Layout e Páginas JSF

José Ribamar Monteiro da Cruz

October 14, 2016

Contents

1 Objetivo **1**

2 Introdução **2**

3 Instalação **3**

4 Convenções e Boas Práticas **3**

5 Template Padrão (main.xhtml) **4**
5.1 Regiões do Template Padrão 4
5.2 Código XHTML do Template Padrão 5

6 Conceito de Página **7**
6.1 Organização das Páginas 7
6.1.1 Organização da Página de Usuários 7
6.1.2 Organização da Página de Gerenciamento de Departamentos 8
6.2 Código Java da Página de Usuário (UsuarioPG.java) 8
6.3 Código XHTML da Página de Usuários (usuarioPage/index.xhtml) 10
6.4 Código Java da Página de Genreciamento de Departametos (DepartamentoMngtPG.java) 11
6.5 Código XHTML da Página de Genreciamento de Departametos (departamentoMngtPage/index.xhtml) . . 12

7 Reescrevendo URLs com Pretty-Faces **12**
7.1 Instalação 12
7.2 Mapeamento de Páginas no Pretty-Faces 13

8 Conclusão **13**

1 Objetivo

O presente documento tem como objetivo apresentar uma proposta de organização de código para projetos JSF utilizando o conceito de páginas (*pages*). Tal proposta visa criar uma correlação entre os **mockups de tela**, os **códigos java** necessários para montar tal tela bem como os respectivos **códigos XHTMLs**.

Juntamento com este conceito (*pages*) é apresentado um forma de criação de um **template padrão** configurável que torna a criação de templates xhtml flexivéis e evita a proliferação de diversos templates pelo projeto que possuam variações mínimas entre si.

2 Introdução

Para iniciarmos as explicações sobre os conceitos de *layouts e páginas* vamos utilizar primeiramente a abordagem tradicional e em seguinda utilizar o conteito de páginas(pages) para reeoganizar o código e ver as diferença entre ambos.

Para tanto, vamos supor que para o sistema de exemplo (**sisdep**) idetificamos as entidade *Usuario*, *Departamento e Documento*, o que resultará na modelagem do diagrama de classe conforme a figura 1.

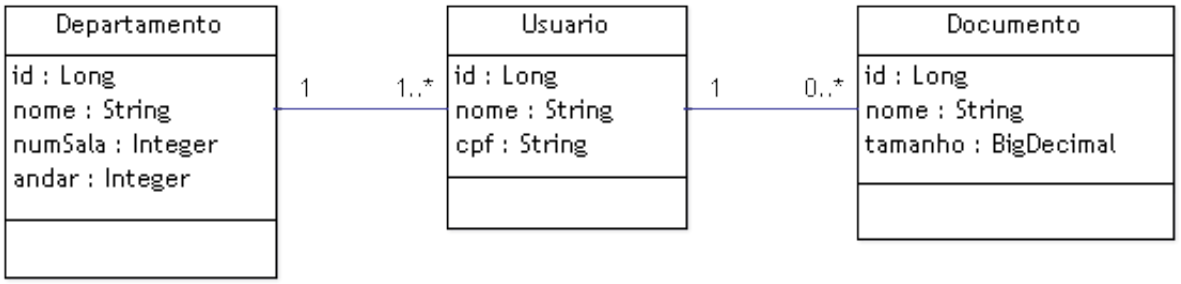


Figure 1: Diagrama de classe do sistema de exemplo.

Após a modelagem inicial, identificamos que o sistema é divido em duas páginas: **Página de Usuários** e **Página de Gerenciamento de Departamentos**. Para a **Página de Usuários** temos as seguintes funcionalidades:

- Listagem de todos os Departamentos.
- Listagem de usuários para um dado departamento.
- Criação/Alteração/Exclusão de usuários para um dado departamento.
- Impressão de relatórios de documento associados aos usuários.

Já para a **Página de Gerenciamento de Usuários** temos:

- Listagem de todos os departamentos.
- Criação/Alteração/Exclusão de departamentos.

Abaixo temos um esborço de como essas páginas poderiam ser implementadas:

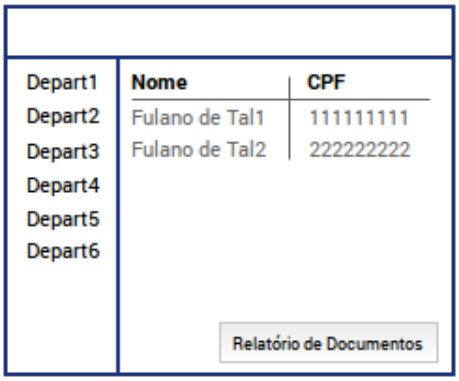


Figure 2: Página de Usuários por Departamento.

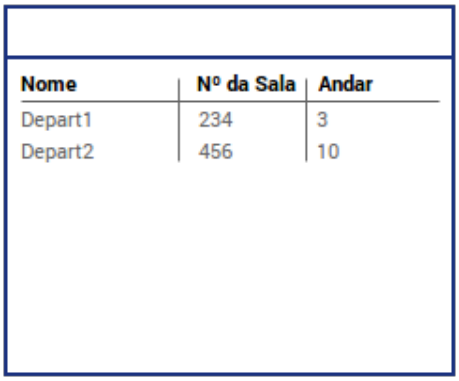


Figure 3: Página de Gerenciamento de Departamentos.

