目录

[1 简介 2](#_Toc402366433)

[1.1 引用术语与缩写解释 2](#_Toc402366434)

[1.2 概要 2](#_Toc402366435)

[1.3 应用设计目标 2](#_Toc402366436)

[2 架构说明 3](#_Toc402366437)

[2.1 模块说明 4](#_Toc402366438)

[2.1.1 服务入口层 SpiderService.py 4](#_Toc402366439)

[2.1.2 任务管理层 SpiderTask.py 5](#_Toc402366440)

[2.1.3 规则读取层 RuleReader.py 5](#_Toc402366441)

[2.1.4 文章转存层 ArticleStorer.py 6](#_Toc402366442)

[2.1.5 文章爬虫层 Spider 7](#_Toc402366443)

[2.1.6 数据模型层 Model 9](#_Toc402366444)

[2.1.7 消息模型层 Message 10](#_Toc402366445)

[3 部署说明 11](#_Toc402366446)

[3.1 运行环境 11](#_Toc402366447)

[3.2 资源目录 11](#_Toc402366448)

[3.3 部署步骤 11](#_Toc402366449)

[3.3.1 Python 11](#_Toc402366450)

[3.3.2 MSMQ 11](#_Toc402366451)

[3.3.3 DataSource Driver 12](#_Toc402366452)

[3.3.4 Pyodbc 12](#_Toc402366453)

[3.3.5 Pywin32 12](#_Toc402366454)

[3.3.6 Pillow 12](#_Toc402366455)

[3.3.7 Requests 12](#_Toc402366456)

[3.3.8 SQLAlchemy 12](#_Toc402366457)

[3.3.9 SQL Server 13](#_Toc402366458)

[3.3.10 Spider Service 13](#_Toc402366459)

[3.4 配置参数 13](#_Toc402366460)

[3.4.1 图片转存API 13](#_Toc402366461)

[3.4.2 数据库连接 13](#_Toc402366462)

[3.4.3 消息队列 14](#_Toc402366463)

[3.4.4 压图API 14](#_Toc402366464)

[3.4.5 图片缓存路径 14](#_Toc402366465)

[3.4.6 日志设置 14](#_Toc402366466)

[3.4.7 爬虫规则路径 14](#_Toc402366467)

[4 扩展说明 15](#_Toc402366468)

[4.1 扩展范围 15](#_Toc402366469)

[4.2 扩展示例 15](#_Toc402366470)

[4.2.1 规则扩展 15](#_Toc402366471)

[4.2.2 模板扩展 17](#_Toc402366472)

[4.2.3 功能扩展 20](#_Toc402366473)

# 简介

## 引用术语与缩写解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
| Python | 一种简洁而强大的解释型脚本语言 |
| pyodbc | Python下的ODBC数据库访问组件 |
| SQLAlchemy | Python下的ORM数据访问组件 |
| pywin32 | Python下的Win32接口访问组件 |
| requests | Python下的Web访问组件 |
| Pillow | Python下的图像处理组件 |
| 解释型语言 | 无需编译源码可敏捷部署并执行的语言 |
| IOC | 控制反转，运行时可由数据反向干涉程序执行逻辑的设计模式 |
| RegularExpression | 正则表达式，查找和替换文本模式的简洁而灵活的表示法 |
| XML | 扩展标记语言，用于配置及数据交互 |

## 概要

本文档针对以下三个方面进行了详细说明

1. 架构说明，对新闻网络爬虫的核心架构进行描述，供开发人员在开发或升级应用时参考；
2. 部署说明，对新闻网络爬虫的部署步骤进行描述，供部署人员进行应用部署或升级时参考；
3. 扩展说明，对新闻网络爬虫的扩展模式进行描述，供开发人员扩展爬虫功能时参考。

## 应用设计目标

* 易于扩展
* 易用
* 易于维护

为了达成这些设计目标我们采用了以下设计机制：

1. 易于扩展：

* 使用解释型、面向对象的Python语言实现程序逻辑：解释型语言易于扩展部署，结合抓取模块的IOC机制，更新升级时无需停机；面向对象易于代码复用，创建新爬虫模块时仅需少量重载代码修改。
* 控制流程IOC化，利用XML配置文件，可动态增减爬虫任务、控制爬虫任务启动条件、更改爬虫规则以及爬虫逻辑。

1. 易用：

* 服务化，爬虫任务被封装为Windows 服务，可便捷地启动与停止，于后台执行长期运行，

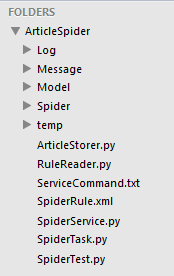
1. 易于维护：

* 编程语言简洁，缩进式风格减少了语法性代码，动态数据类型减少了声明性代码，函数式编程特性减少了重复性函数代码，友好而功能强大的标准库与外部组件也减少逻辑复杂性。
* 数据访问层模型化：使用SQLAlchemy组件，对数据访问进行对象化建模，屏蔽数据访问细节，如SQL，数据模型易于维护。

