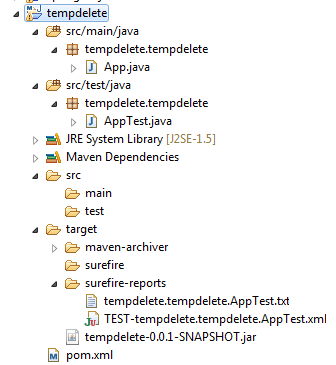
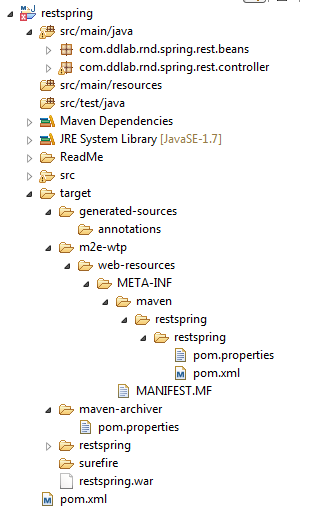
**Maven**

**Structure of Maven thick client – Desktop or Standalone application**



**Structure of Maven thin client – Web Application**



**The basic structure of Maven pom.xml file**

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>tempdelete</groupId>

<artifactId>tempdelete</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>tempdelete</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

**<dependency>**

**<groupId>junit</groupId>**

**<artifactId>junit</artifactId>**

**<version>3.8.1</version>**

**<scope>test</scope>**

**</dependency>**

</dependencies>

</project>

**modelVersion** – This element indicates what version of the object model this POM is using.

Example : **<modelVersion>4.0.0</modelVersion>**

**groupId** will identify your project uniquely across all projects. This element indicates the unique identifier of the organization or group that created the project.

Example : **<groupId>restspring</groupId>**

**artifactId** is the name of the jar without version.

Example : **<artifactId>restspring</artifactId>**

**packaging** This element indicates the package type to be used by this artifact (e.g. JAR, WAR, EAR, etc.).

Example : **<packaging>war</packaging>**

**version** This element indicates the version of the artifact generated by the project.

Example : **<version>0.0.1-SNAPSHOT</version>**

**name** This element indicates the display name used for the project.

Example : **<name>restspring Maven Webapp</name>**

**url** This element indicates where the project's site can be found.

Example: **<url>http://maven.apache.org</url>**

**description** This element provides a basic description of your project.

Example : **<description>restspring Maven Webapp</description>**.

**What is Surefire plugin in Maven**

The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application.

It generates reports in 2 different file formats:

Plain text files (\*.txt)

XML files (\*.xml)

By default, these files are generated at ${basedir}/target/surefire-reports.

The **<build>** element contains informations required to build the project. It contains element like sourceDirectory, outputDirectory , defaultGoal etc.

**Dependency scope**

**Dependency scope** is used to limit the transitivity of a dependency, and also to affect the classpath used for various build tasks.

There are 6 scopes available:

**Compile :** This is the default scope, compile dependencies are available in all classpaths of a project.

**Provided :** This is much like compile, but indicates you expect the JDK or a container to provide the dependency at runtime. For example, when building a web application for the Java Enterprise Edition, you would set the dependency on the Servlet API and related Java EE APIs to scope provided because the web container provides those classes. This scope is only available on the compilation and test classpath, and is not transitive.

**Runtime :** This scope indicates that the dependency is not required for compilation, but is for execution.

**Test :** This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases.

**System :** This scope is similar to provided except that you have to provide the JAR which contains it explicitly. The artifact is always available and is not looked up in a repository.

**import** (only available in Maven 2.0.9 or later)

This scope is only used on a dependency of type pom in the <dependencyManagement> section. It indicates that the specified POM should be replaced with the dependencies in that POM's <dependencyManagement> section. Since they are replaced, dependencies with a scope of import do not actually participate in limiting the transitivity of a dependency.

**What is Archetype?**

Archetype is a Maven project templating toolkit.

An archetype is defined as an original pattern or model from which all other things of the same kind are made.

**How to create a maven project from command line**

The best way to create a simple maven project from command line

**mvn archetype:generate -DgroupId=com.ddlab.rnd -DartifactId=mvnfromcmd -DinteractiveMode=false**

**mvn archetype:generate -DgroupId=com.ddlab.rnd -DartifactId=mvnfromcmd -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

for web application

**maven-archetype-webapp**

**mvn archetype:generate -DgroupId=com.ddlab.rnd -DartifactId=mvnwebappfromcmd -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false**

**How to add custom jar file/s in the project**

Create a directory called lib and place all the required non-open source jar files. Add the following in the dependency section of your pom.xml

<dependency>

<groupId>my.dummy.groupid</groupId>

<artifactId>temp</artifactId>

<version>1.0.0</version>

**<scope>system</scope>**

**<systemPath>${project.basedir}/lib/temp.jar</systemPath>**

</dependency>

**Questions on Maven**

Q: What does it mean when you say Maven uses Convention over Configuration?

A: Maven uses Convention over Configuration which means developers are not required to create build process themselves. Developers do not have to mention each and every configuration details.

Q. What are the different dependency scopes have you used?

A. compile, provided, test, and import.

Q: What are the aspects Maven manages?

A: Maven provides developers ways to manage following:

**Builds**

**Documentation**

**Reporting**

**Dependencies**

**SCMs**

**Releases**

**Distribution**

**mailing list**

What is current version of Maven - 3.0.4

Q: What is POM?

