# Android工程的目录结构

* **src: 项目的java代码**
  + **MainActivity.java**
* **gen:**
* R.java: 保存项目中使用的资源的id名
* buildConfig：应用是否可以debug
* **Android 4.3: 导入这个包，应用才可以使用Android的api**
* **assets**：**资源文件夹，存放视频或者音乐等较大的资源文件**
* **bin：存放应用打包编译后的文件**
  + 所有的.java文件变成的dex文件，存放在这。apk也在这
* **libs：第三方类库的jar包**
* android-support-v4.jar：Android支持类库
* **res：资源文件夹，在这个文件夹中的所有资源，都会有资源id，读取时通过资源id就可以读取**
* drawable-xxdpi :

包含Android应用中的图片资源文件。

按清晰度分为：高清、清晰度低、比较清晰、超高清、超超高清

* values：
  + strings: 存放字符串资源，每个资源都会有一个资源id。做国际化
  + dimens：长度资源文件，用来定义长度资源
  + style：样式和主题资源文件
* layout：存放布局文件，把布局文件通过资源id指定给activity，界面就会显示出该布局文件定义的布局
  + **activity\_main.xml: MainActivity.的布局文件**
  + menu：菜单配置文件夹，保存菜单的配置文件，决定菜单的样式
  + **注意：资源文件名称要小写，资源id不能出现中文**
  + **注意：加资源文件的时候，必须在工程没有错误的情况下才能加。否则资源文件的引用不会被添加到R文件中**
* **AndroidManifest.xml：清单文件。类似web.xml**。
* 配置Adroid四大组件，类似配置Servlet、Fileter
* **project.properties：可以在这里更改SDK 的API 版本。即使用的Android系统版本**

**MainActivity.java**

|  |
| --- |
| **public** **class** MainActivity **extends** Activity {  //Activity创建时，此方法调用  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  //设置显示内容，通过设置资源id指定该Activity显示哪个布局文件  setContentView(R.layout.*activity\_main*);  }  } |

**activity\_main.xml**

|  |
| --- |
| <RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:paddingBottom=*"@dimen/activity\_vertical\_margin"*  android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*  android:paddingRight=*"@dimen/activity\_horizontal\_margin"*  android:paddingTop=*"@dimen/activity\_vertical\_margin"*  tools:context=*".MainActivity"* >  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"@string/hello\_world"* />  </RelativeLayout> |

**AndroidManifest.xml**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.example.hello"  android:versionCode="1"  android:versionName="可随便改。这是应用版本号" >  <uses-sdk  android:minSdkVersion="8"  android:targetSdkVersion="17" />  <application  android:allowBackup="true"  android:icon="@drawable/ic\_launcher"  android:label="@string/app\_name"  android:theme="@style/AppTheme" >  <activity  android:name="com.example.hello.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>  </manifest> |

**project.properties**

|  |
| --- |
| target=android-18 |

# SDK目录介绍

* **Tools：存放SDK工具集，这些工具在开发中都要用。很重要，但不需要管。Eclipse会自动调用**
* **docs：帮助文档**
* **add-ons: 预留目录**
* **.android: avd，创建模拟器提供的文件**
* **build-tools: 工具**
* **extras：**
  + android
    - support：Androoid支持类库
  + intel
    - Hardware\_Accelerated\_Execution\_Manager：模拟器加速器
* **sources：源代码**
* **tmp：临时文件**
* **system-images: 各种架构的系统镜像**
* **platforms： 存放SDK平台。**
* **platforms-tools：存放SDK平台工具集**
* dx.bat: 把所有的.class文件变成了一个dex文件
* aapt.exe：安卓应用的打包工具
* adb.exe: Android调试桥

