# PROGRAMAÇÃO WEB

Java EE com Java Server Faces

# Facelets

- É uma linguagem de declaração de páginas que é utilizada para :
  - Construir telas JSF utilizando templates HTML;
  - Construir árvores de componentes.
- As principais vantagens do Facelets são:
  - Utilização de xhtml para a criação de páginas web;
  - Suporte para as tag libs do Facelets além das tags do JSF e JSTL;
  - Suporte para a Expression Language(EL);
  - Criação de templates para componentes e páginas.

# Facelets

| Tag | Função |
|---|---|
| ui:component | Define um componente que é criado e adicionada na árvore de componentes. |
| ui:composition | Define uma composição de página que pode utilizar um template, opcionalmente. Desconsidera o conteúdo fora da tag. |
| ui:debug | Define um componente de debug que é criado e adicionado na árvore de componentes. |
| ui:decorate | Semelhante a tag composition mas não desconsidera o contéudo for a da tag. |
| ui:define | Define conteúdo que é inserido dentro de uma página por um template. |
| ui:fragment | Semelhante a tag composition mas não desconsidera o contéudo for a da tag. |
| ui:include | Encapsula e reusa o conteúdo para múltiplas páginas. |
| ui:insert | Insere conteúdo no template. |
| ui:param | Utiliazado para passar paramêtros em um arquivo incluso. |
| ui:repeat | Utilizado como alternativa para tags de loop como c:forEach ou h:dataTable. |
| ui:remove | Remove conteúdo de uma página. |

# Facelets

```
<!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">

  <h:head>
    <meta http-equiv="Content-Type"
        content="text/html; charset=UTF-8" />
    <h:outputStylesheet library="css“
name="default.css"/>
    <h:outputStylesheet library="css"
name="cssLayout.css"/>
    <title>Facelets Template</title>
  </h:head>

  <h:body>
    <div id="top" class="top">
      <ui:insert name="top">Top Section</ui:insert>
    </div>
    <div>
    <div id="left">
      <ui:insert name="left">Left Section</ui:insert>
    </div>
    <div id="content" class="left_content">
      <ui:insert name="content">Main
Content</ui:insert>
    </div>
    </div>
  </h:body>
</html>
```

# Facelets

```
<!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">

    <h:body>
        <ui:composition template="./template.xhtml">
            <ui:define name="top">
                Welcome to Template Client Page
            </ui:define>

            <ui:define name="left">
                <h:outputLabel value="You are in the Left Section"/>
            </ui:define>

            <ui:define name="content">
                <h:graphicImage value="#{resource['images:wave.med.gif']}"/>
                <h:outputText value="You are in the Main Content Section"/>
            </ui:define>
        </ui:composition>
    </h:body>
</html>
```

# Facelets

| Tag | Função |
|---|---|
| composite:interface | Declara o contrato de uso do componente. |
| composite:implementation | Define a implementação do componente. Se uma elemento composite:interface for declarado, a declaração desta tag é obrigatória. |
| composite:attribute | Decalra um atributo que pode ser utilizado no componente. |
| composite:insertChildren | Qualquer componente filho ou texto de template terá o pai alterado a partir da declaração desta tag. |
| composite:valueHolder | Declara que o component expõe uma implementação da interface ValueHolder que pode ser utilizado como alvo de objetos vinculados na página. |
| composite:editableValueHolder | Declara que o component expõe uma implementação da interface EditableValueHolder que pode ser utilizado como alvo de objetos vinculados na página. |
| composite:actionSource | Declara que o component expõe uma implementação da interface ActionSource que pode ser utilizado como alvo de objetos vinculados na página. |

# Facelets

```
<!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:composite="http://java.sun.com/jsf/composite"
  xmlns:h="http://java.sun.com/jsf/html">

    <h:head>
        <title>This content will not be displayed</title>
    </h:head>
    <h:body>
        …
    </h:body>
</html>
```

```
<composite:interface>
        <composite:attribute
         name="value"
         required="false"/>
</composite:interface>
<composite:implementation>
        <h:outputLabel
              value="Email id: “/>
      <h:inputText
            value="#{cc.attrs.value}“/>
</composite:implementation>
```

# Facelets

```
<XML…>
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"

xmlns:em="http://java.sun.com/jsf/composite/emcomp/">
    <h:head>
        <title>Using a sample composite
                component</title>
    </h:head>
    <body>
        <h:form>
            <em:email value="Insira seu e-mail" />
        </h:form>
    </body>
</html>
```

- Repare na utilização de cc.attrs.value para a definição do valor do inputText. A palavra cc é **reservada** no JSF para componentes composite.
- A expressão #{cc.attrs.*attribute-name*} é utilizada para acessar os atributos definidos pela interface do componente, que no caso é o atributo value.

