PROGRAMAÇÃO WEB

Java EE com Java Server Faces

Componentes Richfaces: Ui - Input

- Os componentes de UI são disponibilizados pela taglib rich.
- Apesar de não apresentarem comportamento com AJAX nativo esses componentes podem ser utilizados em conjunto com os componentes a4j.
- Poucos componentes como <rich:separator> não possuem integração com ajax.

Componentes Richfaces: Ui - Input

<rich:inplaceInput>

- A opção showControl exibe um componente para confirmação ou cancelamento da ação.
- A posição dos controles pode ser alterada através do atributo controlsHorizontalPosition . Os valores de posição são: left, right e center.
- Também é possível determinar o alinhamento vertical com o atributo controlsVerticalPosition e as opções: bottom, center e top.

Componentes Richfaces: Ui - Input

<rich:inplaceInput>

- Extensão do componente h:input.
- Este componente torna um campo label editável.
- Depois que o usuário edita o campo ele volta a ser exibido com um label.

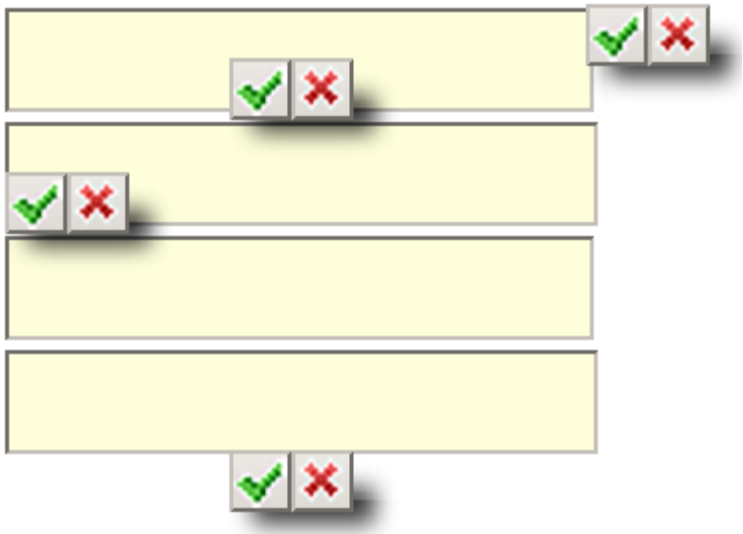
Componentes Richfaces: Ui - Input

<rich:inplaceInput>

```
<rich:inplaceInput value="#{inplaceInputBean.name}"
defaultLabel="Click to edit name" showControls="true" />
<rich:inplaceInput value="#{inplaceInputBean.email}"
showControls="true" defaultLabel="Click to edit email"
controlsHorizontalPosition="center" controlsVerticalPosition="top" />
<rich:inplaceInput value="#{inplaceInputBean.name}"
defaultLabel="Click to edit name" showControls="true"
controlsHorizontalPosition="left" controlsVerticalPosition="top" />
<rich:inplaceInput value="#{inplaceInputBean.email}"
controlsHorizontalPosition="center"
controlsVerticalPosition="bottom" defaultLabel="Click to edit email"
showControls="true" />
```

Componentes Richfaces: Ui - Input

<rich:inplaceInput>



Componentes Richfaces: Ui - Input

<rich:inplaceInput>

- O inputplaceInput ainda disponibiliza os eventos:
  - oneditactivation: Disparado na ativação da edição;
  - oneditactivated: Disparado quando a edição é ativada
  - onviewactivation: Disparado quando o estado de visualizar é ativado;
  - onviewactivated: Disparado depois que o estado de visualização é ativado.

Componentes Richfaces: Ui - Input

<rich:inplaceInput>

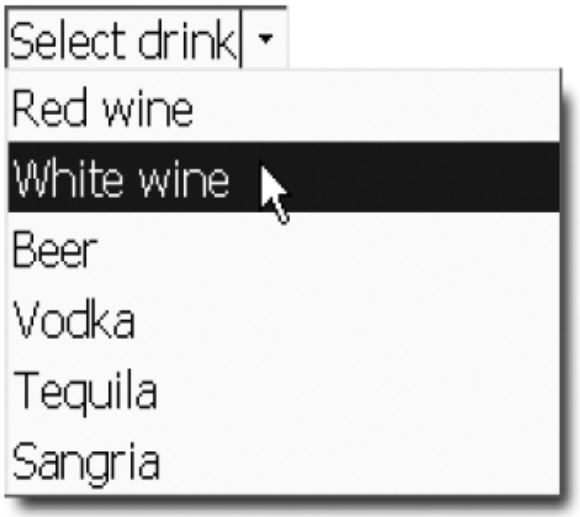
```
<rich:inplaceInput id="inputEmail" value="#{inplaceInputBean.email}"
defaultLabel="Click to edit email" showControls="true"
controlsVerticalPosition="bottom"
controlsHorizontalPosition="left"
oneditactivation="if (confirm('Editar label?')){return false;}">
<f:facet name="controls">
<button onclick="#{rich:component('inputEmail')}.save();"
type="button">Save</button>
<button onclick="#{rich:component('inputEmail')}.cancel();"
type="button">Cancel</button>
</f:facet>
</rich:inplaceInput>
```

Componentes Richfaces: Ui - Input

<rich:inplaceSelect>

- Semelhante ao <rich:inplaceInput> só que ao invés de deixar o usuário informar um valor livre ele exibe uma lista.

