JDBC

O que é JDBC

JDBC significa **J**ava **D**ata**B**ase **C**onnectivity, que é uma API Java padrão para conectividade independente de banco de dados entre a linguagem de programação Java e uma ampla variedade de bancos de dados.

A biblioteca JDBC inclui APIs para cada uma das tarefas mencionadas abaixo que são comumente associadas ao uso do banco de dados.

Fazendo uma conexão com banco de dados.

- Criando SQL ou MySQL statements.
- Executando SQL ou MySQL queries no banco de dados.

Vendo e Modificando resultados do banco

Fundamentalmente, JDBC é uma especificação que fornece um conjunto completo de interfaces que permite acesso portátil a um banco de dados subjacente. Java pode ser usado para escrever diferentes tipos de executáveis, como:

- Java Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs).

Pré-requisitos

Ter uma boa compreensão destes dois assuntos:

• Programação JAVA **Core** (básico)
• SQL ou MySQL Database

BT002

JDBC Arquitetura

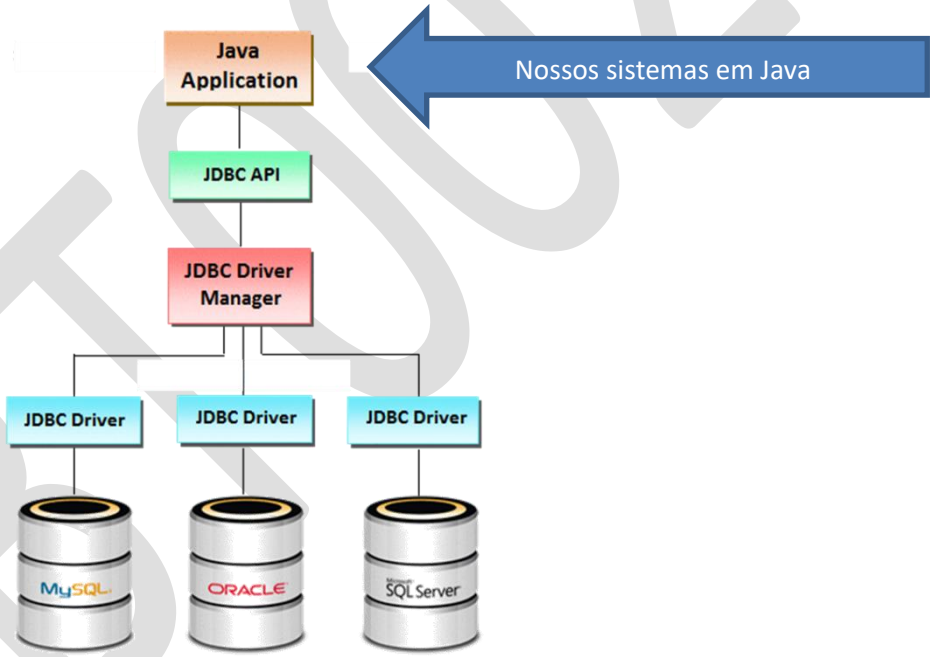
A API JDBC suporta modelos de processamento de **duas** e **três** camadas para acesso ao banco de dados, mas, em geral, a Arquitetura JDBC consiste em duas camadas:

- JDBC API: Isso fornece a conexão do aplicativo para o Gerenciador JDBC.
- JDBC Driver API: Isso suporta a conexão JDBC Manager-to-Driver.

A API JDBC usa um gerenciador de driver e drivers específicos de banco de dados para fornecer conectividade transparente a bancos de dados heterogêneos.

O gerenciador de driver JDBC garante que o driver correto seja usado para acessar cada origem de dados. O gerenciador de driver é capaz de suportar vários drivers simultâneos conectados a vários bancos de dados heterogêneos.

A seguir está o diagrama de arquitetura, que mostra a localização do gerenciador de drivers em relação aos drivers JDBC e ao aplicativo Java:



Nota: Hoje cada empresa de banco de dados disponibiliza o drive JDBC para cada nova versão do seu banco de dados.

