基于 Spring 框架的轻量级 J2EE 架构与应用

胡启敏[1,2],薛锦云[1,2],钟林辉[1]
HU Qi-min[1,2],XUE Jin-yun[1,2],ZHONG Lin-hui[1]

1.江西师范大学瑶湖校区 计算机信息工程学院,南昌 330022
2.中国科学院 软件研究所 计算机科学重点实验室,北京 100080
1.College of Computer Information Engineering,Jiangxi Normal University,Nanchang 330022,China
2.Laboratory of Computer Science,Institute of Software,Chinese Academy of Sciences,Beijing 100080,China
E-mail:qiminhu@163.com

HU Qi-min,XUE Jin-yun,ZHONG Lin-hui.Lightweight J2EE architecture based on spring framework and its application.Computer Engineering and Applications,2008,44(5):115-118.

Abstract: Choosing a reasonable J2EE architecture is very important for application development.The techniques Spring framework provides,such as dependency injection features component assembly,transaction and log management,etc.can sustains lightweight J2EE architecture.This paper introduces the Spring framework and compares lightweight J2EE architecture based on Spring framework with other J2EE architectures and gives a case of its application finally.
Key words: Spring framework;J2EE;dependency injection;lightweight

摘 要:合理的 J2EE 架构设计方案对 J2EE 应用系统的开发至关重要。Spring 框架包含依赖注入方式的构件组装机制、统一的事务和日志管理服务等,为搭建轻量级的 J2EE 架构提供了有力的支持。介绍 Spring 框架的体系结构,将基于 Spring 框架的 J2EE 架构与其他几种 J2EE 架构进行对比分析,最后给出一个运用该架构方案解决的实例。
关键词: Spring 框架;J2EE;依赖注入;轻量级
文章编号:1002-8331(2008)05-0115-04 **文献标识码:**A **中图分类号:**TP311

1 引言

J2EE 平台规范是 SUN 公司制定的,基于 Java 技术的分布式构件运行平台规范,在此平台上可以方便、快速地建立融合 Internet 技术的多层结构的分布式企业应用[1]。符合 J2EE 规范的系统架构包含三种核心构件:(1)业务服务层构件,完成具体业务逻辑的处理,并为表现层构件提供服务;(2)表现层构件,实现接受用户请求,返回处理结果的人机界面;(3)数据访问层构件,通过访问后台数据库完成系统数据的持久化存储。

传统的 J2EE 架构方案采用带远程接口的 Session Bean 实现业务层构件,用 Entity Bean 实现数据访问层构件[9]。该架构方案大量使用 EJB 构件,能够充分利用 EJB 容器(例如:Weblogic)提供的分布式处理、构件定位、统一的事务和安全服务等,然而 EJB 构件却存在着复杂度高,开发效率低;难以进行单元测试;数据库访问效率低等问题[6,7]。普通的轻量级 J2EE 架构方案用 JavaBean 构件取代了传统架构中的重量级 EJB 构件,很大程度上提高了构件开发和构件的数据库访问效率。但是该架构过于简化,缺乏支持 JavaBean 构件运行的容器,构件需自行解决构件组装,事务、日志、安全管理等问题。

Spring 框架是 Rod Johnson、Juergen Hoeller 等开发的,用于支持 JavaBean 构件运行的容器。该框架提供了依赖注入[7,8]方式的构件组装机制和基于 AOP 技术的事务和日志管理等功能。基于 Spring 框架的轻量级 J2EE 架构能够发挥上述两种架构方案的优势,避免它们存在的缺陷,是一种成熟的 J2EE 应用开发方案。

本文介绍 Spring 框架的体系结构,将基于 Spring 框架的 J2EE 架构方案与其他几种 J2EE 架构进行对比分析,最后给出一个运用该架构方案解决的实例。

2 Spring 框架体系结构

Spring 框架体系结构如图 1 所示[10]。Spring core 包是整个框架的基础,它提供了基于依赖注入技术的构件组装机制;Spring DAO 包提供了一系列的数据库访问控制工具,免去了繁琐的数据库访问控制和异常处理的工作。Spring ORM 包允许程序设计人员将流行的 Hibernate、JDO、iBatis 等对象关系映

