基于 Spring AOP 的有序查找算法的重构

曹大有　周 兵　（郧阳师范高等专科学校 计算机科学系 湖北丹江口　442700）

摘要：数据结构有很多查找方法，大体上可分为有序查找和无序查找。但有序查找在查找效率上要好一些。Spring 是 J2EE 的轻量级架构，它通过反向控制和依赖注射的能力来装配和管理任何种类的应用程序对象（Bean）。文中通过 Spring AOP 框架的前置建议对有序查找进行了重构，使得不关数据是否有序都可以按有序序列进行查找，而且这种重构是无侵入性的。

关键词：反向控制　Bean　Spring　依赖注射

查找就是从一些数据中寻找出一个特定的值，常用的数据查找方法包括：线性查找、折半查找、费氏查找等等。线性查找可以用在未经过排序的数据中进行查找，但它的查找效率不高，而其它的查找方法则需要先将数据排序完成后，再按一定的规则进行查找工作。我们能否将查找算法进行统一，即不论数据是否有序，我们都按有序数据来进行查找，如都用折半查找法来进行特定数据的查找。这种想法对有序数据肯定可行，但对无序数据就有点不可以。下面我们就用 Spring AOP 框架来实现我们的这种想法。

Spring 框架是当今流行的 J2EE 应用程序开发的开源产品，它的核心代码均来自于真实的项目。Spring 对各种 J2EE 框架提供了无缝集成、即插即用和非侵入性的功能。而 Spring 框架的非侵入性是由 AOP 技术来支持的。AOP 即面向方面的编程技术，它对面向对象软件应用中存在的许多问题，提出了更好的解决方案。它的理念就是将那些与业务无关，却为业务模块所共同调用的逻辑封装起来，减少代码重复模块，降低模块间的耦合度，从而帮助开发人员提高编程效率。Spring 通过 IoC 实现 AOP，Spring AOP 也是一个独立的 API，可以通过编程的方式在 Spring 之外单独使用。

1　Spring AOP 功能简介

AOP 联盟所给出的 AOP 接口非常抽象，主要集中在 org. aopalliance. aop 和 org. aopalliance. intercept 这两个包内。包内的每个接口都代表了一种 AOP 技术的契约，如建议（Advice）、连接点（Joinpoint）。此外也出现了一些新的接口，如 MethodInvocation 和方法拦截器（MethodInterceptor），这些是围绕方法调用的层面。Spring AOP 实现了 Advice、MethodInterceptor 和 MethodInvocation 接口。其中 Advice 是 AOP 执行的具体逻辑。MethodInvocation 是一系列 Joinpoint（连接点），它主要是使方法在调用期间可对它的代理进行访问。MethodInterceptor 是 Advice 的子接口，在 Spring 中它是比较特殊的一种 Advice，主要功能是提供 Around Advice（环绕拦截建议）。Spring AOP 采用纯 Java 语言实现，所以无需要特别的编译过程。它不仅提供了 AOP 基础框架。还提供了很多现成的方面实现。它支持对方法调用的各类建议（Advice）：如前置建议、后置建议、环绕拦截建议、抛出建议。支持引介和混入。可通过编程使用代理，也可以通过声明配置。使用 Spring AOP 的优势是：由于 Spring AOP 与 Spring IoC 容器的无缝整合，AOP 组件可享受 Spring 提供的一切优势，Advice 可通过普通的 Bean 来定义，Advice 和 Pointcut 也可由 Spring IoC 管理。当然 Spring AOP 实现的目标不是完善 AOP 的实现，而是提供一个与 Spring IoC 紧密整合的 AOP 框架。

2　基于 Spring AOP 的折半查找算法的重构

我们首先来创建一个实现折半查找算法的 Bean 类：bsearch。在该 Bean 类中，我们用 data 代表数据序列，counter 记录查找成功的查找次数，对 data 属性定义相应的 set/get 方法，而 counter 属性只给出 get 方法。方法 public boolean BinarySearch（int KeyValue）则用来在 data 中查找数据 KeyValue，成功返回 true，否则返回 false。bsearch 的实现如下所示：

```
class bsearch{
 private int[ ] data; private int counter =1;//赋初值
 public void setData(int [ ] data){this. data = data;}//给数组赋值
 public int[ ] getData( ){return data;}//返回数组的值
 public int getCounter( ){return counter;}//返回查找成功的次数
 public boolean BinarySearch(int KeyValue){//折半查找算法
  int Left =0; int Right = data. length -1; int Middle; //赋初值
  while(Left < =Right){//设定循环条件
   Middle = (Left + Right)/2;//求出中间值
   if(KeyValue < data[Middle]) Right = Middle -1;//设定下半区
   else if(KeyValue > data[Middle]) Left = Middle +1;//设定上半区
   else if(KeyValue = = data[Middle]){//查找成功返回 true
    System. out. println(" data[ " + Middle +" ] =" +data[Middle]);
    return true;}
   counter + +;}//记录查找次数
  return false;}}//不成功返回 false
```

