Android 可视化环境配置

JDK1.6　　下载网址：　http://java.sun.com/javase/downloads/index.jsp
Eclipse3.4　　下载网址：http://www.eclipse.org/downloads/ (下载 Eclipse IDE for Java Developers)
Android SDK1.5 下载网址：　http://developer.android.com

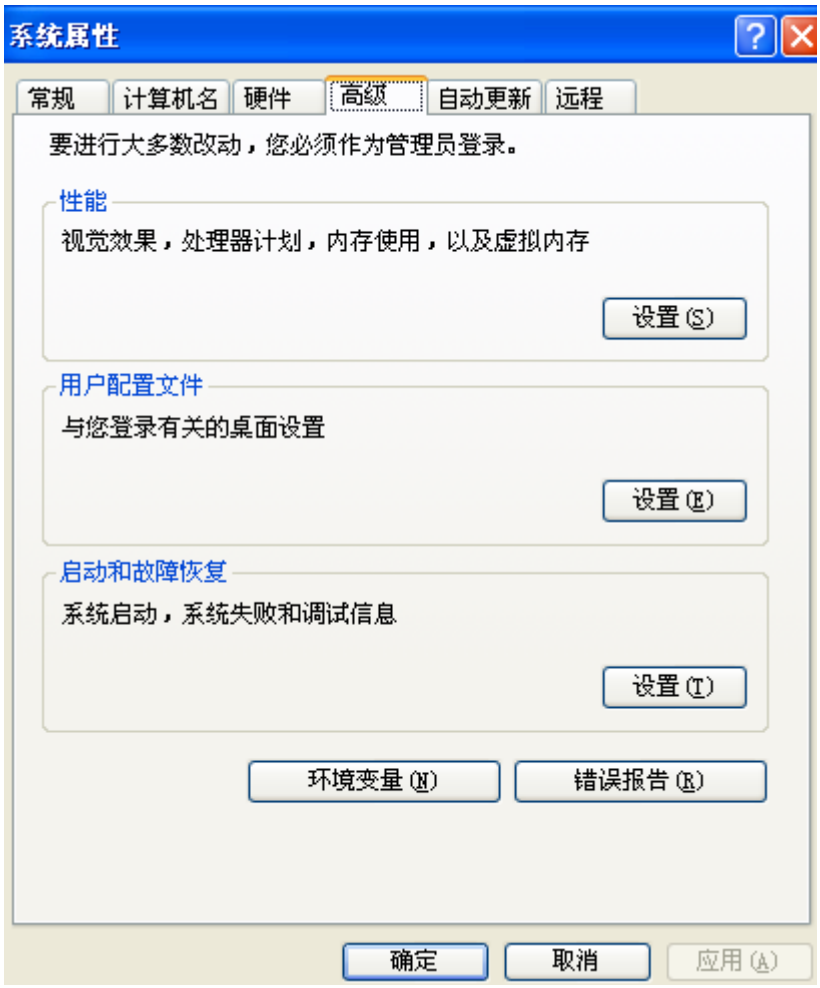
将此 3 个文件下载到 F：目录
并且解压：

并且安装：

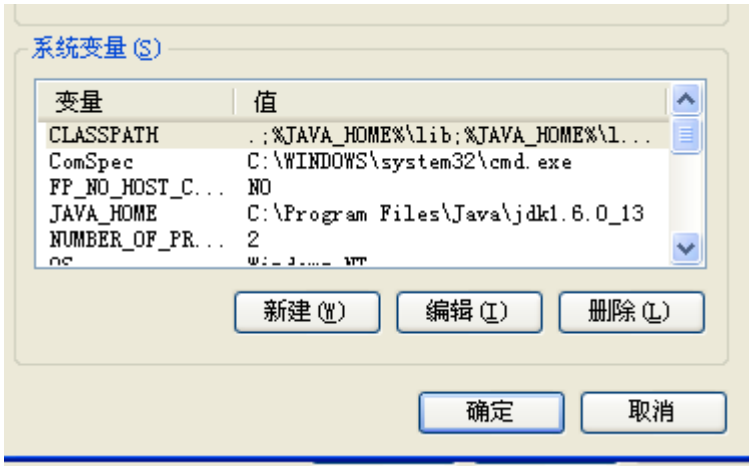
系统环境变量的配置
紧接着就是配置系统环境变量：

右击我的电脑选择属性：

将会出现如下：



点击环境变量：



并且在系统环境变量中新建以及编辑变量：

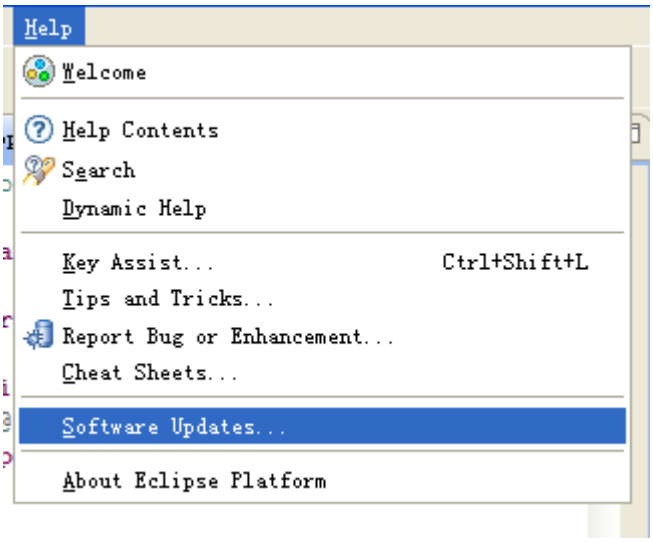
新建：
JAVA_HOME=”” 此处选择 jdk 的位置 安装默认为 C:\Program Files\Java\jdk1.6.0_13
CLASSPATH=.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar;%JAVA_HOME%\lib\dt.jar

编辑：
path 变量
jdk bin 目录的位置 默认为：C:\Program Files\Java\jdk1.6.0_13\bin
abdroid 的 tools 目录如：F:\android-sdk-windows-1.5_r2\android-sdk-windows-1.5_r2\tools

