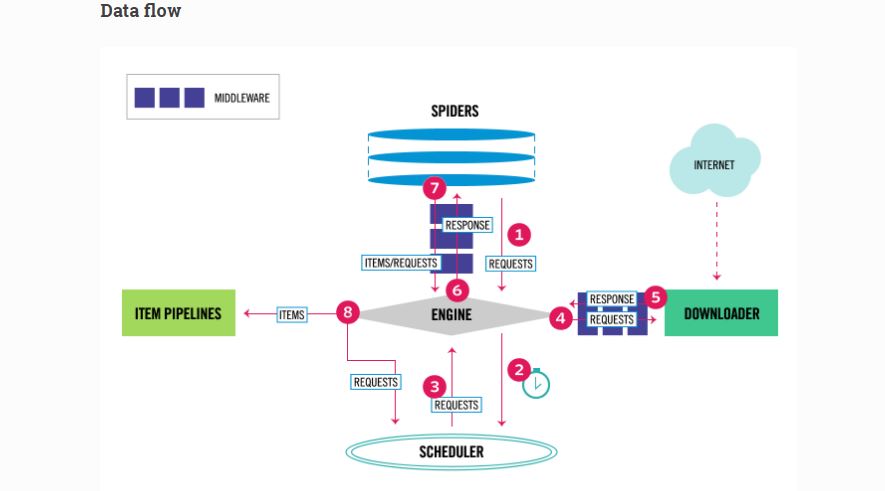
# Scrapy架构



# 主要部件

1. **引擎(Scrapy)**  
   用来处理整个系统的数据流 (框架核心)
2. **调度器(Scheduler)**  
   用来接受引擎发过来的请求, 压入队列中, 并在引擎再次请求的时候返回. 可以想像成一个URL（抓取网页的网址或者说是链接）的优先队列, 由它来决定下一个要抓取的网址是什么, 同时去除重复的网址
3. **下载器(Downloader)**  
   用于下载网页内容, 并将网页内容返回给蜘蛛(Scrapy下载器是建立在twisted这个高效的异步模型上的)
4. **爬虫(Spiders)**  
   爬虫是主要干活的, 用于从特定的网页中提取自己需要的信息, 即所谓的实体(Item)。用户也可以从中提取出链接,让Scrapy继续抓取下一个页面
5. **项目管道(Pipeline)**  
   负责处理爬虫从网页中抽取的实体，主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。当页面被爬虫解析后，将被发送到项目管道，并经过几个特定的次序处理数据。
6. **下载器中间件(Downloader Middlewares)**  
   位于Scrapy引擎和下载器之间的框架，主要是处理Scrapy引擎与下载器之间的请求及响应。
7. **爬虫中间件(Spider Middlewares)**  
   介于Scrapy引擎和爬虫之间的框架，主要工作是处理蜘蛛的响应输入和请求输出。
8. **调度中间件(Scheduler Middewares)**  
   介于Scrapy引擎和调度之间的中间件，从Scrapy引擎发送到调度的请求和响应。

# Scrapy运行流程大概如下



Scrapy中的数据流由执行引擎控制，其过程如下:

1，引擎从Spiders中获取到最初的要爬取的请求（Requests）。

2，引擎安排请求（Requests）到调度器中，并向调度器请求下一个要爬取的请求（Requests）。

3，调度器返回下一个要爬取的请求（Requests）给引擎。

4，引擎将上步中得到的请求（Requests）通过下载器中间件（Downloader Middlewares）发送给下载器（Downloader ），这个过程中下载器中间件（Downloader Middlewares）中的[process\_request()](https://doc.scrapy.org/en/1.3/topics/downloader-middleware.html#scrapy.downloadermiddlewares.DownloaderMiddleware.process_request)函数会被调用到。

5，一旦页面下载完毕，下载器生成一个该页面的Response，并将其通过下载中间件（Downloader Middlewares）发送给引擎，这个过程中下载器中间件（Downloader Middlewares）中的[process\_response()](https://doc.scrapy.org/en/1.3/topics/downloader-middleware.html#scrapy.downloadermiddlewares.DownloaderMiddleware.process_response)函数会被调用到。

6，引擎从下载器中得到上步中的Response并通过Spider中间件(Spider Middlewares)发送给Spider处理，这个过程中Spider中间件(Spider Middlewares)中的[process\_spider\_input()](https://doc.scrapy.org/en/1.3/topics/spider-middleware.html#scrapy.spidermiddlewares.SpiderMiddleware.process_spider_input)函数会被调用到。

