Módulo IVb - JSP

Prof. Ismael H F Santos

Ementa

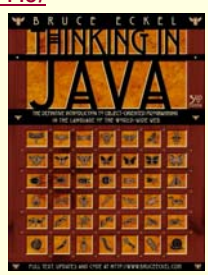
- Módulo IVb – Java Server Pages - JSP
  - Introdução
  - Elementos de Script e Diretivas
  - JSP e Servlets
  - Java Beans e JSP

Bibliografia

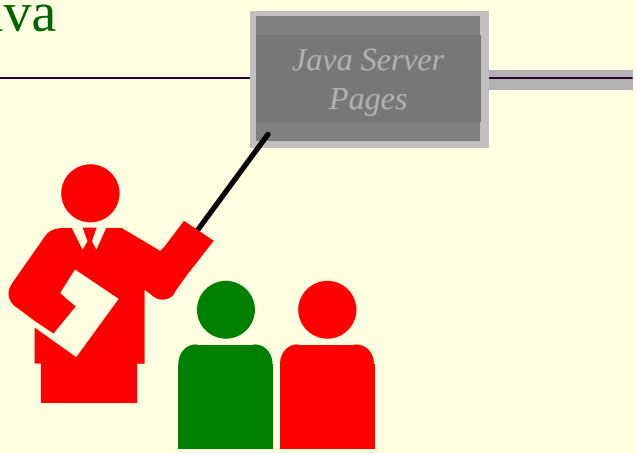
- *Linguagem de Programação JAVA*
  - *Ismael H. F. Santos, Apostila UniverCidade, 2002*
- *The Java Tutorial: A practical guide for programmers*
  - *Tutorial on-line: http://java.sun.com/docs/books/tutorial*
- *Java in a Nutshell*
  - *David Flanagan, O´Reilly & Associates*
- *Just Java 2*
  - *Mark C. Chan, Steven W. Griffith e Anthony F. Iasi, Makron Books.*
- *Java 1.2*
  - *Laura Lemay & Rogers Cadenhead, Editora Campos*

Livros

- **Core Java 2**, Cay S. Horstmann, Gary Cornell
  - Volume 1 (Fundamentos)
  - Volume 2 (Características Avançadas)
- **Java: Como Programar**, Deitel & Deitel
- **Thinking in Patterns with JAVA**, Bruce Eckel
  - **Gratuito**. http://www.mindview.net/Books/TIJ/



POO-Java



Java Server Pages

- Java Server Pages (JSP) é uma **especificação** da SUN que possibilita colocar código fonte em Java dentro de HTML
  - E não o contrário, como acontece com os servlets.
  - Isso permite o controle da aparência da página através de softwares de design (HomeSite, DreamWeaver, FrontPage, etc.)

JSP (Java Server Pages)

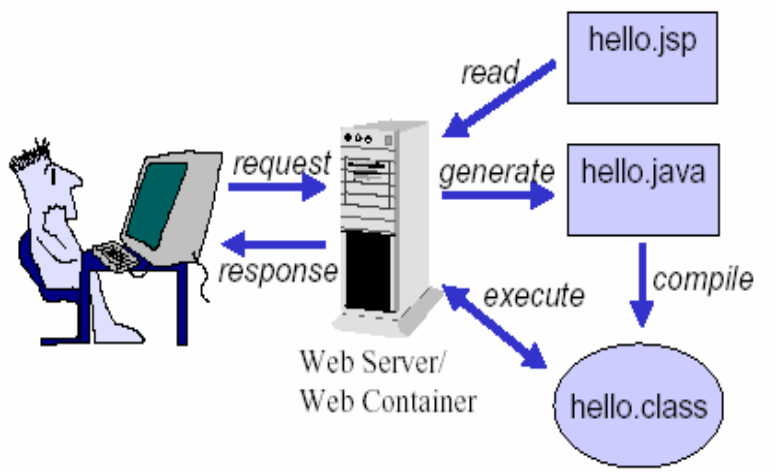
- Tecnologia Java para geração de conteúdo dinâmico
- O texto HTML é escrito junto com as tags JSP e código Java
- Não é uma idéia nova sendo usado em tecnologias concorrentes: ASP, PHP, Server-Side JavaScript e Cold Fusion
- Tecnologias concorrentes e similares à JSP:
  - Cold Fusion (Macromedia)
  - PHP
  - ASP, Active Server Pages (Microsoft)

