# Aplicações de Linguagem de Programação Orientada a Objeto

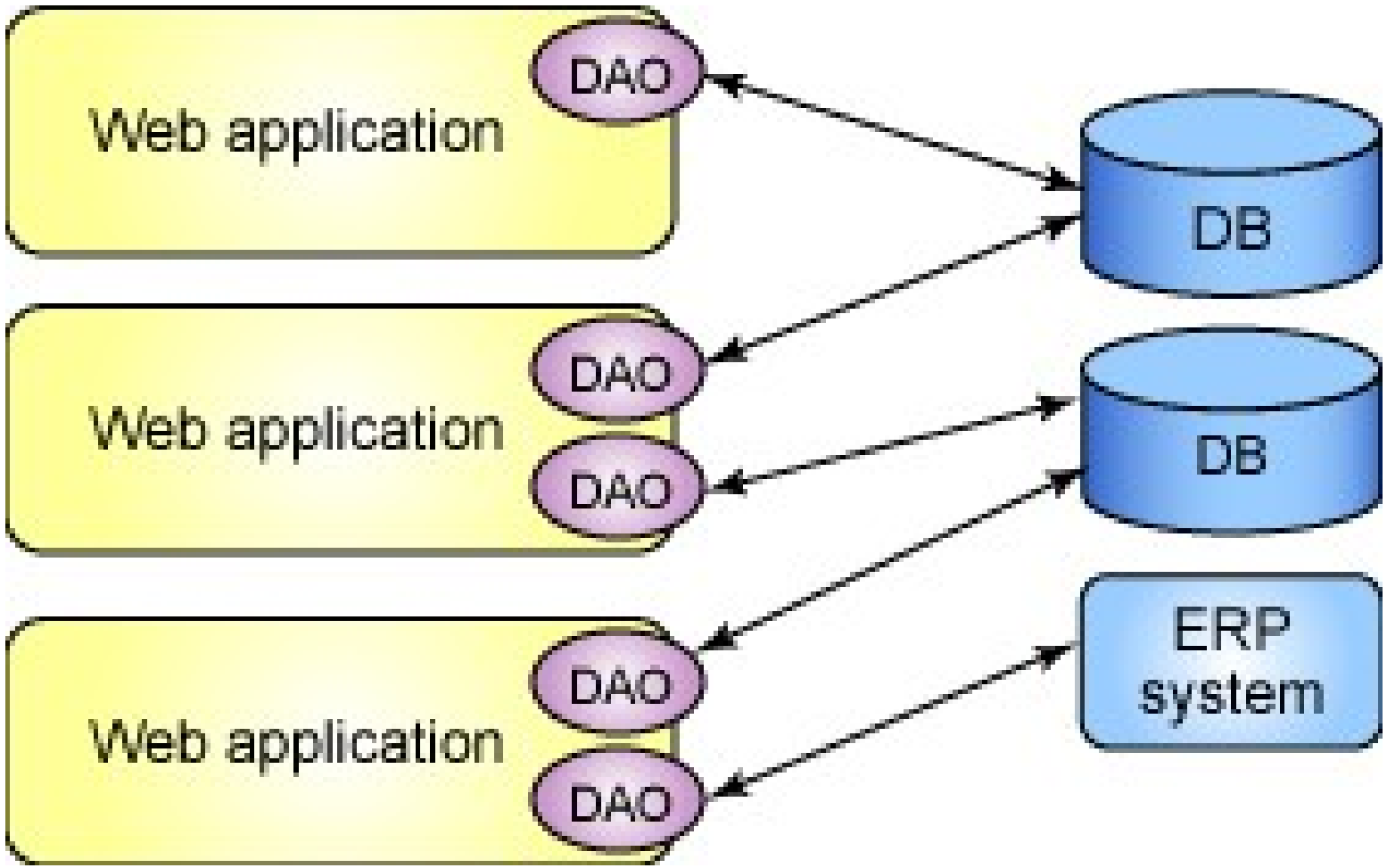
## DAO
## Data Access Object

*Professora Sheila Cáceres*

# Padrões de Projeto - DAO

- **Separa a lógica de acesso a dados da lógica do negócio.**
- A lógica de negócios fica isolada das diversas fontes de dados que podemos utilizar.
- Podemos adicionar fontes de dados ou fazer modificações nelas sem afetar a lógica do negócio.
- Assim, um DAO deve esconder todos os detalhes de implementação de acesso a fontes de dados.

# Padrões de Projeto - DAO




Fonte: Carlos Bazilio, Dpto de Ciencia e Tecnologia. Universidade Federal Fluminense.

