

目录

第1章 基础

01 开篇词：为什么学习本专栏

02 String、Long 源码解析和面试题

03 Java 常用关键字理解

04 Arrays、Collections、Objects 常用方法源码解析

第2章 集合

05 ArrayList 源码解析和设计思路

06 LinkedList 源码解析

07 List 源码会问哪些面试题

08 HashMap 源码解析

09 TreeMap 和 LinkedHashMap 核心源码解析

10 Map源码会问哪些面试题

11 HashSet、TreeSet 源码解析

12 彰显细节：看集合源码对我们实际工作的帮助和应用

13 差异对比：集合在 Java 7 和 8 有何不同和改进

14 简化工作：Guava Lists Maps 实际工作运用和源码

第3章 并发集合类

15 CopyOnWriteArrayList 源码解析和设计思路

16 ConcurrentHashMap 源码解析和设计思路

17 并发 List、Map源码面试题

18 场景集合：并发 List、Map的应用

40 打动面试官：线程池流程编排中的运用实战

更新时间：2019-11-22 09:39:59



“没有智慧的头脑，就像没有蜡烛的灯笼。”——托尔斯泰

引导语

在线程池的面试中，面试官除了喜欢问 ThreadPoolExecutor 的底层源码外，还喜欢问你有没有在实际的工作中用过 ThreadPoolExecutor，我们在并发集合类的《场景集合：并发 List、Map 的应用场景》一文中说过一种简单的流程引擎，如果没有看过的同学，可以返回去看一下。

本章就在流程引擎的基础上运用 ThreadPoolExecutor，使用线程池实现 SpringBean 的异步执行。

1 流程引擎关键代码回顾

《场景集合：并发 List、Map 的应用场景》文中流程引擎执行 SpringBean 的核心代码为：

```
// 批量执行 Spring Bean
private void stagleInvoke(String flowName, StageEnum stage, FlowContent content) {
    List<DomainAbilityBean>
        domainAbillitys =
            FlowCenter.flowMap.getOrDefault(flowName, Maps.newHashMap()).get(stage);
    if (CollectionUtils.isEmpty(domainAbillitys)) {
        throw new RuntimeException("找不到该流程对应的领域行为" + flowName);
    }
    for (DomainAbilityBean domainAbility : domainAbillitys) {
        // 执行 Spring Bean
        domainAbility.invoke(content);
    }
}
```

目录	2 异步执行 SpringBean
第1章 基础	从上述代码中，我们可以看到所有的 SpringBean 都是串行执行的，效率较低，我们在实际业务中发现，有的 SpringBean 完全可以异步执行，这样既能完成业务请求，又能减少业务处理的 rt，对于这个需求，我们条件反射的有了两个想法：
01 开篇词：为什么学习本专栏	<div>1. 需要新开线程来异步执行 SpringBean，可以使用 Runnable 或者 Callable；</div> <div>2. 业务请求量很大，我们不能每次来一个请求，就开一个线程，我们应该让线程池来管理异步执行的线程。</div>
02 String、Long 源码解析和面试题	于是我们决定使用线程池来完成这个需求。
03 Java 常用关键字理解	3 如何区分异步的 SpringBean
04 Arrays、Collections、Objects 常用方法源码解析	我们的 SpringBean 都是实现 DomainAbilityBean 这个接口的，接口定义如下：
第2章 集合	<pre>public interface DomainAbilityBean { /** * 领域行为的方法入口 */ FlowContent invoke(FlowContent content); }</pre>
05 ArrayList 源码解析和设计思路	从接口定义上来看，没有预留任何地方来标识该 SpringBean 应该是同步执行还是异步执行，这时候我们可以采取注解的方式，我们新建一个注解，只要 SpringBean 上有该注解，表示该 SpringBean 应该异步执行，否则应该同步执行，新建的注解如下：
06 LinkedList 源码解析	<pre>/** * 异步 SpringBean 执行注解 * SpringBean 需要异步执行的话，就打上该注解 *author wenhe *date 2019/10/7 */ @Target(ElementType.TYPE)// 表示该注解应该打在类上 @Retention(RetentionPolicy.RUNTIME) @Documented public @interface AsyncComponent { }</pre>
07 List 源码会问哪些面试题	接着我们新建了两个 SpringBean，并在其中一个 SpringBean 上打上异步的注解，并且打印出执行 SpringBean 的线程名称，如下图：
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

