JBoss to Geronimo - Servlets and JSPs Migration

This article will help you migrate servlets and JSPs deployed JBoss v4 to Apache Geronimo. This article is part of a series of migration articles covering different types of applications migration.

This article covers the migration one of the most fundamental aspects of J2EE: servlets and JSPs. The sample application used for this migration exercise is College Fest which just contains servlets and JSPs for handling the flow of control. The College Fest sample application does not use any database, for details on migrating JDBC applications refer to the JBoss to Geronimo - JDBC Migration article.

After reading this article you should be able to reconfigure the JBoss build files and the deployment descriptors to set up the Apache Geronimo destination environment and then deploying simple Web applications.

This article is organized in the following sections:

Servlets and JSPs implementation analysis

Servlets and JSPs implementations may vary from one application server to another. The purpose of this section is to provide servlets and JSPs specific feature-to-feature comparison between JBoss and Apache Geronimo so you can clearly identify the differences and plan accordingly before migration.

Apache Geronimo includes a Web application container supporting J2EE Web applications. The Web container itself supports basic configuration such as network ports and SSL options, and each Web application may include Geronimo-specific configuration information as well. Web applications participate in the Geronimo security infrastructure, so authenticating to a Web application allows access to secure EJBs and Connectors as well.

Apache Geronimo currently supports two Web containers: **Jetty** and **Tomcat**.

Jetty

Jetty is a 100% Java HTTP Server and Servlet Container. This means that you do not need to configure and run a separate Web server in order to use servlets and JSPs to generate dynamic content. Jetty is a fully featured Web server for static and dynamic content.

Unlike separate server/container solutions, Jetty's Web server and Web application run in the same process without interconnection overheads and complications. Furthermore, as a pure java component, Jetty can be easily included in your application for demonstration, distribution or deployment. Jetty is available on all Java supported platforms.

In Geronimo, you need to explicitly configure the pathways used by browsers attempting to connect to the Web container. In the case of Jetty, the default Web container, these pathways are known as connectors. The standard configuration includes a Jetty connector supporting HTTP on port 8080, as well as a HTTPS connector on port 8443.

Each Jetty connector is a GBean, so the process of configuring a Jetty connector involves configuring a GBean.

Apache Tomcat

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies.

http://java.sun.com/products/servlet
http://java.sun.com/products/jsp

The differences

JBoss v4 supports only Tomcat 5.5, which is the default Web container. The embedded Tomcat service is the expanded SAR **jbossweb-tomcat55.sar** in the deploy directory. The `web.xml` file that provides a default configuration set for Web application is also found in this exapanded SAR directory structure.

The HTTP connector is set up on port 8080 and port 8009 is used if you want to connect via a separate Web server such as Apache HTTP.

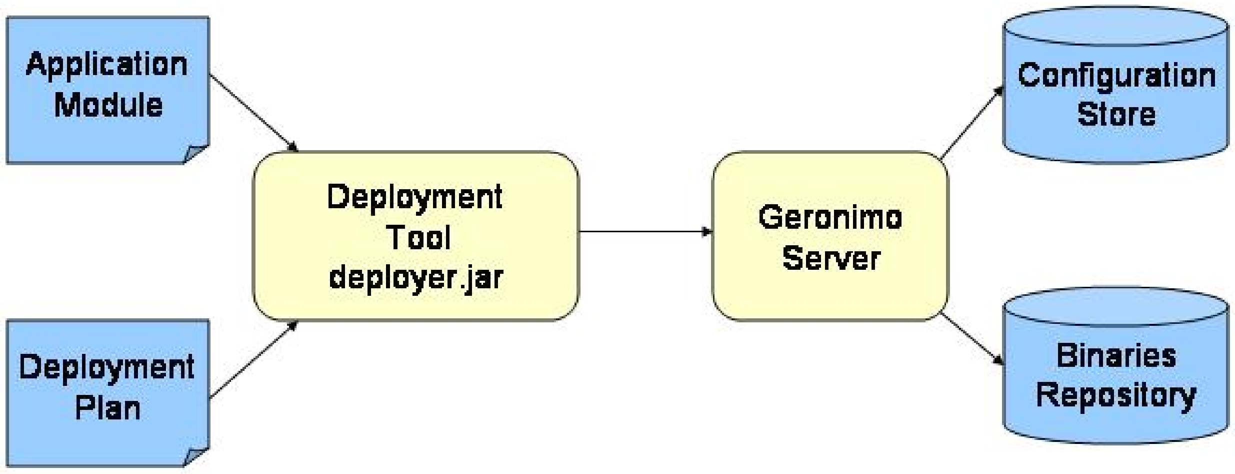
In addition to the default Web container, the second major difference lies in the deployment plan. A deployment plan in Geronimo is similar to a J2EE deployment descriptor in the sense that it is an XML file that contains the configuration information for a specific application module or service.

