

# JavaServer Pages

Carrera: COMPUTACIÓN E INFORMÁTICA

Semestre: 2016 - II

Nombre de Unidad Didáctica: TALLER DE PROGRAMACION CONCURRENTE

Docente: PEDRO HUGO VALENCIA MORALES



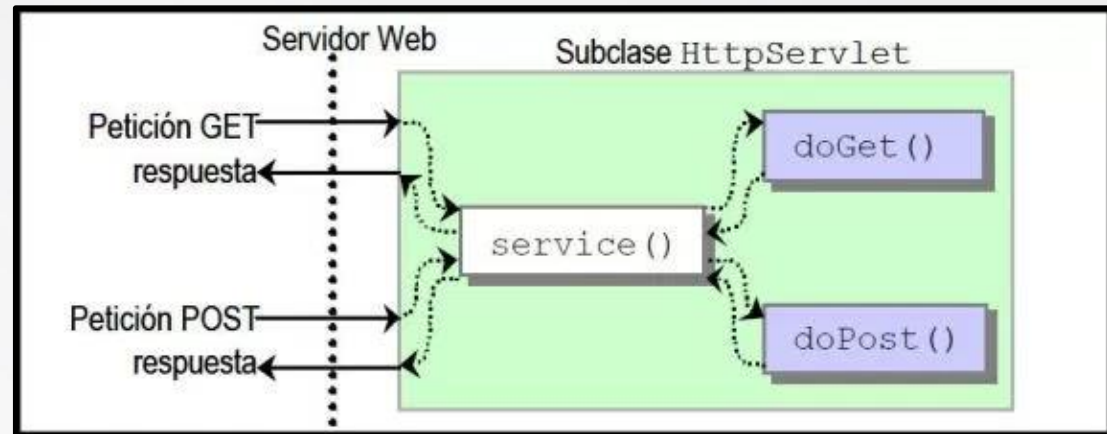
1. **Introducción**
2. **Scripting**
3. **La directiva page**
4. **Inclusión de archivos**
5. **Uso de JavaBeans**
6. **Conclusiones**
7. **Referencia**

# 1. Introducción: Video

<https://youtu.be/2Lw0WOb5cvc>



- Con servlets es sencillo realizar lectura de datos desde formularios



- Pero es laborioso usar instrucciones `println()` para generar HTML

```
response.setContentType("text/html");  
PrintWriter out=response.getWriter();  
out.println("<html>");  
out.println("<body>");  
out.println("Hello World...");  
out.println("</body>");  
out.println("</html>");
```

- **Propósito**

- Uso de HTML en la mayor parte de la pagina
- Delimitar código servlet con etiquetas especiales
- La pagina JSP se traduce a un servlet y esto es invocado por cada petición

- **Ejemplo**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>Saludo</TITLE>
    <LINK REL=STYLESHEET HREF="JSP-Styles.css" TYPE="text/css">
  </HEAD>
  <BODY>
    <H2>Saludo</H2>
    Hola <I><%= request.getParameter("nombre") %></I>!
  </BODY></HTML>
```

## 1.3 Beneficios de JSP

- Hace mas sencillo la escritura, lectura y edición de HTML
- Hace posible el uso de herramientas HTML estándar como Dream Weaver
- Separa el código Java que crea el contenido del código HTML que lo presenta

## 1.4 Ejemplo 1: Etiquetas HTML y Expresiones JSP

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>JSP Expressions</TITLE>
    <META NAME="keywords" CONTENT="JSP, JavaServer Pages">
    <META NAME="descripcion" CONTENT="Ejemplo de expresiones JSP.">
    <LINK REL=STYLESHEET HREF="JSP-Styles.css" TYPE="text/css">
  </HEAD>
  <BODY>
    <H2>Expresiones JSP</H2>
    <UL>
      <LI>Hora actual: <%= new java.util.Date() %>
      <LI>Servidor: <%= application.getServerInfo() %>
      <LI>Sesion ID: <%= session.getId() %>
      <LI>El <CODE>param</CODE> parametro:
        <%= request.getParameter("param") %>
    </UL>
  </BODY></HTML>
```





