Web程序从Struts向Stripes框架的移植 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/491/2021_2022_Web_E7_A8_8B_E5_BA_8F_E4_c67_491716.htm 摘要 把你的现有Struts应用程序移植到Stripes框架能够简化Web开发，并且这一移植过程要比你想象的更为容易。 一、 引言 把一个现有Java Web应用程序移植到一种新框架可能不是大多数开发者最感兴趣的问题。除了要花费时间学习一种新的Web框架外，例如标签、国际化系统和校验等繁重的转化过程可能会迫使每一位程序员考虑再三。我最近就面临这样的一个挑战-从Struts进行移植。 在决定移植一个应用程序前，应该首先问一下"为何不使用现在的框架？"在我看来，Struts是一种稳定的具有良好文档的框架，并且有一大批开发者社区成员，但是其配置很麻烦，而且其表单、行为、应用程序流和校验的分离有时会带来很多麻烦。这种情形在我的Struts应用程序不断变大时越发糟糕。最后，纯粹从一种维护的角度，我决定把它移植到一种新的框架。 开始，我认为没有一种框架（Java ServerFaces，Tapestry，WebWorks，Spring MVC）值得从Struts迁移向其迁移。例如JSF这样的框架看上去极不友好。其它的，例如Tapestry和WebWorks，涉及到整页整页的看上去令人麻烦的国际化系统。而从配置角度来看，Spring MVC看上去并不比Struts好多少。我选择的框架应该仅需适当的学习时间，还要与移植效益相称；而且，它还一定要使我编码、排错与维护更为容易。 二、 发现Stripes框架 后来，我偶然发现了Stripes框架。就象Java社区中的许多发烧友一样，我一直追随着Ruby on Rails（RoR）现象。依我看来，Stripes是最接近

于RoR哲学的Java MVC框架-简单，漂亮，并且要求最小的配置。除了它的简洁外，象我这样一位Struts程序员，Stripes非常适合我的口味。应用程序流和许多命名惯例都与之十分相似。Stripes中的ActionBeans就象Strut的Actions，而ForwardResolutions极象ActionForwards.因此，使用这一框架，我不必抛弃我所有以前的Struts知识。 另外吸引我的是Stripes文档。象框架本身一样，文档也是干净、清洁而简练。其标签库文档和API都具有良好的归档，而且该框架的每一种特征几乎都有相应的示例源码。这些优秀的文档再加上我的现有Struts知识使我坚信，我可以快速地掌握这种Stripes框架。 值得注意的是，Stripes还包括另外一些使其成为一种良好的AJAX平台的特征，例如它提供了一种流式方案，该方案允许对AJAX实现进行改进的错误处理。然而，对于我来说，最终的决定因素还是我能够清楚地看到它会使我的生活更容易些。我估计，在我的应用程序的行为/配置/校验部分，我只需使用约一半的代码就够了。更少的代码意味着了更少的错误、更快的开发时间和更容易的纠错。 三、 移植过程 我从视图层开始移植，然后再向行为层移植。事实上，我也没有很明确的逻辑思路；只是必须从某处开始，而视图部分看起来更适合于作为一个起始点。 （一） JavaServer Pages 就象Struts一样，Stripes使用JSP来实现其视图层。我吃惊地发现，Stripes标签库非常类似于Struts的HTML taglib.事实上，我能够使用这种统一替换方式来升级我的许多标签。 Stripes依赖于JSTL实现JSP视图中的逻辑。我在我的应用程序中混合使用了Struts逻辑标签和JSTL.通过把我的所有逻辑标签移植到JSTL，我能够利用JSTL的优越的if/else和case语句的能力处理，它

们可能是很原始的或者根本不存在于Struts逻辑taglib中。 （二） 国际化 接下来，我要移植我的Struts的消息资源。在配置端，所有要求的操作就是重命名我的Struts消息资源文件。在我的JSP中，我能够使用统一替换方式把我的所有Struts message标签（例如，＜bean：message key="buttons.save"/＞）替换为JSTL格式标签（例如，＜fmt：message key="buttons.save"/＞）。这种JSTL格式标签还支持可用于Struts中的消息资源绑定。 （三） 表单 我的移植的最有意义的部分是去掉了我的Struts Action表单，这是在Action类中进行的，要求大量的XML标记和冗长的转换，如下例所示：＜form-bean name="employeeUpdateForm" type="org.apache.struts.validator.DynaValidatorForm"＞＜form-property name="employeeid" type="java.lang.Long" /＞＜form-property name="firstname" type="java.lang.String" /＞＜form-property name="lastname" type="java.lang.String" /＞＜form-property name="phone" type="java.lang.String" /＞＜form-property name="email" type="java.lang.String" /＞＜form-property name="phone" type="java.lang.String" /＞＜form-property name="socialsecurity" type="java.lang.String" /＞＜form-property name="birthdate" type="java.lang.String" /＞＜form-property name="salary " type="java.lang.String" /＞＜/form-bean＞ public ActionForward executeAction(ActionMapping mapping， ActionForm form，HttpServletRequest request， HttpServletResponse response) throws Exception{Employee employee=new Employee().DynaValidatorForm eaf = (DynaValidatorForm) form.employee.setFirstname(eaf.getString("firstname")).employee.se

