基于scrapy爬虫的天气数据采集(python)

一、实验介绍

1.1 实验内容

本项目介绍如何用Scrapy来采集天气信息（从新浪天气频道采集），完成本项目后能对Python Scrapy有个初步的认识，能进行简单的数据采集需求开发。

1.2 知识点

本节实验中将学习和实践以下知识点：

1. Python基本语法
2. Scrapy框架
3. 爬虫的概念

1.3 实验环境

- python2.7
- Xfce终端
- scrapy 0.24

1.4 适合人群

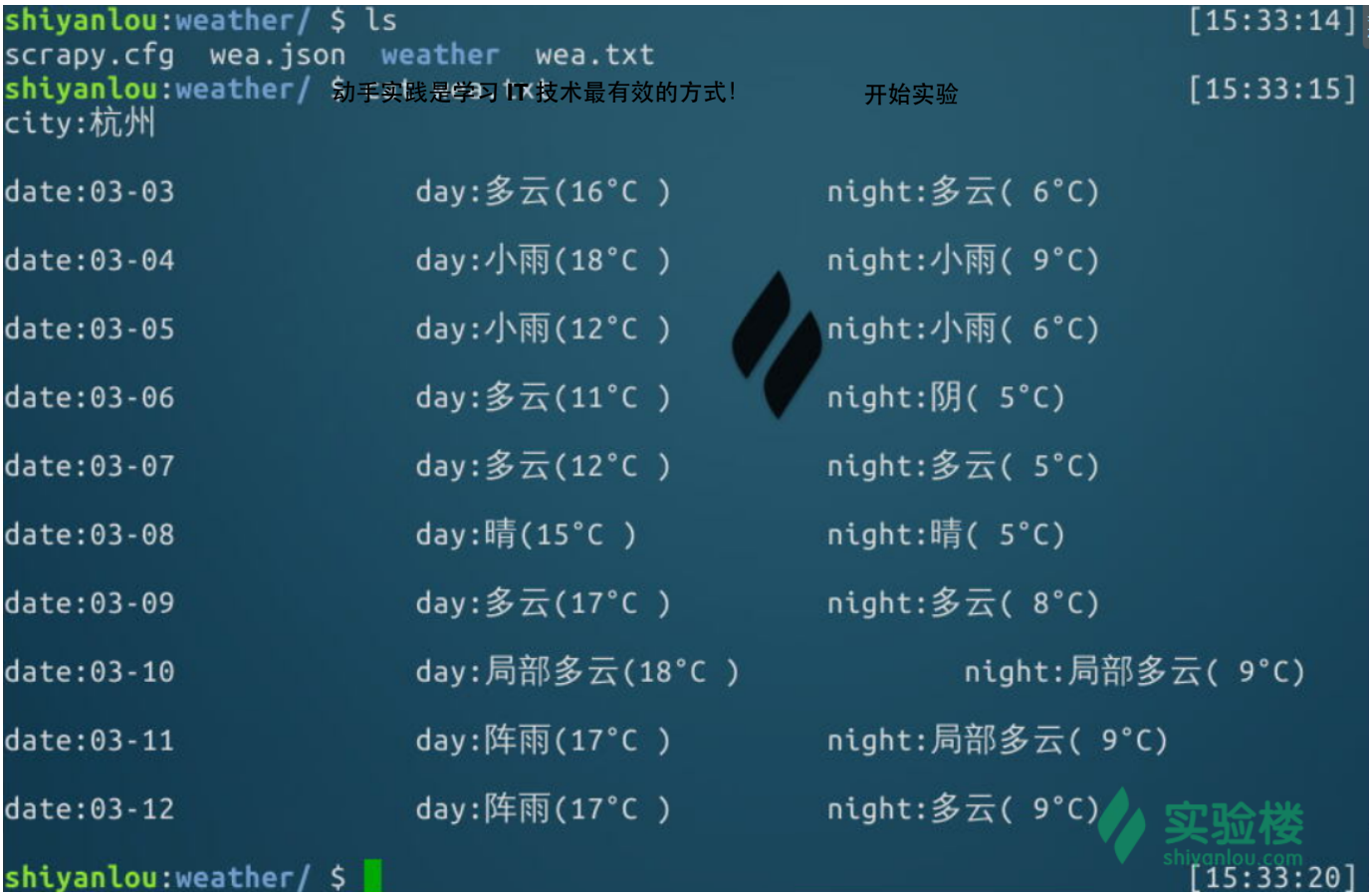
本课程难度为中等，适合具有Python 和 HTML 基础的用户，熟悉python基础知识加深巩固。

