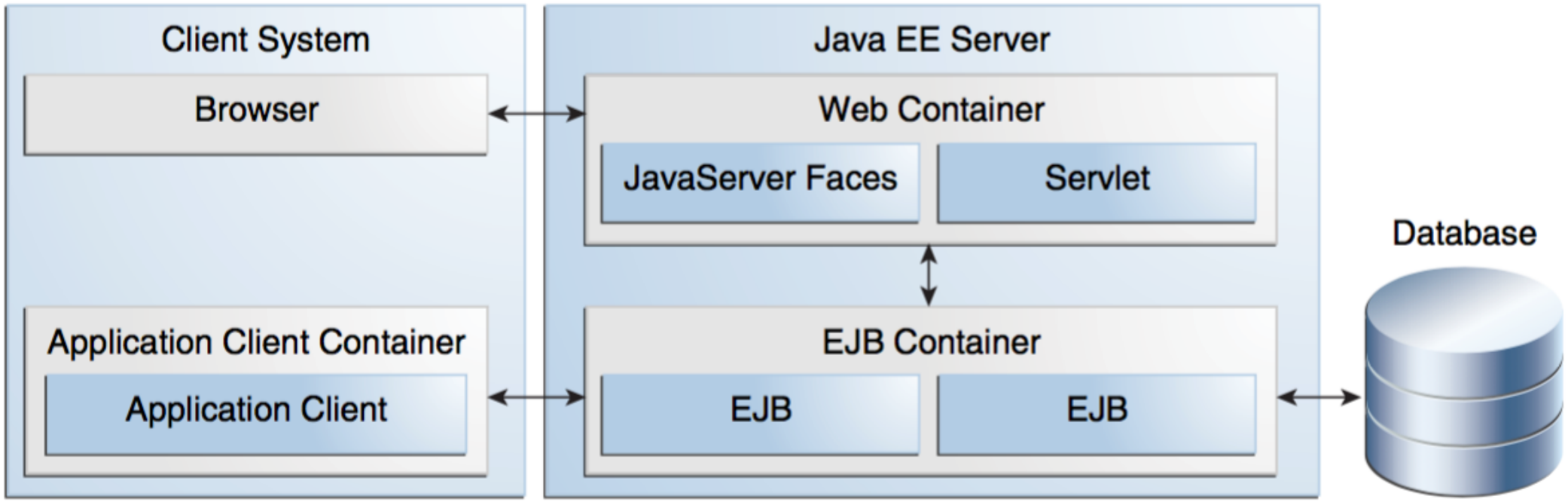
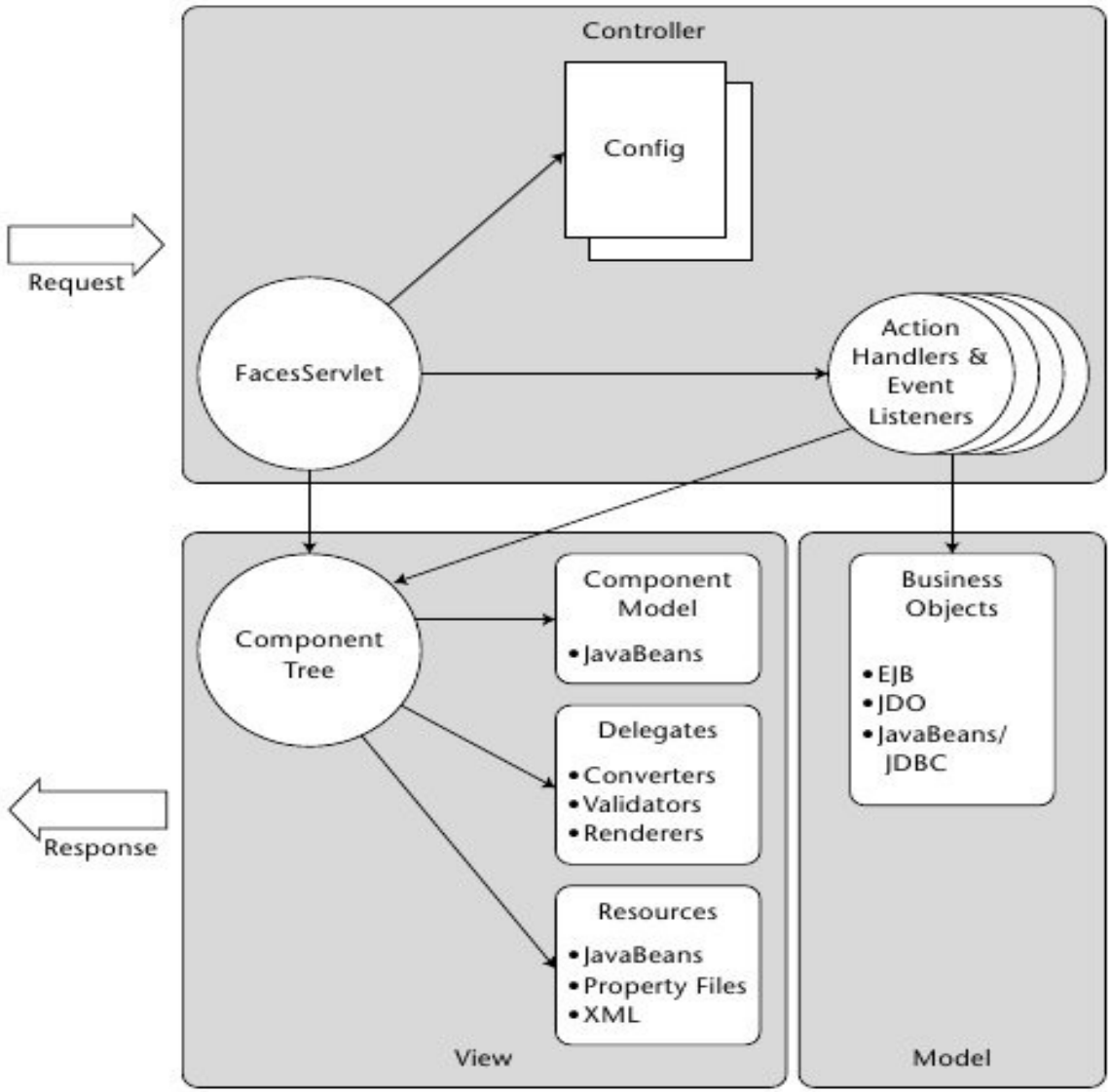
JSF

