JORAM Administration

Jeff Mesnil

jmesnil@inrialpes.fr

Describes JORAM administration.

JORAM is an Open Source JMS server. The administration step of JMS has been deliberately kept away from specifications due to the wide range of existing MOM products. Thus, there are no standard way to administrate JMS. The aim of this paper is to describe how this will be done in JORAM. It first states the requirements for JORAM administration, then it defines the main interfaces of it. Next it shows the internal architecture of administration in JORAM and finally describes the tools used to interact with it.

1. Requirements

JORAM needs a new Administration process. The aim of this section is to describe what we mean by JORAM administration, what we expect from it and what we do not.

1.1. Administration

TODO (what do we mean by "administration", difference/comparison with monitoring/management)

1.2. What Administration include

1.2.1. Connection to a server

JORAM administration should be able to

- **connect to a server** given a server URL and a login/password to authenticate the administrator
- **disconnect** from a previously connected server

1.2.2. Users actions

Once connected to a server, an administrator may

- **create/retrieve/delete** a **user**
- **test existence** of a given **user**

1.2.3. Destination actions

- **create/retrieve/delete** a **queue**
- **create/retrieve/delete** a **topic**
- **get names** of all **queues**
- **get names** of all **topics**
- **get names** of all **destinations**
- **test existence** of a given **destination**

1.2.4. Destination factories actions

- **create/retrieve/delete** a **queue connection factories**
- **create/retrieve/delete** a **topic connection factories**
- **get names** of all **queue connection factories**
- **get names** of all **topic connection factories**
- **get names** of all **destination connection factories**
- **test existence** of a **destination connection factory**

1.2.5. Rights actions

- **set/unset read** right for a given **user** on a given **destination**
- **set/unset read** right on a given **destination** for all users
- **test read** right for a given **user** on a given **destination**
- **set/unset write** right for a given **user** on a given **destination**
- **set/unset write** right on a given **destination** for all users
- **test write** right for a given **user** on a given **destination**

It has to be noted that some of these actions are redundant. For example, getting names for all destinations is identical to getting names for all queues *and* all topics. It remains to be seen if such convenient actions are useful or not...

1.3. What Administration does not include

TODO (describe what will not be taken into account)

2. External specifications

There are a wide range of ways to administrate JORAM. For example, one user will need a command line interface to integrate JORAM administration in scripts, another user may need a graphical user interface to interact with it and another one would like to use JMX to manage JORAM as part of a larger system..

Whatever these users interact with, the actions they want are identical, only differing in the way they are presented to them. Thus, we need to agree on an external specification for administrating JORAM, so that CLI users, GUI users or JMX users will share *common features* but *different and independent interactions* with these features.

2.1. Administration interfaces

TODO (UML class and sequence diagrams, Javadoc?)

3. Internal Architecture

TODO (describe `fr.dyade.aaa.joram.admin.Admin`, `fr.dyade.aaa.joram.mom.proxies.JmsAdminProxy` as well as requests in `fr.dyade.aaa.joram.mom.jms`)

4. Administration tools

We foresee three ways to administrate JORAM:

- a Command Line interface
- a Graphical User Interface
- JMX

4.1. Command Line Tools

TODO (scripted language or Java command line?)

4.2. Graphical Tools

TODO (AWT, Swing, HTML, SWT,...?)

4.3. JMX Tools

TODO (describe JAdmin)