SCC
INFORMATIONSSYSTEME

COMMON CONTROLS · GUIDED TOUR

TabSetControl

Das Kontrollelement Schritt für Schritt in eine bestehende Anwendung einbauen

SCC Informationssysteme GmbH

Version 1.6 · Stand: 14.01.2006

Impressum

HERAUSGEBER

SCC Informationssysteme GmbH
Drosselweg 13
64367 Mühltal
+49 6151 13 63 10
www.scc-gmbh.com

Inhalt

1 Guided Tour TabSetControl

1.1 Gegenstand

Diese Übung demonstriert den Einsatz des TabSetControls. Im Rahmen der Übung wird ein TabSet erstellt, das mehrere JSP-Seiten zur Darstellung der Taben (Reiter) inkludiert. Das TabSetControl kann mit oder ohne Server Roundtrips arbeiten. Zudem läßt sich die Anzahl der sichtbaren Taben beschränken. Sind mehr Taben vorhanden, werden entsprechende Navigationselemente eingeblendet. Das Scrolling durch die Taben ist ebenfalls ohne Server Roundtrips möglich.

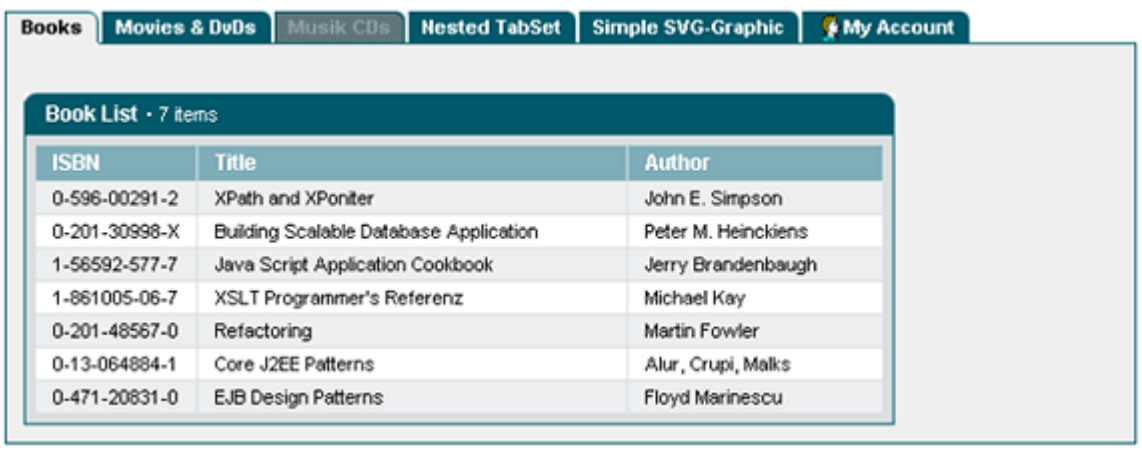

Abbildung 1: Beispieldarstellung TabSetControl

Innerhalb der Übung wollen wir das nachfolgende TabSet erstellen:

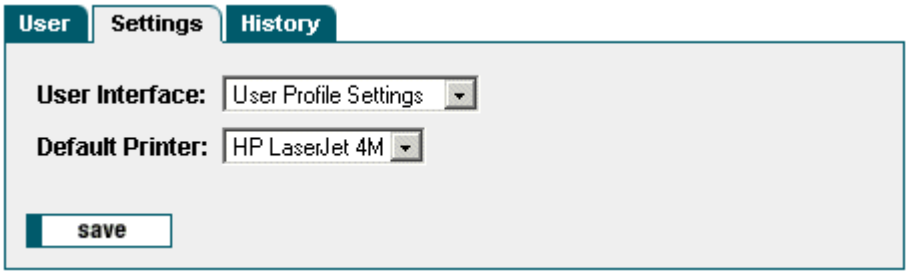

Abbildung 2: Das TabSet dieser Übung

Zum Einsatz des TabSetControls sind folgende Schritte notwendig:

1. Auswahl des Designs für die Benutzeroberfläche
2. Erstellung einer Action-Klasse zur Aufruf der JSP-Seite
3. Bereitstellung der Anzeigedaten je Tabe
4. Konfiguration des Tabsets innerhalb der JSP-Seite

1.2 Registrierung der Painterfactory

Zuerst erfolgt die Registrierung der Painterfactory. Sie legt fest, welches Design die Benutzeroberfläche erhält. Dies kann anwendungsweit in der init()-Methode des Frontcontroler-Servlets geschehen.[1] Wir wählen hier das Standarddesign, welches uns der DefaultPainter bereitstellt.[2]

JAVA

```java
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-Layout
        // In this case we only use the Default-Layout.
        PainterFactory.registerApplicationPainter (
            getServletContext(), DefPainterFactory.instance());
        PainterFactory.registerApplicationPainter (
            getServletContext(), HtmlPainterFactory.instance());
    }
}
```

[1] Wenn der einzelne Benutzer zwischen verschiedenen Oberflächendesigns wählen können soll, dann werden zusätzliche PainterFactorys in der Benutzersession registriert. Dies erfolgt meist in der LoginAction mit PainterFactory.registerSessionPainter() im Session Scope.

