第 1 章

走进Django 5

Django是一个开放源代码的Web应用框架，已成为当前Web开发的首选框架，如果想快速开发一个Web网站，那么Django可以说是最佳选择。本章将介绍Django的基础知识，包括安装Django 5、创建项目、使用Django的操作指令、程序调试方法等内容。

1.1 Django 5的新特性

Django由Python编写而成，最初用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站。它是一个CMS（内容管理系统）软件，于2005年7月在BSD许可证下发布，并以比利时的吉卜赛爵士吉他手Django Reinhardt的名字来命名。Django采用了MTV框架模式，即模型（Model）、模板（Template）和视图（Views），三者之间各自承担不同的职责。

- 模型：数据存取层，用于处理与数据相关的所有事务，例如存取数据、验证数据有效性、定义行为和数据之间的关系等。
- 模板：表现层，用于处理与呈现相关的决策，例如如何在页面或其他类型的文档中进行显示。
- 视图：业务逻辑层，负责存取模型并调取适当的模板，是模型与模板之间的桥梁。

Django的主要目的是简便、快速地开发由数据库驱动的网站。它强调代码复用，多个组件可以很方便地以插件形式服务于整个框架。Django有许多功能强大的第三方插件，可以便捷地开发出自己的工具包，这使得Django具有很强的可扩展性。此外，Django还强调快速开发和DRY（Do Not Repeat Yourself）原则。Django基于MTV的设计十分优美，具有以下特点：

- 对象关系映射（Object Relational Mapping，ORM）：通过定义映射类来构建数据模型，将模型与关系数据库连接起来。使用ORM框架内置的数据库接口可实现复杂的数据操作。
- URL设计：开发者可以自由设计URL（网站地址），同时支持正则表达式的使用。
- 模板系统：提供可扩展的模板语言，支持模板之间的继承。
- 表单处理：能够生成各种表单模型，并提供有效性验证功能。
- Cache系统：具备完善的缓存系统，支持多种缓存方式。
- Auth认证系统：提供用户认证、权限设置和用户组功能，具有较强的功能扩展性。

- 国际化：内置国际化系统，便于开发多种语言的网站。
- Admin后台系统：内置Admin后台管理系统，具有较强的扩展性。

目前，Django最新版本为5.0，它只支持Python 3.10或以上版本。新版本在模型、Admin后台系统、表单、异步身份验证功能、地理空间功能扩展、消息框架等功能上进行了改进。

（1）新增了模型字段GeneratedField，主要用于模型字段计算处理。以下是该字段的属性说明：

- expression：用于设置字段计算规则。
- output_field：用于设置字段数据类型。
- db_persist：表示字段是否占用存储空间，若db_persist=True，则模型字段在数据表中生成相应表字段；若db_persist=False，则数据表不生成相应表字段。

GeneratedField的使用示例如下：

```
from django.db import models
from django.db.models import F
class Square(models.Model):
    side = models.IntegerField()
    area = models.GeneratedField(expression=F("side") * F("side"),
                output_field=models.IntegerField(),db_persist=True)
```

（2）新增了模型字段属性db_default，用于设置表字段的默认值。它与字段属性default相似，但db_default是数据表的默认值，而default是模型字段的默认值。

（3）优化了模型字段属性choices，支持二级选项，如图1-1所示。

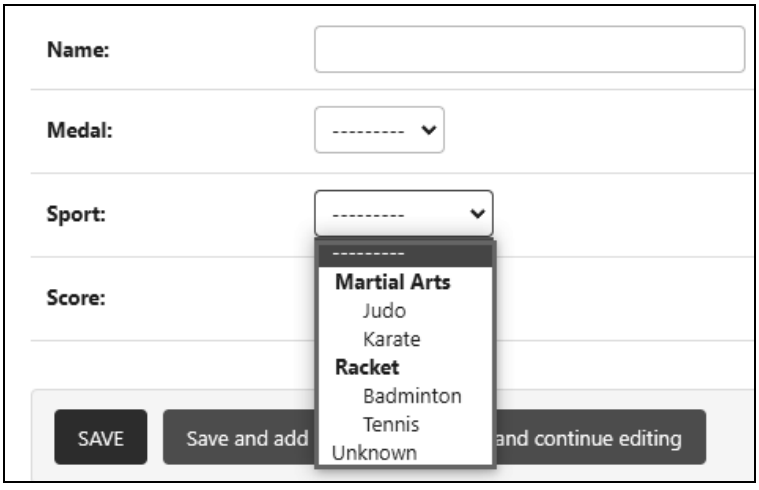


图 1-1　字段属性 choices

（4）新增了Admin后台系统配置属性show_facets，其属性值分别为admin.ShowFacets.ALWAYS、admin.ShowFacets.ALLOW、admin.ShowFacets.NEVER，主要用于控制过滤器的数据量，如图1-2所示。

