

Building from sources

The community distribution of eXo Platform is built very frequently by the open source project members. The corresponding binaries are uploaded to the OW2 forge at the following address http://forge.objectweb.org/project/showfiles.php?group_id=151&release_id=1972 . Although the high frequency allows to get quick bug fixes, it might be needed sometimes to build eXo Platform from the sources. This page gives detailed instructions.

eXo Platform is built using a mix of Maven and JavaScript. Maven is used to compile and download the dependencies. JavaScript is used to package and deploy the project. We selected this language because it allows reflecting immediate changes in the build and because we wanted to reduce the number of languages used in the platform (JavaScript is already used to process AJAX requests).

Create the base structure

Note about Tomcat Our applications are designed in a way that eXoResources has to be deployed before other resources (component resources or custom resources). Hence, we named them so they are in alphabetical order. Unfortunately, Tomcat cannot insure that they will be deployed in a specific order, especially alphabetical order. We already encountered errors because eXoResources was not deployed first, which led us to modify Tomcat. If you encounter such an error, the solution may be to replace the /lib/catalina.jar file of your tomcat, with the one we provide in attachment of this page. This archive contains a modified HostConfig class that sorts the array of files in /conf/Catalina/localhost in alphabetical order. The source file of this class is also available in attachment if you'd like to take a look (line 541), or build tomcat yourself. We used the Tomcat-6.0.16 version of this file for this build. So far, this error only appeared on an Ubuntu 7.10 computer with Jdk 1.5.0_14-b03, but nothing prevents it from appearing on other platforms. The file links are at the bottom of the page

1. Create a "java" directory. We will call it \$EXO_BASE_DIRECTORY. By default, the \$EXO_BASE_DIRECTORY directory should be in your \$HOME directory in linux or in D: drive on Windows.
2. Download and install the JDK (Java Development Kit) 5.0 or higher and Maven 2 in \$EXO_BASE_DIRECTORY. Packages are respectively available from <http://java.sun.com/> and <http://maven.apache.org> web sites.
3. Create "exo-dependencies" and "exo-working" directories in the \$EXO_BASE_DIRECTORY.
4. Download your application server and extract it to exo-dependencies. The name of the extracted directory should match a pattern. The following table gives, for each application server, the download site, the version number and the name of the extracted directory :

Application server	Download site	Version number	Name of the directory
Apache Tomcat (see note below)	http://tomcat.apache.org	6.0+	exo-dependencies/tomcat- \${version}
JBoss Application Server	http://www.jboss.org	4.2.0+	exo-dependencies/jboss- \${version}
OW2 JOnAS	http://jonas.objectweb.org	4.8.4+	exo-dependencies/jonas- \${version}

The \${version} number of your AS is specified in the exoenv script by the CLEAN_SERVER variable. See below in Configuration for more information on the exoenv script.

By the end you should have the following directory structure :

```
java/
  exo-dependencies/
    tomcat-${version}/
    jboss-${version}/
    jonas-${version}/
  exo-working/
  jdk1.5/
  maven2/
```

Check out the source code

Main - Building from sources

- Go to your \$EXO_BASE_DIRECTORY and run the following command :

```
svn co http://svn.exoplatform.org/aliases/head eXoProjects
```

This will check out trunk for all project sources. Alternatively, this [page](#) explains how you can check out a different subset of the main projects directory.

Before you decide what you need to checkout, you may browse the subversion repository at <http://svn.exoplatform.org/projects> .

Configuration

- Copy \$EXO_BASE_DIRECTORY/eXoProjects/tools/trunk/config/maven2/settings.xml to \$HOME/.m2/settings.xml on Unix/Linux or %HOMEDRIVE%\%HOMEPATH% on Windows. Then adapt the configuration to your environment by replacing the tokens @java.dir@ by the absolute location \$EXO_BASE_DIRECTORY. If you do this, maven2/conf/settings.xml will be ignored.

#sign("XWiki.Steph33560") **Note About settings.xml** : don't forget to comment "vietnam" repository, in order to use a closer from you, like <http://maven2.exoplatform.org/rest/maven2>

- Copy the exoenv script.
 - On Linux, Unix or Cygwin Environment, copy \$EXO_BASE_DIRECTORY/eXoProjects/tools/trunk/build/src/main/resources/linux/exoenv.sh to your \$HOME or \$EXO_BASE_DIRECTORY directory. Backup the previous one if existing.
 - On Window And DOS Environment, copy \$EXO_BASE_DIRECTORY/eXoProjects/tools/trunk/build/src/main/resources/window/exoenv.bat to your \$HOME or \$EXO_BASE_DIRECTORY directory. Backup the previous one if existing.
- Update The Configuration
 - Edit exoenv.sh or exoenv.bat file to adapt it to your environment. You should only have to change the PORTABLE_DIR variable at the beginning.

