进度条的主进度值和第二进度值。关于如何修改进度值，请点击阅读：《android dialog ——ProgressDialog 进度条对话框详解》

6、进度对话框中的环形进度条

```
        //创建 ProgressDialog
        final ProgressDialog mProgressDialog = new ProgressDialog(this);

        //设置 ProgressDialog 对象类型为环形
        mProgressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);

        //设置 ProgressDialog 的图标
        mProgressDialog.setIcon(R.drawable.icon);

        //设置 ProgressDialog 的标题
        mProgressDialog.setTitle("ProgressDialog");

        //设置 ProgressDialog 的消息
        mProgressDialog.setMessage("Loading...");

        //设置 ProgressDialog 的当前进度值是否不确定
        mProgressDialog.setIndeterminate(true);

        //设置 ProgressDialog 是否可以通过回退键取消
        mProgressDialog.setCancelable(true);

        //设置 ProgressDialog 的确定按钮以及监听器
        mProgressDialog.setButton( DialogInterface.BUTTON_POSITIVE,"Ok", new
DialogInterface.OnClickListener(){

                        public void onClick(DialogInterface dialog, int which) {
                                // TODO 执行 Button_OK 按钮点击后的响应动作
                                mProgressDialog.dismiss();
                        }

        });

        //设置 ProgressDialog 的取消按钮以及监听器
```

```
    mProgressDialog.setButton(DialogInterface.BUTTON_NEGATIVE,"Cancel", new
DialogInterface.OnClickListener(){

                    public void onClick(DialogInterface dialog, int which) {
                        // TODO 执行 Button_CANCEL 按钮点击后的响应动作
                        mProgressDialog.dismiss();
                    }

     });

    //显示 ProgressDialog
     mProgressDialog.show();
```

下面我们进行实例代码解析：

1、 Incremental

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="progressbar_1_plus">+</string>
   <string name="progressbar_1_minus">-</string>
   <string name="progressbar_1_default_progress">Default progress:</string>
   <string name="progressbar_1_secondary_progress">Secondary progress:</string>
</resources>
```

res-layout-progressbar_1.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

      <!-- 一个 ProgressBar 对象，类型为：progressBarStyleHorizontal -->
    <ProgressBar android:id="@+id/progress_horizontal"
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="200dip"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="50"
        android:secondaryProgress="75" />

      <!-- 一个 TextView 对象，提示下面两个按钮改变的主要进度值-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/progressbar_1_default_progress" />

      <!-- 水平布局内放置两个按钮，分别用于增大和减小进度条的主要进度值 -->
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Button android:id="@+id/decrease"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/progressbar_1_minus" />

        <Button android:id="@+id/increase"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/progressbar_1_plus" />

    </LinearLayout>

      <!-- 一个 TextView 对象，提示下面两个按钮改变的第二进度值-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/progressbar_1_secondary_progress" />

      <!-- 水平布局内放置两个按钮，分别用于增大和减小进度条的第二进度值 -->
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
```

```
        <Button android:id="@+id/decrease_secondary"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/progressbar_1_minus" />

        <Button android:id="@+id/increase_secondary"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/progressbar_1_plus" />

    </LinearLayout>

</LinearLayout>
```

src-com.example.android.apis.view-ProgressBar1.java

```
package com.example.android.apis.view;

import com.example.android.apis.R;

import android.app.Activity;
import android.widget.Button;
import android.widget.ProgressBar;
import android.os.Bundle;
import android.view.View;
import android.view.Window;


/**
 * 演示如何在窗口标题栏中使用进度条。
 * 进度条将会一直显示，直到进度完成，此时进度条消失。
 */
public class ProgressBar1 extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 请求在标题栏中显示进度条
        requestWindowFeature(Window.FEATURE_PROGRESS);
        setContentView(R.layout.progressbar_1);

        //设置标题栏中的进度条可见
        setProgressBarVisibility(true);

        // 通过 findViewById 方法获得一个 ProgressBar 对象
        final ProgressBar progressHorizontal = (ProgressBar)
findViewById(R.id.progress_horizontal);

         //获取 ProgressBar 对象（取值范围 0~100）的主要进度值和第二进度值，经过转换后
设置到标题栏中的 ProgressBar 对象（取值范围 0~10000）。
        setProgress(progressHorizontal.getProgress() * 100);
        setSecondaryProgress(progressHorizontal.getSecondaryProgress() * 100);
```

```
        //该按钮用于增大 ProgressBar 的主要进度值，步进值为 1
        Button button = (Button) findViewById(R.id.increase);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                //progressHorizontal 主要进度值加 1
                progressHorizontal.incrementProgressBy(1);

                //标题栏中的进度条取值范围 0~10000，而 progressHorizontal 取值范围 0~100，
因此，获取 progressHorizontal 进度值，乘以 100，设置成标题栏上进度条的进度值。下同。
                setProgress(100 * progressHorizontal.getProgress());
            }
        });

        //该按钮用于减小 ProgressBar 的主要进度值，步进值为 1
        button = (Button) findViewById(R.id.decrease);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                //progressHorizontal 主要进度值减 1
                progressHorizontal.incrementProgressBy(-1);

                setProgress(100 * progressHorizontal.getProgress());
            }
        });

        //该按钮用于增大 ProgressBar 的第二进度值，步进值为 1
        button = (Button) findViewById(R.id.increase_secondary);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                //progressHorizontal 第二进度值加 1
                progressHorizontal.incrementSecondaryProgressBy(1);

                //标题栏中的进度条取值范围 0~10000，而 progressHorizontal 取值范围 0~100，
因此，获取 progressHorizontal 第二进度值，乘以 100，设置成标题栏上进度条的第二进度
值。下同。
                setSecondaryProgress(100 * progressHorizontal.getSecondaryProgress());
            }
        });

        //该按钮用于减小 ProgressBar 的第二进度值，步进值为 1
        button = (Button) findViewById(R.id.decrease_secondary);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                //progressHorizontal 第二进度值减 1
                progressHorizontal.incrementSecondaryProgressBy(-1);

                setSecondaryProgress(100 * progressHorizontal.getSecondaryProgress());
            }
        });

    }
}
```

