AyugeSpiderTools

Release 1.1.1

shengchenyang

Mar 03, 2023

第一步

在此之前，我们需要了解 Scrapy 是一个快速的高级 网络爬虫 和 网页抓取 的框架，用于抓取网站并从其网页中提取结构化数据。它可以用于广泛的目的，从数据挖掘到监控和自动化测试。

AyugeSpiderTools 是充分发挥 Scrapy 的模板功能的一个工具库，可以很方便地管理 Scrapy 项目，比如可以使得我们方便地生成 Scrapy 项目结构，当使用本库内置工具时可以不用每次手动创建 items，middlewares，pipelines，settings 等，内置了比较通用和常见的 middlewares 和 pipelines。但如果你常用的功能不在此库中，你可以自行添加修改后 build 成为你专属的工具库。

CHAPTER

ONE

GETTING HELP

遇到麻烦？请优先尝试使用以下方式提问！

- 请在本库 ayugespidertools github 上提 `issues`。
- 除非一些功能性 `bug`，其它的功能依赖于 `scrapy`，你或许能在 scrapy issues 或社区中找到答案。
- 若有其它问题也可尝试 邮箱 联系。

CHAPTER

TWO

第一步

2.1 AyugeSpiderTools 一目了然

AyugeSpiderTools 是一个对 Scrapy 扩展模块，对其 spider，item，middleware，pipeline 等模块中的常用功能进行模板化生成和配置。比如使用生成常见的 spider ，运行 sh 和 settings 配置等脚本和固定项目文件结构，也对其不同模块功能扩展，比如给 spider 挂上 Mysql engine 的单例句柄用于 yield 前的去重逻辑，给 pipeline 添加自动生成 Mysql 存储场景下所依赖的数据库、数据表、数据字段及注释，也可以解决常见的（字段编码，Data too long，存储字段不存在等等）错误场景。还有很多功能，请在其 Github 上查看。

AyugeSpiderTools 相关信息：

```
1. 具体请查看对应链接：[AyugeSpiderTools](https://github.com/shengchenyang/
↪AyugeSpiderTools)
```

2.1.1 注意：

如果你觉得某些功能实现未达到你的期望，比如某些中间件或管道等的实现方法你有更好的方式，你完全可以自行修改和 build，让其成为你个人或小组中的专属库。你可以修改任何你觉得有必要的部分，包括库名在内，希望本库能给你在爬虫开发或 scrapy 扩展开发方面有所指引。

2.1.2 示例蜘蛛的演练

为了向您展示 ayugespidertools 带来了什么，我们将带您通过一个 Scrapy Spider 示例，使用最简单的方式来运行蜘蛛。

先创建项目：

```
ayugespidertools startproject <project_name>

eg: 本示例使用的 project_name 为 DemoSpider
```

创建爬虫脚本：

```
进入项目根目录
cd <project_name>

生成脚本
ayugespidertools genspider <spider_name> <example.com>
```

下面是从网站 https://blog.csdn.net/phoenix/web/blog/hot-rank?page=0&pageSize=25&type= 抓取热榜信息的蜘蛛代码：

```
import json
from scrapy.http import Request
from ayugespidertools.Items import MysqlDataItem
from DemoSpider.common.DataEnum import TableEnum
from ayugespidertools.AyugeSpider import AyuSpider
from ayugespidertools.common.Utils import ToolsForAyu

"""
################################################################################################
↪###############
# collection_website: CSDN - 专业开发者社区
# collection_content: 热榜文章排名 Demo 采集示例 - 存入 Mysql (配置根据本地 settings 的␣
↪LOCAL_MYSQL_CONFIG 中取值)
# create_time: 2022-07-30
# explain:
# demand_code_prefix = ''
################################################################################################
↪###############
"""


class DemoOneSpider(AyuSpider):
    name = 'demo_one'
    allowed_domains = ['blog.csdn.net']
    start_urls = ['https://blog.csdn.net/']

    # 数据库表的枚举信息，当前项目所依赖的表信息，一般用于存储数据时使用
    custom_table_enum = TableEnum
    # 初始化配置的类型，初始化设置
    settings_type = 'debug'
    custom_settings = {
        # 日志等级，此参数值与 ayugespidertools 库中的 loguru 日志等级一致
        'LOG_LEVEL': 'DEBUG',
        # 是否开启记录项目相关运行统计信息，其实默认就是关闭，为了展示此参数
```

(continues on next page)

(continued from previous page)

```
        'RECORD_LOG_TO_MYSQL': False,
        # Mysql 数据表的前缀名称，用于标记属于哪个项目（也可不配置此参数，按需配置）
        'MYSQL_TABLE_PREFIX': "demo1_",
        'ITEM_PIPELINES': {
            # 激活此项则数据会存储至 Mysql，其 Mysql 链接配置在 VIT 目录下的 .conf 中查看
            'ayugespidertools.Pipelines.AyuFtyMysqlPipeline': 300,
        },
        'DOWNLOADER_MIDDLEWARES': {
            # 随机请求头，请求头优先级与 `fake_useragent` 中的一致
            'ayugespidertools.Middlewares.RandomRequestUaMiddleware': 400,
        },
    }

    # 打开 mysql 引擎开关，用于数据入库前更新逻辑判断
    mysql_engine_enabled = True

    def start_requests(self):
        """
        获取项目热榜的列表数据
        """
        yield Request(
            url="https://blog.csdn.net/phoenix/web/blog/hot-rank?page=0&pageSize=25&
↪type=",
            callback=self.parse_first,
            headers={
                'referer': 'https://blog.csdn.net/rank/list',
            },
            dont_filter=True
        )

    def parse_first(self, response):
        data_list = json.loads(response.text)['data']
        for curr_data in data_list:
            # 这里的所有解析规则可选择自己习惯的
            article_detail_url = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="articleDetailUrl")

            article_title = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="articleTitle")

            comment_count = ToolsForAyu.extract_with_json(
```

(continues on next page)