For very simple Web applications with no security nor resource references, a Geronimo deployment plan is not required, a default context root and dependencies will be provided automatically at deployment time.

The Geronimo Web application deployment plan is **geronimo-web.xml**. The corresponding deployment descriptor in Jboss is `jboss-web.xml`. For further details on Geronimo deployment architecture you may refer to the Understand Geronimo's deployment architecture.

Another difference with servlets and JSPs lies in the way the Web application is deployed. In Geronimo, the application package (ear, war, rar or jar) is deployed using the deployment tool `deployer.jar` located in the `<geronimo_home>/bin` directory.

The `deployer.jar` deploys the application module based on the information provided in the deployment plan (if a plan is provided) to the Geronimo server. The server then saves the metadata to a configuration store and the executables to a binary repository. The following figure illustrates the behavior of the deployment tool.



In JBoss, a Web application is deployed by simply copying the application package (ear, war, rar or jar) into the `<jboss_home>/server/<your_server_name>/deploy` directory from where the server detects its presence and deploys it accordingly.

The following table summarizes the differences between JBoss and Geronimo.

Feature	JBoss v4	Apache Geronimo
Deployment descriptor /plan	jboss.xml	geronimo-web.xml
Method of deployment	Copy the package (ear, war, rar or jar) to the deploy folder of the JBoss server `<jboss_home>/server/<your_server_name>/deploy`	Deployer tool available in server's bin directory `<geronimo_home>/bin`. Deployment is also available through the administration console. A third option is Hot Deployment, which would be the equivalent to JBoss functionality.
Web container	Apache Tomcat 5.5	Jetty and/or Apache Tomcat

