# LoonAndroid 说明文档白皮书

---- 2014年-5月23日

作者：lic

源程序编写工程师:panc

技术翻译支持：panc

写在文档之前：此文档根据当前编写日期来编写，后期源程序更改将持续跟进。此文档仅限公司内部交流使用，如有其他人拿做其它商业用途产生其它影响，本人概不负责。程序如有雷同，纯属虚构。

程序介绍：程序大体拥有几大常用模块的封装，包括网络请求集成框架，控件View注解，数据库常用方法集成，图片下载控件，通用适配器，文件下载，通知集成，ListView下拉刷新等等。其中功能还在持续完善添加中 @panc ， 此程序在本公司最终解释权归panc 所有，

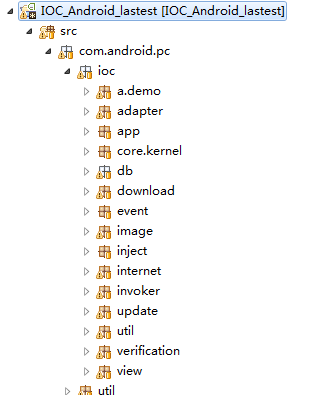
版本维护信息请咨询 @pan 或发送邮件 panc@bluemobi.cn。

# 1 程序集成到项目工程使用。

### 1 集成

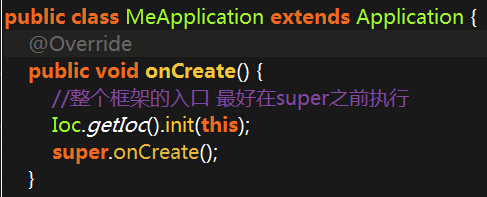
打开安卓开发集成IDE(推荐使用最常用的安卓应用开发工具 Eclipse，下面实例只针对eclipse讲解，其它工具请参见想关工具官方文档)，如下图1.1 ，将程序导入到eclipse，其中两种方式加入到自己工程项目中，1：右键-->preferences--->Android-->勾选 Is Library。然后在自己的项目中 右键-->preferences--->Android-->Add 选中Ioc\_Android，即可使用。 2：右键项目-->Export,将项目打包成jar文件(其中除了com.android.pc.ioc.a.demo包名之外，其他必须全部打包进jar)，本项目中添加到lib，build path 流程。

1.1



### 2 使用

在自己项目的Application中进行框架初始化



其中

Ioc.getIoc().init(this)

是整个框架的入口 必须在super.onCreate()之前调用

# 2 网络传输

## 2.1 网络请求

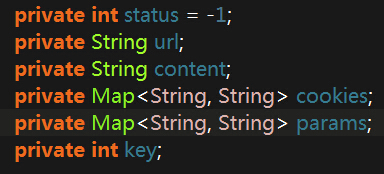
**网络请求模块分为两大部分**

一：手动判断返回状态

二: 自动判断返回状态结果

### 2.1.1 返回对象ResponseEntity

ResponseEntity这个类包含了所有返回结果相关的信息



其中每个方法都可以用get获取set进行设置

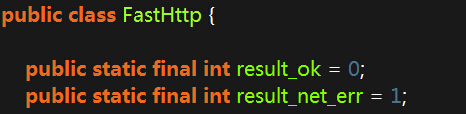
Url表示你提交的请求

Content表示返回数据的字符串

Cookies 表示从请求中获取的cookies

Params 表示你提交的参数

Status 表示请求结果的状态（主要是用来处理异常情况）其值有如下几种



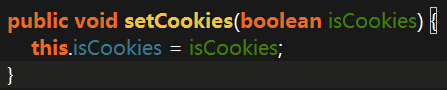
其中result\_ok表示请求成功 result\_net\_err表示请求失败 这里所有的请求 失败全部归结为网络问题

