

# Python Spider

张志鹏

Published  
with GitBook



# 目錄

---

1. [README](#)
2. [目录](#)
3. [第一部分 入门指南](#)
  - i. [第1章 HTTP概述](#)
    - i. [1.1 理解 HTTP](#)
      - i. [1.1.1 HTTP 是什么](#)
      - ii. [1.1.2 HTTP 结构](#)
    - ii. [1.2 HTTP请求](#)
      - i. [1.2.1 Web 客户端](#)
      - ii. [1.2.2 请求地址: 域名/IP/端口](#)
      - iii. [1.2.3 请求类型: GET/POST/...](#)
      - iv. [1.2.4 客户端器发送了什么](#)
      - v. [1.2.5 客户端器接收了什么](#)
      - vi. [1.2.6 我们看到了什么](#)
    - iii. [1.3 HTTP响应](#)
      - i. [1.3.1 Web 服务端](#)
      - ii. [1.3.2 解析HTTP请求](#)
      - iii. [1.3.3 构建响应报文](#)
      - iv. [1.3.4 发送响应至客户端](#)
      - v. [1.3.5 响应正文类型](#)
      - vi. [1.3.6 响应状态](#)
    - iv. [1.4 小结](#)
      - i. [1.4.1 小结](#)
      - ii. [1.4.2 参考](#)
  - ii. [第2章 网络爬虫概述](#)
    - i. [2.1 理解爬虫](#)
      - i. [2.1.1 网络爬虫究竟是什么](#)
      - ii. [2.1.2 如何更好的理解爬虫](#)
      - iii. [2.1.3 爬虫可以用来做什么](#)
    - ii. [2.2 爬虫搜索策略](#)
      - i. [2.2.1 深度优先搜索策略](#)
      - ii. [2.2.2 广度优先搜索策略](#)
      - iii. [2.2.3 最佳优先搜索策略](#)
      - iv. [2.2.4 对比搜索策略\(查水表\)](#)
    - iii. [2.3 爬虫类型](#)
      - i. [2.3.1 通用爬虫](#)
      - ii. [2.3.2 聚焦爬虫](#)
      - iii. [2.3.3 定向爬虫](#)
    - iv. [2.4 小结](#)
      - i. [2.4.1 小结-你真的需要通用爬虫吗](#)
      - ii. [2.4.2 参考](#)
  - iii. [第3章 小试牛刀，写一个小爬虫](#)
    - i. [3.1 如何写一个爬虫](#)
      - i. [3.1.1 你需要什么数据](#)
      - ii. [3.1.2 分析网站-不要相信你看到的](#)
      - iii. [3.1.3 数据抽取-如何拿到你想要的数据](#)
      - iv. [3.1.4 数据存储-选择合适的方式存储你"拿"到的数据](#)
    - ii. [3.2 实战分析：把《花千骨》都"爬"下来](#)
      - i. [3.2.1 分析搜索页面](#)

- ii. 3.2.2 分析播放列表页
    - iii. 3.2.3 分析播放页
    - iv. 3.2.4 完整示例
  - iii. 3.3 小结
    - i. 3.3.1 数据格式 - html/json/...
    - ii. 3.3.2 小结
    - iii. 3.3.3 参考
- 4. 第二部分 进阶爬虫
  - i. 第4章 调试工具
    - i. 4.1 下载 抓包 调试
      - i. 4.1.1 Chrome/插件
      - ii. 4.1.2 Python/PyCharm
      - iii. 4.1.3 wget/curl
      - iv. 4.1.4 Fiddler
    - ii. 4.2
      - i. 4.2.1
      - ii. 4.2.2
      - iii. 4.2.3
      - iv. 4.2.4
    - iii. 4.3
      - i. 4.3.1
      - ii. 4.3.2 小结
      - iii. 4.3.3 参考
  - ii. 第5章 Python库介绍
    - i. 5.1 内置库
      - i. 5.1.1 下载：urllib/urllib2/httpplib
      - ii. 5.1.2 解析、存储：json/re/HtmlParser/lxml/pickle
      - iii. 5.1.3 时间日期：time/datetime/random
      - iv. 5.1.4 其他：os/sys/logging/copy/ConfigParser/xml2dict
    - ii. 5.2 第三方库
      - i. 5.2.1 下载：requests/(python-goose)
      - ii. 5.2.2 解析、存储：simplejson
      - iii. 5.2.3 解析抽取：BeautifulSoup4/bs4
      - iv. 5.2.4 解析抽取：pyquery
      - v. 5.2.5 存储：MySQLdb/mysql-connector/sqlite/redis/...
      - vi. 5.2.6 其他
    - iii. 5.3 小结
      - i. 5.3.2 小结
      - ii. 5.3.3 参考
  - iii. 第6章 高级爬虫
    - i. 6.1 模拟请求之Header处理
      - i. 6.1.1 User-Agent
      - ii. 6.1.2 Accept-Encoding
      - iii. 6.1.3 Referer/Host
      - iv. 6.1.4 Cookies
    - ii. 6.2 如何应对各种反爬虫策略
      - i. 6.2.1 IP限制：proxy-代理
      - ii. 6.2.2 不知道从哪多出来的参数：ajax/js python-spidermonkey, PyV8
      - iii. 6.2.3 看不到更多内容，各种权限限制：登录
      - iv. 6.2.4 反"人类"验证码：验证码图片识别
      - v. 6.2.5 反爬策略太完美：selenium.webdriver (终极大招)
    - iii. 6.3 小结
      - i. 6.3.2 小结

