



# Jahia 5.0 SP3 with Websphere Application Server 6.1 Installation Guide

DRAFT

**Jahia**

9 route des Jeunes, CH-1227, Carouge Switzerland

<http://www.jahia.com> > The company web site

<http://www.jahia.net> > The community web site

---

# VERSION

This table records the versions of this document and their last updates

Version	Author	Date	Modifications
1.0	Damien Saulnier	2007-12-07	Initial document
2.0	Damien Saulnier	2007-12-11	Portlets deployment
2.1	Khue Nguyen	2008-02-01	Clip portlet deployment
2.2	Damien Saulnier	2008-02-06	Portlet deployment: general case -> reorganisation

TODO:

- Portlets: known issues
- Cluster configuration

---

# SUMMARY

	Page
<b>Version</b>	<b>2</b>
<b>Summary</b>	<b>3</b>
<b>Introduction</b>	<b>4</b>
<b>Jahia application deployment</b>	<b>5</b>
Jahia Enterprise ARchive (Ear) creation	5
Data sources definition	6
JDBC Provider creation	6
Data sources creation	7
Authentication alias creation	7
Data sources creation	7
Shared Libraries definition	9
Ear deployment	11
<b>Portlets Deployment</b>	<b>16</b>
Portlet Deployment : General case	16
Configuration	16
Portlet Preparation	16
Portlet Deployment	17
Portlet registration in Jetspeed	18
Known issues	20
Clip Portlet Deployment	21
<b>Cluster Configuration</b>	<b>23</b>

---

# INTRODUCTION

This document explains how to install Jahia 5.0 SP3 on Websphere Application Server version 6.1. You will find here information on how to deploy and configure your Jahia application, on how to deploy portlets on your Jahia Server, and on how to configure your Jahia Server in a cluster architecture.

---

# JAHIA APPLICATION DEPLOYMENT

## JAHIA ENTERPRISE ARCHIVE (EAR) CREATION

To create your Ear file, download your Jahia distribution from our website.  
Then:

- Extract the jahia directory from *tomcat/webapps/*
- Edit the *WEB-INF/web.xml* file and remove all the pre compiled jsp servlets between those two comments :

```
<!--  
Automatically created by Apache Jakarta Tomcat JspC.  
Place this fragment in the web.xml before all icon, display-name,  
description, distributable, and context-param elements.  
-->
```

and

```
<!--  
All session-config, mime-mapping, welcome-file-list, error-page, taglib,  
resource-ref, security-constraint, login-config, security-role,  
env-entry, and ejb-ref elements should follow this fragment.  
-->
```

- Create a war file containing all of your jahia directory
- After that, create a *META-INF* directory at the same level of your *jahia.war* file
- In this *META-INF* directory, we will create an *application.xml* file like this one :

```
<?xml version="1.0" encoding="UTF-8"?>  
<application xmlns="http://java.sun.com/xml/ns/javaee"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
http://java.sun.com/xml/ns/javaee/application_5.xsd"  
version="5">  
<display-name>Jahia</display-name>  
<module id="jahia">  
<web>  
<web-uri>jahia.war</web-uri>  
<context-root>jahia</context-root>  
</web>  
</module>  
</application>
```

- The last step is creation of the ear file grouping *jahia.war* and your *META-INF* directory in the same file

## DATA SOURCES DEFINITION

You need to define two data sources in your Application server. Those data sources will be further mapped on the resources declared in your Jahia application.

### JDBC Provider creation

You need to create a JDBC provider to define connection with your database server. If you want to use a MS SQL Server database, you can use a JDBC Provider included with your WAS server, and so go directly to next paragraph.

For other database providers, you will have to create manually your JDBC Provider:

- Open **Resources / JDBC / JDBC Providers**, specify **Server scope** (Node=node\_name, Server=server\_name)
- Click on “**new**” button
- **Fill in all required fields** on screen :

Create a new JDBC Provider

→ Step 1: Create new JDBC provider

Step 2: Enter database class path information

Step 3: Summary

Create new JDBC provider

Set the basic configuration values of a JDBC provider, which encapsulates the specific vendor JDBC driver implementation classes that are required to access the database. The wizard fills in the name and the description fields, but you can type different values.

