



廈門大學
XIAMEN UNIVERSITY



高校大数据课程

《大数据导论》课程作业实验报告

任务

2

2022.12.5

黄勔 22920212204392

目录

一、	实验准备	3
(1)	任务目标	3
(2)	开发环境	3
(3)	准备工作	3
✓	Python 3 的安装	3
✓	请求库的安装	3
✓	解析库的安装	3
✓	爬虫框架的安装	3
◆	Scrapy 的安装	3
✓	数据库 (MongoDB) 的安装	3
二、	实验过程	4
(1)	创建项目	4
(2)	创建 item	5
(3)	创建 spider	6
(4)	解析网页	6
(5)	使用 item	7
(6)	实现翻页	8
(7)	运行测试	9
(8)	输出文件	10
(9)	使用 Pipeline	10
(10)	数据库实现	12
三、	实验结果与重现	14
(1)	实验结果	14
(2)	重现方法	14
四、	实验问题与总结	15
(1)	实验问题	15
(2)	总结	15

一、实验准备

(1) 任务目标

根据作业的要求，本次任务和任务 1 的网站相同，即网站 “Quotes to Scrape” (<https://quotes.toscrape.com/>) 编写 Python 程序 (利用 Scrapy 框架)，爬取网页中的名言数据，保存到 json 文件与 MongoDB 中。

选择这个网站的原因是首先这个网站的结构较为简单，利于初学者阅读 HTML 并快速上手编程；其次该网站的数据较为丰富，有大量的名言文本，便于我们抓取；最后这个网站的设计初衷就是让爬虫访问这个站点 (而不访问 quotes 主站)，不会让爬虫造成主站的拥堵，有利于互联网生态的维护。

本次任务提交的文件包含本实验报告、工程包 (quotespider 文件夹，包含 Scrapy 项目) 以及采集到的数据文件 (文档目录下.json 文件)，文件均包含在本目录，本实验报告主要叙述了我在分析、练习与实战编写 Scrapy 的思路过程。

(2) 开发环境

- 操作系统: Windows 10 64 位 版本 21H2
- 编程环境: Python 3.9.6
- 编辑器: Pycharm 2020

(3) 准备工作

✓—Python 3 的安装

✓—请求库的安装

✓—解析库的安装

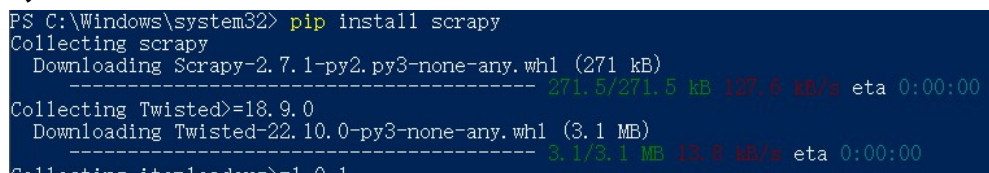
以上的内容均在“实验报告 1”有所提及，此处不再赘述。

✓ 爬虫框架的安装

◆ Scrapy 的安装

Scrapy 是一个十分强大的爬虫框架，依赖的库比较多，至少需要依赖的库有 Twisted、lxml 和 pyOpenSSL。

在命令提示符中输入命令 “pip install scrapy”，等待下载安装完毕即可。



```
PS C:\Windows\system32> pip install scrapy
Collecting scrapy
  Downloading Scrapy-2.7.1-py2.py3-none-any.whl (271 kB)
  ----- 271.5/271.5 kB 127.6 kB/s eta 0:00:00
Collecting Twisted>=18.9.0
  Downloading Twisted-22.10.0-py3-none-any.whl (3.1 MB)
  ----- 3.1/3.1 MB 13.8 kB/s eta 0:00:00
Collecting itemloaders>=1.0.1
```

图 1 安装 Scrapy 界面

✓ 数据库 (MongoDB) 的安装

MongoDB 是由 C++ 语言编写的非关系型数据库，是一个基于分布式文件存储的开源数据库系统，其内容存储形式类似 JSON 对象，它的字段值可以包含其他文档、数组及文档数组，非常灵活。在官方网站 <https://www.mongodb.com> 即可下载安装使用。