Java Server Pages

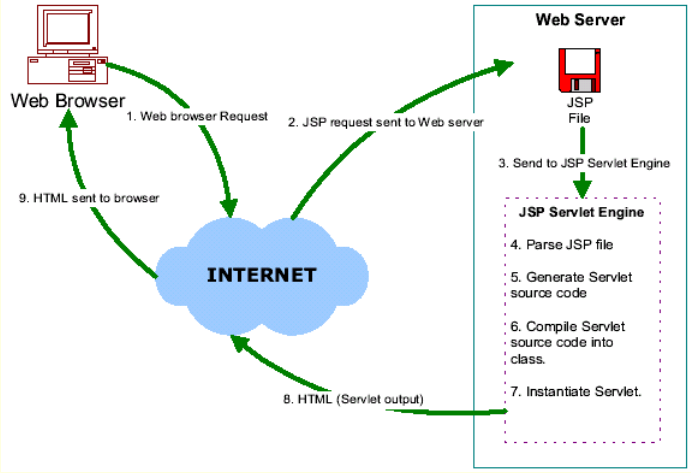
- Uma página JSP consiste de um documento no formato de texto que possui código HTML e código Java.
- Os trechos de código em Java em uma página JSP são denominados scriptlets.
- Páginas JSP possuem uma extensão

```
<H1>Exemplo</H1>
<% for(int i = 0; i < 10/ i++) { %>
     <P>Um número:" <%= i %> </P>
<% }%>
<TABLE><TR><TD>...</TR></TABLE>
```

Processamento de JSP



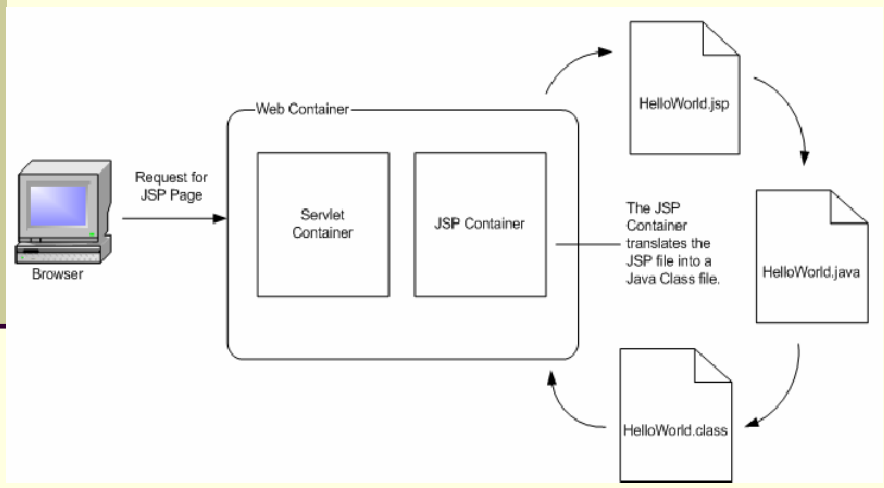
JSP (Java Server Pages)



JSP → Servlets

- Arquivos JSP são automaticamente traduzidos em servlets pelo contêiner.
  - Ou seja, após um pré-processamento dos **scriptlets**, páginas JSP acabam por serem traduzidas em servlets.
- Quando uma página JSP é acessada pela primeira vez, o contêiner gera a servlet correspondente e a executa.
- A servlet permanece em memória após a primeira chamada.
- Essa servlet somente é recompilada quando o arquivo .jsp é modificado.

