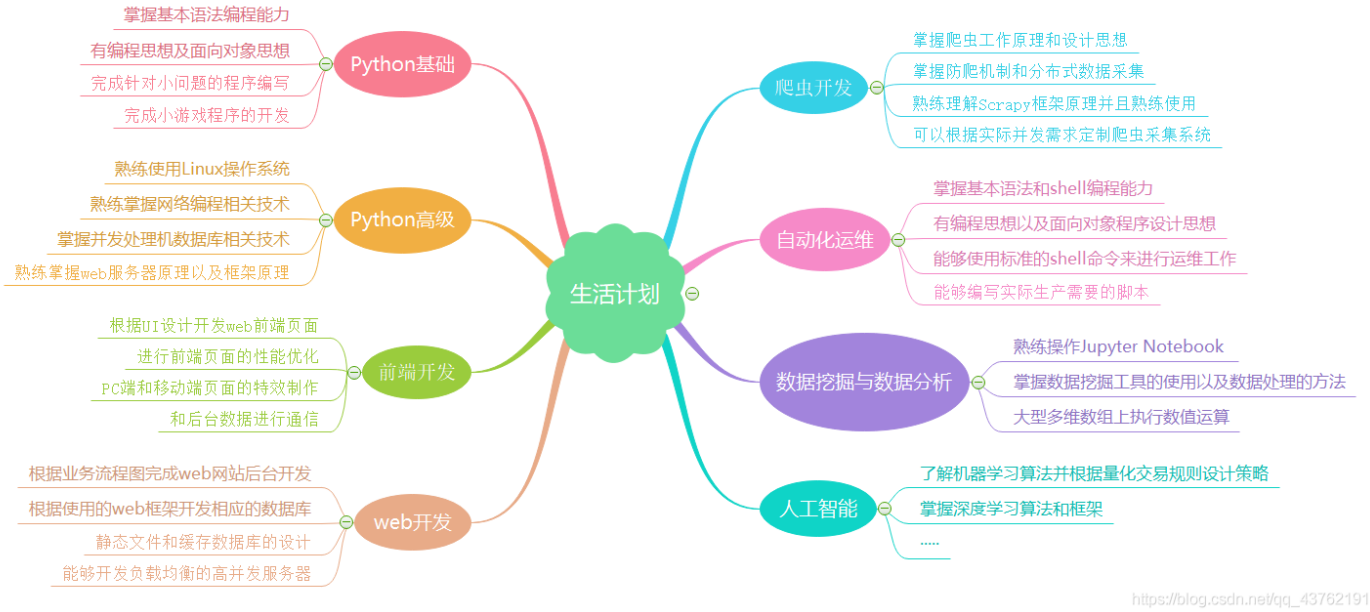


> 标题无意冒犯，就是觉得这个广告挺好玩的
> 上面这张思维导图喜欢就拿走，反正我也学不了这么多

文章目录

前言

前期回顾：我要偷偷学Python（第十二天）

第十二篇的项目还在持续更新中，哎，日理万机虽然谈不上，但是也是手忙脚乱。这不，自动表单生成是写完了，但是学校网络实在是受不了啊，2G，测试不了，就只能一直搁置在那边了。
明天就可以去测试了，明天出学校去逛逛。

今天讲scrapy框架，有不少小伙伴问我说为什么最近没更新，实在是这个不好办呐，安装上就花了些功夫嘞，不过你们跟着我的安装步骤，就不用走那么多的弯路了。

插播一条推送：（如果是小白的话，可以看一下下面这一段）

欢迎来到我们的圈子

我建了一个Python学习答疑群，有兴趣的朋友可以了解一下：这是个什么群

群里已经有一千三百多个小伙伴了哦！！！

直通群的传送门：传送门

本系列文默认各位有一定的C或C++基础，因为我是学了点C++的皮毛之后入手的Python，这里也要感谢齐锋学长送来的支持。
本系列文默认各位会百度，学习‘模块’这个模块的话，还是建议大家有自己的编辑器和编译器的，上一篇已经给大家做了推荐啦？

我要的不多，点个关注就好啦
然后呢，本系列的目录嘛，说实话我个人比较倾向于那两本 Primer Plus，所以就跟着它们的目录结构吧。

本系列也会着重培养各位的自主动手能力，毕竟我不可能把所有知识点都给你讲到，所以自己解决需求的能力就尤为重要，所以我在文中埋得坑请不要把它们看成坑，那是我留给你们的锻炼机会，请各显神通，自行解决。

