

PROGRAMAÇÃO PARA WEB II

AULA 4

Profa. Silvia Bertagnolli

COMPONENTES
PRIMEFACES

DIALOG

DIALOG

```
<p:dialog header="Meu diálogo" widgetVar="meuDialogo"  
    draggable="false" resizable="false" modal="false"  
    minimizable="true" maximizable="true"  
    showEffect="shake" hideEffect="explode">
```

Conteúdo do diálogo

```
</p:dialog>
```

Permite definir uma
caixa de diálogo para
mostrar informações

DIALOG

Para usar devemos chamar o diálogo usando PF, que ativa a caixa de diálogo com o nome meuDialogo

```
<p:button value="Exibir" onclick="PF('meuDialogo').show();  
return false;" />
```



1. Abra a página dialogo1.xhtml, analise o seu código e execute para verificar como funciona
2. Abra a página dialogo2.xhtml, analise o seu código e execute para verificar como funciona
3. Abra a página dialogo3.xhtml, analise o seu código e execute para verificar como funciona

DATATABLE

DATA TABLE

Usado para exibir os dados em uma estrutura tabular

Já possui paginação e ordenação de colunas automática

Permite customizar o número de registros por página

DATA TABLE

```
<p:dataTable value="#{produtoBean.produtos}" var="prod"
    rows="5" paginator="true"
    paginatorPosition="bottom"
    rowsPerPageTemplate="5, 10, 20"
    paginatorAlwaysVisible="true">
```

Definições básicas da
tabela

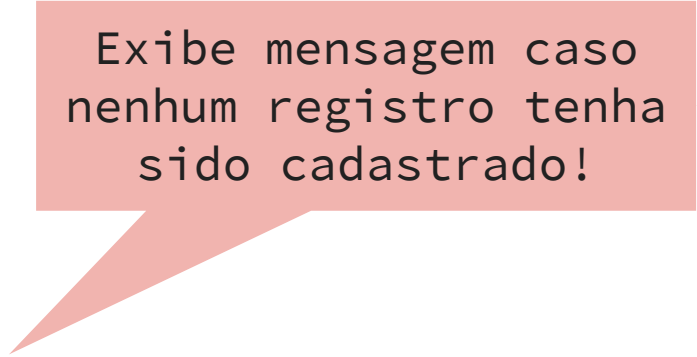
DATA TABLE

```
<p:dataTable value="#{produtoBean.produtos}" var="prod"
    rows="5" paginator="true"
    paginatorPosition="bottom"
    rowsPerPageTemplate="5, 10, 20"
    paginatorAlwaysVisible="true"
    sortMode="multiple">
```

Permite a seleção de
múltiplas colunas
para ordenar os dados

DATA TABLE

```
<p:dataTable value="#{produtoBean.produtos}" var="prod"
    rows="5" paginator="true"
    paginatorPosition="bottom"
    rowsPerPageTemplate="5, 10, 20"
    paginatorAlwaysVisible="true"
    emptyMessage="Nenhum registro encontrado" >
```



Exibe mensagem caso
nenhum registro tenha
sido cadastrado!

DATA TABLE

```
<p:dataTable ....>
```

```
  p:column headerText="Nome" style="text-align: center"
```

```
    width="90" sortBy="#{prod.nome}">
```

```
      <h:outputText value="#{prod.nome}"/>
```

```
    </p:column>
```

```
</p:dataTable>
```

Adiciona colunas e define qual atributo da classe será usado para ordenar

Mostra valor do texto dentro da coluna

OBSERVAÇÃO

Para que a ordenação ocorra corretamente voc deve:

- 1) Criar um método que inicializa os dados, por exemplo, `init()`
- 2) Anotar o método com `@PostConstruct`

POSTCONSTRUCT

É importante colocar as inicializações no método anotado com `@PostConstruct`

Esse método poderá executar alguma ação logo após a inicialização do objeto, porém antes do sistema executar alguma ação do usuário



EXERCÍCIOS

1. Abra a página Funcionario1.xhtml. Analise como ela funciona.
2. O que faz a linha de código "`<f:convertNumber type='currency' locale='pt_BR'/>`"? Explique com suas palavras.
3. Abra a página Funcionario2.xhtml. A página já abre ordenada? Como é o processo de ordenação? Qual componente é usado para ordenar?



