

**基于Scrapy的模板化爬虫程序管理平台**

**软件需求规格说明书**

v1.2.2

赵正阳 SY1906428

郭浩隆 SY1906430

沈一聪 SY1906510

梁远志 SY1906503

宋冰晨 SY1906429

2020年4月9日

**版本变更记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 修改人 | 修改章节 | 备注 |
| 1.0 | 2020/3/25 | 赵正阳 | 1、2.1、2.2.1、3.1.1、3.2.1、3.3.1、3.5、3.7、4.2、4.3 | 完成引言，系统模型-Scrapy框架，Scrapy框架的需求分析、用例图和RUCM图，输入和输出，故障处理，运行环境，整合初稿 |
| 郭浩隆 | 3.3.2 | 绘制Web UI的RUCM图 |
| 沈一聪 | 3.1.3、3.2.2、3.2.3、3.3.3、3.4.2、 | 绘制Web UI和后端的用例图、后端的RUCM图、非功能需求反爬虫 |
| 梁远志 | 2.2.2、3.1.4、3.2.4、3.3.4、3.6、4.1、4.4 | 绘制EasySpider系统架构图，Scrapyd的需求分析、用例图和RUCM图，数据库特性 |
| 宋冰晨 | 3.1.2、3.4.1 | 完成Web UI功能需求和非功能需求的文字描述 |
| 1.1 | 2020/3/31 | 赵正阳 | 3.2.1、3.3.1 | 补充RUCM图文字描述、Scrapy框架非功能需求，整合及调整目录结构 |
| 郭浩隆 | 3.2.2 | 补充RUCM图文字描述 |
| 沈一聪 | 3.1.2、3.2.3 | 补充后端业务需求、RUCM图文字描述 |
| 梁远志 | 3.2.4 | 补充RUCM图文字描述 |
| 宋冰晨 | 1.2、2.2、2.4 | 补充项目背景、用户特点、假定和约束 |
| 1.2.1 | 2020/4/8 | 赵正阳 | 1.3、1.4、2.3、3.2.1、4.2 | 补充了一些术语和缩略语；规范参考资料格式；修正标点；修正用例图和RUCM图；补充支持软件的版本号 |
| 1.2.2 | 2020/4/9 | 宋冰晨 | 1.2、1.4、2.2、2.4、3.1.1、3.3 | 根据A组和I组的部分建议进行修改，另扩充了假定和约束，以及调整增加了Web UI部分的非功能需求 |

**目录**

[1. 引言 - 1 -](#_Toc37280513)

[1.1 编写目的 - 1 -](#_Toc37280514)

[1.2 背景 - 1 -](#_Toc37280515)

[1.3 术语和缩略语 - 2 -](#_Toc37280516)

[1.4 参考资料 - 3 -](#_Toc37280517)

[2. 任务概述 - 5 -](#_Toc37280518)

[2.1 目标 - 5 -](#_Toc37280519)

[2.2 用户特点 - 5 -](#_Toc37280520)

[2.3 系统模型 - 5 -](#_Toc37280521)

[2.3.1 Scrapy框架 - 5 -](#_Toc37280522)

[2.3.2 EasySpider - 6 -](#_Toc37280523)

[2.4 假定和约束 - 7 -](#_Toc37280524)

[3. 需求规定 - 9 -](#_Toc37280525)

[3.1 业务需求 - 9 -](#_Toc37280526)

[3.1.1 Web UI - 9 -](#_Toc37280527)

[3.1.2 后端及调度程序 - 9 -](#_Toc37280528)

[3.2 功能需求 - 9 -](#_Toc37280529)

[3.2.1 Scrapy框架 - 9 -](#_Toc37280530)

[3.2.1.1 发送请求 - 10 -](#_Toc37280531)

[3.2.1.2 解析页面 - 11 -](#_Toc37280532)

[3.2.1.3 处理项目 - 13 -](#_Toc37280533)

[3.2.1.4 日志输出 - 14 -](#_Toc37280534)

[3.2.1.5 系统设置 - 14 -](#_Toc37280535)

[3.2.1.6 添加Spider中间件 - 15 -](#_Toc37280536)

[3.2.1.7 添加下载器中间件 - 15 -](#_Toc37280537)

[3.2.2 Web UI - 16 -](#_Toc37280538)

[3.2.2.1 浏览爬虫列表 - 17 -](#_Toc37280539)

[3.2.2.2 过滤关键字搜索 - 18 -](#_Toc37280540)

[3.2.2.3 添加爬虫任务 - 18 -](#_Toc37280541)

[3.2.2.4 选择爬虫模板 - 19 -](#_Toc37280542)

[3.2.2.5 配置爬虫方案 - 19 -](#_Toc37280543)

[3.2.2.6 查看爬虫进度 - 20 -](#_Toc37280544)

[3.2.2.7 启动爬虫任务 - 20 -](#_Toc37280545)

[3.2.2.8 暂停爬虫任务 - 21 -](#_Toc37280546)

[3.2.2.9 浏览爬虫结果 - 22 -](#_Toc37280547)

[3.2.2.10 查询爬虫任务 - 22 -](#_Toc37280548)

[3.2.3 后端及调度程序 - 23 -](#_Toc37280549)

[3.2.3.1 管理任务 - 24 -](#_Toc37280550)

[3.2.3.2 新增任务 - 25 -](#_Toc37280551)

[3.2.3.3 挂起任务 - 26 -](#_Toc37280552)

[3.2.3.4 管理服务器节点 - 27 -](#_Toc37280553)

[3.2.3.5 新增服务器节点 - 27 -](#_Toc37280554)

[3.2.3.6 停用服务器节点 - 27 -](#_Toc37280555)

[3.2.4 Scrapyd - 28 -](#_Toc37280556)

[3.2.4.1 查询守护进程状态 - 30 -](#_Toc37280557)

[3.2.4.2 列出可用爬虫 - 30 -](#_Toc37280558)

[3.2.4.3 列出当前作业 - 31 -](#_Toc37280559)

[3.2.4.4 运行作业 - 32 -](#_Toc37280560)

[3.2.4.5 取消作业 - 32 -](#_Toc37280561)

[3.2.4.6 列出项目 - 33 -](#_Toc37280562)

[3.2.4.7 删除项目 - 34 -](#_Toc37280563)

[3.2.4.8 列出项目版本 - 34 -](#_Toc37280564)

[3.2.4.9 添加项目版本 - 35 -](#_Toc37280565)

[3.2.4.10 删除项目版本 - 36 -](#_Toc37280566)

[3.3 非功能需求 - 36 -](#_Toc37280567)

[3.3.1 Scrapy框架 - 36 -](#_Toc37280568)

[3.3.2 Web UI - 37 -](#_Toc37280569)

[3.3.3 后端 - 38 -](#_Toc37280570)

[3.4 输入和输出 - 39 -](#_Toc37280571)

[3.5 数据库特性 - 39 -](#_Toc37280572)

[3.6 故障处理 - 41 -](#_Toc37280573)

[4. 运行环境 - 42 -](#_Toc37280574)

[4.1 设备 - 42 -](#_Toc37280575)

[4.1.1 客户端 - 42 -](#_Toc37280576)

[4.1.2 后端服务器 - 42 -](#_Toc37280577)

[4.1.3 爬虫服务器 - 42 -](#_Toc37280578)

[4.2 支持软件 - 42 -](#_Toc37280579)

[4.3 接口 - 42 -](#_Toc37280580)

[4.3.1 硬件接口 - 42 -](#_Toc37280581)

[4.3.2 软件接口 - 43 -](#_Toc37280582)

[4.3.3 通信接口 - 43 -](#_Toc37280583)

[4.3.4 用户接口 - 43 -](#_Toc37280584)

[4.4 安全和保密 - 43 -](#_Toc37280585)

1. **引言**

1.1 编写目的

本文档详细描述了模板化爬虫程序管理平台的需求，是后续软件开发的依据。本文档的适用读者包括系统的开发人员、测试人员、用户以及项目其他相关人员。

1.2 背景

系统名称：基于Scrapy的模板化爬虫程序管理平台(EasySpider)

