OpenCms with Netbeans

Orchestrate OpenCms modules with NetBeans and nbDriva™

Revision 290
March 2014
nbDriva_documentation

componio

Robert-Bosch-Straße 7
D-64293 Darmstadt

www.componio.net

Please use opencms.support@componio.net to get in touch with us.

Table of Content

CHAPTER 1

Purpose

Developing solutions for the open source content management system OpenCms can become cumbersome. Sooner or later you will ask yourself: How do I bridge the gap between the database driven virtual file system and the real file system so that I can use my IDE of choice?

1.1 Plug-in to the rescue

We at componio are fond lovers of the NetBeans IDE. Thus we decided to create a plug-in which enables us to develop for OpenCms in almost the same way that we are developing pure web applications. Most notably to **create, update, import** and **reverse synchronize OpenCms modules**.

In the following chapters you will learn how to configure and build this plug-in as well as how to customize the plug-in to fit your build process and specific needs.

We primarily offer a binary distribution of the plug-in which is ready to use out of the box. The latest version of the installable plug-in can be found at *http://github.com/componio/nbDriva/bin/net-componio-opencms-projectstructure-plugin.nbm*. You may skip directly to chapter 3.

1.2 A technical glimpse

Since OpenCms stores everything in a database (aka. Virtual File System/VFS) and the IDE accesses the file system from the operating system (aka. Real File System/RFS) we need to access the data through the OpenCms APIs.

The often overlooked OpenCms CmsShell is our weapon of choice. It enables the plug-in to execute actions originating in NetBeans in the context of the OpenCms runtime environment and vice versa. Further the CmsShell bridges the gap between the VFS and the RFS enabling the build environment to integrate, annotate and package OpenCms resources automatically.

The diagram below outlines the principle modus operandi of the plug-in.

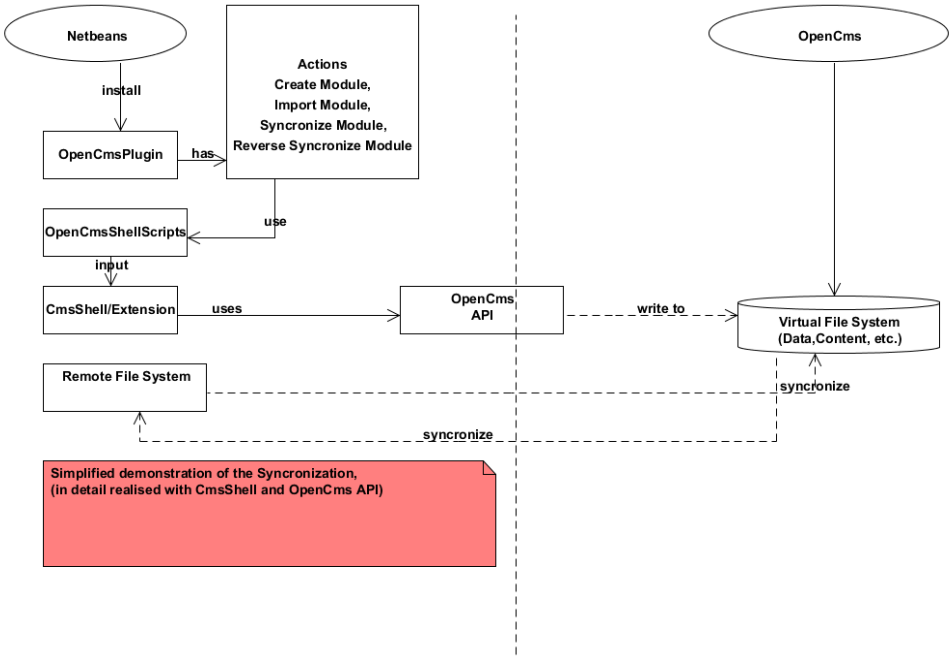

Fig. 1 1: Interaction Overview Diagram

CHAPTER 2

Build the Plug-in

In order to be able to leverage the full potential of the plug-in, we will guide you through the build process. Further we will point out enhancements and alternative settings where appropriate.

2.1 Requirements

- MySQL[1] Installation
- OpenCms[2],[3] Installation 8.5.x
- NetBeans[4] 7.3.x (with Java EE Base plugin)
- the plug-in sources from github[5]

2.2 Plug-in Structure

The plug-in consists of the following components:

- ***OpenCms.project.template***, template used as the default project structure and configuration
- ***net.componio.OpenCms.projectstructure.plugin***, the NetBeans module itself
- ***net.componio.OpenCms.shell.extension***, extended commands based on the CmsShell for OpenCms 8.x, (e.g. the synchronisation between RFS and VFS, synchronization over CMIS interface is also possible)

Please note that modifications to one component may have side-effects which affect the correct functionality of the other components.