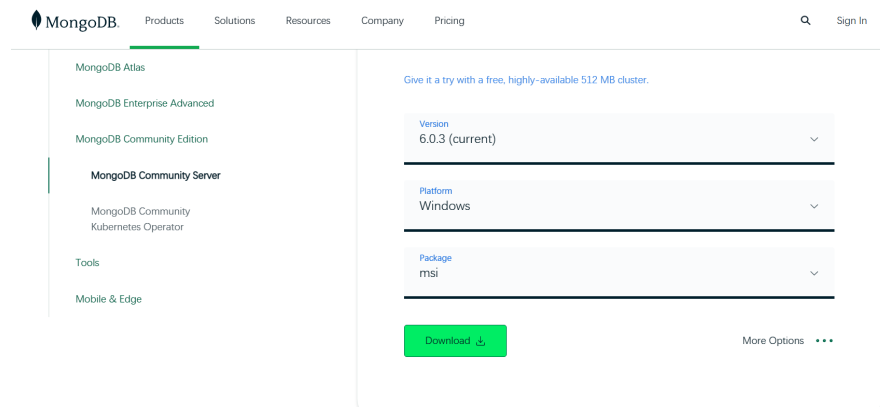


图 2 官网下载页面

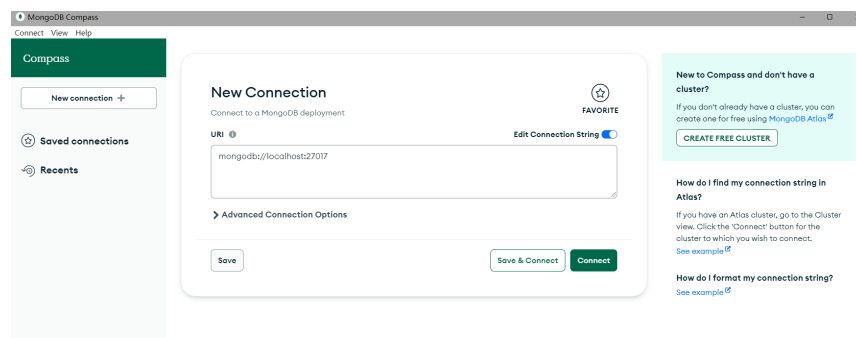


图 3 安装后打开页面

安装完毕后还要在 python 中安装对应库才能编写代码调用 MongoDB, 在命令提示符中输入命令“pip install pymongo”，等待下载安装完毕即可。

```
PS C:\Windows\system32> pip install pymongo
Collecting pymongo
  Downloading pymongo-4.3.3-cp39-cp39-win_amd64.whl (382 kB)
    ----- 382.5/382.5 kB 37.2 kB/s eta 0:00:00
Collecting dnspython<3.0.0,>=1.16.0
  Downloading dnspython-2.2.1-py3-none-any.whl (269 kB)
    ----- 269.1/269.1 kB 14.9 kB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.2.1 pymongo-4.3.3
```

图 4 安装 pymongo

二、实验过程

(1) 创建项目

在 PyCharm 中新建一个名称为“quotespider”的工程。

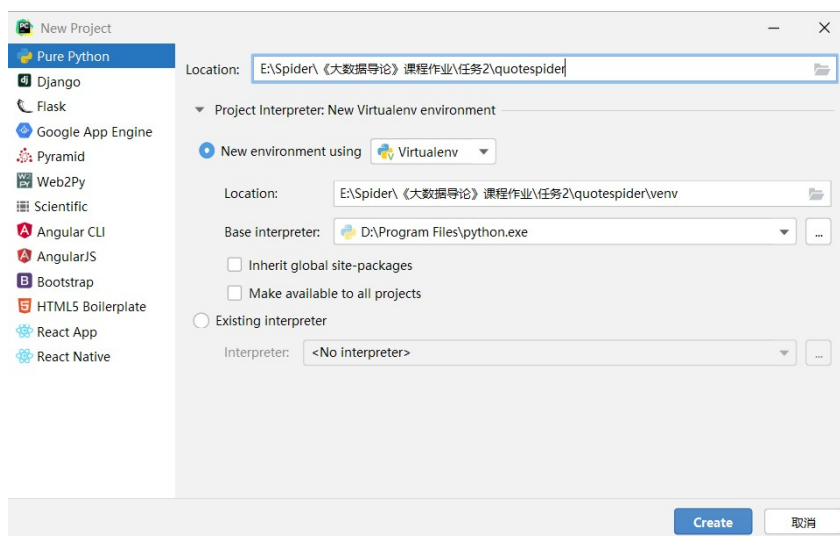


图 5 新建工程

输入命令“`scrapy startproject quotespider`”，创建 Scrapy 爬虫框架相关目录和文件。创建完成以后的具体目录结构如图所示,这些目录和文件都是由 Scrapy 框架自动创建的,不需要手动创建。

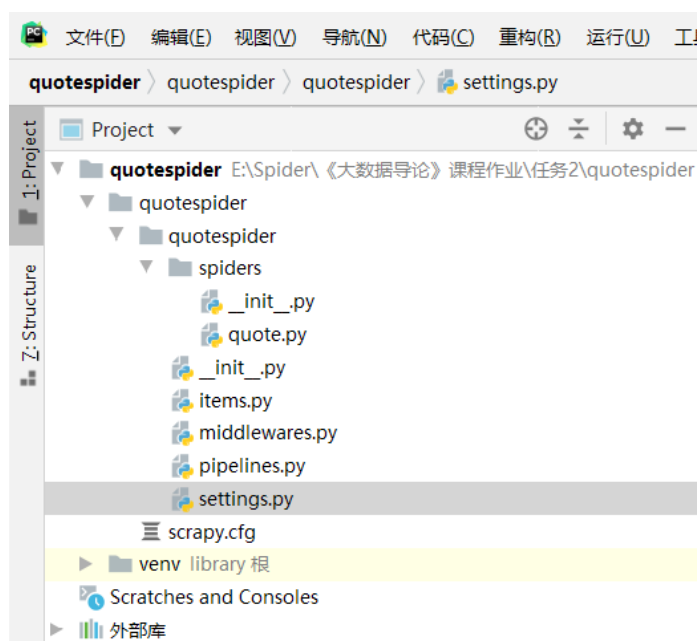


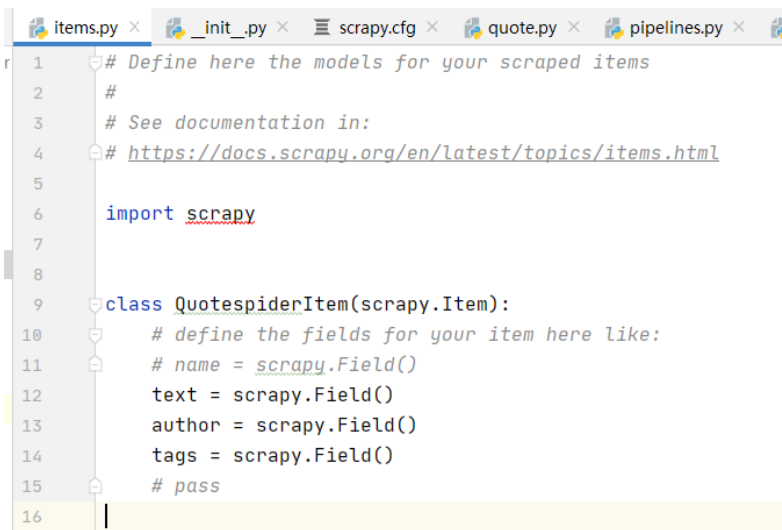
图 6 工程目录

(2) 创建 item

Item 是保存爬取数据的容器，它的使用方法和字典类似。不过，相比字典，Item 多了额外的保护机制，可以避免拼写错误或者定义字段错误。

创建 Item 需要继承 `scrapy.Item` 类，并且定义类型为 `scrapy.Field` 的字段。观察 quotes 网站，我们可以获取到内容有 `text`、`author`、`tags`。