Select drink



Componentes Richfaces: Ui - Input

<rich:inplaceSelect>

```
<rich:inplaceSelect value="#{bean.drink}"
defaultLabel="Select drink">
<f:selectItem itemValue="1" itemLabel="Red wine" />
<f:selectItem itemValue="2" itemLabel="White wine" />
<f:selectItem itemValue="3" itemLabel="Beer" />
<f:selectItem itemValue="4" itemLabel="Vodka" />
<f:selectItem itemValue="5" itemLabel="Tequila" />
<f:selectItem itemValue="6" itemLabel="Sangria" />
</rich:inplaceSelect>
```

Componentes Richfaces: Ui - Input

<rich:inplaceSelect>

- Os mesmos controles aplicados ao inplaceInput podem ser aplicados para o inplaceSelect

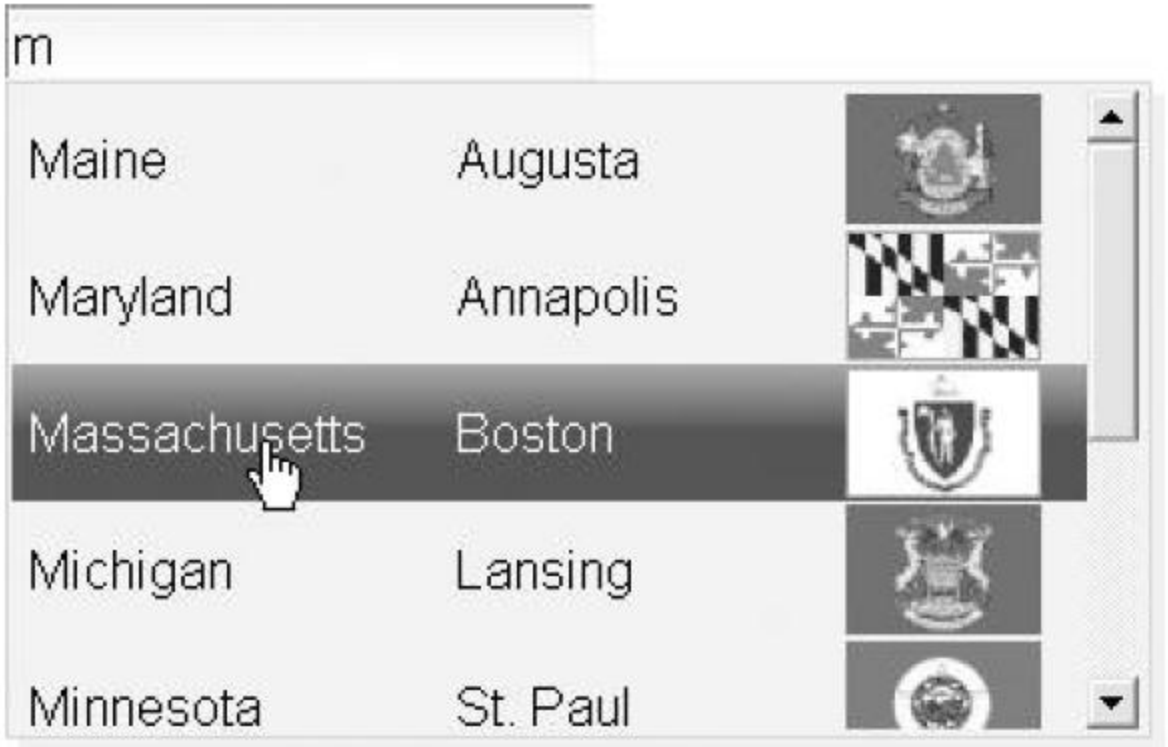
Componentes Richfaces: Ui - Input

<rich:suggestionbox>

- É uma combinação de um campo de entrada livre com um combo. A medida que o usuário digita os dados os elementos de escolha são filtrados.
- A cada letra que o usuário digita uma requisição é enviada ao servidor onde um ouvinte é acionado e a lista com os valores é retornada.

Componentes Richfaces: Ui - Input

<rich:suggestionbox>



Componentes Richfaces: Ui - Input

<rich:suggestionbox>

- O suggestionbox não contém um campo de input. Ele deve ser vinculado a algum campo de entrada para que possa ser utilizado.

```
<h:form>
<h:inputText value="#{statesSuggestionBean.state}" id="stateInput"/>
<rich:suggestionbox for="stateInput">
</rich:suggestionbox>
</h:form>
```

Componentes Richfaces: Ui - Input

<rich:suggestionbox>

- É possível associar cada item retornado a uma variável e realizar o tratamento necessário.

```
<rich:suggestionbox
suggestionAction="#{statesSuggestionBean.suggest}"
        var="state“ for="stateInput">
        <h:column>
                #{state.name}
        </h:column>
</rich:suggestionbox>
```

Componentes Richfaces: Ui - Input

<rich:suggestionbox>

- Fetch: Determina qual atributo deverá settar o valor do campo no qual o suggestion está associado. Se colunas estiverem sendo utilizadas o valor da primeira coluna é o padrão, mas para alterar qual campo deve ser alterado basta utilizar o fetch.
- Exemplo: Primeira coluna, nome, segunda telefone.
- Por padrão nome seria associado à variável do campo. Mas se o fetch for utilizado o campo telefone pode ser o campo utilizado.