2、 Smooth

res-layout-progressbar_2.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

     <!-- 一个 ProgressBar 对象，类型为：progressBarStyleLarge -->
   <ProgressBar android:id="@+android:id/progress_large"
      style="?android:attr/progressBarStyleLarge"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

     <!-- 一个 ProgressBar 对象，类型为默认类型：progressBarStyle -->
   <ProgressBar android:id="@+android:id/progress"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

     <!-- 一个 ProgressBar 对象，类型为默认类型：progressBarStyleSmall -->
   <ProgressBar android:id="@+android:id/progress_small"
      style="?android:attr/progressBarStyleSmall"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

     <!-- 一个 ProgressBar 对象，类型为默认类型：progress_small_title -->
   <ProgressBar android:id="@+android:id/progress_small_title"
      style="?android:attr/progressBarStyleSmallTitle"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</LinearLayout>
```

src-com.example.android.apis.view-ProgressBar2.java

```
package com.example.android.apis.view;

import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.Window;


/**
 * 演示如何在窗口标题栏中使用不确定进度的进度条。
 * 本实例演示了 3 中不同大小的环形进度条，
 */
public class ProgressBar2 extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 请求在标题栏中显示不确定进度类型的进度条
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);

        setContentView(R.layout.progressbar_2);

        // 设置标题栏中的进度条可见
        setProgressBarIndeterminateVisibility(true);
    }
}
```

3、Dialogs

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="progressbar_3_progress">Show Progress</string>
    <string name="progressbar_3_indeterminate">Show Indeterminate</string>
    <string name="progressbar_3_indeterminate_no_title">Show Indeterminate No
Title</string>
</resources>
```

res-layout-progressbar_3.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <!-- 一个 Button 对象，点击显示一个带有标题的 ProgressDialog -->
    <Button android:id="@+id/showIndeterminate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="@string/progressbar_3_indeterminate" />

    <!-- 一个 Button 对象，点击显示一个不带标题的 ProgressDialog -->
    <Button android:id="@+id/showIndeterminateNoTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/progressbar_3_indeterminate_no_title" />

</LinearLayout>
```

src-com.example.android.apis.view-ProgressBar3.java

```
package com.example.android.apis.view;

import com.example.android.apis.R;

import android.app.Activity;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

/**
 * 演示如何使用进度对话框
 */
public class ProgressBar3 extends Activity {

    ProgressDialog mDialog1;
    ProgressDialog mDialog2;

    private static final int DIALOG1_KEY = 0;
    private static final int DIALOG2_KEY = 1;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.progressbar_3);

        Button button = (Button) findViewById(R.id.showIndeterminate);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG1_KEY); //显示带有标题的进度对话框
            }
        });

        button = (Button) findViewById(R.id.showIndeterminateNoTitle);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(DIALOG2_KEY); //显示不带标题的进度对话框
            }
```

```
        });
    }

    @Override
    protected Dialog onCreateDialog(int id) {
        switch (id) {
            case DIALOG1_KEY: {
                ProgressDialog dialog = new ProgressDialog(this);  //创建 ProgressDialog
                dialog.setTitle("Indeterminate"); //设置标题
                dialog.setMessage("Please wait while loading..."); //设置主题信息
                dialog.setIndeterminate(true);  //设置进度条为不明确进度的类型（环形）
                dialog.setCancelable(true);  //设置窗口可以通过退回键取消
                return dialog;
            }
            case DIALOG2_KEY: {
                ProgressDialog dialog = new ProgressDialog(this); //创建 ProgressDialog
                dialog.setMessage("Please wait while loading...");  //设置主题信息
                dialog.setIndeterminate(true);  //设置进度条为不明确进度的类型（环形）
                dialog.setCancelable(true); //设置窗口可以通过退回键取消
                return dialog;
            }
        }
        return null;
    }
}
```

4、In Title Bar

res-value-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="progressbar_4_toggle">Toggle Indeterminate</string>
</resources>
```

res-layout-progressbar_4.xml

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


<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <!-- 一个 Button 对象，点击显示/隐藏标题栏上的进度条 -->
    <Button android:id="@+id/toggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/progressbar_4_toggle" />

