



引言

- Scrapy是一个流行的网络爬虫框架，它使用了一些高级功能以简化网站抓取。本章中，将学习使用Scrapy抓取示例网站，目标任务与第2章相同。



内容提要

Scrapy框架简介

安装Scrapy

创建Scrapy爬虫

测试Scrapy爬虫

使用Scrapy的Shell命令

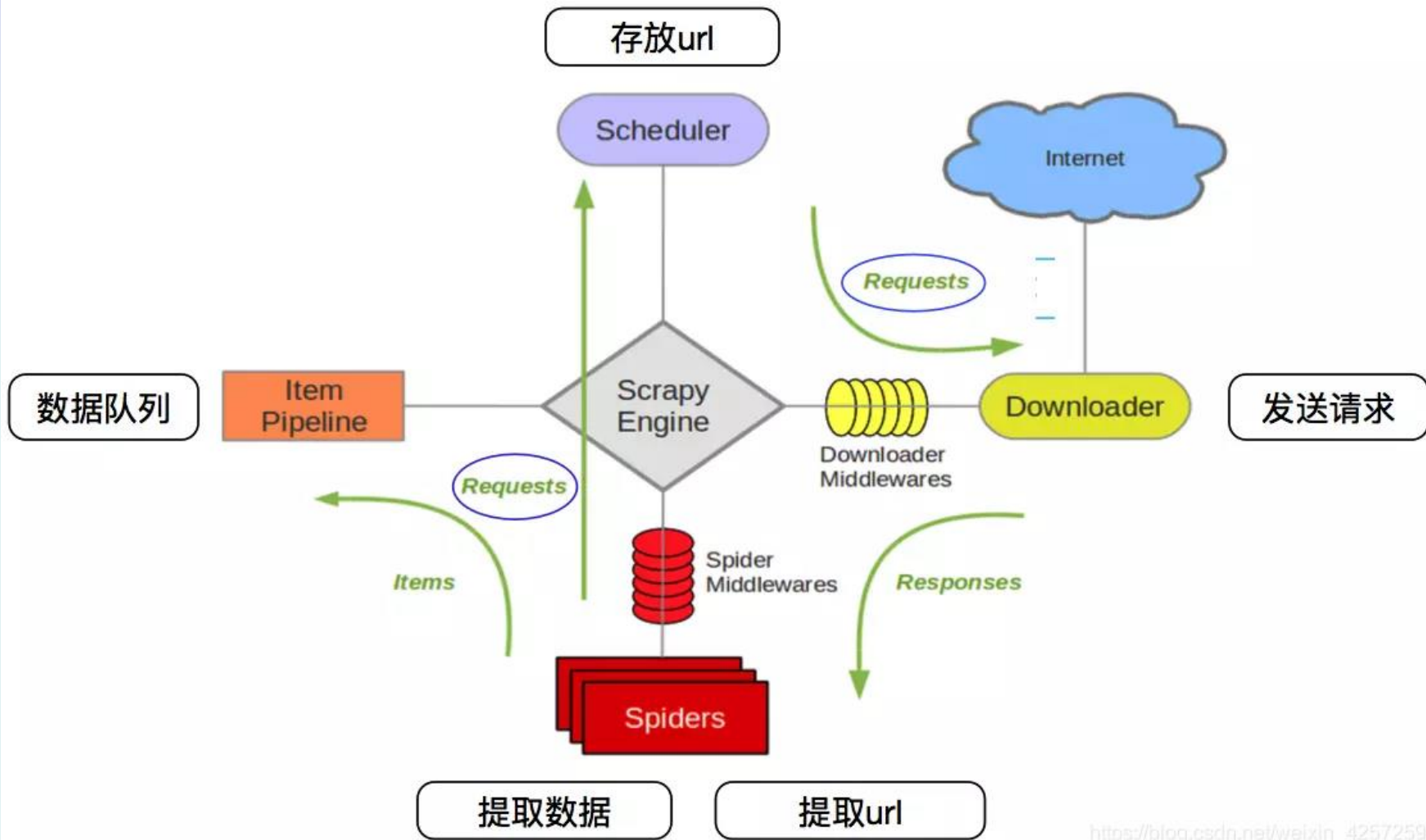


7.1 Scrapy框架简介

- Scrapy是Python开发的一个快速、高层次的web抓取框架，用于抓取web站点并从页面中提取结构化的数据。
- Scrapy广泛用于数据挖掘、监测和自动化测试。
- Scrapy吸引人的地方在于它是一个框架，任何人都可以根据需求方便的修改。它也提供了多种类型爬虫的基类，如BaseSpider、sitemap爬虫。



7.1 Scrapy框架简介



https://blog.csdn.net/weixin_42572590



7.2 安装Scrapy

- 安装请注意版本Python版本!!!

