MyEclipse 开发 SSH2（Struts2+Spring+Hibernate）应用之“超光速”教程

摘要：SSH2（Struts2+Spring+Hibernate）是当前开发 Java 动态网站的流行框架。虽然其涉及的技术较为繁杂，但是我们根据软件工程的原则，避开底层繁杂的技术细节，并分离页面设计和后台编程任务，仅着重于应用层的构建，是可以在仅具有 Java 编程知识和简单 HTML 标签的基础上，快速开发基于 SSH2 的应用系统的。由于该快速入门最快可以在 1 小时之内完成而不失其典型示范作用，故称之为“超光速”教程。
关键字：SSH 框架，动态网站，Java，软件工程

规范化的软件工程，通常要求在系统分层的前提下，划分清楚任务界限，使用简明的接口在不同层次或模块之间交互，避免交叉引用和高耦合等。这些原则对于 SSH 这样一个框架，无疑是非常适合的，因为在 Struts+Spring+Hibernate 的构架模式下，人机交互层、业务层和数据处理层分别由 Struts、Spring 和 Hibernate 完成，同时避免页面代码（从而将程序员和页面设计员严格分工，实现解耦），则 Java 程序员可以专心于后台程序设计，实现快速开发的目的。
本文在 MyEclipse 开发环境下，演示快速 SSH 应用开发的一般过程，同时理解性地介绍相关的 SSH 基本概念，及常见错误的处理方法。除了 Java 编程和 HTML 标签外，您还需要了解 HTML“表单”的概念：通过表单（form）可以将在 web 页面上使用文本框、单选按钮、复选框等收集用户输入，并提交（submit）给系统后台。后台数据处理的结果，也可以通过表单带到前台页面中；阅读本文不需要有 JSP 语言的知识基础。

1. 开发环境
本文开发环境为 MyEclipse7.5 和 Struts 2.2.1，Spring 和 Hibernate 使用 MyEclipse 内置的版本。注意下载 Struts 时，要下载完全版 Full Distribution，该版中包括样例“空工程”struts2-blank-2.2.1.war（在 apps 目录下），这个文件中有我们快速开发可以套用的各种文件。附注：war 文件是部署 Web 应用系统时用的压缩文件格式，但是也可以使用 RAR 等程序解压。

2. 建立 Web 工程并导入 Struts 支持包
在“空工程”的 WEB-INF\lib 目录下，是支持 Struts 所需要的最简化的 jar 包，它们是：

```
commons-fileupload-1.2.1.jar
commons-io-1.3.2.jar
freemarker-2.3.16.jar
javassist-3.7.ga.jar
ognl-3.0.jar
struts2-core-2.2.1.jar
xwork-core-2.2.1.jar
```

将这些包解压到一个目录中（例如 C:\struts），然后在 MyEclipse 中新建一个“Web Project”，命名为“WarpSpeed”并勾选 Java EE 5.0（您需要在系统中已经安装 JDK5.0 以上的版本）；在工程名上面点鼠标右键，选择“Build Path - Configure Build Path”调出配置对话框（如图 1），在该对话框中选择“Libraries”选项卡，使用“Add External JARs”按钮，将上面的 7 个 jar 文件添加到工程目录中。这样 WarpSpeed 工程就已经支持 Struts2 了。