# 架构说明

新闻网络爬虫程序主题为一系列python脚本，通过文件夹进行模块隔离，基于命名约定进行动态逻辑加载，根目录为ArticleSpider。

整体框架如下：



SpiderService.py： 服务入口模块，用以处理Windows服务Article Spider Service的安装、卸载、启动、停止与重启。

SpiderTask.py： 任务管理模块，负责加载控制规则配置、安排爬虫任务计划、组合爬虫任务子逻辑。

ArticleStorer.py： 文章转存模块，包含数据库访问、图片转存与切图、队列消息发送功能。

RuleReader.py： 规则读取模块，用于读取爬虫规则，辅助IOC机制。

Spider： 爬虫逻辑模块，核心模块群，可根据需要添加新爬虫模板，爬虫模板可继承，基模块为Spider.py，多个相似爬虫可根据规则设置复用同一个爬虫模板。

Model： 数据模型模块，维护爬虫相关ORM数据模型，由上下文管理层、数据模型层与事务逻辑层组成。

Message： 消息处理模块，主要负责封装与发送队列消息。

SpiderRule.xml： 爬虫规则配置，XML格式元数据。

Temp： 缓存目录，用以缓存转存完成前的中间文件，如下载图片。

Log： 日志目录，用以保存日志，采用循环日志模式。

ServiceCommand.txt： 服务入口命令，用于参考的爬虫服务基本人机交互命令。

SpiderTest.py： 爬虫测试模块，用于测试的相关脚本。

## 模块说明

### 服务入口层 SpiderService.py

#### SpiderService

win32serviceutil.ServiceFramework：

服务基类，引入自pywin32组件，用以将Python程序封装位Windows服务。

SvcDoRun：

服务启动入口，重载基类方法，用以开启SpiderTask任务管理线程，阻塞等待任务结束事件。

SvcStop：

服务终止入口，重载基类方法，用以终止SpiderTask任务管理线程，发起任务结束事件。

#### ServiceCommand

python SpiderService.py install：

爬虫服务安装，必须指令，用以注册Windows服务，安装成功后可直接于Windows服务管理器中进行服务启动、停止等管理操作。

python SpiderService.py install --startup auto：

自启动爬虫服务安装。

python SpiderService.py start：

启动爬虫服务，服务安装后有效。

python SpiderService.py restart：

重启爬虫服务，服务启动后有效。

python SpiderService.py stop：

停止爬虫服务，服务启动后有效。

python SpiderService.py remove：

删除/卸载爬虫服务，服务安装后有效。

### 任务管理层 SpiderTask.py

#### SpiderTask

threading.Thread：

线程管理基类，引入自python标准库，用以将主任务进行线程化封装。

\_\_init\_\_：

初始化方法，进行一系列任务初始化操作，如线程初始化、日志初始化、调度初始化、存储初始化等操作。

ScheduleTask：

任务调度方法，读取爬虫规则，根据设置生成爬虫调度计划。

RunSpiderByRule：

爬虫驱动方法，按照给定规则，驱动对应爬虫开启任务，核心步骤为，爬虫选用—文章抓取—图片转存—文章入库—后续处理（如压图与消息通知）。

run：

任务子线程启动入口，重载基类方法，以日为周期进行调度-执行-休眠循环，苏醒后调度爬虫任务—按照调度计划处理任务（执行或等待）—计划完成后休眠至下一周期苏醒。

StopTask：

任务终止方法，当前任务完成后终止任务计划，非强行中断任务子线程，若要强行中断，可直接结束主线程，任务子线程将自动中断。

### 规则读取层 RuleReader.py

#### RuleReader

\_\_init\_\_：

初始化方法，确定规则读取模式（目前仅支持XML模式），模式有效时进行初始规则读取。

FreshRules：

规则刷新方法，读取最新规则，默认以XML格式读取规则，若要采用其他方法（如数据库读取、Json接口读取等），可继承该基类后进行重载扩展。

#### GetRuleFromNode

功能性函数，从XML节点中提取爬虫规则字典，属性及简单子节点直接提取入本级字典，复杂子节点递归提取为子级字典后加入本级字典。

#### PrintNode

调试用函数，用于打印XML节点数据，采用前序遍历法。

#### SpiderRule

爬虫规则字典（dict类型），存储爬虫规则参数，以福州旅游网爬虫规则为例：

name： 规则名称，通常以爬取来源站命名，如福州旅游网。

sourceId： 来源标识，参照文章来源枚举值，默认为0。

rule： 子级明细规则字典。

url： 来源Url，明细规则，如<http://lyj.fuzhou.gov.cn/lyjzwgk/lydt/index.htm>。

reAbstract： 文章摘要正则表达式，明细规则，扫描文章清单时使用，如<li>.+?<span>\[(.+?)\].+?href="(.+?)".+?>(.+?)</a>.+?</li>。

