Curso de Java

Geração de Páginas WEB através de JSP

- Servlets constituem um mecanismo conveniente para a geração de páginas HTML dinâmicas e seu tratamento no servidor.
- Em muitos casos, no entanto, o seu uso acaba sendo demasiado trabalhoso
  - As páginas devem aderir a um padrão de apresentação comum.
  - Componentes visuais específicos podem ser necessários.
  - Uso de ferramentas específicas para a geração das páginas.

- JSP – *Java Server Pages* – constituem um mecanismo orientado à página HTML sendo gerado, ao invés de ser orientado a uma classe Java que gera a página.
- JSP são baseados em Servlets de forma que todos os recursos disponibilizados aos Servlets pelo ambiente Java podem ser utilizados.

O servidor de aplicações Java é responsável pela tradução do código JSP para o Servlet correspondente e também pela compilação do Servlet. Esse processo ocorre quando a página JSP é acessada pela primeira vez.

JSP – primeiro exemplo

- Uma página JSP é uma página HTML onde se insere código Java dentro de 'comentários' especiais.
- O código Java é executado no servidor quando este processa a requisição da página.

```
<HTML>
  <BODY>
     Data de hoje: <%= new java.util.Date() %>
  </BODY>
</HTML>
```

- Neste exemplo, o código Java é uma expressão, delimitada por `<%=` e `%>`. Neste caso, a expressão, depois de calculada é convertida num String que substitui todo o 'comentário'.

- Além de expressões inseridas no texto da página, é possível inserir trechos de código Java que são executados no servidor ao se processar a página.
- Esses trechos de código são chamados ‘scriptlets’ e delimitados por <% e %>.
- A execução de um ‘scriptlet’ ocorre durante a apresentação do texto correspondente à página HTML. Os ‘scriptlets’ são portanto executados na ordem em que aparecem na página.

- Neste exemplo, a variável **`dataHoje`** é definida no scriptlet e utilizada numa expressão Java no 'corpo' da página.

```
<HTML>
  <BODY>
    <%
     // código do scriptlet
     System.out.println( "Calculando a data de hoje" );
     java.util.Date dataHoje = new java.util.Date();
    %>
    Data de hoje: <%= dataHoje %>
</BODY>
</HTML>
```

JSP – outro exemplo de Scriptlet

- Um scriptlet pode gerar o texto da página HTML.
- Para isso, usa-se a variável pré-definida **`out`** da classe Writer.

```
<HTML>
  <BODY>
     <%  // código do scriptlet
     System.out.println( "Calculando a data de hoje" );
     java.util.Date dataHoje = new java.util.Date();
    %>
    Data de hoje: <% out.print(dataHoje); %>
  </BODY>
</HTML>
```

JSP – variáveis pré-definidas

- A página JSP tem diversas variáveis pré-definidas que podem ser utilizadas nos scriptlets. Neste exemplo usa-se a variável **`request`**.

```
<HTML>
  <BODY>
   <%
     System.out.println( "Calculando a data de hoje" );
     java.util.Date dataHoje = new java.util.Date();
    %>
    Endereço do servidor:
    <% out.print(request.getRemoteHost()); %>
  </BODY>
</HTML>
```

JSP – intercalando HTML e Scriptlet

- Numa página JSP, os Scriptlets podem ser intercalados ao código HTML.

```
<TABLE BORDER=2>
 <%
     for ( int i = 0; i < n; i++ ) {
 %>
     <TR>
         <TD>Valor:</TD> <TD> <%= i+1%> </TD>
     </TR>
 <%
     }
 %>
</TABLE>
```

- Páginas JSP podem fazer uso de diretivas. O exemplo abaixo usa a diretiva 'import'.

```
<%@ page import="java.util.*" %>
<HTML>
 <BODY>
  <%
     System.out.println( "Evaluating date now" );
     Date date = new Date();
  %>
  Data de hoje: <%= date %>
 </BODY>
</HTML>
```

JSP – diretiva 'include'

- A diretiva 'include' permite incluir o conteúdo de uma página JSP em outra.

```
<HTML>
<BODY>
   Incluindo exemplo1.jsp ...
  <BR> <%@ include file="exemplo1.jsp" %>
</BODY>
</HTML>
```

JSP – declarações

- Uma página JSP pode conter declarações (métodos, constantes, variáveis) delimitadas por <%! e %> .
- Declarações são diferentes de scriptlets porque não contém código Java para ser executado.

```
<%@ page import="java.util.*" %>
<HTML>
<BODY>
  <%!   Date theDate = new Date();
        Date getDate(){
          System.out.println( "executando getDate()" );
          return theDate;
        }
  %>
  Data de hoje: <%= getDate() %>
</BODY>
</HTML>
```

