

Django实战小型图书人物信息网页(MVT的综合运用)

原创 JAP君 Python进击者

2020-02-11原文



小白学Django系列:

- 小白学Django第一天| MVC、MVT以及Django的那些事
- 小白学Django第二天| Django原来是这么玩的！
- 小白学Django第三天| 一文带你快速理解模型Model
- 小白学Django第四天| Django后台管理及配置MySQL数据库
- 小白学Django第五天| 视图View的初步使用
- 小白学Django第六天| 一文快速搞懂模板的使用
- 持续更新中...

主要内容

本文主要总结 Django

系列前面所学的知识，将前面所学的知识通过一个小案例结合起来，让大家对于 MVT 模式更加的熟练。

图书案例

之前我们所学的知识都是基于BookManager的案例来的，这次我们主要实现两个界面，大家可以先看一下。



图书信息如下：

- [西游记](#)
- [水浒传](#)
- [活着](#)

图一

西游记:

- 人物名称: 孙悟空
年龄: 22
- 人物名称: 唐僧
年龄: 30

活着:

- 无相关信息

图三

整体逻辑:

首先我们进入的是图一的界面，在图一的界面我们可以点击每一本书，点击每一本书后，我们可以访问相关书籍的任务信息，如果这本书在数据库中有数据则展示数据,如果没有数据,那么就显示"无相关信息"

一、图一的具体实现:

1. 创建 books.html 文件
2. 在 view.py 中编写 show_books 函数

show_books 函数主要用来获取书籍信息

```
def show_books(request):  
    # 1. 通过M来查找数据库中的书籍信息  
    book = BookInfo.objects.all()  
    # 2. 把数据返回HTML文件  
    return render(request, "Book/books.html", {"books": book})
```

获取书籍信息的方法很简单，在我 Django 系列文章中有详细写过。

3. 编写 books.html 文件

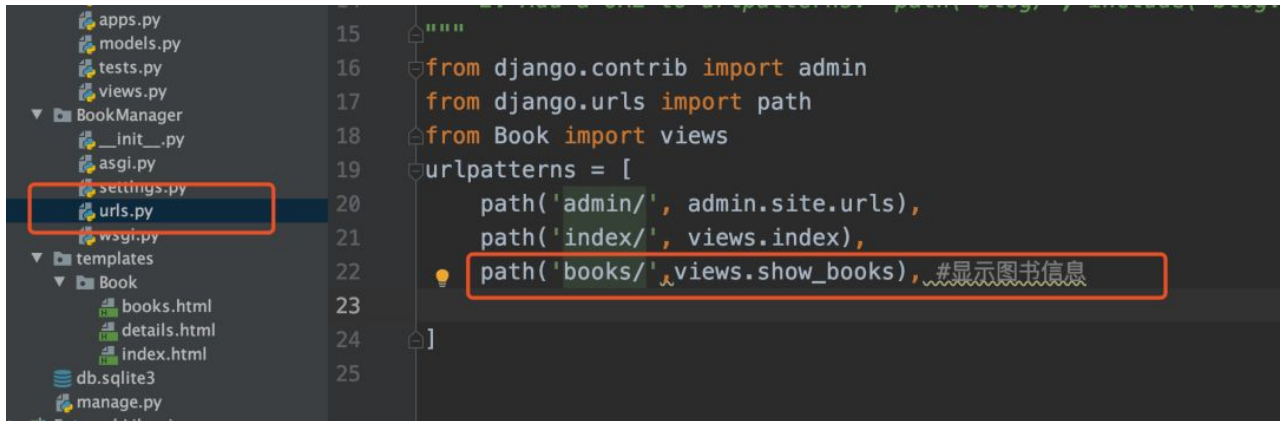
因为我们只是为了巩固之前学的 Django 知识，所以前端我们写的比较简陋

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8">  
    <title>Show Books</title>  
  </head>  
  <body>  
    <h1>图书信息如下: </h1>  
    <ul>  
      {% for i in books %}  
        <li><a href="{ i.id }">{{ i.book_title }}</a></li>  
      {% endfor %}  
    </ul>  
  </body>  
</html>
```

如果相关的语法知识看不懂，建议大家去公众号菜单栏底部找 Django 系列文章看看。

4. 配置相关 URL

我们需要去 `url.py` 文件去配置 `show_books` 视图函数的 `url` 路径



```
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from Book import views
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('index/', views.index),
22     path('books/', views.show_books), #显示图书信息
23 ]
24
25
```

5. 配置完成后,我们可以运行服务器:

```
python manage.py runserver
```

最后可以看到显示和图一是一样的

图书信息如下:

- [西游记](#)
- [水浒传](#)
- [活着](#)

当然这里面的数据我可能和你的不一样，有关于数据的增删改查，我在之前文章中也详细写过了。

二、图二的具体实现

有关于图二的实现可能会稍显复杂，我们再来回顾一下思路。
我们通过点击图一的超链接跳转到该书籍的人物介绍页面，然后在图二界面显示出人物介绍信息。

我们可以通过点击超链接，返回该书籍的 id
给后端的视图函数并且跳转至 /id
界面，然后在视图函数中根据返回的 id
来查询相关书籍人物信息，这样就非常的简单了。

1. 创建 details.html 文件
2. 此时的 books.html 文件代码应该如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Show Books</title>
</head>
<body>
<h1>图书信息如下： </h1>
<ul>
  {% for i in books %}
    <li><a href="{ { i.id } }">{ { i.book_title } }</a></li>
  {% endfor %}
</ul>
</body>
</html>
```

