

FIAP GRADUAÇÃO

DIGITAL BUSINESS ENABLEMENT

Prof. Me. Thiago T. I. Yamamoto

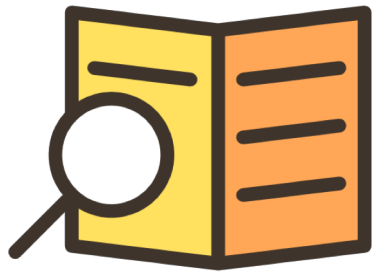
#08 – JSF NAVEGAÇÃO E TEMPLATE



PERCURSO

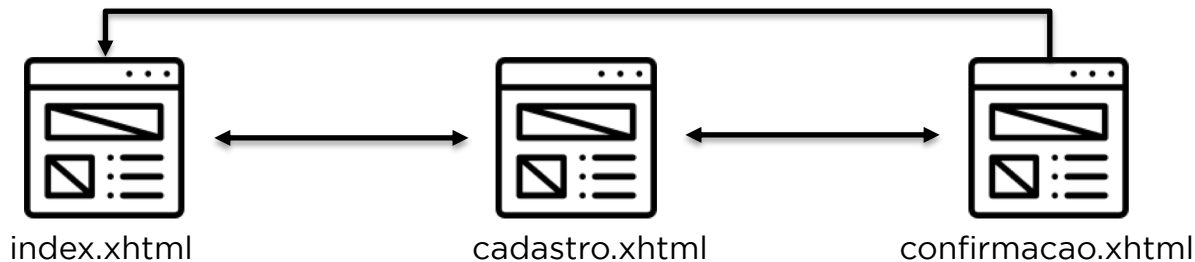
- ✓ Java Application
- ✓ Padrões de Projetos e Frameworks
- ✓ SOA e Web Services
- ✓ Web Services SOAP
- ✓ Web Services Restful
- ✓ JSF - Introdução
- ✓ JSF - Navegação e Template

#08 - AGENDA



- JSF - Navegação
- Navegação implícita
- Navegação explícita
- Redirect x Forward
- Navegação condicional e dinâmica
- Template
- Fragmentos de conteúdo

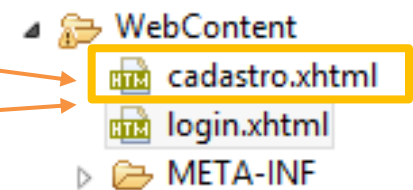
- São responsáveis por **direcionar a navegação** das páginas JSF;
- Existem dois tipos de navegação: **Navegação Implícita** e **Navegação Explícita**.



- Quando um usuário **clica em um botão ou link**, uma **String** (outcome) é enviado para o JSF e será utilizado para a **navegação**, definindo a próxima tela que será exibida;
- Podemos utilizar as tags:
 - **<h:commandButton>** e **<h:commandLink>**, a string de navegação é definido no atributo **action** (necessário h:form);
 - **<h:button>** e **<h:link>** a string de navegação é definida no atributo **outcome**.

```
<h:form>  
  <h:commandButton action="cadastro" value="Cadastrar" />  
</h:form>
```

```
<h:link outcome="cadastro">  
  <h:outputText value="Cadastrar"/>  
</h:link>
```



Mesmo nome da página, sem o .xhtml

- É possível realizar a **navegação após o método do Managed Bean** for executado, de acordo com o **valor retornado** pelo método;

```
<!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:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<h:head></h:head>
<h:body>
  <h:form>
    <h:panelGrid columns="2">
      <h:outputLabel value="Login" for="login" />
      <h:inputText value="#{loginBean.login}" id="login" />
      <h:outputLabel value="Senha" for="senha" />
      <h:inputSecret value="#{loginBean.senha}" id="senha" />
      <h:commandButton action="#{loginBean.logar}" value="Enviar" />
    </h:panelGrid>
  </h:form>
</h:body>
</html>
```



