| | |
|---|---|
| sãojudas universidade | FTCE<br>Faculdade de Tecnologia e Ciências Exatas |

USJT – 2022/1 – Programação de Soluções Computacionais

Professor Bossini

**Aula: 09 – Programação Orientada a Objetos – Interface Gráfica (GUI)**

# 1 Introdução a Interfaces gráficas

Neste material daremos continuidade ao desenvolvimento da aplicação da última aula. Começamos implementando a funcionalidade de login e ilustramos o uso de outros componentes visuais do pacote javax.swing. Se necessário, pegue uma cópia do projeto no site da disciplina.

# 2 Desenvolvimento

**2.1 (Cadastrando novos cursos)** A classe CursosTela possui botões para a manipulação de cursos na base. Começaremos implementando a funcionalidade de inserção de cursos.

- O primeiro passo é criar o método de inserção de cursos, o que deve ser feito na classe DAO. Veja a Listagem 2.1.1.

Listagem 2.1.1

```
public void inserirCurso (Curso curso) throws Exception{
	String sql = "INSERT INTO tb_curso (nome, tipo) VALUES (?, ?);";
	try (Connection conexao = ConexaoBD.obterConexao();
		PreparedStatement ps = conexao.prepareStatement(sql)){
	   ps.setString(1, curso.getNome());
	   ps.setString(2, curso.getTipo());
	   ps.execute();
	}
}
```

- Para cadastrar um curso, iremos utilizar somente seu nome e tipo, já que o id é gerado automaticamente pelo banco. Por isso, implemente o construtor da Listagem 2.1.2. Note que ele pertence à classe Curso.

Listagem 2.1.2

```
public Curso (String nome, String tipo){
      this.nome = nome;
      this.tipo = tipo;
}
```

- Criaremos um enum para representar o estado em que a tela se encontra e decidir o que fazer em função dele.

- Clique duas vezes no botão de inserção de cursos para editar o método que entra em execução quando ele é clicado pelo usuário. A inserção somente irá acontecer caso o usuário tenha preenchido os campos nome e tipo. Veja a sua implementação na Listagem 2.1.3.

Listagem 2.1.3

```
private void novoCursoButtonActionPerformed(java.awt.event.ActionEvent evt) {
      String nomeCurso = nomeCursoTextField.getText();
      String tipoCurso = tipoCursoTextField.getText();
      if (nomeCurso == null || nomeCurso.length() == 0 ||
            tipoCurso == null || tipoCurso.length() == 0){
         JOptionPane.showMessageDialog (null, "Preencha curso e tipo");
      }
      else{
         try{
            int escolha = JOptionPane.showConfirmDialog(null, "Confirmar cadastro
de novo curso?");
            if (escolha == JOptionPane.YES_OPTION){
               Curso curso = new Curso (nomeCurso, tipoCurso);
               DAO dao = new DAO();
               dao.inserirCurso(curso);
               JOptionPane.showMessageDialog(null, "Curso cadastrado com
sucesso");
               nomeCursoTextField.setText("");
               tipoCursoTextField.setText("");
               buscarCursos();
            }
         }
         catch (Exception e){
            JOptionPane.showMessageDialog(null, "Falha técnica, tente mais tarde");
            e.printStackTrace();
         }

      }
```

**2.2 (Atualizando cursos)** Para atualizar um curso, antes de mais nada, o usuário deve selecioná-lo no ComboBox. Quando esse evento acontecer, desejamos obter o curso selecionado e preencher os campos textuais com os valores que fazem parte de seu estado. O evento desejado pode ser encontrado clicando-se com o direito no ComboBox e então em **events >> action >> actionPerformed.** Veja a implementação do método na Listagem 2.2.1.

Listagem 2.2.1

```
private void cursosComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
      Curso curso = (Curso) cursosComboBox.getSelectedItem();
      idCursoTextField.setText(Integer.toString(curso.getId()));
      nomeCursoTextField.setText(curso.getNome());
      tipoCursoTextField.setText(curso.getTipo());
   }
```

- Cabe à classe DAO isolar o código JDBC que acessa a base. Por isso, crie o método da Listagem 2.2.2 nela.

