JSF - Biblioteca de tags HTML

A biblioteca http://java.sun.com/jsf/html possui os componentes básicos para renderização de telas em HTML.

Para utilizar esta biblioteca dentro da página xhtml, precisamos adicionar ela na propriedade da tag html e darmos um apelido (alias) para ela, por padrão é declarado da seguinte forma:

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

</html>
```

Usamos o alias **h** para referenciar a biblioteca html e para utilizar algum componente desta biblioteca utilizamos a sintaxe **h: + nome da tag**, exemplo: **<h:commandButton ...>**.

A biblioteca **html** possui os seguintes componentes: body, head, form, outputFormat, outpuLabel, outputLink, outputScript, outputStylesheet, outputText, button, commandButton, commandLink, link, graphicImage, inputHidden, inputSecret, inputText, inputTextarea, message, messages, selectBooleanCheckbox, selectManyCheckbox, selectManyListbox, selectManyMenu, selectOneListbox, selectOneMenu, selectOneRadio, dataTable, column, panelGrid e panelGroup.

Formulário

Quando montamos uma tela onde o usuário precisa entrar de alguma forma com uma informação, seja através de campos de digitação, itens de seleção, botões e outros, há a necessidade de criarmos um formulário.

Para enviar informações para o servidor, mais precisamente para uma ManagedBean precisamos criar um formulário e associar os campos da tela com os atributos do ManagedBean, exemplo:

Vamos criar um pequeno formulário para preencher informações e enviar uma mensagem para a tela com os dados recebidos.

Para isto vamos criar uma classe para representar um Contato:

```
package br.universidadejava.jsf.modelo;

import java.util.Date;

public class Contato {
  private String nome;
  private String telefone;
  private Date dataNascimento;

  public Date getDataNascimento() {
    return dataNascimento;
  }

  public void setDataNascimento(Date dataNascimento) {
    this.dataNascimento = dataNascimento;
  }

  public String getNome() {
    return nome;
  }

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

  public String getTelefone() {
    return telefone;
  }

  public void setTelefone(String telefone) {
    this.telefone = telefone;
  }
}
```

Agora vamos criar um ManagedBean para armazenar os atributos e ações da página de cadastro dos contatos.

```java
package br.universidadejava.jsf.managedbean;

import br.universidadejava.jsf.modelo.Contato;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

@ManagedBean
public class ContatoMB {
  private Contato contato = new Contato();

  /**
   * Método que irá simular o cadastro do contato.
   * @return página de entrada (index.xhtml)
   */
  public String adicionarContato() {
    //Cria um formatador de datas para o padrão dd/MM/yyyy.
    DateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    //Envia uma mensagem para a tela informando que foi cadastrado o contato.
    String msg = "Contato adicionado: " + contato.getNome() + " - " + contato.getTelefone() + " - "
    FacesMessage fm = new FacesMessage(msg);
    FacesContext.getCurrentInstance().addMessage("msg", fm);

    //Retorna para a página de entrada (index.xhtml).
    return "index";
  }

  public Contato getContato() {
    return contato;
  }

  public void setContato(Contato contato) {
    this.contato = contato;
  }
}
```

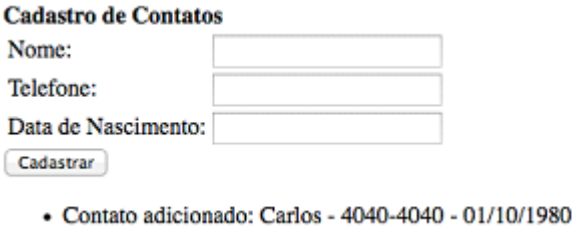
Criamos um simples **ManagedBean** com um atributo do tipo **Contato** e seus métodos get e set.

Vamos agora criar uma tela de cadastro de contatos que terá os campos de digitação do nome, telefone e data de nascimento de cada contato.

