JASMINe Developer's Guide

JASMINe Developer's Guide

Table of Contents

Chapter 1. Building JASMINe

1.1. Software requirements

- Eclipse 3.2 (or more)

- Ant 1.6.5

- JDK 5.0

- JOnAS 4.8.X

1.2. Getting the sources

1.2.1. Structure of the repository

The structure is structured as follows:

```
trunk/
    assembly/            <- build.xml required files
    jasmine-control/     <- Central node module
    jasmine-doc/         <- Documentation
    jasmine-gui/         <- UI
    jasmine-jade/        <- JADE Bundles
    jasmine-probes/      <- Monitoring System
    build.xml
```

1.2.2. Command line

Svn command line client is used to checkout the sources:

```
>$ svn checkout svn://svn.forge.objectweb.org/svnroot/jasmine/jasmine/trunk jasmine
A    jasmine/assembly
A    jasmine/assembly/jasmine-control
A    jasmine/assembly/jasmine-control/ant
A    jasmine/assembly/jasmine-control/ant/README
A    jasmine/assembly/jasmine-control/ant/build.xml
A    jasmine/assembly/jasmine-control/rars
A    jasmine/assembly/jasmine-control/rars/autoload
...
A    jasmine/jasmine-probes/jython
A    jasmine/jasmine-probes/jython/manager.py
A    jasmine/jasmine-probes/jasmine-probe-jython.sh
A    jasmine/jasmine-probes/.settings
A    jasmine/jasmine-probes/.settings/org.eclipse.jdt.core.prefs
A    jasmine/build.xml
Checked out revision 184.
```

This will checkout all the JASMINe code base (Control node components, JADE bundles and wrappers, UI) in a folder named `jasmine`.

1.2.3. Eclipse Subclipse/Subversive

In the SVN Repository Exploring perspective, add a new repository location with the URL `svn://svn.forge.objectweb.org/svnroot/jasmine/jasmine/trunk` (or with 'svn:ssh' for a developer access).

Checkout the trunk as a new project (not Java project). Then, simply import the projects inside in your workspace.

It's important to keep the Jasmine projects structure, otherwise the JASMINe build process will not work.

Note

Use the protocol svn+ssh (instead of svn) for developer access. Anonymous users are unable to commit changes.

1.3. Ant process

Prior to the build execution, the following environment variables must be set:

- `JONAS_ROOT` must points to a valid JOnAS 4.8.X installation directory.

Executing **ant** in the `jasmine/` directory will compile everything and produce the distributable binaries in the `output/dist/` directory:

```
>$ ant jar
Buildfile: build.xml

init:

jar-probes:
...
parent:
     [copy] Copying 1 file to /home/sauthieg/tmp/test-sub/jasmine/output/dist

jar:

BUILD SUCCESSFUL
Total time: 7 minutes 59 seconds
```

This command creates `jasmine-doc-VERSION.zip`, `jasmine-jade-VERSION.zip` and `jasmine-control-VERSION.zip`, which contain all the JASMINe binaries (except the UI).

The structure of the `jasmine-control.zip` file is:

```
jasmine-control.zip
        jonas/
          lib/ext/
              jasmine-shared.jar
          rars/autoload
              drools.rar
          ejb3s/
              jasmine-rules.jar
              jade-ejb.jar
        jasmine_rules/
              defaultRules.drl.xml
              test.drl.xml
        build.xml
        README
```

It contains a `build.xml` file which will ease the install of the components in a JOnAS server.

The file `jasmine-jade.zip` contains:

```
jasmine-jade.zip
        jade-jmx.zip
        jadenode.tar.gz
        jadeboot.tar.gz
        repository/
            ...
```

The use of these files is explained in the user manual.

1.4. Building the UI

Open the `jasmine.product` file from the JASMINe.editor project. It's important to take care of selecting all the necessary plug-ins. To launch it from eclipse click on 'Launch the product'. The conf folder should be visible from the working directory (it's possible to change it in the Run... wizard), usually inside the JASMINe project folder.

To get an independent eclipse RCP client, just click on 'Eclipse Product export wizard' on the overview tab of the jasmine.product.

Chapter 2. Introducing JASMINe

2.1. Structure of JASMINe

JASMINe is structured in 4 parts, each part having one purpose.

2.1.1. jasmine-control (The control node)

A JOnAS will be running in the control node running two EJBs. In this part the following projects are included:

- JasmineShared: Common libraries.

- DroolsResourceAdapter: The resource adapter for the Drools Rules Engine.

- JasmineRules: Generates the EJB that acts as a server for the probes.

- JasmineConnection: Generates the EJB that acts as a server of the GUI and the JADE-JMX bridge.

2.1.2. jasmine-probes

Each node of the JASMINe network will be monitoriced using some probes that will send to the control node in XML format the results of the probes. The package generated is used after for creating its own wrapper.

2.1.3. jasmine-jade

The JADE wrappers (or bundles) are generated in this part, and are copied directly to the repository in the build process. The repository must be accessible from all the JASMINe nodes. These wrappers contain the software and the java classes to configure and launch it. Projects:

- JasmineWrapper: Contains the common files used by the wrappers.

- jasmine-bundle-*: Extends the JADE framework for adapting it to each wrapper.

2.1.4. jasmine-gui

The user interface uses the EMF framework. The projects are:

- JASMINe

- JASMINe.edit

- JASMINe.editor

- JASMINe.jar

2.2. Relations between projects

Part	Requires	Project	Build
jasmine-control	-	JasmineShared	jasmi
jasmine-shared.jar	DroolsResourceAparter	drools-ra.jar drools.rar	
jasmine-shared.jar drools-ra.jar	JasmineRules	jasmine-rules.jar	

-	JasmineConnection	jade-jmx.zip jade-ejb.jar	
jasmine-probes	-	jasmine-probes	jasmi
jasmine-jade	-	JasmineWrapper	jasmi
jasmine-bundle-api.jar	jasmine-bundle-apache	apache2-2-2.jar	
jasmine-bundle-api.jar	jasmine-bundle-jonas-base	jonasbasewrapper.jar	
jasmine-bundle-api.jar	jasmine-bundle-jonas-root	jonasrootwrapper.jar	
jasmine-bundle-api.jar	jasmine-bundle-probes	monitoringSystem.jar	
jasmine-gui	-	JASMINe	-
-	JASMINe.edit	-	
-	JASMINe.editor	-	
-	JASMINe.jar	-	

Note

drools-ra.jar and jasmine-bundle-api.jar are only for internal use.