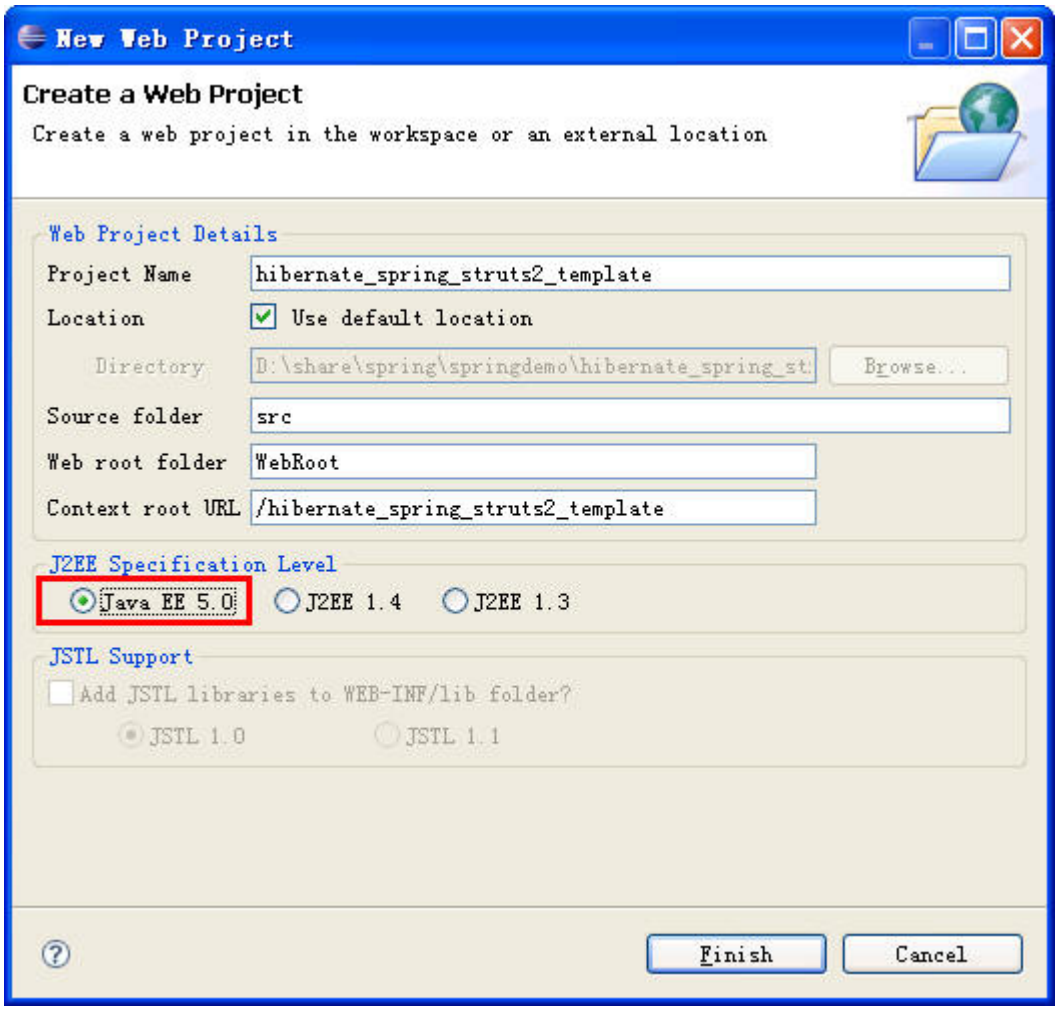