(continued from previous page)

```
                json_data=curr_data,
                query="commentCount")

            favor_count = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="favorCount")

            nick_name = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="nickName")

            # 数据存储方式 1，非常推荐此写法。article_info 含有所有需要存储至表中的字段
            article_info = {
                "article_detail_url": {'key_value': article_detail_url, 'notes': '文章
详情链接'},
                "article_title": {'key_value': article_title, 'notes': '文章标题'},
                "comment_count": {'key_value': comment_count, 'notes': '文章评论数量'},
                "favor_count": {'key_value': favor_count, 'notes': '文章赞成数量'},
                "nick_name": {'key_value': nick_name, 'notes': '文章作者昵称'}
            }

            """
            # 当然这么写也可以，但是不推荐
            article_info = {
                "article_detail_url": article_detail_url,
                "article_title": article_title,
                "comment_count": comment_count,
                "favor_count": favor_count,
                "nick_name": nick_name,
            }
            """

            ArticleInfoItem = MysqlDataItem(
                alldata=article_info,
                table=TableEnum.aritle_list_table.value['value'],
            )

            '''
            # 旧 scrapy 脚本也可方便地改写为 ayugespidertools 支持的示例
            # 也可以不用修改，只需补充上所需要的 table 和 item_mode 字段，来指定存储表名和存储方式
            item = dict()
            item['article_detail_url'] = article_detail_url
            item['article_title'] = article_title
```

(continues on next page)

(continued from previous page)

```
            item['comment_count'] = comment_count
            item['favor_count'] = favor_count
            item['nick_name'] = nick_name

            item['table'] = 'article_info_list'
            item['item_mode'] = 'Mysql'
            yield item

            # 或者这样写
            item = {
                'article_detail_url': article_detail_url,
                'article_title': article_title,
                'comment_count': comment_count,
                'favor_count': favor_count,
                'nick_name': nick_name,

                # 兼容旧写法，但是要添加 table 字段和 item_mode 字段
                # (即需要存储的表格名称，注意：是去除表格前缀的表名，比如表为 demo_article_info_
↪list，前缀为 demo_，则 table 名为 article_info_list)
                'table': 'article_info_list',
                'item_mode': 'Mysql',
            }
            yield item
            '''

            # 数据入库逻辑 -> 测试 mysql_engine 的去重功能
            try:
                sql = '''select `id` from `{}` where `article_detail_url` = "{}"␣
↪limit 1'''.format(self.custom_settings.get('MYSQL_TABLE_PREFIX', '') + TableEnum.
↪aritle_list_table.value['value'], article_detail_url)
                df = pandas.read_sql(sql, self.mysql_engine)

                # 如果为空，说明此数据不存在于数据库，则新增
                if df.empty:
                    yield ArticleInfoItem

                # 如果已存在，1). 若需要更新，请自定义更新数据结构和更新逻辑；2). 若不用更新，则跳
过即可。
                else:
                    logger.debug(f"标题为 ” {article_title}“ 的数据已存在")

            except Exception:
                yield ArticleInfoItem
```

刚刚发生了什么？

刚刚使用 ayugespidertools 创建了项目，并生成了具体的爬虫脚本示例。其爬虫脚本中的各种依赖（比如项目目录结构，配置信息等）在创建项目后就正常产生了，一般所需的配置信息（比如 Mysql，MongoDB，OSS 等）在项目的 VIT 目录下 .conf 文件中修改，不需要配置的不用理会它即可。

只要配置好 .conf 信息，就可以跑通以上示例。如果修改为新的项目，只需要修改上面示例中的 spdider 解析规则即可。

2.1.3 还有什么？

本库依赖 Scrapy，你可以使用 Scrapy 命令来管理你的项目，体会 Scrapy 的强大和方便。

ayugespidertools 根据 scrapy 的模板功能方便的创建示例脚本，比如：

```
# 查看支持的脚本模板示例
ayugespidertools genspider -l

<output>
Available templates:
  async
  basic
  crawl
  csvfeed
  xmlfeed

# 使用具体的示例命令
ayugespidertools genspider -t <Available_templates> <spider_name> <example.com>

eg: ayugespidertools gendpier -t async demom_async baidu.com
```

2.1.4 下一步是什么？

接下来的步骤是 安装 AyugeSpiderTools，按照 Scrapy 的教程学习如何使用 Scrapy 并加入 Scrapy 社区。谢谢你的关注！

2.2 安装指南

2.2.1 支持的 Python 版本

AyugeSpiderTools 需要 Python 3.8+。

2.2.2 安装 AyugeSpiderTools

为了向您展示 ayugespidertools 带来了什么，我们将带您通过一个 Scrapy Spider 示例，使用最简单的方式来运行蜘蛛。

可以使用以下命令从 PyPI 安装 ayugespidertools 及其依赖项：

```
pip install ayugespidertools
```

我们强烈建议您将 ayugespidertools 安装在专用的 virtualenv 中，以避免与您的系统包发生冲突。

值得知道的事情

ayugespidertools 是依赖于 Scrapy 开发的，对其在爬虫开发中遇到的常用操作进行扩展。

使用虚拟环境（推荐）

建议在所有平台上的虚拟环境中安装此库。

有关如何创建虚拟环境的信息，请参阅虚拟环境和包。

2.3 AyugeSpiderTools 教程

在本教程中，我们假设您的系统上已经安装了 ayugespidertools。

我们要抓取 blog.csdn.net，这是一个知识问答社区的网站。

本教程将引导您完成这些任务：

- 创建一个新的 Scrapy 项目
- 编写爬虫来抓取站点并提取数据

- 使用命令行导出抓取的数据
- 更改蜘蛛以递归地跟踪链接
- 使用蜘蛛参数

2.3.1 创建项目

在开始抓取之前，您必须设置一个新的 ayugespidertools 项目。输入您要存储代码的目录并运行：

```
ayugespidertools startproject DemoSpider
```

