第33卷 第23期 计 算 机 工 程 2007年12月
Vol.33 No.23 Computer Engineering December 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)23—0276—03 文献标识码：A 中图分类号：TP393.07

# J2EE平台下社保综合系统的设计与实现

闫宏印，冯 浩
(太原理工大学计算机与软件学院，太原 030024)

**摘 要：**根据社保行业的特点和实际需求，提出一种基于J2EE平台的C/S/S结构社保系统模型。系统设计过程中引入组件化和工作流的思想，使用Business Delegate, Session Facade等多种J2EE设计模式。为了规范系统调用层次和搭建系统服务，构建了企业级框架。应用解决EJB调用瓶颈的优化策略，提供安全服务和日志服务。

**关键词：**J2EE; MVC; Facade; Hibernate

# Design and Realization of Social Security Comprehensive System Based on J2EE

YAN Hong-yin, FENG Hao
(College of Computer and Software, Taiyuan University of Technology, Taiyuan 030024)

**【Abstract】** According to features and demands of the social security industry, this paper presents a C/S/S structural social security system model based on J2EE. In system design process, designers introduce the ideas of component, workflow; use several J2EE design patterns such as Business Delegate, Session Façade, and construct the enterprise framework for normalizing system hiberarchy and building system services. The system applies an optimization strategy for solving bottleneck of EJB calls and provides security service, log service.

**【Key words】** J2EE; MVC; Facade; Hibernate

截至2005年末，全国参加城镇基本养老保险的人数已经超过1亿7千万人，5项社保基金收支总规模在1万亿元以上。面对如此巨大的信息流和资金流，劳动保障部门正在加强社会保障信息化建设，进而改进工作方法和管理方式，提高工作水平和管理效率。

按照社保系统的建设目标，在吸收了传统的C/S结构和目前广泛采用的B/S结构的长处之后，笔者采用了一种J2EE平台下的多层体系架构解决方案。这种多层架构将业务逻辑分布到应用服务器上，数据库不再具有业务逻辑处理单元，只负责基础业务数据的管理，从而充分利用了应用服务器在并发处理和逻辑计算方面的优势。此外Web服务器和应用服务器分别可以做集群的配制，对多台服务器进行统一管理，分配和并行处理全部的外部请求。当计算请求并发量巨大时，集群的多台服务器之间可以动态地进行任务分配，达到负载均衡，确保系统性能不会因为大量并发用户的访问而急剧下降。系统同时也具备了良好的可扩展性和伸缩性，即在请求并发量增大或减少时，可根据实际情况增加或减少服务器数量。在保证性能的前提下，合理利用硬件资源。

## 1 系统总体架构和功能

### 1.1 系统总体架构

该系统技术结构是基于J2EE体系的C/S/S结构，开发环境为Delphi+Eclipse+Weblogic+Oracle。总体架构设计基于MVC模式[1]，如图1所示。

(1)视图层(view)：即人机界面，负责处理与用户的交互和应用程序的数据表示。本系统中的视图层是用Delphi编写的传统应用客户端。

(2)控制层(controller)：视图层用户发出的命令触发控制层框架中的MainServlet，由MainServlet操纵模型层中的对象来完成命令功能。即由控制层控制用户对业务逻辑的操作，同时把操作结果显示给客户。

(3)模型层(model)：用来处理核心业务实体及相关的业务逻辑。该系统的模型层包括业务逻辑层、数据持久层和数据源层，为系统的核心部分。

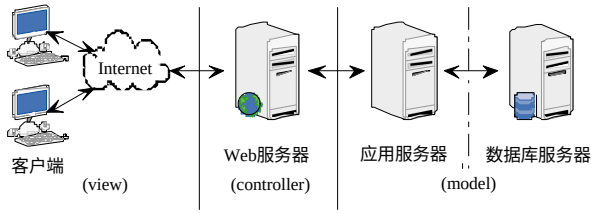


**图1 系统总体架构**

### 1.2 系统功能概述

作为定位于城市级的社保综合系统，能够支持城市级的业务经办和基金管理，并通过其统计查询功能为决策支持提供服务，通过与呼叫中心和网站的连接提供公共服务[2]。整个系统按业务划分为养老、失业、医疗、工伤、生育5个子系统，各子系统既可单独运行，也可任意组合。