JSP – Uso da sessão de usuário

- Servlets e JSP's mantém o controle da sessão de usuário através do objeto 'session'.
- O objeto 'session' associar valores a nomes, e essa associação pode ser compartilhada entre as páginas usadas durante uma 'sessão'.
- Em JSP a sessão é accessível através da variável 'session', pré – definida.
- Num servlet, a sessão é accessível através da chamada a `request.getSession();`
- A associação de um nome a um valor é feita através da chamada a `session.setAttribute(nome,valor);`
- O acesso ao valor associado a um nome é feito através da chamada a `session.getAttribute(valor);`

- Um exemplo (1): Na página abaixo o usuário preenche um formulário que ao ser submetido, é encaminhado a uma página JSP (JSP_9b.jsp).

```
<html>
<head>
<title>Exemplo JSP 9a (Uso da sessão)</title>
</head>
  <BODY>
    <FORM METHOD=POST ACTION="JSP_9b.jsp">
       Nome: <INPUT TYPE=TEXT NAME=username SIZE=20>
       <P><INPUT TYPE=SUBMIT VALUE= "Ok">
    </FORM>
  </BODY>
</HTML>
```

- Um exemplo (2): Na página JSP (JSP_9b.jsp) abaixo, o dado 'username' da requisição é passado à sessão, associado ao nome 'theName'.

```
<%
   String name = request.getParameter( "username" );
   session.setAttribute( "theName", name );
%>
<html>
  <head>
     <title>Exemplo JSP_9b (Uso da sessão)</title>
  </head>
  <body>
    <A HREF="JSP_9c.jsp">continue, <%= name %></A>
  </body>
</html>
```

- Um exemplo (3): Na página JSP (JSP_9c.jsp) abaixo, o dado da sessão é usado numa expressão no corpo da página.

```
<html>
  <head>
     <title>Insert title here</title>
  </head>
  <body>
     Parabéns, <%= session.getAttribute( "theName" ) %>
  </body>
</html>
```

- Os dados preenchidos pelo usuário num formulário (JSP, Servlet ou HTML puro) devem ser disponibilizados à aplicação.
- Para facilitar a transferência de dados entre as páginas e as partes da aplicação, o mecanismo JSP permite que se use um objeto específico para manter os dados do formulário. Esse tipo de classe é conhecido como 'bean' e deve obedecer ao seguinte padrão:
  - Para cada atributo attr, a classe deve
    - Exportar um método getAttr() que devolve o valor do atributo.
    - Exportar um método setAttr() que define/altera o valor do atributo.

- Um exemplo (1): *bean* para manter os dados associados a um Aluno.

```
public class Aluno {
   String matricula;
   String nome;
   String rg;

   public void setMatricula(String m) { matricula = m; }
   public void setNome(String n)      { nome = n;       }
   public void setRg(String r)        { rg = r;         }

   public String getMatricula() { return matricula; }
   public String getNome()      { return nome;      }
   public String getRg()        { return rg;        }
}
```

- Um exemplo (2): No formulário, os campos de entrada têm os mesmos nomes que os atributos do bean: matricula, nome e rg.

```
<html>
<head>
  <title>Exemplo 10 (Uso de beans)</title>
</head>
<BODY>
  <FORM METHOD=POST ACTION="JSP_10b.jsp">
    <h3>Dados do aluno</h3>
    Matricula: <INPUT TYPE=TEXT NAME=matricula SIZE=20><BR>
    Nome: <INPUT TYPE=TEXT NAME=nome SIZE=30><BR>
    RG: <INPUT TYPE=TEXT NAME=rg SIZE=4>
    <P><INPUT TYPE=SUBMIT VALUE="OK">
  </FORM>
</BODY>
</HTML>
```

Um exemplo (3):

- A diretiva useBean define a classe a ser usada como bean, o nome do objeto e seu escopo (sessão ou requisição).
- A diretiva setProperty indica quais atributos devem ser associados aos dados da resposta (“*” indica ‘todos’).

```
<jsp:useBean id="aluno" class="exJSP.Aluno" scope="session"/>
<jsp:setProperty name="aluno" property="*"/> <html>
<head>
<title>Exemplo 10b JSP (Uso de beans)</title>
</head>
<body>
<A HREF="JSP_10c.jsp">Continue</A>
</body>
</html>
```

Um exemplo (4): Nesta página, os dados digitados no formulário HTML são acessados através do bean, usando os métodos padrão getAttr().

```
<jsp:useBean id="aluno" class="exJSP.Aluno" scope="session"/>
<html>
<head>
   <title>Exemplo 10c JSP (uso de beans)</title>
</head>
  <body>
     <h3>Dados de entrada:</h3>
     Matricula: <%= aluno.getMatricula() %> <BR>
     Nome:      <%= aluno.getNome()      %> <BR>
     RG:        <%= aluno.getRg()        %> <BR>
  </body>
</html>
```

Curso de Java

Geração de Páginas WEB através de JSP

- Servlets constituem um mecanismo conveniente para a geração de páginas HTML dinâmicas e seu tratamento no servidor.
- Em muitos casos, no entanto, o seu uso acaba sendo demasiado trabalhoso
  - As páginas devem aderir a um padrão de apresentação comum.
  - Componentes visuais específicos podem ser necessários.
  - Uso de ferramentas específicas para a geração das páginas.