Uma forma comum de organização de *managed beans* e *arquivos XHTMLs* em projeto JSF para o sistema é ilustrada no esquema abaixo:

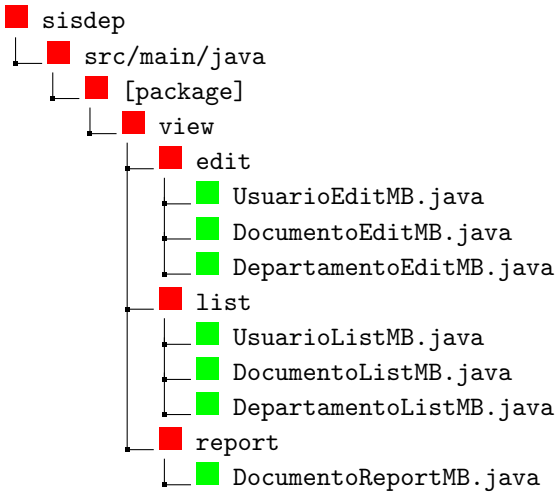


Figure 4: Estrutura de Pacotes.

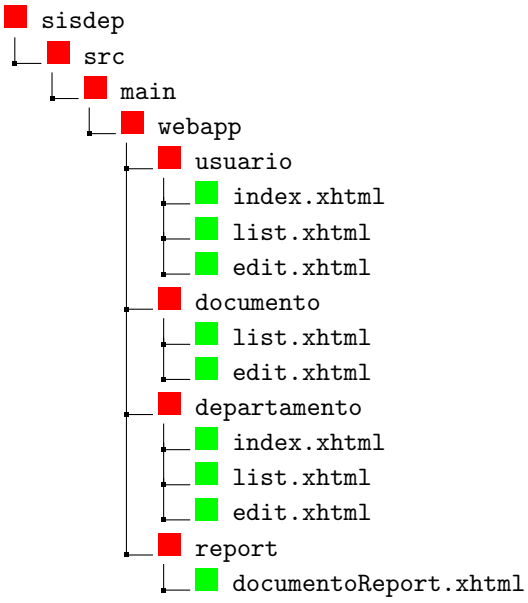


Figure 5: Estrutura de XHTMLs.

Com a estrutura de código (tanto java quando XHTML) mostrada algumas indagações surgem:

1. Apenas analisando o código fonte, quantas páginas o sistema possui?
2. Quais ***managed beans*** *e* ***XHTMLs*** pertencem a Página de Usuários?
3. E quais ***managed beans*** *e* ***XHTMLs*** pertencem a Página de Gerenciamento de Departamentos?

Como podemos observar a resposta não parece óbvia a primeira vista e tende a se tornar mais difícil com o aumento do número de entidades e das páginas derivadas. Tal dificuldade tem impacto direto no desenvolvimento tendo em vista na dificuldade em se enxergar uma correspondência entre os mockups e o código resultante. A manutenção futura também é prejudicada pelo mesmo motivo.

Exposto essa problemática, vamos analizar uma outra abordagem para o problema utilizando o **conceito de página**.

3 Instalação

Todo o código necessário para implementarmos a estuturação por páginas está contido no projeto jbase[1] e é ”instalado” no projeto adicionando a dependência no pom.xml

```
1 <dependency>
2    <groupId>br.jus.tre_pa.jbase</groupId>
3    <artifactId>jbase</artifactId>
4    <version>0.0.3</version>
5 </dependency>
```

4 Convenções e Boas Práticas

Segue abaixo algumas convenções adotas no decorrer do documento:

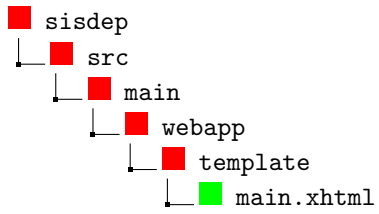
- O nome do template padrão é **main.xhtml** e esta localizado no diretório src/main/webapp/view/template;
- As **pages** estão localizadas dentro do pacote src/main/java/*[package]*/view/;
- O pacote que contém o conteúdo (dialog, datatables e etc.) da **página** é sufixado com **Page**. *(ex: usuarioPage)*;
- O managed bean controlador do template é sufixado com **PG**. *(ex: UsuarioPG.java)*;

[1] Para utilizar o jbase é necessário adicionar o repositório *http://nexus.tre-pa.gov.br:8090/nexus/content/groups/public* no *pom.xml*

- O managed bean controlador do template de páginas que representem o conceito de gerenciamento é sufixado com **MngtPG**. *(ex: DepartamentoMngtPG.java)*;
- O arquivo XHTML que contém as definições(ui:define) da page possui o nome de **index.html**
- Evitar o reuso de códigos Java e XHTML entre páginas.[2]
- Incluir os componentes de dialogs (p:dialog, p:confirmDialog . . .) na região **document.body** do template.[3].
- Nomear as classes Java para que não haja colisão entre páginas, ou seja, não poderá existir duas classes chamadas ***departamentoTreeMB*** mesmo que estejam em pacotes diferentes.

