lifetragedy的专栏

吾以吾血荐我中华之IT

通向架构师的道路（第二十二天）万能框架spring(四)使用struts2

分类： 架构师之路 2012-11-16 23:51 10114人阅读 评论(19) 收藏 举报

annotation maven Maven ssh2 教程 框架

目录(?) [+]

一、前言

SSH有了，现在我们要把我们的struts层从原来的1.3替换成struts2.x，引入了struts2.0后我们会发觉我们的代码和框架的变化还是不小的

二、Struts2的好处

1）在struts2的方法里，一切变量是线程安全的，而原有的struts1不是的;

2）在struts2中如果你声明了如下这样的代码：

```
privater String studentName="";
public void setStudentName(String studentName){
   this.studentName = studentName;
}
public String getStudentName(){
   return this.studentName;
}
```

那么当你对这个studentName进行符值后，不需要再把它用request.setAttribute这样的形式把值带到页面中去了，相当于你可以省去在request中来回的setAttribute{...}getAttribute{...}的操作（有时由于忘记把一个listset 到request中去,经常导致一个页面就是不显示列表，对吧？这样的事可以被极大程度上避免掉）。

3）更丰富且描述简单的页面标签，可以直接支持将一个Object和页面的<input>进行绑定，如：

我在后台如果有一个StudentVO，这个StudentVO如下描述：

```
        private String studentNo = "";
        private String studentName = "";

        public String getStudentNo() {
                return studentNo;
        }

        public void setStudentNo(String studentNo) {
                this.studentNo = studentNo;
        }

        public String getStudentName() {
                return studentName;
        }

        public void setStudentName(String studentName) {
                this.studentName = studentName;
```

```
        }
```

于是我在前台jsp里可以直接这样使用我的<input>标签和我这个VO中的某个字段进行绑定：

```
<s:textfield name="studentVO.studentName" size="24" maxlength="25"/>
```

4）原有在struts1中的formbean彻底消失，去而代之的是使用VO对象，一个strutsaction就是一个普通的类，只是它extendsActionSupport而已。

5）良好的注入机制，连session,request, response都可以注入了，因此你的一个action方法就是一个普通类方法，这样做的好处是极大化将servlet与我们的action进行解耦合。试想如果是原有的struts1的action方法，我现在要改成swing的actionPerform，你是不是要把原有的action方法包括传参都要进行重构啊？而现在有了struts2，由于连session,request, response都是被注入的，因此这个struts2的action方法可以直接重用。

Strtus2还有很多好处，这边不一一列举了，在struts2的官方文档和stepby step等书中详细有说，我们这边主要以实战为主，讲述struts2怎么和spring进行整合并且能够开发我们的应用。

三、整合spring和struts2

我们还是用我们的Maven2。

Struts2变化很大，它是一个几乎被重写的框架，而不是一个“增强”的框架，它是继承自xwrok的框架并且在整个框架中全面使用了filter机制。

对于我们的maven的pom.xml文件来说，这个lib库的改动还是很大的。

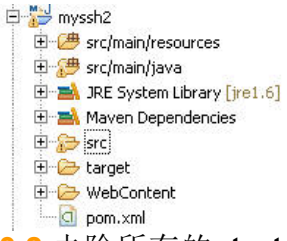
甚至还会出现一些莫名奇妙的错误而其原因是因为lib库的版本不对或者是有冲突，为此笔者整理了一份ssh2的所有需要的jar的mavenpom.xml文件。

虽然，我会在后面把这个xml文件完整列出来但还是希望大家在一开始跟着我能够一步步走，对pom.xml文件和工程进行排错，这样你将对一些常用的框架的lib库有个比较熟悉的过程。

3.1 延用原有的myssh工程中的pom.xml文件

我们新建一个maven的web工程-myssh2，并将原有的myssh工程中的pom.xml文件拷入工程中。

请确保你使用的**jdk**版本为**version1.6.x**。



3.2 去除所有的struts1.3的依赖关系

打开这个pom.xml文件，把下面这段所有的关于struts1.3的依赖包全部去除。

```
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-core</artifactId>
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-el</artifactId>
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-extras</artifactId>
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-faces</artifactId>
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-mailreader-dao</artifactId>
```

```
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-scripting</artifactId>
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-taglib</artifactId>
        <version>1.3.10</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts-tiles</artifactId>
        <version>1.3.10</version>
</dependency>
```

3.3 增加**struts2**的依赖包

我们把原有的struts1.3的依赖包去除后加入struts2的依赖包

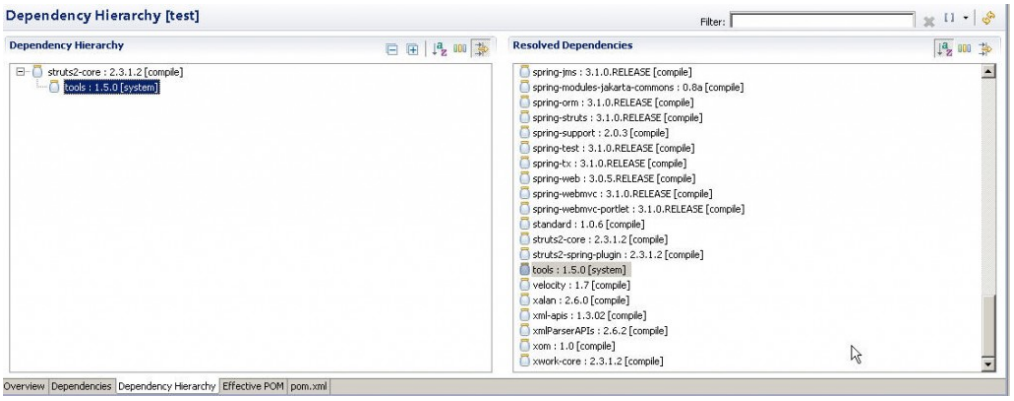
```
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-spring-plugin</artifactId>
        <version>2.3.1.2</version>
</dependency>
<dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
        <version>2.3.1.2</version>
</dependency>
```

存盘后，此时maveneclipse插件会自动开始编译和下载相关的jar到你的本地maven的repository中，然后我们会发觉这个**pom.xml**文件出错了：

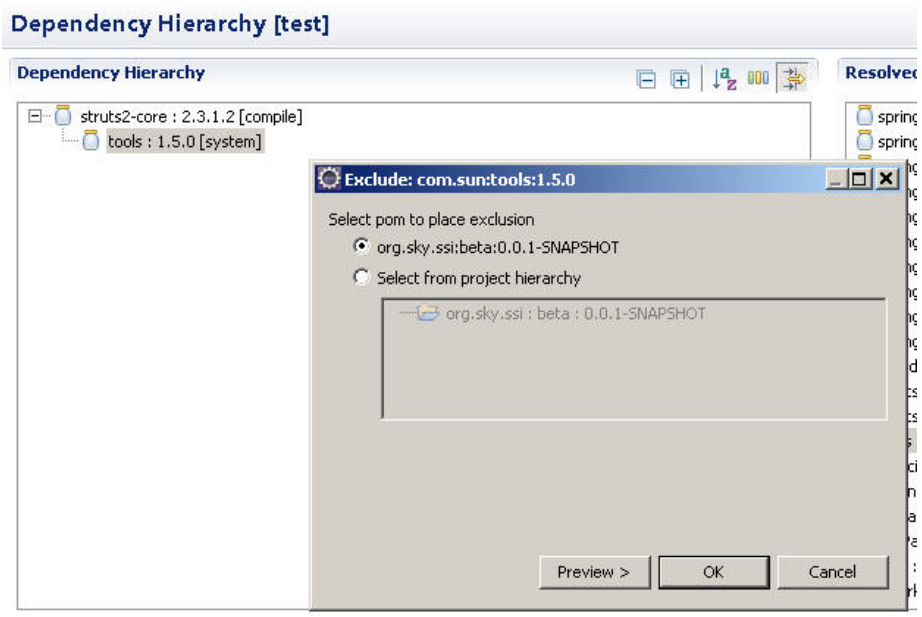
抛一个**sun.tool.jar**没有找到的错误。

道理很简单，因为该tool.jar其实已经存在在我们本地安装的jdk的lib目录下了，因此我们不需要这个包，但是maven是自动依赖的，你没有看到它在pom.xml文件中出现不代表这层依赖关系不存在。

