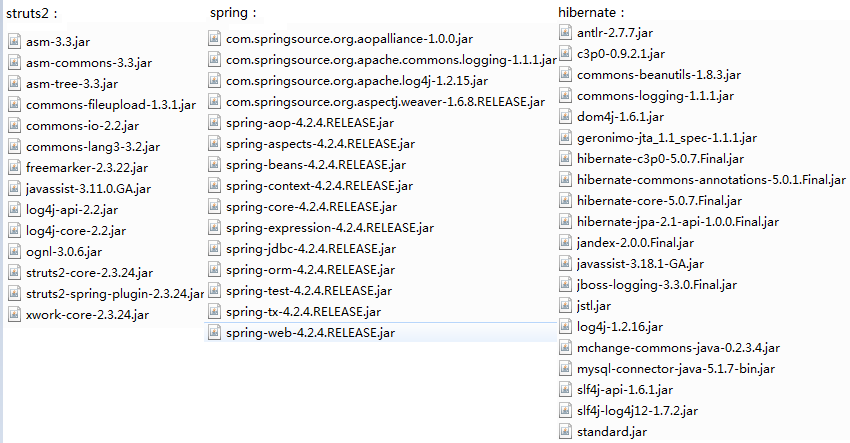
## SSH整合

**一、整合spring和struts2**

1. 导入jar包



1. 复制配置文件到src下



applicationContext.xml内容：

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx.xsd"*>

</beans>

1. 配置xml文件
2. 配置struts2的核心配置文件

<!-- struts的核心过滤器 -->

<filter>

<filter-name>struts</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>struts</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

查找StrutsPrepareAndExecuteFilter类的方法：

快捷键：Ctrl+shift+T --->输入 \*struts\*filter\*

1. 配置spring整合struts2的监听器

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

查找ContextLoaderListener类的方法：

快捷键：Ctrl+shift+T --->输入 \*class\*listener\*

1. 配置spring文件的加载路径

<!-- 配置spring的文件加载路径和文件名 -->

<context-param>

<!-- contextConfigLocation在ContextLoaderListener类的父类ContextLoader类的105行 -->

<param-name>contextConfigLocation</param-name>

<!-- classpath：固定的，代表类路径

冒号后面是 src下spring的配置文件（spring容器）

-->

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

相当于原来在代码里面写的这个：

ApplicationContext c = **new** ClassPathXmlApplicationContext("applicationContext1.xml");

service=(IUserService) c.getBean("se");

1. 配置解决延迟加载的过滤器

<!-- 用来解决延迟加载问题 -->

<filter>

<filter-name>OpenSessionInViewFilter</filter-name>

<filter-class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>OpenSessionInViewFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

1. Action在applicationContext.xml文件中配置（一定要用多例模式）

<!-- 创建dao层的具体实现对象 -->

<bean id=*"customerDao"* class=*"zyf.dao.impl.CustomerDaoImpl"*></bean>

<!-- 创建业务层具体实现对象 -->

<bean id=*"customerService"* class=*"zyf.service.impl.CustomerServiceImpl"*>

<property name=*"dao"* ref=*"customerDao"*></property>

</bean>

<!-- 创建action对象

并改变scope为多例

-->

<bean id=*"customerAction"* class=*"zyf.controller.CustomerAction"* scope=*"prototype"*>

<property name=*"service"* ref=*"customerService"*></property>

</bean>

注意：

所有bean标签中的id属性都是自己随便取的（相当于对象名）

所有bean标签中property标签中的name属性的值都是bean标签中class属性对应类中的一个属性，不可随意更改

在struts.xml中action标签的class属性不能直接像原来一样写包名.类名，而要写sping配置文件中的引用（即bean的id）

<!-- class为spring容器中的引用 -->

<action name=*"customer\_\*"* class=*"customerAction"* method=*"{1}"*>

<result name=*"success"*>/jsp/success.jsp</result>

<result name=*"error"*>/jsp/error.jsp</result>

</action>

1. **整合sping和hibernate**
2. 配置数据源

<!-- 配置数据源 -->

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>

<!--

name的值:ComboPooledDataSource中的属性名称，不可更改

可以在ComboPooledDataSource类的源代码中找到（set方法）

这四个属性是配置数据库连接 -->

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost:3306/ebuy"* />

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"soysouce"* />

</bean>

1. 配置sessionFactory

<!-- sessionFactory -->

<bean id=*"sessionFactory"*

class=*"org.springframework.orm.hibernate5.LocalSessionFactoryBean"*>

<!-- 注入数据源

name="dataSource"：LocalSessionFactoryBean类中的一个属性名,不可更改

ref="dataSource"：自己定义的数据源名称 也就是上面配置数据源的id值

-->

<property name=*"dataSource"* ref=*"dataSource"* />

<!-- hibernate特有的属性

hibernateProperties：Properties类型 -->

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</prop>

<prop key=*"hibernate.show\_sql"*>true</prop>

<prop key=*"hibernate.format\_sql"*>true</prop>

<prop key=*"hibernate.hbm2ddl.auto"*>update</prop>

</props>

</property>

<!-- 配置hibernate映射文件 mappingResources:数组类型 -->

<property name=*"mappingResources"*>

<list>

<value>zyf/entity/Customer.hbm.xml</value>

</list>

</property>

</bean>

1. 配置事务
2. 配置事务

<!-- 事务配置 -->

<bean id=*"transactionManager"*

class=*"org.springframework.orm.hibernate5.HibernateTransactionManager"*>

<property name=*"sessionFactory"* ref=*"sessionFactory"* />

</bean>

1. 开启事务注解

<!-- 开启事务的注解 -->

<tx:annotation-driven transaction-manager=*"transactionManager"* />

1. 在需要开启事务的类中添加注解

@Transactional

**public** **class** CustomerServiceImpl **implements** ICustomerService {

**private** ICustomerDao dao ;

**public** **void** setDao(ICustomerDao dao) {

**this**.dao = dao;

}

@Override

**public** **void** add(Customer cust) **throws** Exception {

dao.add(cust);

