Domain Driven Design

PROF. ELIANE RODRIGUES MARION SANTA ROSA
Profeliane.rosa@fiap.com.br

1

CLASSE DAO – métodos alterar e excluir

A CLASSE EnderecoDao

Agora vamos fazer os métodos alterar e excluir, ambos possuem o mesmo conceito do método inserir.

Método alterar - EnderecoDao

```java
public void alterar(Endereco endereco)  {
    conexao = GerenciadorBD.obterConexao();
    PreparedStatement comandoSql = null;
    try{
        comandoSql = conexao.prepareStatement( sql: "update endereco set cep = ?, rua = ?, complemento = ?, " +
                "bairro = ?, cidade = ?, uf = ? where idEndereco = ?");
        comandoSql.setString( parameterIndex: 1, endereco.getCep());
        comandoSql.setString( parameterIndex: 2, endereco.getLogradouro()); //rua
        comandoSql.setString( parameterIndex: 3, endereco.getComplemento());
        comandoSql.setString( parameterIndex: 4, endereco.getBairro());
        comandoSql.setString( parameterIndex: 5, endereco.getLocalidade()); //cidade
        comandoSql.setString( parameterIndex: 6, endereco.getUf());
        comandoSql.setInt( parameterIndex: 7,endereco.getId());
        comandoSql.executeUpdate();
        conexao.close();
        comandoSql.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Entendendo o código

Criamos um método chamado alterar que recebe um endereco (objeto endereco) e não possui retorno.

```
public void alterar(Endereco endereco) {

}
```

Obtemos a conexão através do método obterConexao que escrevemos na classe GerenciadorBD.

```
conexao = GerenciadorBD.obterConexao();
```

Entendendo o código

Nesta linha estamos criando um objeto chamado comandoSQL do tipo PreparedStatement. Dentro do try tentaremos alterar um registro no banco de dados.

```
PreparedStatement comandoSQL = null;
```

E a grande alteração que teremos é a sintaxe SQL...

Entendendo o código

```
comandoSQL =email_endereco  conexao.prepareStatement("update endereco set cep =
?, rua = ?, complemento = ?, bairro=?, uf=? where idEndereco = ?");
```

- **Nesta linha fazemos o uso da conexão com o banco de dados para preparar um statement com a instrução SQL update. Devemos escrever a instrução da mesma forma que escrevemos no banco de dados e no lugar dos valores colocaremos interrogação(?).**

Entendendo o código

- Para cada interrogação utilizada devemos atribuir valores aos parâmetros indicados no preparedStatement. Devemos seguir a sequência utilizada na linha anterior.

```
comandoSQL.setString(1, endereco.getCep());
comandoSQL.setString(2, endereco.getLogradouro());
comandoSQL.setString(3, endereco.getComplemento());
comandoSQL.setString(4, endereco.getBairro());
comandoSQL.setString(5, endereco.getLocalidade());
comandoSQL.setString(6, endereco.getUf());
comandoSQL.setInt(7, endereco.getId());
```

- A única alteração foi a ordem, pois o id é o último valor já que entra na cláusula Where.

Entendendo o código

- Após montarmos nosso objeto comandoSQL o próximo passo é de fato executar a instrução update no banco, isso é feito com o método executeUpdate().

```
comandoSQL.executeUpdate();
```

- Por fim fechamos nossa conexão

```
conexao.close();
comandoSQL.close();
```

2

Testando a alteração de dados no banco

Testando sua conexão

FIAP

Podemos criar outra classe ou utilizarmos a classe TestaInsercao que já criamos.

```
public class TestaInsercao {

    public static void main(String[] args) {
      Scanner ent = new Scanner(System.in);
      Scanner ent = new Scanner(System.in);
         Endereco endereco = new Endereco();
         EnderecoDao dao = new EnderecoDao();
         System.out.println("Digite o código do endereco: ");
         endereco.setId(ent.nextInt());
         System.out.println("Digite o cep do endereco: ");
         endereco.setCep(ent.nextLine());
         System.out.println("Digite o nome da rua: ");
         endereco.setLogradouro(ent.nextLine());
```

Testando sua conexão

Continuação:

```
System.out.println("Digite o complemento do endereco: ");
endereco.setComplemento(ent.nextLine());
System.out.println("Digite o bairro do endereco: ");
endereco.setBairro(ent.nextLine());
System.out.println("Digite a cidade:");
endereco.setLocalidade(ent.nextLine());
System.out.println("Digite o estado:");
endereco.setUf(ent.nextLine());
/*Esta próxima linha chama o método alterar que criamos na classe
EnderecoDao, lembre-se que o id deve existir na tabela.*/
      dao.alterar(endereco);
      System.out.println("Endereco alterado com sucesso!");
    }
```

3

Excluindo um endereco

Método excluir - EnderecoDao

```
public void excluir(int id)  {
    conexao = GerenciadorBD.obterConexao();
    PreparedStatement comandoSql = null;
    try{
        comandoSql = conexao.prepareStatement( sql: "delete from endereco where idEndereco = ?");
        comandoSql.setInt( parameterIndex: 1,id);
        comandoSql.executeUpdate();
        conexao.close();
        comandoSql.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Entendendo o código

FIAP

Criamos um método chamado excluir que recebe um endereco (objeto endereco) e não possui retorno, poderíamos alterar a assinatura deste método recebendo um inteiro: public void excluir (int id).

```
public void excluir(int id) {

}
```

Obtemos a conexão através do método obterConexao que escrevemos na classe GerenciadorBD.

```
conexao = GerenciadorBD.obterConexao();
```

Entendendo o código

Nesta linha estamos criando um objeto chamado comandoSQL do tipo PreparedStatement. Dentro do try tentaremos excluir um registro no banco de dados.

```
PreparedStatement comandoSQL = null;
```

E a grande alteração que teremos é a sintaxe SQL...