系统功能仅以养老保险管理子系统为例加以概述：

(1)基本信息管理：提供对参保单位、参保人员等基本信

**基金项目：**山西省自然科学基金资助项目(20001035)；山西省回国留学人员科研基金资助项目(2004-26)
**作者简介：**闫宏印(1954－)，男，教授，主研方向：计算机网络，辅助教学；冯 浩，硕士研究生
**收稿日期：**2007-01-31 **E-mail：**fedarain@yahoo.com.cn

息的管理。

(2)基金征集：提供单位、独立缴费人员等缴费的计算和征集、到账处理、划账等业务。

(3)参保人员待遇审核：包括退休待遇审核、计算指数化工资、待遇停发、待遇续发等业务处理。

(4)待遇支付：包括应付通知、支付通知等业务模块。

(5)接口：提供灵活的与外部财务系统的接口。

(6)决策支持：对相关数据进行统计分析，提供决策依据。

(7)系统管理：提供系统运行需要的用户、权限、交易、日志、参数等的管理。

## 2 系统架构

系统各层间通过接口进行调用。每层具有清楚的定义与接口协议，可被看作一个独立的控件。层次的内部结构更新时只需修改相关接口的实现类而接口保持稳定，使系统各层间保持最小依赖性，实现了层次间的松耦合。系统采用方法库方式来管理不同的业务功能，每个功能号对应一种业务处理方法，通过功能号装载其业务方法即可实现相应的功能。

如图 2 所示，客户端通过网络向 Web 服务器发出 SOAP 消息，服务器端框架的 MainServlet 将接收到的 XML 数据流转化成统一的 Java 类 Event，并根据解析出的功能号将 Event 分发给请求处理层中对应的 EJBAction。由 EJBAction 的 perform()方法调用 EJB 组件进行业务处理，然后 BPO 调用 PersistenceManager 访问数据库。接着服务器端请求处理层将 EventResponse 对象(包含业务处理组件的回应数据集)作为返回值传回框架，MainServlet 再将 EventResponse 转换成 SOAP 消息发送到客户端。

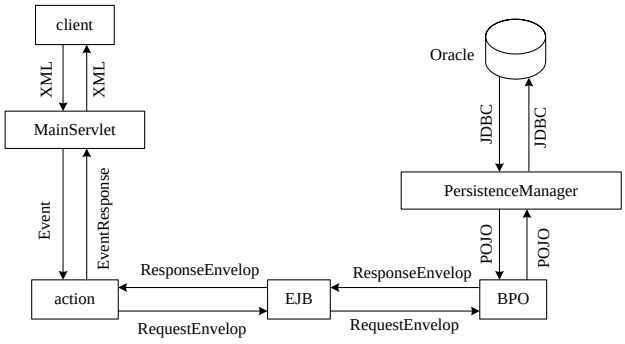


**图 2 系统各层间数据传递的流程**

位于控制层的系统主框架是整个系统的调度中心。MainServlet 负责建立起服务器级的环境，为所有用户提供服务。FrameConfig.xml, actionmap.xml, facademap.xml, EJBCache 等在 MainServlet 启动时配置。

业务逻辑层分为请求处理层和业务组件层，应用了 Action+Session Facade+BPO 的模式。Action 是动作描述，用于实现 Business Delegate 设计模式，其作为业务组件层的代理联系着系统主框架和 EJB 组件层；Session Facade 为一类原子业务的处理类(BPO)提供 EJB 组件级的方法调用接口，本身不出现业务处理逻辑或数据处理逻辑，由 Stateless Session Bean 实现；BPO 则对动作进行实际响应，实现给定业务规则和事务的集中处理。Action 类作为所有 EJBAction 的超类，包含 4 个基本方法：validate()提供对数据的校验，出现校验错误返回 EventError 对象；processRequestEvent()把 Event 转换成 RequestEnvelop；perform()方法由子类覆盖，用于从系统环境中得到 Facade，然后调用接口的方法完成业务逻辑并获得返回的 ResponseEnvelop；processResponseEnvelop()方法将 ResponseEnvelop 转换成 EventResponse。

