AyugeSpiderTools

Release 3.9.5

shengchenyang

Feb 18, 2024

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 Scrapy 的功能扩展模块，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；也对其不同模块进行功能扩展，比如给 spider 挂上 Mysql engine 的单例句柄可用于 yield 入库前的去重方式之一，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。

当然，你也可以选择给此项目做出贡献，比如增加或优化某些功能等，但在此之前请提相关的 ISSUES 经确认后再开发和提交 PULL REQUESTS，以免不太符合本库场景或已废弃等原因造成你的贡献浪费，那就太可惜了！

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：

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

ayuge startproject <project_name>
```

创建爬虫脚本：

```
进入项目根目录
cd <project_name>

生成脚本
ayuge 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 ayugespidertools.items import DataItem, AyuItem
from ayugespidertools.spiders import AyuSpider
from scrapy.http import Request
from scrapy.http.response.text import TextResponse
from sqlalchemy import text


class DemoOneSpider(AyuSpider):
    name = "demo_one"
    allowed_domains = ["blog.csdn.net"]
    start_urls = ["https://blog.csdn.net/"]
    custom_settings = {
        # 数据库引擎开关，打开会有对应的 engine 和 engine_conn，可用于数据入库前去重判断
        "DATABASE_ENGINE_ENABLED": True,
        "ITEM_PIPELINES": {
            # 激活此项则数据会存储至 Mysql
            "ayugespidertools.pipelines.AyuFtyMysqlPipeline": 300,
            # 开启记录项目相关运行统计信息
            "ayugespidertools.pipelines.AyuStatisticsMysqlPipeline": 301,
        },
        "DOWNLOADER_MIDDLEWARES": {
```

(continues on next page)

(continued from previous page)

```
            # 随机请求头
            "ayugespidertools.middlewares.RandomRequestUaMiddleware": 400,
        },
    }

    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: TextResponse):
        _save_table = "demo_one"
        data_list = json.loads(response.text)["data"]
        for curr_data in data_list:
            article_detail_url = curr_data.get("articleDetailUrl")
            article_title = curr_data.get("articleTitle")
            comment_count = curr_data.get("commentCount")
            favor_count = curr_data.get("favorCount")
            nick_name = curr_data.get("nickName")

            # NOTE: 数据存储方式 1，推荐此风格写法。
            article_item = AyuItem(
                article_detail_url=article_detail_url,
                article_title=article_title,
                comment_count=comment_count,
                favor_count=favor_count,
                nick_name=nick_name,
                _table=_save_table,
            )

            # NOTE: 数据存储方式 2，需要自动添加表字段注释时的写法。但不要风格混用。
            """
            article_item = AyuItem(
                # 这里也可以写为 article_detail_url = DataItem(article_detail_url)，但没有
```

(continues on next page)

(continued from previous page)

```
字段
                # 注释功能了，那不如使用 < 数据存储方式 1>
                article_detail_url=DataItem(article_detail_url, " 文章详情链接"),
                article_title=DataItem(article_title, " 文章标题"),
                comment_count=DataItem(comment_count, " 文章评论数量"),
                favor_count=DataItem(favor_count, " 文章赞成数量"),
                nick_name=DataItem(nick_name, " 文章作者昵称"),
                _table=DataItem(_save_table, " 项目列表信息"),
            )
            """

            # NOTE: 数据存储方式 3，当然也可以直接 yield dict
            # 但 _table，_mongo_update_rule 等参数就没有 IDE 提示功能了
            """
            yield {
                "article_detail_url": article_detail_url,
                "article_title": article_title,
                "comment_count": comment_count,
                "favor_count": favor_count,
                "nick_name": nick_name,
                "_table": _save_table,
            }
            """
            self.slog.info(f"article_item: {article_item}")

            # 数据入库逻辑 -> 测试 mysql_engine / mysql_engine_conn 的去重功能。
            # 场景对应的 engine 和 engine_conn 也已经给你了，你可自行实现。以下给出示例：

            # 示例一：比如使用 sqlalchemy2 来实现查询如下：
            if self.mysql_engine_conn:
                try:
                    _sql = text(
                        f"""select `id` from `{_save_table}` where `article_detail_
↪url` = "{article_detail_url}" limit 1"""
                    )
                    result = self.mysql_engine_conn.execute(_sql).fetchone()
                    if not result:
                        self.mysql_engine_conn.rollback()
                        yield article_item
                    else:
                        self.slog.debug(f'标题为 "{article_title}" 的数据已存在')
                except Exception:
                    self.mysql_engine_conn.rollback()
```

(continues on next page)

(continued from previous page)

```
                    yield article_item
            else:
                yield article_item

            # 示例二: 使用 pandas 来实现查询如下:
            """
            try:
                sql = f'''select `id` from `{_save_table}` where `article_detail_url`
↪= "{article_detail_url}" limit 1'''
                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 article_item

                # 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 article_item
                else:
                    self.slog.error(f" 请查看数据库链接或网络是否通畅! Error: {e}")
            """
```

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` 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
ayuge genspider -l

<output>
Available templates:
  async
  basic
  crawl
  csvfeed
  xmlfeed

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

eg: ayuge 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

可以使用以下命令安装 ayugespidertools 及其依赖项：

1. 若你的数据库场景只需要 mysql 和 mongodb，且不需要本库中的扩展功能，那么直接简洁安装即可，命令如下：

```
pip install ayugespidertools
```

2. 若你的数据库场景需要本库中的其他扩展，且同样不需要本库中的扩展功能，那么安装数据库版本最好，命令如下：

   可选 1，通过以下命令安装所有包含所有数据库依赖：

```
pip install ayugespidertools[database]
```

3. 全部依赖安装命令如下：

   可选 2，通过以下命令安装所有依赖：

```
pip install ayugespidertools[all]
```

注意：若你只需要 scrapy 扩展库的简单功能，那么默认的简洁依赖安装即可；一些可选择的开发功能（都会放在 extras 部分）若要使用，请使用完整安装。

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。

可能遇到的问题

在安装时可能会遇到以下问题：

- zsh: no matches found: ayugespidertools[database]

```
# zsh 中需要修改对应的命令
pip install 'ayugespidertools[database]'
pip install 'ayugespidertools[all]'
```

- 无法安装到最新版本

  这是国内源对第三方库同步 (完整度和速度) 的问题。

```
# 1. 首先查看 pypi 上的版本信息
# 如果输出的 latest 版本信息非最新，说明你的 pip 的 pypi 源还未同步，可选择“科学上网”或手动安装。
pip index versions ayugespidertools

# 1.1. 若你可以科学访问，安装只需指定 pypi 官方源即可：
pip install ayugespidertools -i https://pypi.org/simple

# 1.2. 若访问受限，则需要手动安装：
# 先到 https://pypi.org/project/AyugeSpiderTools/#files 或 https://github.com/
↪shengchenyang/AyugeSpiderTools/releases 下载 whl 文件
# 然后 pip 安装此 whl 即可
pip install ayugespidertools-x.x.x-py3-none-any.whl[database] -i https://mirrors.
↪aliyun.com/pypi/simple/
# zsh 中的命令同样需要修改
```

(continues on next page)

(continued from previous page)

```
pip install 'ayugespidertools-x.x.x-py3-none-any.whl[database]' -i https://
↪mirrors.aliyun.com/pypi/simple/
```

- 无法查找到 ayugespidertools

  这也是国内源的完整度问题，推荐优先配置为阿里云源或者清华大学源即可，若还不行请切换到官方源。

  报错详情如下：

```
ERROR: Could not find a version that satisfies the requirement ayugespidertools␣
↪(from versions: none)
ERROR: No matching distribution found for ayugespidertools
```

  解决方法如下：

```
pip install ayugespidertools -i https://mirrors.aliyun.com/pypi/simple/ --trusted-
↪host mirrors.aliyun.com
# 或者使用官方源:
pip install ayugespidertools -i https://pypi.org/simple
```

若遇到其它的各种问题，请提 issues。

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：

```
ayuge 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
|   |-- __init__.py
|   |-- items.py                 # project items definition file, 数据库表枚举信息示例也迁移
至此
|   |-- 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
|   |-- middlewares.py              # project middlewares definition file
|   |-- pipelines.py               # project pipelines definition file
|   |-- run.py                      # scrapy 运行文件
|   |-- run.sh                      # 项目运行 shell，运行以上的 run.py，win 平台不会生成此
文件
|   |-- settings.py                # project settings definition file
|   |-- spiders                        # a directory where you'll later put your␣
↪spiders
|   |   |-- __init__.py
|   `-- VIT
|       `-- .conf                # 配置文件，用于修改 Mysql，MongoDB 等配置
|-- 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 json

from ayugespidertools.items import AyuItem
from ayugespidertools.spiders import AyuSpider
from scrapy.http import Request


class DemoEightSpider(AyuSpider):
    name = "demo_eight"
    allowed_domains = ["blog.csdn.net"]
    start_urls = ["https://blog.csdn.net/"]
    custom_settings = {
        # 打开 mysql 引擎开关，用于数据入库前更新逻辑判断
        "DATABASE_ENGINE_ENABLED": True,
        "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,
        },
    }

