



Desenvolvimento de Sistemas DS

- Conexão JAVA com BD MySql

- JAVA -



Driver de Conexão JDBC

O que é um Driver?


Driver é um tipo de software que tem por finalidade permitir a comunicação entre sistemas.

Os drivers geralmente são fornecidos pelos fabricantes, mas é comum encontrar drivers genéricos distribuídos com o sistema operacional e também drivers de terceiros, gratuitos ou não.

O que se espera de um driver é a máxima compatibilidade e velocidade sem degradar a performance do sistema.

Um driver tem que fazer o seu papel de forma transparente e sem que seja criado um gargalo.

Além de atuar como uma interface entre os SGBDs e as aplicações, também pode ser considerado como um tradutor que ajuda na definição das mensagens binárias trocadas com um protocolo de um SGBD.



O que é JDBC?

JDBC – Java DataBase Connectivity /

Padrão Java de Conectividade a Banco de Dados.

É a interface que possibilita às aplicações Java acessarem bancos de dados relacionais e demais arquivos de dados. É atualmente a forma mais prática e rápida para conectar uma aplicação Java a um arquivo de dados.

Pode se dizer que é uma API que reúne conjuntos de classes e interfaces escritas na linguagem Java na qual possibilita se conectar através de um driver específico do banco de dados desejado.

Com esse driver pode se executar instruções SQL de qualquer tipo de banco de dados relacional. Para fazer a comunicação entre a aplicação e o SGBD é necessário possuir um driver para a conexão desejada.

Geralmente, as empresas de SGBD oferecem o driver de conexão que seguem a especificação JDBC para caso de algum desenvolvedor querer utilizar.



JDBC (Java Database Connectivity)

- Contém métodos para:

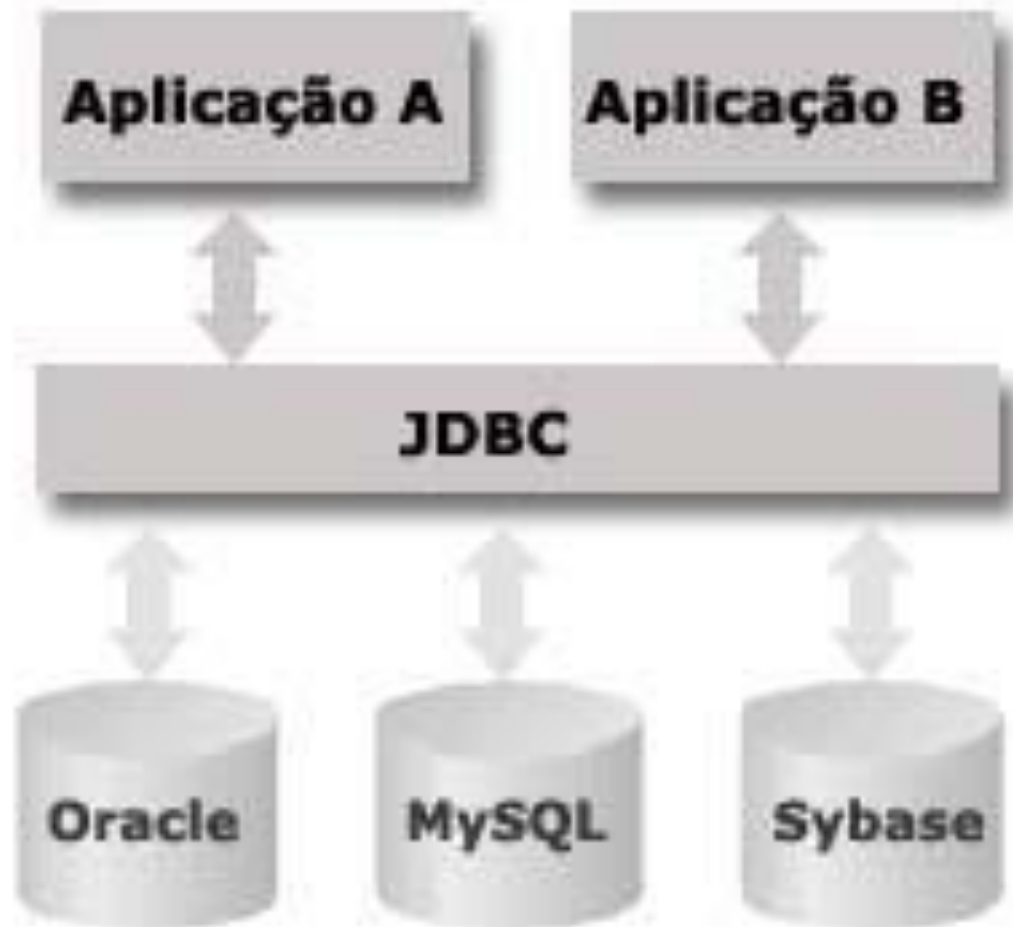
- ☐ Criar conexões a DataSources remotos.
- ☐ Executar comandos SQL.
- ☐ Examinar conjuntos de resultados a partir de comandos SQL executados.
- ☐ Gerenciamento de transações.
- ☐ Verificação de exceções.
- ☐ dentre outros.