JSP → Servlets



Programando scriptlets

- Para importar um pacote em uma página JSP:
  - Use a diretiva <%@ page import
  - Exemplo: <%@ page import="java.sql.*" %>
- Para inserir código Java em uma página JSP, delimite esse código por <% e %>
- O tag <%=
  - Faz com que a expressão Java entre os tags <%= e %> seja avaliada, convertida em string e enviada para o browser.
  - Para escrever código de inicialização, utilize o método jspInit na seção de declarações (entre <%! e %>)

Objetos implícitos no JSP

- O JSP possui diversos objetos predefinidos (objetos implícitos):
  - out
  - request
  - session
- Exemplos de uso
  - out.println(“string”);
  - request.getParameter(“parameterName”)
  - session.setAttribute("username", username);
  - session.getAttribute("username");

Exemplo de aplicação JSP

```
<HTML><HEAD>
<TITLE>Seja bem-vindo</TITLE>
</HEAD><BODY>
<%
  String user =
  request.getParameter("usuario");
  if (user == null)
    user = “Mundo";
%>
<H1>Seja bem-vindo</H1>
<P>Oi, <%= user %>
</BODY></HTML>
```


JSP (Exemplo)

```
<!-- session.jsp
    checks to see if you have visited a page and keeps a counter.
    visualbuilder.com
-->
<html>
<head>
</head>

<body>

<%
 // get the value of the session variable - visitcounter
 Integer totalvisits = (Integer)session.getValue("visitcounter");

 // if the session variable (visitcounter) is null
 if (totalvisits == null)
 {
    // set session variable to 0
    totalvisits = new Integer(0);
    session.putValue("visitcounter", totalvisits);

    // print a message to out visitor
    out.println("Welcome, visitor");
 }
 else
 {
    // if you have visited the page before then add 1 to the visitcounter
    totalvisits = new Integer(totalvisits.intValue() + 1);
    session.putValue("visitcounter", totalvisits);
    out.println("You have visited this page " + totalvisits + " time(s)!");
 }
%>

</body>
</html>
```

Uso de JSP

- Páginas JSP também necessitam de Web Servers específicos.
- Em geral, um servidor de servlets também é um servidor de JSP
- Para disponibilizar um arquivo JSP basta gravá-lo em qualquer pasta visível do servidor com extensão jsp

Uso de JSP - Comparações

- JSP versus ASP
  - A parte dinâmica do JSP é escrita em Java, e não VB ou outra linguagem específica da Microsoft.
    - ASP requer uso do Windows NT e servidor ISS, enquanto JSP roda em diversos sistemas e servidores.
  - JSP é um pouco mais difícil.
  - JSP permite reuso de componentes via JavaBeans e EJB; ASP cria componentes Active X / COM.

Uso de JSP - Comparações

- JSP versus Servlets
  - JSP não permite fazer nada que servlets não façam.
  - JSP é mais fácil.
  - Para escrever o HTML em servlets, código fica cheio de comandos de escrita.
  - JSP separa conteúdo da aparência: pessoas diferentes podem trabalhar em tarefas dfierentes: web designers constróem o HTML, deixando espaços para que programadores insiram o conteúdo dinâmico.

Uso de JSP - Comparações

- JSP versus JavaScript
  - JavaScript é capaz de gerar HTML dinamicamente no cliente, mas esta funcionalidade só é útil se a informação dinâmica depende do ambiente do cliente.
  - JavaScript não pode acessar recursos do servidor, principalmente banco de dados.

Uso de JSP - Comparações

- JSP versus ASP
  - A parte dinâmica do JSP é escrita em Java, e não VB ou outra linguagem específica da Microsoft.
    - ASP requer uso do Windows NT e servidor ISS, enquanto JSP roda em diversos sistemas e servidores.
  - JSP é um pouco mais difícil.
  - JSP permite reuso de componentes via JavaBeans e EJB; ASP cria componentes Active X / COM.

