第 1 章

Django 框架基础与环境搭建

Django 是一个开放源代码的 Web 应用框架，是由高性能的 Python 语言编写而成的。目前，基于 Python 语言的 Web 框架有很多款，而 Django 框架是其中应用范围最广、性能最优异、最具发展前景的一款。当今世界上，许多非常成功的 Web 网站和移动 App 都是基于 Django 框架开发的。

本章作为全书的开篇，主要介绍一下 Django 框架的基础知识、运行环境的搭建，以及开发工具的选择。同时，通过构建一个最基本的基于 Django 框架的 Web 应用程序（应用程序一般简称应用），帮助读者快速掌握 Django 框架的开发流程。

通过本章的学习可以掌握以下知识:

- Django框架的基础知识
- 如何搭建基于Django框架的开发环境
- 基于Django框架的Web应用程序的开发流程

1.1 Django框架基础

本节首先介绍一下Django框架的基础知识、Django框架的设计原理，以及MVC与MTV这两种模式之间的区别。

1.1.1 Django 框架的基础知识

Django（英文发音：`dʒæŋgəʊ）是一个开放源代码的Web应用框架，使用高性能的Python语言编写而成。Django框架的诞生，最初是用来开发和管理Lawrence Publishing Group（劳伦斯出版集团）旗下的新闻网站，它是一款属于CMS（内容管理系统）类的软件，并于2005年7月取得了BSD许可证下的发布权限。然后，经过设计人员的不断努力，Django 1.0版于2008年9月正式发布。

Django框架是一款高水准的、基于Python编程语言驱动的开源模型。Django框架的设计初衷是简便、快速地开发出易于维护的数据库驱动型网站，其所独具的代码复用功能，支持将各种组件以“插件”方式嵌入整个应用框架，从而极大地提高了应用开发的效率。Django框架自身具有很强大的扩展性，在开源社区中存在许多功能强大的第三方插件，设计人员可以非常方便地以“即插即用”的方式将它们应用到自己的项目中。

Django框架主要用于开发数据库驱动型网站，因此具有十分强大的数据库方面的功能。通过使用Python类的继承方式，只需几行代码就可以获取一个完整的、动态的数据库操作接口（Database API）。设计人员还可以通过执行SQL语句实现数据模型与数据库的解耦（即数据模型的设计不需要依赖于特定的数据库），由此通过简单的配置就可以轻松更换数据库。

Django框架自带功能强大的后台功能。设计人员通过在admin.py配置管理文件中写入所需实现功能的代码，就可以轻松地实现只有系统管理员才具有的功能权限，免去了再去设计管理员功能模块的烦琐工作。

Django框架拥有自身所独有的模板系统，该模板系统大大降低了开发者出错的概率。另外，因为模板系统设计简单、容易扩展、代码与样式采取分开设计的方式，所以代码查找起来更清晰、修改起来也更容易。

Django框架的缓存系统采用与memcached、Redis等结合使用的方式，提高了页面的加载速度。

Django框架在urls.py中通过正则表达式来匹配网址，并传递到对应的函数中。设计人员可以根据自己的习惯来自定义网址，具有完全的自主性。

Django框架对于多语言的国际化支持也非常友好。如果打算在网页中显示不同语言（如中文、英文等），设计人员只需要在页面文件的配置中稍微进行修改，就可以实现多种语言的无痕切换。

近年来，得益于Python编程语言地位的不断上升，Django框架的发展势头非常迅猛，版本的更新迭代速度也非常快。由Django官方网站提供的、最新的产品发布路线图（Release-Roadmap），如图1.1所示。

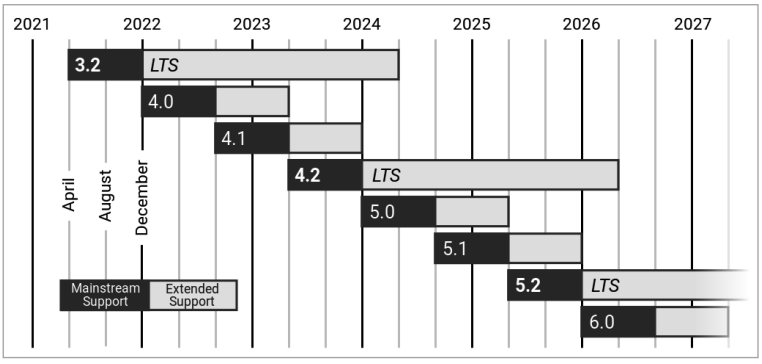

图 1.1　Django 产品发布路线图

由图1.1可知，目前最新的Django框架版本是5.0.1 LTS，规划在未来两年将更新到6.0 LTS版本。

1.1.2　Django 框架设计原理

相信大多数的Web开发者对于MVC（Model、View、Controller）设计模式都不陌生，该设计模式已经成为Web框架中一种事实上的标准了，Django框架自然也是一个遵循MVC设计模式的框

架。不过从严格意义上讲，Django框架采用了一种更为特殊的MTV设计模式，其中的“M”代表模型（Model），“T”代表模板（Template），“V”代表视图（View）。MTV模式是从MVC模式变化而来的。

那么，MTV模式的具体内容是什么呢？下面，我们将MTV拆分开来逐一进行详细介绍。

- 模型：表示的是数据存取层，处于MTV模式的底层。模型负责处理与数据相关的所有事务，包括如何存取数据、如何验证数据有效性和如何处理数据之间的关系等方面的内容。
- 模板：表示的是表现层，处于MTV模式的顶层。模板负责处理与表现相关的操作，包括如何在页面或者在其他类型文档中进行显示等方面的内容。
- 视图：表示的是业务逻辑层，处于MTV模式的中间层。视图负责存取模型及调取适当模板的相关逻辑等方面的内容，是模型与模板之间进行沟通的桥梁。

此外，MTV模式还需要一个URL分发器，其作用是将URL页面请求分发给不同的视图去处理，然后视图再调用相应的模型和模板。其实，仔细去品味就会发现，这个URL分发器所实现的就是MVC模式下的控制器（Controller）功能。URL分发器的设计机制是使用正则表达式来匹配URL，然后再调用相应的Python函数或方法。

任何一个Web前端设计模式都离不开控制器这个模块，它代表着业务处理的核心部分。我们在MTV模式中看不到控制器的设计，并不是Django框架没有设计该模块，而恰恰是Django将该模块的功能封装在底层了。这样做的好处就是将设计人员从烦琐的控制层逻辑中解脱出来，通过编写更少的代码来实现用户需求，而控制层逻辑交由Django框架底层自动去完成，从而大大地提高了设计人员的开发效率。

关于MTV模式的响应原理，可参考图1.2中的描述。

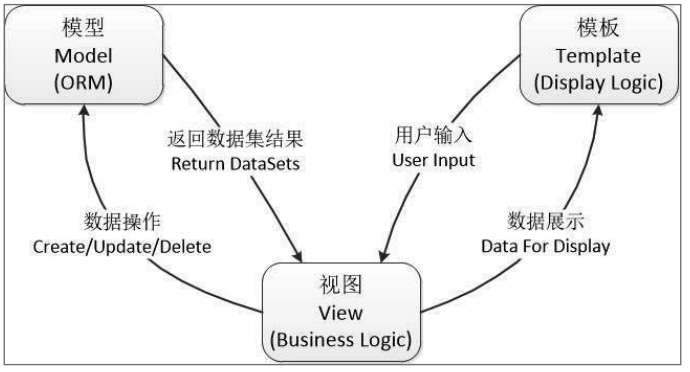

图 1.2　MTV 模式的响应原理

模板接收用户输入后交由视图去处理，视图负责连接模型进行数据操作，并将操作结果传递给模板进行展示，以上就是Django框架所设计的MTV模式的基本工作原理。

1.1.3　Django 框架工作机制

Django框架采用了MTV设计模式，在工作机制上自然也有些特别之处，其中最显著的地方就是视图部分。在图1.2所示的原理中，MTV模式中的视图是不负责处理用户输入的，这点就是最特殊的地方之一。

