

## SSH框架简介：

①SSH框架是由struts2、spring、hibernate三大框架组合起来的一套总框架，一般来说这三个东西我们不会单独使用。

②在学习SSH框架之前建议读者先学mvc，因为SSH是在mvc基础上根据mvc的缺点而产生的一套比较成熟的框架，也比较稳定。

③SSH框架的流程：浏览器（或客户端）发送请求到服务器，先经过项目中web.xml中过滤器（<filter>和<filter-mapping>）审核，通过了再发送给action包中的IndexAction类，struts.xml根据IndexAction类中return的值再进行跳转，跳转的页面是struts.xml中<result>配置的页面名，然后页面响应回客户端（至于怎么响应的就是当客户敲回车之后就有一个页面显示）。

④struts的核心思想：实现mvc

⑤spring的核心思想：解耦，也就是代码中不出现new**实现类**的代码，我们创建了接口不用关心实现类是谁，实现类由spring帮我们注入，我们只需要在定义接口的时候给它一个set方法并且在配置文件里改<property>中的id和ref就行

⑥hibernate的核心思想：(ORM-对象关系映射)连接数据库，我们不用在数据库写创建表的语句，数据库表的字段根据实体类中属性的名字然后我们在BookCard.hbm.xml文件里配置<property>以及<property>的相关属性。

搭建前需要注意事项（搭建前应准备）：

1. 需要用到的技术（必须了解）：[Struts2](#)/[spring](#)（新版spring官网不能直接下载，可以[百度下载别人打包好的](#)）/[hibernate](#)（S-S-H）
2. 需要用到的工具：eclipse
3. 需要用到的包：①struts2.3.30   ②spring-framework-4.2.2.RELEASE-dist   ③hibernate-release-5.2.2.Final

4. 建议运行环境：①Windows 7-64位    ②Tomcat 8.0    ③jdk1.8.0\_91  
④SQL server2008    ⑤Eclipse JavaEE环境下

