

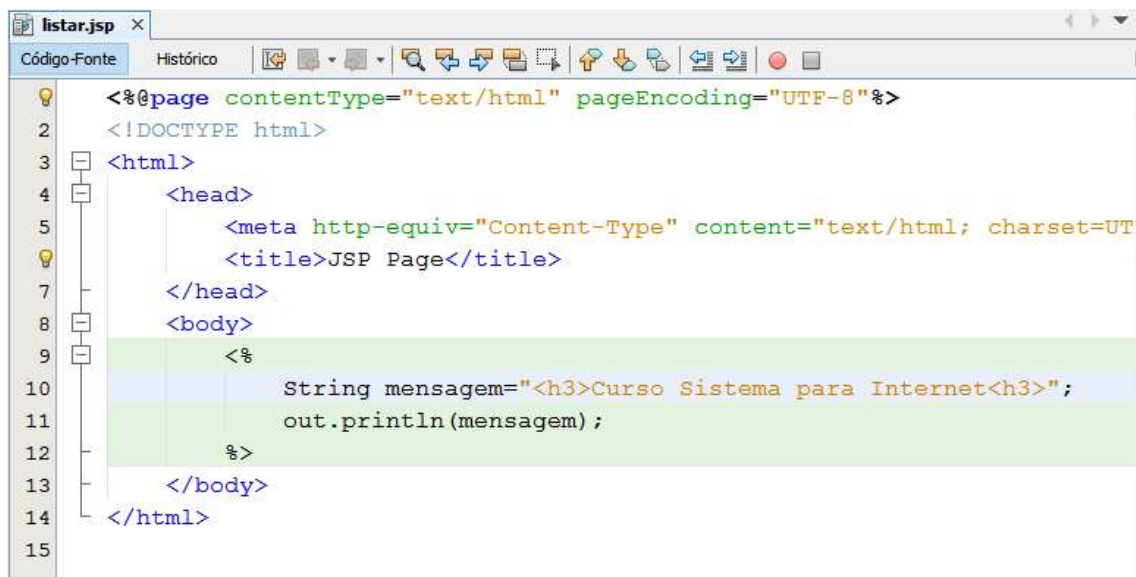
## Introdução

**Java Server Pages** basicamente são páginas HTML com códigos em Java. É uma solução similar ao ASP e PHP.

Veja um pequeno exemplo a seguir:

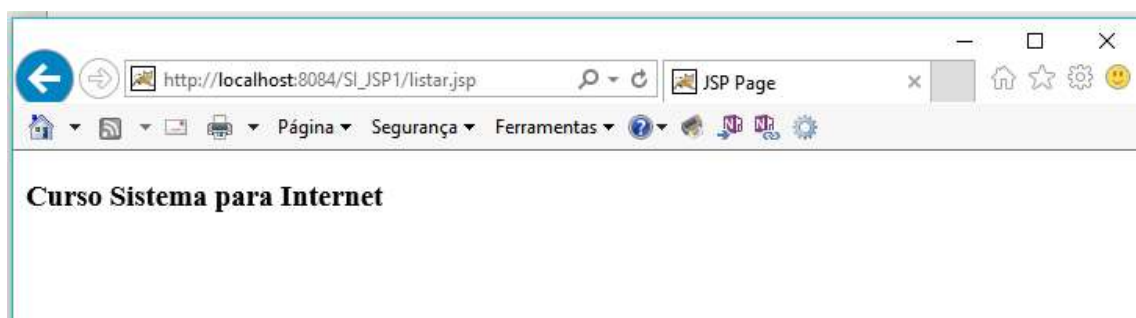
Para construir uma página usando JSP você geralmente escreve o texto HTML e inclui código em Java entre *tags* JSP.

