struts2中使用token避免重复提交

**1.在struts.xml中**　　<action name="register" class="org.sunxin.struts2.action.RegisterAction">  
　　<!-- 配置异常映射，当RegisterAction抛出Exception异常时，向用户显示error.jsp页面-->  
　　 <exception-mapping result="error" exception="java.lang.Exception"/>  
　　<result name="input">/pages/register.jsp</result>  
　　<result name="success">/pages/success.jsp</result>  
　　<result name="error">/pages/error.jsp</result>  
　　<result name="invalid.token">/pages/register.jsp</result>  
　　<result name="wait">wait.jsp</result>  
　　<interceptor-ref name="defaultStack"></interceptor-ref>  
　　<interceptor-ref name="token">  
　　<param name="excludeMethods">input</param>  
　　 </interceptor-ref>  
　　</action>  
　　**2.在页面中加**　　<s:actionerror/>  
　　<s:form action="register" method="post">  
　　<s:token></s:token>  
　　</s:form>  
　　3.<interceptor-ref name="token"/>  
　　<interceptor-ref name="token-session"/>  
　　<!--注意struts2.0 拦截器名字为token-session struts2.1.2 已经更改为tokenSession -->  
　　token: 在活动中检查合法令牌(token), 防止表单的重复提交; 在<s:actionerror/>会产生提示信息  
　　token-session: 同上, 但是在接到非法令牌时将提交的数据保存在session中; 不会在<s:actionerror/>会产生提示信息  
　　只会在后台发出警告并处理，如下：  
　　警告: Form token KO80SIJW4F84034NG5HM1ZBUGOVNY64D does not match the session token null.

　　编辑特别推荐:

为struts1.2增加token标签 令牌标签[修改更新 2009-07-28 修复了一个BUG]

2009-07-23 17:28

|  |
| --- |
| 2009-07-28更新提醒：修改了一个页面不能存在多个表单的BUG package com.tgmsp.CongZhong.tag;  import java.io.IOException;  import javax.servlet.http.HttpServletRequest; import javax.servlet.jsp.JspException; import javax.servlet.jsp.JspWriter; import javax.servlet.jsp.tagext.TagSupport;  import org.apache.struts.Globals; import org.apache.struts.taglib.html.Constants; import org.apache.struts.util.TokenProcessor;  public class TokenTag extends TagSupport {     public int doEndTag() throws JspException {         HttpServletRequest request = (HttpServletRequest)pageContext.getRequest();         JspWriter out = pageContext.getOut();         try {             Object tmp = null;             String token = (tmp = request.getSession().getAttribute(Globals.TRANSACTION\_TOKEN\_KEY))==null?TokenProcessor.getInstance().generateToken(request):tmp.toString();                          if (token != null) {                 request.getSession().setAttribute(Globals.TRANSACTION\_TOKEN\_KEY, token);                 out.println("<input type=\"hidden\" name=\""+Constants.TOKEN\_KEY+"\" value=\""+token+"\"/>");             }         } catch (IOException e) {             e.printStackTrace();         }         return this.EVAL\_PAGE;     } }  .... 标签配置部分 <tag>     <name>token</name>     <tagclass>com.tgmsp.CongZhong.tag.TokenTag</tagclass>     <bodycontent>empty</bodycontent>     <info>         <![CDATA[             保存Struts令牌             org.apache.struts.action.TOKEN             <congzhong:token/>         ]]>     </info> </tag> .... |

**实例一：防止表单重复提交**

防止重复提交：有两种方法：一是使用token拦截器，二是使用tokenSession

第一种方法步骤：

1、  先在页面中放一个令牌。调用Struts标签，服务器自动分派。<s:token></s:token>

2、  在struts.xml文件定义token拦截器。

1. <interceptor-ref name="token"></interceptor-ref>
3. <interceptor-ref name="defaultStack"></interceptor-ref>

3、  在struts.xml文件配置有重复提交的错误页面，即：

1. <result name="invalid.token">/repeatSubmitError.jsp</result>

  第二种方法基本与第一种方法基本相同，只不用配置重复提交的错误页面，如果发生重复提交，返回success。

**实例二：PreResultInteceptor**

这个功能是在Action返回到result时执行。

拦截器 --> Action --> Result --> 拦截器

实现步骤：

1、  定义一个类实现PreResultListener接口，实现beforeResult方法

1. public class TestPreResultInteceptor implements PreResultListener {
3. public void beforeResult(ActionInvocation arg0, String arg1) {
5. System.out.println("在结果集返回之前执行了beforeResult方法");
7. }
9. }

