Servicios Telemáticos Avanzados

3.- PÁGINAS WEB DINÁMICAS

OpenCourseWare 2014

Maider Huarte y Gorka Prieto
Escuela Técnica Superior de Ingeniería de Bilbao
Departamento de Ingeniería de Comunicaciones
Universidad del País Vasco (UPV/EHU)

Servicios Telemáticos Avanzados:
3.- PÁGINAS WEB DINÁMICAS.odp

PÁGINAS WEB DINÁMICAS

ÍNDICE

1.- INTRODUCCIÓN

- Páginas web
  - Definición
  - Estáticas vs dinámicas
    - Estáticas:
    - Dinámicas:
      - Client-side:
      - Server-side:

2.- Client-side scripting

2.1.- DHTML

- Contexto: páginas web dinámicas con client-side scripting
- Definición
- Componentes
  - HTML
  - CSS
  - Javascript
  - DOM

2.- Client-side scripting

2.1.- DHTML

- HTML
  - Incorporación de CSS y scripts: **<head></head>**
    - **<style></style>**
      - **type=”text/css”**
      - **<link>**
    - **<script></script>**
    - **<noscript></noscript>**
  - Eventos HTML
    - Para qué?
    - Atributos para ciertos elementos HTML: *atributo_de_evento*=”*llamada_a_script*”
      - De ventana (<body>): **onload**, **onunload**,...
      - De formulario (y elementos internos): **onblur**/**onfocus**, **onchange**, **onselect**, **onsubmit**,...
      - De teclado: **onkeydown**/**onkeyup**, **onkeypress**
      - De ratón: **onclick**,...

2.- Client-side scripting

2.1.- DHTML

- CSS
  - Uno de los componentes de DHTML
  - No introduce dinamismo en las páginas
  - Gestión sencilla del aspecto de las páginas

2.- Client-side scripting

2.1.- DHTML

- JavaScript
  - Definición
  - Ubicación de scripts JS
    -
      - ►
      - ►
    -
  - Ejecución
    - Sentencias JS
    - Funciones

2.- Client-side scripting

2.1.- DHTML

- JavaScript
  - Sintáxis:
    - Comentarios: //, /* */
    - Sentencias: ;
    - Funciones: **function** *nombre***(***argumentos***)**
      {
      Sentencias
      }
    - Variables y tipos de datos
      - Declaración: **var** *nombre***;**
      - Declaración+inicialización: **var** *nombre***=***valor***;**
    - Operadores:
    - Sentencias selectivas
    - Sentencias repetitivas