Scope  
cells:poule03Node01:Cell:nodes:poule03Node01:servers:server1

\* Database type  
SQL Server

\* Provider type  
WebSphere embedded ConnectJDBC driver for MS SQL Server

\* Implementation type  
Connection pool data source

\* Name  
WebSphere embedded ConnectJDBC driver for MS SQL :

Description  
IBM WebSphere Connect JDBC driver for MS SQL Server.

Next Cancel

- Click on **Next / finish / save**

## Data sources creation

### Authentication alias creation

You need to define the login and password that will be used to connect to your database instance.

- Open **Resources / JDBC / Data sources**
- Specify **Server scope**
- Click on **New / Create an new J2C authentication alias / New**
- **Fill in** all required **fields** on screen :



The screenshot shows a web browser window titled "Data sources". The breadcrumb navigation is "Data sources > JAAS - J2C authentication data > New". Below the breadcrumb, there is a description: "Specifies a list of user identities and passwords for Java(TM) 2 connector security to use." The main content area is titled "Configuration" and contains a "General Properties" section. This section has four fields: "Alias" with the value "Jahia Db login", "User ID" with the value "db\_user", "Password" with the value "\*\*\*\*\*", and "Description" which is empty. At the bottom of the configuration area, there are four buttons: "Apply", "OK", "Reset", and "Cancel".

