TEMENOS™

TAFJ MAVEN PLUGIN

R14/R15

1/14/2014

Temenos

Amendment History:

Revision	Date Amended	Name	Description
1	13 February 2014	JN. Charpin	Initial version
2	15th April 2014	H. Aubert	R14GA Review
3	5th March 5, 2015	JN. Charpin	R15 AMR review

Copyright

Errata and Comments

If you have any comments regarding this manual or wish to report any errors in the documentation, please document them and send them to the address below:
Technology Department

Temenos Headquarters SA
2 Rue de l'Ecole-de-Chimie,
CH - 1205 Geneva,
Switzerland

Tel SB: +41 (0) 22 708 1150
Fax: +41 (0) 22 708 1160

Please include your name, company, address, and telephone and fax numbers, and email address if applicable. TAFJdev@temenos.com

Table of Contents

Introduction

This document presents how TAFJ could be integrated within a maven project and what are the advantages to do so.

As TAFJ is PIC/Basic java runtime and compiler and Maven a tool that can be used to build and manage any Java project, we will see that using TAFJ through the maven lifecycle could be easy and useful. Another advantage is that you could get updated with latest TAFJ product build directly from the maven repository.

To be used from a Maven project TAFJ provides a Maven plugin called tafj-maven-plugin.

This plugin can be used to:

- Install the TAFJ version of your choice
- Configure an existing TAFJ environment
- Compile and package your basic code
- Create the TAFJ database from the J4 files

This document is not a maven guide. It presents some example on how you could use the tafj-maven-plugin but please refer to the maven documentation if you need more information about maven.

http://maven.apache.org/index.html

Prerequisite

You need a Maven installation; version should be at least 3.0.4. Then add M2_HOME/bin to your path.

http://maven.apache.org/download.cgi

You need a JDK; version should be a least 1.6 and JAVA_HOME should point to your JDK installation.

TAFJ maven plugin and TAFJ artifacts are not published to the central maven repository, thus you need to be connected to the Temenos maven repository to be able to use it.

This repository access is granted through the maven file settings.xml. Please send a request to TAFJ Development Team TAFJdev@temenos.com to get a copy of this file.

You could either copy this file under your maven home directory or run your maven command with parameter pointing to this file : mvn -s $TAFJ_HOME/maven/settings.xml.

https://maven.apache.org/settings.html

TAFJ Maven demo

The easiest way to validate that your installation is correct and that you are able to reach Temenos maven repository is to run the TAFJ maven demo.

This demo will download from the Temenos Maven repository to your local machine a maven project configured to demonstrate the TAFJ maven plugin capabilities.

First you will need to create a DEMO_HOME folder and copy under $DEMO_HOME/pom.xml the following content.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>


      <groupId>com.temenos.tafj</groupId>
      <name>TAFJMavenDemoExtractor</name>
      <artifactId>TAFJMavenDemoExtractor</artifactId>
      <version>15.0.1</version>

      <properties>
            <demoPath ${basedir}/../demo</demoPath>
      </properties>

      <dependencies>
            <dependency>
                  <groupId>com.temenos.tafj</groupId>
                  <artifactId>TAFJMavenDemo</artifactId>
                  <version>15.0.1</version>
                  <type>tar</type>
            </dependency>
      </dependencies>

</project>
```

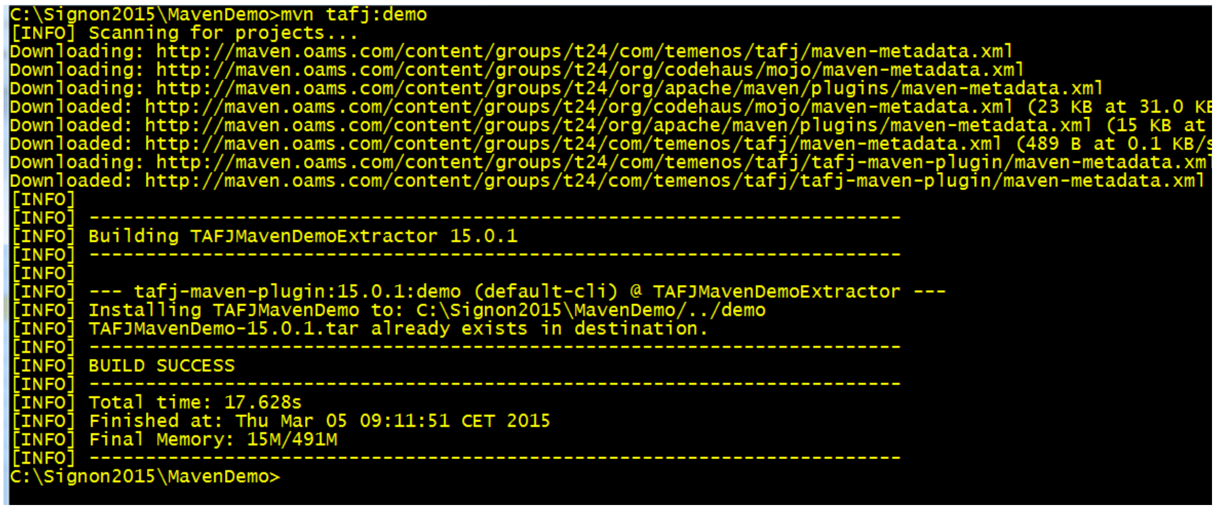
You could see that this file defines a property **demoPath** this is where the demo is going to be installed and where you will have to launch the maven command to test it.

You could change this path to the value you want.

To install the demo launch the command:

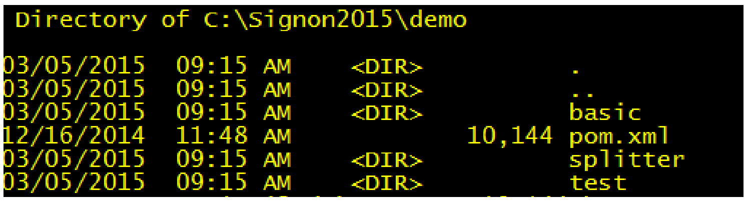
```
mvn tafj:demo
```

Your maven local repository will be updated with the demo artifacts and they will be expanded to your demoPath which by default is **$DEMO_HOME/../demo**.



