01 Apresentação

Transcrição

[00:00] Boas-vindas ao treinamento de **Maven: gerenciamento de dependências e build de aplicações Java**. Meu nome é Rodrigo Ferreira e sou o instrutor que acompanhará vocês durante esse treinamento.

[00:06] O objetivo desse treinamento é aprendermos o que é esse **Maven** do Java - uma ferramenta bem famosa, bem popular e presente em praticamente todos os projetos, que cuida da parte de BIO e de gestão de dependências, dentre outras coisas.

[00:19] No curso, vamos aprender como funciona o Maven, como fazemos para instalá-lo, qual é a vantagem e a necessidade de utilização de Maven. Nesse slide temos, de uma maneira bem resumida o foco do treinamento.

[00:31] Vamos discutir sobre essa questão de **Processo de *build*** em uma aplicação em Java. Por que eu preciso de uma ferramenta para gerenciar o *build* do meu projeto? Como eu faço um *build* utilizando Java? De maneira tradicional, sem usar nenhuma ferramenta e utilizando o Maven posteriormente.

[00:45] Vamos entender todo esse processo de *build*, de compilar arquivos, compilar classes de testes - se você tem testes automatizados - de gerar um relatório de testes, interromper o processo de *build* se

algum teste falhar ou se estiver com o percentual de cobertura abaixo, por exemplo.

[01:02] E também vem a parte de **gestão de dependências**. É muito provável que na sua aplicação você esteja utilizando um monte de bibliotecas de *framework*. *Framework MVC*, *framework* para persistência, *framework* para trabalhar com o XML, com JSON para gerar gráficos, etc.

[01:18] Temos um monte de bibliotecas na sua aplicação. Como você faz para gerenciar essas bibliotecas? Qual é a versão específica de cada biblioteca e as dependências de cada uma dessas bibliotecas? Às vezes, você pode ter até conflitos. Você tem uma biblioteca que está usando uma outra biblioteca "X", que é usada também por uma outra dependência. Elas estão dando um conflitos... Enfim.

[01:38] Gerenciar isso manualmente é muito trabalhoso, muito complexo e você não quer perder tempo com isso. Você tem que focar na aplicação em si, nas funcionalidades do seu software.

[01:47] Vamos discutir sobre o Maven, o que é o Maven, como fazemos para baixar e instalar o Maven, criar um projeto com Maven, analisar a estrutura de diretórios em uma aplicação Maven, entender porque ela tem aquela estrutura e entender aquele arquivo "Pom.xml" - talvez você já tenha esbarrado com esse arquivo.

[02:03] Ver essa questão de gestão de dependências, que é a principal utilização do Maven para essa parte de gerenciar as dependências.

Como adicionamos e atualizamos essas dependências do Maven e como ele faz para gerenciar isso.

[02:17] Também o processo de *build*. Qual é o passo a passo para o Maven gerar o *build* da aplicação se ela tiver testes automatizados... Enfim. Também outros assuntos, como plugins, módulos e outros recursos que temos dentro do Maven.

[02:30] Esse treinamento é bem focado no Maven em si, esse é o objetivo do curso. Esse é um resumo do que vamos ver pela frente. Eu espero que vocês gostem bastante desse treinamento. Vejo vocês na primeira aula, onde vamos falar um pouco sobre esses processos de *build* e gestão de dependências manuais e como o Maven vai fazer para nos ajudar. Vejo vocês nas próximas aulas. Um abraço!

02 Projeto inicial

Baixe aqui o ZIP com os arquivos para o projeto inicial deste treinamento, necessário para a continuidade do mesmo.

03 Build e dependências em uma aplicação Java

Transcrição

[00:00] Agora que já vimos o que será abordado no treinamento de **Maven**, vamos entender um pouco qual é a motivação da utilização do Maven, o que ele veio para resolver.

