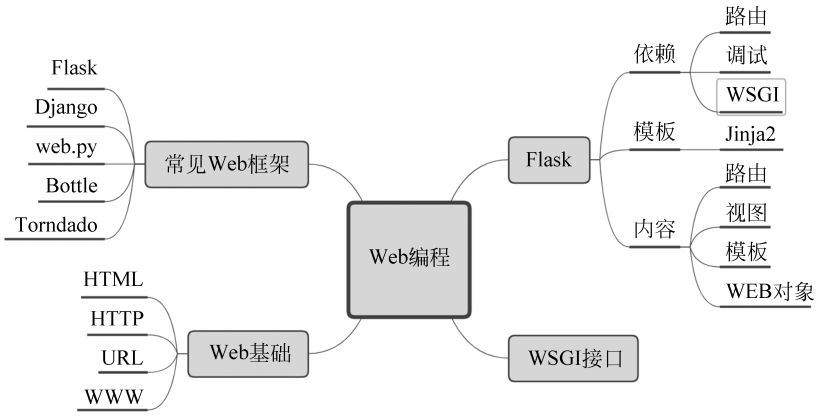
第 5 章

Web 编程

5.1 导学



学习目标：

- 了解 Web 编程基本概念、技术、理论。
- 理解 HTML 语言以及在 Web 编程中的作用。
- 了解常见的 Python 语言 Web 框架。
- 掌握 Flask 框架的下载、安装和使用。

在计算机应用的历史中，早期程序都是桌面应用程序，只能运行在价格昂贵的大型机器上。随着个人计算机的逐步普及，应用程序开始安装到普通机器上，而数据库需要运行在服务器上，这样多个用户就可以共享数据服务，这就是 C/S(Client/Server)架构的由来。自从互联网兴起，越来越多的应用转为基于 Web 的 B/S(Browser/Server)架构程序，主要

原因有以下 4 点。

(1) B/S 架构分布性强,个人计算机上不需要安装客户端,只需要安装浏览器即可,而浏览器通常由操作系统自带,非常方便。

(2) 维护方便,应用程序的升级维护只需要在服务端进行,对客户端不会带来任何影响。

(3) 业务扩展简单方便,通过增加网页即可增加服务器功能。

(4) 开发简单,共享性强。

所以,到目前为止,除了少数对性能要求较高的大型应用之外,大部分软件都是基于 Web 的。Web 开发也经历了以下 4 个阶段。

(1) 静态 Web 页面,通过文本编辑器直接编辑并生成静态的 HTML 页面,这些页面在应用使用过程中不会发生变化,如果要修改,需要再次编辑 HTML 内容。

(2) CGI 技术,静态 Web 页面无法处理用户动态提交的数据,出现了 CGI(Common Gateway Interface)技术,使用 C/C++ 实现,这个问题得以解决。

(3) ASP/JSP/PHP 技术,由于 Web 应用修改频繁,用 C/C++ 这样的语言进行 Web 开发非常麻烦,不如脚本语言开发效率高。因此市场上出现了基于脚本语言的开发技术,常见的有 ASP、JSP 以及 PHP 等。

(4) MVC 模式,为进一步解决直接用脚本语言嵌入 HTML 导致的可维护性差的问题,人们引入了 MVC(Model-View-Controller)模式。现在很多 Web 开发框架都基于 MVC 或者类似的模式。

Python 是一种解释型的脚本语言,开发效率高,对于 Web 编程也是一个非常好的选择。它提供了开发 Web 应用程序的卓越工具和成熟的编程框架。Python 有上百种 Web 开发框架和很多成熟的模板技术,选择 Python 开发 Web 应用,不但开发效率高,而且运行速度快。本章首先介绍了 Web 编程必须掌握的基本概念,供初学者了解基本原理。然后介绍 Flask 编程框架,介绍它的模板、表单、cookies、文件上传和部署等。掌握了这些知识之后,读者就可以开发一些简单的 Web 应用了。