因此我们需要做的是exclude这个包。



让我们在maveneclipse插件中打开这个pom.xml文件，切换到“DependencyHierarchy”视图，然后找到这个tool.jar文件，点左边这个list中的tools:1.5.0然后右键选“ExcludeMaven Artifact”。



选[ok]按钮然后存盘。

我们可以看到这个pom.xml文件一切正常了，没有红色的“叉叉”了，我们切换到pom.xml视图，可以看到它其实做了这么一件事（注意红色标粗的语句）：

```
<dependency>
		<groupId>org.apache.struts</groupId>
		<artifactId>struts2-spring-plugin</artifactId>
		<version>2.3.1.2</version>
</dependency>
<dependency>
		<groupId>org.apache.struts</groupId>
		<artifactId>struts2-core</artifactId>
		<version>2.3.1.2</version>
		<exclusions>
			<exclusion>
				<artifactId>tools</artifactId>
				<groupId>com.sun</groupId>
			</exclusion>
		</exclusions>
</dependency>
```

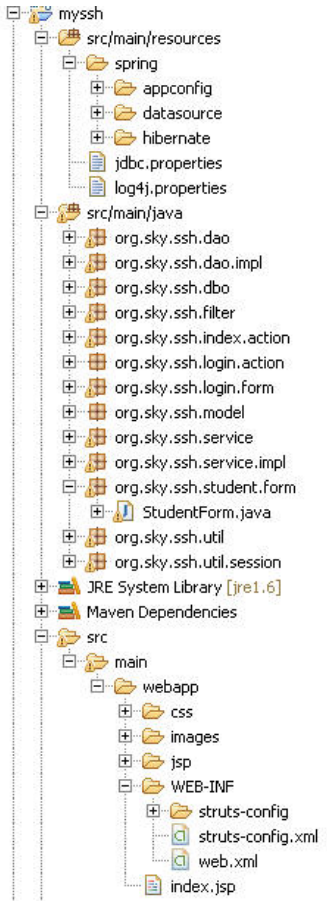
然后：

1）我们把原先ssh工程中的resources目录下所有的东西拷到myssh2工程的resources目录下；

2）我们把原先的ssh工程的java文件拷过来；

3）我们把原先的ssh工程的src/main/webapp目录下的文件也拷贝过来；

4）不要忘了把WEB-INF/web.xml文件和index.jsp文件也拷过来啊！



1）我们把原有的org.sky.ssh.student.form和org.sky.ssh.login.form删了；

2）我们把原有的service类中的一些需要传入StudentForm的方法的中的StudentForm改成org.sky.ssh.vo.StudentVO，其内容如下：

```
package org.sky.ssh.vo;

import java.io.Serializable;

public class StudentVO implements Serializable {
	private String studentNo = "";
	private String studentName = "";

	public String getStudentNo() {
		return studentNo;
	}

	public void setStudentNo(String studentNo) {
		this.studentNo = studentNo;
	}

	public String getStudentName() {
		return studentName;
	}

	public void setStudentName(String studentName) {
		this.studentName = studentName;
	}
}
```

3）把原有的两个action文件也删了吧（删了就删了，反正我们要用struts2来重写）

4）打开web.xml文件，把下面这些内容去掉

```
<jsp-config>
	<taglib>
```

```
                <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
                <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
        </taglib>
        <taglib>
                <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
                <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
        </taglib>
        <taglib>
                <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
                <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
        </taglib>
</jsp-config>
```

5）打开web.xml文件，把.do都改成.action

6）打开web.xml文件，把这些去掉

```
<servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
                <param-name>config</param-name>
                <param-value>/WEB-INF/struts-config.xml,
                                /WEB-INF/struts-config/login.xml,
                                /WEB-INF/struts-config/index.xml
                </param-value>
        </init-param>
        <init-param>
                <param-name>debug</param-name>
                <param-value>3</param-value>
        </init-param>
        <init-param>
                <param-name>detail</param-name>
                <param-value>3</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
</servlet>
```

7）打开web.xml文件，加入struts2的配置

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

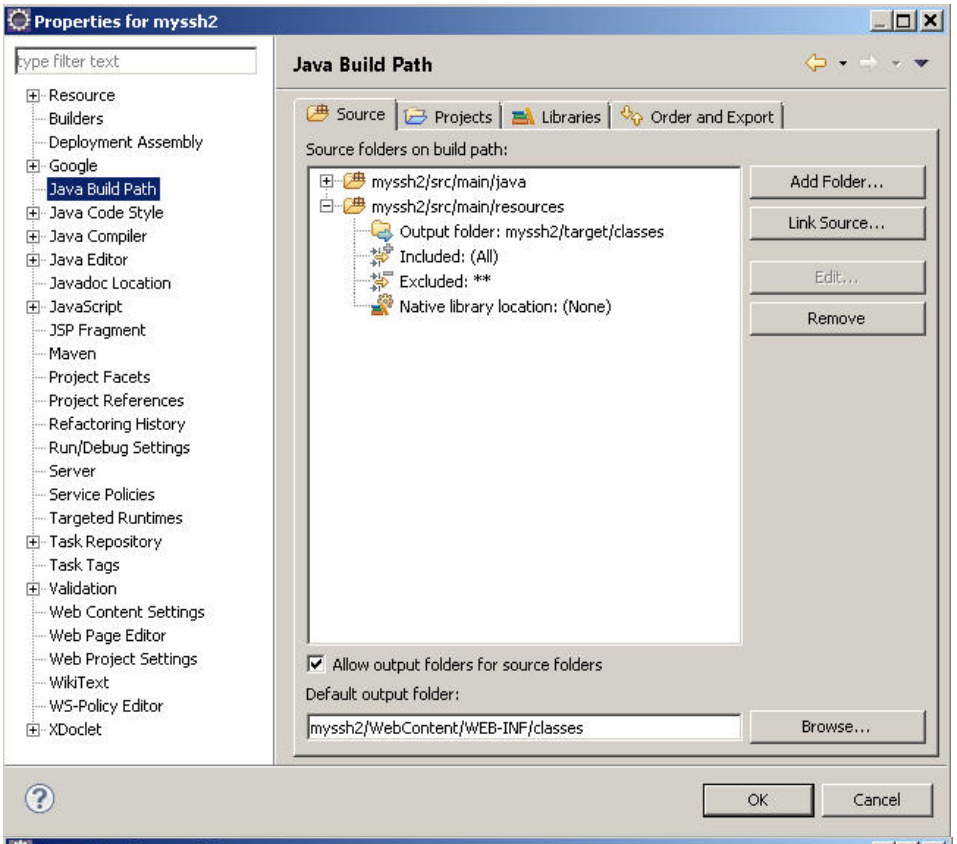
可以看到struts2框架是一个基于filter的框架，确保这个**filter**在所有工程中你自己定义的**filter**的最后面，要不然你自己定义的**filter**会被**struts2**给**filter**掉（这点一定要注意）。