Key用来区分来源于哪个接口

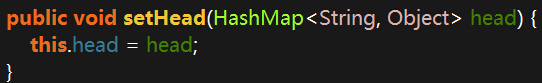
### 2.1.2 请求属性配置InternetConfig

InternetConfig这个类包含了所有请求配置相关的信息

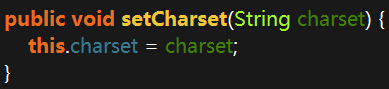
**设置是否从请求中拿取cookies设置以后可以从ResponseEntity获取cookies**



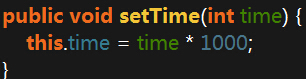
设置请求头 这些键值对会附加在请求头中传递给服务器



设置请求的编码格式

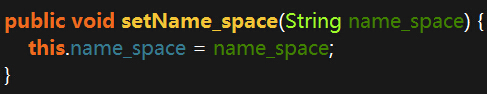


轮询模式下的轮询间隔时间 单位为秒



soap请求时候 设置的域空间 一般我们公司的soap接口无需设置

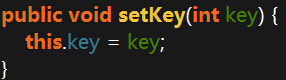
默认为http://tempuri.org/



设置连接超时时间



对于异步请求当多个请求公用一个回调的时候 用这个来进行区分 你可以从返回结果**ResponseEntity**中获得设置的这个key来判断



设置文件上传提交的进度条监听



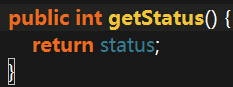
### 2.1.3 手动判断返回状态

**2.1.3.1 同步请求（get,post,soap,form表单）**

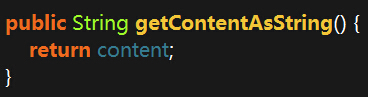
get同步请求



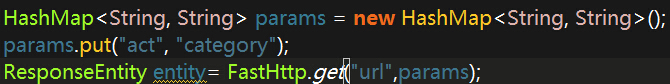
你可以从ResponseEntity中拿取



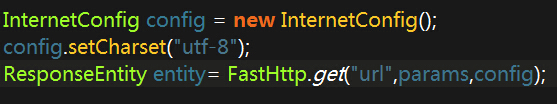
来判断请求是否成功 然后通过下面的方法拿取请求结果



参数可以以键值对传递进去



如果需要设置参数 如下



post同步请求

Post请求和get请求一样 只需要把



改为



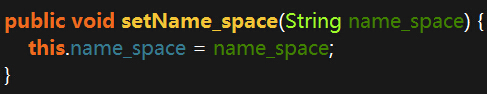
即可

soap同步请求

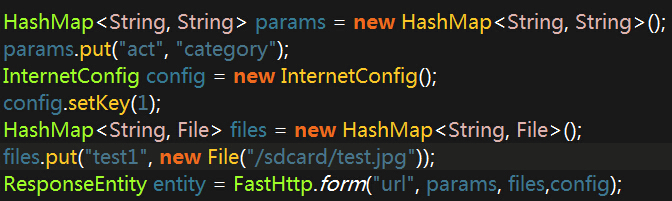


其中第一个为url ，第二个为参数（可以不设置），第三个为配置（可以不设置）第四个为请求方法

其中如果服务器有需要 请设置



表单提交



其中params为表单参数，files为上传文件键值对（可以不传），config为配置（可以不传）

如果要监控上传进度，请在**InternetConfig**中设置Progress



目前上传理论支持最大2GB文件

**2.1.3.2 异步请求（get,post,soap,form表单）**

和上面的都类似只是在所有接口前面添加了ajax

