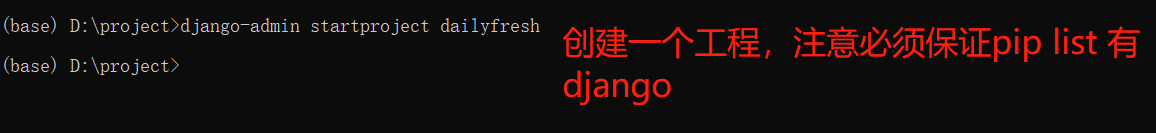
天天生鲜教案

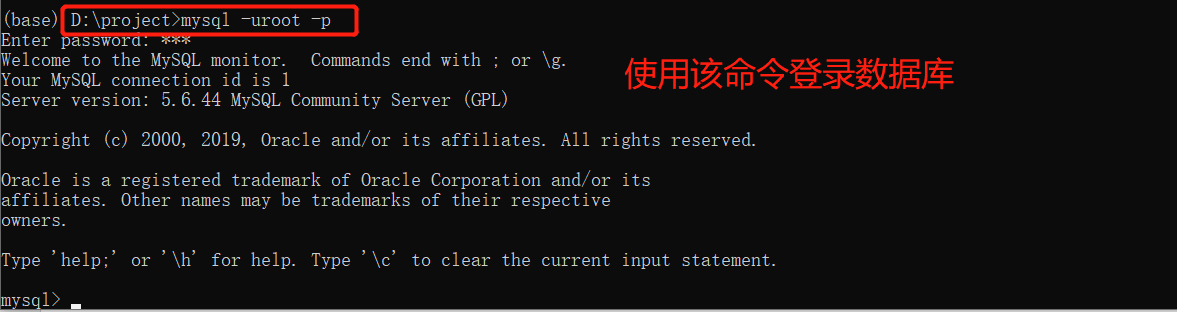
# 创建一个工程



# 使用pycharm打开这个工程

# 使用django连接MySQL数据库

1. 登录MySQL控制台并使用SQL语句创建一个数据库名字叫dailyfresh

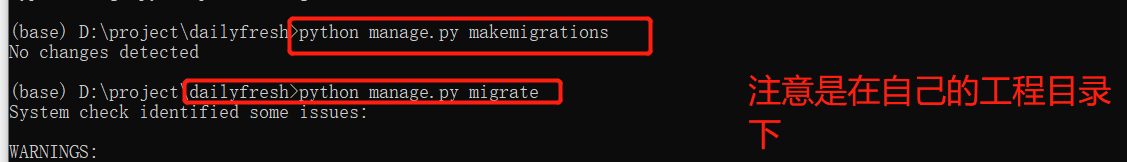




1. 在django的settings.py中做配置连接数据库

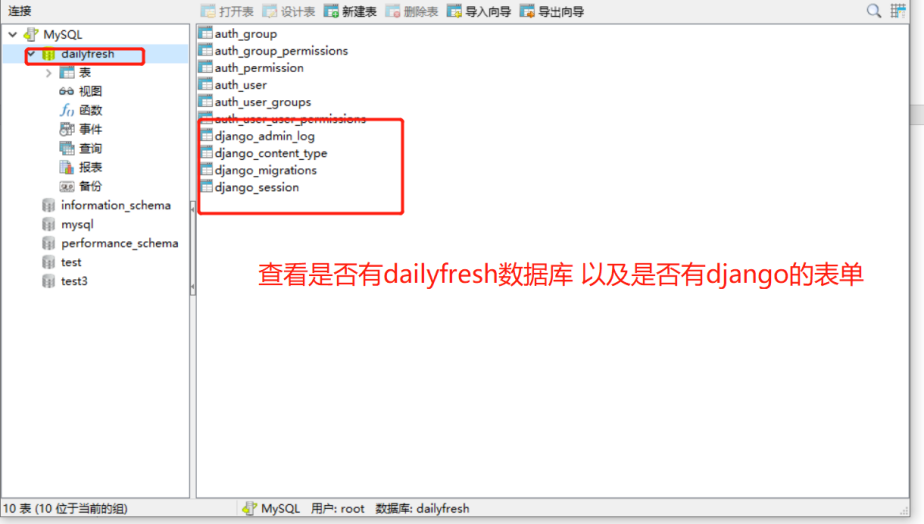


1. 数据库迁移



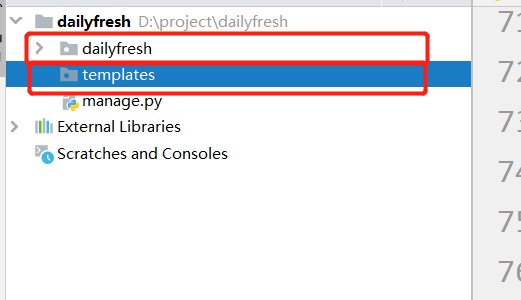
1. 检查

打开Navicat检查

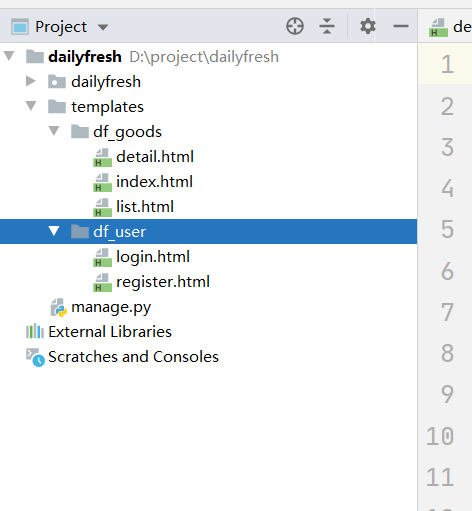


# 将界面添加到工程文件中

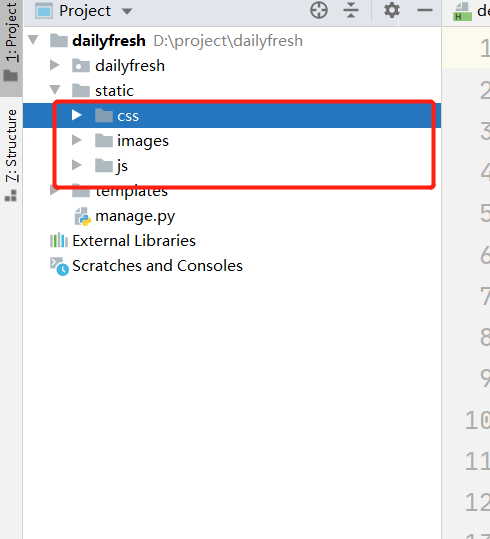
1. 在工程文件目录下创建一个templates文件，这个文件夹主要用于存放html界面。创建后的结果如下图。注意目录之间的层级关系。



