
基于 Python 语言 Django 框架的商品 管理系统

目录

前言.....	1
1 绪论.....	2
2 DJANGO.....	3
2.1 DJANGO 特点.....	3
2.2 DJANGO 工作方式.....	3
2.3 开发环境.....	4
2.4 安装 DJANGO.....	4
2.5 BOOTSTRAP 安装.....	5
3 主要功能.....	6
3.1 功能设计.....	6
3.2 主要界面.....	7
3.2.1 首页.....	7
3.2.2 所有商品（分页）.....	8
3.2.3 商品详情.....	8
3.2.4 管理员登录页.....	9
3.2.5 商品管理页.....	9
3.2.6 商品添加页面.....	10
4 代码实现.....	11
4.1 创建项目.....	11
4.2 设置数据库.....	11
4.3 设置国际化.....	12
4.4 配置静态文件.....	12
4.5 创建 DJANGO APP.....	12
4.6 创建 MODELS.....	13
4.7 运行程序.....	14
4.8 同步数据库.....	15
4.9 设置 ADMIN.....	15
4.10 创建超级用户.....	16
4.11 编写 TEMPLATE.....	16
4.12 编写首页和商品展示页（模板）.....	16
4.13 完善 VIEWS.....	16
4.14 实现分页（模板）.....	17
4.15 完善 URL.....	18
参考文献.....	20

前言

应用程序有两种模式C/S、B/S。C/S是客户端/服务器端程序，也就是说这类程序一般独立运行。而B/S就是浏览器端/服务器端应用程序，这类应用程序一般借助Chrome等浏览器来运行。WEB应用程序一般是B/S模式。Web应用程序首先是“应用程序”，和用标准的程序语言，如C、C++等编写出来的程序没有什么本质上的不同。然而Web应用程序又有自己独特的地方，就是它是基于Web的，而不是采用传统方法运行的。换句话说，它是典型的浏览器/服务器架构的产物。B/S结构能够很好地应用在广域网上，成为越来越多的企业的选择。

一个Web应用程序是由完成特定任务的各种Web组件（web components）构成的并通过Web将服务展示给外界。在实际应用中，Web应用程序是由多个模型(model)－视图(view)－控制器(controller)等组织起来的代码组成，这些组件相互协调为用户提供一组完整的服务。

Web应用程序对企业的重要用途是对数据进行处理，管理信息系统（Management Information System, 简称MIS）就是这种架构最典型的应用。MIS可以应用于局域网，也可以应用于广域网。基于Internet的MIS系统以其成本低廉、维护简便、覆盖范围广、功能易实现等诸多特性，得到越来越多的应用。

本文就是尝试利用Python语言实现商品管理系统，增强对Web应用系统需求、设计、开发的知识和能力。

1 绪论

Python 已经有将近 30 年的历史，在过去 30 年中，Python 在运维工程师和数据科学家群体中受到广泛欢迎，然而却极少有企业将 Python 作为生产环境的首选语言。在最近几年，这一情况有所改变。随着云计算、大数据以及人工智能技术的快速发展，Python 及其开发生态环境正在受到越来越多的关注。

互联网时代来临后，Python被用来在Web开发领域进行尝试，涌现出了一批基于Python 开发一些WEB的网站，还有不少大型的、基于Python 的网站，比如 Youtube、豆瓣等网站。使得一些Web开源框架迅速成长，如Django、Flask等，为程序员高效开发Web程序提供了巨大的帮助。

进入了云计算时代，基于过去一段时间 Python 在系统管理工具的积累，以及其本身具备了非常好的系统集成能力，Python 在云计算领域可以说大放异彩。最著名的是Python开发的Openstack。不仅在私有云领域，在公有云领域，包括 AWS，包括 Google 云，当这些公有云提供出 SDK 的时候，它们首选的技术路线依然是 Python。

最近两年又火起来的人工智能领域，Python靠着过去多年在科学计算等方面的积累出现了大爆发。比如图像识别用的都是 Python OpenCV库。在深度学习领域几乎没有任何其他语言可以跟 Python 相提并论的，比如 Caffe, Theano, TesnorFlow, Keras 这些非常流行的深度学习框架，都是以 Python 为主要开发语言。事实证明了在深度学习领域目前 Python 是处于非常主导地位。

如上所述，为了能跟上人工智能的潮流，从用户体验角度，从开发者角度来讲，Python 是更好的语言，也是更好的接口语言，值得我们学习和掌握它。另一方面，考虑到可以用 Python 集成各种各样的服务，这样能有效降低成本，同时也能够减轻自己开发团队的压力，让开发团队能够减少一些学习成本。

2 Django

Django是Python中目前风靡的Web Framework, 框架能够帮助我们把程序的整体架构搭建好, 而我们所需要做的工作就是填写逻辑, 而框架能够在合适的时候调用你写的逻辑, 而不需要我们去调用逻辑, 让Web开发变的更敏捷.

