B2C电商平台项目开发

一.项目背景

购物网站是大家日常生活中必不可或缺的一部分，只有又网络和相应的设备就能做到足不出户的选购商品，并且可以享受商品送货上门的体验，本项目通过Django框架以及Mysql数据库搭建一个自己的电商平台。

二.实现功能

电商平台共分为前台和后台两个部分：前台主要实现商品展示及销售，后台主要是对商城中的商品信息、会员信息以及对订单信息等进行有效的管理。

网站前台：

- 网站首页商品展示：推荐商品，分类展示部分商品
- 商品列表页：分页展示某类别或指定条件的部分商品列表信息
- 商品详情页：通过商品ID号来展示指定商品详情信息
- 购物车管理：添加商品到购物车，查看购物车的商品信息，删除商品或清空购物车的商品信息
- 会员模块：注册、登录、退出及进入会员中心
- 会员中心：个人信息登录、订单信息

后台管理：

- 后台操作：登录、退出
- 会员信息管理：查看、修改会员状态
- 商品类别信息管理：添加、删除、修改、查看商品类别信息
- 商品信息管理：添加、删除、修改、查看
- 订单管理：查看订单、订单详情、处理订单

三.项目实现

- 创建项目

```
django-admin startproject SHOP
```

- 创建应用

```
python3 manage.py startapp common   # 前台后台通用应用
```

- 项目配置setting设置

添加应用到INSTALLED_APPS

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'common',
    'users',
    'goods',
    'cart'
]
```

数据库设置

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'Shop',
        'USER': 'root',
        'PASSWORD': 'westos',
        'HOST': 'localhost',
        'PORT': '3306',

    }

}
```

模板设置

```
...
'DIRS': [os.path.join(BASE_DIR,'templates')]
....
```

语言时区设置

```
LANGUAGE_CODE = 'zh-hans'

TIME_ZONE = 'Asia/Shanghai'
```

静态文件设置

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
]
```

图片上传设置

```
# 上传文件位置

# 只要是上传图片，一定要设置MEDIA_ROOT这个键，这个键的名字不能修改

MEDIA_ROOT = os.path.join(BASE_DIR, 'static/images')
```

- 设置完setting之后创建后台管理员

```
python3 manage.py createsuperuser
```

- 启动Django项目

```
python3 manage.py runserver
```

- 访问后台管理页面

```
http://127.0.0.1:8000/admin
```

后台管理

数据库设计

- 定义数据库模型

```
import datetime

from django.db import models

# Create your models here.

class Users(models.Model):
    # 用户信息模型
    username = models.CharField(max_length=32, verbose_name="账号")
    name = models.CharField(max_length=16, verbose_name="真实姓名")
    password = models.CharField(max_length=32, verbose_name="密码")
    sex = models.IntegerField(default=1, verbose_name="性别（1-男 2-女）")
    address = models.CharField(max_length=255, verbose_name="收货地址")
    code = models.CharField(max_length=6, verbose_name="邮政编码")
    phone = models.CharField(max_length=16, verbose_name="电话号码")
    email = models.CharField(max_length=50, verbose_name="邮箱地址")
    state = models.IntegerField(default=1, verbose_name="会员状态（1-启用 2-禁用 3-
后台管理员）")
    # 注册时间，auto_now_add：在创建时设置时间，后面修改时间不会改变
    addtime = models.DateTimeField(auto_now_add=True)
    modifytime = models.DateTimeField(auto_now=True)

    def gender(self):
        if self.sex == 1:
            return "男"
        elif self.sex == 2:
            return "女"
        else:
            return "未知"

    def toDict(self):
        return {
            'id' : self.id,
            'username' : self.username,
            'name': self.name,
            'password' : self.password,
            'address' : self.address,
            'phone' : self.phone,
            'email' : self.email,
            'state' : self.state,
            'addtime' : str(self.addtime),
            'modiftime' : str(self.modifytime)
        }

    def __str__(self):
        return self.name

    class Meta:
        # 单数时显示的名称
        verbose_name = "会员信息"
        # 复数时显示的名称
        verbose_name_plural = "会员信息"

class Types(models.Model):
    name = models.CharField(max_length=32, verbose_name="商品名称")