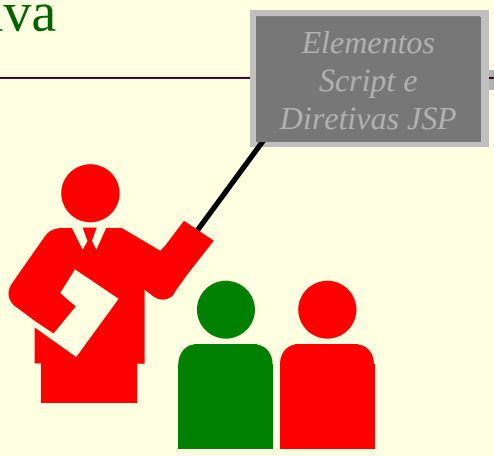
Execução de JSP

- Quando um usuário faz uma requisição a uma página JSP:
  - Se for a primeira, a página jsp é convertida em um servlet e compilada
  - Este servlet é invocado e retorna como resposta uma string HTML
  - Esta string é retornada para o cliente
- Todo o processamento ocorre no servidor, apenas o Html é retornado

Elementos JSP

- Uma página JSP pode ter três tipos de elementos:
  - Elementos de Script que são inseridos diretamente no servlet
  - Diretivas que permitem a manipulação do código gerado como servlet
  - Ações que possibilitam o uso de outros componentes, como Java Beans

POO-Java



Elementos de Script

- Os elementos de script JSP que podem ser inseridos em uma página HTML são:
  - Expressões que são avaliadas e inseridas na saída do servlet
  - Scriptlets, trechos de código inseridos no método _jspservice
  - Declarações inseridas fora dos métodos do servlet

JSP vs. HTML

- A maior parte de um arquivo jsp consiste em tags HTML
- São passadas como saída do servlet
- Páginas jsp são criadas com ferramentas HTML
- Comentários JSP são expressos como <%-- -->

Expressões

- Usadas para inserir valores diretamente na saída
- Sintaxe:

  <%= expressão java %>
- A expressão é avaliada, convertida para string e inserida na página

Expressões

- As expressões podem ser qualquer comando Java que retorne valor
- Exemplo:

  Data: <%= new java.util.Date() %>
- Expressões são muito úteis na atribuição de valores para parâmetros html:

  <font size = '<%Math.random()*5%>' >

Variáveis pré-definidas

- Existem variáveis em JSP que representam os objetos dos servlets:
  - out
  - request
  - response
  - session
- Exemplo:

  Host: <%request.getServerName()%>

Elementos XML

- As tags jsp podem ser escritas no formato XML
- Expressões JSP podem ser escritas:

  <jsp:expression>

  expressão java

  </jsp:expression>

Scriptlets

- Trechos de código Java
- Delimitados por <% e %>
- Sintaxe XML:

```
<jsp:scriptlet>
  código java
</jsp:scriptlet>
```

- Podem acessar as variáveis pré-definidas

Scriptlets

- Exemplo:

```
<%
    String cor = request.getParameter("cor");
    if (cor == null)
      cor = "white";
%>
```

Scriptlets

- Podem ser usados para apresentar html condicionalmente:

```
<%if (hora >= 6 && hora < 12) { %>
        <b> Bom Dia! </b>
<%} else {%>
        <b> Boa Tarde! </b>
<% } %>
```


Declarações

- Para definir métodos ou campos fora dos métodos do servlet gerado
- Sintaxe:

```
<%! declaração java %>
```

- Em XML:

```
<jsp:declaration>
  declaração java
</jsp:declaration>
```

Diretivas JSP

- Afetam a estrtura geral do servlet gerado da página JSP
- Possuem o seguinte formato:
- <%@diretiva atributo=“valor” %>
- ou em XML
- <jsp:directive.diretiva atributo=“valor**”/>**