tLastname(eaf.getString("lastname")).employee.setPhone (eaf.getString("phone")).employee.setEmail (eaf.getString("email")).employee.setEmployeeid ((Integer)eaf.get("employeeid")).employee.setSocialsecurity(Long.parseLong(eaf.getString("socialsecurity"))).employee.setBirthdate(MyUtils.convertStringToDate(eaf.getString ("birthdate"))).employee.setSalary(MyUtils.convertStringToBigDecimal(eaf.getString ("salary"))).EmployeeDAOService.0updateEmployee(employee).return new ActionForward(mapping.getForward()).} 另一方面，Stripes表单处理允许你把你的域对象用作一个表单使用：public class UpdateEmployeeActionBean implements ActionBean {private ActionBeanContext context.private Employee employee.public ActionBeanContext getContext() {return context.}public void setContext(ActionBeanContext context) {this.context = context.}public void setEmployee(Employee employee) {this.employee = employee.}public Employee getEmployee() {return this.employee.}@DefaultHandlerpublic Resolution 0update() {EmployeeDAOService.0updateEmployee(employee).return new ForwardResolution("/employees/0updateEmployee.jsp").}} 在大多数情况中，我能够嵌入我的域对象的一个副本作为我的Stripes ActionBean类的一个属性并且仅包括涉及把该对象移入/移出我的持久层的代码部分。我把所有表单处理都交给了Struts Action来实现，包括初始化配置、把表单强制转换成适当的类以及从域对象中来回转换数据（约30%的代码是在大多

数Action类中实现的）。在Stripes中并不需要这样。 简言之，把域对象作为你的ActionBean类的一个属性嵌入，为该类提供了getter和setter方法。总之，所有有趣的地方（包括列表）都在HTML视图的表单中体现出来。 所有我对表单的操作也都可以通过查询串参数方式来实现。我仅把这些参数作为我的ActionBean的一个属性，而如果它们是请求的一部分的话，可以把它们自动地复制到相应的域中。 （四） 校验 与表单或标签移植相比，把Struts校验移植到Stripes要求更多的工作。在我的应用程序中，我必须在Stripes ActionBean类内部使用Java 5.0注解重写在validation.xml文件中的校验配置。Stripes还为你提供一种良好的基于类型的校验。当用户输入错误值时，不需要用户进行任何配置，Stripes就可以把HTML表单返回给他们（例如，在一个数字或日期域中的字符）。表单能够被自动返回并带有一条向用户友好显示的消息，最后出错域被高亮显示。 （五） 应用程序流程 转换我的Struts应用程序的控制流可能是唯一远离Struts思维的一个地方。在Struts中，控制流（URL请求绑定、行为和结果视图）都以XML标记形式生成并且被集中放到struts-config.xml文件中。在行为层外进行生成使Struts绑定更为灵活。它们没有被硬编码到行为层中，而单个行为可以容易地与不同的输入URL和转发进行耦合。这种方式的不好的地方在于，Struts配置量可能会急剧增加而成为麻烦。控制流与行为层的分离还会使在整个请求周期中的调试相当困难。 为此，Stripes共提供了三种不同方式以便把请求映射到行为层： 1. 使用注解把一个ActionBean显式绑定到一个URL； 2. 允许Stripes在启动期间基于ActionBean类路径和应用程序URL之间的相似性猜测它的ActionBean的绑

定；3. 类路径通过使用Stripes useBean标签，把一个JSP绑定到任何ActionBean，或调用应用程序中一个Java类的任何方法。尽管与Struts配置相比，前两种方法似乎有点"硬编码"特征，但是useBean标签提供了大量的灵活性。借助于该标签，JSP可以存取多个ActionBean或类以得到其所需要的内容。 四、 结论 当选择一个新框架时，迁移的容易性（既包括学习新框架方面，也包括移植你的现有代码方面）是要考虑的要素之一，但是不应该过多地强调。是的，你可能已经在学习一种现有框架上做出很大的投资并且在你的下一个MVC平台上保留这些投资的一部分更好一些。而且，如果你能够在几周而不是在几个月内移植完你的应用程序则最好不过。但是不管问题是多么容易或是多么愉快，你还是要首先应该决定是否目标能够满足你的真正要求。对于我来说，能够把几乎一半的代码放到我的行为层中而把表单、配置和校验放到一起是我最关心的问题。Stripe文档的质量及其它问题则为次要。