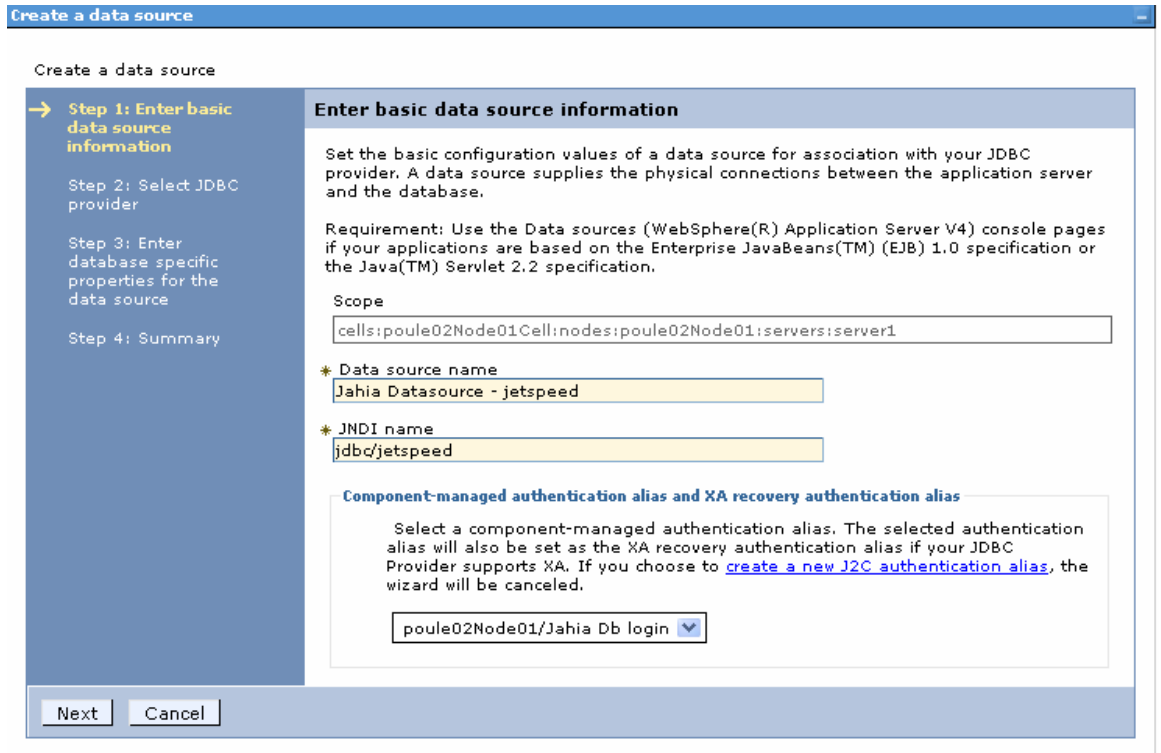
- Click on **Apply / Save**

## Data sources creation

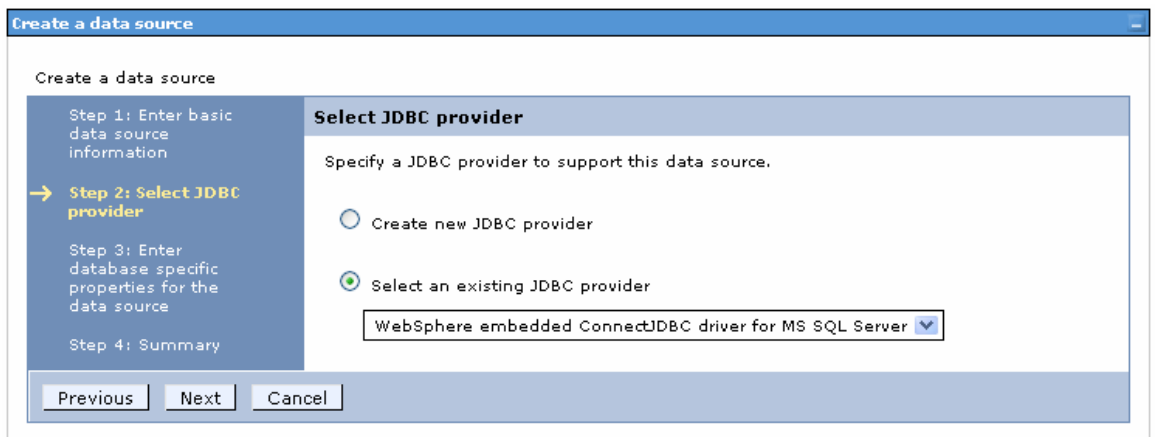
You can now create your Data sources. The Jahia application requires two Data sources: one with **JNDI name jdbc/jetspeed** and one with **JNDI name jdbc/jetspeedNonTx**.

- Open **Resources / JDBC / Data sources**
- Specify **Server scope**
- Click on **New**

- Specify an explicit **name** for your Data source, specify one of the required **JNDI names**, and select your previously defined **authentication alias** :



- Click on **Next**
- Select the **JDBC Provider** you want to use :



- Click on **Next**
- Enter **properties of your database instance** :

**Create a data source**

Create a data source

Step 1: Enter basic data source information

Step 2: Select JDBC provider

→ Step 3: Enter database specific properties for the data source

Step 4: Summary

**Enter database specific properties for the data source**

Set these database-specific properties, which are required by the database vendor JDBC driver to support the connections that are managed through this data source.

\* Database name  
jahia\_db

\* Server name  
10.8.37.8

\* Port number  
1433

Use this data source in container managed persistence (CMP)

Previous Next Cancel

- Click on **Finish / save**
- Redo the same operation for the **second Data source**
- **Test connection** for the two created Data sources on Resources / JDBC / Data sources

## SHARED LIBRARIES DEFINITION

You have now to define some libraries that will be shared by your Jahia application and all its modules.

- **Copy** all the **libraries** found in *tomcat/shared/lib/* directory from your Jahia distribution in a local directory
- Open **Environment / Websphere variables**
- Specify **Server scope**
- Click on **new**
- **Specify a variable** that will design the directory where you have copied the shared libraries. **Enter the path** in value field :

Configuration

---

**General Properties**

\* Name  
jahia\_shared\_libs\_dir

Value  
C:/IBM/jahia\_shared\_libs

Description

Apply OK Reset Cancel

- Click on **Apply / Save**
- Open **Environment / Shared libraries**
- Specify **Server scope**
- Click on **new**
- **Enter** an explicit **name** for your shared libraries, and **reference each jar** copied from tomcat/shared/libs in "Classpath" box. This box must contain only one jar by line, with no other separator that the end of line :

