3.18 更新浏览次数

3.18.1 需求

在用户浏览博文时要实现对应博客浏览量的增加。

3.18.2 思路分析

我们只需要在每次用户浏览博客时更新对应的浏览数即可。

但是如果直接操作博客表的浏览量的话，在并发量大的情况下会出现什么问题呢?

如何去优化呢? /

①在应用启动时把博客的浏览量存储到redis中

②更新浏览量时去更新redis中的数据

③每隔10分钟把Redis中的浏览量更新到数据库中

④读取文章浏览量时从redis读取

3.18.3 铺垫知识

3.18.3.1 CommandLineRunner实现项目启动时预处理

如果希望在SpringBoot应用启动时进行一些初始化操作可以选择使用CommandLineRunner来进行处理。

我们只需要实现CommandLineRunner接口，并且把对应的bean注入容器。把相关初始化的代码重新到需要重新的方法中。

这样就会在应用启动的时候执行对应的代码。

```
@Component
public class TestRunner implements CommandLineRunner {
    @Override
    public void run(String... args) throws Exception {
        System.out.println("程序初始化");
    }
}
```

3.18.4 接口设计

请求方式	请求地址	请求头
PUT	/article/updateViewCount/{id}	不需要token请求头

参数

请求路径中携带文章id

响应格式:

```
{
    "code":200,
    "msg":"操作成功"
}
```

3.18.5 代码实现

①在应用启动时把博客的浏览量存储到redis中

新增一个runner包，新建个Runner类实现CommandLineRunner接口，在应用启动时初始化缓存。

```
@Component
public class ViewCountRunner implements CommandLineRunner {

    @Autowired
    private ArticleMapper articleMapper;
    @Autowired
    private RedisCache redisCache;

    @Override
    public void run(String... args) throws Exception {
        //查询博客信息  id  viewCount
        List<Article> articles = articleMapper.selectList(null);
        HashMap<String, Integer> viewCountMap = new HashMap<>();
        for (Article article : articles) {

 viewCountMap.put(article.getId().toString(),article.getViewCount().intValue());
        }
        System.out.println(viewCountMap);
        //存储到redis中
        redisCache.setCacheMap("viewCount",viewCountMap);
    }
}
```

②更新浏览量时去更新redsi中的数据

RedisCache增加方法

```
/**
     * 对应缓存Map中特定key的值进行算数运算
     * @param keyMap 需要操作Map缓存的key
     * @param key    Map中要被运算值对应的key
     * @param val    运算值
     */
    public void addCacheMapValue(String keyMap, String key, Integer val) {
        redisTemplate.opsForHash().increment(keyMap,key,val);
    }
}
```

ArticleController中增加方法更新阅读数

```
    @PutMapping("/updateViewCount/{id}")
    @ResponseBody
    public ResponseResult updateViewCount(@PathVariable("id")Long id){
        return articleService.updateViewCount(id);
    }
```

ArticleService中增加方法

```
ResponseResult updateViewCount(Long id);
```

ArticleServiceImpl中实现方法

```
    @Override
    public ResponseResult updateViewCount(Long id) {
        //更新redis中对应 id的浏览量
        redisCache.addCacheMapValue("viewCount",id.toString(),1);
        return ResponseResult.okResult();
    }
```

③定时任务每隔10分钟把Redis中的浏览量更新到数据库中

3.18.3.2 定时任务

定时任务的实现方式有很多，比如XXL-Job等。但是其实核心功能和概念都是类似的，很多情况下只是调用的API不同而已。

这里就先用SpringBoot为我们提供的定时任务的API来实现一个简单的定时任务，让大家先对定时任务里面的一些核心概念有个大致的了解。

实现步骤

① 使用@EnableScheduling注解开启定时任务功能

我们可以在配置类上加上@EnableScheduling

```
@SpringBootApplication
@MapperScan("com.my.blog.dao")
@EnableScheduling
public class BlogApplication {

    public static void main(String[] args) {
        SpringApplication.run(BlogApplication.class, args);
    }
}
```

② 确定定时任务执行代码，并配置任务执行时间

使用@Scheduled注解标识需要定时执行的代码。注解的cron属性相当于是任务的执行时间。目前可以使用 0/5 * * * * ? 进行测试，代表从0秒开始，每隔5秒执行一次。

注意：对应的bean要注入容器，否则不会生效。

```
@Component
public class TestJob {

    @Scheduled(cron = "0/5 * * * * ?")
    public void testJob(){
        //要执行的代码
        System.out.println("定时任务执行了");
    }
}
```

3.18.3.2.1 cron 表达式语法

cron表达式是用来设置定时任务执行时间的表达式。

很多情况下我们可以用： 在线Cron表达式生成器 来帮助我们理解cron表达式和书写cron表达式。

但是我们还是有需要学习对应的Cron语法的，这样可以更有利于我们书写Cron表达式。

如上我们用到的 0/5 * * * * ? *，cron表达式由七部分组成，中间由空格分隔，这七部分从左往右依次是：

秒（0~59），分钟（0~59），小时（0~23），日期（1-月最后一天），月份（1-12），星期几（1-7,1表示星期日），年份（一般该项不设置，直接忽略掉，即可为空值）

通用特殊字符：, - * / (可以在任意部分使用)

> *

星号表示任意值，例如：

```
* * * * * ?
```

表示“每年每月每天每时每分每秒”。

Article中增加构造方法

```
    public Article(Long id, long viewCount) {
        this.id = id;
        this.viewCount = viewCount;
    }
```

记得加上无参构造，@NoArgsConstructor ，否则会报错

```
@Data
@NoArgsConstructor
public class Article implements Serializable {
```

```
@Component
public class UpdateViewCountJob {

    @Autowired
    private RedisCache redisCache;
    @Autowired
    private IArticleService articleService;

    @Scheduled(cron = "0 0/1 * * * ?")
    public void updateViewCountJob(){
        //获取redis中的浏览量
        Map<String, Integer> viewCountMap = redisCache.getCacheMap("viewCount");
        List<Article> collect = viewCountMap.entrySet().stream()
                .map(entry -> new Article(Long.valueOf(entry.getKey()),
Long.valueOf(entry.getValue())))
                .collect(Collectors.toList());
        //更新到数据库中
        articleService.updateBatchById(collect);
    }
}
```

④读取文章浏览量时从redis读取

```
@Override
    public ResponseResult getArticleDetail(Long id) {
        //根据id查询文章
        Article article = getById(id);
        Integer viewCount = redisCache.getCacheMapValue("viewCount",
article.getId().toString());
        article.setViewCount(viewCount.longValue());
        //转换成VO
        ArticleDetailVo articleDetailVo = new ArticleDetailVo();
        BeanUtils.copyProperties(article, articleDetailVo);
        //根据分类id查询分类名
        Long categoryId = articleDetailVo.getCategoryId();
        Category category = categoryMapper.selectById(categoryId);
        if(category!=null){
            articleDetailVo.setCategoryName(category.getName());
        }
```

```
        //封装响应返回
        return ResponseResult.okResult(articleDetailVo);
    }
```