2.3 How to build the plugin?

1. Select (*File -> Open Project ->* ***net.componio.OpenCms.projectstructure.plugin)***
2. Right-click on *Project -> Clean and Build*

   Note: To build the plug-in with Netbeans 7.4 following workaround has to be proceeded!(Since version 7.4 following module isn't available per default anymore)
3. Right-click on *Project -> Libraries -> Select Ant -> Remove*
4. Right-click on Project -> *Wrapped Jars -> Add Jar -> Select* ***<Netbeans 7.4 installation directory>/extide/modules/org-apache-tools-ant-module.jar***
5. Right-click on *Project -> Clean and Build Project*

2.4 How to install the plugin?

1. Right-click on Project -> *Create NBM -> Select Tools -> Plugins -> Select Downloaded -> Add Plugins -> Choose* ***<project-directory>/build/net-componio-opencms-projectstructure-plugin.nbm ->****Click Install*

   or
2. Right-click on Project -> *Install/Reload in Development IDE*

   NOTE: After execution of "Install/Reload in Development IDE" the plugin can be just deactivated but not uninstalled.

2.5 Project Template Modifications

Changes to ***OpenCms.project.template*** require the following steps to be taken:

1. Execute the target ***zipme*** in the build.xml of the template which generates the zip file ***OpenCmsProjectStructureTemplateProject.zip*** in return.
2. Replace the zip file in ***net.componio.opencms.projectstructure.plugin*** with the previously generated zip file.

2.6 Shell Extension Modifications

The ***net.componio.opencms.shell.extension*** utilises the CmsShell and OpenCms API. In order to extend the functionality of the plug-in with your own commands please follow along these steps:

1 Create a new class, which implements the interface ***I_CmsShellCommands*** and add your own methods to this class. Additionally you can consult the OpenCms classes ***CmsShell*** and ***CmsShellCommands*** from the OpenCms source code for OpenCms 8[2].

2 Build a jar file of ***net.componio.OpenCms.shell.extension***.

3 Move the jar file to the lib folder of ***OpenCms.project.template***.

4 Create a new CmsShell script in the relative ***scripts*** folder from which you are calling your defined method.

5 Adapt the ***build.xml***, **individual**.***properties or default.properties and buildModuleOperationsWithCmsShell.xml*** according to your needs.

E.g. in the snippet below the target ***prepare_createNewModule*** is used to replace a row in the CmsShell script ***createModule.txt*** before execution.

```
<target name="prepare_createNewModule">
        <replaceregexp file="$scriptDir}/createNewModule.txt"
                       match="login .*"
                       replace="login &quot;
                                ${username}&quot; &quot;
                                ${password}&quot;"
                       byline="true"/>

        <replaceregexp file="$scriptDir}/createNewModule.txt"
                       match="createNewModule .*"
                       replace="createNewModule &quot;
                                ${modulename}&quot; &quot;
                                ${module.version}&quot;"
                       byline="true"/>
</target>
```

5.1 In the next screenshot the target ***run_createNewModule*** is shown in which the class ***CmsShellMain*** is executed and the CmsShell script and the class with the own CmsShell methods are passed to ***CmsShellMain*** as arguments.

```
<target name="run_createNewModule"
        depends="prepare_createNewModule">
         <java classname="main.CmsShellMain">
             <arg line="-webInf ${cmsWebInfDir} -script
                        ${scriptDir}/createNewModule.txt
                        -additional additionalcommands.ModuleCommands"/>
             <classpath path="build">
                 <dirset dir="${build.classes.dir}">
                     <include name="**/*.class"/>
                 </dirset>
                 <pathelement path="${java.class.path}"/>
                 <pathelement path="${javac.classpath}"/>
             </classpath>
        </java>
</target>
```

6. Execute target ***zipme*** in the *build.xml* of ***OpenCms.project.template***
7. Replace the zip file in ***net.componio.openCms.projectstructure.plugin*** with the generated zip file.
8. Add a new action for the context menu with the new functionality in ***net.componio.OpenCms.projectstructure.plug-in*** (see next section).

2.7 Context Menu Modifications

To add a new context action (e.g. a new CmsShell command) you can use the source code of the plug-in as a reference. Additionally, you can check the **NetBeans Utilities API**[6], the wiki page: "**How do I create an Action that is automatically enabled and disabled depending on the selection?**"[7] and the article with a similar topic on ***markiewb's blog***[8].

CHAPTER 3

Cumbersome no more!

