回顾

1. java 面向对象
2. android 基础

基本UI,适配器ui,滑屏组件,碎片组件,导航

IO sharedprefs sqlite

app1

1. web 前端

html+css+js+jq+jqm

1. android 高级

异步处理

联网

app2 获取网络数据在移动显示处理

app3

程序组件 四大组件 后台处理消息机制

5.0以上新特性

第三方开发框架

网络框架

orm框架

第三方云平台

app4

Android 异步处理

同步:所有代码都是在一个线程执行(主线程)

异步:有一些代码是另外一个线程中执行,不在主线程中执行

android 进程/线程

进程:可以分配固定地址空间占用资源 程序都是在进程中运行 切换效率低

人民路(4车道)

线程:在一个进程中运行多个执行流程 (4车道 4个线程) 切换效率高

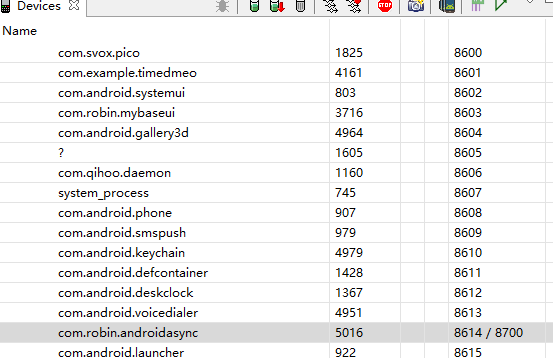
android 并发开发

1.基于java 线程来实现

2.

# android 线程

进程和线程 进程号



Pid=5016 进程号

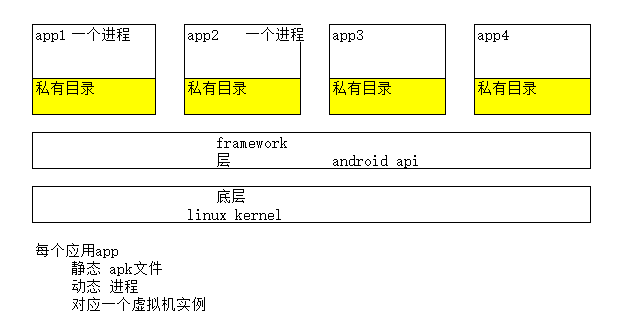
Tid=5016 主线程 tid=50

进程:android应用程序的生命周期,由线程组成.

进程之间是独立的,通常叫不同的应用.

任何一个进程内部还有一个线程没有结束,那么这个进程没有结束.

任何一个应用都有一个主线程 (永不退出,每个进程分配userid)



android 线程编程

java的并发基于线程的.

相同:基于java来定义线程的

不同:基于linux os 来实现并发,主线程不同于java主线程

android 主线程也叫UI线程,它负责处理和UI相关的事件.

Android UI不能违背两个原则:

1.当UI线程被阻塞时,超时(大约5s以上)会弹出ANR对话框.(Do not block the UI thread)

2.在UI线程以外的子线程不能访问UI对象,否则异常终止

android.view.ViewRootImpl$CalledFromWrongThreadException:

Only the original thread that created a view hierarchy can touch its views.

以上两点就是android UI线程 主线程是单线程模型的特点

## android 线程消息机制 :核心类 Handler 类

android 线程的消息机制解决线程之间通信的问题.