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = "商品分类信息"
        verbose_name_plural = "商品分类信息"

class Goods(models.Model):
    # 商品信息模型
    typeid = models.ForeignKey('Types', on_delete=False, verbose_name="类别ID")
    goods = models.CharField(max_length=32, verbose_name="商品名称")
    company = models.CharField(max_length=50, verbose_name="生产厂家")
    content = models.TextField(verbose_name="详情描述")
    price = models.FloatField(verbose_name="商品单价")
    picname = models.ImageField(upload_to='upload/%Y/%m', verbose_name="商品图
片")
    store = models.IntegerField(default=0, verbose_name="库存量")
    num = models.IntegerField(default=0, verbose_name="购买数量")
    clicknum = models.IntegerField(default=0, verbose_name="点击次数")
    state = models.IntegerField(default=1, verbose_name="商品状态（1-新添加 2-在售
3-下架）")
    addtime = models.DateTimeField(auto_now_add=True, verbose_name="添加时间")
    modifytime = models.DateTimeField(auto_now=True, verbose_name="修改时间")

    def __str__(self):
        return self.goods

    def toDict(self):
        return {
            'id' : self.id,
            'typeid' : self.typeid.id,
            'goods' : self.goods,
            'company' : self.company,
            'price' : self.price,
            'store' : self.store,
            'num' : self.num,
            'clicknum' : self.clicknum,
            'state' : self.state,
            'addtime' : str(self.addtime),
            'modifytime' : str(self.modifytime),
            'picname': self.picname.name
        }

    class Meta:
        # 单数时显示的名称
        verbose_name = '商品信息'
        # 复数时显示的名称
        verbose_name_plural = "商品信息"

class Orders(models.Model):
    # 订单信息
    uid = models.ForeignKey('Users', on_delete=False, verbose_name="用户id")
    linkman = models.CharField(max_length=32, verbose_name="联系人")
    address = models.CharField(max_length=255, verbose_name="地址")
    code = models.CharField(max_length=6, verbose_name="邮编")
    phone = models.CharField(max_length=16, verbose_name="联系电话")
    addtime = models.DateTimeField(auto_now_add=True, verbose_name="购买时间")
    total = models.FloatField(verbose_name="总金额")
    state = models.SmallIntegerField(verbose_name="订单状态（0-新订单 1-已发货 2-已收货 3-无效订单）")

    def __str__(self):
        return self.uid

    class Meta:
        verbose_name = "订单信息表"
        verbose_name_plural = "订单信息表"

class Details(models.Model):
    order_id = models.ForeignKey('Orders', on_delete=False, verbose_name="订单编号")
    goods_id = models.ForeignKey('Goods', on_delete=False, verbose_name="商品编号")
    name = models.CharField(max_length=32, verbose_name="商品名称")
    price = models.FloatField(verbose_name="商品单价")
    num = models.IntegerField(verbose_name="商品数量")

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = "订单信息详情表"
        verbose_name_plural = "订单信息详情表"
```

- 定义数据库模型的后台管理

```
from django.contrib import admin

# Register your models here.
```

```
from .models import Users, Goods, Types, Orders, Details

class UserAdmin(admin.ModelAdmin):
    list_display = ['pk', 'name', 'gender', 'address', 'phone', 'addtime']
    list_filter = ['name']
    search_fields = ['name']
    # 每页显示10条信息
    list_per_page = 10

admin.site.register(Users, UserAdmin)
admin.site.register(Goods)
admin.site.register(Types)
admin.site.register(Orders)
admin.site.register(Details)
```

完成数据库设计之后，生成迁移脚本，迁移数据库

```
python3 manage.py makemigrations

python3 manage.py migrate
```

前台管理

用户登录注册与注销模块

路由配置

- 子路由配置

  ```
   from django.conf.urls import url
   from users import views

    urlpatterns = [

    # 会员个人信息的路由配置

  url(r'^login$', views.login, name='login'),
  url(r'^register$', views.register, name='register'),
  url(r'^logout$', views.logout, name='logout')
  ]
  ```

- 添加子路由配置到项目路由配置文件中