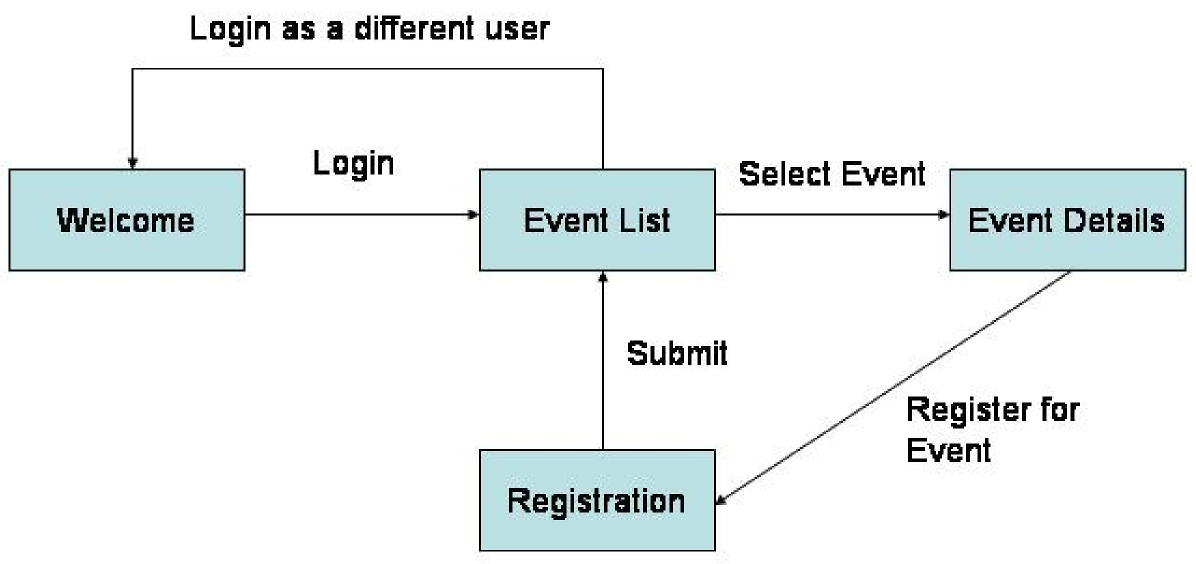
Back to Top

Sample application

The College Fest application handles registration for events at a college festival. This is a very simple application that does not use any type of database. The College Fest application has the following four pages:

- Welcome page
- Event List page
- Event Details page
- Registration page

The following figure illustrates the application flow:



The user access the Welcome page and enters user name and college. From there the user can see the list of available events. The user can access the details for each Event by clicking them from the list. From the Event details page the user can register for that particular event.

Application classes and JSP pages

The College Fest sample application consists of the following two servlets:

- WelcomeServlet - Handles user login and then grabs the user name and dispatches this to the next servlet.
- PersonalServlet - Personalizes the page for the user and hands over control to welcome.jsp.

The College Fest sample application also includes the following JSP pages:

- welcome.jsp - Displays the events list to the user so that s/he can choose on what event to register.
- dc.jsp - Displays the details for the Dumb Charades event.
- pp.jsp - Displays the details for the Pot Potpourri event.
- wtgw.jsp - Displays the details for the What's The Good Word event.
- gq.jsp - Displays the details for the General Quiz event.
- team_reg.jsp - Handles the user registration for one event.

Tools used

The tools used for developing and building the College Fest sample application are:

Eclipse

The Eclipse IDE was used for development of the sample application. This is a very powerful and popular open source development tool. Integration plug-ins are available for both JBoss and Geronimo. Eclipse can be downloaded from the following URL:
http://www.eclipse.org

Apache Ant

Ant is a pure Java build tool. It is used for building the war files for the College Fest application. Ant can be downloaded from the following URL:
http://ant.apache.org

Back to Top

The JBoss environment

This section shows you how and where the sample JBoss reference environment was installed so you can map this scenario to your own implementation.

Detailed instructions for installing, configuring, and managing JBoss are provided in the product documentation. Check the product Web site for the most updated documents.

The following list highlights the general tasks you will need to complete to install and configure the initial environment as the starting point for deploying the sample application.

1. Download and install JBoss v4 as explained in the product documentation guides. From now on the installation directory will be referred as `<jboss_home>`.
2. Create a copy of the default JBoss v4 application server. Copy recursively `<jboss_home>\server\default` to `<jboss_home>\server\<your_server_name>`.
3. Start the new server by running the `run.sh -c <your_server_name>` command from the `<jboss_home>\bin` directory.
4. Once the server is started, you can verify that it is running by opening a Web browser and pointing it to this URL: http://localhost:8080. You should see the JBoss Welcome window and be able to access the JBoss console.
5. Once the application server is up and running, the next step is to install and configure all the remaining prerequisite software required by the sample application. This step is described in the following section.

As mentioned before, Apache Ant is used to build the binaries for the College Fest application. If you do not have Ant installed this is a good time for doing it and make sure that **<ant_home>\bin** directory is added to the system's path variable.

