

Tecnologías Web de cliente y servidor

Laboratorio de Aplicaciones Telemáticas

Jesús Arias Fisteus

jaf@it.uc3m.es

Curso 2007/2008

Universidad Carlos III de Madrid

Necesidad de complementar HTML

- Contenido dinámico:
 - Personalizar el contenido.
 - Acceder a bases de datos.
 - Interaccionar con aplicaciones.
 - Mejorar la presentación.
 - Mejorar la interactividad.
- Multimedia.
- Soporte para nuevos tipos de datos.
- Soporte para nuevos protocolos.

Alternativa cliente / servidor

- Ejecución en el cliente:
 - El navegador descarga el código del servidor.
 - El navegador ejecuta en código en la máquina cliente.
- Ejecución en el servidor:
 - El código se ejecuta en el entorno del servidor.
 - Genera dinámicamente los datos que se envían al cliente.
- Soluciones híbridas:
 - Parte del código se ejecuta en el servidor, y parte del código en el cliente.

Comparativa cliente / servidor (I)

- Escalabilidad:
 - La ejecución en cliente es más escalable, porque distribuye en uso de recursos (CPU, memoria) entre todos los clientes.
- Seguridad:
 - La ejecución en el cliente proporciona menor seguridad para el cliente, porque debe ejecutar código potencialmente malicioso o defectuoso.

Comparativa cliente / servidor (II)

- Compatibilidad / accesibilidad:
 - La ejecución en el cliente impide el uso de navegadores no compatibles con el código, especialmente desde dispositivos de capacidad limitada.
 - Comportamiento distinto, ocasionalmente, del mismo programa en distintos navegadores.

Tecnologías de ejecución en el cliente

- ECMAScript / JavaScript.
- Applets Java.
- Plug-ins.
- Controles ActiveX.
- Adobe Flash.
- AJAX.

ECMAScript / JavaScript

- Características:
 - Sintaxis semejante a C y Java, tipado débil sin clases ni herencia.
 - Ejecución interpretada por el navegador.
 - Sometido a restricciones de seguridad.
 - Facilidad para modificar dinámicamente el documento HTML mediante DOM.
- Algunos usos habituales:
 - Responder a eventos.
 - Comprobar datos de un formulario antes de enviarlos.

ECMAScript / JavaScript

- Limitaciones:
 - Lenguaje de programación no estructurado, no apto para aplicaciones demasiado grandes.
 - Comportamiento dependiente del navegador.
 - Algunos navegadores no lo ejecutan.
 - Código fuente disponible para el cliente.

Applets Java

- Programas Java que se ejecutan en el navegador:
 - El cliente descarga las clases (*bytecodes*).
 - El cliente ejecuta el código en una JVM.
 - Se ejecuta con restricciones de seguridad.
 - Incrustado en navegador o en ventana propia.
- Algunos usos habituales:
 - Aplicaciones gráficas interactivas.
 - Animaciones.
 - Gestión de formularios complejos.

Applets Java

- Algunos usos habituales (más):
 - Implementación de protocolos desconocidos por el navegador.
 - Comunicación con aplicaciones y bases de datos: HTTP, *sockets*, RMI, CORBA, etc.
 - Limitaciones:
 - Navegadores sin JVM o con distintas versiones.
 - Menor interacción con el documento HTML que JavaScript.

Plug-ins

- Programas en código nativo que extienden la funcionalidad del navegador.
- Usos habituales:
 - Visualizar o interactuar con nuevos tipos de datos.
 - Acceder a funcionalidades básicas del navegador (carga de URLs, etc.)
 - Implementar nuevos protocolos de comunicación (por ejemplo, *streaming* multimedia).
 - Audio, vídeo, juegos, animaciones, 3D, etc.

Plug-ins

- Ejemplos:
 - Java, Flash, RealPlayer.
- Limitaciones:
 - Dependencia con navegador, sistema operativo y arquitectura de la plataforma.
 - Seguridad: acceso libre a los recursos de la máquina.
 - Necesidad de instalar el *plug-in* previamente.
 - Programación compleja.

AJAX

- Acrónimo de *Asynchronous JavaScript and XML*.
- Aplicaciones Web ejecutadas en el cliente con comunicación asíncrona con el servidor en segundo plano, basada en XML.
- Aumenta significativamente la interactividad y velocidad de la aplicación.
- Puede reducir la usabilidad y accesibilidad.

