**客户名字和logo**

餐厅订餐系统-客户端详细设计文档

[项目ID]

wang linglong

日期：2015-04-05

文档版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修订日期 | 修订人 | 审核人 | 变更内容 |
| V0.1 | 2014/11/30 | Wang LingLong |  | 初始化 |
| V1.0 | 2014/12/8 | Wang LingLong |  | 补充详细内容 |
| V1.1 | 2014/12/8 | Wang LingLong |  | 更新类图和时序图 |
| V1.2 | 2014/12/27 | Wang LingLong |  | 更新反馈的问题 |
| V1.3 | 2015/1/3 | Wang LingLong |  | 增加Android开发环境搭建和目录结构说明 |
| V1.4 | 2015/1/11 | Wang LingLong |  | 更新部分类图 |
| V1.5 | 2015/1/31 | Wang LingLong |  | 更新文档格式和内容 |
|  |  |  |  |  |

**目录**

[1. 文档介绍 1](#_Toc410739234)

[1.1 目的 1](#_Toc410739235)

[1.2 范围 1](#_Toc410739236)

[1.3 缩写词列表 1](#_Toc410739237)

[1.4 参考内容 1](#_Toc410739238)

[2. 开发环境配置 2](#_Toc410739239)

[2.1 下载及安装JDK 2](#_Toc410739240)

[2.2 下载及安装Android环境 2](#_Toc410739241)

[3. 项目目录结构说明 4](#_Toc410739242)

[4. 移动点菜模块 5](#_Toc410739243)

[4.1 菜品浏览 5](#_Toc410739244)

[4.1.1 功能IPO图 5](#_Toc410739245)

[4.1.2 时序图 6](#_Toc410739246)

[4.1.3 代码构成 7](#_Toc410739247)

[4.1.4 类图 7](#_Toc410739248)

[4.2 点菜 7](#_Toc410739249)

[4.2.1 功能IPO图 7](#_Toc410739250)

[4.2.2 时序图 8](#_Toc410739251)

[4.2.3 代码构成 8](#_Toc410739252)

[4.2.4 类图 9](#_Toc410739253)

[4.3 下单 9](#_Toc410739254)

[4.3.1 功能IPO图 9](#_Toc410739255)

[4.3.2 时序图 10](#_Toc410739256)

[4.3.3 代码构成 10](#_Toc410739257)

[4.3.4 类图 10](#_Toc410739258)

[4.4 历史订单查询 11](#_Toc410739259)

[4.4.1 功能IPO图 11](#_Toc410739260)

[4.4.2 时序图 11](#_Toc410739261)

[4.4.3 代码构成 11](#_Toc410739262)

[4.4.4 类图 12](#_Toc410739263)

[5. 存储数据设计 12](#_Toc410739264)

[5.1 应用设置 12](#_Toc410739265)

[5.1.1 Android常用存储方式说明 12](#_Toc410739266)

[5.1.2 本例实现方式说明 13](#_Toc410739267)

[6. 数据缓存设计 13](#_Toc410739268)

[6.1 功能描述 13](#_Toc410739269)

[6.2 实现方式 15](#_Toc410739270)

[7. 数据管理器 15](#_Toc410739271)

[7.1 功能描述 15](#_Toc410739272)

[7.2 类图 16](#_Toc410739273)

[7.3 函数说明 16](#_Toc410739274)

[7.4 接口 17](#_Toc410739275)

[8. 数据下载服务 18](#_Toc410739276)

[8.1 功能描述 18](#_Toc410739277)

[8.2 类图 18](#_Toc410739278)

[8.3 函数说明 18](#_Toc410739279)

[8.4 接口 20](#_Toc410739280)

# 

# 文档介绍

## 目的

本文描述餐厅订餐系统客户端详细设计，用于指导开发人员进行系统开发。

## 范围

本文档主要指导客户端开发人员进行编码工作，并供测试人员等项目其他人员参考。

## 缩写词列表

## 参考内容

# 开发环境配置

餐厅订餐系统使用Java/Android技术开发客户端应用，所以下面依次介绍在项目所使用到的开发环境和工具的安装配置。

## 下载及安装JDK

请参考餐厅订餐系统\_系统详细设计文档。

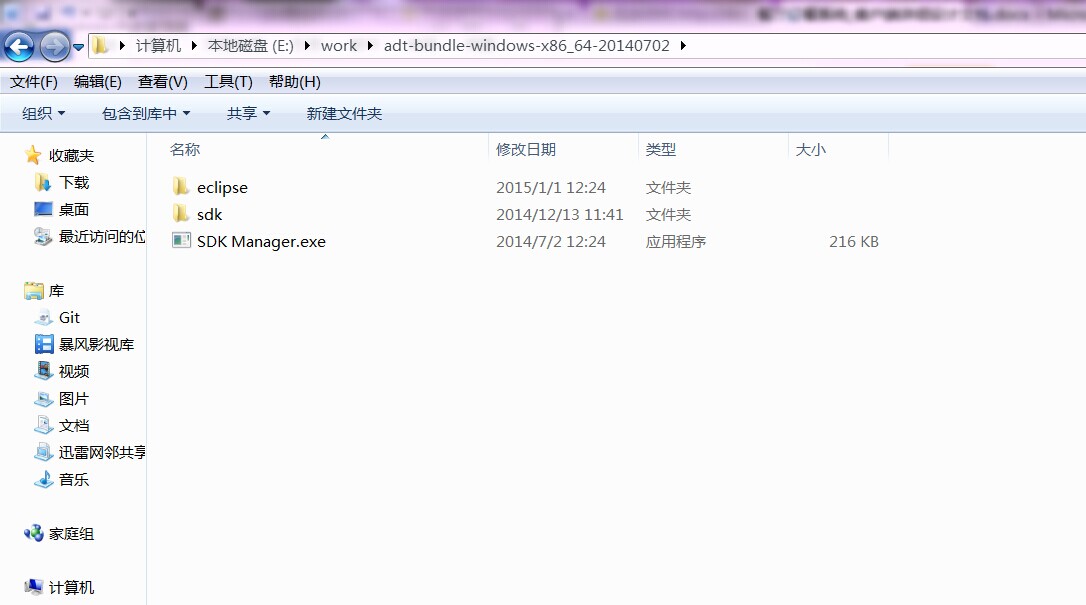
## 下载及安装Android环境

Android的开发IDE有很多，常用的有Eclipse或者Google推出不久的Android Studio，但是目前Android Studio还不是非常完善，推荐还是采用Eclipse。