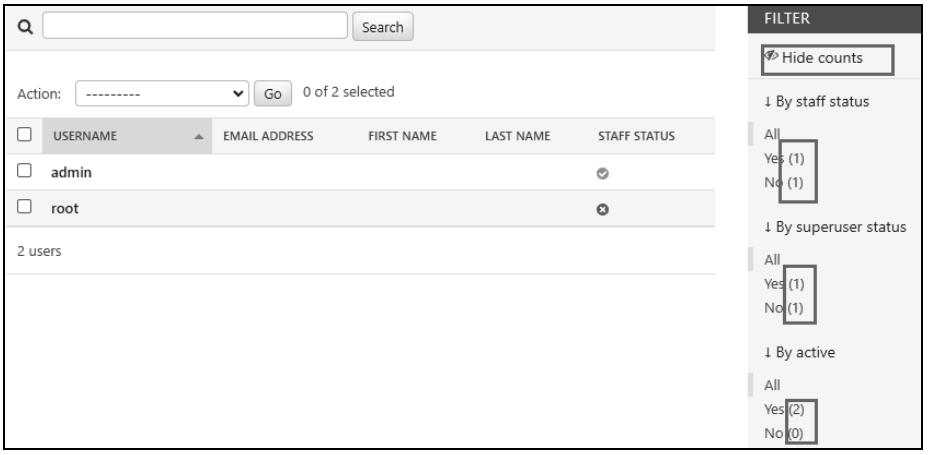


图 1-2　配置属性 show_facets

（5）优化了表单字段渲染，引入了“字段组”和字段组模板的概念。这一改进简化了Django表单字段的渲染，减少了渲染表单所需的HTML和模板代码量，使模板更简洁、更易于维护，示例如下：

```
# 优化前
<form>
...
<div>
  {{ form.name.label_tag }}
  ...
  {{ form.name }}
  <div class="row">
    ...
    <div class="col">{{ form.email.label_tag }}
      ...
      {{ form.email }}
    </div>
    <div class="col">{{ form.password.label_tag }}
      ...
      {{ form.password }}
    </div>
  </div>
</div>
...
</form>

# 优化后
<form>
...
<div>
  {{ form.name.as_field_group }}
  <div class="row">
    <div class="col">{{ form.email.as_field_group }}</div>
    <div class="col">{{ form.password.as_field_group }}</div>
  </div>
```

```
    </div>
    ...
    </form>
```

1.2　安装 Django 5

为了满足广大读者的需求，本书的开发环境要求为Windows并使用Python 3.10或更高版本。对于使用Linux或Mac OS X操作系统的读者，可以在虚拟机上安装Windows操作系统。

在安装Django之前，首先需要安装Python。读者可以从Python官网下载.exe安装包进行安装。建议安装Python 3.10或更高版本，因为Django 5只支持Python 3.10及以上的版本。完成Python的安装后，接下来安装Django，安装方法如下：

（1）使用pip进行安装。首先按快捷键Windows+R打开“运行”对话框，然后在对话框中输入“CMD”并按回车键，进入命令提示符窗口（也称为终端），最后在命令提示符窗口输入以下安装指令：

```
pip install Django
```

（2）输入上述指令后按回车键，系统将自行下载Django的最新版本并进行安装，我们只需等待安装完成即可。

完成Django的安装后，需要进一步校验安装是否成功：再次进入命令提示符窗口，输入“python”并按回车键，即可进入Python交互解释器。在交互解释器下输入以下校验代码：

```
>>> import django
>>> django.__version__
```

1.3　创 建 项 目

一个项目可以理解为一个网站。要创建一个Django项目，可以通过在命令提示符窗口输入创建指令来完成。首先，打开命令提示符窗口，然后将当前路径切换到D盘，再输入项目创建指令：

```
C:\Users\000>d:
D:\>django-admin startproject MyDjango
```

第一行指令是将当前路径切换到D盘；第二行指令是在D盘的路径下创建Django项目。指令中的“MyDjango”是项目名称，读者可自行命名。项目创建后，可以在D盘下看到新创建的文件夹MyDjango，在PyCharm下查看该项目的结构，如图1-3所示。