1.5 代码获取

你可以通过下面命令将代码下载到实验楼环境中，作为参照对比进行学习。

```
$ wget https://github.com/shiyanlou/scrapy-weather
```

1.6 实验效果



二、开发准备

2.1 安装Scrapy

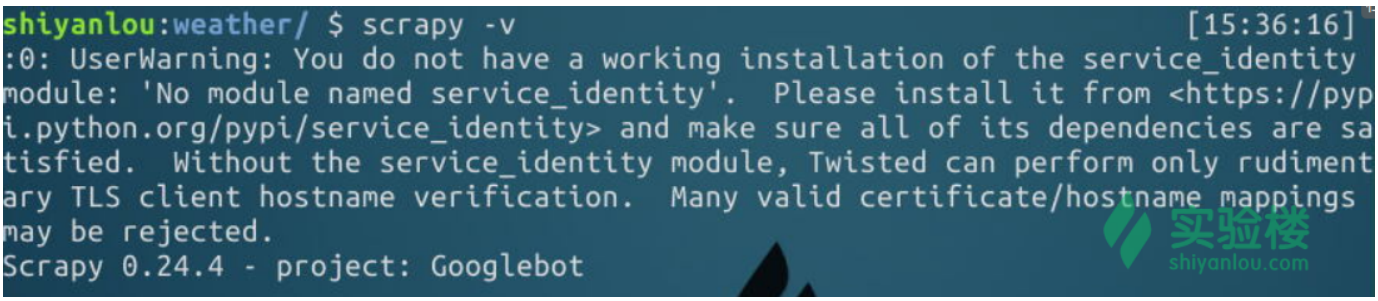
安装 scrapy-0.24：

```
# 安装依赖的包
$ sudo apt-get update
$ sudo apt-get install python-lxml python-dev libffi-dev

# 更新系统默认的 six 包
$ sudo pip install six --upgrade

# 安装指定版本的scrapy
$ sudo pip install scrapy==0.24.4
```

完成这步后，可以用下面的命令测试一下安装是否正确：

```
$ scrapy version
```

如果正常，效果如图所示：



2.2 创建项目

在开始爬取之前，必须创建一个新的Scrapy项目。进入您打算存储代码的目录中，运行下列命令：

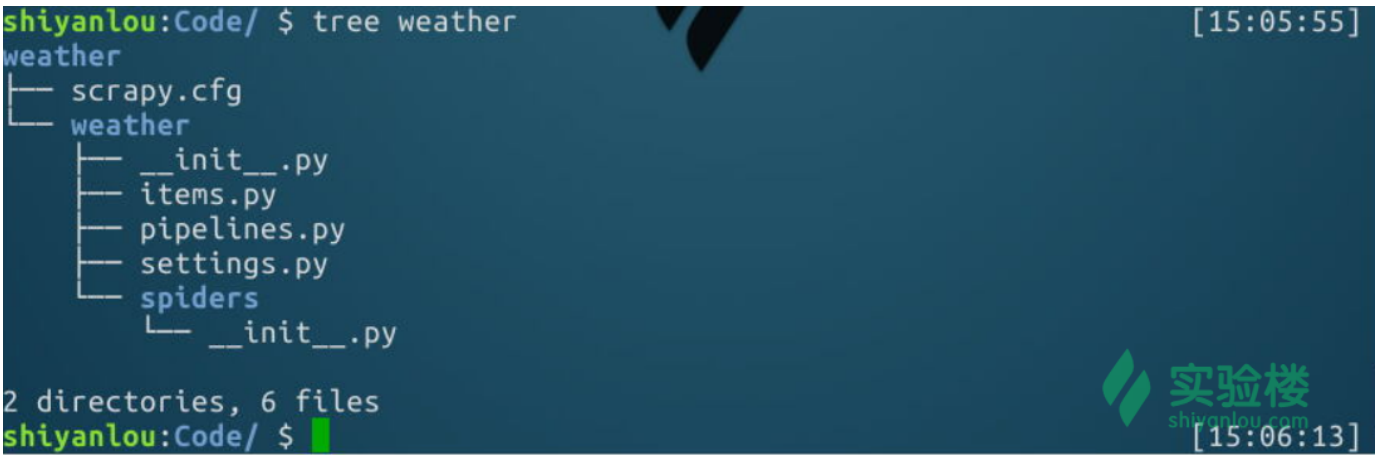
```
$ scrapy startproject weather
```

如果正常，效果如图所示：

这些文件分别是:

- scrapy.cfg: 项目的配置文件
- weather/: 该项目的python模块。之后将在此加入代码。
- weather/items.py: 项目中的item文件.
- weather/pipelines.py: 项目中的pipelines文件.
- weather/settings.py: 项目的设置文件.
- weather/spiders/: 放置spider代码的目录.

三、项目文件结构



四、项目实战

4.1 定义Item

Item 是保存爬取到的数据的容器；其使用方法和python字典类似，并且提供了额外保护机制来避免拼写错误导致的未定义字段错误。

首先根据需要从weather.sina.com.cn获取到的数据对item进行建模。 我们需要从weather.sina.com.cn中获取当前城市名，后续9天的日期，天气描述和温度等信息。对此，在item中定义相应的字段。编辑 weather 目录中的 items.py 文件:


```
# -*- coding: utf-8 -*-

# Define here the models for your scraped items
#
# See documentation in:
# http://doc.scrapy.org/en/latest/topics/items.html

import scrapy


class WeatherItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    # demo 1
    city = scrapy.Field()
    date = scrapy.Field()
    dayDesc = scrapy.Field()
    dayTemp = scrapy.Field()
    pass
```

4.2 编写获取天气数据的爬虫(Spider)

Spider是用户编写用于从单个网站(或者一些网站)爬取数据的类。

其包含了一个用于下载的初始URL，如何跟进网页中的链接以及如何分析页面中的内容， 提取生成 item 的方法。

为了创建一个Spider，必须继承 scrapy.Spider 类， 且定义以下三个属性:

- name: 用于区别Spider。该名字必须是唯一的，您不可以为不同的Spider设定相同的名字。
- start_urls: 包含了Spider在启动时进行爬取的url列表。因此，第一个被获取到的页面将是其中之一。后续的URL则从初始的URL获取到的数据中提取。
- parse() 是spider的一个方法。 被调用时，每个初始URL完成下载后生成的 Response 对象将会作为唯一的参数传递给该函数。 该方法负责解析返回的数据(response data)，提取数据(生成item)以及生成需要进一步处理的URL的 Request 对象。

我们通过浏览器的查看源码工具先来分析一下需要获取的数据网源代码：

```
<h4 class="slider_ct_name" id="slider_ct_name">武汉</h4>
...
<div class="blk_fc_c0_scroll" id="blk_fc_c0_scroll" style="width: 1700px;">
    <div class="blk_fc_c0_i">
        <p class="wt_fc_c0_i_date">01-28</p>
        <p class="wt_fc_c0_i_day wt_fc_c0_i_today">今天</p>
        <p class="wt_fc_c0_i_icons clearfix">
            <img class="icons0_wt png24" src="http://www.sinaimg.cn/dy/weather/main/index14/007/icons_42_yl/w_04
_27_00.png" alt="雨夹雪" title="雨夹雪">
            <img class="icons0_wt png24" src="http://www.sinaimg.cn/dy/weather/main/index14/007/icons_42_yl/w_04
_29_01.png" alt="中雪" title="中雪">
        </p>
        <p class="wt_fc_c0_i_times">
            <span class="wt_fc_c0_i_time">白天</span>
            <span class="wt_fc_c0_i_time">夜间</span>
        </p>
        <p class="wt_fc_c0_i_temp">1°C / -2°C</p>
        <p class="wt_fc_c0_i_tip">北风 3～4级</p>
        <p class="wt_fc_c0_i_tip">无持续风向 小于3级</p>
    </div>
    <div class="blk_fc_c0_i">
        <p class="wt_fc_c0_i_date">01-29</p>
        <p class="wt_fc_c0_i_day ">星期四</p>
        <p class="wt_fc_c0_i_icons clearfix">
            <img class="icons0_wt png24" src="http://www.sinaimg.cn/dy/weather/main/index14/007/icons_42_yl/w_04
_29_00.png" alt="中雪" title="中雪">
            <img class="icons0_wt png24" src="http://www.sinaimg.cn/dy/weather/main/index14/007/icons_42_yl/w_07
_25_01.png" alt="阴" title="阴">
        </p>
        <p class="wt_fc_c0_i_times">
            <span class="wt_fc_c0_i_time">白天</span>
            <span class="wt_fc_c0_i_time">夜间</span>
        </p>
        <p class="wt_fc_c0_i_temp">1°C / -2°C</p>
        <p class="wt_fc_c0_i_tip">无持续风向 小于3级</p>
    </div>
    ...
</div>
```


