Linguagens de Programação para a Internet
Linguagens Web

JSP

- os scripts escritos na linguagem JSP tem extensão .jsp
- o JSP permite colocar código Java dentro de HTML
- uma página JSP contém dois tipos de texto: conteúdo estáctico (baseado em texto HTML, SVG, WML ou XML) e conteúdo dinâmico (com elementos JSP)

- www.jsptut.com
- http://www.jsptutorial.net/
- http://www.roseindia.net/jsp/jsp.htm
- http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro.html
- http://java.sun.com/products/jsp/syntax/1.2/syntaxref12.pdf

- pode ser colocado conteúdo dinâmico dentro de código HTML:

```
<HTML>
<BODY>
Hello!  The time is now <%= new java.util.Date() %>
</BODY>
</HTML>
```

```
<HTML>
<BODY>
<% out.println (“O meu primeiro script JSP”); %>
</BODY>
</HTML>
```

- o conteúdo dinâmico pode ser criado dentro dos elementos de *scripting* através de objectos da linguagem Java
- dentro das páginas JSP pode-se aceder a uma grande variedade de objectos, por exemplo componentes JavaBeans
- a tecnologia JSP automaticamente disponibiliza alguns objectos
- os objectos implicitos são criados pelo web container, e contem informação relacionada com pedidos, páginas ou aplicações
- alguns dos objectos são definidos pela tecnologia Java Servlet contida na tecnologia JSP

- os objectos implicitos:

Variable	Class	Description
application	javax.servlet.ServletContext	The context for the JSP page's servlet and any Web components contained in the same application. See Accessing the Web Context.
config	javax.servlet.ServletConfig	Initialization information for the JSP page's servlet.
exception	java.lang.Throwable	Accessible only from an error page. See Handling Errors.
out	javax.servlet.jsp.JspWriter	The output stream.
page	java.lang.Object	The instance of the JSP page's servlet processing the current request. Not typically used by JSP page authors.
pageContext	javax.servlet.jsp.PageContext	The context for the JSP page. Provides a single API to manage the various scoped attributes described in Sharing Information. This API is used extensively when implementing tag handlers. See Tag Handlers.
request	Subtype of javax.servlet.ServletRequest	The request triggering the execution of the JSP page. See Getting Information from Requests.
response	Subtype of javax.servlet.ServletResponse	The response to be returned to the client. Not typically used by JSP page authors.
session	javax.servlet.http.HttpSession	The session object for the client. See Accessing the Web Context.

- o *scripting* JSP permite inserir codígo Java na *servlet* gerada pela página JSP

- os elementos de scripting (JSP Tags):
  - comment
  - expression
  - scriptlet
  - declaration
  - directive
  - action

- JSP comment: serve para inserir um comentário de forma a explicar a lógica do código

- <%-- --%>

```
<%-- This is a JSP comment --%>

<%--
  This is a JSP comment can span
  in multiple lines
--%>
```

- JSP expression: serve para gerar *output*, sendo convertido automaticamente em *string*

- <%= %>

```
<%= new java.util.Date()%>
```

- a sintaxe XML para uma JSP expression:

```
<jsp:expression>
    Java Expression
</jsp:expression>
```

- JSP *scriptlet*: é semelhante à *tag expression*, mas sem o sinal de "=", e serve para inserir código Java

- <% %>

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
  <head>
    <title>JSP syntax</title>
  </head>
  <body>
    <%
        // using scriptlet
        java.util.Calendar now = new java.util.GregorianCalendar();
        String tod = "";

        if (now.get(now.HOUR_OF_DAY) < 12) {
          tod = "Morning!";
        } else if (now.get(now.HOUR_OF_DAY) < 18) {
          tod = "Afternoon!";
        } else {
          tod = "Evening!";
        }
    %>

    Good <%=tod%>
  </body>
</html>
```

```
<HTML>
<BODY>
<%
   // This scriptlet declares and initializes "date"
   System.out.println( "Evaluating date now" );
   java.util.Date date = new java.util.Date();
%>
Hello!  The time is now
<%
   // This scriptlet generates HTML output
   out.println( String.valueOf( date ));
%>
</BODY>
</HTML>
```

- a sintaxe XML para uma JSP *scriplet*:

