# JSP

¿Qué es un JSP?

Contenido de un JSP

Elementos de script

Directivas JSP

Variables predefinidas

Actions

JSP: INCLUDE

JSP: USEBEAN

JSP: SETPROPERTY

JSP: GETPROPERTY

Propiedades de los JB

Ejemplo final

JSP:FORWARD

# ¿Qué es un JSP?(I)

## SERVLET

***JAVA + HTML insertado***

```
public void doPost(...)....{
nombre=req.getParameter("nombre");
resp.setContentType("text/html");
PrintWriter out = null;
out = resp.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Valores
recogidos en el
formulario</title>");
out.println("</head>");
out.println("<body>");
out.println("<b><font
size=+2>Valores recogidos del
formulario: </font></b>");
out.println("<p><font
size=+1><b>Nombre: </b>" + nombre +
"</font>");
...
```

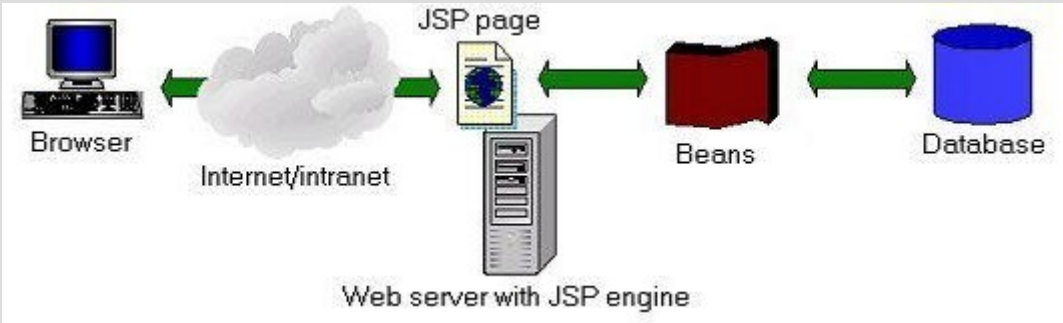
## JSP

***HTML + Java insertado***

```
<HTML>
<HEAD><TITLE>Mi primera pagina
JSP</TITLE></HEAD>
<BODY>
<% String
usuario=request.getParameter("usuario");
%>
<H3>¡Hola <%= (usuario==null) ? "" :
usuario
%>!</H3>
<B>Introduce tu nombre :</B>
<FORM METHOD=get>
<INPUT TYPE="text" NAME="usuario" SIZE=15>
<INPUT TYPE="submit" VALUE="Saluda">
...
```

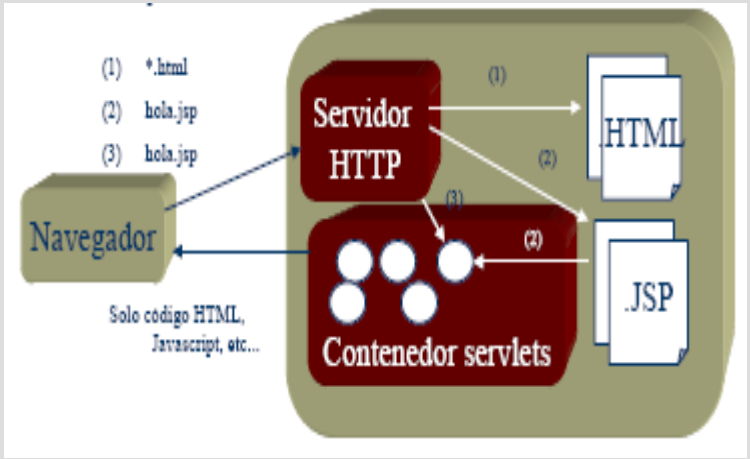
# ¿Qué es un JSP? (II)

- Permiten mezclar en una misma página contenidos HTML estáticos con código de tipo *Script* para la generación de contenidos dinámicos
- Característica diferenciadora de los servlets → separación clara entre parte estática (HTML) y parte encargada de generación dinámica
  - Separar la labor del desarrollador y del diseñador
- Puerta a la siguiente arquitectura:



# ¿Qué es un JSP? (III)

- En realidad:
  - Los JSP se traducen internamente a servlets
  - La primera vez que se pide al servidor, se compila, se traduce a servlet, y las siguientes peticiones se realizan sobre el servlet (hasta que se cambie)
- Primera invocación es más lenta
- Sentencias HTML se copian como out.println
- Código se transforma en código del servlet
- Opciones:
  - JSP + Beans
  - JSP + Servlets + Beans



# Contenido de un JSP

- Elementos que pueden aparecen en una página JSP:
  - Código HTML estático → ocupa la mayor parte de la página y sigue las reglas marcadas por HTML
  - Elementos JSP de Script
  - Directivas JSP
  - Variables predefinidas
  - Actions

# Elementos JSP de Script (I)

- Nos permiten insertar código Java dentro de la página JSP
- Tres elementos de script diferentes:
  - Declaraciones:
    - Definir variables globales o métodos que pueden ser accedidos desde cualquier sitio

      **<%! … %>**
  - Scriplets:
    - Insertar código Java **<% ... %>**
  - Expresiones:
    - Mostrar los resultados producidos por expresiones Java **<% = … %>**

# Elementos JSP de Scrip (II): Declaraciones

- Definir métodos y variables que podrán ser accedidos desde el código situado en cualquier otro punto de la página
- Los métodos se suelen definir dentro de los componentes (beans) que se encargan de la lógica de negocio, pero a veces van dentro de la página
- Ejemplos:

```
<%! private int contador = 0 %>
<%! public String diHola(){
      return "Hola";
%>
```

- Comentarios → <%-- Comentario %> (ignorado por el intérprete)

# Elementos JSP de Scrip (III): Scriplets

- Insertar código java → lógica de presentación
- Podemos usar cualquier clase del API de Java
- El código Java del scriplet se inserta directamente en el servlet (JSP → servlet)

**JSP**

```
<% Calendar ahora=
Calendar.getInstance();
int hora =
ahora.get(Calendar.HOUR_OF_DAY);%>
<b> Kaixo, <i>
<% if ((hora>20)||(hora<6)) { %>
    Gabon
<% }else if ((hora>=6)&&(hora<=12)){
%>
    Egun on
<% } else { %>
    Arratsalde on
<% } %> </i> </b >
```

**Servlet**

```
Calendar ahora= Calendar.getInstance();
int hora =
ahora.get(Calendar.HOUR_OF_DAY);
out.println("<b> Kaixo, <i>");
if ((hora>20)||(hora<6)) {
   out.println("Gabon");
}else if ((hora>=6)&&(hora<=12)){
   out.println("Egun on");
} else {
  out.println("Arratsalde on</i> </b >");
```

# Elementos JSP de Scrip (IV): Expresiones

- Permite insertar el resultado de una expresión Java → <%= Expresión Java %>
- La expresión es evaluada en tiempo de ejecución. Su rtdo. se convierte a String y será insertado dinámicamente en la página resultado
- Ejemplos:

```
Esta página ha sido generada: <%=new Java.util.Date() %>

Debes <%= x+y-z %> euros

La suma es: <%= miClase.sumar(3,2) %>
```

# Directivas (I)

- Permiten configurar ciertos aspectos de la página JSP
- Sintaxis: <%@directiva atributo= “valor”%>
- Puede tener varios atributos
- Dos tipos de directivas básicos:
  - Directiva PAGE: configurar ciertos atributos como el lenguaje script a utilizar, clases a importar, tipo de contenido a generar
  - Directiva INCLUDE: permite insertar el contenido de otros ficheros (p.e. cabeceras o pies de página)

# Directivas (II): PAGE

- Atributos destacados:
  - *import= "package.class"*
    - Clases a importar en la página

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

  - *contentType= "tipoMIME"*
    - Tipo de contenido que va a generar la página JSP

```
<%@ page contentType= "text/html" %>
```

  - *info= "mensaje"*
    - String que identifique a la página. Puede ser consultado mediante getServletInfo()

```
<%@ page info= "JSP que saluda" %>
```

