



DIGITAL
INNOVATION
ONE

Java e o Banco de Dados

Java JDBC Básico

Daniel Karam

Senior Software Developer

Objetivos da Aula

1. Configurar Banco de Dados

2. JDBC e drivers de conexão

3. Consultas com JDBC

Requisitos Básicos

- ü MySQL (SGBD) e noções de SQL
- ü Java Development Kit (JDK) – 1.8 ou superior
- ü IntelliJ 2019.2.3 IDE
- ü Gradle 5.3.1 (Para baixar o Driver JDBC)

Materiais

Endereço no Github dos materiais que serão utilizados nessa aula:

Ø https://github.com/danielkv7/digital-innovation-one/tree/master/Aula_JDBC_basico

Parte 1: Configurar Banco de Dados

Java e o Banco de Dados

Configurar Banco de Dados

Um Banco de Dados (BD) armazena dados de forma estruturada, tornando o acesso e atualização dos dados mais rápido, pois aumenta a eficiência computacional (menor “gasto” de memória, processamento e tempo).

Nesta aula será utilizado o banco de dados relacional MySQL.

Configurar Banco de Dados

Passos para instalar e configurar o banco de dados para esta aula:

1. Instalar MySQL
2. Configurar usuário e senha
3. Instalar MySQL Workbench (Opcional)
4. Criar Banco de dados
5. Criar uma tabela

URL com Instruções de Instalação Ubuntu 18.04

https://github.com/danielkv7/digital-innovation-one/blob/master/Aula_JDBC_basico/jdbc-basico/src/main/java/part1/DatabaseInstructions

Configurar Banco de Dados

Scripts SQL para criar tabela utilizada nessa aula :

```
CREATE database digital_innovation_one;
```

```
USE digital_innovation_one;
```

```
CREATE TABLE aluno (  
    id INTEGER PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(80) NOT NULL,  
    idade INTEGER NOT NULL,  
    estado CHARACTER(2) NOT NULL  
);
```


Exercício final

1. Configure um banco de dados de acordo com os passos explicados nos slides anteriores

Parte 2: JDBC e drivers de conexão

Java e o Banco de
Dados



JDBC e drivers de conexão

JDBC(Java Database Connectivity) é uma API com diversas classes e interfaces escritas na linguagem Java que estão presentes nos pacotes java.sql e javax.sql. Elas permitem que programas em Java realizem conexões em bancos de dados para realizar consultas. Uma dessas classes principais é o driver JDBC que intermedia essa interação.

Sem a API JDBC, seria necessário conhecer o protocolo proprietário de cada banco de dados para se conectar e realizar consultas. Já com a API JDBC, é utilizada somente UMA interface Java para qualquer banco de dados, deixando o driver implementar as especificações de cada banco de dados, enquanto o desenvolvedor se preocupa apenas em selecionar um driver e criar as queries (neste caso, consultas SQL).



JDBC e drivers de conexão

Classes e interfaces que serão utilizadas:

- ü Classe **DriverManager** – Responsável pela comunicação com os drivers disponíveis. É utilizada para criar uma **Connection** com o banco de dados através de uma **URL** (que especifica driver, localização do BD e nome do BD).
- ü Interface **Connection** – Representa a conexão com o banco de dados. Permite criar “**Statements**” que constroem consultas SQL.



JDBC e drivers de conexão

Passos para se conectar ao banco de dados:

1. Realizar download do driver específico para o BD que será utilizado (nesta aula, será o MySQL). É possível baixar o driver manualmente ou através do Gradle ou Maven.
2. Criar URL (string de conexão) com os seguintes parâmetros: driver, endereço do BD e nome do BD.
3. Criar uma connection através do “DriverManager” utilizando o método “getConnection”, passando os parâmetros: string de conexão, usuário e senha.



Exercício final

1. Criar outro usuário do BD e senha deste usuário e se conectar através da API JDBC.
2. Explorar os métodos da classe DriverManager e da interface Connection através da IDE (ex: IntelliJ IDEA, Eclipse...) ou documentos oficiais.
3. Configurar outro banco de dados (ex: PostgreSQL, H2...) e tentar se conectar a ele utilizando a API JDBC.

Parte 3: Consultas com JDBC

Java e o Banco de
Dados



Consultas com JDBC

Existem 3 interfaces para montar comandos SQL:

- Ø **Statement** – Executar SQL comuns
- Ø **PreparedStatement** – Executar SQL parametrizáveis
- Ø **CallableStatement** – Executar stored procedures



Consultas com JDBC

Preferir **PreparedStatement** ao **Statement** quando for parametrizar a consulta pois:

- ü Previne SQL Injection

- ü Melhora legibilidade

- ü Melhora desempenho



Consultas com JDBC

Existem 3 métodos para executar comandos SQL:

- Ø **execute** – Pode executar qualquer tipo de SQL
- Ø **executeQuery** – Usado para executar “SELECT”
- Ø **executeUpdate** – Usado para comandos de alteração de banco de dados (INSERT, UPDATE, DELETE, CREATE, ALTER)



Consultas com JDBC

ResultSet – objeto que contem os dados de uma determinada consulta no banco de dados (normalmente com SELECT)

São utilizados os **métodos getters para buscar dados** do ResultSet. Tais como: **getInt**, **getFloat** e **getString**.

O método **next()** é utilizado para percorrer os registro do **ResultSet**. (Normalmente utilizado junto com **while**)



Exercício final

1. **Crie uma tabela no BD** chamada **curso** que terá como colunas: **id**, **nome**, **duracao_horas**. (no BD, a nomenclatura utilizada é snake_case).
2. **Crie uma classe em Java** chamada **curso** que terá os mesmo atributos que a tabela criada no BD (no exercício anterior). (obs: duracao_horas será camelCase).
3. **Crie CursoDAO** que será responsável por se **conectar ao BD** para **realizar as operações CRUD** (Create, Read, Update, Delete).
4. **Testar os métodos do CursoDAO** em uma classe que tenha o método **public static void main(String[] args)**.

Contato

Linkedin -> <https://www.linkedin.com/in/daniel-kv/>