2.- Client-side scripting

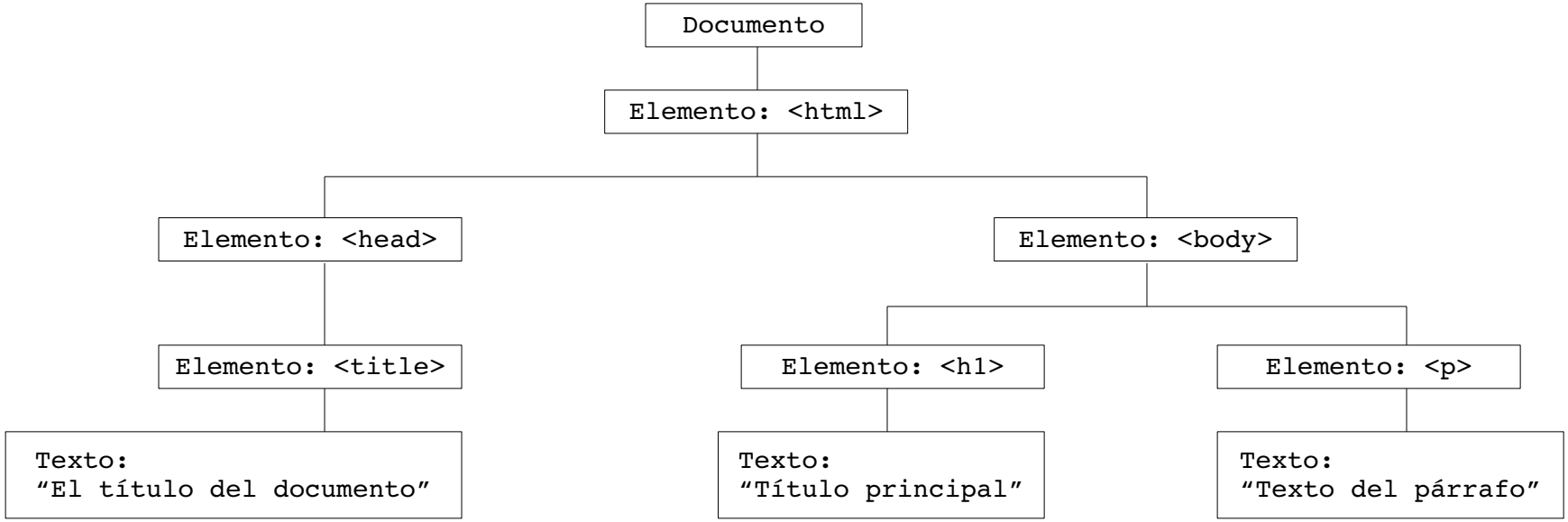
2.1.- DHTML

- JavaScript
  - Sintáxis:
    - Objetos
      - Elementos:
        - Propiedades: valores
        - Métodos: funciones
      - Objetos predefinidos
        - **String, Number, Array, Math**
        - **Window: alert(), confirm(), prompt(), open(), close()**
      - Acceder a los campos: operador **.**
      - Crear objetos: operador **new**
        - excepto:
    - Errores:

2.- Client-side scripting

2.1.- DHTML

- DOM
  - Definición
  - Árbol jerárquico de elementos y contenidos: nodos



  - Acceso a nodos para lenguajes client-side script

2.- Client-side scripting

2.1.- DHTML

- DOM
  - Objetos DOM predefinidos
    - **node**: para cualquier nodo
      - Propiedades: **atributes, nodeName, childNodes**
      - Métodos: **appendChild(), replaceChild(), removeChild()**
    - **document**: nodo raíz, no tiene padre. Todo lo de node +
      - Propiedades: **body, forms, images, title,** ...
      - Métodos: **getElementById(), getElementsByTagName(), createElement(), renameNode(), open(), write(), close()**,...
    - **element**: para cualquier elemento HTML. Todo lo de node +
      - Propiedades: **id, innerHTML, tagName,** …
      - Métodos: **getAttribute(), setAtribute(),**...

2.- Client-side scripting

2.1.- DHTML

- DOM
  - Objetos DOM predefinidos
    - **body**: Todo lo de node y element +
      - Propiedades: **background, bgColor, onload**...
      - Métodos: -
    - **form**: Todo lo de node y element +
      - Propiedades: **name, action, method, target, length**...
      - Método: **reset(), submit()**
    - **checkbox**: Todo lo de node y element +
      - Propiedades: **checked, disabled, name, value**...
      - Métodos: -
    - **image, link, table**: Todo lo de node y element +
    - **style**: para trabajar con estilos CSS

2.- Client-side scripting

2.1.- DHTML

- EJEMPLOS JAVASCRIPT+DOM

```
<!DOCTYPE html>
<html>
<head>
<script>
function mostrarFecha()
{
  document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>Mi primer JavaScript</h1>
<p id="demo">Esto es un párrafo de texto</p>
<button type="button" onclick="mostrarFecha()">Ver fecha</button>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<script>
function cuidadoWrite()
{
   document.write("DESTRUCCIÓN TOTAL!!!!");
}
</script>
</head>
<body>
<h1>Mi segundo JavaScript</h1>
<p id="demo">Esto sigue siendo un párrafo.</p>
<button type="button" onclick="cuidadoWrite()">Comprueba document.write</button>
</body>
</html>
```