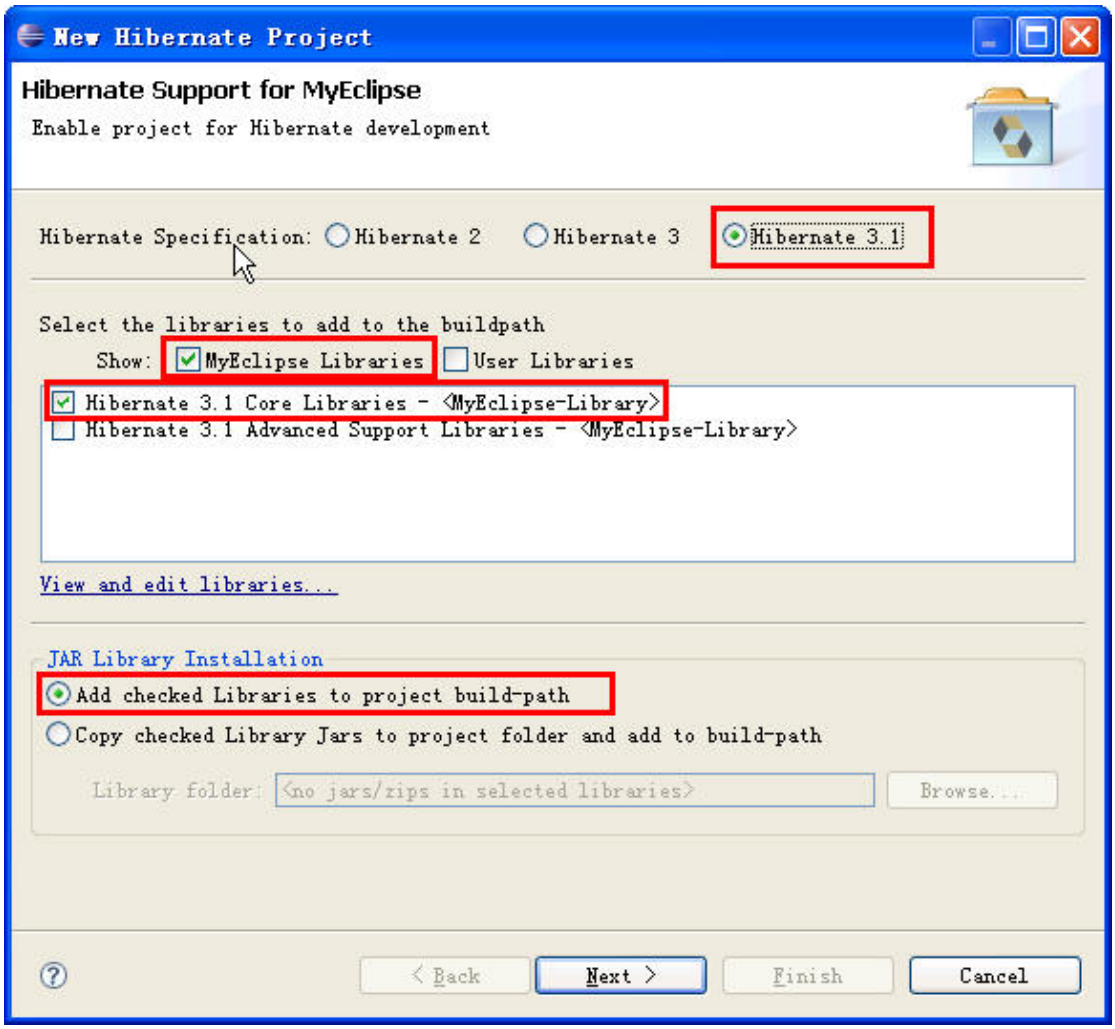
## 1.1 组合 Hibernate 与 Spring