这将创建一个 DemoSpider 包含以下内容的目录：

```
DemoSpider/
|-- DemoSpider                                                                  #␣
↪project's Python module, you'll import your code from here
|   |-- common                                                                  # 这里主要
存放通用方法，自定义方法模块。
|   |   |-- DataEnum.py                                                       # 数据库表枚举信息
示例
|   |-- __init__.py
|   |-- items.py                                                             # project␣
↪items definition file
|   |-- logs                                                                    # 日志管理文
件夹，可以自定义规则
|   |   |-- DemoSpider.log                                                # scrapy 输出日志，文件
名称为项目名
|   |   |-- error.log                                                      # loguru 日志␣
↪error 规则输出文件
|   |   `-- runtime.log                                                    # loguru 日志␣
↪debug 规则输出文件
|   |-- middlewares.py                                                    # project␣
↪middlewares file
|   |-- pipelines.py                                                   # project␣
↪pipelines file
|   |-- run.py                                                                  #␣
↪scrapy 运行文件
|   |-- run.sh                                                                  # 项目运
行 shell，运行以上的 run.py
|   |-- settings.py                                                          # project␣
↪settings file
|   |-- spiders                                                               # a␣
↪directory where you'll later put your spiders
|   |   |-- __init__.py
```

(continues on next page)

(continued from previous page)

```
|   `-- VIT
|       `-- 请修改.conf 中的配置信息                                    # 提示文件
|                   `-- .conf                                                    # 配
置文件，用于修改 Mysql, MongoDB 等配置
|-- .gitignore                                                                  # git␣
↪ignore 文件
|-- pyproject.toml                                                          # 项目配置
|-- README.md                                                                  # 说明文档
|-- requirements.txt                                                    # 依赖文件
`-- scrapy.cfg                                  # deploy configuration file
```

2.3.2 我们的第一个 Spider

这是我们第一个 Spider 的代码。demo_one.py 将其保存在项目目录下命名的文件 DemoSpider/spiders 中：

```
import pandas
from scrapy.http import Request
from DemoSpider.settings import logger
from scrapy.http.response.text import TextResponse
from DemoSpider.common.DataEnum import TableEnum
from ayugespidertools.AyugeSpider import AyuSpider
from ayugespidertools.common.Utils import ToolsForAyu
from ayugespidertools.Items import MysqlDataItem, MongoDataItem

"""
################################################################################
↪###############
# collection_website: CSDN - 专业开发者社区
# collection_content: 热榜文章排名 Demo 采集示例 - 同时存入 Mysql 和 MongoDB 的场景 (配置根据
本地 settings 的中取值)
# create_time: 2023-01-10
# explain:
# demand_code_prefix = ''
################################################################################
↪###############
"""


class DemoOneSpider(AyuSpider):
    name = 'demo_one'
    allowed_domains = ['blog.csdn.net']
```

(continues on next page)

(continued from previous page)

```
    start_urls = ['https://blog.csdn.net/']

    # 数据库表的枚举信息
    custom_table_enum = TableEnum
    # 初始化配置的类型
    settings_type = 'debug'
    custom_settings = {
        # scrapy 日志等级配置
        'LOG_LEVEL': 'DEBUG',
        'LOGURU_ENABLED': False,
        # 是否开启记录项目相关运行统计信息。不配置默认为 False
        'RECORD_LOG_TO_MYSQL': False,
        # 设置 ayugespidertools 库的日志输出为 loguru，可自行配置 logger 规则来管理项目日志。若
不配置此项，库日志只会在控制台上打印
        'LOGURU_CONFIG': logger,
        # Mysql 数据表的前缀名称，用于标记属于哪个项目，也可以不用配置
        'MYSQL_TABLE_PREFIX': "demo_basic_",
        # MongoDB 集合的前缀名称，用于标记属于哪个项目，也可以不用配置
        'MONGODB_COLLECTION_PREFIX': "demo_basic_",
        'ITEM_PIPELINES': {
            # 激活此项则数据会存储至 Mysql
            'ayugespidertools.Pipelines.AyuFtyMysqlPipeline': 300,
            # 激活此项则数据会存储至 MongoDB
            'ayugespidertools.Pipelines.AyuFtyMongoPipeline': 301,
        },
    }

    # 打开 mysql 引擎开关，用于数据入库前更新逻辑判断
    mysql_engine_enabled = True

    def start_requests(self):
        """
        get 请求首页，获取项目列表数据
        """
        yield Request(
            url="https://blog.csdn.net/phoenix/web/blog/hot-rank?page=0&pageSize=25&
↪type=",
            callback=self.parse_first,
            headers={
                'referer': 'https://blog.csdn.net/rank/list',
            },
            cb_kwargs=dict(
                curr_site="csdn",
```

(continues on next page)

(continued from previous page)

```
            ),
            dont_filter=True
        )

    def parse_first(self, response: TextResponse, curr_site: str):
        # 日志使用: scrapy 的 self.logger 或本库的 self.slog 或直接使用全局的 logger handle也行 (根据场景自行选择)
        self.slog.info(f"当前采集的站点为: {curr_site}")

        # 你可以自定义解析规则, 使用 lxml 还是 response.css response.xpath 等等都可以。
        data_list = ToolsForAyu.extract_with_json(json_data=response.json(), query="data")
        for curr_data in data_list:
            article_detail_url = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="articleDetailUrl")

            article_title = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="articleTitle")

            comment_count = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="commentCount")

            favor_count = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="favorCount")

            nick_name = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="nickName")

            # 这是需要存储的字段信息
            article_info = {
                "article_detail_url": {'key_value': article_detail_url, 'notes': '文章详情链接'},
                "article_title": {'key_value': article_title, 'notes': '文章标题'},
                "comment_count": {'key_value': comment_count, 'notes': '文章评论数量'},
                "favor_count": {'key_value': favor_count, 'notes': '文章赞成数量'},
                "nick_name": {'key_value': nick_name, 'notes': '文章作者昵称'}
            }
```

(continues on next page)

