**网络爬虫基础概念**

**核心功能模块**

​URL管理器​：维护待抓取队列和已抓取集合（常用去重算法：Bloom Filter）

​网页下载器​：通过HTTP协议获取网页内容（GET/POST请求）

​网页解析器​：提取数据（XPath/CSS选择器/正则表达式）和新URL

​数据存储器​：结构化存储到数据库/文件（MySQL/MongoDB/CSV）

**流程：**

[种子URL] → URL队列 → 下载网页 → 解析数据 → 存储数据

↑ ↓ ↓

└── 新URL发现 ← 清洗URL ──┘

**防爬机制**

User-Agent检测 验证请求头是否包含浏览器特征 随机切换User-Agent池

IP封禁 监测单个IP请求频率 代理IP池（免费/付费代理服务）

验证码 图形/滑动/点选验证码 OCR识别/Selenium模拟操作

动态渲染 JavaScript生成核心内容 Headless浏览器（Puppeteer）

请求参数加密 接口参数使用Token或加密算法 逆向工程JS代码

**底层技术原理**

HTTP协议层

请求方法​：GET（获取资源）/POST（提交数据）

状态码​：200（成功）、301/302（重定向）、403（禁止访问）

数据解析技术：

​DOM树解析​：通过浏览器渲染引擎生成文档对象模型

动态内容处理​：

拦截Ajax请求（Chrome DevTools协议）

解析JSON API数据（如GraphQL接口

**哈希算法**分布式爬虫架构

快速URL去重 MurmurHash 速度极快，碰撞率低

高可靠性内容校验 SHA-256 抗碰撞性强，安全性高

相似内容检测 SimHash 支持汉明距离计算

内存敏感型场景 Bloom Filter 位图压缩存储，1亿URL仅2MB内存

**异步加载的原理**

典型场景

页面初始HTML不包含数据

通过JavaScript发起API请求（如fetch()或XMLHttpRequest）

动态插入DOM元素（如滚动加载、点击加载更多）

识别方法

浏览器开发者工具 → Network → XHR/JS标签页

查找返回JSON数据的请求（通常为GET/POST）

解决方案

​方法​ ​ 适用场景​ ​ 优点​ ​ 缺点​

直接调用API 数据接口规范且未加密 速度快、资源消耗低 需逆向工程接口参数

无头浏览器渲染 复杂渲染逻辑或接口加密 能处理所有动态内容 速度慢、内存占用高

拦截网络请求 需捕获特定API请求 精准获取数据 需维护请求监听逻辑

Hybrid模式（API+渲染） 部分数据在API，部分需渲染 平衡效率与覆盖率 实现复杂度较高

**无头浏览器（Headless Browser）​​ 是一种没有图形用户界面（GUI）的浏览器，能够通过代码完全控制网页加载、渲染和交互**

常见无头浏览器​：

​Headless Chrome​（通过Chrome DevTools协议控制）

​Headless Firefox​

​Puppeteer​（Node.js库，封装Chrome）

​Playwright​（微软出品，支持多浏览器）

**为什么爬虫需要无头浏览器？**

**​动态网页的挑战​：**

​JavaScript渲染​：现代网站（如React/Vue单页应用）依赖JS生成内容。

​异步加载​：通过Ajax/XHR动态获取数据（如无限滚动、懒加载）。

​用户交互依赖​：需要点击/滚动才能触发数据加载。

**无头浏览器爬虫的核心能力**

​能力​ ​ 应用场景​ ​ 示例代码（Python + Playwright）​​

​完整页面渲染​ 获取JS动态生成的内容 page.goto(url) → page.content()

​模拟用户操作​ 点击按钮、输入文本、滚动页面 page.click("#load-more") → page.type("#search", "关键词")

​拦截网络请求​ 直接捕获API返回的JSON数据 page.on("response", lambda res: print(res.json()))

​处理复杂认证​ 自动填充登录表单、绕过验证码 page.fill("#username", "user") → page.screenshot()

​生成页面快照​ 调试时保存渲染后的页面状态 page.screenshot(path="page.png")