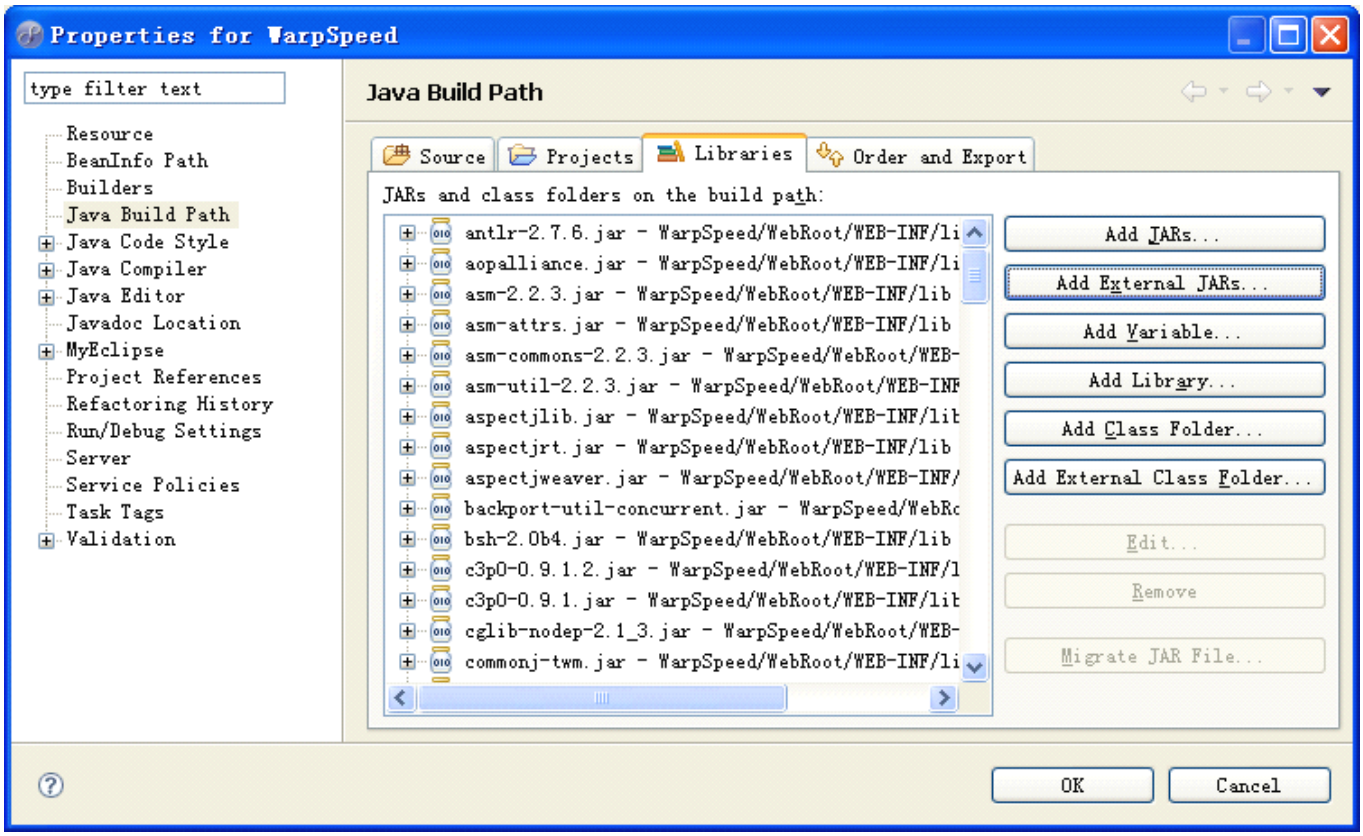

图 1 工程 Build Path 配置窗口

3. 设置全局配置 xml 文件以支持 Struts 过滤器

在系统中加入 Struts 支持，实际上就是要在系统中增加一个 Struts“过滤器（filter）”；所有的文件，在以页面形式展示到用户的浏览器之前，先要通过该过滤器“过滤”（即处理）一遍，这样给了 Struts 控制模块一个处理页面中特有的 Struts 标签的机会；也就是说，后台程序可以将这些 Struts 标签“翻译”成为相关的数据并处理后，才将生成的页面提交给终端用户。

在系统中添加 Struts“过滤器”的方法，是设置系统的全局配置文件 web.xml；该文件在工程的树形结构中位于 WebRoot/WEB-INF 目录下。我们暂不需要理解复杂的配置语法，只需要在 MyEclipse 的 Package 窗口中，找到该文件并双击，即可进入图形化的 web.xml 配置界面（如图 2）。该界面中，树型结构的第二项，即是“Fileters”，选中后使用窗体右边的“Add”按钮，弹出“Add Filters”对话框；这个框中只有两项带“*”号的项目是必填的：

第一项是要为我们的过滤器起一个名字，你可以随便起一个好记的名称，例如“s2”；

第三项是实现该过滤器的 Java 类；点击“Browse”按钮，输入 StrutsPrepareAndExecuteFilter（实际上你只需要输入前几个字母，例如 strutsp 就可找到该类），将该类设为过滤器类即可。

添加完过滤器后，还需要指定哪些文件必须通过该过滤器。在我们的例子中，我们简单地要求所有的文件都要通过 Struts 过滤器，因此我们可以这样来设置过滤器的“mapping”：点击 web.xml 树形目录的 Filters，在右边“Filter Mappings”列表框旁边，点击“Add”按钮，输入我们刚才的过滤器名“s2”，并在“URL-Pattern”处输入“/*”，即要求系统根目录下所有的文件都需要通过该过滤器处理。保存 web.xml 文件；这时我们的 WarpSpeed 工程可以说已经完全支持 Struts2 了。下面我们验证一下。

