DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Módulo 3
Java Server Pages (JSP)

Delano Medeiros Beder
delano@dc.ufscar.br

Material baseado nos slides gentilmente disponibilizados pelo prof. Daniel Lucrédio

“Alô Mundo”

Todo JSP é um *Servlet*

- É apenas uma notação diferente

Servlet

- Código Java, com texto (HTML), dentro

JSP

- Código texto (HTML), com Java dentro

“Alô Mundo” (index.jsp)

```jsp
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="java.util.Date" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Alô Mundo</h1>

        <h1> <%= new Date() %></h1>
    </body>
</html>
```

Servlet resultante (index_jsp.java)

index_jsp.java ×

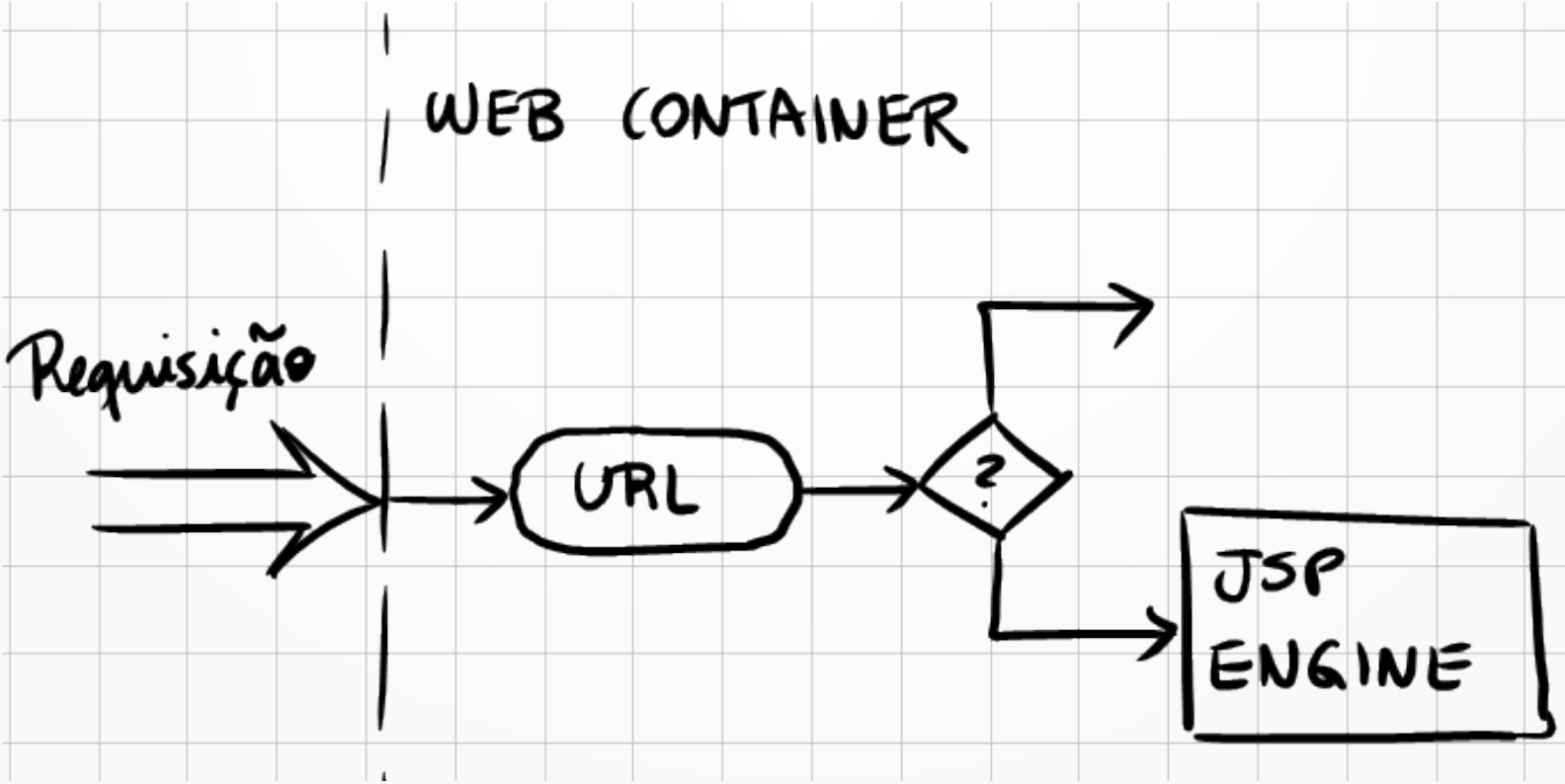
```
    pageContext = _jspxFactory.getPageContext(this, request, response,
                null, true, 8192, true);
    _jspx_page_context = pageContext;
    application = pageContext.getServletContext();
    config = pageContext.getServletConfig();
    session = pageContext.getSession();
    out = pageContext.getOut();
    _jspx_out = out;

    out.write("\n");
    out.write("\n");
    out.write("<!DOCTYPE html>\n");
    out.write("<html>\n");
    out.write("\n");
    out.write("<head>\n");
    out.write("    <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
    out.write("    <title>JSP Page</title>\n");
    out.write("</head>\n");
    out.write("\n");
    out.write("<body>\n");
    out.write("    <h1>Alô Mundo</h1>\n");
    out.write("    \n");
    out.write("    <h1> ");
    out.print( new Date() );
    out.write("</h1>\n");
    out.write("</body>\n");
    out.write("\n");
    out.write("</html>\n");
  } catch (java.lang.Throwable t) {
    if (!(t instanceof javax.servlet.jsp.SkipPageException)){
      out = _jspx_out;
      if (out != null && out.getBufferSize() != 0)
        try {
```

