




# JSP – Java Sever Page

## Curso Técnico em Informática

### JSTL

Prof. William Geraldo Sallum



# JSP - JSTL

- **JSTL**

“Uma página **JSTL** é uma página JSP contendo um conjunto de tags JSTLs”.

Características:

- É uma coleção de bibliotecas, as quais permitem escrever páginas JSPs sem **código** Java.
- Aumenta da legibilidade do **código**.
- Promove a interação entre desenvolvedores e web-designers.

## JSP - JSTL

- O JSTL - Standard Tag Library representa um conjunto de tags para simplificar o desenvolvimento do JSP.
  - **Desenvolvimento Rápido:** O JSTL fornece muitas tags que simplificam o JSP.
  - **Reutilização de código:** Podemos usar as tags JSTL em várias páginas.
  - **Não é necessário usar tag de scriptlet:** Evita o uso de tag de scriptlet.

## JSP - JSTL

- O JSTL fornece principalmente cinco tipos de tags:

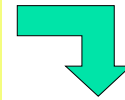
Tag Name	Descrição
Tags principais	A tag principal do JSTL fornece suporte variável, gerenciamento de URL, controle de fluxo, etc. O URL da tag principal é <a href="http://java.sun.com/jsp/jstl/core">http://java.sun.com/jsp/jstl/core</a> . O prefixo da tag core é <code>c</code> .
Tags de função	As tags de funções fornecem suporte para manipulação de string e comprimento de string. A URL para as tags de funções é <a href="http://java.sun.com/jsp/jstl/functions">http://java.sun.com/jsp/jstl/functions</a> e o prefixo é <code>fn</code> .
Formatando tags	As tags Formatação fornecem suporte para formatação de mensagens, formatação de números e datas, etc. O URL para as tags Formatação é <a href="http://java.sun.com/jsp/jstl/fmt">http://java.sun.com/jsp/jstl/fmt</a> e o prefixo é <code>fmt</code> .
Tags XML	As tags XML fornecem controle de fluxo, transformação etc. A URL para as tags XML é <a href="http://java.sun.com/jsp/jstl/xml">http://java.sun.com/jsp/jstl/xml</a> e o prefixo é <code>x</code> .
Tags SQL	As tags SQL do JSTL fornecem suporte ao SQL. A URL para as tags SQL é <a href="http://java.sun.com/jsp/jstl/sql">http://java.sun.com/jsp/jstl/sql</a> e o prefixo é <code>sql</code> .

## JSP - JSTL (Tags principais)

- **<c: out>** - semelhante à tag de expressão JSP, mas só pode ser usada com expressão

Cabeçalho

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title> Exemplo de tag </title>
  </head>
  <body>
    <c:out value = "${'Bem-vindo ao JSTL'}" />
  </body>
</html>
```



Bem-vindo ao JSTL

## JSP - JSTL (Tags principais)

- **<c: import>** - semelhante ao jsp 'include', com um recurso adicional de incluir o conteúdo de qualquer recurso no servidor ou fora do servidor.

```
<c:import var="data" url="http://www.javatpoint.com" />
<c:out value="${data}" />
```

- **<c: set>** - defini o resultado de uma expressão avaliada em um 'escopo'. É útil porque avalia a expressão

```
<c:set var="consumo" scope="session" value="${4000*4}" />
<c:out value="${consumo}" />
```

- **<c: if>** - usada para testar a condição e exibe o conteúdo do corpo, se a expressão avaliada for verdadeira

```
<c:set var="consumo" scope="session" value="${4000*4}" />
<c:if test="${consumo > 8000}">
  <p>Meu consumo é: <c:out value="${consumo}" /><p>
</c:if>
```

## JSP - JSTL (Tags principais)

**<c: choose><c: when><c: otherwise>** - Funciona como uma declaração de switch Java