```
<jsp:scriptlet>
  // Java code of scriptlet
</jsp:scriptlet>
```

- JSP declaration: serve para declarar variáveis ou métodos. "*The difference between a variable using declaration and a variable is declared using Scriptlet is that a variable declare using declaration tag is accessible by all the methods while a variable declared using Scriptlet is only accessible to the method _jspservice of the generated servlet from JSP page*"

- <%! %>

```
<%! int x = 10; %>
```

```
<%!
      public boolean isInRange(int x,int min,int max){
         return x >= min && x <= max;
      }
%>
```

```
<%! int i = 0; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```

- a sintaxe XML para uma JSP declaration:

```
<jsp:declaration>
  Java declaration
</jsp:declaration>
```

- JSP directive: fornece informação especial sobre a página ao JSP engine, alterando a forma como este processa a página. A tag directive permite importar packages, fazer o tratamento de erros ou sessões.

- <%@ %>

```
<%@ page
  [ language="java" ]
  [ extends="package.class" ]
  [ import="{package.class | package.*}, ..." ]
  [ session="true|false" ]
  [ buffer="none|8kb|sizekb" ]
  [ autoFlush="true|false" ]
  [ isThreadSafe="true|false" ]
  [ info="text" ]
  [ errorPage="relativeURL" ]
  [ contentType="mimeType [ ; charset=characterSet ]" |
    "text/html ; charset=ISO-8859-1" ]
  [ isErrorPage="true|false" ]
  [ pageEncoding="characterSet | ISO-8859-1" ]
%>
```

- a sintaxe XML para uma JSP directive:

```
<jsp:directive.page>
    pageDirective
</jsp:directive.page>
```

- exemplos:

```
<%@ page import="java.util.*, java.lang.*" %>
<%@ page buffer="5kb" autoFlush="false" %>

<jsp:directive.page errorPage="error.jsp" />

<%@ page session=”false" %>

<%@ include file = "include/connect.jsp”%>
```

- existem três tipos de tags directive:
  - page
  - include
  - tag lib
- atributos para a tag page directive:
  - language
  - extends
  - import
  - session
  - suffer
  - autoFlush
  - isThreadSafe
  - info
  - errorPage
  - isErrorPage
  - contentType

- JSP action: as *action tags* são usadas para transferir o controlo entre páginas assim como para possibilitar o uso de server side JavaBeans

- as três action tags mais comuns são:
  - include
  - forward
  - useBean

- sintaxe da JSP action include:

```
<jsp:include page="{relativeURL | <%= expression %>}" flush="true" />
      <jsp:param name="parameterName" value="{parameterValue |
            <%= expression %>}" />
</jsp:include>
```

- exemplos de JSP action include:

```
<jsp:include page="example/test.jsp">
      <jsp:include name="username" value=”admin" />
</jsp:include>
```

```
<HTML>
<BODY>
Going to include hello.jsp...<BR>
<jsp:include page="hello.jsp"/>
</BODY>
</HTML>
```

- sintaxe da JSP action forward:

```
<jsp:forward page ="{relativeURL | <%= expression %>}" />
    <jsp:param   name ="parameterName"
                value="parameterValue" | <%= expression %>}" />
</jsp:forward>
```

- exemplo de JSP action forward:

```
<jsp:forward page ="/example/connect.jsp" />
    <jsp:param   name ="username" value="admin" />
</jsp:forward>
```

- sendRedirect em JSP : é um método da interface ***HttpServletResponse***

```
<%
	String name = request.getParameter("name");
	String password = request.getParameter("pwd");
	if(name.equals("Williams") && password.equals("abcde"))
	{
		response.sendRedirect("RedirectIfSuccessful.html");
	}
	else
	{
		response.sendRedirect("RedirectIfFailed.html");
	}
%>
```

- sintaxe da JSP action useBean:

```
<jsp:useBean
   id="beanInstanceName"
   scope="page|request|session|application"
   {
     class="package.class" |
     type="package.class" |
     class="package.class" type="package.class" |
     beanName="{package.class | <%= expression %>}"
     type="package.class"
   }
   { /> |
     > other elements
     </jsp:useBean>
   }
```