实现线程消息机制用到相关类有:

每个android 线程(内部基础设施 相关对象可以用来通信的)

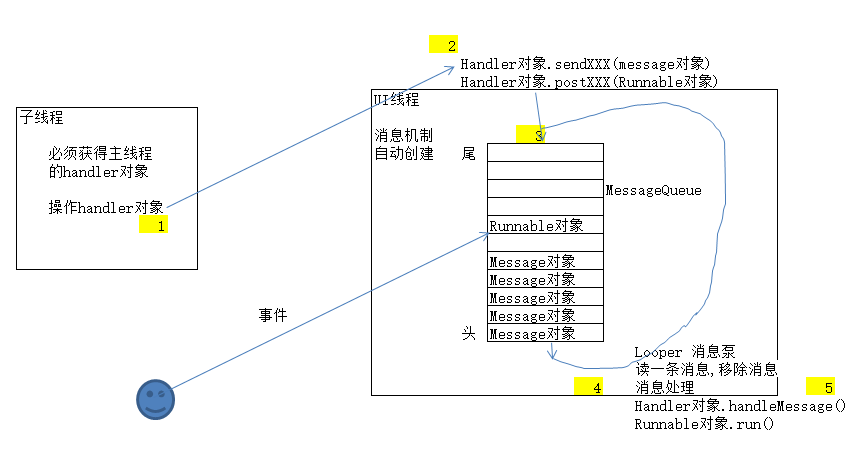
Handler

Message

MessageQueue 消息队列

Looper 消息泵

Runnable



UI线程初始化自动创建 消息队列,消息泵对象

程序员来定义Handler对象

工作原理:

1.每个线程都有一个消息队列,用来实现线程间通信,不用可忽略.

MessageQueue类

Message类

2.在线程中有一个Handler对象用来完成消息发送和消息处理.

消息发送(子线程)和消息处理(UI线程)是两个独立的部分

Handler类

3.消息的读取是由一个Looper对象用循环来执行,每读取一条消息就会回调

Handler对象的消息处理方法,并移除消息,循环读取下一条消息,直到调用

quit()退出循环为止.

子线程给UI线程发消息使用Handler步骤

1.定义 Handler子类

2.重写 handleMessage()方法,实现UI更新操作

3.创建Handler子类对象,让子线程给该对象发消息sendXXXMessage()

### 1.1.1 Handler 用途:

1.可定时或延时

启动页面

2.发消息(异步处理)

子线程异步消息机制更新Activity中的UI

//发消息对象

sendEmptyMessage(int what) 发空消息

sendMessage(Message)

sendMessageAtTime(Message, long) //在哪个时刻发

sendMessageDelayed(Message, long) //延迟多久发

//发Runnable对象

post(Runnable)

postAtTime(Runnable, long)

postDelayed(Runnable, long)

消息处理

public void handleMessage (Message msg)

获取消息

public final Message obtainMessage ()

移除消息

handler.removeMessages(456);//移除消息,但不调用消息处理方法

解释 Activity 对象原理 内部就有一个死循环

class XXXActivity extends Thread{

XXXActivity(){

//UI初始化 按钮,触屏,菜单.....

//事件处理... onXXX.....

//....

}

run(){

//死循环

while(xx){

if(hasHandler()){

//可接收各种消息....(来自于 用handler发的,也可能是用户触屏操作.....)

if(MessageQueue.getMessage()!=null){

handler.handleMessage();

}

}

}

}

onXXX(){

//....

}

}

android.text.format.Time 类似Calendar 类 效率会高些.

// 初始话一个Time对象，也可以写成:Time time = new Time("GTM+8"),即加上时区

Time time = **new** Time();

// 设置当前时间

time.setToNow();

// 获取当前时区

System.***out***.println(Time.*getCurrentTimezone*());

// 输出当前日期

System.***out***.println(time.year + "年" + time.month + "月" + time.monthDay

+ "日" + time.hour + "时" + time.minute + "分" + time.second + "秒"

+ ":现在是一年中的第" + time.yearDay + "天");

// 获取当前是一年中的第几周

System.***out***.println("现在是一年中的第" + time.getWeekNumber() + "周");

// 格式话输出当前日期、时间

System.***out***.println(time.format("%Y-%m-%d %H:%M:%S"));

