Java Acesso e manipulação de dados

RECODE pro

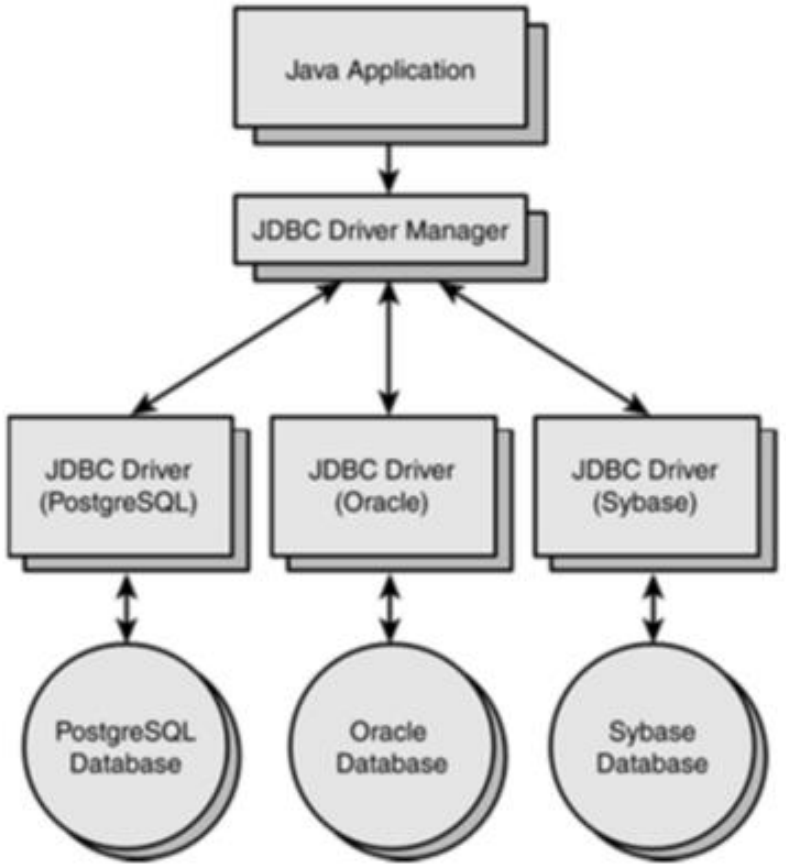
Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

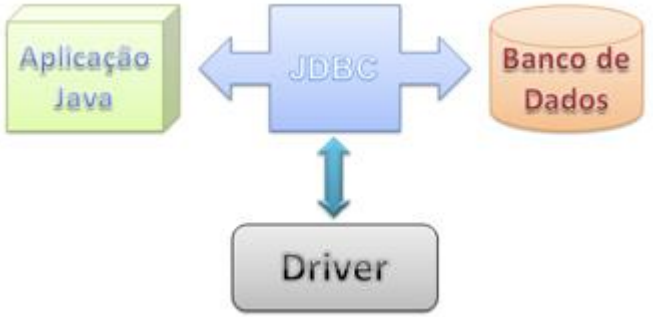
Banco de dados com Java, JDBC e MySql

Conexão com bancos de dados usando JDBC

O que é **JDBC**? 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.

https://www.devmedia.com.br/aprendendo-java-com-jdbc/29116#:~:text=O%20que%20%C3%A9%20JDBC%3F,de%20banco%20de%20dados%20relacional





Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

Banco de dados com Java, JDBC e MySql

Conexão com bancos de dados usando JDBC

CRUD

Uma vez que tivermos um editor, começamos a adicionar a funcionalidade CRUD. Primeiro, o "R" que indica "Read" (ler) é manipulado pelo visualizador acima descrito. A seguir, é manipulado o "U" para "Update" (atualizar), seguido pelo "C" para "Create" (criar) e pelo "D" para "Delete" (excluir).