“Alô Mundo”

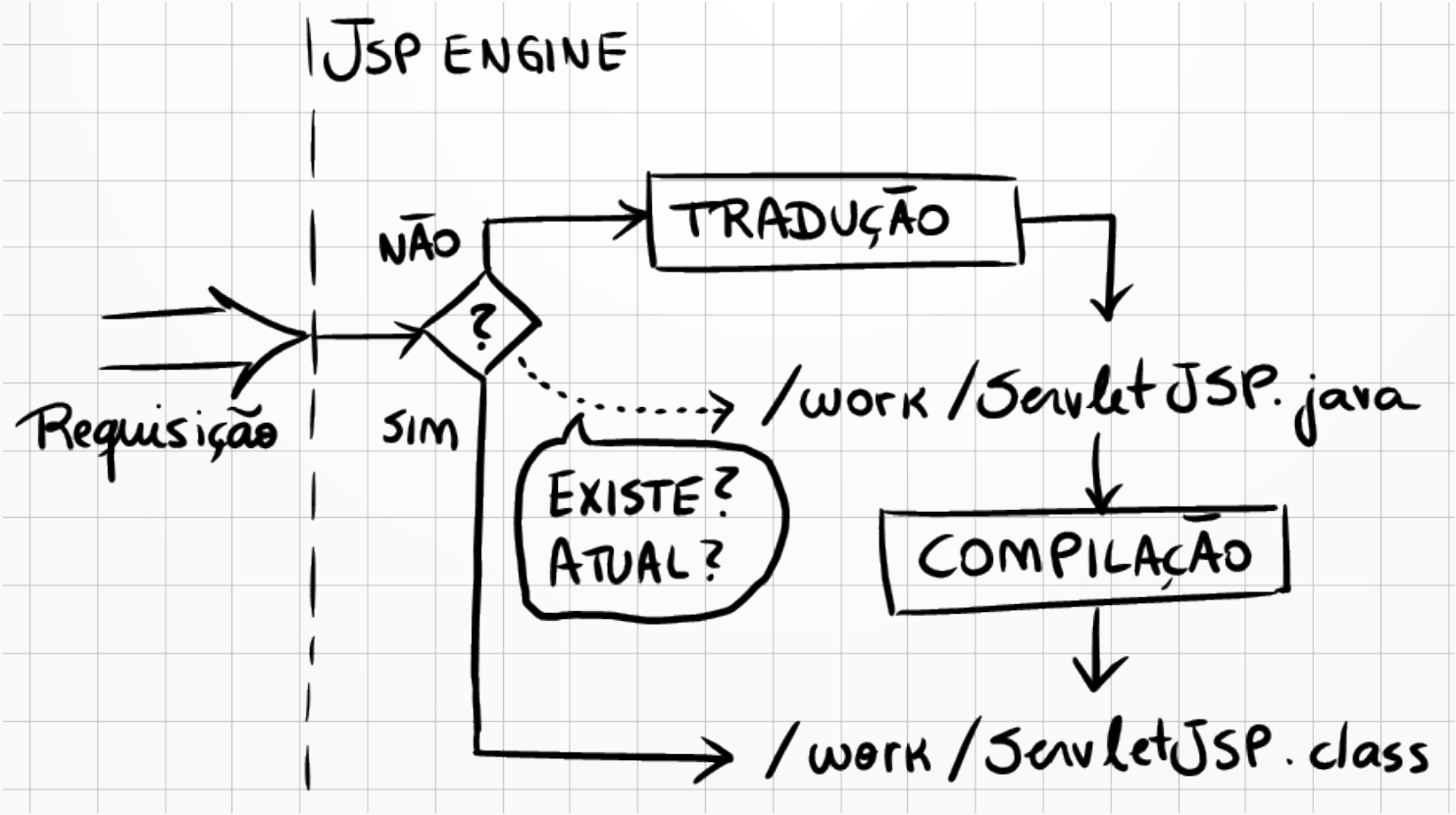
Demonstração 1

Link YouTube: https://youtu.be/-ikAYc4uE8s

Ciclo de vida de uma JSP



Ciclo de vida de uma JSP