</LinearLayout>
```

src-com.example.android.apis.view-ProgressBar4.java

```
package com.example.android.apis.view;

import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.Window;
import android.view.View;
import android.widget.Button;


/**
 * 演示如何在标题栏中使用不确定进度值的进度条
 */
public class ProgressBar4 extends Activity {
    private boolean mToggleIndeterminate = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 请求在标题栏中显示不确定进度类型的进度条
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.progressbar_4);

        // 设置标题栏中的进度条是否可见
        setProgressBarIndeterminateVisibility(mToggleIndeterminate);

        //点击一次，改变一次 mToggleIndeterminate 值
        Button button = (Button) findViewById(R.id.toggle);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                mToggleIndeterminate = !mToggleIndeterminate;
                setProgressBarIndeterminateVisibility(mToggleIndeterminate);
            }
        });
    }
}
```

知识点 1：关于 Progress 的自定义等知识，请点击阅读：《Android 自定义 ProgressDialog 的方法》，《自定义 ProgressBar》以及《多式样 ProgressBar》。

3.9 Radio Group

想想我们上学时候做的单项选择题，其中只有一个是正确答案。在做题的时候，我们只能选择一项。如果我们想在 Android 上设计一道单项选择题的话，可能就要用到 RadioGroup 了。RadioGroup 常常和 RadioButton 一起使用。由一个 RadioGroup 包含若干个 RadioButton，组成一个单项选择群组。我们在同一时间只能选中该组中的一个 RadioButton。

RadioGroup 的创建主要有两种方法：

1、在 xml 布局文件中

```
<RadioGroup
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
android:checkedButton="@+id/lunch"
android:id="@+id/menu">
<RadioButton
   android:text="@string/radio_group_1_breakfast"
   android:id="@+id/breakfast"
   />
<RadioButton
   android:text="@string/radio_group_1_lunch"
   android:id="@id/lunch" />
<RadioButton
   android:text="@string/radio_group_1_dinner"
   android:id="@+id/dinner" />
<RadioButton
   android:text="@string/radio_group_1_all"
   android:id="@+id/all" />
</RadioGroup>
```

注：上面演示的是在一个 RadioGroup 中包含四个 RadioButton 的情况，您可以根据自己实际需要来增加或者减少 RadioButton 的个数。

2、在 Java 代码中

```
// 通过 findViewById 方法获得一个 RadioGroup 对象
RadioGroup  mRadioGroup = (RadioGroup) findViewById(R.id.menu);

// 向 RadioGroup 中动态添加一个 RadioButton 对象
RadioButton newRadioButton = new RadioButton(this);
newRadioButton.setText(R.string.radio_group_snack);
newRadioButton.setId(R.id.snack);
LinearLayout.LayoutParams layoutParams = new RadioGroup.LayoutParams(
      RadioGroup.LayoutParams.WRAP_CONTENT,
      RadioGroup.LayoutParams.WRAP_CONTENT);
      mRadioGroup.addView(newRadioButton, 0, layoutParams);
```

注：这里我们首先创建了一个 RadioGroup 对象和一个 RadioButton 对象，然后通过 addView 函数将 RadioButton 添加到 RadioGroup 中。其中 addView 函数的第二个参数，表示将 RadioButton 添加到 RadioGroup 内部的什么位置（从 0 开始）。 RadioButton 通过 setText 和 setId 分别设定文本和资源 ID 号码。

当我们做单项选择题的时候，可以很快看到选择的是哪个选项。但是在 Android 中使用 RadioGroup 时，我们怎么通过程序来获得用户到底选择的是哪个选项呢？这就要用到我

们常用的方法（监听器）了。具体使用方法如下：

```
    // 通过 findViewById 方法获得一个 RadioGroup 对象
    RadioGroup  mRadioGroup = (RadioGroup) findViewById(R.id.menu);

    //当 RadioButton 状态发生改变时，触发监听器，执行下面的动作。当清除选择项
时,checkedId 为-1。
    mRadioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {

                public void onCheckedChanged(RadioGroup group, int checkedId) {
                    // TODO Auto-generated method stub
                    String choice = null;
                    //根据 checkedId 判断用户选择了哪一个选项，并执行相应的动作。
                    switch (checkedId) {
                        case R.id.breakfast:
                            choice = "breakfast";
                            break;
                        case R.id.lunch:
                            choice = "lunch";
                            break;
                        case R.id.dinner:
                            choice = "dinner";
                            break;
                        case R.id.all:
                            choice = "all";
                            break;
                        case R.id.snack:
                            choice = "snack";
                            break;
                        default:
                            break;
                    }
                }
            });
```

注：我们可以根据监听器中的 checkedId 来判断是哪一个选项被选择，并执行相应的动作。

下面我们进行实例代码解析：

res-values-string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="radio_group_1_breakfast">Breakfast</string>
    <string name="radio_group_1_lunch">Lunch</string>
    <string name="radio_group_1_dinner">Dinner</string>
    <string name="radio_group_1_all">All of them</string>
    <string name="radio_group_1_selection">You have selected: (none)</string>
    <string name="radio_group_1_clear">Clear</string>
</resources>
```

