

PROGRAMAÇÃO PARA WEB I

AULA 16

Profa. Silvia Bertagnolli

JSF

TABELAS

USANDO TABELAS DE DADOS

```
<h:body>
  <h:form>
    <h:dataTable border="1" value="#{pessoaBean.lista}"
                  var="pessoaLista">
      <h:column>
        #{pessoaLista.nome}
      </h:column>
    </h:dataTable>
  </h:form>
```

Cria a variável **pessoaLista** que aponta para o atributo lista da classe

Mostra na tabela somente o nome que está sendo referenciado pela variável **pessoaLista**

Obs.: <h:column> adiciona as colunas na tabela

CLASSE PESSOABEAN

```
import java.util.LinkedList;
import javax.faces.bean.ManagedBean
@ManagedBean
public class PessoaBean {
    private Pessoa pessoa = new Pessoa();
    private List<Pessoa> lista ;
    public Pessoa getPessoa() { return pessoa; }
    public void setPessoa(Pessoa p) { this.pessoa = p; }
    public List<Pessoa> getLista() {
        if(lista == null)
            return pessoa.listar();
        return lista;
    }
}
```

O JSF chama o método getLista() "n" vezes, conforme o número de objetos dentro da lista. Para evitar isso verificamos se a lista está nula então carrega a lista, caso contrário só a retorna para o JSF

CABEÇALHO EM TABELAS

Podemos adicionar cabeçalhos em tabelas usando as TAGs do core

Usamos as facetas (facet) para modificar um componente

Usamos `xmlns:f=http://xmlns.jcp.org/jsf/core`

Note que agora estamos usando o namespace f e não mais h

CABEÇALHO EM TABELAS

Podemos adicionar cabeçalhos em tabelas usando

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<!-- resto do código -->
<h:dataTable border="1" value="#{pessoaBean.lista}"
              var="pessoaLista">
    <h:column>
        <f:facet name="header">Nome</f:facet>
        #{pessoaLista.nome}
    </h:column>
</h:dataTable>
```

EXERCÍCIOS 2 A 4

COMPONENTES

COMPONENTES

h:form – não é necessário indicar nenhuma action, mas é importante atribuir um valor para a propriedade id

INPUT

COMPONENTES INPUT

h:inputText

Entrada de dados restrita – campo de texto

Propriedades principais:

- size determina o número de caracteres permitido
- maxlength tamanho máximo visível
- required se o campo é obrigatório
- requiredMessage a mensagem que deve ser exibida se nenhum valor for informado
- readonly se o campo é somente leitura
- Principal propriedade: value, pois compreende o valor (texto) do componente

COMPONENTES INPUT

h:inputTextarea

Propriedades

- rows para definir o número de linhas
- cols para o número de colunas
- required se o campo é obrigatório
- requiredMessage a mensagem que deve ser exibida se nenhum valor for informado
- disabled se o campo não está habilitado para edição

Principal propriedade: value, pois compreende o valor (texto) do componente

COMPONENTES INPUT

h:inputSecret

Usado para entrada de dados do tipo senha

Propriedades:

- maxlength tamanho máximo visível
- required se o campo é obrigatório
- requiredMessage a mensagem que deve ser exibida se nenhum valor for informado

EXERCÍCIOS 5 E 6

SELECTONEMENU

COMPONENTES SELECT

Selecione o curso:

- Selezione --
- EC
- CIC
- SSI**
- TADS

Geralmente chamado de caixa de combinação ou combobox

h:selectOneMenu – é um input que traz para o usuário opções pré-definidas

h:selectItem – usado para mostrar mensagem estática ao usuário e garantir que um dado foi selecionado

COMPONENTES SELECT

h:selectItems – usado para buscar de um ManagedBean dados, a propriedade value é usada para este fim

itemValue – usado para obter o dado selecionado pelo usuário

itemLabel – usado para especificar o que vai aparecer para o usuário

COMPONENTES SELECT

Define onde será armazenado o valor selecionado