JDBC (Java Database Connectivity)

- É uma coleção de classes e interfaces Java que possibilita acesso ao banco de dados de programas escritos na linguagem Java.
- O JDBC possibilita que um único executável acesse diferentes SGBDs sem recompilação.

JDBC (Java Database Connectivity)





Exemplos de protocolos nativos

- MySQL:
 - `com.mysql.jdbc.Driver`
- Oracle:
 - `oracle.jdbc.friver.OracleDriver`
- MS SQL Server:
 - `com.microsoft.jdbc.sqlserver.SQLServerDriver`
- Postgre:
 - `org.postgresql.Driver`
- DB2:
 - `com.ibm.db2.jcc.DB2Driver`



Pacote java.sql

Esse pacote oferece a biblioteca Java o acesso e processamento de dados em uma fonte de dado. As classes e interfaces mais importantes são:

Classe	Interface
DriverManager	Driver
	Connection
	Statement
	ResultSet
	PreparedStatement



Importando o pacote java.sql

Para desenvolver uma aplicação baseada em uma especificação JDBC é preciso entender algumas das principais classes e interfaces apresentadas a seguir.

Existe um ponto de atenção na importação das classes ou interfaces relacionadas ao pacote a ser usado no momento do desenvolvimento.

```
import java.sql.*;
```

Esta é a forma correta da importação do pacote referente à classe Connection pertencente ao pacote java.sql.

Esse é um fator a ser observado com cautela, pois isso é considerado um dos erros mais comuns justamente pelo fato do desenvolvedor pensar muitas vezes em usar o pacote com.mysql.jdbc sendo que está utilizando o driver JDBC do banco MySQL.



A classe DriverManager

O DriverManager é uma classe do pacote `Java.sql` e serve para:

- ☐ Fazer a conexão com o banco de dados;
- ☐ Gerenciar o conjunto de drivers JDBC correspondente;
- ☐ Controlar o Login;
- ☐ Controlar as mensagens entre o banco de dados e o driver.

Um programador que utiliza um driver não precisa saber como ele foi codificado, mas deve saber como carregá-lo para que ele se registre no DriverManager.

Cada driver que se deseja a usar precisa ser registrado nessa classe, pois oferece métodos estáticos para gerenciar um driver JDBC.

Para a aplicação reconhecer a comunicação com o banco de dados escolhido, é preciso obter um arquivo com a extensão **.jar** que geralmente se consegue através dos sites das empresas que distribuem o SGBD.


Esse arquivo tem o objetivo ajudar no carregamento do driver JDBC.



Instalar o driver JDBC para Mysql

- ☐ Faça o download do Mysql-connector ele é um arquivo .zip
Download: <http://dev.mysql.com/downloads/connector/j/>
- ☐ Descompacte o arquivo.
- ☐ Crie uma pasta chamada DriverMySQL no seu projeto e coloque o driver (mysql-connector-java-8.0.31-bin.jar) dentro dela.

Obs: os números que ficam no final do nome do arquivo “conector” indicam a versão do conector: mysql-connector-java-[8.0.31](#)-bin.jar. Este se refere a versão 12 do Netbeans.



Configuração do Netbeans Apache para conexão JDBC

- O driver JDBC é um arquivo .jar disponibilizado pelos desenvolvedores dos SGBD de forma gratuita ou paga de acordo com a licença adotada.
- O Netbeans necessita de algumas configurações para reconhecer e implementar o driver JDBC a um projeto.
- O driver (o arquivo “.jar”) deve estar sempre disponível para a aplicação e deve ser copiado para “dentro” da pasta do projeto.

Adicionar o driver (.jar) no projeto – parte 1

- 1 – Abra a guia “PROJECTS”,
- 2 – Clique com o botão direito do mouse em “DEPENDENCIES”,
- 3 – Selecione “ADICIONAR DEPENDENCIA”,
- 4 – Digite “group id”, “artefact id” e “version” conforme figura ao lado, depois clique em “ADICIONAR” (add):

Add Dependency

Group ID: grupo

Artifact ID: artefato

Version: 1 Scope: compile

Type: Classifier:

Search Open Projects Dependency Management

Query: (coordinate, class name, project name...)

