爬虫 day9 Scrapy 爬虫

Selector 是 Scrapy的选择器，基于lxml构建，支持xpath, css, 正则表达式匹配

```
scrapy shell https://bbs.hupu.com/gear

result = response.selector.xpath("//a[@class='truetit']/text()")

type(result)

result = response.selector.xpath("//a[@class='truetit']/text()").extract()

result
```

SelectorList 和 Selector都可继续调用xpath()和css() 方法

```
result = response.selector.xpath("//div[contains(@class, 'author')]")
type(result)

result.xpath('./a[1]/text()').extract()
result.xpath('./a[1]/text()').extract_first()
result.xpath('./a[1]/text()')[0]
result.xpath('./a[3]/text()').extract()
result.xpath('./a[3]/text()').extract()[0]
result.xpath('./a[3]/text()').extract_first()
```

extract()[0] 与 extract_first()区别

```
result = response.css('.author.box').xpath('./a[2]/text()').extract()

result = response.css('.endreply.box a::text').extract()
result = response.css('.endreply.box a::attr(href)').extract()

result = response.css('.author.box').xpath('./a[2]/text()').re('(.*?)-.*?-')
```

Spider

1、name：爬虫的名字。
2、allowed_domains：允许爬取的域名，不在此范围的链接不会被跟进爬取。
3、start_urls：起始URL列表，当我们没有重写start_requests()方法时，就会从这个列表开始爬取。
4、custom_settings：用来存放蜘蛛专属配置的字典，这里的设置会覆盖全局的设置。
5、crawler：由from_crawler()方法设置的和蜘蛛对应的Crawler对象，Crawler对象包含了很多项目组件，利用它我们可以获取项目的配置信息，如调用crawler.settings.get()方法。
6、settings：用来获取爬虫全局设置的变量。
7、start_requests()：此方法用于生成初始请求，它返回一个可迭代对象。该方法默认是使用GET请求访问起始URL，如果起始URL需要使用POST请求来访问就必须重写这个方法，发送POST请求使用FormRequest方法
8、parse()：当Response没有指定回调函数时，该方法就会被调用，它负责处理Response对象并返回结果，从中提取出需要的数据和后续的请求，该方法需要返回类型为Request或Item的可迭代对象（生成器当前也包含在其中，因此根据实际需要可以用return或yield来产生返回值）。
9、closed()：当蜘蛛关闭时，该方法会被调用，通常用来做一些释放资源的善后操作。

Downloader Middleware

1、调度器将Request发给Downloader下载之前，可以对Request进行修改
process_request(request, spider)

2、下载后生成的Response发给Spider之前，可以对Response进行修改
process_response(request, response, spider)

3、Downloader或process_request()方法异常
process_exception(request, exception, spider)

Pipeline

Image pipeline

- get_media_requests(self,item, info)：

ImagePipeline根据image_urls中指定的url进行爬取，可以通过get_media_requests为每个

url生成一个Request。如：

```
for image_url in item['image_urls']:
        self.default_headers['referer'] = image_url
        yield Request(image_url, headers=self.default_headers)
```

- item_completed(self, results, item, info)：

图片下载完毕后，处理结果会以二元组的方式返回给item_completed()函数。这个二元组定义如下：
(success, image_info_or_failure)
其中，第一个元素表示图片是否下载成功；第二个元素是一个字典。如：

```
 def item_completed(self, results, item, info):
      image_paths = [x['path'] for ok, x in results if ok]
      if not image_paths:
         raise DropItem("Item contains no images")
      item['image_paths'] = image_paths
      return item
```