Space Details

Key:	PentahoDoc
Name:	BI Server Documentation - Latest
Description:	Latest version of the Pentaho BI Server
Creator (Creation Date):	admin (Nov 15, 2006)
Last Modifier (Mod. Date):	admin (Nov 27, 2006)

Available Pages

- Manual Deployment of Pentaho
  - 01. Before you begin
  - 02. Download the J2EE Deployment Distribution
  - 03. Configure the Sample Data
  - 04. Configure the Solutions
  - 05. Build the application archives
  - 06. Deploy the Application Archives
    - Enabling SSL in Tomcat55 & JBoss
    - JBoss 4.0.4
      - Changing the Server Port
    - Tomcat 5.0
    - Tomcat 5.5

Manual Deployment of Pentaho

This page last changed on Jan 28, 2007 by bhagan.

In order to deploy the Pentaho BI Platforn, you must first have a J2EE-compliant archive file that contains all of the files and resources for the platform. If you are using JBoss or Tomcat as your application server, this guide will help you build the archive. If you are using a different application server, you will need to write your own Ant task to build the archive you want, and add the configuration files that are specific to your application server.

We would be happy to incorporate your application server's Ant task back into the bundle should you wish to contribute it. See the details about contributing to Pentaho at http://www.pentaho.org/contributions.

Pentaho has a pre-configured JBoss application server available with everything already set up and deployed for you. We recommend the Pre-Configured Installation (PCI) to anyone who is looking to get up and running quickly with minimal effort. This guide provides an alternative for people who want to setup the platform in their own environment.

01. Before you begin

This page last changed on Jan 30, 2007 by bhagan.

02. Download the J2EE
Deployment Distribution

You will also need to make sure you have the following items:

- This guide requires a Java SDK or JRE. The Pentaho BI Platform is built and tested against Java SDK 1.4, available for many platforms at http://java.sun.com/j2ee/1.4/download.html#sdk

- Make sure that the environment variable "JAVA_HOME" indicates the directory where the JDK is installed.
- This download also requires that Apache Ant, the open source Java-based build tool. You will need version 1.6.2 of Ant or higher. You can download Ant from http://ant.apache.org.

- Make sure that Ant's bin directory is available on your system path.

02. Download the J2EE
Deployment Distribution

02. Download the J2EE Deployment Distribution

This page last changed on Jan 30, 2007 by bhagan.

01. Before you begin 03. Configure the Sample Data

The J2EE Deployment Distribution is a package that will allow you to build a variety of different web application archives and/or enterprise application archives specifically tailored for certain application servers. To date, we have scripts to build the following archives via Ant targets:

- Tomcat 5.x .war file
- JBoss 4.0.4 .war file
- JBoss 4.0.4 .ear file

This distribution can be found on our downloads page listed as pentaho_j2ee_deployments.

This is also a good starting place if you are attempting to deploy to an application server not listed. All of the Pentaho files you will need exist in this package; all you will need to do is write your own Ant target, and add the configuration files necessary for your environment.

Our primary application server testing ground has been JBoss 4.0.4. The Pentaho community and the development team would greatly appreciate any feedback - whether it is problems faced or just to let us know you had a successful deployment - regarding deployments to other application servers. Please use the discussion forum at http://forums.pentaho.org to share your experience.

01. Before you begin 03. Configure the Sample Data

03. Configure the Sample Data

This page last changed on Jan 30, 2007 by bhagan.

02. Download the J2EE Deployment Distribution | 04. Configure the Solutions

This is simple. Download and unpacknpack the pentaho_data-<version>.zip that you can find on Sourceforge.

In the root of the exploded tree, you should see a startup and a shutdown script. To start the database server, run the start-hypersonic (.bat for Windows, .sh for *nix) script. To stop the database server, run the stop-hypersonic (.bat for Windows, .sh for *nix) script.

