

## Lab 3 – Primefaces

Neste laboratório iremos conhecer alguns componentes do primefaces. Para informações adicionais, tutoriais e documentação acesse o site <http://www.primefaces.org/>.

### Exercícios

**Exercício 1:** Adicionando Primefaces ao projeto

**Exercício 2:** Modificando url-pattern e definindo um tema primefaces

**Exercício 3:** Formulários e Processamento Parcial

### Exercício 1 – Adicionando Primefaces ao projeto

1. Modifique o pom.xml do projeto Livraria-web adicionando o repositório do primefaces.

```
<repositories>

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

</repositories>
```

2. Agora ainda no pom.xml adicione as seguintes dependências.

```
<!-- PrimeFaces (biblioteca de Componentes) -->
<dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>5.3</version>
</dependency>
```

3. Todas as páginas que serão criadas e usarão o primefaces terão que declarar as seguintes tags:

```
<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">
</html>
```

### Exercício 2 – Modificando url-pattern e definindo um tema do primefaces

1. Para adicionar um tema modifique o pom.xml adicionando a seguinte dependência:

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

Nesse Caso o tema selecionado foi o bootstrap, mas você pode colocar qualquer tema do primefaces, inclusive criar um tema no site: <http://jqueryui.com/themeroller/>

2. Agora modifique o web.xml para declarar a dependência.

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

3. Para modificar a url-pattern/url de acesso, edite o arquivo web.xml conforme o código abaixo:

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.xhtml</url-pattern>
</servlet-mapping>

<welcome-file-list>
  <welcome-file>/index.xhtml</welcome-file>
</welcome-file-list>
```

Anteriormente a url de acesso era  
http://localhost:8080/SuaAplicacao/faces/suaPagina.xhtml  
agora será http://localhost:8080/SuaAplicacao/suaPagina.jsf

4. Para testar, crie uma página index.xhtml com seguinte código:

```
<!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"
      xmlns:p="http://primefaces.org/ui">
<h:head>
  <meta charset="UTF-8" />
  <title>Bem Vindo ao Primefaces</title>
</h:head>
<h:body>
  <h:form>
    <h:panelGrid columns="2" cellpadding="5">

      <p:outputLabel for="date" value="Data: " />

      <!-- calendario -->

      <p:calendar id="date" value="01/01/2016" showOn="button">
        <f:convertDateTime pattern="MM/dd/yyyy" type="date" />
      </p:calendar>

      <!-- campo com máscara -->

      <p:outputLabel for="phone" value="Fone: " />
      <p:inputMask id="phone" value="1234567890"
        mask="(999) 999-9999" />

      <p:outputLabel for="cpf" value="CPF: " />
      <p:inputMask id="cpf" value="12345678900"
        mask="99999999-99" />

      <p:outputLabel for="key" value="Chave Produto: " />
      <p:inputMask id="key" value="C1-234-P567"
        mask="a*-999-a999" />

      <p:commandButton value="Limpar" type="reset" />
      <!-- mostra caixa de dialog -->
```

```

<p:commandButton value="Enviar" update="display"
    oncomplete="PF('dlg').show()" />

</h:panelGrid>
<p:panel id="basic" header="Descrição" footer="*seu editor"
    style="margin-bottom:20px">
    <p:editor id="editor" value="" />
</p:panel>

<!-- componente caixa de dialogo -->
<p:dialog widgetVar="dlg" modal="true" resizable="false"
    header="Valores No formulário" showEffect="fade">
    <p:panelGrid id="display" columns="2"
        columnClasses="label,value">

        <h:outputText value="Data: " />
        <h:outputText id="dateValue" value="12345678" />

        <h:outputText value="Fone: " />
        <h:outputText id="phoneValue" value="1234567890" />

