Webmagic

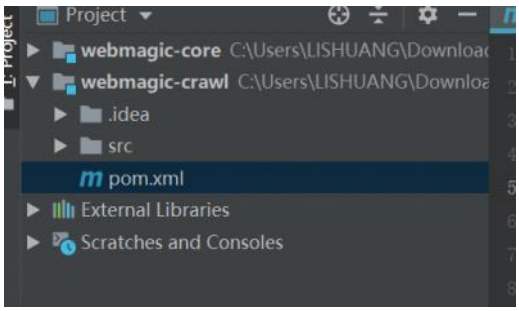
1. 创建maven项目 jdk1.8
2. 从github引入依赖

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/us.codecraft/webmagic-core -->
    <dependency>
        <groupId>us.codecraft</groupId>
        <artifactId>webmagic-core</artifactId>
        <version>0.7.3</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/us.codecraft/webmagic-extension -->
    <dependency>
        <groupId>us.codecraft</groupId>
        <artifactId>webmagic-extension</artifactId>
        <version>0.7.3</version>
    </dependency>
```

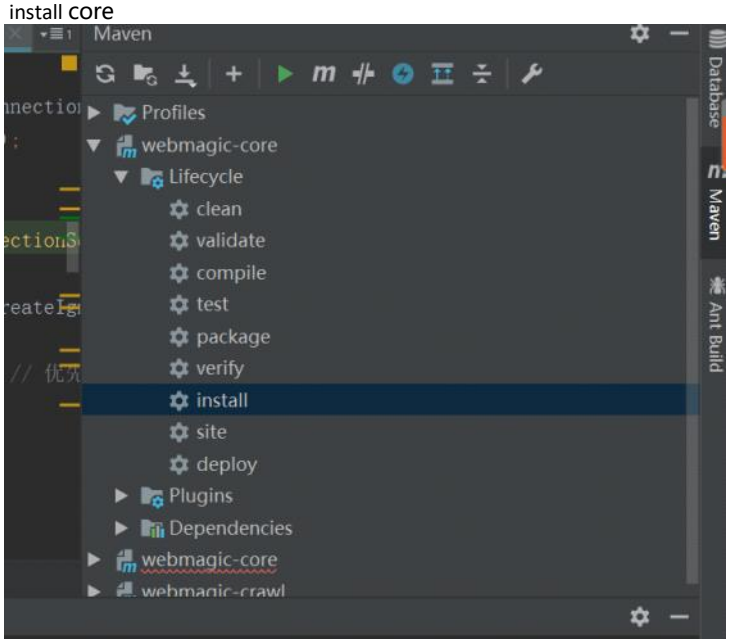
屏幕剪辑的捕获时间: 2019/12/27 16:06

3. 该依赖存在一定访问问题 701，因此需要手动导入webmagic-core
   File->module from existing soure导入github中的core包



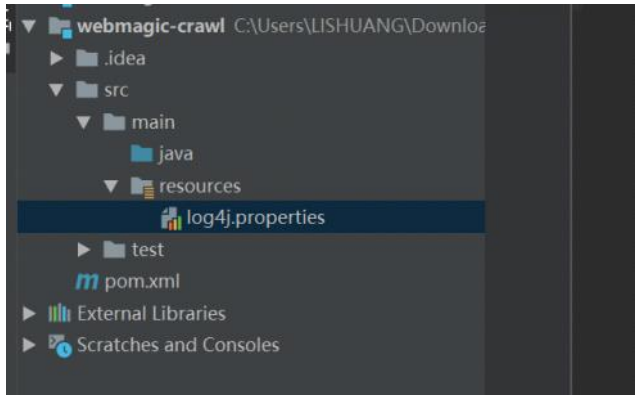
屏幕剪辑的捕获时间: 2019/12/27 16:08

4. 如果 pom文件的project fail to read，那么说明maven库由于网络原因下载失败，需要删除.m2文件中resposity中的文件，重新reimport。
5. install core



屏幕剪辑的捕获时间: 2019/12/27 16:12

6. 添加log4j配置文件

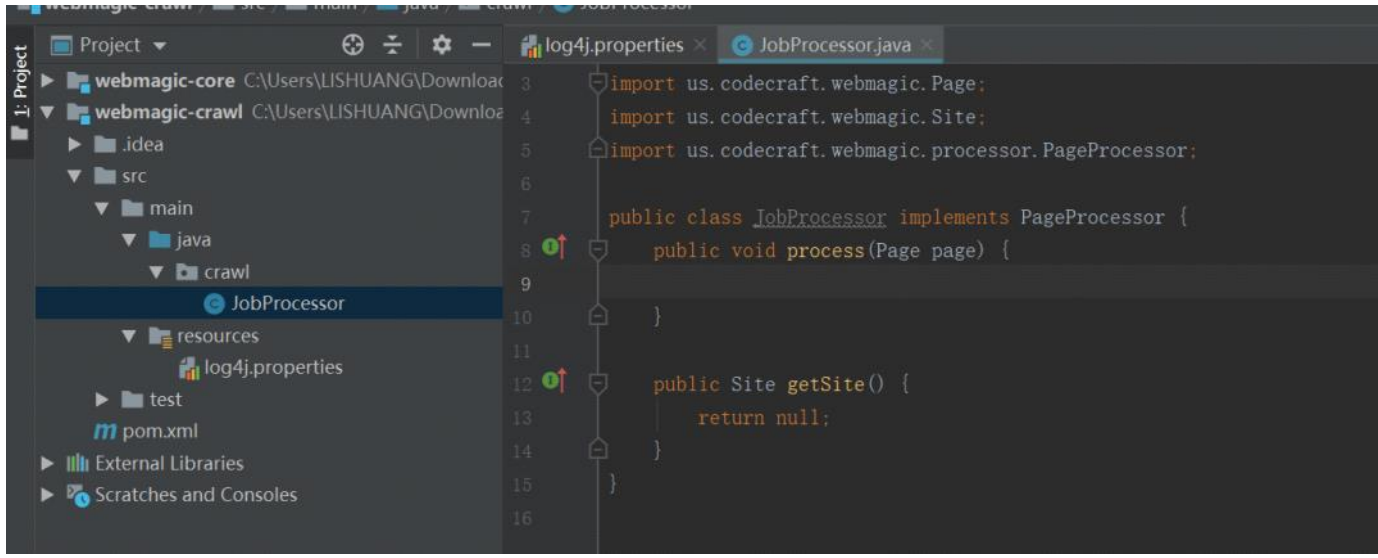


屏幕剪辑的捕获时间: 2019/12/27 16:14