    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",
            },
            dont_filter=True,
        )
```

(continues on next page)

(continued from previous page)

```
    def parse_first(self, response):
        data_list = json.loads(response.text)["data"]
        for curr_data in data_list:
            article_detail_url = curr_data.get("articleDetailUrl")
            article_title = curr_data.get("articleTitle")
            comment_count = curr_data.get("commentCount")
            favor_count = curr_data.get("favorCount")
            nick_name = curr_data.get("nickName")

            article_item = AyuItem(
                article_detail_url=article_detail_url,
                article_title=article_title,
                comment_count=comment_count,
                favor_count=favor_count,
                nick_name=nick_name,
                _table="_article_info_list",
            )
            yield article_item
```

如您所见，Spider 子类化 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）。

另外，一些其它注意事项：

- 示例中的一些配置和一些功能并不是每个项目中都必须要编写和配置的，只是用于展示一些功能
- 据上条可知，可以写出很简洁的代码，删除你认为的无关配置和方法并将其配置成你自己的模板就更容易适配更多人的使用场景。

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)
python 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 1. 快速开始

你可以使用以下两种方式来快速开始

1.1. 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

1.2. 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。

2.4.2 2. 应用场景介绍

根据 DemoSpider 中的各个 spider 对一些应用场景进行简要的补充介绍，总体的介绍为：

```
# 数据存入 Mysql 的场景:
+ 1).demo_one: 根据本地 VIT 中的 .conf 取 mysql 配置
+ 3).demo_three: 根据 consul 中取 mysql 配置
+ 21).demo_mysql_nacos: 根据 nacos 中取 mysql 配置 (其它从 nacos 中获取配置的场景不再举例)
+ 5).demo_five: 异步存入 Mysql 的场景

# 数据存入 MongoDB 的场景:
+ 2).demo_two: 根据本地 VIT 中的 .conf 取 mongodb 配置
+ 4).demo_four: 根据 consul 中取 mongodb 配置
+ 6).demo_six: 异步存入 MongoDB 的场景

- 7).demo_seven: 使用 requests 来请求的场景 (已删除此功能，更推荐使用 aiohttp 方式)
+ 8).demo_eight: 同时存入 Mysql 和 MongoDB 的场景
+ 9).demo_aiohttp_example: 使用 aiohttp 来请求的场景
+ 10).demo_aiohttp_test: scrapy aiohttp 在具体项目中的使用方法示例

+ 11).demo_proxy_one: 快代理动态隧道代理示例
+ 12).demo_proxy_two: 测试快代理独享代理
+13).demo_AyuTurboMysqlPipeline: mysql 同步连接池的示例
+14).demo_crawl: 支持 scrapy CrawlSpider 的示例

# 本库中给出支持 Item Loaders 特性的示例
+15).demo_item_loader: 本库中使用 Item Loaders 的示例
-16).demo_item_loader_two: 已删除，可查看上个 demo_item_loader，目前已经可以很方便的使用 Item␣
↪Loaders 功能了

+17).demo_mongo_async: asyncio 版本存储 mongoDB 的 pipelines 示例
+18).demo_mq: 数据存入 rabbitmq 的模板示例
+19).demo_kafka: 数据存入 kafka 的模板示例
+20).demo_file: 下载图片等文件到本地的模板示例
```

基本查看以上 spider 即可了解使用方法，但有些示例还是不够详细，对以上内容进行补充。

- 以上场景有需要 consul 或 nacos 上的相关配置的示例，与 .conf 中的配置内容一致，以下为 json 格式配置的示例：

```
{
    "mysql":{
        "host":"***",
        "port":3306,
        "user":"***",
```

(continues on next page)

(continued from previous page)

```
        "password":"***",
        "database":"***",
        "charset":" 选填: 默认 utf8mb4"
    },
    "mongodb":{
        "host":"***",
        "port":27017,
        "user":"***",
        "password":"***",
        "database":"***",
        "authsource":"***",
        "authMechanism":" 选填: 默认 SCRAM-SHA-1"
    },
    "postgresql": {
        "host":"***",
        "port":5432,
        "user":"***",
        "password":"***",
        "database":"***",
        "charset":" 选填: 默认 UTF8"
    },
    "mq":{
        "host":"***",
        "port":5672,
        "username":"***",
        "password":"***",
        "virtualhost":"***",
        "queue":"***",
        "exchange":"***",
        "routing_key":"***"
    },
    "kafka":{
        "bootstrap_servers":"127.0.0.1:9092 # 若多个用逗号分隔",
        "topic":"***",
        "key":"***"
    },
    "kdl_dynamic_proxy":{
        "proxy":"o668.kdltps.com:15818",
        "username":"***",
        "password":"***"
    },
    "kdl_exclusive_proxy":{
        "proxy":"http://kps.kdlapi.com/api/getkps?orderid=***&num=100&format=json
```

(continues on next page)

(continued from previous page)

```
↪",
        "username":"***",
        "password":"***",
        "index":1
    }
}
```

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
|   |-- __init__.py
|   |-- items.py
|   |-- logs
|   |   |-- DemoSpider.log
|   |   `-- error.log
|   |-- middlewares.py
|   |-- pipelines.py
|   |-- run.py
|   |-- run.sh
|   |-- settings.py
```

(continues on next page)

(continued from previous page)

```
|   |-- spiders
|   |   |-- __init__.py
|   |   `-- spider1.py
|   `-- VIT
|       `-- .conf
|-- 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:

命令如下:

```
ayuge -h
```

输出示例如下:

```
AyugeSpiderTools 3.0.1 - no active project

Usage:
  ayuge <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 "ayuge <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：

```
ayuge 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：

```
ayuge genspider mydomain mydomain.com
```

3.1.4 启动项目

- 句法：ayuge 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：

```
ayuge 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：

```
ayuge <command> -h
```

您可以使用以下命令查看所有可用命令：

```
ayuge -h
```

启动项目

- 句法：ayuge 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：

```
ayuge startproject myproject
```

genspider

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

使用示例：

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

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

$ ayuge 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 AyuItem 适配方式
- 补充适配 add_value, add_xpath, add_css 等方法的示例

3.2.1 快速开始

scrapy 中 Item 对应本库的 AyuItem，AyuItem 可用 AyuItem(key=value) 的形式直接动态赋值，其中 value 介绍如下：

value 可选择的类型有：

- 普通字段

```
from ayugespidertools.items import AyuItem

_title = "article_title_value"
demo_item = AyuItem(
    article_title=_title
)
```

- DataItem 字段
  - 完整赋值

```
# `DataItem` 有 `key_value` 和 `notes` 两个参数，`key_value` 为存储的值；
# `notes` 在其他 (需要字段注释，比如 `Mysql`，`postgresql`) 场景中表示为字段注释，在
# `ElasticSearch` 中表示 `es document fields`。

# 普通场景
demo_item = AyuItem(
    article_title=DataItem(_title, "文章标题"),
)

# es 场景
from elasticsearch_dsl import Keyword

demo_item = AyuItem(
    article_title=DataItem(_title, Keyword()),
)
```

- 无 notes 赋值

```
# 即不需要注释，也不是 es 存储场景下，只赋值 key_value 参数

demo_item = AyuItem(
    article_title=DataItem(_title),
)

# 不过，这种不如直接使用 【普通字段】的优雅赋值方式。
```

3.2.2 实现原理

以下为 Mysql，MongoDB 和 ElasticSearch 存储时的 AyuItem 示例，其它场景的用法也都一样：

本库将所有需要存储的字段直接在对应的 Item(AyuItem) 中赋值即可，其中 _table 参数为必须参数，也可以使用 add_field 方法动态添加字段。

```
def parse(self, response):
    # 存储到 Mysql 场景时 Item 构建示例：
    ArticleMysqlItem = AyuItem(
        article_detail_url=article_detail_url,
        article_title=article_title,
        comment_count=comment_count,
        favor_count=favor_count,
        nick_name=nick_name,
        _table="_article_info_list",
    )

    # 存储到 MongoDB 场景时 Item 构建示例：
    ArticleMongoItem = AyuItem(
        article_detail_url=article_detail_url,
        article_title=article_title,
        comment_count=comment_count,
        favor_count=favor_count,
        nick_name=nick_name,
        _table="_article_info_list",
        # 这里表示以 article_detail_url 为去重规则，若存在则更新，不存在则新增
        _mongo_update_rule={"article_detail_url": article_detail_url},
    )

    # 存储到 ElasticSearch 场景时 Item 构建示例：
    # 同样地，为保持风格统一，es 存储场景中会把 es Document 中 fields 的声明
    # 放在 AyuItem 中 DataItem 的 notes 参数中。
```

