Beehive Controls Tutorial

Table of contents

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

1. Introduction

1.1. Tutorial Goals

The Controls tutorial is provided as a way to become familiar with the concepts of Controls in Beehive. The tutorial walks through creating, building, and deploying a sample project that uses a Control from a page flow. **Note:** NetUI is not required to use Controls; they just provide a convenient way to demonstrate running a Control. In this tutorial, you will learn:

- What Beehive Controls are for and what they do.
- How to create a Beehive Control interface and implementation.
- How to compile a Beehive Control.
- How to use a Beehive Control as a component in a larger application.
- How to use metadata annotations in Beehive Controls.
- How to create custom metadata annotations.

2. Step 1: Setup a Beehive-enabled Web Application

2.1. Set Shell Variables

Complete all of the necessary and optional steps in the following topic: Beehive Installation and Setup

2.2. Create a Beehive-enabled Web Project

In order to follow the steps in this tutorial, it's necessary to create a Beehive-enabled web application. Beehive-enabled web applications are described here. A skeleton Beehive-enabled web project is provided in the samples/ directory as netui-blank. This contains a basic Ant build file and an example page flow controller. To create the web project, copy and then rename the netui-blank project using these steps:

1. Create a directory `/beehive_projects` (on Windows, this would be `C:\beehive_projects`).
2. Run the Ant target to create a new NetUI project: `ant -f <beehive-root>/beehive-imports.xml new.netui.webapp` and provide a fully-qualified web project root directory named `controls_tutorial` when prompted. Note, `<beehive-root>` is the directory that contains a Beehive distribution; a typical value might be `/apache/apache-beehive-1.0`.
3. Before continuing, confirm that the following directory structure exists:

```
beehive_projects/
```

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

```
        controls_tutorial/
            web/
                Controller.java
                index.jsp
                resources/
                WEB-INF/
            build.properties
            build.xml
```

Note, this directory structure is just an example; you are free to put the `controls_tutorial` directory anywhere on disk. In the remainder of this tutorial, the directory `beehive_projects/controls_tutorial` will simply be referred to as `controls_tutorial`.

2.3. Configure Build Properties

The `build.properties` file contains several project-related properties that must be set in order to build the web application. Specifically, the paths to your Beehive distribution and to the JSP / Servlet API JARs for your application container must be set. The following steps will set these properties for the `controls_tutorial` webapp.

1. Open the file `controls_tutorial/build.properties` in a text editor.
2. Edit the `beehive.home` property so it points to the top-level folder of your Beehive distribution.
3. Edit the `context.path` to use the value `controls_tutorial`.

Note:

The `context.path` property determines both (1) the name of the application WAR file and (2) the application URL.
If `context.path=controls_tutorial`, then the following WAR file will be produced:
`controls_tutorial``.war`
and the following URL will invoke the web application:
`http://someapplicationserver/`**`controls_tutorial`**

For example, if your Beehive distribution is located in `/apache/apache-beehive-1.0`, then your `build.properties` file would appear as follows.

```
beehive.home=/apache/apache-beehive-1.0
servlet-api.jar=${os.CATALINA_HOME}/common/lib/servlet-api.jar
jsp-api.jar=${os.CATALINA_HOME}/common/lib/jsp-api.jar

context.path=controls_tutorial
```

Note:

Properties files should use the '/' character to separate drive, directory, and file names.

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

If you are using an application container other than Tomcat, be sure to set the `servlet-api.jar` and `jsp-api.jar` properties to reference the JAR your server provides which contains the JSP and Servlet API classes.

2.4. Add Sample Control Interface and Implementation Source Files

This tutorial uses a sample Hello World control that is contained in **`<BeehiveDistribution>/samples/controls-blank/src/pkg`**. Copy the `pkg` sub-directory from this path into the directory **`controls_tutorial/src`**.