```
log4j.rooting=INFO,A1
```

```
log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-ddHH:mm:ss,SSS}[%t][%c]-
[%p]%m%n
```

7.创建处理class alt+enter自动创建需要实现的方法



屏幕剪辑的捕获时间: 2019/12/27 16:27

2. Page

Page 代表了从 Downloader 下载到的一个页面——可能是 HTML，也可能是 JSON 或者其他文本格式的内容。

Page 是 WebMagic 抽取过程的核心对象，它提供一些方法可供抽取、结果保存等。

3. ResultItems

ResultItems 相当于一个 Map，它保存 PageProcessor 处理的结果，供 Pipeline 使用。它的 API 与 Map 很类似，值得注意的是它有一个字段 skip，若设置为 true，则不应被 Pipeline 处理。

- **3.1.2. 抽取元素 API**

Selectable 相关的抽取元素链式 API 是 WebMagic 的一个核心功能。使用 Selectable 接口，可以直接完成页面元素的链式抽取，也无需去关心抽取的细节。

在刚才的例子中可以看到，page.getHtml()返回的是一个 Html 对象，它实现了 Selectable 接口。这个接口包含的方法分为两类：抽取部分和获取结果部分。

屏幕剪辑的捕获时间: 2019/12/27 17:40

屏幕剪辑的捕获时间: 2019/12/27 16:36

有三种方式解析页面 返回值均为selectable

1. xpath