Tipos de Diretivas

- Existem três tipos de diretivas JSP:
- page
  - para importação de classes, alteração do tipo do conteúdo, etc.
- include
  - para inclusão de arquivos durante a execução do JSP
- taglib
  - para definição de tags próprias usadas em bibliotecas de tags

Diretiva page

- Atributo import
- Usado para importar classes para o servlet gerado pelo JSP
- Pode aparecer várias vezes no JSP
- Exemplo:
  - <%@page import = “java.util.*” %>

Diretiva page

- Atributo contentType
- Usado para alterar o formato MIME do texto de saída
- Exemplo:
  - <%@page contentType=“text/html”%>
- Também pode ser especificado o conjunto de caracteres com charset

Diretiva page

- Atributo isThreadSafe
- Controla o uso de múltiplas threads no servlet gerado
- O padrão é usar mútithread (true)
- Exemplo:
  - <%@page isThreadSate="false"%>

Diretiva page

- Atributo session
- Indica se a página em questão faz parte de uma sessão sendo que o seu valor padrão é true
- Ex:
  - <%@page session="false" %>
- Neste caso a variável pré-definida session não pode ser acessada

Diretiva page

- Atributo buffer
- Indica o uso e tamanho do buffer usado pela variável out.
- Exemplo:
  - <%@page buffer = “32kb”%>
- ou
  - <%@page buffer = “none” %>

Diretiva page

- Atributo autoflush
- Controla o comportamento do buffer quando ele estiver cheio: true executa um “flush” automática e false gera uma exceção
- Exemplo:
  - <%@page autoflush=“true”%>

Diretiva page

- Atributo extends
- Altera a superclasse do servlet gerado
- Exemplo:
  - <%@page extends="MeuServlet.class"%>

Diretiva page

- Atributo info
- Define a string de descrição do servlet gerado
- Será retornada pelo método getServletInfo()
- Exemplo:
  - <%@page info="descrição"%>

Diretiva page

- Atributo errorPage
- Indica o nome da página que deve ser mostrada em caso de erro
- Exemplo:
  - <%@page errorPage="URL relativa"%>

Diretiva page

- Atributo isErrorPage
- Indica se a página atual pode ser usada como página de erro
- O valor padrão deste atributo é false
- Exemplo:
  - <%@page isErrorPage="true"%>

Diretiva include

- Usada para incluir outros arquivos em páginas JSP
- Possui dois formatos, o primeiro inclui arquivos em tempo de compilação e o segundo em tempo de requisição
- A diretiva deve aparecer no ponto em que o arquivo será incluído

Incluindo Arquivos (compilação)

- Sintaxe:
  - <%@include file="URL relativa"%>
- Arquivos incluídos podem ser arquivos JSP
- Se o arquivo incluído mudar, todos os JSP que o utilizam devem ser recompilados

Incluindo Arquivos (requisição)

- Sintaxe:
  - <jsp:include page="URL relativa" flush="true"%>
- Os arquivos incluídos não podem conter comandos JSP
- Se o arquivo mudar, ele será lido novamente.

Diretiva plugin

- Permite a inclusão de outras classes Java em JSP, como applets e beans
- Tem uma sintaxe semelhante a da tag applet do HTML
- Exemplo:
  - <jsp:plugin type="applet" code="Dados.class" width="200" height="100"> </jsp:plugin>

