

# I

# Desenvolvimento de Aplicações Web

JSF - Frameworks de Componentes

Tassio Sirqueira – 2019/02

# | Componentes | Introdução

- ❑ O JavaServer Faces (**JSF**) é um framework **baseado em componentes**.
  - ❑ Toda interface com o usuário e integração entre diferentes partes da camada de visão é realizada por meio de componentes.
- ❑ Existem mecanismos de extensão que permitem que desenvolvedores criem seus próprios componentes.
  - ❑ Reforçando o reaproveitamento de interfaces.
  - ❑ Facilitando o desenvolvimento de interfaces complexas.



# | Componentes | Introdução

- ❑ A comunidade de JavaServer Faces (**JSF**) incentiva que terceiros desenvolvam e publiquem seus próprios componentes.
- ❑ Nesse contexto, existem alguns frameworks que consistem em bibliotecas de componentes para o desenvolvimento de aplicações JSF.



# | Componentes | Introdução | RichFaces

- ❑ Surgiu pela Americana Exadel em 2006.
  - ❑ Em 2007 a JBoss, em um acordo, adicionou seus próprios componentes da Ajax4Jsf à biblioteca e assumiu seu desenvolvimento.
- ❑ Suporta validação client-side baseada no bean-validation.
  - ❑ Usa JQuery internamente.
- ❑ Pequena, possuindo 80+ componentes.
  - ❑ Possui apenas 3 dependências e está organizada em 2 namespaces (componentes independentes e os dependentes de Ajax.)
- ❑ Varias inovações incorporadas ao JSF 2.
- ❑ **Abandonado em 2016.**



**RichFaces**

# | Componentes | Introdução | PrimeFaces

- ❑ Lançado em 2008 pela empresa Turca PrimeTechnology.
- ❑ Inclui menos inovações em relação ao processamento no servidor, focada na parte visual, com +150 componentes.
  - ❑ Número maior de componentes.
  - ❑ Não possui dependências externas.
- ❑ Possui um único namespace para todos os componentes.
  - ❑ Utiliza JQuery e JQuery UI internamente.
  - ❑ Skins baseadas no *Theme Roller* do JQueryUI
- ❑ Gratuito.
  - ❑ Oferece suporte enterprise pago.
  - ❑ Temas pagos.
- ❑ Possui versões para outras plataformas e linguagens.



# | Componentes | Introdução | ICEFaces

- ❑ Lançado em 2008 pela empresa canadense ICESoft.
- ❑ +100 componentes.
- ❑ Inclui anotações especiais que adicionam possibilidade de controle do ciclo de vida e de escopos da aplicação.
  - ❑ Baseado no JQuery e no Yahoo UI.
- ❑ Gratuito
  - ❑ Duas bibliotecas de componentes gratuitos e uma delas paga.
  - ❑ Suporte pago.



# | Componentes | Introdução | Oracle ADF

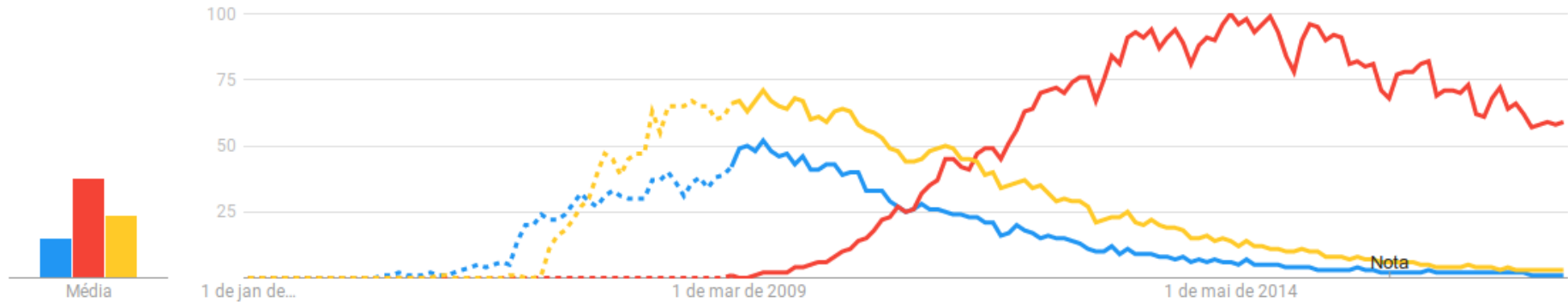
- ❑ O Framework é de 1999, com partes do framework para objetos de negócio. Com o JSF, o Oracle Application Development Framework surge em 2006 com +100 componentes.
- ❑ Possui ferramenta RAD (prototipação rápida) integrada a outras ferramentas da Oracle e criação de interfaces com WYSIWYG (live preview)
- ❑ Licença Comercial
  - ❑ Gratuito para quem usa o Servidor de Aplicação da Oracle.
  - ❑ Componentes gratuitos (ADF Essenciais) doados à Apache.

**ORACLE**®  

---

**ADF**

# Componentes | Introdução | Comparação



- ✓ Primeiro com suporte a JSF 2
- ✓ Ampla documentação e base de usuários.
- ✓ Customizável e com suporte a temas.
- ✓ Maior número de componentes.



# | Componentes | Introdução

## ❑ Importante!!!

- ❑ Não adicione mais de uma biblioteca de componentes à mesma aplicação!
  - ❑ Conflito de versão e namespace no javaScript.
  - ❑ Conflito de CSS gerando erros de leiaute.
  - ❑ Mudanças no funcionamento do JSF 2.
  - ❑ Bugs não testados.



# | PrimeFaces | Instalação



- ❑ O PrimeFaces possui muitos componentes e amplo suporte a AJAX, estendendo o suporte do próprio JSF.
- ❑ Por possuir apenas um JAR, é facilmente adicionado ao projeto.