3. 编写 details_info 视图函数

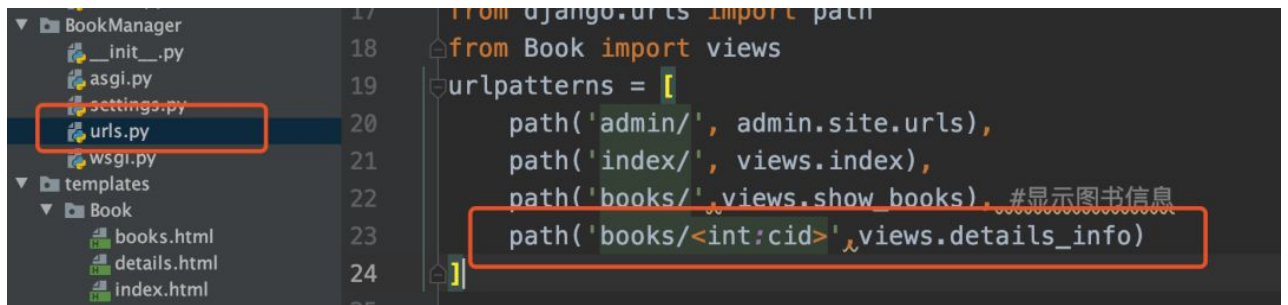
```
def details_info(request, cid):  
    # 1. 根据id来查找书籍信息  
    book = BookInfo.objects.get(id=cid)  
    # 2. 根据书籍信息找到人物信息  
    people = book.people_set.all()  
    # 3. 使用模板  
    return render(request, "Book/details.html", {'book': book, 'people': people})
```

整体的代码相信大家能够轻易理解，首先通过 **id** 返回该 **id** 所对应的 **BookInfo** 对象，然后根据该对象所关联的任务信息，来获取相关信息。

4. 编写 details.html 文件

```
<!DOCTYPE html>  
<html lang="en" xmlns="http://www.w3.org/1999/html">  
<head>  
    <meta charset="UTF-8">  
    <title>people_info</title>  
</head>  
<body>  
<h1>{{ book.book_title }}:</h1>  
  
    {% for i in people %}  
        <li>人物名称: {{ i.name }}</li>  
        年龄: {{ i.age }}  
    {% endfor %}  
</ul>  
</body>  
</html>
```

5. 配置 details_info 视图函数对应的 url



The screenshot shows a code editor with a file explorer on the left and code on the right. The file explorer shows a project named 'BookManager' with files like 'urls.py' (highlighted with a red box) and 'templates' folder. The code on the right shows Django URL patterns. Line 22 has a comment '#显示图书信息' (Display book information) next to the 'show_books' view. Line 23 shows a URL pattern with a converter: 'path('books/<int:cid>', views.details_info)'. This line is also highlighted with a red box.

```
17 from django.urls import path
18 from Book import views
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('index/', views.index),
22     path('books/', views.show_books), #显示图书信息
23     path('books/<int:cid>', views.details_info)
24 ]
```

此处我们运用了转换器，将 id

通过转换器传送给后端视图函数，这里需要注意，**转换器中的变量名需要和相应视图函数中参数的变量名称相同，例如这里的 cid。否则会报错！**

6. 点击访问页面

西游记:

- 人物名称: 孙悟空
年龄: 22
- 人物名称: 唐僧
年龄: 30

可以发现我们实现了图二的功能。

但是大家有没有想过，如果我没有给这本书相关的人物信息，那岂不是返回一个空白给我？所以接下来教大家一个很简单的方法来实现图三的功能。

三、图三的具体实现

其实实现起来非常简单，只需要两行代码即可解决。

```

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/html">

<head>

    <meta charset="UTF-8">

    <title>people_info</title>

</head>

<body>

<h1>{{ book.book_title }}:</h1>


    {% for i in people %}

        <li>人物名称: {{ i.name }}</li>

        年龄: {{ i.age }}

        {% empty %}

            <li>无相关信息</li>

        {% endfor %}

    </ul>

</body>

</html>

```

大家可以看到 `{% empty %}`，它的作用就是如果当前对象 `i` 是为空的话，那么它就会执行它下面的代码，如果不为空则不执行。

总结

在前面的文章中，我们已经大概介绍并且简单的使用了 MVT 的相关知识。我们也通过这个案例来进一步巩固 MVT 之间的整合开发。希望大家在学习基础的同时也伴随这一些具体案例。

精彩文章：

- 漫画：冒泡排序最牛逼的状态！
- 漫画：最最最最最简单的选择排序
- 小白学Django第六天| 一文快速搞懂模板的使用
- 【吐血整理】2019年所有精品文章分类汇总！必收藏！



精选留言

暂无...