开发者：北京航空航天大学计算机学院研究生赵正阳、郭浩隆、沈一聪、梁远志、宋冰晨

Scrapy是使用Python编写的免费开放源代码网络爬虫框架。它最初被设计用于网络抓取，开发者也可以通过调用它提供的API来提取结构性数据，或者在该框架的基础上设计通用的网络爬虫程序。该框架诞生于伦敦的网络聚合和电子商务公司Mydeco，后由Mydeco和Insophia的员工开发和维护。Scrapy首次公开发行是在BSD许可下于2008年8月发布的，里程碑式的1.0版本的发行是在2015年6月。2011年至今，网络爬虫开发和服务公司Scrapinghub成为Scrapy新的官方维护者。

Scrapy项目体系结构围绕Spider构建，Spider是独立的爬虫，具有一组指令。遵循Django等框架，允许其他开发人员重用其代码，从而使构建和扩展大型爬网项目变得更加容易。Scrapy还提供了一个网络爬虫外壳，开发人员可以使用它来测试他们对网站行为的假设。

一些使用Scrapy的知名公司和产品有：Lyst, Parse.ly, Sayone Technologies, Sciences Po Medialab, Data.gov.uk的世界政府数据站点。

Scrapy作为流行的爬虫框架主要有三大优势。

* 快速而强大：开发者只需编写规则以提取数据，然后让Scrapy完成其余工作
* 容易扩展：开发者无需接触核心即可轻松插入新功能
* 兼容性高：框架使用Python编写并可以在Linux, Windows, Mac和BSD上运行

此外，Scrapy还拥有一个非常活跃的社区，在Github上有36.3k个star、8.4k个fork以及1.8k个watcher；在Twitter上有5.1k个粉丝关注；在StackOverflow上有14.7k个问题讨论。

为了便于部署爬虫，Scrapy官方还提供了Scrapyd——一种用于运行Scrapy爬虫的服务程序。它使您可以使用JSON API部署（上载）您的项目并控制爬虫。

但是，Scrapy面向的用户是会编写爬虫代码的开发人员。一些不熟悉Python或Scrapy编程的用户就无法从万维网这个大量信息的载体方便有效地爬取网页并从中提取结构化的数据。考虑到这部分用户，本软件系统试图提供爬虫模板来帮助用户实现数据的可视化自动采集。

1.3 术语和缩略语

本文档涉及的术语和缩略语及其解释如表1.1所示。

表1.1 术语

|  |  |
| --- | --- |
| 术语 | 解释 |
| EasySpider | 本项目（基于Scrapy的模板化爬虫程序管理平台）的英文名称 |
| 爬虫/网络爬虫 | 一种按照一定的规则，自动地抓取万维网信息的程序或者脚本 |
| Scrapy | 一个快速的、高层次的网络爬虫框架 |
| 请求(Request) | 代表HTTP请求，通常在Spider中生成并由Downloader执行，从而生成Response |
| 响应(Response) | 代表HTTP响应，通常由Downloader下载并发送到Spider进行处理 |
| 引擎(Scrapy Engine) | 负责控制数据流在系统中所有组件中流动，并在相应动作发生时触发事件 |
| 调度器(Scheduler) | 负责接受引擎发送过来的请求，并按照一定的方式进行整理排列、入队 |
| 下载器(Downloader) | 负责下载引擎发送的所有请求，并将其获取到的响应交还给引擎 |
| 蜘蛛(Spider) | 负责处理响应，从中提取所需数据，并将需要跟进的URL提交给引擎，再次进入调度器 |
| 管道(Item Pipeline) | 负责处理Spider中获取到的Item，并进行进行后期处理（详细分析、过滤、存储等） |
| 选择器(Selector) | 一个具有对HTML源代码进行解析和提取指定部分内容的类，例如CSS选择器和XPath |
| 项目(Item) | 一个用于收集从页面中提取出的数据的容器类 |
| 蜘蛛中间件  (Spider Middleware) | 处理发送到Spider的响应和从Spider生成的请求和项目过程中的钩子框架，用于在其中插入自定义功能 |
| 下载器中间件  (Downloader Middleware) | 处理请求和获取到的响应过程中的钩子框架，用于在全局范围内自定义下载行为 |
| 反爬虫机制 | 某些网站存在的需要登录、验证码、JavaScript动态加载页面、IP请求速度限制等限制网络爬虫的措施 |
| 分布式爬虫 | 在多台主机上通过共享爬取队列，同时运行爬虫任务协同爬取，从而提高爬取效率的方法 |
| Scrapyd | 一个用于部署和运行Scrapy爬虫的应用程序 |
| Selenium | 一个用于Web应用程序测试的工具 |
| RUCM | Restricted Use Case Modeling，基于用例建模的需求建模方法，目标是降低软件需求模型的二义性 |

1.4 参考资料

[1] GB/T 9385-2008，计算机软件需求规格说明规范[S]. <http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=2790825C43AD0B69E3C38C140BFFCFE6>

[2] 维基百科Scrapy[OL]. <https://en.wikipedia.org/wiki/Scrapy>

[3] Scrapy官方文档[OL]. <https://docs.scrapy.org/en/latest/>

[4] Scrapyd官方文档[OL]. <https://scrapyd.readthedocs.io/en/stable/>

[5] RUCM官方文档[OL]. <http://zen-tools.com/rucm/metamodels/resources/RUCM%20Manual.pdf>

[6] RUCM使用说明[OL]. <https://blog.csdn.net/spring_willow/article/details/65938594>

[7] RUCM使用实例[OL]. <https://blog.csdn.net/spring_willow/article/details/72661961>

[8] 可访问性

<https://developer.mozilla.org/zh-CN/docs/Learn/Accessibility>

[9] 响应式Web设计

<https://developer.mozilla.org/zh-CN/docs/Web_Development/%E5%93%8D%E5%BA%94%E5%BC%8F_Web_%E8%AE%BE%E8%AE%A1>

1. **任务概述**

2.1 目标

本项目通过提供一组特定网站的爬虫模板，使用Scrapy框架自动生成每个模板对应的爬虫脚本，从而使用户无需编写代码即可使用爬虫采集数据，同时用户可以在平台上运行和管理自己的爬虫程序。在根据模板生成爬虫脚本时，应当加入应对反爬虫机制来增强爬虫程序的鲁棒性，通过增加分布式爬虫机制来提高爬取效率。

2.2 用户特点

本软件系统的用户应当对想要爬取的网站有一定的了解，包括但不限于网站内各层级网页的网址、希望执行的翻页次数等，但无需对Python语言和爬虫编程有深入的了解。

此外，用户通过本软件系统完成目标网站的爬取后，应当能够正确地通过数据导出接口来进行数据的导出。其中，如果用户希望将爬取数据发布到数据库中，应该对数据库有一定的了解，包括但不限于设置数据库信息如服务器名称、端口、用户名称、密码、数据编码、数据库名称，以及配置原数据字段和目标数据字段的映射关系等。

2.3 系统模型

2.3.1 Scrapy框架

图2.1是Scrapy官方文档中的整体架构图。



图2.1 Scrapy整体架构

在爬虫工作过程中数据流由引擎控制，主要包括以下几个步骤。

1. 引擎从Spider获取初始请求；
2. 引擎将请求交给调度器，并获取下一个请求；
3. 调度器将下一个请求返回给引擎；
4. 引擎通过下载器中间件将请求发送给下载器；
5. 页面下载完成后，下载器会生成一个（带有该页面的）响应，并通过下载器中间件将其发送给引擎；
6. 引擎从下载器接收响应，并通过爬虫中间件将其发送到Spider进行处理；
7. Spider处理响应，并通过爬虫中间件将抓取的项目和新的（要跟踪的）请求返回给引擎；
8. 引擎将处理后的项目发送到管道，然后将处理后的请求发送到调度器，并获取下一个请求；
9. （从步骤1开始）重复该过程，直到不再有来自调度器的请求为止。