5.2 Web 基础

学习 Web 应用开发,读者首先需要掌握以下概念。

5.2.1 Web

人们经常提到 Web 应用开发,但很多初学者搞不清 Web 到底是什么意思。其实,要清楚 Web 到底是什么,涉及其他几个容易混淆的概念。

计算机网络(Computer Network)概念外延最大,它可以是局域网络,也可以是很多计算机连接而成的互联网。由计算机相互连接而成的大型网络系统都可算是互联网,Internet 是互联网中最大的一个,即因特网。因特网是由许多小的网络互联而成的一个逻辑网,每个小的网络中还连接着若干台计算机(主机),基于通信协议(例如 TCP/IP)和通信子网(例如网络设备和电缆)实现网络通信。因为因特网是互联网中最大的一个,所以有时也将其称为互联网。

而Web（world wide web），即全球广域网，也称为万维网，是一种基于超文本（Hypertext）和HTTP的、全球性的、动态交互的、跨平台的分布式图形信息系统；是建立在因特网上的一种网络服务，为浏览者在因特网上查找和浏览信息提供了图形化的、易于访问的直观界面，其中的文档及超链接（Hyperlink）将因特网上的信息结点组织成一个互为关联的网状结构。Web页面使用前端编程语言和技术来实现，主要有HTML、CSS和JavaScript。Web只是因特网上的一个应用层服务。除了Web外，QQ、E-mail等也是因特网上其他类型的应用层服务。

在因特网上可以找到许多联网的计算机，通过Web可以使用各种共享资源和应用，例如下载文件、听音乐、看视频等。所以因特网是Web的交流活动基础，而Web使得因特网共享信息更为便捷。

5.2.2　HTML

HTML（HyperText Markup Language）的全称为超文本标记语言，预定义了一系列标签。通过这些标签可以统一文档格式，使分散的因特网资源连接为一个逻辑整体。HTML文本是由HTML命令组成的描述性文本，可以用于说明文字、图形、动画、声音、表格、链接等。

超文本是一种组织信息的方式，它通过超链接将文本中的文字、图表与其他信息媒体相关联。这些相互关联的信息媒体可能在同一文本中，也可能是其他文件，或是地理位置相距遥远的某台计算机上的文件。这种组织信息方式将分布在不同位置的信息资源用随机方式进行连接，为人们检索信息提供方便。

HTML是用来标记Web信息如何展示以及其他特性的一种语法规则，最初于1989年由CERN的Tim Berners-Lee发明。HTML基于早期的语言SGML定义，并简化了其中的语言元素，这些元素告诉浏览器如何在用户的屏幕上展示数据。HTML历史上有较多版本，目前流行的版本是HTML5。

下面展示一个简单的HTML页面的示例，如例5-1所示。

【例5-1】　使用HTML示例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>HTML示例</title>
</head>
<body>
    <h1>这是一个标题。</h1>
    <p>这是一个段落。</p>
    <p>这是另一个段落。</p>
</body>
</html>
```

上面的例子中，DOCTYPE用来声明文档标准类型，告诉浏览器这是基于HTML5的

Web 页面，<html>与</html>是一组标签，用来描述文档基于 HTML。标签里面可以嵌入其他标签。这里标签<head>与</head>描述的是网页的相关描述信息，例如网页的标题等，而<body>与</body>标签之间放的主要是可视化网页内容，这些内容就是要呈现给网页用户的。其中，标签<h1>与</h1>之间的是一个标题，而 与 之间是文章的一个段落。一个普通页面通常是由若干个标题和段落以及其他内容组成。

读者可以使用文本编辑器将例 5-1 中的代码编辑好，保存为以后缀为.html 的文件，然后使用任意浏览器打开就可以看到页面内容，如图 5-1 所示。

图 5-1　使用浏览器打开 HTML 页面

HTML 中有一个可向服务端提交数据的重要标签——Form（表单）。表单中包含不同类型的 input 元素、单选按钮、复选按钮和提交按钮等。下面是一个简单的登录页面，如例 5-2 所示。

【例 5-2】　登录页面示例

