

# **Les pages JSP**

**Pierre Lefebvre**

# Le pourquoi des jsp

## ■ **Les servlets permettent de :**

- Lire des données de formulaire
- Lire des en-têtes HTTP
- Positionner des en-têtes et un état de réponse HTTP
- Gérer les cookies et les sessions
- Partager des données entre servlets

## ■ **Les servlets ne sont pas faites pour :**

- Générer du code HTML
- Maintenir du code HTML

# Exemple de servlet

```
■ import javax.servlet.*;
■ import javax.servlet.http.*;
■ import java.io.*;
■ import java.util.*;

■ public class MyDearServlet extends HttpServlet {

    • //Process the HTTP GET request
    • public void doGet(HttpServletRequest request,
        • HttpServletResponse response)
        • throws ServletException, IOException {
        •     doPost(request, response);
    • }

    • //Process the HTTP POST request
    • public void doPost(HttpServletRequest request,
        • HttpServletResponse response)
        • throws ServletException, IOException {
        •     response.setContentType("text/html");
        •     PrintWriter out = response.getWriter();
        •     out.println("<HTML>");
        •     out.println("<HEAD><TITLE>Using Servlets</TITLE></HEAD>");
        •     out.println("<BODY BGCOLOR=#123123>");
```

```
■ //Get parameter names
• Enumeration parameters = request.getParameterNames();
• String param = null;
• while (parameters.hasMoreElements()) {
•     param = (String) parameters.nextElement();
•     out.println(param + ":" + request.getParameter(param) +
•         "<BR>");
• }
• out.println("</BODY>");
• out.println("</HTML>");
• out.close();

• } //End of doPost method

• /* other parts of the class goes here
• .
• .
• .
• */
■ } //End of class
```

# Exemple en JSP

```
■ <%@ page import="java.util.Enumeration" %>
■ <HTML>
■ <HEAD><TITLE>Using JSP</TITLE></HEAD>
■ <BODY BGCOLOR=#DADADA>
■ <%
• //Get parameter names
• Enumeration parameters = request.getParameterNames();
• String param = null;
• while (parameters.hasMoreElements()) {
•     param = (String) parameters.nextElement();
•     out.println(param + ":" + request.getParameter(param) +
•         "<BR>");
• }
• out.close();
■ %>
■ </BODY>
■ </HTML>
```

# Séparation du traitement de la requête de la présentation

## *Servlet*

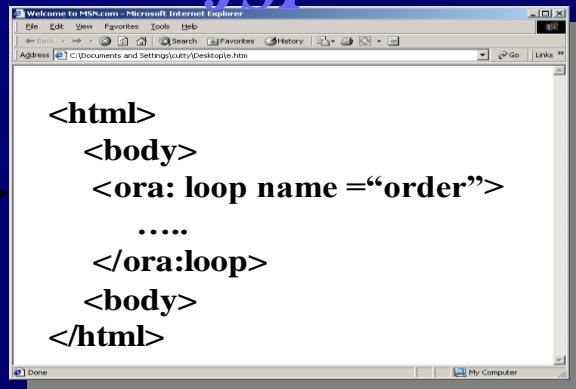
### *Pure Servlet*

```
Public class OrderServlet...{  
    public void doGet(...){  
        if(isOrderValid(req)){  
            saveOrder(req);  
  
            out.println("<html>");  
            out.println("<body>");  
            .....  
        }  
  
        private void isOrderValid(...){  
            .....  
        }  
  
        private void saveOrder(...){  
            .....  
        }  
    }  
}
```

Request processing

```
Public class OrderServlet ...{  
    public void doGet(...){  
        .....  
        if(bean.isOrderValid(..)){  
            bean.saveOrder(...);  
  
            forward("conf.jsp");  
        }  
    }  
}
```

### *JSP*



### *JavaBeans*

Business logic

**isOrderValid( )**

**saveOrder( )**

# Principes des jsp

- Les jsp sont en fait des pages HTML contenant du code Java pouvant effectuer des appels à des JavaBeans
- Les jsp portent l'extension .jsp
- Les jsp sont en fait transformées en servlets par le moteur, compilées à la volée et exécutées
- Les jsp permettent une séparation entre traitement et présentation :
  - Présentation dans la page jsp
  - Traitement dans des JavaBeans

# Fonctionnement du côté du moteur de servlet

- Appel d'une page jsp par le navigateur client
- Le moteur de servlet passe le contrôle au gestionnaire JSP (souvent implémenté sous forme de servlet)
- Vérification de la présence de la page compilée sous forme de servlet
  
- Si la servlet existe, elle est exécutée
- Si la servlet n'existe pas, son code est généré, compilé et exécuté

# Avantages des jsp

- Faciliter l'écriture du code HTML
- Faciliter la maintenance du code HTML
- Donner la possibilité d'utiliser des outils standards tels que Dreamweaver
- Préserver les rôles de chaque personne dans la conception du site web
- Séparer le code java qui crée le contenu du code HTML gérant la présentation
- Précompilation du code

# Premier exemple

## ■ Code JSP

- <HTML>
- <HEAD>
- </HEAD>
- <BODY>
- JSP is easy.
- </BODY>
- </HTML>

## ■ Appel

- <http://localhost:8080/myJSPApp/SimplePage.jsp>

# Eléments de scripts

- Une page JSP peut contenir 4 types d'éléments
  - Scriptlets      `<%              %>`
  - Déclarations    `<%!              %>`
  - Expressions     `<%=              %>`
  - Directives JSP    `<%@              %>`
- Tous ces éléments sont compris entre des balises `<%` et `%>`

Commentaire JSP :  
`<%-- commentaire --%>`

# Les expressions JSP

- Permet d'accéder directement à la valeur d'une variable
  - Syntaxe : <%= *expression* %>
- Exemple : récupération d'un paramètre :
  - <%= request.getParameter(«Price ») %>
  - <%= new java.util.Date() %>
  - <%= request.getRemoteHost() %>
- Syntaxe XML
  - <jsp:expression> Expression java </jsp:expression>

# Les variables prédéfinies

## ■ **request**

- La requête HttpServletRequest

## ■ **response**

- La réponse HttpServletResponse

## ■ **out**

- Le Printwriter utilisée pour envoyer la sortie au client

## ■ **session**

- La session HttpSession associée à la requête

## ■ **application**

- Le ServletContext obtenu par getServletConfig().getContext()

## ■ **config**

- L'objet implicite config représentant ServletConfig

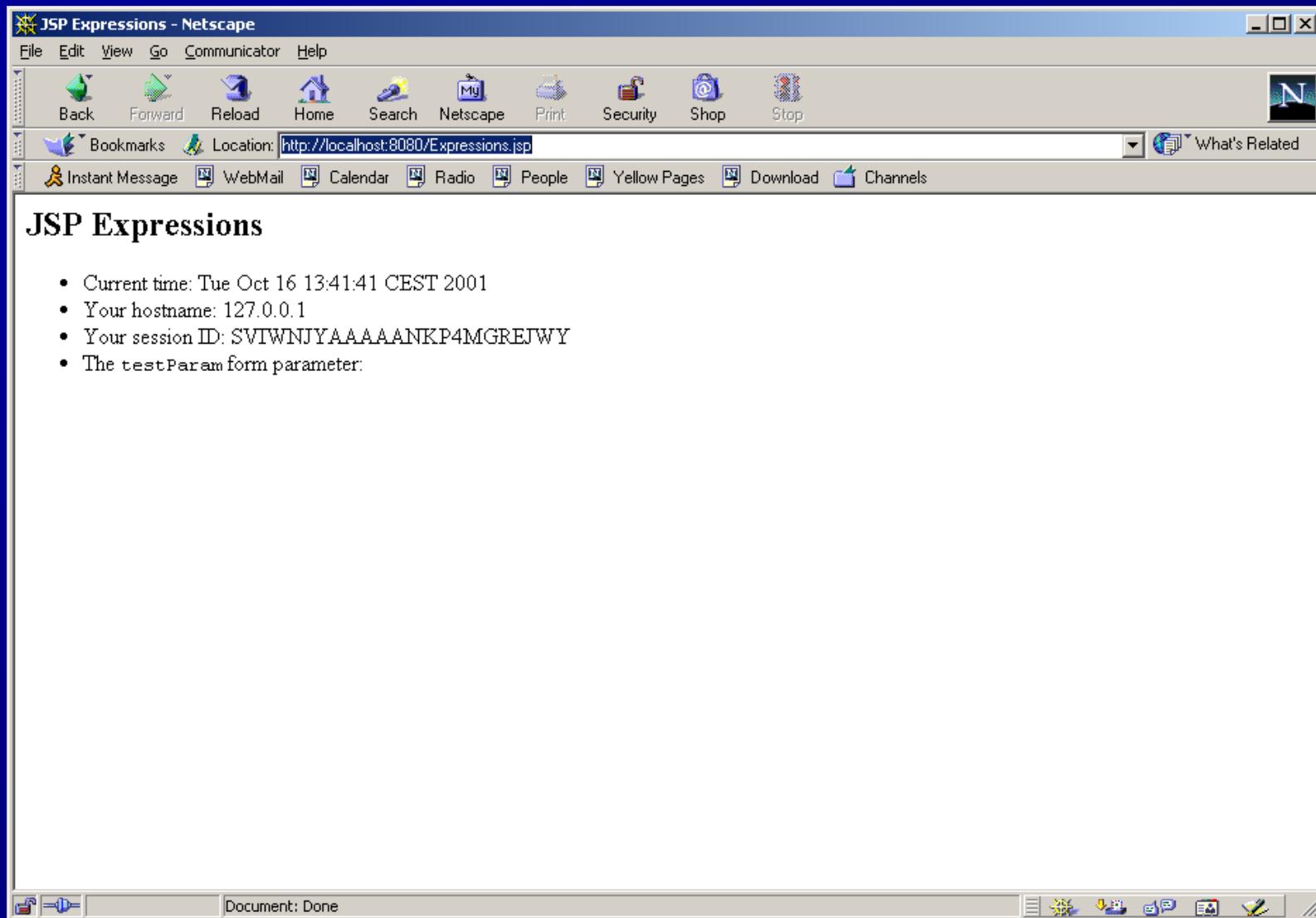
## ■ **exception**

- L'objet exception sur les pages d'erreur

# Exemple utilisant des expressions JSP

```
<HTML>
<HEAD>
<TITLE>JSP Expressions</TITLE>
</HEAD>

<BODY>
<H2>JSP Expressions</H2>
<UL>
    <LI>Current time: <%= new java.util.Date() %>
    <LI>Your hostname: <%= request.getRemoteHost() %>
    <LI>Your session ID: <%= session.getId() %>
    <LI>The <CODE>testParam</CODE> form parameter:
        <%= request.getParameter("testParam") %>
</UL>
</BODY>
</HTML>
```