res-layout-radio_group_1.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
   <!-- 一个 RadioGroup（单项选择组）对象，内含 4 个 RadioButton（单项选择按钮）和
一个 TextView 对象 -->
    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:checkedButton="@+id/lunch"
        android:id="@+id/menu">
        <RadioButton
            android:text="@string/radio_group_1_breakfast"
            android:id="@+id/breakfast"
            />
        <RadioButton
            android:text="@string/radio_group_1_lunch"
            android:id="@id/lunch" />
        <RadioButton
            android:text="@string/radio_group_1_dinner"
            android:id="@+id/dinner" />
        <RadioButton
            android:text="@string/radio_group_1_all"
            android:id="@+id/all" />
        <TextView
            android:text="@string/radio_group_1_selection"
            android:id="@+id/choice" />
    </RadioGroup>

    <!-- 一个 Button 对象，用于清除 RadioGroup 的选择项 -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/radio_group_1_clear"
        android:id="@+id/clear" />
</LinearLayout>
```

src-com.example.android.apis.view-RadioGroup1.java

```
package com.example.android.apis.view;

import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.RadioGroup;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.LinearLayout;


public class RadioGroup1 extends Activity implements
RadioGroup.OnCheckedChangeListener,
        View.OnClickListener {

    private TextView mChoice;
    private RadioGroup mRadioGroup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.radio_group_1);

        // 通过 findViewById 方法获得一个 RadioGroup 对象
        mRadioGroup = (RadioGroup) findViewById(R.id.menu);

        // 向 RadioGroup 中动态添加一个 RadioButton 对象
        RadioButton newRadioButton = new RadioButton(this);
        newRadioButton.setText(R.string.radio_group_snack);
        newRadioButton.setId(R.id.snack);
        LinearLayout.LayoutParams layoutParams = new RadioGroup.LayoutParams(
                RadioGroup.LayoutParams.WRAP_CONTENT,
                RadioGroup.LayoutParams.WRAP_CONTENT);
        mRadioGroup.addView(newRadioButton, 0, layoutParams);

        //为 RadioGroup 添加监听器，当点击 RadioButton 时，会触发监听器， 执行
onCheckedChanged 中的动作
        mRadioGroup.setOnCheckedChangeListener(this);

        //在 TextView 控件上显示被选择项的提示信息
        String selection = getString(R.string.radio_group_selection);
        mChoice = (TextView) findViewById(R.id.choice);
        mChoice.setText(selection + mRadioGroup.getCheckedRadioButtonId());

        // 通过 findViewById 方法获得一个 Button 对象，点击该对象会清除 RadioGroup 中的
选择项
        Button clearButton = (Button) findViewById(R.id.clear);
        clearButton.setOnClickListener(this);
```

```
    }

    //当 RadioButton 状态发生改变时，触发监听器，执行下面的动作。当清除选择项
时,checkedId 为-1。
    public void onCheckedChanged(RadioGroup group, int checkedId) {
        String selection = getString(R.string.radio_group_selection);
        String none = getString(R.string.radio_group_none);
        String choice = null;
        //根据 checkedId 判断用户选择了哪一个选项，并执行相应的动作。
        switch (checkedId) {
        case R.id.breakfast:
                choice = "breakfast";
                break;
        case R.id.lunch:
                choice = "lunch";
                break;
        case R.id.dinner:
                choice = "dinner";
                break;
        case R.id.all:
                choice = "all";
                break;
        case R.id.snack:
                choice = "snack";
                break;
        default:
                break;
        }
        mChoice.setText(selection + choice +
                        (checkedId == View.NO_ID ? none : checkedId));
    }

    //点击 clearButton 时，清空所有 RadioButton 的选择状态
    public void onClick(View v) {
        mRadioGroup.clearCheck();
    }
}
```

3.10 Spinner

上一节我们讲到 RadioGroup 控件。大家试着想想，如果 RadioGroup 里面有很多选项，那是不是会占用大片区域呢？这种情况下，对于本来就不大的手机界面空间来说简直是一种浪费。这时，我们就要用到 Spinner 下拉列表控件了。

下面简单介绍怎么创建和使用一个 Spinner 控件。

首先，在 layout 布局文件文件中定义一个 Spinner 控件。