- JSP – *Java Server Pages* – constituem um mecanismo orientado à página HTML sendo gerado, ao invés de ser orientado a uma classe Java que gera a página.
- JSP são baseados em Servlets de forma que todos os recursos disponibilizados aos Servlets pelo ambiente Java podem ser utilizados.

O servidor de aplicações Java é responsável pela tradução do código JSP para o Servlet correspondente e também pela compilação do Servlet. Esse processo ocorre quando a página JSP é acessada pela primeira vez.

JSP – primeiro exemplo

- Uma página JSP é uma página HTML onde se insere código Java dentro de 'comentários' especiais.
- O código Java é executado no servidor quando este processa a requisição da página.

```
<HTML>
  <BODY>
     Data de hoje: <%= new java.util.Date() %>
  </BODY>
</HTML>
```

- Neste exemplo, o código Java é uma expressão, delimitada por `<%=` e `%>`. Neste caso, a expressão, depois de calculada é convertida num String que substitui todo o 'comentário'.

- Além de expressões inseridas no texto da página, é possível inserir trechos de código Java que são executados no servidor ao se processar a página.
- Esses trechos de código são chamados ‘scriptlets’ e delimitados por <% e %>.
- A execução de um ‘scriptlet’ ocorre durante a apresentação do texto correspondente à página HTML. Os ‘scriptlets’ são portanto executados na ordem em que aparecem na página.

- Neste exemplo, a variável **`dataHoje`** é definida no scriptlet e utilizada numa expressão Java no 'corpo' da página.

```
<HTML>
  <BODY>
    <%
     // código do scriptlet
     System.out.println( "Calculando a data de hoje" );
     java.util.Date dataHoje = new java.util.Date();
    %>
    Data de hoje: <%= dataHoje %>
</BODY>
</HTML>
```

JSP – outro exemplo de Scriptlet

- Um scriptlet pode gerar o texto da página HTML.
- Para isso, usa-se a variável pré-definida **`out`** da classe Writer.

```
<HTML>
  <BODY>
     <%  // código do scriptlet
     System.out.println( "Calculando a data de hoje" );
     java.util.Date dataHoje = new java.util.Date();
    %>
    Data de hoje: <% out.print(dataHoje); %>
  </BODY>
</HTML>
```

JSP – variáveis pré-definidas

- A página JSP tem diversas variáveis pré-definidas que podem ser utilizadas nos scriptlets. Neste exemplo usa-se a variável **`request`**.

```
<HTML>
  <BODY>
   <%
     System.out.println( "Calculando a data de hoje" );
     java.util.Date dataHoje = new java.util.Date();
    %>
    Endereço do servidor:
    <% out.print(request.getRemoteHost()); %>
  </BODY>
</HTML>
```

JSP – intercalando HTML e Scriptlet

- Numa página JSP, os Scriptlets podem ser intercalados ao código HTML.

```
<TABLE BORDER=2>
 <%
     for ( int i = 0; i < n; i++ ) {
 %>
     <TR>
         <TD>Valor:</TD> <TD> <%= i+1%> </TD>
     </TR>
 <%
     }
 %>
</TABLE>
```

- Páginas JSP podem fazer uso de diretivas. O exemplo abaixo usa a diretiva 'import'.

```
<%@ page import="java.util.*" %>
<HTML>
 <BODY>
  <%
     System.out.println( "Evaluating date now" );
     Date date = new Date();
  %>
  Data de hoje: <%= date %>
 </BODY>
</HTML>
```

JSP – diretiva 'include'

- A diretiva 'include' permite incluir o conteúdo de uma página JSP em outra.

```
<HTML>
<BODY>
   Incluindo exemplo1.jsp ...
  <BR> <%@ include file="exemplo1.jsp" %>
</BODY>
</HTML>
```

JSP – declarações

- Uma página JSP pode conter declarações (métodos, constantes, variáveis) delimitadas por <%! e %> .
- Declarações são diferentes de scriptlets porque não contém código Java para ser executado.

```
<%@ page import="java.util.*" %>
<HTML>
<BODY>
  <%!   Date theDate = new Date();
        Date getDate(){
          System.out.println( "executando getDate()" );
          return theDate;
        }
  %>
  Data de hoje: <%= getDate() %>
</BODY>
</HTML>
```