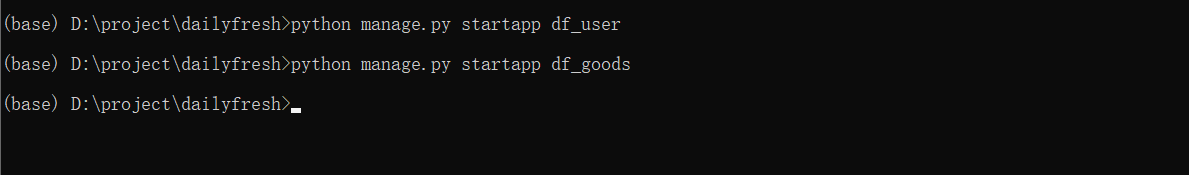
1. 在templates里面创建两个目录，分别是df\_user和df\_goods。因为本次的天天生鲜要实现登录、注册以及首页的功能，所以登录和注册会创建一个app而首页会创建一个app。创建后的结果如下图。并且将index.html,list.html,detail.html复制到df\_goods目录下，将login.html、register.html复制到df\_user目录下。

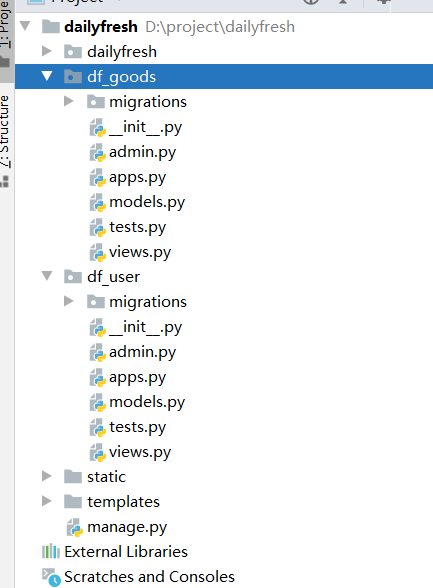


1. 在工程文件目录下创建一个static目录，和templates平级。并将css image js这三个文件夹复制到static目录下。



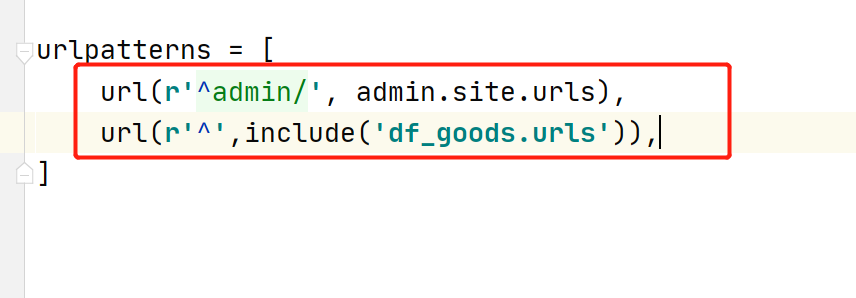
1. 使用命令为django工程创建两个app分别是df\_user和df\_goods分别代表了登录注册功能模块和网站首界面。并且在settings.py中加入两个app，效果图如下：



