基于Scrapy的WebUI开发

——项目设计文档

SY1906514 汪凌风 SY1906508 明　昊

SY1906513 汪丽萍 SY1906431 郑泽西

SY1806220 闫奕涛 SY1906509 邵志钧

2020/05/04

# 版本记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 修改说明 | 参与者 | 审核人 |
| 1.0.0 | 2020.05.04 | 完成第一版 | 全体成员 | 汪凌风 |

注：全体成员即汪凌风，邵志钧，明昊，汪丽萍，闫奕涛，郑泽西

目　录

[一、引言 5](#_Toc39665647)

[1.1 编写目的 5](#_Toc39665648)

[1.2 背景 5](#_Toc39665649)

[1.3 术语与缩略语 6](#_Toc39665650)

[1.4 参考资料 6](#_Toc39665651)

[二、总体设计 7](#_Toc39665652)

[2.1 用户角色 7](#_Toc39665653)

[2.2 系统类图 7](#_Toc39665654)

[2.3 系统架构 8](#_Toc39665655)

[2.4 运行环境 9](#_Toc39665656)

[2.4.1 开发设备环境 9](#_Toc39665657)

[2.4.2 支持软件环境 9](#_Toc39665658)

[三、模块设计 9](#_Toc39665659)

[3.1 组件图 9](#_Toc39665660)

[3.2 组件介绍 10](#_Toc39665661)

[3.2.1 脚本管理模块类图 10](#_Toc39665662)

[3.2.2 脚本生成模块类图 11](#_Toc39665663)

[3.2.3 编辑调试模块类图 12](#_Toc39665664)

[3.2.4 词云生成模块类图 13](#_Toc39665665)

[3.2.5 选择优化模块类图 13](#_Toc39665666)

[3.2.6 去重优化模块类图 14](#_Toc39665667)

[四、关键功能实现 15](#_Toc39665668)

[4.1 运行脚本顺序图 15](#_Toc39665669)

[4.2 脚本生成顺序图 16](#_Toc39665670)

[4.3 编辑调试顺序图 16](#_Toc39665671)

[4.4 词云生成顺序图 17](#_Toc39665672)

[4.5 选择优化顺序图 18](#_Toc39665673)

[4.6 去重优化顺序图 19](#_Toc39665674)

[五、需求追踪 19](#_Toc39665675)

[5.1 脚本管理需求追踪 19](#_Toc39665676)

[5.2 脚本生成需求追踪 20](#_Toc39665677)

[5.3 编辑调试需求追踪 20](#_Toc39665678)

[5.4 词云生成需求追踪 21](#_Toc39665679)

[5.5 选择优化需求追踪 21](#_Toc39665680)

[5.6 去重优化需求追踪 21](#_Toc39665681)

# 一、引言

本软件Scrapy是开源爬虫框架。本项目设计文档详细描述了基于Scrapy框架的WebUI开发的技术路线与详细实现。

## 1.1 编写目的

本项目设计文档，是为了软件设计、软件测试人员和用户所编写的。

本项目设计文档的适用读者，包括参加能力验证的开发测试人员、Scrapy技术人员，以及项目的其他相关人员。

## 1.2 背景

项目名称：基于Scrapy的WebUI开发

Scrapy的组织机构：Scrapy开源项目开发组

Scrapy的实施机构：Github站点上的359位贡献者

项目背景：本项目是一个高速并发的网络爬虫框架，用于对网站进行爬网并从其页面提取结构化数据。

## 1.3 术语与缩略语

　　Engine，Downloader等组件在下面的组件介绍中有解释。

|  |  |
| --- | --- |
| 爬虫 | 是一种按照一定的规则，自动地抓取[万维网](https://baike.baidu.com/item/%E4%B8%87%E7%BB%B4%E7%BD%91/215515" \t "_blank)信息的程序或者脚本。 |
| Item | 项目，它定义了爬取结果的数据结构，爬取的数据会被赋值成该Item对象。 |
| Selectors | 选择器，Scrapy有自己的数据提取机制，它们被称为选择器。因为它们“选择”HTML文档的某些部分 XPath 或 CSS 表达。 |
| Scraper | 刮取器，主要作用是对Spider中间件进行管理，通过中间件完成请求、响应、数据分析等工作。 |
| Request | 请求，是URL经过Engine组件封装后的，Downloader组件即将接收的内容 |

表1 术语定义

## 1.4 参考资料

[1] 中国国家标准化委员会.计算机软件需求规格说明：GB/T 9385-2008 [S].

[2] 中国国家标准化委员会.计算机软件文档编制规范：GB/T 8567-2006 [S].

