

**基于Scrapy的模板化爬虫程序管理平台**

**软件需求规格说明书**

v1.0

赵正阳 SY1906428

郭浩隆 SY1906430

沈一聪 SY1906510

梁远志 SY1906503

宋冰晨 SY1906429

2020年3月25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 | 全文 | 完成初稿 |

**目录**

[1. 引言 - 1 -](#_Toc36234315)

[1.1 编写目的 - 1 -](#_Toc36234316)

[1.2 背景 - 1 -](#_Toc36234317)

[1.3 术语和缩略语 - 1 -](#_Toc36234318)

[1.4 参考资料 - 2 -](#_Toc36234319)

[2. 任务概述 - 3 -](#_Toc36234320)

[2.1 目标 - 3 -](#_Toc36234321)

[2.2 系统模型 - 3 -](#_Toc36234322)

[2.2.1 Scrapy框架 - 3 -](#_Toc36234323)

[2.2.2 EasySpider - 4 -](#_Toc36234324)

[2.3 假定和约束 - 5 -](#_Toc36234325)

[3. 需求规定 - 6 -](#_Toc36234326)

[3.1 功能需求 - 6 -](#_Toc36234327)

[3.1.1 Scrapy框架 - 6 -](#_Toc36234328)

[3.1.2 Web UI - 6 -](#_Toc36234329)

[3.1.3 后端及调度程序 - 8 -](#_Toc36234330)

[3.1.4 Scrapyd - 9 -](#_Toc36234331)

[3.2 用例图 - 10 -](#_Toc36234332)

[3.2.1 Scrapy框架 - 10 -](#_Toc36234333)

[3.2.2 Web UI - 10 -](#_Toc36234334)

[3.2.3 后端及调度程序 - 11 -](#_Toc36234335)

[3.2.4 Scrapyd - 12 -](#_Toc36234336)

[3.3 RUCM图 - 13 -](#_Toc36234337)

[3.3.1 Scrapy框架 - 13 -](#_Toc36234338)

[3.3.2 Web UI - 18 -](#_Toc36234339)

[3.3.3 后端及调度程序 - 22 -](#_Toc36234340)

[3.3.4 Scrapyd - 24 -](#_Toc36234341)

[3.4 非功能需求 - 30 -](#_Toc36234342)

[3.4.1 Web UI - 30 -](#_Toc36234343)

[3.4.2 后端 - 31 -](#_Toc36234344)

[3.5 输入和输出 - 33 -](#_Toc36234345)

[3.6 数据库特性 - 33 -](#_Toc36234346)

[3.7 故障处理 - 35 -](#_Toc36234347)

[4. 运行环境 - 36 -](#_Toc36234348)

[4.1 设备 - 36 -](#_Toc36234349)

[4.1.1 客户端 - 36 -](#_Toc36234350)

[4.1.2 后端服务器 - 36 -](#_Toc36234351)

[4.1.3 爬虫服务器 - 36 -](#_Toc36234352)

[4.2 支持软件 - 36 -](#_Toc36234353)

[4.3 接口 - 36 -](#_Toc36234354)

[4.3.1 硬件接口 - 36 -](#_Toc36234355)

[4.3.2 软件接口 - 37 -](#_Toc36234356)

[4.3.3 通信接口 - 37 -](#_Toc36234357)

[4.3.4 用户接口 - 37 -](#_Toc36234358)

[4.4 安全和保密 - 37 -](#_Toc36234359)

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

项目背景：随着网络的迅速发展，万维网成为大量信息的载体，如何有效地提取并利用这些信息成为一个巨大的挑战。使用网络爬虫可以方便地爬取网页并从中提取结构化的数据，但需要使用者会编写爬虫代码。一些不熟悉编程的用户可能更希望通过界面操作的方式实现数据的自动化采集。

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 |
| 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 |