# Directivas (III): PAGE

- Más atributos:
  - *language= "java"*
    - Indicar el tipo de lenguaje a utilizar para el código
  - *errorPage= "url"*
    - URL de la página de error que será invocada si se produce alguna excepción no capturada
  - *isErrorPage= "true|false"*
    - Indica si la página JSP actúa o no como página de error de otra página. Por defecto es false
  - isThreadSafe= "true|false"
    - Un valor true (por defecto) indica que múltiples peticiones pueden procesarse con un sólo ejemplar del servlet (multihilo)

# Directivas (IV): INCLUDE

- Permite incluir el contenido de un fichero dentro de la página JSP (en el momento en que se transforma a servlet)

```
<%@ include file= "url relativa" %>
```

- El contenido es incrustado como parte de la página JSP. Puede contener: HTML estático, elementos de script, directivas y actions
- Utilidad → reutilizar páginas construidas (frames, barras de herramientas, barra de navegación, etc.)
- Si lo que se inserta cambia después de haber sido convertida la página, los cambios no se verán reflejados → útil cuando no se vaya a modificar frecuentemente

# Directivas (V): INCLUDE

## Fichero Velocidad.jsp

```
<html>
  <head>
    <title>Ejemplo de un include</title>
  </head>
  <body bgcolor="white">
    <%@ include file=/logotipo.html %>
    <font color="blue">
      Fecha y hora actual: <%@ include file="Date.jsp" %>
    </font>
  </body>
</html>
```

## Fichero Date.jsp

```
<%@ page import="java.util.*" %>
<%= (new Date() ).toLocaleString() %>
```

# Ejemplo (I)

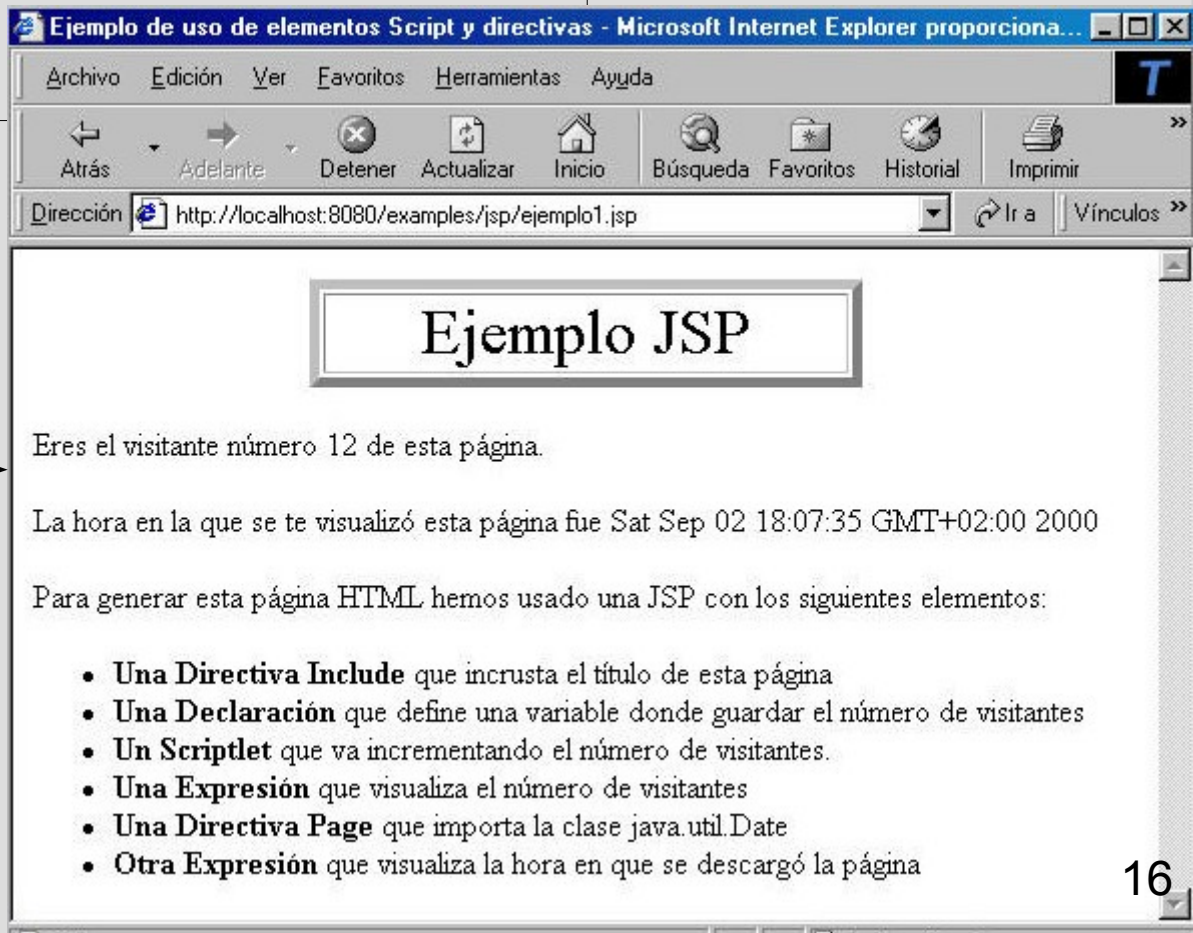
```
<HTML>
<HEAD>
<TITLE>Ejemplo de uso de elementos Script y directivas</TITLE>
</HEAD><BODY>
<%@ include file="cabecera.html" %>
<%@ page import="java.util.Date" %>
<%! int numVisitantes = 0; %>
<% numVisitantes++; %>
Eres el visitante número <%= numVisitantes %> de esta página.
<P>La hora en la que se te visualizó esta página fue <%= new Date() %><P>
Para generar esta página HTML hemos usado una JSP con los siguientes
elementos:
<UL>
<LI><B>Una Directiva Include</B> que incrusta el título de esta página<BR>
<LI><B>Una Declaración</B> que define una variable donde guardar el número de
visitantes<BR>
<LI><B>Un Scriptlet</B> que va incrementando el número de visitantes.<BR>
<LI><B>Una Expresión</B> que visualiza el número de visitantes<BR>
<LI><B>Una Directiva Page</B> que importa la clase java.util.Date<BR>
<LI><B>Otra Expresión</B> que visualiza la hora en que se descargó la
página<BR>
</UL></BODY></HTML>
```

