用Java实现网易云音乐爬虫

David · 12 天前

由Cliff 发表在天码营

起因

前两天在知乎上看到一个帖子《网易云音乐有哪些评论过万的歌曲？》，一时技痒，用Java实现了一个简单的爬虫，这里简单记录一下。

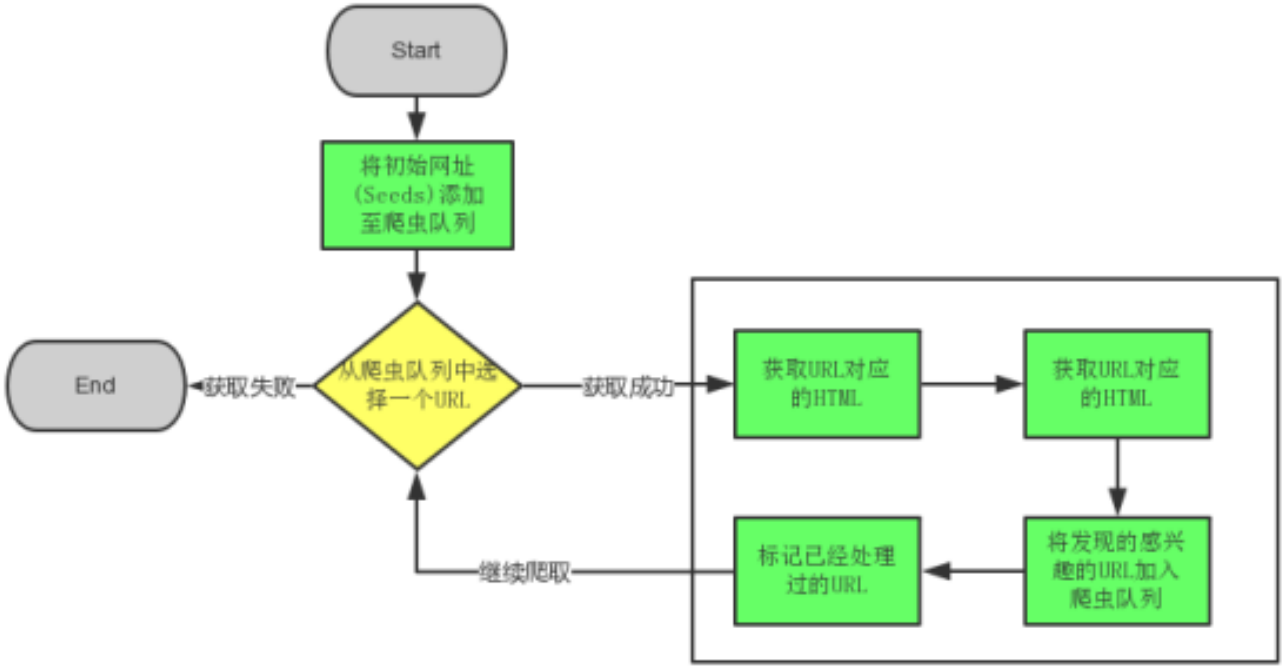
最终的结果开放出来了，大家可以随意访问，请戳这里>>>>>> 网易云音乐爬虫结果 。

完整的源代码戳右上角“参考代码 ”链接。

爬虫简介

网络爬虫是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本，一个通用的网络爬

虫大致包含以下几个步骤：



网络爬虫的大致流程如上图所示，无论你是做什么样的爬虫应用，整体流程都是大同小异。现在，我们就根据网易云音乐来定制一个专门爬取音乐评论数量的特定网络爬虫。

前期准备

网页类型分析

首先，我们需要对网易云音乐 整个网站有个大致的了解，进入网易云音乐首页 ，浏览后发现其大概有这么几种类型的URL：

- 推荐页面
- 排行榜列表以及排行榜页面
- 歌单列表以及歌单页面
- 主播电台列表以及主播电台页面
- 歌手列表以及歌手页面
- 专辑列表(新碟上架)以及专辑页面
- 歌曲页面

