1. 创建Scrapy项目

- 进入命令行（cmd），切换到要创建项目的目录（如：Desktop）

- 运行scrapy startproject [project\_name]如（scrapy startproject tutorial）

- 进入创建的项目，目录如下：

[project\_name]/

scrapy.cfg

[project\_name]/

\_\_init\_\_.py

items.py

middlewares.py

pipelines.py

settings.py

\_\_pycache\_\_/

spiders/

\_\_init\_\_.py

...(待写入部分)

2. 定义Item容器

- 对需要获取的数据建模（如：资源名（title），资源超链接URL（link），资源描述（desc）,...）

- 在[project\_name]/[project\_name]/items.py中建立相应的字段，注意要给类重新命名为[Name]Item，继承自scrapy.Item

import scrapy

class [Name]Item(scrapy.Item):

title=scrapy.Field()

link=scrapy.Field()

desc=scrapy.Field()

...=scrapy.Field()

3. 爬：获得网页内容

- 在[project\_name]/[project\_name]/spider/目录下创建一个py文件，文件名为上一环节中创建的[name]加上\_spider，即[name]\_spider.py，编写用于爬取数据的类，其包含：

（1）用于下载的初始URL；

（2）跟进网页中的链接和分析网页中的内容的方法；

（3）提取生成item的方法

- 在py文件中创建一个类，类名可设置为[Name]Spider（注意[name]的首字母大写），继承自scrapy.Spider:

- 为这个类设置一个常量name，它是唯一的；一个爬取域名allowed\_domains；在此域名内需要爬取URL的对象start\_urls列表；定义一个解析方法parse()，此方法需要一个参数response(response的获取原理见下一步)

import scrapy

class [Name]Spider(scrapy.Spider):

name=[name]

allowed\_domains=[dmoz-odp.org]

start\_urls=["https://dmoz-odp.org/Computers/Programming/Languages/Python/Books/","https://dmoz-odp.org/Computers/Programming/Languages/Python/Resources/"]

def parse(self,response):

- 在cmd中切换到[project\_name]，运行scrapy crawl [name]，此处的[name]即上一步中设置的类常量name。随后scrapy引擎会进入[name]\_spider.py文件，找到名称为[name]的类[Name]Spider，把该类中的allowed\_domains和start\_urls传输给scheduler,scheduler命令downloader按照二者访问网页，得到供parse方法解析的response.

4.取：处理获得的网页内容，利用Selector选择器

- Selector选择器有四个基本方法：

（1）xpath()：传入xpath表达式，返回该表达式所对应的所有节点的selector list列表；

（2）css()：传入css表达式，返回该表达式所对应的所有节点的selector list列 表；

（3）extract()：序列化该节点为unicode字符并返回list;

(4) re()：根据传入的正则表达式对数据进行提取，返回unicode字符串list列表

- 使用scrapy内置的scrapy shell：cmd进入项目根目录即[project\_name]，运行scrapy shell "[url]"（url如"https://dmoz-odp.org/Computers/Programming/Languages/Python/Books/""），运行exit()可以退出shell。

- cmd起始处图标变为>>>表示进入shell，此shell已存在根据上一步的[url]得出的response，可以用response.headers和response.body方法查看response文档的内容

- scrapy会自动根据response的类型初始化一个sel变量，用于调用xpath()等方法

- 利用selector的xpath()方法（sel.selector.xpath或sel.xpath）获取response文档中的特定内容，xpath表达式形式如下：

（1）xpath('/html/head/title'):选择HTML文档中<head>标签内的<title>元素；

（2）xpath('/html/head/title/text()')：选择上面提到的<title>元素的文字；

（3）xpath('//td')：选择所有的<td>元素

（4）xpath('//div[@class="mine"]')：选择所有具有class="mine"属性的div元素。

（5）xpath('html/head/title/@href')：选择<title>元素中的href属性的内容

注意：在同一个目录下，后一个xpath路径会从前一个xpath路径起始，此时后一个xpath路径的表达式不需要以"/"或"//"起始，如：

sites=sel.xpath('ul[@class="directory-url']/li)

for site in sites:

title=site.xpath('a/text()').extract()

print(title)

- 利用selector的extract()方法(sel.xpath(...).extract())生成方便调用的字符列表

- 根据以上的经验改写上一环节中的[project\_name]\_spider.py文件中的[Name]Spider类中的parse()方法如下：

import scrapy

class [Name]Spider(scrapy.Spider):

name=[name]

allowed\_domains=[dmoz-odp.org]

start\_urls=["https://dmoz-odp.org/Computers/Programming/Languages/Python/Books/","https://dmoz-odp.org/Computers/Programming/Languages/Python/Resources/"]

def parse(self,response):

sel=scrapy.selector.Selector(response)

sites=sel.xpath('//div[@id="site-list-content"]/div/div[@class="title-and-desc"]')

for site in sites:

title=site.xpath('a/div/text()').extract()

link=site.xpath('a/@href').extract()

desc=site.xpath('div/text()')

print(title,link,desc)

- 进入cmd切换到项目根目录[project\_name]/，运行scrapy crawl [name]，可以查看运行[project\_name]\_spider.py的结果

5. 将爬取的数据存放到Item容器中

- 在[project\_name]\_spider.py文件中导入第二环节中定义的Item容器的[Name]Item类，

- 在parse()方法的for循环中初始化一个[Name]Item类的实例对象item，把爬取到的title,link,desc数据以字典的键-值格式传入到item中；

- 在parse()方法中初始化一个items列表，用于存放item；

- 返回items列表

import scrapy

from [project\_name].items import [Name]Item

class [Name]Spider(scrapy.Spider):

name=[name]

allowed\_domains=[dmoz-odp.org]

start\_urls=["https://dmoz-odp.org/Computers/Programming/Languages/Python/Books/","https://dmoz-odp.org/Computers/Programming/Languages/Python/Resources/"]

def parse(self,response):

sel=scrapy.selector.Selector(response)

sites=sel.xpath('//div[@id="site-list-content"]/div/div[@class="title-and-desc"]')

items=[]

for site in sites:

item=[Name]Item()

item[‘title’]=site.xpath('a/div/text()').extract()

item[‘link’]=site.xpath('a/@href').extract()

item[‘desc’]=site.xpath('div/text()')

items.append(item)

return items

- 进入cmd切换到项目根目录[project\_name]/，运行scrapy crawl [name]，可以查看运行[project\_name]\_spider.py的结果

- 导出结果为json格式：运行scrapy crawl [name] -o items.json -t json

-o表示导出，后面空格接导出的文件名（如items.json）

-t表示导出的形式，后面空格接格式名(如 json)

- 打开项目的根文件夹[project\_name]/找到items.json文件并打开，完成爬取。