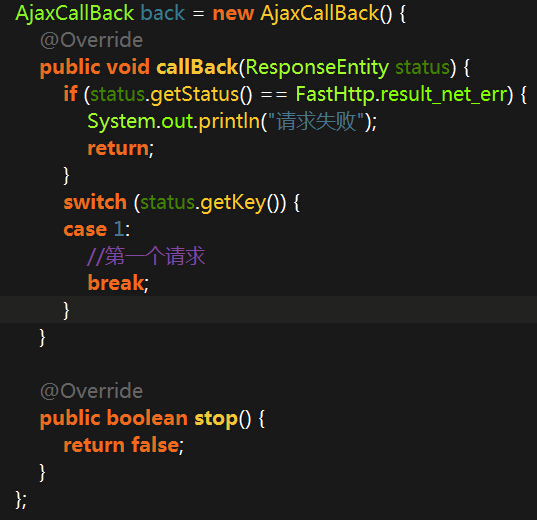
Get 为FastHttp.ajaxGet

Post 为FastHttp.ajax

Soap 为FastHttp.ajaxWebServer

Form 为FastHttp.ajaxForm

其中每个方法都多了一个参数为



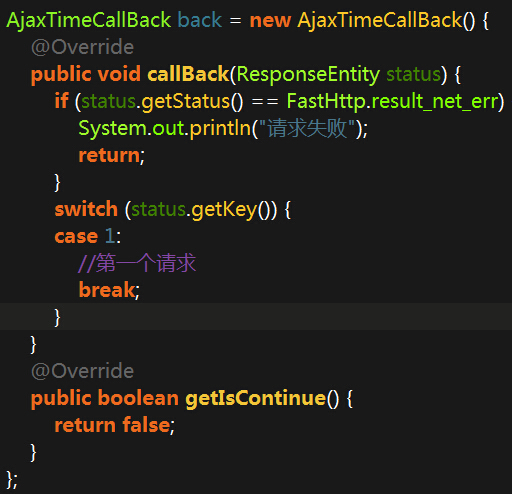
这个AjaxCallBack 是用来处理异步请求回调

其中可以获取getStatus来判断是否请求成功

其中如果多个请求公用一个AjaxCallBack ,那么getKey来判断是哪个请求

其中 stop是用来表示是否取消回调，比如在activity中 可以用 **this**.isDestroyed()来判断当前activity来判断当前页面是否被 销毁 那样就不需要回调了

其中轮询 只需要把AjaxCallBack 换成AjaxTimeCallBack即可



默认是30秒一次 如果需要 请在internetconfig中设置

GetIsContinue表示是否继续轮询，可以来终止轮询

### 2.1.4 自动判断返回状态

其核心还是调用的FastHttp的接口

如下



和FastHttp的区别在于 FastHttp改为FastHttpHander

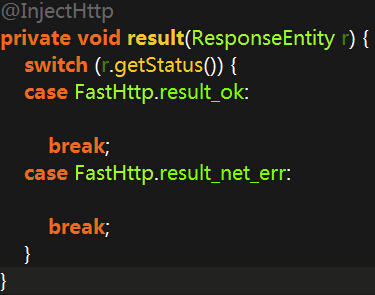
AjaxCallBack 改为this

其中AjaxCallBack 已经不需要了

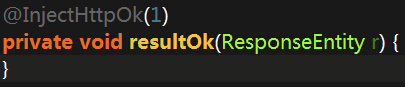
你只需要在当前类下 随便建一个参数为ResponseEntity的方法然后加上

@InjectHttp注解即可

其中InjectHttp表示请求结果总的回调



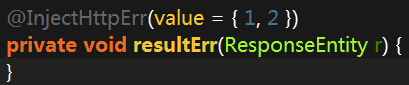
其中@InjectHttpOk表示成功的回调 其中1对应internetconfig中的setkey 如果这里不设置1 则会走@InjectHttp注解的方法



其中@InjectHttpErr表示失败的回调其中value是多个参数的回调

可以对应多个请求的不同的setkey 网络请求失败会走这个方法

如果设置的key在这里不存在 即走@InjectHttp注解的方法



返回的结果可以调用

Handler\_Json来处理结果

它的方法**JsonToBean**

表示结果转为实体类Bean

它的方法**JsonToCollection**

表示结果转为实体类HashMap

提示

如果不是发布状态 可以打印log

ApplicationBean.*logger*.d(obj.toString());

