Leading Open Source Middleware

JOnAS 5 Developer guide

JOnAS Team ( )

- March 2009 -

Table of Contents

This guide explains how to modify the JONAS source code in order to contribute to the jonas development or to modify some modules.

Chapter 1. Developing JOnAS

1.1. Getting JOnAS From the SVN Repository

Anyone can check out source code from the SVN server using the following command (for GUI SVN client use, configuration values are the same as for command line use):

svn checkout svn://svn.forge.objectweb.org/svnroot/jonas/jonas/trunk/

It is also possible to retrieve a particular branch or a particular tag. For example:

svn checkout svn://svn.forge.objectweb.org/svnroot/jonas/jonas/branches/jonas_4_10

svn checkout svn://svn.forge.objectweb.org/svnroot/jonas/jonas/tag/JONAS_5_1_0_M3

This will get the 3 modules :

jonas	source of JOnAS server
jonas_doc	documentation of JOnAS (in docbook format)
jonas_tests	Miscelaneous JOnAS tests

Access for developpers is available using :

svn checkout svn+ssh://developername@svn.forge.objectweb.org/svnroot/jonas/jonas/trunk/

1.2. Building JOnAS From Source.

1.2.1. Requirements

1.2.1.1. JDK

A Java SE 5 [http://java.sun.com/javase/downloads/index_jdk5.jsp] is required to build JOnAS. Make sure that the JDK used to build JOnAS is compliant with the new Java 5 features.

1.2.1.2. Maven

The maven tool is used with `pom.xml` files to build JOnAS. This tool is available at http://maven.apache.org. The 2.0.7 or later version is recommanded

1.2.2. Optional Requirements

1.2.2.1. Eclipse

The JOnAS project provides .project and .classpath for Eclipse 3.1 or greater. A project is ready to use once the source has been imported using the Eclipse tool. Eclipse tool is available at http://www.eclipse.org.

1.2.2.2. Eclipse Plugins

1.2.2.2.1. Checkstyle Plugin

The eclipse-checkstyle plugin is used to check the javadoc of JOnAS project. A warning will print if the JOnAS coding convention is not used. This plugin is available at http://eclipse-cs.sourceforge.net.

1.2.2.2.2. AnyEdit Plugin

As part of the JOnAS coding convention, the use of tabulation characters is disallowed. Files should contain only spaces. The AnyEdit plugin allows tabs to be converted to spaces when saving the file. Also, trailing spaces can be removed automatically.

This plugin is available at http://andrei.gmxhome.de/anyedit/.

1.2.2.2.3. Maven 2 plugin

To compile JOnAS under eclipse, use m2eclipse plugin available at http://m2eclipse.codehaus.org/.

1.2.2.2.4. Subversion plugin

JOnAS uses subversion as revision control system. The use of stable version of Subversive is advised. This plugin is available at http://www.polarion.org/index.php?page=overview&project=subversive.

1.2.3. Compiling JOnAS

To compile JOnAS, launch the command **mvn** in the root directory of the project (named JOnAS by default) being launched.

Note

The default maven goal is install if not specified.

Once the command has been run successfully, the maven artifacts generated by maven are available in the maven local repository. The `target` directories contain the generated jars or assemblies.

Note

Bundles that are used in **jonas** command, must be updated manually from local repository to `$JONAS_ROOT/lib` directory if they have been modified. Please refer to `$JONAS_ROOT/bin` for available commands.

mvn clean install is used to clean and regenerate classes.

1.2.4. Maven assembly

JOnAS build generates several assemblies. Assemblies are located in the `assemblies` folder.

1.2.4.1. End-User assembly

The assembly contains examples and is available with the zip or tgz format in the `assemblies/jonas-osgi/target` folder. The unzipped assembly folder should directly be used as `$JONAS_ROOT.` Moreover, the `/lib` directory of this `$JONAS_ROOT` contains all the bundles of the assembly.

Note

Lauchning JOnAS in developper mode (see Section 2.1, “Running the JOnAS server.”) will set the use of bundles stored in maven local repository.