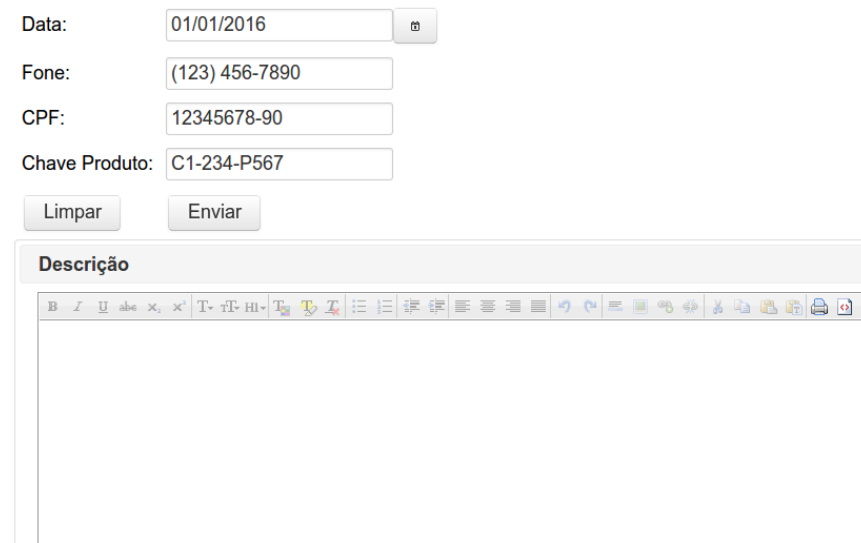
        <h:outputText value="CPF: " />
        <h:outputText id="cpfValue" value="1234567890" />

        <h:outputText value="Chave Produto: " />
        <h:outputText id="keyValue" value="abcd-123-a890" />
    </p:panelGrid>
</p:dialog>
</h:form>
</h:body>
</html>

```

5. Agora execute no servidor tomcat acessando a seguinte url:

<http://localhost:8080/Livraria-web/index.xhtml>



Data: 01/01/2016

Fone: (123) 456-7890

CPF: 12345678-90

Chave Produto: C1-234-P567

Limpar Enviar

**Descrição**

Rich text editor toolbar and area.

6. Brinque um pouco mude o tema, procure em <http://www.primefaces.org/themes>, veja o quanto e simples mudar o tema de um sistema desenvolvido em Primefaces.

### Exercício 3 – Formulários e Processamento Parcial

1. Crie uma nova página cadastroPF.xhtml, vamos testar algumas tags do primefaces. Primeiro, iremos testar um formulário de cadastro juntamente com um Managed Bean, notando as funcionalidades do primefaces. Copie o código abaixo, prestando atenção nas tags.

```
<!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/jsp/jstl/core"
      xmlns:p="http://primefaces.org/ui">
<h:head>
  <meta charset="UTF-8" />
  <title>Formulario Primefaces</title>
</h:head>
<h:body>
  <h:form>
    <p:messages autoUpdate="true" />
    <p:panelGrid columns="2">
      <f:facet name="header">Cadastro</f:facet>

      <p:outputLabel value="Nome" for="nome" />
      <p:inputText id="nome" value="#{cadastroBean.nome}" />

      <p:outputLabel value="Email" for="email" />
      <p:inputText id="email" value="#{cadastroBean.email}"
        required="true" label="email" />

      <p:outputLabel value="Senha" for="senha" />
      <p:password id="senha" value="#{cadastroBean.senha}"
        required="true" label="senha" />

      <f:facet name="footer">
        <p:commandButton value="Cadastrar"
          action="#{cadastroBean.cadastrar}"
          icon="ui-icon-disk"
          iconPos="right" />
      </f:facet>
    </p:panelGrid>
  </h:form>
</h:body>
</html>
```

2. Agora, para testar nosso formulário, vamos criar um ManagedBean **CadastroBean** para ler e nos mandar as informações recolhidas no formulário.

