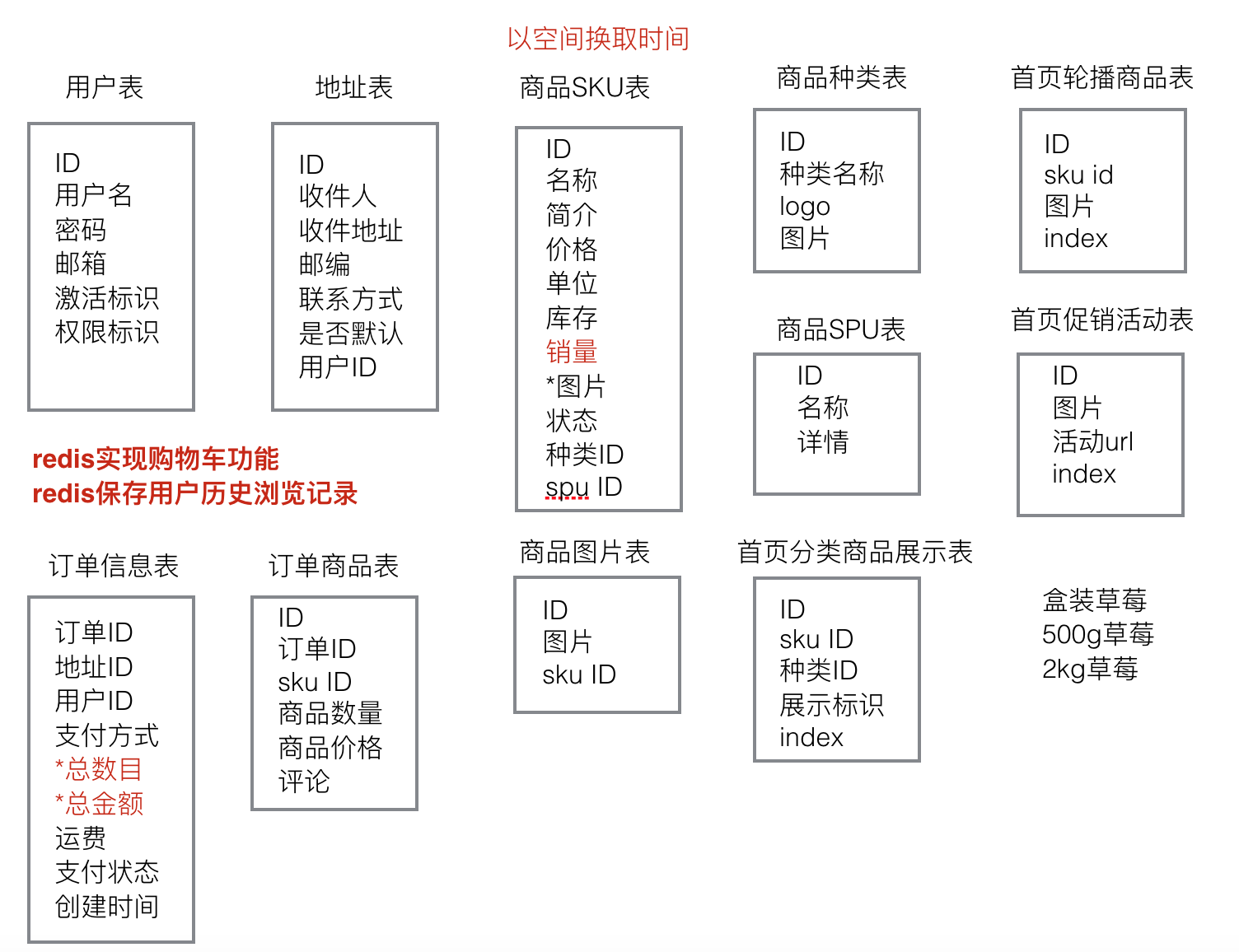
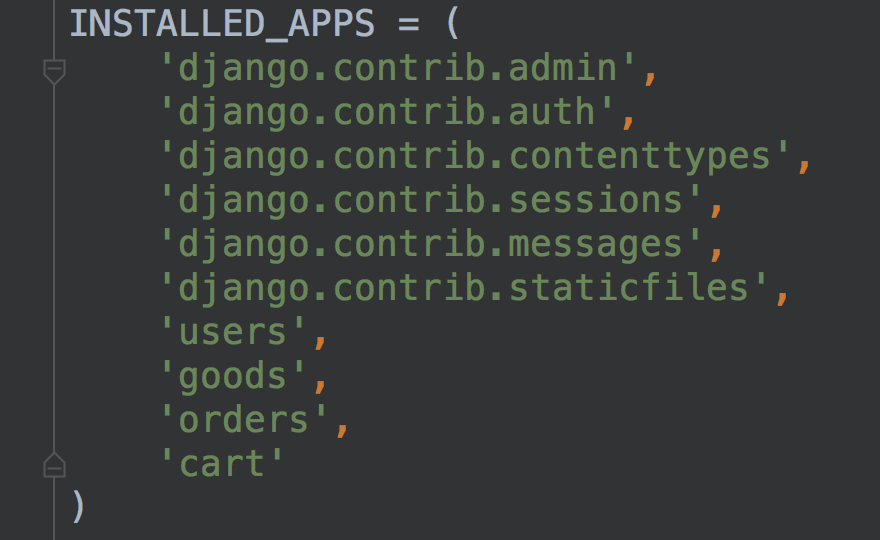
# 项目架构

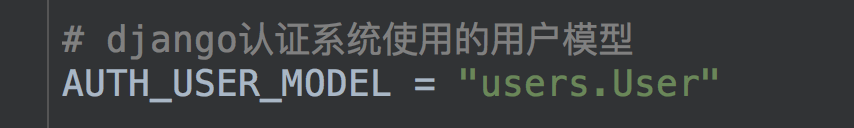


# 数据库表结构



# 用户认证模型





注意： AUTH\_USER\_MODEL配置参数要在第一次迁移数据库之前配置，否则可能django的认证系统工作不正常

# 类视图

将视图view以类的形式定义

通用类视图基类：

django.views.generic.View ( 与django.views.generic.base.View 是同一个)

urls.py中配置路由使用类视图的as\_view()方法

由dispatch()方法具体将请求request分发至对应请求方式的处理方法中（get、post等）

扩展阅读：

<http://python.usyiyi.cn/translate/django_182/topics/class-based-views/intro.html>

**类视图资料：**

http://python.usyiyi.cn/documents/django\_182/ref/class-based-views/base.html#django.views.generic.base.View.as\_view

# 用户模块开发

## Django认证系统文档

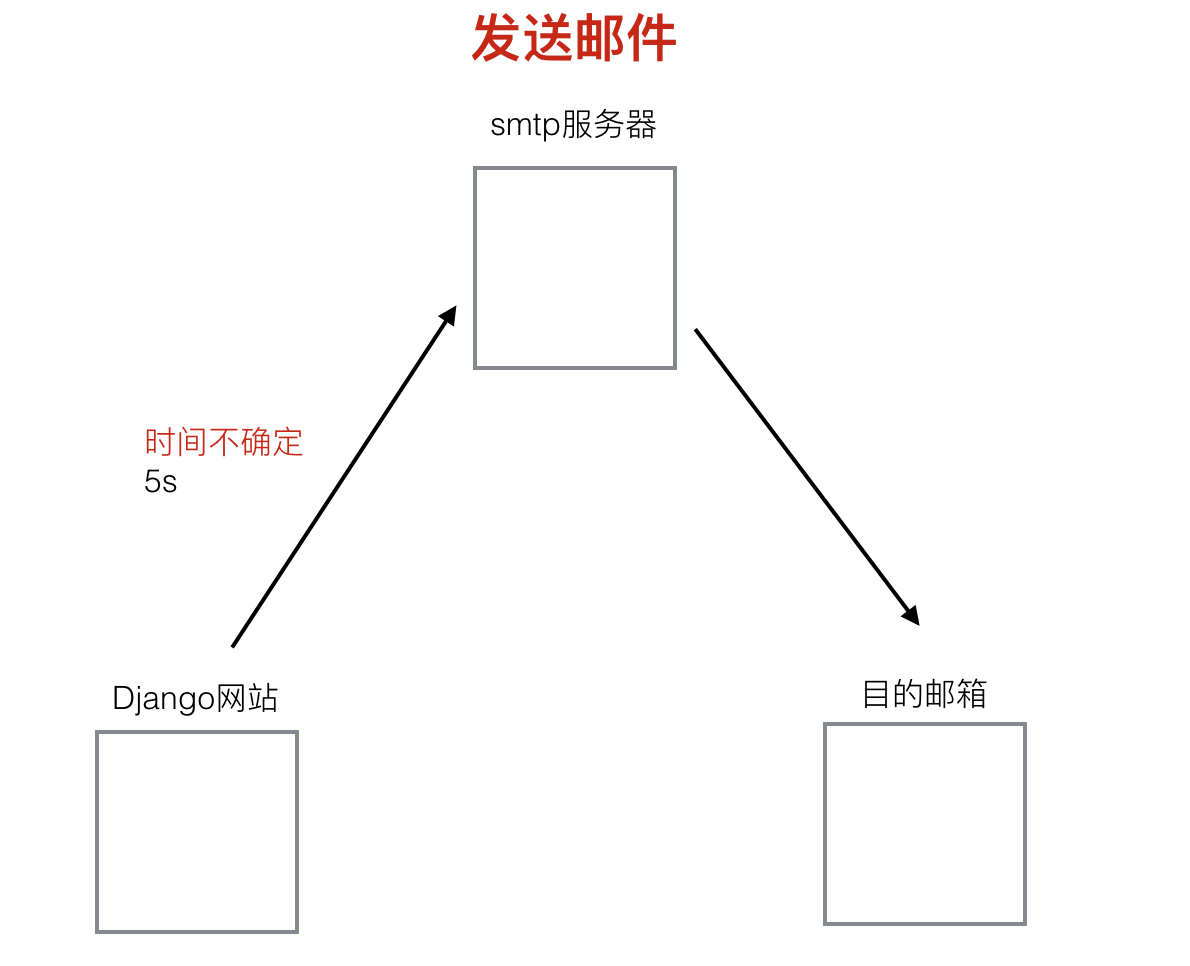
<http://python.usyiyi.cn/documents/django_182/topics/auth/default.html>

