# [applicationContext.xml详解](http://blog.csdn.net/heng_ji/article/details/7022171)

想必用过[Spring](http://www.3y11.com/tags_article.asp?tag_id=spring" \t "http://blog.csdn.net/heng_ji/article/details/_blank)的程序员们都有这样的感觉，**[spring](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "http://blog.csdn.net/heng_ji/article/details/_blank)**把逻辑层封装的太完美了（个人感觉View层封装的不是很好）。以至于有的初学者都不知道Spring配置文件的意思，就拿来用了。所以今天我给大家详细解释一下Spring的applicationContext.xml[文件](http://www.3y11.com/tags_article.asp?tag_id=%E6%96%87%E4%BB%B6" \t "http://blog.csdn.net/heng_ji/article/details/_blank)。Ok，还是通过代码加注释的方式为大家演示：

**以下是详解Spring的applicationContext.xml文件代码：**<!-- 头文件，主要注意一下编码 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_260](https://code.csdn.net/snippets/1749109)[IMG_261](https://code.csdn.net/snippets/1749109/fork)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN""http://www.springframework.org/dtd/spring-beans.dtd"**>**
3. **<beans>**

<!-- 建立数据源 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_262](https://code.csdn.net/snippets/1749109)[IMG_263](https://code.csdn.net/snippets/1749109/fork)

1. **<bean** id="dataSource"class="org.apache.commons.dbcp.BasicDataSource"**>**

  <!-- **[数据库](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "http://blog.csdn.net/heng_ji/article/details/_blank)**驱动，我这里使用的是**[MySQL](http://lib.csdn.net/base/14" \o "MySQL知识库" \t "http://blog.csdn.net/heng_ji/article/details/_blank)**数据库 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_264](https://code.csdn.net/snippets/1749109)[IMG_265](https://code.csdn.net/snippets/1749109/fork)

1. **<**propertyname**propertyname**="driverClassName"**>**
2. **<value>**com.mysql.jdbc.Driver**</value>**
3. **</property>**

  <!-- 数据库地址，这里也要注意一下编码，不然乱码可是很郁闷的哦！ -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_266](https://code.csdn.net/snippets/1749109)[IMG_267](https://code.csdn.net/snippets/1749109/fork)

1. **<property** name="url"**>**
2. **<value>**
3. jdbc:mysql://localhost:3306/tie?useUnicode=true&characterEncoding=utf-8
4. **</value>**
5. **</property>**

  <!-- 数据库的用户名 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_268](https://code.csdn.net/snippets/1749109)[IMG_269](https://code.csdn.net/snippets/1749109/fork)

1. **<property** name="username"**>**
2. **<value>**root**</value>**
3. **</property>**

  <!-- 数据库的密码 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_270](https://code.csdn.net/snippets/1749109)[IMG_271](https://code.csdn.net/snippets/1749109/fork)

1. **<property** name="password"**>**
2. **<value>**123**</value>**
3. **</property>**
4. **</bean>**

<!-- 把数据源注入给Session工厂 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_272](https://code.csdn.net/snippets/1749109)[IMG_273](https://code.csdn.net/snippets/1749109/fork)

1. **<bean** id="sessionFactory"
2. class="org.springframework.orm.hibernate3.LocalSessionFactoryBean"**>**
3. **<property** name="dataSource"**>**
4. **<ref** bean="dataSource" **/>**
5. **</property>**

  <!-- 配置映射文件 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_274](https://code.csdn.net/snippets/1749109)[IMG_275](https://code.csdn.net/snippets/1749109/fork)

1. **<property** name="mappingResources"**>**
2. **<list>**
3. **<value>**com/alonely/vo/User.hbm.xml**</value>**
4. **</list>**
5. **</property>**
6. **</bean>**

<!-- 把Session工厂注入给hibernateTemplate -->  
<!-- 解释一下hibernateTemplate：hibernateTemplate提供了很多方便的方法，在执行时自动建立 HibernateCallback 对象，例如：load()、get()、save、delete()等方法。 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_276](https://code.csdn.net/snippets/1749109)[IMG_277](https://code.csdn.net/snippets/1749109/fork)

1. **<bean** id="hibernateTemplate"
2. class="org.springframework.orm.hibernate3.HibernateTemplate"**>**
3. **<constructor-arg>**
4. **<ref** local="sessionFactory" **/>**
5. **</constructor-arg>**
6. **</bean>**

<!-- 把DAO注入给Session工厂 -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_278](https://code.csdn.net/snippets/1749109)[IMG_279](https://code.csdn.net/snippets/1749109/fork)

1. **<bean** id="userDAO" class="com.alonely.dao.UserDAO"**>**
2. **<property** name="sessionFactory"**>**
3. **<ref** bean="sessionFactory" **/>**
4. **</property>**
5. **</bean>**

<!-- 把Service注入给DAO -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_280](https://code.csdn.net/snippets/1749109)[IMG_281](https://code.csdn.net/snippets/1749109/fork)

1. **<bean** id="userService"class="com.alonely.service.UserService"**>**
2. **<property** name="userDAO"**>**
3. **<ref** local="userDAO" **/>**
4. **</property>**
5. **</bean>**

<!-- 把Action注入给Service -->

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_282](https://code.csdn.net/snippets/1749109)[IMG_283](https://code.csdn.net/snippets/1749109/fork)

1. **<bean** name="/user"class="com.alonely.struts.action.UserAction"**>**
2. **<property** name="userService"**>**
3. **<ref** bean="userService" **/>**
4. **</property>**
5. **</bean>**
6. **</beans>**