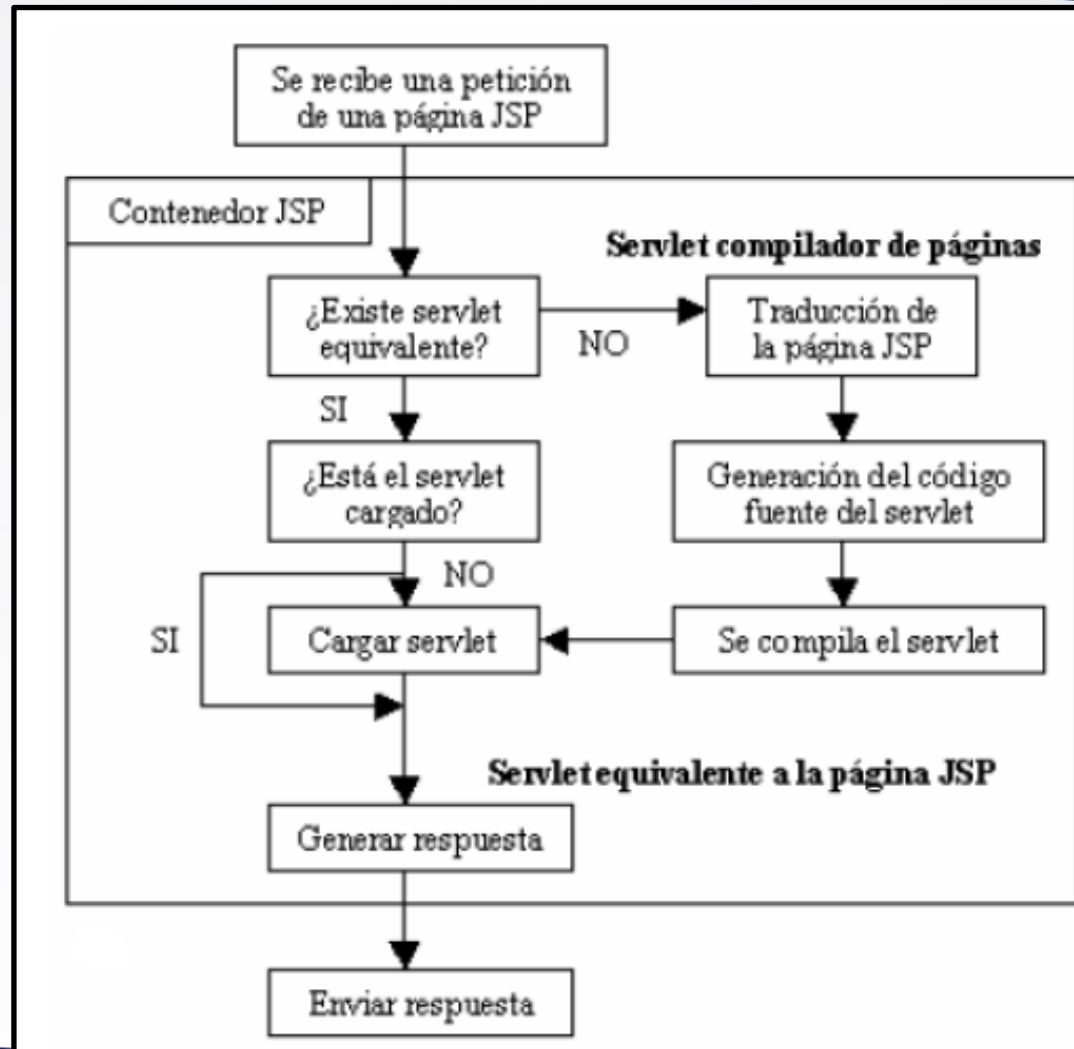
<http://localhost:8080/PryWebJSP/Expresiones.jsp?param=HugoVM>

### Expresiones JSP

- Hora actual: Sat Aug 27 12:28:01 COT 2016
- Servidor: GlassFish Server Open Source Edition 4.1.1
- Sesion ID: d0ac47239506a8a20c0abe90e722
- El param parametro: HugoVM



## 1.5 Ciclo de vida de un JSP



Comment tag

< %...comment...% >



Directive tag

< %@ directive % >



Scriptlet tag

< % scriptlets % >



Expression tag

< %= expression % >



Declaration tag

< %! declarations % >



- **Expresiones**
  - Formato: `<%= expresión %>`
  - Evalúa e inserta `out.println("expresión")` en el método `_jspService()` del servlet
- **Scriptlets**
  - Formato: `<% código java %>`
  - Inserta código java en el método `_jspService()` del servlet
- **Declaraciones**
  - Formato: `<%! código java %>`
  - Inserta código java dentro de la definición de la clase servlet, fuera de cualquier método existente