**预习方法：**

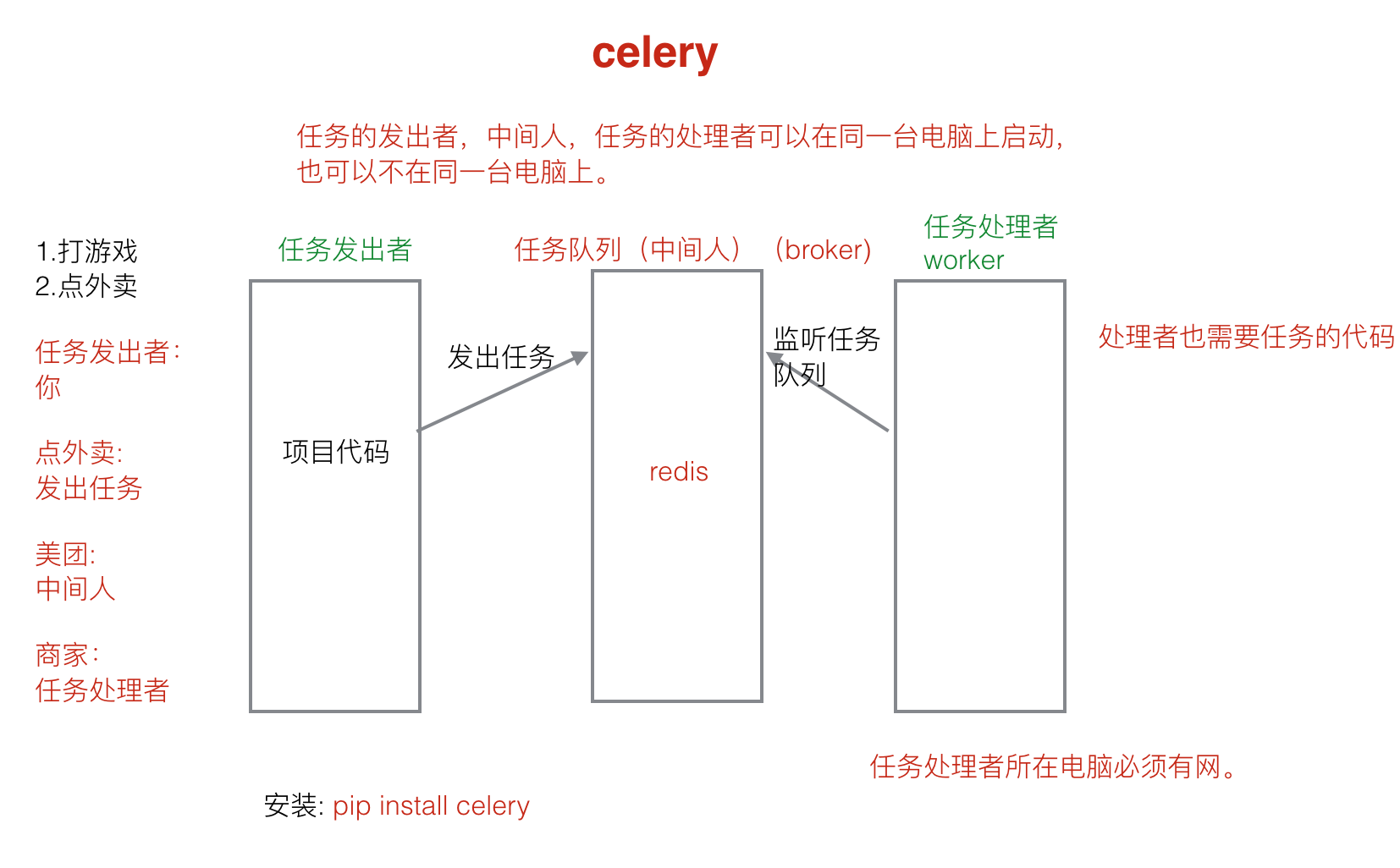
|  |  |
| --- | --- |
| **方法名** | **备注** |
| create\_user | **创建用户** |
| authenticate | **登录验证** |
| login | **记录登录状态** |
| logout | **退出用户登录** |
| is\_authenticated | **判断用户是否登录** |
| login\_required装饰器 | **进行登录判断** |

