

Java Keywords

- **Java** -- Application Program
- **JDK** -- Java Development Kit and it is Software Bundle
- **JRE** -- Java Runtime Environment which is having set of Lib and Bin
- **JVM** -- Java Virtual Machine to execute Java Program

32

32

Application Build:

- Java ->
 - Source File - FileName.java
 - Binary File - FileName.class

Source code is managed in 'src' directory

Binary code is managed in 'build/classes or release or target' directory
- Converting your source code into binary
 - Compilation:
 - --> Source Code --> Compiler --> Binary Code
- Binaries:- Executable, artifacts, outcomes

33










33

Web Project Build

- Two Applications:
 - 1) Standard Alone - Stand Alone Machine (Using Jdk)
 - Contains Java Classes
 - The packaging file format is .jar
 - 2) Web Apps - Hosted on App Server
- Web Application: Binary -> .war
 - src/ .java
 - build/classes/ .class
 - WebContent/ .css
 - .js
 - .html
 - .jsp
 - .xml
 - images

34

Web Application Structure

Directory or File	Description
 MyWebApp	Public document root of Web application
 WEB-INF	Private resources not served directly to clients
 classes	Classes, such as servlets, filters, listeners
 lib	Java libraries (JAR files)
 web.xml	Optional Java EE deployment descriptor
 weblogic.xml	Optional WebLogic deployment descriptor
 index.html	Static and dynamic Web content
 page1.jsp	
 page2.jsp	

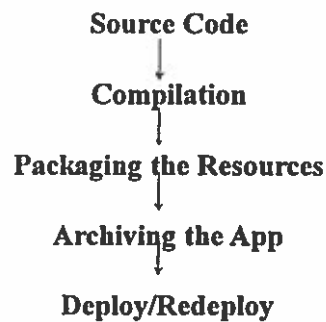


MyWebApp.war

35

35

Deployment Process



36

36

What is a Build Tool?

- A build tool is a tool that automates everything related to building the software project and it typically includes one or more of these activities:
- Generating source code (if auto-generated code is used in the project).
- Generating documentation from the source code.
- Compiling source code.
- Packaging compiled code into JAR files or ZIP files.
- Installing the packaged code on a server, in a repository or somewhere else.

37

37

Database Design?

RISE 'N' SHINE TECHNOLOGIES

111

111

Table Name - Employee

Columns

Rows

Emp ID	Name	Department	Salary
1	x	DevOps	5Lakhs
2	y	AWS	4Lakhs
3	z	DevOps & AWS	6 Lakhs

RISE 'N' SHINE TECHNOLOGIES

112

112

Table Name - Address

Emp ID	Line 1	Line 2	City	State
1	11	L2	Hyd	TS
2	11	L2	Hyd	TS
3	11	L2	Hyd	TS

RISE 'N' SHINE TECHNOLOGIES

113

113

What is XML?

➤ A meta language that allows you to create and format your own document markups

➤ These files are

- easy to read
- unambiguous
- extensible
- platform-independent

➤ File Extension is

- .xml

RISE 'N' SHINE TECHNOLOGIES

114

114

XML Key Words:

- Attribute
 - Name
 - Value
- Element
 - can contain text, other elements or a combination
- Root Element

RISE 'N' SHINE TECHNOLOGIES

115

115

A well-formed XML document

➤ **Elements** have an open and close tag, unless it is an empty element

```
<tag> text </tag>
```

➤ if a tag is an **empty element**, it has a closing / before the end of the tag

```
<tag/>
```

➤ open and close tags are nested correctly

RISE 'N' SHINE TECHNOLOGIES

116

116

XML basics

the XML declaration

```
<?xml ?>
```

-not required, but typically used

-attributes include:

- version
- encoding: the character encoding used in the document

```
<?xml version="1.0" encoding="UTF-8">
```

RISE 'N' SHINE TECHNOLOGIES

117

117

XML basics

```
<!-- --> comments
```

- contents are ignored by the processor
- cannot come before the XML declaration
- cannot appear inside an element tag
- may not include double hyphens

RISE 'N' SHINE TECHNOLOGIES

118

118

Sample XML File:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<Employees>
  <Employee>
    <id>1</id>
    <name>x</name>
    <dept>DevOps</dept>
    <sal>5lakhs</sal>
    <Address>
      <line1>l1</line>
      <line2/>
      <city>hyd</city>
      <state>ts</state>
    </Address>
  </Employee>
```

RISE 'N' SHINE TECHNOLOGIES

119

119

Sample XML File: Cont...