Now, confirm that in addition to the files described above that the following directories and files exist:

```
controls_tutorial/
    src/
        pkg/
            Hello.java
            HelloImpl.java
```

2.5. Start the Server

If you are using Tomcat, enter the following at the command prompt:

```
$CATALINA_HOME/bin/startup.bat
```

If you aren't using Tomcat, start your application container as per its directions.

2.6. Using URLs in the Examples

In the Beehive tutorials, you will often encounter URLs like `http://localhost:8080/controls_tutorial/begin.do`. When you see these URLs, they are meant to be accessed via your web browser to demonstrate functionality built in the tutorial. These URLs are setup to run on Tomcat, so if you are unable to run these URLs, be sure that they are appropriate for your application server. Specifically, check the port numbers to make sure that your server is running on the referenced port.

3. Step 2: Compile the Control Implementation and Interface Files

3.1. Introduction

A Beehive Control consists of two files: an interface file and an implementation file. The interface file is the public face of your control. It lists all of the methods which can be invoked by users. The implementation file contains the implementation code for the methods listed in the interface file.

© 2005, Apache Software Foundation

Copyright © 2004 The Apache Software Foundation. All rights reserved.

3.2. Examine the Control Files

Open the file `/beehive_projects/controls_tutorial/WEB-INF/src/pkg/HelloImpl.java`.

The implementation file appears as follows. (There is no need to edit the file at this point in the tutorial.)

```
package pkg;

import org.apache.beehive.controls.api.bean.ControlImplementation;

@ControlImplementation(isTransient=true)
public class HelloImpl
    implements Hello {

    public String hello() {
        return "hello!";
    }
}
```

Open the file `controls_tutorial/WEB-INF/src/pkg/Hello.java`. The interface file should appear appears as follows.

```
package pkg;

import org.apache.beehive.controls.api.bean.ControlInterface;

@ControlInterface
public interface Hello {
    String hello();
}
```

3.3. Edit the Controller.java File

To test the Hello control, you need to call the control from some other resource, such as a Java application, JSP, a web application, etc. In the following two steps you will call the control from a page flow contoller in a web application and display the results on a JSP.

Open the file `controls_tutorial/Controller.java` in a code editor and modify so it appears as:

```
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;
```

© 2005, Apache Software Foundation

Copyright © 2004 The Apache Software Foundation. All rights reserved.

```
import org.apache.beehive.controls.api.bean.Control;
import pkg.Hello;

@Jpf.Controller(
    simpleActions={
        @Jpf.SimpleAction(name="old_begin", path="index.jsp")
    },
    sharedFlowRefs={
        @Jpf.SharedFlowRef(name="shared", type=shared.SharedFlow.class)
    }
)
public class Controller
    extends PageFlowController {

    @Control
    private Hello _helloControl;

    @Jpf.Action(
      forwards={
        @Jpf.Forward(name="success", path="index.jsp")
      }
    )
    protected Forward begin()
        throws Exception {
        Forward f = new Forward("success");
        f.addActionOutput("helloMessage", _helloControl.hello());
        return f;
    }

}
```

The two import statements import the @Control annotation and the `Hello` control, respectively.

The `@Jpf.SimpleAction` named `begin` is renamed to `old_begin` because the Beehive runtime automatically looks for and runs any action named `begin` when a page flow is first instantiated. The renaming makes the runtime run the action method `begin()`, which allows us to call the `Hello` control inside of the method body.

3.4. Edit the index.jsp Test Page

Edit the file `/beehive_projects/controls_tutorial/index.jsp` to appear as follows; changes are shown in **bold**:

```
<%@ 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"%>
```

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

```
<netui:html>
  <head>
    <title>Control Tutorial Test Page</title>
    <netui:base/>
  </head>
  <netui:body>
    <h3>Control Tutorial Test Page</h3>
    <p>
        Response from the hello() method on the Hello Control:
        <netui:span style="color:#FF0000"
value="${pageInput.helloMessage}"/>
    </p>
  </netui:body>
</netui:html>
```

3.5. Compile and Deploy the Web Application

You are now ready to compile the page flow and deploy it to Tomcat.