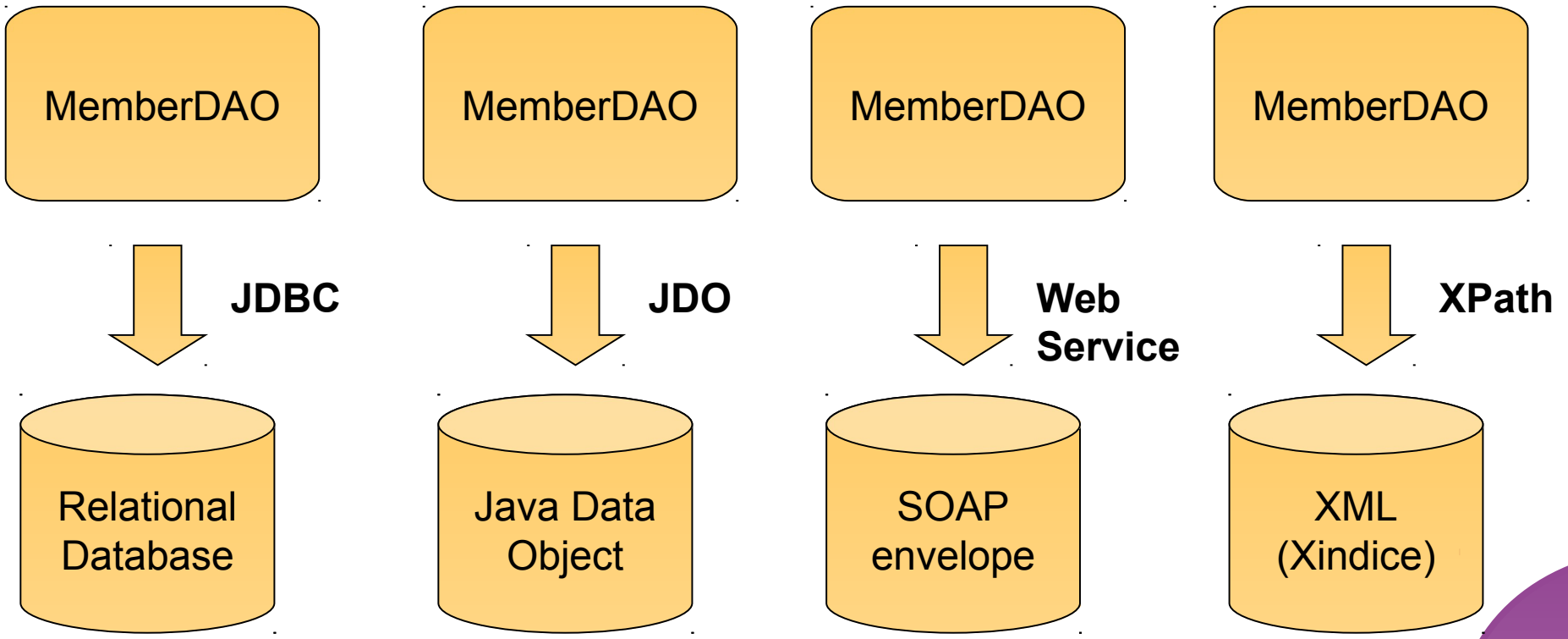
# DAO em Java

- Persistência refere-se ao armazenamento não-volátil de dados.  Existem diversas tecnologias de persistência.
- Em Java há algumas tecnologias de persistência como:
  - JDBC, JDO, EJB CMP, TopLink, Hibernate, iBATIS, openJPA e muitas outras
- O padrão DAO é bastante útil para separar a lógica de negócio das tecnologias de persistência
- Com este padrão permitimos que estas tecnologias possam ser substituídas ou atualizadas sem prejuízo ao restante da aplicação

# Abstração com DAO

- DAO abstrai a origem e o modo de obtenção / gravação dos dados.
- O restante do sistema manipula os dados de forma transparente, sem saber os detalhes da implementação.
- Isso ajuda muito em processos de migrações de fonte de dados.
- O DAO ajuda a evitar um Mal design: Códigos de programadores sendo acessados em vários pontos da aplicação repetitivamente: código redundante, difícil manutenção, difíceis possíveis migrações.