(continues on next page)

(continued from previous page)

```
    # 这个参数在其他 (需要字段注释，比如 Mysql，postgresql) 场景中表示为字段注释。
    from elasticsearch_dsl import Keyword, Search, Text

    book_info_item = AyuItem(
        book_name=DataItem(
            book_name, Text(analyzer="snowball", fields={"raw": Keyword()})
        ),
        book_href=DataItem(book_href, Keyword()),
        book_intro=DataItem(book_intro, Keyword()),
        _table=DataItem(_save_table, "这里的索引注释可有可无，程序中不会使用。"),
    )


# 以上可以做到只赋值一次 AyuItem ，然后在 ITEM_PIPELINES 中激活对应的 pipelines 即可，这里是为了
方便展示；
# 如非场景需要，不推荐使用 DataItem 的方式构建 AyuItem，不太优雅。
```

以上可知，目前可直接将需要的参数在对应 Item 中直接按 key=value 赋值即可，key 即为存储至库中字段，value 为存储内容。

当然，目前也支持动态赋值，但我还是推荐直接创建好 AyuItem ，方便管理：

```
def parse(self, response):
    mdi = AyuItem(_table="table0")
    mdi.add_field("add_field1", "value1")
    mdi.add_field("add_field2", DataItem(key_value="value2"))
    mdi.add_field("add_field3", DataItem(key_value="value3", notes="add_field3 值"))
    # _table 修改可通过以下方式，同样不推荐使用
    mdi["_table"] = "table1"


# 不允许 AyuItem 中字段值的类型 (str 和 DataItem) 混用，这里是用于示例展示。
```

注：在使用 AyuItem 时，其中各字段值（除了 _mongo_update_rule）的类型都要统一，比如要么都使用 str 类型，要么都使用 DataItem 类型。

另外，本库的 item 提供类型转换，以方便后续的各种使用场景：

```
# 将本库 AyuItem 转为 dict 的方法
item_dict = mdi.asdict()
# 将本库 AyuItem 转为 scrapy Item 的方法
item = mdi.asitem()
```

3.2.3 AyuItem 使用详解

详细介绍 AyuItem 支持的使用方法：

创建 AyuItem 实例：

```
item = AyuItem(_table="ta")
```

获取字段：

```
>>> item["_table"]
'ta'
>>>
>>> # 注意：虽然也可以通过 item._table 的形式获取，但是不建议这样，显得不明了。
```

添加 / 修改字段（不存在则创建，存在则修改）：

```
>>> item["_table"] = "tab"
>>> item["title"] = "tit"
>>>
>>> # 也可通过 add_field 添加字段，但不能重复添加相同字段
>>> item.add_field("num", 10)
>>>
>>> [ item["_table"], item["title"], item["num"] ]
['tab', 'tit', 10]
```

类型转换：

```
>>> # 内置转为 dict 和 scrapy Item 的方法
>>>
>>> item.asdict()
{'title': 'tit', '_table': 'tab', 'num': 10}
>>>
>>> type(item.asitem())
<class 'ayugespidertools.items.ScrapyItem'>
```

删除字段：

```
>>> # 删除字段：
>>>
>>> del item["title"]
>>> item
{'_table': 'tab', 'num': 10}
```

3.2.4 使用示例

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

`AyuItem` 在 `spider` 中常用的基础使用方法示例，以本库模板中的 `basic.tmpl` 为例来作解释：

```
import json

from ayugespidertools.items import AyuItem
from ayugespidertools.spiders import AyuSpider
from scrapy.http import Request
from scrapy.http.response.text import TextResponse
from sqlalchemy import text


class DemoOneSpider(AyuSpider):
    name = "demo_one"
    allowed_domains = ["csdn.net"]
    start_urls = ["https://www.csdn.net/"]
    custom_settings = {
        # 数据库引擎开关，打开会有对应的 engine 和 engine_conn，可用于数据入库前去重判断
        "DATABASE_ENGINE_ENABLED": True,
        "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,
        },
    }

    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",
```

(continues on next page)

(continued from previous page)

```
            },
            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}")

        _save_table = "_article_info_list"
        # 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 = json.loads(response.text)["data"]
        for curr_data in data_list:
            article_detail_url = curr_data.get("articleDetailUrl")
            article_title = curr_data.get("articleTitle")
            comment_count = curr_data.get("commentCount")
            favor_count = curr_data.get("favorCount")
            nick_name = curr_data.get("nickName")

            article_item = AyuItem(
                article_detail_url=article_detail_url,
                article_title=article_title,
                comment_count=comment_count,
                favor_count=favor_count,
                nick_name=nick_name,
                _table=_save_table,
                # 这里表示 MongoDB 存储场景以 article_detail_url 为去重规则, 若存在则更新, 不存
在则新增
                _mongo_update_rule={"article_detail_url": article_detail_url},
            )
            self.slog.info(f"article_item: {article_item}")

            # 注意: 同时存储至 mysql 和 mongodb 时, 不建议使用以下去重方法, 会互相影响。
            # 此时更适合:
            #    1.mysql 添加唯一索引去重 (本库会根据 on duplicate key update 更新),
            #      mongoDB 场景下设置 _mongo_update_rule 参数即可;
            #    2. 或者添加爬取时间字段并每次新增的场景, 即不去重, 请根据使用场景自行选择。
            # 这里只是为了介绍使用 mysql_engine / mysql_engine_conn 来对 mysql 去重的方法。
            if self.mysql_engine_conn:
                try:
```

(continues on next page)

(continued from previous page)

```
                    _sql = text(
                        f"""select `id` from `{_save_table}` where `article_detail_
↪url` = "{article_detail_url}" limit 1"""
                    )
                    result = self.mysql_engine_conn.execute(_sql).fetchone()
                    if not result:
                        self.mysql_engine_conn.rollback()
                        yield article_item
                    else:
                        self.slog.debug(f'标题为 "{article_title}" 的数据已存在')
                except Exception as e:
                    self.mysql_engine_conn.rollback()
                    yield article_item
            else:
                yield article_item
```

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: AyuItem
- 构建对应场景的 Item
  - Mysql 存储场景需要配置 _table 参数
  - MongoDB 存储场景可能会需要 _mongo_update_rule 来设置去重的更新条件
- 最后 yield 对应 item 即可

补充：其中 AyuItem 也可以改成 DataItem 的赋值方式，那么 mysql 场景下在表字段不存在时会添加字段注释，mongodb 则没有影响。推荐直接赋值的方式，更明了。

3.2.5 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 (即本库中的 ScrapyItem)，还有就是本库推荐的 AyuItem 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：

一些规则：

注，对以上表格中内容进行扩充解释：

- 一般不推荐使用规则的方式来使用 AyuItem，推荐自行构建 Ayuitem 的逻辑更清晰更易维护，这里只是给出代码示例。

3.2.6 自定义 Item 字段和实现 Item Loaders

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。

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。即以下内容示例：

```
import scrapy


class Product(scrapy.Item):
    book_name = scrapy.Field()
    book_href = scrapy.Field()
    book_intro = scrapy.Field()
```

但是本库不固定 Item field 的内容，这样丧失了解放双手的目的。

虽然，scrapy 也可以通过使用如下方法来新增字段，但总归 scrapy 是不推荐这样的写法且不太方便：

```
Product.fields["add_field1"] = scrapy.Field()
```

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 Item 的 asitem 方法实现。具体使用方法请看文章后半段。

3.3.1 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
from ayugespidertools.items import AyuItem


```

(continues on next page)

(continued from previous page)

```
# 这是 scrapy 中的实现示例:
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
    yield l.load_item()


