04

Externalizando senhas com variáveis de ambiente

Transcrição

[00:00] Olá, bem-vindos de volta ao curso de Spring Boot na Alura. Agora que aprendemos como fazer para gerar o Build do projeto, gerar o JAR e executá-lo pela linha de comando, podemos partir para o próximo passo.

[00:14] Uma coisa importante que eu não falei no último vídeo: nós estamos simulando o Build para gerar um pacote da aplicação para a equipe de infraestrutura fazer o Deploy e coloca-la em produção. Mas e as configurações? Quais configurações ele vai pegar?

[00:32] Não sei se vocês lembram, mas no nosso diretório `src/main/resources` temos o `application.properties`. Chegamos a criar o `application-test.properties` por causa dos testes automatizados. Mas por padrão, no ambiente de produção se não definirmos qual é o *profile* ativo do momento ele vai pegar o *profile* ativo *default*, ele vai carregar o `application.properties`.

[00:56] Então ele vai ler essas configurações, só que essas configurações são do banco de dados de desenvolvimento. Então faltou ter essa separação de criar configurações específicas para o *profile*, o ambiente de produção.

[01:14] É isso que vamos fazer nessa aula e ver como funciona isso na prática. A ideia é parecida com o que fizemos nos testes. Nós criamos um arquivo application.properties específico para os testes.

[01:25] Então vou duplicar o application.properties, só que agora o nome vai ser application-prod.properties. Esse "prod" seria uma abreviação para *production*, um ambiente de produção.

[01:36] Agora abrimos o application-prod.properties. Como eu o copiei a partir do application.properties, os dois vão estar iguais, com todas as configurações. E a ideia é sobrescrever as configurações que queremos que sejam diferentes no ambiente de produção.

[01:53] Vamos pegar o banco de dados como exemplo. Imagina que eu não quero que o banco de dados seja o `jdbc:h2:mem:alura-forum`, quero que seja outro. Estamos usando o H2, mas imagina que em produção é o MySQL; ou até o próprio H2, mas com o nome do banco de dados diferente; o usuário e a senha diferentes, enfim. Essas propriedades eu quero modificar.

[02:12] Só que tem outro ponto importante: estamos passando dados sensíveis, de senha, de acesso a recursos protegidos diretamente no arquivo de propriedades da nossa aplicação, e isso não é uma boa prática de segurança. Então geralmente se passa essas configurações via variáveis de ambiente.

[02:30] Então vamos aprender como passar essas configurações via variáveis de ambiente, para não ficar expondo isso no próprio código da aplicação.

[02:39] Vamos colocar dessa maneira: `${}` e dentro passamos qual é o nome da variável de ambiente, geralmente em letra maiúscula. Então `${FORUM_DATABASE_URL}`. Então eu quero que a propriedade seja lida de uma variável de ambiente chamada `FORUM_DATABASE_URL`.

[03:00] E eu vou fazer a mesma coisa para o usuário, só que ao invés de URL é `USERNAME`, então `${FORUM_DATABASE_USERNAME}`. E para senha a mesma coisa: `${FORUM_DATABASE_PASSWORD}`.

[03:10] A partir de agora essas propriedades não são fornecidas diretamente no código da aplicação, elas serão lidas pelas variáveis de ambiente que serão configuradas no ambiente de produção, no servidor de produção.

[03:22] E uma última configuração, o *secret* do JSON Web Token. Então vou passar também: `${FORUM_JWT_SECRET}`.

[03:34] Pronto, acabei de criar o *profile* de produção, o arquivo `properties` para o ambiente de produção e substituí as informações, só que em vez de deixar fixo no código eu estou puxando de variáveis de ambiente.

[03:48] Vou aproveitar e fazer outra mudança. Quando geramos o JAR ele pegou o nome `0.0.1-SNAPSHOT`, ficou um nome meio estranho. Caso você queira mudar o nome do JAR basta você vir na *tag* `<build>`, pode ser antes ou depois do *plugin*, e colocar uma *tag* chamada `<finalName>`. E eu vou colocar `forum`, então `<finalName>forum</finalName`. Agora o nome do arquivo JAR vai ser `forum.jar`. Ele não vai colocar aquele `0.0.1-SNAPSHOT`.

[04:17] E pronto, essas são as mudanças que eu fiz. Criei um novo arquivo Properties e mudei o `pom.xml`. Agora já podemos voltar no terminal e fazer um novo Build da aplicação utilizando o Maven. Lembra que era só acessar o diretório fórum, o diretório raiz aplicação e digitar `mvn clean package`.

