Validadores e Conversores

Prof. Leonardo Vianna do Nascimento
Disc. de Desenvolvimento de Sistemas I

Validadores

- Permitem verificar se o valor de um campo é válido ou não
- Validadores fornecidos pelo JSF
  - Campos requeridos
  - Validador de tamanho
  - Validadores de intervalo
  - Validador baseado em expressão regular

validateLength

- Permite definir um intervalo de tamanhos aceitos para o conteúdo que deve ser digitado em campos de texto
- Exemplo:

```
<h:inputText id="nome" value="#{bean.nome}">
    <f:validateLength minimum="10" maximum="30"/>
  </h:inputText>
```

A tag definindo o validador deve ser colocada dentro do elemento que será validado

Intervalos Numéricos

- Para campos numéricos, é possível especificar um intervalo de valores válidos através das tags *validateDoubleRange* e *validateLongRange*
  - `<f:validateDoubleRange minimum="mínimo" maximum="máximo" />`
  - `<f:validateLongRange minimum="mínimo" maximum="máximo" />`

validateRegex

- Permite validar um padrão textual, segundo uma expressão regular
- O que é uma expressão regular?
  - Consulte: http://pt.wikipedia.org/wiki/Express%C3%A3o_regular
- Exemplo

```
<h:inputText id="email" value="bean.email">

  <f:validateRegex pattern="[a-zA-Z0-9\-\_\.]
  +@[a-zA-Z0-9\-\_\.]+"/>

  </h:inputText>
```

Atributo *validatorMessage*

- Permite especificar uma mensagem de erro caso o conteúdo do componente não esteja de acordo com as regras de validação especificadas
- Exemplo

```
<h:inputText id="email" value="bean.email"
  validatorMessage="E-mail inválido!">

    <f:validateRegex pattern="[a-zA-Z0-9\-\_\.]+@[a-zA-Z0-
  9\-\_\.]+"/>

  </h:inputText>
```

Conversores

- Os conversores funcionam tanto para exibir dados formatados quanto apara entrada de dados formatados
- Números
  - Tag *convertNumber*
- Data e hora
  - Tag *convertDateTime*

convertDateTime

- Atributos
  - *type*
    - Especifica qual o tipo do valor sendo convertido (*date*, *time* ou *both*)
  - *dateStyle*
    - Formato da data (*default*, *short*, *medium*, *long* ou *full*)
  - *timeStyle*
    - Formato da hora (*default*, *short*, *medium*, *long* ou *full*)
  - *timeZone*
    - Fuso horário
  - *pattern*
    - Formato da data/hora definido manualmente

Exemplo

- Entrada de data no formato padrão

```
<h:inputText id="data" value="...">
    <f:convertDateTime type="date" />
  </h:inputText>
```

- Saída de hora no formato hh:mm

```
<h:outputText id="hora" value="...">
    <f:convertDateTime type="time"
                  pattern="HH:mm" />
  </h:outputText>
```

O Atributo *required*

- Impede que o formulário seja processado sem que o campo em questão tenha sido preenchido
- Exemplo:

```
<h:inputText id="nome" label="Nome"
        value="#{usuarioBean.usuario.nome}"
        size="30" maxlength="30"
        required="true"
        requiredMessage="Você não tem nome?">
```

O atributo *requiredMessage* permite definir uma mensagem de erro a ser exibida caso o campo não seja preenchido

convertNumber

- Atributos
  - *type*
    - Tipo do número (*number* – número normal,*currency* – moeda,*percent* – percentual)
  - *pattern*
    - Permite especificar um padrão personalizado para um número
  - *maxFractionDigits, minFractionDigits*
    - Número máximo e mínimo de dígitos após a vírgula
  - *maxIntegerDigits, minIntegerDigits*
    - Número máximo e mínimo de dígitos antes da vírgula

Exemplo

```
<h:outputText id="valorPreco"
        value="#{c.preco}">

      <f:convertNumber type="currency" />

  </h:outputText>
```