一种基于 J2EE、Spring 和 Hibernate 的轻量级 EAI 构架①

A lightweight EAI Architecture Based on J2EE, Spring and Hibernate

王卫平 王松涛 王名茗 （合肥 中国科学技术大学信息管理与决策科学系 230026）

摘要：本文结合 J2EE、Spring 与 Hibernate 的特点，提出一种以 J2EE 为基本框架、以 Spring 作为业务方法层组件、以 Hibernate 作为持久层组件的面向中小企业的轻量级 EAI 构架，并对该构架主要环节的关键技术实现作了分析。

关键词：J2EE Spring Hibernate EAI

1 引言

EAI[3,5]是研究将不同的应用程序和数据集成在一起的过程，从而在不对已有的应用程序作过多修改的情况下，实现数据共享和业务流程的集成。通常，在 EAI 中包括如下的重要过程：平台（硬件）集成、数据级集成、业务逻辑集成、表示层集成[5]。本文主要关注业务逻辑集成。业务逻辑集成是两个集成阶段的统称——应用接口集成和业务方法集成。应用接口集成主要解决关联到现有应用的功能性复用的技术方面的问题，通过使用现有应用程序所列的、或者是后来加上的 API 来完成。业务方法集成是在应用接口集成的基础上开发高层业务接口，也就是业务服务，它基于新的集成体系结构，提出了带有明确需求的集成信息系统。传统的基于 J2EE 的业务逻辑层集成通常采用 EJB[4,7]实现。EJB 组件设计用于封装业务逻辑，它方便了应用程序开发者，使他们不需要担心许多系统级问题，如事务、安全性、可扩展性、并发性、通信、资源管理、持续性、出错处理以及操作环境无关性。EJB 提供了一种现代的组件体系结构，以定义和编辑的方式提供了对上述服务的支持，这使得业务组件的开发成为一件相对容易的工作。因此，EJB 在业务逻辑集成中发挥着重要的作用。然而，EJB 并没有解决所有的问题[6]。EJB 是解决企业级分布式应用的重量级组件，存在性能差、测试困难、难以维护等多方面的问题。由于 EJB 的内聚性较强，许多工作，比如数据持久层管理、事务管理、生命周期管理都交给了 EJB 容器管理，内聚性的白盒特征使我们必须放弃一部分可控性而去信任容器能力。EJB 具有高侵入和紧耦合的客观缺点，同时购买 EJB 容器，价格昂贵。因此，对于中小企业，我们应该采用某种开源的轻量级构架取代 EJB 处理业务逻辑。

2 J2EE、Spring 和 Hibernate 技术简介

J2EE[4,5]（Java2 Platform Enterprise Edition）是由 Sun 公司提出来的分布式企业计算平台，它利用 Java 2 技术，提供了一系列的中间件服务来简化诸多企业解决方案的开发、部署和管理相关的复杂问题。J2EE 最重要的一个方面是对运行环境基础框架的抽象，主要通过容器（Container）来实现。容器的最主要的职责是提供一个组件可以运行的环境，它负责装载组件，并使得客户端程序能进行远程调用，同时还提供组件池和生命周期管理、事务协调、数据存储与访问控制等服务。J2EE 之所以可以方便的作为 EAI 平台，是因为 J2EE 平台建立在多层应用模型之上，它引入了以下几层：客户层、Web 组件层、业务逻辑层、EIS 层。

Spring[1,8]是一个解决了许多在 J2EE 开发中常见问题的强大框架。Spring 提供了管理业务对象的一致方法，并可通过对接口编程而不是对类编程去实现。Spring 的架构基础是基于使用 JavaBean 属性的 IOC（Inversion of Control）容器[8]，这使得 Spring 在使用 IOC 容器作为构建所有架构层的完整解决方案方面是独一无二的。

Hibernate[2]是一个开放源代码的 ORM（Object/Relational Mapping）框架，它对对象/关系映射（ORM）

① 基金项目：国家 863 高技术研究发展计划项目资助（编号：2003AA103710）

进行了很好的对象封装，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP[4]（Container – managed persistence），完成数据持久化的重任。

