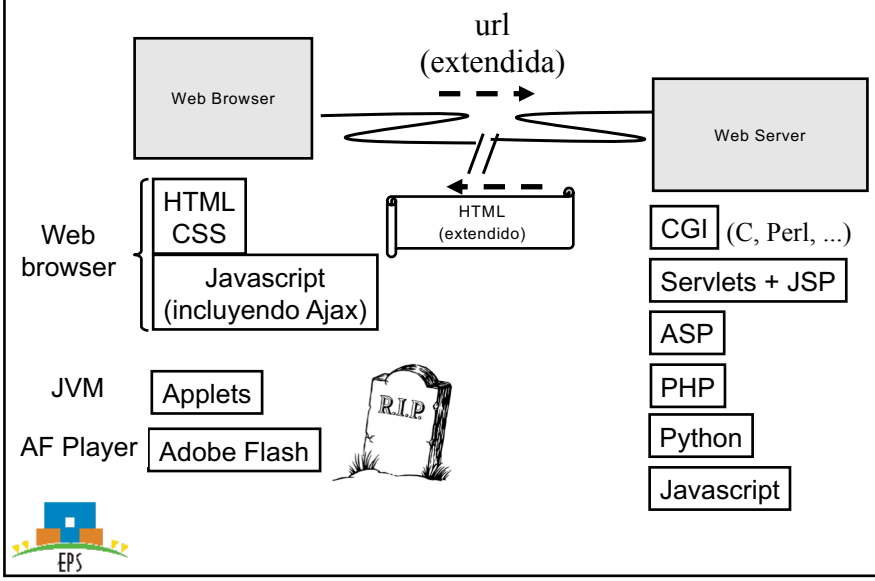
Sistemas Informáticos

Tema 2: Sistemas distribuidos basados en WWW

2.3 Web interactiva (Aplicaciones Web)

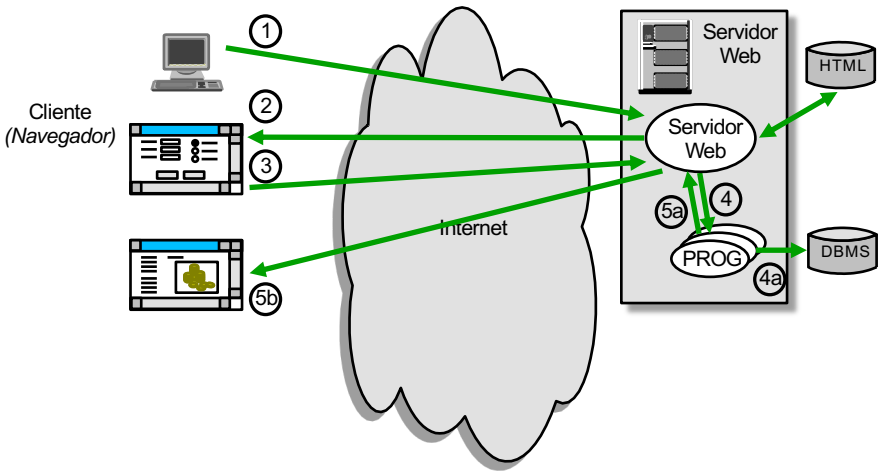
Evolución de los contenidos en la WWW



Web interactiva

- El modelo *Web* hipertexto no permite más interacción del usuario que seleccionar URLs.
- Necesario establecer comunicación entre programas del servidor con datos que proporciona el usuario.
  - Datos introducidos mediante formularios *(Forms).*
  - Ejecución de programas en el servidor mediante por ejemplo *Common Gateway Interface, CGI*.

Funcionamiento básico (I)



PROG = PROGRAMA que se ejecuta en el lado servidor

Funcionamiento básico (II)

1. El cliente solicita una página al servidor Web.
2. El servidor envía una página que contiene un formulario.
3. El cliente rellena el formulario y lo envía al servidor.
4. El servidor, por la URL que se solicita, decide ejecutar un programa, al que pasa los datos recibidos del cliente.
   a. El programa puede realizar operaciones normales, tales como lectura de ficheros o accesos a bases de datos.
5. El programa genera una página con los resultados de su ejecución, y la devuelve al cliente a través del servidor Web.

Toda la comunicación se produce mediante protocolo HTTP y páginas HTML.

Formularios HTML

- Definidos mediante el elemento <FORM>
- Contenedor que agrupa los elementos de entrada de datos, mezclados con otros elementos estándar HTML.
- Estructura:

```
<FORM ACTION=url
      METHOD=método de envío>
<INPUT> | <TEXTAREA> | <SELECT> | <BUTTON>
otros elementos HTML
</FORM>
```

Programa que atiende la petición en lado servidor

- *Action*: URL que se debe solicitar al enviar el formulario.
- *Method*: Método de envío de los datos asociados al formulario.