签名发布以后会自动隐藏打印，记得在签名前 请去掉自动编译

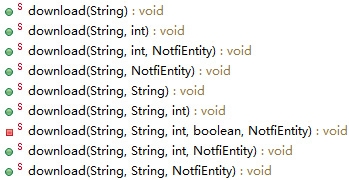
# 3下载

## 3.1说明

下载的主要类文件是FileLoaderManager，其中其他有关类文件FileResultEntity（文件下载结果类），NotfiEntity顶部通知栏所需要的实体类

## 3.2 普通下载

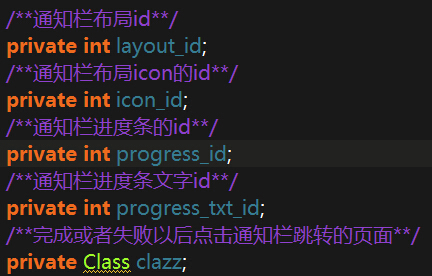
### 3.2.1 调用方法



其中第一个String为下载的url 第二个String 为文件存储的路径，可以自定义存储路径(默认状态下没有传入保存路径，程序会自动将文件保存在当前程序包路径下 　　　/files/file\_(当前时间第四位开始 long格式).\* 。即Android/data/程序包名/......路径下。)，其中Int为线程数默认为1（不设置则以单线程下载），其中NotfiEntity

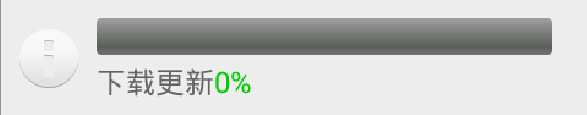
为顶部通知栏的参数，如果不传默认为null，则不会有通知栏。

### 3.2.2 NotfiEntity

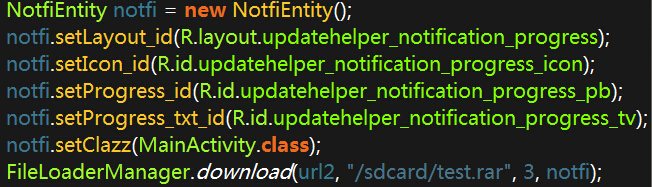


其中每个属性都有get和set方法 可以对其进行设置，其中layout\_id表示你的顶部通知栏布局

Layout,其中icon\_id,表示顶部通知栏的ICON的imagview的Id，其中progress\_id表示进度条的id，表示进度条ProgressBar的ID，progress\_txt\_id表示下载的进度显示TextView的ID，clazz表示下载完成以后点击顶部通知栏跳转的activity



实例：



### 3.2.3FileResultEntity

当我们开始下载以后，如何得知下载情况的实时回调，那么我们就要借助eventbus了

首先EventBus eventBus = EventBus.*getDefault*();

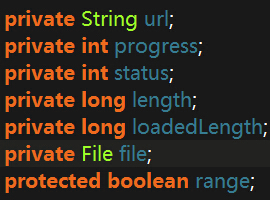
然后eventBus.register(**this**);