(continued from previous page)

```
            ArticleInfoMysqlItem = MysqlDataItem(
                alldata=article_info,
                table=TableEnum.article_list_table.value['value'],
            )

            self.slog.info(f"ArticleInfoMysqlItem: {ArticleInfoMysqlItem}")

            # 数据入库逻辑，你可以使用 mysql_engine 来去重或自定义规则
            try:
                sql = '''select `id` from `{}` where `article_detail_url` = "{}" limit 1'''.format(
                    self.custom_settings.get('MYSQL_TABLE_PREFIX', '') + TableEnum.article_list_table.value['value'],
                    article_detail_url)
                df = pandas.read_sql(sql, self.mysql_engine)

                # 如果为空，说明此数据不存在于数据库，则新增
                if df.empty:
                    yield ArticleInfoMysqlItem

                # 如果已存在，1). 若需要更新，请自定义更新数据结构和更新逻辑；2). 若不用更新，则跳过即可。
                else:
                    self.slog.debug(f"标题为 ” {article_title} “ 的数据已存在")

            except Exception as e:
                if any(["1146" in str(e), "1054" in str(e), "doesn't exist" in str(e)]):
                    yield ArticleInfoMysqlItem
                else:
                    self.slog.error(f"请查看数据库链接或网络是否通畅! Error: {e}")

            self.slog.error(f"test {nick_name}")
            # 这是 MongoDB 存储场景的示例
            AritleInfoMongoItem = MongoDataItem(
                # alldata 用于存储 mongo 的 Document 文档所需要的字段映射
                alldata=article_info,
                # table 为 mongo 的存储 Collection 集合的名称
                table=TableEnum.article_list_table.value['value'],
                # mongo_update_rule 为查询数据是否存在的规则
                mongo_update_rule={"article_detail_url": article_detail_url},
            )
            yield AritleInfoMongoItem
```

如您所见，我们的 Spider 子类化 AyugeSpider.AyuSpider 并定义了一些属性和方法：

- name: 标识蜘蛛。在一个项目中必须是唯一的，即不能为不同的 Spiders 设置相同的名字。
- start_requests(): 必须返回一个可迭代的请求（你可以返回一个请求列表或编写一个生成器函数），蜘蛛将从中开始爬行。后续请求将从这些初始请求中依次生成。
- parse_first(): 将被调用以处理为每个请求下载的响应的方法。response 参数是 TextResponse 的一个实例，它保存页面内容，并有进一步的有用方法来处理它。该 parse_first() 方法通常解析响应，将抓取的数据提取为字典，并找到要遵循的新 URL 并从中创建新请求 (Request)。

如何运行我们的蜘蛛

要让我们的蜘蛛工作，请转到项目的顶级目录并运行：

```
# 本身就是 scrapy 的项目，所以可以使用 scrapy 可以执行的任何形式即可
scrapy crawl demo_one

# 或者执行项目根目录下的 run.py(需要编辑自己需要执行的脚本)
python3 run.py

# 或者执行项目根目录下的 run.sh，其实它也是通过调用 run.py 来执行的。只不过 shell 文件中包含了虚拟环境的 activate 了而已
sh run.sh
```

2.4 例子

本教程将引导您完成这些任务：

- 快速熟悉 ayugespidertools 库的使用方法和支持场景
- 编写爬虫来抓取站点并提取数据

2.4.1 快速开始

你可以使用以下两种方式来快速开始

方式一：ayugespidertools

通过跑通本库 Github 中的 GIF 示例

具体请点击跳转至 AyugeSpiderTools 查看

方式二：DemoSpider

通过另一个的演示项目 DemoSpider 来选择复现某些场景

最好的学习方法是通过 Github 上的 DemoSpider 示例，您可以使用它快速复现某些场景下的功能。

本库 ayugespidertools 的 github README.md 中所有功能，都可以在 DemoSpider 中找到示例。

DemoSpider 项目位于：https://github.com/shengchenyang/DemoSpider ，您可以在项目的自述文件中找到有关它的更多信息。

***AyugeSpiderTools* 一目了然**

了解什么是 AyugeSpiderTools 以及它如何为您提供帮助。

安装指南

在您的设备上安装 AyugeSpiderTools。

***AyugeSpiderTools* 教程**

编写您的第一个 AyugeSpiderTools 项目。

例子

通过一个简单示例来了解一些信息。

CHAPTER

THREE

基本概念

3.1 命令行工具

AyugeSpiderTools 是直接使用 scrapy 命令行工具来控制的，这里简称为 AyugeSpiderTools 工具，以区别于我们简称为“命令”或“Scrapy 命令”的子命令。

AyugeSpiderTools 工具只提供了常用的几个命令，用于多种用途，每个命令都接受一组不同的参数和选项。但是，其它缺失的命令你可以直接使用 Scrapy 的即可。

3.1.1 配置设置

未改变，也是在在标准位置的 ini 样式文件中查找配置参数 scrapy.cfg：

这些文件中的设置按列出的优先顺序合并：用户定义的值比系统范围的默认值具有更高的优先级，并且项目范围的设置将在定义时覆盖所有其他设置。

3.1.2 AyugeSpiderTools 项目的默认结构

在深入研究命令行工具及其子命令之前，让我们先了解一下项目的目录结构。

虽然可以修改，但是所有的项目默认都有相同的文件结构，类似这样：

```
|-- DemoSpider
|   |-- common
|   |   `-- DataEnum.py
|   |-- __init__.py
|   |-- items.py
|   |-- logs
|   |   |-- DemoSpider.log
|   |   `-- error.log
|   |-- middlewares.py
|   |-- pipelines.py
|   |-- run.py
```

(continues on next page)

(continued from previous page)

```
|   |-- run.sh
|   |-- settings.py
|   |-- spiders
|   |   |-- __init__.py
|   |   `-- spider1.py
|   `-- VIT
|       `-- 请修改.conf 中的配置信息
|-- .gitignore
|-- pyproject.toml
|-- README.md
|-- requirements.txt
`-- scrapy.cfg
```

文件所在的目录 `scrapy.cfg` 称为项目根目录。该文件包含定义项目设置的 `python` 模块的名称。这是一个例子:

```
[settings]
default = DemoSpider.settings
```

3.1.3 使用 AyugespiderTools 工具

您可以先运行不带参数的 `AyugeSpiderTools` 工具，它会打印一些使用帮助和可用的命令:

```
AyugeSpiderTools None - no active project

Usage:
  ayugespidertools <command> [options] [args]

Available commands:
  genspider     Generate new spider using pre-defined templates
  startproject  Create new project
  version       Print AyugeSpiderTools version

  [ more ]      More commands available when run from project directory

Use "ayugespidertools <command> -h" to see more info about a command
```

