

基于 Tapestry + Spring + Hibernate 框架的 Web 应用

龚雪冰, 何彪

(华南理工大学 计算机学院, 广东 广州 510640)

摘 要: Web 应用程序可以分为表示层、业务层、持久层和领域模型层。按照这种分层结构, 分别介绍了 Tapestry, Spring 和 Hibernate 三种开源框架。Tapestry 使得程序具有一致性的结构, 通过开发 Tapestry 组件, 增强了代码的复用和程序的健壮性; Spring 实现了业务层和表现层的分离; Hibernate 大幅度减少开发时人工使用 SQL 和 JDBC 处理数据的时间。结合项目实例介绍了如何整合这三种框架构建 Web 的应用。应用这种整合框架, 可以使系统层次清晰, 并实现了层之间的解耦, 各层之间可以独立开发, 极大地提高了开发效率。

关键词: Tapestry; Spring; Hibernate; 框架

中图分类号: TP311.5

文献标识码: A

文章编号: 1673-629X(2007)04-0131-05

Web Application Based on Tapestry + Spring + Hibernate Framework

GONG Xue-bing, HE Biao

(Computer College, South China Univ. of Techn., Guangzhou 510640, China)

Abstract: Web application can be divided into presentation layer, business layer, persistence layer and domain model layer. According to such layered structure three open source frameworks, Tapestry, Spring and Hibernate, are briefly introduced in this paper. Tapestry makes program have consistent structure, and by developing Tapestry component the reuse of code and robustness of program are enhanced. Spring realizes the separation of business layer and presentation layer. Hibernate greatly reduces time of using SQL and JDBC to process data. Finally, an example is given to introduce how to integrate the three frameworks to develop Web application. With this integrated framework, it makes the layers of application clear and decouples them. Different layer can be developed respectively, which largely improve the efficiency of software development.

Key words: Tapestry; Spring; Hibernate; framework

0 引言

随着我国社会信息化程度不断提高, 越来越多的软件开发人员需要开发 Web 应用程序。目前网络编程主要有两大技术体系: 基于 J2EE 的网络开发和基于 .NET 的网络开发。J2EE 以其开放性、灵活性、安全性和技术成熟度在企业级信息系统的开发中占据了重要的地位。J2EE 提供了一种基于组件的方法, 对于一个多层的应用模型, 根据功能将其应用逻辑划分为组件。J2EE 规范中定义了应用客户组件、EJB 组件、Web 组件和 Applet^[1]。组件通常设计成框架的一个组成部分, 框架的作用就是将独立的组件组合在一起形成应用程序进行发布。选择了一个好的框架, 开发人员基本上只需要在限制范围内编写相应的组件, 由框

架管理这些组件的协作, 从而得到结构良好、质量可靠的 Web 应用^[2]。一个好框架具备以下几点: 减轻开发者处理复杂问题的负担; 内部定义为可扩展的; 有一个强大的用户群支持。好的框架一般针对性地处理某一类问题, 这样在 Web 应用中可能存在几个层, 需要使用几个特定的框架。如何整合框架以便让每个层以一种松耦合的方式彼此相互作用, 而不管低层的实现细节, 是一个挑战^[3]。

1 Web 应用程序的分层

大部分 Web 应用至少可分为四层, 如图 1 所示。这四层是: Presentation layer(表示层)、Persistence layer(持久层)、Business layer(业务层)、Domain model layer(领域模型层)。每一层在处理程序上都有一明确的任务, 不应该在功能上与其它层混合; 每个层要与其它层分开, 但它们之间存在一个通信接口^[4]。

●表示层: 框架有 Tapestry, Struts, JSF 等。

* 关注请求/响应动作;