1.4 参考资料

* GB/T 9385-2008 《计算机软件需求规格说明规范》   
  <http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=2790825C43AD0B69E3C38C140BFFCFE6>
* Scrapy官方文档：<https://docs.scrapy.org/en/latest/>
* Scrapyd官方文档：<https://scrapyd.readthedocs.io/en/stable/>
* RUCM官方文档  
  <http://zen-tools.com/rucm/metamodels/resources/RUCM%20Manual.pdf>
* RUCM相关博客  
  <https://blog.csdn.net/spring_willow/article/details/65938594>  
  <https://blog.csdn.net/spring_willow/article/details/72661961>

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 系统模型

2.2.1 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。

2.2.2 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。

2.3 假定和约束

无

1. **需求规定**

3.1 功能需求

3.1.1 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。

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。
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。
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）。
4. 日志输出：程序运行中的事件日志记录。
5. 添加设置：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。
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。
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。

3.1.2 Web UI

1. 业务需求

从业务建设方的角度，模板化爬虫程序管理平台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。
6. 用户需求

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。
5. 功能需求
6. 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）。

3.1.3 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

3.1.4 Scrapyd

Scrapyd是一个用于部署和运行Scrapy爬虫的应用程序。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）。

3.2 用例图

3.2.1 Scrapy框架

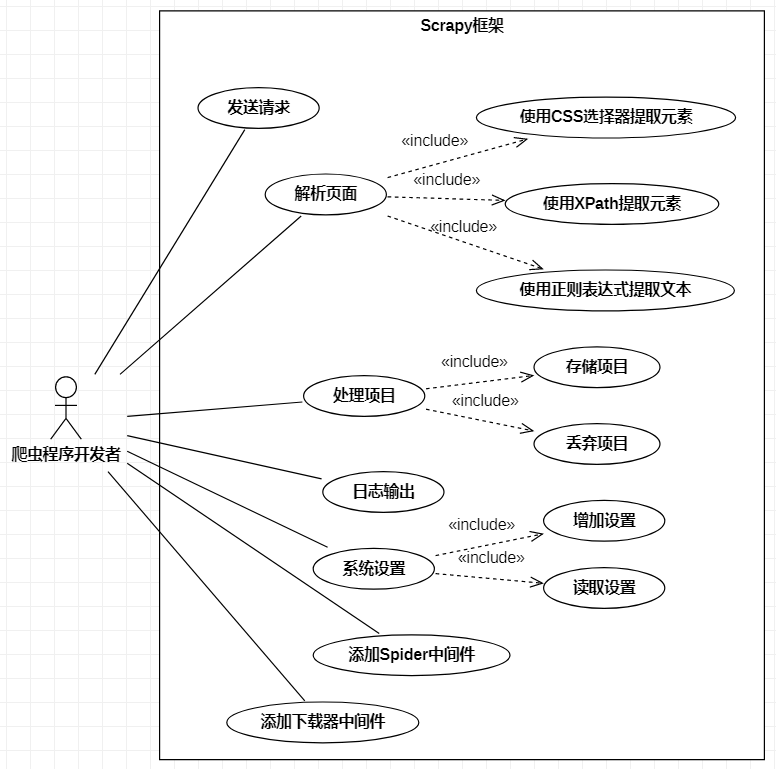


图3.1 Scrapy框架用例图

3.2.2 Web UI

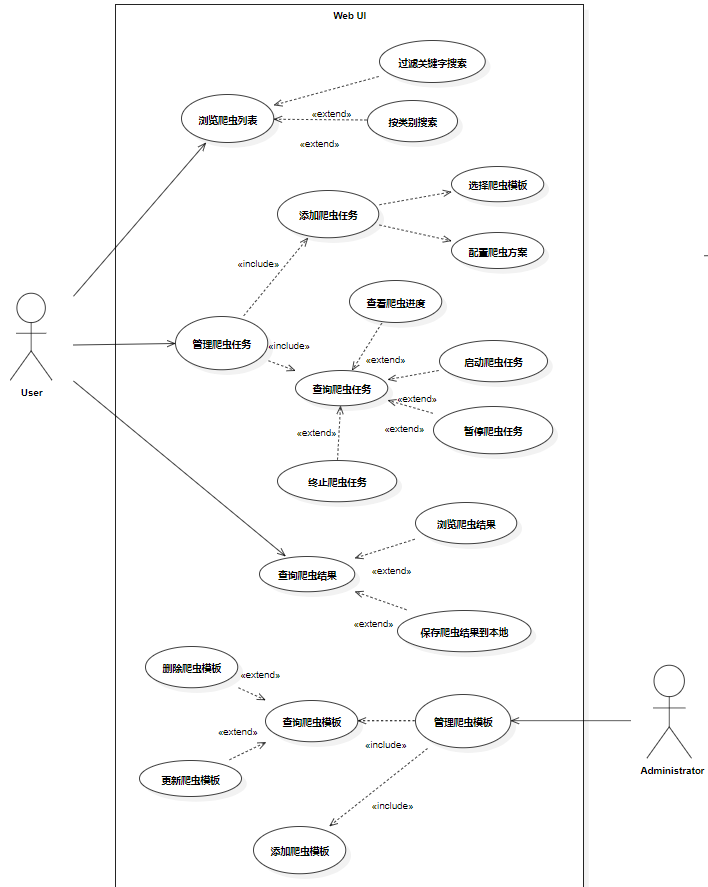


图3.2 EasySpider Web UI用例图

3.2.3 后端及调度程序



图3.3 EasySpider后端及调度程序用例图

3.2.4 Scrapyd



图3.4 Scrapyd用例图

3.3 RUCM图

3.3.1 Scrapy框架



图3.5 发送请求功能RUCM图



图3.6 使用CSS选择器提取元素功能RUCM图



图3.7 使用XPath提取元素功能RUCM图



图3.8 使用正则表达式提取文本功能RUCM图



图3.9 解析页面功能RUCM图

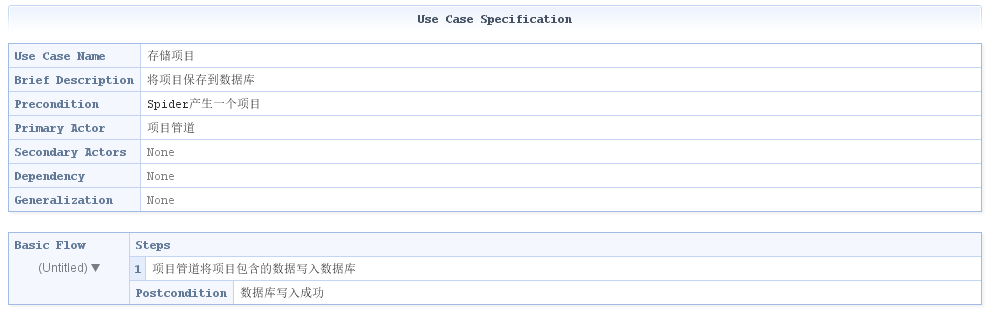


图3.10 存储项目功能RUCM图

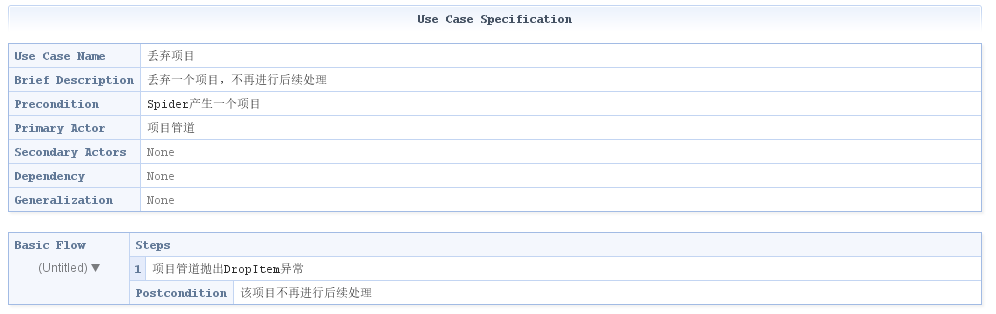


图3.11 丢弃项目功能RUCM图

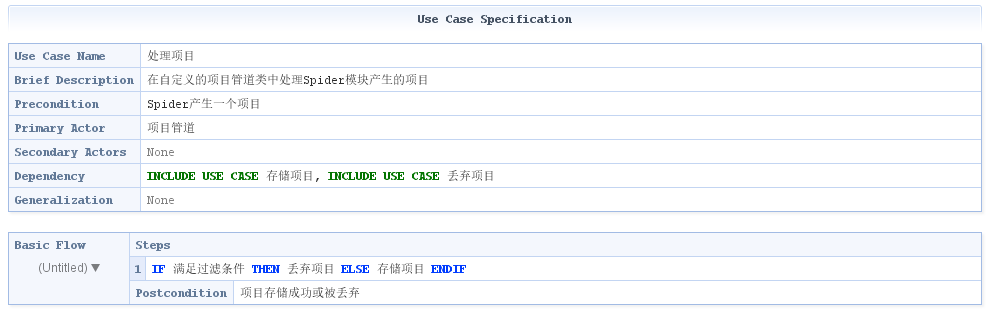