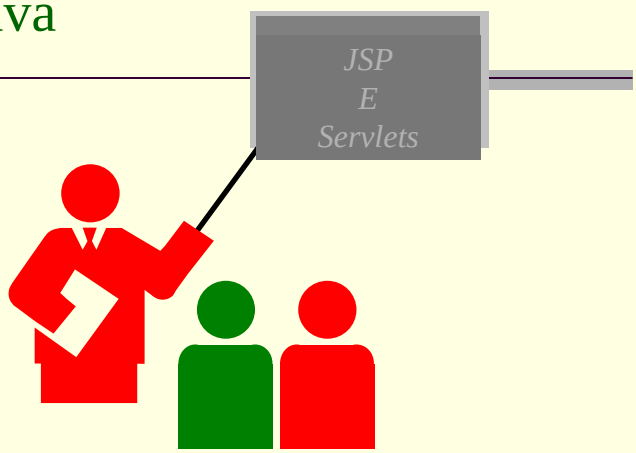
Diretiva plugin

- Para passar parâmetros para a classe envolvida com o plugin utiliza-se a diretiva jsp:params dentro de jsp:plugin
- Exemplo:
  - <jsp:params> <jsp:param name="cor" value = "red"> <jsp:params>

Diretiva plugin

- Se o browser não suportar objetos pode ser mostrado texto alternativo
- O texto é definido em jsp:fallback dentro de jsp:plugin
- Exemplo:
  - <jsp:fallback> Seu browser não suporta JAVA </jsp:fallback>

POO-Java



JSP e Servlets - Utilização

- Em um sistema Web deve-se pensar na separação entre código e visualização
- Uma das abordagens para esta separação é utilizar servlets e JSP
- Servlets para o controle e processamento dependente de programação extensiva e JSP para visualização

JSP e Servlets - Utilização

- Podemos realizar esta integração de três maneiras:
  - Através de links ou botões inseridos nas páginas Html
  - Com o uso do método sendRedirect do objeto response
  - Redirecionando o controle através de RequestDispatcher

JSP e Servlets - Utilização

- Nos dois primeiros métodos, o controle é totalmente repassado e uma nova conexão tem que ser aberta com o cliente
- Em certas situações é necessário mesclar a saída de servlet com JSP
- Para realizar esta integração é necessário usar um RequestDispatcher

Redirecionando Requisições

- Para redirecionar uma requisição de um servlet para um JSP deve-se:
  - Recuperar um redirecionador de getServletContext
  - Redirecionar o controle com forward
- Note que este redirecionamento difere de sendRedirect pois não é criada uma nova conexão

Redirecionando Requisições

- Exemplo:

```
RequestDispatcher d =
  getServletContext().
  getRequestDispatcher(url)
d.forward(request, response);
```

- A partir deste momento o controle sai do servlet atual

Redirecionando Requisições

- Se o recurso destino for estático (página HTML), o redirecionamento só funcionada para requisições GET
- Para evitar este problema, mude a extensão do recurso estático para JSP.
- Assim o recurso passa a ser dinâmico e pode tratar POST.

Enviando Informações

- Para enviar informações para o recurso destino (dinâmico):
  - Parâmetros na URL no caso de requisições GET
  - Inserir um atributo na requisição (request)
  - Inserir um atributo na sessão (session)
  - Inserir um atributo no contexto (application)

Enviando Informações (request)

- Para enviar um objeto com duração request deve-se usar antes do redirecionamento:
  - request.setAttribute(“atrib”, valor);
- Para recuperar o valor no recurso redirecionado:
  - Tipo valor = (Tipo) request.getAttribute(“atrib”);

Enviando Informações (application)

- Se o objeto deve durar durante toda a vida do servlet, usar antes do redirecionamento:
  - getServletContext().setAttribute(“atrib”, valor);
- No recurso redirecionado:
  - Tipo valor = (Tipo) getServletContext().getAttribute(“atrib”);

Enviando Informações (session)

- Para objetos de sessão, utilizar o procedimento normal:
  - HttpSession session = request.getSession(true);
  - session.setAttribute(“atrib”, valor);
- Para recuperar o atributo:
  - Tipo valor = (Tipo) session.getAttribute(“atrib”);

Recuperando Informações no JSP

- Para recuperar informações enviadas por um dos três métodos anteriores deve-se utilizar Beans com escopo:
  - <jsp:useBean id=“atrib” class=“Tipo” scope = “...”/>
- A recuperação dos parâmetros, segue a método normal

