02

Gerando o jar da aplicação

Transcrição

[00:00] Olá, bem-vindos de volta o curso de Spring Boot na Alura. Já terminamos a nossa API, já implementamos as funcionalidades, colocamos segurança, colocamos documentação com Swagger, a parte de monitoramento, *cache* e por último vimos sobre testes automatizados.

[00:21] E chegou a hora de colocar nossa aplicação em produção, que foi um assunto que também ficou faltando nos outros treinamentos, aprender como colocar no ar, colocar para rodar em produção, como gerar um pacote, um Build do projeto e fazer Deploy de uma aplicação com Spring Boot.

[00:37] Esse assunto acabou ficando de fora dos outros treinamentos e nessa aula é justamente esse o nosso objetivo, aprender a fazer um Deploy, gerar o pacote de uma aplicação com Spring Boot.

[00:48] Se abrirmos o nosso arquivo `pom.xml`, o arquivo de configurações do Maven, numa aplicação Web com Java, utilizando Maven, geralmente tem uma tag chamada *packaging* e nela vem a opção WAR. Porque quando terminamos de desenvolver a aplicação, o Build, o pacote é um arquivo `.war`, que é o arquivo que tem a pasta WebContent, web.xml, enfim.

[01:14] E nós pegamos esse WAR e passamos depois dentro de um servidor de aplicação, Tomcat, Jboss, Websphere, seja lá qual for o servidor de aplicação.

[01:22] Só que nesse caso vai ser diferente, o empacotamento não é um WAR, não tem essa propriedade. E quando não colocamos essa tag, o padrão é JAR.

Então quando gerarmos o Build do projeto vai ser gerado um arquivo `.jar`, uma aplicação Java tradicional.

[01:40] Lembra que não temos o Tomcat, não temos o servidor rodando. O servidor está embutido dentro da aplicação.

[01:45] Inclusive, se abrirmos a nossa classe `ForumApplication`, que usamos para rodar o projeto, é uma classe Java que tem um método Main. Então é uma classe que roda como se fosse uma aplicação Standalone, uma aplicação que você roda pelo prompt e ela abre uma janela, enfim.

[02:03] Então o processo de Deploy, de Build é um pouco diferente no Spring Boot. Mas é bem simples, bem tranquilo como quase todos os recursos do Spring Boot. Então vamos ver como fazer para gerar o Build do projeto e rodar esse projeto.

[02:16] Tem duas opções. Você pode fazer isso pelo próprio Eclipse, clicando com o botão direito em cima do projeto, indo na opção "Run As > Maven install". E ele vai fazer o Build utilizando o Maven.

[02:31] Ou você pode vir em "Maven Build..." e na janela que abrir você pode colocar quais são os *goals*, quais são as tarefas que o Maven vai rodar. Geralmente se passa "clean package". O "clean" vai recompilar as classes do projeto e o "package" vai gerar o pacote da aplicação. Então pode ser dessa forma também.

[02:53] Ou, se você preferir, também pode ser pelo prompt de comandos. Você pode entrar no diretório do projeto. Eu já estou no diretório raiz, que tem o `pom.xml`. Se você tiver o Maven instalado na sua máquina, tiver o comando `mvn`, você pode simplesmente rodar o comando `mvn clean package`, isso dentro do diretório raiz do projeto, do diretório fórum.

[03:24] E dá na mesma. Ou por aqui ou pela sua IDE, tanto faz. O Maven vai ser chamado e vai iniciar o processo de compilação e de empacotamento do

projeto.

[03:37] Na verdade não tem muito a ver com Spring Boot, é mais a parte do Maven. Eu o estou mandando gerar o Build da aplicação e gerar um pacote. A questão é que uma aplicação por Spring Boot por padrão não gera um arquivo `.war`. Ela gera um arquivo `.jar`.

[03:52] E assim que ele terminar o Build nós veremos como fazer para rodar essa aplicação, porque numa aplicação web tradicional nós geramos um WAR, copiamos esse WAR e jogamos dentro da pasta do servidor de aplicação.

[04:06] Se for o Tomcat tem uma pasta chamada “web apps” dentro do diretório do Tomcat. E cada servidor de aplicação tem um diretório específico onde você joga o WAR. E quando você inicializa o servidor ele detecta que tem esse WAR e já faz o Deploy e a inicialização desse projeto dentro do servidor.

[04:24] Mas no Spring Boot é diferente, porque o servidor está embutido. Então é o contrário: não é aplicação que jogamos dentro do servidor, é o servidor que está dentro da aplicação. Então o processo é um pouco diferente.