System.***out***.println("service ----------add");

}

}

1. 让dao层的实现类继承HibernateDaoSupport类

利用 Session session = **this**.getSessionFactory().getCurrentSession() 获取session对象

前提是：在配置文件中 配置dao层实体类的bean中加上sessionFactory的属性注入

即：

<!-- 创建dao层的具体实现对象 -->

<bean id=*"customerDao"* class=*"zyf.dao.impl.CustomerDaoImpl"*>

<!-- 因为CustomerDaoImpl类中继承了HibernateDaoSupport类 -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*></property>

</bean>

1. 最终文件

web.xml:  
<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*

id=*"WebApp\_ID"* version=*"3.0"*>

<display-name>SSHTemplate</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

<welcome-file>default.html</welcome-file>

<welcome-file>default.htm</welcome-file>

<welcome-file>default.jsp</welcome-file>

</welcome-file-list>

<!-- struts的核心配置文件 -->

<filter>

<filter-name>struts</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>struts</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置监听 监听上下文 加载applicationContext.xml文件 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 配置spring的文件加载路径和文件名-->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

<!-- 用来解决延迟加载问题 -->

<filter>

<filter-name>OpenSessionInViewFilter</filter-name>

<filter-class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>OpenSessionInViewFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

</web-app>

application.xml:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx.xsd"*>

<!--

dao层和service以及action的配置可以用import导入

<import resource=""/>

-->

<!-- 创建dao层的具体实现对象 -->

<bean id=*"customerDao"* class=*"zyf.dao.impl.CustomerDaoImpl"*>

<!-- 因为CustomerDaoImpl类中继承了HibernateDaoSupport类 -->

<property name=*"sessionFactory"* ref=*"sessionFactory"*></property>

</bean>

<!-- 创建业务层具体实现对象 -->

<bean id=*"customerService"* class=*"zyf.service.impl.CustomerServiceImpl"*>

<property name=*"dao"* ref=*"customerDao"*></property>

</bean>

<!-- 创建action对象 并改变scope为多例 -->

<bean id=*"customerAction"* class=*"zyf.controller.CustomerAction"*

scope=*"prototype"*>

<property name=*"service"* ref=*"customerService"*></property>

</bean>

<!-- 配置数据源 -->

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>

<!-- name的值:ComboPooledDataSource中的属性名称，不可更改

这四个属性是配置数据库连接 -->

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost:3306/ebuy"* />

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"soysouce"* />

</bean>

<!-- sessionFactory -->

<bean id=*"sessionFactory"*

class=*"org.springframework.orm.hibernate5.LocalSessionFactoryBean"*>

<!-- 注入数据源

name="dataSource"：LocalSessionFactoryBean类中的一个属性名,不可更改

ref="dataSource"：自己定义的数据源名称

-->

<property name=*"dataSource"* ref=*"dataSource"* />

<!-- hibernate特有的属性 hibernateProperties：Properties类型 -->

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</prop>

<prop key=*"hibernate.show\_sql"*>true</prop>

<prop key=*"hibernate.format\_sql"*>true</prop>

<prop key=*"hibernate.hbm2ddl.auto"*>update</prop>

</props>

</property>

<!-- 配置hibernate映射文件 mappingResources:数组类型 -->

<property name=*"mappingResources"*>

<list>

<value>zyf/entity/Customer.hbm.xml</value>

</list>

</property>

</bean>

<!-- 事务配置 -->

<bean id=*"transactionManager"*

class=*"org.springframework.orm.hibernate5.HibernateTransactionManager"*>

<property name=*"sessionFactory"* ref=*"sessionFactory"* />

</bean>

<!-- 开启事务的注解 -->

<tx:annotation-driven transaction-manager=*"transactionManager"* />

</beans>

注意：

1. 一定要在业务层添加@Transactional的注解添加事务，否则会报错，因为事务必须开启
2. 一定要让dao层的实现类继承HibernateDaoSupport类，然后利用**this**.getSessionFactory().getCurrentSession();获取session对象对数据库进行操作
3. 在dao层继承了HibernateDaoSupport类之后，一致要在配置的相应bean中注入sessionFactory属性

<property name=*"sessionFactory"* ref=*"sessionFactory"*/>

1. action配置文件即struts.xml中action的class一定要用spring容器中声明的bean的id值（不能再用包名.类名）
2. Spring容器中最好只配置数据源、sessionFactory、事务这种核心的配置，其他如service、dao、action的实现类应该配置在相应的包中（命名如applicationContext-struts.xml），然后在核心配置文件中用import标签引入，实现分包。

<import resource="zyf/dao/impl/applicationContext-dao.xml"/>

1. property中的name的值一定是bean中class类中的属性，（di注入），一定要一一对应
2. Action一定要设为多例模式，即bean必须有scope=*"prototype"*属性（spring默认所有bean都是单例的）