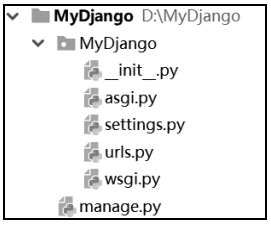


图 1-3　目录结构

MyDjango项目里包含MyDjango文件夹和manage.py文件，而MyDjango文件夹中又包含5个.py文件。项目的各个文件说明如下：

- manage.py：命令行工具，内置多种方式与项目进行交互。在命令提示符窗口下，将路径切换到MyDjango项目并输入“python manage.py help”，可以查看该工具的指令信息。
- __init__.py：初始化文件，一般情况下无须修改。
- asgi.py：开启一个ASGI服务，ASGI是异步网关协议接口。
- settings.py：项目的配置文件，项目的所有功能都需要在该文件中进行配置，配置说明会在第2章详细讲述。
- urls.py：项目的路由设置，设置网站的具体网址内容。
- wsgi.py：全称为Python Web Server Gateway Interface，即Python服务器网关接口，是Python应用与Web服务器之间的接口，用于Django项目在服务器上的部署和上线，一般不需要修改。

完成项目的创建后，接着创建项目应用。项目应用简称为App，每个App代表网站的一个功能。App的创建由文件manage.py实现，创建指令如下：

```
D:\>cd MyDjango
D:\MyDjango>python manage.py startapp index
```

从D盘进入项目MyDjango，然后执行python manage.py startapp XXX命令创建App，其中XXX是App的名称，读者可以自行命名。上述指令创建了网站首页，再次查看项目MyDjango的目录结构，结果如图1-4所示。

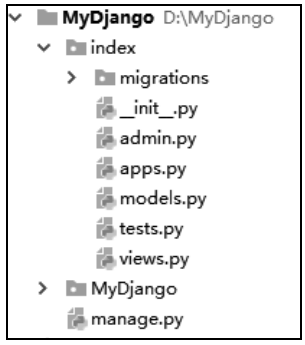


图 1-4　目录结构

MyDjango项目新建了index文件夹，它可以作为网站首页。在index文件夹中可以看到多个.py文件和1个migrations文件夹，说明如下：

- migrations：用于生成数据迁移文件，通过数据迁移文件可以在数据库里自动生成相应的数据表。
- __init__.py：index文件夹的初始化文件。
- admin.py：用于设置当前App的后台管理功能。
- apps.py：当前App的配置信息，一般情况下无须修改。
- models.py：定义数据库的映射类，每个类可以关联一张数据表，实现数据持久化，即MTV里面的模型（Model）。
- tests.py：自动化测试的模块，用于实现单元测试。
- views.py：视图文件，用于处理功能的业务逻辑，即MTV里面的视图（Views）。

完成项目和App的创建后，在命令提示符窗口输入以下指令启动项目：

```
C:\Users\000>d:
D:\>cd MyDjango
D:\MyDjango>python manage.py runserver 8001
```

将命令提示符窗口的路径切换到项目的路径，输入运行指令“python manage.py runserver 8001”，如图1-5所示。其中8001是端口号，如果指令里没有设置端口，端口就默认为8000。最后在浏览器上输入http://127.0.0.1:8001/，可看到项目的运行情况，如图1-6所示。

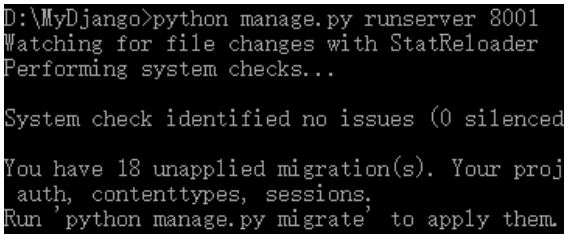


图 1-5　输入运行指令

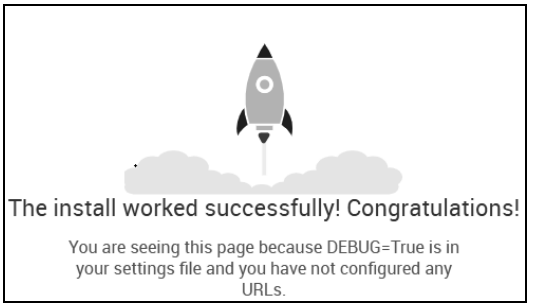


图 1-6　项目运行情况

1.4　PyCharm 创建项目

除了在命令提示符窗口中创建项目之外，还可以在PyCharm中创建项目。PyCharm是一种广受欢迎的Python集成开发环境（IDE），我们可以用它开发和调试代码和管理项目。

PyCharm必须为专业版才能创建与调试Django项目，社区版是不支持此功能的。首先，打开PyCharm，在左上方菜单栏中依次单击“File→New Project”，创建新项目，如图1-7所示。