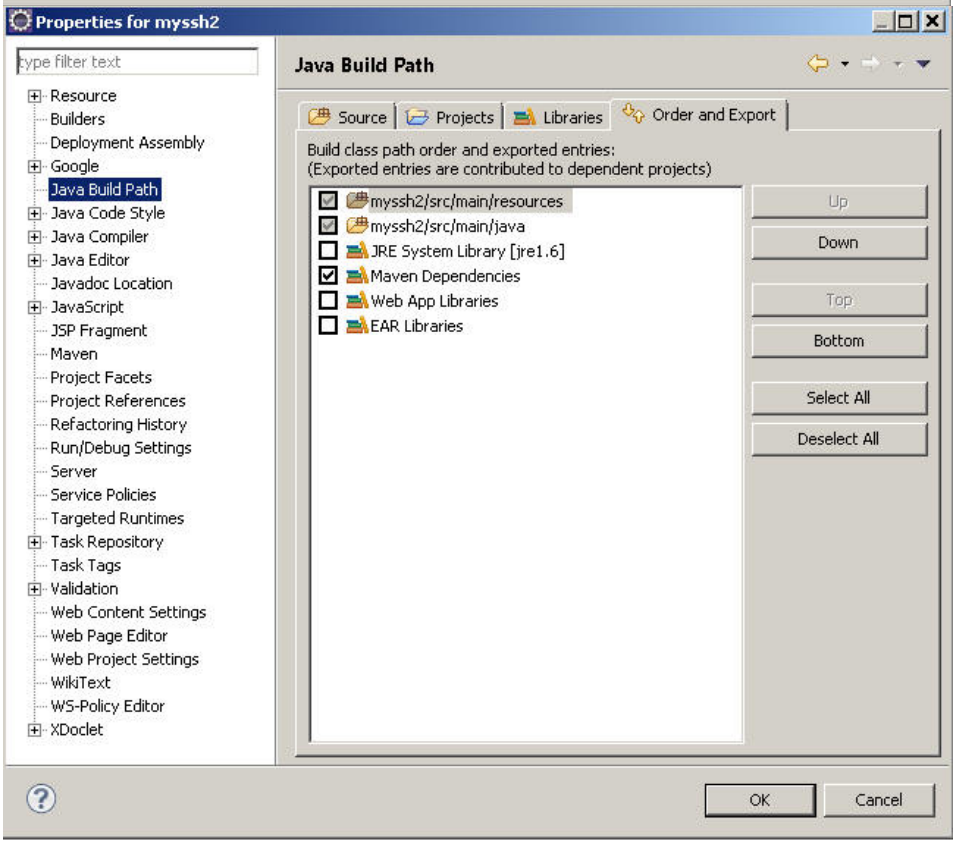
最后别忘了把：

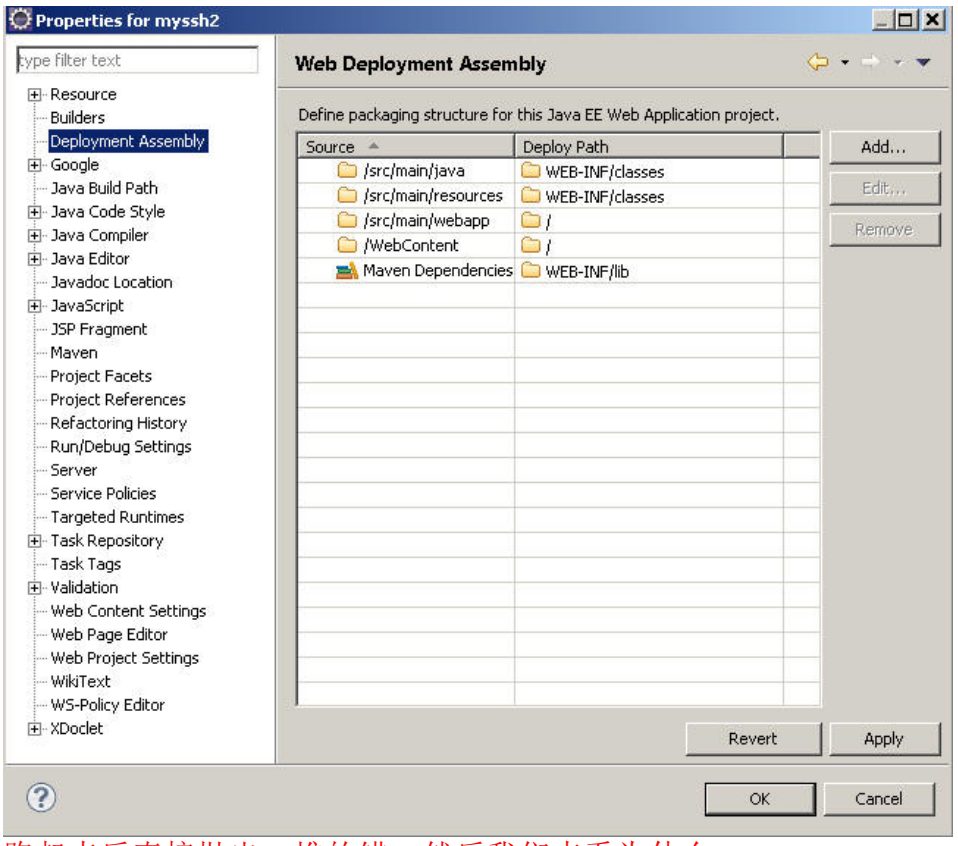
```
<init-param>
        <param-name>exclude</param-name>
        <param-value>/jsp/login/login.jsp,
                                /login.action
        </param-value>
</init-param>
```

这边的原有的/login.do改成/login.action哦。

确保工程编译没有任何问题，然后我们按照番外篇《第十九天》中的“四、如何让**Maven**构建的工程在**eclipse**里跑起来”对工程进行设置，使得工程可以在eclipse的tomcat中跑起来。



