基于 JSF 与 EJB3 的 Web 信息系统设计①

The Design of web information system based on the integrated of JSF and EJB3

范会联 （涪陵师范学院 计算机科学系 重庆涪陵 408003）

摘要：JSF 是 Web 应用开发的新框架，EJB3.0 规范中采用 Java EE5 中的程序注释工具和基于 Hibernate 的 O/R 映射模型，引入 IoC、AOP 等新的思想，给出了一个构建轻量级分布式组件的解决方案，本文提出基于 JSF 与 EJB3.0 规范的 Web 应用框架，并分析其中的关键技术。

关键词：EJB3.0　JSF　注释　对象/关系映射

1　引言

Web 应用开发的关键是提供良好的数据持久化层、简捷快速的表示层构造以及对业务逻辑、页面导航的有效管理。目前，数据持久层有多种解决方案，比如 Hibernate、EJB2.0 等都提供了各自的对象关系映射方案，然而 Hibernate 缺少 IoC（Inverse of Control）、AOP（Aspect－Oriented Programming）方面的服务，一般还需配合 Spring Framework 使用，EJB2.0 虽然作为 J2EE 的核心技术，但其复杂性使它在 J2EE 架构中的表现一直不是很好。EJB3.0 规范则利用 Java EE5 中的程序注释工具和基于 Hibernate 的 O/R 映射模型，引入 IoC、AOP 等新的思想，提出了一个轻量级分布式组件的解决方案。基于 UI（User Interface）组件构建 Web 界面的 JSF 则提出了 Web 应用的新框架，支持快速界面构造、分离表示和业务逻辑层及可配置的页面导航管理。

2　基于 JSF 与 EJB3.0 的 Web 系统结构

EJB 3.0 是至今为止 EJB 历史上最大的一次改动，它充分吸收了一些开源项目，比如 Spring、Hibernate 的经验，一改以往 EJB2.X 规范中复杂的接口定义，而是用 Java EE5 中的程序注释（Annotation）的描述机制将所有 EJB 实现为 POJO（Plain Old Java Object）类，大大方便了 EJB 的创建、使用和测试。不仅如此，EJB3.0 规范带来的新的变化还包括基于注释的依赖注入、拦截、事务管理。更为重要的是 EJB3.0 的 Java 持久化 API（JPA）作为 EJB3.0 规范（JSR－220）中的一部分，标准化了 Java 平台下的持久化 API，使 EJB3.0 规范为 Java 企业应用构建持久化层提供了一个强制性的选择。EJB3.0 JPA 通过注释（Annotations）来描述对象/关系（O/R）映射，它支持继承、多表映射、连表以及提供功能齐全的 EJBQL 查询语言。因此，新的实体 Bean 是一个加了注释的简单 Java 对象，该对象经过 EntityManager 的访问就变为一个持久化对象，成为持久化上下文（Context）的一部分。

JSF 为基于 Java 的 Web 应用提供了一个新的用户界面开发及组件管理的框架，其设计目的是实现简捷、快速的表示层构造和良好的业务逻辑管理。它具有一套预制的 UI 组件和基于事件驱动编程模型。利用 JSF 所提供的组件模型可以方便地创建用户接口，并且组件中封装事件处理、输入验证等操作。JSF 通过控制器 Faces Servlet 提供 Web 应用程序生命周期管理，整个前端控制就是由 Faces Selvet 和配置文件（web.xml、faces－config.xml 等）及一系列 action 组成。JSF 所提供的运行环境可以很容易地将客户端事件与服务器端的事件处理程序绑定，将 HTTP 请求映射为具体组件的事件处理从而改变了以往基于 java web 应用的 request－response 处理机制。

① 基金资助：重庆市教育科研项目（04－GJ－063）

3　基于 JSF 与 EJB3.0 的 Web 系统设计

3.1　对象/关系映射

以"系(院) - 用户 - 教师关系"为例分析对象/关系映射,其数据库逻辑设计(部份)如图 2 所示。EBJ3.0 规范提供面向对象的设计与关系型数据表的映射,对数据表进行面向对象的封装,实现对象到关系型数据库的持久化服务。进行对象/关系映射的方法是:

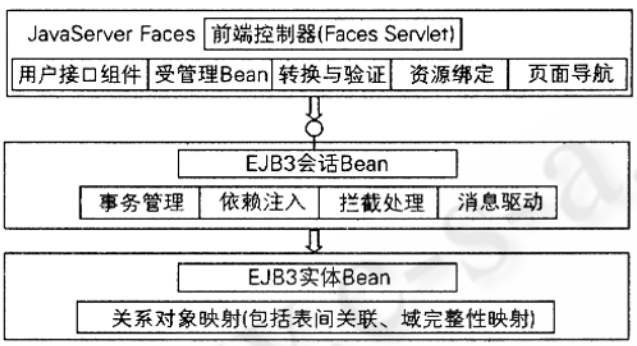

图 1　基于 JSF 与 EJB3 的 Web 系统结构

首先,将数据表映射为 POJO 类,EJB3.0 规范用@Entity 注释创建实体 Bean,该 Bean 对应的数据表由@Table 指定,Bean 中封装的属性与数据表中的字段应具有相同的名字(如若不同名,则需用@Column 注释单独指定对应的数据表字段)并给出针对每个属性的 set/get 方法。值得注意的是,可以在实体 Bean 中进一步用注释指定主键的生成方式以及给出针对每个字段的域完整性约束。

然后,进行数据表间的关联映射。将关联的数据表分为关系维护端与被维护端(都以双向关联为例)。一对一关系的映射用@OneToOne 注释定义,其中在关系维护端指定 mappedBy 属性,由该属性指定的实体 Bean 维护其关联的对象。在关系被维护端建立外键列指向关系维护端的主键列。因此,用户与教师的一对一关系映射注释如下:

User(维护端):@OneToOne(optional = true, cascade = CascadeType.ALL, mappedBy = "user")

Teacher(被维护端):@OneToOne(optional = false, cascade = CascadeType.ALL)

@JoinColumn(name = "teacherID", referencedColumnName = "userName",unique = true)

其中,optional 属性是定义其关联类是否必须存在。cascade 属性定义类和类之间的级联关系,该级联关系将被容器视为对当前类对象及其关联类对象是否采取相同的操作(删除、更新)。@JoinColumn 指明被维护端的 teacherID 列作为外键与 User 表中的 userName 列关联。

一对多及多对一关系中,一是关系的维护端,多是关系被维护端,映射时分别用@OneToMany 和@ManyToOne 注释。系(院)与用户之间的映射如下:

Department(关系维护端):

@OneToMany(mappedBy = "department", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

public Set<User> getUsers(){…} //获取多端对象集合的 get/set 方法

public void setUsers(Set<User> users){…}

public void addUser(User user){…} //从维护端添加/删除被维护端

public void removeUser(User user){…}

User(关系被维护端):

@ManyToOne(cascade = CascadeType.ALL, optional = true) @JoinColumn(name = "departmentDM")

public Department getDepartment(){…} //获取一端对象的 get/set 方法

public void setDepartment(Department department){…}

其中,关系维护端中的 fetch 属性有两个可选择项:FetchType.EAGER 和 FetchType.LAZY。前者表示关联类(User 类)在主类(Department 类)加载的时候同时加载,后者表示关系类在被访问时才加载(延迟加载)。