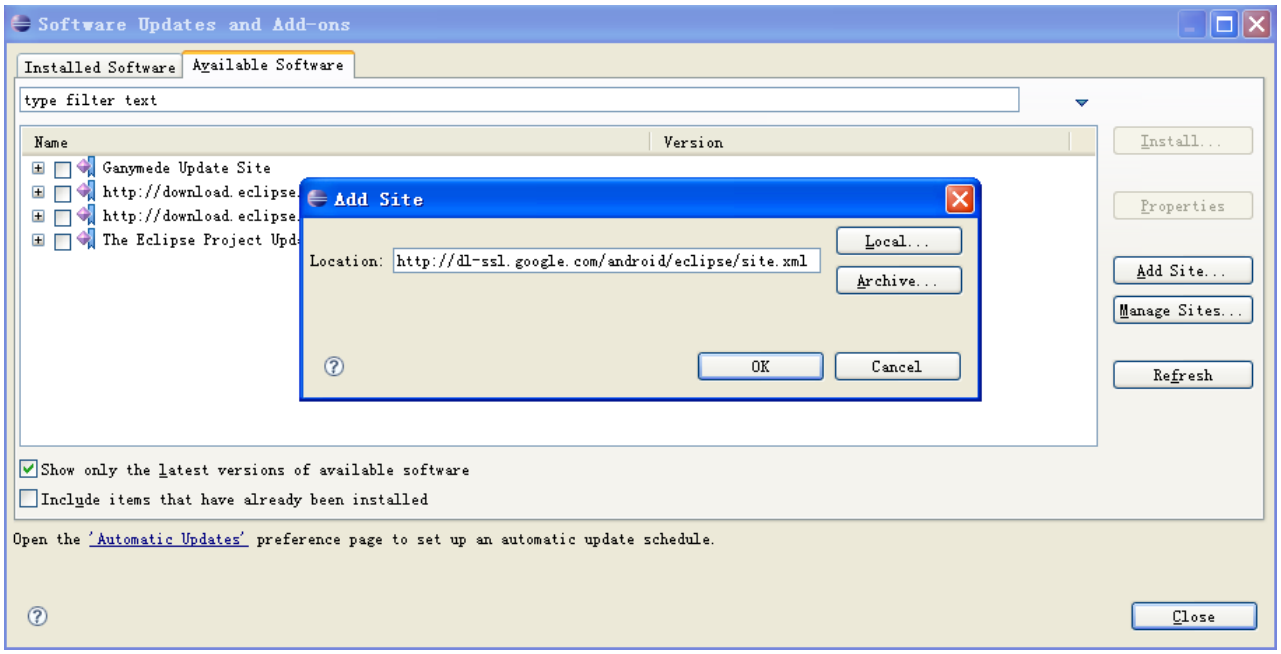
紧接着打开开始解压的

打开 eclipse

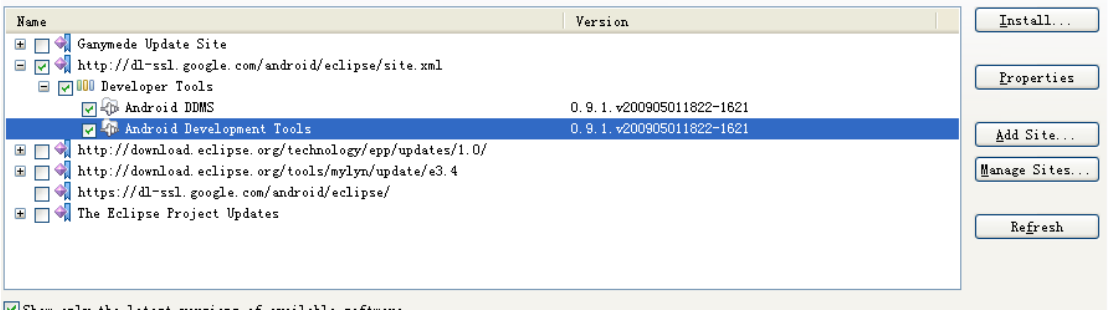
点击 help 选择更新软件



点击 Add Site 在 Location 里面键入地：http://dl-ssl.google.com/android/eclipse/site.xml



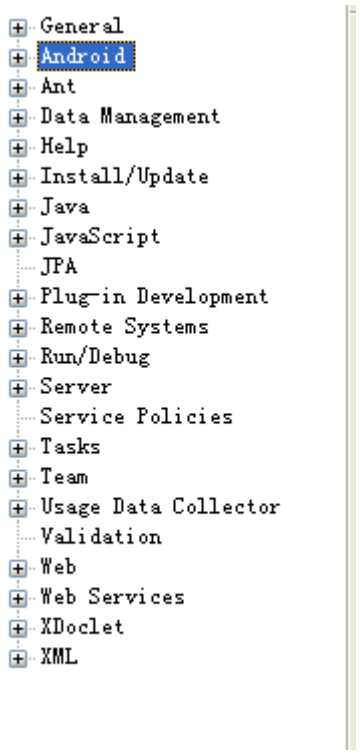
紧接着点击安装



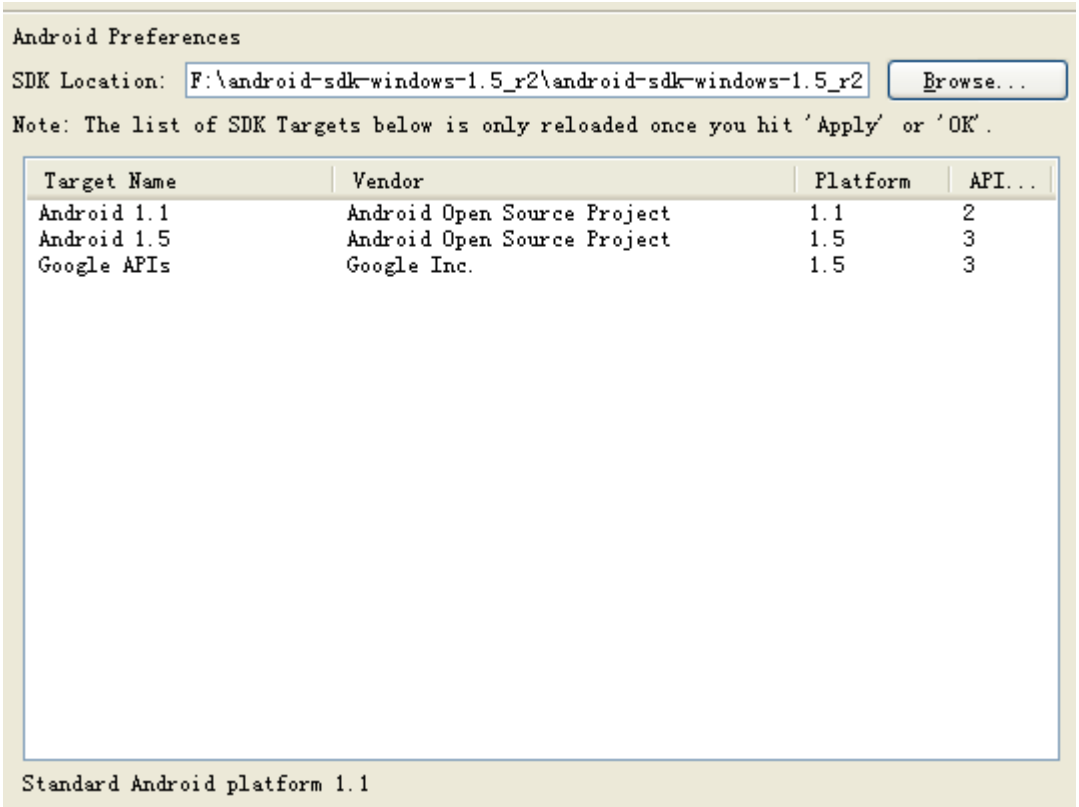
安装成功会提示重启 Eclipse

再次进入 Eclipse
点击 Window 选择 preferences