- 步骤概览:

1、网络安装wheel: `pip install wheel`

2、本地安装lxml; lxml下载地址

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml>

3、本地安装pyWin32; pyWin32下载地址

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#pywin32>

4、本地安装Twisted; Twisted下载地址

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#twisted>

5、网络安装Scrapy: `pip install scrapy`



7.2 安装Scrapy

- 本地安装示例: `pip install Twisted-20.3.0-cp38-cp38-win_amd64.whl`

```
C:\Users\ruanz\Desktop>pip install Twisted-20.3.0-cp38-cp38-win_amd64.whl
Processing c:\users\ruanz\desktop\twisted-20.3.0-cp38-cp38-win_amd64.whl
Requirement already satisfied: attrs>=19.2.0 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (19.3.0)
Requirement already satisfied: zope.interface>=4.4.2 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (5.2.0)
Requirement already satisfied: PyHamcrest!=1.10.0, >=1.9.0 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (2.0.2)
Requirement already satisfied: incremental>=16.10.1 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (17.5.0)
Requirement already satisfied: hyperlink>=17.1.1 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (20.0.1)
Requirement already satisfied: Automat>=0.3.0 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (20.2.0)
Requirement already satisfied: constantly>=15.1 in c:\program files\python38\lib\site-packages (from Twisted==20.3.0) (15.1.0)
Requirement already satisfied: setuptools in c:\program files\python38\lib\site-packages (from zope.interface>=4.4.2->Twisted==20.3.0) (49.1.0)
Requirement already satisfied: idna>=2.5 in c:\users\ruanz\appdata\roaming\python\python38\site-packages (from hyperlink>=17.1.1->Twisted==20.3.0) (2.10)
Requirement already satisfied: six in c:\program files\python38\lib\site-packages (from Automat>=0.3.0->Twisted==20.3.0) (1.15.0)
Installing collected packages: Twisted
```



7.2 安装Scrapy

注意：Twisted-20.3.0-cp38-cp38-win_amd64.whl对应的python版本是64位的；如果安装的是32位python，则使用Twisted-19.2.0-cp37-cp37m-win32.whl包安装。

不要在线安装，否则出错！

管理员: 命令提示符

```
C:\Windows\system32>pip3 install C:\Users\ruanz\Desktop\Twisted-20.3.0-cp38-cp38-win_amd64.whl
ERROR: Twisted-20.3.0-cp38-cp38-win_amd64.whl is not a supported wheel on this platform.
```

查看python版本：

管理员: 命令提示符

```
D:\教学资料\大数据采集与清洗>python
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:37:30) [MSC v.1927 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```

管理员: 命令提示符

```
Microsoft Windows [版本 10.0.18363.1198]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Windows\system32>python
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```




7.2 安装Scrapy

- 如果Scrapy安装成功，就可以在终端里执行scrapy命令了。

管理员: 命令提示符

```
C:\WINDOWS\system32>scrapy
Scrapy 2.4.1 - no active project
```

Usage:

```
scrapy <command> [options] [args]
```

Available commands:

bench	Run quick benchmark test
commands	
fetch	Fetch a URL using the Scrapy downloader
genspider	Generate new spider using pre-defined templates
runspider	Run a self-contained spider (without creating a project)
settings	Get settings values
shell	Interactive scraping console
startproject	Create new project
version	Print Scrapy version
view	Open URL in browser, as seen by Scrapy

[more] More commands available when run from project directory

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



7.2 安装Scrapy

- 本章中将会使用如下几个命令
 - **startproject**: 创建一个新项目。
 - **genspider**: 根据模板生成一个新爬虫。
 - **crawl**: 执行爬虫。
 - **shell**: 启动交互式抓取控制台，展示Scrapy的API。