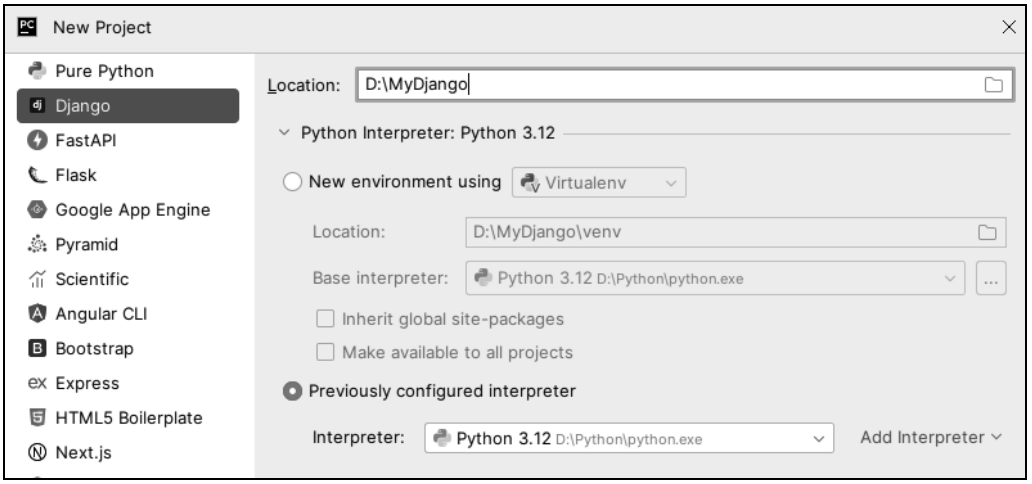


图 1-7　PyCharm 创建 Django

项目创建后，可以看到目录结构中多出了templates文件夹，该文件夹用于存放HTML模板文件，如图1-8所示。

接着，创建App，可以在PyCharm的Terminal中输入创建指令，创建指令与命令提示符窗口中输入的指令是相同的，如图1-9所示。

完成项目和App的创建后，启动项目。如果项目是由PyCharm创建的，就直接单击“运行”按钮启动项目，如图1-10所示。

如果项目是在命令提示符窗口中创建的，而又想要在PyCharm中启动项目，但PyCharm没有运行脚本，就需要为该项目创建运行脚本，如图1-11所示。

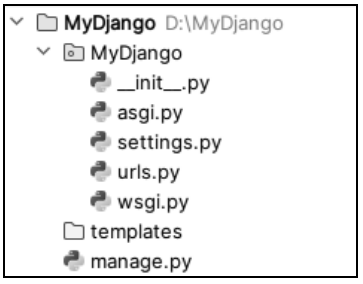


图 1-8　项目目录结构

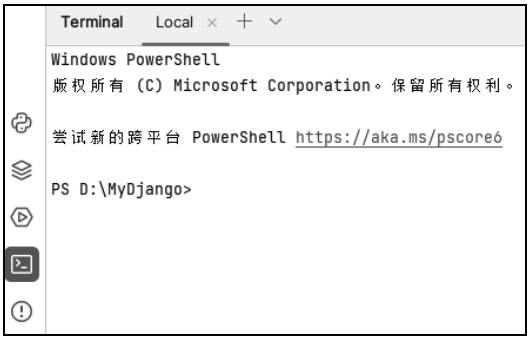


图 1-9　PyCharm 创建 App

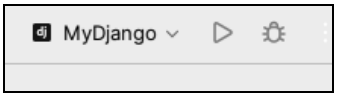


图 1-10　PyCharm 启动项目

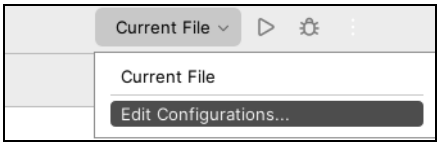


图 1-11　创建运行脚本

单击图1-11中的Edit Configurations就会出现Run/Debug Configurations界面，单击该界面左上方的 + 按钮，选择Django server，再输入脚本名字和Python安装目录，最后单击OK按钮即可创建运行脚本，如图1-12所示。

