
FIAP

Domain Driven Design
using Java

AGENDA

1. Acesso ao Banco de Dados Oracle
2. Exercícios

Introdução

- capítulo aborda como configurar o driver JDBC para Oracle, estabelecer conexões, criar declarações SQL, executar consultas e atualizações, e processar resultados.
- Vamos explorar cada etapa em detalhes e realizar exercícios práticos para reforçar o aprendizado.

Configurando o Driver JDBC para Oracle

- Primeiro, você precisará do driver JDBC do Oracle
- O *driver* é geralmente chamado *ojdbc8.jar* para Java 8 e versões mais recentes.
- Ele deve estar incluído no *classpath* do seu projeto.

Exemplo: Adicione o driver JDBC ao classpath do seu projeto em sua IDE ou ferramenta de build (como Maven ou Gradle).

Carregar o Driver JDBC

- A partir de Java 6, o carregamento explícito do driver não é mais necessário, mas caso você queira garantir que o driver está carregado, use:

```
try {
    Class.forName("oracle.jdbc.driver.OracleDriver");
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

Estabelecer uma Conexão

- Para se conectar a um banco de dados Oracle, você precisa fornecer a URL do banco de dados, o nome de usuário e a senha.

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

public class OracleConnectionExample {
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:xe"; // URL para Oracle XE
        String user = "usuario"; // Nome de usuário do banco de dados Oracle
        String password = "senha"; // Senha do banco de dados Oracle

        try (Connection connection = DriverManager.getConnection(url, user, password)) {
            System.out.println("Conexão estabelecida com sucesso!");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Criar uma Declaração (Statement)

- Você pode criar um *Statement* ou *PreparedStatement* para executar comandos SQL.

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

public class OracleStatementExample {
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:xe";
        String user = "usuario";
        String password = "senha";

        try (Connection connection = DriverManager.getConnection(url, user, password);
             Statement statement = connection.createStatement()) {
            String sql = "CREATE TABLE funcionarios (id NUMBER GENERATED BY DEFAULT AS IDENTITY, nome VARCHAR2(50), salario NUMBER)";
            statement.executeUpdate(sql);
            System.out.println("Tabela criada com sucesso!");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Executar Consultas ou Atualizações

- Utilize *executeQuery* para *SELECT* e *executeUpdate* para *INSERT*, *UPDATE* e *DELETE*.

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

public class OracleInsertExample {
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:xe";
        String user = "usuario";
        String password = "senha";
        String insertSQL = "INSERT INTO funcionarios (nome, salario) VALUES (?, ?)";

        try (Connection connection = DriverManager.getConnection(url, user, password);
             PreparedStatement preparedStatement = connection.prepareStatement(insertSQL)) {
            preparedStatement.setString(1, "João Silva");
            preparedStatement.setBigDecimal(2, new java.math.BigDecimal("3000.00"));
            int rowsAffected = preparedStatement.executeUpdate();
            System.out.println("Inserção concluída com sucesso! Linhas afetadas: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Processar os Resultados

- Após executar uma consulta *SELECT*, você pode processar os resultados usando *ResultSet*.

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

public class OracleSelectExample {
    public static void main(String[] args) {
        String url = "jdbc:oracle:thin:@localhost:1521:xe";
        String user = "usuario";
        String password = "senha";
        String selectSQL = "SELECT * FROM funcionarios";

        try (Connection connection = DriverManager.getConnection(url, user, password);
             PreparedStatement preparedStatement = connection.prepareStatement(selectSQL);
             ResultSet resultSet = preparedStatement.executeQuery()) {
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String nome = resultSet.getString("nome");
                java.math.BigDecimal salario = resultSet.getBigDecimal("salario");
                System.out.println("ID: " + id + ", Nome: " + nome + ", Salário: " + salario);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Conexões e Recursos

- Sempre feche *Connection*, *Statement*, *PreparedStatement* e *ResultSet* para evitar vazamentos de recursos.
- O bloco *try-with-resources* simplifica isso.

Exercício

1. Crie um C.R.U.D. para alguma funcionalidade do projeto elaborado em sala de aula.


FIAP