# Les scriplets java

- Insertion directe de code Java dans la page
  - Syntaxe : <% .... **code java** ..... %>
- Le code java est placé dans une méthode `_jspService( )` de la servlet générée
- Quelques variables prédéfinies accessibles :
  - *request*
    - La requête de la Servlet, un objet `HttpServletRequest`
  - *response*
    - La réponse de la servlet, un objet `HttpServletResponse`
  - *out*
    - Buffer de sortie, un objet `PrintWriter`
- Syntaxe XML
  - `<jsp:scriptlet> code java </jsp:scriptlet>`

# Une page JSP « hello world »

## ■ Exemple hello.jsp :

```
<HTML>
<HEAD><TITLE>Hello</HEAD>
<BODY>
<%
if request.getParameter("name") == null) {
    out.println("hello,world");
}
else {
    out.println("hello, " + request.getParameter("name"));
}
%>
</BODY>
</HTML>
```

Appel:

[http://my\\_server/servlet/hello.jsp](http://my_server/servlet/hello.jsp)  
[http://my\\_server/servlet/hello.jsp?name=toto](http://my_server/servlet/hello.jsp?name=toto)

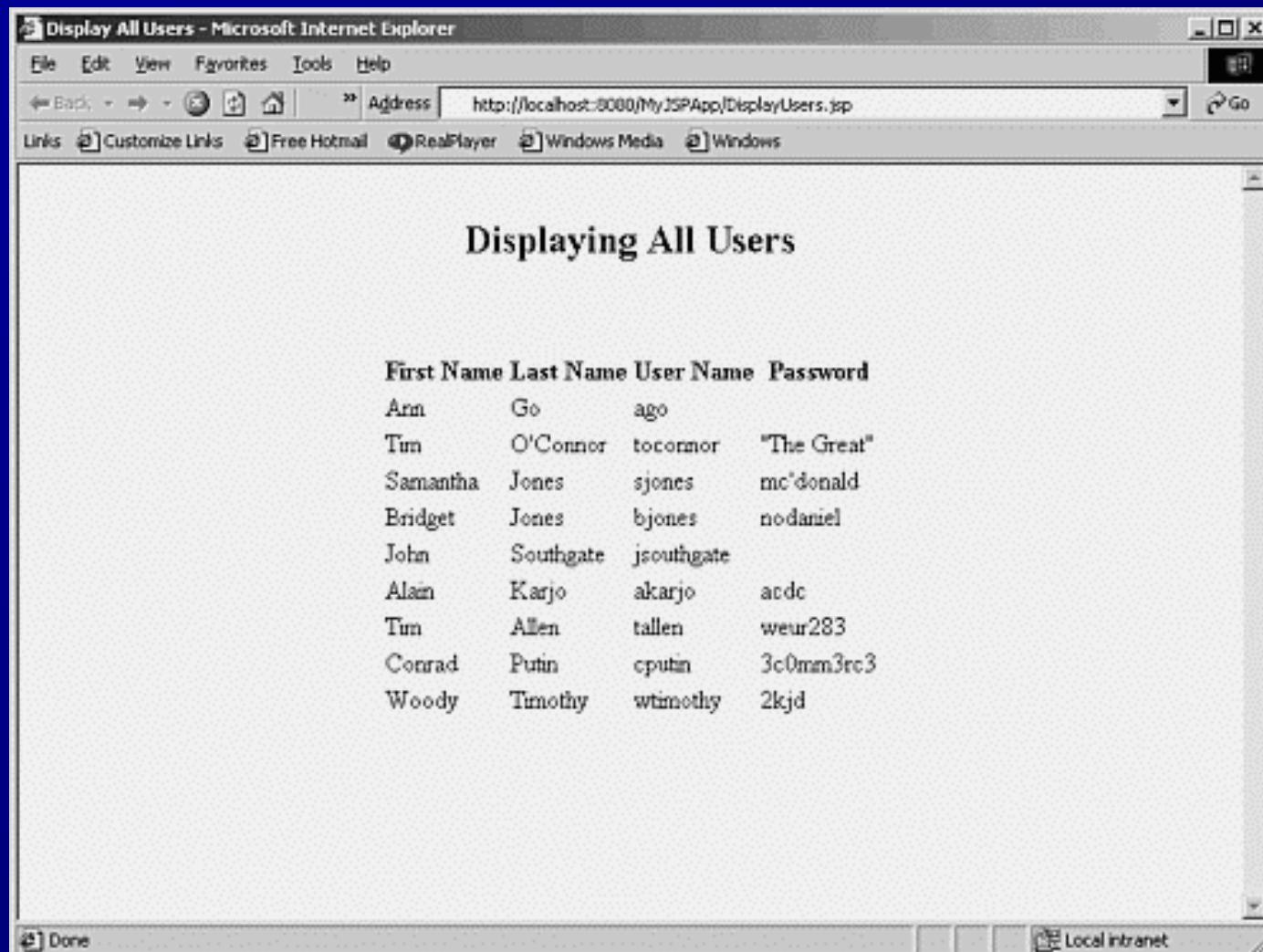
# Le code généré par le serveur

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class _hello_xjsp extends HttpServlet {
    public void service (HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    BufferedReader in = request.getReader();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello</HEAD>");
    out.println("<BODY>");
    if request.getParameter("name") == null) {
        out.println(" hello,world ");
    }
    else {
        out.println(" hello, " + request.getParameter("name"));
    }
    out.println("</BODY>");
    out.println("</HTML>");
```

# Utilisation de scriplets pour insérer du code HTML

- Les scriplets sont insérées dans la servlet telles qu'elles sont écrites
- Maintenance plus difficile malgré tout
- Exemple
  - <% if(Math.random() < 0.5) { %>
  - Have a <B>nice</B! day!
  - <% } else { %>
  - Have a <B>lousy</B> day!
  - < } %>
- Résultat
  - if (Math.random() < 0.5) {
  - out.println(« Have a <B>nice day! »);
  - else out.println(« Have a lousy<B> day! »);
  - }

# Exemple d'accès à une base de donnée via une JSP



```
<%@ page session="false" %>
<%@ page import="java.sql.*" %>
<%
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    System.out.println("JDBC driver loaded");
}
catch (ClassNotFoundException e) {
    System.out.println(e.toString());
}
%>
<HTML>
<HEAD>
<TITLE>Display All Users</TITLE>
</HEAD>
<BODY>
<CENTER>
<BR><H2>Displaying All Users</H2>
<BR>
<BR>
<TABLE>
<TR>
<TH>First Name</TH>
<TH>Last Name</TH>
<TH>User Name</TH>
<TH>Password</TH>
</TR>
```

```
■ <%
• String sql = "SELECT FirstName, LastName, UserName, Password" +
  " FROM Users";
• try {
•   Connection con = DriverManager.getConnection("jdbc:odbc:JavaWeb");
•
•   Statement s = con.createStatement();
•   ResultSet rs = s.executeQuery(sql);
•
•   while (rs.next()) {
•     out.println("<TR>");
•     out.println("<TD>" + rs.getString(1) + "</TD>");
•     out.println("<TD>" + rs.getString(2) + "</TD>");
•     out.println("<TD>" + rs.getString(3) + "</TD>");
•     out.println("<TD>" + rs.getString(4) + "</TD>");
•     out.println("</TR>");
•   }
•   rs.close();
•   s.close();
•   con.close();
• }
• catch (SQLException e) {
• }
• catch (Exception e) {
• }
■ %>
■ </TABLE>
■ </CENTER>
■ </BODY>
■ </HTML>
```

# Les déclarations

■ Permettent de définir des méthodes et des variables dans la page jsp

■ **Syntaxe :**

- `<%! Declaration %>`

■ **Exemple :**

- `<%! char c = 0; %>`
- `<%! private void uneMethode(...) {...}`

■ **Syntaxe XML**

- `<jsp:declaration> code java </jsp:declaration>`

# Correspondance JSP/Servlet

## ■ Code JSP

- <H1> Some Heading </H1>
- <%!
- public String randomHeading() {
- return(« <H2> » + Math.random()
- + « </H2> »);
- %>
- <%= randomHeading() %>

# Code servlet

## ■ Code de la servlet