# Ejemplo (II)

```
<CENTER>
<TABLE WIDTH="50%" BORDER="5">
<TR>
<TD ALIGN="CENTER"><FONT SIZE="+3">Ejemplo JSP</FONT></TD>
</TR>
</TABLE>
</CENTER><BR>
```

**Código del fichero cabecera.html**

**Página HTML generada**



# Variables predefinidas (I)

- Los JSP definen una serie de variables implícitas que pueden ser usadas dentro de las expresiones JSP y Scriptlets. Las más importantes son:
  - ***OUT:*** JSPWriter para enviar la salida al cliente (parecido a PrintWriter de los servlets) mediante el método println

```
<% out.println(<H2> Uso del objeto out</H2>); %>
```

  - ***REQUEST***: Objeto de tipo HttpServletRequest
    - Nos permite obtener los parámetros que llegan a través de un formulario HTML (igual que en los servlets)
    - Permite obtener las cabeceras y el tipo de petición (igual que en los servlets)

```
<%= request.getParameter("nombre") %>
```

# Variables predefinidas (II)

- ***RESPONSE***: Objeto de la clase HttpServletResponse
  - Contiene una serie de métodos para trabajar sobre la respuesta que genera la página (servlets)

```
<% response.setContentType ("text/html"); %>
```

que sería equivalente a...

```
<%@ page contentType= "text/html" %>
```

- ***SESSION***: Objeto de tipo HttpSession asociado con la petición
  - Misma utilidad que en los servlets

- ***EXCEPTION***: Es un objeto de la clase Throwable. Sólo se crea en el caso de que empleemos la directiva siguiente

```
<%@page isErrorPage= "true" %>
```

