Maven 3.x

Krishantha Dinesh
[kdinesh@virtusa.com]
@ 2013-Nov

2000 West Park Drive
Westborough MA 01581 USA
Phone: 508 389 7300 Fax: 508 366 9901

virtusa®
Accelerating Business Outcomes

Objectives – Learn about

- Not to make expert of Maven in over night.. But make a engineers who can get the start of art technology benefits

virtusa

Prerequisites

- Mobilephone.soundmode=soundmode.completesilent && soundmode.completesilent != soundmode.vibrate

virtusa

What's going to Discuss

- Introduction about Maven
- Main concepts of maven
  - Conception over configuration
- Daily usage of Maven
- Integration with IDE eclipse

virtusa

What is Maven

- Simple surname for Maven is build tool
  - Provide artifacts
  - Resolve dependencies
- Maven has a capacity of project management
  - Can be use for versioning
  - Can produce the java docs

virtusa

Who is the parent or owner for Maven

- It free and open source
- Manage by apache software foundation

virtusa

Why we need maven

- Reputable build make easy
- Download dependency of dependency
- Can maintain with local repository. (download once use any time)
- Nice integration with IDE

virtusa

Ant vs Maven

- ANT
  - Very procedural tool
  - Build on top of xml and java and it make it to be cross platform
  - It is not a comprehensive build tool
  - We need to define every single step
- MAVEN
  - Maven is build tool over scripting tool
  - Lot of build in functionality
  - Consistency across the projects
  - Version control build

virtusa

Install Maven

- Download maven from following url
- http://maven.apache.org/download.cgi
- Setup environment variable
- JAVA_HOME → Directory that you installed jdk not include bin directory
- MAVEN_HOME → directory that you extracted Maven not include bin
- Update PATH variable, at end of it
- ;%JAVA_HOME%\bin;%MAVEN_HOME%\bin

virtusa

Verify the installation

- Go to command prompt end enter following command
  - mvn –version

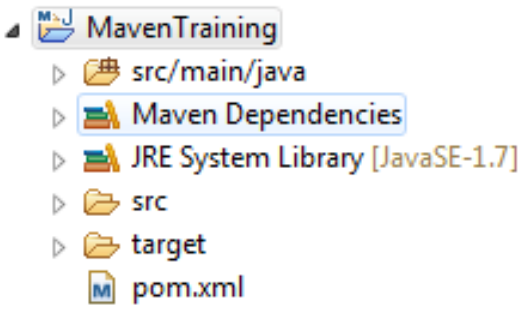
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\kdinesh>mvn -version
Apache Maven 3.1.1 (0728685237757ffbf44136acec0402957f723d9a; 2013-09-17 20:52:2
2+0530)
Maven home: C:\Krishantha\developments\java\frameworks\apache-maven-3.1.1\bin\..