System.***out***.println(time.format("%Y%m%dT%H%M%S"));// 与time.format2445相同

Studio导入工程修改的gradle设置成自己的版本:

project gradle:

buildscript {  
 repositories {  
 jcenter()  
 }  
 dependencies {  
 classpath **'com.android.tools.build:gradle:1.3.0'** *// NOTE: Do not place your application dependencies here; they belong  
 // in the individual module build.gradle files* }  
}

module app gradle文件

android {  
 compileSdkVersion 23  
 buildToolsVersion **"23.0.2"** defaultConfig {  
 applicationId **"com.robin.myasynctask"** minSdkVersion 15  
 targetSdkVersion 23  
 versionCode 1  
 versionName **"1.0"** }  
 buildTypes {  
 release {  
 minifyEnabled **false** proguardFiles getDefaultProguardFile(**'proguard-android.txt'**), **'proguard-rules.pro'** }  
 }  
}  
  
dependencies {  
 compile fileTree(dir: **'libs'**, include: [**'\*.jar'**])  
 testCompile **'junit:junit:4.12'** compile **'com.android.support:appcompat-v7:23.4.0'**}

# 2.异步处理

## 2.1 AsyncTask 异步处理核心类

### 2.1.1 异步任务处理类

异步:把代码放在另外一个线程中来执行异步操作

同步:代码都在ui线程上操作

用异步处理核心类为了简化handler操作

java.lang.Object

android.os.AsyncTask<Params, Progress, Result> 泛型类

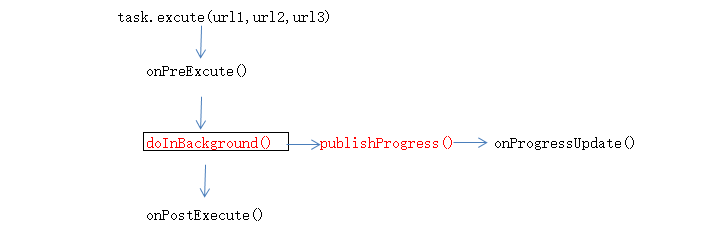
Params 输入参数的类型 URL(网址)

Progress 更新值的类型 Integer(下载进度值)

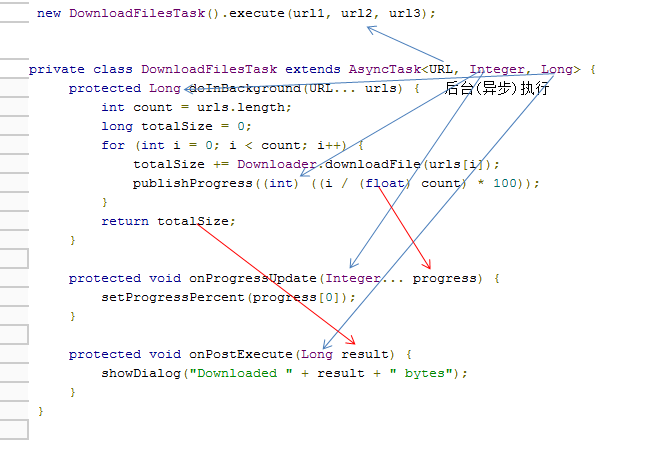
Result 操作完成值的类型 Long(下载文件多大)

作用:不用线程和handler就可实现前台(UI线程)和后台(子线程)的操作,并实现他们之间的通信.

调用顺序



参数对应关系



### 2.1.2步骤

1.定义AsynTask子类

2.重写方法 4个方法

onPreExecute() 前台处理,在调用doInBackground() 之前

doInBackground() 后台处理 (异步处理) onPreExecute()之后处理

在做后台处理时可调用publishProgress(Progress...) 告诉前台更新值的

onProgressUpdate() 前台处理,在调用publishProgress(Progress...)之后

onPostExecute() 前台处理 在doInBackground() 执行完成后

3.用该子类构造对象并调用execute()方法

private class AsyncBarTask extends AsyncTask<Void, Integer, Void>{

@Override

protected Void doInBackground(Void... params) {

// TODO Auto-generated method stub

return null;

