



Whitepaper

Knowledge sharing leads to personal growth

Hippo Portal in 21 steps

Overview

Keywords

- Hippo Portal
- Portlets
- JSR-168
- Hippo Repository
- Hippo CMS 6
- Sitemap
- Java
- Maven 1.0.2
- Tomcat
- Jetspeed-2
- Velocity
- Freemarker

Challenge

How to create a Hippo Portal

Solution

This document

Frameworks

- Hippo Portal
- Hippo CMS 6
- Apache Jetspeed 2

A step by step description on how to start your first Hippo Portal project.

This document is a step by step description on how to start your first Hippo Portal project. The steps described contain more background information than the getting started track on the Hippo Portal website¹, that has been used as a base for this document, such as best practices and tips and tricks.



Introduction

Getting started with Hippo Portal

In this document we create a starting directory where we will put all our programs, sources and other files. It is possible to create your own structure and use existing versions of the programs used, just update the files with the correct paths and file names. The order of the steps is crucial as you need the previous step(s) to do the next step(s).

Hippo Portal version 1.07.00 is the version used in this example and is based on Jetspeed-2 version 2.1.3. "Hippo Portal allows partners, employees and customers to chose their user experience, with personalized applications based on role, context, actions, location, preferences and team collaboration needs. Hippo Portal software provides a composite application or business mashup framework and the advanced tooling needed to build flexible, SOA-based solutions, as well as the unmatched scalability required by any size organization."²

With this paper comes a zip file containing all the code examples found in this paper and some additional files.

Step 1; Creating the directory structure

The first action is to create your directory structure in order to have everything together and have the same structure for every project. A best practice is to use lower case and avoid spaces in the directory names. The directory structure used in this example is as follows:

```
getting_started
  progs
    apache
      maven
      tomcat
    hippo
    java
  project
    source
    workspace
  maven_local_repo
```

Step 2; Installing Java

For Hippo Portal we will need a Java 1.5 JDK (Java Development Kit). This can be found here:

http://java.sun.com/javase/downloads/index_jdk5.jsp. Install the JDK in your newly created directory structure under the directory java. The version used in this example is JDK 1.5.0_16. To find out what version is used type the following in a command prompt:

```
java -version
```



In general this will show the java version the JAVA_HOME is pointing to. As you can set the JAVA_HOME for a command prompt you need to make sure the correct version is used by your project. The JRE (Java Runtime Environment) used by internet explorer can throw this of. To solve this you need to rename the files mentioned below in the directory WINDOWS\system32:

- java.exe
- javaw.exe
- javaws.exe

Step 3; Installing Hippo repository

The Hippo Repository contains the site map of the portal. For now we will use the default site map, Step 20 contains an explanation on how to update and deploy your own site map.

The Hippo Repository used can be found here:

<http://www.hippoportal.org/resources/>, more versions of the Hippo Repository can be found here:

<http://repository.hippocms.org/hippo-repository/>. The version used in this example is hippo-repository-1.2.11-helloworld.zip.

The different versions have different sample site maps. Install the Hippo Repository in the hippo directory by unzipping the downloaded file.

Step 4; Installing Hippo CMS

Hippo CMS is not a must have when running a Hippo Portal but most of the time it is used to store the content of the portal. For this example we are using a small part of the Hippo CMS.

The Hippo Content Management System (CMS) can be found here:

<http://repositroy.hippocms.org/hippo-cms/>. The version used in this example is hippo-cms-v6.05.04.zip.

Install the Hippo CMS in the hippo directory by unzipping the downloaded file.

Step 5; Installing Apache tomcat

For Hippo Portal the apache tomcat version 5.5 is used. The version 5.5.23 of tomcat is tested for Hippo Portal. A known bug in version 5.5.25 is that the JAAS (Java Authentication and Authorization Service) does not work correctly. The latest version as of writing this paper was 5.5.27, as this example does not contain any JAAS the latest version is used here.

Tomcat version 5.5.x can be found here:

<http://tomcat.apache.org/download-55.cgi>. Install tomcat in the tomcat directory by unzipping the downloaded file. A tip is to



rename your tomcat instance to the application deployed to it, for example tomcat-gettingstarted. This is useful so you know what the main application deployed to that tomcat version is. You can install more than one instance of tomcat on your system.

The following jars need to be copied into the directory, common/endorsed, found in the installed tomcat version:

- xml-apis-2.0.2.jar
- xalan-2.4.1.jar
- xerces-2.3.0.jar

These jar files can be found in the zip file under the directory jar_files.