# SDK Manager.exe：管理sdk的工具

* **Tools**
* Android SDK Toools：SDK工具集，**必须下载**
* Android SDK platforms-tools：SDK平台工具集，**必须下载**
* **Android 4.3(API 18):**
* **SDK Platform: SDK平台，就是API，只要下载了这个，就可以进行对应版本的开发了。必须下载。**
* **Documentation for Android SDK：帮助文档**
* **Sources for Android SDK: Android API源码**
* **Samples for SDK: 代码示例**
* **ARM EABI v7a System Image: ARM架构的系统镜像。创建模拟器的时候，可以选择ARM架构的CPU。**
* **mips System Image: mips架构的系统镜像。创建模拟器的时候，可以选择mips架构的CPU。**
* **Intel x86 Atom System Image: Intel架构的系统镜像。创建模拟器的时候，可以选择Intel架构的CPU。**
* **Googles API：api，额外的api**
* **Extras：**
* Android Support Library: Android支持类库。是jar包。低版本使用高版本新特性的时候使用的（导入相应版本的jar包即可）。
* Intel x86 Emulator Accelerator(HAXM): 模拟器加速器，装这个的要求是，cpu是Interl的（I3以上级别）。
* Google USB Driver：进行USB调试的驱动，这个是谷歌亲儿子才能用的：HTC、LG
* Google Play Service：谷歌商店服务

# DDMS透视图：在eclipose右上角

* **Devices：当前跟Eclipse建立连接的Android设备**
  + 列出开启的所有模拟器，和已连接电脑的手机（开启了usb调试）
  + 还有当前手机正在运行的进程
* **Screen Capture：照相机的图标；可以截图，截模拟器的图**
* **File Explo：右边。文件的浏览。都是模拟器的文件。相当于手机的文件管理器**
  + mnt/sdcart；模拟器的sd卡目录。相当于手机sd卡目录
  + data/app: 第三方应用保存径
  + dta/data/应用包名文件夹：系统为每一个应用提供的一个专属空间。用来保存应用运行时产生的一些文件
* **Emulator**
  + Telephony Actions选项卡:可以模拟来电，和短信
    - Telephony Actions选项卡:可以模拟来电，和短信
    - Voice：来电
    - SMS: 短信
    - Massage：短信内容，模拟器不支持中文
    - call：拨通
    - send：发送短信

# 创建模拟器

* **AVD Name：模拟器名称**
* **Device：模拟器分辨率大小。电脑配置好，就选大点**
  + 安卓常见分辨率：320 \* 480、480 \* 800、1280 \* 720
* **Target：模拟器的安卓版本。**
  + 对应api级别。例：2.3 -- 10、3.0 -- 11、4.0 -- 14、 4.1.2 – 16
* **CPU/ABI：选择系统镜像。就是选择什么架构的cpu。**
  + 前提是你选择的Android版本，下载好了Intel/ARM架构的系统镜像。否则模拟器不可创建
* **Keyboard: 实体按键**
* **Skin： 实体按键的皮肤。不勾上的话，实体按键显示不出来**
* **Front Camera：前置摄像头**
* **Back Camera：后置摄像头**
* **Memory Options：**
  + RAM: 运行时内存。相当于电脑的内存条。默认不改。
  + ROM: 存储内存，相当于电脑的一块微小的硬盘， 断电数据不丢失
  + VM Heap：虚拟机的堆内存。每个应用能占多少内存。给32/16
* **Internl Storage： 内部存储空间。相当于手机自带的存储空间。这是必须有的**
* **SD Card：SD卡。可有可无**
  + Size：给sd卡分配多大的空间，会在电脑硬盘上占用对应的空间。给40
  + File：直接在电脑上指定一个空间，当做sd卡空间。不推荐
* **Emulation Options：一般不勾选**
  + Snapshpot：快照。相当于window的休眠功能。
  + Use Host GPU：图形渲染功能

# 模拟器创建好后。启动模拟器时

* **Wipa user data**：擦除用户数据。相当于恢复出厂设置
* **启动后：**
  + 左上角顶部的编号：5556:xin 5554:ha76。
  + 数字：相当于电话号码，可在一个模拟器中拨打另一个模拟器的电话号码
* **Scale display to real size: 对模拟器的屏幕进行缩放，不会改变分辨率的大小**