## 2.2 Ejemplo 2: Expresiones JSP

```
<UL>
<LI>Hora actual: <%= new java.util.Date() %>
<LI>Servidor: <%= application.getServerInfo() %>
<LI>Sesion ID: <%= session.getId() %>
<LI>El <CODE>param</CODE> parametro:
    <%= request.getParameter("param") %>
</UL>
```

Expresiones.jsp

Expresiones\_jsp.java

```
out.write(" <UL>\r\n");
out.write(" <LI>Hora actual: ");
out.print(new java.util.Date());
out.write("\r\n");
out.write(" <LI>Servidor: ");
out.print(application.getServerInfo());
out.write("\r\n");
out.write(" <LI>Sesion ID: ");
out.print(session.getId());
out.write("\r\n");
out.write(" <LI>El <CODE>param</CODE> parametro:\r\n");
out.write(" ");
out.print(request.getParameter("param"));
out.write("\r\n");
out.write(" </UL>\r\n");
```

- **request**
  - Primer argumento del método service/doGet
- **response**
  - Segundo argumento del método service/doGet
- **out**
  - Usado para enviar contenido web al cliente
- **session**
  - Objeto sesión asociado al request
- **application**
  - Contexto del servlet obtenido por `getServletContext()`

## 2.4 Ejemplo 3: Scriptlets JSP

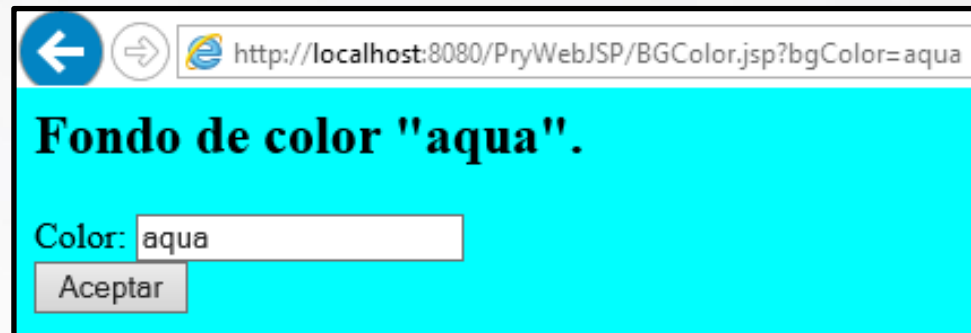
```
<%  
String bgColor = request.getParameter("bgColor");  
if ((bgColor == null) || (bgColor.trim().equals(""))){  
    bgColor = "WHITE";  
}  
%>  
<BODY BGCOLOR="<%= bgColor%>">  
  <H2 ALIGN="CENTER">Fondo de color "<%= bgColor%>".</H2>  
  <BR>  
  <FORM>  
    Color: <INPUT TYPE="TEXT" NAME="bgColor"><BR>  
    <INPUT TYPE="SUBMIT" VALUE="Aceptar">  
  </FORM>  
</BODY>
```




← →  http://localhost:8080/PryWebJSP/BGColor.jsp

**Fondo de color "WHITE".**

Color:



← →  http://localhost:8080/PryWebJSP/BGColor.jsp?bgColor=aqua

**Fondo de color "aqua".**

Color:



- Los scriptlets se insertan en el servlet exactamente como se escriben
- **Ejemplo**

