



Introdução ao Java Server Faces

Prof. Leonardo Vianna do Nascimento
Disc. de Desenvolvimento de Sistemas I

Framework???

- Um conjunto de classes que serve como base para construção de aplicações de um determinado tipo
 - Muitas ações já se encontram implementadas
 - O que é necessário implementar é o que é específico de cada aplicação

JSF

- *Java Server Faces*
 - Especificação para um framework de componentes para desenvolvimento web em Java
 - O framework permite a construção de páginas web através de componentes e a conexão destes com objetos de negócio
 - Por exemplo, se desejamos mostrar uma tabela
 - Utilizando Servlets e JSP precisaríamos fazer um *loop* para gerar as tags HTML para linhas e colunas
 - Utilizando JSF basta adicionar um componente para tabela de dados

Componentes JSF

Servlets + JSP

```
<table>
<%
    for(String item: lista) {
        out.println("<tr><td>" + item + "</td></tr>");
    }
%>
</table>
```

JSF

```
<h:dataTable value="#{bean.lista}" var="item">
    <h:column>#{item}</h:column>
</h:dataTable>
```

Partes do JSF

- Um conjunto de componentes de interface com o usuário
 - Tabelas, botões, entrada de texto, etc.
- Um modelo de programação orientado a eventos
- Um modelo de componentes que permite o desenvolvimento de componentes adicionais

Implementações do JSF

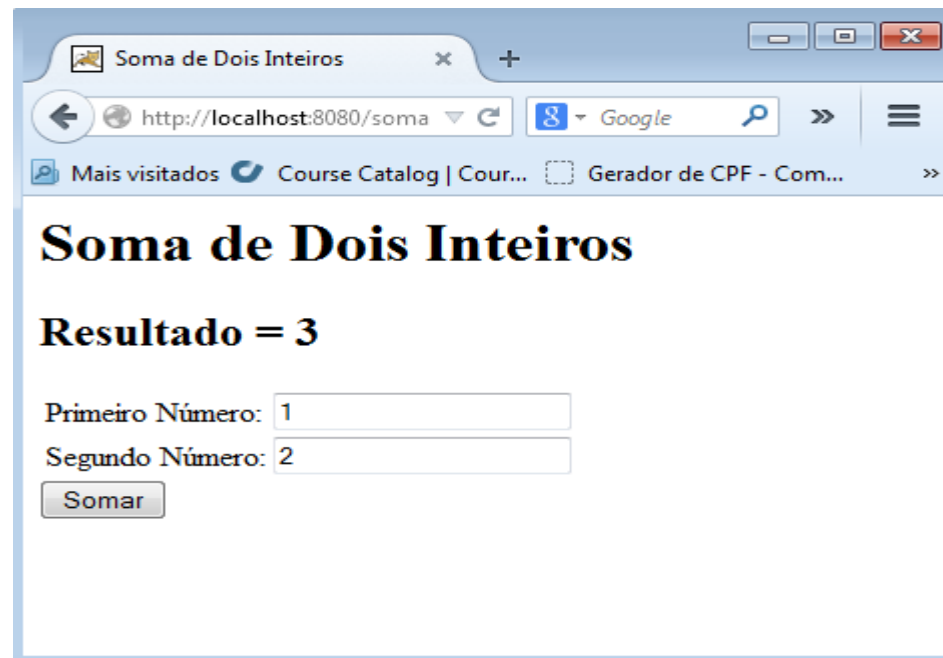
- O JSF por si só é apenas uma especificação
- Para construir uma aplicação usando JSF, é necessário fazer o download de uma de suas implementações
 - Sun Mojarra (antigo JSF Reference Implementation – JSF RI)
 - <http://javaserverfaces.java.net/>
 - Apache MyFaces
 - <http://myfaces.apache.org/>

Recursos Adicionais

- Existem grupos e empresas que desenvolveram componentes adicionais para o JSF:
 - Apache
 - Tomahawk, Trinidad
 - JBoss
 - RichFaces
 - ICESoft
 - ICEFaces
 - PrimeFaces

