FuGE Project : FuGE EJB 3.0 STK Development Environment

This page last changed on Jul 22, 2008 by Leandro.Hermida@fmi.ch.

First we need to install and set up all of the necessary components and environment before we can start doing FuGE development. I will assume you are using Unix/Linux with Bash as your shell and that you are doing everything from within your **$HOME** directory.

Java JDK 5.0

```
[user@server ~]$ mkdir ~/java
[user@server ~]$ cd ~/java
```

Go to http://java.sun.com/javase/downloads/index_jdk5.jsp. Scroll down and click to download **JDK 5.0 Update xx** (presently xx = 16). Select the correct file for your hardware/OS and then save the file (e.g. **jdk-1_5_0_16-linux-i586.bin**) into the **~/java** folder. Back in the terminal, execute the installer:

```
[user@server java]$ chmod 744 jdk-1_5_0_16-linux-i586.bin
[user@server java]$ ./jdk-1_5_0_16-linux-i586.bin
```

The installer will unpack and create the directory **jdk1.5.0_16**.

```
[user@server java]$ ln -s jdk1.5.0_16 current
```

Edit your **~/.bash_profile** and add the following:

```
JAVA_HOME=$HOME/java/current
export JAVA_HOME
```

Also append **$JAVA_HOME/bin** to the front of your **PATH**, so you should see something like:

```
PATH=$JAVA_HOME/bin:$PATH:$HOME/bin
export PATH
```

If the computer has has a proxy to the outside world also add the following to your **~/.bash_profile**, for example:

```
JAVA_OPTS="-Dhttp.proxyHost=proxyserver.domain.com -Dhttp.proxyPort=8080 -
Dhttp.nonProxyHosts=*.domain.com|localhost"
export JAVA_OPTS
```

JBoss Application Server

```
[user@server ~]$ mkdir ~/jboss
[user@server ~]$ cd ~/jboss
```

Go to http://labs.jboss.com/jbossas/downloads/. Scroll down and download the latest version (presently **4.2.2.GA**) into the **~/jboss** folder. Next, unzip the files:

```
[user@server jboss]$ unzip jboss-4.2.2.GA.zip
```

For both installation methods do the rest of the steps. Symlink the directory:

```
[user@server jboss]$ ln -s jboss-4.2.2.GA current
```

Edit your **~/.bash_profile** and add the following:

```
JBOSS_HOME=$HOME/jboss/current
export JBOSS_HOME
```

We need to adjust some of the JBoss Java server parameters. Edit the file **$JBOSS_HOME/bin/run.conf** and adjust the **JAVA_OPTS** line to look like:

```
JAVA_OPTS="-Xms256m -Xmx512m -XX:PermSize=128m -XX:MaxPermSize=256m -
Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"
```

Or, if you really have a lot of memory:

```
JAVA_OPTS="-Xms512m -Xmx1024m -XX:PermSize=256m -XX:MaxPermSize=512m -
Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"
```

We want to enable more detailed logging of the persistence provider (i.e. Hibernate). Edit the file **$JBOSS_HOME/server/default/conf/jboss-log4j.xml** and add the following category at the end of the file **before** the root category:

```
<category name="org.hibernate.type">
<priority value="TRACE"/>
<appender-ref ref="CONSOLE"/>
</category>
```

To run the application server:

```
[user@server ~]$ cd $JBOSS_HOME
[user@server current]$ ./bin/run.sh -b 0.0.0.0
```

In order to stop the application server simply hit **Ctrl-C** from within the terminal window where it is running or, from another terminal window:

```
[user@server ~]$ cd $JBOSS_HOME
[user@server current]$ ./bin/shutdown.sh
```

Maven2

Installation of Maven2:

```
[user@server ~]$ mkdir ~/maven
[user@server ~]$ cd ~/maven
```

Download Maven version 2.0.8:

```
[user@server maven]$ wget http://archive.apache.org/dist/maven/binaries/apache-maven-2.0.8-bin.zip
[user@server maven]$ unzip apache-maven-2.0.8-bin.zip
[user@server maven]$ ln -s apache-maven-2.0.8 current
```

Edit your **~/.bash_profile** and add:

```
M2_HOME=$HOME/maven/current
export M2_HOME

M2_REPO=$HOME/.m2/repository
export M2_REPO

MAVEN_OPTS="-Xmx512m -Xms256m -XX:PermSize=128m -XX:MaxPermSize=256m"
export MAVEN_OPTS
```

Also append **$M2_HOME/bin** to the front of your path behind java:

```
PATH=$JAVA_HOME/bin:$M2_HOME/bin:$PATH:$HOME/bin
export PATH
```

If the computer has a proxy to the outside world create the file **$HOME/.m2/settings.xml** with the following, for example:

```
<settings>
<proxies>
<proxy>
<active>true</active>
<protocol>http</protocol>
<host>proxyserver.domain.com</host>
<port>8080</port>
<nonProxyHosts>*.domain.com|localhost</nonProxyHosts>
</proxy>
</proxies>
</settings>
```