Componentes Richfaces: Ui - Input

<rich:suggestionbox>

<rich:suggestionbox
suggestionAction="#{statesSuggestionBean.suggest}"
var="pessoa"
for="telefonePessoa"
width="350"
fetchValue="#{pessoa.telefone}">

Componentes Richfaces: Ui - Input

<rich:suggestionbox>

- Uma expressão pode ser utilizada no fetch:

fetchValue="#{pessoa.nome} – #{pessoa.telefone}“

- Outra opção importante é minChars. Que especifica o número mínimo de caracteres para que uma requisição seja realizada.
- A variável nothingLabel especifica uma mensagem para quando nenhum registro for encontrado.

Componentes Richfaces: Ui - Input

<rich:comboBox>

- Exibe uma lista com valores para o usuário, porém carrega todos os valores no cliente de uma vez.
- Pode ser criado utilizado os componentes <f:selectItem> e <f:selectItems> do JSF.
- Por exemplo:
- O combo da figura a seguir poderia ser criado de duas maneiras.

Componentes Richfaces: Ui - Input

<rich:comboBox>

Componentes Richfaces: Ui - Input

<rich:comboBox>

```
<rich:comboBox value="#{bean.drink}">
    <f:selectItem itemValue="Red wine" />
    <f:selectItem itemValue="White wine" />
    <f:selectItem itemValue="Beer"  />
    <f:selectItem itemValue="Vodka"  />
    <f:selectItem itemValue="Tequila"  />
    <f:selectItem itemValue="Sangria"  />
</rich:comboBox>
```

Componentes Richfaces: Ui - Input

<rich:comboBox>

```
<rich:comboBox value="#{bean.drink}"
    suggestionValues="#{bean.drinksList}">
```

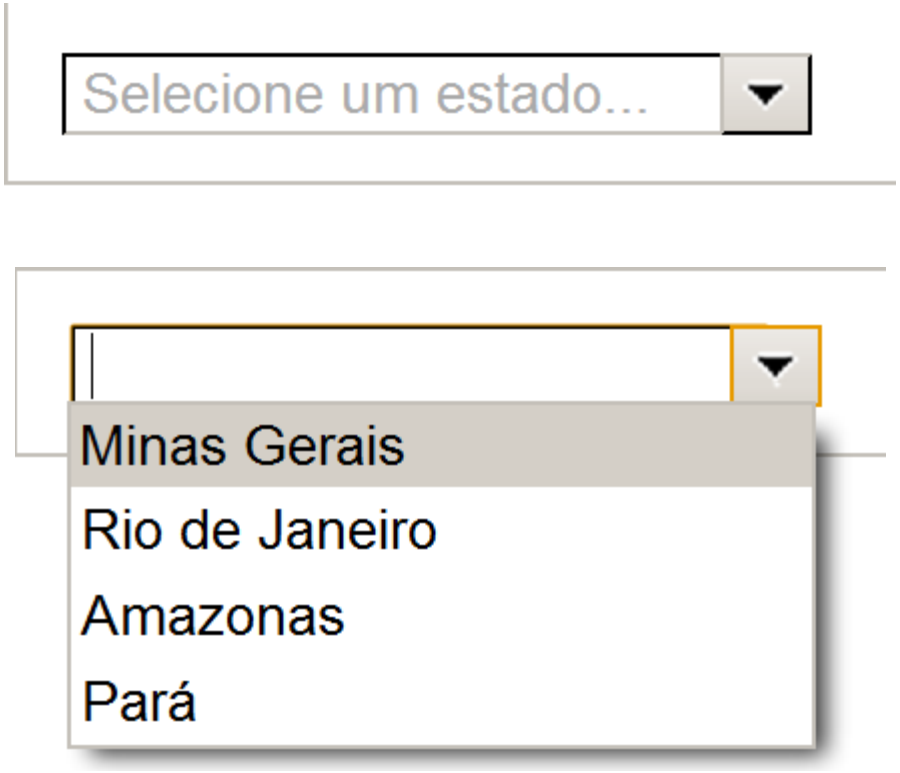
Componentes Richfaces: Ui - Input

<rich:comboBox>

- No primeiro caso a lista é criada no próprio componente.
- No segundo é utilizada uma coleção que deve existir no backing-bean.
- Outra opção no comboBox é o atributo defaultLabel. Este atributo permite definir um label padrão para o componente.