# 这是本库中的实现示例:
def parse(self, response):
    # 先定义所需字段
    my_product = AyuItem(
        book_name=None,
        book_href=None,
        book_intro=None,
        _table="table_name",
    )
    # 然后可根据 asitem 来使用常规的 ItemLoader 功能
    mine_item = ItemLoader(item=my_product.asitem(), selector=None)
    mine_item.default_output_processor = TakeFirst()
    mine_item.add_value("book_name", book_name)
    mine_item.add_xpath("book_href", '//div[@class="product_title"]')
    mine_item.add_css("book_intro", "p#stock")
    item = mine_item.load_item()
    yield item
```

3.3.2 使用数据类项

那本库的方式在使用 ItemLoader 时有没有缺点呢?

是的，有缺点，本库虽然支持动态添加 Item 字段，但是其实不太好实现 dataclass items 的字段类型约束和参数 default 的相关设置。本库是不推荐固定 Item 字段（比如 ayugespidertools v3.0.0 之前的版本中，会把数据都存入 alldata 的固定字段中），也不推荐 spider 中若存在不同 Item 数据类型就需要定义其对应的 Item class 的。其实各有优缺点，只是本库选择了牺牲此部分。

3.3.3 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）：

```
l = ItemLoader(my_product.asitem(), some_selector)
l.default_output_processor = TakeFirst()
l.add_xpath("name", xpath1)  # (1)
l.add_xpath("name", xpath2)  # (2)
l.add_css("name", css)  # (3)
l.add_value("name", "test")  # (4)
return l.load_item()  # (5)
```

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

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。

3.4 Settings

AyugeSpiderTools 设置允许您自定义所有 Scrapy 及 AyugeSpiderTools 组件的行为，包括核心、扩展、管道和蜘蛛本身。

若您还不清楚 Scrapy 设置的知识，请跳转至 https://docs.scrapy.org/en/latest 查看教程。

以下内容主要介绍本库在具体场景下的配置示例：

3.4.1 VIT_DIR

Default: <project_dir>/<project_name>/VIT

项目运行配置 .conf 的路径，用于存放项目在不同场景下的配置，比如数据库链接配置。

3.4.2 logger

Default: `loguru.logger`

用于日志记录，可按需设置。可与 `scrapy` 中的 `LOG_LEVEL` 和 `LOG_FILE` 一同设置。

```
logger.add(
    "logs/error.log",
    level="ERROR",
    rotation="1 week",
    retention="7 days",
    enqueue=True,
)
```

3.4.3 LOGURU_ENABLED

Default: `True`

是否开启 `loguru` 日志记录功能，开启时 `slog.<Log levels>("This is a log.")`。具体的说明请在*logging* 中查看。

3.4.4 DATABASE_ENGINE_ENABLED

旧配置名：`MYSQL_ENGINE_ENABLED`，已删除此配置名称

Default: `False`

是否打开 `database` 引擎开关，用于数据入库前更新逻辑判断。如果是 `mysql` 场景，打开此项会激活 `mysql_engine` 和 `mysql_engine_conn`；如果是 `postgresql` 场景，打开此项会激活 `postgres_engine` 和 `postgres_engine_conn`;

3.4.5 APP_CONF_MANAGE

Default: `False`

开启远程配置服务，支持 `consul` 和 `nacos` 工具，配合 `VIT_DIR` 中的 `.conf` 中的对应 `consul` 或 `nacos` 链接配置使用。如果两者都有，那优先取值 `consul`，即优先级 `consul` 大于 `nacos`。

3.4.6 AIOHTTP_CONFIG

Default:

```
{
    "proxy": None,
    "sleep": None,
    "limit": None,
    "ssl": None,
    "verify_ssl": None,
    "limit_per_host": None,
    "allow_redirects": None,
    "retry_times": None,
    "verify_ssl": None,
    "allow_redirects": None,
}
```

其中的 proxy，sleep proxy allow_redirects cookies 会在 meta 中的 args 参数中重置，即这些参数在全局的优先级小于 meta 的 args 。timeout 默认和 scrapy 的 DOWNLOAD_TIMEOUT 一致，如果想设置小于 DOWNLOAD_TIMEOUT，则在 meta 的 args 中设置。其中的默认值请以 aiohttp 为准，这里统一为 None。非常建议在 DemoSpider 项目中查看具体场景的示例代码。

3.5 Configuration

AyugeSpiderTools 将项目中的所依赖的敏感信息放入了项目的 VIT 下的 .conf 文件中独立管理。

若你有多个过多的 scrapy 项目需要管理，你可以统一指定 .conf 所在文件夹的 VIT_DIR 路径参数。

若你还有过多的机器需要维护，更推荐使用本库中的 consul 及 nacos 扩展来远程配置管理，灵活性更高。

下面来介绍 .conf 文件中的配置内容：

3.5.1 Introduction

配置格式使用 ini。

3.5.2 [nacos]

nacos 可用于远程配置管理服务，可以更敏捷和容易地管理微服务平台。

3.5.3 [consul]

consul 的配置管理功能同 nacos，是本库提供的另一选择。但请注意：consul 比 nacos 的优先级更高，如果两者都配置了会优先使用 consul 配置。

不同的是配置中的鉴权 token 参数独立了出来。

3.5.4 [mysql]

用于 mysql 存储相关场景中使用，比如创建对应的 sqlalchemy 的 engine，engine_conn 来用于去重，创建数据库连接来解决表格缺失，字段缺失等问题。

注：charset 参数选择有 Literal["utf8mb4", "gbk", "latin1", "utf16", "utf16le", "cp1251", "euckr", "greek"]，若不存在你所需请提前手动创建好表，或者随意指定后续修改表皆可。

3.5.5 [mongodb:uri]

mongodb 链接的 uri 配置方式。

3.5.6 [mongodb]

mongodb 链接的普通方式，[mongodb:uri] 和 [mongodb] 按需选择一种即可。若两种都设置了，会优先从 mongodb:uri 中获取配置。

3.5.7 [postgresql]

用于 postgresql 存储相关场景中使用，比如创建对应的 sqlalchemy 的 engine，engine_conn 来用于去重，创建数据库连接来解决表格缺失，字段缺失等问题。

3.5.8 [elasticsearch]

用于 elasticsearch 存储相关场景中使用，也具有对应的 es_engine，es_engine_conn 来用于存储前的去重 (查询及更新等自定义) 逻辑。

注：ca_certs，client_cert，client_key，ssl_assert_fingerprint 中只用配置一个即可，若 verify_certs 设置为 false 则都不用配置以上参数，但推荐开启此参数。

3.5.9 [mq]

推送到 RabbitMQ 场景所需的参数。以下配置参数与 pika 中一致，这里放入 pika 文档，请自行对照查看。

3.5.10 [oracle]

用于 oracle 存储相关场景中使用，比如创建对应的 sqlalchemy 的 engine，engine_conn 来用于去重，但不会处理数据库表及字段缺失等错误，请提前创建好，因为其部分报错不如 mysql 及 postgresql 那样清晰明了，虽然也能解决，但必要性不高。

3.5.11 [kafka]

推送到 kafka 场景所需的参数。以下配置参数与 kafka-python 中一致，这里放入 kafka-python 文档，请自行对照查看。

3.5.12 [kdl_dynamic_proxy]

快代理动态代理配置参数。

3.5.13 [kdl_exclusive_proxy]

快代理动态代理配置参数。

3.5.14 [oss:ali]

上传到阿里云 oss 的配置参数。

遵守规则时的 oss 上传逻辑时使用，但更推荐自行实现和构建 AyuItem 的方式。具体请看 demo_oss 和 demo_oss_sec 的场景示例。请自行选择可接受的风格。

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`。

Settings

了解如何配置 `AyugeSpiderTools` 并查看所有可用设置。

Configuration

了解如何配置 `AyugeSpiderTools` 的 `.conf` 内容。

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 = "logs/DemoSpider.log"

# 配置中 DemoSpider 是与 ayuge 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(
    "logs/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

以下是本库 slog 日志在 spider 中的使用示例:

```
import ayugespidertools


class MySpider(ayugespidertools.AyuSpider):
    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}")
```

注: 不影响 scrapy 自带的日志记录，可自行选择或同时使用。

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.spiders 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=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=http://kps.kdlapi.com/api/getkps?orderid=***&num=100&format=json
username=***
password=***
index=1
```

注：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. 发送请求方式改为 aiohttp

3.1. 使用方法

激活 DOWNLOADER_MIDDLEWARES 中的对应配置。

```
custom_settings = {
    "TWISTED_REACTOR": "twisted.internet.asyncioreactor.AsyncioSelectorReactor",
    "DOWNLOADER_MIDDLEWARES": {
        # 将 scrapy Request 替换为 aiohttp 方式
        "ayugespidertools.middlewares.AiohttpDownloaderMiddleware": 543,
    },
    # scrapy Request 替换为 aiohttp 的配置示例
    "AIOHTTP_CONFIG": {
        "proxy": "http://127.0.0.1:7890",
        "sleep": 1,
        # 同时连接的总数
        "limit": 100,
        # 同时连接到一台主机的数量
        "limit_per_host": 0,
        "retry_times": 3,
        "verify_ssl": False,
        "allow_redirects": False,
    },
    # aiohttp 的超时时间也用这个配置，统一管理且不允许修改
    "DOWNLOAD_TIMEOUT": 25,
}
```