`pom.xml`

```
<dependencies>
```

```
...
```

```
    <dependency>
```

```
        <groupId>org.primefaces</groupId>
```

```
        <artifactId>primefaces</artifactId>
```

```
        <version>6.1</version>
```

```
    </dependency>
```

```
...
```

```
</dependencies>
```

# | PrimeFaces | Getting Started



- ❑ Os componentes do primefaces estão no namespace **`http://primefaces.org/ui`**.

example.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">
  <h:head></h:head>
  <h:body>
    <p:button value="Clique-me!" />
  </h:body>
</html>
```

# | PrimeFaces | Configuração



- ❑ Nenhuma configuração é obrigatória, porém diferentes configurações estão disponíveis via parâmetro de contexto. [Temas](#)

web.xml

```
<context-param>
  <param-name>primefaces.THEME</param-name>
  <param-value>omega</param-value>
</context-param>
```

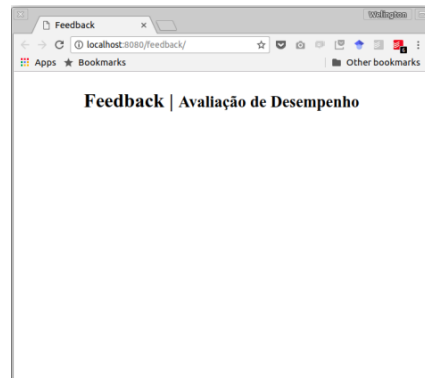
É possível importar os temas independentemente ou todos eles como dependências do Maven:

```
<dependency>
  <groupId>org.primefaces.themes</groupId>
  <artifactId>all-themes</artifactId>
  <version>1.0.10</version>
</dependency>
```

# | PrimeFaces | Estudo de Caso



- ❑ Vamos criar uma aplicação utilizando a biblioteca de Componentes Primefaces.
- ❑ A aplicação será uma ferramenta de feedbacks corporativos, onde pessoas podem ser cadastradas e receber feedbacks anônimos de colegas de trabalho.
- ❑ No fim relatórios sobre cada pessoa estarão disponíveis.
- ❑ Utilize como ponto de partida:
  - ❑ Este [projeto](#).
  - ❑ Este [script](#).
- ❑ Depois de rodar o script e rodar o projeto você deve visualizar a tela ao lado.



# | PrimeFaces | Estudo de Caso | Passo 1



- ❑ Incluir o PrimeFaces no projeto.

pom.xml

```
<dependencies>
```

```
...
```

```
    <dependency>
```

```
        <groupId>org.primefaces</groupId>
```

```
        <artifactId>primefaces</artifactId>
```

```
        <version>6.1</version>
```

```
    </dependency>
```

```
...
```

```
</dependencies>
```

Sempre depois de alterar o POM, limpe e reconstrua o projeto.

# | PrimeFaces | Estudo de Caso | Passo 2



- ❑ Inclua o namespace do PrimeFaces no index.xhtml

index.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">
  <h:head></h:head>
  <h:body>
    <p:button value="Clique-me!" />
  </h:body>
</html>
```

Sempre depois de alterar o POM, limpe e reconstrua o projeto.

# | PrimeFaces | Estudo de Caso | Passo 3



- ❑ Verifique alguns componentes do [PrimeFaces](http://primefaces.org)

index.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">
  <h:head></h:head>
  <h:body>
    <p:calendar />
    <p:spinner/>
    <p:button value="Clique-me!" />
  </h:body>
</html>
```

Sempre depois de alterar o POM, limpe e reconstrua o projeto.



# | PrimeFaces | Estudo de Caso | Passo 4



❏ Tela de Login (index.xhtml).

```
<?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"
      xmlns:p="http://primefaces.org/ui">
  <h:head>
    <title>Feedback</title>
  </h:head>
  <h:body style="display: flex">
    <div style="margin: auto; text-align: center">
      <h:form id="login-form">
        <p:panel header="Entrar" footer="Sistema de Feedback" style="width: fit-content">
          <p:messages autoUpdate="true" showDetail="true" />
          <h:panelGrid columns="2" cellpadding="10">
            <p:graphicImage url="/resources/img/logo.jpg" />
            <h:panelGrid columns="2" cellpadding="5">
              <p:outputLabel value="Login" />
              <p:inputText id="login" />
              <p:outputLabel value="Senha" />
              <p:password id="senha" />
              &nbsp;
              <p:commandButton value="Entrar" id="commandButton-acessar" icon="ui-icon-gear" />
            </h:panelGrid>
          </h:panelGrid>
        </p:panel>
      </h:form>
    </div>
  </h:body>
</html>
```

# | PrimeFaces | Estudo de Caso | Passo 5



❑ Bean para a Tela de Login (LoginBean).

```
@Named
@ViewScoped
public class LoginBean implements Serializable {

    private static final long serialVersionUID = 1L;

    @Inject
    private UsuarioLogado usuarioLogado;

    @Inject
    UsuarioFacade facade;

    private String login, senha;

    // Getters e Setters ...

    public String login(){
        Optional<Usuario> usuario = facade.findByCredenciais(login, senha);
        if (usuario.isPresent()) {
            this.usuarioLogado.setUsuario(usuario.get());
            return "principal";
        }

        FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, "Falha ao Autenticar", "Usuário ou senha inválido!");
        FacesContext.getCurrentInstance().addMessage(null, msg);
        return null;
    }
}
```

# | PrimeFaces | Estudo de Caso | Passo 6



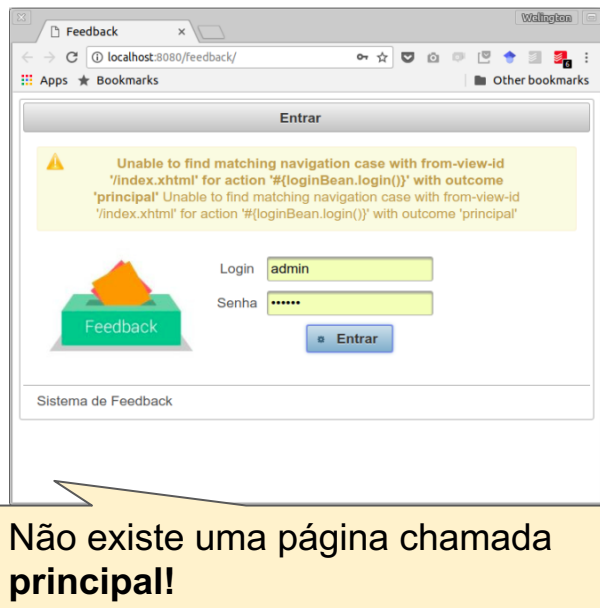
## ❏ Associar a view ao controller.

```
<?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"
      xmlns:p="http://primefaces.org/ui">
  <h:head>
    <title>Feedback</title>
  </h:head>
  <h:body style="display: flex">
    <div style="margin: auto; text-align: center">
      <h:form id="login-form">
        <p:panel header="Entrar" footer="Sistema de Feedback" style="width: fit-content">
          <p:messages autoUpdate="true" showDetail="true" />
          <h:panelGrid columns="2" cellpadding="10">
            <p:graphicImage url="/resources/img/logo.jpg" />
            <h:panelGrid columns="2" cellpadding="5">
              <p:outputLabel value="Login"/>
              <p:inputText id="login" value="#{loginBean.login}"/>
              <p:outputLabel value="Senha"/>
              <p:password id="senha" value="#{loginBean.senha}"/>
              &nbsp;
              <p:commandButton value="Entrar" id="commandButton-acessar" icon="ui-icon-gear" action="#{loginBean.login()}" />
            </h:panelGrid>
          </h:panelGrid>
        </p:panel>
      </h:form>
    </div>
  </h:body>
</html>
```

# PrimeFaces | Estudo de Caso | Passo 6



- ❑ Associar a **view** ao **controller**.
- ❑ Ao executar a seguinte mensagem será exibida:



Não existe uma página chamada **principal**!