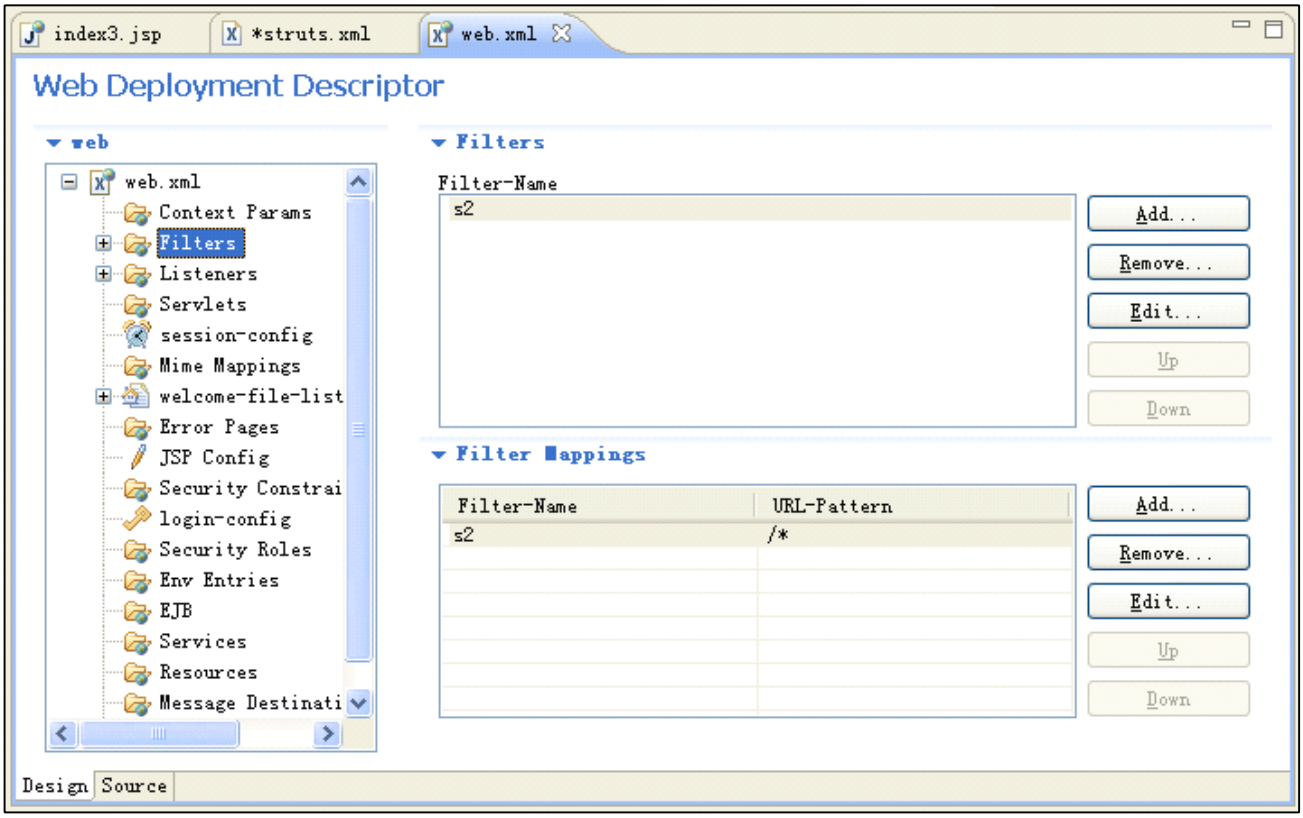

图 2 配置 web.xml 添加 Struts 过滤器

4. 设计一个最简单的 Struts2 系统

在设计一个最简单的 Struts 系统之前，我们需要理解这样两个概念：“action”的执行和导航。

(1) 所谓“action”（“动作”），就是在表单提交后系统会自动执行的一个 Java 类，该类必须继承 ActionSupport（在 com.opensymphony.xwork2 中）并重写其中的 execute()方法。表单提交后，系统会将表单里包含的字段数据传递给该 action 类，并执行其中的 execute()方法。

(2) execute()方法必须返回一个字符串，而该字符其将决定系统要转向那个页面；这就是所谓的“导航”。

因此，一个最简单的 Struts 系统，起码包括这样几个文件：

带有表单的页面文件（jsp 文件）
表单提交后要执行的 action（java 类）
Action 执行完毕要转向的页面（jsp 文件）

因此，在构建系统前，我们首先要构思我们的系统到底需要怎样的功能结构，以及需要哪些文件来实现这些功能——这正是软件工程的“设计”阶段的任务。作为最简单的例子，我们可以这样设计我们的系统：

(1) 页面 index.jsp，其中包含一个简单的 form，该 form 的 action 名为“aCheck”，其中包含两个文本字段：name 和 password；
(2) 表单提交后要执行的 action 类为 AccountCheck；
(3) AccountCheck 类执行时，打印出传递过来的 name 和 password 参数的值
(4) AccountCheck 执行后转向页面 index2.jsp。

为了将这一个构思传递给系统，我们需要使用 struts.xml 来配置这几个文件之间的逻辑关系。在“空”样例工程 struts2-blank-2.2.1.war 的 WEB-INF\src\java 目录下可以找到一个 example.xml 文件，该文件可以作为我们编写 struts.xml 配置文件的起点。其内容为：

```
<?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>
```

```
    <package name="example" namespace="/example" extends="struts-default">

        <action name="HelloWorld" class="example.HelloWorld">
            <result>/example/HelloWorld.jsp</result>
        </action>

        <action name="Login_*" method="{1}" class="example.Login">
            <result name="input">/example/Login.jsp</result>
            <result type="redirectAction">Menu</result>
        </action>

        <action name="*" class="example.ExampleSupport">
            <result>/example/{1}.jsp</result>
        </action>

        <!-- Add actions here -->
    </package>
</struts>
```

在上面的代码中，有四行代码我们做了下划线标记。我们采用第四行带下划线的代码的语法形式（即含有“name”属性的 result 标签）来修改第三行带下划线的代码，并删除后面两个“action”标签；修改后的这三行带下划线的代码变为：

```
<package name="default" namespace="/" extends="struts-default">
    <action name="aCheck" class="AccountCheck">
        <result name="toIndex2">/index2.jsp</result>
    </action>
</package>
```

说明：

(1) “action”标签定义了一个动作，该动作由其 name 属性确定，并和页面表单中的“action=”属性相对应；class 属性则指定了该动作由哪个 Java 类来实现。例如，上面的代码相对应的页面表单应该有如下形式：

```
<form action="aCheck">...</form>
```

则上述 form 提交时，系统将寻找 AccountCheck 类并执行之。

(2) “result”标签定义了在动作执行之后的跳转（导航），其中 name 属性和动作中 execute()函数的返回值相应，也就是说，如果动作中 execute()返回值为字符串“toIndex2”，则系统跳转到 index2.jsp 页面。

(3) package 是对 action 分类的标签，其最核心的属性是“name”，该 name 是与其他 package 区分的依据；而 namespace（命名空间）属性则定义了到哪个地址寻找其下属的 action；例如，namespace="/example" 表示，如果请求的动作的 URL 是

```
/example/HelloWorld.action
```

的话，系统首先到/example 命名空间寻找“HelloWorld”动作对应的 Java 类；如果没有找到，系统还会到默认的命名空间去寻找，以及到“/”根命名空间寻找。对于我们修改后的例子，我们只是简单的定义了一个名为“default”的 package 并指定根命名空间。需要注意的是，命名空间只有“一层”，而不是像文件目录那样可以有多个层次。另注意：URL 中“xxx.action”和简化形式的“xxx”是等同的。

5. 实现最简单的 Struts2 系统

对于上面设计好的简单 Struts2 系统，我们可以根据 struts.xml 的要求按下面步骤实现之——注意这里体现的软件工程思想：先设计，后实现：

(1) 页面表单编码：Web 工程默认的“入口”页面是 index.jsp，位于 WebRoot（相当于网站的根目录）下。最简单的支持 Struts2 框架的 JSP 文件结构为：

