01

Gerenciando dependências no Maven

Transcrição

[00:00] Agora que já aprendemos como fazemos para criarmos um projeto ou migrar um projeto para o Maven, chegou a hora de tratarmos de outros recursos.

[00:10] Talvez o principal recurso, a principal funcionalidade de utilização do Maven é para a parte de gerenciamento de dependências. Nesse vídeo, vamos aprender justamente isso, como funciona essa parte de gestão de dependências com o Maven.

[00:24] Estou com aquele nosso projeto "loja", que tínhamos construído do zero. Se abrimos o "pom.xml", ele está praticamente vazio, só está com aquelas informações do `groupId`, `artifactId` e a versão do nosso projeto. Porém, não tem nenhuma dependência.

[00:39] Vamos aprender como funciona isso. Tínhamos discutido em vídeos anteriores que um dos problemas de uma aplicação Java é justamente esse de gerenciar dependências. "Eu quero usar o Hibernate, quero usar o Spring MVC, quero usar o XStream, quero usar o JFreeChart etc".

[00:57] Você tem um monte de bibliotecas para usar, você tem que baixá-las manualmente, tem que entrar no site de cada uma, fazer o download, descompactar, você tem que saber qual é a versão que você quer usar e tem que tomar cuidado com bibliotecas compartilhadas.

[01:10] Às vezes, o JFreeChart usa uma biblioteca qualquer, o Log4j na versão 1.2, mas o Hibernate usa na versão 1.4. E você sem querer baixou esses dois

JARs e colocou junto com o projeto. Você fica com a mesma biblioteca em versões distintas e isso pode gerar um problema na sua aplicação.

[01:29] Depois, para você evoluir essa biblioteca também é chato, tem que baixar manualmente. Para livrar-se de toda essa dor de cabeça e todo esse problema, entre o Maven com a sua principal funcionalidade de gerenciamento de dependências. Como funciona isso no Maven? É bem tranquilo. Aqui no arquivo "pom.xml" é exatamente neste arquivo que fazemos essa declaração das dependências que você quer utilizar no seu projeto.

[01:54] Aqui dentro da *tag* raiz, a *tag* `project`, pode ser aqui embaixo da *tag* `version`, existe uma *tag* chamada `dependencies` e dentro dela você declara cada uma das pendências que você quer utilizar no seu projeto, as bibliotecas e *frameworks* que você for utilizar.

[02:11] Por exemplo: eu quero utilizar o JUnit. Ele já está aqui na aplicação, só que ele está adicionado com esse recurso do Eclipse, ele está vinculado por fora do Maven. Vamos fazer essa alteração. Eu quero tirar ele daqui e declará-lo como uma dependência.

[02:28] Vou clicar aqui com o botão direito do mouse no projeto "loja" e selecionar "Build Path > Configure Build Path > Libraries". Está aqui o "JUnit 4". Eu vou remover daqui, "Remover > Apply and Close". Perceba que começou a dar erro de compilação na nossa classe de teste, porque ela depende do JUnit e eu acabei de excluí-lo do projeto.

[02:48] Só que eu vou adicioná-lo como uma dependência do Maven. Para fazer isso, lá no "pom.xml", dentro da *tag* `dependencies` existe uma *tag* chamada 'dependency' e é aqui que você declara uma dependência.

[03:01] Toda *tag* 'dependency' tem algumas coisas que você precisa informar. Por exemplo: o 'grupoId'. Exatamente igual temos aqui o nosso 'groupId'. O Maven precisa saber qual é a organização que é dona dessa biblioteca, dessa dependência que você quer baixar aqui no seu projeto.

[03:17] No caso do JUnit, o “groupId”, quando eles criaram a biblioteca JUnit eles colocaram aqui JUnit, simplesmente JUnit. Esse é o “groupId”.

[03:27] Qual que é o “artifactId”? Perceba que o nosso projeto pode virar uma biblioteca, uma dependência de um outro projeto. Uma dependência nada mais é do que um “groupId” e um “artifactId”. Quem é organização e quem é o projeto dessa organização que você quer utilizar como dependência. No caso do JUnit, o “artifactId” também se chama “JUnit”, `<artifactId>junit</artifactId>`.

[03:50] Além dessas duas informações tem mais uma importante: existe a *tag* `version`. Qual é a versão do JUnit que você quer utilizar? Aqui no nosso projeto nós colocamos que é a versão 1.0, `<version>1.0.0</version>`.

[04:03] Conforme vamos evoluindo o nosso projeto, podemos ir lançando versões, tem a 1.1, a 1.2, a 1.3, a 2.0, 2.1, 2.2, 3.0 etc - você pode manter múltiplas versões de um projeto, de um `artifactId`. Se você quiser baixar uma versão específica na *tag* ‘version’, é onde você determina qual é a versão desse artefato.