```
import java.io.Serializable;

import javax.faces.bean.ManagedBean;
import javax.faces.view.ViewScoped;

@ManagedBean
@ViewScoped
public class CadastroBean implements Serializable{

    private static final long serialVersionUID = 1L;

    private String nome;
    private String email;
    private String senha;
```

```
public void cadastrar(){
    System.out.println("Nome: "+this.nome);
    System.out.println("Email: "+this.email);
    System.out.println("Senha: "+this.senha);
}
//getters e setters omitidos
}
```

3. Ao clicar em cadastrar, o sistema irá rodar o objeto **cadastar()** do seu **CadastroBean** e imprimir as informações de cada campo no console do Eclipse. Você pode checar essas e outras aplicações do primefaces no site <http://www.primefaces.org/showcase/>. Nesse site estão reunidas todas as tags e elementos que o primefaces pode oferecer, além de poder testar outros temas.
4. Uma das vantagens de usar o **Primefaces** é que em seus elementos de envio de formulário (**commandButton** e **commandLink**) já vem com o ajax, que faria o envio destes sem ter que recarregar a página todas as vezes, isso se chama **Processamento Parcial**. No seu formulário do exercício anterior, faça as seguintes alterações.
  - a. Na classe **CadastroBean**, adicione o seguinte código.

```
public void verificarDisponibilidade(){
    FacesMessage msg = null;

    if("joao".equalsIgnoreCase(this.nome)){
        msg= new FacesMessage("Esse usuário já está em uso");
        msg.setSeverity(FacesMessage.SEVERITY_WARN);
    }
    else{
        msg=new FacesMessage("Usuário Disponível!");
    }

    FacesContext.getCurrentInstance().addMessage(null, msg);
}
```

- b. Agora, vá ao seu arquivo **cadastroForm.xhtml** e modifique conforme mostrado abaixo.

```
<p:outputLabel value="Nome" for="nome"/>
<h:panelGroup>
    <p:inputText id="nome" value="#{cadastroBean.nome}"/>
    <p:commandButton value="Verificar"
        action="#{cadastroBean.verificarDisponibilidade}"
        process="nome @this"/>
</h:panelGroup>
```

Esse **commandButton**, com o atributo **process**, fará com que você submeta parte do formulário, verificando o nome do usuário sem precisar percorrê-lo por inteiro. Faça os testes, retire o atributo **process** e veja como funciona.

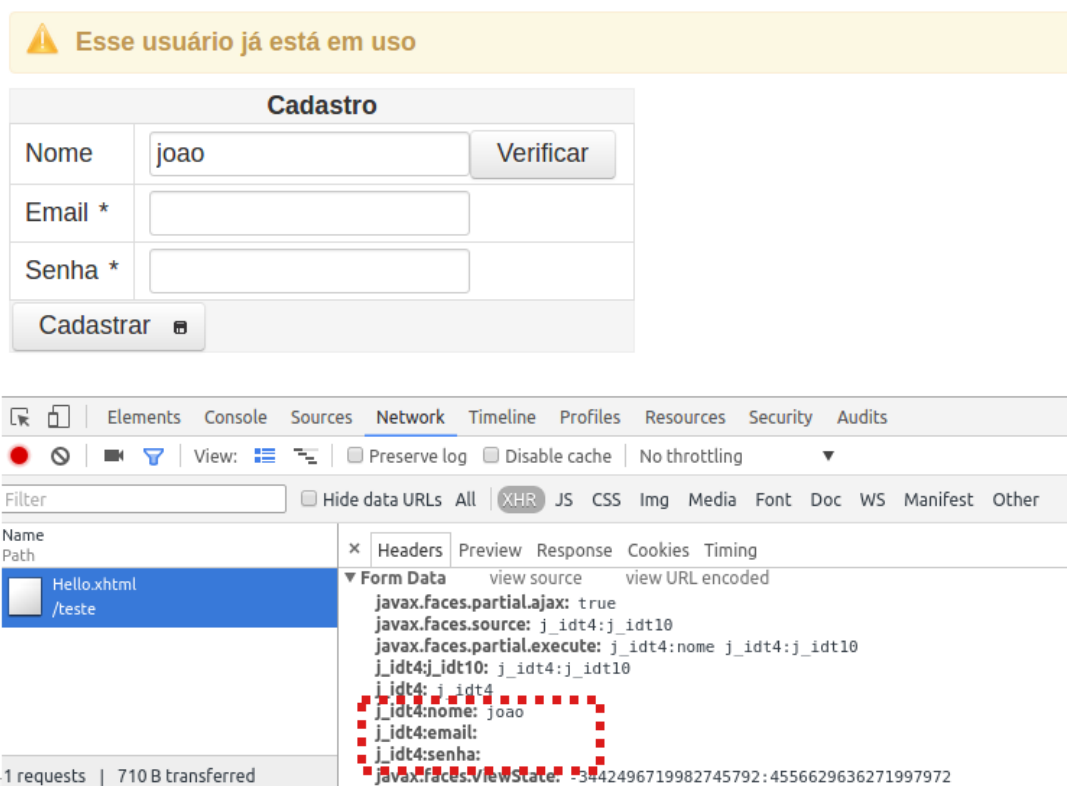
5. Agora, vamos fazer esse processamento com ajax. Assim que você escreve o nome e tira o foco da caixa de texto, o

ajax fará o processamento do campo e enviará a mensagem, sem precisar de botão para isso. Modifique seu código assim como está abaixo.