# Ejemplo

```
<HTML>
<HEAD>
<TITLE>Resultados</TITLE>
</HEAD>

<% String nombre = request.getParameter("nombre");
String apellidos = request.getParameter("apellidos");

if ((nombre.length()==0)||(apellidos.length()==0)){
%>
<H2>¡Gracias por tu información, <%= nombre %>!</H2>
<H3>
<% }else{
    out.println("Has dejado sin rellenar tu nombre o tu comentario.");
%>
</H3>
Por favor, <a href=formulario.html> rellenalo de nuevo</a>
<% }
%>
</BODY>
</HTML>
```

# Actions

- Las acciones nos permiten dar funcionalidad a las páginas JSP:
  - Insertar dinámicamente el contenido de un fichero en la página JSP
  - Invocar otros JSP
  - Trabajar sobre componentes JavaBean
- Las acciones tienen una estructura común basada en construcciones de sintaxis XML (recordar web.xml)
  - Cada etiqueta puede contener otras etiquetas
  - Etiquetas sensibles a mayúsculas y minúsculas

# JSP:Include (I)

- Permite insertar el contenido de un fichero en la página JSP
- Sintaxis:

```
<jsp: include page= "URL" />
```

- Diferencia con respecto a la directiva INCLUDE
  - Con la directiva se inserta el contenido en el momento en que se hace la traducción al servlet.
  - En este caso la traducción se hace cada vez que se solicita la página. Flexibilidad frente a cambios. Además, es posible modificar el contenido de la petición mediante parámetros

# JSP:Include (II)

- Atributos:
  - *page* → página a la cual reenviamos la petición
  - *flush* → especifica si la página incluida va a enviar al cliente inmediatamente el contenido del buffer. Por defecto es false y no hace falta modificarla
  - *name (jsp:param)* → nombre del atributo que le pasamos a la página especificada
  - *value (jsp:param)* → valor del atributo name. Podría ser el resultado de una expresión

```
<jsp: include page= "URL" />
    <jsp:param name="nombre" value= "valor"/>
</jsp:include>
```

# JSP: USEBEAN (I)

- Esta acción nos permite cargar y utilizar un JavaBean en nuestra página JSP.
- ¿Qué es un JavaBean?
  - Componente SW reutilizable que debe cumplir una serie de reglas
  - Clase Java llamada desde una página JSP
- Sintaxis

```
<jsp:useBean id="nombre" class="package.class" scope="page|request|
session|application" />
```

- Ejemplo:

```
<jsp:useBean id="miBean" class="com.clases.miBeanClase" scope="page" />
```

# JSP: USEBEAN (II)

- Atributo *class* → clase de JavaBean que vamos a utilizar (ruta completa de paquetes)
- Atributo *id* → Indica el nombre asociado a la instancia de la clase que vamos a utilizar. (análogo a Libro l1=new Libro())
- Atributo *scope* → Indica el ámbito de la instancia con la que vamos a trabajar
- Por lo tanto, la acción usebean implica instanciar un nuevo *bean* de la clase especificada mediante *class*.
- En el caso de que ya exista un bean instanciado con el mismo nombre (id) y el ámbito de ese bean le haga estar todavía en curso, esta sentencia implicará obtener una referencia al bean existente.

# JSP: USEBEAN (III)

- Ámbito de un bean → contexto dentro del cual está disponible
- Cuatro tipos de ámbitos:
  - *PAGE*: Ámbito por defecto. Sólo disponible en la página en la que fue creado
  - *REQUEST*: Sólo disponible durante la petición en curso
  - *SESSION*: Disponible para todas las páginas durante el tiempo de vida de una sesión de usuario
  - *APPLICATION*: Disponible para todas las páginas durante todo el tiempo que dure la ejecución de la aplicación

# JSP: SETPROPERTY (I)

- Nos permite asignar valores a las propiedades de los beans referenciados anteriormente

```
<jsp:setProperty name="myName" property="someProperty"
value="someValue" param="someParam" />
```

- Significado de los atributos:
  - *NAME*: Identifica al bean cuyas propiedades van a ser modificadas. Se debe corresponder con el atributo id de alguna action *USEBEAN*
  - *PROPERTY*: Identifica la propiedad del bean que va a ser modificada
  - *VALUE*: Atributo opcional que indica el valor a asignar a la propiedad. Siempre en forma de String. La conversión de String al tipo de la propiedad se realiza de forma automática
  - PARAM: Atributo opcional que identifica qué parámetros de los que llegan de un formulario serán asignados a la propiedad