<div><div>← 慕课专栏</div><div>面试官系统精讲Java源码及大厂真题 / 40 打面试官：线程池流程编排中的运用实战</div></div>	
目录	的线程的名称。
第1章 基础	4 mock 流程引擎数据中心
01 开篇词：为什么学习本专栏	《场景集合：并发 List、Map 的应用场景》一文中，我们说可以从数据库中加载出流程引擎需要的数据，此时我们 mock 一下，mock 的代码如下：
02 String、Long 源码解析和面试题	<pre>@Component public class FlowCenter { /** * flowMap 是共享变量，方便访问 */ public static final Map<String, Map<StageEnum, List<DomainAbilityBean>>> flowMap = Maps.newConcurrentMap(); /** * PostConstruct 注解的意思就是 * 在容器启动成功之后，初始化 flowMap */ @PostConstruct public void init() { // 初始化 flowMap mock Map<StageEnum, List<DomainAbilityBean>> stageMap = flowMap.getOrDefault("flow1",Maps.newConcurrentMap()); for (StageEnum value : StageEnum.values()) { List<DomainAbilityBean> domainAbilities = stageMap.getOrDefault(value, Lists.newArrayList()); if(CollectionUtils.isEmpty(domainAbilities)){ domainAbilities.addAll(ImmutableList.of(ApplicationContextHelper.getBean(BeanOne.class), ApplicationContextHelper.getBean(BeanTwo.class))); stageMap.put(value,domainAbilities); } } flowMap.put("flow1",stageMap); // 打印出加载完成之后的数据结果 log.info("init success,flowMap is {}", JSON.toJSONString(flowMap)); } }</pre>
03 Java 常用关键字理解	5 新建线程池
04 Arrays、Collections、Objects 常用方法源码解析	在以上操作完成之后，只剩下最后一步了，之前我们执行 SpringBean 时，是这行代码：domainAbility.invoke(content);
第2章 集合	现在我们需要区分 SpringBean 是否是异步的，如果是异步的，丢到线程池中去执行，如果是同步的，仍然使用原来的方法进行执行，于是我们把这些逻辑封装到一个工具类中，工具类如下：
05 ArrayList 源码解析和设计思路	<pre>/** * 组件执行器 * author wenhe * date 2019/10/7 */ public class ComponentExecutor { // 我们新建了一个线程池</pre>
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

目录	
第1章 基础	
01 开篇词：为什么学习本专栏	
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

```
// 如果 SpringBean 上有 AsyncComponent 注解，表示该 SpringBean 需要异步执行，就丢到线程池中执行。  
public static final void run(DomainAbilityBean component, FlowContent content) {  
    // 判断类上是否有 AsyncComponent 注解  
    if (AnnotationUtils.isAnnotationPresent(AsyncComponent.class, AopUtils.getTargetClass(component))) {  
        // 提交到线程池中  
        executor.submit(() -> { component.invoke(content); });  
        return;  
    }  
    // 同步 SpringBean 直接执行。  
    component.invoke(content);  
}
```

我们把原来的执行代码替换成使用组件执行器执行，如下图：

6 测试

以上步骤完成之后，简单的流程引擎就已经完成了，我们简单地在项目启动的时候加上测试，代码如下：

```
1 package demo;  
2  
3 import ...  
4  
5 /**  
6  * DemoApplication  
7  * author wenhe  
8  * date 2019/9/24  
9  */  
10  
11 @SpringBootApplication(scanBasePackages = {"demo"})  
12 public class DemoApplication {  
13  
14     public static void main(String[] args) {  
15         SpringApplication.run(DemoApplication.class);  
16         ApplicationContextHelper.getBean(FlowStart.class)  
17             .start( flowName: "flow1",new FlowContent());  
18     }  
19 }  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29
```

项目启动类上加上测试代码

更严谨的做法，是会写单元测试来测试流程引擎，为了快一点，我们直接在项目启动类上加上了测试代码。

运行之后的关键结果如下：

← 慕课专栏

面试官系统精讲Java源码及大厂真题 / 40 打面试官：线程池流程编排中的运用实战

目录

第1章 基础

01 开篇词：为什么学习本专栏

02 String、Long 源码解析和面试题

03 Java 常用关键字理解

04 Arrays、Collections、Objects 常用方法源码解析

第2章 集合

05 ArrayList 源码解析和设计思路

06 LinkedList 源码解析

07 List 源码会问哪些面试题

08 HashMap 源码解析

09 TreeMap 和 LinkedHashMap 核心源码解析

10 Map源码会问哪些面试题

11 HashSet、TreeSet 源码解析

12 彰显细节：看集合源码对我们实际工作的帮助和应用

13 差异对比：集合在 Java 7 和 8 有何不同和改进

14 简化工作：Guava Lists Maps 实际工作运用和源码

第3章 并发集合类

15 CopyOnWriteArrayList 源码解析和设计思路

16 ConcurrentHashMap 源码解析和设计思路

17 并发 List、Map源码面试题

18 场景集合：并发 List、Map的应用

```
[main] demo.three.now.FlowCenter : init success,nowmap is { nowmap : { PARAM_VALID :[{},{}] } }
o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
[main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
[main] demo.DemoApplication : Started DemoApplication in 5.377 seconds (JVM running for 10.011s)
[main] demo.three.flow.BeanOne : BeanOne is run,thread name is main
[main] demo.three.flow.BeanOne : BeanOne is run,thread name is main
[pool-1-thread-1] demo.three.flow.BeanTwo : BeanTwo is run,thread name is pool-1-thread-1
[main] demo.three.flow.BeanOne : BeanOne is run,thread name is main
[pool-1-thread-2] demo.three.flow.BeanTwo : BeanTwo is run,thread name is pool-1-thread-2
[pool-1-thread-3] demo.three.flow.BeanTwo : BeanTwo is run,thread name is pool-1-thread-3
[main] demo.three.flow.BeanOne : BeanOne is run,thread name is main
[pool-1-thread-4] demo.three.flow.BeanTwo : BeanTwo is run,thread name is pool-1-thread-4
```

从运行结果中，我们可以看到 BeanTwo 已经被多个不同的线程异步执行了。

总结

这是一个线程池在简单流程引擎上的运用实战，虽然这个流程引擎看起来比较简单，但在实际工作中，还是非常好用的，大家可以把代码拉下来，自己尝试一下，调试一下参数，比如当我新增 SpringBean 的时候，流程引擎的表现如何。

← 39 经验总结：不同场景，如何使用线程池

41 突破难点：如何看 Lambda 源码 →

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

！

目前暂无任何讨论

千学不如一看，千看不如一练

www.imooc.com/read/47/article/882

5/5