03

Estrutura de um projeto com Maven

Transcrição

[00:00] No último vídeo aprendemos a criar um projeto no Eclipse utilizando o Maven. Eu até comentei por alto sobre a estrutura de diretórios, mas no vídeo de hoje eu quero focar especificamente nisso.

[00:13] Todo projeto Maven, por padrão, tem essa estrutura de diretórios.

[00:16] Ele possui esses quatro *source folders*, "src/main/java", "src/main/resources", "src/test/java", "src/test/resources" e o arquivo "pom.xml", onde ficam as configurações. Dentro de cada *source folder* desse vai determinados tipos de arquivos.

[00:34] Isso é justamente para você não ficar quebrando a cabeça, perdendo tempo pensando onde que você coloca aquele arquivo de configuração e onde que coloca o tipo de classe. Já está tudo aí pré-determinado.

[00:44] Aqui no "src/main/java", vamos dizer assim, é o principal *source folder*. Aqui é onde você vai criar as classes, interfaces, *enum's*, todo código-fonte de código Java da sua aplicação.

[00:55] Por exemplo: vamos criar uma classe aqui. "src/main/java > Class > Next > New class > Name: 'Produto' > Package: 'br.com.alura.loja' > Finish" e a classe vai se chamar "Produto".

[01:08] Aqui a classe Java vem os seus pacotes, classes, interfaces, toda a parte de Java mesmo fica aqui.

[01:18] Ao invés de você ficar perdendo tempo discutindo, pensando onde colocar código Java, saiba que ele deveria fica aqui no “src/main/java”. Esse *source folder*, o objetivo dele é esse, ficar seus códigos Java.

[01:30] “Rodrigo, eu tenho páginas HTML, arquivo de configuração! ” Não é aqui que vai ficar isso. No “src/main/java” fica apenas código Java, classes, interfaces e *enums*. No geral, esses três tipos de código.No “src/main/resorces”, arquivos de configuração.

[01:48] “Eu estou usando Hibernate, onde eu coloco o `persistence.xml` , o `hibernate.sfg.xml` ? Estou usando o *spring build*, onde fica o `application.properties` , arquivo *Bean Validation*, `Messages.properties` , `Validation message.properties` ?”

[02:02] Tudo que não é classe Java, mas que precisa estar no diretório acessível ali para as suas bibliotecas ou para o seu código-fonte, você coloca no “src/main/resorces”.

[02:15] Aqui, por exemplo, você poderia criar um arquivo de propriedades. Clique em “ "src/main/resorces > New File > Next> File name: messages.properties > Finish”. Arquivo de propriedades, arquivo de configurações dos seus *frameworks*, dependendo do projeto - se for uma aplicação com *Spring boot* e as páginas vão ser no *server-side*, as páginas vão ficar dentro do “src/main/resorces”. Você cria uma pasta para colocar as páginas.

[02:39] Arquivos estáticos de CSS java script, no caso do *Spring boot* ficam aqui dentro do “src/main/resorces”. Esses arquivos que não são código Java. Tudo bem?

[02:48] No “src/test/java” - o nome já diz, códigos de teste. Você quer criar uma classe de teste para sua classe `Produto` ? "Ctrl + N > Junit > JUnit Test Case”. Perceba que o Eclipse até detecta que a sua aplicação é uma aplicação com

Maven e ele já coloca aqui automaticamente em “source folder”: “loja/src/test/java”.

[03:06] “Opa! É uma classe de testes com `JUnit`, então vou colocar no ‘src/test/java’!” E depois clicar em “Finish”. Se você criar a classe, aqui ele vai só pedir para adicionar o JUnit do projeto - porque não tinha. Ele já cria exatamente lá, “src/test/java”.

[03:20] Aqui ficam as classes de testes com JUnit, TestNG ou qualquer biblioteca de testes que vocês for utilizar no seu projeto. Fica aqui nesse diretório.

[03:30] Se você estiver usando alguma biblioteca utilitária para facilitar os testes, Selenium, Dbunit etc., provavelmente essa biblioteca pode precisar de alguns arquivos de configuração. Como são configurações específicas para os testes, o ideal é você colocar aqui no “src/test/resources” para não misturar com os recursos de produção, que são utilizados pela sua aplicação. Aqui é somente utilizado para os testes automatizados. Tudo bem?

[04:00] O “pom.xml” é aquele arquivo onde ficam as configurações da sua aplicação, as configurações do *build*, das dependências e dos *plugins* que você quiser adicionar no seu projeto. Coisas que veremos ao longo do treinamento.

[04:13] Nesse vídeo era só para dar uma passada mais detalhada em cada um desses diretórios e explicar o objetivo de cada um deles. “Rodrigo, mas eu não quero usar essa estrutura de diretórios, eu quero ter o diretório "src" e o diretório "test", apenas esses dois *source folders*, posso fazer desse jeito? ”

[04:31] Pode! Você pode vir aqui no “pom.xml”, tem algumas *tags* aqui para você personalizar o *source folder* do seu projeto. Porém, não é uma boa prática e não é recomendado. Toda aplicação Maven segue essa estrutura de diretórios padrão, isso é algo universal. Todo projeto, toda empresa em qualquer lugar do mundo faz desse jeito.

[04:53] Embora você consiga trocar esse padrão, o ideal é que você siga esse padrão já que isso é universal e é utilizado assim no mundo inteiro. Para que ficar reinventando a roda, se já foi definido que dessa maneira é a maneira padrão universal? Vamos seguir do jeito padrão. Não é uma boa prática você trocar essa estrutura de diretórios.

[05:12] Esse era o objetivo da aula de hoje, fazer essa discussão sobre a estrutura de diretórios de uma aplicação Maven. Vejo vocês no próximo vídeo, onde vamos aprender outros recursos do Maven. Um abraço e até lá!