护城项目

爬虫基础功能模块设计

版本号：V1.2

|  |  |  |  |
| --- | --- | --- | --- |
| 拟制: | 丁永强 | 日期： | 2016.5.6 |
| 审核: |  | 日期： |  |
| 批准: |  | 日期： |  |

**文档修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修订版本** | 日期 | **作者** | 修改内容描述 | **备注** |
| V1.2 | 2016.5.6 | 丁永强 | 调整爬虫框架结构设计 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 引言 4](#_Toc450308059)

[1.1 目的 4](#_Toc450308060)

[1.2 预期读者 4](#_Toc450308061)

[2 爬虫模块设计 4](#_Toc450308062)

[2.1 爬虫整体结构 4](#_Toc450308063)

[2.1.1 基础模块类库 4](#_Toc450308064)

[2.1.2 爬虫基类 5](#_Toc450308065)

[2.1.3 爬虫子类 5](#_Toc450308066)

[2.1.4 模块调度类 5](#_Toc450308067)

[2.2 各模块设计 5](#_Toc450308068)

[2.2.1 爬虫基类/子类包图 5](#_Toc450308069)

[2.2.2 爬虫类设计图 6](#_Toc450308070)

[2.2.3 爬虫基类 7](#_Toc450308071)

[2.2.4 爬虫子类 7](#_Toc450308072)

[2.2.5 模块类 7](#_Toc450308073)

[2.2.6 模块调度类 8](#_Toc450308074)

[2.3 典型的爬虫开发流程 8](#_Toc450308075)

[2.3.1 开发CrawlerBase 8](#_Toc450308076)

[2.3.2 开发Crawler子类 9](#_Toc450308077)

[3 小结 10](#_Toc450308078)

[3.1.1 复用性 10](#_Toc450308079)

[3.1.2 可靠性 10](#_Toc450308080)

[3.1.3 开发和维护 11](#_Toc450308081)

# 引言

## 目的

通过爬虫基础模块的设计和开发，提高代码复用性，通过横向模块拆分，使功能模块之间强内聚、低耦合，有助于解决现有爬虫代码逻辑不统一、代码复用率低、维护工作量大的问题。

## 预期读者

护城项目爬虫开发人员、产品经理、项目管理人员。

# 爬虫模块设计

## 爬虫整体结构

爬虫整体结构如下图所示，分为爬虫基类，爬虫子类，爬虫基础模块库和模块调度类。



### 基础模块类库

提供通用模块的封装并为爬虫基础模块库和爬虫基类提供支持，提供诸如Proxy、Browser、Log、Monitor、DB、Mail、验证码识别等功能。

此处列出用于展示其与爬虫基础模块库和爬虫基类的依赖关系，本质不属于爬虫模块

### 爬虫基类

通常为某一垂直领域的爬虫基类，例如企业信息的爬虫基类为QyxxCrawler，相关信息的爬虫基类为XgxxCrawler等。该基类可为子类提供运行时需要的所有依赖，例如运行时的状态、中间结果、通用功能模块支持等。

基类提供所有爬虫所共用的通用方法，例如读取首页获取令牌，读取验证码，获得搜索列表，读取公司基本信息等。

基类各通用方法之间相互独立，每个模块专注于自身的业务逻辑，使其更易于使用和维护。

### 爬虫子类

负责具体垂直领域下的具体站点爬取，例如对于企业信息下的北京企业信息站点的爬取。由于经过爬虫基类的两次抽象和封装，爬虫子类的实现将变得简单且易于维护。

子类中仅需要关注爬取流程中需要经过哪些步骤，每一步所需的输入参数，期望的输出参数，以及各种情况下的休眠策略、重试策略等。

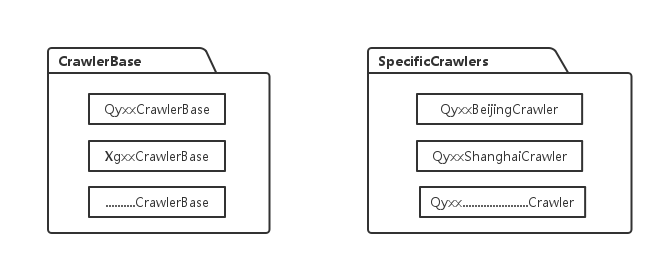
### 模块调度类

负责子类配置中的功能模块调度，根据配置逐步获取输入参数，解析结果，获取输出参数，处理各种情况下的跳转等。

## 各模块设计

### 爬虫基类/子类包图

爬虫基类和子类Package中类的组织结构如下图所示。

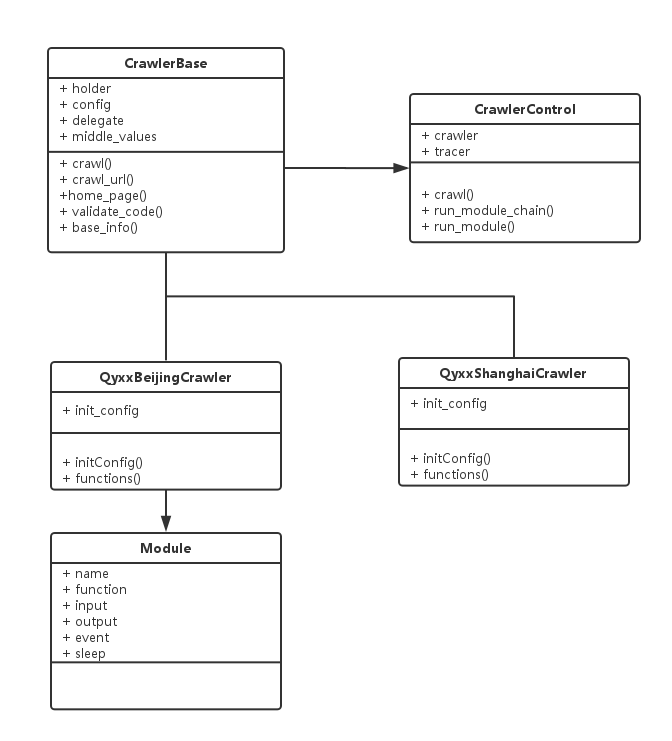


爬虫基类Package中包含各个领域垂直爬虫系统的基类，例如企业信息信用公示领域爬虫基类QyxxCrawlerBase、相关信息领域爬虫基类XgxxCrawlerBase等。