2、  自定义一个拦截器与上面定义类进行绑定。

1. public class TestInterceptor extends AbstractInterceptor {
3. @Override
5. public String intercept(ActionInvocation invocation) throws Exception {
7. invocation.addPreResultListener(new TestPreResultInteceptor());
9. System.out.println("在action执行execute方法之前执行进行拦截");
11. String result = invocation.invoke();
13. System.out.println("在action执行execute方法之后执行进行拦截");
15. return result;
17. }
19. }

很多时候要防止重复提交，比如论坛在发表提问的时候，要是不控制那就可以一直提交~

　　使用方法：论坛为例

　　ask.do ：为跳到提问页面的action

　　ask.jsp：为提问页面

　　askok.do:为提问成功提交的action

　　askok.jsp：处理提问之后的跳转的jsp

　　首先在ask这个action里面加入saveToken(request);这是为了在提问页面即ask.jsp页面生成一个隐藏的表单

　　<input type="hidden" name="org.apache.struts.taglib.html.TOKEN" value="a6b9ff6834284444001b8206723b654d">

　　在ask。jsp页面做以下处理：使用form为<html:form 　method="post" action="askok.do">　 因此要引入这个标签 否则不会生成这个隐藏表单。

　　form写法 1.1之后不写name 没有这个属性 但要在struts-config。xml里面配置这个actionform 我们可以定义为question　 同时要配置askok这个action里面要写入对应的form

　　<action path="/askok" type="....." name="question">必须写上这个否则会报找不到FORM的错误

　　如果要在ask。jsp这个页面做js处理要用到FORM name的话可以直接写 question 　 因为struts自动生成了这个form的name 生成之后的:<form name="bangquestion" method="post" action="/askok.do">

　　ask.jsp 这个页面自动生成加入FORM name

　　然后在askok.do里面写if (isTokenValid(ac.getRequest(),true)) {

添加提问  
  
}else  
  
{  
  
跳转提示  
  
}

**Struts1.2之Token重复提交**

[遇难集中营](http://zhym86848658.blog.163.com/blog/#m=0&t=1&c=fks_080068080084089067087085074064085081083066087087086) 2007-12-17 12:16:24 阅读130 评论0 字号：大中小

--------------------------------------------------------Struts-config.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "<http://struts.apache.org/dtds/struts-config_1_2.dtd>">

<struts-config>  
  <data-sources />  
  <form-beans >  
    <form-bean name="insertForm" type="org.zhym.struts.form.InsertForm" />

  </form-beans>

  <global-exceptions />  
  <global-forwards />  
  <action-mappings >  
    <action  
      attribute="insertForm"  
      input="/insert.jsp"  
      name="insertForm"  
      path="/insert"  
      scope="request"  
      type="org.zhym.struts.action.InsertAction" />  
    <action path="/perpareInsert" type="org.zhym.struts.action.PerpareInsertAction" >  
     <forward name="insert" path="/insert.jsp"></forward>  
    </action>

  </action-mappings>

  <message-resources parameter="org.zhym.struts.ApplicationResources" />  
</struts-config>

-------------------------------------------------------------------------index.jsp

<%@ page language="java" import="java.util.\*" pageEncoding="ISO-8859-1"%>  
<%@ taglib uri="<http://jakarta.apache.org/struts/tags-html>" prefix="html"%>  
<html>  
  <head>  
    <title>My JSP 'index.jsp' starting page</title>  
  </head>  
    
  <body>  
    <html:link action="perpareInsert">insert</html:link>  
  </body>  
</html>

---------------------------------------------------------------------insert.jsp

<%@ page language="java" pageEncoding="ISO-8859-1"%>  
<%@ taglib uri="<http://jakarta.apache.org/struts/tags-bean>" prefix="bean"%>   
<%@ taglib uri="<http://jakarta.apache.org/struts/tags-html>" prefix="html"%>  
   
<html>   
 <head>  
  <title>JSP for InsertForm form</title>  
 </head>  
 <body>  
  <html:form action="/insert">  
   password : <html:text property="password"/><br/>  
   username : <html:text property="username"/><br/>  
   email : <html:text property="email"/><br/>  
   <html:submit/><html:cancel/>  
  </html:form>  
  <html:errors/>  
 </body>  
</html>

