Building a Page Flow Web App

Table of contents

1. Introduction

Now that the theory of Java Page Flows has been explained, you need to know how to concretely put together a web-app that uses JPFs. Beehive ships with a set of Ant buildfiles to make the building of an application much easier.

2. Source Tree Layout

The layout of your project may be anywhere on your local disk. We assume that the environment variable $WEBAPP_DIR points to the top-level of your application.

- `$WEBAPP_DIR/`
  - `Controller.jpf`
  - `index.jsp`
  - `login.jsp`
  - `signup.jsp`
  - `mypage.jsp`
  - `resources/`
  - `WEB-INF/`
    - `classes/`
    - `lib/`
    - `src/`

2.1. $WEBAPP_DIR/

The top-level of the web-app, at `$WEBAPP_DIR` should contain your JSP pages and a `Controller.jpf`. When built, the `Controller.jpf` will be compiled to `WEB-INF/classes/Controller.class`.

2.2. $WEBAPP_DIR/WEB-INF/

The `$WEBAPP_DIR/WEB-INF/` directory is just as it is with any other servlet-based application.

2.3. $WEBAPP_DIR/WEB-INF/src/

The `$WEBAPP_DIR/WEB-INF/src/` directory contains any other application source files that need to be compiled into Java classes. They will be compiled to `$WEBAPP_DIR/WEB-INF/classes/`. Additionally, any extra `.properties` or `.xml` files that need to be deployed with your application will be copied from the `src/` directory to the `classes/` directory during the build.

2.4. $WEBAPP_DIR/WEB-INF/lib/

As with any other web application, the `$WEBAPP_DIR/WEB-INF/lib/` directory should contain the jars for each dependency of your application, including those required by Beehive itself. The jars required by Beehive are listed below.

Note: this Ant command will deploy the necessary JARs listed below (plus a few other optional control JARs) to the WEB-INF/lib directory.

```
ant -f $BEEHIVE_HOME\ant\webappRuntimeCore.xml -Dwebapp.dir=$WEBAPP_DIR
deploy.beehive.webapp.runtime
```

Project	Jar	Version
XMLBeans	apache-xbean.jar	2.0.0
Axis	axis.jar	1.2RC2
Axis	axis-ant.jar	1.2RC2
Beehive NetUI	beehive-netui-pageflow.jar	*distribution*
Beehive NetUI	beehive-netuid-scoping.jar	*distribution*
Beehive NetUI	beehive-netui-tags-databinding.jar	*distribution*
Beehive NetUI	beehive-netui-tags-html.jar	*distribution*
Beehive NetUI	beehive-netui-tags-template.jar	*distribution*
Beehive NetUI	beehive-netui-util.jar	*distribution*
Jakarta Commons Bean Utils	commons-beanutils	1.6
Jakarta Commons Codec	commons-codec-1.3.jar	1.3
Jakarta Commons Collections	commons-collections.jar	2.1.1
Jakarta Commons Digester	commons-digester.jar	1.5
Jakarta Commons Discovery	commons-discovery-0.2.jar	0.2
Jakarta Commons Discovery	commons-discovery.jar	0.2-dev
Jakarta Commons EL	commons-el.jar	1.0
Jakarta Commons Servlet File Upload	commons-fileupload.jar	1.0
Jakarta Commons Logging	commons-logging.jar	1.0.3
Jakarta Commons Validator	commons-validator.jar	1.1.3

Beehive Controls	controls.jar	*distribution*
Jakarta Commons ORO (Text Processing)	jakarta-oro.jar	2.0.8
JAX RPC	jaxrpc.jar	1.1
JSR 173 (Streaming API for XML)	jsr173_1.0_api.jar	1.0
JSTL	jstl.jar	1.1.0-D13
Log4J	log4j-1.2.8.jar	1.2.8
SAAJ	saaj.jar	1.2
JSTL	standard.jar	1.1.0-D13
Struts	struts.jar	1.2.4
Velocity	velocity-1.4.jar	1.4
Velocity	velocity-dep-1.4.jar	1.4
WSDL4J	wsdl4j.jar	1.5
Beehive Web Services	wsdltypes.jar	*distribution*
Beehive Web Services	wsm-axis.jar	*distribution*
Beehive Web Services	wsm.jar	*distribution*

3. Running Ant

Before you can build the web-app using ant, you must ensure that the `BEEHIVE_HOME` and `CATALINA_HOME` variables are set correctly, along with `WEBAPP_DIR`.

Variable	Value
BEEHIVE_HOME	Top level of the Beehive distribution
CATALINE_HOME	Top level of the installed Tomcat server
WEBAPP_DIR	Top level of the web-app to be built

Once these variables are set correctly, building the deployable web-app requires a single ant invocation using the build file at `$BEEHIVE_HOME/ant/buildWebapp.xml`. This build-file is invoked using ant's `-f <buildfile-path>` option. The directory of the web-app is passed on the commandline using the `-Dname=value` functionality of ant.

Finally, the `build.webapp` target is invoked.

```
ant -f $BEEHIVE_HOME/ant/buildWebapp.xml -Dwebapp.dir=$WEBAPP_DIR build
```

4. Deploying the Web-app

The easiest way to deploy the web-app is to create a symlink/shortcut from $WEBAPP_DIR to $CATALINA_HOME/webapps.

```
ln -s $WEBAPP_DIR $CATALINE_HOME/webapps
```

Another way would be to simply copy $WEBAPP_DIR to $CATALINE_HOME/webapps.

```
cp -R $WEBAPP_DIR $CATALINE_HOME/webapps
```

Another way is to visit the following link in a browser. This method assumes that you have created the manager role in the configuration file `CATALINA_HOME/conf/tomcat-users.xml`. For details, see Installation and Setup (../setup.html) . <**Context-Path**> is the desired URL path to the application.
<**Full-Path-to-Development-Dir**> is the location of the application on your machine.

```
http://localhost:8080/manager/deploy?path=<Context-Path>&war=<Full-Path-to-Development-
```

5. Next...

Now that you've built and deployed an application, you can see how easy it is to modify the flow between pages, adding and removing pages.

- Altering a Page Flow (pageflow_altering.html)