5 Template Padrão (main.xhtml)

Conforme dito anteriormente, a proposta do template padrão é de reduzir a quantidade de templates que venham a surgir no projeto JSF. O arquivo está localizado conforme a estrutura de diretório abaixo:



5.1 Regiões do Template Padrão

O template padrão é dividido em diversas *regiões* configuráveis (que podem ser exibidas ou não) dinâmicamente de acordo com o estrutura de cada página. A figura 6 ilustra graficamente as regiões do template (as regiões em cinza representam as partes configuráveis dinamicamente):

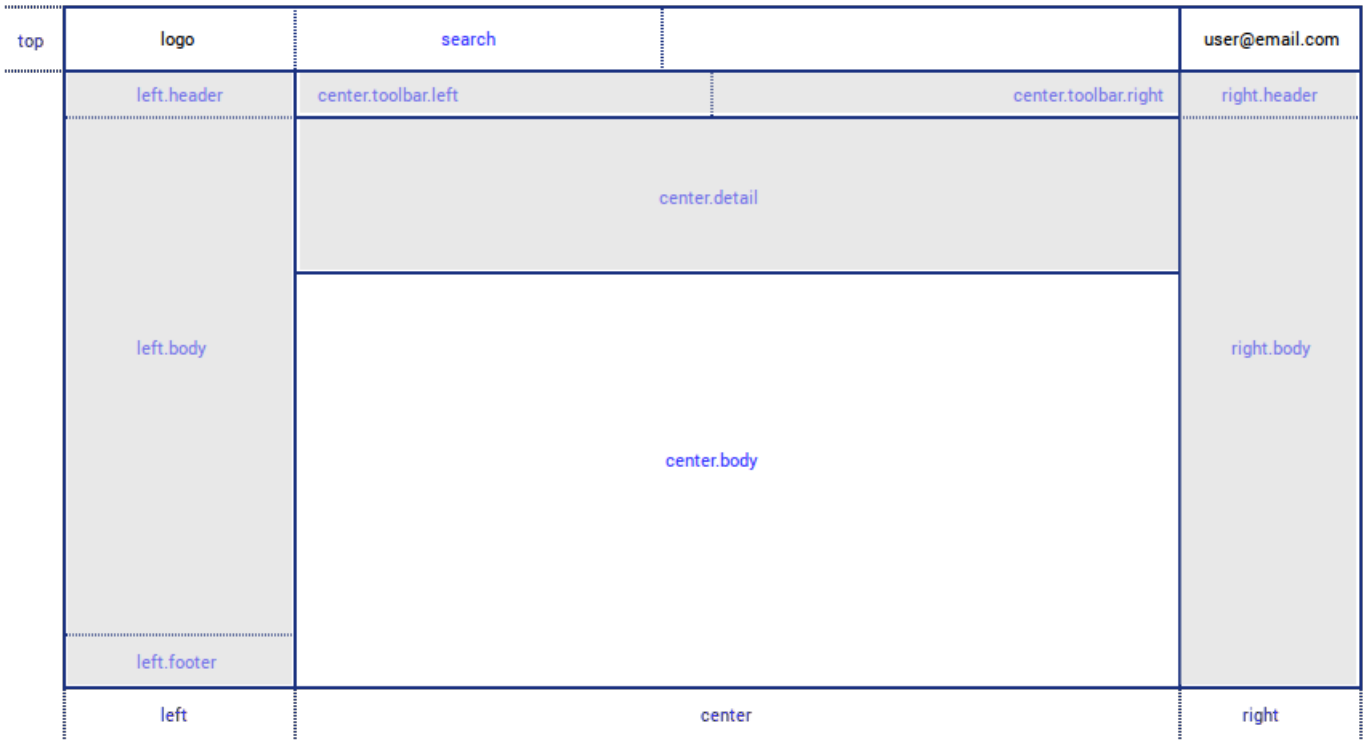


Figure 6: Regiões do template padrão

[2]Com esta abordagem criamos linhas independentes de código evitando conflito de merges
[3]Evita-se com isso alguns bugs relacionados a componentes que precisem de *overlay*

5.2 Código XHTML do Template Padrão

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns:f="http://java.sun.com/jsf/core" xmlns:p="http://primefaces.org/ui"
    xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets" xmlns="http://www.w3.org/1999/xhtml">

<f:view contentType="text/html" locale="#{locales.currentLocale}" encoding="UTF-8" />

<h:head>
   <title>[NOME DO SISTEMA]</title>
   <h:outputStylesheet library="css" name="layout.css" />
   <h:outputStylesheet library="css" name="app.css" />
   <h:outputStylesheet library="css" name="font.css" />

   <h:outputScript library="js" name="primefaces-locale-pt_br.js" />
   <h:outputScript library="js" name="app.js" />

</h:head>

<h:body id="body_id">

   <!-- *** -->
   <!-- TOP -->
   <!-- *** -->
   <h:panelGroup id="top_id" layout="block" styleClass="top">
      <ui:include src="/view/application/top.xhtml" />
   </h:panelGroup>

   <!-- CONTAINER -->
   <h:panelGroup id="container_id" layout="block" styleClass="container">

      <!-- **** -->
      <!-- LEFT -->
      <!-- **** -->
      <h:panelGroup layout="block" styleClass="left" rendered="#{layout.rendered['left']}">
         <!-- LEFT CONTAINER -->
         <h:panelGroup layout="block" styleClass="left-container">
            <h:panelGroup id="left_header_id" layout="block"
                rendered="#{layout.rendered['left.header']}" styleClass="left-header">
               <!-- LEFT HEADER -->
               <ui:insert name="left.header" />
            </h:panelGroup>
            <!-- LEFT BODY CONTAINER -->
            <h:panelGroup layout="block" styleClass="left-body-container">
               <h:panelGroup id="left_body_id" layout="block" styleClass="left-body">
                  <!-- LEFT BODY -->
                  <ui:insert name="left.body" />
               </h:panelGroup>
            </h:panelGroup>
            <h:panelGroup id="left_footer_id" layout="block" styleClass="left-footer">
               <!-- LEFT FOOTER -->
               <ui:insert name="left.footer" />
            </h:panelGroup>
         </h:panelGroup>
      </h:panelGroup>