Now that we have configured and built our plug-in, the time-consuming process of build, copy and hit the sync button are over for good.

Install the previously built net-componio-opencms-projectstructure-plugin.nbm file with your NetBeans IDE (see *Tools → Plugins*), restart your IDE and you are set to create your first project.

3.1 Create New Project

1. Select (*File* -> *New Project*) in the NetBeans menu bar
2. Choose "*OpenCmsProjectStructureTemplate*"
3. Click *Next* -> enter project name -> click *Finish*

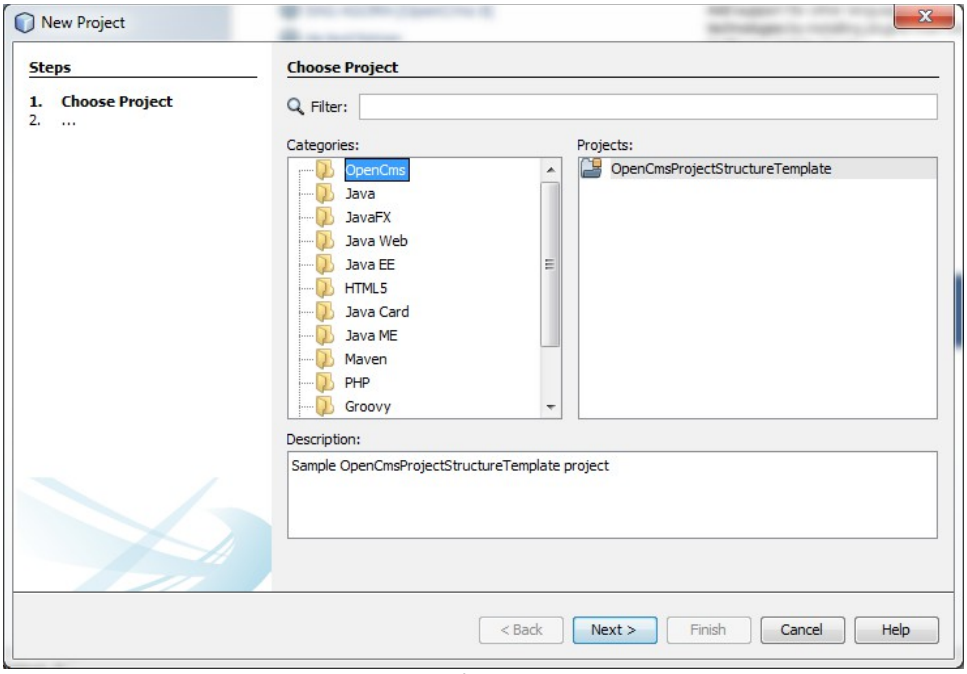

Fig 3 1: NetBeans' New Project Dialogue

3.2 Initial configuration

1. Include the OpenCms libraries in the libraries section
    - Option 1: Run target "ivy-resolve-dependencies" in the project's build.xml. Apache Ivy[9] will be downloaded, installed and the dependencies described in the *<project directory>/ivy.xml* are resolved in conjunction with the Maven repository (per default copied into relative lib directory of the project). For now these are the required OpenCms and MySQL libraries.

      Note: If required please adapt the versions in the *ivy.xml*. The dependencies(version) described in the *ivy.xml* for the OpenCms system must match with your target OpenCms installation.
    - Option 2: Include the required libraries from an OpenCms installation manually. Additionally you need Apache Chemistry libraries(client side) according to the supported Apache Chemistry version by OpenCms.
2. Right-click on *Project →Properties →Libraries →Add Jar/Folder →open the relative lib/default folder →select all jar files →Open →OK*