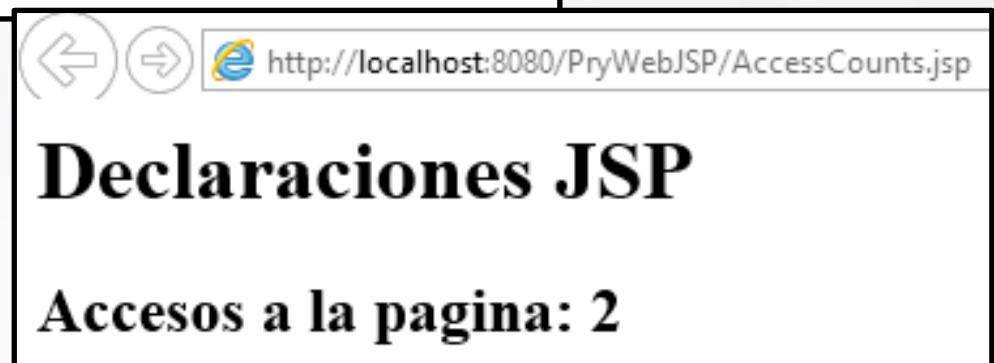
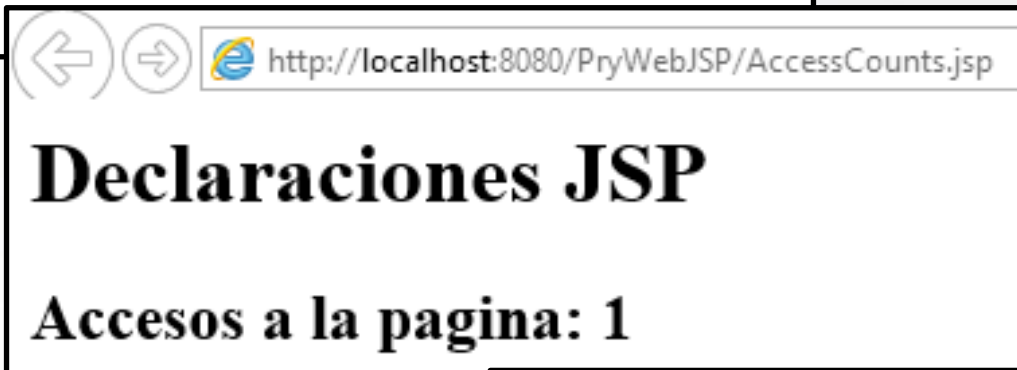
```
<% if (Math.random() < 0.5) { %>  
<H1>Nivel <I>bajo</I>!</H1>  
| <% } else { %>  
<H1>Nivel <I>alto</I>!</H1>  
| <% } %>
```

- **Resultado representativo**

```
if (Math.random() < 0.5) {  
    out.write(" <H1>Nivel <I>bajo</I>!</H1>\r\n");  
} else {  
    out.write(" <H1>Nivel <I>alto</I>!</H1>\r\n");  
}
```

## 2.6 Ejemplo 4: Declaraciones JSP

```
<BODY>
<H1>Declaraciones JSP</H1>
<%! private int accessCount = 0;%>
<H2>Accesos a la pagina:
<%= ++accessCount%></H2>
</BODY>
```





### 3. La directiva page

<% @

page language="java"

buffer="10kb"

autoflush="true"

errorPage="/error.jsp"

import="java.util.\*, javax.sql.RowSet"

%>

## 3.1 Propósito de la directiva page

- Proporciona información del servlet que resulta del JSP
- Permite controlar:
  - Que clases se importan
  - Que clase extiende el servlet
  - Como es manejado los múltiples hilos
  - Si el servlet participa en sesiones
  - El tamaño y comportamiento del búfer de salida
  - Que pagina maneja errores inesperados

- **Formato**

- `<%@ page import="paquete.clase" %>`
- `<%@ page  
import="paquete.clase1,...,paquete.claseN" %>`

- **Propósito**

- Generar instrucciones import al inicio de la definición del servlet

## 3.3 Ejemplo 5: El atributo import

```
<H2>El Atribute import</H2>
<%-- JSP page Directive --%>
<%@ page import="java.util.*" %>
<%-- JSP Declaration --%>
<%!
    private String randomID() {
        int num = (int) (Math.random() * 100000000.0);
        return ("id" + num);
    }
%>
<%-- JSP Scriptlet --%>
<%
    String ID;
    ID = randomID();
%>
<%-- JSP Expressions --%>
Esta pagina fue accedido el <%= new Date() %><br>
y genero el identificador <%= ID %>.
```



http://localhost:8080/PryWebJSP/ImportAttribute.jsp

### El Atribute import

Esta pagina fue accedido el Sat Aug 27 20:00:43 COT 2016  
y genero el identificador id9000038.

