用Struts的Token机制解决表单重复提交

Struts的Token（令牌）机制能够很好的解决表单重复提交的问题，基本原理是：服务器端在处理到达的请求之前，会将请求中包含的令牌值与保存在当前用户会话中的令牌值进行比较，看是否匹配。在处理完该请求后，且在答复发送给客户端之前，将会产生一个新的令牌，该令牌除传给客户端以外，也会将用户会话中保存的旧的令牌进行替换。这样如果用户回退到刚才的提交页面并再次提交的话，客户端传过来的令牌就和服务器端的令牌不一致，从而有效地防止了重复提交的发生。

　　这时其实也就是两点，第一：你需要在请求中有这个令牌值，请求中的令牌值如何保存，其实就和我们平时在页面中保存一些信息是一样的，通过隐藏字段来保存，保存的形式如： 〈input type="hidden" name="org.apache.struts.taglib.html.TOKEN" value="6aa35341f25184fd996c4c918255c3ae"〉，这个value是TokenProcessor类中的generateToken()获得的，是根据当前用户的session id和当前时间的long值来计算的。第二：在客户端提交后，我们要根据判断在请求中包含的值是否和服务器的令牌一致，因为服务器每次提交都会生成新的Token，所以，如果是重复提交，客户端的Token值和服务器端的Token值就会不一致。下面就以在数据库中插入一条数据来说明如何防止重复提交。

　　在Action中的add方法中，我们需要将Token值明确的要求保存在页面中，只需增加一条语句：saveToken(request);，如下所示：

public ActionForward add(ActionMapping mapping, ActionForm form,

　　HttpServletRequest request, HttpServletResponse response)

　　//前面的处理省略

　　saveToken(request);

　　return mapping.findForward("add");

　　}在Action的insert方法中，我们根据表单中的Token值与服务器端的Token值比较，如下所示：

　　public ActionForward insert(ActionMapping mapping, ActionForm form,

　　HttpServletRequest request, HttpServletResponse response)

　　if (isTokenValid(request, true)) {

　　// 表单不是重复提交

　　//这里是保存数据的代码

　　} else {

　　//表单重复提交

　　saveToken(request);

　　//其它的处理代码

　　}

　　}

其实使用起来很简单，举个最简单、最需要使用这个的例子：

　　一般控制重复提交主要是用在对数据库操作的控制上，比如插入、更新、删除等，由于更新、删除一般都是通过id来操作（例如：updateXXXById, removeXXXById），所以这类操作控制的意义不是很大（不排除个别现象），重复提交的控制也就主要是在插入时的控制了。

　　先说一下，我们目前所做项目的情况：

　　目前的项目是用Struts＋Spring＋Ibatis，页面用jstl，Struts复杂View层，Spring在Service层提供事务控制，Ibatis是用来代替JDBC，所有页面的访问都不是直接访问jsp，而是访问Structs的Action，再由Action来Forward到一个Jsp，所有针对数据库的操作，比如取数据或修改数据，都是在Action里面完成，所有的Action一般都继承BaseDispatchAction，这个是自己建立的类，目的是为所有的Action做一些统一的控制，在Struts层，对于一个功能，我们一般分为两个Action，一个Action里的功能是不需要调用Struts的验证功能的（常见的方法名称有add,edit,remove,view,list），另一个是需要调用Struts的验证功能的（常见的方法名称有insert,update）。

　　就拿论坛发贴来说吧，论坛发贴首先需要跳转到一个页面，你可以填写帖子的主题和内容，填写完后，单击“提交”，贴子就发表了，所以这里经过两个步骤：

　　1、转到一个新增的页面，在Action里我们一般称为add，例如：

public ActionForward add(ActionMapping mapping, ActionForm form,

　　HttpServletRequest request, HttpServletResponse response)

　　throws Exception {

　　//这一句是输出调试信息，表示代码执行到这一段了

　　log.debug(":: action - subject add");

　　//your code here

　　//这里保存Token值

　　saveToken(request);