      <!-- ******* -->
      <!-- CENTER -->
      <!-- ******* -->
      <h:panelGroup layout="block" styleClass="center">
         <!-- CENTER CONTAINER -->
         <h:panelGroup layout="block" styleClass="center-container">
```

```
60          <!-- CENTER HEADER -->
61          <h:panelGroup id="center_header_id" layout="block" styleClass="center-header"
                rendered="#{layout.rendered['center.header']}">
62             <!-- CENTER TOOLBAR -->
63             <h:form id="center_toolbar_form_id">
64                <p:toolbar id="center_toolbar_id" styleClass="center-toolbar"
65                   style="border-radius: 0px; border-left:0px; border-top:0px; border-right:0px">
66                   <p:toolbarGroup align="left">
67                      <!-- CENTER TOOLBAR LEFT -->
68                      <ui:insert name="center.toolbar.left" />
69                   </p:toolbarGroup>
70                   <p:toolbarGroup align="right">
71                      <!-- CENTER TOOLBAR RIGHT -->
72                      <ui:insert name="center.toolbar.right" />
73                   </p:toolbarGroup>
74                </p:toolbar>
75             </h:form>
76          </h:panelGroup>
77          <!-- CENTER DETAIL -->
78          <h:panelGroup id="center_detail_id" layout="block" styleClass="center-detail"
                rendered="#{layout.rendered['center.detail']}">
79             <ui:insert name="center.detail" />
80          </h:panelGroup>
81          <!-- CENTER BODY CONTAINER -->
82          <h:panelGroup layout="block" styleClass="center-body-container">
83             <!-- CENTER BODY -->
84             <h:panelGroup id="center_body_id" layout="block" styleClass="center-body">
85                <ui:insert name="center.body" />
86             </h:panelGroup>
87          </h:panelGroup>
88       </h:panelGroup>
89    </h:panelGroup>
90
91    <!-- ***** -->
92    <!-- RIGHT -->
93    <!-- ***** -->
94    <h:panelGroup layout="block" styleClass="right" rendered="#{layout.rendered['right']}">
95       <h:panelGroup id="right_header_id" layout="block" styleClass="right-header">
96          <!-- RIGHT HEADER -->
97          <ui:insert name="right.header" />
98       </h:panelGroup>
99       <hr style="border: 1px solid #ddd" />
100      <h:panelGroup id="right_body_id" layout="block" styleClass="right-body">
101         <!-- RIGHT BODY -->
102         <ui:insert name="right.body" />
103      </h:panelGroup>
104   </h:panelGroup>
105</h:panelGroup>
106
107<!-- DOCUMENT BODY -->
108<ui:insert name="document.body" />
109
110<!-- APPLICATION INFO -->
111<p:growl id="growl_id" />
112
113<!-- APPLICATION.ERROR -->
114<h:form id="error_form_id">
115   <p:dialog id="error_id" header="Aviso" closable="false" resizable="false"
          widgetVar="error_wvar" modal="true">
116      <h:panelGrid id="error_body_id" columns="2" style="max-width: 600px">
117         <h:messages />
118      </h:panelGrid>
119      <f:facet name="footer">
```