7.3 创建Scrapy爬虫

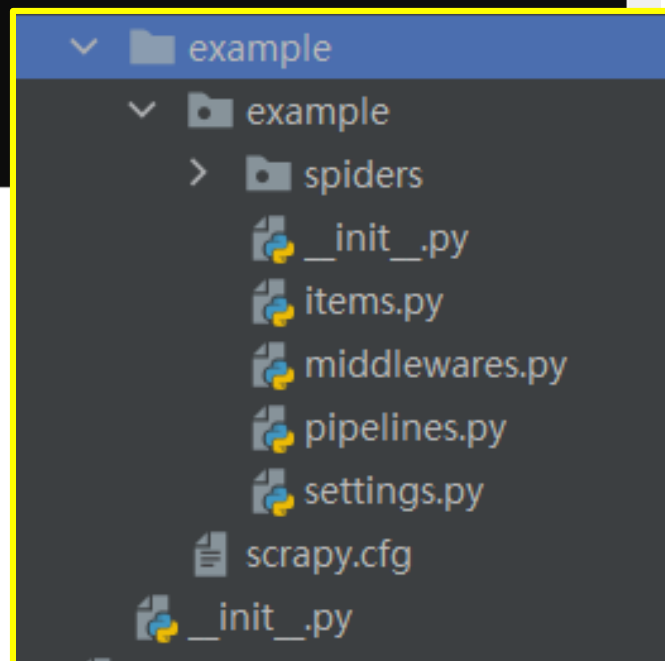
- 运行**startproject**命令生成第一个Scrapy项目的默认结构

scrapy startproject <project name>

```
C:\> 管理员: 命令提示符

D:\PycharmProjects>scrapy startproject example
New Scrapy project 'example', using template directory 'c:\program files
\python38\lib\site-packages\scrapy\templates\project', created in:
    D:\PycharmProjects\example

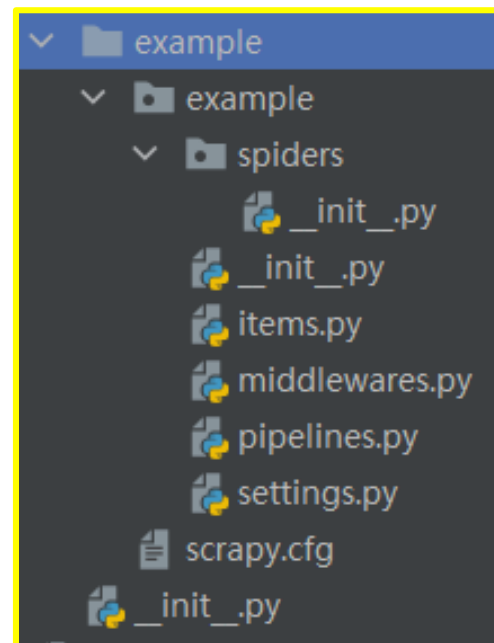
You can start your first spider with:
    cd example
    scrapy genspider example example.com
```





7.3 创建Scrapy爬虫

- 比较重要的几个文件
 - `items.py`: 定义待抓取字段(Field)的模型
 - `settings.py`: 定义一些设置, 如用户代理、爬取延时等。
 - `spiders/`: 用于存储实际的爬虫代码, 即定义感兴趣的链接和抓取数据的选择器。
- Scrapy使用`scrapy.cfg`设置项目配置, `pipelines.py`处理抓取的字段, `middlewares.py`控制请求和响应中间件, 不过本例中无须修改这几个文件。





7.3.1 定义模型

- 默认情况下，example/items.py文件包含如下代码。

```
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 ExampleItem(scrapy.Item):
10     # define the fields for your item here like:
11     # name = scrapy.Field()
12     pass
```

ExampleItem类是一个模板，需要将其中的内容替换为我们希望从示例国家（或地区）页面中抽取到的信息。对于目前来说，我们仅抓取国家（或地区）名称和人口数量，而不是所有信息。



7.3.1 定义模型

- 下面是修改后支持该功能的模型代码。

```
# Define here the models for your scraped items
#
# See documentation in:
# https://docs.scrapy.org/en/latest/topics/items.html

import scrapy

class CountryOrDistrictItem(scrapy.Item):
    # define the fields for your item here like:
    name = scrapy.Field() # 国家名字字段
    population = scrapy.Field() # 国家人口数量字段
```



7.3.2 创建爬虫

- 现在，要开始编写真正的爬虫代码了，在Scrapy里又被称为**spider**。通过**genspider**命令，传入**爬虫名**、**域名**以及可选的**模板参数**，就可以生成初始模板。

scrapy genspider country_or_district
example.python-scraping.com
--template=crawl

注意先进入
example目录

```
D:\PycharmProjects\example>scrapy genspider country_or_district
example.python-scraping.com --template=crawl
Created spider 'country_or_district' using template 'crawl' in m
odule:
example.spiders.country_or_district
```

这里使用了内置的crawl模板，以利用Scrapy库的CrawlSpider。相对于简单的抓取爬虫来说，Scrapy的CrawlSpider拥有一些网络爬取时可用的特殊属性和方法。



