QAT Global
Code Sample

JMS

Contents

Introduction

As part of QAT training, we are continuing to develop several code samples. These code samples illustrate various technical as well as business aspects. These samples will help the developer get a quick understanding of the technology and framework. This will considerably reduce the learning time. The samples can be used as a reference to develop the application.

Prerequisites

In order to extract the maximum benefit out of the samples, it is expected that the developers have

- undergone Basic Java training
- undergone Basic SOA training
- undergone Basic Spring training – Batch, WebServices, AOP and JUnit
- completed Development environment setup – JDK , Eclipse, PostgreSQL/Oracle, Tomcat, SOAPUI, etc. Especially make sure you have applied Eclipse settings as described in the Eclipse_Settings_Quick_Reference.doc.
- Acquired the sysmgmt Maven project from SVN or local file server or tech-lead/instructor. See the SystemMgmt_Sample document for instructions concerning how to do this. The JMS sample module is part of the sysmgmt-multimodule project.

Building the projects

The dependency order and, therefore, the build order is as follows:

1. sysmgmt-interface
2. sysmgmt-implementation
3. sysmgmt-service-jms-war

Follow the instructions in the SysMgmt_Sample document to build the sysmgmt-interface and sysmgmt-implementation projects. Once they are built, simply right-click on the sysmgmt-service-jms-war and select “Run As” and then “Maven Install”.

There are two application server environments described wherein the JMS sample will run. These are:

- Tomcat with ActiveMQ Running in Separate JVM’s
- Tomcat with ActiveMQ Embedded

The following sections describe the changes needed to run in the target environment.

Tomcat with ActiveMQ Running in Separate JVM’s

This is the “default” environment. The JMS projects are saved in SVN with settings appropriate to run in this environment. Here are the particulars:

Project `sysmgmt-service-jms-war`, `pom.xml`. Note the comments in the file and comment and un-comment accordingly.

Project `sysmgmt-service-jms-war`, `src/main/webapp/classes/qat-jms-context.properties`. Note the comments in the file and comment and un-comment accordingly.

Project `sysmgmt-service-jms-war`, `src/test/resources/qat-jms-context.properties`. Note the comments in the files and comment and un-comment accordingly.

Tomcat with ActiveMQ Embedded

NOTE: This configuration can only be used with permission from your Architect.

Project `sysmgmt-service-jms-war`, `pom.xml`. Note the comments in the file and comment and un-comment accordingly.

Project `sysmgmt-service-jms-war`, `src/main/webapp/classes/qat-jms-context.properties`. Note the comments in the file and comment and un-comment accordingly.

Project `sysmgmt-service-jms-war`, `src/test/resources/qat-jms-context.properties`. Note the comments in the files and comment and un-comment accordingly.

Deploying to Tomcat

If your Tomcat server is not running, start it. If you are running the environment which requires ActiveMQ to run as a separate process (in a separate JVM), start ActiveMQ first. Refer to Appendix A: Running Apache ActiveMQ for ActiveMQ start instructions.

Copy the WAR file you created using the build process to the Tomcat hot deploy directory (`C:\Apache\apache-tomcat-x.x.x\webapps`).

Wait 30 seconds. During that time you should see a directory created in the `webapps` directory created with the same name as your WAR file name without the `.war` extension. Then proceed to testing.

Testing via Eclipse

You can run the same tests from within Eclipse.

1. Right-click on a test client in the `sysmgmt-service-jms-war/src/test/java/com.qat.samples.sysmgmt.service`.
2. Select Run As
3. Java Application

Running the JUnit Tests

The JMSListenersTest class is a JUnit test class which uses MockRunner to enable testing of the JMS listeners in the jms-implementation project without needing JBoss.

1. Right-click on JMSListenersTest.java
2. Select Run As
3. Select JUnit Test.

Appendix A. Running Apache ActiveMQ

To enable support for JMS running under the standard version of Tomcat, Apache ActiveMQ is required. The instructions below should be used when running ActiveMQ and Tomcat as separate processes (in separate JVM's).

From a console window, change to the installation directory and run `ActiveMQ`:

```
cd C:\apache\apache-activemq-x.x.x
```

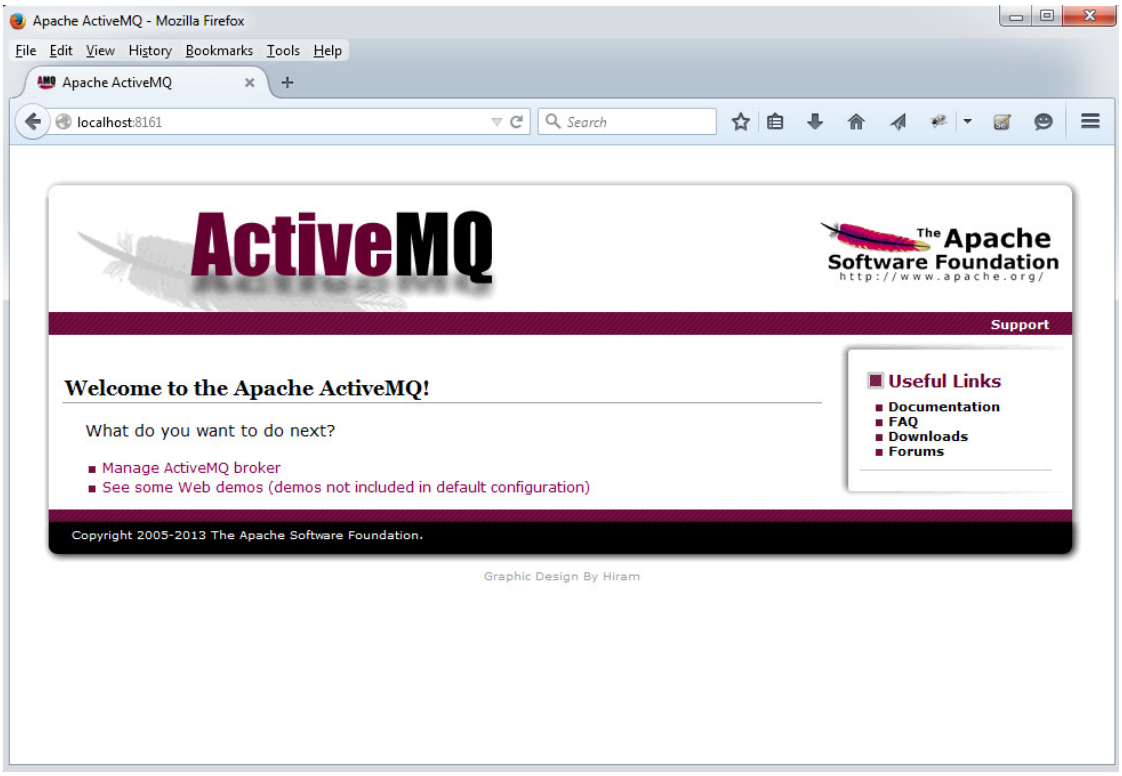
where `C:\apache\apache-activemq-x.x.x` is the directory in which `ActiveMQ` was installed.

Then type:

```
bin\activemq start
```

NOTE: Working directories get created relative to the current directory. To create working directories in the proper place, ActiveMQ must be launched from its home/installation directory.

When ActiveMQ is up, the administration console should be available. By default, the console listens on port 8161. So, if you use URL http://localhost:8161 in your browser you should be prompted for a username and password. The default username and password is admin/admin. You can configure this in the conf/jetty-real.properties file. After logging on you should see a page similar to the following:



From this page you can select "Manage ActiveMQ broker" to view queues, topics, connections, etc.