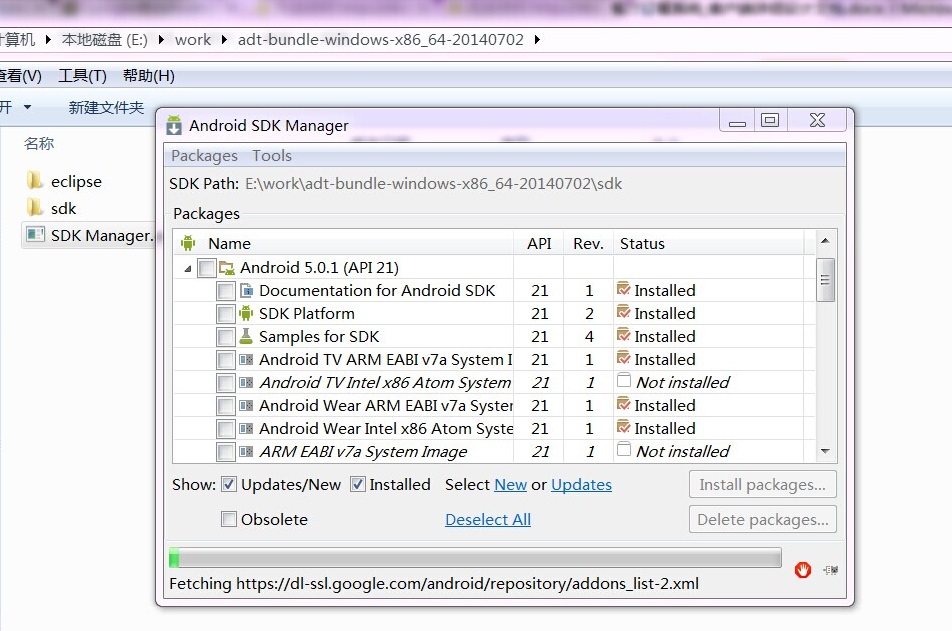
这里推荐使用官方的Eclipse集成IDE工具。Google前不久推出了自家Android版的eclipse，自动集成了Android SDK，下载解压就能直接运行，开始Android开发工作。此举是非常方便对eclipse配置Android环境感觉麻烦的人。

Android版eclipse下载地址：<http://developer.android.com/sdk/index.html>

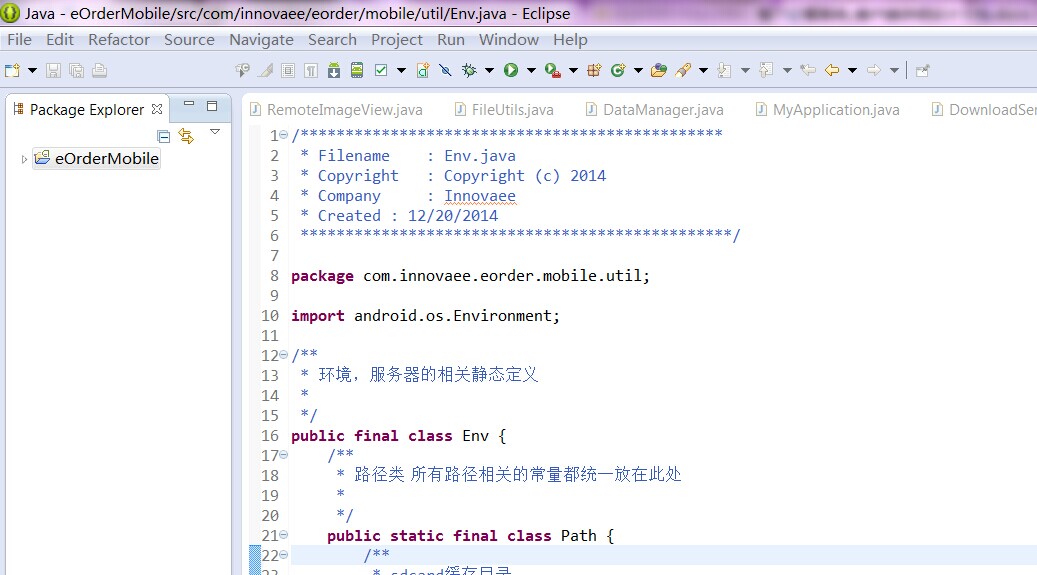
解压后是一个adt-bundle-windows文件夹，里面分别是eclipse和sdk两个文件夹，自动集成了所有需要的工具，不用再进行繁琐的配置，非常方便简单！