最后在你的当前类中加一个方法

**public** **void** **onEventMainThread**(FileResultEntity entity) {}

这个方法中你可以得到实时的下载回调事件

其中FileResultEntity参数如下



Url:下载地址

Progress:下载进度

Status:下载状态 其中状态有4种

public static final int status\_loading = 0;//下载中

public static final int status\_start = 1;//开始

public static final int status\_fail = 2;//失败

public static final int status\_sucess = 3;//成功

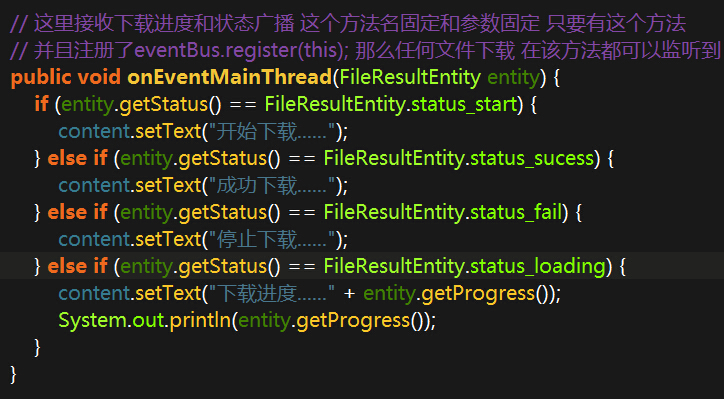
Length：下载总共大小

LoadedLength：已经下载了多少

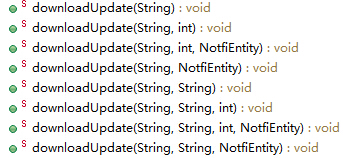
File:下载后存储的文件

Range:是否支持断点下载

实例：



## 3.3 版本更新下载



版本更新下载和普通下载调用方法以及回调基本上都一样 最主要是它下载完成以后会自动跳转到安装界面

## 3.4 其他方法

FileLoaderManager.*hideNotif*(url);关闭顶部通知栏

FileLoaderManager.*showNotif*(url, NotfiEntity);显示顶部通知栏

FileLoaderManager.*stop*(url);停止下载

FileLoaderManager.*getAllDownload*();获取所有下载记录

FileLoaderManager.*getAllFailureDownload*();失败下载记录

FileLoaderManager.*getAllFinishDownload*();成功下载记录

FileLoaderManager

**setProxyServer 设置代理的IP**

**setProxyPort设置代理的端口**

**setUseProxy设置是否使用代理**

**setProxyUser 设置代理的用户名**

**setProxyPassword 设置代理的密码**

# 4 数据库操作

## 1 数据库操作

数据库操作请参照下面示例：

DbUtils db **=** DbUtils**.**create**(this);**

User user **=** **new** User**();** *//这里需要注意的是User对象必须有id属性，或者有通过@ID注解的属性*

user**.**setEmail**(**"wyouflf@qq.com"**);**

user**.**setName**(**"wyouflf"**);**

db**.**save**(**user**);** *// 使用saveBindingId保存实体时会为实体的id赋值*

**...**

*// 查找*

Parent entity **=** db**.**findById**(**Parent**.**class**,** parent**.**getId**());**

Parent entity **=** db**.**findFirst**(**entity**);***//通过entity的属性查找*

List**<**Parent**>** list **=** db**.**findAll**(**entity**);***//通过entity的属性查找*

List**<**Parent**>** list **=** db**.**findAll**(**Parent**.**class**);***//通过类型查找*

Parent Parent **=** db**.**findFirst**(**Selector**.**from**(**Parent**.**class**).**where**(**"name"**,**"="**,**"test"**));**

*// IS NULL*

Parent Parent **=** db**.**findFirst**(**Selector**.**from**(**Parent**.**class**).**where**(**"name"**,**"="**,** **null));**

*// IS NOT NULL*

Parent Parent **=** db**.**findFirst**(**Selector**.**from**(**Parent**.**class**).**where**(**"name"**,**"!="**,** **null));**

*// WHERE id<54 AND (age>20 OR age<30) ORDER BY id LIMIT pageSize OFFSET pageOffset*

List**<**Parent**>** list **=** db**.**findAll**(**Selector**.**from**(**Parent**.**class**)**

**.**where**(**"id" **,**"<"**,** 54**)**

**.**and**(**WhereBuilder**.**b**(**"age"**,** ">"**,** 20**).**or**(**"age"**,** " < "**,** 30**))**

**.**orderBy**(**"id"**)**

**.**limit**(**pageSize**)**

**.**offset**(**pageSize **\*** pageIndex**));**

*// op为"in"时，最后一个参数必须是数组或Iterable的实现类(例如List等)*

Parent test **=** db**.**findFirst**(**Selector**.**from**(**Parent**.**class**).**where**(**"id"**,** "in"**,** **new** **int[]{**1**,** 2**,** 3**}));**