# DAOs


MemberDAO
MemberDAO
MemberDAO
MemberDAO
JDBC
JDO
Web Service
XPath
Relational Database
Java Data Object
SOAP envelope
XML (Xindice)

# Exemplo 1

```
public class User {
   private String name, password;
  // Construtores e sets / gets omitidos
}
```

UserDao não contem nenhuma informação específica da origem dos dados. Agora, podemos codificar os nosso Daos, implementando esta interface.

```
public interface UserDao {
   public void save (User user);
   public void delete (User user);
   public List list ();
   public User find (String name);
}
```

```
class JDBCUserDao implements UserDao {...}
```

```
class HibernateUserDao implements UserDao {...}
```

```
class TextFileUserDao implements UserDao {...}
```

# Exemplo 2

# Classe Pessoa

```
public class Pessoa {
    private int id;
    private String nome;
    private String sobrenome;
    private int idade;

    // ************ Constructores ******************
    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }

    public Pessoa(int id, String nome, String sobrenome, int idade) {
        this.id = id;
        this.nome = nome;
        this.sobrenome = sobrenome;
        this.idade = idade;
    }
    // ************ Gets e Sets ******************
    public String getNome() {    return nome;}
    public void setNome(String name) {   this.nome = name;    }
    public int getId() {     return id;   }
    public void setId(int id) {      this.id = id;    }
    public String getSobrenome() {       return sobrenome;    }
    public void setSobrenome(String sobrenome) {     this.sobrenome = sobrenome;}
    public int getIdade() {      return idade;    }
    public void setIdade(int idade) {        this.idade = idade;  }
}
```

# Interface IPessoaDao

```
import java.util.List;

public interface IPessoaDao {
    public List<Pessoa> getAllPessoas();
    public Pessoa findPessoa(int id);
    public void insertPessoa (Pessoa pessoa);
    public void updatePessoa(Pessoa pessoa);
    public void deletePessoa(Pessoa pessoa);
}
```

# Classe Pessoa

```
public class Pessoa {
    private int id;
    private String nome;
    private String sobrenome;
    private int idade;

    // ************ Constructores ******************
    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }

    public Pessoa(int id, String nome, String sobrenome, int idade) {
        this.id = id;
        this.nome = nome;
        this.sobrenome = sobrenome;
        this.idade = idade;
    }
    // ************ Gets e Sets ******************
    public String getNome() {    return nome;}
    public void setNome(String name) {   this.nome = name;    }
    public int getId() {     return id;   }
    public void setId(int id) {      this.id = id;    }
    public String getSobrenome() {       return sobrenome;    }
    public void setSobrenome(String sobrenome) {     this.sobrenome = sobrenome;}
    public int getIdade() {      return idade;    }
    public void setIdade(int idade) {        this.idade = idade;  }
}
```

# Classe PessoaDaoList

```
public class PessoaDaoList implements IPessoaDao {
   // *** Esta lista está substituindo um banco de dados ***
   List<Pessoa> pessoas;

   public PessoaDaoList(){
      pessoas = new ArrayList<Pessoa>();
   }
   @Override
   public void insertPessoa(Pessoa pessoa) {
       pessoas.add(pessoa);
  }
   @Override
   public void deletePessoa(Pessoa pessoa) {
      pessoas.remove(pessoa);
      System.out.println("Pessoa: Id " + pessoa.getId() +" apagada da lista (bd)");
   }
   // Recupera a lista de estudantes
   @Override
   public List<Pessoa> getAllPessoas() {   return pessoas;   }
   @Override
   public Pessoa findPessoa(int id) {  // TODO     }
   @Override
   public void updatePessoa(Pessoa newPessoa) {
       // A pessoa já foi atualizada ao usar as funções set.
      System.out.println("A Pessoa com Id: " + newPessoa.getId() +", foi atualizada
        na lista (bd)");
   }
}
```

# Classe PessoaDaoJDBC

```
public class PessoaDaoJDBC implements IPessoaDao{
     private String driver = "org.postgresql.Driver";
     private String sUsuario = "postgres";
     private String sSenha = "root";
     private String sFonte = "jdbc:postgresql://localhost:5432/postgres";
     private Connection con;
     private Statement st;

     public PessoaDaoJDBC(){
          this.criarTabela();
     }

     public void openConnection(){
          try {
               Class.forName(driver);
               con = DriverManager.getConnection(sFonte, sUsuario, sSenha);
          } catch (Exception e) { // demais exceções
               e.printStackTrace();
               System.out.println("Falha com o driver ou com a conexão com o banco!\n"
+e.getMessage());
               System.exit(0);
          }
     }

     /* Cria a Tabela pessoa no banco de dados postgres
      */
     public void criarTabela(){
          System.out.print("Criando a tabela pessoa: ");
          String sentencaSQL = "CREATE TABLE pessoa (id integer PRIMARY KEY, nome
VARCHAR(50), sobrenome VARCHAR(50), idade integer );";
          this.atualizar(sentencaSQL);
     }
```