BTQ02

Componentes JDBC comuns

A API JDBC fornece as seguintes interfaces e classes:

• **DriverManager**: Esta classe gerencia uma lista de drivers de banco de dados. Corresponde as solicitações de conexão do aplicativo Java com o driver de banco de dados adequado usando o subprotocolo de comunicação. O primeiro driver que reconhece um determinado subprotocolo em JDBC será usado para estabelecer uma conexão com o banco de dados.

Driver: Essa interface trata das comunicações com o servidor de banco de dados. Você irá interagir diretamente com objetos Driver muito raramente. Em vez disso, você usa DriverManager objetos, que gerencia objetos desse tipo. Ele também abstrai os detalhes associados ao trabalho com objetos Driver.

• **Connection**: Esta interface com todos os métodos para entrar em contato com um banco de dados. O objeto de conexão representa o contexto de comunicação, ou seja, toda comunicação com o banco de dados é somente através do objeto de conexão.

• **Statement**: Você usa objetos criados a partir dessa interface para enviar as instruções SQL ao banco de dados. Algumas interfaces derivadas aceitam parâmetros além de executar procedimentos armazenados.

• **ResultSet**: Esses objetos retêm dados recuperados de um banco de dados depois que você executa uma consulta SQL usando objetos Statement. Ele atua como um iterador(laço de repetição) para permitir que você percorra seus dados.

• **SQLException**: Essa classe trata quaisquer erros que ocorrem em um aplicativo de banco de dados.

Pacotes JDBC

O java.sql e o javax.sql são os pacotes primários para JDBC. Oferece as principais classes para interagir com suas fontes de dados.

Os novos recursos desses pacotes incluem alterações nas seguintes áreas:

- Carregamento automático de driver de banco de dados.
- Melhorias no tratamento de exceções.
- Funcionalidade BLOB/CLOB aprimorada.
- Aprimoramentos de interface de conexão e instrução.
- Suporte ao conjunto de caracteres nacionais.
- Acesso SQL ROWID.
- Suporte ao tipo de dados XML do SQL 2003.
- Annotations.

BTOZ

JDBC – SQL Sintaxe

Structured Query Language (SQL) é uma linguagem padronizada que permite executar operações em um banco de dados, como criar, ler, atualizar e excluir dados.

SQL é suportado por quase todos os bancos de dados, ele permite que você escreva o código do banco de dados independentemente do banco de dados em questão.

Abaixo uma breve revisão de banco de dados para que possamos abordar JDBC com tranquilidade.

Criando Banco de Dados

A instrução CREATE DATABASE é usada para criar um novo banco de dados. A sintaxe é:

```
SQL> CREATE DATABASE DATABASE_NAME;
```

A seguinte instrução SQL cria um banco de dados chamado DB_PROJETO1:

```
SQL> CREATE DATABASE DB_PROJETO1;
```

Deletando o banco de dados

A instrução DROP DATABASE é usada para excluir um banco de dados existente, sintaxe:

```
SQL> DROP DATABASE DB_PROJETO1;
```

Nota: Para criar ou descartar um banco de dados, você deve ter privilégios de administrador em seu servidor de banco de dados. Tenha cuidado, a exclusão de um banco de dados elimina todos os dados armazenados, caso não tenha backup não tem como recuperar os dados.

Criando tabela - CREATE TABLE

A instrução CREATE TABLE é usada para criar uma nova tabela. A sintaxe é:

```
SQL> CREATE TABLE
table_name(
   coluna_nome coluna_tipo_dado,
   coluna_nome coluna_tipo_dado,

   coluna_nome coluna_tipo_dado
   ...
);
```

BTQ02

A seguinte instrução SQL cria uma tabela chamada funcionarios com quatro colunas:

```
SQL> CREATE TABLE funcionarios(
   id INT NOT NULL,
   idade INT NOT NULL,
   nome VARCHAR(255),
   sobrenome VARCHAR(255),
   PRIMARY KEY ( id )
);
```

Deletando Tabela - DROP TABLE

A instrução DROP TABLE é usada para excluir uma tabela existente. A sintaxe é:

```
SQL> DROP TABLE table_name;
```

A instrução SQL a seguir exclui uma tabela chamada funcionarios:

```
SQL> DROP TABLE funcionarios;
```