　　//跳转到add页面，在Structs-config.xml里面定义，例如，跳转到subjectAdd.jsp

　　return mapping.findForward("add");

　　}

2、在填写标题和内容后，选择 提交 ，会提交到insert方法，在insert方法里判断，是否重复提交了。

public ActionForward insert(ActionMapping mapping, ActionForm form,

　　HttpServletRequest request, HttpServletResponse response){

　　if (isTokenValid(request, true)) {

　　// 表单不是重复提交

　　//这里是保存数据的代码

　　} else {

　　//表单重复提交

　　saveToken(request);

　　//其它的处理代码

　　}

　　}

下面更详细一点（注意，下面所有的代码使用全角括号）：

　　1、你想发贴时，点击“我要发贴”链接的代码可以里这样的：

　　〈html:link action="subject.do?method=add"〉我要发贴〈/html:link〉

　　subject.do 和 method 这些在struct-config.xml如何定义我就不说了，点击链接后，会执行subject.do的add方法，代码如上面说的，跳转到subjectAdd.jsp页面。页面的代码大概如下：

　　〈html:form action="subjectForm.do?method=insert"〉

　　〈html:text property="title" /〉

　　〈html:textarea property="content" /〉

　　〈html:submit property="发表" /〉

　　〈html:reset property="重填" /〉

　　〈html:form〉如果你在add方法里加了“saveToken(request);”这一句，那在subjectAdd.jsp生成的页面上，会多一个隐藏字段，类似于这样〈input type="hidden" name="org.apache.struts.taglib.html.TOKEN" value="6aa35341f25184fd996c4c918255c3ae"〉，

　　2、点击发表后，表单提交到subjectForm.do里的insert方法后，你在insert方法里要将表单的数据插入到数据库中，如果没有进行重复提交的控制，那么每点击一次浏览器的刷新按钮，都会在数据库中插入一条相同的记录，增加下面的代码，你就可以控制用户的重复提交了。

if (isTokenValid(request, true)) {

　　// 表单不是重复提交

　　//这里是保存数据的代码

　　} else {

　　//表单重复提交

　　saveToken(request);

　　//其它的处理代码

　　}

注意，你必须在add方法里使用了saveToken(request)，你才能在insert里判断，否则，你每次保存操作都是重复提交。

　　记住一点，Struts在你每次访问Action的时候，都会产生一个令牌，保存在你的Session里面，如果你在Action里的函数里面，使用了saveToken(request);，那么这个令牌也会保存在这个Action所Forward到的jsp所生成的静态页面里。

　　如果你在你Action的方法里使用了isTokenValid，那么Struts会将你从你的request里面去获取这个令牌值，然后和Session里的令牌值做比较，如果两者相等，就不是重复提交，如果不相等，就是重复提交了。

　　由于我们项目的所有Action都是继承自BaseDispatchAction这个类，所以我们基本上都是在这个类里面做了表单重复提交的控制，默认是控制add方法和insert方法，如果需要控制其它的方法，就自己手动写上面这些代码，否则是不需要手写的，控制的代码如下：

public abstract class BaseDispatchAction extends BaseAction {

　　protected ActionForward perform(ActionMapping mapping, ActionForm form,

　　HttpServletRequest request, HttpServletResponse response)

　　throws Exception {

　　String parameter = mapping.getParameter();

　　String name = request.getParameter(parameter);

　　if (null == name) { //如果没有指定 method ，则默认为 list

　　name = "list";

　　}

　　if ("add".equals(name)) {

　　if ("add".equals(name)) {

　　saveToken(request);

　　}

　　} else if ("insert".equals(name)) {

　　if (!isTokenValid(request, true)) {

　　resetToken(request);

　　saveError(request, new ActionMessage("error.repeatSubmit"));

　　log.error("重复提交！");

　　return mapping.findForward("error");

　　}

　　}

　　return dispatchMethod2(mapping, form, request, response, name);

　　}

　　}