Build Lifecycle

Introduction to Build Lifecycle

The Maven Build Lifecycle is `a track that is comprised of different number of phases.` A phase is a job unit or a specific stage in a lifecycle. Maven follows a specific life cycle in order to deliver a target project.

There are `three built-in lifecycles` in Maven: These are `default, clean, and site`. **Default** is the main life cycle as it's responsible for project deployment. **Clean** is used for cleaning the project from all the files generated by the previous build or builds. The third lifecycle is **site** lifecycle. It is used for building your project's website documentation.

Each life cycle has a different number of phases. The `default build lifecycle has 23, the clean lifecycle has 3 and the site lifecycle has 4 phases.`

> **Note:** While building a project, Maven goes through a sequence of phases and goals. A goal is a small task that has a specific output or a duty either inside a build phase or independently. An independent goal or any build phase could be executed from the command line.

Using Command-Line

You should run the appropriate maven command to achieve what you want as an output. For example, if you want jar file to be produced, you should run the command `mvn package`. Or if you want to run unit tests, then you should run the command `mvn test`.

For a clean build, you had better use `mvn clean` command first and then other commands that produce outputs into your project's target directory or to your local or remote maven repo.

Clean Lifecycle

There are `three phases in Clean Lifecycle`. These are `pre-clean, clean, and post-clean phases.` These phases are in sequence. So, when a phase is called (for example "mvn post-clean"), the `phases prior to that phase are also automatically run`. Which means the pre-clean phase is always run.

Maven's "clean:clean" goal is one of the **bound goals** to the clean phase. It cleans the project's target directory. In this context, clean means to **delete all build files located in the target directory.** The pre-clean phase can be used for any tasks required prior to the cleanup, and the post-clean phase can be used for tasks following the cleanup.

We can bind any other goal to a phase. In the POM file below, maven-antrun-plugin:run goal is bound to the pre-clean, clean, and post-clean phases. This will result in **echoing text messages** displaying the phases of the clean lifecycle.

```
<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>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
               <execution>
                  <id>id.pre-clean</id>
                  <phase>pre-clean</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>pre-clean phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.clean</id>
                  <phase>clean</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>clean phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.post-clean</id>
                  <phase>post-clean</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>post-clean phase</echo>
                     </tasks>
                  </configuration>
               </execution>
            </executions>
         </plugin>
      </plugins>
   </build>
</project>
```

When "mvn post-clean" command is run, the outcome will be as in the case below. You can see the phase switches in the lines starting with [echo].

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------< com.clarus.maven:maven-experiment >-----------------
[INFO] Building maven-experiment 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.pre-clean) @ maven-experiment ---
[INFO] Executing tasks
     [echo] pre-clean phase
[INFO] Executed tasks
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ maven-experiment ---
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.clean) @ maven-experiment ---
[INFO] Executing tasks
     [echo] clean phase
[INFO] Executed tasks
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.post-clean) @ maven-experiment ---
[INFO] Executing tasks
     [echo] post-clean phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.856 s
[INFO] Finished at: 2020-07-25T22:08:56+03:00
[INFO] ------------------------------------------------------------------------
```

Default Lifecycle

The default lifecycle is `used for application build.` In total there are 23 phases in this lifecycle. The most important phases are seen below.

- **validate:** validates the project if it is correct and all necessary information is available
- **compile:** compiles the source code
- **test-compile:** compiles the test source code
- **test:** runs unit tests
- **package:** packages compiled source code into the distributable format (jar, war, ear)
- **integration-test:** processes and deploys the package if needed to run integration test
- **install:** installs the package (jar, war ,ear...) to a local repository (namely into ~/.m2/repository/)
- **deploy:** copies the package to a remote repository (This can vary...)

All the phases are listed in here.

As it's explained in the clean lifecycle, maven commands are run in a sequence and it's also valid for default lifecycle commands. Another important concept in Maven is that `you can bind different types of goals` according to selected package type like jar, war, or ear.

In the example below, maven-antrun-plugin: the run goal is attached to see the switches to validate, compile, test, package, and deploy phases. With an echo tag, you can follow the steps in the terminal.

In the example below, maven-antrun-plugin: the run goal is attached to see the switches to validate, compile, test, package, and deploy phases. With an echo tag, you can follow the steps in the terminal.