  ```
  urlpatterns = [
      path('admin/', admin.site.urls),
      url(r'^', include('users.urls')),
  ]
  ```

视图函数

完成用户的登录，注册和退出

```
```

~~~
from django.http import HttpResponse, Http404
from django.shortcuts import render, redirect, get_object_or_404

# Create your views here.

from django.urls import reverse

from common.models import Users

def index(request):
    return HttpResponse('success')

def login(request):
    # 会员登陆表单
    if request.method == 'GET':
        return render(request, 'users/login.html')
    else:
        try:
            username = request.POST['username']
            password = request.POST['password']
            # 根据账号获取登陆者的信息
            user = Users.objects.get(username=username)
            # 判断当前用户是否后台管理员用户
            if user.state == 0 or user.state == 1:
                if user.password == password:
                    # 此处登录成功，将当前登陆信息放到session中，并调转页面
                    request.session['vipuser'] = user.toDict()
                    # //todo:
                    # return redirect(reverse('index'))
                    return HttpResponse('login success')
                else:
                    context = {'info': '登陆密码错误'}
                    print(context)
            else:
                context = {'info': '此用户为非法用户'}
                print(context)
        except Exception as e:
            # print(e)
            context = {'info': '登陆账号错误:' + str(e)}
            print(context)

```
    return render(request, 'users/login.html', context=context)
```

def logout(request):
    # 会员退出
    # 清除登陆的session信息
    del request.session['vipuser']
    # 调转登录页面（url地址改变）
    return redirect(reverse('login'))

def register(request):
    # 会员注册
    if request.method == 'GET':
        return render(request, 'users/register.html')
    else:
        # 获取post提交的数据
~~~

```
        username = request.POST['username']
        password = request.POST['password']
        try:
            # 根据账号获取登录者信息
            user = get_object_or_404(Users, username=username)
        except Http404 as e:
            user = None
        if user:
            context = {'info':'该用户名已存在，请重新输入'}
            print(context)
            return redirect(reverse('register'))
        else:
            user = Users(username=username, password=password)
            user.save()
            request.session['vipuser'] = user.toDict()
            context = {'info' : 'success'}
            print(context)
            return redirect(reverse('login'))
```

登录模板层

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap 101 Template</title>

```

<!-- Bootstrap -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
rel="stylesheet">

<!-- HTML5 shim 和 Respond.js 是为了让 IE8 支持 HTML5 元素和媒体查询（media queries）
功能 -->
<!-- 警告：通过 file:// 协议（就是直接将 html 页面拖拽到浏览器中）访问页面时 Respond.js 不
起作用 -->
<!--[if lt IE 9]>
  <script
src="https://cdn.jsdelivr.net/npm/html5shiv@3.7.3/dist/html5shiv.min.js">
</script>
  <script
src="https://cdn.jsdelivr.net/npm/respond.js@1.4.2/dest/respond.min.js">
</script>
<![endif]-->
```


  </head>
  <body>

  <p>{{ context.info }}</p>
   <form method='post' action="{% url 'login' %}" class="form-horizontal">
       <span style="color: red">{{ info }}</span>
       {% csrf_token %}
  <div class="form-group">
    <label  class="col-sm-2 control-label">Username</label>
    <div class="col-sm-10">
      <input type="text" class="form-control" id="inputEmail3" placeholder="用户
名" name="username">
    </div>

  </div>

  <div class="form-group">
    <label  class="col-sm-2 control-label">Password</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="inputPassword3"
placeholder="Password" name="password">
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <div class="checkbox">
        <label>
          <input type="checkbox"> Remember Me
        </label>
      </div>
    </div>
  </div>
  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn btn-default">Sign in</button>
    </div>
  </div>

</form>

` ` `
<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->
<script src="https://cdn.jsdelivr.net/npm/jquery@1.12.4/dist/jquery.min.js">
</script>
<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js">
</script>
` ` `

  </body>
</html>
```