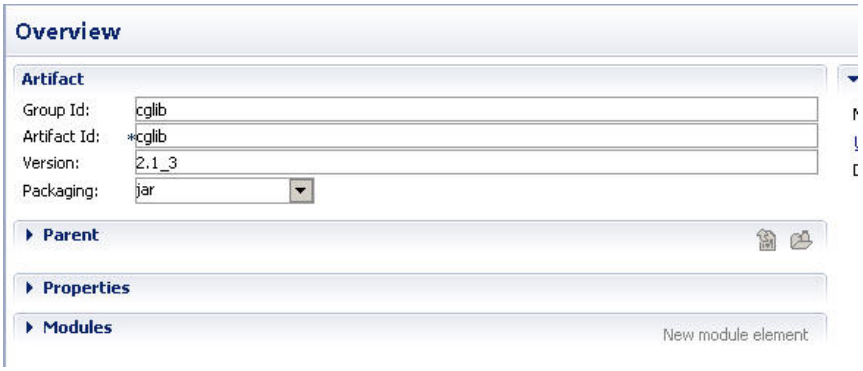


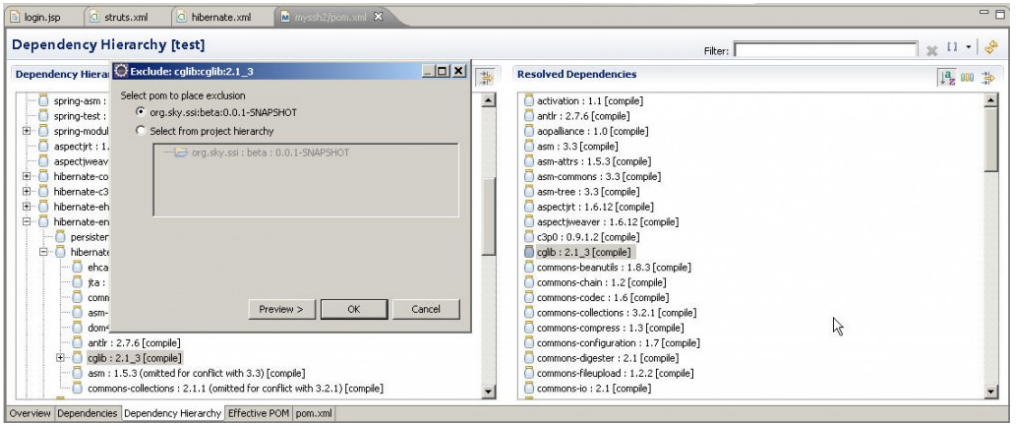


跑起来后直接抛出一堆的错，然后我们来看为什么

就是下面这个狗屁错。

其原因在于由于使用的struts2。原有的这个cglib:2.1.3，这个包对于spring3和hibernate3还有struts1.3来说没有任何问题，在遇到struts2时就冲突了，因此我们需要把这个包也给exclude掉





Exclude掉了后没有cglib包了，AOP类反射没法玩了，怎么办？

简单：

手工在pom.xml文件中添加一个cglib较新版本的包，如下：

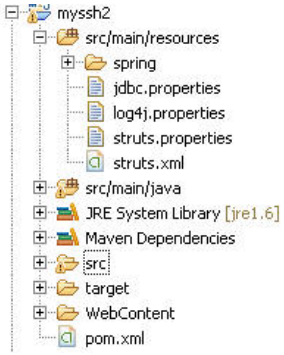
```
<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-entitymanager</artifactId>
		<version>3.3.1.ga</version>
		<exclusions>
			<exclusion>
				<artifactId>cglib</artifactId>
				<groupId>cglib</groupId>
			</exclusion>
		</exclusions>
</dependency>
<dependency>
		<groupId>cglib</groupId>
		<artifactId>cglib-nodep</artifactId>
		<version>2.2</version>
</dependency>
```

再运行工程，一切无误，tomcat启动时正常，说明我们的框架已经搭好了，接下来我们就要开始改我们的struts的action层了。

四、使用struts2重写action层

4.1 struts的配置文件

Struts2只有一个核心的struts.xml文件，它应该放在运行工程的classpath路径下即WEB-INF/classes目录下，所以我们把它放在工程的resources目录下，使得它在工程被编译时会被自动编译到工程的WEB-INF/classes目录下.



其内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
		<include file="struts-default.xml" />
```

```
	<package name="login" extends="struts-default">
		<global-results>
			<result name="error">/jsp/error/syserror.jsp
			</result>
		</global-results>
		<global-exception-mappings>
			<exception-mapping result="error" exception="java.lang.Exception"
/>
		</global-exception-mappings>
		<action name="login" class="org.sky.ssh.login.action.LoginAction">
			<result name="success" type="redirectAction">
				<param name="namespace">/</param>
				<param name="actionName">indexInit</param>
			</result>
		</action>
	</package>
	<package name="index" extends="login">
		<action name="indexInit" class="org.sky.ssh.student.action.StudentAction"
			method="indexInit">
			<result>/index.jsp</result>
		</action>
	</package>
	<package name="studentAdmin" extends="login">
		<action name="popAddStudent"
class="org.sky.ssh.student.action.StudentAction"
			method="popAddStudent">
			<result>/jsp/student/studentAdd.jsp</result>
		</action>
		<action name="addStudent" class="org.sky.ssh.student.action.StudentAction"
			method="addStudent">
			<result>/jsp/student/studentAdd.jsp</result>
		</action>
		<action name="addStudent" class="org.sky.ssh.student.action.StudentAction"
			method="addStudent">
			<result>/jsp/student/studentAdd.jsp</result>
		</action>
		<action name="delStudent" class="org.sky.ssh.student.action.StudentAction"
			method="delStudent">
			<result name="success" type="redirectAction">
				<param name="namespace">/</param>
				<param name="actionName">indexInit</param>
			</result>
		</action>
	</package>
</struts>
```

4.2 struts2中的类的书写

struts2中的action就是一个普通的class，它的public方法名就是一个具体的action，这相当于原有struts1中的DispatchAction，只不过它做的更加灵活。

如：我有一个按钮叫addStudent，如果你在你的action的类中有一个publicString addStudent()方法，那么你的这个按钮对应的action名就叫/addStudent.action（神奇吧）。

我们对应着“**<action name="login"class="org.sky.ssh.login.action.LoginAction">**”来看我们的login.action吧，前面的这个<actionname="">后面的东西就是我们的web路径即相当于“/login.action”。

org.sky.ssh.login.action.LoginAction类

```
package org.sky.ssh.login.action;
```

```
import java.util.Map;

import javax.annotation.Resource;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.struts2.interceptor.ServletRequestAware;
import org.apache.struts2.interceptor.ServletResponseAware;
import org.apache.struts2.interceptor.SessionAware;
import org.sky.ssh.service.LoginService;
import org.sky.ssh.util.Constants;
import org.sky.ssh.util.session.UserSessionInfo;

import com.opensymphony.xwork2.ActionSupport;

public class LoginAction extends ActionSupport implements SessionAware, ServletRequestAware,
ServletResponseAware {
	protected final Log logger = LogFactory.getLog(getClass());
	private Map att;
	private HttpServletRequest request = null;
	private HttpServletResponse response;

	@Resource
	LoginService loginService;

	public void setServletRequest(HttpServletRequest request) {
		this.request = request;
	}

	public void setServletResponse(HttpServletResponse response) {
		this.response = response;
	}

	public void setSession(Map att) {
		this.att = att;
	}

	public String execute() throws Exception {
		String loginId = "";
		String loginPwd = "";
		HttpSession session = request.getSession();
		String loginCode = "100";
		try {
			loginId = (String) request.getParameter("loginId");
			loginPwd = (String) request.getParameter("loginPwd");
			if (loginService.login(loginId, loginPwd)) {
				UserSessionInfo uinfo = new UserSessionInfo();
				uinfo.setLoginId(loginId);
				session.setAttribute(Constants.USER_SESSION, uinfo);
```