```
<Spinner
   android:id="@+id/spinner1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:drawSelectorOnTop="true"
    android:prompt="@string/spinner_1_color_prompt" />
```

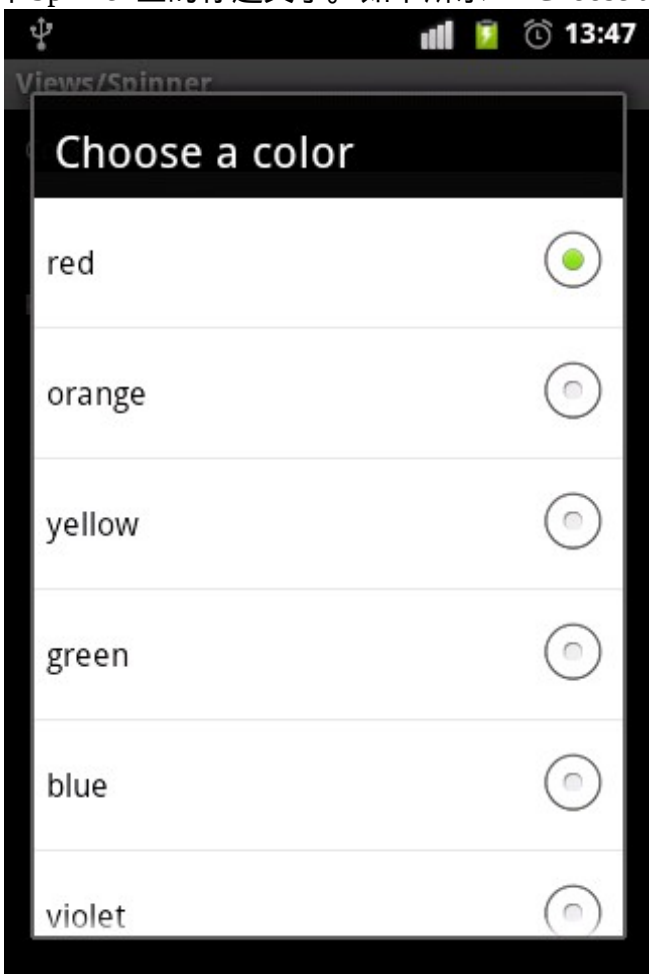
注：

1、android:drawSelectorOnTop="true"

有关该属性的相关知识请参考：http://blog.csdn.net/jincf2011/article/details/6598689

2、android:prompt="@string/spinner_1_color_prompt"

表示下拉列表控件 Spinner 上的标题文字。如下所示：“Choose a color”就是该属性定义的标题文字。



接着，定义 Spinner 控件的数据源数组。在 value 目录下创建一个 arrays.xml 文件。在里面定义一个数组，用作 Spinner 控件的数据源。格式如下：

```
<!-- Used in View/Spinner1.java -->
<string-array name="colors">
    <item>red</item>
    <item>orange</item>
    <item>yellow</item>
    <item>green</item>
    <item>blue</item>
    <item>violet</item>
    </string-array>
```

最后，在 Spinner1.java 文件中实例化 Spinner 控件。

```
    // 通过 findViewById 方法获得一个 Spinner 对象 s1，下同
    Spinner s1 = (Spinner) findViewById(R.id.spinner1);

    //根据颜色数组 R.array.colors 创建一个数组适配器
    //第二个参数 R.array.colors 为数组适配器数据源
    //第三个参数 是下拉列表中每个数据所占据的 View 的样式 。这里采用系统默认样式
android.R.layout.simple_spinner_item
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
            this, R.array.colors, android.R.layout.simple_spinner_item);

    //设置适配器下拉样式，这里是系统默认样式
android.R.layout.simple_spinner_dropdown_item
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

    //将 Spinner 控件绑定适配器 adapter
     s1.setAdapter(adapter);
```

这样，一个 Spinner 控件就成功创建好了。那么这个控件怎么用呢？和按钮等大多数控件一样，Spinner 控件也是通过创建一个监听器来监听用户的动作的。当用户选择其中一个选项时，便会触发响应。

```
    //设置监听器。当用户选择其中一个选项时，触发响应。
    s1.setOnItemSelectedListener(
                    //选择其中一个选项时，触发该响应
                    //parent:装载数据的视图列表
                    //view:列表中当前选中的 item 视图
                    //position:选中的 item 在 AdapterView 中的索引
                    //id:选中的 item 视图控件的 id 值
            new OnItemSelectedListener() {
              public void onItemSelected(
                      AdapterView<?> parent, View view, int position, long id) {
                showToast("Spinner1: position=" + position + " id=" + id);
              }

              //什么选项都没选择时，触发该响应
              public void onNothingSelected(AdapterView<?> parent) {
                showToast("Spinner1: unselected");
              }
                    });
```

下面我们进行实例代码解析：

res-values-arrays.xml

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

<resources>
    <!-- Used in View/Spinner1.java -->
    <string-array name="colors">
        <item>red</item>
        <item>orange</item>
        <item>yellow</item>
        <item>green</item>
        <item>blue</item>
        <item>violet</item>
    </string-array>

    <!-- Used in View/Spinner1.java -->
    <string-array name="planets">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
        <item>Pluto</item>
    </string-array>

</resources>
```

res-layout-spinner_1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="10dip" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/spinner_1_color" />

