LAB EXERCISE 9

Aim: Build Profile using Maven

A Build profile is a set of configuration values, which can be used to set or override default values of Maven build. Using a build profile, you can customize build for different environments such as Production v/s Development environments.

Profiles are specified in pom.xml file using its activeProfiles/profiles elements and are triggered in variety of ways. Profiles modify the POM at build time, and are used to give parameters different target environments (for example, the path of the database server in the development, testing, and production environments).

Types of Build Profile

Build profiles are majorly of three types.

Type	Where it is defined
Per Project	Defined in the project POM file, pom.xml
Per User	Defined in Maven settings xml file (%USER_HOME%/.m2/settings.xml)
Global	Defined in Maven global settings xml file (%M2_HOME%/conf/settings.xml)

Profile Activation

A Maven Build Profile can be activated in various ways.

- Explicitly using command console input.
- Through maven settings.
- Based on environment variables (User/System variables).
- OS Settings (for example, Windows family).
- Present/missing files.

Profile Activation Examples

Explicit Profile Activation

In the following example, we will attach maven-antrun-plugin:run goal to test the phase. This will allow us to echo text messages for different profiles. We will be using pom.xml to define different profiles and will activate profile at command console using maven command.

Assume, we've created the following pom.xml in C:\MVN\project folder.

```
<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.companyname.projectgroup</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
   <profiles>
      <profile>
         <id>test</id>
         <build>
            <plugins>
               <plugin>
                  <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-antrun-plugin</artifactId>
                  <version>1.1</version>
                  <executions>
                     <execution>
                        <phase>test</phase>
                        <goals>
                           <goal>run</goal>
                        </goals>
                        <configuration>
                           <tasks>
                              <echo>Using
env.test.properties</echo>
                              <copy
file="src/main/resources/env.test.properties"

tofile="${project.build.outputDirectory}
                                 /env.properties"/>
                           </tasks>
                        </configuration>
                     </execution>
                  </executions>
               </plugin>
            </plugins>
         </build>
      </profile>
   </profiles>
</project>
```

Now open the command console, go to the folder containing pom.xml and execute the following **mvn** command. Pass the profile name as argument using -P option.

```
C:\MVN\project>mvn test -Ptest
```

Maven will start processing and displaying the result of test build profile.

```
[INFO] Scanning for projects...
[INFO] -----------------------------------------------------------------
---------
[INFO] Building Unnamed -
com.companyname.projectgroup:project:jar:1.0
[INFO] task-segment: [test]
[INFO] -----------------------------------------------------------------
---------
[INFO] [resources:resources {execution: default-resources}]

[WARNING] Using platform encoding (Cp1252 actually) to copy
filtered resources,
i.e. build is platform dependent!

[INFO] Copying 3 resources
[INFO] [compiler:compile {execution: default-compile}]
[INFO] Nothing to compile - all classes are up to date
[INFO] [resources:testResources {execution: default-
testResources}]

[WARNING] Using platform encoding (Cp1252 actually) to copy
filtered resources,
i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory
C:\MVN\project\src\test\resources
[INFO] [compiler:testCompile {execution: default-testCompile}]
[INFO] Nothing to compile - all classes are up to date
[INFO] [surefire:test {execution: default-test}]
[INFO] Surefire report directory:
C:\MVN\project\target\surefire-reports

-------------------------------------------------------
T E S T S
```

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

There are no tests to run.
Results :
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO] [antrun:run {execution: default}]
[INFO] Executing tasks
[echo] Using env.test.properties
[INFO] Executed tasks

[INFO] -------------------------------------------------------
---------
[INFO] BUILD SUCCESSFUL
[INFO] -------------------------------------------------------
---------

[INFO] Total time: 1 second
[INFO] Finished at: Sun Jul 08 14:55:41 IST 2012
[INFO] Final Memory: 8M/64M
[INFO] -------------------------------------------------------
---------
```

Now as an exercise, you can perform the following steps −

- Add another profile element to profiles element of pom.xml (copy existing profile element and paste it where profile elements ends).
- Update id of this profile element from test to normal.
- Update task section to echo env.properties and copy env.properties to target directory.
- Again repeat the above three steps, update id to prod and task section for env.prod.properties.
- That's all. Now you've three build profiles ready (normal/test/prod).

Now open the command console, go to the folder containing pom.xml and execute the following **mvn** commands. Pass the profile names as argument using -P option.

```
C:\MVN\project>mvn test -Pnormal

C:\MVN\project>mvn test -Pprod
```

Check the output of the build to see the difference.

Profile Activation via Maven Settings

Open Maven **settings.xml** file available in %USER_HOME%/.m2 directory where **%USER_HOME%** represents the user home directory. If settings.xml file is not there, then create a new one.

Add test profile as an active profile using active Profiles node as shown below in example.

```
<settings 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/settings-1.0.0.xsd">
   <mirrors>
      <mirror>
         <id>maven.dev.snaponglobal.com</id>
         <name>Internal Artifactory Maven repository</name>
         <url>http://repo1.maven.org/maven2/</url>
         <mirrorOf>*</mirrorOf>
      </mirror>
   </mirrors>
   <activeProfiles>
      <activeProfile>test</activeProfile>
   </activeProfiles>
</settings>
```

Now open command console, go to the folder containing pom.xml and execute the following **mvn** command. Do not pass the profile name using -P option. Maven will display result of test profile being an active profile.

```
C:\MVN\project>mvn test
```

Profile Activation via Environment Variables

Now remove active profile from maven settings.xml and update the test profile mentioned in pom.xml. Add activation element to profile element as shown below.

The test profile will trigger when the system property "env" is specified with the value "test". Create an environment variable "env" and set its value as "test".

```
<profile>
   <id>test</id>
   <activation>
      <property>
         <name>env</name>
         <value>test</value>
      </property>
   </activation>
</profile>
```

Let's open command console, go to the folder containing pom.xml and execute the following **mvn** command.

```
C:\MVN\project>mvn test
```

Profile Activation via Operating System

Activation element to include os detail as shown below. This test profile will trigger when the system is windows XP.

```
<profile>
   <id>test</id>
   <activation>
      <os>
         <name>Windows XP</name>
         <family>Windows</family>
         <arch>x86</arch>
         <version>5.1.2600</version>
      </os>
   </activation>
</profile>
```

Now open command console, go to the folder containing pom.xml and execute the following **mvn** commands. Do not pass the profile name using -P option. Maven will display result of test profile being an active profile.

```
C:\MVN\project>mvn test
```

Profile Activation via Present/Missing File

Now activation element to include OS details as shown below. The test profile will trigger when **target/generated-sources/axistools/wsdl2java/com/companyname/group** is missing.

```
<profile>
   <id>test</id>
   <activation>
      <file>
         <missing>target/generated-sources/axistools/wsdl2java/
           com/companyname/group</missing>
      </file>
   </activation>
</profile>
```

Now open the command console, go to the folder containing pom.xml and execute the following **mvn** commands. Do not pass the profile name using -P option. Maven will display result of test profile being an active profile.

```
C:\MVN\project>mvn test
```