注册模板层

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<p>{{ info }}</p>
<form method="post" action="{% url 'register' %}">
```

```
     <span style="color: red">{{ info }}</span><br/>
    {% csrf_token %}
    <input type="text" name="username" placeholder="username" ><br/>
    <br/>
    <input type="password" name="password" placeholder="password"><br/>
    <br/>
    <input type="submit" value="注册">
</form>
</body>
</html>
```

商品模块

商品列表与商品详情

路由配置

- 子路由配置

  ```
  urlpatterns = [
      url(r'^list$', views.lists, name='list'),
      url(r'^list/(?P<page>[0-9]+)$', views.lists, name="list"), # 分页商品列表展示
      url(r'^detail/(?P<gid>[0-9]+)$', views.detail, name="detail"),  # 商品详情
  ]
  ```

- 添加子路由配置到项目路由配置文件中

```
from django.conf.urls import url
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^', include('users.urls')),
    url(r'^', include('goods.urls')),

]
```

视图函数

```
from django.core.paginator import Paginator, PageNotAnInteger, InvalidPage, EmptyPage
from django.http import HttpResponse, Http404
from django.shortcuts import render

# Create your views here.
from common.models import Goods, Types
from common.views import loadinfo


def lists(request, page=1):
    """商品列表页（搜索&分页）"""
    context = loadinfo(request)
    # 获取商品信息查询对象
```

```
    goods = Goods.objects
    # 根据条件筛选商品列表
    tid = request.GET.get('tid', None)
    if tid:
        # 根据tid筛选
        goods = goods.filter(typeid=tid)
    kw = request.GET.get('kw')
    if kw:
        # 根据关键字模糊搜索
        goods = goods.filter(goods__contains=kw)
    goods = goods.all().order_by('-addtime')
    # 分页

    paginator = Paginator(goods, per_page=2)
    try:
        goods = paginator.page(page)    # 获取指定页的商品
        print("当前页的商品信息：", goods.object_list)
    except PageNotAnInteger:
        # 如果请求的页数不是整数，返回第一页
        goods = paginator.page(1)
    except EmptyPage:
        # 如果请求的页数不在合法的页数范围内，返回结果的最后一页
        goods = paginator.page(paginator.num_pages)
    except InvalidPage:
        # 如果请求的页数不存在，重定向页面
        return Http404('找不到页面的内容')

    # 封装信息加载模板输出
    context['goods'] = goods
    context['paginator'] = paginator

    print(goods)
    return render(request, 'goods/list.html', context)


def detail(request, gid):
    """商品详情页"""
    context = loadinfo(request)
    # 加载商品详情信息
    ob = Goods.objects.get(id=gid)
    ob.clicknum += 1   # 点击量加1
    ob.save()
    context['good'] = ob
    return render(request, 'goods/detail.html', context)
```

商品列表页模板层

```


<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap 101 Template</title>
```

~~~~
```
<!-- Bootstrap -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
rel="stylesheet">

<!-- HTML5 shim 和 Respond.js 是为了让 IE8 支持 HTML5 元素和媒体查询（media queries）
功能 -->
<!-- 警告：通过 file:// 协议（就是直接将 html 页面拖拽到浏览器中）访问页面时 Respond.js 不
起作用 -->
<!--[if lt IE 9]>
  <script
src="https://cdn.jsdelivr.net/npm/html5shiv@3.7.3/dist/html5shiv.min.js">
</script>
  <script
src="https://cdn.jsdelivr.net/npm/respond.js@1.4.2/dest/respond.min.js">
</script>
<![endif]-->
```

  </head>
  <body>

  <ul class="nav nav-pills">
  <li role="presentation" class="active"><a href="#">首页</a></li>
      {% for type in types %}
          <li role="presentation"><a href="/list?tid={{ type.id }}">{{ type.name
}}</a></li>
      {% endfor %}
  </ul>