爬虫子类按不同垂直领域分不同的包，上图为企业信息垂直领域包的组织情况，每个具体的Crawler对应一个具体网站的爬虫。

### 爬虫类设计图

爬虫类设计如下图所示。



### 爬虫基类

爬虫基类设计的指导思想是高内聚、低耦合、高复用，将不同的对外处理逻辑切分为独立的处理单元，每个处理单元为一个独立的模块方法，各方法之间完全解耦，模块方法之间不用关心其他模块方法是否存在，关注并处理好指定的处理逻辑。每个模块方法在接口不变的情况下，可以各自独立优化和使用，各个模块方法可在复用过程中不断优化和打磨，沉淀为可靠的基础组件。

模块方法的基本组成包括验证码识别、令牌获取、搜索列表获取和解析，基本信息获取和解析等。

#### 属性

holder 持有所有与业务无关的功能对象，如proxy、log等，当外部对象发生变化时仅需修改holder而不影响业务相关代码

config 持有运行配置文件

delegate 将执行流程委托给外部模块调度类

middle\_values 持有各种在运行期产生的中间状态、中间值等，例如在调用链条中存放各种输入输出的value\_dict，存放阶段性结果的result\_dict等

#### 方法

crawl 通过keyword抓取公司信息入口

crawl\_url 通过url抓取公司信息入口

home\_page、validate\_code、base\_info 通用业务功能方法

### 爬虫子类

爬虫子类的设计充分考虑了易用性，尽可能地减少子类中的重复开发量

#### 属性

init\_config 持有该子类运行需要的配置文件

#### 方法

initConfig 该子类各模块配置的初始化方法

functions 代表该站点特有处理方法的一组实现

### 模块类

模块类为模块的配置提供模型

#### 属性

name模块名称

function 模块对应的模块方法

input 输入参数

output 输出参数

event 模块事件，配置了在某个事件发生后，应该从哪个模块开始重新执行，重试次数，重试失败后如何处理

sleep 可配置有条件或无条件休眠时间

#### 方法

提供一组快速添加模块参数的方法，例如添加input、output、event、sleep等

### 模块调度类

实现对模块方法的通用调度

#### 属性

crawler 委托调度操作给调度类的爬虫

tracer 用于各模块方法执行过程的全链路跟踪

#### 方法

crawl 爬行调度入口

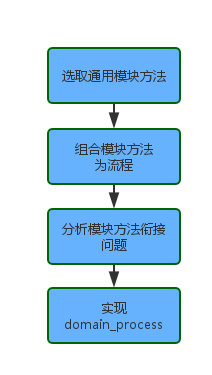
run\_module\_chain 执行模块链入口

run\_module 执行某个模块入口

## 典型的爬虫开发流程

### 开发CrawlerBase

具体流程如下图所示



1. 从现有CrawlerBase模块方法中选取适合本领域的通用模块

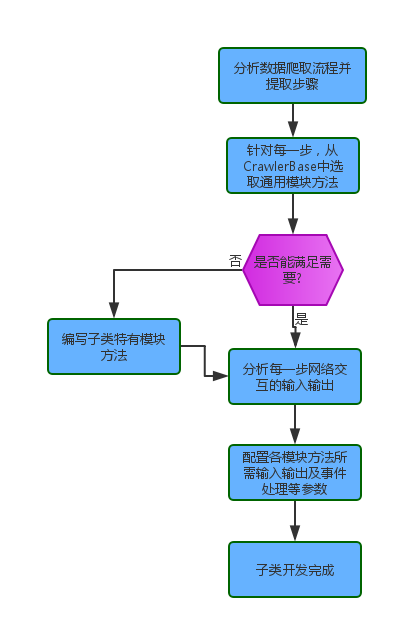
2. 根据业务需要和网站实际访问路径和要求组合处理流程

3. 对某些模块间不能衔接或达不到要求的部分进行分析

4. 实现该领域内通用方法domain\_process

### 开发Crawler子类

子类开发流程如下图所示



1. 分析并提取网站爬取过程中需要经历的流程

2. 从CrawlerBase中选取通用方法

3. 若CrawlerBase中的通用方法无法满足每一步处理的需要，则需要编写子类特有的模块处理方法

4. 分析每一步网络交互的参数组成和结果提取

5. 详细配置子类的处理模块

# 小结

### 复用性

由于所有功能模块方法被独立切分，各个模块方法可以保持独立优化、独立发展，使其尽可能支持各种复杂场景，同时任何爬虫的开发都可优先选用已有模块方法，从而提高其复用性。

### 可靠性

各个功能模块方法经过被不断使用和优化，在使用过程中发现的通用模块相关问题，一处修改处处生效，而且经过长期使用和打磨，会使通用模块不断趋于可靠和稳定。

### 开发和维护

从整个开发过程来看，所有通用方法尽可能向上抽象，各个子类的负担大大减轻，子类必需实现的仅有模块配置，从而可以快速提供领域内不同站点的爬虫，同时由于核心处理逻辑的质量在通用模块方法进行保障，各站点爬虫出现逻辑问题的概率也大大降低。