我们最终需要爬取的数据在歌曲页面中，该页面里包含了歌曲的名称以及歌曲的评论数量。

另外，我们还需要尽可能多的获取歌曲页面，这些信息我们可以从前面6种类型的页面拿到。其中，歌单列表以及歌单页面结构最简单，歌单列表直接分页就可以拿到。因此，我们选择歌单页面作为我们的初始页面，然后歌单列表--歌单--歌曲一路爬下去即可。

设计数据模型

通过上述分析，我们可以知道我们要做两件事情，一是爬取页面歌单列表--歌单--歌曲，二是将最终的结果存储起来。因此，我们只需要两个对象，一个用来存储页面相关的信息，url、页面类型、是否被爬过（html和title作为临时数据存储），另外一个用来存储歌曲相关信息，url、歌曲名，评论数。因此，model类如下：

```
public class WebPage {
    public enum PageType {
        song, playlist, playlists;
    }

    public enum Status {
        crawled, uncrawl;
    }

    private String url;
    private String title;
    private PageType type;
    private Status status;
    private String html;
    ...
}
```

```
public class Song {
    private String url;
    private String title;
    private Long commentCount;
    ...
}
```

获取网页内容并解析

根据之前的分析，我们需要爬的页面有三种：歌单列表、歌单以及歌曲。为了验证想法的可行性，我们先用代码来解析这三种类型的网页，我们将网页内容获取以及解析的代码都放入CrawlerThread当中。

获取html

无论想要从什么网站中拿到数据，获取其html代码都是最最基础的一步，这里我们使用jsoup来获取页面信息，在CrawlerThread中添加如下代码：

```
private boolean fetchHtml(WebPage webPage) throws IOException {
    Connection.Response response = Jsoup.connect(webPage.getUrl()).time
    webPage.setHtml(response.body());
    return response.statusCode() / 100 == 2 ? true : false;
}

public static void main(String[] args) throws Exception {
    WebPage playlists = new WebPage("http://music.163.com/#/discover/pl
    CrawlerThread crawlerThread = new CrawlerThread();
    crawlerThread.fetchHtml(playlists);
    System.out.println(playlists.getHtml());
}
```

运行后即可看到html文本的输出

解析歌单列表页面

得到html后，我们来解析歌单列表，拿到页面中的所有歌单,Jsoup包含了html解析相关的功能，我们无需添加其他依赖，直接在CrawlerThread中添加如下代码：

```
private List<WebPage> parsePlaylist(WebPage webPage) {
    Elements songs = Jsoup.parse(webPage.getHtml()).select("ul.f-hide l
    return songs.stream().map(e -> new WebPage(BASE_URL + e.attr("href"
}
```

```
public static void main(String[] args) throws Exception {
    WebPage playlists = new WebPage("http://music.163.com/discover/play
    CrawlerThread crawlerThread = new CrawlerThread();
    crawlerThread.fetchHtml(playlists);
    System.out.println(crawlerThread.parsePlaylists(playlists));
}
```

解析歌单页面

和歌单列表页面类似，只需要将歌曲相关的元素找出来即可：

```
private List<WebPage> parsePlaylist(WebPage webPage) {
    Elements songs = Jsoup.parse(webPage.getHtml()).select("ul.f-hide li
    return songs.stream().map(e -> new WebPage(BASE_URL + e.attr("href"
}

public static void main(String[] args) throws Exception {
    WebPage playlist = new WebPage("http://music.163.com/playlist?id=454
    CrawlerThread crawlerThread = new CrawlerThread();
    crawlerThread.fetchHtml(playlist);
    System.out.println(crawlerThread.parsePlaylist(playlist));
}
```