# ADB：建立Eclipse和Android设备之间的连接

* **ADB指令：首先要在path下配置：adb.exe所在的目录配置进path变量中**
* **adb kill-server: 杀死adb进程（DDMS中，与模拟器/手机，断开连接）**
* **adb start-server: 启动adb进程（与模拟器建立连接）**
* **adb devices：查看当前与开发环境连接的设备，此命令也可以启动adb进程**
* **adb install E:\xx.apk: 往模拟器/真机安装apk**
* **adb uninstall 应用包名：快速从（模拟器/真机）卸载一个Android应用。**
* **adb shell: 进入Android命令行。在这里输入liux命令**
  + ls：显示当前手机目录的所有文件（包括文件夹）
  + cd 目录/子目录：进入指定的目录
  + ps：查看运行进程
* **adb push text.txt /目录/子目录: 把电脑上的文件，传到手机上指定的目录**
* **adb pull /目录/子目录/test.txt: 把手机上指定目录的文件，传到电脑上。当前cmd所在的目录**
* **netstat -ano: 看系统的端口占用情况。ADB占用的端口是5037**

# Hello Word

* **创建Android工程：new-->Android Application Project**
* **Application Name：应用名称**
* **Project Name：工程名称**
* **Package Nmae：包名**
* **Minimum Reqrired SDK：最小的sdk版本。此应用程序最低可以在什么版本的Android系统上运行。一般2.3.3。最重要**
* **Target SDK：目标的sdk版本。没什么意义**
* **Compile With：编译的sdk版本， 导入什么版本的jar包，一般选最新。很重要**
* **Theme：主题，UI界面有关。默认即可**
* **Create custom launcher icon: 创建一个自定义的图标**
* **Create activity：创建一个界面。**
* **Mark this project as a library: 把项目当成类库。项目充当为jar包的功能**
* **部署Android应用**
  + 项目--> Run As --> Android Application
  + 绿色播放按钮
  + Ctrl + F11

# Android应用开发步骤

* + 需求分析
  + 画UI
  + 根据UI写对应的业务逻辑
  + 测试

# 案例：电话拨号器

* **UI设计，为MainActivity进行布局**

**activity\_main.xml**

|  |
| --- |
| <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  tools:context=*".MainActivity"*  android:orientation=*"vertical"*  >  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"请输入手机号码"* />    <EditText  android:id=*"@+id/et\_phone"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*/>    <Button  android:id=*"@+id/bt\_call"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"拨打"*/>  </LinearLayout> |

* **实现业务逻辑：在MianActivity中，编写打打电话操作**

**MainActivity.java**

|  |
| --- |
| **package** com.example.dialer;  **import** android.app.Activity;  **import** android.content.Intent;  **import** android.net.Uri;  **import** android.os.Bundle;  **import** android.text.Editable;  **import** android.view.View;  **import** android.view.View.OnClickListener;  **import** android.widget.Button;  **import** android.widget.EditText;  **public** **class** MainActivity **extends** Activity {    //Activity被创建时调用  @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  //为Activity设置布局界面  setContentView(R.layout.*activity\_main*);    /\*  \* 1.得到拨打按钮  \* > 通过资源id获取。前提是在布局文件中，为按钮组件添加了资源id  \* > 得到一个View对象。布局文件中的所有节点(组件)，在Android API中都有对应的类  \* 并且都继承至View类  \*/  Button bt\_call = (Button) findViewById(R.id.*bt\_call*);    /\*  \* 2. 为拨打按钮添加点击事件  \* > 需要接收View.OnClickListener参数，它是View的内部接口  \* > 并且重写它的onClick接口  \*/    bt\_call.setOnClickListener(**new** MyClickListener());    }    //定义一个自己的点击事件监听类，继承OnClickListener  **private** **class** MyClickListener **implements** OnClickListener {  //按钮被点击时调用  **public** **void** onClick(View v) {  //3. 得到输入框  EditText et\_phone = (EditText) findViewById(R.id.*et\_phone*);    /\*  \* 4.得到输入框的文字  \* > 得到的是一个Editable，它是输入框的内容的对象  \* > 使用toString()，即可得到输入框内容对象的字符串  \*/  Editable ea = et\_phone.getText();  String callPhone = ea.toString();    //5. 我们需要告诉系统，我们的动作：我要打电话  //创建意图对象  Intent intent = **new** Intent();  //把我们的动作封装进意图对象中  intent.setAction(Intent.*ACTION\_CALL*);  //告诉意图对象，我们要打给谁。这个先这么写。后面会讲为什么这么写  intent.setData(Uri.*parse*("tel:" + callPhone));    //把我们打电话的动作，告诉系统。  //启动一个打电话应用的Activity，这个Activity的意图是要打电话  startActivity(intent);    }  }  } |