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60273092);国家重点基础研究发展规划(973)(the National Grand Fundamental Research 973 Program of China under Grant No.2003CCA02800);江西师大青年成长基金资助。
作者简介:胡启敏(1979-),男,讲师,博士研究生,主要研究方向为:软件体系结构、软件形式化与自动化;薛锦云(1947-),男,江西师大教授,中科院软件所博导,主要研究方向:软件形式化与自动化;钟林辉(1974-),男,讲师,博士研究生,主要研究方向为:软件工程与环境、构件技术。
收稿日期:2007-04-18 **修回日期:**2007-09-20

射工具集成到现有系统中。Spring AOP 包实现了面向方面编程的支持,可以为构件提供统一的事务、日志、安全管理等服务。Spring Web MVC 包提供了对基于模型-视图-控制器模式的 Web 应用程序开发的支持。

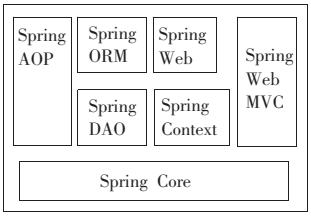

图 1 Spring 框架体系结构

2.1 基于依赖注入技术的构件组装

基于依赖注入技术的构件组装是 Spring 框架的技术核心和独特之处,它是指构件之间的依赖关系(即构件间的调用关系)用 XML 配置文件描述;在系统运行时,构件容器根据 XML 文件的描述完成构件组装。

当类包含它所实现的接口以及它所期望的其他类接口的显式说明的情况下,可以把类看成构件[11]。考虑如下几个构件和接口的定义:

```
Public interface Ai{
  public int d();}
Public class B implements T{
  private Ainterface Ai;
  public void setAi(Ainterface Ai){
    this.Ai=Ai;}
  public int d(){
    return Ai.d();}
}
Public class C implements Ainterface{
…}
Public class D implements Ainterface{
…}
```

构件 B 实现接口 T,期望从其他构件获取的服务在接口 Ainterface 中定义。构件 C 和 D 实现了接口 Ainterface,可以为构件 B 提供服务,即构件 B 可以依赖于 C 或 D 构件。

假设 B 构件依赖于 C 构件,则用 XML 定义的配置文件如下所示:

```
<beans>
 <bean id="B" class="B">
  <property name="Ai">
    <ref bean="C"/>
  </property>
  </bean>
  <bean id="C" class="C"> </bean>
  <bean id="D" class="D"> </bean>
</beans>
```

系统运行时,将由 Spring 框架实例化 C 构件,然后注入到 B 构件的 Ai 属性中,完成 B 和 C 构件的组装。如果软件需求变更,B 构件需依赖于 D 构件,则只需将配置文件中的<ref bean="C"/>项目,改为<ref bean="D"/>。

依赖注入技术将构件组装任务交给构件容器完成,显著地降低了构件之间的耦合程度,提高了系统的可维护性。

2.2 Spring 框架对数据访问层构件的支持

数据访问层构件完成整个应用系统数据的持久化存储,需要具备较高的处理性能和安全保障机制。采用实体 EJB 实现该层构件难以满足处理性能的要求,在 JavaBean 中使用 JDBC 访问数据库需要进行繁琐的异常处理编码工作。

Spring 提供了通用的数据库访问异常体系,用统一的模板(JdbcTemplate 模板)机制解决了开闭连接,处理异常等问题。利用 JdbcTemlate 模板,只需提供数据源,就可以简洁地完成各种查询和更新等数据操作。

2.3 Aspect 和统一的事务、日志管理

Aspect 是指软件系统中一些贯穿全局(cross-cutting)的特性,例如事务、日志等。面向 Aspect 的软件开发试图将这种特性的实现模块化,并与其他功能性的实现体分离开来,使用声明的方式将这些特性插入到系统实现中[2]。

Spring 框架集成了事务、日志等公共服务(允许自定义其他服务),能够将这些 Aspect 作用于应用系统中。Aspect 通常以类似于普通构件的方式在 XML 文件中配置。