```
120             <p:commandButton value="Fechar" onclick="error_wvar.hide(); return false"
                    process="@this" />
121           </f:facet>
122         </p:dialog>
123     </h:form>
124
125 </h:body>
126 </html>
```

- ***Linha 8*** - Título da página *(Nome do Sistema)*.
- ***Linha 9*** - Arquivo com as definições de layout da aplicação. Como a definição das *regiões* do template.
- ***Linha 23*** - Definição a região de **"top"** da aplicação
- ***Linha 24*** - Inclui o trecho XHTML do topo.
- ***Linha 38*** - Define a região **"left.header"** do template.
- ***Linha 44*** - Define a região **"left.body"** do template.
- ***Linha 49*** - Define a região **"left.footer"** do template.
- ***Linha 68*** - Define a região **"center.toolbar.left"** do template.
- ***Linha 72*** - Define a região **"center.toolbar.right"** do template.
- ***Linha 79*** - Define a região **"center.detail"** do template.
- ***Linha 85*** - Define a região **"center.body"** do template.
- ***Linha 97*** - Define a região **"right.header"** do template.
- ***Linha 102*** - Define a região **"right.body"** do template.
- ***Linha 108*** - define a região **"document.body"** do template. Esta região deverá ser usada nos "*indexs*" quando existir a necessidade de se incluir algum componente diretamente no *body*.
- ***Linha 111*** - Growl padrão para informções do sistema.
- ***Linha 114 a 123*** - Trecho XHTML do dialog de erro padrão do sistema.

6 Conceito de Página

De maneira direta, uma página é definida como **todo o conteúdo xhtml que se mantém sob uma mesma URL** (mantém o scopo de visão do JSF), ou seja, todos os componentes do Primefaces (dialogs, datatables, datalists etc.) utilizados para montar um index (xhtml que *extende* o template padrão) pertencem a uma página.

De posse deste conceito, é proposto que a organização do código java e xhtml necessário para montar uma página seja agrupado de forma coerente.

6.1 Organização das Páginas

As seções a seguir mostram a forma de organização dos códigos Java e XHTML. Para tanto, tomemos como exemplo a entidade *Usuário* que conceitualmente representa uma página do sistema. Para esta entidade iremos criar uma página onde irá conter a listagem de usuários, o dialog de edição de usuários e qualquer outro elemento que faça parte da estrurua que forma a página.

6.1.1 Organização da Página de Usuários

Na estrutura de código Java a forma mais adequada para organizarmos os códigos java é através de *pacotes*. De acordo com as funcionalidades elencadas na seção de introdução deste documento daremos o nome para o pacote da **Página de Usuários** de *usuarioPage*. Em seguida criamos uma classe java chamada *UsuarioPG.java* que irá gerenciar todos os aspectos referente a página e ,além disso, iremos trazer para este novo pacote(*usuarioPage*) os managed beans correpondentes.

Na parte XHTML é criado um diretório com o mesmo nome do pacote em java (usuarioPage) e o arquivo que gerencia a página é chamado de index.xhtml[4]

Como resultado teremos a seguinte estrutura:

[4]Este é o arquivo que extende e possui todos os "*defines*" do template padrão main.xhtml.

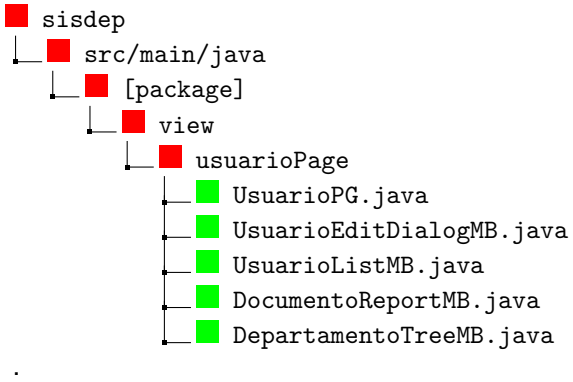


Figure 7: Estrutura de de pacotes.

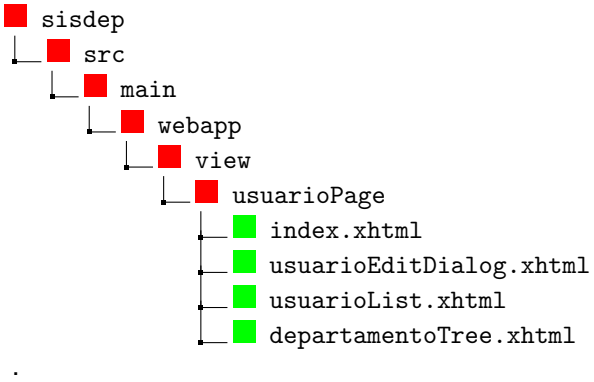


Figure 8: Estrutura de diretórios.

Com isso, qualquer *managed bean* pertencente a página de usuários estará contido dentro do pacote **view.usuarioPage** e qualquer componente XHTML estará contido no diretório /**view**/**usuarioPage**

6.1.2 Organização da Página de Gerenciamento de Departamentos

Utilizado o mesmo raciocínio do seção 6.1.1 devemos criar um pacote para a guardar o conteúdo da **Página de Gerenciamento de Departamentos**. Como o página é uma página de gerenciamento o nome do pacote resultante será: ***departamentoMngtPage***

Na parte XHTML é criado um diretório com o mesmo nome do pacote em java (departamentoMngtPage) e o arquivo que gerencia a página é chamado de index.xhtml.

Como resultado teremos a seguinte estrutura:

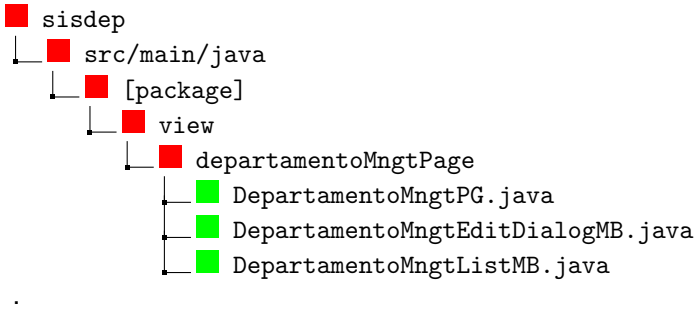


Figure 9: Estrutura de de pacotes.

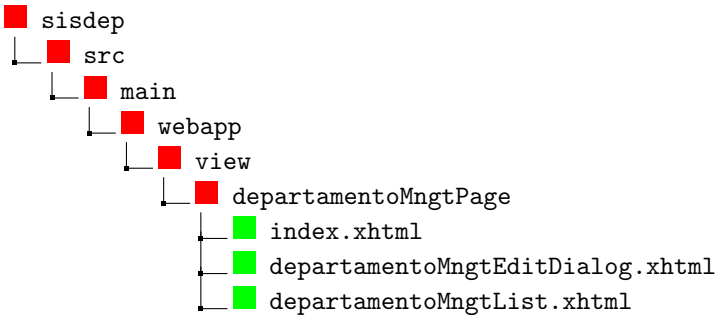


Figure 10: Estrutura de diretórios.

Como podemos perceber, apesar de utilizarmos o nome da entidade para criarmos o nome do pacote na seção 6.1.1 (*usuarioPage*) aqui neste exemplo definimos o nome como *departamentoMngtPage* que representa o conceito lógico (Gerenciar Departamento), ou a finalizade da página em si, que dá o nome do pacote correspondente.

6.2 Código Java da Página de Usuário (UsuarioPG.java)

Conforme dito em seções anteriores, umas das vantagens da utilização do Template Padrão é a flexibilidade de configuração das regiões de acordo com o layout definido pela equipe de design. A forma como esta configuração é feita é ilustrada pela seguinte alteração no código da página a seguir:

```
1  // UsuarioPG.java
2  package view.usuarioPage;
3
4  import javax.inject.Inject;
5
6  import br.gov.frameworkdemoiselle.stereotype.ViewController;
7  import br.jus.tre_pa.jbase.jsf.layout.DocumentPageBean;
8  import br.jus.tre_pa.jbase.jsf.layout.UILayout;
9
10 @ViewController
11 public class UsuarioPG implements DocumentPageBean {
12
13    @Inject
14    private Logger log;
```

```
15
16    @Inject
17    private UILayout layout;
18
19    @Override
20    public String load() {
21       layout.setRendered("left", true);
22       layout.setRendered("right", false);
23       layout.setRendered("center.detail", true);
24       return null;
25    }
26
27 }
```

- ***Linha 17*** - Injeção da classe *UILayout* que é responsável por manipular o template padrão.
- ***Linha 21*** - Método da classe *UILayout* que define se uma região do template padrão é renderizada ou não. No exemplo desta linha a região **"left"** do template é setada para ser renderizada.
- ***Linha 22*** - Define que a região **"right"** do template padrão **não** é renderizada.
- ***Linha 23*** - Define que a região **"center.detail"** do template padrão é renderizada.

Como resultado do código acima temos o seguinte layout:

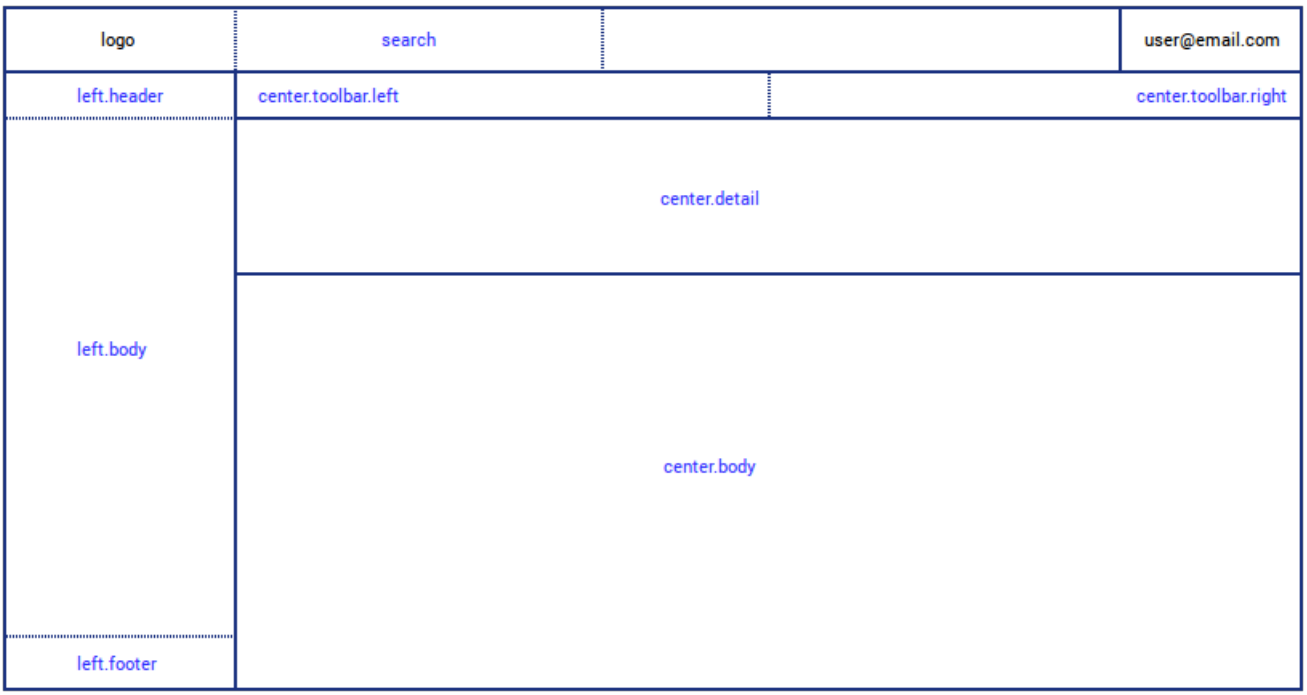


Figure 11: Layout da Página de Usuários.

6.3 Código XHTML da Página de Usuários (usuarioPage/index.xhtml)

Abaixo temos um exemplo de um código xhtml de uma página de *index* vazia. No momento esta possui apenas as definições(*ui:define*) das regiões definidas(*ui:insert*) no template padrão(*main.xhtml*). Caso alguma seção(*ui:define*) não seja utilizada, esta pode ser deletada do arquivo de *index.xhtml* sem maiores problemas.

