Common-Controls Guided Tour TabSetControl

Version 1.0.3 - Last changed: 01. August 2004

Common Controls

Herausgeber:
SCC Informationssysteme GmbH
64367 Mühltal

Tel: +49 (0) 6151 / 13 6 31 0
Internet www.scc-gmbh.com

Product Site:
www.common-controls.com

Inhaltsverzeichnis

1 Guided Tour TabSetControl

1.1 Object

This exercise demonstrates the use of the TabSetControl. In the course of this exercise, a TabSet will be generated, that includes several JSP pages for depicting the tab (Rider). The TabSetControl can work through round-trips with or without the server. In addition, the number of visible taps is restricted. Is there are more taps present, the corresponding navigation elements are hidden. The scrolling through the tabs is also possible without server roudtrips.

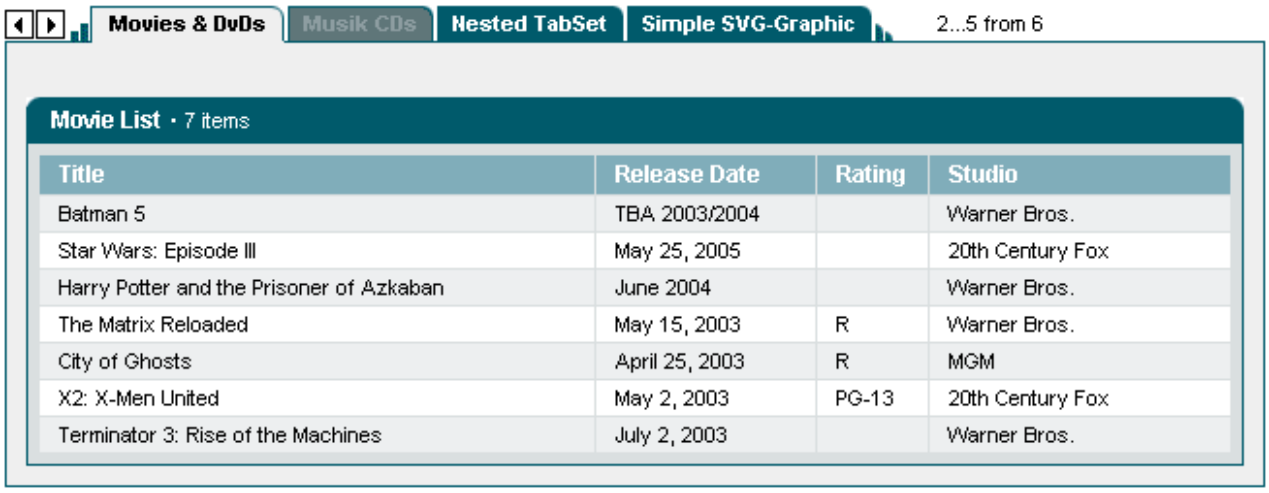
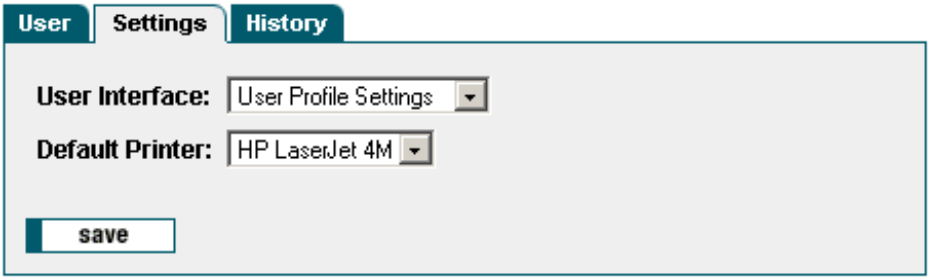


Figure 1: Example TabSetControl

In the exercise, we would like to generate the following TabSet.



The following steps are required for using the TabSetControls:

1. Selection of the design for the user interface.
2. Generation of an action class for calling the JSP-Page.
3. Provision of display data per tab.
4. Configuration of the Tabset within the JSP-Page.

1.2 Registration of the Painterfactory

The registration of the Painterfactory takes place first. It defines which design the user interface will get. This can be done across the application in the init()-method of the Frontcontroler-Servlet.[1] Here, we select the standard design that the DefaultPainter offers us.[2]

```
import javax.servlet.ServletException

import org.apache.struts.action.ActionServlet;
import com.cc.framework.ui.painter.PainterFactory
import com.cc.framework.ui.painter.def.DefPainterFactory;
import com.cc.framework.ui.painter.html.HtmlPainterFactory;

public class MyFrontController extends ActionServlet {

    public void init() throws ServletException {

       super.init();

       // Register all Painter Factories with the preferred GUI-Design
       // In this case we use the Default-Design.
       PainterFactory.registerApplicationPainter (
              getServletContext (), DefPainterFactory.instance());
       PainterFactory.registerApplicationPainter (
              getServletContext (), HtmlPainterFactory.instance());
    }
}
```

1.3 Creation of the action class for the Struts adapter

In our example, we want to generate a TabSet that includes three tabs via JSP-pages. The TabSetControl is supposed to work without server round-trips. This option is available where the data to be presented cannot be completely made available at any time and, as in our case, hang together logically. In the case of complex screens with heterogeneous information contents, on the other hand, it is mostly more convenient to load to the corresponding tab only when there is a change.