```
                } else {
                        loginCode = "101";
                }
                request.setAttribute("loginCode", loginCode);
                if (!loginCode.equals("100")) {
                        RequestDispatcher dispatcher =
request.getRequestDispatcher("/jsp/login/login.jsp");
                        dispatcher.forward(request, response);
                }
                return SUCCESS;
            } catch (Exception e) {
                logger.error("UserLogin Exception:" + e.getMessage(), e);
                throw new Exception("UserLogin Exception:" + e.getMessage(), e);
            }
        }
}
```

在查看此类时注意以下几点地方：

1） 注意request, response, session是怎么被应用到struts2的action的class中去的；

2） public String execute() throws Exception {…}方法就相当于原有struts1.x中的unspecified方法，是被默认执行的；

3） action方法必须返回一个String类型，默认有SUCCESS和FAIL（注意大小写），它就对应着你的struts.xml文件中的：**<result name="success" type="redirectAction">**。

4.3 struts2中的跳转

不带**request**值的跳转写法

```
<result name="success" type="redirectAction">
        <param name="namespace">/</param>
        <param name="actionName">indexInit</param>
</result>
```

注意这个type=”redirectAction”，它只是说明这个跳转是一个redirect，不管你在上一个action的request中setAttribute了什么值，当它顺利到达下一个action或者是.jsp时，它是带不出上一个request中的值的。

如果你要从一个action到一个action或者是从一个action到一个jsp并且要把值在request中带过去该怎么跳呢？如下所示：

带**request**值的从**action**跳**jsp**的写法

```
<action name="indexInit" class="org.sky.ssh.student.action.StudentAction" method="indexInit">
        <result>/index.jsp</result>
</action>
```

这边默认如果action方法返回SUCCESS就会触发这个跳转

带**request**值的从**action**跳**action**的写法

```
<action name="delStudent" class="org.sky.ssh.student.action.StudentAction"
method="delStudent">
        <result name="success" type="redirectAction">
                <param name="namespace">/</param>
                <param name="actionName">indexInit</param>
        </result>
</action>
```

带**request**值的直接从**acton**类中（不通过**struts**配置）的跳转写法

```
RequestDispatcher dispatcher = request.getRequestDispatcher("/jsp/login/login.jsp");
dispatcher.forward(request, response);
```

4.4 /jsp/login/login.jsp

我们打开这个login.jsp，它来自于原有的myssh工程，我们把头上的这些东西去掉，它们是struts1的标签：

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
```

把<form>中的action=后的login.do改成：

```
action="${pageContext.request.contextPath}/login.action"
```

4.5 继续排除maven库的jar包的错误

然后我们启动tomcat，输入http://localhost:8080/myssh2 ，在登录页面敲入相关的用户名/密码后一点登录，直接看到一堆的前台jsp页面显示错误和后台错误。

其原因就是原有的struts1.x的依赖包中会自动带入jstl相关的jar包，它们是：

Jstl和standard两个包，而struts2是不带这两个jar包的依赖的，因为struts2有着自己强大的且丰富的tag。因此我们在使用struts2时，要把这个两个jar包加入到pom.xml文件中去，在pom.xml文件中加入：

```
<dependency>
		<groupId>javax.servlet</groupId>
		<artifactId>jstl</artifactId>
		<version>1.0.2</version>
</dependency>
<dependency>
		<groupId>taglibs</groupId>
		<artifactId>standard</artifactId>
		<version>1.0.6</version>
</dependency>
```

4.6 使用struts2风格在request中set一个list

在类中声明一个局部变量且创建一对set{...}get{...}

```
private List<StudentVO> stdList = new ArrayList<StudentVO>();
public void setStudentVO(StudentVO studentVO) {
		this.studentVO = studentVO;
}
public List<StudentVO> getStdList() {
		return stdList;
}
```

此时你一旦在某个public方法中对这个值进行过操作，那么在这个action跳转到下一个jsp时，你在jsp的request中会自动取得这个list的值而不需要在原有的struts的action中写诸如：setAttributer(“stdList”,stdList);这样的东西了，是不是很优雅？

5. org.sky.ssh.student.action. StudentAction类

```
package org.sky.ssh.student.action;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import javax.annotation.Resource;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.struts2.interceptor.ServletRequestAware;
import org.apache.struts2.interceptor.ServletResponseAware;
import org.apache.struts2.interceptor.SessionAware;
import org.sky.ssh.service.StudentService;
import org.sky.ssh.vo.StudentVO;

import com.opensymphony.xwork2.ActionSupport;
```

```
public class StudentAction extends ActionSupport implements SessionAware,
ServletRequestAware, ServletResponseAware {
	private Map att;
	private HttpServletRequest request = null;
	private HttpServletResponse response;
	private List<StudentVO> stdList = new ArrayList<StudentVO>();
	private StudentVO studentVO = new StudentVO();

	public StudentVO getStudentVO() {
		return studentVO;
	}

	public void setStudentVO(StudentVO studentVO) {
		this.studentVO = studentVO;
	}

	public List<StudentVO> getStdList() {
		return stdList;
	}

	public void setStdList(List<StudentVO> stdList) {
		this.stdList = stdList;
	}

	@Resource
	private StudentService stdService = null;
	protected final Log logger = LogFactory.getLog(getClass());

	public void setServletRequest(HttpServletRequest request) {
		this.request = request;
	}

	public void setServletResponse(HttpServletResponse response) {
		this.response = response;
	}

	public void setSession(Map att) {
		this.att = att;
	}

	public String indexInit() throws Exception {
		try {
			stdList = stdService.getAllStudent();
			return SUCCESS;
		} catch (Exception e) {
			logger.error("Init Index Exception:" + e.getMessage(), e);
			throw new Exception("Init Index Exception:" + e.getMessage(), e);
		}
	}

	public String popAddStudent() throws Exception {
		return SUCCESS;
	}
```

```
	public String addStudent() throws Exception {
		try {
			stdService.addStudent(studentVO.getStudentName());
		} catch (Exception e) {
			logger.error("addStudent error:" + e.getMessage(), e);
			throw new Exception("addStudent error:" + e.getMessage(), e);
		}
		return SUCCESS;
	}

	public String delStudent() throws Exception {
		String[] stdArray = null;
		try {
			stdArray = request.getParameterValues("selectedStudents");
			if (stdArray != null && stdArray.length > 0) {
				stdService.delStudent(stdArray);
			}
		} catch (Exception e) {
			logger.error("delStudent error:" + e.getMessage(), e);
			throw new Exception("delStudent error:" + e.getMessage(), e);
		}
		return SUCCESS;
	}
}
```

6. myssh2工程的完整pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>org.sky.ssi</groupId>
	<artifactId>beta</artifactId>
	<packaging>war</packaging>
	<version>0.0.1-SNAPSHOT</version>
	<name>Alpha_MVN Maven Webapp</name>
	<url>http://maven.apache.org</url>
	<dependencies>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>3.8.1</version>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>log4j</groupId>
			<artifactId>log4j</artifactId>
			<version>1.2.8</version>
		</dependency>
		<dependency>
			<groupId>c3p0</groupId>
			<artifactId>c3p0</artifactId>
			<version>0.9.1.2</version>
		</dependency>
		<dependency>
```