reArticle： 文章正文正则表达式，明细规则，扫描文章正文时使用，如<div class="content-boxtext">(.+?)</div>\s\*<div class="content-boxbottom">。

reImage： 文章图片正则表达式，明细规则，扫描文章图片时使用，如<IMG.+?src="(.+?)".+?/>

module： 爬虫模块名，明细规则，反射加载的爬虫模块名与类名，如FuZhouSpider。

regionId： 目的地标识，明细规则，爬虫目的地对应的乐途目的地字典标识值，如130。

spiderName： 爬虫编辑ID，明细规则，利用爬虫对内发布文章的虚拟编辑用户名，如jishutest。

isValid： 有效标志，明细规则，启用该规则时为1，停用为0，禁用为-1。

minPage： 最小页码，明细规则，分页爬取时第一页的页码参数，默认为-1（不分页）。

maxPage： 最大页码，明细规则，分页爬取时最后页的页码参数，默认为-1（不分页）。

wakeTime： 子级苏醒时间字典，明细规则，可包含多个时间点。

timePotX： 时间点，X代表时间点标识，多个时间点标识不可相同，时、分、秒必须。

### 文章转存层 ArticleStorer.py

#### ArticleStorer

文章转存器，组织下层通信、数据模块进行数据交互，如数据库访问、队列消息发送、文件上传、远程接口调用等。

\_\_init\_\_：

初始化方法，设定图片转存API（imageApi）、数据库连接（dbConStr）、消息队列（msmqPath）及压图API（picCutApi）等通信参数，并进行初始化。

DumpImage：

图片转存方法，转存文章正文中已缓存的的下载图片至图片转存API，同时关联转存图片路径。

DumpArticle：

文章入库方法，将已经完成图片转存并替换Url的文章正文按照正文页入库，同时记录爬取关联信息。

NewArticleCheck：

新文章检查方法，比对爬取关联信息，确定是否为新文章或可更新文章。

SendSuccessMessage：

后期处理消息信息发送方法，向消息队列发送信息，通知后续处理程序处理刚刚发布或变更的新正文页。

CutImaages：

正文页压图方法，调用压图接口对指定正文页正文内的图片进行压图处理。

ArticleRepublish：

正文页重发布方法，将正文页重新发布，即向消息队连续发送一条删除发布请求与新发布请求。

### 文章爬虫层 Spider

该层为目录，包含多个XSpider.py模块，其中Spider.py为基础模块，其他模块如FuZhouSpider.py为基于Spider.py模块的扩展的模板化模块。

#### Spider

爬虫基类，封装爬虫所需的基本操作，可由子类继承复用（如规则加载、HTML下载、图片下载等共同操作）或重载（文章抓取、图片替换等区别操作）。

ReadRule：

规则加载方法，可继承复用，可嵌套重载，加载爬虫规则字典内参数信息，并对爬虫伪装（Http Header）、数据缓存、数据清理（Css清理、Herf清理等）等参数进行设置。