}

@Override

protected void onProgressUpdate(Integer... values) {

// TODO Auto-generated method stub

super.onProgressUpdate(values);

}

@Override

protected void onPostExecute(Void result) {

// TODO Auto-generated method stub

super.onPostExecute(result);

}

}

Void foo(Integer … values){

Values[0]=>1

Value[1]=>2

}

Foo(1,2,3,4,5);

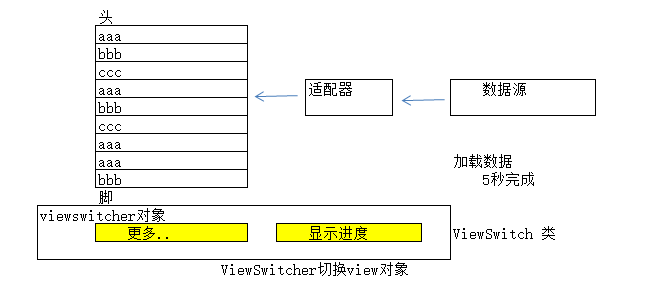
ViewSwitcher 类 切换View对象的开关

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [java.lang.Object](file:///D:\zhongl\2016work\android\docs4.0\docs\reference\java\lang\Object.html) | | | | | |
| **↳** | [android.view.View](file:///D:\zhongl\2016work\android\docs4.0\docs\reference\android\view\View.html) | | | | |
|  | **↳** | [android.view.ViewGroup](file:///D:\zhongl\2016work\android\docs4.0\docs\reference\android\view\ViewGroup.html) | | | |
|  |  | **↳** | [android.widget.FrameLayout](file:///D:\zhongl\2016work\android\docs4.0\docs\reference\android\widget\FrameLayout.html) | | |
|  |  |  | **↳** | [android.widget.ViewAnimator](file:///D:\zhongl\2016work\android\docs4.0\docs\reference\android\widget\ViewAnimator.html) | |
|  |  |  |  | **↳** | android.widget.ViewSwitche |

AddView()

ShowNext()

showPrevious() 切换View对象



2day

回顾

1. android 进程和线程
2. android UI线程 主线程 不能两个原则
3. android 消息机制 定义Handler 子类
4. Handler 用途
5. AsyncTask 简化handler操作

# 3.异步操作中简化用法:

## 3.1 Activity.runOnUiThread(Runnable)

## 3.2View.post(Runnable)

## 3.3View.postDelayed(Runnable, long)

类似 handler.postxxx()

# 4 .android 网络编程介绍

## 4.1网络通信基本技术:

移动互联网 3G,4G,5G...(广域网)

WLAN 局域网 wifi

蓝牙短距离 NFC

socket通信

最多web通信 是web网 http协议

常用作应用客户端

## 4.2 api工具类

1.java.net包 (java 编程用的api socket通信,URL通信 重点)

2.org.apache.http包 (核心HttpClient类 了解)

3.android.net包 (辅助类 了解)