- **Formato**

- `<%@ page contentType="Tipo-MIME" %>`
- `<%@ page contentType="Tipo-MIME;  
charset=Character-Set" %>`

- **Propósito**

- Especificar el tipo MIME de la pagina generado por el servlet que resulta de la pagina JSP

## 3.5 Ejemplo 6: Generando hojas de calculo Excel

First Last Email Address

Pedro Valencia pedro.valencia@institutoemprendedores.pe

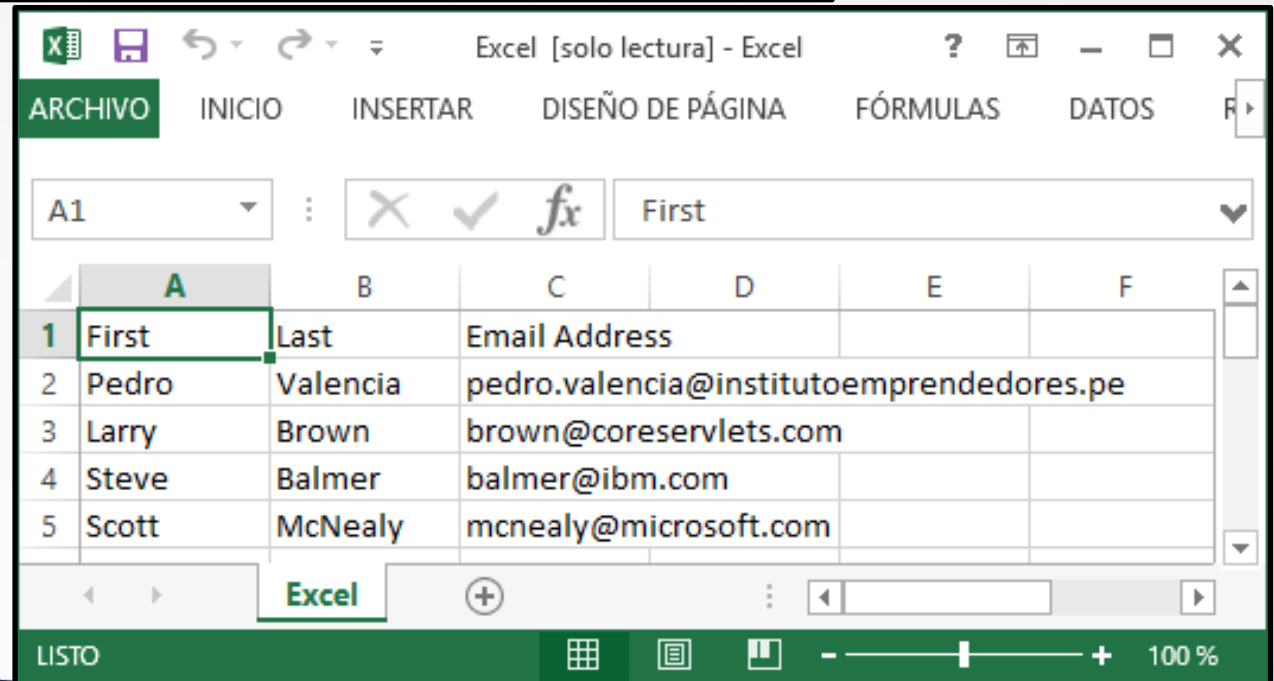
Larry Brown brown@coreservlets.com

Steve Balmer balmer@ibm.com

Scott McNealy mcnealy@microsoft.com

<%@ page contentType="application/vnd.ms-excel" %>

<%-- Columnas separados por TAB --%>



The screenshot shows the Microsoft Excel application window titled "Excel [solo lectura] - Excel". The ribbon includes "ARCHIVO", "INICIO", "INSERTAR", "DISEÑO DE PÁGINA", "FÓRMULAS", and "DATOS". The formula bar shows "First". The spreadsheet has columns A through F and rows 1 through 5. The data is as follows:

	A	B	C	D	E	F
1	First	Last	Email Address			
2	Pedro	Valencia	pedro.valencia@institutoemprendedores.pe			
3	Larry	Brown	brown@coreservlets.com			
4	Steve	Balmer	balmer@ibm.com			
5	Scott	McNealy	mcnealy@microsoft.com			

The status bar at the bottom shows "LISTO", a grid icon, a print icon, a zoom slider set to 100%, and a plus icon.