2.3.2 EasySpider

EasySpider的整体架构如下图所示。



图2.2 EasySpider整体架构

EasySpider整体上包含四个部分：

* 用户前端界面（Web UI）
* 网站后端
* 调度程序
* Scrapyd上部署的爬虫程序

本系统主要的工作流程包括以下几个步骤。

1. 用户界面前端界面为一般用户操作本系统的各类操作；
2. 网站接收并处理来自用户的操作请求，按需与调度程序交互以生成并执行爬虫任务；
3. 调度程序将任务（基于模板生成的爬虫程序）调度分配到Scrapyd节点执行作业（Job）；
4. 爬虫程序从目标站点抓取内容后，将用户所需的内容（爬虫结果）存储到一非关系数据库（MongoDB）中，并通知调度程序作业完成；
5. 网站后端处理爬虫结果并最终在用户前端页面显示或提供下载。

2.4 假定和约束

为了保证软件系统的正常发布、运行、维护，对其运行环境和过程做如下假设和约束：

1. 法律政策

软件设计、制作、发布等遵循相关法律政策，包括但不限于《软件产品管理办法》、《通信网络安全防护管理办法》、《信息安全等级保护管理办法》、《计算机信息系统安全保护条例》、《中国软件行业基本公约》等。

1. 编程语言

前端使用JavaScript ES6、JQuery 4.3.1作为编程语言，后端使用Python 3作为编程语言。

1. 工具约束

前端使用浏览器内置的JavaScript引擎作为运行工具，后端使用Python 3解释器作为运行工具。

1. 代码体积

软件系统的源代码体积应控制在100 M以内，以方便工具的编译、发布和使用。

1. 安全和保密

软件系统安全和保密方面的相关说明参考4.4节的描述。

1. **需求规定**

3.1 业务需求

3.1.1 Web UI

从业务建设方的角度，模板化爬虫程序管理平台的前端应满足如下业务需求：

1. 内置多种包含主流网站数据源的模板，如京东、天猫、大众点评等热门采集网站，模板提供设置参数即采集的字段；
2. 提供管理员入口，允许对模板执行增加、删除、更新、查询四种基本操作；
3. 支持远程控制采集任务，包括启动、暂停、终止、查看进度；
4. 内置采集登录模块，配置目标网站的账号密码，即可用该模块采集到登录后的数据；
5. 支持采集Cookie自定义，首次登录以后，自动记住Cookie，免去多次输入密码的繁琐，支持更多网站的采集。

3.1.2 后端及调度程序

网站后端及调度功能服务器为前台的Web UI提供相应的功能接口，同时还负责调度和管理部署了实际爬虫任务的Scrapyd分布式服务及其所在的服务器。主要业务需求如下所示：

1. 后端服务器一方面实时的响应并处理来自前端Web UI中用户提供的各类任务操作请求，包括根据用户提供的爬虫模板及关键字动态生成爬虫脚本、部署爬虫任务到爬虫服务器中的Scrapyd服务、管理各个爬虫服务器上的爬虫任务以及服务器的负载均衡调节；
2. 为平台管理者开放了服务器节点的管理，允许管理员新增和管理服务器节点以及对服务器完成Scrapyd的初始配置从而使其成为一个可用的分布式节点；
3. 负责爬虫存储结果的管理和统计。

3.2 功能需求

3.2.1 Scrapy框架

开发者使用Scrapy框架编写爬虫程序时，通过使用框架提供的各种组件，可以专注于核心逻辑的实现，而不必关心底层细节，从而可以快速地完成爬虫程序的开发。基于对Scrapy框架工作流程的分析，得出以下主要的功能需求，其中爬虫程序开发者是指使用Scrapy框架编写爬虫程序的人。

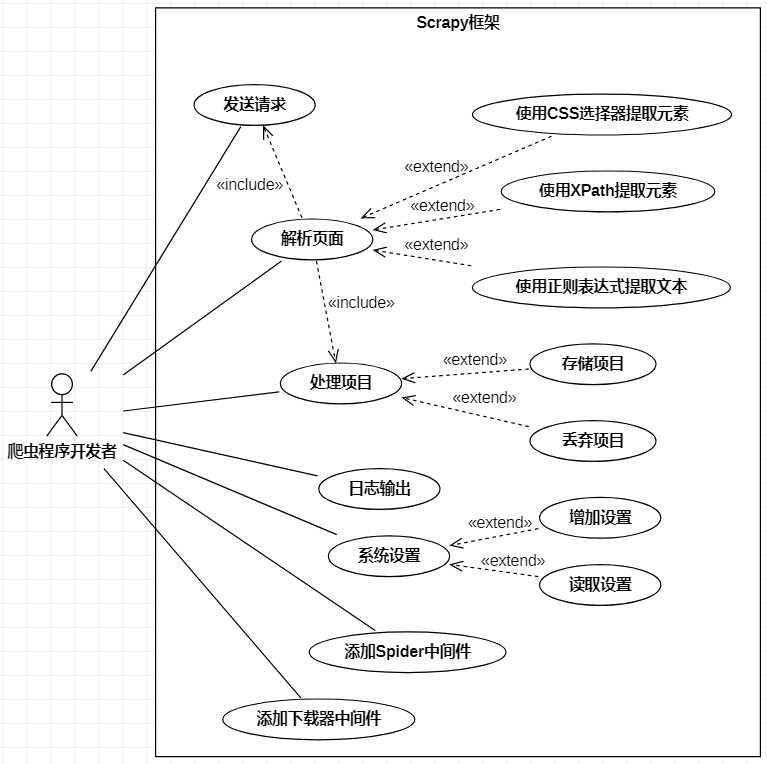


图3.1 Scrapy框架用例图

3.2.1.1 发送请求

在Spider模块产生一个包含URL的Request对象，引擎将其发送到调度器，加入爬取队列。主要有两类请求对象——初始请求和解析页面时产生的跟踪后续链接的请求。



图3.2 发送请求功能RUCM图

3.2.1.2 解析页面

根据实际需求，在Spider模块的parse方法中使用CSS选择器、XPath和正则表达式从包含在Response对象的HTML页面中提取需要的数据，并以项目(Item)的形式返回。



