Android 介绍

1. Android 开发环境搭建
2. 第一个项目—HelloAndroid
3. Android项目目录结构
4. Android应用解析
5. Android生命周期
6. Widget开发
7. Android中的显示单位
8. DDMS的简介与使用
9. apk的安装与卸载

1. Android 开发环境搭建

- 所需软件：
- JDK： 1.6以上
- Eclipse： 3.4以上
- Android SDK ： http://developer.android.com/sdk
- ADT ： https://dl-ssl.google.com/android/eclipse

Android 开发环境搭建

步骤1：安装JDK 、配置java环境

步骤2：Eclipse安装

步骤3：安装SDK：下载解压后，运行“SDK Setup.exe”，选择要安装的API。版本和SDK文档

SDK配置：将SDK安装文件夹下的tools文件夹的路径加入环境变量“Path”中

步骤4：ADT ：Android Development Tools Plug-in, 是Android在Eclipse上的开发工具。

安装ADT：启动eclipse，点击“Help”菜单，依次 选择“Software Update”和“Avaiable Software”选项，点击“Add Site”按钮，输入地址“http：//dl-ssl.google.com/android/eclipse”,根据提示即可完成

关联SDK：打开菜单“window”，依次选择“Preferences”-——“Android”，点击“Browse…”,选择Android SDK的安装路径，点击”OK”即可

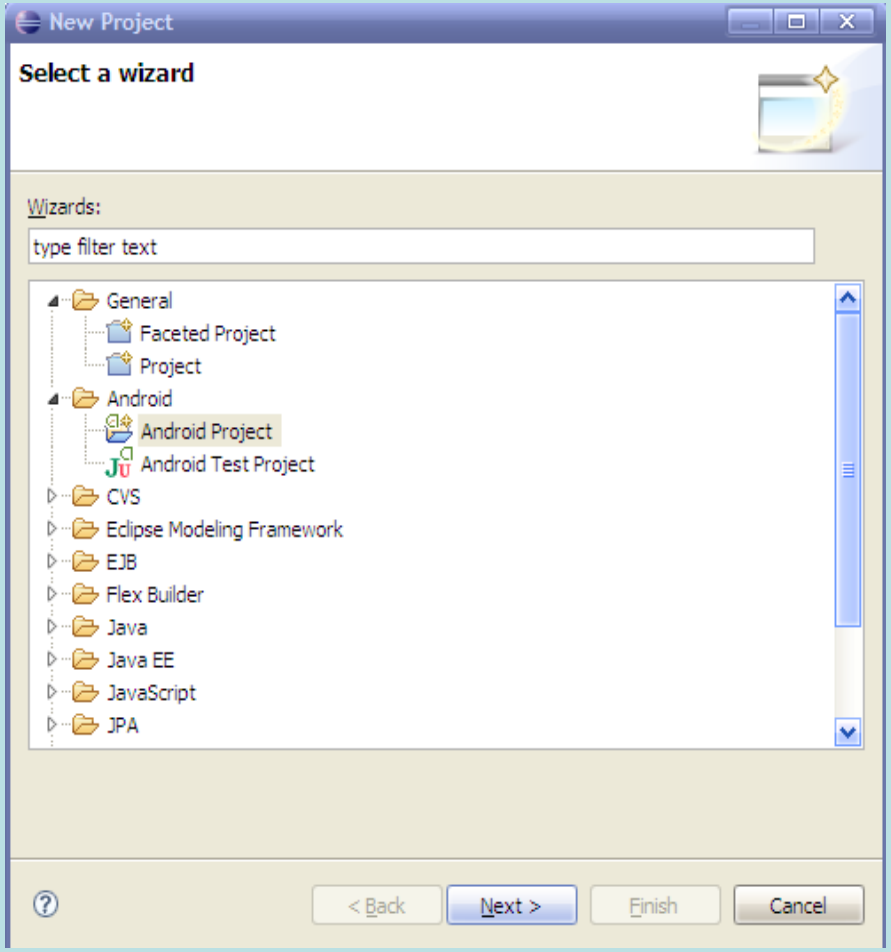
Android 开发环境搭建

SDK目录：

add-ons:第3方插件或者是jar包
docs：相关文档，点击index.html可查看
platforms：各Android版本资源
samples：google提供的应用，导入工程即可运行
tools：各种Android工具,如adb.exe等
usb_driver：USB驱动