C para "Create" (criar)

R que indica "Read" (ler)

U para "Update" (atualizar)

D para "Delete" (excluir)

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

Nosso projeto usando JDBC e banco de dados Mysql

Vamos implementar um projeto usando o console do JAVA com acesso a banco de dados aplicando as quatro operações básicas – (CRUD) insert, update, delete, select

1º passo – criar o banco de dados no Workbench

```
create database crud;
Use crud;
create table contato(
    id int primary key auto_increment,
    nome varchar(50),
    idade int,
    dataCadastro Date
);
```

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

2º passo – criar a classe contato para representar o objeto que será enviado ao banco de dados.

```
import java.util.Date;

public class Contato {

        private int id;
        private String nome;
        private int idade;
        private Date dataCadastro;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getNome() {
                return nome;
        }

        public void setNome(String nome) {
                this.nome = nome;
        }

        public int getIdade() {
                return idade;
        }

        public void setIdade(int idade) {
                this.idade = idade;
        }

        public Date getDataCadastro() {
                return dataCadastro;
        }

        public void setDataCadastro(Date dataCadastro) {
                this.dataCadastro = dataCadastro;
        }
}
```

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

3º passo – criar a classe conexão com o banco de dados

```
import java.sql.Connection;
import java.sql.DriverManager;

public class Conexao {

    //Nome do usuário do mysql
    private static final String USERNAME = "root";

    //Senha do mysql
    private static final String PASSWORD = "120202";

    //Dados de caminho, porta e nome da base de dados que irá ser feita a
conexão
    private static final String DATABASE_URL =
"jdbc:mysql://localhost:3306/crud";
```

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

3º passo – criar a classe conexão com o banco de dados

```
//    * Cria uma conexão com o banco de dados MySQL utilizando o nome de
usuário e senha fornecidos
//    * @param username
//    * @param senha
//    * @return uma conexão com o banco de dados
//    * @throws Exception

    public static Connection createConnectionToMySQL() throws Exception{

        //Cria a conexão com o banco de dados
        Connection connection = DriverManager.getConnection(DATABASE_URL,
USERNAME, PASSWORD);

        return connection;
    }
}
```

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

4º passo – criar a classe contatoDAO para gerenciar as operações CRUD do banco de dados. MÉTODO SAVE

```
import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class ContatoDAO {

        public void save(Contato contato) {

                // Isso é uma sql comum, os ? são os parâmetros que nós vamos adicionar na base
                // de dados


                String sql = "INSERT INTO contato(nome,idade,dataCadastro)" + " VALUES(?,?,?)";


                Connection conn = null;

                PreparedStatement pstm = null;
```

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

4º passo – criar a classe contatoDAO para gerenciar as operações CRUD do banco de dados. MÉTODO SAVE

```
try {
        // Cria uma conexão com o banco
        conn = Conexao.createConnectionToMySQL();

        // Cria um PreparedStatment, classe usada para executar a query
        pstm = conn.prepareStatement(sql);

        // Adiciona o valor do primeiro parâmetro da sql
        pstm.setString(1, contato.getNome());
        // Adicionar o valor do segundo parâmetro da sql
        pstm.setInt(2, contato.getIdade());
        // Adiciona o valor do terceiro parâmetro da sql
        pstm.setDate(3, new Date(contato.getDataCadastro().getTime()));

        // Executa a sql para inserção dos dados
        pstm.execute();
```

4º passo – criar a classe contatoDAO para gerenciar as operações CRUD do banco de dados. MÉTODO SAVE

```
        } catch (Exception e) {
                        e.printStackTrace();
                } finally {
                        // Fecha as conexões
                        try {
                                if (pstm != null) {

                                        pstm.close();
                                }

                                if (conn != null) {
                                        conn.close();
                                }

                        } catch (Exception e) {

                                e.printStackTrace();
                        }
                }
        }
```

Módulo 03 – Java Acesso e Manipulação de Dados - Aula 01

4º passo – criar a classe contatoDAO para gerenciar as operações CRUD do banco de dados. MÉTODO REMOVE

```
public void removeById(int id) {

        String sql = "DELETE FROM contato WHERE id = ?";

        Connection conn = null;
        PreparedStatement pstm = null;

        try {
                conn = Conexao.createConnectionToMySQL();

                pstm = conn.prepareStatement(sql);

                pstm.setInt(1, id);

                pstm.execute();

        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
```

4º passo – criar a classe contatoDAO para gerenciar as operações CRUD do banco de dados. MÉTODO REMOVE

```
        } finally {

                        try {
                                if (pstm != null) {

                                        pstm.close();
                                }

                                if (conn != null) {
                                        conn.close();
                                }

                        } catch (Exception e) {

                                e.printStackTrace();
                        }
                }
        }
```

RECODE

www.recode.org.br

RECODE pro

recodepro.org.br

Institucional

/rederecode

/recoderede