# Classe PessoaDaoJDBC

```
    /* Realiza updates no banco de dados.*/
    public void atualizar(String sentencaSQL){
        try{
            this.openConnection();
            st = con.createStatement();// cria statement para consultar banco de dados
            st.executeUpdate(sentencaSQL);
            System.out.println("Update com sucesso!");
            st.close();   con.close();
        } catch (SQLException eSQL) {
            eSQL.printStackTrace();
            System.out.println("Não foi possível realizar o update!\n"+eSQL.getMessage());
            System.exit(2);
        }
    }
    /* Realiza uma consulta sobre a tabela pessoa e retorna uma lista contendo objetos
Pessoa */
    public List<Pessoa> consultarPessoas(String setencaSQL){
        List<Pessoa> pessoas = new ArrayList<Pessoa>();
        try {
            this.openConnection();
            st = con.createStatement();
            ResultSet rs = st.executeQuery(setencaSQL);
            while (rs.next()) {
                Pessoa p=new Pessoa(rs.getInt("id"), rs.getString("nome"),
rs.getString("sobrenome"), rs.getInt("idade"));
                pessoas.add(p);
            }
            rs.close();   st.close();   con.close();
        } catch (SQLException eSQL) {
            System.out.print("Erro na consulta SQL" + eSQL.getMessage() );
            System.exit(1);
        }
        return pessoas;
    }
```

# Classe PessoaDaoJDBC

```
    @Override
    public void insertPessoa (Pessoa pessoa){
        System.out.print("Inserindo uma pessoa: ");
        String sentencaSQL = "INSERT INTO pessoa values("+pessoa.getId()+","
                + "'"+pessoa.getNome() +"','"+pessoa.getSobrenome()
+"','"+pessoa.getIdade()+");";
        this.atualizar(sentencaSQL);
    }

    @Override
    public void updatePessoa(Pessoa pessoa) {
        // TODO
    }

    @Override
    public List<Pessoa> getAllPessoas() {
        return this.consultarPessoas("SELECT * FROM pessoa;");
    }

    @Override
    public Pessoa findPessoa(int id) {
        // TODO
        return null;
    }

    @Override
    public void deletePessoa(Pessoa pessoa) {
        // TODO
    }
}
```

# Rodando

```
public class DaoPatternDemo {
     public static void main(String[] args) {
          Pessoa pessoa0 = new Pessoa(0, "Joao", "Pereira", 25);
          Pessoa pessoa1 = new Pessoa(1, "Roberto", "Carvalho",30);
          Pessoa pessoa2 = new Pessoa(2, "Maria", "Ferraz",50);
          // Podemos usar qualquer implementacao de IPessoaDao
          IPessoaDao pessoaDao =new PessoaDaoList();// Ex: PessoaDaoList, PessoaDaoJDBC
          // Inserindo pessoas
          pessoaDao.insertPessoa(pessoa0);
          pessoaDao.insertPessoa(pessoa1);
          pessoaDao.insertPessoa(pessoa2);
          for (Pessoa pessoa : pessoaDao.getAllPessoas()) {
               System.out.println("Pessoa => [Id: " + pessoa.getId()+
                         ", Nome : " + pessoa.getNome() +
                         ", Sobrenome : " + pessoa.getSobrenome() +
                         ", Idade : " + pessoa.getIdade() + " ]");
          }
          // update pessoa
          Pessoa pessoa = pessoaDao.getAllPessoas().get(0);
          pessoa.setNome("Michael");
          pessoa.setSobrenome("da Silva");
          pessoaDao.updatePessoa(pessoa);

          System.out.println("Pessoa Atualizada => [Id: " + pessoa.getId()+
                    ", Nome : " + pessoa.getNome() +
                    ", Sobrenome : " + pessoa.getSobrenome() +
                    ", Idade : " + pessoa.getIdade() + " ]");
          pessoaDao.deletePessoa(pessoa0); //Apagar uma pessoa
          Pessoa p= pessoaDao.findPessoa(1); //Achar uma pessoa
          System.out.println("O nome da pessoa achada é: "+p.getNome());
     }
}
```

```
Pessoa => [Id: 0, Nome : Joao, Sobrenome : Pereira, Idade : 25 ]
Pessoa => [Id: 1, Nome : Roberto, Sobrenome : Carvalho, Idade : 30 ]
Pessoa => [Id: 2, Nome : Maria, Sobrenome : Ferraz, Idade : 50 ]
A Pessoa com Id: 0, foi atualizada na lista (bd)
Pessoa Atualizada => [Id: 0, Nome : Michael, Sobrenome : da Silva, Idade
: 25 ]
Pessoa: Id 0, apagada da lista (bd)
O nome da pessoa achada é: Roberto
```

# Bibliografia


- Javafree.org. Acessado o 12 de outubro do 2014.
- Dessign Patterns, Eric Gamma et al.
- Carlos Bazilio, Depto de Ciência e Tecnologia, Pólo Universitário de Rio das Ostras, Universidade Federal Fluminense