链滴

你有多久没有看星星了呢？【爬取 NASA 的科普网站上的所有图片】

作者：MingGH

原文链接：https://ld246.com/article/1636206449328

来源网站：链滴

1. 前景提要

在网上冲浪的时候看到原来NASA航天局有科普网站，每天一张科普图片，而且是非常高清的那种，想下载下来做壁纸。

所以打算写一个Java爬虫爬取所有的图片，也可以加入一些通知，每天晚上进行检查，当检查到有更的时候，第二天早上起床推送到手机端。当然这个功能还没有实现。

2. 进行开发

2.1 开发环境

- Maven
- Java 17 (为了顺行Spring Framework 6 之后的潮流，所以打算之后的开发都用Zulu JDK17)
- WebMagic（Java的爬虫框架）

2.2 Maven pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.yeahvin</groupId>
    <artifactId>getStars</artifactId>
    <version>1.0.1-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>us.codecraft</groupId>
            <artifactId>webmagic-core</artifactId>
            <version>0.7.5</version>
        </dependency>
        <dependency>
            <groupId>us.codecraft</groupId>
            <artifactId>webmagic-extension</artifactId>
            <version>0.7.5</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.25</version>
        </dependency>
```

```
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>ch.qos.logback</groupId>
            <artifactId>logback-classic</artifactId>
            <version>1.2.3</version>
        </dependency>
    </dependencies>


    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <mainClass>com.yeahvin.StartApplication</mainClass>
                    <layout>ZIP</layout>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