A suggested directory layout is below:
/pentaho
    /appserver
    /pentaho-data
    /pentaho-solutions

02. Download the J2EE Deployment Distribution | 04. Configure the Solutions

04. Configure the Solutions

This page last changed on Jan 30, 2007 by bhagan.

03. Configure the Sample Data | 05. Build the application archives

You will also need to download and unpack the pentaho-solutions-<version>.zip, which you can find on Sourceforge.

The Pentaho web application will find your pentaho-solutions directory as long as the pentaho-solutions directory is a sibling to the appserver directory.
A suggested directory layout is below.
/pentaho
  /appserver
  /pentaho-data
  /pentaho-solutions

An alternative solutions configuration
If the application has trouble finding the sample solutions or if you just want to put the sample solutions somewhere such that the pentaho-solutions directory is not a sibling of your application server"s root directory, you can enter a configuration parameter to point to them.

Here's how to do it:

1. Extract the web.xml file (from the .war).
2. Find the <param-name> element that has the value "solution-path."
3. Directly after the <param-name> element, enter a <param-value> element, and set its value to the absolute path to the "pentaho-solutions" directory.

```
<param-value>d:\pentaho\pentaho-solutions</param-value>
```

Then, rebuild the .war file.

The user account that starts the BI Platform needs to have permission to create directories and files in the pentaho-demo/pentaho-solutions/system/content directory. If JBoss is installed as a service, the user account that starts the service needs to have create permissions also.

03. Configure the Sample Data | 05. Build the application archives

05. Build the application archives

This page last changed on Mar 16, 2007 by bhagan.

04. Configure the Solutions

06. Deploy the Application Archives

pentaho.war and pentaho.ear

The pentaho_j2ee_deployments-<version>.zip will build two different kinds of archives - .war file or .ear file. The .ear file has all of the files necessary for the Pentaho platform demo and samples to run successfully in the application server. The .war file only contains the Pentaho web application, under which case there is more work for you after you deploy the Pentaho .war file. The archives will be configured to run against hypersonic or mysql. To do this, we must configure a number of files with the HOST and PORT of your database, including:

- hibernate.cfg.xml
- *-ds.xml files (for jboss)

The build will make these replacements based upon values that it finds in the build.properties file. Specifically, the build will be looking for the following properties:

- hsqldb.jdbc.port
- mysql.jdbc.port
- hsqldb.jdbc.host
- mysql.jdbc.host

Without modification, these properties are set to their defaults:

- hsqldb.jdbc.port=9001
- mysql.jdbc.port=3306
- hsqldb.jdbc.host=localhost
- mysql.jdbc.host=localhost

Here's how to do it:
1. Download and unpack the pentaho_j2ee_deployments-<version>.zip file into a working directory. For our examples here, we will use D:\work.
2. Decide which archive you want to build:

- .war file configured for Tomcat 5.X and hypersonic
- .war file configured for Tomcat 5.X and mysql
- .war file configured for JBoss 4.0.4 (with jboss portal installed) and hypersonic
- .war file configured for JBoss 4.0.4 (with jboss portal installed) and mysql
- .war file configured for JBoss 4.0.4 (without jboss portal installed) and hypersonic
- .war file configured for JBoss 4.0.4 (without jboss portal installed) and mysql
- .ear file configured for JBoss 4.0.4 and hypersonic
- .ear file configured for JBoss 4.0.4 and mysql

3. Open a terminal window (*nix) or command prompt (Windows).
4. Change directory to the work directory where you unzipped the deployment bundle.
5. Execute the Ant command that is appropriate for the archive of your choice.