```
C:\Signon2015\MavenDemo>mvn tafj:demo
[INFO] Scanning for projects...
Downloading: http://maven.oams.com/content/groups/t24/com/temenos/tafj/maven-metadata.xml
Downloading: http://maven.oams.com/content/groups/t24/org/codehaus/mojo/maven-metadata.xml
Downloading: http://maven.oams.com/content/groups/t24/org/apache/maven/plugins/maven-metadata.xml
Downloaded: http://maven.oams.com/content/groups/t24/org/codehaus/mojo/maven-metadata.xml (23 KB at 31.0 K
Downloaded: http://maven.oams.com/content/groups/t24/org/apache/maven/plugins/maven-metadata.xml (15 KB at
Downloaded: http://maven.oams.com/content/groups/t24/com/temenos/tafj/maven-metadata.xml (489 B at 0.1 KB/
Downloading: http://maven.oams.com/content/groups/t24/com/temenos/tafj/tafj-maven-plugin/maven-metadata.xm
Downloaded: http://maven.oams.com/content/groups/t24/com/temenos/tafj/tafj-maven-plugin/maven-metadata.xml
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building TAFJMavenDemoExtractor 15.0.1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- tafj-maven-plugin:15.0.1:demo (default-cli) @ TAFJMavenDemoExtractor ---
[INFO] Installing TAFJMavenDemo to: C:\Signon2015\MavenDemo/../demo
[INFO] TAFJMavenDemo-15.0.1.tar already exists in destination.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 17.628s
[INFO] Finished at: Thu Mar 05 09:11:51 CET 2015
[INFO] Final Memory: 15M/491M
[INFO] ------------------------------------------------------------------------
C:\Signon2015\MavenDemo>
```

By browsing your demo path you should find the following content.



```
 Directory of C:\Signon2015\demo

03/05/2015  09:15 AM    <DIR>          .
03/05/2015  09:15 AM    <DIR>          ..
03/05/2015  09:15 AM    <DIR>          basic
12/16/2014  11:48 AM            10,144 pom.xml
03/05/2015  09:15 AM    <DIR>          splitter
03/05/2015  09:15 AM    <DIR>          test
```

The main demo file is the **pom.xml**. It's a maven project demonstrating tafj-maven-plugin across the maven lifecycle.

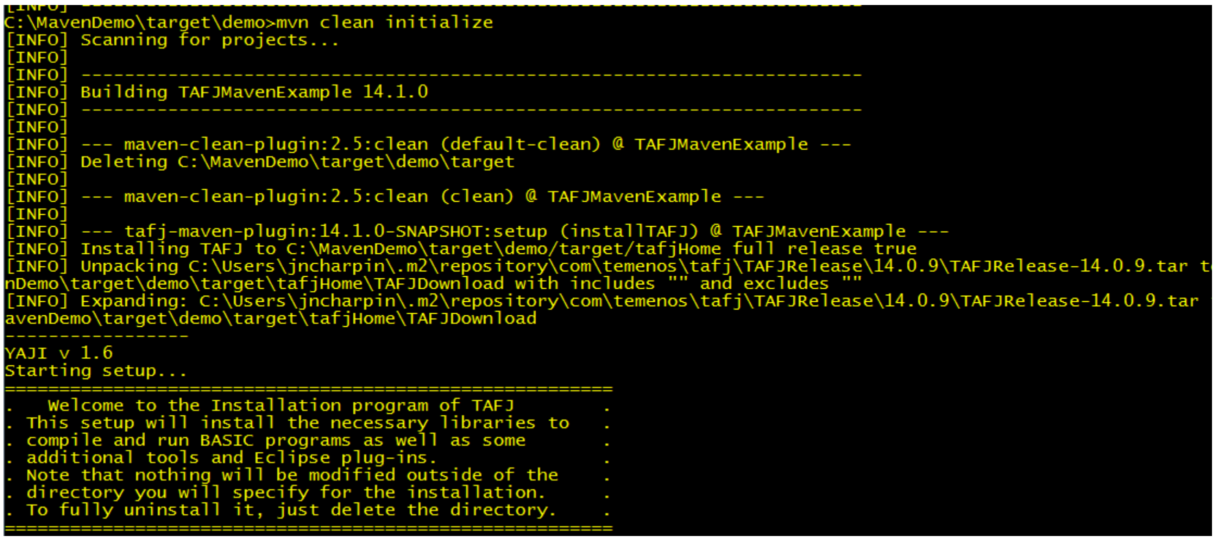
We will explain the content of this file when running the demo. You could also refer directly to the comments in this file.

To execute the demo, go to your demo path and run command

mvn clean initialize

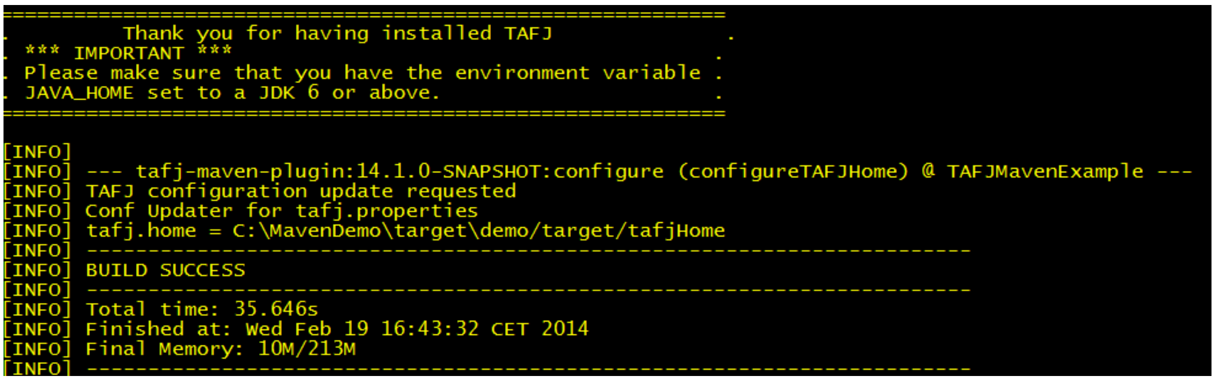
TAFJ Maven setup

You should see the following output.



```
[INFO] ------------------------------------------------------------------------
C:\MavenDemo\target\demo>mvn clean initialize
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building TAFJMavenExample 14.1.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ TAFJMavenExample ---
[INFO] Deleting C:\MavenDemo\target\demo\target
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (clean) @ TAFJMavenExample ---
[INFO]
[INFO] --- tafj-maven-plugin:14.1.0-SNAPSHOT:setup (installTAFJ) @ TAFJMavenExample ---
[INFO] Installing TAFJ to C:\MavenDemo\target\demo/target/tafjHome full release true
[INFO] Unpacking C:\Users\jncharpin\.m2\repository\com\temenos\tafj\TAFJRelease\14.0.9\TAFJRelease-14.0.9.tar t
nDemo\target\demo\target\tafjHome\TAFJDownload with includes "" and excludes ""
[INFO] Expanding: C:\Users\jncharpin\.m2\repository\com\temenos\tafj\TAFJRelease\14.0.9\TAFJRelease-14.0.9.tar
avenDemo\target\demo\target\tafjHome\TAFJDownload
-----------------
YAJI v 1.6
Starting setup...
========================================================
.   Welcome to the Installation program of TAFJ        .
. This setup will install the necessary libraries to   .
. compile and run BASIC programs as well as some       .
. additional tools and Eclipse plug-ins.               .
. Note that nothing will be modified outside of the    .
. directory you will specify for the installation.     .
. To fully uninstall it, just delete the directory.    .
========================================================
```

….