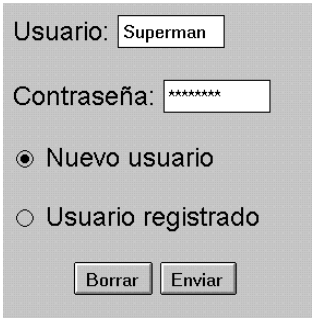
GET o POST

Definición de elementos en los formularios

- Mediante definición en HTML de elementos de interacción.
  - INPUT: *text, password, checkbox, radio, hidden, image, submit, reset.*
  - SELECT: Listas de selección con múltiples valores.
  - TEXTAREA: Campo de texto multilínea.
  - BUTTON: Botones de acción.
- A cada elemento se le asocia un nombre.
- La interacción con el usuario produce un valor para el mismo.
- Los resultados se envían al servidor en la petición HTTP mediante parejas `<nombre>=<valor>`.
  - Según el método elegido para el formulario, viajan asociados a la URL que se pide o en el cuerpo del mensaje HTTP.

Formularios - Ejemplo

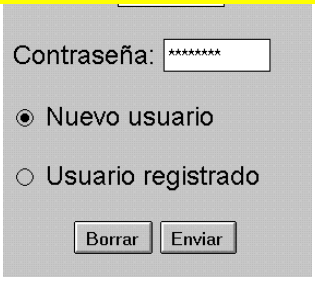
```
<FORM ACTION="http://host/cgi-bin/logon" METHOD=GET>
<p>Usuario:
<INPUT NAME="usuario" TYPE=text
       SIZE="10" MAXLENGTH="8">
<p>Contrase&ntilde;a:
<INPUT NAME="clave" TYPE=password
       SIZE="10" MAXLENGTH="8">
<P><INPUT NAME="nuevo" TYPE=radio
          VALUE=si CHECKED>
Nuevo usuario
<P><INPUT NAME="nuevo" TYPE=radio
          VALUE=no>
Usuario registrado
<CENTER><P>
<INPUT TYPE=RESET VALUE="Borrar">
<INPUT TYPE=SUBMIT VALUE="Enviar">
</CENTER>
Pruduce el envío de la siguiente URL:
   http://host/cgi-bin/logon?usuario=Superman&clave=loislane&nuevo=si
```



Formularios - GET

En caso de METHOD=GET
El comando HTTP sería:
GET /logon?usuario=Superman&clave=loislane&nuevo=si HTTP/1.1
(otras cabeceras HTTP)
(línea en blanco)

```
<FORM ACTION="htt
<p>Usuario:
<INPUT NAME="usua
       SIZE="10"
<p>Contrase&ntilc
<INPUT NAME="clave" TYPE=password
       SIZE="10" MAXLENGTH="8">
<P><INPUT NAME="nuevo" TYPE=radio
          VALUE=si CHECKED>
Nuevo usuario
<P><INPUT NAME="nuevo" TYPE=radio
          VALUE=no>
Usuario registrado
<CENTER><P>
<INPUT TYPE=RESET VALUE="Borrar">
<INPUT TYPE=SUBMIT VALUE="Enviar">
</CENTER>
```



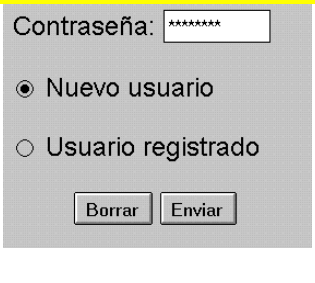
Pruduce el envío de la siguiente URL:

```
   http://host/cgi-bin/logon?usuario=Superman&clave=loislane&nuevo=si
```

Formularios - POST

En caso de METHOD=POST, el mensaje habría sido:
POST /logon HTTP/1.1
CONTENT-TYPE: **application/x-www-form-urlencoded**
CONTENT-LENGT: xxx
(otras cabeceras HTTP)
(línea en blanco)
usuario=Superman&clave=loislane&nuevo=si

```
<FORM ACTION="htt
<p>Usuario:
<INPUT NAME="usua
       SIZE="10"
<p>Contrase&ntilc
<INPUT NAME="clav
       SIZE="10" MAXLENGTH="8">
<P><INPUT NAME="nuevo" TYPE=radio
          VALUE=si CHECKED>
Nuevo usuario
<P><INPUT NAME="nuevo" TYPE=radio
          VALUE=no>
Usuario registrado
<CENTER><P>
<INPUT TYPE=RESET VALUE="Borrar">
<INPUT TYPE=SUBMIT VALUE="Enviar">
</CENTER>
```



Pruduce el envío de la siguiente URL:

```
   http://host/cgi-bin/logon?usuario=Superman&clave=loislane&nuevo=si
```

Common Gateway Interface, CGI

- EL CGI define el método para que un servidor WWW pueda ejecutar programas externos y recoger información de ellos.
- Es una extensión del protocolo HTTP.
- El programa externo recibe del servidor información:
  - Asociada a la transmisión: Origen, URL, protocolo utilizado...
  - Introducida por el usuario, en un formulario de entrada.
- El paso de información a través del CGI se realiza mediante:
  - Variables de entorno.
  - Línea de comandos.
  - Entrada / salida estándar (*stdin* y *stdout*).

