Multi-State Login Dialog example

1. overview

Muliti-state and multi-view login dialog is an extension of single-state login dialog. It not only allows to log in, it also tracks user's login status. Whenever a logged-in user navigates to login dialog's location, the login dialog displays user information and allows to log out.

Converting a single-state login dialog into multi-state login dialog is easy. We need to track two different states of a user, and to change the mapping for result pages, because now we have two pages instead of one.

2. Action Class

Because new login dialog has two views instead of one, we need to override `getView` method and to return proper mapping for each user state.

```
public class DialogLoginActionControl extends DialogLoginAction {

   // Mapping submit button names to methods
   protected Map getKeyMethodMap() {
       Map map = super.getKeyMethodMap();
       map.put("DIALOG-EVENT-LOGOUT", "logout");
       return map;
   }

   /**
    * Overrides login method, always reloads the control instead of navigating
    * to user page on success.
    */
   public ActionForward login (
           ActionMapping mapping,
           ActionForm form,
           HttpServletRequest request,
           HttpServletResponse response) throws Exception {

       // Try to log in
       super.login(mapping, form, request, response);

       // Reload login control even if login was successful.
       return mapping.findForward(DialogConstants.DIALOG_RELOAD_KEY);
   }
```

```
    /**
     * Logs a user out and reloads this login control
     */
    public ActionForward logout (ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response) throws Exception {

        DialogLoginForm inputForm = (DialogLoginForm) form;

        // Clean password in the input/output form bean
        inputForm.setPassword(null);

        // Remove login name from session, effectively logging out
        request.getSession().removeAttribute("login.username");

        // Reload login control
        return mapping.findForward(DialogConstants.DIALOG_RELOAD_KEY);
    }

    /**
     * Render page, corresponding to user's state
     */
    public ActionForward getDialogView(ActionMapping mapping,
                                       ActionForm form,
                                       HttpServletRequest request,
                                       HttpServletResponse response) throws Exception {
        // Modify response header to make page non-cachable.
        super.getDialogView(mapping, form, request, response);

        // Display page, corresponding to login state.
        HttpSession session = request.getSession();
        if (session.getAttribute("login.username") == null) {
            return mapping.findForward("notloggedin");
        } else {
            return mapping.findForward("loggedin");
        }
    }
}
```

3. Action Mapping

Notice, that instead of using default "DIALOG-VIEW" mapping for a view, we use two different mappings for each user state: "logged in" and "not logged in".

```
 <action path="/logincomponent"
         type      = "package net.jspcontrols.dialogs.samples.login.DialogLoginActionC
         name      = "dialogloginform"
         scope     = "session"
         validate  = "false"
         parameter = "DIALOG-EVENT">
         <!-- Reload login component -->
         <forward name = "DIALOG-RELOAD" path="/logincomponent.do" redirect="true"/>
```

```
        <!-- Login page -->
        <forward name="notloggedin" path="/logincomponent-login.jsp"/>
        <!-- Userinfo/Logout page -->
        <forward name="loggedin" path="/logincomponent-logout.jsp"/>
</action>
```

4. Live Demo

Live demo of multi-state login dialog