Step 6; Installing MySql

In this example we are using a MySql database. This can be changed to any other database, just update the files to the correct locations and names. MySql can be downloaded here:

<http://dev.mysql.com/downloads/>, downloading the free community server version is enough for development purposes.

This is by default a windows service so it is not recommended to have more than one instance of MySql on your system.

In order for Hippo portal to find the drivers you need to copy the jar mysql-connector-java-5.0.4.jar into the common/endorsed directory found in the installed tomcat version. This jar file can be found in the zip file under the directory jar_files.

Create a database with the name helloWorldPortal. In the MySql command line client enter the following command:

```
create database helloworldportal
```

Step 7; Installing Maven

Hippo Portal uses maven version 1.0.2, this can be downloaded from: <http://maven.apache.org/maven-1.x/index.html>. This Hippo Portal version is restricted to maven 1.0.2 as the repository plugin we are using needs this maven version. The local repository for maven by default is defined in the documents and settings directory of the logged in user. This can be changed by setting the variable MAVEN_HOME_LOCAL. We are setting this to the directory maven_local_repo. Maven uses remote repositories in order to get the resources defined in the build files. These repositories are defined in the file build.properties located in the root of the local repository. In our directory structure this will be the directory maven_local_repo. The file contains the following line:

```
maven.repo.remote=http://www.bluesunrise.com/maven/,http://repository.hippocms.org/maven/,http://repo1.maven.org/maven/,http://dist.codehaus.org/,http://cvs.apache.org/repository
```



The order of the remote repositories should not be changed as this is the order in which maven will search for the resources.

Step 8; Creating batch files

Creating batch files for all the things you will do more than ones while developing a Hippo portal makes it possible to work on more than one project and with different versions of tools without having to change the environment settings in between builds. In order for the example files to work you need to replace {PROJECT_LOCATION} with the correct location of your project. As the files are very similar we will only show one here. All the files can be found in the zip file in the directory batch_files. The following files are useful when developing Hippo Portals:

- cmd_project_build.bat; opens a command prompt with the correct java version and maven settings. This command prompt will be used for executing maven commands.

```
@echo off
ECHO.
ECHO JAVA_HOME is set to: %JAVA_HOME%
SET
JAVA_HOME={PROJECT_LOCATION}\gettingstarted\progs\java\jdk1.5.0_16
ECHO JAVA_HOME is set to: %JAVA_HOME%
ECHO.
ECHO MAVEN_HOME is set to: %MAVEN_HOME%
SET
MAVEN_HOME={PROJECT_LOCATION}\gettingstarted\progs\apache\maven\maven-1.0.2
ECHO MAVEN_HOME is set to: %MAVEN_HOME%
ECHO.
ECHO PATH is set to: %PATH%
SET PATH=.;%JAVA_HOME%\bin;%MAVEN_HOME%\bin;%PATH%
ECHO PATH is set to: %PATH%
ECHO.
SET
MAVEN_HOME_LOCAL={PROJECT_LOCATION}\gettingstarted\maven_local_repo\maven
ECHO MAVEN_HOME_LOCAL set to: %MAVEN_HOME_LOCAL%
ECHO.

TITLE CMD getting started

CMD
```

- cmd_tomcat_run.bat; opens a command prompt from which it is easy to start and stop your tomcat instance.
- cmd_repo_start.bat; opens a command prompt from which it is easy to start and stop your Hippo repository.
- cmd_cms_start.bat; opens a command prompt from which it is easy to start and stop your Hippo CMS.



Step 9; Installing Jetspeed-2 plugin

Hippo Portal is based on the Jetspeed-2 portal from Apache. In order to use the Hippo Portal plugin you will need to first install the Jetspeed-2 plugin. This is done through the following steps:

- Run the file cmd_project_build.bat
- Run the command in the opened command prompt:

```
maven plugin:download -DartifactId=maven-jetspeed2-plugin -  
DgroupId=org.apache.portals.jetspeed-2 -Dversion=2.1.3
```

Step 10; Installing Hippo Portal Maven 1 plugin

Installing the Hippo Portal Maven 1 plugin can be done from the command prompt, started with the file cmd_project_build.bat, by running the following command:

```
maven plugin:download -DartifactId=hippo-portal-maven-plugin -  
DgroupId=nl.hippo.portal -Dversion=1.07.00
```

There is no need to start a new command prompt, you can use the one opened in step 9.