```
==========================================================
.         Thank you for having installed TAFJ            .
. *** IMPORTANT ***                                      .
. Please make sure that you have the environment variable .
. JAVA_HOME set to a JDK 6 or above.                     .
==========================================================

[INFO]
[INFO] --- tafj-maven-plugin:14.1.0-SNAPSHOT:configure (configureTAFJHome) @ TAFJMavenExample ---
[INFO] TAFJ configuration update requested
[INFO] Conf Updater for tafj.properties
[INFO] tafj.home = C:\MavenDemo\target\demo/target/tafjHome
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 35.646s
[INFO] Finished at: Wed Feb 19 16:43:32 CET 2014
[INFO] Final Memory: 10M/213M
[INFO] ------------------------------------------------------------------------
```

You can see that a TAFJ release has been downloaded from the Temenos maven repository and installed under **$DEMO_HOME/../demo/tafjHome.**

The full pack is available under **$DEMO_HOME/../demo/tafjHome/TAFJDownload.**

The pom section related to the TAFJ installation is the following:

```
<plugin>
      <groupId>com.temenos.tafj</groupId>
      <artifactId>tafj-maven-plugin</artifactId>
      <executions>

      <!-- setup goal - Install specified tafjVersion under specified
tafjHome -->
            <execution>
                  <id>installTAFJ</id>
                  <goals>
                  <!-- default phase is initialize -->
                        <goal>setup</goal>
                  </goals>
                  <configuration>
                        <tafjHome>${tafjHome}</tafjHome>
      <!-- when set to false an existing install won't be modified -->
      <!-- default plugin value is true -->
                        <override>false</override>
                  </configuration>
            </execution>
```

We can see that we just invoke the goal setup of the tafj-maven-plugin and we specify under the configuration section where we want TAFJ to be installed with the parameter tafjHome.

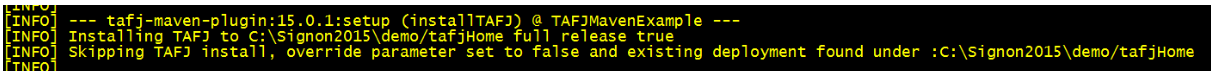
The goal setup is bound by default to the maven phase initialize, you could specify whatever phase you want by adding a tag <phase> in the execution section but you will have to take care of the maven lifecycle if you plan to use other tafj maven plugin features.

tafjHome parameter refers to a maven property ${tafjHome} defined in the properties section of the pom as we will reuse this property in multiple occasion in the pom.

```
<properties>
      <!-- Path where you want to install TAFJ or where TAFJ is installed
-->
      <tafjHome>${basedir}/tafjHome</tafjHome>
       …
```

No TAFJ product version has been specified. Latest TAFJ release available is getting deployed. This is the tafj-maven-parent described in next section that is managing the version.

When executing this goal multiple times the parameter “override” is going to be considered to decide whether or not an existing installation will be overridden.



```
[INFO]
[INFO] --- tafj-maven-plugin:15.0.1:setup (installTAFJ) @ TAFJMavenExample ---
[INFO] Installing TAFJ to C:\Signon2015\demo/tafjHome full release true
[INFO] Skipping TAFJ install, override parameter set to false and existing deployment found under :C:\Signon2015\demo/tafjHome
[INFO]
```

TAFJ Maven parent

We can see at the top of the pom.xml that it defines a parent pom:

```
      <parent>
            <groupId>com.temenos.tafj</groupId>
            <artifactId>tafj-maven-parent</artifactId>
            <version>1.0.0</version>
      </parent>
```

It means this demo project will inherit the configuration of the tafj-maven-parent project.

All projects depending on tafj-maven-plugin should define tafj-maven-parent as parent project.

It will ensure that the correct plugin version and configuration is being used.

As seen in previous section, by default tafj-maven-project defines a dependency on latest stable TAFJ release available.

Installing latest snapshot

This pom parent also defines a specific maven profile to depend on latest TAFJ snapshot available. This profile should be used only for specific development purpose. To install latest TAFJ snapshot run the following command.

```
mvn clean initialize -P latest
```

You can see that a snapshot is getting installed instead of a release.

Installing specific version

You can also install the version of your choice by overriding in the pom.xml the parameter tafjVersion in the properties section and running command “mvn clean initialize”

i.e

```
<properties>
<!-- Path where you want to install TAFJ or where TAFJ is installed -->
            <tafjHome>${basedir}/tafjHome</tafjHome>

<!-- To override tafjVersion defined in the tafj-maven-parent -->
<!-- tafj-maven-parent is bound by default to latest TAFJ release available
-->
<!-- using profile -P latest will bind tafj-maven-parent to latest snapshot
      available -->
            <tafjVersion>15.0.13</tafjVersion>
</properties>
```

TAFJ Maven configure

To have a TAFJ environment ready to use you may have to configure your TAFJ properties file and your logger levels.

This is what the tafj-maven-plugin goal configure is for. This goal is also bound by default to the maven phase initialize thus it has been executed after the tafj installation when you ran the command “mvn clean initialize”.

When multiple goals are bound to same maven phase, they will be executed according to the order they appear in the pom.

The pom section related to the TAFJ HOME configuration is the following:

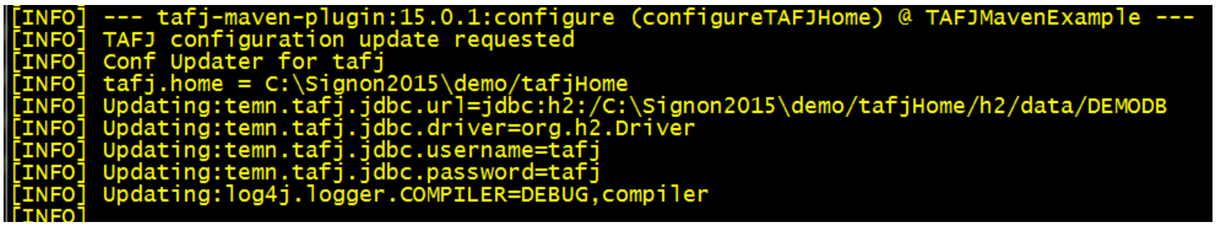
```
<!-- configure goal - Create a properties file under specified tafjHome
with specified name -->
<!-- generated properties file will be based on default with specified
values override -->
<execution>
      <id>configureTAFJHome</id>
      <goals>
      <!-- default phase is initialize -->
            <goal>configure</goal>
      </goals>
      <configuration>
            <tafjHome>${tafjHome}</tafjHome>
            <tafjProperties>tafj</tafjProperties>
            <!-- Change some properties value -->
            <!-- applies to $tafjHome/conf/$tafjProperties -->
            <!-- i.e. configure the DB paramaters -->
            <properties>
                  <temn.tafj.jdbc.url>jdbc:h2:/$
                  {tafjHome}/h2/data/DEMODB</temn.tafj.jdbc.url>
                  <temn.tafj.jdbc.driver>org.h2.Driver</temn.tafj.jdbc.driv
            er>
                  <temn.tafj.jdbc.username>tafj</temn.tafj.jdbc.username>
                  <temn.tafj.jdbc.password>tafj</temn.tafj.jdbc.password>
            </properties>
            <!-- Change some default loggers configuration -->
            <!-- applies to $tafjHome/conf/TAFJTrace.properties -->
            <tafjTraceProperties>
                  <log4j.logger.COMPILER>DEBUG,compiler</log4j.logger.COMPI
            LER>
            </tafjTraceProperties>
      </configuration>
</execution>
```

We specify in this section, with parameters tafjHome and tafjProperties, the properties file we want to create or update within the related tafj home.

If the properties file doesn't exist it will create a new one with the specified name based on .properties template from **<TAFJ_HOME>/conf.**

The <properties> section defines the properties that should be configured.

The <tafjTraceProperties> section allows you to configure TAFJ_HOME/conf/TAFJTrace.properties with appropriate logger level for the appenders of your choice.