The command will be:

```
ant <ant_target>
```

where <ant_target> is one of the following:

- **war-pentaho-tomcat-hypersonic**
- **war-pentaho-tomcat-mysql**
- **war-pentaho-jboss-hypersonic** - for use with a jboss server that has jboss portal installed
- **war-pentaho-jboss-hypersonic-no-portal** - for use with a jboss servet that does not have jboss portal installed
- **war-pentaho-jboss-mysql** - for use with a jboss server that has jboss portal installed
- **war-pentaho-jboss-mysql-no-portal** - for use with a jboss servet that does not have jboss portal installed
- **ear-pentaho-jboss-hypersonic** - for use with a jboss server that has jboss portal installed
- **ear-pentaho-jboss-hypersonic-no-portal** - for use with a jboss servet that does not have jboss portal installed
- **ear-pentaho-jboss-mysql** - for use with a jboss server that has jboss portal installed
- **ear-pentaho-jboss-mysql-no-portal** - for use with a jboss servet that does not have jboss portal installed

So, for example, if you wanted to build a .war file for Tomcat on Windows, the command would look similar to this:

```
D:\work> ant war-pentaho-tomcat-hypersonic
```

Wait for the build script to finish. The resulting .ear or .war file can be found under the /build/pentaho-wars/<appserver> subdirectory, which is created in your work directory.

pentaho-style.war and sw-style.war

If you are deploying the platform as a .war file, you may be missing some other key components - the pentaho-style.war, the steel-wheels-style.war, and the pentaho-portal-layout.war. For the sake of easy and maintainable style customization, we have moved the content styling for the platform into the separate styles web application.

No need to worry about this if you built and deployed an .ear file - the .ear package includes all extra web resources that the platform needs.

So all you need to do here is simply deploy the pentaho-style.war, the steel-wheels-style.war, and the pentaho-portal-layout.war. into the same application server along side the platform.

Execute the following command:

```
ant zip-pentaho-style-war
```

And when that's finished, execute the following command:

```
ant zip-pentaho-steel-wheels-style-war
```

And when that's finished, execute the following command:

```
ant zip-pentaho-portal-layout-war
```

You can find all of the wars/ears in the following path, /work/build/pentaho-wars
Now that you have built the appropriate archive for your application server, you can deploy the platform. Deployment procedures are going to be specific to the application server that you use, which we assume you are already familiar.

Remember

If you have any problems getting the platform deployed, post your dilemma in our discussion forum at http://forums.pentaho.org and chances are someone has found a solution or can help find one. Post your successes too, and how you managed it!

You can build everything at once and just use what you need. To do this just run:

```
ant build-all
```

04. Configure the Solutions

06. Deploy the Application Archives

06. Deploy the Application Archives

This page last changed on Jan 30, 2007 by bhagan.

05. Build the application archives

Use these links to find instructions for deploying pentaho application archives to the following application servers:

- Enabling SSL in Tomcat55 & JBoss
- JBoss 4.0.4
- Tomcat 5.0
- Tomcat 5.5

05. Build the application archives

Enabling SSL in Tomcat55 & JBoss

This page last changed on Feb 01, 2007 by mdamour.

Introduction

This is a small guide which will help you enable SSL mode in your Tomcat application server. The information provided here is based on Tomcat 5.5 and JBoss 4.0.4 but should be generally useful for a broad
range of their product versions.

Certificate

In a production environment you should obtain a certificate from one of the trusted certification authorities.
But for this example we'll walk through the steps needed to create your own self-signed certificate using the keytool which
comes with the Java Development Kit.

To generate your certificate enter the following command:

```
keytool -genkey -alias tomcat -keyalg RSA
```

You will be prompted for your name and organization, simply enter the details asked for and your certificate will be create in
your keystore. You may have to hunt down the location of this on disk, but typically this is going to be a .keystore file in
your home directory.

Now move the .keystore file into

{JBOSS}/server/default/conf. This is a fairly arbitrary location, but it is the default location specified in the tomcat 5.5
configuration. If you drop the keystore in a different location, be sure to keep this in mind.