EXERCÍCIOS

4. Abra a página Funcionario3.xhtml. Ela permite a paginação de registros e possibilita a ordenação por salário
5. Quais propriedades do DataTable permitem a ordenação? Explique essas propriedades
6. Modifique a página para que ela permita mostrar 15 registros por página. Depois modifique a página para que ela permita a ordenação por qualquer um dos campos da tabela

DATALIST

DATALIST (PÁGINA LISTAUSUARIOS.XHMTL)

```
<p:dataList value="#{usuarioBean.usuarios}" var="user"
  type="unordered" itemType="none" paginator="true" rows="10"
  styleClass="paginated">
  <f:facet name="header">Usuários Cadastrados</f:facet>

  <p:commandLink update=":abrirForm"
    oncomplete="PF('userDialog').show()" title="Detalhe Usuário"
    styleClass="ui-icon ui-icon-search"
    style="float:left;margin-right:10px">
    <f:setPropertyActionListener value="#{user}"
      target="#{usuarioBean.usuarioSelecioneado}" />
    <h:outputText value="#{user.nome}, #{user.identificador}" />
  </p:commandLink>
  <h:outputText value="#{user.nome}" style="display:inline-block"/>
</p:dataList>
```

DATALIST

Configura o dataList para
mostrar 10 linhas e
paginação

```
<p:dataList value="#{usuarioBean.usuarios}" var="user"
  type="unordered" itemType="none" paginator="true" rows="10"
  styleClass="paginated">
  <f:facet name="header">Usuários Cadastrados</f:facet>

  <p:commandLink update=":abrirForm"
    oncomplete="PF('userDialog').show()" title="Detalhe Usuário"
    styleClass="ui-icon ui-icon-search"
    style="float:left;margin-right:10px">
    <f:setPropertyActionListener value="#{user}"
      target="#{usuarioBean.usuarioSelecionado}" />
    <h:outputText value="#{user.nome}, #{user.identificador}" />
  </p:commandLink>
  <h:outputText value="#{user.nome}" style="display:inline-block"/>
</p:dataList>
```

DATALIST

Define que a caixa de diálogo
userDialog deve ser aberta
quando o item for clicado

```
<p:dataList value="#{usuarioBean.usuarioSelecionado" type="user"
  type="unordered" itemType="none" paginator="true" rows="10"
  styleClass="paginated">
  <f:facet name="header">Usuários Cadastrados</f:facet>

  <p:commandLink update=":abrirForm"
    oncomplete="PF('userDialog').show()" title="Detalhe Usuário"
    styleClass="ui-icon ui-icon-search"
    style="float:left;margin-right:10px">
    <f:setPropertyActionListener value="#{user}"
      target="#{usuarioBean.usuarioSelecionado}" />
    <h:outputText value="#{user.nome}, #{user.identificador}" />
  </p:commandLink>
  <h:outputText value="#{user.nome}" style="display:inline-block"/>
</p:dataList>
```

DATALIST

Monta a janela de diálogo

```
<p:dialog header="Informações do Usuário" widgetVar="userDialog" >
  <h:form id="abrirForm">
    <p:outputPanel id="userDetail" style="text-align:center;">
      <p:panelGrid columns="2" rendered="#{not empty
        usuarioBean.usuarioSelecionado}" columnClasses="label,value">

        <h:outputText value="Nome:" />
        <h:outputText value="#{usuarioBean.usuarioSelecionado.nome}" />
        <h:outputText value="Identificador:" />
        <h:outputText
          value="#{usuarioBean.usuarioSelecionado.identificador}" />
      </p:panelGrid>
    </p:outputPanel>
  </h:form>
</p:dialog>
```

Mostra o usuário selecionado na DataList em um diálogo – exibe somente nome e identificador, mas pode exibir todos os dados



EXERCÍCIOS

1. Refaça a página `listarProdutos.xhtml` usando o componente `DataList` – modifique o nome da página para `listarProdutosDataList.xhtml`
2. Refaça a página `listarProdutos.xhtml` usando o componente `repeat` – modifique o nome da página para `listarProdutosRepeat.xhtml`
3. Refaça a página `listarProdutos.xhtml` usando o componente `dataGrid` – modifique o nome da página para `listarProdutosDataGrid.xhtml`



EXERCÍCIOS

4. Abra a página Aluno1.xhtml. O que ela faz? Como é realizada a entrada com máscara?
5. Adicione um campo nessa página e crie uma máscara customizada para ele.
6. Abra a página Aluno2.xhtml. O que ela faz? Explique como funciona o componente `p:inputTextarea`?



EXERCÍCIOS

7. No método `cadastrar()` do bean o que faz a linha de código abaixo:

```
FacesContext.getCurrentInstance().addMessage(null, new  
FacesMessage("Cadastro realizado!"));
```

8. Abra a página `Aluno3.xhtml`. Explique por que foi usado o componente `selectBooleanCheckbox`. Qual outro componente você usaria para mapear a funcionalidade de um booleano?



EXERCÍCIOS

9. Abra a página Aluno4.xhtml. Explique por que na classe AlunoBean_4 os interesses foram mapeados como uma List?
10. Refaça a página AlunoBean_4.xhtml usando o componente p:selectCheckboxMenu
11. Abra a página Aluno5.xhtml. Utilize um campo de auto completar para as profissões



EXERCÍCIOS

12. Abra a página Aluno6.xhtml. Modifique o componente Sexo para utilizar o componente p:selectOneListbox.
13. Abra a página Aluno7.xhtml. Explique como funciona o componente p:calendar.
14. Quais configurações podem ser realizadas no componente p:calendar? Inclua um novo componente para registrar a data da matrícula, possibilite abrir o calendário com um efeito.