```markup
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xht
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
```

```
    <h:head>
        <title>Cadastro de Contato</title>
    </h:head>
    <h:body>
      <h:form id="formulario">
        <h:outputText id="titulo" value="Cadastro de Contatos" style="font-weight: bold;"/>
        <h:panelGrid id="dados" columns="2">
          <h:outputText id="nomeLabel" value="Nome:"/>
          <h:inputText id="nome" label="Nome" value="#{contatoMB.contato.nome}" maxlength="50"/>
          <h:outputText id="telefoneLabel" value="Telefone:"/>
          <h:inputText id="telefone" label="Telefone" value="#{contatoMB.contato.telefone}" maxleng
          <h:outputText id="nascimentoLabel" value="Data de Nascimento:"/>
          <h:inputText id="nascimento" label="Data de Nascimento" value="#{contatoMB.contato.dataNa
            <f:convertDateTime id="padraoData" pattern="dd/MM/yyyy" timeZone="America/Sao_Paulo"/>
          </h:inputText>
        </h:panelGrid>
        <h:commandButton id="cadastrar" value="Cadastrar" action="#{contatoMB.adicionarContato}"/>
        <h:messages id="msg"/>
      </h:form>
    </h:body>
</html>
```

Nesta tela usamos as tags **<h:form>** conteúdo do formulário </h:form> para informar a área que terá o formulário.

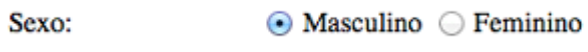


Seleção

Alguns componentes podem permitir que o usuário escolha uma opção, sem a necessidade de digitar alguma informação.

Com o componente h:selectOneRadio é possível mostrar diversas opções para que apenas uma possa ser selecionada.

Exemplo:



O código para montar esse h:selectOneRadio é:

```
<h:selectOneRadio id="sexo" value="#{contatoMB.contato.sexo}">
  <f:selectItem itemLabel="Masculino" itemValue="Masculino"/>
  <f:selectItem itemLabel="Feminino" itemValue="Feminino"/>
</h:selectOneRadio>
```

As opções disponíveis nos componentes de seleção são criados com o componente f:selectItem, exemplo:

```
<f:selectItem itemLabel="Masculino" itemValue="Masculino"/>
```

Nele podemos especificar um texto que será apresentado na tela com a propriedade itemLabel e utilizamos a propriedade itemValue para definir o seu valor.

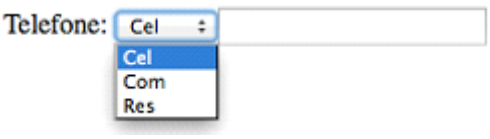
Uma outra forma se apresentar diversas opções onde o usuário tenha que escolher apenas uma, é utilizando o componente h:selectOneListbox exemplo:

O código para montar esse h:selectOneListbox é:

```
<h:selectOneListbox id="categ" value="#{contatoMB.contato.categoria}" size="2">
  <f:selectItem itemLabel="Amigo" itemValue="Amigo"/>
  <f:selectItem itemLabel="Familia" itemValue="Familia"/>
  <f:selectItem itemLabel="Trabalho" itemValue="Trabalho"/>
</h:selectOneListbox>
```

O selectOneListbox possui diversas propriedades como por exemplo a size que informa o tamanho de elementos aparecerão na lista.

Com o componente h:selectOneMenu podemos montar um ComboBox onde é possível selecionar apenas uma opção também, exemplo:



O código para montar esse h:selectOneMenu é:

```
<h:selectOneMenu id="tipo" value="#{contatoMB.contato.tipoTelefone}">
  <f:selectItem itemLabel="Cel" itemValue="Cel"/>
  <f:selectItem itemLabel="Com" itemValue="Com"/>
  <f:selectItem itemLabel="Res" itemValue="Res"/>
</h:selectOneMenu>
```

Se tivermos diversas opções onde o usuário pode selecionar mais de uma opção podemos utilizar uma lista com h:selectManyListbox, por exemplo:

O código para montar esse h:selectManyListbox é:

```
<h:selectManyListbox id="muitasCategorias" size="4">
  <f:selectItem itemLabel="Amigo" itemValue="Amigo"/>
  <f:selectItem itemLabel="Familia" itemValue="Familia"/>
  <f:selectItem itemLabel="Trabalho" itemValue="Trabalho"/>
</h:selectManyListbox>
```

Também podemos usar o h:selectManyCheckbox para permitir que o usuário selecione mais de uma opção:

O código para montar esse h:selectManyCheckbox é:

```
<h:selectManyCheckbox id="redesSociais" value="#{contatoMB.contato.redesSociais}">
  <f:selectItem itemLabel="Google+" itemValue="Google+"/>
  <f:selectItem itemLabel="Twitter" itemValue="Twitter"/>
  <f:selectItem itemLabel="Facebook" itemValue="Facebook"/>
  <f:selectItem itemLabel="LinkedIn" itemValue="LinkedIn"/>
</h:selectManyCheckbox>
```

Para armazenar quais as opções selecionadas podemos utilizar um vetor de Strings, exemplo:

```
private String[] redesSociais;

public String[] getRedesSociais() { return redesSociais; }
```

```
public void setRedesSociais(String[] redesSociais){
  this.redesSociais = redesSociais;
}
```

Um checkbox também pode ser utilizado para obter valores do tipo true (verdadeiro) ou falso (false), para isto utilizamos o h:selectBooleanCheckbox.

O código para montar esse h:selectManyCheckbox é:

```
<h:selectBooleanCheckbox id="statusAtivo" value="#{contatoMB.contato. ativo}"/>
```

Quando ele estiver selecionado significa true (verdadeiro), caso contrário significa false (falso), para armazenar este valor podemos utilizar um atributo boolean, exemplo:

```
private boolean ativo;

public boolean isAtivo() { return ativo; }

public void setAtivo(boolean ativo) { this.ativo = ativo; }
```

Tabela

Podemos definir uma tabela utilizando o componente h:dataTable, exemplo:

Nome	Tipo	Telefone	Data de Nascimento	Sexo	Categoria	Redes Sociais	Status
João	Cel	9876-5432	01/08/1990	Masculino	Amigo	Google+	Ativo
Maria	Res	1234-4321	10/03/1984	Feminino	Familia	Twitter; LinkedIn	Ativo
Carlos	Com	4444-2222	12/09/1978	Masculino	Trabalho	LinkedIn	Inativo

Para criar uma tabela podemos passar para ela um vetor ou lista através da propriedade **value** e com a propriedade **var** é criado uma variável para representar cada elemento do vetor ou lista, exemplo:

```
<h:dataTable id="contatos" value="#{contatoMB.contatos}" var="c" style="width: 100%">
```

Foi passado para o dataTable uma lista de objetos Contato através da propriedade value="#{contatoMB.contatos}" e para cada objeto Contato da lista será armazenado em uma variável chamada c através da propriedade var="c" para ser utilizada dentro da tabela.

Para definir os valores de cada coluna podemos utilizar o componente h:column, exemplo:

```
<h:column id="columnNome">
   <f:facet name="header">
     <h:outputText id="headerNome" value="Nome"/>
   </f:facet>
   <h:outputText id="valorNome" value="#{c.nome}"/>
 </h:column>
```

Note que dentro da coluna estamos usando um f:facet que é usado para representar um cabecalho (header) ou rodapé (footer) da coluna.

O exemplo completo da cadastro de contato vai ficar da seguinte forma:

Contato

Na classe Contato vamos adicionar mais alguns atributos para representar todos os valores que podem ser informados pelo usuário.

```
package br.universidadejava.jsf.modelo;

import java.util.Date;

public class Contato {
  private String nome;
  private String tipoTelefone;
  private String telefone;
  private Date dataNascimento;
  private String sexo;
  private String categoria;
  private String[] redesSociais;
  private boolean ativo;