Variables de entorno

- Un programa CGI recibe información del servidor HTTP por medio de variables de entorno.
- Los nombres de las variables de entorno pueden ser específicos del sistema.
  - Variables de entorno no dependientes de la petición:
    - SERVER_SOFTWARE
    - SERVER_NAME
    - GATEWAY_INTERFACE
  - El resto de las variables de entorno dependen de la consulta:
    - SERVER_PROTOCOL
    - SERVER_PORT
    - REQUEST_METHOD
    - PATH_INFO
    - PATH_TRANSLATED.
    - SCRIPT_NAME
    - QUERY_STRING
    - REMOTE_HOST
    - REMOTE_ADDR
  - Variables de entorno relacionadas con la seguridad de acceso:
    - AUTH_TYPE
    - REMOTE_USER
    - REMOTE_IDENT
  - Variables de entorno sobre la información adicional asociada a la petición:
    - CONTENT_TYPE
    - CONTENT_LENGTH
  - Variables de entorno de la cabecera HTTP. Formato:
    - HTTP_nombre_variable_cabecera

Ventajas e inconvenientes CGI

- Ventajas:
  - Sencillez de programación.
  - Uso de cualquier lenguaje de programación, incluso lenguajes interpretados (típicamente PERL).
  - El programa CGI no puede afectar el funcionamiento del servidor, por ejecutarse como un proceso independiente.
  - Estándar. Garantiza la portabilidad entre servidores de distintos fabricantes.
- Inconvenientes:
  - **Lento**. Cada ejecución requiere crear un proceso y finalizarlo, que implica reservas de memoria, aperturas de ficheros, conexiones a bases de datos, etc.
  - El programa CGI termina con cada llamada. No se puede mantener un estado de la comunicación entre peticiones (sesión).

Mantenimiento de la sesión en CGIs

Mediante campos ocultos en los formularios:

- El usuario rellena un formulario con una serie de datos, que envía al servidor.
- El programa CGI que los recibe genera una página HTML en la que introduce un nuevo formulario para pedir más datos.
  - Almacena también los datos del formulario anterior como campos ocultos *(hidden)*.
- El usuario, al enviar el nuevo formulario, proporciona al programa CGI los datos nuevos y los de la petición anterior.

Web Application Programming Interfaces

- Surgen para tratar de evitar el problema de bajo rendimiento de la interfaz CGI.
  - Los nuevos programas se enlazan junto con el servidor en una librería dinámica.
  - El servidor llama a las funciones de librería como tareas dentro del propio proceso servidor.
  - El proceso servidor no finaliza: Se mantienen ficheros abiertos, conexiones a bases de datos, etc. entre llamadas a funciones.
  - Se proporciona una API de acceso a los datos y estado del servidor.
- Inconvenientes:
  - Un fallo en una rutina puede hacer caer el servidor completo.
  - Lenguajes de programación normalmente limitados a C y C++.
  - Difícil de programar. Es preciso conocer el funcionamiento interno del servidor para aprovecharla a máximo.
- Proporcionadas por casi todos los fabricantes: Netscape (NSAPI), Microsoft (ISAPI), IBM (ICAPI, GWAPI)...