[00:11] Eu estou com a minha IDE aberta, estou usando o *eclipse*, mas você pode usar a IDE de sua preferência. Eu tenho um projeto de exemplo, uma aplicação Java Web tradicional. Vamos considerar que essa aplicação está pronta. Temos do lado esquerdo da tela várias classes que foram criadas e temos os `controllers`, `dao`. Toda a parte implementada.

[00:35] Agora, preciso gerar um *build* desse projeto, gerar um pacote para passar para a galera da infraestrutura para fazer o *deploy* e colocar em produção, colocar no ar essa aplicação.

[00:46] Vem aquela questão: “como é que eu faço para gerar um WAR? Já que é aplicação web. Como que eu gero um WAR dessa aplicação?” Como não tem nenhuma ferramenta, podemos fazer isso diretamente pela IDE, diretamente pelo Eclipse. Eu poderia simplesmente chegar aqui e clicar com botão direito em cima do projeto “controle-products > Export > filtrar “WAR file”" e ir clicando em “Next” em “Destination” posso escolher onde quero salvar o WAR.

[01:09] Porém, nem sempre é tão simples e fácil assim de você gerar um *build* do projeto, vou selecionar "Cancel". Às vezes, você tem que fazer algumas coisas antes disso, você tem um passo a passo, tem várias tarefas para serem executadas antes de gerar o WAR ou JAR da sua aplicação.

[01:27] Por exemplo: essa aplicação tem testes automatizados. Então, antes de gerar o JAR ou WAR da aplicação, preciso rodar esses testes, ver se está tudo passando, se está tudo funcionando, e não gerar o JAR se tiver algum teste falhando. Antes de gerar o WAR, eu tenho que lembrar de rodar os testes.

[01:45] Eu teria que clicar com o botão direito do mouse em “test > Run As > JUnit test” e rodar os testes da aplicação. Só se estiver tudo passando, eu gero o WAR da aplicação. Além disso, há outra coisa que eu tenho que fazer também. Em "src > br > hibernate.properties" tem umas propriedades do Hibernate, a biblioteca para fazer integração com banco de dados.

[02:04] Só que essas configurações estão apontando para o banco de dados local. Para gerar o WAR, para gerar o pacote da aplicação, eu tenho que trocar `localhost`, tenho que colocar o IP do servidor, as propriedades do servidor do banco de dados de produção ou do ambiente de homologação. Eu tenho que trocar essas propriedades.

[02:23] Como é que eu faço para trocar essas propriedades? Eu vou ter que lembrar também de abrir o arquivo e modificar `localhost` para "IP" do servidor. Outra coisa para eu ter que lembrar, não tem como fazer isso de uma maneira automática - tem dois arquivos, um para produção, um para cada ambiente. Fica mais um trabalho para fazermos.

[02:43] Se eu tivesse outras atividades nessa configuração também teria que renomear um arquivo, criar um diretório, jogar um arquivo para aquele diretório e apagar um determinado arquivo. Todo esse passo a passo faz parte do processo de *build* da aplicação.

[02:58] Antes de você gerar o pacote, o JAR ou WAR, você tem que lembrar de executar o passo a passo - e provavelmente é comum as pessoas esquecerem isso. É comum criarmos um documento, um passo a passo, um tutorial explicando como fazer o *build* do projeto.

Isso é um negócio muito trabalhoso e está sujeito a falhas. Seria melhor se automatizarmos uma ferramenta para automatizar isso.

[03:21] O pessoal do Java há muito tempo criou uma ferramenta bem famosa, que é o ***Apache Ant***. Talvez você já tenha trabalhado com *Apache Ant*. Para você automatizar esse processo de compilar as classes do projeto, compilar as classes de teste, rodar os testes, criar um arquivo, mover um arquivo, apagar um diretório, apagar um arquivo e renomear coisas, você poderia utilizar o *Apache Ant* para fazer isso.

[03:46] Bastaria você ir no projeto, criar um arquivo `build.xml` e configurar o *Ant* para executar o *build* do seu projeto. Agora, você não faz todo esse passo a passo manualmente, você executa o *Ant*. Resolveria o caso.