```
– public class xxxx extends HttpServlet {  
–     public String randomHeading() {  
–         return(« <H2> + Math.random() + «  
–                 </H2> »);  
–     }  
–     public void _jspService(HttpServletRequest request,  
–                             HttpServletResponse response) throws ServletException, IOException {  
–         request.setContenetType(« text/html »);  
–         HttpSession session = request.getSession(true);  
–         JspWriter out = response.getWriter();  
–         out.println(<H1> Some Heading </H1>);  
–         out.println(randomHeading());  
–         ...  
–     }  
– }
```

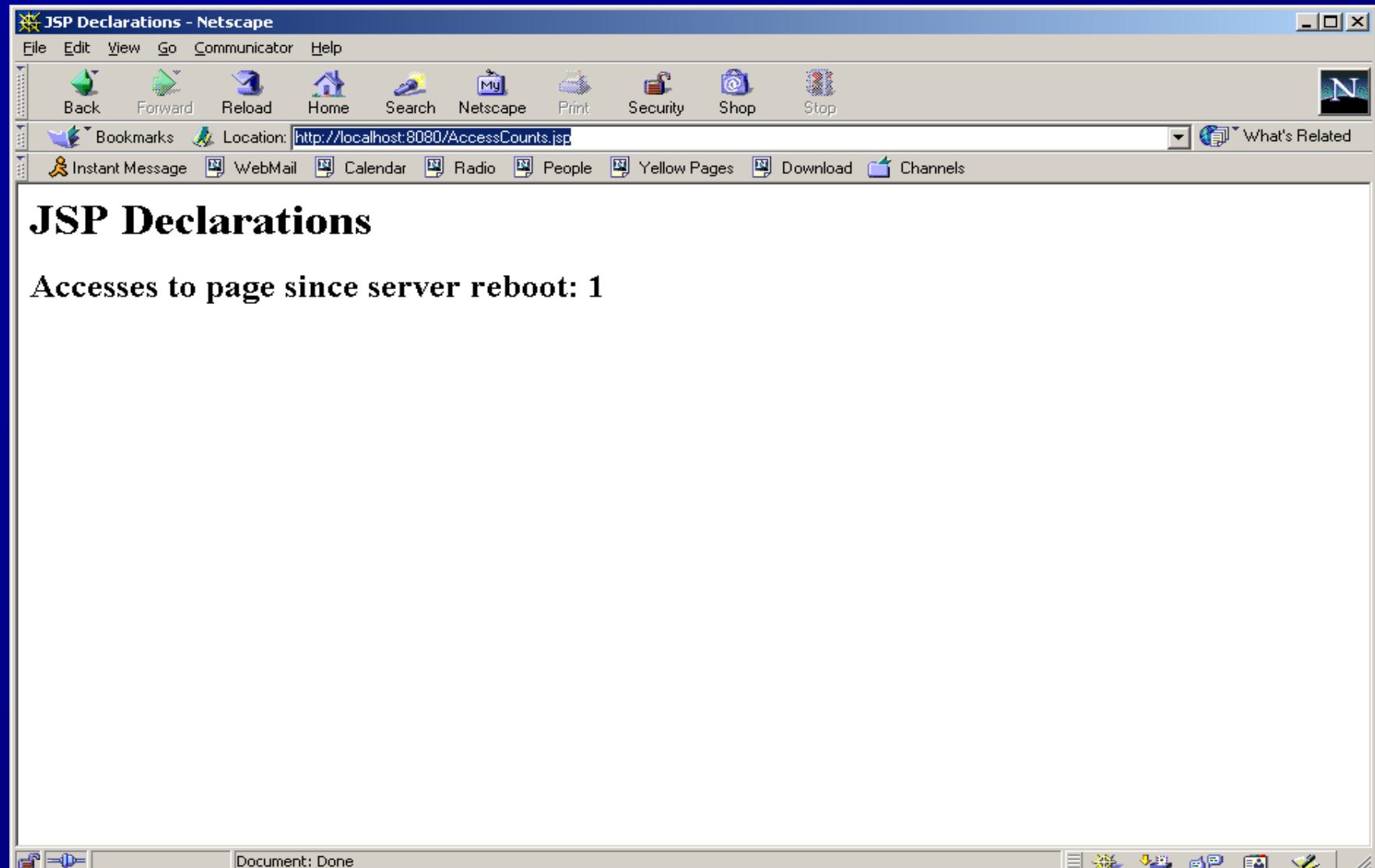
# Exemples d'utilisation des déclarations

```
<HTML>
<HEAD>
<TITLE>JSP Declarations</TITLE>
</HEAD>
```

```
<BODY>
<H1>JSP Declarations</H1>
```

```
<%! private int accessCount = 0; %>
<H2>Accesses to page since server reboot:
<%= ++accessCount %></H2>
```

```
</BODY>
</HTML>
```



# Types de directive JSP

## ■ Trois types :

- **page**

- Contrôler la structure de la servlet

- Importer des classes
  - Personnaliser la super-classe de la servlet
  - Définir le type du contenu

- **include**

- Insérer un fichier dans la classe de la servlet

- **taglib**

- Définir des étiquette personnalisées

## ■ Syntaxe

- `<%@ page attribute1="value1" attribute2="value2" ... %>`

# La directive page JSP

## ■ Permet d 'indiquer :

- Le langage de script utilisé
- Les interfaces qu'une servlet implémente
- La classe qu'une servlet étend
- Les packages qu'une servlet importe
- etc...

✂ Syntaxe : <%@ page *nom\_attribut=valeur* %>

✂ Exemple : <%@ page *import = java.io.\** %>

✂ Les différents attributs sont : *content\_type, import, extends, implements, method, language, isThreadSafe, session, buffer, autoflush, information, errorPage, isErrorPage*

# L'attribut page

## ■ Exemples correctes

- <%@ page buffer="16384" %>
- <%@ page session="false" %>
- <%@ page import="java.io.\*,  
java.util.Enumeration" %>

## ■ Exemples incorrectes

- <%@ page info="Example Page" %>
- <%@ page buffer="16384" %>
- <%@ page info="Unrestricted Access" %>

# L'attribut info et language

## ■ L'attribut language

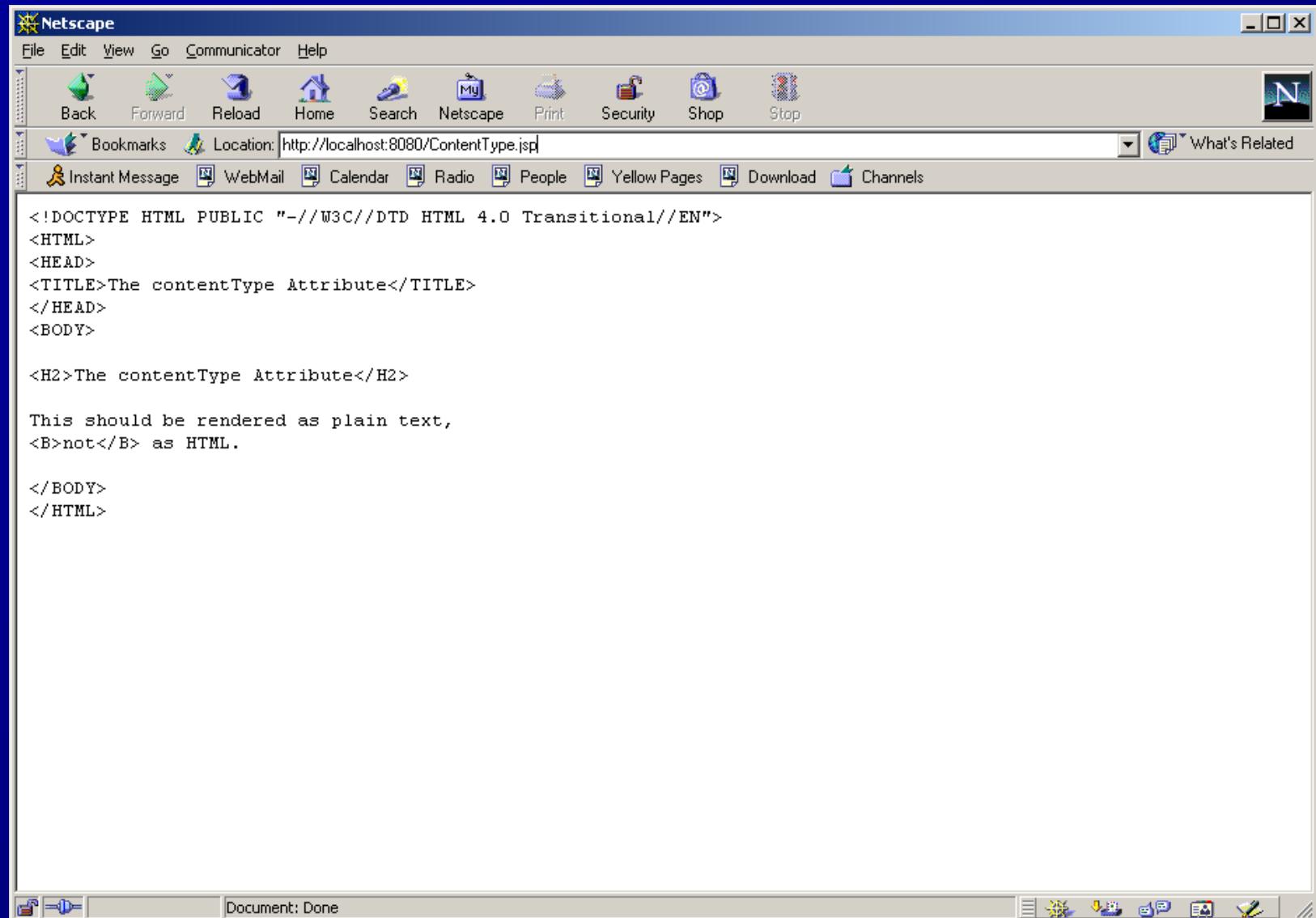
- <%@ page language="java" %>
- <%
- out.println("JSP is easy");
- %>

## ■ L'attribut info

- <%@ page info="Written by Bulbul" %>
- <%
- out.println(getServletInfo());
- %>

# L'attribut contentType

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0  
Transitional//EN">  
<HTML>  
<HEAD>  
<TITLE>The contentType Attribute</TITLE>  
</HEAD>  
<BODY>  
  
<H2>The contentType Attribute</H2>  
<%@ page contentType="text/plain" %>  
This should be rendered as plain text,  
<B>not</B> as HTML.  
  
</BODY>  
</HTML>
```