```
Info: Running on PrimeFaces 6.1
Info: Loading application [feedback] at [/feedback]
Info: feedback was successfully deployed in 804 milliseconds.
Fine: SELECT ID, LOGIN, PERFIL, SENHA FROM USUARIO WHERE ((LOGIN = ?) AND
(SENHA = ?))
      bind => [admin, 4Qrc0Um6Wau+VuBX8g+IPg==]
Warning: JSF1064: Unable to find or serve resource, /principal.xhtml.
```

A exibição do SQL executado está habilitada no persistence.xml !!!

# | PrimeFaces | Estudo de Caso | Passo 7



- ❑ Renomeie a página index.xhtml para login.xhtml.
- ❑ Altere a página inicial da aplicação para login.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ... >
    ...

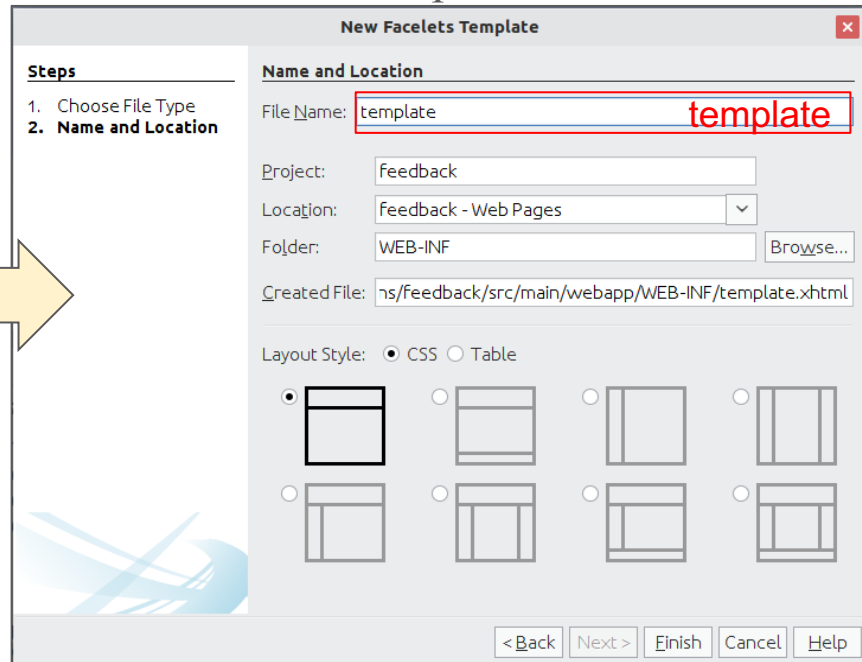
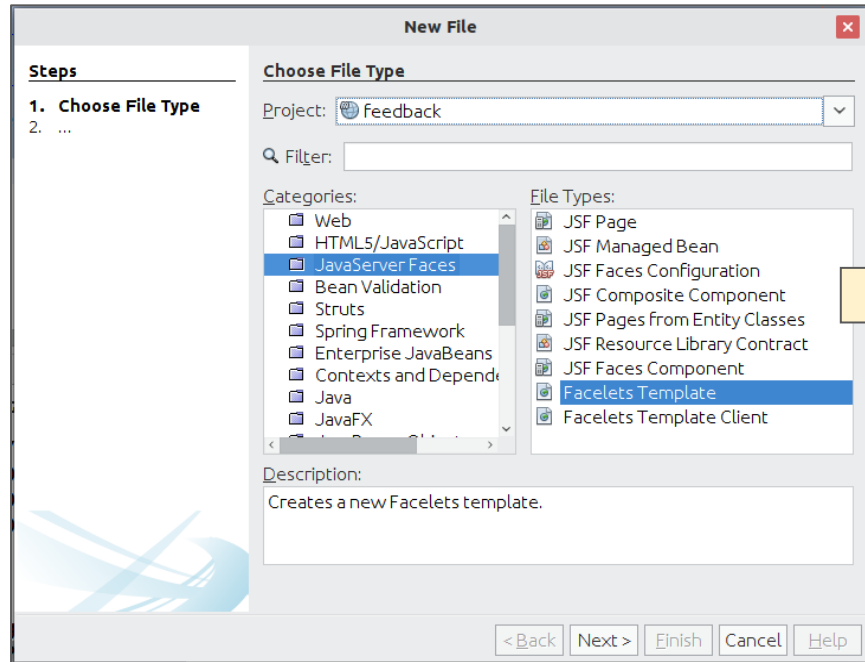
    <welcome-file-list>
        <welcome-file>faces/login.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

# | PrimeFaces | Estudo de Caso | Passo 8



❑ Vamos criar um leiaute padrão para as páginas da Aplicação.

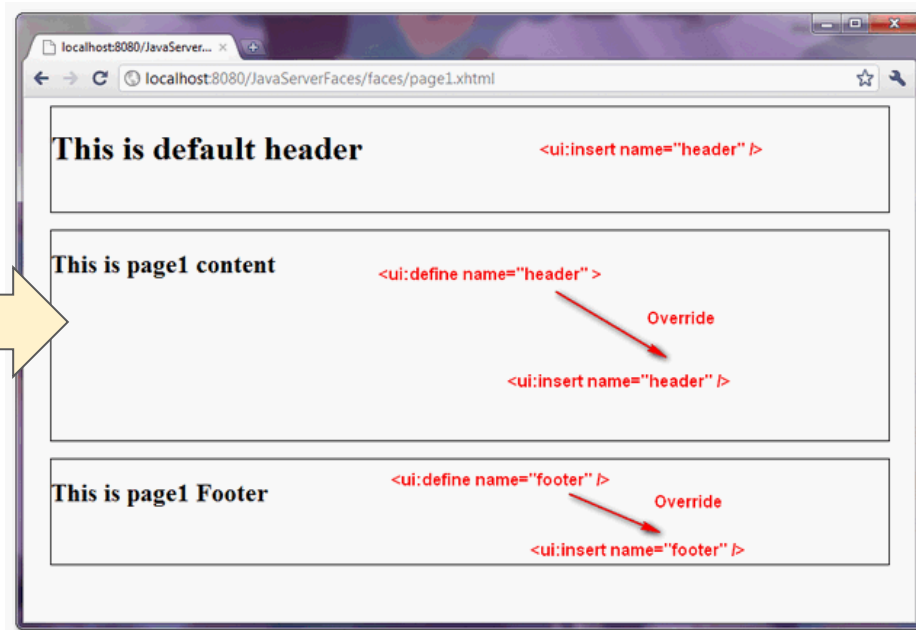
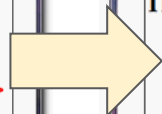
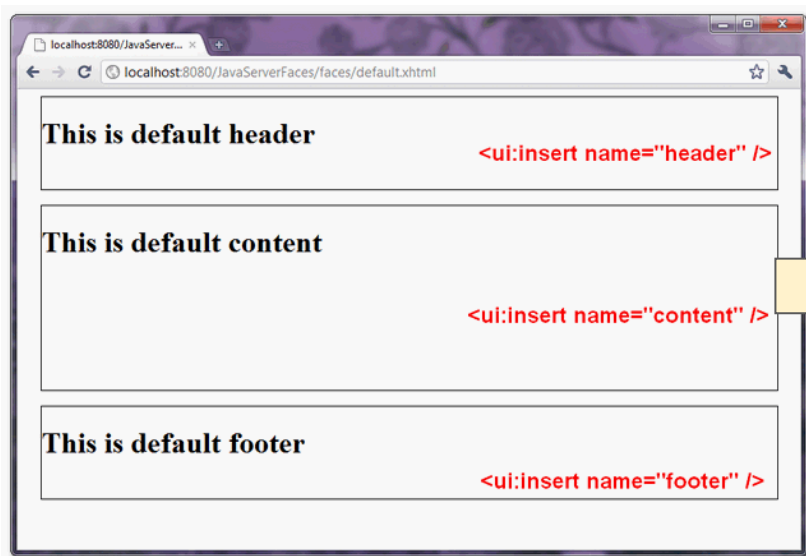
❑ File > New > Other... > JavaServer Faces > Facelets Template



# PrimeFaces | Estudo de Caso | Passo 8



- ❑ É possível compor as telas através do template criado.



# | PrimeFaces | Estudo de Caso | Passo 9

❑ Primeiro vamos adicionar uma área para permitir um título customizado por página.