If you have another JVM location, specify it in quotes, e.g: â# set JAVA_HOME="C:\Program Files\Java\jdk1.5.0_12" â#

- Run the command to setup the environment:

On linux or cygwin: source exoenv.sh

To check that it has worked well, open the \$EXO_BASE_DIRECTORY/maven2/conf/settings.xml and check that the file corresponds with your environment, especially the definition of the repositories. Set the balise localRepository to the maven repository directory.

Example : /home/romain/Desktop/exoplatform/portail/exo-dependencies/repository

On window and dos prompt: exoenv.bat

Build and deploy

The exobuild script is used for building and assembling products. Run it with --help argument form options :

```
exobuild --help
```

Use of the exobuild command:

```
exobuild --product=name
         [--version=version]
         [--update]
         [--build]
         [--exclude=modules]
         [--deploy[=server]]
         [--release[=server]]
```

Main - Building from sources

```
[--workflow[=jbpm|bonita]]
[--clean-mvn-repo]
[--database[=dialect]]
[--dbsetup=option]
```

Options:

```
* --product=name      Name of the product you want to build.
                      The possible names are cs,ks,ecm,portal,ultimate,wcm,webos, ...
                      Default is portal.
* --version=number    Allows to specify which version of the product
                      to build such as trunk, tags/2.0, branches/2.0,....
                      Default is trunk.
* --update            Run a svn update before it builds.
* --build             Compile and install the sub projects of the product,
* --exclude           Exclude the given modules (comma separated) from compilation and fetch jars from repository
                      You can specify any module name in all,pc,jcr,ws,tools,ecm,cs,ks,portal.
                      Use
```

this to avoid full build or when a module breaks the build

```
* --deploy=server     Deploy to a given application server. Possible values are: 'all', [jboss, tomcat, jonas, ear, ...]
                      Default is tomcat.
* --release=server    Release
```

for the target application server. Produce a zip named after the current SVN revision.

```
Possible values are: 'all', [jboss, tomcat, jonas, ear, jboss].
Default is tomcat
* --clean-mvn-repo    Clean your local repository of eXo artifacts before building.
* --database=dialect Specify target database dialect. The possible values are [db2v8, postgresql, mysql, sqlserver, ...]
                      This will configure the appropriate JCR dialects and deploy the JDBC driver.
                      Used with --dbsetup=file option, exobuild tries to get database settings in a file named
                      database-configuration.{dialect}.xml
                      Default is hsqldb.
* --dbsetup=option    Use
```

this option with --database option to specify the database setup behaviour.

```
dbsetup=file will use the database and jcr files you provided.
dbsetup=ask allow you to enter the connection url , username and password of the database server.
dbsetup=defaults is the
```

default option

if dbsetup is not specified and will override settings by those defined in Database.js

```
* --workflow=engine   Specify the workflow engine to bundle with the product. The possible values are bonita or jbpm
                      This option is only used
```

for products that use workflow. Default engine is jbpm

```
* --help              To print
```

this help. Also you can use option: '-help' or 'help' or '?'

For Example:

```
#build ecm and all its dependencies and deploy
$exobuild --product=ecm --build --deploy
```

```
#build ecm but exclude jcr and portlet container, the jcr and pc artifact will be
#downloaded from the maven server
$exobuild --product=ecm --clean-mvn-repo --exclude=jcr,pc --build --deploy
```

```
#build ecm and all its dependencies, to exo-tomcat, exo-jonas, exo-jboss and create the release zip file
$exobuild --product=ecm --build --release
```

#sign("XWiki.Steph33560") Note about release : release needs a working svn command as it grabs the revision number to build an archive. Sometimes you can have problems during build, due to outdated libraries in maven repository. Clean the suspicious libraries, or, at the very worst, all maven repository (in this last case the build will re-download all the dependant libraries and therefore will take much more time than normal build). For example, the portlet container 2.0 is dependant on portlet API 2.0. But this API is still in draft version, and so on the number version won't change until final 2.0, even if content of library do. If portlet container doesn't build. Try to remove "javaxportletportlet-api2.0" in maven repository and re-build.

Single module build

Main - Building from sources

If you want to build a single module, you can use exoproject command:

```
$cd ~/java/eXoProjects/portal/trunk
$exoproject --deploy=module
```

This will build the module and deploy it to your exo-working server.

```
#sign("XWiki.Steph33560")
```

Launch Tomcat...

