09

Conclusão

Transcrição

[00:00] Chegamos ao final do nosso treinamento sobre **Maven**, sobre como funciona o Maven. Durante esse treinamento, aprendemos o que é o Maven, quais são os dois pilares principais do Maven - que é a gestão de dependências e *build* do projeto. Fomos aprendendo esses recursos e algumas aplicações de exemplo.

[00:21] Aprendemos como se faz para instalar o Maven. Você pode baixar o ZIP, descompactar e executar pelo *prompt* de comandos ou através da sua IDE. No caso do Eclipse, ele já tem uma integração nativa com o Maven. Aprendemos como se faz para criar uma aplicação Maven usando aquela opção do Eclipse, "New > Maven Project".

[00:39] Vimos a questão do *archetype*, que podemos pular ou escolher um *archetype* específico que já vem com uma estrutura específica. Entendemos essa estrutura de diretórios de uma aplicação Maven, o que é cada um desses diretórios - "src/main/java", "src/main/resources" , "src/test/java" e "src/test/resources".

[00:57] Aprendemos a trabalhar no "pom.xml", aquele arquivo XML de configurações do Maven para esse projeto. Vimos a declaração do `groupId` , `artifactId` , as dependências. Uma das principais utilizações do Maven, que é como declaramos dependências, como pesquisamos por essas dependências.

[01:15] Aprendemos sobre o repositório central do Maven, como adicionar outros repositórios no Maven. Aprendemos sobre o segundo pilar do Maven · que é essa parte de *build*, como é que eu faço para fazer o *build* da minha

aplicação, para compilar, para rodar testes, gerar o JAR, o WAR e fazer o *deploy*.

[01:31] Aprendemos como customizar esse *build*, trocando o nome do arquivo, do artefato gerado. Aprendemos a como adicionar *plugins*, como o *plugin* do compilador do Maven para trocar versão do Java, o *plugin* do Jacoco para ver o relatório de cobertura de testes.

[01:45] Aprendemos a mexer naquele arquivo "settings.xml" para declararmos um *proxy*, por exemplo - dentre outras coisas que podem ser declaradas aqui.

[01:54] Por fim, aprendemos sobre essa questão de modularização. Como ter uma aplicação modularizada, uma aplicação cujo o *packaging* é "pom" e ela declara dependências que são compartilhadas entre os seus módulos e faz essa declaração de quais módulos pertencem a essa aplicação.

[02:10] Você tem o projeto *parent* - o projeto raiz que agrupa esses subprojetos e cada projeto filho declara essa *tag* `<parent>`, então "pom.xml" fica alterado nessa situação. Você está herdando do "pom.xml", daquele projeto e você herda todas as dependências e configurações que estão descritas lá. Com isso, você pode quebrar a sua aplicação e ter um reaproveitamento, evitar código duplicado de dependências e configurações.

[02:38] Um módulo pode depender do outro. Por exemplo: aqui temos esse módulo `rh-web`, que ele depende do módulo `rh-persistencia`. Basta declarar como sendo uma dependência. O módulo nada mais é do que uma dependência, é um projeto que pode ser adicionado como dependência a outro projeto.

[02:56] Aprendemos sobre essa questão de modularização e que pode ser usada para criar aplicações que seguem aquela arquitetura de microsserviços.

[03:04] Esses foram os recursos que aprendemos durante esse treinamento sobre o Maven para te dar esse *overview*, essa noção do que é o Maven, para

que ele serve, como que funciona e como eu crio uma aplicação.

[03:15] O Maven, como eu tinha dito, é a principal ferramenta para gerenciamento de dependências e *build* de aplicações em Java. Muito provavelmente você vai trabalhar em aplicações que utilizam o Maven. É extremamente importante e recomendado que você utilize alguma ferramenta como o Maven.

[03:29] Tem o Gradle também, que é uma outra ferramenta bem popular e concorrente forte do Maven. O Maven acaba sendo o mais utilizado no mercado, o "mais padrão", digamos assim, no caso do Java.

[03:41] Espero que vocês tenham gostado desse treinamento e que tenham trabalhado com o Maven. Vejo vocês em outros treinamentos da Alura. Um abraço e até lá!