```
<h:head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>
    <ui:insert name="title">
      Feedback | Sistema de Avaliação de Desempenho
    </ui:insert>
  </title>
</h:head>
```

Páginas 'filhas'  
podem sobrescrever  
esse valor.



# | PrimeFaces | Estudo de Caso | Passo 10



- ❑ Vamos definir um leiaute básico dividido em regiões para a aplicação.

```
<h:body>
```

O layout define uma área que pode ser dividida em regiões.  
Norte, Sul, Leste, Oeste e Centro.

```
<p:layout style="width: 80%; margin: 0 auto;">  
  <p:layoutUnit position="west" size="256">
```

No **Oeste** vamos Inserir um Menu padrão  
para toda a aplicação.

```
  </p:layoutUnit>
```

```
  <p:layoutUnit position="center">  
    <ui:insert name="content"/>
```

No **Centro** vamos deixar uma área  
para que a página filha insira conteúdo.

```
  </p:layoutUnit>
```

```
</p:layout>
```

```
</h:body>
```

# | PrimeFaces | Estudo de Caso | Passo 10

❏ Vamos adicionar um menu para a navegação entre páginas.

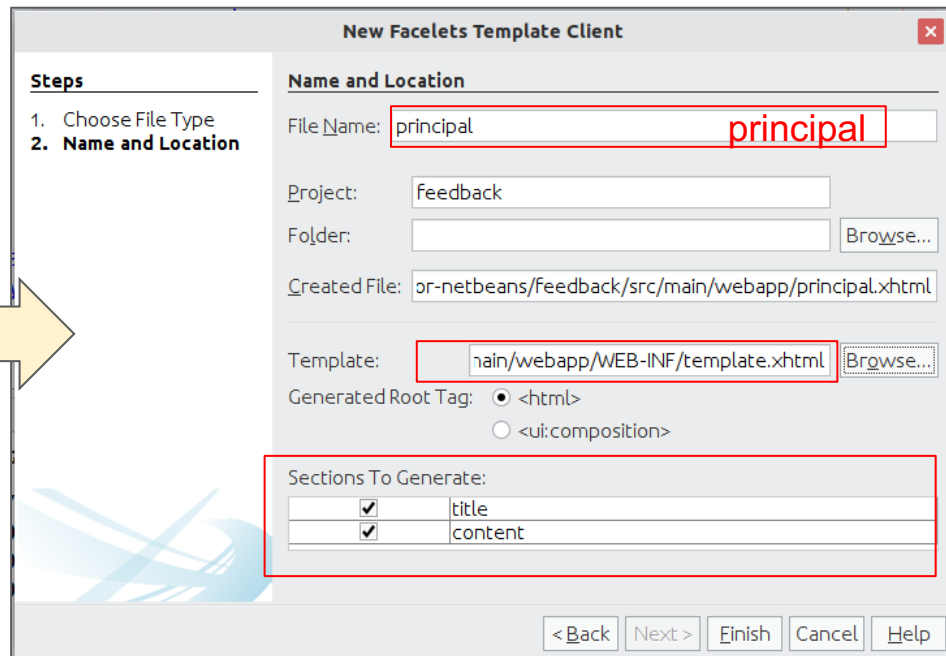
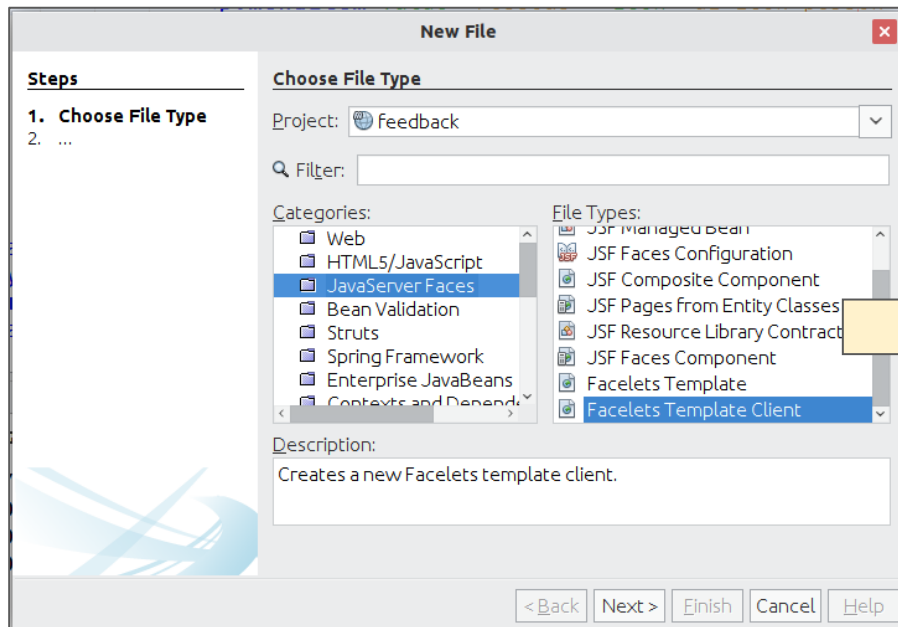
```
<p:layout style="width: 80%; margin: 0 auto;">
  <p:layoutUnit position="west" size="256">
    <div style="width: fit-content; margin: 0 auto; text-align: center;">
      <p:graphicImage url="/resources/img/logo.jpg" />
      <h:form>
        <p:menu>
          <p:submenu label="Cadastros">
            <p:menuitem value="Pessoas" icon="ui-icon-person" />
            <p:menuitem value="Avaliações" icon="ui-icon-clipboard" />
          </p:submenu>
          <p:submenu label="Relatórios">
            <p:menuitem value="Usuários por Cargos" icon="ui-icon-document" />
          </p:submenu>
        </p:menu>
      </h:form>
    </div>
  </p:layoutUnit>
  <p:layoutUnit position="center">
    <ui:insert name="content"/>
  </p:layoutUnit>
</p:layout>
```

Logo + Menu

# | PrimeFaces | Estudo de Caso | Passo 11



- ❑ Vamos criar uma página principal que de acordo com o leiaute criado.
- ❑ File > New > Other... > JavaServer Faces > Facelets Template Client



# PrimeFaces | Estudo de Caso | Passo 11

## ❏ Página principal.xhtml

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