Django框架下的视图不负责处理用户输入，仅负责选择要展示的数据并传递到模板上。然后，由模板负责展示数据（展示效果），并最终呈现给终端用户。进一步来讲，就是Django框架将MVC中的视图解构为视图和模板两个部分，分别用于实现“展现数据”和“如何展现”这两部分功能，这样就可以实现将模板根据用户需求来随时更换，而不仅仅限制于内置的模板。

Django框架工作机制的流程如图1.3所示。

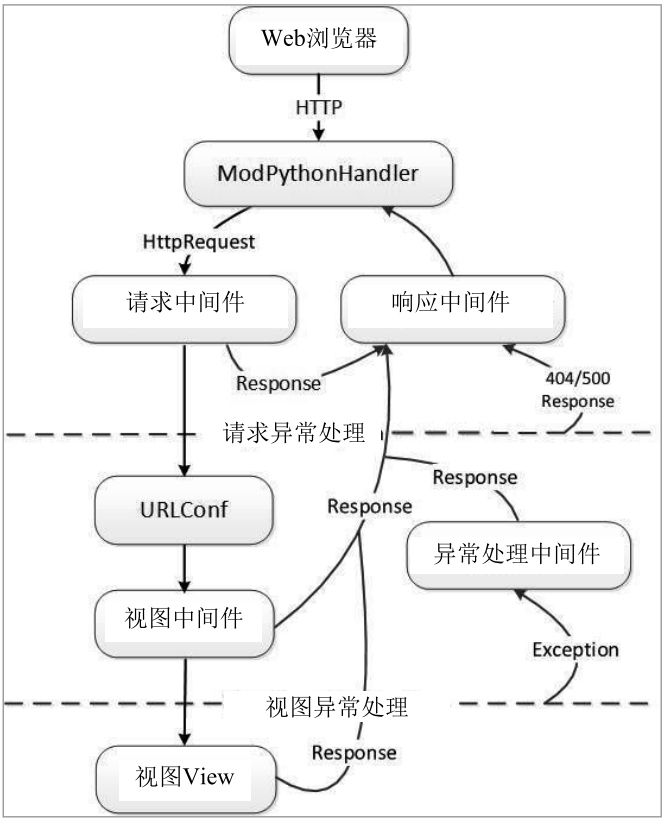

图 1.3　Django 框架工作机制

当启动Django服务器时，在同一目录下会自动加载配置文件（settings.py），该配置文件涵盖了项目所需的全部配置参数。其中，最重要的配置参数就是ROOT_URLCONF，它定义了Django服务器使用哪个Python模块来用作本项目的URLConf（一般默认是urls.py）。

当用户在Web浏览器（Web Browser）中访问URL时，Django服务器会接收到一个HTTP请求，通过服务器端特定的Handler（ModPythonHandler）创建HttpRequest并传递给中间件（Request Midware）进行处理，这些中间件起着功能增强的作用。

Django服务器会根据ROOT_URLCONF配置的参数来加载URLConf，然后按顺序逐个匹配URLConf中的URLpatterns，如果匹配成功，则会调用相关联的视图中间件函数，并把HttpRequest对象作为第一个参数向下传递。最后，通过视图返回一个HttpResponse对象（通常是Response）。

另外，Django框架还实现了完整的异常处理机制，主要是通过异常处理中间件（Exception Midware）来实现的。当系统出现异常时，异常处理中间件（Exception Midware）会截获并判断异常类型，并返回异常错误（404或500等）信息。

1.1.4　Django 框架用户操作流程

Django框架设计的MTV模式是基于传统的MVC模式的，本质上也是为了让各组件之间保持松耦合关系，只是定义上有些许不同。MVC模式之所以能够成为Web框架最流行的设计标准，正是因为它比较完美地契合了用户的操作流程。

MVC模式是软件工程中的一种通用的软件架构模式，同样也适用于Web应用程序。MVC将Web框架分为3个基本部分：模型、视图和控制器，并以一种插件式的、松耦合的方式将它们连接在一起。

在MVC模式中，模型负责编写具体的程序功能，建立业务对象与数据库的映射（ORM）；视图为图形界面，负责与用户的交互（HTML页面）；控制器负责转发请求，并对请求进行处理。

MVC模式的用户操作流程如图1.4所示。

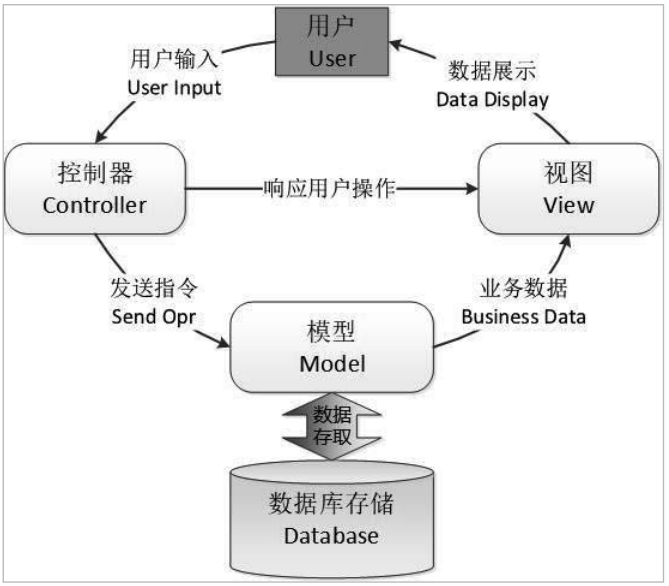

图 1.4　MVC 模式用户操作流程

正如前文中介绍的，Django框架的MTV模式指的是模型、模板和视图。最重要的是，MTV模式另外实现了一个URL分发器模块，其作用是将每一个URL页面请求分发给相应的视图进行处理，再由视图去调用相应的模型和模板。

Django框架的用户操作的流程图如图1.5所示。

如图1.5中的描述，用户通过浏览器向服务器端的URL分发器模块发起一个URL请求，这个URL请求会去访问视图函数（View.py）进行匹配，再进一步通过数据模型（Models）访问数据库进行数据操作，然后将操作结果逐级返回到模板，并最终返回网页给用户。

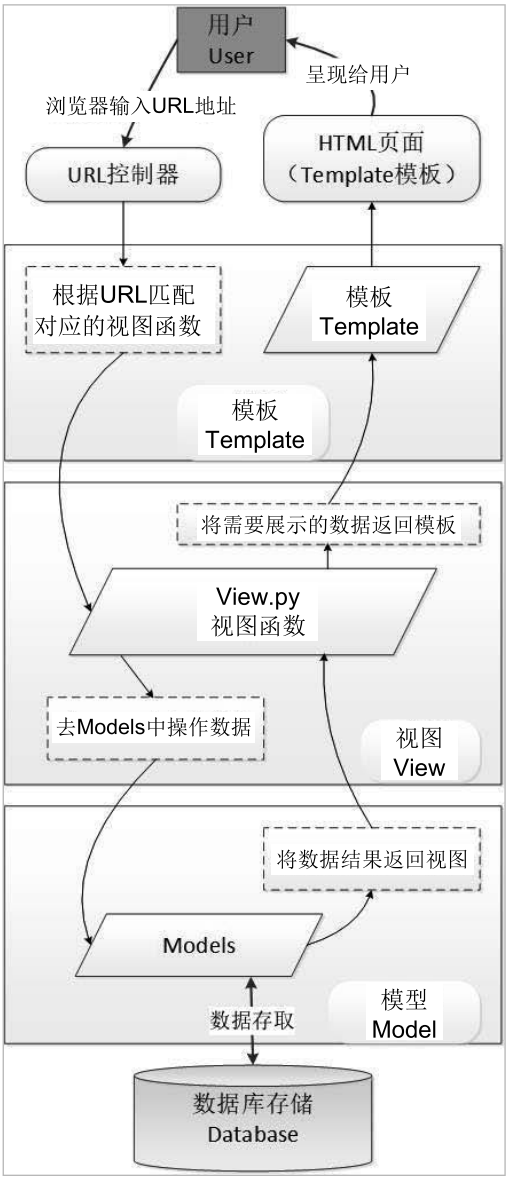

图 1.5　Django 框架用户操作流程制

1.1.5　Django 框架特点

Django框架是基于Python语言及MVC模式设计的优秀Web框架，具有开发快捷、低耦合、部署方便、可重用性高和维护成本低等显著特点。

Django框架通过一个URL分发器模块进行URL分派，该URL分发器模块使用正则表达式来匹配URL，支持设计人员采用自定义URL方式，并且没有框架的特定限定，使用起来非常灵活。