Listagem 2.2.2

```
public void atualizarCurso (Curso curso) throws Exception{
      String sql = "UPDATE tb_curso SET nome = ?, tipo = ? WHERE id = ?";
      try (Connection conexao = ConexaoBD.obterConexao();
            PreparedStatement ps = conexao.prepareStatement(sql)){
         ps.setString (1, curso.getNome());
         ps.setString (2, curso.getTipo());
         ps.setInt (3, curso.getId());
         ps.execute();
      }
   }
```

- A seguir, podemos implementar o funcionamento do método que executa quando o usuário clica no botão Atualizar. Para isso, na classe CursosTela, clique duas vezes sobre ele. A implementação é dada na Listagem 2.2.3.

Listagem 2.2.3

```
private void atualizarCursoButtonActionPerformed(java.awt.event.ActionEvent evt)
{
      int escolha = JOptionPane.showConfirmDialog(null, "Atualizar curso?");
      if (escolha == JOptionPane.YES_OPTION){
         try{
            int id = Integer.parseInt (idCursoTextField.getText());
            String nome = nomeCursoTextField.getText();
            String tipo = tipoCursoTextField.getText();
            Curso curso = new Curso (id, nome, tipo);
            DAO dao = new DAO();
            dao.atualizarCurso(curso);
            JOptionPane.showMessageDialog(null, "Curso atualizado com sucesso");
            buscarCursos();
            idCursoTextField.setText("");
            nomeCursoTextField.setText("");
            tipoCursoTextField.setText("");
         }
         catch (Exception e){
            JOptionPane.showMessageDialog(null, "Falha técnica. Tente novamente
mais tarde.");
            e.printStackTrace();
         }
      }
   }
```

**2.3 (Removendo cursos)** O procedimento para remoção de cursos é análogo. Começamos escrevendo o método de acesso à base na classe DAO, como na Listagem 2.3.1.

Listagem 2.3.1

```
public void removerCurso (Curso curso) throws Exception{
      String sql = "DELETE FROM tb_curso WHERE id = ?";
      try (Connection conexao = ConexaoBD.obterConexao();
            PreparedStatement ps = conexao.prepareStatement(sql);){
         ps.setInt (1, curso.getId());
         ps.execute();
      }
   }
```

- A seguir, implementamos o método que executa quando o botão de remoção é clicado. Basta clicar duas vezes sobre o botão para encontrar o método. A implementação é dada na Listagem 2.3.2. Note que precisamos de um novo construtor na classe Curso, para receber somente o id, já que esse campo é suficiente para a operação de remoção.

Listagem 2.3.2

```
private void removerCursoButtonActionPerformed(java.awt.event.ActionEvent evt)
{
      int escolha = JOptionPane.showConfirmDialog(null, "Remover curso?");
      if (escolha == JOptionPane.YES_OPTION){
         try{
            int id = Integer.parseInt (idCursoTextField.getText());
            Curso curso = new Curso (id);
            DAO dao = new DAO();
            dao.removerCurso(curso);
            JOptionPane.showMessageDialog(null, "Curso removido com sucesso!");
            buscarCursos();
            nomeCursoTextField.setText("");
            tipoCursoTextField.setText("");
            idCursoTextField.setText("");
         }
         catch (Exception e){
            JOptionPane.showMessageDialog(null, "Falha técnica. Tente novamente
mais tarde.");
            e.printStackTrace();
         }

      }
   }

//na classe Curso
public Curso (int id){
      this.id = id;
}
```

**2.4 (O Botão cancelar)** O funcionamento do botão cancelar é simples. Quando clicado, a tela atual se fecha e a anterior (DashboardTela) é exibida. Veja a implementação na Listagem 2.4.1.

Listagem 2.4.1

```
private void cancelarCursoButtonActionPerformed(java.awt.event.ActionEvent evt)
{
      DashboardTela dt = new DashboardTela();
      dt.setVisible(true);
      this.dispose();
}
```

## *Exercícios*

1. Note que a aplicação tenta fazer a atualização e a remoção de cursos mesmo nos casos em que o usuário ainda não selecionou nenhum curso no ComboBox. Tente realizar uma operação dessas e veja o que acontece. Ajuste a aplicação para que, quando um desses dois botões forem clicados, ela informe ao usuário que ele precisa selecionar o curso envolvido na operação, caso ainda não o tenha feito.

## *Referências*

DEITEL, P. e DEITEL, H. **Java Como Programar**. 8ª Edição. São Paulo, SP: Pearson, 2010.