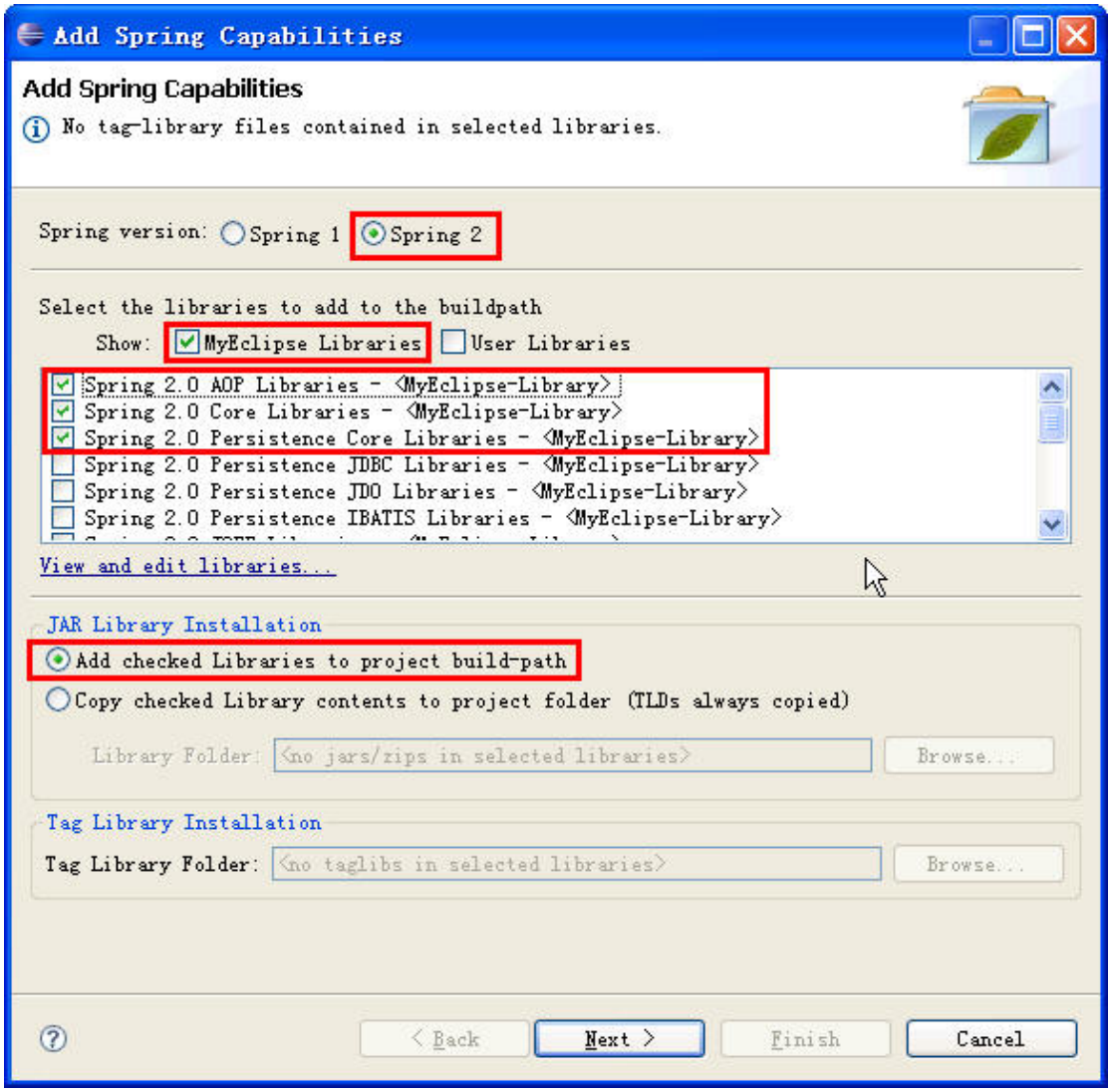
1. 在 Eclipse 中，新建一个 Web project。

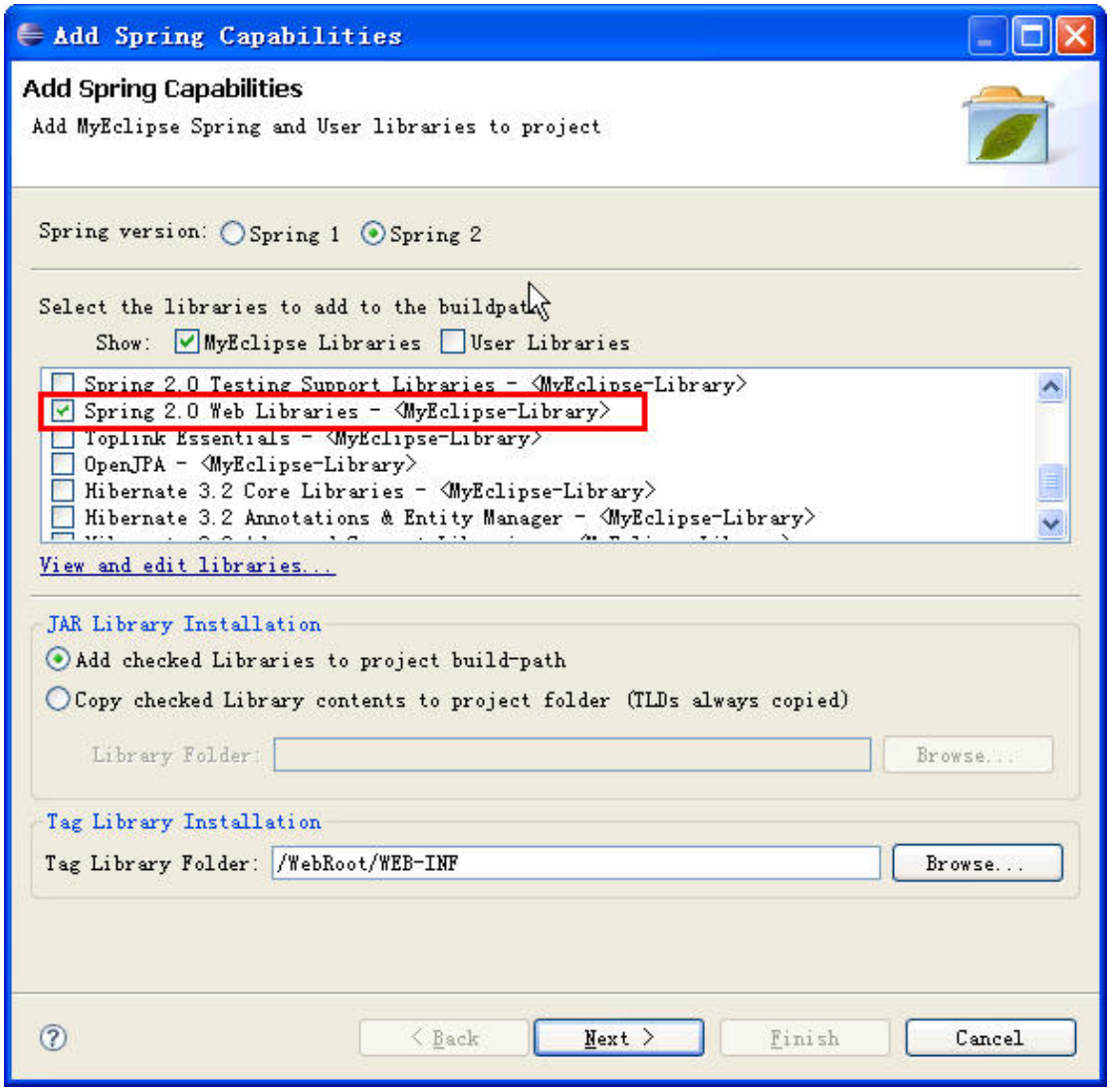



2. 给该项目增加 Hibernate 开发能力，增加 Hibernate 相关类库到当前项目的 Build Path，同时也提供了 hibernate.cfg.xml 这个配置文件。




3. 给该项目增加 Spring 开发能力，增加 spring 相关类库到当前项目的 Build Path，同时也提供了 applicationContext.xml 文件。注意：最好把 applicationContext.xml 文件保存到当前项目的 WebRoot/WEB-INF 的根目录下。


Add Spring Capabilities
Add Spring Capabilities
No tag-library files contained in selected libraries.
Spring version: Spring 1 Spring 2
Select the libraries to add to the buildpath
Show: MyEclipse Libraries User Libraries
Spring 2.0 AOP Libraries - <MyEclipse-Library>
Spring 2.0 Core Libraries - <MyEclipse-Library>
Spring 2.0 Persistence Core Libraries - <MyEclipse-Library>
Spring 2.0 Persistence JDBC Libraries - <MyEclipse-Library>
Spring 2.0 Persistence JDO Libraries - <MyEclipse-Library>
Spring 2.0 Persistence IBATIS Libraries - <MyEclipse-Library>
View and edit libraries...
JAR Library Installation
Add checked Libraries to project build-path
Copy checked Library contents to project folder (TLDs always copied)
Library Folder: <no jars/zips in selected libraries>
Browse...
Tag Library Installation
Tag Library Folder: <no taglibs in selected libraries>
Browse...
< Back
Next >
Finish
Cancel



