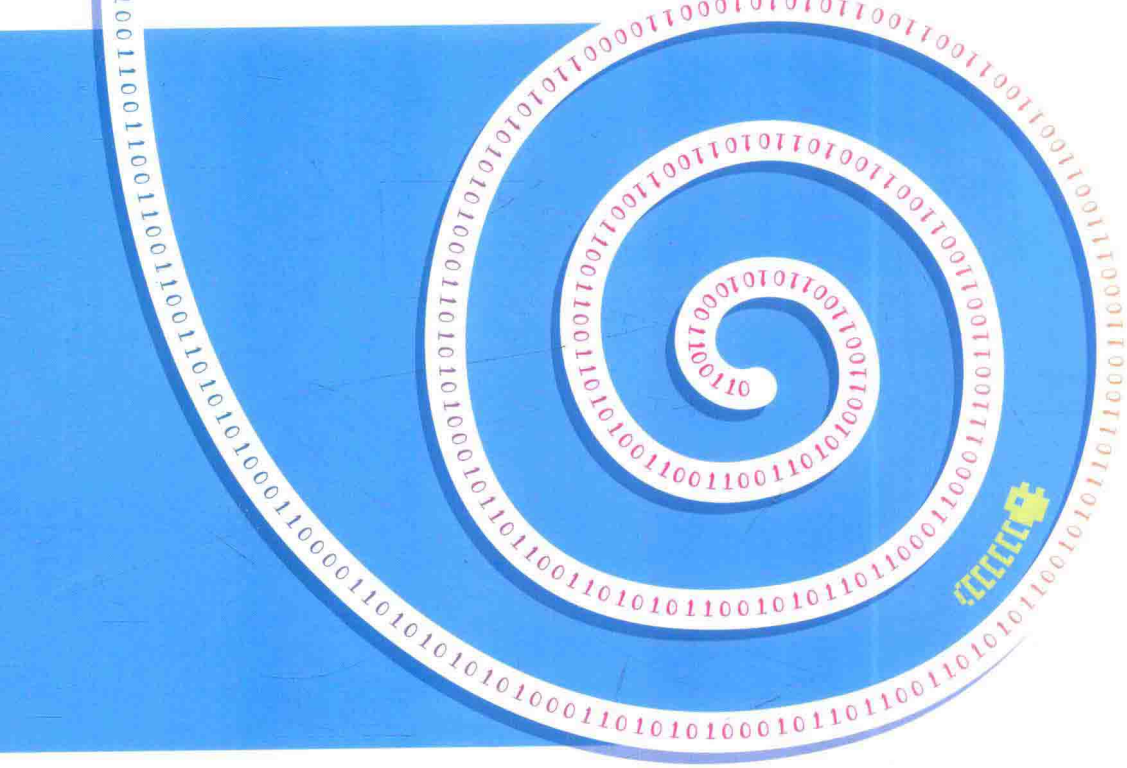
Python 大数据技术与应用系列

数据采集技术

——Python网络爬虫项目化教程

黄锐军　主编

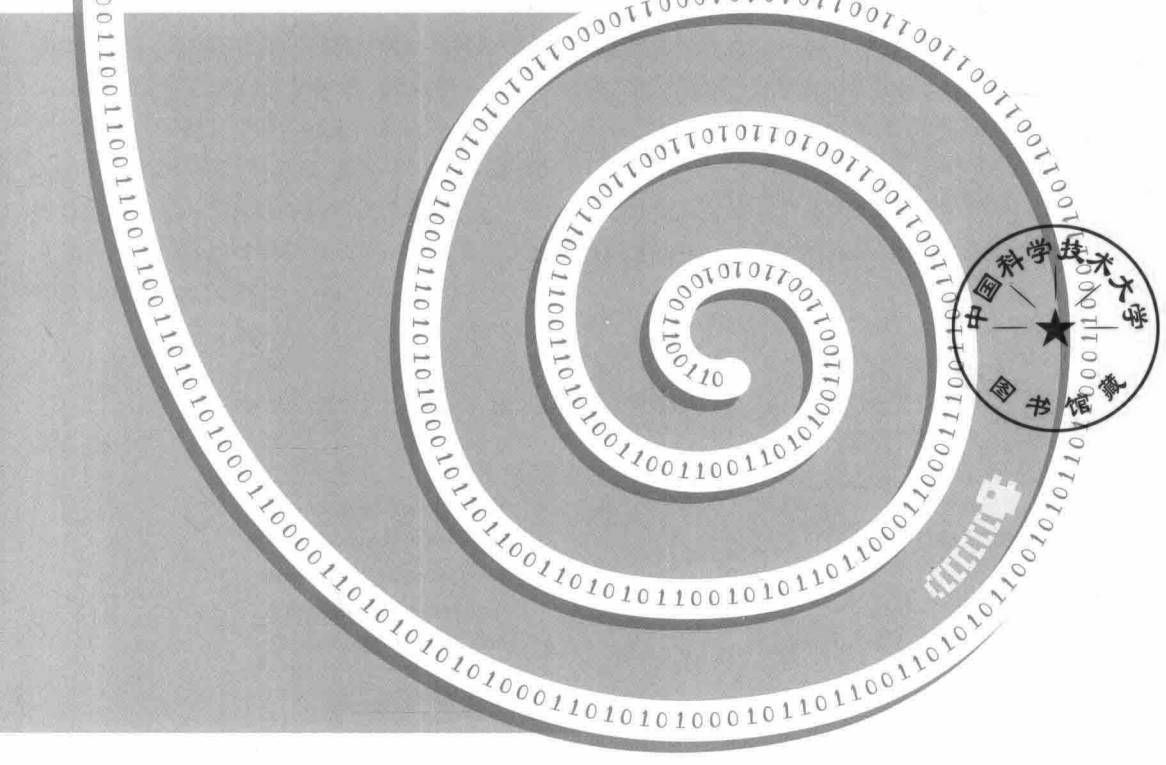


高等教育出版社

Python 大数据技术与应用系列

数据采集技术
——Python网络爬虫项目化教程

黄锐军　主编



中国科学技术大学 图书馆藏

高等教育出版社·北京

内容提要

网络爬虫就是一组能自动从网站的相关网页中搜索与提取数据的程序，这些数据是进一步实现数据分析的关键与前提。Python 语言程序简单高效，编写网络爬虫有特别的优势，尤其业界有专门为 Python 编写的各种各样的爬虫程序框架，使得 Python 编写爬虫程序更加简单高效。

本书主要分成 4 个项目，项目 1 以爬取学生信息的项目为依托，讲解 Python 的 Web 访问技术，它是爬虫程序的基础。项目 2 以爬取城市天气预报项目为依托，讲解 BeautifulSoup 网页数据的爬取方法。项目 3 以爬取网络图像项目为依托，讲解网页的深度优先与广度优先顺序爬取路径的构造方法与多线程分布式网页爬取技术。项目 4 以爬取网站的图书信息项目为依托，讲解目前功能强大的分布式爬取框架 Scrapy 的程序设计技术。每个部分都遵循由浅入深的学习规律，理论与实践相结合，提高学生的实践能力。

本书为新形态一体化教材，配有丰富的教学资源，包括微课、教学大纲、课程标准、教学课件、案例源码、课后习题及习题答案等。本书同时配套建设了在线开放课程，学习者可登录智慧职教 MOOC 学院（mooc.icve.com.cn）平台，在“Python 程序设计”页面进行学习。教师可发邮件至编辑邮箱 1548103297@qq.com 索取教学资源。本书可作为计算机软件技术专业、大数据技术应用专业及其他专业的专业教材，也可作为数据采集技术学习者的自学参考书。

图书在版编目（CIP）数据

数据采集技术：Python 网络爬虫项目化教程 / 黄锐军主编. --北京：高等教育出版社，2018.8

ISBN 978-7-04-049781-6

Ⅰ. ①数… Ⅱ. ①黄… Ⅲ. ①软件工具－程序设计－高等职业教育－教材 Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2018）第 107353 号