## 3.6 Ejemplo 7: Generando hojas de calculo condicional

```
<H2>Comparando Manzanas y Naranjas</H2>
```

```
<%  
    String format = request.getParameter("format");  
    if ((format != null) && (format.equals("excel"))) {  
        response.setContentType("application/vnd.ms-excel");  
    }  
%>
```

```
<TABLE BORDER=1>
```

```
    <TR><TH></TH>                <TH>Manzanas<TH>Naranjas  
    <TR><TH>Primer Trimestre <TD>2307    <TD>4706  
    <TR><TH>Segundo Trimestre<TD>2982    <TD>5104  
    <TR><TH>Tercer Trimestre <TD>3011    <TD>5220  
    <TR><TH>Cuarto Trimestre <TD>3055    <TD>5287
```

```
</TABLE>
```

### 3.6 Ejemplo 7: Generando hojas de calculo condicional

← → <http://localhost:8080/PryWebJSP/ApplesAndOranges.jsp?format=excel>

## Comparando Manzanas y Naranjas

	Manzanas	Naranjas
Primer Trimestre	2307	4706
Segundo Trimestre	2982	5104
Tercer Trimestre	3011	5220
Cuarto Trimestre	3055	5287

Excel

ARCHIVO INICIO INSERTAR DISEÑO DE PÁGINA FÓRMULAS

A1 :

	A	B	C	D	E
1	Comparando Manzanas y Naranjas				
2					
3		Manzanas	Naranjas		
4	Primer Trimestre	2307	4706		
5	Segundo Trimestre	2982	5104		
6	Tercer Trimestre	3011	5220		
7	Cuarto Trimestre	3055	5287		

ApplesAndOra

LISTO 100 %



- **Formato**
  - `<%@ page errorPage="URL relativo" %>`
- **Propósito**
  - Especifica la pagina JSP que procesa cualquier excepción lanzada pero no capturada en la pagina

- **Formato**
  - `<%@ page isErrorPage="true" %>`
  - `<%@ page isErrorPage="false" %> <%-- defecto --%>`
- **Propósito**
  - Indica si la pagina actual puede actuar como pagina de error para otras paginas JSP

## 3.9 Ejemplo 8: Pagina de error

```
<%@ page errorPage="PaginaError.jsp" %>
<%!
    private double toDouble(String value) {
        return (Double.parseDouble(value));
    }
%>
<%
    double decimal = toDouble(request.getParameter("param"));
%>
Decimal: <%= decimal%>.
```

Conversion.jsp

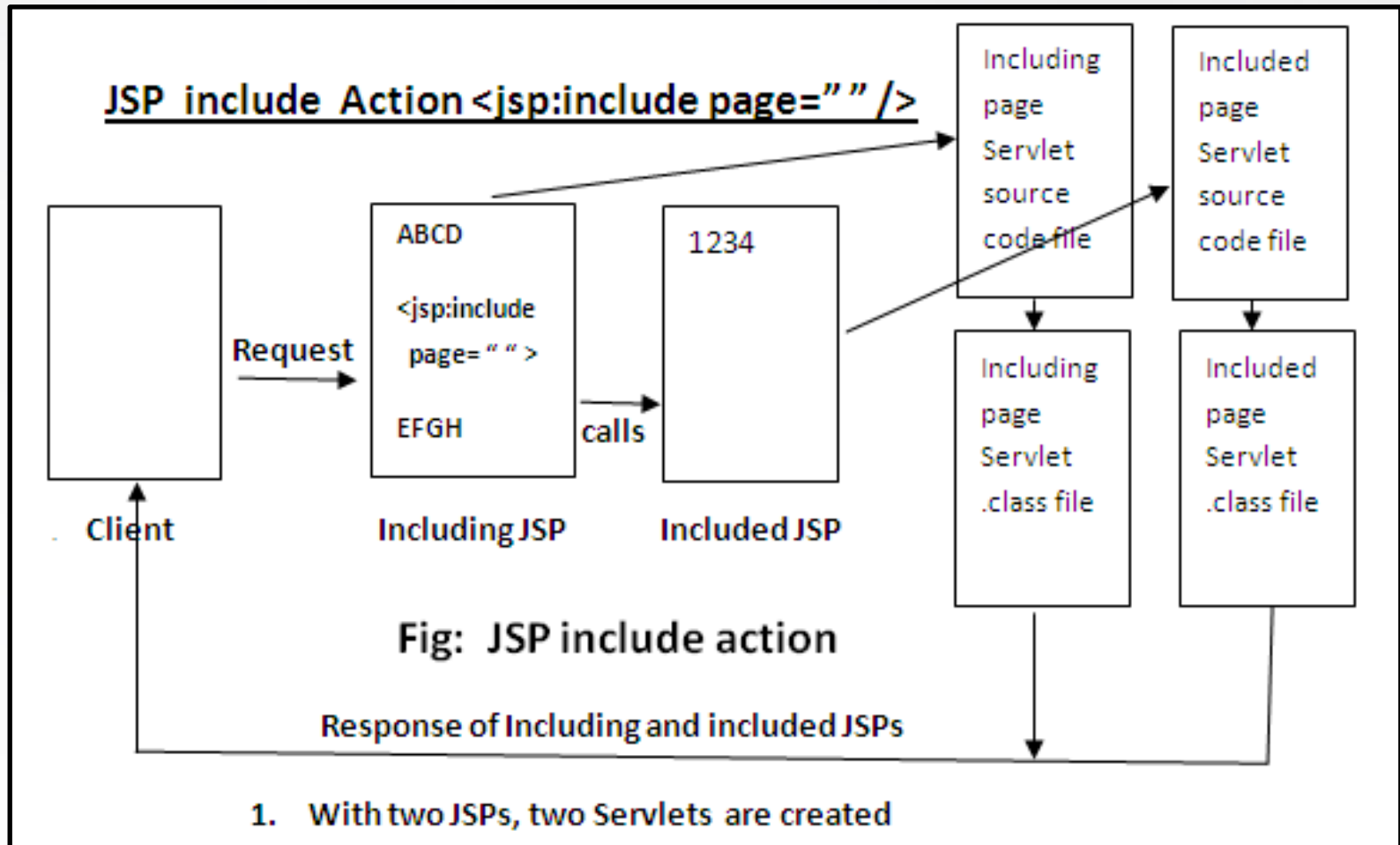
PaginaError.jsp

```
<%@ page isErrorPage="true" %>
Conversion.jsp reporto el siguiente error:
<I><%= exception%></I> en el lugar:
<PRE>
    <%@ page import="java.io.*" %>
    <% exception.printStackTrace(new PrintWriter(out));%>
/PRE>
```