2. 第一个项目—HelloAndroid

(1)右键New——Project…，
在“New Project”对话框中选择
Android——Android Project



第一个项目—HelloAndroid

(2)点击“next”按钮，进入“New Android Project”，

Project name中输入“HelloAndroid”，

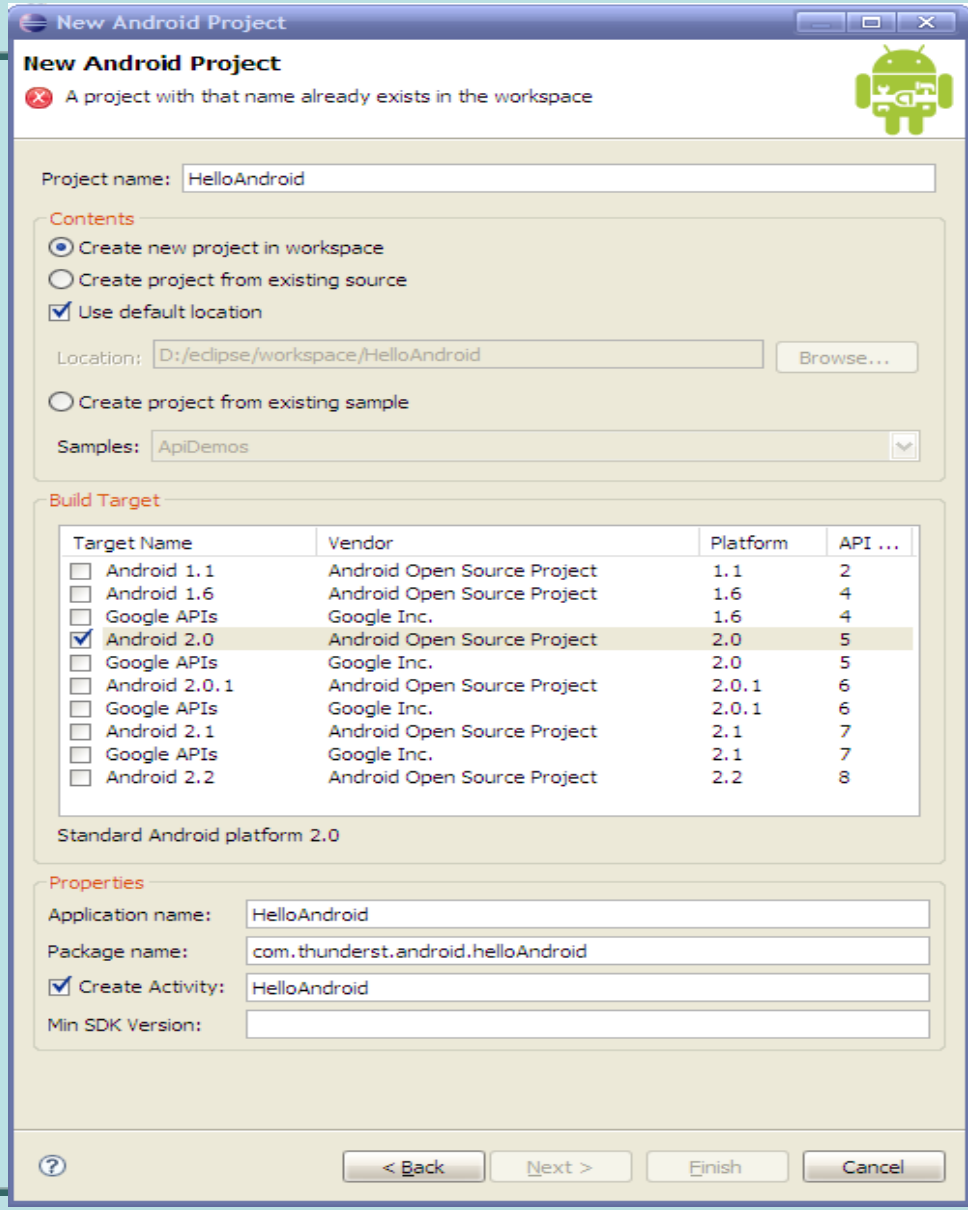
Build Target中选择“Android 2.0”或其他

Application name中输入“HelloAndroid”

Package name中输入“com.thunderst.android.helloAndroid”

Create Activity中输入“HelloAndroid”

点击“Finish”，HelloAndroid项目创建完成



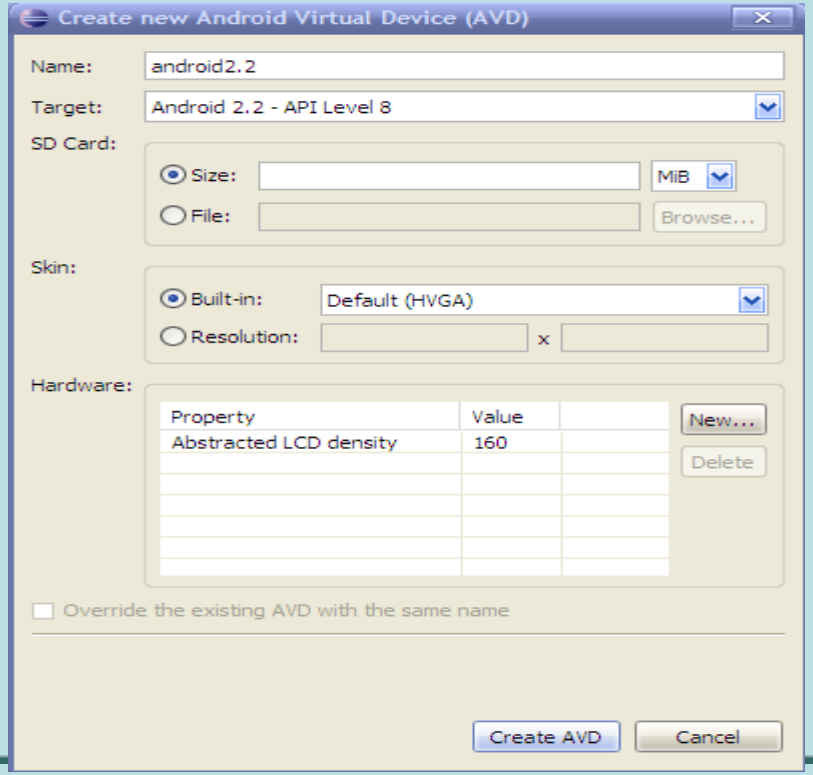
第一个项目—HelloAndroid

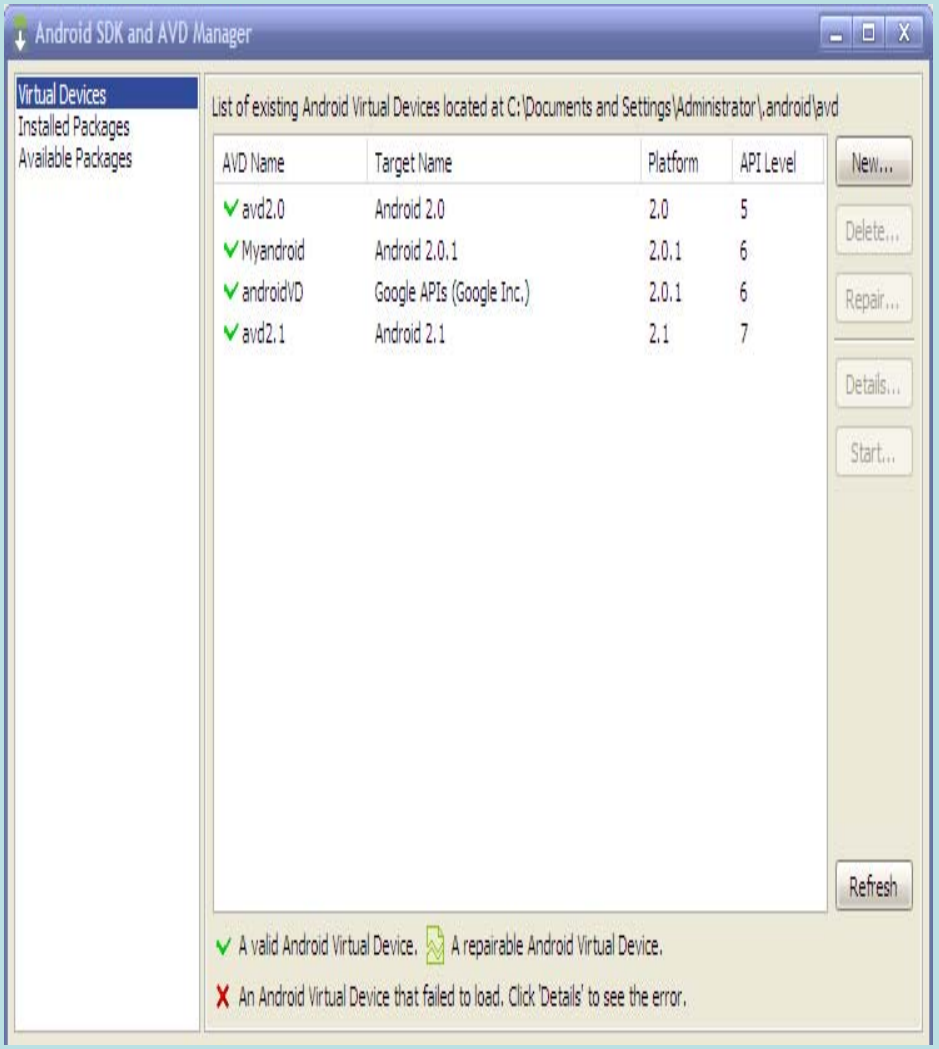
(3)创建AVD（Android Virtual Device）

点击 “window”菜单，选择“Android SDK and AVD manager”