```
<Employee>
  <id>2</id>
  <name>y</name>
  <dept>AWS</dept>
  <sal>5Lakhs</sal>
  <Address>
    <line1>l1</line>
    <line2>l2</line2>
    <city>hyd</city>
    <state>ts</state>
  </Address>
</Employee>
```

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

```
</Employees>
```

RISE 'N' SHINE TECHNOLOGIES

120

120

Maven

44

44

What is Maven?

- It is a powerful build tool for Java software projects.
- Maven is developed in Java
- The Maven can be downloaded from:
- <http://maven.apache.org>

45

45

What is Maven?

- Maven provides developers a complete build lifecycle framework.
- Maven simplifies and standardizes the project build process.
- It handles compilation, distribution, documentation, team collaboration and other tasks seamlessly.
- Maven increases reusability and takes care of most of build related tasks.

46

46

Maven Environment Setup

- System Requirement
 - JDK 1.5 or above.
 - Memory no minimum requirement.
 - Disk Space no minimum requirement.
 - Operating System no minimum requirement.

47

47

Java installation

Step 1: Set JAVA environment

- Set the JAVA_HOME environment variable to point to the base directory location where Java is installed.

Step 2 - verify Java installation

- Windows Open Command Console `c:\> java -version`
- Linux Open Command Terminal `$ java -version`

48

48

Maven Installation

- **Step 3: Download Maven archive**

Download Maven 3.3.3 from

-- <http://maven.apache.org/download.cgi>

- **OS Archive name**

- Windows `apache-maven-3.3.3-bin.zip`
- Linux `apache-maven-3.3.3-bin.tar.gz`

- **Step 4: Extract the Maven archive**

- Extract the archive, to the directory you wish to install Maven 3.3.3. The subdirectory `apachemaven-3.3.3` will be created from the archive.

49

49

Maven Installation

- **Step 5: Set Maven environment variables**
- Add M2_HOME, MAVEN_OPTS to environment variables.
- Windows: Set the environment variables using system properties.
 - *M2_HOME=... \apache-maven-3.3.3*
 - *MAVEN_OPTS=-Xms256m -Xmx512m*
- Linux: Open command terminal and set environment variables.
 - *export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.3*
 - *export MAVEN_OPTS=-Xms256m -Xmx512m*

50

50

Maven Installation

Step 6: Add Maven bin directory location to system path

- Now append M2 variable to System Path
- Windows
 - Append the string ;%M2% to the end of the system variable, Path.
- Linux
 - *export PATH=\$M2_HOME/bin:\$PATH*

51

51

Maven Installation

- **Step 8: Verify Maven installation**
- Now open console, execute the following **mvn** command.
- Windows:
 - Open Command Console `c:\> mvn --version`
- Linux:
 - Open Command Terminal `$ mvn --version`

52

52

Maven Core Concepts – POM File

- Maven is centered around the concept of POM files (Project Object Model).
- A POM file is an XML representation of project resources like source code, test code, dependencies (external JARs used) etc.
- It contains a detailed description of your project, including information about versioning and configuration management, dependencies, application and testing resources, team members and structure, and much more.
- The POM file should be located in the root directory of the project it belongs to.

53

53

Maven POM Files

- All POM files require the **project** element and three mandatory fields: **groupId**, **artifactId**, **version**.
- Projects notation in repository is **groupId:artifactId:version**.
- Root element of POM.xml is **project** and it has three major sub-nodes :

54

Example POM

```
<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> org.rns.training</groupId>
  <artifactId>maven-training</artifactId>
  <version>1.0</version>