收稿日期: 2006-07-09

作者简介: 龚雪冰(1977-), 男, 江西南昌人, 硕士研究生, 研究方向为网络软件开发环境; 何彪, 副教授, 硕士生导师, 研究方向为网络软件开发环境。

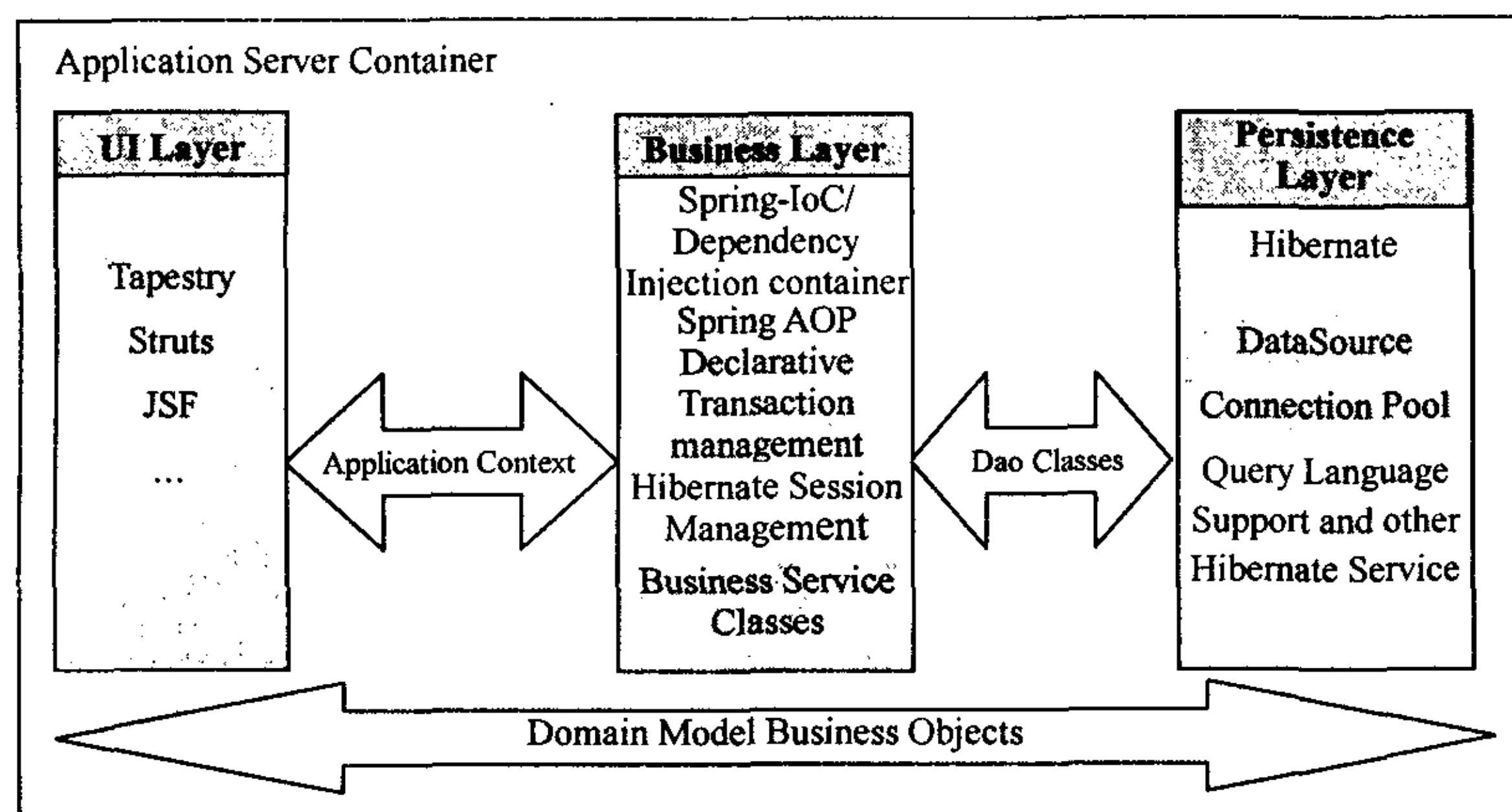


图 1 Web 应用程序分层示意图

vlet 容器(例如 Tomcat)或包含 servlet 容器的应用服务器中(如 Jboss, Websphere, 或者 WebLogic)。Tapestry 应用其实是由一系列页面组成,而每个页面是由可以复用的组件构成^[5]。这提供了一致性的结构,使得 Tapestry 可以负责创建和转发 URL、客户端或服务端的持久化存储、用户输入校验、本地化/国际化以及异常报告。开发 Tapestry 应用程序包括用纯 HTML 创建 HTML 模板并通过 XML 描述文件将其和少量 Java 代码结合。在 Tapestry 中,你用对象、方法、和