Java version: 1.7.0_40, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.7.0_40\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"
C:\Users\kdinesh>
```

Lets create a project to demonstrate maven

- Open spring STS or eclipse
- Create new project [File→ New → Project
- Create new files pom.xml in side project directory [Project Object Model]
- Create directory structure as src\main\java (this is required by maven – convention over configuration)
- Add following for pom.xml file



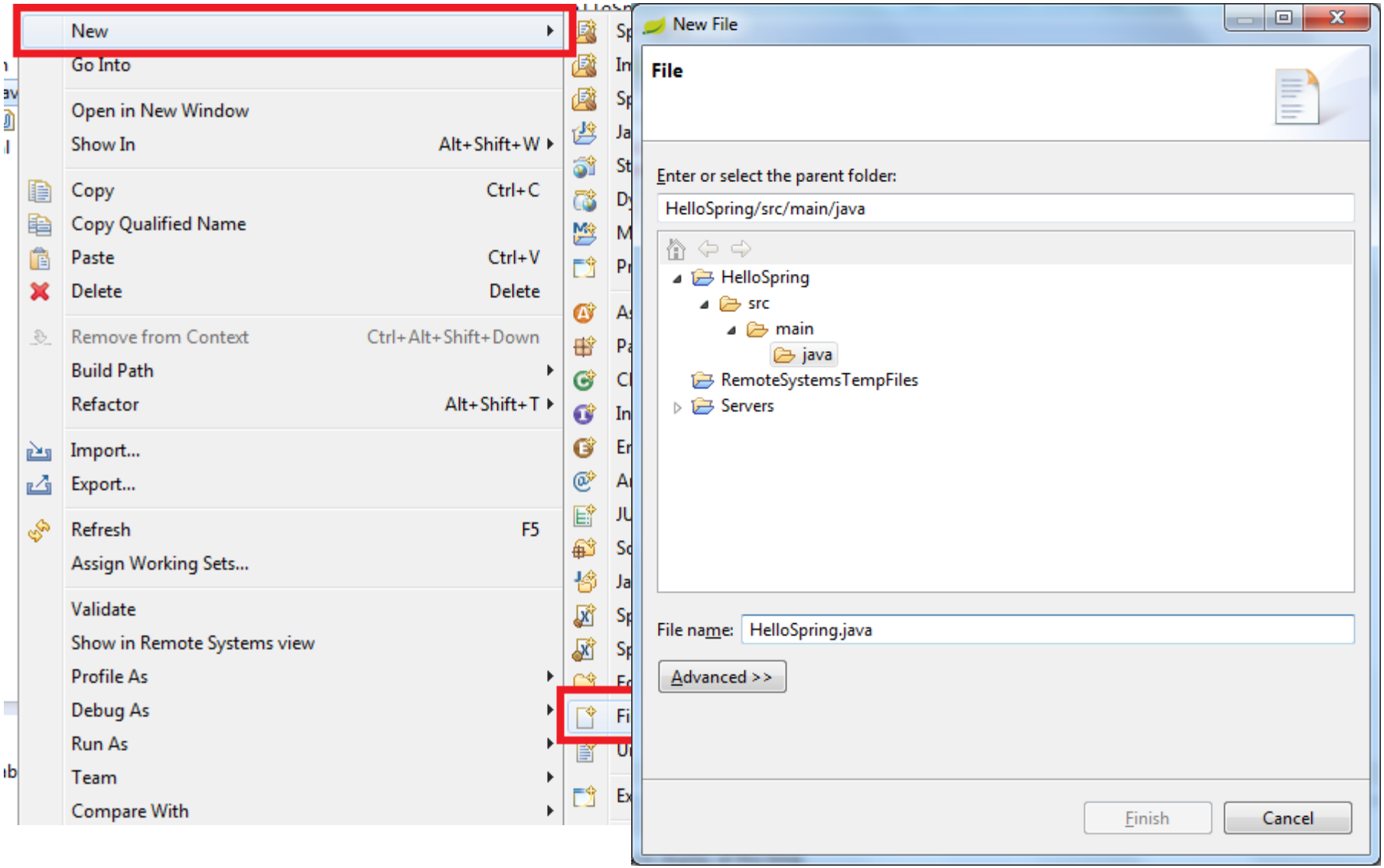

```
xsi:schemaLocation="http://maven.apache
<groupId>com.virtusa</groupId>
<artifactId>HelloSpringApp</artifactId>
<version>1.0-SNAPSHOT</version>
<modelVersion>4.0.0</modelVersion>
<packaging>jar</packaging>
```

What those mean

- Group id → similar to company (com.virtusa or things like that)
- Artifact id → application name
- Version → what is the version going to build
- Model version → xml version which we going to process
- Packaging → output format (jar)

virtusa

Create Java class



First class compile through Maven

- Program will look like

```
public class HelloSpring{
    public static void main (String args[]){
        System.out.println("Hello.. spring you look nice :) ")
    }
}
```

- Now go to command prompt and navigate to workspace

```
C:\Windows\system32\cmd.exe

c:\Krishantha\developments\java\ws-sts\HelloSpring>dir
 Volume in drive C is Windows
 Volume Serial Number is 84B4-2A63

 Directory of c:\Krishantha\developments\java\ws-sts\HelloSpring

11/06/2013  12:19 AM    <DIR>          .
11/06/2013  12:19 AM    <DIR>          ..
11/06/2013  12:12 AM               215 .project
11/06/2013  12:19 AM               212 pom.xml
11/06/2013  12:19 AM    <DIR>          src
               2 File(s)            427 bytes
               3 Dir(s)  441,459,298,304 bytes free

