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>*.jsf</url-pattern>
</servlet-mapping>

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

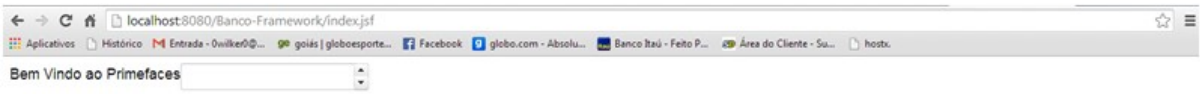
Anteriormente a url de acesso era http://localhost:8080/SuaAplicacao/faces/suaPagina.xhtml agora será http://localhost:8080/SuaAplicacao/suaPagina.jsf

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

```
<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:outputLabel value="Bem Vindo ao Primefaces"/>

    <p:spinner />

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

Da forma que o primefaces foi declarado em xmlns:p="http://primefaces.org/ui" Todos os componentes do primefaces são acessados com **<p:**ComponenteQueForUtilizar />

5. Agora execute no servidor tomcat acessando a seguinte url: http://localhost:8080/Livraria-web/index.jsf



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. No arquivo index.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.

```
<?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://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>

        <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" />

                        <p:outputLabel value="Email" for="email"/>
                        <p:inputText id="email" required="true" label="email"/>

                        <p:outputLabel value="Senha" for="senha"/>
                        <p:password id="senha" required="true" label="senha"/>
                        <f:facet name="footer">
                                <p:commandButton value="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. Faça algumas alterações em seu arquivo xhtml para conseguir pegar as informações da sua página.

```
<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>
```

Ao clicar em cadastrar, o sistema irá rodar o objeto **cadastrar()** do seu **CadastroBean** e imprimir as informações de cada campo no console do Eclipse, como mostrado abaixo.

Você pode checar essas e outras aplicações do primefaces no site http://www.primefaces.org/showcase/ . Nesse site estao reunidas todas as tags e elementos que o primefaces pode oferecer, alem 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 exercicio anterior, faça as seguintes alterações.

a. Na classe **CadastroBean**, adicione o seguinte código.

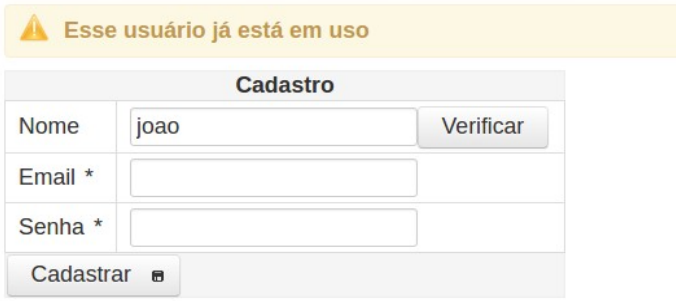
```
public void verificarDisponibilidade(){

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

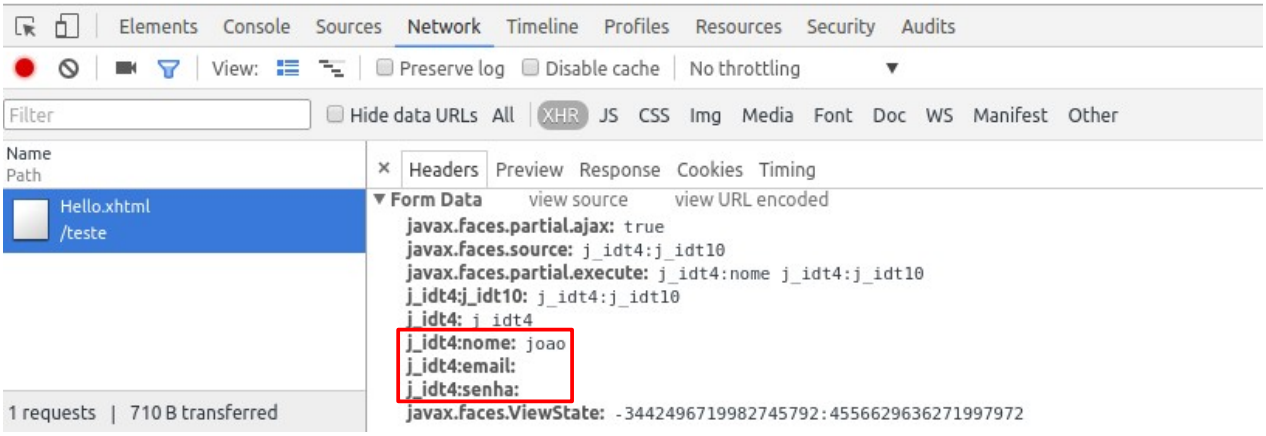


5. Agora, vamos fazer esse processamento com ajax. Assim que você escreve o nome e tira o foco da caixa de texto, o ajax fará o processamento do campo e enviará a mensagem, sem precisar de botão para isso. Modifique seu código assim como está abaixo.

```
<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.

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



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 necesárias 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>
```