[3] Scrapy documentation [EB/OL]. https://docs.scrapy.org/en/latest/

[4] RUCM documentation [EB/OL]. http://www.zen-tools.com/tools/rucm.html

[5] Flask documentation [EB/OL]. <https://flask.palletsprojects.com/en/1.0.x/>

其他参考资料：

　　Web编码规范：https://www.html.cn/archives/5361

Bloom-Filter算法相关参考：

<https://blog.csdn.net/kun1280437633/article/details/80643879>

<https://blog.csdn.net/z434890/article/details/85330117>

Scrapy组件图例：

<http://www.jianshu.com/p/078ad2067419>

# 二、总体设计

## 2.1 用户角色

本系统具备在线编辑调试与自动生成模版功能，因此根据用户是否在本系统上编写代码，将用户群体划分为代码用户和非代码用户。

1. 代码用户

代码用户是指需要使用本系统代码编辑调试功能的用户。代码用户具备一定的爬虫代码编写能力，可以自行对爬虫脚本根据自身需要进行编辑。

1. 非代码用户

非代码用户是指使用本系统自动生成模版功能的用户。非代码用户不具备爬虫代码编写能力，使用本系统根据自身需求生成的模版直接对网页进行爬取。

## 2.2 系统类图

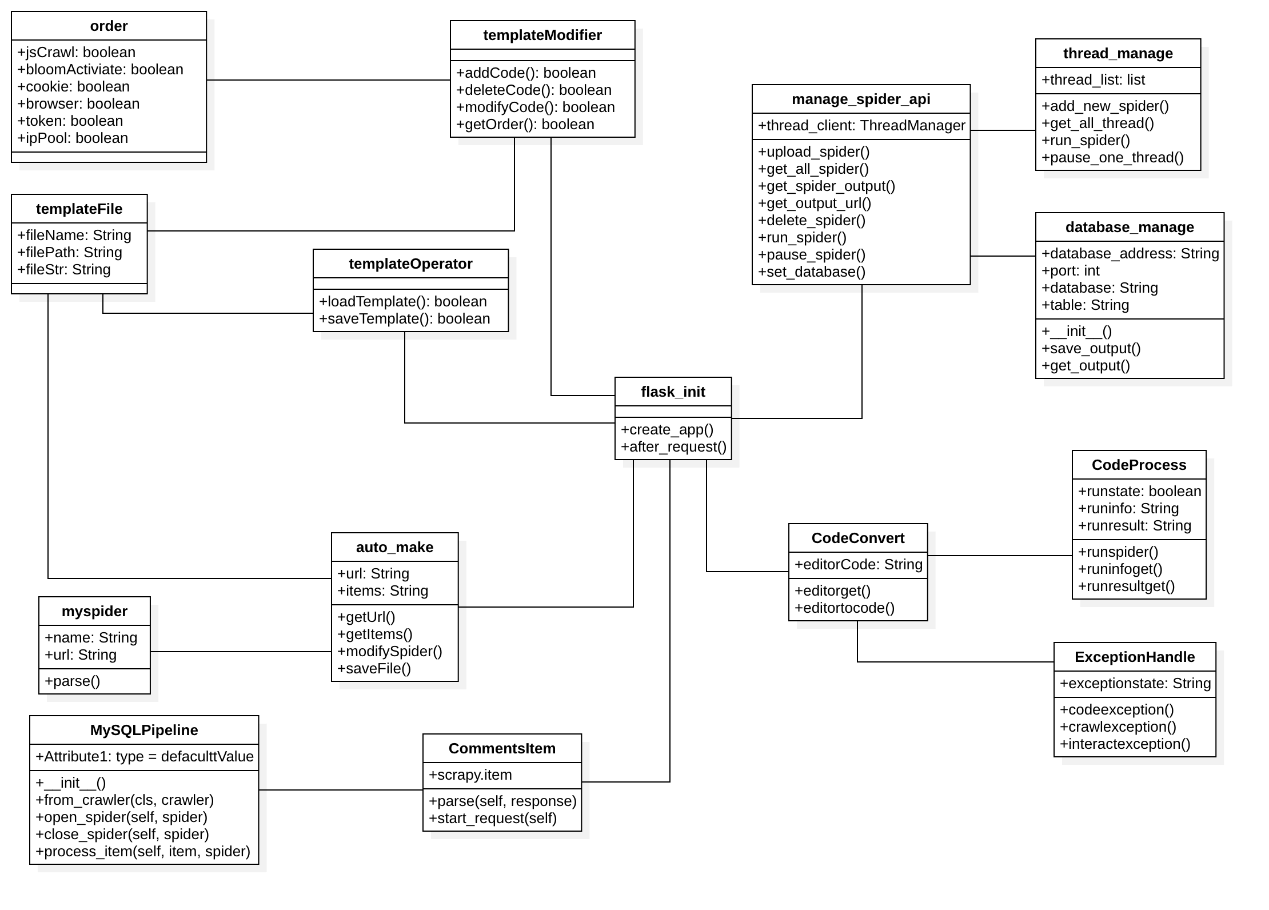


图1 系统总类图

　　图1为本系统后端系统的总类图。系统中包括15个类，其中templateModifier类、templateOperator类、auto\_make类、CommentsItem类、CodeConvert类、manage\_spider\_api类是控制类，flask\_init类是后端系统的核心，对后端系统的各个组件进行调用。本系统对原生Scrapy框架做出了性能优化，因为该部分与后端系统不属于一部分，因此该部分的类图独立于后端系统总类图在后续章节中给出。