```
[INFO] --- tafj-maven-plugin:15.0.1:configure (configureTAFJHome) @ TAFJMavenExample ---
[INFO] TAFJ configuration update requested
[INFO] Conf Updater for tafj
[INFO] tafj.home = C:\Signon2015\demo/tafjHome
[INFO] Updating:temn.tafj.jdbc.url=jdbc:h2:/C:\Signon2015\demo/tafjHome/h2/data/DEMODB
[INFO] Updating:temn.tafj.jdbc.driver=org.h2.Driver
[INFO] Updating:temn.tafj.jdbc.username=tafj
[INFO] Updating:temn.tafj.jdbc.password=tafj
[INFO] Updating:log4j.logger.COMPILER=DEBUG,compiler
[INFO]
```

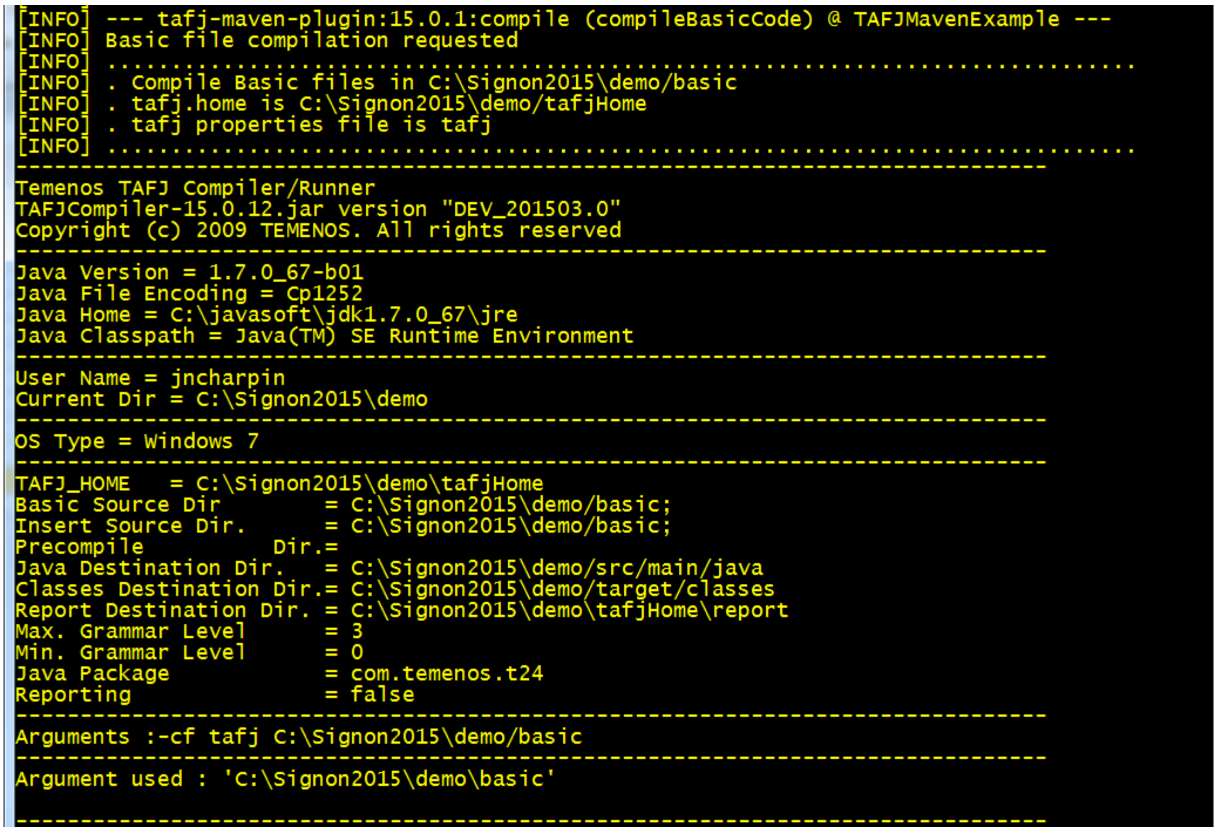
TAFJ Maven compile

You can compile BASIC files with tafj-maven-plugin and the goal compile. By default the goal compile of the plugin is associated to the maven phase generate-sources.

To execute the compilation of the basic files $DEMO_HOME/target/demo/basic, simply run the command:

```
mvn clean generate-sources
```

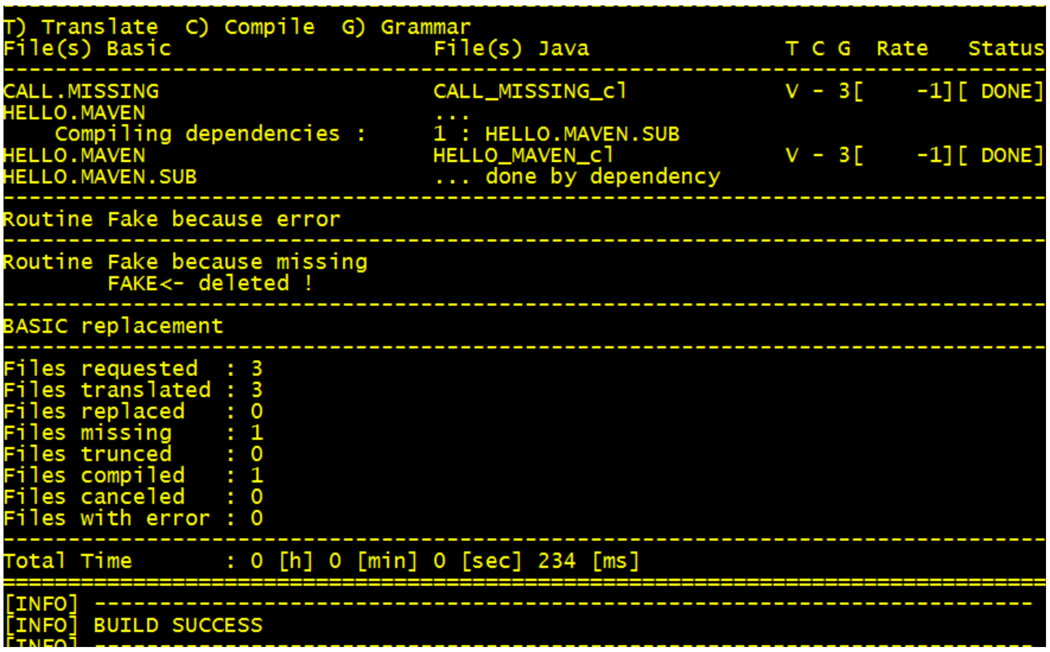
You should see the following output:



