第 25 章

大型购物网站——

天天生鲜

如果要开发大型Web应用,比如京东网、淘宝网等购物网站,使用Python语言自己去处 理HTTP协议相关服务，数据库连接操作等，工作量巨大且很麻烦，所以我们通常会借助一些 成熟的Web开发框架,例如Djangoo本章就为大家介绍如何使用Django框架开发一个名为天 天生鲜的大型购物网站项目。

25O1 Django

Django是一个开放源代码的Web应用框架，由Python写成。它源自一个在线新闻Web站点， 于2005年开放源码。它采用了 MTV的框架模式，即模型M、模板T和视图V。Django框架 在近年来发展迅速，应用也越来越广泛。

25JJ 携闔 Djang©

使用Django框架可以省去处理通信协议的相关细节,把精力都放在网站的相关服务设计上, 而Django框架本身提供了一个网站所需要的程序代码，所以只要按照这些流程编写程序就可 以轻松完成许多原本非常复杂的事情。

另外，Djang。在设计的时候遵循了模块化的设计概念，把数据库和Python做了抽象化设计, 因此在网站设计中不需要使用SQL查询语言，而是使用Pythoii方法处理数据库中的数据，日 后如果需要更换数据库种类，也不至于修改大量程序代码。

因此，对于使用Python建设网站的初学者来说，一旦熟悉了 Djmigo的运行逻辑，就可以 在非常短的时间内构建一个出色的专业网站。

2SJ.2 Ojango

Django是一个基于MVC架构构造的框架，但是在Django中，控制器接受用户输入的部 分由框架自行处理,所以Django里更关注的是模型(Model)、视图(Views )和模板(Template ), 称为MTV模式。它们各自的职责如下:

。模型(Model)：用来构建和操作Web应用中的数据。负责业务对象和数据库的关系映 射(ORM)?负责处理与数据相关的所有事务，包括如何存取、如何验证有效性、包含 哪些行为以及数据之间的关系等。通常，每个模型对应数据库中唯一的一张表。

。视图(View):负责接收用户请求，进行业务处理，并返回响应逻辑。

。模® (Template):负责决定显示内容。模板包含所需HTML输岀的静态部分，以及一 些特殊的语法，描述如何将动态内容插入模板中。

除了以上三层之外,还需要一个URL配置文件(URLconf) 9它的作用是将一个个URL 的页面请求分发给不同的视图处理，视图再调用相应的模型和模板。URLconf机制是使用正则 表达式匹配URL,然后调用合适的Pythoii函数。URLcmf对于URL的规则没有任何限制，你 完全可以设计成任意的URL风格。

'MTV架构的流程如下所示。

(1 ) Web服务器(中间件)收到一个HTTP请求;

(2 ) Django在URLconf里查找对应的视图(View)函数来处理HTTP请求；

1. 视图函数调用相应的数据模型来存取数据、调用相应的模板向用户展示页面；

(4 )视图函数处理结束后返回一个HTTP的响应给Web服务器；

(5 ) Web服务器将响应发送给客户端。

在本章中，不会对Djang框架的使用进行单独介绍，而是结合天天生鲜项目的开发过程进 行介绍。

2①*2*夭无生鲜项目展示

天天生鲜网站是一个在线购物网站，提供生鲜商品的售卖功能。本项目包含6个页面，分 别是首页商品展示页面、商品分类展示页面、商品详情页面、购物车页面、订单提交界面、订 单成功显示页面。限于篇幅，这些页面只提供了用户浏览和购物的整体流程，但是不包含用户 管理和后台管理功能。大家在学会了如何实现这些页面之后，可以尝试着自己实现用户管理和 后台管理功能，从而实现一个完整的电商网站。

25.2J .

首页商品展示页面是天天生鲜网站的首页，也是用户访问网站时看见的第一个页面。该首 页负责向用户展示网站出售的所有商品类别，并展示商场的明星商品、促销活动等重要信息。 该页面展示的信息主要分为三部分：

(1)页面左上角展示商品的所有分类。

(2 )页面右上角展示“我的购物车”图标和购物车内的商品数量，当鼠标移动到该图标上时，会显示添加到“我的购物车”的商品列表。

(3)页面主体部分按照商品的分类展示最新商品，每个分类选择最新添加的四个商品进 行展示。

首页的页面展示效果如图254所示。

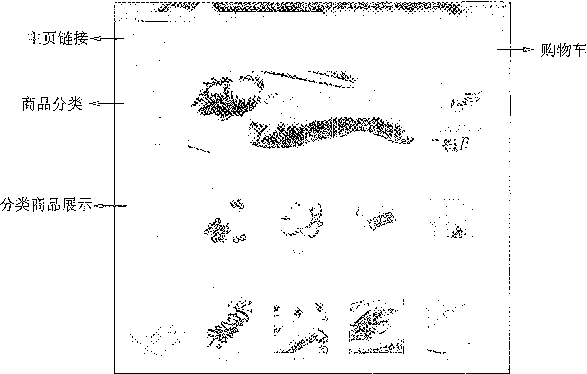


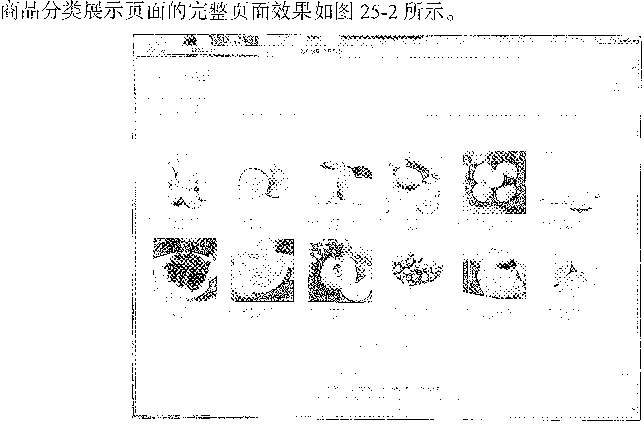
图254首页展示效果

25d2d2商靄翁羹屐康典逾

当单击主页上的某个商品类别，即可进入商品分类展示页面。该页面负责分页展示某个具 体商品类别下的所有商品。商品分类展示页面展示的信息主要分为两部分：

(1)页面右上角展示的“我的购物车”图标和购物车内商品数量。

(2)页面主体部分展示当前分类下的所有商品，并使用分页显示，每页展示12个商品。



25.2.3

当单击某个商品的图片或名称时，页面跳转到商品详情页面。该页面主要展示某个具体商 品的详细信息（如商品图片、名称、价格、描述等），以及提供购买链接。当用户单击该页面的“加 入购物车”按钮，会将该商品加入到购物车中。

商品详:情页面的展示效果如图25-3所示。

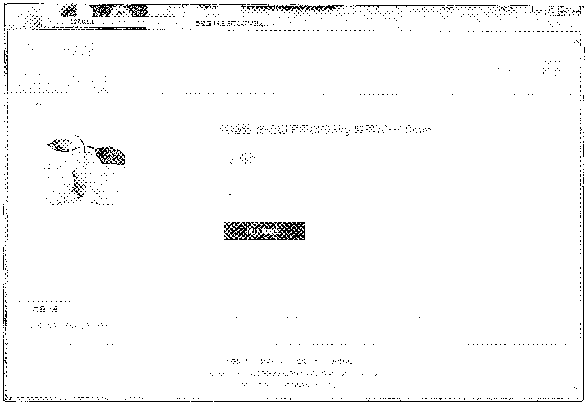
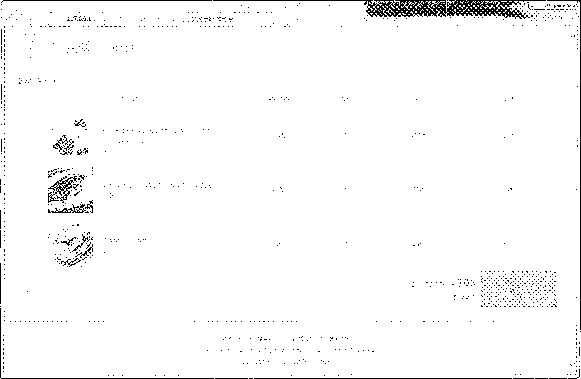


图25；商品详情页面

25.2.4购物军炭衝

单击“我的购物车”图标即可进入购物车页面，该页面展示了用户添加到购物车的商品列 表和详细信息，用户可以对购物车内的商品执行删除操作。

购物车页面的完整展示效果如图25-4所示。

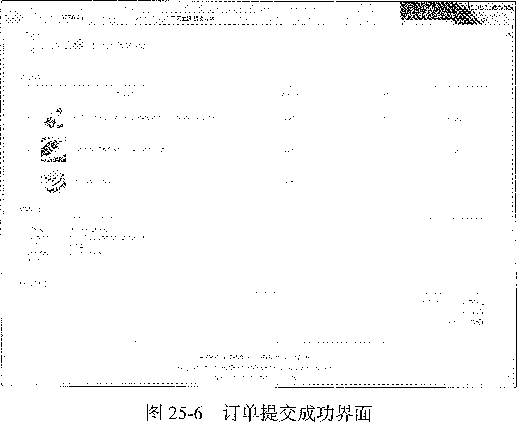


2S.2.S "

用户在购物车页面确认要购买的商品之后，单击“去结算”链接，可以进入订单提交页面。 该页面列出要购买的商品列表，用户需要填写收货地址、收货人、联系电话、备注等信息。填 写完毕，单击“提交订单”按钮，系统生成订单并存储在数据库中。订单提交页面展示效果如 图25-5所示。



用户提交订单之后，进入订单提交成功页面。该页面显示该订单的详细信息。 订单提交成功页面的展示效果如图25・6所示。



\* 项同側建

接下来，就可以准备使用Django框架开发天天生鲜项目了。我们并不会单独讲解Django 框架的使用,而是通过开发天天生鲜项目，在开发过程中结合对Django框架的介绍，使大家 一步步掌握Django框架的基本使用。

25.3J

与其他项目类似，本项目基于Windows 7操作系统，使用的开发语言是Python 3&开发 工具是PyCharm。除此之外，还需要具备以下开发环境：

° Django框架，版本是182。之所以选择Django L82?是因为该版本稳定性高、使用广、 并且文档多。

° PyMySQL包，版本是080。由于项目使用MySQL数据库，所以需要安装PyMySQL包。 在Windows 7系统下,可以使用pip命令安装这两个框架和组件。打开命令行管理工具, 执行以下两个命令即可:

pip install django==l.8.2

pip install pymysql

25.3.2

开发环境准备好之后，就可以创建项目了。我们需要用代码来创建一个Django项目，即 一个Django框架开发的网站。

创建Django项目的命令如下:

django-admin startproject 项 目名称 -

打开命•令行,执行Cd命令,切换到要保存项目的目录下（例如本章使用的“D: \PythonProject,， 目录），并执行以下命令:

django-admin startproject ttsx

此时,我们已经成功创建一个名为ttsx的Django 项目。使用PyCh皿打开该项目，可以看到项目的结构 如图25"所示。

V 縁 ttsx

settings.py

篠讎

経 wsgi.py

*¥矗* msnage^py

图25・7项目结构

其中，各个模块文件的介绍如下所示。

® manage.py文件:项目运行的入口文件,执行项 目配置文件路径。

• —init .py文件:空文件,指定当前目录可作为 包使用。

* settings.py文件：整个项目的配置文件，例如配置应用、模板目录、静态文件目录等。
* urkpy文件：是项目的URL配置文件，在该文件中将用户请求的URL对应到某个视图 函数。