O JFS no Container Web



JSF Model-View-Controler



Configurando

Para que seja possível a utilização de JSF em uma aplicação WEB é necessário que o Faces Servlet seja configurado no arquivo de configuração da aplicação, o web.xml.

```
<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>*.xhtml</url-pattern>
</servlet-mapping>
```

Nesta configuração também é informado qual o padrão de URL para que o Faces Servlet identifique as páginas que ele irá processar.

JSF Components Tags

Tag	Functions	Rendered As	Appearance
`h:column`	Represents a column of data in a data component	A column of data in an HTML table	A column in a table
`h:commandButton`	Submits a form to the application	An HTML `<input type=`*value*`>` element for which the `type` value can be `"submit"`, `"reset"`, or `"image"`	A button
`h:commandLink`	Links to another page or location on a page	An HTML `<a href>` element	A link
`h:dataTable`	Represents a data wrapper	An HTML `<table>` element	A table that can be updated dynamically
`h:form`	Represents an input form (inner tags of the form receive the data that will be submitted with the form)	An HTML `<form>` element	No appearance
`h:graphicImage`	Displays an image	An HTML `<img>` element	An image
`h:inputSecret`	Allows a user to input a string without the actual string appearing in the field	An HTML `<input type="password">` element	A field that displays a row of characters instead of the actual string entered
`h:inputText`	Allows a user to input a string	An HTML `<input type="text">` element	A field
`h:inputTextarea`	Allows a user to enter a multiline string	An HTML `<textarea>` element	A multirow field
`h:message`	Displays a localized message	An HTML `<span>` tag if styles are used	A text string