```
[INFO] --- tafj-maven-plugin:15.0.1:compile (compileBasicCode) @ TAFJMavenExample ---
[INFO] Basic file compilation requested
[INFO] ..........................................................................
[INFO] . Compile Basic files in C:\Signon2015\demo/basic
[INFO] . tafj.home is C:\Signon2015\demo/tafjHome
[INFO] . tafj properties file is tafj
[INFO] ..........................................................................
--------------------------------------------------------------------------------
Temenos TAFJ Compiler/Runner
TAFJCompiler-15.0.12.jar version "DEV_201503.0"
Copyright (c) 2009 TEMENOS. All rights reserved
--------------------------------------------------------------------------------
Java Version = 1.7.0_67-b01
Java File Encoding = Cp1252
Java Home = C:\javasoft\jdk1.7.0_67\jre
Java Classpath = Java(TM) SE Runtime Environment
--------------------------------------------------------------------------------
User Name = jncharpin
Current Dir = C:\Signon2015\demo
--------------------------------------------------------------------------------
OS Type = Windows 7
--------------------------------------------------------------------------------
TAFJ_HOME    = C:\Signon2015\demo\tafjHome
Basic Source Dir         = C:\Signon2015\demo/basic;
Insert Source Dir.       = C:\Signon2015\demo/basic;
Precompile          Dir.=
Java Destination Dir.    = C:\Signon2015\demo/src/main/java
Classes Destination Dir.= C:\Signon2015\demo/target/classes
Report Destination Dir. = C:\Signon2015\demo\tafjHome\report
Max. Grammar Level       = 3
Min. Grammar Level       = 0
Java Package             = com.temenos.t24
Reporting                = false
--------------------------------------------------------------------------------
Arguments :-cf tafj C:\Signon2015\demo/basic
--------------------------------------------------------------------------------
Argument used : 'C:\Signon2015\demo\basic'

--------------------------------------------------------------------------------
```

…



```
T) Translate  C) Compile  G) Grammar
File(s) Basic                        File(s) Java               T C G  Rate   Status
---------------------------------------------------------------------------------
CALL.MISSING                         CALL_MISSING_cl            V - 3[    -1][ DONE]
HELLO.MAVEN                          ...
    Compiling dependencies :         1 : HELLO.MAVEN.SUB
HELLO.MAVEN                          HELLO_MAVEN_cl             V - 3[    -1][ DONE]
HELLO.MAVEN.SUB                      ... done by dependency
---------------------------------------------------------------------------------
Routine Fake because error
---------------------------------------------------------------------------------
Routine Fake because missing
        FAKE<- deleted !
---------------------------------------------------------------------------------
BASIC replacement
---------------------------------------------------------------------------------
Files requested  : 3
Files translated : 3
Files replaced   : 0
Files missing    : 1
Files trunced    : 0
Files compiled   : 1
Files canceled   : 0
Files with error : 0
---------------------------------------------------------------------------------
Total Time       : 0 [h] 0 [min] 0 [sec] 234 [ms]
=================================================================================
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

The corresponding pom section is the following:

```
<!-- compile goal - Compile basic files with specified parameters -->
<!-- the example below is using the tafj compiler to generate the java
only -->
<!-- in that case maven compiler is used to generate the class -->
<execution>
      <id>compileBasicCode</id>
      <goals>
      <!-- default phase is generate-sources -->
            <goal>compile</goal>
      </goals>
      <configuration>
            <tafjHome>${tafjHome}</tafjHome>
            <tafjProperties>tafj</tafjProperties>
            <basicDir>${basedir}/basic</basicDir>
            <insertDir>${basedir}/basic</insertDir>
            <!-- maven compiler default java src path is
            project/src/main/java -->
            <javaDir>${basedir}/src/main/java</javaDir>
            <!-- not used when translate only set to true -->
            <!-- but should be the destination path to have maven packager
            finding the classes-->
            <classesDir>${basedir}/target/classes</classesDir>
            <javaPackage>com.temenos.t24</javaPackage>
            <!-- setup compilation to keep the java files and to just do
            the translation -->
            <properties>
                  <temn.tafj.compiler.keep.java>true</temn.tafj.compiler.ke
                  ep.java>
            </properties>
      </configuration>
</execution>
```

We specify in the configuration section:

- The tafj home
- The properties file (tafj project) to use during compilation
- The basic folder to compile
- The insert directory, optional, could be read from the tafj properties file
- The java directory, optional, could be read from the tafj properties file
- The classes directory, optional, could be read from the tafj properties file
- The java package, optional, could be read from the tafj properties file
- Some additional specific tafj properties under the <properties> section, in this example we want to keep the java files generated after compilation for the purpose of the demo.

Please note that you could use maven compiler to compile the java files by generating them under ${basedir}/src/main/java.

By using “keep java” true, and “translate only” true, you could speed up your compilation by using TAFJ parser to translate basic files to java; and maven compiler to compile.

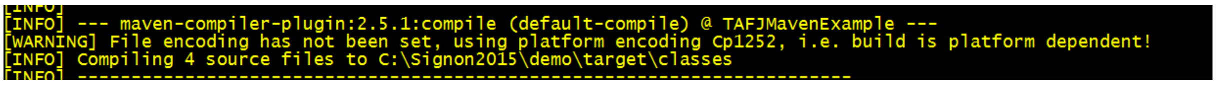
Please refer to compiler documentation for further informations.

```
<execution>
      <id>compileBasicCode</id>
      <goals>
            <goal>compile</goal>
      </goals>
      <configuration>
            <tafjHome>${tafjHome}</tafjHome>
            <tafjProperties>tafj</tafjProperties>
            <basicDir>${basedir}/basic</basicDir>
            <properties>
            <temn.tafj.compiler.keep.java>true</temn.tafj.compiler.keep.java>
            <temn.tafj.compiler.translate.only>true</temn.tafj.compiler.translate.
            only>
            </properties>
      </configuration>
</execution>
```

You could run following command to test this configuration.

```
mvn clean compile
```

You can see that TAFJ parser has translated the file only and that maven compiler compiled them.



```
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ TAFJMavenExample ---
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 4 source files to C:\Signon2015\demo\target\classes
[INFO] ------------------------------------------------------------------------
```

If you don't want maven compiler to compile your java files you would either have to not set the property "keep java" to true or don't generate the java files under ${basedir}/src/main/java

Clean fake

Fake classes could be generated during compilation when missing dependencies are found.

