数据爬虫说明

Python版本：Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)] on win32

选择任务：用scrapy实现多层爬取

反爬已经在文件中实现，我设置了USER\_AGENT、延迟等等方式来完成反爬

custom\_settings = {  
 'USER\_AGENT': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134',  
 'DOWNLOAD\_DELAY': 2,  
 'ROBOTSTXT\_OBEY': False  
}

网页分析：为了测试程序的稳定性和逻辑的正确性，我专门选择了一本4000多章的小说来进行爬虫的测试。[特种兵王在山村最新章节\_特种兵王在山村全文在线阅读\_圣手方方的小说\_笔趣阁 (bqgui.cc)](https://www.bqgui.cc/book/1510/)这个是书本所在的位置





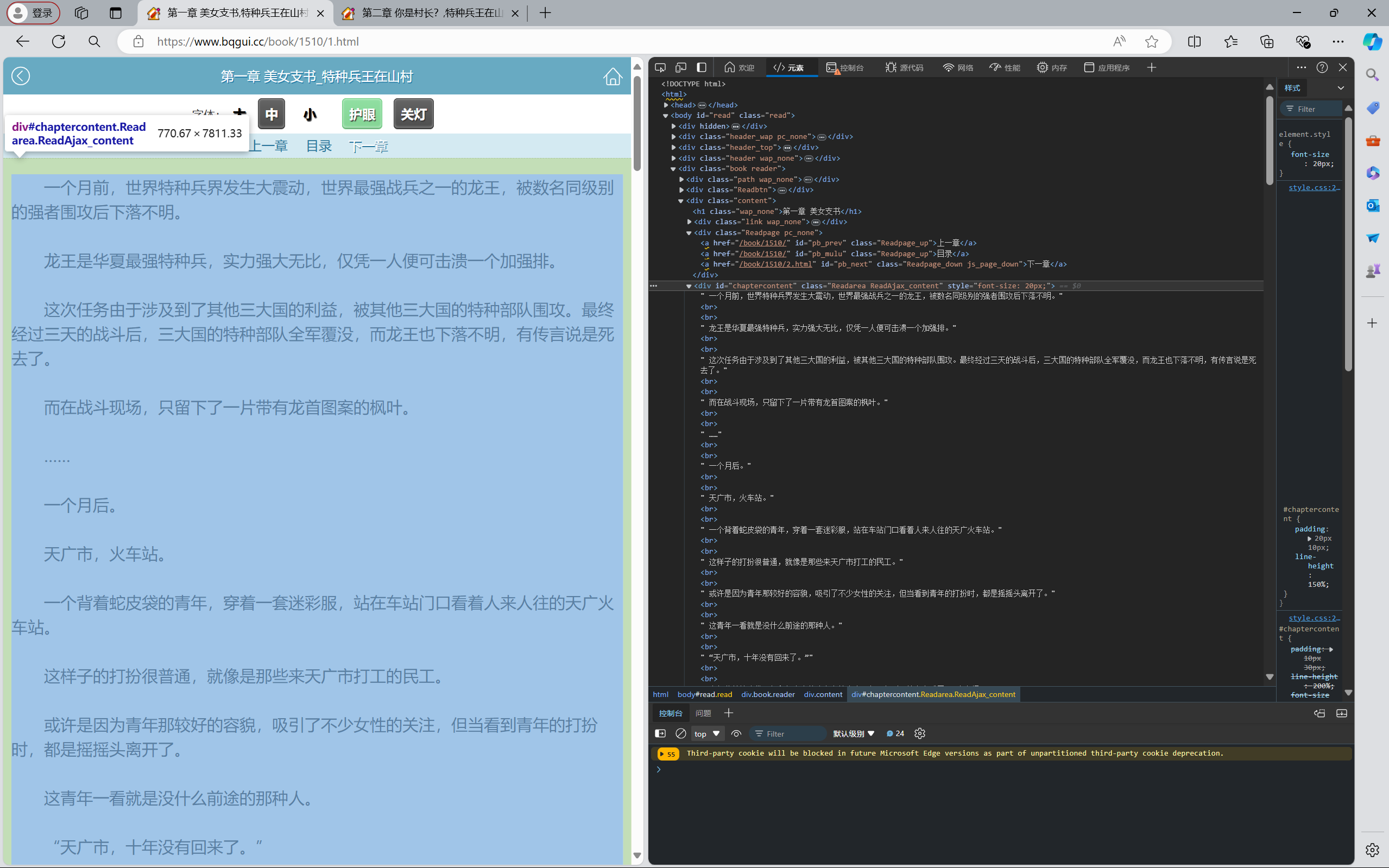


文章的章节区分是由[https://www.bqgui.cc/book/1510/{page}.html](https://www.bqgui.cc/book/1510/%7bpage%7d.html)来决定页数的（即章数），