2.- Client-side scripting

2.1.- DHTML

- EJEMPLOS JAVASCRIPT+DOM

```
<!DOCTYPE html>
<html>
<head>
<script>
function cambiaNodos()
{
   var ps=document.getElementsByTagName("p");
   ps[0].innerHTML+=" ";
   var subp0=document.createElement("strong");
   subp0.innerHTML="Esto es texto importante al final del párrafo";
   ps[0].appendChild(subp0);

   var ns=document.body.childNodes;
   document.body.removeChild(ns[5]);
   document.body.removeChild(ns[4]);
}

function cambiaEstilo()
{
   var p1=document.getElementById("p1");  // Acceso al elemento
   p1.style.backgroundColor="red";    // Cambio de estilo
}
</script>

</head>
<body>
<h1>ESTE ES EL PRIMER TÍTULO DEL DOCUMENTO</h1>
<p id="p1">Este es el primer párrafo del documento</p>
<h2>ESTE ES EL PRIMER SUBTÍTULO DEL PRIMER TÍTULO DEL DOCUMENTO</h2>
<p>Este es el segundo párrafo del documento</p>
<p>Este es el tercer párrafo del documento</p>

<button type="button" onclick="cambiaNodos()">Cambia los párrafos</button>
<button type="button" onclick="cambiaEstilo()">Cambia el fondo</button>

</body>
</html>
```

2.- Client-side scripting

2.1.- DHTML

- EJEMPLOS JAVASCRIPT+DOM

```
<!DOCTYPE html>
<html>
<head>
<script>
function validaFormulario()
{
  var r=true;
  var h=document.getElementById("i1").value;
  var m=document.getElementById("i2").value;

  if(h==null||h==""||h<0||h>23||m==null||m==""||m<0||m>59)
  {
      alert("La hora no es válida");
      document.getElementById("i1").value="";
      document.getElementById("i2").value="";
      r=false;
  }
  else
      alert("La hora es válida");
  return r;
}
</script>
</head>
<body>
<h1>Mi cuarto JavaScript: validar formularios</h1>
<form action="" onsubmit="return validaFormulario()" method="post">
Introduce la hora <br>
<input id="i1" type="text" name="hora" maxlength="2">:<input id="i2" type="text" name="min" maxlength="2"><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

3.- Server-side scripting

- Entornos de ejecución más utilizados
  - Servidor web+CGI
  - Servidor web+conector servidor de aplicaciones
  - Servidor de aplicaciones standalone
  - Ejemplos
    - Apache
    - Glassfish
    - JBoss
    - Tomcat
    - Jetty
- Clientes: navegadores

3.- Server-side scripting

3.1.- CGI

- Definición
- Variables de entorno y E/S estándar
  - Cualquier lenguaje
  - Típico PERL
- Request
  - GET
    - Variables en la URL
    - QUERY_STRING
  - POST
    - Variables en el cuerpo
    - CONTENT_LENGHT y stdin

3.- Server-side scripting

3.1.- CGI

- Response
  - stdout
  - Campos cabecera HTTP
    - Content-type: text/html
    - [línea en blanco]
  - Cuerpo
    - HTML, imagen, etc.
  - Otras variables de entorno
    - REQUEST_METHOD, HTTP_USER_AGENT, REMOTE_ADDR, REMOTE_USER, HTTPS, etc.

3.- Server-side scripting

3.2.- Servlets

- Sustitución de CGIs
  -
  -
  -
  -
- Jerarquía Java de servlets

java.lang.Object
↳javax.servlet.GenericServlet (abstract): **javax.servlet.Servlet**,...
↳javax.servlet.http.HttpServlet (abstract)

  - javax.servlet.Servlet

3.- Server-side scripting

3.2.- Servlets

- Estructura básica de un servlet para HTTP

Estructura básica de u servlet para HTTP

```
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;//en proyectos JavaEE
import javax.servlet.http.*;

public class PrimerServlet extends HttpServlet
{
    //Atributos

    protected void doGet(HttpServletRequest request,HttpServletResponse response)
                                throws ServletException, IOException
    {
        //Código a ejecutar para HTTP-request GET
    }

    protected void doPost(HttpServletRequest request,HttpServletResponse response)
                                throws ServletException, IOException
    {
        //Código a ejecutar para HTTP-request POST
    }

    //Resto de métodos, propios o heredados (doHead,...)
}
```

  - Cualquier IDE de programación moderno ofrece plantillas

3.- Server-side scripting

3.2.- Servlets

- Estructura básica de un servlet para HTTP
  - Parámetros de métodos de servicio
    - **request**: Qué información?
      - **.getMethod()**
      - **.getHeader(“*cabecera_HTTP*”)**
      - Datos de usuario (por ejemplo, de formulario)
        - **.getParameter(“*nombre*”)**
        - **.getParameterValues(“*nombre*”)**
      - ...