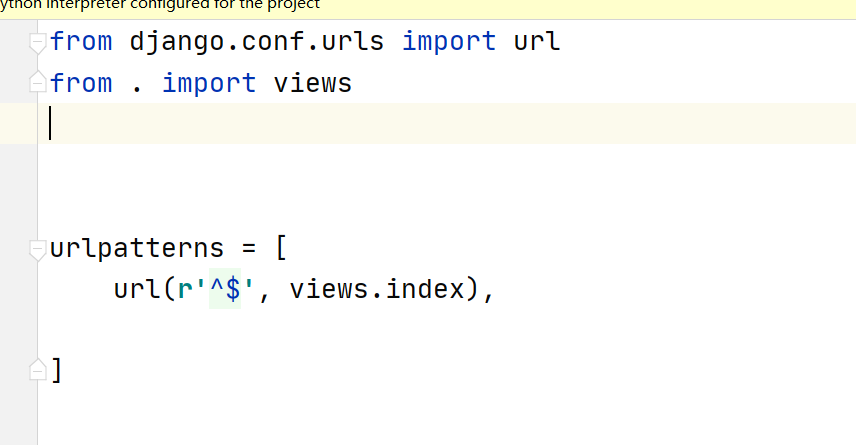




1. 设置Url（路由）使我们的工程在输入ip加Port时能够访问到主界面。
2. 在dailyfresh的url中添加如下一句话。



1. 在df\_goods中创建urls.py文件，并添加如下内容。



1. 在df\_goods目录下的views中添加如下代码。



1. 设置templates目录



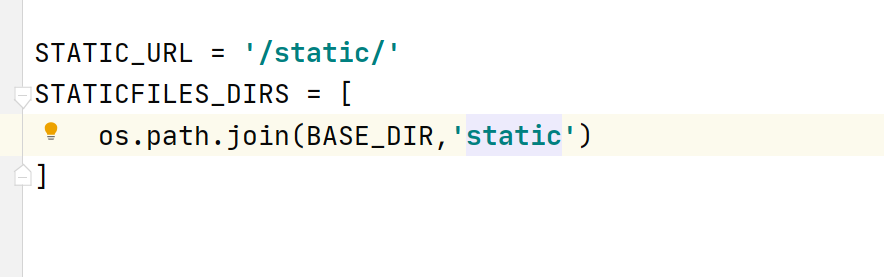
1. 启动工程查看结果



这个时候你会发现你的网站没有排版 只有界面为什么呢？

1. 添加静态文件

在settings.py里面添加两行配置，如图



然后需要在模板文件里做更改 ，未更改的index.html如下图

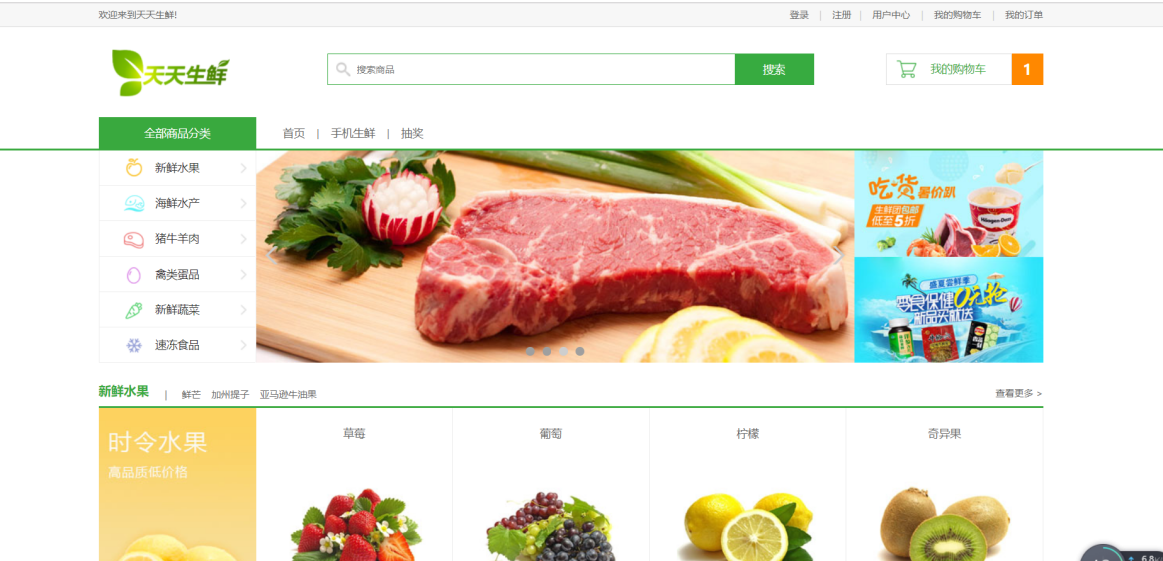


更改后的图，注意每个图片也要做更改





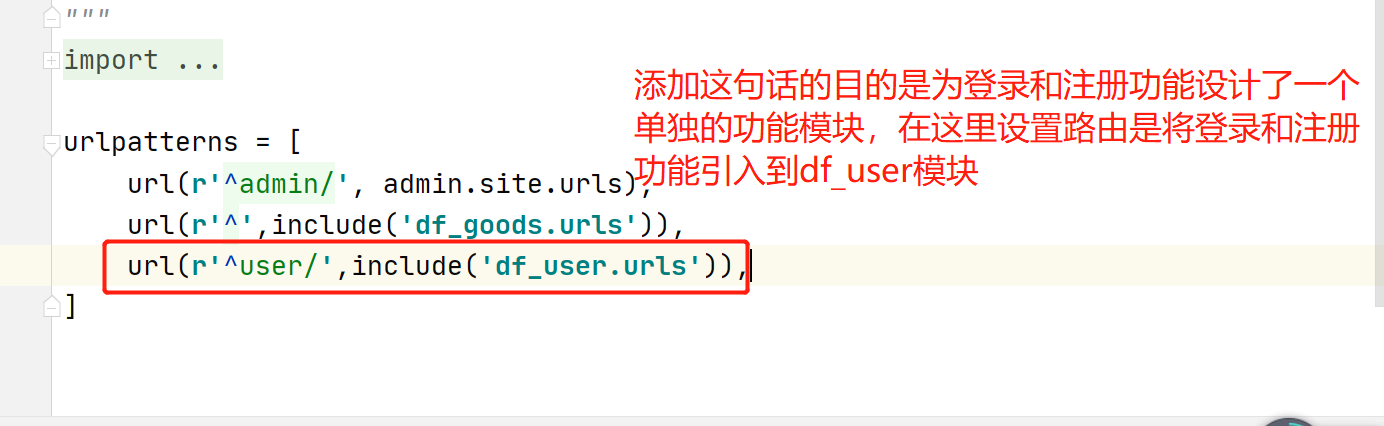
然后刷新



# 实现注册和登录功能

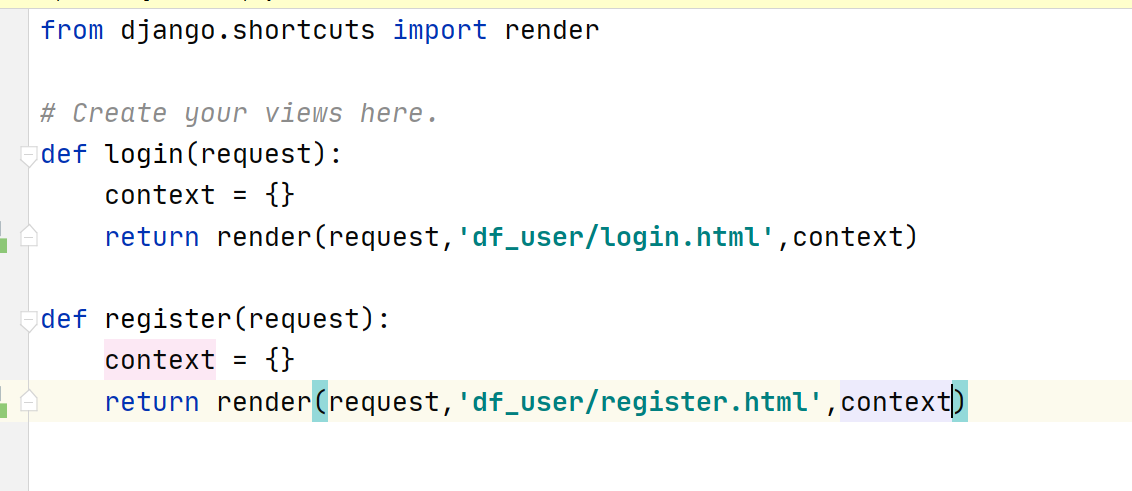