Struts、spring、hibernate中所有导的包都放在WEB-INF-->lib目录下

软件工程思想（灵魂）：高内聚、低耦合

必须熟知每个包的作用：

action包（跳转），

service包（服务/业务逻辑），

dao包（访问数据库），

entity包（实体类），

util包（工具包）。在

创建包的同时我们要新建的接口以及实现类有：

①在service包创建接口IndexService（名字随意但最好有意义）以及实现类IndexServiceImpl（名字随意但最好有意义）

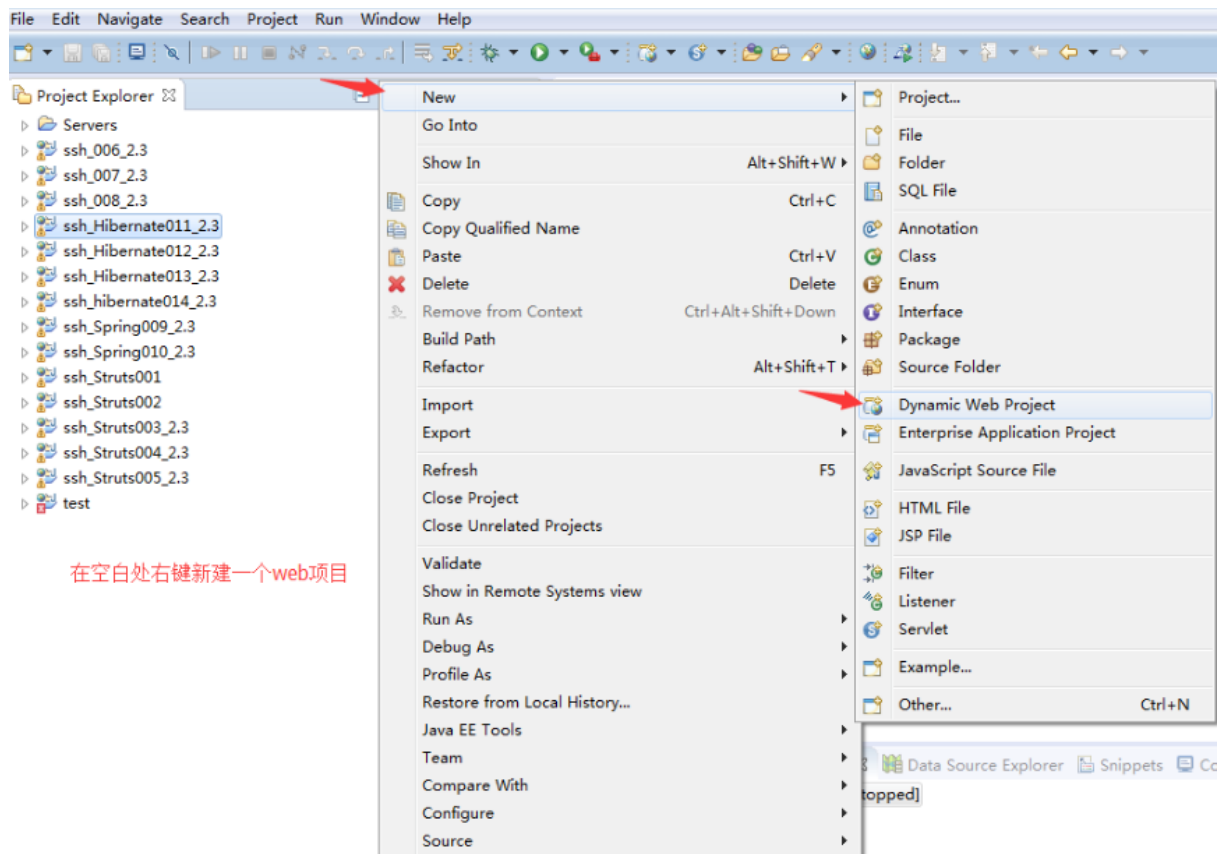
②在dao包新建接口IndexDao以及实现类IndexDaoImpl

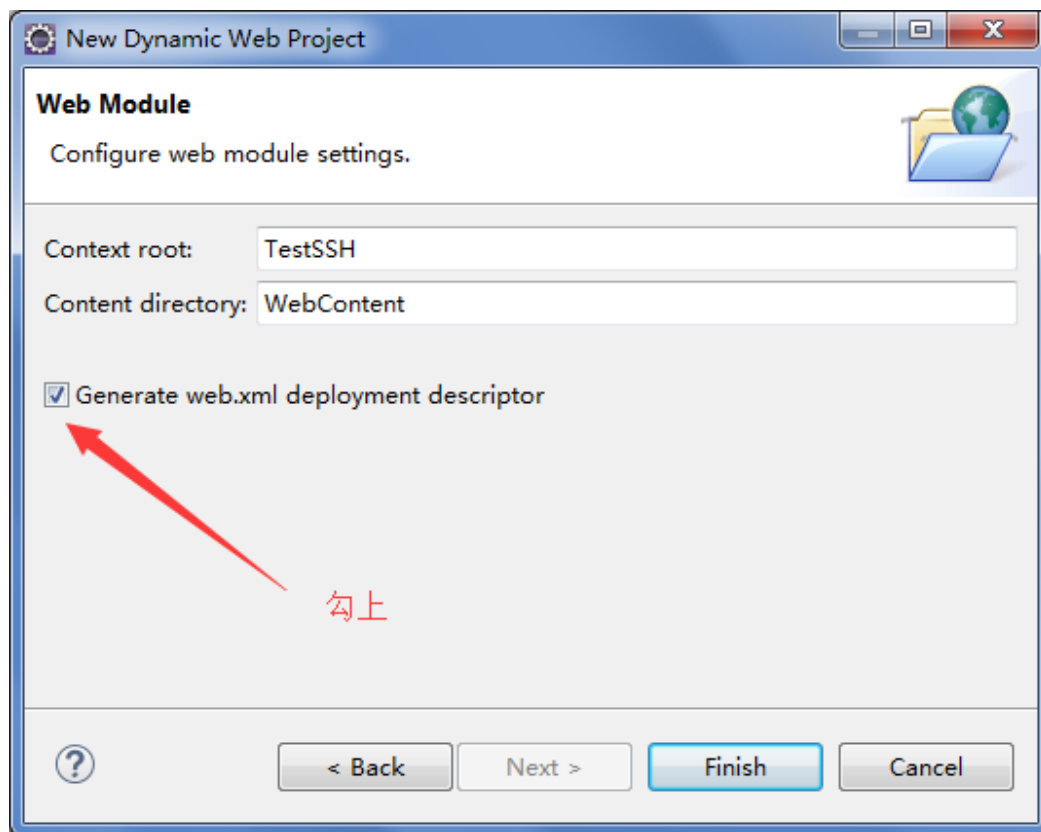
③在util包新建接口MyConnection以及实现类MyConnectionImpl