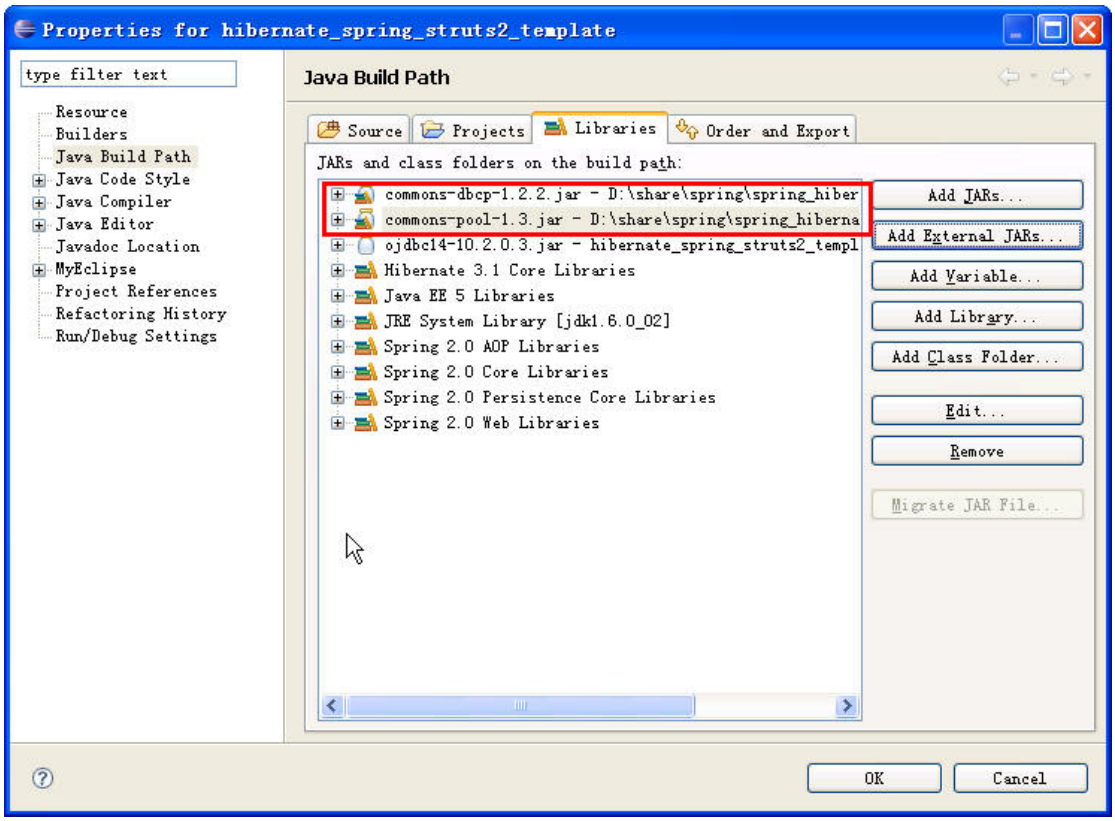

4. 删除 hibernate.cfg.xml 文件，修改 applicationContext.xml 文件的内容，增加 sessionFactory 和 dataSource 的设置。修改的内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

   <bean id="dataSource"
         class="org.apache.commons.dbcp.BasicDataSource">
         <property name="driverClassName"
               value="oracle.jdbc.driver.OracleDriver">
         </property>
         <property name="url"
               value="jdbc:oracle:thin:@liuweiv3000:1521:ora9">
         </property>
```

```
        <property name="username" value="scott"></property>
        <property name="password" value="tiger"></property>
    </bean>

    <bean id="sessionFactory"
        class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource">
            <ref bean="dataSource" />
        </property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">
                    org.hibernate.dialect.Oracle9Dialect
                </prop>
            </props>
        </property>
        <property name="mappingResources">
            <list>
                <value>cn/com/jobedu/shop/model/Catalog.hbm.xml</value>
            </list>
        </property>
</bean>
</beans>
```




5. 通过 MyEclipse 的向导方式，生成 POJO 类和对应的映射文件。

6. 修改 applicationContext.xml 文件中<property name="mappingResources">元素的内容。如下所示：

