B站：脾气暴躁的产品经理

练习网站：http://www.spiderbuf.cn

课程链接：https://www.bilibili.com/cheese/play/ss19916

代码链接：https://github.com/hhuayuan/spider-course

本课程的目标：让大家知其然，也知其所以然！

授人以渔。大部分爬虫教程都是教一些基础或者是直接找一些案例讲解，已经入门但未熟练的人难以找到适合的课程及练习网站；只教人爬不教原理，以至于部分人学完还是知其然不知其所以然，无法灵活应用；而且很多课程掺杂了大量Python基础语法等内容充集数、知识点不连贯或者避重就轻等。

本课程以横向教学为主，介绍爬虫实际工作中用到的技术、思路及工具，并且以边开发网页边爬取的方式逐步深入爬虫与反爬虫的攻防知识，知己知彼。

编程的一些注意事项：开发环境的路径以及Python脚本文件名**不要包含中文、空格等内容**；初学者推荐选用开箱即用的开发工具，如：PyCharm等。

本课程所用到的环境及软件：

Python解释器：Python 3.11.1

Node.js 18.20.2

Python开发工具：PyCharm Community Edition 2022.2.5

Web开发工具：VS Code 1.88.1

Web服务器：Nginx 1.24.0

抓包工具：Fiddler Classic 5.0.20211.51073 for .NET 4.6.1

浏览器：Google Chrome 123.0.6312.123（正式版本） （64 位）

操作系统：Windows 10 64位 家庭中文版

- User Agent
- Ajax动态加载
- 用户登录
- 无序号翻页
- Referer
- CSS样式偏移
- Base64编码图片
- 复杂样式
- JavaScript混淆
- 时间戳验证
- 浏览器指纹
- CSS Sprites（雪碧图）
- 用户行为检测

网页基本结构及HTML\CSS\JS基础知识：网页的基本构成及相关知识，为后期的网页内容提取打下基础。

最简单的完整网页

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8" />
    <title>这里是页面标题</title>
</head>
    <body>
        这里是网页内容。
    </body>
</html>
```

加入了CSS代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8" />
    <title>这里是页面标题</title>
    <style type="text/css">
        body{
            margin: 0;
            padding: 0;
            color: #fff;
            background-color: #141d2b;
        }
    </style>
</head>
    <body>
        这里是网页内容。
    </body>
</html>
```

加入了JavaScript代码

```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8" />
    <title>这里是页面标题</title>
    <style type="text/css">
        body{
            margin: 0;
            padding: 0;
            color: #fff;
            background-color: #141d2b;
        }
    </style>
</head>
    <body>
        这里是网页内容。
    </body>
    <script type="text/javascript">
        test();
        function test(){
            alert("spiderbuf.cn");
        }
    </script>
</html>
```

让网页跑起来 - Web服务器Nginx的搭建：

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的，公开版本1.19.6发布于2020年12月15日。

```
server {
    listen       80;
    server_name  localhost;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }
```