策划编辑 马万里　　责任编辑 马万里　　封面设计 赵 阳　　版式设计 于 婕

责任校对 陈 杨　　责任印制 韩 刚

出版发行	高等教育出版社	网　址	http://www.hep.edu.cn
社　址	北京市西城区德外大街 4 号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.hepmall.com.cn
印　刷	北京印刷一厂		http://www.hepmall.com
开　本	787mm × 1092mm　1/16		http://www.hepmall.cn
印　张	9.25		
字　数	230 千字	版　次	2018 年 8 月第 1 版
购书热线	010-58581118	印　次	2018 年 8 月第 1 次印刷
咨询电话	400-810-0598	定　价	28.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

版权所有　侵权必究

物 料 号　49781-00

前 言

网络爬虫是一组能自动从网站的相关网页中搜索与提取数据的程序，而提取与存储这些数据是进一步实现数据分析的关键与前提。Python 是一种面向对象的解释型计算机程序设计语言。该语言开源、免费、功能强大，而且语法简洁清晰，具有丰富和强大的库，是目前应用广泛的程序语言。Python 编写网络爬虫有特别的优势，尤其业界有专门为 Python 编写的各种各样的爬虫程序框架，使得爬虫程序的编写更加简单高效。

本书内容分 4 个项目进行讲解。其中，项目 1 以爬取学生信息的项目为依托，讲解 Python 的 Web 访问技术，它是爬虫的程序基础；项目 2 以爬取城市天气预报项目为依托，讲解网页数据的爬取方法，重点讲解了 BeautifulSoap 的数据分析与提取方法；项目 3 以爬取网络图像项目为依托，讲解爬取多个网页数据的方法，重点讲解网页的深度优先与广度优先顺序爬取路径的构造方法，同时还讲解了程序多线程的网页爬取技术；项目 4 以爬取网站的图书信息项目为依托，讲解目前功能强大的分布式爬取框架 Scrapy 的程序设计技术。每个项目的内容都由浅入深地进行讲解，理论与实践相结合，提高实践能力。

本书为新形态一体化教材，配有丰富的教学资源，包括微课、教学大纲、课程标准、教学课件、案例源码、课后习题及习题答案等。本书同时配套建设了在线开放课程，学习者可登录智慧职教 MOOC 学院（mooc.icve.com.cn）平台，在“Python 程序设计”页面进行学习。老师可发邮件至编辑邮箱 1548103297@qq.com 索取教学资源。本书可作为计算机软件技术专业、大数据技术与应用专业及其他相关专业的专业教材，也可作为数据采集技术学习者的自学参考书。由于编者知识水平有限，本书中难免出现错误或不妥之处，欢迎批评指正，编者邮箱为 3076928517@qq.com。

黄锐军

2017 年 12 月

目 录

试读结束：需要全本请在线购买：www.ertongbook.com

项目 1　爬取学生信息

本项目介绍使用 Flask 微型框架快速建立 Python 的 Web 网站服务器程序的方法，以及使用 Urllib 程序包访问网站的过程，并通过一个爬取学生信息的综合案例巩固所学习知识与技能。

PPT　爬虫程序开发环境

PPT

1.1　爬虫程序开发环境

1.1.1　爬虫程序简介

微课 1
爬虫程序开发环境

爬虫程序是一组客户端程序，它的功能是访问 web 服务器，从服务器中获取网页代码。网页代码中包含了各种各样的数据信息，程序从中提取所需要的数据，把数据整理后存储到本地数据库中，这些数据将应用在数据分析等领域中。

例如，要想知道一个城市一段时间内的天气预报，就可以设计一组程序去访问有天气预报数据的网站，图 1-1-1 所示为一个天气预报的网站代码。爬虫程序的任务就是要从这一大段复杂的代码中提取所需要的天气状况、温度、风力等数据，把这些数据存储在数据库中。

