



Desenvolvimento para Servidores-II

Facelets

Neste tópico abordaremos o uso de Facelets para definição de um *layout* padrão para um conjunto de páginas JSF e o suporte a HTML5

Prof. Ciro Cirne Trindade

Facelets (1/2)

- Facelets se tornou a tecnologia de apresentação padrão a partir do JSF 2.0
- É um framework de *templating* que suporta todos os componentes JSF e é usado para construir e renderizar a árvore de componentes JSF para a saída da aplicação

Facelets (2/2)

- É comum que em uma aplicação web várias páginas possuam o mesmo *layout* básico
- Nesta situação é possível criar um *template* (modelo) de facelet que pode ser compartilhado por vários clientes de *facelets*

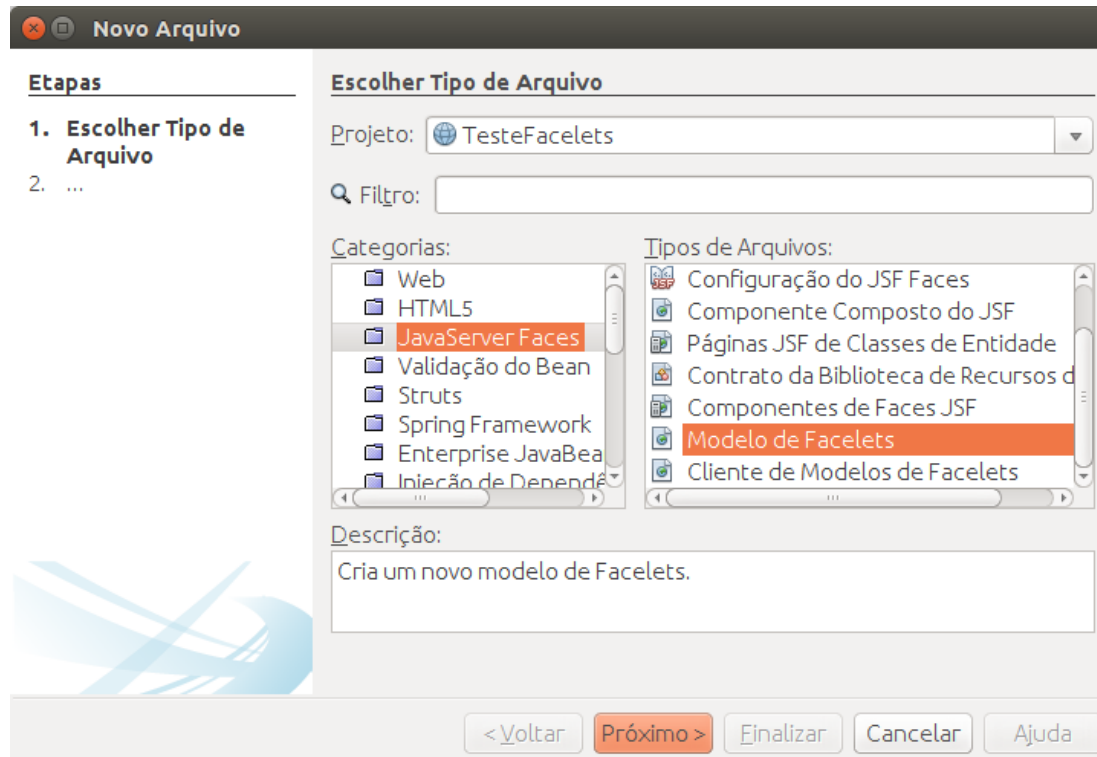
Tags Facelets

- Tags Facelets podem ser agrupadas em 2 categorias principais:
 - Inclusão de conteúdo de outras páginas XHTML (`ui:include`)
 - Construção de páginas a partir de *templates* (`ui:composition`, `ui:decorate`, `ui:insert`, `ui:define`, `ui:param`)
- Para utilizar tags Facelets acrescente o seguinte namespace a sua página JSF:
 - `xmlns:ui="http://xmlns.jcp.org/jsf/facelets"`

Criando um *template* de facelet (1/3)

