JSP - Java Server Pages

Um pouco de tudo

Alisson Chiquitto

[1]Sistemas para Internet

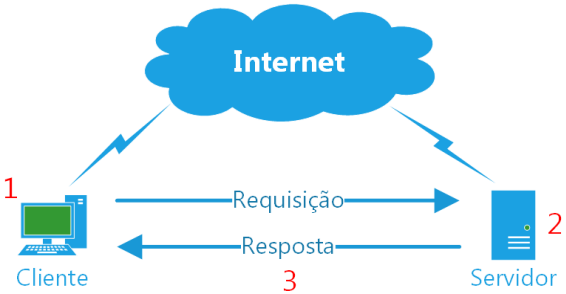
Umuarama, 2016

Outline - Subseção

Protocolo HTTP

"O Hypertext Transfer Protocol (HTTP), em português Protocolo de Transferência de Hipertexto, é um protocolo de comunicação (...). Ele é a base para a comunicação de dados da World Wide Web (www)."

Protocolo HTTP



1. Cliente requisita um arquivo para o Servidor HTTP;
2. Servidor processa a solicitação;
3. Servidor envia a resposta;

Protocolo HTTP

Exemplo de Requisição (request)

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; pt-BR; rv:1.9.0.6) Firefox/3.0.6
Accept: text/html,application/xhtmlxml,application/xml;q=0.9,* *;q=0.8
Accept-Language: pt-br,pt;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Protocolo HTTP

Requisição (tradução para linguagem humana)

```
Bom dia Google.com.
Gostaria de ler sua página.
Estou usando o navegador Firefox na versão 3.0.6.
No momento eu aceito HTML.
Gostaria de receber o conteúdo em português,
  mas também entendo inglês.
Uso o padrão de caracteres (letras) ISO-8859 e UTF-8.
```

Protocolo HTTP

Exemplo de Resposta (response)

```
HTTP/1.1 200 OK
Location: http://www.google.com.br/
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Date: Fri, 11 Jun 2010 19:12:34 GMT
Content-Length: 438

<html> ... </html>
```

Protocolo HTTP

Resposta (tradução para linguagem humana)

```
Aqui é o Google.com.br.
Estou enviando a página que pediu.
Você pode guardar as informações e usa-las em outras visitas.
Estou enviando apenas texto e HTML como você pediu.
Essa carta foi feita as 19:12:34 horas do dia 11/06/2010.
O conteúdo é formado por 438 letras, segue:
<html> ... </html>
```

Métodos do Protocolo HTTP

Mais utilizados:

- **GET** Método que solicita algum recurso ao servidor;
- **POST** Método usado para envio de dados ao servidor.

Outros: HEAD, OPTIONS, DELETE, PUT, TRACE, CONNECT.

Códigos de Status HTTP

- 200 OK;
- 301 Redirecionamento permanente;
- 302 Redirecionamento temporário;
- 304 Não modificado;
- 4xx Erros de cliente;
- 400 Requisição inválida;
- 403 Forbidden - Proibido;
- 404 Não encontrado;
- 5xx Erros do servidor;
- 500 Erro interno do servidor;
- 503 Serviço indisponível;

Outline - Subseção

Porque usar páginas dinâmicas?

Problema

A página inicial de um blog exibe uma lista com 10 posts. Se o blog tiver 100 posts (arquivos HTML), então precisaremos de 10 páginas linkando para todos os posts. Para um blog nestas condições, seriam precisos **110 páginas HTML**.

Porque usar páginas dinâmicas?

Problema

A página inicial de um blog exibe uma lista com 10 posts. Se o blog tiver 100 posts (arquivos HTML), então precisaremos de 10 páginas linkando para todos os posts. Para um blog nestas condições, seriam precisos **110 páginas HTML**.

Solução

2 páginas com programação: 1 página para exibir a listagem dos posts + 1 página para exibir o conteúdo do post.