You could clean them even if the maven compiler is being used by running the goal clean-fake.

```
<!-- clean up the fake class marked during compile goal - translate
only true -->
<execution>
      <id>cleanFake</id>
      <goals>
            <!-- default phase is prepare-package -->
            <goal>clean-fake</goal>
      </goals>
</execution>
```

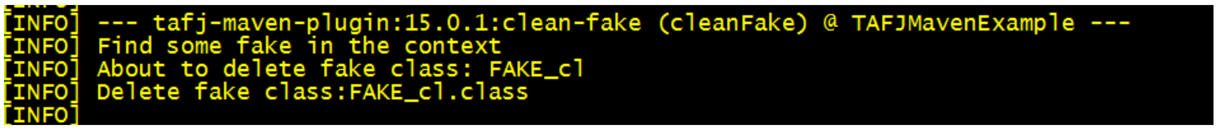
It has to be used in conjunction with compiler property:

```
<temn.tafj.compiler.no.fake.missing.java>true</temn.tafj.compiler.no.fake.m
issing.java>
```

This property needs to be added in the property section of the compile goal.

You could run following command to test this configuration.

```
mvn clean prepare-package
```



```
[INFO] --- tafj-maven-plugin:15.0.1:clean-fake (cleanFake) @ TAFJMavenExample ---
[INFO] Find some fake in the context
[INFO] About to delete fake class: FAKE_cl
[INFO] Delete fake class:FAKE_cl.class
[INFO]
```

TAFJ maven splitter

tafj-maven-plugin offers the capability to package the generated T24 classes according the product they belong to. You will have to use the plugin goal split, this goal is bound by default to the maven phase prepare-package.

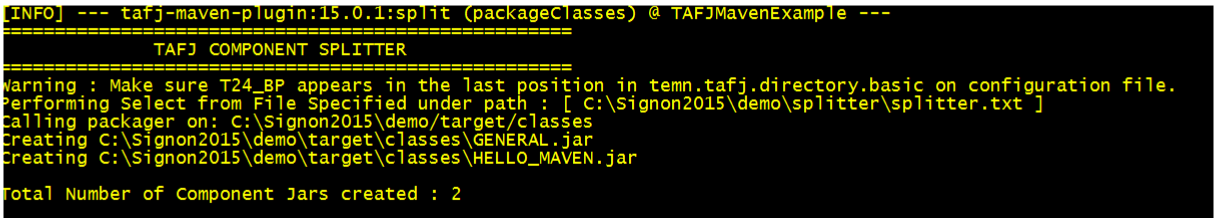
To package the classes in the appropriate jar, TAFJ provides a tool called component splitter.

This tool will either use F.PGM.DATA.CONTROL or a generated build file called splitter.txt.

Run the command

```
mvn clean prepare-package
```

You should see the following output.



```
[INFO] --- tafj-maven-plugin:15.0.1:split (packageClasses) @ TAFJMavenExample ---
====================================================
              TAFJ COMPONENT SPLITTER
====================================================
Warning : Make sure T24_BP appears in the last position in temn.tafj.directory.basic on configuration file.
Performing Select from File Specified under path : [ C:\Signon2015\demo\splitter\splitter.txt ]
Calling packager on: C:\Signon2015\demo/target/classes
Creating C:\Signon2015\demo\target\classes\GENERAL.jar
Creating C:\Signon2015\demo\target\classes\HELLO_MAVEN.jar

Total Number of Component Jars created : 2
```

The corresponding pom section is the following:

```
<!-- Package the classes according to T24 product code -->
<execution>
      <id>packageClasses</id>
      <goals>
            <goal>split</goal>
      </goals>
      <configuration>
            <splitterTxtFile>$
      {basedir}/splitter/splitter.txt</splitterTxtFile>
      </configuration>
</execution>
```

It also depends on your properties file, these properties are applied to the properties file during the configure goal (refer to configure section).

We use here the splitter.txt file packaged with the demo as an example.

The following configuration doesn't specify the splitterTxtFile argument, thus the splitter will use F.PGM.DATA.CONTROL to package the classes.

```
<!-- Package the classes according to T24 product code -->
<execution>
      <id>packageClasses</id>
      <goals>
```

```
            <goal>split</goal>
      </goals>
</execution>
```

In that case you will need to configure your properties file with property `temn.tafj.pgm.data.control` pointing a correct path to F.PGM.DAT000 file (refer to configure section).

```
<!-- tComponentSplitter properties -->
<!-- Path to F.PGM.DATA.CONTROL if you don't want to use splitter.txt
file -->

<temn.tafj.pgm.data.control>${basedir}/F.PGM.DAT000</temn.tafj.pgm.data.control>
```

You will find the generated jars under your classes directory which is for the demo ${basedir}/target/classes.

TAFJ Maven dbimport

By using goal dbimport of tafj-maven-plugin you could use TAFJ DBImport capabilities from a maven project (please refer to related TAFJ DB documentation).

This goal is bound by default to maven phase pre-integration-test.

If you are targeting an H2 database you could create it from scratch. This is demonstrated in the demo in the following pom section:

```
<execution>
      <goals>
            <goal>dbimport</goal>
      </goals>
      <configuration>
            <tafjHome>${tafjHome}</tafjHome>
            <createDefaultDB>true</createDefaultDB>
            <dbImportProperties>
                  <Url>jdbc:h2:${tafjHome}/h2/Data/DEMODB</Url>
                  <Driver>org.h2.Driver</Driver>
                  <User>tafj</User>
                  <Password>tafj</Password>
                  <!-- Add a path to a VOC file -->
                  <VocFile>Valid path to VOC file</VocFile>
                  <UserDirectories>${tafjHome}/h2/Data/UD</UserDirectories>
                  <ClearTables>Yes</ClearTables>
                  <Layout>STRING</Layout>
                  <!-- Change this parameter to All if you want to perform
                  full DBImport-->
                  <File>FBNK.CURRENCY</File>
            </dbImportProperties>
      </configuration>
</execution>
```