## 用户注册

### 5.2.1 django发送邮件

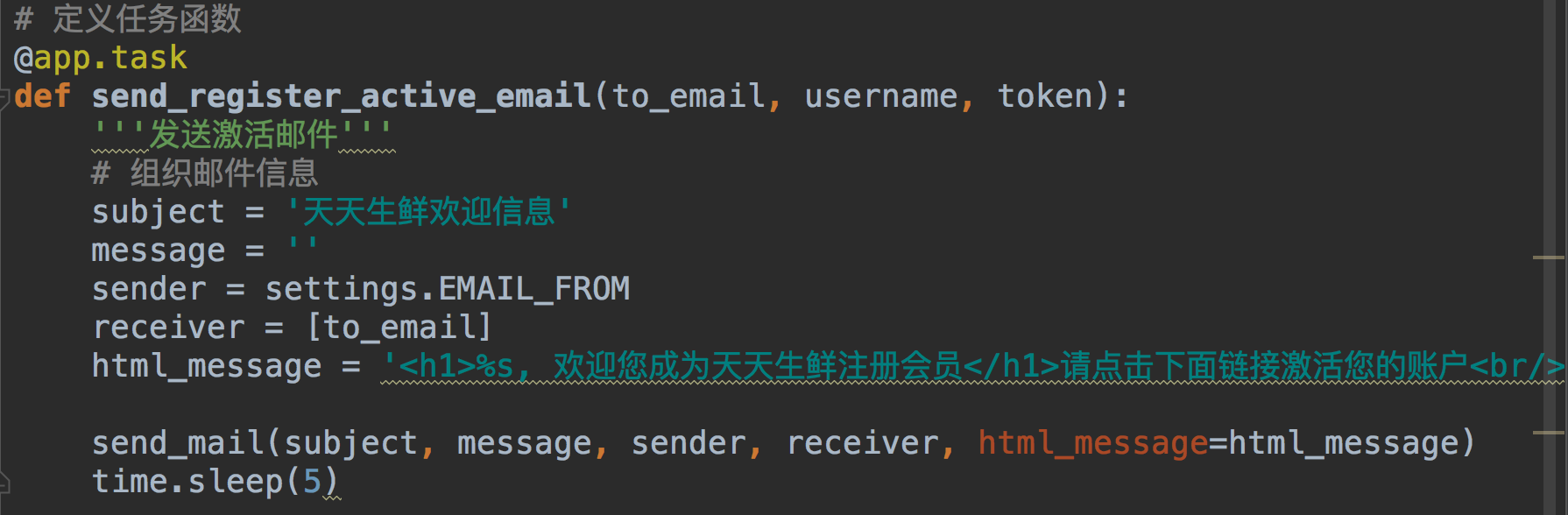


### celery

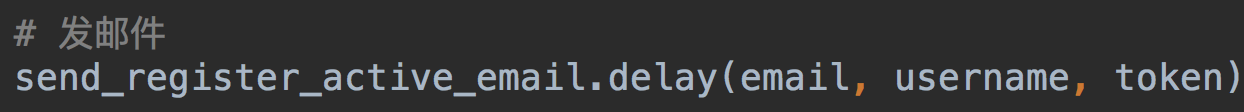


1. 使用

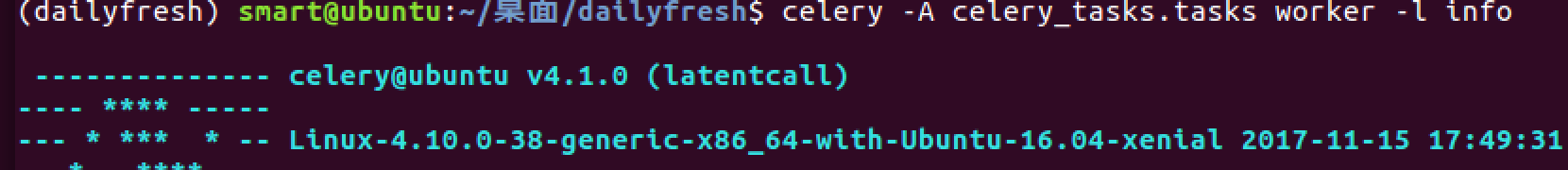




1. 发出任务



1. 启动worker



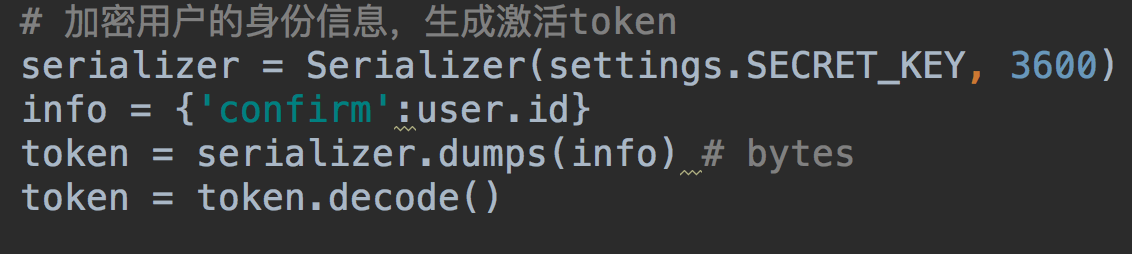
## 用户激活

使用itsdangerous 加密用户的身份信息。

参考资料：

http://itsdangerous.readthedocs.io/en/latest/

1. 加密用户身份信息



1. 解密用户身份信息



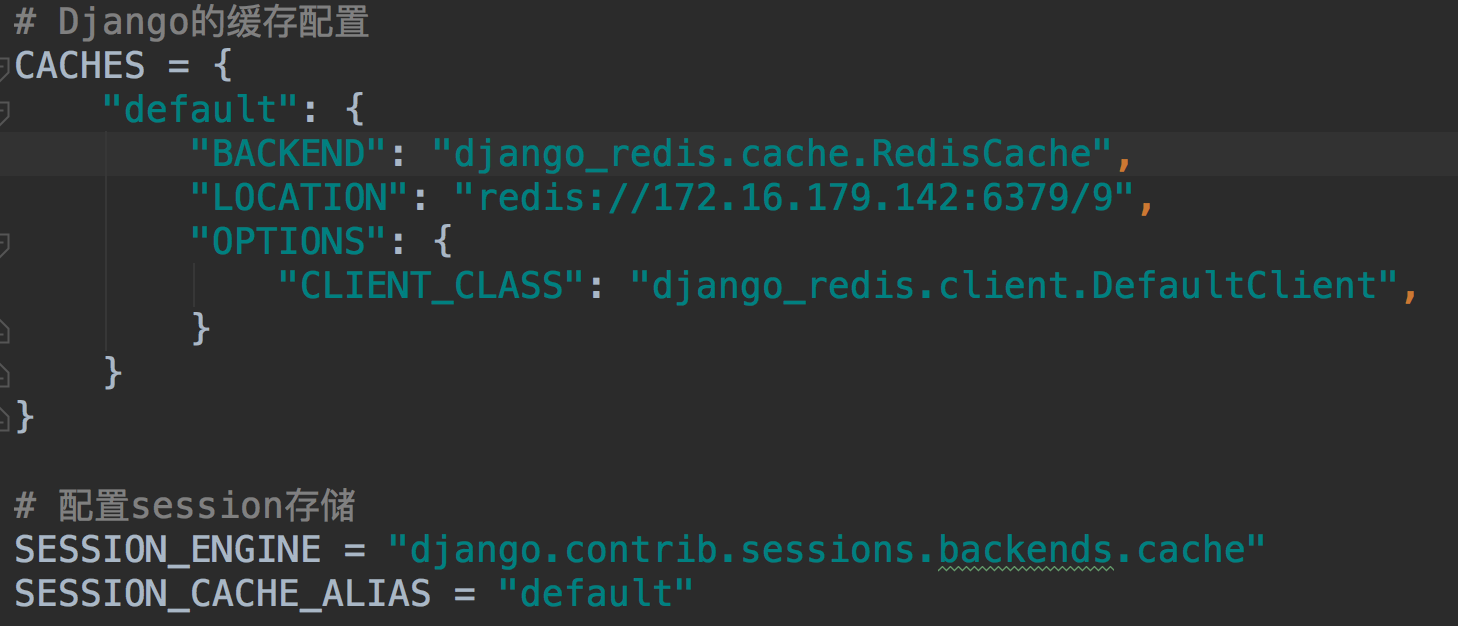
## 用户登录

### 5.4.1 配置redis作为Django缓存和session后端

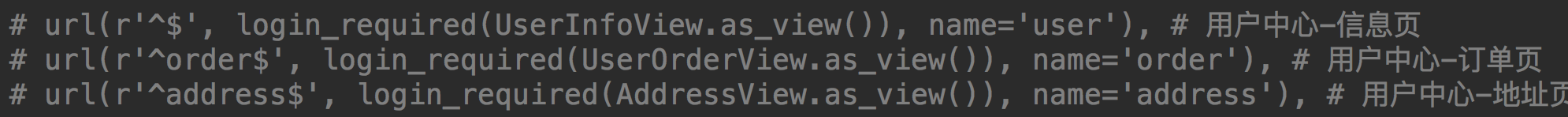
django-redis文档：

[http://django-redis-chs.readthedocs.io/zh\_CN/latest/#](http://django-redis-chs.readthedocs.io/zh_CN/latest/)

配置：



### 5.4.2 登录判断装饰器login\_required

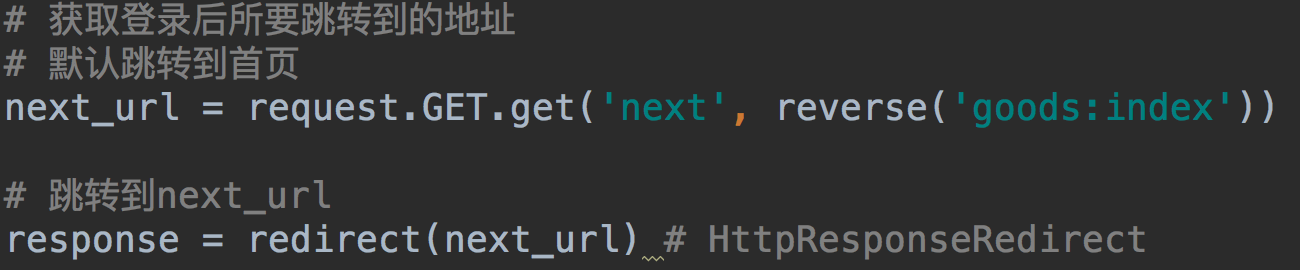




使用LoginRequireMixin:



登录后跳转：

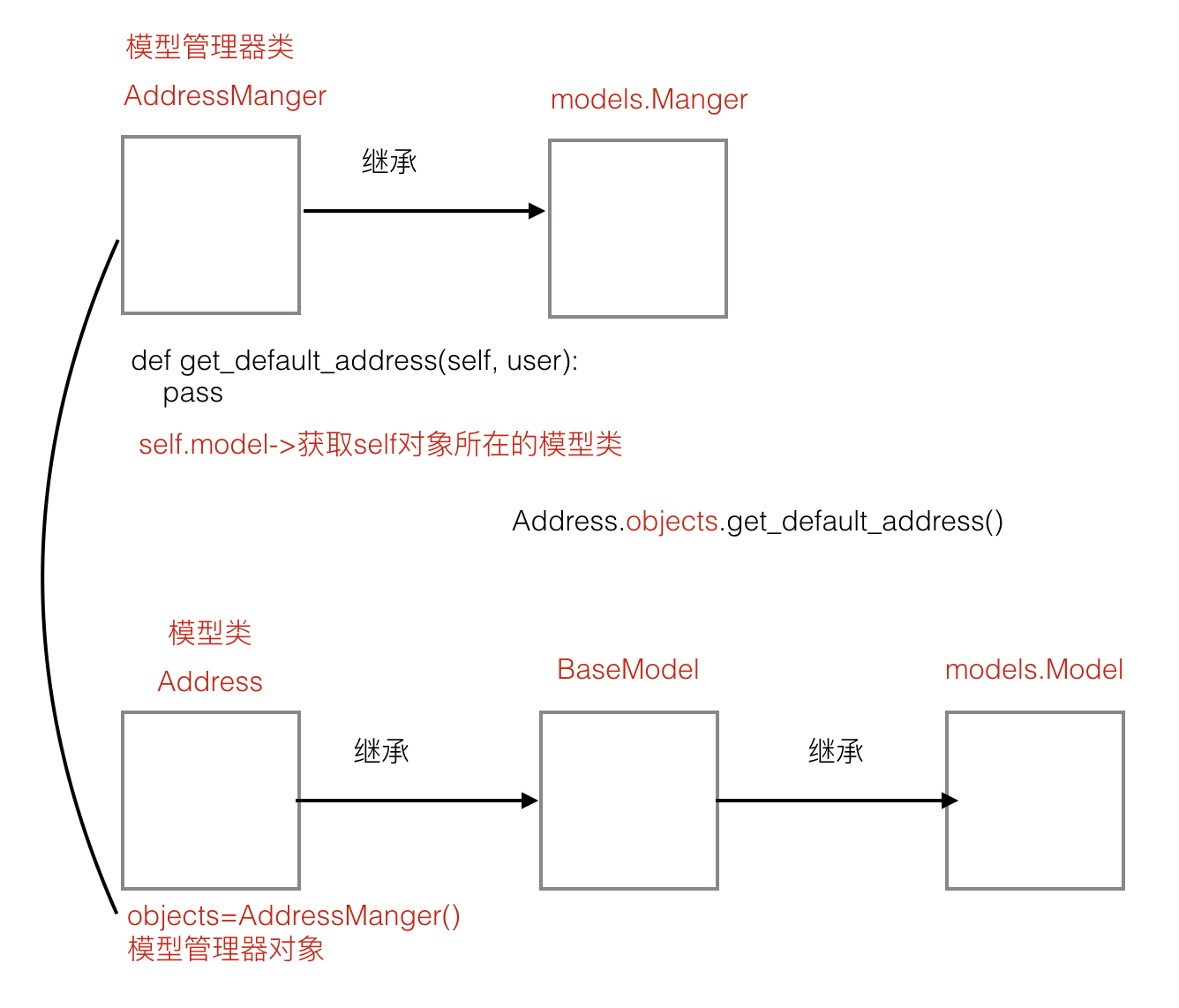


## 用户退出

logout函数清除登录用户的session信息。

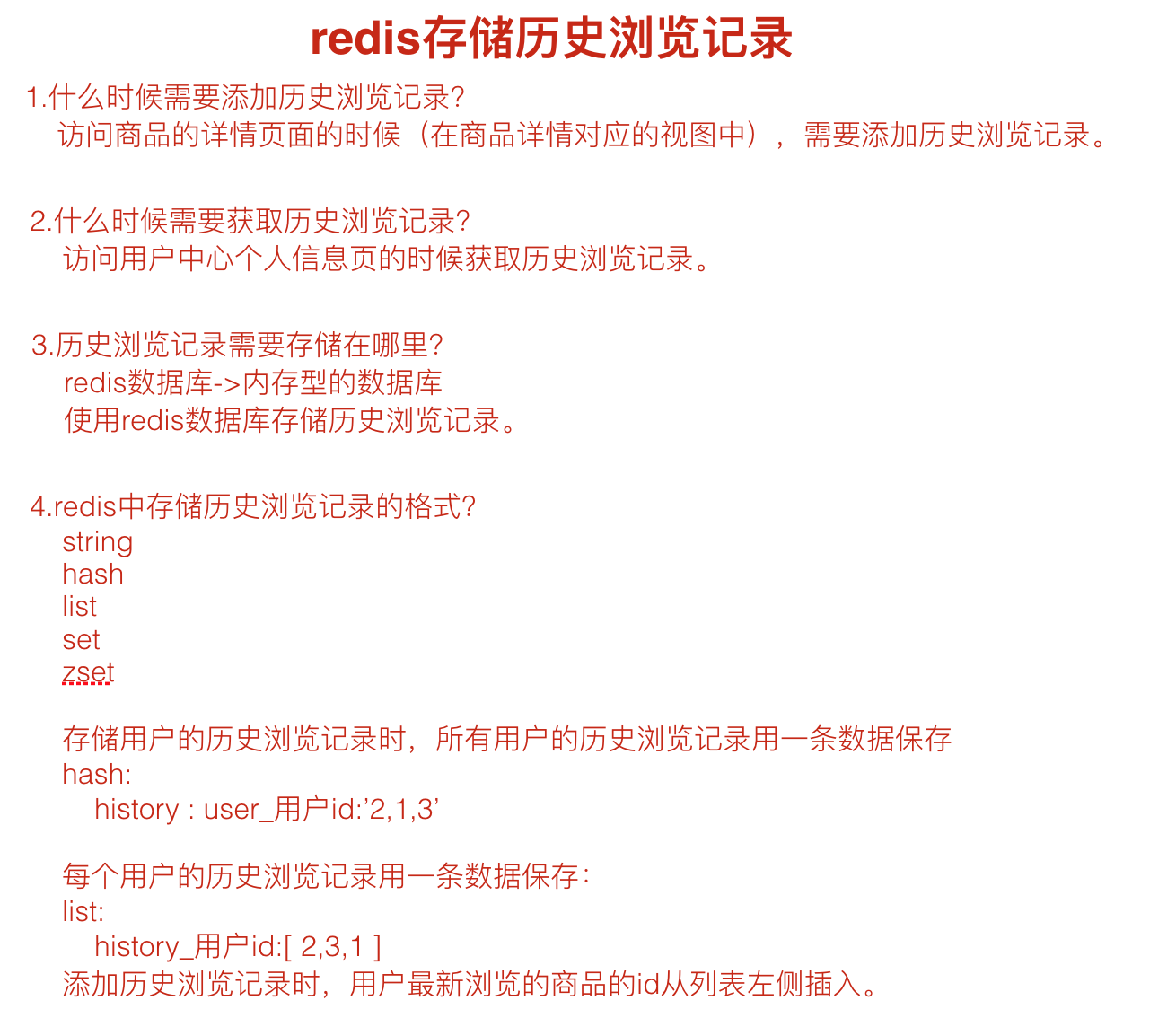
## 用户地址

### 5.6.1 模型类和模型管理器类



## 用户个人信息

### 5.7.1 redis存储历史浏览记录分析

****

**参考资料：**

<http://redisdoc.com/index.html>

<http://redis-py.readthedocs.io/en/latest/#indices-and-tables>

* + 1. **django-redis获取redis链接**

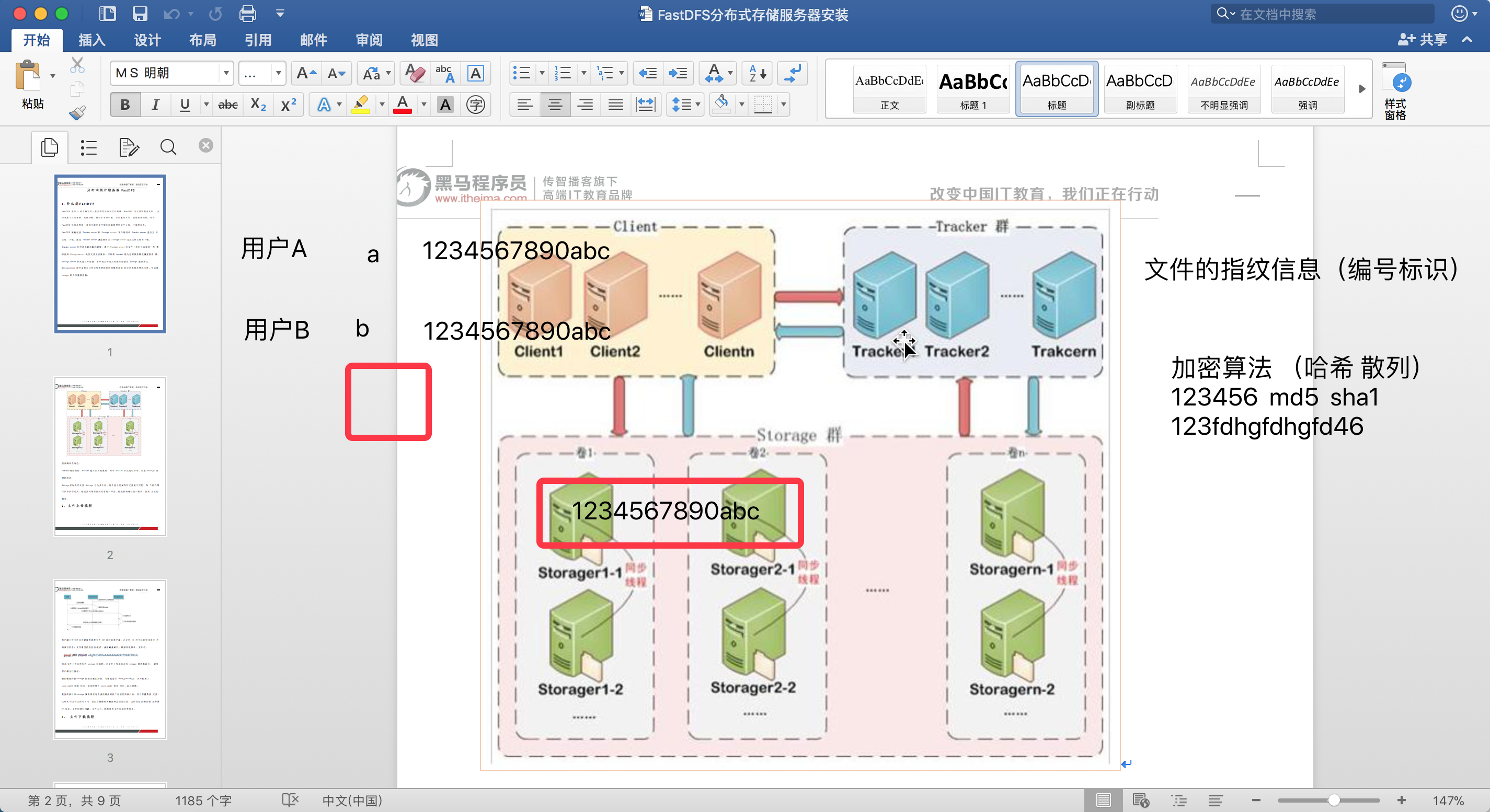


****

# 商品模块开发

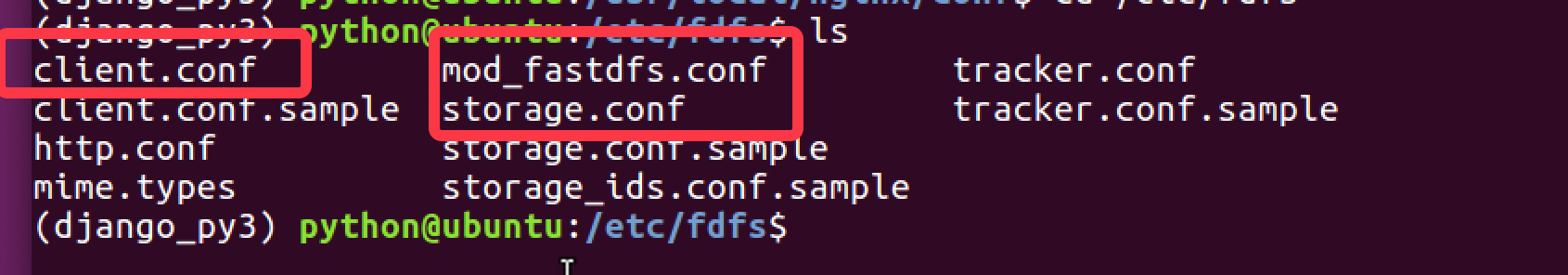
## 6.1了解FastDFS分布式文件系统

集群



启动FastDFS的方法，需要的操作：

1. 修改如下的配置文件 （在/etc/fdfs目录中）



tracker\_server=自己的ip地址:22122

1. 启动tracker、storage、nginx服务：

