

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E  
TECNOLOGIA DO RIO GRANDE DO NORTE  
CAMPI JOÃO CÂMARA

# CONECTANDO A APLICAÇÃO COM O BANCO DE DADOS

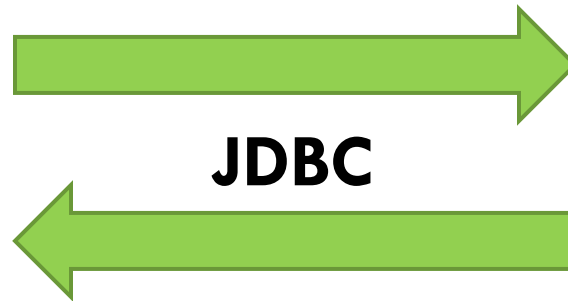
Nickerson Fonseca Ferreira  
[nickerson.ferreira@ifrn.edu.br](mailto:nickerson.ferreira@ifrn.edu.br)

# Software

2



API SWING



SGDB + SQL

# Banco de dados

3

- Local onde os dados do nosso sistema serão armazenados.
- SGDB é o sistema que realiza a gestão do nosso banco de dados.
- SQL é a linguagem que utilizamos para manipular os dados.
  - DML
  - DDL
  - DCL

# Java + SWING

4

- Java: Linguagem de programação utilizada para o desenvolvimento de software.
- SWING: API da linguagem Java que é utilizada para criação de interfaces gráficas.

**JAVA É O PODER!!!**

# JDBC

- ❑ JDBC é uma interface baseada em Java para acesso a banco de dados.
- ❑ Pacote padrão: `java.sql`.
- ❑ A maioria dos fornecedores de SGBD oferece uma implementação específica (Driver) para seu SGBD (sempre implementando a interface).
- ❑ Utilizando o JDBC é possível obter acesso direto ao banco de dados através de aplicações Java.

# Passos para criar a primeira conexão

6

1. Realizar o download do driver específico do SGBD utilizado pela aplicação. Ex: MySQL;
2. Carregar drivers;
3. Estabelecer uma conexão com o SGBD através do método `getConnection` da classe `DriverManager`;
  - ▣ Esse método retornará um objeto do tipo `Connection`.
4. Criar um objeto `Statement` a partir do método `createStatement` do objeto `Connection` criado no passo anterior;
5. Executar o comando SQL utilizando os métodos existentes no objeto `Statement`;
6. Fechar a conexão.

# Passo 1

7

- Acessar o site do fornecedor do SGBD utilizado no sistema e baixar o driver JDBC.

The screenshot shows a web browser at the URL `dev.mysql.com/downloads/connector/j/`. The page is the MySQL Connector/J download page. It features a navigation menu with 'Downloads' selected, and sub-menus for 'Enterprise', 'Community', 'Yum Repository', 'APT Repository', 'Windows', and 'Archives'. The main content area is titled 'Download Connector/J' and includes the following text:

MySQL Connector/J is the official JDBC driver for MySQL.

Online Documentation:

- [MySQL Connector/J Installation Instructions, Documentation and Change History](#)

Please report any bugs or inconsistencies you observe to our [Bugs Database](#).

**Thank you for your support!**

MySQL open source software is provided under the GPL License.

OEMs, ISVs and VARs can purchase commercial licenses.

The 'Generally Available (GA) Releases' section is active, showing the following release information:

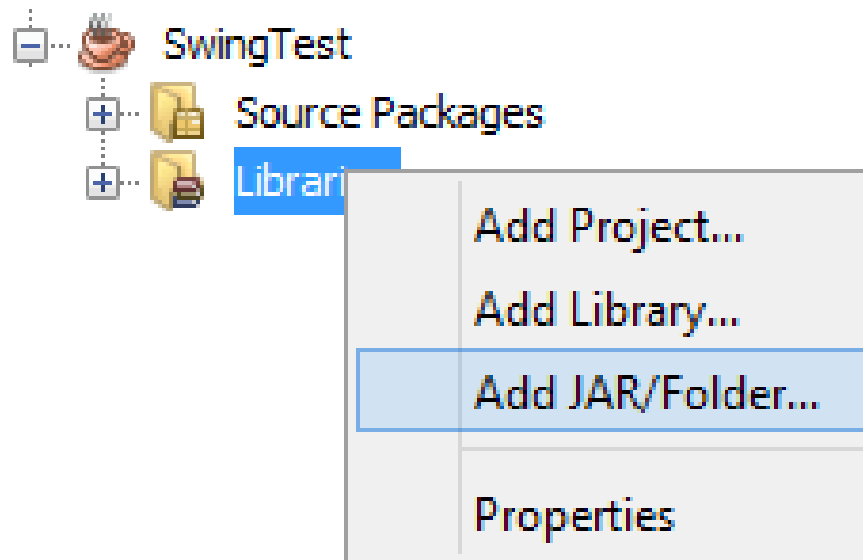
Platform	Version	Size	Action
Windows (x86, 32-bit), MSI Installer	5.1.34	6.2M	<a href="#">Download</a>

Other elements include a search bar, social media icons (Facebook, Google+, Twitter, YouTube), and a sidebar with links to various MySQL products and connectors.

# Passo 1

8

- Depois de baixar o driver JDBC basta adicionar o jar nas bibliotecas do projeto.





# Passo 2

9

- O segundo passo é informar ao Java e carregar a classe do driver adicionado nas bibliotecas do projeto.
- Para isso utilizamos o comando:  
**Class.forName("nomeDoDriver");**
  - `Class.forName("oracle.jdbc.driver.OracleDriver");`
  - `Class.forName("com.mysql.jdbc.Driver");`
  - `Class.forName("org.hsqldb.jdbcDriver");`
  - `Class.forName("org.postgresql.Driver");`

# Passo 3

10

- Agora devemos, através do DriverManager, criar uma instância da conexão com o banco de dados.
- O método responsável pela criação dessa instância é o: `getConnection(String url, String usuario, String senha);`

```
Connection conn =  
    DriverManager.getConnection("jdbc:mysql://localhost:3306/meubanco"  
    , "usuario", "senha");
```

# Passo 4

11

- A criação do Statement é necessária para que possamos executar os comandos SQL diretamente no banco de dados.
- A criação do Statement é realizada através da instância da classe Connection criada no passo anterior.

**Statement stm = conn.createStatement();**

- Os métodos do Statement utilizados para executar os comandos SQL são:
  - `execute()`, `executeQuery()` e `executeUpdate()`.

# Passo 4

12

```
stmt.execute("CREATE TABLE dinossauros  
+ "(codigo INT PRIMARY KEY, "  
+ "genero CHAR(20), "  
+ "especie CHAR(20));");
```

```
int linhasModificadas =
```

```
stmt.executeUpdate("INSERT INTO dinossauros "  
+ "(codigo, genero, especie) VALUES "  
+ "(499,'T-Rex','carnívoro')");
```

```
ResultSet cursor =
```

```
stmt.executeQuery("SELECT genero, especie "+  
" FROM dinossauros "+  
" WHERE codigo = 355");
```

**Método retorna um booleano**

**Retorna um inteiro. (número de linhas).**

**Retorna ResultSet com o resultado da consulta.**

# Passo 5

13

- Depois de executar todos os comandos, a conexão com o banco de dados deve ser fechada.
- Todos os componentes envolvidos na conexão com o banco de dados devem ser fechados: Connection, Statement e ResultSet.
  - **cursor.close();**
  - **stmt.close();**
  - **conn.close();**

# EXERCÍCIO

14

- ❑ Criar uma tabela cliente no MySQL com as colunas: id, nome e e-mail.
- ❑ Criar um JFrame para inserir registros nesta tabela.
- ❑ Adicionar um botão para selecionar um cliente pelo nome e preencher os campos do formulário.
- ❑ Adicionar um botão para alterar os dados do cliente que foi selecionado.
- ❑ Adicionar um botão para excluir os dados do cliente que foi selecionado.