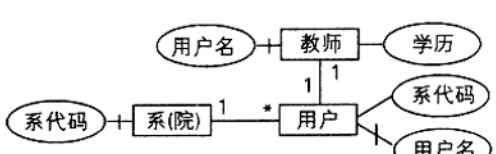

图 2　数据库逻辑设计

3.2　业务逻辑管理

业务逻辑封装在会话 Bean 中,用@Stateless 或者

@Stateful 注释区别无状态还是有状态会话 Bean，@Remote 或者@Local 注释指定相应的远程/本地接口。借签其他开源项目的经验，EJB3.0 规范在会话 Bean 中引入了两个非常重要的思想来降低应用组件之间的依赖性。首先是依赖注入(Dependency injection，也称反转控制 IoC)。为了存取服务对象，通常需要通过服务器的 JNDI 来查找相应资源，因此，JNDI 的有效查找是实现组件间解藕的关键。基于依赖注入的思想，EJB3.0 改变以往直接使用字符串来进行 JNDI 查找的方式，对任何 POJO 对象提供一个简单和优雅的方法来解藕服务对象和资源，使用@EJB 注释，可以将 EJB 存根对象注入到 EJB 3.0 容器管理的 POJO 中，而@Resource 注释用来注入来自 JNDI 的其他资源(比如数据源)。其次是基于 AOP 思想的拦截器，用于监听程序的一个或所有方法，在这些方法执行前首先执行拦截器中定义的行为，因此，拦截器对方法调用流提供了细粒度控制，可以在无状态会话 Bean、有状态会话 Bean 和消息驱动 Bean 上使用它们。定义和使用拦截器的方法是：在会话 Bean 中用@Interceptors 注释指定一个外部的检查方法类，在检查方法类中用@AroundInvoke 注释指定要用作拦截器的方法，在具体的检查方法中，如果通过检查，则用 proceed()来返回到会话 Bean 中被检查的业务方法中，从而实现当调用会话 Bean 中的业务方法时的拦截检查。

EJB3.0 中，@TransactionAttribute 注释用来为一个 POJO 方法申明它的事务属性，容器就可以在合适的上下文中运行这个方法，以保证当方法出现失败或异常时数据的完整性。@TransactionAttribute 默认的事务行为是 REQUIRED，可以通过指的@TransactionAttribute 的参数指定为其他的事务行为(MANDATORY、REQUIRESNEW、SUPPORTS、NOT_SUPPORTED)。

3.3 JSF 在系统设计中的作用

JSF 提供基于 MVC 模式的管理框架，并负责与客户端交互。首先，Model 层是 JSF 中的受管理 Bean，在配置文件中使用 < managed - bean > 定义，由于 EJB3.0 中的 Bean 都是 POJO 类，因此，Model 层中的 Bean 可由 EJB 承担，并且可以直接利用实体 Bean 作为数值对象(Value Object)，在表示层和业务逻辑层间传输数据，更为重要的是，如果在实体 Bean 中为属性指定了域完整性约束，则数据在提交时会进行相应的验证。View 层是用户视图，JSF 在它的核心标签库和 HTML 标签库中提供了丰富的标签用于快速设计和构造用户界面，并且 JSF 中的消息包绑定机制提供了视图层中文字的集中管理从而方便 Web 应用的国际化。Control 层连接 View 与 Model 层，由 javax. faces. webapp. FacesServlet 控制器联合 faces - config. xml 文件中 < navigation - rule > 指定的导航规则来控制整个 Web 应用的响应流程。

JSF 提供三种类型的事件响应：值变化事件、动作事件和阶段事件，当输入组件的值发生变化并且比较表单时，输入组件(如 h: inputText)就触发相应的值变化事件，当激活按钮或链接时，命令组件(如 h: commandButton)会触发动作事件，而阶段事件则由 JSF 生命周期触发。

缓存对于系统性能的提升具有非常重要的作用，基于 JSF 与 EJB3 的 Web 应用可以从以下两方面建立缓存，其一是 EJB3.0 规范为实体 Bean 提供了@Cache 注释从而达到缓存该实体 Bean 对象的目的；其二是 JSF 框架提供了方便的用户缓存实现方式，通过分别创建 ApplicationBean、SessionBean 并纳入 JSF 管理，在配置文件中设置 managed - bean - scope 属性为 application 或者 Session 实现数据缓存。

4 结束语

基于 EJB3.0 构建轻量级分布式组件、用 JSF 创建表示层及管理组件的 Web 应用开发方案能有效地缩短开发周期、提高开发质量，并使系统具有良好的交互性、可扩展性和可维护性。

参考文献

1 JBoss org . JBoss EJB 3.0 Tutorial[EB/OL]. http://docs. jboss. org/ejb3/app - server/tutorial/index. html ,2005 - 06.

2 Sun Microsystems. JSR 220: Enterprise JavaBeansTM, Version 3.0[EB/OL]. https://sdlc1e. sun. com/ECom/EComActionServlet/DownloadPage,2006 - 05 - 08

3 (美)DAVID GEARY. CAY HORSTMANN 著. 王军，马振萍等译. JavaServer Faces 核心编程[M]. 北京：电子工业出版社，2005 - 4.