```
<c:set var="consumo" scope="session" value="{4000*4}"/>
<p>Seu consumo é: <c:out value="{consumo}"/></p>
<c:choose>
  <c:when test="{consumo <= 1000}">
    Consumo não é bom.
  </c:when>
  <c:when test="{consumo > 10000}">
    Consumo é muito bom.
  </c:when>
  <c:otherwise>
    Consumo é indeterminado...
  </c:otherwise>
</c:choose>
```

Seu consumo é: 16000 Consumo é muito bom.
--

## JSP - JSTL (Tags principais)

**<c: forEach>** - é uma tag de iteração usada para repetir o conteúdo do corpo aninhado por um número fixo de vezes ou pela coleção.

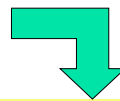
```
<c:forEach var="j" begin="1" end="3">
  <p>Item <c:out value="{j}"/></p>
</c:forEach>
```

**<c: forTokens>** - é usado para dividir uma cadeia em tokens e percorrer cada um dos tokens para gerar a saída.

```
<c:forTokens items="Mayumi-Seiji-Harumi-Luiz" delims="-" var="name">
  <p><c:out value="{name}"/></p>
</c:forTokens>
```

**<c: param>** - permite que o parâmetro de solicitação de URL adequado seja especificado na URL e execute automaticamente qualquer codificação de URL necessária.

```
<c:url value="/index1.jsp" var="completeURL"/>
  <c:param name="idade" value="57"/>
  <c:param name="user" value="Sallum"/>
</c:url>
${completeURL}
```



//localhost:8080/WebApp1/index1.jsp?idade=57&user=Sallum

## JSP - JSTL (Tags principais)

**<c: redirect>** - redireciona o navegador para um novo URL. Suporta as URLs relativas ao contexto e a tag <c: param>.

```
<c:set var="url" value="0" scope="request"/>
```

```
<c:if test="{url<1}">
```

```
  <c:redirect url="http://www.cefetmg.br"/>
```

```
</c:if>
```

```
<c:if test="{url>1}">
```

```
  <c:redirect url="http://www.ufmg.br"/>
```

```
</c:if>
```

**<c: url>** - cria uma URL com o parâmetro de consulta opcional. É usado para codificação de url ou formatação de url. Esta tag executa automaticamente a operação de regravação de URL.

```
<c:url var="reg" value="/RegisterDao.jsp"/>
```

```
  <c:param name="comando" value="load"/>
```

```
</c:url>
```

## JSP - JSTL (Tags de funções)

- Tags de funções JSTL é um conjunto de funções padrão, sendo a maioria destas funções, funções comuns de manipulação de sequencias.

Cabeçalho



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
```

## JSP - JSTL (Tags de função)

### Lista de Tags de Função JSTL

Funções JSTL	Descrição
<b>fn: contains ()</b>	Ele é usado para testar se uma string de entrada contém a substring especificada em um programa.
<b>fn: containsIgnoreCase ()</b>	Ele é usado para testar se uma string de entrada contém a substring especificada como um modo insensível a maiúsculas e minúsculas.
<b>fn: endsWith ()</b>	É usado para testar se uma string de entrada termina com o sufixo especificado.
<b>fn: escapeXml ()</b>	Ele escapa dos caracteres que seriam interpretados como marcação XML.
<b>fn: indexOf ()</b>	Ele retorna um índice dentro de uma string da primeira ocorrência de uma substring especificada.
<b>fn: trim ()</b>	Remove os espaços em branco de ambas as extremidades de uma string.
<b>fn: startsWith ()</b>	Ele é usado para verificar se a string dada é iniciada com um valor de string específico.
<b>fn: split ()</b>	Ele divide a string em uma matriz de substrings.
<b>fn: toLowerCase ()</b>	Converte todos os caracteres de uma string em minúscula.
<b>fn: toUpperCase ()</b>	Converte todos os caracteres de uma string em letras maiúsculas.
<b>fn: substring ()</b>	Ele retorna o subconjunto de uma string de acordo com a posição inicial e final dada.
<b>fn: substringAfter ()</b>	Ele retorna o subconjunto de string após uma subsequência específica.
<b>fn: substringBefore ()</b>	Ele retorna o subconjunto de string antes de uma subsequência específica.
<b>fn: length ()</b>	Retorna o número de caracteres dentro de uma string ou o número de itens em uma coleção.
<b>fn: replace ()</b>	Substitui toda a ocorrência de uma string por outra sequência de string.