*// op为"between"时，最后一个参数必须是数组或Iterable的实现类(例如List等)*

Parent test **=** db**.**findFirst**(**Selector**.**from**(**Parent**.**class**).**where**(**"id"**,** "between"**,** **new** String**[]{**"1"**,** "5"**}));**

DbModel dbModel **=** db**.**findDbModelAll**(**Selector**.**from**(**Parent**.**class**).**select**(**"name"**));***//select("name")只取出name列*

List**<**DbModel**>** dbModels **=** db**.**findDbModelAll**(**Selector**.**from**(**Parent**.**class**).**groupBy**(**"name"**).**select**(**"name"**,** "count(name)"**));**

**...**

List**<**DbModel**>** dbModels **=** db**.**findDbModelAll**(**sql**);** *// 自定义sql查询*

db**.**execNonQuery**(**sql**)** *// 执行自定义sql*

**...**

## 2 混淆时注意事项

# 5 图片控件下载

## 1 图片控件使用

//目前未添加 参考demo

## 2 图片下载在 ListView等适配器中的使用

//目前未添加 参考demo

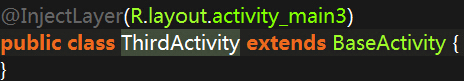
参见 LazyAdapter 类的使用。

# 6 注解



## 1 Activity注解

第一种



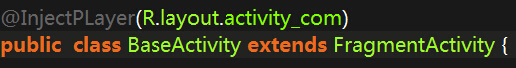
其中R.layout.activity\_main3表示当前activity使用的布局

InjectLayer只针对activity进行注解

第二种 其中value和上面一样 parent表示当前界面是一个子view 它具有和其它页面共用的组件



如果设置了parent 那么在父类中 设置的injectPlayer中必须含有ID为parent的组件



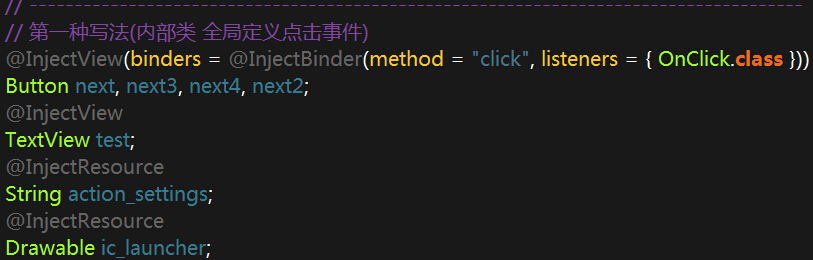
## 2 普通View注解

其中参数说明 id对应layout中的组件的id，binders 是一个数组，表示view绑定的事件

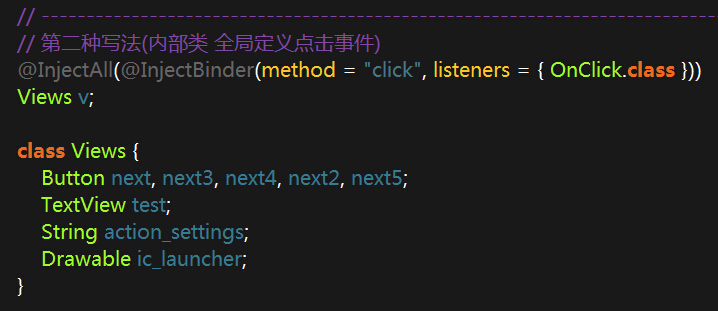
Method=”click”表示这个事件会调用当前类中的click方法，该方法可以有参数也可以无参数，如果有参数则必须是View类型，具体参考demo，其中listeners表示这个事件是一个什么事件

具体可以参考FirstFragment类

### ① 第一种写法(内部类 全局定义点击事件)



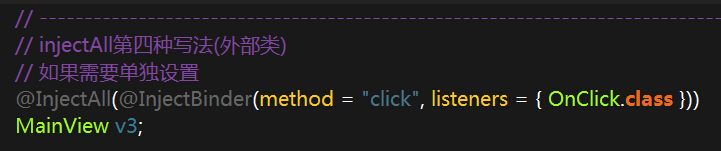
### ② 第二种写法(注解与定义分开，全局点击)