定义 Item,此时将 `items.py` 修改如下：



```

1  # Define here the models for your scraped items
2  #
3  # See documentation in:
4  # https://docs.scrapy.org/en/latest/topics/items.html
5
6  import scrapy
7
8
9  class QuotespiderItem(scrapy.Item):
10     # define the fields for your item here like:
11     # name = scrapy.Field()
12     text = scrapy.Field()
13     author = scrapy.Field()
14     tags = scrapy.Field()
15     # pass
16

```

(3) 创建 spider

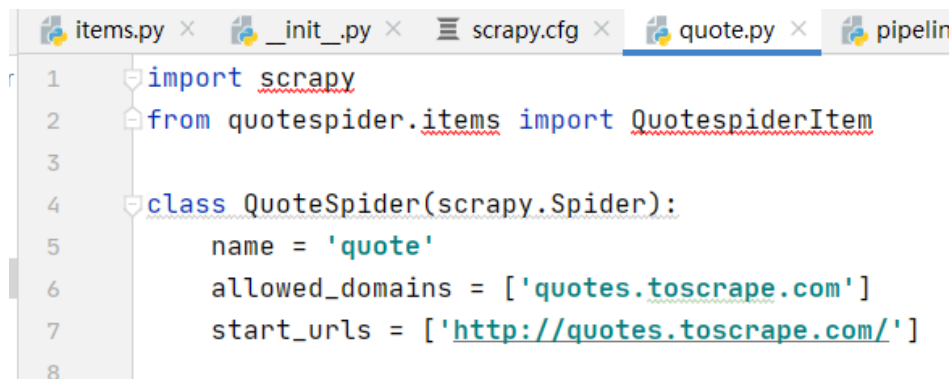
spider 是可以自己定义的类, Scrapy 用它来从网页里抓取内容, 并解析抓取的结果。不过这个类必须继承 Scrapy 提供的 Spider 类 scrapy. Spider, 还要定义 spider 的名称和起始请求, 以及怎样处理爬取后的结果的方法。

也可以使用命令行创建一个 spider。比如要生成 Quotes 这个 spider, 可以执行如下命令:

scrapy genspider quote quotes.toscrape.com

具体方式即进入刚才创建的文件夹, 然后执行 genspider 命令。第一个参数是 spider 的名称, 第二个参数是网站域名。执行完毕之后, spiders 文件夹中多了一个 quote.py, 它就是刚刚创建的 Spider

内容如下所示:



```

1  import scrapy
2  from quotespider.items import QuotespiderItem
3
4  class QuoteSpider(scrapy.Spider):
5      name = 'quote'
6      allowed_domains = ['quotes.toscrape.com']
7      start_urls = ['http://quotes.toscrape.com/']
8

```

(4) 解析网页

通过查询资料, parse() 方法的参数 response 是 start_urls 里面的链接爬取后的结果。所以在 parse() 方法中, 我们可以直接对 response 变量包含的内容进行解析, 比如浏览请求结果的网页源代码, 或者进一步分析源代码内容, 或者找出结果中的链接而得到下一个请求。

我们可以看到网页中既有我们想要的结果, 又有下一页的链接, 这两部分内容我们都要进行处理。

首先看看网页结构, 如图所示。每一页都有多个 class 为 quote 的区块, 每个区块内都包含 text、author、tags。那么我们先找出所有的 quote, 然后提取每一个 quote 中的