# JSP: SETPROPERTY (II)

```
<jsp:setProperty name="BeanOrdenar" property="numElementos"
param="numero" />
```

- En esta sentencia se indica que a la propiedad "*numElementos*" del bean llamado "*BeanOrdenar*" se le debe asignar el valor del parámetro que llega del formulario que llama a la página JSP. En caso de que ese parámetro no existiera, la propiedad no se modificaría
- Si se omiten *value* y *param*, el comportamiento será el mismo que si se proporcionara un atributo *param* con el nombre de la propiedad.
- Si en *property* se pone un *, se emparejan todas las propiedades del bean con los parámetros que tengan su mismo nombre

# JSP: SETPROPERTY (III)

- La acción JSP: SETPROPERTY puede ser usada:
  - Después de una action de tipo JSP:USEBEAN, pero fuera de ella.

```
<jsp:useBean id="myName" ... />
<jsp:setProperty name="myName" property="someProperty" .../>
```

    - En este caso, se ejecuta siempre, tanto si se ha creado un nuevo bean como si se referencia a uno existente
  - Dentro del cuerpo de inicialización de la acción USEBEAN

```
<jsp:useBean id="myName" ... >
<jsp:setProperty name="myName" property="someProperty" .../>
</jsp:useBean>
```

    - En este caso, se ejecuta sólo si se crea una nueva instancia

# JSP: SETPROPERTY (IV)

- Ejemplos:

```
<jsp:useBean id="miBean" class="beans.BeanValores" scope="request">
<jsp:setProperty name="miBean" property="miPropiedad"
value="123" />
</jsp:useBean>

<jsp:useBean id="miBean" class="beans.BeanValores" scope="request">
<jsp:setProperty name="miBean" property="miPropiedad"
param="elementoFormulario" />
</jsp:useBean>
```

- No tiene sentido usar en una misma acción el atributo value y el atributo param
- Mediante el primero el programador le da un valor de forma explícita y el segundo lo recoge del formulario

# JSP: GETPROPERTY

- Permite obtener el valor de una propiedad del bean, convertirlo a String e insertarlo en la página html de salida generada por la página jsp

```
<jsp:getProperty name="myName" property="myProperty"/>
```

- El atributo name identifica el bean sobre el que se va a consultar la propiedad Deberá ser el nombre de un bean referenciado anteriormente por una action de tipo JSP:USEBEAN
- El atributo PROPERTY identifica la propiedad del bean de la que se va a obtener su valor

```
<jsp:useBean id="miBean" class="beans.BeanValores"
scope="request" />
<H2>Valor: <jsp:getProperty name="miBean"property="atributo"/
></H2>
```

# Propiedades de los JavaBeans

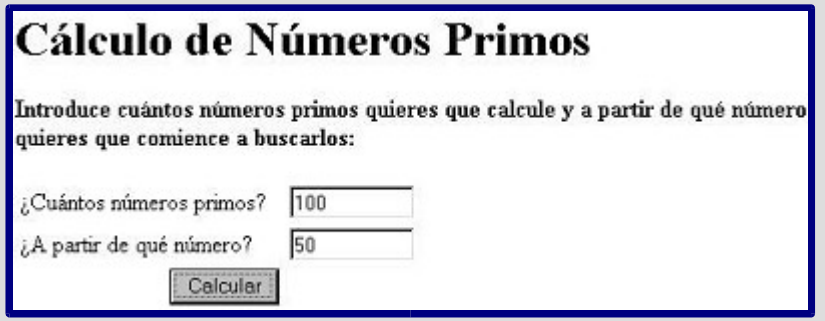
- Todas las propiedades de un JavaBean deben tener una pareja de métodos get/set que permitan consultar y modificar los valores de esas propiedades
  - getXX → Devuelve el valor de la propiedad XX
  - setXX → Permite modificar el valor de la propiedad XX
- Ejemplo:

**Acciones**
**setProperty**
**getProperty**

**Métodos get/set en el Scriptlet**

```
public class PersonaBean{
   String nombre;
   public String getNombre(){
      return n;
   }
   public void setNombre(String n){
     nombre = n;
   }
}
```

# Ejemplo final(I): Formulario




```
<HTML>
<HEAD>
<TITLE>Cálculo de números primos</TITLE>
</HEAD>
<BODY>
<H1>Cálculo de Números Primos</H1>
<P><B>Introduce cuántos números primos quieres que calcule y a partir de qué
número quieres que comience a buscarlos:</B> </P>
<FORM ACTION="numPrimos.jsp" METHOD="GET">
<TABLE WIDTH="50%">
<TR> <TD WIDTH="180">¿Cuántos números primos?</TD>
<TD WIDTH="93">
<P><INPUT TYPE="TEXT" NAME="cuantosPrimos" SIZE="10"></P></TD> </TR>
<TR> <TD WIDTH="180">¿A partir de qué número?</TD>
<TD WIDTH="93">
<P><INPUT TYPE="TEXT" NAME="aPartirNumero" SIZE="10"></P></TD>
</TR> <TR>
<TD COLSPAN="2" ALIGN="CENTER" WIDTH="273">
<P><INPUT TYPE="SUBMIT" VALUE="Calcular"></P></TD> </TR>
</TABLE></FORM>
</BODY>
</HTML>
```

# Ejemplo final(II): Componente JavaBean

```
package numPrimos;
public class PrimosBean{
   String listaPrimos; // La lista de numeros primos generada
   int cuantosPrimos; // El numero de primos a generar
   int desdeNumero; // A partir de que número se generan los primos
   public void setCuantosPrimos(int num){
       this.cuantosPrimos = num;
   }
   public void setDesdeNumero(int num){
       this.desdeNumero = num;
   }
   public String getListaPrimos(){
      return listaPrimos;
   }
   public int getCuantosPrimos(){
      return cuantosPrimos;
   }
   public int getDesdeNumero(){
     return desdeNumero;}
```

# Ejemplo final(III): Componente JavaBean

```
    public void calcularNumerosPrimos(){ // Algoritmo
      listaPrimos = "";
      int primo = desdeNumero;
      for (int i=0;i<cuantosPrimos;){
          int divisor = 2;
          boolean noEncontrado = true;
          while(noEncontrado && (divisor < primo)){
              if (primo%divisor == 0){
                 noEncontrado = false;
              }else{
                 divisor++;
              }
          }
          if (noEncontrado){
               listaPrimos = listaPrimos + " " + primo;
               i++;
          }
          primo++;
      }
    }
}
```

# Ejemplo final (IV): Página JSP

```
<HTML>
<HEAD>
<TITLE>Resultados</TITLE>
</HEAD>
<BODY>
<jsp:useBean id="bean" class="numPrimos.PrimosBean" scope="page" />
<jsp:setProperty name="bean" property="cuantosPrimos" />
<jsp:setProperty name="bean" property="desdeNumero"
param="aPartirNumero" />
<% bean.calcularNumerosPrimos(); %>
<H1>Resultados Obtenidos</H1>
<P><B>Estos son los <jsp:getProperty name="bean"
property="cuantosPrimos"/>
primeros números primos que existen a partir del número
<jsp:getProperty name="bean" property="desdeNumero" />:</B></P>
<P><jsp:getProperty name="bean" property="listaPrimos" /></P>
<P><A HREF="formulario.html">Volver al formulario</A></P>
</BODY>
</HTML>
```

# Ejemplo final (V): Página HTML resultado

## Resultados Obtenidos

**Estos son los 100 primeros números primos que existen a partir del número 50:**

53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157
163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269
271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389
397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509
521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631

Volver al formulario

# Action JSP:Forward

- Permite redirigir una petición desde una página JSP hacia otra

```
<jsp:forward page="/utils/errorReporter.jsp" />

<jsp:forward page="<%= someJavaExpresion %>" />
```

- El atribut page contiene la URL relativa de la página hacia la que se redirige la petición. Puede ser un valor estático o una expresión
- El control pasa a la página redireccionada y ya no vuelve a la página de origen