Um Exemplo Simples

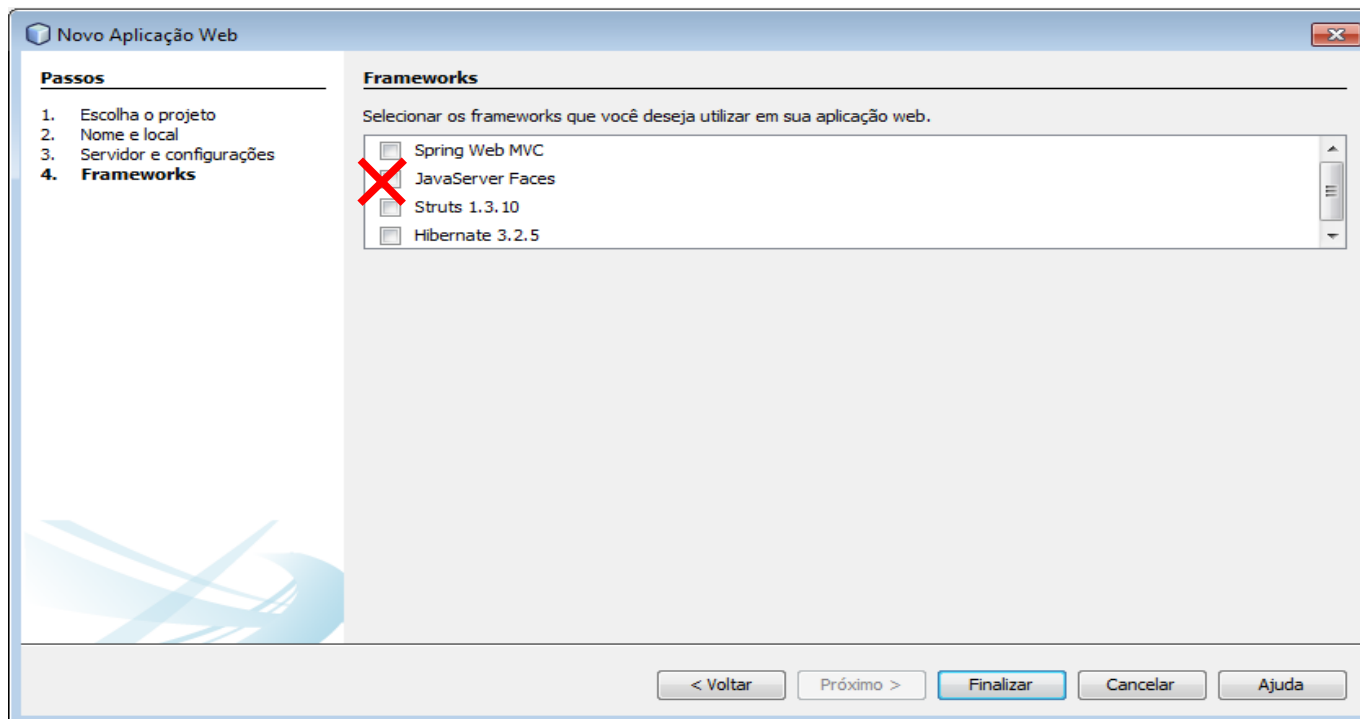
- Vamos implementar a aplicação de soma de números inteiros da última aula
- Nesta aplicação o usuário deve informar dois números inteiros e pode calcular a soma deles



The screenshot shows a web browser window with the title "Soma de Dois Inteiros". The address bar displays "http://localhost:8080/soma". The page content includes the title "Soma de Dois Inteiros" in bold, followed by "Resultado = 3". Below this, there are two input fields: "Primeiro Número:" with the value "1" and "Segundo Número:" with the value "2". A "Somar" button is located below the input fields.