Step 11; Creating portal project

The portal project contains the basic structure of the portal and the pages contained within the portal. As this is the top level of the portal the database connections are defined here as well as the location the portal will be running from.

A portal project can be created using the following steps:

- create a directory gettingstarted-portal in the directory projects/source
- run the file cmd_project_build.bat
- change to the newly created directory gettingstarted-portal
- run the following command:

```
maven hp:create-portal-project
```

- change the artifact-id in the project.xml from portal-template to gettingstarted.



- change the projects.properties file to:

```
# Location of the web application, e.g. Tomcat.
nl.hippo.portal.server.home=c:/tomcat-5.5

# Portal Database Settings.
nl.hippo.portal.database.default.name=mysql
nl.hippo.portal.database.url=jdbc:mysql://127.0.0.1:3306/helloWorldPortal
nl.hippo.portal.database.driver=com.mysql.jdbc.Driver
nl.hippo.portal.database.user=root
nl.hippo.portal.database.password=password

# The location of the database JDBC driver.
nl.hippo.portal.jdbc.drivers.path=${nl.hippo.portal.server.home}/common/endorsed/mysql-connector-java-5.0.4.jar
```

- change the nl.hippo.portal.server.home to the correct tomcat directory

Step 12; Creating a portal application project

The portal application project contains one or more portlets and their content. The portlets contained within the project are defined in the portlet.xml file which is used for referencing to the portlets in the portal project. The portlet fragments can also be found here, be it the freemarker template files (see Step 20 for more info) or the database objects to retrieve data from the back-end.

A portal application project or pa-project can be created using the following steps:

- create a directory hello-world-pa in the directory projects/source
- run the file cmd_project_build.bat
- change to the newly create directory hello-world-pa
- run the following command:

```
maven hp:create-pa-project
```

- change the artifact-id in the project.xml from default-pa to hello-world-pa

Step 13; Configuring a portlet

Portlets are defined in the file portlet.xml found in the directory pa-project/src/webapp/WEB-INF. Hippo has several predefined portlets that can be used out of the box and some base classes for your own portlet classes. These classes can be found in the hippo-portal-pac jar. The complete sources of Hippo Portal can be downloaded via subversion from

http://svn.hippocms.org/repos/hippo/hippo-portal/tags/Release-PORTAL-v1_07_00.



Frequently used portlets are:

- nl.hippo.portal.cms.portlets.FreeMarkerListContentPortlet
- nl.hippo.portal.cms.portlets.FreeMarkerContentPortlet
- nl.hippo.portal.cms.portlets.XSLTContentPortlet
- nl.hippo.portal.cms.portlets.XSLTListContentPortlet

For this example we are going to use the XSLTContentPortlet, this portlet uses a XSLT file to parse a file found in the Hippo CMS.

In the newly created hello-world-pa/src/webapp/WEB-INF directory go to the file portlet.xml and replace all the content with:

```
<portlet-app
  xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:noNamespaceSchemaLocation="http://java.sun.com/xml/ns/portlet/portlet-
  -app_1_0.xsd" version="1.0">

  <portlet>
    <portlet-name>HelloWorldPortlet</portlet-name>
    <portlet-class>nl.hippo.portal.cms.portlets.XSLTContentPortlet</portlet-
    class>
    <supports>
      <mime-type>text/html</mime-type>
    </supports>
    <portlet-info>
      <title>Hello World Portlet</title>
    </portlet-info>
    <portlet-preferences>
      <preference>
        <name>xslt</name>
        <value>/WEB-INF/xslt/helloWorld.xslt</value>
      </preference>
    </portlet-preferences>
  </portlet>
</portlet-app>
```



Here we define the XSLT the portlet should use. Also the type of portlet is defined in the portlet-class element. The XSLT is a portlet preference for the XSLTContentPortlet. Other portlets will have non or other preferences. The defined XSLT should be placed in the directory hello-world/src/webapp/WEB-INF/xslt and be called helloWorld.xslt. The file should contain the following text:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

  <xsl:template match="/">
    <!-- the title of the document -->
    <h2><xsl:value-of select="/root/title"/></h2>
    <!-- the HTML content of the document -->
    <p>
      <xsl:copy-of select="/root/body/html/body/node()"/>
    </p>
  </xsl:template>