  <br/>

  <form  method="get" action="/list">
      <input name="kw" type="text" placeholder="关键字">
      <input type="submit" value="搜索">
  </form>

```
<div class="row">
    {% for good in goods.object_list %}
        <div class="col-sm-6 col-md-4">
            <div class="thumbnail">
                <img src="/static/images/{{ good.picname }}" alt="..."
style="width: 200px;height: 220px">
                <div class="caption">
                    <h3>商品名称：{{ good.goods }}</h3>
                    <p>{{ good.context }}</p>
                    <p><a href="/detail/{{ good.id }}" class="btn btn-primary"
role="button">详情页</a>
                        <a href="/cart/add/{{ good.id }}" class="btn btn-
default" role="button">加入购物车</a></p>
                </div>
            </div>
        </div>
    {% endfor %}
```
~~~~

```
</div>

  <nav aria-label="Page navigation">
  <ul class="pagination">
    <li>
        {% if goods.has_previous %}
        <a href="/list/{{ goods.previous_page_number }}" aria-label="Previous">
            <span aria-hidden="true">&laquo;</span>
        </a>
        {% else %}
            <a class="disabled" href="#" aria-label="Previous">
                <span aria-hidden="true">&laquo</span>
            </a>
        {% endif %}
    </li>
    <li>
        {% if goods.has_next %}
        <a href="/list/{{ goods.next_page_number }}" aria-label="Next">
             <span aria-hidden="true">&laquo</span>
        </a>
        {% else %}
            <a class="disabled" href="#" aria-label="Next">
               <span aria-hidden="true">&laquo</span>
            </a>
        {% endif %}
    </li>
  </ul>
</nav>

```
<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->
<script src="https://cdn.jsdelivr.net/npm/jquery@1.12.4/dist/jquery.min.js">
</script>
<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js">
</script>
```


  </body>
</html>
```

商品详情页模板层

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
rel="stylesheet">

    <!-- HTML5 shim 和 Respond.js 是为了让 IE8 支持 HTML5 元素和媒体查询（media
queries）功能 -->
    <!-- 警告：通过 file:// 协议（就是直接将 html 页面拖拽到浏览器中）访问页面时
Respond.js 不起作用 -->
    <!--[if lt IE 9]>
      <script
src="https://cdn.jsdelivr.net/npm/html5shiv@3.7.3/dist/html5shiv.min.js">
</script>
      <script
src="https://cdn.jsdelivr.net/npm/respond.js@1.4.2/dest/respond.min.js">
</script>
    <![endif]-->
</head>
<body>

<ul class="nav nav-tabs">
    <li role="presentation" class="active"><a href="#">Home</a></li>
    {% for type in types %}
    <li role="presentation"><a href="#">{{ type.name }}</a></li>
    {% endfor  %}
</ul>


<div class="media">
  <div class="media-left">
    <a href="#">
      <img src="/static/images/{{ good.picname }}" alt="..." style="width:320px;
height: 320px">
    </a>
  </div>
  <div class="media-body">
    <h4 class="media-heading">{{ good.name }}</h4>
    {{ good.content }}
  </div>
</div>

<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->
<script src="https://cdn.jsdelivr.net/npm/jquery@1.12.4/dist/jquery.min.js">
</script>
<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js">
</script>
</body>
</html>
```

购物车模块

购物车添加，删除，清空

路由配置

- 子路由配置

```
from django.conf.urls import url

from cart import views

urlpatterns = [
# 购物车路由
url(r'^cart$', views.index,name='cart_index'), #浏览购物车
url(r'^cart/add/(?P<gid>[0-9]+)$', views.add,name='cart_add'), #添加购物车
url(r'^cart/del/(?P<gid>[0-9]+)$', views.delete,name='cart_del'), #从购物车中删除一个商品
url(r'^cart/clear$', views.clear,name='cart_clear'), #清空购物车
url(r'^cart/change/(?P<gid>[0-9]+)$', views.change,name='cart_change'), #更改购物车中商品数量
]
```

