04

Criando a ConnectionFactory

Transcrição

[00:00] Fala, aluno. Tudo bom? Nas últimas aulas nós vimos como recuperar uma conexão no nosso banco de dados e como listar os produtos que foram inseridos na nossa tabela. Aparentemente, o código, ele está ok. Se eu mandar executar a nossa classe `TestaConexao`, ele vai retornar o que nós esperamos, que é uma string informando que a conexão foi fechada. E a `TestaListagem` vai listar todos os produtos que temos na nossa tabela.

[00:36] Se analisarmos bem os dois códigos, nós temos aqui, em `System.out.println("Fechando Conexão! !");` um ponto de atenção. Quando eu recupero a conexão, eu estou utilizando todo esse trecho de código aqui, do `Connection connection = DriverManager`, chamando o Driver manager, o gerenciador de driver, chamando o método `.getConnection` e ainda passando a string de conexão. Na `TestaListagem`, a mesma coisa.

[01:04] Qual é o ponto de atenção aqui? Se um dia eu mudar a minha senha do banco de dados, de root para 123, eu vou ter que mudar nessa classe `TestaListagem` e nessa classe `TestaConexao`. Se eu tiver uma classe Testa Inserção, eu vou ter que trazer todo esse código da `TestaConexao` para a Testa Inserção. Então nós vemos que vai aumentando a complexidade, aumentando o risco de quebra do nosso código.

[01:30] Então, para isso, para refatorarmos o nosso código, eu vou criar uma nova classe chamada `CriaConexao`. Dentro dessa classe `public Class CriaConexao` eu vou ter um método, que vai retornar uma Connection e vou

botar aqui `public Connection recuperarConexao` . O nome fica a critério de vocês, o que vocês acharem melhor. O meu aqui vai ser `recuperarConexao` .

[02:01] E toda essa linha de código aqui, a `Connection connection = DriverManager` em `TestaListagem` , eu posso retirar e voltar as linhas para `CriarConexao` . Eu vou precisar adicionar o `throws SQLException` no `CriaConexao` e vou ter que colocar um `return DriverManager` , que ele retorna uma Connection.

[02:20] O código, ele vai ficar aqui, nessa minha `CriaConexao` . E, na `TestaListagem` , eu não preciso mais passar todo aquele código, eu sou vou precisar instanciar aqui a minha classe, que eu acabei de criar: `CriarConexao criaConexao = new CriaConexao();` . Vou receber uma `Connection connection = criaConexao.recuperarConexao();` do método `.recuperarConexao();` .

[02:53] Já demos uma melhorada no código aqui. Vou remover o `import java.sql.DriverManager` que não está sendo mais utilizado, que é do Driver manager. A mesma coisa eu vou fazer na minha `TestaConexao` . Vou apagar o código, eu instancio a classe que acabamos de criar, a `CriaConexao criaConexao = new CriaConexao();` .

[03:16] E chamo o método `.recuperarConexao();` , que vai devolver uma `Connection connection = criaConexao.recuperarConexao();` . Vamos novamente retirar esse `import java.sql.DriverManager` , que não está sendo mais utilizado. Se eu mandar executar, o retorno do código vai ser a mesma coisa que tínhamos visto no começo da aula. Ele vai informar, é a mesma string `("Fechando Conexão! !")` .

[03:47] Se eu executar a `TestaListagem` , nós vamos ver a mesma lista que nós tínhamos visto anteriormente. Então vemos que agora o código, ele ficou encapsulado, ele ficou centralizado em uma única classe, a `CriarConexao` . Se eu precisar mudar alguma coisa na minha string de conexão, se eu precisar

mudar o meu banco de dados de MySQL para SQL, eu só vou precisar mexer nessa nossa classe `CriaConexao`.

[04:18] Só que, o mais interessante disso tudo, é que esse conceito ele não foi criado por mim agora. Nós temos, no Java, um Design pattern, ou seja, um padrão de projeto, que se chama Factory Method. Esse Factory Method tem por objetivo centralizar, encapsular um código que vai criar um objeto. Ou seja, essa classe `CriaConexao`, ela vai ser uma fábrica de conexões.

[04:52] Então toda vez que essa classe for chamada, é porque eu quero uma conexão. Então, esse Factory Method, ele tem vários outros conceitos aplicados nele, vocês podem pesquisar que é bem interessante. Para ficar mais claro aqui, eu vou refatorar o nome da minha classe clicando sobre "CriaConexao.java" com o botão direito para selecionar "Refactor > Rename..." e vou botar como uma "ConnectionFactory" no campo "New Name" da janela "Rename Compilation Unit", porque é isso o que ela é.

[05:25] Ela é uma fábrica de conexões, sempre que eu precisar de uma conexão, eu vou chamar a minha `ConnectionFactory`. Com isso, agora o nosso código fica bem bacana, porque além de nós centralizarmos o nosso código, encapsulamos essa instanciação de uma conexão, nós já estamos utilizando um padrão que é utilizado pelos desenvolvedores da linguagem Java.

[06:04] Então, só para ficar mais bonito aqui, eu vou mudar o nome da minha variável de `CriaConexao` para `ConnectionFactory` no restante do código, e agora nós temos a nossa `ConnectionFactory` funcionando normalmente. Só para testar, vou executar mais uma vez as duas classes e nós vamos ver que o resultado foi o mesmo. Então é isso, aluno. Espero que vocês tenham gostado e até a próxima aula.