## 2.3 系统架构

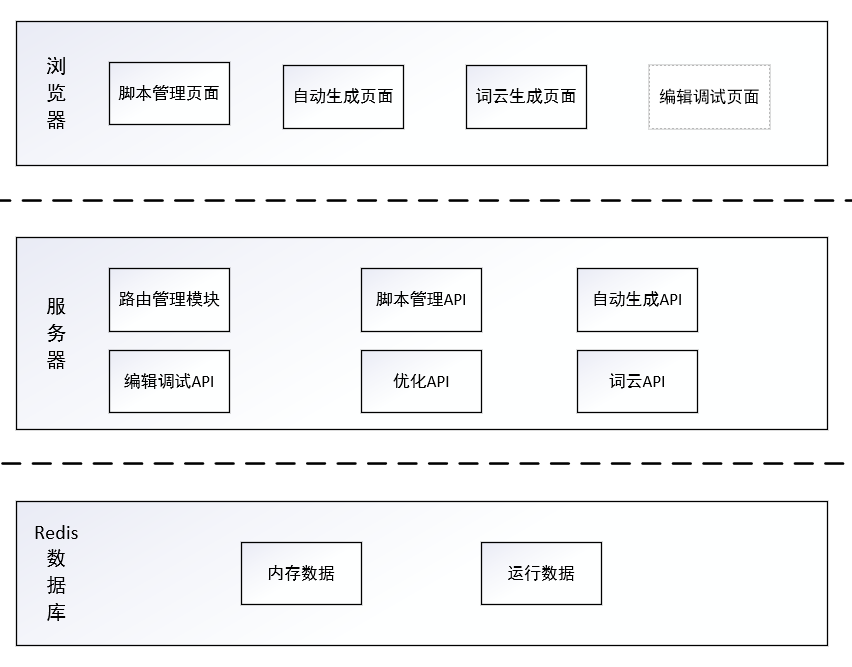


图2 系统架构图

　　WebUI的实现系统架构图如图2所示。本系统采用的是非常经典的B/S架构，有前端浏览器，后端服务器以及数据库支撑组成。前端采用vue编写浏览器页面与JS代码。主要分为脚本管理页面，自动生成页面，词云生成页面和编辑调试页面。优化功能在脚本管理页面提供优化选项，没有单独的展示页面。服务器主要采用flask编写，前端的每个页面模块在后端有专门负责处理的API模块。另外后端还需要提供路由管理模块与优化API以保证功能的实现与正常运行。最后，本系统需要依靠redis提供底层的数据保存与使用，主要用于脚本管理与优化阶段。

## 2.4 运行环境

### 2.4.1 开发设备环境

　　系统：Linux、Mac、Windows

　　内存：256MB以上

硬盘：20G以上

用于开发Python爬虫的PC机或网络服务器

Scrapy、jieba、WordCloud等Python类库

　　依赖Flask 1.1.1提供的Web程序框架

### 2.4.2 支持软件环境

　　从用户角度出发，正常的浏览器即可访问服务器。

# 三、模块设计

## 3.1 组件图

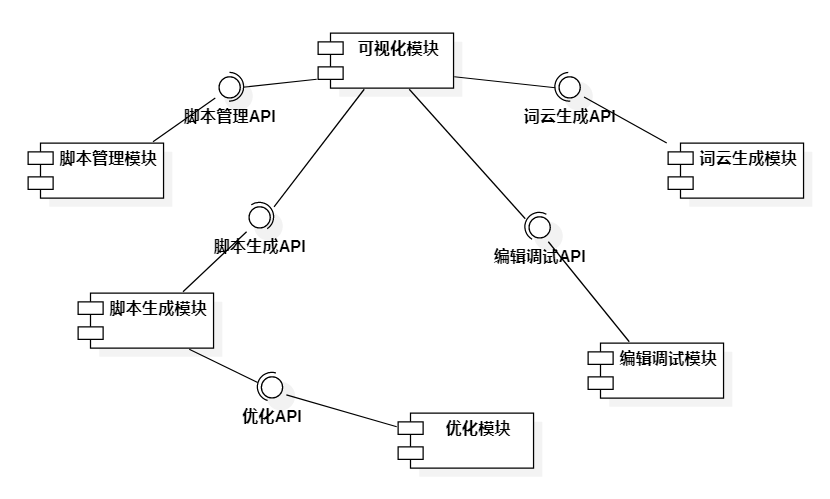


图3 WebUI模块组件图

　　基于Scrapy的WebUI的模块组件图如图3所示，整体模块组件以可视化模块为基础，脚本管理模块、脚本生成模块、编辑调试模块、词云生成模块通过向可视化模块提供应用接口来完成在整体框架下的统一，使得用户可以在可视化界面下自由选择并使用其中需要的功能，而优化模块通过向脚本生成模块提供应用接口，完善了脚本生成模块的功能，这样的设计使得各个模块组件间耦合较低，可拓展性较强。

　　具体每个模块组件的功能按照最终完善的项目规格说明书来进行定义与设计。脚本管理模块提供给用户可以在web界面上对spider脚本进行简单的管理的功能，调用者可以通过该模块提供的API在当前界面选择自定义脚本进行运行，同时可以查看脚本的运行状态、运行时间与特定脚本的结构化输出，如果需要，也可以进行下载与保存进数据库。选择优化模块会根据用户选择的优化选项对目标URL进行爬取，同时提供本次爬取的结果以及用时等信息。脚本生成模块则提供半自动化生成脚本的功能，调用者设定输入的url和标签，该模块可以套用模板生成简单可用的scrapy爬取脚本，同时该模块可以通过调用优化API来在生成的脚本中实现优化脚本性能的目的。编辑调试模块提供在线编辑、编译与调试功能，使得爬虫的编写、调试都可以在 Web界面中进行。词云生成模块提供对爬取的文本解析后生成词云的功能。而可视化模块则通过上述模块接口的调用将这些组件集成统一在一个共同的WebUI框架下，这样的功能设计使得可以很方便在这个框架下添加新的构件实现新的功能，满足了可扩展性。