XHTML

- A **string** que é **retornada** do método **logar()** vai determinar a **página** que será **exibida para o usuário**;

```
import javax.faces.bean.ManagedBean;

@ManagedBean
public class LoginBean {

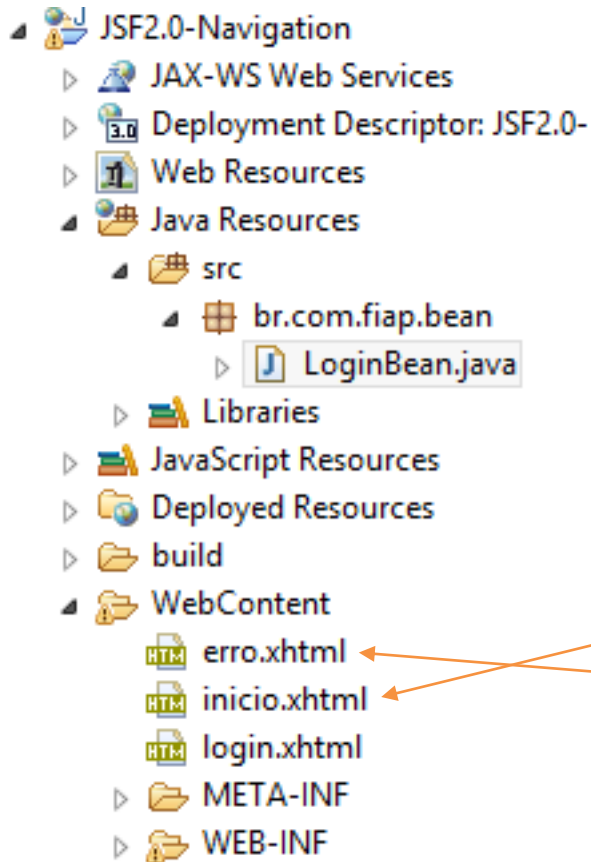
    private String login;
    private String senha;

    public String logar() {
        if ("thiago".equals(login) && "fiap".equals(senha))
            return "inicio";
        else
            return "erro";
    }

    /*gets e sets*/
}
```

Java

- O **botão** invoca o método **logar()** do managed bean e o retorno do método (inicio ou erro) determina a **página que será exibida**;



```
<h:commandButton action="#{loginBean.logar}"  
value="Enviar" />
```

XHTML

```
public String logar() {  
    if ("thiago".equals(login)  
        && "fiap".equals(senha))  
        return "inicio";  
    else  
        return "erro";  
}
```