As *tags* começam com **<%** e terminam com **%>**



```
1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"%>
6     <title>JSP Page</title>
7   </head>
8   <body>
9     <%
10      String mensagem="<h3>Curso Sistema para Internet<h3>";
11      out.println(mensagem);
12    %>
13   </body>
14 </html>
15
```

Para ver o resultado basta chamar, diretamente a página através do browser



**Código fonte (View Source) do exemplo anterior**

```
listar.jsp
1
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6     <title>JSP Page</title>
7   </head>
8   <body>
9     <h3>Curso Sistema para Internet</h3>
10
11   </body>
12 </html>
13
```

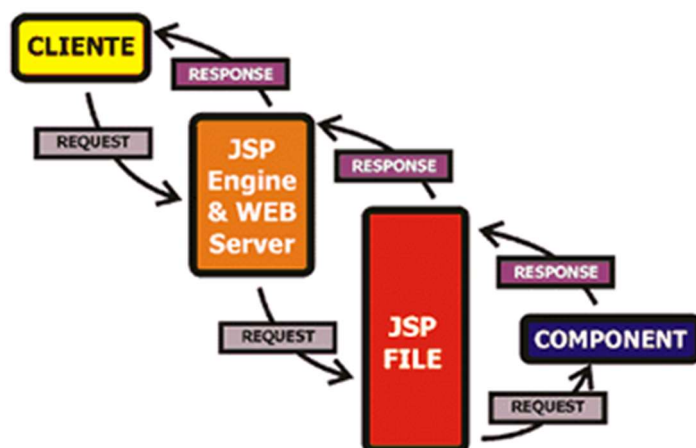
Observe que no código fonte não há nada de Java.  
Apenas HTML

### Funcionamento de uma página JSP

A primeira vez que uma página JSP é carregada pelo *container* JSP, o código Java é compilado gerando um *servlet* que é executado, gerando uma página HTML que é enviada para o navegador.

As chamadas subseqüentes são enviadas diretamente ao *servlet* gerado na primeira requisição, não ocorrendo mais as etapas de geração e compilação do *servlet*.

A figura seguinte mostra um esquema das etapas de execução de uma página JSP na primeira vez em que a mesma é requisitada.



Inicialmente a requisição é enviada para um servidor Web que reencaminha a requisição para o *container servlet/JSP*.

Em seguida o *container* verifica que não existe nenhuma instância de *servlet* correspondente à página JSP.

Neste caso, a página JSP é traduzida para código fonte de uma classe *servlet* que será usada em resposta à requisição.

O código fonte do *servlet* é compilado, e em seguida é criada uma instância da classe.

Finalmente, é invocado o método *service()* da instância *servlet* para gerar a resposta à requisição.

## Categorias de tags

JSP fornece cinco categorias de *tags*:

- Declarações
- Diretivas
- Expressões
- Scriptlets
- Comentários

## Declarações

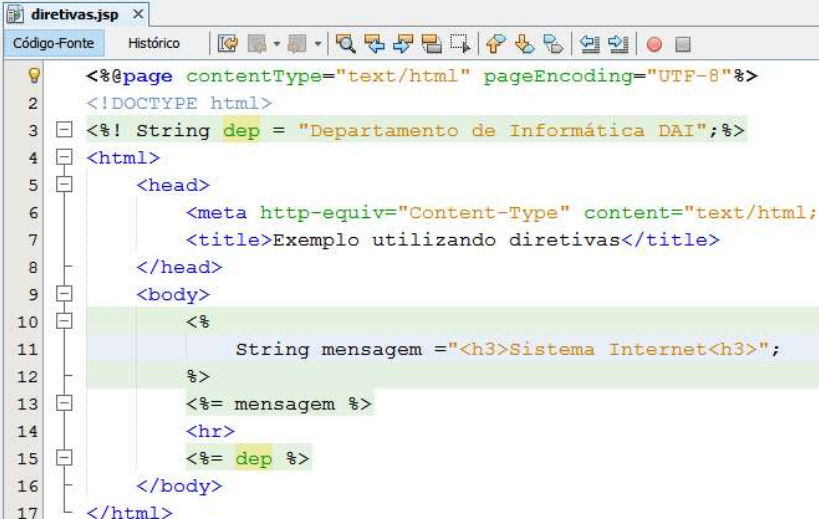
Usadas entre `<%! e %>`. São utilizadas para definir variáveis e métodos específicos para uma página JSP.