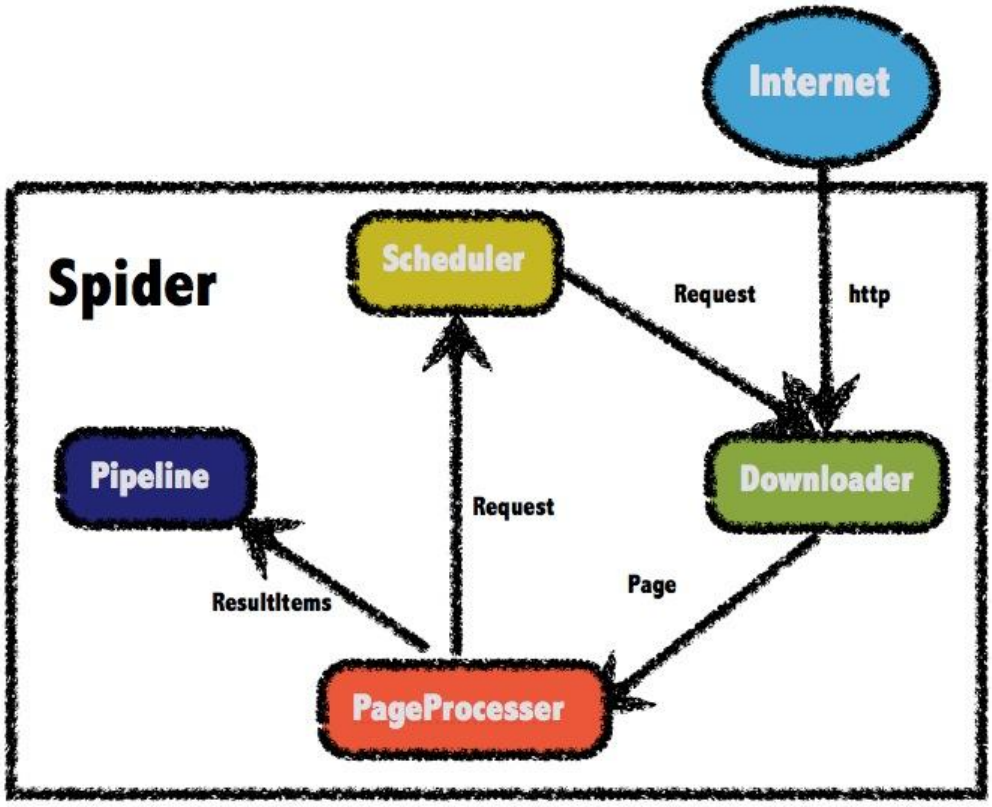
2.3 Java爬虫webmagic 的使用

这个框架也是非常容易上手和使用，官方手册在这里 WebMagic in Action

对于这个框架的使用，这里简单的介绍一下，如下图，我们开发者需要在整个流程中需要做的是：

1. 编写PageProceser
2. 编写Pipeline

就可以完成一个非常简单的爬虫。推荐大家先看看这个例子：第一个例子



在这个项目中，对于整个nasa网站的分析其实非常的简单，流程大概是

请求总览页面-->获取已经存在所有链接，进行遍历-->请求详情网页--->下载图片

启动类

ok, 那么来进行开发，编写StartApplication，这是启动爬虫的地方。

```
package com.yeahvin;

import com.yeahvin.pageProcessors.NasaStarsProcessor;
import com.yeahvin.pipelines.NasaStarsPipeline;
import lombok.extern.slf4j.Slf4j;
import us.codecraft.webmagic.Spider;

/**
 * @author Asher
 * on 2021/11/6
 */
@Slf4j
public class StartApplication {

    //下载图片存储路径
    public static final String DOWNLOAD_PATH = "/Users/asher/gitWorkspace/temp/";
    public static final String INDEX_PATH = "https://apod.nasa.gov/apod/archivepix.html";

    public static void main(String[] args) {
        Spider.create(new NasaStarsProcessor())
                .addUrl(INDEX_PATH)
```

```
                .addPipeline(new NasaStarsPipeline())
                .run();
    }
}
```

以上代码中，Spider.create(Processor)可以非常快速的构建一个爬虫，然后 addUrl配置请求的地址，ddPipeline配置当爬虫抓取到数据时应该怎么进行处理，最后就是run进行启动了

编写NasaStarsProcessor

编写这个的原因是为了，请求到网页之后，怎么对网页上的内容进行一个分析和处理

```
package com.yeahvin.pageProcessors;

import com.yeahvin.StartApplication;
import com.yeahvin.utils.StringUtil;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import us.codecraft.webmagic.Page;
import us.codecraft.webmagic.Site;
import us.codecraft.webmagic.processor.PageProcessor;

import java.io.File;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

import static com.yeahvin.StartApplication.DOWNLOAD_PATH;

/**
 * @author Asher
 * on 2021/11/6
 */
@Slf4j
public class NasaStarsProcessor implements PageProcessor {

    private Site site = Site.me().setRetryTimes(5).setSleepTime(1000).setTimeOut(10000);

    @Override
    public void process(Page page) {
        if (page.getUrl().regex(StartApplication.INDEX_PATH).match()){

            //获取文件夹下已经保存的文件
            File saveDir = new File(DOWNLOAD_PATH);
            List<String> fileNames = Arrays.stream(Optional.ofNullable(saveDir.listFiles()).orElse(n
w File[]{}))
                    .map(File::getName)
                    .collect(Collectors.toList());

            log.info("开始主页抓取");
            List<String> starsInfos = page.getRawText()
                    .lines()
```

```
                .map(line -> {
                    List<String> infoList = StringUtil.getRegexString(line, "^\\d{4}.*<br>$");
                    boolean flag = infoList.size() > 0;
                    return flag ? infoList.get(0) : null;
                })
                .filter(StringUtils::isNoneBlank)
                .collect(Collectors.toList());

        starsInfos.forEach(info -> {
            String link = StringUtil.getRegexString(info, "ap.*html").get(0);
            String name = StringUtil.getRegexString(info, ">.*</a>").get(0)
                    .replace("</a>", "")
                    .replace(">", "");

            //如果已经存在对应的文件，则进行跳过
            if (fileNames.stream().filter(fileName -> fileName.contains(name)).count() > 0) {
                log.info("文件 [{}] 已经存在,进行跳过", name);
                return;
            }
            page.addTargetRequest("https://apod.nasa.gov/apod/" + link);
        });
        log.info("完成主页抓取，一共有{}条数据", starsInfos.size());
    }else {
        page.getRawText()
            .lines()
            .filter(line -> line.contains("<title>")
                    || Pattern.compile("<a href=\"image.*jpg.*\">").matcher(line).find()
                    || Pattern.compile("<a href=\"image.*jpg.*\"").matcher(line).find()
            )
            .forEach(line -> {
                if (line.contains("<title>")){
                    String date = StringUtil.getRegexString(line, "\\d{4}.*-").get(0)
                            .replace("-", "");
                    String name = StringUtil.getRegexString(line, "-.*").get(0)
                            .replace("-", "");
                    page.putField("date", date);
                    page.putField("name", name);
                }else {
                    String imageLink = StringUtil.getRegexString(line, "image.*jpg").get(0);
                    page.putField("imageLink", imageLink);
                }
                page.putField("url", page.getUrl());
            });

    }
  }

  @Override
  public Site getSite() {
    return site;
  }
}
```