上面给出了折半查找算法的 Bean 类：bsearch。当然使用 BinarySearch(int KeyValue)的前提条件是 data 中的数据必须是有序的，但这里并没有保证 data 中的数据一定是有序的。我们可以通过创建一个 AOP 的前置建议(Before Advice)来实现在查找之前先将 data 中的数据排序。在 Spring AOP 中，前置建议是一种非常简单的建议类型，使用它可以在一个方法调用之前织入自定义的相关处理。目前 AOP 的 Advice 接口和 Spring AOP 的 BeforeAdvice 接口都只是标识性接口，它们并没有给出任何方法的定义。要创建前置建议可以实现 MethodBeforeAdvice 接口，代码如下所示：

```
public interface MethodBeforeAdvice extends BeforeAdvice{
 void before(Method method, Object[ ] args, Object target) throws Throwables;
 }
```

参数 method 代表将要调用的方法体本身，args 代表所要传入的参数，target 是方法调用的目标对象。这里 target 指的就是 bsearch 类，在下面的前置建议中，我们先通过 bsearch 类的 getData( )方法求出要排序的数组属性 data，然后用任何一种排序方法对它进行排序，之后再调用 bsearch 类的 setData( )方法将排序后的数组传回去供查找方法使用。这样一来，不论 bsearch 类 data 属性中的数据是否有序，保证在查找之前一定是有序的。前置建议 bsearchBeforeAdvice 的代码如下所示：

```
public class bsearchBeforeAdvice implements MethodBeforeAdvice{//前置建议
 public void before(Method method, Object[ ] args, Object target) throws Throwable{
  int[ ]data = ((bsearch)target). getData( );//求出需要排序的数组
  int i,j,k;//循环计数变量
  int MinValue;//最小值变量
  int indexMin;//最小值下标变量
  int Temp;//暂存变量
  System. out. println(" Before Select Sorting:");
  for(i =0;i < data. length;i + +) System. out. print(" " +data[i] +" ");
  System. out. println(" ");
  for(i =0;i < data. length;i + +){//选择排序
   MinValue =32767; indexMin =0;//设定最小值和排序起点
   for(j =i;j < data. length;j + +){//求第 i 个最小值并交换位置
   if(data[j] < MinValue){MinValue = data[j]; indexMin =j;}}
   Temp = data[i]; data[i] = data[indexMin]; data[indexMin] =Temp;
   System. out. print(" Current sorting result:");
   for(k =0;k < data. length;k + +) System. out. print(" " +data[k] +" ");
   System. out. println(" ");//输出排序后的数组内容
```

```
}
((bsearch)target).setData(data);}}//重置数组内容
```

默认情况下，前置建议会作用于目标对象的所有方法调用，而上面的前置建议仅需要在方法 BinarySearch(int KeyValue)之前织入，所以我们还要定义切入点(Pointcut)，用于匹配目标对象及相关的方法。这里只需要扩展 StaticMethodMatcherPointcut 类，并且在 bsearchStaticPointcut 类中明确指出该前置建议只用于 bsearch 类的 BinarySearch 方法。

```
public class bsearchStaticPointcut extends StaticMethodMatcherPointcut{
 public boolean matches(Method method,Class cls){//匹配指定的方法
  return method.getName().equals("BinarySearch");}
 public ClassFilter getClassFilter(){//匹配指定的类
  return new ClassFilter(){
   public boolean matches(Class cls){
    return cls==bsearch.class;}};}}
```

3 通过 ProxyFactory 进行验证

上面我们定义了实现折半查找的 Bean 类 bsearch，前置建议 bsearchBeforeAdvice 和切入点 bsearchStaticPointcut。实际使用时，首先通过前置建议和切入点创建一个缺省顾问：DefaultPointcutAdvisor 对象，然后再通过 ProxyFactory 类对象进行相应的设置即可，具体过程如下所示：

```
bsearch target=new bsearch();//创建目标对象
bsearchBeforeAdvice advice=new bsearchBeforeAdvice();//创建前置建议对象
Pointcut pc=new bsearchStaticPointcut();//创建切入点对象
Advisor bsearchAdvisor=new DefaultPointcutAdvisor(pc,advice);//创建顾问
ProxyFactory factory=new ProxyFactory();//创建工厂代理
factory.setTarget(target);//设置目标对象
factory.addAdvisor(bsearchAdvisor);//设置顾问
return (bsearch)factory.getProxy();//返回代理
```

经过以上设置之后，我们就可以对任何数组通过折半查找进行特定数据的查找，不论该数组中的元素是否有序都可以进行。具体程序见 BsearchTestProxyFactory.java 所示。

4 总结与讨论

在用 JDK 执行时，一定要在 classpath 路径属性中指定出 spring.jar(Spring 框架开发包)、commons-logging-api.jar(在 Apache Tomcat 安装目录下 bin 子目录中)、cglib-nodep-2.1_3.jar(支持 CGLIB 代理)文件的存放路径。本文的 JDK 版本为 jdk-1_5_0_04。前置建议中的排序算法这里用的是选择排序，可以根据实际情况换用其它排序算法，查找算法用的是折半查找，当然也可以用递归查找或费氏查找等其它有序查找算法。另外在验证时是通过 Spring AOP 提供的 ProxyFactory 编程设置的，也可以通过 ProxyFactoryBean 进行声明性设置，配置文件见 beforeadvice.xml 所示，验证程序见 BsearchTestProxyFactoryBean.java 所示。

参考文献

1 (美)Rob Harrop、Jan Machacek 著，Redsaga 翻译小组译，Spring 专业开发指南[M]，北京：电子工业出版社，2006.

2 黄国瑜、叶乃菁编著，数据结构(Java 语言版)[M]，北京：清华大学出版社，2002.