JAVA: JDBC e DAO

Prof. Dr. Razer A N R Montaño
2020

1

JDBC

- Biblioteca de acesso a banco de dados
- Importações a partir de **`java.sql.*`**

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

- Usa-se linguagem SQL para manipulação dos dados
- Dois tipos de comandos:
  - Consulta : quando se executa SELECT
  - Atualização: quando se executa INSERT, UPDATE e DELETE

2

Consultas em JDBC

```
List<String> lista = new ArrayList<String>();
String url = "jdbc:postgresql://localhost:5432/banco";
String sql = "SELECT nome FROM tb_pessoa";
String login = "postgres";
String senha = "postgres";
try {
   Class.forName("org.postgresql.Driver");
}
catch (ClassNotFoundException e) {
   System.out.println("Driver do PostgreSQL não instalado.");
   e.printStackTrace();
   return;
}
```

REGISTRA O DRIVER

Para Java Web este passo é necessário

Prof. Dr. Razer A N R Montaño

JDBC e DAO

3

3

Consultas em JDBC.

```
try (Connection con = DriverManager.getConnection(url, login, senha);
     PreparedStatement st = con.prepareStatement(sql)) {

    // O ResultSet está em outro try para permitir a
    // atribuição de parâmetros na query com
    // st.setString(), st.setDate(), etc..
    try (ResultSet rs = st.executeQuery()) {
       while (rs.next()) {
          lista.add(rs.getString("nome"));
       }
    }
}
catch(SQLException e) {
   e.printStackTrace();
}
return lista;
```

Prof. Dr. Razer A N R Montaño

JDBC e DAO

4

4

Atualizações em JDBC

```
String url = "jdbc:postgresql://localhost:5432/banco";
String sql = "insert into tb_pessoa values (?, ?)";
String login = "postgres";
String senha = "postgres";
try {
   Class.forName("org.postgresql.Driver");
}
catch (ClassNotFoundException e) {
   System.out.println("Driver do PostgreSQL não instalado.");
   e.printStackTrace();
   return;
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
5

5

Atualizações em JDBC..

```
try (Connection con = DriverManager.getConnection(url, login, senha);
     PreparedStatement st = con.prepareStatement(sql)) {

   st.setString(1, "Razer");
   st.setString(2, "razer@razer");
   st.executeUpdate();

}
catch(Exception e) {
   e.printStackTrace();
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
6

6

DAO – DATA ACCESS OBJECTS

7

DAO.

- DAO : *Data Access Objects*
- Um dos padrões usados para acesso a dados em aplicações
- Abstrai cada entidade como uma tabela no banco
- Cada entidade tem seu DAO
  - Ex. Pessoa e PessoaDAO
- Outros padrões:
  - **Repository** : Abstrai os dados como coleções em memória. Pode ser implementado usando, por exemplo, DAOs
  - **Active Record**: na classe da entidade tem-se os métodos de acesso aos dados. Quebra o Princípio da Responsabilidade Única

8

Organização do DAO.

- Pacotes
  - **exception** : Contém as exceções criadas para a aplicação
    - **DAOException** – Classe de exceção que desacopla os erros de SQL. Adiciona mensagens de erro mais amigáveis
    - **RemoverPessoaException** – Classe de exceção que desacopla de erros genéricos
  - **dao** : Contém as classes de DAO
    - **DAO<T>** – Interface genérica que todo DAO implementa. Força todos os DAOs a implementarem, no mínimo, os métodos aqui definidos
    - **PessoaDAO** – Classe de acesso ao banco de dados, todos os códigos JDBC estarão em métodos desta classe
    - **ConnectionFactory** – Classe que cria as conexões com o banco de dados
  - **domain** : Contém todas as classes de domínio
    - **Pessoa** – classe que contém os dados a serem persistidos, geralmente equivale a uma tabela do banco de dados, mas seus atributos não possuem os mesmos nomes que os dos campos do BD
  - **teste** : Contém classe de teste

9

Tabela no PostgreSQL.

```
create table tb_pessoa (
     id_pessoa serial PRIMARY KEY,
     nm_pessoa character varying(50) NOT NULL,
     dt_pessoa date NOT NULL
)
```


10

Classe de Domínio: Pessoa

```
public class Pessoa implements Serializable {
    private int              id;
    private String           nome;
    private java.util.Date   dataNascimento;

    public Pessoa() { }

    public Pessoa(String nome, Date dataNascimento) {
        this.nome           = nome;
        this.dataNascimento = dataNascimento;
    }

    public Pessoa(int id, String nome, Date dataNascimento) {
        this.id             = id;
        this.nome           = nome;
        this.dataNascimento = dataNascimento;
    }
```


