

Lumi

Java Server Pages

Laércio Baldochi

COM222
Aula 06

Conteúdo

- Introdução
- Uso de expressões Java
- Scriptlets
- Misturando HTML com scriptlets
- Diretivas JSP
- Declarações JSP
- Tags JSP
- Sessões
- Beans e processamento de forms
- Tratamento de exceções

JSP

- Exercício
 - Pegue uma página HTML, qualquer uma!
 - Hello World
 - Mude a extensão do arquivo para jsp
 - Carregue a página jsp no browser
 - O que acontece?
 - A página é exibida, da mesma maneira que antes
 - A primeira exibição demora mais. Por que?
 - O arquivo jsp é transformado num Servlet, compilado e executado

JSP

Adicionando conteúdo dinâmico através de expressões

- O poder de JSP consiste na possibilidade de incluir código Java dentro de páginas Web
- Isso pode ser feito de duas formas
 - Expressões
 - Scriptlets
- Um exemplo do uso de expressões

```
<html>
```

```
<body>
```

```
<p>Olá, a hora atual é <%= new java.util.Date() %>
```

```
</body>
```

```
</html>
```

JSP

Adicionando conteúdo dinâmico através de expressões

- A sequência de caracteres `<%=` e `%>` servem para inserir código Java na página
- Permite a geração de conteúdo dinâmico toda vez que a página é carregada
- Exercício
 - Elabore uma página JSP que seja capaz de mostrar os valores retornados por `System.getProperty` para as seguintes propriedades:
 - `java.version`, `java.home`, `os.name`, `user.name`, `user.home`
 - Usar: `System.getProperty("propriedade");`

JSP - Scriptlets

- Usando apenas expressões Java não é possível realmente “programar” o conteúdo das páginas
- JSP permite a escrita de blocos de código Java, desde que esse código seja colocado entre `<%` e `%>`
 - Esses blocos de código são chamados scriptlets
- A princípio, o código dos scriptlets não geram conteúdo
- O código de um scriptlet é carregado toda vez que uma página é carregada

JSP - Scriptlets

- Exemplo

```
<HTML>
```

```
<BODY>
```

```
<%
```

```
    // Esse é um scriptlet. Note que a variável date declarada aqui
```

```
    // fica disponível para ser usada mais tarde numa expressão
```

```
    System.out.println( "Avaliando a hora");
```

```
    java.util.Date date = new java.util.Date();
```

```
%>
```

```
Olá! A hora atual é <%= date %>
```

```
</BODY>
```

```
</HTML>
```

JSP - Scriptlets

- Ao executar o exemplo anterior é possível notar a saída do “System.out.println” no log do servidor
 - Como estamos rodando o servidor dentro do Netbeans, a saída aparece na janela inferior do ambiente
- Nota
 - A impressão de dados na saída padrão consiste numa forma eficiente de debugar o código de uma página JSP

JSP - Scriptlets

- Por si só um scriptlet não gera conteúdo HTML
- Para permitir que um scriptlet gere conteúdo HTML, deve-se usar a variável out
- A variável out não precisa ser declarada
 - Ela é predefina para scriptlets

JSP - Scriptlets

- Exemplo

```
<HTML>
```

```
<BODY>
```

```
<%
```

```
    // Este scriptlet declara e inicializa a variável date
    System.out.println( "Avaliando a hora");
    java.util.Date date = new java.util.Date();
```

```
%>
```

```
Olá! A hora atual é
```

```
<%
```

```
    // Este scriptlet produz a saída em HTML
    out.println(date);
```

```
%>
```

```
</BODY>
```

```
</HTML>
```

JSP - Scriptlets

- Além da variável out, outra variável pré-definida é request
 - Lembrem-se que já utilizamos request e response nos servlets
- request
 - Diz respeito à requisição de um cliente
 - Usando a variável request é possível obter informação a respeito de uma requisição de um cliente
 - O exemplo a seguir usa o request para obter o nome do host do cliente (ou seu endereço IP)