打开SDK Manager.exe 可以对当前已经下载和未下载的Android SDK进行管理，如图所示。

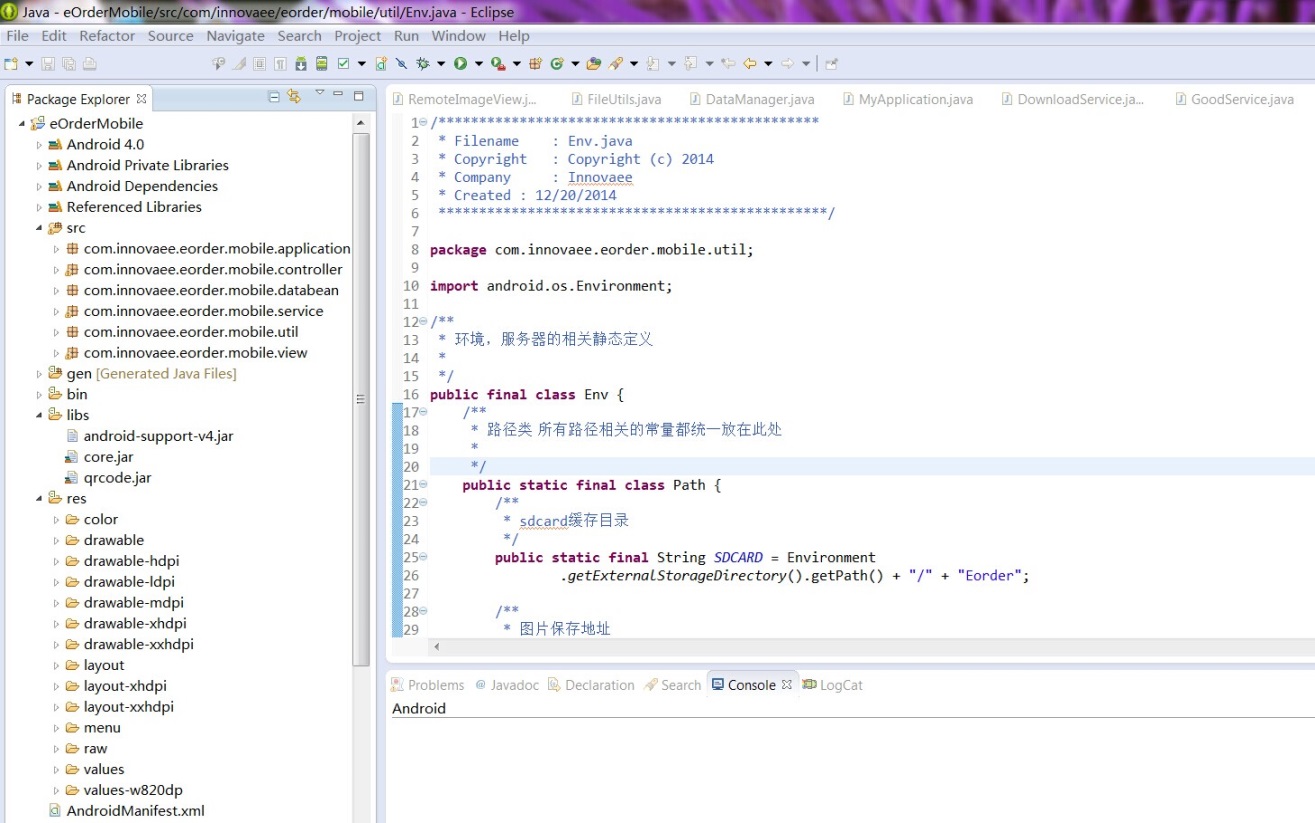


最后点击eclipse目录下eclipse.exe文件，可以打开eclipse进行Android的开发工作了。



# 项目目录结构说明

首先，我们看看Android客户端的目录结构，如下图所示。



**src:** 该目录下主要是存放Android客户端的实现代码，我们可以看到下面有很多包，在包下面就是具体实现的类；

**com.innovaee.eorder.mobile.application：**该包里面主要是MyApplication类，该类主要是为了在某些地方需要获取全局Context的地方能获取到应用的Context;

**com.innovaee.eorder.mobile.controller**：该包里面主要是DataManager类，该类为整个应用层提供相关的数据控制器，及MVC中的Controller；

**com.innovaee.eorder.mobile.databean：**该包里面主要是需要的各种数据Bean;

**com.innovaee.eorder.mobile.service：**该包里面主要是各种数据访问接口和数据下载服务，提供给Controller调用，及为MVC中的Model层；

**com.innovaee.eorder.mobile.util**：该包主要是各种工具类，提供给其他类调用；

**com.innovaee.eorder.mobile.view：**该包主要是视图相关的类，及MVC中的View层，该处将部分Activity和Adapter放到该包中，严格说这部分应该也是属于Android中的Controller，但是为了阅读方便和便于理解，所以放到View层；

**libs:**该目录主要是存放客户端需要的一些核心库文件，android-support-v4.jar,core.jar是系统自动生成，我们不用管它，qrcoder.jar是我们需要用到二维码扫描识别的核心库。

**res:** 该目录主要是存放资源文件，工程中的所有资源文件都分别放到下面目录中，下面又分了各种子目录，分别再对他们作详细介绍。

**color:** 主要是存放对颜色的定义;

**drawable:** 主要放置Xml配置文件或者对分辨率要求较低的图片;

**drawable-hdpi:** 里面存放高分辨率的图片,如WVGA (480×800),FWVGA (480×854);

**drawable-ldpi:** 里面存放低分辨率的图片,如QVGA (240×320);

**drawable-mdpi:** 里面存放中等分辨率的图片,如HVGA (320×480);

**drawable-xhdpi:** 里面存放1280×720的图片;

**drawable-xxhdpi:** 里面存放1920×1080的图片;

**layout:**该目录放一些通用布局xml文件，比如界面顶部和底部的布局，不会随着屏幕大小变化;

**layout-xhdpi:** 里面存放1280×720的布局文件;

**layout-xxhdpi:** 里面存放1920×1080的布局文件;

**menu:**存放弹出式Menu，ActionBar等相关的数据;

**raw:**存放程序中可能会需要用到的一些数据文件;

**values:**主要存放多语言字符串数据, 颜色，id，样式等等;

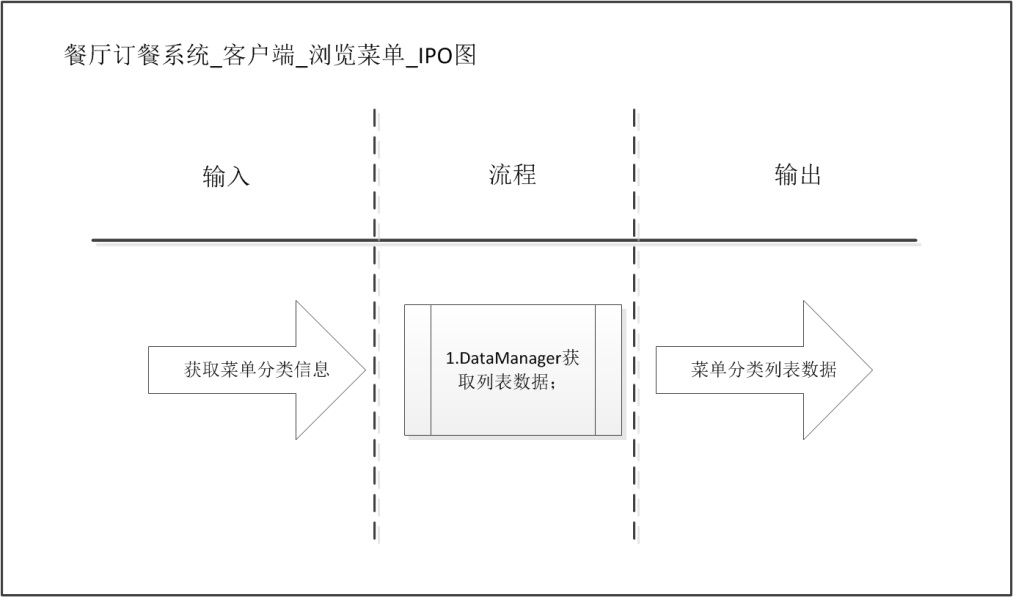
**AndroidManifest.xml：**AndroidManifest.xml 是每个android程序中必须的文件。它位于整个项目的根目录，描述了package中暴露的组件（activities, services, 等等），他们各自的实现类，各种能被处理的数据和启动位置。 除了能声明程序中的Activities, ContentProviders, Services, 和Intent Receivers,还能指定permissions和instrumentation（安全控制和测试）

