**Requirements**

* Apache Maven 3.0+
* Java 7
* Eclipse Kepler or above
  + Version for Java EE Developers
* UnZip (for linux only)

Other setups were not tested but may work.

**Installation Linux/Windows:**

1. Linux: Run “epp” script from the root repository folder and follow the prompts for installation procedure.  
   Windows:
   * Edit your credentials for MySQL in:  
     /EapCommons/src/main/resources/META-INF/persistence.xml

(if not existent use the provided ‘persistence\_template.xml’ to create it)

* + Edit your credentials for Tomcat in:  
    /EapWebinterface/pom.xml

<project>

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<configuration>

<!-- edit Tomcat path and credentials here -->

<server>TomcatServer</server>

<url>http://localhost:8080/manager/text</url>

<path>/EapWebInterface</path>

<username>tomcat</username>

<password>tomcat</password>

</configuration>

</plugin>

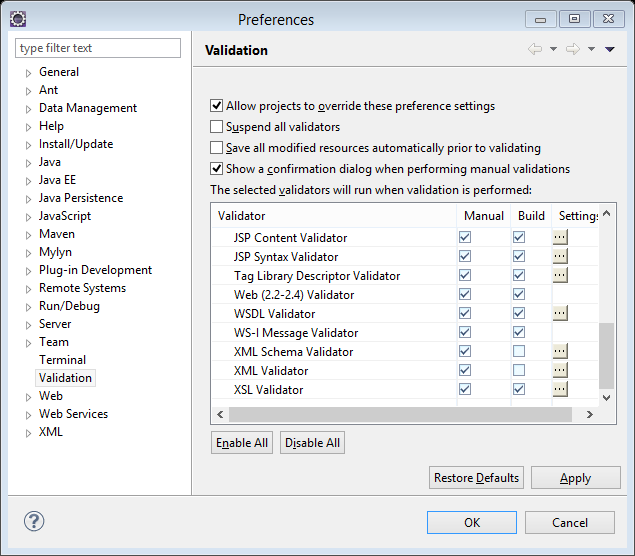
</plugins>

</build>

</project>

* + Run “epp.bat” script from the root repository folder.

1. Import desired projects only into your eclipse workspace as existing maven projects.
   * Confirm permanent disabling of missing axis plugin in EapWebInterface project (this is an eclipse GUI hint next to the marked error in the specified file)
   * Turn off automatic build validation of xsd and xml files



**Run the EPP locally**

1. Make sure that all jbpt projects, the Esper project and all Eap projects are imported in eclipse.
2. Make sure that MySQL is installed and running. The databases specified in the persistence.xml must be available.
3. In eclipse, run EapWebInterface/src/test/java/hpi/eap/application/StartEapWebInterface.java as Java application.
   1. The application starts a Jetty server at the port specified in StartEapWebInterface.java  
        
      SocketConnector connector = new SocketConnector();  
      connector.setPort(8081);  
        
      and deploys the Event Processing Platform on <http://localhost:8081/>
4. Close the Java application to stop the Jetty server.

**Deployment**

1. Commit (maybe create a new one) to the branch for your feature to the git repository
2. visit the Hudson CI Server at <http://172.16.64.114:8888/> (just register if you have no login yet)
3. create a new hudson job
   * please do only enter one-word job names like "get2013development" or use underscores like in "get\_2013\_development"
   * select to 'Copy an existing job' and enter the job name that fits your repository (like 'integration\_dev' if you work on the 'integration' git repository)
4. Configure your job:
   * enter your newly created branch under 'Source-Code-Management' > 'Branches to build'
   * adjust the three lines in the 'Execute shell script on remote host using ssh' section to your job name (e.g. 'integration\_newfeature'):
   * for a job name like 'get2013\_development' it would be:

echo "CREATE database IF NOT EXISTS eap\_development\_get2013\_development" | mysql -ueap –peapfisch

echo "CREATE database IF NOT EXISTS eap\_testing\_get2013\_development" | mysql -ueap –peapfisch

mkdir -p -m 777 /var/lib/GETsemantic/get2013\_development

* + adjust the first shell script in the build steps according to the created folder above:

location=/var/lib/GETsemantic/get2013\_development/data

**Deployment to a new server**

1. (if not already present) create a user on the server you want to deploy to (e.g. via ‘sudo adduser hudson’); this user will perform non-sudo actions like creating new folders on this server
2. (if not already present) create a folder that the user from 1. can access and create folders and files in (e.g. via ‘sudo mkdir /var/lib/GETsemantic’ and ‘sudo chmod a=xwr /var/lib/GETsemantic’)
3. (if not already present) copy cache delete script for tomcat into /var/lib/tomcat7/ and name the file tomcatCacheDelete. The red line in the script with corresponds to the first command line argument and should be the packed war file name and will be configured in Hudson later!

