UNIP
UNIVERSIDADE PAULISTA

UNICLUSTER

Data Access Objects

(Objeto de Acesso a Dados)

Carlos Arruda Baltazar

UNIP – Cidade Universitária

Objeto de Acesso a Dados

Objeto de acesso a dados (acrônimo do inglês Data Access Object - DAO), é um padrão para aplicações que utilizam persistência de dados, onde tem a separação das regras de negócio das regras de acesso a banco de dados, implementada com linguagens de programação orientadas a objetos (como por exemplo Java) e arquitetura MVC, onde todas as funcionalidades de bancos de dados, tais como obter conexões, mapear objetos para tipos de dados SQL ou executar comandos SQL, devem ser feitas por classes DAO.

Vantagens

A vantagem de usar objetos de acesso a dados é a separação simples e rigorosa entre duas partes importantes de uma aplicação que não devem e não podem conhecer quase que nada uma da outra, e que podem evoluir frequentemente e independentemente. Alterar a lógica de negócio podem esperar apenas a implementação de uma interface, enquanto que modificações na lógica de persistência não alteram a lógica de negócio, desde que a interface entre elas não seja modificada.

Vantagens

- Pode ser usada em uma vasta porcentagem de aplicações;
- Esconde todos os detalhes relativos a armazenamento de dados do resto da aplicação;
- Atua como um intermediário entre a aplicação e o banco de dados;
- Mitiga ou resolve problemas de comunicação entre a base de dados e a aplicação, evitando estados inconsistentes de dados.

Vantagens

No contexto específico da linguagem de programação Java, um objeto de acesso a dados como padrão de projeto de software pode ser implementado de várias maneiras. Pode variar desde uma simples interface que separa partes de acesso a dados da lógica de negócio de uma aplicação até frameworks e produtos comerciais específicos. Os paradigmas para programação usando DAO's demandam alguma proficiência. O uso de tecnologias como Java persistence technologies e JDO garantem a implementação do padrão de projeto até certo ponto. Tecnologias como Enterprise JavaBeans trazem para a aplicação servidores montados e que podem ser usados em aplicações que usem um servidor de aplicação JEE. Produtos comerciais como o TopLink estão disponíveis, baseados em mapeamento objeto-relacional (ORM). Produtos ORM populares em código aberto incluem Doctrine, Hibernate, iBATIS e Apache OpenJPA.

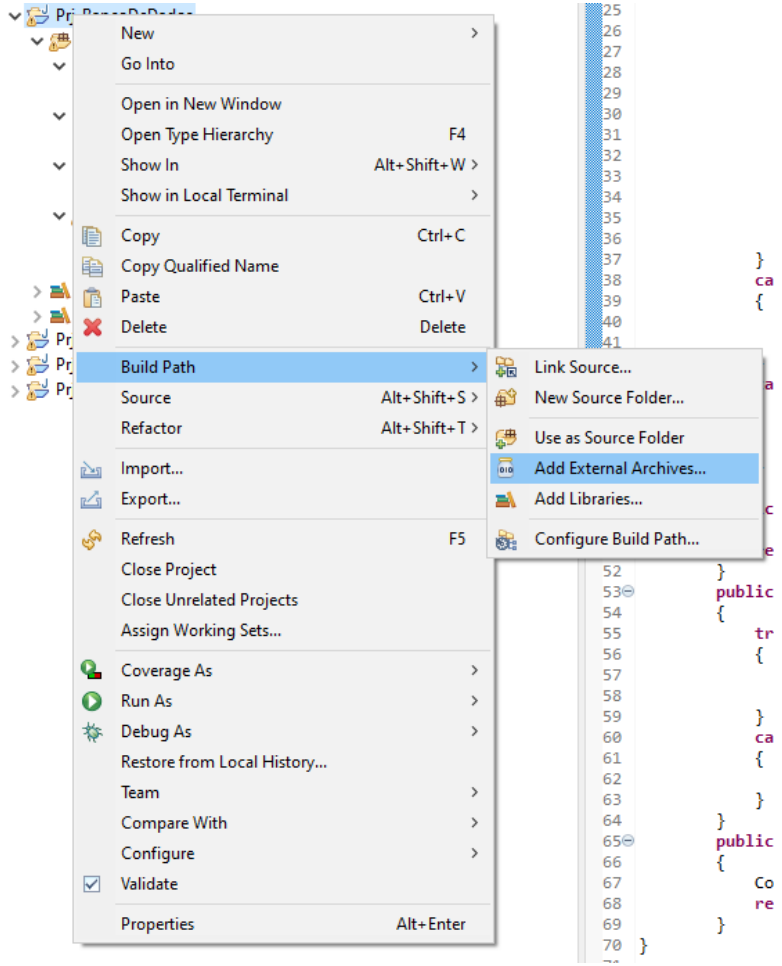
Download JDBC MySQL

https://downloads.mysql.com/archives/get/p/3/file/mysql-connector-java-8.0.25.zip