图3.3 使用CSS选择器提取元素功能RUCM图



图3.4 使用XPath提取元素功能RUCM图



图3.5 使用正则表达式提取文本功能RUCM图

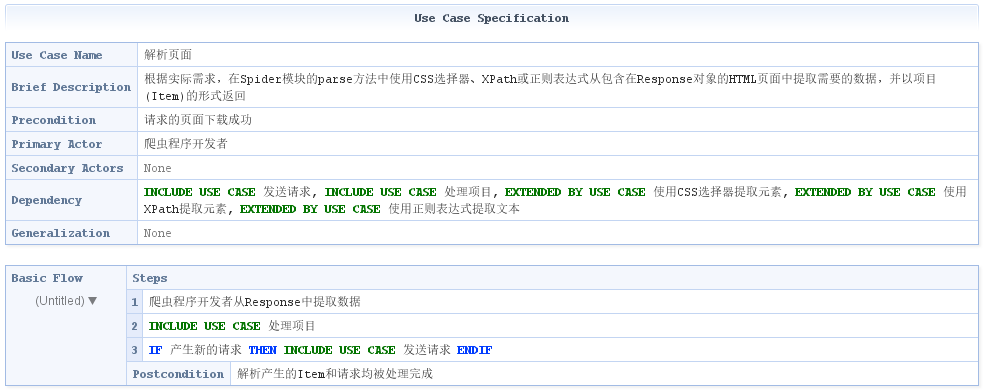


图3.6 解析页面功能RUCM图

3.2.1.3 处理项目

在自定义的项目管道类中处理Spider模块产生的项目，常用的处理方式包括两类——存储项目（例如将项目中的数据写入数据库）和丢弃项目（例如过滤去重）。



图3.7 存储项目功能RUCM图

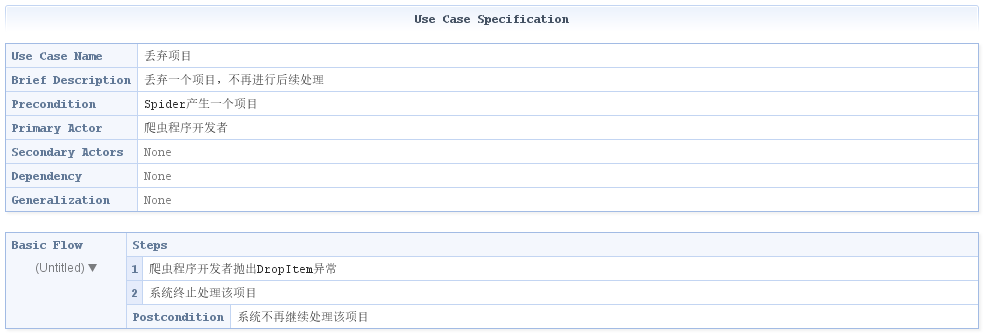


图3.8 丢弃项目功能RUCM图



图3.9 处理项目功能RUCM图

3.2.1.4 日志输出

程序运行中的事件日志记录。

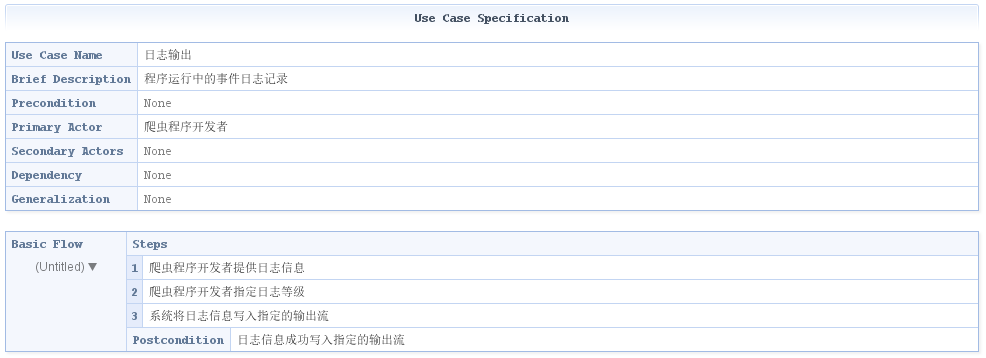


图3.10 日志输出功能RUCM图

3.2.1.5 系统设置

通过Scrapy设置可以自定义Scrapy所有组件的行为，例如下载间隔、自定义项目管道、自定义中间件等；除此之外还可以添加其他任何需要的设置，例如数据库连接参数等。

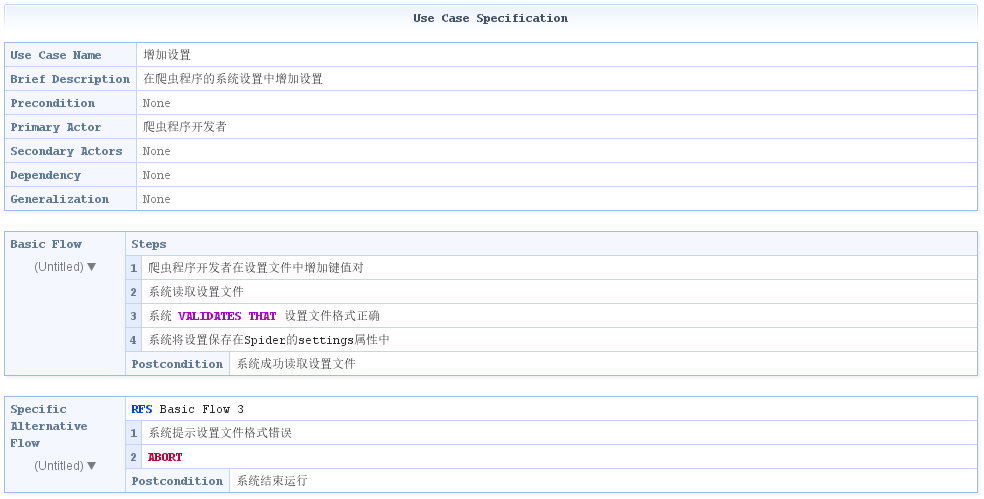


图3.11 增加设置功能RUCM图



图3.12 读取设置功能RUCM图



图3.13 系统设置功能RUCM图

3.2.1.6 添加Spider中间件

在自定义的Spider中间件类中处理发送到Spider的响应和从Spider生成的请求和项目。

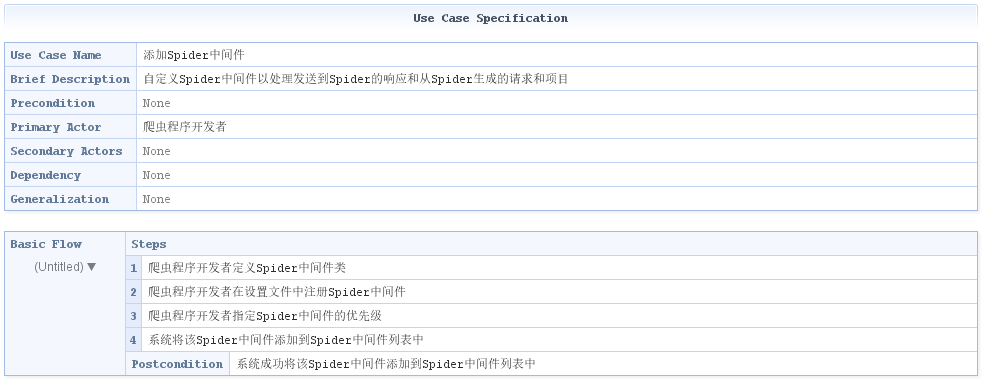


图3.14 添加Spider中间件功能RUCM图

3.2.1.7 添加下载器中间件

在自定义的下载器中间件类中处理请求和获取到的响应。

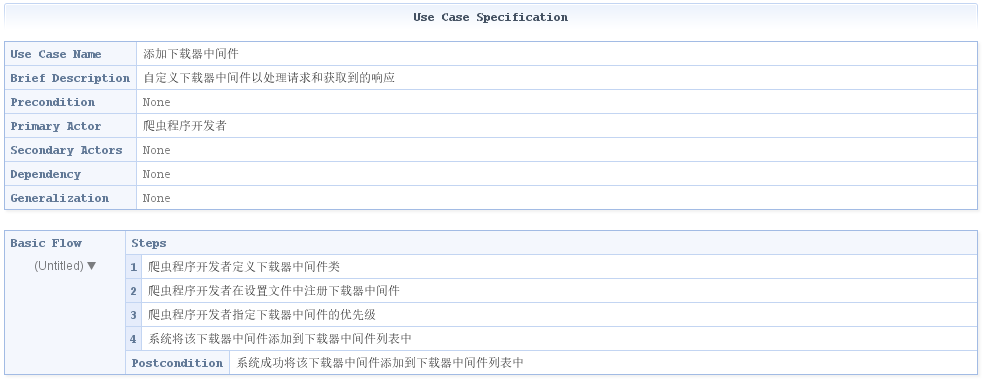


图3.15 添加下载器中间件功能RUCM图

3.2.2 Web UI

1. 全自动数据格式化

内置数据格式化引擎，支持字符串替换、正则表达式替换或匹配、去除空格、添加前缀或后缀、日期时间格式化、HTML转码等多项功能，采集过程中全自动处理，无需人工干预，即可得到所需格式数据。

1. 多种网页采集策略

对于特定的网站，分析其网页结构，识别各种网页元素，模拟操作包括输入文本、点击、移动鼠标、下拉框、滚动页面、等待加载、循环操作和判断条件等，实现模拟登录、动态渲染页面数据的采集。

1. 多层级采集

很多主流新闻、电商类的网站，里面包含一级商品列表页，也包含二级商品详情页，还有三级评论详情页面。不论网站有多少层级，模板化爬虫程序管理平台支持不限制层级的数据采集。

1. 多种数据导出方式

采集结果可以导出到本地，支持TXT、EXCEL、CSV和HTML文件格式，也可以直接发布到数据库（MySQL、MongoDB等）。

模板化爬虫程序管理平台旨在为不了解网络爬虫技术的用户提供可视化的数据采集工具，从用户角度，应满足如下需求：

1. 支持通过关键字过滤内置的多种模板；
2. 提供模板的描述、采集字段预览、采集参数预览、示例数据；
3. 允许管理采集任务，其中包括添加采集任务、查看采集的实时进度、启动采集、暂停采集、终止采集；
4. 允许管理采集结果，其中包括浏览采集结果、保存采集结果到本地。

