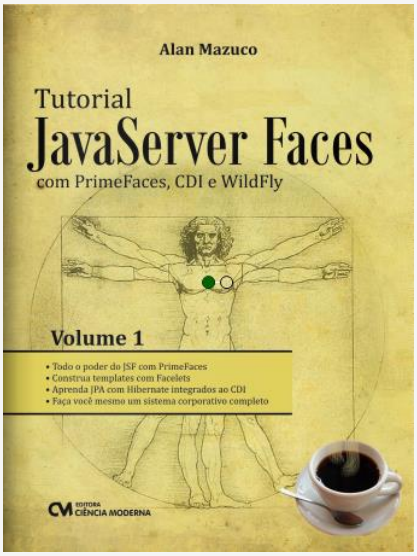


****

GUIA DO CURSO DO VOLUME 1

****

* sempre quando o código-fonte for diferente do livro (se isto ocorrer, é porque houve uma modernização), ele será colocado neste guia, na íntegra (código que foi modernizado) ";

** Quando o código-fonte (ou qualquer outro recurso) não for encontrado neste guia, é porque ele não sofreu alteração alguma, e deverá ser utilizado o disponível no site www.amazu.com.br, ou junto com o material distribuído nas respectivas lições deste curso;

*** A teoria sobre cada uma das videoaulas apresentadas pode ser obtida no livro **Tutorial JavaServer Faces, com PrimeFaces, CDI e WildFly** disponível em todas as livrarias do Brasil;

**** A sequência das videoaulas está rigorosamente sincronizada com os laboratórios do livro, mas as videoaulas não dependem dele e podem ser executadas sem a ajuda dele;

***** O livro é importante para o aprendizado teórico e complementa este curso, embora também seja mais focado na prática.

* * * * *

PALAVRAS INICIAIS

VIDEOAULA 1

Palavras iniciais do instrutor

Aula inaugural de boas vindas do instrutor.

* * * * *

INTRODUÇÃO AOS LABORATÓRIOS DO VOL. I

VIDEOAULA 2

Panorama das vídeoaulas do Volume 1

A videoaula apresenta um panorama completo de todas as lições do Volume I do curso e orienta o aluno sobre o passo-a-passo de cada uma delas, sobre as ferramentas e recursos disponibilizados.

* * * * *

LIÇÃO 1 - APRESENTANDO O ECLIPSE

VIDEOAULA 3

LAB 1.1 – Download do JDK

Sequência de passos:

1) Navegar até http://www.oracle.com/technetwork/java/javase/downloads/index.html

2) Baixar o jdk 8u251 (não baixe versão diferente desta). O nome do arquivo é jdk-8u251-windows-x64.exe

VIDEOAULA 4

LAB 1.2 – Instalando o JDK e o Eclipse

Sequência de passos:

1) Baixar e instalar o 7-Zip: https://www.7-zip.org/
2) Instalar o JDK
3) Alterar a variável PATH do sistema
4) Baixar e instalar a última versão do Eclipse: https://www.eclipse.org/downloads/packages/

VIDEOAULA 5

LAB 1.3 – Disponibilização de um atalho para o Eclipse na Área de Trabalho

Sequência de passos:

1) Disponibilizar o atalho na Área de Trabalho do Windows

VIDEOAULA 6

LAB 1.4 – Abrindo o Eclipse pela primeira vez

Sequência de passos:

1) Abrir o Eclipse
2) Configurar o Workspace
3) Observar as visões do Eclipse

VIDEOAULA 7

LAB 1.5 – Configurando a Package Explorer

Sequência de passos:

1) Conhecer as perspectivas no Eclipse
2) Configurar Java Element Filters

VIDEOAULA 8

LAB 1.6 – Configurando a Perspectiva Java

Sequência de passos:

1) Como despoluir a Perspectiva Java
2) Configurar as visões da Perspectiva Java

VIDEOAULA 9

LAB 1.7 – Configurando as Preferências do Usuário

Sequência de passos:

1) Configurar elementos da *Code Editor*

VIDEOAULA 10

LAB 1.8 – Meu Primeiro Projeto no Eclipse

Sequência de passos:

1) Criar o projeto
2) Criar os pacotes
3) Criar a classe AloEclipse
4) Editar a classe AloEclipse
5) Rodar o programa AloEclipse

VIDEOAULA 11

LAB 1.9 – Utilizando o recurso *Quick Fix* do Editor de Código

Sequência de passos:

1) Demonstração do uso do *Quick Fix*

VIDEOAULA 12

LAB 1.10 – Utilizando o recurso *Code Assist* do Editor de Código

Sequência de passos:

1) Demonstração do uso do *Code Assist*

VIDEOAULA 13

Checklist dos objetivos da Lição 1

A ***Lição 1 – APRESENTANDO O ECLIPSE*** teve por finalidade:

- ☐Instalar o Eclipse;
- ☐ Customizar o ambiente de trabalho;
- ☐ Reconhecer um projeto Java dentro do Eclipse;
- ☐ Manipular o Editor de Códigos;
- ☐ Criar pacotes e classes;
- ☐ Criar o primeiro projeto dentro do Eclipse.

Tente repassar cada um desses objetivos, localizando-os nas vídeoaulas assistidas.

* * * * *

LIÇÃO 2 – WEB: PRIMEIROS PASSOS

VIDEOAULA 14

LAB 2.1 – Tornando o seu Computador um Servidor Web para Testes

Sequência de passos:

1) Realizar o download do WildFly: https://wildfly.org
2) Instalar o WildFly

VIDEOAULA 15

LAB 2.2 – Teste de Funcionamento do WildFly

Sequência de passos:

1) Realizar o teste do WildFly fora do Eclipse

VIDEOAULA 16

Checklist dos objetivos da Lição 2

A ***Lição 2 – WEB PRIMEIROS PASSOS*** teve por finalidade:

- ☐ Conhecer os os termos comuns usados na World Wide Web
- ☐ Entender o protocolo HTTP/IP
- ☐ Entender como funciona a comunicação entre portas em um computador
- ☐ Instalar e testar o WildFly

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 3 – CONFIGURANDO O ECLIPSE PARA PROJETOS WEB

VIDEOAULA 17

LAB 3.1 – Configurando a Code Editor

Sequência de passos:

1) Definir a aparência dos arquivos XML
2) Definir a aparência dos arquivos CSS, HTML e JSP

VIDEOAULA 18

LAB 3.2 – Configurando o Encoding e inabilitando as validações

Sequência de passos:

1) Configurar o encoding do workspace para UTF-8
2) Configurar o Eclipse para que ele suspenda todas as validações de arquivos XML, HTML e XHTML

VIDEOAULA 19

LAB 3.3 – Configurando o Web Browser interno

Sequência de passos:

1) Disponibilizar um atalho do Web Browser na Perspectiva Java

VIDEOAULA 20

LAB 3.4 – Equalizando a JVM

Sequência de passos:

1) Equalizar a JVM do Eclipse com a JDK instalada

VIDEOAULA 21

LAB 3.5 – Configurando o WildFly no Eclipse

Sequência de passos:

1) Instalar um Server Adapter para o WildFly
2) Reiniciar o Eclipse
3) Criar uma instância do servidor WildFly
4) Testar o WildFly pelo Eclipse

VIDEOAULA 22

Checklist dos objetivos da Lição 3

A ***Lição 3 – CONFIGURANDO O ECLIPSE PARA PROJETOS WEB*** teve por finalidade:

- ☐ Customizar o Eclipse para projetos Web
- ☐ Configurar e testar o WildFly de dentro do Eclipse

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 4 – NOÇÕES DE SERVLETS

VIDEOAULA 23

LAB 4.1 – Criando um novo projeto com o Maven: O BlankApp

Sequência de passos:

1) Criar o projeto Web com Maven (o BlankApp)
2) Fixar o Java 1.8
   - Java Compiler
   - Project Faces (mudar a versão do Dynamic Web Module para 4.0 e Java para 1.8)
3) Criar o arquivo web.xml
4) Corrigir o arquivo pom.xml atualizando-o com as bibliotecas mais atuais
5) Comandar <Maven Update>
6) Verificar se o projeto está livre de erros antes de continuar
7) Configurar o servidor WildFly no projeto
8) Testar se o projeto foi adicionado no WildFly (rodar o projeto e constatar livre de erros)
9) Criar pacotes
10) Criar a classe HelloServlet
11) Rodar o projeto e testar os recursos do servidor

Recursos modificados (modernizados)

web.xml

```
<web-app
	xmlns="http://xmlns.jcp.org/xml/ns/javaee"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
		http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
	version="4.0">

</web-app>
```

Pom.xml

```
<project
	xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
		https://maven.apache.org/xsd/maven-4.0.0.xsd">

	<modelVersion>4.0.0</modelVersion>
	<groupId>br.eti.amazu</groupId>
	<artifactId>blankapp</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>war</packaging>

	<name>BlankApp</name>
	<description>adf asdf</description>

	<properties>
		<project.build.sourceEncoding>ISO-8859-1</project.build.sourceEncoding>
		<maven.compiler.source>1.8</maven.compiler.source>
		<maven.compiler.target>1.8</maven.compiler.target>
	</properties>

	<build>
		<pluginManagement>

			<!--plugins compiladores da versao 3.8.1 do Maven e JSE 8-->
			<plugins>
```

```
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.8.1</version>
                    <configuration>
                        <source>1.8</source>
                        <target>1.8</target>
                    </configuration>
                </plugin>

                <plugin>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>3.2.3</version>
                    <configuration>
                        <failOnMissingWebXml>false</failOnMissingWebXml>
                        <warName>blankapp</warName>
                    </configuration>
                </plugin>
                <!--fim de plugins compiladores da vers 3.8.1 do Maven e JSE 8-->

            </plugins>

        </pluginManagement>

        <outputDirectory>
            ${basedir}/src/main/webapp/WEB-INF/classes
        </outputDirectory>

    </build>

    <dependencies>

        <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
        <dependency>
          <groupId>javax.servlet</groupId>
          <artifactId>javax.servlet-api</artifactId>
          <version>4.0.1</version>
          <scope>provided</scope>
        </dependency>

    </dependencies>
</project>
```

VIDEOAULA 24

LAB 4.2 – Meu Segundo Servlet: JSP

Sequência de passos:

1) Criar uma página index.jsp
2) Rodar o servidor

VIDEOAULA 25

Checklist dos objetivos da Lição 4

A ***Lição 4 – NOÇÕES DE SERVLETS*** teve por finalidade:

- ☐ Descrever um Servlet
- ☐ Aprender noções básicas de Maven
- ☐ Construir um simples aplicativo usando Maven
- ☐ Analisar a estrutura básica de um projeto Web

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 5 – HELLO JSF!

VIDEOAULA 26

LAB 5.1 – Tornando o BlankApp um projeto de JSF

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Capacitar o Eclipse para o uso de **JSF-2.3**
3) Adicionar as dependências de **wildfly-jsf** no arquivo pom.xml
4) Reconfigurar e formatar o arquivo **web.xml**
5) Reposicionar o arquivo **faces-config.xml**
6) Adicionar o arquivo **beans.xml** (habilitar o **CDI**)
7) Comandar <Maven Update Project>
8) Comandar <Maven clean install>
9) Observar e interpretar os logs na view Console
10) Comandar <Refresh> na raiz do projeto
11) Comandar <Full publish> na view Servers e observar o comportamento
12) Rodar o projeto e constatar livre de erros
13) Parar o projeto (botão “Terminate”)

Recursos modificados (modernizados)

Nova dependência para o pom.xml (WildFly 19)

```
<dependency>
    <groupId>org.wildfly</groupId>
    <artifactId>wildfly-jsf</artifactId>
    <version>19.0.0.Final</version>
    <scope>provided</scope>
</dependency>
```

beans.xml

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

<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/beans_2_0.xsd"
    bean-discovery-mode="all"
    version="2.0">

</beans>
```

web.xml

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

<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
    version="4.0">

    <!-- Titulo do aplicativo -->
    <context-param>
       <param-name>appTitle</param-name>
       <param-value>Amazu Tecnologia</param-value>
    </context-param>
```

```
    <!--Pagina inicial -->
    <welcome-file-list>
       <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <!-- Configuracoes do JSF/Facelets -->
    <context-param>
        <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
        <param-value>.xhtml</param-value>
    </context-param>

    <context-param>
        <param-name>javax.faces.CONFIG_FILES</param-name>
        <param-value>/WEB-INF/xml/faces-config.xml</param-value>
    </context-param>
    <!-- Fim de configuracoes do JSF/Facelets -->

    <!-- Configuracoes do FacesServlet -->
    <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>*.faces</url-pattern>
    </servlet-mapping>
    <!-- Fim de configuracoes do FacesServlet -->

</web-app>
```

VIDEOAULA 27

LAB 5.2 – Adicionando recursos ao BlankApp e testando

Sequência de passos:

1) Criar as pastas **pages** e **resources** sob *src/main/webapp*
2) Criar a pasta *pages/home*
3) Criar a pasta *resources/images*
4) Criar a pasta *resources/images/skin*
5) Copiar o arquivo **loader.gif** para dentro da pasta *resources/images*
6) Criar a página webapp/**index.html**
7) Criar a página webapp/pages/**homePage.xhtml**
8) Sincronizar o servidor (comandar <Full publish> na view Servers)
9) Testar

Checklist dos objetivos da Lição 5

A ***Lição 5 – HELLO JSF!*** teve por finalidade:

- ☐ Conhecer o JSF
- ☐ Entender o projeto BlankApp: suas possibilidades e limitações
- ☐ Criar um aplicativo Web baseado em JSF

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 6 – FACELETS

VIDEOAULA 29

LAB 6.1 – Criando pacotes e folders

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Parar o servidor (se estiver ligado)
3) Criar as pastas **animatedHtmls**, **logos** e **css**, livremente dentro de *src/main/webapp/resources*
4) Criar a pasta **vetruvian** dentro de *src/main/webapp/resources/images/skin*
5) Criar a pasta **layout** dentro de *WEB-INF*
6) Adicionar os arquivos:
   - ✓ **animated_favicon.gif** e **favicon.ico** na pasta *src/main/webapp/resources/images*
   - ✓ **blank.gif**, **ui-further-row.png** e **ui-just-row.png** na pasta *src/main/webapp/resources/images/skin*
   - ✓ **blankapp_topo_anime.xhtml** na pasta *src/main/webapp/resources/animatedHtmls*
   - ✓ **amazuLogo.gif** na pasta *src/main/webapp/resources/logos*
   - ✓ **img_background.gif**, **topo_left.png** e **topo_right.png** na pasta *src/main/webapp/resources/images/skin/vetruvian*
7) Observar como ficou a estrutura do projeto comparando-a com a do instrutor