创建项目

您通常使用该工具做的第一件事是创建您的项目：

```
ayugespidertools startproject myproject [project_dir]
```

这将在该目录下创建一个 Scrapy 项目 project_dir。如果 project_dir 未指定，则项目目录将与 myproject 相同。

接下来，进入新项目目录：

```
cd project_dir
```

您已准备好使用该命令从那里管理和控制您的项目。

控制项目

您可以使用项目内部的工具来控制和管理它们。

例如，要创建一个新的蜘蛛：

```
ayugespidertools genspider mydomain mydomain.com
```

3.1.4 启动项目

- 句法：ayugespidertools startproject <project_name> [project_dir]
- 需要项目：无

在 project dir 目录下创建一个名为 project_name 的新项目。如果未指定项目目录，则项目目录将与项目名称相同。

使用示例：

```
ayugespidertools startproject myproject
```

3.1.5 可用的工具命令

本节包含可用内置命令的列表以及说明和一些用法示例。请记住，您始终可以通过运行以下命令获取有关每个命令的更多信息：

```
ayugespidertools <command> -h
```

您可以使用以下命令查看所有可用命令：

```
ayugespidertools -h
```

启动项目

- 句法：`ayugespidertools startproject <project_name> [project_dir]`
- 需要项目：无

在 `project dir` 目录下创建一个名为 `project name` 的新项目。如果未指定项目目录，则项目目录将与项目名称相同。

使用示例：

```
ayugespidertools startproject myproject
```

genspider

- 句法：`ayugespidertools genspider [-t template] <name> <domain or URL>`
- 需要项目：无

使用示例：

```
$ ayugespidertools genspider -l
Available templates:
  async
  basic
  crawl
  csvfeed
  xmlfeed

$ ayugespidertools genspider example example.com
Created spider 'example' using template 'basic'

$ ayugespidertools genspider -t crawl scrapyorg scrapy.org
Created spider 'scrapyorg' using template 'crawl'
```

3.2 Item

演示在使用本库场景下 Item 的使用方法。

本教程将引导您完成这些任务：

- 演示本库推荐的 Item 适配方式
- 补充适配 add_value, add_xpath, add_css 等方法的示例

3.2.1 实现原理

以下为本库中推荐的 mysql 和 MongoDB 存储时的主要 Item 示例：

其实，本库就是推荐把所有字段统一存入 alldata 字段中，其它字段用于场景补充，比如：table 字段用于说明要存储的表名/集合名，item_mode 字段用于说明存储的方式，mongo_update_rule 字段是 item_mode 为 MongoDB 存储场景时的去重条件 (可不设置此字段)。

```
@dataclass
class BaseItem:
    """
    用于构建 scrapy item 的基本结构, 所有需要存储的表对应的结构都放在 alldata 中
    """
    alldata: dict = field(default=None)
    table: str = field(default=None)


@dataclass
class MysqlDataItem(BaseItem):
    """
    这个是 Scrapy item 的 Mysql 的存储结构
    """
    item_mode: str = field(default="Mysql")


@dataclass
class MongoDataItem(BaseItem):
    """
    这个是 Scrapy item 的 MongoDB 的存储结构
    这个 mongo_update_rule 字段是用于 Mongo 存储时作查询使用
    """
    item_mode: str = field(default="MongoDB")
    mongo_update_rule: dict = field(default=None)
```

3.2.2 使用示例

只需要在 `yield item` 时，按需提前导入 `MysqlDataItem`，`MongoDataItem`，将所有的存储字段和场景补充字段全部添加完整即可。

以本库模板中的 `basic.tmpl` 为例：

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import pandas
from scrapy.http import Request
from DemoSpider.settings import logger
from scrapy.http.response.text import TextResponse
from DemoSpider.common.DataEnum import TableEnum
from ayugespidertools.AyugeSpider import AyuSpider
from ayugespidertools.common.Utils import ToolsForAyu
from ayugespidertools.Items import MysqlDataItem, MongoDataItem

"""
################################################################################################
↪###############
# collection_website: csdn.net - 采集的目标站点介绍
# collection_content: 采集内容介绍
# create_time: xxxx-xx-xx
# explain:
# demand_code_prefix = ''
################################################################################################
↪###############
"""


class DemoOneSpider(AyuSpider):
    name = 'demo_one'
    allowed_domains = ['csdn.net']
    start_urls = ['https://www.csdn.net/']

    # 数据库表的枚举信息
    custom_table_enum = TableEnum
    # 初始化配置的类型
    settings_type = 'debug'
    custom_settings = {
        # scrapy 日志等级配置
        'LOG_LEVEL': 'DEBUG',
        # 以 loguru 来管理日志，本库会在 settings 中生成规则示例，可自行修改。也可不配置
        'LOGURU_CONFIG': logger,
```

(continues on next page)

(continued from previous page)

```
        # Mysql 数据表的前缀名称，用于标记属于哪个项目，可不配置
        'MYSQL_TABLE_PREFIX': "demo_basic_",
        # MongoDB 集合的前缀名称，用于标记属于哪个项目，可不配置
        'MONGODB_COLLECTION_PREFIX': "demo_basic_",
        'ITEM_PIPELINES': {
            # 激活此项则数据会存储至 Mysql
            'ayugespidertools.Pipelines.AyuFtyMysqlPipeline': 300,
            # 激活此项则数据会存储至 MongoDB
            'ayugespidertools.Pipelines.AyuFtyMongoPipeline': 301,
        },
        'DOWNLOADER_MIDDLEWARES': {
            # 随机请求头
            'ayugespidertools.Middlewares.RandomRequestUaMiddleware': 400,
        },
    }

    # 打开 mysql 引擎开关，用于数据入库前更新逻辑判断
    mysql_engine_enabled = True

    def start_requests(self):
        """
        get 请求首页，获取项目列表数据
        """
        yield Request(
            url="https://blog.csdn.net/phoenix/web/blog/hot-rank?page=0&pageSize=25&
↪type=",
            callback=self.parse_first,
            headers={
                'referer': 'https://blog.csdn.net/rank/list',
            },
            cb_kwargs={
                "curr_site": "csdn",
            },
            dont_filter=True
        )

    def parse_first(self, response: TextResponse, curr_site: str):
        # 日志使用：scrapy 的 self.logger 或本库的 self.slog 或直接使用全局的 logger handle␣
↪也行（根据场景自行选择）
        self.slog.info(f"当前采集的站点为：{curr_site}")

