Beehive Page Flow Tutorial

Table of contents

1. Introduction

1.1. Tutorial Goals

In this tutorial, you will learn:

- How to create a basic Page Flow web application
- How to coordinate user navigation with Forward methods
- How to handle data submission and processing with data binding and Form Beans
- How to create a user interface with the <neuti> JSP tag library
- How Page Flows help to separate data processing and data presentation

2. Step 1: Begin the Page Flow Tutorial

2.1. To Set up the Development Environment

Complete all of the necessary and optional steps in the following topic: Beehive Installation and Setup (../setup.html)

Open a command shell and confirm that you have the following variables have been set:

- `ANT_HOME`
- `JAVA_HOME`
- `CATALINA_HOME`

Also ensure that the following elements are on your `PATH`:

- `ANT_HOME/bin`
- `JAVA_HOME/bin`

2.2. Make a Project Folder

On your C: drive, create a directory named `beehive_projects`.

Copy the folder `<BeehiveRoot>/samples/netui-blank` into `C:/beehive_projects`. (`<BeehiveRoot` is the top level folder of your Beehive installation; a typical value might be `C:/apache/apache-beehive-1.0`.)

Rename the folder `C:/beehive_projects/`**`netui-blank`** to the name `C:/beehive_projects/`**`pageflow_tutorial`**

Before proceeding, confirm that the following directory structure exists:

```
C:
  beehive_projects
    pageflow_tutorial
```

```
      resources
      WEB-INF
      Controller.jpf
      error.jsp
      index.jsp
```

2.3. Edit the build.properties File

In this section you will edit the `build.properties` file--the file that sets the build-related properties for your web application.

Open the file `C:/beehive_projects/pageflow_tutorial/WEB-INF/src/build.properties` in a text editor.

Edit the `beehive.home` property points to the top-level folder of your beehive installation.

Add the line **`contextPath=pageflow_tutorial`** (as shown below).

For example, if your beehive installation resides at `C:/apache/apache-beehive-1.0`, then your `build.properties` file would appear as follows.

```
beehive.home=C:/apache/apache-beehive-1.0
contextPath=pageflow_tutorial

servlet-api.jar=${os.CATALINA_HOME}/common/lib/servlet-api.jar
jsp-api.jar=${os.CATALINA_HOME}/common/lib/jsp-api.jar
```

2.4. To Start the Tomcat Server

At the command prompt, enter:

```
%CATALINA_HOME%\bin\startup.bat
```

3. Step 2: Create a New Page Flow Application

3.1. Introduction

In this step you will create a Controller file and a JSP page. These are the basic files in a Beehive Page Flow web application. Each Page Flow contains one Controller file and any number of JSP pages. A Controller file is a Java class (with the JPF file extension) that controls how your web application functions and what it does. The methods in the Controller file determines all of the major features of a web application: how users navigate from page to page, how user requests are handled, and how the web application accesses back-end resources. The JSP pages determine what a visitor to the web sees in the browser. (In terms of the Model-View-Controller paradigm for web applications: the Controller.jpf file is the

Controller (naturally), and the JSP pages are the View. This web application's Model is very simple: it consists of two fields that represent the user's age and name.)

Controller files contain Action methods. An Action method may do something simple, such as forward a user from one JSP page to another; or it may do a complex set of tasks, such as receive user input from a JSP page, calculate and/or retrieve other data based on the user input, and forward the user to a JSP page where the results are displayed.

The Controller file you create in this step contains one simple Action method. This simple navigational Action method forwards users to the index.jsp page. In the next step, you will create a more complex Action method.

3.2. To Examine the Controller.jpf and index.jsp Files

There are no edits to make in this step. The point of this step is to learn about the code you are about to run.

Open the file `C:/beehive_projects/pageflow_tutorial/Controller.jpf`. [todo: explain]

`Controller.jpf`

```
import javax.servlet.http.HttpSession;

import org.apache.beehive.netui.pageflow.Forward;
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")
    },
    sharedFlowRefs={
        @Jpf.SharedFlowRef(name="shared", type=shared.SharedFlow.class)
    }
)
public class Controller
    extends PageFlowController
{
    @Jpf.SharedFlowField(name="shared")
    private shared.SharedFlow sharedFlow;

    /**
     * Callback that is invoked when this controller instance is created.
     */
    protected void onCreate()
    {
    }
```

```
    /**
     * Callback that is invoked when this controller instance is destroyed.
     */
    protected void onDestroy(HttpSession session)
    {
    }
}
```

Open the file `C:/beehive_projects/pageflow_tutorial/index.jsp`. [todo: explain]

`index.jsp`

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ 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-template"%>
<netui:html>
  <head>
    <title>Web Application Page</title>
    <netui:base/>
  </head>
  <netui:body>
    <p>
      New Web Application Page
    </p>
  </netui:body>
</netui:html>
```

3.3. To Compile and Deploy the Page Flow

You are now ready to compile the Page Flow and deploy it to Tomcat.

At the command prompt, enter:

```
ant
  -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml
  clean
  build
  war

Copy and Paste version:
ant -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml clean
build war
```

To deploy the application, copy the WAR file into Tomcat's `webapps` directory.