Apache Ant can be downloaded from the following URL:

http://ant.apache.org

Build the sample application

The College Fest application included with this article provides an Ant script that you will use in order to build the application. Download the College Fest application from the following link:

College Fest

After extracting the zip file, a college_fest directory is created. In that directory open the **build.properties** file and edit the properties to match your environment as shown in the following example:

Update the build.properties file

```
#Replace java.home with your jdk directory
java.home=<JAVA_HOME>
#Replace j2ee.home with the parent directory of lib/j2ee.jar
j2ee.home=<jboss_home>/server/<your_server_name>
#Replace with jboss home directory
jboss.server=<jboss_home>/server/<your_server_name>
#geronimo home directory
geronimo.home=<geronimo_home>
```

In the college_fest directory you can also find two build files, `build.xml` and `jboss-build.xml`. `build.xml` is the default build file so if you just type **an** this file will be used for building the application. For this particular sample application the `jboss-build.xml` file is provided.

From a command line, still within the college_fest directory type the following command:

```
ant -f jboss-build.xml clean deploy
```

With this command, ant will use the targets defined in the `jboss-build.xml` file to build the College Fest application and deploy it to the JBoss server. Take a special look at **<target name="deploy" ...>**, here is where the `jboss-build.xml` tell ant where to deploy the WAR file. The following example shows the definitions in the `jboss-build.xml` file.

jboss-build.xml

```
<?xml version="1.0"?>
<!-- build file for building a war -->

<project name="build" default="war" basedir=".">

    <property file="build.properties"/>
    <property name="src.dir" value="src"/>
    <property name="dest.dir" value="bin"/>

    <target name="clean" description="Delete all generated files">
        <echo message="Deleting bin directory" />
        <delete dir="bin" />
    </target>

        <target name="compile">
                <mkdir dir="${dest.dir}"/>
                <javac srcdir="${src.dir}" destdir="${dest.dir}">
                        <classpath path="${java.home}/lib/tools.jar"/>
                        <classpath path="${j2ee.home}/lib/j2ee.jar"/>
                </javac>
        </target>

        <target name="war" depends="compile">
                <war destfile="college_fest.war" webxml="WEB-INF/web.xml">
                        <zipfileset dir="jsp" prefix="jsp"/>
                        <zipfileset dir="pix" prefix="pix"/>
                        <classes dir="${dest.dir}"/>
                        <webinf dir="WEB-INF" />
                </war>
        </target>

        <target name="deploy" depends="war">
                <copy file="college_fest.war" todir="${jboss.server}/deploy"/>
        </target>

        <target name="undeploy">
                <delete file="${jboss.server}/deploy/college_fest.war"/>
    </target>
</project>
```

The war created by the ant build contains a JBoss specific deployment descriptor, the `jboss-web.xml` file in the `WEB-INF` directory of the WAR is shown in the following example.

JBoss deployment descriptor

```
<?xml version="1.0" encoding="UTF-8"?>

<jboss-web>

    <context-root>college_fest</context-root>
    <context-priority-classloader>
      false
    </context-priority-classloader>
</jboss-web>
```

Back to Top

Deploy the sample application

The previous step showed how to deploy the application at build time by specifying a customized `jboss-build.xml` file. If you used the default `build.xml` file at build time you still need to deploy the College Fest application manually. To deploy the College Fest application in JBoss, copy the `college_fest.war` file you just built with Ant to the following directory:

```
<jboss_home>\server\<your_server_name>\deploy
```

If JBoss is already started, it will automatically deploy and start the application; otherwise, the application will be deployed and started at the next startup.

Back to Top

Testing the application

To test the application, open a Web browser and access the following URL:

http://localhost:8080/college_fest

You should see the Welcome screen where you can login with your name and college. When you enter your name a college and click Submit you will see a message at the end on the page stating your name with a link to "Click here" to enter the site. Browse the site and check the options, at this point the College Fest application is configured and running.

Back to Top

The Geronimo environment

Download and install Geronimo from the following URL:

http://geronimo.apache.org/downloads.html

The release notes available there provide clear instructions on system requirements and how to install and start Geronimo. Throughout the rest of this article we will refer to the Geronimo installation directory as **<geronimo_home>**.

TCP/IP ports conflict

If you are planning to run JBoss and Geronimo on the same machine consider to change the default service ports on, at least, one of these servers.

