01

Qual menu a usar?

Transcrição

Começando daqui? Você pode fazer o DOWNLOAD (https://s3.amazonaws.com/caelum-online-public/jsf-primefaces/stages/capitulo-6.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Neste capítulo, iremos melhorar a navegação da nossa aplicação. Atualmente só conseguimos acessar as páginas digitando as suas urls no navegador, e o ideal seria que nós tivéssemos um menu customizado com as páginas da aplicação, podendo assim acessá-las mais facilmente.

Qual menu a usar?

Como sempre, iremos primeiramente olhar o *ShowCase* do Primefaces, lá tem uma sessão Menu, com várias opções, iremos utilizar o **MenuButton** (http://www.primefaces.org/showcase/ui/menu/menuButton.xhtml). Podemos copiar todo o exemplo e ir adaptando-o para a nossa aplicação. Como queremos que o menu apareça na aplicação toda, devemos colocá-lo no `_template.xhtml` , mas não podemos nos esquecer que, antes de mais nada, devemos importar o *namespace* do Primefaces, dentro da tag `<html>` :

```
xmlns:p="http://primefaces.org/ui"
```

E o exemplo, dentro da `div cabecalho` :

```
<div id="cabecalho">
    <h:graphicImage library="img" name="logo.png" />

    <h:form>
        <p:growl id="messages" showDetail="true"/>

        <p:menuButton value="Options">
            <p:menuitem value="Save" actionListener="#{menuView.save}" update="messages" icon="
            <p:menuitem value="Update" actionListener="#{menuView.update}" update="messages" ic
            <p:menuitem value="Delete" actionListener="#{menuView.delete}" ajax="false" icon="u
            <p:separator />
            <p:menuitem value="Homepage" url="http://www.primefaces.org" icon="ui-icon-extlink",
        </p:menuButton>
    </h:form>

    <h:form rendered="#{usuarioLogado != null}">
        <h:commandLink value="Logout" action="#{loginBean.deslogar}"/>
    </h:form>

    <h1>
        <ui:insert name="titulo"></ui:insert>
    </h1>
</div>
```

Ajustando os ícones

Vamos começar a adaptar o menu, só precisaremos de três itens: as páginas de login e autor e o logout. Vamos começar pelo logout, basta copiar os atributos do `commandLink` que já existe, removendo os antigos, só deixando o ícone (mas no caso utilizaremos o ícone `fa fa-fw fa-sign-out`):

```
<p:menuButton value="Options">
    <p:menuitem value="Save" actionListener="#{menuView.save}" update="messages" icon="ui-icon-
    <p:menuitem value="Update" actionListener="#{menuView.update}" update="messages" icon="ui-i
    <p:separator />
    <p:menuitem value="Logout" value="Logout" action="#{loginBean.deslogar}" icon="fa fa-fw fa-
</p:menuButton>
```

Os outros itens são atualmente o *Save* e o *Update*, que virarão os itens **Livros** e **Autores**, com o atributo `action` fazendo o `faces-redirect` para as páginas `livro` e `autor` , respectivamente, e utilizando o ícone `fa fa-fw fa-edit` . Vamos alterar também o nome do menu, ao invés de **Options**, será **Menu**:

```
<p:menuButton value="Menu">
    <p:menuitem value="Livros" action="livro?faces-redirect=true" icon="fa fa-fw fa-edit" />
    <p:menuitem value="Autores" action="autor?faces-redirect=true" icon="fa fa-fw fa-edit" />
    <p:separator />
    <p:menuitem value="Logout" action="#{loginBean.deslogar}" icon="fa fa-fw fa-sign-out" />
</p:menuButton>
```

Ótimo, mas o menu também aparece na página de login, para resolver isso vamos fazer a mesma solução utilizada no `commandLink` antigo de logout, utilizando o atributo `render` do formulário:

```
<h:form rendered="#{usuarioLogado != null}">
    <p:growl id="messages" showDetail="true"/>

    <p:menuButton value="Menu">
        <p:menuitem value="Livros" action="livro?faces-redirect=true" icon="fa fa-fw fa-edit" /
        <p:menuitem value="Autores" action="autor?faces-redirect=true" icon="fa fa-fw fa-edit" /
        <p:separator />
        <p:menuitem value="Logout" action="#{loginBean.deslogar}" icon="fa fa-fw fa-sign-out" /
    </p:menuButton>
</h:form>
```

Pequenos ajustes no logout

Com isso já podemos remover o `commandLink` antigo de logout, deixando o template mais limpo.

Para finalizar o menu, vamos mover o `p:growl` para fora do formulário. Esse componente irá mostrar as mensagens de uma forma diferente, como se aparecessem notificações na tela.

```
<div id="cabecalho">
    <h:graphicImage library="img" name="logo.png" />

    <p:growl id="messages" showDetail="true" />
```

```
    <h:form rendered="#{usuarioLogado != null}">
        <p:menuButton value="Menu">
            <p:menuitem value="Livros" action="livro?faces-redirect=true" icon="fa fa-fw fa-edit
            <p:menuitem value="Autores" action="autor?faces-redirect=true" icon="fa fa-fw fa-edi
            <p:separator />
            <p:menuitem value="Logout" action="#{loginBean.deslogar}" icon="fa fa-fw fa-sign-out
        </p:menuButton>
    </h:form>

    <h1>
        <ui:insert name="titulo"></ui:insert>
    </h1>
</div>
```

Mas se formos testar, parece que nada mudou, as mensagens continuam sendo exibidas como antes. Isso porque como estamos fazendo ajax, devemos definir qual componente devemos atualizar, então deveríamos alterar cada página dizendo que esse componente `growl` também deverá ser atualizado. Mas para evitar isso, o `growl` já tem um atributo para se auto-atualizar, o `autoUpdate`. Com isso não precisamos mexer nos componentes das outras páginas:

```
<p:growl id="messages" showDetail="true" autoUpdate="true" />
```

Reparem que agora as mensagens são exibidas na tela como antigamente e também na forma de notificações, tornando a página mais intuitiva.