会发现在树列表中会多一个名为 Android 的节点：
选中此节点：



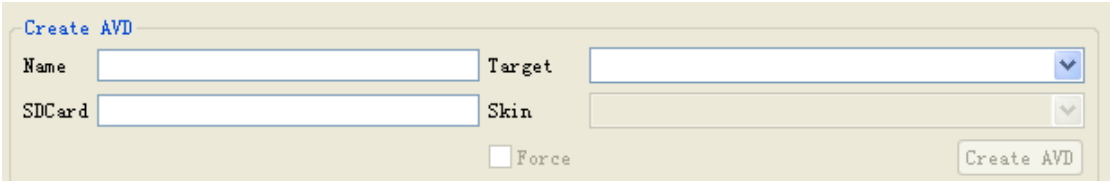
配置 SDK 的目录即开始解压的



配置完毕紧接着是点击 Apply 之后点击 OK （完毕）

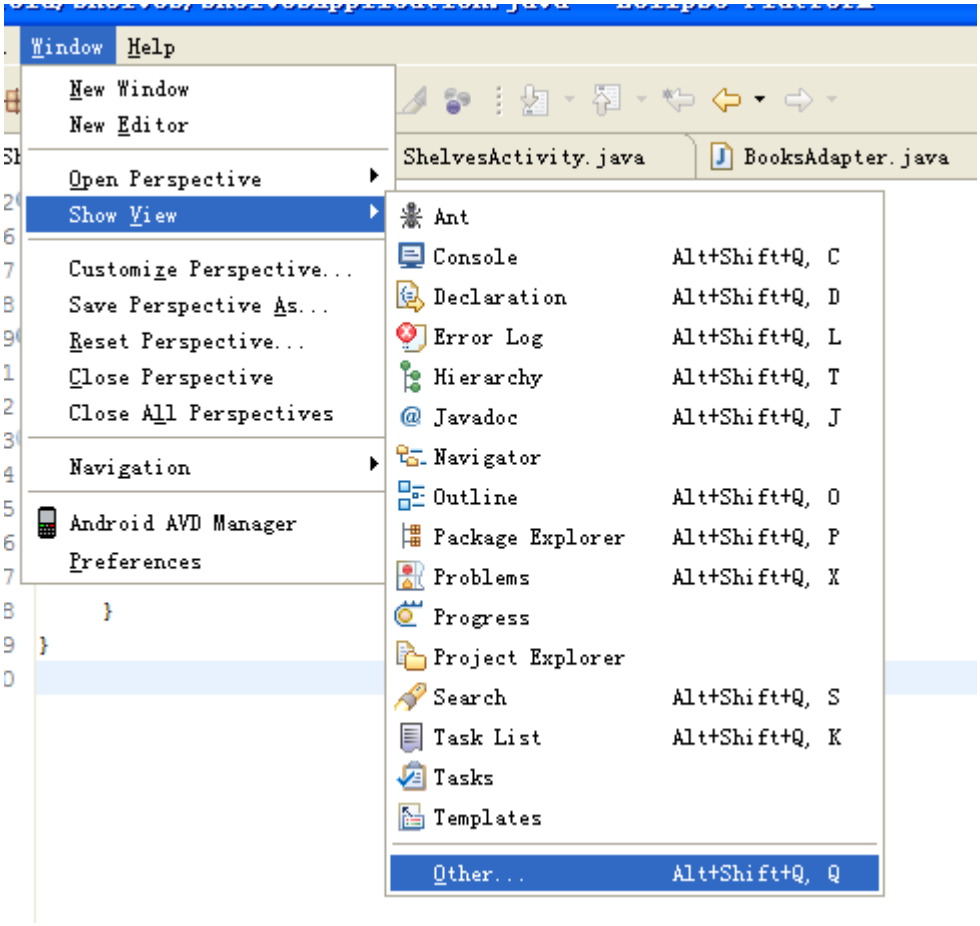
接着点击 Window 进入 AVD 管理

接着创建一个 AVD

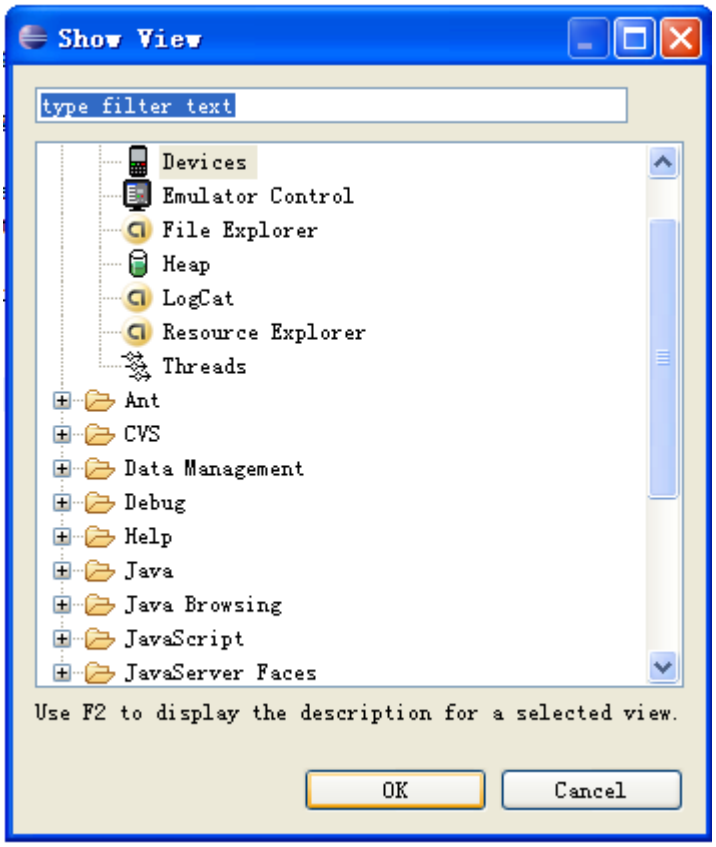


NAME:随便起一个名字 如 android1.5
Target:选项框 选择开始的 android1.5 即我们开始配置好的
SDCard:虚拟内存 填写 126M 注意 M 的大小写
Skin 模拟器的样式 这里就选择默认的
所有配置已经完毕