下面是工伤待遇登记模块的一个 EJBAction 实例：

```
public class InsertGssgAction extends Action{
public EventResponse perform(Event event){
super.validate(event);
RequestEnvelop requestEnvelop=
                                    super.processRequestEvent(event);
FacadeCache cache=(FacadeCache)SysContext.getInstance()
.getAttribute(GlobalNames.CONTEXT_FACADE_NAME);
ResponseEnvelop responseEnvelop=null;
try{
DydjHome home=(DydjHome)cache.getFacadeObj("gsdy_dydj");
DydjObject obj=home.create();
//调用业务方法
responseEnvelop=obj.DydjInsertGssg(requestEnvelop);
}catch (Exception e){
    return new EventResponse(false, "  调用 Dydj 处理出错",e);
}
return super.processResponseEnvelop(responseEnvelop);
}
}
```

在持久层的实现上选用了 Hibernate3 进行对象和关系数据库之间的转换。根据已有的数据库，使用 Middlegen 工具可生成 hbm.xml 映射文件和与之对应的 POJO 文件。Hibernate 实体类不涉及业务相关流程。

## 3 关键技术

### 3.1 系统主框架

系统的主框架分为 2 大部分：客户端 DLL 函数库以及服务器端框架。

如图 3 所示，客户端的 DLL 函数库是负责客户端和服务器端之间联系的纽带，主要完成接收客户端的数据，转化客户端的数据为标准 SOAP 格式的 XML 字符串，发送 XML 字符串；接收服务器端回应的 XML 字符串，解析 XML 字符串，通过接口提供客户端对服务器返回数据的调用。

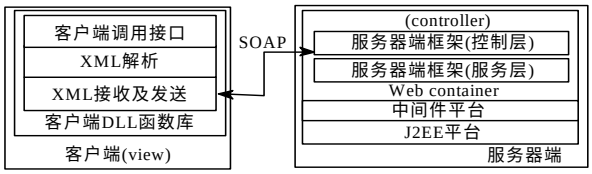


**图 3 系统主框架**

服务器端框架分为控制层和服务层，分别用于实现流程控制和系统服务。由 Filter 对发送过来的各种请求统一进行约束限制处理。其中 HeadFilter 类过滤出 XML 的 Header 信息；SaftyFilter 类对请求的合法性进行判定和校验，并建立用户级的环境，以便为指定用户提供服务。通过 Filter 的合法请求全部由 MainServlet 主控制器的请求分发处理器 RequestProcessor 依据配置文件 actionmap.xml 动态分发，并添加了对线程的约束和控制，保证所有请求可以顺利地分发到对应的 EJBAction。经业务逻辑层处理后，服务器端框架将返回数据集打包成 XML 字符串，返回客户端。

除流程控制外，利用集成的系统服务，服务器端框架可以方便地进行统一的日志管理、安全管理和缓存服务。系统主框架的设计与具体业务无关，能够在相同的技术构架下得到最大限度的重用。

### 3.2 业务组件

Enterprise JavaBeans 是基于分布式事务处理的企业级应用程序的组件，用于实现业务逻辑和 O/R 映射。由 EJB 容器管理和调度这些 EJB 组件，实现多线程、事务、集群等复杂的功能。

系统采用 J2EE 组件技术，将核心业务做成组件形式，使业务层得到最大限度的扩展和重用。用户只需将新的业务组件部署在服务器上，修改配置文件将业务处理类配入系统，不需修改系统架构的相关代码，不需重新编译系统即可完成业务的扩展。并且在组件的数据交互接口不变的前提下，如果业务逻辑发生变化则只要修改组件内部逻辑，不用修改系统的其他部分。实现了组件间的松耦合及业务逻辑的封装。

### 3.3 数据持久层框架

Hibernate的精髓是持久层实现模式。它处于数据库与应用程序之间，不仅管理Java类到数据库表的映射，还提供数据查询和获取数据的方法[3]。

使用 Hibernate 的步骤：

(1)根据配置文件创建 SessionFactory；

(2)打开一个 Session；

(3)开始事务；

(4)业务操作；

(5)提交事务；

(6)关闭 Session。

Hibernate 提供的类和方法，灵活、功能强大，但如果直接使用比较繁琐，而且和 Hibernate 绑定在一起不利于升级替换 Hibernate 包。因此，使用了新的框架，应用程序只需与 PersistenceManager 进行交互。PersistenceManager 通过 HibernateUtil 操作 SessionFactory, Session 和 Transaction，HibernateUtil 类使用线程提供了对它们的访问。PersistenceManager 包装了 Hibernate，提供基本的数据访问方法和事务控制，BPO 使用这些方法可以方便地进行记录的插入、查询、修改、删除和事务的提交、回滚等操作。PersistenceManager 不只提供方法实现 HQL 对数据库的访问，考虑到多表级联和复杂的业务查询处理等因素，同时给出了用 SQL 访问数据库的方法。