c:\Krishantha\developments\java\ws-sts\HelloSpring>
```

Lets do it

- mvn clean

- Above command will be download few jar files and other plugins it need
  - mvn complie

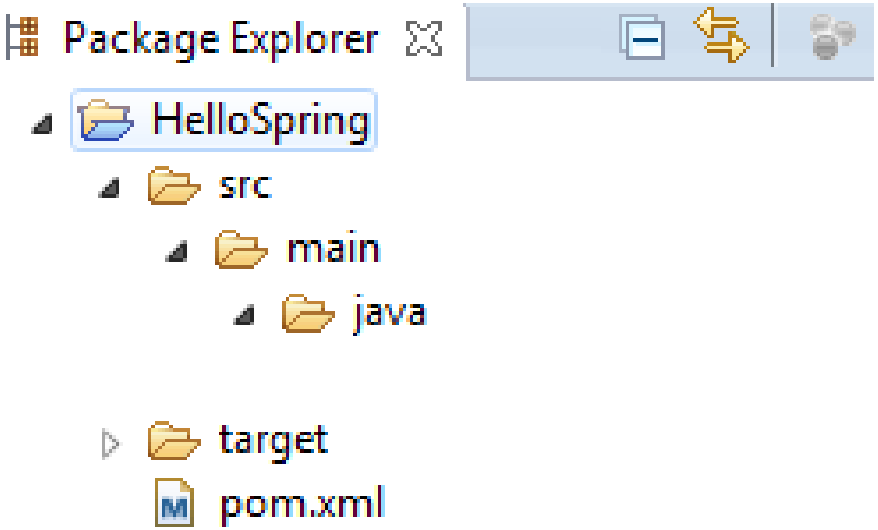
- It will compile your program and now should have target directory
  - mvn package

- It will package your program as jar file as expected in pom.xml

virtusa

Structure

- By default it need src/main/java directories
- Src/test/java is for hold the testing code
- Compile source code to target directory
- Run according to the pom.xml file



Pom.xml

- There are major 4 categories for pom.xml
- Project information
  - Groupid
  - Artifactid
  - Version
  - Packaging
- Dependencies
  - Mention about the project dependencies
- Build
  - Directory structure
- Repository
  - From where we can download

virtusa

Dependencies

- It is a registry of required dependent libraries
- It s need to have 3 minimum parameters to work
  - Groupid
  - Artifactid
  - Version

```
<project>
    <groupId>com.virtusa.training.spring</groupId>
    <artifactId>HelloSpringApp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>

    <dependencies>
    <dependency>
    <groupId>commons-lang</groupId>
    <artifactId>commons-lang</artifactId>
    <version>2.1</version>
    </dependency>
    </dependencies>
</project>
```

virtusa

What commands mean

- Clean
  - Delete target directories and generated resource
- Compile
  - Compile source code and copy required resources
- Package
  - Run compile , run unit test , package app based on pom.xml
- Install
  - Package command + install in local repo
- Deploy
  - Install command + install corporate repository

virtusa

Local repository

- Default maven repository is you home directory and .m2
- For previous configuration it will store in `C:\Users\kdinesh\.m2\repository\commons-lang\commons-lang\2.1`
- It will help to avoid the duplication of lib files over SCM and file system.

virtusa

Default override

- There is a way to override this defaults of the maven
- Build section can do the job
- You have evidence that how maven going to build the final artifact name
- Lets see how we can override that method
- Change the code as follows and run mvn clean package

virtusa

Override the defaults

```
<project>
    <groupId>com.virtusa.training.spring</groupId>
    <artifactId>HelloSpringApp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>

    <dependencies>
        <dependency>
            <groupId>commons-lang</groupId>
            <artifactId>commons-lang</artifactId>
            <version>2.1</version>
        </dependency>
    </dependencies>
    <build>
        <finalName>HelloSpringApplication</finalName>
    </build>
</project>
```

C:\Windows\system32\cmd.exe

```
elloSpringApplication.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.970s
[INFO] Finished at: Wed Nov 06 12:35:27 IST 2013
[INFO] Final Memory: 11M/120M
[INFO] ------------------------------------------------------------------------
c:\Krishantha\developments\java\ws-sts\HelloSpring>cd target

c:\Krishantha\developments\java\ws-sts\HelloSpring\target>dir
 Volume in drive C is Windows
 Volume Serial Number is 84B4-2A63

 Directory of c:\Krishantha\developments\java\ws-sts\HelloSpring\target

11/06/2013  12:35 PM    <DIR>          .
11/06/2013  12:35 PM    <DIR>          ..
11/06/2013  12:35 PM    <DIR>          classes
11/06/2013  12:35 PM             2,018 HelloSpringApplication.jar
11/06/2013  12:35 PM    <DIR>          maven-archiver
               1 File(s)          2,018 bytes
               4 Dir(s)  441,442,684,928 bytes free

