**Introdução ao JSF e Primefaces**

O JavaServer Faces (JSF) é uma tecnologia que nos permite criar aplicações Java para Web utilizando componentes (formulário, campos de entrada de texto e senhas, rótulos, links, botões, mensagens, painéis, tabela de dados, etc**.**)visuais **pré-prontos**, de forma que o desenvolvedor não se preocupe com Javascript e HTML. Basta adicionarmos os componentes (calendários, tabelas, formulários) e eles serão renderizados e exibidos em formato html.

JSF é baseado no padrão de projeto MVC (**Model** **View** **Controller**), o que torna o desenvolvimento de sistemas menos complicado. O padrão MVC separa o sistema em três responsabilidades (modelo, visualização e controle).

**Características do JSF**

A API de JSF suporta a extensão e criação de novos componentes, que podem fornecer

funcionalidades adicionais. Os principais frameworks que implementam e estendem a especificação JSF, são:

* **Richfaces** (<http://richfaces.org>)
* **Primefaces** (<http://primefaces.org>)
* **Icefaces** (<http://icefaces.org>)

**Preparação do ambiente**

Inclusão das dependências de para o JSF e Primefaces no arquivo **pom.xml.**

**<dependency>**

**<groupId>**javax**</groupId>**

**<artifactId>**javaee-api**</artifactId>**

**<version>**7.0**</version>**

**<scope>**provided**</scope>**

**</dependency>**

**<dependency>**

**<groupId>**org.primefaces**</groupId>**

**<artifactId>**primefaces**</artifactId>**

**<version>**5.3**</version>**

**</dependency>**

Como estamos usando Maven, não precisamos acessar o site do Hibernate para baixar os arquivos necessários e incluir manualmente no projeto. Podemos incluir todas as dependências no arquivo **pom.xml**, que o Maven baixará os arquivos necessários automaticamente.

**Adicionar suporte ao JSF no projeto do eclipse**

Botão direito no projeto => propriedades => project facets => marcar a opção JavaServerFaces => Desabilitar a configuração de biblioteca de implementação => Confirmar.

**Configuração do controlador do JSF**

Adicionar a configuração no arquivo web.xml, para fazer o mapeamento das requisições que serão processadas pelo Servlet do JSF. É comum usarmos o padrão **\*. xhtml** para o mapeamento do Servlet para evitar que o usuário consiga ter acesso ao código fonte da nossa página (View).

<servlet>

<servlet-name>FacesServlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>FacesServlet</servlet-name>

<url-pattern>\*.xhtml</url-pattern>

</servlet-mapping>

**Obs.:** Esse mapeamento é opcional, por convenção as requisições com o padrão **\*.faces** serão tratadas pelo Servlet.

**A primeira página com JSF**

<!DOCTYPE html>

<html xmlns="http://www.w3c.org/1999/xhtml"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:f="http://java.sun.com/jsf/core"

<h:head>

<meta charset="utf-8" />

</h:head>

<h:body>

<h1>PROJETO CARRINHO DE COMPRAS</h1>

<h:button value="BOTÃO JSF"></h:button>

<br/>

<hr/>

</h:body>

</html>

**Adicionando suporte ao Primefaces**

<!DOCTYPE html>

<html xmlns="http://www.w3c.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"

xmlns:ui="http://java.sun.com/jsf/facelets">

<h:head>

<meta charset="utf-8" />

</h:head>

<h:body>

<h1>PROJETO CARRINHO DE COMPRAS</h1>

<h:button value="BOTÃO JSF"></h:button>

<br/>

<hr/>

<p:button value="BOTÃO PRIMEFACES"></p:button>

</h:body>

</html>

**Integração com o controller da aplicação**

Para mostrar tais informações, precisaremos executar um código Java e certamente não faremos isso na camada de visualização: esse código ficará separado da **view**, em uma classe de modelo. Essas classes de modelo que interagem com os componentes do JSF são os **Managed Beans**.