## 3.2 组件介绍

### 3.2.1 脚本管理模块类图

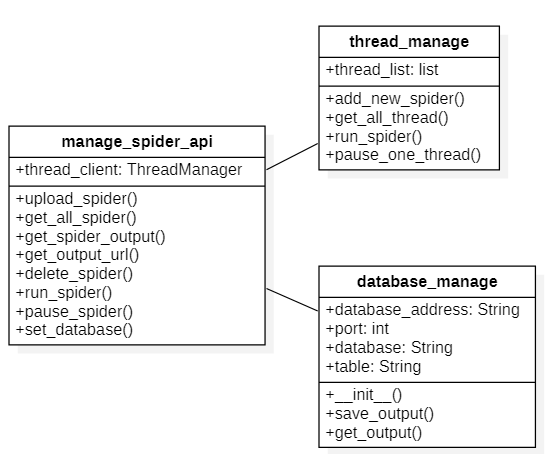


图4 脚本管理模块类图

　　脚本管理模块详细类图如图4所示，脚本管理模块主要进行脚本的上传，脚本的运行，脚本的输出展示和一些记录的保存。总的来说脚本管理模块除了一个专门实现后端API的控制类外，还需要实现两个控制类，分别是进程监视类和数据库管理类。进程监视类主要为了实现多脚本的运行和脚本的暂停功能。该类维护一个进程列表用以保存当前正在运行的进程和对应脚本的信息。并且向外提供获取当前所有的进程暂停指定进程的接口。脚本执行的输出在用户没有配置数据库的时候会保存在本地，文件管理类提供访问的接口。数据库管理类需要在用户在WebUI界面完成数据库设置之后才会进行使用。数据库管理主要进行脚本输出的保存工作和读取数据库保存的输出。

### 3.2.2 脚本生成模块类图

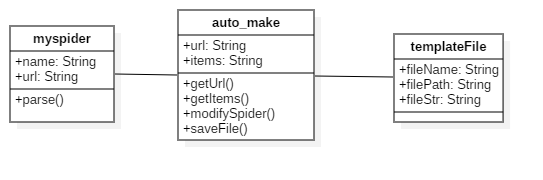


图5 脚本生成模块类图

　　如图5自动化生成脚本组件类图所示，自动化生成脚本组件由myspider、auto\_make、templateFile三个类组成。自动化生成脚本组件根据用户输入的url和items套用模板生成爬取脚本。其中myspider类是初始的脚本模板，包含了脚本名，爬取链接和爬取方法，由auto\_make接口从前端页面获取用户输入的url和items，并修改初始的脚本模板，生成脚本文件，并将脚本文件保存为templateFile，保存的文件包含文件名，文件路径和文件内容。

### 3.2.3 编辑调试模块类图

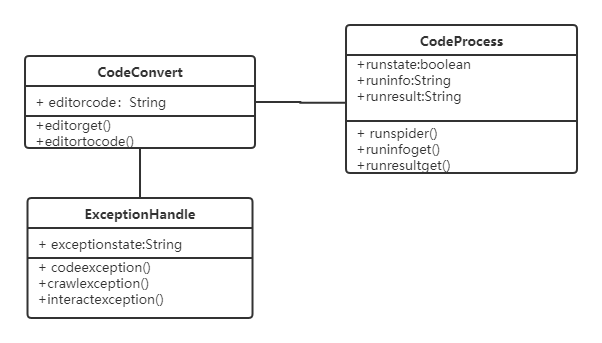


图6 编辑调试模块类图

如上述编辑调试组件类图所示，编辑调试组件由CodeConvert、CodeProcess、ExceptionHandle三个类组成。编辑调试模块提供在线编辑、编译与调试功能，使得爬虫的编写、调试都可以在 Web界面中进行。其中CodeConvert类用于获取Web界面在线编辑器中的代码字符串并将其转化为可运行的代码格式，获取在线编辑器中的代码字符串主要由editorget函数实现，而editortocode则完成将editorget返回的字符串转换为可运行的python代码；CodeProcess类则用于对代码运行结果的处理，runspider函数实现对转换后的代码的运行，而runinfoget和runreseltget则分别输出运行的一些相关信息（如当前爬取网站的url和响应码）和用户想要爬取的结果信息，runstate，runinfo，runresult则分别存储运行的状态信息，一些相关信息和想要爬取的结果；ExceptionHandle类则负责运行过程中的一些错误与异常处理，codeexception，crawlexception，interactexception分别负责在线编辑中内容向代码转换，爬虫爬取，与用户交互过程中的一些错误与异常处理。

### 3.2.4 词云生成模块类图

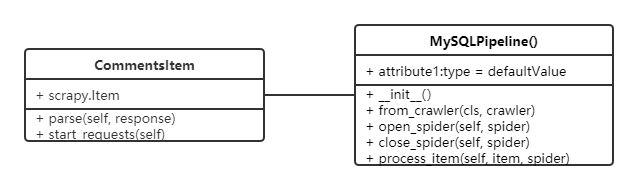


图7 词云生成模块类图

在item.py中定义一个item，叫做CommentsItem，在这个类里定义了4个字段，包括评论的用户名、打分、赞同数、评论内容以及评论时间，另外还有table，代表MySQL存储的表名称。根据URL获取网页源码后，通过CSS选择器来提取信息。将读取的数据保存在MySQL数据库中，在pipelines.py文件中添加类MySQLPipeline()。在数据库中读取所有评论数据，使用jieba进行分词，根据停用词词典屏蔽一些无意义的词云，例如“今天”、“那时”等。通过WordCloud库生成词云，simfang.ttf为必要的字体库。在运行结果中，字号越大，代表出现的频率越高。