Criando um Projeto JSF (1)

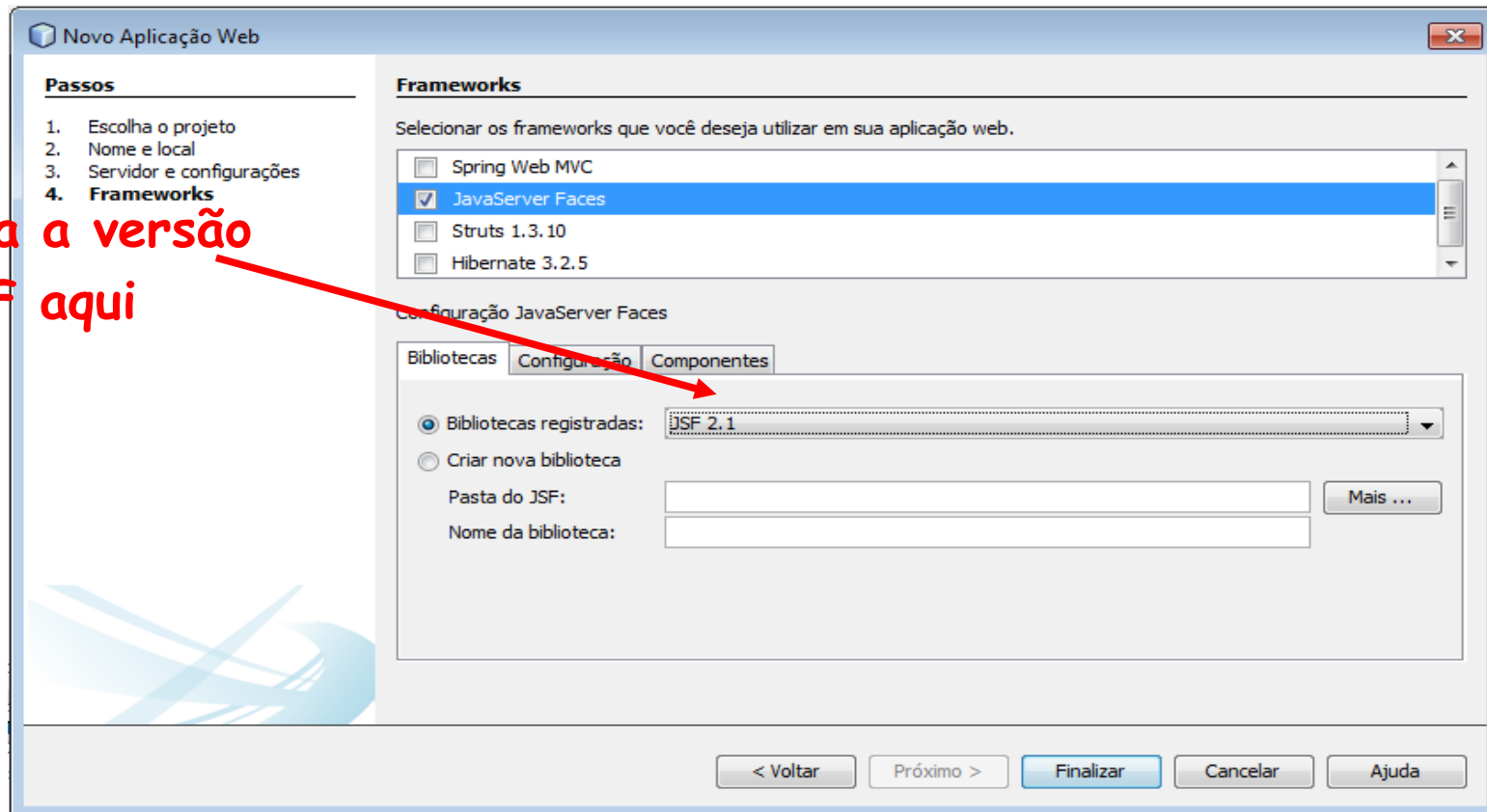
- No NetBeans, basta criar um novo projeto Java Web
- Após passar pelas etapas “Nome e Local”, “Servidor e Configurações”, deve-se selecionar o framework JSF na tela que aparecer, como abaixo



Criando um Projeto JSF (2)