* **打电话需要获取到打电话的权限。在清单文件中，获取到打电话的权限**

**在根节点中，添加一个子节点**

**AndroidManifest.xml**

|  |
| --- |
| **<uses-permission android:name="android.permission.CALL\_PHONE"/>** |

* **应用程序部署到模拟器/真机中，测试**

# 按钮点击事件

源代码：code/btonclick

# 短信发送器

* UI布局

activity\_main.xml

|  |
| --- |
| <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  xmlns:tools=*"http://schemas.android.com/tools"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:paddingBottom=*"@dimen/activity\_vertical\_margin"*  android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*  android:paddingRight=*"@dimen/activity\_horizontal\_margin"*  android:paddingTop=*"@dimen/activity\_vertical\_margin"*  tools:context=*".MainActivity"*  android:orientation=*"vertical"* >  <EditText  android:id=*"@+id/et\_phone"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:inputType=*"phone"*  android:hint=*"请输入对方号码"* />  <EditText  android:id=*"@+id/et\_content"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:lines=*"5"*  android:gravity=*"top"*  android:hint=*"请输入短信内容"* />    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"发送"*  android:onClick=*"send"*/>    </LinearLayout> |

* 业务逻辑实现

MainActivity.java

|  |
| --- |
| **package** com.example.sendsms;  **import** java.util.ArrayList;  **import** android.app.Activity;  **import** android.os.Bundle;  **import** android.telephony.SmsManager;  **import** android.view.View;  **import** android.widget.EditText;  **public** **class** MainActivity **extends** Activity {  @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);  }    //发送按钮被点击时调用  **public** **void** send(View v) {  //因为短信拨号器的api可以直接调用，所有不用设置动作了。直接调用短信的API    //拿到用户的接收者号码和内容  EditText et\_phone = (EditText) findViewById(R.id.*et\_phone*);  EditText et\_content = (EditText) findViewById(R.id.et\_content);    String phone = et\_phone.getText().toString();  String content = et\_content.getText().toString();    //1. 获得短信管理器  SmsManager sm = SmsManager.*getDefault*();    /\*  \* 2. 发送短信  \* > 需要给出短信接收者  \* > 短信服务中心的号码。直接给null  \* > 短信内容  \* > 短信发送成功后的广播。null  \* > 短信被对方成功接收后的广播。 null  \*/  //sm.sendTextMessage(phone, null, content, null, null);    //如果短信的内容过多是无法发送的，所有我们把内容截取成几个内容。分段发送  ArrayList<String> al = sm.divideMessage(content);  **for**(String sms : al) {  sm.sendTextMessage(phone, **null**, sms, **null**, **null**);  }  }  } |

* 在清单文件中获取发送短信的权限

AndroidManifest.xml

|  |
| --- |
| <uses-permission android:name="android.permission.SEND\_SMS"/> |