A: POM stands for Project Object Model. It is fundamental Unit of Work in Maven. It is an XML file. It always resides in the base directory of the project as pom.xml. It contains information about the project and various configuration details used by Maven to build the project(s).

Q: What information does POM contain?

A:POM contains the some of the following configuration information:

**project dependencies**

**plugins**

**goals**

**build profiles**

**project version**

**developers**

**mailing list**

Q: What are the build life cycles in Maven ?

There are three built-in build lifecycles: default, clean and site. The default lifecycle handles your project deployment, the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle has the following build phases (for a complete list of the build phases, refer to the Lifecycle Reference):

**validate** - validate the project is correct and all necessary information is available

**compile** - compile the source code of the project

**test** - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

**package** - take the compiled code and package it in its distributable format, such as a JAR.

**integration-test** - process and deploy the package if necessary into an environment where integration tests can be run

**verify** - run any checks to verify the package is valid and meets quality criteria

**install** - install the package into the local repository, for use as a dependency in other projects locally

**deploy** - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

**Q: What is plugin goal ?**

A plugin goal represents a specific task (finer than a build phase) which contributes to the building and managing of a project. It may be bound to zero or more build phases. The order of execution depends on the order in which the goal(s) and the build phase(s) are invoked. For example, consider the command below. The clean and package arguments are build phases, while the dependency:copy-dependencies is a goal (of a plugin).

mvn clean dependency:copy-dependencies package

**What is Plugin ?**

Plugins : The second way to add goals to phases is to configure plugins in your project. Plugins are artifacts that provide goals to Maven. Furthermore, a plugin may have one or more goals wherein each goal represents a capability of that plugin. For example, the Compiler plugin has two goals: compile and testCompile. The former compiles the source code of your main code, while the latter compiles the source code of your test code.

Q: What is Maven artifact?

A: An artifact is a file, usually a JAR that gets deployed to a Maven repository. A Maven build produces one or more artifacts, such as a compiled JAR and a "sources" JAR.

Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string. The three together uniquely identify the artifact.

Q: What would the command mvn clean do ?

A: This command removes the target directory with all the build data before starting the build process.

Q: What phases does a Clean Lifecycle consist?

A: The clean lifecycle consists of the following phases:

pre-clean

clean

post-clean

Q: What is Build Profile?

A: 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.

Q: Why profile is used in Maven?

A: To give portability to projects ( e.g. windows, linux etc).

Q: What is a Maven Repository?

A: A repository is a place i.e. directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

Q: What types of Maven repository?

A: Maven repositories are of three types: **local, central, remote**

Q: What is local repository?

A: Maven local repository is a folder location on your machine. It gets created when you run any maven command for the first time. Maven local repository keeps your project's all dependencies (library jars, plugin jars etc).

Q: What is the default location for your local repository?

A: ~/m2./repository.

Q: What is Central Repository?

A: It is repository provided by Maven community. It contains a large number of commonly used libraries. When Maven does not find any dependency in local repository, it starts searching in central repository using following URL: http://repo1.maven.org/maven2/.

Q: What is Remote Repository?

A: Sometimes, Maven does not find a mentioned dependency in central repository as well then it stops the build process and output error message to console. To prevent such situation, Maven provides concept of Remote Repository which is developer's own custom repository containing required libraries or other project jars.

Q: What is the sequence in which Maven searches for dependency libraries?

A: Following is the search pattern:

Step 1 - Search dependency in local repository, if not found, move to step 2 else if found then do the further processing.

Step 2 - Search dependency in central repository, if not found and remote repository/repositories is/are mentioned then move to step 4 else if found, then it is downloaded to local repository for future reference.

Step 3 - If a remote repository has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency).

Step 4 - Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference otherwise Maven as expected stop processing and throws error (Unable to find dependency).

Q: Why are Maven Plugins used?

A: Maven Plugins are used to :

create jar file.

create war file.

compile code files.

unit testing of code

create project documentation.

create project reports.

Q: What is SNAPSHOT in Maven?

A: SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for a new SNAPSHOT version in a remote repository for every build.

Q: What is difference between Snapshot and Version?

A: In case of Version, if Maven once downloaded the mentioned version say data-service:1.0, it will never try to download a newer 1.0 available in repository. To download the updated code, data-service version is be upgraded to 1.1.

In case of SNAPSHOT, Maven will automatically fetch the latest SNAPSHOT (data-service:1.0-SNAPSHOT) everytime app-ui team build their project.

Q: What is transitive dependency in Maven?

A: Transitive dependency means to avoid needing to discover and specify the libraries that your own dependencies require, and including them automatically.

Q: How Maven handles and determines what version of dependency will be used when multiple version of an artifact are encountered?

A: Maven determines what version of a dependency is to be used when multiple versions of an artifact are encountered. If two dependency versions are at the same depth in the dependency tree, the first declared dependency will be used. This is called dependency mediation.

Q: What is the minimal set of information for matching a dependency references against a dependencyManagement section ?

A: {groupId,artifactId,type,classifier}.

Q: What is the use of execution element in pom file?

A: The <execution> element contains information's required for the execution of a plugin.

Q: What is a project's fully qualified artifact name?

A: <groupId>:<artifactId>:<version>

Q: How can you build your project offline?

A: Use the command:

mvn o package.

What is SCM ? SCM (Software Configuration Management)

**Q: What is a Mojo?**

**A:** A mojo is a Maven plain Old Java Object. Each mojo is an executable goal in Maven, and a plugin is a distribution of one or more related mojos.