# Autres attributs

## ■ Attribut session

- <@ page session = « true » %> <%-- Default --%>
- <@ page session = « false » %>

## ■ Attribut buffer

- <@ page buffer = « sizekb » %>
- <@ page buffer = « none » %>

## ■ Attribut autoflush

- <@ page autoflush = « true » %> <%-- Default --%>
- <@ page autoflush = « false » %>

## ■ Attribut extends

- <@ page extends=« package.class » %>

## ■ Attribut info

- <@ page info=« un message » %>

## ■ Attribut errorPage

- <@ page errorPage=« URL » %>

## ■ Attribut isErrorPage

- <@ page isErrorPage = « false » %> <%-- Default --%>
- <@ page isErrorPage = « true » %>

## ■ Attribut language

- <@ page language=« java » %>

# Inclure des fichiers au moment de la traduction de la page

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

<%-- The following become fields in each servlet that
    results from a JSP page that includes this file. --%>
<%!
private int accessCount = 0;
private Date accessDate = new Date();
private String accessHost = "<I>No previous access</I>";
%>

<P>
<HR>
This page &copy; 2000
<A HREF="http//www.my-company.com/">my-company.com</A>.
This page has been accessed <%= ++accessCount %>
times since server reboot. It was last accessed from
<%= accessHost %> at <%= accessDate %>.

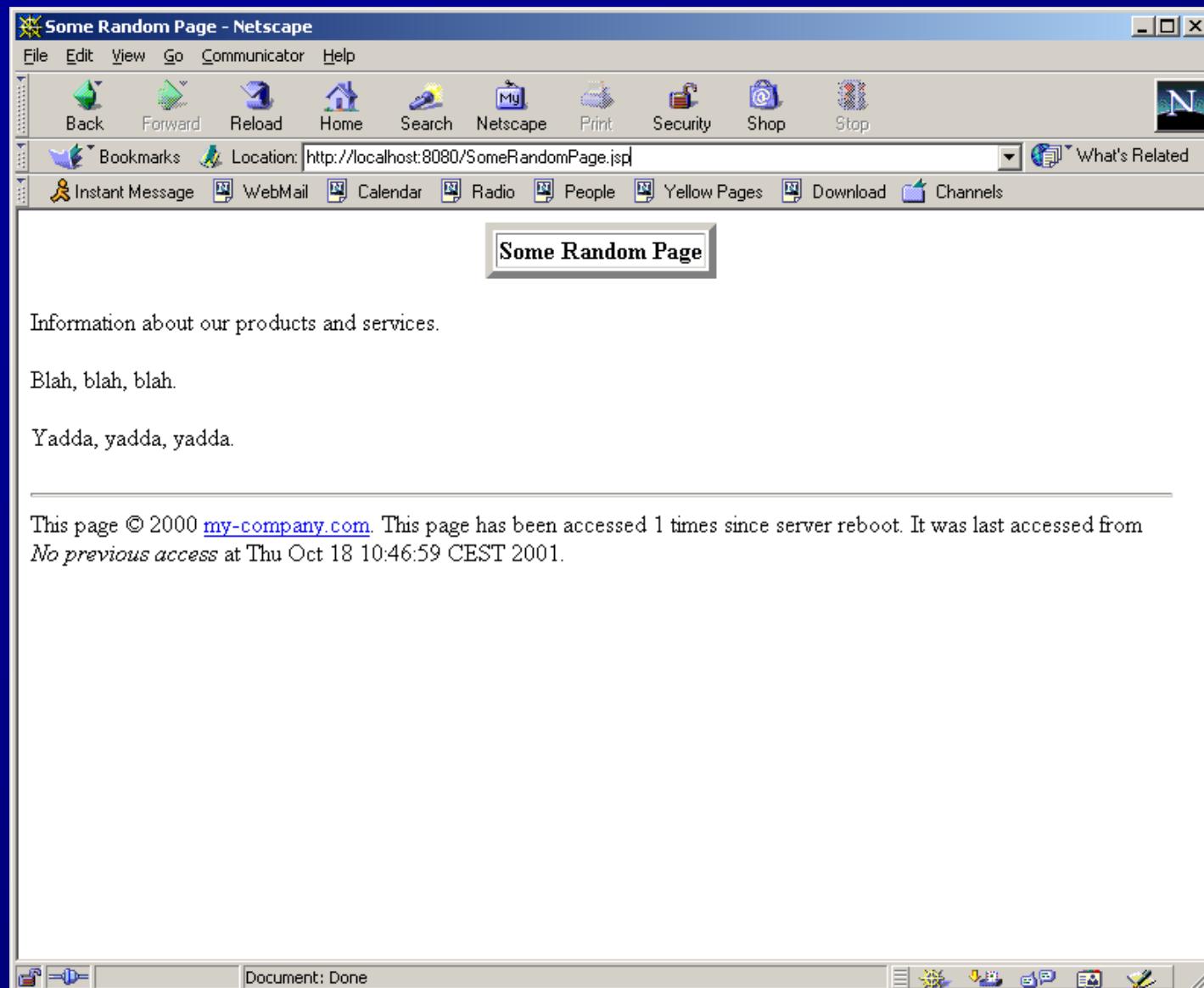
<% accessHost = request.getRemoteHost(); %>
<% accessDate = new Date(); %>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>Some Random Page</TITLE>
</HEAD>

<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">
    Some Random Page</TABLE>
<P>
Information about our products and services.
<P>
Blah, blah, blah.
<P>
Yadda, yadda, yadda.

<%@ include file="ContactSection.jsp" %>

</BODY>
</HTML>
```



# Gestion d'erreur

```
<HTML>
<HEAD>
<TITLE>Computing Speed</TITLE>
</HEAD>

<BODY>

<%@ page errorPage="SpeedErrors.jsp" %>

<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">
    Computing Speed</TABLE>

<%!
// Note lack of try/catch for NumberFormatException if
// value is null or malformed.

private double toDouble(String value) {
    return(Double.valueOf(value).doubleValue());
}
%>

<%
double furlongs = toDouble(request.getParameter("furlongs"));
double fortnights = toDouble(request.getParameter("fortnights"));
double speed = furlongs/fortnights;
%>

<UL>
    <LI>Distance: <%= furlongs %> furlongs.
    <LI>Time: <%= fortnights %> fortnights.
    <LI>Speed: <%= speed %> furlongs per fortnight.
</UL>

</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Error Computing Speed</TITLE>
</HEAD>

