TokenizedRequestFilter

NOTE: This is information applies to Tapestry 4 but *may* be applicable to newer versions.

Many of my applications require a significant degree of server-side state, but unfortunately this can lead to user frustration if the application is not tolerant to stale sessions. Tapestry provides a nice mechanism for page validation via the PageValidateListener interface, giving the application a convenient way to redirect to (for example) a login page to verify user credentials and possibly for creation of a Visit or other necessary application state objects. In my situation, all non-validated users are redirected to a central authentication server, which redirects back to the original application upon successful login.

The frustration arises when the application is left idle for an extended period of time and the user later returns with the expectation of being able to continue uninterrupted. Since frequently the server session no longer exists, the authentication server must be called upon to re-establish credentials. In my case, this step is often performed transparently by the use of cookies, so all the user sees after submitting a form or clicking a link is either the Home page, a Stale Session page, or possibly the current page re-rendered without the user's changes. Without a way of preserving the original request, this redirection (sometimes multiple redirections) results in the loss of the user's form data (or other requested actions).

I've tried the brute force method of encoding the POST form data into the URL, but browser URL limits of anywhere from 2K to 4K bytes severely limit the usefulness of this method. Jeff Lubetkin described a method he has used of implementing WebRequestServicerFilter to be able to participate in the Tapestry request process via the WebRequestServicerPipeline. I have made use of this in the past for other purposes such as monitoring for certain types of requests, and it seemed appropriate and fairly straight-forward to implement it here as well.

Since the specifics of pageValidate are very application dependent, I will only show a portion of that below.

To start with, please add this to your hivemodule.xml

```
<service-point id="tokenizedRequestFilter" interface="org.apache.tapestry.services.WebRequestServicerFilter">
   <invoke-factory>
      <construct class="com.foo.TokenizedRequestFilter">
         <set-service property="request" service-id="tapestry.globals.HttpServletRequest"/>
         <set-service property="response" service-id="tapestry.globals.HttpServletResponse"/>
      </construct>
   </invoke-factory>
</service-point>

<contribution configuration-id="tapestry.request.WebRequestServicerPipeline">
   <filter name="tokenizedRequestFilter" object="service:tokenizedRequestFilter"/>
</contribution>
```

Next, TokenizedRequestFilter.java. Please notice the comments as well, although that situation should not normally occur unless the user expired a session, recovered using the TokenizedRequestFilter, allowed that second session to also eventually expire, and then performed a browser refresh, resubmitting the old postToken. In that unlikely case, the application-defined ExceptionPage will be rendered.

```
package com.foo;

import java.io.IOException;

import org.apache.tapestry.services.WebRequestServicer;
import org.apache.tapestry.services.WebRequestServicerFilter;
import org.apache.tapestry.web.WebRequest;
import org.apache.tapestry.web.WebResponse;

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

/**
 * Web request filter to service tokenized requests.
 *
 * @author Shawn M Church
 * @version TokenizedRequestFilter, Oct 23, 2006, 10:21:01 AM
 */
public class TokenizedRequestFilter implements WebRequestServicerFilter
{
   private HttpServletRequest _request;
   private HttpServletResponse _response;

   public void service(WebRequest request, WebResponse response, WebRequestServicer servicer)
      throws IOException
   {
      // Looking only for a single parameter named postToken.
      // All exceptions are caught so the current request may be
      // attempted as a last resort.  This will most likely fail since
      // the expected parameters will not have been supplied,
      // but our application ExceptionPage should at least
      // get invoked.
      try
      {
         String postToken = request.getParameterValue(TokenizedRequest.TOKEN_NAME);
         if( postToken != null )
         {
            TokenizedRequest newrequest = new TokenizedRequest(_request, _response, postToken);
            servicer.service(newrequest, response);
            return;
         }
      }
      catch ( Exception e )
      {
      }

      servicer.service(request, response);
   }

   public HttpServletRequest getRequest() {
      return _request;
   }

   public void setRequest(HttpServletRequest _request) {
      this._request = _request;
   }

   public HttpServletResponse getResponse() {
      return _response;
   }

   public void setResponse(HttpServletResponse _response) {
      this._response = _response;
   }
}
```

You will need a ServletWebRequest wrapper called TokenizedRequest.java:

```
package com.foo;

import org.apache.tapestry.web.ServletWebRequest;
import org.apache.tapestry.web.WebUtils;
import org.apache.tapestry.web.WebSession;
import org.apache.hivemind.util.Defense;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.*;

/**
 * Wrapper for ServletWebRequest to find and extract parameter
 * values and names from a Map session attribute based on the
 * specified post token.
 *
 * @author Shawn M Church
 * @version TokenizedRequest, Mon, Oct 23 2006, 10:37:54 AM
 */
public class TokenizedRequest extends ServletWebRequest
{
   public static String TOKEN_NAME = "postToken";
   public static String TOKEN_MAP_NAME = "postTokenMap";

   private Map _parameterMap;

   public TokenizedRequest(HttpServletRequest request, HttpServletResponse response, String postToken) {
      super(request, response);

      Map parameterMap = null;
      WebSession session = getSession(true);
      if( session != null )
         parameterMap = ( Map ) session.getAttribute(TOKEN_MAP_NAME + postToken);
      _parameterMap = parameterMap;
   }

   public List getParameterNames() {
      if( _parameterMap == null )
         return null;

      return WebUtils.toSortedList(Collections.enumeration(_parameterMap.keySet()));
   }

   public String[] getParameterValues(String name) {
      if( _parameterMap == null )
         return null;

      Defense.notNull(name, "name");

      Object values = _parameterMap.get(name);
      if( values == null || values instanceof String[] )
         return ( String[] ) values;

      String value = ( String ) values;

      return new String[]{value};
   }

   public String getParameterValue(String name) {
      if( _parameterMap == null )
         return null;

      Defense.notNull(name, "name");

      Object values = _parameterMap.get(name);
      if( values == null || values instanceof String )
         return ( String ) values;

      String[] array = ( String[] ) values;
      return array[0];
```

```
   }
}
```

Finally, you will need a way to store the request parameters, for example invoked from your pageValidate method (this is just an example):

```
   /**
    * Capture and save the request URL to the visit object,
    * storing the parameter Map from the request.
    *
    * @param visit
    */
   protected void buildRedirect(Visit visit) {
      if( visit == null )
         return;

      try
      {
         int postToken = new Random().nextInt();

         StringBuffer urlbuf = new StringBuffer();
         urlbuf.append(getWebRequest().getScheme()).append("://");
         urlbuf.append(getWebRequest().getServerName());
         if( getWebRequest().getServerPort() != 80 )
            urlbuf.append(":").append(getWebRequest().getServerPort());
         urlbuf.append(getWebRequest().getRequestURI());
         urlbuf.append("?").append(TokenizedRequest.TOKEN_NAME).append("=").append(postToken);

         visit.setRedirect(urlbuf.toString());

         WebSession session = getWebRequest().getSession(true);
         if( session != null )
            session.setAttribute(TokenizedRequest.TOKEN_MAP_NAME + postToken, getHttpServletRequest().
getParameterMap());
      }
      catch ( Exception e )
      {
         _log.error(e);
      }
   }
```

Hopefully you will find this useful.