7.3.2 创建爬虫

- 运行genspider命令之后，下面的代码将会在example/spiders/country_or_district.py中自动生成

```
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule

class CountryOrDistrictSpider(CrawlSpider):
    name = 'country_or_district'
    allowed_domains = ['example.python-scraping.com']
    start_urls = ['http://example.python-scraping.com/']

    rules = (
        Rule(LinkExtractor(allow=r'Items/'), callback='parse_item', follow=True),
    )

    def parse_item(self, response):
        item = {}
        # item['domain_id'] = response.xpath('//input[@id="sid"]/@value').get()
        # item['name'] = response.xpath('//div[@id="name"]').get()
        # item['description'] = response.xpath('//div[@id="description"]').get()
        return item
```



7.3.2 创建爬虫

- 优化设置

在运行前面生成的爬虫之前，需要更新Scrapy的设置，**避免爬虫被封禁**。默认情况下，Scrapy对同一域名允许最多16个并发下载，并且两次下载之间没有延时，这样就会比真实用户浏览时的速度快很多。该行为很容易被服务器检测到并阻止。



7.3.2 创建爬虫

● 优化设置

- 当下载速度持续高于每秒一个请求时，抓取的示例网站会暂时封禁爬虫，也就是说使用默认配置会造成爬虫被封禁。
- 在example/settings.py文件中添加如下几行代码，使爬虫同时只能对每个域名发起一个请求，并且每两次请求之间存在合理的5秒延时：

CONCURRENT_REQUESTS_PER_DOMAIN = 1
DOWNLOAD_DELAY = 5

```
28 #DOWNLOAD_DELAY = 3
29 # The download delay setting will honor only one of:
30 #CONCURRENT_REQUESTS_PER_DOMAIN = 16
31 #CONCURRENT_REQUESTS_PER_IP = 16
32
33 CONCURRENT_REQUESTS_PER_DOMAIN = 1
34 DOWNLOAD_DELAY = 5
```

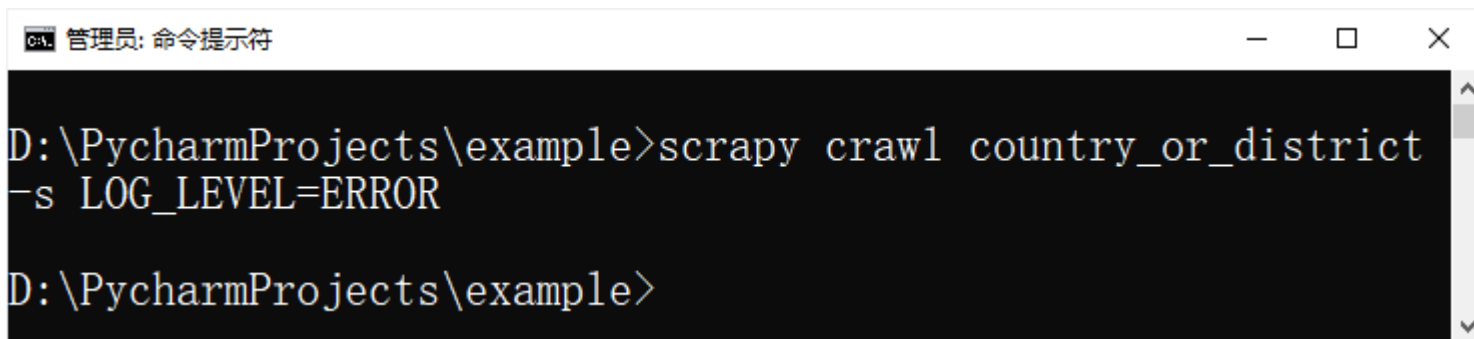


7.3.2 创建爬虫

● 测试爬虫

- 想要从命令行运行爬虫，需要使用**crawl命令**，并且带上爬虫的名称。

scrapy crawl country_or_district -s LOG_LEVEL=ERROR



```
管理员: 命令提示符
D:\PycharmProjects\example>scrapy crawl country_or_district
-s LOG_LEVEL=ERROR
D:\PycharmProjects\example>
```

脚本运行后，完全没有输出。命令中有一个**-s LOG_LEVEL=ERROR**标记，这是一个Scrapy设置，等同于在settings.py文件中定义**LOG_LEVEL = 'ERROR'**。默认情况下，Scrapy会在终端上输出所有日志信息，而这里是将日志级别提升至只显示错误信息。