```
    <ui:composition template="./WEB-INF/template.xhtml">
```

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

```
        title
```

Área onde podemos sobrescrever o título para a página.

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

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

```
        content
```

Área onde podemos sobrescrever o conteúdo da página.

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

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

```
  </body>
```

```
</html>
```

# | PrimeFaces | Estudo de Caso | Passo 12

## ❏ Personalizando a `principal.xhtml`

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

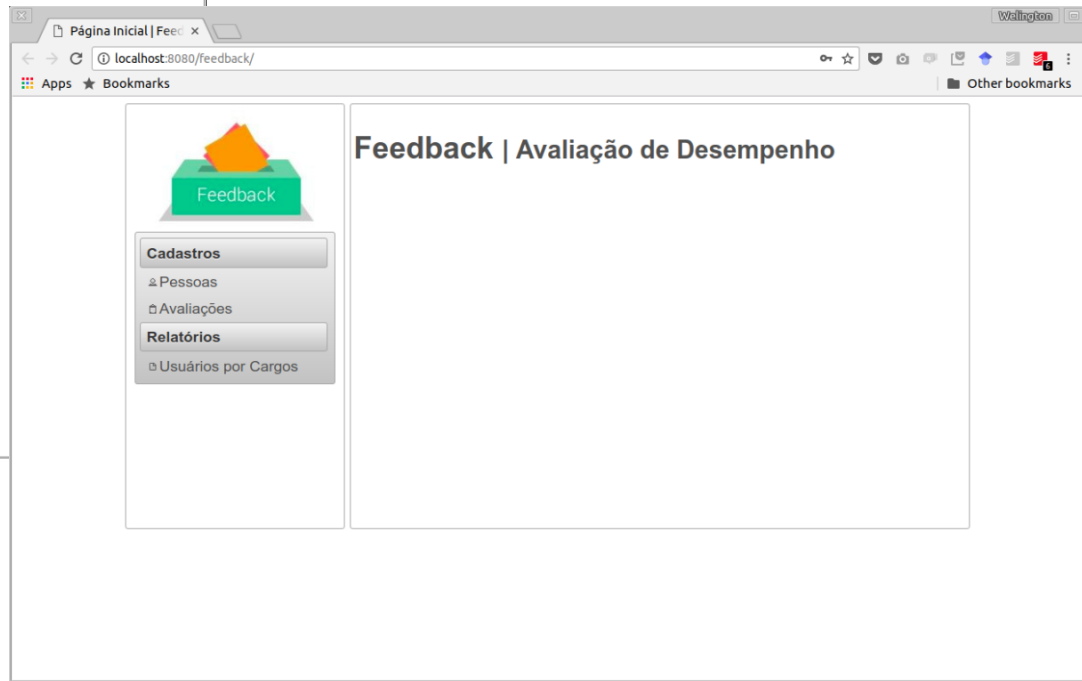
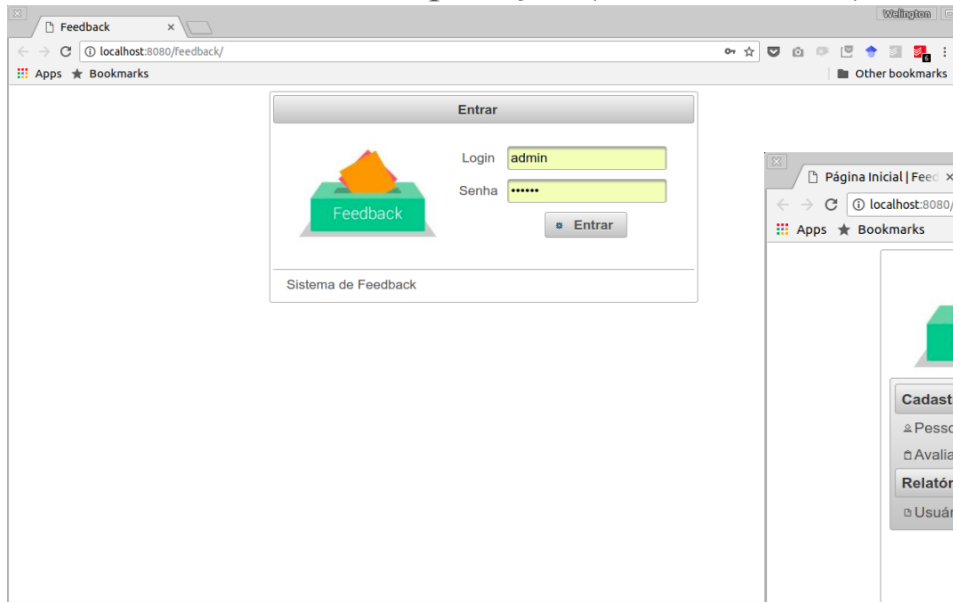
    <ui:composition template="./WEB-INF/template.xhtml">
      <ui:define name="title">
        Página Inicial | Feedback
      </ui:define>

      <ui:define name="content">
        <h1>Feedback <small>| Avaliação de Desempenho </small></h1>
      </ui:define>

    </ui:composition>
  </body>
</html>
```