#!/bin/bash

echo "manually deleting tomcat cache"

service tomcat7 stop

cd /var/lib/tomcat7/work/Catalina/localhost

directory=$(ls)

for i in $directory

do

:

if [[ $i == $1\* ]] ;

then

folder="$i"

echo "Expunge: $folder"

rm -r $folder

mkdir $folder

chown -R tomcat7 $folder

fi

done

service tomcat7 start

1. Make sure all development preconditions are met (e.g. MySQL, Java, Maven, etc. are in place and configured)
2. register the user in the Hudson interface at <http://172.16.64.114:8888/configure> by adding a new site at ‘SSH remote hosts’, with the server to deploy to and the user & password from 1.
3. Copy an existing job and configure it (see also steps in section ‘Deployment’ above):
4. Additionally to section ‘Deployment’:
   1. Adjust the user and server (e.g. [debug@172.14.64.105](mailto:debug@172.14.64.105), where ‘debug’ is the user) at ‘SSH Site’ under 'Execute shell script on remote host using ssh' to your server and user from 1.
   2. Adjust the shell script text box with the comment ‘##### configure pom.xml from EapWebInterface ####’ (usually the second script box and the last three lines there) to your new server and tomcat user&password (e.g. replace (1) 172.16.64.105 by your new server, (2) admin by your tomcat user, and (3) eapfisch by your tomcat password)
   3. Adjust the shell script box with the comment ‘###### configure persistence.xml ######’ (usually the third script box) if need-be (the mysql user and password are stored in variables in the beginning of this script, simply change these if necessary)

**Script Documentation**

Pre-Build Script creates databases for the branch, job name has to be inserted into script:

echo "CREATE database IF NOT EXISTS eap\_development\_<job\_name>" | mysql -ueap -peapfisch

echo "CREATE database IF NOT EXISTS eap\_testing\_<job\_name>" | mysql -ueap -peapfisch

mkdir -p -m 777 /var/lib/GETsemantic/<job\_name>

Post-Build Script for Tomcat Cache Expunge (always make sure, that this script is available in that position on the tomcat!) needs to have the prefix of packed war file:

sudo /var/lib/tomcat7/tomcatCacheDelete <war-name>

Build Script for semantic part:

###### configure triplestore location ######

sed -i '/location/c\location=/var/lib/GETsemantic/integration\_dev/data' EapSemantic/src/main/resources/Triplestore.properties

Configuration Script Tomcat is responsible to configure the pom file of the EapWebInterface to enable communication with the tomcat server (no changes are necessary since all naming is derived from the Hudson job name):

sep="s"

###### configure pom.xml from EapWebInterface #####

tomcatLine=$(sed -n "/tomcat-maven-plugin/ =" EapWebInterface/pom.xml)

tomcatLastLine=$(($tomcatLine + 10))

sed -i "s%<finalName>\(.\*\)</finalName>%<finalName>\1-$JOB\_NAME</finalName>%" EapWebInterface/pom.xml

sed -i "s%<path>\(.\*\)</path>%<path>\1-$JOB\_NAME</path>%" EapWebInterface/pom.xml

sed -i "$tomcatLine,$tomcatLastLine$sep%<url>.\*</url>%<url>http://172.16.64.112:8080/manager/text</url>%" EapWebInterface/pom.xml

sed -i "$tomcatLine,$tomcatLastLine$sep%<username>.\*</username>%<username>admin</username>%" EapWebInterface/pom.xml

sed -i "$tomcatLine,$tomcatLastLine$sep%<password>.\*</password>%<password>eapfisch</password>%" EapWebInterface/pom.xml

Configuration Script Database is responsible to create the persistence.xml and provide correct access to the database (no changes are necessary since all naming is derived from the Hudson job name):

cp EapCommons/src/main/resources/META-INF/persistence\_template.xml EapCommons/src/main/resources/META-INF/persistence.xml

sep="s"

mysqlUser="eap"

mysqlPassword="eapfisch"

eapDevelopmentLine=$(sed -n "/persistence-unit.\*eap\_development/ =" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

eapTestLine=$(sed -n "/persistence-unit.\*eap\_test/ =" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

eapLastLine=$(wc -l < EapCommons/src/main/resources/META-INF/persistence\_template.xml)

mySQLURLLine=$(sed -n "$eapDevelopmentLine,$eapTestLine {/javax\.persistence\.jdbc\.url/ =}" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

mySQLUserLine=$(sed -n "$eapDevelopmentLine,$eapTestLine {/javax\.persistence\.jdbc\.user/ =}" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