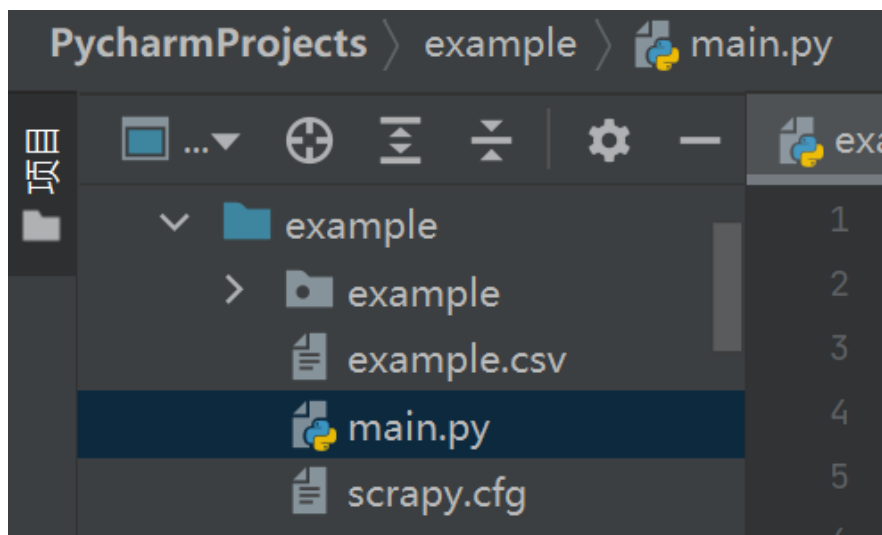
7.3.2 创建爬虫

- 测试爬虫

- 或者用scrapy库的cmdline模块来运行爬虫

`scrapy crawl country_or_district -s LOG_LEVEL=ERROR`

```
# ----- main.py -----  
from scrapy import cmdline  
  
cmd = "scrapy crawl country_or_district -s LOG_LEVEL=ERROR"  
cmdline.execute(cmd.split())
```





7.3.2 创建爬虫

- 为了真正抓取页面上的一些内容，需要在爬虫文件 country_or_district.py 中添加几行代码。为了确保可以启动构建并且抽取 item，必须先从使用 CountryOrDistrictItem 开始，并更新爬取规则。下面是更新后的爬虫版本。

```

.....
from example.items import CountryOrDistrictItem
class CountryOrDistrictSpider(CrawlSpider):
    .....
    rules = (
        Rule(LinkExtractor(allow=r'/index/', deny=r'/user/'), follow=True),
        Rule(LinkExtractor(allow=r'/view/', deny=r'/user/'), callback='parse_item'),
    )

    def parse_item(self, response):
        item = CountryOrDistrictItem()
        name_css = 'tr#places_country_or_district__row td.w2p_fw::text'
        item['name'] = response.css(name_css).extract()
        pop_xpath = '//tr[@id="places_population__row"]/td[@class="w2p_fw"]/text()'
        item['population'] = response.xpath(pop_xpath).extract()
        return item

```




7.3.2 创建爬虫

- 把日志级别设为DEBUG以显示更多的爬取信息，来看一下这个改进后的爬虫是如何运行的：

scrapy crawl country_or_district -s LOG_LEVEL=DEBUG

```
管理员: 命令提示符
D:\PycharmProjects\example>scrapy crawl country_or_district -s LOG_L
EVEL=DEBUG
2020-12-14 21:18:21 [scrapy.utils.log] INFO: Scrapy 2.4.1 started (b
ot: example)
2020-12-14 21:18:21 [scrapy.utils.log] INFO: Versions: lxml 4.6.2.0,
libxml2 2.9.5, cssselect 1.1.0, parsel 1.6.0, w3lib 1.22.0, Twisted
20.3.0, Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [
MSC v.1924 64 bit (AMD64)], pyOpenSSL 20.0.0 (OpenSSL 1.1.1i  8 Dec
2020), cryptography 3.3.1, Platform Windows-10-10.0.18362-SP0
2020-12-14 21:18:21 [scrapy.utils.log] DEBUG: Using reactor: twisted
.internet.selectreactor.SelectReactor
2020-12-14 21:18:21 [scrapy.crawler] INFO: Overridden settings:
{'BOT_NAME': 'example',
 'CONCURRENT_REQUESTS_PER_DOMAIN': 1,
 'DOWNLOAD_DELAY': 5,
 'NEWSPIDER_MODULE': 'example.spiders',
```



7.3.2 创建爬虫

```
C:\> 选择管理员: 命令提示符

2020-12-14 21:20:18 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.python-scraping.com/places/default/view/Bahrain-18> (referer: http://example.python-scraping.com/places/default/index/1)
2020-12-14 21:20:18 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://example.python-scraping.com/places/default/view/Bahrain-18>
{'name': ['Bahrain'], 'population': ['738,004']}
2020-12-14 21:20:22 [scrapy.extensions.logstats] INFO: Crawled 16 pages (at 6 pages/min), scraped 13 items (at 5 items/min)
2020-12-14 21:20:26 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.python-scraping.com/places/default/view/Bahamas-17> (referer: http://example.python-scraping.com/places/default/index/1)
2020-12-14 21:20:26 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://example.python-scraping.com/places/default/view/Bahamas-17>
{'name': ['Bahamas'], 'population': ['301,790']}
2020-12-14 21:20:30 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.python-scraping.com/places/default/view/Azerbaijan-16>
```