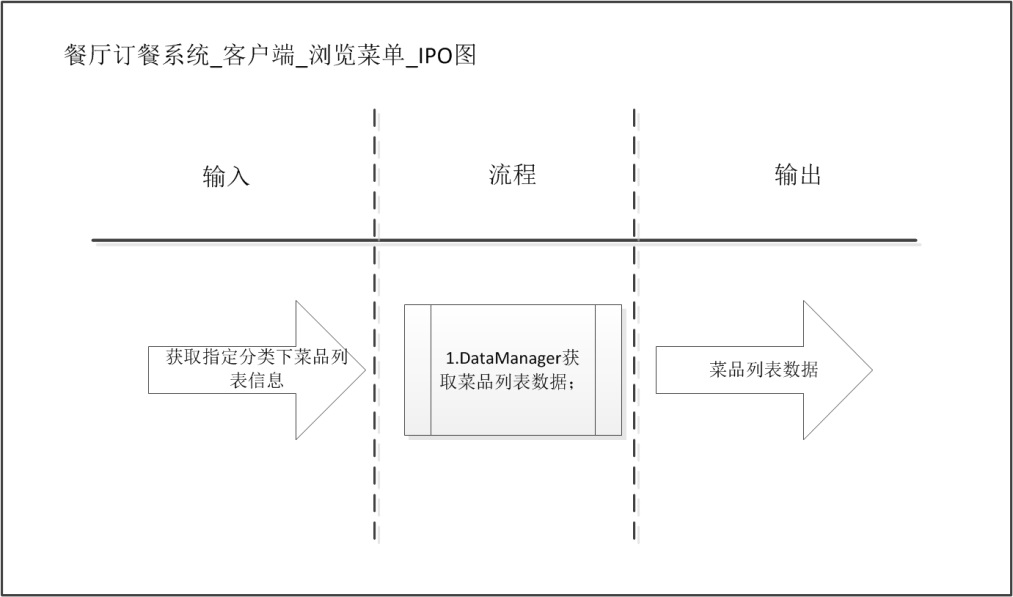
# 移动点菜模块

## 菜品浏览

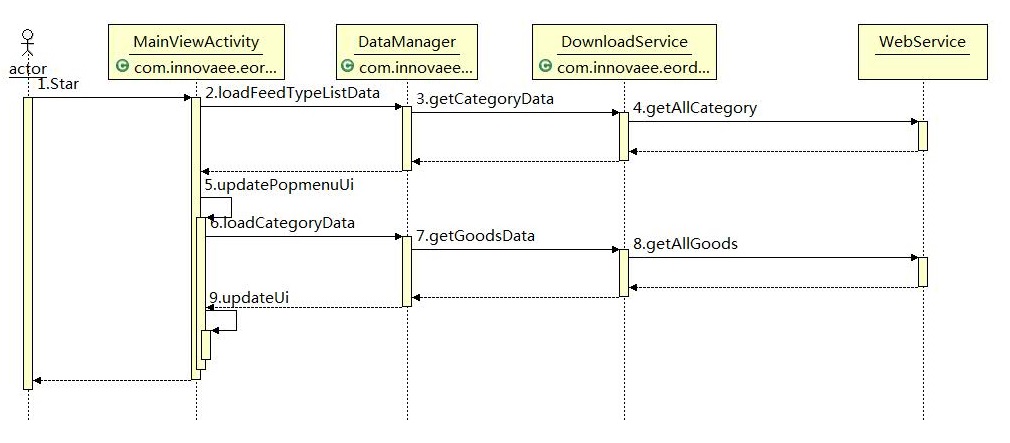
移动端发送请求到服务端请求菜品分类信息，并默认根据菜品分类第一项再次请求该分类列表菜品数据，并展示给用户浏览。需求详情参见需求规格说明书-UC\_001\_浏览菜单。

### 功能IPO图





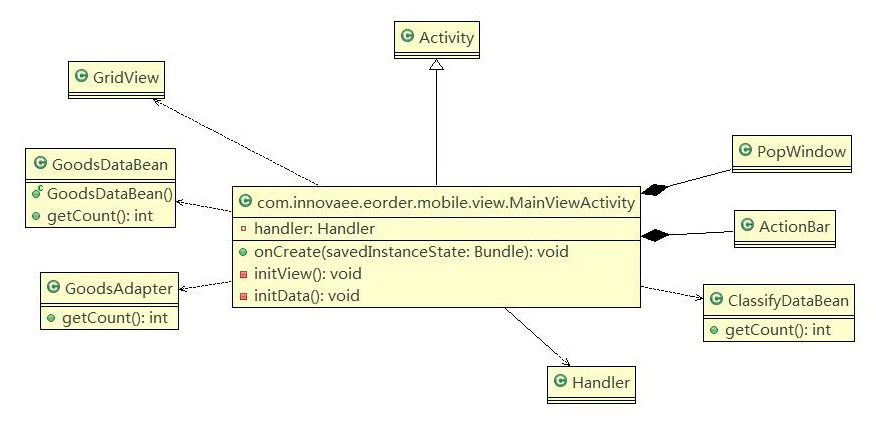
### 时序图



### 代码构成

|  |  |  |
| --- | --- | --- |
| 文件名 | 文件路径 | 文件说明 |
| MainViewActivity.java | com.innovaee.eorder.mobile.view | 浏览菜单View的显示，主要处理View和数据控制。 |
| DataManager.java | com.innovaee.eorder.mobile.controller | 对数据层读取的封装，从服务端获取相关数据。 |
| DownloadService.java | com.innovaee.eorder.mobile.service | 和服务端通讯获取相关数据的类。 |
| GoodsAdapter.java | com.innovaee.eorder.mobile.view | 菜品数据适配器，配合菜品显示容器GridView使用。 |

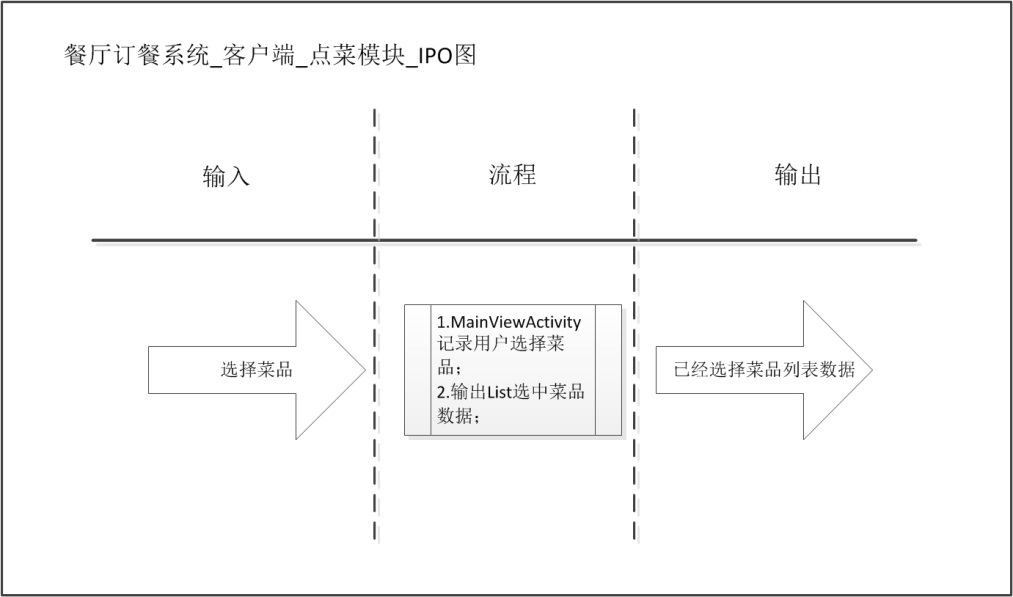
### 类图



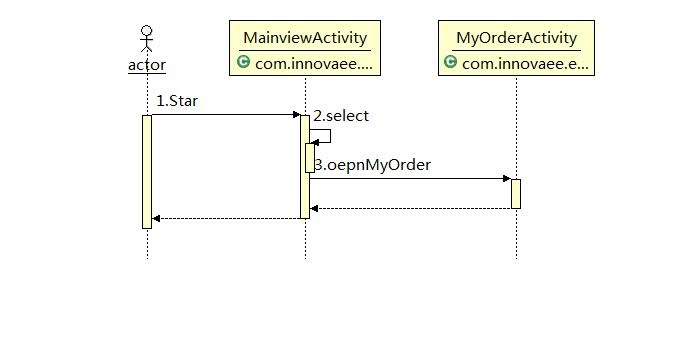
## 点菜

用户在菜品浏览界面选择菜品，并且进入我的订单页面浏览已经选中的菜品列表。需求详情参见需求规格说明书-UC\_003\_点菜。

### 功能IPO图



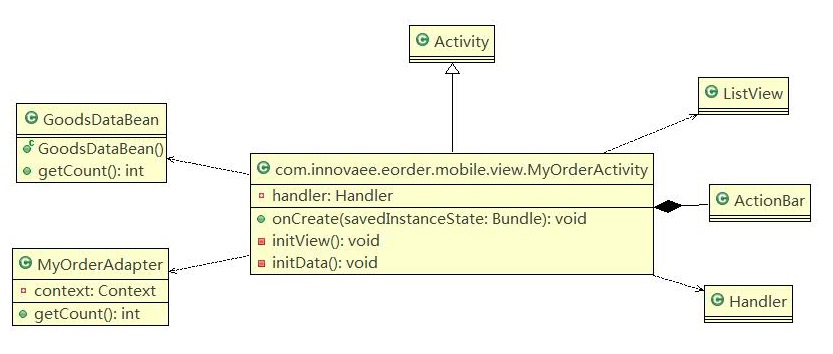
### 时序图



### 代码构成

|  |  |  |
| --- | --- | --- |
| 文件名 | 文件路径 | 文件说明 |
| MainViewActivity.java | com.innovaee.eorder.mobile.view | 浏览菜单View的显示，主要处理View和数据控制。 |
| GoodsAdapter.java | com.innovaee.eorder.mobile.view | 菜品数据适配器，配合菜品显示容器GridView使用。 |
| MyOrderActivity.java | com.innovaee.eorder.mobile.view | 我的订单View的显示，主要处理View和数据控制。 |
| MyOrderAdapter.java | com.innovaee.eorder.mobile.view | 我的订单数据适配器，配合菜品显示容器GridView使用。 |