For us, the determining of the display data is taken care of by the action class "UserProfileEditAction". It is derived from the class FWAction, which capsulates the Struts-action class and extends with functionalities of the presentation framework. Instead of the execute()-method, the doExecute()-method is called. [FWAction is derived from org.apache.struts.action.Action]. On calling, it contains the ActionContext, through which the access to additional objects such as the Request- and Response-object is capsulated.

```
import java.io.IOException;
import javax.servlet.ServletException;

import com.cc.framework.adapter.struts.FWAction;
import com.cc.framework.adapter.struts.ActionContext;

public class UserProfileEditAction extends FWAction {

    /**
     * @see com.cc.framework.adapter.struts.FWAction#doExecute(ActionContext)
     */
    public void doExecute(ActionContext ctx)
        throws IOException, ServletException {
        // see next chapter
    }
}
```

[1] If it has to be possible for the individual user to choose between different interface designs, then additional PainterFactorys are registered in the user session. This is done mostly in the LoginAction with PainterFactory.registerSessionPainter() in the session Scope.

[2] Additional designs (PainterFactories) are included in the kit of the Professional Edition, or you can develop them yourself.

1.4 Provision of display data

Since our TabSetControl should switch between the tabs on the client-side, i.e. without the server, all the data must be loaded at the beginning. We use a Formbean for handing over to the JSP-Page. It provides the various tabs access to the display data.

```
import java.io.IOException;

import javax.servlet.ServletException;

import com.cc.framework.adapter.struts.ActionContext;
import com.cc.framework.adapter.struts.FWAction;
import com.cc.framework.adapter.struts.FormActionContext;
import com.cc.sampleapp.common.Messages;
import com.cc.sampleapp.common.User;
import com.cc.sampleapp.tabset.sample402.form.UserProfileEditForm;

public class UserProfileEditAction extends FWAction {

    /**
     * @see com.cc.framework.adapter.struts.FWAction#doExecute(ActionContext)
     */
    public void doExecute(ActionContext ctx)
        throws IOException, ServletException {

        try {
            // Generate a Default User for our Example
            User user = new User("FAS");
            user.load();

            initFormBean(ctx, user);
        }
        catch (Throwable t) {
            ctx.addGlobalError("Error while loading User Object", t);
            log.error("Error: ", t);
        }

        // Display the JSP
        ctx.forwardToInput();
    }

    /**
     * Initializ the Form with the User-Data
     * @param   ctx     ActionContext
     * @param   user    User-Object
     * @exception       java.lang.Exception
     */
    private void initFormBean(ActionContext ctx, User user) throws Exception {
        UserProfileEditForm form = (UserProfileEditForm) ctx.form();

        form.setUserId( user.getUserId());
        form.setFirstName( user.getFirstName() );
        form.setLastName( user.getLastName() );
        form.setRole( user.getRole() );

        form.setGuitype( user.getSettings().getGuitype() );
        form.setDefprinter( user.getSettings().getDefprinter() );
    }
}
```

1.5 Configuration of the Tabset within the JSP-Page

In order to use the TabSet-Tag on a JSP page, the corresponding tag library must be declared at the start of the page. Then, the Common-Controls can be used with the prefix <ctrl:*tagname* />. [In addition, the incorporation of the tag libraries must be included in the Deployment descriptor, the WEB-INF/web.xml file]

```
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/tlds/cc-controls.tld" prefix="ctrl" %>

<html:form action="/sample402/userprofilEdit" method="post">

    <ctrl:tabset
        id="uptabset"
        property="tabset"
        tabs="3"
        labellength="20"
        width="450"
        runat="client">

        <ctrl:tab
            id="tab1"
            title="User"
            content="Tab_Page1.jsp"
            tooltip="User Display"/>

        <ctrl:tab
            id="tab2"
            title="Settings"
            content="Tab_Page2.jsp"
            tooltip="User Settings"/>

        <ctrl:tab
            id="tab3"
            title="History"
            content="Tab_Page3.jsp"
            tooltip="History"/>
    </ctrl:tabset>

</html:form>
```

In our example, for the sake of clarity, we have included the individual tabs as independent JSP-pages. The details source code is available with the demo-version.

Alternatively, the HTML-code can also be specified directly within the tag-body of a tab:

```
    <ctrl:tab   id="tab3"  title="History"     tooltip="History"/>
        <b>Hello World!</b>
    </ctrl:tab>
```

1.6 Tour End

The TabSetControl can be easily and quickly integrated. Using the configuration in the JSP-Page, new tabs can be quickly added, which can be additionally controlled in an authorization-dependent manner.

The HTML code is generated with a Painter. Other designs can also be implemented by customizing the Painter. Various Designs can also be used in parallel.

Features of the TabSetControl:

- Switching of the tabs configurable via Server Roundtrips or the client-side (without server roundtrip).
- Supports client-side scrolling through the tabs.
- Allows the including of any desired JSP-pages as tabs.
- Icons can be used in front of the labels on the tabs.
- Individual tabs can be disabled.
- Design through Painterfactory can be matched to own StyleGuide (Corporate Identity).
- Nested TabSets possible.
- Optimized HTML-Code.
- Same Look and Feel in Microsoft InternetExplorer > 5.x and Netscape Navigator > 7.x

2 Glossary

C

CC

Common-Controls