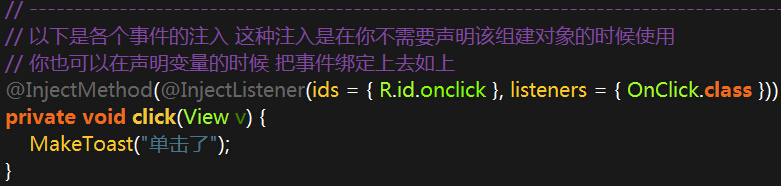
### ③ 第三种写法(内部类 单独设置点击事件 injectAll)



### ④ 第四种写法(内部类 单独设置点击事件 injectAll)



### 第五种写法



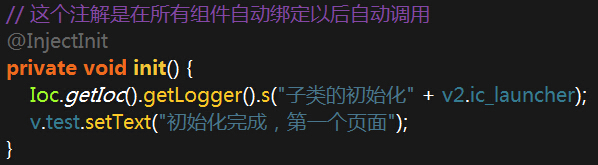
其中OnClick.class为public class OnClick extends OnListener implements OnClickListener {}自定义类。

## 3 生命周期注解

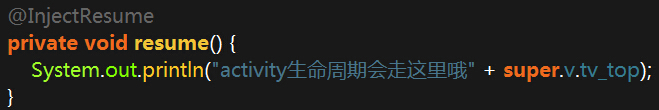
//执行在oncreat之前



// 这个注解是在所有组件自动绑定以后自动调用 即OnCreat方法里所有内容执行完后才执行。

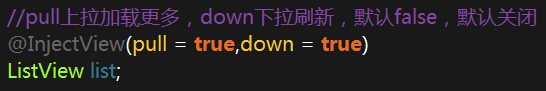


//OnResume注解



只要是activity的生命周期 除了onDestroy没有对应的注解意外 其他的生命周期都有其注解比如injectpasue等等

## 4 上拉加载更多下拉刷新注解



必须为listview,必须加上如上注解

其中PullToRefreshManager为刷新和加载更多设置管理类

如下

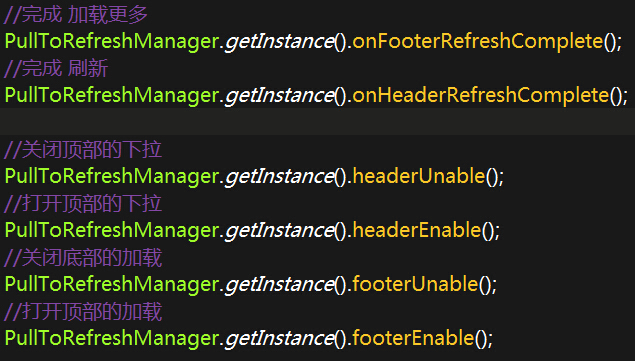


具体可以设置的属性如下



都有相应的set方法

状态改变方法

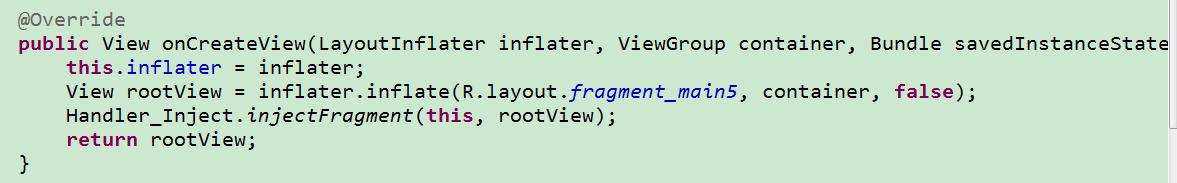


下拉和上拉的回调方法



## 5 FragMent 注解

需要在OnCreateView里添加注解，如下图



其中控件的属性注解部分请参照 FifthFragment