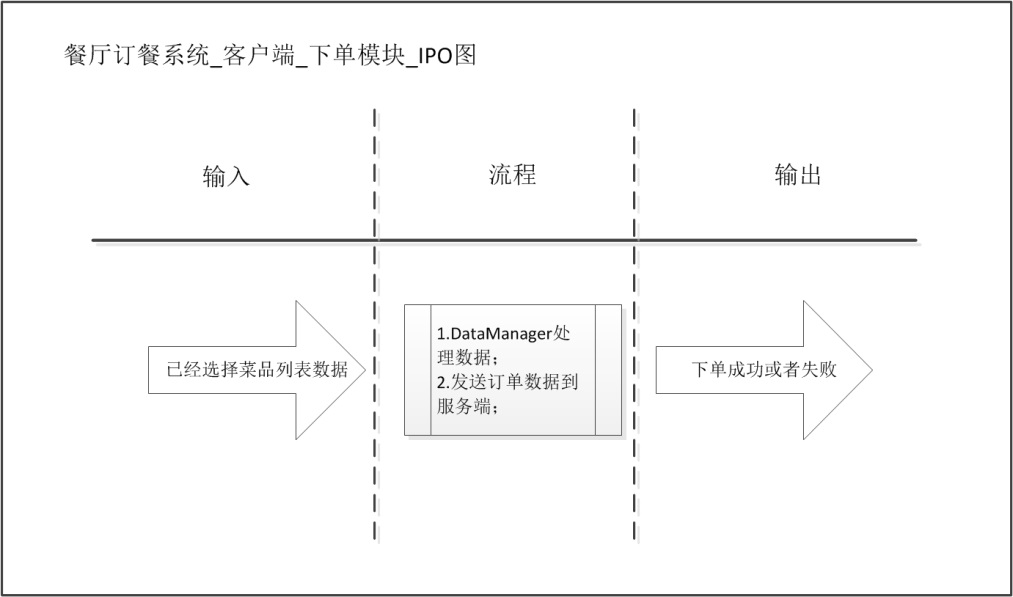
### 类图



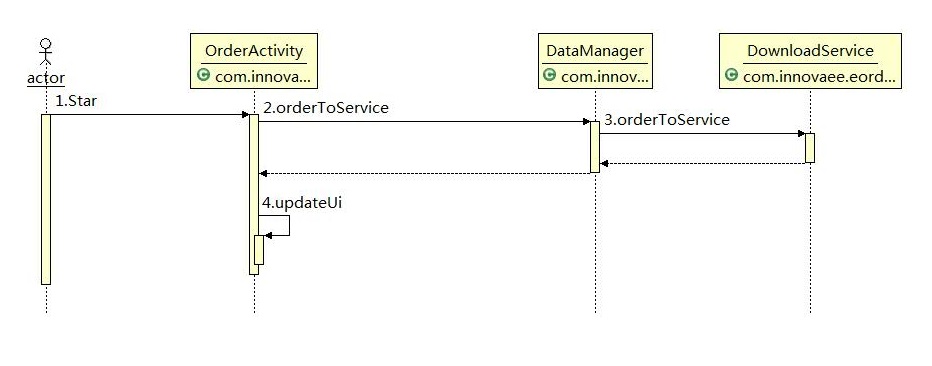
## 下单

用户选择相应的菜品，然后进入已选择菜品界面查看，可以删除不需要的菜品或者修改数量，最终选择确认下单到服务器

### 功能IPO图



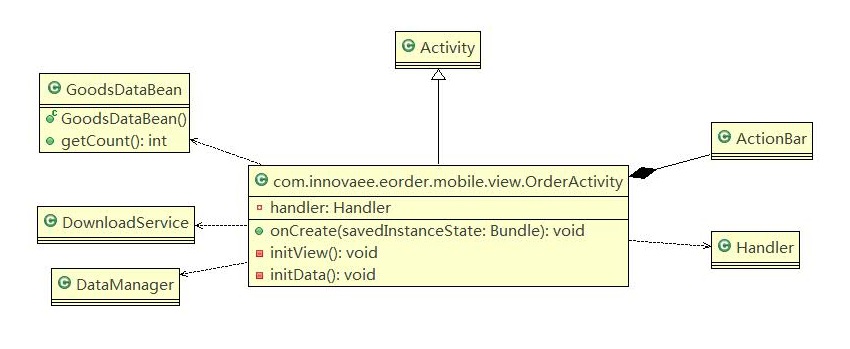
### 时序图



### 代码构成

|  |  |  |
| --- | --- | --- |
| 文件名 | 文件路径 | 文件说明 |
| OrderActivity.java | com.innovaee.eorder.mobile.view | 订单View的显示，主要处理View和数据控制。 |
| DataManager.java | com.innovaee.eorder.mobile.controller | 对数据层读取的封装，从服务端获取相关数据。 |
| DownloadService.java | com.innovaee.eorder.mobile.service | 和服务端通讯获取相关数据的类。 |
| OrderAdapter.java | com.innovaee.eorder.mobile.view | 菜品数据适配器，配合菜品显示容器GridView使用。 |