Componentes Richfaces: Ui - Input

<rich:comboBox>



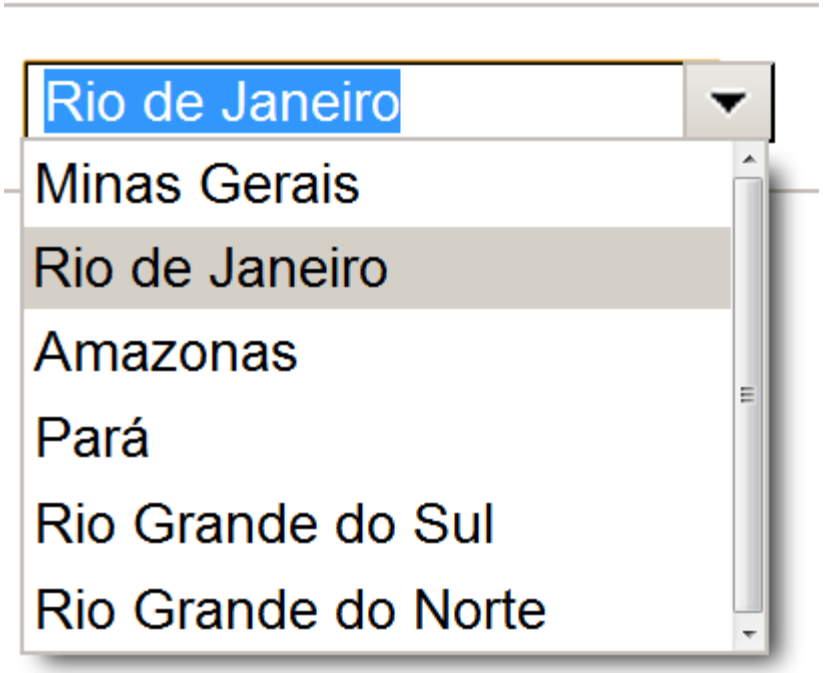
Componentes Richfaces: Ui - Input

<rich:comboBox>

- Outro recurso visualmente interessante é o atributo directInputSuggestions que exibe as opções em destaque de acordo com a digitação do usuário.

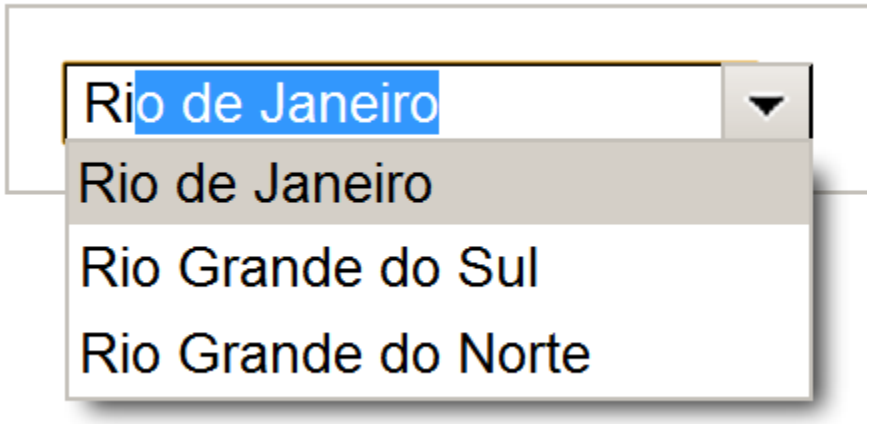
Componentes Richfaces: Ui - Input

<rich:comboBox>



Componentes Richfaces: Ui - Input

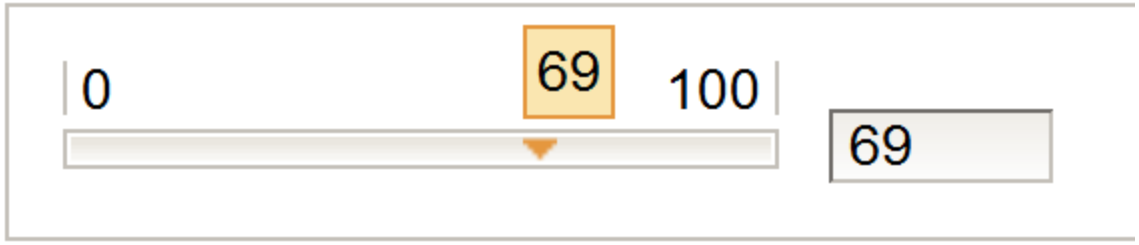
<rich:comboBox>



Componentes Richfaces: Ui - Input

<rich:inputNumberSlider>

- Exibe uma componente para a entrada de números em um intervalo definido.



Componentes Richfaces: Ui - Input

<rich:inputNumberSlider>

- É possível configurar :
  - Se o usuário pode informar um valor digitando:
    - enableManualInput.
  - Valor mínimo e máximo:
    - minValue;
    - maxValue.
  - Quantidade de passos:
    - Step;
  - Posição do campo:
    - inputPosition.
  - Exibição do campo de entrada:
    - showInput.

Componentes Richfaces: Ui - Input

<rich:inputNumberSpiner>

- Semelhante ao slider.
- Possui inclusive as mesmas opções de configuração.