util包有两个作用：工具包、连接数据库包，但加入了hibernate之后取代了连接数据库的功能，现在的util包只做工具包。

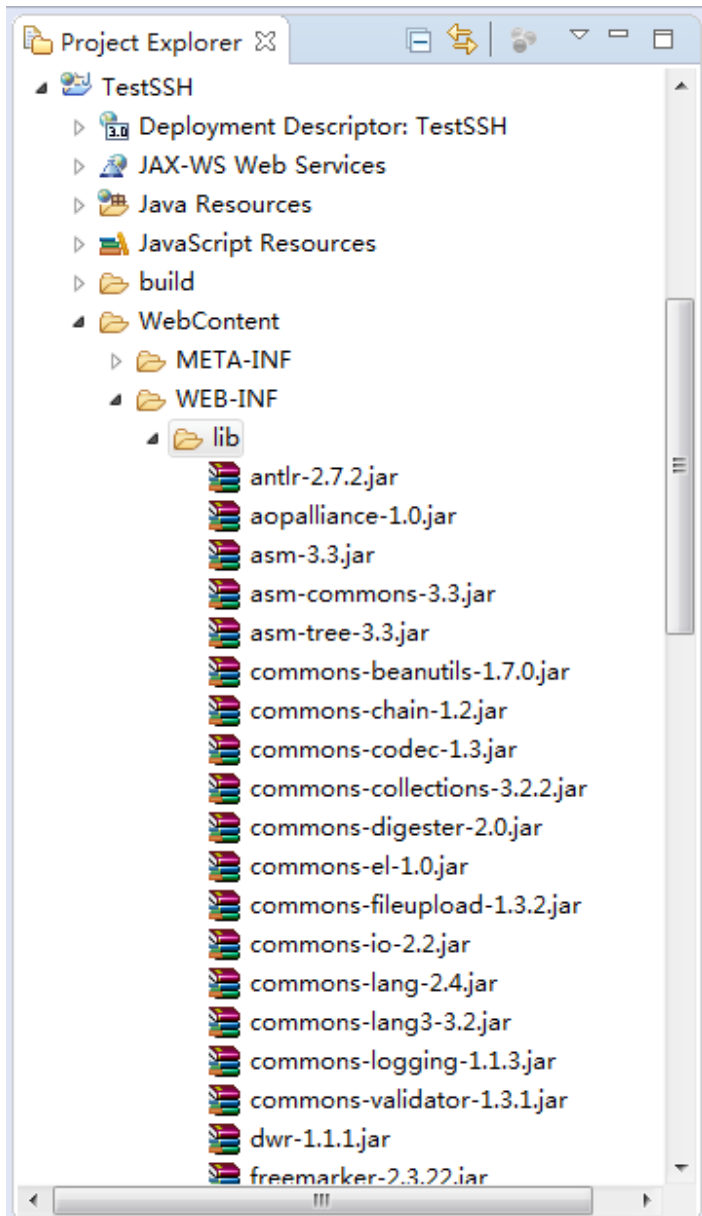
正式开始：

第一步：新建一个web项目，并勾上Generate web.xml deployment descriptor。如下图：





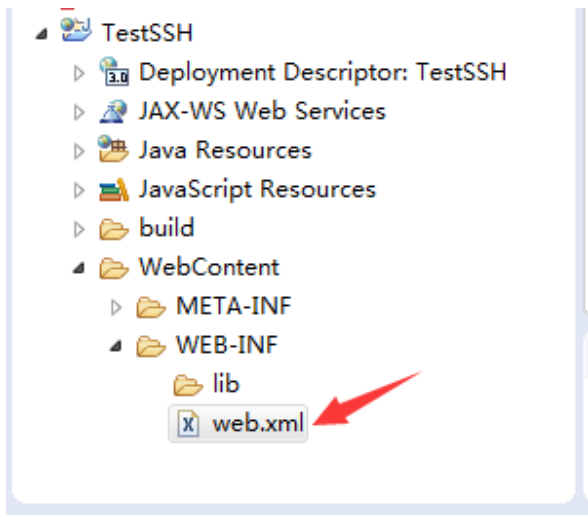
第二步：导入struts的所有jar包放在lib目录下  
(struts2.3.30\struts-2.3.30-apps\struts-2.3.30\apps\struts2-showcase\WEB-INF\lib)



第三步： 打开web.xml文件。

目录： 【WebContent-->WEB-INF目录-->web.xml】 如下

图：



打开web.xml之后出现如下代码：



复制代码

```
<?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>TestSSH</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>

</web-app>
```



复制代码

改成如下（直接把以下代码拷贝覆盖原来的代码）



复制代码

```
<?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>TestSSH</display-name>
</web-app>
//配置首页
<welcome-file-list>
    <welcome-file>index.action</welcome-file>
</welcome-file-list>
//配置过滤器
<filter>
    <filter-name>struts2</filter-name>