注：

- TWISTED_REACTOR 的配置在本库的 settings 中就默认打开的，这里配置是为了演示，不用再次配置的；
- LOCAL_AIOHTTP_CONFIG 为 aiohttp 的全局配置，一般可用来配置 proxy，timeout，retry_times，sleep 等全局的参数的；

然后需要构造 AiohttpRequest 或 AiohttpFormRequest 请求对象，具体的 aiohttp 参数在 args 中配置：

具体使用习惯和方式看个人选择，如下：

```
# 方式一： 通过 args 参数传值
yield AiohttpRequest(
    url="http://httpbin.org/get?get_args=1",
    callback=self.parse_get_fir,
    meta={
        "meta_data": "这是用来测试 parse_get_fir meta 的功能",
    },
    args=AiohttpRequestArgs(
        method="GET",
        headers={
            "Cookie": "headers_ck_key1=ck; headers_ck_key2=ck",
        },
        cookies={
            "ck_key": "ck",
        },
    ),
    dont_filter=True,
)

# 方式二：使用 scrapy 传统方式传值
yield AiohttpRequest(
    url="http://httpbin.org/get?get_args=1",
    callback=self.parse_get_fir,
    headers={
        "Cookie": "headers_ck_key1=ck; headers_ck_key2=ck",
    },
    cookies={
        "ck_key": "ck",
    },
    meta={
        "meta_data": "这是用来测试 parse_get_fir meta 的功能",
    },
    dont_filter=True,
)

# 同样，发送 post 也可以选择两种方式
# 测试 POST 请求示例一 - normal
post_data = {"post_key1": "post_value1", "post_key2": "post_value2"}
yield AiohttpRequest(
    url="http://httpbin.org/post",
```

(continues on next page)

(continued from previous page)

```
    method="POST",
    callback=self.parse_post_fir,
    headers={
        "Cookie": "headers_ck_key=ck; headers_ck_key2=ck",
    },
    body=json.dumps(post_data),
    cookies={
        "ck_key": "ck",
    },
    meta={
        "meta_data": "这是用来测试 parse_post_fir meta 的功能",
    },
    cb_kwargs={
        "request_name": "normal_post1",
    },
    dont_filter=True,
)
# 测试 POST 请求示例一 - aiohttp args
yield AiohttpRequest(
    url="http://httpbin.org/post",
    callback=self.parse_post_fir,
    args=AiohttpRequestArgs(
        method="POST",
        headers={
            "Cookie": "headers_ck_key=ck; headers_ck_key2=ck",
        },
        cookies={
            "ck_key": "ck",
        },
        data=json.dumps(post_data),
    ),
    meta={
        "meta_data": "这是用来测试 parse_post_fir meta 的功能",
    },
    cb_kwargs={
        "request_name": "aiohttp_post1",
    },
    dont_filter=True,
)

# 测试 POST 请求示例二 - normal
yield AiohttpFormRequest(
    url="http://httpbin.org/post",
```

(continues on next page)

(continued from previous page)

```
    headers={
        "Cookie": "headers_ck_key=ck; headers_ck_key2=ck",
    },
    cookies={
        "ck_key": "ck",
    },
    formdata=post_data,
    callback=self.parse_post_sec,
    meta={
        "meta_data": "这是用来测试 parse_post_sec meta 的功能",
    },
    cb_kwargs={
        "request_name": "normal_post2",
    },
    dont_filter=True,
)
# 测试 POST 请求示例二 - aiohttp args
yield AiohttpFormRequest(
    url="http://httpbin.org/post",
    callback=self.parse_post_sec,
    args=AiohttpRequestArgs(
        method="POST",
        headers={
            "Cookie": "headers_ck_key=ck; headers_ck_key2=ck",
        },
        cookies={
            "ck_key": "ck",
        },
        data=post_data,
    ),
    meta={
        "meta_data": "这是用来测试 parse_post_sec meta 的功能",
    },
    cb_kwargs={
        "request_name": "aiohttp_post2",
    },
    dont_filter=True,
)
```

5.2 pipelines

介绍本库中自带的常用 pipelines 管道。

需要使用本库中的 pipelines 的模板，需要在 spider 中修改如下：

```
from ayugespidertools.items import AyuItem
from ayugespidertools.spiders import AyuSpider


class DemoOneSpider(AyuSpider):  # 第一处
    name = "demo_es"
    custom_settings = {
        "ITEM_PIPELINES": {
            # 比如使用 Mysql 存储场景，激活此项则数据会存储至 Mysql
            "ayugespidertools.pipelines.AyuFtyMysqlPipeline": 300,  # 第二处
        },
    }
    ...

    def parse(self, response):
        ...
        yield AyuItem(...)  # 第三处
```

由上可知，使用还是比较简单的，记得其中三处的内容即可。以下内容不再对写法进行描述，都是相同的。

5.2.1 1. Mysql 存储

1.1. AyuFtyMysqlPipeline

1.1.1. 介绍

AyuFtyMysqlPipeline 为 mysql 同步存储的普通模式，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。

属于经典的示例，也是网上教程能搜到最多的存储方式。

1.1.2. 相关示例

可在 DemoSpider 中的 demo_one，demo_three，demo_crawl，demo_eight，demo_file，demo_item_loader，demo_mysql_nacos 中查看具体的代码示例。

1.2. AyuTwistedMysqlPipeline

1.2.1. 介绍

结合 twisted 实现 Mysql 存储场景下的异步操作。同样不用手动创建数据库表及字段。

比较推荐此方式方式，比较成熟。

1.2.2. 相关示例

可在 DemoSpdider 项目中的 demo_five 中查看。

1.3. AyuAsyncMysqlPipeline

1.3.1. 介绍

结合 aiomysql 实现的 async 异步存储功能。目前需要手动创建数据库表及字段。

1.3.2. 相关示例

可在 DemoSpdider 项目中的 demo_aiomysql 中查看示例。

5.2.2 2. MongoDB 存储

2.1. AyuFtyMongoPipeline

AyuFtyMysqlPipeline 为 mongodb 同步存储的普通模式。

可在 DemoSpider 中的 demo_two，demo_four，demo_eight 中查看具体的代码示例。

2.2. AyuTwistedMongoPipeline

结合 twisted 实现 mongodb 存储场景下的异步操作。

可在 DemoSpdider 项目中的 demo_six 中查看示例。

2.3. AyuAsyncMongoPipeline

结合 motor 实现的 async 的异步存储功能。

可在 DemoSpdider 项目中的 demo_mongo_async 中查看示例。

5.2.3 3. PostgreSql 存储

就不再分别介绍了，命名规则一致，可通过对应的 AyuFtyPostgresPipeline，AyuTwistedPostgresPipeline，AyuAsyncPostgresPipeline 即可知其具体的场景及功能。其中 asyncio 场景下也暂不支持自动创建库表及字段。

5.2.4 4. Oracle 存储

同样地，具有的 pipelines 有 AyuFtyOraclePipeline 和 AyuTwistedOraclePipeline，但全都没有自动创建库表的功能，因为其相关报错没有其他库那么精准，虽也可实现但没有必要，请手动创建所需的库表及字段。

开发时候 oracledb 还不支持 asyncio 异步编程，目前 v2.0.0 已经支持，我也会在其稳定时添加其支持。

5.2.5 5. ElasticSearch 存储

同样地，具有的 pipelines 有 AyuFtyESPipeline 和 AyuAsyncESPipeline，没有结合 twisted 实现的异步方式。

可在 DemoSpdider 项目中的 demo_es 和 demo_es_async 中查看示例。

5.2.6 6. 消息推送服务

6.1. mq

> 此场景下给出的是以 pika 实现的 RabbitMQ 的示例

对应的 pipelines 名称为 AyuMQPipeline，其中 .conf 中的所需配置如下：

```
[mq]
host=***
port=5672
username=***
password=***
virtualhost=***
queue=***
exchange=***
routing_key=***

# 一般只需配置以上参数即可，因为会有一些默认值，如果不需更改则不用配置，比如以下为非必须参数及其默认值：
durable=True
exclusive=False
auto_delete=False
content_type="text/plain"
delivery_mode=1
mandatory=True
```