```
<p:outputLabel value="Nome" for="nome" />
<p:inputText id="nome" value="#{cadastroBean.nome}">
    <p:ajax listener="#{cadastroBean.verificarDisponibilidade}"
            event="change" process="@this" />
</p:inputText>
```

Faça os testes e veja a função da tag **ajax** no primefaces.

- Note que, ao fazer a verificação, o ajax envia todas as informações do formulário, sendo que só precisamos do nome. Isso gera um desperdício de rede, de memória, etc.



The screenshot shows a web application interface with a registration form titled 'Cadastro'. The form has three input fields: 'Nome' (containing 'joao'), 'Email \*', and 'Senha \*'. There is a 'Verificar' button next to the 'Nome' field and a 'Cadastrar' button at the bottom. A yellow message bar at the top says 'Esse usuário já está em uso'. Below the form, the browser's developer tools are open, showing the 'Network' tab. A request to 'Hello.xhtml' is selected, and the 'Form Data' section is expanded, showing the following data:

```
javax.faces.partial.ajax: true
javax.faces.source: j_idt4:j_idt10
javax.faces.partial.execute: j_idt4:nome j_idt4:j_idt10
j_idt4:j_idt10: j_idt4:j_idt10
j_idt4: j_idt4
j_idt4:nome: joao
j_idt4:email:
j_idt4:senha:
javax.faces.ViewState: -3442496719982745792:4556629636271997972
```

Para que possamos enviar somente o campo necessário, a tag **p:ajax** tem um atributo que faz a submissão parcial do formulário, ou seja, irá mandar somente o campo desejado. Faça as mudanças necessárias e teste o código a seguir.

```
<p:outputLabel value="Nome" for="nome" />
<p:inputText id="nome" value="#{cadastroBean.nome}">
    <p:ajax listener="#{cadastroBean.verificarDisponibilidade}"
            event="change" process="@this" partialSubmit="true" />
</p:inputText>
```

⚠ Esse usuário já está em uso

### Cadastro

Nome	<input type="text" value="joao"/>
Email *	<input type="text"/>
Senha *	<input type="password"/>
<input type="button" value="Cadastrar"/>	

Elements Console Sources **Network** Timeline Profiles Resources Security Audits

View: ☐ Preserve log ☐ Disable cache No throttling

Filter ☐ Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

Name	Path	Headers	Preview	Response	Cookies	Timing
Hello.xhtml	/teste	User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36 X-Requested-With: XMLHttpRequest				
		▼ Form Data	view source	view URL encoded		
		javax.faces.partial.ajax: true javax.faces.source: j_idt4:nome javax.faces.partial.execute: j_idt4:nome javax.faces.behavior.event: change javax.faces.partial.event: change j_idt4:nome: joao				

1 / 9 requests | 710 B / 4.6 KB transfer