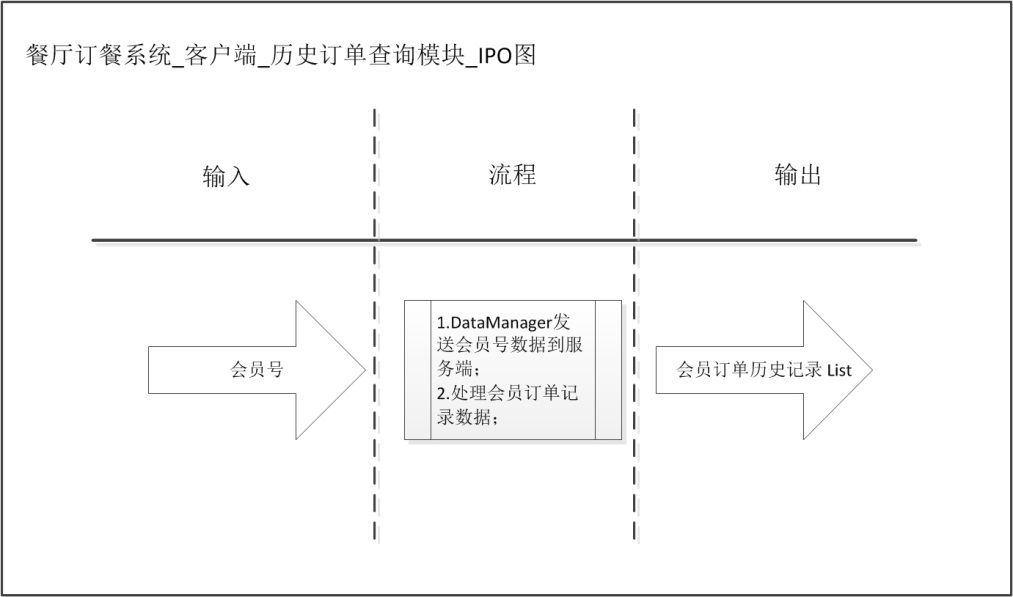
### 类图



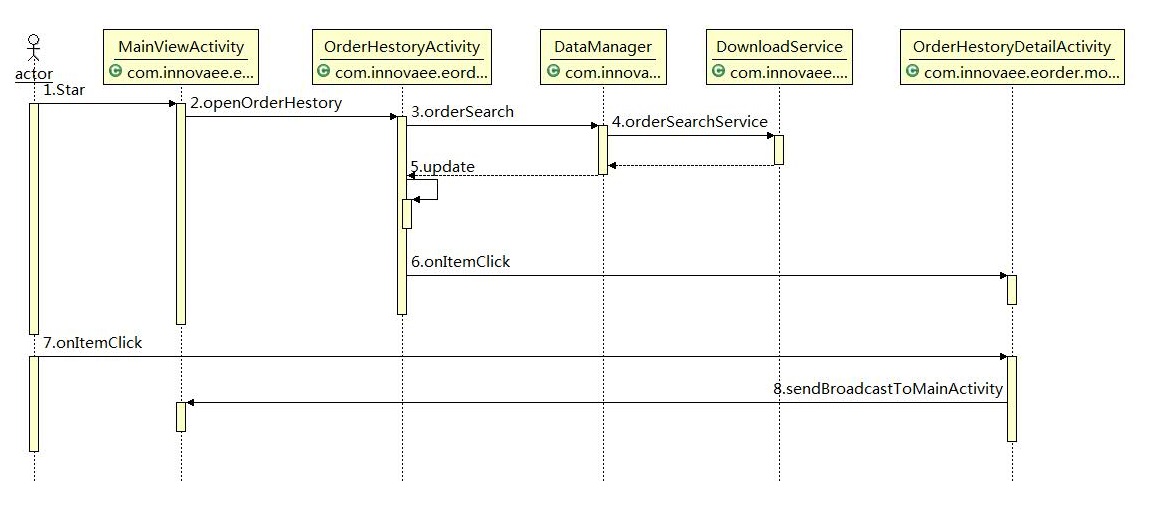
## 历史订单查询

用户可以在主界面输入会员id，并且选择查询，则可以查询到该会员id的所有历史订单记录，并且用户可以点击某个订单记录查看详情。需求详情参见需求规格说明书- UC\_002\_浏览历史订单。

### 功能IPO图



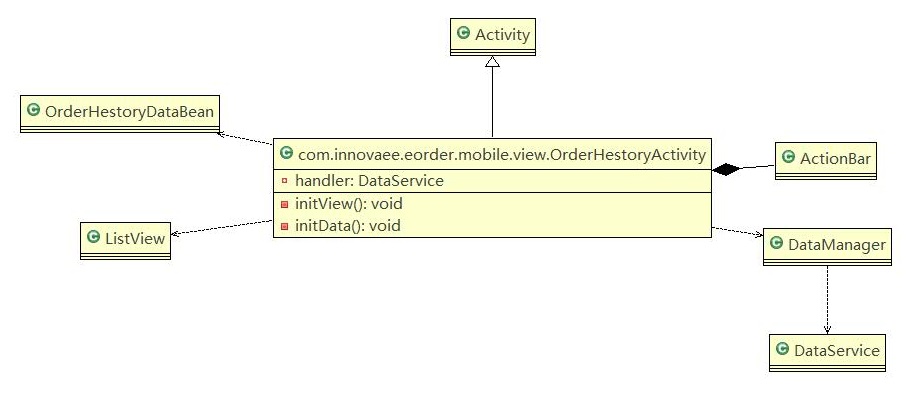
### 时序图



### 代码构成

|  |  |  |
| --- | --- | --- |
| 文件名 | 文件路径 | 文件说明 |
| MainViewActivity.java | com.innovaee.eorder.mobile.view | 浏览菜单View的显示，主要处理View和数据控制。 |
| DataManager.java | com.innovaee.eorder.mobile.controller | 对数据层读取的封装，从服务端获取相关数据。 |
| DownloadService.java | com.innovaee.eorder.mobile.service | 和服务端通讯获取相关数据的类。 |
| GoodsAdapter.java | com.innovaee.eorder.mobile.view | 菜品数据适配器，配合菜品显示容器GridView使用。 |
| OrderHestoryDetailActivity.java | Com.innovaee.eorder.mobile.view | 历史订单详情View，主要展示详情数据和再下单控制。 |

### 类图



# 存储数据设计

## 应用设置

### Android常用存储方式说明

在Android应用程序中，数据存储操作是必不可少的。因此，Android系统一共提供了四种数据存储方式。分别是：SharePreference、SQLite、Content Provider和File。这几种方式有着各自的特点，分别适用于不同的场景。

SQLite： SQLite是一个轻量级的数据库，支持基本SQL语法，是常被采用的一种数据存储方式。Android为此数据库提供了一个名为SQLiteDatabase的类，封装了一些操作数据库的API。

SharedPreference： 除SQLite数据库外，另一种常用的数据存储方式，其本质就是一个xml文件，常用于存储较简单的参数设置。

File： 即常说的文件（I/O）存储方法，常用语存储大数量的数据，但是缺点是更新数据将是一件困难的事情。

ContentProvider: Android系统中能实现所有应用程序共享的一种数据存储方式，由于数据通常在各应用间的是互相私密的，所以此存储方式较少使用，但是其又是必不可少的一种存储方式。例如音频，视频，图片和通讯录，一般都可以采用此种方式进行存储。每个Content Provider都会对外提供一个公共的URI（包装成Uri对象），如果应用程序有数据需要共享时，就需要使用Content Provider为这些数据定义一个URI，然后其他的应用程序就通过Content Provider传入这个URI来对数据进行操作。

而本应用主要是为了存储设置参数，参数均比较简单，所以这里比较合适的是选择SharePreference方式。

### 本例实现方式说明

本程序客户端只需要存储简单数据到手机端，比如手机的设置项等等简单参数，所以采用Android中应用比较广泛，轻量级的存储类SharedPreferences．

在Android系统中，这些信息以XML文件的形式保存在 /data/data/　PACKAGE\_NAME /shared\_prefs 目录下。

存储实现方式：

1. 打开Preferences，名称为setting，如果存在则打开它，否则创建新的Preferences

SharedPreferences settings = getSharedPreferences(“setting”, 0);

2. 让setting处于编辑状态

SharedPreferences.Editor editor = settings.edit();

3. 存放数据

editor.putString(“name”, ”xxxxx”);

editor.putString(“URL”, ”xxxxx”);

4. 完成提交

editor.commit();