爬取第一个网页，学习Python发送HTTP请求及处理：爬虫分为向服务器请求数据、解析网页内容、保存内容三个步骤，难点在于请求数据及解析网页内容上。

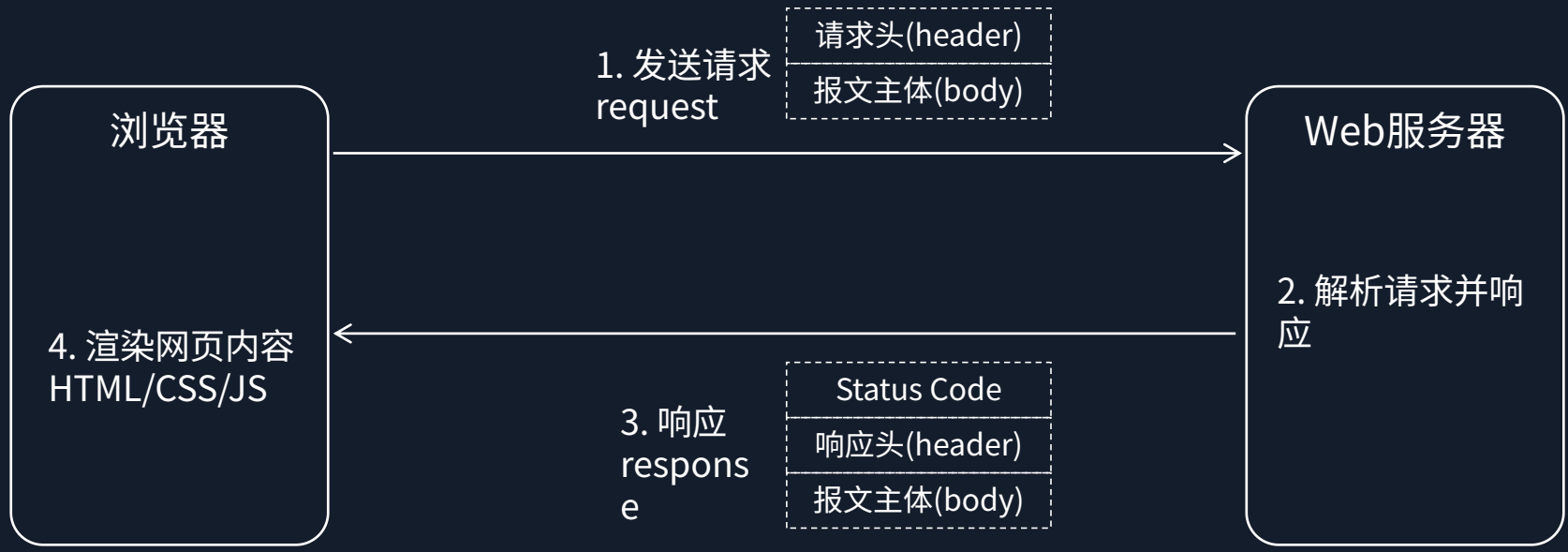
HTTP是基于客户/服务器模式，且面向连接的。典型的HTTP事务处理有如下的过程：

（1）客户与服务器建立连接；

（2）客户向服务器提出请求；

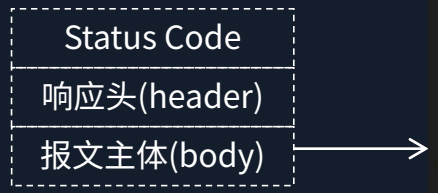
（3）服务器接受请求，并根据请求返回相应的文件作为应答；

（4）客户与服务器关闭连接。



注意：这里的body指的是HTTP的主体，不是HTML文件里的<body></body>

3. 响应 response




```html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="utf-8" />
    <title>这里是页面标题</title>
</head>
    <body>
        这里是网页内容。
    </body>
</html>
```

查找定位网页内容的方法：

selector

JS path

XPath

```python
#coding=utf-8
import requests
from lxml import etree

my_content = requests.get('http://localhost') # 这里返回的是对象
print(my_content.status_code)
if my_content:
    html_bytes = my_content.content
    html = html_bytes.decode()
    # print(html)
    root = etree.HTML(html)
    bodys = root.xpath("body")
    if len(bodys) > 0:
        print(bodys[0].text)
else:
    print('error: content was empty.')
```

User-Agent：最常见的反爬措施，也是最容易绕过的，了解检测原理及绕过方法。

Cookie：Cookie的原理及在反爬虫中的应用。

Referrer：为什么经常爬不了图片 - 图片防盗链原理。

Authorisation：身份认证。

君子协议：robots.txt

```
location /sub.html{
    #禁止指定UA及UA为空的访问
    if ($http_user_agent ~ "python-requests|Python-urllib|^$" )
    {
        return 403;
    }
    valid_referers localhost;
    if ($invalid_referer) {
        return 403;
    }
    root html;
}
```