内容。

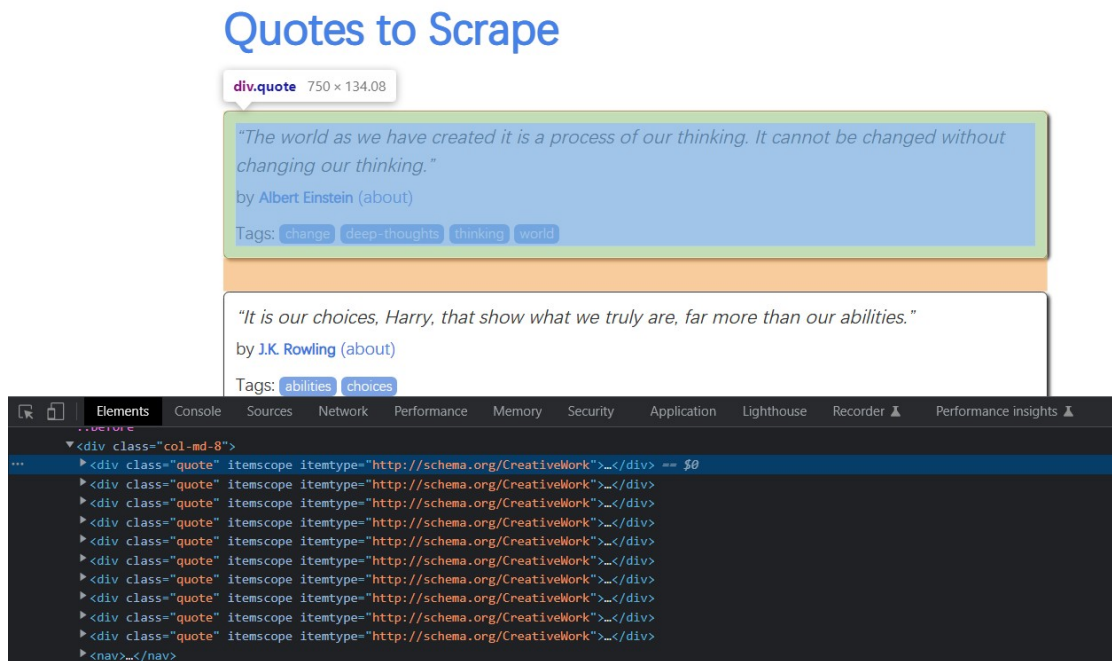


图 7 网页内容分析

在这里我使用 css 选择器选择。

这里首先利用选择器选取所有的 quote,并将其赋值为 quotes 变量,然后利用 for 循环对每个 quote 遍历,解析每个 quote 的内容。

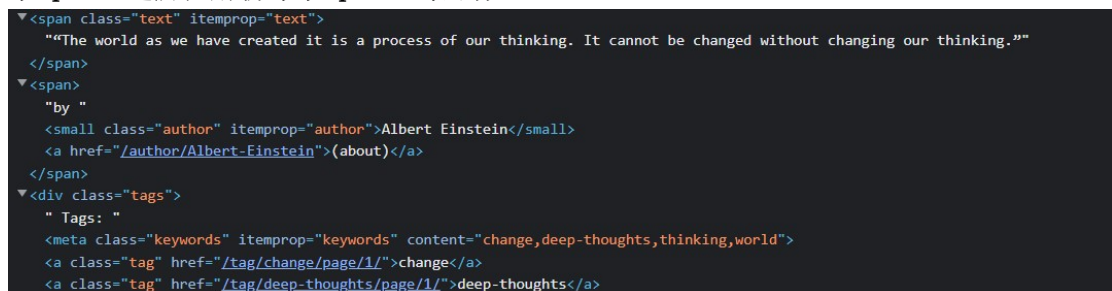


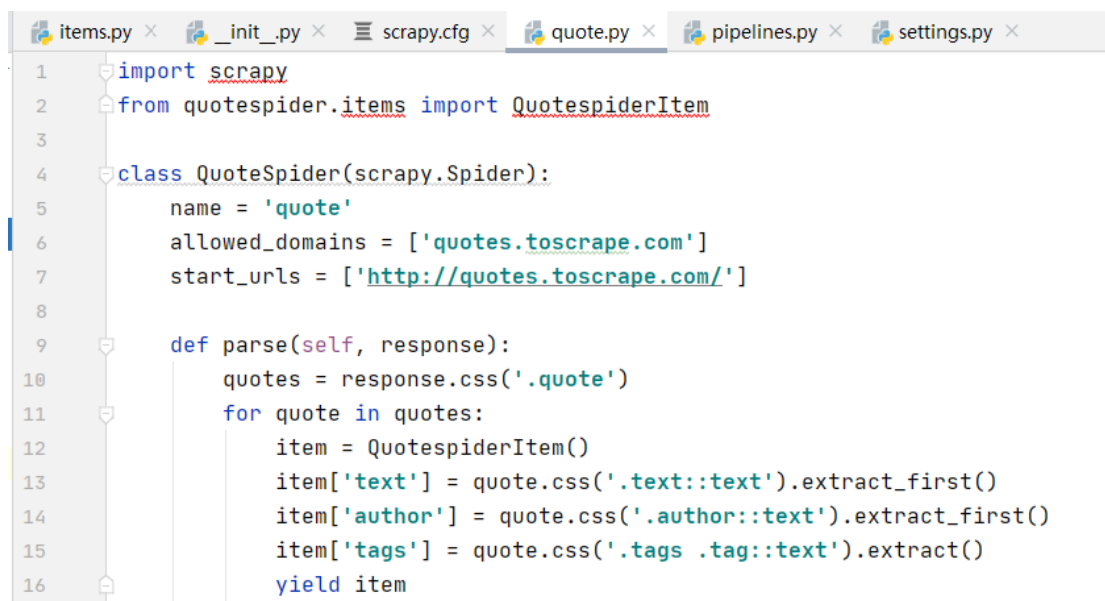
图 8 名言数据分析

对 text 来说,观察到它的 class 为 text,所以可以用 .text 选择器来选取,这个结果实际上是整个带有标签的节点,要获取它的正文内容,可以加::text 来获取。这时的结果是长度为 1 的列表,所以还需要用 extract_ first()方法来获取第一个元素。而对于 tags 来说,由于我们要获取所有的标签,所以用 extract()方法获取整个列表即可。

(5) 使用 item

上面定义了 item,接下来就要使用它了。item 可以理解为一个字典,不过在声明的时候需要实例化。然后依次用刚才解析的结果赋值 item 的每一个字段,最后将 item 返回即可。

最后 spider 的代码如下:



```

1  import scrapy
2  from quotespider.items import QuotespiderItem
3
4  class QuoteSpider(scrapy.Spider):
5      name = 'quote'
6      allowed_domains = ['quotes.toscrape.com']
7      start_urls = ['http://quotes.toscrape.com/']
8
9      def parse(self, response):
10         quotes = response.css('.quote')
11         for quote in quotes:
12             item = QuotespiderItem()
13             item['text'] = quote.css('.text::text').extract_first()
14             item['author'] = quote.css('.author::text').extract_first()
15             item['tags'] = quote.css('.tags .tag::text').extract()
16             yield item
    
```

以上的工作使得我们可以正确地解析网页的内容,并且实例化为一个个 QuotespiderItem。

(6) 实现翻页

上面的操作实现了从初始页面抓取内容。那么,下一页的内容该如何抓取?这就需要我们从当前页面中找到信息来生成下一个请求,然后在下一个请求的页面里找到信息再构造再下一个请求。这样循环往复迭代,从而实现整站的抓取。

将刚才的页面拉到最底部,如图所示。

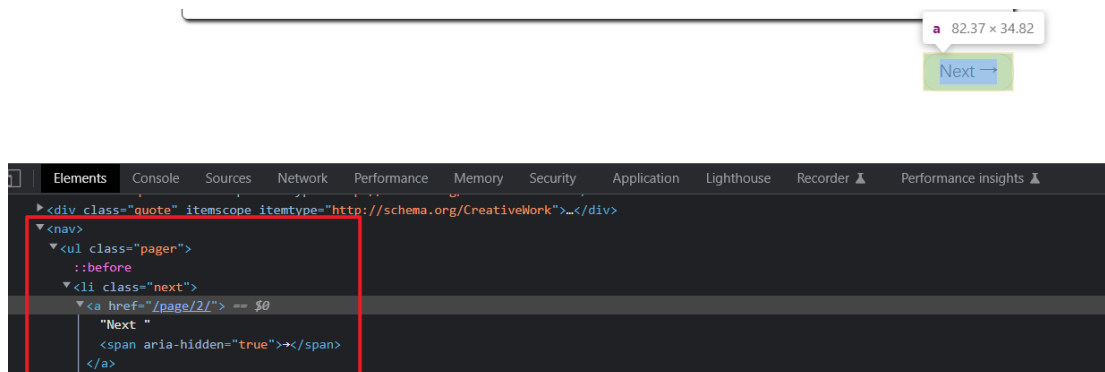


图9 下一页按钮分析

这里有一个“Next→”按钮。查看它的源代码,可以发现它的链接是/page/2/,全链接就是: <http://quotesetoscrape.com/page/2/>,通过这个链接我们就可以构造下一个请求。