3 基于 J2EE、Spring 与 Hibernate 的轻量级企业应用集成构架

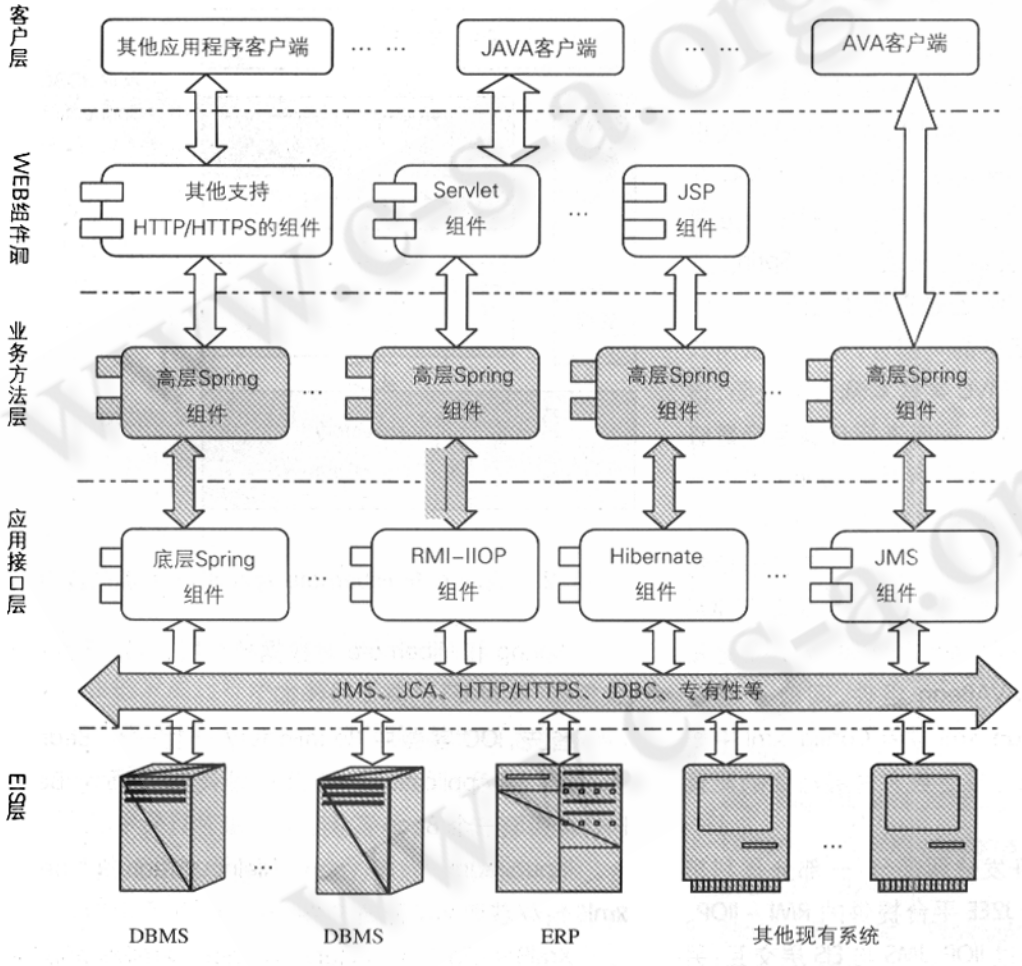

图 1　基于 J2EE、Spring 和 Hibernate 的轻量级 EAI 构架

根据以上的讨论和说明，以 Spring 框架（IOC，AOP）取代 EJB 容器作为业务逻辑层的基础构架，Hibernate 取代 CMP，构成轻量级 EAI 平台具有十分重要的现实意义。本文所设计的轻量级 EAI 构架以 J2EE 为基本框架，充分利用 J2EE 提供的便利性；对于业务逻辑层，采用 Spring 和 Hibernate 相结合，IOC 容器作为业务逻辑层的高层组件容器、Hibernate 作为访问数据持久层的底层组件。该构架如图 1 所示。整个构架划分为五个层：客户层、WEB 组件层、应用接口层、业务方法层和 EIS 层。

3.1 客户层与 Web 组件层

客户层包含 Java 客户端和其它应用程序客户端。Java 客户端通过 HTTPs 与 Web 组件交互，同时也可以直接通过 RMI – IIOP 等协议与业务逻辑层组件交互。显然，企业现有应用并非全由 Java 编写，可能一部分是用 C + +，Delhpi 等完成的，其他应用程序客户端就是为了满足企业这种应用而扩展实施的。Web 组件层的目标是提供一个通用统一的表示层，即整个集成信息系统的用户接口。Web 组件主要是 Servlet 和 JSP，它们响应 HTTP 请求，生成瘦客户端用户接口。JSP 和 Servlet 是 J2EE 组件层技术，特别适合基于 Web 的用户接口实现，它们能够通过 JMS、IIOP 协议等与业务逻辑层的 Spring 进行通信，并且可以向客户端动态地生成内容。

3.2 业务逻辑层（应用接口层和业务方法层）

业务逻辑层是该集成构架的体系核心所在，它体现并处理企业的业务过程。按抽象程度与粒度大小的不同，该层组件分为高层业务方法层组件与低层应用接口层组件。低层应用接口层组件主要是对外部系统 API 的封装，屏蔽现有应用 API 使用的协议