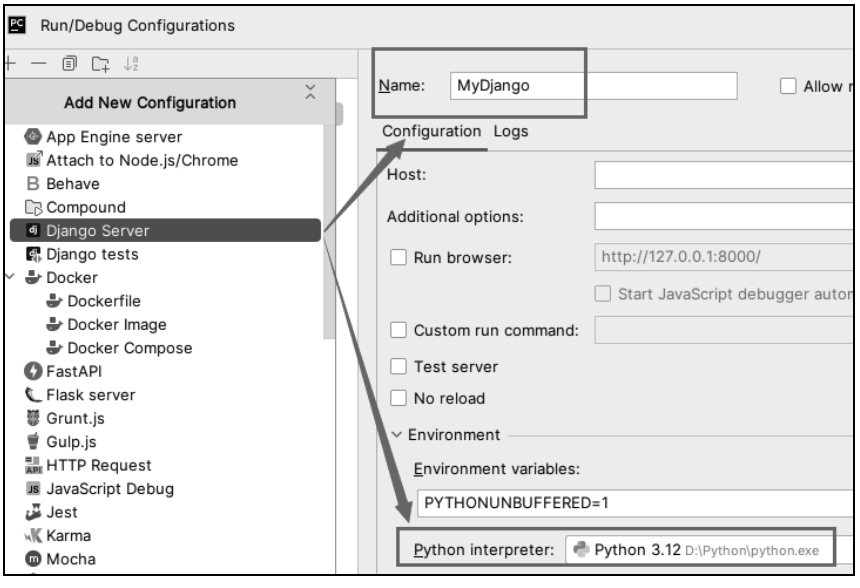


图 1-12　创建运行脚本

1.5　初试 Django 5

要学习Django，首先需要了解Django的操作指令。本节将大致介绍Django的操作指令，然后在MyDjango项目里编写Hello World网页，通过该网页来简单讲解Django的开发过程。

1.5.1　Django的操作指令

上一节介绍了如何在命令提示符窗口或PyCharm下创建Django项目和项目应用，无论是创建项目还是项目应用，都需要输入相关的指令才能实现，而这些都是Django内置的操作指令。

在PyCharm的Terminal中输入指令“python manage.py”后按回车键，即可看到相关的指令信息，如图1-13所示。

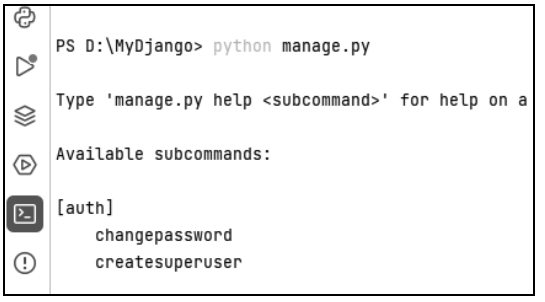


图 1-13　Django 指令信息

Django的操作指令共有31条，每条指令的说明以表格形式展示，如表1-1所示。

表 1-1　Django 操作指令说明

指　　令	说　　明
changepassword	修改内置用户表的用户密码
createsuperuser	为内置用户表创建超级管理员账号
remove_stale_contenttypes	删除数据库中已经不使用的数据表
check	检测整个项目是否存在异常
compilemessages	编译语言文件，用于项目的区域语言设置
createcachetable	创建缓存数据表，为内置的缓存机制提供存储功能
dbshell	进入Django配置的数据库，可以执行数据库的SQL语句
diffsettings	显示当前settings.py的配置信息与默认配置的差异
dumpdata	导出数据表的数据并以JSON格式存储，如python manage.py dump data index > data.json，这是index的模型所对应的数据导出，并保存在data.json文件中
flush	清空数据表的数据信息
inspectdb	获取项目所有模型的定义过程
makemessages	创建语言文件，用于项目的区域语言设置
loaddata	将数据文件导入数据表，如python manage.py loaddata data.json
makemigrations	从模型对象创建数据迁移文件并保存在App的migrations文件夹中
migrate	根据迁移文件的内容，在数据库里生成相应的数据表
optimizemigration	优化迁移操作并覆盖现有的迁移文件
sendtestemail	向指定的收件人发送测试的电子邮件
shell	进入Django的Shell模式，用于调试项目功能
showmigrations	查看当前项目的所有迁移文件

（续表）

指　　令	说　　明
sqlflush	查看清空数据库的SQL语句脚本
sqlmigrate	根据迁移文件内容输出相应的SQL语句
sqlsequencereset	重置数据表递增字段的索引值
squashmigrations	对迁移文件进行压缩处理
startapp	创建项目应用App
startproject	创建新的Django项目
test	运行App里面的测试程序
testserver	新建测试数据库，并使用该数据库运行项目
clearsessions	清除会话（Session）数据
collectstatic	收集所有的静态文件
findstatic	查找静态文件的路径信息
runserver	在本地计算机上启动Django项目

表1-1简单讲述了Django操作指令的作用，对于刚接触Django的读者来说，可能并不理解每个指令的具体作用，本节只对这些指令进行概述，读者只需要大概了解，在后续的学习中会具体讲述这些指令的使用方法。此外，有兴趣的读者也可以参考官方文档（https://docs.djangoproject.com/zh-hans/5.0/ref/django-admin/）。

1.5.2　开启Hello World之旅

相信读者现在已经对Django有了大概的认知，接下来我们将在MyDjango项目中开发Hello World网页，让读者打开Django的大门。

首先，在templates文件夹中新建一个index.html文件，该文件是Django模板文件。如果在命令提示符窗口下创建了MyDjango，则需要自行创建templates文件夹。项目目录结构如图1-14所示。