## 3.9 Ejemplo 8: Resultados



## 4. Inclusión de archivos



- **Formato**
  - `<jsp:include page="dirección relativa" />`
- **Propósito**
  - Para reutilizar JSP, HTML o texto plano
  - Permite actualizar contenido incluido sin cambiar la pagina JSP principal

## 4.2 Ejemplo 9: jsp:include

```
<OL>
<LI><jsp:include page="Item1.jsp" /></LI>
<LI><jsp:include page="Item2.jsp" /></LI>
</OL>
```

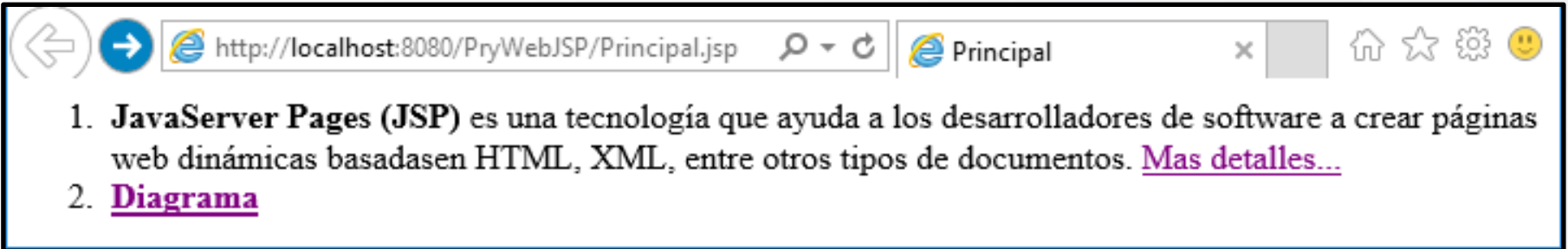
### Item1.jsp

**JavaServer Pages (JSP)** es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos.

[Mas detalles...](https://es.wikipedia.org/wiki/JavaServer_Pages)

### Item2.jsp

**[Diagrama](#)**



- **Formato**
  - `<%@ include file="dirección relativa" %>`
- **Propósito**
  - Para reutilizar contenido JSP en múltiples paginas, donde el contenido afecta a la pagina principal