输出的日志信息显示，索引页和国家（或地区）页都可以正确爬取



7.3.3 保存爬取结果

- 要想保存结果，可以定义管道（Pipeline），或在settings.py文件中配置输出设置。不过，Scrapy还提供了一个更方便的**--output选项**，用于自动保存已抓取的条目，其可选格式包括CSV、JSON和XML。
- 下面是该爬虫的最终版运行时的结果，它将会输出到一个CSV文件中，此外该爬虫的日志级别被设定为INFO以过滤不重要的信息。

```
scrapy crawl country_or_district
```

```
--output=../../data/scrapy_countries_or_districts.csv
```

```
-s LOG_LEVEL=INFO
```



管理员: 命令提示符

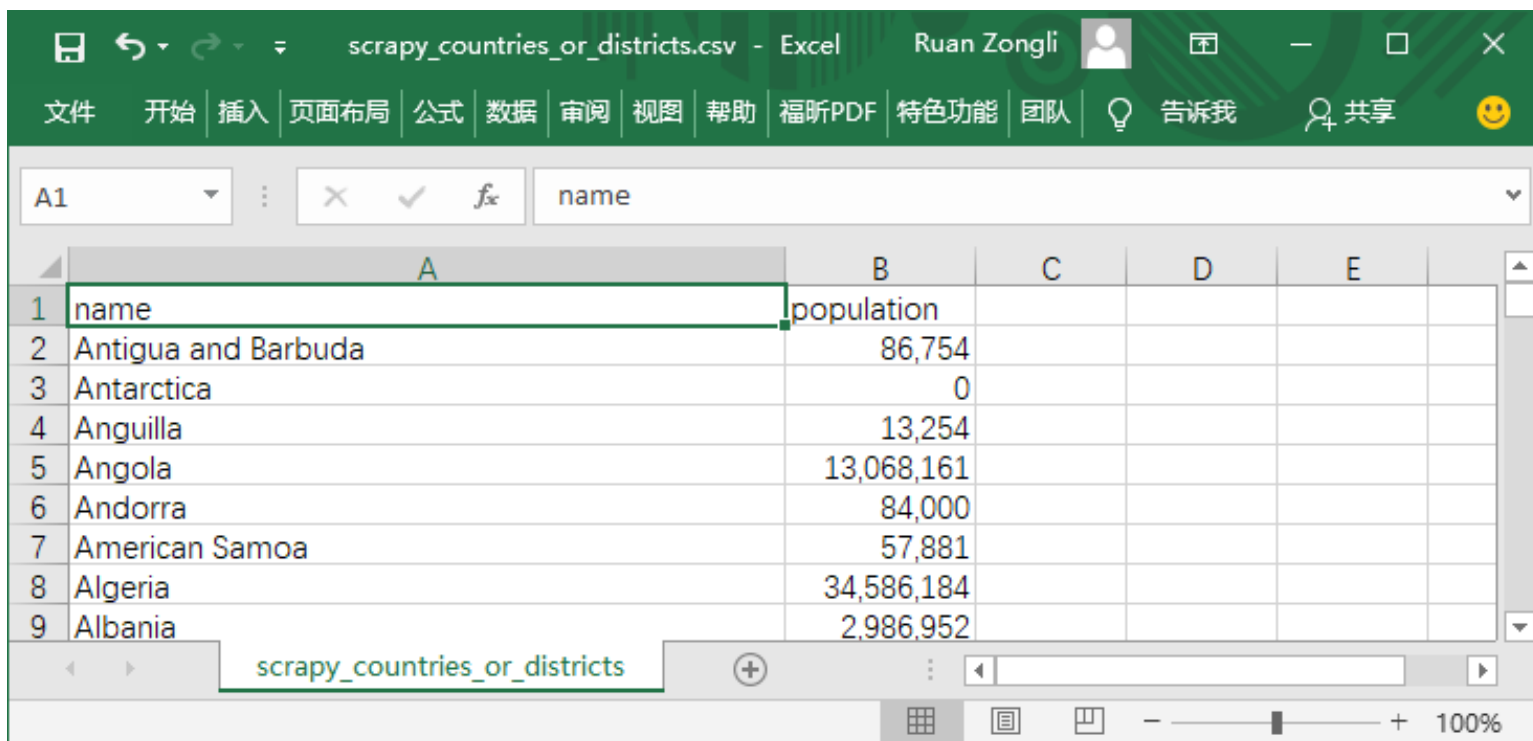
```
2020-12-14 22:20:46 [scrapy.core.engine] INFO: Closing spider (finished)
2020-12-14 22:20:46 [scrapy.extensions.feedexport] INFO: Stored csv feed (246
items) in: ../../../../data/scrapy_countries_or_districts.csv
2020-12-14 22:20:46 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 193479,
'downloader/request_count': 273,
'downloader/request_method_count/GET': 273,
'downloader/response_bytes': 2880417,
'downloader/response_count': 273,
'downloader/response_status_count/200': 273,
'dupefilter/filtered': 59,
'elapsed_time_seconds': 1835.795718,
'finish_reason': 'finished',
'finish_time': datetime.datetime(2020, 12, 14, 14, 20, 46, 425535),
'item_scraped_count': 246,
'log_count/INFO': 41,
'request_depth_max': 25,
'response_received_count': 273,
'robotstxt/request_count': 1,
'robotstxt/response_count': 1,
'robotstxt/response_status_count/200': 1,
'scheduler/dequeued': 272,
'scheduler/dequeued/memory': 272,
'scheduler/enqueued': 272,
'scheduler/enqueued/memory': 272,
'start_time': datetime.datetime(2020, 12, 14, 13, 50, 10, 629817)}
2020-12-14 22:20:46 [scrapy.core.engine] INFO: Spider closed (finished)
```