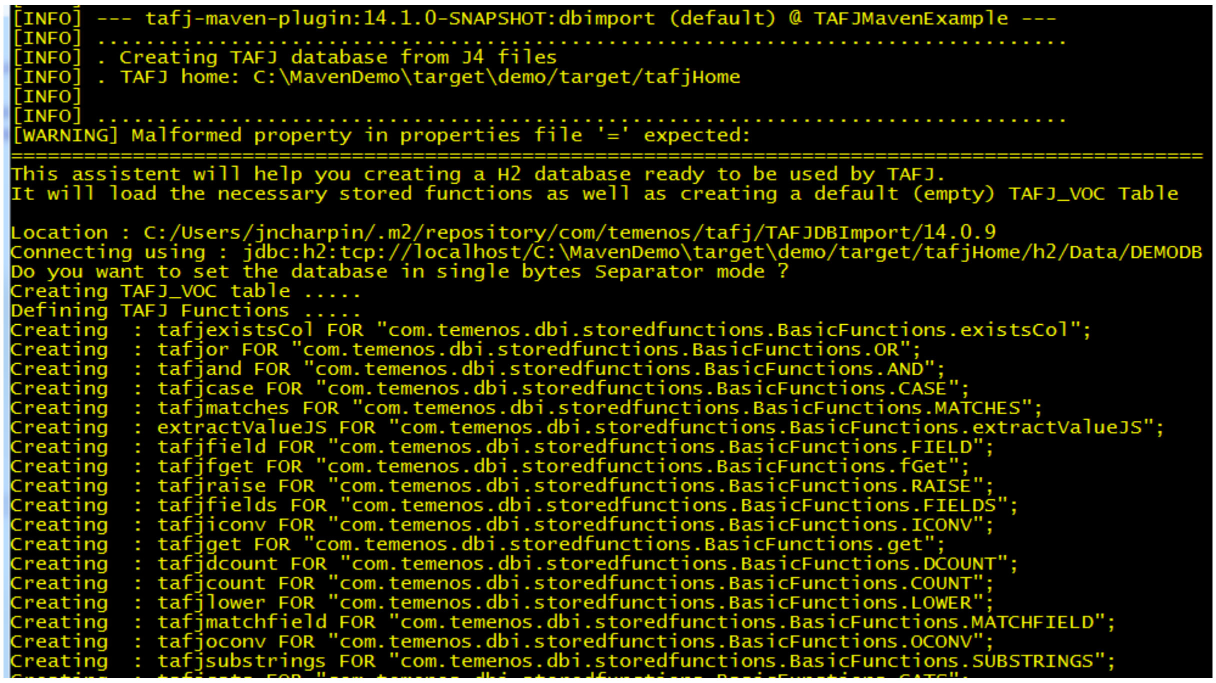
We set here a parameter “createDefaultDB” to true, this parameter is available for h2 only and will create a database under the specified “Url” parameter and load the TAFJ stored functions.

You will have to set the parameter “VocFile” to a valid VOC file or the demo will fail.

This configuration is defined within a profile named h2, run the following command to test it:

```
mvn clean pre-integration-test -P h2
```

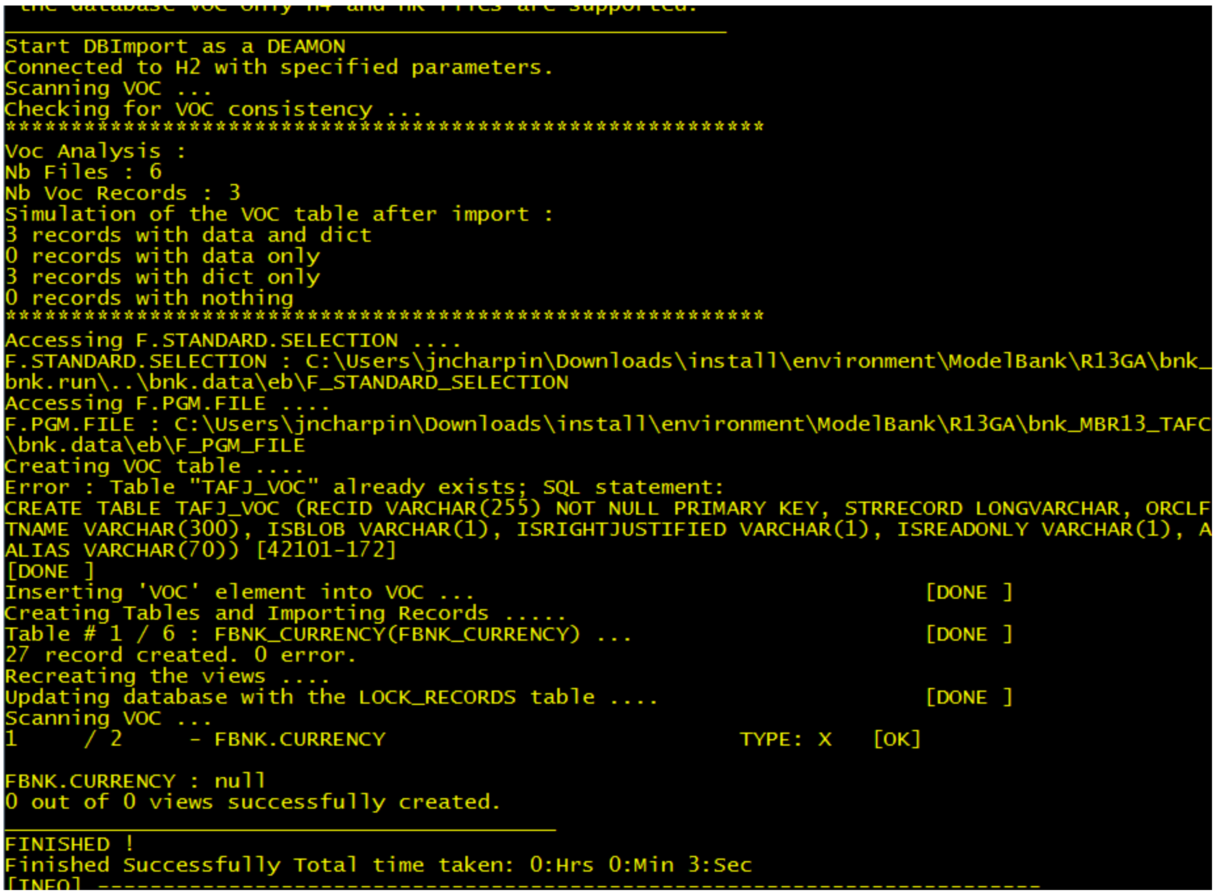
You should see the following output:



```
[INFO] --- tafj-maven-plugin:14.1.0-SNAPSHOT:dbimport (default) @ TAFJMavenExample ---
[INFO] ..............................................................................
[INFO] . Creating TAFJ database from J4 files
[INFO] . TAFJ home: C:\MavenDemo\target\demo/target/tafjHome
[INFO]
[INFO] ..............................................................................
[WARNING] Malformed property in properties file '=' expected:
==================================================================================================
This assistent will help you creating a H2 database ready to be used by TAFJ.
It will load the necessary stored functions as well as creating a default (empty) TAFJ_VOC Table

Location : C:/Users/jncharpin/.m2/repository/com/temenos/tafj/TAFJDBImport/14.0.9
Connecting using : jdbc:h2:tcp://localhost/C:\MavenDemo\target\demo/target/tafjHome/h2/Data/DEMODB
Do you want to set the database in single bytes Separator mode ?
Creating TAFJ_VOC table .....
Defining TAFJ Functions .....
Creating  : tafjexistsCol FOR "com.temenos.dbi.storedfunctions.BasicFunctions.existsCol";
Creating  : tafjor FOR "com.temenos.dbi.storedfunctions.BasicFunctions.OR";
Creating  : tafjand FOR "com.temenos.dbi.storedfunctions.BasicFunctions.AND";
Creating  : tafjcase FOR "com.temenos.dbi.storedfunctions.BasicFunctions.CASE";
Creating  : tafjmatches FOR "com.temenos.dbi.storedfunctions.BasicFunctions.MATCHES";
Creating  : extractValueJS FOR "com.temenos.dbi.storedfunctions.BasicFunctions.extractValueJS";
Creating  : tafjfield FOR "com.temenos.dbi.storedfunctions.BasicFunctions.FIELD";
Creating  : tafjfget FOR "com.temenos.dbi.storedfunctions.BasicFunctions.fGet";
Creating  : tafjraise FOR "com.temenos.dbi.storedfunctions.BasicFunctions.RAISE";
Creating  : tafjfields FOR "com.temenos.dbi.storedfunctions.BasicFunctions.FIELDS";
Creating  : tafjiconv FOR "com.temenos.dbi.storedfunctions.BasicFunctions.ICONV";
Creating  : tafjget FOR "com.temenos.dbi.storedfunctions.BasicFunctions.get";
Creating  : tafjdcount FOR "com.temenos.dbi.storedfunctions.BasicFunctions.DCOUNT";
Creating  : tafjcount FOR "com.temenos.dbi.storedfunctions.BasicFunctions.COUNT";
Creating  : tafjlower FOR "com.temenos.dbi.storedfunctions.BasicFunctions.LOWER";
Creating  : tafjmatchfield FOR "com.temenos.dbi.storedfunctions.BasicFunctions.MATCHFIELD";
Creating  : tafjoconv FOR "com.temenos.dbi.storedfunctions.BasicFunctions.OCONV";
Creating  : tafjsubstrings FOR "com.temenos.dbi.storedfunctions.BasicFunctions.SUBSTRINGS";
Creating  : tafjcats FOR "com.temenos.dbi.storedfunctions.BasicFunctions.CATS";
```