安装scrapy

这里我并不打算说我安装过程中踩了多少坑，反正你现在跟着我来做：

1、win+R，cmd，打开终端
2、

```
pip install pywin32
pip install pyopenssl
pip install wheel
```

3、打开https://www.lfd.uci.edu/~gohlke/pythonlibs/，找到twisted和lxml两个whl文件，下载下来。

4、进入两个文件的存放目录下，

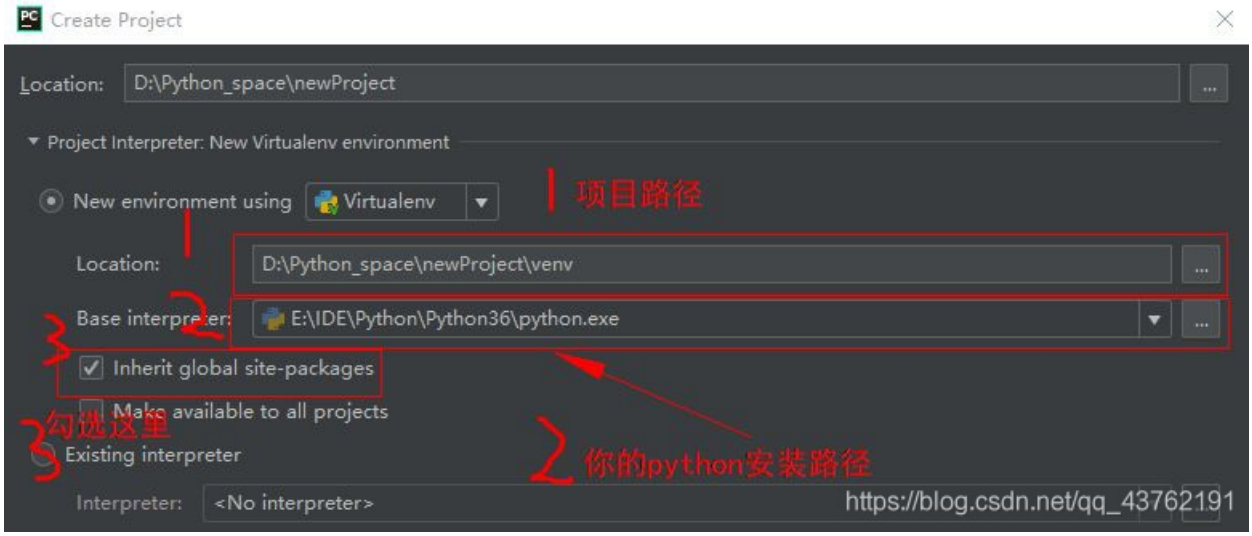
```
pip install Twisted·····
pip install lxml····
```

5、安装scrapy，这里需要引入国内源。

```
pip install Scrapy -i https://pypi.tuna.tsinghua.edu.cn/simple
```

整完之后速度嗖嗖的。

好，当它显示给你说success，就万事大吉了吗？并不是这样的。
这里你需要做两件事情：
1、先弄清楚你安装在了那个Python上，如果你的电脑上只有一个Python就零担别论了，像我的电脑上就有三个Python。
这时候：终端输入：python --version，就可以看到Python的版本号了。
2、在新建文件的时候，



