

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

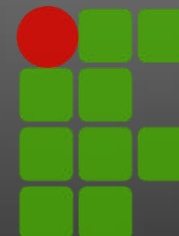
Programação com Acesso a BD

Data Access Objetc

Pedro Baesse (pedro.baesse@ifrn.edu.br)

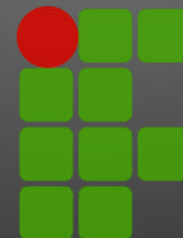
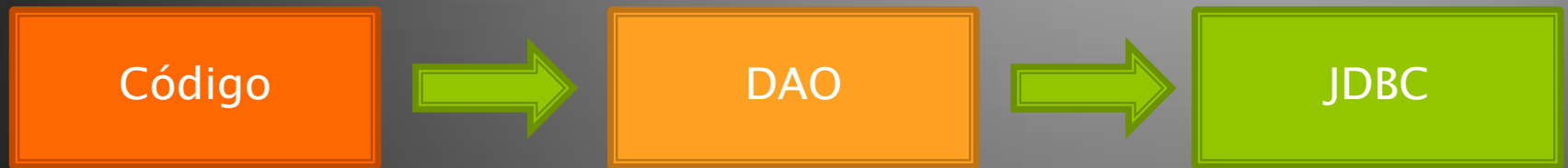
Agenda

- ▶ Conceito
- ▶ Vantagens
- ▶ Como usar



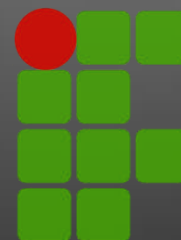
Data Access Objetc (DAO)

- ▶ É um padrão utilizado para abstrair a persistência de dados. Assim as regras de negócio ficam separadas do acesso ao banco de dados



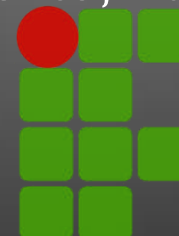
Vantagens do DAO

- ▶ Age como intermediário separando a aplicação e a persistência
 - Podem ser alteradas independentemente
- ▶ Esconde os detalhes da persistência do resto da aplicação
- ▶ Pode ser usada em uma grande variedade de aplicações – qualquer local onde armazenamento seja necessário (mesmo que não seja banco de dados)



Como usar

1. Criar uma classe `ConnectionFactory`
 - Responsável por retornar um objeto conexão
2. Criar a classe `Modelo`
 - Classe com os atributos, getters e setters
3. Criar a classe `ModeloDAO`
 - Responsável pela persistência da modelo
4. Utilizar a classe `ModeloDAO` na aplicação
 - Criar um instância de `ModeloDAO` e utilizar
 - Modelo deve ser entendido como qualquer entidade que se deseje fazer persistência
 - Ex: Estudante, Empresa, Departamento, Turma...



1. Criar uma classe ConnectionFactory

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

public class ConnectionFactory {
    String driverClassName = "com.mysql.jdbc.Driver";
    String connectionUrl = "jdbc:mysql://localhost/basededados";
    String dbUser = "root";
    String dbPwd = "root";

    private static ConnectionFactory connectionFactory = null;

    private ConnectionFactory() {
        try {
            Class.forName(driverClassName);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

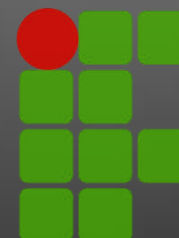
    public Connection getConnection() throws SQLException {
        Connection conn = null;
        conn = DriverManager.getConnection(connectionUrl, dbUser, dbPwd);
        return conn;
    }

    public static ConnectionFactory getInstance() {
        if (connectionFactory == null) {
            connectionFactory = new ConnectionFactory();
        }
        return connectionFactory;
    }
}
```



2. Criar a classe Modelo