```
<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>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
               <execution>
                  <id>id.validate</id>
                  <phase>validate</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>validate phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.compile</id>
                  <phase>compile</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>compile phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.test</id>
                  <phase>test</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>test phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
```

When you execute "mvn install" command in the directory where your POM file is located, you will see the output as below.

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.yourcompany.maven:maven-test >------------------
[INFO] Building maven-test 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.validate) @ maven-test ---
[INFO] Executing tasks
     [echo] validate phase
[INFO] Executed tasks
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven-test
    ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
    build is platform dependent!
[INFO] skip non existing resourceDirectory /Users/home/Desktop/folders/15_Maven
    /maven-test/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ maven-test ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build
    is platform dependent!
[INFO] Compiling 1 source file to /Users/home/Desktop/folders/15_Maven/maven-test
    /target/classes
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.compile) @ maven-test ---
[INFO] Executing tasks
     [echo] compile phase
[INFO] Executed tasks
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ maven
    -test ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
    build is platform dependent!
[INFO] skip non existing resourceDirectory /Users/home/Desktop/folders/15_Maven
    /maven-test/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ maven-test
    ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build
    is platform dependent!
[INFO] Compiling 1 source file to /Users/home/Desktop/folders/15_Maven/maven-test
    /target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-test ---
[INFO] Surefire report directory: /Users/home/Desktop/folders/15_Maven/maven-test
    /target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.yourcompany.maven.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.019 sec
```

Site Lifecycle

The site lifecycle has **four phases.** These are; `pre-site, site, post-site, and site-deploy.` As usual, calling one phase of the site lifecycle results in the execution of all phases up to that target phase.

For the phases of Site Lifecycle, the Site Plugin is used. The plugin's `main duty is to generate a website for the project`. The generated site can also include the project's reports configured in the POM file.

Maven's "site:site" goal is typically bound to the site phase, and the Maven "site:deploy" goal is bound to the site-deploy phase. The pre-site and post-site phases aren't used, but they can be utilized to perform any kind of pre-documentation or post-documentation tasks, and the post-site phase could potentially be used for any tasks required prior to site deployment.

The Site Plugin has seven goals:

- **site:site** is used to generate a site for a single project. Note that `it will not work in a multi-module build.`
- **site:deploy** is used to deploy the generated site to the site URL specified in the section of the POM.
- **site:run** starts the site up, rendering documents as requested for faster editing. It uses Jetty as the webserver.
- **site:stage** is used to test the links between module sites in a multi-module build work. This goal requires the site to already have been generated using the site goal, such as by calling `mvn site.`
- **site:stage-deploy** deploys the generated site to a staging or mock directory to the site URL specified in the section of the POM.
- **site:attach-descriptor** adds the site descriptor (site.xml) to the list of files to be installed/deployed.
- **site:jar** bundles the site output into a JAR so that it can be deployed to a repository.
- **site:effective-site** calculates the effective site descriptor, after inheritance and interpolation.

> ⚠ **Caution:** If you are having trouble with the default Maven site plugin (3.3.1), you should add the following plugins into your POM file.

```
<build>
   <plugins>
      <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>3.7.1</version>
      </plugin>
      <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-project-info-reports-plugin</artifactId>
            <version>3.0.0</version>
      </plugin>
   </plugins>
</build>
```

In the example below, maven-antrun-plugin:run goal is attached to see the switches for site lifecycle phases. With an echo tag, you can follow the steps in the terminal.

```
<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>
   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
               <execution>
                  <id>id.pre-site</id>
                  <phase>pre-site</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>pre-site phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.site</id>
                  <phase>site</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>site phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.post-site</id>
                  <phase>post-site</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>post-site phase</echo>
                     </tasks>
                  </configuration>
               </execution>
               <execution>
                  <id>id.site-deploy</id>
                  <phase>site-deploy</phase>
                  <goals>
                     <goal>run</goal>
                  </goals>
                  <configuration>
                     <tasks>
                        <echo>site-deploy phase</echo>
                     </tasks>
                  </configuration>
               </execution>
            </executions>
         </plugin>
      </plugins>
   </build>
</project>
```

When you run the "mvn site" command, the output will be as seen below.

```
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building Unnamed - com.companyname.projectgroup:project:jar:1.0
[INFO] task-segment: [site]
[INFO] ------------------------------------------------------------------------
[INFO] [antrun:run {execution: id.pre-site}]
[INFO] Executing tasks
[echo] pre-site phase
[INFO] Executed tasks
[INFO] [site:site {execution: default-site}]


[INFO] Generating "About" report.
[INFO] Generating "Issue Tracking" report.
[INFO] Generating "Project Team" report.
[INFO] Generating "Dependencies" report.
[INFO] Generating "Project Plugins" report.
[INFO] Generating "Continuous Integration" report.
[INFO] Generating "Source Repository" report.
[INFO] Generating "Project License" report.
[INFO] Generating "Mailing Lists" report.
[INFO] Generating "Plugin Management" report.
[INFO] Generating "Project Summary" report.

[INFO] [antrun:run {execution: id.site}]
[INFO] Executing tasks
[echo] site phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3 seconds
[INFO] Finished at: Sat Jul 07 15:25:10 IST 2012
[INFO] Final Memory: 24M/149M
[INFO] ------------------------------------------------------------------------
```