```
		<groupId>jaxen</groupId>
		<artifactId>jaxen</artifactId>
		<version>1.1.1</version>
		<exclusions>
			<exclusion>
				<artifactId>xercesImpl</artifactId>
				<groupId>xerces</groupId>
			</exclusion>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.apache.struts</groupId>
		<artifactId>struts2-spring-plugin</artifactId>
		<version>2.3.1.2</version>
	</dependency>
	<dependency>
		<groupId>org.apache.struts</groupId>
		<artifactId>struts2-core</artifactId>
		<version>2.3.1.2</version>

		<exclusions>
			<exclusion>
				<artifactId>tools</artifactId>
				<groupId>com.sun</groupId>
			</exclusion>
		</exclusions>
	</dependency>
	<!-- springframework 3.1 -->
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-struts</artifactId>
		<version>3.1.0.RELEASE</version>
		<exclusions>
			<exclusion>
				<artifactId>struts</artifactId>
				<groupId>struts</groupId>
			</exclusion>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-core</artifactId>
		<version>3.1.0.RELEASE</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-context</artifactId>
		<version>3.1.0.RELEASE</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-context-support</artifactId>
		<version>3.1.0.RELEASE</version>
	</dependency>
```

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc-portlet</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jms</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-asm</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springmodules</groupId>
        <artifactId>spring-modules-jakarta-commons</artifactId>
        <version>0.8a</version>
```

```
</dependency>
<!-- aspectj -->
<dependency>
	<groupId>org.aspectj</groupId>
	<artifactId>aspectjrt</artifactId>
	<version>1.6.12</version>
</dependency>
<dependency>
	<groupId>org.aspectj</groupId>
	<artifactId>aspectjweaver</artifactId>
	<version>1.6.12</version>
</dependency>
<!-- hibernate 3.3.1 -->
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-core</artifactId>
	<version>3.3.1.GA</version>
</dependency>
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-c3p0</artifactId>
	<version>3.3.1.GA</version>
</dependency>
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-ehcache</artifactId>
	<version>3.3.1.GA</version>
</dependency>
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-entitymanager</artifactId>
	<version>3.3.1.ga</version>
	<exclusions>
		<exclusion>
			<artifactId>cglib</artifactId>
			<groupId>cglib</groupId>
		</exclusion>
	</exclusions>
</dependency>
<dependency>
	<groupId>cglib</groupId>
	<artifactId>cglib-nodep</artifactId>
	<version>2.2</version>
</dependency>
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-commons-annotations</artifactId>
	<version>3.3.0.ga</version>
</dependency>
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-annotations</artifactId>
	<version>3.3.1.GA</version>
</dependency>
```

```
<!-- log4j 1.2.14 -->
<dependency>
	<groupId>log4j</groupId>
	<artifactId>log4j</artifactId>
	<version>1.2.16</version>
</dependency>
<dependency>
	<groupId>org.slf4j</groupId>
	<artifactId>slf4j-api</artifactId>
	<version>1.5.10</version>
</dependency>
<dependency>
	<groupId>org.slf4j</groupId>
	<artifactId>slf4j-log4j12</artifactId>
	<version>1.5.10</version>
</dependency>
<!-- commons utils -->
<dependency>
	<groupId>commons-beanutils</groupId>
	<artifactId>commons-beanutils</artifactId>
	<version>1.8.3</version>
</dependency>
<dependency>
	<groupId>commons-chain</groupId>
	<artifactId>commons-chain</artifactId>
	<version>1.2</version>
</dependency>
<dependency>
	<groupId>commons-codec</groupId>
	<artifactId>commons-codec</artifactId>
	<version>1.6</version>
</dependency>
<dependency>
	<groupId>commons-collections</groupId>
	<artifactId>commons-collections</artifactId>
	<version>3.2.1</version>
</dependency>
<dependency>
	<groupId>commons-configuration</groupId>
	<artifactId>commons-configuration</artifactId>
	<version>1.7</version>
</dependency>
<dependency>
	<groupId>commons-digester</groupId>
	<artifactId>commons-digester</artifactId>
	<version>2.1</version>
</dependency>
<dependency>
	<groupId>commons-fileupload</groupId>
	<artifactId>commons-fileupload</artifactId>
	<version>1.2.2</version>
</dependency>
<dependency>
	<groupId>commons-io</groupId>
```

```
            <artifactId>commons-io</artifactId>
            <version>2.1</version>
        </dependency>
        <dependency>
            <groupId>commons-lang</groupId>
            <artifactId>commons-lang</artifactId>
            <version>2.6</version>
        </dependency>
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.1.1</version>
        </dependency>
        <dependency>
            <groupId>commons-net</groupId>
            <artifactId>commons-net</artifactId>
            <version>3.0.1</version>
        </dependency>
        <dependency>
            <groupId>commons-pool</groupId>
            <artifactId>commons-pool</artifactId>
            <version>1.6</version>
        </dependency>
        <dependency>
            <groupId>commons-validator</groupId>
            <artifactId>commons-validator</artifactId>
            <version>1.3.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-compress</artifactId>
            <version>1.3</version>
        </dependency>
        <!-- jsp servlet api -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.4</version>
            <scope>compile</scope>
        </dependency>
        <!-- mail -->
        <dependency>
            <groupId>org.apache.velocity</groupId>
            <artifactId>velocity</artifactId>
            <version>1.7</version>
        </dependency>
        <dependency>
            <groupId>javax.mail</groupId>
            <artifactId>mail</artifactId>
            <version>1.4.4</version>
        </dependency>
        <!-- jasypt -->
        <dependency>
            <groupId>org.jasypt</groupId>
```

```
                <artifactId>jasypt</artifactId>
                <version>1.9.0</version>
            </dependency>
            <dependency>
                <groupId>org.jasypt</groupId>
                <artifactId>jasypt-spring3</artifactId>
                <version>1.9.0</version>
            </dependency>
            <dependency>
                <groupId>org.jasypt</groupId>
                <artifactId>jasypt-springsecurity3</artifactId>
                <version>1.9.0</version>
            </dependency>
            <!-- ehCache -->
            <dependency>
                <groupId>net.sf.ehcache</groupId>
                <artifactId>ehcache</artifactId>
                <version>1.6.2</version>
            </dependency>
            <!-- test -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.10</version>
            </dependency>
            <dependency>
                <groupId>javax.servlet</groupId>
                <artifactId>jstl</artifactId>
                <version>1.0.2</version>
            </dependency>
            <dependency>
                <groupId>taglibs</groupId>
                <artifactId>standard</artifactId>
                <version>1.0.6</version>
            </dependency>

            <dependency>
                <groupId>org.dbunit</groupId>
                <artifactId>dbunit</artifactId>
                <version>2.4.8</version>
            </dependency>
            <dependency>
                <groupId>mockit</groupId>
                <artifactId>jmockit</artifactId>
                <version>0.999.4</version>
            </dependency>
        </dependencies>
        <build>
            <finalName>myssh2</finalName>
        </build>
</project>
```