3 几种 J2EE 架构方案的比较

3.1 传统的 J2EE 架构方案

传统 J2EE 架构如图 2[9]所示,包含运行于客户端的应用程序和 Applet,以及运行于服务器端 Web 容器中的表现层构件 Servlet 和 JavaServerPages(JSP),EJB 容器中的业务服务层构件会话 EJB,数据访问层构件实体 EJB,消息驱动 EJB。

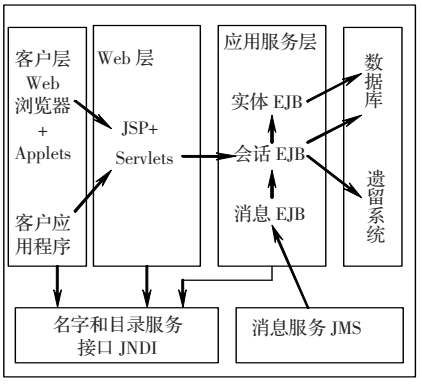

图 2 传统的 J2EE 架构

EJB 是传统 J2EE 架构的主要构件,EJB 容器为 EJB 构件提供了生命周期管理、基于 JNDI 的构件定位和统一的事务管理、分布式处理等服务,使 EJB 构件可以关注于具体的业务问题。

然而,EJB 是包含分布式处理,数据持久化等众多技术的重量级构件。一方面,EJB 十分复杂、开发效率低、难以进行单元测试;另一方面,实体 EJB 的数据访问效率很低,将直接影响整个系统的运行性能。研究数据[6]表明,直接访问实体 EJB 和用会话 Bean 封装实体 Bean 的数据访问效率远低于用会话 Bean 封装 JavaBean 的数据访问效率。

传统 J2EE 架构比较适用于各个业务构件分布在不同的服务器中,构件之间需要通过远程分布式操作进行关联或者构件间需要大量使用异步消息通讯的应用系统中;不适用于构件集中在同一服务器,不需要分布式处理,对系统运行性能要求高的系统。

3.2 普通的轻量级 J2EE 架构

该架构方案用轻量级的 JavaBean 构件取代了传统架构中的重量级 EJB 构件,如图 3:

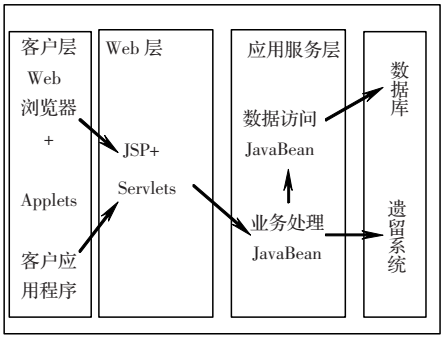

图 3　普通的轻量级 J2EE 架构

表现层构件 JSP 和 Servlet 负责接受用户请求，并交给业务构件处理；业务构件对数据库的访问将发给数据访问构件，由数据访问构件完成对数据库的增、删、查询等具体操作。

该架构中的 JavaBean 构件有较高的设计开发和数据访问效率,能够保证整个系统运行性能。但是该架构过于简化,缺乏支持 JavaBean 构件运行的容器,存在下列问题:

首先,构件组装要由构件自身完成,构件之间的耦合程度高，以业务层构件 Business 和数据访问层构件 DAO 间的耦合为例:

```
Public class Business implements T{
  private DAinterface DAi=DAO.getInstant();
  public List query(){
    return DAi.query();}
}
Public class DAO implements DAinterface{
public synchronized static DAO getInstant(){
  if(instant==null){
    instant=new DAO();
    }
    return instant;
  };
  public list query(){
     SQL 的查询语句;}
}
```

Business 构件要与 DAO 构件组装，就必须在 Business 构件中声明对 DAO 构件的引用,相互之间耦合紧密。

其次，没有事务管理、日志等公共的系统级服务的支持，JavaBean 构件在关注具体业务的同时,还需要关注上述系统级的服务;