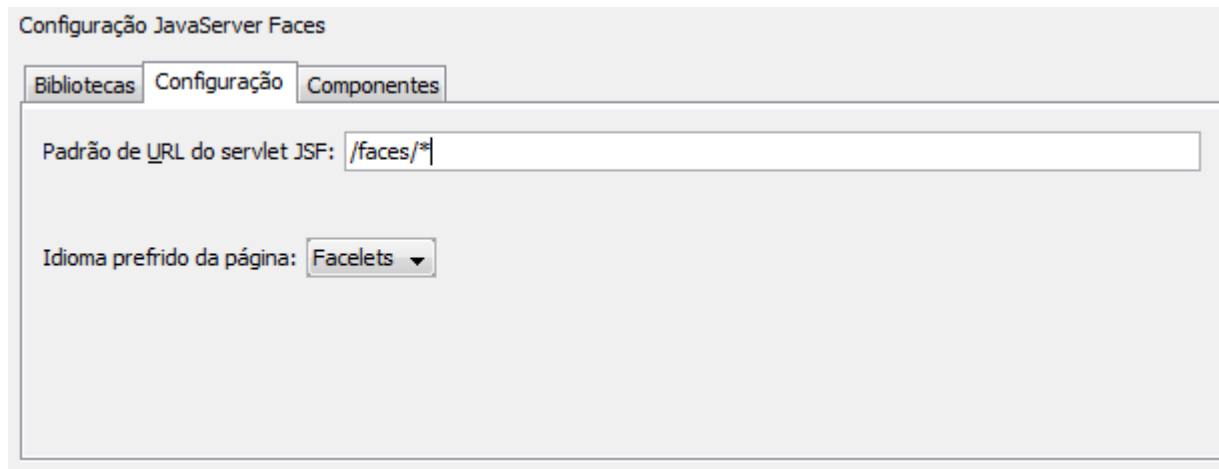
- Após selecionar o framework JSF, selecione a versão desejada
 - Nesta disciplina estaremos trabalhando com a versão 2.1

Escolha a versão do JSF aqui



Criando um Projeto JSF (3)

- Na aba “Configurações” é possível especificar o padrão de URL para acesso ao servlet do JSF e o “Idioma Preferido da Página”
- O “Idioma Preferido da Página” permite dizer se trabalharemos com o JSF usando o modelo JSP ou o modelo Facelets



Arquivos do Exemplo

- Você pode baixar o exemplo no Moodle
- O exemplo contém três arquivos
 - *SomaBean.java*
 - Classe da camada de modelo
 - *index.xhtml*
 - Arquivo XHTML da camada de apresentação (*view*)
 - *web.xml*
 - Arquivo de configuração da aplicação

Apresentação: *index.xhtml*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <html xmlns="http://www.w3.org/1999/xhtml"
3       xmlns:h="http://java.sun.com/jsf/html">
4   <h:head>
5       <title>Soma de Dois Inteiros</title>
6   </h:head>
7   <h:body>
8       <h1>Soma de Dois Inteiros</h1>
9       <h:messages/>
10      <h2>Resultado = #{bean.soma}</h2>
11      <h:form>
12          <table>
13              <tr>
14                  <td>Primeiro Número: </td>
15                  <td><h:inputText value="#{bean.numero1}"/></td>
16              </tr>
17              <tr>
18                  <td>Segundo Número: </td>
19                  <td><h:inputText value="#{bean.numero2}"/></td>
20              </tr>
21          </table>
22          <h:commandButton value="Somar" action="index"/>
23      </h:form>
24  </h:body>
25</html>
```

Apresentação: *index.xhtml*

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <html xmlns="http://www.w3.org/1999/xhtml"
3     xmlns:h="http://java.sun.com/jsf/html">
4     <h:head>
5         <title>Soma de Dois Inteiros</title>
6     </h:head>
7     <h:body>
8         <h1>Soma de Dois Inteiros</h1>
9         <h:messages/>
10        <h2>Resultado = #{bean.soma}</h2>
11        <h:form>
12            <table>
13                <tr>
14                    <td>Primeiro Número: </td>
15                    <td><u><h:inputText value="#{bean.numero1}"/></td>
16                </tr>
17                <tr>
18                    <td>Segundo Número: </td>
19                    <td><h:inputText value="#{bean.numero2}"/></td>
20                </tr>
21            </table>
22            <h:commandButton value="Somar" action="index"/>
23        </h:form>
24    </h:body>
25</html>
```