        # 你可以自定义解析规则，使用 lxml 还是 response.css response.xpath 等等都可以。
        data_list = ToolsForAyu.extract_with_json(json_data=response.json(), query=
```

(continues on next page)

(continued from previous page)

```
↪"data")
        for curr_data in data_list:
            article_detail_url = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="articleDetailUrl")

            article_title = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="articleTitle")

            comment_count = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="commentCount")

            favor_count = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="favorCount")

            nick_name = ToolsForAyu.extract_with_json(
                json_data=curr_data,
                query="nickName")

            # 这是需要存储的字段信息
            article_info = {
                "article_detail_url": {'key_value': article_detail_url, 'notes': '文章
详情链接'},
                "article_title": {'key_value': article_title, 'notes': '文章标题'},
                "comment_count": {'key_value': comment_count, 'notes': '文章评论数量'},
                "favor_count": {'key_value': favor_count, 'notes': '文章赞成数量'},
                "nick_name": {'key_value': nick_name, 'notes': '文章作者昵称'}
            }

            ArticleInfoMysqlItem = MysqlDataItem(
                alldata=article_info,
                table=TableEnum.article_list_table.value['value'],
            )
            self.slog.info(f"ArticleInfoMysqlItem: {ArticleInfoMysqlItem}")

            # 数据入库逻辑，你可以使用 mysql_engine 来去重或自定义规则
            try:
                sql = '''select `id` from `{}` where `article_detail_url` = "{}"␣
↪limit 1'''.format(
                    self.custom_settings.get('MYSQL_TABLE_PREFIX', '') + TableEnum.
```

(continues on next page)

(continued from previous page)

```
↪article_list_table.value['value'],
                    article_detail_url)
                df = pandas.read_sql(sql, self.mysql_engine)

                # 如果为空，说明此数据不存在于数据库，则新增
                if df.empty:
                    yield ArticleInfoMysqlItem

                # 如果已存在，1). 若需要更新，请自定义更新数据结构和更新逻辑；2). 若不用更新，则跳
过即可。
                else:
                    self.slog.debug(f"标题为 ” {article_title}“ 的数据已存在")

            except Exception as e:
                if any(["1146" in str(e), "1054" in str(e), "doesn't exist" in␣
↪str(e)]):
                    yield ArticleInfoMysqlItem
                else:
                    self.slog.error(f"请查看数据库链接或网络是否通畅! Error: {e}")

            # 这是 MongoDB 存储场景的示例
            ArticleInfoMongoItem = MongoDataItem(
                # alldata 用于存储 mongo 的 Document 文档所需要的字段映射
                alldata=article_info,
                # table 为 mongo 的存储 Collection 集合的名称
                table=TableEnum.article_list_table.value['value'],
                # mongo_update_rule 为查询数据是否存在的规则
                mongo_update_rule={"article_detail_url": article_detail_url},
            )
            yield ArticleInfoMongoItem
```

由上可知，本库中的 Item 使用方法还是很方便的。

对以上 Item 相关信息解释：

- 先导入所需 Item
  - mysql 场景导入 MysqlDataItem
  - mongo 场景导入 MongoDataItem
- 构建对应场景的 Item
  - MysqlDataItem 构建 Mysql 存储场景
  - MongoDataItem 构建 MongoDB 存储场景
- 最后 yield 对应 item 即可

3.2.3 yield item

这里解释下 item 的格式问题，虽说也是支持直接 yield dict ， scrapy 的 item 格式 (即 ScrapyClassicItem)，还有就是本库推荐的 MysqlDataItem 和 MongoDataItem 的形式。

这里介绍下 item 字段及其注释，以上所有 item 都有参数提示：

注，对以上表格中内容进行扩充解释：

- alldata 字段格式：

```
# alldata 示例一，推荐此代码编写风格
alldata1 = {
    "article_detail_url": {
        "key_value": article_detail_url,
        "notes": "文章详情链接",
    },
    "article_title": {
        "key_value": article_title,
        "notes": "文章标题",
    },
}
```

```
# alldata 示例二
alldata2 = {
    "article_detail_url": article_detail_url,
    "article_title": article_title,
}
```

3.2.4 自定义 Item 字段和实现 Item Loaders

本库支持 scrapy Item 的格式，或者 dict 格式，DemoSpider 中 demo_one 已有示例。

由于本库推荐将所有存储字段统一存储至 alldata 中来维护的，而且本库 Item 已使用了 scrapy 推荐的 dataclass 实现，自然是不推荐自定义 Item 字段的，但是也是可以实现的。具体请在下一章浏览。

3.3 Item Loaders

Item Loaders 提供了一种方便的机制来填充已抓取的 ITEM。尽管项目可以直接填充，项目加载器提供了一个更方便的 API 来从抓取过程中填充它们，通过自动化一些常见的任务，比如在分配它之前解析原始提取的数据。

换句话说，ITEM 提供了抓取数据的 * 容器，而项目加载器提供了**填充**该容器的机制。

Item Loaders 旨在提供一种灵活、高效和简单的机制来扩展和覆盖不同的字段解析规则，无论是通过蜘蛛，还是通过源格式（HTML、XML 等），而不会成为维护的噩梦。

3.3.1 使用方法

具体请查看 scrapy 中对应的 Item Loaders 的文档。

由文档可知，如果使用 Item Loaders 需要先声明 Item 子类，并固定 Field 字段。即以下内容：

```
import scrapy

class Product(scrapy.Item):
    name = scrapy.Field()
    price = scrapy.Field()
    stock = scrapy.Field()
    tags = scrapy.Field()
    last_updated = scrapy.Field(serializer=str)
```

但是本库不推荐每次都自定义重写 Item 的 Field 字段，这样丧失了解放双手的目的。由上一章可知，本库只提供公共常用且固定的字段。

那本库如何实现使用项目加载器填充项目的效果呢，本库是通过 ItemLoader 配合 make_dataclass 的方法，不过这也丢失了优雅便捷性。

3.3.2 使用项目加载器填充项目

这是 Spider 中典型的 Item Loader 用法：