Configuration

---

**General Properties**

\* Scope

\* Name

Description

\* Classpath

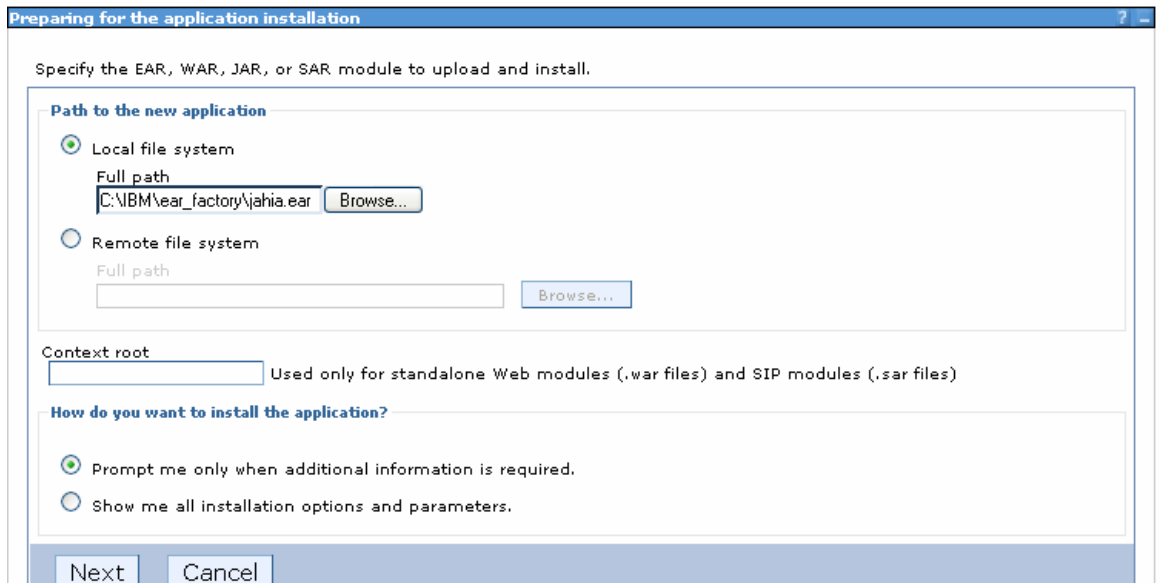
Native Library Path

- Click on **Apply / Save**

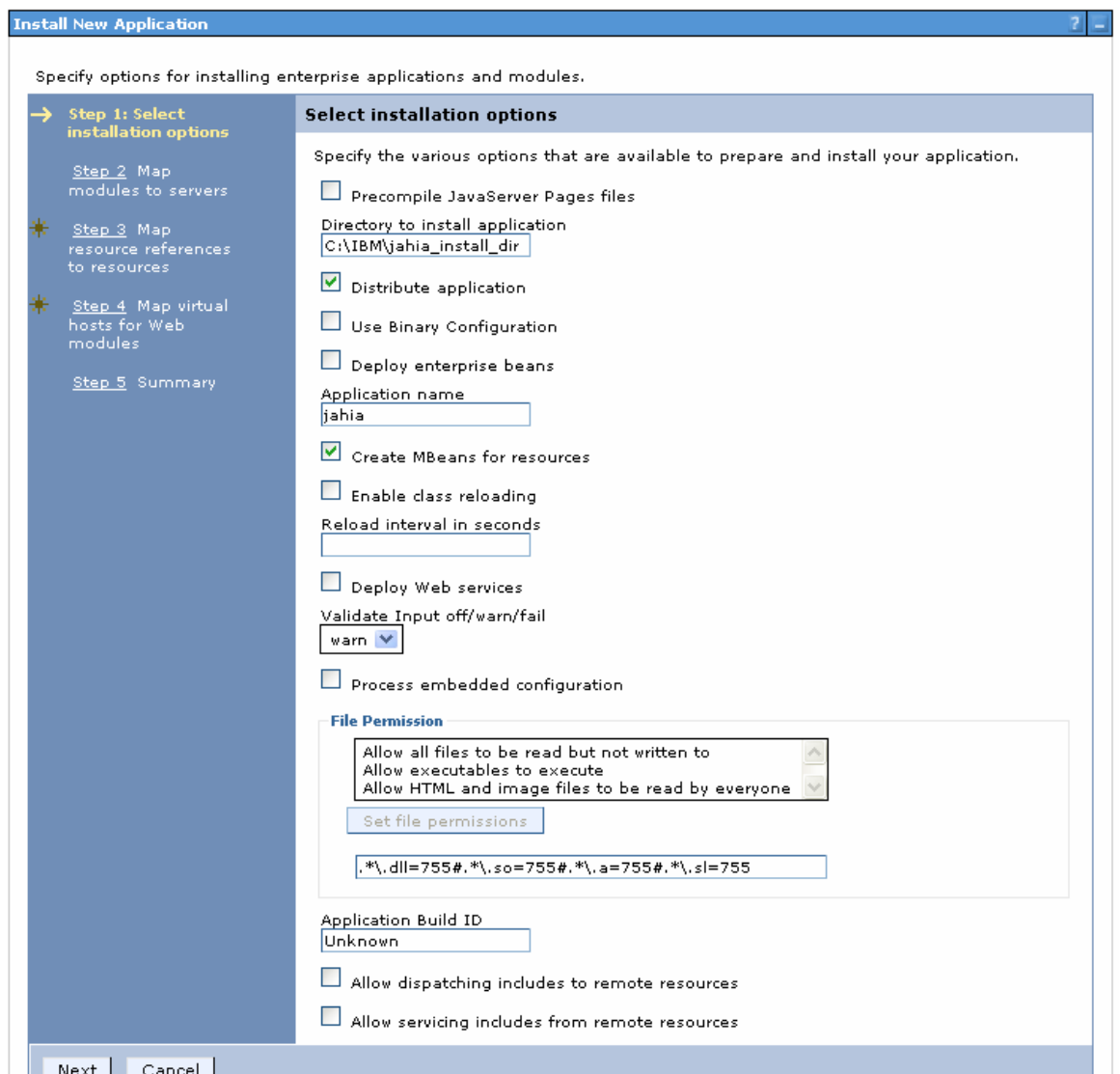
