📚 Principais Tags da JSTL Core (`<c:...>`)

Todas as tags core vêm do namespace:

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

🔹 `<c:out>` – Exibe conteúdo de forma segura

- Evita erros se o valor for `null`.
- Escapa caracteres especiais HTML automaticamente.

<c:out value="${usuario.nome}" default="Desconhecido" />

🔹 `<c:set>` – Define valores em escopos

- Semelhante ao `setAttribute()` do Java.

<c:set var="nome" value="Patrick" scope="session"/>

🔹 `<c:remove>` – Remove variáveis de escopo

<c:remove var="nome" scope="session"/>

🔹 `<c:if>` – Estrutura condicional simples (como `if` em Java)

```
<c:if test="${usuario.idade >= 18}">
   <p>Maior de idade</p>
</c:if>
```

🔹 `<c:choose>`, `<c:when>`, `<c:otherwise>` – Substitui `if...else if...else`

```
<c:choose>
   <c:when test="${usuario.sexo == 'm'}">
```

```
        <p>Bem-vindo, senhor!</p>
    </c:when>
    <c:when test="${usuario.sexo == 'f'}">
        <p>Bem-vinda, senhora!</p>
    </c:when>
    <c:otherwise>
        <p>Olá!</p>
    </c:otherwise>
</c:choose>
```

🔹 `<c:forEach>` – Laço de repetição (equivale ao `for`/`foreach`)

Percorre arrays, listas, maps e ranges.

```
<c:forEach var="item" items="${produtos}">
    <p>${item.nome}</p>
</c:forEach>
```

Ou com contagem:

```
<c:forEach var="i" begin="1" end="5">
    <p>Item ${i}</p>
</c:forEach>
```

🔹 `<c:forTokens>` – Divide uma string por um delimitador

```
<c:forTokens items="Java,Python,C++,Go" delims="," var="linguagem">
    <p>${linguagem}</p>
</c:forTokens>
```

🔹 `<c:catch>` – Captura exceções para evitar que quebrem a página

```
<c:catch var="erro">
    <%-- código que pode lançar erro --%>
</c:catch>

<c:if test="${not empty erro}">
    <p>Erro capturado: ${erro}</p>
</c:if>
```

🔹 `<c:url>` – Cria URLs seguras (com session ID)

```
<c:url var="minhaUrl" value="/produtos.jsp">
   <c:param name="categoria" value="eletronicos"/>
</c:url>

<a href="${minhaUrl}">Ver Eletrônicos</a>
```

🔹 `<c:import>` – Importa conteúdo de outra URL (interna ou externa)

```
<c:import url="/header.jsp"/>
```

🔹 `<c:redirect>` – Redireciona para outra página

```
<c:redirect url="login.jsp"/>
```

✅ Exemplo Completo Usando Várias Tags

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

<c:set var="nome" value="${param.nome}" scope="request"/>
<c:if test="${empty nome}">
   <p>Nome não informado.</p>
</c:if>

<c:if test="${not empty nome}">
   <p>Olá, <c:out value="${nome}"/>!</p>
</c:if>

<c:set var="linguagens" value="Java,Python,C++,Go"/>
<c:forTokens items="${linguagens}" delims="," var="ling">
   <p>${ling}</p>
</c:forTokens>
```

📚 Principais Tags da Biblioteca SQL da JSTL

Todas essas tags fazem parte da URI:

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

🔹 `<sql:setDataSource>` – Define a fonte de dados (conexão com o banco)

Pode ser usada com conexão direta (JDBC) ou com `DataSource` do servidor.

Exemplo com JDBC direto:

```
<sql:setDataSource
   var="conexao"
   driver="com.mysql.cj.jdbc.Driver"
   url="jdbc:mysql://localhost:3306/meubanco"
   user="root"
   password="senha" />
```

🔹 `<sql:query>` – Realiza consultas (SELECT)

```
<sql:query var="resultado" dataSource="${conexao}">
   SELECT * FROM clientes
</sql:query>