```
<!DOCTYPE html>
<htmllang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<form action="" method="post">
    用户名:<br>
    <input type="text" name="username" placeholder="输入用户名">
    <br>
    密码:<br>
    <input type="password" name="password" placeholder="输入密码">
    <br><br>
    <input type="submit" value="登录">
    <input type="reset" value="重置">
</form>
```

```
</body>
</html>
```

上例中有一个表单，表单中的第一个属性 action 表示服务端的哪个对象来处理用户的数据，method 表示使用哪种 HTTP 方式提交，常见的是 post 和 get 两种方式。表单中有两个 input 标签，表示两个输入框。输入框还有多个子类型，使用 type 属性来区分。输入框的 name 属性不能缺少。最后有一个“登录”按钮，单击这个按钮后，用户输入的信息就会提交到服务端，而单击“重置”按钮，用户输入的信息会被清空。运行效果如图 5-2 所示。

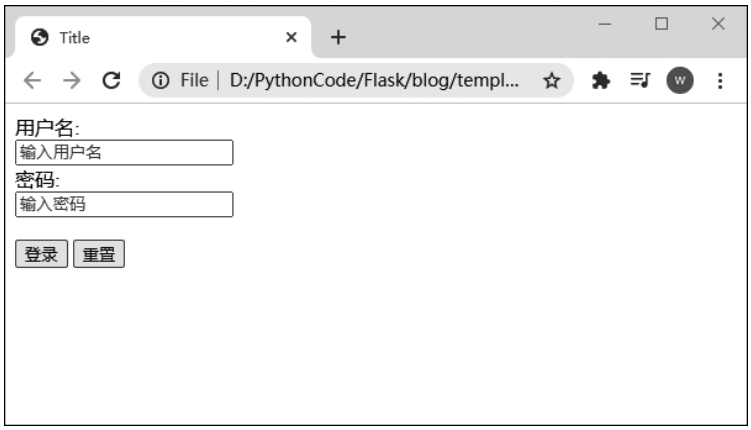


图 5-2　登录页面

5.2.3　URL

5.2.2 节中编写的静态 HTML 网页可以使用浏览器打开，浏览器的上方是一个用来输入 URL 的地址栏。通常可以使用该地址栏输入一个网址，来定位一个万维网上面的资源。

所谓 URL(Uniform Resource Locator)就是统一资源定位器，用来访问存在的资源，由如下几部分组成。

(1) 传送协议，常见的协议有 HTTP、FTP、HTTPS 等。

(2) 层级 URL 标记符号(为[//]，固定不变)。

(3) 访问资源需要的凭证信息(可省略)。

(4) 服务器(通常为域名，也可以使用 IP 地址)。

(5) 端口号(以数字方式表示，若为 HTTP 协议，默认端口“:80”可省略)。

(6) 路径(以“/”字符区别路径中的每一个目录名称)。

(7) 查询(GET 模式的窗体参数，以“?”字符为起点，每个参数以“&”隔开，再以“=”分开参数名称与数据，通常以 UTF8 的 URL 编码，避开字符冲突的问题)。

(8) 片段，以“#”字符为起点。

以 http://www.nuist.edu.cn:80/index.html? page=1#firstSection 为例，其中 http 是传输的协议，5.2.4 节再重点介绍；www.nuist.edu.cn 是域名；冒号后面的 80 是服务器上的默认网络端口号；后面的/news/index.html 是路径；“? page=1”是该请求附带的参数。最后的#firstSection 是该页面上面的某个区域或者片段。这里需要注意的是，大多数网页

浏览器不要求用户输入网页中“http://”的部分，因为绝大多数网页内容是超文本传输协议文件。同样，80 是 HTTP 的默认端口号，一般也不必写明。

5.2.4 HTTP

前面提到的 HTTP(HyperText Transfer Protocol)称为超文本传输协议，它是一种用于分布式、协作式和超媒体信息系统的应用层协议。