注意，这里为了方便，我们将歌曲的名称也拿到了，这样后面我们就不需要再次获取歌曲名称了。

解析歌曲页面

终于到歌曲页面了，这里网易云音乐做了反爬处理，获取数据时的参数需要经过加密处理，这里我们不纠结于具体算法，如果有兴趣的直接看参考代码，我们只看关键代码：

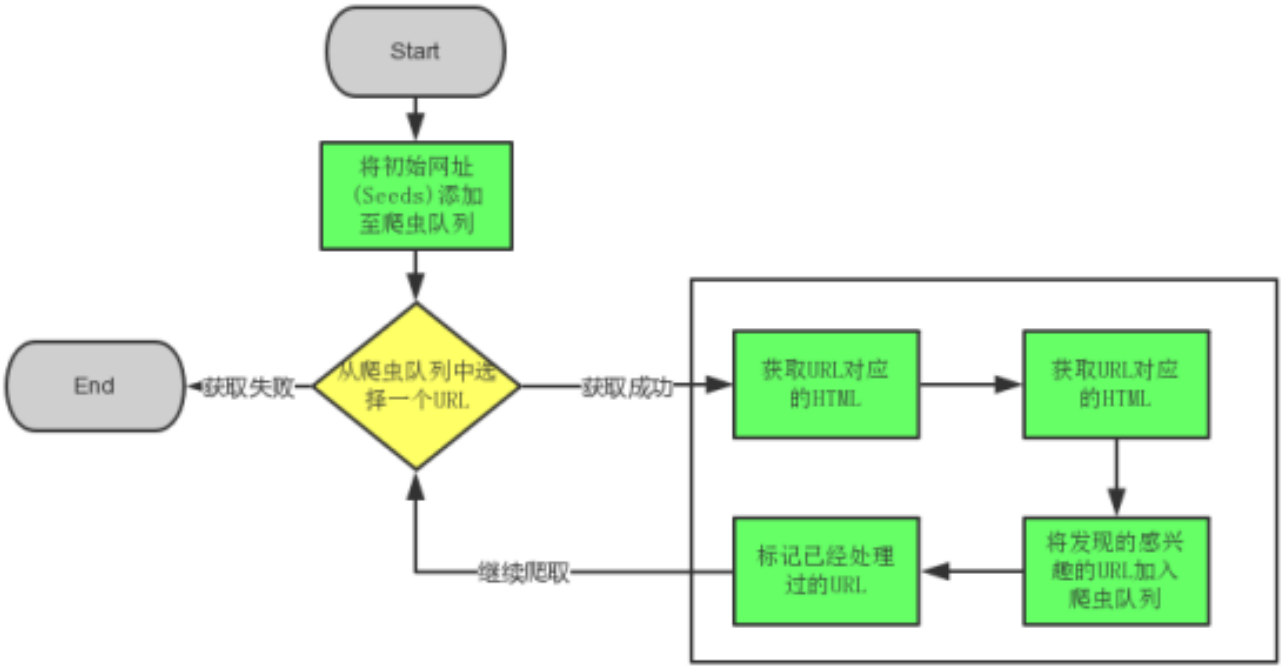
```
private Song parseSong(WebPage webPage) throws Exception {
        return new Song(webPage.getUrl(), webPage.getTitle(), getCommen
}

public static void main(String[] args) throws Exception {
```

```
    WebPage song = new WebPage("http://music.163.com/song?id=29999506",
    CrawlerThread crawlerThread = new CrawlerThread();
    crawlerThread.fetchHtml(song);
    System.out.println(crawlerThread.parseSong(song));
}
```

好吧，获取过程确实比较曲折，经过了多次的加密，不过不管怎么样，最终我们还是拿到了我们想要的数据。接下来，就是使用爬虫将整套机制run起来了。

实现爬虫

重新回顾一下流程图，我们发现其中有很重要的一个对象是爬虫队列，爬虫队列的实现方法有很多种，自己实现，mysql、redis、MongoDB等等都可以满足我们的需求，不同的选择会导致我们实现的不一致。



综合考虑，我们使用Mysql+ Spring Data JPA + Spring MVC来跑我们的整套框架，最终还可以将爬下来的数据通过web服务展现出来。更深入地学习Spring MVC，请大家参考Spring MVC实战入门训练 。