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<body>
</body>
</html>
```

也就是说，最上面两行最关紧要！使用上面的模板替换系统中 index.jsp 原来的内容，并在<body>标签内加入 struts 表单：

```
<s:form action="aCheck">
    <s:textfield name="name" value="wang"></s:textfield>
    <s:textfield name="password" value="123"></s:textfield>
    <s:submit></s:submit>
</s:form>
```

其中“s:”是由上面所示第二行代码

```
<%@ taglib prefix="s" uri="/struts-tags" %>
```

规定的 struts 标签的前缀，也就是说，所有 struts 标签都要以“s:”打头。“form”和“submit”标签和 HTML 表单中类似的标签的意义相同，而“textfield”标签相当于 HTML 表单的 input 标签。请注意下划线的部分：action 属性定义的名称"aCheck"和 struts.xml 中的 action name 是一致的；两个 textfield 定义了两个表单字段，一个名为“name”，另一个名为“password”。用户在这两个字段中输入的数据将被 Struts 传递到后台 action 程序中。

小贴士：MyEclipse7.5 中有一个 Web Page Editor，用这个编辑器可以插入 struts2 的标签！方法是在 JSP 文件上点击鼠标右键，选择“Open with”，并从弹出的窗口中找到 Web Page Editor 并确定即可。展开右边工具栏中“Struts Tags”即可使用该工具栏添加 Struts 标签了，可以节省不少键盘输入的时间（见图 3）。

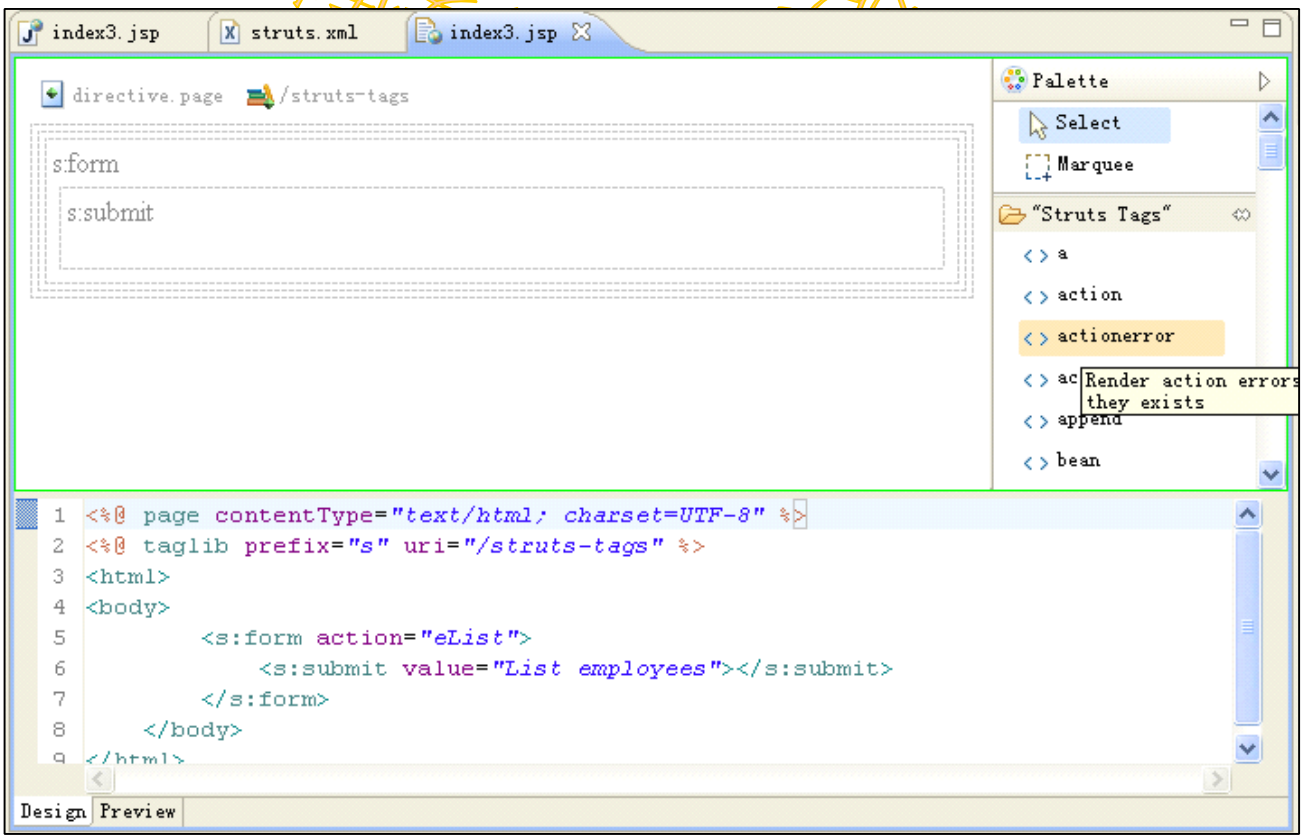

图 3 Web Page Editor 的编辑界面

(2) 动作编程：根据 struts.xml 要求，我们要编写一个名为“AccountCheck”的类来实现 action“aCheck”。这个类有 4 个要求：

a) 继承 ActionSupport；
b) 包含表单字段对应的属性变量；
c) 对上面的属性变量编写 getter 和 setter 方法；

d) 重写 execute()方法，并返回与 struts.xml 对应的字符串以进行导航。

AccountCheck.java 的代码如下：