Django框架可以很方便地生成各种表单模型，实现表单的有效性检验，并且支持从自定义的模型实例生成相应的表单。

Django框架具有强大且可扩展的模板语言，支持分隔设计内容和Python代码，并且具有可继承性。

Django框架以Python类的形式定义数据模型，通过ORM（对象关系映射）将模型与关系数据库进行连接，从而让设计人员能够得到一个非常容易使用的数据库API。同时，Django框架也支持直接使用原始SQL语句。

Django框架内置了国际化系统，支持开发多种语言的Web网站。

Django框架内置了一个可视化的自动化管理员界面（Admin Site），其类似于一个CMS系统，设计人员可以方便快捷地通过该界面进行人员管理和更新内容等操作。

1.2　搭建Django框架开发环境

本节将介绍搭建Django框架开发环境的相关内容，包括Python开发环境安装、Django框架安装和开发工具选择等。

另外，为了方便大多数初学者进行更有效的学习，全书的开发环境配置和代码实例均在Windows系统下完成。相信读者在熟练掌握了全书的内容之后，如果打算尝试在Linux系统环境或mac OS系统环境下进行开发，也会很快上手。

1.2.1　安装 Python 语言环境

在安装Django开发环境之前，务必先安装好Python语言解释器，这是因为Django框架是基于Python语言开发的。建议读者安装最新版的Python安装包，这样可以保证最好的兼容性。

首先，判断个人计算机的操作系统环境中是否已经安装了Python语言解释器。判断方法就是在命令行下输入如下命令查看Python版本：

```
python --version
```

假设操作系统环境中并未安装Python语言解释器，那么命令行中一般会输出类似“python不是内部或外部命令，也不是可运行的程序或批处理文件”这样的提示信息。

然后，在Python官方网站（https://www.python.org）上下载最新版的Python安装包（见图1.6），并在本地进行安装。

从图1.6中可以看到，当前最新版的Python安装包版本号为“3.11.2”。另外，Python官方网站会自动识别当前用户所使用的操作系统类型，并提供合适的Python安装包类型，具体如图1.7所示。下载后我们会得到一个名称为“python-3.11.2-amd64.exe”的可执行文件，这个就是Python安装包。其中，“3.11.2”表示版本号，“amd64”表示用户当前操作系统是64位的。

接下来，双击运行python-3.11.2-amd64.exe可执行文件，安装Python语言解释器，同时配置Python语言开发环境。具体安装步骤如下：

步骤01 在安装界面（见图1.8）中，我们可以选择“Install Now”默认安装方式，也可以选择“Customize installation”用户自定义安装方式。如果选择了默认安装方式，那么会将Python语言解释器安装到系统盘（C盘）的用户目录下。笔者这里选择了用户自定义安装方式，这样可以将Python语言解释器安装到指定的路径下。

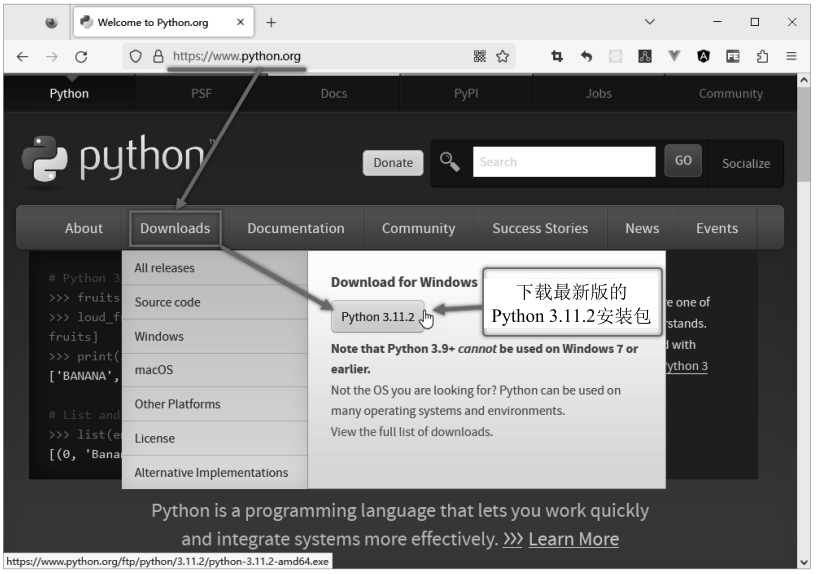

图 1.6　下载最新版 Python 安装包（1）

图 1.7　下载最新版 Python 安装包（2）

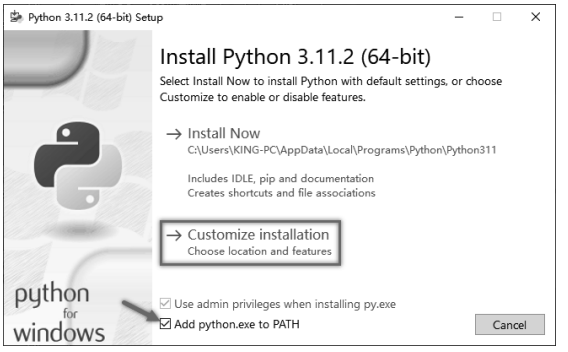

图 1.8　安装最新版 Python 安装包（1）

另外，建议同时勾选“Use admin privileges when installing py.exe”和“Add python.exe to PATH”复选框。其中，第一个复选框会将Python语言解释器指派给全部系统用户使用，第二个多选框则会将Python语言解释器添加到Windows系统的PATH环境变量中去。

步骤 02　单击“Customize installation”菜单进入“Optional Features”（可选特性）界面，我们可以选择界面中的任意项，这里笔者全部勾选了，如图1.9所示。其中，“pip”工具是强烈建议勾选的，它是Python的包安装及管理工具。

步骤 03　选择完毕后，单击Next（下一步）按钮继续安装，为Python指定“Advanced Options”（高级选项），具体如图1.10所示。这里笔者也勾选了全部选项。其中，最后一个选项中所备注的信息（requires VS 2017 or later）表示需要提前安装Visual Studio 2017 + 版本的开发套件。如果读者想尝试该选项，可以去Visual Studio官方网站下载预览版的开发套件（离线版或Web安装版均可），提前进行安装。

