文章编号：1005-8451(2007)10-0041-04

基于Spring与Hibernate的J2EE持久层解决方案

谭 待[1]，谭人杰[2]

(1.华中科技大学 计算机科学与技术学院，武汉 430074；2.武汉铁路局 信息技术处，武汉 430071)

摘 要：通过Spring与Hibernate的整合，探讨如何有效地应用Spring的核心技术IOC和AOP实现J2EE应用的持久层解决方案。并从实用角度阐明用Spring IOC动态注入Hibernate SessionFactory和数据源，用Spring DAO简化Hibernate的持久化操作；用Spring AOP实现声明式事务管理。

关键词：Spring；IOC；AOP；Hibernate；J2EE

中图分类号：TP302 文献标识码：A

Solution of J2EE persistent tier based on integrating Spring with Hibernate

TAN Dai[1], TAN Ren-jie[2]

(1.College of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074, China;
2.Computer Center of Wuhan Railway Administration, Wuhan 430071, China)

Abstract: It was presented a solution of J2EE persistent framework by integrating Spring with Hibernate. The solution was provided by Applying Spring IOC and AOP. Firstly, the object dependencies of Hibernate SessionFactory and DataSource were injected by Spring IOC. Secondly, the Hibernate CRUD was addressed by Spring DAO support in a universal way. Finally, the transaction management was added to business logical methods by Spring AOP in the way of declaration.

Key words: Spring; IOC; AOP; Hibernate; J2EE

Spring是近年来引人注目的开源框架，用于简化传统的J2EE开发过程。Spring与Hibernate配合使用可以减少编程代码量，加快J2EE应用的开发速度。

Hibernate是Java平台上一个功能全面、开源的OR映射框架。Hibernate编程基于POJO（Plain Ordinary Java Object）和Hibernate映射。在开发J2EE应用程序时，用POJO表示域对象，与数据库中的表一一对应。用映射文件（*.hbm.xml）文件定义Java类与数据库表之间的映射关系。Hibernate根据映射关系，用JDBC技术实现POJO与数据库对应表之间的同步。可以说，Hibernate是一个脱离了应用服务器的类似于EJB的轻量级软件框架，使我们能够以面向对象方式轻松访问数据库。

Spring是一个提供Web应用表示层、业务层和持久层解决方案的开源框架。

Spring的表示层是一个MVC框架，与Struts类似，功能比Struts更强，且更易使用。根据需要，也可与其它表示层框架集成。

Spring业务层有IOC容器和AOP支持。

IOC指控制反转（Inversion of Control），即将Java对象获得依赖对象的方式反转，由容器来控制。所以，IOC又称为依赖注入DI（Dependency Injection），也就是说，当一个对象需要引用另一个对象时，由容器Spring动态实现这种引用。用Java术语来讲，就是创建引用对象的实例并赋初值。其好处是去掉了这些用于构造对象的硬代码，降低了模块间的耦合，使类可以重用。

AOP指面向方面编程（Aspect Oriented Programming）。在不修改目标对象的前提下，为目标类抽象出一个接口，再写一个实现类或直接使用Spring的动态代理类，以切面代码对目标对象进行包裹，实现声明式事务等与业务逻辑无关的服务，降低模块间的耦合，增加系统的灵活性。