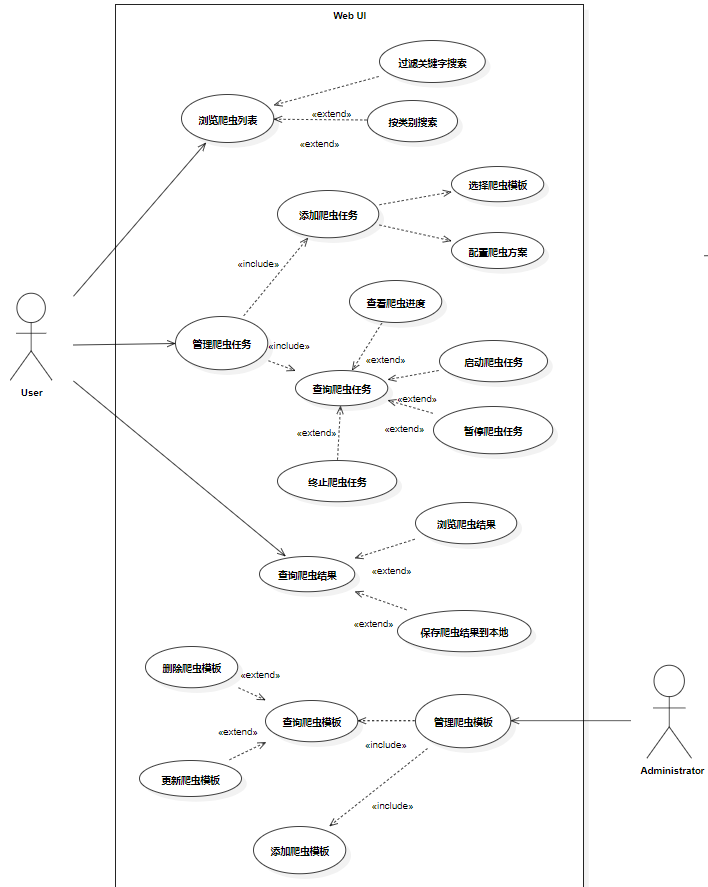


图3.16 EasySpider Web UI用例图

3.2.2.1 浏览爬虫列表

浏览爬虫列表提供给用户查看系统内现有爬虫的功能，当用户想要浏览爬虫模板时，首先进入客户端，选择浏览爬虫列表功能，此时系统会获取当前爬虫列表并返回给用户，用户会得到系统内的爬虫模板列表。



图3.17 浏览爬虫列表功能RUCM图

3.2.2.2 过滤关键字搜索

考虑到爬虫模板数量问题，系统提供给用户根据关键字来搜索爬虫模板的功能。



图3.18 过滤关键字搜索功能RUCM图

3.2.2.3 添加爬虫任务

当用户想要添加一个爬虫任务时，首先进入爬虫管理界面并选择添加爬虫任务功能。随后用户为新任务进行配置，如选择爬取的字段，选择爬虫模板等等；当配置完成后用户将新的爬虫任务添加到系统中。



图3.19 添加爬虫任务功能RUCM图

3.2.2.4 选择爬虫模板

用户在为新添加爬虫任务选择爬虫模板时，首先进入添加爬虫任务界面并选择爬虫模板功能，当用户选择模板有效时，系统将为爬虫任务配置该爬虫模板。若爬虫模板无效或系统内没有模板时，系统会提示错误信息并返回添加任务界面。



图3.20 选择爬虫模板功能RUCM图

3.2.2.5 配置爬虫方案



图3.21 配置爬虫方案功能RUCM图

3.2.2.6 查看爬虫进度

用户想要查看爬虫任务进度时，首先进入管理爬虫任务界面并选择查看爬虫进度功能，此时系统获取系统内现有的爬虫任务的状态并返回给用户，用户获得当前爬虫进度。此功能只能查看该用户添加的爬虫任务进度。



图3.22 查看爬虫进度功能RUCM图

3.2.2.7 启动爬虫任务

系统为用户提供启动爬虫任务功能，当一个爬虫任务是新添加的或者之前被暂停，此时需要用户或管理员进行手动启动。首先用户进入管理爬虫任务界面并查看当前爬虫任务功能（只能查看当前用户的爬虫任务）。用户在得到的爬虫任务列表中选择一个爬虫任务进行启动，若该爬虫任务已经在运行或已经停止，则启动失败；否则爬虫任务启动成功。



图3.23 启动爬虫任务功能RUCM图

3.2.2.8 暂停爬虫任务

系统为用户提供暂停爬虫任务的功能，当用户想要暂停一个运行中的爬虫任务时，首先进入管理爬虫任务界面并查看当前的爬虫任务（当前用户的爬虫任务），在系统所列出的爬虫任务中选择一个进行暂停。若该爬虫任务不在运行中，则暂停失败；否则爬虫任务暂停成功，爬虫任务状态从运行变为暂停。



图3.24 暂停爬虫任务功能RUCM图

3.2.2.9 浏览爬虫结果

系统提供给用户浏览爬虫结果的功能。首先用户进入爬虫任务的管理界面并选择一个爬虫任务，选择浏览爬虫结果功能，系统获取爬虫爬取的结果并返回给用户。



图3.25 浏览爬虫结果功能RUCM图

3.2.2.10 查询爬虫任务

查询爬虫任务功能提供给用户查询当前系统内所有爬虫任务的功能。用户首先进入爬虫任务界面，选择查询爬虫任务；系统获取该用户下的所有爬虫任务并将任务列表返回给用户。

查询爬虫任务功能有四项扩展功能：查看爬虫进度、终止爬虫任务、暂停爬虫任务、启动爬虫任务。



图3.26 查询爬虫任务功能RUCM图

3.2.3 后端及调度程序

网站后端及调度功能服务器为前台的Web UI提供相应的功能接口，同时还负责调度和管理部署了实际爬虫任务的Scrapyd分布式服务及其所在的服务器。后端服务器一方面实时的响应并处理来自前端Web UI中用户提供的各类任务操作请求，包括根据用户提供的爬虫模板及关键字动态生成爬虫脚本、部署爬虫任务到爬虫服务器中的Scrapyd服务、管理各个爬虫服务器上的爬虫任务以及服务器的负载均衡调节，另一方面为平台管理者开放了服务器节点的管理，允许管理员新增和管理服务器节点以及对服务器完成Scrapyd的初始配置从而使其成为一个可用的分布式节点。

* 1. 从请求队列中获取指令：从当前的请求队列中获取来自用用户的任务管理指令
  2. 动态生成爬虫脚本：根据用户提供的爬虫需求选项，生成对应的爬虫脚本
  3. 查询任务所在服务器节点：根据用户提供的爬虫任务，在关系数据库中找到对应的服务器节点
  4. 管理任务：根据用户请求，添加新的爬虫任务或者对现有的任务进行管理
  5. 新增任务：根据用户请求，生成一个新的爬虫任务并根据服务器负载以及任务类型将其部署到某一服务器
  6. 唤醒任务：将位于某个服务器上的已挂起任务转为运行状态
  7. 挂起任务：将位于某个服务器上的正在运行的任务转为挂起状态
  8. 终止任务：终止每某一个正在运行或者已经被挂起的任务
  9. 调用负载均衡调度算法：根据服务器负载情况，挑选出合适的服务器
  10. 获取服务器负载情况：获取当前所有服务器的负载压力
  11. 管理服务器节点：对系统包含的所有服务器节点进行管理
  12. 新增服务器节点：为系统新增一个可用的服务器节点
  13. 查询服务器节点：从当前的服务器节点列表中找到某一节点
  14. 停用服务器节点：停用某一正在运行的服务器节点，挂起其上运行的所有爬虫任务
  15. 删除服务器节点：删除某一服务器积极点，终止其上运行的所有爬虫任务
  16. 恢复服务器节点：恢复某一已停用的服务器节点，恢复其上所有已挂起爬虫任务的运行
  17. 配置服务器节点：对某一新增的服务器节点进行配置，使其具备正常执行爬虫任务的能力