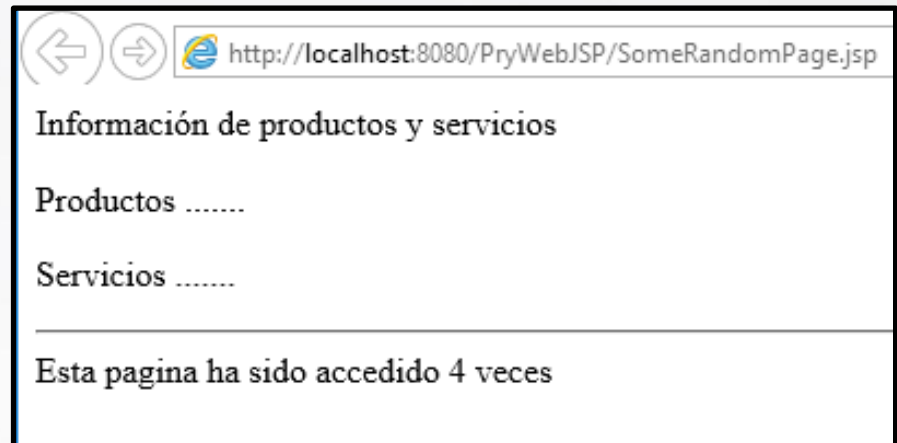
## 4.4 Ejemplo 10: Reutilización de paginas

```
<%!  
private int accessCount = 0;  
%>  
<HR>  
Esta pagina ha sido accedido <%= ++accessCount%> veces
```

**SeccionContacto.jsp**

```
<P>  
Información de productos y servicios  
<P>  
Productos .....  
<P>  
Servicios .....  
<%@ include file="SeccionContacto.jsp" %>
```

**Resultado**



name of bean i.e object

fully qualified classname

```
<jsp:useBean id="person" class="PersonBean"  
             scope="request" >
```

scope of bean

- **Clases Java con ciertas convenciones**
  - Debe tener un constructor con cero argumentos
  - No debe tener variables de instancia publicas
  - Valores de las variables de instancia deben ser accedidos a través de métodos getXxx() y setXxx()

- **jsp:useBean**
  - Crea un nuevo bean
  - `<jsp:useBean id="nombre" class="paquete.Clase" />`
- **jsp:setProperty**
  - Modifica el valor de una propiedad
  - `<jsp:setProperty name="nombre" property="propiedad" value="valor" />`
- **jsp:getProperty**
  - Lee y muestra el valor de una propiedad
  - `<jsp:getProperty name="nombre" property="propiedad" />`

## 5.3 Ejemplo 11: Uso de Beans

```
package beans;

public class StringBean {

    private String message = "Sin mensaje";

    public String getMessage() {
        return (message);
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

```
<jsp:useBean id="stringBean" class="beans.StringBean" />
<OL>
    <LI>Valor inicial:
        <I><jsp:getProperty name="stringBean"
            property="message" /></I>
    <LI><jsp:setProperty name="stringBean"
        property="message"
        value="Uso de JavaBeans" />
    Valor luego de asignacion:
        <I><jsp:getProperty name="stringBean"
            property="message" /></I>
</OL>
```

## 5.3 Ejemplo 11: Resultado



1. Valor inicial: *Sin mensaje*
2. Valor luego de asignacion: *Uso de JavaBeans*

- **JSP es mas conveniente pero no mas potente**
  - Hace mas simple la creación y mantenimiento de HTML, mientras permite acceso a código del servlet
- **Las paginas JSP se traducen a servlets**
  - Es el servlet que se ejecuta al realizar el request
  - Clientes no observan nada relacionado al JSP
- **Se requiere comprender servlets**
  - Para conocer el trabajo real del JSP
  - Conocer cuando servlets es preferible a JSP
  - Combinar servlets y JSP

### **Building Web Apps in Java: Beginning & Intermediate Servlet & JSP Tutorials**

Interested in training from the author of these tutorials? See the upcoming [public J2EE training courses](#) in Maryland (co-sponsored by Johns Hopkins *Engineering for Professionals*, or contact [hall@coreservlets.com](mailto:hall@coreservlets.com) for info on customized courses at *your* location.

Following is an extensive series of tutorials on servlets and JSP, aimed at developers that already [know Java](#) but who have little or no experience with servlets and JSP. Since each section includes exercises and exercise solutions, this can also be viewed as a self-paced JSP and servlet training course. All the slides, source code, exercises, and exercise solutions are free for unrestricted use. Click on a section below to expand its content. The relatively few parts on IDE development and deployment use Eclipse, but of course none of the actual code is Eclipse-specific. Also, although servlets and JSP are very widely used, new projects should seriously consider [the JSF 2 framework](#) as a higher-level alternative to servlets and JSP.