好极，这样就解决了pycharm上有些包无法安装的问题，以及有些包不翻墙解决不了的问题。

好，我们开始今天的主题：Scrapy。

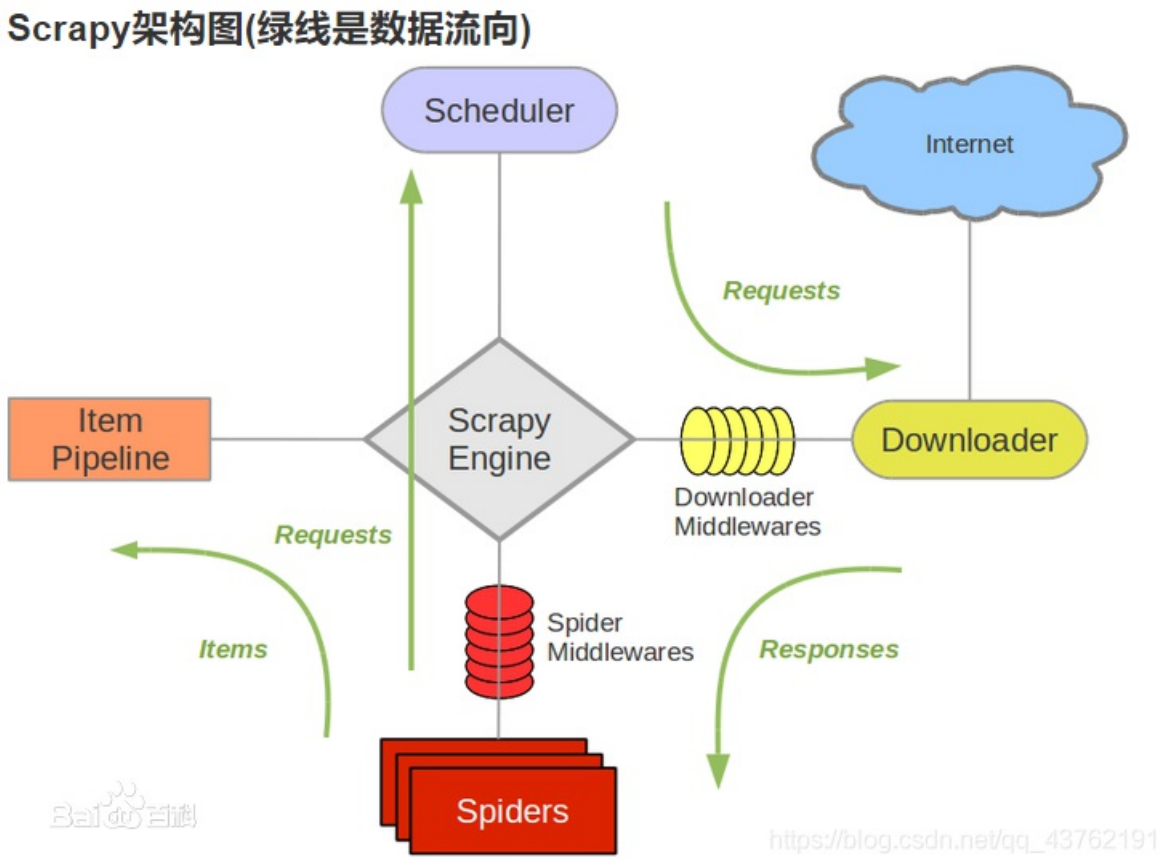
什么是scrapy

Scrapy，Python开发的一个快速、高层次的屏幕抓取和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用途广泛，可以用于数据挖掘、监测和自动化测试.

牛顿说，他只是站在了巨人的肩膀上。说真的，作为一个学C++过来的人，我还没有体验过框架的力量，听说框架可以定制功能，我们只需要将我们所需要的主要功能填充进去，便可以快速的得到我们想要的效果，就像把不同的光碟，插入DVD。

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爬虫等，最新版本又提供了web2.0爬虫的支持。

scrapy架构



这张架构图能看懂不？百度一搜基本全是这张。

```
Scrapy Engine(引擎)：负责Spider、ItemPipeline、Downloader、Scheduler中间的通讯，信号、数据传递等。
Scheduler(调度器)：它负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。
Downloader（下载器）：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。
Spider（爬虫）：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。
Item Pipeline(管道)：它负责处理Spider中获取到的Item，并进行进行后期处理（详细分析、过滤、存储等）的地方。
Downloader Middlewares（下载中间件）：一个可以自定义扩展下载功能的组件。
Spider Middlewares（Spider中间件）：一个可以自定扩展和操作引擎和Spider中间通信的功能组件。
```