图3.27 EasySpider后端及调度程序用例图

3.2.3.1 管理任务

后端及调度程序最重要的任务之一便是通过对来自用户的各类任务请求进行应答，从而完成对系统中所有的爬虫任务进行管理。可能存在的用户任务请求包括“新增任务”、“唤醒任务”、“挂起任务”以及终止任务。无论是哪一种任务请求，都要求当前系统中存在可用的服务器节点。系统在工作时，不断的从请求队列中获取来自用户的指令并执行。如果队列中不存在任何新的请求，那么系统休眠100 ms从而节省系统资源。当系统检测到不存在任何可用的服务器节点时，需要提醒管理员添加新的服务器节点，或者对不可用的服务器节点进行维护。

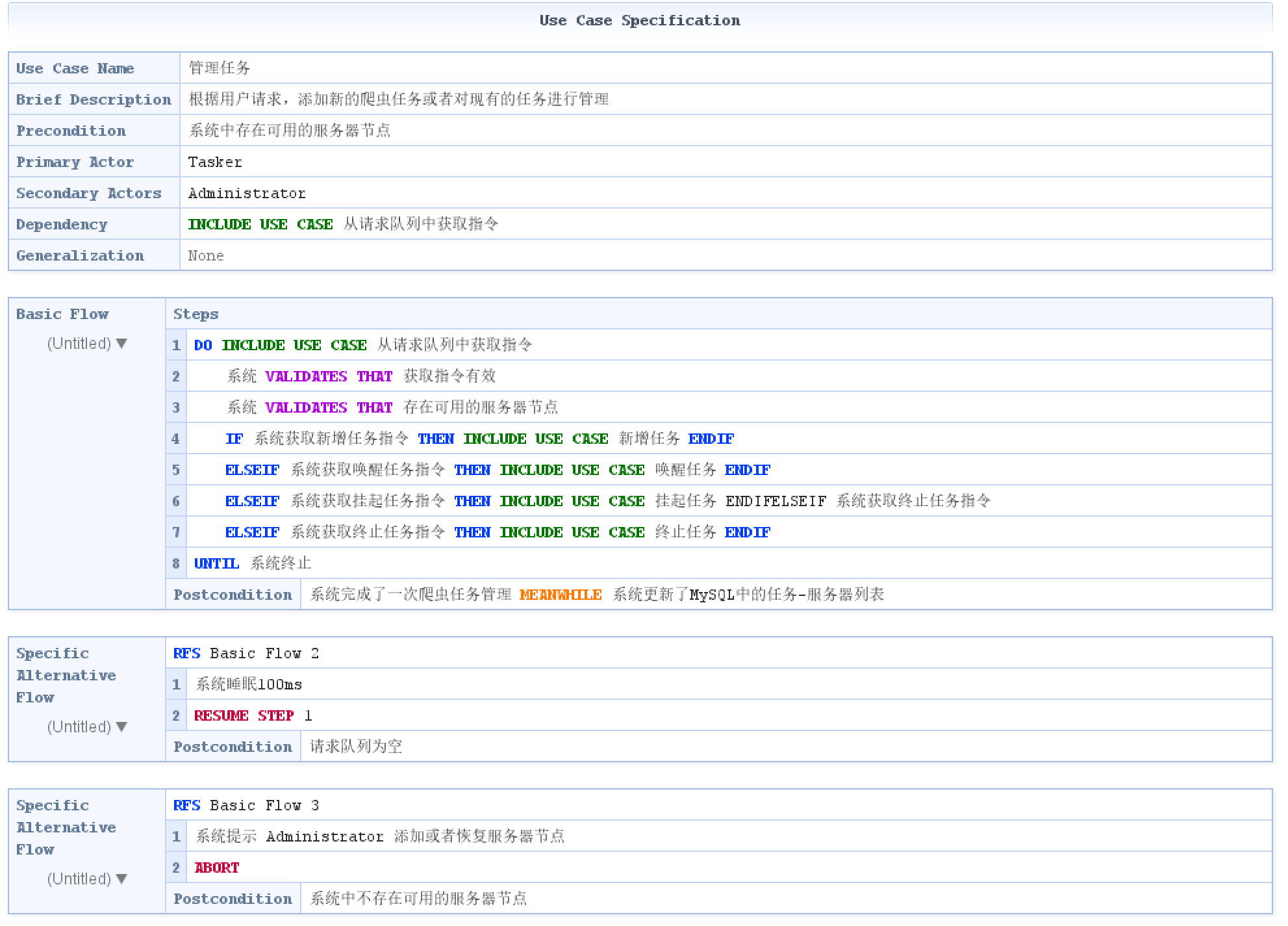


图3.28 管理任务功能RUCM图

3.2.3.2 新增任务

新增（爬虫）任务是系统中的一个非常重要的用例，需要系统中其他众多功能的共同参与。系统接收到来自用户的新建任务请求，以及目标任务的模板以及任务参数关键字之后，开始一个新建任务的过程。首先，系统需要获取当前所有服务器节点的工作负载情况；在此之后，系统需要使用负载均衡调度的算法，从服务器节点列表中选择一个合适的服务器节点；如果到此为止没有出现任何问题，则根据用户选择的爬虫模板以及提供的关键字，动态生成对应的爬虫脚本；然后，将该爬虫脚本打包并部署到之前选择的服务器节点上；最后将爬虫任务及服务器的相关信息记录到关系数据库中。至此，系统完成了一次完整的新建（爬虫任务）的过程。同样，如果无法获取可用的服务器节点，系统需要向管理员进行提示。



图3.29 新增任务功能RUCM图

3.2.3.3 挂起任务

“挂起任务”、 “恢复任务”以及终止任务的系统行为有相似之处，这里选择“挂起任务”用例进行具体陈述。首先，需要查询（爬虫）任务所在的服务器节点；然后需要确认数据库中存在该任务及其对应的服务器节点；再之后，系统需要发送请求挂起给目标服务器；最后，若该服务器上确实存在该任务（由于延迟，可能出现任务已经完成的情况），系统就将该任务挂起



图3.30 挂起任务功能RUCM图

3.2.3.4 管理服务器节点

系统可以为管理员提供服务器节点管理的功能，具体的管理操作包括查询服务器的负载情况，查询服务器的节点以及新增服务器的节点。



图3.31 管理服务器节点功能RUCM图

3.2.3.5 新增服务器节点

管理员为系统新增服务器时，首先需要对目标服务器节点进行配置，部署Scrapyd服务，确保其可以正常行使爬虫服务器的职责。之后，系统将该服务器节点的地址等各类信息写入数据库，使得该服务器节点可以为系统中的其他部分所见。



图3.32 新增服务器节点功能RUCM图

3.2.3.6 停用服务器节点

“停用服务器节点”、“恢复服务器节点”以及“删除服务器节点”这几个系统功能的实现也是类似的流程，这里以“停用服务器节点”用例为例进行详细阐述。首先，一如既往，系统需要判断目标服务器节点是否存在；若该服务器节点存在，则需要将该服务器节点上的所有正在运行的爬虫任务进行挂起操作；当该服务器节点上的所有任务都被挂起之后，该服务器彻底进入挂起状态，需要将相关信息写入关系数据库。



图3.33 停用服务器节点功能RUCM图

3.2.4 Scrapyd

Scrapyd是一个用于部署和运行Scrapy爬虫的应用程序。Scrapyd中使用Scrapy项目（Project，以下简称项目）的概念，可以同时管理多个项目的爬虫运行。一个项目由若干个爬虫（Spider）构成，Scrapyd能够细化到对一个项目内的具体的一个爬虫进行调度运行。Scrapyd同时提供对项目的版本管理功能，一个项目的代码版本称为一个版本（Version），编程者通过接口能够上传一个Python Egg格式的项目，并将其添加为指定项目的一个新版本。在实际运行时，缺省情况使用最新的爬虫版本运行。编程者可以通过它提供的JSON API来部署（上传）爬虫项目和控制爬虫。本项目使用Scrapyd作为在服务器上部署的软件。对Scrapyd的API进行分析，得出以下的功能需求。