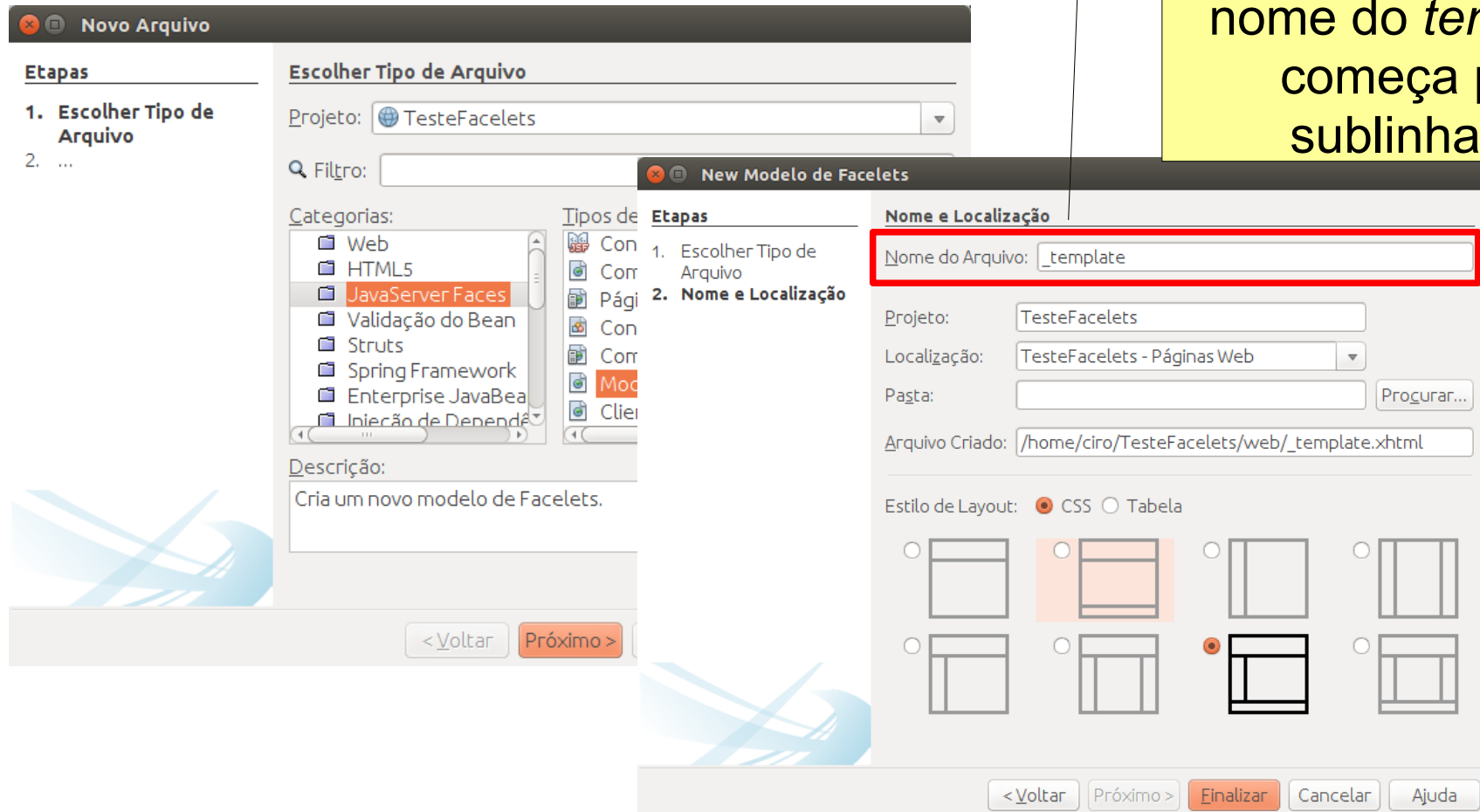
- O NetBeans possui um *wizard* para a criação de *templates* (modelo) de facelets e outro para a criação de clientes de facelets que utilizam o *template*

Criando um *template* de facelet (2/3)



Criando um *template* de facelet (3/3)

Por convenção o nome do *template* começa por sublinhado



Novo Arquivo

Etapas

1. Escolher Tipo de Arquivo
2. ...

Escolher Tipo de Arquivo

Projeto: TesteFacelets

Filtro:

Categorias:

- Web
- HTML5
- JavaServer Faces**
- Validação do Bean
- Struts
- Spring Framework
- Enterprise JavaBea
- Injeção de Dependência

Tipos de Arquivo:

- Con
- Cor
- Pági
- Con
- Cor
- Mod
- Clie

Descrição:

Cria um novo modelo de Facelets.