sudo service fdfs\_trackerd start

sudo service fdfs\_storaged start

sudo /usr/local/nginx/sbin/nginx

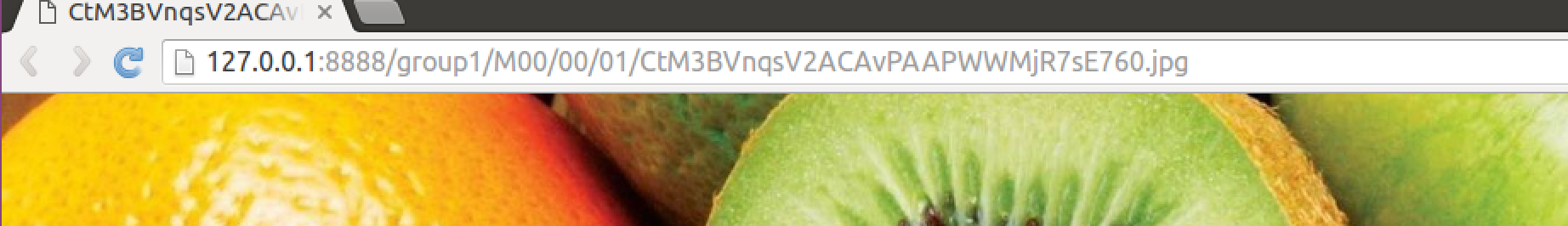
1. 执行如下命令测试是否成功

**fdfs\_upload\_file /etc/fdfs/client.conf 要上传的图片文件**

如果返回类似group1/M00/00/00/rBIK6VcaP0aARXXvAAHrUgHEviQ394.jpg的文件id则说明文件上传成功

在浏览器中可以用 127.0.0.1:8888/返回的文件id

访问图片



## 6.2 python对接fastdfs

1. workon django\_py3

2. 进入fdfs\_client-py-master.zip所在目录

3. pip install fdfs\_client-py-master.zip

>>> from fdfs\_client.client import Fdfs\_client

>>> client = Fdfs\_client('/etc/fdfs/client.conf')

>>> ret = client.upload\_by\_filename('test')