</project>
```

55

Elements of POM:

- **groupId:**
- This is an Id of project's group. This is generally unique amongst an organization or a project.
 - For example, a banking group `com.company.bank` has all bank related projects.
- **artifactId:**
- This is an Id of the project. This is generally name of the project.
 - For example, `consumer-banking`. Along with the `groupId`, the `artifactId` defines the artifact's location within the repository.
- **Version:**
- This is the version of the project. Along with the `groupId`, It is used within an artifact's repository to separate versions from each other.

56

56

Elements of POM:

- For example:
 `com.company.bank:consumer-banking:1.0`
 `com.company.bank:consumer-banking:1.1.`
- **name** - This is the human-readable name of the project.
- **url** - This is the url for the project website.
- **packaging** - This defines the “style” of project your building (e.g. ear, jar, war, etc.).
 - For more information on packaging, check out the respective plugins (maven-jar-plugin, maven-war-plugin, etc.)

57

57

Elements of POM

- **dependencies** - Specifies the dependencies of the project. This will be materialized from any defined maven repositories. More details later
- **repositories** - Specifies alternative locations for maven to look when materializing dependencies.
- **pluginRepositories** - Specifies alternative location for maven to look when materializing build plugins
- **build** - Specifies configuration on **how** to build the project
- **reports** - Specifies configuration on **what** reports to generate for the project.

58

58

Maven - Project Templates

- Maven provides a very large list of different types of project templates using concept of **Archetype**.
- Maven helps to quickly start a new java project using following command
 - `mvn archetype:generate`
- What is Archetype?
 - Archetype is a Maven plug-in whose task is to create a project structure as per its template.

We are going to use *quickstart* archetype plug-in to create a simple java application here.

59

59

Maven - Creating Project

- To create a simple java application, we'll use *maven-archetype-quickstart* plugin. In example below, We'll create a maven based java application project.