```
import com.opensymphony.xwork2.ActionSupport;
public class AccountCheck extends ActionSupport{
    private String name;
    private String password;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String execute(){
        System.out.println(name+password);
        return "toIndex2";
    }
}
```

上文中的核心代码只有 4 行（使用下划线标出），前两行只是声明了与表单字段相对应的变量（Struts 会自动在表单和 action 类之间进行数据传递）；第三行将传递过来的字段值打印出来（一个简单的数据处理）；最后一行返回字符串“toIndex2”以指示系统导航到/index2.jsp 页面。注意其中的 setter 和 getter 函数分别用于设置和获取字段变量的值；但是我们在上面的代码中并没有看到对 getter 和 setter 的引用，而其实这正是 Struts 的**奇妙**之处：当 execute()被执行时，所有的字段变量已经在 Struts 的控制调度下被设置好了！

(3) 结果页面编程：根据 struts.xml 的要求，我们的结果页面是 index2.jsp；目前我们暂不需要在该页面中工作，因此您可以随便写个页面并命名为 index2.jsp 即可。

现在我们可以运行我们的系统了。由于我们的工程是 Web Project，因此只要在工程名上点鼠标右键，并选择

```
“Run As”—》“MyEclipse Server Application”
```

MyEclipse 会自动配置运行参数，启动内置的 Tomcat 应用服务器，打开浏览器并提供 index.jsp 页面。页面地址形如：

```
http://computername:8080/WarpSpeed/index.jsp
```

其中 computername 是您的计算机的名称。我们可以输入一些内容，点击“Submit”按钮，系统将自动转到 index2.jsp，并在控制台打印出来您输入的内容（由 execute()函数中的 println()实现）。

在这一阶段常见的错误是

```
HTTP Status 404 - There is no Action mapped for namespace / and action name xxx
```

该错误意味着在你指定的命名空间没有找到名为 xxx 的动作（同时在“根”命名空间也没有找到）。你需要查看并修改你的 xtruts.xml 配置文件。

6. 莫名其妙的错误？

MyEclipse 首次启动的时候，会要求你建一个“工作区（workspace）”，后继的工程文件均存放到这个工作区文件夹中。如果在你的学习过程中，编写了多个工程，有时候会出现莫名其妙的错误，那么除了认真阅读错误提示并进行更正意外，有些错误可能是你前期的工程遗留下来的，即使你早已删除了那些工程，其发布到 tomcat 中的应用**并没有**移除干净！

解决的方法很简单：

退出 MyEclipse，到 workspace 目录，在.metadata\.me_tcat\webapps 中有你已经发布所有工程的文件夹，找到原来的工

程文件夹，删除那些有问题的工程（或者干脆全部删除，因为这并不影响你的源文件），然后重新启动 MyEclispse 就可以了。
因此，如果你怕麻烦，最好一个工程使用一个独立的工作区。

7. 结果页面编程

从上面的实验当中可以看到，Java 的打印语句只能在服务器的控制台上输出结果，而不能将结果显示到用户的浏览器中。Struts 将 action 处理的数据（包括表单提交给 action 的数据）和应用（Application）范围内的数据，都存储在一个叫做“值栈”（valueStack）的地方；有很多种方法可以从这里取出值来。现在我们重新对 index2.jsp 编程，已将这些值取出到页面中：
首先，使用通用的 struts 页面模板替换 index2.jsp 的内容，并在<body>标签内加上：

```
<s:text name="password"/><hr/>
<s:property value="password" /><hr/>
${password}<hr/>
```

运行一下试试。可以看到，三种方法都可以将输入的 password 显示出来。其中前两种为 Struts 标签形式，${password}的写法是 OGNL 表达式语言，其用法暂略。

8. 结果页面导航

如果我们需要根据后台程序的处理结果转向不同的页面的话，必须同时配置 struts.xml 和 action。例如，如果要在传递过来的 name 和 password 参数的值分别为“wang”和“123”时，系统导航转向 index2.jsp 页面，否则的话转向验证失败页面 fail.jsp，则我们可以先配置 struts.xml 如下：

```
<action name="aCheck" class="AccountCheck">
    <result name="toIndex2">/index2.jsp</result>
    <result name="toFail">/fail.jsp</result>
</action>
```

根据这一配置，我们需要在 action 的 execute()方法作相应修改：

```
public String execute() {
    if (name.equals("wang") && password.equals("123"))
        return "toIndex2";
    else
        return "toFail";
}
```

最后，需要实现 fail.jsp；由于这个页面比较简单，只需要将 index2.jsp 另存为 fail.jsp，并将其页面内容修改为“失败”信息即可（略）。
再次运行工程，则输入数据不同会导航到不同的页面。至此，我们已经完整的体验了 Struts2 简单而强大的前后台交互及表现能力。下面我们看看如何方便地与底层数据处理进行交互。

9. 为工程添加 Spring 支持

在工程名上点鼠标右键并选择