…



```
 the database VOC only H4 and HK files are supported.
_________________________________________________________
Start DBImport as a DEAMON
Connected to H2 with specified parameters.
Scanning VOC ...
Checking for VOC consistency ...
**********************************************************
Voc Analysis :
Nb Files : 6
Nb Voc Records : 3
Simulation of the VOC table after import :
3 records with data and dict
0 records with data only
3 records with dict only
0 records with nothing
**********************************************************
Accessing F.STANDARD.SELECTION ....
F.STANDARD.SELECTION : C:\Users\jncharpin\Downloads\install\environment\ModelBank\R13GA\bnk_
bnk.run\..\bnk.data\eb\F_STANDARD_SELECTION
Accessing F.PGM.FILE ....
F.PGM.FILE : C:\Users\jncharpin\Downloads\install\environment\ModelBank\R13GA\bnk_MBR13_TAFC
\bnk.data\eb\F_PGM_FILE
Creating VOC table ....
Error : Table "TAFJ_VOC" already exists; SQL statement:
CREATE TABLE TAFJ_VOC (RECID VARCHAR(255) NOT NULL PRIMARY KEY, STRRECORD LONGVARCHAR, ORCLF
TNAME VARCHAR(300), ISBLOB VARCHAR(1), ISRIGHTJUSTIFIED VARCHAR(1), ISREADONLY VARCHAR(1), A
ALIAS VARCHAR(70)) [42101-172]
[DONE ]
Inserting 'VOC' element into VOC ...                                      [DONE ]
Creating Tables and Importing Records .....
Table # 1 / 6 : FBNK_CURRENCY(FBNK_CURRENCY) ...                          [DONE ]
27 record created. 0 error.
Recreating the views ....
Updating database with the LOCK_RECORDS table ....                        [DONE ]
Scanning VOC ...
1     / 2     - FBNK.CURRENCY                            TYPE: X    [OK]

FBNK.CURRENCY : null
0 out of 0 views successfully created.
__________________________________________
FINISHED !
Finished Successfully Total time taken: 0:Hrs 0:Min 3:Sec
[INFO] ------------------------------------------------------------------------
```

For the demo purpose we are just importing a unique table specified in parameter `<File>FBNK.CURRENCY</File>` you could simply remove this parameter from the configuration or specify the value "All" if you want to run a full DBImport. It should take about 5 minutes to complete for a full T24 database.

Same kind of configuration would work for Oracle or DB2 database, you would just have to use an existing database instance and change the pom parameters to match your database.

i.e for Oracle database:

```
<execution>
      <goals>
            <goal>dbimport</goal>
      </goals>
      <configuration>
            <tafjHome>${tafjHome}</tafjHome>
            <dbImportProperties>
                  <Url>jdbc:oracle:thin:@localhost:1521:TESTDB</Url>
                  <Driver>oracle.jdbc.driver.OracleDriver</Driver>
                  <User>tafj</User>
                  <Password>tafj</Password>
                  <VocFile>/home/tafj/jenkinsBuild/T24_DEV_DBIMPORT/bnk/bnk
                  .run/VOC</VocFile>
                  <UserDirectories>$
            {basedir}/Oracle/Data/UD</UserDirectories>
            </dbImportProperties>
      </configuration>
</execution>
```

i.e for DB2 database:

```
<execution>
      <goals>
            <goal>dbimport</goal>
      </goals>
      <configuration>
            <tafjHome>${tafjHome}</tafjHome>
            <dbImportProperties>
                  <Url>jdbc:db2://localhost:50000/TESTDB</Url>
                  <Driver>com.ibm.db2.jcc.DB2Driver</Driver>
                  <User>tafj</User>
                  <Password>tafj</Password>
                  <VocFile>/home/tafj/jenkinsBuild/T24_DEV_DBIMPORT/bnk/bnk
                  .run/VOC</VocFile>
                  <UserDirectories>${basedir}/DB2/Data/UD</UserDirectories>
            </dbImportProperties>
      </configuration>
</execution>
```

Please note that the plugin will not create the TAFJ stored functions for oracle and DB2.