  public Date getDataNascimento() { return dataNascimento; }
  public void setDataNascimento(Date dataNascimento) {
    this.dataNascimento = dataNascimento;
  }
```

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

  public String getTelefone() { return telefone; }

  public void setTelefone(String telefone) {
    this.telefone = telefone;
  }

  public boolean isAtivo() { return ativo; }
  public void setAtivo(boolean ativo) { this.ativo = ativo; }

  public String getTipoTelefone() { return tipoTelefone; }

  public void setTipoTelefone(String tipoTelefone) {
    this.tipoTelefone = tipoTelefone;
  }

  public String getSexo() { return sexo; }
  public void setSexo(String sexo) { this.sexo = sexo; }

  public String getCategoria() { return categoria; }
  public void setCategoria(String categoria) {
    this.categoria = categoria;
  }

  public String[] getRedesSociais() { return redesSociais; }

  public void setRedesSociais(String[] redesSociais) {
    this.redesSociais = redesSociais;
  }

  public String getRedesSociaisFormatadas() {
    String redes = "";

    if(getRedesSociais() != null) {
      for(int cont = 0; cont < getRedesSociais().length; cont++) {
        redes += getRedesSociais()[cont];

        if(cont < getRedesSociais().length - 1) {
          redes += "; ";
        }

      }
    }

    return redes;
  }
}
```

ContatoMB

No ManagedBean vamos adicionar agora a lista de contatos que será apresentado na tela no formato de tabela.

```
package br.universidadejava.jsf.managedbean;

import br.universidadejava.jsf.modelo.Contato;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

@ManagedBean
@SessionScoped
public class ContatoMB {
  private Contato contato = new Contato();
  private List<Contato> contatos = new ArrayList<Contato>();

  /**
   * Método que irá simular o cadastro do contato.
   * @return página de entrada (index.xhtml)
   */
  public String adicionarContato() {
    contatos.add(contato);

    //Cria um formatador de datas para o padrão dd/MM/yyyy.
    DateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    /* Envia uma mensagem para a tela informando que foi cadastrado o contato. */
    String msg = "Contato adicionado: " + contato.getNome() + " - " + contato.getTelefone() + " - "
    FacesMessage fm = new FacesMessage(msg);
    FacesContext.getCurrentInstance().addMessage("msg", fm);

    contato = new Contato();

    //Retorna para a página de entrada (index.xhtml).
    return "index";
  }

  public Contato getContato() {
    return contato;
  }

  public void setContato(Contato contato) {
    this.contato = contato;
  }

  public List<Contato> getContatos() {
    return contatos;
  }
```

```
  public void setContatos(List<Contato> contatos) {
    this.contatos = contatos;
  }
}
```

index.xhtml

E na tela vamos adicionar mais alguns componentes de seleção do usuário e a tabela para apresentar todos os contratos cadastrados.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xht
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
    <title>Cadastro de Contato</title>
  </h:head>
  <h:body>
    <h:form id="formulario">
      <h:outputText id="titulo" value="Cadastro de Contatos" style="font-weight: bold;"/>
      <h:panelGrid id="dados" columns="2">
        <h:outputText id="nomeLabel" value="Nome:"/>
        <h:inputText id="nome" label="Nome" value="#{contatoMB.contato.nome}" maxlength="50"/>
        <h:outputText id="telefoneLabel" value="Telefone:"/>
        <h:panelGroup id="groupTelefone">
          <h:selectOneMenu id="tipoTelefone" value="#{contatoMB.contato.tipoTelefone}":
            <f:selectItem itemLabel="Cel" itemValue="Cel"/>
            <f:selectItem itemLabel="Com" itemValue="Com"/>
            <f:selectItem itemLabel="Res" itemValue="Res"/>
          </h:selectOneMenu>
          <h:inputText id="telefone" label="Telefone" value="#{contatoMB.contato.telefone}" maxleng
        </h:panelGroup>
        <h:outputText id="nascimentoLabel" value="Data de Nascimento:"/>
        <h:inputText id="nascimento" label="Data de Nascimento" value="#{contatoMB.contato.dataNasc
          <f:convertDateTime id="padraoData" pattern="dd/MM/yyyy" timeZone="America/Sao_Paulo"/>
        </h:inputText>
        <h:outputText id="sexoLabel" value="Sexo:"/>
        <h:selectOneRadio id="sexo" value="#{contatoMB.contato.sexo}">
          <f:selectItem itemLabel="Masculino" itemValue="Masculino"/>
          <f:selectItem itemLabel="Feminino" itemValue="Feminino"/>
        </h:selectOneRadio>
        <h:outputText id="categoriaLabel" value="Categoria:"/>
        <h:selectOneListbox id="categoria" value="#{contatoMB.contato.categoria}" size="2">
          <f:selectItem itemLabel="Amigo" itemValue="Amigo"/>
          <f:selectItem itemLabel="Familia" itemValue="Familia"/>
          <f:selectItem itemLabel="Trabalho" itemValue="Trabalho"/>
        </h:selectOneListbox>
        <h:outputText id="redesSociaisLabel" value="Redes Sociais:"/>
```