```
<h:selectOneMenu value="#{alunoBean.aluno.curso}">
    <f:selectItem itemLabel="-- Selecione --"
                  noSelectionOption="true" />
    <f:selectItems value="#{alunoBean.cursos}" var="curso"
                  itemValue="#{curso}" />
</h:selectOneMenu>
```

Define a origem dos dados que serão exibidos

EXERCÍCIO 7

SELECTONERADIO

COMPONENTE SELECTONERADIO

Selecione o curso: ☐ EC ☐ CIC ☒ SSI ☐ TADS

Geralmente chamado de radiobutton

h:selectOneRadio

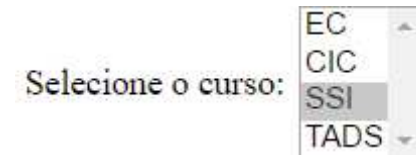
Quase igual ao h:selectOneMenu

Tem a propriedade `lineDirection`, para colocar os botões de rádio em linha; ou `pageDirection`, para colocar as opções uma abaixo da outra

EXERCÍCIO 8

SELECTONELISTBOX

COMPONENTES SELECTONELISTBOX



Geralmente chamado de caixa de listagem ou listbox

h:selectOneListbox

Carrega a lista aberta com os dados

Tem a propriedade `size` que é usada para determinar o número de linhas da lista que ficará visível sem a necessidade de rolagem

EXERCÍCIO 9

SELECTMANYCHECKBOX

COMPONENTE SELECTMANYCHECKBOX

Selecione o(s) curso(s): ☐ EC ☐ CIC ☒ SSI ☐ TADS

Geralmente chamado de caixa de verificação ou checkbox

h:selectManyCheckbox

Carrega a lista aberta com os dados

Tem a propriedade `lineDirection`, para colocar os botões de rádio em linha; ou

`pageDirection`, para colocar as opções uma abaixo da outra

SELECTBOOLEANCHECKBOX

SELECTBOOLEANCHECKBOX

Aluno Matriculado: ☒

Deve ser vinculado à atributos booleanos

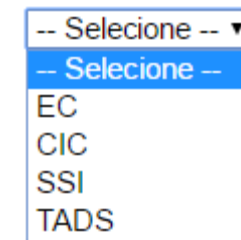
```
<h:selectBooleanCheckbox
```

```
value="#{alunoBean.aluno.matriculado}"/>
```

O checkbox assume o valor do atributo matriculado – true ou false

EXIBINDO OUTROS COMPONENTES

Aluno Matriculado: ☒



```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
```

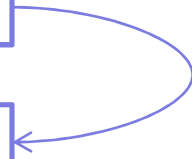
```
<h:outputLabel value="Aluno Matriculado:"/>
  <h:selectBooleanCheckbox value="#{alunoBean.aluno.matriculado}">
    <f:ajax render="matriculado" />
  </h:selectBooleanCheckbox>
```

.....

```
    <f:selectItem itemLabel="-- Selezione --"
                  noSelectionOption="true"/>
    <f:selectItems value="#{alunoBean.cursos}" var="curso"
```

EXIBINDO OUTROS COMPONENTES