Inclusão de Saída

- O comando forward redireciona o controle e não permite que o servlet atual insira dados na saída
- Se for necessário enviar estes dados e redirecionar o controle deve ser usado o método include:
  - dispatcher.include(request, response);

Inclusão de Saída

- A principal diferencá com o forward é que podem ser enviados dados para a saída tanto antes do include como depois
- Os servlets ou JSPs chamados com este comando NÃO podem alterar o cabeçalho de resposta (setHeader)

Inclusão de Saída

- O método include tem as mesmas características do forward com relação ao envio de informações e tratamento de GET e POST
- Adicionalmente o forward define cinco atributos (setAttribute) no request que representam o caminho original

Inclusão de Saída

- Estes atributos são recuperados com request. getAttribute
  - javax.servlet.include.request_uri
  - javax.servlet.include.context_path
  - javax.servlet.include.servlet_path
  - javax.servlet.include.path_info
  - javax.servlet.include.query_string

Redirecionamento com JSP

- Embora seja mais comum o servlet redirecionar o controle, é possível redirecionar a partir de um JSP
- Este redirecionamento é feito com a tag forward:
  - <jsp:forward page="URL"/>

Objeto envolvidos em uma servlet

- HttpServletRequest contém informações provenientes do cliente.
  - Parâmetros e seus valores podem ser consultados através dos métodos getParameter
- HttpServletResponse permite gerar a saída a ser enviado ao cliente.
  - Com o auxílio de um objeto PrintWriter
  - PrintWriter out = response.getWriter();

Manipulando requisições do usuário

- getParameter(“nome”)
  - Retorna o valor da primeira ocorrência de “nome” na string de consulta
  - Funciona de forma idêntica para requisições GET e POST
  - Retorna null se o parâmetro não existe na consulta.
- getParameterValues("nome")
  - Retorna um array dos valores de todas as ocorrências do nome na consulta.
  - Retorna null se o parâmetro não existe na consulta.

Manipulando requisições do usuário (cont.)

- getParameterNames()
  - Retorna uma lista (Enumeration) de parâmetros de requisição.
- response.setContentType
  - Define o cabeçalho (Content-Type)

Problemas com Servlets

- Parar gerar páginas dinâmicas, é preciso embutir HTML ou XML dentro de instruções Java.

```
out.print("<H1>Exemplo</H1>");
for(int i = 0; i < 10/ i++) {
  out.print("<P>Um número:" + i + "</P>" );
}
out.print("<TABLE><TR><TD>...</TR></TABLE>");
```


Problemas com Servlets

- Conseqüência: dificuldade para o profissional que é somente web designer.
  - O design das páginas acaba ficando a cargo do programador, em vez do web designer.
  - Mais complicado de trabalhar do que com HTML e Javascript.
- Solução: **scripts de servidor** ou Java Server Pages

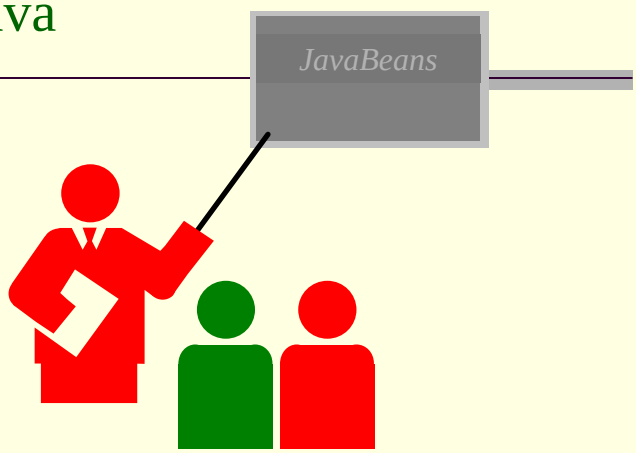
Exercício I

- Para testar a aplicação anterior (SaudarUsuario.jsp), copie o arquivo para o diretório raiz do container JSP sendo utilizado.
  - (SaudarUsuario.jsp está na pasta web/jsp dos exemplos)
  - No caso do Tomcat: CATALINA_HOME\webapps\ROOT
- A seguir, acesse a página JSP
  - http://127.0.0.1:8080/SaudarUsuario.jsp
  - http://127.0.0.1:8080/SaudarUsuario.jsp?usuario=Eduardo
- Verifique que o código Java das scriptlets não é enviado ao browser
  - *view source* no navegador WEB