接着我们配置 Android 的辅助视图：
进入 Other

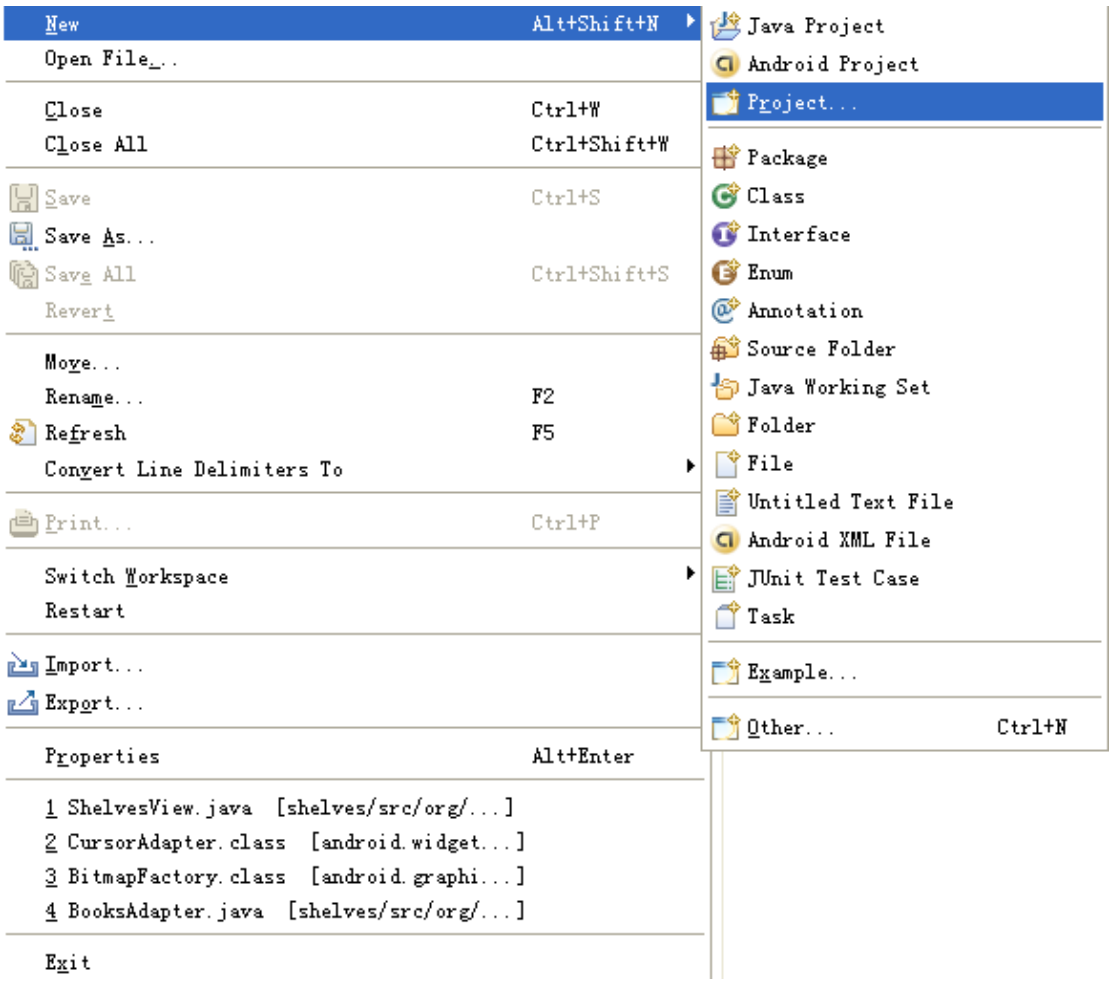


这里我们经常用到的为 LogCat, Devices, LogCat:
LogCat:　可以查看到模拟手机内部的文件信息　以及一系列出错信息
Devices:　显示模拟器运行过程
Emulator Control：　模拟发送短信信息 等一系列事情