点击左边的“Virtual Devices”，再点击右边的“New…”,新建AVD

填写Name，Target处选择API等级，Size处填写SD卡的大小，Skin处设计模拟器的风格





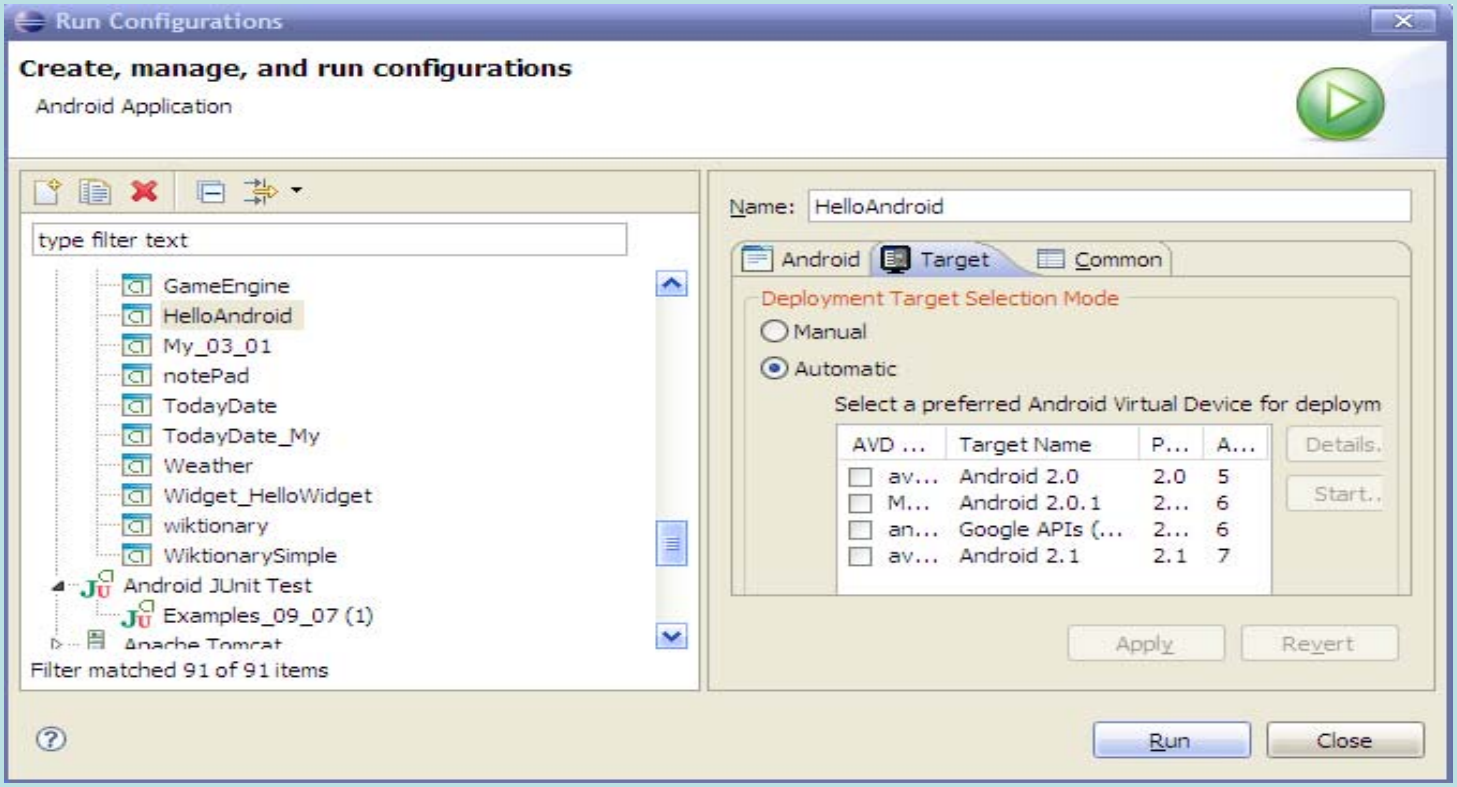
第一个项目—HelloAndroid

(4)配置AVD

点击 “Run”菜单，选择“Run configurations”

点击左边的项目，在右边的Target中选择已创建的AVD

点击Run按钮，即可运行HelloAndroid项目



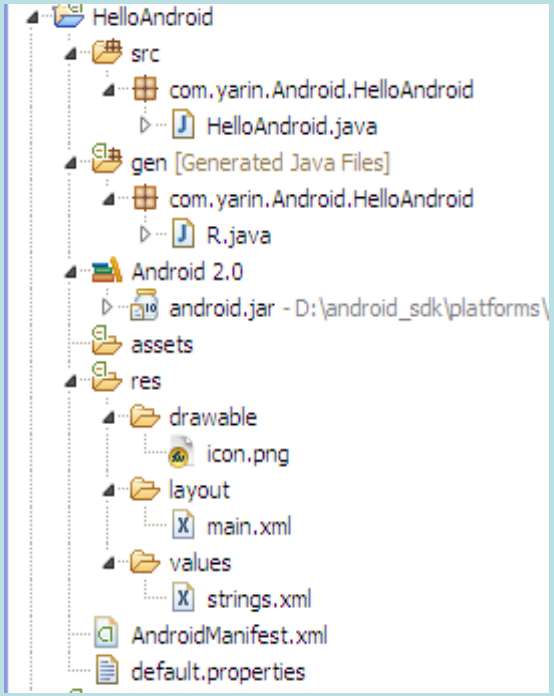
3. Android项目目录结构

- **src/ java原代码存放目录**
- **gen/ 自动生成目录，项目中所有资源的索引文件**

  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，图片或数据。
- **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 项目环境信息**，一般是不需要修改此文件



Android项目目录结构

HelloAndroid.java分析：

1.此类必须继承Activity，至少应该覆盖onCreate()方法

2.setContentView(R.layout.main)方法设置了此Activity显示的UI

3.Log.v(“”,””)是Android提供的查看日志的方法

v:verbose

d:debug

i:info

e:error

w:warn

```
1  package com.yarin.Android.HelloAndroid;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.util.Log;
6
7  public class HelloAndroid extends Activity {
8
9      private static final String TAG = "HelloAndroid";
10
11     public void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13
14         Log.v(TAG, "VERBOSE");
15         Log.d(TAG, "DEBUG");
16         Log.i(TAG, "INFO");
17         Log.w(TAG, "WARN");
18         Log.e(TAG, "ERROR");
19
20         setContentView(R.layout.main);
21     }
22 }
23
```

Android项目目录结构

R.java分析：

1. 在建立项目自动生成，是只读文件，不能更改，是项目中所有资源的索引文件
2. 定义了很多常量，这些常量的名字都与res文件夹中的文件名相同
3. 在项目中加入新的资源时，只要刷新一下该项目，R.java文件便可以自动生成新的资源索引

```
1  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
7
8  package com.yarin.Android.HelloAndroid;
9
10 public final class R {
11     public static final class attr {
12     }
13     public static final class drawable {
14         public static final int icon=0x7f020000;
15     }
16     public static final class layout {
17         public static final int main=0x7f030000;
18     }
19     public static final class string {
20         public static final int app_name=0x7f040001;
21         public static final int hello=0x7f040000;
22     }
23 }
24
```