Django是一个高级Python Web框架, 鼓励快速, 简洁, 以程序设计的思想进行开发. 通过使用这个框架, 可以减少很多开发麻烦, 使你更专注于编写自己的app, 而不需要重复造轮子. Django免费并且开源.

2.1 Django 特点

- 1) 完全免费并开源源代码。
- 2) 快速高效开发。
- 3) 使用MTV架构(熟悉Web开发的应该会说是MVC架构)。
- 4) 强大的可扩展性。

2.2 Django 工作方式

用户在浏览器中输入URL后的回车, 浏览器会对URL进行检查, 首先判断协议, 如果是http就按照 Web 来处理, 然互调用DNS查询, 将域名转换为IP地址, 然后经过网络传输到达对应Web服务器, 服务器对url进行解析后, 调用View中的逻辑(MTV中的V), 其中又涉及到Model(MTV中的M), 与数据库的进行交互, 将数据发到Template(MTV中的T)进行渲染, 然后发送到浏览器中, 浏览器以合适的方式呈现给用户。

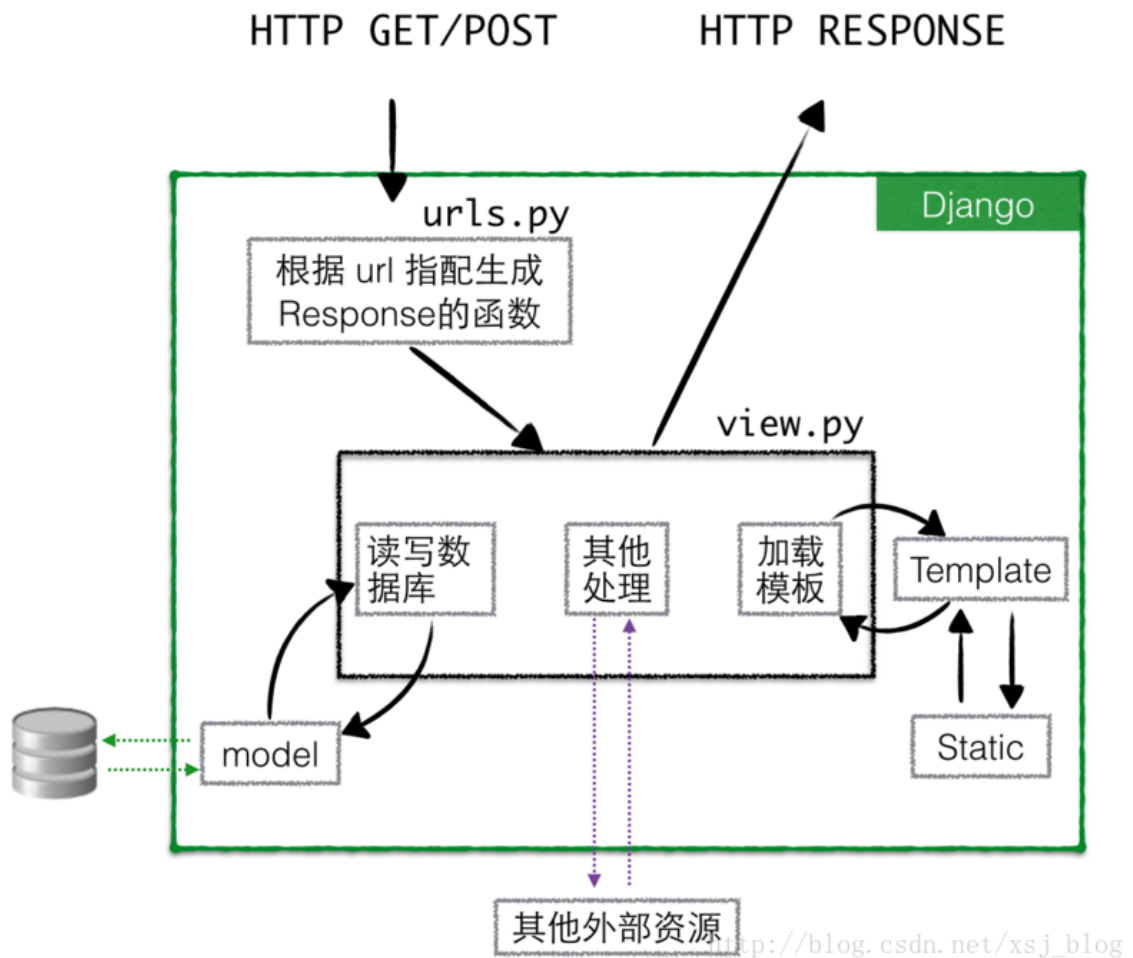


图 2-1 工作方式

2.3 开发环境

本文使用的主要开发环境为：

- 1) 操作系统：macOS Sierra 10.12.4
- 2) 开发语言：Python 3.6.1
- 3) Web 框架：Django1.11.1
- 4) 数据库：SQLite3
- 5) 前端框架：Bootstrap3.3.7
- 6) 开发 IDE：Pycharm CE 2017.1