Outline - Subseção

Servlet

“As Servlets são a primeira forma que veremos de criar páginas dinâmicas com Java. Usaremos a própria linguagem Java para isso, criando uma classe que terá capacidade de gerar conteúdo HTML. O nome "servlet"vem da ideia de um pequeno servidor (servidorzinho, em inglês) cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente.“

Código Java Servlet

```
public class HelloWorld extends HttpServlet {
    protected void service (HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();

        // saida HTML
        out.println("<html>");
        out.println("<body>");
        out.println("Hello World");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Outline - Subseção

JSP - Java Server Pages

"JSP é uma tecnologia para desenvolvimento de aplicações WEB que permite a **inserção de código Java** dentro de páginas HTML. Esse código Java, denominado **scriptlet**, é executado em um servidor web (Tomcat, Jboss, Glassfish, etc), convertido em um servlet, processado e o resultado enviado para o cliente".

Hello World

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
  </head>
  <body>
    <h1><%= "Hello World" %></h1>
  </body>
</html>
```

Hello World com Data

```
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
  </head>
  <body>
      <h1><%= new Date() %></h1>
  </body>
</html>
```

Scriptlet

- Proporcionam uma maneira de inserir diretamente pedaços de instruções Java em diferentes partes do modelo de dados;
- São blocos de código que são executados sempre que uma página JSP é processada.

```
<% String pais = "Brasil"; %>
<h1><% out.print(pais); %></h1>
```

```
<ul>
  <%
  int i;
  for (int i = 0; i < 3; i++) {
  %>
  <li><% out.print(i); %></li>
  <% } %>
</ul>
```

Comentários

Comentários

"Comentários é uma boa forma de documentar a finalidade de um determinado bloco de código. Durante o desenvolvimento do código estamos com a lógica fresca na mente. Mais tarde, somente comentários nos ajudarão a entender o que fizemos duas ou três semanas atrás."

Comentário JSP:

```
<%-- Comentario de uma linha --%>
<%-- Comentario
de várias linhas --%>
```

Comentário Java em scriptlet:

```
<% // Comentario de uma linha %>
<% /* Comentario
de várias linhas */ %>
```

Expressões

"Avalia e converte um valor para String, e então insere a String no local onde a expressão esta localizada."

Aviso

Expressões não aceitam o uso do ponto-e-vírgula (;) ao final da instrução.

Exemplo 1:

```
<% String pais = "Brasil"; %>
<h1><%= pais %></h1>
```

Exemplo 2:

```
<ul>
  <%
  int i;
  for (int i = 0; i < 3; i++) {
  %>
  <li><%= i %></li>
  <% } %>
</ul>
```

Exercício 3.1

Criar uma página que, de acordo com o horário, exiba apenas uma das seguintes mensagens:

- Boa madrugada;
- Bom dia;
- Boa tarde;
- Boa noite.

Resolução: Exercício 3.1 I

```
<%@page import="java.util.Calendar"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<p>Olá usuário, <%
  Calendar cal = Calendar.getInstance();
  int hora = cal.get(Calendar.HOUR_OF_DAY);

  if (hora < 6) {
    out.print("boa madrugada");
  } else if (hora < 12) {
    out.print("bom dia");
  } else if (hora < 18) {
    out.print("boa tarde");
  } else {
    out.print("boa noite");
  }
  %>.</p>
```

Exercício 3.2

Criar um programa JSP que irá gerar a seguinte saída na tela do navegador:

```
*
**
***
****
*****
```

Resolução: Exercício 3.2 I

```
<%@page import="java.util.Calendar"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<pre><%
  int i1, i2;
  for (i1 = 1; i1 <= 5; i1++) {
    for (i2 = 1; i2 <= i1; i2++) {
      out.print("*");
    }
    out.println("");
  }
  %></pre>
```

Exercício 3.3

Criar uma página que irá gerar um *array* de números inteiros, e em seguida exibi-los em forma de lista não ordenada (`<ul>`).

Resolução: Exercício 3.3 I

```
<ul><%
  int n[] = new int[10];
  int i;

  for (i = 0; i < n.length; i++) {
    n[i] = i % 2;
  }

  for (i = 0; i < n.length; i++) {
    %><li><%= n[i] %></li><%
  }
%></ul>
```

Declarações

- Definem variáveis para uso subsequente em expressões ou *scriptlets*;
- A declaração da variável é feita apenas na primeira execução do código, nas seguintes o valor é mantido.