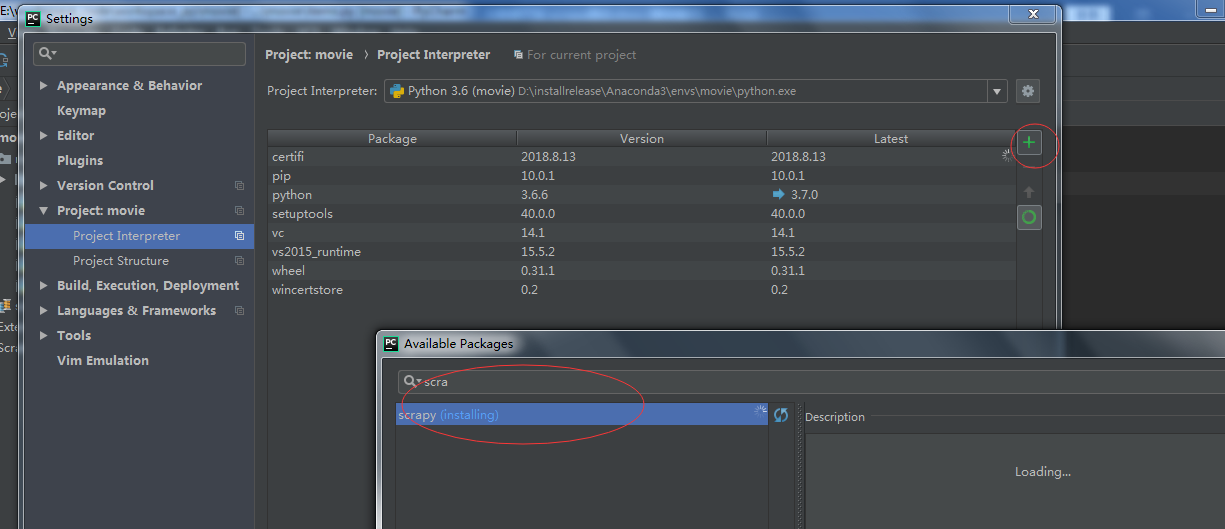
7，Spider处理Response并通过Spider中间件(Spider Middlewares)返回爬取到的Item及(跟进的)新的Request给引擎，这个过程中Spider中间件(Spider Middlewares)的[process\_spider\_output()](https://doc.scrapy.org/en/1.3/topics/spider-middleware.html#scrapy.spidermiddlewares.SpiderMiddleware.process_spider_output)函数会被调用到。

8，引擎将上步中Spider处理的其爬取到的Item给Item 管道（Pipeline），将Spider处理的Request发送给调度器，并向调度器请求可能存在的下一个要爬取的请求（Requests）。

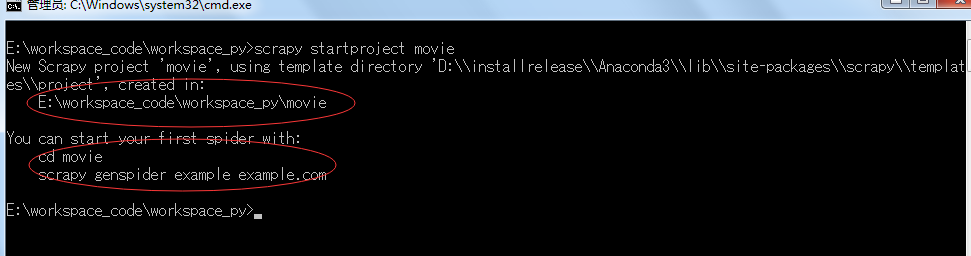
9，(从第二步)重复直到调度器中没有更多的请求（Requests）。

# 安装scrapy

Pip install scrapy等



# 创建项目



E:\workspace\_code\workspace\_py>scrapy startproject movie

New Scrapy project 'movie', using template directory 'D:\\installrelease\\Anaconda3\\lib\\site-packages\\scrapy\\templat

es\\project', created in:

E:\workspace\_code\workspace\_py\movie

# 创建爬虫

You can start your first spider with:

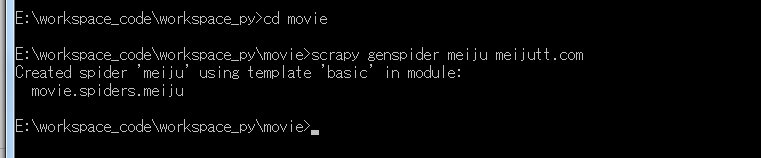
cd movie

genspider 生成爬虫命令

example 爬虫的名字

example.com爬取的网站

scrapy genspider example example.com



E:\workspace\_code\workspace\_py>cd movie

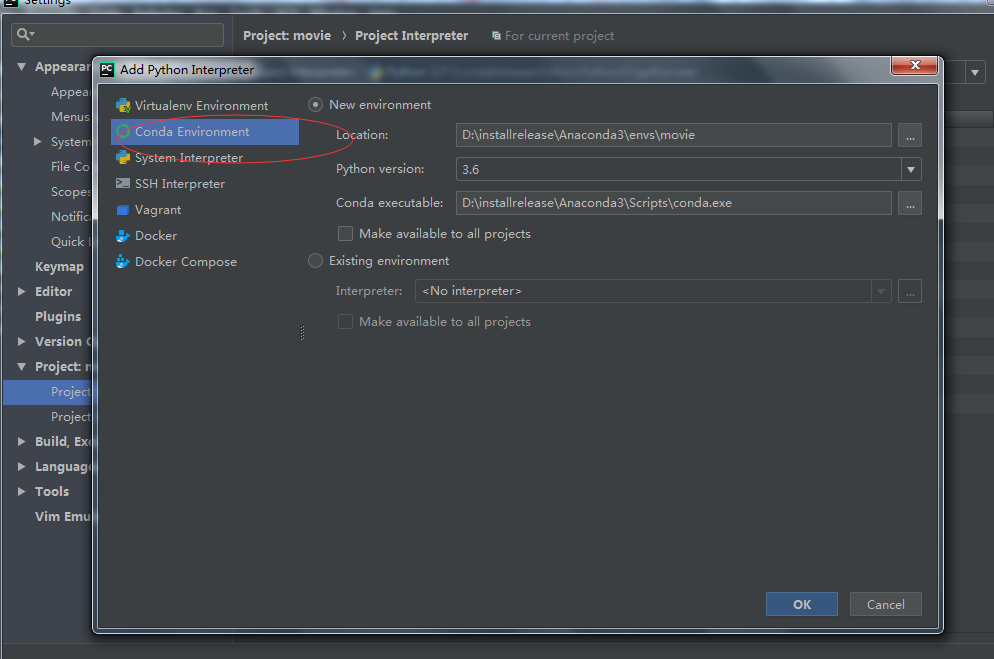
E:\workspace\_code\workspace\_py\movie>scrapy genspider meiju meijutt.com

Created spider 'meiju' using template 'basic' in module:

movie.spiders.meiju

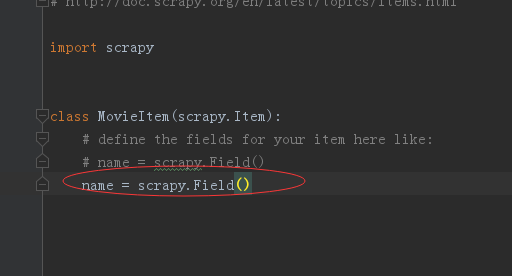
# 打开项目

选择解释器



# 设置数据结构

根据业务，需要取那些内容，然后按OO得思路创建item对象，对象必须继承自scrapy.item



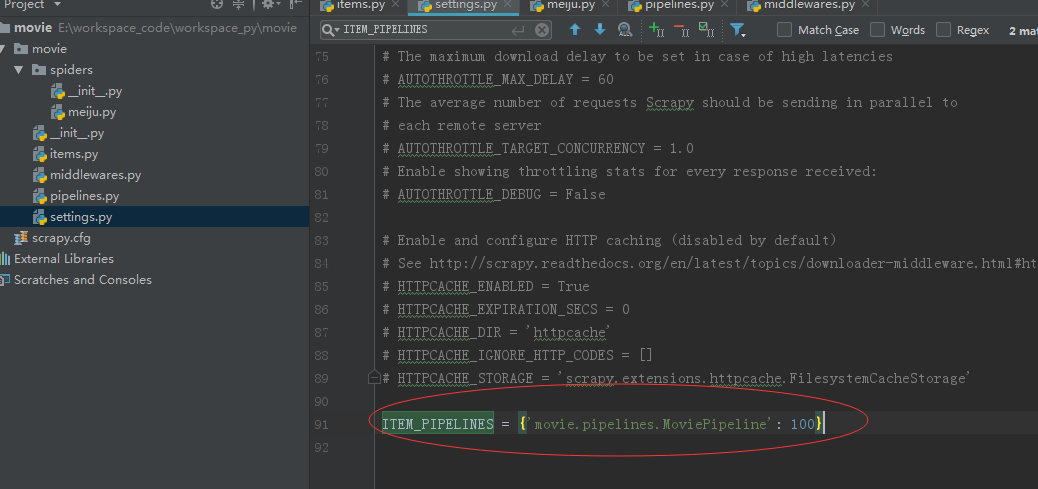
# 编写爬虫



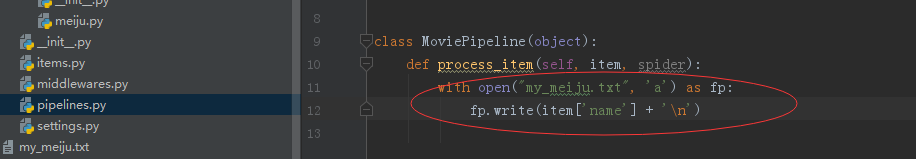
# 配置中间件

这是爬虫中间件：

介于Scrapy引擎和爬虫之间的框架，主要工作是处理蜘蛛的响应输入和请求输出

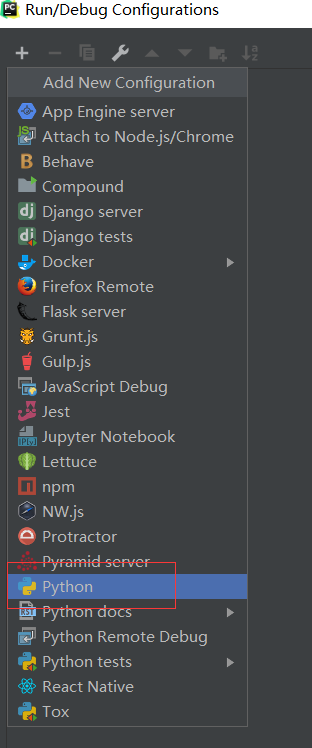


# 数据存储



# 配置参数

新建配置



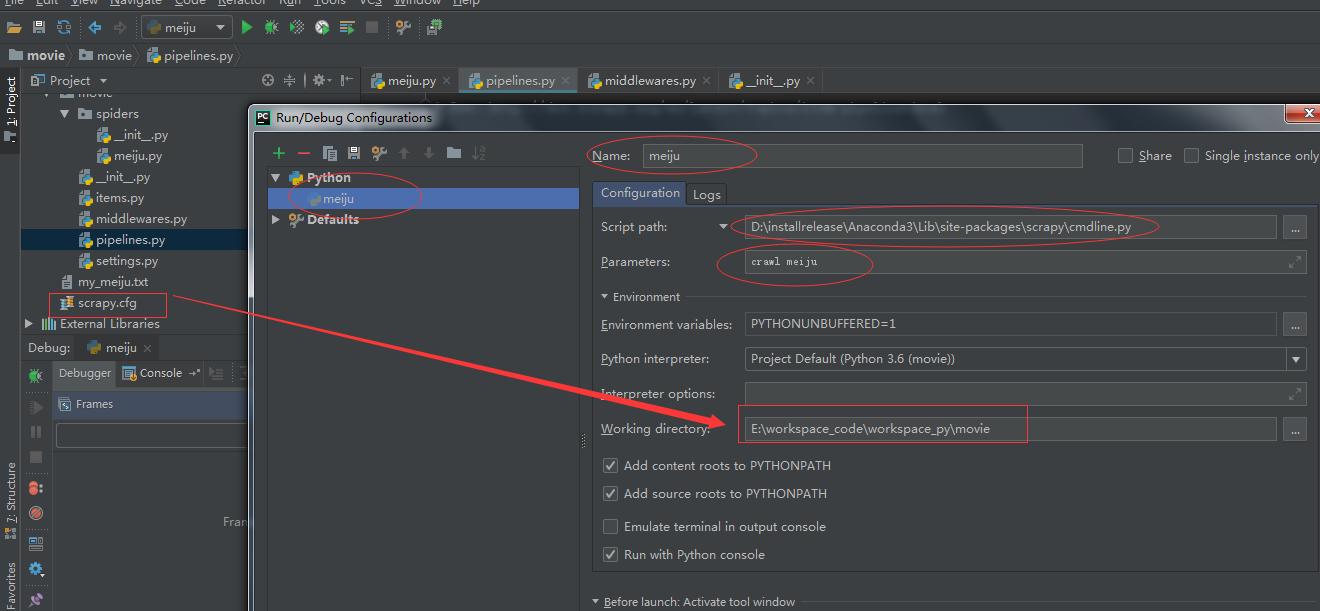
取名字：meiju（在example文件中定义的名字）

要在 pycharm中调试：

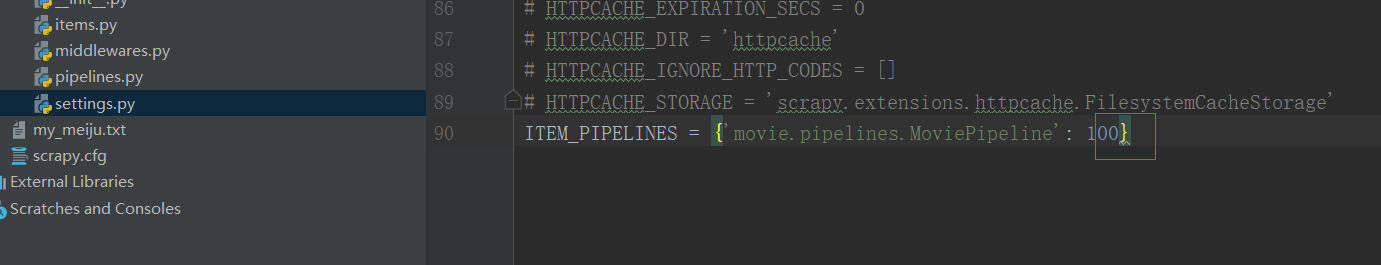
1： 必须设置 script path，这个cmdline.py 在scrapy的安装目录下。若是使用conda安装scrapy，这个就需要到conda下面找，否则到python的Lib\site-packages\scrapy\cmdline.py下面找

2：设置parameters: crawl meiju

3: 设置工作目录：scrapy.cfg 有这个文件的目录



# 配置管道



# 执行程序

方式一：

1. 直接在pycharm执行
2. 单独执行：

scrapy crawl meiju

方式二：

from scrapy.cmdline import execute

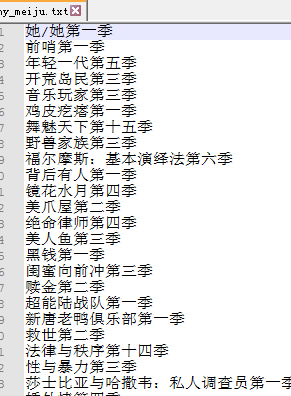
import sys

import os

sys.path.append(os.path.dirname(os.path.abspath(\_\_file\_\_)))

execute(['scrapy', 'crawl', 'spider\_name'])

# 查看结果



# Xpath基本语法

## 语法

|  |  |
| --- | --- |
| **表达式** | **描述** |
| // | 选取节点，使用/表示嵌套 |
| . | 表示当前元素 |
| .. | 表示当前元素的父元素 |
| @ | 通过属性选取节点 |

## 模板

<!DOCTYPE html>

<html lang="en">

<head>

<base href='http://example.com/'/>

<title>网页实例</title>

</head>

<body>

<div id='images'>