    <!-- 一个 Spinner 对象 -->
    <!-- android:drawSelectorOnTop 具体参考：
http://blog.csdn.net/jincf2011/article/details/6598689 -->
    <!-- android:prompt 下拉列表控件 Spinner 的标题文字 -->
    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:drawSelectorOnTop="true"
        android:prompt="@string/spinner_1_color_prompt" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:text="@string/spinner_1_planet" />

    <Spinner
        android:id="@+id/spinner2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true"
        android:prompt="@string/spinner_1_planet_prompt" />

</LinearLayout>
```

src-com.example.android.apis.view-Spinner1.java

```
package com.example.android.apis.view;

/**
 * 演示如何使用下拉列表控件 Spinner
 */
import com.example.android.apis.R;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
import android.widget.AdapterView.OnItemSelectedListener;


public class Spinner1 extends Activity {

    // 调用 Toast 控件来显示消息 msg，用来提示用户
    void showToast(CharSequence msg) {
        Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.spinner_1);

        // 通过 findViewById 方法获得一个 Spinner 对象 s1，下同
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);

        //根据颜色数组 R.array.colors 创建一个数组适配器
```

```
        //第二个参数 R.array.colors 为数组适配器数据源
        //第三个参数 是下拉列表中每个数据所占据的 View 的样式 。 这里采用系统默认样式
android.R.layout.simple_spinner_item
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
                this, R.array.colors, android.R.layout.simple_spinner_item);

        //设置适配器下拉样式，这里是系统默认样式
android.R.layout.simple_spinner_dropdown_item
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

        //将 Spinner 控件绑定适配器 adapter
        s1.setAdapter(adapter);

        //设置监听器。当用户选择其中一个选项时，触发响应。
        s1.setOnItemSelectedListener(
                        //选择其中一个选项时，触发该响应
                        //parent:装载数据的视图列表
                        //view:列表中当前选中的 item 视图
                        //position:选中的 item 在 AdapterView 中的索引
                        //id:选中的 item 视图控件的 id 值
                new OnItemSelectedListener() {
                    public void onItemSelected(
                            AdapterView<?> parent, View view, int position, long id) {
                        showToast("Spinner1: position=" + position + " id=" + id);
                    }

                    //什么选项都没选择时，触发该响应
                    public void onNothingSelected(AdapterView<?> parent) {
                        showToast("Spinner1: unselected");
                    }
                });

        // 通过 findViewById 方法获得一个 Spinner 对象 s2
        Spinner s2 = (Spinner) findViewById(R.id.spinner2);
        adapter = ArrayAdapter.createFromResource(this, R.array.planets,
                android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        s2.setAdapter(adapter);
        s2.setOnItemSelectedListener(
                new OnItemSelectedListener() {
                    public void onItemSelected(
                            AdapterView<?> parent, View view, int position, long id) {
                        showToast("Spinner2: position=" + position + " id=" + id);
                    }

                    public void onNothingSelected(AdapterView<?> parent) {
                        showToast("Spinner2: unselected");
                    }
                });
    }
}
```

3.11 Tabs

Tab 与 TabHost 应用很广泛。打开 android 手机的默认电话拨号程序，上面就是由“拨号”，“通话记录”，“通讯录”以及“收藏”四个选项卡组成的。

TabHost 有两种实现方式，一种是继承 TabActivity，另一种是自己定义 TabHost，不继承 TabActivity。APIDemo 中的三个实例都是第一种。想了解 TabHost 的第二种实现方式，请参考以下内容：

TabHost 两种实现方式 http://www.eoeandroid.com/thread-35836-1-1.html

下面我们逐一介绍 APIDemo 实例中 TabHost 的三种实现方法：

方法一、Content By Id

1、在一个 **FrameLayout** 内定义几个控件/布局，每个布局用作一个 Tab 的内容。

在这个实例中，**FrameLayout** 内定义三个不同背景颜色的 TextView 对象 view1，view2 以及 view3。

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

  <!-- 一个 TextView 对象，背景色为蓝色 -->
  <TextView
    android:id="@+id/view1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/blue"
    android:text="@string/tabs_1_tab_1" />

  <!-- 一个 TextView 对象，背景色为红色 -->
  <TextView
    android:id="@+id/view2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/red"
    android:text="@string/tabs_1_tab_2" />

  <!-- 一个 TextView 对象，背景色为绿色 -->
  <TextView
    android:id="@+id/view3"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/green"
    android:text="@string/tabs_1_tab_3" />

</FrameLayout>
```

2、新建一个 Activity,继承自 TabActivity

```
public class Tabs1 extends TabActivity {
```