```
“MyEclipse”—》“Add Spring Capabilities”
```

为简单起见，因为系统内置的 Spring 支持是 2.5 版本的，我们选中所有 2.5 版的 Spring 包，并选择“拷贝支持包到 lib 目录”；另特别注意：

（1）将 Spring 的配置文件 applicationContext.xml 放到 WEB-INF/目录下（而不是缺省的 src 目录）；

（2）重要：web.xml 中增加一个 listener：ContextLoaderListener；

上述第（2）项使得我们可以方便的获取当前程序的运行上下文，从而得到 DAO 对象以操纵数据库。

10. 添加 Hiberbate 支持

和上面的步骤类似，我们为工程添加 Hibernate 支持。同样选择将 Hibernate 的支持包拷贝到 lib 目录下；在“配置文件”选项中，选择使用已有的 Spring 配置文件（即上一步中的 WEB-INF/applicationContext.xml）；然后，设定我们要使用的

数据库为 MyEclipse 自带的小型数据库 Derby（在选项窗口的“DB Driver”下拉菜单中选择）。在最后的选项窗口中，我们还要为 Hibernate 创建一个专用的包：点击“Java Package”右边的“New”按钮，在 src 目录下面输入你的包名（例如 hib）即可；这个包是你以后放置 Hibernate 有关文件的包。

附注：Derby 数据库是 Apache 的开源小型数据库，具有体积小，易安装，支持 Java、JDBC 和 SQL 标准，并可用于嵌入式环境等特点，参见 http://db.apache.org/derby/。

11. 启动 Derby 数据库

为了在后继的工作中使用数据库，我们需要先启动 MyEclipse 内置的 Derby 数据库，方法是：

在 MyEclipse 的右上角，有一个带加号的窗口的小按钮，名为“Open Perspective”（鼠标悬停在该按钮上方的时候会显示它的名称）；点击该按钮并选择“MyEclipse Database Explorer”（或者使用“Window”菜单的“Show View”—》“DB Browser”）即可打开数据库浏览窗口；双击其中的“MyEclipse Derby”即可启动数据库。

Derby 启动以后，展开已经启动的数据库连接，可以看到有多个数据库，包括“APP”，“CLASSICCARS”等。展开“CLASSICCARS”下面的“TABLE”，会看到该数据库下属的多个表，例如“CUSTOMER”、“EMPLOYEE”等（见图 4），展开后者可以观察其包含的字段。如果你希望编辑该表中的数据，可以在表名上点击鼠标右键并选择“Edit Data”。可以看到，在“EMPLOYEE”表中，已经有数十条数据；您可以试着修改一下这些数据，我们下面的工作就以这些数据为基础。

然后使用屏幕左下角的按钮回到“Package Explorer”视图中。

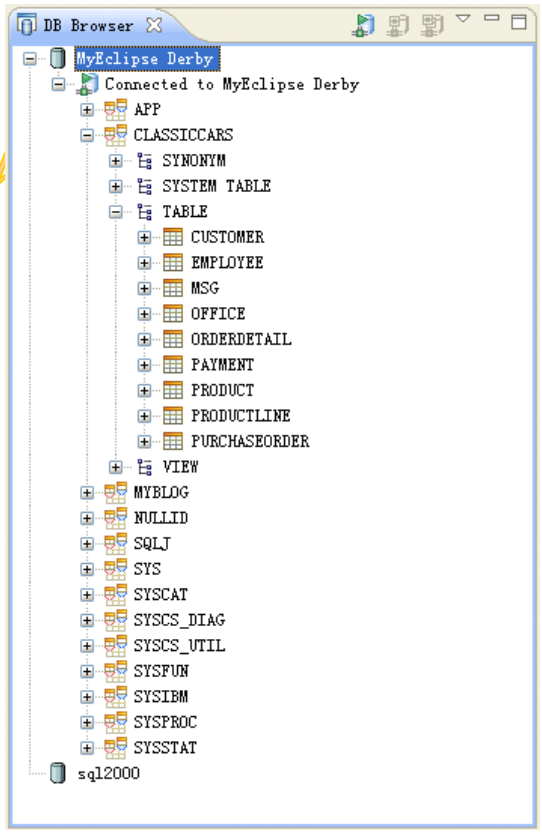

图 4 启动内置的 Derby 数据库

12. 更为奇怪的错误

直到目前为止，我们还没有进行任何涉及底层数据库操作的编码工作，仅仅是为我们的 WarpSpeed 工程添加了 Spring 和 Hibernate 的支持而已。现在我们看看在新的支持下，原来的页面还能正常运行否：

在工程名上点击鼠标右键并选择

“Run As”—》“MyEclipse Server Application”

——竟然是“404”（资源无法找到）错误！

您可以仔细研究 Console 控制台中显示的异常信息，但是，我可以告诉你一个简单得多的经验：这个错误本来不应该发生！出错的原因是你添加的各种支持里面所包含的 JAR 包有重复的，并且版本不一致！

展开“Package Explorer”中的“Referenced Libraries”，可以看到工程中所引用的所有 JAR 包；现在知道为什么在前面

的步骤中要将支持的 JAR 包拷贝到同一个 lib 目录下面了吧：起码我们检查重复 JAR 包的时候方便多了。
仔细检查后可以发现，“asm-2.2.3.jar”和“asm.jar”以及“cglib-nodep-2.1.3.jar”和“cglib-2.1.3.jar”是重复的；我们要删除这两对中的后者，方法是：在要删除的包上面点击鼠标右键并选择

“Build Path”—》“Remove From Build Path”

然后就可以删除了。
再运行一下试试，如果还不行，就要按照上面第“6”步所说的方法，彻底地清空一下已经发布的文件了。

13. 设计一个最简单的 Table 记录显示系统

目的：从“CLASSICCARS.EMPLOYEE”表中读出所有数据，并显示在页面中。
流程：
(1) 在 index3.jsp 中建立一个表单，对应的动作是“eList”；
(2) “eList”对应的 Java 类命名为“EmployeeList”，它在 Struts 的调控下获取数据表中的记录；
(3) “eList”执行完毕后转向 elist.jsp，在该页面上显示动作类获取的数据。

14. 实现最简单的 Table 记录显示系统

根据上述设计，首先修改 struts.xml 文件，添加如下的动作配置：

```
<action name="eList" class="EmployeeList">
    <result name="toList">/elist.jsp</result>
    <result name="toFail">/fail.jsp</result>
</action>
```

然后依次编码所需要的各个文件：
首先是起始页面 index3.jsp：

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<body>
    <s:form action="eList">
        <s:submit value="List employees"></s:submit>
    </s:form>
</body>
</html>
```