Sintaxe:

```
<%! int contador = 1; %>
```

Exemplo:

```
<%! int contador = 1; %>
<%= contador %>
<% contador++; %>
```

Diretivas

As diretivas são tags que afetam a estrutura dos servlets na construção página JSP.

page	Importa classes e controlar outros parâmetros da página;
include	Coloca o conteúdo de um arquivo em outro;
taglib	Permite a carga de uma biblioteca de tags.

Diretiva Page

A diretiva **page** permite controlar a estrutura do servlet, tornado-o capaz de importar classes, configurar o *contentType*, etc. Esta diretiva pode ser colocada em qualquer ponto da página.

Exemplo:

```
<%@ page atributo="valor" %>
```

Diretiva Page ContentType e PageEncoding

contentType Define o *mime-type* da resposta;

pageEncoding Define o esquema de codificação de caracteres.

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

Obs.: O uso da diretiva `contentType` gera o mesmo efeito da instrução `response.setContentType(String type)`

Diretiva Page Import

Especifica uma lista de pacotes ou classes que serão usadas na página.

```
<%@ page import="java.util.Date" %>
<%@ page import="java.time.LocalDate,java.text.DateFormat" %>
```

Diretiva Page Session

Indica se a página usa ou não sessões.

```
<%@ page session="true" %>
```

Diretiva Page isErrorPage e errorPage

errorPage Especifica uma página que irá manipular exceções lançadas por esta página;

```
<%@ page errorPage="ExceptionHandling.jsp"%>
```

isErrorPage Define se a página pode ser utilizada como página de erros.

```
<%@ page isErrorPage="true"%>
```

Diretiva Include

Coloca o conteúdo de um arquivo em outro. A diretiva se substitui pelo conteúdo indicado.

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

Taglibs

Uma Taglib é uma biblioteca de tags customizadas que são utilizadas na composição de páginas JSP. Podemos dizer que uma Taglib é uma biblioteca de Classes que são utilizadas *tags* para auxiliar na geração de arquivos JSP.

```
<%@ taglib uri="/diretorio/biblioteca" prefix="mt"%>
Olá! Hoje é <mt:horaAtual/>
```

Objetos implícitos

Os objetos implícitos do JSP são objetos Java que estão disponíveis em cada página JSP. Eles podem ser acessados diretamente, sem a necessidade da declaração explícita de variáveis. São eles: `request`, `response`, `out`, `session`, `application`, `config`, `pageContext`, `page` e `exception`.

Objeto implícito: request I

Objeto request

Quando um usuário solicita uma página web, ele envia várias informações para o servidor. Todas estas informações podem ser acessadas através do objeto `request`. Instância de `javax.servlet.http.HttpServletRequest`.

Principais métodos:

Enumeration getParameterNames()

Retorna um `Enumeration` de `String` contendo os nomes dos parâmetros contidos na requisição;

String getHeader(String name)

Retorna o valor do cabeçalho `name`, enviado na requisição;

Objeto implícito: request II

String getMethod()

Retorna o nome do método HTTP em que a requisição foi feita (GET, POST, etc);

String getParameter(String name)

Retorna o valor do parâmetro `name`, ou `null` se o parâmetro não existir;

String getRequestURI()

Retorna o arquivo solicitado na URI;

Objeto implícito: request I

Exemplo de uso

Enviar dados por GET/POST:

```
<form action="request-receive.jsp">
Nome: <input type="text" name="nome"><br>
Email: <input type="text" name="email"><br>
<input type="submit" value="Salvar">
</form>
```

Manipulação de dados enviados pelo usuário:

```
<%
String nome=request.getParameter("nome");
String email=request.getParameter("email");
out.print("Nome:"+nome+" Email: "+email);
%>
```

Exercício 4.1

Conforme a Agência Nacional de Petróleo só vale a pena você abastecer com Etanol se o valor for até 70% o valor da Gasolina, caso contrário, abasteça com Gasolina. Para realizar o cálculo é bem simples, **divida o valor do litro do Etanol pelo da gasolina. Se o resultado for menor que** 0.7**, abasteça com Etanol. Se for maior, escolha a Gasolina**.

Exemplo: se o Etanol custa $R\$2.19$ e a Gasolina, $R\$3.35$ o resultado da divisão do primeiro pelo segundo é 0.65 (< 0.7). Logo, é mais vantajoso abastecer com álcool.