VIDEOAULA 30

LAB 6.2 – Modificando o web.xml

Sequência de passos:

1) Adicionar código ao arquivo **web.xml**, após a última tag </context-param>

Recursos modificados (modernizados)

Adicione isto ao web.xml

```
<context-param>
    <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
    <!-- desenvolvimento 0 milissegundo.-->
    <param-value>0</param-value>
    <!-- producao desligado.
    <param-value>-1</param-value>   -->
</context-param>
```

```
<!-- omitido o param javax.faces.PROJECT_STAGE
     assumindo o default (Production) que servirah, tambem, para o
     desenvolvimento, desabilitando algumas msgs de erro.-->

<!-- forcando a limpeza do cache -->
<context-param>
     <param-name>com.sun.faces.defaultResourceMaxAge</param-name>
     <!-- desenvolvimento 1 milissegundo.-->
     <param-value>1</param-value>
     <!-- producao 6 weeks.
     <param-value>3628800000</param-value>  -->
</context-param>

<context-param>
     <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
     <param-value>server</param-value>
</context-param>
<!--Fim de Configuracoes do JSF/Facelets -->
```

VIDEOAULA 31

LAB 6.3 – Criando estilos

Sequência de passos:

1) Criar e editar o arquivo **my_style.css**, dentro da pasta *src/main/webapp/resources/css*

VIDEOAULA 32

LAB 6.4 – Um Value Object para as configurações globais

Sequência de passos:

1) Criar o pacote *br.eti.amazu.infra.view.vo*
2) Criar e codificar a classe **Config** dentro do pacote *br.eti.amazu.infra.view.vo*

VIDEOAULA 33

LAB 6.5 – Criando um bean de configurações

Sequência de passos:

1) Criar o pacote *br.eti.amazu.infra.view.bean*
2) Criar a classe **ConfigBean** dentro do pacote *br.eti.amazu.infra.view.bean.*
3) Substituir as duas anotações (ManagedBean e SessionScoped) por:

```
@Named
@SessionScoped

(Antecipação da Lição 11)
```

VIDEOAULA 34

LAB 6.6 – Criando as templates do layout

Sequência de passos:

1) Dentro da pasta WEB-INF/layout, crie as páginas:

   footer.xhtml
   hDynaMenu.xhtml
   vDynaMenu.xhtml
   header.xhtml
   template.xhtml

2) Modificar a estrutura da **homePage.xhtml**

VIDEOAULA 35

LAB 6.7 – O primeiro teste do BlankApp

Sequência de passos:

1) Rodar o servidor e testar
2) Parar o servidor. Na classe ConfigBean, alterar o valor de **menuType** para **"tiered"** e testar
3) Parar o servidor. Na classe ConfigBean, trocar o valor de **skinAnimatedTop** para **"T"** (true) e testar novamente
4) Abrir o Google Chrome e testar

VIDEOAULA 36

Checklist dos objetivos da Lição 6

A ***Lição 6 – TEMPLATES!*** teve por finalidade:

- ☒ Entender como funciona o Facelets no JSF 2
- ☐ Criar templates
- ☐ Desenvolver arquivos de estilo
- ☐ Criar um bean de configurações

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 7 – POR DENTRO DO JSF

VIDEOAULA 37

LAB 7.1 – Uma classe para controlar o FacesContext

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Criar o pacote *br.eti.amazu.infra.util*
3) Criar e codificar a classe **FacesUtil** dentro do pacote *br.eti.amazu.infra.util*

VIDEOAULA 38

Checklist dos objetivos da Lição 7

A ***Lição 7*** teve por finalidade:

- ☐ Apontar noções gerais sobre o JSF Framework
- ☐ Investigar o gerenciamento de beans
- ☐ Aprender como funciona os escopos do JSF
- ☐ Usar, de forma simples, os componentes commandButton, commandLink, form, panelGrid, panelGroup, inputText e outputText
- ☐ Utilizar o Ajax como mecanismo de renderização
- ☐ Manipular genericamente o FacesContext

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 8 – CICLO DE VIDA DO JSF

VIDEOAULA 39

LAB 8.1 – Implementando PhaseListener

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Sob o pacote *br.eti.amazu.infra.view* crie o pacote *showcase.listener*
3) Criar a classe **PhaseTracker** dentro do pacote *br.eti.amazu.infra.view.showcase.listener*
4) Adicionar código no arquivo **faces-config.xml**
5) Testar

VIDEOAULA 40

Checklist dos objetivos da Lição 8

A ***Lição 8*** teve por finalidade:

- ☐ Entender o ciclo de vida do JSF
- ☐ Construir um PhaseListener customizado
- ☐ Testar uma implementação de PhaseListener

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 9 – PRIMEFACES, PRIMEIROS PASSOS

VIDEOAULA 41

LAB 9.1 – Adicionando o PrimeFaces ao pom.xml

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Adicionar a dependência do **PrimeFaces** ao arquivo pom.xml
3) Comandar **Maven update**
4) Comandar **clean install**
5) Comprovar que a biblioteca do PrimeFaces **não está sob o escopo de provided**

VIDEOAULA 42

LAB 9.2 – Configurando o PrimeFaces no BlankApp

Sequência de passos:

1) Adicionar, em todas as páginas criadas até agora, o NameSpace **xmlns:p="http://primefaces.org/ui"** (exceto nas páginas index)

VIDEOAULA 43

LAB 9.3 – Redigitando a Template

Sequência de passos:

1) Redigitar a página template.xhtml

VIDEOAULA 44

LAB 9.4 – Observando o comportamento responsivo do BlankApp

Sequência de passos:

1) Modificar a página footer.xhtml
2) Testar no browser

VIDEOAULA 45

LAB 9.5 – Navegando entre Home Page e Home Show Case

Sequência de passos:

1) Criar a pasta *src/main/webapp/pages/**showcase***
2) Criar a página **homeShowCase.xhtml** dentro da pasta *src/main/webapp/pages/showcase*
3) Comentar (desabilitar) a progressBar
4) Abra a homePage e substitua o título da página por um botão do PrimeFaces
5) Dentro da homePage, comente todos os includes após o comentário da progressBar
6) Rodar o servidor e chamar o aplicativo no browser
7) Clicar no botão “Show Case”
8) Role a página para baixo e perceba o rodapé fixo na página
9) Clicar no botão para retornar à homePage

VIDEOAULA 46

Checklist dos objetivos da Lição 9

A ***Lição 9*** teve por finalidade:

- ☐ Configurar o PrimeFaces
- ☐ Conhecer o CommandButton do PrimeFaces
- ☐ Compreender as novas regras de navegação do JSF

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 10 – INTERNACIONALIZAÇÃO

VIDEOAULA 47

LAB 10.1 – Criando os arquivos properties

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Criar e editar, dentro de *src/main/resources*, os arquivos:
   messages_en_US.properties
   messages_pt_BR.properties

VIDEOAULA 48

LAB 10.2 – Configurar o Resource Bundle e Locale

Sequência de passos:

1) Abrir o arquivo **faces-config.xml** e:

   Configurar o **ResourceBundle**
   Configurar o **idioma (locale)**
   Desabilitar a classe **PhaseTracker**

VIDEOAULA 49

LAB 10.3 – Criando a classe UserSessionInBean

Sequência de passos:

1) Sob pacote *br.eti.amazu.infra.view.bean*, crie a classe **UserSessionInBean**
2) Abra a classe **ConfigBean** e adicione o atributo ***locales*** e os métodos ***getLocales*** / ***setLocales***