CatchArticles：

文章抓取方法，不可直接复用，必须替换重载，为各爬虫模板的独有逻辑，目前主要有页面抓取及异步JS抓取两种基本模式。

DownLoadHtml：

Html源码下载方法，可继承复用，可重载(如进行拟人化访问伪装) ，用于获取抓取页面的相关Html源码，如文章列表页、文章正文页以及异步加载API。

DownLoadImage：

图片下载方法，可继承复用，可重载（如应对图片防盗链机制），用于下载文章正文中的图片链接至缓存路径，同时进行图片格式标准化。

ReplaceArticleImages：

缓存文章正文图片替换方法，可继承复用，可重载（如对转存图片标签添加属性），用于将抓取文章正文中原来的图片链接替换为转存后的图片链接。

CacheArticle：

文章信息缓存方法，可继承复用，可重载（如定制文章特有属性信息），用于组装文章属性信息并加入缓存等待下一步处理。

ClearArticles：

文章正文清洗方法，可继承复用，可重载（如添加JS清晰），用于清洗文章正文中的无效信息，如当前已支持的CSS、Herf、空白字符及样式清洗。

ClearTempImages：

缓存图片清理方法，可继承复用，可重载（如添加缓存备份机制），用于清理缓存的已下载图片。

#### Functions

文章爬虫层中的相关功能性函数。

ReplaceImage：

图像Url替换函数，可将文章正文中的正则匹配项（原始图片链接）替换为转存图片链接。

ClearExternalCss：

CSS清理函数，可将文章正文中的正则匹配项（Css类）清空。

ClearExternalHerf：

Herf清理函数，可将文章正文中的正则匹配项（超链接）去链接处理。

ClearExternalBlank：

空白字符清理函数，可将文章正文中的正则匹配项（空白字符）去除。

ClearExternalStyle：

样式清理函数，可将文章正文中的正则匹配项（style样式）去除。

ComposeUrl：

相对Url组装函数，将页面中的相对Url结合页面Url组装为绝对Url。

ConvertImageToJpg：

图片格式标准化函数，将不同格式的图片（如PNG、BMP）同意转化为JPG格式。

#### XSpider

各种爬虫模板类，通常直接继承自Spider，但也可继承其他XSpider，区别主要在于CatchArticles方法的重载实现，目前主要分为两种模式。

页面爬虫：

CatchArticles方法直接解析页面源码，根据制定的编码格式，提取文章关键信息，由扫描列表页的文章摘要、扫描正文页的正文、扫描正文域的图片三步组成，典型模板如FuZhouSpider。

Json Api爬虫：

CatchArticles方法获取Json API响应，将结果反序列化为字典对象，提取文章关键信息，由提取列表API的文章摘要、提取正文API的正文与图片两步组成。，典型模板如FuZhouTourSpider。

其他类型的模板可根据实际情况自行扩展，如XML API、SOAP API等。

### 数据模型层 Model

该层为目录，包含多个SpiderContext.py、SpiderEntity.py与SpiderData.py三个模块。

SpiderContext.py：

数据上下文模块，负责数据库连接、数据模型初始化、会话管理。

SpiderEntity.py：

数据模型模块，负责数据实体对象模型映射，即表与类的映射。

SpiderData.py：

数据逻辑模块，负责组织会话内的数据访问逻辑，也是对外接口。

#### SpiderDataHelper

爬虫数据访问类，属于数据逻辑模块。

SaveArticle：

文章入库方法，将爬去并完成过滤的文章信息写入数据库，若是新文章，文章信息入库同时写入文章导入信息，若不是新文章（导入信息已存在），仅进行修改。

NewArticleCheck：

新文章检查方法，用于防止文章重复导入，对比文章导入信息，若不匹配（根据文章Url与发布日期），则认定为新文章，否则认定为重复文章。

GetArticlesLikeDampSquibs：

未成功发布文章扫面方法，用于查询出已发布但未成功的文章，返回必要属性信息，如正文页ID，目的地ID等。

#### ModelMapper

实体关系映射（类表映射）函数，属于数据模型模块，将指定的实体类与数据表绑定，当前已映射对如下：

Cms\_Information—Cms\_Information：

正文页信息类与正文页信息表。

Cms\_Information\_Inported—Cms\_Information\_Inported：

正文页导入信息类与正文页导入信息表。

Cms\_InformationRegion—Cms\_InformationRegion

正文页目的地关联类与正文页目的地关联表。

#### ModelContext

数据模型上下文类，属于数据上下文模块，管理数据连接上下文。

\_\_init\_\_：

上下文初始化方法，注册数据库连接、初始化数据模型元数据并建立会话生成器。

Session：

会话入口，申请一个数据库会话。

#### Session\_Scope

数据会话域函数，属于数据上下文模块，负责隔离会话事务的生命周期，具有contextmanager特性，即以迭代生成器的方式隔离会话事务逻辑无关的细节，确保成功自动提交、失败自动回滚并且结束自动释放。