```
copy C:\beehive_projects\pageflow_tutorial.war %CATALINA_HOME%\webapps
```

3.4. To Test the Page Flow Web Application

Visit the following address:

http://localhost:8080/pageflow_tutorial

You will be directed to the `index.jsp` page.

4. Step 3: Navigation

4.1. To Create a Destination JSP Page

In the directory `C:/beehive_projects/pageflow_tutorial`, create a file named `page2.jsp`.

Edit page2.jsp so it appears as follows.

`page2.jsp`

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-html-1.0"
prefix="netui"%>
<netui:html>
  <head>
    <title>page2.jsp</title>
    <netui:base/>
  </head>
  <netui:body>
    <p>
      Welcome to page2.jsp!
    </p>
  </netui:body>
</netui:html>
```

Save `page2.jsp`.

4.2. To Create a Link to the Destination Page

Open the file `C:/beehive_projects/pageflow_tutorial/index.jsp`.

Edit `index.jsp` so it appears as follows. Code to add appears in bold type.

`index.jsp`

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ 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-template"%>
<netui:html>
```

```
  <head>
    <title>Web Application Page</title>
    <netui:base/>
  </head>
  <netui:body>
    <p>
      New Web Application Page
    </p>
    <p>
      <netui:anchor action="toPage2">Link to page2.jsp</netui:anchor>
    </p>
  </netui:body>
</netui:html>
```

Save `index.jsp`.

4.3. To Add a Simple Action to Handle the Link

Open the file `C:/beehive_projects/pageflow_tutorial/Controller.jpf`.

Edit `Controller.jpf` so it appears as follows. Don't forget the comma after the first `Jpf.SimpleAction(...)` element!

`Controller.jpf`

```
import javax.servlet.http.HttpSession;

import org.apache.beehive.netui.pageflow.Forward;
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"),
        @Jpf.SimpleAction(name="toPage2", path="page2.jsp")
    },
    sharedFlowRefs={
        @Jpf.SharedFlowRef(name="shared", type=shared.SharedFlow.class)
    }
)
public class Controller
    extends PageFlowController
{
    ...
}
```

Save `Controller.jpf`.

4.4. To Compile and Deploy the Page Flow

You are now ready to compile the Page Flow and deploy it to Tomcat.

At the command prompt, enter:

```
ant
  -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml
  clean
  build
  war

Copy and Paste version:
ant -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml clean
build war
```

To deploy the application, copy the WAR file into Tomcat's `webapps` directory.

```
copy C:\beehive_projects\pageflow_tutorial.war %CATALINA_HOME%\webapps
```

If you are asked to overwrite the old WAR file, enter 'Yes'.

Wait a few seconds for Tomcat to redeploy the WAR file, then move on to the next step.

4.5. To Test the Page Flow Web Application

Visit the following link:

http://localhost:8080/pageflow_tutorial

You will be directed to the index.jsp page.

Click the link.

You will be directed to page2.jsp.

5. Step 4: Submitting Data

5.1. To Create a Submission Form

Edit the file `C:/beehive_projects/pageflow_tutorial/page2.jsp` so it appears as follows.

`page2.jsp`

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-html-1.0"
prefix="netui"%>
<netui:html>
  <head>
    <title>page2.jsp</title>
    <netui:base/>
  </head>
  <netui:body>
    <p>
```

```
      Welcome to page2.jsp!
    </p>
    <p>
      <netui:form action="processData">
        Name:<netui:textBox dataSource="actionForm.name"/><br/>
        Age:<netui:textBox dataSource="actionForm.age"/><br/>
        <netui:button type="submit">Submit</netui:button>
      </netui:form>
    </p>
  </netui:body>
</netui:html>
```

Save `page2.jsp`.

5.2. To Create a Server Side Representation of the Submission Form (a.k.a. Create a Form Bean)

In this step you will create a Java class that represents the submission form created in the previous task. When the form data is submitted, the Java class will be instantiated, and the form data will be loaded into the members of the Java class.

In the directory `C:/beehive_projects/pageflow_tutorial/WEB-INF/src` create a directory named **`forms`**.

In the directory `C:/beehive_projects/pageflow_tutorial/WEB-INF/src/forms` create a JAVA file named **`ProfileForm.java`**.

Edit `C:/beehive_projects/pageflow_tutorial/WEB-INF/src/forms/ProfileForm.java` so it appears as follows.

`ProfileForm.java`

```
package forms;

public class ProfileForm implements java.io.Serializable
{
    private int age;
    private String name;

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

    public String getName()
    {
        return this.name;
    }
```

```
    public void setAge(int age)
    {
        this.age = age;
    }

    public int getAge()
    {
        return this.age;
    }
}
```

Save and close `ProfileForm.java`.