Tags começando com
o prefixo *h* são
componentes JSF

Apresentação: *index.xhtml*

- Tags HTML padrão foram usadas
 - *html*, *title*, *table*, entre outras
- As que começam com o prefixo *h* são tags da biblioteca HTML do JSF
 - Incluímos essa biblioteca de tags na página com a entrada
 - `xmlns:h="http://java.sun.com/jsf/html"`
 - Todas as tags devem iniciar com o identificador *h*:
 - Poderia ser outro se trocássemos o *h* em *xmlns* pelo prefixo desejado

View em Aplicações JSF

- Nas primeiras versões do JSF (1.0, 1.1 e 1.2) as páginas da camada *view* deveriam ser feitas usando JSP e as tags específicas das bibliotecas JSF
- Assim, o JSF funcionava em cima do JSP

JSF 2.0

- Na versão 2.0, a tecnologia Facelets tornou-se o padrão para a construção de páginas JSF
 - Com esse novo padrão, as páginas JSF devem ser escritas em arquivos XHTML
- Os arquivos XHTML são arquivos XML com conteúdo HTML
 - Todas as tags seguem o padrão XML

Tags JSF (1)

- *h:head*
 - Componente JSF equivalente à tag *head* do HTML
- *h:body*
 - Componente JSF que equivale à tag *body*
- Por que usar esses componentes?
 - Ao processar essas páginas, o JSF insere código HTML, e Javascript importantes para o processamento da página
 - Se usarmos as tags HTML normais esses códigos não serão inseridos e alguns componentes JSF podem não funcionar

Tags JSF (2)

- *h:form*
 - Componente JSF equivalente à tag *form* do HTML
- *h:inputText*
 - Componente JSF equivalente à tag *input* do tipo *text* do HTML
 - A propriedade *value* é geralmente associada diretamente a uma propriedade de um *bean* na camada de modelo

Tags JSF (3)

- *h:commandButton*
 - Gera um botão que executa uma ação qualquer na página
 - O atributo *value* especifica o texto que aparecerá no botão
 - No atributo *action* especificamos a página que deve ser exibida ao se clicar no botão
 - Em nosso exemplo, *index.xhtml*
- *h:messages*
 - Exibe uma lista de mensagens retornada pelo controle

Classe *SomaBean*

- Classe da camada de modelo
- É um *Java Bean*
 - É uma classe Java comum que segue as seguintes especificações:
 - Deve possuir um construtor sem parâmetros
 - Possuem propriedades especificadas pelos nomes dos métodos *get* e *set*
 - Por exemplo, se uma classe possui os métodos *getNome* e *setNome*, então o bean possui uma propriedade de leitura/escrita chamada *nome*

Classe *SomaBean*

```
8 public class SomaBean {  
9     private int numero1;  
10    private int numero2;  
11  
12    public int getNumero1() {  
13        return numero1;  
14    }  
15  
16    public void setNumero1(int numero1) {  
17        this.numero1 = numero1;  
18    }  
19  
20    public int getNumero2() {  
21        return numero2;  
22    }  
23  
24    public void setNumero2(int numero2) {  
25        this.numero2 = numero2;  
26    }  
27  
28    public int getSoma() {  
29        return numero1 + numero2;  
30    }  
31 }
```

Classe *SomaBean*

- Propriedades do *bean*
 - *numero1*
 - Especificada pelos métodos *getNumero1* e *setNumero1*
 - Leitura/escrita
 - *numero2*
 - Especificada pelos métodos *getNumero2* e *setNumero2*
 - Leitura/escrita
 - *soma*
 - Especificada pelo método *getSoma*
 - Somente leitura

Managed Beans

- A classe *SomaBean* é um *managed bean*
- É uma classe cujos objetos podem ser manipulados em páginas da camada de apresentação

O campo de entrada preencherá a propriedade *numero1* do objeto bean da classe *SomaBean*

```
<td>Primeiro Número: </td>
```

```
<td><h:inputText value="#{bean.numero1}"/></td>
```