    <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-
class>
</filter>
//配置映射
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

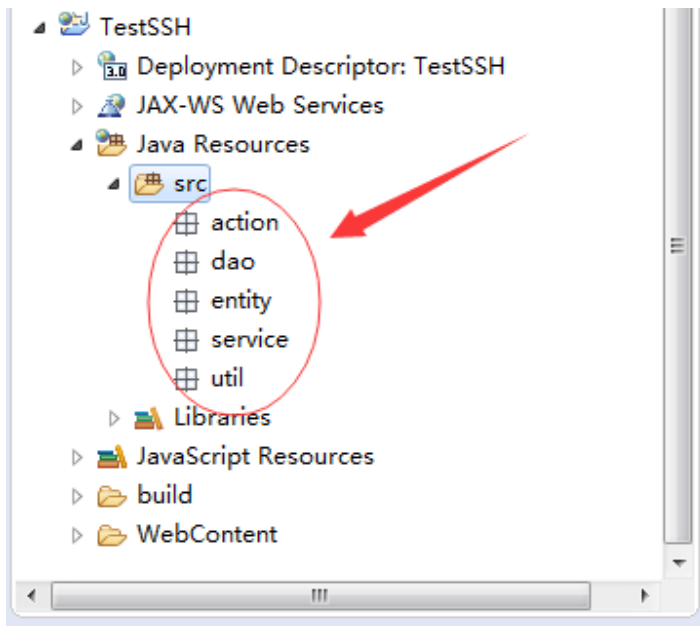
</web-app>

```



复制代码

**第四步：**在src目录下新建五个包：action（跳转）包、service（服务/业务逻辑）包、dao（访问数据库）包、entity（实体类）包、util（工具）包。如下图：



**第五步：**在action包下新建IndexAction类并继承ActionSupport，目的是为了本类拥有ActionSupport类的方法



复制代码

```
package action;

import com.opensymphony.xwork2.ActionSupport;

public class IndexAction extends ActionSupport{
    //定义Struts默认的方法
    public String execute(){
        //返回字符串类型给struts.xml文件接收
        return "success";
    }
    //定义错误时应调用方法
    public String error(){
        //返回字符串类型给struts.xml文件接收
        return "error";
    }
}
```



复制代码

**第六步：**在src目录下创建struts.xml文件





复制代码

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<!-- 上面的头，注意版本，从样例里复制过来 showcase.war\WEB-INF\src\java\struts.xml -->