那么我们可以在代码中实现：

current\_page = int(response.url.split('/')[-1].split('.')[0])  
next\_page = current\_page + 1  
next\_page\_link = f"https://www.bqgui.cc/book/1510/{next\_page}.html"

就可以实现翻页功能来进行多层内容的爬取。



对于网页内容和文章章节标题的解析，我们在对于html页面的解析页可以看的很清楚，title和正文所在的位置。

因为这个格式用美丽汤来解析很简单，所以我考虑用美丽汤来解决这个问题，而不是用xpath来解决和爬取。

try:

html = response.body.decode('utf-8')

except UnicodeDecodeError:

html = response.body.decode('gbk', errors='ignore')

soup = BeautifulSoup(html, 'html.parser')

这里先尝试将 response.body 解码为 UTF-8 编码的字符串，如果失败则尝试使用 GBK 编码并忽略错误。然后使用 BeautifulSoup 解析解码后的 HTML 字符串。（因为在测试代码时多次遇到了编码问题，所以在这防止报错或者是再遇到编码问题采用了这个方式）

# 提取章节标题

title = soup.select\_one('h1.wap\_none').get\_text()

# 提取章节内容

content = [p.get\_text() for p in soup.select('#chaptercontent')]

这里使用了 select\_one 方法来选择单个符合 CSS 选择器 h1.wap\_none 的元素，并获取其文本内容作为章节标题。然后使用 select 方法选择所有符合 CSS 选择器 #chaptercontent 的元素，并将它们的文本内容提取到列表中作为章节内容。

然后，对提取到的章节内容进行清理：

# 清理内容中的特殊字符和不必要的空白

cleaned\_content = ' '.join(content).replace('\u3000', '').strip()

# 移除不需要的部分,这个是每一页都有的重复的不必要的内容，可以直接删除掉

cleaned\_content = cleaned\_content.replace(

'请收藏本站：https://www.bqgui.cc。笔趣阁手机版：https://m.bqgui.cc',

''

).replace('『点此报错』『加入书签』', '').strip()

因为每一章的末尾都是固定的广告和文字，所以这里直接通过字符串替换和拼接操作，移除了内容中的特殊字符、不必要的空白和广告文字，方便并且效果很好，不会影响其他内容。

修改了以下文件：

BookSpider.py:

import scrapy  
from bookspider.items import BookspiderItem  
from bs4 import BeautifulSoup  
  
  
class BookspiderSpider(scrapy.Spider):  
 name = "BookSpider"  
 allowed\_domains = ["bqgui.cc"]  
 start\_urls = ["https://www.bqgui.cc/book/1510/1.html"]  
  
 custom\_settings = {  
 'USER\_AGENT': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134',  
 'DOWNLOAD\_DELAY': 2,  
 'ROBOTSTXT\_OBEY': False  
 }  
  
 def parse(self, response):  
 try:  
 html = response.body.decode('utf-8')  
 except UnicodeDecodeError:  
 html = response.body.decode('gbk', errors='ignore')  
  
 soup = BeautifulSoup(html, 'html.parser')  
  
 # 提取章节标题  
 title = soup.select\_one('h1.wap\_none').get\_text()  
  
 # 提取章节内容  
 content = [p.get\_text() for p in soup.select('#chaptercontent')]  
  
 # 清理内容中的特殊字符和不必要的空白  
 cleaned\_content = ' '.join(content).replace('\u3000', '').strip()  
  
 # 移除不需要的部分,这个是每一页都有的重复的不必要的内容，可以直接删除掉  
 cleaned\_content = cleaned\_content.replace(  
 '请收藏本站：https://www.bqgui.cc。笔趣阁手机版：https://m.bqgui.cc',  
 ''  
 ).replace('『点此报错』『加入书签』', '').strip()  
  
 item = BookspiderItem()  
 item['title'] = title  
 item['content'] = cleaned\_content  
  
 yield item  
  
 # Extract next page link and follow  
 current\_page = int(response.url.split('/')[-1].split('.')[0])  
 next\_page = current\_page + 1  
 next\_page\_link = f"https://www.bqgui.cc/book/1510/{next\_page}.html"  
 self.log(f'Following next page link: {next\_page\_link}')  
 yield response.follow(next\_page\_link, self.parse)

items.py:

# Define here the models for your scraped items  
#  
# See documentation in:  
# https://docs.scrapy.org/en/latest/topics/items.html  
  
import scrapy  
  
  
class BookspiderItem(scrapy.Item):  
 # define the fields for your item here like:  
 # name = scrapy.Field()  
 title = scrapy.Field()  
 content = scrapy.Field()

pipelines.py:

# Define your item pipelines here  
#  
# Don't forget to add your pipeline to the ITEM\_PIPELINES setting  
# See: https://docs.scrapy.org/en/latest/topics/item-pipeline.html  
  