Back in the terminal we need to generate our Maven2 repository. This is done by creating a "dummy" project:

```
[user@server maven]$ cd ~/
[user@server maven]$ mvn archetype:create -DgroupId=testproj -DartifactId=testproj
```

This will force Maven to build the repository - you should get **BUILD SUCCESSFUL** at the end. Finally:

```
[user@server maven]$ rm -rf ~/testproj
```

Database Connectivity JARs

For MySQL, download and install MySQL Connector-J version 5.1.6:

```
[user@server ~]$ mkdir ~/mysql
[user@server ~]$ cd ~/mysql
[user@server mysql]$ wget ftp://mirror.switch.ch/mirror/mysql/Downloads/Connector-J/mysql-
connector-java-5.1.6.zip
[user@server mysql]$ unzip mysql-connector-java-5.1.6.zip
[user@server mysql]$ cp -a mysql-connector-java-5.1.6/mysql-connector-java-5.1.6-bin.jar
$JBOSS_HOME/server/default/lib/
[user@server mysql]$ cd $JBOSS_HOME/server/default/lib
[user@server lib]$ ln -s mysql-connector-java-5.1.6-bin.jar mysql-connector-java.jar
```

Manually install the JAR into your Maven repository:

```
[user@server lib]$ cd ~/
[user@server ~]$ mvn install:install-file -DgroupId=mysql -DartifactId=mysql-connector-java -
Dversion=5.1.6 -Dpackaging=jar -Dfile=$HOME/mysql/mysql-connector-java-5.1.6/mysql-connector-
java-5.1.6-bin.jar
```

MagicDraw

Download the MagicDraw 15.0 SP1 installer and then run it:

```
[user@server ~]$ wget ftp://ftp.magicdraw.com/pub/MagicDraw/client/
standard_professional_enterprise/15.0_sp1/MD_UML_150_sp1_unix.sh
[user@server ~]$ chmod 744 MD_UML_150_sp1_unix.sh
[user@server ~]$ ./MD_UML_150_sp1_unix.sh
```

A graphical installer will start. MagicDraw will automatically find the path to the Java binaries installed above which you should select. Install the software into the default location suggested by the installer, e.g. **/home/user/MagicDraw_UML** and unselect a shared launcher.

Next, edit your **~/.bash_profile** and append **$HOME/MagicDraw_UML/bin** to the front of your path behind java:

```
PATH=$JAVA_HOME/bin:$HOME/MagicDraw_UML/bin:$M2_HOME/bin:$PATH:$HOME/bin
export PATH
```

Resource your bash profile if necessary. Before launching the application, copy your MagicDraw license into the **$HOME/MagicDraw_UML** folder. Next launch the application

```
[user@server ~]$ mduml &
```

When prompted for the license, choose **Select Unlock Key File** and browse to the license file you copied above. Next, set up the MagicDraw general environment:

- Go to **Options --> Environment**
  - General --> Create Backup File = False
  - General --> Show Memory Montior = True
  - General --> Show Status Bar = True

Development Workspace

Create the **workspace** directory in your **$HOME** directory:

```
[user@server ~]$ mkdir ~/workspace
```

Eclipse

Download the latest **Eclipse Europa Package Eclipse IDE for Java EE Developers** gzipped tar file for you architecture (for 64-bit should be eclipse-jee-europa-winter-linux-gtk-x86_64.tar.gz) into your **$HOME** directory and unpack it http://www.eclipse.org/downloads/.

```
[user@server ~]$ tar -xvzf eclipse-jee-europa-winter-linux-gtk.tar.gz
```

It will create a directory called **eclipse**. Edit your **~/.bash_profile** and append **$HOME/eclipse** to you **PATH**:

```
PATH=$JAVA_HOME/bin:$HOME/MagicDraw_UML/bin:$M2_HOME/bin:$HOME/eclipse:$PATH:$HOME/bin
export PATH
```

You probably want to increase the memory that Eclipse has to work with. Edit the file **$HOME/eclipse/eclipse.ini** and add/change the Java memory settings to look like the following:

```
-Xms256m
-Xmx512m
-XX:MaxPermSize=256m
```

After sourcing your bash profile, start eclipse for the first time:

```
[user@server ~]$ eclipse &
```

It will first ask you to select a workspace, type the full path to the **workspace** folder (e.g. **/home/user/workspace**) and check the box to always use this as the default. The Eclipse Welcome page will appear, click on the **Workbench** button. Finally, configure Eclipse:

- Go to the Project menu and ***uncheck*** **Build Automatically**
- Go to **Window -> Preferences -> Java -> Build Path -> Classpath Variables** and create a new variable called **M2_REPO** with the full path to your Maven2 repository from above (e.g. /home/user/.m2/repository)
- Go to **Window -> Preferences -> Team -> File Content** and add two new extensions, *.vsl and *.vm as ASCII files.
- Go to **Window -> Preferences -> General -> Appearance -> Colors and Fonts** then under **Basic -> Text Font**, click **Change...** and reduce the font size.
- Go to **Window -> Preferences -> General -> Appearance -> Editors -> Text Editors** and check **Insert Spaces for tabs**

JBoss Tools (optional but recommended)

As root, if it is not installed already you need to first install some compatibility libstdc++ libraries:

```
[root@server ~]# yum install compat-libstdc++-33
```

You also have to define the LD_LIBRARY_PATH environment variable to look in the right directories for shared libraries. Edit your **~/.bash_profile** and add the following, for 32-bit JVM:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib:/usr/lib:/usr/local/lib:$JAVA_HOME/jre/lib/i386:$JAVA_HOME/
jre/lib/i386/client:$JAVA_HOME/jre/lib/i386/server
export LD_LIBRARY_PATH
```

for 64-bit JVM:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lib64:/lib:/usr/lib64:/usr/lib:/usr/local/lib64:/usr/local/lib:
$JAVA_HOME/jre/lib/amd64:$JAVA_HOME/jre/lib/amd64/server
export LD_LIBRARY_PATH
```

Download the latest JBoss Tools for your architecture (for 64-bit should be JBossTools-2.1.2.GA-ALL-linux-gtk-x86_64.zip) from http://www.jboss.org/tools/download/ into your **$HOME** directory. Unzip the package:

```
[user@server ~]$ unzip JBossTools-2.1.2.GA-ALL-linux-gtk.zip
```

It will automatically unzip all the tools into your Eclipse plugins directory. Start Eclipse cleanly so that it re-reads all of the available plugins:

```
[user@server ~]$ eclipse -clean
```

FuGE EJB 3.0 STK SVN Checkout

From within the **~/workspace** directory, checkout the latest FuGE EJB 3.0 STK:

```
[user@server ~]$ cd ~/workspace
[user@server workspace]$ svn co https://fuge.svn.sourceforge.net/svnroot/fuge/ejb3-stk/trunk fuge-ejb3-stk
```

AndroMDA 3.4-SNAPSHOT and AndroMDA Plugins Project for FuGE EJB 3.0 STK

Copy the two AndroMDA project zip files from the STK to our workspace and then unzip both of them:

```
[user@server workspace]$ cp -a fuge-ejb3-stk/andromda/* .
[user@server workspace]$ unzip andromda-all-3.4-SNAPSHOT-fuge-ejb3-stk.zip
[user@server workspace]$ unzip andromda-plugins-fuge-ejb3-stk.zip
```

This will generate two Maven project directories, **andromda-all** and **andromda-plugins**. First let's build **andromda-all**

```
[user@server ~]$ cd ~/workspace/andromda-all
[user@server andromda-all]$ mvn -N antrun:run
```

You should see **BUILD SUCCESSFUL** at the end.

```
[user@server andromda-all]$ mvn install
```

This will take a while (5 to 20 minutes), you should also see **BUILD SUCCESSFUL** at the end. Sometimes you will get the following strange failure:

```
[INFO] ------------------------------------------------------------------------
[ERROR] BUILD ERROR
[INFO] ------------------------------------------------------------------------
[INFO] Error executing ant tasks

Embedded error: Warning: Could not find file /home/user/workspace/andromda-all/maven/2/andromdapp/null/andromdapp-maven-plugin-3.3-SNAPSHOT.jar to copy.
```

This requires an **mvn clean** followed by another **mvn install** and then it will successfully compile.

Next let's build the **andromda-plugins** project:

```
[user@server ~]$ cd ~/workspace/andromda-plugins
[user@server andromda-plugins]$ mvn install
```

The build will most likely fail due to the following bug in the Database cartridge:

```
[INFO] ------------------------------------------------------------------------
[ERROR] BUILD ERROR
[INFO] ------------------------------------------------------------------------
[INFO] Error running AndroMDA

Embedded error: org.netbeans.lib.jmi.util.DebugException: JAR entry andromda-profile-persistence-3.2-SNAPSHOT.xml.zip not found in /home/user/workspace/andromda-plugins/cartridges/andromda-database/src/test/uml/DatabaseCartridgeTestModel.xml.zip
```

But this is completely **OK** as we only need to do the top-level build to get some important dependencies that need to be set up and installed which happen when you first attempt to do an mvn install.

Now let's move to building the EJB3 cartridge:

```
[user@server ~]$ cd ~/workspace/andromda-plugins/cartridges/andromda-ejb3
[user@server andromda-ejb3]$ mvn install
```

Should say **BUILD SUCCESSFUL** at the end.