JSP - Scriptlets

- Exemplo

```
<HTML>
```

```
<BODY>
```

```
<%
```

```
    // Este scriptlet declara e inicializa a variável date
```

```
    System.out.println( "Avaliando a hora");
```

```
    java.util.Date date = new java.util.Date();
```

```
%>
```

```
Olá! A hora atual é
```

```
<%
```

```
    out.println(date);
```

```
    out.println("<BR>O endereço da sua máquina é: ");
```

```
    out.println(request.getRemoteHost());
```

```
%>
```

```
</BODY>
```

```
</HTML>
```

Misturando Scriptlets e HTML

- O exemplo anterior traz um `
` dentro de um comando out de um scriptlet
- Podemos, entretanto, misturar tags HTML com scriptlets
- Considere o exemplo a seguir, o qual permite criar uma tabela de n colunas

Misturando Scriptlets e HTML

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ ) {
    %>
    <TR>
    <TD>Número</TD>
    <TD><%= i+1 %></TD>
    </TR>
    <%
  }
%>
</TABLE>
```

Misturando Scriptlets e HTML

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ ) {
    %>
    <TR>
    <TD>Número</TD>
    <TD><%= i+1 %></TD>
    </TR>
    <%
  }
%>
</TABLE>
```

Misturando Scriptlets e HTML

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ ) {
    %>
    <TR>
    <TD>Número</TD>
    <TD><%= i+1 %></TD>
    </TR>
    <%
  }
%>
</TABLE>
```


Misturando Scriptlets e HTML

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ ) {
    %>
    <TR>
    <TD>Número</TD>
    <TD><%= i+1 %></TD>
    </TR>
    <%
  }
%>
</TABLE>
```

Misturando Scriptlets e HTML

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ ) {
    %>
    <TR>
    <TD>Número</TD>
    <TD><%= i+1 %></TD>
    </TR>
    <%
  }
%>
</TABLE>
```

Misturando Scriptlets e HTML

```
<TABLE BORDER=2>
<%
  for ( int i = 0; i < n; i++ ) {
    %>
    <TR>
    <TD>Número</TD>
    <TD><%= i+1 %></TD>
    </TR>
    <%
  }
%>
</TABLE>
```

Misturando Scriptlets e HTML

- Misturando scriptlets com HTML é possível fazer com que expressões for, while e if controlem também a execução de código HTML
- O maior problema é controlar o início e fim de cada scriptlet e de cada fragmento de código HTML

Misturando Scriptlets e HTML

- Exercício
 - Fixe um valor para n (digamos, $n=4$) e implemente uma página JSP que construa uma tabela utilizando o código apresentado

Diretivas JSP

- Até agora temos usado as classes Java totalmente qualificadas
 - `java.util.Date`
- É possível usar imports no JSP, mas a sintaxe é um pouco diferente

Diretivas JSP

```
<%@ page import="java.util.*" %>
```

```
<HTML>
```

```
<BODY>
```

```
<%
```

```
    // Esse é um scriptlet. Note que a variável date declarada aqui
```

```
    // fica disponível para ser usada mais tarde numa expressão
```

```
    System.out.println( "Avaliando a hora");
```

```
    java.util.Date date = Date();
```

```
%>
```

```
Olá! A hora atual é <%= date %>
```

```
</BODY>
```

```
</HTML>
```

Diretivas JSP

- A linha

`<%@ page import="java.util.*" %>`

é chamada diretiva

- Diretivas começam com `<%@`
- A diretiva que estamos usando para fazer imports é chamada diretiva `page`
- Para importar vários pacotes usando a diretiva `page`, basta separá-los por vírgula

`<%@ page import="java.util.*, java.io.*" %>`

Diretivas JSP

- Além da diretiva page, existem outras
 - **include, taglib, ...**
- A diretiva include é usada para incluir numa página JSP o conteúdo de um outro arquivo, que pode ser HTML, JSP ou outro tipo de arquivo

Diretivas JSP

- Exemplo do uso de include

```
<HTML>
```

```
<BODY>
```

```
Incluindo a página hello.jsp...<BR>
```

```
<%@ include file="hello.jsp" %>
```

```
</BODY>
```

```
</HTML>
```

Declarações JSP

- Vimos que as páginas JSP são convertidas em uma classe Java (Servlet)
- O que acontece com os scriptlets?
 - Todos os scriptlets são colocados em um único método dessa classe
- Utilizando JSP é possível adicionar declarações de variáveis e de métodos a essa classe
- As declarações podem ser adicionadas por meio das tags `<%!` e `%>`

Declarações JSP