- 7) 虚拟环境：Anaconda 4.3.17
- 8) 版本管理：GitHub

2.4 安装 Django

使用终端安装最新版的Django：

```
conda install Django
```

2.5 Bootstrap 安装

Bootstrap, 来自 Twitter, 是目前最受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的, 它简洁灵活, 使得 Web 开发更加快捷。

从官网下载所需资源, 稍后复制到开发目录中:

<http://getbootstrap.com/getting-started/#download>

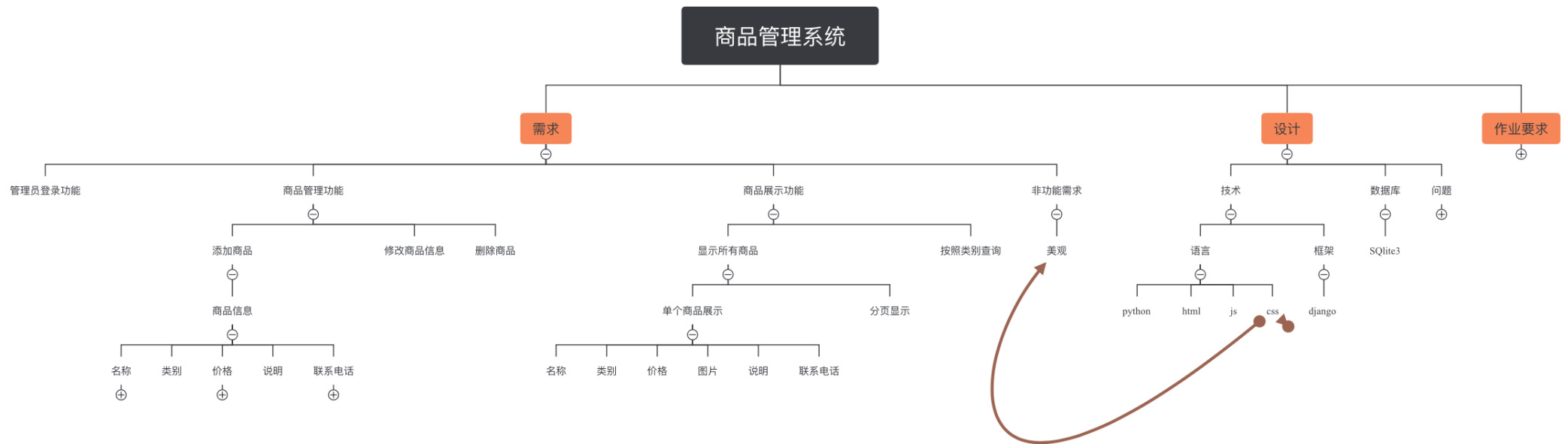
```
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   └── bootstrap-theme.min.css.map
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphs-halflings-regular.eot
    ├── glyphs-halflings-regular.svg
    ├── glyphs-halflings-regular.ttf
    ├── glyphs-halflings-regular.woff
    └── glyphs-halflings-regular.woff2
```