[04:34] Pode ser assim, pelo terminal se você tiver o Maven instalado na máquina ou pelo próprio Eclipse. Nesse caso clicando com o botão direito no projeto, e na opção “Run As > Maven build...”. No campo “Goals” da caixa de informações que vai abrir coloca “clean package” e clica no botão “Run”.

[05:04] Então se fosse pelo Eclipse seria esse processo. Vou fechar porque já coloquei para rodar pelo próprio prompt de comandos.

[05:12] E é aquele esquema, agora já não tem mais a ver com Spring Boot, isso é o Maven. Então o Maven vai compilar as nossas classes, compilar as classes de teste, executar os testes automatizados e se tudo passar, fazer o Build, gerar o JAR do projeto. E dessa vez ele vai gerar um JAR chamado `forum.jar`, não vai ter aquele sufixo `0.0.1-SNAPSHOT`.

[05:36] E dessa vez, dentro desse JAR vai ter o arquivo `application-prod.properties`. E na hora de rodar, simulando que seria alguém executando no servidor de produção, vamos ver como passar para o Spring qual é o *profi* ativo do momento e como passar aquelas informações que estão no

application.properties, as informações das variáveis de ambiente; geralmente é assim que o pessoal faz por questões de segurança.

[06:08] Ele imprimiu que pegou o *profile* de teste. Lembra que aprendemos como passar qual é o *profile* ativo do momento? Vou voltar no Eclipse para mostrar para vocês.

[06:19] Vou abrir alguma classe de teste, um Controller, por exemplo. Nós forçamos o *profile* ativo no momento com a anotação `@ActiveProfiles` . E no teste faz sentido, se eu estou rodando aqui o teste, eu consigo passar para o Spring qual é o *profile* ativo do momento quando ele for rodar o teste.

[06:38] Outra maneira que vimos de como passar o *profile* ativo do momento era clicando com o botão direito em cima do projeto, indo na opção "Run As > Run Configurations..." . E tinha um campo em que passávamos uma variável.

[06:58] Nós escolhemos o "Java Application > Forum Applications" no menu à esquerda. E em "Arguments" tem o campo VM arguments, escrito `-Dspring.profiles.active=dev` . Sempre que rodamos a classe `ForumApplicatoin` dentro do Eclipse passamos essa variável de ambiente para dizer que o *profile* é o Dev.

[07:21] É exatamente isso que temos que fazer, só que pelo terminal. Eu vou até copiar o `-Dspring.profiles.active=dev` . Na hora que eu for rodar o JAR eu passo essa variável para o Spring para ele saber qual é o *profile* ativo do momento quando eu rodar pelo prompt de comandos.

[07:35] Então o pessoal da infraestrutura que vai pegar esse JAR e rodar no ambiente de produção tem que ter esse cuidado. Eles não podem simplesmente rodar o `Java -jar forum.jar` . Senão ele não vai saber qual é o *profile* ativo no momento.

[07:49] Então temos que rodar `Java -jar -Dspring.profiles.active=prod forum.jar` . Então na hora de rodar pelo prompt de comandos passamos aque…

parâmetro e o Spring sabe que estou rodando pelo prompt de comandos e o *profile* ativo do momento foi passado como um argumento, como um parâmetro da JVM.

[08:11] É dessa maneira que o pessoal da infraestrutura tem que fazer quando eles forem rodar o projeto em produção. Seja rodando manualmente, ou seja, configurando alguma máquina de integração contínua, algum servidor do tipo Jenkins, enfim.

[08:26] O Build foi rodado com sucesso. Vamos entrar no diretório `target`, lembra que é onde ele gera. Vou dar um `ls` e aparece o `forum.jar`. Ele gerou o JAR com o nome correto agora, conforme eu configurei no `pom.xml`. E agora vamos rodar aquele mesmo comando `Java -jar -Dspring.profiles.active=prod forum.jar`.

[08:55] Então esse é o comando que temos que rodar para subir com *profile* de produção. Se eu der um "Enter" agora e o mandar rodar, é para dar problema. É para ele imprimir que está rodando no *profile* de produção. Só que é para dar problema porque eu não passei aquelas informações do application.properties.

[09:14] Ele imprimiu. Está lendo o *profile* ativo como prod, conforme eu passei na variável.

[09:21] Agora, quando ele for carregar o application-prod.properties, vão ter aquelas configurações que era para puxar das variáveis de ambiente. Mas eu não passei as variáveis de ambiente, então o Spring vai dar erro. Ele não consegue achar o *data source*, não consegue achar o *secret* do JWT. E deu *exception*.

[09:43] Se subirmos um pouco, tem o `java.lang.RuntimeException`. Ele falou que não conseguiu identificar o `${FORUM_DATABASE_URL}`. Ele achou que isso era a URL, porque ele não achou essas variáveis.