Criar um formulário para que o usuário possa fornecer os valores do Etanol e da Gasolina. Assim que o usuário preencher o formulário e clicar em Calcular, os dados fornecidos devem ser enviados para um *servlet* que irá processar as informações e finalmente exibir para o usuário qual a melhor escolha.

Resolução: Exercício 4.1 I

```
<% //exer-calc-etanolgasolina1.jsp %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
(...)
<h1>Calculadora Etanol x Gasolina</h1>
<form method="post" action="exer-calc-etanolgasolina2.jsp">
  <p>Informe o preço do Etanol e da Gasolina, e em seguida clique em Calcular.</p>
  <p>
    <label>Etanol: <input type="text" name="etanol"></label>
  </p>
  <p>
    <label>Gasolina: <input type="text" name="gasolina"></label>
  </p>
  <p><input type="submit" name="submit" value="Calcular"></p>
  <p><small>Obs.: Use ponto para casas decimais.</small></p>
</form>
(...)
```

Resolução: Exercício 4.1 II

```
<% //exer-calc-etanolgasolina2.jsp %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
(...)
<h1>Resultado do cálculo</h1>
<%
String etanolString = request.getParameter("etanol");
String gasolinaString = request.getParameter("gasolina");
double etanol = Double.parseDouble(etanolString);
double gasolina = Double.parseDouble(gasolinaString);
double relacao = etanol / gasolina;
%>
<h2>Valores informados</h2>
<p>Etanol: R$ <%= etanolString %></p>
<p>Gasolina: R$ <%= gasolinaString %></p>
<p>Relação entre os preços: <%= relacao %></p>

<h2>Resultado:</h2>
<% if (relacao < 0.7) { %><p>A melhor escolha é Etanol</p>
<% } else { %><p>A melhor escolha é Gasolina</p>
<% } %>
(...)
```

Objeto implícito: response I

Objeto response

Quando o servidor responde à requisição HTTP, ele envia várias informações para o usuário. Todas estas informações podem ser lidas/alteradas através do objeto `response`. Instância de `javax.servlet.http.HttpServletResponse`.

Principais métodos:

String encodeURL(String url)
: Codifica `url` de acordo com as regras de URL do protocolo HTTP;

boolean containsHeader(String name)
: Indica se existe o cabeçalho `name` em `response`;

Objeto implícito: response II

void addHeader(String name, String value)

Adiciona o cabeçalho `name` em `response`;

void sendRedirect(String location)

Envia para o cliente um redirecionamento temporário para `location`;

void setContentType(String type)

Define o tipo do conteúdo que é enviado para o usuário;

void setHeader(String name, String value)

Define um cabeçalho de resposta (`name`) para `value`;

void setStatus(int sc)

Define um código de *status*;

Objeto implícito: response I

Exemplo de uso

```
<%
response.setStatus(500);
response.setContentType("text/html");
%>
<h1>Não podemos processar sua requisição</h1>
```

Exercício 4.2

Criar um programa JSP que irá conter uma lista com 5 nomes de clientes. O resultado do processamento deste programa deverá ser o envio de um XML com os dados da lista. Para complementar, o programa deverá informar ao usuário que o conteúdo gerado é do tipo XML (*text/xml*), ao invés de HTML.

Resolução: Exercício 4.2 I

```
<%
  response.setContentType("text/xml");

  String[] clientes = new String[5];
  clientes[0] = "Jose Filipino";
  clientes[1] = "João Pé de Feijão";
  clientes[2] = "Maria Feia";
  clientes[3] = "Lamela Banguela";
  clientes[4] = "Bill Clill";
%>
<clientes>
  <% for (int i = 0; i < 5; i++) { %>
  <cliente><%= clientes[i] %></cliente>
  <% } %>
</clientes>
```

Objeto implícito: out I

Objeto out

Permite escrever no stream de saída. Instância de `javax.servlet.jsp.JspWriter`.

Principais métodos:

`void clear()`
: Limpa o conteúdo do *buffer*;

`void flush()`
: Envia o conteúdo do buffer para o cliente;

Objeto implícito: out II

void print(boolean b)

void print(char c)

void print(int i)

void print(String s)

Escreve o valor do parâmetro de entrada.