• wsgi.py文件:项目与支持WSGI协议的Web服务器对接的入口文件。

25.3.3劍篷應願

Django框架通过应用的方式来管理整个网站项目。一个网站中包含多个子业务模块，比如 用户模块，商品模块，新闻模块等。我们可以将这些子模块称作一个应用（App）。通过项目 页面展示的功能，我们可以将项目分为两个应用，分别是goods和cart。其中：

•和商品展示相关的应用，叫作goods应用。

•和购物车相关的应用，叫作cart应用。

每一个应用中包含了该子业务所有的操作。创建这两个应用的命令如下所示。

python manage »py startapp goods python manage.py startapp cart

在命令行执行上述两个命令,就可以创建两个名为 goods和cart的应用了。此时项目的目录结构如图25书所示。

ttsx项目中增加了名为cart和goods的两个目录,两个 目录内所包含的文件是一模一样的。新建应用中所包含的 文件含义介绍如下：

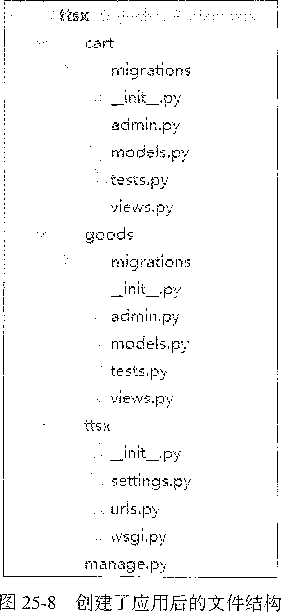
* \_init\_.py文件:一个空文件,表示当前目录可以 当作一个Python包来使用。

*鸯*tests.py文件:用于开发测试用例。在实际开发中, 如果需要对模块进行测试，可在此文件中编写测试 代码。

* views.py文件:视图文件,在该文件中编写视图相 关代码。
* models.py文件:模型文件,在该文件中编写模型相 关代码。

*佥*migrations包：与模型移植有关，稍后讲解模型创建 时会进行讲解。

要注意的是，创建的应用需要在settings.py模块中进 彳亍配置才能够被项目识别o为此，打开ttsx包下的settings.

py模块，找到INSTALLED\_APPS配置项，该配置项列出了已经安装的应用。在该配置项下加

入cart和goods应用9添加后的配置项如下所示。

INSTALLED\_APPS =(

* dj ango。contrib.admin'f
* dj ango。contrib.auth ?,

1dj ango.contrib.contenttypes',

® dj ango。contrib.sessions 17 星 dj ango.contrib.messages J/

, dj ango。contrib。staticfiles J r

cart® *v*

至此，天天生鲜网站的项目以及应用就创建完毕了。

25.3.4 .

在开发阶段，为了能够快速预览到网站的显示和运行效果，Django提供了一个纯Python 编写的轻量级Web服务器，仅在开发阶段使用。运行开发服务器的方法是，打开终端，使用 cd命令进入ttsx项目所在的目录下（即manage.py文件所在的目录），然后运行如下命令。

python manage.py runserver ip: 端口

实际应用中，可以不指定IP和端口，如下所示。

python manage epy runserver

此时使用的是系统默认的IP和端口。开发服务器成功启动后的效果如图25・9所示。

ps KPythonPro JectXttsx>python nanag-e run server —

^Performing' system checks . „« '

^ysteim check identified no issues <0 silenced）=

谷 on have unapplied nigrat ions ； j/our app not work properly until-they are app lied»

限tin J python manage migrateJ to applst them„ ■ .

^ebyu..ary 01, 2018 ™ 11 s!5 ：28

©Jango uersion 1 = 8-2, using- setting-：s ttsx-setting-sJ

Starting deuelopment seruey at http://127-0\_0.1s8000/

Quit the server with CTRL~BREfiK„ ；

图2们9服务器启动后效果

此时，在浏览器中输入网址"27。0丄8000”，可以看到图25M所示的显示结果须表示 开发服务器能够正常访问。在往后的项目开发过程中，就可以使用该网址和端口查看当前项目 的开发效果了。

.-127^0.1 . • •- - vVelcome ta Cyango

I !t worked?

I C。厂:密"滋权"舟苦終’#"广菸 *沁翌.'方•*

*! Qi* course, you haven't 8cfcaHy dore ary work yet Next, start your First app by running python *gg泸.河 曰■:\*有* 物弦\_丄云盗％：-

| Yaifre seeing thfe message because yau hsve 理ehw，=二亍工 in yaur Django settings fife and you haven't configured ^ny URLs. Get ta-说odd

J

图2540开发服务器正常访问

如果在项目中执行了增加、修改、删除文件，开发服务器会自动重启，不需要手动重启。 如果要停止开发服务器，可以使用组合键［Ctrl+C ］或［Ctrl+Break ］ o

25o4刨建模型

25.4J ◎底關滅蠢 」. -

当前项目的开发都是数据库驱动，即分析出项目中所需要存储的数据，然后设计数据表结

构，接下来通过编写SQL语句对数据库中的表进行CURD (增查改删)操作。

Django提供了一套针对数据库操作的代码库，通过该代码库中提供的工具，可以创建查询、 生成SQL语句，并且这些SQL无须开发人员自己处理。这样的好处是开发人员无须对SQL和 关系型数据库有深入的了解就能直接编写自己的查询。

我们把这套库称作ORM库,ORM是对象关系映射,对象表示的是Python中的对象概念, 关系指的是关系型数据库。通过ORM完成对象和数据库中的表之间的——对应关系。对对象 的任何操作就是对表的操作。

另外ORM强大的地方在于它能支持多种数据库后端，可非常简单地帮助我们切换后端数 据库o Django中提供了自己内置的ORM库。我们不需要面向数据库编程，而是通过定义模型类， 通过对模型类和对象的方法调用，完成对数据库的CRUD操作。

简而言之,ORM库封装了对关系型数据库的CRUD操作,我们不需要编写一行SQL语句, 就可以完成对数据库的操作。

打开ttsx/settings.py文件，在该配置文件中默认使用的是SQLite数据库,原SQLite配置内 容如下：

DATABASES = {

1 default \*: {

1. ENGINE 1: 'django•db.backends.sqlite3'F 'NAME®: osepath.join(BASE\_DIR,'dbesqlite31),

}

}

天天生鲜项目使用的是MySQL数据库，所以要对该配置内容进行修改，改为MySQL数 据库的对应配置，内容如下：

DATABASES = {

|  |  |  |  |
| --- | --- | --- | --- |
| 1 default': ( | | | |
| 'ENGINE1 | :!django.db.backends.mysql1, | # | mysql数据库引擎 |
| FAME® : | 1ttsx', | # | 数据库名字 |
| 'USER': | 'root5 *耳* | # | 用户名 |
| !PASSWORD1: 123456, | | # | 密码 |
| "HOST，: | \* localhost, p | # | 主机 |
| 'PORT ?:  } | '3306, | # | 端口 |

其中用户名、密码、主机和端口号可以根据实际情况进行修改。

配置完成之后，在ttsx目录下的\_init\_.py文件中增加如下代码。

import pymysql

#当成mysqldb 一样使用 pymysql.install\_as\_MySQLdb()

要注意的是，一定要保证配置文件中指定的数据库是存在的，如果不存在需要手动创建对 应的数据库。

接下来启动开发服务器，如果没有报错，则表示数据库配置成功。

在Django中进行数据库开发的主要步骤如下:

(1 )定义模型类：在models.py模块中定义模型类。

1. 创建迁移文件：根据模型类创建用于生成SQL的配置文件。
2. 执行迁移言根据SQL配置文件创建SQL语句并执行。

(4 )数据库操作:通过模型类中的方法对数据库执行CRUD操作。

接下来就按照这几个步骤完成天天生鲜项目的模型创建功能。

Django中的一个模型类对应着数据库中的一张表，对模型类的任何操作都是对数据库表的 操作。基于对天天生鲜项目的业务分析可知，该项目一共需要四张数据库表，分别介绍如下M

(1 )关于商品展示部分，需要两张数据库表：

® GoodsCafegory表，用于存储商品分类信息，具体信息如下：

-分-类显示名称。

■分类显示样式。

-分-类显示图片。

。Goodslnfo表，用于存储商品具体信息，具体信息如下：

■商品名称。

-商品价格。

■商品图片。

-商品描述。

-商品分类。

(2)关于购物车部分，也需要两张数据库表：

° Orderinfo ?存储订单基本信息，具体信息如下：

-订单编号。

-收货地址。

-收货人。

-联系电话。

-订单运费。

-订单备注。

-订单状态。

° OrderGoods表,存储订单所对应的商品信息，具体信息如下:

口订单商品。

■商品数量。

-所属订单。

接下来针对分析得到的表结构，创建对应的模型。要注意的是，自定义模型类需要定义在 models.py模块中，并且该类必须继承自models.Model类。另夕卜，定义模型类时可不指定主键，

在模型迁移过程中，Django框架会自动帮我们生成主键，主键类型为自动增长类型。

首先创建商品展示相关的两个模型类(GoodsCategoty和Goodslnfo ),为此，在goods应 用的models.py模块中添加如下代码：

from dj ango o db import models

class GoodsCategory(models oModel):

商品分类模型心E

#分类名称

cag\_name = models.CharField(max\_length=30)

#稣类样式 ~

cag\_css = models . CharField (max\_length:=20)

#务类图片 —

cag\_img = models «ImageField (upload\_to:= \* cag ?)

class Goodslnfo(models.Model):

5"商品信息模型"e，

#商品名字

goods\_na.me = models . CharField (max\_length=l 0 0)

#商品侨格 一

goods\_\_price = models . IntegerField (default=0)

#商品图片

goods\_\_img = models . ImageField (upload\_to= , goods !)