确定好之后，我们就可以开始一步步实现了。这里Spring Data JPA的代码就不展示了。了解Spring Data JPA，请参考Spring Data JPA实战入门训练 。直接上核心代码，所有和爬虫整体

流程相关的代码我们都放进CrawlerService中。

初始网址

第一步建立一个初始网址，我们可以根据歌单列表分页的特征得到：

```
private void init(String catalog) {
    List<WebPage> webPages = Lists.newArrayList();
    for(int i = 0; i < 43; i++) {
        webPages.add(new WebPage("http://music.163.com/discover/playlis
    }
    webPageRepository.save(webPages);
}

public void init() {
    webPageRepository.deleteAll();
    init("全部");
    init("华语");
    init("欧美");
    init("日语");
    init("韩语");
    init("粤语");
    init("小语种");
    init("流行");
    init("摇滚");
    init("民谣");
    init("电子");
    init("舞曲");
    init("说唱");
    init("轻音乐");
    init("爵士");
    init("乡村");
    init("R&B/Soul");
    init("古典");
    init("民族");
    init("英伦");
    init("金属");
    init("朋克");
```

```
    init("蓝调");
    init("雷鬼");
    init("世界音乐");
    init("拉丁");
    init("另类/独立");
    init("New Age");
    init("古风");
    init("后摇");
    init("Bossa Nova");
    init("清晨");
    init("夜晚");
    init("学习");
    init("工作");
    init("午休");
    init("下午茶");
    init("地铁");
    init("驾车");
    init("运动");
    init("旅行");
```



```
    init("清新");
    init("浪漫");
    init("性感");
    init("伤感");
    init("治愈");
    init("放松");
    init("孤独");
    init("感动");
    init("兴奋");
    init("快乐");
    init("安静");
    init("思念");
    init("影视原声");
    init("ACG");
    init("校园");
    init("游戏");
    init("70后");
```

```
    init("80后");
    init("90后");
    init("网络歌曲");
    init("KTV");
    init("经典");
    init("翻唱");
    init("吉他");
    init("钢琴");
    init("器乐");
    init("儿童");
    init("榜单");
    init("00后");
}
```

这里，我们初始化了歌单所有分类的列表，通过这些列表，我们就能拿到网易云音乐大部分的歌曲。

从爬虫队列中拿到一个URL

这里的逻辑非常简单，从mysql中获取一个状态为未爬的网页即可，但是由于我们需要爬的网址非常的多，肯定要用到多线程，因此需要考虑异步的情况：

```
public synchronized WebPage getUnCrawlPage() {
    WebPage webPage = webPageRepository.findTopByStatus(Status.uncrawl)
    webPage.setStatus(Status.crawled);
    return webPageRepository.save(webPage);
}
```

爬取页面

刚刚说到，我们需要爬取的页面很多，因此我们使用多线程的方式来运行我们的代码，首先我们来将CrawlThread改写成线程的方式，核心代码如下：

```
public class CrawlerThread implements Runnable {

    @Override
```

```
    public void run() {
        while (true) {
            WebPage webPage = crawlerService.getUnCrawlPage(); // TODO:
            if (webPage == null)
                return; // 拿不到url，说明没有需要爬的url，直接退出
            try {
                if (fetchHtml(webPage))
                    parse(webPage);
            } catch (Exception e) {}
        }
    }
}
```

在CrawlerService中，我们还需要提供一个启动爬虫的入口：

```
public void crawl() throws InterruptedException {
    ExecutorService executorService = Executors.newFixedThreadPool(MAX_
    for(int i = 0; i < MAX_THREADS; i++) {
        executorService.execute(new CrawlerThread(this));
    }
    executorService.shutdown();
    executorService.awaitTermination(Long.MAX_VALUE, TimeUnit.DAYS);
    Ehcache ehcache = cacheManager.getEhcache(cacheName);
    ehcache.removeAll();
}
```