1. 查询守护进程状态：Scrapyd作为一个守护进程（Daemon）运行，使用者需要能获取其运行状态，包括各种状态下运行爬虫的数量及节点的名称。
2. 列出项目中可用爬虫：Scrapyd以项目管理一组爬虫，使用者需要能获取一个项目中所包含的爬虫列表。
3. 列出项目中各种状态下的作业：使用者需要获取一个项目中等待运行、正在运行和完成运行的爬虫的列表。
4. 运行作业：使用者需要能调度一个爬虫运行，获取运行后的作业编号。
5. 取消作业：使用者需要能取消一个作业的运行。如果一个作业等待运行，该作业会从被移除。 如果作业正在运行，该作业会被中止。
6. 列出项目：使用者需要能取得一个上传到该Scrapyd服务器的项目列表。
7. 删除项目：使用者需要能删除一个项目以及项目所有上传的版本。
8. 列出项目的可用版本：使用者需要能取得一个项目可用的版本列表。
9. 添加项目版本：使用者需要能为一个项目添加一个爬虫版本，即上传爬虫代码作业一个版本。
10. 删除项目版本：使用者需要能删除一个项目的特定版本（代码版本）。



图3.34 Scrapyd用例图

3.2.4.1 查询守护进程状态

调度器（Scheduler）需要获取Scrapyd实例的运行状态。运行状态包括处于等待运行、正在运行和完成运行的作业的数量，以及节点名称作为节点的标识。调度器需要这些信息来判断该Scrapyd实例节点的负载情况。

A screenshot of a cell phone

Description automatically generated

图3.35 查询守护进程状态功能RUCM图

3.2.4.2 列出可用爬虫

调度器需要获取指定项目名称的项目具有哪些可用的爬虫的列表。

A screenshot of a cell phone

Description automatically generated

图3.36 列出可用爬虫功能RUCM图

3.2.4.3 列出当前作业

调度器需要了解指定一个项目中的多个作业的完成情况。完成情况包括等待运行、正在运行和完成运行的作业编号的列表。

A screenshot of a cell phone

Description automatically generated

图3.37 列出当前作业功能RUCM图

3.2.4.4 运行作业

调度器需要能够创建指定一个项目的一个爬虫的特定版本的作业，需要获取创建作业后的编号以在后续操作中查询该作业的运行情况。

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

图3.38 运行作业功能RUCM图

3.2.4.5 取消作业

调度器需要能够取消指定编号的作业。当该作业已经被调度运行处于运行状态时，作业需要被中止；当作业还在等待运行时，需要将作业从等待队列中移除。

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

图3.39 取消作业功能RUCM图

3.2.4.6 列出项目

调度器需要获取在节点已经上传的项目的列表。

A screenshot of a cell phone

Description automatically generated

图3.40 列出项目功能RUCM图

3.2.4.7 删除项目

调度器需要删除在节点上项目。删除项目时需要将项目相关的所有版本删除。

A screenshot of a cell phone

Description automatically generated

图3.41 删除项目功能RUCM图

3.2.4.8 列出项目版本

调度器需要查看指定项目的可用版本的列表。

A screenshot of a cell phone

Description automatically generated

图3.42 列出项目版本功能RUCM图

3.2.4.9 添加项目版本

调度器需要为一个项目上传一个项目的版本，即代码。

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

图3.43 添加项目版本功能RUCM图

3.2.4.10 删除项目版本

调度器需要能够删除指定项目的一个版本。

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

图3.44 删除项目版本功能RUCM图

3.3 非功能需求

3.3.1 Scrapy框架

Scrapy需要支持以下非功能性需求以便更好地使用框架编写爬虫。

* 1. 高适用性。默认情况下，Scrapy每个进程运行一个Spider。Scrapy通过内部API也支持单进程多个Spider。除了支持单机爬取，也可以进行分布式爬取，支持启动多个Scrapyd，并分配到不同机器上。
  2. 高可靠性。有些网站实现了特定的机制，以一定规则来避免被爬虫爬取。框架需要能够实现避免被禁止，并且可以设置全局并发以便同时处理多个请求。
  3. 高通用性。针对非英语语系中不标准或者错误的编码声明，提供了自动检测以及健壮的编码支持。
  4. 高扩展性。通过使用信号、设计好的API（管道、中间件、扩展）来定制实现功能。内置的中间件及扩展为下列功能提供支持:Cookies和session处理、HTTP压缩、HTTP认证、HTTP缓存、用户登录模拟、爬取深度限制。

3.3.2 Web UI

1. 可访问性

在网页开发的过程中，应使用正确的HTML标签来表达正确的意图，如：采用具有标题、段落、列表等内容的良好结构；避免滥用<div>元素，替换为适当的HTML5语义元素，典型的HTML5元素包括<nav>、<footer>、<article>等；允许用户通过键盘操作网页上的UI控件；为图像提供描述性文字，避免空的alt属性。

在CSS的使用方面，可以采用的最佳实践包括：为网站选择配色方案时，应确保文本（前景）颜色与背景颜色形成鲜明对比；在隐藏内容时采用绝对定位等机制实现，以便屏幕阅读器可以看到这些内容；允许用户覆盖原本样式，即确保网站CSS的设计具有足够的灵活性，以便用户的自定义样式可以起作用。对于Javascript而言，遵守 unobtrusive JavaScript原则：网页的基本功能在没有JavaScript的情况下正常工作，JavaScript只被用于提供增强功能。

1. 兼容性

要求网站兼容当前五大主流浏览器，即Edge、火狐（Firefox）、谷歌（Chrome）、Safari和Opera。考虑到越来越多的用户使用移动设备来浏览网站，网站页面在设计时应采用响应式设计，主要包括如下三大策略：流式布局，按照浏览器视窗的百分比来设定所有容器的宽度，从而使容器在浏览器窗口大小变化时自动缩放；媒体查询，基于显示设备的物理特性（例如：尺寸、分辨率、宽高比、颜色位深等）来调用不同的样式表；流式图片，设置图像所占宽度至多为设备的最大宽度。

1. 易用性

模板化爬虫程序管理平台的目标用户是不了解网络爬虫技术希望简易采集数据的用户。因此在可视化用户界面的设计上，采用常见的仪表盘前端模板，并对于每个模板提供描述、示例、预览功能，实现在采集的各个阶段可见可控。如果用户是首次注册的新用户，则要求网站提供一些基本操作的引导动画，以便用户快速上手。

3.3.3 后端

1. 模板的可扩充性

为了满足用户简易采集数据的需求，模板化爬虫程序管理平台需内置多种热门采集网站的模板。考虑到模板网站的网页结构或内容的变化、热门网站的增加、采集策略的更新等因素，模板化爬虫程序管理平台应满足模板的可扩充性，即：设计通用的Scrapy爬虫模板，将爬取规则、页面解析方式等抽离出来做成配置文件。这样在新增模板的时候，只需要实现目标网站的爬取规则和提取规则即可，有效避免代码的重用。

1. 安全性

为了满足用户采集特定网站登录后的数据的需求，模板化爬虫程序管理平台允许用户配置目标网站的账号密码。账号和密码属于敏感数据，在传输和后端处理的过程中应做到加密，保障用户的隐私安全。

此外，用户登录本平台使用的密码应该具有一定的复杂度，比如密码的长度、大小写、字符种类等，如果用户注册时输入的密码过于简单，前端应给出合法密码相关要求的提示信息，以便用户进行修改。

1. 防攻击性

针对网站常见的漏洞如SQL注入、XSS跨站脚本攻击、admin密码漏洞、请求伪造漏洞等设置相应的安全策略，比如管理员账号的密码复杂度应和普通用户一样，对用户输入的表单数据进行合法性判断和过滤，避免后台数据库信息的泄露。

1. 可维护性

可维护性是衡量一个系统的可修复(恢复)性和可改进性的难易程度。所谓可修复性是指在系统发生故障后能够排除(或抑制)故障予以修复，并返回到原来正常运行状态的可能性。而可改进性则是系统具有接受对现有功能的改进，增加新功能的可能性。在系统运行期间，确保服务器能够7×24小时连续不间断地运行，如果出现故障，要求在半小时内将系统重新部署到备用服务器。在代码开发过程中，应采用统一的编码风格如驼峰式命名规则，使用模块化、结构化程序设计，为关键变量或函数添加注释，以降低后期维护的难度。