Os métodos e variáveis declaradas podem então ser referenciados por outros elementos de criação de *scriptlets* na mesma página.

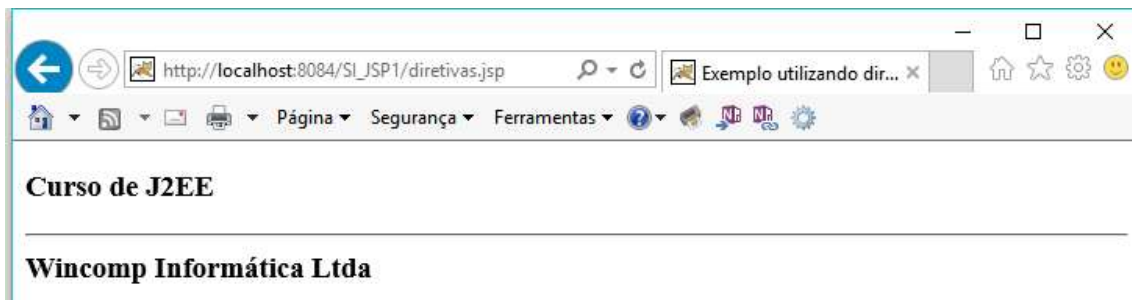
Cada declaração deve ser finalizada ou separada por "ponto-e-vírgula" e pode assumir a seguinte sintaxe:

`<%! declaração1; declaração2; %>`

Exemplo



```
1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <!DOCTYPE html>
3 <%! String dep = "Departamento de Informática DAI";%>
4 <html>
5   <head>
6     <meta http-equiv="Content-Type" content="text/html;
7     <title>Exemplo utilizando diretivas</title>
8   </head>
9   <body>
10     <%
11       String mensagem = "<h3>Sistema Internet<h3>";
12     %>
13     <%= mensagem %>
14     <hr>
15     <%= dep %>
16   </body>
17 </html>
```



Você deve declarar uma variável ou um método antes de usá-lo.

O escopo de uma declaração é geralmente o arquivo JSP, mas se for incluído outros arquivos com a diretiva *include*, o escopo se expande para o arquivo incluído.

## Expressões

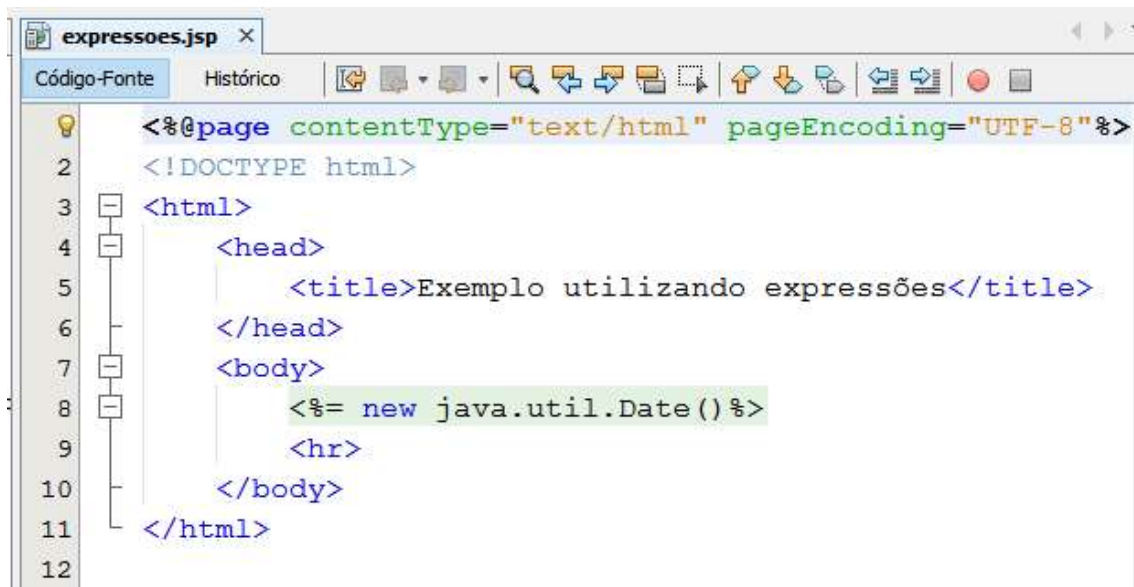
Usadas entre `<%= e %>`. Podem conter alguma expressão válida da linguagem de script usada pela página (Java).