Inserindo dados - INSERT

A sintaxe para INSERT é semelhante à seguinte, em que coluna1, coluna2 e assim por diante representam os novos dados a serem exibidos nas respectivas colunas:

```
SQL> INSERT INTO table_name VALUES (column1, column2, ...);
```

A seguinte instrução SQL INSERT insere uma nova linha no banco de dados funcionarios:

```
SQL> INSERT INTO funcionarios VALUES (1, 18, 'Maria', 'da Silva');
```

Trazendo dados do banco - SELECT

A instrução SELECT é usada para recuperar dados de um banco de dados. A sintaxe para SELECT é:

```
SQL> SELECT column_name, column_name, ...
     FROM table_name
     WHERE
```

A cláusula WHERE pode usar os operadores de comparação como =, !=, <, >, <= e >=, bem como os operadores BETWEEN e LIKE.

A instrução SELECT abaixo seleciona o nome, o sobrenome e a idade da tabela funcionários, onde a coluna id é 1:

```
SQL> SELECT nome, sobrenome, idade
     FROM funcionarios
     WHERE id = 1;
```

A seguinte instrução SQL seleciona o nome, o sobrenome e a idade da tabela funcionarios, onde a coluna nome contém Maria:

```
SQL> SELECT nome, sobrenome, idade

     FROM funcionarios
     WHERE nome LIKE '%Maria%';
```

Atualizando dados - UPDATE

A instrução UPDATE é usada para atualizar dados. A sintaxe para UPDATE é:

```
SQL> UPDATE table_name
     SET column_name = value, column_name = value,
     ...WHERE ...
```

A cláusula WHERE pode usar os operadores de comparação como =, !=, <, >, <= e >=, bem como os operadores BETWEEN e LIKE.

A seguinte instrução SQL UPDATE altera a coluna de idade do funcionário cujo id é 1:

```
     SQL> UPDATE funcionarios SET idade=15 WHERE id=1;
```

DELETE Dados

A instrução DELETE é usada para excluir dados de tabelas. A sintaxe para DELETE é:

```
SQL> DELETE FROM table_name WHERE ...;
```

A cláusula WHERE pode usar os operadores de comparação como =, !=, <, >, <= e >=, bem como os operadores BETWEEN e LIKE.

A seguinte instrução SQL DELETE exclui o registro do funcionário cujo id é 1:


```
SQL> DELETE FROM funcionarios WHERE id=1;
```

BIOZ

JDBC

Para começar a desenvolver com JDBC, você deve configurar seu ambiente JDBC seguindo as etapas mostradas abaixo. Assumimos que você está trabalhando em uma plataforma Windows.

Instalação:

Instale o Java Development Kit

As variáveis de ambiente são definidas conforme descrito abaixo:

• **JAVA_HOME**: C:\Program Files\Java\jdk1.8.0_311

• **PATH**: %JAVA_HOME%\bin

É possível que você já tenha essas variáveis definidas, verificar para ter certeza.

Você obtém automaticamente os pacotes JDBC java.sql e javax.sql, ao instalar Java Development Kit.

Instalar banco de dados

Instale um banco de dados que seja mais adequado para você. Você pode ter muitas opções e as mais comuns são:

• **MySQL DB: MySQL** é um banco de dados de código aberto. Você pode baixá-lo do site oficial do MySQL. Recomendamos baixar a instalação completa do Windows.

Finalmente, baixe e descompacte o **MySQL Connector/J** (o driver MySQL JDBC) em um diretório conveniente. Para o propósito deste tutorial, vamos supor que você instalou o driver em C:\Program Files\MySQL\mysql-connector-java.

• **PostgreSQL DB**: PostgreSQL é um banco de dados de código aberto. Você pode baixá-lo do site oficial do PostgreSQL.

A instalação do Postgres contém uma ferramenta administrativa baseada em GUI chamada pgAdmin III. Drivers JDBC também estão incluídos como parte da instalação.

• **Oracle DB**: Oracle DB é um banco de dados comercial vendido pela Oracle. Assumimos que você possui a mídia de distribuição necessária para instalá-lo.

A instalação do Oracle inclui uma ferramenta administrativa baseada em GUI chamada Enterprise Manager. Drivers JDBC também estão incluídos como parte da instalação.