Entendendo o código

```
comandoSQL = conexao.prepareStatement("delete from endereco where idEndereco = ?");
```

- **Nesta linha fazemos o uso da conexão com o banco de dados para preparar um statement com a instrução SQL delete. Devemos escrever a instrução da mesma forma que escrevemos no banco de dados e no lugar do valor colocaremos interrogação(?).**

Entendendo o código

- Neste caso só temos o id, portanto teremos uma única linha atribuindo valor ao parâmetro indicado no PreparedStatement.

```
comandoSQL.setInt(1, id);
```

- Não esqueça que o id deve existir no banco e devemos colocar a cláusula Where, caso contrário excluímos todos registros da tabela.

Entendendo o código

- Após montarmos nosso objeto comandoSQL o próximo passo é de fato executar a instrução delete no banco, isso é feito com o método executeUpdate().

```
comandoSQL.executeUpdate();
```

- Por fim fechamos nossa conexão

```
conexao.close();
comandoSQL.close();
```

3

Testando a exclusão de um endereco

Testando sua conexão

Podemos criar outra classe ou utilizarmos a classe TestaInsercao que já criamos.

```
public class TestaInsercao {

    public static void main(String[] args) {
      Scanner ent = new Scanner(System.in);
      Endereco endereco = new Endereco();
      EnderecoDao dao = new EnderecoDao();
      System.out.println("Digite o id do endereco: ");
      int id = ent.nextInt();
      dao.remover(id);
      System.out.println("Endereco excluido!");
    }
}
```

Entendendo o código

- Mas esta forma não é a mais adequada, já que o usuário está digitando um número (id) e estamos chamando o método excluir passando esse número e "mandando" excluir no banco, porém não temos certeza se este código existe no banco de dados.

- Então o ideal seria passar um endereço para o método excluir, mas antes fazer uma pesquisa no banco para verificar se existe esse endereço no banco.

Método excluir - EnderecoDao

```
public void excluir(Endereco endereco)  {
    conexao = GerenciadorBD.obterConexao();
    PreparedStatement comandoSql = null;
    try{
        comandoSql = conexao.prepareStatement( sql: "delete from endereco where idEndereco = ?");
        comandoSql.setInt( parameterIndex: 1,endereco.getId());
        comandoSql.executeUpdate();
        conexao.close();
        comandoSql.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Entendendo o código

Criamos um método chamado excluir que recebe um endereco (objeto endereco) e não possui retorno, poderíamos alterar a assinatura deste método recebendo um inteiro: public void excluir (int id).

```
public void excluir(Endereco endereco) {

}
```

Obtemos a conexão através do método obterConexao que escrevemos na classe GerenciadorBD.

```
conexao = GerenciadorBD.obterConexao();
```

Entendendo o código

Nesta linha estamos criando um objeto chamado comandoSQL do tipo PreparedStatement. Dentro do try tentaremos excluir um registro no banco de dados.

```
PreparedStatement comandoSQL = null;
```

E a grande alteração que teremos é a sintaxe SQL...

Entendendo o código

```
comandoSQL = conexao.prepareStatement("delete from endereco where idEndereco = ?");
```

- **Nesta linha fazemos o uso da conexão com o banco de dados para preparar um statement com a instrução SQL delete. Devemos escrever a instrução da mesma forma que escrevemos no banco de dados e no lugar do valor colocaremos interrogação(?).**

Entendendo o código

- Neste caso só temos o id, portanto teremos uma única linha atribuindo valor ao parâmetro indicado no PreparedStatement.

```
comandoSQL.setInt(1, endereco.getId());
```

- Não esqueça que o id deve existir no banco e devemos colocar a cláusula Where, caso contrário excluímos todos registros da tabela.

Entendendo o código

- Após montarmos nosso objeto comandoSQL o próximo passo é de fato executar a instrução delete no banco, isso é feito com o método executeUpdate().

```
comandoSQL.executeUpdate();
```

- Por fim fechamos nossa conexão

```
conexao.close();
comandoSQL.close();
```

3

Testando a exclusão de um endereco

Testando sua conexão

Podemos criar outra classe ou utilizarmos a classe TestaInsercao que já criamos.

```
public class TestaInsercao {

    public static void main(String[] args) {
      Scanner ent = new Scanner(System.in);
      Endereco endereco = new Endereco();
      EnderecoDao dao = new EnderecoDao();
      System.out.println("Digite o id do endereco: ");
      int id = ent.nextInt();
      endereco = dao.buscarPorId(id);
      dao.remover(endereco);
      System.out.println("Endereco excluido!");
    }
}
```

Testando sua conexão

Agora temos um problema, porque para excluirmos um registro no banco precisamos antes consulta-lo, por isso precisamos construir um método de busca, que deverá fazer um select no banco e guardar os dados no objeto endereço.

```
endereco = dao.buscarPorId(id);
```