### 消息模型层 Message

该层为目录，包含SpiderMessageQueue.py模块，负责格式化消息，并与消息队列通信。

#### Message

消息类，负责消息格式化，将消息转化为制定输出格式。

\_\_init\_\_：

初始化方法，将消息字典初始化为XML结构，同时添加必要属性。

ToFormatString：

序列化方法，将XML结构的消息转化为字符串，同时添加必要说明性内容。

#### ToXmlElement

功能性函数，将字典转化为XML结点格式，简单键值对直接转化为子节点，复杂键值对进行嵌套转化，值数组转化为多个子节点，值为字典则进行递归转化。

#### MessageQueue

消息队列访问类，封装消息队列接口。

\_\_init\_\_：

初始化方法，组装MSMQ队列基本信息，包含主机名与队列路径。

SendMessage：

消息发送方法，根据给定MSMQ队列基本信息，创建会话并发送消息。

#### Queue\_Scope

队列会话域函数，负责隔离队列会话的生命周期，具有contextmanager特性，即以迭代生成器的方式隔离队列会话逻辑无关的细节，确保会话结束自动释放。

# 部署说明

## 运行环境

PC配置： 2G以上内存

操作系统： Windows XP Professional 2002 Service Pack 3 +

相关工具： Python 3.3

Pyodbc 3.0.7

SQLAlchemy 0.9.7

pywin32 219

requests 2.4.1

Miscrosoft Message Queue

Miscrosoft SQL Server 2005

SQL Server Native Client 10.0

## 资源目录

源码路径，192.168.80.157主机 E:\shuaxin\ArticleSpider。

工具路径：192.168.80.157主机 E:\tools\爬虫项目部署包

## 部署步骤

### Python

确认部署主机python-3.x版本已安装，建议使用python-3.3稳定版本。

若未安装，执行爬虫项目部署包内python-3.3.5.1395976247.msi文件，以安装python虚拟机，安装目录可设置为E:\tools\Python33。

确保系统环境路径（高级系统设置-环境变量）含有python安装目录，确保路径内可引用python.exe。

### MSMQ

确保Miscrosoft Message Queue已开启，且存在消息队列路径\private$\queuepathnews，且该队列对EveryOne开放消息发送权限，若该队列不存在，则依赖部署条件不满足（后续处理程序未部署），不可进行应用部署。

### DataSource Driver

确保主机ODBC数据源中已安装SQL Server Native Client 10.0驱动程序（版本可以更高，但ArticleStorer.py中dbConStr也应对应修改）已正确安装。

若未安装，根据系统环境，选择执行爬虫项目部署包内sqlncli\_X86.msi或sqlncli\_X64.msi，以安装数据源驱动。

### Pyodbc

确保python安装目录（如E:\tools\Python33）下，存在以下相对路径的目录Lib\site-packages\pyodbc…，即Pyodbc已安装。

若未安装，根据系统环境，执行爬虫项目部署包内pyodbc-3.0.7.win32-py3.3.exe，以安装Python ODBC组件。

### Pywin32

确保python安装目录（如E:\tools\Python33）下，存在以下相对路径的目录Lib\site-packages\pythonwin，即Pywin32已安装。

若未安装，根据系统环境，执行爬虫项目部署包内pywin32-219.win32-py3.3.exe，以安装Python Win32组件。

### Pillow

确保python安装目录（如E:\tools\Python33）下，存在以下相对路径的目录Lib\site-packages\PIL，即Pillow已安装。

若未安装，根据系统环境，执行爬虫项目部署包内Pillow-2.6.0.win32-py3.3.exe，以安装Python Image Library组件。

### Requests

确保python安装目录（如E:\tools\Python33）下，存在以下相对路径的目录Lib\site-packages\requests，即Requests已安装。

若未安装，启动cmd，切换至爬虫项目部署包下requests-2.4.1目录，执行命令python setup.py install，以安装Python Requests组件。

### SQLAlchemy

确保python安装目录（如E:\tools\Python33）下，存在以下相对路径的目录Lib\site-packages\sqlalchemy，即SQLAlchemy已安装。

若未安装，启动cmd，切换至爬虫项目部署包下SQLAlchemy-0.9.7目录，执行命令python setup.py install，以安装Python SQLAlchemy组件。

### SQL Server

确保81主机上的lotour库，已存在Cms\_Information\_Inported表。

若不存在，执行初始化脚本：

CREATE TABLE [dbo].[Cms\_Information\_Inported] (

[SourceUrl] VARCHAR (256) NOT NULL,

[Status] SMALLINT NOT NULL,

[InformationId] INT NOT NULL,

[SourceTime] DATETIME NOT NULL,

[RegionId] INT NOT NULL,

[SourceName] VARCHAR(50) NULL,

PRIMARY KEY CLUSTERED ([SourceUrl] ASC)

);

CREATE NONCLUSTERED INDEX [IX\_Cms\_Information\_Inported\_InformationId]

ON [dbo].[Cms\_Information\_Inported]([InformationId] ASC);

### Spider Service

检查服务管理程序中，是否存在Article Spider Service服务，即爬虫服务是否已安装。

若未安装，选定部署路径，将ArticleSpider目录移动至对应路径，启动cmd，切换至部署目录，执行命令python SpiderService.py install，以安装爬虫服务。

应用扩展升级时无须重新安装，除非变更SpiderTask.py，亦无须停止Article Spider Service服务，直接添加或替换对应源文件即可（次日生效），但重启服务可确保变更立即生效。