图2-2 Bootstrap目录

3 主要功能

3.1 功能设计

主要功能和设计考虑如图3-1所示：



3.2 主要界面

3.2.1 首页

最简单的商品管理系统

首页	所有商品	商品管理	后台管理	系统帮助	技术支持
----	------	------	------	------	------

Python无处不在！

Python 已经有将近 30 年的历史，在过去 30 年中，Python 在运维工程师和数据科学家群体中受到广泛欢迎，然而却极少有企业将 Python 作为生产环境的首选语言。在最近几年，这一情况有所改变。随着云计算、大数据以及人工智能技术的快速发展，Python 及其开发生态环境正在受到越来越多的关注。

Django是一个高级Python Web框架，鼓励快速、简洁，以程序设计的思想进行开发。通过使用这个框架，可以减少很多开发麻烦，使你更专注于编写自己的app，而不需要重复造轮子。Django免费并且开源。本程序基于Python语言Django框架开发。

本程序基于Python语言Django框架开发。

[点击阅读开发手册](#)



实木多抽柜
4199.00



高清头戴音乐耳机
699.00



便携充电器
79.00



空蓝经典飞行员墨镜
109.00



商务出行多功能双肩包
239.00



牛皮简约短款钱包
189.00



绅雅伯爵牛皮公文包
499.00



有田烧白金敞口杯
269.00



日式和风声波式电动牙刷 黑色
129.00



20寸 全铝镁合金登机箱
559.00



不锈钢炫彩渐变吸管杯
79.00



微电流滚轮身体按摩仪
189.00

3.2.2 所有商品（分页）

最简单的商品管理系统

首页



所有商品

商品管理

后台管理

系统帮助

技术支持

商品名称	商品类别	商品价格	登记日期	商品图片
实木多抽屉	家具	2017/05/28	4199.00	
高清头戴音乐耳机	数码	2017/05/28	699.00	
便携充电器	数码	2017/05/28	79.00	

1 2 3 4

何慧君 © 2017

3.2.3 商品详情

最简单的商品管理系统

首页

商品管理

后台管理

系统帮助

技术支持

首页 > 箱包



绅雅伯爵牛皮公文包

¥: 499.00

登记日期: 2017/05/23

商品介绍:
颜色黑色材质牛剖层移膜革规格39*30cm重量1kg注意事项
1.五金需防水; 2.皮质防压、防暴晒、防刮、防潮; 3.禁止
接触化学用品; 4.定期皮具专用油保养。

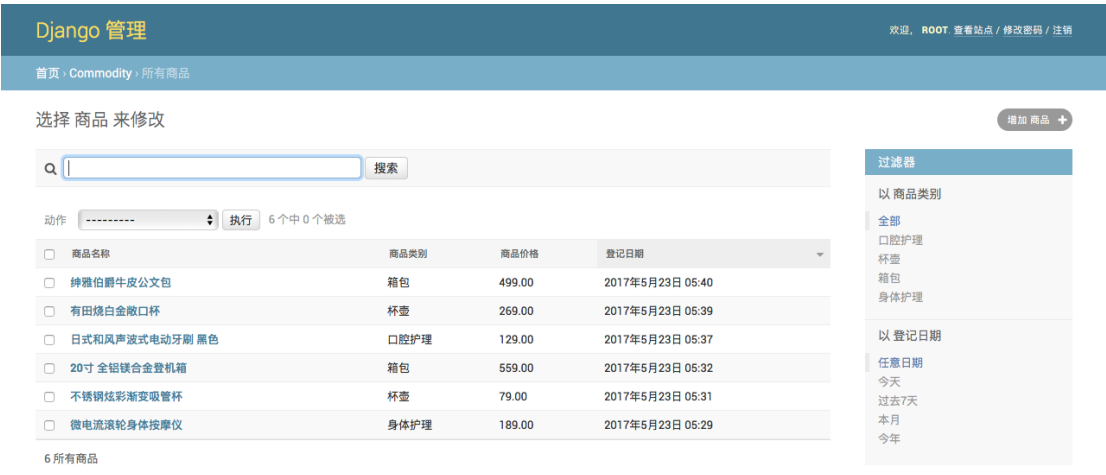
联系电话:
18612249696

何慧君 © 2017

3.2.4 管理员登录页



3.2.5 商品管理页



3.2.6 商品添加页面

Django 管理

欢迎, ROOT. [查看站点](#) / [修改密码](#) / [注销](#)

[首页](#) > [Commodity](#) > [所有商品](#) > [增加 商品](#)