Objeto implícito: session I

Objeto session

Permite manipular a sessão do cliente no servidor. Instância de `javax.servlet.http.HttpSession`.

Principais métodos:

Object getAttribute(String name)
: Retorno o objeto ligado ao atributo `name` da sessão, ou `null` se `name` não existir na sessão;

Enumeration getAttributeNames()
: Retorna um `Enumeration` contendo o nome dos atributos contidos na sessão;

Objeto implícito: session II

long getCreationTime()

Retorna o *timestamp* da data de criação da sessão;

String getId()

Retorno o ID da sessão;

boolean isNew()

Informa se a sessão foi criada na requisição atual;

void removeAttribute(String name)

Remove o atributo `name` da sessão;

void setAttribute(String name, Object value)

Define o valor do atributo `name`;

Objeto implícito: session I

Exemplo de uso

```
<%@ page session="true" %><%@ page import="java.io.*,java.util.*" %>
<%
   Date createTime = new Date(session.getCreationTime());
   // Last access time of this web page.
   Date lastAccessTime = new Date(session.getLastAccessedTime());
   String title = "Welcome Back to my website";
   Integer visitCount = new Integer(0);
   String visitCountKey = new String("visitCount");
   String userIDKey = new String("userID");
   String userID = new String("ABCD");
   // Check if this is new comer on your web page.
   if (session.isNew()){
      title = "Welcome to my website";
      session.setAttribute(userIDKey, userID);
      session.setAttribute(visitCountKey,  visitCount);
   }
   visitCount = (Integer) session.getAttribute(visitCountKey);
   visitCount = visitCount + 1;
   userID = (String)session.getAttribute(userIDKey);
   session.setAttribute(visitCountKey,  visitCount);
%>
```

Objeto implícito: application I

Objeto application

Este objeto é a representação da página JSP através do seu ciclo de vida. Este objeto é criado quando a página JSP é inicializada e removida quando a página é destruída. Instância de `javax.servlet.ServletContext`.

Objeto implícito: config I

Objeto config

Usado para pegar informações de configuração de uma página. Instância de `javax.servlet.ServletConfig`.

Objeto implícito: pageContext I

Objeto pageContext

Utilizado para acessar atributos da página, requisição, resposta e sessão.
Instância de `javax.servlet.jsp.PageContext`.

Objeto implícito: page I

Objeto page

Referência a uma instância da página. Acessado através da variável `this`.

Objeto implícito: exception I

Objeto exception

Representa uma `exception`. Utilizado na manipulação de excecões para a exibição de mensagens de erro. Instância de `javax.servlet.jsp.JspException`.

Objeto implícito: exception I

Exemplo de uso

Página que dispara exception:

```
<%@ page errorPage="divisao-zero-exception.jsp" %>
<%
int res= 10/0;
out.print("Output is: "+ res);
%>
```

Página que manipula exception:

```
<%@ page isErrorPage="true" %>
Exception: <%= exception %>
```

Referências I


HORSTMANN, Cay. Big Java. Bookman, 2009.


MENDES, Douglas Rocha. Programação Java com Ênfase em Orientação a Objetos. Novatec, 2009.


Jsp Implicit Objects
Disponível em
http://beginnersbook.com/2013/11/jsp-implicit-objects/


JSP - Implicit Objects
Disponível em
http://www.tutorialspoint.com/jsp/jsp_implicit_objects.htm


JSP - Client Request
Disponível em
http://www.tutorialspoint.com/jsp/jsp_client_request.htm

Referências II


JSP - Server Response
Disponível em
http://www.tutorialspoint.com/jsp/jsp_server_response.htm


JSP - Session Tracking
Disponível em
http://www.tutorialspoint.com/jsp/jsp_session_tracking.htm


Application Implicit Object in JSP with examples
Disponível em http://beginnersbook.com/2013/11/
jsp-implicit-object-application-with-examples/


Config Implicit Object in JSP with examples
Disponível em http://beginnersbook.com/2013/11/
jsp-implicit-object-config-with-examples/

Referências III


Beginners Book - JSP - Exception Handling
Disponível em `http://www.tutorialspoint.com/jsp/jsp_exception_handling.htm`