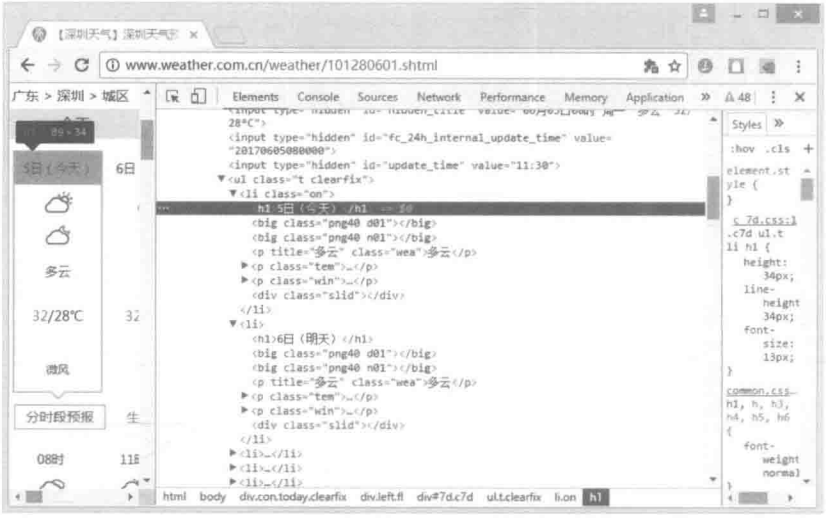


图 1-1-1
网页代码

编写一个爬虫程序可以使用 Python、Java、C++、C#等各种常用的开发语言，使用 Python 是比较简单也是比较流行的一种方法。

爬虫程序爬取的数据往往很多，而且相关的数据往往分布在很多不同的网页中，甚至分布在相关联的很多不同的网站中，爬虫程序必须能按链接自动往返于这些不同的网站中去爬取数据，一个爬虫程序爬取成百万上千万条的数据是常有的事，如何设计一个高效的爬虫程序是本书的重点。

1.1.2　Python 开发环境搭建

Python 是一种面向对象的解释型计算机程序设计语言，由荷兰人 Guido van Rossum 于 1989 年发明，第一个公开发行版发行于 1991 年。

Python 语言具有以下特点：

- 开源、免费、功能强大；
- 语法简洁清晰，强制用空白符（White Space）作为语句缩进；
- 具有丰富和强大的库；
- 易读、易维护，用户欢迎且用途广泛；

- 是解释性语言，变量类型可变，类似 JavaScript；

Python 安装后带有一个命令行工具以及一个小的 IDE 程序，但是这个 IDE 很弱，因此在此基础上可以搭配第三方的各种 IDE 开发工具，下面介绍几种主流的开发工具与环境。

（1）Python 自带开发环境

Python 的开发环境十分简单，用户可以到官网 https://www.python.org/中直接下载 Python 的程序包。目前 Python 有两个主流的版本，一个是 Python2.7，另外一个是 Python 3.6，这两个版本在语法上有些差异，本书主要使用 Python3.6。

下载 Python 3.6 程序包后，选择安装目录直接安装。Python 安装完毕后，在 WIndows 的启动菜单中可以看到 Python3.6 的启动项，启动 Python3.6 可以看到 Python 的命令行界面。这个环境是命令行环境，只能运行一些简单的测试语句。Python 自带一个 IDE，但是这个 IDE 的功能十分有限，不适合开发 Python 工程项目。

（2）PyCharm 与 Python 的开发环境

一个比较流行的开发环境是 PyCharm，它的风格类似 Eclipse，是一种专门为 Python 开发的 IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。

读者可以到 PyCharm 的官网 http://www.jetbrains.com/pycharm/下载免费的 PyCharm Community 版本。

（3）Anaconda 与 Python 的开发环境

另一个比较流行的开发环境是 Anaconda，这个程序比较庞大，但它是一个十分强大的 Python 开发环境，自带 Python 解释器，在安装 Anaconda 的同时自动安装 Python。它还带有一个功能强大的 IDE 开发工具 Spider。Anaconda 最大的好处是可以帮助用户找到与安装 Python 的各种各样的开发库，使得 Python 的开发十分方便与高效。另外，Anaconda 对 Windows 用户十分有用，因为 Python 的一些开发库在 Windows 环境下安装常常出现各种问题，而 Anaconda 能顺利解决这些问题。读者可以到官网 https://www.continuum.io/downloads 下载 Anaconda。

PPT Flask 简介

PPT

1.2 Flask Web 网站

1.2.1 Flask 简介

Python 的 Web 程序开发工具很多。Flask 是一种非常容易上手的 Python Web 开发框架，不需要知道太多的 MVC 的概念，只需要具备基本的 Python 开发技能，就可以开发出一个 Web 应用。

