NetUI Project Template

Table of contents

1. Introduction

NetUI enabled web applications require a set of JARs, XML configuration files, and `web.xml` entries. In order to make it easy to get started with a NetUI web application, the `samples/netui-blank` project template is available in the Beehive distribution. Use this as a starting point for building new NetUI-enabled applications. This document describes how to build a new NetUI-enabled application from the `netui-blank` project template.

Note:

In the documentation below, <beehive-root> refers to the top-level directory of your local Beehive installation. A typical value for <beehive-root> would be `/apache/apache-beehive-1.0`.

2. Using the NetUI Webapp Template

The following instruction assume that you have completed all of required and optional steps in the Beehive set up procedure at Installation and Setup.

To use the `netui-blank` template, follow these steps:

2.1. Create a New Webapp

A new NetUI web project can be created by running the following Ant command from `<beehive-root`:

```
ant -f beehive-imports.xml new.netui.webapp
```

This will prompt you for a fully-qualified path for your web project. For example, the following path `/projects/fooWeb` would create a project layout similar to that described here. This project will include the runtime, configuration files / entries, and a basic Ant build.xml file. The description below assumes that you have created the NetUI-enabled web project `fooWeb` under a `projects/` directory.

2.2. Configure the Webapp's Build Properties

In this section you will edit the `build.properties` file in order to set build-related paths and values for your web project.

Open the file `/projects/fooWeb/build.properties` in a text editor and edit the following properties:

- Set the `beehive.home` property to point to the top-level folder of your Beehive installation.
- Set the `context.path` property to some value that is appropriate to your web

application. This value is used when the web-project is built into a `.war` archive. When deploying the `.war` archive, it also often determines thecontext path for the web project when deployed to the server.

For example, if Beehive is installed at `/apache/apache-beehive-1.0`, and you created a project named `fooWeb` then your `build.properties` file might 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=fooWeb
```

Note:

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

Note:

This `.properties` file also assumes that you are using Tomcat as your Servlet container. If you are not using Tomcat, set the paths to the Servlet and JSP API classes appropriately for your container. In some cases, these may point to the same JAR.

2.3. Build the Webapp

By default, the build associated with a new NetUI-enabled webapp stores the Ant `build.xml` file in the project's root directory; in the example above, this is the `projects/fooWeb` directory.

To build the webapp, run `ant build`. This will create a `build/` directory into which the webapp's resources are copied and Java source files are compiled. This `build` directory will be archived when creating a `.war` file. It can also be deployed, exploded directly to the server.

To clean the webapp, run `ant clean`.

To create a `.war` file, run `ant war`.

To update the JSPs when doing iterative development, run `ant copy.jsps`.

2.4. Deploy and Run the Web App

If you are using Tomcat, the web application can be deployed by copying the built `.war` file from `projects/fooWeb/fooWeb.war` to Tomcat's `webapps` directory.

If you are using a different Servlet container, follow your container's deployment instructions for deploying a `.war` web projct arcive.

Once the web application is deployed, you can run your web application by accessing the following URL from a browser: http://localhost:8080/<**fooWeb**/