《Django 教程》

- 讲师: 魏明择
- 时间: 2019

目录

创建数据对象实验案例讲解

- 目标
  - 了解Django框架开发的基本步骤
  - 能够搭建网站来保存图书数据信息,能实现图书信息的添加，删除,修改，查看操作基本操作
- 基本功能:
  - 添加图书，查看所有图书，修改图书售价，删除图书
- MySQL 数据库名命名为: my_bookstore
- 基本步骤为:
  1. 创建工程配置数据库
  2. 创建 bookstore 应用能够显示主页功能
  3. 为存放图书信息，创建模型类。
  4. 实现添加新书功能
  5. 实现查看新书功能
  6. 实现修改图书 零售价 信息功能
  7. 实现删除图书信息
- 详细步骤如下:

    1. 创建工程配置数据库

        1. 创建Django工程

```
$ django-admin startproject mywebsite_bookstore
$ cd mywebsite_bookstore
$ ls  # 查看当前工程目录
```

        2. 创建一个新的数据库 my_bookstore

```
$ mysql -u root -p123456
Enter password:
SQL> create database my_bookstore default charset utf8
collate utf8_general_ci;
```

        3. 修改mywebsite_bookstore 工程的的配置文件 settings.py 文件配置 MySql 数据库连接

```
# file : mywebsite_bookstore/settings.py
DATABASES = {
```

```
    'default' : {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'my_bookstore',  # 数据库名称,需要自己定义
        'USER': 'root',  # 修改为自己的用户名
        'PASSWORD': '123456',  # 管理员密码
        'HOST': '127.0.0.1',  # 修改自己的IP地址
        'PORT': 3306,   # 一般不要修改
    }
}
```

4. 添加 mysql 数据库 python 接口支持
    - 修改项目中__init__.py 加入如下内容来提供pymysql引擎的支持

```
# file : mywebsite_bookstore/__init__.py
import pymysql
pymysql.install_as_MySQLdb()
```

    - 此处可以用 '$ python manage.py runserver' 来验证数据库是否配置成功

2. 创建 bookstore 应用能够显示主页功能
    1. 创建 bookstore app 应用

```
$ python3 manage.py startapp bookstore
```

    2. 在 settings.py 中注册此app

```
# file : mywebsite_bookstore/settings.py
INSTALLED_APPS = [
    ...
    'bookstore',  #  新加此行
]
```

    3. 在当前工程文件夹内添加存放模板的文件夹

```
$ mkdir templates
```

    4. 添加配置模板文件夹

```
TEMPLATES = [
    {
        ......
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  # 加
```

```
入模板文件目录
        ......
    },
]
```

5. 在模板文件夹mywebsite_bookstore/templates下创建`index.html`模版内容如下:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>书库首页</title>
</head>
<body>
    <ul>
        <li><a href="/bookstore/add">添加新图书</a></li>
        <li><a href="/bookstore/list_all">显示所有图书</a>
</li>
    </ul>
</body>
</html>
```

6. 为bookstore/views.py 中添加视图 homepage 函数

```
# file : bookstore/views.py

def homepage(request):
    return render(request, 'index.html')
```

7. 为 bookstore app 添加子路由文件urls.py

```
# 创建文件: bookstore/urls.py
from django.conf.urls import url
from . import views
urlpatterns = [
    url(r'^$', views.homepage),
]
```

8. 在mywebsite_bookstore/urls.py 将主路由的/bookstore指向此分支路由

```
# file : mywebsite_bookstore/urls.py
from django.conf.urls import url
from django.conf.urls import include  # <-- 新导入 include 函
数
urlpatterns = [
```

```
    ...
    url(r'^bookstore/', include('bookstore.urls')),
]
```

9. 测试程序是否能显示bookstore中的主页/bookstore/
    - 启动程序进行测试
        - `$ python3 manage.py runserver`
    - 在浏览器中输入地址:
        - http://127.0.0.1:8000/bookstore

3. 为存放图书信息，创建模型类。

    1. 修改 bookstore/models.py 添加数据模型类表.
        - 添入一个模型类,有如下五个字段
            1. title - 书名, CharField, 唯一不能重复
            2. price - 定价, DecimalField, 添加最大长度7位有效数字，小数点以后保留2位
            3. pub_house - 出版社, CharField, 最大长度100字节,缺省参数
            4. market_price - 零售价, DecimalField，最大长度7位有效数字，小数点以后保留2位，缺省值是9999
        - 示例代码

```
# file : bookstore/models.py
from django.db import models


class Book(models.Model):
    title = models.CharField("书名", max_length=50,
default="untitled")
    pub_house = models.CharField("出版社",
max_length=100, default='')
    price = models.DecimalField('定价', max_digits=7,
decimal_places=2, default=0.0)
    market_price = models.DecimalField('零售价',
max_digits=7, decimal_places=2, default=9999)
```

    2. 生成迁移文件并执行迁移

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

        - 进入MYSQL 数据库查看数据表是否自动生成