---------------------------------------------------------------------InsertAction

/\*  
 \* Generated by MyEclipse Struts  
 \* Template path: templates/java/JavaClass.vtl  
 \*/  
package org.zhym.struts.action;

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

import org.apache.struts.action.Action;  
import org.apache.struts.action.ActionForm;  
import org.apache.struts.action.ActionForward;  
import org.apache.struts.action.ActionMapping;  
import org.apache.struts.action.ActionMessage;  
import org.apache.struts.action.ActionMessages;  
import org.zhym.struts.form.InsertForm;

/\*\*   
 \* MyEclipse Struts  
 \* Creation date: 12-17-2007  
 \*   
 \* XDoclet definition:  
 \* @struts.action path="/insert" name="insertForm" input="/insert.jsp" scope="request" validate="true"  
 \*/  
public class InsertAction extends Action {  
 /\*  
  \* Generated Methods  
  \*/

 /\*\*   
  \* Method execute  
  \* @param mapping  
  \* @param form  
  \* @param request  
  \* @param response  
  \* @return ActionForward  
  \*/  
 public ActionForward execute(ActionMapping mapping, ActionForm form,  
   HttpServletRequest request, HttpServletResponse response) {  
  InsertForm insertForm = (InsertForm) form;// TODO Auto-generated method stub  
  if(!isTokenValid(request)){  
   ActionMessages errors = new ActionMessages();  
   errors.add("insertToken", new ActionMessage("error.token"));  
   this.saveErrors(request, errors);  
   this.saveToken(request);  
   return mapping.getInputForward();  
  }else{  
   this.resetToken(request);  
  }  
  return null;  
 }  
}

--------------------------------------------------------------------PerpareInsertAction

/\*  
 \* Generated by MyEclipse Struts  
 \* Template path: templates/java/JavaClass.vtl  
 \*/  
package org.zhym.struts.action;

import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import org.apache.struts.action.Action;  
import org.apache.struts.action.ActionForm;  
import org.apache.struts.action.ActionForward;  
import org.apache.struts.action.ActionMapping;

/\*\*   
 \* MyEclipse Struts  
 \* Creation date: 12-17-2007  
 \*   
 \* XDoclet definition:  
 \* @struts.action validate="true"  
 \*/  
public class PerpareInsertAction extends Action {  
 /\*  
  \* Generated Methods  
  \*/

 /\*\*   
  \* Method execute  
  \* @param mapping  
  \* @param form  
  \* @param request  
  \* @param response  
  \* @return ActionForward  
  \*/  
 public ActionForward execute(ActionMapping mapping, ActionForm form,  
   HttpServletRequest request, HttpServletResponse response) {  
  // TODO Auto-generated method stub  
  this.saveToken(request);  
  return mapping.findForward("insert");  
 }  
}

------------------------------------------------------------------ApplicationResources

# Resources for parameter 'org.zhym.struts.ApplicationResources'  
# Project Struts1.2Token  
error.token=repeat submit!

================================说  明=======================================

利用Struts的同步令牌（token）机制来解决重复提交问题。

执行流程：

PerpareInserAction---->insert.jsp---->InsertAction

在PrepareInsertAction中创建一个token令牌

saveToken(request)

在InsertAction对其进行判断

isTokenValid(request)如果为假表示进行了重复的操作

saveToken(request)

在session中创建一个令牌（org.apache.struts.util.TokenProcessor完成）

resetToken(request)

从session中删除token

## 使用JavaScript和Struts防止表单页面重复提交

(2007-04-09 20:40:37)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载](javascript:;)

|  |  |
| --- | --- |
| 标签：  [j2ee](http://uni.sina.com.cn/c.php?t=blog&k=j2ee&ts=bpost&stype=tag) | 分类：[J2EE](http://blog.sina.com.cn/s/articlelist_1252262387_1_1.html) |

环境:

TOMCAT5.0.28和STRUTS1.2(MYECLIPSE5.1.1集成)

1.使用JAVASCRIPT防止表单重复提交:

原理很简单,通过计算时间来判断是否是已提交过的表单,代码如下:

<%@ page language="java" contentType="text/html; charset=gb2312"  
    pageEncoding="gb2312"%>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">  
<title>提交</title>