JSF Components Tags

Tag	Functions	Rendered As	Appearance
`h:messages`	Displays localized messages	A set of HTML `<span>` tags if styles are used	A text string
`h:outputFormat`	Displays a formatted message	Plain text	Plain text
`h:outputLabel`	Displays a nested component as a label for a specified input field	An HTML `<label>` element	Plain text
`h:outputLink`	Links to another page or location on a page without generating an action event	An HTML `<a>` element	A link
`h:outputText`	Displays a line of text	Plain text	Plain text
`h:panelGrid`	Displays a table	An HTML `<table>` element with `<tr>` and `<td>` elements	A table
`h:panelGroup`	Groups a set of components under one parent	A HTML `<div>` or `<span>` element	A row in a table
`h:selectBooleanCheckbox`	Allows a user to change the value of a Boolean choice	An HTML `<input type="checkbox">` element	A check box
`h:selectManyListbox`	Allows a user to select multiple items from a set of items all displayed at once	An HTML `<select>` element	A box
`h:selectOneMenu`	Allows a user to select one item from a set of items	An HTML `<select>` element	A menu
`h:selectOneRadio`	Allows a user to select one item from a set of items	An HTML `<input type="radio">` element	A group of options

Pass-Throught HTML5 Elements

HTML5 Element Name	Identifying Attribute	Facelets Tag
a	jsf:action	h:commandLink
a	jsf:value	h:outputLink
body		h:body
button		h:commandButton
form		h:form
head		h:head
img		h:graphicImage
input	type="button"	h:commandButton
input	type="checkbox"	h:selectBooleanCheckbox
input	type="color"	h:inputText
input	type="date"	h:inputText
input	type="email"	h:inputText
input	type="number"	h:inputText
input	type="password"	h:inputSecret
input	type="submit"	h:commandButton
label		h:outputLabel
link		h:outputStylesheet
script		h:outputScript
select		h:selectOneListbox
textarea		h:inputTextArea

Facelets para Templates

Tag	Function
`ui:component`	Defines a component that is created and added to the component tree.
`ui:composition`	Defines a page composition that optionally uses a template. Content outside of this tag is ignored.
`ui:debug`	Defines a debug component that is created and added to the component tree.
`ui:decorate`	Similar to the composition tag but does not disregard content outside this tag.
`ui:define`	Defines content that is inserted into a page by a template.
`ui:fragment`	Similar to the component tag but does not disregard content outside this tag.
`ui:include`	Encapsulates and reuses content for multiple pages.
`ui:insert`	Inserts content into a template.
`ui:param`	Used to pass parameters to an included file.
`ui:repeat`	Used as an alternative for loop tags, such as `c:forEach` or `h:dataTable`.
`ui:remove`	Removes content from a page.

TagLibs Suportadas

Tag Library	URI	Prefix	Example	Contents
JavaServer Faces Facelets Tag Library	`http://xmlns.jcp.org/jsf/facelets`	`ui:`	`ui:component` `ui:insert`	Tags for templating

JavaServer Faces HTML Tag Library	`http://xmlns.jcp.org/jsf/html`	`h:`	`h:head`
			`h:body`
			`h:outputText`
			`h:inputText`
				JavaServer Faces component tags for all `UIComponent` objects

JavaServer Faces Core Tag Library	`http://xmlns.jcp.org/jsf/core`	`f:`	`f:actionListener` `f:attribute`	Tags for JavaServer Faces custom actions that are independent of any particular render kit
Pass-through Elements Tag Library	`http://xmlns.jcp.org/jsf`	`jsf:`	`jsf:id`	Tags to support HTML5-friendly markup
Pass-through Attributes Tag Library	`http://xmlns.jcp.org/jsf/passthrough`	`p:`	`p:type`	Tags to support HTML5-friendly markup
Composite Component Tag Library	`http://xmlns.jcp.org/jsf/composite`	`cc:`	`cc:interface`	Tags to support composite components
JSTL Core Tag Library	`http://xmlns.jcp.org/jsp/jstl/core`	`c:`	`c:forEach` `c:catch`	JSTL 1.2 Core Tags
JSTL Functions Tag Library	`http://xmlns.jcp.org/jsp/jstl/functions`	`fn:`	`fn:toUpperCase` `fn:toLowerCase`	JSTL 1.2 Functions Tags