<struts>
    <!-- 第1步：先定义一个包，名字任意取 并继承struts-default-->
    <package name="mypck" extends="struts-default">
        <!-- 第2步：①定义一个action，配置跳转信息 name 类似于Servlet，这个name是浏览器要
访问的name,很重要，②class的值：包名.类名
        http://xxxx/xxx/Index.action http://xxxx/xxx/Index class 对应于自己写的
Action类 当不写method属性时，默认调用的是execute -->
        <action name="Index" class="ssh.IndexAction" method="exe2">
            <!-- 跳转是forward /WEB-INF/是防止jsp不经过action就可以访问 -->
            <!-- success和error是从action类中return回的值，要与action类return值一样 -->
            <result name="success"/WEB-INF/index2.jsp</result>
            <result name="error"/WEB-INF/error.jsp</result>
        </action>
    </package>
</struts>
```























复制代码

经过以上步骤我们SSH中的struts2已经配置好了，下面是配置Spring：

**第七步：**导jar包（\spring-framework-4.2.2.RELEASE\libs全部拷贝 以Javadoc和source为后缀的不要

如下图：

名称	修改日期	类型
 spring-aop-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-aspects-4.2.2.RELEASE	2015/10/15 5:15	WinRAR 压缩文
 spring-beans-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-context-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-context-support-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-core-4.2.2.RELEASE	2015/10/15 5:11	WinRAR 压缩文
 spring-expression-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-instrument-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-instrument-tomcat-4.2.2.RELEASE	2015/10/15 5:15	WinRAR 压缩文
 spring-jdbc-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-jms-4.2.2.RELEASE	2015/10/15 5:15	WinRAR 压缩文
 spring-messaging-4.2.2.RELEASE	2015/10/15 5:14	WinRAR 压缩文
 spring-orm-4.2.2.RELEASE	2015/10/15 5:13	WinRAR 压缩文
 spring-oxm-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-test-4.2.2.RELEASE	2015/10/15 5:15	WinRAR 压缩文
 spring-tx-4.2.2.RELEASE	2015/10/15 5:12	WinRAR 压缩文
 spring-web-4.2.2.RELEASE	2015/10/15 5:13	WinRAR 压缩文
 spring-webmvc-4.2.2.RELEASE	2015/10/15 5:14	WinRAR 压缩文
 spring-webmvc-portlet-4.2.2.RELEASE	2015/10/15 5:14	WinRAR 压缩文
 spring-websocket-4.2.2.RELEASE	2015/10/15 5:15	WinRAR 压缩文

## 第八步：在web.xml文件里配置监听器



复制代码

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">
    <display-name>ssh_001</display-name>
    <welcome-file-list>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
    <!-- Struts过滤器 -->
    <filter>
```

```

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

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

        <!-- spring的监听器配置开始 -->
        <!-- spring监听器的作用：提供实例 -->
        <context-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </context-param>
        <listener>
            <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>
        <!-- spring的监听器配置结束 -->
    </web-app>

```



复制代码

## 第九步：在struts.xml文件里配置<constant name="struts.objectFactory" value="spring" />



复制代码

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<!-- 上面的头，注意版本，从样例里复制过来 showcase.war\WEB-INF\src\java\struts.xml -->

<struts>
    <!-- 这里是spring配置 -->
    <!-- 告知Struts2运行时使用Spring来创建对象 -->
    <constant name="struts.objectFactory" value="spring" />

```

```

<!-- 第1步：先定义一个包，名字任意取 -->
<package name="mypck" extends="struts-default">
    <!-- 第2步：定义一个action，配置跳转信息 name 类似于Servlet，这个name是浏览器要访问的name，很重要，class的值：包名.类名
    http://xxxx/xxx/Index.action http://xxxx/xxx/Index class 对应于自己写的Action类 当不写method属性时，默认调用的是execute -->
    <action name="Index" class="ssh.IndexAction" method="exe2">
        <!-- 跳转是forward /WEB-INF/是防止jsp不经过action就可以访问 -->
        <!-- success和error是从action类中return回的值，要与action类return值一样 -->
    ->
        <!-- index2.jsp和error.jsp表示要跳转的页面 -->
        <result name="success">/WEB-INF/index2.jsp</result>
        <result name="error">/WEB-INF/error.jsp</result>
    </action>
</package>
</struts>

```



复制代码

## 第十步：在src目录下新建application.xml文件



复制代码