# Expression language (EL)

- A EL provê um importante mecanismo para habilitar a comunicação entre a camada de apresentação (páginas web) e a camada de aplicação (managed beans)
- A EL é usa tanto pelo JSF quanto pelos JSP e Facelets.
- Ela representa a união das linguagens de expressão oferecidas pelo JSF e JSP.

# Expression language (EL)

- JSF utiliza a EL para:
  - Avaliar expressões;
  - Recuperar ou alterar dados;
  - Acionar métodos.
- A EL pode acessar:
  - Componentes JavaBeans;
  - Coleções;
  - Enumerações;
  - Objetos implícitos na página.

# Expression language (EL)

- Exemplos:

${customer}

public enum Suit {hearts, spades, diamonds, clubs}

${mySuit == "hearts"}

${customer.address["street"]}

${customer.age + 20}

${"literal"}

${customer.orders[1]}

# Expression language (EL)

- Para referenciar um atributo de uma classe pode se utilizar o . ou [] indicando a propriedade:
  - ${customer.name} ou ${customer["name"]} tem o mesmo efeito.
  - Vale ressaltar que a classe acessada deve seguir o padrão Java Beans.
- Para acessar um array ou ArrayList:
  - ${customer.orders[1]}

# Expression language (EL)

- Para acessar um item de uma map uma String deve ser parametrizada:
  - ${customer.orders["socks"]}

# Expression language (EL)

- A EL pode ser utilizada em:
  - Texto estático;
  - Qualquer atributo de uma tag customizada custom tag que aceite expressão.

```
<some:tag>
some text ${expr} some text
</some:tag>
<some:tag value="${expr}"/>
<another:tag value="#{expr}"/>
```

# Expression language (EL)

- A EL também pode seu utilizada em métodos:

```
<h:form>
    <h:inputText
        id="name"
        value="#{customer.name}"
        validator="#{customer.validateName}"/>
    <h:commandButton
        id="submit"
        action="#{customer.submit}" />
</h:form>
```

# Expression language (EL)

- As expressões de métodos só podem ser utilizadas em atributos de tags e nas seguintes condições:
- Com uma simples expressão onde o bean referenciado seja um JavaBean e o método referenciado seja um método deste JavaBean:

```
<some:tag value="#{bean.method}"/>
```

# Expression language (EL)

- A expressão avalia para um método que é passado para uma tratador de tag. O método representado pela expressão pode então ser acionado posteriormente.

```
<some:tag value="sometext"/>
```

- Métodos parametrizados também podem ser acionados:

*expr-a*[*expr-b*](*parameters*)

*expr-a.identifier-b*(*parameters*)

```
<h:inputText
    value="#{userNumberBean.userNumber('5')}">
<h:commandButton
    action="#{trader.buy('SOMESTOCK')}"
    value="buy"/>
```

# Expression language (EL)

- Operadores também podem ser utilizados. Os tipos de operadores permitidos são:
  - **Aritméticos**: +, - (binário), *, / e div, % e mod, - (unário)
  - **Lógicos**: and, &&, or, ||, not, !
  - **Relacionais**: ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le. Comparações podem ser realizadas contra outros valores ou contra valores Boolean, string, integer, ou literais ponto-flutuante
  - **Empty**: O operador empty é uma operação pré-fixa que pode ser utilizada para determinar quando um valor é nulo ou vazio.
  - **Conditional**: A ? B : C. Avalia B ou C, dependendo do resultado da avaliação de A.