## 5.1 为注册和登录设计路由

在dailyfresh文件夹下的urls.py添加如下代码。同时在df\_user目录下创建urls.py文件，并添加下图代码。





在df\_user目录下的views.py中添加如下代码



在login.html和 register.html中为静态文件以及图像标签加上/static/





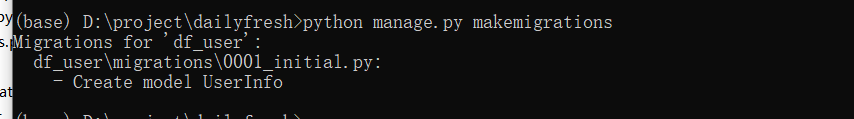
运行django系统测试一下

## 5.2 为用户设计一个模型类

实现一个注册的功能要首先为每个用户创建一个模型。



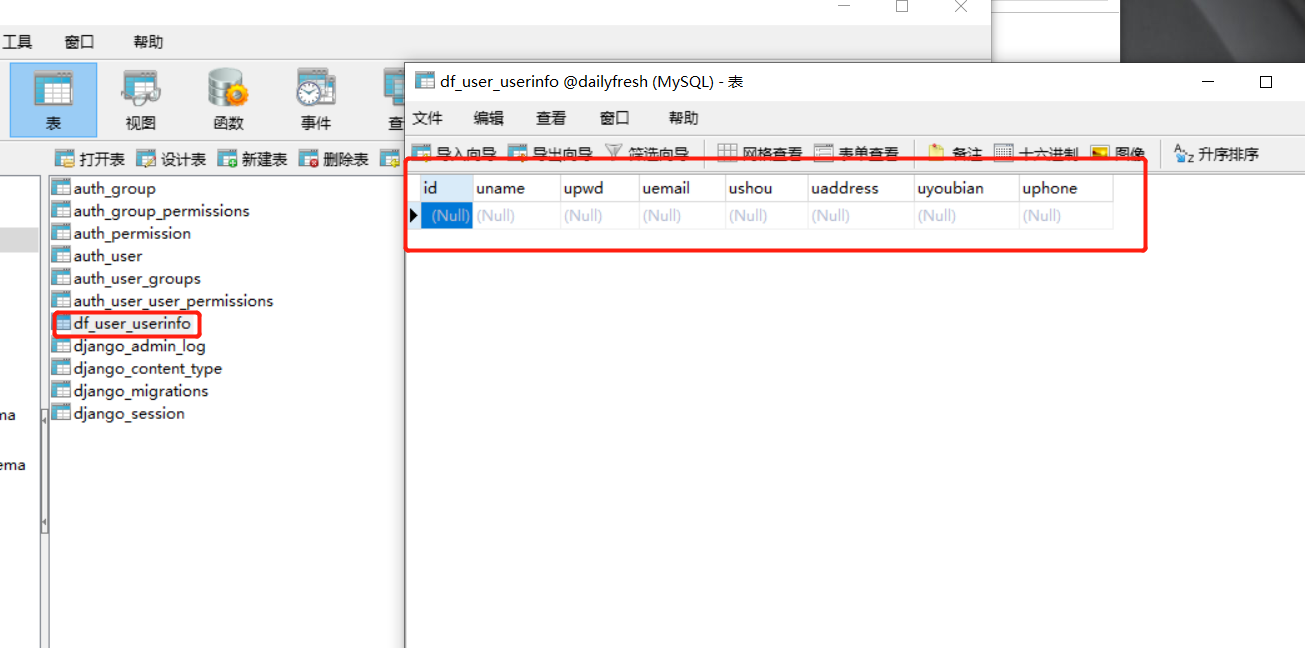
并且生称SQL语句：



迁移到数据库中：



可以在Navicat检查



## 5.3 实现注册功能



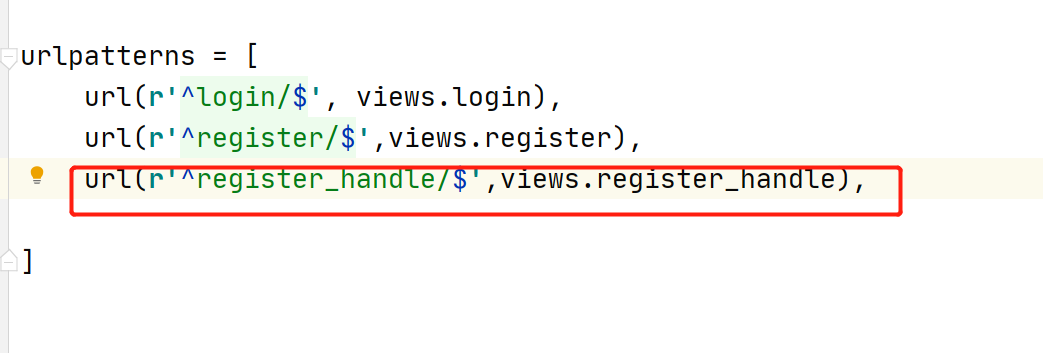
1. 在界面register.html界面中做更改。Form标签表示一个模块，本文的注册表单多个输入框，以及注册按钮 都在form标签内。