这样，爬虫的所有核心代码就搞定了，先运行CrawlerService.init()方法初始化爬虫队列，之后运行CrawlerService.crawl()就能让我们的爬虫跑起来啦。

提供WEB应用

之前我们提到，我们还要使用Spring MVC，通过Spring MVC，我们就能很方便的提供爬虫管理的API啦。更深入地学习Spring MVC，请大家参考Spring MVC实战入门训练 。

```
@RestController
public class CrawlerController {
```

```
    @Autowired
    private CrawlerService crawlerService;

    @Value("${auth.key}")
    private String key;

    @ModelAttribute
    public void AuthConfig(@RequestParam String auth) throws AccessExce
        if(!key.equals(auth)) {
            throw new AccessException("auth failed");
        }
    }

    @GetMapping("/init")
    public void init() {
        crawlerService.init();
    }

    @GetMapping("/crawl")
    public void crawl() throws InterruptedException {
        crawlerService.crawl();
    }

}
```

启动后，按顺序访问localhost:8080/init&aut... 和localhost:8080/crawl&au... 即可，注意，这里的xxx是自己设置的密码，大家可以在application.properties里自行修改。

最后，我们将所有爬取到的音乐通过页面展示出来：

```
@Controller
public class SongController {

    @Autowired SongRepository songRepository;

    @GetMapping("/songs")
    public String songs(Model model,
            @PageableDefault(size = 100, sort = "commentCount", directi
```

```
        model.addAttribute("songs", songRepository.findAll(pageable));
        return "songs";
    }

}
```

这样，我们的整个爬虫就完成了，整个应用是通过Spring Boot运行的，感兴趣的话可以参考Spring Boot——开发新一代Spring Java应用 。

后续

爬取效率

爬虫爬了两天后，一共爬到了573945条数据，此时数据库访问速度已经变成龟速... 事实证明，对于大型爬虫而言，这样简单粗暴的将数据库作为爬虫队列是不科学的，简单想了一下，我们可以用下列方式来优化爬虫的效率：

- 将webpage表分拆成playlist、album、song三张表，按照数据顺序先爬playlist，再爬album，最后再爬song（甚至将song拆成多张表）
- 由于网易云音乐的各种对象都有id，将id作为索引，提高mysql的效率
- 获取url的时候按照id从小到大获取，获取完一条删除一条
- 既然mysql达不到我们的要求，可以考虑直接将mysql替换掉，使用redis作为爬虫队列

优化的方式有很多种，有些可以借助工具来实现，有些需要考虑具体的业务逻辑。这里我们不具体实现，感兴趣的同学可以自行实现，看看如何优化可以达到最大的效率。

音乐页面访问效率

数据量大了之后，影响的不仅仅是爬虫爬的效率，当然还有访问音乐列表的速度，随意访问一个页面都需要4秒左右。最后，我通过缓存解决了这个问题，具体实现我们也不多讲了，可以参考文章基于Spring的缓存 。加上缓存之后页面访问速度达到了100ms左右。

数据更新

除了爬虫的爬取效率外，还有一个很重要环节，就是数据的更新，评论数据是每天都会变化

的，我们的数据当然也要每天更新。这里，我们使用最简单粗暴的方式，建立一个定时任务（有关定时任务可以参考基于Spring Boot的定时任务 ），在每天的凌晨1点，找到评论数量大于5000的歌曲，将其状态设置为uncrawl(未爬)，启动爬虫即可：

```
@GetMapping("/update")
@Scheduled(cron = "0 1 0 * * ?")
public void update() throws InterruptedException {
    crawlerService.update();
}
```

```
@Async
public void update() throws InterruptedException {
    List<Song> webPages = songRepository.findByCommentCountGreaterThan(
    webPages.forEach(s -> {
        WebPage p = webPageRepository.findOne(s.getUrl());
        p.setStatus(Status.uncrawl);
        webPageRepository.save(p);
    });
    crawl();
}
```