New Modelo de Facelets

Etapas

1. Escolher Tipo de Arquivo
2. Nome e Localização

Nome e Localização

Nome do Arquivo: _template

Projeto: TesteFacelets

Localização: TesteFacelets - Páginas Web

Pasta:

Procurar...

Arquivo Criado: /home/ciro/TesteFacelets/web/_template.xhtml

Estilo de Layout: ☒ CSS ☐ Tabela

Layout templates grid:

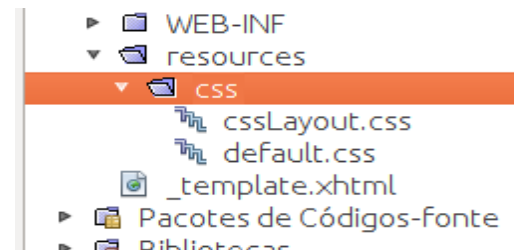
- Top-left
- Top-right**
- Bottom-left
- Bottom-right

Buttons: < Voltar, Próximo >, Finalizar, Cancelar, Ajuda

_template.xhtml (1/2)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=UTF-8" />
    <h:outputStylesheet name="./css/default.css"/>
    <h:outputStylesheet name="./css/cssLayout.css"/>
    <title>Facelets Template</title>
  </h:head>
```

Arquivos CSS
usados pelo
template

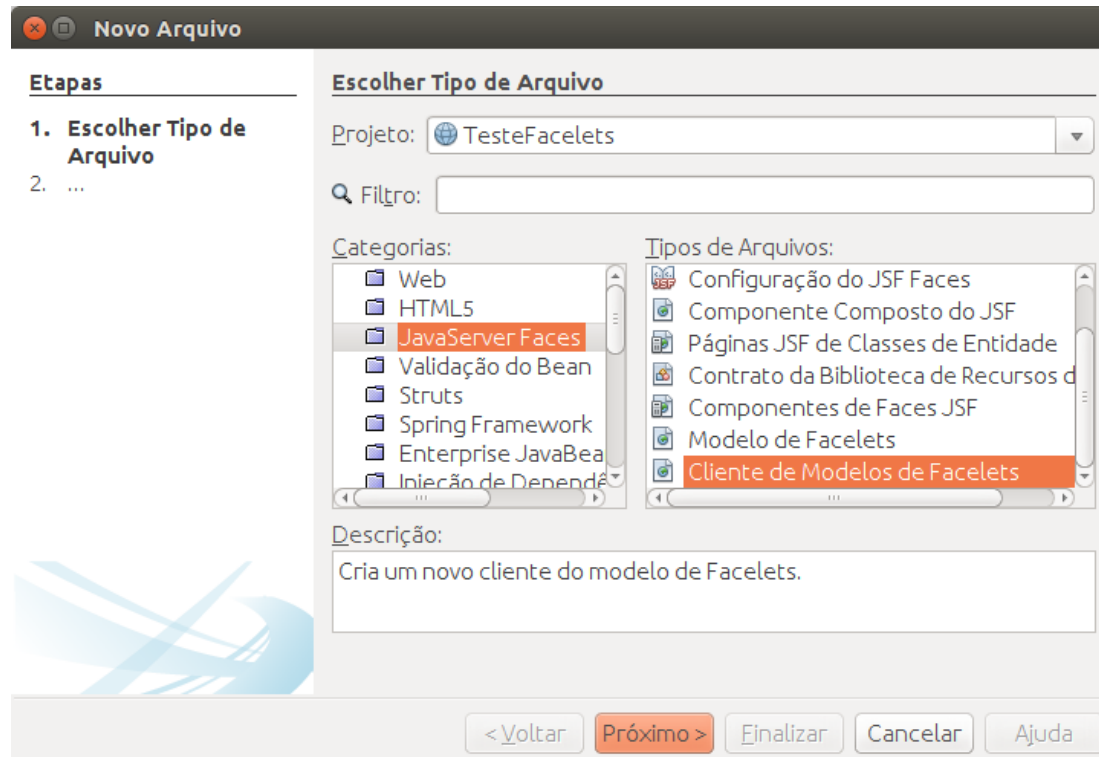


_template.xhtml (2/2)