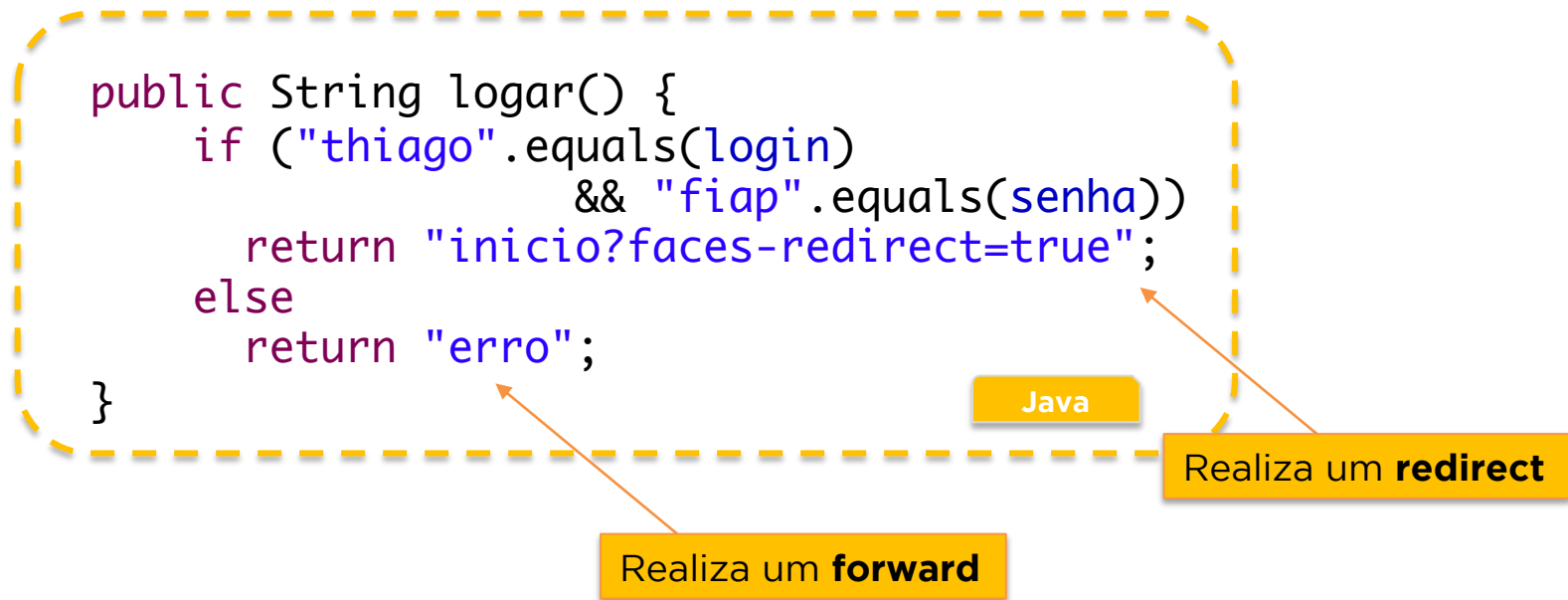
Java

- Um **redirect** é uma nova requisição que o cliente (browser) faz a pedido da aplicação web, logo ele fica ciente sobre como está ocorrendo a navegação e para onde ele está sendo redirecionado;
- Um **forward** pode executar várias requisições no lado servidor sem o conhecimento do cliente;
- Um **forward** mantém os atributos e parâmetros do request original, já um redirect não;

Com **forward**, após um cadastro, caso o usuário atualize a página (F5) a requisição original será processada novamente, ou seja, o cadastro será realizado novamente! Então, sempre após um **POST** devemos realizar um **redirect**!



- Para determinar se a navegação será realizada por **redirect** ou **forward**, podemos utilizar o parâmetro **faces-redirect**;
- **Por padrão** o faces **redirect** é **falso**, ou seja, realiza um **forward**;



- Configurado no arquivo de configuração do JSF: ***faces-config.xml***
- Para registrar uma regra de navegação precisamos:
 - **Tela de origem**
 - **Outcome (String de navegação)**
 - **Tela de destino**

```
<navigation-rule>
  <from-view-id>/login.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>correto</from-outcome>
    <to-view-id>/inicio.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>errado</from-outcome>
    <to-view-id>/erro.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

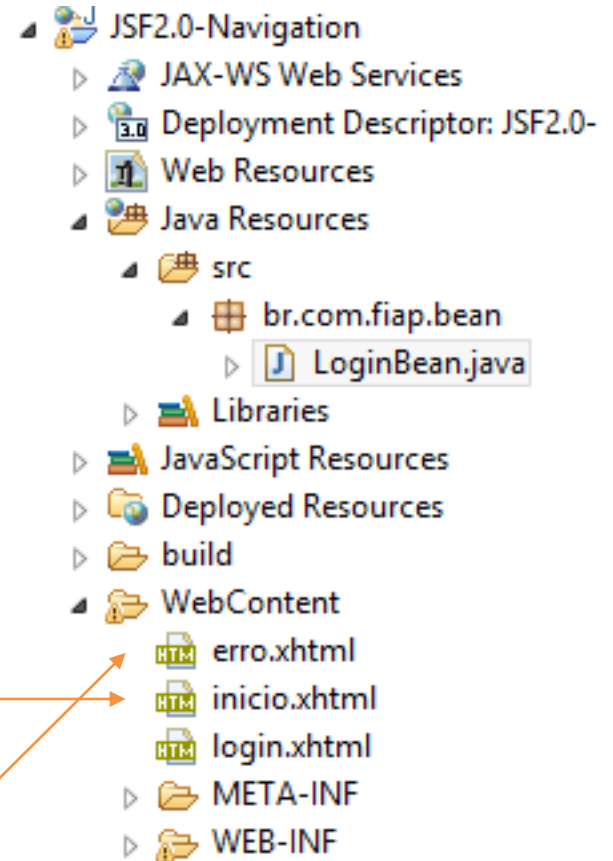
Faces-config.xml