Android项目目录结构

AndroidManifest.xml分析：

manifest:根节点

xmlns:命名空间

package：应用程序包

application：application级别组件的根节点

application:icon 应用程序图标

application:label 应用程序名称

activity：与实际的Activity类对应

Intent-filter：此activity支持的intent值

action：组件支持的Intent action

category：组件支持的Intent Category

uses-sdk：此应用程序使用的SDK版本

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.yarin.Android.HelloAndroid"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  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="5" />
</manifest>
```

Android项目目录结构

drawable分析：

1. 存放应用需要的图片等资源

Android项目目录结构

Layout/main.xml分析：

1. UI界面的布局文件
2. <LinearLayout>:线性版面配置，所有组件由上到下排列

   android:orientation 表示从上到下垂直排列

   android:layout_width 当前视图占屏幕的宽度

   android:layout_heigth 当前视图占屏幕的高度

   android:text 填充的文字

   fill_parent 填充整个屏幕

   wrap_content 根据文字栏位的大小改变此视图的高或宽

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     >
7 <TextView
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content"
10    android:text="@string/hello"
11    />
12 </LinearLayout>
13
```

Android项目目录结构

Values/strings.java分析：

1. 定义了字符串资源
2. 对应于R.java中的

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

3. 程序中使用这些资源：

```
Resource res = this.getContext().getResources();
String appName = (String)res.getString(R.string.app_name);
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <string name="hello">Hello World, HelloAndroid!</string>
4     <string name="app_name">HelloAndroid</string>
5 </resources>
6
```

4. Android应用解析

- Activity——活动
- Intent——意图
- Content Provider——内容
- Service——服务

(1)Activity

- 最基本的模块，称之为“活动”
- 一个activity就是一个单独的屏幕
- 每个activity都被实现为一个独立的类，都继承android.app.Activity
- 每个activity都会显示由视图UI组成的用户接口，对事件进行 响应

Android应用解析

(2)Intent

- 此类实现在Activity与Activity之间进行切换
- 描述了应用的功能，即某个Activity能够做什么事情
- Intent描述中两部分：action(动作)、data(数据)
- action表示activity能做什么样的动作：MAIN、VIEW、 PICK、 EDIT
- data表示动作对应的数据，以URI的形式表示
- 通过解析Intent，可以实现从一个屏幕到另一个屏幕的导航
- Activity中调用startActivity(Intent myIntent)，系统会在所有注册的activity中查找IntentFilter ，找到最匹配myIntent的Intent所对应的activity，此activity便启动了

Android应用解析

(3)Content Provider

- 使应用与应用之间的数据可以共享
- Android中，每个应用都运行在各自的进程中，当应用需要访问其他应用的数据时，数据需要在不同的虚拟机之间传递，这样的操作有些困难（一般的，不能读取其他应用的db文件），content provider正是用来解决不同应用包之间共享数据的

Android应用解析

(4) Service

- “服务”，生命周期长且没有用户界面，比如当正在选择播放音乐时，又要进行其他的操作（如写短信、发邮件...），此时的音乐应该在后台继续播放
- Activity中通过Context.startServiceI()启动一个service

5. Android生命周期

Activity：

- 每个activity都要继承android.app.Activity类，
- 并重写其中的某些方法

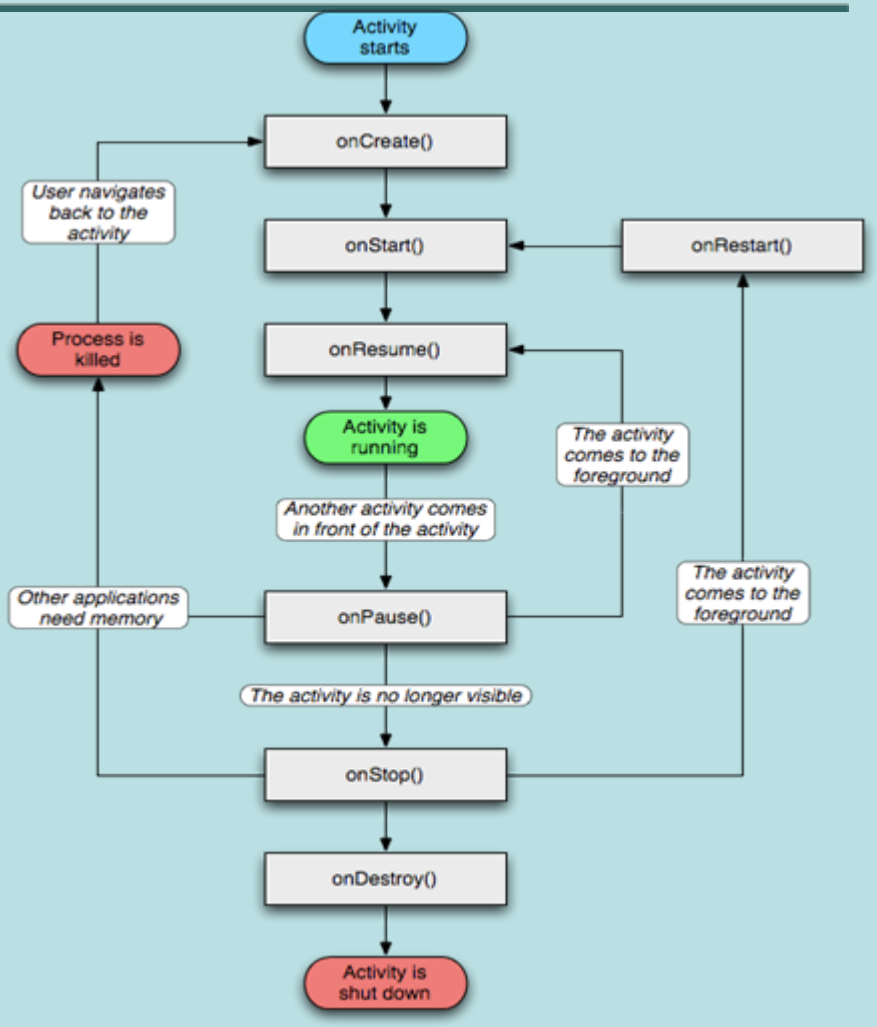
onCreate()

onStart()

onResume()