[09:57] E agora vem essa questão, como eu passo uma variável de ambiente para o Spring, para ele substituir no application-prod.properties? Tem duas maneiras principais de você fazer isso.

[10:08] Você pode exportar uma variável de ambiente na máquina. Eu deixei salvo num arquivo, só para não ter que digitar. No ambiente Linux, ou no Mac, que é o meu caso, é só você rodar da seguinte maneira: `export` seguido do dome da variável, `=` e qual é o valor. Podemos exportar cada uma dessas variáveis dessa maneira.

[10:30] Vou copiar tudo e vou colar no meu terminal. Vamos só garantir que ele imprimiu. Vou dar um `echo` para ele imprimir: `echo $FORUM_DATABASE_URL` . E ele não imprimiu.

[10:51] Vou ver se digitei alguma coisa errada. Achei o erro, eu coloquei `JDBC` quando na verdade era `DATABASE` . Vou substituir e colar de novo no terminal. Vou dar um `echo` agora. E imprimiu.

[11:30] Então já estão configuradas as variáveis de ambiente. Vou tentar rodar de novo o JAR, passando o *profile* de produção. Agora é para rodar tudo normalmente porque eu passei as variáveis de ambiente. Então vamos ver se ele vai conseguir pegar.

[11:45] Enquanto isso, tem outra maneira que eu deixei salva também para agilizar, para não perder muito tempo. Em vez de exportar dessa maneira, pode ser no próprio comando `java -jar` . Pode rodar com esse comando, seguido de `-D` e do nome da variável igual ao valor.

```
java -jar -DFORUM_DATABASE_URL=DATABASE:h2:
mem:alura-forum
-DFORUM_DATABASE_USERNAME=sa
-DFORUM_DATABASE_PASSWORD=
-DFORUM_JWT_SECRET=123456 forum.jar
```

COPIAR CÓDIGO

[12:15] Você pode passar cada uma das variáveis de ambiente via parâmetros, no comando de rodar a aplicação. Então ou você exporta dessa maneira ou você passa na linha de comando como parâmetros, da mesma maneira que passamos o *profile* ativo do momento. Então fica seu critério.

[12:35] Vou corrigir uma parte que era para ser `jdbc`. Não deu problema, mas na hora em que ele for conectar no banco ele vai perder. Vou parar.

[12:49] Então esse é o jeito de exportar as variáveis, e vamos rodar dessa maneira também. Só faltou passar o *profile*: `java -jar -Dspring.profiles.active=prod -DFORUM_DATABASE_URL=jdbc:h2:mem:alura-forum`. Sempre com o nome da variável igual ao valor.

[13:12] Então é desse jeito que você vai passar nas variáveis, como o nome da variável igual ao valor. Só cuidado que é `-D` maiúsculo e o nome é grudado, não tem espaço.

[13:21] Então dessa maneira nós passamos via linha de comando. Vou limpar e vou colar. E vai dar na mesma. Agora em vez de passar pelo *environment*, sendo exportadas as variáveis, eu passo direto no comando de rodar o arquivo `.jar`.

[13:40] Então essas são duas maneiras de você passar as variáveis de ambiente para o Spring. Ou exportado no sistema operacional ou direto no comando, no prompt. Vai depender de qual é o padrão usado pelo time que cuida da infraestrutura de rodar o projeto em produção.

[13:57] Dessa maneira, acabamos de aprender como fazer para ter o *profile* de produção, ter o arquivo `properties` de produção separado e não ficar colocando senhas, dados sensíveis no application.properties, porque isso é uma má prática de segurança. O ideal é passar via variáveis de ambiente.

[14:15] Vimos no Eclipse, no arquivo `properties` como fazer para passar uma variável de ambiente ao invés de passar o valor fixo. E na hora de rodar ou você

exporta as variáveis de ambiente ou na hora de rodar o comando você passa via parâmetro da JVM. Então são duas maneiras de preencher essas variáveis de ambiente.

[14:38] E funcionou, ele rodou, inicializou a aplicação normalmente. Só deu um erro por causa do Actuator, que eu não estou rodando o Spring Boot Admin.

[14:56] Mas inicializou, apareceu `Started ForumApplication`, então ele subiu a aplicação normalmente. Ele leu as variáveis de ambiente.

[15:03] Espero que vocês tenham gostado desse vídeo e aprendido como fazer para tratar essa questão de ambientes e de variáveis de ambientes para não ficar colocando senhas em arquivos do projeto, porque isso é uma má prática de segurança.

[15:14] Vejo vocês no próximo vídeo. Um abraço e até lá.