```
<h:body>
  <div id="top">
    <ui:insert name="top">Top</ui:insert>
  </div>
  <div>
    <div id="left">
      <ui:insert name="left">Left</ui:insert>
    </div>
    <div id="content" class="left_content">
      <ui:insert name="content">
        Content</ui:insert>
      </div>
    </div>
    <div id="bottom">
      <ui:insert name="bottom">Bottom</ui:insert>
    </div>
  </h:body>
</html>
```

A tag `ui:insert` define aonde o conteúdo deve ser inserido

Criando um cliente de *template* de facelet (1/2)



Criando um cliente de *template* de facelet (2/2)

Novo Arquivo

Etapas

1. Escolher Tipo de Arquivo
2. ...

Escolher Tipo de Arquivo

Projeto:

Filtro:

Categorias:

- Web
- HTML5
- JavaServer Faces**
- Validação do Bean
- Struts
- Spring Framework
- Enterprise JavaBeans
- Interação de Dependência

Descrição:

Cria um novo cliente do n

[< Voltar](#)

New Cliente de Modelos de Facelets

Etapas

1. Escolher Tipo de Arquivo
2. **Nome e Localização**

Nome e Localização

Nome do Arquivo:

Projeto:

Pasta: [Procurar...](#)

Arquivo Criado:

Modelo: [Procurar...](#)

Tag Raiz Gerada: ☐ <html>

☒ <ui:composition>

Seções a Gerar:

<input checked="" type="checkbox"/>	top
<input checked="" type="checkbox"/>	left

[< Voltar](#) [Próximo >](#) [Finalizar](#) [Cancelar](#) [Ajuda](#)

index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<ui:composition xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    template="._template.xhtml">

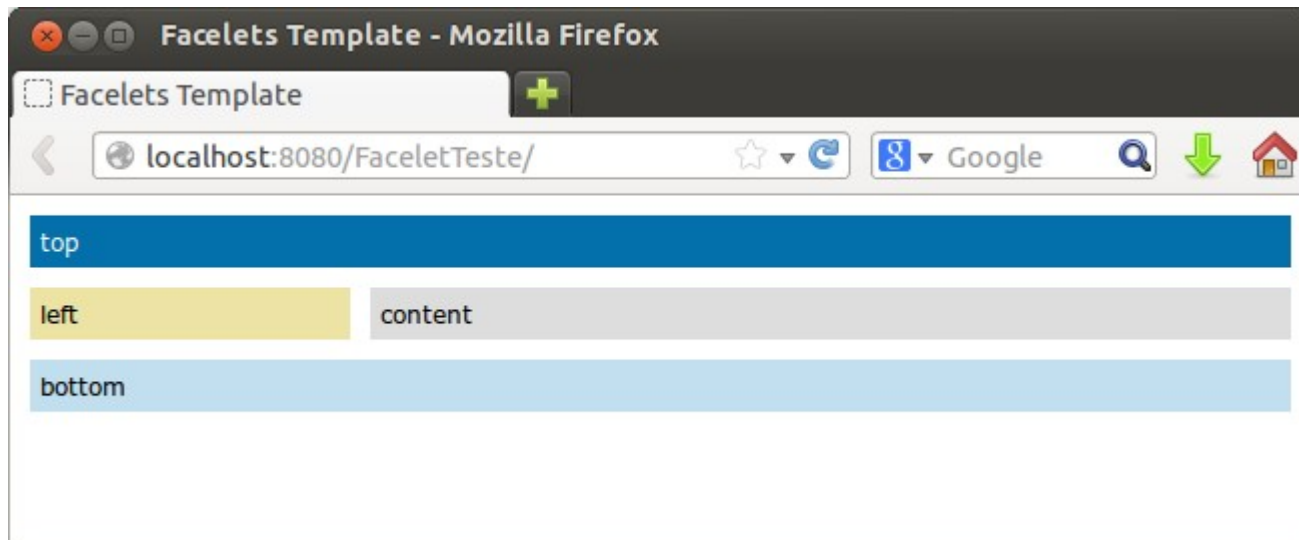
    <ui:define name="top">
        top
    </ui:define>
    <ui:define name="left">
        left
    </ui:define>
    <ui:define name="content">
        content
    </ui:define>
    <ui:define name="bottom">
        bottom
    </ui:define>

</ui:composition>
```