我们可以看到：

- 城市名可以通过获取id为slider_ct_name的h4元素获取
- 日期可以通过获取id为blk_fc_c0_scroll下的class为wt_fc_c0_i_date的p元素获取
- 天气描述可以通过获取id为blk_fc_c0_scroll下的class为icons0_wt的img元素获取
- 温度可以通过获取id为blk_fc_c0_scroll下的class为wt_fc_c0_i_temp的p元素获取

因此，我们的Spider代码如下，保存在 weather/spiders 目录下的 localweather.py 文件中:

```
# -*- coding: utf-8 -*-
import scrapy
from weather.items import WeatherItem


class WeatherSpider(scrapy.Spider):
    name = "myweather"
    allowed_domains = ["sina.com.cn"]
    start_urls = ['http://weather.sina.com.cn']

    def parse(self, response):
        item = WeatherItem()
        item['city'] = response.xpath('//*[@id="slider_ct_name"]/text()').extract()
        tenDay = response.xpath('//*[@id="blk_fc_c0_scroll"]');
        item['date'] = tenDay.css('p.wt_fc_c0_i_date::text').extract()
        item['dayDesc'] = tenDay.css('img.icons0_wt::attr(title)').extract()
        item['dayTemp'] = tenDay.css('p.wt_fc_c0_i_temp::text').extract()
        return item
```

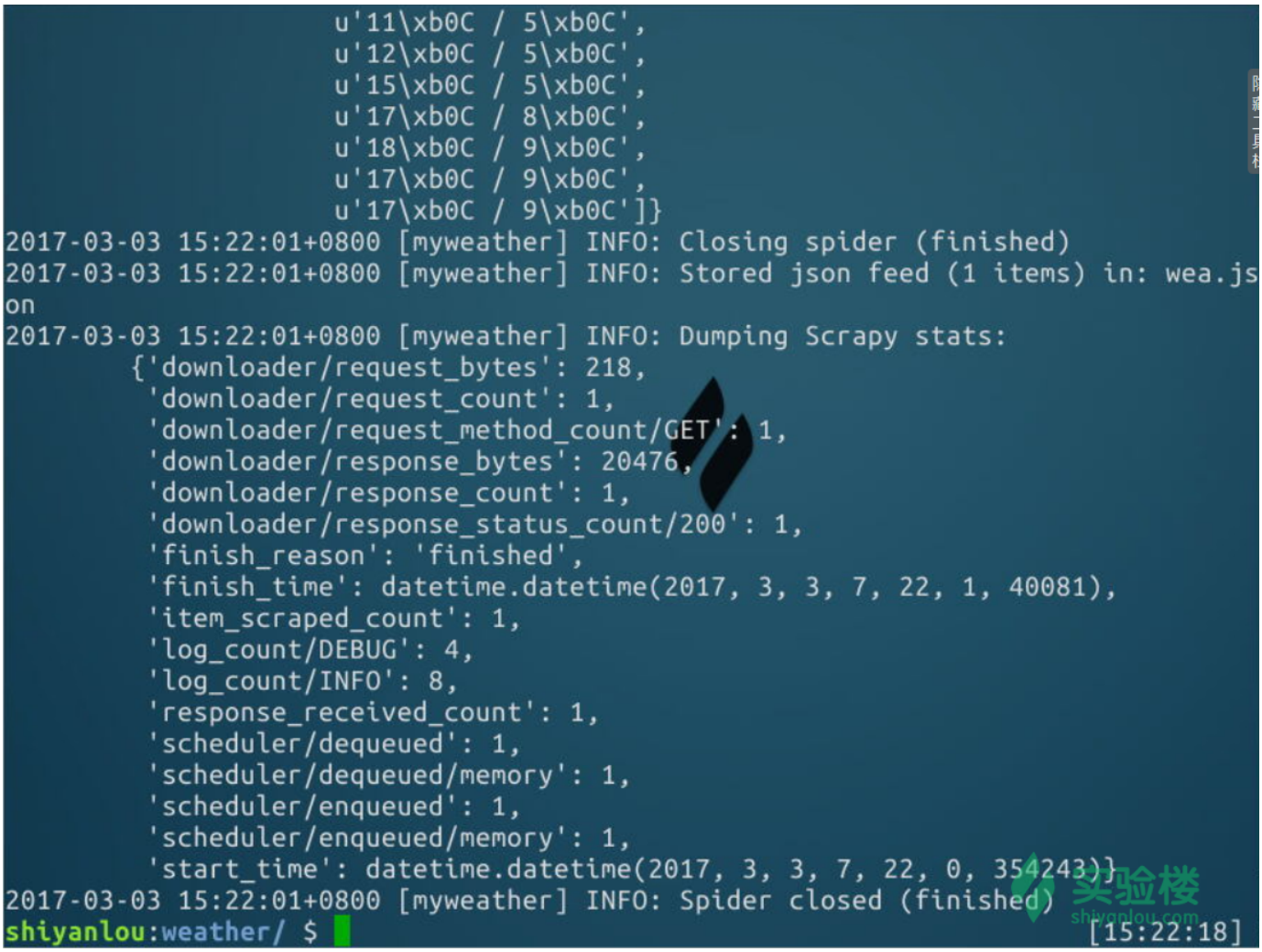
代码中的xpath和css后面括号的内容为选择器，关于xpath和css选择器的内容可参考官方教程：http://doc.scrapy.org/en/0.24/topics/selectors.html (http://doc.scrapy.org/en/0.24/topics/selectors.html)