Tomcat server.xml Configuration

Edit the server.xml file located in {JBOSS}

/server/default/deploy/jbossweb-tomcat55.sar. Look for this section:

```
<!-- SSL/TLS Connector configuration using the admin devl guide keystore
<Connector port="8443" address="${jboss.bind.address}"
  maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
  emptySessionPath="true"
  scheme="https" secure="true" clientAuth="false"
  keystoreFile="${jboss.server.home.dir}/conf/chap8.keystore"
```

```
    keystorePass="rmi+ssl" sslProtocol = "TLS" />
  -->
```

Notice that this node is commented between <!-- and -->, you'll need to uncomment this node. The port is defaulted to 8443,
if you'll want to use the default SSL port you should change this to 443. Change the keystoreFile property to match the
location and filename of your keystore file.

Lastly, modify the keystorePass to that which you selected when you created the certificate.

Deployment

Now you are ready to start JBoss / Tomcat and use SSL. Bring your server up as normal and hit your web application with an https
url. For example.

```
https://localhost/pentaho/Home
```

Errata

Depending on your luck and your version of Java, you may encounter an error in your server.log such as:

```
HTTPS hostname wrong:  should be <localhost>
```

Followed by a stack trace, preventing you from using SSL. We have fixed this problem in the PentahoSystem by registering our own
hostname verifier. If you see an error like this then the PentahoSystem is not being initialized properly.

Another problem which you may encounter is:

```
Caused by: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification
path to requested target
```

To resolve this, add these settings to your JAVA_OPTS in

{JBOSS}/bin/run.conf:

```
-Djavax.net.ssl.keyStore=/home/mdamour/workspace/pentaho-preconfiguredinstall/server/default/conf/.keystore
-Djavax.net.ssl.keyStorePassword=changeit
-Djavax.net.ssl.trustStore=/home/mdamour/workspace/pentaho-preconfiguredinstall/server/default/conf/.keystor
-Djavax.net.ssl.trustStorePassword=changeit"
```

JBoss 4.0.4

Enabling SSL in Tomcat55 & JBoss | 06. Deploy the Application Archives | Tomcat 5.0

If you are deploying the Pentaho BI Platform to a vanilla JBoss 4.0.4, you will want to follow the tips below.

Datasource Naming
The Pentaho web application depends on JNDI for datasource naming, so in order to run the platform and the samples successfully, these datasources need to be defined to JBoss. The standard in JBoss 4.0.4 is to use an xml file with the -ds.xml naming convention. If you will be deploying an .ear file, you will not have to define these datasources, but if you are deploying a .war, you can use the datasource definitions that we provide for you in the pentaho_j2ee_deployments-<version>.zip. After extracting the zip file, you can find these files in the pentaho-res/jboss/datasources folder:

- PentahoHibernate-ds.xml
- quartz-ds.xml
- sampledata-ds.xml
- sampledata_admin-ds.xml
- shark-ds.xml
- datasourceX-ds.xml (X = numbers 1 through 5)
- solutionX-ds.xml (X = numbers 1 through 5)

Jar Replacement
For Pentaho to work in JBoss, we will have to add/replace a few jars:

- jaxen - The Pentaho platform relies on version 1.1beta4 of the jaxen library. JBoss does not contain the jaxen library, which we use in order to render the platform user interface properly.
- dom4j - The Pentaho platfrom also relies on version 1.6.1 of the dom4j library to render the platform. By default, JBoss 4.0.4 ships with dom4j 1.5.2.
- mail.jar - In addition, we will replace the mail.jar that ships with JBoss with the mail.jar that comes packaged in pentaho-third-party directory inside the pentaho_j2ee_deployments-<version>.zip.

Here's how we do it:

1. Stop the server if it's running.
2. Download jaxen 1.1 beta 4 from http://jaxen.org/releases.html, or use the jaxen.jar packaged in the pentaho_j2ee_deployments-<version>.zip.
3. Download dom4j 1.6.1 from http://dom4j.org/download.html, or use the dom4j_1.6.1.jar that ships in the root of the pentaho_j2ee_deployments-<version>.zip package.
4. **Rename** the jaxen 1.1 beta 4 jar to jaxen.jar and copy the jaxen.jar file into the <jboss_home>/lib directory.
5. Delete the dom4j.jar file from the <jboss_home>/lib directory.
6. **Rename** the dom4j_1.6.1.jar to dom4j.jar and copy the new dom4j.jar file into the <jboss_home>/lib directory.
7. Replace the mail.jar in <jboss_home>/server/default/lib with the mail.jar that ships in the pentaho_j2ee_deployments-<version>.zip.