使用Python flask库编写第一个动态网页

命令行执行：pip install flask

```python
#coding=utf-8
# 导入Flask模块
from flask import Flask
# 创建一个Flask应用
app = Flask(__name__)

# 定义一个路由/，接口返回文本'spiderbuf'
@app.route('/')
def index():
    return 'spiderbuf'

# 运行程序
if __name__ == '__main__':
    app.run()
```

```python
#coding=utf-8
# 导入Flask模块
from flask import Flask, render_template
# 创建一个Flask应用
app = Flask(__name__)

# 定义一个路由/，接口返回网页
@app.route('/')
def index():
    return render_template('index.html')

# 运行程序
if __name__ == '__main__':
    app.run()
```

```python
#coding=utf-8
import requests

my_content = requests.get('http://localhost:5000') # 这里返回的是对象
print(my_content.status_code)
if my_content:
    html_bytes = my_content.content
    html = html_bytes.decode()
    print(html)
else:
    print('error: content was empty.')
```

对第8章节的动态网页进行升级，输出接收到的请求内容。

```python
# coding=utf-8
# 导入Flask模块
from flask import Flask, render_template, request

# 创建一个Flask应用
app = Flask(__name__)

# 定义一个路由/，接口返回网页
@app.route('/')
def index():
    print(request.headers)
    print('----------------------------------------\n')
    return render_template('index.html')


# 运行程序
if __name__ == '__main__':
    app.run()
```

```python
#coding = utf-8
import requests

my_headers = {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.3
              "Referer":"http://localhost"}
my_content = requests.get('http://localhost:5000', headers=my_headers) # 这里返回的是对象
print(my_content.status_code)
if my_content:
    html_bytes = my_content.content
    html = html_bytes.decode()
    print(html)
else:
    print('error: content was empty.')
```

第一轮章节涉及的知识点：

- 网页HTML的基本结构
- CSS的基础知识
- JavaScript的基础知识
- Web服务器的基础知识
- flask开发网页的基础知识
- HTTP的基础原理
- XPath的基本用法

到了这里，你已经成为了一名既能用Python做Web开发，又能用Python写爬虫，还会用Nginx部署Web应用的开发者。

接下来，我们一起横向拓展知识点并加深了解，成上一个台阶。。。

DevTools协议：

通常是指浏览器开发工具，例如Chrome DevTools、Firefox DevTools等，它们提供了一组工具和功能，用于调试和分析网页。这些工具本身并不是一个统一的协议，而是基于一些通信协议和标准构建的。例如，Chrome DevTools Protocol (CDP) 是 Chrome DevTools 的基础，它定义了一组用于与浏览器通信的协议。因此，可以说 DevTools 是建立在一系列协议之上的工具集合。