HTTP 是一个客户端和服务器端请求和应答的标准。通过使用网页浏览器，客户端发起一个 HTTP 请求到服务器上某个端口。应答的服务器上存储着一些资源，例如 HTML 文件和图像。

HTTP 中有若干请求类型，最常用的是两种请求方式：GET 与 POST 请求。简单来说，这两种方式的区别就是，GET 提交的数据会放在 URL 之后，也就是请求行里面，以“?”分割 URL 和传输数据，参数之间以“&”相连，上面的 URL 示例使用的就是 GET 请求。而 POST 方法是把提交的数据放在 HTTP 包的请求体中。GET 提交的数据大小有限制，而 POST 方法提交的数据没有限制。

所有 HTTP 响应的第一行都是状态行，依次包括当前 HTTP 版本号，3 位数字组成的状态代码，以及描述状态的短语，彼此之间由空格分隔。状态代码的第一个数字代表当前响应的类型：

(1) 1xx 表示消息，请求已被服务器接收，继续处理；

(2) 2xx 表示成功，请求已成功被服务器接收、理解、接受；

(3) 3xx 表示重定向，需要后续操作才能完成这一请求；

(4) 4xx 表示请求错误，请求含有词法错误或者无法被执行；

(5) 5xx 表示服务器错误，服务器在处理某个正确请求时发生错误。

最常见的状态码 404，表示资源不存在，通常是用户请求的 URL 错误所致。另一个就是 200，表示请求成功。图 5-3 展示了浏览器中的请求成功情况，其中的状态码都是 200。

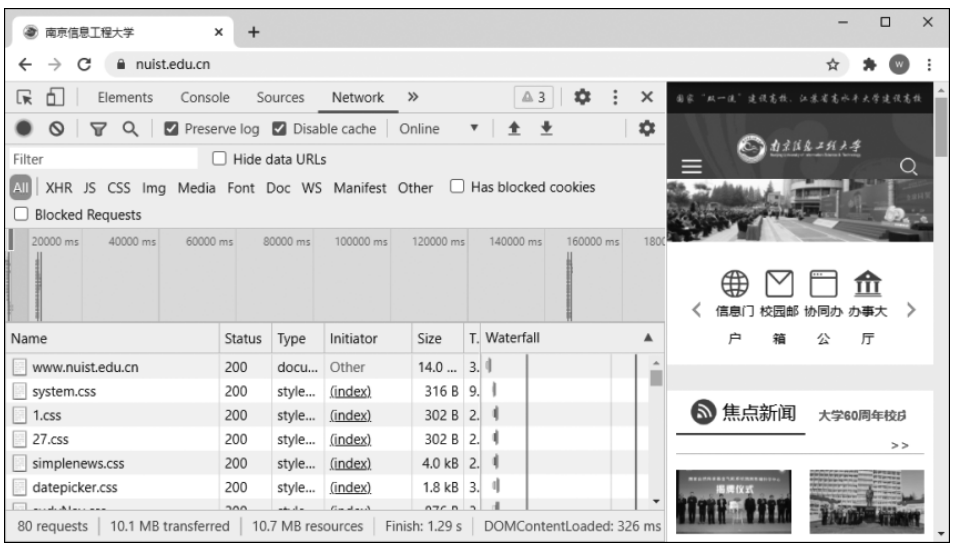


图 5-3 状态码

5.3 WSGI 接口

5.3.1 WSGI 接口简介

WSGI，全称 Web Server Gateway Interface，或者 Python Web Server Gateway Interface，是为 Python 语言定义的 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口。

WSGI 是一个网关，负责在协议之间进行转换。它作为 Web 服务器与 Web 应用程序或应用框架之间的一种低级别接口，是基于现存的 CGI 标准而设计的。

很多框架都自带了 WSGI server，例如 Flask，web.py，Django、CherryPy 等，性能不高，自带的 Web server 主要用于测试。正式发布时多使用生产环境的 WSGI server 或者 nginx＋uwsgi。

简而言之，WSGI 就像是一座桥梁，一边连着 Web 服务器，另一边连着用户的 application 应用。但是，这个桥的功能很弱，有时候还需要别的桥来帮忙才能进行处理。

WSGI 有两方：服务器或网关一方，以及应用程序或应用框架一方。服务方调用应用方，提供环境信息，以及一个回调函数（提供给应用程序用来将消息头传递给服务器方），并接收 Web 内容作为返回值。

所谓的 WSGI 中间件同时实现了 API 的两方，因此可以在 WSGI 服务和 WSGI 应用之间起调解作用：从 WSGI 服务器的角度来说，中间件扮演应用程序；而从应用程序的角度来说，中间件扮演服务器。

“中间件”组件可以执行以下功能：

（1）重写环境变量后，根据目标 URL，将请求消息路由到不同的应用对象；

（2）允许在一个进程中同时运行多个应用程序或应用框架；

（3）负载均衡和远程处理，通过在网络上转发请求和响应消息；

（4）进行内容后处理。

5.3.2 WSGI 接口示例

下面是一个简单的接口示例，如例 5-3 所示。

【例 5-3】 WSGI 应用示例