<a href='image1.html'>图片 1 <br/><img src='img1.jpg'/></a>

<a href='image2.html'>图片 2 <br/><img src='img2.jpg'/></a>

<a href='image3.html'>图片 3 <br/><img src='img3.jpg'/></a>

<a href='image4.html'>图片 4 <br/><img src='img4.jpg'/></a>

<a href='image5.html'>图片 5 <br/><img src='img5.jpg'/></a>

<div id="images1">

<a href='image1.html'>图片 1 <br/><img src='img1.jpg'/></a>

<a href='image2.html'>图片 2 <br/><img src='img2.jpg'/></a>

<ul>

<li><span class="test1 类属性">test1</span></li>

<li><span class="test2 类属性">test2</span></li>

</ul>

</div>

</div>

</body>

</html>

## 测试

import requests

import scrapy

r = requests.get("http://127.0.0.1:5000/")

hxs = scrapy.Selector(r)

# 【 // 】查找所有a元素

# data = hxs.xpath('//a')

# for i in data:

# print(i)

# # 使用 @+属性名称 查找div这个元素

# data = hxs.xpath('//div[@id="images1"]')

# for i in data:

# print(i)

#

# # 使用 @+属性名称 查找div这个元素

# data = hxs.xpath('//div[@id="images1"]/ul/li')

# for i in data:

# print(i)

# 从当前节点开始遍历子节点

# lis = hxs.xpath('//div[@id="images1"]/ul/li')

# for li in lis:

# cls\_name = li.xpath('./span/@class').extract()[0]

# content = li.xpath('./span/text()').extract()[0]

# print("元素的类名称：%s 具体内容：%s" % (cls\_name, content))

更多内容，参阅：

https://doc.scrapy.org/en/0.14/intro/tutorial.html