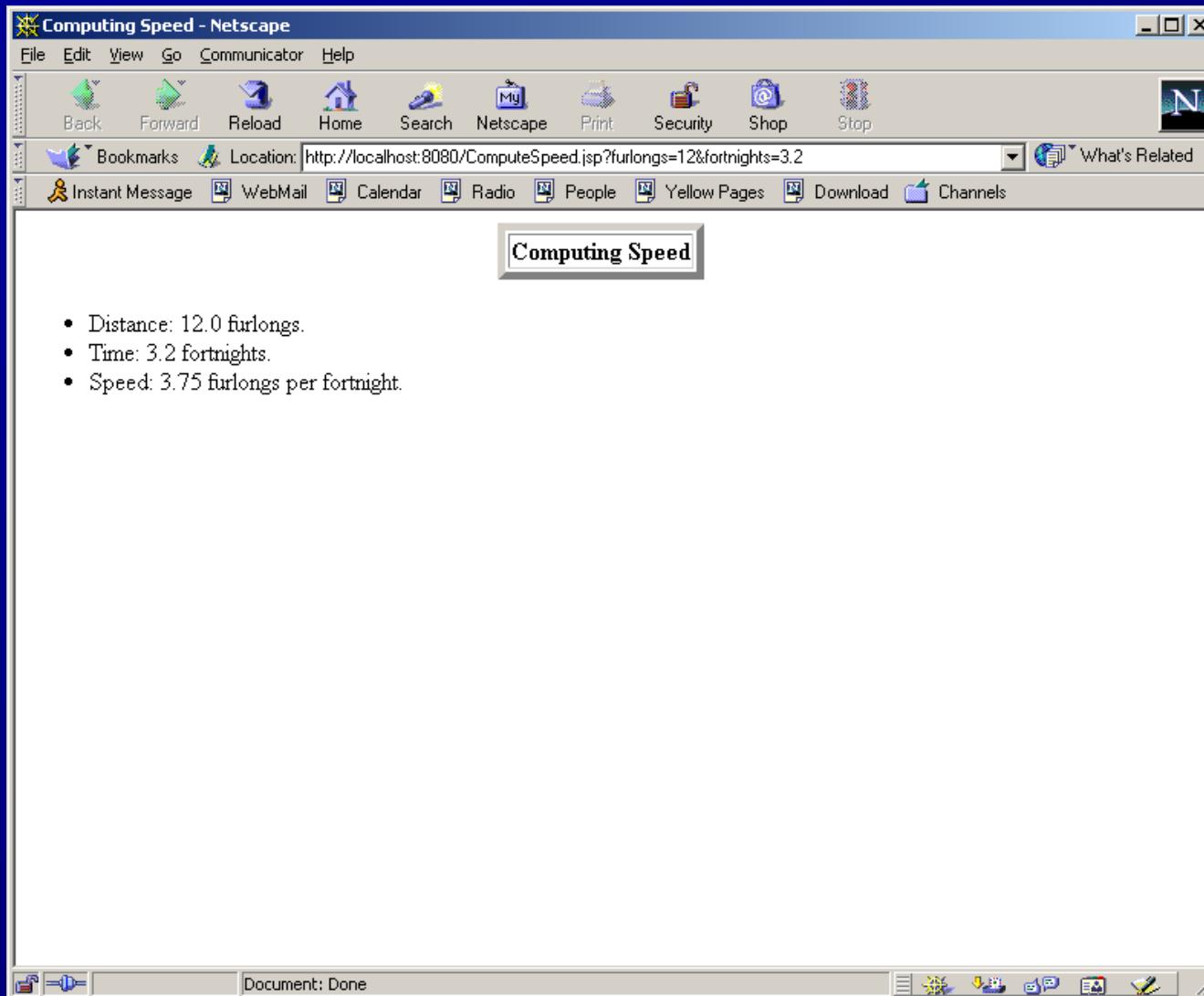
<BODY>

<% @ page isErrorPage="true" %>

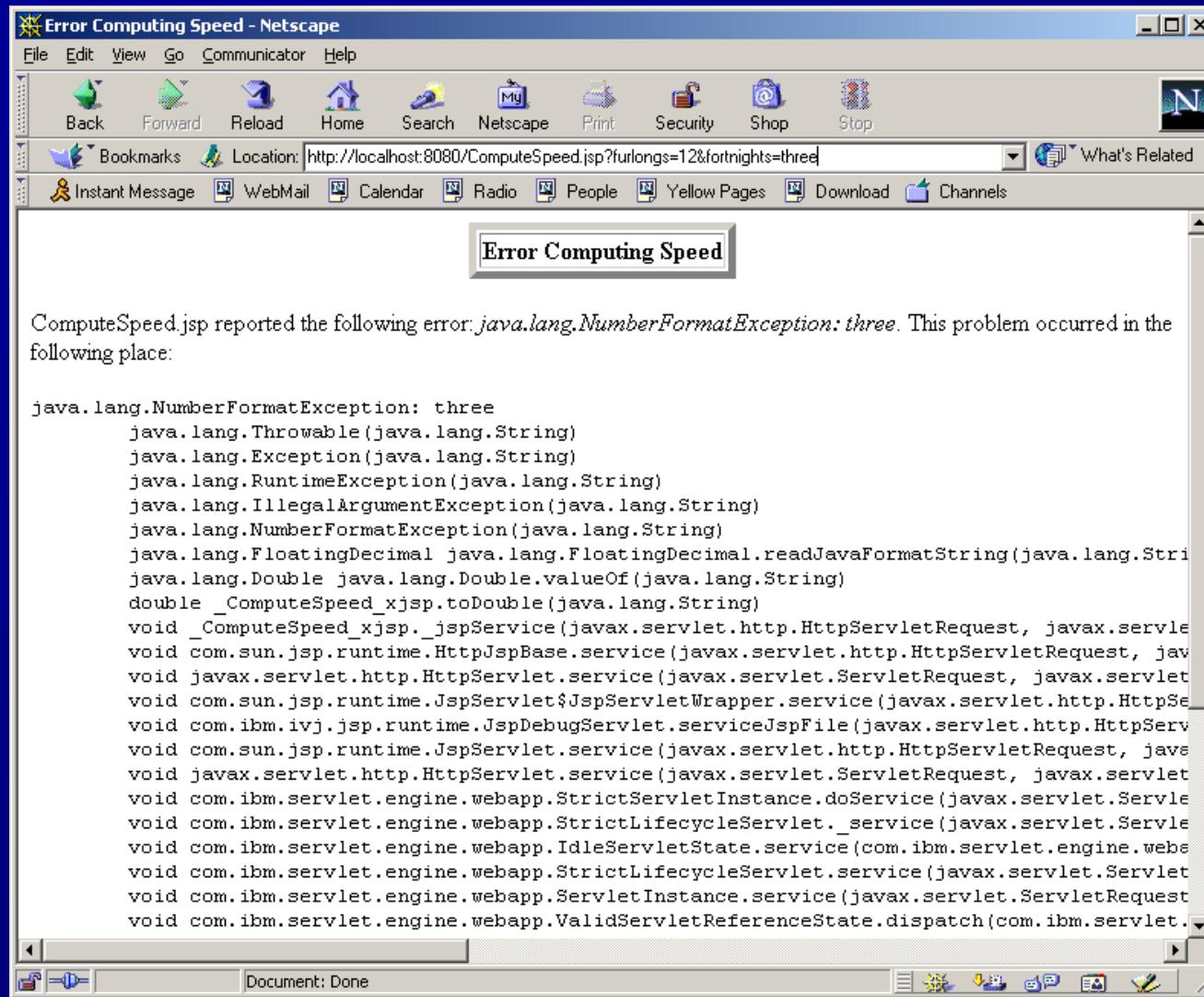
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">
    Error Computing Speed</TABLE>
<P>
ComputeSpeed.jsp reported the following error:
<I><%= exception %></I>. This problem occurred in the
following place:
<PRE>
<% exception.printStackTrace(new PrintWriter(out)); %>
</PRE>

</BODY>
</HTML>
```

# Exemple



# Exemple



# Les éléments d'actions standard

- jsp:useBean
- jsp:setProperty
- jsp:getProperty
- jsp:param
- jsp:include
- jsp:forward
- jsp:plugin
- jsp:params
- jsp:fallback

# Inclure des fichiers au moment de la requête

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>What's New</TITLE>
</HEAD>

<BODY>

<CENTER>
<TABLE BORDER=5>
<TR><TH CLASS="TITLE">
    What's New at JspNews.com</TABLE>
</CENTER>
<P>
```

Here is a summary of our four most recent news stories:

```
<OL>
<LI><jsp:include page="news/Item1.html" flush="true" />
<LI><jsp:include page="news/Item2.html" flush="true" />
<LI><jsp:include page="news/Item3.html" flush="true" />
<LI><jsp:include page="news/Item4.html" flush="true" />
</OL>
</BODY>
</HTML>
```

# Fichiers HTML à include

## Item1.html

<B>Bill Gates acts humble.</B> In a startling and unexpected development, Microsoft big wig Bill Gates put on an open act of humility yesterday.  
<A HREF="http://www.microsoft.com/Never.html">More details...</A>

## Item2.html

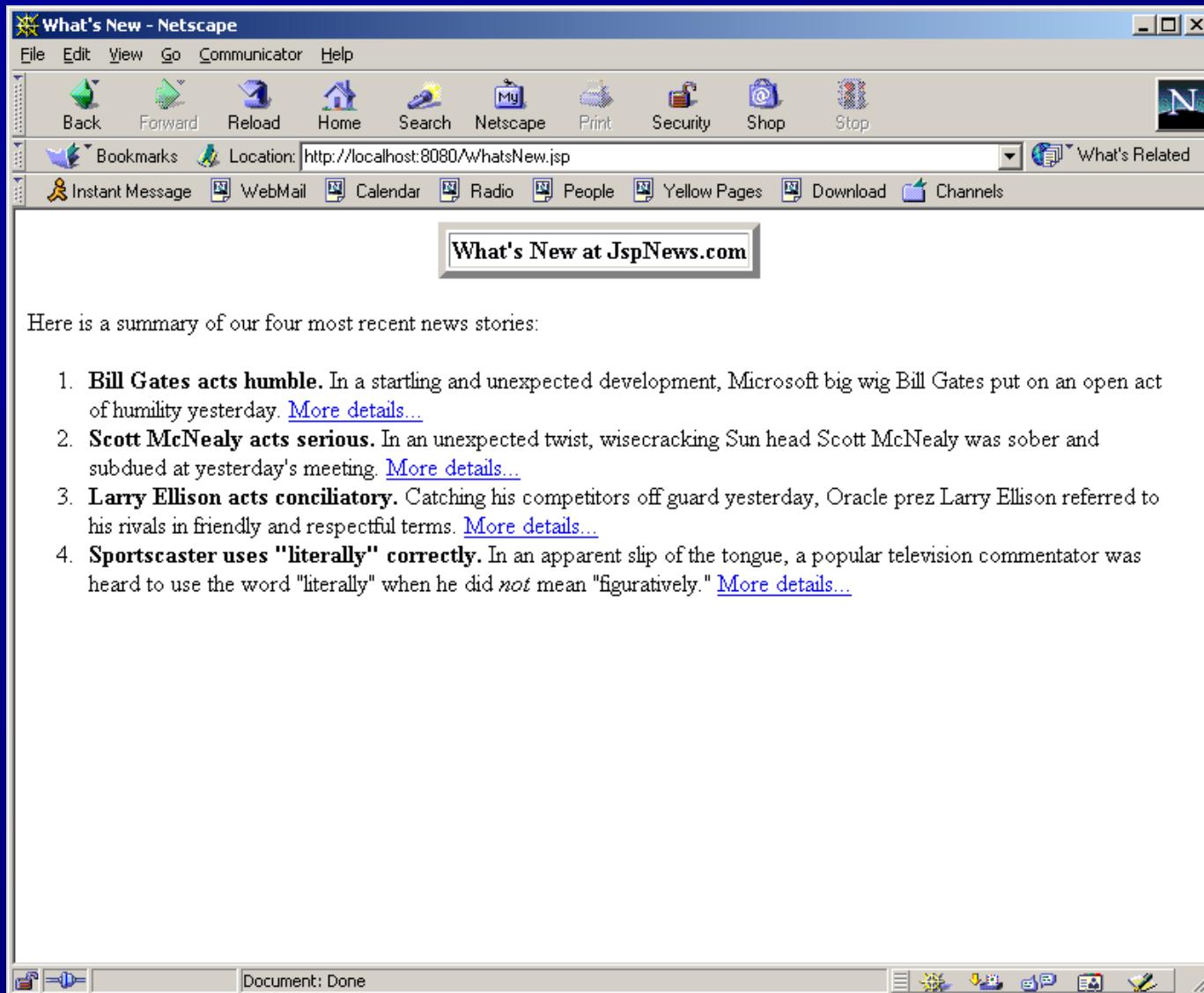
<B>Scott McNealy acts serious.</B> In an unexpected twist, wisecracking Sun head Scott McNealy was sober and subdued at yesterday's meeting.  
<A HREF="http://www.sun.com/Imposter.html">More details...</A>