```
<%@ page import="java.util.*" %>
<HTML>
<BODY>
<%!
    Date data = new Date();
    Date getDate()
    {
        System.out.println( "Executando o método getDate() " );
        return data;
    }
%>
Olá! A data/hora atual é <%= getDate() %>
</BODY>
</HTML>
```

Declarações JSP

- Qual o problema com o código anterior?
 - A data será a mesma todas as vezes que a página for carregada.
 - Por que?
- Exercício
 - Modifique o código anterior adicionando a função `computeDate` a qual deve reinicializar o valor da variável `data`

Tags jsp

- Assim como a linguagem HTML possui suas tags próprias (
, <p>, <a>, <href>, etc.), JSP define também um conjunto de tags
 - Não usa <%, simplesmente <
- Tags jsp podem ser de 2 tipos
 - Predefinidas
 - Provenientes de bibliotecas de tags
 - Tag libraries
- Tags predefinidas começam com o prefixo jsp.
 - Exemplo: <jsp:include>
 - Vejamos a seguir um exemplo de jsp:include

Tags jsp

<HTML>

<BODY>

Incluindo o arquivo hello.jsp...

<jsp:include page="hello.jsp"/>

</BODY>

</HTML>

- Qual a diferença entre esse include, baseado no uso de tag, e no anterior, baseado no uso de diretiva?

Tags jsp

- Qual a diferença entre esse include, baseado no uso de tag, e no anterior, baseado no uso de diretiva?
 - No anterior o texto original do arquivo é incluído no novo arquivo (é gerado um novo arquivo contendo todo o conteúdo)
 - No atual, o include ocorre em tempo de execução
 - O browser carrega o arquivo atual e depois carrega o conteúdo do arquivo incluído
 - Não há geração de um novo arquivo com tudo incluído

Sessões JSP

- O estabelecimento de uma sessão é de fundamental importância para diversas aplicações Web
 - Carrinhos de compra, por exemplo
- Problema
 - É necessário identificar as interações de um visitante
- Solução
 - Sessão JSP
 - Uma sessão é um objeto que pode ser associado a um visitante
 - É possível colocar dados em um objeto sessão e recuperá-los posteriormente. Vejamos um exemplo a seguir

Sessões JSP

Index.html

```
<HTML>
<BODY>
<FORM METHOD=POST ACTION="SaveName.jsp">
Qual o seu nome? <INPUT TYPE=TEXT NAME=username SIZE=20>
<P><INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

SaveName.jsp

```
<%
String name = request.getParameter( "username" );
session.setAttribute( "nomeUsuario", name );
%>
<HTML>
<BODY>
<A HREF="NextPage.jsp">Continue</A>
</BODY>
</HTML>
```

Sessões JSP

- O primeiro arquivo é uma página HTML estática contendo um formulário
 - Especifica que a ação está direcionada para uma página chamada `SaveName.jsp`
- A página `SaveName.jsp` recebe o atributo `username` e seta a sessão com esse atributo
 - Disponibiliza um link para outra página JSP
 - `NextPage.jsp`

Sessões JSP

NextPage.jsp

```
<HTML>
<BODY>
Olá, <%= session.getAttribute("nomeUsuario") %>
</BODY>
</HTML>
```

- Se n pessoas acessarem a página simultaneamente, n sessões serão criadas
 - O servidor guardará dados de todas as sessões simultâneas

Sessões JSP

- Exercício
 - Adicione o atributo idade ao exemplo anterior, recupere e exiba esse atributo na página seguinte

Beans e Processamento de Forms

- Um dos pontos fortes de JSP diz respeito ao processamento de forms
- A forma padrão de processar forms em JSP consiste na definição de um bean
 - Classe que tem um atributo para cada campo do formulário
 - Métodos setters (e getters)
- Vejamos um exemplo que estende o arquivo index.html que acabamos de utilizar