构造请求时需要用到 scrapy.Request。这里我们传递两个参数 url 和 callback 来迭代发送请求。

最后的代码如下:


```

1  import scrapy
2  from quotespider.items import QuotespiderItem
3
4  class QuoteSpider(scrapy.Spider):
5      name = 'quote'
6      allowed_domains = ['quotes.toscrape.com']
7      start_urls = ['http://quotes.toscrape.com/']
8
9      def parse(self, response):
10         quotes = response.css('.quote')
11         for quote in quotes:
12             item = QuotespiderItem()
13             item['text'] = quote.css('.text::text').extract_first()
14             item['author'] = quote.css('.author::text').extract_first()
15             item['tags'] = quote.css('.tags .tag::text').extract()
16             yield item
17
18         next = response.css('.pager .next a::attr(href)').extract_first()
19         url = response.urljoin(next)
20         yield scrapy.Request(url=url, callback=self.parse)
21         # pass
    
```

(7) 运行测试

进入根目录，运行命令：“scrapy crawl quote”
就可以看到正确爬取内容的信息了。

```

(venv) E:\Spider\《大数据导论》课程作业\任务2\quotespider\quotespider>scrapy crawl quote
2022-12-05 14:24:49 [scrapy.utils.log] INFO: Scrapy 2.7.1 started (bot: quotespider)
2022-12-05 14:24:49 [scrapy.utils.log] INFO: Versions: lxml 4.7.1.0, libxml2 2.9.12, cssselect 1.1.0
8 2021, 15:26:21] [MSC v.1929 64 bit (AMD64)], pyOpenSSL 22.0.0 (OpenSSL 1.1.1n 15 Mar 2022), crypt
2022-12-05 14:24:49 [scrapy.crawler] INFO: Overridden settings:
{'BOT_NAME': 'quotespider',
 'NEWSPIDER_MODULE': 'quotespider.spiders',
 'REQUEST_FINGERPRINTER_IMPLEMENTATION': '2.7',
 'ROBOTSTXT_OBEY': True,
    
```

图 10 运行输出

图 11 运行输出 2

(8) 输出文件

运行完 Scrapy 后, 我只在控制台看到了输出结果。如果想保存结果需要 Scrapy 提供的 Feed Exports。例如, 我想将上面的结果保存成 JSON 文件, 可以执行如下命令:

`scrapy crawl quote -o quotes.json`

命令运行后, 项目内多了一个 `quotes.json` 文件, 文件包含了刚才抓取的所有内容, 内容是 JSON 格式。

图 12 json 输出

(9) 使用 Pipeline

如果想进行更复杂的操作, 如将结果保存到 MongoDB 数据库, 或者筛选某些有用的 Item, 则我们可以定义 Item Pipeline 来实现。

Item Pipeline 为项目管道。当 Item 生成后, 它会自动被送到 Item Pipeline 进行处理, 要实现 Item Pipeline 很简单, 只需要定义一个类并实现 `process_item()` 方法即可。启用 Item Pipeline 后, Item Pipeline 会自动调用这个方法。 `process_item()` 方法必须返回包含数据的字典或 Item 对象, 或者抛出 `DropItem` 异常。

`process_item()` 方法有两个参数。一个参数是 item, 每次 spider 生成的 item 都会作为

参数传递过来。另一个参数是 spider,就是 Spider 的实例。

接下来,我们实现一个 Item Pipeline,筛掉 text 长度大于 50 的 Item,并将结果保存到 MongoDB。修改项目里的 pipelines.py 文件,之前用命令行自动生成的文件内容可以删掉,增加一个 TextPipeline 类,内容如下所示:

```

5 class TextPipeline(object):
6     def __init__(self):
7         self.limit = 50
8
9     def process_item(self, item, spider):
10        if item['text']:
11            if len(item['text']) > self.limit:
12                item['text'] = item['text'][0:self.limit].rstrip() + '...'
13            return item
14        else:
15            return DropItem('Missing Text')

```

这段代码在构造方法里定义了限制长度为 50,实现了 process_item()方法,其参数是 item 和 spider。首先该方法判断 item 的 text 属性是否存在,如果不存在,则抛出 DropItem 异常;如果存在,再判断长度是否大于 50,如果大于,那就截断然后拼接省略号,再将 item 返回即可。