- ii. 6.3.3 参考
    - iv. 第7章 爬虫实战 ..???
  - 5. 第三部分 打开框架的大门
    - i. 第8章 爬虫框架 (scrapy-redis/python-rq/webscraping)
      - i. 8.1 scrapy - 基于 twisted
      - ii. 8.2 pypider - 基于 tornado、可在线调试
      - iii. 8.3 Cola - 分布式爬虫框架
      - iv. 8.4 selenium.webdriver - 看不见的浏览器
      - v. 8.5 小结
    - ii. 第9章 打造属于你自己的框架
      - i. 9.1 需求分析
      - ii. 9.2 架构设计
      - iii. 9.3 代码编写
      - iv. 9.4 小结
        - i. 9.4.1 小结
        - ii. 9.4.2 这样就够了吗(页面不止两层, 甚至更多; 是否考虑分布式)
        - iii. 9.4.2 参考
    - iii. 第10章 多线程、多进程、协程、分布式
  - 6. 第四部分 总结整理
    - i. 第11章 大大小小的爬虫
    - ii. 第12章 环境搭建
      - i. 12.1 安装 Python
        - i. 12.1.1 Windows
        - ii. 12.1.2 Linux
      - ii. 12.2 安装 Python 包管理工具
        - i. 12.2.1 easy\_install/pip
        - ii. 12.2.2 安装第三方包(Windows/Linux)
      - iii. 12.3 选择合适你的IDE
        - i. 12.3.1 PyCharm
        - ii. 12.3.2 Sublime Text
        - iii. 12.3.3 其他 Emeditor/Vi/Vim ...
      - iv. 12.4 Windows上的Linux环境
        - i. 12.4.1 安装linux虚拟机
        - ii. 12.4.2 安装bash环境(Git bash/MobaXTerm)
      - v. 12.5 数据库安装
        - i. 12.5.1 安装MySQL
        - ii. 12.5.2 安装Redis
        - iii. 12.2.3 其他
      - vi. 12.6 其他
        - i. 12.6.1 Chrome
        - ii. 12.6.2 Fiddler
  - 7. 附录 防坑必备

这是一本python 爬虫的教学书籍

# 目录

---

## 第一部分 入门指南

---

### 第1章 HTTP概述

---

- **1.1 理解 HTTP**
    - **1.1.1 HTTP 是什么**
    - **1.1.2 HTTP 结构**
  - **1.2 HTTP请求**
    - **1.2.1 Web 客户端**
    - **1.2.2 请求地址: 域名/IP/端口**
    - **1.2.3 请求类型: GET/POST/...**
    - **1.2.4 客户端器发送了什么**
    - **1.2.5 客户端器接收了什么**
    - **1.2.5 我们看到了什么**
  - **1.3 HTTP响应**
    - **1.3.1 Web 服务端**
    - **1.3.2 解析HTTP请求**
    - **1.3.3 构建响应报文**
    - **1.3.4 发送响应至客户端**
    - **1.3.5 响应正文类型**
    - **1.3.6 响应状态**
  - **1.4 小结**
    - **1.4.1 小结**
    - **1.4.2 参考**
-