c:\Krishantha\developments\java\ws-sts\HelloSpring\target>
```

Version

- Version can be anything and no hard rules
- However SNAPSHOT is a specific version over releases.
- Changes always downloaded before compile the code
- Save you by sending development version for production
- SNAPSHOT must be in capital letter to server the purpose

virtusa

Transitive dependencies

- This is the reason to Maven become a super star
- If we refer one dependency its automatically pulled all relevant dependencies.
- This is very useful as creator only know what are compatible of.

virtusa

Scopes

- There are 6 scopes available for define
- Compile → default scope. Artifacts ships with app
- Provided → artifact will provided by container
- Runtime → no need for compile. Need to execute Dynamic library. (like jdbc)
- Test → only need for test execution
- System → DO NOT USE ☹ hard code file path to your file system
- Import → dealing with dependency management.

virtusa

Demo configuration

```
<project>
    <groupId>com.virtusa.training.spring</groupId>
    <artifactId>HelloSpringApp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>
    <repositories>
        <repository>
            <id>jboss-public-repository-group</id>
            <name>JBoss Public Repository Group</name>
            <url>http://repository.jboss.org/nexus/content/groups/public</url>
        </repository>
    </repositories>
    <dependencies>
        <dependency>
            <groupId>commons-lang</groupId>
            <artifactId>commons-lang</artifactId>
            <version>2.1</version>
        </dependency>

        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>4.1.6.Final</version>
        </dependency>

    </dependencies>
```

new report for hibernate

scope demo

Transitive dependencies demo

- commons-lang : 2.1 [compile]
- junit : 4.10 [test]
  - hamcrest-core : 1.1 [test]
- hibernate-core : 4.1.6.Final [compile]
  - antlr : 2.7.7 [compile]
  - jboss-logging : 3.1.0.GA [compile]
  - jboss-transaction-api_1.1_spec : 1.0.0.Final [compile
  - dom4j : 1.6.1 [compile]
  - hibernate-jpa-2.0-api : 1.0.1.Final [compile]
  - javassist : 3.15.0-GA [compile]
  - hibernate-commons-annotations : 4.0.1.Final [comp
    - jboss-logging : 3.1.0.CR2 (omitted for conflict w

virtusa

Repositories

- There are two types of repositories
  - Dependency repository
  - Plugin repository
- Local Repository
  - This is the place where maven looked first. If not available it download form maven repo.
- Remote repository
  - Simple storage which has http access

virtusa

Phases

- Validate
  - Validate the project requirement and information
- Compile
  - Compile the source code
- Test
  - Test the compiled code
- Package
  - Packaging based on pom
- Integration-test (from M3)
  - Run integration test

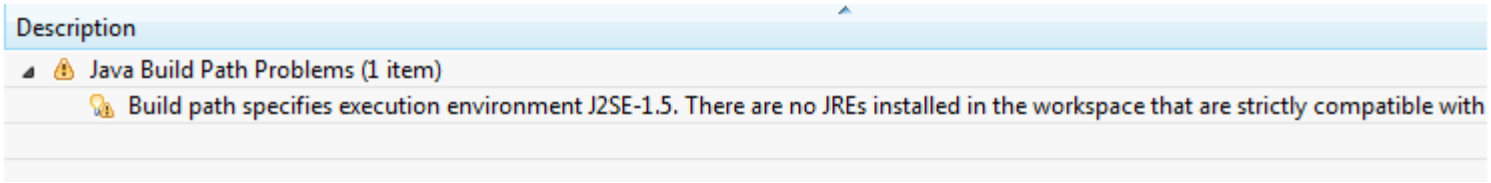
virtusa

- Verify
  - Run integrity checks
- Install
  - Install package on local repo
- Deploy
  - Copy package for report repo

virtusa

Compiler plugin

- Use to compile both test and source code
- Invoke javac with the class path setup from dependencies
- Default is 1.5 regardless to the installed jdk
- You can use configuration section to override this setting



resolved

```
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.1</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

Console | Markers | Progress

0 items

Description

Jar plugin

- Tight to package phase
- Convert application/package in to jar
- Configuration allows to
- Include/exclude

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>2.4</version>
    <configuration>
        <useDefaultManifestFile>true</useDefaultManifestFile>
    </configuration>
</plugin>
```

virtusa

Java doc plugin

- Use for attach java doc to jar

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>2.9</version>
    <executions>
        <execution>
            <id>attach-javadoc</id>
            <phase>verify</phase>
            <goals>
                <goal>jar</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <useDefaultManifestFile>true</useDefaultManifestFile>
    </configuration>
</plugin>
```

override the phase

now this will not execute on package. execute only on install

Happy Automation ☺

virtusa