### 3.2.5 选择优化模块类图

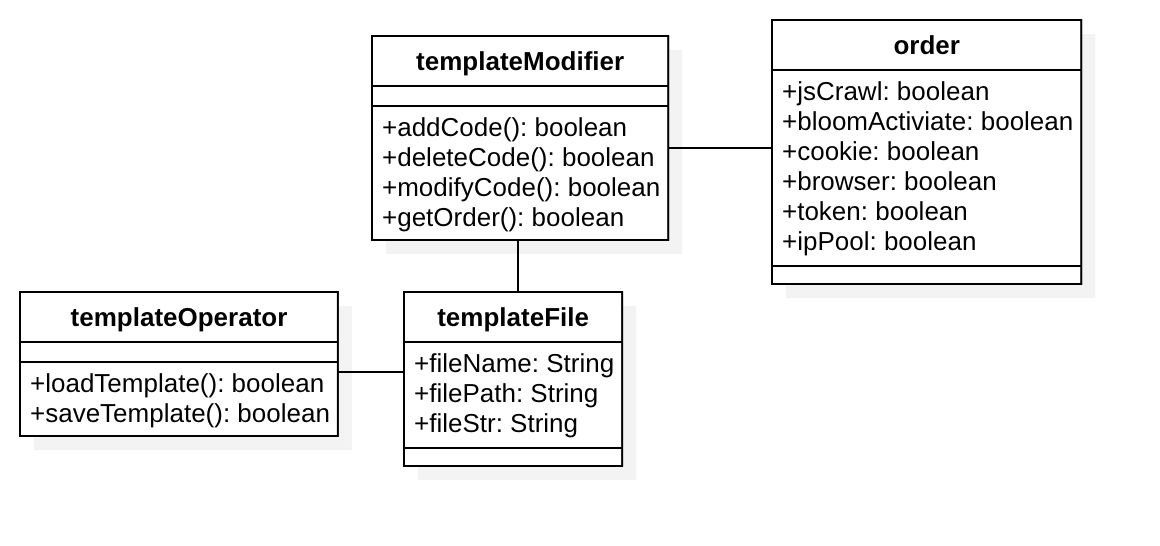


图8 选择优化模块类图

选择优化组件由templateModifier、order、templateOperator、templateFile四个类组成。选择优化组件根据用户的优化选择，通过修改脚本代码的形式对用户的所选项目进行性能优化。其中templateFile类用于表示当前所需优化的文件，其中包括文件的命名、路径以及文件的内容；templateOperator类用于对目标文件进行存取操作；order类则记录了用户选择的优化项目，作为后续优化的根据；templateModifier类根据order类中用户的选择对代码文件进行增加、删除、修改操作。

### 3.2.6 去重优化模块类图

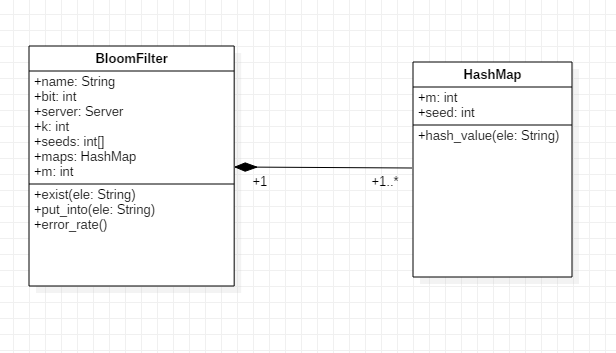


图9 去重优化模块类图

Scrapy的URL去重策略是使用Python自带的Set去重，这是一种原始有效的方法，它的优点在于算法简单且运行速度较快，但面对大数据量的时候，这一算法的弊端就开始显现。为了优化去重机制，将原先的使用Set变为现在的使用Bloom过滤器去重，现在原框架中加入BloomFilter.py文件，其中包含两个类：HashMap类和BloomFilter类。

1. **HashMap类**

　　HashMap类是存储URL特征的数据结构，其中有两个变量：m，位数组的位数（默认为千万级）；seed，用于产生URL映射的种子值。一个函数hash\_value(ele)，传入参数为URL经过Scrapy计算后的fingerprint，用于计算每个fingerprint的hash值。

1. **BloomFilter类**

BloomFilter是根据HashMap算出的hash值进行去重的过滤器。其中有7个变量：bit：设置位数组位数的值，位数为2的bit次方；m：位数组位数，由1左移bit位得到，将传入HashMap类中；name：设置redis中位数组的key值；server：用于配置redis；k，hash函数的数目；seeds，1-k的List，用于生成k个HashMap；maps：生成的HashMap组成的List。

BloomFilter中有三个函数：exist()：用于判定某个请求是否已经存在；put\_into()：将URL经过hash函数映射得到的位数组对应位置置为1；error\_rate()：用于给出指定数据量、hash函数个数和位数组位数下的预期错误率。