最后，轻量级的 JavaBean 不具备 EJB 构件的分布式处理的能力,不能解决业务构件在不同服务器中分布的问题。

3.3　基于 Spring 框架的轻量级 J2EE 架构

Spring 框架为 JavaBean 构件提供了构件组装、统一的事务和日志管理、数据访问支持等服务,是支持 JavaBean 构件运行的容器(类似于支持 EJB 运行的 EJB 容器)。

该架构方案中,构件组装工作由 Spring 框架完成,构件之间的几乎零耦合,以 3.2 节中的业务层构件和数据访问层构件的组装为例:

```
Public class Business implements T{
  private DAinterface DAi;
  public void setDAi(DAinterface DAi){
    this.DAi=DAi;}
```

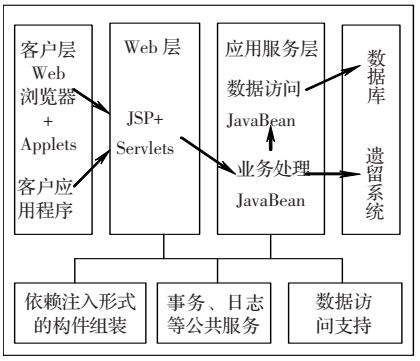

图 4　基于 Spring 框架的轻量级 J2EE 架构

```
  public List query(){
    return DAi.query();}
}
Public class DAO implements DAinterface{
public list query(){
  SQL 的查询语句;}
}
```

Spring 框架将根据 XML 配置文件为业务构件 Business 提供数据访问构件 DAO 的引用,Business 构件只需暴露相应的 bean 属性 DAi,不需要对具体的 DAO 构件进行声明。

Spring 框架通过模板类 JdbcTemplate 为数据访问层构件提供了便利的实现途径,以查询为例:

```
Final List products=new ArrayList();
JdbcTemplate jdbcTemplate=
     new JdbcTemplate(dataSource);
jdbcTemplate.query(
"select pr_id,pr_name from product",new RowCallbackHandler(){
  public void processRow(ResultSet rs)
  throws SQLException{
  Product product=new Product();
  product.SetId(rs.getInt("id"));
  product.SetName(rs.getString("name"))
  products.add(product);
}});
```

数据访问构件只需为 JdbcTemplate 提供数据源 dataSource,就能完成查询、更新等操作,免去了建立和关闭数据连接、数据访问异常处理等繁琐的工作。

Spring 框架提供了基于 AOP 技术的事务、日志管理等系统级服务,只要在 XML 文件中进行简单的配置即可使用,JavaBean 构件不需关注这些服务的具体实现。以管理 3.2 节 DAO 构件中的事务为例,XML 文件可配置如下:

```
<bean id="myTransactionManager"
 class="org.springframework.transaction.jta.JtaTransactionManager">
</bean>
<bean id="DAOTarget" class="DAO"></bean>
<bean id="myTransactionInterceptor"
   class = "org.springframework.transaction.interceptor.TransactionInterceptor">
     <property name="transactionManager">
     <ref bean="myTransactionManager"/>
   </property>
</bean>
```

```
<bean id="DAO"
  class="org.springframework.aop.framework.ProxyFactoryBean">
  <property name="interceptorNames">
  <value>myTransactionInterceptor,DAOTarget
  </value>
  </property>
</bean>
```

事务拦截器 myTransactionInterceptor 在 DAO 构件的方法执行之前启动 myTransactionManager 管理的事务,在方法执行完毕后再由 myTransactionManager 提交或者回滚事务,DAO 构件本身不需要编写事务处理代码。

4 基于 Spring 框架的轻量级 J2EE 架构的应用

利用基于 Spring 框架的轻量级 J2EE 架构方案为某省财政厅实现了用款单位对帐系统,并且使该系统与某省财政厅正在使用的国库集中支付系统进行了成功的集成。单位对帐系统的总体结构如图 5 所示。