A associação da página com o *template* é feita através da tag `ui:composition`

As tags `ui:define` definem o conteúdo a ser inserido nas tags `ui:insert` do *template*

Renderização da página index.xhtml



Definindo uma imagem de cabeçalho (1/3)

- Uma imagem pode ser inserida em uma página JSF através da tag `<h:graphicImage>`
- As imagens devem ficar em uma subpasta de `resources`, por exemplo, `img`
- O atributo `library` da tag indica a pasta aonde a imagem está e o atributo `name` indica o nome da imagem
- Por exemplo:
 - `<h:graphicImage library="img" name="logo.png" />`

Definindo uma imagem de cabeçalho (2/3)

- Vamos incluir a imagem e um título no cabeçalho do *template*

```
<div id="top">  
    <h:graphicImage library="img"  
                    name="logo.png"/>  
    <h1><ui:insert name="titulo">  
        </ui:insert></h1>  
</div>
```

Definindo uma imagem de cabeçalho (3/3)

- Na página que utiliza o *template* basta definir o valor do título

```
<ui:define name="titulo">
```

```
    Página Inicial
```

```
</ui:define>
```


Incluindo pedaços de conteúdo

- É possível incluir pedaços de conteúdo no *template* ou no cliente através da tag `<ui:include>`

```
<ui:insert name="bottom">
```

```
    <ui:include src="./footer.xhtml"/>
```

```
</ui:insert>
```

Suporte a HTML5 (1/3)

- HTML5 oferece novos elementos e atributos e muitas novas capacidades para componentes antigos
- Por exemplo, o atributo **type** de elementos **input** agora suporta valores como `text`, `search`, `email`, `url`, `tel`, `range`, `number` e `date`

Suporte a HTML5 (2/3)

- Ao invés de incluir novos componentes com que imitam os componentes do HTML5, JSF suporta HTML5 permitindo que você use as tags HTML5 diretamente
- Também é possível usar atributos JSF em elementos HTML5

Suporte a HTML5 (3/3)

- O suporte do JSF a HTML5 recai em duas categorias:
 - Elementos *pass-through*
 - Atributos *pass-through*

Usando elementos *pass-through* (1/2)

- Elementos *pass-through* permitem que você use tags e atributos HTML5, mas trata-os como componentes JSF
- Para tornar um elemento que não é JSF um elemento *pass-through*, especifique pelo menos um de seus atributos usando o namespace `http://xmlns.jcp.org/jsf`

Usando elementos *pass-through* (2/2)

- Por exemplo, o código a seguir declara o namespace com o prefixo `jsf`:

```
<html ... xmlns:jsf="http://xmlns.jcp.org/jsf"
```

```
...
```

```
<input type="email" jsf:id="email" name="email"  
      value="#{reserva.email}"  
      required="required"/>
```

- Aqui, o prefixo `jsf` foi usado no atributo `id`, de tal forma que os atributos da tag `input` do HTML5 são tratadas como parte da página JSF
- Isto significa que você pode, por exemplo, usar expressões EL para acessar propriedades de um *managed bean*

Usando atributos *pass-through* (1/2)

- Atributos *pass-through* permitem passar atributos que não são atributos JSF para o browser sem interpretação
- Se você especificar um atributo *pass-through* em um componente JSF, o nome do atributo e seu valor são passados diretamente para o browser sem serem interpretados por componentes ou renderizadores JSF