7. SSH1还是SSH2与Annotation还是Xml配置的问题

这个问题是没有绝对的优点和缺点的。

有人喜欢说：我就爱用SSH2，因为都是最新的，我也喜欢用全annotation的方式来编程，因为这样做比较潮

流，比较优雅。但是。。。。。。这些都不是真正的justification。

对于在框架选型时，不仅仅是开发者自己喜欢不喜欢的问题，就和有人说：开发者们永远喜欢推倒重做，永远喜欢开发新项目而对于修修改改维护类项目感冒是一个道理。

试想，大部分的金融保险客户，他们的系统都是有一定的年头了，而且像这样的企业中的一个IT项目是不可能跟着潮流经常去变化的，因为这些企业中的IT项目或者我们称作IT资产涉及到数据、机密性、稳定性，一旦这个项目自上线之日起它就一直在稳定的运行了，除非是重大变故一般是不会轻易去改它的架构或者是核心的，一般都是围绕着已有项目来进行扩展和维护，因此这些企业中的J2EEAPP Server或者是JDK版本都不一定是最新的。比如说有些银行到现在还一直在用was6.1，或者是weblogic9.x，更有甚者还在用jdk1.4，如果这时你在接手项目时不去了解企业的现状况，一拍脑袋说：我们用SSH2吧！结果你的项目做完后连上线都无法上线，到那时就不仅仅是再让你重构的问题了，呵呵，对吧.

你不要试图去和客户解释说：唉呀你怎么还在用WAS5.1，你怎么还在用Tomcat5.5啊，我给你搞个tomcat6.x也行，把你的JDK装成1.6吧，呵呵，千万不要这么做。

客户可以告诉你，它的服务器是小型机，上百万元购买来的，购买小型机时赠送了WAS5.1因此在上面有许多的应用且已经使用了5年之久了，现在你为了说你的框架是STRUTS2而逼着客户升级JDK和WAS版本，如果万一有问题了，出错了，导致了客户的实时交易延误而引起的经济损失，你能支付得起吗？如果你说因为我们的环境而不能使用你们的框架，那么对不起，我们公司不会采用你方的架构。呵呵！！

框架的目的在于最大程度上减化一些底层的，重复性的劳动，把对数据库的访问，对resource等的访问从程序员的实际工作中分离出来，使程序员有更多的时间去关注“业务逻辑”——摘自2001版的《EJB2从入门到精通》。所以在实际工作中不能够为了用框架而用框架。

好比，我用Annotation写DAO是很方便，很优雅，但是你有没有想过，当你的SQL如果是经常需要变，或者是需要通过外部动态传入的时候甚至允许客户自己构建SQL再传给我们的DAO的场景下，那么对于我们来说只需要改改SQL逻辑重新启动一下服务器就可以实现的而因为你用了全部基于Annotation的框架，我甚至需要去动我们的代码，要知道代码不管你动了多少哪怕你只是加了一个注释也是需要按照流程来重新测试、重新打包的。因为没有人敢保证你的改动不引起regressionbug。所以这时把sql或者一些配置写成xml或者是properties的外部配置形式要比你用annotation来得更灵活，这就是我在第十八天的spring+jdbcTemplate时为什么喜欢把SQL写在XML里再通过spring注入到DAO层的原因。因为你的SQL不是一次写成的，就算是一次写成，你的工程在将来也会面临SQL调优这么一个过程，到时你每改动一次SQL，就要动一次代码层，而你的改动可能只是把in变成了=或者是把innerjoin改成hashjoin，那么此时我的SQL如果是配置在外部配置文件中的话我改起来是不是更方便？尤其是一些涉及到大数据量出报表的SQL是经常面临调优的。

Struts2是基于filter框架的，你可以使用它的filter，你甚至可以不用去使用spring而直接使用struts2或者使用spring的MVC而抛弃struts2，都是没问题的，没有什么所谓“不正统/正统”框架之说。好比我有一个servlet叫LoginFilter，这个filter诞生在8年前，经历了好几个项目了已经是非常稳定了，因此当我碰到了struts2的框架时我不是说因为struts2的技术新我就必须全部用struts2来重做我的feature，稳定性重用性在哪里？我既然手上有一个这么稳定的历经了好几年的一些个组件，虽然它们历史久远了些可是我也是照用不误，原因就在于它稳定实用。

说了这些，主要的目的还是要告诉大家，框架和设计模式是一样的，它只是在最大程度上解放你的生产力，减少你的重复劳动，避免了不要去重复造轮子。不要为了框架而框架，不要被框架套死。好比刚练武时，一招一式都要照着书本和师傅的样子去学，但是真正的武功高手是什么样的呢？“无招胜有招”，对不对？活用活用，要把框架和模式为你所用而不是做框架的奴隶。

这也是我为什么强调框架而不仅仅是强调SSX体系的原因，其实在我的SSX框架中还经常可以看到一些古老的jstl,servlet的存在，我的目的就在于充分利用各个技术的优点，把各个技术各个框架的优点集中起来使用这样才能搭出一本葵花宝典来。

小知识普及 SSH与SSH2这种框架组合的历史原由

早在2001年时当时的J2EE推崇的是EJB，EJB被称为J2EE的核心，当时要学J2EE就是Servlet+EJB，在EJB里其实早已经有了AOP与实体映射这些概念了。

EJB有三种形态的BEAN，SessionBean, Entity Bean, MBEAN对吧？其中，EntityBean就是Hibernate，大家看看，嘿嘿，所以技术这个东西所谓的新也是换汤不换药，在2001时就已经有了Hibernate这种概念了，而且Spring经典的声明式事务代理也早就有了，就是你的SessionBean如果抛出一个java.runtime.exception，EJB容器就会自动回滚事务，而且我们在声明EntityBean时就是和Hibernate2.x一样，写一个xml文件将字段表名和类名和属性名进行一一对应的。

但是有许多人会说ejb2.0是一个失败之作，它的实体映射隐藏了数据库底层的操作把对于表的操作转化成了OOP的操作，这是一个非常好的理念，但是在早期的EJB中连对于如：selectcount(), select max()这样的操作都没有，当然在那时如果碰到这样的处理时有经验的程序员一般会采用EJB中的BPM即直接使用SessionBean+jdbc去完成的，但是就如我前面所说的，为了使用框架而用框架的事情和人数大有所在，为了在工程中使用一套纯正

的EJB，纯CMP BEAN，很多程序员们就去用ArrayList.size或者是Vector.size来做这些个count,max等操作，甚至在碰到多表连接时对于每个表取一个List然后在Java代码层去用数据结构来拼装出一个View来。

EJB2.x的配置也是非常繁琐的，没有一个好的现成的工具，一般不包括MBEAN的话仅要使用SessionBean和EntityBean就要配4个xml文件，同时每个EJB的容器如：JBOSS,WEBLOGIC,WAS又彼此间不能通用，在使用不同的J2EE容器时还要为这个容器单独配一个厂商支持的xml文件。

等等等等。。。。。。这一系列导致了使用EJB的工程变得臃肿复杂，难于调试，并伴有严重的性能问题，当时网上骂声也是一片，EJB一度走入低谷。于是，人们就在想，我能否保留EJB中CMPBean的这种实体映射的特性呢？而且我也只希望使用实体映射，于是Hibernate诞生了.Hibernate就是一个除去了EJB2.x一切特性只保留EntityBean的一种技术。

03年，04年随着Hibernate的推广，人们又在想，Hibernate现在有了，EJB原有的声明式事务也是一个不错的设计，我可以让程序员不需要关心他们的数据库层面的transaction处理而只关心业务层面的transaction，所以原有EJB2.x中的AOP概念又被单独剥离了出来，这导致了Spring的诞生.