注意：当只有调度器中没剩下request的时候，整个项目的运转才会停止。
也就是说，如果某个任务在运行过程中失败了，该网址会被重新访问。

我在网上看到一段挺形象生动的：

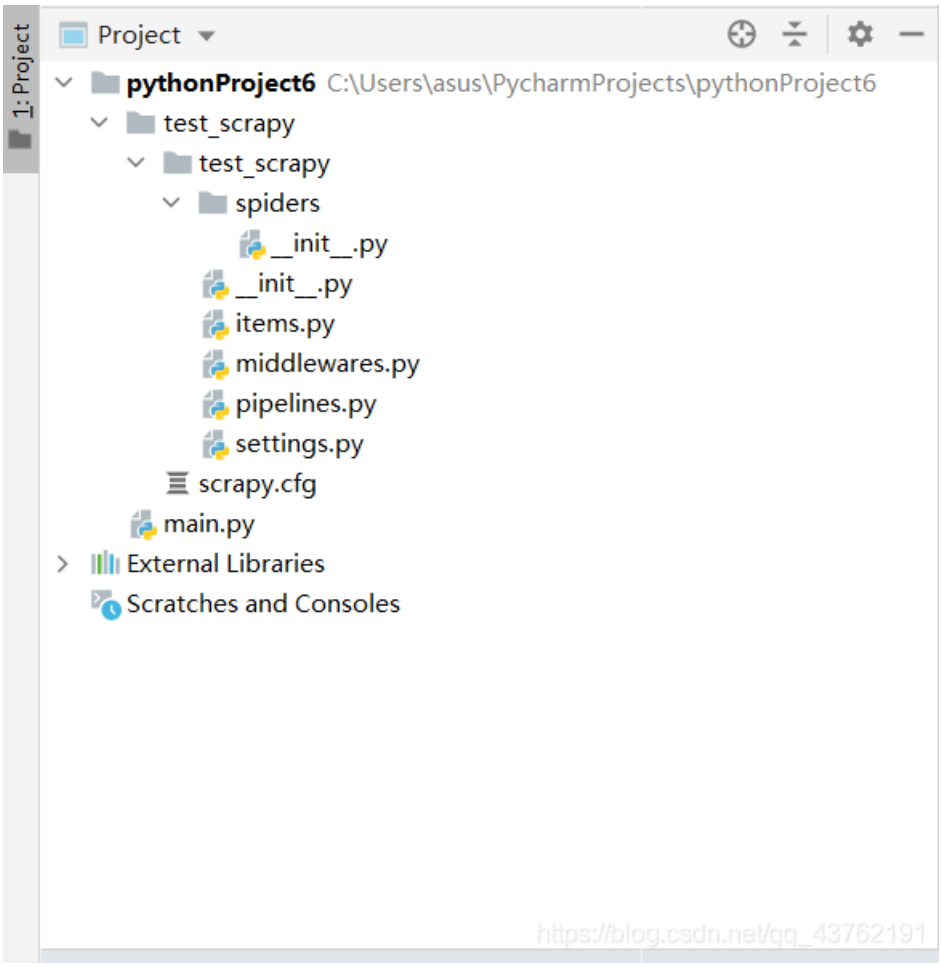
```
! Spider, 你要处理哪一个网站?

2 Spider：老大要我处理xxxx.com。

3 引擎：你把第一个需要处理的URL给我吧。

4 Spider：给你，第一个URL是xxxxxxx.com。

5 引擎：Hi！调度器，我这有request请求你帮我排序入队一下。

6 调度器：好的，正在处理你等一下。

7 引擎：Hi！调度器，把你处理好的request请求给我。

8 调度器：给你，这是我处理好的request

9 引擎：Hi！下载器，你按照老大的下载中间件的设置帮我下载一下这个request请求

10 下载器：好的！给你，这是下载好的东西。（如果失败：sorry，这个request下载失败了。然后引擎告诉调度器，这个request下载失败了，你记录一下，我们待会儿再下载）

11 引擎：Hi！Spider，这是下载好的东西，并且已经按照老大的下载中间件处理过了，你自己处理一下（注意！这儿responses默认是交给def parse()这个函数处理的）

12 Spider：（处理完毕数据之后对于需要跟进的URL），Hi！引擎，我这里有两个结果，这个是我需要跟进的URL，还有这个是我获取到的Item数据。

13 引擎：Hi ！管道 我这儿有个item你帮我处理一下！调度器！这是需要跟进URL你帮我处理下。然后从第四步开始循环，直到获取完老大需要全部信息。

14 管道``调度器：好的，现在就做！
```

---

pycharm体验scrapy项目

天才第一步：创建scrapy项目

由于pycharm无法直接创建scrapy项目，所以这样操作：
1、在pycharm中打开终端，输入scrapy startproject 项目名（比如说：Test_Scrapy）
弄完之后，刷新两下，会出现这么个东西：



说明创建成功了。

下面来简单介绍--下各个主要文件的作用:

```
scrapy.cfg :项目的配置文件
test_scrapy/ :项目的Python模块,将会从这里引用代码
test_scrapy/items.py :项目的目标文件
test_scrapy/pipelines.py :项目的管道文件
test_scrapy/settings.py :目的设置文件
test_scrapy/spiders/ :存储爬虫代码目录
```