<script Language='JavaScript'>  
<!--  
function formsubmit()  
{  
 Today = new Date();  
 var NowHour = Today.getHours();  
 var NowMinute = Today.getMinutes();  
 var NowSecond = Today.getSeconds();  
 var mysec = (NowHour\*3600)+(NowMinute\*60)+NowSecond;  
 if((mysec-document.formsubmitf.mypretime.value)>600)//600是一个时间值  
 {  
  document.formsubmitf.mypretime.value=mysec;  
 }  
 else  
 {  
  alert(' 请勿重复提交！请耐心等待！谢谢合作！');  
  return false;  
 }  
 document.forms.formsubmitf.submit();  
}  
-->  
</script>

</head>  
<body>  
<center><font color=red size=5>javascript防止表单重复提交</font></center>  
<form name=formsubmitf id ="the" method="post" action="submint.jsp">

<input name="user\_nickname" type="text" size="20">

<input name="user\_psw" type="password" size="20">

<input type="button" value="写好了" onclick='formsubmit()'>  
&nbsp;<font class="red">(请按一次，耐心等待！)</font>&nbsp;  
<input type="reset" value="重 写">  
</center>  
</form>  
</body>  
</html>

2.使用Struts的Token来防止表单(FORM)重复提交:

基本原理: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。

工作流程:index.jsp->TokenAction(toregister)->register.jsp->TokenAction(register)->message.jsp

index.jsp:

<jsp:forward page="/toregister.do"></jsp:forward>

struts-config.xml:

<struts-config>  
  <data-sources />  
  <form-beans>  
   <form-bean name="tokenForm" type="form.TokenForm" />  
  </form-beans>  
  <global-exceptions />  
  <global-forwards />  
  <action-mappings>  
   <action path="/toregister" type="struts.TokenAction" parameter="toregister">  
    <forward name="register" path="/register.jsp"></forward>  
   </action>  
     
   <action path="/register" type="struts.TokenAction" parameter="register"  
    name="tokenForm" scope="request" input="/register.jsp" validate="false">  
    <forward name="message" path="/message.jsp"></forward>  
   </action>  
  </action-mappings>  
  <message-resources parameter="struts.ApplicationResources" />  
</struts-config>

TokenAction(toregister):

public ActionForward toregister(ActionMapping mapping, ActionForm form,  
   HttpServletRequest request, HttpServletResponse response)  
   throws Exception {  
  // 保存Token  
  saveToken(request);  
  System.out.println("after saveToken in toregister:"  
    + generateToken(request));  
  return mapping.findForward("register");  
 }

这个方法中saveToken(request)是关键.它向服务器端的SESSION中保存1个TOKEN,同时向客户端发送一个相同的TOKEN,这个TOKEN是用隐藏域表示的.

register.jsp:

<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>  
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
    pageEncoding="ISO-8859-1"%>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  
<title>Insert title here</title>  
</head>  
<body>  
<center>  
 <h3>register</h3>  
 <hr>  
 <html:form method="post" action="/register.do">  
  name:<input type="text" name="name"/>  
  password:<input type="password" name="password"/>  
  <input type="submit" value="OK"/>  
 </html:form>  
 <hr>  
 <%  
  java.util.Enumeration attributeNames = session.getAttributeNames();  
    while(attributeNames.hasMoreElements())  
    {  
     String name = (String)attributeNames.nextElement();  
     Object value = session.getAttribute(name);  
     out.println(name + "=" + value);  
    }  
  %>  
</center>  
</body>  
</html>

中间的java语句可以让我们看到此页面保存的TOKEN的值.

这里有一点值得注意:我通过测试,使用TOKEN防止表单重复提交只能用STRUTS的THML标签来生成表单,如果用我们常用的HTML语句的FORM表单则无法正常工作,同时这里的FORM对应一个TokenForm类,TokenForm类代码如下:

package form;

import org.apache.struts.action.ActionForm;

public class TokenForm extends ActionForm {

 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;  
 }

}  
下面看一下关键的TokenAction中的register方法:

public ActionForward register(ActionMapping mapping, ActionForm form,  
   HttpServletRequest request, HttpServletResponse response)  
   throws Exception {  
  String tokenString = (String)request.getParameter("org.apache.struts.taglib.html.TOKEN");  
  System.out.println("in register token is: " + tokenString);  
  if(isTokenValid(request, true))  
        {  
            //success operation!  
            request.setAttribute("message", "Congratulations!");  
        }else  
        {  
            request.setAttribute("message", "Sorry, the request has been handled!");  
        }  
         
        return mapping.findForward("message");  
 }

isTokenValid(request, true){}这个方法是验证客户端的TOKEN是否与服务器端的一样.如果一样,则在SESSION中保存一个值为success的message信息.如果不一样的保存Sorry, the request has been handled!并且修改服务器端的TOKEN的值(true的作用便在于此).