Usando atributos *pass-through* (2/2)

- Há 3 formas diferentes de especificar atributos *pass-through*:
 - Usando um atributo do namespace "http://java.sun.com/jsf/passthrough" na tag do componente
 - Usando a tag `f:passThroughAttribute` para informar apenas um atributo
 - Usando a tag `f:passThroughAttributes` para informar múltiplos atributos

Usando um atributo do namespace "http://java.sun.com/jsf/passthrough"

- Use um namespace JSF para prefixar os atributos *pass-through* do componente JSF
- Por exemplo, o código a seguir declara o namespace com prefixo `p`, e então utiliza os atributos `type`, `min`, `max`, `required` e `title` do HTML5 em um `h:inputText` do JSF:

```
<html ...  
  xmlns:p="http://xmlns.jcp.org/jsf/passthrough"  
  ...  
<h:form prependId="false">  
  <h:inputText id="nights" p:type="number"  
    value="#{reserva.noites}"  
    p:min="1" p:max="30" p:required="required"  
    p:title="Informe um número entre 1 e 30"> 25  
  ...
```

Usando a tag

`f:passThroughAttribute`

- Para passar um único atributo, inclua a tag `f:passThroughAttribute` dentro da tag de um componente JSF
- Por exemplo:

```
<h:inputText value="#{reserva.email}">  
    <f:passThroughAttribute name="type"  
                            value="email" />  
</h:inputText>
```

Usando a tag

`f:passThroughAttributes` (1/2)

- Para passar um grupo de atributos, inclua a tag `f:passThroughAttributes` dentro da tag de um componente JSF, especificando um valor EL que deve ser um `Map<String, Object>`

- Por exemplo:

```
<h:inputText value="#{bean.nights}">
    <f:passThroughAttributes
        value="#{reserva.nameValuePairs}" />
</h:inputText>
```

Usando a tag

`f:passThroughAttributes (2/2)`

- O *managed bean* teria possuir um atributo do tipo **Map** que poderia ser inicializado no construtor da classe, como ilustra o código a seguir:

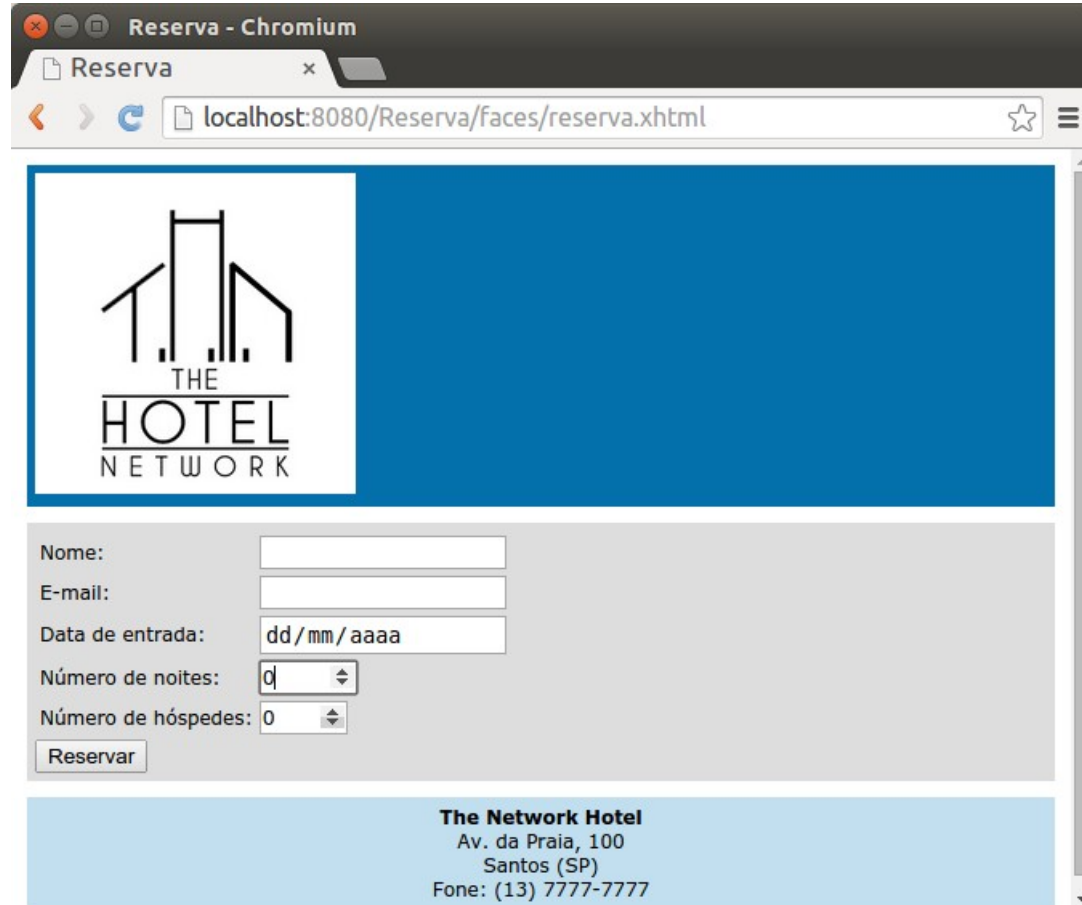
```
private Map<String, Object> nameValuePairs;

public Reserva() {
    this.nameValuePairs = new HashMap<>();
    this.nameValuePairs.put("type", "number");
    this.nameValuePairs.put("min", "1");
    this.nameValuePairs.put("max", "30");
    this.nameValuePairs.put("required", "required");
    this.nameValuePairs.put("title",
        "Informe um número entre 1 e 30");
}
```

