Programação IV
Aula 04 – JSP
Parte 1

Prof. Alex Marin

alex.marin@iffarroupilha.edu.br

Introdução

- Inicialmente as páginas na Web eram apenas páginas estáticas, isto é, seu conteúdo não variava a cada solicitação conforme algum parâmetro.
- Com a sofisticação dos serviços disponibilizados via Web, surgiu a necessidade de disponibilizar informações com natureza dinâmica (lista de preços atualizados, compras on-line, etc).
- Isso exigiu que o servidor web fizesse algum processamento adicional da solicitação a fim de gerar uma resposta personalizada.

Introdução

- Algumas tecnologias para geração de conteúdo dinâmico para a Internet são:
  - CGI
  - APIs do Servidor
    - Servlets Java
  - Scripts de Servidor
    - Microsoft Active Pages (ASP)
    - PHP
    - Java Server Pages (JSP)

Introdução

- Problemas dos Servlets
  - Para gerar páginas dinâmicas é preciso embutir o HTML dentro de instruções de um programa.

```
out.print("<h1>Servlet</h1>");
for(int i=1; i<=5; i++) {
   out.print("<p>Paragrafo " + i + "</p>");
}
```

  - Compromete a manutenção;
  - O design acaba ficando a cargo do programador e não do web designer.

Introdução

- Solução
  - Colocar a linguagem de programação dentro do HTML

```
<h1>Servlet</h1>
<% for(int i=1; i<=5; i++) { %>
    <p>Paragrafo <% out.print(i); %> </p>
<% } %>
```

  - O web designer pode projetar as páginas em ferramentas como o DreamWeaver;
  - Quando houver muita programação, o código pode ser escondido em servlets, JavaBeans, etc.

Java Server Pages – JSP

- Segundo a Sun: "*o JSP é uma tecnologia JavaEE para a construção de aplicações para geração de conteúdo dinâmico para a web, a qual provê facilidades de autoria e desenvolvimento*".

- Simplificando, é a composição de elementos Java com código HTML através de marcações semelhantes ao XML.

Java Server Pages – JSP

- Seu objetivo principal é auxiliar na separação da lógica de negócios (código Java) da apresentação da informação (código HTML).

- Possibilita flexibilizar o design das páginas, o qual pode ser melhorado através do emprego de outras técnicas apropriadas (CSS).

Java Server Pages – JSP

- Resumindo:
  - Servlets e JSP são as soluções Java para estender o servidor e gerar conteúdo dinâmico;
  - Páginas JSP são traduzidas para servlets e então compiladas (o processo de execução é o mesmo);
  - Suportam os métodos de requisição padrão HTTP (GET, POST, etc);
  - Interagem com *cookies;*
  - Suportam controle de sessão de forma transparente

Um primeiro exemplo

- Exemplo de um Servlet

  `Confira: ServletSimples.java`

- Exemplo de JSP

  `Confira: ServletSimplesEquivalente.jsp`

Java Server Pages – JSP

- Estrutura de uma página JSP:

```
<%@    %>          -> DIRETIVA
<html>
    <body>
        <%--   --%>     -> COMENTÁRIO
        <%!      %>     -> DECLARAÇÃO
        <%       %>     -> SCRIPLET
        <%=      %>     -> EXPRESSÃO
    </body>
</html>
```

Java Server Pages – JSP

- Uma página JSP pode conter três tipos de comentários:
  - **Comentários de cliente** – são os comentários HTML que serão enviados ao cliente, ou seja, serão exibidos caso o código recebido seja visualizado.

```
<!-- Comentário HTML: enviado para o cliente --%>
```

Java Server Pages – JSP

- **Comentários de programação JSP** – são os comentários JSP que não serão enviados ao cliente, ou seja, nunca serão exibidos.

```
<%-- Comentário JSP: não enviado para o cliente --%>
```

- **Comentários de scripting** – são os comentários permitidos pela linguagem de scripting, no caso é o Java, e que não são enviados aos clientes.

```
\\ Comentário de uma linha
\* Comentário de
   múltiplas linhas *\
```

Java Server Pages – JSP

- O JSP é formado por alguns elementos básicos:
  - **Scripting** – elementos que possibilitam o uso de código Java capaz de interagir com outros elementos do JSP.
  - **Diretivas** – são mensagens enviadas ao JSP *engine* que permitem especificar configurações de página, inclusão de recursos e bibliotecas;
  - **Ações** – são etiquetas predefinidas que podem ser incorporadas em páginas JSP, cujas ações, dependem das informações enviadas ao servidor.

Elementos de Criação de Scripts

- Os elementos de *scripting* são usados para manipulação de objetos, realização de cálculos e tomadas de decisão que afetam o conteúdo gerado.

- Podem ser classificados em:
  - Declarações
  - Expressões
  - Scriptlets

Declarações

- Servem para definir variáveis e métodos específicos para uma página JSP.
- Sua sintaxe é:

```
<%! declarações %>
```

- Os métodos e variáveis declarados podem ser referenciados por outros elementos de criação de *script* dentro da mesma página JSP independente da ordem na qual a declaração ocorre em relação a estes elementos.

