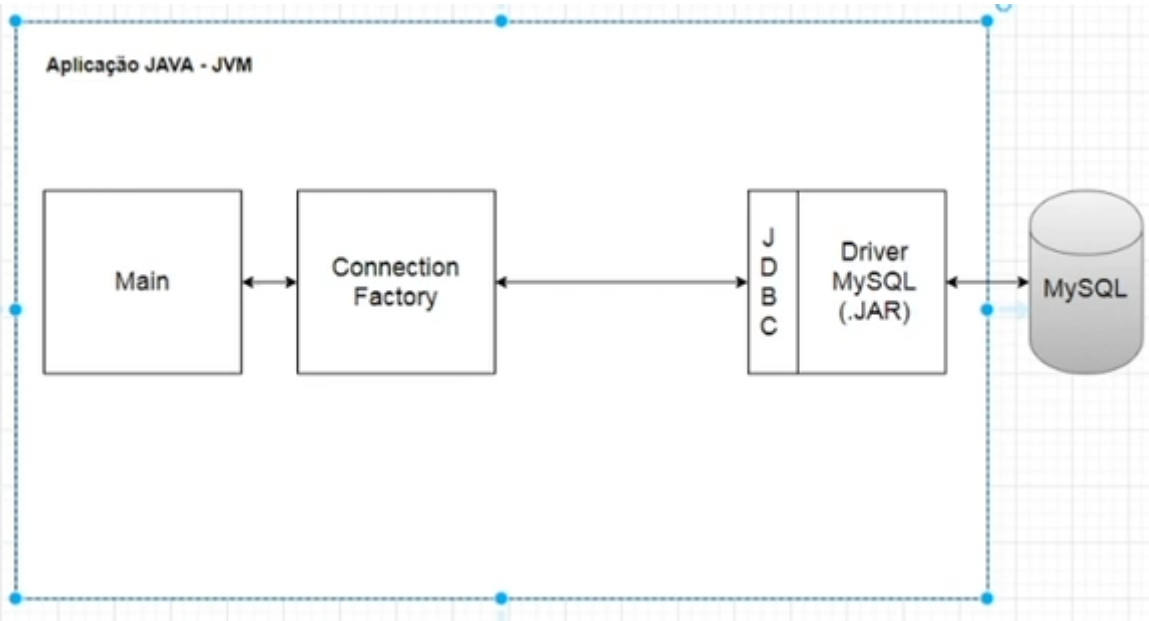
03

O que é pool e datasource?

Transcrição

[00:00] Fala, aluno. Tudo bom? Dando continuidade ao nosso curso de JDBC, eu gostaria de revisar agora como está a estrutura da nossa aplicação. Nós estamos comunicando a nossa aplicação em Java com o banco de dados MySQL. Só que nós vimos que essa aplicação, ela não se comunica com o banco de dados de forma nativa.



[00:20] Justamente porque de um lado nós temos a linguagem Java e do outro lado, nós temos uma linguagem MySQL, que não nos importa qual é, mas nós sabemos que não é simplesmente codificar nas nossas classes que nós vamos conseguir comunicar com o MySQL. Então, para isso, nós vimos que é necessário um driver.

[00:46] Esse driver é específico dos bancos de dados. O que isso quer dizer? Hoje, utilizando o MySQL como base dados, nós vamos ter que utilizar um

driver do próprio MySQL. Se um dia eu quiser trocar o meu banco de dados de MySQL para Postgre, eu vou ter que utilizar aqui um driver do Postgre.

[01:04] Só que isso já nos levantou uma outra dúvida: se eu tenho um driver específico, então quando eu for trocar o MySQL por um Postgre, então eu vou ter que sair procurando no código onde é utilizada essa implementação do driver, onde estamos chamando as classes, interfaces do driver, e vamos ter que alterar para classes e interfaces do Postgre.

[01:31] Só que nós vimos que não é necessário, porque nós temos uma interface, que é chamada JDBC, que ela tem as suas classes e interfaces, que expõe essa implementação dos drivers. Então quando eu quero solicitar uma conexão para o MySQL, eu vou na minha interface JDBC e eu solicito um `getConnection`, que é uma interface do próprio JDBC e ele que se vire com o driver e faça a requisição da minha conexão.

[02:08] Então, com a interface do JDBC, nós vimos que não precisamos conversar diretamente com as classes dos drivers, que são bem específicas. Com isso, nós vimos então que quando eu quero chamar, quando eu quero abrir uma conexão com o banco de dados o meu Driver manager, o meu get connection e passar a string de conexão entre parênteses. Só que nós vimos que quando necessitávamos abrir uma conexão, nós tínhamos que repetir esse código.