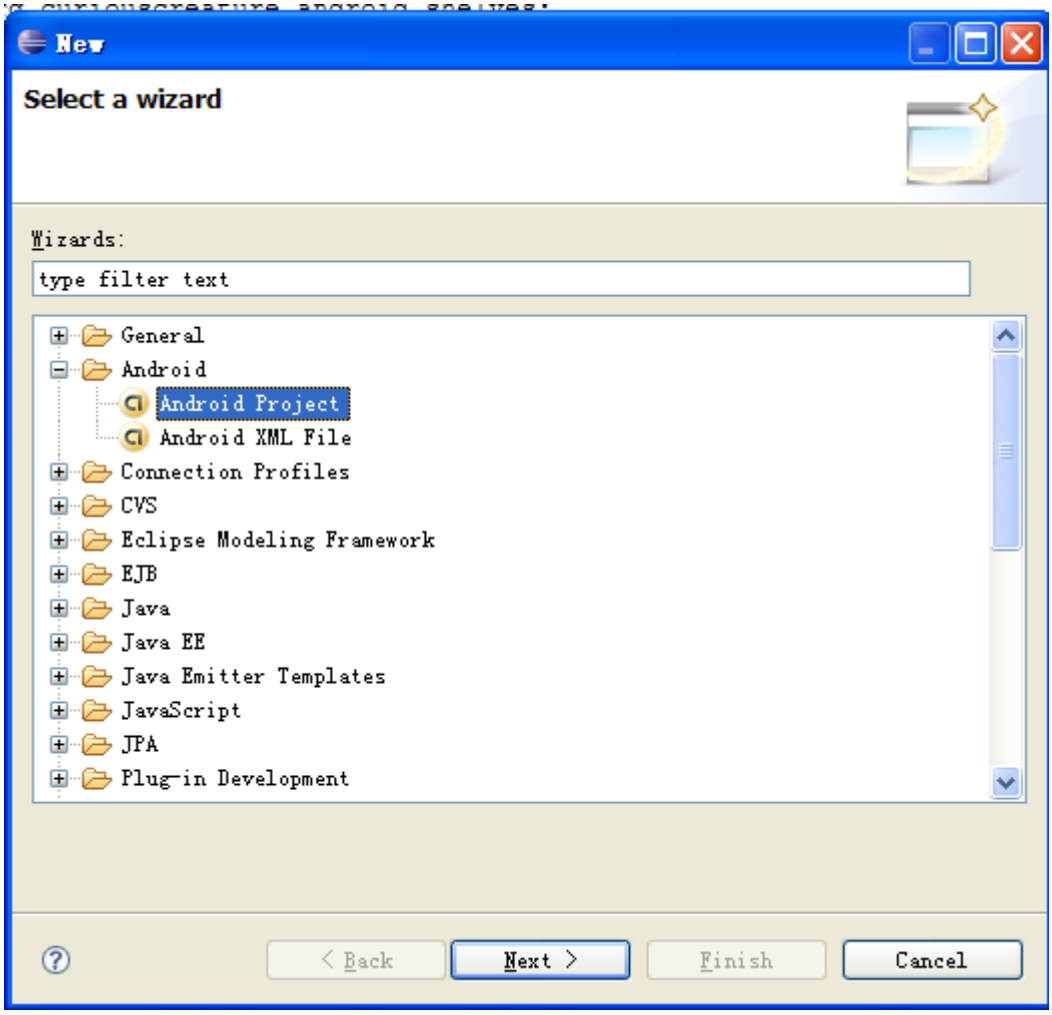


现在一系列配置都已经完毕 我们来做一个 Hello Android 的实例

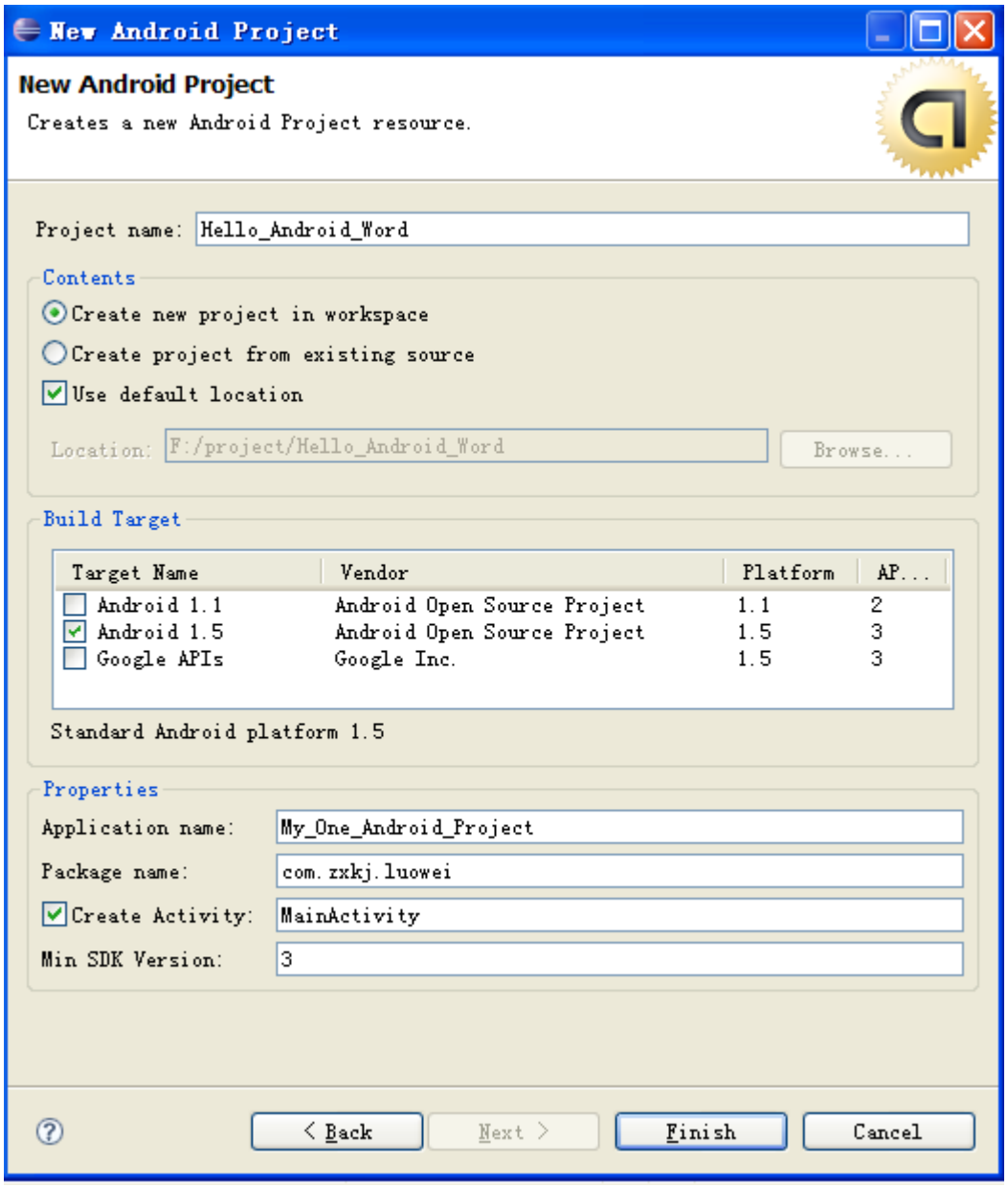
首先我们新建一个工程：



选择 Android project

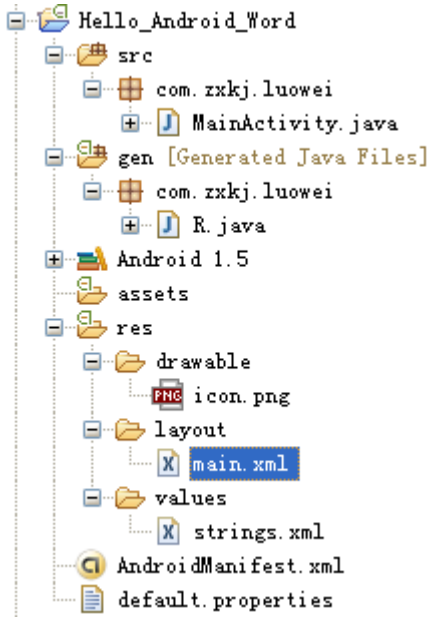

New
Select a wizard
Wizards:
type filter text
General
Android
Android Project
Android XML File
Connection Profiles
CVS
Eclipse Modeling Framework
EJB
Java
Java EE
Java Emitter Templates
JavaScript
JPA
Plug-in Development
< Back
Next >
Finish
Cancel