A common way to set JONAS_ROOT is to make a symbolic link (linux only) on the directory :

ln -s .../assemblies/jonas-osgi/target/jonas-osgi-...-bin.dir/jonas-osgi-... ***your_jonas_root_path***

You can alternatively copy all this tree under your target JONAS_ROOT directory.

1.3. Code Conventions in JOnAS

Contributions should follow the JOnAS code convention. A good document to begin with is Java code convention [http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html]. Other conventions are also listed in this document.

In addition, JOnAS uses tools to check the compliance: the checkstyle plugin [http://checkstyle.sourceforge.net/] and the eclipse checkstyle plugin [http://eclipse-cs.sourceforge.net/]. The configuration settings are available on JOnAS SVN [http://forge.objectweb.org/plugins/scmsvn/index.php?group_id=5].

1.3.1. File Organization

1.3.1.1. Header

All files should have a header that contains the LGPL and the date.

If a file is modified, the modification year should be appended to the existing year, which is the year it was initially created. For example, if the create date is '1999' or '2004' it should be edited to '1999-2006' or '2004-2006', respectively.

Also, the tag $Id: code_convention.xml 314 2006-04-04 09:39:43Z pinheirg $ should be added. The following is a header example:

```
/**
 * JOnAS: Java(TM) Open Application Server
 * Copyright (C) 1999-2009 Bull S.A.S.
 * Contact: jonas-team@ow2.org
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with this library; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
 * USA
 *
 * Initial developer(s):
 * --------------------------------------------------------------------------
 * $Id:code_convention.xml 13113 2008-03-11 10:58:50Z eyindanga $
 * --------------------------------------------------------------------------
 */
```

1.3.1.2. Imports

Imports should reference a valid class name, instead of using wildcard imports. Wildcard imports are not authorized.

For example, if the interface and class List and ArrayList are used, the imports should not be as follows:

```
import java.util.*;
```

The imports should have each class as follow:

```
import java.util.List;
import java.util.ArrayList;
```

The classes should not have an unused import.

Note

The Eclipse IDE provides facilities to do this job. There is the option Organize Imports (**Shift+Ctrl+O**) in the menu Source that correctly inserts the imports and removes the unused imports. However, this option does not work well with 'import static'.

1.3.1.3. Class and Interface Declarations

The class and interface names should begin with an uppercase letter. Also, each class and interface has an @author tag in the comment. For example:

```
/**
 * This is an example that shows a class/interface declaration.
 * @author Loris Bouzonnet
 * @author Stephane Zeng
 */
public class ClassExample implements InterfaceExample {
}
```

1.3.2. Indentation / whitespace

1.3.2.1. Indentation

The space character is used instead of the tab character. The number of spaces for an indent is *4 spaces*.

Wrapping a single source line into multiple lines should follow the Java code convention [http://java.sun.com/docs/codeconv/html/CodeConventions.doc3.html#248].

1.3.2.2. whitespace

Any trailing spaces should be removed. Eclipse provides a plugin that removes the trailing spaces and converts the tab into spaces. The plugin is AnyEdit [http://andrei.gmxhome.de/anyedit/].

Use whitespaces in for() loop, while(), when concatenating strings. One space should be added before the operator and another after the operator. For example, the correct syntax is:

```
for (int i = 0; i < arTest.length; i++) {
    String strResult = "The element " + i + " has the value " + arTest[i];
}
```

The following code does not adhere to the convention:

```
for (int i = 0; i< arTest.length; i++) {
    String strResult = "The element "+ i+" has the value "+arTest[i];
}
```

1.3.3. JavaDoc Comments

All methods and attributes (including protected and private) must have a comment. The parameters, the exceptions thrown, and the method return should have a comment in the method comment. For example:

```
/**
 * This is an example that is used in the JOnAS Code Convention.
 */
private int intValue;

/**
 * This is an example method to show a class comment.
 * @param a an example of parameter.
 * @param b other example of parameter.
 * @return the method result.
 * @throws Exception the exception thrown by the method.
 */
public int add(final int a, final int b) throws Exception {
    return a + b;
}
```