```
from scrapy.loader import ItemLoader
from myproject.items import Product

def parse(self, response):
    l = ItemLoader(item=Product(), response=response)
    l.add_xpath('name', '//div[@class="product_name"]')
    l.add_xpath('name', '//div[@class="product_title"]')
    l.add_xpath('price', '//p[@id="price"]')
    l.add_css('stock', 'p#stock')
    l.add_value('last_updated', 'today') # you can also use literal values
    return l.load_item()
```

3.3.3 使用 dataclass items

本库中 Item 中的除了 ScrapyClassicItem，其它均为此类。

3.3.4 输入和输出处理器

那么，在本库中的使用方法如下：

Item Loader 包含一个输入处理器和一个用于每个 item 字段的输出处理器。输入处理器在收到数据后立即处理提取的数据（通过add_xpath(),add_css()或 add_value()方法），输入处理器的结果被收集并保存在 ItemLoader 中。收集完所有数据后，ItemLoader.load_item()调用该方法填充并获取填充的项对象。那是使用先前收集的数据（并使用输入处理器处理）调用输出处理器的时候。输出处理器的结果是分配给项目的最终值。

让我们看一个示例来说明如何为特定字段调用输入和输出处理器（这同样适用于任何其他字段）：

ScrapyClassicItem

本库是为了用手动管理 item 模块，比如需存储的字段数量，字段类型等；其需要存储的字段全部放在 all-data 的 item 字段中，可以自定义扩展，但是无法使用 Item Loaders 的 add_value 等特性若要支持 Item Loaders 的特性，需要自己补充完整 item 字段，比如下面代码：

```
# 先补充需要管理的 item 字段
ScrapyClassicItem.fields["add_key1"] = scrapy.Field()
ScrapyClassicItem.fields["add_key2"] = scrapy.Field()
```

然后就可以使用 [使用项目加载器填充项目](# 使用项目加载器填充项目) 中的代码了

make_dataclass

本库中的 MysqlDataItem 和 MongoDataItem 已经使用了 @dataclass 的装饰器了，不再很优雅和方便地对其扩展字段了，推荐使用 make_dataclass 自行设置

```
MineItem = make_dataclass(
    "MineItem",
    [
        ("book_name", str, field(default=None)),
        ("book_intro", str, field(default=None)),
        ("item_mode", str, field(default="Mysql")),
        ("table", str, field(default="save_table_name")),
    ],
)
```

(continues on next page)

(continued from previous page)

```
mine_item = ItemLoader(item=MineItem(), selector=None)
mine_item.default_output_processor = TakeFirst()
mine_item.add_value("book_name", "book_name_data")
# mine_item.add_xpath("book_intro", "get_book_intro_xpath")
item = mine_item.load_item()
print("item:", item)
```

以上，可以发现 scrapy 也是推荐固定 Item 字段的，需要什么类型的字段就提前创建好其字段。

本库中 Item 则是直接将存储字段全存到 alldata 中即可。本库主推便捷，不太推荐使用以上代码自定义增加 Item 字段来适配 Item Loaders 的特性，除非某些场景下使用 Item Loaders 能够极大方便开发时，才推荐使用下。

命令行工具

了解用于管理 Scrapy 项目的命令行工具。

Item

定义要采集的数据。

Item Loaders

用提取的数据填充你的 item。

CHAPTER

FOUR

内置服务

4.1 Logging

Scrapy 用 logging 来记录日志，日志记录开箱即用，并且可以在某种程度上使用日志设置中列出的 Scrapy 设置进行配置。

本文不再介绍其详细配置及用法，请移步其官网文档中查看。AyugeSpiderTools 库会在 settings 中默认设置一个日志存储配置，默认放在项目的 logs 文件夹下，其名称为项目名称，如下所示：

```
# 日志管理
LOG_FILE = f"{LOG_DIR}/DemoSpider.log"

# 配置中 DemoSpider 是与 ayugespidertools startproject <project_name> 中的项目名称对应的
```

4.1.1 slog 日志

ayugespidertools 添加了 loguru 库来管理日志，可以很方便的查看不同日志等级的信息。可以在 spider 脚本中使用 spider.slog 或 self.slog 即可记录日志。同样，本库会在 settings 中也默认设置一个 loguru 的日志存储配置，也放在项目的 logs 文件夹下，如下所示：

```
# 本库只会持久化记录 error 级别的日志，但在调试时也可以方便地查看（包括其它等级的）控制台日志
logger.add(
    env.str("LOG_ERROR_FILE", f"{LOG_DIR}/error.log"),
    level="ERROR",
    rotation="1 week",
    retention="7 days",
    enqueue=True,
)
```

4.1.2 日志级别

ayugespidertools 中的 loguru 日志等级与 Python 的内置日志记录定义一致，大致分为 5 个不同的级别来指示给定日志消息的严重性。以下是标准的，按降序排列：

1. slog.CRITICAL- 对于严重错误（最高严重性）
2. slog.ERROR- 对于常规错误
3. slog.WARNING- 警告信息
4. slog.INFO- 用于信息性消息
5. slog.DEBUG- 用于调试消息（最低严重性）

4.1.3 如何记录消息

至于如何使用 scrapy logging 来记录的示例就不再展示了，具体使用方法请看文档：scrapy logging 使用说明，本库更推荐在调试阶段使用 loguru 来打印日志，会更快捷和明显地查看自己注意的部分。

ayugespidertools 会在 startproject 后默认再 settings 中添加一个日志配置，用于当前项目全局使用，可以在项目的各个目录文件中使用。

以下是如何使用 loguru 的 WARNING 级别记录消息的快速示例：

```
# project_name 为当前所在的 scrapy 项目名称
from <project_name>.settings import logger
logger.warning("This is a warning")
```

因为 Loguru 的理念是：**只有一个 logger**。将日志消息分派给已配置处理程序的对象。Logger 是 loguru 的核心对象，每个日志配置和使用都要通过对其中一个方法的调用。只有一个记录器，因此在使用之前不需要检索一个记录器。具体请查看文档：loguru 使用说明

所以，你也可以直接使用如下方式，也会在全局中使用同一个 loguru。