[02:48] Voltando à parte de se um dia precisássemos alterar o nosso banco de dados, em todo lugar que fosse chamado o driver manager e o `.getConnection`, eu teria que sair alterando a string de conexão, porque eu agora me conecto com outro banco. E nós vimos que isso não é muito legal, porque poderíamos esquecer, enfim, tudo aquilo que já vimos que poderia prejudicar a nossa aplicação.

[:03:14] Então por isso construímos uma Connection Factory. Essa Connection Factory é justamente para isolar a chamada do Driver manager em um único lugar e expor um método recuperar conexão para os nossos main, que de fato

vão ser os `TestaInsercao` , `TestaListagem` , tudo aquilo que viemos fazendo ao longo do curso.

[03:36] Revisada estrutura da nossa aplicação, agora temos que nos atentar a um problema. Não a um problema, mas a uma situação. Hoje, toda requisição que estamos fazendo está sendo apenas de uma única conexão. Então quando eu chamo o meu main, ele vai, abre uma conexão com o banco de dados e me retorna o que eu quero, fechei a conexão. Quando eu faço outro main, ele faz o mesmo procedimento e fecha a conexão.

[04:06] Só que vamos agora no site da Alura. Se eu fizer o *login* e entrar no meu Dashboard, vamos supor que eu quero ver quais são as formações. Quando eu entro, por exemplo, nas formações em mobile de Android, quando eu estou fazendo essas requisições de página, a plataforma faz a solicitação ao banco de dados para me retornar essas informações da página para eu conseguir ver tudo isso que estamos vendo na nossa tela.

[04:39] Só que no momento em que eu estou na página, vamos supor que eu tenho um outro aluno também esteja fazendo a mesma coisa. Voltando ao nosso diagrama, como seria o funcionamento desse caso na nossa aplicação? Eu fiz uma requisição e o meu banco de dados está processando. O outro aluno, ele fez uma requisição, então eu vou enfileirar a requisição dele para só quando a minha for processada a dele seja processada?

[05:06] Faz muito sentido? Para mim não faz. Então o que poderia fazer mais sentido? No momento em que eu fiz uma conexão, outro aluno fez também, então nós abrimos duas conexões. Dessa maneira já fica um pouco melhor, porque agora eu não estou enfileirando as conexões e o meu banco de dados está processando as duas requisições ao mesmo tempo.

[05:34] Só que temos agora, no site da Alura, uma parte de novidades. Vamos supor que disparou um e-mail falando sobre um curso de Build de uma aplicação .NET. Esse curso teve um alcance muito grande, vários alunos gostaram do que se propõe a fazer no curso, e vários alunos começaram a fazer

requisições na plataforma para conhecer mais sobre o curso. Eles receberam o e-mail mas foram na plataforma da Alura para conhecer mais sobre o curso.

[06:09] Então a tendência é que essas conexões entre a aplicação e o JDBC, elas vão aumentando. O alcance de uma plataforma como a Alura é de milhões de alunos. Quando eu tenho milhões de alunos, com essa estrutura de abrir conexões descontroladamente, eu vou abrir milhares de conexões. E o que vai acontecer com o meu banco de dados?

[06:33] Ele não vai aguentar, porque, de fato, o MySQL é um banco de dados muito bom, é um dos mais utilizados hoje em dia, nós já vimos sobre isso. Só que nenhum banco de dados vai aguentar um processamento de milhões de conexões de uma vez, de forma descontrolada. Aqui nós encontramos um problema. Como solucionamos esse problema?

[06:58] O ideal seria então que invés de eu sair abrindo conexões de forma descontrolada, antes da minha interface JDBC, eu tivesse uma outra caixa no diagrama que fizesse o seguinte: quando eu subir a minha aplicação, eu quero ter um número X de conexões abertas. Eu não quero que seja apenas uma, porque isso nós vimos que pode enfileirar as nossas requisições.

[07:37] Mas eu também não quero que ele abra de forma descontrolada, porque isso pode fazer com que o nosso banco de dados, ele caia. Então eu quero agora ter um número X de conexões estabelecidas. Com essa caixa entre a aplicação e o JDBC, o que ela faria? Ela seria responsável por abrir essas conexões com o banco de dados.

[08:06] Então agora invés de ter todo aquele descontrole de conexões, agora eu tenho um número X, que eu posso até colocar um número mínimo e um número máximo de conexões e essa caixa que vai conversar agora com a interface JDBC. Essa estrutura agora parece ficar muito legal, porque passamos a ter o controle dessas conexões, a nossa aplicação fica de uma forma bem legal.

