Mapeamento Relacional para Orientado a Objetos

1

Motivação

- Disponibilizar as informações do banco de dados para as aplicações;
- Separar as aplicações dos mecanismos de acesso as dados;
- Armazenar informações no banco de dados;
- Transformar as informações do modelo relacional para o orientado a objetos;

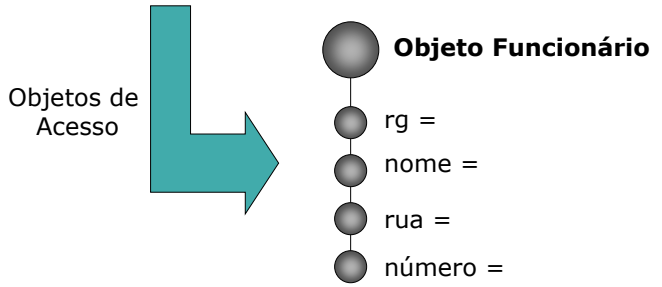
2

2

Motivação

Tabela Funcionário

rg	nome	rua	número
102356	João Silva	Rua das Palmeiras	458
104632	José Souza	Av. das Nações	523



3

3

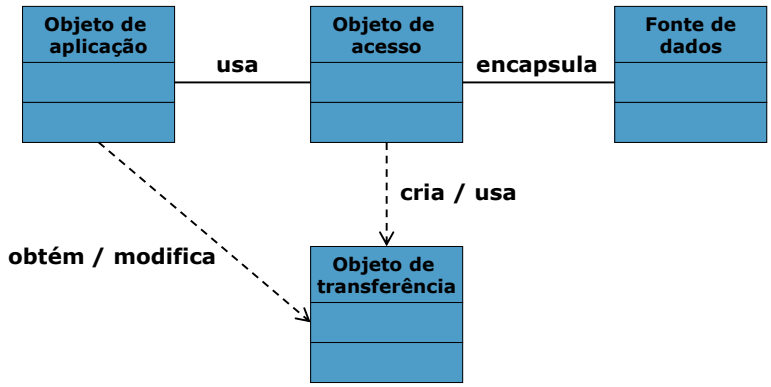
Método

- Serve como forma de padronizar a maneira com que as informações são acessadas / armazenadas no bando de dados;
- Segue um modelo único para acessar qualquer tipo de informação do banco de dados;
- O modelo utilizado pode ser: um modelo particular, ou um padrão de projeto (Design Pattern);
- O modelo aqui empregado será o padrão de projeto Java chamando *Data Access Object* (DAO).

4

4

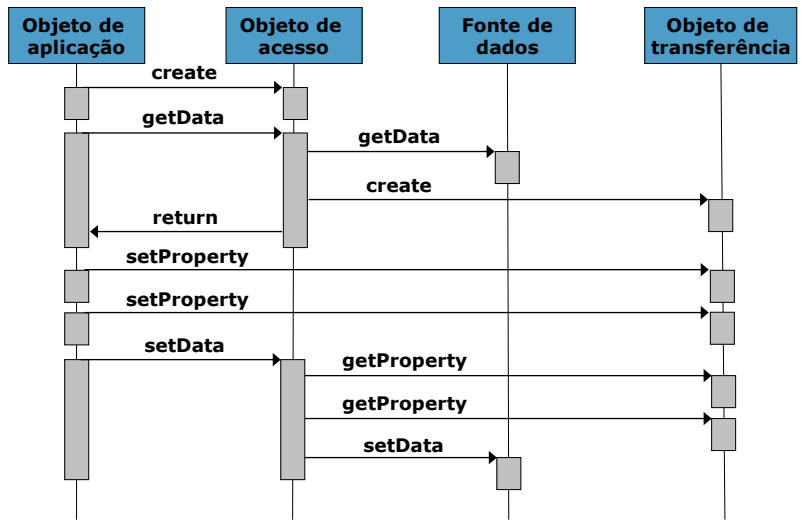
Diagrama de Classes



5

5

Diagrama de Sequência



6

6

Implementação

- 1º Passo: Implementar a classe que fará a conexão com o BD;
- 2º Passo: Implementar as classes dos objetos de transferência;
- 3º Passo: Implementar as classes dos objetos de acesso ao BD;
- 4º Passo: Implementar as classes dos objetos de aplicação que farão uso dos objetos de acesso e de transferência.

7

7

1º Passo: Classe de Conexão

```
public class DatabaseConnection {
   static private Connection connection = null;

   static public Connection getConnection()  {
      if (connection == null) {
         String url = "jdbc:postgresql://localhost:5432/mecanica";
         String username = "postgres";
         String password = "postgres";
         try   {
            Class.forName("org.postgresql.Driver");
            connection = DriverManager.getConnection(url, username, password);
         }
         catch (SQLException e)  {
            e.printStackTrace();
         }
         catch (ClassNotFoundException e1) {
            e1.printStackTrace();
         }
      }
      return connection;
   }
}
```

8

2º Passo: Classes de Transferência

```
public class Empresa {

   private int cnpj;
   private String nome;
   private String rua;
   private String bairro;
   private String cidade;

   public int getCnpj() {
      return cnpj;
   }
   public void setCnpj(int cnpj) {
      this.cnpj = cnpj;
   }
   public String getNome() {
      return nome;
   }
   public void setNome(String nome) {
      this.nome = nome;
   }
   ...
}
```

9

9

3º Passo: Classes DAO 1/2

```
public class EmpresaDAO {
   private static EmpresaDAO instance = null;
   private PreparedStatement selectEmpresa;
   private PreparedStatement insertEmpresa;

   public static EmpresaDAO getInstance() {
      if (instance == null)
         instance = new EmpresaDAO();
      return instance;
   }

   private EmpresaDAO() {
      Connection conn = DatabaseConnection.getConnection();
      try {
         selectEmpresa = conn.prepareStatement("select * from Empresa where cnpj = ? ");
         insertEmpresa = conn.prepareStatement("insert into Empresa "
         + "(cnpj, nome, rua, bairro, cidade, cep, estado) values (?, ?, ?, ?, ?, ?, ?)");
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
   …
```

10

10

3º Passo: Classes DAO 2/2

```
…
public Empresa load(int code) {
   ResultSet rs;
   Empresa emp = null;
   try {
      selectEmpresa.setInt(1, code);
      rs = selectEmpresa.executeQuery();
      if (rs.next()) {
         emp = new Empresa();
         emp.setCnpj(rs.getInt("cnpj"));
         emp.setNome(rs.getString("nome"));
         emp.setRua(rs.getString("rua"));
         emp.setBairro(rs.getString("bairro"));
         emp.setCidade(rs.getString("cidade"));
         emp.setCep(rs.getInt("cep"));
      }
   } catch (SQLException e) {
      e.printStackTrace();
   }
   return emp;
}
}
```

11

11