存储数据读取实现方式：

1. 获取Preferences

SharedPreferences settings = getSharedPreferences(“setting”, 0);

2. 取出数据

String name = settings.getString(“name”, ”默认值”);

String url = setting.getString(“URL”, ”default”);

以上就是Android中SharedPreferences的使用方法，其中创建的Preferences文件存放位置可以在Eclipse中查看：

DDMS->File Explorer /<package name>/shared\_prefs/setting.xml

# 数据缓存设计

## 功能描述

网络应用通常会采用缓存技术，可以进一步大大缓解数据交互的压力，又能提供一定的离线浏览，缓存一般适合下面这样的应用．

1. 提供网络服务的应用

2. 数据更新不需要实时更新，哪怕是3-5分钟的延迟也是可以采用缓存机制。

3. 缓存的过期时间是可以接受的

这样所带来的好处：

1. 减小服务器的压力

2. 提高客户端的响应速度

3. 一定程度上支持离线浏览

常用的Android缓存方式有下面几种方式：

软引用方式SoftReference。

SoftReference 的特点是它的一个实例保存着一个 Java 对象的软引用，该软引用的存在不妨碍垃圾收集器线程对该 Java 对象的回收。也就是说，一旦SoftReference 保存着一个 Java 对象的软引用之后，在垃圾收集器线程对这个 Java 对象回收之前， SoftReference 类所提供的 get() 方法都会返回 这个Java 对象的强引用。另外，一旦垃圾线程回收该 Java 对象之后， get() 方法将返回 null 。

但是目前不推荐使用这种方式。从Android 2.3 (API Level 9) 开始，垃圾回收开始强制的回收掉 soft/weak 引用 从而导致这些缓存没有任何效率的提升。另外，在 Android 3.0 (API Level 11)之前，这些缓存的Bitmap数据保存在底层内存(native memory)中，并且达到预定条件后也不会释放这些对象，从而可能导致程序超过内存限制并崩溃。

使用内存缓存。

内存缓存提高了访问图片的速度，但是要占用不少内存。 LruCache类（在API 4之前可以使用Support Library 中的类 ）特别适合缓存Bitmap， 把最近使用到的Bitmap对象用强引用保存起来（保存到LinkedHashMap中），当缓存数量达到预定的值的时候，把不经常使用的对象删除。

但是使用该种方式，很明显会比较占用内存，所以使用的时候要特别小心。

使用磁盘缓存。

在访问最近使用过的图片中，内存缓存速度很快，但是您无法确定图片是否在缓存中存在。像GridView 这种控件可能具有很多图片需要显示，很快图片数据就填满了缓存容量。

同时您的程序还可能被其他任务打断，比如打进的电话 — 当您的程序位于后台的时候，系统可能会清除这些图片缓存。一旦用户恢复使用您的程序，您还需要重新处理这些图片。在这种情况下，可以使用磁盘缓存来保存这些已经处理过的图片，当这些图片在内存缓存中不可用的时候，可以从磁盘缓存中加载从而省略了图片处理过程。当然， 从磁盘载入图片要比从内存读取慢很多，并且应该在非UI线程中载入磁盘图片。

## 实现方式

本应用也采用了缓存技术，因为本客户端主要是对网络图片的读取比较频繁，需要不断的刷新和加载图片，所以保存网络下载的图片比较重要．客户端采用了对图片进行缓存的技术，结合了上面两种方式，采用了方式2和方式3结合的方式来保存图片，及将从网络端下载的图片保存到缓存和本地sdcard上，使用的时候会先去缓存获取图片，如果获取不到图片再去sdcard获取相应的图片．

实现类如下：

private static final int DEFAULT\_SIZE = 5 \* 1024 \* 1024;

/\*\*

\* SDcard storage path

\*/

private final static String STORAGE\_PATH = Environment

.getExternalStorageDirectory().getPath()

+ "/eOrderMobile";

private static LruImageCache softBitmapCache = new LruImageCache(

DEFAULT\_SIZE);

private static ImageDataManager instance = null;

……

｝

# 数据管理器

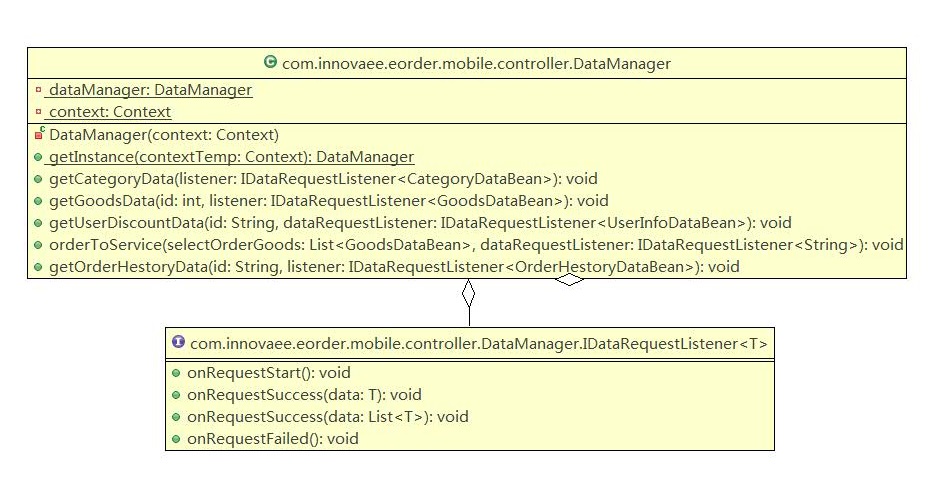
## 功能描述

数据管理器部分主要是对应用所需要的数据接口部分进行封装，提供给上层应用调用，即为MVC部分的Controller。

该类为单例类，避免在应用中重复创建。通过下面的方式调用：

DataManager.getInstance(Context).xxx

## 类图



## 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包名 | com.innovaee.eorder.mobile.controller | | | |
| 类名/接口名 | DataManager | | | |
| 父类 | N/A | | | |
| 实现接口 | N/A | | | |
| 成员变量 | | | | |
| 变量名 | 数据类型 | 说明 | | |
| self | DataManager | 单例类，只有一个实例 | | |
| context | Context | 调用应用或者Activity的context | | |
|  |  |  | | |
| 成员函数 | | | | |
| 函数名 | 返回值 | | 参数 | 说明 |
| getInstance | 无 | | Context contextTemp | 获取实例 |
| getCategoryData | 无 | | IDataRequestListener<CategoryDataBean> listener(监听器) | 获取所有分类列表 |
| getGoodsData | 无 | | IDataRequestListener<GoodsDataBean> listener(监听器) | 获取所有商品列表 |
| getUserDiscountData | 无 | | String id(会员id),  IDataRequestListener<UserInfoDataBean> iDataRequestListener(监听器) | 得到会员的折扣信息 |
| orderToService | 无 | | List<GoodsDataBean> selectOrderGoods(选择菜品),  final IDataRequestListener<String> iDataRequestListener(监听器) | 下单到服务器 |
| getOrderHestoryData | 无 | | String id(会员id),  final IDataRequestListener<OrderHestoryDataBean> listener(监听器) | 得到某个会员的历史订单记录 |