微课 2
Flask 简介

Flask 的官网：http://flask.pocoo.org/。

Flask 中文文档：http://dormousehole.readthedocs.org/en/latest/。

就像文档中提示的，可以先读《安装》，然后阅读《快速上手》。Flask 支持强大的扩展功能，以各种方式扩展了 Flask 的功能，比如增强对数据库的支持等。

（1）Flask 安装

在 Windows 系统中使用 Flask，安装方法非常简单，根据文档的介绍直接在命令行窗口执行：

```
pip install flask
```

如果最后显示：

```
Successfully installed flask Werkzeug Jinja2 itsdangerous markupsafe
Cleaning up...
```

则表示 Flask 已安装成功。

（2）Flask 实例

编写程序：

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

@app.route("/")
def hello():
    return "你好"

@app.route("/hi")
def hi():
    return "Hi,你好"

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

执行该程序可以看到显示 http://127.0.0.1:5000 的 Web 地址，在浏览器中输入这个 Web 地址看到显示“你好”，如果输入 http://127.0.0.1:5000/hi，则显示“Hi，你好”。

下面分析程序的功能：

① import flask

这条语句是引入 Flask 程序包，在 Flask 正确安装后都能正常引入。

② app=flask.Flask(__name__)

这条语句是初始化一个 Flask 对象，参数__name__是程序的名称。

③

```
@app.route("/")
def hello():
    return "你好"
```

这是一段路由控制语句，每个路由地址用"@app.route(...)"来指明，在访问相对地址是"/"时就执行函数 hello()，因此访问地址 http://127.0.0.1:5000 时看到“你好”。

④

```
@app.route("/hi")
def hi():
    return "Hi,你好"
```

这也是一段路由控制语句，在访问相对地址是"/hi"时就执行函数 hi()，因此访问地址 http://127.0.0.1:5000/hi 时看到“Hi，你好”。

⑤

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

这两句语句表示在主程序中执行 app.run()，app.run()执行后就启动了一个 Web 服务器，它的默认地址是 http://127.0.0.1:5000.

（3）Flask 显示静态网页

如果在程序的同一文件夹中有一个静态网页，如 index.htm，那么很容易用 Flask 编写一个 Web 网站程序 server.py，它的主页就是 index.htm，具体程序如下：

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

@app.route("/")
def index():
    try:
        fobj=open("index.htm","rb")
        data=fobj.read()
        fobj.close()
        return data
    except Exception as err:
        return str(err)

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

程序 server.py 功能的是启动一个 Web 服务，在访问网站时读取同一个文件夹下的 index.htm 文件，然后向客户端（浏览器）返回 index.htm 文件的内容。

例如 index.htm 的内容是：

```
<h1>Welcome Python Flask Web</h1>
It is very easy to make a website by Python Flask.
```

把这个文件按 UTF8 编码保存到 Python 程序所在的文件夹中，运行程序后访问网址 http://127.0.0.1:5000，结果如图 1-2-1 所示。

图 1-2-1
Flask Web 网站

试读结束：需要全本请在线购买：www.ertongbook.com

微课 3
Urllib 程序包访问 Web 网站

1.2.2　Urllib 程序包访问 Web 网站

server.py 程序的这个网站除了可以使用浏览器访问外，也可以使用 Urllib 程序包中的相关函数编写程序来访问。设计一个 client.py 程序如下：

```
import urllib.request
url="http://127.0.0.1:5000"
html = urllib.request.urlopen(url)
html = html.read()
html = html.decode()
print(html)
```

在运行了 server.py 程序后运行 client.py 得到如下结果：

```
<h1>欢迎使用 Python Flask Web</h1>
It is very easy to make a website by Python Flask.<p>
```

很容易用 Python Flask 制作一个 Web 网站<p>

由此可见，得到的是 index.htm 网页的内容，现在来分析这个程序的功能：

```
import urllib.request
```

这条语句的作用是引入 urllib.request 程序包。这是 Python 自带的程序包，不需要安装，其作用是访问网站。

```
html = urllib.request.urlopen(url)
```

这条语句的作用是打开 URL 网址的网址，其中，urllib.request 是 Urllib 中的一个子程序包；urlopen 是打开网站的函数。

```
html = html.read()
```