图1.10中的箭头所指的地方可以指定用户自定义的安装路径。

步骤 04　用户选择好安装路径后，单击Install（安装）按钮继续安装，具体如图1.11所示。

图1.11中显示的是Python语言解释器的安装进度条。

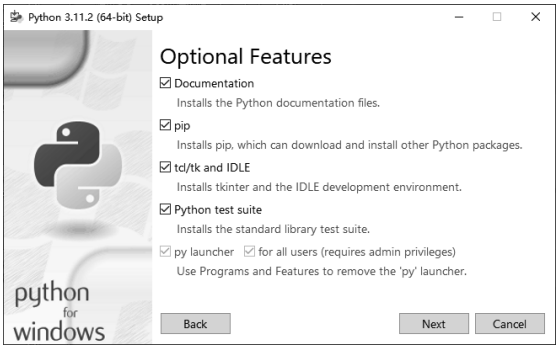

图 1.9　安装最新版 Python 安装包（2）

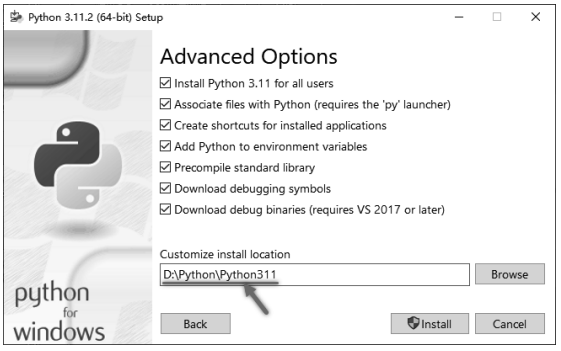

图 1.10　安装最新版 Python 安装包（3）

步骤 05 安装完毕后，界面如图1.12所示，提示“Setup was successful”就表示Python语言解释器安装成功了。

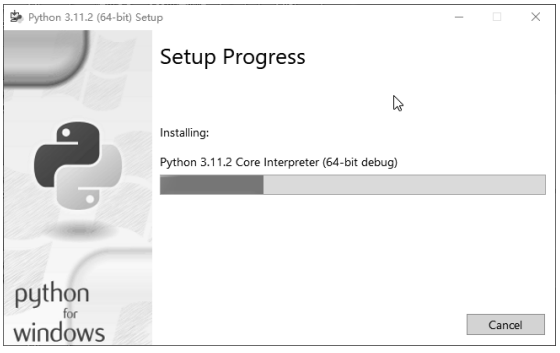

图 1.11　安装最新版 Python 安装包（4）

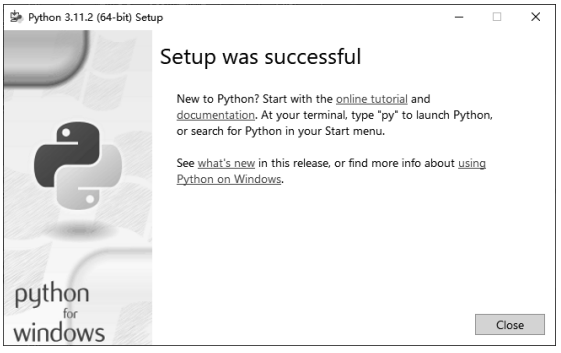

图 1.12　安装最新版 Python 安装包（5）

为了验证Python语言解释器已经在系统中安装成功，我们可以再次在命令行窗口或终端管理员中运行如下命令进行测试：

```
python --version
```

结果如图1.13所示。

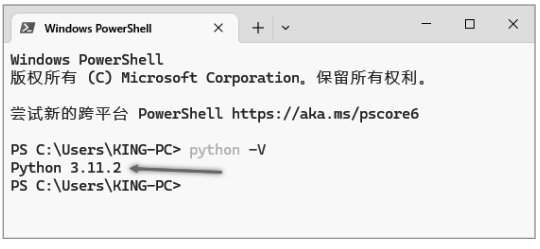

图 1.13　测试 Python 语言环境

图1.13中的命令行提示信息“Python 3.11.2”，表示当前操作系统中已经成功安装了Python 3.11.2版本的语言解释器。

现在Python语言解释器已经安装成功了，那么如何进行编程使用呢？Python提供了一个交互式的命令行开发环境，通过在命令行窗口或终端管理员输入“python”命令就可以进入该开发环境，然后就可以一行一行输入Python代码并实时查看运行结果了。如图1.14所示。

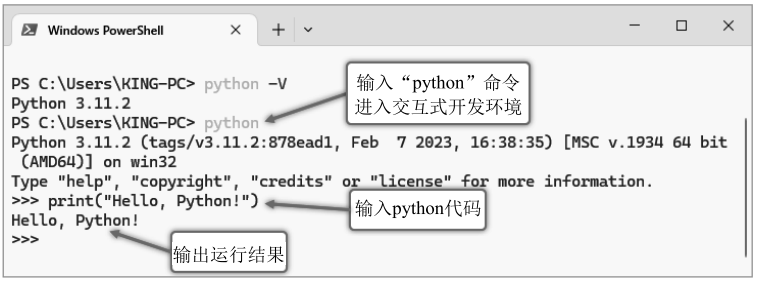

图 1.14　测试 Python 语言开发环境

1.2.2　安装 setuptools 工具

setuptools工具是源自Python Enterprise Application Kit（PEAK）的一个子项目，它是一组用于Python自带的distutilsde工具的增强工具包（32位平台适用于Python 2.3.5以上的版本，而64位平台则适用于Python 2.4以上的版本）。setuptools工具包可以让程序员更方便地创建和发布Python开发包，尤其是那些对于其他包具有依赖性的情况。

其实，Python自带了一个用于发布Python开发包的模块——distutils。那为什么大多数开发人员更青睐使用setuptools工具包来替代distutils模块呢？原因在于setuptools工具包的真正优点不单单是实现了替代distutils模块所设计的功能，更是增强了distutils模块所设计的功能。同时，setuptools工具包还简化了setup.py脚本中的内容。

setuptools工具包在性能方面的优势主要包括：

- setuptools工具包在Python包管理性能方面进行了增强，实现了一种更加透明的方法来查找、下载和安装Python依赖包。
- setuptools工具包可以在一个Python依赖包的多个版本中自由进行切换，这些版本都安装在同一个系统上。
- setuptools工具包支持声明对某个Python依赖包特定版本的需求。
- setuptools工具包可以只使用一个简单的命令就能更新到某个Python依赖包的最新版本。
- 最特别的一点是，即使有些设计人员在开发过程中还未认真考虑某个Python依赖包的setuptools工具兼容性问题，setuptools工具包依然可以使用这些Python依赖包。

总之，setuptools工具包确实是比Python自带的distutils模块要好用得多，这已经是广大设计人员在实践中公认的了。

接下来，我们就具体介绍一下setuptools工具包的安装方法。早期安装setuptools工具包的方法有好几种，例如pip安装方式、ez_setup.py安装方式和setuptools源码编译安装方式等。目前，最新版的setuptools官方网站推荐使用pip安装方式（pip是Python官方的包安装和管理工具，安装最新版的Python就已经默认安装了该工具）。pip安装的操作步骤如下：

步骤 01　打开setuptools工具包的官方网站（https://pypi.org/project/setuptools/），目前setuptools工具的最新版本为“setuptools 67.7.2”，如图1.15所示。

同时，页面中给出了通过 pip 工具安装 setuptools 工具包的命令，具体如下：

```
pip install setuptools
```

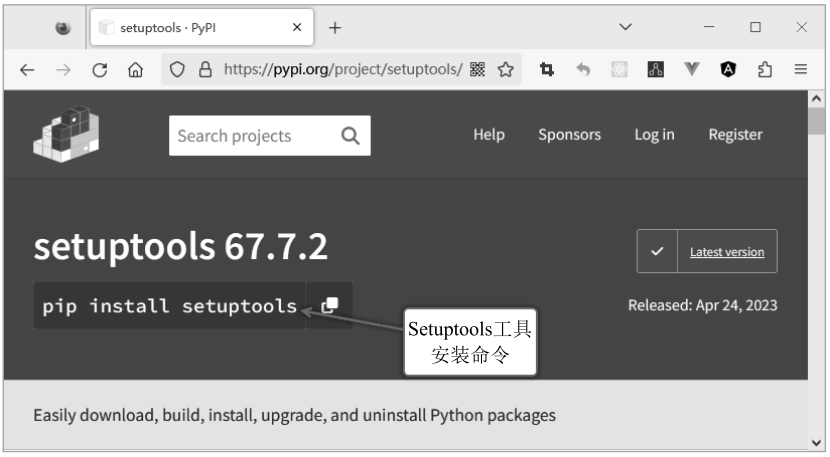

图 1.15　setuptools 工具包官方网站

步骤 02 在使用上述命令行安装 setuptools 工具包之前，先确认 Python 语言环境和 pip 工具已经成功安装，具体方法如图 1.16 所示。图中显示了 Python 和 pip 的版本号，表明 Python 语言环境和 pip 工具已经成功安装。