增加 商品

商品名称:

商品类别:

商品价格:

商品图片:

选取文件

未选择文件

商品说明:

联系电话:

保存并增加另一个

保存并继续编辑

保存

4 代码实现

从现在开始正式的进入代码阶段，简述开发过程。

4.1 创建项目

我们创建一个名为WebStore的Django项目，创建项目的指令¹（终端）如下：

```
django-admin.py startproject WebStore
```

文件结构如下：

```
WebStore
├── manage.py
├── WebStore
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
```

4.2 设置数据库

Django项目建成后，就可以设置数据库了。本文默认使用SQLite数据库，在WebStore/WebStore/setting.py中可以查看和修改数据库设置：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

还可以设置其他数据库，如MySQL，PostgreSQL，现在为了便于发布和提供老师审核，使用默认数据库设置。

¹ 在 windows 下可以使用 `django-admin startproject WebStore` 的命令，无 `py` 后缀。

4.3 设置国际化

Django 支持国际化，多语言。Django的国际化是默认开启的，如果不需要国际化支持，可以在设置文件中设置 `USE_I18N = False`，那么Django会进行一些优化，不加载国际化支持机制。在WebStore/WebStore/setting.py修改默认的语言和时区。

```
LANGUAGE_CODE = 'zh-hans'
TIME_ZONE = 'Asia/Shanghai'
USE_I18N = True
```

4.4 配置静态文件

静态文件是指网站中的js,css,图片，视频等文件，通常在WebStore/WebStore/setting.py中已经设置好了。在WebStore根目录下创建static文件夹，在static下再创建upload、css、js、image等目录。

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"), '/Users/alexmac/Library/Mobile\
Documents/com~apple~CloudDocs/DEV/WebStore/static/', #物理路径
]
```

4.5 创建 Django app

在Django中的app被认为是一个功能模块，与其他的web框架可能有很大的区别，将不通功能放在不同的app中，方便代码的复用。在WebStore目录下，终端输入创建app指令：

```
python manage.py startapp commodity
```

WebStore根目录下，文件结构如下：

```
— commodity
| |— __init__.py
| |— admin.py
| |— migrations
| |   |— __init__.py
| |— models.py
| |— tests.py
| |— views.py
|— db.sqlite3
|— manage.py
|— WebStore
|— __init__.py
|— settings.py
|— urls.py
```

必须在WebStore/WebStore/setting.py下添加新建app

```
INSTALLED_APPS = (
...
'commodity', #这里填写的是 app 的名称
)
```

4.6 创建 models

在WebStore/commodity/models.py下编写如下程序：

```

class Commodity(models.Model):
    CommodityName = models.CharField('商品名称', max_length=100)
    CommodityCategory = models.CharField('商品类别', max_length=50,
blank=True)
    CommodityPrice = models.DecimalField('商品价格', max_digits=11,
decimal_places=2)
    CommodityImage = models.ImageField('商品图片', upload_to='static',
default='static/upload/None/no-img.jpg')
    CommodityDateTime = models.DateTimeField('登记日期',
auto_now_add=True)
    CommodityContent = models.TextField('商品说明', blank=True, null=True)
    CommodityContactMobile = models.CharField('联系电话', max_length=11,
blank=True, null=True)

    def __str__(self):
        return self.CommodityName # 一般系统默认使用 <Commodity:
Commodity object> 来表示对象，通过这个函数可以告诉系统使用 CommodityName 字
段来表示这个对象

    class Meta:
        verbose_name = '商品' #给模型起个更好听的名字，这儿相当于进行了汉
化。
        verbose_name_plural = '所有商品'
        # 按时间下降排序
        ordering = ['-CommodityDateTime']

```

4.7 运行程序

在WebStore根目录下，用终端命令行输入以下指令：

```
$ python manage.py runserver #启动Django中的开发服务器
```

启动浏览器，输入<http://127.0.0.1:8000/>，就可以访问网站了。

4.8 同步数据库

在WebStore根目录下，用终端命令行输入以下指令：

```
python manage.py migrate #同步在model中建立的数据库
```

如果对model进行了修改，需要先执行一次makemigrations命令，再执行migrate。