Ciclo de vida de uma JSP

Com *Servlets*, é preciso compilar e implantar manualmente

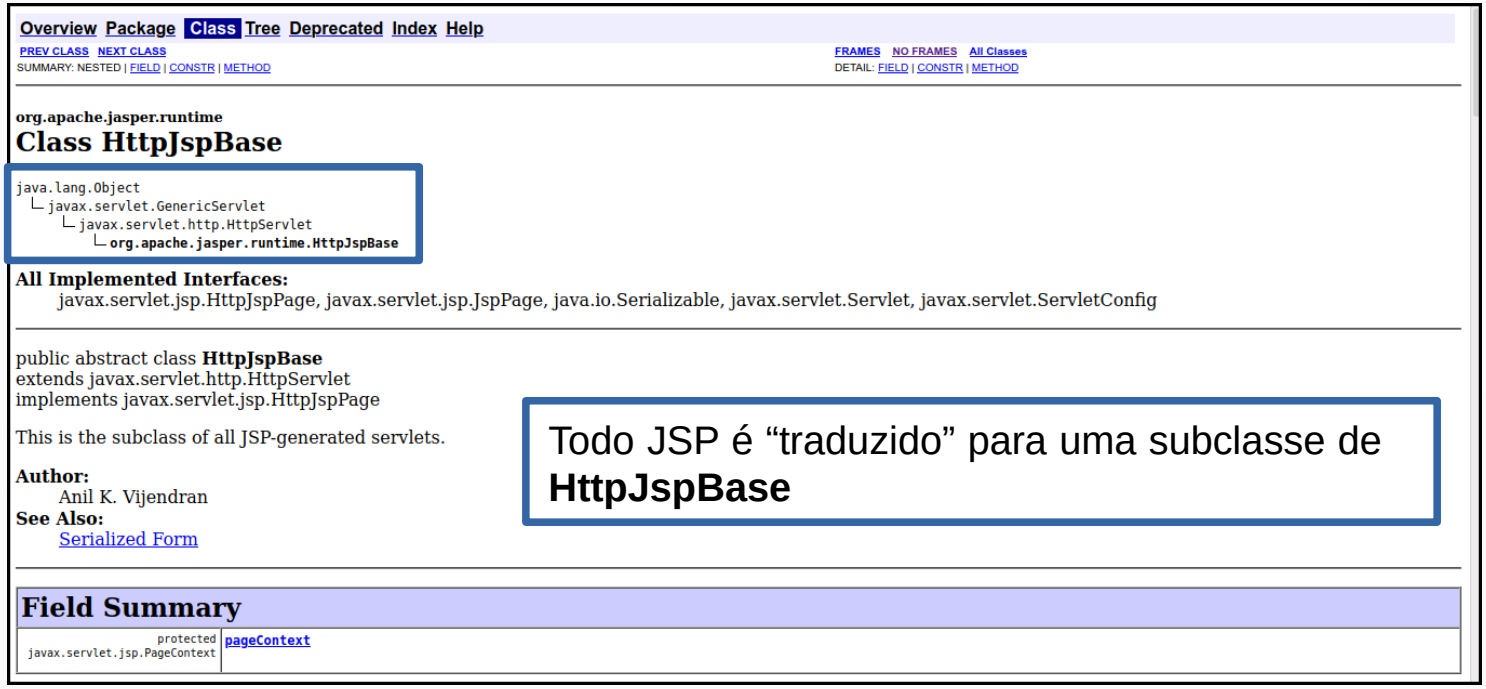
Com JSPs, o web container se encarrega de:

- Traduzir (se necessário)
- Compilar (se necessário)
- Implantar (se necessário)

Jasper: Tradução (JSP → Servlet)

Jasper [edit]

Jasper is Tomcat's JSP Engine. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.



Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS — FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

org.apache.jasper.runtime

Class HttpJspBase

```
java.lang.Object
  └ javax.servlet.GenericServlet
      └ javax.servlet.http.HttpServlet
          └ org.apache.jasper.runtime.HttpJspBase
```

All Implemented Interfaces:
javax.servlet.jsp.HttpJspPage, javax.servlet.jsp.JspPage, java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

public abstract class **HttpJspBase**
extends javax.servlet.http.HttpServlet
implements javax.servlet.jsp.HttpJspPage

This is the subclass of all JSP-generated servlets.

Author:
Anil K. Vijendran
See Also:
Serialized Form

Field Summary

protected javax.servlet.jsp.PageContext	pageContext

Todo JSP é “traduzido” para uma subclasse de **HttpJspBase**

Ciclo de vida de uma JSP

- Com *Servlets*:
  - Erros de compilação aparecem antes da execução

- Com JSPs:
  - Erros de tradução e compilação podem aparecer DURANTE a execução
    - primeira requisição para o JSP, obviamente
  - Ou seja, ao executar um JSP, deve-se estar atento a esses três momentos diferentes
    - E identificar o tipo de erro

Erros JSP

Demonstração 2

Link YouTube: https://youtu.be/ZC8NQoZG_AY

Geração de conteúdo

- Basta digitar o conteúdo
- Ex: para gerar o HTML
  `<h1>Alô mundo</h1>`
- Basta digitar, no JSP
  `<h1>Alô mundo</h1>`
- Ex: para gerar o XML
  `<cliente>João</cliente>`
- Basta digitar, no JSP
  `<cliente>João</cliente>`

Geração de conteúdo

- Cuidado apenas para especificar o tipo correto na diretiva da página

```
<%@page contentType="text/html; charset=UTF-8" %>

<%@page contentType="text/xml; charset=UTF-8"%>
```

Conteúdo dinâmico

Elementos	**Sintaxe**
Comentários	`<%-- ... --%>`
Declarações	`<%! ... %>`

Diretivas	`<%@ include ... %>`
	`<%@ page ... %>`
	`<%@ taglib ... %>`

Scriptlets	`<% ... %>`
Expressões	`<%= ... %>`

Declarações

Arquivo .jsp

```
<%!
int a = 2;
%>
Alô mundo!
```

Servlet gerado (.java)

```
public class ... {
 int a = 2;
 ...
 void service(...) {
  ...
  out.write("Alô mundo!");
 }
}
```

Declarações

Arquivo .jsp

```
<%!
void m1() {
   ...
}
%>
Alô mundo!
```

Servlet gerado (.java)

```
public class ... {
void m1() {
  ...
}
...
 void service(...) {
  ...
  out.write("Alô mundo!");
 }
```