这个网站打开后就如同打开文件一样，要使用 read 函数读取网站的内容，读出的是二进制数据。

```
html = html.decode()
```

这条语句的作用是把二进制数据 html 转为字符串，转换的编码是 UTF-8，默认时 decode()是使用 UTF-8 编码，也可以指定转换编码。例如：html=html.decode("utf-8")或者 html= html.decode("gbk")。具体采用什么编码是看网站的网页说明编码，如果编码不正确会出现汉字乱码。

```
print(html)
```

显示网站的网页内容。由此可见传递过来的就是 index.htm 的网页数据。

由此可见，urllib.request.urlopen(url)是一个很重要的函数，它可以打开一个 URL 网址的网站。

PPT Get 方法访问网站

PPT

1.3 GET 方法访问网站

微课 4
GET 方法访问网站

访问网站最常用的一种方法是 GET 方法。这种方法主要用于客户端从服务器端获取网站数据，如果有交互，客户端把参数附加在网址的后面向服务器提供参数，服务器接收这些参数后作出不同的响应。

1.3.1 客户端 GET 方式发送数据

微课 5
客户端 GET 方式发送数据

GET 方式发送使得数据附加在 URL 后面，在 URL 后面先接一个“?”号，数据采用“名称 1=值 1&名称 2=值 2&名称 3=值 3……”的方式，多个数据之间用“&”符号隔开。例如，向服务器传递省份和城市的数据就可以这样写：

```
urllib.request.urlopen("http://127.0.0.1:5000?province=GD&city=SZ")
```

如果参数值包含汉字，那么必须使用 urllib.parse.quote 对参数值进行编码。例如：

```
province= urllib.parse.qoute("广东")
city= urllib.parse.qoute("深圳")
urllib.request.urlopen("http://127.0.0.1:5000?province="+province+"&city="+city)
```

编写客户端程序 client.py 如下：

```
import urllib.parse
import urllib.request
url="http://127.0.0.1:5000"
try:
    province= urllib.parse.quote("广东")
    city= urllib.parse.quote("深圳")
    data="province="+province+"&city="+city
    html=urllib.request.urlopen("http://127.0.0.1:5000? "+data)
    html = html.read()
    html = html.decode()
    print(html)
except Exception as err:
    print(err)
```

1.3.2 服务器获取 GET 发送的数据

服务器用 Flask 中的 request 对象的 args 来存储 GET 的参数，用 get 方法来获取参数，即用 flask.request.args.get(参数)来获取参数的值。例如：

```
province=flask.request.args.get("province")
city=flask.request.args.get("city")
```

就可以获取 GET 传递的参数 province 与 city 的值。

编写服务器程序 server.py 如下：

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

@app.route("/")
def index():
    try:
        province=flask.request.args.get("province")
        city = flask.request.args.get("city")
        return province+","+city
    except Exception as err:
        return str(err)

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

先运行 server.py 建立 Web 网站，再运行 client.py，可以看到 client.py 的结果是：

```
广东,深圳
```

注意，Web 网址是 http://127.0.0.1:5000，如果直接访问这个网站会出现错误，因为没有提供 province 与 city 参数，服务器执行：

```
province=flask.request.args.get("province")
city = flask.request.args.get("city")
```

这时会到 args 中查找 province 和 city 参数的值，结果因没有找到而出现错误。为了避免这种错误，可以把语句写成：

```
province=flask.request.args.get("province") if "province" in flask.request.args else ""
city = flask.request.args.get("city") if "city" in flask.request.args else ""
```

这样，在"province"和"city"没有出现在 flask.request.args 中时，province、city 值就设置为空字符串。服务器程序改进如下：

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

@app.route("/")
def index():
    try:
        province=flask.request.args.get("province") if "province" in flask.request.args
else ""
        city = flask.request.args.get("city") if "city" in flask.request.args else ""
        return province+","+city
```

```
        except Exception as err:
            return str(err)

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

1.4 POST 方法向网站发送数据

PPT 客户端 POST 发送数据

1.4.1 客户端 POST 发送数据

采用 POST 方法访问网站时，客户端向服务器发送表单数据，表单数据的组织方式与 GET 方法的参数列表十分相似，结构如下：

```
"名称 1=值 1&名称 2=值 2&名称 3=值 3……"
```