Exercício II

- Este exercício testa a página ListaEmpregados.jsp
  - Esta página está na pasta web/jsp dos exemplos.
  - Copie este arquivo para CATALINA_HOME/webapps/oficina
  - Crie um descritor de contexto da aplicação.
    - (Vide o apêndice deste slide)
- Copie o driver do MySQL (.jar) para a pasta do Tomcat denominada CATALINA_HOME/common/lib.
- Levante o servidor MySQL
- Acesse a página:
  - http://localhost:8080/oficina/ListaEmpregados.jsp

POO-Java



Java Beans e JSP

- Um objeto bean pode ser declarado em uma página JSP através do tag jsp:useBean.
- Exemplo:

```
<jsp:useBean id="user" class="com.foo.UserInfoBean"/>
<% if (user.isValid()) { %>
 <p> Usuário <%= user.getFirstName() %> válido.
<% } else { %>
 <p> Usuário <%= user.getFirstName() %> inválido.
<% } %>
```

Escopo de um Java Bean

- Java Beans possuem escopo.
- JSP usa um modelo de persistência que permite manter um objeto Java (bean) em diferentes escopos.
  - Página atual (page). Este é o valor default.
  - Requisição atual (request)
  - Sessão atual (session)
  - Durante o tempo em que o contêiner estiver “no ar” (application)
- O escopo de um bean determina o quanto este bean irá “existir”.
- O atributo scope da tag jsp:useBean é utilizado para definir o escopo de um bean.

Tags JSP relacionadas a Beans

- jsp:useBean
  - Utilizada para tornar um bean disponível em uma página JSP

```
<jsp:useBean id="u" scope="session" class="exemplobean.Usuario" />
```

- jsp:setProperty
  - Utilizada para invocar um método modificador (setter)

```
<jsp:setProperty name="Bean Name" property="PropertyName“
   param="parameterName"/>
<jsp:setProperty name="Bean Name" property="PropertyName"/>
<jsp:setProperty name="Bean Name" property="*"/>
```

- jsp:getProperty
  - Utilizada para invocar um método seletor (getter)

```
<jsp:getProperty name="Nome Bean" property="propertyName"/>
```

Exemplo de uso de um Bean

- Página JSP que usa HelloBean

```
<HTML><HEAD>
<jsp:useBean id="hello" class="beans.HelloBean"/>
<jsp:setProperty name="hello" property="mensagem“
   param="usuario" />
<TITLE>Exemplo de uso de Bean</TITLE>
</HEAD>
<BODY>
<H1>Exemplo de uso de Bean</H1>
<P>Hello, <jsp:getProperty name="hello"
property="mensagem" />
</BODY></HTML>
```


Exercício III

- Esse exercício usa os arquivos DefinirUsuario.jsp, ObterUsuario.jsp e Usuario.java.
  - Todos eles estão abaixo da pasta web
- Essa aplicação tem duas partes: uma página JSP e uma classe Java (JavaBean)
  - a) Compile o JavaBean (observe que ele está em um pacote) e copie-o para o **webapps/ROOT/WEB-INF/classes**
    - **javac –d CATALINA_HOME/webapps/ROOT/WEB-INF/classes Usuario.java**

Exercício III

- b) Copie os arquivos JSP para **webapps/ROOT**
- c) Acesse via **http://localhost:8080/DefinirUsuario.jsp**
- d) Acesse via **http://localhost:8080/ObterUsuario.jsp**