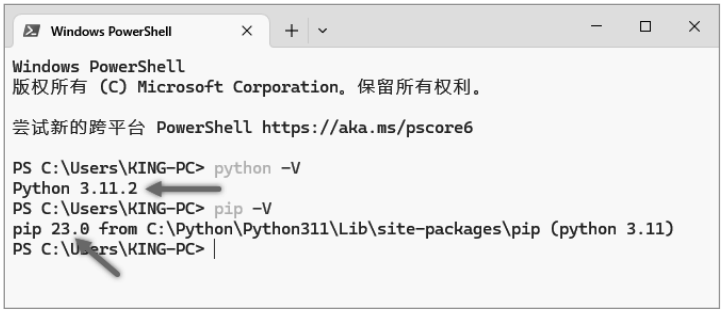

图 1.16　确认 python 语言环境和 pip 工具

步骤 03 在命令行中输入“pip install setuptools”命令来安装setuptools工具包，具体如图1.17所示。

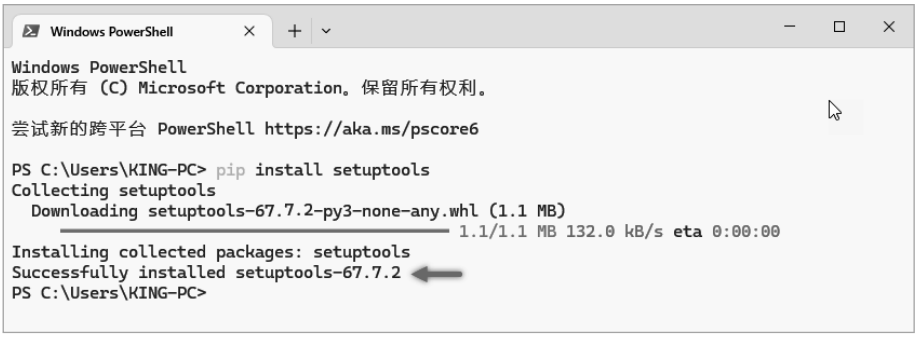

图 1.17　安装 setuptools 工具包

如图1.17中的箭头所示，系统已经成功安装了setuptools-67.7.2版本的setuptools工具包。另外，通过“pip list”命令还可以查看已经安装的Python第三方插件包，具体如图1.18所示。在已安装的Python第三方插件包列表中，我们找到了“setuptools 67.7.2”版本的列表项。

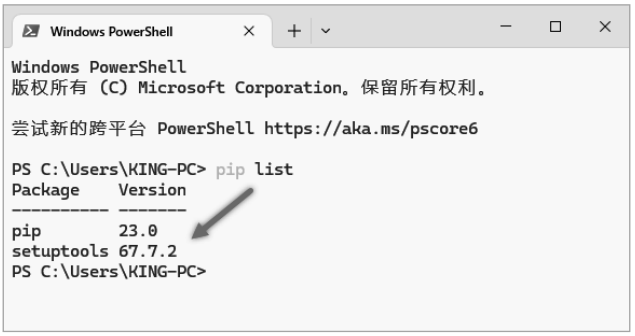

图 1.18　查看 Python 第三方插件包列表

1.2.3　安装 Django 框架

目前，Django框架支持多种安装方式，常见的是Django源码编译安装方式和pip工具安装方式。

1. Django源码编译安装方式

Django源码编译安装方式的具体步骤如下：

步骤 01 使用源码编译安装方式，要先访问Django框架官方网站（https://www.djangoproject.com/download/）下载源码安装包，具体如图1.19所示，单击链接“Django-5.0.1.tar.gz”下载最新版的Django框架源码安装包。

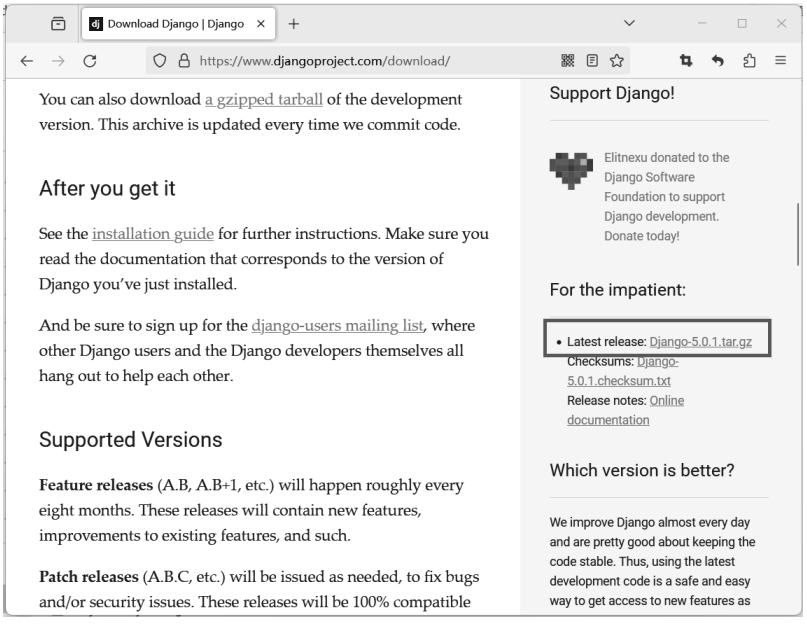

图 1.19　Django 框架官方网站下载源码包

步骤 02 将下载得到的Django安装包（Django-5.0.1.tar.gz）解压到Python安装目录的同一级目录下。

步骤 03 通过命令行窗口进入Django安装包目录c:\python\Django-5.0.1，执行“python setup.py install”命令开始安装。

步骤 04 Django框架安装完毕后，默认会被安装到Python安装目录下Lib子目录下的site-packages子目录中。如图1.20所示，在site-packages子目录中已经存在了Django框架目录。

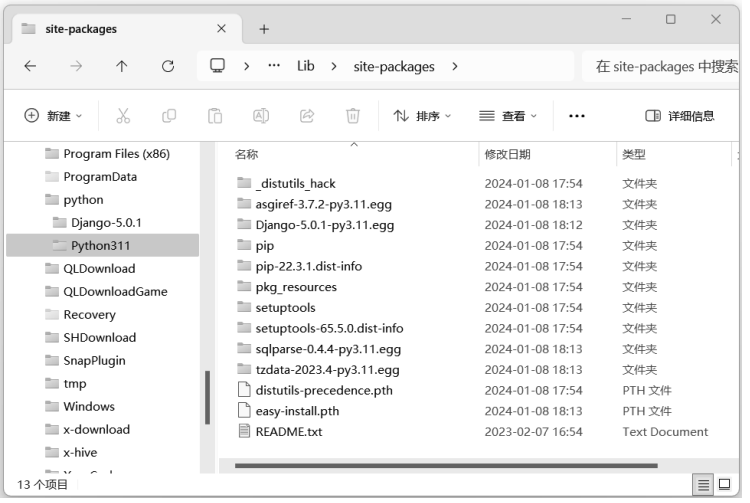

图 1.20　Django 源码安装方式

2. pip工具安装方式

pip工具安装方式是Django框架官方推荐的方式，通过pip工具方式安装Django框架的具体步骤如下：

步骤 01 打开Django框架官方网站（https://docs.djangoproject.com/en/5.0/topics/install/），找到安装命令，如图1.21所示。