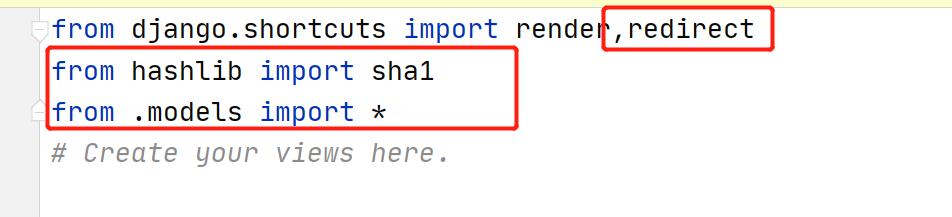
更改后如下



1. 为处理路由设置相应的处理方法。在df\_user目录下的 urls.py 设置如下



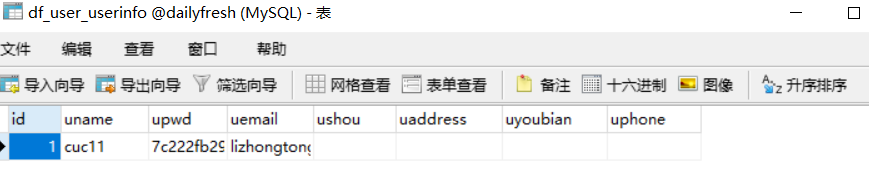
在views.py中设置如下：





1. 输入自己的数据后点注册，查看自己的Navicat





注册功能已经实现。

# 5.4 实现登录功能



1 更改login.html代码。未更改如图



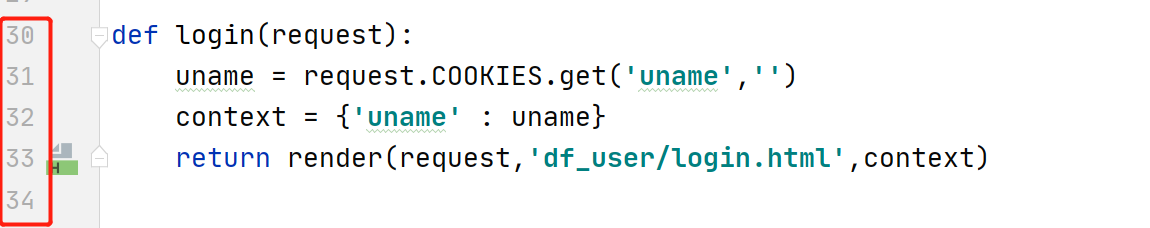
更改后如图：



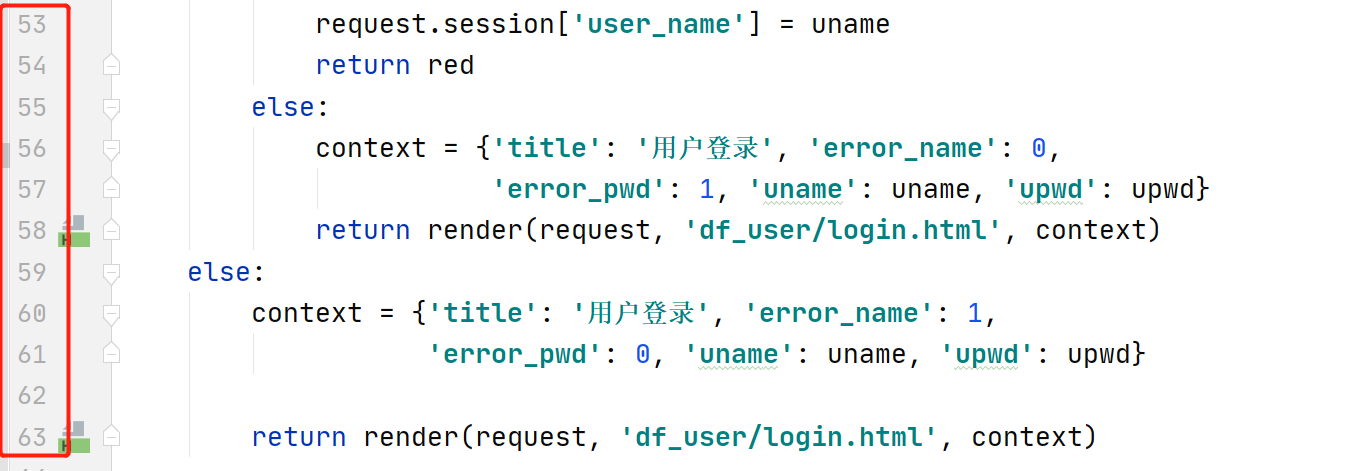
2 设置路由并设置相应的处理函数。在df\_user目录下的urls.py中添加以下代码。