## 第2章 网络爬虫概述

---

- **2.1 理解爬虫**
  - **2.1.1 网络爬虫究竟是什么**
  - **2.1.2 如何更好的理解爬虫**
  - **2.1.3 爬虫可以用来做什么**
- **2.2 爬虫搜索策略**
  - **2.2.1 深度优先搜索策略**
  - **2.2.2 广度优先搜索策略**
  - **2.2.3 最佳优先搜索策略**
  - **2.2.4 对比搜索策略(查水表)**
- **2.3 爬虫类型**
  - **2.3.1 通用爬虫**
  - **2.3.2 聚焦爬虫**
  - **2.3.3 定向爬虫**
- **2.4 小结**
  - **2.4.1 小结-你真的需要通用爬虫吗**
  - **2.4.2 参考**

---

## 第3章 小试牛刀，写一个小爬虫

---

- **3.1 如何写一个爬虫**
  - **3.1.1 你需要什么数据**
  - **3.1.2 分析网站-不要相信你看到的**
  - **3.1.3 数据抽取-如何拿到你想要的**
  - **3.1.4 数据存储-选择合适的方式存储你"拿"到的数据**
- **3.2 实战分析：把《花千骨》都"爬"下来**

- **3.2.1** 分析搜索页面
  - **3.2.2** 分析播放列表页
  - **3.2.3** 分析播放页
  - **3.2.4** 完整示例
  - **3.3** 小结
    - **3.3.1** 数据格式 - html/json/...
    - **3.3.2** 小结
    - **3.3.3** 参考
- 

## 第二部分 进阶爬虫

---

### 第4章 调试工具

---

- **4.1** 下载 抓包 调试
    - **4.1.1** Chrome/插件
    - **4.1.2** Python/PyCharm
    - **4.1.3** wget/curl
    - **4.1.4** Fiddler
  - **4.2**
    - **4.2.1**
    - **4.2.2**
    - **4.2.3**
    - **4.2.4**
  - **4.3**
    - **4.3.1**
    - **4.3.2** 小结
    - **4.3.3** 参考
-



---

## 第5章 Python库介绍

---

- **5.1 内置库**
  - **5.1.1 下载** : `urllib/urllib2/httpplib`
  - **5.1.2 解析、存储** : `json/re/HtmlParser/lxml/pickle`
  - **5.1.3 时间日期** : `time/datetime/random`
  - **5.1.4 其他** : `os/sys/logging/copy/ConfigParser/xml2dict`
- **5.2 第三方库**
  - **5.2.1 下载** : `requests/(python-goose)`
  - **5.2.2 解析、存储** : `simplejson`
  - **5.2.3 解析抽取** : `BeautifulSoup4/bs4`
  - **5.2.4 解析抽取** : `pyquery`
  - **5.2.5 存储** : `MySQLdb/mysql-connector/sqlite/redis/...`
  - **5.2.6 其他**
- **5.3 小结**
  - **5.3.2 小结**
  - **5.3.3 参考**

---

## 第6章 高级爬虫

---

- **6.1 模拟请求之Header处理**
  - **6.1.1 User-Agent**
  - **6.1.2 Accept-Encoding**
  - **6.1.3 Referer/Host**
  - **6.1.4 Cookies**
- **6.2 如何应对各种反爬虫策略**

- **6.2.1 IP限制**：**proxy**-代理
  - **6.2.2 不知道从哪多出来的参数**：**ajax/js python-spidermonkey, PyV8**
  - **6.2.3 看不到更多内容，各种权限限制**：**登录**
  - **6.2.4 反"人类"验证码**：**验证码图片识别**
  - **6.2.5 反爬策略太完美**：**selenium.webdriver (终极大招)**
  - **6.3 小结**
    - **6.3.2 小结**
    - **6.3.3 参考**
- 

## 第7章 爬虫实战 ..???

---

### 第三部分 打开框架的大门

---

## 第8章 爬虫框架 (scrapy-redis/python-rq/web scraping)

---

- **8.1 scrapy** - 基于 **twisted**
- **8.2 pypider** - 基于 **tornado**、可在线调试
- **8.3 Cola** - 分布式爬虫框架
- **8.3 selenium.webdriver** - 看不见的浏览器
- **8.4 小结**

## 第9章 打造属于你自己的框架

---

- **9.1 需求分析**
- **9.2 架构设计**
- **9.3 代码编写**
- **9.4 小结**
  - **9.4.1 小结**
  - **9.4.2 这样就够了吗(页面不止两层，甚至更多; 是否考虑分布式)**

