StrutsRememberMeCookie

It is often needed to set/unset/show "Remeber Me" checkbox. The shown method works together with the cookie, stored on user's machine. When a user navigates to the page, action must set "Remember Me" checkbox if cookie is supplied by browser.

Cookie/checkbox thing has one small issue: when cookie was set before, and no checkbox value is submitted by browser. Application must distinguish between two cases:

(1) Action was called directly from browser location bar, or from other page. In other words, action was called for the first time, and it should display the checkbox based on saved cookie.

(2) Action was called after postback. In this case cookie should be set according to checkbox value.

These two cases can be distinguished one of the following way:

- using hidden field. Submitted field identifies postback.
- by differentiating between POST and GET request. POST should be used for input and model update, GET should be used to load page only.

First, a simple login page:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<html>
<body>
  <center><H1>Log In</H1></center><br>
  <html:form action="/Login">
    <!-- Identifies request as postback.
         This field is not needed if POST/GET differentiation is used. -->
    <html:hidden property="usersubmit" value="TRUE"/>
    <table border="0" cellspacing="0" cellpadding="0" align="center">
      <tr valign="middle">
        <td><html:checkbox property="rememberMe"/></td>
        <td>Remember me</td>
        <td><html:submit value="Log In"/></td>
      </tr>
      <tr>
      </tr>
    </table>
  </html:form>
</body>
</html>
```

Now the corresponding struts-config.xml. Notice redirection to itself, this trick allows to access the same action using GET method. Redirection allows to reload page without POSTDATA situation.

```
  <form-bean name="loginForm"
      type="org.acme.LoginForm"/>

  <action-mappings>
      <action path  = "/Login"
              type  = "org.acme.LoginAction"
              name  = "loginForm"
              scope = "session"
              validate = "true">
          <forward name="getview" path="/Login.do" redirect="true"/>
          <forward name="showpage" path="/login.jsp"/>
      </action>
  </action-mappings>
```

Now the form bean:

```
package org.acme;

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

public class LoginForm extends ActionForm
{
    // "Remeber me" checkbox
    private String rememberMe;
    public String getRememberMe() {return rememberMe;}
    public void setRememberMe(String rememberMe){ this.rememberMe = rememberMe;}

    // This property indicates that input was submitted from a client.
    // In this case  the checkbox value should be consulted instead of
    // stored cookie value.
    // This property is NOT needed if POST/GET differentiation is used.
    private String usersubmit;
    public String getUsersubmit() {return usersubmit;}
    public void setUsersubmit(String usersubmit) {this.usersubmit = usersubmit;}

    public void reset(ActionMapping mapping, HttpServletRequest request) {
      rememberMe = null;
      usersubmit = null;
    }
}
```

And finally the action class:

```
package org.acme;

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

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class LoginAction extends Action
{
  public static final String COOKIE_NAME = "COOKIE_REMEMBER_ME";
  public static final int COOKIE_REMOVE = 0;
  public static final int COOKIE_ONEHOUR = 3600;

  public ActionForward execute(ActionMapping mapping, ActionForm form,
            HttpServletRequest request, HttpServletResponse response) {

    LoginForm login = (LoginForm) form;

    // Model should be updated via POST only
    boolean isInput = "POST".equalsIgnoreCase(request.getMethod());

    // Can also use hidden field, if for some reason
    // model can be updated by GET as well
    // isInput = login.getUsersubmit() != null;

    // Page is navigated from external location
    // or after cookie was set appropriately
    if (!isInput) {

      // If cookie is set...
      if (isCookieSet(request)) {
        // ...turn on checkbox
        login.setRememberMe("on");

      // If cookie is not set...
```

```
      } else {
        // ...there is nothing to do, field was cleared in reset()
      }

      // Display the page. Checkbox is set according to the cookie,
      // other fields are pulled from the form, it has session scope.
      return mapping.findForward("showpage");

    // Page is [re]-submitted
    } else {

      // Must [re]-set cookie
      if (login.getRememberMe() != null) {
        Cookie cookie = new Cookie(COOKIE_NAME, "");
        cookie.setMaxAge(COOKIE_ONEHOUR);
        response.addCookie(cookie);

      // Must clean cookie
      } else if (login.getRememberMe() == null && isCookieSet(request)) {
        Cookie cookie = new Cookie(COOKIE_NAME, "");
        cookie.setMaxAge(COOKIE_REMOVE);
        response.addCookie(cookie);
      }

      // Redirect to itself to prepare and display the page.
      // Redirect allows to reload a page safely.
      //
      // Browser is redirected to the same /Login.do action
      // and loads page with empty GET request. Because request has
      // no parameters, getUsersubmit() == null and checkbox
      // is shown correctly, because cookie was already set.
      return mapping.findForward("getview");
    }
  }

  private boolean isCookieSet(HttpServletRequest request) {
    Cookie [] cookies = request.getCookies();
    if ( cookies != null ) {
      for (int i = 0; i < cookies.length; i++) {
        if ( cookies[i].getName().equals(COOKIE_NAME)) {
          return true;
        }
      }
    }
    return false;
  }
}
```