3、通过 getTabHost()函数，获得 TabActivity 内置的 TabHost 对象。将样式 R.layout.tabs1 解压出来，并应用于 TabHost 对象。最后根据三个 TextView 控件创建了三个 Tab 选项卡。

```
//获取 TabActivity 内置的 TabHost 对象
TabHost tabHost = getTabHost();

//将样式 R.layout.tabs1 解压出来，并应用于 TabHost 对象
LayoutInflater.from(this).inflate(R.layout.tabs1, tabHost.getTabContentView(), true);

//新建选项卡
//tabHost.newTabSpec("tab1") 新建一个以“tab1”为标记的选项卡
//setIndicator("tab1")   设置选项卡标签标题为 tab1
//setContent(R.id.view1)) 设置选项卡内容为 R.id.view1
tabHost.addTab(tabHost.newTabSpec("tab1")
        .setIndicator("tab1")
        .setContent(R.id.view1));
tabHost.addTab(tabHost.newTabSpec("tab3")
        .setIndicator("tab2")
        .setContent(R.id.view2));
tabHost.addTab(tabHost.newTabSpec("tab3")
        .setIndicator("tab3")
            .setContent(R.id.view3));
```

方法二、Content By Factory

1、新建一个 Activity,继承自 TabActivity，并且继承 TabHost.TabContentFactory 接口。

```
public class Tabs2 extends TabActivity implements TabHost.TabContentFactory {
```

2、通过 getTabHost()函数，获得 TabActivity 内置的 TabHost 对象。和上面的例子比较，发现这里创建的三个选项卡使用的代码是：setContent(this))。这里的选项卡内容就得通过实现 TabHost.TabContentFactory 接口来创建了。

```
//获取 TabActivity 内置的 TabHost 对象
final TabHost tabHost = getTabHost();

//新建选项卡
//tabHost.newTabSpec("tab1") 新建一个以“tab1”为标记的选项卡
//setIndicator("tab1")   设置选项卡标签标题为 tab1 ，图标为 getResources().getDrawable(R.drawable.star_big_on)
//setContent(this))  由于该 TabActivity 继承了 TabHost.TabContentFactory 接口，因此，点击选项卡标签时，会调用
// public View createTabContent(String tag)来创建每个选项卡内容。该函数中的参数 tag 和 newTabSpec 中的参数是一样的。
tabHost.addTab(tabHost.newTabSpec("tab1")
        .setIndicator("tab1", getResources().getDrawable(R.drawable.star_big_on))
        .setContent(this));
tabHost.addTab(tabHost.newTabSpec("tab2")
        .setIndicator("tab2")
        .setContent(this));
tabHost.addTab(tabHost.newTabSpec("tab3")
        .setIndicator("tab3")
            .setContent(this));
```

3、实现TabHost.TabContentFactory接口。这样，每点击一个选项卡，该接口就会根据选项卡标记（tabHost.newTabSpec("tab1") 中的参数 "tab1" ）来创建对应的View。

```
    //每次点击选项卡标签时，根据选项卡 tag 标记来创建具体内容。
  /** {@inheritDoc} */
  public View createTabContent(String tag) {
    final TextView tv = new TextView(this);
    tv.setText("Content for tab with tag " + tag);
    return tv;
    }
```

方法三、 Content By Intent

1、新建一个 Activity,继承自 TabActivity。

```
public class Tabs3 extends TabActivity {
```

2、通过 getTabHost()函数，获得 TabActivity 内置的 TabHost 对象。和以上两个实例比较，发现这里创建的三个选项卡使用的代码是：setContent(**Intent**))。点击每个选项卡，便会根据 intent，跳转到对应的 Activity 界面。例如：点击 "tab1" 选项卡，便会跳转到 List1 界面。

```
      //获取 TabActivity 内置的 TabHost 对象
    final TabHost tabHost = getTabHost();

    //新建选项卡
    //tabHost.newTabSpec("tab1") 新建一个以“tab1”为标记的选项卡
    //setIndicator("tab1")   设置选项卡标签标题为 list
    //setContent(new Intent(this, List1.class))) 点击该选项卡时，通过 Intent，跳转到 List1 界面
    tabHost.addTab(tabHost.newTabSpec("tab1")
        .setIndicator("list")
        .setContent(new Intent(this, List1.class)));

    tabHost.addTab(tabHost.newTabSpec("tab2")
        .setIndicator("photo list")
        .setContent(new Intent(this, List8.class)));

    // 这个选项卡设置 Intent.FLAG_ACTIVITY_CLEAR_TOP 标记，是为了 每次点击该选项卡时，重新创建选项卡内容。
    tabHost.addTab(tabHost.newTabSpec("tab3")
        .setIndicator("destroy")
        .setContent(new Intent(this, Controls2.class)
          .addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)));
```

以上只是 TabHost 的一些基本用法，如果你想了解更多有关 TabHost 的知识，请点击参考以下帖子：《史上最全的 **Android** 的 **Tab** 与 **TabHost** 讲解》
http://www.eoeandroid.com/thread-1035-1-1.html

下面我们进行实例代码解析：

实例一、Content By Id

res-layout-tabs1.xml

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

    <!-- 一个 TextView 对象，背景色为蓝色 -->
    <TextView
        android:id="@+id/view1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/blue"
        android:text="@string/tabs_1_tab_1" />