```
$ mysql -uroot -p123456
SQL> show databases;
SQL> use my_bookstore;
SQL> show tables;
```

- 注: 看到 bookstore_book 数据表意为上述操作成功

4. 实现添加新书功能

    1. 添加新模板 templates/new_book.html 实现带有表单的html界面

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>添加新书</title>
</head>
<body>
    <form action="/bookstore/add" method="post">
        <div>
            <label for="id_new">书名</label>
            <input type="text" name="title" placeholder="请输
入新书名" required id="id_new">
        </div>
        <div>
            <label for="id_pub">出版社</label>
            <input type="text" name="pub_house"
placeholder="请输入出版社名" required id="id_pub">
        </div>
        <div>
            <label for="id_price">定价</label>
            <input type="text" name="price" placeholder="请输
入定价" required id="id_price">
        </div>
        <div>
            <label for="id_mk_price">零售价</label>
            <input type="text" name="market_price"
placeholder="请输入零售价" required id="id_mk_price">
        </div>
        <div>
            <input type="submit" value="添加新书">
        </div>
    </form>
</body>
</html>
```

    2. 在bookstore 应用内添加视图函数 new_book 来处理添加新书的业务逻辑

```
from django.http import HttpResponse
from . import models
def new_book(request):
    if request.method == 'GET':
        return render(request, 'new_book.html')
    elif request.method == 'POST':
```

```
        title = request.POST.get('title', '')
        pub_house = request.POST.get('pub_house', '')
        price = request.POST.get('price', '')
        market_price = request.POST.get('market_price', '')
        # 用Book.object管理器的create方法来创建新书
        abook = models.Book.objects.create(title=title,
                                           price=price,

market_price=market_price,
                                           pub_house=pub_house)
        print('添加新书,成功添加新书:', abook.title)
        return HttpResponse('<a href="/bookstore">添加新书成
功，点我跳转到首页</a>')
```

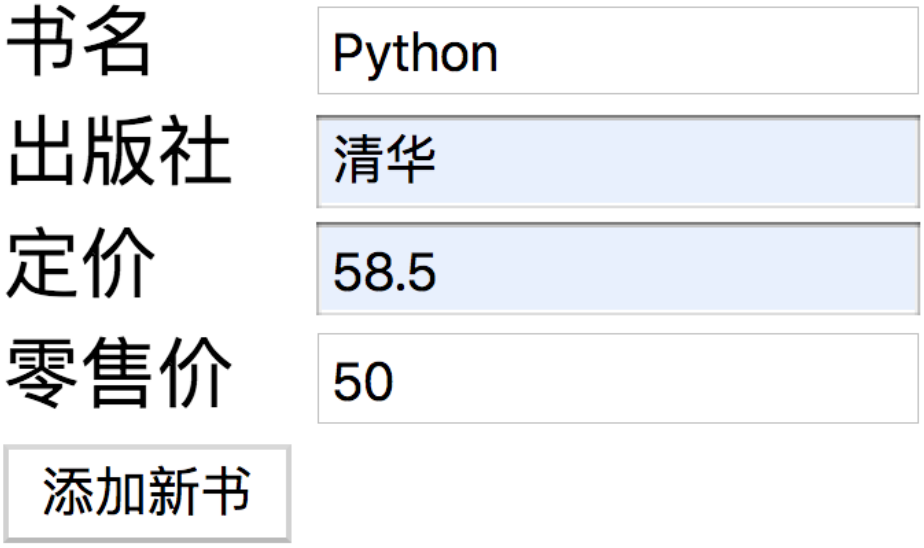
3. 在bookstore/urls.py 中添加 /bookstore/add 路由来映射此视图函数

```
urlpatterns = [
    url(r'^$', views.homepage),
    url(r'^add$', views.new_book),
]
```

4. 去掉项目配置文件 settings.py 中的 CSRF中间件

```
MIDDLEWARE = [
    ...
    # 'django.middleware.csrf.CsrfViewMiddleware',  # 注释掉此
行
    ...
]
```

5. 测试添加图书是成功
   - 进入网址http://127.0.0.1:8000/bookstore/add
   - 添加图书



- 显示

  添加新书成功，点我跳转到首页

- 查看数据库显示如下:

```
mysql> select * from bookstore_book;
+----+--------+-------+----------------------+---------
-----+
| id | title  | price | pub_house            |
market_price |
+----+--------+-------+----------------------+---------
-----+
|  1 | Python | 58.50 | 清华大学出版社       |
50.00 |
+----+--------+-------+----------------------+---------
-----+
1 rows in set (0.00 sec)
```

5. 实现查看新书功能

    1. 添加新模板 templates/book_list.html 实现带有表单的html界面

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>图书列表</title>
</head>
<body>
    <a href="/bookstore">返回首页</a>
    <table>
        <tr>
            <th>书名</th>
```

```
            <th>出版社</th>
            <th>定价</th>
            <th>零售价</th>
            <th>修改</th>
            <th>删除</th>
        </tr>
    {% for abook in books %}
        <tr>
            <td>{{ abook.title }}</td>
            <td>{{ abook.pub_house }}</td>
            <td>{{ abook.price }}</td>
            <td>{{ abook.market_price }}</td>
            <td><a href="/bookstore/mod/{{ abook.id }}">修改
</a></td>
            <td><a href="/bookstore/del/{{ abook.id }}">删除
</a></td>
        </tr>
    {% endfor %}
    </table>
</body>
</html>
```

2. 在bookstore 应用内添加视图函数 list_books 来处理获取图书列表，显示图书的目的