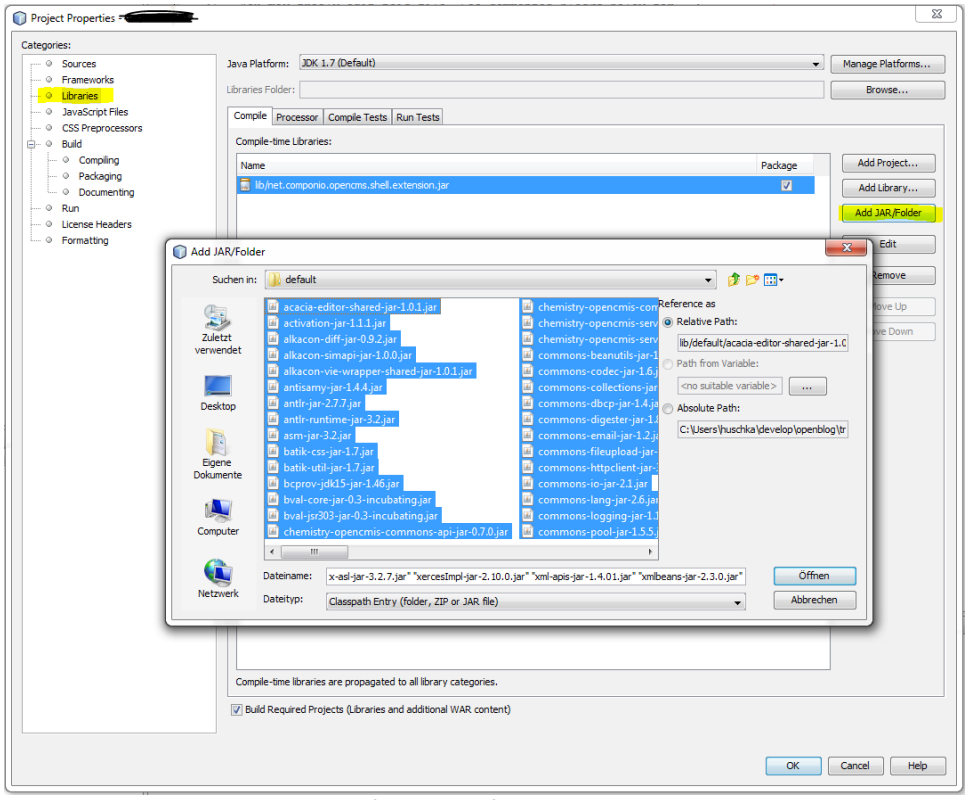

Fig 3 2: NetBeans' Project Library Dialogue

3. Change project-name in ***build.xml*** manually

   <project name="change_me" ...>

4. Change the values in ***individual.properties or default.properties nbDriva.properties*** to meet your build environment. The values are used in ***build.xml*** and ***buildModuleOperationsWithCmsShell.xml***.

   Note: Do not change *buildModuleOperationsWithCmsShell.xml* unless you know how the file affects the plugin. Beyond that configuration values in *default.properties* can be overriden with the values in *individual.properties. I*n *nbDriva.properties* you can define your own default and individual property files!

   Properties in default.properties/individual.properties

   - ***modulename***, the name of the OpenCms module to be created
   - ***modulepath***=/system/modules, path to the OpenCms-modules at the OpenCms-system
   - ***username***=Admin, login data for OpenCms, used in the CmsShell-scripts
   - ***password***=admin, login data for OpenCms, used in the CmsShell-scripts
   - ***module.version***=1.0, version of the exported module, has to be modified manually
   - ***acPackage***=${modulepath}/${modulename}, directory of the actual module at the OpenCms-System
   - ***cmsSync***=../../cmsSync, Syncronization folder, used to synchronize changes between the Remote File System and the Virtual File System(OpenCms-System)
   - ***includedSyncFolders***= paths(separated with a ",") of the Virtual File System, which will be included in the synchronization (can be used to avoid that other directories in the cmsSync path will be deleted at the sync process)
   - ***syncOverCMIS***=false, flag to define if the synchronisation is proceeded via CMIS(Apache Chemistry[10])

- ***cmsRepoURL***=http://localhost:8080/opencms/cmisatom/, URL to the CMIS repository
- ***cmsRepoId***=cmis-offline, repository id of a CMIS repository
- ***scriptDir***=./scripts, directory to the CmsShell scripts

  NOTE: The scripts in this directory are changed dynamically via Ant targets so the scripts in this folder shouldn't be changed manually)
- ***packagePath***=../package, relative package path at the project, used to save the exported module (from VFS)
- ***cmsWebInfDir***= <somepath>, path to the WEB-INF folder of the OpenCms installation you want to work with
- ***moduleDir***=${cmsWebInfDir}/packages/modules, path to the OpenCms modules of the installed OpenCms System.
- ***useExclusionFile***=true, flag
- ***importModules***=..., import files(separated with a ","), the importModules can be selected over a GUI dialog and be imported, so this property is changed dynamically over the GUI.
- ***exclusionFile***=${basedir}/exclusion/exclusion.patterns.txt, Files and directories can be excluded from the sync process via exclusion patterns, which are defined in the exclusionFile.
- ***ivy.install.version***=2.3.0, Apache Ivy version
- ***ivy.jar.dir***=${basedir}/ivy, directory for the Apache Ivy installation
- ***ivy.jar.file***=${ivy.jar.dir}/ivy-${ivy.install.version}.jar
- ***ivy.resolved.libs.dir***=${basedir}/lib, destination folder for the resolved libraries over Apache Ivy

