06

Implementando as controllers

Transcrição

[00:00] Olá, aluno. Tudo bom? Agora que nós entendemos o nosso cenário de como a nossa aplicação *view* vai se comunicar com as nossas classes de negócio, que é a nossa Controller, a nossa DAO, chegou a hora de colocarmos a mão na massa e terminar o desenvolvimento das nossas classes que não estão funcionais, que na nossa situação é a `CategoriaController` e a `ProdutoController`.

[00:24] Antes, eu gostaria de lembrar vocês para tomarmos cuidado de não trabalhar no projeto que estávamos utilizando no curso, que é o "loja-virtual-repository", para não correr o risco de fazermos alterações nele achando que estamos fazendo alterações no "loja-virtual-view-repository". Recomendo novamente que fechemos esse projeto "loja-virtual-repository".

[00:42] Para quem não sabe como fechar, é só clicar com o botão direito do mouse em cima do projeto, tem uma opção "Close Project". Fechando o projeto, ele continua no nosso Workspace, mas não conseguimos acessar o projeto. Voltando agora para o nosso "loja-virtual-view-repository", nós vamos começar a mexer no `CategoriaController`.

[01:07] O `CategoriaController`, ele só vai ter um método de `listar()`, que é o responsável por popular aquela nossa `ComboBox` com as categorias, para podermos incluir um novo produto. Essa lista, ela vai ter que chamar de alguma forma a nossa DAO e aí sim fazer a comunicação para devolver a lista de categorias para a nossa *view*. Só que, se vocês lembram, as nossas DAOs, elas esperam uma Connection no momento que instancia o objeto.

[01:44] Então o responsável por fornecer essa Connection vai ser a nossa `CategoriaController`. Para isso, nós vamos usar o construtor. Então, no momento em que a nossa *view* chamar o `CategoriaController`, instanciar o `CategoriaContronller`, a `CategoriaController` já recupera uma conexão, já instancia a nossa DAO podemos usar essa categoria para chamar as nossas DAO já fazendo essa conexão. Como podemos fazer isso?

[02:11] Vamos lá, vamos criar em `CategoriaController`, um `private CategoriaDAO`, porque nós vamos precisar utilizar essa `CategoriaDAO` no método, então ela vira um atributo da nossa classe. Aqui criamos o nosso construtor, o `public CategoriaController()`. Vamos chamar o `new ConnectionFactory(). recuperarConexao();`, que é o responsável por recuperar a nossa conexão com o seu método `recuperarConexao()`.

[02:32] E ele vai me fornecer uma `Connection connection = new ConnectionFactory(). recuperarConexao();`, que vai ser o que nós vamos fornecer para a nossa DAO. Recuperamos a conexão, chamo aqui agora o `CategoriaDAO`, deixa eu instanciar ele aqui, `new CategoriaDAO(connection);`, ele vai esperar uma Connection, que é a que recuperamos no nosso construtor.

[02:57] E atribuímos essa linha para o nosso `this.categoriaDAO = new CategoriaDAO(connection);`, da nossa classe. Dessa forma já fizemos o necessário para instanciar a nossa DAO passando a conexão, agora o que eu preciso fazer é: vamos chamar a DAO e o método de listar dessa DAO. Então eu vou fazer aqui `public List<Categoria> listar()`, `return this.categoriaDAO.listar();`. Está pronta a nossa classe.

[03:30] Só que eu gostaria de chamar a atenção de vocês aqui para esses warnings que está dando na nossa classe. O nosso método está correto, o nosso construtor está correto, mas ele continua dando esse alerta. É porque, quando criamos as nossas DAOs, tanto no `ConnectionFactory` quanto no `listar()`, nós não fizemos o tratamento da SQL Exception, nós apenas subimos a SQL Exception.

[03:59] Quando fazemos isso, que nós subimos essa exceção, que é quando usamos o `throws SQLException` ou qualquer outra exceção, eu estou avisando para quem me chamar que esse código pode dar um SQL Exception, mas eu não estou tratando. Então o que vai acontecer? Se eu não tratar esse SQL Exception na minha `CategoriaController`, eu vou ter que subir essa exceção para a nossa `ProdutoCategoriaFrame`.

[04:28] Não faz sentido a nossa *view* informar que aquela método, aquela chamada, pode ocasionar uma SQL Exception. O SQL Exception, ele pode dar no momento em que eu faço a operação com o banco de dados, no momento em que eu recupero uma conexão com o banco de dados. Então nada mais justo do que trazermos o tratamento dessas exceções onde realmente pode dar exceção.

[04:54] No nosso caso é a `ConnectionFactory` e a `CategoriaDAO`. Então o que nós vamos fazer aqui? Eu vou tirar esse `throws SQLException`. E nós vamos fazer o seguinte, em `Connection recuperarConexao()`, eu vou fazer um `try`, então tente recuperar uma conexão com o meu banco de dados.

[05:12] Caso você não consiga, podemos dar uma `catch(SQLException e)` e usar um *pattern*, que é para lançar invés de uma exceção checada, nós vamos relançar uma exceção *unchecked*, que uma `throw new RuntimeException()` - calma que o meu computador deu uma travada aqui. Então nós vamos relançar a `RuntimeException(e)` passando o motivo, que pode ser uma SQL Exception.

[05:49] Feito isso nós tratamos o `recuperarConexao()` e nós vemos que sumiu no nosso construtor o problema que nós tínhamos no momento de recuperar a conexão. Vamos fazer a mesma coisa no `listar();` do `CategoriaDAO`. Nós vamos tirar o `throws SQLException` e vamos colocar o nosso `try` em `listar()`.

[06:13] Tentei então recuperar uma lista de categorias, se você não conseguir, vai dar uma SQL Exception e nós vamos relançar com o *pattern*, que é

relançando uma exceção *unchecked*. Então uma `catch (SQLException e)`, `throw new RuntimeException(e);`. Está aí o nosso tratamento. Agora, se nós formos ver, o nosso método também parou de reclamar.

[06:48] Uma vez aqui com a `CategoriaController` pronta, nossa `ConnectionFactory` agora com o tratamento de exceção correto, `CategoriaDAO`, o método `listar()` também. Vamos chamar aqui o `ProdutoCategoriaFrame` para ver se vai ter algum problema. Não, desculpa, é no `TestaOperacaoComView`. Vamos chamar ele e vamos ver se ele já está carregando a nossa combo com as categorias.

[07:15] Se formos ver as categorias, eletrônicos, eletrodomésticos e móveis, são as três categorias que nós temos no nosso banco de dados. Funcionou então a ideia de criarmos os nossos Controllers e fazer o tratamento melhor das nossas exceções. Só que agora eu vou deixar o desafio para vocês, para vocês fazerem isso também nos outros métodos da `CategoriaDAO` e em `ProdutoDAO` também.

[07:38] Também vou pedir para vocês, vou desafiar vocês a também criarem a nossa `ProdutoController` com todas essas melhorias que nós fizemos no código. Então na próxima aula nós vamos ver como vai ficar a nossa `ProdutoController` e vamos ver o funcionamento total do sistema, agora com a nossa *view*, Controller e DAO todas implementadas. Então encontro vocês no próximo vídeo.