# 四、关键功能实现

## 4.1 运行脚本顺序图

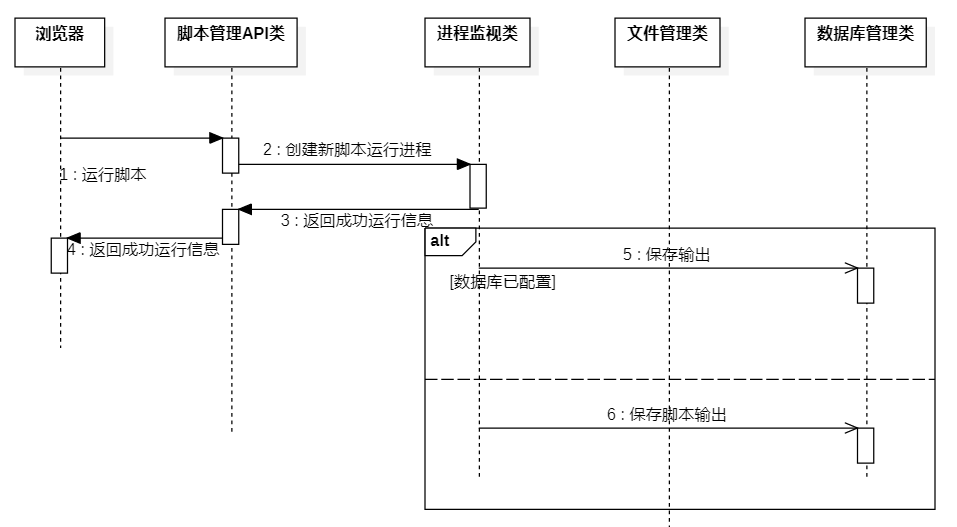


图10 运行脚本顺序图

运行脚本的顺序图如图10所示，WebUI界面选择了一个脚本进行运行之后，脚本管理API类接受这个请求，进行一系列鲁棒性检查之后，调用进程监视类创建新脚本运行进程运行，同时更新进程列表。之后返回成功运行的信息，脚本管理API在接受到成功运行信息之后，就可以向WebUI返回脚本成功运行的信息。运行进程根据数据库是否完成配置，在运行阶段选择是将输出保存到本地文件还是保存到用户指定的数据库内。

## 4.2 脚本生成顺序图

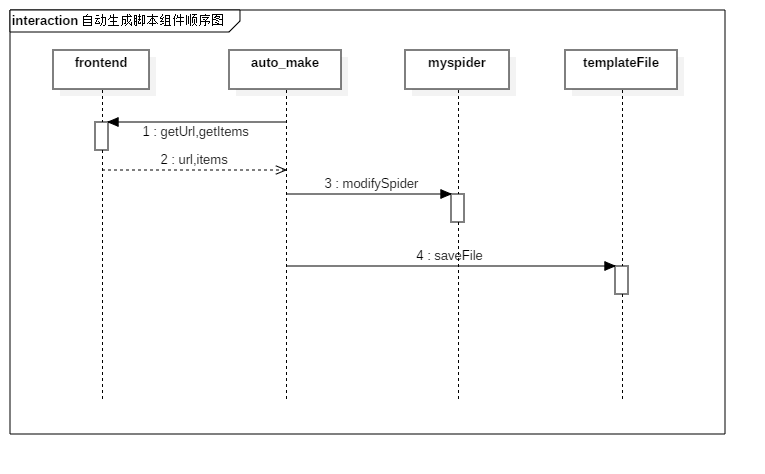


图11 脚本生成顺序图

　　如图所示为自动化生成脚本组件运行的顺序图。首先控制类auto\_make会从前端表单获取url和items，接着，auto\_make根据获取的url和items修改myspider脚本，然后将修改后的脚本文件保存为templateFile文件。

## 4.3 编辑调试顺序图

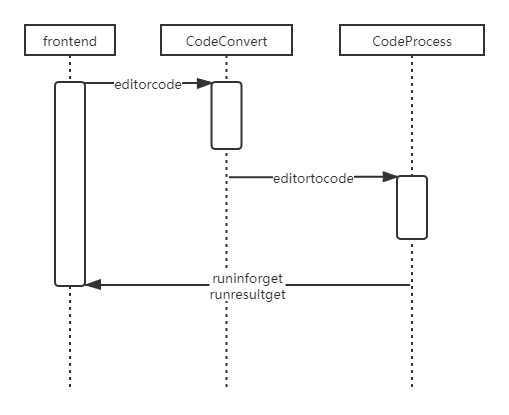


图12 编辑调试顺序图

如上图编辑调试组件运行的顺序图所示。首先控制类CodeConvert会调用editorget函数获取在线编辑器中的代码字符串，然后调用editortocode函数完成将editorget返回的字符串转换为可运行的python代码，并将转换后的代码内容传递给CodeProcess类；CodeProcess类调用runspider函数实现对转换后的代码的运行，而runinfoget和runreseltget则分别输出运行的一些相关信息（如当前爬取网站的url和响应码）和用户想要爬取的结果信息，并将其输出到前端页面，至此完成一次代码的编辑运行功能。