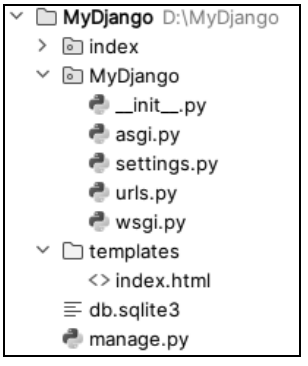


图 1-14　目录结构

接着，打开MyDjango文件夹的配置文件settings.py，找到配置属性INSTALLED_APPS和TEMPLATES，分别将项目应用index和模板文件夹templates添加到相应的配置属性中，其配置如下所示：

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # 添加项目应用index
    'index'
]
TEMPLATES = [
 {
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
```

```
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
          'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
          ],
        },
    },
]
```

Django的所有功能都需要在配置文件settings.py中设置，否则项目在运行的时候无法生成相应的功能。有关配置文件settings.py的配置属性将会在第2章讲述。

最后，在项目的urls.py（位于MyDjango文件夹中的urls.py）、views.py（位于项目应用index中的views.py文件）和index.html（位于templates文件夹中的index.html）中编写相应的代码，即可实现简单的Hello World网页。代码如下：

```
# MyDjango的urls.py
from django.contrib import admin
from django.urls import path
# 导入项目应用index
from index.views import index
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index)
]