天才第二步：明确目标，构建items

我们打算抓取：`http://www.itcast.cn/channel/` 网站里的所有讲师的姓名、职称和个人信息。

```
打开test_scrapy目录下的items.py
Item 定义结构化数据字段，用来保存爬取到的数据，有点像Python中的dict，但是提供了一些额外的保护减少错误。
可以通过创建一个 scrapy.Item 类， 并且定义类型为
scrapy.Field的类属性来定义一个Item（可以理解成类似于ORM的映射关系）。

接下来，创建一个ItcastItem 类，和构建item模型（model）。
```

```
class ItcastItem(scrapy.Item):
 name = scrapy.Field()
    title = scrapy.Field()
    info = scrapy.Field()
```

制作爬虫

进入spiders目录，终端输入命令：

```
scrapy genspider itcast "itcast.cn"
```

将在test_scrapy/test_scrapy/spider目录下创建一个名为itcast的爬虫，并指定爬取域的范围：

打开 spider目录里的 itcast.py，默认增加了下列代码:

```
import scrapy

class ItcastSpider(scrapy.Spider):
    name = "itcast"
    allowed_domains = ["itcast.cn"]
    start_urls = (
        'http://www.itcast.cn/',
    )

    def parse(self, response):
        pass
```

要建立一个Spider， 你必须用scrapy.Spider类创建一个子类，并确定了三个强制的属性 和 一个方法。

`name = ""` ：这个爬虫的识别名称，必须是唯一的，在不同的爬虫必须定义不同的名字。

`allow_domains = []` 是搜索的域名范围，也就是爬虫的约束区域，规定爬虫只爬取这个域名下的网页，不存在的URL会被忽略。

`start_urls = ()` ：爬取的URL元祖/列表。爬虫从这里开始抓取数据，所以，第一次下载的数据将会从这些urls开始。其他子URL将会从这些起始URL中继承性生成。

parse(self, response) ：解析的方法，每个初始URL完成下载后将被调用，调用的时候传入从每一个URL传回的Response对象来作为唯一参数，主要作用如下：

```
负责解析返回的网页数据(response.body)，提取结构化数据(生成item)
生成需要下一页的URL请求。
```

将start_urls的值修改为需要爬取的第一个url

```
start_urls = ("http://www.itcast.cn/channel/teacher.shtml",)
```

修改parse()方法

```
def parse(self, response):
    filename = "teacher.html"
    open(filename, 'w').write(response.body)
```

然后运行一下看看，在test_scrapy/test_scrapy目录下执行：

```
scrapy crawl itcast
```

是的，就是 itcast，看上面代码，它是 ItcastSpider 类的 name 属性，也就是使用 scrapy genspider命令的唯一爬虫名。

运行之后，如果打印的日志出现 [scrapy] INFO: Spider closed (finished)，代表执行完成。 之后当前文件夹中就出现了一个 teacher.html 文件，里面就是我们刚刚要爬取的网页的全部源代码信息。

取数据

爬取整个网页完毕，接下来的就是的取过程了。

我们之前在test_scrapy/test_scrapy/items.py 里定义了一个ItcastItem类。 这里引入进来

```
from test_scrapy/test_scrapy/.items import ItcastItem
```

好，这样写妥妥是要报错的，不这样写又没法子，搞了半天。

```
from ..items import ItcastItem
```

然后将我们得到的数据封装到一个 ItcastItem 对象中，可以保存每个老师的属性：

```
from test_scrapy.items import ItcastItem

def parse(self, response):
    #open("teacher.html","wb").write(response.body).close()

    # 存放老师信息的集合
    items = []

    for each in response.xpath("//div[@class='li_txt']"):
        # 将我们得到的数据封装到一个 `ItcastItem` 对象
        item = ItcastItem()
        #extract()方法返回的都是unicode字符串
        name = each.xpath("h3/text()").extract()
        title = each.xpath("h4/text()").extract()
        info = each.xpath("p/text()").extract()

        #xpath返回的是包含一个元素的列表
        item['name'] = name[0]
        item['title'] = title[0]
        item['info'] = info[0]

        items.append(item)

    # 直接返回最后数据
    return items
```

scrapy保存信息的最简单的方法主要有四种，-o 输出指定格式的文件，这里我采用的是csv文件

csv 逗号表达式，可用Excel打开

```
scrapy crawl itcast -o teachers.csv
```

最近都比较忙了些，先把这个框架走通吧，后面还要拓展再拓展，难搞哦。。。