#商品I苗述 ~

goods\_\_desc = models . CharField (max\_\_length=2000)

#所属务类 •

goods\_cag = models„ForeignKey(? GoodsCategory1)

要注意的是，每一个类属性都和数据库表中的一个字段对应。

类属性后面的字段类型是由Django定义的，方便将字段类型映射到不同的数据库中。例 如下述代码：

goods\_\_name = models.CharField(max\_length=100)

上述代码中，字段名叫作goods\_name,字段的类型为CharField,对应到MySQL数据库的 字段类型为varchar类型,并且长度为100。

接下来创建购物车相关的两个模型类(Orderinfo和OrderGoods )*。*为此，在cart应用的 models.py模块中添加如下代码:

from dj ango o db import models

class Orderinfo (models ..Model):

”，顷订单信息模型H n n

status =(

dz ,待付款\*

(2, 5待发货\*

(3,-待收货\*

(4,已完成!),

)

#订单编号

order\_id = models.CharField(max\_length=100)

#收货地址

order\_addr = models.CharField(max\_length=100)

#收货人

order\_recv = models.CharField(max\_length=50)

#联系电话

order\_tele = models CharField(max\_length=ll)

#运费— —

order\_fee = models.IntegerField(default=10)

#订单盔注

order\_extra = models.CharField(max\_length=200)

#订单状态

order\_status = models eIntegerField(default=l,choices^status)

class OrderGoods(models.Model):

""H订单商品模型° ° °

#所属商品

goods\_info = models o ForeignKey('goods.Goodslnfo s)

#商品数量

goods\_num = models . IntegerField.()

#商品所属订单

goods\_order = models.ForeignKey(10rderinfo f)

其中有如下的代码：

order\_\_status = models.IntegerField(default=lf choices=status)

该代码中的choices=status表示该列的值为status指定的4个值中的一个。

在代码 goods\_infd = models.ForeignKey(!goods.GoodsInfos)中使用了 models ToreignKey^® ? 该字段主要用于建立外键，表示表和表之间的一对多关系。商品分类和商品信息表之间就属于 一对多的关系，所以需要在商品信息模型中建立对商品分类的外键。

(泞多学一招：Django框架的字段类型

|  |  |
| --- | --- |
| Django框架的常用字段类型以及含义如表25』所示。 表25T常用字段类型 | |
| 字段类型 | 描 述 |
| AutoField | 自动增长的IntegerField,通常不用指定，不指定时Django会自动创 建属性名为id的自动增长属性 |
| BooleanField | 布尔字段，值为True或False |
| NullBooleanF ield | 支持Null, True, False三种值 |
| CharF ield(max\_\_length=长度) | 字符串，参数rnaxjength表示最大字符个数。CharField必须指定 max\_length 大小 |
| TextField | 大文本字段 |
| IntegerField | IntegerField 一个整数。在Django所支持的所有数据库中， 从-2 147 483 648到2! 47 483 647范围内的值是合法的 |

|  |  |
| --- | --- |
| 续表 | |
| 字段类型 | 描 述 |
| DecimalField(max\_digits=None, decimaljplaces=None) | 十进制浮点数。参数max\_digits表示总位数。参数decimal\_places表  示小数位数 |
| DateField[auto\_now=False, auto\_now\_add=False]) | 日期。参数auto\_now表示每次保存对象时，自动设置该字段为当前 时间，用于“最后一次修改”的时间戳，它总是使用当前日期，默认 为falseo参数auto\_now\_add表示当对象第一次被创建时自动设置当前 时间，用于创建的时间戳，它总是使用当前日期，默认为false。参数 auto\_now\_add和auto\_now是相互排斥的，组合将会发生错误 |
| TimeField | 时间，参数同DateField |
| DateTimeField | 日期时间，参数同DateField |
| ' FileField | 上传文件字段 |
| ImageField | 继承于FileField,对上传的内容进行校验，确保是有效的图片 |
| BiglntegerField | 一个64位整数，类似于一个IntegerField ,它的值的范围是  -9 223 372 036 854 775 808 〜9 223 372 036 854 775 807 |
| SmalllntegerF ield | 该字段取值范围是-32 768-32 767 |

每个字段都会有一些字段选项，如表25-2所示。

|  |  |
| --- | --- |
| 表25-2字段常用选项 | |
| 选项类型 | 描 述 |
| null | 如果为True,表示允许为空，默认值是Fa屁 |
| db\_column | 字段的名称，如果未指定，则使用属性的名称 |
| dbindex | 若值为True,则在表中会为此字段创建索引，默认值是False |
| default | 默认值 |
| primary\_key | 若为True,则该字段会成为模型的主键字段，默认值是False, 一般作为AutoField的选项使用 |
| unique | 如果为Tnie,这个字段在表中必须有唯一值，默认值是Faise |

这些常用选项中，primary\_key的默认值是Falseo如果为True,那么这个字段就是模型的 主键。如果没有指定任何一个字段的primary\_key=True, Django就会自动添加一个IntegerField 字段作为主键，所以除非想覆盖默认的主键行为，否则没必要设置任何一个字段的piim&ry\_ key-Trueo主键字段是只读的，如果在一个已存在的对象上面更改主键的值并且保存，一个新 的对象将会在原有对象之外创建出来。

25n4a4創蠶迸滲気瓣

模型类定义完成之后,Django框架就可以按照已定义的模型类来创建对应的数据库表了。 创建数据库表分为两步:先创建迁移文件,然后执行该迁移文件。在此之前,我们先查看一下 项目的目录结构，会发现goods应用和cart应用的migrations目录都为空。在执行完迁移命令 之后，migrations目录会发生变化。

打开终端，进入项目所在目录，然后执行以下模型迁移命令：

python manage.py makemigrations

模型迁移命令的执行效果如图2541所示。

D = ^ythonPro ject \ttsx>p«ython manage - make miff rat ions ilig-rations for g,oods,:

01 \_in i t ia 1 o ： -

-Create model GoodsCateg-oy> \*

~ Create model GoodsInfo

iligrations J!cartf：

ialDio:

-Create model OrderGoods

- Create model Order!nfo

-ftdd iFie M gjoodsjrder to orderffoods

图2541模型迁移命令执行效果

现在再次查看migrations目录顼会发现该目录下增加了名为0001\_initiaLpy的文件。该文

件就是当前应用的迁移文件，只需使用如下命令，就可以查看该文件对应的SQL语句：

python manage.py sqlmigrate goods 0001

上述命令表示将goods应用下的0001迁移文件转换为SQL语句,命令的执行效果如图25-12 所示。

D s SP^thonPro ject Xttsx>»thon manage rapy sqlinigrate goods 0001 :

BEGIN；

CREATE TABLE ' goods\_\_^oodscateSopy' LLT integer fiUTOJNCHEMENT NOT NULL PRIMfiiffl

KEV, ^cag-\_name varcharOB) NOT NULL, ^cagjss11 yarchar(20> NOT NULL, Wa■題\_i冊旷 yarchaF<100> NOT NULL)；

CREfiTE TABLE 1 goods\_goods info1 C'id1 integer AUTOJNCBEMENT NOT NULL PHIMfiM *m* ,agoods\_jiame11 varchar<100> NOT NULL, ' goods\_u.n it' varcliar<50> NOT NULL, ',gioodls\_ price integer NOT NULL, \* 題oo戒礬爛” uarchar(100> NOT NULL, "" goods\_desc yard-oar (2000> NOT NULL, 'goods\_cag\_idinteger NOT NULL)；

ft LT ER TABLE goods \_^oods in£ o fiDl CONSTRAINT goods ^oods inf o\_\_soods\_ca9\_id\_22dl®7 085\_f k.\_goods\_goodscategory„id' FOREIGN KEV < 'ffoods„caff„id> HEFERENCES ^goods^g-o odscategoryid ； :

COMMIT;

图2512转换为SQL语句

从图2542中可以看到迁移文件对应的SQL语句，这个SQL语句可以直接对数据库进行

操作。

25q4d5 ‘

迁移文件创建成功之后，下一步就是执行迁移，即执行对应的SQL语句，在数据库中创 建对应的数据库表。同样在终端中进入项目目录，然后使用如下命令执行迁移：

python manage.py migrate

该命令的执行效果如图2543所示。

D s XP^tlionPro ject \ttsx>python manage .py migrate Operations to performs

S^nchroniHe immigrated apps s messages,, stat ic files

fippl^ all migrations s admin, contentt^pes, sessions, goods, cart, autli S^nchyoniHing apps without migrations s

Creating tables„„„ Running deferred SQL.»„

Installing- custom £QL»\_ =

Run n in g m igrat io n s s •:

Hendering model states.„« DONE

Applying- contenttypes-0001\_initial„. „ 01

Applying .au.th»0001\_„initial» -. OK

ftppljying fipplyinff fipplying fipplying Applying fippl^ing fipplying Applying

admin.0001„initial..„ OK contenttypes ..BSOSj'emoye^ontent^typejiame „. - OK auth - 0082\_\_a 11 e r\_pe rm is s io n \_n ame \_jnax\_\_le n gt h - -. OK anth. 8003\_aItey„user\_\_email\_jnax„\_length.«. „ 0M aut h. 0004\_a It e r\_us e r„u.s e rn awe \_o pt s -目内 OK ant h „ 8005„a It e f\_lis e r\_las t \_lo g in ,\_ji u 11 -. „ OK auth.0086„requ.ire\_contenttypes\_\_0002 =... OK goods.0001\_initial=.„ OK

cart.0001„initial„.. 0M

Applying- sessions «0001\_init ial- -. 01

图25-13执行迁移

执行迁移之后，我们对数据库进行查看，.会发现对应的数据库表已经创建成功。但是数据 中的表名与已定义的模型类类名并不完全一致，数据库中的表名格式为：应用名—模型类类名 小写，如图25-14所示。

H ttSX

*胃呷聲*Tabies：

* gl 37jth„grGUp
* *fgl*

*靜* §S authjjser

：'广 suth\_\_G5er\_g roups

*点窮襟4ss,臨sj蜀泓血保 电 a* cart^rd^rgc^ds ? ：；r wrt\_cr砖出对 fb [

*!： - ~ -\** - ; 7 , - . 1 7 , - , y «, *h*

►懲里 ^jang&\_sdmircfog

*勘* |S dj a n g 0\_cg nt entity pe

><85 n g grsfj crss

*，* dja^go\_s£^iGn acd-\_grc Js^&gch

Info j Cohms ：• indexes j Trigger：? J Foreign keys.j Rsrti&ns j Gr^nfe J- DOL

|  |  |  |
| --- | --- | --- |
| nn | T>pe | Defeult Vshe Wdtebfe Character Set *g池n* |
| id | int(ll) | *HQ* |
|  | *蚀g* |  |
| gacdsjnfojd | intfll) | *冲* |
| gsDds^orderjd |  | ND |

图25J4创建的数据库表

在图25-14中，与模型类对应的4张表名字分别*为*:cart\_ordergoods、cart\_orderinfo goods\_\_goodscategory和goods\_goodsinfo。除此之外，还可以查看创建的表的结构，包括自动生 成列，外键列等。

2鼠冬（3

数据库创建完成之后，就可以对该数据库进行操作了。在Django框架中，不需要编写一 行SQL语句，使用完全基于面向对象的方式，就可以对数据库中的数据进行操作。为了演示如 何在Django框架中操作数据库，可以在Django shell （ Django框架提供的shell）中执行代码。

打开终端,进入项目所在的文件夹,然后执行以下命令打开Django shello



python manage.py shell

执行后命令行显示信息如下所示。

D:\PythonProject\ttsx>python manage »py shell

Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSG v.1900 32 bit (Intel)]

on Win32

Type "helpn, "copyright"credits" or "license" for more information.