```

<?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:p="http://www.springframework.org/schema/p"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:jee="http://www.springframework.org/schema/jee"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-4.2.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.2.xsd
        http://www.springframework.org/schema/jee
        http://www.springframework.org/schema/jee/spring-jee-4.2.xsd

```

<http://www.springframework.org/schema/tx>

[http://www.springframework.org/schema/tx/spring-tx-4.2.xsd">](http://www.springframework.org/schema/tx/spring-tx-4.2.xsd)

<!-- ①一个bean标签对应一个类，id为myIndexAction的bean就对应项目中的IndexAction类  
②id的值是随便起，但最好有意义 ③class的值是包名.类名 ④scope="prototype"是非单例，不用理解，但一定要写这句代码，记住有这回事就行 -->

```
<bean id="myIndexAction" class="ssh.action.IndexAction" scope="prototype">
```

<!-- ①name的值是要注入的变量名 ②ref是引用类的类名，name为"is"的变量引用的是myIndexService的值 -->

```
<property name="is" ref="myIndexService"/>
```

```
</bean>
```

<!-- myIndexService = new ssh.service.IndexServiceImpl() id为myIndexService的bean对应项目中的IndexService类-->

```
<bean id="myIndexService" class="ssh.service.IndexServiceImpl"
scope="prototype">
```

<!-- name为id的变量引用的是myIndexDao的值 -->

```
<property name="id" ref="myIndexDao"/>
```

```
</bean>
```

```
<bean id="myIndexDao" class="ssh.dao.IndexDaoImpl" scope="prototype">
```

```
<property name="c" ref="myConnection"></property>
```

```
</bean>
```

<!-- 下面这个bean是对应项目中的connection类，class的值是包名.类名 -->

```
<bean id="myConnection" class="ssh.util.MyConnectionImpl_SQLSERVER"
scope="prototype">
```

<!-- 这里没有<property>是因为connection这个类已经是连接数据库的类，我们已经不需要通过new实现类了 -->