```
<property name="mappingResources">
        <list>
            <value>cn/com/jobedu/shop/model/Catalog.hbm.xml</value>
          <value>cn/com/jobedu/shop/model/Product.hbm.xml</value>
        </list>
</property>
```

7. 编写 DAO 接口。

```
package cn.com.jobedu.shop.dao;

import java.util.List;

import cn.com.jobedu.shop.model.Catalog;

public interface CatalogDao {

   public void create(Catalog c);

   public Catalog getCatalog(Long id);

   public List getCatalogs();

   public void update(Catalog c);

   public void remove(Long id);

}
```

8. 编写 DAO 接口的实现类，实现类使用 Spring 提供的帮助类。示意如下：

```
package cn.com.jobedu.shop.dao.hibernate;
import java.util.List;

import org.springframework.orm.hibernate3.support.HibernateDaoSupport;

import cn.com.jobedu.shop.dao.CatalogDao;
import cn.com.jobedu.shop.model.Catalog;
```

```
public class CatalogDaoHibernate extends HibernateDaoSupport implements CatalogDao {

    @Override
    public void create(Catalog c) {
    //      HibernateTemplate template=getHibernateTemplate();
    //      template.save(c);
            getHibernateTemplate().save(c);

    }

    @Override
    public Catalog getCatalog(Long id) {
            return (Catalog)getHibernateTemplate().get(Catalog.class, id);
    }

    @Override
    public List getCatalogs() {
            return getHibernateTemplate().find("from Catalog order by id desc");
    }

    @Override
    public void remove(Long id) {
    //      Catalog c=this.getCatalog(id);
    //      getHibernateTemplate().delete(c);
            getHibernateTemplate().delete(getCatalog(id));
    }

    @Override
    public void update(Catalog c) {
            getHibernateTemplate().update(c);
    }
}
```

9. 修改 applicationContext.xml 文件，增加对 Dao 实现类的配置。

```
<bean id="catalogDao"
            class="cn.com.jobedu.shop.dao.hibernate.CatalogDaoHibernate">
            <property name="sessionFactory">
                  <ref bean="sessionFactory" />
            </property>
</bean>
```

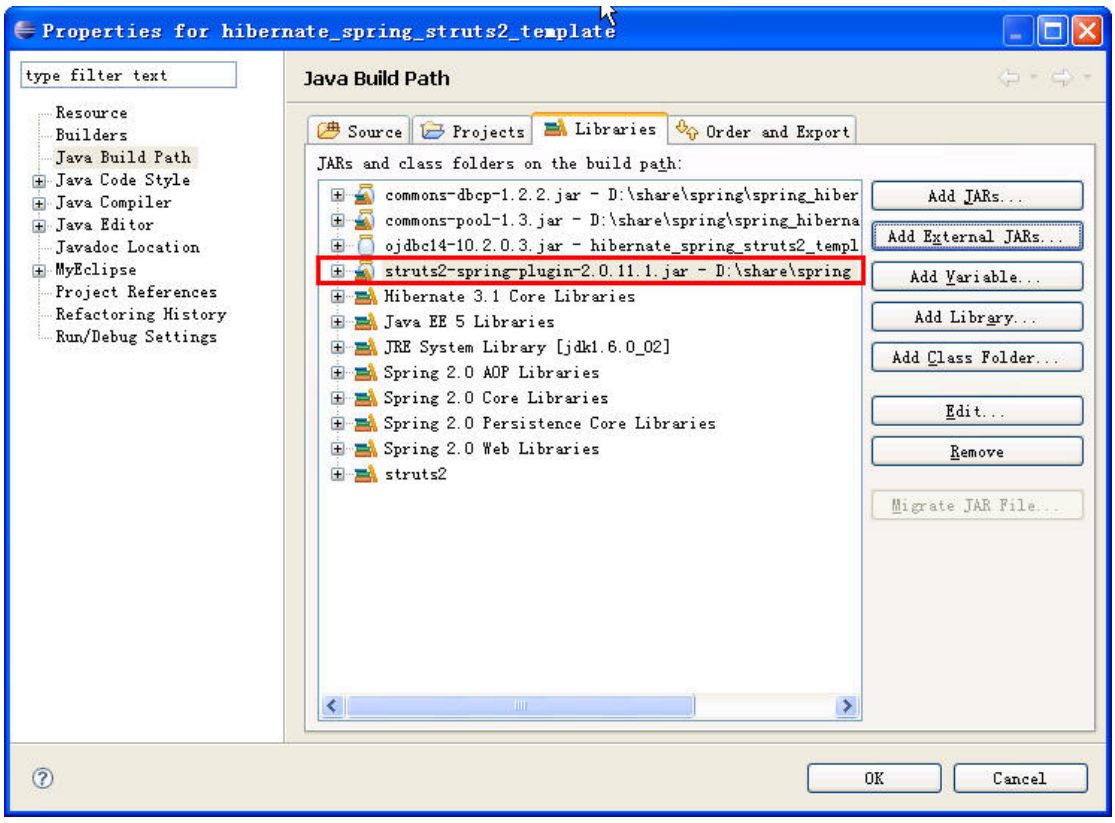
## 1.2　　再组合 Struts2，完成 Struts2、Spring 和 Hibernate 的三者组合

0.　修改 web.xml 文件，增加 struts2 的所需要的过滤器配置。