O valor da propriedade *soma* será mostrado

```
<h2>Resultado = #{bean.soma}</h2>
```


Configurando o Managed Bean

- Para que o *managed bean* que criamos possa ser referenciado em nossas páginas de apresentação, precisamos configurá-lo
- A forma mais simples de se fazer isso é através de duas anotações em Java, colocadas antes da linha com *public class* da classe:

```
@ManagedBean(name = "bean")
```

```
@RequestScoped
```

Configurando o Managed Bean

- A linha `@ManagedBean (name = "bean")` define que nosso MB será chamado na apresentação de *bean*
- A anotação `@RequestScoped` indica que nosso MB será um bean de requisição, ou seja, será criado ao se receber uma solicitação HTTP vinda do formulário de inclusão e será destruído quando a resposta for enviada
- Outras opções:
 - `@SessionScoped`
 - `@ApplicationScoped`

Configurando o Managed Bean

```
1 package modelo;
2
3 import javax.faces.bean.ManagedBean;
4 import javax.faces.bean.RequestScoped;
5
6 @ManagedBean(name="bean")
7 @RequestScoped
8 public class SomaBean {
9     private int numero1;
10    private int numero2;
11
12    public int getNumero1() {
13        return numero1;
14    }
15
16    public void setNumero1(int numero1) {
17        this.numero1 = numero1;
18    }
19
20    public int getNumero2() {
21        return numero2;
22    }
23
24    public void setNumero2(int numero2) {
25        this.numero2 = numero2;
```

Estes imports devem ser feitos para que as anotações funcionem

Configurando o Managed Bean

```
6 @ManagedBean (name="bean")  
7 @RequestScoped
```

O nome usado na
configuração do bean é o
mesmo que usamos para
identificá-lo nas páginas
XHTML

```
<td>Primeiro Número: </td>
```

```
<td><h:inputText value="#{bean.numero1}"/></td>
```

A Aplicação JSF

- O JSF organiza todas as aplicações utilizando o modelo MVC
- Como fizemos em nossas aplicações MVC usando servlets e JSP, existe um servlet controlador
- Este servlet já está pronto dentro do JSF e é conhecido como *FacesServlet*, implementado na classe `javax.faces.webapp.FacesServlet`

Configuração do Servlet JSF

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.java.sun.com/xml/ns/xsi" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" >
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <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>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/welcomeJSF.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

 **Configuração do servlet controlador
fornecido pelo JSF**

Acesso ao Controlador

- O url-pattern do FacesServlet não precisa ser necessariamente /faces/*
 - O padrão /faces/* faz com que todos os endereços que comecem com /faces/ sejam direcionados para o servlet controlador
 - Podemos trocá-lo no arquivo web.xml ou na criação do projeto para qualquer outro padrão desejado
 - Por exemplo, o padrão *.jsf fará com que todo o acesso terminado com “.jsf” seja enviado para o FacesServlet

context-param?

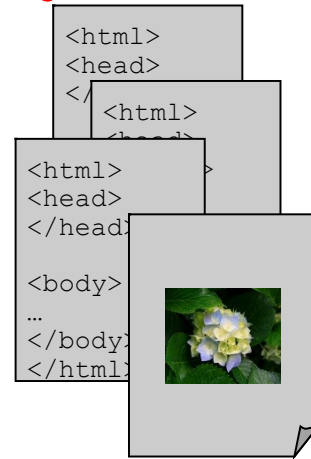
- Essa entrada no arquivo web.xml nos permite definir um parâmetro de contexto
 - Um parâmetro de contexto é um parâmetro visível para todas as páginas e servlets de uma determinada aplicação
- Nos projetos JSF, podemos definir um parâmetro de contexto opcional chamado `javax.faces.PROJECT_STAGE` onde especificamos o estágio atual do projeto
 - Valores possíveis: Production, Development, UnitTest, SystemTest

Arquitetura Básica do JSF



Usuário

O usuário efetua ações nas páginas da aplicação



As ações em páginas JSF são direcionadas para o servlet controlador do JSF (segundo configuração do arquivo web.xml)

FacesServlet

O controlador executa os métodos apropriados dos objetos managed beans