整个站点是用Spring MVC假设的，学习Spring MVC，请大家参考和Spring MVC实战入门训练 和Spring MVC的入门实例 。

进一步阅读

Spring MVC的入门实例 。

更深入地学习Spring MVC，请大家参考Spring MVC实战入门训练 。

欢迎关注天码营微信公众号： TMY-EDU

小编重点推荐：

Spring MVC实战入门训练

Spring Data JPA实战入门训练

Java Web实战训练

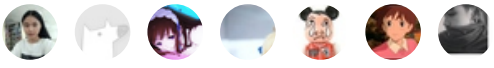
Node.js全栈开发

更多精彩内容请访问天码营网站

「真诚赞赏，手留余香」

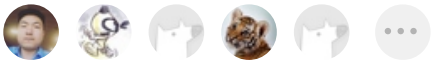
赞赏

7 人赞赏



Java 爬虫 SpringMVC框架

380 分享 举报



文章被以下专栏收录

David教你学Java Web开发

天码营：你从未拥有的实战训练神器

进入专栏

30 条评论

写下你的评论

范志超

很不错

12 天前 1 赞

妞妞的傻小子

6

12 天前

军师

Jsoup好评！

12 天前

David

卧槽我刚要做爬虫，你就搞出来了

12 天前 1 赞

David（作者）回复 **David**

查看对话

你搞一个更牛逼的：）

12 天前 2 赞

杨top1

jsoup

12 天前

Marshall

手动点赞

12 天前

安之 回复 **David** 查看对话

你们俩这ID。。。。

12 天前 2 赞

pig pig

刚才看到一篇说Python爬网易音乐评论的，链接在这里【zhuanlan.zhihu.com/p/22 ... 】。

那个栏目作者说他长得比你帅，请问这是真的吗？

12 天前

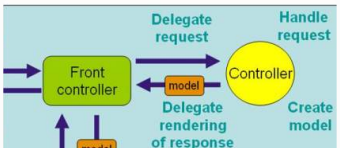
路人甲

看到之前回答的题目了～恩哼～Java写的别有一番风味啊！

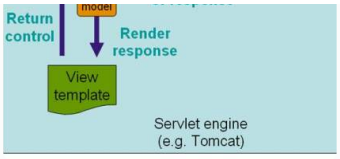
12 天前

1 2 3 下一页

推荐阅读



Spring MVC@RequestMapping 方法所支持的参数类型和返回类型详解



由David发表在天码营使用@RequestMapping标注的处理方法可以拥有非常灵活的方法签名，它支持的方法参数及返回值类型种类极其丰富。大多数参数… 查看全文 >

David · 15 天前 发表于 David教你学Java Web开发

海洋深处 暗潮汹涌：《未知海域》里的壁画和音乐

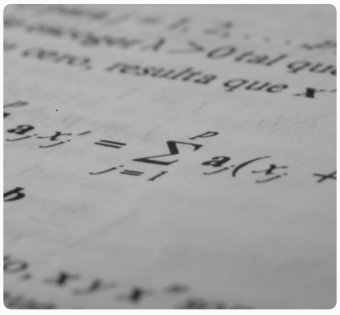
在这个燥热的夏天里，来《未知海域》找片清凉和宁静吧。从壁画、音乐的角度，我来给你说说这款游戏的优美之处。本文可能含有剧透内容，体验游… 查看全文 >

游戏时光VGtime · 1 个月前 · 编辑精选

美国大学生如何看待12大社交网络？

编者按：本文来自微信公众号“峰瑞资本”（微信号：freesvc），授权知乎发布，转载请联系原作者。这可能是关于社交网络最好的一篇分析。这位 … 查看全文 >

峰小瑞 · 7 天前 · 编辑精选 发表于 FREE Thinking



三个应知应会的数学技巧

这是本专栏的第 16 篇日记题外话：上周的行为经济学课，我去找教授要课件（我是旁听，登不上网络课堂），教授说你是不是经常去面试啊，为什么… 查看全文 >

Richard Xu · 4 天前 · 编辑精选 发表于 经济学博士生的日记本