[04:00] Porém, tem outro detalhe nessa aplicação: e quanto às bibliotecas, os *framework* e as dependências dessa aplicação? Como é uma aplicação Java web, no "Web-Content", dentro da pasta "WEB-INF" tem o diretório "lib" e todos os JARs da aplicação se encontram nesta pasta.

[04:16] Temos essa aplicação está usando o `Spring framework`, está usando *Hibernate*, está usando biblioteca de `log`. Tem um monte de bibliotecas sendo utilizadas.

[04:27] Sempre que eu precisar atualizar uma biblioteca dessa, vou ter que atualizar o Hibernate, por exemplo, baixar os JARs do Hibernate e substituir os JARs do Hibernate. Que são "hibernate-annotations.jar",

“hibernate-commons-annotations.jar” e “hibernate3.jar” - mas será que são só esses três mesmo?

[04:43] E as dependências do Hibernate? O Hibernate depende das bibliotecas “cglib-nodep-2.1_3.jar”, “commons-fileupload-1.2.1.jar” e “commons-logging.jar”.

[04:48] Vai ter que saber de cabeça quais são os JARs que vai ter que substituir. Se uma dessas dependências também é utilizada pelo *Spring*, agora eu vou ter que atualizar também o *Spring*?

[04:59] Baixar esses JARs, ficar gerenciando esses JARs manualmente é um negócio muito trabalhoso, é muito complicado e sujeito às falhas também. Para resolver esse problema, mais uma ferramenta que foi criada para o Java, o ***Apache Ivy***. Talvez você já tenha ouvido falar, tenha utilizado, talvez não. Não é uma ferramenta tão popular assim.

[05:19] O *Ivy* era outra ferramenta que você poderia utilizar em conjunto até com o *Ant* para resolver esses problemas da sua aplicação. Você tem o *Ant* fazendo a parte do *build* e o *Ivy* cuidando especificamente das dependências.

[05:35] Você criaria um arquivo chamado `Ivy.xml` e só declara qual é a dependência. Quero o Hibernate na versão 3, quero o *Spring* na versão "X". Você declara as dependências e o *Ivy* baixa essas dependências, resolve todas as dependências necessárias para essa biblioteca e configura tudo no seu projeto.

[05:55] Uma solução comum era utilizar essas duas ferramentas em conjunto, o *Apache Ant* e o *Apache Ivy*. Uma para cuidar da parte

de *build*, de compilar, rodar testes, criar diretórios e mover arquivos; e a outra para gerenciar as dependências do projeto. Isso era bem comum. Porém, são duas ferramentas separadas que se integram entre si, mas não seria melhor juntar essas duas ferramentas? Esse é justamente o objetivo do ***Maven***.

[06:20] O Maven já cuida dessas duas etapas para você, ele já gerencia o *build* da aplicação, todo o processo, todo passo a passo para gerar o *build* no pacote da sua aplicação - com todo esse passo a passo de compilar, de executar testes, de interromper o *build* se algum teste falhar, renomear arquivos e mover diretórios.

[06:49] Ele usa o *Ant* por por trás e também faz o trabalho lá que o Ivy fazia com a gestão das dependências das bibliotecas e *frameworks* que você quiser usar no seu projeto. O *Maven* veio justamente para substituir essas duas ferramentas. Ao invés de você usar duas ferramentas separadas, você poderia simplesmente utilizar o *Maven*.

[06:58] Tem outras ferramentas que são concorrentes, que são similares ao *Maven* - como o *Gradle*, o *SBT*, dentre outros que fazem algo parecido; mas o Maven é o mais popular, o mais famoso no mundo Java.

[07:12] Vai ser justamente o foco do treinamento. Vamos aprender a utilizar o *Maven* para resolver esses problemas. Problema de *build* de aplicação e de gerenciamento de dependências em Java.
O *Maven* resolve isso e esse vai ser o foco do curso. Vamos aprender como utilizar o *Maven* para resolvermos esses dois problemas clássicos em aplicações Java.