```

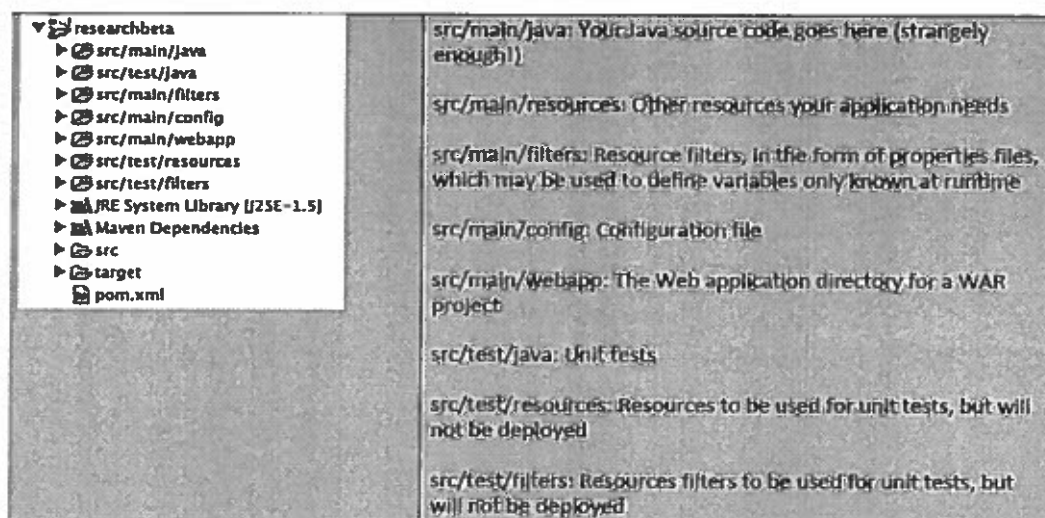
-mvn archetype:generate -DgroupId=com.rns.app -
-DartifactId=maven-app -DarchetypeArtifactId=maven-
archetype-quickstart -DinteractiveMode=false

```

60

60

Default Directory Layout



61

61

Maven - Build & Test Project

- Let's open command console, go the project directory and execute the following **mvn** command.

> mvn clean package

- Now open command console, go the project\target\classes directory and execute the following java command.

> java com.rns.app.App

62

62

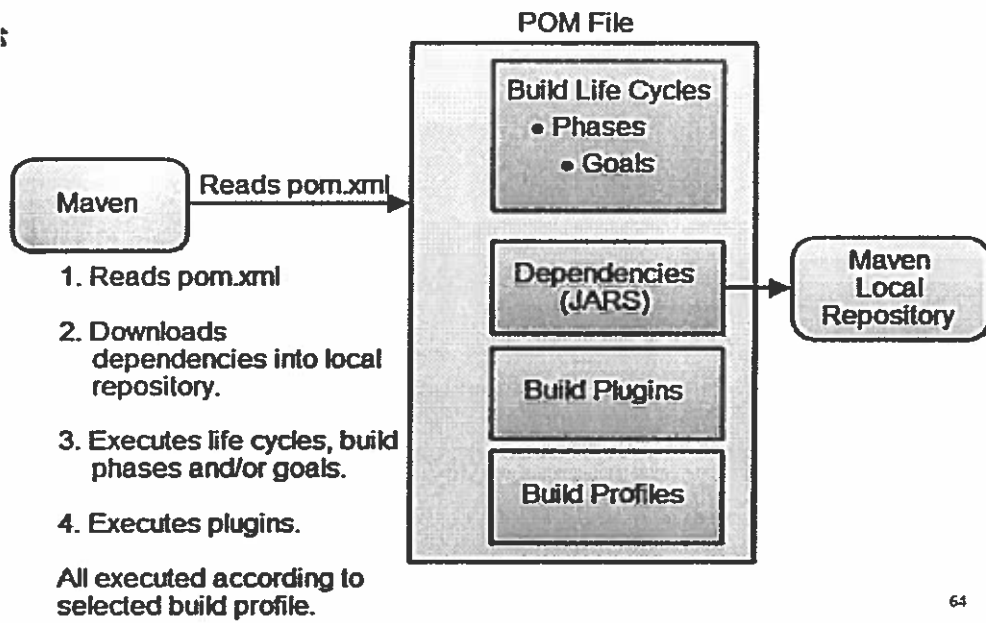
key learning concepts

- We give maven two goals, first to clean the target directory (clean) and then package the project build output as jar(package).
- Packaged jar is available in artifact_id\target folder as artifact_id-1.0.jar.
- Test reports are available in artifact_id\target\surefire-reports folder.
- Maven compiled source code file(s) and then test source code file(s).
- Then Maven run the test cases.
- Finally Maven created the package.

63

63

Maven Core Concepts – POM File



64

Maven core concepts - Build Life Cycles, Phases and Goals

- The build process in Maven is split up into build life cycles, phases and goals.
- A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals.
- Ex: clean, compile, testcompile, test, package, install, deploy
- When you run Maven you pass a command to Maven. This command is the name of a build life cycle, phase or goal.

65

65

Build Life Cycles

- Maven has 3 built-in build life cycles. These are:
 - 1. default
 - 2. clean
 - 3. site
- Each of these build life cycles are executed independently of each other.
- You can get Maven to execute more than one build life cycle, but they will be executed in sequence, separately from each other, as if you had executed two separate Maven commands.

Ex : mvn clean package site

66

66

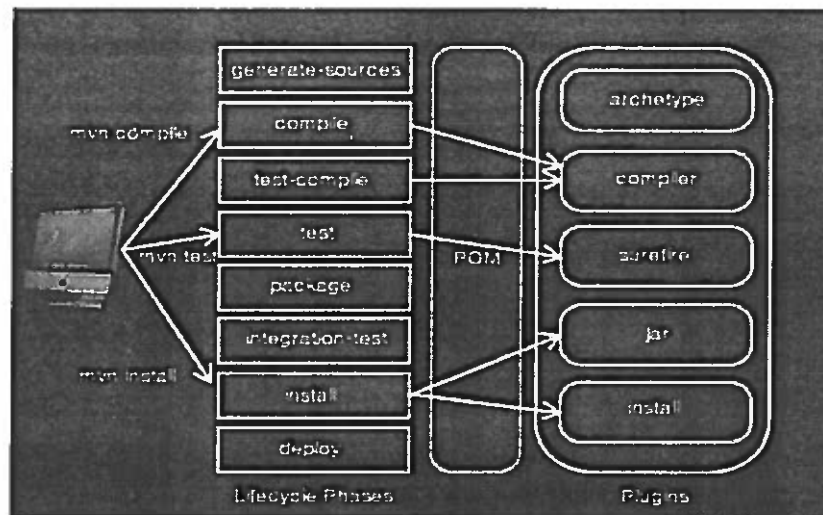
Build Life Cycles

- The **default** life cycle handles everything related to compiling and packaging your project.
- The **clean** life cycle handles everything related to removing temporary files from the output directory, including generated source files, compiled classes, previous JAR files etc.
- The **site** life cycle handles everything related to generating documentation for your project. In fact, site can generate a complete website with documentation for your project.

67

67

Maven Lifecycle



68

68

Maven lifecycle phases

Some of the more useful Maven lifecycle phases are the following:

- **generate-sources:** Generates any extra source code needed for the application, which is generally accomplished using the appropriate plug-ins
- **compile:** Compiles the project source code
- **test-compile:** Compiles the project unit tests
- **test:** Runs the unit tests (typically using JUnit) in the `src/test` directory
- **package:** Packages the compiled code in its distributable format (JAR, WAR, etc.)
- **integration-test:** Processes and deploys the package if necessary into an environment where integration tests can be run
- **install:** Installs the package into the local repository for use as a dependency in other projects on your local machine
- **deploy:** Done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and project.

69

Running Maven

- You can execute multiple build life cycles or phases by passing more than one argument to the mvn command. Here is an example:
 - mvn clean install
- This command first executes the clean build life cycle, which removes compiled classes from the Maven output directory, and then it executes the install build phase.

70

70

Demo

- How to generate Java Docs of the project?
 - mvn clean site
- How to run Junit Test Cases?
 - mvn clean test
- How to generate the Junit Test Case Reports?
 - mvn clean test site

71

71

Maven Settings File

- In the settings file you can configure settings for Maven across all Maven POM files. For instance, you can configure:
 - Location of local repository
 - Active build profile, etc....
- The settings files are called settings.xml and it is located at:
- The Maven installation directory: \$M2_HOME/conf/settings.xml

72

72

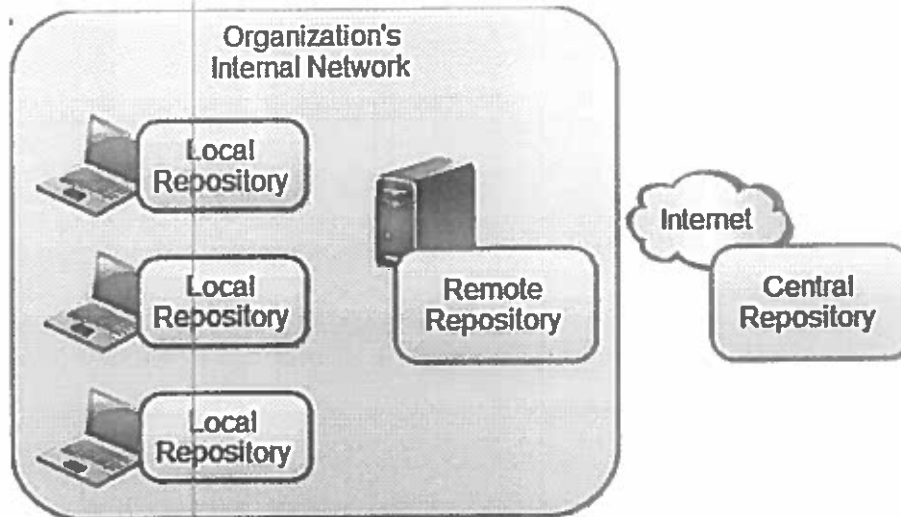
Maven - Repositories

- What is a Maven Repository?
- In Maven terminology, 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.
- Maven repository are of three types
 - local
 - central
 - remote

73

73

Maven - Repositories



74

74

Local Repository

- Maven local repository keeps your project's all dependencies (library jars, plugin jars etc).
- When you run a Maven build, then Maven automatically downloads all the dependency jars into the local repository. It helps to avoid references to dependencies stored on remote machine every time a project is build.
- Maven local repository by default get created by Maven in %USER_HOME% directory.
- To override the default location, mention another path in Maven settings.xml file available at %M2_HOME%\conf directory.

75

75

Local Repository

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <localRepository>C:/MyLocalRepository</localRepository>

</settings>
```