Em expressões não utilizamos ponto-e-vírgula no final, mesmo sendo Java a linguagem de script.

A sintaxe para este elemento de criação de scripts é a seguinte:

`<%= expressão %>`

## Exemplo



## Resultado



A expressão Java é avaliada da esquerda para a direita, convertida em String e depois inserida na página.

Essa avaliação é feita em tempo de execução.

Para construir uma expressão em JSP você pode colocar entre as *tags* qualquer expressão definida na especificação da Linguagem Java.

Ao contrário dos *scriptlets* (que veremos a seguir), uma expressão não aceita ponto e vírgula e define somente uma expressão da Linguagem.

### Comentários

Existem dois tipos de comentários que podem ser usados em uma página JSP:

#### Comentário de Conteúdo (HTML)

Transmitidos de volta para o navegador como parte da resposta de JSP.

São visíveis na visualização do código-fonte da página.

Comentários de conteúdo possuem a seguinte sintaxe:

```
<!-- comentário -->
```

#### Exemplo

```
<!--  
    Esta página gera uma lista simples de produtos  
--!>
```

#### Comentários JSP (Java)

Não são enviados para o cliente e são visíveis apenas nos arquivos-fonte JSP.

O corpo do comentário é ignorado pelo container JSP.

Os comentários JSP podem assumir dois tipos de sintaxe:

#### Comentário para uma única linha

```
<% //comentário --%>
```

#### Exemplo

```
// Lista simples de produtos
```

ou

## Comentário para múltiplas linhas

```
/*
    for(int i=1; i<=10; i++){
        String produto = "P" + i + "<br>";
        out.println(produto);
    }
*/
```

A sintaxe anterior é introduzida dentro da página através de scriptlets, usando a sintaxe de comentário da linguagem Java.

## Exemplo completo

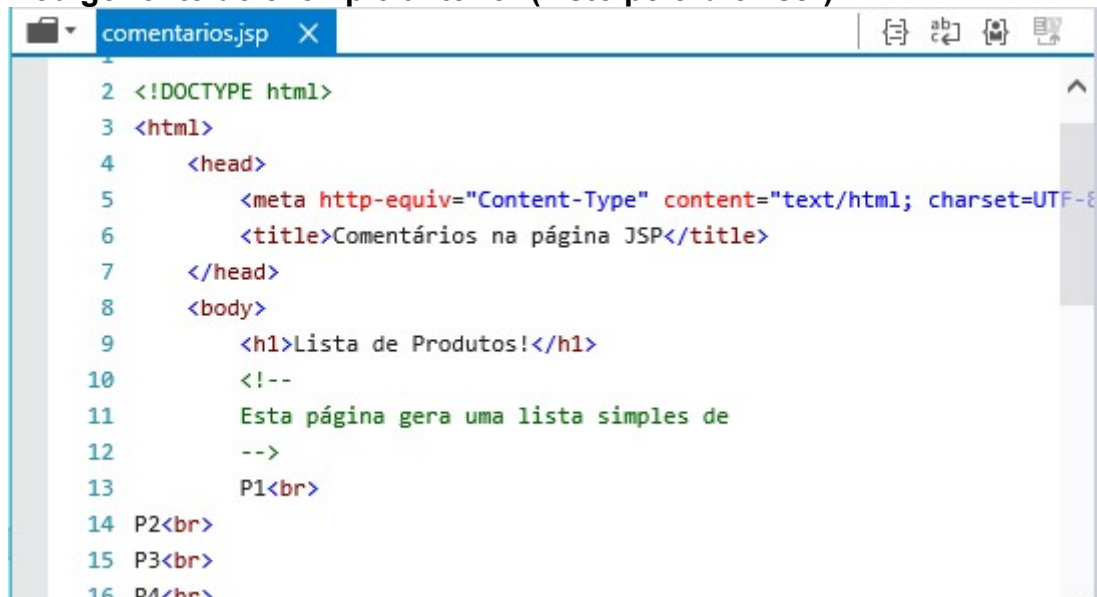


```
comentarios.jsp x
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"%>
6      <title>Comentários na página JSP</title>
7  </head>
8  <body>
9      <h1>Lista de Produtos!</h1>
10     <!--
11     Esta página gera uma lista simples de
12     -->
13     <%
14         // Lista simples de produtos
15         for(int i=1; i<=10; i++){
16             out.println("P"+i+"<br>");
17         }
18         /*
19         for(int i=1; i<=10; i++){
20             String produto= "P" + i + "<br>";
21             out.println("P"+i+"<br>");
22         }
23         */
24     %>
25 </body>
26 </html>
```