多个数据之间用“&”符号隔开，如果参数值包含汉字，那么必须使用 urllib.parse.quote 对参数值进行编码。例如：

```
province= urllib.parse.qoute("广东")
city= urllib.parse.qoute("深圳")
data="province="+province+"&city="+city
data=data.encode()
urllib.request.urlopen("http://127.0.0.1:5000",data=data)
```

这里，data=data.encode()是把 data 字符串按 UTF-8 编码转换为二进制数据。POST 方法与 GET 方法最大的不同是：GET 的参数放在地址栏的后面，而 POST 的数据放在 urlopen 函数的 data 参数中，而且这个参数值必须是二进制数据。

编写客户端 client.py 程序如下：

```
import urllib.parse
import urllib.request
url="http://127.0.0.1:5000"
try:
    province= urllib.parse.quote("广东")
    city= urllib.parse.quote("深圳")
    data="province="+province+"&city="+city
    data=data.encode()
    html=urllib.request.urlopen("http://127.0.0.1:5000",data=data)
    html = html.read()
    html = html.decode()
    print(html)
except Exception as err:
    print(err)
```

1.4.2　服务器获取 POST 的数据

服务器用 Flask 中的 request 对象的 form 来存储 GET 的参数，用 get 方法来获取参数，即用 flask.request.form.get(参数)来获取参数的值。例如：

```
province=flask.request.form.get("province")
city=flask.request.form.get("city")
```

该语句可以获取 GET 传递的参数 province 与 city 的值。

编写服务器程序 server.py 如下：

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

@app.route("/",methods=["POST"])
def index():
    try:
        province=flask.request.form.get("province") if "province" in flask.request.form
else ""
        city = flask.request.form.get("city") if "city" in flask.request.form else ""
        return province+","+city
    except Exception as err:
        return str(err)

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

值得注意的是，在服务器中要指定：

```
@app.route("/",methods=["POST"])
def index():
```

表明这个函数接收 POST 请求。默认时只接收 GET 请求，要接收 POST 请求就必须明确指明，如果写成：

```
@app.route("/",methods=["GET","POST"])
def index():
```

那么，这个函数即可以接收 GET 请求，也可以接收 POST 请求。

先运行 server.py 建立 Web 网站，再运行 client.py，可以看到 client.py 的结果是：

```
广东,深圳
```

微课 6
GET 与 POST 的混合使用

1.4.3　GET 与 POST 的混合使用

实际上，在应用中客户端同时使用 GET 与 POST 向服务器发送数据，一般 GET 的数据放在地址栏的后面，参数简单，数据量少，而 POST 的数据是表单数据，数据量大。

例如，客户端向服务器发送一个城市的简介，服务器接收后返回收到的信息。

我们把省份 province 与城市 city 的名称放在 URL 地址栏后面采用 GET 方式发送给服务器，然后把城市的简介用 POST 方法发送给服务器，客户端程序如下：

```
import urllib.parse
import urllib.request
url="http://127.0.0.1:5000"
```

note="深圳依山傍海，气候宜人，实在是适合人类居住的绝佳地。这里四季如春，干净整洁，比邻香港，拥有着丰富的自然景观和人文气息。匆匆过客注意到的也许只有它的时尚繁华，忙碌的暂居者也可能对它有着不识城市真面目之感。只有世代在此生活的老深圳人，才明白它从贫穷走向富饶经历了怎样的艰辛。"

```
try:
    province= urllib.parse.quote("广东")
    city= urllib.parse.quote("深圳")
    note= "note="+urllib.parse.quote(note)
    param="province="+province+"&city="+city
    html=urllib.request.urlopen("http://127.0.0.1:5000?"+param,data=note.encode())
    html = html.read()
    html = html.decode()
    print(html)
except Exception as err:
    print(err)
```

服务器程序如下：

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

@app.route("/",methods=["GET","POST"])
def index():
    try:
        province=flask.request.args.get("province") if "province" in flask.request.args
else ""
        city = flask.request.args.get("city") if "city" in flask.request.args else ""
        note = flask.request.form.get("note") if "note" in flask.request.form else ""
        return province+","+city+"\n"+note
    except Exception as err:
        return str(err)

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


试读结束：需要全本请在线购买： www.ertongbook.com