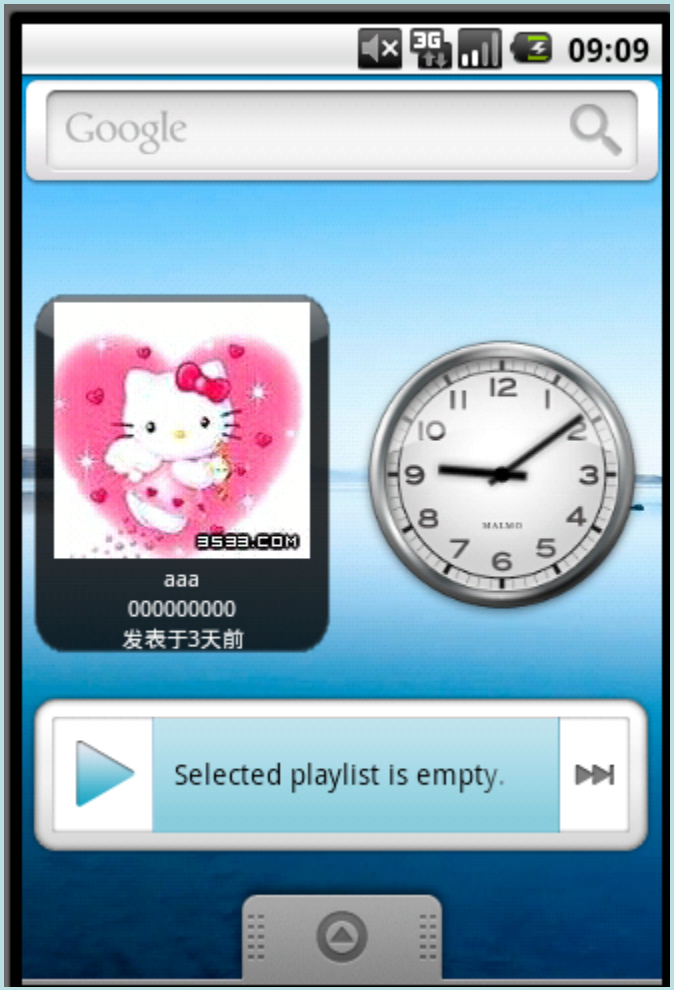
onPause()

onDestroy()



6. Widget开发

- 一种很小的应用程序
- 可以拖到用户桌面并进行交互
- 比如脑中、日历、音乐播放器等



Widget开发

1. 布局文件：res / layout / appwidget_provider.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/appwidget_text"
    android:textColor="#ff000000"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>

```

Widget开发

2. 描述此桌面组件的文件：res / xml / xmlappwidget_provider.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="100dp"
    android:minHeight="50dp"
    android:updatePeriodMillis="1000"
    android:initialLayout="@layout/appwidget_provider"
    android:configure="com.yarin.android.Examples_09_07.Activity01"
    >
</appwidget-provider>
```

注：android:minWidth 此Widget的宽

android:minHeight 此Widget的高

android:updatePeriodMillis 定时更新

（根据官方定义，此设置不一定会生效）

android:initialLayout 关联创建的布局文件

android:configure 如果启动此Widget需要进行配置，必须设置此项，一般为activity

Widget开发

3. 创建一个类ExampleAppWidgetProvider，
   继承自AppWidgetProvider类，
   至少覆盖onUpdate()方法，用于更新此Widget

```
1  package com.yarin.android.Examples_09_07;
2
3  import android.appwidget.AppWidgetManager;
9
10 public class ExampleAppWidgetProvider extends AppWidgetProvider {
11     // □□↓□□□□□□□
12     public void onUpdate(Context context, AppWidgetManager appWidgetManager,
13             int[] appWidgetIds) {
14         final int N = appWidgetIds.length;
15         for (int i = 0; i < N; i++) {
16             int appWidgetId = appWidgetIds[i];
17             String titlePrefix = Activity01.loadTitlePref(context, appWidgetId);
18             updateAppWidget(context, appWidgetManager, appWidgetId, titlePrefix);
19         }
20     }
```