## JSP - JSTL (Tags de função)



**<fn: contains()>** - usado para testar se a string contém a substring especificada.

```
<c:set var="String" value="Bem vindo a disciplina de AW"/>
<c:if test="${fn:contains(String, 'AW')}">
  <p>Encontrada a String AW<p>
</c:if>
<c:if test="${fn:contains(String, 'aw')}">
  <p>Encontrada a String aw<p>
</c:if>
```

**<fn: replace()>** - procura em uma string de entrada e a substitui pela string fornecida.

```
< c: set var = "autor" valor = "William Sallum" />
< c: set var = "string" valor = "pqr meus alunos!" />
$ {fn: replace (autor, "William", "Mayumi")}
$ {fn: replace (string, "pqr", "olá")}
```

Mayumi Sallum Olá meus alunos!
-----------------------------------

## JSP - JSTL (Tags de função)

**<fn: length()>** - usado para calcular o comprimento da string e descobrir o número de elementos em uma coleção.


```
<c:set var="str1" value="Esta é a 1ª String" />
<c:set var="str2" value="Olá" />
```

Comprimento da String-1 é: 18  
 Comprimento da String-2 é: 3


Comprimento da String-1 é: \${fn:length(str1)}  
 Comprimento da String-2 é: \${fn:length(str2)}


**<fn: substring()>** - usado para retornar a subsequência de dados da string de entrada de acordo com as posições inicial e final especificadas.

```
<c:set var = "string" value = "Esta é a primeira string." />
$ {fn: substring (string, 5, 18)}
```



é a primeira s





## JSP - JSTL (Tags de função)

**<fn: toUpperCase()> <fn: toLowerCase>** - usados para retornar uma string como maiúsculas e minúsculas, respectivamente.

```
<c:set var="site" value="PROGRAMA" />
<c:set var="author" value="Prof. Sallum" />
```

Olá, Este \${fn:toLowerCase(site)} foi desenvolvido  
 por \${fn:toUpperCase(author)}.

Olá, Este programa foi desenvolvido por PROF. SALLUM.

**<fn: split()> <fn: join>** - divide a string (split) em uma matriz de substrings e (join) converte um array em string.

```
<c:set var = "str1" value = "Bem-vindo-à-programação-JSP." />
<c:set var = "str2" value = "$ {fn: split (str1, '-')} " />
<c:set var = "str3" value = "$ {fn: join (str2, ' ')} " />
```

**<p>** String-3: \$ {str3} **</p>**

```
<c:set var = "str4" value = "$ {fn: split (str3, ' ')} " />
<c:set var = "str5" value = "$ {fn: join (str4, '-')} " />
```

**<p>** String-5: \$ {str5} **</p>**

String-3: Bem vindo à programação JSP.  
 String-5: Bem-vindo-à-programação-JSP.



## JSP - JSTL (Tags de função)

**<fn: indexOf>** - usado para determinar o índice da string especificada na substring.

```
<c:set var="string1" value="Esta é a 1ª String."/>
<c:set var="string2" value="Esta é a <xyz> 2ª String.</xyz>" />
<p>Index-1 : ${fn:indexOf(string1, "1ª ")}</p>
<p>Index-2 : ${fn:indexOf(string2, "2ª ")}</p>
```

Index-1: 9  
Index-2: 15

**<fn: escapeXml>** - escapa os caracteres que seriam interpretados como marcação XML.

```
<c:set var="string1" value="Esta é a 1ª String."/>
<c:set var="string2" value="Esta é a <xyz>2ª String.</xyz>" />
<p>Com a função escapeXml():</p>
<p>string-1 : ${fn:escapeXml(string1)}</p>
<p>string-2 : ${fn:escapeXml(string2)}</p>
<p>Sem a função escapeXml():</p>
<p>string-1 : ${string1}</p>
<p>string-2 : ${string2}</p>
```

**Com a função escapeXml():**  
string-1 : Esta é a 1ª String.  
string-2 : Esta é a <b>2ª String.</b>  
**Sem a função escapeXml():**  
string-1 : Esta é a 1ª String.  
string-2 : Esta é a 2ª String.