Beans e Processamento de Forms

```
<HTML>
<BODY>
<FORM METHOD=POST ACTION="SaveName.jsp">
Digite seu nome: <INPUT TYPE=TEXT NAME=username SIZE=20><BR>
Digite seu e-mail: <INPUT TYPE=TEXT NAME=email SIZE=20><BR>
Digite sua idade: <INPUT TYPE=TEXT NAME=idade SIZE=4>
<P><INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

Beans e Processamento de Forms

- Para coletar os dados do formulário, definiremos uma classe (bean) com os campos username, email e idade
 - Definiremos também métodos setters para ajustar os valores dessas variáveis a partir do conteúdo recebido (conteúdo que foi digitado pelo usuário)

Beans e Processamento de Forms

```
package user;
public class UserData {
    String username;
    String email;
    String idade;

    public void setUsername( String value ){
        username = value;
    }
    public void setEmail( String value ){
        email = value;
    }
    public void setIdade( String value ){
        idade = value;
    }

    public String getUsername() { return username; }
    public String getEmail() { return email; }
    public String getIdade() { return idade; }
}
```

Beans e Processamento de Forms

- O próximo passo consiste em ajustar SaveName.jsp para que este use o bean para recuperar os dados do formulário

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>  
<jsp:setProperty name="user" property="*" />  
<HTML>  
<BODY>  
<A HREF="NextPage.jsp">Continue</A>  
</BODY>  
</HTML>
```

Beans e Processamento de Forms

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>
<jsp:setProperty name="user" property="*" />
<HTML>
<BODY>
<A HREF="NextPage.jsp">Continue</A>
</BODY>
</HTML>
```

- A tag useBean faz com que seja procurada uma instância de “user.UserData” na sessão
 - Se existir a instância, a mesma será atualizada
 - Caso contrário uma instância será criada e colocada na sessão

Beans e Processamento de Forms

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>  
<jsp:setProperty name="user" property="*" />  
<HTML>  
<BODY>  
<A HREF="NextPage.jsp">Continue</A>  
</BODY>  
</HTML>
```

- A tag setProperty é responsável por coletar automaticamente os dados do formulário e ajustar os atributos da classe UserData
 - Chama os métodos setters
 - É por isso que deve haver a correspondência entre os nomes usados nos formulários e os nomes dos atributos do bean

Beans e Processamento de Forms

- O último passo consiste em ajustar nextPage.jsp para que este recupere os dados do bean

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>
<HTML>
<BODY>
Você digitou os seguintes dados<BR>
Nome: <%= user.getUsername() %><BR>
Email: <%= user.getEmail() %><BR>
Idade: <%= user.getIdade() %><BR>
</BODY>
</HTML>
```

Beans e Processamento de Forms

- Exercício
 - Modifique o exemplo para que o mesmo leia o ano de nascimento da pessoa e informe sua idade aproximada

Exercícios

Uso de beans

1) Contador de acessos

- Neste exercício criaremos duas instâncias de um bean contador
 - Uma instância de sessão
 - Uma instância de aplicação
- Veremos como a aplicação se comporta a ser acessada por diferentes browsers

Contador.jsp

```
<%@ page import="contador.ContadorBean" %>
<jsp:useBean id="cont_sessao" class="contador.ContadorBean" scope="session" />
<jsp:useBean id="cont_aplicacao" class="contador.ContadorBean" scope="application" />

