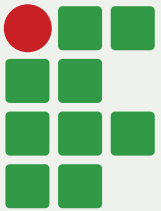
JDBC - Java Database Connectivity

Linguagem de Programação 1

Prof. Fábio José Rodrigues Pinheiro

Janeiro de 2022



INSTITUTO FEDERAL
Catarinense
Campus Videira

JDBC

- JDBC foi concebido para oferecer acesso universal a dados para programas escritos em Java.
- O pacote JDBC (java.sql) provê classes conexão a uma grande variedade de SGBDs (Oracle, SQLServer, MySQL, PostgreSQL, DB2, ODBC, Firebird, etc.).
- Além disso, provê classes para inserção, atualização e consulta de dados, através de queries sql.

Arquitetura JDBC

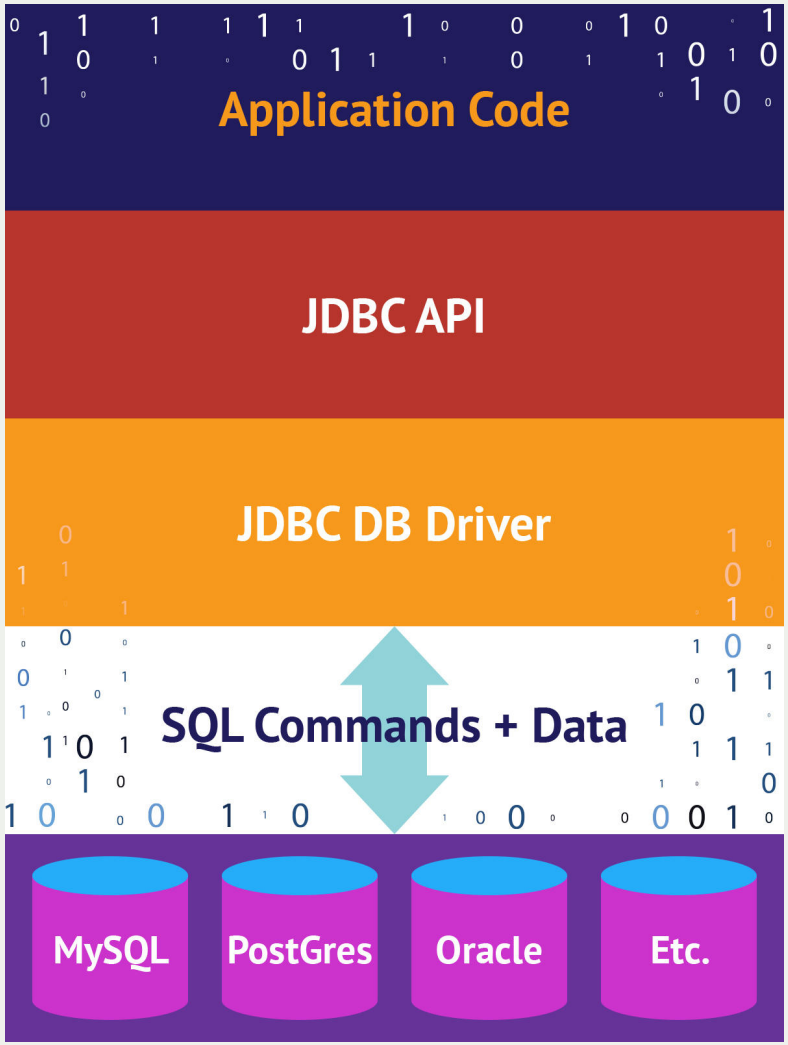


Figura: Fonte: infoworld.com

Conexão com Banco de dados

- Antes da utilização efetiva de uma determinada base de dados, se faz necessário **criar uma conexão** com esta.
- A conexão é feita em duas etapas:
  - **Carga do driver** relativo ao tipo do banco.
  - **Efetuar a conexão** através da passagem de parâmetros essenciais.

Driver de Conexão

- Antes de iniciar uma conexão, é preciso carregar a classe do driver na aplicação que irá utilizá-lo:

```
1    Class.forName("org.postgresql.Driver");
```

- Outros drivers:
  - PostgreSQL: org.postgresql.Driver
  - Oracle: oracle.jdbc.driver.OracleDriver
  - MySQL: com.mysql.cj.jdbc.Driver
  - Firebird: org.firebirdsql.jdbc.FBDriver
- Os principais fabricantes de BDs já fornecem um pacote (.jar) para conexão JDBC.