Widget开发

4. AndroidManifestl.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.yarin.android.Examples_09_07"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <receiver android:name=".ExampleAppWidgetProvider">
            <meta-data android:name="android.appwidget.provider"
                       android:resource="@xml/appwidget_provider" />
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
            </intent-filter>
        </receiver>
        <activity android:name=".Activity01">
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET_CONFIGURE" />
            </intent-filter>
        </activity>
        <receiver android:name=".ExampleBroadcastReceiver" android:enabled="false">
            <intent-filter>
                <action android:name="android.intent.ACTION_TIMEZONE_CHANGED" />
                <action android:name="android.intent.ACTION_TIME" />
            </intent-filter>
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="5" />
</manifest>
```

注：<receiver>:注册的appwidget

android:name 已创建的Widget类-ExampleAppWidgetProvider

<meta-data>:描述Widget的元数据

android:name 必须为“android.appwidget.provider”

android:resource 此Widget的描述文件，位于xml文件夹下

<intent-filter>的<action>:必须为“android.appwidget.action.APPWIDGET_UPDATE”

7. Android中的显示单位

- px (pixels)像素
  一般HVGA代表320x480像素，这个用的比较多。
- dip或dp (device independent pixels)设备独立像素
  这个和设备硬件有关，一般为了支持WVGA、HVGA和QVGA 推荐使用这个，不依赖像素。
- sp (scaled pixels — best for text size)比例像素
  主要处理字体的大小，可以根据系统的字体自适应。

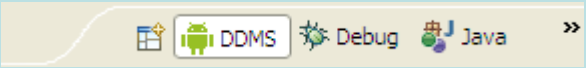
下面几个不太常用：
- in (inches)英寸
- mm (millimeters)毫米
- pt (points)点，1/72英寸

为了适应不同分辨率，不同的像素密度，推荐使用dip ，文字使用sp。

8. DDMS的简介与使用

1. DDMS的简介：

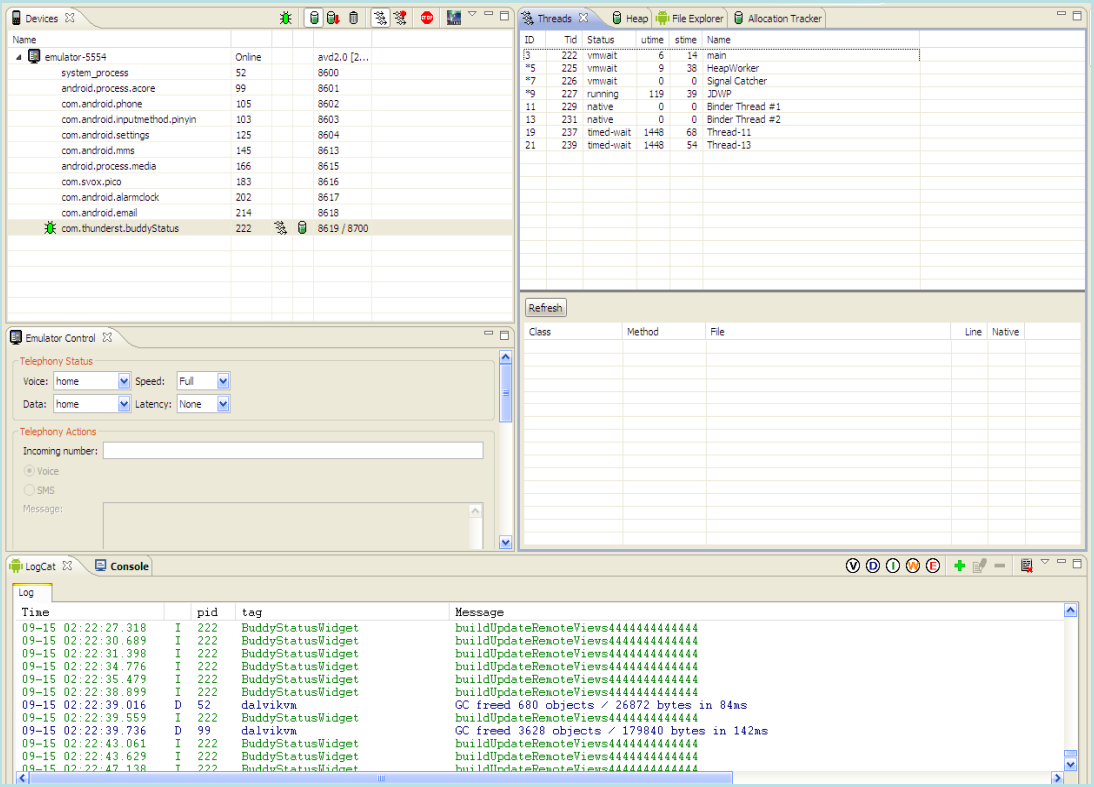