- exemplo de JSP action useBean:

```
<HTML>
<BODY>
<FORM METHOD=POST ACTION="SaveName.jsp">
What's your name? <INPUT TYPE=TEXT NAME=username SIZE=20><BR>
What's your e-mail address? <INPUT TYPE=TEXT NAME=email SIZE=20><BR>
What's your age? <INPUT TYPE=TEXT NAME=age SIZE=4>
<P><INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

UserData.java

```
package user;

public class UserData {
   String username;
   String email;
   int age;

   public void setUsername( String value ) { username = value;}

   public void setEmail( String value ) { email = value; }

   public void setAge( int value ) { age = value;}

   public String getUsername() { return username; }

   public String getEmail() { return email; }

   public int getAge() { return age; }
}
```

- exemplo de JSP action useBean (cont.):

SaveName.jsp

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>
<jsp:setProperty name="user" property="*"/>
<HTML>
<BODY>
<A HREF="NextPage.jsp">Continue</A>
</BODY>
</HTML>
```

NextPage.jsp

```
<jsp:useBean id="user" class="user.UserData" scope="session"/>
<HTML>
<BODY>
You entered<BR>
Name: <%= user.getUsername() %><BR>
Email: <%= user.getEmail() %><BR>
Age: <%= user.getAge() %><BR>
</BODY>
</HTML>
```

- exemplos de JSP scriptlet e HTML:

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

```
<%
   if ( hello ) {
      %>
      <P>Hello, world
      <%
   } else {
      %>
      <P>Goodbye, world
      <%
   }
%>
```

exemplos de JSP session:

PrimeiraPagina.jsp

```
<HTML>
<BODY>
<FORM METHOD=POST ACTION="SegundaPagina.jsp">
What's your name? <INPUT TYPE=TEXT NAME=username SIZE=20>
<P><INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```

SegundaPagina.jsp

```
<%
  String name = request.getParameter( "username" );
  session.setAttribute( "theName", name );
%>
<HTML>
<BODY>
<A HREF="TerceiraPagina.jsp">Continue</A>
</BODY>
</HTML>
```

TerceiraPagina.jsp

```
<HTML>
<BODY>
Hello, <%= session.getAttribute( "theName" ) %>
<A HREF="PrimeiraPagina.jsp">Continue</A>
</BODY>
</HTML>
```

funções para manipular JSP session:

1	public Object **getAttribute**(String name) This method returns the object bound with the specified name in this session, or null if no object is bound under the name.
2	public Enumeration **getAttributeNames**() This method returns an Enumeration of String objects containing the names of all the objects bound to this session.
3	public long **getCreationTime**() This method returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
4	public String **getId**() This method returns a string containing the unique identifier assigned to this session.
5	public long **getLastAccessedTime**() This method returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
6	public int **getMaxInactiveInterval**() This method returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.
7	public void **invalidate**() This method invalidates this session and unbinds any objects bound to it.
8	public boolean **isNew**(This method returns true if the client does not yet know about the session or if the client chooses not to join the session.
9	public void **removeAttribute**(String name) This method removes the object bound with the specified name from this session.
10	public void **setAttribute**(String name, Object value) This method binds an object to this session, using the name specified.
11	public void **setMaxInactiveInterval**(int interval) This method specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

exemplos de JSP session:

```
<%@ page import="java.io.*,java.util.*" %>
<%
  // Get session creation time.
  Date createTime = new Date(session.getCreationTime());
  // Get 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);
%>
<html>
<head>
<title>Session Tracking</title>
…
```

registration.jsp

```
<%@ page import ="java.sql.*" %>
<%
    String user = request.getParameter("uname");
    String pwd = request.getParameter("pass");
    String fname = request.getParameter("fname");
    String lname = request.getParameter("lname");
    String email = request.getParameter("email");
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbname",
          "root", "dbpass");
    Statement st = con.createStatement();
    int i = st.executeUpdate("insert into members(first_name, last_name, email, uname, pass, regdate)
values ('" + fname + "','" + lname + "','" + email + "','" + user + "','" + pwd + "', CURDATE())");

 if (i > 0) {
        session.setAttribute("userid", user);
        out.print("Registration Successfull!"+"<a href='welcome.jsp'>Go to Welcome</a>");
    } else {
        response.sendRedirect("index.jsp");
    }
%>
```

welcome.jsp