属性创建程序,而不是根据 URLs 和查询参数。Tapestry 将真正的面向对象运用于 Web 应用程序的开发。使用 Tapestry 你很容易创建你自己的组件,这增强了代码的复用和程序的健壮性。Tapestry 很容易和 J2EE, HiveMind 和 Spring 等后端进行整合^[6]。

2.2 Spring 框架简介

Spring 是由七个定义良好的模块组成的分层架构,是一个服务于所有层面的应用程序框架。Spring 提供了 bean 的配置基础, AOP 的支持, JDBC 的提取框架, 抽象事务支持等, 它有一个非常显著的特点: 在某个层面上如果你不需要 Spring 的支持, 可以不使用它的这部分功能。现在使用的 WebWork, Struts, Tapestry 或其它的 UI 框架的前端程序可极佳地与基于 Spring 的中间层进行集成, 使你使用 Spring 的事务处理特性。从它的设计理念, 可以看到 Spring 帮助你真正地实现了逻辑层和 Web 层的分离。Spring 模块构建在核心容器之上, 核心容器定义了创建、管理和配置 bean 的方式。组成 Spring 框架的每个模块都可以单独存在, 或者与其它模块联合。相对与 EJB 来说, Spring 是一个轻量级的 J2EE 应用开发框架。它为已建立的企业级应用提供了一个轻量级的解决方案, 这个方案包括声明性事务管理, 通过 RMI 或 Web Service 远程访问业务逻辑, mail 支持工具以及对于数据和数据库之间持久层各种配置的支持。Spring 还提供了一个 MVC 应用框架, 可以通过集成 AOP 透明地嵌入你的软件。Spring 还有一个优秀的异常处理体系, 这个体系可以自动地从属性异常体系进行映射^[7]。

Spring 的优点集中体现在以下几个方面:

- * 利用依赖注入思想组装代码, 提高系统扩展性、灵活性, 实现插件式编程。
- * 利用 AOP 思想集中处理业务逻辑, 减少重复代码, 构建优雅的解决方案。
- * 利用其对 Hibernate 的 SessionFactory、事务管

- * 处理来自模型的 UI 提交;
- * 包含格式逻辑和与业务无关的校验逻辑;
- * 处理从其它层抛出的异常。

●持久层: 框架有 Hibernate, iBATIS, JDBC, JDO 等。

* 用于和持久化存储如关系型数据库之间的通信;

- * 提供一种查询语言;
- * 可能具有 O/R 映射能力。

●业务层: 框架有 Spring, PicoContainer 等。

- * 处理应用程序的业务逻辑和业务验证;
- * 管理事务;
- * 预留和其它层交互的接口;
- * 管理业务层对象之间的依赖;
- * 增加在表现层和持久层之间的灵活性, 使它们互不直接通讯;

* 从表现层中提供一个上下文(context)给业务层获得业务服务(business services);

- * 管理从业务逻辑到持久层的实现。

●领域模型层。

- * 含有上面各层所需的对象;
- * 含有和其它领域对象之间的复杂的关系;
- * 可能有 ORM 映射;
- * 领域对象只依赖于其它领域对象。

文中将介绍基于 Tapestry + Spring + Hibernate 框架的 Web 应用。

2 基于 Tapestry + Spring + Hibernate 的软件架构及实现

2.1 Tapestry 框架简介

Tapestry 框架是一个位于 Java servlet 容器和 Tapestry 应用程序之间的层。Tapestry 不是一个独立运行的服务器, 它是一个 servlet 的扩展, 它运行于 ser-