- DDMS 的全称是Dalvik Debug Monitor Service
- 作用：Debug、为测试设备截屏，针对特定的进程查看正在运行的线程以及堆信息、Logcat、广播状态信息、模拟电话呼叫、接收SMS、虚拟地理坐标等等
- 安装好ADT后会有一个DDMS的perspective



DDMS的简介与使用

DDMS的简介：

- DDMS 视图为：

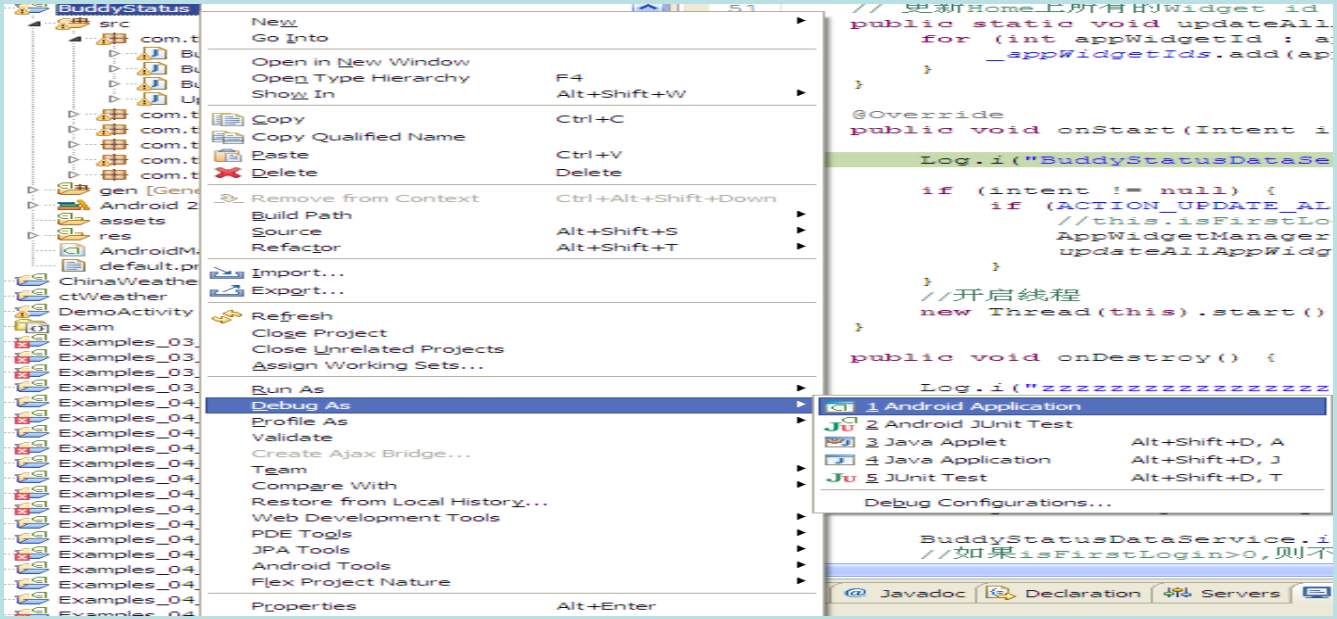


DDMS的简介与使用

2. DDMS的使用——Debug

(1) 在程序中加入断点

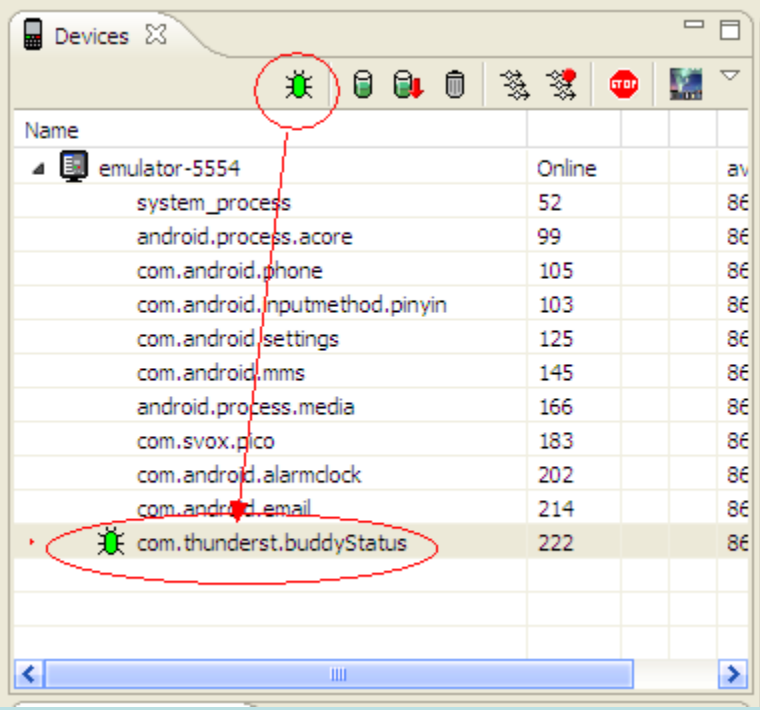
(2) 工程点击右键，开启Debug，然后打开DDMS视图



DDMS的简介与使用

DDMS的使用——Debug

(3) 在坐上角的Devices中，选中要Debug的工程，后点击 图标，这样便可以进入Debug模式



DDMS的简介与使用

3. DDMS的使用——emulator control

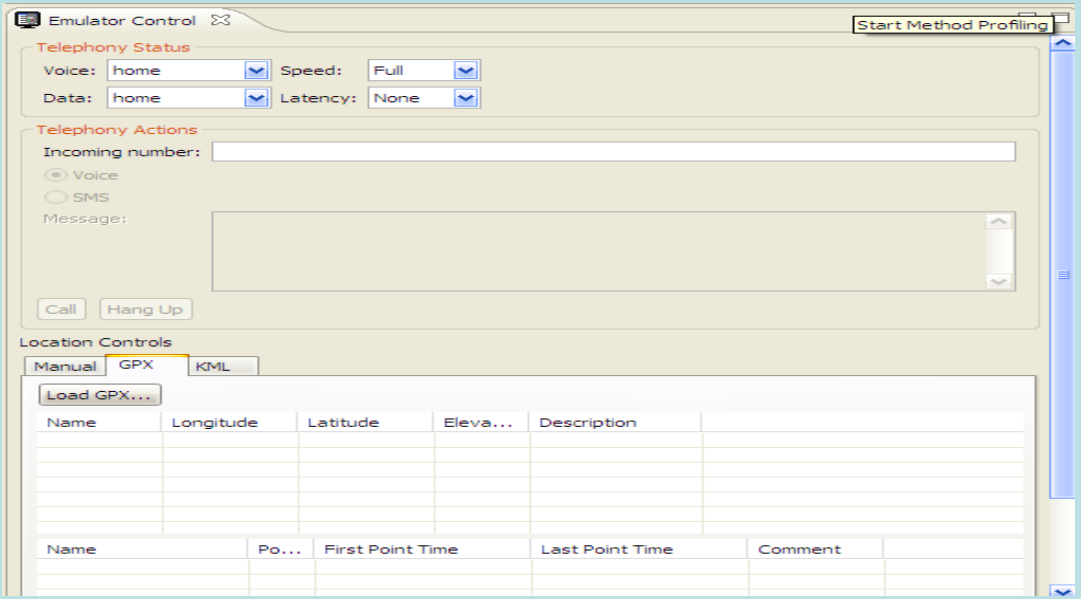
emulator control也是非常重要的，通过它可以像手机发送短信， 打电话，更新手机位置信息。

- Telephony Status: 通过选项模拟语音质量以及信号连接模式。
- Telephony Actions: 模拟电话接听和发送SMS到测试终端。
- Location Control: 模拟地理坐标或者模拟动态的路线坐标变化并显示预设的地理标识，可以通过以下3种方式：

  Manual： 手动为终端发送二维经纬坐标。

  GPX： 通过GPX文件导入序列动态变化地理坐标，从而模拟行进中GPS变化的数值。

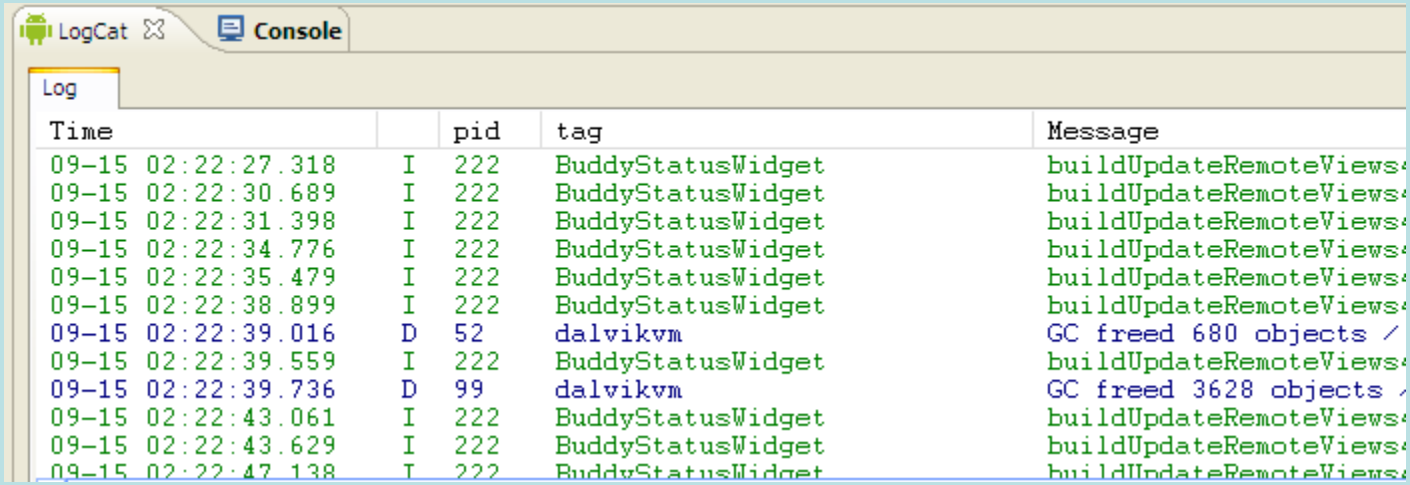
  KML： 通过KML文件导入独特的地理标识，并以动态形式根据变化的地理坐标显示在测试终端



DDMS的简介与使用

4. DDMS的使用——Logcat

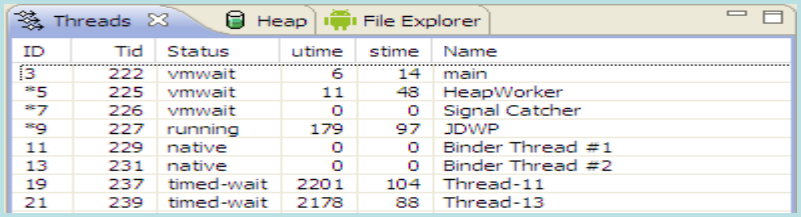
- LogCat：显示输出的调试信息,即在程序中加入的Log日志。
- Console（控制台）：是Android模拟器输出的信息，加载程序等信息；



DDMS的简介与使用

5. DDMS的使用——信息查看

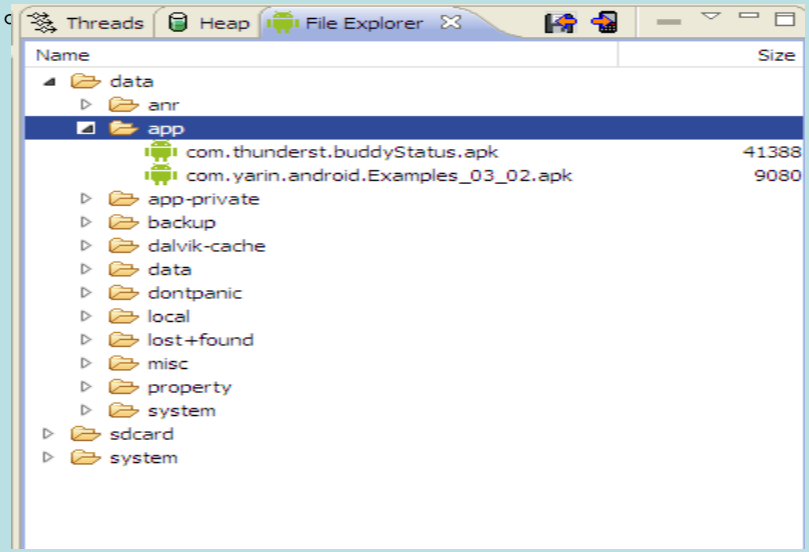
- 右边窗口中有threads， heap ， file explorer选项卡。分别显示线程统计信息，栈信息，以及android的文件系统。



ID	Tid	Status	utime	stime	Name
3	222	vmwait	6	14	main
*5	225	vmwait	11	48	HeapWorker
*7	226	vmwait	0	0	Signal Catcher
*9	227	running	179	97	JDWP
11	229	native	0	0	Binder Thread #1
13	231	native	0	0	Binder Thread #2
19	237	timed-wait	2201	104	Thread-11
21	239	timed-wait	2178	88	Thread-13

- file explorer非常有用，他可以把文件上传到android手机，或者从手机下载下来，也可以进行删除操作。

  选中file explorer选项卡后，按下面三个按钮便可实现对android手机文件系统的上传，下载，删除操作。



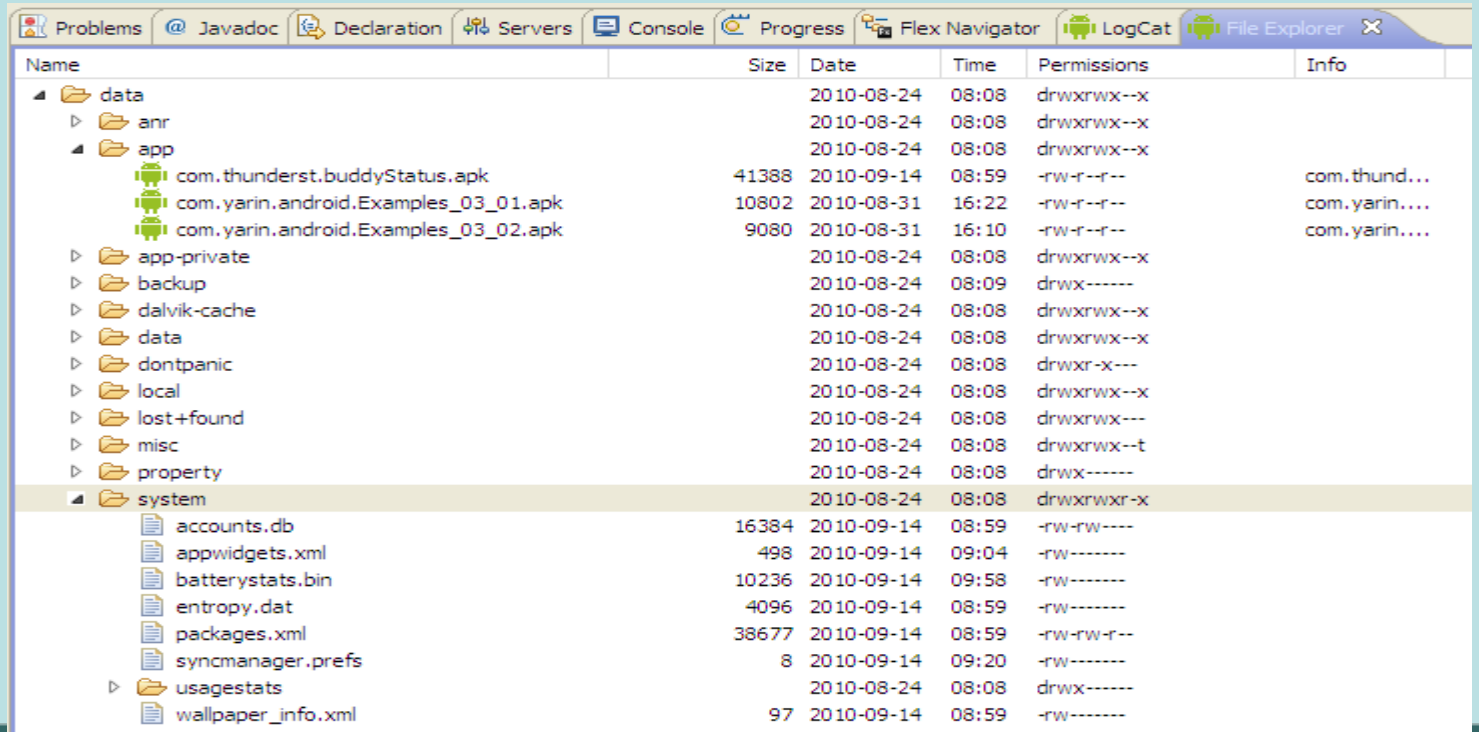
9. apk的安装与卸载

1. apk的安装：

   制作好apk后，通过adb install xx.apk即可安装

