

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

18 场景集合：并发 List、Map的应用场景

更新时间：2019-10-08 16:45:03



“
一个不注意小事情的人，永远不会成功大事业。
——戴尔·卡耐基”

引导语

并发 List、Map 使用最多的就是 CopyOnWriteArrayList 和 ConcurrentHashMap，在考虑 API 时，我们也无需迟疑，这两个并发类在安全和性能方面都很好，我们都可以直接使用。

并发的场景很多，但归根结底其实就是共享变量被多个线程同时访问，也就是说 CopyOnWriteArrayList 或 ConcurrentHashMap 会被作为共享变量，本节我们会以流程引擎为案例，现身说法，增加一下大家的工作经验积累。

流程引擎在实际工作中经常被使用，其主要功能就是对我们需要完成的事情，进行编排和组装，比如在淘宝下单流程中，我们一共会执行 20 个 Spring Bean，流程引擎就可以帮助我们调起 20 个 Spring Bean，并帮助我们去执行，本文介绍的重点在于如何使用 Map + List 来设计流程引擎的数据结构，以及其中需要注意到的线程安全的问题。

1 嵌套 Map，简单流程引擎

市面上有很多流程引擎，比如说 Activiti、Flowable、Camunda 等等，功能非常齐全，但我们本小节只实现一种最简单的流程引擎，只要能对我们需要完成的事情进行编排，并能依次的调用就行。

1.1 流程引擎设计思路

我们认为每个流程都会做 4 个阶段的事情，阶段主要是指在整个流程中，大概可以分为几个大的步骤，每个阶段可以等同为大的步骤，分别如下：

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

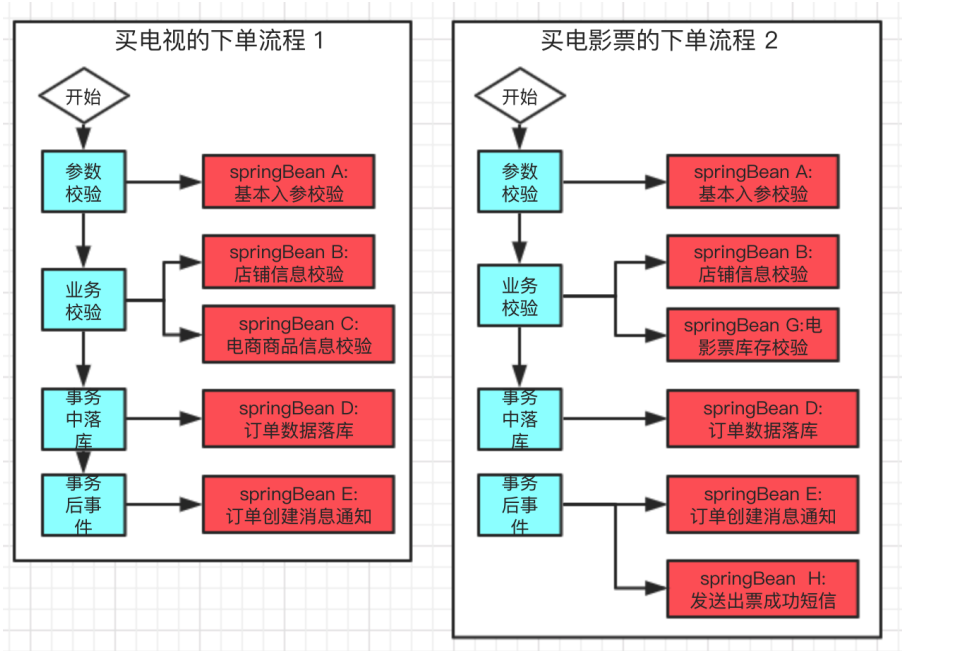
3. 事务中落库，主要把数据落库，控制事务；
4. 事务后事件，我们在数据落库，事务提交之后，可能会做一些其他事情，比如说发消息出来等等。

以上每个大的阶段，都会做一些粒度较细的事情，比如说业务校验，我们可能会对两个业务对象进行校验，那么此时业务校验阶段就会做两件事情，每件具体的事情，我们叫做领域行为，在实际项目中，一个领域行为一般都是一个 Spring Bean。