然后在 spider 中 yield 你所需结构的 item 即可（类型为 dict）。

6.2. kafka

此场景给出的是以 kafka-python 实现的 kafka 推送示例

对应的 pipelines 名称为 AyuKafkaPipeline，其中 .conf 中的所需配置如下：

```
[kafka]
bootstrap_servers=127.0.0.1:9092 # 若多个用逗号分隔
topic=***
key=***
```

然后在 spider 中 yield 你所需结构的 item 即可（类型为 dict）。

5.2.7 7. 文件下载

需要激活 ITEM_PIPELINES 对应的配置，然后在项目中配置相关参数。

spider 中的 custom_settings 所需配置如下：

```
"ITEM_PIPELINES": {
    "ayugespidertools.pipelines.FilesDownloadPipeline": 300,
    # 以下 AyuFtyMysqlPipeline 非必须，但只激活 FilesDownloadPipeline 时只会下载文件，但是
    # 并不会将信息与网页中的标题、描述等信息绑定，激活 AyuFtyMysqlPipeline 之类的选项后，可以自行
    # 添加其它可以描述文件的详细字段并存储对应场景的数据库中。
```

(continues on next page)

(continued from previous page)

```
    "ayugespidertools.pipelines.AyuFtyMysqlPipeline": 301,
}
```

spider 等其它项目配置中的所需详细设置示例如下：

```
from pathlib import Path

custom_settings = {
    "ITEM_PIPELINES": {
        "ayugespidertools.pipelines.FilesDownloadPipeline": 300,
        "ayugespidertools.pipelines.AyuFtyMysqlPipeline": 301,
    },
    # 下载文件保存路径
    "FILES_STORE": Path(__file__).parent.parent / "docs",
}
```

具体示例请在 DemoSpider 项目中的 demo_file 和 demo_file_sec 查看。

5.2.8 8. oss 上传

此场景给出的是以 oss2 实现的 oss 上传示例

对应的 pipelines 名称为 AyuAsyncOssPipeline，其中 .conf 中的所需配置如下：

具体的配置解释不再介绍了，请在 item 部分查看。

```
[oss:ali]
access_key=
access_secret=
endpoint=
bucket=
doc=
upload_fields_suffix=_file_url
oss_fields_prefix=_
```

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。

CHAPTER

SIX

构建你的专属库

6.1 How-To-Build-Your-Own-Library

本库由 poetry 包管理工具构建，任何修改本库后自定义打包等需求请以 poetry 官方文档为准。

6.1.1 前言

本库是把 Scrapy 的一些常用功能（扩展功能和开发中常用方法）封装成了一个库，方便大家快速使用。

但在使用本库的过程中，你可能会遇到一些问题:

- 比如模板中某些配置不符合你的项目需求;
- 不喜欢项目结构的设计;
- 依赖库的版本不适合你的需求。

像这些可能包含非常个性化的定制，无法适配所有人的喜好，也无法通过 Pull Requests 合并来优雅地解决此类问题，这时候你可能会想要修改一些东西，那么你可以参考本文档，来快速构建你自己的专属库。

6.1.2 构建方法

你可以 clone 源码后，修改任意方法，修改完成后 poetry build 或 make build 即可打包并内部使用。

以更新 kafka-python 版本为例:

- Prepare: clone 项目并准备开发环境

  将项目克隆到本地，创建 python 3.8+ 虚拟环境并安装 poetry，然后在项目根目录下运行 poetry install 安装依赖即可。

- Make your changes: 自定义更改的内容

  修改你所关注的部分，比如你的项目场景下可能需要其它的日志配置默认值，或添加其它的项目结构模板，更改库名等。

若需要更新本项目的 kafka-python 依赖库版本为 x.x.x，那只需 poetry add kafka-python==x.x.x 安装目标版本即可。

- Run tests & Rebuild: 测试功能并重打包

  修改完毕并测试可用后，即可通过 poetry build 或 make 工具的 make build 打包即可使用。

6.1.3 补充

若你自定义的方法对大多数人都合适的话，可以尝试将此功能添加到本项目，但是在此之前请先发相关的 ISSUES 确认可行后再开发和提交对应 PULL REQUESTS，以免浪费了你做出的贡献。

6.2 贡献

在编写和提交 pull request 前，建议先创建一个对应 issues 并在其中讨论相关详细信息。

6.2.1 前提准备

本指南假设您已拥有 github 账户，以及 python3，虚拟环境和 git 的安装配置。但不会限制你使用的工具，比如你可以使用 virtualenv 代替 pyenv。

1. fork AyugeSpiderTools
2. Clone your forked repository

```
git clone https://github.com/<username>/AyugeSpiderTools
cd AyugeSpiderTools
```

3. Create a virtual environment

```
pyenv virtualenv 3.8.5 venv
pyenv activate venv
pip install poetry
poetry install
pre-commit install
```

4. run a test

```
pytest tests/test_items.py
```

6.2.2 开发工作流

本项目有两个分支，分别是 master 和 feature，master 为稳定分支，feature 分支活跃度较高，通常情况下新功能及 bug 修复等通过此分支测试后才会最终同步到 master 分支。所以，若您有 pull request 需求请推送至 feature 。若您不太了解 pull request 流程，我会在以下部分介绍，并给出参考文章。

注意：请完成以上前提准备，以下操作皆在你的 repo 中操作。

1. checkout the feature branch

```
git checkout feature
```

2. Create and checkout a new branch

   > 通常情况下，都不推荐直接在当前分支下操作（会影响后续与原作者的同步操作），需要新建一个新分支，如果这个分支修复了某个 issues，那么新建分支的名称可以为 issue#<id>

   比如本项目中，有一个 issue 标题为 安装后运行报错：ModuleNotFoundError: No module named 'yaml'，假设你的 pull request 修复了此问题，那么新建的分支名称就可以为 issue#9，或者为 fix-ModuleNotFoundError-yaml，这里只是给出建议，具体名称可自定义，通俗易懂即可。

```
git checkout -b <new-branch>
```

   以上两步也可以直接优化为一句命令 git checkout -b <new-branch> feature

3. Make your changes

4. Run tests

```
make test
```

5. Commit and push your work

```
git add .
git commit -m "Your commit message goes here"
git push -u origin <new-branch>
```

6. Create a pull request

   完成上一步后，在你 fork 的 github 项目页面上就会有创建 pull request 合并的按钮了，记得要从你 repo 的 <new-branch> 分支 pull request 到我 repo 的 feature 中，到此已完成整个流程。

6.2.3 补充

以上规则适用于绝大部分 github 开源软件的 pull request，但我的开源 repo （和绝大部分开源项目）也都允许贡献者以自己习惯的方式来操作，特别是像我一样 Star 关注度较少的项目，更不容易通过用户多次的贡献来形成 pull request 规则。所以，如果你有 pull request 的想法完全不用畏手畏脚。

How-To-Build-Your-Own-Library

如何将本库构建成为你的专属库。

贡献

贡献。

CHAPTER

SEVEN

补充说明

7.1 Release notes

7.1.1 AyugeSpiderTools 3.9.6 (2024-02-18)

Deprecations

- 无。

New features

- 无。

Bug fixes

- 修复 mysql 存储引擎 engine 参数未生效的问题。(1240e37)

Code optimizations

- 更新 `aiohttp` 依赖库版本以解决破坏兼容性的问题，同步更新 `scrapy` 依赖版本。(3f0dc5a, 246c824)
- 文档更新。

7.1.2 AyugeSpiderTools 3.9.5 (2024-01-30)

Deprecations

- 无。

New features

- mysql 场景添加 odku_enable 配置来设置是否开启 ON DUPLICATE KEY UPDATE 功能。(25d71dd)
- 添加 oss pipeline 的示例，请在 DemoSpider 中 demo_oss 和 demo_oss_sec 查看具体使用方法。(issue 16)

Bug fixes

- 解决文件下载不支持多字段下载的问题，请在 DemoSpider 中 demo_file 和 demo_file_sec 查看具体使用方法。(f836f02, f504c45)
- 解决远程配置管理中缺失的 mongodb:uri 优先级设置。(51ea7da)

Code optimizations

- mq 场景添加关闭链接处理。(ac54fd0)
- 更新 readthedocs 中的教程指南，以方便快速上手。
- 更新部分依赖库版本。

7.1.3 AyugeSpiderTools 3.9.4 (2024-01-10)

Deprecations

- 无。

New features

- 添加 elasticsearch 支持，具体示例请在 DemoSpider 中 demo_es 和 demo_es_async 查看。(issue 15, c4d048e, 7651dd3)

Bug fixes

- 无。

Code optimizations

- mypy check。(785e36a)

7.1.4 AyugeSpiderTools 3.9.3 (2023-12-30)

Deprecations

- 无。

New features

- 无。

Bug fixes

- 解决 `pip install ayugespidertools` 并执行简单场景时提示 `oracledb` 的依赖缺失问题。(e363937)