## JSP - JSTL (Tags de formatação)

- As tags de formatação fornecem suporte para formatação de mensagens, formatação de números e datas, etc.
- As tags de formatação do JSTL são usadas para sites da Web internacionalizados para exibir e formatar texto, a hora, a data e os números.

Cabeçalho



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
```



## JSP - JSTL (Tags de formatação)

### ■ Lista de Tags de Formatação JSTL

Tags de formatação	Descrições
<b>fmt: parseNumber</b>	Ele é usado para analisar a representação de string de uma moeda, porcentagem ou número.
<b>fmt: timeZone</b>	Ele especifica uma ação de análise aninhada em seu corpo ou o fuso horário para qualquer formatação de hora.
<b>fmt: formatNumber</b>	É usado para formatar o valor numérico com formato ou precisão específicos.
<b>fmt: parseDate</b>	Ele analisa a representação de sequência de caracteres de uma hora e data.
<b>fmt: bundle</b>	Ele é usado para criar os objetos ResourceBundle que serão usados pelo corpo da tag.
<b>fmt: setTimeZone</b>	Ele armazena o fuso horário dentro de uma variável de configuração de fuso horário.
<b>fmt: setBundle</b>	Ele carrega o pacote de recursos e os armazena em uma variável de configuração de pacote configurável ou na variável com escopo definido.
<b>fmt: message</b>	Exibe uma mensagem internacionalizada.
<b>fmt: formatDate</b>	Ele formata a hora e / ou data usando o padrão e os estilos fornecidos.

## JSP - JSTL (Tags de formatação)

**<fmt: parseNumber>** - é usada para analisar a representação de string de uma moeda, porcentagem ou número.

**<h3>**Um exemplo da tag **fmt:parseNumber**:**</h3>**

**<c:set** var="montante" value="786,970" **/>**

**<fmt:parseNumber** var="j" type="number" value="{monta ne}" **/>**

**<p><i>**O montate é:**</i>**




**<c:out** value="{j}" **/></p>**

**<fmt:parseNumber** var="j" integerOnly="true" type="number" value="{montante}" **/>**

**<p><i>**Montante inteiro:**</i>**

**<c:out** value="{j}" **/></p>**

O montate é: 786.97
Montante inteiro: 786



## JSP - JSTL (Tags de formatação)

**<fmt:formatNumber>** - usada para formatar o valor numérico usando o formato ou a precisão específicos.

1 - Número formatado: R\$ 9.850,14
2 - Número formatado: 9.850,141
3 - Número formatado: 850,141
4 - Número formatado: 9.850,14115
5 - Número formatado: 5.014%
6 - Número formatado: R\$9.850,141

```

<h3>Formatação de número:</h3>
<c:set var="Montante" value="9850,14115" />
<p> Número formatado - 1:
  <fmt:formatNumber value="{Montante}" type="currency" /></p>
<p> Número formatado - 2: :
  <fmt:formatNumber type="number" groupingUsed="true" value="{Montante}" /></p>
<p> Número formatado - 3:
  <fmt:formatNumber type="number" maxIntegerDigits="3" value="{Montante}" /></p>
<p> Número formatado - 4:
  <fmt:formatNumber type="number" maxFractionDigits="6" value="{Montante}" /></p>
<p> Número formatado - 5:
  <fmt:formatNumber type="percent" maxIntegerDigits="4" value="{Montante}" /></p>
<p> Número formatado - 6:
  <fmt:formatNumber type="number" pattern="R$###.###" value="{Montante}" /></p>
  
```


## JSP - JSTL (Tags de formatação)

**<fmt:parseDate>** - usado para formatar a hora e a data de acordo com um padrão de formatação personalizado.

```

<h3>Parsed Date:</h3>
<c:set var="date" value="28-05-1961" />
<fmt:parseDate value="{date}" var="parsedDate" pattern="dd-MM-yyyy" />
<c:out value="{parsedDate}" />
  
```

Sun May 28 00:00:00 BRT 1961



Formatação da hora: 17:43:19  
Formatação da data: 31/10/2021  
Formatação de data e hora: 31-10-2021 (304) PM  
Formatação de data e hora em estilo reduzido: 31/10/21 17:43  
Formatação de data e hora em estilo médio: 31/10/2021 17:43:19  
Formatação de data e hora em estilo longo: 31 de Outubro de 2021 17h43min19s BRT






## JSP - JSTL (Tags de Formatação)

**<fmt:formatDate>** - usado para formatar a hora e a data de acordo com o padrão de formatação personalizado.