1.3.4. Statements

1.3.4.1. If/else

Braces must be used in the if/else blocks, even if there is a single statement. To illustrate:

```
if (true) {
    doThis();
}
```

The following is not allowed:

```
if (true)
    doThis();
```

The position of the braces should be the same as in the first example. The following format is incorrect:

```
if (true)
{
    test1();
    test2();
}
```

1.3.4.2. Try/catch

All exceptions require a statement; no silent catching is allowed. For example:

```
try {
    doThis();
} catch (Exception e) {
    // should not occur
}
```

A logger can be used:

```
try {
    doThis();
} catch (Exception e) {
    logger.logDebug("Exception while doing .....", e);
}
```

1.3.4.3. Naming Conventions

1.3.4.3.1. Static Final Attributes

Declarations are static final, not final static. This is a JLS recommendation.

1.3.4.3.2. Constants

Constants should be static and final, and should adhere to the following:

```
'^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$'
```

1.3.4.3.3. No Magic Numbers, Use Constants

Constants must be used in the code and magic number must be avoided. For example, the following is not allowed:

```
private int myAttribute = 5;
```

The correct format is:

```
/**
 * Default value
 */
private static final int DEFAULT_VALUE = 5;

/**
 * This attribute is initialized with the default value
 */
private int myAttribute = DEFAULT_VALUE;
```

1.3.4.3.4. Attribute Name

The attribute name should not have an underscore (_). The _ is valid for constants that are in uppercase.

Use pValue and mValue instead of p_Value and m_Value.

The pattern for attribute name is:

```
'^[a-z][a-zA-Z0-9]*$'
```

1.4. Contributing to JOnAS

1.4.1. Mailing Lists

Developers wanting to contribute information about JOnAS can share their thoughts via the JOnAS mailing list.

The steps necessary for subscribing to the list are described at the following url : http://jonas.objectweb.org/wws/info/jonas [http://jonas.objectweb.org/wws/infos/jonas]

Mailing lists for JOnAS are available at :

http://forge.objectweb.org/mail/?group_id=5

1.4.2. Ideas for Contributing

There are many ways to contribute to JOnAS. New ideas are also welcome.

The following is a list of some of the ways to make contributions:

- Documentation: Improve or add to the existing documentation, create new chapters, translate, etc.
- Code: Some glue could be added so that JOnAS could be integrated in other servers.
- Tests: Add new tests to the current test suite.

Chapter 2. Testing JOnAS

2.1. Running the JOnAS server.

2.1.1. Requirements

Review the requirements discussed in Section 1.2, “Building JOnAS From Source.”

2.1.2. Running

2.1.2.1. Exporting environment variables

Environment variables to export are, `$JAVA_HOME`, `$JONAS_ROOT`, `$JONAS_BASE`. If `$JONAS_BASE` is not set, then it will point to `$JONAS_ROOT`.

Caution

Please ensure that `$JONAS_ROOT`\bin is added to the system `PATH.`

2.1.2.2. Start/Stop JOnAS

For more information see jonas command [command_guide.html#commands.jonas]

2.2. Using JOnAS Examples.

2.2.1. Getting from SVN

The examples are available in `jonas_tests` module of `jonas` trunk. Please Refer to the section Section 1.1, “Getting JOnAS From the SVN Repository”.

2.2.2. Compiling the Examples

2.2.2.1. Requirements

Before running the examples, be sure to follow the requirements for compiling and running these JOnAS examples.

2.2.2.2. Compile

The ant tool is used to build the examples. Each example is associated a `build.xml`for individual compilation. All the examples can be compiled by launching the **ant** task in `jonas_tests` root folder.

After having compiled an example, putting the generated archive(war, ear) in the `$JONAS_ROOT/deploy` folder will allow JOnAS to deploy it. If JOnAS is running the archive will be deployed automatically, if not, it will be deployed on startup.