索引页数量: 25 (0~24)
详情页数量: 246
总链接数量: 273



7.3.3 保存爬取结果

- 在爬取过程的最后阶段，Scrapy会输出一些统计信息，给出爬虫运行的一些指标。从统计结果中，可以了解到爬虫总共爬取了273个网页，并抓取到其中的246个条目，这与数据库中的国家（或地区）数量一致，因此爬虫已经找到了所有的国家（或地区）数据。



	A	B	C	D	E
1	name	population			
2	Antigua and Barbuda	86,754			
3	Antarctica	0			
4	Anguilla	13,254			
5	Angola	13,068,161			
6	Andorra	84,000			
7	American Samoa	57,881			
8	Algeria	34,586,184			
9	Albania	2,986,952			



7.3.3 保存爬取结果

● 通过管道 (Pipeline) 保存结果

- 在 `pipelines.py` 文件中定义管道
- 在 `settings.py` 中启用管道选项

```
from csv import DictWriter
from example.items import CountryOrDistrictItem

class ExamplePipeline:
    def open_spider(self, spider):
        """当爬虫打开后会调用这个函数"""
        filename = 'example.csv'
        spider.logger.info(f'启动管道, 准备将数据写入文件{filename}')
        self.file = open(filename, 'wt', newline='')
        self.dictWriter = DictWriter(self.file, fieldnames=CountryOrDistrictItem.fields.keys())
        self.dictWriter.writeheader() # 写入表头

    def process_item(self, item, spider):
        """当spiders执行 yield 语句 或 return 返回数据时, 就会执行该方法"""
        spider.logger.info(f'保存记录至文件{self.file.name}')
        # 保存记录到csv文件: 利用csv.DictWriter对象写入
        self.dictWriter.writerow(item) # 写入一行(单个字典)
        self.file.flush() # (不等缓存后满就) 立即将缓存写入文件

    def close_spider(self, spider):
        """爬虫完成后会调用该方法"""
        spider.logger.info(f'文件{self.file.name}写入完成')
        self.file.close()
```




7.3.3 保存爬取结果

- 通过管道 (Pipeline) 保存结果
 - 在pipelines.py文件中定义管道
 - 在settings.py中启用管道选项

```
# Configure item pipelines
# See https://docs.scrapy.org/en/latest/topics/item-pipeline.html

# 把ITEM_PIPELINES的注释取消

ITEM_PIPELINES = {
    'example.pipelines.ExamplePipeline': 300,
}
```