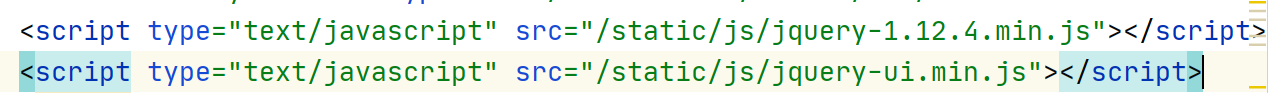
在views.py中添加如下代码：

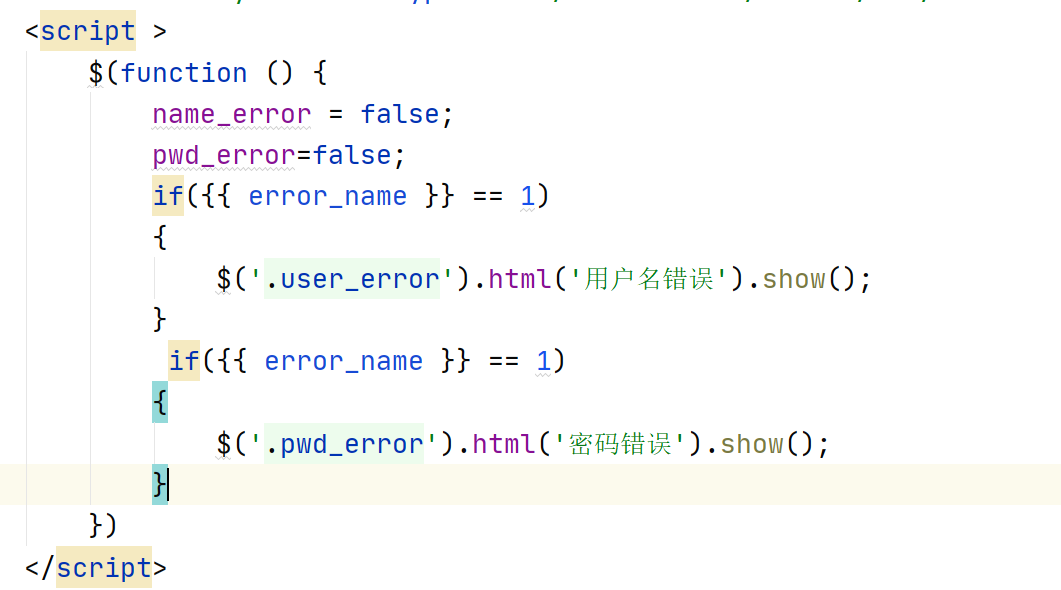




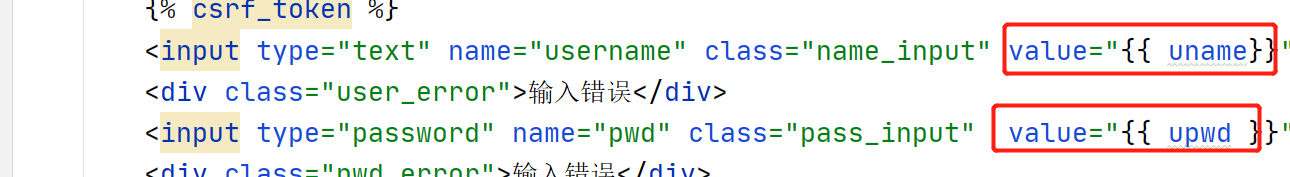


同时在html文件的<head>标签中添加如下代码





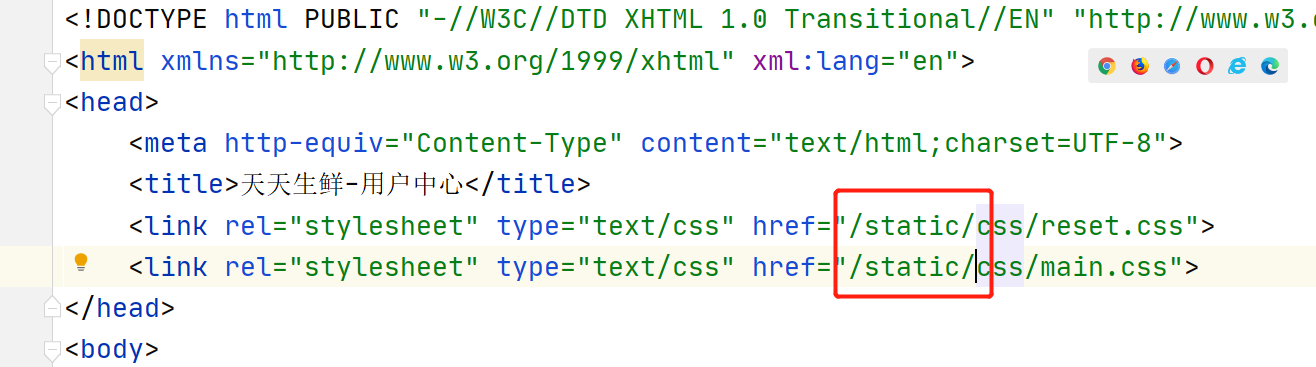
在<form>标签中添加如下代码



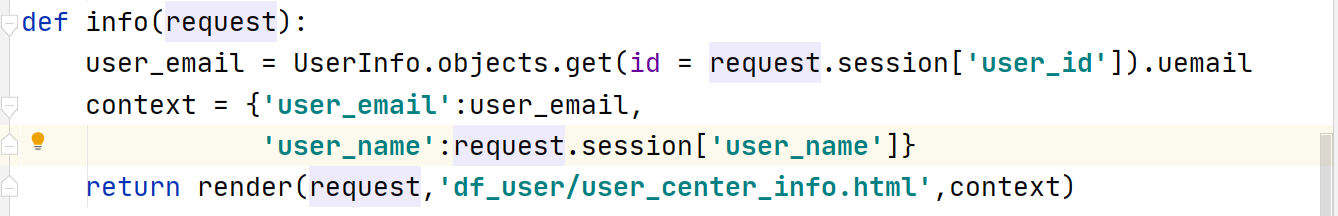
3 为user/info设置处理的路由函数，在df\_user的urls.py文件下添加如下代码