8. Download the commons-http_3.0.1.jar from http://jakarta.apache.org/site/downloads/downloads_commons-httpclient.cgi.
9. Copy the commons-http_3.0.1.jar to the <jboss_home>/server/default/lib
10. Delete the commons-http.jar (old one that was already there) and rename the commons-http_3.0.1.jar to commons-http.jar
11. Deploy the .war or .ear. If deploying a .war, make sure you name it pentaho.war.
12. If deploying the pentaho.war file, add the *-ds.xml files listed above to the <jboss_home>/server/default/deploy directory. If you are not using the default server, then copy the file into your server's deploy directory.
13. Make sure your database is running.
14. Restart the server.

The JBoss 4.0.4 application server does not come with the JBoss Portal deployed (unless you get JBoss' portal bundle), so any Pentaho portal features or samples will not work. Instructions for setting up the Pentaho platform with a Pre\-existing JBoss Portal will be entered in the future. Those tips that pertain to the pentaho.war file for JBoss apply ONLY to the .war archive of the platform. You can save yourself some hassle if you instead build the pentaho.ear for JBoss which incorporates several steps for you that otherwise would be manual.

Changing the Server Port

This page last changed on Jan 30, 2007 by bhagan.

By default, the platform web application is configured to use port 8080. If you already have another server using port 8080, you can modify your server's configuration so it will use a different port. Each application server has unique requirements on setting the server port. Because our reference implementation runs on JBoss, we include the instructions for changing the port in JBoss. For other application servers, refer to your server's documentation for configuring ports. To change the server port, we will modify the pentaho web application's web.xml and JBoss' server.xml.

Here's how we do it:

1. Extract the web.xml file (from the .war, located at pentaho.war/WEB-INF/web.xml). If you are using the preconfigured-installation, the web.xml is located under pentaho-demo/jboss/server/default/deploy/pentaho.war/WEB-INF/web.xml.
2. Add or edit the "base-url" context parameter.
3. Change "server-name" to the name or IP address used to access the server. If you will only be running the server and web browser on the same machine, you can leave this as "localhost".
4. Change "port" to the port number that you want to use.

```
<context-param>
 <param-name>base-url</param-name>
 <param-value>http://server-name:port/pentaho</param-value>
</context-param>
```

The reason we have the `base-url`setting in the deployment descriptor is so that the application server can generate content containing URLs back to the server (for example, e-mailing content).

Here's how to edit the JBoss server.xml:

1. Edit `<jboss-home>/server/default/deploy/jbossweb-tomcat55.sar/server.xml`.The server.xml is located under jboss/server/default/deploy/jbossweb-tomcat55.sar/server.xml. If you're using Tomcat 5.0 or Tomcat 5.5, the following instructions also apply, but the location of the server.xml will be different.
2. A few lines from the top of the server.xml file, you will see the following lines:

```
<!-- A HTTP/1.1 Connector on port 8080 -->
 <Connector port="8080" address="${jboss.bind.address}"
```

   Change the *port="8080"* text to be the port number that you want to use. You may also change the other connector's ports as well if you want.

Tomcat 5.0

This page last changed on Jan 30, 2007 by bhagan.

JBoss 4.0.4 | 06. Deploy the Application Archives | Tomcat 5.5

If you are deploying the Pentaho BI Platform to Tomcat 5.0, you will want to follow the tips below. In order to access the default databases for the platform and samples, you must add the hsqldb.jar (the Hypersonic database drivers) to the common/lib directory in Tomcat 5.0. The Pentaho web application depends on JNDI for datasource naming, so in order to run the platform and the samples successfully, you will need to define these datasources to Tomcat. To do that, we use xml entries in the server.xml file.