以上代码的逻辑大概是：

- 属性配置一个Site，作为爬虫请求网页时的配置。
- 实现接口 PageProcessor，需要重写方法process
- 在方法 process中，先请求总览页面，通过if进行了判断，如果进来的请求和总览页一样，则跳详情页处理，就是在else中。为什么可以这样进行操作，因为在 if(true)的条件处理中，可以通过以代码直接在当前 Processor再请求一次放入的link，就不必重复编写Processor，节省开发量。

```
page.addTargetRequest("https://apod.nasa.gov/apod/" + link);
```

- 其中在if条件==true的时候，会先拿取指定保存文件夹中的所有文件的文件名，当应用中途停止在启动时，不用把之前的网页再跑一遍。然后 getRawText()获取到网页文本内容，通过正则进行过滤拿到所有详情页的link
- 所以else中的就是详情页的代码处理过程，例如 Astronomy Picture of the Day 界面获取到网页容后的处理过程。其中 getRawText()获取到网页文本内容，通过正则进行过滤，page.putField("dat", date); 放入page变量中，当中放入的值会被传入启动类 StartApplication指定的 NasaStarsPipeline 的方法。

编写NasaStarsPipeline

编写这个NasaStarsPipeline就是为了在上面的Processor放入Field之后，我们需要对Field的内容进处理的过程。

其中大致的逻辑是：

- 获取到详情页的具体信息，比如name，datel以及图片的下载链接，然后对图片进行下载。这里需注意的是，有时候打开的详情页并不一定是图片，还有可能是视频，当出现无法以图片的方式进行解的时候，这里的处理方式是直接进行跳过。保存错误日志。

```
package com.yeahvin.pipelines;

import com.yeahvin.utils.X509TrustManager;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import us.codecraft.webmagic.ResultItems;
import us.codecraft.webmagic.Task;
import us.codecraft.webmagic.pipeline.Pipeline;

import java.io.File;

import static com.yeahvin.StartApplication.DOWNLOAD_PATH;

/**
 * @author Asher
 * on 2021/11/6
 */
@Slf4j
public class NasaStarsPipeline implements Pipeline {

    @Override
    public void process(ResultItems resultItems, Task task) {
        if (resultItems.getAll()!=null && resultItems.getAll().size()>0){
            String date = resultItems.get("date");
            String name = resultItems.get("name");
```

```
            String imageLink = resultItems.get("imageLink");

            if (StringUtils.isBlank(imageLink)){
                log.error("获取图片下载链接失败，name:{}, date:{}, url:{}", name, date, resultItems.get(
url"));
                return;
            }

            String fileName = date + name + ".jpg";
            if (new File(DOWNLOAD_PATH + fileName).exists()) {
                log.info("下载图片{}已经存在，跳过，name:{}, date:{}", imageLink, name, date);
                return;
            }
            try {
                X509TrustManager.downLoadFromUrlHttps("https://apod.nasa.gov/apod/" + imag
Link, fileName, DOWNLOAD_PATH);
            } catch (Exception e) {
                log.error("图片下载失败：", e);
                e.printStackTrace();
            }
            log.info("下载图片{}完成，name:{}, date:{}", imageLink, name, date);
        }
    }

}
```

当你写完这么多的时候，整个代码其实就已经完成了，后面可以再做一些扩展。

比如图片多线程的下载，每天定时任务去执行，这里的话我是通过shell脚本去完成这个定时任务的，不打算常挂这个应用在内存中。

3. 代码

地址：getStars