**WebMagic in Action**

Little book of WebMagic.



[WebMagic](https://github.com/code4craft/webmagic)是我业余开发的一款简单灵活的爬虫框架。基于它你可以很容易的编写一个爬虫。

这本小书以WebMagic入手，一方面讲解WebMagic的使用方式，另一方面讲解爬虫开发的一些惯用方案。

文章预览请点<http://webmagic.io/docs/>，页面基于[gitbook](https://github.com/GitbookIO/gitbook/)进行构建。

本文档遵循CC-BYNC协议。

### 1.1 设计思想



#### 1. 一个框架，一个领域

一个好的框架必然凝聚了领域知识。WebMagic的设计参考了业界最优秀的爬虫Scrapy，而实现则应用了HttpClient、Jsoup等Java世界最成熟的工具，目标就是做一个Java语言Web爬虫的教科书般的实现。

如果你是爬虫开发老手，那么WebMagic会非常容易上手，它几乎使用Java原生的开发方式，只不过提供了一些模块化的约束，封装一些繁琐的操作，并且提供了一些便捷的功能。

如果你是爬虫开发新手，那么使用并了解WebMagic会让你了解爬虫开发的常用模式、工具链、以及一些问题的处理方式。熟练使用之后，相信自己从头开发一个爬虫也不是什么难事。

因为这个目标，WebMagic的核心非常简单——在这里，功能性是要给简单性让步的。

#### 2. 微内核和高可扩展性

WebMagic由四个组件(Downloader、PageProcessor、Scheduler、Pipeline)构成，核心代码非常简单，主要是将这些组件结合并完成多线程的任务。这意味着，在WebMagic中，你基本上可以对爬虫的功能做任何定制。

WebMagic的核心在webmagic-core包中，其他的包你可以理解为对WebMagic的一个扩展——这和作为用户编写一个扩展是没有什么区别的。

#### 3. 注重实用性

虽然核心需要足够简单，但是WebMagic也以扩展的方式，实现了很多可以帮助开发的便捷功能。例如基于注解模式的爬虫开发，以及扩展了XPath语法的Xsoup等。这些功能在WebMagic中是可选的，它们的开发目标，就是让使用者开发爬虫尽可能的简单，尽可能的易维护。

### 1.2 总体架构

WebMagic的结构分为Downloader、PageProcessor、Scheduler、Pipeline四大组件，并由Spider将它们彼此组织起来。这四大组件对应爬虫生命周期中的下载、处理、管理和持久化等功能。WebMagic的设计参考了Scapy，但是实现方式更Java化一些。

而Spider则将这几个组件组织起来，让它们可以互相交互，流程化的执行，可以认为Spider是一个大的容器，它也是WebMagic逻辑的核心。

WebMagic总体架构图如下：

### 1.2.1 WebMagic的四个组件

#### 1.Downloader

Downloader负责从互联网上下载页面，以便后续处理。WebMagic默认使用了[Apache HttpClient](http://hc.apache.org/index.html)作为下载工具。

#### 2.PageProcessor

PageProcessor负责解析页面，抽取有用信息，以及发现新的链接。WebMagic使用[Jsoup](http://jsoup.org/)作为HTML解析工具，并基于其开发了解析XPath的工具[Xsoup](https://github.com/code4craft/xsoup)。

在这四个组件中，PageProcessor对于每个站点每个页面都不一样，是需要使用者定制的部分。

#### 3.Scheduler

Scheduler负责管理待抓取的URL，以及一些去重的工作。WebMagic默认提供了JDK的内存队列来管理URL，并用集合来进行去重。也支持使用Redis进行分布式管理。

除非项目有一些特殊的分布式需求，否则无需自己定制Scheduler。

#### 4.Pipeline

Pipeline负责抽取结果的处理，包括计算、持久化到文件、数据库等。WebMagic默认提供了“输出到控制台”和“保存到文件”两种结果处理方案。

Pipeline定义了结果保存的方式，如果你要保存到指定数据库，则需要编写对应的Pipeline。对于一类需求一般只需编写一个Pipeline。

### 1.2.2 用于数据流转的对象

#### 1. Request

Request是对URL地址的一层封装，一个Request对应一个URL地址。

它是PageProcessor与Downloader交互的载体，也是PageProcessor控制Downloader唯一方式。

除了URL本身外，它还包含一个Key-Value结构的字段extra。你可以在extra中保存一些特殊的属性，然后在其他地方读取，以完成不同的功能。例如附加上一个页面的一些信息等。

