# python爬虫框架scrapy实例详解

2013-08-14 09:24:47

生成项目scrapy提供一个工具来生成项目，生成的项目中预置了一些文件，用户需要在这些文件中添加自己的代码。打开命令行，执行：scrapy st...

生成项目

scrapy提供一个工具来生成项目，生成的项目中预置了一些文件，用户需要在这些文件中添加自己的代码。

打开命令行，执行：scrapy startproject tutorial，生成的项目类似下面的结构

tutorial/

  scrapy.cfg

  tutorial/

    __init__.py

    items.py

    pipelines.py

    settings.py

    spiders/

      __init__.py

      ...

scrapy.cfg是项目的配置文件

用户自己写的spider要放在spiders目录下面，一个spider类似

```
from scrapy.spider import BaseSpider
class DmozSpider(BaseSpider):
    name = "dmoz"
    allowed_domains = ["dmoz.org"]
    start_urls = [
        "http://www.dmoz.org/Computers/Programming/Languages/Python/Books/",
        "http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/"
    ]
    def parse(self, response):
        filename = response.url.split("/")[-2]
        open(filename, 'wb').write(response.body)
```

name属性很重要，不同spider不能使用相同的name

start_urls是spider抓取网页的起始点，可以包括多个url

parse方法是spider抓到一个网页以后默认调用的callback，避免使用这个名字来定义自己的方法。

当spider拿到url的内容以后，会调用parse方法，并且传递一个response参数给它，response包含了抓到的网页的内容，在parse方法里，你可以从抓到的网页里面解析数据。上面的代码只是简单地把

频道总排行

python下redis安装和使用
ImportError: No module named
python 在互联网应用是如此强大
python读取和生成excel文件
python多线程编程2—线程的创建、
分别用python2和python3伪装浏览器
python批量抓取美女图片
Python简单两步实现天气爬虫采集器
成人网站性能提升 20 倍之经验谈
python抓取google搜索结果

频道本月排行

python多线程编程2—线程的创建、
python爬虫框架scrapy实例详解
python读取和生成excel文件
python批量抓取美女图片
分别用python2和python3伪装浏览器
成人网站性能提升 20 倍之经验谈
一步步来用C语言来写python扩展
用python实现的抓取腾讯视频所有电
用python 装饰器打log
python抓取google搜索结果

最新文章

IP地址网段表示法总结
pyOpenSSL版本问题导致Scrapy安装报
Linux开机启动程序或脚本详解
一步步教你理解Python装饰器
字符串的encode与decode解决乱码问题
Python操作cookie之cookielib模块
作为面试官的一些心得
图像库PIL相关笔试题大总结
为什么你会迁移到Python3.5？
滴滴悄然提价 - 中国打车市场补贴战或
众筹3万起诉广电总局 - 起因游戏审核条
在Python中添加自定义模块

相关文章

python 在互联网应用是如此强大
sys.argv[] 的使用详解
ImportError: No module named
python下redis安装和使用

网页内容保存到文件。

开始抓取

你可以打开命令行，进入生成的项目根目录tutorial/，执行 scrapy crawl dmoz， dmoz是spider的name。

解析网页内容

scrapy提供了方便的办法从网页中解析数据，这需要使用到HtmlXPathSelector

```
from scrapy.spider import BaseSpider
from scrapy.selector import HtmlXPathSelector
class DmozSpider(BaseSpider):
    name = "dmoz"
    allowed_domains = ["dmoz.org"]
    start_urls = [
        "http://www.dmoz.org/Computers/Programming/Languages/Python/Books/",
        "http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/"
    ]
    def parse(self, response):
        hxs = HtmlXPathSelector(response)
        sites = hxs.select('//ul/li')
        for site in sites:
            title = site.select('a/text()').extract()
            link = site.select('a/@href').extract()
            desc = site.select('text()').extract()
            print title, link, desc
```

HtmlXPathSelector使用了Xpath来解析数据

//ul/li表示选择所有的ul标签下的li标签

a/@href表示选择所有a标签的href属性

a/text()表示选择a标签文本

a[@href="abc"]表示选择所有href属性是abc的a标签

我们可以把解析出来的数据保存在一个scrapy可以使用的对象中，然后scrapy可以帮助我们把这些对象保存起来，而不用我们自己把这些数据存到文件中。我们需要在items.py中添加一些类，这些类用来描述我们要保存的数据

from scrapy.item import Item, Field

class DmozItem(Item):

  title = Field()

  link = Field()

  desc = Field()