```
def list_books(request):
    books = models.Book.objects.all()
    return render(request, 'book_list.html', locals())
```

3. 在bookstore/urls.py 中添加 /bookstore/add 路由来映射此视图函数

```
urlpatterns = [
    ...
    url(r'^list_all', views.list_books),  # <<== 此行是新加的
]
```

4. 测试显示图书功能
    - 启动 网站服务器 'python3 manage.py runserver'
    - 进入网址http://127.0.0.1:8000/bookstore/list_all 显示添加的所有图书信息

6. 实现修改图书 零售价 信息功能

1. 添加新模板 templates/mod_price.html 实现带有表单的html界面

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>修改价格</title>
</head>
<body>
    <form action="/bookstore/mod/{{ abook.id }}"
method="post">
        <div>书名: {{ abook.title }}</div>
        <div>版社名: {{ abook.pub_house }}</div>
        <div>定价: {{ abook.price }}</div>
        <div>
            <input type="text" name="market_price" value="{{
abook.market_price }}"placeholder="零售价" required>
        </div>
        <div>
            <input type="submit" value="修改价格">
        </div>
    </form>
</body>
</html>
```

2. 在bookstore 应用内添加视图函数 mod_book_info 来处理获修改图书的信息

```
def mod_book_info(request, book_id):
    # 先根据book_id 找到对应的一本书
    try:
        abook = models.Book.objects.get(id=book_id)
    except:
        return HttpResponse("没有找到ID为" + book_id + "的图书
信息")

    if request.method == 'GET':
        return render(request, "mod_price.html", locals())
    elif request.method == 'POST':
        try:
            m_price = request.POST.get('market_price',
'0.0')
            abook.market_price = m_price
            abook.save()  # 提交修改
            return HttpResponse("修改成功")
        except:
            return HttpResponse("修改失败")
```

3. 在bookstore/urls.py 中添加 /bookstore/add 路由来映射此视图函数

```
urlpatterns = [
    ...
    url(r'^mod/(\d+)$', views.mod_book_info),
]
```

4. 测试修改图书价格功能
    - 启动 网站服务器 'python3 manage.py runserver'
    - 进入网址http://127.0.0.1:8000/bookstore/list_all 点击某个图书的修改信息

7. 实现删除图书信息

    1. 在bookstore 应用内添加视图函数 del_book 来处理获删除图书的信息

```
from django.http import HttpResponseRedirect  # 改入
HttpResponseRedirect模块用于 重定向url
def del_book(request, book_id):
    try:
        abook = models.Book.objects.get(id=book_id)
        abook.delete()
        return HttpResponseRedirect('/bookstore/list_all')
    except:
        return HttpResponse("没有找到ID为" + book_id + "的图书
信息,删除失败")
```

    2. 在bookstore/urls.py 中添加 /bookstore/del 路由来映射此视图函数

```
urlpatterns = [
    ...
    url(r'^del/(\d+)$', views.del_book),
]
```

    3. 测试修改图书价格功能
        - 启动 网站服务器 'python3 manage.py runserver'
        - 进入网址http://127.0.0.1:8000/bookstore/list_all 点击某个图书的删除来删除该图书

8. 将以下表格中的内容存放到数据库中

```
+----+---------+-------+--------------+----------------+
| id | title   | price | market_price | pub_house      |
+----+---------+-------+--------------+----------------+
|  1 | C       | 30.00 |        35.00 | 清华大学出版社 |
|  2 | C++     | 40.00 |        45.00 | 清华大学出版社 |
|  3 | Java    | 50.00 |        55.00 | 清华大学出版社 |
|  4 | Python  | 20.00 |        25.00 | 清华大学出版社 |
|  5 | Python3 | 60.00 |        65.00 | 清华大学出版社 |
|  6 | Django  | 70.00 |        75.00 | 清华大学出版社 |
|  7 | JQuery  | 90.00 |        85.00 | 机械工业出版社 |
|  8 | Linux   | 80.00 |        65.00 | 机械工业出版社 |
|  9 | Windows | 50.00 |        35.00 | 机械工业出版社 |
| 10 | HTML5   | 90.00 |       105.00 | 清华大学出版社 |
+----+---------+-------+--------------+----------------+
```

9. 扩展功能(自己实现,不提供源代码)

    1. 写一个页面，让用户输入输入一个价格，列出所有低于这个价格的图书（使用 `__lt`查询谓词)
    2. 写一个页面，输入一个图书名字的一部分，列出所有包含此关键字的图书（使用 `__contains`查询谓词)
    3. 写一个页面来列出所有降价销售的图书(使用F()对象进行查询)
    4. 写一个页面显示出不是"清华大学出版社" 且 价格小于50元的图书(提示: 使用Q()对象,~Q(pub_houst="清华大学出版社")&Q(price__lt=50))