Aula 3

Desenvolvimento Web Back End

Prof. Rafael Veiga de Moraes

1

Conversa Inicial

2

Aplicação *web*

- **Desenvolvimento da aplicação**
  - ***Back-end***
  - ***Front-end***

3

- **Sistema de controle de estoque**
  - **Cadastro de cliente, fornecedor e produto**
  - **Cadastro de nota de entrada e de saída**
  - **Controle de estoque**

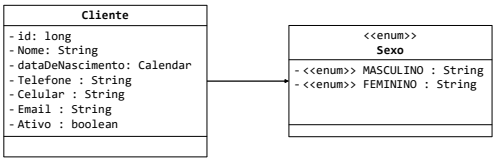
4

Criando a *model*

5

Cadastro de cliente

- **Diagrama UML**



6

Adicionando as dependências de acesso a dados

7

Dependências

- Bibliotecas externas utilizadas pela aplicação
- Arquivo de dependências do Spring
  - pom.xml

8

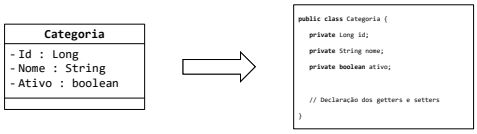
- Dependências de acesso a dados
  - MySQL Connector/J
    - *Driver* de conexão do MySQL
  - Spring Boot Starter Data JPA
    - Mapeamento objeto-relacional
    - Hibernate

9

- Hibernate
  - *Framework* ORM (*object relational mapping*)
  - Abstração da camada JDBC
  - Abstração dos comandos SQL

10

- Classe categoria



11

- Classe de acesso a dados (JDBC X JPA)

```
public class CategoriaDAO {
   private Connection connection;

   public CategoriaDAO() {
      this.connection = new ConnectionFactory().getConnection();
   }
   public void insere(Categoria categoria) throws SQLException {
      String sql = "insert into categoria(nome, ativo) values (?, ?)";
      PreparedStatement stmt = this.connection.prepareStatement(sql);
      stmt.setString(1, categoria.getNome());
      stmt.setBoolean(2, categoria.isAtivo);
      stmt.execute();
      stmt.close();
   }
}
```

```
@Transactional
public class CategoriaDAO {
    @PersistenceContext
   private EntityManager entityManager;

   public void insere(Categoria categoria) {
      entityManager.persist(categoria);
   }
}
```

12

Mapeamento objeto-relacional

13

Mapeamento objeto-relacional

- Utilizado para conversão de dados entre banco de dados relacionais e linguagens orientadas a objetos

14

- Anotações
  - Automatização de tarefas por meio de metadados
  - Identificadas no código Java pelo prefixo da arroba (@)

15

- Anotações JPA
  - @Entity
  - @Table
  - @Column
  - @Transient
  - Entre outras

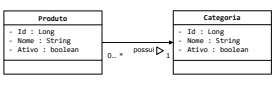
16

- Mapeamento objeto-relacional
  - Classe

```
@Entity
@Table(name="categorias")
public class Categoria {
   @Id
   @Column(name="categoria_id")
   @GeneratedValue(strategy=GenerationType.AUTO)
   private Long id;
   private String nome;
   private boolean ativo;
   // Declaração dos getters e setters
}
```

17

- Mapeamento objeto-relacional
  - Relacionamento entre classes




```
@Entity
@Table(name="produtos")
public class Produto {
      @Id
      @Column(name="produto_id")
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      private Long id;
      private String nome;
      @ManyToOne
      @JoinTable(name="categoria_id")
      private Categoria categoria;
      private boolean ativo;
      // Declarar getters e setters
}
```

18

Criando a classe de acesso a dados

19

Desenvolvimento

- Classe Cliente
  - Realizar o mapeamento objeto-relacional
- Classe ClienteDAO
  - Implementar os métodos de acesso a dados

20

Configurando o ambiente de acesso a dados

21

Configuração do projeto

- Arquivo de configuração do Spring
  - application.properties

22

- Propriedades de acesso a dados

```
# Configuração do SGBD
spring.datasource.url=jdbc:mysql://[servidor]:[porta]/[base_dados]
spring.datasource.username=[usuario]
spring.datasource.password=[senha]

# Configuração do Hibernate
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.ddl-auto=[hibernate_schema]
```

23