Maven Properties

=> we can create custom properties in pom.xml with certain values in order to reuse those values through out pom.xml file.. especially very useful towords changing version of the multiple jar files from single point

problem code

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support --> <dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context-support</artifactId>
<version> 6.0.11
</dependency>
</version>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version> 6.0.11
</dependency>
</version>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-orm</artifactId>
<version> 6.0.11
</dependency>
</version>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc --> <dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
```

To modify version of the

spring api, we need to modify

in four places

```
6.0.11
<version>
</dependency>
</version>
```

Solution using properties

In pom.xml

```
<properties>
```

```xml
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>17</maven.compiler.source>
<maven.compiler.target>17</maven.compiler.target>
<sp.version>6.0.8</sp.version>
</properties>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.11</version>
```

Here we need to modify the spring api version only in one place that <sp.version> tag and that will reflect in all the places.

```xml
<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support --> <dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context-support</artifactId>
<version>${sp.version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>${sp.version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-orm</artifactId>
<version>${sp.version}</version>
</dependency>
```

${....}

---> property place holder

Maven scopes for dependencies (jar files)

=> maven dependency scopes allows the maven to use the jar file at different levels

compile (default)

of the Project Buid Process

runtime

test

provided

import

system

compile ::=> it is default scope, if no scope is specified

=> This scope makes the jar files visible right from compilation that to in all build phases

from

(jar is made visible right compilation to till the end of build process)

of the Project

<dependency>

<groupId>commons-lang</groupId>

<artifactId>commons-lang</artifactId>

"compile" most regulary used scope for ..isnost maven dependencies (jar files)

<version>2.6</version>

<scope>compile </scope> </dependency>

runtime :: The specified jar file is visible only at run time (only at the execution of the App)

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.28</version>

<scope>runtime</scope>

</dependency>

=>MySQL JDBC driver s/w or any other JDBC driver s/w

is actually required only during runtime of the app becoz the jdbc driver class loading, activation of jdbc driver s/w, connection establishment with Db s/w and etc.. takes place only at the runtime

provided :: The specified jar file will be visible in all the build phases but the underlying JDK or

Container should supply the jar file

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

The servlet api jar file should be supplied by the underlying server

in which the current web application is executing

</dependency>

system:: same as "provided" scope, but the developer must specify the location of jar file in the System/computer

<dependency>

<groupId>com.baeldung</groupId>

<artifactId>custom-dependency</artifactId>

<version>1.3.2</version>

<scope>system</scope>

<systemPath>${project.basedir}/libs/custom-dependency-1.3.2.jar</systemPath>

</dependency>

name and location of the jar file

test

import ::

(fixed property

giving name and location of the project)

This scope jar file comes to maven project from the specified parent maven project.

<dependency>

<groupId>com.baeldung</groupId> <artifactId>custom-project</artifactId>

<version>1.3.2</version>

<type>pom</type>

For this the current maven project should be linked

:: The "test" scope jar file is visible and accessible right from junit based "test" phase to end of build process

with another maven project which is called parent project To specify parent project details in current project we need to use <parent> tag in the pom.xml (maven inheritance is used) <!--The identifier of the parent POM--> <parent>

<dependency>

<groupId>org.junit.jupiter</groupId> <artifactId>junit-jupiter-params</artifactId>

<scope>test</scope> </dependency>

note:: Most of the times we prefer working with "compile" scope with out specifying the scope name becoz the default scope is "compile" and it makes the jar file visible through out build process right from "compilation" to "execution" to packing

<scope>import</scope>

</dependency>

(Refer bottom of the page for maven Inheritence

Example)

Excluding the dependencies

=============================

<groupId>com.howtodoinjava.demo</groupId>

<artifactId>MavenExamples</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

=> After adding set of dependencies(jar files) to maven project through pom.xml file, if want to exclude one of the dependent jar file with out distrubing the main jar file and its other dependent jar file then

use maven dependency exclusion option

=>we generally try to execlude one or other dependent jar file of the main jar file becoz that jar file is not used in the project

So we want to save the memory or that jar file might be giving clash to other main jar files or dependent jar files

execluding spring-aop-<ver>.jar from the list of dependent jar files came for the spring-context-support-<ver>.jar file

a) add spring context-support jar file info in pom.xml

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support --> <dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${sp.version}</version>

</dependency>

b) execlude spring-aop-<ver>.jar file from the dependency hierarchy

Go to "Dependency Hierarchy tab" of pom.xml file ---->

junit: 4.11 [test]

hamcrest-core: 1.3 [test]

spring-context-support: 6.0.8 [compile]

spring-beans: 6.0.8 [compile]

spring-core: 6.0.8 [compile]

spring-context: 6.0.8 [compile]

spring-aop: 6.0.8 [compile]

spring-beans: 6.0.8 [compile]

spring-core: 6.0.8 [compile]

spring-beans: 6.0.8 [compile]

spring-core: 6.0.8 [compile]

Ospring-expression: 6.0.8 [compile]

spring-core: 6.0.8 [compile]

spring-core: 6.0.8 [compile]

spring-jcl: 6.0.8 [compile]

select spring-aop jar --->right click --->exclude maven artifact --->

...

(or) Expand maven dependencies from the Eclipse Project ---> select jar file to exclude ---> right click on the jar file ---> maven --->exclude maven artifact ---> ... ... --> ....

observe the following addtional code in pom.xml file

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context-support -->
<dependency>
<artifactId>spring-context-support</artifactId>
<groupId>org.springframework</groupId>
<version>${sp.version}</version>
<exclusions>
<exclusion>
<groupId>org.springframework</groupId>
<artifactId>spring-aop</artifactId>
</exclusion> </exclusions> </dependency>
```