## EAR DEPLOYMENT

You can now deploy the Ear you had previously packaged.

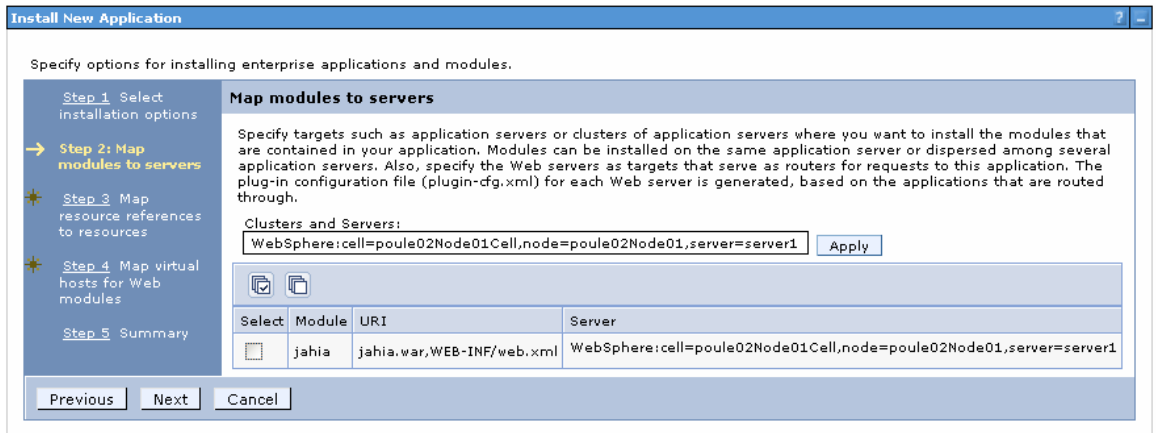
- Open **Applications / Install new application**
- Select your **ear file** :