```
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

1.增加 Struts2 相应类库，增加 struts2-spring-plugin-x-x-x.jar 文件。（Struts 的压缩包中，Struts 所提供的。）




2. 需要拷贝 struts.xml 文件到 src 跟目录下，再修改 struts.xml 文件，进行常量配置。

```
<struts>
        <constant name="struts.objectFactory" value="spring" />
</struts>
```

3. 修改 web.xml 文件，配置 Spring 监听器和上下文变量。

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext*.xml</param-value>
</context-param>

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

修改 web.xml 文件，增加 OpenSessionInViewFilter 的设置。

```
<filter>
   <filter-name>lazyLoadingFilter</filter-name>
<filter-class>org.springframework.orm.hibernate3.support.OpenSessionInViewFilter
</filter-class>
```

```
</filter>
<filter-mapping>
        <filter-name>lazyLoadingFilter</filter-name>
        <url-pattern>*.action</url-pattern>
</filter-mapping>
```

4. 编写 Action 类。

5. 配置 struts.xml 文件。

```
struts.xml
<!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
        "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <constant name="struts.objectFactory" value="spring" />
        <include file="struts-default.xml"/>

            <package name="default" extends="struts-default">
            <action name="hello" class="helloBean">
                <result>hello.jsp</result>
            </action>
        ....
        </package>
</struts>
```

6. 修改 applicationConext.xml，默认情况下，Spring 从下面的文件中寻找为 action 所做的配置。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans default-autowire="byName">
        <bean id="helloBean" class="cn.com.jobedu.HelloWorld"    scope="prototype"/>
        ...
</beans>
```

7.编写所需要的 JSP 文件。

8. 部署，调试整个项目。

```
   <!-- 配置事务管理器 -->
```

```
<bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
        <property name="sessionFactory">
            <ref local="sessionFactory"/>
        </property>
   </bean>

    <!-- 配置事务特性 ，配置 add、delete 和 update 开始的方法，事务传播特性为
required-->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <tx:method name="add*" propagation="REQUIRED"/>
            <tx:method name="delete*" propagation="REQUIRED"/>
            <tx:method name="update*" propagation="REQUIRED"/>
            <tx:method name="*" read-only="true"/>
        </tx:attributes>
    </tx:advice>

    <!-- 配置那些类的方法进行事务管理，当前 cn.com.jobedu.crm.service 包中的子包、
类中所有方法需要，还需要参考 tx:advice 的设置 -->
    <aop:config>
        <aop:pointcut id="allManagerMethod" expression="execution (*
cn.com.jobedu.crm.service.*.*(..))"/>
        <aop:advisor advice-ref="txAdvice" pointcut-ref="allManagerMethod"/>
    </aop:config>
```