将user\_cetner\_info.html拷贝到template目录下的df\_user中，并对静态文件作出更改如下图



在views.py中填入以下代码



在user\_center\_info.html中作出以下更改



4 运行系统测试

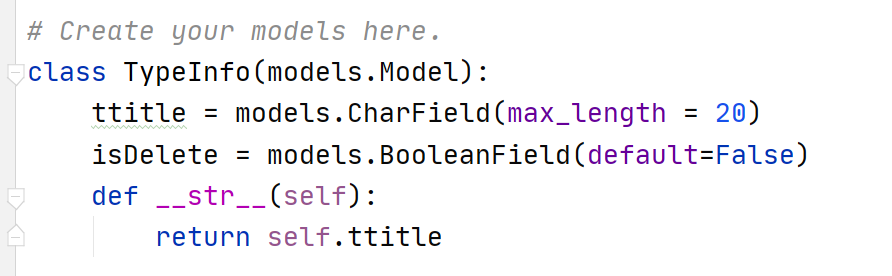


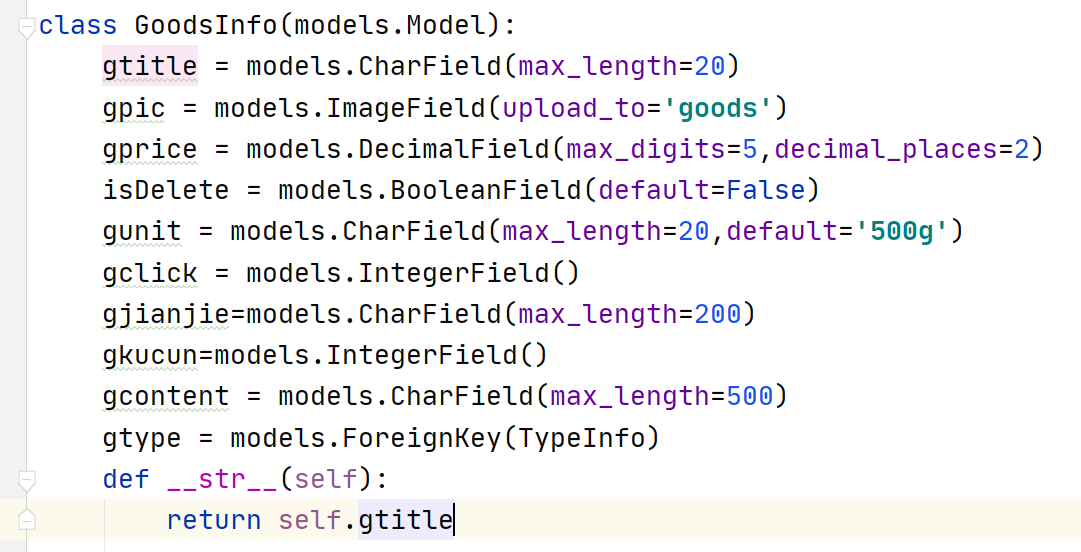
# 实现主页的功能

本章所有的功能都是在df\_goods目录下实现

# 6.1 设计模型类

设计两个模型一个商品的类型，另一个是商品本身，在df\_goods下的models.py中添加如下代码：

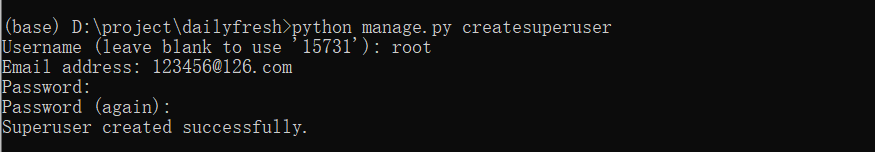




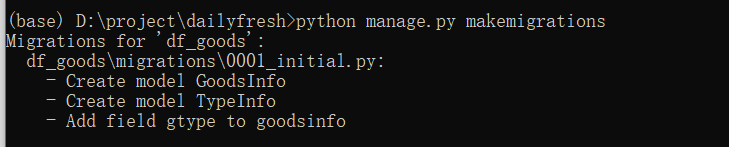
将模型注册到admin.py中，添加如下代码



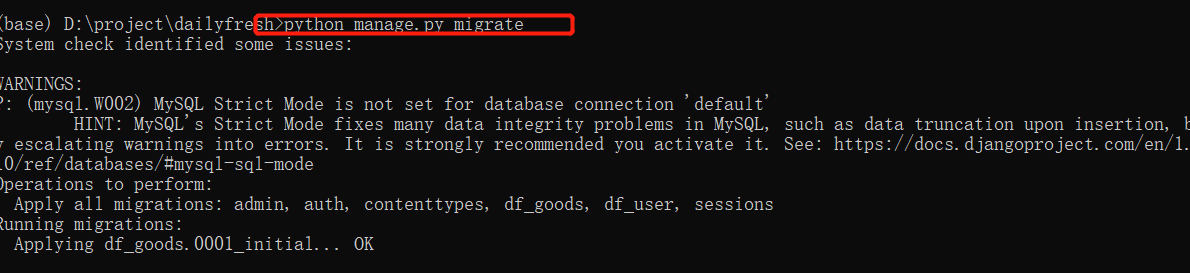
使用命令行创建超级用户