- **9.4.2 参考**

## 第10章 多线程、多进程、协程、分布式

---

## 第四部分 总结整理

---

## 第11章 大大小小的爬虫

---

## 第12章 环境搭建

---

- **12.1 安装 Python**
  - **12.1.1 Windows**
  - **12.1.2 Linux**
- **12.2 安装 Python 包管理工具**
  - **12.2.1 easy\_install/pip**
  - **12.2.2 安装第三方包(Windows/Linux)**
- **12.3 选择合适你的IDE**
  - **12.3.1 PyCharm**
  - **12.3.2 Sublime Text**
  - **12.3.3 其他 Emeditor/Vi/Vim ...**
- **12.4 Windows上的Linux环境**
  - **12.4.1 安装linux虚拟机**
  - **12.4.2 安装bash环境(Git bash/MobaXTerm)**
- **12.5 数据库安装**
  - **12.5.1 安装MySQL**
  - **12.5.2 安装Redis**
  - **12.2.3 其他**
- **12.6 其他**
  - **12.6.1 Chrome**

- **12.6.2 Fiddler**

## 附录 防坑必备

---

## part 1 —— 第一部分 入门指南

---

第一部分部分宽泛的讲解学习爬虫要了解到的哪些知识，包括HTTP介绍、爬虫介绍，并且做一个稍微复杂爬虫示例。

第1章主要讲解HTTP，什么是HTTP，它是如何运作的。

第2章讲解爬虫的基本知识，什么是爬虫，爬虫搜索策略和类型。

第3章介绍如何抓取数据，并以一个示例进行介绍如何分析一个爬虫需求。

## 第1章 —— HTTP概述

---

超文本传输协议（HyperText Transfer Protocol，HTTP）是互联网上应用最为广泛的一种网络协议。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。通过HTTP或者HTTPS协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

HTTP的发展是万维网协会（World Wide Web Consortium，W3C）和互联网工程任务组（Internet Engineering Task Force，IETF）合作的结果，（他们）最终发布了一系列的RFC，其中最著名的是1999年6月公布的RFC 2616，定义了HTTP协议中现今广泛使用的一个版本—HTTP 1.1。

本章将讲解HTTP是什么，web客户端、web服务器是什么，它们之间是如何通过HTTP进行通信的，以及通信结构、格式是什么样子的。

HTTP是一个客户端终端（用户）和服务器端（网站）请求和应答的标准（TCP）。通过使用Web浏览器、网络爬虫或者其它的工具，客户端发起一个HTTP请求到服务器上指定端口（默认端口为80）。我们称这个客户端为用户代理程序（user agent）。应答的服务器上存储着一些资源，比如HTML文件和图像。我们称这个应答服务器为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理、网关或者隧道（tunnel）。

尽管TCP/IP协议是互联网上最流行的应用，HTTP协议中，并没有规定必须使用它或它支持的层。事实上，HTTP可以在任何互联网协议上，或其他网络上实现。HTTP假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用。因此也就是其在TCP/IP协议族使用TCP作为其传输层。

通常，由HTTP客户端发起一个请求，创建一个到服务器指定端口（默认是80端口）的TCP连接。HTTP服务器则在那个端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息、或者其它信息。

那么，HTTP究竟是什么呢？

通俗地讲，HTTP（又指HTTP协议，全名为“超文本传输协议”）指一种约束，协议并不是听起来的特别的深奥，它只是一个专业词。用于服务端（网站）和客户端（浏览器/用户）之间的数据传输。

要理解HTTP，用我们日常生活中能够形象表示的HTTP协议的东西，就是收发信件。

服务端，就是指一个资源站点。服务端，并不是我们想象中的，云里来雾里去。每一台电脑，都是服务端，都是客户端。通俗地讲，服务器可以理解为邮局、收信人家门口信箱。

客户端，就是指某个用户(电脑)、用户浏览器、访问服务端的程序。通俗地讲，客户端可以理解为发信人家里门口的信箱。

HTTP协议，就是指发送、接受信件过程（称之为邮递）的一种约束，可以指过程，也可以指对过程的规范。具体，可以认为是信封上面要写的内容，必须要写的内容有发信人、发信地址、发信邮编、收信人、收信地址、收信邮编、邮票等。这些内容，就可以认为是HTTP协议，是必须要填的信息，用于确认信件可以准确传达到指定邮局、指定收信人。