[07:33] No próximo vídeo, continuamos discutindo como funciona o Maven.

04 Ferramentas de build

Quais as principais vantagens de se utilizar o Maven em uma aplicação Java?

- Alternativa correta

  Maior cobertura de testes automatizados

- Alternativa correta

  Gerenciamento de dependências

  Alternativa correta! O Maven facilita bastante o trabalho de gerenciamento de dependências em uma aplicação Java.

- Alternativa correta

  Automatização do build

  Alternativa correta! O Maven facilita bastante o trabalho de build de uma aplicação Java.

05 Instalação do Maven

Transcrição

[00:00] Agora que já vimos a motivação do *Maven*, para que ele foi criado, já podemos partir para ver como é que funciona, como faço para usar, instalar, configurar e utilizar o *Maven*.

[00:12] Primeiro passo seria baixar o *Maven* no meu computador. Se você não tem o Maven instalado na sua máquina, você pode fazer o download. Bastaria entrar no seu navegador e pesquisar no Google: "download Maven" - vai cair no site do Maven neste link.

[00:35] No momento de gravação desse curso, a versão do Maven, a última, era `3.6.3`. Estando na versão 3 está valendo, a versão estável, que tem bastante recursos e que vamos utilizar no treinamento.

[00:46] Se você descer na página, te mostra os links para fazer o download. Tem o binário mesmo (“Binary tar.gz archive e Binary zip archive”) e tem o código-fonte (“Source tar.gz archive e Source zip archive”). O ideal é baixarmos pelo binário. Tem a versão `tar.gz` e tem a versão `.zip`. Vou clicar no link “apache-maven-3.6.3-bin.tar.gz.sha512”. Ele vai iniciar o download, mando ele abrir e vai baixar o Maven na máquina.

[01:07] É um arquivo `.zip`, então o Maven basicamente é um `zip`. Você descompacta ele em algum diretório. Vou escolher, mandar ele extrair no diretório *home* da minha máquina. Vou exibir os arquivos. Baixado o *Maven* na minha máquina. Ele é um `zip` que tem esses diretórios "bin", "boot", "conf" e "lib". Você joga isso para algum diretório do seu computador.

[01:29] Está o Maven! Como é que eu faço para rodar? Selecione o diretório “bin”, dentro tem um arquivo chamado “mvn”, esse é executável do Maven. Se quisermos executar o Maven temos que chamar esse executável pelo *prompt* de comandos.

[01:43] Estou no Linux, se você estiver no Linux, no Mac, use o terminal. Basta acessar o diretório onde você descompactou o Maven, no terminal vamos digitar: ‘cd apache-maven-3.6.3/’, depois aperte a

tecla “Enter”. Em seguida, na linha seguinte, coloque o comando `ls` e aperte a tecla “Enter” novamente.

[02:05] Em seguida digite o comando `cd bin/` e aperte a tecla “Enter”. Estou no diretório “bin”. Basta você rodar: `./mvn --version` e apertar a tecla “Enter”, só pra ele imprime qual é a versão do Maven - **3.6.3**, que é a versão que tinha baixado. O diretório do Maven, onde está descompactado o Maven na minha máquina, já identificou o Java instalado na minha máquina, pegou a variável de ambiente do Java, Java versão 11.

[02:28] Está pronto, instalado o Maven na sua máquina! A coisa mais fácil do mundo. Poderíamos executar os comandos, pedir para ele criar um projeto e fazer todo o trabalho do Maven pelo terminal.

[02:39] Para não termos que toda vez que quisermos executar o Maven entrarmos nesse diretório “/home/usuário/pasta Maven/bin”, poderíamos adicionar o Maven no *path* do sistema operacional e configurar como uma variável de ambiente esse diretório.

[02:57] Bastaria você fazer o seguinte: digitar 'mvn', independente do diretório onde você está, `mvn --version` e apertar a tecla “Enter”. No meu computador eu já tenho adicionado o Maven, configurado, e ele está adicionado no *path* do sistema operacional.