76

76

Central Repository

- Maven central repository 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/>
- Key concepts of Central repository
 - This repository is managed by Maven community.
 - It is not required to be configured.
 - It requires internet access to be searched.
- To browse the content of central maven repository URL: <http://search.maven.org/#browse>

77

77

Remote Repository

- Sometime, Maven does not find a mentioned dependency in central repository as well then it stopped 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.

78

Remote Repository

- ```
<dependencies>
 <dependency>
 <groupId>com.companyname.common-lib</groupId>
 <artifactId>common-lib</artifactId>
 <version>1.0.0</version>
 </dependency>
</dependencies>
```
- ```
<repositories>
  <repository> <id>companyname.lib1</id>
    <url>http://download.companyname.org/maven2/lib1</url>
  </repository>
  <repository> <id>companyname.lib2</id>
    <url>http://download.companyname.org/maven2/lib2</url>
  </repository>
</repositories>
```

79

Maven Dependency Search Sequence

- **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).

80

Maven - Snapshots

- A large software application generally consists of multiple modules and it is common scenario where multiple teams are working on different modules of same application.
- For example consider a team is working on the front end of the application as app-ui project (app-ui.jar:1.0) and they are using data-service project (data-service.jar:1.0).
- Now it may happen that team working on data-service is undergoing bug fixing or enhancements at rapid pace and the they are releasing the library to remote repository almost every other day.

81

81

Maven - Snapshots

- Now if data-service team uploads a new version every other day then following problem will arise
 - data-service team should tell app-ui team every time when they have released an updated code.
 - app-ui team required to update their pom.xml regularly to get the updated version
- To handle such kind of situation, **SNAPSHOT** concept comes into play.

82

82

What is SNAPSHOT?

- **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.
- Now data-service team will release **SNAPSHOT** of its updated code everytime to repository say data-service:1.0-SNAPSHOT replacing a older **SNAPSHOT** jar.

83

83

Snapshot vs Version

- 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) every time app-ui team build their project.

84

84

app-ui pom.xml

- app-ui project is using 1.0-SNAPSHOT of data-service

```
<dependencies>
  <dependency>
    <groupId>data-service</groupId>
    <artifactId>data-service</artifactId>
    <version>1.0-SNAPSHOT</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

85

85

data-service pom.xml

- Data-service project is releasing 1.0-SNAPSHOT for every minor change

```
<groupId>data-service</groupId>
```

```
<artifactId>data-service</artifactId>
```

```
<version>1.0-SNAPSHOT</version> <packaging>jar</packaging>
```

- Although, In case of SNAPSHOT, Maven automatically fetches the latest SNAPSHOT on daily basis. You can force maven to download latest snapshot build using -U switch to any maven command.

```
- > mvn clean package -U
```

86

86

Maven - Web Application

- how to manage a web based project using version control system **Maven**.
- Here you will learn how to create/build/deploy and run a web application:

87

87

Create Web Application

- To create a simple java web application, we'll use maven-archetype-webapp plugin. So let's open command console, and execute the following **mvn** command.

```
>mvn archetype:generate -DgroupId=com.rns.simpleweb -DartifactId=webapp -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

- Maven will start processing and will create the complete web based java application project structure.

88

88

Create Web Application

- Maven uses a standard directory layout. Using above example, we can understand following key concepts.
- Webapp contains src folder and pom.xml
 - src/main/webapp contains index.jsp and WEB-INF folder
 - src/main/webapp/WEB-INF contains web.xml
 - src/main/resources it contains images/properties files .

89

89

Build Web Application

- Let's open command console, go the project directory and execute the following **mvn** command.
 - **>mvn clean package**
- Deploy Web Application
 - Now copy the **webapp.war** created in **C:\ > MVN > webapp > target >** folder to your webserver webapp directory and restart the webserver.
- Test Web Application
 - Run the web-application using URL : **http://<server-name>:<port-number>/webapp/index.jsp**
- Verify the output.

90