4.3 运行爬虫，对数据进行验证

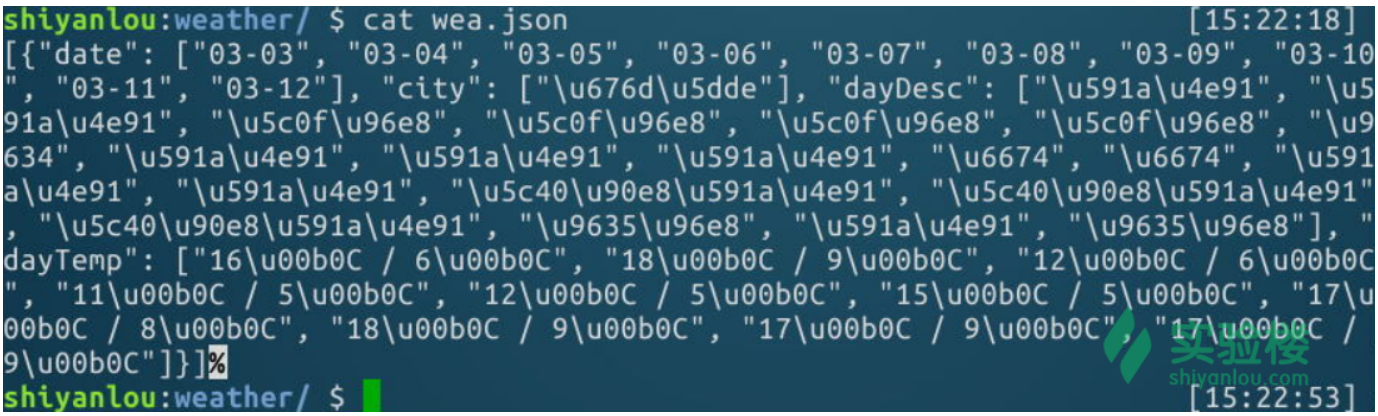
到这里为止，我们需要验证一下爬虫是否能正常工作（即能否取到我们想要的数据），验证的方法就是在命令行（重要：在项目的**scrapy.cfg**文件同级目录运行命令，下同）中运行下面的代码：


```
$ scrapy crawl myweather -o wea.json
```