VIDEOAULA 50

LAB 10.4 – Internacionalize a Home Page

Sequência de passos:

1) Internacionalizar a **homePage**
2) Disponibilizar novas imagens em *images/skin*

VIDEOAULA 51

LAB 10.5 – Alterando header e template (internacionalização)

Sequência de passos:

1) Alterar a página **header.xhtml**
2) Alterar a **template.xhtml**

VIDEOAULA 52

LAB 10.6 – Criando a página que lista os idiomas suportados

Sequência de passos:

1) Criar a pasta *src/main/webapp/pages/showcase/locales*
2) Criar a página **listaLocales.xhtml** dentro de *src/main/webapp/pages/showcase/locales*

VIDEOAULA 53

LAB 10.7 – Testando a internacionalização

Sequência de passos:

1) Rodar o servidor
2) Abrir o BlankApp no browser e comprovar o funcionamento da internacionalização em todos os aspectos vistor até gora

VIDEOAULA 54

Checklist dos objetivos da Lição 10

A ***Lição 10*** teve por finalidade:

- ☐ Utilizar a internacionalização em um aplicativo JavServer Faces
- ☐ Entender o MessageBundle
- ☐ Reconhecer arquivos properties
- ☐ Utilizar o componente <ui:repeat>

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 11 – NOÇÕES BÁSICAS DE CDI

VIDEOAULA 55

LAB 11.1 – Configurando o CDI no arquivo beans.xml

Sequência de passos:

1) Revisar a configuração em beans.xml

VIDEOAULA 56

LAB 11.2 – Substituindo anotações de JSF por CDI

Sequência de passos:

1) Revisar as anotações básicas de CDI

VIDEOAULA 57

LAB 11.3 – Testando o CDI

Sequência de passos:

1) Relembrando o teste do CDI

VIDEOAULA 58

Checklist dos objetivos da Lição 11

A ***Lição 11*** teve por finalidade:

- ☐ Aprender noções básicas de CDI
- ☐ Entender as principais anotações do container CDI
- ☐ Conhecer e empregar escopos de CDI

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 12 – CRIANDO UM COMPONENTE DE MENSAGENS

VIDEOAULA 59

LAB 12.1 – Preparando pacotes e pastas para o componente Dialog

Sequência de passos:

1) Panorama de mensagerias do BlankApp (demonstração)
2) Baixar e descompactar os recursos da videoaula
3) Criar o pacote *br.eti.amazu.component.dialog*
4) Criar a pasta *src/main/webapp/pages/showcase/primefaces*
5) Criar a pasta *src/main/webapp/resources/components*

VIDEOAULA 60

LAB 12.2 – Criando a classe DialogUtil

Sequência de passos:

1) Criar a classe **DialogUtil**

VIDEOAULA 61

LAB 12.3 – Criando a classe DialogType

Sequência de passos:

1) Criar a classe **DialogType**

VIDEOAULA 62

LAB 12.4 – Criando a classe DialogBean

Sequência de passos:

1) Criar a classe **DialogBean**

VIDEOAULA 63

LAB 12.5 – Criando a classe DialogListener

Sequência de passos:

1) Criar a classe **DialogListener**

VIDEOAULA 64

LAB 12.6 – Criando o arquivo faces-config do componente Dialog

Sequência de passos:

1) Criar o arquivo **faces-config.xml** dentro da pasta src/main/resources/META-INF para configurar o DialogListener

VIDEOAULA 65

LAB 12.7 – Criando as páginas do componente Dialog

Sequência de passos:

1) Criar a página **dialog.xhtml**
2) Criar a página **dialogConfirm.xhtml**
3) Adicionar o **NameSpace** do componente Dialog na template
4) Ainda dentro da template, incluir as duas páginas do componente Dialog, **dialog.xhtml** e **dialogConfirm.xhtml**

VIDEOAULA 66

LAB 12.8 – Adicionando as imagens do componente Dialog

Sequência de passos:

1) Adicionar as imagens **warnIcon.png** e **status.gif** dentro da pasta *src/main/webapp/resources/images/skin*

VIDEOAULA 67

LAB 12.9 – O componente Status

Sequência de passos:

1) Criar o componente Status – a página **status.xhtml**, dentro de *resources/components*

VIDEOAULA 68

LAB 12.10 – Preparando as classes de teste para o Dialog

Sequência de passos:

1) Criar o pacote *br.eti.amazu.infra.view.showcase.**vo***
2) Dentro do pacote *br.eti.amazu.infra.view.showcase.vo*, criar a classe **Veiculo**
3) Dentro do pacote *br.eti.amazu.infra.view.showcase*, criar a classe **MessageCaseBean**

VIDEOAULA 69

LAB 12.11 – Preparando a página de teste para o Dialog

Sequência de passos:

1) Dentro de *src/main/webapp/pages/showcase/primefaces,* criar a página **msgPrimeFaces.xhtml**

VIDEOAULA 70

LAB 12.12 – Reconfigurando o pom.xml e testando

Sequência de passos:

1) Abra o arquivo **pom.xml** e verifique que não há a necessidade de adicionar mais nenhuma dependência, até o presente

2) Comandar Maven *clean install*
3) Realize os testes

VIDEOAULA 71

LAB 12.13 – Preparando o empacotamento do componente

Sequência de passos:

1) Comandar Maven *clean install*
2) Comandar Maven *Update project*
3) Comandar Refresh na raiz do projeto
4) Comandar *Full publish*
5) Analisar a estrutura do componente Dialog
6) Criar uma pasta fora do projeto (dentro do seu *dev – preferencialmente dev/**build***), para guardar os compilados do componente
7) Criar a estrutura do componente dentro da pasta *dev/build*, conforme as orientações
8) Copiar os arquivos compilados do Eclipse para dentro da estrutura do componente em *dev/build*

VIDEOAULA 72

LAB 12.14 – Empacotando o Dialog para distribuição

Sequência de passos:

1) Compactar o componente com o **7-Zip**
2) Renomear o componente para **dialog-1.0.jar**

VIDEOAULA 73

LAB 12.15 – Criação do arquivo *jar* de distribuição

Sequência de passos:

1) Compilar o componente para dentro do repositório **Maven** local
2) Navegar até o repositório local do Maven e analisar o que foi criado

VIDEOAULA 74

LAB 12.16 – Como utilizar um componente no projeto

Sequência de passos:

1) Faça, agora, uma cópia do projeto. Pode copiar o projeto para dentro da pasta dev/**backup** (sugestão)
2) Apague o pacote *br.eti.amazu.component.dialog*
3) Abra o arquivo src/main/resources/META-INF/**faces-config.xml** e apague o código dentro da tag <faces-config>
4) Exclua todo o conteúdo da pasta *src/main/webapp/resources/components*
5) Perceba que agora existem vários erros, reclamando a ausência de um componente Dialog
6) Inclua a dependência do componente Dialog no arquivo **pom.xml**
7) Comande Maven **Update Project**
8) Comande **Refresh** na raiz do projeto e perceba que os erros desaparecem

VIDEOAULA 75

LAB 12.17 – Testando o componente Dialog (*jar*)

Sequência de passos:

1) Conferir o arquivo **settings.xml** (Maven)
2) Comandar **Full publish**
3) Rodar o servidor e testar
4) Para o servidor
5) Adicionar o repositório do instrutor e a dependência do **Dialog-1.5.jar** e testar
6) Parar o servidor
7) Abrir a pasta *C:\java\wildFly-19.0.0-Final\standalone\deployments* e **apagar tudo o que está dentro dela**
8) Abra o Eclipse e copie o arquivo **blankapp-0.0.1-SNAPSOHOT.war** e cole-o dentro da pasta *C:\java\wildFly-19.0.0-Final\standalone\deployments*, renomeando-o para **blankapp.war**
9) Rodar o servidor por fora do Eclipse (acionando o arquivo **standalone.bat**)

10) Abra o browser e teste o aplicativo novamente

Recursos modificados (modernizados)

Repositório do instrutor (pom.xml)

```
<repositories>
    <repository>
        <id>amazu-repo</id>
        <url>http://amazu.eti.br/maven/repo</url>
        <releases>
            <checksumPolicy>warn</checksumPolicy>
            <enabled>true</enabled>
            <updatePolicy>always</updatePolicy>
        </releases>
    </repository>
</repositories>
```

Dependência no pom.xml (JSF-2.3 e PrimeFaces-8.0)

```
<dependency>
    <groupId>br.eti.amazu.component</groupId>
    <artifactId>dialog</artifactId>
    <version>1.5</version>
</dependency>
```

VIDEOAULA 76

Checklist dos objetivos da Lição 12

A ***Lição 12*** teve por finalidade:

- ☐ Criar um listener personalizado
- ☐ Criar Expression Method
- ☐ Empregar o componente <p:dialog>
- ☐ Manipular mensagens com o componente <p:messages>
- ☐ Criar um componente JSF
- ☐ Manipular o Maven com segurança

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 13 – TRATAMENTO DE ERROS HTTP

VIDEOAULA 77

LAB 13.1 – Criando as páginas de erros HTTP

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Adicionar código no arquivo web.xml
3) Criar a pasta *src/main/webapp/pages/**errorpages***
4) Criar as seguintes páginas dentro de *errorpages*:

   error404.xhtml
   error500.xhtml
   errorServlet.xhtml
   error502.xhtml
   error503.xhtml

5) Adicionar as imagens **icon_warn**, **icon_error**, **icon_bad**, **icon_busy**, **icon_info** e **icon_servlet** dentro da pasta *webapp/resources/images/skin*

VIDEOAULA 78

LAB 13.2 – Teste das páginas de erros HTTP

Sequência de passos:

1) Rodar o servidor
2) Abrir um browser e digitar na barra de endereços

   http://localhost:8080/blankapp/pagina1
   Tecla enter - O que aconteceu?

3) Abrir a página homePage.xhtml e trocar a tag <h:form> por <h:forms> - salvar
4) No browser, digite:

   http://localhost:8080/blankapp
   Tecla enter - OLa que aconteceu?

5) Breve análise

VIDEOAULA 79

LAB 13.3 – Tratando a sessão expirada com PrimeFaces

Sequência de passos:

1) Dentro de *src/main/webapp/pages/errorpages*, crie a página **viewExpired.xhtml**

2) Abra a template.xhtml e adicione o código para habilitar o <p:idleMonitor>
3) Modifique o valor do timeout para 3000 milissegundos, apenas para testar
4) Rode o servidor e veja o que acontece
5) Parar o servidor
6) Retornar o valor de timeout para 2.400.000 Milissegundos (default do projeto)

VIDEOAULA 80

LAB 13.4 – Tratando a sessão expirada com OmniFaces

Sequência de passos:

1) Justificativa de uso do OmniFaces
2) Abrir o arquivo web.xml e adicionar novo código (habilitar o OmniFaces)
3) Adicionar a dependência do **OmniFaces** no arquivo pom.xml
4) Executar os seguintes comandos:
   Update Project
   Clean install
   Refresh
   Full publish

Recursos modificados (modernizados)
Dependência do OmniFaces (pom.xml)

```
<dependency>
    <groupId>org.omnifaces</groupId>
    <artifactId>omnifaces</artifactId>
    <version>3.6</version>
</dependency>
```

VIDEOAULA 81

Checklist dos objetivos da Lição 13

A ***Lição 13*** teve por finalidade:

- ☐ Conhecer os principais erros HTTP
- ☐ Tratar erros de HTTP
- ☐ Lidar com sessão do usuário expirada

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 14 – CRIANDO UM COMPONENTE PROGRESSBAR

VIDEOAULA 82

LAB 14.1 – Codificando a Progressbar – Classes Java

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Criar o pacote **progressbar**, dentro de *br.eti.amazu.component*
3) Dentro de *br.eti.amazu.component.progressbar*, Crie as seguintes classes:

   ProgressUtil
   ProgressBean

VIDEOAULA 83

LAB 14.2 – Criando a classe de teste da Progressbar

Sequência de passos:

1) Dentro do pacote *br.eti.amazu.infra.view.showcase*, crie a classe **progressbarCaseBean**

VIDEOAULA 84

LAB 14.3 – Criando a página do componente Progressbar

Sequência de passos:

1) Dentro da pasta *webapp/resources/components*, crie a página **progressbar.xhtml**
2) Por que este componente não precisa estar registrado no faces-config.xml?

VIDEOAULA 85

LAB 14.4 – Teste da Barra de Progresso

Sequência de passos:

1) Certifique-se do NameSpace, prefixo “a” na página homeShowCase
2) Habilitar o componente declarando-o na página homeShowCase
3) Analisar o comando que coloca a barra de progresso na tela (commandLink ou commandButton)
4) Rodar o servidor
5) Testar
6) Parar o servidor
7) Deseja empacotar o componente?

VIDEOAULA 86

Checklist dos objetivos da Lição 14

A ***Lição 14*** teve por finalidade:

☐ Utilizar o componente <p:progressbar>

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 15 – THEMES

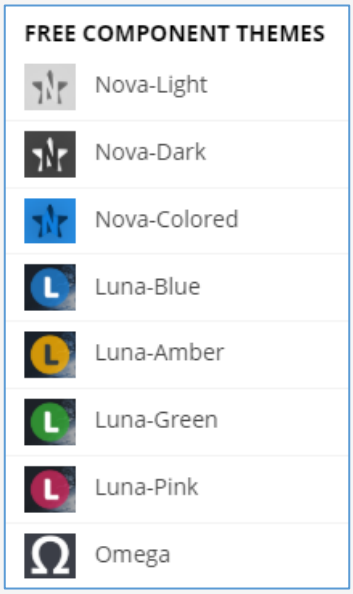
VIDEOAULA 87

LAB 15.1 – Configurando Temas

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula

2) Os temas da **PrimeFaces** foram modernizados



3) Abra a página **homeShowCase** e troque o valor do atributo “icon” do botão que redireciona para a Home Page, para:

```
pi pi-home
```

4) Navegar até o site
https://www.primefaces.org/showcase/ui/misc/primeicons.xhtml

e observe a nova biblioteca de ícones do PromeFaces

5) Abra o arquivo **web.xml** e adicione isto (após a última tag de <contexto-param>:

```
<context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>#{configBean.skinTheme}</param-value>
</context-param>
```

6) Adicione isto na sseção **head** da **template.xhtml**:

```
<h:outputStylesheet
    library="#{configBean.config.skinTheme}"
    name="theme.css"/>
```

7) Dentro do pacote *br.eti.amazu.component*, crie mais um pacote com o nome **pworld.domain**