On Linux or Cygwin environment

- Go to \$EXO_BASE_DIRECTORY/exo-working/exo-tomcat/bin directory
- Change the rights on .sh files to make them executable : `chmod +x *.sh`
- Run the command : `eXo.sh run`

On Windows environment

- Go to %EXO_BASE_DIRECTORY%\exo-working\exo-tomcat\bin directory
- Run the command : `eXo.bat run`

â#| or launch JOnAS 5 (yes it does)

Please see [JOnAS wiki](#) to find help on building eXo and JOnAS 5 ! On Linux or Cygwin environment

- Go to \$EXO_BASE_DIRECTORY/exo-working/exo-tomcat/bin/unix directory
 - Adapt eXo.sh to your environment
 - Change the rights on .sh files to make them executable : `chmod +x *.sh`
- Go to \$EXO_BASE_DIRECTORY/exo-working/exo-tomcat/bin directory and run the following commands :
 - `ln -s unix/eXo.sh`
 - `chmod +x eXo.sh jonas setenv unix/*.sh`
- Then start exo-jonas with start parameter with this command :
 - `/eXo.sh start`

On Windows environment (-not tested-)

- Go to %EXO_BASE_DIRECTORY%\exo-working\exo-tomcat\bin\unix directory
 - Adapt eXo.sh to your environment
- Go to %EXO_BASE_DIRECTORY%\exo-working\exo-tomcat\bin directory and run the following commands :
 - `copy unixeXo.sh .`
- Then start exo-jonas with start parameter with this command :
- Open your Internet browser and reference the URL <http://localhost:8080/portal/>.

Configure Eclipse for eXoProjects

This documentation will help you configure Eclipse for the eXoProjects environment.

Generate Eclipse Configuration

Before generating the eclipse configuration, you need to have built successfully the ecm or portal product.

Create the Eclipse projects

Main - Building from sources

Go to \$EXO_BASE_DIRECTORY/java/eXoProjects/\$module/trunk directory and run the command

```
mvn eclipse:eclipse
```

This command will generate the files .classpath , .project and .setting in each maven project

For Advanced developer

- Usually , you only need to work on a module such portal, ecmâ# So you only need to generate the eclipse configuration for a module only. When you create this configuration, you only have the projects within the module are linked to each other. You can also generate configuration for 2 or 3 different modules and import them into the same Eclipse workspace. But you won't have the feature like refactor, code link....
- If you want generate the configuration for more than one module, create a pom.xml in eXoProjects dir and enter the modules you want to work on. In the pom.xml, you should have some thing like:

```
[...]
<modules>
  <module>kernel/trunk</module>
  <module>core/trunk</module>
  <module>portal/trunk</module>
</modules>
[...]
```

In the eXoProjects directory, run the command

```
mvn eclipse:eclipse
```

To generate the eclipse configuration

Configure Eclipse

- Download the latest Eclipse version from <http://www.eclipse.org> .
- Launch eclipse. Specify a location for your workspace.
- Open "Window" => "Preferences...". In the menu, go to "Java" - "Build Path" - "Classpath Variables" and create a classpath variable named "M2_REPO" with the value of your maven repository location. It should be "D:/java/exo-dependencies/repository".
- Import Projects
 - In the menu bar choose File => Import. Choose General -> Existing Projects into Workspace.
 - In the Select Root directory field, enter \$javaDir/eXoProjects/\$module/trunk. Click refresh to refresh the list of the available projects
 - Once you have selected the directory, a list of projects should appear. Select all and make sure that "Copy projects into workspace" is NOT selected before finishing.
 - Projects are now imported to Eclipse.
- Configure Working Sets
 - In the "Package Explorer" window, click on the small down arrow, at the top right of the panel.
 - In the list that appears, select "Top Level Elements" - "Working Sets".
 - In the same list, click "Configure Working Sets..." and create a new working set with the name eXoProjects/\$module/trunk. This name can change in the future, depending on which version you are working on.
 - Select the projects with the id exo.\$module.... from the list under, and click "Finish".
 - In the "Select Working Sets" window, check the checkbox in front of the Working Set we've just created, and click OK to finish.
- Configure Code Style And Formater
 - Open "Window" - "Preferences..." and navigate in the menu to : "Java" - "Code Style" - "Code Templates". Click import and choose the path \$javaDir/eXoProjects/tools/trunk/config/EclipsePreferences/Java-CodeStyle-CodeTemplates.xml
 - In the new template, select "Code" - "New Java files" and click "Edit..." to change your name, email address...
 - Navigate to "Java" - "Code Style" - "Formatter". Click import and enter \$javaDir/eXoProjects/tools/trunk/config/EclipsePreferences/Java-CodeStyle-Formater.xml
 - Navigate to "General" - "Editors" - "Text Editors" : and change the following values :

Main - Building from sources

- Displayed tab width : 2
- Check Show print margin and set the column to 100
- Check Show line numbers

Click "OK" to finish and "Yes" if a full rebuild is required.