Here is an example of log messages for a deployment on startup.

```
2008-03-11 10:14:41,734 : DeployableMonitorService.doStart : Use the deploy directories
 '[]', development mode is 'true'
2008-03-11 10:14:41,796 : DeployableMonitor.detectNewArchives : Deployables to deploy at
 startup: [[c:\cluster\JONAS5-OSGI-BIS\deploy\jonas-ctxroot.war, c:\cluster\JONAS5-OSGI-BIS
\deploy\jonasAdmin.war]]
```

An example of log messages for hot deployment(when JOnAS is already running)

```
2008-03-11 10:14:41,812 : J2EEServer.info : JOnAS server 'jonas' RUNNING
2008-03-11 10:20:54,437 : Rar.processRar : Starting deployment of /c:/cluster/JONAS5-OSGI-
BIS/work/apps/jonas/earsample_2008.03.11-10.20.52.ear/ra-sample.rar
2008-03-11 10:20:54,687 : Rar.processRar : /c:/cluster/JONAS5-OSGI-BIS/work/apps/jonas/
earsample_2008.03.11-10.20.52.ear/ra-sample.rar available
2008-03-11 10:20:54,734 : JOnASEJBService.checkGenIC : JOnAS version was not found in the
 '/c:/cluster/JONAS5-OSGI-BIS/work/apps/jonas/earsample_2008.03.11-10.20.52.ear/secusb.jar'
 manifest file. Auto-generating container cla
sses...
2008-03-11 10:20:56,750 : GenIC.<init> : GenIC for JOnAS 5.0.2-SNAPSHOT: 'EarOp'
 generation ...
2008-03-11 10:20:56,921 : GenIC.<init> : Sources classes successfully generated for 'EarOp'
2008-03-11 10:20:56,937 : GenIC.compilClasses : Compiling Interposition classes ...
2008-03-11 10:20:59,734 : GenIC.compilClasses : Sources classes successfully compiled with
 Eclipse compiler.
2008-03-11 10:20:59,765 : GenIC.compilClasses : Running fastrmic
2008-03-11 10:21:00,109 : GenIC.compilClasses : Stubs and Skels successfully generated with
 fastrmic for rmi/jrmp
2008-03-11 10:21:00,343 : GenIC.clean : Deleting [D:\DOCUME~1\zengeyls
\LOCALS~1\Temp\genic20567.tmp\org\ow2\jonas_gen\org\objectweb\earsample\beans\secusb
\JOnASEarOp_1317676915Home.java, D:\DOCUME~1\zengeyls\LOCALS~1\Temp\gen
ic20567.tmp\org\ow2\jonas_gen\org\objectweb\earsample\beans\secusb
\JOnASEarOp_1317676915Remote.java, D:\DOCUME~1\zengeyls\LOCALS~1\Temp\genic20567.tmp\org
\ow2\jonas_gen\org\objectweb\earsample\beans\secusb\JOnASEarOp_13176769
15LocalHome.java, D:\DOCUME~1\zengeyls\LOCALS~1\Temp\genic20567.tmp\org\ow2\jonas_gen\org
\objectweb\earsample\beans\secusb\JOnASEarOp_1317676915Local.java, D:\DOCUME~1\zengeyls
\LOCALS~1\Temp\genic20567.tmp]
2008-03-11 10:21:01,359 : JContainer.addBean : EarOp available
2008-03-11 10:21:01,390 : EarDeployer.deployWARs : There are WAR files in the
 EAR 'EARDeployableImpl[archive=c:\cluster\JONAS5-OSGI-BIS\work\apps\jonas
\earsample_2008.03.11-10.20.52.ear]' but the web service is not available
2008-03-11 10:21:01,421 : EarDeployer.deployEAR : 'EARDeployableImpl[archive=c:\cluster
\JONAS5-OSGI-BIS\deploy\earsample.ear]' EAR Deployable is now deployed
```

2.2.3. Running Examples

For running an example, please refer to the README file in root folder of each example.