```
    <h:selectManyCheckbox id="redesSociais" value="#{contatoMB.contato.redesSociais}">
      <f:selectItem itemLabel="Google+" itemValue="Google+"/>
      <f:selectItem itemLabel="Twitter" itemValue="Twitter"/>
      <f:selectItem itemLabel="Facebook" itemValue="Facebook"/>
      <f:selectItem itemLabel="LinkedIn" itemValue="LinkedIn"/>
    </h:selectManyCheckbox>
    <h:outputText id="statusLabel" value="Status:"/>
    <h:selectBooleanCheckbox id="ativostatus" value="#{contatoMB.contato.ativo}"/>
  </h:panelGrid>
  <h:commandButton id="cadastrar" value="Cadastrar" action="#{contatoMB.adicionarContato}"/>
  <h:messages id="msg"/>
  <h:dataTable id="contatos" value="#{contatoMB.contatos}" var="c" style="width: 100%">
    <h:column id="columnNome">
      <f:facet name="header">
        <h:outputText id="headerNome" value="Nome"/>
      </f:facet>
      <h:outputText id="valorNome" value="#{c.nome}"/>
    </h:column>
    <h:column id="columnTipoTelefone">
      <f:facet name="header">
        <h:outputText id="headerTipoTelefone" value="Tipo"/>
      </f:facet>
      <h:outputText id="valorTipoTelefone" value="#{c.tipoTelefone}"/>
    </h:column>
    <h:column id="columnTelefone">
      <f:facet name="header">
        <h:outputText id="headerTelefone" value="Telefone"/>
      </f:facet>
      <h:outputText id="valorTelefone" value="#{c.telefone}"/>
    </h:column>
    <h:column id="columnDataNascimento">
      <f:facet name="header">
        <h:outputText id="headerDataNascimento" value="Data de Nascimento"/>
      </f:facet>
      <h:outputText id="valorDataNascimento" value="#{c.dataNascimento}">
        <f:convertDateTime id="padraoDataNascimento" pattern="dd/MM/yyyy" timeZone="America/Sac
      </h:outputText>
    </h:column>
    <h:column id="columnSexo">
      <f:facet name="header">
        <h:outputText id="headerSexo" value="Sexo"/>
      </f:facet>
      <h:outputText id="valorSexo" value="#{c.sexo}"/>
    </h:column>
    <h:column id="columnCategoria">
      <f:facet name="header">
        <h:outputText id="headerCategoria" value="Categoria"/>
      </f:facet>
      <h:outputText id="valorCategoria" value="#{c.categoria}"/>
    </h:column>
    <h:column id="columnRedes ">
      <f:facet name="header">
        <h:outputText id="headerRedes" value="Redes Sociais"/>
      </f:facet>
      <h:outputText id="valorRedesSociais" value="#{c.redesSociaisFormatadas}"/>
```

```
        </h:column>
        <h:column id="columnStatus">
          <f:facet name="header">
            <h:outputText id="headerStatus" value="Status"/>
          </f:facet>
          <h:outputText id="valorStatus" value="#{c.ativo ? 'Ativo' : 'Inativo'}"/>
        </h:column>
      </h:dataTable>
    </h:form>
  </h:body>
</html>
```

A tela da aplicação ficará assim depois de alguns cadastros:

Exercícios

Exercício 1

Crie um formulário para agendar a data/hora para fazer o café, dado o seguinte formulário:

Data/Hora: (dd/mm/aaaa hh:mm)
Quantidade: (xicaras)
Açucar: Sim Não

Agendar

Ao clicar no botão Agendar, apresentar uma tela com a seguinte informação:

Café agendado para as 30/09/2011
Serão 10 xicaras com açucar.