3.3 How to use the plug-in?

Right-click on the project to call one of the following actions:

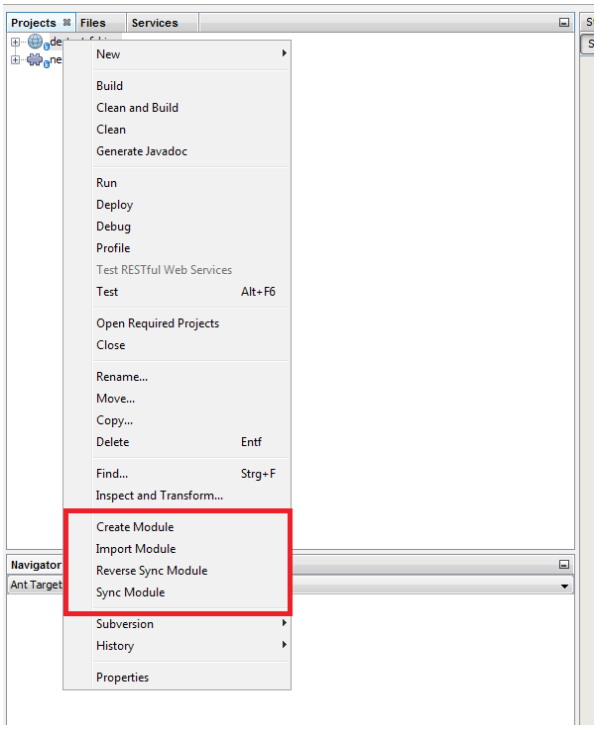

Fig 3 3: Available Plug-In Actions

1. ***Create Module***, the module will be created on the target OpenCms
2. ***Import Module***, can be used to import OpenCms modules over following GUI.

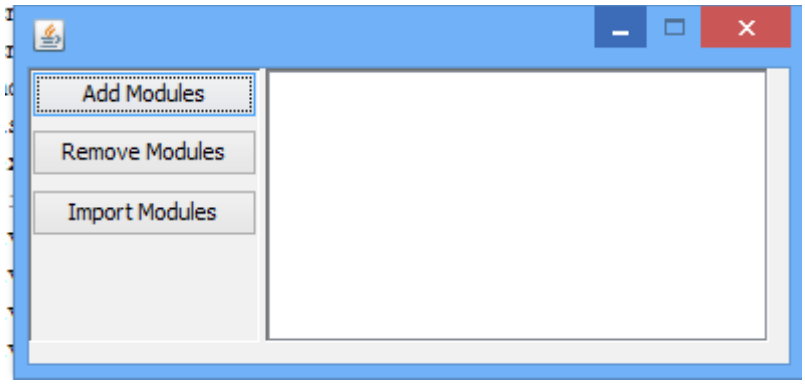

Fig 3 4: Import Modules Dialog

3. ***Reverse Sync Module***, exports the module from the target OpenCms to *<project-directory>/package*
4. ***Sync Module***, two-way *s*ynchronisation between the synchronisation folder(RFS) and the VFS of the target OpenCms via the CmsShell or a one-way synchronisation with Apache Chemistry over CMIS.

Annex A List of References

1: Oracle, MySQL Downloads, 2013, http://dev.mysql.com/downloads/

2: Alkacon, OpenCms download archive, 2013, http://www.opencms.org/de/download/archive.html

3: Alkacon, Local Installation of OpenCms 8, 2013, http://www.opencms.org/en/support/opencms-8-user-manual/installation/index.html

4: Oracle, NetBeans IDE Download, 2013, https://netbeans.org/downloads/index.html

5: componio GmbH, nbDriva - OpenCms with Netbeans, 2013, http://www.github.com/componio/nbDriva

6: Oracle, ContextAwareAction (Utilities API) - NetBeans, 2013, http://bits.netbeans.org/dev/javadoc/org-openide-util/org/openide/util/ContextAwareAction.html

7: Oracle, How do I create an Action that is automatically enabled and disabled depending on the selection?, 2011, http://wiki.netbeans.org/DevFaqActionContextSensitive

8: Markiewb, NetBeans: How to create a context aware action with an icon for the context menu, 2012, http://benkiew.wordpress.com/2012/12/28/netbeans-how-to-create-a-context-aware-action-with-an-icon-for-the-context-menu/

9: Apache, Download Ivy, 2013, http://ant.apache.org/ivy/download.cgi

10: The Apache Software Foundation, Welcome to OpenCMIS, 2013, http://chemistry.apache.org/java/opencmis.html

Annex B List of Figures