# PrimeFaces | Estudo de Caso | Passo 13

📄 Vamos testar a aplicação (admin, 123456)



# | PrimeFaces | Estudo de Caso | Passo 14

- ❑ Vamos testar outros estilos de componentes, para isso vamos acrescentar o repositório do PrimeFaces ao Maven.

```
pom.xml
<dependencies>
...
</dependencies>
<repositories>
  <repository>
    <id>prime-repo</id>
    <name>PrimeFaces Maven Repository</name>
    <url>http://repository.primefaces.org</url>
    <layout>default</layout>
  </repository>
</repositories>
...
```

Sempre depois de alterar o POM, limpe e reconstrua o projeto.

# | PrimeFaces | Estudo de Caso | Passo 14

❏ Em seguida vamos adicionar os temas ao projeto.

```
pom.xml
<dependencies>
...
    <dependency>
        <groupId>org.primefaces.themes</groupId>
        <artifactId>all-themes</artifactId>
        <version>1.0.10</version>
    </dependency>
...
</dependencies>
<repositories>
...
</repositories>
...
```

Sempre depois de alterar o POM, limpe e reconstrua o projeto.



# | PrimeFaces | Estudo de Caso | Passo 14



❑ E então configurar o projeto para o tema desejado.

`web.xml`

```
<web-app ... >
    ...
    <context-param>
        <param-name>primefaces.THEME</param-name>
        <param-value>bootstrap</param-value>
    </context-param>
    ...
</web-app>
```

# | PrimeFaces | Estudo de Caso | Passo 15



- ❑ Vamos criar um cadastro de pessoas, para isso vamos criar outra página.
  - ❑ Crie uma pasta chamada pessoas dentro de 'Web pages'.
  - ❑ File > New > Other... > JavaServer Faces > Facelets Template Client
    - ❑ Chame a página de 'listar'
    - ❑ 'Herde' do template **template.xhtml**

The screenshot illustrates the process of creating a new Facelets Template Client in the PrimeFaces IDE. It features three main windows:

- Left Window (Project Explorer):** Shows the project structure. The 'Feedback' project is expanded, showing 'Web Pages' with sub-items 'principal.xhtml' and 'login.xhtml'. A yellow arrow points from this window to the 'New Facelets Template Client' dialog.
- Center Window (New Facelets Template Client):** A dialog box with the following fields:
  - Steps:** 1. Choose File Type, 2. Name and Location.
  - Name and Location:**
    - File Name:** 'listar'
    - Project:** 'Feedback'
    - Folder:** 'pessoas' (with a 'Browse...' button)
    - Created File:** 'tbeans/Feedback/src/main/webapp/pessoas/listar.xhtml'
    - Template:** 'main/webapp/WEB-INF/template.xhtml' (with a 'Browse...' button)
    - Generated Root Tag:** Radio buttons for '<html>' (selected) and '<ui:composition>'.
    - Sections To Generate:** Checkboxes for 'title' and 'content' (both checked).
- Right Window (Browse Files):** A file selection dialog showing the 'Web Pages' folder. The 'template.xhtml' file is selected. A yellow arrow points from this window to the 'New Facelets Template Client' dialog.

At the bottom of the 'New Facelets Template Client' dialog, there are buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

# | PrimeFaces | Estudo de Caso | Passo 16

❏ Vamos construir uma página de listagem.

listar.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html ... >
<html ... >
  <body>
    <ui:composition template="./../WEB-INF/template.xhtml">
      <ui:define name="title">
        Feedback | Lista de Pessoas
      </ui:define>
      <ui:define name="content">
        <p:panelGrid columns="2">

          </p:panelGrid>
      </ui:define>
    </ui:composition>

  </body>
</html>
```

# PrimeFaces | Estudo de Caso | Passo 17

❑ Vamos adicionar uma dataTable na página de listagem.

listar.xhtml

```
<ui:define name="content">
    <p:panelGrid columns="1">
        <f:facet name="header">Lista de Pessoas</f:facet>
        <p:dataTable var="pessoa" >
            <p:column headerText="Foto" width="72px">
                <h:graphicImage width="64px"/>
            </p:column>
            <p:column headerText="Nome">
                <h:outputText />
            </p:column>
            <p:column headerText="Nascimento">
                <h:outputText />
            </p:column>
            <p:column headerText="Cargo">
                <h:outputText />
            </p:column>
            <p:column headerText="Ações">
                <p:button value="Editar" icon="ui-icon-edit" />
            </p:column>
        </p:dataTable>
    </p:panelGrid>
</ui:define>
```

**f:facet** define um papel no componente.

**p:dataTable** define uma tabela para exibição de dados.

Coluna  
Foto

Coluna  
Nome

Coluna  
Nascimento

Coluna  
Cargo

Coluna  
Ações

# | PrimeFaces | Estudo de Caso | Passo 17

- ❑ Associar os atributos de pessoa as colunas da tabela.

listar.xhtml

```
<ui:define name="content">
    <p:panelGrid columns="1">
        <f:facet name="header">Lista de Pessoas</f:facet>
        <p:dataTable var="pessoa" >
            <p:column headerText="Foto" width="72px">
                <h:graphicImage width="64px" value="#{pessoa.fotoUrl}"/>
            </p:column>
            <p:column headerText="Nome">
                <h:outputText value="#{pessoa.nome}" />
            </p:column>
            <p:column headerText="Nascimento">
                <h:outputText value="#{pessoa.nascimento}" />
            </p:column>
            <p:column headerText="Cargo">
                <h:outputText value="#{pessoa.cargo}" />
            </p:column>
            <p:column headerText="Ações">
                <p:button value="Editar" icon="ui-icon-edit" />
            </p:column>
        </p:dataTable>
    </p:panelGrid>
</ui:define>
```

# | PrimeFaces | Estudo de Caso | Passo 18

❏ Vamos criar o botão de **adicionar** novas pessoas.

listar.xhtml