## Resultado do exemplo anterior



## Código fonte do exemplo anterior (visto pelo browser)



```
1
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6     <title>Comentários na página JSP</title>
7   </head>
8   <body>
9     <h1>Lista de Produtos!</h1>
10    <!--
11     Esta página gera uma lista simples de
12     -->
13    P1<br>
14    P2<br>
15    P3<br>
16    P4<br>
```

### Diretivas

Usadas para fornecer informações especiais ao container JSP sobre a página que lhe foi enviada quando esta é compilada gerando um *servlet*.

#### import

Permite que seja especificado qual pacote deverá ser importado.

É o único atributo que pode aparecer mais de uma vez em uma mesma página JSP.

Exemplo:

import="pacote.classe"

ou...

import="pacote.classe1, pacote.classe2,..."

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

#### include

A diretiva *include* permite que sejam incluídos arquivos na hora em que a página JSP é transformada em um *servlet*.

A sintaxe para o uso de uma diretiva *include* pode ser vista a seguir:

<%@ include file="url relativa" %>

Pode ser implementada em situações como:

Muitos *sites* possuem uma barra de navegação em cada página.



Devido a problemas com *frames* HTML isto é normalmente implementado com uma tabela repetindo o código HTML referente à tabela em cada página do *site*. Com o uso da diretiva `include`, podemos minimizar esforços como mostra o exemplo a seguir

### Exemplo

```
titulo.html x
<!DOCTYPE html>
<html>
  <head>
    <title>Título da página</title>
    <meta charset="iso-8859-1">
    <meta name="viewport" content="width=device-widt
  </head>
  <body>
    <font face="verdana" size="1" color="#ffffcc">
    <a href="p1.jsp">
      Página 1
    </a> -
    <a href="p2.jsp">
      Página 2
    </a> -
    <a href="p3.jsp">
      Página 3
    </a>
  </font>
</body>
</html>

listar3.jsp x
<%@page contentType="text/html" pageEncoding="iso-8859-1"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; c
    <title>Aula de JSP</title>
  </head>
  <body>
    <%@include file="titulo.html" %>
    <hr>
    <%
      String mensagem="<h3>Curso Internet - DAI<h3>";
      out.println(mensagem);
    %>
  </body>
</html>
```

Resultado do exemplo anterior





## Linguagem Java dentro de páginas JSP

Basicamente você pode utilizar qualquer instrução em Java dentro de suas páginas.