</xsl:stylesheet>
```

The above XSLT will transform a simple XML document to a HTML snippet which will be the output of the portlet. The simple XML document is located in the Hippo CMS.

Step 14; Updating a portlet (psml)

A page in the portal is defined in a psml file which can be found in the following location: getting-started-portal/src/webapp/WEB-INF/pages/___site1. The filename should also be defined in the site map so it is possible to link to the page. For more information on the site map see Step 20. The psml file contains the fragments that define the fragments (windows) to be aggregated on the page. Also the layout of the page is defined. This example uses a default generic layout that will present the portlets below each other. More layout types and examples can be found in the Jetspeed-2 demo and the Hippo Portal demo.



In the example we are going to define a page for the portal with two portlets on it; the Hello World portlet and an example Pick A Number portlet. The PickANumberPortlet is provided by the Demo portal application from the Apache Jetspeed-2 portal. It is a simple game portlet which repeatedly asks you to guess a number, until you get it right. The home.psmml file is already part of the portal so there is no need to add it to the site map. Change the content of home.psmml to:

```
<fragment id="home" type="layout" name="jetspeed-
layouts::SimpleLayout">
  <preference name="ViewPage"><value>fixed-layout</value></preference>
  <preference name="layoutType"><value>default</value></preference>
  <!--
    add page fragments for home page
  -->
  <fragment id="home-HelloWorld" type="portlet" name="hello-world-
pa::HelloWorldPortlet"/>
  <fragment id="home-PickANumberPortlet" type="portlet"
name="demo::PickANumberPortlet"/>
</fragment>
```

Step 15; Building and deploying portal

To build and deploy the portal we need to go to the portal root directory, in this case the gettingstarted-portal directory and execute the following commands in a command prompt started with the file cmd_build_project.cmd:

- maven hp:build-portal
- maven hp:init-portal-db
- maven hp:deploy-portal

Note that sometimes the execution of the init-portal-db command can cause conflicts with your firewall and/or virus scanner.



Step 16; Building and deploying portal application

Building and deploying the hello-world portal application is done with the following steps:

- run file cmd_build_project.cmd
- go to directory hello-world-pa
- run the following command:

```
maven clean war
```

- copy the generated file hello-world-pa/target/hello-world-pa.war to tomcat-gettingstarted/webapps/gettingstarted/WEB-INF/deploy

Every portlet application running on Hippo Portal has to be deployed via the deploy folder of tomcat. Hippo Portal and Jetspeed-2 need to register every portlet application internally in order to invoke portlets. This is done at the startup of tomcat.

Step 17; Importing and deploying external application war

To show how to import and deploy an external application war we are going to use the Jetspeed-2 demo. This demo can be downloaded here:

<http://repo1.maven.org/maven/org.apache.portals.jetspeed-2/wars/demo-2.1.3.war>. You will need to save the file to tomcat-gettingstarted/webapps/gettingstarted/WEB-INF/deploy and rename it to demo.war.

Step 18; Running and viewing portal

To admire your first Hippo Portal you need to first start the Hippo Repository, this can be done by running the cmd_repo_start.bat file. The repository is started by typing:

```
wrapper -c wrapper.conf
```

Stopping the repository can be done by hitting CTRL + C. To test if the Hippo Repository is running open the following url in your favorite browser: <http://localhost:60000/default>. The user is root and the password is password, here you can browse to the site map

(<http://localhost:60000/default/files/default.preview/content/documents/construction>). It is not possible to edit the site map here, see Step 20 on how to edit the site map.

Next you need to start the Hippo CMS, this can be done by running the file cmd cms_start.bat. The Hippo CMS can only run when the Hippo Repository is also running. Starting the CMS is done by typing: `wrapper -c wrapper.conf`, stopping the CMS can be done by hitting CTRL + C. Using the close button on the command



prompt box will also work but it is not guaranteed that all resources are freed up correctly when you do that. To test if the Hippo CMS is running open the url: <http://localhost:50000/> in your browser. The user is root and the password is password. Last thing to start is the tomcat instance tomcat-gettingstarted, this can be done by running the file cmd_tomcat_run.bat. The tomcat instance can be started by typing: 'startup', stopping the instance is done by typing: 'shutdown' from the command prompt. These commands start batch files that are located in the bin directory of your tomcat installation. Tomcat will open a new window, it is best to use the shutdown command as restarting after the use of the close button on the new window can result in problems of locked connections and other locked resources. Go to <http://localhost:8080/gettingstarted> in your favorite browser to admire your work.

Step 19; Editing in Hippo CMS

To show how to edit a file in Hippo CMS we are going to change the text in the Helloworld portlet. The helloWorld.xml file is both located in the Hippo Repository and in the Hippo CMS, in the Repository it is not possible to edit the file this is done in the CMS. The Hippo CMS needs to be running to start editing. Now navigate to Documents (middle top tab) and in the left tree open the node root/documents/example. Select the document named helloworld and open the document editor using the workflow box on the right side, using the item; Edit document. Change the text and save the document. Go to your browser showing your portal and refresh the page to see that the content text has changed. No restart of any of the elements is needed for the change to take effect.

Step 20; Own portlet and portal

Site map

The site map is an important document that is present in the Hippo Repository. It serves primarily as a mapping between (a part of) an URL and a portal page. Secondly a mapping may be present to a node in the repository with which portlets can retrieve their content.

To update your site map you need to first have a version of the xml file outside your repository. Best is to keep it in a directory structure similar to the structure in the repository. Next you need the Hippo WebDav Maven 1 plugin which can be downloaded via the following command:

```
maven plugin:download -DartifactId=hippo-webdav-plugin -DgroupId=hippo-webdav-plugin -Dversion=1.0
```



To upload the site map run the following command in the root directory of your newly created directory structure for the site map. Make sure the repository is running when executing this command:

```
maven webdav:deploy
-Dwebdav.destinationurl=http://localhost:60000/default/files/default.preview/content
-Dwebdav.srcpath=content
-Dwebdav.username=<username>
-Dwebdav.password=<password>
```

More information about the content of the site map can be found at this location: <http://www.hippoportal.org/build/sitemap.html>.

Layouts

The layouts used in this example are the predefined Jetspeed-2 layouts. You can also define your own layouts. Layouts are defined in Velocity scripts located in the portal project in subdirectory `src\webapp\WEB-INF\templates\layout`. More info about Velocity can be found here: <http://velocity.apache.org/>.

Freemarker

"Freemarker is a 'template engine'; a generic tool to generate text output (anything from HTML to autogenerated source code) based on templates. It is a Java package, a class library for Java programmers. It is not an application for end-users in itself but something that programmers can embed into their products."³

In Hippo Portal freemarker is used to define the dynamic pages. The pages are located in the `pa-project` directory, subdirectory `src\webapp\WEB-INF\templates` or its subdirectories. The file extension is `ftl` and the references to these files is located in the `pmsl` files defining the portlet content. The dynamic content and handling of the portlet can be found in the base class `nl.hippo.portal.cms.portlets.FreeMarkerListContentPortlet`, for lists on the page with links and the base class `nl.hippo.portal.cms.portlets.FreeMarkerContentPortlet`, for general content.

Eclipse IDE

Eclipse can be used for development of a Hippo Portal, there are even plugins for freemarker and velocity that will greatly improve the readability of the template files. It is useful to create a classvariable in eclipse pointing to the Maven local repository.

Versioning

Hippo uses subversion as a versioning tool but you are free to choose any tool you like. Not all files should be part of subversion as they are dependent on the installation locations. Also do not add the `.classpath` files of Eclipse projects to your versioning as these are generated by Maven and will be subject to change via that route. Other changes to the `.classpath` file should either be



IPROFS

incorporated with the maven build file or should not be part of the general .classpath file.

Step 21; Security

Security can be arranged by using filters or by using the Jetspeed-2 security. An additional example of a Hippo Portal with a login can be found in the hippo portal demo. The demo can be downloaded using subversion from this location:

http://svn.hippocms.org/repos/hippo/hippo-portal/tags/Release-PORTAL-v1_07_00. You need the demo-pa and demo-portal projects.

Conclusion

After reading this document you know how to create your own Hippo Portal and gained an inside on best practices on how to create a Hippo Portal project.



More information

For more information about **IPROFS** and what we can do for you, you can contact us at +31 (0)23 5476369 or via www.iprofs.nl

References:

[1]

<http://www.hippoportal.org/>

[2]

<http://www.onehippo.com>

[3]

<http://www.freemarker.org>

[4]

<http://jduchess.org>

Author

Régina ten Bruggencate

Régina is a senior java developer with IPROFS. She has ten plus years of experience with Java development in various projects for different clients. These projects vary in length, size and customers. She also has extensive knowledge of teaching and training other Java developers. She is an active member of Duchess 'a community of female java developers'.⁴