<%
    cont_sessao.aumentaCont();
    synchronized(page) {
        cont_aplicacao.aumentaCont();
    }
%>
<h3>
Numero de acessos nesta sessão:
<jsp:getProperty name="cont_sessao" property="cont" />
</h3>
<p>
<h3>
Número total de acessos:
<% synchronized(page) { %>
<jsp:getProperty name="cont_aplicacao" property="cont" />
<% } %>
</h3>
```

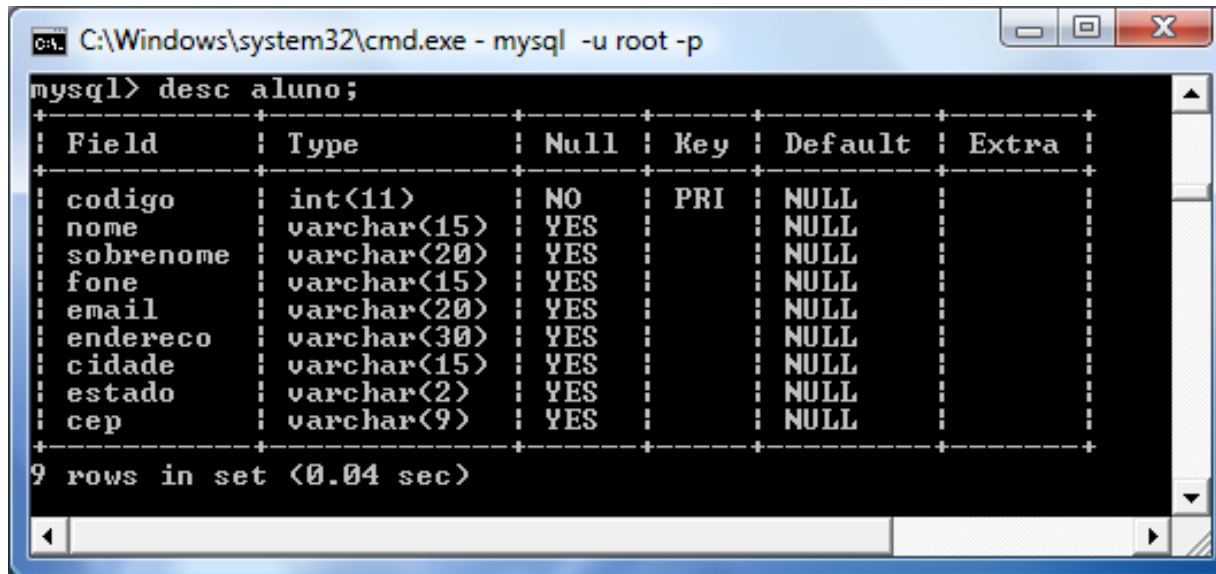
ContadorBean.java

```
package contador;
public class ContadorBean {
    int cont;

    public int getCont() {
        return cont;
    }

    public void aumentaCont() {
        cont++;
    }
}
```

Exercício 2 - JSP + Servlets

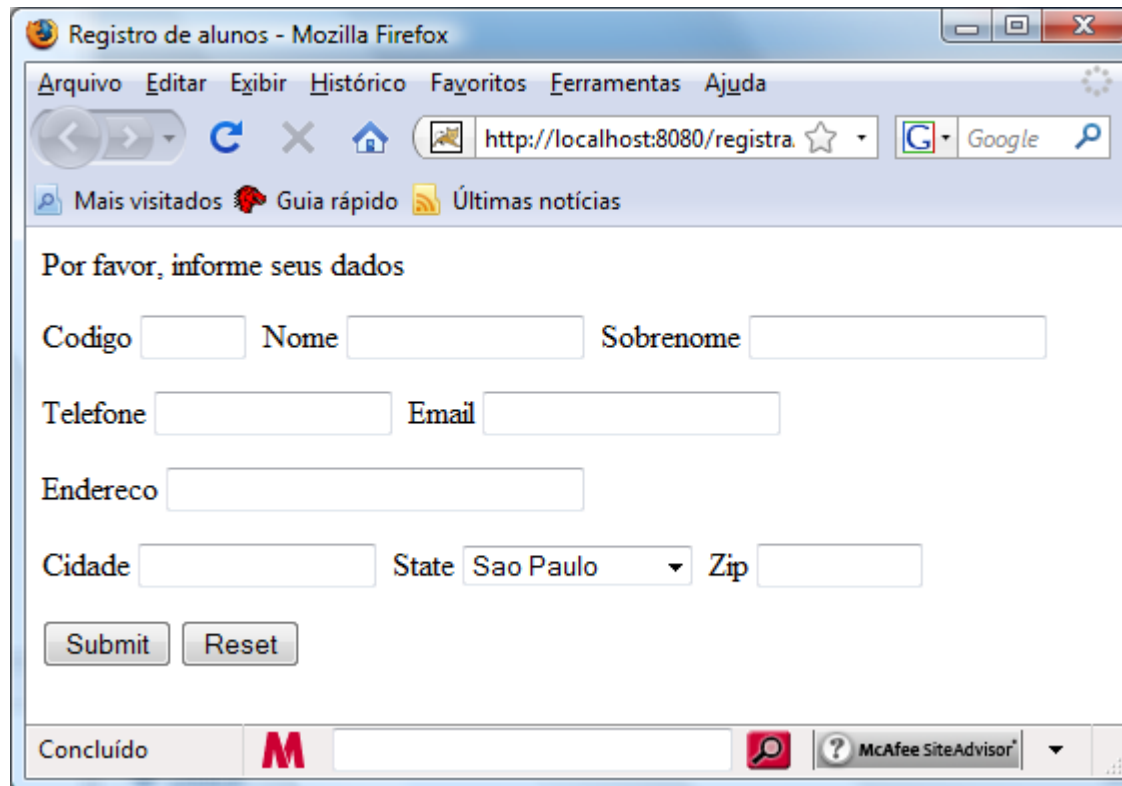