[04:27] No caso aqui do JUnit, eu quero usar versão 4.12, eu sei aqui de cabeça que existe a versão 4.12.

[04:33] Coloquei aqui a *tag* `version`, `<version>4.12</version>`. Para fechar, existe mais uma *tag* importante, chamada `scope` - que é para dizer qual é o escopo dessa dependência. “Como assim, Rodrigo, 'escopo'? ” Existem alguns escopos que podemos adicionar aqui.

[04:49] Eu utilizei o atalho “Ctrl + Espaço” e ele completou para nós. Tem o escopo de “compile”, “provided”, “runtime”, “system”, “test”.

[04:56] O “compile”, você estaria dizendo: “olhe, Maven, eu preciso dessa dependência em tempo de compilação só para compilar o meu código-fonte. Quando eu for gerar o *build* do projeto você pode ignorar essa dependência.

Ela só vai ser usada para compilar". Se fosse esse o caso, você marcaria como escopo "compile".

[05:17] Tem o "provided". O "provided" é para quando você utiliza, por exemplo, servidores de aplicação. Você precisa da dependência aqui para compilar, para rodar o seu projeto, mas na hora de gerar o *build* você não precisa se preocupar porque ela é provida pelo próprio servidor de aplicação.

[05:32] Tem a "runtime", que ela vai ser provida só em tempo de execução. E tem a "test", que é o exemplo aqui do JUnit. Quando você tem uma dependência que só é usada durante a execução dos testes, você não vai compilar o seu código-fonte do projeto e nem usá-la em produção. Ela serve apenas durante a fase de rodar os testes do projeto.

[05:56] No caso do JUnit, é exatamente essa situação. O JUnit é usado apenas para rodar o teste do projeto. Existe essa *tag* `scope`, `<scope>test</scope>` e ela tem esses tipos de escopos. Dependendo da biblioteca, você terá que dizer qual é o escopo dela para que você não a utilize em momentos errados durante o *build* e o empacotamento da sua aplicação.

[06:17] Utilizei o atalho "Ctrl + S" para salvar. Na hora que você salvar, o Maven vai baixar essa biblioteca da internet.

[06:25] Depois vamos ver com calma de onde que ele baixa essa biblioteca. "Da internet, mas de onde?" Vamos ver com calma isso quando formos discutir sobre repositórios.

[06:33] Ele baixou a dependência e adicionou no projeto. Baixou com sucesso. Se quisermos ver se ele baixou mesmo a dependência - se expandirmos aqui o projeto loja, perceba que embaixo de "JRE System Library" apareceu essa nova opção, "Maven Dependencies".

[06:48] Se expandirmos "Maven Dependencies", aparece "junit-4.12". Ele baixou o JUnit. Apareceu também um tal de "hamcrest-core-1.3", eu não declarei ess

“hamcrest” aqui no "pom.xml". Esse “hamcrest” é uma dependência do JUnit, uma dependência pode ter dependência.

[07:08] O JUnit pode usar outra biblioteca, que pode usar outra biblioteca e o Maven é que vai se virar para saber essa árvore, essa hierarquia de dependências. Eu só preciso dizer: “Maven quero o JUnit. Se o JUnit depende de outra biblioteca, o problema é seu. Você que vai descobrir isso e você que vai baixar essa biblioteca”.

[07:26] E se essa biblioteca é dependência de outra, ele quem iria sair procurando, encontrando essa hierarquia de dependências e baixando uma por uma para satisfazer a nossa dependência do JUnit no projeto.

[07:39] Perceba como é simples declarar uma dependência aqui no Maven. É só você ir lá no seu “pom.xml”, ir na *tag* `dependencies` e dentro dela adicionar a dependência. Você precisa digitar o ‘groupId’ ou ‘artifactId’, dependendo do caso você precisa da versão e dependendo do caso você precisa do escopo.

[07:58] “Poxa, Rodrigo! Mas você sabia de cabeça que esse era o ‘groupId’ do JUnit, esse era o ‘artifactId’ e essa era a versão. Eu vou ter que decorar isso, saber ao certo? Eu quero baixar aqui o *Spring* no meu projeto, eu vou ter que saber qual é o ‘artifactId’ e o ‘groupId’? ” Não, você não precisa disso. Você pode pesquisar no Google, pesquisar na internet. É só copiar e colar aqui a dependência.

[08:19] No próximo vídeo, vamos aprender exatamente a fazer isso - como pesquisar uma dependência e apenas copiar e colar aqui no nosso arquivo “pom.xml”. Vejo vocês lá, um abraço!