```
</bean>
```

```
</beans>
```



复制代码

以上就是struts和spring的使用，接下来是SSH中的最后一个，也是最重要的一个：hibernate

## 第十一步：导jar包 ( hibernate-release-5.2.2.Final\lib\required所有包 )

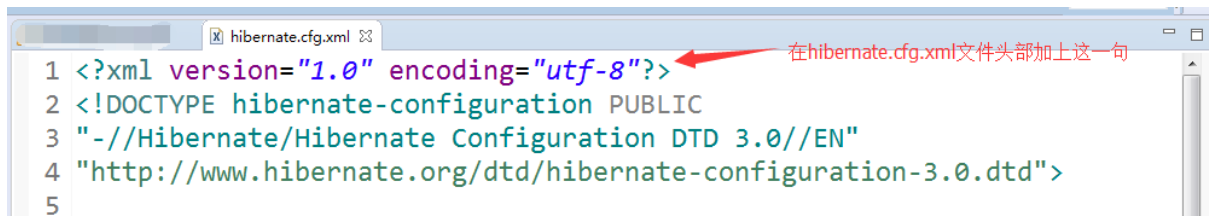
名称	修改日期	类型
antlr-2.7.7	2014/6/19 15:23	WinRAR 压
cdi-api-1.1	2014/6/23 11:56	WinRAR 压
classmate-1.3.0	2015/10/7 12:45	WinRAR 压
dom4j-1.6.1	2014/6/19 15:21	WinRAR 压
el-api-2.2	2014/6/19 15:23	WinRAR 压
geronimo-jta_1.1_spec-1.1.1	2015/5/11 13:42	WinRAR 压
hibernate-commons-annotations-5.0....	2015/11/26 13:12	WinRAR 压
hibernate-core-5.2.2.Final	2016/8/4 14:34	WinRAR 压
hibernate-jpa-2.1-api-1.0.0.Final	2014/6/19 15:23	WinRAR 压
jandex-2.0.0.Final	2015/11/26 13:12	WinRAR 压
javassist-3.20.0-GA	2015/10/9 18:53	WinRAR 压
javax.inject-1	2014/6/19 15:23	WinRAR 压
jboss-interceptors-api_1.1_spec-1.0.0...	2014/6/19 15:23	WinRAR 压
jboss-logging-3.3.0.Final	2015/5/30 11:27	WinRAR 压
jsr250-api-1.0	2014/6/19 15:23	WinRAR 压

第十二步：把 ( hibernate-release-5.2.2.Final\project\hibernate-core\src\test\resources ) 目录下的hibernate.cfg.xml文件放在src目录下。

hibernate-release-5.2.2.Final > project > hibernate-core > src > test > resources >				
名称	修改日期	类型	大小	
META-INF	2015/6/10 22:29	文件夹		
org	2016/5/4 19:02	文件夹		
DB2JccConfiguration.properties	2015/6/10 22:29	PROPERTIES 文件	1 KB	
hibernate.cfg.xml	2016/5/6 14:03	XML 文档	1 KB	
hibernate.properties	2016/6/21 17:02	PROPERTIES 文件	1 KB	
log4j.properties	2016/7/26 15:48	PROPERTIES 文件	3 KB	
spy.properties	2015/6/10 22:29	PROPERTIES 文件	1 KB	

**第十三步：**在hibernate.cfg.xml文件里最顶部加上<?xml version="1.0" encoding="utf-8"?>

如图：



**第十四步：**配置hibernate.cfg.xml文件，并配置映射文件



复制代码

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- hibernate配置文件 -->
        <!-- 配置数据库名，以及用户名，密码 -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property
name="connection.url">jdbc:mysql://localhost:3306/CardDB</property>
        <property name="connection.username">root</property>
        <property name="connection.password">123456</property>
        <!-- 每个数据库都有1个 -->
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="connection.pool_size">5</property>
        <property name="show_sql">true</property>
        <property name="format_sql">true</property>
        <property name="hbm2ddl.auto">update</property>
        <!-- 配置映射文件 -->
        <mapping resource="ssh/entity/BookCard.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```



复制代码

## 第十五步：配置BookCard.hbm.xml（实体类配置）文件



复制代码

```
<?xml version="1.0" encoding="UTF-8"?>
<hibernate-mapping xmlns="http://www.hibernate.org/xsd/hibernate-mapping">
    <!-- ①class的值是包名.实体类名    ②table的值是数据库表名 -->
    <class name="ssh.entity.BookCard" table="BookCard">
        <!-- ①<id>标签是要作为主键的属性或字段才能用    ②column是数据库的字段名-->
        <id name="cid" column="cid">
            <generator class="native"></generator>
        </id>
        <!-- <property>标签对应于属性（数据库字段）在<property>标签中设置数据库相关的属性，
        比如长度、类型、是否为空、列名...等等 -->
        <property name="name" type="string" length="50" column="name" not-
null="true"></property>
        <property name="sex" type="string" length="2" column="sex"></property>
        <property name="cardDate" type="date" column="cardDate"></property>
        <property name="deposit" type="double" column="deposit"></property>
    </class>
</hibernate-mapping>
```



复制代码

## 第十六步：在IndexDaoImpl实现类中构造SessionFactory



复制代码

```
package ssh.dao;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.hibernate.query.Query;

public class IndexDaoImpl implements IndexDao {
    <!-- SessionFactory是hibernate的内置对象 -->
    private SessionFactory sessionFactory;

    <!-- 给SessionFactory一个set方法，便于spring注入 -->
    public void setSessionFactory(SessionFactory sf) {
        this.sessionFactory = sf;
    }
}
```



```
}
```

```
@Override
```

```
public List<BookCard> getAllBookCard() {
```

```
    <!-- sessionFactory这个实例可以自己按常规的hibernate传统写法创建也可以交给spring  
去托管sessionFactory = new Configuration().configure().buildSessionFactory(); -->
```

```
    Session session = sessionFactory.openSession();
```

```
}
```

```
}
```



复制代码

以上就是eclipse搭建SSH框架的全过程，接下来就可以在service相关类以及dao相关类编写我们的代码！