04

Alterando o repositório remoto de dependências

Transcrição

[00:00] Agora que já vimos como adicionar uma dependência no Maven através da *tag* `dependency` lá no arquivo "pom.xml" e também a pesquisar pela dependência para não ter que ficar memorizando o "groupId", "artifactId".

[00:13] Chegou a hora de entendermos melhor como o Maven baixa essas dependências. Se analisarmos o "pom.xml", em nenhum local dissemos de onde que o Maven vai baixar essas dependências, se é da internet ou do computador.

[00:28] Isso não foi configurado porque, por padrão, o Maven vai primeiro buscar no repositório local que está no seu computador. Daqui a pouco eu mostro esse repositório para vocês. E se ele não encontrar essa dependência no repositório local, ele pesquisa na internet em um repositório central do próprio Maven.

[00:45] Se você quiser conhecer um pouco sobre esse repositório central, você pode abrir o Google e pesquisar "Maven repository" e aí vai aparecer esse site aqui, mvnrepository.com (https://mvnrepository.com/). Esse é o repositório central do Maven.

[01:02] Aqui é onde estão todos os projetos, os *frameworks*, as bibliotecas e com tudo disponibilizado nesse repositório central do Maven. É aqui que ele faz a consulta e faz o download do JAR e das dependências conforme o seu projeto demandar. Esse é o repositório central.

[01:18] Porém, pode acontecer de determinada dependência que você quer utilizar no seu projeto não estar disponibilizada no repositório central no Maven e estar disponibilizado em um outro repositório de outra empresa, ou pode ser até um repositório privado, interno da sua organização.

[01:36] Você quer manter um controle melhor, não quer que o pessoal da área de programação baixe diretamente da internet. Você quer deixar tudo local na sua empresa. Tem como você configurar um repositório interno da sua empresa ou utilizar um outro repositório que não seja o do próprio Maven.

[01:52] Para fazer isso é só você vir aqui no seu arquivo "pom.xml". Dentro da *tag* `project`, da *tag* raiz aqui do "pom.xml", você adiciona uma *tag* chamada "repositories". Eu já tenho ela copiada, só para não perdermos muito tempo. Vou colar e essa é a *tag* `<repositories>`. Aqui dentro você pode adicionar um ou mais repositórios, que o Maven sempre vai procurar a ordem de declaração.

[02:17] Tem a *tag* `<repository>`, tem o `<id>`. Aqui, por exemplo, eu coloquei o repositório do Spring. O Spring tem um repositório próprio que tem lá os JAR, as bibliotecas do Spring.

[02:27] Tem o `<id>` e uma `<url>`, o que interessa mesmo é só a URL, `<url>https://repo.spring.io/release</url>`. A partir de agora quando eu solicitar para o Maven adicionar uma nova dependência aqui no projeto. Ele vai pesquisar nesse repositório aqui: `<repository> <id>spring-repo</id> <url>https://repo.spring.io/release</url> </repository>`, um novo repositório remoto que você acabou de adicionar.

[02:41] Aqui em https://repo.spring.io/release (https://repo.spring.io/release) não precisa necessariamente ser um repositório hospedado na internet. Ao invés de "http" poderia colocar aqui "file:///home/", uma pasta da minha máquina, ou poderia ser "http://ip" (http://ip%E2%80%9D) de um servidor interno da sua empresa. Pode ser algo interno, algo local no seu computador ou algo hospedado na internet - seja um repositório no Maven ou de outros projetos.

[03:16] Tem o repositório do Spring, tem o repositório também da JBoss, tem alguns outros repositórios que você pode utilizar. Vou deixar esse salvo aqui de exemplo, `<url>https://repo.spring.io/release</url>` .

[03:25] Toda vez que você adiciona uma nova dependência aqui no "pom.xml", o Maven primeiramente vai olhar no *cache* local da sua máquina para ver se já existe essa dependência salva no seu computador, porque aí ele não precisa acessar nenhum endereço para baixar essas dependências, então acaba sendo mais rápido.

[03:43] Só se ele não encontrar no *cache* local é que ele vai acessar os repositórios que você configurou ou acessar o repositório central do Maven. Esse repositório local, esse *cache*, é um diretório chamado ".m2".

[03:58] Como a pasta começa com "." é um diretório oculto. Ele fica localizado no diretório "Home" do seu usuário no seu computador. Isso vai depender do sistema operacional. Eu estou usando o Linux. Eu estou aqui dentro do meu diretório "Home", ele só está mostrando as pastas comuns. Se eu quiser exibir as pastas ocultas eu utilizo o atalho "Ctrl + H" - e perceba que aqui tem uma pasta chamada ".m2".