注：出现此问题又是因为未重新新建环境来测试，且使用 `Pycharm ssh` 远程开发时不会自动索引环境依赖导致，所以未检视出项目依赖的问题。后续也会添加对 `DemoSpider` 场景的测试自动化来完善测试流程。

由于对用户体验影响较大，先修复此问题并发布。

Code optimizations

- 统一代码风格。(ecb97e8)

7.1.5 AyugeSpiderTools 3.9.2 (2023-12-28)

Deprecations

- 无。

New features

- `mysql` 配置项支持自定义自动创建库表场景的 `engine` 和 `collate` 参数。(e652666)

Bug fixes

- 解决 settings 模板生成的 LOG_FILE 不是当前项目名的问题。(93c19d6)

Code optimizations

- 更新 spider 模板，模板中解析方式改为 scrapy 的形式，防止对开发者造成理解成本。(91ad948)
- 更新 spider 模板中的 type hint，优化了开发者使用体验。(c2a0908)
- 优化一些数据库连接处理和配置解析方法等。

7.1.6 AyugeSpiderTools 3.9.1 (2023-12-22)

Deprecations

- 无。

New features

- 添加 postgresql 的 asyncio 的 AsyncConnectionPool 存储场景支持。(341e768)

Bug fixes

- 解决 asyncio 协程场景下的 spider 的 AyuItem 写法风格不兼容的问题。(66177e4)

Code optimizations

- 更新 spider 模板示例。(61e10b1)

7.1.7 AyugeSpiderTools 3.9.0 (2023-12-18)

Deprecations

- AsyncMysqlPipeline 改名为 AyuAsyncMysqlPipeline。
- AsyncMongoPipeline 改名为 AyuAsyncMongoPipeline。
- 删除 oss 的模块及依赖。

注：最新示例请在 DemoSpider 中查看，以往旧版请切换对应分支查看。

New features

- 添加 oracle 的存储场景支持，目前有 fty 及 twisted 两种方式。
- 添加 mongodb:uri 的配置方式。

Bug fixes

- 解决 asyncio mysql 协程场景下可能会出现的被垃圾回收而阻塞的问题。
- 解决 mysql 或 postgresql 的错误处理场景下由于权限等问题造成的循环递归问题。

Code optimizations

- 优化 .conf 模板示例，配置更明确且更易管理。
- mypy check.

7.1.8 AyugeSpiderTools 3.8.0 (2023-12-03)

Deprecations

- MYSQL_ENGINE_ENABLED 的配置项名改为 DATABASE_ENGINE_ENABLED，目前支持 msyql 和 postgresql。
- 安装再添加 database 选项，可通过 pip install ayugespidertools[database] 安装所需的所有数据依赖及扩展。

注意：此变更包含不兼容部分，需要着重注意的部分如下：

- 删除了 MYSQL_ENGINE_ENABLED 配置项；
- 由于 SQLAlchemy 依赖升级到了 2.0+ 新版本，与以往的去重使用有变化，具体请查看本库 readthedocs 文档。

New features

- 支持 python3.12。
- 添加 postgresql 的存储场景支持，目前有 fty 及 twisted 两种方式。
- DATABASE_ENGINE_ENABLED 的配置目前会激活对应场景中数据库的 engine 和 engine_conn 以供去重使用。
- 将 psycopg 相关的数据库扩展依赖改为可选项，可通过 pip install ayugespidertools[database] 安装所需依赖。

Bug fixes

- 无。

Code optimizations

- 优化 `type hints`。
- 更新生成脚本模板以匹配新版本，也可使用以往 `pandas` 去重方式。
- 更明确的日志信息。

7.1.9 AyugeSpiderTools 3.7.0 (2023-11-23)

Deprecations

- 获取 `nacos` 和 `consul` 中的配置时不再转小写，请按照 readthedocs 示例填写。
- 删除 `html2text` 相关依赖及代码，此场景更适合自行实现。
- 安装不再包含非核心依赖，可通过 `pip install ayugespidertools[all]` 安装全部依赖。
- 一些 `api` 变动：
- 以下是对 `extras` 相关模块所影响较大部分的介绍：

注意：

- **此变更包含不兼容部分，如果你只使用其中 `scrapy` 扩展库部分，那么除了 `nacos`，`consul` 的 `yaml` 和 `hcl` 解析外对你无影响。**
- **再次提醒，使用时请做好依赖管理，以免不兼容部分对你的影响！**

New features

- `mongo` 场景添加 `authMechanisem` 配置选项，为可选配置，默认为 `SCRAM-SHA-1`。
- 将 `numpy`，`oss`，`pillow` 等非核心依赖改为可选项，可通过 `pip install ayugespidertools[all]` 安装所有依赖。

Bug fixes

- 无。

Code optimizations

- 优化 aiohttp, cvnpil 等测试用例，将图像相关功能整理并放入 cvnpil 模块中。
- ayuge version 修改为从 __version__ 获取信息的方式。
- 更新模板，mysql_engine 的示例改为通过 sqlalchemy 的方式，减少依赖数且大部分场景运行效率更好。
- 将可选装依赖的相关的功能代码统一放入 extras 中，更易管理。

7.1.10 AyugeSpiderTools 3.6.1 (2023-11-06)

New features

- 无。

Bug fixes

- 解决 mq 推送场景时 yield AyuItem 时的错误，现可支持多种格式。
- 解决 VIT_DIR 默认参数未存储至 settings 中的问题。

Code optimizations

- 文件下载场景添加 FILES_STORE 路径不存在时的自动创建处理。
- settings 模板删除无关配置。
- 项目添加 question issues template。

7.1.11 AyugeSpiderTools 3.6.0 (2023-10-31)

Deprecations

- 一些 api 变动：

注意：**此变更包含不兼容内容，请修改不兼容部分并调试正常后再投入生产；本项目在有一些不兼容变更时，会发布 Minor 及以上的版本包，请做好依赖版本管理。**

New features

- 无。

Bug fixes

- 无。

Code optimizations

- 设置 `VIT_DIR` 默认参数。
- 去除冗余配置，统一配置风格。将一些过于复杂的模块拆分，便于管理。

7.1.12 AyugeSpiderTools 3.5.2 (2023-10-17)

New features

- 添加从 `nacos` 中获取配置的方法，若 `.conf` 中同时存在 `consul` 和 `nacos` 配置则优先使用 `consul`；即优先级 `consul` > `nacos`。

Bug fixes

- 无

Code optimizations

- 删除 `.conf` 示例中的无用配置 `wxbot`。
- 优化从本地 `.conf` 获取配置的逻辑，也提供更清晰明确的报错信息。
- `tox` 重新添加了 `windows` 场景。
- 更新 `CI` 工具版本。

7.1.13 AyugeSpiderTools 3.5.1 (2023-09-28)

Bug fixes

- 修复在 `py 3.11` 及以上版本的 `mongo` 相关场景的报错。(issue 11)

Code optimizations

- 优化 AyuItem 实现，增强可读性及用户输入体验，比如 add_field 增加 IDE 参数提示功能。
- 更新文档中 AyuItem 的使用建议及对应测试。
- 更新测试文件，比如 test_crawl 及 spider 相关方法。

7.1.14 AyugeSpiderTools 3.5.0 (2023-09-21)

Bug fixes

- 无。

Code optimizations

- scrapy 依赖升级为 2.11.0。
- 统一运行统计的方法，修改运行 stats 中有关时间的获取和计算方法。
- 添加 pre-commit 工具和 CI，提升 commit 和 pull request 体验。
- 更新 readthedocs 的新配置。
- 优化 test_crawl 的测试方法。

7.1.15 AyugeSpiderTools 3.4.2 (2023-09-15)

Bug fixes

- 修复 crawl 模板文件中 TableEnum 的导入问题。
- 修改文档中 kafka 推送示例 typo 问题。

Code optimizations

- 优化文件下载本地的逻辑，处理当 file_url 不存在时的情况。
- 优化 items，typevar 等模块的 type hint，并删除无用的类型内容。
- 设置包源的优先级。
- 增加测试用例。
- 添加 mypy 工具。

7.1.16 AyugeSpiderTools 3.4.1 (2023-09-07)

Bug fixes

- 解决 Twisted 版本更新到 23.8.0 不兼容的问题。(issue 10)