mySQLPasswordLine=$(sed -n "$eapDevelopmentLine,$eapTestLine {/javax\.persistence\.jdbc\.password/ =}" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

sed -i "$mySQLURLLine,$mySQLURLLine$sep%.\*%<property name=\"javax.persistence.jdbc.url\" value=\"jdbc:mysql://localhost:3306/eap\_development\_$JOB\_NAME\" />%" EapCommons/src/main/resources/META-INF/persistence.xml

sed -i "$mySQLUserLine,$mySQLUserLine$sep%.\*%<property name=\"javax.persistence.jdbc.user\" value=\"$mysqlUser\" />%" EapCommons/src/main/resources/META-INF/persistence.xml

sed -i "$mySQLPasswordLine,$mySQLPasswordLine$sep%.\*%<property name=\"javax.persistence.jdbc.password\" value=\"$mysqlPassword\" />%" EapCommons/src/main/resources/META-INF/persistence.xml

mySQLURLLine=$(sed -n "$eapTestLine,$eapLastLine {/javax\.persistence\.jdbc\.url/ =}" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

mySQLUserLine=$(sed -n "$eapTestLine,$eapLastLine {/javax\.persistence\.jdbc\.user/ =}" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

mySQLPasswordLine=$(sed -n "$eapTestLine,$eapLastLine {/javax\.persistence\.jdbc\.password/ =}" EapCommons/src/main/resources/META-INF/persistence\_template.xml)

sed -i "$mySQLURLLine,$mySQLURLLine$sep%.\*%<property name=\"javax.persistence.jdbc.url\" value=\"jdbc:mysql://localhost:3306/eap\_testing\_$JOB\_NAME\" />%" EapCommons/src/main/resources/META-INF/persistence.xml

sed -i "$mySQLUserLine,$mySQLUserLine$sep%.\*%<property name=\"javax.persistence.jdbc.user\" value=\"$mysqlUser\" />%" EapCommons/src/main/resources/META-INF/persistence.xml

sed -i "$mySQLPasswordLine,$mySQLPasswordLine$sep%.\*%<property name=\"javax.persistence.jdbc.password\" value=\"$mysqlPassword\" />%" EapCommons/src/main/resources/META-INF/persistence.xml

Compilation script runs maven (no changes are necessary since all naming is derived from the Hudson job name):

mvn -f Esper/pom.xml clean install -DskipTests

mvn -f EapCommons/pom.xml clean install -DskipTests

mvn -f EapImport/pom.xml clean install -DskipTests

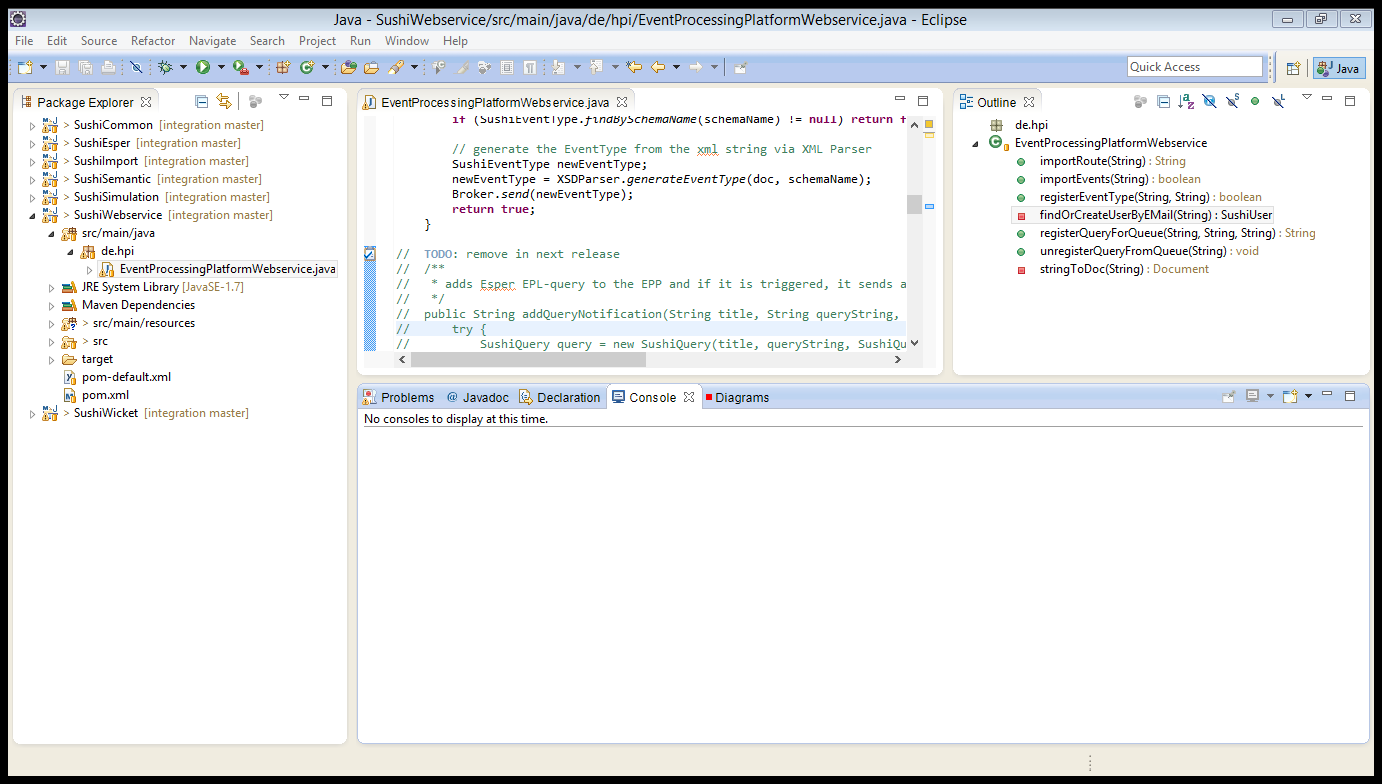
mvn -f EapSemantic/pom.xml clean install -DskipTests