```
public String logar() {  
    if ("thiago".equals(login)  
        && "fiap".equals(senha))  
        return "correto";  
    else  
        return "errado";  
}
```

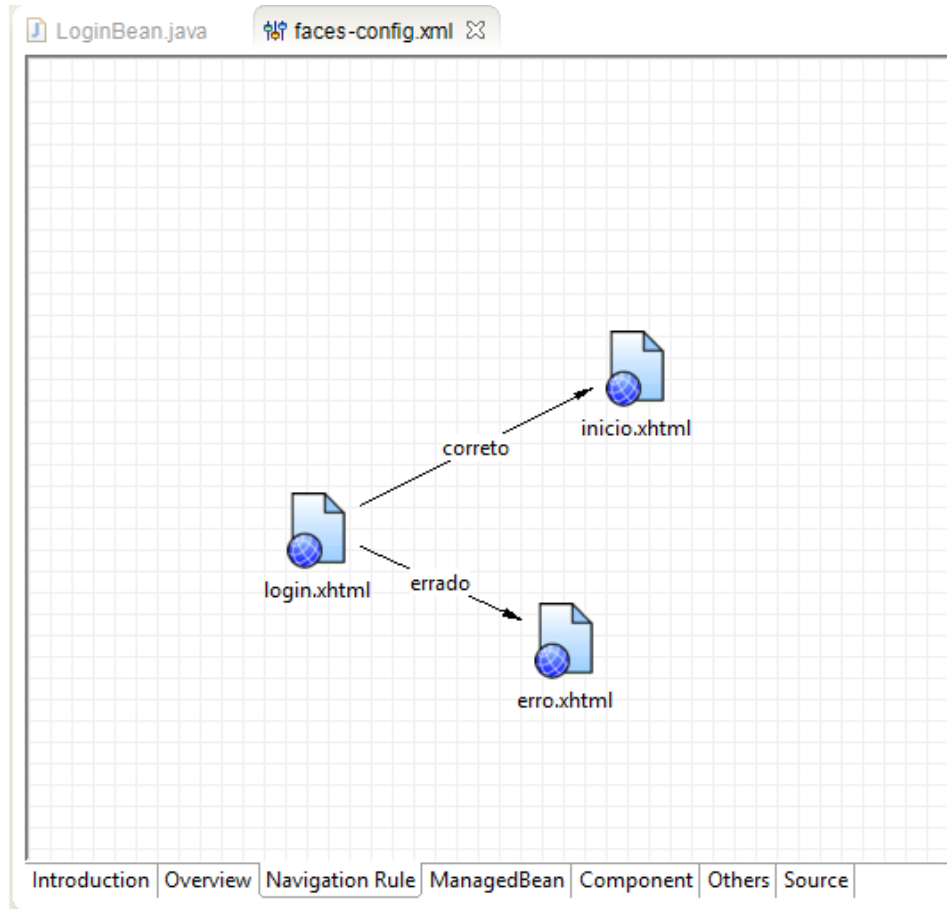
LoginBean.java