综上所述，流程引擎嵌套数据结构就是：流程 -> 阶段 -> 领域行为，前者对应后者，都是一对一或者一对多的关系。

我们以在淘宝上买东西时，下单为例，下单指的是我们在淘宝选择好了商品和优惠券后，点击购买按钮时触发的动作。

为了方便举例，我们假设在淘宝上买电视和电影票，在后端，会分别对应着两个下单流程，我们画图示意一下：



上图中，左右两个黑色长方形大框代表着两个流程，流程下面有多个阶段，阶段用蓝色表示，每个阶段下面有多个领域行为，用红色表示。

可以看到两个流程中，都包含有四个阶段，阶段都是相同的，但每个阶段中的领域行为，有的相同，有的却是特有的。

三个概念，每个概念层层嵌套，整体组装起来，用来表示一个流程，那么这个数据结构，我们应该如何表示呢？

使用 Map + List 即可！

1.2 数据结构的定义

流程的数据结构定义分成两步：

1. 定义出阶段、领域行为基础概念；

← 慕课专栏	面试官系统精讲Java源码及大厂真题 / 18 场景集合：并发 List、Map的应用场景
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的应用场景	

首先给阶段定义一个枚举，如下 StageEnum 代表流程中的阶段或步骤：

```
public enum StageEnum {  
    PARAM_VALID("PARAM_VALID", "参数校验"),  
  
    BUSINESS_VALID("BUSINESS_VALID", "业务校验"),  
  
    IN_TRANSACTION("IN_TRANSACTION", "事务中落库"),  
  
    AFTER_TRANSACTION("AFTER_TRANSACTION", "事务后事件"),  
    ;  
  
    private String code;  
    private String desc;  
  
    StageEnum(String code, String desc) {  
        this.code = code;  
        this.desc = desc;  
    }  
}
```

领域行为我们无需定义，目前通用的技术框架都是 Spring Boot，领域行为都是 Spring Bean，为了简单起见，我们给领域行为定义了一个接口，每个领域行为都要实现这个接口，方便我们编排，接口如下：

```
/**  
 * 领域行为  
 * author wenhe  
 * date 2019/8/11  
 */  
public interface DomainAbilityBean {  
  
    /**  
     * 领域行为的方法入口  
     */  
    FlowContent invoke(FlowContent content);  
  
}
```

接着我们使用 Map + List 来定义流程，定义如下：

```
/**  
 * 第一个 key 是流程的名字  
 * 第二个 map 的 key 是阶段，为 StageEnum 枚举，值为多个领域行为的集合  
 */  
Map<String,Map<StageEnum,List<DomainAbilityBean>>>> flowMap
```

至此，我们定义出了，简单流程引擎的数据结构，流程引擎看着很复杂，利用 Map + List 的组合，就巧妙的定义好了。

2 容器初始化时，本地缓存使用

我们定义好 Map 后，我们就需要去使用他，我们使用的大体步骤为：

1. 项目启动时，把所有的流程信息初始化到 flowMap(刚刚定义的流程的数据结构叫做 flowMap) 中去，可能是从数据库中加载，也可能是从 xml 文件中加载；

← 慕课专栏	面试官系统精讲Java源码及大厂真题 / 18 场景集合：并发 List、Map 的应用场景
目录	2.1 初始化
第1章 基础	以上两步最为关键的点就是 flowMap 必须是可以随时访问到的，所有我们会把 flowMap 作为共享变量使用，也就是会被 static final 关键字所修饰，我们首先来 mock 一下把所有信息初始化到 flowMap 中去的代码，如下：
01 开篇词：为什么学习本专栏	<pre>@Component public class FlowCenter { /** * flowMap 是共享变量，方便访问，并且是 ConcurrentHashMap */ 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，执行 init 方法，初始化 flowMap */ @PostConstruct public void init() { // 初始化 flowMap，可能是从数据库，或者 xml 文件中加载 map } }</pre>
02 String、Long 源码解析和面试题	以上代码，关键地方在于三点：
03 Java 常用关键字理解	1. flowMap 被 static final 修饰，是个共享变量，方便访问；
04 Arrays、Collections、Objects 常用方法源码解析	2. flowMap 是 ConcurrentHashMap，所以我们所有的操作都无需加锁，比如我们在 init 方法中，对 flowMap 进行初始化，就无需加锁，因为 ConcurrentHashMap 本身已经保证了线程安全；
第2章 集合	3. 这里我们初始化的时机是在容器启动的时候，在实际的工作中，我们经常在容器启动的时候，把不会经常发生变动的数据，放到类似 List、Map 这样的共享变量中，这样当我们频繁要使用的时候，直接从内存中读取即可。
05 ArrayList 源码解析和设计思路	2.2 使用
06 LinkedList 源码解析	那我们实际使用的时候，只需要告诉 flowMap 当前是那个流程的那个阶段，就可以返回该流程该阶段下面的所有领域行为了，我们写了一个流程引擎使用的工具类入口，如下：
07 List 源码会问哪些面试题	<pre>// 流程引擎对外的 API public class FlowStart { /** * 流程引擎开始 * * @param flowName 流程的名字 */ public void start(String flowName, FlowContent content) { invokeParamValid(flowName, content); invokeBusinessValid(flowName, content); invokeInTramsactionValid(flowName, content); invokeAfterTramsactionValid(flowName, content); } // 执行参数校验</pre>
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的应用场景	