Here's how we do it:

1. Stop the Tomcat server.
2. Copy the hsqldb.jar from pentaho-third-party directory to <tomcat-home>/common/lib directory, where <tomcat-home> is the root directory of your Tomcat application server. You can retrieve the hsqldb.jar from the pentaho-third-party directory, which is part of the 'pentaho_j2ee_deployments-<version>.zip'.
3. Open the server.xml file, located in the <tomcat-home>/conf directory.
4. **Carefully** copy and paste the xml listed below in between the `<DefaultContext></DefaultContext>` nodes in the server.xml file. If you do not have `<DefaultContext></DefaultContext>` nodes, create them and paste the new xml in between.
5. Save and close the server.xml file.
6. Start your database, if it's not already started.
7. Restart the Tomcat server.

Sample JNDI Datasource Mappings for Tomcat 5.0

```
<Resource name="jdbc/SampleData" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/SampleData">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/SampleDataAdmin" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/SampleDataAdmin">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_admin</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/Hibernate" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/Hibernate">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>hibuser</value></parameter>
```

```
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/hibernate</value></parameter>
</ResourceParams>

<Resource name="jdbc/Quartz" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/Quartz">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/quartz</value></parameter>
</ResourceParams>

<Resource name="jdbc/Shark" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/Shark">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>sa</value></parameter>
        <parameter><name>password</name><value></value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/shark</value></parameter>
</ResourceParams>

<Resource name="jdbc/datasource1" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/datasource1">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/datasource2" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/datasource2">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/datasource3" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/datasource3">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/datasource4" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/datasource4">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/datasource5" auth="Container" type="javax.sql.DataSource"/>
```

```
<ResourceParams name="jdbc/datasource5">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/solution1" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/solution1">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/solution2" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/solution2">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/solution3" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/solution3">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/solution4" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/solution4">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>

<Resource name="jdbc/solution5" auth="Container" type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/solution5">
        <parameter><name>factory</name><value>org.apache.commons.dbcp.BasicDataSourceFactory</value></parame
        <parameter><name>maxActive</name><value>20</value></parameter>
        <parameter><name>maxIdle</name><value>5</value></parameter>
        <parameter><name>maxWait</name><value>10000</value></parameter>
        <parameter><name>username</name><value>pentaho_user</value></parameter>
        <parameter><name>password</name><value>password</value></parameter>
        <parameter><name>driverClassName</name><value>org.hsqldb.jdbcDriver</value></parameter>
        <parameter><name>url</name><value>jdbc:hsqldb:hsql://localhost/sampledata</value></parameter>
</ResourceParams>
```

The datasourceX and solutionX JNDI names are provided in our sample PCI to make it very easy for people evaluating the server to communicate with their data sources without having to modify the Pentaho web.xml. However, in a resource constrained environment, defining multiple JNDI connections against the same server may use up valuable database server connections

(depending upon the JNDI implementation by the server). If you choose not to include the datasourceX and solutionX data sources, you will have to manually removethe declarations from the web application deployment descriptor (web.xml). At the bottom of the web.xml, you will find all the resource reference (`resource-ref`) entries that refer to datasourceX/solutionX JNDI names.

In this procedure, we've assumed that Tomcat is setup to auto-deploy new web applications.

Tomcat 5.5 (at the time of this writing) is not distributed with a JSR168 compatible portlet container, so any Pentaho portal features or samples will not work. Stringbeansis an open source JSR-168 portal that is known to work within Apache Tomcat. We hope to get instructions from the community on How to integrate Stringbeans in Tomcat 5.5, or other portals like it from the community.

If you receive an error similar to IOException while loading persisted sessions, add the following line to your <Context> element:

<Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>

Tomcat 5.5

This page last changed on Jan 30, 2007 by bhagan.

