NetUI Template Tags

Table of contents

1. Introduction

The NetUI template tags provide a very simple templating mechanism. The basic concept is to create a template page that contains the layout and information you want on each page within a site or page flow. Then content pages are created which provide the content and represent the pages of the site. You directly access the content pages and they pull in their template to produce the final rendered document.

2. Template Page

The following JSP page defines a simple template. This template provides the overall layout of the site and places to drop content. There are two types of content `<netui-temp:includeSection>` and `<netui-temp:attribute>`. The `<netui-temp:includeSection>` tag is the primary placeholder for content. The content page will provide sections of content that fit into the placeholders defined in the template. In addition, the `<netui-temp:attribute>` tags is a named attribute that represents some type of value. These may be found in more than one place in the page.

template.jsp

```
<%@ page language="java"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-template-1.0"
prefix="netui-temp" %>
<%@ taglib uri="http://beehive.apache.org/netui/tags-html-1.0"
prefix="netui" %>
<netui:html>
    <head>
        <title><netui-temp:attribute name="title"/></title>
        <netui:base />
    </head>
    <netui:body>
        <table width="70%">
            <caption>
                <span style="font-size:
large;color:blue"><netui-temp:attribute name="title"/></span>
            </caption>
            <tr>
            <td width="200pt" style="border: 1pt solid black">
                <div style="height: 300pt">
                    <netui-temp:includeSection name="left"/>
                </div>
            </td>
            <td valign="top" style="padding:0,10">
                <netui-temp:includeSection name="right"/>
            </td></tr>
        </table>
    </netui:body>
```

```
</netui:html>
```

In the template above there is an `<netui-temp:attribute>` which defines the named attribute `title`. This attribute is used in two places, the first place is the within the HTML `<title>` in the header. The second place this is used is the visual title on the page.

```
<title><netui-temp:attribute name="title"/></title>
...
<caption>
    <span style="font-size: large;color:blue"><netui-temp:attribute
name="title"/></span>
</caption>
```

There are two section defined for this page. These sections will contain the primary content of the page. The `<netui-temp:includeSection>` tag defines the placement of the section and gives a name to it. A content page will define the contents of each section. In the example there are two sections defined `left` and `right`.

```
<td width="200pt" style="border: 1pt solid black">
    <div style="height: 300pt">
        <netui-temp:includeSection name="left"/>
    </div>
</td>
<td valign="top" style="padding:0,10">
    <netui-temp:includeSection name="right"/>
</td></tr>
```

3. Content Page

The content pages provide the content displayed within a template. The content pages also act as the addressable artifacts. This means that index.jsp can be directly accessed and the template will be applied when the content is rendered. Below is a typical content page. This content page will define a set of links which are displayed in the `left` section and some content that is displayed in the `right` section.

The template page may be located anywhere in the site. It does not have to be located in the same location as the content page. The `templatePage` attribute specifies the template page an may be either absolute to the webapp root or relative to content page.

index.jsp

```
<%@ page language="java"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-databinding-1.0"
prefix="netui-data"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-html-1.0"
prefix="netui"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-template-1.0"
prefix="netui-temp" %>
```

```
<netui-temp:template templatePage="template.jsp">
    <netui-temp:setAttribute name="title" value="Basic Template Example"/>
    <netui-temp:section name="left">
        <h4>Links</h4>
        <div style="padding:0,0,0,10pt;color:gray;">
        <netui-data:repeater dataSource="pageFlow.links">
            <netui:anchor
href="http://${container.item.href}">${container.item.name}</netui:anchor><br>
        </netui-data:repeater>
        </div>
    </netui-temp:section>
    <netui-temp:section name="right">
        <p>${pageFlow.content}</p>
    </netui-temp:section>
</netui-temp:template>
```

All of the rendered content inside of a content page appears inside of tags. The `<netui-temp:template>` tag defines the name of the template to apply to the content. In this example the `template.jsp` is the template applied to the content defined.

```
<netui-temp:template templatePage="template.jsp">
```

Next all of the attributes defined inside of the template are defined. The `<netui-temp:setAttribute>` tag provides the value for a named attribute. The `name` and `value` attributes must both be set.

```
<netui-temp:setAttribute name="title" value="Basic Template Example"/>
```

Finally, the content of the two sections is provided. The body of the `<netui-temp:seciton>` provides the content for the named section. You must set the `name` attribute to match one of the sections defined in the template.

```
<netui-temp:section name="left">
    <h4>Links</h4>
    <div style="padding:0,0,0,10pt;color:gray;">
    <netui-data:repeater dataSource="pageFlow.links">
        <netui:anchor
href="http://${container.item.href}">${container.item.name}</netui:anchor><br>
    </netui-data:repeater>
    </div>
</netui-temp:section>
```

4. Page Flow Controller

This example uses a very simple page flow controller that simply provides some read-only properties that are bound to by the content pages. Notice that the begin action goes to the content page `index.jsp`.

Controller.jpf

```
package template;

import org.apache.beehive.netui.pageflow.PageFlowController;
import org.apache.beehive.netui.pageflow.annotations.Jpf;

@Jpf.Controller(
    simpleActions={
        @Jpf.SimpleAction(name="begin", path="index.jsp")
    },
    multipartHandler=Jpf.MultipartHandler.memory
)
public class Controller extends PageFlowController
{
    private Link[] links;

   private String content =
            "<h4>Today's Content</h4>" +
            "<p>This is some content that would be found in a database or
content managment system of some" +
            "type.  For this example it's just hardwired into the page
flow.";

    protected void onCreate()
    {
        links = new Link[5];
        links[0] = new Link("ESPN", "www.espn.com");
        links[1] = new Link("Google", "www.google.com");
        links[2] = new Link("CNN", "www.cnn.com");
        links[3] = new Link("BEA", "www.bea.com");
        links[4] = new Link("gmail", "www.google.com/gmail");
    }

    public Link[] getLinks() {
        return links;
    }
    public String getContent() {
        return content;
    }


    public static class Link {
        private String name;
        private String href;

        public Link(String name, String link)
        {
            this.name = name;
            this.href = link;
        }

        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }

        public String getHref() {
            return href;
        }
        public void setHref(String href) {
            this.href = href;
        }
    }
}
```