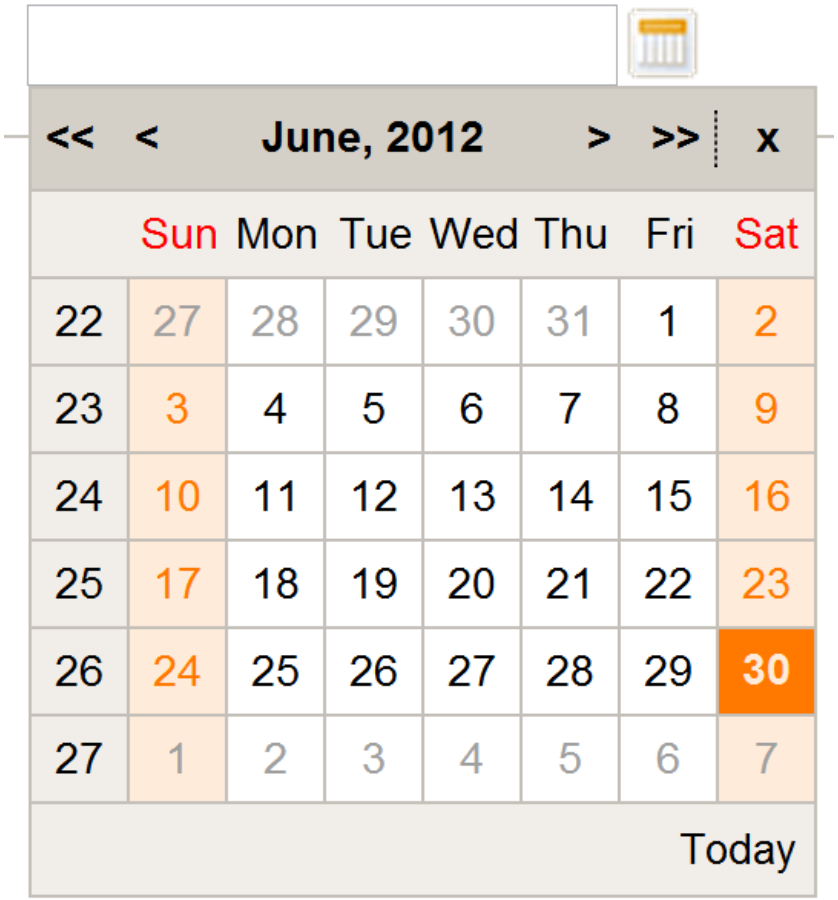
Componentes Richfaces: Ui - Input

<rich:calendar>

- Exibe um calendário que permite interação do usuário.
- Trabalha com data no formato java.util.Date.
- Permite a customização do formato de exibição de data.

Componentes Richfaces: Ui - Input

<rich:calendar>



Componentes Richfaces: Ui - Input

<rich:richDataTable>

- **Funciona de maneira semelhante ao h:dataTable com algumas funcionalidades extra como “skinnability”, atualização parcial de linhas e span em linhas e colunas.**

Componentes Richfaces: Ui - Input

<rich:richDataTable>

Nome	Código
American Airlines	AA
United Airlines	UA
Delta	DL
Northwest Airlines	NW
US Airways	US
Continental	CO

```
<rich:dataTable value="#{dataTable.companhias}"
var="companhia" >
        <h:column>
            <f:facet name="header">
                Nome
            </f:facet>
                #{companhia.nome}
        </h:column>
        <h:column>
            <f:facet name="header">
                Código
            </f:facet>
                #{companhia.codigo}
        </h:column>
</rich:dataTable>
```

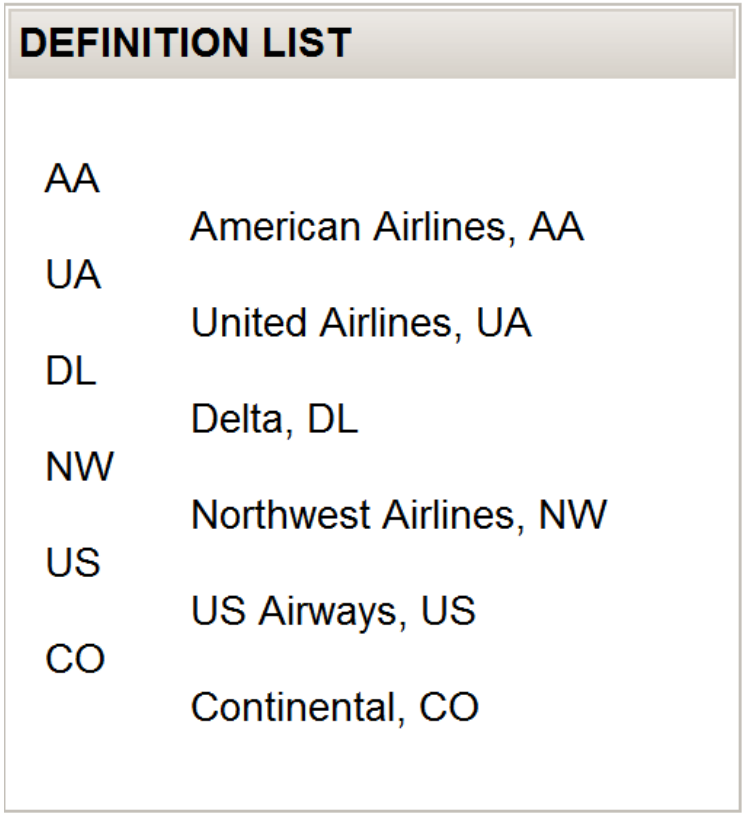
Componentes Richfaces: Ui - Input

<rich:richDataDefinitionList>, <rich:dataOrderedList> e <rich:dataList>

- **Criam listas de exibição dos dados.**
- **São mais simples que as tabelas.**
- **Veja os exemplos:**