Declarações

- Declaração de variáveis :

```
<%! int y, x=0; String text = "Texto"; %>
```

- Declaração de métodos :

```
<%! int soma (int x, int y) {
        return (x + y); } %>
```

  - As variáveis aqui declaradas são criadas na inicialização da página e seus valores são compartilhados por todos que a acessarem. Se uma pessoa mudar o valor de **`x`** para **`1`**, todos que acessarem o servlet depois disso obterão **`x=1`**

Expressões

- São elementos de scripting que são avaliadas pelo web container e cujo resultado é convertido em um objeto **`String`**.

- Este resultado é enviado para a saída, por meio do objeto implícito **`out`** compondo a resposta a ser enviada para o cliente.

Expressões

- O resultado da avaliação de uma expressão é inserido no *output* da página no lugar da *tag* de expressão original.

- Sua sintaxe é:

```
<%= expressão %>
```
(não requer ; no final)

Expressões

- Não há restrição quanto ao tipo de retorno das expressões.
- As expressões em JSP possuem o objetivo explícito de geração de *output*.
- Exemplos:

  <%= "Mensagem" %> --> imprime: Mensagem

  <%= msg %> --> imprime o conteúdo da var. **msg**

  <%= fatorial(0) %> --> imprime o resultado da função

Confira: **Expressoes.jsp**

Scriptlets

- Os **scriptlets** são utilizados para criação de códigos arbitrários, que realizam qualquer processamento.
  - Em outras palavras, servem para criação de *scripts* de objetivos gerais.

- Dentro de uma *tag* de **scriptlet** é possível declarar variáveis, instanciar objetos, imprimir no *output* da página.
  - Enfim, realizar qualquer codificação válida com a linguagem de *script* que está sendo utilizada, no nosso caso Java.

Scriptlets

- São blocos de código executados cada vez que a página JSP é processada.
- Sua sintaxe é:

```
<% scriptlet %>
```

- Uma variável definida em um **scriptlet** estará disponível para uso em expressões e **scriptlets subseqüentes** na mesma página.

Confira: **Celsius2Far.jsp**

Scriptlets

- Embora já possamos escrever algumas aplicações em JSP com o que já aprendemos até agora, elas serão ainda fracas.

- Um dos grandes potenciais de qualquer linguagem de programação é a utilização de controles de fluxo (condicionais e loops) para executar diferentes partes de um programa baseado em testes.

Scriptlets

- Comandos condicionais: IF

Confira: **Exemplo_IF.jsp**

- Comandos condicionais: SWITCH

Confira: **Exemplo_SWITCH.jsp**

Scriptlets

- Comandos de laços de repetição: FOR

  Confira: **Exemplo_FOR.jsp**

- Comandos condicionais: WHILE

  Confira: **Exemplo_WHILE.jsp**

- Comandos de laços de repetição: DO...WHILE

  Confira: **Exemplo_DOWHILE.jsp**

Diretivas JSP

- As diretivas são usadas para fornecer informações especiais ao container JSP sobre a página quando esta é compilada (são processadas na fase de tradução) para servlet.

- Elas não produzem qualquer *output* que seja visível, ao invés disso, elas geram efeitos colaterais que mudam o modo como o container JSP processa a página.

Diretivas JSP

- Principais tipos:
  - **Page:** permite a importação de classes, customização de super classes servlet, entre outras.
  - **Include:** permite que um conteúdo de um arquivo seja inserido no servlet. Isto permite a construção de pedaços reutilizáveis.
  - **Taglib:** permite que o ambiente JSP importe uma determinada biblioteca de tags.

Diretiva Page

- Define atributos que afetam toda a página JSP.

- Sintaxe :

```
<%@ page atributo1="valor1" atributo2="valor2"
         atributo3=...   %>
```

ou

```
<%@ page atributo1 = "valor1" %>
<%@ page atributo2 = "valor2" %> ...
```

Diretiva Page – Atributos

- **`info`**: Permite ao autor adicionar uma cadeia de documentação à página que sumariza sua funcionalidade. O valor padrão para o atributo **`info`** é a cadeia vazia. Pode ser recuperada pelo método **`getServletInfo`**.

- Ex :

```
<%@ page info = "Aula de JSP,
  desenvolvida por Alex Marin"  %>
```

Diretiva Page – Atributos

- **`language:`** Define a linguagem de criação de scripts da página. O padrão para este atributo é java.

- Ex :

```
<%@ page language = "java" %>
```

Diretiva Page – Atributos

- **`contentType`**: Define o tipo MIME, i.e, o tipo de informação contida em uma resposta de HTTP. Os tipos mais comuns para JSP são : “text/html” (padrão), “text/xml” e “text/plain”, indicando respostas em HTML, XML e texto puro.