4.网络框架 volley 重点

## 4.3 java.net包

socket通信

TCP:面向连接 慢 采用一种顺序流的无差错控制的数据流 基于字节的特殊IO流(InputStream/OutputStream) 用得最广泛

类似打电话

UDP:无连接 快 有限制 数据包(报)封装数据 基于报文 响应快速的实时场景

类似发短信

socket 编程分为两种

流式socket 基于TCP实现

数据报式socket 基于UDP实现

基于TCP socket通信 两个核心类

ServerSocket 此类实现服务器套接字。

Socket 此类实现客户端套接字（也可以就叫“套接字”）。

InetAddress 此类表示互联网协议 (IP) 地址。 获得主机名,ip信息

InetSocketAddress 此类实现 IP 套接字地址（IP 地址 + 端口号）。

基于UDP socket通信

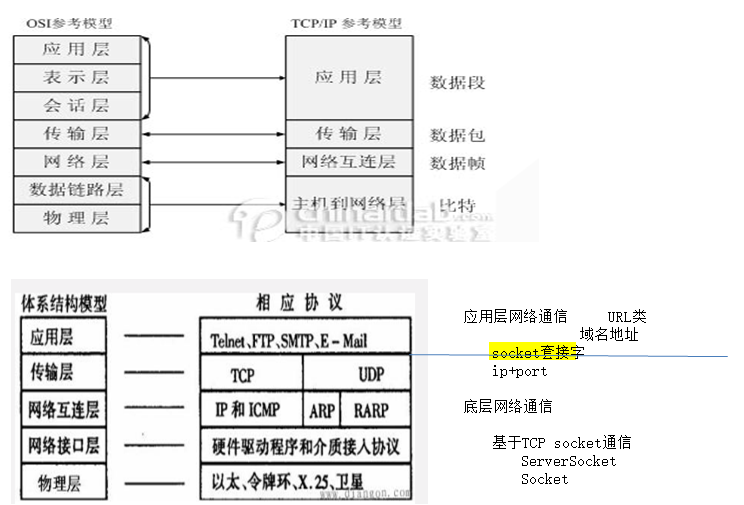
DatagramPacket 此类表示数据报包。

DatagramSocket 此类表示用来发送和接收数据报包的套接字。

与应用层通信用的

URL 类 URL 代表一个统一资源定位符，它是指向互联网“资源”的指针。

HttpURLConnection 支持 HTTP 特定功能的 URLConnection。



### 4.3.1 基于TCP socket通信

ServerSocket 此类实现服务器套接字。

端口:0-1023 系统占用

构造方法

ServerSocket(int port)

方法

Socket accept() 监听,在连接传入之前一直阻塞(线程暂停).

Socket 此类实现客户端套接字（也可以就叫“套接字”）。

构造方法

Socket(String host, int port) host:主机名/ip

方法

OutputStream getOutputStream()

InputStream getInputStream()

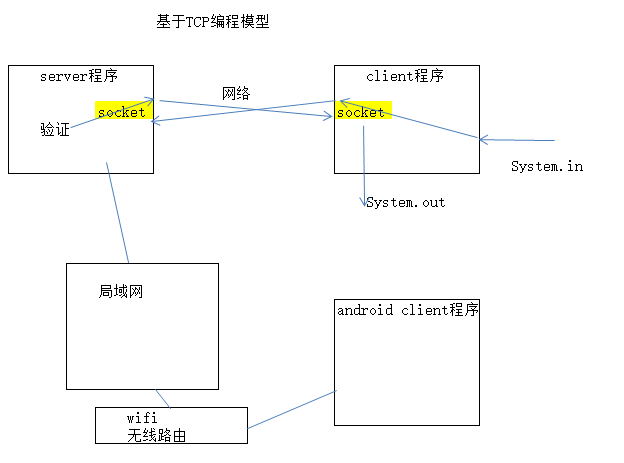
很好适配任何流.

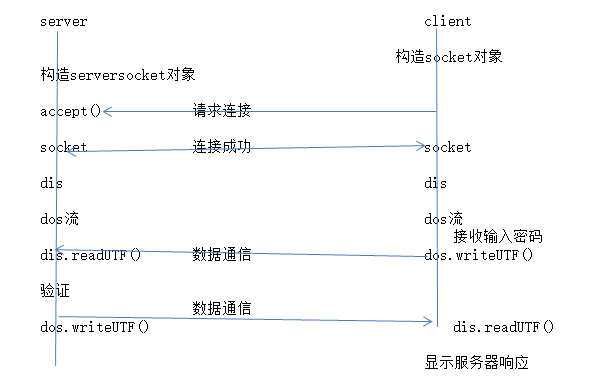
DataInputStream / DataOutputStream

readXXX() / writeXXX()

代码可能会阻塞的点

1. 创建socket连接
2. 读网络数据





TCP编程 阻塞式api

server

主线程 子线程

client

主线程

### 4. 2.1 Android Socket/ServerSocket 通信

网络api使用

使用网络必须加权限 :

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

api版本也在变

2.x

3.x 全部采用异步访问网络否则会出现异常

4.x

5.x

java.lang.RuntimeException: Unable to start activity ComponentInfo{com.robin/com.robin.Main}:

android.os.NetworkOnMainThreadException

在主线程中使用网络会产生异常.