# Expression language (EL)

- A precedência dos operadores é a seguinte (do maior para o menor):
  - [] .
  - () (utilizado para alterar a precedência dos operadores)
  - - (unário) not ! empty
  - * / div % mod
  - + - (binário)
  - < > <= >= lt gt le ge
  - == != eq ne
  - && and
  - || or
  - ? :

# Expression language (EL)

- Os termos a seguir são palavras reservadas da EL:

| | | | |
|---|---|---|---|
| and | or | not | eq |
| ne | lt | gt | le |
| ge | true | false | null |
| instanceof | empty | div | mod |

# Expression language (EL)

| EL Expression | Resultado |
|---|---|
| ${1 > (4/2)} | false |
| ${4.0 >= 3} | true |
| ${100.0 == 100} | true |
| ${(10*10) ne 100} | false |
| ${'a' < 'b'} | true |
| ${'hip' gt 'hit'} | false |
| ${4 > 3} | true |
| ${1.2E4 + 1.4} | 12001.4 |
| ${3 div 4} | 0.75 |
| ${10 mod 4} | 2 |
| ${!empty param.Add} | False se o paramêtro de requisição Add é nulo ou uma String vazia. |
| ${pageContext.request.contextPath} | O caminho do contexto. |
| ${sessionScope.cart.numberOfItems} | O valor da propriedade numberOfItems do atributo de sessão cart. |
| ${param['mycom.productId']} | O valor do parâmetro de requisição mycom.productId. |
| ${header["host"]} | O host. |
| ${departments[deptName]} | O valor da entrada deptName no mapa departments. |
| ${requestScope['javax.servlet.forward.servlet_path']} | O valor do atributo de requisição javax.servlet.forward.servlet_path. |
| #{customer.lName} | O valor da propriedade lName do bean customer durante uma requesição inicial. Altera o valor de lName durante uma requisição postback. |
| #{customer.calcTotal} | O valor de retorno do método calcTotal do bean customer. |

# Seam - Segurança

- O Seam suporta a in'tegração de medidas de segurança nas aplicações.
- Ele facilita a adição de autenticação e autorização na aplicação.
- Existem alguns aspectos que devem ser considerados em segurança:
  - Gerenciamento de identidade;
  - Encriptação de dados;
  - Detecção de intrusos;
  - Etc.
- O Seam foca nas necessidade mais comuns e o resto pode ser integrado através de outros meios.

# Seam - Segurança

- Autenticação: O Seam facilita a injeção da funcionalidade de login na aplicação onde é necessário identificar o usuário e/ou verificar se tem o acesso a determinada página, função ou dados.
- O Seam permite autenticação em diferentes níveis (página, controle JSF, componente, método de ação) utilizando arquivos de configuração e/ou anotações

# Seam - Segurança

- O processo de login pode ser implementado utilizando componentes padrão do Seam e forms JSF Em conjunção com os componentes providos pelo.
- Além dos componentes pré-existentes o Seam define componentes que suportam o tratamento da identidade do usuário para facilitar o acesso com o modelo de componentes contextuais do Seam.

# Seam - Segurança

- Autorização: Seam permite a autorização em vários níveis.
- É possível definir a autorização para grupos de páginas, páginas únicas, componentes específicos, etc.
- Os acessos são especificados utilizando papéis e/ou permissões.
- Papéis no Seam são simples atribuições de grupos, como por exemplo: Gerente, departamento-vendas, etc.

# Seam - Segurança

- Permissões são ações que podem ser executadas em geral ou para entidade específicas.
- Como na autenticação é possível especificar regras de acesso para entidades utilizando arquivos de configuração e/ou anotações.
- Ainda é possível utilizar autorizações mais avançadas utilizando o conjunto de regras JBoss Rules.

# Seam - Segurança

- Para realizar a configuração de segurança é necessário adicionar o componente security:identity no arquivo components.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<components xmlns:security="http://jboss.com/products/seam/security"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://jboss.com/products/seam/security
http://jboss.com/products/seam/security-1.2.xsd">
. . .
<security:identity authenticate-method="#{authenticator.login}"/>
. . .
</components>
```

# Seam - Segurança

- O método do bean que realiza a segurança deve:
- Não receber parâmetro;
- Retornar um valor booleano indicando quando as credenciais foram aceitas;
- Ser acessível via EL ( o que não é um problema utilizando o Seam)

# Seam - Segurança

- O método login do bean authenticator é responsável por realizar a autenticação do usuário na aplicação.
- Para acionar a autenticação o bean identity do Seam deve ser acionado de um formulário JSF.

```
<h:form>
        …
          <h:commandButton id="submit"
                  value="Login" action="#{identity.login}"/>
        …
</h:form>
```

# Seam - Segurança

- O componente identity utilizado no formulário JSF no exemplo anterior é o mesmo que foi configurado no arquivo components.xml!