8) Dentro do pacote *br.eti.amazu.component.pworld.domain*, crie a classe **AbstractEntity**, conforme lista abaixo:

```
package br.eti.amazu.component.pworld.domain;

import java.io.Serializable;

/*
 * Classe base para entidades que possuem um id.
 * Sobrescrevendo corretamente os metodos hashCode() e equals().
 */
public abstract class AbstractEntity<ID> implements Serializable {

	private static final long serialVersionUID = -6499669634164304829L;

	public abstract ID getId();

	@Override
	public int hashCode() {
		return 31 + ((getId() == null) ? 0 : getId().hashCode());
	}

	@Override
	public boolean equals(Object obj) {
		return (this == obj) ||
			(obj instanceof AbstractEntity &&
				(this.getId() != null &&
					this.getId().equals(((AbstractEntity<?>) obj).getId())) ||
					(this.getId() == null &&
						obj != null &&
							((AbstractEntity<?>) obj).getId() == null));
	}

}
```

9) Dentro do pacote *br.eti.amazu.infra.view* crie mais um pacote com o nome **converter**

10) Dentro de *br.eti.amazu.infra.view.converter* crie a classe **ObjectConverter**, conforme lista abaixo:

```
package br.eti.amazu.infra.view.converter;

import java.util.Map;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.FacesConverter;
import br.eti.amazu.component.pworld.domain.AbstractEntity;

@FacesConverter(value = "objectConverter")
public class ObjectConverter implements Converter<Object> {

	@Override
	public Object getAsObject(FacesContext ctx, UIComponent component,
		String value) {
		if (value != null) {
			return this.getAttributesFrom(component).get(value);
		}
		return null;
	}

	@Override
	public String getAsString(FacesContext ctx, UIComponent component,
		Object value) {

		if (value != null && ! "".equals(value)) {
			@SuppressWarnings("unchecked")
			AbstractEntity<Long> entity = (AbstractEntity<Long>) value;

			if (entity.getId() != null) {
				this.addAttribute(component, entity);

				if (entity.getId() != null) {
					return String.valueOf(entity.getId());
				}
				return (String) value;
			}
		}
		return "";
	}
```

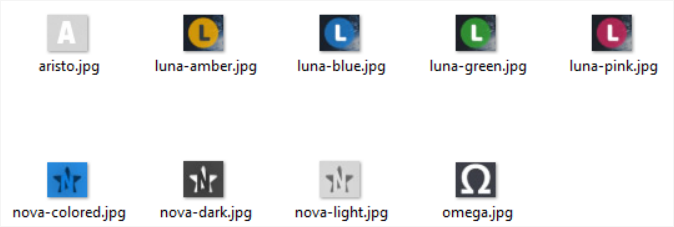
```
    private void addAttribute(UIComponent component, AbstractEntity<Long> o) {
        this.getAttributesFrom(component).put(o.getId().toString(), o);
    }

    private Map<String, Object> getAttributesFrom(UIComponent component) {
        return component.getAttributes();
    }


}
```

11) Dentro de *br.eti.amazu.infra.view.vo* crie a classe **Theme**, conforme lista abaixo:

```
package br.eti.amazu.infra.view.vo;

import java.util.ArrayList;
import java.util.List;

import br.eti.amazu.component.pworld.domain.AbstractEntity;

public class Theme extends AbstractEntity<Object>{

    private static final long serialVersionUID = 1L;

    private Long id;
    private String name;
    private String value;
    private List<Theme> themes;

    public Theme(Long id, String name, String value) {
        this.id= id;
        this.name = name;
        this.value = value;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public List<Theme> getThemes() {
        themes = new ArrayList<>();
        themes.add(new Theme(0L, "Aristo", "aristo"));
        themes.add(new Theme(1L, "Nova-Light", "nova-light"));
        themes.add(new Theme(2L, "Nova-Dark", "nova-dark"));
        themes.add(new Theme(3L, "Nova-Colored", "nova-colored"));
        themes.add(new Theme(4L, "Luna-Blue", "luna-blue"));
        themes.add(new Theme(5L, "Luna-Amber", "luna-amber"));
        themes.add(new Theme(6L, "Luna-Green", "luna-green"));
        themes.add(new Theme(7L, "Luna-Pink", "luna-pink"));
        themes.add(new Theme(8L, "Omega", "omega"));
        return themes;
    }

    public void setThemes(List<Theme> themes) {
        this.themes = themes;
    }


}
```

12) Criar a pasta **themeswitcher** dentro de *src/main/webapp/resources/images/skin*

13) Copiar mais 9 (nove) arquivos de imagens para dentro da pasta *src/main/webapp/resources/images/skin/themeswitcher*



14) Dentro da pasta */pages/showcase/primefaces* criar a página **themes.xhtml**, conforme a lista abaixo:

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    template="/WEB-INF/layout/template.xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:p="http://primefaces.org/ui"
    xmlns:a="http://java.sun.com/jsf/composite/components">

    <ui:param name="pageTitle" value="Themes"/>

    <ui:define name="content">

        <h:form id="formThemes">

            <div style="width:100%">

                <p:selectOneMenu
                    id="advanced"
                    style="width:165px;"
                    effect="fade"
                    converter="objectConverter"
                    value="#{configBean.theme}" var="t"
                    valueChangeListener="#{configBean.saveTheme}">

                    <p:ajax event="change" oncomplete="submit()"/>

                    <f:selectItems
                        value="#{configBean.themes}"
                        var="theme" itemLabel="#{theme.name}"
                        itemValue="#{theme}" />

                    <p:column>
                        <h:graphicImage
                            value="/resources/images/skin/themeswitcher
                        /#{t.value}.jpg"/>
                    </p:column>

                    <p:column>
                        #{t.value}
                    </p:column>
                </p:selectOneMenu>
            </div>

            <br/>
            <br/>

            <p:panel style="width:400px" >
                <p:calendar  mode="inline"></p:calendar>
            </p:panel>

            <br/>

            <p:commandButton value="Show Case"
                action="/pages/showcase/homeShowCase?faces-redirect=true"/>

        </h:form>
    </ui:define>
</ui:composition>
```

15) Redigitar a classe ConfigBean deixando-a conforme a lista abaixo:

```
package br.eti.amazu.infra.view.bean;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.enterprise.context.SessionScoped;
import javax.faces.event.ValueChangeEvent;
import javax.inject.Named;

import br.eti.amazu.infra.util.FacesUtil;
import br.eti.amazu.infra.view.vo.Config;
import br.eti.amazu.infra.view.vo.Theme;

@Named
@SessionScoped
public class ConfigBean implements Serializable{

	private static final long serialVersionUID = -6663659948453061860L;

	//O objeto config contem as variaveis de configuracoes do sistema.
	private Config config;

	private String skinTheme;

	//Armazena uma lista de idiomas suportados.
	private List<String> locales = new ArrayList<>();

	//-------themes--------------------
	private Theme theme;
	private List<Theme>  themes;
	//-------------------------------------

	public void setConfiguracoes(){

		/* Aqui carregando parametros defalt, a titulo de demonstracao.
		 * Esses dados serao obtidos de um arquivo de configuracoes. */

		config = new Config();

		/* TIPOS DE MENU:
		 * - menuBar ----------HORIZONTAL (default)
		 * - tiered ---------- VERTICAL
		 * - slide ---------- VERTICAL
		 * - panelMenu ------- VERTICAL */
		config.setMenuType("menuBar"); //O menu default eh menuBar (horizontal).