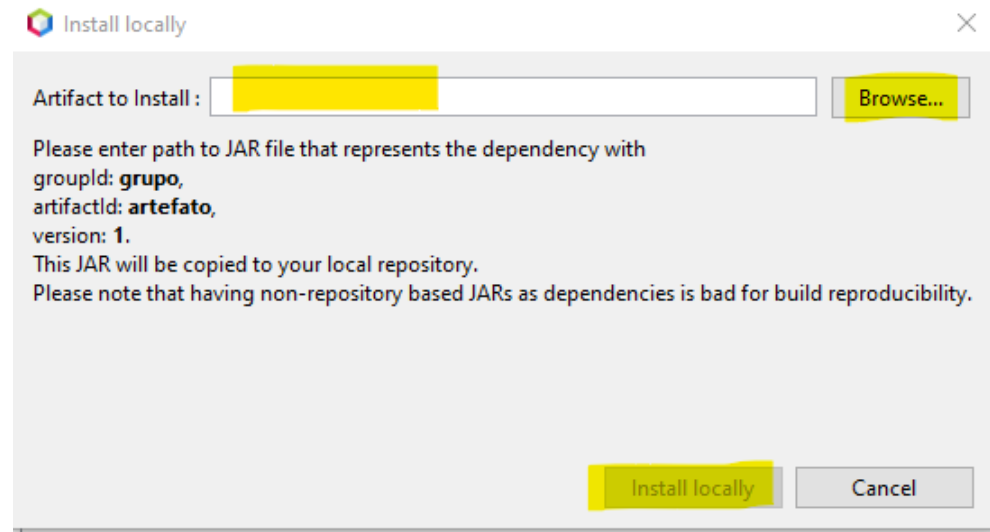
Search Results:

Add Cancel

Adicionar o driver (.jar) no projeto – parte 2

- 1 – Expandir “DEPENDENCIES”,
- 2 - Clique com o botão direito do mouse no “ARTEFATO” (recém criado no passo anterior – parte 1),
- 3 – Selecione “INSTALAR MANUALMENTE O ARTEFATO”, abrirá a janela abaixo:
- 4 – Através do botão “BROWSE” encontre a pasta do projeto em que está copiado o driver e selecione-o,
- 5 – Clique em “INSTALAR LOCALMENTE”.

Pronto driver de conexão instalado!!





Métodos

A interface Connection

Representa uma conexão ao banco de dados. Nessa interface são apresentados os métodos mais utilizados:

- **Método close**

Geralmente é inserido dentro do bloco finally para realizar sempre a sua execução, pois esse método serve para fechar e liberar imediatamente um objeto Connection.

- **Método isClosed**

Serve para verificar se o objeto Connection está fechado.

- **Método createStatement**

É usado para criar um objeto **Statement** que executa instruções SQL ao banco de dados.



Execução de comandos sql

- `java.sql.Statement`: Permite a execução dos comandos fundamentais de SQL.
 - O método **`Connection.createStatement()`** retorna um objeto `java.sql.Statement`, que representa uma query.
 - O **`Statement.executeQuery()`** retorna um `java.sql.ResultSet` com o resultado obtido (utilizado com `select`).
 - Se a query não retornar nada, utiliza-se o **`Statement.executeUpdate()`** (utilizado com `insert`, `update` e `delete`).



A interface Statement

Nesta interface são listados os métodos **executeQuery** e **executeUpdate** que são considerados os mais importantes referente a execução de uma query.

- **executeQuery:** Executa uma instrução SQL que retorna um único objeto ResultSet.
- **executeUpdate** Executa uma instrução SQL referente a um INSERT, UPDATE e DELETE. Esse método retorna a quantidade de registros que são afetados pela execução do comando SQL.

Interface ResultSet

Representa o conjunto de resultados de uma tabela no banco de dados. Esse objeto mantém o cursor apontando para a sua linha atual de dados, sendo que seu início fica posicionado na primeira linha.

Além disso, esse objeto fornece métodos getters referentes aos tipos de dados como: **getInt**, **getString**, **getDouble**, **getFloat**, **getLong** entre outros. Com esses métodos são possíveis recuperar valores usando, por exemplo, o nome da coluna ou número do índice.

```
Statement st;
```

```
ResultSet res;
```

```
st = con.createStatement();
```

```
res = st.executeQuery("SELECT * FROM Cadastro ORDER BY Nome");
```

```
int codigo = rs.getInt(1);
```

```
String nome = rs.getString(2);
```

```
String sobreNome = rs.getString(3);
```

codigo 1	nome 2	sobrenome 3	idade 4	salario 5
1	João	Flores	32	2132.12
2	Carla	Fachin	35	4332.44