5.3. To Edit the Controller File to Handle the Submitted Data

Open the file `C:/beehive_projects/pageflow_tutorial/Controller.jpf`

Edit `Controller.jpf` so it appears as follows. Code to add appears in bold type.

`Controller.jpf`

```
import javax.servlet.http.HttpSession;

import org.apache.beehive.netui.pageflow.Forward;
import org.apache.beehive.netui.pageflow.PageFlowController;
import org.apache.beehive.netui.pageflow.annotations.Jpf;
import forms.ProfileForm;

@Jpf.Controller(
    simpleActions={
        @Jpf.SimpleAction(name="begin", path="index.jsp"),
        @Jpf.SimpleAction(name="toPage2", path="page2.jsp")
    },
    sharedFlowRefs={
        @Jpf.SharedFlowRef(name="shared", type=shared.SharedFlow.class)
    }
)
public class Controller
    extends PageFlowController
{
    @Jpf.Action(
        forwards = {
            @Jpf.Forward(name = "success", path = "page2.jsp")
        }
    )
    public Forward processData(ProfileForm form)
    {
        System.out.println("Name: " + form.getName());
        System.out.println("Age: " + form.getAge());
        return new Forward("success");
    }
```

```
    @Jpf.SharedFlowField(name="shared")
    private shared.SharedFlow sharedFlow;

    /**
     * Callback that is invoked when this controller instance is created.
     */
    protected void onCreate()
    {
    }

    /**
     * Callback that is invoked when this controller instance is destroyed.
     */
    protected void onDestroy(HttpSession session)
    {
    }
}
```

Save `Controller.jpf`.

5.4. To Compile and Redeploy the Page Flow

You are now ready to compile the Page Flow and deploy it to Tomcat.

At the command prompt, enter:

```
ant
  -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml
  clean
  build
  war

Copy and Paste version:
ant -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml clean
build war
```

To deploy the application, copy the WAR file into Tomcat's `webapps` directory.

```
copy C:\beehive_projects\pageflow_tutorial.war %CATALINA_HOME%\webapps
```

If you are asked to overwrite the old WAR file, enter 'Yes'.

Wait a few seconds for Tomcat to redeploy the WAR file, then move on to the next step.

5.5. To Test the Page Flow Web Application

Visit the following link:

http://localhost:8080/pageflow_tutorial

You will be directed to the index.jsp page.

Click the link.

You will be directed to page2.jsp.

Enter values in the Name and Age fields, and click Submit.

Notice the name and age values you entered are displayed in the Tomcat console shell.

6. Step 5: Processing and Displaying Data

6.1. To Create a JSP Page to Display Submitted Data

In the directory `C:/pageflow_tutorial` Create a file named **`displayData.jsp`**.

Edit `displayData.jsp` so it appears as follows.

`displayData.jsp`

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ taglib uri="http://beehive.apache.org/netui/tags-html-1.0"
prefix="netui"%>
<netui:html>
  <head>
    <title>displayData.jsp</title>
    <netui:base/>
  </head>
  <netui:body>
    <p>
      You submitted the following information:
    </p>
    <p>
      Name:<netui:content value="${requestScope.data.name}"/><br/>
      Age:<netui:content value="${requestScope.data.age}"/>
    </p>
  </netui:body>
</netui:html>
```

Save and close `displayData.jsp`.

6.2. To Process the Submitted Data

Edit the `processData` method in the `Controller.jpf` file so it appears as follows. Code to add appears in bold.

`Controller.jpf`

```
    ...

    @Jpf.Action(
        forwards = {
            @Jpf.Forward(name = "success", path = "displayData.jsp")
        }
```

```
    )
    public Forward processData(ProfileForm form)
    {
        System.out.println("Name: " + form.getName());
        System.out.println("Age: " + form.getAge());
        getRequest().setAttribute("data", form);
        return new Forward("success");
    }

    ...
```

Save `Controller.jpf`.

6.3. To Compile and Redeploy the Page Flow

You are now ready to compile the Page Flow and deploy it to Tomcat.

At the command prompt, enter:

```
ant
  -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml
  clean
  build
  war

Copy and Paste version:
ant -f C:\beehive_projects\pageflow_tutorial\WEB-INF\src\build.xml clean
build war
```

To deploy the application, copy the WAR file into Tomcat's `webapps` directory.

```
copy C:\beehive_projects\pageflow_tutorial.war %CATALINA_HOME%\webapps
```

If you are asked to overwrite the old WAR file, enter 'Yes'.

Wait a few seconds for Tomcat to redeploy the WAR file, then move on to the next step.

6.4. To Test the Page Flow Web Application

Visit the following link:

http://localhost:8080/pageflow_tutorial

You will be directed to the index.jsp page.

Click the link.

You will be directed to page2.jsp.

Enter values in the Name and Age fields. Click the Submit button.

You will be forwarded to the displayData.jsp page. Notice the values you entered are

displayed.