接下来,我们将处理后的 item 存入 MongoDB,定义另外一个 Pipeline。同样在 pipelines.py 中,我们实现另一个类 MongoPipeline,内容如下所示:

```

18 class MongoDBPipeline(object):
19     def __init__(self, connection_string, database):
20         self.connection_string = connection_string
21         self.database = database
22
23     @classmethod
24     def from_crawler(cls, crawler):
25         return cls(
26             connection_string=crawler.settings.get(
27                 'MONGODB_CONNECTION_STRING'),
28             database=crawler.settings.get('MONGODB_DATABASE')
29         )
30
31     def open_spider(self, spider):
32         self.client = pymongo.MongoClient(self.connection_string)
33         self.db = self.client[self.database]
34
35     def process_item(self, item, spider):
36         name = item.__class__.__name__
37         self.db[name].insert_one(dict(item))
38         return item
39
40     def close_spider(self, spider):
41         self.client.close()

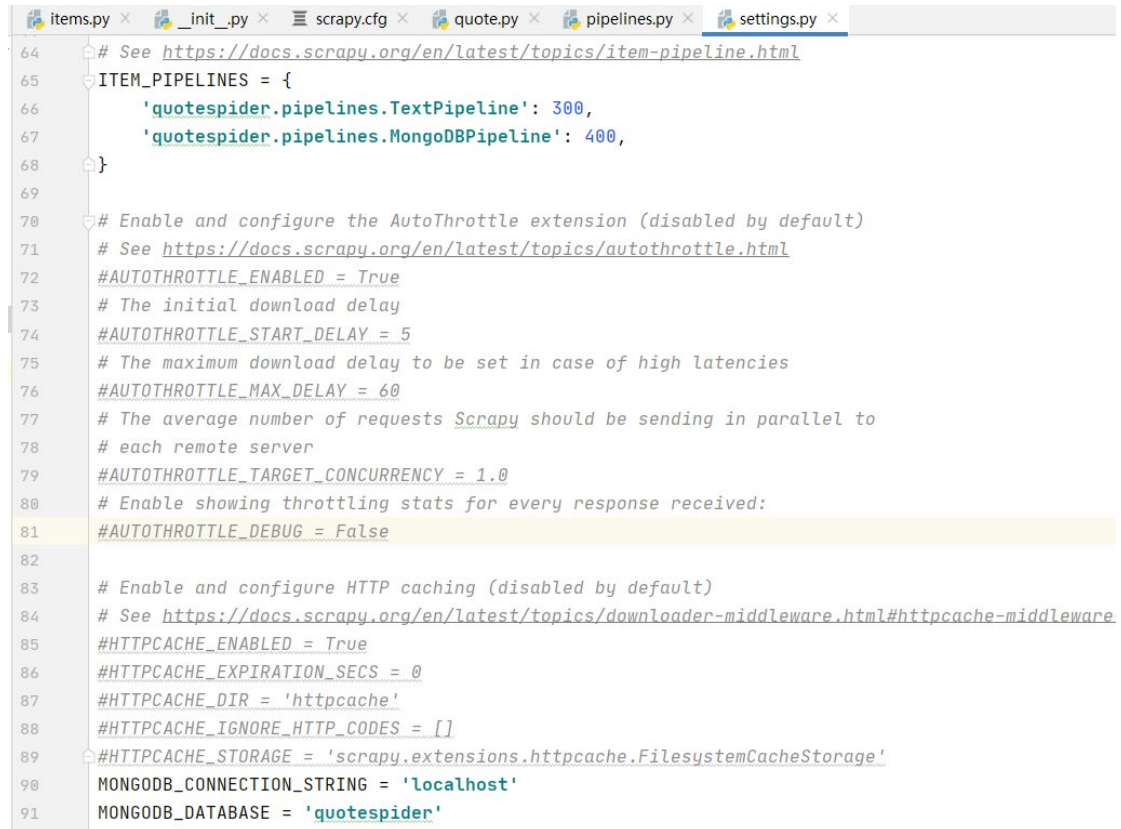
```

这个调用的方法是查询文档中给出的,可以直接构建出 MongoDB 的数据库。

定义好 TextPipeline 和 MongoPipeline 这两个类后,我们需要在 settings.py 中使用

它们。(注意在使用 MongoDB 时需要在代码头 `import pymongo`)

我们在 `settings.py` 中加入如下内容:



```

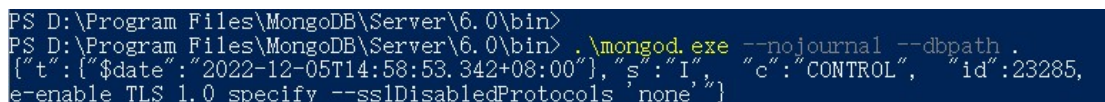
64 # See https://docs.scrapy.org/en/latest/topics/item-pipeline.html
65 ITEM_PIPELINES = {
66     'quotespider.pipelines.TextPipeline': 300,
67     'quotespider.pipelines.MongoDBPipeline': 400,
68 }
69
70 # Enable and configure the AutoThrottle extension (disabled by default)
71 # See https://docs.scrapy.org/en/latest/topics/autothrottle.html
72 #AUTOTHROTTLE_ENABLED = True
73 # The initial download delay
74 #AUTOTHROTTLE_START_DELAY = 5
75 # The maximum download delay to be set in case of high latencies
76 #AUTOTHROTTLE_MAX_DELAY = 60
77 # The average number of requests Scrapy should be sending in parallel to
78 # each remote server
79 #AUTOTHROTTLE_TARGET_CONCURRENCY = 1.0
80 # Enable showing throttling stats for every response received:
81 #AUTOTHROTTLE_DEBUG = False
82
83 # Enable and configure HTTP caching (disabled by default)
84 # See https://docs.scrapy.org/en/latest/topics/downloader-middleware.html#httpcache-middleware
85 #HTTPCACHE_ENABLED = True
86 #HTTPCACHE_EXPIRATION_SECS = 0
87 #HTTPCACHE_DIR = 'httpcache'
88 #HTTPCACHE_IGNORE_HTTP_CODES = []
89 #HTTPCACHE_STORAGE = 'scrapy.extensions.httpcache.FilesystemCacheStorage'
90 MONGODB_CONNECTION_STRING = 'localhost'
91 MONGODB_DATABASE = 'quotespider'
    
```

赋值 `ITEM_PIPELINES` 字典, 键名是 Pipeline 的类名称, 键值是调用优先级, 是一个数字, 数字越小则对应的 Pipeline 越先被调用, 如上图这样即可先对爬取的谚语文本进行截断再传入 MongoDB 中。

(10) 数据库实现

在 PyCharm 中调用 MongoDB 数据库时, 首先需要启动本地的 MongoDB 服务。

首先打开命令行窗口, `cd` 到文档夹找到 MongoDB 的 `bin` 目录。输入 “`./mongod.exe --nojournal --dbpath .`”



```

PS D:\Program Files\MongoDB\Server\6.0\bin>
PS D:\Program Files\MongoDB\Server\6.0\bin> .\mongod.exe --nojournal --dbpath .
{"t":{"$date":"2022-12-05T14:58:53.342+08:00"},"s":"I" "c":"CONTROL", "id":23285,
e-enable TLS 1.0 specify --sslDisabledProtocols 'none','}
    
```