这里面实际上只有一个 submit 按钮。
然后是动作类 EmployeeList.java；因为该类需要读取数据表“CLASSICCARS.EMPLOYEE”，因此在编写该类之前需要先对所涉及的表进行所谓“**逆向工程**”以生成相关的实体类和 DAO（数据存取对象）类。方法是：
(1) 在 Database Explorer 中，找到数据表“CLASSICCARS.EMPLOYEE”并点击右键，选择“Hibernate Reverse Engineering”，设定 Java src folder（点击 Browse 选择/WarpSpeed/src）和 Java Package（还记得我们在第 10 步中创建的 hib 包吗？就是这个），勾选下面的三个复选框（其中“Create Abstract Class”不用选；这三个复选框的含义就是指定使用逆向工程生成数据表相关的实体类或说 POJO 类以及对应的 DAO 类）。
(2) 第二页中，比较重要的是确定 Id 生成策略，例如，如果你的主键是自增生成的 id，请使用 identity 策略。我们目前的例子暂时不需要设置这项，直接点击“Next”到下一页。
(3) 第三页中，注意在表清单的下方，有两个复选框，“Include referenced tables”和“Include referencing tables”，意思是在逆向工程的时候要包含和当前表有关联的其他表。选中这两个框后我们可以看到，表清单中多了一个表名“OFFICE”，因为 EMPLOYEE 通过“OFFICECODE”字段作为外键和 OFFICE 表关联。
执行逆向工程后，在 Package Explorer 中可以看到 src/hib 包下面多了许多源文件，例如 Employee.java 是 EMPLOYEE 表对应的实体类，而 EmployeeDAO.java 则是其 DAO 类。

说明：DAO 类：

DAO 类中集成了对实体类的通用操作，例如 findAll()，findBy()，attachDirty()等；特别地，DAO 类包含一个静态方法用于从应用上下文中获取该 DAO 类的一个实例：getFromApplicationContext(ApplicationContext ctx)；这样只要我们获

取了当前应用的上下文（Context）后，即可取得 DAO 实例并通过该实例操作数据对象了（参见下面的步骤）。

数据处理 action 的“三板斧”为：

(1) 获取当前应用上下文；

(2) 获取实体类的 DAO 对象实例；

(3) 使用 DAO 对象操作数据；

特别提醒：在使用 DAO 对象操作数据时，经常使用 Java 的集合类对象，例如 List 等。

EmployeeList.java 核心代码为：

```
public class EmployeeList extends ActionSupport {
    public String execute() {
        ApplicationContext ct = WebApplicationContextUtils
                .getWebApplicationContext(ServletActionContext
                        .getServletContext());

        EmployeeDAO dao = EmployeeDAO.getFromApplicationContext(ct);
        List<Employee> list = dao.findAll();
        int n = list.size();
        for (int i = 0; i < n; i++) {
            Employee c = list.get(i);
            String name = c.getFirstname();
            String lastname = c.getLastname();
            System.out.println(name+"."+lastname);
        }
        return "toList";
    }
}
```

我们暂且不实现 elist.jsp，先测试一下已经实现的部分：

运行工程；系统自动打开一个浏览器，其地址栏中显示的是工程的 Web 主目录，形如：

```
http://computername:8080/WarpSpeed/
```

现在，在该地址后面加上“index3.jsp”并回车，即可显示该起始页面。点击“List Employee”按钮，则浏览器提示“404”无法找到资源错误（暂且不用管它，我们下面再来解决）；但是你可以看到，控制台中已经打印出来所有的员工名单了。

15. 在结果页面上显示记录

上面的实现还有缺陷：我们并没有将 action 获取的数据带到结果页面上。要做到这一点，必须进行两项工作：

(1) 在 action 中将需要传递给结果页面的数据声明为类属性并为其编写 setter 和 getter 函数。

(2) 在页面中使用 Struts 标签取出 action 传递过来的数据。

因此我们首先将 EmployeeList.java 做如下修改：将 list 声明为类属性，并为其 list 添加 getter 和 setter：

```
List<Employee> list;
public List<Employee> getList() {
    return list;
}
public void setList(List<Employee> list) {
    this.list = list;
}
```

注意 execute()中对 list 引用的时候就不需要重复声明了。

然后编写 elist.jsp：

```
<s:iterator value="list" >
    Firstname: <s:property value="firstname" /><br/>
    Lastname: <s:property value="lastname" /><br/>
```

```
        Email: ${email}<br/>
        Jobtitle: <s:text name="jobtitle"/><p>
    </s:iterator>
```

这里面，我们尝试了几种不同的表现方法。

首先看<s:iterator>标签：它的作用是遍历由 value 指定的“集合型”对象（一般为 Map 或 List 类型）；在我们的例子中，我们要遍历的是“list”对象（该对象由 action 传递过来，对应于 action 内的一个类属性）。

其次要注意<s:property>标签，它的作用是将 iterator 中的对象的属性取出来，即<s:property value="firstname" />的作用相当于调用 list 的 getFirstname()方法（action 中的 getter 在这里得到了“隐形的”调用）。

从上面的代码中可以看出，除了使用<s:property>之外，我们还尝试了另外两种标签方法，一是<s:text>，它可以生成一个国际化的信息文本；另一个是${email}，用的是 OGNL 表达式语言。网上有争论说，既然是 struts2，就尽量用 struts2 的标签；也有人说：OGNL 是主流，应该和<s:>标签配合使用；请自行深入学习后选择。