mvn -f EapEventProcessing/pom.xml clean install -DskipTests

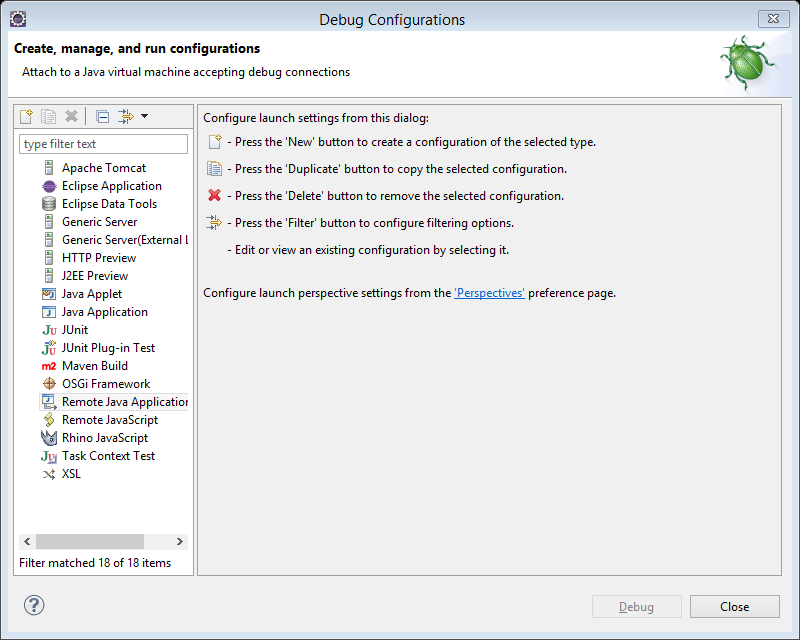
mvn -f EapWebInterface/pom.xml clean install tomcat:redeploy -DskipTests

**Debugging**

1. Open an ssh tunnel to the remote host with user “debug” and password “whiskey”. Tunnel from an arbitrary port on your side <tunnel-port> to the remote hosts 8000 port.
2. Open “Debug Configurations” dialog window



1. Select “Remote Application” and add new launch configuration.



1. Select your project and enter the host and port of the debug server 172.16.64.105:8000.
2. Hit “Debug” to start debugging the current eclipse project.
3. Browse to http://localhost:<tunnel-port>/EapWebInterface-<your git branch name> to work on your current code branch.

Be aware that only one developer at a time can be connected to the remote debugger

**Issues**

* the build fails due to unknown host exception
  + the host needs to be added to the known\_hosts for hudson
  + connect yourself to the hudson via ssh and the account 'hudson' with the password 'hudson'
  + perform 'ssh <repository address>' (without the ')
  + confirm that you want to add the host permanently
  + you can disconnect from the ssh session with hudson and build again
* the build fails because it cannot connect to the repository due to authentication issues
  + make sure that the hudson's SSH key is registered at the repository
  + you can find the key by opening a ssh session to the hudson (username and password ‘hudson') on the hudson server, use 'less ~/.ssh/id\_rsa.pub' (without the ') and you can see the public key

*Last updated on July 30, 2014*