进入服务管理程序，启动Article Spider Service，若启动成功，整个部署流程完成。

## 配置参数

### 图片转存API

ArticleStorer.py中的imageApi参数，指向CMS正文页的图片上传接口，默认为'http:// localhost:8037/WS/newsUploadImage.ashx'。

### 数据库连接

ArticleStorer.py中的dbConStr参数，对应Lotour库的数据连接字符串，默认为'mssql+pyodbc:// testUser:test@localhost:1433/news?driver=SQL Server Native Client 10.0'，其中mssql表示DBMS为Microsoft Sql Server，pyodbc表示驱动模式为Python ODBC，testUser为数据库用户名，test为用户密码，@至?间区域表示数据库路径，?后区域表示数据库驱动名称。

### 消息队列

ArticleStorer.py中的msmqPath参数，指向CMS正文页的发布消息队列，默认为'\\PRIVATE$\\queuepathnews'。

### 压图API

ArticleStorer.py中的picCutApi参数，指向CMS正文页的压图接口，默认为' http://cms.lotour.com:8343/WS/newsCutImage.ashx'。

### 图片缓存路径

Spider.py中的temp参数，指向文章爬取过程中下载中间图片的缓存目录，默认为相对路径'\\temp\\'。

### 日志设置

SpiderTask.py中的logging.basicConfig函数的相关入口参数，作为爬虫日志设置，可参考python官方文档中的logging部分，当前使用循环日志，部分参数如下：

filename， 日志文件名，默认使用相对路径'\\Log\\spider.log；'

mode， 文件使用模式，默认使用添加模式，即'a'；

maxBytes， 循环日志块大小，默认为2M；

backupCount，循环日志块数量，默认为8；

encoding， 日志编码，默认为'utf-8'；

format， 日志信息格式，用于将Trace信息与Message信息格式化，默认为'%(asctime)s %(levelname)-10s[%(filename)s:%(lineno)d(%(funcName)s)] %(message)s'。

level， 日志记录级别，默认为DEBUG，线上部署推荐使用WARN或ERROR。

### 爬虫规则路径

SpiderTask.py中的rulePath参数，默认为XML文件路径，默认值为相对路径'\\SpiderRule.xml'。

# 扩展说明

## 扩展范围

网络爬虫服务扩展分为三个级别，分别为规则扩展、模板扩展以及功能扩展，以应对不同的扩展需求。

规则扩展：

改动规则文件，即SpiderRule.xml，用于爬虫规则微调，以及添加仅需复用爬虫模板的新爬虫（如同站新频道，或同网站架构）。

模板扩展：

新增爬虫实现，即添加XSpider.py，继承Spider基类，重载实现特有逻辑（如文章抓取逻辑），必要时可单独添加独立功能，如防盗链破解功能。

功能扩展：

变更或重组爬虫功能，任何文件都可能改动，请在理解架构的前提下进行，如将规则元数据由XML元数据变更为可维护更大规模元数据的数据库元数据表，或者将爬虫服务重组以添加文章智能过滤层。

## 扩展示例

### 规则扩展

规则扩展仅需修改SpiderRule.xml文件，以福州旅游网爬虫为例：

<rule name="福州旅游网" sourceId="0">

<url>http://lyj.fuzhou.gov.cn/lyjzwgk/lydt/index.htm</url>

<reAbstract>&lt;li&gt;.+?&lt;span&gt;\[(.+?)\].+?href="(.+?)".+?&gt;(.+?)&lt;/a&gt;.+?&lt;/li&gt;</reAbstract>

<reArticle>&lt;div class="content-boxtext"&gt;(.+?)&lt;/div&gt;\s\*&lt;div class="content-boxbottom"&gt;</reArticle>

<reImage>&lt;IMG.+?src="(.+?)".+?/&gt;</reImage>

<module>FuZhouSpider</module>

<regionId>130</regionId>

<spiderName></spiderName>

<isValid>1</isValid>

<minPage>-1</minPage>

<maxPage>-1</maxPage>

<wakeTime>

<timePot0>08:00:00</timePot0>

<timePot1>13:00:00</timePot1>

</wakeTime>

</rule>

一个Rule节点便是一项爬虫规则，结点属性及子节点含义参照3.1.3.4。

每增加一个爬虫任务，则需要添加对应的Rule节点，添加流程如下：

#### 确定文章来源

>> name

根据抓取需求，将name设置为[站名][-频道名]。

>> sourceId

进入CMS媒体管理模块，检查媒体来源，若不存在现有对应媒体来源，进行添加操作，将sourceId设置为对应媒体来源ID，若不指定媒体来源，忽略sourceId属性或置0；