# index的views.py
from django.shortcuts import render
def index(request):
    return render(request, 'index.html')

# templates的index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
</head>
<body>
    <span>Hello World!!</span>
</body>
</html>
```

在上述代码里可以简单地映射出用户访问网页的过程，说明如下：

- 当用户在浏览器中访问某个网址时，该网址将在项目中设置的路由（urls.py文件）里找到相应的路由信息。
- 然后，从路由信息里找到对应的视图函数（views.py文件），由视图函数处理用户的请求。

- 视图函数将处理结果传递给模板文件（index.html文件），由模板文件生成网页内容，并在浏览器中展示。

启动MyDjango项目，然后在浏览器中访问路由地址http://127.0.0.1:8000，即可看到Hello World网页，如图1-15所示。

图 1-15　Hello World 网页

注意 由于Django默认使用SQLite作为数据库，因此在启动MyDjango项目后，将在MyDjango的目录中自动新建一个名为db.sqlite3的文件。

1.6　调试 Django 项目

在开发网站的过程中，为了确保功能可以正常运行并验证是否实现了开发需求，开发人员需要对已实现的功能进行调试。Django提供了两种调试方式：PyCharm断点调试和调试异常。

1.6.1　PyCharm断点调试

我们知道，要使用PyCharm调试Django开发的项目，需要使用PyCharm专业版，因为社区版不具备Web开发功能。在使用PyCharm启动Django时，可以在PyCharm上看到一个带有爬虫图标的按钮，该按钮用于开启Django的调试模式，如图1-16所示。

单击图1-16中的调试按钮（带有爬虫图标的按钮），即可开启调试模式，在PyCharm的正下方可以看到相关的调试信息，如图1-17所示。

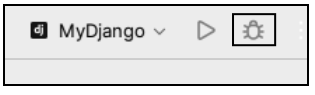


图 1-16　调试按钮

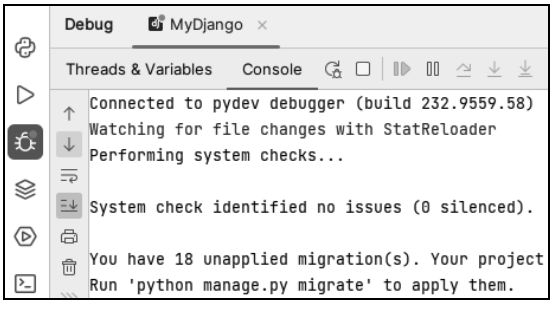


图 1-17　调试信息

在图1-17中的调试界面内可以看到多个操作按钮，其中常用的调试按钮的功能如表1-2所示。

表 1-2　常用的调试按钮的功能说明

按　钮	说　明
Console	显示项目的运行信息
Threads & Variables	显示程序的对象信息
	重新运行项目
	继续往下执行程序，直到下一个断点才暂停程序
	暂停当前运行的程序

（续表）

按　钮	说　明
	停止程序的运行
	查看所有断点信息
	清空Console的信息
	程序断点后，执行下一行的代码

下面将通过示例讲述如何使用PyCharm的调试模式。以MyDjango为例，在项目的index应用中，打开views.py文件，向视图函数index添加变量value，并在返回值return处设置一个断点，如图1-18所示。

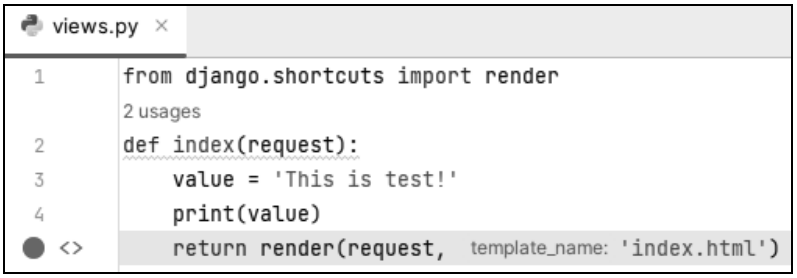


图 1-18　设置断点

在图1-18中单击方框，即可出现红色的圆点，该圆点代表断点已经设置好了。当项目开启调试模式并运行到断点所在的代码位置时，程序就会暂停运行。

运行MyDjango的调试模式，并在浏览器上访问127.0.0.1:8000，在PyCharm正下方的调试界面中可以看到相关的代码信息，如图1-19所示。

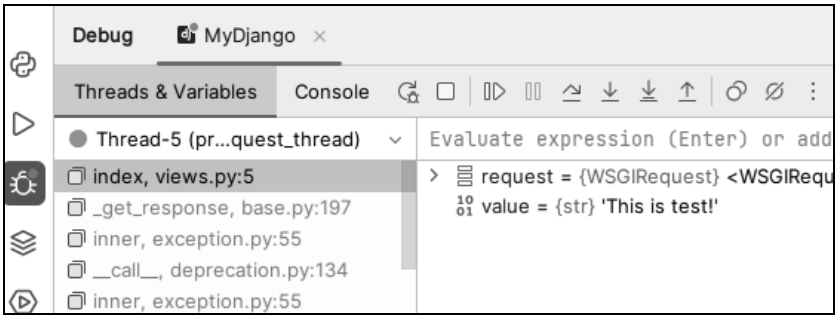


图 1-19　代码信息

调试界面Debug中的Frames是当前断点所在程序所依赖的程序文件。单击某个文件，Variables就会显示当前文件对应的程序所生成的对象信息。

单击 按钮，PyCharm就会自动往下执行程序，直到下一个断点才暂停程序。单击 按钮，PyCharm只会执行当前暂停位置的下一步代码，这样可以清晰地看到每行代码的执行情况。这两个按钮是断点调试最为常用的，它们能让开发者清晰地了解代码的执行情况和运行逻辑。

如果程序在运行过程中出现异常或者代码中设有输出功能（如print），那么这些信息就可以在PyCharm的正下方调试界面的Console中查看，如图1-20所示。

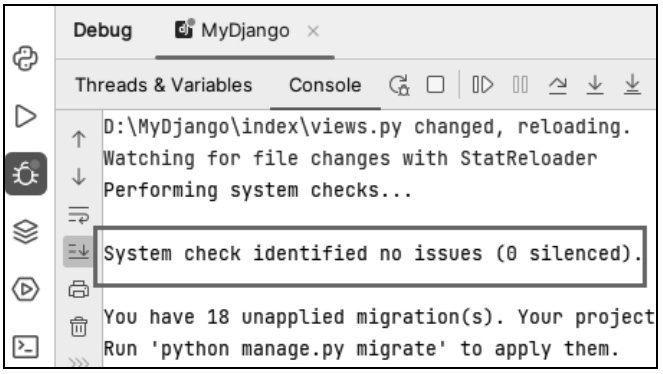
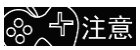


图 1-20　输出信息

启动项目时，在图1-20中的运行信息中看到“System check identified no issues (0 silenced)”信息，该信息表示Django对项目里所有的代码语法进行了检测，如果代码语法存在错误，在启动过程中就会报出相关的异常信息。

如果在图1-20中出现“This is test!”，就表示视图函数index成功执行了；如果出现“"GET / HTTP/1.1" 200”，表示浏览器成功访问了127.0.0.1:8000，其中200代表HTTP的状态码。



由于无法在模板文件（templates的index.html）中设置断点，因此无法使用断点调试对模板文件进行调试。相反，可以通过PyCharm调试界面的Console或浏览器开发者工具来进行调试。

1.6.2　调试异常

PyCharm的调试模式无法调试模板文件，而模板文件需要使用Django的模板语法。如果想调试模板文件，那么最有效的方法是查看PyCharm或浏览器提示的异常信息。

调试异常需要根据项目运行时所产生的异常信息进行分析。使用浏览器访问路由地址时，如果出现异常信息，就可以直接查看异常信息来找出错误位置。例如，在templates的模板文件index.html里添加错误的代码，如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
</head>
<body>
    {# 添加错误代码static#}
    {% static %}
    <span>Hello World!!</span>
</body>
</html>
```