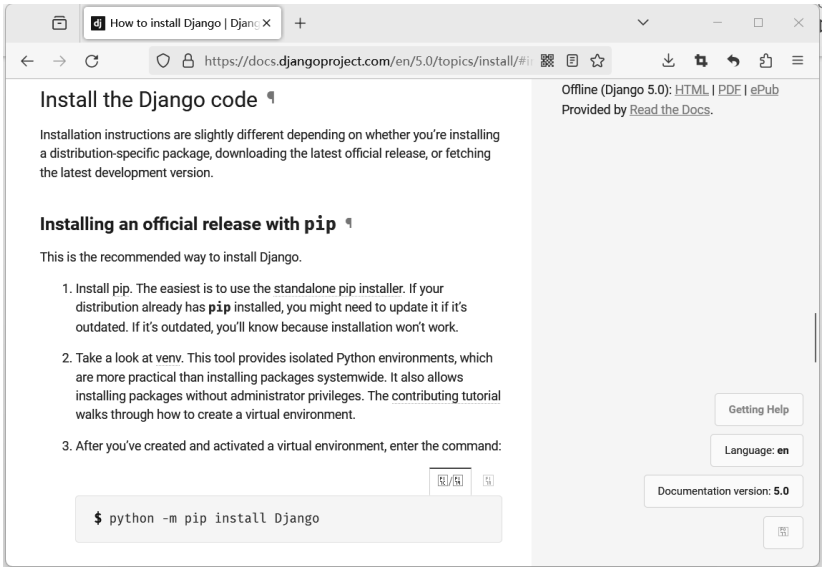

图 1.21　Django 框架官方网站推荐的安装方式

步骤 02 在命令行中输入“python –m pip install Django”命令，就可以自动安装最新版的Django框架了，具体如图1.22所示。从图中的提示信息可以看到，已经成功安装了Django 5.0.1。

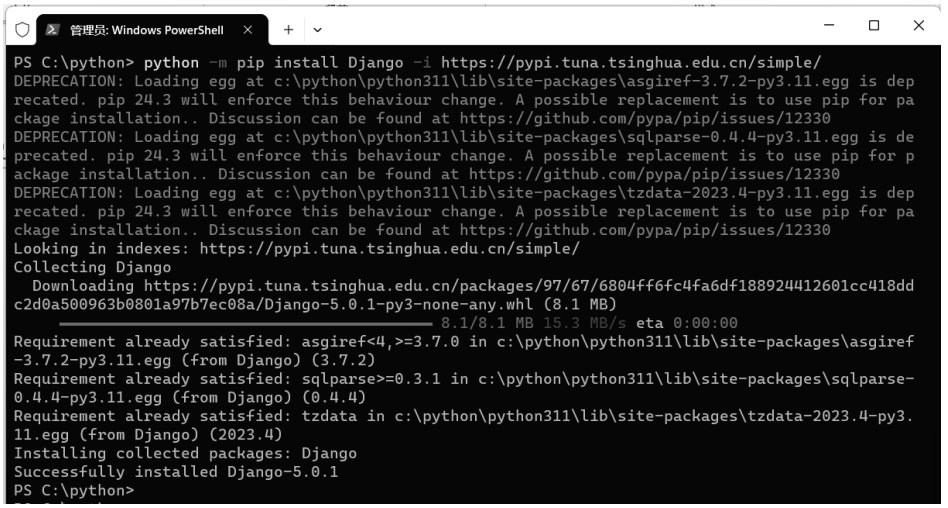

图 1.22　pip 方式安装 Django 框架

另外，如果想安装指定的Django框架版本，需要在上面的命令中加上版本号，具体如下：

```
python -m pip install Django==5.0.1（指定版本号）
```

3. 验证Django框架是否安装成功

在通过上面的方式将Django框架安装完毕后，如何证明Django框架已经安装成功了呢？最简单的方式还是使用“pip list”命令查询Python第三方插件列表，具体如图1.23所示。

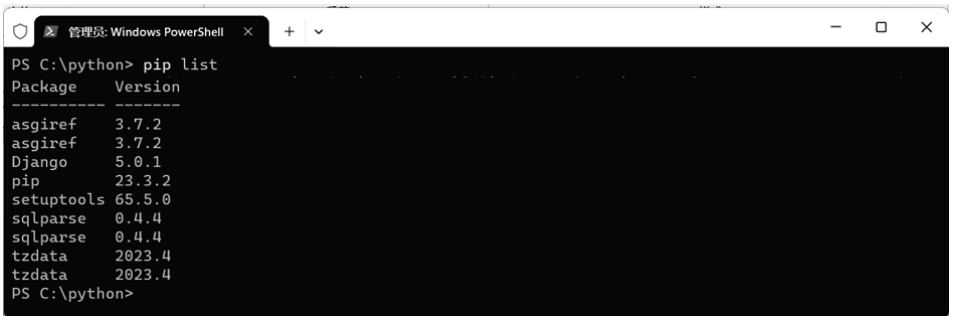

图 1.23　pip list 命令查询 Django

从上图可以看到，通过“pip list”命令查询得到的第三方Python插件列表中，显示出已安装的Django框架版本为5.0.1。

还有一种方法，就是通过Python代码调用Django框架内置的函数 get_version()来查询其版本。具体如图1.24所示，先通过命令行进入Python语言交互环境，然后通过“import django”导入Django框架，再调用 get_version()函数查询已安装的Django框架的版本号，可以看到查询版本结果为5.0.1。

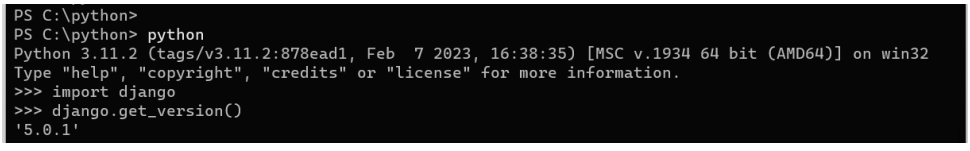

图 1.24　调用 Django 函数方法查询版本

1.3　开发第一个Django框架应用程序

本节将介绍Django框架应用程序开发的大致流程，包括如何通过命令行构建最基本的Django框架应用程序、如何选择Django框架应用程序的开发平台（IDE）和Django框架应用程序基本配置等。

1.3.1　通过命令行构建 Django 应用程序

安装好Django开发环境后，我们就可以通过命令行构建Django应用程序了。通过命令行构建Django应用程序的关键，是使用一个Django框架自带的管理工具——django-admin.py，这是一个Python脚本文件。

那么，这个django-admin.py管理工具在操作系统中的保存路径是什么呢？请读者再查看一下图1.20，它就在django目录下的bin目录中，具体如图1.25所示。

图 1.25　django-admin.py 管理工具

图1.25中的django-admin.py脚本文件表示的就是Django框架管理工具。默认情况下，通过pip工具自动安装Django框架管理工具时，django-admin已经被添加到系统的环境变量PATH中了。

下面开始通过命令行构建Django应用程序，具体操作步骤如下：

步骤 01　通过 django-admin 管理工具在命令行创建 Django 应用程序，命令如下：

```
django-admin startproject ProjectName
```

其中，参数 startproject 是 django-admin.py 工具自带的命令，用于创建用户自定义项目；参数 ProjectName 是用户自定义项目名称。通过在命令行运行上述命令创建 Django 应用程序，效果如图 1.26 所示。