此外,在TokenAction中还有reset(request)方法,这个方法是删除服务器保存的Token.

这样.就可以实现使用STRUTS来防止表单的重复提交了.

总结:这2种方法都可以防止重复提交表单.但是在下列情况下还是不能防止的:比如一个用户登录系统后点回退键,之后刷新页面,再次登录页面.

遇到这种情况,个人认为则需要去判断SESSION中保存的内容来判断用户是否已经登录了.并通过返回不同的结果来显示不同的页面.如果有更好的意见,请予以提出,不胜感激.

开始一直都对这个重复提交的问题理解的都不是很好，就知道有几个方法用了就可以达到效果，之于这几个方法，为什么能够达到这个效果就不得而知了。   
  
现在，对这几个方法回头整理一下，一下子就明白了。我用的是 struts 的令牌。   
举例，以论坛留言来说明：   
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：   
第一个进入发帖页面：   
/InsertToken/liuYan.do?method=add

Java代码 [复制代码](http://www.javaeye.com/topic/216801)

1. public ActionForward add(ActionMapping mapping, ActionForm form,
2. HttpServletRequest request, HttpServletResponse response) {
3. this.saveToken(request);
4. return mapping.findForward("success");
5. }

public ActionForward add(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response) {

this.saveToken(request);

return mapping.findForward("success");

}

成功后，会生成新的页面：   
      这个生成的新页面中都会包含类似如下的类容：

Hmtl代码 [复制代码](http://www.javaeye.com/topic/216801)

1. <div>
2. <input type="hidden" name="org.apache.struts.taglib.html.TOKEN" value="70569d6c53ecfbe4748cf5b7b1a4d87e">
3. </div>

<div>

<input type="hidden" name="org.apache.struts.taglib.html.TOKEN" value="70569d6c53ecfbe4748cf5b7b1a4d87e">

</div>

这一组标签中value的值是经常变换的，每刷新一次，重新生成新页面的时候就会变化一次   
而且这组标签都会生成在form标签内部，如果没有form标签是不会生成的。   
看了源代码之后过程如下：   
首先

Java代码 [复制代码](http://www.javaeye.com/topic/216801)

1. public synchronized void saveToken(HttpServletRequest request) {
2. HttpSession session = request.getSession();
3. String token = generateToken(request);
4. if (token != null) {
5. session.setAttribute(Globals.TRANSACTION\_TOKEN\_KEY, token);
6. }
7. }
8. public synchronized String generateToken(HttpServletRequest request) {
9. HttpSession session = request.getSession();
10. return generateToken(session.getId());
11. }

public synchronized void saveToken(HttpServletRequest request) {

HttpSession session = request.getSession();

String token = generateToken(request);

if (token != null) {

session.setAttribute(Globals.TRANSACTION\_TOKEN\_KEY, token);

}

}

public synchronized String generateToken(HttpServletRequest request) {

HttpSession session = request.getSession();

return generateToken(session.getId());

}

这里在保存后 这里获得了 session 的 id 并进行了generateToken(session.getId())的操作返回一个可变的值，然后保存在 session 中。并用Globals.TRANSACTION\_TOKEN\_KEY指代，这个是第一步。   
  
然后 mapping.findForward("success"); 生成页面。这个新的页面如果有 form 标签那么就会在 FormTag.class 中进行如下的执行，以形成控制重复提交的必要标签

Java代码 [复制代码](http://www.javaeye.com/topic/216801)

1. protected String renderToken() {
2. StringBuffer results = new StringBuffer();
3. HttpSession session = pageContext.getSession();
4. if (session != null) {
5. String token =(String) session.getAttribute(Globals.TRANSACTION\_TOKEN\_KEY);
6. if (token != null) {
7. results.append("<div><input type=\"hidden\" name=\"");
8. results.append(Constants.TOKEN\_KEY);
9. results.append("\" value=\"");
10. results.append(token);
11. if (this.isXhtml()) {
12. results.append("\" />");
13. } else {
14. results.append("\">");
15. }
16. results.append("</div>");
17. }
18. }
19. return results.toString();
20. }

protected String renderToken() {

StringBuffer results = new StringBuffer();