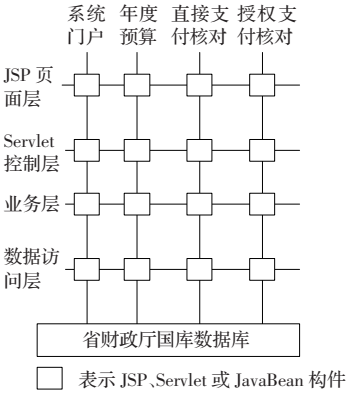

图 5 单位对帐系统的总体结构

系统总体上呈正交化体系结构,从 J2EE 规范的技术角度出发,将系统在横向上划分为表现层(包括 JSP 页面层和 Servlet 控制层)、业务层、数据访问层;从包含的功能角度出发,系统在纵向上划分为系统门户、年度预算、直接支付核对、授权支付核对。以“年度预算”功能为例进行说明。

4.1 表现层

该层由 JSP 页面层和 Servlet 控制层组成。JSP 页面用于将数据显示在浏览器中,Servlet 构件接受用户请求,结合业务构件处理用户请求后,将结果返回 JSP 页面。

用户与浏览器交互的请求首先发送到 DispatchServlet 中,DispatchServlet 主要用于分发请求,它定义在 web.xml 中,如下所示:

```
<servlet>
    <servlet -name >gkzf </servlet -name >  <servlet -class >org.
springframework.web.servlet.DispatcherServlet
    </servlet-class>
</servlet>
```

上述 XML 定义了一个名为 gkzf 的 DispatchServlet。它根据在gkzf-servlet.xml(该 XML 文件是整个系统的配置文件,定义了页面构件与控制构件、控制构件与业务构件和业务构件与数据访问构件间的对应关系)中定义的页面构件与 Servlet 控制构件的对应关系,将请求转发到相应的控制构件进行处理。

“年度预算”功能的页面构件和 Servlet 控制构件的关系如下所示:

```
<bean id="urlMapping"
  class = "org.springframework.web.servlet.handler.SimpleUrlHan-
dlerMapping">
<property name="mappings">
<props>
<prop key="/Budget.jsp">BudgetController</prop>
</props>
</property>
</bean>
<bean id=" BudgetController"
class="cn.edu.jxnu.gkzf.web.BudgetController"/>
```

Budget.jsp 的请求将被映射到名为 BudgetController 的控制构件进行处理。

4.2 业务层

业务层构件一般被控制层构件调用,控制层构件以接口的形式声明所需的业务层构件的服务,由 Spring 框架根据 gkzf-servlet.xml 中的配置实例化业务层构件(BudgetPlanManager),并注入到控制层构件(BudgetController)中,如下所示:

```
public class BudgetController extends
  MultiActionController {
  private IBudgetPlanManager budgetPlanMan;
  public void
setIBudgetPlanManager(IBudgetPlanManager bdm){this.budgetPlan-
Man=bdm;
  }
}
```

gkzf-servlet.XML 中定义了控制层构件要调用的业务构件:BudgetPlanManager

```
<bean id="budgetController"
class="cn.edu.jxnu.gkzf.web.BudgetController">
  <property name="budgetPlanManager">
      <ref bean="budgetPlanMan"/>
  </property>
</bean>
<bean id="budgetPlanMan"
class="cn.edu.jxnu.ainet.gkzf.bus.BudgetPlanManager">
```

4.3 数据访问层

数据访问层构件由业务层构件调用,Spring 框架将根据 gkzf-servlet.xml 中的配置,将数据访问层构件(EnterpriseDao-Jdbc)注入到业务构件(BudgetPlanManager)中,如下所示:

```
public class BudgetPlanManager {
  private IEnterpriseDao enterpriseDao;
    //声明需引入数据访问构件提供的服务
    public void setEnterpriseDao (IEnterpriseDao epd){this.enter-
priseDao=epd;
  }
}
```

gkzf-servlet.xml 中定义了业务层构件要调用的数据访问构件:EnterpriseDaoJdbc

```
<bean id="budgetPlanMan"
class="cn.edu.jxnu.gkzf.bus.BudgetPlanManager">
  <property name="enterpriseDao">
    <ref bean="enterpriseDao"/>
  </property>
```

(下转 133 页)