网络爬虫简介

爬虫与反爬技术

目录

爬虫技术

- 数据采集流程
- 网页分析
- 动态页面抓取
- 可视化爬虫
- 爬虫框架
- 案例

反爬技术

- 反爬技术及解决方案

反爬技术

- User_agent判断来源
- 根据IP访问频率判断，封锁IP或者账号
- 验证码识别
  - Pillow库
  - Tesseract库
  - Numpy
  - 机器学习
  - 根据实际成本制定策略
- 频繁变更网页结构
- 非正常请求提供虚假信息（隐含输入字段或缺失参数）

谢谢!

2018.01.01

前言

- **robots.txt: Robots**协议被称为爬虫协议，或机器人协议。是国际互联网界通行的道德规范。
  - User-agent:用于描述搜索引擎robot的名字;
  - Disallow:禁止robot访问该网站的目录或文件;
  - Allow:允许robot访问该网站的目录或文件;
- **sitemap.xml/txt/html/...: Sitemaps**协议使网站能够告知搜索引擎网站中可供抓取的网址。
  - 自动生成:更方便地了解一个网站的内容、布局、架构。
  - 主动提交:向百度、Google、雅虎、和微软等提交，被搜索引擎收录。

数据采集流程

- 需求分析：业务类型/方向决定抓取策略和抓取频率。（干什么用）
- 抓取内容：具体需要什么数据/字段。（怎么用）
- 数据来源：数据来源于具体网站或其他。
- 抓取方式：获取数据的方式、网页结构分析、API等。
- 代码实现：自定义或开源爬虫框架。
- 数据清洗：根据业务具体情况转换数据格式、类型，进行数据计算等。数据清洗分为入库前点清洗和入库后点清洗。
- 数据存储：写文件或入库方式，远程数据入库，数据加密等。
- 爬虫任务分发：分布式爬虫、多线程爬虫。
- 反爬策略、日志管理、监控报警等。

网页分析

- 数据抓取方式
  - 网页
  - 移动网页
  - 移动客户端
  - API
- 网页解析
  - 正则
  - Xpath
  - Beautiful Soup
  - Json格式解析
- 网页编码

动态页面抓取

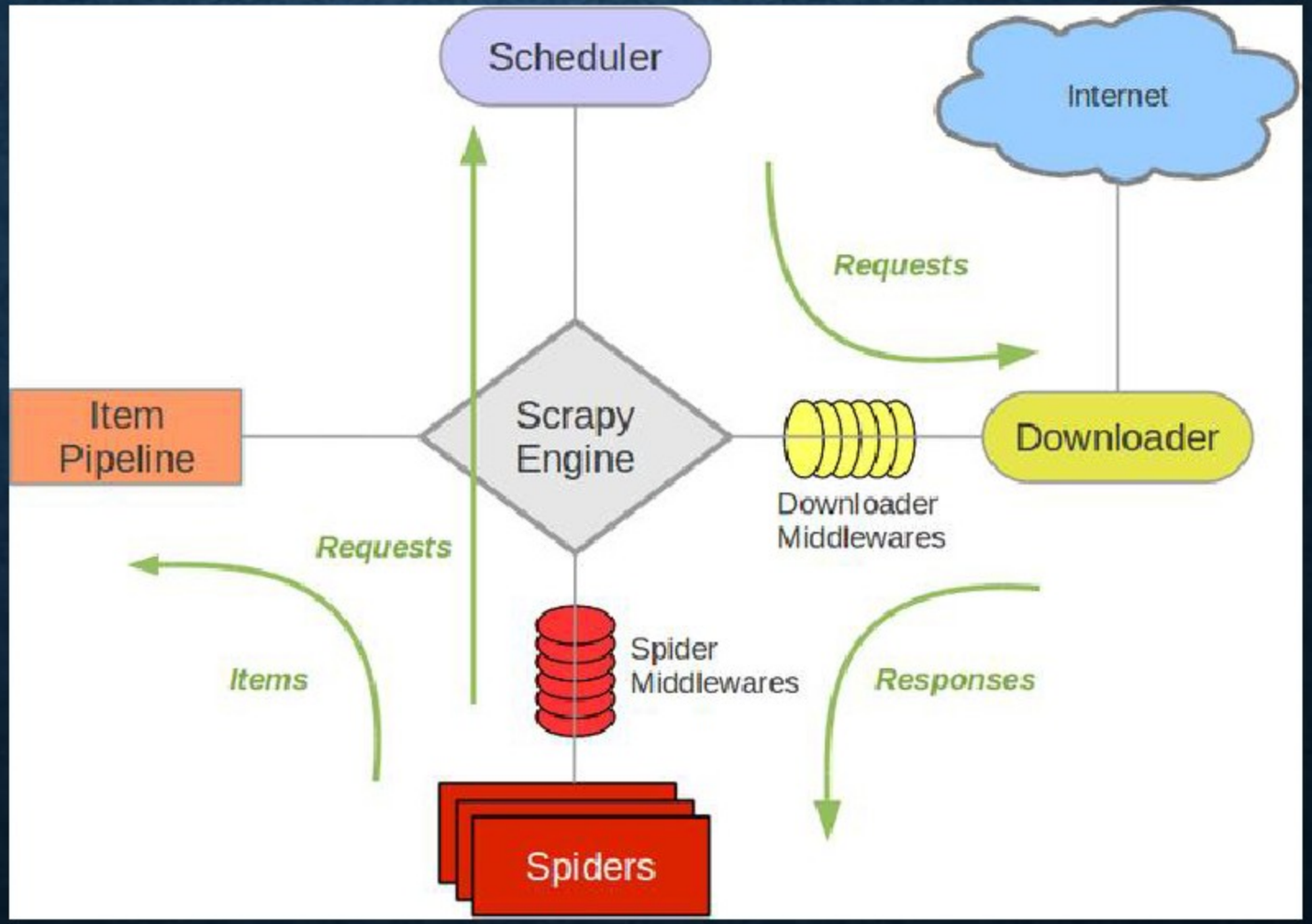
- JS渲染页面
  - Web kit
  - Render类
- Ajax请求
- Iframe
- Selenium库
- 重定向
  - Status_code
  - Html meta 的refresh
  - js

可视化爬虫

- Import.io
- Portia
- 八爪鱼
- 集搜客
- 造数
- BBD

爬虫框架

- Scrapy框架：



案例1：58同城-简历中心

- url：http://cd.58.com/qzzpshengchankaifa/?key=%E7%AE%80%E5%8E%86%E4%B8%AD%E5%BF%83

案例2：阿里巴巴-搜索

- url：http://m.1688.com/offer_search/-cee4baeec7f8.html
- Set-Cookie
- _csrf