11

Classe de Domínio: Pessoa.

```
    public String toString() {
        return "id=" + id + " / nome=" + this.nome +
               " / dataNascimento=" + this.dataNascimento;
    }

    // setters/getters

}
```


12

Classe DAOException.

```
public class DAOException extends Exception {
    public DAOException() {
    }
    public DAOException(String string) {
        super(string);
    }
    public DAOException(String string, Throwable thrwbl) {
        super(string, thrwbl);
    }
}
```


13

Classe ConnectionFactory

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


public class ConnectionFactory implements AutoCloseable {
    private static String DRIVER = "org.postgresql.Driver";
    private static String URL    = "jdbc:postgresql://localhost:5432/razer";
    private static String LOGIN  = "postgres";
    private static String SENHA  = "postgres";

    private Connection con = null;
```


14

Classe ConnectionFactory

```
public Connection getConnection() throws DAOException {
    if (con == null) {
        try {
            Class.forName(DRIVER);
            con = DriverManager.getConnection(URL, LOGIN, SENHA);
        }
        catch(ClassNotFoundException e) {
            throw new DAOException("Driver do banco não encontrado: " +
                                   DRIVER, e);
        }
        catch(SQLException e) {
            throw new DAOException("Erro conectando ao BD: " + URL + "/" +
                                   LOGIN + "/" + SENHA, e);
        }
    }
    return con;
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
15

15

Classe ConnectionFactory.

```
@Override
public void close() {
    if (con!=null) {
        try {
            con.close();
            con = null;
        }
        catch(Exception e) {
            System.out.println("Erro fechando a conexão. IGNORADO");
            e.printStackTrace();
        }
    }
}
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
16

16

Interface Genérica DAO.

```
import exceptions.DAOException;
import java.util.List;

public interface DAO<T> {
    T buscar(long id) throws DAOException;
    List<T> buscarTodos() throws DAOException;
    void inserir(T t) throws DAOException;
    void atualizar(T t) throws DAOException;
    void remover(T t) throws DAOException;
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
17

17

Classe PessoaDAO

```
public class PessoaDAO implements DAO<Pessoa> {
    private static final String QUERY_INSERIR =
              "INSERT INTO tb_pessoa (nm_pessoa, dt_pessoa) VALUES (?, ?)";
    private static final String QUERY_BUSCAR_TODOS =
              "SELECT id_pessoa, nm_pessoa, dt_pessoa FROM tb_pessoa";

    private Connection con = null;

    public PessoaDAO(Connection con) throws DAOException {
        if (con == null) {
            throw new DAOException("Conexão nula ao criar PessoaDAO.");
        }
        this.con = con;
    }
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
18

18

Classe PessoaDAO

```
@Override
public void inserir(Pessoa p) throws DAOException {

    try (PreparedStatement st = con.prepareStatement(QUERY_INSERIR)) {
       st.setString(1, p.getNome());
       st.setDate(2, new java.sql.Date(
                                 p.getDataNascimento().getTime()));
       st.executeUpdate();
    }
    catch(SQLException e) {
       throw new DAOException("Erro inserindo pessoa: " +
                              QUERY_INSERIR +
                              "/ " + p.toString(), e);
    }
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
19

19

Classe PessoaDAO

```
@Override
public List<Pessoa> buscarTodos() throws DAOException {
    List<Pessoa> lista = new ArrayList<>();
    try (PreparedStatement st = con.prepareStatement(QUERY_BUSCAR_TODOS);
         ResultSet rs = st.executeQuery()) {
        while (rs.next()) {
           Pessoa p = new Pessoa();
           p.setId(rs.getInt("id_pessoa"));
           p.setNome(rs.getString("nm_pessoa"));
           p.setDataNascimento(rs.getDate("dt_pessoa"));
           lista.add(p);
        }
        return lista;
    }
    catch(SQLException e) {
       throw new DAOException("Erro buscando todas as pessoas: " +
                              QUERY_BUSCAR_TODOS, e);
    }
}
```

Prof. Dr. Razer A N R Montaño
JDBC e DAO
20

20

Classe PessoaDAO.

```
    @Override
    public Pessoa buscar(long id) throws DAOException {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void atualizar(Pessoa p) throws DAOException {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void remover(Pessoa p) throws DAOException {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```


21

Teste do DAO..

```
public class TesteDAO {
    public static void main(String[] args) {
        try (ConnectionFactory factory = new ConnectionFactory()) {
            Pessoa p = new Pessoa();
            p.setNome("Razer");
            p.setDataNascimento(new java.util.Date());

            PessoaDAO dao = new PessoaDAO(factory.getConnection());
            dao.inserir(p);

            List<Pessoa> lista = dao.buscarTodos();
            lista.forEach(pessoa -> System.out.println(pessoa.toString()));
        }
        catch(DAOException e) {
            System.out.println("#### ERRO DE DAO: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```


22