BTO02

Instalando Drivers do Banco de Dados

O JDK mais recente inclui um driver JDBC-ODBC Bridge que disponibiliza a maioria dos drivers Open Database Connectivity (ODBC) para programadores que usam a API JDBC.

Hoje em dia, a maioria dos fornecedores de banco de dados está fornecendo drivers JDBC apropriados junto com a instalação do banco de dados, tornando mais fácil configurar sem precisar baixar nada da internet.

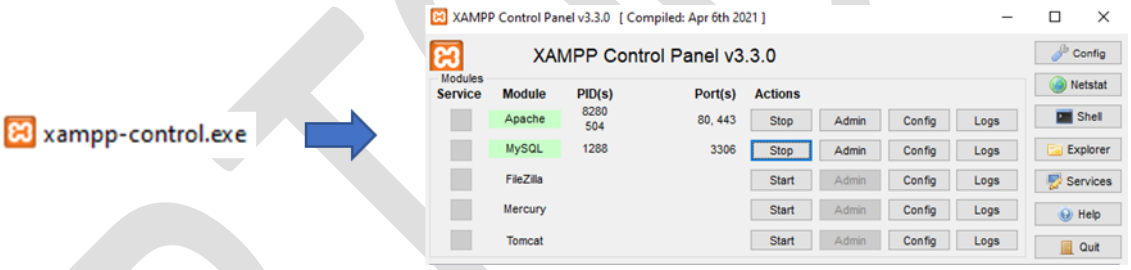
JDBC com MySQL

Quando instalamos o MySQL na nossa máquina ele pede para definirmos uma senha para o usuário root, no nosso caso vamos utilizar o Xampp e ele já configura o usuário root sem senha.

Criando Banco de Dados Database

Para criar o banco de dados do **DB_PROJETO1**, use as seguintes etapas:

Passo 1 Ligar o MySQL no XAMPP



Passo 2

Abra um prompt de comando e mude para o diretório de instalação do MySQL da seguinte forma:

```
C:\>cd C:\xampp\mysql\bin

C:\xampp\mysql\bin>
```

Nota: O caminho para mysql.exe pode variar dependendo do local de instalação do Xampp ou MySQL em seu sistema.

Passo 3

Logue no servidor de banco de dados executando o seguinte comando:

```
C:\Program Files\MySQL\bin>mysql.exe -u root -p

Enter password:    // Tecle enter novamente, assim loga com senha vazia
mysql>create database DB_PROJETO1;
```

Criando uma Tabela

Para criar a tabela de funcionais no banco de dados **DB_PROJETO1**, siga os seguintes passos:

Passo 1

Crie a tabela funcionarios da seguinte forma:

```
mysql> use DB_PROJETO1;
mysql> create table funcionarios
    -> (
    -> id int(11) not null,
    -> idade int(11) not null,
    -> nome varchar (255),
    -> sobrenome varchar (255)
    -> );
Query OK, 0 rows affected (0.08 sec)
mysql>
```

Inserindo dados no banco de dados

Finalmente você cria alguns registros na tabela funcionarios da seguinte forma:

```
mysql> INSERT INTO funcionarios VALUES (2, 15, 'Ali', 'Babaa');
 Query OK, 1 row affected (0.05 sec)
mysql> INSERT INTO funcionarios VALUES (3,  2500, 'Matu', 'Zalem');
 Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO funcionarios VALUES (4, 30, 'Gato', 'de Botas');
 Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO funcionarios VALUES (5, 60, 'Professor', 'Xavier');
 Query OK, 1 row affected (0.00 sec)
```

Para um entendimento completo sobre o banco de dados MySQL, estude o Tutorial MySQL.

https://dev.mysql.com/doc/refman/8.0/en/tutorial.html

JDBC – Exemplo de código CRUD

Vamos criar um aplicativo com um JDBC simples. Isso mostrará como abrir uma conexão com o banco de dados, inserir dados, atualizar, executar uma consulta SQL e exibir os resultados.