Code optimizations

- scrapy 依赖版本更新为 2.10.1。

7.1.17 AyugeSpiderTools 3.4.0 (2023-08-10)

Deprecation removals

- 无。

Deprecations

- 无。

New features

- 无。

Bug fixes

- aiohttp 超时参数由 AIOHTTP_CONFIG 中的 timeout 获取改为直接从 DOWNLOAD_TIMEOUT 中获取。解决在 aiohttp 超时参数值大于 DOWNLOAD_TIMEOUT 时，与程序整体超时设置冲突，考虑程序的整体性，而不去根据 aiohttp 的超时设置来更新项目的整体设置。

Code optimizations

- aiohttp 添加 allow_redirects 配置参数，优化对应文档示例。
- 更新 scrapy 依赖版本为 2.10.0。
- 解决部分 typo 及注解问题。

7.1.18 AyugeSpiderTools 3.3.3 (2023-08-03)

Deprecation removals

- 无。

Deprecations

- 无。

New features

- 无。

Bug fixes

- 修复解析 `yaml` 格式数据依赖缺失的问题。(issue 9)

Code optimizations

- 本库中解决 `Mysql` 的 `Unknown column 'xx' in 'field list'` 部分代码变动，比如不再根据 `item` 字段是 `crawl_time` 类型格式来给其默认字段格式 `DATE`，因为用户可能只是存储字段是这个名称，意义并不同，或者它存的是个时间戳等情况。这样需要考虑的问题太复杂了，且具有隐患，不如优先解决字段缺失问题，后续按需再手动调整表字段类型。

7.1.19 AyugeSpiderTools 3.3.2 (2023-07-26)

Deprecation removals

- 无。

Deprecations

- 无。

New features

- 增加贝塞尔曲线生成轨迹的示例方法。

Bug fixes

- 无。

Code optimizations

- 将项目中有关文件的操作统一改为 `pathlib` 的方式。
- 根据 `consul` 获取配置的方式添加缓存处理，不用每次运行都多次调用同样参数来获取配置。减少请求次数，提高运行效率。
- 更新 `README.md` 内容，增加对应英文版本。

7.1.20 AyugeSpiderTools 3.3.1 (2023-06-29)

Deprecation removals

- 无。

Deprecations

- 无。

New features

- 无。

Bug fixes

- 无。

Code optimizations

- 优化 item 使用体验，完善功能及对应文档内容，具体请查看 readthedocs item 部分。

7.1.21 AyugeSpiderTools 3.3.0 (2023-06-21)

Deprecation removals

- 优化了 Item 体验，升级为 AyuItem，使用更方便，但注意与旧版本写法并不兼容：
  - 删除了 MysqlDataItem 实现。
  - 删除了 MongoDataItem 实现。
  - 增加了 AyuItem 参数以方便开发和简化 pipelines 结构，新示例请查看 DemoSpider 项目或 readthedocs 文档对应内容。

Deprecations

- 无。

New features

- 添加文件下载的示例，具体内容及示例请查看 readthedocs 上对应内容，具体案例请查看 DemoSpider 中的 demo_file 项目。

Bug fixes

- 无。

Code optimizations

- 升级依赖库中 numpy 和 loguru 版本，避免其漏洞警告提示。
- 更新对应的模板生成示例，简化一些不常用的配置即注释等。

7.1.22 AyugeSpiderTools 3.2.0 (2023-06-07)

Deprecation removals

- 去除数据表前缀和集合前缀的鸡肋功能：
  - 删除了 `MYSQL_TABLE_PREFIX` 参数。
  - 删除了 `MONGODB_COLLECTION_PREFIX` 参数。
- 删除其它的鸡肋功能：
  - 移除 `runjs` 方便运行 `js` 方法的鸡肋封装。
  - 移除 `rpa` 管理自动化程序的方法。
- 删除了使用 `requests` 作为 `scrapy` 请求库的功能，推荐使用本库中 `aiohttp` 的方式。

Deprecations

- 无。

New features

- 添加 `kafka` 推送的示例，具体内容及示例请查看 readthedocs 上对应内容，具体案例请查看 DemoSpider 项目。

Bug fixes

- 无。

Code optimizations

- 增加 `RabbitMQ` 中 `heartbeat` 和 `socket_timeout` 参数可自定义的功能。
- 整理依赖文件。

7.1.23 AyugeSpiderTools 3.1.0 (2023-05-30)

Deprecation removals

- 无。

Deprecations

- 无。

New features

- 添加 mq 推送的示例，具体内容及示例请查看 readthedocs 上对应内容，具体案例请查看 DemoSpider 项目。

Bug fixes

- 无。

Code optimizations

- 修复部分 typo 问题。

7.1.24 AyugeSpiderTools 3.0.1 (2023-05-17)

这是一个 major 版本更新，含有 bug 修复、代码优化等。

Deprecation removals

- 删除 ayugespidertools 的 cli 名称 -> 改为 ayuge 来管理。

Deprecations

- 无。

New features

- 修改 item 实现方式，不再通过将字段都存入 alldata 中即可实现动态设置字段的功能，使用更清晰，且能更方便地使用 ItemLoaders 的功能，具体内容及示例请查看 readthedocs 上对应内容，具体案例请查看 DemoSpider 项目。

Bug fixes

- 修复不会创建表注释的问题。

Code optimizations

- 修改 `dict_keys_to_lower` 和 `dict_keys_to_upper` 的将字典 key 转为大写或小写的功能优化为嵌套字典中所有 key 都转为大写或小写。
- 将模板中 `settings.py` 中的配置读取放入库中 `update_settings` 实现，简化 `settings.py` 文件内容。
- 优化 `Makefile` 功能，简化清理 `__pycache__` 文件夹的功能。
- 修改部分 `typo` 问题。
- 更新 `readthedocs` 内容，更新测试文件。

7.1.25 AyugeSpiderTools 2.1.0 (2023-05-09)

这是一个主要更改了 `scrapy` 依赖库为 `2.9.0` 版本，含有 `bug` 修复。

Deprecation removals

- `tox` 去除 `windows` 平台的测试场景。

Deprecations

- 下一大版本将删除 `ayugespidertools` 的 `cli` 名称 -> 改为 `ayuge` 来管理。

New features

- 本库依赖库 `scrapy` 版本升级为 `2.9.0`。

Bug fixes

- 修复使用 `ayuge` 及 `ayuge -h` 命令时，未显示当前库版本的问题。

Code optimizations

- 无。

7.1.26 AyugeSpiderTools 2.0.3 (2023-05-06)

此版本为微小变动。

Deprecation removals

- 无

Deprecations

- 下一大版本将删除 `ayugespidertools` 的 `cli` 名称 -> 改为 `ayuge` 来管理。

New features

- 添加 `mongodb` 的 `asyncio` 的示例。

Bug fixes

- 无

Code optimizations

- `readthedocs` 的 `markdown` 解析由 `recommonmark` 改为 `myst-parser`，以支持更多的 `markdown` 语法。

7.1.27 AyugeSpiderTools 2.0.1 (2023-04-27)

此版本为大版本更新，修改了项目结构以统一本库及与 `scrapy` 结合的代码风格，也有一些功能完善等。最新功能示例请在 DemoSpider 或 readthedocs 中查看。

Deprecation removals

- 一些 api 变动：
- 一些参数配置变动：

注：所有配置的 key 都统一改为小写

- 一些使用方法更改：
  - 使用 AiohttpRequest 构造请求时，由之前的 meta 中的 aiohttp_args 配置参数，改为由 args 的新增参数取代，其参数类型同样为 dict，也可以为 AiohttpRequestArgs 类型，更容易输入。

Deprecations

- 下一大版本将删除 ayugespidertools 的 cli 名称 -> 改为 ayuge 来管理。

New features

- 丰富 aiohttp 请求场景，增加重试，代理，ssl 等功能。

Bug fixes

- 无

Code optimizations

- 更新测试用例。

7.1.28 AyugeSpiderTools 1.1.9 (2023-04-20)

这是一个维护版本，具有次要功能、错误修复和清理。

Deprecation removals

- 无

Deprecations

- 无

New features

- 增加 `ayuge startproject` 命令支持 `project_dir` 参数。

```
# 这将在 project dir 目录下创建一个 Scrapy 项目。如果未指定 project dir，则 project dir 将与 myproject 相同。
ayuge startproject myproject [project_dir]
```

Bug fixes

- 修复模板中 `settings` 的 `CONSUL` 配置信息没有更新为 `v1.1.6` 版本推荐的使用方法的问题。(releases ayugespidertools-1.1.6)
- 修复在 `startproject` 创建项目时生成的 `run.sh` 中的路径信息错误问题。

Code optimizations

- 添加测试用例。
- 以后的版本发布说明都会在 ayugespidertools readthedocs 上展示。

Release notes

查看最近的 `AyugeSpiderTools` 版本中有哪些变化。