这行命令的意思是，运行名字为 myweather 的爬虫（我们在上一步中定义的），然后把结果以json格式保存在wea.json文件中。命令运行结果如下：



然后，我们查看当前目录下的**wea.json**文件，正常情况下效果如下：



我们看到，**wea.json**中已经有数据了，只是数据是以unicode方式编码的。

4.4 保存爬取到的数据

上面只是把数据保存在json文件中了，如果我们想自己保存在文件或数据库中，如何操作呢？

这里就要用到 Item Pipeline 了，那么 Item Pipeline 是什么呢？

当Item在Spider中被收集之后，它将会被传递到Item Pipeline中，一些组件会按照一定的顺序执行对Item的处理。

每个item pipeline组件(有时称之为“Item Pipeline”)是实现了简单方法的Python类。他们接收到Item并通过它执行一些行为，同时也决定此Item是否继续通过pipeline，或是被丢弃而不再进行处理。

item pipeline的典型应用有：

- 清理HTML数据
- 验证爬取的数据(检查item包含某些字段)
- 查重(并丢弃)
- 将爬取结果保存到文件或数据库中

每个item pipeline组件都需要调用 process_item 方法，这个方法必须返回一个 Item (或任何继承类)对象， 或是抛出 DropItem异常，被丢弃的item将不会被之后的pipeline组件所处理。

我们这里把数据转码后保存在 wea.txt 文本中。

pipelines.py文件在创建项目时已经自动被创建好了，我们在其中加上保存到文件的代码：

```
# -*- coding: utf-8 -*-

# Define your item pipelines here
#
# Don't forget to add your pipeline to the ITEM_PIPELINES setting
# See: http://doc.scrapy.org/en/latest/topics/item-pipeline.html


class WeatherPipeline(object):
    def __init__(self):
        pass

    def process_item(self, item, spider):
        with open('wea.txt', 'w+') as file:
            city = item['city'][0].encode('utf-8')
            file.write('city:' + str(city) + '\n\n')

            date = item['date']

            desc = item['dayDesc']
            dayDesc = desc[1::2]
            nightDesc = desc[0::2]

            dayTemp = item['dayTemp']

            weaitem = zip(date, dayDesc, nightDesc, dayTemp)

            for i in range(len(weaitem)):
                item = weaitem[i]
                d = item[0]
                dd = item[1]
                nd = item[2]
                ta = item[3].split('/')
                dt = ta[0]
                nt = ta[1]
                txt = 'date:{0}\t\tday:{1}({2})\t\tnight:{3}({4})\n\n'.format(
                    d,
                    dd.encode('utf-8'),
                    dt.encode('utf-8'),
                    nd.encode('utf-8'),
                    nt.encode('utf-8')
                )
                file.write(txt)
        return item
```

代码比较简单，都是python比较基础的语法，如果您感觉比较吃力，建议先去学一下python基础课。

4.5 把 ITEM_PIPELINES 添加到设置中

写好ITEM_PIPELINES后，还有很重要的一步，就是把 ITEM_PIPELINES 添加到设置文件 settings.py 中。

```
ITEM_PIPELINES = {
    'weather.pipelines.WeatherPipeline': 1
}
```

另外，有些网站对网络爬虫进行了阻止（注：本项目仅从技术角度处理此问题，个人强烈不建议您用爬虫爬取有版权信息的数据），我们可以在设置中修改一下爬虫的 **USER_AGENT** 和 **Referer** 信息，增加爬虫请求的时间间隔。

整个 **settings.py** 文件内容如下：


```
# -*- coding: utf-8 -*-

# Scrapy settings for weather project
#
# For simplicity, this file contains only the most important settings by
# default. All the other settings are documented here:
#
#     http://doc.scrapy.org/en/latest/topics/settings.html
#

BOT_NAME = 'Googlebot'

SPIDER_MODULES = ['weather.spiders']
NEWSPIDER_MODULE = 'weather.spiders'

# Crawl responsibly by identifying yourself (and your website) on the user-agent
#USER_AGENT = 'weather (+http://www.yourdomain.com)'
USER_AGENT = 'User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.
0.2171.95 Safari/537.36'

DEFAULT_REQUEST_HEADERS = {
    'Referer': 'http://www.weibo.com'
}

ITEM_PIPELINES = {
    'weather.pipelines.WeatherPipeline': 1
}

DOWNLOAD_DELAY = 0.5
```