1. 可靠性

系统可靠性一般是指在规定的时间内和规定的工况下，系统完成规定功能的能力或概率。对于网站来说，最基本的要素就是网站的托管。如果网站经常宕机，那么其可靠性就会大大降低。因此要检查网站托管平台是否提供了必要的资源，如果未满足要求，也许需要升级相应的网站托管计划。或者可能需要改变托管方式，最初网站可能是托管在共享主机上，但随着网站的流量增长，出现网站工作缓慢的情况，那么就可能需要选择VPS或专用服务器来提高网站性能。

1. 良好的访问性能

考虑到本平台包含大量的图片资源，可能会降低网站的加载速度，影响用户体验，因此需要优化网站上的图片，可以采用的措施包括缩小图片的尺寸，或者用新的图片格式如“WebP”或”JPegXR”，实现在不降低图像质量的条件下大幅度减少图像文件所占空间，减少页面加载时间。除了图片资源之外，对JavaScript脚本应采用压缩后的版本，可以有效减小脚本文件的大小，同时使用异步方式加载，这样即使第三方脚本出现故障，也不会影响网站的访问。

此外，启用网站缓存来避免服务器重复执行一些复杂耗时的操作如生成页眉和页脚、查找网站侧栏小部件等，这些操作大多时候计算结果是相同的。通过网站缓存，服务器可以以最快的速度提供网页页面，做到提升网站的性能和用户满意度。

1. 采集数据的完整性与有效性

模板化爬虫程序管理平台应根据不同网站，提供多种网页爬取策略与配套资源，允许自定义配置、组合运用、自动化处理，从而帮助整个采集过程实现数据的完整性与有效性。目前已有的爬虫以及反爬虫策略如图3.45所示。



图3.45 爬虫和反爬虫

上图展示了爬虫以及反爬虫技术的对抗史。其场景以及解决方案归纳如下。

1. IP地址验证

场景：有些网站通过IP地址来识别是请求否是都来源于同一个客户端。

解决方案：不断地随机更换代理服务器的IP 地址。middlewares.py通过自定义的下载中间件为Scrapy设置了代理服务器。需要开发者事先准备好一系列代理服务器。settings.py文件设置启用自定义的下载中间件。

1. 禁用Cookie

场景：有些网站通过跟踪 Cookie 来识别是否是同一个客户端。

解决方案：在配置文件中关闭默认开启的Cookie选项COOKIES\_ENABLED = False。

1. 违反爬虫规则文件

场景：有些网站robots.txt 文件，制定了爬虫规则。

解决方案：指定不遵守爬虫规则，ROBOTSTXT OBEY = False。

1. 限制访问频率

场景：当同一个 IP 地址、同一个客户端访问目标网站过于频繁时，很可能会被当成机器人。

解决方案：为了更好地模拟正常用户的访问速度，可以限制 Scrapy 的访问频率。比如开启频率限制，设置访问开始延迟，访问之间最大延迟，并行每台服务器请求数量，下载后的自动延迟。

1. 图形验证码

场景：为了防止机器程序访问，对同客户同IP达到一定访问次数，会要求输入图形验证码。

解决方案：让机器识别验证码。使用PIL、Libsvrn 等第三方库，自己开发程序来识别图形验证码。或是通过第三方图形验证码的在线识别网站进行识别，但识别率高的背后往往需要额外的收费。

1. 动态页面加载

场景：增加动态网站，数据通过JS动态加载，增加网络分析复杂度；

解决方案：对于那些充斥着大量复杂JavaScript渲染的页面，请求非常难构造，或者构造方式经常发生变化，这种情况下采用Selenium直接模拟请求的方法，极大的减少了爬虫的项目难度和工作量。然而，若是所有的网页无论静态动态都采用Selenium来完成，整体效率又会变得十分低下。所以我们需要结合两者的优势，对于拥有大量静态元素的网页，直接采用Scrapy来完成爬虫；对于网页的登录界面，或者是拥有大量动态JS加载的网页，Selenium直截了当的风格成为我们的首选。

3.4 输入和输出

对于Scrapy框架，输入是URL请求，输出是包含下载的页面的响应。对于EasySpider，输入是用户选择的爬虫模板，输出是根据模板生成的爬虫程序。对于后端及调度程序，输入是用户创建的爬虫程序任务，输出是保存在数据库中的爬取结果。

3.5 数据库特性

EasySpider的数据流图如下图所示。



图3.46 数据流图

EasySpider使用两个数据库：

（1）关系数据库（MySQL）：用于存储用户信息、爬虫任务信息等结构化的数据；

（2）非关系数据库（MongoDB）：用于存取非结构化的爬虫结果数据。

EasySpider设计的数据处理和存储如数据流图所示，用户通过网站界面，发送爬虫任务创建信息，后端服务器接收任务创建信息，并存储进关系数据库数据库，同时这个新任务的信息也经由调度程序进行调度运行的处理，最终在调度程序调度后，将生成的作业信息发送到Scrapyd服务器，由Scrapyd根据作业创建信息进行爬虫。爬虫程序得到爬取结果后，将经过处理的结果存储在爬虫结果数据库。

EasySpider的需求分析阶段分析得到5个实体（Entity）：

* 用户：EasySpider的使用者
* 任务：由用户创建的一项爬虫任务（Task）
* 作业：爬虫任务落实在Scrapyd实例上运行的作业（Job）
* 节点：系统部署的Scrapyd实例
* 模板：系统预设的一系列爬虫程序模板

以及4个关系（Relation）：

* (用户,任务)：表示用户创建了任务
* (任务,模板)：表示任务使用的模板
* (任务,作业)：表示任务包含的Scrapyd节点上的作业
* (作业,节点)：表示作业运行于节点

EasySpider的UML数据库图如下图所示。



图3.47 UML数据库图

EasySpider使用非关系数据库MongoDB来存储爬虫程序的输出（爬虫结果），MongoDB使用BSON（一种扩展的JSON）作为非结构化的存储数据。一个存储的记录称为一个文档（Document），其本质上相当于一个JSON对象。一个数据库（Database）包含若干个集合（Collection），一个集合中包含若干个文档（Document）。

EasySpider使用Collection的名称来区分不同的任务（Task），在Collection中存储由一个任务产生的所有爬虫结果，一个文档对应一次爬虫程序输出。

3.6 故障处理

对于网络原因导致的Scrapy下载请求页面失败，或其他原因导致没有正确得到响应（即HTTP状态码不是200），使用Scrapy的错误回调函数(Request.errback)进行处理，当发生异常时记录日志并重新发送请求，同时可设置最大重试次数。

对于硬件故障导致的爬虫作业任务异常中断，根据日志记录判断中断点，并从中断点处继续运行。

1. **运行环境**

4.1 设备

4.1.1 客户端

* 操作系统：Windows, Linux, macOS
* 内存：2 GB以上
* 浏览器：Google Chrome, Mozilla Firefox等现代浏览器

4.1.2 后端服务器

* 操作系统：openSUSE, CentOS等Linux发行版本
* 内存：8 GB以上
* 硬盘：20 GB以上
* 软件：Python, MySQL, MongoDB

4.1.3 爬虫服务器

* 操作系统：openSUSE, CentOS等Linux发行版本
* 内存：8 GB以上
* 硬盘：20 GB以上
* 软件：Python, Scrapyd, Selenium

4.2 支持软件

* Python 3.5+
* Scrapy 2.0
* Scrapyd 2.0
* Selenium
* Django 3.0
* MySQL 8.0
* MongoDB 4.0

4.3 接口

4.3.1 硬件接口

无

4.3.2 软件接口

* Python类库
* Scrapy组件
* Scrapyd API

4.3.3 通信接口

* HTTP协议
* TCP/IP协议

4.3.4 用户接口

* Web UI

4.4 安全和保密

* 数据库的安全：访问控制（Access Control）、远程访问控制（防火墙）
* Scrapyd节点服务器的安全：基于源地址的防火墙策略用于保护节点API仅能被调度程序所在服务器访问、基于目标地址的防火墙策略用于保护节点仅访问限定白名单列表内的目标站点
* 用户数据的安全：用户登录鉴定，用户密码加密存储，HTTP传输层加密