## Item3.html

<B>Larry Ellison acts conciliatory.</B> Catching his competitors off guard yesterday, Oracle prez Larry Ellison referred to his rivals in friendly and respectful terms.  
<A HREF="http://www.oracle.com/Mistake.html">More details...</A>

## Item4.html

<B>Sportscaster uses "literally" correctly.</B> In an apparent slip of the tongue, a popular television commentator was heard to use the word "literally" when he did <I>not</I> mean "figuratively."  
<A HREF="http://www.espn.com/Slip.html">More details...</A>



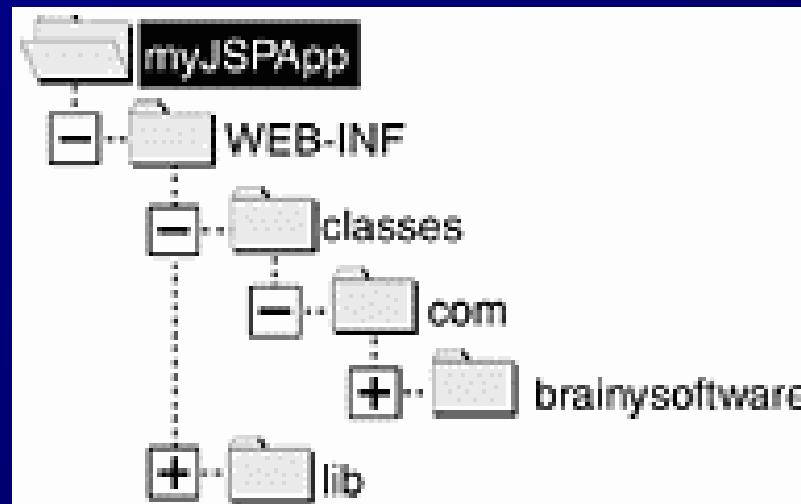
# Les javabeans et JSP

# Javabean : définition

- Composants java réutilisables (classes)
- Pas de champs publics !
- Méthodes et variables conformes à une convention de nommage (getXxx, setXxx, isXxx)
- Constructeur sans argument
- Permettent l'échange de données entre les Servlets applicatives et les JSP de présentation

# Exemple de bean

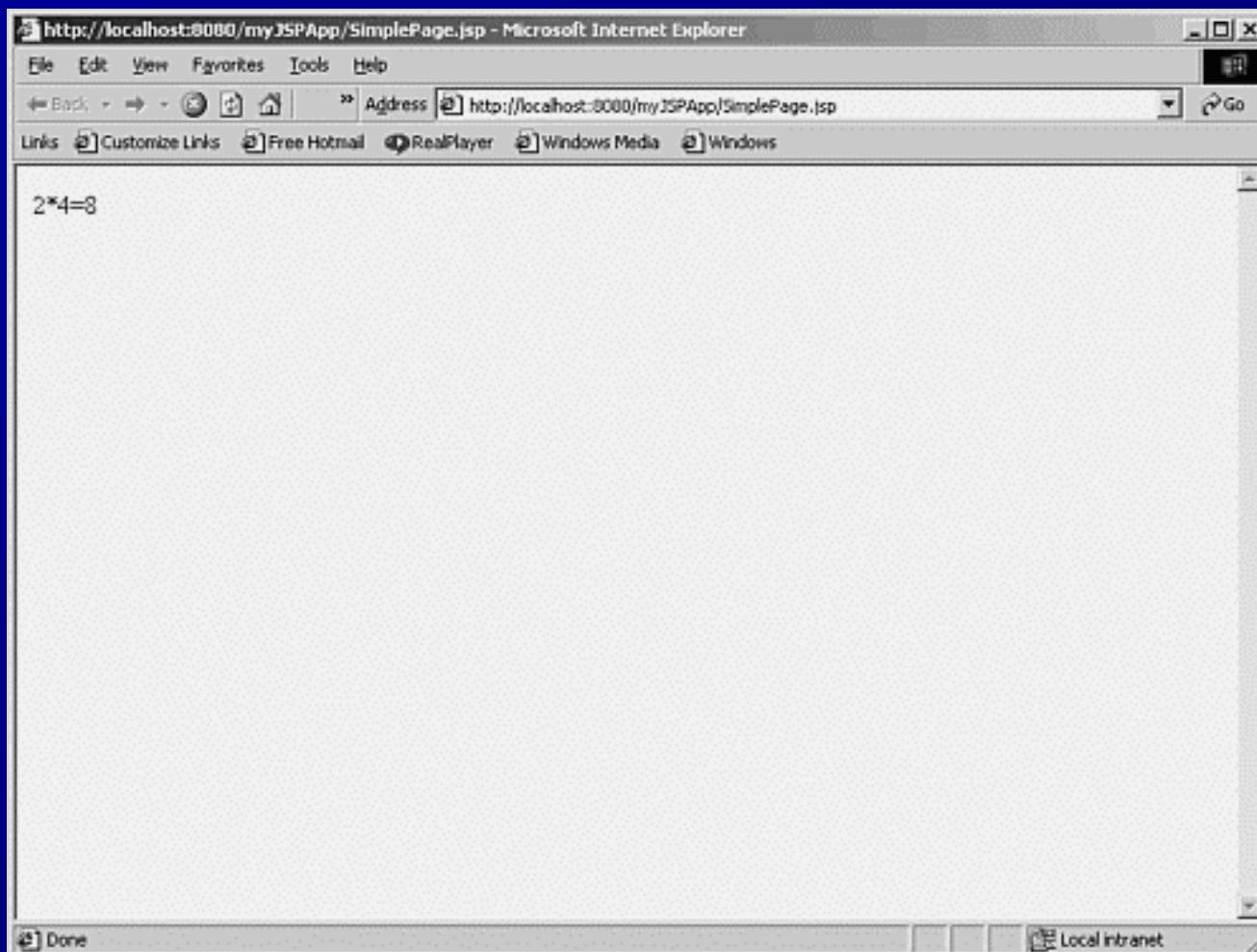
```
■ package com.brainysoftware;  
■ public class CalculatorBean {  
•   public int doubleIt(int number) {  
•       return 2 * number;  
•   }  
■ }
```



# Page JSP d'appel au bean

```
■ <jsp:useBean id="theBean"  
■   class="com.brainysoftware.CalculatorBean"/>  
■ <HTML>  
■ <HEAD>  
■ </HEAD>  
■ <BODY>  
■ <%  
•   int i = 4;  
•   int j = theBean.doubleIt(i);  
•   out.print("2*4=" + j);  
■ %>  
■ </BODY>  
■ </HTML>
```

# Résultat



# Syntaxe

## ■ Format

- <jsp:useBean (attribute="value")+>
- <jsp:useBean (attribute="value")+>  
    initialization code </jsp:useBean>

## ■ Attributs

- id
- class
- scope (page, request, session, application)

# Portée requête

## ■ Exemple 1

- <jsp:usebean id=« personne » class=« testjsp.Personne » scope=« request » />

## ■ Equivalence

- *<% testjsp.Personne personne = (testjsp.Personne)  
request.getAttribute(« personne »);*
- *If(personne == null)*
- *{*
- *personne = new testjsp.Personne();*
- *request.setAttribute(« personne », personne);*
- *}*
- *%>*

# Portée session

## ■ Exemple 2

- *<jsp:useBean id=« personne » class=« testjsp.Personne » scope=« session »*

## ■ Equivalence

- *<testjsp.Personne personne = (testjsp.Personne)  
session.getAttribute(« personne »);*
- *If(personne == null) {*
- *personne = new testjsp.Personne();*
- *session.setAttribute(« personne », personne);*
- }

# Portée application

## ■ Exemple 3

- *<jsp:useBean id=« personne » class=« testjsp.Personne » scope=« application » />*

## ■ Equivalence

- *If(personne == null)*
- {
- *personne = new testjsp.Personne();*
- *application.setAttribute(« personne », personne);*
- }
- %>

# Portée page

## ■ Exemple 4

- *<jsp:usebean id=« personne » class=« testjsp.Personne », scope=« page » />*

## ■ Equivalence

- *<%testjsp.Personne personne = new testjsp.Personne(); %>*

# Positionner les propriétés d'un bean

- <jsp:setProperty name="Bean Name" property="PropertyName" value="value"/>
- <jsp:setProperty name="Bean Name" property="PropertyName"/>
- <jsp:setProperty name="Bean Name" property="PropertyName" param="parameterName"/>
- <jsp:setProperty name="Bean Name" property="\*"/>