图 13 运行本地 MongoDB 服务

如果出现 “`[listener] waiting for connections on port 27017`” 提示说明服务器已经启动成功。


```

Version value, attr: {featureCompatibilityVersion: 6.0, context: setPCV}}
{"t":{"date":"2022-12-05T14:58:54.499+08:00"},"s":"I", "c":"NETWORK", "id":4915702, "ctx":"initandlisten","msg":"Updated wire specification",
  "attr":{"oldSpec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":17},
  "outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient":true},"newSpec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},
  "incomingInternalClient":{"minWireVersion":17,"maxWireVersion":17},"outgoing":{"minWireVersion":17,"maxWireVersion":17},"isInternalClient":true}}}
{"t":{"date":"2022-12-05T14:58:54.501+08:00"},"s":"I", "c":"NETWORK", "id":4915702, "ctx":"initandlisten","msg":"Updated wire specification",
  "attr":{"oldSpec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":17},
  "outgoing":{"minWireVersion":17,"maxWireVersion":17},"isInternalClient":true},"newSpec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},
  "incomingInternalClient":{"minWireVersion":17,"maxWireVersion":17},"outgoing":{"minWireVersion":17,"maxWireVersion":17},"isInternalClient":true}}}
{"t":{"date":"2022-12-05T14:58:54.501+08:00"},"s":"I", "c":"REPL", "id":5853300, "ctx":"initandlisten","msg":"current featureCompatibilityVersion value, attr: {featureCompatibilityVersion: 6.0, context: startup}}
{"t":{"date":"2022-12-05T14:58:54.506+08:00"},"s":"I", "c":"CONTROL", "id":20536, "ctx":"initandlisten","msg":"Flow Control is enabled on this deployment"}
{"t":{"date":"2022-12-05T14:58:54.808+08:00"},"s":"I", "c":"FTDC", "id":20625, "ctx":"initandlisten","msg":"Initializing full-time diagnostic data capture", attr: {dataDirectory: "/diagnostic.data"}}
{"t":{"date":"2022-12-05T14:58:54.817+08:00"},"s":"I", "c":"STORAGE", "id":20320, "ctx":"initandlisten","msg":"createCollection", attr: {"namespace": "local.startup_log", "uuidDisposition": "generated", "uuid": {"uuid": "00f24571-209f-4c3a-bafb-6c6e6e43fe4d"}, "options": {"capped": true, "size": 10485760}}}
{"t":{"date":"2022-12-05T14:58:54.831+08:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"initandlisten","msg":"Index build: done building", attr: {"buildUUID": null, "collectionUUID": {"uuid": {"uuid": "00f24571-209f-4c3a-bafb-6c6e6e43fe4d"}, "namespace": "local.startup_log", "index": "_id", "ident": "index-3--7298329955528475654", "collectionIdent": "collection-2--7298329955528475654", "commitTimestamp": null}}}
{"t":{"date":"2022-12-05T14:58:54.832+08:00"},"s":"I", "c":"REPL", "id":6015317, "ctx":"initandlisten","msg":"Setting new configuration state, attr: {"newState": "ConfigReplicationDisabled", "oldState": "ConfigPreStart"}}
{"t":{"date":"2022-12-05T14:58:54.832+08:00"},"s":"I", "c":"STORAGE", "id":22262, "ctx":"initandlisten","msg":"Timestamp monitor starting"}
{"t":{"date":"2022-12-05T14:58:54.835+08:00"},"s":"I", "c":"NETWORK", "id":23015, "ctx":"listener","msg":"Listening on", attr: {"address": "0.0.0.0", "port": 27017, "ssl": {"off"}}}
{"t":{"date":"2022-12-05T14:58:54.836+08:00"},"s":"I", "c":"NETWORK", "id":23016, "ctx":"listener","msg":"Waiting for connections", attr: {"port": 27017, "ssl": {"off}}}
{"t":{"date":"2022-12-05T14:58:54.840+08:00"},"s":"I", "c":"CONTROL", "id":20712, "ctx":"LogicalSessionCacheReap","msg":"Sessions collection is not set up; waiting until next sessions reap interval", attr: {"error": "NamespaceNotFound: config.system.sessions does not exist"}}
{"t":{"date":"2022-12-05T14:58:54.840+08:00"},"s":"I", "c":"STORAGE", "id":20320, "ctx":"LogicalSessionCacheRefresh","msg":"createCollection", attr: {"namespace": "config.system.sessions", "uuidDisposition": "generated", "uuid": {"uuid": {"uuid": "ec7a46fc-0017-4df8-b0fd-e3eceaeaea02"}, "options": {}}}}
{"t":{"date":"2022-12-05T14:58:54.866+08:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"LogicalSessionCacheRefresh","msg":"Index build: done building", attr: {"buildUUID": null, "collectionUUID": {"uuid": {"uuid": "ec7a46fc-0017-4df8-b0fd-e3eceaeaea02"}, "namespace": "config.system.sessions", "index": "_id", "ident": "index-5--7298329955528475654", "collectionIdent": "collection-4--7298329955528475654", "commitTimestamp": null}}}
{"t":{"date":"2022-12-05T14:58:54.866+08:00"},"s":"I", "c":"INDEX", "id":20345, "ctx":"LogicalSessionCacheRefresh","msg":"Index build: done building", attr: {"buildUUID": null, "collectionUUID": {"uuid": {"uuid": "ec7a46fc-0017-4df8-b0fd-e3eceaeaea02"}, "namespace": "config.system.sessions", "index": "lsidIndex", "ident": "index-6--7298329955528475654", "collectionIdent": "collection-4--7298329955528475654", "commitTimestamp": null}}}