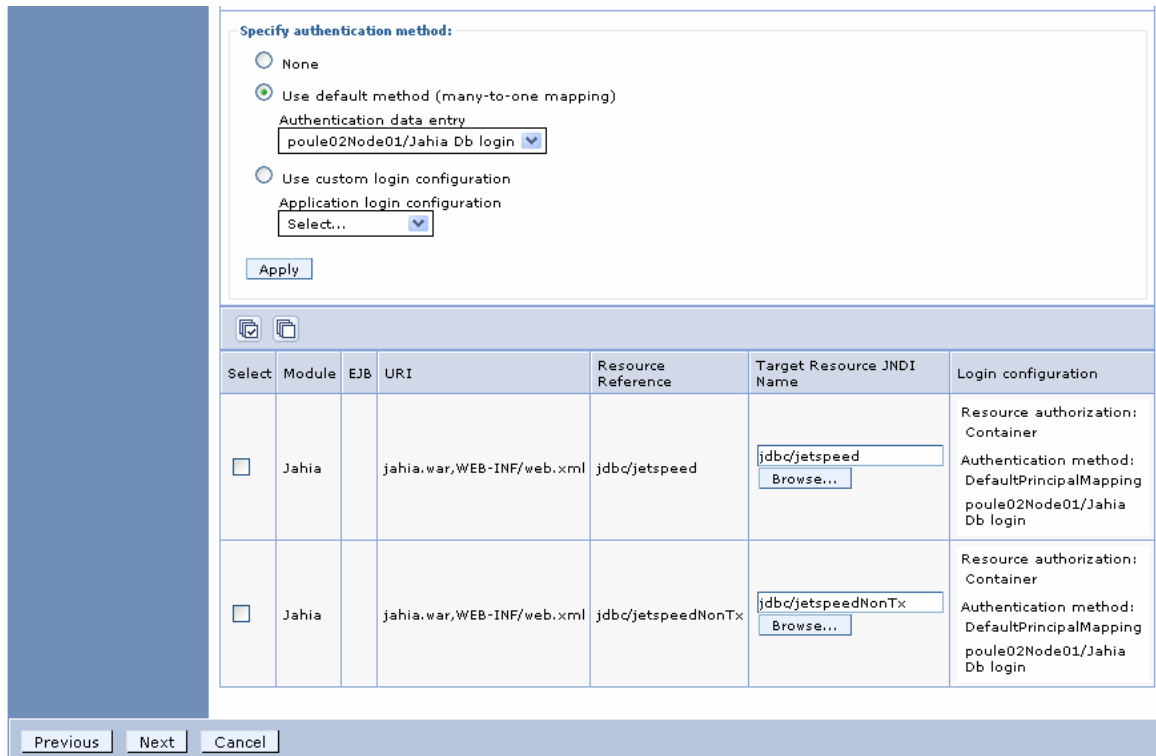
➤ Specify where Jahia will be installed :



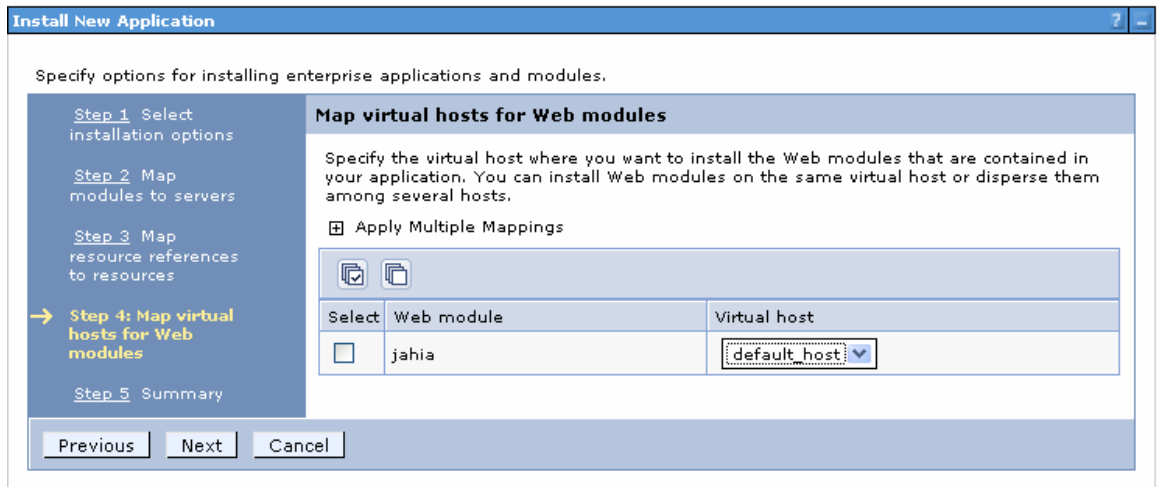
➤ Nothing to do :



- **Map Jahia resources** declared in the web.xml file on previously defined Data sources
- Select each Jahia resources, **select the authentication method** previously defined and click on apply



- Nothing to do :



- Click on **Next / finish / save**

Now that your Jahia application is deployed on your Websphere Application Server, you have to perform some configuration operations before it is operational.