```
Método path_o_URL versión_HTTP
Cabecera1:valor1
Cabecera2:valor2
…
línea _en_blanco
Cuerpo_del_mensaje
```

3.- Server-side scripting

3.2.- Servlets

- Estructura básica de un servlet para HTTP
  - Parámetros de métodos de servicio
    - **response**: Qué información?
      - **OBLIGATORIO?**
        - Código html
          - Escribir en objeto **PrintWriter**
          - **response.getWriter()**
        - Fichero binario
          - Escribir en objeto **ServletOutputStream**
          - **response.getOutputStream()**
      - OPCIONAL
        - Campos de la 1ª línea de HTTP-response
        - Cabeceras HTTP-response: **.setContentType(**"*mime*"**),** **.addCookie(**Cookie**),...**

```
versión_HTTP Código Explicación
Cabecera1:valor1
Cabecera2:valor2
…
línea_en_blanco
Cuerpo_del_mensaje
```

- Estructura básica de un servlet para HTTP
  - Esquema general de un método de servicio
    - Extraer información del request: zuál?
    - Acceder a recursos externos
    - Operar con datos
    - Escribir para response: cómo?
      - Cabeceras HTTP-response
      - Cuerpo HTTP-response
        - Código HTML o binario
        - Cerrar streams

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Servlet simple q genera código HTML sin preocuparse de la información del HTTP-request

Servlet simple

```
//imports

public class KaixoWWW extends HttpServlet //Servlet q no utiliza información del request
{
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
                       throws ServletException, IOException
  {
      PrintWriter out = response.getWriter(); //Para escribir código HTML en el HTTP-response
      out.println("<!DOCTYPE html>");
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Kaixo WWW</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Hello WWW</h1>");
      out.println("</body>");
      out.println("</html>");
      out.close();

      return;
  }
//El servidor web introduce las cabeceras content-type y content-length

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException
  {
      doGet(request,response);

      return;
  }
}
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Servlet simple mejorado con clase de apoyo (POJO)

Servlet simple mejorado con modularidad

```
//imports

public class KaixoWWW_MOD extends HttpServlet //Servlet q no utiliza información del request
{
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
                       throws ServletException, IOException
  {
      response.setContentType("text/html");//para código html no hace falta, pero es el estándar

      PrintWriter out = response.getWriter();
      out.println(UtilidadesServlets.DOCTYPE);
      out.println(UtilidadesServlets.headConTitle("KaixoWWW_MOD"));
      out.println("<body>");
      out.println("<h1>Hello WWW MODULAR</h1>");
      out.println(UtilidadesServlets.FIN);
      out.close();

      return;
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException
  {
     doGet(request,response);

     return;
  }
}
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Servlet simple mejorado con clase de apoyo (POJO)

Clase POJO para tareas comunes

```
public class UtilidadesServlets
{
  public static final String DOCTYPE = "<!DOCTYPE html>";

  public static String headConTitle(String tit)
  {
      String s="<html>";
      s+="\n<head>";
      s+="\n<title>"+tit+"</title>";
      s+="\n</head>";

      return s;
  }

  public static final String FIN = "</body>\n</html>";

}
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Aplicación Web con servlet y clases de apoyo (POJO)

Página html estática para recoger datos del usuario

```
<!DOCTYPE html>
<html>
<head>
<title>Página inicial de la aplicación aritmética</title>
<link rel="stylesheet" type="text/css" href="estilosAplicArit.css">
</head>
<body>
<h1>Introduzca los datos necesarios</h1>
<form method="post" action="ServletProcesadoFormularioYResultado">
  <p>
  <input type="number" name="operando1" maxlength="2" size="12" value="1er operando">
  <select name="operador">
  <option value="+">+</option>
  <option value="-">-</option>
  <option value="*">*</option>
  <option value="/">/</option>
  </select>
  <input type="number" name="operando2" maxlength="2" size="11" value="2° operando">
  </p>
  <p>
  <input type="submit" value="Enviar">
  </p>
</form>
</body>
</html>
```