```

<h2>Diferentes formatos de data</h2>
<c:set var="Date" value="%=new java.util.Date()%" />
<p> Formatação da hora:
<fmt:formatDate type="time" value="${Date}" /> </p>
<p> Formatação da data :
<fmt:formatDate type="date" value="${Date}" /> </p>
<p> Formatação de data e hora :
<fmt:formatDate type="both" value="${Date}" /> </p>
<p> Formatação de data e hora em estilo reduzido:
<fmt:formatDate type="both" dateStyle="short" timeStyle="short" value="${Date}"/>
</p>
<p> Formatação de data e hora em estilo médio:
<fmt:formatDate type="both" dateStyle="medium" timeStyle="medium"
value="${Date}" /> </p>
<p> Formatação de data e hora em estilo longo:
<fmt:formatDate type="both" dateStyle="long" timeStyle="long"
value="${Date}" /> </p>

```

## JSP - JSTL (Tags XML)



- As tags JSTL XML são usadas para fornecer uma maneira centrada em JSP de manipular e criar documentos XML.
- As tags xml fornecem controle de fluxo, transformação, etc.

Cabeçalho

↓

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
```

## JSP - JSTL – (Tags XML)

### ■ Tags XML

Tags XML	Descrições
<b>x:out</b>	Semelhante à tag <code>&lt;% = ...&gt;</code> , mas para expressões XPath.
<b>x:parse</b>	Ele é usado para analisar os dados XML especificados no corpo da tag ou em um atributo.
<b>x:set</b>	É usado para definir uma variável para o valor de uma expressão XPath.
<b>x:choose</b>	É uma tag condicional que estabelece um contexto para operações condicionais mutuamente exclusivas.
<b>x:when</b>	É uma subtag de <code>x:choose</code> que inclui seu corpo se a condição avaliada for "verdadeira".
<b>x:otherwise</b>	É uma subtag de <code>x:choose</code> que executa somente se todas as condições anteriores avaliadas forem 'falsas'.
<b>x:if</b>	Ele é usado para avaliar a expressão XPath do teste e, se for verdade, processará o conteúdo do corpo.
<b>x:transform</b>	Ele é usado em um documento XML para fornecer a transformação XSL (Extensible Stylesheet Language).
<b>x:param</b>	Ele é usado junto com a tag de transformação para definir o parâmetro na folha de estilo XSLT.

## JSP - JSTL (Tags XML)

**<x: out>** - usada para exibir o resultado de uma expressão xml Path e gravar o resultado no objeto JSP gravador.

```
<h2> Informação Vegetal: </h2>
<c:set var = "vegetal">
<vegetais>
  <vegetal>
    <nome> cebola </nome>
    <preco> 40 / kg </preco>
  </vegetal>
  <vegetal>
    <nome> Batata </nome>
    <preco> 30 / kg </preco>
  </vegetal>
  <vegetal>
    <nome> Tomate </nome>
    <preco> 90 / kg </preco>
  </vegetal>
</vegetais>
</c:set>
```

```
<x: parse xml = "${vegetal}" var = "saida" />
<b> Nome do vegetal é </b>
<x: out select="${saida/vegetais/vegetal [1]
/ nome" /> <br>
<b> O preço da batata é </b>
<x: out select = "${saida/vegetais/vegetal [2]
/preco" />
```



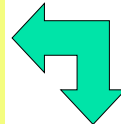
#### Informação Vegetal:

Nome do vegetal é   cebola  
O preço da batata é   30 / kg

## JSP - JSTL (Tags XML)

**<x: parse>** - usada para analisar os dados XML especificados no corpo da tag ou em um atributo.

```
<h2>Informações de livros:</h2>
<c:import var="bookInfo" url="livros.xml" />
<x:parse xml="{bookInfo}" var="saida" />
1o título de livros: <x:out select="$saida/livros/livro[1]/nome" /><br>
1o preço de livros: <x:out select="$saida/livros/livro[1]/preco" /><br>
2o título de livros: <x:out select=
"$saida/livros/livro[2]/nome" /><br>
2o preço de livros: <x:out select=
"$saida/livros/livro[2]/preco" /><br>
```