Maven Build Commands (phases)

To pack the entire app into jar file

right click on the Project ---> run as ---> build ... --->

goals :: package

✓ target

> generated-sources

>generated-test-sources

>maven-archiver

>maven-status

>surefire-reports

What to be done in the pom.xml file to get the jar/war file with our choice name as packaging activitiy? place <finalName>App11

</finalName> inside the <build> of the pom.xml

is

note: if the project created using "maven-archetype-quickstart" then

we will get jar file representing the standalone Project

note: if the project created using "maven-archetype-webappp" then we will ar file representing the web application

MavenProj01-0.0.1-SNAPSHOT.jar (gives the jar file representing the project)

(project name as

To clean the target folder

the jar file name)

right click on the Project ---> run as ---> build ... ---> goals :: clean

(empties the target folder)

To perform clean and packaging togather

right click on the Project ---> run as ---> build ... ---> goals: clean package

To keep current project packing jar file in maven local repository

right click on the Project ---> run as ---> build ... ---> goals :: install

ctrl+shift+f: for formatting editor content

note: if we use mvn install command then the jar file name

in target folder comes with name specified in the <finalName> tag

and the jar file name in maven Local repository comes with name of maven Project name

=> keeps the maven current project packing jar file in maven's Local repository

..

repository

> nit

> MavenProj01-Depenency Management > 0.0.1-SNAPSHOT

App.java public class App

```
{
public int sum(int x,int y) {
return x+y;
}
}
```

↑ Sort

View ✓

Name

Date modified

SQL

remote.repositories

2/15/2024 10:04 AM

maven-metadata-local

2/15/2024 10:04 AM

MavenProj01-Depenency Management-0.0.1-SNAPSHOT MavenProj01-Depenency Management-0.0.1-SNAPSHOT

2/15/2024 10:04 AM

2/14/2024 10:56 AM

HQI☆

Procedure to use the above Project jar file classes in the new Maven Project of the same system by collecting the above project from the Maven Local Repository of the same system

step1) create new Project having the name Proj2 using maven-archetype--quickstart

change

and its java version to 21

Proj02

src/main/java

com.nt.test

› ▸ App2.java

>

src/test/java

>

JRE System Library [JavaSE-17]

>

Maven Dependencies

<

src

>

target

Mpom.xml

step2) add the above project jar file to current Project (Proj2) as the dependency

In pom.xml of the Proj02

```
<dependency>
<groupId>nit</groupId>
<artifactId>Maven Proj01-Depenency Management</artifactId>
<version>0.0.1-SNAPSHOT</version>
</dependency>
```

✓ Proj02

src/main/java

#com.nt.test

› App2.java

>

src/test/java

>

JRE System Library [JavaSE-17]