<c:forEach var="linha" items="${resultado.rows}">
   <p>Nome: ${linha.nome}</p>
</c:forEach>
```

🔹 `<sql:update>` – Executa comandos de modificação (INSERT, UPDATE, DELETE)

```
<sql:update dataSource="${conexao}">
   INSERT INTO clientes (nome, idade) VALUES ('João', 30)
</sql:update>
```

Com parâmetros dinâmicos:

```
<sql:update dataSource="${conexao}">
   INSERT INTO clientes (nome, idade) VALUES (?, ?)
   <sql:param value="${param.nome}" />
   <sql:param value="${param.idade}" />
</sql:update>
```

🔹 `<sql:param>` – Insere parâmetros com segurança (evita SQL Injection)

É usado dentro de `<sql:query>` ou `<sql:update>` para passar valores dinamicamente:

```
<sql:query var="resultado" dataSource="${conexao}">
   SELECT * FROM clientes WHERE nome = ?
   <sql:param value="${param.nome}" />
</sql:query>
```

🔹 `<sql:transaction>` – Executa várias instruções em uma única transação

Garante que todas as instruções dentro dela sejam executadas com sucesso ou revertidas.

```
<sql:transaction dataSource="${conexao}">
   <sql:update>
      INSERT INTO pedidos (cliente_id, total) VALUES (1, 500)
   </sql:update>

   <sql:update>
      UPDATE estoque SET quantidade = quantidade - 1 WHERE produto_id = 10
   </sql:update>
</sql:transaction>
```

✅ Exemplo Completo com Tudo

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

<sql:setDataSource
   var="conexao"
   driver="com.mysql.cj.jdbc.Driver"
   url="jdbc:mysql://localhost:3306/meubanco"
   user="root"
   password="123456" />

<sql:query var="clientes" dataSource="${conexao}">
   SELECT * FROM clientes
</sql:query>
```

```
<h2>Lista de Clientes</h2>
<ul>
   <c:forEach var="cliente" items="${clientes.rows}">
      <li>${cliente.nome} - ${cliente.idade} anos</li>
   </c:forEach>
</ul>
```

📚 URI da Biblioteca XML

Antes de usar, você deve declarar a biblioteca XML no topo da página JSP:

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

🔹 `<x:parse>` – Faz o *parsing* (análise) de um documento XML

Usado para transformar um conteúdo XML em um objeto manipulável por JSTL.

```
<x:parse var="doc">
  <cliente>
    <nome>João</nome>
    <idade>30</idade>
  </cliente>
</x:parse>
```

🔹 `<x:out>` – Imprime o valor de um elemento do XML

```
<x:out select="$doc/cliente/nome" />
```

Resultado: João

🔹 `<x:set>` – Cria uma variável a partir de uma expressão XPath

Você pode armazenar parte do documento para usar depois.

```
<x:set var="nomeCliente" select="$doc/cliente/nome" />
<p>Nome: ${nomeCliente}</p>
```

🔹 `<x:if>` – Condicional baseada em XPath

```
<x:if select="$doc/cliente/idade > 18">
  <p>Cliente é maior de idade.</p>
</x:if>
```

🔹 Exemplo Completo com Iteração (`forEach` aninhado + XML)

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

<%! String xmlDados =
"<clientes>" +
"  <cliente><nome>João</nome><idade>30</idade></cliente>" +
"  <cliente><nome>Maria</nome><idade>25</idade></cliente>" +
"</clientes>"; %>

<x:parse var="clientes" xml="${xmlDados}" />

<c:forEach var="c" items="${clientes.selectNodes('/clientes/cliente')}">
  <p>
    Nome: <x:out select="$c/nome"/> <br>
    Idade: <x:out select="$c/idade"/>
  </p>
</c:forEach>
```

✳️ Observações importantes

- O JSTL XML usa **XPath** para navegar e acessar dados dentro do XML.
- Para projetos modernos, o uso de JSTL XML caiu bastante. Hoje é mais comum manipular XML via Java (com DOM, JAXB ou Jackson XML) e enviar os dados já tratados para a JSP.