```
from loguru import logger

logger.info("this is a info log")
```

4.1.4 从 spider 记录

```
import ayugespidertools


class MySpider(ayugespidertools.AyuSpider):

```

(continues on next page)

(continued from previous page)

```
    name = 'myspider'
    start_urls = ['https://scrapy.org']

    def parse(self, response):
        # 此条 (error 级别以下的) 日志默认下只会在控制台输出
        self.slog.info(f"info: Parse function called on {response.url}")
        # 此条日志在默认下会持久化存储至 error.log 中
        self.slog.error(f"error: Parse function called on {response.url}")
```

Logging

在 ayugespidertools 上学习如何使用日志。

CHAPTER

FIVE

扩展 SCRAPY

5.1 downloader-middleware

介绍本库中自带的常用 DOWNLOADER_MIDDLEWARES 中间件。

需要使用本库中的配置，需要在 spider 中修改如下，此为前提：

```
from ayugespidertools.AyugeSpider import AyuSpider


# 当前 spider 要继承 AyuSpider
class DemoOneSpider(AyuSpider):
    ...
```

5.1.1 1. 随机 UA

使用 fake_useragent 库中的 ua 信息，在每次发送请求时将随机取 ua 信息，将比较常用的 ua 标识的权重设置高一点，这里是根据 fake_useragent 库中的打印信息来规划权重的，即类型最多的 ua 其权重也就越高。

1.1. 使用方法

只需激活 DOWNLOADER_MIDDLEWARES 对应的配置即可。

```
custom_settings = {
    "DOWNLOADER_MIDDLEWARES": {
        # 随机请求头
        "ayugespidertools.Middlewares.RandomRequestUaMiddleware": 400,
    },
}
```

若想查看是否正常运行，只需查看其 scrapy 的 debug 日志，或在 spider 中打印 response 信息然后查看其信息即可。

5.1.2 2. 代理

2.1. 动态隧道代理

本库以快代理为例，其各个代理种类的使用方法大致相同

2.1.1. 使用方法

激活 DOWNLOADER_MIDDLEWARES 中的动态代理配置。

```
custom_settings = {
    "DOWNLOADER_MIDDLEWARES": {
        # 动态隧道代理激活
        "ayugespidertools.Middlewares.DynamicProxyDownloaderMiddleware": 125,
    },
}
```

需要修改此项目中的 VIT 文件夹下的 .conf 对应的配置信息。

```
[KDL_DYNAMIC_PROXY]
PROXY_URL=o668.kdltps.com:15818
USERNAME=***
PASSWORD=***
```

然后即可正常运行。

2.2. 独享代理

2.2.1. 使用方法

激活 DOWNLOADER_MIDDLEWARES 中的独享代理配置。

```
custom_settings = {
    "DOWNLOADER_MIDDLEWARES": {
        # 独享代理激活
        "ayugespidertools.Middlewares.ExclusiveProxyDownloaderMiddleware": 125,
    },
}
```

需要修改此项目中的 VIT 文件夹下的 .conf 对应的配置信息。

```
[KDL_EXCLUSIVE_PROXY]
PROXY_URL=http://kps.kdlapi.com/api/getkps?orderid=***&num=100&format=json
USERNAME=***
PASSWORD=***
PROXY_INDEX=1
```

注：PROXY_INDEX 为在有多个独享代理时取的代理对应的索引值。

5.1.3 3. 发送请求库改为 requests

3.1. 使用方法

激活 DOWNLOADER_MIDDLEWARES 中的对应配置。

```
custom_settings = {
    "DOWNLOADER_MIDDLEWARES": {
        # 替换 scrapy Request 请求为 requests 的中间件
        "ayugespidertools.Middlewares.RequestByRequestsMiddleware": 401,
    },
}
```

然后在 spdier 中正常 yield scrapy.Request 即可。

5.2 pipelines

介绍本库中自带的常用 pipelines 管道。

需要使用本库中的配置，需要在 spider 中修改如下，此为前提：

```
from ayugespidertools.AyugeSpider import AyuSpider


# 当前 spider 要继承 AyuSpider
class DemoOneSpider(AyuSpider):
    ...
```

5.2.1 1. Mysql 存储

1.1. 普通存储

AyuFtyMysqlPipeline 为 mysql 存储的普通模式，具有自动创建所需数据库，数据表，自动动态管理 table 字段，表注释，也会自动处理常见（字段编码，Data too long，存储字段不存在等等）的存储问题。

1.1.1 使用方法

只需激活 DOWNLOADER_MIDDLEWARES 对应的配置即可。

```
custom_settings = {
    "ITEM_PIPELINES": {
        # 激活此项则数据会存储至 Mysql
        "ayugespidertools.Pipelines.AyuFtyMysqlPipeline": 300,
    },
}
```

然后在 spider 中按照约定的格式进行 yield item 即可，具体请查看 yield item，然后不用再去管 pipelines 了。

1.2. 异步存储

1.2.1. twisted 实现

使用 twisted 的 adbapi 实现 Mysql 存储场景下的异步操作

1.2.1.1. 使用方法

同样地，只需激活 DOWNLOADER_MIDDLEWARES 对应的配置即可。

```
custom_settings = {
    "ITEM_PIPELINES": {
        # 激活此项则数据会存储至 Mysql
        "ayugespidertools.Pipelines.AyuTwistedMysqlPipeline": 300,
    },
}
```

1.3. 运行日志记录

打开 RECORD_LOG_TO_MYSQL 参数会记录 spider 的运行情况和所依赖的数据库下（带有 crawl_time 字段的）所有表格的当前采集情况统计。

5.2.2 2. MongoDB 存储

这里就直接介绍其依赖方法，因为其它配置与上方 Mysql 场景一模一样

2.1. 普通存储

```
# 依赖 AyuFtyMongoPipeline
```

2.2. 异步存储

```
# 依赖 AyuTwistedMongoPipeline
```

downloader-middleware

了解本库中的下载中间件及使用方法。

pipelines

了解本库中的管道及使用方法。