(InteractiveConsole)

»>

此时就进入了 Django shell,我们可在shell中通过面向对象方式对数据库中的数据进行 CRUD操作。

1JSAW

在Django中，要创建一条数据，只需要创建一个对应模型的实例对象，然后调用该对象 的save方法，即可向数据库中插入一条数据。接下来在shell中执行如下代码，在数据库的商 品分类表中插入数据。

>>> from goods.models import \*

»> categories =[(，时令水果 f, 1 fruit') z (f 海鲜水产 1, ! seafoodf) f (1 全品肉 类 1z 'meet \*

)f (1 美味蛋品 \* r \* egg1) r (7 新鲜蔬菜 1, s vegetables ?),(，低温奶制品'p f ice \*)]

>>> for index,cag in zip(range(1A 7)r categories):

*…。* c = GoodsCategory ()

。… c. cag\_name = cag [ 0 ]

c . cag\_css = cag[l]

。… c . cag\_\_img = s images/bannerO%d. jpg f % index

c » save ()

上述代码中，通过fo『循环向数据库中插入了我们项目需要的商品分类数据。

查看数据库中执行结果，如图25T5所示。

|  |  |  |  |
| --- | --- | --- | --- |
| id |  |  |  |
| *► %* | “”1 玲.yi | fruit | fmages/bannerO Ljpg |
| *\ 2* | 海鲜水产 |  | 「nil国，虹 |
| 3 | 全品肉类 |  | images/bannerOS s jpg |
| 4 | 美味畫品 |  |  |
| 5 | 新鲜蔬菜 | vegetables | imagesBanf^erO 5 Jpg |
| 6 | ■ft温奶制品 |  |  |

图2545商品分类表中的数据

接下来再通过同样的方式向数据库中插入一条商品数据，代码如下所示。要注意的是，对 应商品的图片数据放置在static/goods目录下，商品的goods\_img字段存储的是图片名。

>>> from goods.models import \*

»> #创建商品对象

>>> goods = Goodslnfo()

»> #设置属性值

»> goods . goods\_name = s新疆库尔勒香梨5斤1

>>> goods » goods\_price = 7 9

>>> goods.goods\_img = \* xiangli \*

>>> goods . goods\_desc = ?库尔勒香梨果面光滑,香气浓郁,酥脆爽口 ,

>>> goods.goods\_cag\_id = 1

>>> goods.save()

商品数据添加完成后，可以在数据库中查看到执行的结果，如图2546所示。

*» id* jiams goods juries gGcdsJmg -gDGdsjiesc -goGd5\_cag Jd

92E 新鑿屋尔勒香梨5斤79 .:E 摩尔勒香梨果面光潛，1

『 香气衆郁，醺脆熒口

图25-16商品表数据

这里需要额外说明一点，"goods\_cag = models.ForeignKey(? GoodsCategory')” 实际在数据 库中存储的是商品分•类数据的主键。以下代码在添加商品信息时，使用了商品分类对象来表示 商品的分类9与使用goods.goods\_cag\_id设置商品分类是一样的效果。

#创建商品分类

cag = GoodsCategory()

*cag*.save()

goods = Goodslnfo()

goods . goods\_\_cag = cag

goods.save()

2 WWB作

查询操作是项目中使用最多的数据库操作，本项目主要用到了 Django框架提供的以下3 种方法：

° get方法：查询一条数据。

-疝方法：获得所有数据。

° filter方法？根据条件过滤数据。

接下来就对这三个方法的使用进行一一介绍。为了增加测试数据量，可将项目随附的疝. datalxt中包含的批量SQL语句粘贴到数据库中执行，就可以为商品信息表中增加几百条测试 数据。

(1 ) get方法

如果已知只有一个对象满足查询条件，可以使用啓t方法，它直接返回该对象。例如，查 询商品名称为“新疆库尔勒香梨5 kg”的对象，代码如下：

>>> from goods.models import \*

>» goods = Goodslnfo . objects .get (goods\_name= f 新疆库尔勒香梨 5 kg ")

>» print (goods . goods\_name7 goods . goods\_desc? goods . goods\_price)

新疆库尔勒香梨5 kg库尔窈香梨果面光滑，香气夜郁，酥脆爽口 79 —

objects > Goodslnfo类的对象属性，叫作管理器对象。在该对象中封装了用于数据库操作 的方法,我们介绍的get方法、all方法和filter方法都是objects对象定义的方法。

如果匹配到的对象有多个，则 绑方法会触发MuitipleObjectsReturned异常。如果根据给 出的查询条件匹配不到任何对象，则将触发DoesNotExisf异常。

(2)疝方法

使用管理器对象的浏方法将获取一个表中所有对象，例如以下代码：

all\_cags = GoodsCategory.obj ects.all()

疝方法返回的是包含所有对象的一个查询结果集。在实际应用中，可以对结果集使用切片 语法，例如，获得结果集的前两个对象组成的结果集，可使用以下代码：

cags = GoodsCategory,objects.all()[1: 3]

(3 ) filter 方法

虽然使用浏方法能够获得一张数据库表中所有数据的集合，但在通常情况下罗我们往往 只需要获取满足特定条件的部分数据。这时，就可以使用句馅T方法来查询。

例如，查询分类ID为1的所有商品。执行效果如下所示：

>>> from goods.models import \*

»> #查询ID 1的分类

>>> cag = GoodsCategory.obj ects.get(id=l)

>>> goods\_list = GoodsInfo » obj ects filter(goods\_cag=cag)

»> goods\_list. count ()

70

3.更新操作

Django创建和更新对象使用了同一个方法save()。当调用save()方法保存数据时，Django 会判断当前对象是否有主键，如果主键存在则执行更新操作，如果主键不存在则执行插入数据 的操作。例如以下代码，对数据库中的商品“库尔勒香梨”的价格进行了更改：

>>> goods = Goodslnfo « obj ects . get (goods\_name=5 新疆库尔勒香梨 5 kg?)

>>> goods »goods\_price = 100

>>> goods.save()

&删除操作

删除操作使用了管理器对象的delete方法。例如，以下代码删除了名为“新疆库尔勒香梨5蛭”

的商品。

>>> goods = Goodslnfo . obj ects . get (goods\_name='新疆库尔勒香梨 5 kgf )

»> goods o goods\_price = 100

»> goods . save ()

>>> goods o delete()

>>> goods = Goodslnf o o obj ects . get (goods\_name= ,新疆库尔勒香梨 5 kg')

Traceback (most recent call last):

File "<console>"r line 1in <module>,

File "C:\Users\admin\AppData\Local\Programs\Python\Python3 6-32\lib\

site-packag

es\django\db\models\manageropy°r line 127z in manager\_method return getattr(self.get\_queryset(), name)(\*args, \*\*kwargs)

File °C:\Users\admin\AppData\Local\Programs\Python\Python3 6-32\lib\ site-packag

es\dj ango\db\models\query.py"7 line 334 z in get

self omodel»\_meta.obj ect\_name

goods.models.Do㊀sNotExist: Goodslnfo matching query does not exist.

»>

在上述代码中，使用了 <H乖方法将商品删除。删除之后再次尝试查询，1J报出DoesNotExist 异常。

2乱 5

当我们在浏览器输入不同的网址，对应着浏览器发出的不同的请求。对于不同的请求，服 务器需要使用对应的函数来处'理该请求，这就是请求处理函数。我们定义在应用的views.py模 块中，每一个处理请求的函数，我们叫作视图函数。

对于Django的设计框架MVT,用户在URL中请求的是视图,视图接收请求后进行处理, 并将处理的最终数据结果返回给用户浏览器，并解析呈现给用户。也就是说，视图用来封装负 •责处理用户请求和返回响应的逻辑。

使用视图需要进行两步操作：定义视图和配置URLconfo接下来，就按照这两个步骤，创 建一个简单的视图。

25.5J

在Django中,视图一般表现为定义在views.py中的一个Python函数。视图接收一个或多 个参数，并且必须有返回值。视图的参数是一个HttpRequest类型的对象request,包含了所有 请求的信息。视图的返回值是一个HttpResponse对象，包含了返回给请求者的响应信息。

接下来,我们在goods应用的views.py模块中*，*创建一个视图函数index,代码如下: from djangoshortcuts import renderf rom dj ango.http import HttpResponse def index(request)

return HttpResponse(!hello dj angoI 1)

上述视图函数的处理方式非常简单，只是向浏览器返回"hellodjango"字符串。

'2§n5n2

Django使用UJRLconf模块作为网址的解析并且对应到views.py中函数的主要处理者。请 求者在浏览器输入URL请求网站资源，Django获取到URL信息，然后与配置好的URLconf 逐条匹配视图，如果匹配成功，则调用对应视图，如果所有的URLconf没有匹配成功，则返回 404 (Page Not Found)错误。

一条URLconf包括URL规则和视图两部分,其中URL规则使用正则表达式定义,而视图 就是在views.py中定义的函数。

上一小节中已经创建了 index函数，为了让浏览器能够看到"hello django"字符串，必须 配置index函数和某个请求网址对应。

接下来，打开ttsx目录下的urls.py模块中，在该模块中添加代码。内容如下列代码中黑体 字所示。

from dj ango » conf o urls import include^ url from dj ango.contrib import admin

from goods views import index # 导入视图函数 urlpatterns =[

url(r \* 八admin/f*《*include(admin。site o urls))’ url (z，! Aindex/$ \* index)

]

添加的配置代码为：

url(r 5 Aindex/$ \*,index)

第一部分(r 1 Aindex/$1 )为正则表达式,第二部分(index)表示与前述正则表达式匹配 上的请求将调用的视图函数。那么Django具体是如何匹配网址的呢？举例来说，如果请求的 网址是localhost:8000/index/,那么Django首先会去除域名部分(localhost:8000/),然后用余 下的部分(index/)在ttsx/urls.py中的urlpatterns中逐个判断，如果哪条正则表达式与请求的网 址匹配成功，则调用对应的视图函数；如果都不匹配则报错。

配置成功之后，我们启动开发服务器须并在浏览器中进行访问。在浏览器输入网址：

[http://localhost:8000/index/,可以看到浏览器的显示效果,如图2547所示。](http://localhost:8000/index/,%e5%8f%af%e4%bb%a5%e7%9c%8b%e5%88%b0%e6%b5%8f%e8%a7%88%e5%99%a8%e7%9a%84%e6%98%be%e7%a4%ba%e6%95%88%e6%9e%9c,%e5%a6%82%e5%9b%be2547%e6%89%80%e7%a4%ba%e3%80%82)

'localhost

hello djatigc^

图2547浏览器显示效果

2SO6模板渲染

前面小节中我们已经能在浏览器中访问视图内容了，但是如果想让页面显示的样式更漂亮 一些，该如果实现呢？我们可以对视图函数index进行修改，修改后的代码如下所示。

from dj ango.shortcuts import render

from dj ango »http import HttpResponse def index(request):

return HttpResponse(!<hl>hello dj ango!</hl>1)