[04:24] Esse é o *cache* local do Maven. Se entrarmos aqui tem uma pasta chamada "repository" e dentro dessa pasta tem todos os JARs, todas as bibliotecas que ele já baixou.

[04:34] Percebe que tem muitas pastas aqui, porque são as dependências, as dependências das dependências e aí vai de maneira hierárquica. Por exemplo: baixamos o JUnit, tem uma pasta aqui "JUnit" e dentro dela tem as versões do "JUnit". Já tinham algumas aqui de outros projetos desse computador. A "4.12" foi a que baixámos, está aqui "Junit-4.12.jar". Tem alguns arquivos específicos que o Maven utiliza.

[05:02] Esse é o repositório local. Toda vez que você adiciona uma nova dependência no projeto, primeiro o Maven olha na sua pasta ".m2" para ver s

ele encontra essa dependência aqui. Se não encontrar, ele pesquisa na internet no repositório central dele ou nos repositórios que você configurar no “pom.xml” do seu projeto.

[05:20] Uma dica importante: às vezes acontece um determinado problema. Você adiciona uma dependência no "pom.xml", copia ela da internet e ela está certa só que está dando algum erro. Ele fica marcando de vermelho e você olha e está tudo ok, a dependência está certa, o “groupId” está certo e “artifactId” está certo.

[05:38] Às vezes acontece algum problema na hora que o Maven baixa essa dependência da internet. Se tiver um erro aqui e ele não se resolver. Se você clicar com o botão direito do mouse no projeto em “loja > Maven > Update Project > OK”, o Maven vai meio que sincronizar. Mesmo assim, caso o erro continue. Às vezes, pode ser um problema do seu repositório local, do seu *cache* que deu algum problema na hora de baixar uma dependência.

[06:06] Uma dica que eu dou é: quando isso acontecer entra na pasta “.m2 > repository > seleciona tudo > Shift + Delete”, apague tudo. Depois que ele concluir a exclusão, você vai no Eclipse e faz aquele “Update Project” novamente: “loja > Maven > Update Project > OK”. Isso vai forçar o Maven a baixar as dependências.

[06:31] Perceba que aqui embaixo, eu vou clicar aqui no ícone no canto inferior do lado direito da tela para ver a aba "Progress". Agora ele está atualizando o projeto.

[06:39] Ele não achou no repositório local aqui na minha máquina, por isso ele está baixando da internet. Quando ele vai baixar da internet, geralmente demora um pouco. Vai depender da sua conexão, do número de dependências que você tem no seu projeto e ele vai baixando cada uma dessas dependências e vai criando aqui os diretórios.

[06:55] Os diretórios são de acordo com o “artifactId” e o “groupId”. Se for um nome aqui, tipo “com.alura.projeto” ele vai criando esses diretórios como se fossem pacotes do Java. Ele vai criando os diretórios aqui e baixando os JARs, as dependências.

[07:14] Tem coisas que ele baixa aqui que não estão no nosso projeto, que são *plugins* e coisas internas do próprio Maven. Não necessariamente ele só vai baixar essas duas dependências, ele vai baixar também as dependências de cada uma dessas dependências e *plugins* e coisas internas do próprio Maven - por isso que está demorando e provavelmente vai demorar alguns minutos.

[07:32] Esse era o objetivo do vídeo de hoje, explicar para vocês essa questão do repositório - que o Maven tem esse *cache* local no seu computador, essa pasta “.m2”, que fica no diretório “Home” do seu usuário.

[07:44] Dependendo do sistema operacional, é de um jeito diferente para você acessar este diretório. Se ele não encontrar lá, ele busca na internet, no repositório central do Maven, no site mvnrepository.com, ou no seu “pom.xml”. Se você tiver configurado algum repositório, ele vai buscar por lá.

[08:03] É assim que funciona os repositórios e você pode ter um repositório privado, interno da sua empresa, se você quiser filtrar e limitar as bibliotecas e dependências que a sua equipe e o time de desenvolvimento da sua empresa, pode baixar. Você pode utilizar um repositório privado e pode usar também uma ferramenta mais avançada para gerenciar as dependências e artefatos, como Nexus, dentre outras. Mas esse não será o foco desse treinamento.

[08:28] Espero que vocês tenham gostado. Vejo vocês no próximo vídeo onde continuaremos aprendendo outros recursos do Maven. Um abraço!