```
public class Modelo{  
    int id;  
    String nome;  
    String endereco;  
  
    public Modelo() {  
  
    }  
  
    public Modelo(int id, String nome, String endereco) {  
        this.id = id;  
        this.nome = nome;  
        this.endereco = endereco;  
    }  
  
    //getters e setters
```



3. Criar a classe ModeloDAO

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class ModeloDAO {
    Connection connection = null;
    PreparedStatement ptmt = null;
    ResultSet resultSet = null;

    public ModeloDAO() {
    }

    private Connection getConnection() throws SQLException {
        Connection conn;
        conn = ConnectionFactory.getInstance().getConnection();
        return conn;
    }

    public int add(Modelo modelo) {
        try {
            String queryString = "INSERT INTO modelo(id, Nome, Endereco) VALUES(?,?,?)";
            connection = getConnection();
            ptmt = connection.prepareStatement(queryString);
            ptmt.setInt(1, modelo.getId());
            ptmt.setString(2, modelo.getNome());
            ptmt.setString(3, modelo.getCourse());
            return ptmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            finishConnetion();
        }
    }
}
```



3. Criar a classe ModeloDAO

```
public void finishConnetion(){
    try {
        if (ptmt != null)
            ptmt.close();
        if (connection != null)
            connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public int update(Modelo modelo) {
    //Código segue a lógica de método add, porém atualiza um Modelo
}

public int delete(int id) {
    //Código segue a lógica de add, porém apaga uma tupla de acordo com id
}

public Modelo find(int id){
    //Código segue a lógica de add, porém retorna um objeto Modelo
}

public List<Modelo> findAll() {
    //Código segue a lógica de add, porém retorna uma lista de objetos Modelo
}
}
```



3. Criar a classe ModeloDAO

```
public void finishConnetion(){
    try {
        if (ptmt != null)
            ptmt.close();
        if (connection != null)
            connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public int update(Modelo modelo) {
    //Código segue a lógica de método add, porém atualiza um Modelo
}

public int delete(int id) {
    //Código segue a lógica de add, porém apaga uma tupla de acordo com id
}

public Modelo find(int id){
    //Código segue a lógica de add, porém retorna um objeto Modelo
}

public List<Modelo> findAll() {
    //Código segue a lógica de add, porém retorna uma lista de objetos Modelo
}
}
```



3. Criar a classe ModeloDAO

```
public void finishConnetion(){
    try {
        if (ptmt != null)
            ptmt.close();
        if (connection != null)
            connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public int update(Modelo modelo) {
    //Código segue a lógica de método add, porém atualiza um Modelo
}

public int delete(int id) {
    //Código segue a lógica de add, porém apaga uma tupla de acordo com id
}

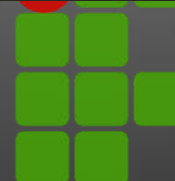
public Modelo find(int id){
    //Código segue a lógica de add, porém retorna um objeto Modelo
}

public List<Modelo> findAll() {
    //Código segue a lógica de add, porém retorna uma lista de objetos Modelo
}
}
```



4. Utilizar ModeloDAO na aplicação

```
public class Teste{  
    public static void main(String[] args) {  
        ModeloDAO modeloDAO = new ModeloDAO();  
        Modelo exemplo = new Modelo();  
        exemplo.setNome("Exemplo");  
        exemplo.setId(8);  
        exemplo.setEndereco("Dinamarca");  
  
        Modelo intercambista = new Modelo();  
        intercambista.setNome("Rasmus");  
        intercambista.setId(3);  
        //Adicinando um objeto  
        modeloDAO.add(exemplo);  
        // Apagando uma tupla  
        modeloDAO.delete(7);  
        // Atualizando uma tupla  
        modeloDAO.update(intercambista);  
        // Buscando somente um objeto  
        modeloDAO.find(2)  
        // Buscando todas os objetos  
        ArryList<Modelo> lista = modeloDAO.findAll();  
    }  
}
```



Dúvidas