  
# useful for handling different item types with a single interface  
from itemadapter import ItemAdapter  
import csv  
  
class BookspiderPipeline:  
 def open\_spider(self, spider):  
 self.file = open('bookspider.csv', 'w', newline='', encoding='utf-8')  
 self.writer = csv.writer(self.file)  
 self.writer.writerow(['title', 'content'])  
  
 def close\_spider(self, spider):  
 self.file.close()  
  
 def process\_item(self, item, spider):  
 self.writer.writerow([item['title'], item['content']])  
 return item

settings.py:

# Scrapy settings for bookspider project  
#  
# For simplicity, this file contains only settings considered important or  
# commonly used. You can find more settings consulting the documentation:  
#  
# https://docs.scrapy.org/en/latest/topics/settings.html  
# https://docs.scrapy.org/en/latest/topics/downloader-middleware.html  
# https://docs.scrapy.org/en/latest/topics/spider-middleware.html  
  
BOT\_NAME = "bookspider"  
  
SPIDER\_MODULES = ["bookspider.spiders"]  
NEWSPIDER\_MODULE = "bookspider.spiders"  
  
  
# Crawl responsibly by identifying yourself (and your website) on the user-agent  
USER\_AGENT = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/18.17763"  
  
# Obey robots.txt rules  
ROBOTSTXT\_OBEY = False  
# Configure maximum concurrent requests performed by Scrapy (default: 16)  
#CONCURRENT\_REQUESTS = 32  
  
# Configure a delay for requests for the same website (default: 0)  
# See https://docs.scrapy.org/en/latest/topics/settings.html#download-delay  
# See also autothrottle settings and docs  
#DOWNLOAD\_DELAY = 3  
# The download delay setting will honor only one of:  
#CONCURRENT\_REQUESTS\_PER\_DOMAIN = 16  
#CONCURRENT\_REQUESTS\_PER\_IP = 16  
  
# Disable cookies (enabled by default)  
#COOKIES\_ENABLED = False  
  
# Disable Telnet Console (enabled by default)  
#TELNETCONSOLE\_ENABLED = False  
  
# Override the default request headers:  
#DEFAULT\_REQUEST\_HEADERS = {  
# "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8",  
# "Accept-Language": "en",  
#}  
  
# Enable or disable spider middlewares  
# See https://docs.scrapy.org/en/latest/topics/spider-middleware.html  
#SPIDER\_MIDDLEWARES = {  
# "bookspider.middlewares.BookspiderSpiderMiddleware": 543,  
#}  
  
# Enable or disable downloader middlewares  
# See https://docs.scrapy.org/en/latest/topics/downloader-middleware.html  
#DOWNLOADER\_MIDDLEWARES = {  
# "bookspider.middlewares.BookspiderDownloaderMiddleware": 543,  
#}  
  
# Enable or disable extensions  
# See https://docs.scrapy.org/en/latest/topics/extensions.html  
#EXTENSIONS = {  
# "scrapy.extensions.telnet.TelnetConsole": None,  
#}  
  
# Configure item pipelines  
# See https://docs.scrapy.org/en/latest/topics/item-pipeline.html  
ITEM\_PIPELINES = {  
 "bookspider.pipelines.BookspiderPipeline": 300,  
}  
DOWNLOAD\_DELAY = 2  
ROBOTSTXT\_OBEY = False  
# Enable and configure the AutoThrottle extension (disabled by default)  
# See https://docs.scrapy.org/en/latest/topics/autothrottle.html  
#AUTOTHROTTLE\_ENABLED = True  
# The initial download delay  
#AUTOTHROTTLE\_START\_DELAY = 5  
# The maximum download delay to be set in case of high latencies  
#AUTOTHROTTLE\_MAX\_DELAY = 60  
# The average number of requests Scrapy should be sending in parallel to  
# each remote server  
#AUTOTHROTTLE\_TARGET\_CONCURRENCY = 1.0  
# Enable showing throttling stats for every response received:  
#AUTOTHROTTLE\_DEBUG = False  
  
# Enable and configure HTTP caching (disabled by default)  
# See https://docs.scrapy.org/en/latest/topics/downloader-middleware.html#httpcache-middleware-settings  
#HTTPCACHE\_ENABLED = True  
#HTTPCACHE\_EXPIRATION\_SECS = 0  
#HTTPCACHE\_DIR = "httpcache"  
#HTTPCACHE\_IGNORE\_HTTP\_CODES = []  
#HTTPCACHE\_STORAGE = "scrapy.extensions.httpcache.FilesystemCacheStorage"  
  
# Set settings whose default value is deprecated to a future-proof value  
REQUEST\_FINGERPRINTER\_IMPLEMENTATION = "2.7"  
TWISTED\_REACTOR = "twisted.internet.asyncioreactor.AsyncioSelectorReactor"  
FEED\_EXPORT\_ENCODING = "utf-8"

程序运行无异常：爬取的文件保存到文件夹之下，文件名为bookspider.csv