```
C:\Windows\system32\cmd.exe - mysql -u root -p
mysql> desc aluno;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| codigo | int(11) | NO   | PRI | NULL    |       |
| nome   | varchar(15) | YES |     | NULL    |       |
| sobrenome | varchar(20) | YES |     | NULL    |       |
| fone   | varchar(15) | YES |     | NULL    |       |
| email  | varchar(20) | YES |     | NULL    |       |
| endereco | varchar(30) | YES |     | NULL    |       |
| cidade | varchar(15) | YES |     | NULL    |       |
| estado | varchar(2)  | YES |     | NULL    |       |
| cep    | varchar(9)  | YES |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.04 sec)
```

Usando a mesma tabela SQL do exercício 2 da aula 5, elabore uma aplicação para cadastramento de funcionários com a seguinte funcionalidade:

Exercício 2 - JSP + Servlets

1) Gerar o formulário utilizando o index.html fornecido



The screenshot shows a Mozilla Firefox browser window titled "Registro de alunos - Mozilla Firefox". The address bar displays "http://localhost:8080/registra". The page content includes a menu bar with "Arquivo", "Editar", "Exibir", "Histórico", "Favoritos", "Ferramentas", and "Ajuda". Below the menu bar are navigation buttons (back, forward, home, stop) and a search bar with "Google". The main content area contains the text "Por favor, informe seus dados" followed by a registration form. The form fields are: "Codigo", "Nome", "Sobrenome", "Telefone", "Email", "Endereco", "Cidade", "State" (a dropdown menu currently showing "Sao Paulo"), and "Zip". At the bottom of the form are "Submit" and "Reset" buttons. The status bar at the bottom of the browser shows "Concluído", a red "M" logo, and a "McAfee SiteAdvisor" icon.

Registro de alunos - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://localhost:8080/registra

Mais visitados Guia rápido Últimas notícias



Por favor, informe seus dados

Codigo Nome Sobrenome

Telefone Email

Endereco

Cidade State Zip

Concluído  