Componentes Richfaces: Ui - Input

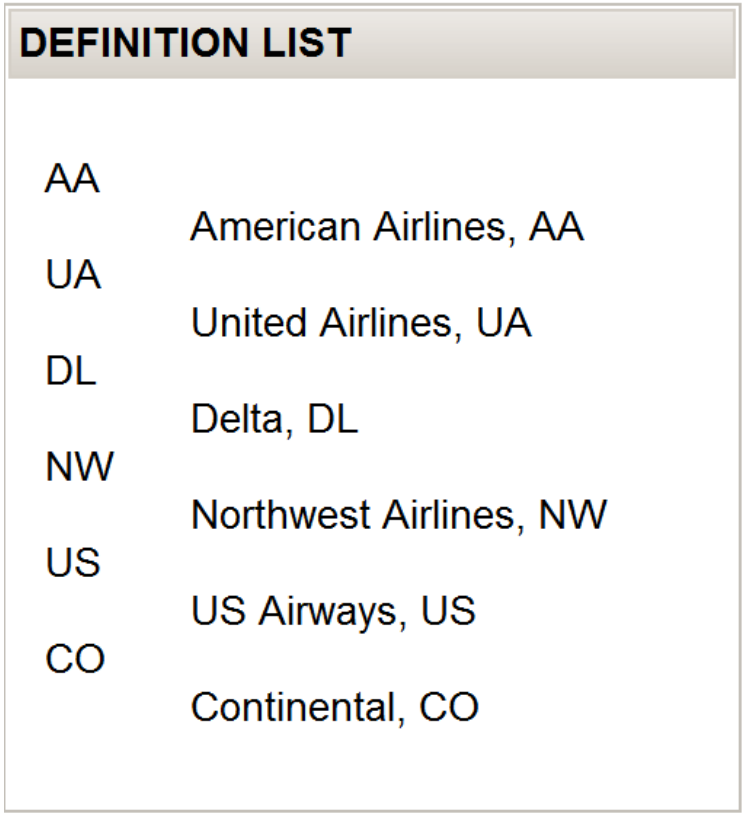
<rich:richDataTable>




```
<rich:panel style="width:200px">
  <f:facet name="header">
        DEFINITION LIST
  </f:facet>
  <rich:dataDefinitionList
        value="#{dataTable.companhias}"
        var="companhia">
        <f:facet name="term">
              #{companhia.codigo}
        </f:facet>
          #{companhia.nome},
          #{companhia.codigo}
  </rich:dataDefinitionList>
</rich:panel>
```

Componentes Richfaces: Ui - Input

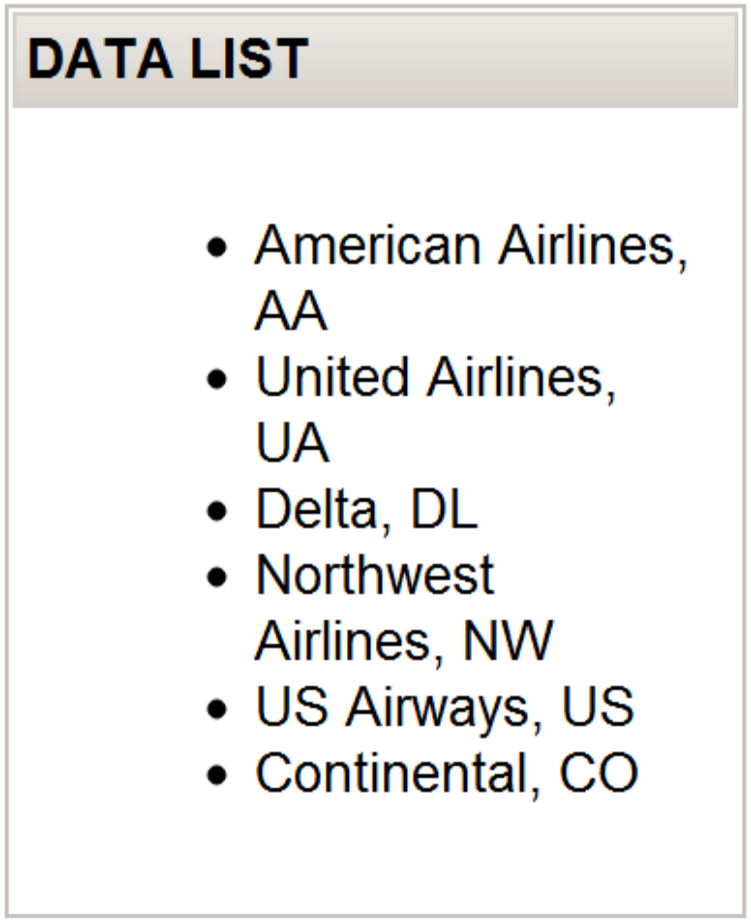
<rich:richDefinitionList>




```
<rich:panel style="width:200px">
  <f:facet name="header">
        DEFINITION LIST
  </f:facet>
  <rich:dataDefinitionList
        value="#{dataTable.companhias}"
        var="companhia">
        <f:facet name="term">
              #{companhia.codigo}
        </f:facet>
          #{companhia.nome},
          #{companhia.codigo}
  </rich:dataDefinitionList>
</rich:panel>
```