## 4.4 词云生成顺序图

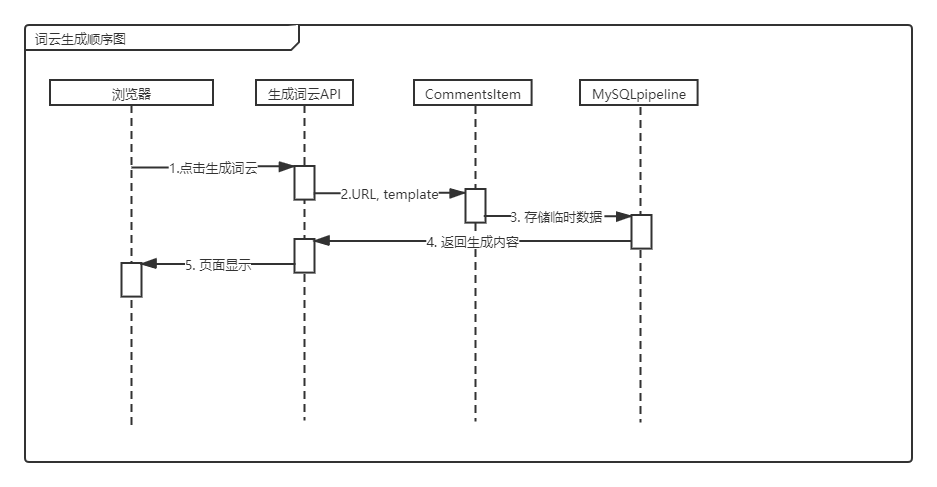


图13 词云生成顺序图

　　如图13所示，用户在浏览器页面输入URL并选择生成的词云模板后，点击“生成词云”，生成词云API便会接受到这些数据，传递给CommentsItem类，CommentsItem类会将从网页获取的数据传递给MySQLpipeline中，这一部分的数据是临时存储的，根据所选择的模板，使用WordCloud等库函数，生成对应的词云，词云内容返回到生成词云的API，最终在浏览器的页面上显示。

## 4.5 选择优化顺序图

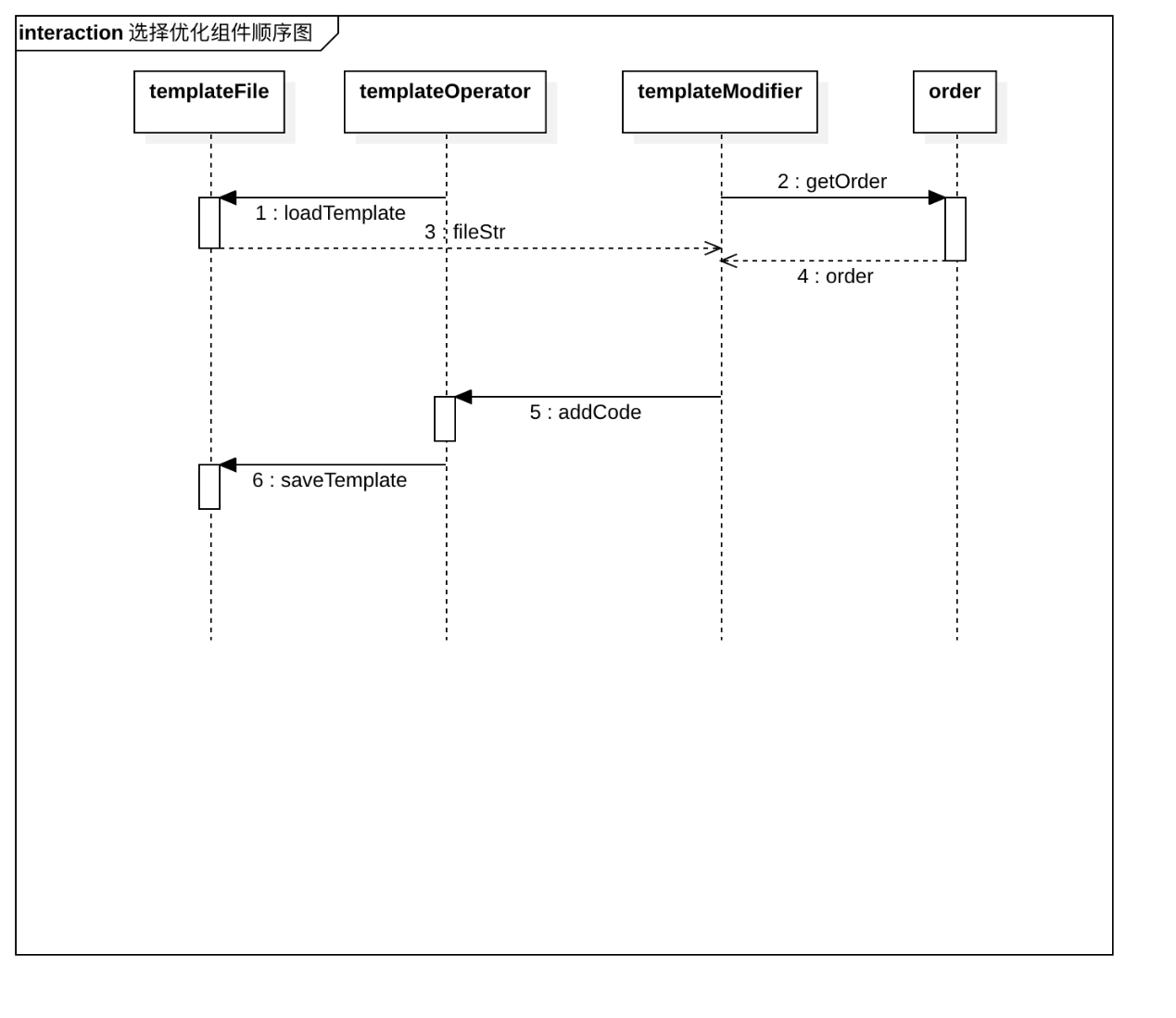


图14 选择优化顺序图

选择优化组件运行的顺序图如图14。首先控制类templateOperator会根据脚本生成组件给出的文件路径加载文件代码，之后templateModifier根据Order类中用户的优化要求对代码进行增删改。完成对代码的修改后将代码重新保存至模版。