### Informações de livros:




1o título de livros: Internet e World Wide Web  
 1o preço de livros: 300  
 2o título de livros: Java Servlets JSP  
 2o preço de livros: 120

```
<livros>
<livro>
  <nome>Internet e World Wide Web</nome>
  <autor>Deitel; Deitel; Nieto</autor>
  <preco>300</preco>
</livro>
<livro>
  <nome>Java Servlets JSP</nome>
  <autor>Vijay Mukhi</autor>
  <preco>120</preco>
</livro>
</livros>
```

## JSP - JSTL (Tags XML)

**<x: set>** - usada para definir uma variável com o valor de uma expressão XPath.




```
<h3>Informações de Livros:</h3>
<c:set var="book">
  <books>
    <book>
      <name>Internet e World Wide Web</name>
      <author>Deitel; Deitel; Nieto</author>
      <price>300</price>
    </book>
    <book>
      <name>Java Servlets JSP</name>
      <author>Vijay Mukhi</author>
      <price>120</price>
    </book>
  </books>
</c:set>
<x:parse xml="{book}" var="output"/>
<x:set var="fragment" select="$output/books/book[2]/price"/>
<b> O preço do livro de Java Servlets JSP</b>:
<x:out select="$fragment" />
```

## JSP - JSTL (Tags XML)

**<x: choose>** - tag condicional que estabelece um contexto para operações condicionais mutuamente exclusivas.

<pre> &lt;h3&gt;Books Information:&lt;/h3&gt; &lt;c:set var="xmltext"&gt; &lt;books&gt;   &lt;book&gt;     &lt;name&gt;Three mistakes of my life&lt;/name&gt;     &lt;author&gt;Chetan Bhagat&lt;/author&gt;     &lt;price&gt;200&lt;/price&gt;   &lt;/book&gt;   &lt;book&gt;     &lt;name&gt;Tomorrow land&lt;/name&gt;     &lt;author&gt;Brad Bird&lt;/author&gt;     &lt;price&gt;2000&lt;/price&gt;   &lt;/book&gt; &lt;/books&gt; &lt;/c:set&gt; </pre>	<pre> &lt;x:parse xml="{xmltext}" var="output"/&gt; &lt;x:choose&gt;   &lt;x:when select="\$output//book/author =     'Chetan bhagat'"&gt;     Book is written by Chetan bhagat   &lt;/x:when&gt;   &lt;x:when select="\$output//book/author =     'Brad Bird'"&gt;     Book is written by Brad Bird   &lt;/x:when&gt;   &lt;x:otherwise&gt;     The author is unknown...   &lt;/x:otherwise&gt; &lt;/x:choose&gt; </pre>
---	---

## JSP - JSTL (Tags XML)

**<x: if>** - tag condicional simples que é usada para avaliar seu corpo se a condição fornecida for verdadeira.

<pre> h2&gt;Vegetable Information:&lt;/h2&gt; &lt;c:set var="vegetables"&gt; &lt;vegetables&gt;   &lt;vegetable&gt;     &lt;name&gt;onion&lt;/name&gt;     &lt;price&gt;40&lt;/price&gt;   &lt;/vegetable&gt;   &lt;vegetable&gt;     &lt;name&gt;Potato&lt;/name&gt;     &lt;price&gt;30&lt;/price&gt;   &lt;/vegetable&gt;   &lt;vegetable&gt;     &lt;name&gt;Tomato&lt;/name&gt;     &lt;price&gt;90&lt;/price&gt;   &lt;/vegetable&gt; &lt;/vegetables&gt; &lt;/c:set&gt; </pre>	<pre> &lt;x:parse xml="{vegetables}" var="output"/&gt; &lt;x:if select="\$output/vegetables/vegetable/price &lt; 100"&gt;   Vegetables prices are very low. &lt;/x:if&gt; </pre>
---	--