记住一个app如果在主线程中有网络请求操作,将会抛出该异常.

网络请求会产生阻塞主线程会ANR, 3.x以后要求网络操作必须异步请求.

解决办法:

1.StrictMode 类 强制在主线程中使用 (不推荐)

2.异步操作

Handler 框架代码

Handler handler=new Handler(){

void handleMessage(Message msg){

if(msg.what==100)

//连接网络成功.

if(msg.what=-1)

//连接失败

}

}

onCreate(){

//主线程代码

//connection();

new Thread(){

public void run(){

connection();

handler.sendEmptyMessage(100);

}

}.start();

}

connection(){

//网络请求可能会阻塞的代码

}

Android 网络编程必须做到两点:

1. 记得加权限
2. 记得将网络带有阻塞代码放到异步操作中去执行.

如何用android 客户端程序连接java tcp server?

必须知道:

1. ip=? port=911
2. 对方用的流的类型

DataInputStream DataOuputStream

readUTF() writeUTF()

3.怎么通信?

server端 用读client数据的线程来实现的 ReadThread

client 端

9patch文件

普通的图片文件经过

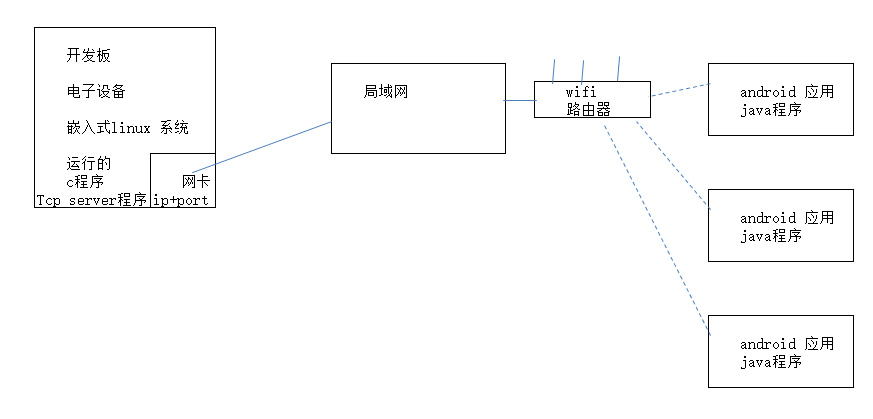
D:\zhongl\soft\android-sdk6\android-sdk\tools 下draw9patch.bat 运行对该图片进行处理

生成.9文件 可实现图片拉伸,局部保持不变形



案例

用android端应用去控制开发板设备



3day

回顾

android 网络编程

tcp socket通信(底层网络通信)

1. 加网络权限
2. 网络操作异步执行,否则产生异常

server.accept()

dis.readUTF()

new Socket()

HTTP协议用在web网络中通信,访问网络资源(绝大部分web资源)

特点:

1.http建立在TCP之上的通信协议

2.不可靠连接(短连接) http 无状态 不可靠连接

用java.net包api来实现

### 4.2.2 URL通信

URL代表访问互联网资源的地址

对于web资源表示成:

URL 可选择指定一个“端口”，它是用于建立到远程主机 TCP 连接的端口号。如果未指定该端口号，则使用协议默认的端口。例如，http 协议的默认端口为 80。还可以指定一个备用端口，如下所示：

http://www.socs.uts.edu.au:80/MosaicDocs-old/url-primer.html

<http://192.168.7.244:8080/webdata2/images/hgmm.jpeg>

URLConnection 协议无关

HttpURLConnection (支持 HTTP 特定功能的 URLConnection 和http协议相关的)