理的封装,更简洁地应用 Hibernate。

2.3 Hibernate 框架简介

Hibernate 是一个面向 Java 环境的对象/关系数据库映射工具。对象/关系数据库映射(ORM)用来把对象模型表示的对象映射到基于 SQL 的关系模型结构中去。Hibernate 不仅管理 Java 类到数据库表的映射,还提供数据查询和获取的方法。可以大幅度减少开发时人工使用 SQL 和 JDBC 处理数据的时间。Hibernate 技术本质上是一个提供数据库服务的中间件,它的架构如图 2 所示。

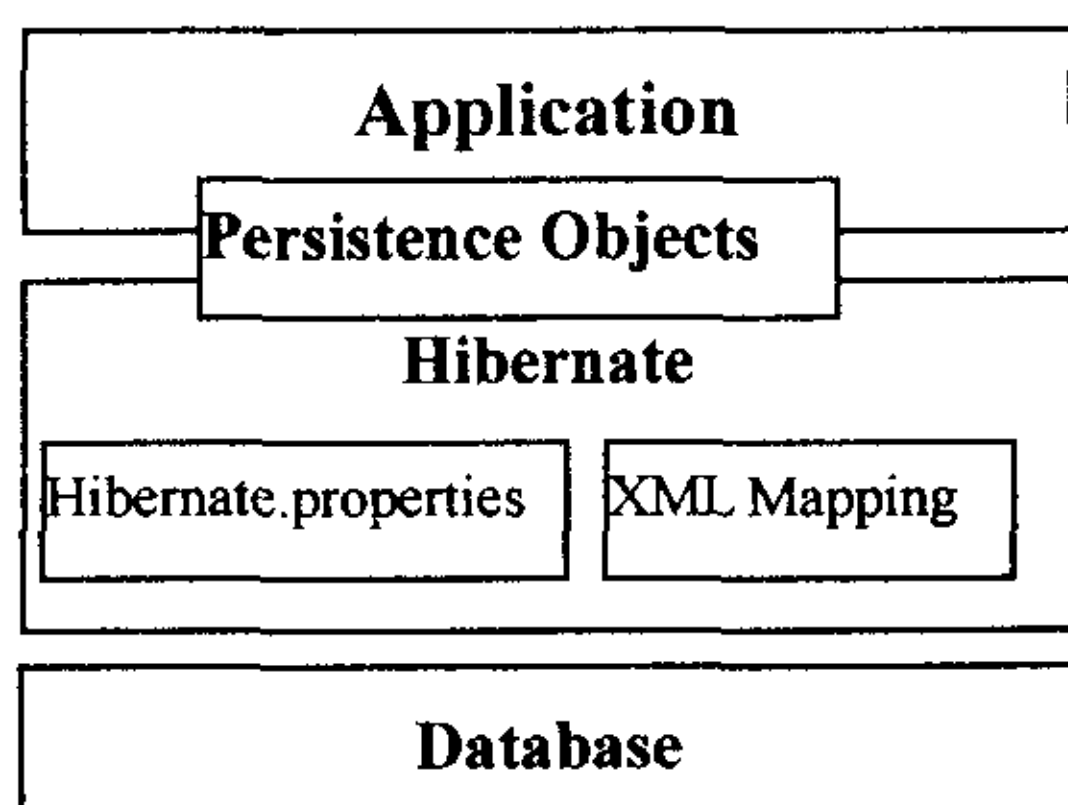


图 2 Hibernate 体系结构

图 2 显示了 Hibernate 的工作原理,它是利用数据库以及其它一些配置文件如 Hibernate.properties, XML Mapping 等来为应用程序提供数据持久服务^[8]。

Hibernate 使用了 J2EE 中如下技术:JDBC, JTA, JNTI。其中 JDBC 是支持关系数据库操作的一个基础层,它和 JNDI, JTA 一起结合,使得 Hibernate 可以方便地集成到 J2EE 应用服务器中。Hibernate 的优点集中体现在以下几个方面:

- * 减少了编写 SQL 语句工作量,由代码比较可以看出,数据表可以像对象一样被操作,这样使代码显得更加简洁,可读性也增强。

- * Hibernate 封装了数据库访问、事务管理、数据缓存等工作,省去了自己编写这些代码。

- * 数据表数据映射到对象中,能更好地在系统各层传输。

2.4 项目实例

我们使用 Tapestry + Spring + Hibernate 框架构建了某省出入境管理信息系统(简称 EECMIS)。客户浏览器通过 HTTP 与系统进行交互。在 Web 层,系统提供了前端控制器,比如 Tapestry, Flash Remoting, Spring Web MVC 等。借助于它们,系统能够有效、准确地将用户请求分发给目标服务,这些服务宿主在 Spring IoC 容器中。Spring IoC 容器负责提供业务服务,因此整个业务逻辑层交付给 IoC 容器处理。整个系统的业务逻辑层采用 POJO 实现。这些 POJO 将使用到 Spring 提供的 J2EE 服务抽象,比如 Hibernate、JMS、事务、JDBC,而且 POJO 本身不必去理会资源的具体操作,这些

工作由 Spring 完成。POJO 在执行持久化操作时,会借助于 O-R Mapping 技术实现。系统的持久化层大量地采用了 O-R Mapping 技术,包括 VDB 本身。

先介绍 Tapestry 和 Spring 的整合。在 Tapestry 框架里的 web.xml 文件中有这样一段代码:

```

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/classes/conf/spring/appContext.xml</param-value>
</context-param>
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
  
```

这样,在 Web 应用程序启动时, Spring 的 ApplicationContext 就可以通过 ContextLoaderListener 注册。这个 Listener 检查 ContextConfigLocation 参数,将 Spring 的 ApplicationContext 载入。然后 Tapestry 框架就可以通过 Engine 加载 Spring 的服务,在 Engine 里获得 ApplicationContext(),并将其写到 Tapestry 的 global 对象里边去。如下面代码所示:

```

public class EecmisEngine extends BaseEngine {
  public static final String APPLICATION_CONTEXT_KEY = "appContext";

  protected void setupForRequest(RequestContext context) {
    super.setupForRequest(context);
    ApplicationContext ac = (ApplicationContext) Global.getAppContext();
    if (ac == null) {
      ServletContext servletContext = context.getServlet().getServletContext();
      ac = WebApplicationContextUtils.getWebApplicationContext(servletContext);
      Global.setAppContext(ac);
      Global.setServletContext(servletContext);
    }
  }
}
  
```

然后在 eecmis.application 中指定 Engine:

```

<application name="eecmis" engine-class="cn.hnisi.eecmis.common.EecmisEngine">
  ...
</application>
  
```

这样,就可以在某个 Tapestry 页面如: Gyyw302 Main 的 page 文件中获得 ApplicationContext 中的名为“gyywService”的 Bean 了,如以下代码所示:

```

<page-specification class="cn.hnisi.eecmis.jbyw.gyyw.Gyyw302Main">
  ...<property-specification name="gyywService"
  
```



```

    type = "cn.hnisi.eecmis.service.IGyywService">
    global.appContext.getBean("gyywService")
... </property-specification>
</page-specification>

```

在该页面的 Java 文件里增加一个 abstract getter:

```
public abstract IGyywService getGyywService();
```

之后就可以调用底层的“gyywSevice”服务了。

接下来以 gyyw(公用业务)子模块为例介绍 Spring 和 Hibernate 的整合。

首先,用 Hibernate 映射工具编写 Hibernate 映射文件和持久化对象。所有映射文件和持久化对象放在 WEB-INF/classes/cn/hnisi/eeecmis/persistence/jbyw/gyyw 目录下。然后编写和配置 DAO(Spring 就是通过 DAO 和持久层进行通信的)。Spring 强调面向接口编程,所以首先编写一个 DAO 的接口:IGyywDao。在 IGyywDao 中定义了一系列对数据操作的接口方法;接下来编写实现类,代码如下:

```
public class GyywDaoImpl extends AbstractDaoSupport implements IGyywDao {...
```