当运行MyDjango项目并在浏览器访问http://127.0.0.1:8000时，PyCharm正下方的调试界面的Console中就会出现异常信息，从异常信息中可以找到具体的异常位置，如图1-21所示。

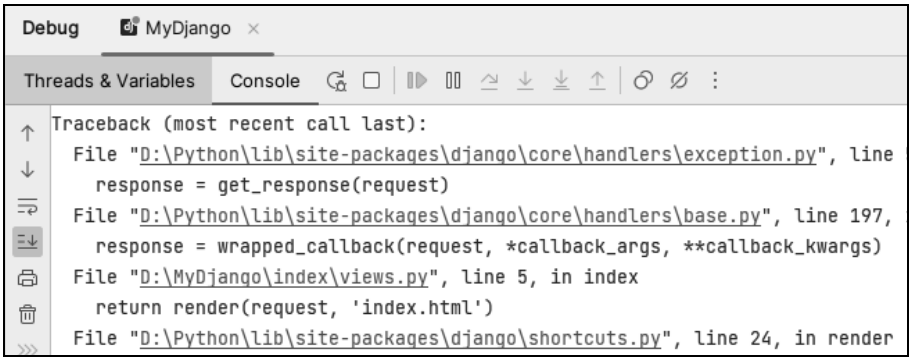


图 1-21　异常信息

除了在PyCharm正下方的调试界面的Console中查看异常信息外，还可以在浏览器上分析异常信息，比如模板文件index.html的错误语法。Django能够标记出错的位置，便于开发者进行调试和跟踪，如图1-22所示。

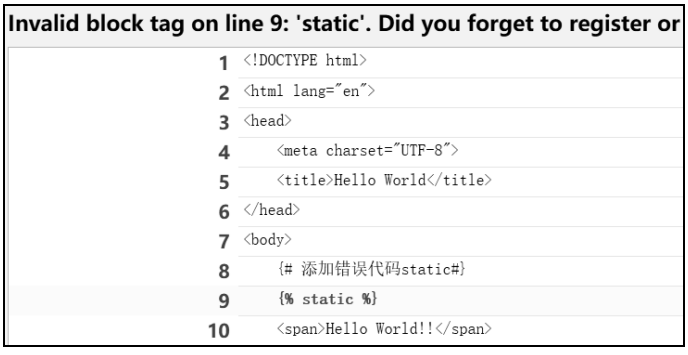


图 1-22　异常信息

还有一种常见的情况是，网页能正常显示，但网页内容出现部分缺失。对于这种情况，只能使用浏览器的开发者工具对网页进行分析处理。以templates的模板文件index.html为例，在其中添加正确的代码，但在网页中出现了内容缺失，如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
</head>
<body>
    {# 添加正确代码，但不出现在网页 #}
    <div>Hi,{{ value }}</div>
    <span>Hello World!!</span>
</body>
</html>
```

再次启动MyDjango项目并在浏览器中访问http://127.0.0.1:8000时，浏览器能够正常访问网页，但无法显示{{ value }}的内容。打开浏览器的开发者工具可以看到，{{ value }}的内容是不存在的，如图1-23所示。

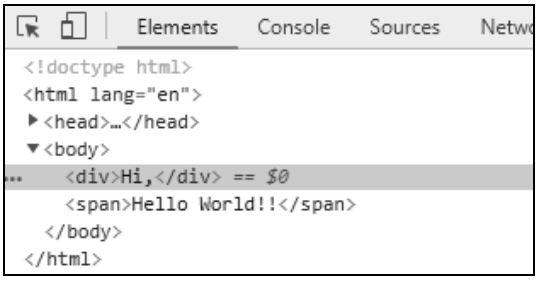


图 1-23　开发者工具

此外，浏览器的开发者工具对于调试AJAX和CSS样式非常有用。通过对生成的网页内容进行分析来反向检测代码的合理性是常见的手段之一。这种方法是通过校验结果与开发需求是否一致来调试项目功能的。

1.7　本 章 小 结

本章介绍了Django的概念及安装和基本使用，读者应重点了解和掌握以下内容：

（1）Django的MTV框架模式，即模型（Model）、模板（Template）和视图（Views），了解三者之间各自承担的不同职责。

（2）了解使用pip安装Django的方法，即：

```
pip install Django
```

（3）掌握如何创建Django项目，并了解Django项目的目录结构及各部分的含义。

（4）了解Django的操作指令以及使用Django编写Hello World网页的方法，同时掌握项目开发的调试方法。