02. Executando comandos SQL no Java

Date	@12/12/2022
Categoria	Java e persistência
Curso	Java e JDBC: trabalhando com um banco de dados

Tópicos

- Projeto da aula anterior
- Listagem com Statement
- Connection, Statement e ResultSet
- Criando a ConnectionFactory
- Factory Pattern
- Inserindo com Statement
- Retorno do método execute()
- Removendo dados
- Faça que eu fiz
- O que aprendemos?

Connection, Statement e ResultSet

O que o `java.sql.Connection`, `java.sql.Statement` e `java.sql.ResultSet` possuem em comum?

- Todas são interfaces: `Connection`, `Statement` e `ResultSet` são algumas das interface do pacote `java.sql`.

Factory Pattern

Qual a vantagem de utilizar uma `ConnectionFactory` na nossa aplicação?

- Fornecer uma maneira mais simples para criar um objeto
    - Criamos objetos sem expor a lógica ou as configurações de criação ao cliente. Além disso, podemos nos referir ao objeto recém-criado usando uma interface (usando uma abstração), desacoplando a implementação.

Retorno do método execute()

Como funciona o retorno do método `execute`, da interface `java.sql.Statement`?

- O método devolve `true` quando o seu resultado é um `java.sql.ResultSet` e `false` caso contrário (*update*, *delete*, etc).

O que aprendemos?

Nesta aula, aprendemos que:

- Para simplificar e encapsular a criação da conexão, devemos usar uma classe `ConnectionFactory`
    - A classe `ConnectionFactory` segue o padrão de criação *Factory Method*
    - O *Factory Method* encapsula a criação de um objeto
- Para executar um comando SQL, podemos usar a interface `java.sql.Statement`
    - O método `execute` envia o comando para o banco de dados
    - Dependendo do comando SQL, podemos recuperar a chave primária ou os registros selecionados.