EstilosAplicArit.css

```
h1{text-align:center;}
p{text-align:center;}
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Aplicación Web con servlet y clases de apoyo (POJO)

Servlet que procesa los datos introducidos en el formulario y genera HTML para visualizar resultado

```
//imports comunes
import logica.LogicaAplicacion;

public class ServletProcesadoFormularioYResultado extends HttpServlet
{
  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ...
  {
      PrintWriter out = response.getWriter();
      out.println(UtilidadesServlets.DOCTYPE);
      out.println(UtilidadesServlets.headConTitleStyle("Error de acceso","estilosAplicArit.css"));
      out.println("<body>");
      out.println("<h1>ERROR: este servlet sólo ha de accederse mediante POST</h1>");
      out.println(UtilidadesServlets.FIN);
       out.close();
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ...
  {
     String sOp1=request.getParameter("operando1");
     String sOp=request.getParameter("operador");
     String sOp2=request.getParameter("operando2");

     LogicaAplicacion la=new LogicaAplicacion(sOp1,sOp,sOp2);
     la.operar();

      PrintWriter out = response.getWriter();
      out.println(UtilidadesServlets.DOCTYPE);
      out.println(UtilidadesServlets.headConTitleStyle("Resultado","estilosAplicArit.css"));
      out.println("<body>");
      out.println("<h1>RESULTADO DE LA OPERACIÓN SOLICITADA</h1>");
      out.println("<p>"+sOp1+sOp+sOp2+"="+la.getRes()+"</p>");
      out.println(UtilidadesServlets.FIN);
      out.close();
  }
}
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Aplicación Web con servlet y clases de apoyo (POJO)

Clase POJO para tareas comunes

```
public class UtilidadesServlets
{
  public static final String DOCTYPE = "<!DOCTYPE html>";

  public static String headConTitleStyle(String tit,String styleSheetFileName)
  {
      String s="<html>";
      s+="\n<head>";
      s+="\n<title>"+tit+"</title>";
      s+="\n<link rel=\"stylesheet\" type=\"text/css\" href="+styleSheetFileName+">";
      s+="\n</head>";

      return s;
  }

  public static final String FIN = "</body>\n</html>";

}
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Aplicación Web con servlet y clases de apoyo (POJO)

Clase POJO para lógica de la aplicación

```
package logica;

public class LogicaAplicacion
{
  private float op1;
  private char op;
  private float op2;
  private float res;

  public LogicaAplicacion(String sOp1,String sOp,String sOp2)
  {
      op1=Float.valueOf(sOp1);
      op=sOp.charAt(0);
      op2=Float.valueOf(sOp2);
  }

  public void operar()
  {
      switch(op)
      {
            case '+':
                  res=op1+op2;
                  break;

//Resto de casos posibles
      }

      return;
  }

  public float getRes()
  {
      return res;
  }
}
```

- Gestión de sesión con servlets
  - HTTP no mantiene estado
  - El servidor de servlets puede mantener información de sesión en memoria y asociarla al HTTP-request mediante
    -
    -
  - Acceso a información de sesión en servlets
    - Objeto **HttpSession**: **request.getSession()**
    - Identificador de sesión:
    - Insertar/extraer información:
    - Finalizar sesión:
  - Navegación: a través de enlaces preparados en los servlets
    - **response.encodeURL(“***url_a_enlazar***”)**

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Sesión

Servlet1: Crea Sesión

```
public class Servlet1 extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      HttpSession s = request.getSession();

      if (s != null) {
            s.setAttribute("contador", "0");
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            out.println(UtilidadesServlets.DOCTYPE);
            out.println(UtilidadesServlets.headConTitleStyle("Servlet1",
                        "estilosAplicacionAritmetica.css"));
            out.println("<body>");
            out.println("<h1>Se ha creado una nueva sesión: " + s.getId() + "</h1>");