Estes, são apenas “classezinhas” simples que com as quais o JSF consegue interagir através do acesso a seus métodos. Nada mais são do que **POJOs** anotados com **@ManagedBean**.

@ManagedBean

public class OlaMundoBean {

public String

getHorario() {

SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");

return "Atualizado em " + sdf.format(new Date());

}

}

**Lembretes**:

* Todo **Managed Bean** é acessado através de seu nome (nome da classe definida com a primeira minúscula)
* É possível modificar o nome do ManagedBean atráves do atributo **name** da anotação **@ManagedBean**.
* O acesso aos **ManagedBeans** da aplicação a partir das **views** (páginas) é feito através da EL (**Expression Language**).

**Escopos de Managed Beans**

**@NoneScoped:** o bean será instanciado a cada vez que for referenciado**.**

**@RequestScoped (padrão):** tem vida curta, começando quando é referenciado em uma única requisição HTTP e terminando quando a resposta é enviada de volta ao cliente.

**@ViewScoped:** a instância permanece ativa até que o usuário navegue para uma próxima página.

**@SessionScoped:** mantém a instância durante diversas requisições e até mesmo navegações entre páginas, até que a sessão do usuário seja invalidada ou o tempo limite é atingido. Cada usuário possui sua sessão de navegação, portanto, os objetos não são compartilhados entre os usuários.

**@ApplicationScoped**: mantém a instância durante todo o tempo de execução da aplicação. É um escopo que compartilha os objetos para todos os usuários do sistema.

**Obs.:** Essas anotações estão contidas no pacote **javax.faces.bean.** A especificação Java EE também possui suas próprias anotações de escopo, e estão contidas no pacote **javax.enterprise.context**

**CDI**

CDI (**Contexts and Dependency Injection**) é a especificação da Java EE que trabalha com injeção de dependências. Podemos usar CDI para instanciar e injetar objetos de nossa aplicação.

**Escopos**

**@RequestScoped**: Escopo relativo a requisições

**@SessionScoped**: Escopo relativo a sessão do navegador

**@ApplicationScoped**: O maior escopo possível

**@ConversationScoped**: Esse escopo fornece a possibilidade de controlar até onde o escopo é válido.

**@Dependent**: O escopo padrão do CDI

**@ViewScoped**: O escopo de mantido enquanto o usuário permanecer na mesma página.

**@Inject**: Para fazer o uso de injeções.

**@Named**: Para dar nomes aos beans fazendo com que eles sejam acessíveis através da EL (Expression Language). Essa anotação equivale a **@ManagedBean** que será vista posteriormente.

**Obs.:** Um **Managed Bean** pode ser identificado e manipulado pelo controller JSF utilizando também a anotação **@Named,** indicando que o controller da aplicação é um **Bean CDI**, podendo ser acessado por EL por meio do próprio nome. As anotações de escopo do CDI estão contidas no pacote **javax.enterprise.context.** Enquanto a anotação **@Named** pertence ao pacote **javax.inject.**

**Habilitando o uso de CDI no projeto**

* Botão direito no projeto => propriedades => project facets => marcar a opção CDI => Confirmar.
* No diretório **src/main/resources/META-INF**, criamos um arquivo vazio chamado **beans.xml**. A existência desse arquivo habilita CDI no projeto. No Eclipse EE já é oferecido a opção de criação desse arquivo.

**Referências**

**Livro de apoio**: <http://alga.works/livro-javaee/>

**Tutorial JSF e Primefaces:** <http://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/introducao-ao-jsf-e-primefaces/>

**Componentes Primafaces:** <http://www.primefaces.org/showcase/>

**Tutorial Primefaces:** <http://www.primefaces.org/docs/guide/primefaces_user_guide_5_3.pdf>

**EL (Expression Language**): <http://docs.oracle.com/javaee/7/tutorial/jsf-el.htm#GJDDD>