- Open **Applications / Enterprise Applications / Jahia / Shared libraries references**
- For both the application and the module, **reference** the previously defined **shared library**
- Click on **Ok / save**
- Open **Applications / Enterprise Applications / Jahia / Class loading and update detection**

- Set that you want only **one class loader for all the application** :

Configuration

**General Properties**

Reload classes when application files are updated

Polling interval for updated files  
10 Seconds

**Class loader order**

Classes loaded with parent class loader first  
 Classes loaded with application class loader first

**WAR class loader policy**

Class loader for each WAR file in application  
 Single class loader for application

Apply OK Reset Cancel

- Click on **Apply / Save**
- Open Servers / Application servers
- Select the server on which your Jahia application is deployed
- Ensure that classloader policy is set to multiple (at server level)
- Open Applications / Enterprise Applications
- **Start your Jahia application**
- You can now **access Jahia** on [http://server\\_name\\_or\\_IP:9080/jahia](http://server_name_or_IP:9080/jahia)
- Follow the Jahia **install wizard**
- In `jahia/WEB-INF/etc/config/jahia.properties` , uncomment `localAccessUri` and set port to 9080
- In `jahia/WEB-INF/etc/spring/applicationcontext-services.xml` , modify the value of the `syncUrl` property so that the port is 9080 in url
- **Restart your Websphere Application Server** (not only the jahia application)

**Your Jahia server is now ready to use!**

---

# PORTLETS DEPLOYMENT

We recommend to use Websphere Application Server 6.1.0.7 or newer, as it will let you prevent your portlets being deployed into the portlet container provided in WAS. Indeed, if you do not, you may encounter some problem as Jahia provides its own portlet container.

See <http://www-1.ibm.com/support/docview.wss?uid=swg24016004> for more details.

## PORTLET DEPLOYMENT : GENERAL CASE

You need to deploy your portlet as a module of your Jahia application, and not as a new Websphere application.

### Configuration

- Open Applications / Enterprise Applications / Jahia / Manage Modules / jahia
- Change Starting weight to a lower value if not already done (default value is 10'000) so that the Jahia module starts before the deployed portlets
- Apply / save

### Portlet Preparation

You have now to perform some modifications to your war file before deploying your portlet on WAS. First, modify its web.xml file so that it can be loaded by Jetspeed. Add the following lines in this file if missing, replacing the highlighted param-value by the "context root" you want to define for your portlet:

```

<servlet>
  <servlet-name>JetspeedContainer</servlet-name>
  <display-name>Jetspeed Container</display-name>
  <description>MVC Servlet for Jetspeed Portlet Applications</description>
  <servlet-
class>org.apache.jetspeed.container.JetspeedContainerServlet</servlet-class>
  <init-param>
    <param-name>contextName</param-name>
    <param-value>iframeportlet</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>JetspeedContainer</servlet-name>
  <url-pattern>/container/*</url-pattern>
</servlet-mapping>

<taglib>
  <taglib-uri>http://java.sun.com/portlet</taglib-uri>
  <taglib-location>/WEB-INF/tld/portlet.tld</taglib-location>
</taglib>

```

Ensure that the “portlet.tld” file that you reference in web.xml actually exists. This file has to be provided by the developer of the portlet. If missing, this file is automatically generated by jetspeed when deployed in a Jahia application running on tomcat. But as your portlet has been deployed by the portlet container provided by WAS, and not by jetspeed, this file is required on such environment. Refer to your portlet provider if this file is missing.

## Portlet Deployment

You can now deploy your war file:

- Open Applications / Enterprise Applications
- Select your Jahia application, and click on “update”
- Select “replace or add a single module”
  - Specify where you want to install the module. This path is relative to Jahia install path
  - Select the war file to deploy
  - Specify the Context root (must be equal to the one you defined in the web.xml file)

**Replace or add a single module**

If the path to the new module matches an existing path to a module in the installed application, the new module replaces the existing module. If the path to the module does not exist in the installed application, the new module is added to the application.

Specify the path beginning with the installed application archive file to the module to be replaced or added.