```
Registration is Successful.
Please Login Here <a href='index.jsp'>Go to Login</a>
```

login.jsp

```
<%@ page import ="java.sql.*" %>
<%
    String userid = request.getParameter("uname");
    String pwd = request.getParameter("pass");
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbname",
          "root", "dbpass");
    Statement st = con.createStatement();
    ResultSet rs;
    rs = st.executeQuery("select * from members where uname='" + userid + "' and pass='" + pwd +
"'");
    if (rs.next()) {
        session.setAttribute("userid", userid);
        response.sendRedirect("success.jsp");
} else {
        out.println("Invalid password <a href='index.jsp'>try again</a>");
    }
%>
```

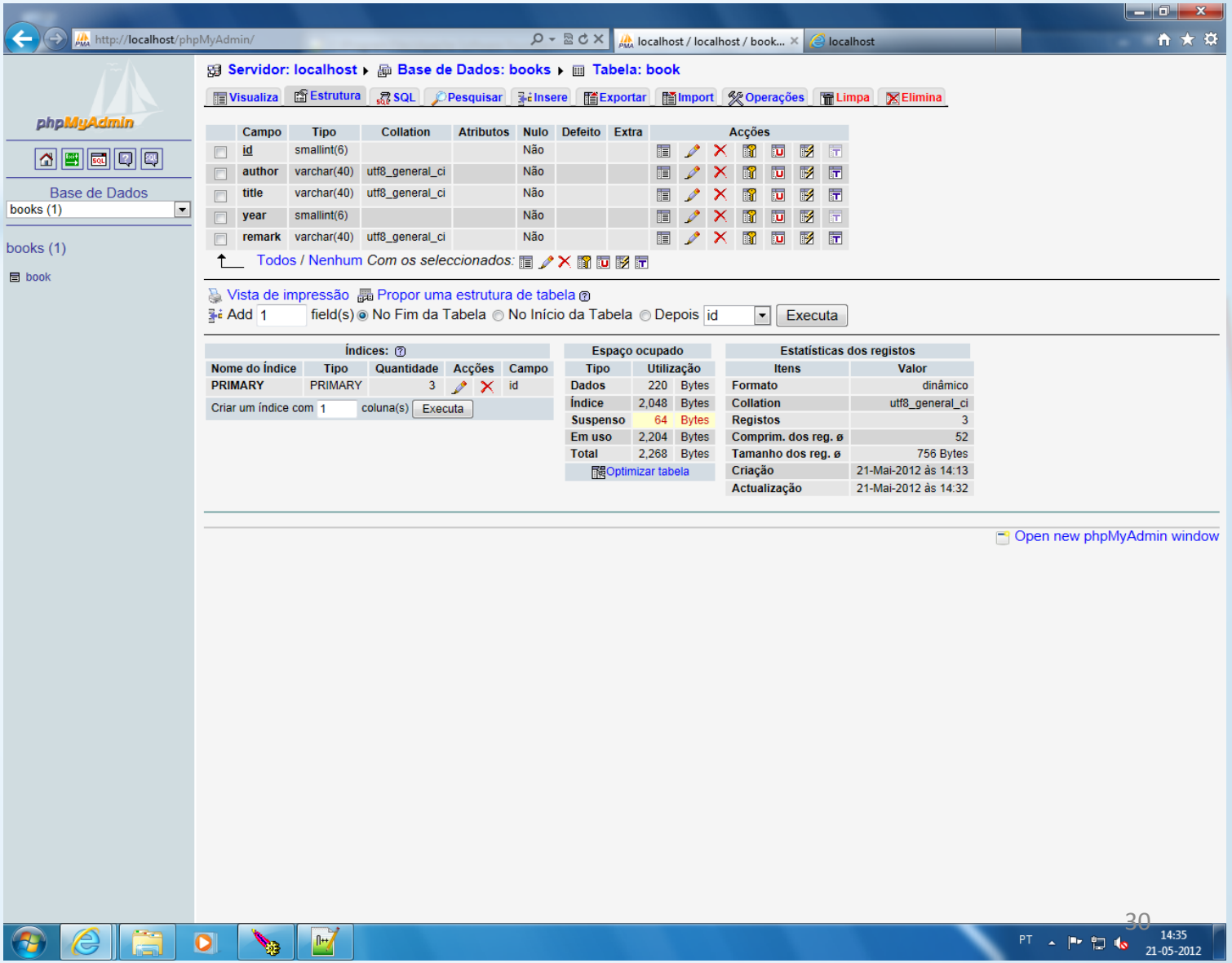
sucess.jsp