AJAX

- Combinación de tecnologías existentes previamente:
 - XHTML y CSS para presentación.
 - JavaScript para ejecución en cliente. Modifica el documento XHTML mediante DOM.
 - Comunicación asíncrona con el servidor mediante XMLHttpRequest.
 - La comunicación asíncrona suele basarse en intercambio de documentos XML.

Otras tecnologías de cliente

- ActiveX (Microsoft)
 - Controles incrustables en páginas Web.
 - Requiere instalación.
 - Riesgos de seguridad elevados.
 - Sólo para MS–Windows.
- Adobe Flash:
 - Multimedia, animaciones, juegos, 3D, etc.
 - Lenguaje de programación ActionScript.

Tecnologías de ejecución en el servidor

- Common Gateway Interface (CGI).
- Sistemas de plantillas:
 - ColdFusion.
 - Active Server Pages (ASP / ASP.NET).
 - Server–Side JavaScript (SSJS).
 - PHP.
 - JSP.
- Servlets.
- Python.

Common Gateway Interface (CGI)

- Interfaz común entre servidor Web y programas externos.
- Funcionamiento:
 1. El servidor recibe petición y detecta si se corresponde a un programa externo (CGI).
 2. El servidor crea un nuevo proceso para el programa; le pasa los datos de la petición en variables de entorno y entrada estándar.
 3. El programa procesa la petición y devuelve la respuesta por salida estándar.
 4. El servidor reenvía la respuesta al cliente.

Common Gateway Interface (CGI)

- Limitaciones:
 - Poco escalable / eficiente:
 - Nuevo proceso y entorno de ejecución para cada petición; datos de sesión en BD o disco.
 - Cálculo de todas las variables de entorno, aunque no las use el programa.
 - Para lenguajes interpretados, tiempo de carga del intérprete.
 - Código de programación entremezclado con código HTML.

Common Gateway Interface (CGI)

```
#!/usr/local/bin/perl
use CGI;

$query = new CGI;
$nombre= $query->param( 'nombre' );
if ( $nombre eq " " ) { $nombre="Mundo"; }

print "Content-type: text/html\n\n";
print <<"EOF";
<html>
  <head>
    <title>Hola Mundo CGI (Perl)</title>
  </head>
  <body>
    <h1>Hola, $nombre</h1>
  </body>
</html>
EOF
```

ASP / ASP.NET (Microsoft)

- ASP nace como sistema de plantillas.
- ASP.NET evoluciona hacia:
 - Entornos orientados a eventos.
 - Programación en cualquier lenguaje integrable en .NET.
 - Mayor eficiencia (DLLs cargadas en el servidor Web).
- Otras características:
 - Acceso a componentes ActiveX.
 - Integración fácil con productos Microsoft.

ASP / ASP.NET (Microsoft)

```
<%@ Page Language="VB" %>
<script runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)
        HelloWorld.Text = "Hello World!"
    End Sub
</script>

<html>
<head>
<title>ASP.NET Hello World</title>
</head>
<body bgcolor="#FFFFFF">

<p><asp:label id="HelloWorld" runat="server" /></p>

</body>
</html>
```

PHP

- Código libre.
- Lenguaje de *scripting* con sintaxis basada en C.
- Portado a las plataformas más extendidas.
- Integrable con los servidores más utilizados.
- Acceso potente a bases de datos.
- Comunicación por red con otros recursos: correo electrónico, servidores de directorio, etc.
- Eficiente.

PHP

```
<html>
  <head>
    <title>Hola Mundo PHP</title>
  </head>
  <body>
    <p>
      <?
        // Hello world in PHP
        print("Hello World");
      ?>
    </p>
  </body>
</html>
```

Servlets (Java)

- Programas Java que añaden funcionalidad en servidores.
- Modelo de ejecución:
 - El servidor carga al inicio una JVM.
 - Se instancia un objeto por cada Servlet.
 - Para cada petición se crea / reutiliza un hilo y se invoca un método la instancia cargada del Servlet asociado.
 - Cuando se para el servidor se destruyen los objetos.