JSP – Uso da sessão de usuário

- Servlets e JSP's mantém o controle da sessão de usuário através do objeto 'session'.
- O objeto 'session' associar valores a nomes, e essa associação pode ser compartilhada entre as páginas usadas durante uma 'sessão'.
- Em JSP a sessão é accessível através da variável 'session', pré – definida.
- Num servlet, a sessão é accessível através da chamada a `request.getSession();`
- A associação de um nome a um valor é feita através da chamada a `session.setAttribute(nome,valor);`
- O acesso ao valor associado a um nome é feito através da chamada a `session.getAttribute(valor);`

- Um exemplo (1): Na página abaixo o usuário preenche um formulário que ao ser submetido, é encaminhado a uma página JSP (JSP_9b.jsp).

```
<html>
<head>
<title>Exemplo JSP 9a (Uso da sessão)</title>
</head>
  <BODY>
    <FORM METHOD=POST ACTION="JSP_9b.jsp">
       Nome: <INPUT TYPE=TEXT NAME=username SIZE=20>
       <P><INPUT TYPE=SUBMIT VALUE= "Ok">
    </FORM>
  </BODY>
</HTML>
```

- Um exemplo (2): Na página JSP (JSP_9b.jsp) abaixo, o dado 'username' da requisição é passado à sessão, associado ao nome 'theName'.

```
<%
   String name = request.getParameter( "username" );
   session.setAttribute( "theName", name );
%>
<html>
  <head>
     <title>Exemplo JSP_9b (Uso da sessão)</title>
  </head>
  <body>
    <A HREF="JSP_9c.jsp">continue, <%= name %></A>
  </body>
</html>
```

- Um exemplo (3): Na página JSP (JSP_9c.jsp) abaixo, o dado da sessão é usado numa expressão no corpo da página.

```
<html>
  <head>
     <title>Insert title here</title>
  </head>
  <body>
     Parabéns, <%= session.getAttribute( "theName" ) %>
  </body>
</html>
```

- Os dados preenchidos pelo usuário num formulário (JSP, Servlet ou HTML puro) devem ser disponibilizados à aplicação.
- Para facilitar a transferência de dados entre as páginas e as partes da aplicação, o mecanismo JSP permite que se use um objeto específico para manter os dados do formulário. Esse tipo de classe é conhecido como 'bean' e deve obedecer ao seguinte padrão:
  - Para cada atributo attr, a classe deve
    - Exportar um método getAttr() que devolve o valor do atributo.
    - Exportar um método setAttr() que define/altera o valor do atributo.

- Um exemplo (1): *bean* para manter os dados associados a um Aluno.

```
public class Aluno {
   String matricula;
   String nome;
   String rg;

   public void setMatricula(String m) { matricula = m; }
   public void setNome(String n)      { nome = n;       }
   public void setRg(String r)        { rg = r;         }

   public String getMatricula() { return matricula; }
   public String getNome()      { return nome;      }
   public String getRg()        { return rg;        }
}
```

- Um exemplo (2): No formulário, os campos de entrada têm os mesmos nomes que os atributos do bean: matricula, nome e rg.

```
<html>
<head>
  <title>Exemplo 10 (Uso de beans)</title>
</head>
<BODY>
  <FORM METHOD=POST ACTION="JSP_10b.jsp">
    <h3>Dados do aluno</h3>
    Matricula: <INPUT TYPE=TEXT NAME=matricula SIZE=20><BR>
    Nome: <INPUT TYPE=TEXT NAME=nome SIZE=30><BR>
    RG: <INPUT TYPE=TEXT NAME=rg SIZE=4>
    <P><INPUT TYPE=SUBMIT VALUE="OK">
  </FORM>
</BODY>
</HTML>
```

Um exemplo (3):

- A diretiva useBean define a classe a ser usada como bean, o nome do objeto e seu escopo (sessão ou requisição).
- A diretiva setProperty indica quais atributos devem ser associados aos dados da resposta (“*” indica ‘todos’).

```
<jsp:useBean id="aluno" class="exJSP.Aluno" scope="session"/>
<jsp:setProperty name="aluno" property="*"/> <html>
<head>
<title>Exemplo 10b JSP (Uso de beans)</title>
</head>
<body>
<A HREF="JSP_10c.jsp">Continue</A>
</body>
</html>
```

Um exemplo (4): Nesta página, os dados digitados no formulário HTML são acessados através do bean, usando os métodos padrão getAttr().

```
<jsp:useBean id="aluno" class="exJSP.Aluno" scope="session"/>
<html>
<head>
   <title>Exemplo 10c JSP (uso de beans)</title>
</head>
  <body>
     <h3>Dados de entrada:</h3>
     Matricula: <%= aluno.getMatricula() %> <BR>
     Nome:      <%= aluno.getNome()      %> <BR>
     RG:        <%= aluno.getRg()        %> <BR>
  </body>
</html>
```