Diretivas

Diretiva	Descrição
<%@page ...	Define informações sobre o processo de tradução Exs: uso de buffer, charset, etc.
<%@include ...	Permite incluir outros arquivos dentro da página, em tempo de tradução
<%@taglib...	Declaração de bibliotecas de tags sendo utilizadas nesta página

Scriptlets

Arquivo .jsp

```
Alô
<%
int a = 2;
%>
mundo!
```

Servlet gerado (.java)

```
public class ... {
...
 void service(...) {
  ...
  out.write("Alô");
  int a = 2;
  out.write("mundo!");
 }
```

Scriptlets

Arquivo .jsp

```
Alô
<%
while(true) {
%>
mundo!
<% } %>
```

Servlet gerado (.java)

```
public class ... {
...
 void service(...) {
  ...
  out.write("Alô");
  while(true) {
  out.write("mundo!");
  }
 }
```

Expressões

Arquivo .jsp

```
<%
String nome="Fulano";
%>
Olá <%= nome %>!
```

Não pode ter ponto e vírgula no final

Servlet gerado (.java)

```
public class ... {
...
 void service(...) {
  ...
  String nome="Fulano";
  out.write("Olá ");
  out.print(nome);
  out.write("!");
  }
 }
}
```

Objetos implícitos

```
pageContext = _jspxFactory.getPageContext(this, request,
response, null, true, 8192, true);

application = pageContext.getServletContext();

config = pageContext.getServletConfig();

session = pageContext.getSession();

out = pageContext.getOut();
```

Objetos implícitos

Demonstração 3

Link YouTube: https://youtu.be/EHIBVmOSyO4

Tags JSP básicas

- A tecnologia JSP define algumas tags úteis
  - Facilitam a programação das páginas
- Tarefas básicas:
  - Encaminhamento/inclusão de outras páginas
  - Declaração de variáveis/objetos (beans)
  - Leitura/escrita de propriedades

<jsp:forward>

- Permite fazer o encaminhamento para outra página
  - Corresponde ao getRequestDispatcher().forward() dos servlets
- Ex:

```
<jsp:forward page="/teste.jsp" />
```