<div>← 慕课专栏</div> <div>面试官系统精讲Java源码及大厂真题 / 18 场景集合：并发 List、Map 的应用场景</div>		
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的应用场景		

```
// 执行业务校验
private void invokeBusinessValid(String flowName, FlowContent content) {
    stageInvoke(flowName, StageEnum.BUSINESS_VALID, content);
}

// 执行事务中
private void invokeInTramsactionValid(String flowName, FlowContent content) {
    stageInvoke(flowName, StageEnum.IN_TRANSACTION, content);
}

// 执行事务后
private void invokeAfterTramsactionValid(String flowName, FlowContent content) {
    stageInvoke(flowName, StageEnum.AFTER_TRANSACTION, content);
}

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

}
```

从代码中可以看到，我们在流程引擎的入口，只要根据参数校验、业务校验、事务中、事务后四个阶段，从 flowMap 中得到领域行为的集合，然后对领域行为进行顺序执行即可。

我们在使用时，直接使用上述类的 start 方法即可。

当然以上演示的流程引擎只是一个大的框架，还有很多地方需要改进的地方，比如如何找到 flowName，如何初始化 flowMap，但这些都不是本节重点，本节主要想通过流程引擎案例来说明几点：

1. 把 List 和 Map 作为共享变量非常常见，就像咱们这种项目启动时，从数据库中把数据捞出来，然后封装成 List 或 Map 的结构，这样做的优点就是节约资源，不用每次用的时候都去查数据库，直接从内存中获取即可；
2. 并发场景下，我们可以放心的使用 CopyOnWriteArrayList 和 ConcurrentHashMap 两个并发类，首先用 static final 对两者进行修饰，使其成为共享变量，接着在写入或者查询的时候，无需加锁，两个 API 内部已经实现了加锁的功能了；
3. 有一点需要澄清一下，就是 CopyOnWriteArrayList 和 ConcurrentHashMap 只能作为单机的共享变量，如果是分布式系统，多台机器的情况下，这样做不行了，需要使用分布式缓存了。

3 总结

本节内容，以流程引擎为例，说明了如何使用 Map + List 的嵌套结构设计流程引擎，以及在并发情况下，如何安全的使用 List 和 Map。

本案列是高并发项目的真实案例，感兴趣的同学可以在此流程引擎框架基础上进行细节补充，实现可运行的流程引擎。

<div><div>← 慕课专栏</div><div>面试官系统精讲Java源码及大厂真题 / 18 场景集合：并发 List、Map的应用场景</div></div>	
目录	
第1章 基础	精选留言 2
01 开篇词：为什么学习本专栏	<div>欢迎在这里发表留言，作者筛选后可公开显示</div> <div><div>👍 0</div><div>回复</div><div>2019-10-09</div></div>
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	<div><div>👍 0</div><div>回复</div><div>2019-10-09</div></div> <div><div>文贺 回复 🍻</div><div>棒，我觉得蛮好的哈，我虽然不是强化记忆方面的专家，但也看过这方面的文章，这应该叫做加深理解，刻意练习，保持加油，如果能写写demo，多多debug就更好了。</div><div>回复</div><div>2019-10-10 22:59:01</div></div>
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	<div><div>👍 0</div><div>回复</div><div>2019-10-08</div></div> <div><div>文贺</div><div>demo.three.flow.* 目录下的代码可能和文章的不太一样，因为这个流程引擎的例子，我们会在第七章线程池的章节中再次实践，所以 GitHub 的代码可能和文章不太一样，GitHub 的代码是加上线程池内容的代码，文章中的代码是没有加线程池内容的代码。</div></div>
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	<div>千学不如一看，千看不如一练</div>
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	