- 添加子路由配置到项目路由配置文件中

```
from django.conf.urls import url
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^', include('users.urls')),
    url(r'^', include('goods.urls')),
    url(r'^', include('cart.urls'))
]
```

视图函数

```
from django.http import HttpResponse
from django.shortcuts import render, redirect

# Create your views here.
from django.urls import reverse
from common.models import Types, Goods
from common.views import loadinfo



def index(request):
    """浏览购物车"""
    context = loadinfo(request)
    # 缓存/会话session中没有ShopList（购物车列表），默认指定为空
    if 'ShopList' not in request.session:
        request.session['ShopList'] = {}
    context['shoplist'] = request.session['ShopList']
    return render(request, 'cart/cart_list.html', context)

def add(request, gid):
    """在购物车中放入的商品信息"""
    # 获取要放入购物车中的商品信息
    goods = Goods.objects.get(id=gid)
    shop = goods.toDict()
    shop['m'] = int(request.POST.get('m', 1))

    # 从session获取的购物车信息，没有默认空字典
    ShopList = request.session.get('ShopList', {})
    # 判断该商品是否存在在购物车中
    if gid in ShopList:
        # 商品数量加
        ShopList[gid]['m'] += shop['m']
    else:
        # 新商品添加
        ShopList[gid] = shop
    # 将购物车信息放回到session中
    request.session['ShopList'] = ShopList
    # 重定向到浏览购物车页面
    return redirect(reverse('cart_index'))

def delete(request, gid):
    """删除一个商品"""
    ShopList = request.session['ShopList']
    del ShopList[gid]
    request.session['ShopList'] = ShopList
    return redirect(reverse('cart_index'))

def clear(request):
    """清空购物车"""
    request.session['ShopList'] = {}
    return redirect(reverse('cart_index'))

def change(request,gid):
    """更改购物车数量"""
    ShopList = request.session['ShopList']
    # 获取信息
    shopid = gid
    num = int(request.GET['num'])
    if num < 1:
        num = 1
    ShopList[shopid]['m'] = num   # 更改商品数量
    request.session['ShopList'] = ShopList
    return redirect(reverse('cart_index'))
```

购物车模板层

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
rel="stylesheet">

    <!-- HTML5 shim 和 Respond.js 是为了让 IE8 支持 HTML5 元素和媒体查询（media
queries）功能 -->
    <!-- 警告：通过 file:// 协议（就是直接将 html 页面拖拽到浏览器中）访问页面时
Respond.js 不起作用 -->
```

```
    <!--[if lt IE 9]>
      <script
src="https://cdn.jsdelivr.net/npm/html5shiv@3.7.3/dist/html5shiv.min.js">
</script>
      <script
src="https://cdn.jsdelivr.net/npm/respond.js@1.4.2/dest/respond.min.js">
</script>
    <![endif]-->
</head>
<body>

<h1>浏览购物车信息</h1>
<table class="table">
    <tr>
        <th>商品缩略图</th>
        <th>商品名称</th>
        <th>商品单价</th>
        <th>购买数量</th>
        <th>编辑数量</th>
        <th>删除</th>
    </tr>

    {% for key,good in shoplist.items %}
        <tr>
            <td><img style="width: 50px; height: 50px;" src="/static/images/{{
good.picname }}"></td>
            <td>{{ good.goods }}</td>
            <td>{{ good.price }}</td>
            <td>{{ good.m }}</td>
            <td>
                <form action="/cart/change/{{ good.id }}">
                    <input type="text" name="num" placeholder="购买数量">
                    <input type="submit" value="更改">
                </form>
            </td>
            <td>
                <a href="/cart/del/{{ good.id }}" class="btn btn-primary btn-
danger" role="button">删除</a>
            </td>
        </tr>

    {% endfor %}

</table>
<a href="/cart/clear" class="btn btn-primary btn-lg btn-danger align-right"
role="button">清空购物车</a>


<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->
<script src="https://cdn.jsdelivr.net/npm/jquery@1.12.4/dist/jquery.min.js">
</script>
<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js">
</script>
</body>
</html>
```