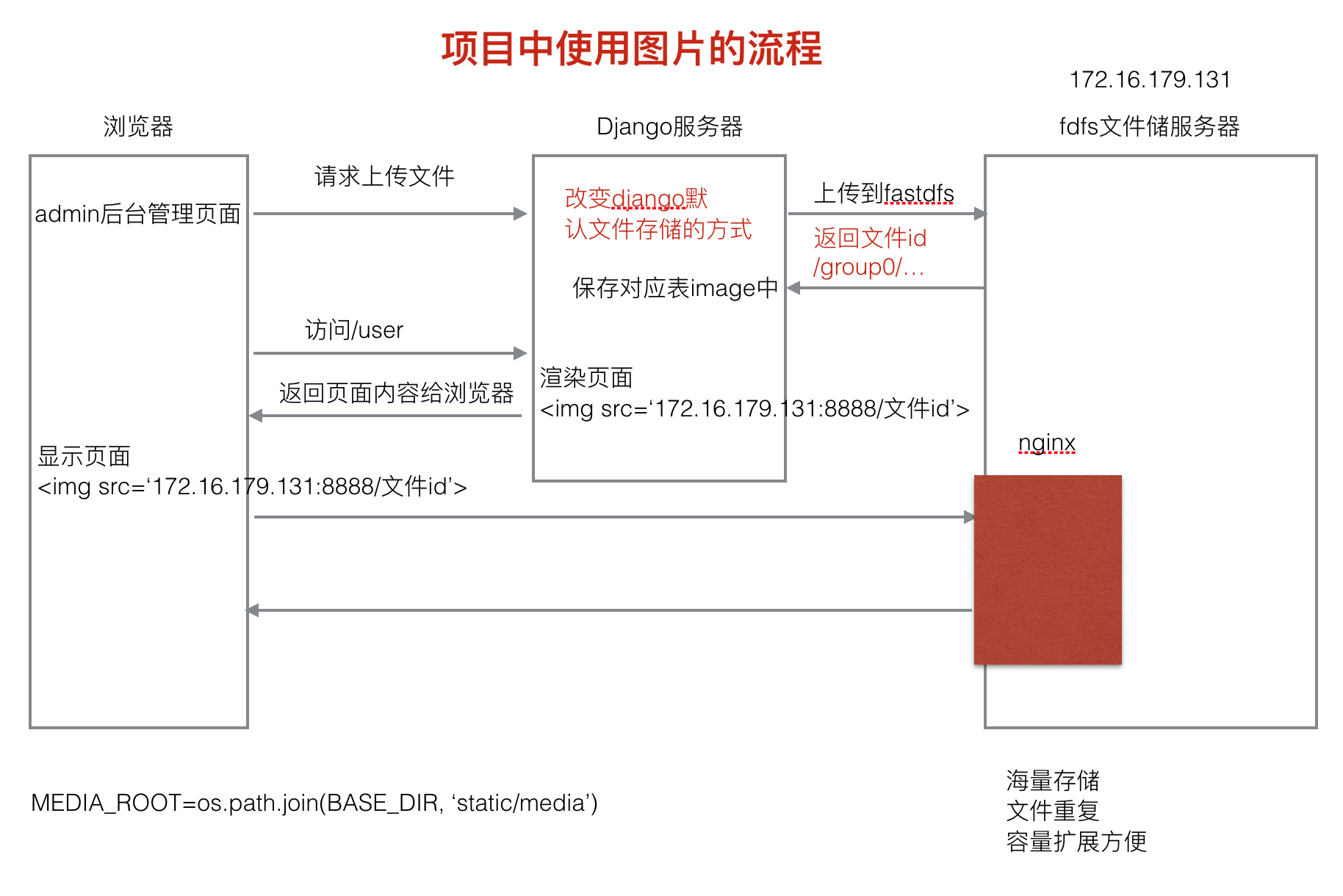
>>> ret

{'Group name':'group1','Status':'Upload successed.', 'Remote file\_id':'group1/M00/00/00/

wKjzh0\_xaR63RExnAAAaDqbNk5E1398.py','Uploaded size':'6.0KB','Local file name':'test'

, 'Storage IP':'192.168.243.133'}

## 项目上传图片和使用图片流程



## Django二次开发对接FastDFS

Django文档：

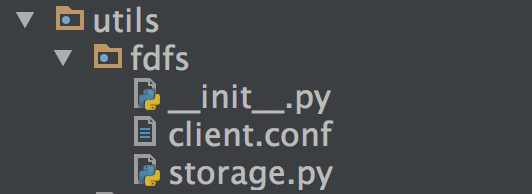
<http://python.usyiyi.cn/translate/django_182/ref/files/storage.html>