            String urlNoCookie = response.encodeURL("Servlet2");
            out.println("<form action=\"" + urlNoCookie + "\" method=\"get\">");
            out.println("<p><input type=\"submit\" value=\"Comience el recorrido\"></p>");
            out.println("</form>");
            out.println("<a href=\"" + urlNoCookie + "\" target=\"_blank\">Siguiente</a>");

            out.println(UtilidadesServlets.FIN);
            out.close();
      } else
            System.out.println("NO SE PUDO INICIAR LA SESION");
  }
// ...
```

3.- Server-side scripting

3.2.- Servlets

- EJEMPLOS SERVLETS: Sesión

Servlet2: Usa sesión

```
public class Servlet2 extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      HttpSession s = request.getSession();

      if (s != null && !s.isNew()) {
            try {
                  int c = Integer.parseInt((String) s.getAttribute("contador"));
                  String[] inc = request.getParameterValues("incrementos");
                  for (int i = 0; i < inc.length; i++)
                        c += Integer.parseInt(inc[i]);
                  s.setAttribute("contador", "" + c);
            } catch (Exception ex) {
                  System.out.println("S2: " + ex.getStackTrace());
            }

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            out.println(UtilidadesServlets.DOCTYPE);
            out.println(UtilidadesServlets.headConTitleStyle("Servlet2",
                        "estilosAplicacionAritmetica.css"));
            out.println("<body>");
            out.println("<h1>Seguimos en la misma sesión: " + s.getId() + "</h1>");
            out.println("<h1>Valor actual del contador: " + s.getAttribute("contador") + "</h1>");

            String urlNoCookie = response.encodeURL("Servlet2");
            out.println("<form action=\"" + urlNoCookie + "\" method=\"get\">");
            out.println("<p>Introduzca en cuanto quiere incrementar el contador:</p>");

            out.println("<p>+1<input type=\"checkbox\" name=\"incrementos\" value=\"1\"></p>");
            out.println("<p>+2<input type=\"checkbox\" name=\"incrementos\" value=\"2\"></p>");
            out.println("<p>+3<input type=\"checkbox\" name=\"incrementos\" value=\"3\"></p>");
// ...
```

- EJEMPLOS SERVLETS: Sesión

Servlet3: Cierra sesión

```
public class Servlet3 extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      HttpSession s = request.getSession();

      if (s != null && !s.isNew()) {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            out.println(UtilidadesServlets.DOCTYPE);
            out.println(UtilidadesServlets.headConTitleStyle("Servlet3",
                        "estilosAplicacionAritmetica.css"));
            out.println("<body>");
            out.println("<h1>Fin de la sesión: " + s.getId() + "</h1>");
            out.println("<h1>Valor actual del contador: "
                        + s.getAttribute("contador") + "</h1>");

            s.invalidate();