然后在spider的parse方法中，我们把解析出来的数据保存在DomzItem对象中。

```
from scrapy.spider import BaseSpider
from scrapy.selector import HtmlXPathSelector
from tutorial.items import DmozItem
class DmozSpider(BaseSpider):
   name = "dmoz"
   allowed_domains = ["dmoz.org"]
   start_urls = [
       "http://www.dmoz.org/Computers/Programming/Languages/Python/Books/",
       "http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/"
   ]
   def parse(self, response):
       hxs = HtmlXPathSelector(response)
       sites = hxs.select('//ul/li')
       items = []
       for site in sites:
           item = DmozItem()
           item['title'] = site.select('a/text()').extract()
           item['link'] = site.select('a/@href').extract()
           item['desc'] = site.select('text()').extract()
           items.append(item)
       return items
```

在命令行执行scrapy的时候，我们可以加两个参数，让scrapy把parse方法返回的items输出到json文件中

scrapy crawl dmoz -o items.json -t json

items.json会被放在项目的根目录

python链接mysql数据库详解

让scrapy自动抓取网页上的所有链接

上面的示例中scrapy只抓取了start_urls里面的两个url的内容，但是通常我们想实现的是scrapy自动发现一个网页上的所有链接，然后再去抓取这些链接的内容。为了实现这一点我们可以在parse方法里面提取我们需要的链接，然后构造一些Request对象，并且把他们返回，scrapy会自动的去抓取这些链接。代码类似：

```
class MySpider(BaseSpider):
    name = 'myspider'
    start_urls = (
        'http://example.com/page1',
        'http://example.com/page2',
        )
    def parse(self, response):
        # collect `item_urls`
        for item_url in item_urls:
            yield Request(url=item_url, callback=self.parse_item)
    def parse_item(self, response):
        item = MyItem()
        # populate `item` fields
        yield Request(url=item_details_url, meta={'item': item},
            callback=self.parse_details)
    def parse_details(self, response):
        item = response.meta['item']
        # populate more `item` fields
        return item
```

parse是默认的callback, 它返回了一个Request列表，scrapy自动的根据这个列表抓取网页，每当抓到一个网页，就会调用parse_item，parse_item也会返回一个列表，scrapy又会根据这个列表去抓网页，并且抓到后调用parse_details

为了让这样的工作更容易，scrapy提供了另一个spider基类，利用它我们可以方便的实现自动抓取链接. 我们要用到CrawlSpider

```
from scrapy.contrib.linkextractors.sgml import SgmlLinkExtractor
class MininovaSpider(CrawlSpider):
    name = 'mininova.org'
    allowed_domains = ['mininova.org']
    start_urls = ['http://www.mininova.org/today']
    rules = [Rule(SgmlLinkExtractor(allow=['/tor/\d+'])),
             Rule(SgmlLinkExtractor(allow=['/abc/\d+']), 'parse_torrent')]
    def parse_torrent(self, response):
        x = HtmlXPathSelector(response)
        torrent = TorrentItem()
        torrent['url'] = response.url
        torrent['name'] = x.select("//h1/text()").extract()
        torrent['description'] =
x.select("//div[@id='description']").extract()
        torrent['size'] = x.select("//div[@id='info-left']/p[2]/text()
[2]").extract()
        return torrent
```

相比BaseSpider，新的类多了一个rules属性，这个属性是一个列表，它可以包含多个Rule，每个Rule描述了哪些链接需要抓取，哪些不需要。这是Rule类的文档http://doc.scrapy.org/en/latest/topics/spiders.html#scrapy.contrib.spiders.Rule

这些rule可以有callback，也可以没有，当没有callback的时候，scrapy简单的follow所有这些链接.

pipelines.py的使用

在pipelines.py中我们可以添加一些类来过滤掉我们不想要的item，把item保存到数据库。

```
from scrapy.exceptions import DropItem
class FilterWordsPipeline(object):
    """A pipeline for filtering out items which contain certain words in
their
    description"""
    # put all words in lowercase
    words_to_filter = ['politics', 'religion']
    def process_item(self, item, spider):
        for word in self.words_to_filter:
            if word in unicode(item['description']).lower():
                raise DropItem("Contains forbidden word: %s" % word)
        else:
            return item
```

如果item不符合要求，那么就抛一个异常，这个item不会被输出到json文件中。

要使用pipelines，我们还需要修改settings.py

添加一行

ITEM_PIPELINES = ['dirbot.pipelines.FilterWordsPipeline']

现在执行scrapy crawl dmoz -o items.json -t json，不符合要求的item就被过滤掉了

**相关文章：**

·python 在互联网应用是如此强大
·sys.argv[] 的使用详解
·ImportError: No module named setuptools 解决方法
·python下redis安装和使用
·python链接mysql数据库详解
·python+ mysql存储二进制流的方式

**相关词搜索：** python 爬虫 框架

**上一篇：** Python 如何将一变量做为函数名？

**下一篇：** 解决ImportError: libmysqlclient_r.so.16: cannot open shared object file



如果你遇到什么问题，请留言回复，我们会及时帮助解决...

发布

还没有评论，沙发等你来抢