```
//Xpath
page.putField( key: "div", page.getHtml().xpath("//div[@id=shortcut]/div/ul/li/a"));
```

屏幕剪辑的捕获时间: 2019/12/27 17:27

div的id为shortcut下的div下的ul下的li的a标签内的内容

```
<div id="shortcut">
    <div class="w">
        <ul class="fl" clstag="h|keycount|head|topbar_01">
            <li class="dropdown" id="ttbar-mycity"></li>
        </ul>

        <ul class="fr">
            <li class="fore1 dropdown" id="ttbar-login" clstag="h|keycount|head|topbar_02">
                <a target="_blank" href="javascript:login();" class="link-login">你好，请登录</a>  <a
                    href="javascript:regist();" class="link-regist style-red">免费注册</a>
            </li>
            <li class="spacer"></li>
```

屏幕剪辑的捕获时间: 2019/12/27 17:28

2. css选择器
3. 正则表达式

使用Pipeline保存结果

WebMagic 用于保存结果的组件叫做 Pipeline。我们现在通过"控制台输出结果"这件事也是通过一个内置的 Pipeline 完成的，它叫做 ConsolePipeline。

那么，我现在想要把结果用保存到文件中，怎么做呢？只将 Pipeline 的实现换成"FilePipeline"就可以了。

```
public static void main(String[] args) {
    Spider.create(new JobProcessor())
            //初始访问 url 地址
            .addUrl("https://www.jd.com/moreSubject.aspx")
            .addPipeline(new FilePipeline("D:/webmagic/"))
            .thread(5)//设置线程数
            .run();
}
```

屏幕剪辑的捕获时间: 2019/12/27 18:04

加入Selemiun实现自动点击下一页进行信息的爬取，因为url不会变，而无法获取下一页的url

1. 引入selenium和Web driver的依赖
2. 添加配置文件

```
System.setProperty("selenuim_config", "C:\\Users\\LISHUANG\\Downloads\\webmagic-crawl\\src\\main\\resources\\se
```

屏幕剪辑的捕获时间: 2019/12/30 15:41

3. 自动启动浏览器

```
.setDownloader(new SeleniumDownloader( chromeDriverPath: "E:\\chromedriver.exe").setSleepTime(3000))
```

屏幕剪辑的捕获时间: 2019/12/30 15:40

4. selenium自动控制打开窗口

```
//selenium自动控制
WebDriver driver = new ChromeDriver();
driver.manage().window().maximize();//窗口最大化
driver.manage().timeouts().implicitlyWait( l: 8, TimeUnit.SECONDS);
driver.get(url);
```

5. 获取第一页包含cJba的标签列表

```
List<WebElement> list = driver.findElements(By.xpath("//td[contains(text(),'CJba')]"));
```

屏幕剪辑的捕获时间: 2019/12/30 15:47

6. 将webElement格式转化为String文本

```
List<String> listStr=new ArrayList<String>();
for(WebElement element:list){
    listStr.add(element.getText());
    //System.out.println(element.getText());
}
```

屏幕剪辑的捕获时间: 2019/12/30 15:47

7. 自动点击下一页 并抓取文本

```
//选择下一页元素进行点击
driver.findElement(By.cssSelector("div.btn-container ul.pager button:nth-of-type(3)")).click();
```

屏幕剪辑的捕获时间: 2019/12/30 15:48

8. 放入page管理中

```
page.putField(word,listStr);
```

屏幕剪辑的捕获时间: 2019/12/30 15:49

9. 输出到文件

```
.addPipeline(new FilePipeline( path: "C:\\Users\\LISHUANG\\Desktop\\result"))//输出到文件
```

屏幕剪辑的捕获时间: 2019/12/30 15:49