Build Profiles

Introduction to Build Profiles

A Build profile is a kind of mechanism for `triggering a set of build configurations`. These configurations mainly determine the `values for different build environments` like **production, stage, test, or development environment**. But it doesn't have to be just about environmental values. Any kind of configuration could be added to build.

There are three types of build profiles. Which are `"Per Project", "Per User" and "Global"`. During the course, we will be dealing with the "Per Project" type of profile. The project type of profile is specified in the project's POM file. For more information about "Per User" and "Global" profile type, you can click on the related link.

So, What something like is a profile?

In the example below there are two profiles. The first one is executed by default. The other can be executed by using one of the profile activation methods explained below and in upcoming pages.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">
    <parent>
        <groupId>com.clarusway.maven</groupId>
        <artifactId>profiles</artifactId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>profile-1</artifactId>
    <profiles>
        <profile>
            <id>dev</id>
            <activation>
                <!-- this profile is active by default -->
                <activeByDefault>true</activeByDefault>
                <!-- activate if system properties 'env=dev' -->
                <property>
                    <name>env</name>
                    <value>dev</value>
                </property>
            </activation>
            <properties>
                <db.driverClassName>com.mysql.jdbc.Driver</db.driverClassName>
                <db.url>jdbc:mysql://localhost:3306/dev</db.url>
                <db.username>clarus</db.username>
                <db.password>123456789</db.password>
            </properties>
        </profile>
        <profile>
            <id>prod</id>
            <activation>
                <!-- activate if system properties 'env=prod' -->
                <property>
                    <name>env</name>
                    <value>prod</value>
                </property>
            </activation>
            <properties>
                <db.driverClassName>com.mysql.jdbc.Driver</db.driverClassName>
                <db.url>jdbc:mysql://database-1.cdbs4t6jyjpw.us-east-1.rds
                    .amazonaws.com:3306/prod</db.url>
                <db.username>clarus</db.username>
                <db.password>123456789</db.password>
            </properties>
        </profile>
    </profiles>
    <build>
        <!-- you can map a variable with the  ${} syntax -->
        <resources>
            <resource>
                <directory>src/main/resources</directory>
                <filtering>true</filtering>
            </resource>
        </resources>
    </build>
</project>

```

Profile Activation

Profiles can be active by default using a POM file configuration like the following:

```xml
<profiles>
  <profile>
    <id>profile-1</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

This profile will automatically be active for all builds unless another profile in the same POM is activated using any other method which will be discussed later. All profiles that are active by default are automatically **deactivated when a profile in the POM file is activated on the command line** or **through its activation config.**

For a "Per Project" type of profile, after being defined in the POM, a way of profile activation could be declared. Maven Build Profiles can be activated in five different ways. Which are using `explicit profile activation, maven settings, environment variables (user/system variables), Operating System Settings, present/missing files.`

Explicit Profile Activation

Profiles can be explicitly triggered using the `-P option in a CLI command`. This option takes an argument that is a `comma-delimited list of profile-ids`. When this option is used, the profile(s) specified in the CLI command will be activated. -P option takes `<id>` tag value of the `profile`.

```
mvn groupId:artifactId:goal -P profile-1,profile-2
```

```xml
<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>

```

When you run `mvn test -P test` command, the output will be as in below.

```
[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] ------------------------------------------------------------------

```

Profile Activation via Maven Settings

When you install Maven to your local environment, a directory named `.m2` is created under your Home Directory. Under this directory, there should be a file named `settings.xml`. If it's not there, you can create one. For more information about the settings.xml file, you can visit this site.

To activate a profile with settings.xml, the profile id should be declared under `<activeProfile>` tag in the settings.xml file.

```xml
<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>

```

After this declaration, `you don't need to specify -P option` while you're executing mvn commands. Your profile will be automatically in use.

Profile Activation via Environment Variables

Similar to explicit profile activation, you can also activate your profile using `<activation>` tag in your POM file under `<profile>` tag.

The profile below will be activated when the system property "debug" is specified with -D option in your CLI commands.

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>debug</name>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

The following profile will be activated when the system property "debug" is not defined at all:

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>!debug</name>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

The following profile will be activated when the system property "debug" is not defined, or is defined with a value which is not "true".

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>debug</name>
        <value>!true</value>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

To activate this you would type one of those on the command line:

`mvn groupId:artifactId:goal`, `mvn groupId:artifactId:goal -Ddebug=false`

The next example will trigger the profile when the system property "environment" is specified with the value "test":

```
<profiles>
  <profile>
    <activation>
      <property>
        <name>environment</name>
        <value>test</value>
      </property>
    </activation>
    .
    .
    .
  </profile>
</profiles>
```

To activate this you would type this on the command line:

`mvn groupId:artifactId:goal -Denvironment=test`

Profile Activation via Operating System

Activation with Operating System is specified under `<os>` tag. The below example requires Windows XP operating system and other features declared in the configurations. So, your profile with automatically triggered if you are using Windows XP operating system.

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

Profile Activation via Present/Missing File

In the example below, the profile will be triggered when the generated file `target/generated-sources/axistools/wsdl2java/org/apache/maven` is missing.

```
<profiles>
  <profile>
    <activation>
      <file>
        <missing>target/generated-sources/axistools/wsdl2java/org/apache/maven</missing>
      </file>
    </activation>
    ...
  </profile>
</profiles>
```

As of Maven 2.0.9, the tags `<exists>` and `<missing>` could be interpolated. Supported variables are system properties like ${user.home} and environment variables like ${env.HOME}.