GyywDaoImpl 通过扩展 Spring 提供的 Hibernate 支持类 HibernateDaoSupport 而建立, HibernateDaoSupport 封装了 HibernateTemplate, 而 HibernateTemplate 封装了 Hibernate 所提供几乎所有的数据操作方法,如 execute(HibernateCallback action), load(Class entityClass, Serializable id), save(final Object entity) 等等。所以我们的 DAO 只需要简单地调用父类的 HibernateTemplate 就可以完成几乎所有的数据库操作了。

由于 Spring 通过代理 Hibernate 完成数据层的操作,所以原 Hibernate 的配置文件 hibernate.cfg.xml 的信息也转移到 Spring 的配置文件中:

```

<!-- Hibernate 会话工厂配置 -->
<bean id="sessionFactoryA" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="lobHandler"><ref bean="oracleLobHandler"/></property>
    <property name="dataSource"><ref bean="dataSource"/></property>
    <property name="mappingDirectoryLocations">
        <list>
            <value>WEB-INF/classes/cn/hnisi/eeecmis/persistence/jbyw/gyyw</value>
        </list>
    </property>
    <property name="hibernateProperties">
        .....
    </property>

```

```
</bean>
```

会话工厂类用 Spring 提供的 LocalSessionFactoryBean 维护,它注入了数据源和资源映射文件,此外还通过一些键值对设置了 Hibernate 所需的属性。

在业务层定义了一个服务接口 IGyywService,这个服务就是前面 Tapestry 框架通过“global.appContext.getBean(“gyywService”)”得到的服务。然后编写服务实现类:

```
public class GyywServiceImpl implements IGyywService {...},
```

在这个服务类中有一个

```

    setter:public void setGyywDao(IGyywDao gyywDao) {
        this.gyywDao = gyywDao;
    }

```

以后会通过 Spring 的配置文件将前面编写的 GyywDaoImpl 注入。

接下来会在 Spring 的配置文件中为 gyywSevice 配置声明性的事务:

```

<bean id="transactionManagerA"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactoryA"/>
</bean>
<!-- 事务处理的 AOP 配置 -->
<bean id="gyywTransactionProxy" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean" abstract="true">
    <property name="transactionManager">
        <ref bean="transactionManagerA"/></property>
    <property name="transactionAttributes">
        <props>
            <prop key="queryDB *">PROPAGATION_REQUIRED,readOnly</prop>
            ...
        </props>
    </property>
</bean>
<bean id="gyywDao" class="cn.hnisi.eecmis.jbyw.gyyw.impl.GyywDaoImpl">
    <property name="sessionFactory" ref="sessionFactoryA"/>
    <property name="jdbcTemplate" ref="jdbcTemplate"/>
</bean>
<bean id="gyywService" parent="gyywTransactionProxy">
    <property name="target" ref="gyywServiceTarget"/>
</bean>
<bean id="gyywServiceTarget"
class="cn.hnisi.eecmis.service.impl.GyywServiceImpl">

```



```
<property name="gyywDao" ref="gyywDao"/>
</bean>
```

Spring 的事务配置包括两个部分:其一,定义事务管理器 transactionManagerA,使用 HibernateTransactionManager 实现事务管理;其二,对各个业务接口进行定义,其实 gyywTransactionProxy 和 gyywService 是父子节点的关系,本来可以将 gyywTransactionProxy 定义的内容合并到 gyywService 中一起定义,但由于系统有多个业务接口需要定义,将这些业务接口定义内容的共同部分抽取到一个父节点中,然后在子节点中通过 parent 进行关联,就可以大大简化业务接口的配置了。父节点 gyywTransactionProxy 注入了事务管理器,此外还定义了业务接口事务管理的方法(允许通过通配符的方式进行匹配声明),有些接口方法仅对数据进行读操作,而另一些接口方法需要涉及到数据的更改。对于前者,可以通过 readOnly 标识出来,这样有利于操作性能的提高,需要注意的是由于父类节点定义的 Bean 仅是子节点配置信息的抽象,并不能具体实例化一个 Bean 对象,所以需要特别标注为 abstract = "true"。GyywServiceImpl 作为一个目标类注入事务管理器中,而其所需的 gyywDao 通过注入 GyywDaoImpl 实现。