```
<h:outputLabel value="Aluno Matriculado:"/>
  <h:selectBooleanCheckbox value="#{alunoBean.aluno.matriculado}">
    <f:ajax render="matriculado" />
  </h:selectBooleanCheckbox>
  <h:panelGroup id="matriculado">
    <h:panelGroup rendered="#{alunoBean.aluno.matriculado}">
      <h:selectOneMenu value="#{alunoBean.aluno.curso}">
        <f:selectItem itemLabel="-- Selecione --"
                      noSelectionOption="true"/>
        <f:selectItems value="#{alunoBean.cursos}" var="curso"
                      itemValue="#{curso}" />
      </h:selectOneMenu>
```

A blue curved arrow originates from the 'render="matriculado"' attribute in the checkbox tag and points to the 'id="matriculado"' attribute in the panel group tag, indicating that the panel group is rendered only when the checkbox is checked.

EXERCÍCIO 10

GRIDS

H: PANELGRID

Renderiza uma tabela cujo número de colunas é definido pela propriedade columns

Vantagem: é possível mostrar os componentes com 2, 4, 6 ou n colunas modificando apenas a propriedade columns do componente grid

H: PANELGROUP

É um container que serve apenas para agrupar outros componentes

Ele é usado também quando desejamos renderizar um componente ou vários a partir da ação realizada em outro componente

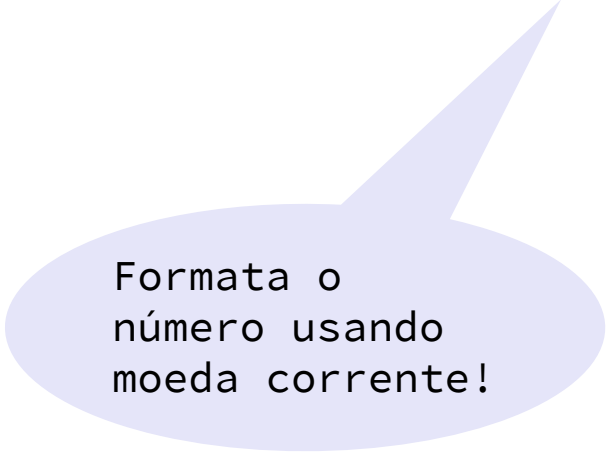
No exemplo a caixa de combinação de curso é exibida somente se o usuário marca a caixa de seleção – o componente é atualizado via AJAX através do seu id com valor “matriculado”

EXERCÍCIOS 11 E 12

FORMATAÇÕES

EXIBINDO VALORES CONVERTIDOS NO OUTPUTTEXT

```
<h:outputText value="#{alunoBean.aluno.mensalidade}">  
    <f:convertNumber type="currency"/>  
</h:outputText>
```



Formata o
número usando
moeda corrente!

EXERCÍCIO 13

RECURSOS

INCLUINDO RECURSOS

Passos:

1. Criar a pasta resources dentro do projeto

`PaginasWeb -> resources`

2. Criar a pasta para colocar as imagens

3. Criar a pasta para colocar os arquivos.css ou .js

USANDO IMAGENS

1. Usando imagens de um servidor

```
<h:graphicImage url="caminhoImagem"/>
```

2. Usando imagens incluídas no projeto

```
<h:graphicImage library="pasta" name="nomeImagem"/>
```

CSS

USANDO .CSS

Podemos usar as folhas de estilo de 4 formas:

- 1 - inline - regras dentro da TAG
- 2 - interna - regras definidas no cabeçalho do HTML
- 3 - externa - regras definidas em um documento separado
- 4 - usando a library do JSF

USANDO .CSS: INLINE

Tudo definido na linha do componente

```
<h:outputText value="mudando a cor da fonte e do background  
do texto" style="color>#0066ff; font-family:Arial;"/>
```

USANDO .CSS: INTERNA

```
<html>
  <h: head>
    <style type="text/css">
      form {color:#0066ff; font-family:Arial;}
    </style>
    ...
    <h:outputText value="mudando a cor da fonte e do background
do texto" />
```

USANDO .CSS: EXTERNA

```
<html>  
    <h:head>  
        <link href="resources/css/estilos.css"  
rel="stylesheet" type="text/css"/>  
    ...  
<h:outputText value="mudando a cor da fonte e do background  
do texto" class="botoes"/>
```


USANDO .CSS

Definir o estilo usando class ou id em um arquivo de estilos

Incorporar aos recursos do JSF como se fosse HTML:

```
<h:outputStylesheet library="css" name="estilos.css"/>
```

```
<h:commandButton value="Ok" class="botoes"/>
```

EXERCÍCIO 14

VALIDADORES

VALIDADORES NATIVOS

f:validateLongRange: Usado para validar números inteiros

f:validateDoubleRange: Usado para validar números reais

Propriedades:

minimum – indica o menor valor aceito

maximum – indica o maior valor que pode aceitar

VALIDADORES NATIVOS

`f:validateLenth:`

Usado para validar o tamanho de uma String (mínimo e máximo)

`f:validateRegex`

Permite criar uma regra para validar uma determinada informação

Exemplo letras de a-z ou A-Z a expressão regular seria:

`[A-Z][a-z]+([] [A-Z][a-z]+)*`