16. 向数据表中插入新记录

在讲解这个例子之前，先介绍一下 Struts2 的新特性：Struts2 表单的对象注入；也就是说，Struts2 在表单级即支持对象的概念，例如，假设我们要在页面 index4.jsp 的表单中输入要插入到 EMPLOYEE 表中的数据，可以使用如下形式：

```
    <s:form action="insert" method="post">
        employeenumber<s:textfield name="em.employeenumber"></s:textfield>
        lastname<s:textfield name="em.lastname"></s:textfield>
        firstname<s:textfield name="em.firstname"></s:textfield>
        extension<s:textfield name="em.extension"></s:textfield>
        email<s:textfield name="em.email"></s:textfield>
        officecode<s:textfield name="em.officecode"></s:textfield>
        reportsto<s:textfield name="em.reportsto"></s:textfield>
        jobtitle<s:textfield name="em.jobtitle"></s:textfield>
        <s:submit></s:submit>
    </s:form>
```

注意各个字段都使用了复合的形式，其第一部分为要作为整体传递到 action 的对象的名称，第二部分为该对象的属性（字段）名。自然地，其相对应的 action 类中，要有对“em”属性的声明和处理。请看我们的 InsertEmployee.java 的主要代码：

```
    Employee em;
    public String execute() {
        ApplicationContext ct = WebApplicationContextUtils
                .getWebApplicationContext(ServletActionContext
                        .getServletContext());
        EmployeeDAO dao = EmployeeDAO.getFromApplicationContext(ct);
        Session s = dao.getSessionFactory().openSession();
        Transaction tx = s.beginTransaction();
        try {
            dao.attachDirty(em);
            tx.commit();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return "toList";
    }
```

别忘了在上面代码中还需要为 em 属性生成 getter 和 setter 函数。

从上面的例子中还可以学到**数据操纵的四板斧**：

(1) 依次获取应用上下文、DAO 对象和当前会话对象；

(2) 启动事务；
(3) 数据操纵；
(4) 提交事务；
和上面程序相对应的，需要在 struts.xml 中添加如下的 action 定义：

```
<action name="insert" class="InsertEmployee">
    <result name="toList">/elist.jsp</result>
</action>
```

您可以重新部署系统并访问 index4.jsp 来测试程序运行结果。
想一想：如果你想把出错时的 Exception 信息显示到一个单独的结果页面上，应该怎么做？
提示：不仅要配置独立的出错页面，还要注意：“出错信息”也应该声明成一个“类属性”才能传递到结果页面上。

17. 自定义页面格式

页面 index4.jsp 的运行结果没有问题，但是其显示格式却有些古怪：所有的字段标签并没有象我们预期的那样和文本框一一对应，而是统统放到了页面最上端。您可以在 IE 浏览器中重新打开这个页面并观察其 HTML 源码，就会发现里面除了由 Struts 标签生成的表单代码外，还有许多类似<table>这样的表格标签等。这些标签是 Struts 为美化表单的排版而自动加上的。如果你想自己控制表单的排版，可以抑制 Struts 的这个功能，方法是在表单中指定“simple”主题：

```
<s:form action="insert" method="post" theme="simple">
```

这样，生成的表单的排版格式就完全由你控制了。

18. 为虎添翼的小提示：

Tips 1：从一个 action 跳转到另一个 action：
在 struts.xml 中可以这样配置：

```
<result name=”xxx” type=”redirect”>action name</result>
```

这种方法是直接跳转，不传递上一个 action 中的属性数据。

```
<result name=” xxx”  type=” chain” >action name</result>
```

这种方法可以将上一个 action 中的属性数据带上。
注意：action name 需要直接写名字，不能带路径，也不能带.action 这样的后缀。
Tips 2：输入数据校验：
最简单的输入数据校验方法，是重写 validate()方法（当然，你的 action 需要继承 Struts 的 ActionSupport）。
例如，在页面中，使用如下表单：

```
<s:textfield name="u.name" label="用户名"></s:textfield>
<s:textfield name="u.password" label="密码"></s:textfield>
```

那么，如果我们想捕捉没有输入密码的错误，则在 action 中加入这样的语句即可：

```
public void validate(){
    if(u.getPassword().length()==0){
        addFieldError("u.password", "你忘了密码了！ ");
    }
}
```

需要注意，你必须在 struts.xml 中定义 input 页面（即错误输出页面）。例如，如果你想在输入页面的本身显示错误信息的话，可以这样写：假设你的表单在 myform.jsp，那么 struts.xml 中应包含这样的语句：

```
<result name="input">myform.jsp</result>
```

如果你希望将错误输出定向到别的页面，除了要修改上面的配置语句外，还需要在错误输出页面内加上

```
<s:fielderror/>
```

这个标签是“表单字段输入错误”标签，它负责显示字段的校验错误信息，也就是你在 action 中使用 addFieldError()函数要显示的信息。当然，如果你使用了别的错误信息函数，例如，addActionError()或者 addActionMessage()，则需要在错误输出页面加上相应的显示标签，例如<s:actionerror/>和<s:actionmessage/>。
Tips 3：关于自动提交（auto commit）：
在 Hibernate 配置文件（applicationContext.xml）中的数据源（DataSource）的配置中加入自动提交的设置，如下

```
<property name="defaultAutoCommit" value="true"></property>
```

即可不需要在 action 中加入 Session，Transaction 的语句，就可以将数据直接更新到数据库。

19. 小结：

在这个短短的教程中，我们学习使用 Struts 处理用户输入，并通过实体类存取数据库；数据可以在前台（页面）和后台（Java 类）之间方便以面向对象的方式传递；Java 程序员可以专注于后台的 Java 编程，前后台分工更加合乎软件工程思想，从而为构建复杂系统提供有力支持。本文中的样例工程 WarpSpeed 的目录打包后存放在文件 WarpSpeed.rar 中，您可以从本刊网站下载。