```
<navigation-rule>  
    <from-view-id>/login.xhtml</from-view-id>  
    <navigation-case>  
        <from-outcome>correto</from-outcome>  
        <to-view-id>/inicio.xhtml</to-view-id>  
    </navigation-case>  
    <navigation-case>  
        <from-outcome>errado</from-outcome>  
        <to-view-id>/erro.xhtml</to-view-id>  
    </navigation-case>  
</navigation-rule>
```

Faces-config.xml



- É possível configurar de forma **visual** a **navegação**:



Markers Properties Servers D:

Link

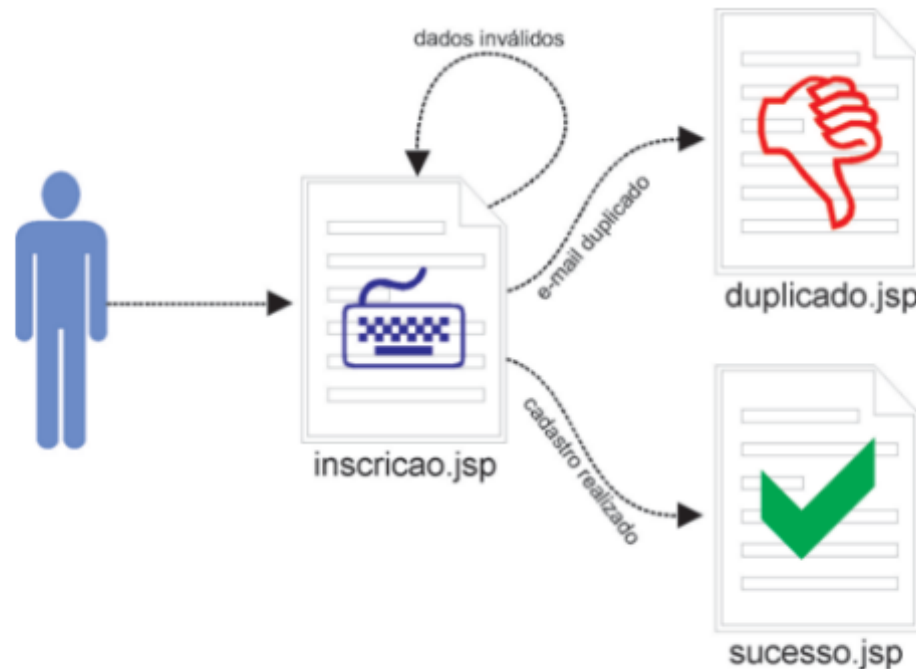
| | | |
|------------|---------------|--------|
| Quick Edit | From Action: | |
| Attributes | From Outcome: | errado |
| | Redirect: | false |

Markers Properties Servers I

Link

| | | |
|------------|---------------|---------|
| Quick Edit | From Action: | |
| Attributes | From Outcome: | correto |
| | Redirect: | false |

- No JSF podemos utilizar mais **2 mecanismos** de navegação:
 - **Navigation Rule com EL (Expression Language);**
 - **Navigation Rule com CASE (IF);**



- **Dois métodos** do managed bean retornam **a mesma string**;
- É necessário criar uma regra para a navegação correta, utilizando a tag **<from-action>**;

```
<navigation-rule>
  <from-view-id>/confirma.xhtml</from-view-id>
  <navigation-case>
    <from-action>
      #{inscricaoBean.inscrever}
    </from-action>
    <from-outcome>sucesso</from-outcome>
    <to-view-id>/sucesso.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-action>
      #{inscricaoBean.cancelarInscricao}
    </from-action>
    <from-outcome>sucesso</from-outcome>
    <to-view-id>/cancelado.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Faces-config.xml

- A navegação é lida de **cima para baixo**, assim a primeira condição encontrada é utilizada na navegação.