以上Spring的applicationContext.xml文件我是用的SSH**[架构](http://lib.csdn.net/base/16" \o "大型网站架构知识库" \t "http://blog.csdn.net/heng_ji/article/details/_blank)**，如果您用Spring的[MVC](http://www.3y11.com/tags_article.asp?tag_id=mvc" \t "http://blog.csdn.net/heng_ji/article/details/_blank)架构，其原理也是一样的。

对里面的一些概念还不熟悉,一个字"晕"啊,在网上搜搜资料,解释一下applicationcontext.xml

我们以项目中的订单Order为例简要说明一下Spring与**[hibernate](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "http://blog.csdn.net/heng_ji/article/details/_blank)**的集成。关于如何使用Hibernate来对数据库表做映射，我们在前面已经做了介绍，这里我们关心的是如何配置Spring，使它能管理Hibernate。其实，只要在Spring的配置文件（我们这里是applicationContext.xml）中配置一个叫做sessionFactory的bean，Spring就可以和Hibernate联系起来了。

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_284](https://code.csdn.net/snippets/1749109)[IMG_285](https://code.csdn.net/snippets/1749109/fork)

1. **<**beanid**beanid**="sessionFactory" class="org.springframework.orm.hibernate.LocalSessionFactoryBean"**>**
3. **<property** name="configLocation"**>**
5. **<value>**com/ascent/bean/hibernate.cfg.xml**</value>**
7. **</property>**
9. **</bean>**

这样，Spring和Hibernate的第一步整合就完成了，现在到了关键的地方——如何让Spring和Hibernate双剑合璧来实现业务逻辑？

还是在applicationContext.xml文件中先做一个配置。

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_286](https://code.csdn.net/snippets/1749109)[IMG_287](https://code.csdn.net/snippets/1749109/fork)

1. **<**beanid**beanid**="transactionManager"class="org.springframework.orm.hibernate.HibernateTransactionManager"**>**
3. **<property** name="sessionFactory"**>**
5. **<ref** local="sessionFactory"**/>**
7. **</property>**
9. **</bean>**

在上面你大概可以感觉到Spring给我们带来的好处了，Spring的IoC模式可以统一管理各层，而又使各层松散耦合在一起，使各层之间实现最大的解耦性，这也是Web架构一向的追求。

但是，Spring带来的好处还不止于此，除了IoC还有AOP，Spring可以运用AOP来实现很多功能，最常用的就是事务处理。这里我们用了业务服务（business service）层和数据存取对象（Data Access Object）层，在业务服务层我们增加事务处理，数据存取对象层负责数据读写。

首先，组装配置好Service Beans。

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_288](https://code.csdn.net/snippets/1749109)[IMG_289](https://code.csdn.net/snippets/1749109/fork)

1. **<**beanid**beanid**="orderService" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean"**>**
3. **<property** name="transactionManager"**>**
5. **<ref** local="transactionManager"**/>**
7. **</property>**
9. **<property** name="target"**>**
11. **<ref** local="orderTarget"**/>**
13. **</property>**
15. **<property** name="transactionAttributes"**>**
17. **<props>**
19. **<propkey**propkey="find\*"**>**PROPAGATION\_REQUIRED,readOnly,-OrderException**</prop>**
21. **<**propkey**propkey**="save\*"**>**PROPAGATION\_REQUIRED,-OrderException,-OrderMinimumAmountException**</prop>**
23. **</props>**
25. **</property>**
27. **</bean>**

之后，需要把业务服务对象和数据存取对象也组装起来，并把这些对象配到一个事务管理器（transaction manager）里。

在Spring中的配置信息。

**[html]** [view plain](http://blog.csdn.net/heng_ji/article/details/7022171" \o "view plain) [copy](http://blog.csdn.net/heng_ji/article/details/7022171" \o "copy) [IMG_290](https://code.csdn.net/snippets/1749109)[IMG_291](https://code.csdn.net/snippets/1749109/fork)

1. **<**beanid**beanid**="orderTarget"class="com.ascent.business.service.OrderServiceImpl"**>**
3. **<property** name="orderDAO"**>**
5. **<ref** local="orderDAO"**/>**
7. **</property>**
9. **</bean>**


13. **<beanid**beanid="orderDAO"class="com.ascent.dao.hibernate.OrderHibernateDAO"**>**
15. **<property** name="sessionFactory"**>**
17. **<ref** local="sessionFactory"**/>**
19. **</property>**
21. **</bean>**

每个对象都联系着Spring，并且能通过Spring注入到其他对象。把它与Spring的配置文件比较，观察他们之间的关系。

Spring就是这样基于配置文件，将各个Bean搭建在一起的。

这里我们使用一个TransactionProxyFactoryBean，它定义了一个setTransactionManager()，这个对象很有用，它能很方便地处理你申明的Service Object中的事物，你可以通过transaction Attributes属性来定义怎样处理。

TransactionProxyFactoryBean还有个setter，这会被业务服务对象（orderTarget）引用，orderTarget定义了业务服务层，并且它还有个属性，由setOrderDAO()引用这个属性。

还有一点要注意，bean可以用两种方式创造，这些都在单例模式（Sington）和原型模式（propotype）中定义了。默认的方式是singleton，这意味着共享的实例将被束缚，而原型模式是在Spring用到bean的时候允许新建实例的。当每个用户需要得到他们自己Bean的Copy时，你应该仅使用prototype模式。

这样，Spring和Hibernate就完成了对业务对象的管理。