```
fromwsgiref.simple_server import make_server

defrun_server(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/html;charset=urf-8')])
    return [b'<h1>Hello,world!</h1>',]

def start():
    httpd =make_server('127.0.0.1', 9900, run_server)  # 启动监听服务，端口 9900
    print('listening 9900.....')
```

```
    httpd.serve_forever()

if __name__ =='__main__':
    start()
```

上例中，start()函数里面定义了一个服务器对象 httpd，该服务器基于 HTTP，127.0.0.1 表示本机 IP 地址，9900 就是监听端口，run_server 是一个处理函数。Serve_forever 方法表示正式启动服务器，此时服务器一直处于被动监听状态。在 run_server 中有两个参数，其中 environ 表示包含所有 HTTP 请求信息的 dict 对象，而 start_response 表示发送 HTTP 响应的函数。start_response 发送了 HTTP 响应头部，第一个参数是 HTTP 状态码，第二个参数是 HTTP 头部信息，然后函数返回值作为 HTTP 响应的 Body 发送给浏览器。

接下来观察结果如何，首先需要将服务器启动起来（本例中服务端和客户端都在一台机器上），运行上面的程序，当看到下面的运行结果表示运行成功：

```
listening 9900.....
```

此时，服务器已经处于监听状态，下面打开浏览器，在地址栏中输入 URL，如图 5-4 所示。

图 5-4　访问首页

如果能看到图 5-4 中的信息，说明成功了，此时如果查看源代码，就会发现网页的内容就是从服务端写回的内容，如图 5-5 所示。

上面的代码只能作为初级演示用，实际上的 Web 应用远比其复杂，而复杂的 Web 应用程序使用 WSGI 接口处理还是很麻烦的，在 WSGI 基础之上，一些用来简化 Web 开发的 Web 框架相继出现。

目前主流的 Python Web 框架主要有以下几种。

（1）Flask：热门轻量级 Web 框架。

（2）Django：全能型 Web 框架。

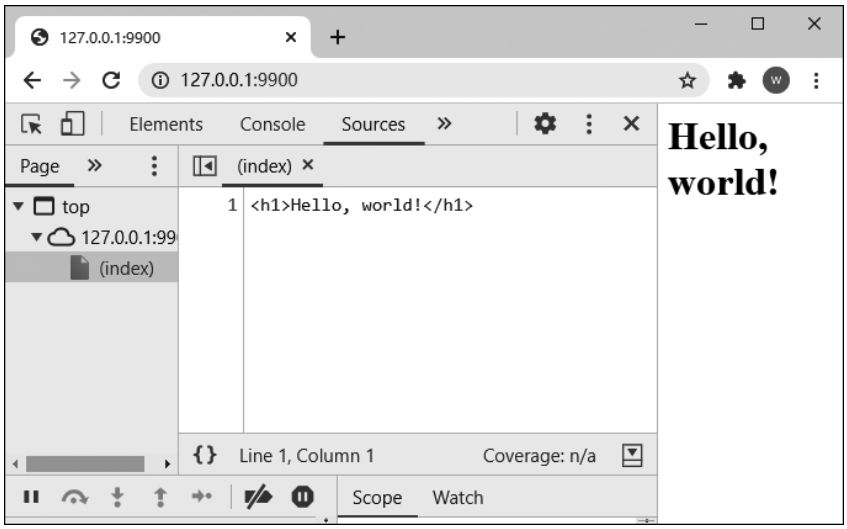


图 5-5　网页内容

(3) web.py：一个小巧的 Web 框架。

(4) Bottle：和 Flask 类似的 Web 框架。

(5) Tornado：Facebook 的开源异步 Web 框架。

其中，Flask 框架相对比较简单，同时也常用于中小型 Web 系统的开发，下面对其做简单介绍。