[03:20] Se eu digitar `mvn`, ele já sabe que é para pegar o executável que está no diretório “x”, já está configurado tudo certo. No meu caso ele está o "Maven home" está no `/usr/share/maven`. Já está configurado. Não preciso ficar entrando em nenhum diretório. Sempre que eu

quiser executar o Maven é só digitar `mvn` no terminal e pronto, ele já saberá que é para executar o Maven.

[03:41] Tem vários comandos do Maven. O `mvn --help`. Tem vários comandos que podemos utilizar. Temos, por exemplo, `mvn --projects` para listar os projetos, `mvn --settings` para atualizarmos as configurações do Maven, `update` etc.

[04:04] Vários comandos que poderíamos utilizar no terminal. Essa é uma das maneiras de utilizamos o Maven, via *prompt* de comandos, pelo terminal, bastando você ter baixado e descompactado em algum lugar do seu computador.

[04:19] Porém, essa maneira não é tão prática assim, a não ser que você trabalhe com a parte de infraestrutura e já esteja acostumado a trabalhar com o terminal. Outra alternativa é trabalharmos direto da IDE.

[04:31] As IDEs, hoje em dia, já têm uma integração nativa com o Maven. O *eclipse* é um desses casos. Por exemplo: se eu clicar em um projeto no meu Eclipe e clicar com o botão direito (clicar com o botão direito do mouse em “controle-products”), se ele não for aplicação Maven tem essa opção: “configure > Convert to Maven Project”. Ele já sabe converter uma aplicação para o Maven.

[04:51] Ou se eu for no *package explorer* do lado esquerdo da tela e apertar as teclas "Ctrl + N" e digitar "maven", perceba que já tem uma opção do Maven para ele baixar o Maven que está hospedado em algum diretório, criar um projeto do Maven, criar um módulo do Maven.

[05:04] Dentro da sua IDE provavelmente já tem um Maven embutido. Você pode trabalhar diretamente da IDE. Em vez de ter que ficar baixando, configurando variável de ambiente, rodando pelo terminal, você pode rodar de dentro da IDE. Já está com um projeto, já roda diretamente daqui - o que é muito mais prático e muito mais ágil.

[05:21] Essa vai ser a alternativa que vamos utilizar no curso, vamos executar o Maven de dentro da IDE. Eu estou usando o Eclipse, existem outras IDEs, você só precisa ver se já tem integração. Se não tiver, provavelmente tem como você instalar um *plugin* que já faz essa ponte, essa integração com o Maven.

[05:37] Instalação e configuração é bem simples, bem fácil e bem prático de você trabalhar. Espero que vocês tenham gostado. Fechamos essa primeira aula, onde conhecemos, entendemos a motivação do Maven, instalamos e vimos como funciona o Maven.

[05:51] No próximo vídeo, na próxima aula, vamos, de fato, começar a trabalhar em uma aplicação utilizando o Maven e entender como ele interage com a nossa aplicação Java.

06 Utilização do Maven

Quais as principais maneiras de se utilizar o Maven?

- Alternativa correta

  Via plugin do navegador

- Alternativa correta

  Pela interface Web

- Alternativa correta

  Pelo prompt de comandos

  Alternativa correta! O Maven pode ser utilizado via comandos do *prompt* ou *terminal*.

- Alternativa correta

  Pela integração com a IDE

  Alternativa correta! O Maven pode ser utilizado via integração com as `IDE`'s, como o *Eclipse*.

07 Faça como eu fiz

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula. Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.

Opinião do instrutor

-
-

Continue com os seus estudos, e se houver dúvidas, não hesite em recorrer ao nosso fórum!

08 O que aprendemos?

Nesta aula, aprendemos:

- As dificuldades de se realizar manualmente o build de uma aplicação Java;
- As dificuldades de se gerenciar manualmente as dependências de uma aplicação Java;
- O que é o Maven;
- Como instalar e executar o Maven.