    <!-- 一个 TextView 对象，背景色为红色 -->
    <TextView
        android:id="@+id/view2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/red"
        android:text="@string/tabs_1_tab_2" />

    <!-- 一个 TextView 对象，背景色为绿色 -->
    <TextView
        android:id="@+id/view3"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/green"
        android:text="@string/tabs_1_tab_3" />

</FrameLayout>
```

src-com.example.android.apis.view-Tabs1.java

```
package com.example.android.apis.view;

import android.app.TabActivity;
import android.os.Bundle;
import android.widget.TabHost;
import android.view.LayoutInflater;
import com.example.android.apis.R;

/**
 * 演示如何使用 TabHost
 *
 */
public class Tabs1 extends TabActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        //获取 TabActivity 内置的 TabHost 对象
        TabHost tabHost = getTabHost();

        //将样式 R.layout.tabs1 解压出来，并应用于 TabHost 对象
        LayoutInflater.from(this).inflate(R.layout.tabs1, tabHost.getTabContentView(), true);

        //新建选项卡
        //tabHost.newTabSpec("tab1") 新建一个以“tab1”为标记的选项卡
        //setIndicator("tab1")   设置选项卡标签标题为 tab1
        //setContent(R.id.view1)) 设置选项卡内容为 R.id.view1
        tabHost.addTab(tabHost.newTabSpec("tab1")
                .setIndicator("tab1")
                .setContent(R.id.view1));
        tabHost.addTab(tabHost.newTabSpec("tab3")
                .setIndicator("tab2")
                .setContent(R.id.view2));
        tabHost.addTab(tabHost.newTabSpec("tab3")
                .setIndicator("tab3")
                .setContent(R.id.view3));
    }
}
```

实例二、Content By Factory

src-com.example.android.apis.view-Tabs2.java

```
package com.example.android.apis.view;

import android.app.TabActivity;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.TextView;
import android.view.View;
import com.example.android.apis.R;

/**
 * 演示如何使用 TabHost
 *
 */
public class Tabs2 extends TabActivity implements TabHost.TabContentFactory {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //获取 TabActivity 内置的 TabHost 对象
        final TabHost tabHost = getTabHost();

        //新建选项卡
        //tabHost.newTabSpec("tab1") 新建一个以“tab1”为标记的选项卡
        //setIndicator("tab1")   设置选项卡标签标题为 tab1 ，图标为
getResources().getDrawable(R.drawable.star_big_on)
        //setContent(this))  由于该 TabActivity 继承了 TabHost.TabContentFactory 接口，因此，
```

```
点击选项卡标签时，会调用
        // public View createTabContent(String tag)来创建每个选项卡内容。该函数中的参数
tag 和 newTabSpec 中的参数是一样的。
        tabHost.addTab(tabHost.newTabSpec("tab1")
                .setIndicator("tab1", getResources().getDrawable(R.drawable.star_big_on))
                .setContent(this));
        tabHost.addTab(tabHost.newTabSpec("tab2")
                .setIndicator("tab2")
                .setContent(this));
        tabHost.addTab(tabHost.newTabSpec("tab3")
                .setIndicator("tab3")
                .setContent(this));
    }


    //每次点击选项卡标签时，根据选项卡 tag 标记来创建具体内容。
    /** {@inheritDoc} */
    public View createTabContent(String tag) {
        final TextView tv = new TextView(this);
        tv.setText("Content for tab with tag " + tag);
        return tv;
    }
}
```

实例三、Content By Intent
src-com.example.android.apis.view-Tabs3.java

```
package com.example.android.apis.view;

import android.app.TabActivity;
import android.os.Bundle;
import android.widget.TabHost;
import android.content.Intent;

/**
 * 演示如何使用 TabHost
 *
 */
public class Tabs3 extends TabActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //获取 TabActivity 内置的 TabHost 对象
        final TabHost tabHost = getTabHost();

        //新建选项卡
        //tabHost.newTabSpec("tab1") 新建一个以“tab1”为标记的选项卡
        //setIndicator("tab1")   设置选项卡标签标题为 list
        //setContent(new Intent(this, List1.class))) 点击该选项卡时，通过 Intent，跳转到 List1
```

```
界面
        tabHost.addTab(tabHost.newTabSpec("tab1")
                .setIndicator("list")
                .setContent(new Intent(this, List1.class)));

        tabHost.addTab(tabHost.newTabSpec("tab2")
                .setIndicator("photo list")
                .setContent(new Intent(this, List8.class)));

        // 这个选项卡设置 Intent.FLAG_ACTIVITY_CLEAR_TOP 标记，是为了 每次点击该
选项卡时，重新创建选项卡内容。
        tabHost.addTab(tabHost.newTabSpec("tab3")
                .setIndicator("destroy")
                .setContent(new Intent(this, Controls2.class)
                        .addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)));
    }
}
```