此外，Spring持久层还提供了DAO模板支持Hibernate等ORM框架，使Hibernate的应用更为简单。下文以银行转帐为例说明整合过程。

1 建立Spring与Hibernate的集成环境

1.1 加入开源框架Spring和Hibernate

Spring与Hibernate均为轻量级框架，不要求应用服务器的支持。只需将框架提供的jar包路径加入

收稿日期：2007-05-25

作者简介：谭 待，在读本科生；谭人杰，高级工程师。

Java应用的classpath中，或将jar包拷贝到Web应用的WEB-INF\lib目录下。

1.2 整合配置文件

将Hibernate连接数据库的配置信息（DataSource和SessionFactory）集成到Spring的配置文件applicationContext.xml中，由Spring统一管理。其优点是避免硬编码的资源查找代码与应用程序对象的紧密耦合。Spring允许在application context中以bean的方式定义JDBC DataSource和Hibernate SessionFactory的数据访问资源。任何需要进行资源访问的应用程序直接引用这些由Spring统一管理的对象，不再需要单独实例化和赋初值。下面是对MySQL数据库的连接配置定义：

```
<bean id="dataSource"
    class="org.apache.commons.dbcp.
BasicDataSource">
    <property name="driverClassName">
        <value>com.mysql.jdbc.Driver</
value>
    </property>
    <property name="url">
        <value>jdbc:mysql://localhost:
3306/test</value>
    </property>
    <property name="username">
        <value>root</value>
    </property>
    <property name="password">
        <value>weblogic</value>
    </property>
</bean>
<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.
LocalSession-FactoryBean">
    <property name="dataSource">
        <ref bean="dataSource"></ref>
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.
dialect">
                org.hibernate.dialect.
MySQLDialect
            </prop>
        </props>
    </property>
</bean>
```

纳入Spring管理的Java类统称为Bean，用id命名，可以重用。在Spring配置文件中定义的dataSource和sessionFactory，将由Spring以动态注入方式对数据库的连接进行管理，由此降低应用Hibernate的复杂性，减少编程工作量。

Spring集成了Hibernate配置信息并提供了一个现成的LocalSessionFactoryBean，不再需要在Hibernate下定义配置文件和建立HibernateSessionFactory。

1.3 创建POJO和Hibernate映射文件

创建应用的域对象POJO，这是一个实现了Serializable的普通Java类或称JavaBean：

```
public class Account implements java.io.Serializable
{
  private String accountNo;
  public String getAccountNo() {
      return this.accountNo;
  }
  public void setAccountNo(String accountNo) {
      this.accountNo = accountNo;
  }
  ……
}
```

创建Hibernate映射文件account.hbm.xml，定义POJO与数据库表的对应关系：

```
<hibernate-mapping>
    <class name="com.pnx.bean.Account"
table="account" >
      <id name="accountNo" type="java.lang.
String">
        <column name="accountNo" length="45" />
        <generator class="assigned" />
      </id>
      ……
    </class>
</hibernate-mapping>
```

在配置文件applicationContext.xml中注册Hibernate映射文件的路径：

```
<property name="mappingResources">
  <list>
```

```
        <value>com/pnx/bean/Account.hbm.xml</
value>
      </list>
    </property>
```

2　Spring IOC应用

应用程序访问数据库首先需获取Hibernate的Session对象。Spring根据配置文件中的定义，以IOC方式为应用程序动态注入Hibernate的Session对象。一旦有了Session对象，应用程序就可以使用Session对象提供的各种方法，以OO方式轻松访问数据库。

使用Spring集成Hibernate的优点：不再需要手工实现单子设计模式（Singleton）HibernateSessionFactory类，而由Spring动态注入SessionFactory对象。在Spring中，依赖关系用JavaBean的setter方法来表示。也就是说，只需在JavaBean中写一个setter方法，并在Spring配置文件中声明对应属性，当应用程序调用Spring的getBean()方法时，由Spring自动调用setter方法进行动态注入，并以单子设计模式返回SessionFactory实例，同时封装了异常处理代码。

2.1　在应用AccountDAO中定义属性sessionFactory及其setter方法

```
private SessionFactory sessionFactory;
public void setSessionFactory(SessionFactory
sessionFactory) {
this.sessionFactory = sessionFactory;
}
```

2.2　在Spring配置文件中声明为accountDAO

```
<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.
LocalSessionFactoryBean">
……
</bean>
<bean id="accountDAO" class="com.pnx.dao.
AccountDAO" abstract="true">
<property name="sessionFactory">
      <ref bean="sessionFactory"/>
</property>
</bean>
```

2.3　在应用程序中用getBean()获取sessionFactory对象

```
ApplicationContext ctx =
      new FileSystemXmlApplication
Context(
      "file:src/applicationContext.xml");
IAccountDAO dao = (IAccountDAO)ctx.getBean
("accountDAO");
```

3　用Spring DAO简化Hibernate的持久化操作

Spring提供DAO模板简化Hibernate DAO编程，使用统一的CRUD，去掉大量的try/catch/finally异常处理代码，也不需要在方法之间传递连接对象。下面用模板重新定义AccountDAO类。

3.1　继承父类HibernateDaoSupport

```
public class AccountDAO extends HibernateDao-
Support implements IAccountDAO {
```