## 接口

|  |  |  |  |
| --- | --- | --- | --- |
| 包名 | com.innovaee.eorder.mobile.controller | | |
| 类名/接口名 | IDataRequestListener<T> | | |
| 父类 | N/A | | |
| 实现接口 | N/A | | |
| 成员变量 | | | |
| 变量名 | 数据类型 | 说明 | |
| 无 | 无 | 无 | |
| 成员函数 | | | |
| 函数名 | 返回值 | 参数 | 说明 |
| onRequestStart() | 无 | 无 | 数据请求开始回调接口 |
| onRequestSuccess | 无 | T data(泛型数据) | 数据请求成功回调接口 |
| onRequestSuccess | 无 | List<T> data(泛型数据List) | 数据请求成功回调接口 |
| onRequestFailed | 无 | 无 | 数据请求失败回调接口 |
|  |  |  |  |

# 数据下载服务

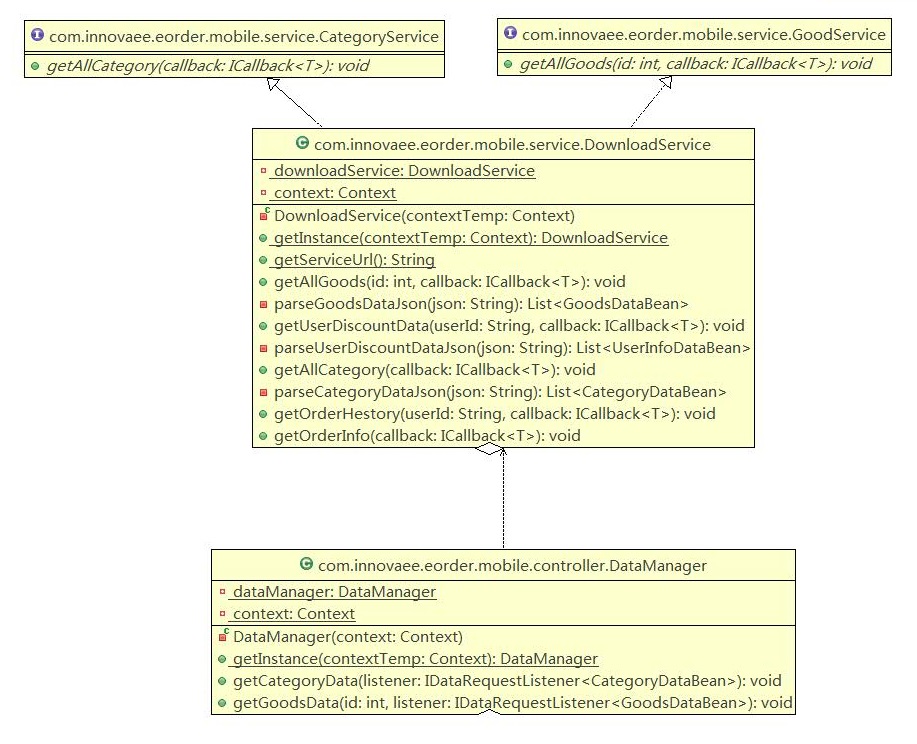
## 功能描述

数据下载服务部分主要是对数据控制器所需要的数据接口部分进行封装，提供给数据控制器调用，即为MVC部分的Model。

该类为单例类，避免在应用中重复创建。通过下面的方式调用：

DownloadService.getInstance(Context).xxx

## 类图



## 函数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 包名 | com.innovaee.eorder.mobile.service | | |
| 类名/接口名 | DownloadService | | |
| 父类 | N/A | | |
| 实现接口 | N/A | | |
| 成员变量 | | | |
| 变量名 | 数据类型 | 说明 | |
| self | DownloadService | 单例类，只有一个实例 | |
| context | Context | 调用应用或者Activity的context | |
|  |  |  | |
| 成员函数 | | | |
| 函数名 | 返回值 | 参数 | 说明 |
| getInstance | 无 | Context contextTemp(调用者的context) | 获取实例 |
| getServiceUrl | String(服务器地址) | 无 | 得到设置中的服务器地址 |
| getAllGoods | 无 | int id(分类id), ICallback<T> callback(回调接口) | 获取某分类下面商品列表信息 |
| parseGoodsDataJson | List<GoodsDataBean>(解析完的菜品列表) | String json(需要解析的数据) | 解析菜品json数组对象 |
| getUserDiscountData | 无 | String userId(会员id), ICallback<T> callback(回调接口) | 获取会员的折扣信息 |
| parseUserDiscountDataJson | List<UserInfoDataBean>(会员数据list) | String json(需要解析的数据) | 解析会员信息json数组对象 |
| getAllCategory | 无 | ICallback<T> callback(回调接口) | 获取最新的商品分类表 |
| parseCategoryDataJson | List<CategoryDataBean>(菜品分类信息list) | String json(需要解析的数据) | 解析分类json数据 |
| getOrderHestory | 无 | String userId(会员id), ICallback<T> callback(回调接口) | 获取某个会员号的历史订单记录 |
| parseOrderHestoryDataJson | List<OrderHestoryDataBean>(订单数据Bean列表list) | String json(需要解析的数据) | 解析订单历史json数据 |
| getOrderInfo | 无 | ICallback<T> callback(回调接口) | 获取某个订单详情 |
| parseOrderInfoDataJson | List<OrderInfoDataBean>(订单详情数据Bean列表list) | String json需要解析的数据 | 解析订单详情json数据 |
| postOrderInfo | 无 | TableInfoDataBean tableInfo(下单台数据信息), List<OrderInfoDataBean> dataBeanList(下单数据详情信息), ICallback<T> callback(回调函数) | 提交订单详情 |
| writeJSON | JSONObject(转换完成后的数据) | TableInfoDataBean tableInfo(下单台数据信息), List<OrderInfoDataBean> dataBeanList(下单数据详情信息), | 转换数据到json格式 |
| getBitmapUrl | String(真实的图片url地址) | String bitmapPath(服务器下发的图片地址) | 由服务器图片地址转换到真实url地址 |

## 接口

|  |  |  |  |
| --- | --- | --- | --- |
| 包名 | com.innovaee.eorder.mobile.service | | |
| 类名/接口名 | CategoryService | | |
| 父类 | N/A | | |
| 实现接口 | N/A | | |
| 成员变量 | | | |
| 变量名 | 数据类型 | 说明 | |
| 无 | 无 | 无 | |
| 成员函数 | | | |
| 函数名 | 返回值 | 参数 | 说明 |
| getAllCategory | 无 | ICallback<T> callback(回调接口) | 获取最新的分类信息 |

|  |  |  |  |
| --- | --- | --- | --- |
| 包名 | com.innovaee.eorder.mobile.service | | |
| 类名/接口名 | ICallback<T> | | |
| 父类 | N/A | | |
| 实现接口 | N/A | | |
| 成员变量 | | | |
| 变量名 | 数据类型 | 说明 | |
| 无 | 无 | 无 | |
| 成员函数 | | | |
| 函数名 | 返回值 | 参数 | 说明 |
| onStarted() | 无 | 无 | 数据请求开始回调接口 |
| onSuccess | 无 | List<T> response(泛型list数据) | 成功回调函数，返回泛型list数据 |
| onSuccessT | 无 | T response(泛型数据) | 成功回调函数，返回泛型数据 |
| onFailed | 无 | String error(错误码) | 失败回调函数，返回自定义错误码 |