```
<%
   if ((session.getAttribute("userid") == null) || (session.getAttribute("userid") == "")) {
%>
You are not logged in<br/>
<a href="index.jsp">Please Login</a>
<%} else {
%>
Welcome <%=session.getAttribute("userid")%>
<a href='logout.jsp'>Log out</a>
<%
   }
%>
```

logout.jsp

```
<%
session.setAttribute("userid", null);
session.invalidate();
response.sendRedirect("index.jsp");
%>
```

- Após criar a tabela "book" na base de dados "books"



O seguinte exemplo permite fazer insert

```
<%@ page language="java" import="java.sql.*" %>

<%
String url = "jdbc:mysql://localhost:3306/books";


     String id ="101";
     String author= "Manel";
     String title= "Nao Sei";
     String year = "1999";
     String remark = "Pois";

    try {

       String insert = "INSERT INTO book (id, author, title, year, remark)" +
             "VALUES (?, ?, ?, ?, ?)";

       Class.forName("com.mysql.jdbc.Driver");
       Connection con = DriverManager.getConnection(url, "root", "root");

       PreparedStatement ps = con.prepareStatement(insert);

                          ps.setString(1, id);
       ps.setString(2, author);
       ps.setString(3, title);
       ps.setString(4, year);
       ps.setString(5, remark);
       ps.executeUpdate();
       con.close();

    }
    catch (Exception ex) {
       out.println ("ERRO: Insert");
    }
```

- O seguinte exemplo permite fazer insert

```
Connection dbConnection = null;
Statement statement = null;

String insertTableSQL = "INSERT INTO DBUSER"
                + "(USER_ID, USERNAME, CREATED_BY, CREATED_DATE) " + "VALUES"
                + "(1,'mkyong','system', " + "to_date('"
                + getCurrentTimeStamp() + "', 'yyyy/mm/dd hh24:mi:ss')) »;
try {
        dbConnection = getDBConnection();
        statement = dbConnection.createStatement();

        System.out.println(insertTableSQL);

        // execute insert SQL stetement
        statement.executeUpdate(insertTableSQL);

        System.out.println("Record is inserted into DBUSER table!");

} catch (SQLException e) {

        System.out.println(e.getMessage());

} finally {

        if (statement != null) {
                statement.close();
        }

        if (dbConnection != null) {
                dbConnection.close();
        }
}
```

select

```
try {

        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(url, "root", "root");

        Statement stmt = con.createStatement();

        ResultSet result = stmt.executeQuery("SELECT * FROM book");

        while(result.next())
        {
          out.println (result.getString("id"));
          out.println (result.getString("author"));
          out.println (result.getString("title"));
          out.println (result.getString("year"));
          out.println (result.getString("remark"));
        }

        con.close();

    } catch (Exception ex) {
        out.println ("ERRO: Select");
    }
```

delete

```
id ="101";

    try {
       String delete = "DELETE from book WHERE id = ?";

       Class.forName("com.mysql.jdbc.Driver");
       Connection con = DriverManager.getConnection(url, "root", "root");
       PreparedStatement ps = con.prepareStatement(delete);

       ps.setString(1, id);
       ps.executeUpdate();
       con.close();

    } catch (Exception ex) {
       out.println ("ERRO: Delete");
    }


%>
```