[04:36] E como vocês podem observar, quando mandamos o Maven rodar o comando `package`, ele tem um passo anterior, que é de rodar os testes. Então o Maven vai detectar que no nosso projeto tem os testes automatizados, aqueles que vimos na última aula de exemplo, o do Controller e do Repository, e vai executar os testes.

[04:55] Então inclusive ele está subindo o servidor. Ele está rodando o teste da mesma maneira que rodamos dentro do Eclipse, da nossa IDE.

[05:06] E caso algum teste falhe ele interrompe o Build, ele não vai gerar o JAR do projeto. E se todos os testes passarem ele gera o Build do projeto e tudo vai funcionar corretamente. Então é até uma segurança para nós, para garantir que o pacote foi gerado sem erros.

[05:25] Então ele gerou o Build, apareceu a mensagem `BUILD SUCCESS`. Subindo um pouco, ele mostra a parte dos testes: quatro testes rodaram, nenhum falhou, não deu nenhum erro e nenhum teste foi pulado. Então os testes passaram.

[05:40] Limpando a tela, se dermos um `ls` para listar os diretórios, tem uma pasta chamada `target`. Vamos entrar na pasta `target`, vamos dar um `ls`, e aparece o JAR do projeto.

[05:51] Ele gera uns arquivos a mais, mas não precisa se preocupar. O que interessa é o arquivo `forum-0.0.1-SNAPSHOT.jar`. Ele só coloca o `0.0.1-SNAPSHOT` porque esse é o padrão do versionamento que está no `pom.xml` do nosso projeto.

[06:07] Se abrirmos o `pom.xml`, tem o `<version>0.0.1-SNAPSHOT<version>`. Por padrão ele coloca o nome do artefato e coloca esse sufixo. Depois eu mostro como tirar esse sufixo e mudar o nome do JAR que é gerado pelo Maven.

[06:27] Então esse é o JAR que você vai passar para o pessoal da infraestrutura para eles rodarem o projeto. Só que, de novo, como não é um WAR, é um JAR, como eles vão fazer para rodar esse JAR?

[06:37] É agora que entra a diferença. Então é um JAR, você roda como se fosse um aplicação Java Standalone. Para rodar um JAR você executa o comando com o nome do arquivo: `java -jar forum-0.0.1-SNAPSHOT.jar`.

[06:54] Então é um comando que eles vão executar no prompt, no terminal. E dá um “Enter”. Feito isso ele vai carregar, vai procurar qual é a classe Main do nosso projeto. Ele sabe que é a classe `ForumApplication` e ele vai rodar da mesma maneira, como se estivéssemos rodando dentro do Eclipse, só que rodando pelo terminal.

[07:14] Um cuidado quando vocês forem fazer isso: verifica se o projeto não está rodando no Eclipse. No meu está parado, ele não está escutando. Porque

senão na hora que ele for tentar inicializar no terminal vai dar conflito, porque ele vai usar a porta 8080, mas se já estiver rodando no Eclipse a porta 8080 já vai estar ocupada.

[07:33] Então lembra de encerrar o processo no Eclipse caso ele esteja executando. No meu caso não está executando, então ele vai inicializar. Ele vai carregar as classes, vai ler os Repositories, vai ler as configurações do banco de dados, vai rodar o projeto da mesma maneira.

[07:52] Assim que ele terminar o projeto está *deployado*, digamos assim. No ambiente de produção é assim que você roda o projeto. Então para gerar o JAR é via Maven, tanto faz se pela IDE ou pelo prompt de comandos, `mvn clean package`.

[08:13] Gerou o JAR, você vai passar para a equipe que cuida da parte de infraestrutura. E o que eles vão fazer no servidor, na máquina onde eles vão rodar é executar esse comando `java -jar` seguido do nome do arquivo e `-jar`. Simples assim, um comando e tudo funciona.

[08:31] Ele *startou* normalmente, já está rodando o projeto localmente, na porta 8080.

[08:37] Perceba que é bem simples de gerar o Build do projeto e rodar pelo prompt o JAR da aplicação com Spring Boot. É um negócio bem simples, bem fácil, sem complicação. Então você roda o JAR e ele sobe o Tomcat.

[08:51] Então essa era a aula de hoje, espero que vocês tenham gostado, tenham aprendido como fazer para gerar o Build do projeto, como fazer para rodar o projeto pelo prompt de comandos. Espero vocês na próxima aula. Em abraço e até lá.