- Ex :

```
<%@ page contentType = "text/html" %>
```

Diretiva Page – Atributos

- **`import:`** Permite especificar quais pacotes serão importados, estendendo assim o conjunto de classes que podem ser referenciadas em uma página. É o único atributo que pode se repetir.
- Ex :

Importa a classe List

```
<%@ page import = "java.util.List" %>

<%@ page import = "java.util.* , java.text.*" %>
```

Importa todas as classes do pacote java.util e java.text

Diretiva Page – Atributos

- **`session:`** Indica se uma página participa ou não do gerenciamento de sessão. O valor padrão é **`true`**. Se a página não irá utilizar recursos de sessão, deve-se passá-lo para **`false`** pois isso resulta em ganho de desempenho.

- Ex :

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

Diretiva Page – Atributos

- **`buffer:`** Controla o uso da saída bufferizada. O padrão é um buffer de 8kb.

- Ex :

```
<%@ page buffer="none" %>
<%@ page buffer="12kb" %>
```

Diretiva Page – Atributos

- **`autoFlush`:** Também controla a saída buferizada. Se definido como **`true`** (padrão), quando o buffer fica cheio seu conteúdo é enviado para o servidor HTTP para transmissão ao navegador solicitante. Se definido como **`false`**, quando o buffer fica cheio é disparada uma exceção, interrompendo o processamento e enviando um erro como resposta.

- Ex :

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

Diretiva Page – Atributos

- **`errorPage`**: Define uma página alternativa a ser exibida se um erro (não tratado) ocorrer enquanto o container estiver processando a página. A página de erro deve ser local e é especificada da seguinte forma:

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

Diretiva Page – Atributos

- **`isErrorPage`**: Quando **`true`**, define que a página serve como página de erro para uma ou mais páginas JSP. O valor padrão para este atributo é **`false`**.

- Ex:

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

Confira: **`Exemplo_DiretivaPage.jsp`**

Diretiva Include

- Permite incluir o conteúdo de um arquivo em outro. A diretiva tem o efeito de substituir a si mesma pelo conteúdo do arquivo indicado.

- O conteúdo pode ser texto estático (HTML) ou comandos JSP, que serão processados como se fossem parte da página original.

- Ex :

```
<%@ include file="localURL" %>
```

Diretiva Include

cabecalho.html

```
<h1 style="color:red; background-color:#EEEEEE">
    Este é o cabeçalho
</h1>
```

rodape.html

```
<strong>Esse texto faz parte do rodapé! </strong>
```

Confira: **Exemplo_DiretivaInclude.jsp**

Diretiva Taglib

- Permite indicar quais bibliotecas de tags poderão ser utilizadas pela página JSP.

- Para utilizá-la, é necessário indicar qual prefixo será utilizado pelas tags e qual a URI (*Uniform Resource Identifier*).

- Ex :

```
<%@ taglib prefix="prefixo"
           uri="URI_da_taglib" %>
```

Tratando Formulários

- Os formulários são ferramentas úteis e muito usados em diversas aplicações: cadastro de registros em um banco de dados, validação de um login/senha, envio de email, envio de dados de um pesquisa, etc.

- Hoje é quase impossível desenvolver uma aplicação para Web que não exija o uso de formulários.

Tratando Formulários

Confira: **Exemplo_Formulario.html**

Confira: **trata_dados.jsp**

Objetos Implícitos

- Como visto anteriormente, podemos criar, dentro de scriptlets na página JSP, instâncias de uma classe Java e manipulá-las a fim de produzir conteúdo dinâmico.
- Por exemplo, podemos criar um objeto de uma classe que acessa uma base de dados e então usar métodos desse objeto para exibir na página uma consulta.
- Ou seja, através da manipulação desse objeto, quer seja acessando seus métodos ou suas variáveis, podemos gerar conteúdo na página JSP.

Objetos Implícitos

- Além de objetos como esses, que estão completamente sobre o controle do programador, o container JSP se encarrega de instanciar automaticamente, durante a execução de uma página JSP, outros objetos.

- Tais objetos podem ser usados dentro da página JSP e são conhecidos como "Objetos Implícitos", pois sua disponibilidade em uma página JSP é automática.

Objetos Implícitos

Objeto	Classe ou Interface	Descrição
page	javax.servlet.jsp.HttpJspPage	Instância de um servlet da página
config	javax.servlet.ServletConfig	Dados de configuração de servlet
request	javax.servlet.http.HttpServletRequest	Dados de solicitação, incluindo parâmetros
response	javax.servlet.http.HttpServletResponse	Dados de resposta
out	javax.servlet.jsp.JspWriter	Fluxo de saída para conteúdo da página
session	javax.servlet.http.HttpSession	Dados de sessão específicos de usuário
application	javax.servlet.ServletContext	Dados compartilhados por todas as páginas da aplicação
pageContext	javax.servlet.jsp.PageContext	Dados do contexto para execução da página
exception	javax.lang.Throwable	Erros não capturados ou exceção