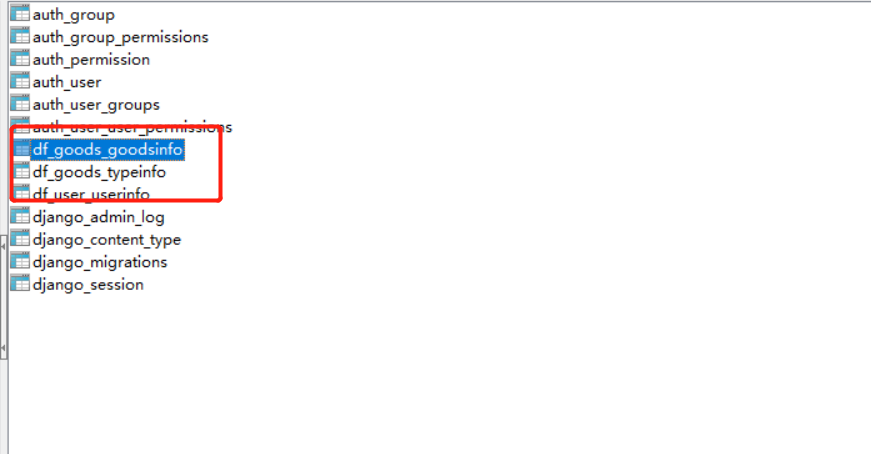
生成SQL语句



执行SQL语句在数据库生成相应的表。



在Navicat中检查表是否生成

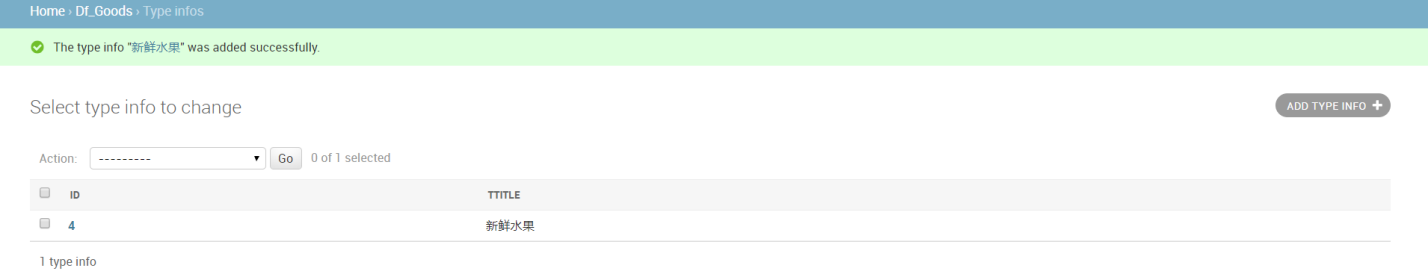
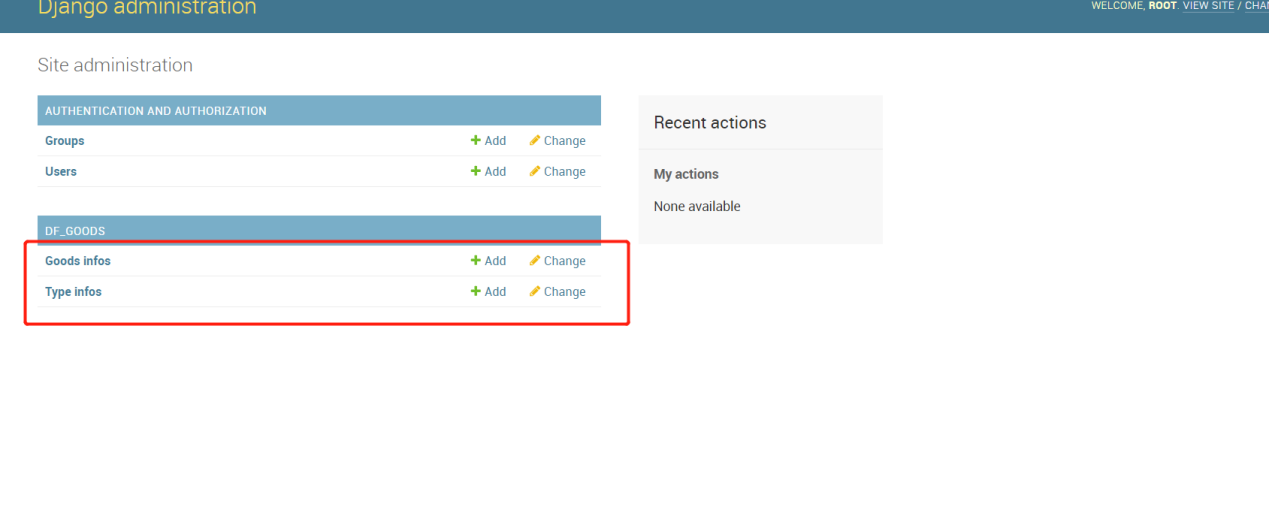


启动服务器

Python manage.py runserver

进入到管理员端

查看



## 主界面和业务服务器实现

1. 修改主页面的代码。



修改视图代码

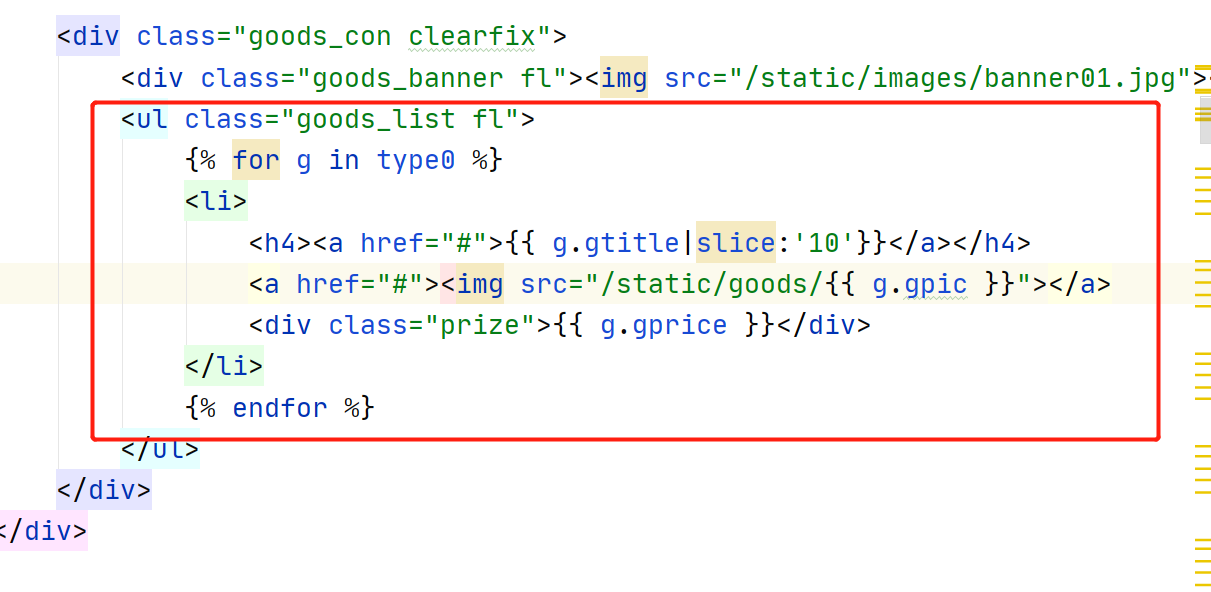
视图代码主要是从数据库中查找数据，然后传递到前端。

代码如下:



修改index.html代码用于接收服务器传递的数据

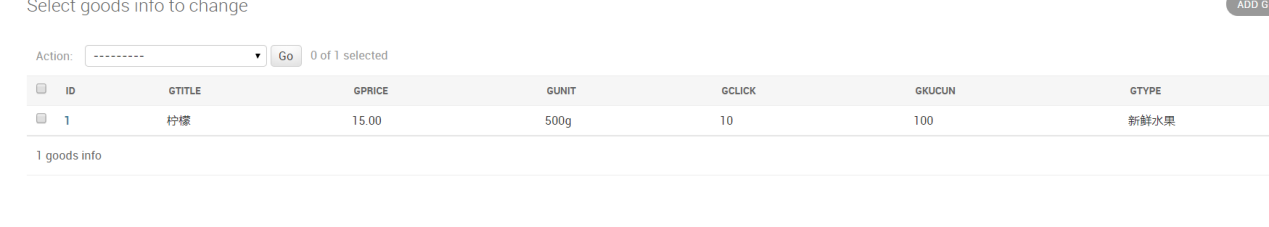




为上传成功图片还需要 在settings.py中添加如下代码



启动服务器 通过管理员添加数据。





返回首页查看结果