与技术的不同,起到一种应用接口代理作用;高级业务方法层提供基于业务需求的接口,是业务过程的映射,它反应高层业务过程。一般而言,高层业务方法组件会把一个调用请求分解与映射到一系列的低层应用接口层组件,并调用它们来共同协作完成请求任务。

业务方法层组件是在 Spring 框架下开发的,Spring 框架是一个基于 POJO[1,8](Plain Ordinary JAVA Object)、IOC(Inversion of Control)和 AOP(Aspect Oriented Programming)的开放源代码的轻量级 J2EE 应用框架。Spring 的架构基础是基于使用 JavaBean 属性的 IOC(控制反转)容器。IOC 容器提供一种无侵入式的高扩展性框架,无需代码中涉及 Spring 专有类,即可将其纳入 Spring 容器进行管理。该框架需要自主开发许多 JAVABean 类,只需在 Bean. xml 配置文件中修改相应的 properties,这些 JAVABean 就可直接嵌入 IOC 容器,成为新的业务方法层组件,与 EJB 组件相比十分简洁方便,充分体现出 IOC 容器的无侵入性。Spring 提供解决企业应用的完整框架,但是它是非强迫性框架,即如果不需要 Spring 框架中的一部分,完全可以用其它框架代替。由于 Spring MVC 某些紧耦合的缺陷,该集成构架不需要 Spring MVC,采用流行且满足轻量级应用的 WebWork 2 或者 Struts[8]代替。Spring 提供一个用标准 Java 语言编写的 AOP 框架,给 POJO 提供声明式的事务管理和其他企业事务,使得事务管理相对简单清晰。基于上面的讨论,业务方法层是基于 IOC 容器的 BeanFactory 或 ApplicationContext 以及在此基础上基于 AOP 框架的 JAVABean 组件,同时只需在 JAVABean 的配置文件 Bean. xml 或者 Config. xml 中修改有关 properties,就可以完成业务方法层组件的配置与加载。

应用接口层组件的开发分两部分:一部分是利用 J2EE 提供的便利性,使用 J2EE 平台提供的 RMI – IIOP、JMS[4,7]等技术来开发,通过 IIOP、JMS 与 EIS 层交互;另一部分是利用 spring 和 Hibernate 开发数据持久层框架。Spring 提供统一的 DAO 支持,包括 HibernateDaoSupport、JDBCDaoSupport 和 JDODaoSupport。其中 JDO[6]不是开放源代码的组件,价格昂贵;而且就该集成框架而言,JDBCDaoSupport 以及 HibernateDaoSupport 已经完全能够完成数据持久层的构架,因此不需要使用 JDO 组件。Spring 采用模块化 JDBC Template 取代 BMP[4](bean – managed persistence),同时采用 Hibernate 组件取代 CMP(container – managed persistence)。由于 Spring 对容器事务的良好支持,在基于 Spring Framework 的应用开发中,采用容器管理事务,可以获取数据逻辑代码的最佳可读性。因此,选择采用 Hibernate 作为数据持久层构架,而 JDBC Template[9]作为一个扩展框架是较好的方案。这样,通过 Spring 和 Hibernate 框架,对数据持久层的操作就变得十分简便高效。Spring 和 Hibernate 对数据持久层的操作如图 2 所示。

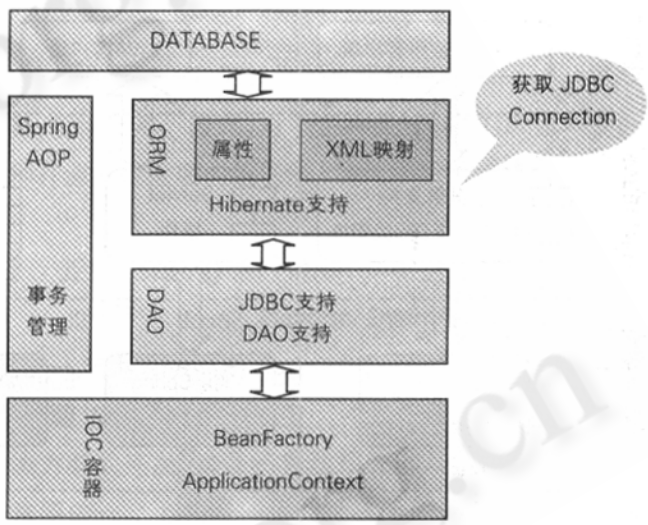

图 2　Spring 和 Hibernate 对数据持久层的操作

Spring 和 Hibernate 对数据持久层,特别是关系型数据库(RDBMS)的具体操作如下:

首先,IOC 容器中 BeanFactory 产生一个 Bean 实例,即某种 Application。下面一段代码是通过 BeanFactor 获取一个 Bean 实例:

```
InputSream is = new FileInputSream ("bean.xml");//获取 xml 配置文件
XmlBeanFactory factory = new XmlBeanFactory (is);//创建一个 factory
//创建 myinstance 实例
MyInstance myinstance = (MyInstance) factory.getbean("This is my Instance");
```

接着,该 Application 需要对数据持久层,特别是 RDBMS 进行某种交互,例如更新、添加或者删除某些纪录。此时,需要 Spring AOP 对事物处理进行粒度划

分,以减少事务处理之间的紧耦合性和高关联性。

然后,该 Application 通过 DAO 产生 Persistent Object,此 PO(Persistent Object)在 Hibernate 框架下修改 HibernateSupport.xml 的 properties 产生相应的 xml 映射(ORM),生成一个 JDBC Connection,与数据库交互。下面一段代码是获取一个 JDBC Connection:

```
Configuration config = new Configuration().configure();
SessionFactory sessionFactory = config.buildSessionFactory();
Session session = sessionFactory.openSession();
```

最后,数据库把交互信息传送给 ORM,最终通知 Bean 实例,完成该 Application。下面一段代码是通过 Spring 和 Hibernate 增加一个用户,显然其他的操作(删除,更新)与此类似:

```
public class UserDAO extends HibernateDaoSupport implements IUserDao
    {
     public void insertUser(User user){
       getHibernateTemplate().SaveOrUpdate(user);}
    }
```

3.3 EIS 层

EIS 层包含企业现有的信息系统[5],这些信息系统是在企业的发展过程中为解决专门问题而陆续建立起来的,如 ERP 系统、数据库管理系统、其他现有系统等,这些系统都在处理着企业的一些关键任务。这些系统在设计时并没有考虑到要作为整个企业信息系统的一部分,它的建立主要是为孤立的职能部门服务,所以在构建时使用了不同的技术、不同的语言、不同的数据库,这导致了这些信息系统间的协同工作能力十分有限。要集成这些系统,关键的问题是如何与它们交互,显然,J2EE 平台为实现 EIS 层与业务逻辑层的交互提供了丰富的方法与技术,如 JDBC、IIOP、RMI – IIOP、JCA、JMS 等[4]。由于 J2EE、Hibernate 以及 Spring 本身的便利性,我们可以便捷地通过 JDBC、IIOP、RMI – IIOP、JCA、JMS 等相应的技术和方法,实现业务逻辑层特别是应用接口层和 EIS 层的交互。

4 结束语

本文结合了 J2EE、Spring 与 Hibernate 的特点,设计了一个面向中小企业的基于 J2EE、Spring 与 Hibernate 的轻量级 EAI 构架,并剖析了各个环节的关键技术实现问题。此构架已经在实际的项目中成功的实施,表现出了良好的性能与健壮性。Spring 和 Hibernate 构架并不是完美无缺的,仍然存在一些缺点:(1)Hibernate 限制了所使用的对象模型,即只支持 ORM(关系对象模型);(2)Spring 框架下,JSP 中要写很多代码[8],这对程序员和使用者来说是个不小的麻烦;(3)Spring 控制器过于灵活,缺少一个公用控制器。

目前 Spring 和 Hibernate 作为轻量级的企业计算方案已经渐渐被业界所接受,特别是对于中小企业,开放源代码的轻量级框架正是它们所需要的。该框架如何处理在 JSP 中编写太多代码,确实是一个值得解决的问题。本文所提的 EAI 构架只是个解决方案,其它工作有待进一步研究。

参考文献

1 Spring 官方网站:http://www.springframework 以及 SpringFramework 中文论坛:http://spring.jactiongroup.net/

2 Hibernate 官方网站:http://www.hibernate.org 以及 Hibernate 论坛:www.jdon.com

3 张玉东、刘广钟,基于 J2EE 平台和 Web 服务的企业应用集成方案,计算机工程与设计,2004 Vol.25 No.11 。

4 Subrahmany Allamaraju, Cedric Buest, John Davies etc. Professional Java Server Programming J2EE 1.3 Edition. Sun Microsystems Inc. 2002 pages: 649 – 692 757 – 829.

5 Matjaz B. Juric. Professional J2EE EAI. America: Wrox, 2003 .

6 Rod Johnson, Juergen Hoeller. Expert One – on – One J2EE Development without EJB. America: Wrox, 2004 pages: 113 – 129.

7 William Crawford, Johathan Kaplan. J2EE Design Patterns. America: Wrox. 2003 pages: 41 – 49.

8 Rod Johnson, Juergen Hoeller etc. Spring 1.1.2: J2EE Application Framework. 2005 Reference Documentations.