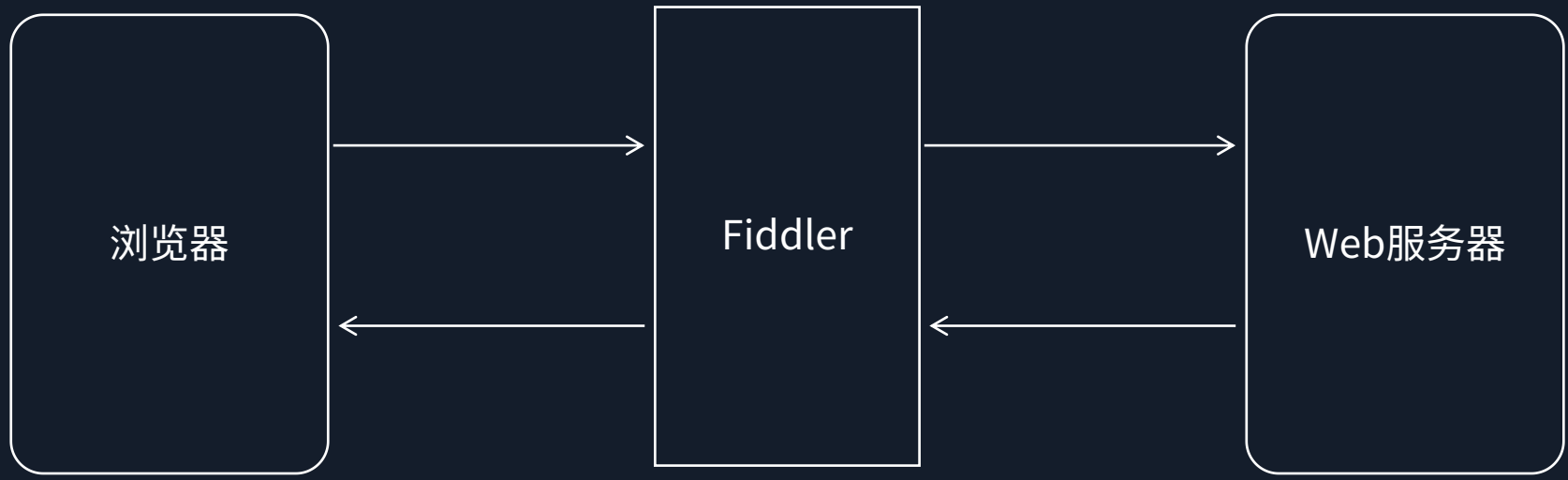
应用场景：

- 在浏览器中调试网页；
- 通过模拟浏览器及该协议执行复杂的爬虫动作。

练习爬虫中的苦恼：是我的代码语法错误还是构造的请求不对?

Fiddler是一个HTTP协议调试代理工具：

- 分析HTTP报文
- 模拟HTTP请求



CSS：

层叠样式表(英文全称：Cascading Style Sheets)不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。

CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。

JavaScript：

JavaScript在1995年由Netscape公司的Brendan Eich，在网景导航者浏览器上首次设计实现而成。因为Netscape与Sun合作，Netscape管理层希望它外观看起来像Java，因此取名为JavaScript。但实际上它的语法风格与Self及Scheme较为接近。

获取JavaScript访问后台API接口的动态数据

方法一：分析出接口的访问方式并使用Python访问；

方法二：使用模拟浏览器打开页面获取； pip install selenium

- JavaScript基础语法
- fetch

fetch() 是 ECMAScript6 (ES6) 特性，返回解析为 Response 对象的 Promise。

Promise 是一个 ECMAScript 6 提供的类，目的是更加优雅地书写复杂的异步任务。

由于 Promise 是 ES6 新增加的，所以一些旧的浏览器并不支持，苹果的 Safari 10 和 Windows 的 Edge 14 版本以上浏览器才开始支持 ES6 特性。

常见的加密算法及编码：

- Base64
- MD5
- SHA1
- SHA256
- AES
- CRC

为了增加哈希值碰撞的难度，通常会加盐值（salt）。

通常会通过在前端获取时间戳、随机数等方式，使用与后台约定的规则与哈希算法生成签名，后台接收到前端请求后第一时间校验签名，如果签名不符则拒绝访问。

Selenium是怎么被识别到的：

pip install selenium

- navigator.webdriver
- navigator.plugins.length
- headless
- automation

浏览器指纹通过收集各种公开的浏览器和设备属性来创建一个独特的“指纹”。这些属性可以包括但不限于：

- 浏览器类型和版本：如Chrome, Firefox, Safari等及其具体版本号。
- 操作系统：如Windows, macOS, Linux等。
- 浏览器插件：已安装的插件及其版本。
- 屏幕分辨率和颜色深度。
- 时区。
- 语言和区域设置。
- 用户代理字符串：包含浏览器、操作系统和设备信息。
- HTTP头信息：包括接受的语言、编码等。
- 字体列表：浏览器中可用的字体。
- 硬件信息：如GPU型号。
- Canvas指纹：利用HTML5 Canvas绘图特性，生成独特的图像数据。
- WebGL指纹：利用WebGL生成的图像数据。
- 媒体设备信息：如可用的音频和视频设备。