Exemplo de uma aplicação de reserva (1/2)

- Vamos implementar uma aplicação para reserva de quarto em um hotel usando funcionalidades do HTML5
- Recursos web da aplicação
 - Managed bean: Reserva.java
 - Páginas web: reserva.xhtml, confirmacao.xhtml, footer.xhtml
 - Template: _template.xhtml
 - Imagem: hotel_logo.jpg

Exemplo de uma aplicação de reserva (2/2)



The screenshot shows a web browser window titled "Reserva - Chromium". The address bar displays "localhost:8080/Reserva/faces/reserva.xhtml". The page features a logo for "THE HOTEL NETWORK" on the left and a large blue rectangular area on the right. Below the logo, there is a reservation form with the following fields and controls:

- Nome:
- E-mail:
- Data de entrada:
- Número de noites: - Número de hóspedes: - Reservar:

At the bottom of the page, there is a light blue footer containing the following text:

The Network Hotel
Av. da Praia, 100
Santos (SP)
Fone: (13) 7777-7777

Reserva.java (1/3)

```
package beans;

import java.util.Date;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

@ManagedBean
@RequestScoped
public class Reserva {
    private static final double DIARIA_SIMPLES = 190.;
    private static final double DIARIA_DUPLA = 350.;
    private static final double DIARIA_TRIPLO = 500.;
    private static final double DIARIA_QUADRUPLA = 620.;
    private String nome;
    private String email;
    private Date entrada;
    private int noites;
    private int hospedes;
    private double valor;

    public Reserva() { }
```

Reserva.java (2/3)

```
public String getEmail() { return email; }

public void setEmail(String email) {
    this.email = email;
}

public String getNome() { return nome; }

public void setNome(String nome) {
    this.nome = nome;
}

public Date getEntrada() { return entrada; }

public void setEntrada(Date entrada) {
    this.entrada = entrada;
}

public int getNoites() { return noites; }

public void setNoites(int noites) {
    this.noites = noites;
}
```


Reserva.java (3/3)

```
public int getHospedes() { return hospedes; }

public void setHospedes(int hospedes) {
    this.hospedes = hospedes;
}

public double getValor() { return valor; }

public String reservar() {
    switch(hospedes) {
        case 1: valor = DIARIA_SIMPLES * noites;
                break;
        case 2: valor = DIARIA_DUPLO * noites;
                break;
        case 3: valor = DIARIA_TRIPLO * noites;
                break;
        case 4: valor = DIARIA_QUADRUPLO * noites;
                break;
    }
    return "/confirmacao";
}
}
```