>> url

观察抓取url对应源代码，若页面含有文章列表，即可按照网页抓取模式处理，url可直接设置为页面url；

若列表内容为异步加载（可参考福州旅游资讯网），此时应分析源代码，找到对应的文章列表API，将url设置为对应API；

若url中含有动态参数或分页参数，可使用{n}参数顺序替代（从{0}开始），在抓取逻辑中填入参数。

>> minPage & maxPage

若需求抓取页面不只一页，则将minPage置为起始页，maxPage置为终止页，同时定制扩展功能，利用参数化的url在抓取逻辑中进行处理。

>> regionId

抓取源应指定目的地，通常以乐途目的地字典中的ID值配置regionId即可，若有特殊情况（如混合目的地抓取），应在抓取逻辑中提取指定。

#### 确定正则表达式

>> reAbstract

若文章摘要为页面抓取模式，应分析页面源码，抽象出提取文章摘要信息的正则表达式，其中文章链接必须提取，文章标题与发布时间则是能提取尽量提取，将正则表达式进行html转义处理后置入reAbstract（参考在线转义工具如http://www.cnblogs.com/relucent/p/3314831.html）；

若文章摘要为异步加载模式，reAbstract可按照抓取逻辑定制填写，正则表达式不必须，可空置或置为其他数据，如参数化文章正文API。

>> reArticle

若文章正文为页面抓取模式，应分析页面源码，抽象出提取文章正文信息的正则表达式，其中文章正文必须提取，文章标题与发布时间若在摘要中未提取也必须提取（无法提取时，应在抓取逻辑中另行处理）

若文章正文为异步加载模式，reArticle可按照抓取逻辑定制填写，正则表达式不必须，可空置或置为其他数据，如参数化图片API。

>> reImage

若图片隐藏在文章正文中，应分析正文代码，抽象出提取图片的正则表达式，图片链接必须提取。

若图片信息独立于正文，reImage可按照抓取逻辑定制填写，正则表达式不必须，可空置或置为其他数据，如防盗链破解参数。

#### 确定爬虫模板

>> module

不同规则的爬虫抓取逻辑可能各不相同，也可能存在复用，因此可在规则中指定爬虫模板，若无可复用模板，也可创建新爬虫模板，然后将module置为模板名称；

模板名称格式通常为XSpider，不同模板名称不可重复，模板定义可参考模板扩展示例。

#### 确定任务计划

>> isValid

爬虫规则可自由预设，但只有将isValid被置为1的规则，在爬虫任务计划扫描时进入任务队列，isValid置为0时表示停用，置为-1时表示禁用，其他取值可在功能扩展中确定。

>> spiderName

爬虫抓取文章后会发布正文页，默认会以匿名的乐途小编身份发表，此时spiderName置空，若要指定发布人，应将spiderName置为特定编辑者的CMS用户名。

>> wakeTime

爬虫任务若要执行，至少还需要一个执行时间，爬虫将在指定时间之后被唤醒，执行任务，此时应在wakeTime中添加timePotX子节点，X用于区别多个不同时间点（如0、1、2），意味着统一爬虫可在一日之内的不同时间启动多次。

### 模板扩展

模板扩展需添加XSpider.py文件，在其中实现继承自Spider或其他模板Spider的XSpider类，其中CatchArticles必须重载以实现定制化的抓取逻辑，以福州福州旅游网爬虫为例：

class FuZhouSpider(Spider.Spider):

"""福州旅游网 Spider"""

def \_\_init\_\_(self):

Spider.Spider.\_\_init\_\_(self)

def CatchArticles(self):

recAbstract = re.compile(self.reAbstract, re.DOTALL)

recArticle = re.compile(self.reArticle, re.DOTALL)

recImage = re.compile(self.reImage, re.DOTALL)

html = self.DownLoadHtml(self.url, '文章列表页{0}访问失败，异常信息为:{1}')

if html == None:

return self.articles

for x in recAbstract.findall(html):

article = dict(

time = datetime.datetime.strptime(x[0],'%Y-%m-%d'),

# url = self.url[0:self.url.rfind('/')] + x[1][1:],

url = Spider.ComposeUrl(self.url, x[1]),

title = x[2]

)

html = self.DownLoadHtml(article['url'], '文章页{0}访问失败，异常信息为:{1}')

if html == None:

continue

content = None

images = []

imageCount = 0

for y in recArticle.findall(html):

content = y

for z in recImage.findall(content):

imageCount += 1

# imageUrl = article['url'][0:article['url'].rfind('/')] + z[1:]

imageUrl = Spider.ComposeUrl(article['url'], z)

image = self.DownLoadImage(imageUrl, '图片{0}提取失败，异常信息为:{1}')

if image == None:

continue

images.append(image)

if not content \

or imageCount != len(images):

continue

self.CacheArticle(article, content, images, '成功自{0}提取文章')

return self.articles

由于采用继承机制，\_\_init\_\_方法中应调用父级\_\_init\_\_方法，CatchArticles方法则可利用基类模板中的方法、字段及函数，结合自有方法、字段及函数扩展抓取逻辑，扩展流程如下：