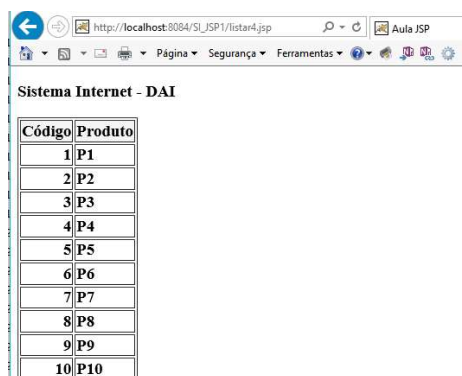
Ao contrário de outras tecnologias (ASP por exemplo) a linguagem Java é uma só, independente se é uma aplicação Desktop ou Web.

Estaremos utilizando neste tópico as estruturas de controle que podem ser utilizadas com JSP:

- for;
- while;
- do while.

### Exemplo com for

```
listar4.jsp x
6      <title>Aula JSP</title>
7      </head>
8      <body>
9      <%
10         String mensagem="<h3>Sistema Internet - DAI<h3>";
11         out.println(mensagem);
12     %>
13     <%
14         String tab="";
15         tab += "<table border=1>";
16         tab += "<tr bgcolor=whitesmoke>";
17         tab += "<td>Código</td>";
18         tab += "<td>Produto</td>";
19         tab += "</tr>";
20         for(int i=1;i<=10; i++){
21             tab += "<tr bgcolor=white>";
22             tab += "<td align=right>"+i+"</td>";
23             tab += "<td>P"+i+"</td>";
24             tab += "</tr>";
25         }
26         tab += "</table>";
27         out.println(tab);
28     %>
29     </body>
30 </html>
```



## Exemplo com while

```
listar5.jsp x
8      <body>
9      <%
10         String mensagem="<h3>Sistema Internet - DAI<h3>";
11         out.println(mensagem);
12     %>
13     <%
14         String tab="";
15         tab += "<table border=1>";
16         tab += "<tr bgcolor=whitesmoke>";
17         tab += "<td>Código</td>";
18         tab += "<td>Produto</td>";
19         tab += "</tr>";
20         int i=1;
21         while(i<=10){
22             tab += "<tr bgcolor=white>";
23             tab += "<td align=right>"+i+"</td>";
24             tab += "<td>P"+i+"</td>";
25             tab += "</tr>";
26             i++;
27         }
28         tab += "</table>";
29         out.println(tab);
30     %>
31 </body>
32 </html>
```

## Exemplo com do..while

```
listar6.jsp x
8      <body>
9      <%
10         String mensagem="<h3><h3>";
11         out.println(mensagem);
12     %>
13     <%
14         String tab="";
15         tab += "<table border=1>";
16         tab += "<tr bgcolor=whitesmoke>";
17         tab += "<td>Código</td>";
18         tab += "<td>Produto</td>";
19         tab += "</tr>";
20         int i=1;
21         do{
22             tab += "<tr bgcolor=white>";
23             tab += "<td align=right>"+i+"</td>";
24             tab += "<td>P"+i+"</td>";
25             tab += "</tr>";
26             i++;
27         }while(i<=10);
28         tab += "</table>";
29         out.println(tab);
30     %>
31 </body>
32 </html>
```

## Exemplo com if

### decisao1.jsp

```
1 <%@ page import="java.util.Date" %>
2 <html>
3   <body>
4     <%
5       String msg = "";
6       Date agora = new Date();
7       int hora = agora.getHours();
8       if ((hora >= 5) && (hora < 12)) {
9         msg = "Bom Dia";
10      } else if ((hora >= 12) && (hora < 18)) {
11        msg = "Boa Tarde";
12      } else if ((hora >= 18) && (hora < 24)) {
13        msg = "Boa Noite";
14      } else {
15        msg = "Madrugada";
16      }
17      out.print(msg);
18    %>
19  </body>
20 </html>
```