```
<html>
<head>
  <title>Cadastramento de alunos</title>
</head>
<body>
  Por favor, informe seus dados
  <form method = "post" action = "/registraAluno/Registra">
    <p>Codigo
      <input type = "text" name = "codigo" size = "4">&nbsp;
      Nome
      <input type = "text" name = "nome" size = "15">&nbsp;
      Sobrenome
      <input type = "text" name = "sobrenome" size = "20">
    </p>
    <p>Telefone
      <input type = "text" name = "fone" size = "15">&nbsp;
      Email
      <input type = "text" name = "email" size = "20">&nbsp;
    </p>
    <p>Endereço <input type = "text" name = "endereco" size = "30">
    </p>
    <p>Cidade <input type = "text" name = "cidade" size = "15">&nbsp;
      State
      <select size = "1" name = "estado">
        <option value = "SP">Sao Paulo</option>
        <option value = "MG">Minas Gerais</option>
        <option value = "RJ">Rio de Janeiro</option>
      </select>&nbsp;
      Zip <input type = "text" name = "cep" size = "9">
    </p>
    <p><input type = "submit" name = "Submit" value = "Submit">
      <input type = "reset" value = "Reset">
    </p>
  </form>
</body>
</html>
```

Exercício 2 - JSP + Servlets

- 2) Definir uma classe bean para processar os campos do formulário
- 3) Usando JSP, exibir o conteúdo digitado e solicitar confirmação, por meio de dois links ou dois botões
 - > Salvar dados
 - > Cancelar

Exercício 2 - JSP + Servlets

- 4) O link Salvar dados deve ativar um Servlet que persiste as informações no banco, informando sucesso ou falha na operação

Importante: para fazer a persistência deve-se utilizar o bean

- 5) O link Cancelar deve retornar à página do formulário para colher novos dados

Exercícios 3, 4 e 5

Entrega:

3) Processamento de forms

- Vocês devem elaborar uma aplicação JSP para coletar os dados de um formulário
- Para tanto você precisará de dois arquivos:
 - `form.jsp`
 - `FormBean.java`

3) Processamento de forms

- form.jsp
 - Deve prover uma interface Web como a mostrada a seguir
 - Deve enviar os dados coletados para um bean para processamento
 - FormBean.java
 - Deve recuperar os dados do bean e exibi-los
 - Veja o código parcial de [form.jsp](#)

form.jsp FormBean.java http://localhost:8080/Aula3/form.jsp

http://localhost:8080/Aula3/form.jsp

Registro de usuário

Primeiro nome
Laercio

Último nome
Baldochi

E-Mail
baldochi@gmail.com

Em quais linguagens você programa?

☒ Java ☒ C++ ☒ C
☐ Perl ☐ COBOL ☒ VB

Com qual frequência poderemos notificá-lo sobre novidades?

☐ Semanalmente ☒ Mensalmente ☐ Quinzenalmente

form.jsp FormBean.java http://localhost:8080/Aula3/form.jsp

http://localhost:8080/Aula3/form.jsp

Registro de usuário

Primeiro nome

Último nome

E-Mail

Em quais linguagens você programa?

☐ Java ☐ C++ ☐ C
☐ Perl ☐ COBOL ☐ VB

Com qual frequência poderemos notificá-lo sobre novidades?

☒ Semanalmente ☐ Mensalmente ☐ Quinzenalmente

Você submeteu as seguintes informações:

Primeiro nome:
Laercio

Último nome:
Baldochi

Email:
baldochi@gmail.com

Linguagens:

- Java
- C++
- C
- VB

Notificação:
Mensalmente

Exercício 3

- Dica

No beans:

```
private String[] linguagens;  
public void setLinguagens(String[] linguagens) {  
    this.linguagens = linguagens;  
}  
public String[] getLinguagens() {  
    return linguagens;  
}
```

Exercício 3

- Dica

No jsp:

```
String linguagens[] = formHandler.getLinguagens();  
    for (int i = 0; i < linguagens.length; i++) {  
        out.println(linguagens[i]);  
    }
```

Exercício 4

- Modifique o exercício 3 de tal forma que:
 - Após o pressionamento do botão submit, os dados coletados sejam exibidos em uma nova página, chamada Confirm.jsp
 - A página de confirmação deve conter um botão que, se pressionado, invoca um servlet para salvar os dados
 - O Servlet gera uma última página informando sucesso na gravação dos dados

Exercício 4

- Script para criar tabela:
 - `CREATE TABLE usuario (email varchar(30) NOT NULL, nome varchar(15) NOT NULL, sobrenome varchar(20) NOT NULL, linguagens varchar(40) NOT NULL, notificacao varchar(15) NOT NULL, PRIMARY KEY (email))ENGINE=InnoDB ;`

Exercício 5: Uso de sessões

- Usando beans e sessões, implemente uma aplicação JSP com a seguinte pergunta:
- Quem será o campeão brasileiro de 2016
 - Palmeiras
 - São Paulo
 - Corinthians
 - Cruzeiro
 - Atlético
 - Seu time
 - Opções utilizando radio buttons

Exercício 5: Uso de sessões

- O bean deve permitir apenas 1 voto por sessão. Dessa forma, após votar, duas opções de resposta são possíveis:
 - 1) Seu voto foi computado. O placar atual é o seguinte...
 - 2) Seu voto não pode ser computado porque você já votou nesta enquete

Exercício 5: Uso de sessões

- Após concluir o exercício, peça para seus colegas votarem na enquete e verifique se a aplicação está se comportando corretamente