		//Setando os diversos parametros de skin para o aplicativo.
		config.setSkinAnimatedTop("F"); //.....Topo default "nao-animado".
		config.setSkinBackground("vetruvian"); //.....Skin default - vetruvian.
		config.setSkinImageLogo("amazuLogo.gif"); //.....Imagem logotipo da empresa.
		config.setSkinLogo("T"); //.....Topo - logotipo da empresa.
		config.setSkinTextLogo("Tecnologia Java"); //.....Texto abaixo do logotipo.
		config.setSkinColorTextLogo("13f02d"); //.....A cor do texto do logotipo;
		config.setSkinAnimatedHtml("blankapp_topo_anime.xhtml"); //.O Html5 anim

		//Isto eh o que serah escrito no rodapeh da pagina.
		config.setSkinFooter(
				"Privacy Policy | Amazu Technology | Copyright \u00A9 2018 -"
				+ " All rights reserved");


		/* ======================================
		 * MODERNIZACAO DOS TEMAS DO PRIMEFACES
		 * O tema default agora serah o nova-light.
		 ======================================*/
		theme = new Theme(1L, "Nova-Light", "nova-light");
		themes = theme.getThemes();
		skinTheme=theme.getValue();
		config.setSkinTheme(skinTheme);
		/* ====================================== */

		//logs
		System.out.println("Rodando o tema: " + config.getSkinTheme());
		System.out.println("Rodando o skin: " + config.getSkinBackground());
		System.out.println("Rodando o menu: " + config.getMenuType());
	}

	public void saveTheme(ValueChangeEvent e){
		theme = (Theme) e.getNewValue();
		theme.setThemes(themes);
		skinTheme=theme.getValue();
		config.setSkinTheme(skinTheme);
	}
```

```
    /*---------
     * get/set
     ---------*/
    public Config getConfig() {
        if(config == null){
            this.setConfiguracoes();
        }

        return config;
    }

    public void setConfig(Config config) {
        this.config = config;
    }
    public String getSkinTheme() {
        return skinTheme;
    }
    public void setSkinTheme(String skinTheme) {
        this.skinTheme = skinTheme;
    }

    public List<String> getLocales() {

        //Utliza a classe FacesUtil para obter a lista de idiomas suportados.
        if(locales.isEmpty()){
            locales = FacesUtil.getLocales();

            //realiza apenas um log
            StringBuffer strb = new StringBuffer();
            strb.append("Linguagens suportadas: ");

            int i=1;
            for(String str:locales){
                if(i == locales.size()){
                    strb.append(str);

                }else{
                    strb.append(str + ", ");
                }
                i++;
            }
            System.out.println(strb.toString());
        }
        return locales;
    }

    public void setLocales(List<String> locales) {
        this.locales = locales;
    }
    public Theme getTheme() {
        return theme;
    }
    public void setTheme(Theme theme) {
        this.theme = theme;
    }
    public List<Theme> getThemes() {
        return themes;
    }
    public void setThemes(List<Theme> themes) {
        this.themes = themes;
    }

}
```

16) Redimensionar o componente **Progressbar**, tendo em vista a modernização de temas

17) Executar os seguintes comandos:

    Update Project, Clean install, Refresh e Full publish

VIDEOAULA 88

LAB 15.2 – Testando Temas

Sequência de passos:

1) Rodar o servidor

2) Testar os temas – navegar nas páginas criadas até o presente

VIDEOAULA 89

Checklist dos objetivos da Lição 15

A ***Lição 15*** teve por finalidade:

- ☐ Configurar tipos diferentes de temas para o PrimeFAces
- ☐ Criar seus próprios temas
- ☐ Manipular arquivos *.jar*

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 16 – SIMPLES CRUD

VIDEOAULA 90

LAB 16.1 – Criando a classe CrudCaseBean

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Sob o pacote *br.eti.amazu.infra.view.showcase* crie a classe **CrudCaseBean**

VIDEOAULA 91

LAB 16.2 – Criando as páginas do módulo CRUD

Sequência de passos:

1) Dentro da pasta *pages/showcase/primefaces*, crie mais uma com o nome de **crud**
2) Dentro da pasta *pages/showcase/primefaces/crud*, crie as páginas

   veículos.xhtml
   dlgVeiculo.xhtml (fragmento)

VIDEOAULA 92

LAB 16.3 – Bloqueando o botão "Voltar" dos browsers

Sequência de passos:

1) Dentro da pasta *src/main/webapp/resources*, crie mais uma pasta com o nome **js**
2) Dentro da pasta *src/main/webapp/resources/js,* crie o arquivo **navegacao.js**
3) Dentro de *src/main/webapp/WEB-INF/layout*, crie o fragmento de página **dlgButtonBack.xhtml**
4) Abra a **template.xhtml** e habilite o arquivo JavaScript criado (na seção head)

```
<h:outputScript library="js" name="navegacao.js" target="head"/>
```

5) Ainda dentro da template.xhtml, adicione o fragmento **dlgButtonBack.xhtml**

```
<!-- tela de aviso ao utilizar botao retorno do browser -->
<ui:include src="dlgButtonBack.xhtml"/>
```

VIDEOAULA 93

LAB 16.4 – Teste do módulo CRUD

Sequência de passos:

1) Rodar o servidor
2) Acompanhar a explanação do instrutor sobre o comportamento ideal de um CRUD
3) Parar o servidor

VIDEOAULA 94

Checklist dos objetivos da Lição 16

A ***Lição 16*** teve por finalidade:

- ☐ Construir um CRUD sem usar um banco de dados
- ☐ Conhecer noções de validação de dados
- ☐ Manipular classe de domínio através de Managed Bean

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 17 – IMPLEMENTAÇÃO DE LOGS

VIDEOAULA 95

LAB 17.1 – Criando as classes de geração de log

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Abra o arquivo pom.xml e adicione apenas a dependência do **Log4j**

```
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

3) Executar os seguintes comandos:
   Update Project, Clean install e Refresh
4) Sob o pacote *br.eti.amazu.infra.util*, crie mais um pacote com o nome **log**
5) Dentro do pacote *br.eti.amazu.infra.util.log,* crie as seguintes classes:
   Ansi (enum)
   Log

VIDEOAULA 96

LAB 17.2 – Instalando o plugin ANSI Escape in Console

Sequência de passos:

1) Instalar o plugin **Ansi Escape in Console**
2) Configurar o plugin no Eclipse

VIDEOAULA 97

LAB 17.3 – Testando os nossos logs

Sequência de passos:

1) Abra a classe **MessageCaseBean** e adicione o método **testarLogs()**
2) Iniciar o servidor
3) Testar (acionando o link da Lição 17)
4) Abrir o arquivo de log do servidor **WildFly** para observar os logs

5) Desligar o servidor
6) Log de level **INFO** deve ser desabilitado em produção?