```
1  <ui:composition xmlns="http://www.w3.org/1999/xhtml" xmlns:f="http://java.sun.com/jsf/core"
       xmlns:p="http://primefaces.org/ui"
2      xmlns:h="http://java.sun.com/jsf/html" xmlns:ui="http://java.sun.com/jsf/facelets"
           template="/template/main.xhtml">
3
4
5      <!-- LEFT BODY -->
6      <ui:define name="left.body">
7         <ui:include src="/view/usuarioPage/departamentoTree.xhtml" />
8      </ui:define>
9
10     <!-- LEFT FOOTER -->
11     <ui:define name="left.footer">
12     </ui:define>
13
14     <!-- CENTER TOOLBAR LEFT -->
15     <ui:define name="center.toolbar.left">
16     </ui:define>
17
18     <!-- CENTER TOOLBAR RIGHT -->
19     <ui:define name="center.toolbar.right">
20        <p:button value="Imprimir Relatorio" action="documentoReportMB.print" ajax="false">
21     </ui:define>
22
23     <!-- CENTER DETAIL -->
24     <ui:define name="center.detail">
25     </ui:define>
26
27     <!-- CENTER BODY -->
28     <ui:define name="center.body">
29        <ui:include src="/view/usuarioPage/usuarioList.xhtml" />
30     </ui:define>
31
32     <!-- DOCUMENT BODY -->
33     <ui:define name="document.body">
34        <ui:include src="/view/usuarioPage/usuarioEditDialog.xhtml" />
35     </ui:define>
36
37 </ui:composition>
```

- ***Linha 4*** - Atributo *template* da tag ***ui:composition*** que define qual atributo o xhtml irá utilizar.
- ***Linha 7*** - Definição da região **"left.header"** do template.
- ***Linha 11*** - Definição da região **"left.body"** do template. É nesta região que usualmente se encontra a listagem em árvore (*p:tree*) da página.
- ***Linha 15*** - Define a região **"left.footer"** do template. É nesta região que usualmente se encontra a totalização de itens que se encontram na listagem em árvore.
- ***Linha 19*** - Define a região **"center.toolbar.left"** do template. É nesta região que é inserido os *buttons* no lado esquerdo da barra de ferramentas.
- ***Linha 23*** - Define a região **"center.toolbar.right"** do template. É nesta região que é inserido os *buttons* no lado direito da barra de ferramentas.
- ***Linha 27*** - Define a região **"center.detail"** do template.
- ***Linha 31*** - Define a região **"center.body"** do template.
- ***Linha 35*** - Define a região **"document.body"** do template.

6.4 Código Java da Página de Genreciamento de Departametos (DepartamentoMngtPG.java)

```java
// DepartamentoMngtPG.java
package view.departamentoMngtPage;

import javax.inject.Inject;

import br.gov.frameworkdemoiselle.stereotype.ViewController;
import br.jus.tre_pa.jbase.jsf.layout.DocumentPageBean;
import br.jus.tre_pa.jbase.jsf.layout.UILayout;

@ViewController
public class DepartamentoMngtPG implements DocumentPageBean {

   @Inject
   private Logger log;

   @Inject
   private UILayout layout;

   @Override
   public String load() {
      layout.setRendered("left", false);
      layout.setRendered("right", false);
      layout.setRendered("center.detail", false);
      return null;
   }

}
```

Como resultado do código acima temos o seguinte layout:

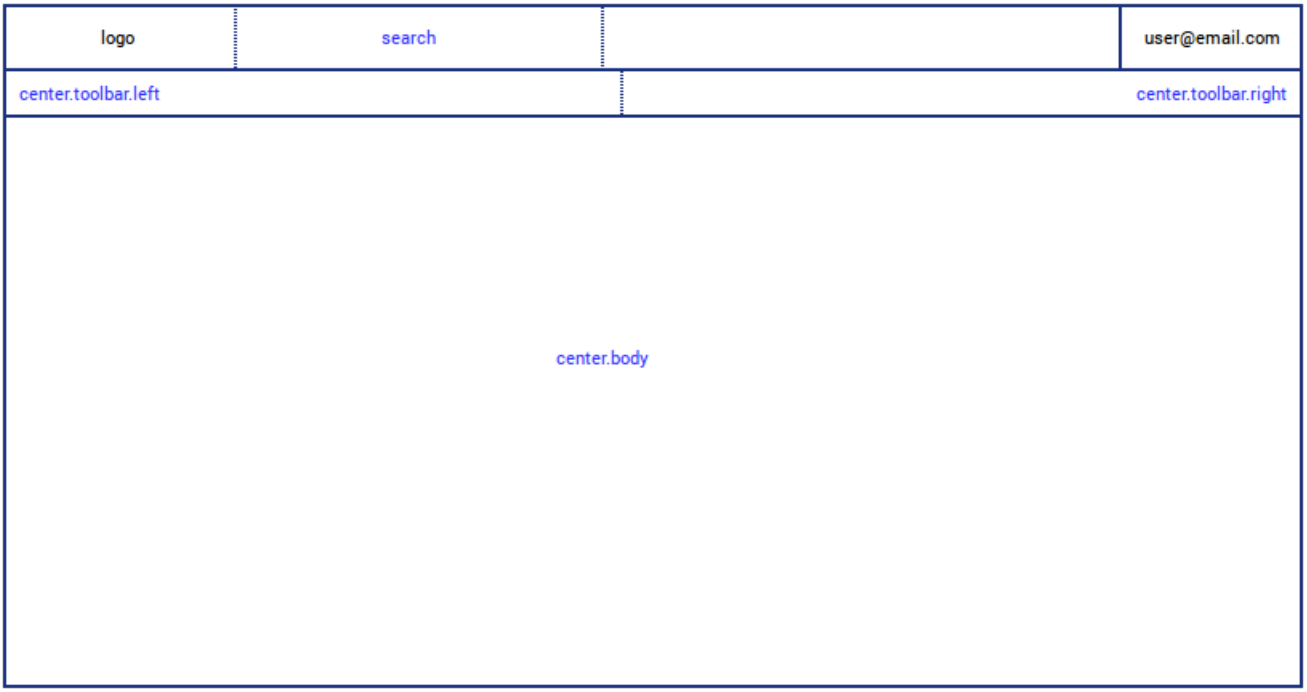


Figure 12: Layout da Página de Gerenciamento de Departamentos.

6.5 Código XHTML da Página de Genreciamento de Departametos (departamentoMngtPage/index.xhtml)

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml" xmlns:f="http://java.sun.com/jsf/core"
    xmlns:p="http://primefaces.org/ui"
   xmlns:h="http://java.sun.com/jsf/html" xmlns:ui="http://java.sun.com/jsf/facelets"
       template="/template/main.xhtml">


   <!-- CENTER TOOLBAR LEFT -->
   <ui:define name="center.toolbar.left">
   </ui:define>

   <!-- CENTER TOOLBAR RIGHT -->
   <ui:define name="center.toolbar.right">
   </ui:define>

   <!-- CENTER BODY -->
   <ui:define name="center.body">
      <ui:include src="/view/departamentoMngtPage/departamentoMngtList.xhtml" />
   </ui:define>

   <!-- DOCUMENT BODY -->
   <ui:define name="document.body">
      <ui:include src="/view/departamentoMngtPage/departamentoMngtEditDialog.xhtml" />
   </ui:define>

</ui:composition>
```