<http://python.usyiyi.cn/translate/django_182/howto/custom-file-storage.html>

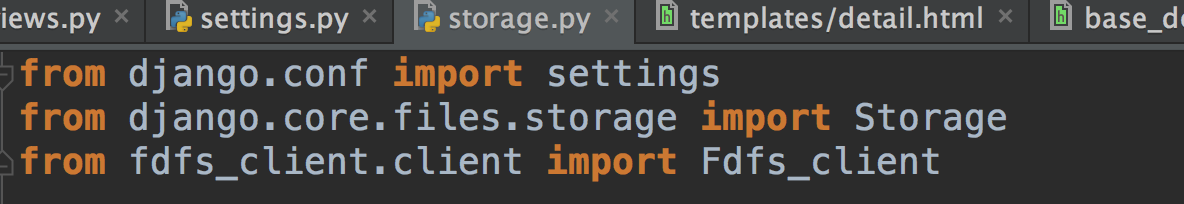
1. 配置文件settings中加入如下配置

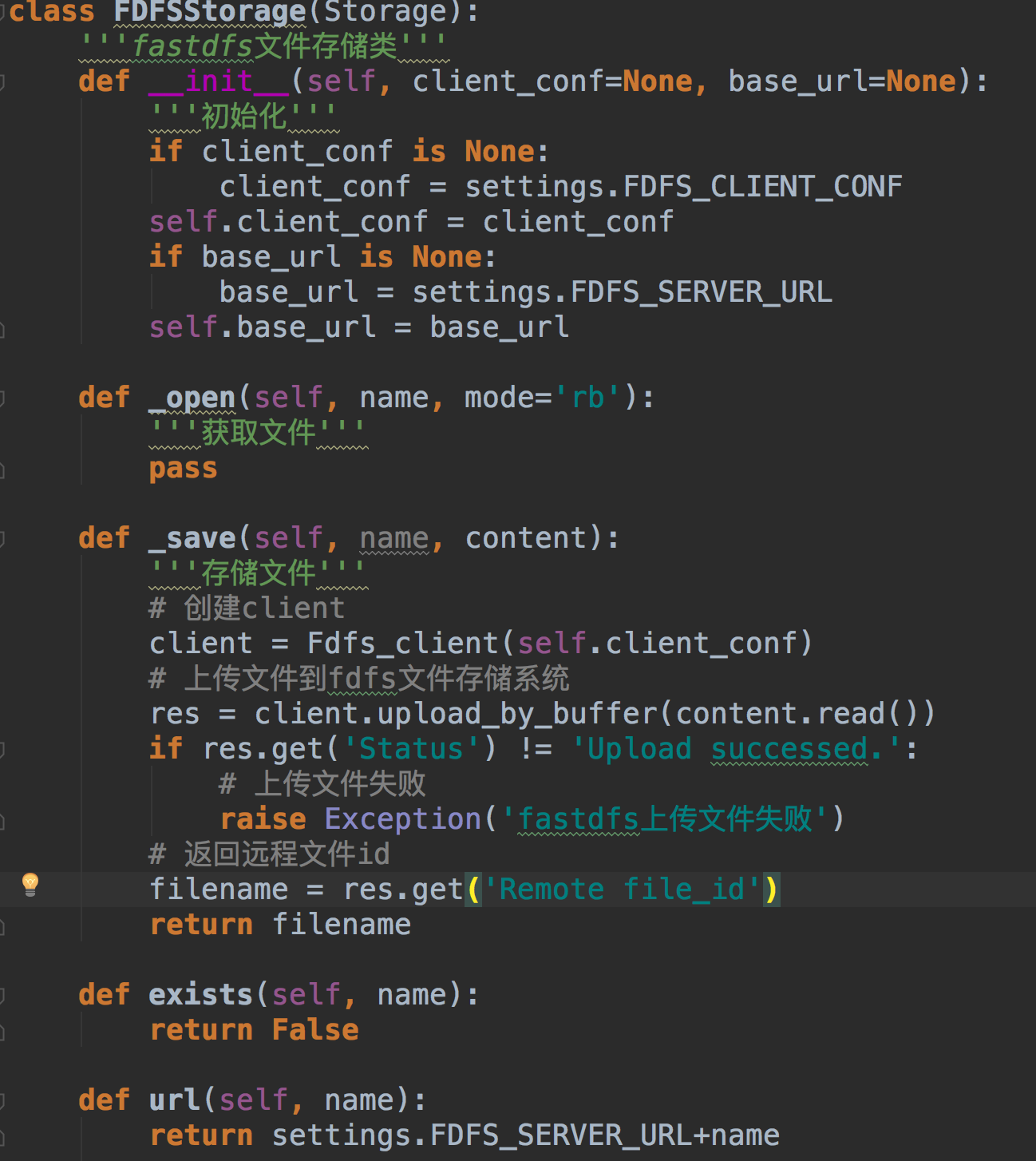


1. 创建utils/fdfs 目录



1. storage.py文件中自定义文件存储类

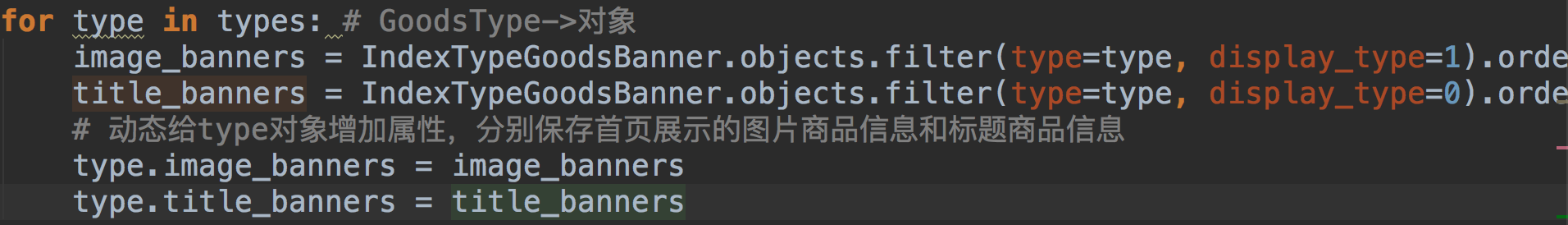




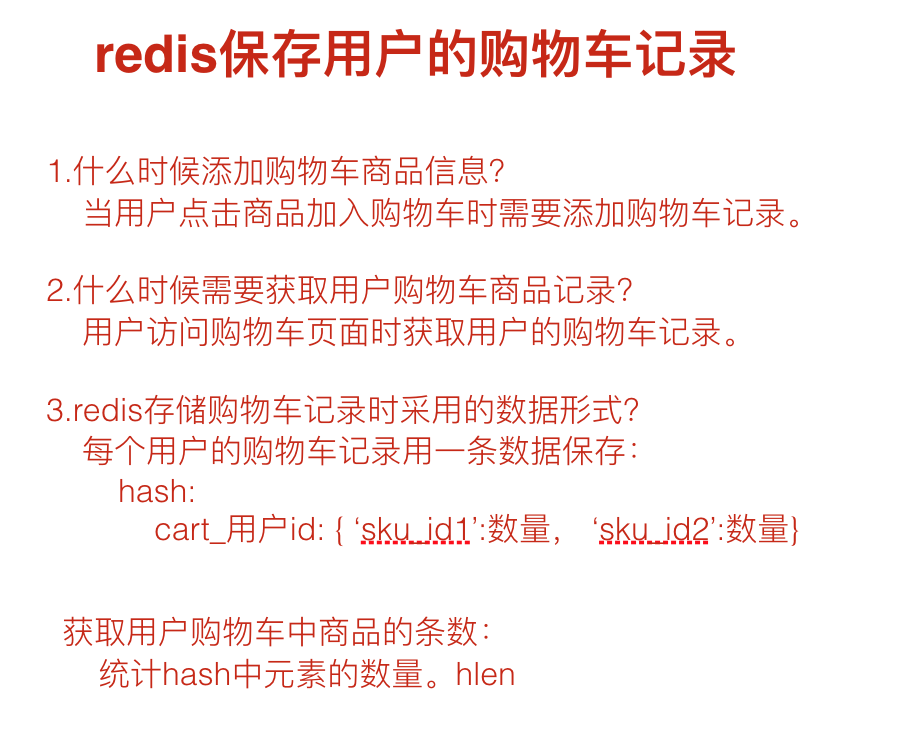
## 7.1商品首页

### 7.4.1 基本页面处理

动态给对象增加属性

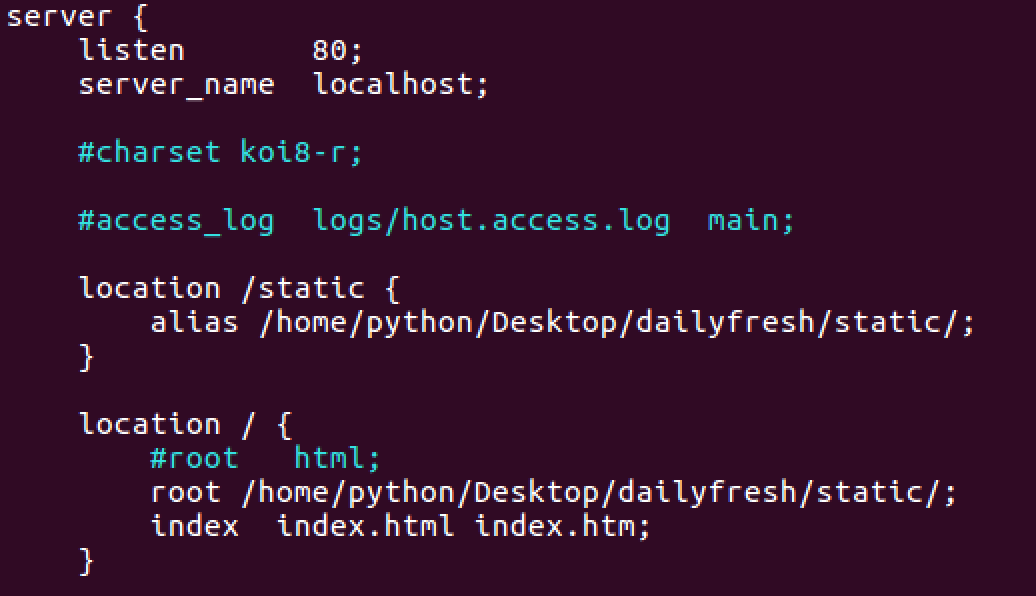


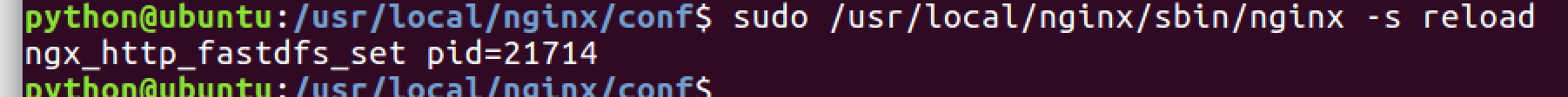
