**从零开始编写一个网页数据爬虫**

# 网页相关基础知识

## 爬虫的基本原理和过程

**模拟手动操作浏览器从网络获取信息。**

### ****什么是爬虫？****

爬虫（Crawler），又称网络爬虫或蜘蛛，是一种自动化程序，用来在互联网上浏览并抓取网页数据。爬虫的工作类似于网络上的“蜘蛛”，它会自动地在互联网上“爬行”，抓取网页内容，通常是按一定的规则进行。

例如，搜索引擎中的爬虫（如 Googlebot）会定期抓取网页，并将抓取的内容建立索引，供用户查询。

### ****爬虫的基本原理****

爬虫的核心原理是模拟用户访问网页，通过发起 HTTP 请求获取网页内容，然后对这些内容进行解析和存储。它的工作流程主要包括以下几个步骤：

1. **发起请求（Request）**

爬虫首先向目标网站发起 HTTP 请求。HTTP 请求包含了爬虫需要访问的网页地址（URL），并通过浏览器发送请求。爬虫通常会使用 GET 方法来获取网页内容，也可以使用 POST 方法提交表单数据。

1. **获取响应（Response）**

目标网站接收到爬虫的请求后，返回一个 HTTP 响应，通常是网页的 HTML 内容。这个响应中可能包含了网页的所有资源（文本、图片、JavaScript 等）。

1. **解析网页内容**

爬虫从网页的响应中获取 HTML 内容后，需要解析这些数据。解析的方式通常是通过正则表达式、XPath、CSS 选择器等技术，提取网页中的关键信息，如文章标题、链接、图片 URL 等。

1. **数据存储**

解析完网页中的有用数据后，爬虫将提取的数据存储到本地或数据库中，以便后续分析和处理。常见的存储方式包括文本文件、CSV 文件、JSON 文件，或者直接存入数据库（如 MySQL、MongoDB 等）。

1. **遍历与抓取（递归）**

爬虫的任务并不是一次性抓取单个网页，它通常会根据网页中的链接继续抓取其他相关网页，形成一个遍历的过程。爬虫会从一个网页开始，提取页面上的链接，并继续访问这些链接。

### ****常见的爬虫工具与技术****

1. **Requests：** 一个 Python 库，用于发送 HTTP 请求，非常适合爬虫中的请求发送部分。简单易用，常用于发送 GET 或 POST 请求。
2. **BeautifulSoup：** 用于解析 HTML 和 XML 文件。它能够非常方便地从网页中提取数据，支持 CSS 选择器和 XPath。
3. **Selenium：** 一个自动化测试工具，可以模拟用户与网页的交互，适用于动态加载内容的网页。它能够处理 JavaScript 渲染、点击按钮等操作。
4. **Scrapy：** 一个 Python 的爬虫框架，功能非常强大，适合大规模的数据抓取。Scrapy 提供了丰富的工具集，可以自动化地管理请求、解析响应、存储数据等。
5. **Puppeteer：** 基于 Node.js 的一个爬虫框架，适合抓取动态加载的页面，可以模拟浏览器操作，支持 JavaScript 渲染。

### ****爬虫的常见挑战与应对策略****

1. **反爬机制：** 许多网站会采取反爬措施，例如检测请求的频率、IP 封禁、验证码等。应对策略包括使用代理 IP、调整请求频率、模拟用户行为等。
2. **动态加载：** 一些网站内容通过 JavaScript 动态加载，爬虫抓取时可能只获得部分内容。应对策略是使用 Selenium 或 Puppeteer 等工具，模拟浏览器环境来加载 JavaScript。
3. **数据存储：** 在大规模抓取时，如何高效地存储数据成为问题。可以使用数据库（如 MySQL、MongoDB）来存储和管理大规模的抓取数据。
4. **网页结构变化：** 网页的 HTML 结构可能会发生变化，这会导致爬虫的解析失败。应对策略是使用容错设计，定期检查和更新爬虫的解析逻辑。

## HTML基础

### HTML结构

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>网页标题</title>**

**</head>**

**<body>**

**<div class="container">**

**<h1>这是标题</h1>**

**<p>这是一个段落。</p>**

**<a href="https://example.com">点击这里</a>**

**<img src="image.jpg" alt="图片描述">**

**</div>**

**</body>**

**</html>**

### HTML 标签

* **<html>：网页的根标签，表示整个网页。**
* **<head>：包含页面的元数据（如字符集、页面标题、链接、脚本等）。**
* **<body>：包含网页内容，所有页面显示的内容都在 <body> 标签内。**
* **<div>：用于布局的容器元素，通常用于分块网页内容。**
* **<span>：行内元素，通常用于包装文本或小的块级内容。**
* **<a>：超链接元素，用于跳转到其他网页或资源。**
* **<img>：图像元素，用于展示图片。**
* **<p>：段落元素，用于显示文本段落。**
* **<h1>, <h2>, <h3> 等：标题元素，用于表示页面的标题（<h1> 为最大的标题）。**
* **<table>：表格元素，用于展示表格数据。**
* **<ul>、<ol>、<li>：无序列表、有序列表和列表项。**

### JavaScript 或 AJAX

JavaScript 用于添加交互性和动态效果，通常在用户与网页交互时执行，例如点击按钮、提交表单或加载新内容。很多现代网站使用 JavaScript 或 Ajax 加载动态数据，比如商品信息、评论或分页内容。

1. **JavaScript** 会修改页面的 DOM（文档对象模型）结构，爬虫抓取页面时，若页面内容由 JavaScript 动态生成，则需要使用浏览器模拟工具（如 Selenium 或 Puppeteer）来抓取数据。
2. **AJAX**（异步 JavaScript 和 XML）用于在后台异步加载数据，常见的请求方式有 GET 和 POST。AJAX 请求返回的数据通常是 JSON 格式，爬虫可以通过模拟这些请求来获取数据。

# 爬虫的编写

从CNKI里面爬取论文题目/作者/发表的期刊。

输出结果

翻页操作

提取文献数据

进行文献搜索

进入 CNKI 网站

登录操作

初始化 Selenium WebDriver

# 爬虫与数据库