图3.12 处理项目功能RUCM图



图3.13 日志输出功能RUCM图



图3.14 增加设置功能RUCM图



图3.15 读取设置功能RUCM图



图3.16 系统设置功能RUCM图



图3.17 添加Spider中间件功能RUCM图



图3.18 添加下载器中间件功能RUCM图

3.3.2 Web UI



图3.19 浏览爬虫列表功能RUCM图



图3.20 过滤关键字搜索功能RUCM图



图3.21 添加爬虫任务功能RUCM图



图3.22 选择爬虫模板功能RUCM图



图3.23 配置爬虫方案功能RUCM图



图3.24 查看爬虫进度功能RUCM图



图3.25 启动爬虫任务功能RUCM图



图3.26 暂停爬虫任务功能RUCM图



图3.27 浏览爬虫结果功能RUCM图



图3.28 查询爬虫任务功能RUCM图

3.3.3 后端及调度程序

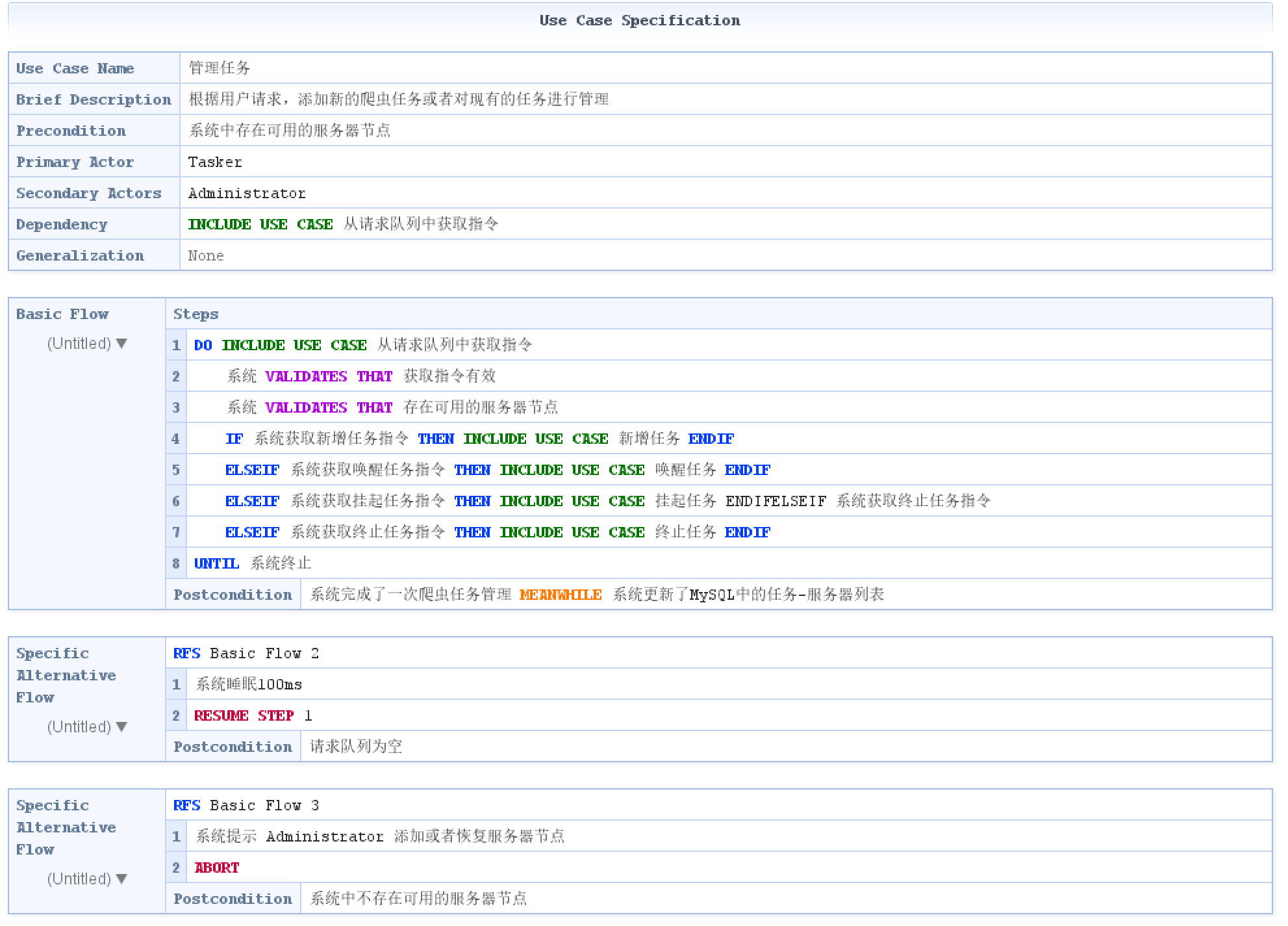


图3.29 管理任务功能RUCM图



图3.30 新增任务功能RUCM图



图3.31 挂起任务功能RUCM图



图3.32 管理服务器节点功能RUCM图



图3.33 新增服务器节点功能RUCM图



图3.34 停用服务器节点功能RUCM图

3.3.4 Scrapyd

A screenshot of a cell phone

Description automatically generated

图3.35 查询守护进程状态功能RUCM图

A screenshot of a cell phone

Description automatically generated

图3.36 列出可用爬虫功能RUCM图

A screenshot of a cell phone

Description automatically generated

图3.37 列出当前作业功能RUCM图

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

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

A screenshot of a cell phone

Description automatically generated

图3.40 列出项目功能RUCM图

A screenshot of a cell phone

Description automatically generated

图3.41 删除项目功能RUCM图

A screenshot of a cell phone

Description automatically generated

图3.42 列出项目版本功能RUCM图

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

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

3.4 非功能需求

3.4.1 Web UI

1. 模板的可扩充性与可维护性

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和可维护性，采用合理的软件设计模式实现模板的增加、删除、更新、查询四种基本操作。

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。

1. 系统的易用性

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，在采集的各个阶段可见可控的基础上实现系统的易用性。

1. 采集数据的完整性和有效性

模板化爬虫程序管理平台应根据不同网站，提供多种网页采集策略与配套资源，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稳定性。

3.4.2 后端



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。

3.5 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。

3.6 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。

3.7 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
* Selenium
* Django
* MySQL
* MongoDB

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