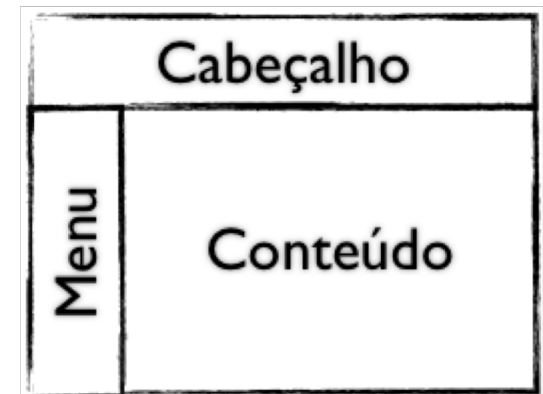
```
<navigation-rule>
  <from-view-id>/confirma.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>index</from-outcome>
    <if>#{clienteBean.novoCadastro}</if>
    <to-view-id>/cadastro.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>index</from-outcome>
    <to-view-id>/index.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

Faces-config.xhtml



TEMPLATE

- Templates são recursos em que **reaproveitamos parte do código de uma página** para elementos que são repetitivos;
- Esta técnica é amplamente utilizada para montagem de **layout**;
- Esta técnica divide uma página em 2 partes:
 - **Template**
 - **Fragmentos de Conteúdo**
- Os componentes de templates estão na biblioteca **Facelet Core**;

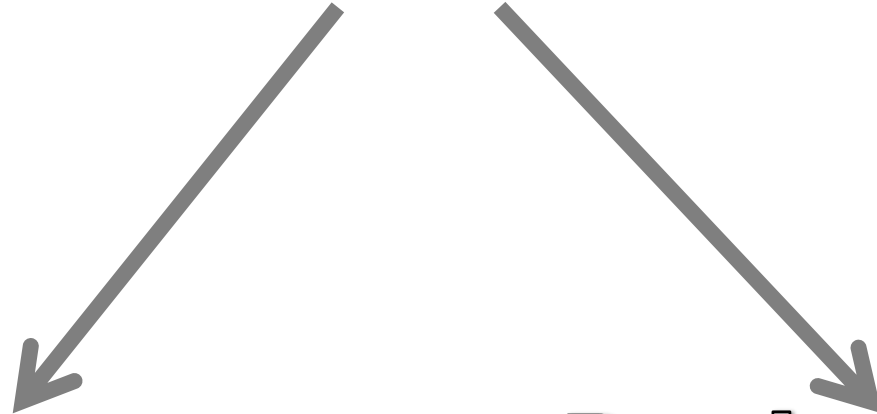


Facelets Template

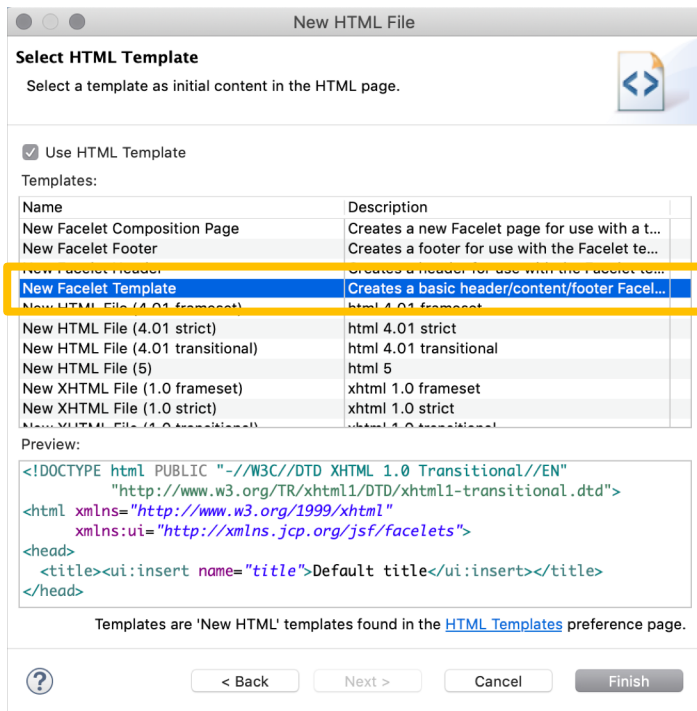
_template.xml

Pagina1.xhtml

Pagina2.xhtml



- Possui a **estrutura completa** da página HTML, definindo o **conteúdo comum** a todas as páginas;
- Utiliza a tag **<ui:insert>** para permitir que as outras páginas **adicionem conteúdo**;
- Por padrão, páginas incompletas começam com “_”;




Para criar um **template**, crie um arquivo **XHTML** com o nome **_template.xhtml** e escolha o template “**New Facelet Template**”;

- No exemplo, o arquivo **_template.xhtml** possui duas áreas para as **outras páginas adicionarem conteúdo**: título e corpo;