7 Reescrevendo URLs com Pretty-Faces

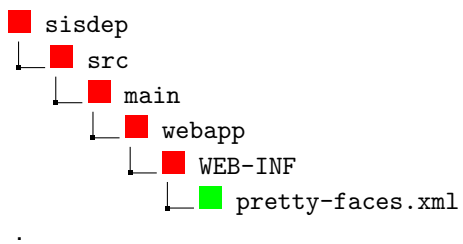
O projeto *pretty-faces* apesar de possuir o ”faces” na sua nomenclatura, não possui nenhum componente visual para utilizarmos em projetos JSF. Seu objetivo principal é reescrever as URLs do JSF tornado-as ”amigáveis” para o usuário final sem a necessidade de renomearmos ou reestruturarmos os diretórios onde se encontram os XHTMLs da aplicação.

7.1 Instalação

Para se instalar o Pretty-Faces é apenas necssário a inclusão da respectiva dependência no *pom.xml* conforme abaixo:

```
<dependency>
   <groupId>com.ocpsoft</groupId>
   <artifactId>prettyfaces-jsf2</artifactId>
   <version>3.3.2</version>
</dependency>
```

Após a instalação, todo o processo de mapeamento de URLs do pretty-faces estará concentrado no arquivo ***pretty-faces.xml*** localizado na estrutura de diretório conforme a seguir:



.

7.2 Mapeamento de Páginas no Pretty-Faces

O trecho de código abaixo exibe a forma de como é realizado o mapeamento dentro do arquivo ***pretty-faces.xml***:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <pretty-config xmlns="http://ocpsoft.com/prettyfaces/3.3.2"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://ocpsoft.com/prettyfaces/3.3.2
          http://ocpsoft.com/xml/ns/prettyfaces/ocpsoft-pretty-faces-3.3.2.xsd">
5
6     <!-- USUARIO PAGE -->
7     <url-mapping id="usuario-page">
8        <pattern value="/usuarios"></pattern>
9        <view-id value="/view/usuarioPage/index.jsf" />
10       <action>#{usuarioPG.load}</action>
11    </url-mapping>
12
13    <!-- GERECIAMENTO DEPARTAMENTO PAGE -->
14    <url-mapping id="departament-mngt-page">
15       <pattern value="/gereciamento/departamento"></pattern>
16       <view-id value="/view/departamentoMngtPage/index.jsf" />
17       <action>#{departamentoMngtPG.load}</action>
18    </url-mapping>
19
20 </pretty-config>
```

- ***Linha 5*** - ID do mapeamento do pretty-faces. Deve ser único no arquivo.
- ***Linha 6*** - Define o padrão da URL que será exposta para o usuário final. É esta URL que deverá ser indicada no navegador para acessar a página correspondente.
- ***Linha 7*** - Caminho a partir do diretório *src/main/webapp* do arquivo XHTML correspondente a página.
- ***Linha 8*** - É nesta linha que a chamada do metodo load da página correspondente, ou seja, caso a URL digitada no navegador seja *http://sisdep/usuarios* o método *load* é invocado.
- ***Linha 14 a 18*** - Definições para a Página de Gerenciamento de Departamentos

A cada nova página do sistema um novo bloco *"url-mapping"* deverá ser definido com os dados da respectiva página.

8 Conclusão

Com a abordagem explicada anteriormente podemos elencar os seguintes ganhos:

1. Criação de uma relação consistente e direta entre o código Java e XHTML facilitando a visualização em código do conteúdo de cada página.
2. Flexibilidade na criação de templates.
3. Utilização consistente do escope de visão (*@ViewScope*).
4. Melhor estruturação do código fonte.