在图1.26中可以看到，目录中已经有了通过django-admin命令新创建的Django项目（HelloDjango）。

图 1.26　通过 django-admin 命令创建 Django 应用程序

步骤 02 通过命令行进入该项目并查看目录下的文件，具体如图1.27所示。

图 1.27　通过 DOS 命令 tree 查看新创建的 Django 项目

在图1.27中，通过DOS命令 tree查看到了新创建的Django项目HelloDjango的文件清单。下面，我们具体介绍一下这些项目文件的作用。

- manage.py：一个Django命令行工具，可让设计人员以各种方式与Django项目进行交互。
- HelloDjango：Django项目容器。
- HelloDjango/asgi.py：一个ASGI兼容的Web服务器的入口，方便运行Django项目。
- HelloDjango/settings.py：Django项目的配置文件。
- HelloDjango/urls.py：定义了Django项目的URL声明，一份由Django驱动的网站目录。
- HelloDjango/wsgi.py：定义了一个WSGI兼容的Web服务器的入口，支持运行Django项目。
- HelloDjango/__init__.py：一个Python空文件，通知Python解析器当前目录是一个Python包。

步骤 03 通过命令行窗口进入HelloDjango项目的根目录，输入以下命令来启动Web开发服务器：

```
python manage.py runserver 0.0.0.0:8000
```

其中，“0.0.0.0”表示支持其他终端连接到开发服务器；“8000”（默认端口号）为开发服务器的端口号，如果省略则表示端口号为“8000”。另外，上述命令可以使用下面的简写方式：

```
python manage.py runserver
```

进入 Django 项目的根目录，运行上述简写命令，Django 框架会以 127.0.0.1:8000（ip:port）这个默认配置启动开发服务器。命令行的运行效果如图 1.28 所示，命令行日志信息表示 Django 开发服务器已经在“http://127.0.0.1:8000”启动了。

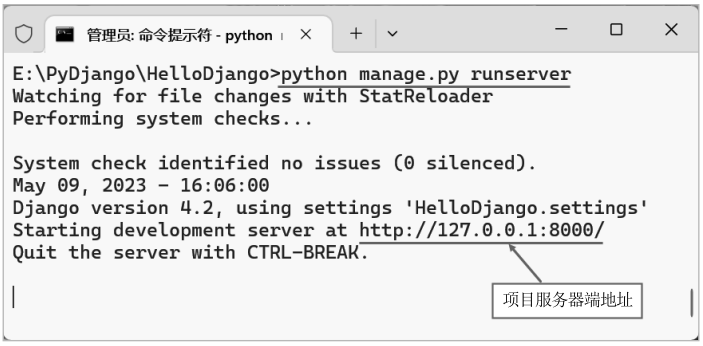

图 1.28　启动 Django 开发服务器

步骤 04 打开浏览器（FireFox）并输入日志信息中的服务器地址及端口号（http://127.0.0.1:8000）。如果浏览器页面效果如图1.29所示，就说明Django应用程序已经成功运行了。

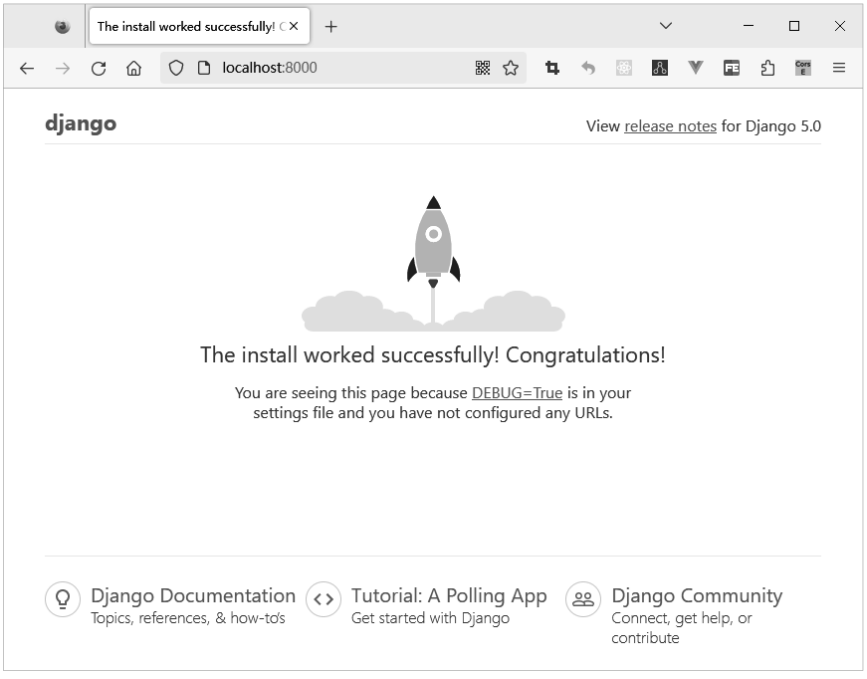

图 1.29　测试 Django 应用程序

1.3.2　通过 PyCharm 平台开发 Django 应用程序

学会使用命令行工具开发Django应用程序是基础，不过更多的时候还是要借助平台开发工具。目前，最好的Django应用程序开发工具就是jetBrains公司推出的PyCharm平台了。

借助PyCharm开发平台，可以极大提高开发Django应用程序的效率，同时可以使用到很多非常

实用的第三方插件。不过读者也要清楚，PyCharm开发平台所实现的功能，在底层也是借助Django命令行工具完成的。

PyCharm开发平台有专业版（Professional）和社区版（Community）两个版本。专业版开发平台是要付费的，不过有30天的试用时间，并且提供了对Django的支持；社区版是免费的，但是不支持Django。我们需要使用专业版PyCharm平台开发Django应用程序，具体操作步骤如下：

步骤 01 打开PyCharm Pro专业版开发平台，通过文件菜单（File）的新建工程（New Project）选项创建Django项目，如图1.30所示。

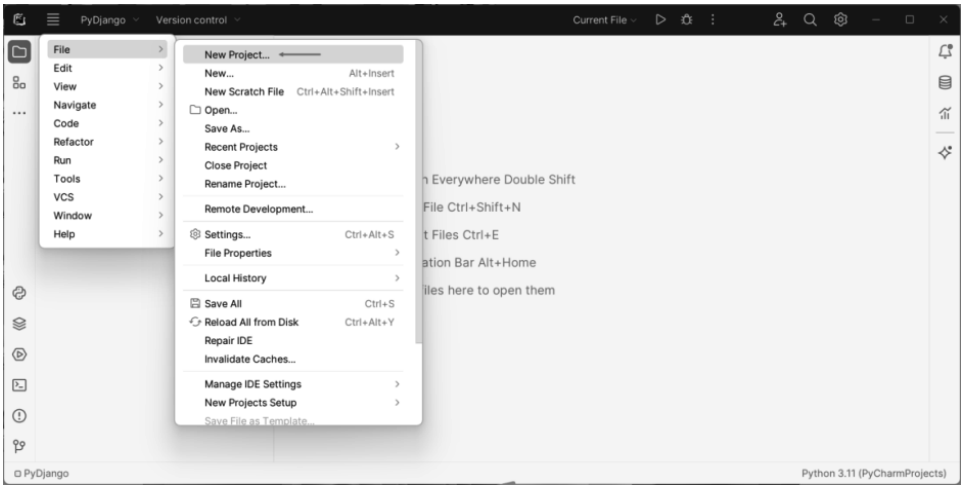

图 1.30　通过 PyCharm 开发 Django 应用程序（1）

步骤 02 在New Project界面左侧选择Django，在Location中设置好Django项目路径，如图1.31所示；打开More Settings，设置Template language为Django，Template folder为templates，Application name为HelloDjango，如图1.32所示。