Maven Dependencies

>

junit-4.11.jar - C:\Users\Nataraz\.m2\repository\junit\junit\4.11

>hamcrest-core-1.3.jar - C:\Users\Nataraz\.m2\repository\org\h

>

spring-context-support-6.0.8.jar - C:\Users\Nataraz\.m2\repos

>spring-beans-6.0.8.jar - C:\Users\Nataraz\.m2\repository\org\ >spring-context-6.0.8.jar - C:\Users\Nataraz\.m2\repository\org >spring-core-6.0.8.jar - C:\Users\Nataraz\.m2\repository\org\sp

spring-jdbc-6.0.8.jar - C:\Users\Nataraz\.m2\repository\org\sp >spring-tx-6.0.8.jar - C:\Users\Nataraz\.m2\repository\org\sprir >spring-orm-6.0.8.jar - C:\Users\Nataraz\.m2\repository\org\sp

>spring-webmvc-6.0.8.jar - C:\Users\Nataraz\.m2\repository\or

spring-expression-6.0.8.jar - C:\Users\Nataraz\.m2\repository\ >spring-web-6.0.8.jar -

C:\Users\Nataraz\.m2\repository\org\sp

>

> micrometer-observation-1.10.6.jar - C:\Users\Nataraz\.m2\rep micrometer-commons-1.10.6.jar - C:\Users\Nataraz\.m2\repos MavenProj01-DepenencyManagement [without test code]

add under <dependencies> tag => collect gropu id, artifact id, version

info from .pom file of maven Local repository from same folder where the above project jar file is saved

step3) Use "App" class of the above project in the current project to get the feeling Proj2 is using

previous/above project content

App2.java

```
package com.nt.test;

import com.nt.client.App;

/**

* Hello world!

*/

public class App2

{

public static void main(String[] args)

{

App app=new App();

int result= app.sum(10, 20);

System.out.println("result is::"+result);

}

}
```

step4) Run the Application

In Proj2 ---> main class with main(-) method ----> run as ---> Java App

What is realtime use case for "maven install " command?

Ans) maven "install" command creates jar file representing the current project and also keeps that jar file

in maven Local Repository as dependency.. So we can use that dependency in any other maven

project of same system having reusability

Pratical example Scenario for Maven Inheritence (Linking one maven project with

==

step1) Keep any existing maven Project ready (MavenProj01)

another maven project)

step2) create new maven Project (MavenProj02)

step3) add Maven Proj01 details as parent project details in the

pom.xml file MavenProj02 as shown below

```
<parent>
```

```
<groupId>org.nit</groupId>
<artifactId>Maven Proj01</artifactId>
<version>0.0.1-SNAPSHOT</version>
</parent>
```

step4) add jar file as dependency in pom.xml file of Maven Proj02 specifying the scope as import

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context-support</artifactId>
<version>6.2.3</version>
<type>pom</type>
<scope>import</scope>
</dependency>
```

step5) Save the changes and see jar files that are added to Maven Dependencies folder

```
Maven Dependencies
>

junit-jupiter-api-5.11.0.jar - C:\Users\Nataraz\.m2\reposi
opentest4j-1.3.0.jar - C:\Users\Nataraz\.m2\repository\c
> junit-platform-commons-1.11.0.jar - C:\Users\Nataraz\.r
> apiguardian-api-1.1.2.jar - C:\Users\Nataraz\.m2\reposit
junit-jupiter-params-5.11.0.jar - C:\Users\Nataraz\.m2\re
spring-beans-6.2.3.jar - C:\Users\Nataraz\.m2\repositor
>
>
>spring-context-6.2.3.jar - C:\Users\Nataraz\.m2\reposito
>
>
micrometer-commons-1.14.4.jar - C:\Users\Nataraz\.m2
micrometer-observation-1.14.4.jar - C:\Users\Nataraz\.n
>spring-core-6.2.3.jar - C:\Users\Nataraz\.m2\repository\ >spring-jcl-6.2.3.jar - C:\Users\Nataraz\.m2\repository\or
>spring-context-support-6.2.3.jar - C:\Users\Nataraz\.m2' >spring-jdbc-6.2.3.jar - C:\Users\Nataraz\.m2\repository\
>spring-tx-6.2.3.jar - C:\Users\Nataraz\.m2\repository\or! >spring-orm-6.2.3.jar - C:\Users\Nataraz\.m2\repository\
>spring-webmvc-6.2.3.jar - C:\Users\Nataraz\.m2\reposit >spring-aop-6.2.3.jar - C:\Users\Nataraz\.m2\repository\
>spring-expression-6.2.3.jar - C:\Users\Nataraz\.m2\repo >spring-web-6.2.3.jar - C:\Users\Nataraz\.m2\repository\
```