Servlets (Java)

- Principales características:
 - Funcionalidad semejante a CGI: recibe una petición a través del servidor y genera la respuesta adecuada.
 - Comunicación con el servidor HTTP mediante *Servlet API*
 - Respuesta mediante `PrintWriter`.

Servlets (Java)

- Ventajas:
 - Portabilidad: arquitecturas, sistemas operativos y servidores HTTP.
 - Potencia: disponibles todos los recursos Java: APIs, bases de datos, comunicaciones, etc.
 - Elegancia: código fuente limpio, simple y orientado a objetos.
 - Integración: el servlet está integrado con el servidor, colaborando con éste más fácilmente que CGI.
 - Extensibilidad y flexibilidad.

Servlets (Java)

- Más ventajas:
 - Eficiencia:
 - Máquina virtual permanentemente cargada en el servidor.
 - El servlet se instancia sólo una vez durante el tiempo de vida del servidor.
 - Creación de hilos más eficiente que procesos.
 - Acceso a recursos compartido.
 - Mantenimiento de sesión en memoria.

Servlets (Java)

- Desventajas:
 - Concurrencia: por ser un único objeto ejecutado desde varios hilos, es necesario gestionar la concurrencia.
 - Código del programa mezclado con código HTML del documento.
 - Cualquier cambio requiere recompilar el servlet.
 - Bastante limitado a lenguaje Java.

Servlets (Java)

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

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");

        PrintWriter out = res.getWriter();

        out.println("<html>");
        out.println("<head><title>Hola Mundo Servlet</title></head>");
        out.println("<body>");
        out.println("<p>Hello World</p>");
        out.println("</body></html>");
    }
}
```

Java Server Pages (JSP)

- Código insertado en el documento HTML en marcas especiales:
 - Directivas.
 - Elementos de *scripting*: declaraciones, expresiones, *scriptlets*.
 - Comentarios.
 - Acciones.
- Objetos implícitos: petición, respuesta, sesiones, etc.
- Utilización de *JavaBeans*.
- Normalmente compilado a servlets.

Java Server Pages (JSP)

- Misma portabilidad, eficiencia y potencia que Servlets.
- Elegancia: separación del código del documento del código de la aplicación (*JavaBeans*).
- Facilidad de programación: gestiona automáticamente sesiones, atributos de peticiones y respuestas, redirecciones, ciclo de vida de *JavaBeans*, etc.
- Integrable con Servlets.

Java Server Pages (JSP)

```
<html>
  <head>
    <title>Hola Mundo JSP</title>
  </head>
  <body>
    <p>
      <% String visitor = request.getParameter("name");
        if (visitor == null) visitor = "World";
        %>
      Hola, <%= visitor %>!
    </p>
  </body>
</html>
```

Envío de datos de formularios

Envío de datos de formularios

- Métodos y codificaciones de envío:
 - Método GET con codificación *application/x-www-form-urlencoded*.
 - Método POST con codificación *application/x-www-form-urlencoded*.
 - Método POST con codificación *multipart/form-data*.

Método GET

- Los parámetros se envían en la línea de petición del mensaje HTTP, concatenados a la cadena de ruta (*path*).
- Parámetros visibles para el usuario en el navegador.
- Permite enviar datos tanto desde formularios como desde hiperenlaces.
- No adecuado para operaciones idempotentes.
- No adecuado para el envío de ficheros.

Método POST con *URL-encoded*

- Los parámetros se envían en el cuerpo de la petición HTTP.
- Parámetros no visibles en la URL.
- Únicamente se pueden enviar desde formularios.
- No adecuado para el envío de ficheros.

Método POST con *multipart*

- Los parámetros se envían en el cuerpo de la petición HTTP al estilo de MIME.
- Parámetros no visibles en la URL.
- Cada parámetro es una parte del cuerpo, separado del resto por cadenas de frontera.
- Únicamente se pueden enviar desde formularios.
- Adecuado para el envío de ficheros.

Gestión de sesiones



Sesiones

- HTTP no tiene estado.
- Necesidad de relacionar peticiones entre sí.
- Concepto de sesión.

Mecanismos de gestión de sesiones

- Campos ocultos de formularios.
- Reescritura de URL:
 - Cadena de ruta (*path*) añadida.
 - Parámetro añadido.
 - A medida.
- *Cookies*.