```
python manage.py makemigrations #先检查更新
```

4.9 设置 Admin

Django有一个优秀的特性，内置了Django admin后台管理界面，方便管理者进行添加和删除网站的内容。新建的项目系统已经为我们设置好了后台管理功能，可以在WebStore/WebStore/setting.py中查看。

在WebStore/commodity/admin.py中增加代码，让后台管理界面能对“商品”信息进行管理。默认管理界面中仅显示上面这是的“CommodityName”，为更方便的在后台管理信息，需要增加一些一些代码。具体的如下：

```
from django.contrib import admin
from commodity.models import Commodity

class AdminCommodity(admin.ModelAdmin):
    list_display =
('CommodityName', 'CommodityCategory', 'CommodityPrice', 'CommodityDateTime')
#后台显示的列表内容
    search_fields = ('CommodityName', 'CommodityCategory',)#从哪些字段中搜索
    list_filter = ('CommodityCategory', 'CommodityDateTime',)#筛选器
    admin.site.register(Commodity, AdminCommodity)
```

4.10 创建超级用户

初次登陆后台管理界面，需要使用如下命令账号创建超级用户：

```
python manage.py createsuperuser
```

按照提示输入用户名、邮箱、密码，创建第一个超级用户。

4.11 编写 template

在commodity目录下创建template目录，在template下增加base.html, 做为本文程序的基础模版。模版样式采用Bootstrap样式表。

4.12 编写首页和商品展示页（模板）

用{% extends "base.html" %}引入模版页面，增加需要展示商品的相关代码。

首页为index.html。〈代码见附录〉

商品展示页为post.html。〈代码见附录〉

4.13 完善 Views

通常访问网页程序的逻辑是：request进来->从服务器获取数据->处理数据->把网页呈现出来，Django也是按照这个流程来处理信息的，它使用url和views来处理：

- 1) url设置相当于客户端向服务器发出request请求的入口，并用来指明要调用的程序逻辑。
- 2) views用来处理程序逻辑，然后呈现到template(一般为GET方法，POST方法略有不同)。
- 3) template一般为html+CSS的形式，主要是呈现给用户的表现形式。

Django中views里面的代码就是一个一个函数逻辑，处理客户端(浏览器)发送的HTTPRequest，然后返回HTTPResponse。我们在WebStore/commodity/views.py中编写简单的逻辑：

```

from django.http import HttpResponse
from commodity.models import Commodity
from datetime import datetime
from django.http import Http404
from django.shortcuts import render
from django.core.paginator import PageNotAnInteger, Paginator

def list(request):#系统默认的Paginator
    limit = 3 # 每页显示的记录数
    post_list = Commodity.objects.all()#获取全部对象
    paginator = Paginator(post_list, limit) # 实例化一个分页对象
    page = request.GET.get('page') # 获取页码
    try:
        post_list = paginator.page(page) # 获取某页对应的记录
    except PageNotAnInteger: # 如果页码不是个整数
        post_list = paginator.page(1) # 取第一页的记录
    except EmptyPage: # 如果页码太大，没有相应的记录
        post_list = paginator.page(paginator.num_pages) # 取最后一
    页的记录
    return render(request, 'list.html', {'post_list': post_list})

def index(request):
    post_index = Commodity.objects.all()[:8]#获取全部对象,仅显示前8
    条数据
    return render(request, 'index.html', {'post_index': post_index})

def detail(request, id):
    try:
        post = Commodity.objects.get(id=str(id))
    except Commodity.DoesNotExist:
        raise Http404
    return render(request, 'detail.html', {'post': post})

```

4.14实现分页（模板）

用 `{% extends "base.html" %}` 引入模版页面，增加需要展示商品的相关代码。

在 `views.py` 中使用 `pagination`，实现简单分页。〈代码见附录〉

商品展示页为post.html。〈代码见附录〉

4.15完善 url

如何使这个逻辑在http请求进入时，被调用呢，这里需要WebStore/WebStore/urls.py中进行url设置，设置了访问主页（list.html）和商品详情页面（post.html）。代码如下：

```
from django.conf.urls import url,include
from django.contrib import admin
from commodity import views #1.8以上新的写法
urlpatterns = (
    url(r'^admin/', admin.site.urls),
    url(r'^$', views.index,name='index'), # alex: 1.8以上新的写法
    url( r'^(?P<id>\d+)/$',views.detail, name='detail'),
    url(r'^list/$',views.list),
)
```

参考文献

Django官方网站: <https://docs.djangoproject.com/en/1.11/>

Django基础教程: <http://code.ziqiangxuetang.com/django/django-tutorial.html>
