06

Usando o try-with-resources

Transcrição

[00:00] Fala, aluno. Tudo bom? Anteriormente nós vimos como fazer o controle das nossas transações de forma mais fina, de forma mais bonita, por assim dizer. Nós vimos que uma transação, ela vai envolver uma ou mais operações, então no nosso caso, é a adição de dois produtos na mesma transação.

[00:24] E com aquela velha história, ou ele vai adicionar os dois produtos ou ele não vai adicionar ninguém, porque se ele der alguma exceção, que é caso do nosso exemplo do `if`, ele vai desfazer a operação da primeira inserção, e ficamos com uma transação bem certa, conforme o que esperamos de uma transação mesmo.

[00:55] Só que uma coisa que nós vamos começar a reparar no código, são por exemplo esses Statements, que eles devem ser fechados. Nós temos uma conexão que quando abrimos, deve ser fechada. Temos um `PreparedStatement` que quando abre, ele deve ser fechado. Tem o `ResultSet` também na mesma situação. E hoje nós vemos que é fácil de esquecer de explicitar o close desses Statements.

[01:29] Esse é um problema, porque você esquece, então ele depende de ser explícito, esse fechamento. Então, no caso de uma conexão, ele pode abrir várias conexões e eu esquecer de fechar, isso pode trazer problemas de performance no futuro, pode trazer problemas de banco de dados no futuro. Esse é um problema.

[01:52] Só que nós temos outro problema também. Por exemplo, no nosso `catch`, que recuperamos a exceção caso aconteça alguma coisa na transação, ele vai fazer o `rollback` dentro do `catch`. Mas, por exemplo, dependendo de como eu tratar a minha exceção, ele pode nunca sair desse `catch`, pode nunca fechar o close.

[02:10] Então eu posso até ter nunca esquecido de fechar a minha `connection`, só que o fato de eu tratar errado a minha exceção dentro do meu `catch`, ele pode nunca sair do `catch` e eu posso ter o mesmo problema que eu teria se eu tivesse esquecido de fechar a minha conexão. Então, para isso, eu tenho, a partir do Java 7, um recurso que chama "try-with-resources", que é o try com recursos.

[02:39] Esse try com recursos, ele serve para nos auxiliar nos fechamentos desses Statements que precisamos explicitar que sejam fechados. Se verificarmos no `PreparedStatement`, ele vai estender o Statement e também que o Statement vai estender um `AutoCloseable`. Esse `AutoCloseable` é o que tem o nosso método `close()`.

[03:07] Com o "try-with-resources", quando eu abro o `try`, eu por exemplo, posso colocar o `PreparedStatement stm = connection.prepareStatement("INSERT INTO PRODUTO (nome, descricao) VALUES (?, ?)", Statement.RETURN_GENERATED_KEYS);` entre os parênteses do `try()`.

[03:26] Com essa ação que eu acabei de fazer, e pelo o `PreparedStatement`, ele estender de `AutoClosable`, eu não preciso explicitar o fechamento desse `PreparedStatement`, porque ao final do `try`, quando eu terminar a execução do meu bloco `try`, ele está garantido que o Statement será fechado.

[03:51] Então, com essa ação de colocar o bloco do `PreparedStatement` entre os parênteses do `try()`, eu não preciso mais me preocupar agora em ficar explicitando os meus fechamentos. Então a mesma situação nós podemos fazer no `ResultSet`. Invés de eu fechar o `ResultSet` explícito em

`rst.close();`, no final do bloco, eu vou abrir antes de `ResultSet` um try com recursos.

[04:21] E eu só preciso substituir o `rst.close();` por uma chave, por um fechamento da minha chave, do meu bloco do `try`. Aqui eu já estou garantindo que o recurso do `ResultSet` será fechado pois ele estende também o recurso `AutoCloseable` do `PreparedStatement`. Podemos também fazer a mesma coisa com a nossa `Connection connection = factory.recuperarConexao();`.

[04:41] Então vou abrir `try(Connection connection = factory.recuperarConexao());{`, os parênteses do meu `try`, vou botar o recurso dentro do `try`, que é o recurso da `Connection`, da minha conexão e no `connection.close();`, no seu fechamento, invés de eu explicitar o `.close`, eu só vou apagar a linha e fechar as chaves. Deixa eu abrir a chave no `try` do `Connection`.

[05:06] Pronto, com isso, agora eu tenho um código mais enxuto, porque nós temos menos linhas de código, e nós não temos como - quer dizer, não é que nós não temos, é que fica mais fácil, porque não precisamos agora explicitar esses fechamentos e não ficamos com aqueles problemas de um esquecimento de fechar uma conexão.

[05:34] Ou então até menos na hora de tratar uma exceção ele não sair do bloco, enfim, aqueles problemas que nós estávamos falando no começo da aula. Então o “try-with-resources” vai servir para isso. Além de ficar com um código mais bonito, ele vem com esse benefício de tirar da nossa responsabilidade explicitar esses fechamentos. Então é isso, pessoal. Espero que tenham gostado e até a próxima aula.