   或者直接Run程序，系统会自动打包成apk文件并运行

- 通过File Explorer可以查看文件目录
- apk文件放在了/data/app/目录下
- /data/system/packages.xml中增加了条记录
- 如果使用到了数据库，首次运行后/data/data下增加了个apk使用到的数据目录



Name	Size	Date	Time	Permissions	Info
data		2010-08-24	08:08	drwxrwx--x	
anr		2010-08-24	08:08	drwxrwx--x	
app		2010-08-24	08:08	drwxrwx--x	
com.thunderst.buddyStatus.apk	41388	2010-09-14	08:59	-rw-r--r--	com.thund...
com.yarin.android.Examples_03_01.apk	10802	2010-08-31	16:22	-rw-r--r--	com.yarin....
com.yarin.android.Examples_03_02.apk	9080	2010-08-31	16:10	-rw-r--r--	com.yarin....
app-private		2010-08-24	08:08	drwxrwx--x	
backup		2010-08-24	08:09	drwx------	
dalvik-cache		2010-08-24	08:08	drwxrwx--x	
data		2010-08-24	08:08	drwxrwx--x	
dontpanic		2010-08-24	08:08	drwxr-x---	
local		2010-08-24	08:08	drwxrwx--x	
lost+found		2010-08-24	08:08	drwxrwx---	
misc		2010-08-24	08:08	drwxrwx--t	
property		2010-08-24	08:08	drwx------	
system		2010-08-24	08:08	drwxrwxr-x	
accounts.db	16384	2010-09-14	08:59	-rw-rw----	
appwidgets.xml	498	2010-09-14	09:04	-rw-------	
batterystats.bin	10236	2010-09-14	09:58	-rw-------	
entropy.dat	4096	2010-09-14	08:59	-rw-------	
packages.xml	38677	2010-09-14	08:59	-rw-rw-r--	
syncmanager.prefs	8	2010-09-14	09:20	-rw-------	
usagestats		2010-08-24	08:08	drwx------	
wallpaper_info.xml	97	2010-09-14	08:59	-rw-------	

apk的安装与卸载

2. apk的卸载：

- 优雅式：通过界面 settings-->applications-->manage applications--> 找到应用-->application Info-->uninstall
- 粗暴式 直接到/data/app目录下删除apk
  adb shell
  cd /data/app
  rm xxx.apk

  不推荐这种方式，只删除了apk，如果应用还有其他的目录，如数据库目录/data/data/xxx/databases/并没有清理干净，留有隐患。
- 隐藏式：
  adb uninstall *package-name* (这个在adb的帮助文档中没有说明，但可以用)
  AndroidManifest.xml中有个必须的属性就是package，它所指定的就是package-name。