于是在早期的人们采用Spring+Hibernate这样的架构时，大家其实还是在把这两者的组合在当作EJB来使用的，这样的组合其实就是一个轻量级的EJB2.x，一个缩微了的EJB。因为EJB的设计太超前太好了，只是它的一些缺陷一些瓶劲导致程序员们错用乱用EJB而给EJB造成了不好的口碑，但是因为J2EE的核心就是EJB，因此在04年对于Spring+Hibernate这样的组合有一句口号叫“Thisis not a J2EE”，因为我们不是EJB但我能做到EJB所有的优点，呵呵。这其实就是一种无招胜有招的典形应用场景，把各自的优点最大化的发挥出来而不拘泥于框架而来论框架.

当然，随着EJB3的回归，EJB反过来吸收了Hibernate3与Spring3的一切优点而且它借助着SUN（现在叫ORACLESUN）的工业标准和强大的技术支持，J2EE终还将回归EJB。

EJB3前途无量，它不仅仅把SSH全部又整回成了一个EJB还简化了配置，同时还彻底做到了厂商无关，数据库无关。如著名的SEAM3框架，SCA编程模形（EJB3的SessionBean中可以调用Webservice，这为EJB满足SCA编程模型中的引用、导入等概念带来了极大的便利）都是基于EJB3的，大家有时间我觉得还是可以好好的去关注和学习EJB3技术。

结束今天的教程，下次开始要讲在SSX体系中如何来做unit testing以及如何使用Spring来构建一个单独运行的应用程序如：银行保险业中的批处理业务的框架的搭建。

上一篇 通向架构师的道路（第二十一天）万能框架spring(三)之SSH
下一篇 通向架构师的道路（第二十三天）maven与ant的奇妙整合

查看评论

12楼 dlc1023 2014-02-27 10:20发表

袁老师，你好！
我把ssh2的数据改成mysql插入数据的时候会报错：
Hibernate: insert into T_STUDENT (STUDENT_NAME, STUDENT_NO) values (?, ?)
2014-02-27 10:20:06 WARN JDBCExceptionReporter:77 - SQL Error: 0, SQLState: 07001
2014-02-27 10:20:06 ERROR JDBCExceptionReporter:78 - No value specified for parameter 2
2014-02-27 10:20:06 ERROR LoggerAdvice:62 - could not insert: [org.sky.ssh.model.TStudent]; SQL [insert into T_STUDENT (STUDENT_NAME, STUDENT_NO) values (?, ?)]; nested exception is org.hibernate.exception.SQLGrammarException: could not insert: [org.sky.ssh.model.TStudent]
org.springframework.dao.InvalidDataAccessResourceUsageException: could not insert: [org.sky.ssh.model.TStudent]; SQL [insert into T_STUDENT (STUDENT_NAME, STUDENT_NO) values (?, ?)]; nested exception is org.hibernate.exception.SQLGrammarException: could not insert: [org.sky.ssh.model.TStudent]
我在数据里面student_no是自动递增，在TStudent类里面改成这样：
@Id
@GeneratedValue
@Column(name="STUDENT_NO")
private int studentNo;
是还需要修改什么地方吗，麻烦了！

Re: dlc1023 2014-05-06 14:28发表

回复dlc1023：原来是hibernate包问题，换一个ok了

11楼 wlgpep 2013-10-29 23:35发表

struts2出问题以后,我把项目中的strtus2的jar包升级到了struts-2.3.15.3,项目启动就报错,在网上找了很久没有解决,还要麻烦楼主给看看啊!
不知道是不是包冲突了,要怎么解决?
org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'hibernateSessionFactory' defined in ServletContext resource [/WEB-INF/classes/spring/hibernate/hibernate.xml]: Instantiation of bean failed; nested exception is org.springframework.beans.BeanInstantiationException: Could not instantiate bean class [org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean]: Constructor threw exception; nested exception is java.lang.NoClassDefFoundError: javax/persistence/Entity

谢谢了!

Re: 红肠啃僵尸 2013-10-30 17:39发表

回复wlgpep：haha

依赖包出问题了，因为STRUTS的JAR升了，影响到了其它的包，你留下EMAIL，我给你发一份基于STRUTS2.1.35的完整 的MAVEN的DEPENDENCY包来

Re: wlgpep 2013-10-31 09:51发表

回复红肠啃僵尸：袁老师非常感谢您啊!136150345@qq.com以后有什么问题还望不吝赐教啊!谢谢了!

Re: 红肠啃僵尸 2013-10-31 16:05发表

回复wlgpep：依赖包关系已经发你邮箱了，请查收

10楼 u011894119 2013-10-15 21:48发表

楼主，能讲一些spring底层技术和案件吗？因为看你讲的都挺通俗易懂的。。。

9楼 bb_tarek 2013-05-19 14:48发表

博主最后那段话很有道理，现在公司的项目用的是struts1 tomcat5.5 jdk1.4, DAO层还用的是jdbc...

8楼 红肠啃僵尸 2012-12-02 18:18发表

http://download.csdn.net/detail/lifetragedy/4837507
以上地址是重新整理了的sample的下载地址，不好意思，没能及时上传。

7楼 lamplk 2012-11-22 16:04发表

感谢楼主的无私分享，我也相信EJB会回来的

6楼 红肠啃僵尸 2012-11-19 09:53发表

回复红肠啃僵尸：给你找到在线安装的地址了：http://download.eclipse.org/technology/m2e/releases/在eclipse里->help->install new software把上面这个网址输入后过一会会搜出maven eclipse integration，勾选后在线升级吧，很小的.

Re: k1191e 2012-11-19 21:36发表

回复红肠啃僵尸：博主真是热心，感激不尽啊.....

5楼 水哥709 2012-11-19 09:26发表

在理啊！你可以知道怎么搭建一套框架，但是你不一定知道为什么要这样搭框架，为什么用这样的框架，你也不一定知道这样的框架组合和搭建方式是不是最好的，有没有更好的。论述很好啊，知其然，还要知其所以然。

4楼 k1191e 2012-11-18 13:42发表

崩溃了，小菜伤不起啊；博主你还是直接给我一个吧...,搞得头晕脑胀的.....，眼睛都痛死了

Re: 红肠啃僵尸 2012-11-19 09:53发表

回复就这样吧呵呵：在eclipse里->help->install new software把上面这个网址输入后过一会会搜出maven eclipse integration，勾选后在线升级吧，很小的.

3楼 yaerfeng 2012-11-17 23:50发表

呵呵。持续关注袁大哥的博客中。希望关于一些底层的东西和经验可以扩展的提示一下。比如像这样的玩意： implements SessionAware, ServletRequestAware, ServletResponseAware，对手那些新手，可能如我当初一般不解。

2楼 乐山乐水2015 2012-11-17 00:45发表

顶楼主。

1楼 红肠啃僵尸 2012-11-17 00:00发表

与该天教程相关的myssh2工程已经上传至了我的博客的“资源”中了，大家可以免费下载.

Re: 红肠啃僵尸 2012-11-17 00:24发表

回复红肠啃僵尸：myssh2示例工程的下载地址为：http://download.csdn.net/detail/lifetragedy/4776391