**Specify the path to the module.**

**Local file system**

Full path

**Remote file system**

Full path

Context root  
 Used only for standalone Web modules (.war files) and SIP modules (.sar files)

**How do you want to install the application?**

**Prompt me only when additional information is required.**

**Show me all installation options and parameters.**

- Click on Next, and then select required installation options, depending to your portlet
- Next / Next / Next/ Finish / save
- Open **Applications / Enterprise Applications / Jahia / Shared libraries references**
- Reference your shared libraries for your just deployed portlet. Ensure that it is still referenced by Jahia and your possible other deployed portlets
- OK / Save
- Restart your Websphere Application Server

**Your portlet is now deployed on WAS.**

### **Portlet registration in Jetspeed**

- Go in Jahia Administration Center, and open Manage portlets / Manually add a new portlet
- Specify the path to your deployed portlet in "Full Path" field, then click on "Scan Portlet"

### Manually register a new portlet in Jahia :

You can manually add a new portlet by entering the full path to the portlet ( which can be a directory, an .ear or .war file ).

Note : Only .ear or .war files are deployed.  
If the source is a directory, only the component definition is registered in Jahia.

Full Path :

**Scan Portlet**

### Portlet Details :

### List of Web Applications :

Portlet Name :

Portlet Context : /iframeportlet

Description :

List of Servlets :

Name :	JetspeedContainer
Source :	org.apache.jetspeed.container.JetspeedContainerServlet
Type :	Servlet
Description :	

**Deploy**

Other operations : • [Back to portlets list](#)  
• [Back to menu](#)



© Copyright 2002-2007 **Jahia Ltd** -Tous droits réservés..

Jahia 5.0.3 r18957

➤ Click on Deploy

**Your portlet is now registered in Jetspeed and ready to use on your Jahia sites!**

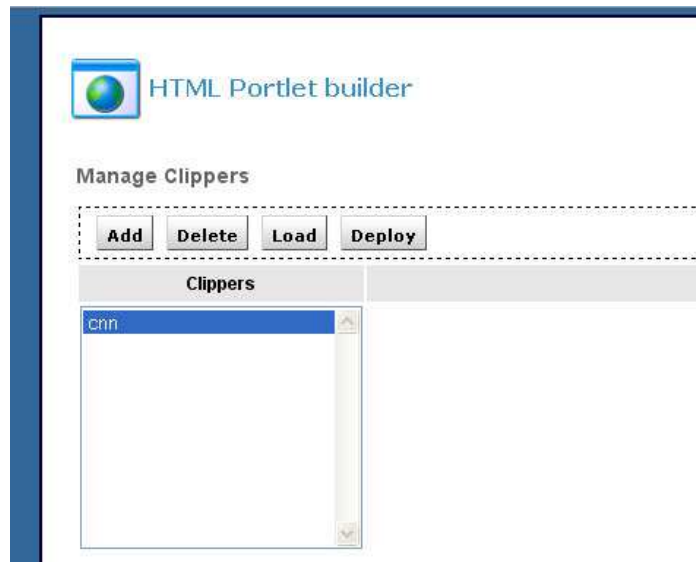
## KNOWN ISSUES

**Work in progress**

## CLIP PORTLET DEPLOYMENT

This is the procedure to deploy a clip-portlet created with the clip-portlet Builder.

- Save a clip-portlet and click on Deploy



This will produce the corresponding war portlet:

```
<Jahia>/WEB-INF/var/new_webapps/jahia_clip_cnn.war
```

- Open the generated war file and change the web.xml as documented in chapter "Portlets Deployment: General case":

```
<servlet>
  <servlet-name>JetspeedContainer</servlet-name>
  <display-name>Jetspeed Container</display-name>
  <description>MVC Servlet for Jetspeed Portlet Applications</description>
  <servlet-
class>org.apache.jetspeed.container.JetspeedContainerServlet</servlet-class>
  <init-param>
    <param-name>contextName</param-name>
    <param-value>jahia_clip_cnn</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>JetspeedContainer</servlet-name>
  <url-pattern>/container/*</url-pattern>
</servlet-mapping>

<taglib>
  <taglib-uri>http://java.sun.com/portlet</taglib-uri>
  <taglib-location>/WEB-INF/tld/portlet.tld</taglib-location>
</taglib>
```

You may give each portlet a distinct display Name:

```
<display-name>cnnClipperPortlet</display-name>  
<description>cnnClipperPortlet</description>
```

- Deploy your portlet as documented in chapter “Portlets Deployment: General case”

---

# CLUSTER CONFIGURATION

**Work in progress**