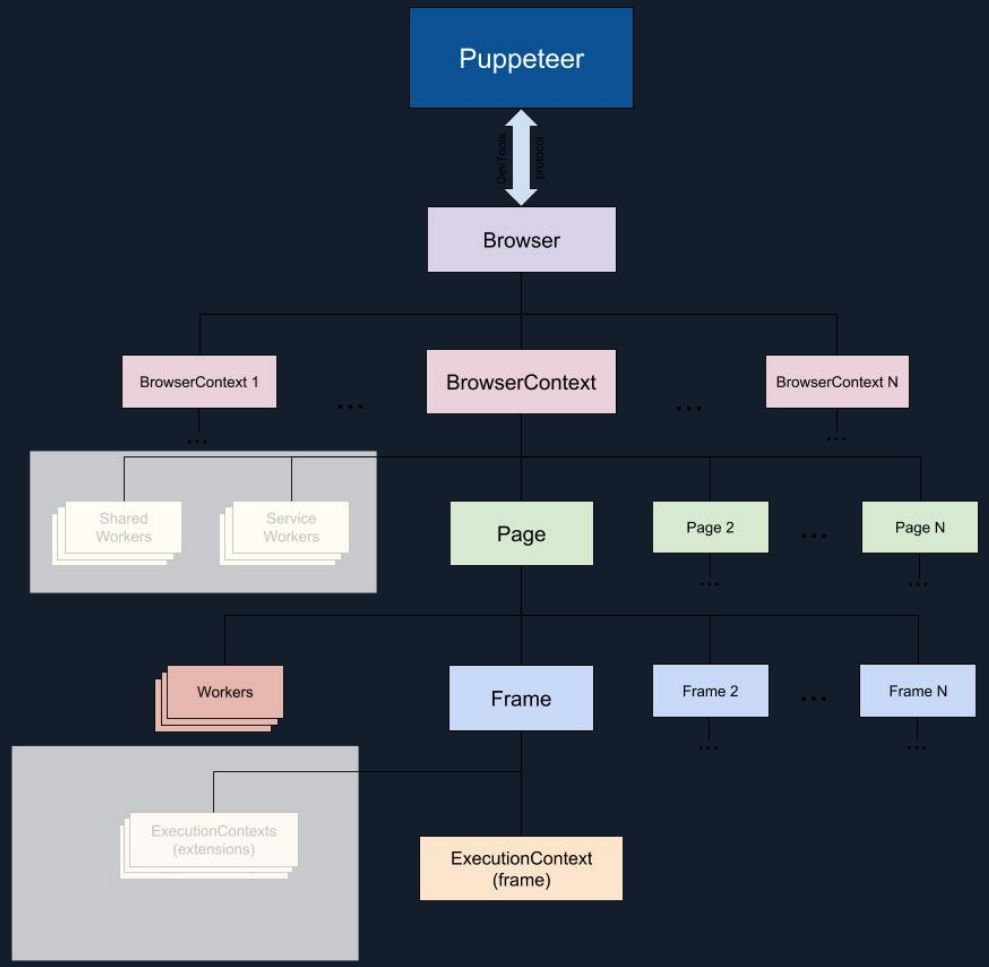
JavaScript做爬虫的优势：

JavaScript（简称“JS”）是一种具有函数优先的轻量级，解释型或即时编译型的编程语言。虽然它是作为开发Web页面的脚本语言而出名，但是它也被用到了很多非浏览器环境中，JavaScript基于原型编程、多范式的动态脚本语言，并且支持面向对象、命令式、声明式、函数式编程范式。

JavaScript的标准是ECMAScript。截至2012年，所有浏览器都完整的支持ECMAScript 5.1，旧版本的浏览器至少支持ECMAScript 3标准。2015年6月17日，ECMA国际组织发布了ECMAScript的第六版，该版本正式名称为ECMAScript 2015，但通常被称为ECMAScript 6或者ES2015。

提供了一个高级 API 来通过 DevTools 协议控制 Chromium 或 Chrome。

npm i puppeteer



第二轮章节涉及的知识点：

- DevTools协议
- 抓包工具Fiddler的使用
- CSS与JavaScript对网页内容的影响
- JavaScript动态获取数据
- 常见加密算法及编码
- Selenium的攻防
- 浏览器指纹
- 深入了解JavaScript
- Node.js库Puppeteer

到了这里，你已经学习了很多关于网页与爬虫的相关知识，具备了成为独立爬虫开发者的能力。

学到这里你已经很厉害了，我已经没什么可教的了，接下来我们过几招你就下山去闯荡江湖吧。。。

涉及的知识点：

- 前端表单开发及提交
- HTTP POST 请求的发送及处理
- Cookie生成、传递及验证

涉及的知识点：

- JavaScript时间戳及哈希算法
- Ajax提交并处理响应结果
- Python时间戳及哈希算法

涉及的知识点：

- JavaScript检测用户行为
- 网页鼠标事件检测