通过以上介绍可以发现正如图 1 所示,Tapestry 框架通过 Spring 框架的 Application Context 调用底层服务,而 Spring 框架通过将 Hibernate 的 sessionFactory 注入 DAO 以及将 transactionmanager 注入 service,实现对底层数据库的操作。

(上接第 85 页)

助业务的干扰,整个开发团队中只需要很少的一部分人负责辅助业务方面开发即可,有利于成员之间的分工合作,发挥团队的作用。

由于 AOP 组合器的实现引入了 Java 动态代理和反射机制,不可否认会对性能产生一定的影响,但这一点开销与它所带来的清晰架构相比是很值得的,另外还可以借鉴 EJB 对象池的思想在组合器中为业务对象建立适当缓存优化性能。

4 总 结

文中框架的设计尚显粗糙,例如,框架中对所有主业务方法都进行事务处理,而在实际应用中,很多业务方法仅仅只是读数据,无需事务处理。比较简单的解决办法是对业务方法名做一定约束,方法名以 get、find、query 等打头表示该业务不需要事务处理,否则需要,那么在 com.aop.PointHandler 中对方法名作判断

3 结束语

Tapestry + Spring + Hibernate 框架是一种非常优秀的基于 J2EE 平台的 Web 应用开发框架,在业界已经有了许多成功的案例。应用 Tapestry + Spring + Hibernate 框架构建 Web 应用程序,使得该系统结构层次清晰并实现了层之间的解耦,开发过程中层与层之间的工作几乎完全独立,极大地提高了系统的开发效率。同时也提高了系统的可重用性和灵活性,为日后的扩展和维护留有很大余地。

参考文献:

- [1] 宋秀琴,侯殿昆,方中纯.基于 Struts 和 Hibernate 的 Web 应用的构建[J].微计算机信息,2005,21(11):125-127.
- [2] 孙刚,孟祥武.基于 Struts 框架的 J2EE Web 应用[C]//第六届 JAVA 技术及应用大会论文集.北京:电子工业出版社,2003:124-129.
- [3] 黄华.框架技术在 web 系统开发中的应用[J].微机发展,2005,15(5):77-79.
- [4] Egle M. Wiring Your Web Application with Open Source Java [EB/OL]. 2004-04-07. <http://www.onjava.com/pub/a/onjava/2004/04/07/wiringwebapps.html>.
- [5] Howard M, Ship L. Tapestry in Action[M]. [s.l.]: Manning, 2004.
- [6] What is Tapestry[EB/OL]. 2006-01-25. <http://jakarta.apache.org/tapestry>.
- [7] Spring Framework 开发参考手册[EB/OL]. 2005-06. Spring 中文论坛.
- [8] Hibernate-version 3.0.4[EB/OL]. 2005-03. <http://www.hibernate.org>.

可选择是否组合事务处理。

AOP 仍处在不断发展之中,受到越来越多人的关注,是 OOP 之后的又一次编程语言的重要创新,不过和当初一样,AOP 面临的还是不同的标准和想法。

参考文献:

- [1] Amandxp. AOP 技术简介[EB/OL]. 2005. <http://dev2dev.bea.com.cn/bbs>.
- [2] Nyberg G, Patrick R. 精通 BEA WebLogic Server 构建与部署 J2EE 应用的最佳策略[M]. 北京:电子工业出版社, 2004.
- [3] 孙卫琴.精通 Hibernate:Java 对象持久化技术详解[M]. 北京:电子工业出版社,2006.
- [4] Wxh1028. 用 Java 动态代理实现 AOP[EB/OL]. <http://www.yesky.com/SoftChannel/72342371961929728/20041014/1864192-1.shtml>.
- [5] Jacobson I, Pan Wei Ng. Aspect-Oriented Software Development with Use Cases[M]. 北京:电子工业出版社,2003.