# Exemple

```
■ package com.brainysoftware;
■ public class CalculatorBean {
•     private int memory;

•     public void setMemory(int number) {
•         memory = number;
•     }

•     public int getMemory() {
•         return memory;
•     }
•     public int doubleIt(int number) {
•         return 2 * number;
•     }
■ }
```

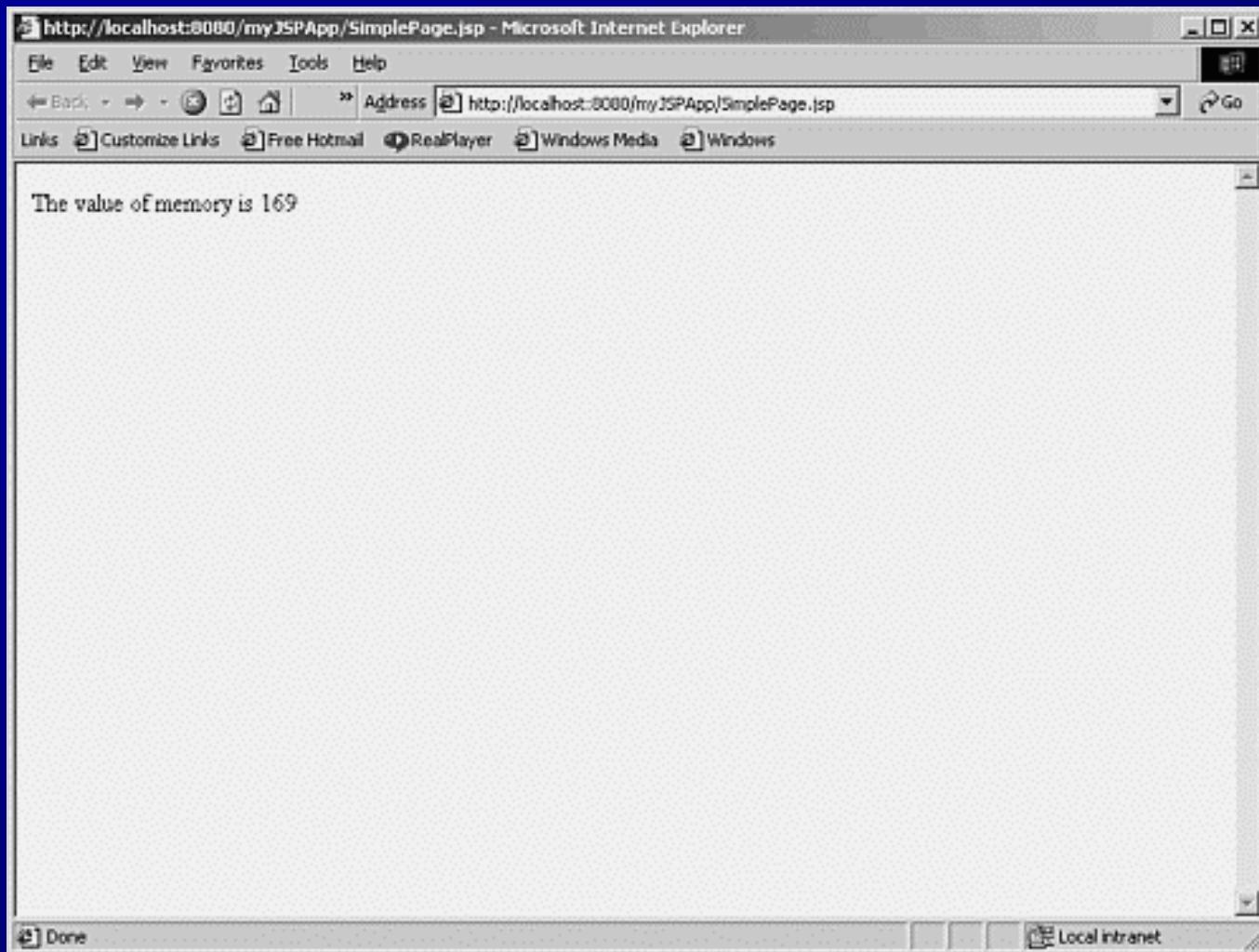
# Page JSP

- <jsp:useBean id="theBean" class="com.brainysoftware.CalculatorBean"/>
- <jsp:setProperty name="theBean" property="memory" value="169"/>
- The value of memory is <jsp:getProperty name="theBean" property="memory"/>
- -----
- <jsp:useBean id="theBean" class="com.brainysoftware.CalculatorBean"/>
- <%
- theBean.setMemory(987);
- %>
- The value of memory is <%= theBean.getMemory()%>

# Utilisation d'un attribut dans une page JSP

- <jsp:useBean id="theBean" class="com.brainysoftware.CalculatorBean"/>
- <%
  - int i = 2;%>
- <jsp:setProperty name="theBean" property="memory" value="<% = 100 \* i %>"/>
- The value of memory is <jsp:getProperty name="theBean" property="memory"/>

# Résultat



# Gestion d'un formulaire et d'un bean

```
■ <HTML>
■ <HEAD>
■ <TITLE>Passing a value</TITLE>
■ </HEAD>
■ <BODY>
■ <CENTER>
■ Please type in a number in the box
■ <BR>
■ <FORM METHOD=POST ACTION=SimplePage.jsp>
■ <INPUT TYPE=TEXT NAME=memory>
■ <INPUT TYPE=SUBMIT>
■ </FORM>
■ </CENTER>
■ </BODY>
■ </HTML>
```

# Page JSP

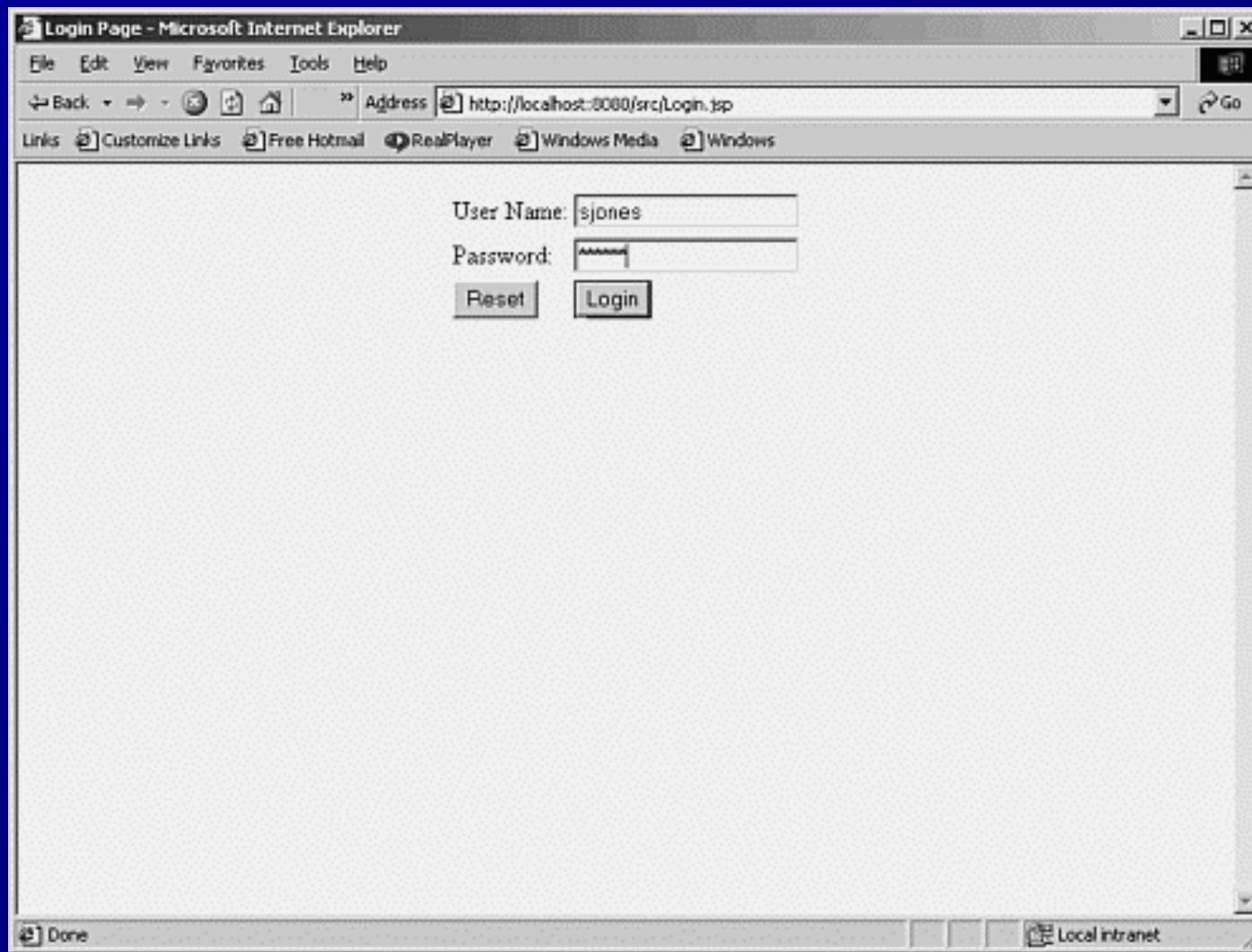
- <jsp:useBean id="theBean" class="com.brainysoftware.CalculatorBean"/>
- <jsp:setProperty name="theBean" property="memory" param="memory"/>
- The value of memory is <jsp:getProperty name="theBean" property="memory"/>