## Exemplo com switch

### decisao2.jsp

```
1 <%@ page import="java.util.Date" %>
2 <html>
3   <body>
4     <%
5       String msg = "";
6       Date agora = new Date();
7       int mesAtual = (agora.getMonth()+1);
8       String mes;
9       switch (mesAtual) {
10        case 1: mes="Janeiro"; break;
11        case 2: mes="Fevereiro"; break;
12        case 3: mes="Março"; break;
13        case 4: mes="Abril"; break;
14        case 5: mes="Maio"; break;
15        case 6: mes="Junho"; break;
16        case 7: mes="Julho"; break;
17        case 8: mes="Agosto"; break;
18        case 9: mes="Setembro"; break;
19        case 10: mes="Outubro"; break;
20        case 11: mes="Novembro"; break;
21        default: mes="Dezembro"; break;
22      }
23      out.print("Mês atual: " + mes);
24    %>
25  </body>
26 </html>
```

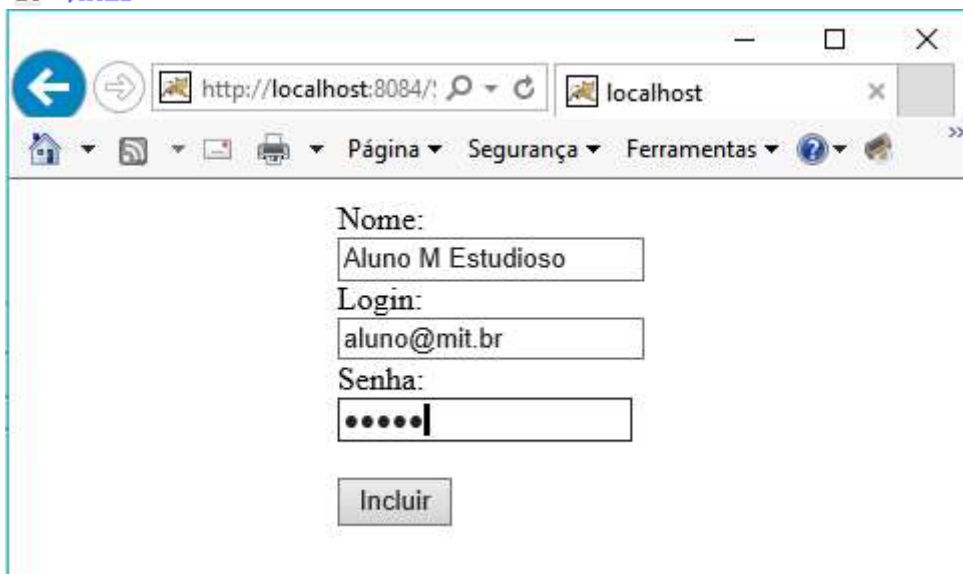
## Introdução Formulários

Uma página JSP, da mesma forma que um *servlet*, pode usar o objeto referenciado pela variável *request* para obter os valores dos parâmetros de um formulário.

O exemplo seguinte mostra uma página JSP com formulário.

### incluirusuario.jsp

```
1 <%page contentType="text/html"%>
2 <html>
3   <body>
4     <table border="0" align="center">
5       <tr>
6         <td>
7
8           <form action="salvarusuario.jsp" method="POST">
9             Nome:<br><input type="text" name="nome" value="" /><br>
10            Login:<br><input type="text" name="login" value="" /><br>
11            Senha:<br><input type="password" name="senha" value="" /><br>
12            <br><input type="submit" value="Incluir" name="incluir" />
13          </form>
14        </td>
15      </tr>
16    </table>
17  </body>
18 </html>
```



Nome:  
Aluno M Estudioso

Login:  
aluno@mit.br

Senha:  
.....

Incluir

### Recebendo informações do formulário

Ao clicar no botão Incluir do formulário anterior

Note que um *scriptlet* é usado para obter os valores dos parâmetros contidos no formulário.