#### 处理继承要素

>>选定基类

通常使用Spider.Spider，其中前一个Spider为模块名，对应Spider.py，后一个Spider为类名，如需更细粒度的模板复用，如定制页面抓取模板或异步抓取模板，可继承对应二级模板。

>>选定重载方法

继承模板通常会复用基类模板的大部分方法，但作为区别，必定有自身的特定逻辑，比如重载CatchArticles方法用以实现不同爬虫抓取逻辑（该方法在Spider中为虚方法，继承Spider后必须实现）；

除CatchArticles方法外，其他基类方法也存在扩展空间，可参考3.1.5.1中的方法说明进行重载。

>>添加自有元素

通常情况下，基类元素已经足够使用，但在一些特殊场景，可能需要添加一些新元素，比如文章过滤函数，图片抗反盗链下载之类的功能型函数，然后在重载的方法中使用。

#### 组织抓取逻辑

>>抓取源加载

根据爬虫规则中定义的url或者Api，从抓取源下载数据；

页面抓取时，需指定指定抓取页面的编码格式，默认为utf-8编码，但部分页面会使用gbk或者gb2312编码；

异步抓取或多页抓取时，API通常需要一些需要二次指定的参数，此时应将参数赋值再发起请求；

抓取源通依照抓取模式存在区别，页面抓取源通常由三段式的文章列表页、文章正文页与文章正文组成，异步抓取源则随API的定义不同而不同，混合抓取则结合了前两者。

基类抓取相关方法包括DownLoadHtml与DownLoadImage，DownLoadHtml用于加载Html源码或者Api响应数据，DownLoadImage则用于下载图片等文件至缓存路径。

>>数据解析与提取

抓取数据最终需转化为文章基本信息，用于后期处理，其中必要组成部分包含文章摘要信息（文章源链接、标题及发布日期等）、文章正文以及图片清单（图片链接及缓存路径等）；

页面抓取数据需要使用正则表达式解析，对应正则表达式由基类从爬虫规则读取，由reAbstract（文章摘要规则）、reArticle（文章正文规则）以及reImage（图片规则）组成，分别可提取信息至article（文章基本信息）、content（文章正文）以及images（图片列表）；

异步抓取通常不需要使用正则表达式，因为抓取数据通常为结构化数据，可以直接反序列化为友好的数据结构，如Json数据，可先进行清洗（将null替换为Python中的None），然后直接使用eval函数转化为数据字典；

reAbstract、reArticle及reImage在异步抓取或混合抓取中的存储涵义开放，可配合模板自行定义，如API，筛选规则等，只需要保证最终可正确提取文章基本信息；

提取文章发布时间时，应注意时间数据的表示形式，指定转化格式（如%Y-%m-%d可匹配2014-10-1），并注意从摘要数据和从正文数据提取的区别；

正则解析函数主要使用findall进行全文搜索，注意该函数匹配结果为多项时将返回匹配结果迭代器，为单项时将直接返回匹配结果；

信息提取完成后，应调用基类的CacheArticle方法，将article、content与images组装并缓存，等待后起批量处理。

#### 文章后期处理

通常情况下图像后期处理对模板扩展透明，但其中的部分环节可进行独立定制，文章后期处理的流程为：

有效性检查—图片转存—**文章正文图片链接替换**—**正文清洗**—文章入库—文章图片压图—发布。

其中模板扩展可参与环节主要为文章正文图片链接替换（ReplaceArticleImages）与正文清洗（ClearArticles）环节。

ReplaceArticleImages扩展示例如更改img标签的替换形式，如加入alt等属性。

ClearArticles扩展示例如增加过滤规则，如JS过滤。

### 功能扩展

功能扩展自由度较大，但通常情况下是对爬虫服务任务流程进行重组。

爬虫服务主流程为：

扫描爬虫任务—执行爬虫任务—休眠，其功能扩展主要集中在任务管理层与规则读取层。

特定规则的执行流程为：

**加载任务**—文章抓取—**有效性检查**—**图片转存**—文章正文图片链接替换—正文清洗—**文章入库**—**文章图片压图**—**发布**，其功能扩展主要集中在文章转存层、数据模型层以及消息模型层。

功能扩展的方式主要有两种，加入中间层以添加中间步骤以及重载替换特定步骤，对应的可扩展空间请参阅架构说明部分。