```
<ui:define name="content">
  <p:panelGrid columns="1">
    ...
    </p:dataTable>
  </p:panelGrid>
  <div style="clear: both; padding-top: 10px" />
  <p:button
    value="Adicionar"
    outcome="/pessoas/cadastro"
    icon="ui-icon-plus"
    style="float: right;" />

</ui:define>
```

# | PrimeFaces | Estudo de Caso | Passo 19

❏ Vamos criar uma página **cadastro.xhtml** de pessoas seguindo o template.xhtml

Listar.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html ... ">
<html ... >
  <body>
    <ui:composition template="../../../WEB-INF/template.xhtml">
      <ui:define name="title">
        Feedback | Cadastro
      </ui:define>
      <ui:define name="content">
        content
      </ui:define>
    </ui:composition>
  </body>
</html>
```

# PrimeFaces | Estudo de Caso | Passo 19

cadastro.xhtml

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

```
<h:form>
```

```
<p:messages ></p:messages>
```

```
<p:panelGrid columns="2" style="width: 100%">
```

```
<p:outputLabel value="Nome:" for="nome" />
```

```
<p:inputText id="nome"/>
```

**Nome**

```
<p:outputLabel value="Sobrenome:" for="sobrenome" />
```

```
<p:inputText id="sobrenome" />
```

**Sobrenome**

```
<p:outputLabel value="Data de Nascimento:" for="datanasc" />
```

```
<p:calendar id="datanasc" pattern="dd/MM/yyyy" />
```

**Nascimento**

```
<p:outputLabel value="URL Foto:" for="url" />
```

```
<p:inputText id="url" />
```

**URL**

```
<p:outputLabel value="E-Mail:" for="email" />
```

```
<p:inputText id="email" />
```

**E-mail**

```
<p:outputLabel value="Cargo" for="cargo" />
```

```
<p:selectOneMenu id="cargo" ></p:selectOneMenu>
```

**Cargo**

```
<p:outputLabel value="Bio" for="bio" />
```

```
<p:editor id="bio" />
```

**Bio**

```
</p:panelGrid>
```

```
<div style="clear: both; padding-top: 10px" />
```

```
<p:button value="Voltar" style="float: left" icon="ui-icon-back" outcome="/pessoas/listar"/>
```

```
<p:commandButton ajax="false" value="Salvar" style="float: right" icon="ui-icon-save"/>
```

**Ações para  
Salvar e Retornar**

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

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



# | PrimeFaces | Estudo de Caso | Passo 21

❏ Vamos criar um managed bean para a página de cadastro!

```
com.acme.feedback.controller.PessoaCadastroBean.java
```

```
@Named
@RequestScoped
public class PessoaCadastroBean {

    private Pessoa pessoa = new Pessoa();

    @Inject
    private PessoaFacade facade;

    // getters e setters

    public String salvar(){
        facade.create(pessoa);
        return "/pessoas/listar";
    }
}
```

# | PrimeFaces | Estudo de Caso | Passo 22

❏ Vamos criar um managed bean para listar os cargos!

```
com.acme.feedback.controller.CargoListaBean.java
```

```
@Named
@RequestScoped
public class CargoListaBean {

    private List<Cargo> cargos;

    @Inject
    private CargoFacade facade;

    // getters e setters ...

    @PostConstruct
    public void init() {
        cargos = facade.findAll();
    }

    public Cargo findCargo(Long id) {
        return facade.find(id);
    }
}
```

# | PrimeFaces | Estudo de Caso | Passo 23



cadastro.xhtml

...

```
<p:panelGrid columns="2" style="width: 100%">
  <p:outputLabel value="Nome:" for="nome" />
  <p:inputText id="nome" value="#{pessoaCadastroBean.pessoa.nome}"/>
  <p:outputLabel value="Sobrenome:" for="sobrenome" />
  <p:inputText id="sobrenome" value="#{pessoaCadastroBean.pessoa.sobrenome}"/>
  <p:outputLabel value="Data de Nascimento:" for="datanasc" />
  <p:calendar id="datanasc" pattern="dd/MM/yyyy" value="#{pessoaCadastroBean.pessoa.nascimento}" />
  <p:outputLabel value="URL Foto:" for="url" />
  <p:inputText id="url" value="#{pessoaCadastroBean.pessoa.fotoUrl}"/>
  <p:outputLabel value="E-Mail:" for="email" />
  <p:inputText id="email" value="#{pessoaCadastroBean.pessoa.email}"/>
  <p:outputLabel value="Cargo" for="cargo" />
  <p:selectOneMenu id="cargo" value="#{pessoaCadastroBean.pessoa.cargo}">
    <f:selectItems value="#{cargoListaBean.cargos}"
      var="cargo" itemLabel="#{cargo.nome}" itemValue="#{cargo}"/>
  </p:selectOneMenu>
  <p:outputLabel value="Bio" for="bio" />
  <p:editor id="bio" value="#{pessoaCadastroBean.pessoa.bio}"/>
</p:panelGrid>
<div style="clear: both; padding-top: 10px" />
<p:button value="Voltar" style="float: left" icon="ui-icon-back" outcome="/pessoas/listar"/>
<p:commandButton ajax="false" value="Salvar" action="#{pessoaCadastroBean.salvar()}" style="float: right"
icon="ui-icon-save"/>
```

...

# | PrimeFaces | Estudo de Caso | Passo 24

❏ Precisamos criar um conversor para que o JSF consiga exibir o select

```
com.acme.feedback.converter.CargoConverter.java
```

```
@FacesConverter(forClass = Cargo.class)
```

```
public class CargoConverter implements Converter {
```

```
    @Override
```

```
    public Object getAsObject(FacesContext context, UIComponent component, String value) {
```

```
        Application app = context.getApplication();
```

```
        CargoListaBean cargos = app.evaluateExpressionGet(context, "#{cargoListaBean}", CargoListaBean.class);
```

```
        return cargos.findCargo(Long.valueOf(value));
```

```
    }
```

```
    @Override
```

```
    public String getAsString(FacesContext context, UIComponent component, Object value) {
```

```
        return ((Cargo) value).getId().toString();
```

```
    }
```

```
}
```

# | PrimeFaces | Estudo de Caso | Passo 25



❑ Precisamos exibir as pessoas criadas na listagem, para isso vamos criar um bean

```
com.acme.feedback.controller.PessoaListaBean.java
```

```
@Named
@ViewScoped
public class PessoaListaBean implements Serializable {

    @Inject
    private PessoaFacade facade;

    private List<Pessoa> pessoas;

    public List<Pessoa> getPessoas() {
        return pessoas;
    }

    @PostConstruct
    public void init(){
        pessoas = facade.findAll();
    }

}
```