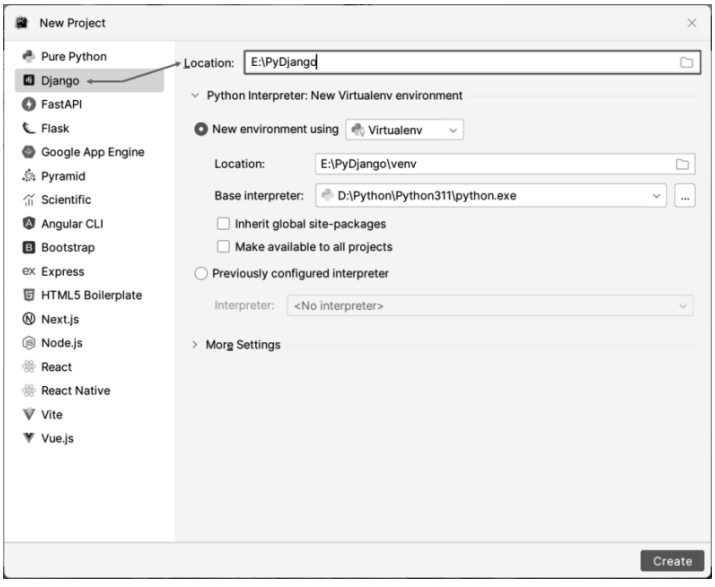

图 1.31　通过 PyCharm 开发 Django 应用程序（2）

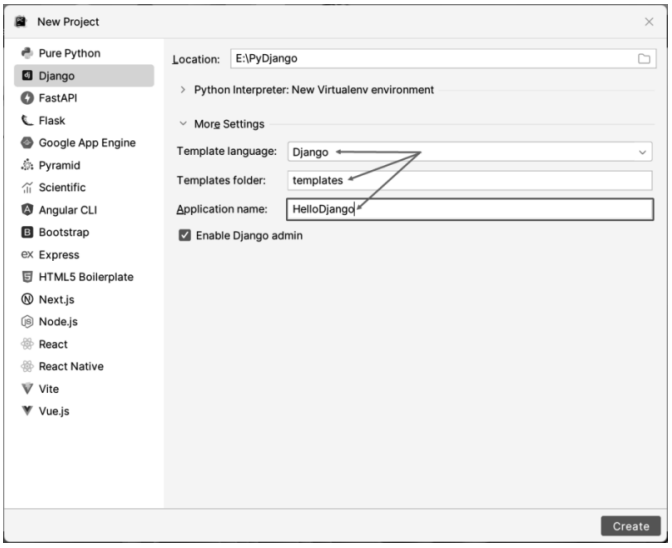

图 1.32　通过 PyCharm 开发 Django 应用程序（3）

步骤 03 单击Create按钮创建项目，在弹出的对话框中单击“This Window”按钮，如图1.33所示。

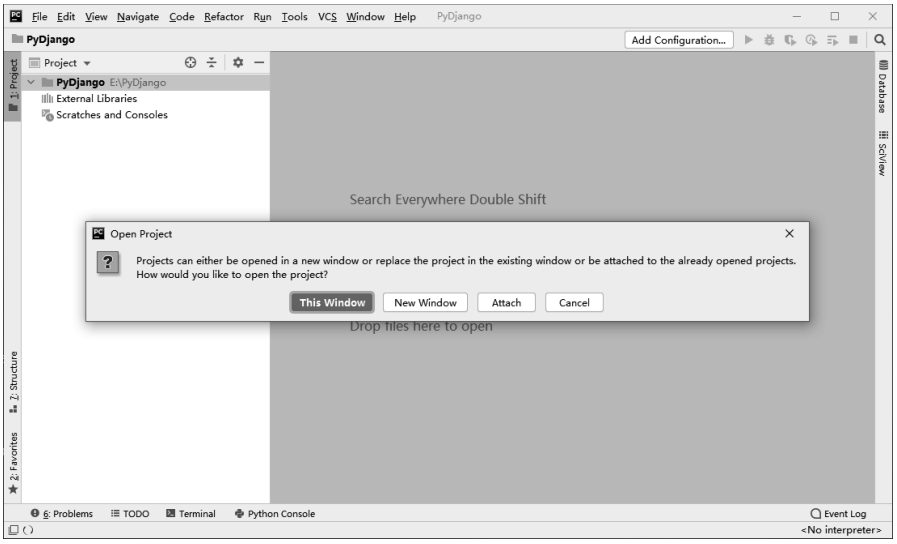

图 1.33　通过 PyCharm 开发 Django 应用程序（4）

步骤 04 耐心等待一会儿，PyCharm平台会为设计人员自动创建好Django应用程序框架和文件，如图1.34所示。

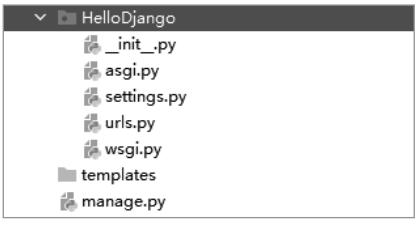

图 1.34　通过 PyCharm 开发 Django 应用程序（5）

由上图可知，HelloDjango应用程序中的文件与之前通过命令行工具（django-admin.py）创建的结果是一致的。

1.3.3 添加简单的 Django 应用程序代码

现在我们已经拥有了一个完整的Django应用程序框架和文件，下面就在此基础上添加一些简单的Django代码，体验一下Django应用程序的具体开发过程。

步骤01 添加视图页面。

在HelloDjango项目下的HelloDjango目录中新建一个views.py视图文件，并输入如下代码：

【代码1-1】

```
01  from django.http import HttpResponse
02  def sayHello(request):
03      return HttpResponse("Hello Django!")
```

【代码分析】

- 在第01行代码通过调用django.http模块导入了HttpResponse对象（请求与响应）。
- 在第02、03行代码定义了一个Python函数（sayHello）。
- 第03行代码通过调用HttpResponse对象返回一行文本信息。

步骤02 配置URL路由。

打开HelloDjango项目下的HelloDjango目录中的urls.py路由文件，添加如下代码绑定URL路由与视图页面：

【代码1-2】

```
01  from django.contrib import admin
02  from django.urls import path
03  from . import views
04
05  urlpatterns = [
06      path('admin/', admin.site.urls),
07      path('hello/', views.sayHello)
08  ]
```

【代码分析】

- 第03行代码导入了【代码1-1】定义的视图页面（views.py）。
- 第07行代码通过调用路由方法（path()）将视图页面（views.py）匹配到路由路径（/hello/）。

步骤03 测试视图页面。

首先通过PyCharm平台启动Web开发服务器，具体效果如图1.35所示，日志信息提示开发服务器（http://127.0.0.1:8000/）已经成功启动了。

然后，打开浏览器（FireFox）并输入地址http://127.0.0.1:8000/hello，页面效果如图1.36所示。

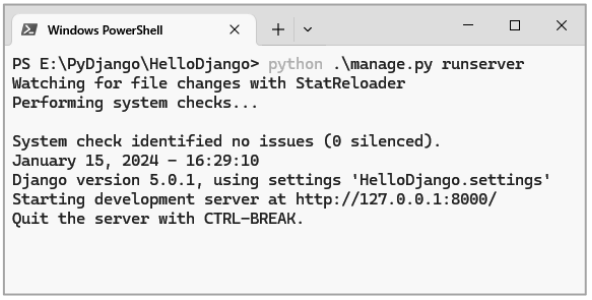

图 1.35　测试 Django 应用程序视图页面（1）

图 1.36　测试 Django 应用程序视图页面（2）

页面中成功显示了【代码1-1】中所定义的文本信息，说明视图页面（views.py）通过URL路由（urls.py）已经被成功解析了！

1.4　本 章 小 结

本章详细介绍了Django框架应用程序开发的基础知识，具体包括Django框架的基础知识、如何搭建Django框架应用程序的开发环境、如何定义Django应用程序基础配置的信息等方面。另外，还通过开发一个具体的Django应用程序，进一步帮助读者理解使用Django框架开发应用程序的流程。