Componentes Richfaces: Ui - Input

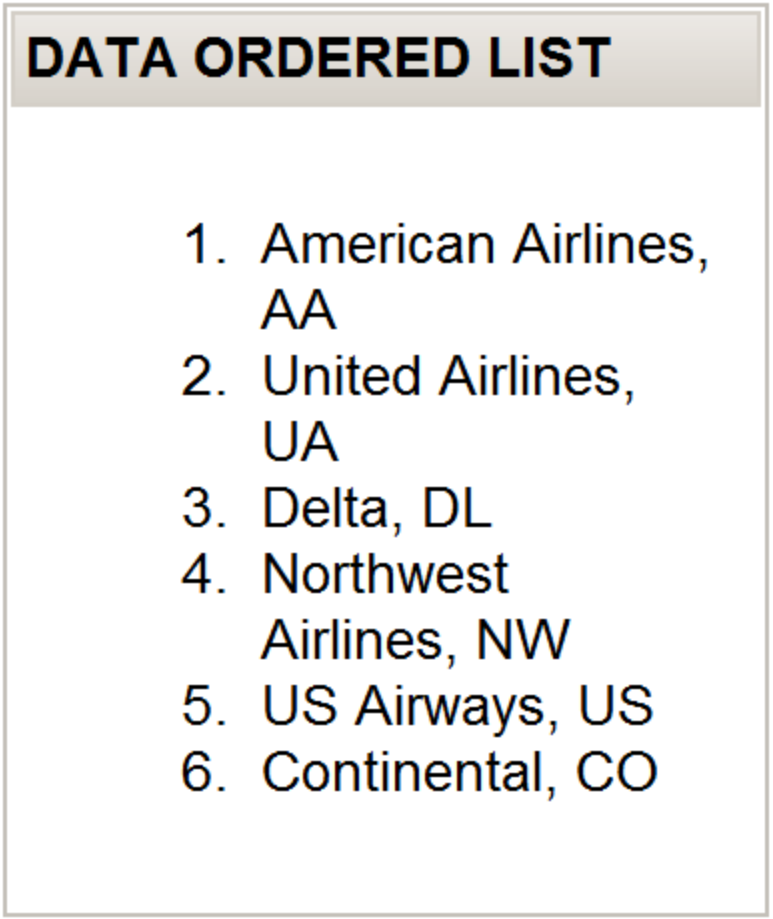
<rich:richDataList>




```
<rich:panel style="width:200px">
  <f:facet name="header">
        DATA LIST
  </f:facet>
  <rich:dataList
        value="#{dataTable.companhias}"
        var="companhia">
              #{companhia.nome},
            #{companhia.codigo}
  </rich:dataList>
</rich:panel>
```

Componentes Richfaces: Ui - Input

<rich:richOrderedDataList>




```
<rich:panel style="width:200px">
  <f:facet name="header">
        DATA LIST
  </f:facet>
  <rich:dataOrderedList
        value="#{dataTable.companhias}"
        var="companhia">
              #{companhia.nome},
             #{companhia.codigo}
  </rich:dataOrderedList>
</rich:panel>
```

Componentes Richfaces: Ui - Output

<rich:panel>

- Exibe um simples painel.

<rich:simpleTogglePanel>

- Exibe um painel que pode ser aberto e fechado.

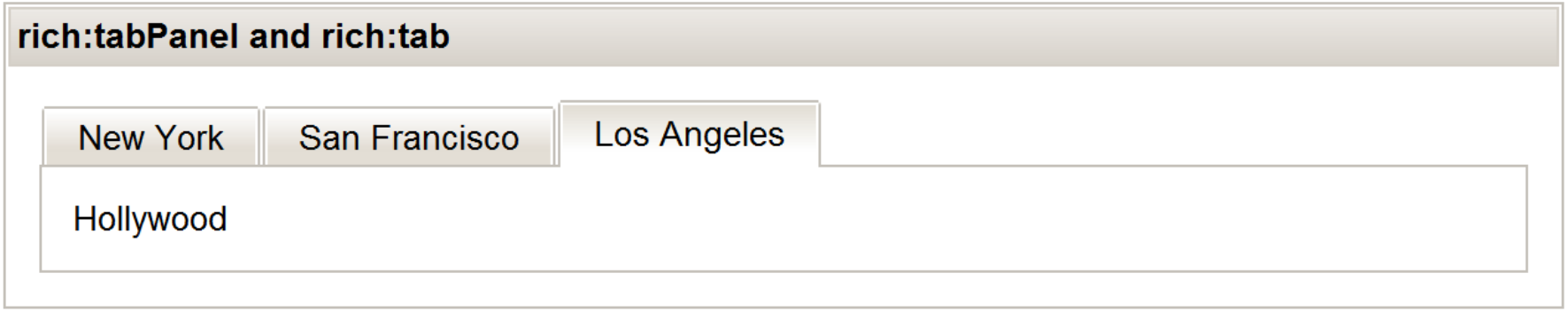
Componentes Richfaces: Ui - Output

<rich:tabPanel>

- Container para abas.

<rich:richTab>

- A aba.



Componentes Richfaces: Ui - Output

<rich:tabPanel>

- O atributo switchType. Indica se toda a página será carregada ou se uma requisição ajax irá carregas as abas.
- O atributo selectedTab indica qual a aba deve aparecer selecionada. O id da aba deve ser utilizado como identificador

<rich:richTab>

- O atributo disabled serve para indicar se uma aba está desabilitada e o atributo rendered indica se a aba deve ser exibida.

JBoss Seam

- Projeto liderado por Gavin King.
- Mesmo líder do projeto Hibernate.
- Criado em 2006.
- Desde o início teve boa aceitação da comunidade.
- Facilita a integração do JSF com EJBS, Spring, JPA(Hibernate).

JBoss Seam

- Componentes do Seam não precisam implementar nenhuma interface ou extender um classe. Eles podem ser:
  - Simples classes POJO;
  - Stateful Session Beans;
  - Stateless Session Beans;
  - Uma entidade JPA;
  - Etc.