[2] Weitere Designs (PainterFactories) sind im Lieferumfang der Professional Edition enthalten, oder können selbst entwickelt werden.

1.3 Ableitung der Action Klasse für den Struts-Adapter

In unserem Beispiel wollen wir ein TabSet erstellen, dass drei Taben über JSP-Seiten inkludieren. Das TabSet soll ohne Server Roundtrips arbeiten. Dies bietet sich dann an, wenn die zu präsentierenden Daten zu einem Zeitpunkt vollständig bereitgestellt werden können und, wie in unserem Fall, logisch zusammen hängen. Bei komplexen Masken mit heterogenen Informationsinhalten ist es hingegen meist günstiger die Daten erst bei einem Wechsel auf die entsprechende Tabe zu laden.

Die Ermittlung der Anzeigedaten übernimmt bei uns die Aktionklasse "UserProfileEditAction". Sie ist von der Klasse FWAction abgeleitet, welche die Struts-Action Klasse kapselt und um Funktionalitäten des Präsentationsframeworks erweitert. Dabei wird anstelle der execute()-Methode die doExecute()-Methode aufgerufen. [FWAction ist von org.apache.struts.action.Action abgeleitet] Sie erhält beim Aufruf den ActionContext, über den der Zugriff auf weitere Objekte, wie das Session-, Request- und Response-Objekt gekapselt ist.

JAVA

```java
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
    }
}
```

1.4 Bereitstellung der Anzeigedaten

Da unser TabSetControl clientseitig, d.h ohne Server Roundtrips, zwischen den Taben umschalten soll, müssen zu Begin alle Daten geladen werden. Zur Übergabe an die JSP-Seite verwenden wir dabei eine Formbean. Sie dient bietet den verschiedenen Taben den Zugriff auf die Anzeigedaten.

JAVA

```java
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 Konfiguration des TabSets innerhalb der JSP-Seite

Um das TabSet-Tag auf einer JSP Seite einzusetzen, muss am Anfang der Seite die entsprechende Tag Library deklariert werden. Anschließend können die Common-Controls mit dem Präfix <ctrl:tagname /> verwendet werden. [Zudem muss die Aufnahme der Tag Bibliothek im Deployment-Deskriptor, der WEB-INF/web.xml Datei, erfolgen].

JSP

```jsp
<%@ 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 unserem Beispiel haben wir der Übersichtlichkeit die einzelnen Taben als eigenständige JSP-Seiten inkludiert. Der ausführliche Source-Code ist mit der Demoversion erhältlich.

Alternativ kann der HTML-Code auch direkt innerhalb des Tag-Bodys einer Tabe angegeben werden:

JSP

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

1.6 Tour Ende

Das TabSetControl lässt sich schnell und einfach integrieren. Über die Konfiguration in der JSP-Seite lassen sich schnell neue Taben hinzufügen, die sich zusätzlich berechtigungsabhängig steuern lassen. Der HTML-Code wird über einen Painter erzeugt. Andere Designs lassen sich durch eine Anpassung der Painter realisieren. Dabei können verschiedene Designs auch parallel verwendet werden.

Features des TabSetControls:

- Umschalten der Taben über Serverroundtrip oder clientseitig (ohne serverroundtrip) konfigurierbar.
- Unterstützt clientseitiges scrolling durch die Taben.
- Erlaubt das Includieren beliebiger JSP-Seiten als Taben.
- Icons vor den Labels auf den Taben einsetzbar.
- Einzelne Taben deaktivierbar.
- Design durch Painterfactory an eigenen StyleGuide (Corporate Identity) anpassbar.
- Verschachtelte TabSets möglich.
- Optimierter HTML-Code.
- Gleiches Look and Feel in Microsoft InternetExplorer > 5.x und Netscape Navigator > 7.x

2 Glossar

CC	Common Controls
JSP	JavaServer Pages
Painter	Erzeugt den HTML-Code eines Kontrollelements; über eine eigene PainterFactory lässt sich das Layout an ein Corporate Design anpassen.
DataModel	Schnittstelle, über die ein Kontrollelement seine Anzeigedaten bezieht.