            out.println(UtilidadesServlets.FIN);
            out.close();
      } else
            System.out.println("NO HAY SESION o LA SESIÓN NO SE HA MANTENIDO");
  }
// ...
```

4.- Patrones de diseño

4.1.- Introducción

- Definición
- Importancia
- Tipos de patrones
  - Creación
  - Estructurales
  - Comportamiento
  - Concurrencia
  - Arquitectura
- Conceptos generales
  - Cada lenguaje su implementación específica

4.- Patrones de diseño

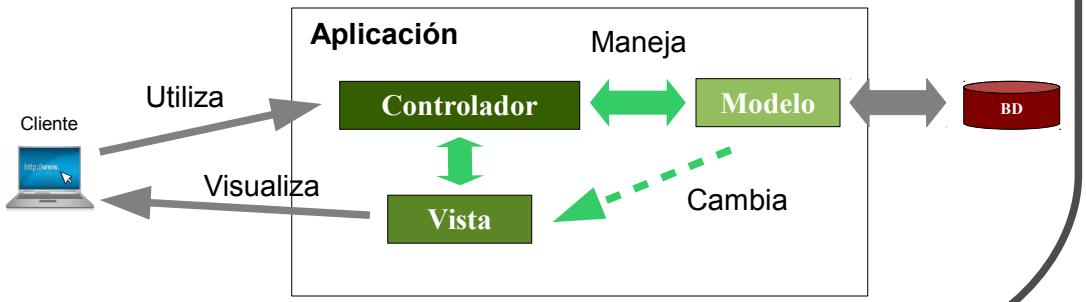
4.2.- POJO y Java Beans

- POJO (Plain Old Java Object)
  - Clases Java “normales”
  - Sin obligaciones de:
    - ...
- Java Bean
  - POJO Serializable
  - Condiciones:
    - ...
    - ...
    - ...

4.- Patrones de diseño

4.3.- MVC

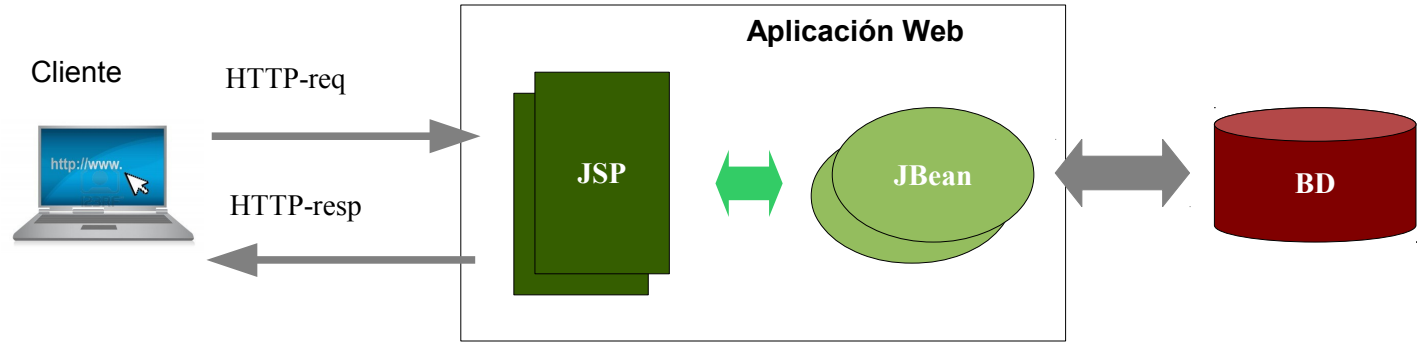
- Muy utilizado para implementar interfaces de usuario
- Componentes
  - Modelo
    - ...
    - ...
  - Vista
    - ...
    - ...
  - Controlador
    - ...



4.- Patrones de diseño

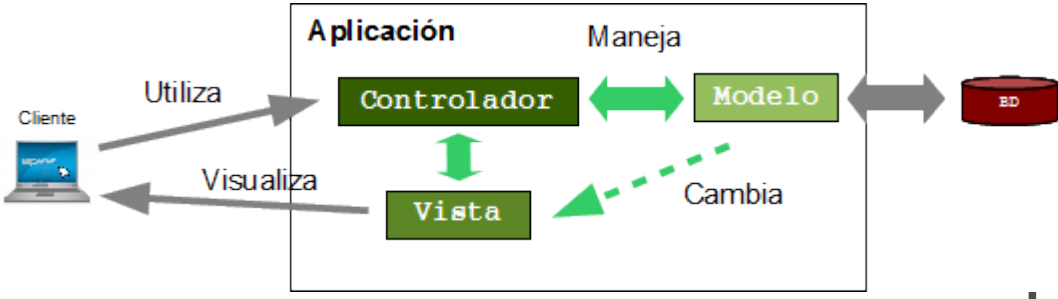
4.4.- Ejemplos de patrones utilizados en JavaEE

- Model 1
  - Aplicaciones web muy sencillas
  - JavaBeans
    - Información para crear el cuerpo del response
  - Problemático para aplicaciones complejas
    - JSPs: mezcla de HTML y scriptlets
    - Navegación descentralizada



4.- Patrones de dise

4.4.- Ejemplos de patrones utilizad



- Model2
  - Aplicaciones web complejas: caso común
  - Adaptación del patrón MVC
  - Componentes:
    - Modelo: JavaBeans, EJBs, POJOs
    - Vista: JSFs (inicialmente JSPs)
    - Controlador: Servlet (FacesServlet)