再次通过浏览器访问localhost:8000/index/?显示效果如图2548所示。

Iccalhost

图25-18美化后的HTML页面

也就是说，如果想让视图返回漂亮的页面，就需要返回包含了 html和css样式的代码。但 是，在视图中编写网页的设计编码会导致一些问题的出现膏例如：

•任何页面的改动会牵扯Python代码的改动，网站的设计改动会比Python代码改动更频繁。 ®后台Pythori代码与设计HTML是不同的工作,页面设计人员不应该编辑Python代码, 应该只与HTML打交道。

•程序员编写后台程序与页面设计人员编写HTML模板同时进行会更高效，而不是一个 人等待另一个人编辑同样的文件。

因此，Django使用了模板系统来负责网页的设计，视图中只需要使用Pythoii编写业务逻辑， 从而将网页设计和Python代码分离开来。模板是一种很便利的方法,可以动态地生成HTMLO 模板包含了所需HTML输出的静态部分，以及一些如何添加动态内容的特殊语法描述。

25a@J飜置"

如何让视图找到模板文件？我们需要配置模板的路径，以便Djang。模板引擎能够找到并 加载模板文件。

为此，编辑ttsx/settings.py文件，修改模板配置如下：

TEMPLATES =

{

'BACKEND! : !dj ango o template.backends.dj ango。Dj angoTemplates'$ 'DIRS 1: [osopathojoin(BASE\_DIR,\* templates!)]f

,APP\_DIRS1: True, OPTIONS 1 : {

?context\_processorsv:[

'dj ango.template。context\_processors.debug'z

? dj ango.template。context\_processors\request? f

, dj ango。contrib.auth.context\_processors.auth \* r

!dj ango.contrib ^messages.context\_processors.messages'/ L

}’ h

]

模板配置的参数含义如表25；所示。

|  |  |
| --- | --- |
| a 25-3模板配置的参数含义 | |
| 项 目 | 解 释 |
| backend | 使用何种模板引擎 |
| dirs | 定义了一个目录列表，模板引擎按照列表顺序在这些目录中查找模板文件 |
| app\_dirs | 是否进入已安装应用中查找模板。每种模板引擎都定义了一个默认的名称作为搜索模板的子目录。  例如,Django为它自己的模板引擎指定的是＞ templates 1 |

上述添加的代码 TORS5: [os.pathjoin(BASE\_DIR, ?templates?)],表示模板存放在项目根目 录下的templates目录,BASE\_DIR就是由Django定义的项目根目录路径。

2亂&2瘻魇樓靈 .

接下来，我们在项目根目录下创建templates目录，并在该目录中创建模板文件index.

html,该文件内容如下:

<!DOCTYPE html>

<html lang=nenn>

<head>

<meta charset="UTF-8"> <title>Title</title>

</head>

<body>

<hl>hello django!</hl>

</body>

</html>

修改index视图函数，使用快捷函数render返回一个包含了 HTML模板渲染后内容的 HTTPResponse对象,如下所示。

from django.shortcuts import render

from dj ango.http import HttpResponse

def index(request):

return render(request, 'index.html\*)

再次访问[http://localhost:8000/index/网站,页面如图25-19所示。](http://localhost:8000/index/%e7%bd%91%e7%ab%99,%e9%a1%b5%e9%9d%a2%e5%a6%82%e5%9b%be25-19%e6%89%80%e7%a4%ba%e3%80%82)

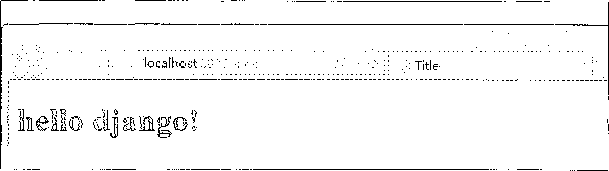


图25J9使用模板后显示的页面

25d®h3瓣樓糠憧儷

我们还可以通过血如 函数给indexJitml传递值，并将这些值显示在模板中。例如，修改 index视图函数如下：

from dj ango o shortcuts import render

from django.http import HttpResponse

def index(request):

return render (requestz ' index . html1 , { \* name 1 : \* 张三 \* r 1 age 1 : 20 })

上述代码在index视图中给index.html模板传递了一个参数。给模板传递的参数必须是字 典类型。

那么如何在模板中显示这些数据呢？接下来，修改文件，具体如下言

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8°>

<title>Title</title>

</head>

<body>

我的名字是:{ ( name } } ’年龄是:{ ( age } }.

</body>

</html>

在浏览器输入网址localhost:8000/index/访问网站,显示效果如图25-20所示。

羅IliiS德縱II!亨券 Title »■

:我的名字是:张三，年龄是：2"

图25-20传递参数后的显示效果

我们把数据填充到模板中的这一过程，叫作模板渲染。在模板中，使用特殊的语法“{{字 典键名}) ”来显示该键对应的值数据。

2Sn®n4静窓支繹

除了由服务器生成的HTML文件外，网页应用通常还需要提供其他必要的文件歹如图片 文件、JavaScript脚本和CSS样式表，来为用户呈现出一个完整的网站。Django中将这些文件 统称为“静态文件”。

通常，当用户请求一个URL的时候，Django需要将其映射到对应的视图中进行处理。但 是当用户发送对静态文件的请求时，显然并不需要对应的视图处理，只需要将该静态文件内容 发送到浏览器端即可。

因此*，*对静态文件应该单独进行配置。在ttsx/settings.py文件中对静态文件的访问进行配置, 代码如下：

STATIC\_URL = J/static/J

STATICFILES\_DIRS = [os .path j oin (BASE\_DIR:, ! static9 )] 上述代码中，第一行是Django默认添加的，第二行是我们添加的配置内容。当发现请求 的网址路径是以/static/开头时，程序就会从STATICFILES\_\_DIRS配置的路径下去搜索静态文件。

以图片文件为例,我们在项目的根目录下创建static文件夹,在该文件夹下放入一张名为 defaultjpg的图片。然后修改index.html文件，代码如下:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<img src="/static/default o jpg”〉

</body> .

</html>

在浏览器使用网址[http://localhost:8000/index/访问网站，显示效果如图25\*21所示。](http://localhost:8000/index/%e8%ae%bf%e9%97%ae%e7%bd%91%e7%ab%99%ef%bc%8c%e6%98%be%e7%a4%ba%e6%95%88%e6%9e%9c%e5%a6%82%e5%9b%be25*21%e6%89%80%e7%a4%ba%e3%80%82)

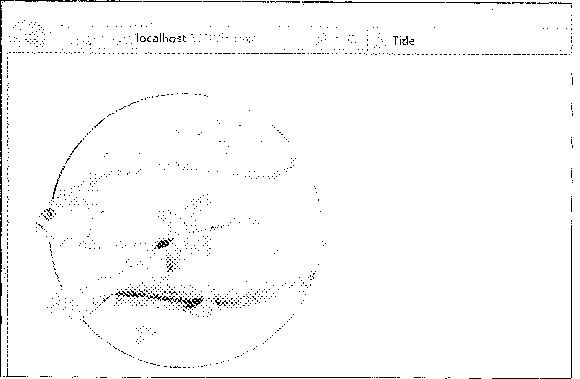


图25-21网页上显示图片

从图25・21可知，图片文件已经访问成功。除了图片文件，其他静态文件都可通过上述方 式进行配置和访问。

7 Coofcie

Cookie指某些网站为了辨别用户身份、在用户本地终端上存储的数据（通常经过加密）。

Cookie是由网景公司的前雇员Lou Montulli在1993年3月发明的。Cooki©内包含了若干 键值对，由服务器端定义和生成，发送给User^Agent （一般是浏览器）。浏览器会将Cookie的 键值对保存到本地，下次请求同一网站时就发送该网站的Cookie数据给服务器（前提是浏览器 设置为启用cookie ） o

Cookies最典型的应用是判定注册用户是否已经登录网站。另外，浏览器可能会提示用户， 是否在本地保留用户信息以便在下一次进入此网站时简化登录手续，这些都是通过Cookies实 现的。

另一个重要应用场合是“购物车”之类的处理。用户可能会在一段时间内在同一家网站的 不同页面中选择不同的商品，这些信息都会写入Cookie,以便在最后付款时提取信息。

要注意的是，Cookie是存储在浏览器中的一段纯文本信息，一般不要存储敏感信息如密码， 因为计算机上的浏览器可能被其他人使用。

（1 ）设置 Cookie

在服务器端设置Cookie使用的方法如下言

HttpResponse . set\_\_cookie （key,

value== \* ? ’

max\_age=None,

expires=Nonef

path= f/\* 7

domain=Nonef secure=Nonef

644 .紐户Pjthon褻臓飜麗：關霉雾Pyfthoo

httponly=False)

该方法的参数介绍如下：

® key： Cookie存储的键。

* value: Cookie 存储的值。
* max age:表示Cookie的有效时间,以s (秒)为单位。

。expires：表示 Cookie 的过期时间，应该是一个 UTC “Wdy, DD^Mon^YY HH:MM:SS GMT”格式的字符串,或者一个datetime.datetime对象。

* domain：用于设置一个跨域的Cookieo例如,domain=f,.lawrence.com"将设置一个 [www.lawrence.comblogs.lawrence.com](http://www.lawrence.comblogs.lawrence.com) 和 calendars.lawrence.com 都可读的 Cookieo 默 认情况下，Cookie只能被设置它的域读取。

® httponly：表示是否只能由HTTP代码访问Cookieo默认值为FMse,如果要阻止客户端 的 JavaScript 访问 Cookie 9 可以设置 httponly=Tme0

(2 )删除 Cookie

删除Cookie可以使用HttpResponse的delete\_cookie方法,代码如下所示。

HttpResponse . delete\_cookie (key, path= \* / \* , d.omain=None)

(3 )读取 Cookie

HttpRequest对象的COOKIES属性包含了该请求的所有Cookie,它是一个字典类型,其键 和值都为字符串类型。

一般通过Cookie的键名来读取Cookie,格式如下所示。

request^COOKIES[key]

#或

request.COOKIES.get(key,\*')

2鼠g首黄礙爾助離实瑰

接下来，使用前面介绍的创建视图和模板渲染知识来实现天天生鲜项目的各个网页。 首先是首页商品展示页面，该页面主要展示的数据包括：

1. 产品的分类数据。
2. 每个分类的最新产品数据。
3. 购物车商品数据。

在实现页面之前，先将随附的静态文件拷贝到项目的static目录下。

实现页面功能的步骤大致可分为以下三步：(1)仓0建视图函数；(2)仓U建模型文件；(3)配 置视图函数与访问路径的对应信息。接下来，依次实现上三个步骤。

25.8J OSSMaB

在goods目录下的views.py文件里添加首页视图函数index*，*代码如下:

1. def index(request):
2. ° ° "首页页面"° "
3. #获取所有的商品分类
4. categories = GoodsCategory.obj ects « all ()
5. #获取每个分类中的4个商品
6. for cag in categories:
7. cag o goods\_list = cag.goodsinfo\_set e order\_by(v-id?) [:4]
8. #读取购物车商畐列表 — ~
9. cart\_goods\_list =[]
10. #商品总数
11. cart\_goods\_count = 0
12. for goods\_iclf goods\_num in request» COOKIES . items ():
13. #我们祐商品工D幕为数字，非数字的cookie过滤掉
14. if not goods\_id.isdigit():
15. continue
16. cart\_goods = Goodsinfo.objects.get(id=goods\_id)
17. cart\_goods.goods\_num = goods\_num
18. cart\_goods\_list <, append (cart\_goods)
19. #累嬴购物车商品总数
20. cart\_goods\_count = cart\_goods\_count + int(goods\_num)
21. return render(requestz f index.html1f
22. (5 categories!: categories f
23. 5 cart\_goods\_list \*: cart\_goods\_listr
24. ' cart\_goods\_count \* : cart\_\_goods\_count})

上述代码的第9〜20行是购物车数据展示功能。由于购物车数据存储在Cookie中，所以遍 历requestXJOOKIES从中读取所有商品信息。其中，第14〜15行代码用于从Cookie中过滤非商 品数据，这是由于Cookie中除了商品数据之外，Django还会添加其他的信息，需要通过if判 断将非商品数据的Cookie过滤掉。

25.8.2劍還滇海安俸 ’

视图定义之后，需要编写首页对应的模板文件indexohtml。indexhtml模板的篇幅较长，我 们对该模板的完整内容进入分段介绍，以期将整个页面的完整内容进行全面介绍。在介绍后续 页面时，只需要介绍关键信息即可。

首先是HTML文件的头部信息，包含了该页面的标题、链接库和css链接文件，代码如下 所示。

<!DOCTYPE html PUBLIC °-//W3C//DTD XHTML 1.0 Transitional//EN°

"[http://www.w3.org/TR/xhtmll/DTD/xhtmll-transitional.dtd](http://www.w3.org/TR/xhtmll/DTD/xhtmll-transitional.dtdn)[n](http://www.w3.org/TR/xhtmll/DTD/xhtmll-transitional.dtdn)> <html xmlns=nhttp://www.w3.org/1999/xhtml° xml:lang=°enn> <head>

<meta http-equiv=nContent-Type" content^"text/html； charset=UTF-8"> <title>天天生鲜-首页</title>

<link rel="stylesheet" type=ntext/css" href/static/css/reset« cssf?> <link rel="stylesheet" type=°text/css" href="/static/css/main.cssn> <script type="text/javascript"

src="/static/j s/j query-1.12.4.min。j s"></script>

<script type="text/javascript" / static/j s/j query-ui .min . j s °></script>

<script type="text/j avascript"

src="/static/js/slide.js°></script>

</head>

接下来是HTML文件体。HTML文件体包含了页面的主要数据,包含了产品的分类数据、

每个分类的最新产品数据、以及购物车商品数据。

下列代码包含了购物车商品数据,如下所示。

<body>

<div class="search\_\_bar clearfix">

<a href="/index/" class="logo fl"ximg src=" / static/images/logo。pngnx/a>

<div class="guest\_cart fr°>

<a href=n/cart/show\_cart/n class="cart\_name fl°> 我的购物车 </a> <div class="goods\_count fln>{( cart\_goods\_count }}</div> <ul class=°cart\_goods\_showM>

{% for cart\_goods in cart\_goods\_list %}

<li> — ~

<img src=n/static/goods：/ ( { cart\_goods . goods\_img } } . jpgn alt=n商品图片叩〉 一 ~

<h4> { { cart\_\_goods . goods\_name } } </h4> <div> { { cart\_\_goods e goods\_\_num } ) </div>

</li> 一 ~~

{% endfor %}

</ul>

</div>

</div>

上述代码包含了由index视图函数传递给index.html模板的数据。其中9 cart\_goods\_list是 一个集合，包含了用户添加到购物车中的商品。在模板中使用Django提供的模板语法来遍历 和输出了该集合的商品信息。

遍历集合采用了模板的for语法，其使用格式如下所示。

{% for。…in%}

{% endfor %}

输出变量值，使用的是如下格式的语法：

( {变量名}}

接下来是输出产品分类信息的HTML代码，如下所示。

<div class="center\_con clearfix">

<ul class=nsubnav fl">

(% for cag in categories %)

<li><a href="/goods/?cag={{ cag.id })&page=l° class="{{ cag.cag\_css }}">{{ cag.cag\_name )) </a>

</li>

{% endfor %)

</ul>

<div class=nslide fl°>

<ul class=°slide\_pics"> <liximg src="/static/images/slide » jpg" alt="幻灯片"></li> <li><img src="/ static/images/s 1 ide02 o jpg" alt=° 幻灯片 °></li>

<liximg src="/static/images/slide 03 . jpg" alt=° 幻灯片 °></li> <liximg src=°/static/images/slide04 . jpg" alt="幺丁灯片"></li> </ul>

<div class=nprev°></div>

<div class=°nextn></div>

<ul class="points"></ul>

</div>

<div class=nadv fl">

<a href=n#nXimg src=n/static/images/advOl. jpg°x/a> <a href=°#"><img src=n/static/images/advO2.jpg"></a> </div>

</div>

上述代码同样使用了 fcff循环遍历贓坨goiies中包含的数据。 每个分类的最新商品信息，代码如下所示。

{% for cag in categories %} <div class="list\_model">

<div class="list\_title clearfix">

<h3 class="fr? id=nmodel01°>{{ cag.cag\_name *}}</h3>* <a href="/goods/?cag={{ cag» id }}&page=l° class=°goods\_\_more f 查看更多 >

</a>

</div>

<div class-"goods\_con clearfix°>

<div class=ngoods\_banner fl">

<img src=°/static/({ cag.cag\_img }}"> </div>

<ul class="goods\_list fl"〉

(% for goods in cag.goods\_list %}

<li>

<h4>

< a href="/detail/?id={{ goods.id }}"〉{{ goods»goods\_name }}</&〉 </h4>

<a href="/detail/?id= { { goods o id. }}">

<img src="/static/goods/{{ goods.goods\_img })o jpg°> </a>

<div class=npriz:en>¥ {{ goods . goods\_price } }</div> </li> {% endfor %}

</ul>

</div>

</div>

{% endfor %}

上述代码遍历输出每个分类下的商品数据。当点击某个商品的图片或名称时，页面会跳转 到该商品的详细页面，详细页面的链接地址如下所示。

/detail/?id={{ goods.id }}

其中，问号之后的内容为请求携带的参数，参数的键为id,值为商品ID。

HTML内容的最后是页尾部分，代码如下：

648 w. Python觐濺爨曜旨映霸寥Python

<div class="footer°>

<div class="foot\_link">

<a href="#"> 关于我们 </a>

<span>I</span>

<a href="#"> 联系我们 </a>

<span>I</span>

<a href="#°> 招聘人才 </a>

<span>I</span>

<a href="#°〉友情链接 </a>

</div>

<p>CopyRight © 2016北京天天生鲜信息技术有限公司All Rights Reserved</p> <p〉电话：010-\*\*\*\*888 京工CP 备 号 </p〉

</div>

</body>

</html>

25q®q3 •

之前已经在ttsx/urls.py文件下配置了 index访问路径,如下所示。

url(r!Aindex/$'rindex)

现在可以直接运行开发服务器，并通过路径localhost:8000/index/访问天天生鲜网站的主页 内容了。

:2扇。9廠齬诽细施西碰餞实现

25.9J創篷觀圖幽飆

在goods应用的views.py模块中新增视图函数detail,用于实现商品详细信息显示功能。

代码如下:

def detail(request):

° n "商品详细页面"° "

#获得产品ID

goods\_id = request.GET.get ( J id \*,1)

#查询袞商品

goods\_data = Goodslnfo . obj ects o get (id=goods\_id)

#查询商品分类 —

categories = GoodsCategory o obj ects.all()

#读取购物车商品列表

cart\_goods\_list =[]

#商冨总数一

cart\_goods\_count = 0

for goods\_idz goods\_num in request.COOKIES s items():

#我们糸商品工D赢为数字，非数字的cookie过滤掉

if not goods\_id.isdigit():

continue

cart\_goods = Goodslnfo o obj ects . get (id=goods\_id.)

cart\_goods.goods num = goods num

cart\_goods\_\_list • append (cart\_goods)

#累无购物车商品总数 —

cart\_goods\_count = cart\_goods\_count + int(goods num) '

return render(request, fdetail.htmlf r{r categories 1: categories^ \*goods\_data\*: goods\_dataz

'cart\_goods\_list5: cart\_goods\_listf

s cart\_goods\_count \*: cart\_goods\_count})

以请求网址localliost:8000/detail/?id-100为例，Django会自动将问号(?)后面的参数进行解 析，并存储在request.GET字典中。detail函数负责获取商品的ID,根据ID到数据库中查询商 品的详细数据，并传入模板中显示。

2鼠新2劍蹩縷撮黨繹 ’

商品详细页面的模板文件名称为detaiLhtml,这里，我们在标题行中使用商品名称作为网 页的标题，代码如下。

<title>{{ goods\_data. goods\_ name }} -■商品详情 </title>

展示商品详细信息页面的代码如下所示。

<div class="breadcrumb°>

<a href="#n> 全部分类 </a>

<span>x/span>

<a href=n#°>{{ goods\_data« goods\_cags cag\_name }}</&>

<span>x/span>

<a. href=W〉商品详情 </3〉

</div>

<div class=ngoods\_detail\_con clearfix°>

<div class=°goods\_detail\_\_pic fV'ximg src=°/static/goods/({ goods\_data。goods\_img }}.jpgH></div>

<div class=°goods\_detail\_\_list fr">

<h3>{{ goods\_data o goods\_name }}</h3>

<br>

<div class="prize\_bar">

<span class=nshow\_pirzeH>¥<em>

({ goods\_data„goods\_price }}</em></span>

</div>

<div class="goods\_num clearfix°>

<div class="num\_name fl"〉数量:</div>

<div c 1 ass="num\_add fl">

<input type=ntext° class=nnum\_show fl° value=nln disabled> </div>

</div>

<div class=noperate \_btn">

<a href="/cart/add\_cart/?id={{ goods\_data.id }}° class=°add\_cartn id= °add\_cart"> 加入购物车

</a>

</div>

</div>

</div>

<div class="main\_wrap clearfix">

<ul class=ndetail\_tab clearfix">

<li class=nactive"> 商品介绍 </li>

</ul>

<div class="tab\_\_contentn>

<dl>

<dd>{{ goods\_data » goods\_name }}</dd>

</dl>

</div>

</div>

25b9b3

在ttsx目录下的urls.py模块中，配置该视图函数和网页访问路径之间的对应关系。

from django.conf.urls import include,url

from dj ango.contrib import admin

from goods . views import index # 导入视图函数

from goods.views import detail

urlpatterns =[

url(r'Aadmin/9, include(admin.site.urls))r

url(r \* Aindex/$ ? 7 index)z

url(r \* Adetail/$ 5 ^detail)*号*

2瓦凱凰曩嬲

商品详细页面还提供了商品购买功能。当用户单击“加入购物车”按钮，会将产品信息力口 入购物车。为此,在模板中创建了加入购物车的链接，代码如下:

<a href=n/cart/add\_cart/ ?id= { { good.s\_data « id } } ° class = °add\_cart" id= °add\_cart"> 加入购物车 </a>

为T处理加入购物车的请求,在cart应用下的views.py模块中,新增add cart视图函数, 代码实现如下：

def add\_cart(request):

叫5添加商品到购物车心顷

#获得商品ID

goods\_id = request.GET.get(? id\ !') if goods\_\_id:

#获得上一页面地址

prev\_url = request»META[v HTTP\_REFERERf]

#写入到cookie中 response = redirect(prev\_url)

#判断商品是否存在 —

goods\_count = request.COOKIES.get(goods\_idz 5 !) if goods\_count:

goods\_count = int(goods\_count) + 1

else :

goods\_count = 1 response.set\_cookie(goods\_idz goods\_\_count) return response

在上述代码中9首先获得了要加入购物车的商品ID,因为购物车是通过cookie来实现的。 代码如下：

goods\_id = request,GET.get(!id1, J 1 )

然后获得上一页面的地址，在request对象的META字典中保存用户访问的上一页面地址, 代码如下：

prev\_url = request.META[\*HTTP\_REFERER\*]

当处理完购物车添加之后，我们希望页面跳回到上一页面。所以调用快捷函数redirect,创 建另一个重定向响应，该函数接收一个参数，表示要跳转的页面地址，返回的响应对象类型为 HttpResponseRedirect,它是 HttpResponse 类型的子类。代码如下:

response = redirect(prev\_url)

然后判断当前要添加的商品ID是否已经存在购物车中，如果存在则更新商品数量，如果 不存在则新增一条cookie信息，代码如下：

goods\_\_count = request o COOKIES e get (goods\_id7 ? ,)

if goods\_count:

goods\_count = int (goods\_\_count) + 1

else :

goods\_count = 1

最终将商品添加到cookie中，代码如下：

response.set\_cookie(goods\_id7 goods\_count)

当返回该重定向响应之后，浏览器会重新请求上一页面地址，完成页面跳转。

return response

接下来在ttsx/urls.py模块中配置该视图函数和网址之间的对应关系，配置内容如下:

urlpatt㊀rns =[

url(r'Aadmin/\* finclude(admin.site.urls))z

url(r'Aindex/$ \*,index),

url(r'Adetail/$1,detail)r ,

url(r'Acart/add\_cart/$ s r add^cart)e

]

25010商齬豹獎黄IS碰離实嬲

当点击首页分类链接、或者商品详细页面的分类链接，会进入商品分类页面，显示当前分 类下的所有商品。接下来,就来实现商品分类页面的功能。

25J0J ailSBHB ' '

在goods应用的views.py模块中添加goods视图展示分类商品数据。该视图函数需要获取 2个关键数据：当前分类的ID,以及显示数据的页码。

goods视图函数的实现代码如下:

def goods(request):

° ° "商品展示页面° ° °

#获得当前分类

cag\_id = request.GET.get(\*cag'f1)

#获得当前页码

page\_id = request.GET.get(1page \* 71)

#查镉所有数据

goods\_data = Goodslnfo.objects.filter(goods\_cag\_id=cag\_id)

#数据务页 ~

paginator = Paginator(goods\_dataf12)

#获得当前页码数据 ~

page\_data = paginator.page(page\_id)

#查葡商品分类

categories = GoodsCategory.objects.all()

#查询当前商品分类

current\_\_cag = GoodsCategory.objects.get(id=cag\_id)

#读取购必车商品列表

cart\_goods\_list =[]

#商嘉总数~

cart\_goods\_count = 0

for goods\_idf goods\_num in request.COOKIES.items():

#我们¥勺商品工D新为数字，非数字的cookie过滤掉

if not goods\_ideisdigit(): continue

cart\_goods = Goodslnfo.obj ects.get(id=goods\_id)

cart\_goods »goods\_num = goods\_num cart\_goods\_list.append(cart\_goods) #累亦购物车育品总数 —

cart\_goods\_count = cart\_goods\_count + int(goods\_num) return render(requestf 1 goods.htmlJ f { f page\_data ,: page\_dataf

\* categories \*: categories,

\*current\_cag \*: current\_cagz

\*cart\_goods\_list \*: cart\_goods\_listz

s cart\_goods\_counts: cart\_goods\_count z 'paginator \*: paginator,

v cag\_\_id \* : cag\_id})

在上述代码中，首先获得当前的分类][D和当前的页码数据。例如，请求的网址为：

locahost:8000/goods/?id=2&page=3,则表示要查看分类id为2,页码为3的数据。"&” 用于连接多个请求参数。

然后根据分类DD查询当前分类下所有的数据歹查完数据之后，对商品的数据进行分页显示。 分页显示数据使用了分页类Paginator来实现。创建Paginator对象时需要指定2个参数，第一 个是要分页的结果集，第二个参数是每页显示的数据量。创建完Paginator对象之后，通过调 用page方法即可获得当前页码应显示的数据。

(泠多学T召：分页类的使用

1 □ Paginator 类

Paginator类是Django的内置分页类，其定义如下所示。

class Paginator(object\_list,

per\_pagez orphans=0 7 allow\_empty\_first\_page=True)

它的参数介绍如下：

* objectjist:表示分页的数据集，可以是list, tuple. Django QuerySet类型。
* per\_page:表示每页显示多少条•数据。
* orphans :表示最后一页数据的最小数量，只有最后一页的数据量大于orphans时才会 显示最后一页。
* allow\_empty\_first\_page:表示是否允许空数据，默认值为True。如果该值为False,那么 当数据集为空时会抛出EmptyPage异常。

Paginator类的常用方法主要是page方法，该方法介绍如下:

* Paginatorpage(number):返回在提供的页面下标处的Page对象，下标以1开始。如果 提供的页码不存在,抛出 InvalidPage 异常。PageNotAnlnteger 和 EmptyPage 是 Invalid- Page类的子类。

Paginator类的常用属性如表25-4所示。

|  |  |
| --- | --- |
| 表25-4 Paginator类的常用属性 | |
| 属 性 | 描 述 |
| Paginator.numjpages | 页面的总数 |
| Paginator.page\_range | 页码的范围，从1开始，例如[123,4] |
| Paginator.count | 所有页面的对象总数 |

2. Page^

Page类表示分页数据中的一页,通常不需要手动构建Page对象,可以从Paginator.page方 法来获得。

Page类的常用方法如表25-5所示。

表25・5 Page类的常用方法

|  |  |
| --- | --- |
| 方 法 | 痼.述 |
| Page.has\_next() | 如果有下一页,则返回True |
| Page.has\_\_previous() | 如果有上一页，返回True |
| Page.has\_other\_pages() | 如果有上一页或下一页,返回True |
| Page.next\_page\_number() | 返回下一页的页码。如果下一页不存在,抛出InvalidPage异常 |
| Page.previous\_page\_number() | 返回上一页的页码。如果上一页不存在，抛出InvalidPage异常 |

25J0.2創還滇瀕支排

商品分类页面的模板文件名称为gmxRhtml ,该模板代码如下:

<div class="breadcrumb°>

<a href=n#n> 当前分类:</a>

<span></span>

<a href=°#">{{ current\_cag.cag\_name })</a>

</div>

<div class=°main\_wrap clearfix°>

<div class=° sort\_bar"x/div>

<ul class=="goods\_type\_list clearfix">

{% for goods in page\_data %}

<li>

<a href="/detail/?id={{ goods id }}°>

<img src=°/static7goods/{{ goods,goods\_img }).jpg"> </a>

<div class=°operate">

<h4>

<a href=°/detail/?id={ { goods . id. }}">{{ goods . goods\_name }}</a> </h4>

<span class="prize"> ¥({ goods.goods\_price }}</span> </div> ■

</li>

{% endfor %}

</ul>

<div class="pagenation">

{% if page\_datahas\_previous %}

<a href="?cag={ { cag\_id } }&page={ ( page\_data. s previousjpage\_number }尸> 上一页 </a〉 {% endif %7

{% for index in paginator »page\_range %)

{% if index == page\_data.number %}

index })°

<a href=n?cag={{ cag\_id }}&page={{

class="active">{{ index })

</a>

{% else %)

<a href=°?cag={{ cag\_id </a>

{%

{%

{%

<a

} ) °> 下一页 ></a>

{%

endif %} endfor %} if page\_data□has\_next href=° ?cag={{ cag\_id

endif %}

}}&page={{

%}

}}&page={{

index })">{{ index }}

page\_data.next\_page\_number

</div>

</div>

25J0.3 SiKOBS

在ttsx/urls.py模块中配置该视图函数和网址之间的对应关系,添加的配置如下: url(r ? Agoods/$ ? ’ goods)/

现在就可以单击商品分类链接，访问商品分类页面了。

'2BO11购物菲噸面晒能实嬲 ’’

购物车页面主要包含两个功能：显示购物车商品详细数据，以及商品删除功能。本节将实 现这两个功能。

25JU創還襁圖圍灘

在cart应用的views.py模块中增加show\_cart视图函数，用来显示购物车商品数据。代码 如下言

def show\_cart(request):

U H n展示购物车商品5“

#读取购物车商品列表 cart\_goods\_list = []

#商嘉总数~

c£irt\_goods\_count = 0

#商品总价

cart\_goods\_money = 0

for goods\_idz goods\_num in request.COOKIES.items():

#我们畐商品:ED都品数字，非数字的cookie过滤掉

if not goods\_idoisdigit(): continue

cart\_goods = GoodsInfo.obj sets » get(id=goods\_id)

cart\_goods . goods\_\_num = goods\_num

cart\_goods.total\_money = int(goods\_num) 大

cart\_goods.goods\_price cart\_goods\_list.append(cart\_goods)

#累加购物车商品总数

cart\_goods\_count = cart\_goods\_count + int(goods\_num)

#累讦商品，启介 ~ —

cart\_goods\_money += int(goods\_\_num) 大 cart\_goods.goods\_price return render(request, 1 cart.html1, {

1 cart\_goods\_list? : cart\_\_goods\_list

\* cart\_goods\_count ,: cart\_goods\_count 7

1cart\_goods\_money1: cart\_goods\_money})

25J12劍篷滇淚支歸

购物车页面的模板名称为cart^html,该模板的核心代码如下所示。

<div class="total\_count°> 全咅F商品 <em> { ( cart\_goods\_count } } </em> 件 </div>

<ul cla.ss=ncart\_list\_th clearfixn>

<li class=ncol01°> 商品名称 </li>

<li class=°col03n> W^<# </li>

<li class="col04"> Mi? </li>

<li class="col05"> </li>

<li c1ass="co丄06”> 操作 </li>

</ul>

{% for cart\_goods in cart\_goods\_list %}

<ul class="cart\_list\_td clearfix°>

<li class="col01°></li>

<li class="col02°>

<img src="/static/goods/{ { cart\_goods . goods\_img } } jpg"x/li> <li class=ncol03">{{ cart\_goods.goods\_name }}<br>