到现在为止，代码主要部分已经写完了。

4.6 运行爬虫

在项目的**scrapy.cfg**同级目录下用下面的命令运行爬虫：

```
$ scrapy crawl myweather
```

正常情况下，效果如下：

```
aped 0 items (at 0 items/min)
2017-03-03 15:32:33+0800 [scrapy] DEBUG: Telnet console listening on 127.0.0.1:6
023
2017-03-03 15:32:33+0800 [scrapy] DEBUG: Web service listening on 127.0.0.1:608
2017-03-03 15:32:33+0800 [myweather] DEBUG: Crawled (200) <GET http://weather.s
na.com.cn> (referer: http://www.weibo.com)
2017-03-03 15:32:33+0800 [myweather] DEBUG: Scraped from <200 http://weather.sin
a.com.cn>
        (u'03-12', u'\u9635\u96e8', u'\u591a\u4e91', u'17\xb0C / 9\xb0C')
2017-03-03 15:32:33+0800 [myweather] INFO: Closing spider (finished)
2017-03-03 15:32:33+0800 [myweather] INFO: Dumping Scrapy stats:
        {'downloader/request_bytes': 241,
         'downloader/request_count': 1,
         'downloader/request_method_count/GET': 1,
         'downloader/response_bytes': 20476,
         'downloader/response_count': 1,
         'downloader/response_status_count/200': 1,
         'finish_reason': 'finished',
         'finish_time': datetime.datetime(2017, 3, 3, 7, 32, 33, 779484),
         'item_scraped_count': 1,
         'log_count/DEBUG': 4,
         'log_count/INFO': 7,
         'response_received_count': 1,
         'scheduler/dequeued': 1,
         'scheduler/dequeued/memory': 1,
         'scheduler/enqueued': 1,
         'scheduler/enqueued/memory': 1,
         'start_time': datetime.datetime(2017, 3, 3, 7, 32, 33, 11957)}
2017-03-03 15:32:33+0800 [myweather] INFO: Spider closed (finished)
shiyanlou:weather/ $                                                [15:32:33]
```

然后，在当前目录下会多一个 wea.txt 文件，内容如下：

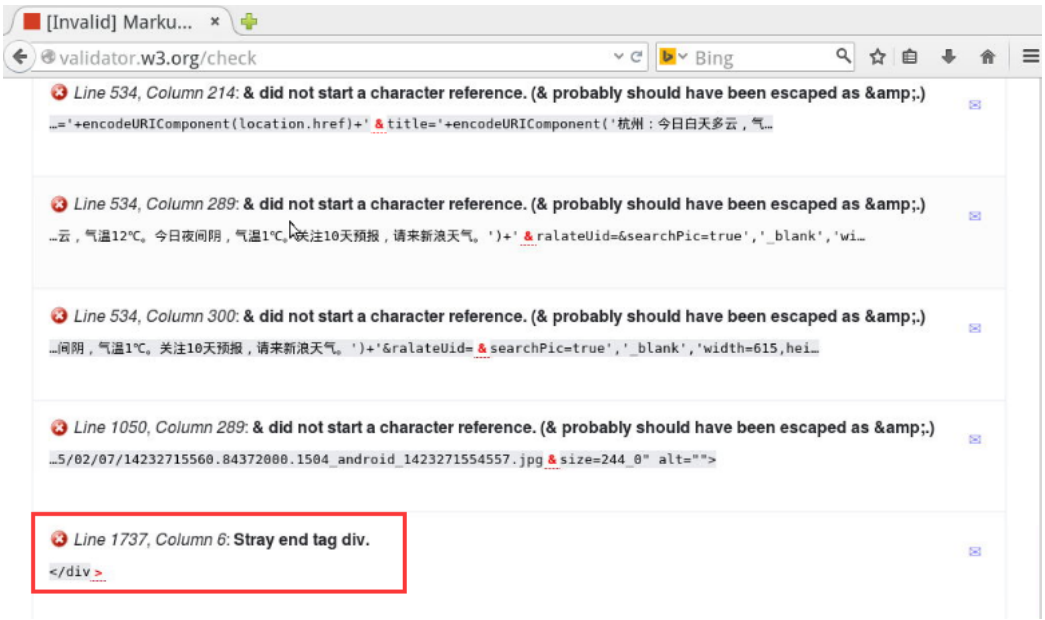
```
shiyanlou:weather/ $ ls                                             [15:33:14]
scrapy.cfg  wea.json  weather  wea.txt
shiyanlou:weather/ $ cat wea.txt                                    [15:33:15]
city:杭州

date:03-03              day:多云(16°C )         night:多云( 6°C)

date:03-04              day:小雨(18°C )         night:小雨( 9°C)

date:03-05              day:小雨(12°C )         night:小雨( 6°C)

date:03-06              day:多云(11°C )         night:阴( 5°C)

date:03-07              day:多云(12°C )         night:多云( 5°C)

date:03-08              day:晴(15°C )           night:晴( 5°C)

date:03-09              day:多云(17°C )         night:多云( 8°C)

date:03-10              day:局部多云(18°C )             night:局部多云( 9°C)

date:03-11              day:阵雨(17°C )         night:局部多云( 9°C)

date:03-12              day:阵雨(17°C )         night:多云( 9°C)

shiyanlou:weather/ $                                                [15:33:20]
```