### 3.4 日志处理框架

按照系统需求，只提供调试和跟踪应用程序的功能是不够的，发生严重错误时还需要将操作人员的行为记录到数据库中，因此日志处理部分要更加灵活。

Apache Commons Logging 框架提供了一个工厂方法 LogFactory.getLog(Class class)来获取 Log 实例的引用。在配置文件 commons-logging.properties 中指定系统中实现 org.apache.commons.logging.Log 接口的类，然后交给 LogFactory 来管理 Log 接口的具体实现类。用户程序只和 Commons Logging 接口交互。

Log 接口的实现类包装了 Log4j 和 JMS(Java 消息服务)。当输出的是程序开发或运行过程中的普通调试信息时，使用 Apache 的开源项目 Log4j 将日志信息输出到 Weblogic 控制台。系统提供了 5 个日志层次，从低到高依次为 debug, info, warn, error 和 fatal。当一个日志被赋予层次之后，只有日志输出请求的层次大于或等于该日志的层次才会被允许执行，否则其输出日志将被屏蔽。

要将错误信息写入数据库时，为使系统日志处理不阻塞系统的正常运行，提高响应速度，采用了用 JMS 实现的异步处理模式。实现了符合 Log4j AppenderSkeleton 的 JMSAppender 类，用于处理日志事件，把各类日志信息转送到不同的日志消息队列中。系统框架则实现了处理相应日志消息的 Message Driven Bean，它们监听日志消息队列，并把日志信息输入到数据库载体中。

### 3.5 安全机制

系统的安全性主要体现在 2 点：

(1)通信机制和敏感数据的安全性

1)采用防火墙和安全监控系统等手段对网络访问进行限制，保证信息不被非法用户访问。在限制网络访问权限的同时，注意路由器、交换机等核心网络设备的配置和使用，保证授权访问的口令安全。

2)网络传输数据时以密文的方式进行传递，以保证信息传递安全。即使第三方获得了传递的信息，也因为信息已加密而使其无法理解信息内容。加密方案支持传输过程和存储过程的加、解密处理和身份认证；数据传输过程中产生的检验码可以及时发现数据遭破坏的情况；密钥保证了足够的长度。

3)敏感数据(用户名、密码、金额等)的加密由专业的加密模块完成，加密模块具有完整、严谨的密钥管理和维护机制，确保密码系统的高强度。

(2)软件的安全性

1)采用用户存取权限分级、集中授权的方式，整个系统按照组-用户的关系模式管理。每个系统的使用者都有自己的用户名及密码。由特殊的管理员账号集中设定系统中应用账号的用户名、密码、所属的组等权限。

2)禁止系统用户在登录成功后调用其权限之外的模块，从而非法访问系统。

3)服务器框架对用户的每次请求进行记录，调用安全日志模块记载用户登录信息，包括：用户名，登录时间，登录位置；功能操作信息包括：功能号，操作号记录等。对重要的数据访问活动进行审计，追查恶意或无意的数据修改和破坏，以做到信息处理的保密要求。

## 4 结束语

本文设计实现了基于 J2EE 平台的 C/S/S 结构的社保综合系统。该系统遵循开放的行业标准，具有可扩展性、可维护性、安全性好等特点，适用不同地域社保机构的信息化建设。利用系统的统计决策功能模块，可以方便地分析处理数据库中的数据，为有关部门提供决策支持。

### 参考文献


1 杨 涛, 周志波, 凌 力. 基于 Struts 和 Hibernate 的 J2EE 快速开发框架的设计与实现[J]. 计算机工程, 2006, 32(10): 83-85.

2 社会保险管理信息系统核心平台二版(SIMISCP2.0)介绍[Z]. (2005-12). http://www.molss.gov.cn/gb/zt/2005-12/14/content_98254.htm.

3 Bauer C, King G. Hibernate in Action[M]. Greenwich: Manning Publications Co., 2005.