3.2　用父类的getHibernateTemplate()方法取得模板HibernateTemplate实例

当引用getHibernateTemplate()方法时，Spring框架以单子模式（Singleton Pattern）获取HibernateTemplate实例，并动态注入由AccountDAO的Bean属性定义的sessionFactory对象：

```
<bean id="accountDAO" class="com.pnx.spring.
dao.impl.AccountDAO">
      <property name="sessionFactory">
            <ref bean="sessionFactory"></ref>
      </property>
</bean>
```

3.3　用统一的方法实现CRUD

有了HibernateTemplate实例及其sessionFactory，就可以用简单统一的方法实现CRUD：

```
getHibernateTemplate().save(transientInstance);
getHibernateTemplate().delete(persistentInstance);
……
```

4　Spring AOP应用

Spring使用动态的AOP Proxy对象提供切面服务，声明式的事务管理（declarative transaction management）是一个典型的应用。使用声明式事务可以将J2EE应用程序中大量的事务处理及异常处理代码减掉，使业务逻辑更加清晰和易于重用。使用声明式事务的另一个好处是J2EE所强调的可移植

性，如果从单一数据库的事务处理移植到分布式事务处理，只需要修改Spring相关配置，而不需要修改每一个与事务相关的程序代码，减少维护工作量。

使用Spring AOP要引入两个概念：一是面向接口的编程，因为动态代理是针对接口的；二是引入服务层，在服务层可以重用DAO模块，组合新的业务逻辑，使J2EE应用的可扩展性更好。有了这两点，就可以实现以声明方式注入Spring的事务管理。这里用大家熟悉的银行转帐为例加以说明。

4.1 编写服务层接口和实现类

```
public interface IBank {
        public void transfer(Account fromAccount,
Account toAccount);
}
public class BankService implements IBankService
{
        private IAccountDAO accountDAO;
         public void setAccountDAO(IAccountDAO
accountDAO) {
            this.accountDAO = accountDAO;
}
public void transfer(Account fromAccount, Ac-
count toAccount) {
accountDAO.saveAccount(fromAccount);
accountDAO.saveAccount(toAccount);
}
}
```

类中的transfer()方法重用DAO的saveAcount()方法实现转帐交易。

4.2 定义声明式事务

在Spring配置文件中为transfer()方法定义声明式事务。通过Spring的事务管理代理类，以AOP方式对目标类中的transfer()方法增加事务管理。

（1）增加事务管理器Bean定义

在配置文件中增加事务声明

```
<bean id="transactionManager"
class="org.springframework.orm.hibernate3.
HibernateTransact-ionManager">
<property name="sessionFactory">
        <ref bean="sessionFactory"/>
</property>
</bean>
```

（2）定义一个事务模板，由TransactionProxyFactoryBean提供事务管理，并声明需要事务的方法transfer()

```
<bean id="txTemplate"
 class="org.springframework.transaction.interceptor.
TransactionProxyFactoryBean" abstract="true">
<property name="transactionManager">
        <ref bean="transactionManager"/>
</property>
<property name="transactionAttributes">
<props>
        <prop key="transfer">
         PROPAGATION_REQUIRED
        </prop>
      </props>
    </property>
</bean>
```

（3）改写bankService的定义，由Spring以AOP方式为服务层目标类BankService的方法transfer()增加事务处理

```
<bean id="bankService" parent="txTemplate">
        <property name="target">
          <bean class="com.pnx.service.impl.
BankService" autowire="byName">
        </bean>
    </property>
</bean>
```

5 结束语

Spring和Hibernate都是基于POJO、摆脱EJB容器的轻量级框架。集成Spring与Hibernate实现J2EE持久层应用，不仅可以降低模块间的耦合性，还可以减少70%代码。用Spring实现服务层后，在前端可以与多种表示层框架集成，如Struts，JSF，WebWork等。在铁路专用线项目中，使用Spring DAO与Hibernate集成，实现持久层。在表示层则与JSF集成，实现MVC架构，取得了很好的效果。

参考文献：

[1] Spring Java/J2EE Application Framework Reference Documentation Version 2.0[Z]. 2002.

[2] 谭 待. 在MyEclipse下整合Spring和Hibernate[OB/OL]. http://www.blogjava.net/dyerac/archive/2006/08/04/61805.html，2006，8.