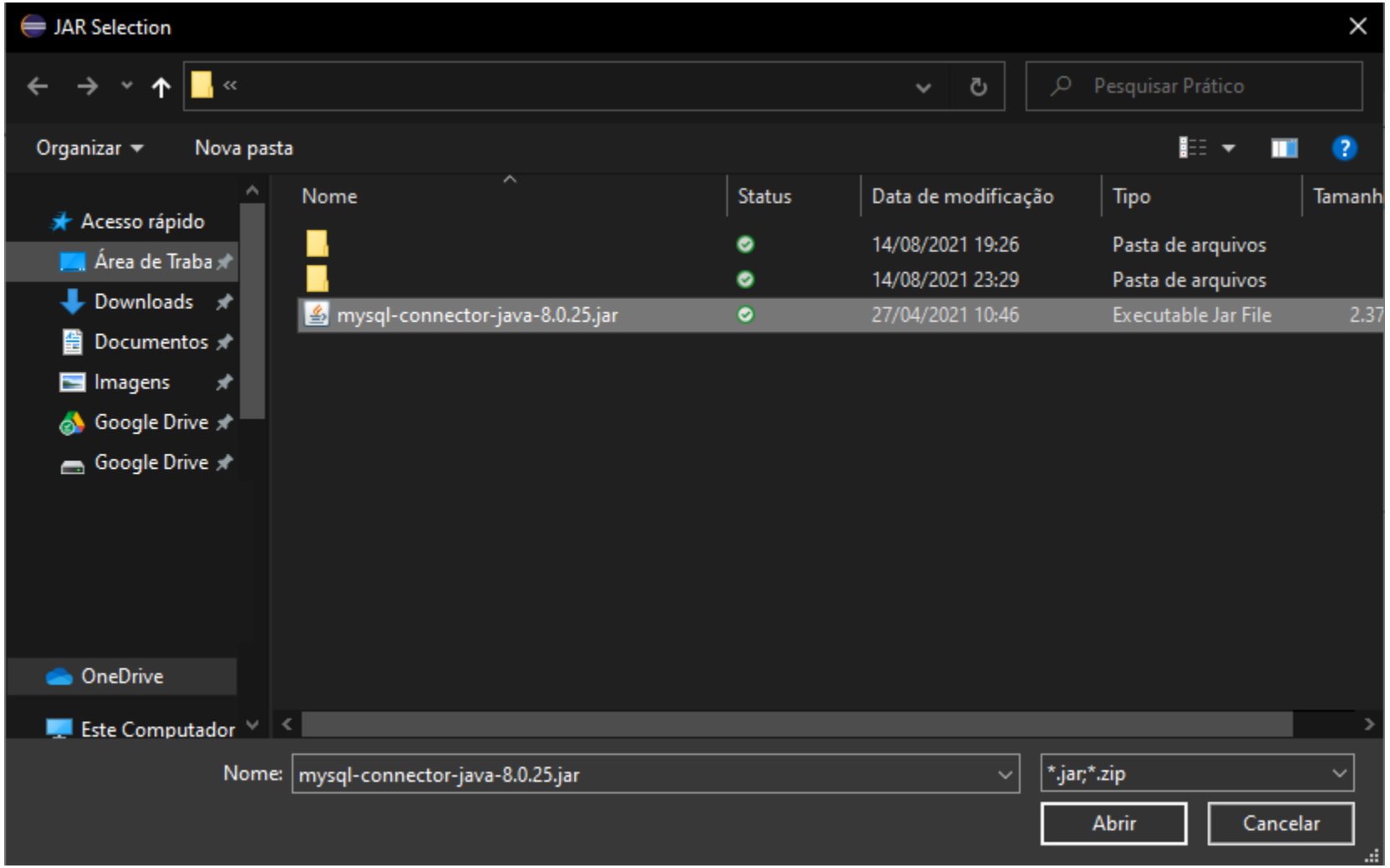
JDBC

Java Database Connectivity ou JDBC é um conjunto de classes e interfaces (API) escritas em Java que fazem o envio de instruções SQL para qualquer banco de dados relacional; Api de baixo nível e base para api's de alto nível; Amplia o que você pode fazer com Java; Possibilita o uso de bancos de dados já instalados; Para cada banco de dados há um driver JDBC que pode cair em quatro categorias.

Objeto de Acesso a Dados



Objeto de Acesso a Dados



Objeto de Acesso a Dados


```
package Pck_DAO;

import java.sql.Connection;

public class MySqlConnection
{
    private static String status = "Não conectou...";
    private static String serverName = "127.0.0.1";
    private static String mydatabase ="lpbd";
    private static String url = "jdbc:mysql://" + serverName + "/" + mydatabase;
    private static String username = "aluno";
    private static String password = "4@11";

    public MySqlConnection() {}

    public java.sql.Connection getMySqlConnection()
    {
            Connection connection = null;
            try
            {
                String driverName = "com.mysql.jdbc.Driver";
                Class.forName(driverName);
                connection = DriverManager.getConnection(url, username, password);
                if (connection != null)
                {
                    status = ("STATUS--->Conectado com sucesso!");
                    System.out.println(status);
                }
                else
                {
                    status = ("STATUS--->Não foi possivel realizar conexão");
                    System.out.println(status);
                }
                return connection;
            }
            catch (ClassNotFoundException e)
            {
                System.out.println("O driver expecificado nao foi encontrado.");
                return null;
            }
            catch (SQLException e)
            {
                System.out.println("Nao foi possivel conectar ao Banco de Dados.");
                return null;
            }
        }
        public String statusConection()
        {
            return status;
        }
        public boolean ColseConnection()
        {
            try
            {
                this.getMySqlConnection().close();
                return true;
            }
            catch (SQLException e)
            {
                return false;
            }
        }
        public java.sql.Connection RestartConnection()
        {
            ColseConnection();
            return this.getMySqlConnection();
        }
}
```

OBRIGADO