5.4 Flask 框架

5.4.1　Flask 框架简介

Flask 是一个用 Python 编写的 Web 应用程序框架，由 Armin Ronacher 开发，他领导一个名为 Pocco 的国际 Python 爱好者团队。Flask 基于 Werkzeug WSGI 工具包和 Jinja2 模板引擎，两者都属于 Pocco 项目。Werkzeug 是一个 WSGI 工具包，它实现了请求、响应对象和实用函数，可以在其上构建 Web 框架。Jinja2 是 Python 的一个流行的模板引擎。Web 模板系统将模板与特定数据源组合以呈现动态网页。

Flask 通常被称为微框架，它旨在保持应用程序的核心简单且可扩展。默认情况下，Flask 不包含数据库抽象层、表单验证，或是其他任何已有的库可以处理的功能。但是，Flask 可以通过为应用添加这些功能，如同是 Flask 原生功能一样。众多的扩展提供了数据库集成、表单验证、上传处理、各种各样的开放验证等功能。

总体来说，Flask 具有如下特点：

(1) 良好的文档；

(2) 丰富的插件；

(3) 包含开发服务器和调试器；

(4) 支持单元测试；

(5) RESTful 请求调度;
(6) 支持安全 cookies;
(7) 基于 Unicode 编码。

5.4.2 安装 Flask

安装 Flask 非常简单,进入 cmd 窗口,然后输入"pip3 install flask"就可以开始安装了,如图 5-6 所示。

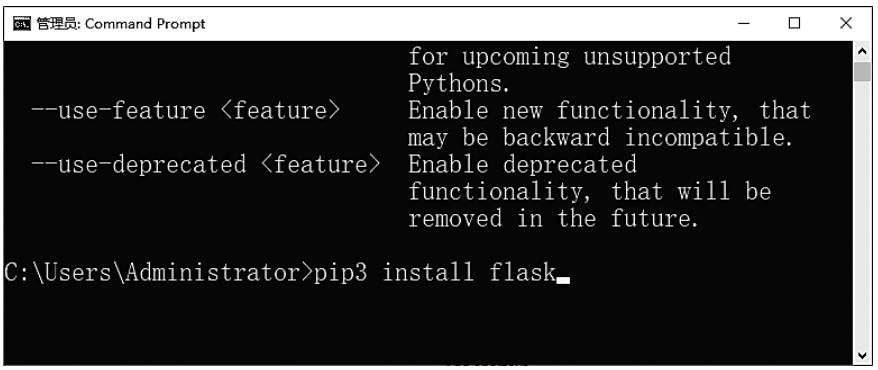


图 5-6 安装 Flask

5.4.3 简单 Flask 应用

接下来使用 Flask 开发一个最简单的 Web 程序,代码如例 5-4 所示。

【例 5-4】 Flask 示例

```
from flask import Flask
app =Flask(__name__)

@app.route('/')
defhello_world():
    return 'Hello World'

if __name__ =='__main__':
    app.run()
```

这里首先导入了 Flask 模块,然后使用 Flask 构造方法创建了一个 app 对象,该对象实际上就是 WSGI 应用程序。"@app.route('/')"是一个装饰器,给出了调用的 URL。当用户的前端请求符合该 URL 规则,就调用 hello_world()函数,然后返回该函数的信息到客户端页面。Hello_world 函数也称为视图函数,通常用来处理用户的请求并返回结果。运行该程序,结果如图 5-7 所示。

结果可以看出,此时服务端运行起来了,并且在 5000 端口监听客户端发起的请求。打开浏览器,在地址栏中输入本地地址 localhost:5000,就可以得到结果,如图 5-8 所示。注意,端口号 5000 不可以省略。