Back to Top

Step-by-step migration

In order to migrate the College Fest application to Geronimo you need to replace the `jboss-web.xml` file with a `geronimo-web.xml` file which is the Geronimo specific descriptor file. The geronimo-web.xml file is located in the WEB-INF directory withing the college_fest directory structure. The Geronimo deployment plan geronimo-web.xml is illustrated in the following example.

The following geronimo-web.xml was generated using the J2G tool.

Geronimo specific deployment plan geronimo-web.xml

```
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"
         xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1"
         xmlns:security="http://geronimo.apache.org/xml/ns/security-1.1"
         xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1">
  <sys:environment>
    <sys:moduleId>
      <sys:groupId>j2g</sys:groupId>
      <sys:artifactId>web-module</sys:artifactId>
      <sys:version>1.0</sys:version>
      <sys:type>war</sys:type>
    </sys:moduleId>
    <sys:dependencies/>
  </sys:environment>
  <context-root>college_fest</context-root>
</web-app>
```

Given that College Fest is a very simple application, the Geronimo deployment plan will also be very simple. Remember that this application does not use any database access nor has security configured. While reading other articles in the **JBoss to Geronimo** migration series, you will notice how the complexity of the deployment plan increases as the sample applications for the different migration scenarios also grow in complexity.

Earlier in this article it was discussed the behavior of the deployment tool. During the deployment process, you provide to the deployment tool the application module and a deployment plan. The College Fest sample application is so simple that you may choose not to provide any deployment plan and let Geronimo do the deployment with a default set of values (a default context root for example).

Last time you built the College Fest sample application it was configured for ant to use the `jboss-build.xml` file instead of the default `build.xml`. The following example shows the content of the default `build.xml` file.

build.xml

```
<?xml version="1.0"?>
<!-- build file for building a war -->

<project name="build" default="war" basedir=".">

    <property file="build.properties"/>
    <property name="src.dir" value="src"/>
    <property name="dest.dir" value="bin"/>


    <target name="clean" description="Delete all generated files.">
        <echo message="Deleting bin folder" />
        <delete dir="bin"/>
    </target>

      <target name="compile">
            <mkdir dir="${dest.dir}"/>

        <javac srcdir="${src.dir}" destdir="${dest.dir}">
            <classpath path="${java.home}/lib/tools.jar"/>
            <!--classpath path="${j2ee.home}/lib/j2ee.jar"/-->
            <classpath path="${geronimo.home}/repository/org/apache/geronimo/specs/geronimo-servlet_2.5_spec/1.1
/geronimo-servlet_2.5_spec-1.1.jar" />
        </javac>
    </target>

    <target name="war" depends="compile">
        <war destfile="college_fest.war" webxml="WEB-INF/web.xml">
            <zipfileset dir="jsp" prefix="jsp"/>
            <zipfileset dir="pix" prefix="pix"/>
            <classes dir="${dest.dir}"/>
            <webinf dir="WEB-INF" />
        </war>
</target>


</project>
```

To build the migrated application run **ant** from the command line without specifying any additional parameters, a college_fest.war file will be created in the root directory of the College Fest application directory structure.

To delete the generated files for a clean build run **ant clean** followed by an **ant.**

Back to Top

Deploy the migrated sample application

To deploy the migrated College Fest application, make sure the Geronimo server is up and running.

From a command line, change directory to `<geronimo_home>/bin` and type the following command:

`deploy --user system --password manager deploy <college_fest_home>/college_fest.war`

Once the application is deployed, open a Web browser and access the following URL:

http://localhost:8080/college_fest/

Repeat the steps you did when Testing the application on the JBoss environment.

Back to Top

Summary

This article has shown you how to migrate a simple Servlet and JSPs application, from JBoss to the Apache Geronimo application server. You followed step-by-step instructions to build the application, deploy and run it, and then migrate it to the Geronimo environment.

Some remarks after reading this article:

- Apache Geronimo provides two different Web containers, Jetty and Apache Tomcat.
- You learnt how to use the deployer too for deploying an application in Geronimo.
- Not always a Geronimo deployment plan is needed, if the application does not use resource references you can accept the deployment defaults from Geronimo.