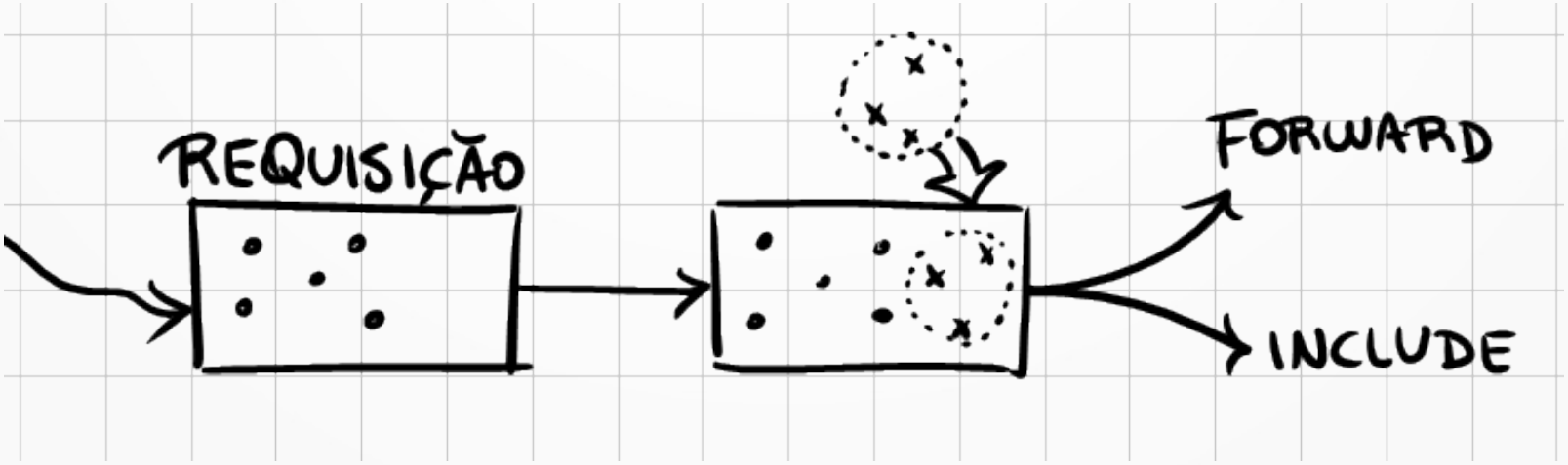
<jsp:include>

- Permite fazer a inclusão dinâmica de outras páginas
  - Corresponde ao getRequestDispatcher().include() dos servlets
- Ex:

```
<jsp:include page="/teste.jsp" />
```

<jsp:param>

- Usado em conjunto com <jsp:forward> e <jsp:include>
  - Permite adicionar parâmetros à requisição



<jsp:include> vs <%@include>

Ambos fazem a inclusão de um outro recurso

Porém:

<jsp:include> é interpretado em tempo de execução

- Permite adicionar parâmetros facilmente
- Mas as variáveis locais não podem ser acessadas

<%@include> é interpretado em tempo de tradução

- Permite que a página incluída acesse variáveis da página inclusora

<jsp:include> vs <%@include>

<%@include>
- O fragmento JSP não precisa ser compilável sozinho
- Não gera um novo *Servlet*
- O fragmento incluído pode acessar variáveis locais
- Conteúdo é “sempre” incluído

<jsp:include>
- Um servlet para cada fragmento
- Cada fragmento deve ser um elemento auto-contido
- Compilável, sem acesso a variáveis externas
- Mas existem outras maneiras de compartilhar informações
- Pode ser feita inclusão condicional

Tags JSP básicas

Demonstração 4

Link YouTube: https://youtu.be/cqdgKASsYXw

Expression Language

- A EL permite simplificar a escrita de páginas JSP, facilitando:
  - Leitura de dados armazenados em JavaBeans, outras estruturas de dados e objetos implícitos
  - Escrita de dados em formulários
  - Realização de operações aritméticas
  - Testes e comparações lógicas
- Sintaxe: ${...} ou #{...}
  - A diferença está no momento da avaliação
  - $ = imediata, # = atrasada
    - Em JSP não há diferença
    - Em JSF (Java Server Faces) há necessidade de #

Expression Language - Objetos implícitos

- Basta iniciar uma expressão com o escopo desejado:

```
${applicationScope.servidorBD ...}

${sessionScope.usuarioLogado ...}

${requestScope.umBeanQualquer ...}

${param.nomeParametro ...}
```

Expression Language

- Expressões aritméticas/relacionais

```
${3 + 2}

${55 < x}

${10 mod 4}

${!empty params.nome}
```

Expression Language

Demonstração 5

Link YouTube: https://youtu.be/zSyxFCzsmms

JSTL

- *JSP Standard Tag Library*
- Conjunto de tags úteis para muitas aplicações baseadas em JSP
- Padronizadas conforme a necessidade
- Dessa forma, reduz-se a necessidade de utilização de bibliotecas de terceiros

JSTL

Principais funcionalidades

- Iteradores
- Condicionais
- Manipulação de documentos XML
- Acesso a banco de dados
- I18n
- Funções comuns

Nome	URI	Funções principais	Prefixo
Core	http://java.sun.com/jsp/jstl/core	Suporte a variáveis Controle de fluxo Gerenciamento de URLs Miscelânea	c
XML	http://java.sun.com/jsp/jstl/xml	Principal Controle de fluxo Transformações	x
I18N	http://java.sun.com/jsp/jstl/fmt	Localização Formatação de mensagens Formatação de datas e números	fmt
Banco de dados	http://java.sun.com/jsp/jstl/sql	SQL	sql
Funções	http://java.sun.com/jsp/jstl/functions	Comprimento de coleções Manipulação de strings	fn

JSTL

Demonstração 6

Link YouTube: https://youtu.be/DD1ObQyNQVo

FIM