_template.xhtml (1/2)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

  <h:head>
    <meta http-equiv="Content-Type" content="text/html;
          charset=UTF-8" />
    <h:outputStylesheet name="./css/default.css"/>
    <h:outputStylesheet name="./css/cssLayout.css"/>
    <title>
      <ui:insert name="title">template</ui:insert>
    </title>
  </h:head>
```

_template.xhtml (2/2)

```
<h:body>
  <div id="top">
    <ui:insert name="top">
      <h:graphicImage library="img"
        name="hotel_logo.jpg" height="200"
        width="200"/>
    </ui:insert>
  </div>

  <div id="content" class="center_content">
    <ui:insert name="content">Content</ui:insert>
  </div>

  <div id="bottom">
    <ui:insert name="bottom">
      <ui:include src="./footer.xhtml"/>
    </ui:insert>
  </div>
</h:body>

</html>
```

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    <b>The Network Hotel</b><br/>
    Av. da Praia, 100<br/>
    Santos (SP)<br/>
    Fone: (13) 7777-7777
  </h:body>
</html>
```

reserva.xhtml (1/2)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://xmlns.jcp.org/jsf/passthrough"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

  <body>
    <ui:composition template="._template.xhtml">
      <ui:define name="title">Reserva</ui:define>
      <ui:define name="content">
        <h:form>
          <h:panelGrid columns="2">
            Nome:
            <h:inputText value="#{reserva.nome}"
                          required="true" label="nome"/>
            E-mail:
            <h:inputText value="#{reserva.email}"
                          p:type="email" required="true"
                          label="e-mail"/>
          </h:panelGrid>
        </h:form>
      </ui:define>
    </ui:composition>
  </body>
</html>
```

Data de entrada:

```
<h:inputText value="#{reserva.entrada}"  
    p:type="date" required="true" label="data">  
    <f:convertDateTime pattern="yyyy-MM-dd"/>  
</h:inputText>
```

Número de noites:

```
<h:inputText value="#{reserva.noites}"  
    p:type="number" p:min="1" p:max="30"  
    required="true" label="noites"  
    p:title="Informe um número entre 1 e 30"/>
```

Número de hóspedes:

```
<h:inputText value="#{reserva.hospedes}"  
    p:type="number" p:min="1" p:max="4"  
    required="true" label="hóspedes"/>
```

```
</h:panelGrid>
```

```
<h:commandButton value="Reservar"  
    action="#{reserva.reservar}"/>
```

```
</h:form>
```

```
</ui:define>
```

```
</ui:composition>
```

```
</body>
```

```
</html>
```

confirmacao.xhtml (1/2)

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

  <body>

    <ui:composition template="._template.xhtml">

      <ui:define name="title">
        Confirmação
      </ui:define>

      <ui:define name="content">
        <h1>Confirmação da Reserva</h1>
        <h:panelGrid columns="2">
          Nome:
          <h:outputText value="#{reserva.nome}"/>
          E-mail:
          <h:outputText value="#{reserva.email}"/>
        </h:panelGrid>
      </ui:define>
    </ui:composition>
  </body>
</html>
```

confirmacao.xhtml (2/2)

```
Data de entrada:
<h:outputText value="#{reserva.entrada}">
    <f:convertDateTime
        pattern="dd/MM/yyyy"/>
</h:outputText>
Número de noites:
<h:outputText value="#{reserva.noites}"/>
Número de hóspedes:
<h:outputText
    value="#{reserva.hospedes}"/>
Valor da reserva:
<h:outputText value="#{reserva.valor}">
    <f:convertNumber type="currency"/>
</h:outputText>
</h:panelGrid>
<h:button value="Voltar" outcome="/reserva"/>
</ui:define>
```

```
</ui:composition>
```

```
</body>
</html>
```


Referências

- GEARY, David; HORSTMANN, Cay. *Core JavaServer Faces*. 3. ed., Prentice-Hall, 2010.
- ORACLE Corporation. *The Java EE 7 Tutorial*. Disponível em: <https://docs.oracle.com/javaee/7/JEETT.pdf>, 2014.