Tomcat 5.0 06. Deploy the Application Archives

If you are deploying the Pentaho BI Platform to Tomcat 5.5, follow the tips below. In order to access the default databases for the platform and samples, you must add the hsqldb.jar (the Hypersonic database drivers) to the common/lib directory in Tomcat 5.5. The Pentaho web application depends on JNDI for datasource naming, so in order to run the platform and the samples successfully, you will need to define these datasources to Tomcat. We've found that adding the resources to a <Context> element nested within the <Host> element works well. We've provided some sample xml below.

Here's how we do it:

1. Stop the Tomcat server.
2. Copy the pentaho.war into <tomcat-home>/webapps
3. Copy the hsqldb.jar from pentaho-third-party directory to <tomcat-home>/common/lib directory, where <tomcat-home> is the root directory of your Tomcat application server. You can retrieve the hsqldb.jar from the pentaho-third-party directory, which is part of the pentaho_j2ee_deployments-<version>.zip.
4. Open the server.xml file, located in the <tomcat-home>/conf directory.
5. **Carefully** add the xml listed below within the `<Host></Host>` elements in the server.xml file.
6. Save and close the server.xml file.
7. Start your database, if it's not already started.
8. Restart the Tomcat server.

Sample JNDI Datasource mappings for Tomcat 5.5

```
<Context path="/pentaho" docbase="webapps/pentaho/">
  <Resource name="jdbc/SampleData" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/Hibernate" auth="Container" type="javax.sql.DataSource"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
    maxWait="10000" username="hibuser" password="password"
    driverClassName="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:hsql://localhost/hibernate" />
  <Resource name="jdbc/Quartz" auth="Container" type="javax.sql.DataSource"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
    maxWait="10000" username="pentaho_user" password="password"
    driverClassName="org.hsqldb.jdbcDriver" url="jdbc:hsqldb:hsql://localhost/quartz" />
  <Resource name="jdbc/Shark" auth="Container" type="javax.sql.DataSource"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory" maxActive="20" maxIdle="5"
    maxWait="10000" username="sa" password="" driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/shark" />
  <Resource name="jdbc/SampleDataAdmin" auth="Container" type="javax.sql.DataSource"
maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_admin" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/solution1" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/solution2" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
```

```
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/solution3" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/solution4" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/solution5" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/datasource1" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/datasource2" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/datasource3" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/datasource4" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
  <Resource name="jdbc/datasource5" auth="Container" type="javax.sql.DataSource" maxActive="20"
    maxIdle="5" maxWait="10000" username="pentaho_user" password="password"
    factory="org.apache.commons.dbcp.BasicDataSourceFactory"
driverClassName="org.hsqldb.jdbcDriver"
    url="jdbc:hsqldb:hsql://localhost/sampledata" />
</Context>
```

The datasourceX and solutionX JNDI names are provided in our sample PCI to make it very easy for people evaluating the server to communicate with their data sources without having to modify the Pentaho web.xml. However, in a resource constrained environment, defining multiple JNDI connections against the same server may use up valuable database server connections (depending upon the JNDI implementation by the server). If you choose not to include the datasourceX and solutionX data sources, you will have to manually removethe declarations from the web application deployment descriptor (web.xml). At the bottom of the web.xml, you will find all the resource reference (`resource-ref`) entries that refer to datasourceX/solutionX JNDI names.

In this procedure, we've assumed that Tomcat is setup to auto-deploy new web applications.

Tomcat 5.5 (at the time of this writing) is not distributed with a JSR168 compatible portlet container, so any Pentaho portal features or samples will not work. Stringbeansis an open source JSR-168 portal that is known to work within Apache Tomcat. We hope to get instructions from the community on How to integrate Stringbeans in Tomcat 5.5, or other portals like it from the community.

If you receive an error similar to IOException while loading persisted sessions, add the following

line to your <Context> element:

```
<Manager className="org.apache.catalina.session.PersistentManager" saveOnRestart="false"/>
```