Obs.: Baixar o Drive e adicionar no Eclipse:

https://downloads.mysql.com/archives/c-j/

Criando aplicativo JDBC

Seguem oito etapas envolvidas na construção de um aplicativo CRUD com JDBC:

1. **Importando os pacotes**: Requer que você inclua os pacotes que contêm as classes JDBC necessárias para a programação do banco de dados. Na maioria das vezes, usar import java.sql.* será suficiente.
2. **Registre o driver JDBC**: Requer que você inicialize um driver para poder abrir um canal de comunicação com o banco de dados.
3. **Abra uma conexão**: Requer o uso do método DriverManager.getConnection() para criar um objeto Connection, que representa uma conexão física com o banco de dados.
4. **Criando caminho**: Criar o Statement para executar as instrução SQL no banco de dados.
5. **Inserir dados**: Instrução INSERT SQL sobre o Statement.
6. **Atualizar dados**: Instrução UPDATE SQL sobre o Statement.
7. **Deletar**: Instrução DELETE SQL sobre o Statement.
8. **Consultar e Extrair dados**: ResultSet para recuperar os dados do conjunto de resultados fornecidos pelo banco de dados.
9. **Limpe o ambiente**: Requer fechar explicitamente todos os recursos do banco de dados liberando recursos do sistema e uso de memória da aplicação.

BIOZ

Exemplo de Codigo CRUD com JDBC

```
// Crie uma classe “ExemploCRUD_JDBC” no Eclipse e Copie e cole o exemplo a seguir

package exemplo;

//Passo 1. Import required packages
import java.sql.*;

public class ExemploCRUD_JDBC {
	//JDBC nome do drive e URL do banco de dados
	static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
	static final String DB_URL = "jdbc:mysql://localhost/db_projeto1";

	// Usuário e senha do banco de dados
	static final String USUARIO = "root";
	static final String SENHA = ""; // Quando instala o banco ele pede para criar a senha, no xampp vem sem senha

	public static void main(String[] args) {
		Connection conn = null;
		Statement stmt = null;
		try {
			//Passo 2: Registrar o drive JDBC
			Class.forName("com.mysql.cj.jdbc.Driver");

			//Passo 3: Abrir a conexão com o banco de dados
			System.out.println("Conectando o banco de dados...");
			conn = DriverManager.getConnection(DB_URL, USUARIO, SENHA);

			//Passo 4: Criando caminho para executar comandos SQL
			System.out.println("Criando statement...");
			stmt = conn.createStatement();

			//Passo 5: Inserir dados
			System.out.println("Criando statement...");
			stmt = conn.createStatement();
			String sql = "INSERT INTO funcionarios (id, idade, nome, sobrenome) VALUES (7, 50, 'Jammes', 'Bond')";
			stmt.execute(sql);

			//Passo 6: Atualizar dados
			sql = "UPDATE funcionarios SET nome='James' WHERE id='7'";
			stmt.execute(sql);

			//Passo 7: Deletar dados
			sql = "DELETE FROM funcionarios WHERE id = 1";
			stmt.execute(sql);

			//Passo 8: Consultar e Extrair dados do ResultSet
			sql = "SELECT * FROM funcionarios";
			ResultSet rs = stmt.executeQuery(sql);
			while (rs.next()) {
				//Recuperar dados pelo nome da coluna
				int id = rs.getInt("id");
				int age = rs.getInt("idade");
				String first = rs.getString("nome");
				String last = rs.getString("sobrenome");
				//Display values
				System.out.print("ID: " + id);
				System.out.print(", Idade: " + age);
				System.out.print(", Nome: " + first);
				System.out.println(", Sobrenome: " + last);
			}

			//Passo 9: Fechando conexão com o banco de dados
			rs.close();
			stmt.close();
			conn.close();
		} catch (SQLException e) {
			//Tratando erros tipo "SQLException"
			e.printStackTrace();
		} catch (Exception e) {
			//Tratando erros tipo "Class.forName"
			e.printStackTrace();
		} finally {
			//bloco finally usado para fechar as conexões
			try {
				if (stmt != null)
					stmt.close();
			} catch (SQLException e) {
				e.printStackTrace();
			}

			try {
				if (conn != null)
					conn.close();
			} catch (SQLException e) {
				e.printStackTrace();
			}
		}
		System.out.println("Fim do sistema");
	}
	}
```

Este exemplo pode servir como um modelo quando você precisar criar seus outros aplicativos com acesso à banco de dados com JDBC.