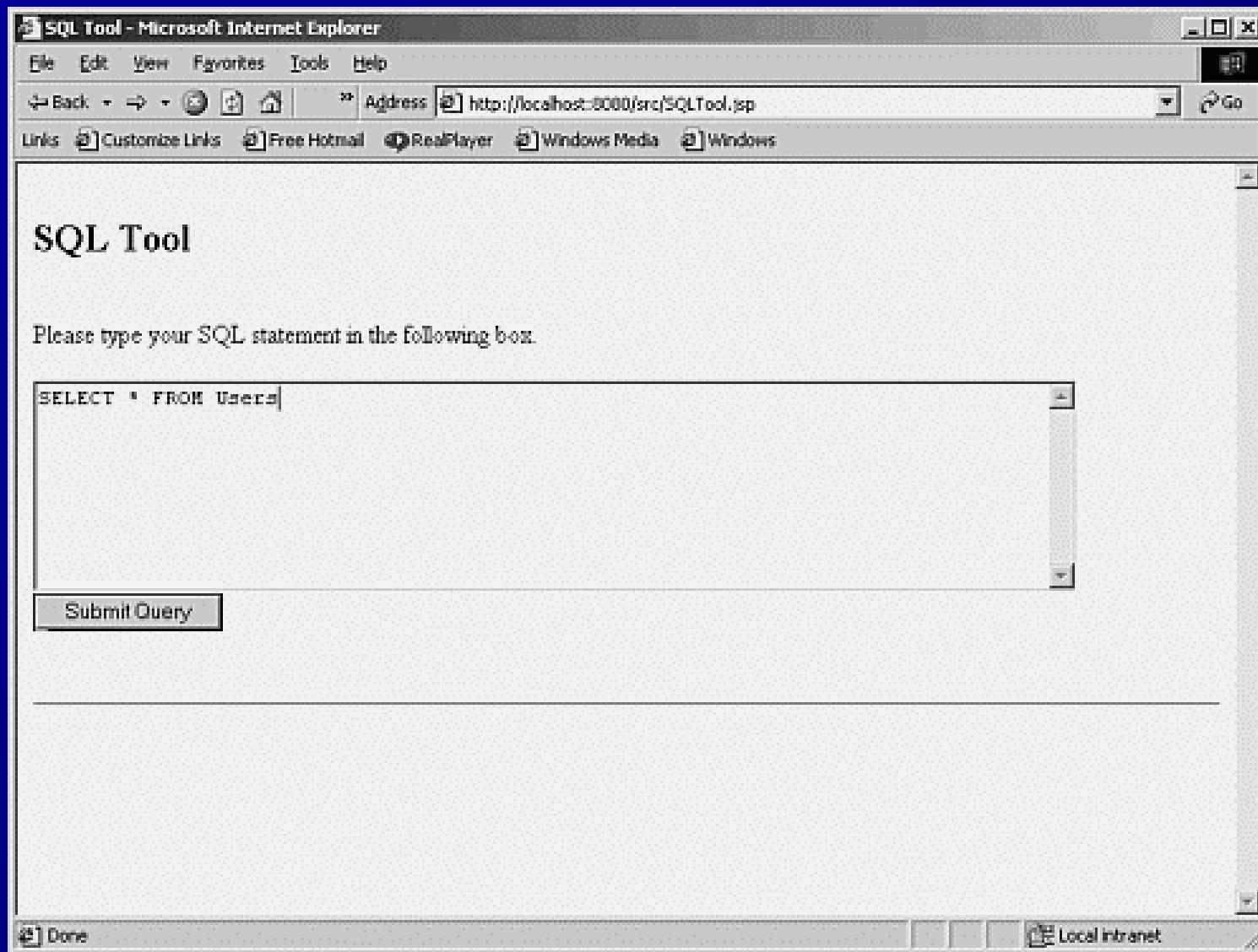
# Partager les beans

## ■ Quatre emplacements possibles

- page
  - Valeur par défaut
  - Le bean est placé dans l'objet PageContext pendant la requête actuelle
- application
  - Le bean sera enregistré dans le ServletContext
- session
  - Le bean est enregistré dans l'objet HttpSession
- request
  - Le bean est placé dans l'objet HttpServletRequest

# JSP/Bean/JDBC





sql Tool - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Address  Go

Links [Customize Links](#) [Free Hotmail](#) [RealPlayer](#) [Windows Media](#) [Windows](#)

## SQL Tool

Please type your SQL statement in the following box.

```
SELECT * FROM Users
```

---

<b>Id</b>	<b>FirstName</b>	<b>LastName</b>	<b>UserName</b>	<b>Password</b>
2	Ann	Go	ago	 
3	Tim	O'Connor	toconnor	"The Great"
6	Samantha	Jones	sjones	mc'donald

Done Local intranet

```
■ import java.sql.*;
■ import com.brainysoftware.java.StringUtil;

■ public class SQLToolBean {
    private String sql = "";
    private String userName = "";
    private String password = "";
    private String connectionUrl;

    • public String getSql() {
        return StringUtil.encodeHtmlITag(sql);
    }

    • public void setSql(String sql) {
        if (sql!=null)
            this.sql = sql;
    }

    • public void setUserName(String userName) {
        if (userName!=null)
            this.userName = userName;
    }
    public String getUserName() {
        return StringUtil.encodeHtmlITag(userName);
    }
    public void setPassword(String password) {
        if (password!=null)
            this.password = password;
    }
    public String getPassword() {
        return StringUtil.encodeHtmlITag(password);
    }
}
```

```
■ public void setConnectionUrl(String url) {  
•     connectionUrl = url;  
• }  
■ public String getResult() {  
•     if (sql==null || sql.equals(""))  
•         return "";  
•     StringBuffer result = new StringBuffer(1024);  
•     try {  
•         Connection con = DriverManager.getConnection(connectionUrl, userName,  
■ password);  
•         Statement s = con.createStatement();  
•         if (sql.toUpperCase().startsWith("SELECT")) {  
•             result.append("<TABLE BORDER=1>");  
•             ResultSet rs = s.executeQuery(sql);  
•             ResultSetMetaData rsmd = rs.getMetaData();  
•             // Write table headings  
•             int columnCount = rsmd.getColumnCount();  
•             result.append("<TR>");  
•             for (int i=1; i<=columnCount; i++) {  
•                 result.append("<TD><B>" + rsmd.getColumnName(i) + "</B></TD>\n");  
•             }  
•             result.append("</TR>");  
•             while (rs.next()) {  
•                 result.append("<TR>");  
•                 for (int i=1; i<=columnCount; i++) {  
•                     result.append("<TD>" + StringUtil.encodeHtmlTag(rs.getString(i)) +  
■ " </TD> ");  
•                 }  
•                 result.append("</TR>");  
•             }  
•         }  
•     }  
• }
```

```
• rs.close();
•     result.append("</TABLE>");
• }
• else {
•     int i = s.executeUpdate(sql);
•     result.append("Record(s) affected: " + i);
• }
• s.close();
• con.close();
• result.append("</TABLE>");
• }
• catch (SQLException e) {
•     result.append("<B>Error</B>");
•     result.append("<BR>");
•     result.append(e.toString());
• }
• catch (Exception e) {
•     result.append("<B>Error</B>");
•     result.append("<BR>");
•     result.append(e.toString());
• }
• return result.toString();
• }
```

# Login.jsp

```
■ <HTML>
■   <HEAD>
■     <TITLE>Login Page</TITLE>
■   </HEAD>
■   <BODY>
■     <CENTER>
■       <FORM METHOD=POST ACTION=SQLTool.jsp>
■         <TABLE>
■           <TR>
■             • <TD>User Name:</TD>
■             • <TD><INPUT TYPE=TEXT NAME=userName></TD>
■           </TR>
■           <TR>
■             • <TD>Password:</TD>
■             • <TD><INPUT TYPE=PASSWORD NAME=password></TD>
■           </TR>
■           <TR>
■             • <TD><INPUT TYPE=RESET></TD>
■             • <TD><INPUT TYPE=SUBMIT VALUE="Login"></TD>
■           </TR>
■         </TABLE>
■       </FORM>
■     </CENTER>
■   </BODY>
■ </HTML>
```

# SQLTool.jsp

```
■ <jsp:useBean id="theBean" class="com.brainysoftware.web.SQLToolBean">
■ <%
•   try {
•     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
•   }
•   catch (Exception e) {
•     out.println(e.toString());
•   }
■ %>
■ </jsp:useBean>
■ <jsp:setProperty name="theBean" property="userName"/>
■ <jsp:setProperty name="theBean" property="password"/>
■ <jsp:setProperty name="theBean" property="connectionUrl"
■ value="jdbc:odbc:JavaWeb"/>
■ <jsp:setProperty name="theBean" property="sql"/>
■ <HTML>
■   <HEAD>
■     <TITLE>SQL Tool</TITLE>
■   </HEAD>
■   <BODY>
■     <BR><H2>SQL Tool</H2>
■     <BR>Please type your SQL statement in the following box.
■     <BR>
```

```
<BR><FORM METHOD=POST>
<INPUT TYPE=HIDDEN NAME=userName VALUE=<jsp:getProperty name="theBean"
property="userName"/>">
<INPUT TYPE=HIDDEN NAME=password VALUE=<jsp:getProperty name="theBean"
property="password"/>">
<TEXTAREA NAME=sql COLS=80 ROWS=8>
<jsp:getProperty name="theBean" property="sql"/>
</TEXTAREA>
<BR>
<INPUT TYPE=SUBMIT>
</FORM>
<BR>
<HR>
<BR>
<%= theBean.getResult() %>
</BODY>
</HTML>
```