#### 2. Page

Page代表了从Downloader下载到的一个页面——可能是HTML，也可能是JSON或者其他文本格式的内容。

Page是WebMagic抽取过程的核心对象，它提供一些方法可供抽取、结果保存等。在第四章的例子中，我们会详细介绍它的使用。

#### 3. ResultItems

ResultItems相当于一个Map，它保存PageProcessor处理的结果，供Pipeline使用。它的API与Map很类似，值得注意的是它有一个字段skip，若设置为true，则不应被Pipeline处理。

### 1.2.3 控制爬虫运转的引擎--Spider

Spider是WebMagic内部流程的核心。Downloader、PageProcessor、Scheduler、Pipeline都是Spider的一个属性，这些属性是可以自由设置的，通过设置这个属性可以实现不同的功能。Spider也是WebMagic操作的入口，它封装了爬虫的创建、启动、停止、多线程等功能。下面是一个设置各个组件，并且设置多线程和启动的例子。详细的Spider设置请看第四章——[爬虫的配置、启动和终止](http://webmagic.io/docs/zh/posts/ch4-basic-page-processor/spider-config.html)。

public static void main(String[] args) {

Spider.create(new GithubRepoPageProcessor())

//从https://github.com/code4craft开始抓

.addUrl("https://github.com/code4craft")

//设置Scheduler，使用Redis来管理URL队列

.setScheduler(new RedisScheduler("localhost"))

//设置Pipeline，将结果以json方式保存到文件

.addPipeline(new JsonFilePipeline("D:\\data\\webmagic"))

//开启5个线程同时执行

.thread(5)

//启动爬虫

.run();

}

### 1.2.4 快速上手

上面介绍了很多组件，但是其实使用者需要关心的没有那么多，因为大部分模块WebMagic已经提供了默认实现。

一般来说，对于编写一个爬虫，PageProcessor是需要编写的部分，而Spider则是创建和控制爬虫的入口。在第四章中，我们会介绍如何通过定制PageProcessor来编写一个爬虫，并通过Spider来启动。

### 1.3 项目组成

WebMagic项目代码包括几个部分，在根目录下以不同目录名分开。它们都是独立的Maven项目。

### 1.3.1 主要部分

WebMagic主要包括两个包，这两个包经过广泛实用，已经比较成熟：

#### webmagic-core

WebMagic-core是WebMagic核心部分，只包含爬虫基本模块和基本抽取器。WebMagic-core的目标是成为网页爬虫的一个教科书般的实现。

#### webmagic-extension

webmagic-extension是WebMagic的主要扩展模块，提供一些更方便的编写爬虫的工具。包括注解格式定义爬虫、JSON、分布式等支持。

### 1.3.2 外围功能

除此之外，WebMagic项目里还有几个包，这些都是一些实验性的功能，目的只是提供一些与外围工具整合的样例。因为精力有限，这些包没有经过广泛的使用和测试，推荐使用方式是自行下载源码，遇到问题后再修改。

#### webmagic-samples

这里是作者早期编写的一些爬虫的例子。因为时间有限，这些例子有些使用的仍然是老版本的API，也可能有一些因为目标页面的结构变化不再可用了。最新的、精选过的例子，请看webmaigc-core的us.codecraft.webmagic.processor.example包和webmaigc-core的us.codecraft.webmagic.example包。

#### webmagic-scripts

WebMagic对于爬虫规则脚本化的一些尝试，目标是让开发者脱离Java语言，来进行简单、快速的开发。同时强调脚本的共享。

目前项目因为感兴趣的用户不多，处于搁置状态，对脚本化感兴趣的可以看这里：[webmagic-scripts简单文档](https://github.com/code4craft/webmagic/tree/master/webmagic-scripts)

#### webmagic-selenium

WebmMgic与Selenium结合的模块。Selenium是一个模拟浏览器进行页面渲染的工具，WebMagic依赖Selenium进行动态页面的抓取。

#### webmagic-saxon

WebMagic与Saxon结合的模块。Saxon是一个XPath、XSLT的解析工具，webmagic依赖Saxon来进行XPath2.0语法解析支持。

### 1.3.3 webmagic-avalon

webmagic-avalon是一个特殊的项目，它想基于WebMagic实现一个产品化的工具，涵盖爬虫的创建、爬虫的管理等后台工具。[Avalon](http://zh.wikipedia.org/wiki/%E9%98%BF%E7%93%A6%E9%9A%86)是亚瑟王传说中的“理想之岛”，webmagic-avalon的目标是提供一个通用的爬虫产品，达到这个目标绝非易事，所以取名也有一点“理想”的意味，但是作者一直在朝这个目标努力。

对这个项目感兴趣的可以看这里[WebMagic-Avalon项目计划](https://github.com/code4craft/webmagic/issues/43)。