将会出现如下信息



Priject name:工程名字
Contents:单选框一个工程还是导入现有工程
Build Target:选择使用那一个 JDK
Application name:应用程序名字
Package name: 创建包
Create Activity:创建一个 Activity 如果你是 J2EE 程序员这个就相当于 struts 的 Action 类
Min SDK Version: JDK 版本

创建完毕：



目录介绍：

Src: java 源文件即我们写的 java 后缀名的文件代码 在里面有我们之前所填写创建的一个 MainActivity.java 文件

Gen:并没有创建 gen 这个目录 但是为什么出现此目录呢？没错这个是 Android 给我们自动生成的一个目录 并且还在次目录下创建了一个 R 文件（此 R 文件后面会讲到）

Android 1.5：如果你是 java 程序员 就应该很熟悉 这个就书库文件 即 Android 的核心文件

Assets: 没有用到过

Res.:放置资源文件的目录

Res.drawable:一般用来存储相关应用的图片以及 mp3 播放文件等

Res.layout:用来存储布局信息 如果你是 j2ee 程序员那么此目录下的文件相当于 jsp 文件即 html 文件，只是 Android 是以 xml 方式进行布局的

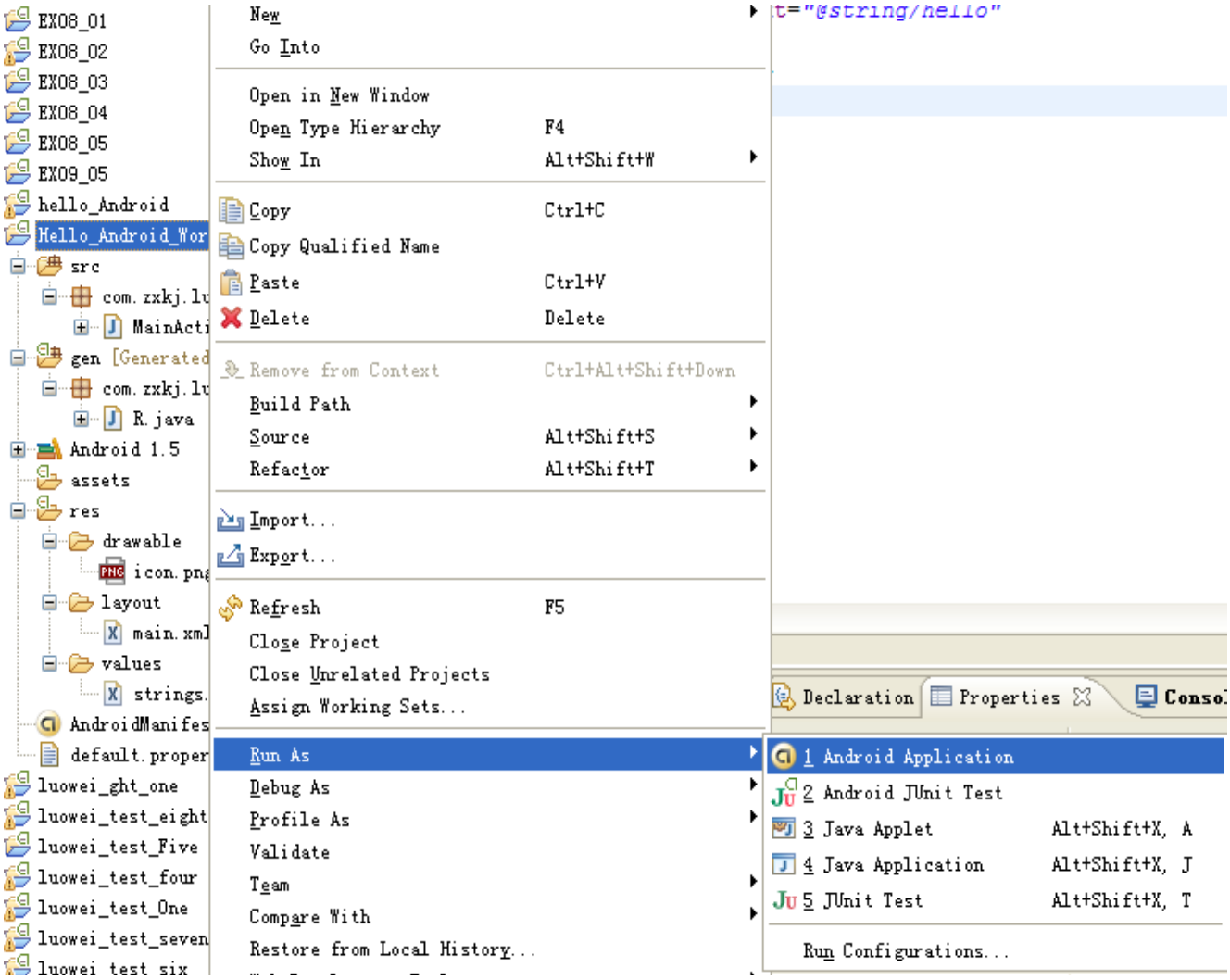
Res.values:存储的相关的样式文件（CSS）以及经常用到的字符串信息的声明，但是也是以 xml 进行封装的

AndroidManifest.xml:工程描述文件，相当于 j2ee 的 web.xml 文件 ，它可以设置第一启动的 Activity 文件（即 j2ee 的 Action 类）

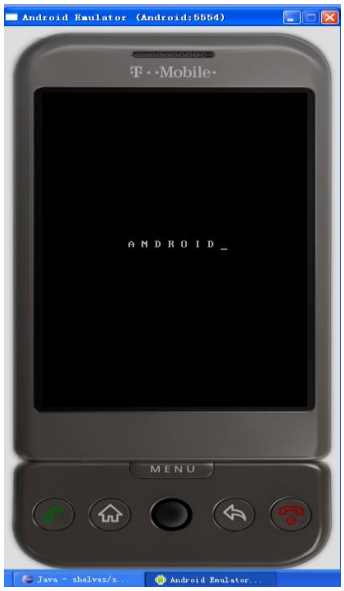
现在我们将此应用程序运行起来：

将鼠标移动到工程名右击:

选择……



此时应用程序将运行起来并弹出 dos 界面 即 模拟器



在此信息栏可以看到模拟器的运行过程：

emulator-5554	Online		Android [1.5, debug]
system_process	582		8600
com.android.phone	622		8601
android.process.acore	625		8602
com.android.alarmclock	655		8614
android.process.media	668		8618
com.android.mms	679		8620
com.example.android.SimpleWiktionary	692		8622
com.zxkj.luowei	727		8625 / 8700

可能由于等待过久 模拟器将处于省电状态 这个时候我们点击 MENU 可以使它运行我们的程序:

这个时候将出现如下字：
那么它是如何出现如下字体的呢？

首先我们运行程序的时候程序会去查看 AndroidManifest.xml(工程描述文件)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.zxkj.luowei"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

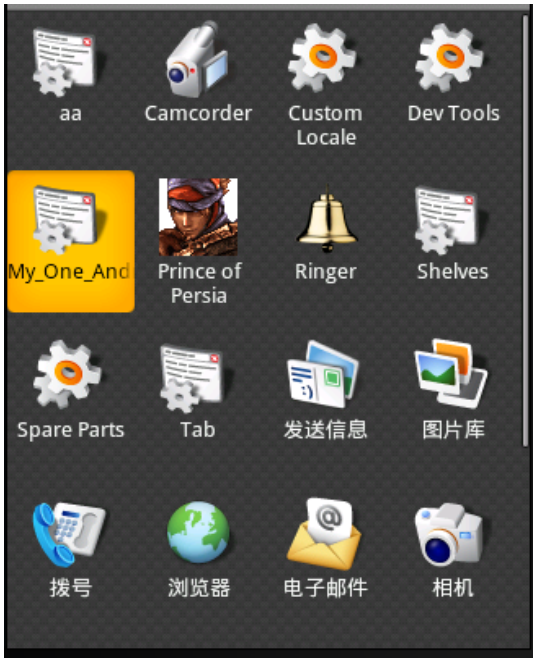
我们主要关注的就是 application 里面的配置信息：

android:icon:

指此应用程序的图片 在模拟器里面可以看到 ： 点击家的按钮

接着拉开抽屉

可以看到:



此图片就在：

Res.drawable（般用来存储相关应用的图片以及 mp3 播放文件等）

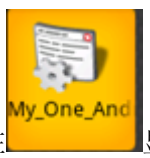
目录下

```
@drawable/icon
```

@代表在当前应用找

```
android:label
```

```
android:label="@string/app_name"
```

即在 

显示的名字

这里会在

我们打开 values/Strings.xml 文件

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">My_One_Android_Project</string>
</resources>
```

会发现一个 name 为 app_name 的 String 声明并且其值于我们之前模拟器所显示的标题一样

```
<activity android:name=".MainActivity">
```

声名一个Activity类　此类在 .MainActivity下其中点代表com.zxkj.luowei
即之前配置的 package="com.zxkj.luowei"

```
<activity android:label="@string/app_name">
```

代表

My_One_Android_Project

```
<intent-filter>
     <action android:name="android.intent.action.MAIN" />
     <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

<intent-filter>代表一个Action能做些什么事情 这里代表此Activity是第一启动项
<action android:name="android.intent.action.MAIN" />
一般情况下此name是可以任意改动的 但是除此之外 因为sdk后台会根据这个名字来调如果你改动则找不到了
<category android:name="android.intent.category.LAUNCHER" />
标志为第一启动项

接下来进入
MainActivity.java类

```
package com.zxkj.luowei;

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

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

会发现我们之前填写的MainActivity 类 继承自 Activity类
并且重写了此类的onCreate(Bundle savedInstanceState)方法
此方法会在实例化此类的时候一并调用（建议了解下Activity的生命周期）
import android.os.Bundle 用于映射字符串的值 可以在Android之间进行通讯
super.onCreate(savedInstanceState); 代表调用父类的方法并且将savedInstanceState传给父类
setContentView(R.layout.main);

现在打开R文件

```
package com.zxkj.luowei;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

此目录保存了res目录下的所有资源 并且给它们一个标识码 好让程序直接访问
如

```
public static final class string {
     public static final int app_name=0x7f040001;
     public static final int hello=0x7f040000;
}
```

代表Strings.xml文件下生命的String变量

此类是不可以被修改的并且当你更新
res此目录下的文件也同时被更新

如你向drawable丢进去一个

文件

会发现：

```
public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int qkss=0x7f020001;
}
```

多出一个qkss

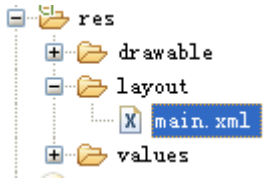
然而

setContentView(R.layout.main);

则代表

```
    public static final class layout {
        public static final int main=0x7f030000;
    }
```

打开:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

LinearLayout: 这是一个布局信息 标志它所包含的 View 都是线性布局是

android:orientation:
android:orientation="vertical"
可以改成：
android:orientation="horizontal"
vertical 此属性代表 View 是以垂直进行排序
horizontalvertical 此属性代表 View 是以横向进行排序

android:layout_width:
android:layout_width="fill_parent"
可以改成：
android:layout_width="wrap_content"
fill_parent 横向全部填充
wrap_content 横向顺其改变（如图片是多大就显示多大）
当然我们还可以为它设置大小如：
android:layout_width="61px"

android:layout_height 与 android:layout_width 类似

android:text:
android:text="@string/hello"
这里 text 代表是显示什么内容
@string/hello 代表在 values/Strings.xml 文件里面读取

我们打开 values/Strings.xml 文件

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">My_One_Android_Project</string>
</resources>
```

会发现一个 name 为 hello 的 String 声明并且其值于我们之前模拟器所显示的内容一样

可能还有些人对于
LinearLayout
布局中的
android:orientation="vertical"
android:orientation="horizontal"
这 2 者不是很了解 好的现在我做一个列子：

将 main.xml 进行修改部分

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

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="此为线性布局的垂直布局"
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="此为线性布局的垂直布局"
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="此为线性布局的垂直布局"
/>
</LinearLayout>
```

这里我多添加了几个 Button 即按钮　运行起来：

可以看倒这就是线性垂直的效果

再将

android:orientation 改成：

android:orientation="*horizontal*"