The following Ant command assumes that you are in the `controls_tutorial` directory. At the command prompt, enter:

```
ant clean build war
```

This will build the webapp by running the Beehive annotation processors and will produce class files in `WEB-INF/classes`. Now, the application is ready to deploy to your server. On Tomcat, copy the WAR file into Tomcat's `$CATALINA_HOME/webapps` directory.

On Windows:

```
    copy controls_tutorial.war %CATALINA_HOME%\webapps /Y
```

Everywhere else:

```
    cp controls_tutorial.war $CATALINA_HOME/webapps
```

If you are asked to overwrite the old WAR file, enter 'yes'. Note, during redeployment, you may have to wait a few seconds for Tomcat to redeploy the WAR file. Once deployment or redeployment has completed, the webapp can be accessed through a browser.

If you are not using Tomcat, follow your server's web application deployment instructions to deploy the webapp.

3.6. Test the Control

Visit the following address in a web browser:

```
http://localhost:8080/controls_tutorial/begin.do
```

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

This will render the `index.jsp` page.

Note the message on the page: "hello!" which is provided by `Hello` control.

4. Step 3: Add a Parameterized Method to the Control

4.1. Edit the Interface and Implementation Files

Edit `controls_tutorial/WEB-INF/src/pkg/HelloImpl.java` so it appears as follows. Code to add appears in bold.

```
package pkg;

import org.apache.beehive.controls.api.bean.ControlImplementation;

@ControlImplementation(isTransient=true)
public class HelloImpl
    implements Hello {

    public String hello() {
        return "hello!";
    }

    public String helloParam(String name) {
        return "Hello, " + name + "!";
    }

}
```

Edit `controls_tutorial/WEB-INF/src/pkg/Hello.java` so it appears as follows; modifications are shown in **bold**:

```
package pkg;

import org.apache.beehive.controls.api.bean.ControlInterface;

@ControlInterface
public interface Hello {

    String hello();

    String helloParam(String name);
}
```

4.2. Edit the Controller.java File

Edit the `begin()` method in `Controller.java` so it appears as follows; modifications are shown in **bold**.

© 2005, Apache Software Foundation

Copyright © 2004 The Apache Software Foundation. All rights reserved.

```
...

public class Controller
    extends PageFlowController {

    ...

    @Jpf.Action(
      forwards={
        @Jpf.Forward( name="success", path="index.jsp" )
      }
    )
    protected Forward begin()
        throws Exception {
        Forward f = new Forward("success");
        f.addActionOutput("helloMessage", _helloControl.hello());
        f.addActionOutput("helloParamMessage",
_helloControl.helloParam("World"));
        return f;
    }

   ...

}
```

4.3. Edit a Test JSP

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

```
<%@ 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>Control Tutorial Test Page</title>
    <netui:base/>
  </head>
  <netui:body>
    <jsp:useBean class="pkg.HelloBean" id="helloBean" scope="session"/>
    <h3>Control Tutorial Test Page</h3>
    <p>
        Response from the hello() method on the Hello Control:
        <netui:span style="color:#FF0000"
value="${pageInput.helloMessage}"/>
    </p>
    <p>
        Response from the helloParam() method on the Hello Control:
        <netui:span style="color:#FF0000"
```

© 2005, Apache Software Foundation

Copyright © 2004 The Apache Software Foundation. All rights reserved.

```
value="${pageInput.helloParamMessage}"/>
    </p>
  </netui:body>
</netui:html>
```

4.4. Compile and Redeploy the Web Application

Compile and deploy the web application using the Ant commands described here.

4.5. Test the Control

Visit the following address in a web browser:

```
http://localhost:8080/controls_tutorial/begin.do
```

This will render the `index.jsp` page.

Note the messages on the page: "hello!" and "Hello, World!" which are provided by the `Hello` control.

Java, J2EE, and JCP are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

© 2005, Apache Software Foundation