Os prefixos do JSF

O JSF utiliza páginas XHTML e para configurarmos os prefixos utilizados pelo JSF necessitamos registra-los no elemento HTML.

```
<html lang="pt-br"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:jsf="http://xmlns.jcp.org/jsf">
```

O Managed Bean e o JSF

```java
@ManagedBean(name="cadaluno")
public class CadAluno {
   @EJB
   private AlunoDao dao;

   private String nome;
   private String matricula;

   public String getNome() {
       return nome;
   }

   public void setNome(String nome) {
       this.nome = nome;
   }

   public String getMatricula() {
       return matricula;
   }

   public void setMatricula(String matricula) {
       this.matricula = matricula;
   }

   public String salvar()  {
}
```

```xml
<h:head>
<meta charset="UTF-8" />
<title>Cadastro Aluno</title>
</h:head>
<h:body>
   <f:view>
       <h:messages />
       <h:form>
           <h3>Cadastro de Alunos</h3><hr />

           <h:outputLabel value="Nome: " />
           <h:inputText value="#{cadaluno.nome}" size="20" />

           <h:outputLabel value="Matricula: "/>
           <h:inputText size="8" value="#{cadaluno.matricula}" />

           <h:commandButton value="Enviar" action="#{cadaluno.salvar}" />
           <h:commandButton value="Voltar" action="menu.xhtml" />
       </h:form>
   </f:view>
</h:body>
</h:html>
```

Os métodos get e set do ManagedBean são utilizados pela página JSF.

SelectOneMenu

Um <h:selectOneMenu/> é implementado segundo o exemplo ao lado.

```
<h:selectOneMenu value="#{cadaluno.curso}">
    <f:selectItems  value="#{cadaluno.cursos}" var="item"
        itemLabel="#{item.nome}" itemValue="#{item.id}" />
</h:selectOneMenu>
```

Dos objetos retornados pelo método getCursos() são selecionados o label e o valor com os atributos itemLabel e itemValue respectivamente.

No atributo value em selectOneMenu é informado que receberá o valor do objeto selecionado.

```
private Integer curso;

public Integer getCurso() {
    return curso;
}

public void setCurso(Integer curso) {
    this.curso = curso;
}

public List<Curso> getCursos() {
    return dao.getCursos();
}
```

DataTable

Um <h:dataTable/> é implementado de forma similar ao for do JSP, porém cada coluna deve ser definida utilizando o elemento <h:column/>.

```
<h:dataTable value="#{cadaluno.alunos}" var="aluno">
    <h:column>
        <f:facet name="header">Del</f:facet>
        <h:selectBooleanCheckbox value="#{aluno.del}" />
    </h:column>

    <h:column>
        <f:facet name="header">Nome</f:facet>
        <h:outputText value="#{aluno.nome}" />
    </h:column>
</h:dataTable>
```

```
private ListDataModel<Aluno> alunos;

public ListDataModel<Aluno> getAlunos() {
    alunos = new ListDataModel<>(dao.getAlunos());
    return alunos;
}

@SuppressWarnings("unchecked")
public void remove() {
    for (Aluno item : (List<Aluno>) alunos.getWrappedData()) {
        if (item.isDel())
            dao.delAluno(item.getId());
    }
}
```

Uma coleção do tipo ListDataModel deve ser utilizada para conter os objetos de um <h:dataTable/>.

É através do método getWrappedData que obtemos os dados retornados pela página JSF para efetuar ações.