为什么只看见一个 Button 呢 按下 Ctrl 不放接着按下 F12 就知道原因了

现在我们将

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
 />
```

改成

```
<TextView
    android:id="@+id/text_Id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
```

意思就是为TextView声明一个标识符（Id）

并且名字是*text_Id*

查看R文件：

```
public static final class id {
    public static final int text_Id=0x7f050000;
 }
```

会发现多了一个text_Id并且还分配给它一个识别码

此识别码直接指向TextView

现在我们修改 MainActivity 的 OnCreate 方法

```
package com.zxkj.luowei;

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

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    private TextView one_Text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        one_Text=(TextView)findViewById(R.id.text_Id);
        one_Text.setText("Hello Word");
    }
}
```

```
private TextView one_Text;
```

声明一个 TextView 起名叫 one_Text

```
one_Text=(TextView)findViewById(R.id.text_Id);
```

找到 R.id.text_Id 标识码的 View

在 Android 里面所有的视图都继承自 View 这个类

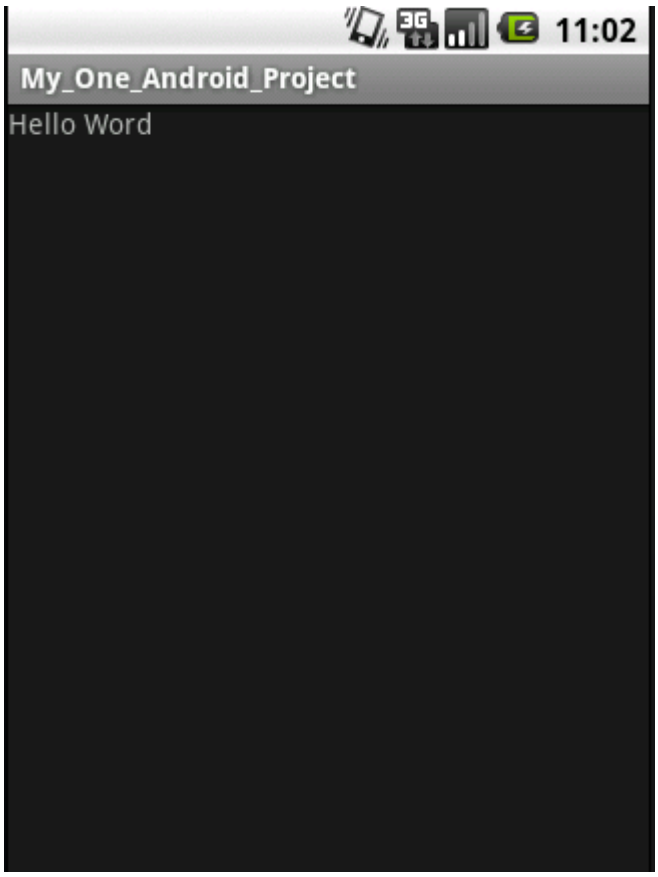
因此在这里我们需要强制转换

```
one_Text.setText("Hello Word");
```

为 one_Text 设置显示内容为 Hello Word

运行起来：

显示结果为：



浅谈 Android SDK

在 Android 的应用开发中，通常使用的是 java 语言开发，除了需要熟悉 JAVA 语言基础知识之外，还需要了解 Android 提供扩展的 java 功能

Android 重要包的描述

android.app	封装了 Android 应用程序全局模型的高级类。
android.content	包含用于在设备上访问和发布数据的类。
android.database	包含了用于浏览内容提供源返回数据的类。
android.database.sqlite	包含了 SQLLite 数据库管理类，应用程序可以利用这些类来管理其私有数据库。
android.graphics	允许你直接在屏幕上绘图的绘图工具，比如画布、颜色过滤器、点和矩形等。
android.graphics.drawable	提供了用于管理多种可视界面元素的类，这些可视界面元素仅用于显示，例如 bitmap 和 gradient。

android.graphics.glutils	提供了大量能够在 Android 设备上使用 OpenGL 嵌入式系统版(OpenGL ES)绘图的类.
android.hardware	提供对硬件设备的支持，这些硬件设备不一定会出现在每一个 Android 设备上。
android.location	定义 Android 定位和相关服务的类。
android.media	定位，视频，音频 和相关的服务
android.net	用于网络连接的类，功能比 ava.net.* 强大。
android.opengl	提供 OpenGL（高性能图形算法行业标准）工具。 3D 加速等
android.os	提供设备上基础的操作系统服务、信息传递和进程间通信。
android.provider	提供用于方便地访问 Android 支持的内容提供源的类。
android.sax	一个可以方便地编写高效、健壮的 SAX handler 的框架。
android.speech.recognition	提供用于语音识别的类。
android.telephony	提供了用于拨打、接收以及监听电话和电话状态的工具。
android.telephony.gsm	提供了用于从 GSM 电话上控制或读取数据的类。
android.text	提供了用于在屏幕上绘制或跟踪文本和文本跨度的类。
android.text.method	提供了用于监听或修改键盘输入的类。
android.text.style	提供了用于预览或修改视图对象中文本跨度形式的类。
android.util	提供了通用的工具方法，例如日期/时间操作、64 位编码解码器、字符串数组互换方法和与 XML 相关的方法。
android.view	提供了用于处理屏幕布局和用户交互的基本 UI 类。
android.view.animation	提供了动画处理的类
android.webkit	提供了浏览网页的工具。
android.widget	widget 包包含了用在应用程序屏幕上的 UI 元素(绝大部分可视)。

文件格式描述：

Android 的相关文件类型：

Java---应用程序源文件

Android 本身相当一部分是由 java 编写而成，而且 android 应用必须使用 java 开发

Class---java 编译后的目标文件：

是由 java 虚拟机编译而成一个字节码文件，在之前我们用所学的 j2ee 以及 j2se 它是一个可执行文件，但是在 Android 当中它只是一个目标文件即过渡文件

Dex---Android 平台可执行文件：

Android 自己提供了一个虚拟机（Dalvik）,这种虚拟机执行的并非 java 字节码，而是另一种字节码：dex 格式的字节码,在 JVM 将 java 文件编译成 Class 文件后，再次通过 Android 平台工具将此 Class 文件转换成 dex 字节码

Apk 文件---Android 上的安装文件

Apk 是 Android 安装包的扩展名，一个 Android 安装包包含了与某个 Android 应用程序相关的所有文件，apk 文件将 androidManifest.xml 文件，应用程序代码（dex 文件）资源文件和其他文件打成一个压缩包，一个工程只能打进一个 apk 文件（有点类似 exe 文件）