HttpSession session = pageContext.getSession();

if (session != null) {

String token =(String) session.getAttribute(Globals.TRANSACTION\_TOKEN\_KEY);

if (token != null) {

results.append("<div><input type=\"hidden\" name=\"");

results.append(Constants.TOKEN\_KEY);

results.append("\" value=\"");

results.append(token);

if (this.isXhtml()) {

results.append("\" />");

} else {

results.append("\">");

}

results.append("</div>");

}

}

return results.toString();

}

在此取得 Globals.TRANSACTION\_TOKEN\_KEY 中之然后一起合成新页面的标签 就是上面的   
<div></div> 中的代码.   
在新页面形成之后，可以进行相应的操作，然后提交。   
/insertinfo?method=insertinfo

Java代码 [复制代码](http://www.javaeye.com/topic/216801)

1. public ActionForward insertinfo(ActionMapping mapping, ActionForm form,
2. HttpServletRequest request, HttpServletResponse response) {
3. if (this.isTokenValid(request)) {
4. System.out.println("\*\*\*\*执行\*\*\*\*" + saved.equals(token));
5. this.resetToken(request);
6. return mapping.findForward("insertok");
7. } else {
8. System.out.println("++重复提交++");
9. this.saveToken(request);
10. return mapping.findForward("insertnotok");
11. }
12. }

public ActionForward insertinfo(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response) {

if (this.isTokenValid(request)) {

System.out.println("\*\*\*\*执行\*\*\*\*" + saved.equals(token));

this.resetToken(request);

return mapping.findForward("insertok");

} else {

System.out.println("++重复提交++");

this.saveToken(request);

return mapping.findForward("insertnotok");

}

}

进入这块代码 ，首先是验证是不是重复提交(if)，然后通过函数resetToken从当前session范围内删除令牌属性。   
或者 已经提交（else），然后再通过函数saveToken在session中重新设置新的值。   
过程分析具体如下：

Java代码 [复制代码](http://www.javaeye.com/topic/216801)

1. public synchronized boolean isTokenValid(HttpServletRequest request,
2. boolean reset) {
3. // Retrieve the current session for this request
4. HttpSession session = request.getSession(false);
5. if (session == null) {
6. return false;
7. }
8. // Retrieve the transaction token from this session, and
9. // reset it if requested
10. String saved =
11. (String) session.getAttribute(Globals.TRANSACTION\_TOKEN\_KEY);
12. if (saved == null) {
13. return false;
14. }
15. if (reset) {
16. this.resetToken(request);
17. }
18. // Retrieve the transaction token included in this request
19. String token = request.getParameter(Globals.TOKEN\_KEY);
20. if (token == null) {
21. return false;
22. }
23. return saved.equals(token);
24. }

public synchronized boolean isTokenValid(HttpServletRequest request,

boolean reset) {

// Retrieve the current session for this request

HttpSession session = request.getSession(false);

if (session == null) {

return false;

}

// Retrieve the transaction token from this session, and

// reset it if requested

String saved =

(String) session.getAttribute(Globals.TRANSACTION\_TOKEN\_KEY);

if (saved == null) {

return false;

}

if (reset) {

this.resetToken(request);

}

// Retrieve the transaction token included in this request

String token = request.getParameter(Globals.TOKEN\_KEY);

if (token == null) {

return false;

}

return saved.equals(token);

}

第一次得到的 saved 和 token 值是一样的。这里 Globals.TRANSACTION\_TOKEN\_KEY   
= "org.apache.struts.action.TOKEN"   
Globals.TOKEN\_KEY 就是生成标签的name属性org.apache.struts.taglib.html.TOKEN   
通过比较第一次 相同 提交成功。   
重设和第一次一样。   
然后刷新提交后的页面，或者后退重填提交。此时生成的标签的value不变。但是在第一次的时候 resetToken 已经从当前 session 范围内删除令牌属性。所以直接进入 else。这里报告重复提交。同时再次保存 token ，此时的保存值又和value值不一样。所以重复体提交失败。

* [InsertToken.rar](http://dl.javaeye.com/topics/download/0cf1d1cb-49c5-3e8b-8d4a-24c3d2227acf) (17 KB)
* 描述: 代码
* 下载次数: 209