## 4.6 去重优化顺序图

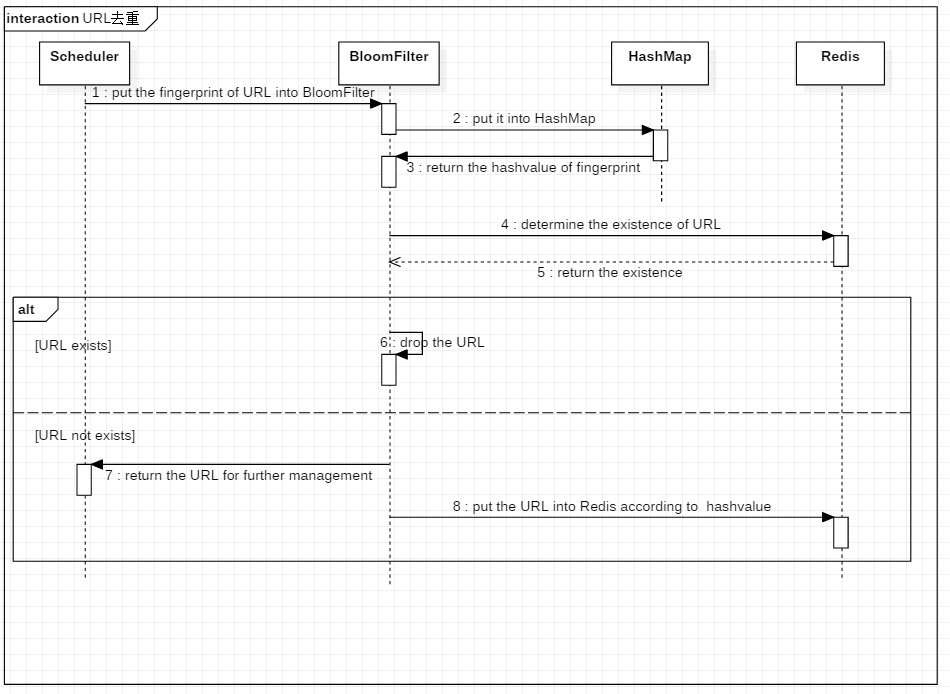


图15 去重优化顺序图

# 五、需求追踪

## 5.1 脚本管理需求追踪

|  |  |
| --- | --- |
| 功能 | 设计 |
| 运行脚本 | 脚本管理API类的运行脚本()调用进程监视类的创建新脚本运行进程(0 |
| 查看结果 | 脚本管理API类的获取脚本输出()读取本地文件或者数据库管理类的读取输出() |
| 管理记录 | 脚本管理API类的获取当前所有脚本()会读取返回对应的本地脚本记录 |
| 下载结果 | 脚本管理API类的获取输出下载链接()返回输出的下载链接或者跳转error页面 |
| 连接数据库 | 数据库管理类的所有函数都用来实现本块供能 |

表2 脚本管理需求追踪

## 5.2 脚本生成需求追踪

|  |  |
| --- | --- |
| 功能 | 设计 |
| 自动化生成脚本 | auto\_make类的modifySpider()修改myspider  auto\_make类的saveFile()保存生成的脚本文件 |

表3 脚本生成需求追踪

## 5.3 编辑调试需求追踪

|  |  |
| --- | --- |
| 功能 | 设计 |
| 在线编辑 | 编辑调试API类的CodeConvert ()调用editorget函数获取在线编辑器中的代码字符串 |
| 在线运行 | 编辑调试API类的CodeConvert ()调用editortocode函数将editorget返回的字符串转换为可运行的python代码，然后调用CodeProcess类的runspider函数实现对转换后的代码的运行。 |
| 在线调试 | 编辑调试API类的CodeProcess类调用runinfoget函数输出运行的一些相关信息（如当前爬取网站的url和响应码）runinfo |
| 异常处理 | 脚本管理API类的获取输出下载链接()返回输出的下载链接或者跳转error页面 |
| 结果输出 | 编辑调试API类的ExceptionHandle类调用codeexception，crawlexception，interactexception函数分别负责在线编辑中内容向代码转换，爬虫爬取，与用户交互过程中的一些错误与异常处理。 |

表4 编辑调试需求追踪

## 5.4 词云生成需求追踪

|  |  |
| --- | --- |
| 功能 | 设计 |
| 生成词云 | CommentsItem类定义了4个字段，根据URL获取网页源码后，通过CSS选择器来提取信息。 |

表5 词云生成需求追踪

## 5.5 选择优化需求追踪

|  |  |
| --- | --- |
| 功能 | 设计 |
| 性能优化 | Order类  templateModifier类addCoder()  templateModifier类modifyCode() |

表6 选择优化需求追踪

## 5.6 去重优化需求追踪

|  |  |
| --- | --- |
| 功能 | 设计 |
| URL去重 | BloomFilter类先使用exist()函数判断URL是否已经存在，如存在则不处理，不存在则通过HashMap提供的映射关系将redis中的位数组进行填充。 |
| 查看预期错误率 | BloomFilter类使用error\_rate()函数计算当前参数和数据量下的预期错误率。 |

表7 去重优化需求追踪