做基本socket通信

访问web资源

http协议 无状态不可靠连接 请求/响应机制

使用方法:

String actionUrl=”<http://....>”;//网址

// 定义 URL 实例

URL uri = new URL(actionUrl);//网址封装成URL对象

URLConnection con=uri.openConnection(); //协议无关连接对象

// 定义 HttpURLConnection 实例， 打开连接 访问web资源

HttpURLConnection conn = (HttpURLConnection)uri.openConnection();

//http协议连接对象 请求/响应头信息

// 设置从主机读取数据超时（ 单位： 毫秒）

conn.setReadTimeout(5 \* 1000);

// 设置允许输入

conn.setDoInput(true);

// 设置允许输出

conn.setDoOutput(true);

// 设置不允许使用缓存

conn.setUseCaches(false);

// 设置为 POST 发送方法

conn.setRequestMethod("POST");

// 设置维持长连接

conn.setRequestProperty("connection", "keep-alive");

// 设置文件字符集为 UTF-8

conn.setRequestProperty("Charsert", "UTF-8");

//发起连接请求 (可能阻塞)

conn.connect()

// 得到响应码

int res = conn.getResponseCode();

InputStream in = null;

// 上传成功则会返回响应码 200

// 200 成功 404 找不到url 500 web 后台内部错误

if (res == 200) {

// 读取数据

try {

InputStream in = new BufferedInputStream(urlConnection.getInputStream());

readStream(in);

finally {

urlConnection.disconnect();//断开连接

}

注意：如果URL传入的地址中含有中文，要对含有的中文进行URL编码，然后再拼接成地址，否则

传递数据会出现中文乱码问题，解决方式如下：

String param1 = URLEncoder.encode(中文);

请求方法

GET请求 请求参数附加url ?key1=value1&key2=value2

post请求

json解析专用网站: http://www.sojson.com/

访问web资源

1. 图片,媒体文件 二进制
2. 常见解析的数据

Json 数据

Xml 数据

Html 数据

检查更新功能的实现

1. 进入shell

nox\_adb shell

cd sdcard

ls myapp.apk

rm myapp.apk

Apk文件生成的流程

Appmodule下 gradle配置中设置版本信息:

defaultConfig {  
 applicationId **"com.robin.androidnet"** minSdkVersion 15  
 targetSdkVersion 23  
 versionCode 2  
 versionName **"2.0"**}

<http://192.168.7.10:8080/webdata2/GetJsonData?req=one> 菜单json数据

apkUrl=**"http://192.168.7.10:8080/webdata2/apk/app2.apk"**; 下载apk文件

verUrl=**"http://192.168.7.10:8080/webdata2/GetVer"**;版本信息的json数据

http://192.168.7.10:8080/webdata2/images/hgmm.jpeg

代码分析:

doUpgrade()

checkVersion();

new DownLoadTask(this, new DownLoadTask.CallBackListener() {

@Override //异步任务结束时才能调用

public void onResponse(String string) {

startLoadApk();//异步获取网络输入流

}).execute(verUrl);

startLoadApk()

new LoadStreamTask(this, new LoadStreamTask.CallBackListener() {

@Override

public void onResponse(InputStream inputStream) {

startCopyApk(inputStream);

}

}).execute(apkUrl);

startCopyApk(inputStream);

new CopyApkTask(this).execute(inputStream);

onPostExecute(){

//自动安装

}

## 4.3 org.apache.http包

核心接口:HttpClient

实现http客户端,具有浏览器的基本功能,但不能对页面进行解析和显示.

可实现http连接管理.

对java.net包再封装功能更为强大和简便.

Google已不再采用android studio 使用方法见

http://stackoverflow.com/questions/32153318/httpclient-wont-import-in-android-studio