JBoss Seam

- Os componentes são definidos através da anotação **@Name :**

```
@Stateless
@Name(“acaoCalculo")
public class AcaoCalculo implements Calculo {
...
}
```

JBoss Seam

- Como o Seam foi criado para realizar a “costura” entre a camada de fronteira e modelo ele fornece algumas funcionalidades para facilitar esta comunicação.
- Com o Seam é possível realizar a injeção(Injection) e exposição(Outjection) de objetos.
- Essa funcionalidade facilita em situações onde objetos precisam ser trafegados entre camadas e seja necessário a criação de um objeto de tranferência. Exemplo : Padrão de projeto Data Transfer Object

JBoss Seam

- **Injeção de dependência:** Injeta componentes de maneira transparente para o desenvolvedor que possam ser acessados pelo Seam.
- **Exposição de dependência:** Expõe o objeto para a aplicação.
- Para fazer a injeção a anotação @In deve ser utilizada no componente.
- Para fazer a exposição a anotação @Out deve ser utilizada.

JBoss Seam - Navegação

- Um dos pontos negativos do JSF é o controle de navegação bastante simplório.
- O Seam resolve isto de maneira fácil de ser utilizada.
- Três maneiras de navegação são possíveis:
- Navegação simples;
- Navegação no estilo JSF;
- Navegação Seam jPDL.

Navegação Simples

- É realizada apenas redirecionando o usuário para alguma página ou então voltando a resposta para a mesma página.
- Suponha que tenhamos um método executar em um backing-bean:
- Ao acionar o comando:

```
<h:commandButton action="#{bean.executar}"
value=“Executar" />
```

Navegação Simples

- O método executar do bean será acionado

```
@Name(“bean")
…
public String executar() {
return "";
}
```

- Como o método retorna uma String vazia, a resposta exibe a mesma página da requisição.

Navegação JSF

- Caso fosse necessário redirecionar o usuário para alguma página o método abaixo poderia ser utilizado:

```
...
public String executar() {
return “/outraPagina.html";
}
```

- Neste caso a resposta seria redirecionada para a outraPagina. Repare que destaquei a extensão. Ela importa. outraPagina.html pode não ser um arquivo html!!!

Navegação Seam

- Elimina a necessidade de criar código "*hard-coded*"para navegação.
- Geralmente no JSF os fluxos são definidos no arquivo faces-config.xml utilizando regras de navegação:

```
<navigation-rule>
  <from-view-id>/home.xhtml</from-view-id>
  <navigation-case>
   <from-action>executar</from-action>
   <from-outcome>sucesso</from-outcome>
   <to-view-id>/exemploSucesso.xhtml</to-view-id>
  </navigation-case>
 </navigation-rule>
```

Navegação Seam

- Essas regras podem ser globais :

```
<navigation-rule>
<from-view-id>/myPage.jsp</from-view-id>
<navigation-case>
<from-outcome>success</from-outcome>
<to-view-id>/success.jsp</to-view-id>
</navigation-case>
<navigation-case>
<from-outcome>fail</from-outcome>
<to-view-id>/failure.jsp</to-view-id>
</navigation-case>
</navigation-rule>
<!--Regra global-->
<navigation-rule>
<navigation-case>
<from-outcome>logout</from-outcome>
<to-view-id>/logout.jsp</to-view-id>
</navigation-case>
</navigation-rule>
```

Navegação Seam

- O Seam utiliza um conceito de navegação semelhante ao do JSF, porém separa a lógica de navegação da lógica de aplicação através de um arquivo de configuração.
- Geralmente o arquivo tem o nome:
  - nomePagina.pages.xml

Navegação Seam

- Além de separa a lógica de navegação da lógica de negócio ainda temos outras vantagens:
  - pages.xml permitem a passagem de parâmetros;
  - A Expression Language (EL) do JSP pode ser utilizada.
- O arquivo .pages.xml deve ficar no mesmo local do arquivo .xhtml/.jsp original para o qual a navegação será realizada.

Fluxo de Navegação Seam

- Um fluxo de navegação é geralmente definido especificando uma ou mais regras de navegação.
- Regras de navegação podem ser especificadas para uma simples página ou para um conjunto de páginas.
- Veja o exemplo a seguir

Fluxo de Navegação Seam

```
<?xml version="1.0" encoding="UTF-8"?>
<pages xmlns="http://jboss.com/products/seam/pages"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jboss.com/products/seam/pages
http://jboss.com/products/seam/pages-2.1.xsd">
  <page view-id="/ferias.jsp">
        <navigation from-action=    "#{acaoFerias.tipoFerias}">
              <rule if-outcome="“cidade">
                    <redirect view-id="/cidade.jsp" />
              </rule>
        </navigation>
   </page>
</pages>
```

Fluxo de Navegação Seam

- Os elementos de navegação são os seguintes:
  - Pages: agrupa um conjunto de páginas;
  - Page: Pode conter várias regras de navegação;
  - Navigation: Contém as definições da navegação com uma ou mais regras;
  - Rule: Define uma regra que é incorporada na navegação.
- Dos elementos citados vale ressaltar o elemento Rule.

Fluxo de Navegação Seam

- O elemento Rule permite definir a regra baseada em um retorno ou em uma expressão.
- Baseada em retorno:

```
<rule if-outcome="cidade">
<redirect view-id="/cidade.jsp" />
</rule>
```

- Baseada em expressão:

```
<rule if="#{valor lt 100.0}">
<redirect view-id="/valoresInsuficientes.jsp" />
</rule>
```

Fluxo de Navegação Seam

- No último exemplo poderiamos ter utilizado a EL(Expression Language) para cria a expressão.
- É importante ressaltar que as regras de navegação são executadas na ordem em que aparecem.