```

图 14 MongoDB 输出信息

再重新执行爬取，命令如下所示：

scrapy crawl quote

爬取结束后，MongoDB 中创建了一个 quotespider 的数据库、QuotespiderItem 的表，如图所示。

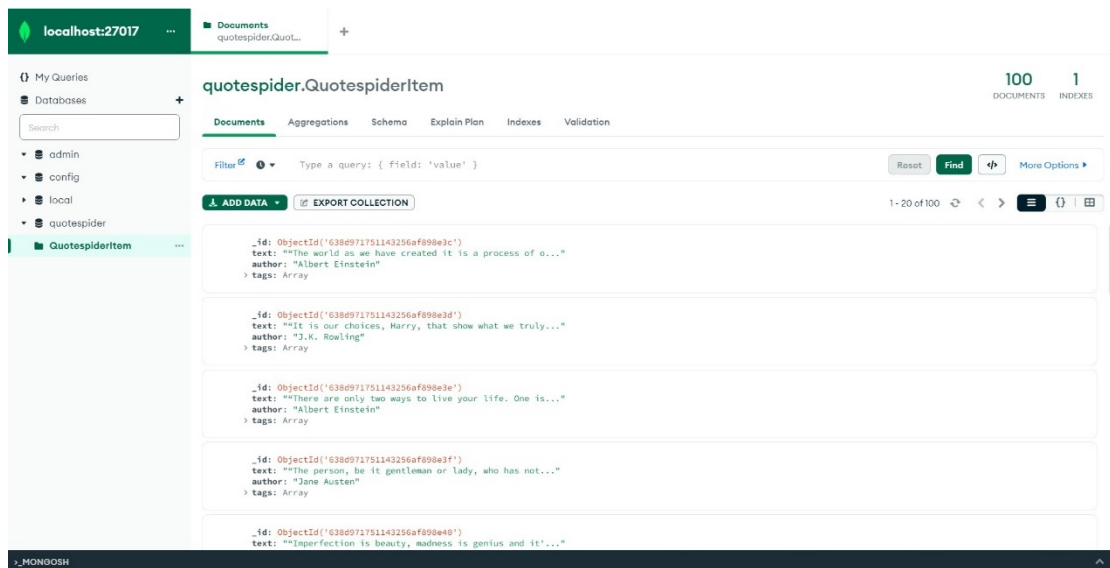


图 15 数据库页面

长的 text 已经被处理并追加了省略号, 短的 text 保持不变, author 和 tags 也都相应保存。

三、实验结果与重现

(1) 实验结果

通过上述介绍的实验过程，运行过程如下图：

```
Terminal: Local x +
'robotstxt/response_status_count/404': 1,
'scheduler/dequeued': 10,
'scheduler/dequeued/memory': 10,
'scheduler/enqueued': 10,
'scheduler/enqueued/memory': 10,
'start_time': datetime.datetime(2022, 12, 5, 7, 0, 37, 603592)}
2022-12-05 15:00:47 [scrapy.core.engine] INFO: Spider closed (finished)

(venv) E:\Spider\《大数据导论》课程作业\任务2\quotespider\quotespider>
```

图 16 运行过程

最后在 MongoDB 中查看到获取的谚语数据，已经成功完整爬取：

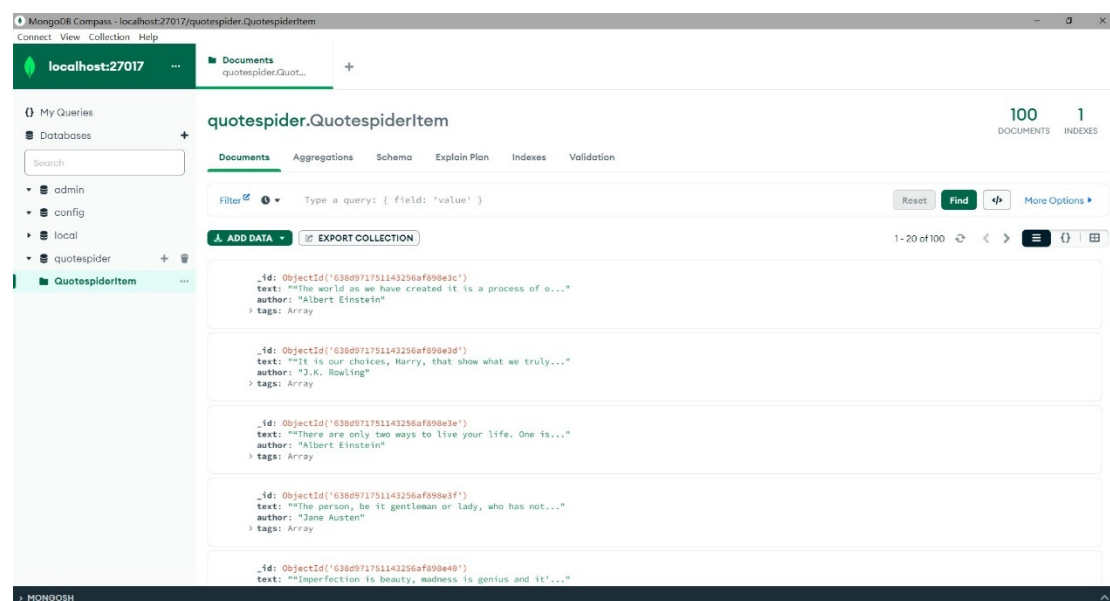


图 17 数据库页面

(2) 重现方法

进入 Project 根目录，运行命令：“scrapy crawl quote”（需要按照“准备工作”中的步骤安装相关库与运行 MongoDB）

运行成功后便能在 MongoDB 中看到爬取到的数据。

四、实验问题与总结

(1) 实验问题

- 1 运行爬虫出现下图错误

```
File "D:\Program Files\lib\site-packages\scrapy\spiderloader.py", line 77, in load
    raise KeyError(f"Spider not found: {spider_name}")
KeyError: 'Spider not found: quotespider'
```

答：这是由于没有定位到根目录中，cd 切换目录即可解决。

- 2 出现下图报错

```
pymongo.errors.ServerSelectionTimeoutError: localhost:27017: [WinError 10061] 由于目标计算机积极拒绝，无法连接
88804a9d7f4da, topology_type: Unknown, servers: [<ServerDescription ('localhost', 27017) server_type: Unknown,
计算机积极拒绝，无法连接。'>]>
```

答：在 PyCharm 中调用 MongoDB 数据库时，由于没有启动本地的 MongoDB 服务引起。

(2) 总结

在这一次实验中我对爬虫的框架运行进行了一次完整的实操，在实验过程中虽然遇到了许多困难，但是通过一步步解决问题，我也对具体 Scrapy 的编写方法有了更深刻的认识，并对设计爬虫的流程有了更多的思考，这让我受益匪浅，这是一次很有意义的实验！