### salvarusuario.jsp

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4   </head>
5   <body>
6     <%
7       String nome = "";
8       String login = "";
9       String senha = "";
10      nome = request.getParameter("nome");
11      login = request.getParameter("login");
12      senha = request.getParameter("senha");
13      out.println("Nome: " + nome + "<br>");
14      out.println("Login: " + login + "<br>");
15      out.println("Senha: " + senha + "<br>");
16    %>
17  </body>
18 </html>
```

nome: Aluno M Estudioso  
 login: aluno@mit.br  
 senha: 12345  
 incluir: Incluir

### Obtendo todos os parâmetros do formulário

Como o método `getParameterNames()` retornamos uma referência a um objeto *Enumeration*.

É preciso importar o pacote `java.util`, por meio da diretiva `page` para utilizar a classe *Enumeration*.

#### salvarusuario.jsp

```

1 <%@ page import="java.util.*" %>
2 <html>
3   <head>
4     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5   </head>
6   <body>
7     <%
8       Enumeration campos = request.getParameterNames();
9       while (campos.hasMoreElements()) {
10         String campo = (String) campos.nextElement();
11         String valor = request.getParameter(campo);
12         out.println(campo + ": " + valor + "<br>");
13       }
14     %>
15   </body>
16 </html>

```

### Avaliação da Aula

Criar 3 formulários

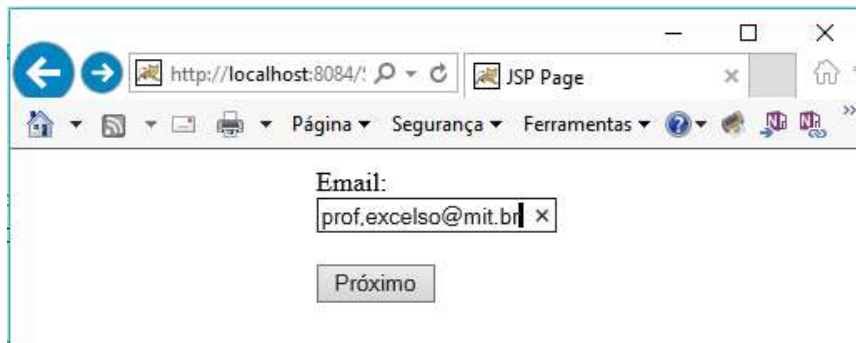
#### Primeiro Formulário

1- No primeiro você entrará apenas com o nome de uma pessoa.

Nome:

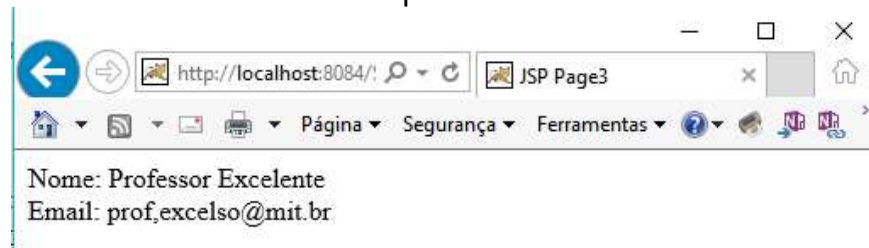
#### Segundo Formulário

2- No segundo você entrará apenas com o email da pessoa.



### Terceiro Formulário

2- No Terceiro deverá ser apresentado o nome e o email da pessoa.



Dica: Utilize campo oculto no segundo formulário.

```
<html>
  <body>
    <%
      String nome = "";
      nome = request.getParameter("nome");
    %>
    <table border="0" align="center">
      <tr><td>
        <form action="listarpessoaemail.jsp" method="POST">
          <input type="hidden" name="nome1" value = "<%= nome %>" />
          Email:<br><input type="text" name="email" value="" /><br>
          <br><input type="submit" value="listar" name="listar" />
        </form>
      </td></tr>
    </table>
  </body>
</html>
```

**Envie os 3 arquivos zipados para análise e liberação da próxima aula**

### Observação

Neste exemplo, utilizando request, é necessário utilizar um campo oculto, pois o request só pega valores do formulário anterior.

Ou seja, no terceiro formulário, via request não dá para obter informações do primeiro, mas apenas do segundo.