### 7.4.2 保存购物车信息的数据设计

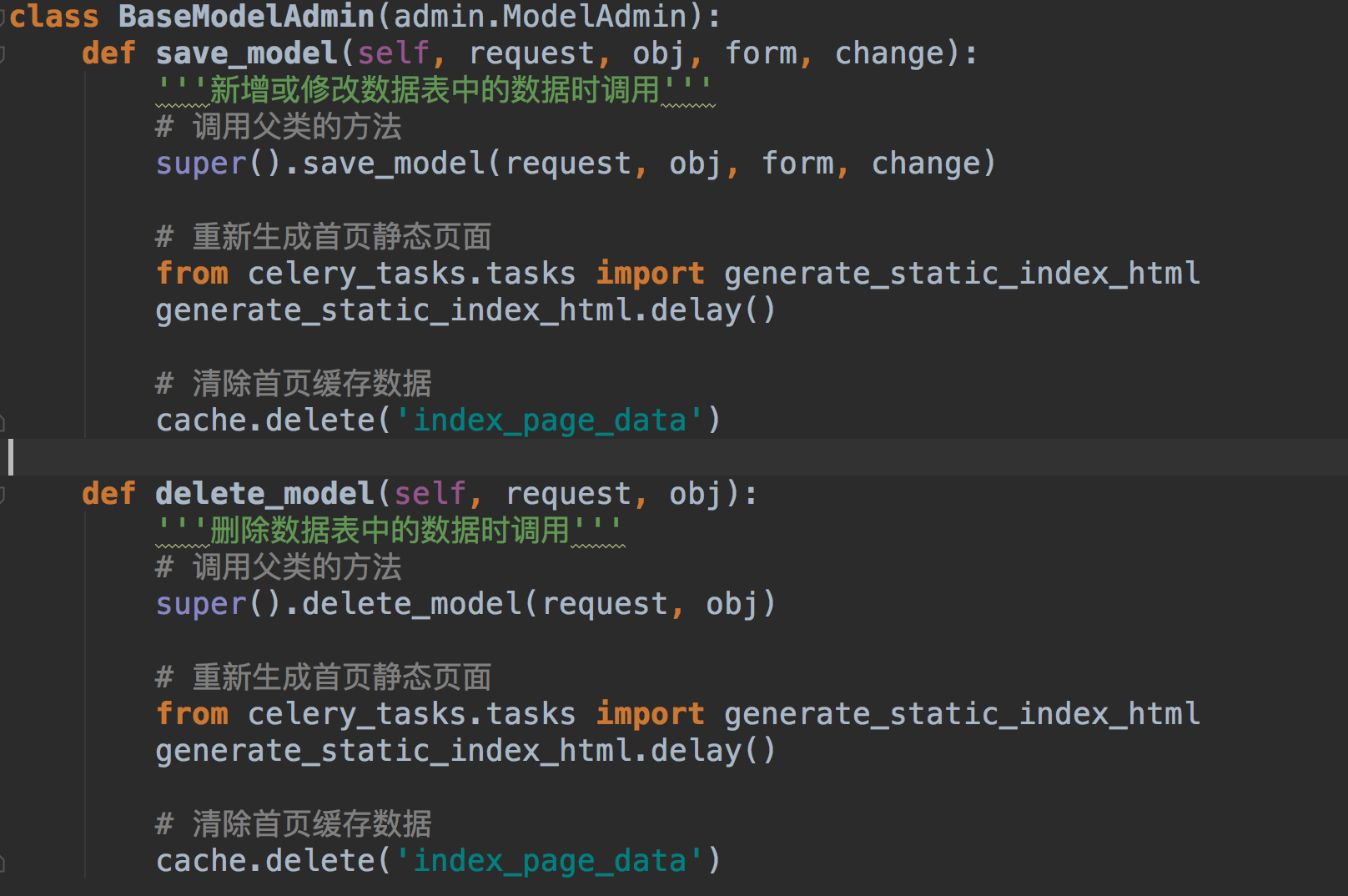


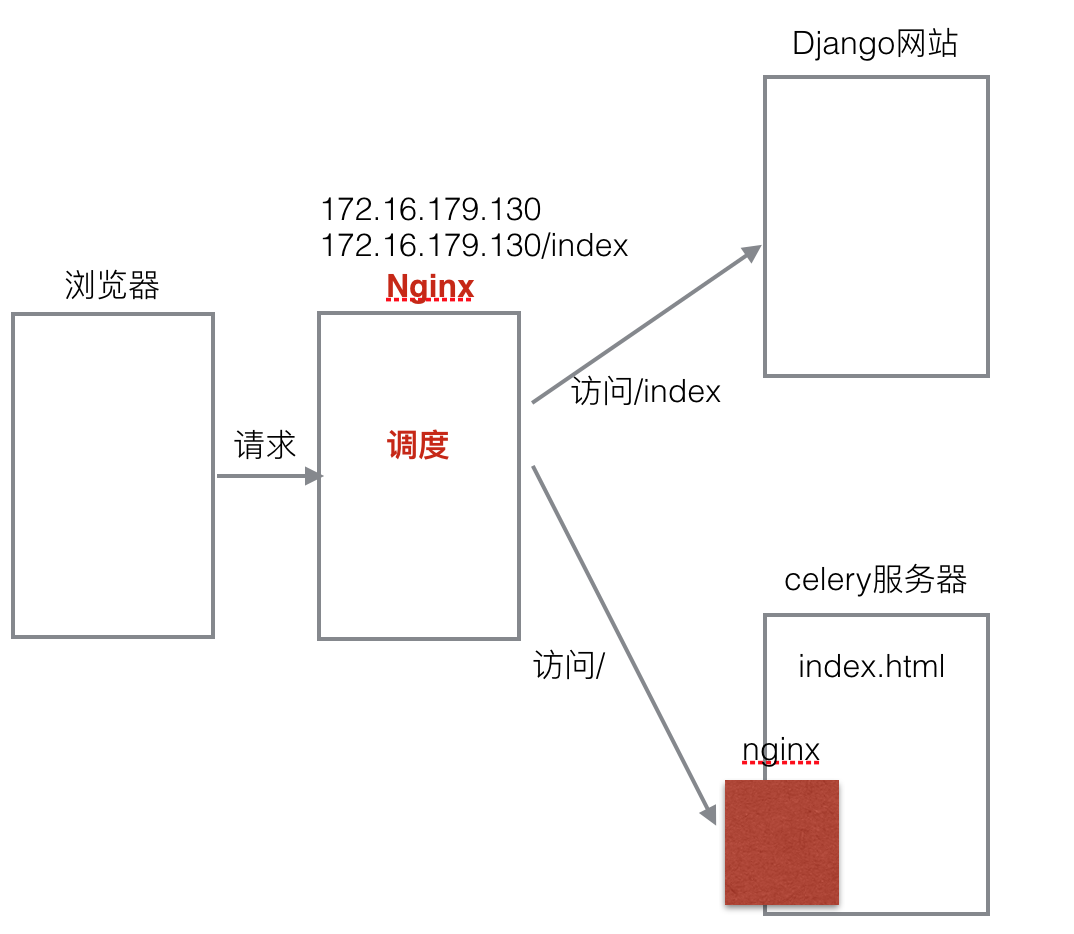
### 7.4.3 页面静态化

把原本动态的页面处理结果保存成html文件，让用户直接访问这个生成出来的静态的html页面









### 8.4.4 使用缓存

将处理计算的结果先临时保存起来，下次使用的时候可以先直接使用，如果没有这个备份的数据，重新进行计算处理

将缓存数据保存在内存中 （本项目中保存在redis中）

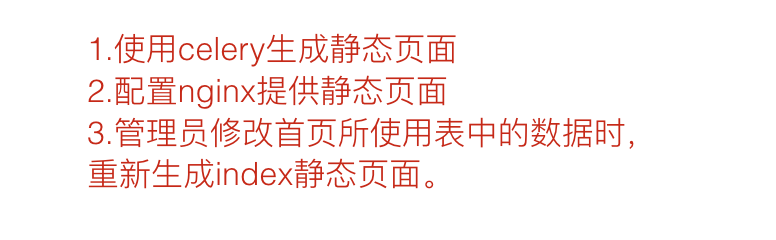
cache

修改了数据库的数据，直接删除缓存

缓存要设置有效期

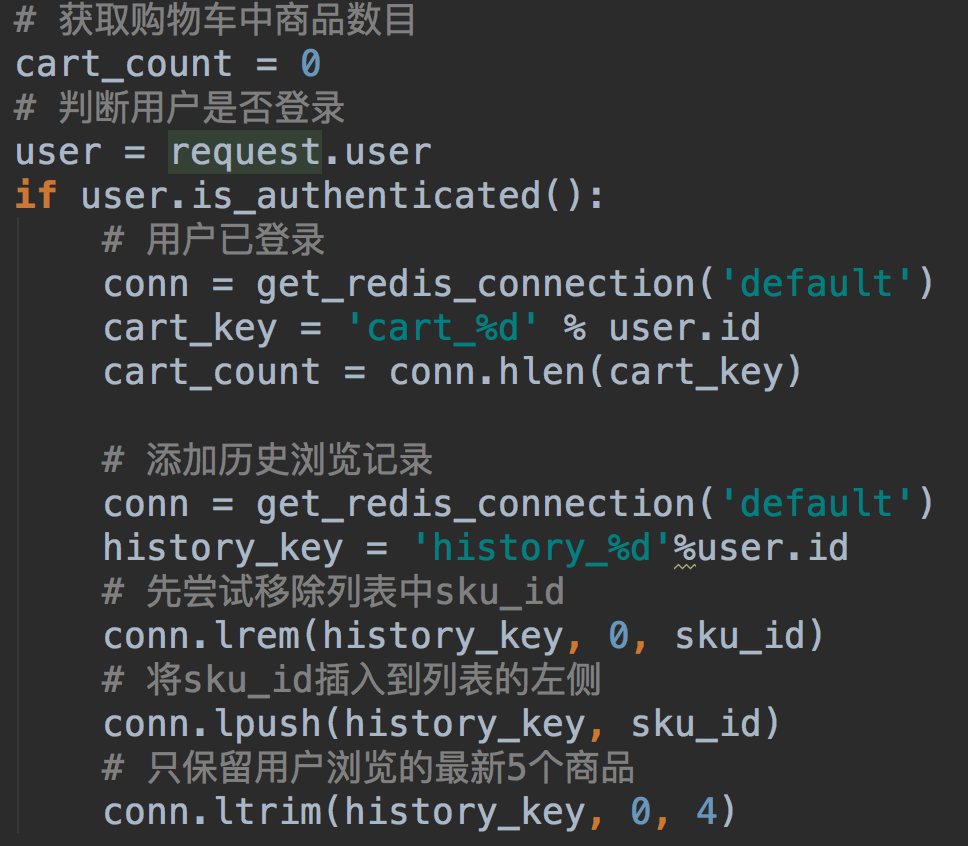
django文档：

<http://python.usyiyi.cn/translate/django_182/topics/cache.html>



## 商品详情页

添加历史浏览记录：



## 商品列表页