[08:42] Agora, o que precisaríamos fazer? Nós precisaríamos, de fato, implementar essa caixa, que até então não sabemos como seria o código. Só que olha que maravilha: nós não vamos precisar implementar essa caixa, porque ela já existe hoje para nós, ela é chamada de Pool de conexões.

[09:10] Então agora, essa caixa entre a aplicação e o JDBC, nós vamos ter um Pool de conexões, que é exatamente para isso que estamos vendo, é para ter um número de conexões abertas para nós. Deixa eu só aumentar a fonte para ficar mais fácil de ver, para ficar bonito o nosso desenho. Então agora com esse Pool de conexões, deixa eu só abaixar ele, para não ficar em cima.

[09:41] Então é essa caixa aqui. O Pool de conexões, para nós, vai ser também um driver, assim como nós temos o driver do MySQL, nós temos o driver do Pool de conexões. O driver que nós vamos utilizar, para o nosso Pool de conexões, para se comunicar com o MySQL, vai se chamar C3P0.

[10:05] Ele não é o único, eu tenho vários drivers, assim como nós temos os drivers de banco de dados, o MySQL, o Postgre, eu vou ter também vários drivers de Pool de conexões. Só que o C3P0, ele vai nos servir para tudo o que vamos precisar usar na hora que formos implementar o nosso código de fato.

[10:34] A documentação dele não é complexa, a documentação dele é tranquila, nós encontramos na internet tudo o que precisa para configurar, para termos um Pool de conexões correto, de acordo com as boas práticas, enfim. Só que se o Pool de conexões, ele é um driver e ele pode ser alterado, eu posso mudar o C3P0 para outro driver, vamos ter aquele problema que teríamos sem a interface JDBC.

[11:26] Porque eu vou implementar um Pool de conexões, eu posso injetar ele em alguns pontos na minha aplicação, mas quando eu fosse alterar ele, eu teria que sair modificando, talvez, na minha aplicação onde eu chamo esse meu Pool de conexões. Só que temos uma vantagem: como temos a interface JDBC, nós temos também a interface para o nosso Pool de conexões, que se chama Datasource.

[11:58] Então deixa eu só criar um retângulo para o diagrama, que ele vai ficar um pouco antes do Pool de conexões. Então esse retângulo, ele vai ser a minha interface do Pool de conexões. Essa interface vai ser a minha interface Datasource. Deixa eu escrever, Datasource. Com o Datasource, vamos conseguir expor todas as configurações do nosso Pool de conexões.

[12:41] Como eu tinha falado anteriormente, no Pool de conexões eu posso limitar, pôr uma quantidade mínima de conexões abertas e uma quantidade máxima. Então eu faço todas essas configurações no meu Pool de conexões e com o meu Datasource, eu apenas exponho apenas para a minha Connection Factory.

[12:58] Com essa estrutura agora, eu não ia mais precisar solicitar da Connection factory a abertura de uma conexão, eu só ia ter que perguntar para o meu Datasource o seguinte: Datasource, eu tenho conexão aberta? Tenho. Então eu vou utilizá-la. E a nossa Connection factory passa a fazer esse tipo de trabalho.

[13:28] Então qual é o objetivo do Pool de conexões? Eu ter um controle maior das minhas conexões, de quantas conexões vão estar abertas, porque nós vimos que não é uma boa prática eu abrir e fechar conexões descontroladamente e eu também não vou ter apenas uma conexão, para enfileirar as minhas requisições. Então, com essa estrutura de Pool de conexões, já começamos a ver como funcionam aplicações do mundo real.

[13:59] Então todas as aplicações que vão trabalhar com o banco de dados hoje, na verdade usam um Pool de conexões com interface Datasource. Então agora a intenção é que nas próximas aulas, nós vamos botar a mão na massa mesmo, que vai ser para exatamente implementarmos essa caixa do Pool de conexões e o Datasource no nosso código. E vamos verificar como fazer as configurações do Pool de conexões e vamos ver como é a interface Datasource.

[14:33] Então, nessa aula nós vamos ficando por aqui. Espero que vocês tenha entendido o desenho. Quem ainda está com alguma dúvida, na hora que

formos programar, que formos codificar, eu acho que vai sanar essas dúvidas. A ideia agora é que a nossa aplicação agora, ela já vá para um patamar de uma aplicação real. Então é isso, aluno. Vejo vocês no próximo vídeo.