到此我们基于scrapy的天气数据采集就完成了。

五、FAQ

5.1. 结果只出现城市?

scrapy内置的html解析是基于lxml库的，这个库对html的解析的容错性不是很好，通过检查虚拟机中获取到的网页源码，发现有部分标签是不匹配的（地区和浏览器不同取到的源码可能不同），检查结果如图：



所以导致在spider中取到的日期数据(item['date'])为空，然后在pilepine代码中做zip操作后，整个 weaitem 为空，所以最终只有城市数据了。

既然如此，我们换个html代码解析器就可以了，这里建议用 BeautifulSoup （官网：http://www.crummy.com/software/BeautifulSoup/bs4/doc/index.html (http://www.crummy.com/software/BeautifulSoup/bs4/doc/index.html) ），这个解析器有比较好的容错能力，具体用法可以参考上面的文档。

BeautifulSoup安装：

```
#下载BeautifulSoup
$ wget http://labfile.oss.aliyuncs.com/beautifulsoup4-4.3.2.tar.gz

#解压
$ tar -zxvf beautifulsoup4-4.3.2.tar.gz

#安装
$ cd beautifulsoup4-4.3.2
$ sudo python setup.py install
```

安装成功后，优化WeatherSpider代码，改进后的代码如下：

```
# -*- coding: utf-8 -*-
import scrapy
from bs4 import BeautifulSoup
from weather.items import WeatherItem


class WeatherSpider(scrapy.Spider):
    name = "myweather"
    allowed_domains = ["sina.com.cn"]
    start_urls = ['http://weather.sina.com.cn']

    def parse(self, response):
        html_doc = response.body
        #html_doc = html_doc.decode('utf-8')
        soup = BeautifulSoup(html_doc)
        itemTemp = {}
        itemTemp['city'] = soup.find(id='slider_ct_name')
        tenDay = soup.find(id='blk_fc_c0_scroll')
        itemTemp['date'] = tenDay.findAll("p", {"class": 'wt_fc_c0_i_date'})
        itemTemp['dayDesc'] = tenDay.findAll("img", {"class": 'icons0_wt'})
        itemTemp['dayTemp'] = tenDay.findAll('p', {"class": 'wt_fc_c0_i_temp'})
        item = WeatherItem()
        for att in itemTemp:
            item[att] = []
            if att == 'city':
                item[att] = itemTemp.get(att).text
                continue
            for obj in itemTemp.get(att):
                if att == 'dayDesc':
                    item[att].append(obj['title'])
                else:
                    item[att].append(obj.text)
        return item
```


然后再次运行爬虫:

```
$ scrapy crawl myweather
```

然后查看 wea.txt，数据如下:

```
shiyanlou:Code/ $ cat scrapy-weather/weather/wea.txt                    [11:04:49]
city:杭

date:02-07              day:阴(12°C )           night:多云( 1°C)

date:02-08              day:多云(8°C )          night:多云( -1°C)

date:02-09              day:多云(9°C )          night:晴( 0°C)

date:02-10              day:多云(14°C )         night:多云( 3°C)

date:02-11              day:多云(14°C )         night:多云( 4°C)

date:02-12              day:晴(15°C )           night:多云( 3°C)

date:02-13              day:多云(15°C )         night:晴( 4°C)

date:02-14              day:阵雨(18°C )         night:少云( 7°C)

date:02-15              day:零散阵雨(21°C )             night:少云( 8°C)

date:02-16              day:零散阵雨(18°C )             night:阵雨( 6°C)

shiyanlou:Code/ $                                                       [11:05:49]
```

六、课后习题

晚上只取到了9天的数据?

如果是晚上运行爬虫，当天的白天天气是没有的（已经过去了），针对这部分我们需要修改代码来适应，留作作业给大家思考。

课程教师