Estabelecendo Conexão

- A classe *DriverManager* manipula objetos do tipo *Driver*
  - Possui métodos para registrar drivers, removê-los ou listá-los.
  - É usado para retornar Connection, que representa uma conexão a um banco de dados, a partir de uma URL JDBC recebida como parâmetro

```
1 Connection con = DriverManager.getConnection(url, usuario, senha);
```

- A URL de conexão é estabelecida pelo formato abaixo:

```
1     jdbc:<subprotocolo>:<dsn>
```


Estabelecendo Conexão

- O **subprotocolo** identifica e seleciona o driver/banco de dados a ser instanciado.
- O **dsn** (*data source name*) é o nome/caminho que o subprotocolo utilizará para localizar um determinado servidor ou base de dados.
  - O **dsn** é composto de: IP:porta + nome da base de dados
  - A sintaxe dependente do fabricante.
- Alguns exemplos de URLs:

```
1    jdbc:odbc:anuncios
2    jdbc:postgresql://192.168.2.223:5432/academico
3    jdbc:oracle:thin:@200.206.192.216:1521:exemplo
4    jdbc:mysql://alnitak.orion.org/clientes
5    jdbc:sqlserver://127.0.0.1/minhabase
```


Principais Interfaces JDBC

- Interfaces implementadas nos drivers JDBC.
  - *Connection*
    - Representa uma conexão ao banco de dados, que é retornada pelo DriverManager na forma de um objeto.
  - *Statement*
    - Oferece meios de passar instruções SQL para o sistema de bancos de dados.
    - A interface *PreparedStatement*, que herda de *Statement*,apresenta uma solução mais robusta para manipular o banco de dados com instruções SQL
  - *ResultSet*
    - É um objeto para os dados recebidos. Recebe resultados de consultas ao banco de dados.

Statement

- Através do objeto *Connection*, chama-se sobre ele o método *createStatement()* para obter um objeto do tipo *Statement*:

```
1 Statement stmt = con.createStatement()
```

- Através do objeto criado é possível utilizar métodos como *execute(), executeQuery() e executeUpdate()* para enviar instruções SQL ao BD.
- A interface *PreparedStatement* também é criada através do objeto de *Connection*, mas possui o funcionamento um pouco diferente

```
1 PreparedStatement pstmt = con.prepareStatement(sql)
```


Statement - Exemplos

```
1 //SQL de criação da tabela "disciplinas" com o método
2 stmt.execute("CREATE TABLE disciplinas (codigo INT PRIMARY KEY,
      descricao CHAR(20));");
3
4 //Inserção de dados na tabela "disciplina;
5 //executeUpdate() retorna o número de linhas afetadas na tabela
6 int num = stmt.executeUpdate("INSERT INTO disciplinas (codigo,
      descricao) VALUES (183,LPG1);");
7
8 //Consulta de todos os dados na tabela "disciplinas"
9 ResultSet rs = stmt.executeQuery("SELECT * FROM disciplinas;");
```


PreparedStatement - Exemplos

```
1  //Primeiro é montado o SQL com os valores substituídos por "?"
2  String sql = "INSERT INTO livros VALUES(?, ?, ?)";
3
4  /*Em seguida, cria-se o objeto PreparedStatement
5  passando o SQL como parâmetro*/
6  PreparedStatement pstmt = con.prepareStatement(sql);
7
8  /*A partir desse momento, é possível fazer o "set" dos valores
9  que serão inseridos, respeitando o tipo de dado definido no BD*/
10 pstmt.setInt(1, 18943);
11 pstmt.setString(2, "Gabriel Garcia Márquez");
12 pstmt.setString(3, "Cem anos de solidão");
13
14 //Por fim, o SQL é executado
15 pstmt.executeUpdate();
```


ResultSet

- O método *executeQuery()*, da interface *Statement*, retorna um objeto *ResultSet*.
  - Cursor para as linhas de uma tabela
  - Pode-se navegar pelas linhas da tabela recuperar as informações armazenadas nas colunas
- Os métodos de navegação são:

```
1 next(), previous(), absolute(), first() e last()
```

- Alguns métodos para obtenção de dados:

```
1 getInt (int índice) ou getInt (String nome coluna)
2 getString (int índice) ou getString (String nome coluna)
3 getDate (int índice) ou getDate (String nome coluna)
```


ResultSet - Exemplo

```
1 ResultSet rs = stmt.executeQuery("SELECT cod, texto, data FROM anuncio"
      );
2 while (rs.next()) {
3   int c = rs.getInt("codigo");
4   String t = rs.getString("texto");
5   Date d = rs.getDate("data");
6   ...
7 }
```


Encerrando Conexões

- Após o uso, os objetos *Connection*, *Statement* e *ResultSet* devem ser fechados.
- Isto pode ser feito com o método *close()*:

```
1 con.close();
2 stmt.close();
3 rs.close();
```

- A exceção SQLException é a principal exceção a ser observada em aplicações JDBC

Padrões de Projeto para Banco de Dados

DAO - *Data Access Object*

- DAO é um padrão de projeto responsável por encapsular operações de acesso a dados em classes dedicadas exclusivamente a isso, de modo a evitar que o código de interação com o banco de dados fique espalhado pelo código da sua aplicação.
- Classe responsável por toda interface com o banco de dados da aplicação
  - inserção
  - remoção
  - atualização
  - consulta

Leitura complementar

Caelum

Apostila Caelum FJ-21 Java Web

disponível na pasta compartilhada na página da disciplina

- Capítulo 2 - Banco de Dados e JDBC