```
<!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:ui="http://xmlns.jcp.org/jsf/facelets">
<head>
    <title><ui:insert name="titulo">Título</ui:insert></title>
</head>

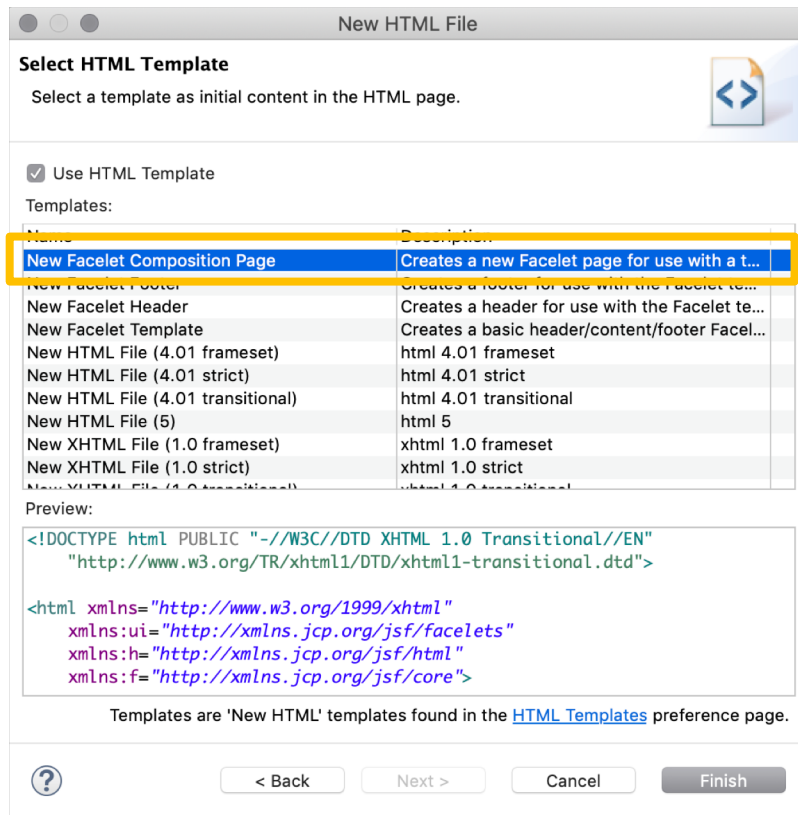
<body>
    <ui:insert name="corpo"></ui:insert>
</body>
</html>
```



XHTML

FRAGMENTOS DE CONTEÚDO

- Utiliza o **template** e **define o conteúdo**, através da tag **<ui:define>**, que será adicionado às **áreas do template**:



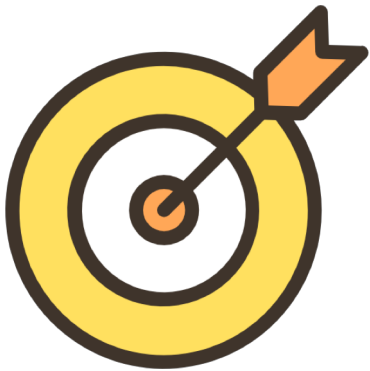
Para criar uma **página** que utiliza o template, crie um arquivo **XHTML** e escolha o template **“New Facelet Composition Page”**;

- A tag **<ui:composition>** define o *template* que será utilizado na página:

```
<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">

  <ui:composition template="_template.xhtml">
    <ui:define name="titulo">
      Exemplo de título
    </ui:define>
    <ui:define name="corpo">
      <!-- Código... -->
    </ui:define>
  </ui:composition>
</html>
```


VOCÊ APRENDEU...



- Realizar **navegação** entre **páginas** JSF e através dos **métodos** no managed bean;
- Navegação **implícita** e **explícita**;
- Implementar a navegação com **redirect** ou **forward**;
- Utilizar **templates** para evitar código repetido e manter a consistência visual das páginas;

Copyright © 2013 – 2019

Prof. Me. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

“Para conquistar o sucesso, você precisa aceitar todos os desafios que vierem na sua frente. Você não pode apenas aceitar os que você preferir”
- Mike Gafka