7.3.3 保存爬取结果

- 通过管道 (Pipeline) 保存结果
 - 在pipelines.py文件中定义管道
 - 在settings.py中启用管道选项

```
2021-12-14 19:27:40 [scrapy.utils.log] INFO: Scrapy 2.5.1 started (bot: example)
...
2021-12-14 19:27:55 [scrapy.core.engine] DEBUG: Crawled (200) <GET
http://example.python-scraping.com/places/default/view/Antigua-and-Barbuda-10>
(referer: http://example.python-scraping.com/)
2021-12-14 19:27:56 [country_or_district] INFO: 保存记录至文件example.csv
2021-12-14 19:27:56 [scrapy.core.scrapers] DEBUG: Scraped from <200
http://example.python-scraping.com/places/default/view/Antigua-and-Barbuda-10>
...
2021-12-14 19:58:22 [scrapy.core.engine] DEBUG: Crawled (200) <GET
http://example.python-scraping.com/places/default/view/Vietnam-241> (referer:
http://example.python-scraping.com/places/default/index/24)
2021-12-14 19:58:22 [country_or_district] INFO: 保存记录至文件example.csv
2021-12-14 19:58:22 [scrapy.core.scrapers] DEBUG: Scraped from <200
http://example.python-scraping.com/places/default/view/Vietnam-241>
...
2021-12-14 19:58:22 [scrapy.core.engine] INFO: Spider closed (finished)
文件example.csv写入完成
```



7.3.4 不同的爬虫类型

- 在这个Scrapy的例子中，使用了Scrapy的CrawlSpider，它在爬取一个或一系列网站时非常有用。Scrapy还有其他几种爬虫，根据网站和想要抽取的内容不同来选用：
 - Spider：普通的抓取爬虫。通常只用于抓取一个类型的页面。
 - CrawlSpider：爬取爬虫。通常用于遍历域名，并从它通过爬取链接发现的页面中抓取一个（或几个）类型的页面。
 - XMLFeedSpider：遍历XML流并从每个节点中抽取内容的爬虫。
 - CSVFeedSpider：与XML爬虫类似，但此处是解析输出中的CSV行
 - SitemapSpider：通过先解析站点地图，使用不同的规则爬取网站。
- 这些爬虫都包含在Scrapy的默认安装当中，想要构建一个新的网络爬虫时，都可以使用。



7.4 使用shell命令抓取

- 为了帮助测试如何从网页中抽取数据，Scrapy提供shell命令，可以通过Python解释器展示Scrapy的API。
- 命令行启动shell，首先传入一个url：

scrapy shell http://example.python-scraping.com
/view/United-Kingdom-239

```
命令提示符 - scrapy shell http://example.python-scraping.com/view/United-Kingdom-239
-scraping.com/view/United-Kingdom-239> (referer: None)
[s] Available Scrapy objects:
[s] scrapy      scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s] crawler     <scrapy.crawler.Crawler object at 0x0000027D55BC24F0>
[s] item        {}
[s] request     <GET http://example.python-scraping.com/view/United-Kingdom-239>
[s] response    <200 http://example.python-scraping.com/view/United-Kingdom-239>
[s] settings    <scrapy.settings.Settings object at 0x0000027D55BC20D0>
[s] spider      <DefaultSpider 'default' at 0x27d56089610>
[s] Useful shortcuts:
[s] fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[s] fetch(req)                  Fetch a scrapy.Request and update local objects
[s] shelp()                    Shell help (print this help)
[s] view(response)            View response in a browser
>>>
```



7.4 使用shell命令抓取

- 可以查看返回对象，检查哪些数据可以使用

命令提示符 - scrapy shell http://example.python-scraping.com/view/United-Kingdom-239

```
>>> response.url
'http://example.python-scraping.com/view/United-Kingdom-239'
>>> response.status
200
>>>
```

- Scrapy使用lxml抓取数据，选择器可以是css或xpath

命令提示符 - scrapy shell http://example.python-scraping.com/view/United-Kingdom-239

```
>>> selector='tr#places_country_or_district__row td.w2p_fw::text'
>>> response.css(selector).extract()
['Vatican']
>>> selector='//tr[@id="places_population__row"]/td[@class="w2p_fw"]/text()'
>>> response.xpath(selector).extract()
['921']
>>> _
```



7.5 创建Scrapy爬虫爬取石大新闻网

● 主要步骤:

- ① 审查页面，确定选择器需要爬取的数据项及其选择器
- ② 创建Scrapy爬虫项目
- ③ 定义爬虫对象中的字段，表示需要爬取的数据项
- ④ 定义爬虫的爬取规则和抓取（解析）函数
- ⑤ 定义下载中间件，用于爬取动态数据
- ⑥ 定义管道，用于保存爬取数据至文件





7.6 本章小结

- Scrapy是一个流行的网络爬虫框架，可用于创建爬虫。
本章介绍了其如下主要使用步骤：
 - 安装Scrapy；
 - 使用`startproject`命令创建Scrapy爬虫项目；
 - 定义爬取的数据项；
 - 使用`genspider`命令生成爬虫代码框架；
 - 设置爬取并发数量、延时等参数；
 - 修改爬虫代码：定义爬取链接过滤器、元素选择器；
 - 使用`crawl`命令运行爬虫。



谢谢大家！