VIDEOAULA 98

LAB 17.4 – Configurando o log no WildFly

Sequência de passos:

1) Habilitar o **Management Console** do WildFly atribuindo usuário e senha
2) Abrir o **Management Console**
3) Verificar as configurações do **Subsystem Logging**
4) Ajustar o Level do console para **ERROR**
5) Testar (somente os erros de nível ERROR e FATAL_ERROR serão exibidos) – ver isto também no arquivo de log
6) Visualizar os arquivos de log na aba *Runtime/Standalone Server/Log Files*
7) Parar o servidor
8) Desfazer as alterações do WildFly, voltando às configurações anteriores

VIDEOAULA 99

Checklist dos objetivos da Lição 17

A ***Lição 17*** teve por finalidade:

- ☐ Conhecer o Log4j
- ☐ Aprender a configurar o Log4j com o WildFly
- ☐ Implementar operações de *logging* para o aplicativo
- ☐ Utilizar o plugin Ansi Escape in Console do Eclipse

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 18 – VALIDAÇÃO E CONVERSÃO NA PRÁTICA

VIDEOAULA 100

LAB 18.1 – Criação da classe abstrata AbstractEntity

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Revisitar a classe **pworld.AbstractEntity**
3) Considerações sobre o projeto **pworld**

VIDEOAULA 101

LAB 18.2 – Estendendo a base de domínio

Sequência de passos:

1) Dentro do pacote *br.eti.amazu.infra.view.showcase.vo*, crie a classe **ValueTest**

VIDEOAULA 102

LAB 18.3 – Criando validadores

Sequência de passos:

1) Dentro do pacote *br.eti.amazu.infra.view*, crie mais um pacote com o nome de **validator**
2) Dentro do pacote *br.eti.amazu.infra.view.validator*, crie as classes:

   CnpjValidator
   Phonevalidator
   CpfValidator
   EmailValidator

VIDEOAULA 103

LAB 18.4 – Criando conversores

Sequência de passos:

1) Revisitar a classe **ObjectConverter**
2) Sob o pacote *br.eti.amazu.infra.view.converter*, criar as classes:

   CepConverter
   DateConverter

VIDEOAULA 104

LAB 18.5 – Criando um validador com Regex

Sequência de passos:

1) Sob o pacote *br.eti.amazu.infra.util*, crie a classe **RegexUtil**

VIDEOAULA 105

LAB 18.6 – Criando máscaras de validação com JavaScript

Sequência de passos:

1) Sob a pasta *src/main/webapp/resources/js*, criar o arquivo **maskValidator.js**
2) Abrir a **template.xhtml** e registrar o arquivo **maskValidator.js** em sua head

VIDEOAULA 106

LAB 18.7 – Criando um bean para testar validadores e conversores

Sequência de passos:

1) Sob o pacote *br.eti.amazu.infra.view.showcase*, crie a classe **ConvValidatorCasBean**

VIDEOAULA 107

LAB 18.8 – Criando a página de teste

Sequência de passos:

1) Sob a pasta *src/main/webapp/pages/showcase/primefaces*, crie a página **valueTest.xhtml**
2) Entenda a página **ValueTest.xhtml**

VIDEOAULA 108

LAB 18.9 – Testando validações e conversões na prática

Sequência de passos:

1) Rodar o servidor
2) Abra o browser e navegue até o Blankapp
3) Abra a página Home Show Case e acione o link da Lição 18 (veja que todos os campos estão em branco)
4) Selecione os objetos na lista e verifique o que acontece
5) Clique no botão Update e veja o que acontece

VIDEOAULA 109

Checklist dos objetivos da Lição 18

A ***Lição 18*** teve por finalidade:

- ☐ Aprender a criar validadores e conversores
- ☐ Criar um conversor genérico para objetos
- ☐ Criar máscaras com JavaScript

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

LIÇÃO 19 – CRIANDO UM COMPONENTE UPLOAD

VIDEOAULA 110

LAB 19.1 – Criando as classes do componente Upload

Sequência de passos:

1) Baixar e descompactar os recursos da videoaula
2) Possibilidades para o componente Upload
3) Abrir o arquivo pom.xml e adicionar a dependência abaixo:

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
```

4) Dentro do pacote *br.eti.amazu.component*, crie mais um pacote com o nome **upload**
5) Sob o pacote *br.eti.amazu.component.upload*, crie as classes

   UploadMode

   Upload
   - Acertar os imports
   - Modificar o método **putScImage**, deixando-o como na listagem abaixo

```
    void putScImage(FileUploadEvent event) throws Exception{
        file = event.getFile();

        scImage = DefaultStreamedContent.builder()
                .contentType(file.getContentType())
                .name(file.getFileName())
                .stream(() -> {
                    try {
                        return file.getInputStream();

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    return null;
                })
                .build();
    }
```

VIDEOAULA 111

LAB 19.2 – Elementos de apoio ao componente Upload

Sequência de passos:

1) Abrir o arquivo pom.xml e adicione a dependência abaixo:

```
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
    <version>1.14</version>
</dependency>
```

2) Executar os seguintes comandos:
Update Project, Clean install, Refresh e Full publish

3) Sob o pacote *br.eti.amazu.infra.util*, crie a classe **FileUtil**

VIDEOAULA 112

LAB 19.3 – Criação do xml do componente Upload

Sequência de passos:

1) Sob a pasta *src/main/webapp/resources/componentes*, crie a página **upload.xhtml**

VIDEOAULA 113

LAB 19.4 – Teste do componente Upload

Sequência de passos:

1) Sob o pacote *br.eti.amazu.infra.view.showcase*, crie a classe **UploadCaseBean**
2) Sob a pasta *src/main/webapp/pages/showcase/primefaces*, crie a página **uploadTest.xhtml**
3) Inicie o servidor
4) No navegador, inicie o aplicativo e abra a página Home Show Case e acione o link da Lição 19, realizando os testes

VIDEOAULA 114

Checklist dos objetivos da Lição 19

A ***Lição 19*** teve por finalidade:

- ☐ Realizar upload de arquivos da máquina local para o servidor de aplicações utilizando o PrimeFaces

Tente repassar cada um desses objetivos, localizando-os nas videoaulas assistidas.

* * * * *

BÔNUS

BÔNUS 1

LOMBOK

Sequência de passos:

1) O que é o plugin Lombok?
2) Baixar o plugin em https://projectlombok.org/download
3) Instalar o plugin
4) Reiniciar o Eclipse (obrigatório)
5) Abrir o pom.xml e adicionar a dependência:

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.12</version>
    <scope>provided</scope>
</dependency>
```

6) Análise das interfaces

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@RequiredArgsConstructor
@EqualsAndHashCode
```

7) Configurar o Lombok em todas as classes criadas, se necessário, mitigando o código e tratando os erros
8) Constatar que o código está livre de erros e warnings
9) Inicie o servidor
10) No navegador, inicie o aplicativo e faça os testes de todas as Lições do curso
11) Desligar o servidor

* * * * *

BÔNUS 2

REFACTORING COM O PLUGIN SONARLINT

Parte 1

1) O que é o Sonar
2) O que é o plugin SonarLint
3) Instalar o plugin SonarLint
4) O arquivo lombok.config
5) O arquivo pom.xml e outras considerações

Parte 2

1) Refactorar a classe HelloServlet
2) Refactorar as classes úteis

Parte 3

Refactorar o componente Dialog

Parte 4

Refactorar o componente Progressbar

Parte 5

1) Refactorar a classe AbstractEntity
2) Refactorar o componente Upload

Parte 6

Refactorar o pacote *infra.view.bean*

Parte 7

Refactorar o pacote *infra.view.showcase*

Parte 8

Refactorar o pacote *infra.view.validator*

Parte 9

Refactorar o pacote *infra.view.vo*

Parte 10

Refactorar os arquivos xhtml

Parte 11

Como debugar com segurança

* * * * *

BÔNUS 3

SONARQUBE

Sequência de passos:

1) Baixar o SonarQube em https://www.sonarqube.org/downloads/
2) Instalar Java11 (arquivo zip)
3) Instalar e testar o SonarQube
4) Configurar o SonarQube no projeto
5) Verificação final com SonarQube e Maven
6) Implantação de testes unitários e cobertura de código

* * * * *

AGRADECIMENTOS FINAIS

Palavras finais do instrutor

* * * * *

FIM