# | PrimeFaces | Estudo de Caso | Passo 25

❏ Obter os dados do controller para exibir na View


listar.xhtml

```
<ui:define name="content">
    <p:panelGrid columns="1">
        <f:facet name="header">Lista de Pessoas</f:facet>
        <p:dataTable var="pessoa" value="#{pessoaListaBean.pessoas}" >
            <p:column headerText="Foto" width="72px">
                <h:graphicImage width="64px" value="#{pessoa.fotoUrl}"/>
            </p:column>
            <p:column headerText="Nome">
                <h:outputText value="#{pessoa.nome}" />
            </p:column>
            <p:column headerText="Nascimento">
                <h:outputText value="#{pessoa.nascimento}" />
            </p:column>
            <p:column headerText="Cargo">
                <h:outputText value="#{pessoa.cargo.nome}" />
            </p:column>
            <p:column headerText="Ações">
                <p:button value="Editar" icon="ui-icon-edit" />
            </p:column>
        </p:dataTable>
    </p:panelGrid>
</ui:define>
```

# | PrimeFaces | Estudo de Caso



☐ Já conseguimos cadastrar e listar as pessoas do banco de dados!






Feedback

**Cadastros**

- [Pessoas](#)
- [Avaliações](#)

**Relatórios**

- [Usuários por Cargos](#)

Lista de Pessoas				
Foto	Nome	Nascimento	Cargo	Ações
	John	Tue Nov 21 00:00:00 BRST 2017	Ciências Atuariais	<a href="#">^ Editar</a>
	John 2	Tue Nov 21 00:00:00 BRST 2017	Ciências Atuariais	<a href="#">^ Editar</a>
	com.acme.feedback.model.Ca id=1 ]	Thu Nov 16 00:00:00 BRST 2017		<a href="#">^ Editar</a>
	AAA	Thu Nov 23 00:00:00 BRST 2017	Ciências Contábeis	<a href="#">^ Editar</a>

[+ Adicionar](#)

# PrimeFaces | Estudo de Caso | Passo 26

## ❏ Adicionando função **editar**

listar.xhtml

```
<ui:define name="content">
    <p:panelGrid columns="1">
        <f:facet name="header">Lista de Pessoas</f:facet>
        <p:dataTable var="pessoa" value="#{pessoaListaBean.pessoas}" >
            ...

            <p:column headerText="Ações">
                <p:button value="Editar" outcome="/pessoas/cadastro" icon="ui-icon-edit">
                    <f:param name="id" value="#{pessoa.id}" />
                </p:button>
            </p:column>
        </p:dataTable>
    </p:panelGrid>
</ui:define>
```

**Vamos passar para o cadastro  
o id da pessoa na linha atual.**



# | PrimeFaces | Estudo de Caso | Passo 27

❏ Vamos receber o parâmetro e associá-lo a uma propriedade do managed bean.

cadastro.xhtml

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

```
  <f:metadata>
```

```
    <f:viewParam name="id" value="#{pessoaCadastroBean.selectedId}"/>
```

```
  </f:metadata>
```

```
  <h:form>
```

```
    ...
```

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

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

**Vamos receber o id da linha atual  
e associá-lo ao controller.**

# | PrimeFaces | Estudo de Caso | Passo 28

❏ Vamos suportar inclusão e edição no **PessoaCadastroBean**

```
com.acme.feedback.controller.PessoaCadastroBean.java
```

```
@Named
@RequestScoped
public class PessoaCadastroBean {

    private Long selectedId;
    private Pessoa pessoa = new Pessoa();

    @Inject
    private PessoaFacade facade;

    // ... getter e setter de pessoa

    public Long getSelectedId() {
        return selectedId;
    }

    public void setSelectedId(Long selectedId) {
        this.selectedId = selectedId;
        pessoa = facade.find(selectedId);
    }

    public String salvar(){
        if (pessoa.getId() != null) {
            facade.edit(pessoa);
        } else {
            facade.create(pessoa);
        }
        return "/pessoas/listar";
    }
}
```

# | PrimeFaces | Estudo de Caso | Passo 29

❏ Vamos suportar remoção adicionando o botão em `personas/listar.xhtml`

listar.xhtml

```
<p:dataTable var="pessoa" value="#{pessoaListaBean.pessoas}" >
// ...
<p:column headerText="Ações">
  <p:button value="Editar" outcome="/personas/cadastro" icon="ui-icon-edit">
    <f:param name="id" value="#{pessoa.id}" />
  </p:button>
  <h:form>
    <p:commandButton value="Excluir" ajax="false"
      action="#{pessoaListaBean.excluir(pessoa)}" icon="ui-icon-trash">
    </p:commandButton>
  </h:form>
</p:column>
</p:dataTable>
```

# | PrimeFaces | Estudo de Caso | Passo 29

❏ Precisamos adicionar o método `excluir` à listagem de pessoas

```
com.acme.feedback.controller.PessoaListBean.java

@Named
@ViewScoped
public class PessoaListBean implements Serializable {

    @Inject
    private PessoaFacade facade;

    private List<Pessoa> pessoas;

    public List<Pessoa> getPessoas() {
        return pessoas;
    }

    @PostConstruct
    public void init(){
        pessoas = facade.findAll();
    }

    public String excluir(Pessoa pessoa){
        facade.remove(pessoa);
        init();
        return "/pessoas/listar?faces-redirect=true";
    }

}
```

# | PrimeFaces | ATIVIDADE I

- ❑ A partir do Estudo de Caso proposto:
  - ❑ Restrinja o acesso à páginas internas de todos os usuários que não estiverem autenticados.
  - ❑ Exiba o nome do usuário logado e uma opção de sair no menu lateral.

# | Componentes | Referências

1. [https://drive.google.com/file/d/0B7kPrfCRft\\_qZzZyeUhmWIVXZU0/preview?pli=1](https://drive.google.com/file/d/0B7kPrfCRft_qZzZyeUhmWIVXZU0/preview?pli=1)
2. <http://www.oracle.com/technetwork/articles/java/java-primefaces-2191907.html>
3. <https://github.com/juneau001/AcmePools>
4. <https://www.youtube.com/watch?v=5aTFiNxzXF4&list=UUMCAnviCQhTUAMisGZ0cb0Q>
5. [https://www.primefaces.org/docs/guide/primefaces\\_user\\_guide\\_6\\_1.pdf](https://www.primefaces.org/docs/guide/primefaces_user_guide_6_1.pdf)
6. <https://www.primefaces.org/documentation/>
7. <https://www.primefaces.org/themes/>
8. <https://www.primefaces.org/showcase>