<em>{{ cart\_goods.goods\_price }} XL </em></li>

cart\_goods.goods\_price })元 </li> cart\_goods.goods\_num })</li> cart\_goods . total\_\_money } } 元 </li>

<li

<li

<li

<li

class="col05n>{{

class="col06°>{{

class=ncol07°>({

class=°col08">

<a href=" /cart/remove\_cart/?id= { { cart\_goods . id }} ”〉删除 </a></li>

</ul>

{% endfor %}

<ul class="settlements°>

<li class="col01">.</li>

<li class=ncol02°>.</li>

<li class=°col03"> 合计(不含运费):

<span>¥</span><em>( { cart\_goods\_money } }</em><br> 共计

<b>{ { cart\_goods\_count } }</b> 件商品 </li>

<li class=°col04"><a href="/cart/place\_order/°> 去结算 </a></li>

</ul>

25J1d3

在配置文件urls.py中添加购物车访问路径，如下所示。

url(r!Acart/show\_cart/$ \*, show\_cart)卩

此时，单击页面上的购物车图标，就可以查看购物车的详细内容了。

25J1a4

当用户要删除某个购物车商品时，该请求由曙rt应用下的remove\_cart视图来处理。该视 图函数实现如下：

def remove\_cart(request):

° ° "删除购物车商品° °

#获得要删除的商品ID goods\_id = request.GET.get(1 id!z \* \*) if goods\_id:

#获得上一页面地址 prev\_url = request.META[1HT T P\_RE FERER?]

#写入cookie中

response = redirect(prev\_url)

#判断商品是否存在 -

goods\_count = request.COOKIES.get(goods\_id7 '') if goods\_count:

response.delete\_cookie(goods\_id)

return response

上述代码中通过商品的HD,从购物车中删除该商品，删除完成之后仍然跳转到购物车页面。 为此，需要在urls.py中添加一下配置代码：

url(r \* Acart/remove\_cart/$1} remove\_cart),

现在，就可以在购物车页面删除商品了。

2Sol2订犖提衮施函现能实现

当购物车商品数据确认无误之后，单击“结算”按钮跳转到订单提交页面，在该页面用户 需要填写收货地址、联系电话、联系人等信息。

25J2J t!»BB數

在cart应用下的views.py模块中新增视图函数placeholder,代码如下:

def place\_order(request):

° ° "提交订单页面° ° "

#读取购物车商品列表 cart\_goods\_list =[]

#商品总数

cart\_goods\_count = 0

#商品总价

cart\_goods\_money = 0

for goods\_idf goods\_num in request.COOKIES.items():

#我们的商品ID都为数字*己*非数字的cookie过滤掉

if not goods\_id.isdigit():

continue

cart\_goods = Goodslnfo.objects s get(id=goods\_id)

cart\_goods goods\_num = goods\_num

cart\_goods.total\_money = int(goods\_num) \*

cart\_goods•goods\_price cart\_goods\_list.append(cart\_goods)

#累无购物车商品总数 —

cart\_goods\_count = cart\_goods\_count + int(goods\_num)

#累开商品总称 ~ -

cart\_goods\_money += int(goods\_num) \* cart\_goods.goods\_price return render (requestz \*place\_\_order . html' r {

1cart\_goods\_list \*: cart\_goods\_listf

'cart\_goods\_count1: cart\_goods\_count f

'cart\_goods\_money\*: cart\_goods\_money})

25J2n2創還複飆黨俸

place\_ordeEhtml模板的核心代码包括商品列表和填写收货人信息。其中，商品列表的代码 如下：

<h3 class="common\_title"> 商品歹"表 </h3>

<div class=ncommon\_list\_con clearfix°>

<ul class="goods\_list\_th clearfix°>

<li class="col01n> </li>

<li class=°col03"> W^«</li>

<li class="col04°> 数量' </li>

<li class=ncol05n> </li>

</ul>

(% for cart\_goods in cart\_goods\_list %}

<ul class="goods\_list\_tci clearfix">

<li class=ncol01n>l</li>

<li class="col02°>

<img src="/static/goods/ { { cart\_\_goods . goods\_img } } . jpg°x/li>

cart\_goods.goods\_name}}</li> cart\_goods.goods\_price }}元 </li> cart\_goods.goods\_num }}</li> cart\_goods.total\_money }) 元 </li>

<li class=°col03n>{(

<li class="col05H>{{

<li class="col06n>{{

<li class-v,col07n>{ { </ul>

{%

</div>

endfor %)

填写收货人信息和提交订单的代码如下：

<h3 class="common\_title"> 收货地址 </h3>

<form action=°/cart/submit\_order/" method=npost">

{% csrf\_token %)

<div class=°common\_list\_con clearfix">

<dl>

<dd>

<table>

<tr>

<td> 收货地址:</td><td><input style=nwidth: 350px; height: 23px;" type="text° name=Kaddr°></td> </tr>

<tr>

<td> 收货人:</td><td><input style=nwidth : 350px; height: 23px; " type=°text" name=nrecv"x/td> </tr>

<tr>

ctd〉联系电话:</td><tdxinput style=°width : 350px; height: 23px;" type=°text" name=°telen></td> </tr>

<tr>

<td〉备注:</tdxtdxtextarea style=Hwidth: 350px; height: lOOpx; font-size: 12px;° name=°extran> </textarea>

</td>

</tr>

</table>

</dd>

</dl>

</div>

<h3 class="common\_titlen> 支付方式 </h3>

<div class="common\_list\_con clearfix"> <div cla,ss=npay\_style\_con clearfix°>

<input type=nradio" name="pay\_style" checked>

<label class=ncash°> 货到付款 </label>

</div>

</div>

<h3 class="common\_title°> 总金额结算 </h3> <div class=ncommon\_list\_con clearfix,?>

<div class=° settle\_con°>

<div class="total\_goods\_count">

共 <em> ( { cart\_goods\_count } } </em> 件商品,总金额 <b> { { cart\_goods\_money } }元 </bx/div>

<div class=,,transit°> 运费:<b>10 元 </bx/div>

<div class=°totaljpayn> <b>{ { cart\_goods\_money | add: 10 }} </b></div>

</div>

</div>

<div class=Horder\_submit clearfix">

<input type="submit" valuer"提交订单 n/>

</div>

</form>

我们使用了一个form表单来提交订单数据，该form表单将订单数据提交到/cart/submit\_ 页面进行处理，代码如下：

<form action="/cart/submit\_order/" method="post°>

(% csrf\_token %)

其中(% csrf\_token %)主要用于安全设置，这是Django提供的防护措施，用于防护跨站请 求伪造。如果提交的表单是以post方式提交的话,那么需要我们在form表单内部加上该代码。

代码{{ cart\_goods\_money|add: 10 }}的含义为 cart goods money 的值加上 10。add 是 Django 模板语法中提供的过滤器函数，使用语法如下：

{{变量丨过滤器函数：参数}}

25J2d3醍置访阍鬣鬣

在ttsx/urls.py模块中配置当前视图对应的请求网址,新增配置如下: url(r1Acart/place\_order/$', place\_order), 这样，就可以测试提交订单的功能了。

25J2.4

用户填写完收货信息之后，单击“提交订单”按钮，将表单提交到了/srt/wbmit\_orM/页 面。我们在cwt应用下的views.py模块中新增submit\_order视图函数用于处理订单提交，该视 图函数实现如下：

def submit\_order(request): "° n保存订单"°

#获得订单信息

addr = request o POST.get(\* addr!z \* \*) recv = request«POST.get(\* recv', , \*) tele = request o POST.get(1 tele \* r \* \* ) extra = request« POST„get(v extra \* z 1 ) #保存订单信息

order\_info = Orderinfo() order\_info.order\_addr = addr order\_info.order\_tele = tele order info.order recv = recv

order\_info.order\_extra = extra

#生成订单编号 order\_info•order\_id = str(int(time.time() \* 1000)) + str(int(time.clock() \* 1000000))

order\_info。save() #跳转页面 response = redirect('/cart/submit\_\_success/?id=%s\* %

order\_inf o. order\_\_id)

#保存订单商品信息 - "

for goods\_\_idr goods\_num in request s COOKIES . items ():

if goods\_id == ? csrftoken' : continue

#查询商品信息 cart\_goods = Goodslnfo。objects•g㊀t(id=goods\_id)

#创建订单商品信息

order\_goods = OrderGoods() order\_\_goods . goods\_inf o = cart\_\_goods order\_\_goods o goods\_order = order\_inf o order\_\_goods . goods\_\_num = goods\_num order\_\_goods。save ()

#删除购物车信息 response。delete\_cookie(goods\_id)

return response

然后配置该视图与网址的对应关系，编辑ttsx/urls.py模块如下:

url (r J Acart/submit\_\_order/$ \* r submit\_order),

当订单提交成功之后，删除购物车中的商品信息，跳转到subniit\_success页面，并将订单 编号传递过去。

'订单提奏成晒页斯助離实现

订单提交成功页面主要用于展示订单提交成功之后的数据。

25.13J

在cart应用下的views.py模块中新增视图函数submit\_success,代码如下:

def submit\_success(request):

° n °显示订单结果""°

order\_id = request.GET„get(sidJ) order\_\_info = Orderinfo o obj ects；get(order\_id=order\_id) order\_goods\_list =

OrderGoods . objects .filter (goods\_ order=order\_info)

#商品总价

totla\_money = 0

#商品怎数量

total\_num = 0

for goods in order\_goods\_list: goods o total\_money = goods.goods\_num \*

goods.goods\_info.goods\_price totla\_money += goods.total\_money total\_num += goods.goods\_num

return render(request, , success.html', {

\* order\_info ?: order\_infoz

5 order\_goods\_list1: order\_goods\_list 1totla\_money': totla\_moneyz

5 total\_num': total\_num))

配置该视图与网址的对应关系,编辑ttsx/urls.py模块如下:

url (r 1 ^cart/submit\_\_success/$ ? 7 submit\_success) ?

至此,整个天天生鲜项目已经开发完成,我们可以访问该项目的所有页面和功能了。

项SB会

本章使用Django框架制作了一个天天生鲜在线购物项目网站，通过实现项目，我们学习 T如何使用Django框架快速搭建一个Web项目,并体会了 Django框架的运行机制。

通过本章项目的学习，还可以体会到MTV架构与MVC框架的关系。在Django框架中， 模板和视图相当于MVC架构的View,模型相当于MVC中的Model,而Django框架承担了 控制器Controller的工作。Django框架根据URLconf文件的设置,对给定的URL调用适当的 Python函数。由于控制层功能由Django自动完成，程序员只需要编写非常少的代码就可以完 成很多事情，从而大大提高了工作效率。