Interfaces híbridas

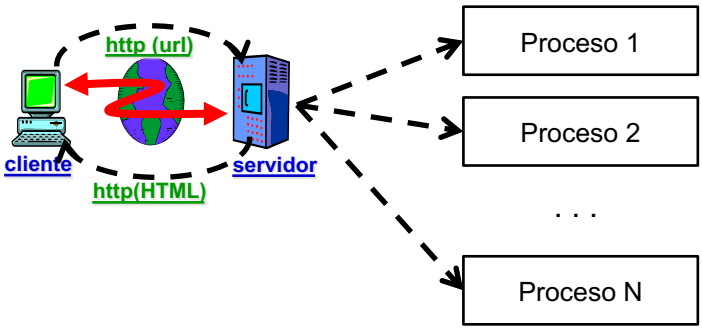
- Intentan conseguir las ventajas de CGIs y Web APIs evitando sus inconvenientes.
- Los programas se realizan de modo independiente al servidor Web, y en cualquier lenguaje.
- El servidor Web, durante su inicialización, puede arrancar los programas en procesos independientes.
- Los programas arrancados, tras inicializarse, quedan a la espera de recibir peticiones.
  - Todo el proceso de inicialización se realiza antes de recibir una petición.
- La comunicación mediante variables de entorno y *stdin* y *stdout* se sustituye por otro mecanismo de comunicación entre procesos, más rápido.
  - Puede permitir acceso remoto empleando un mecanismo de comunicación apropiado.
  - Empleando los mismos elementos que en CGI se consigue facilidad de migración de programas CGI a las nuevas interfaces.
- Tras atender una petición, el programa no finaliza: vuelve a esperar la siguiente petición.
  - Es posible mantener el estado de la aplicación entre peticiones sucesivas.
- Actualmente siguen este modelo:
  - FastCGI, de Open Market, Inc. Comunicación Servidor - Programas por Sockets.
  - Netscape *Web Application Interface (WAI)*. Comunicación mediante CORBA.

Páginas dinámicas

- Extensiones del lenguaje HTML para permitir mayor capacidad de proceso:
  - En el cliente *(client side scripts)*: Inclusión de código que el cliente interpretará para variar dinámicamente la presentación de la página.
    - Proporciona "inteligencia" en el navegador.
  - En el servidor *(server side scripts)*:
    - Inclusión de código en el fichero que contiene la descripción de la página.
    - El servidor lo interpretará para variar la generación de la página antes de su envío al cliente.
    - Alternativa a la programación CGI.

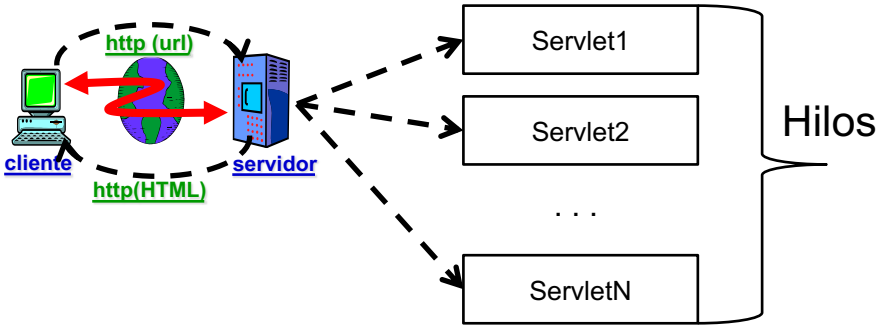
Modelos de implementación en servidor

- CGIs



Modelos de implementación en servidor

- JEE



Modelos de implementación en servidor

- En PHP, depende del servidor web asociado.
- Con Apache hay dos modelos posibles:
  - Multi-proceso: también llamado pre-forked, porque al iniciarse el servidor crea un pool de procesos que son reusados para atender las distintas peticiones

StartServers	5
MinSpareServers	5
MaxSpareServers	10
MaxClients	150
MaxRequestsPerChild	0

Máxima cantidad de procesos a crear

Modelos de implementación en servidor

- En PHP, depende del servidor web asociado.
- Con Apache hay dos modelos posibles:
  - Multi-hilo (workers) : aunque el motor PHP está preparado para funcionar correctamente con múltiples hilos, hay muchas bibliotecas adicionales que no lo están.

StartServers	2 ← procesos
MaxClients	150
MinSpareThreads	25
MaxSpareThreads	75
ThreadsPerChild	25 ← Hilos x proceso
MaxRequestsPerChild	0

Modelos de implementación en servidor

- Modelo alternativo: asíncrono.
- Es el modelo implementado por el sistema **NodeJS**:
  - Sistema para poder programa los servidores web en JavaScript.
  - **Un hilo** (y proceso, por supuesto)
  - Llamadas no bloqueantes y programación basada en eventos
    - Por ejemplo, si quiero leer un fichero (para responder a petición http), invoco a la función de lectura, dándole el nombre de otra función (callback) y sigo ejecutando (posiblemente atendiendo otras peticiones).
    - Cuando se termina de leer el fichero (se produce el evento), el sistema invoca automática la función callback.

Sistemas Informáticos

Tema 2.3: Web Interactiva

El lenguaje PHP

Introducción a PHP

- **Lenguaje de *script***
  - PHP = **P**HP **H**ypertext **P**reprocesor.
  - PHP es un lenguaje de *script* del lado del servidor. Otros lenguajes similares son ASP o JSP
  - Los scripts PHP están incrustados en los documentos HTML y el servidor los interpreta y ejecuta antes de servir las páginas al cliente.
  - El cliente no ve el código PHP sino los resultados que produce.
  - Software abierto y gratuito
  - PHP + [PostgreSQL, MySQL] es multiplataforma

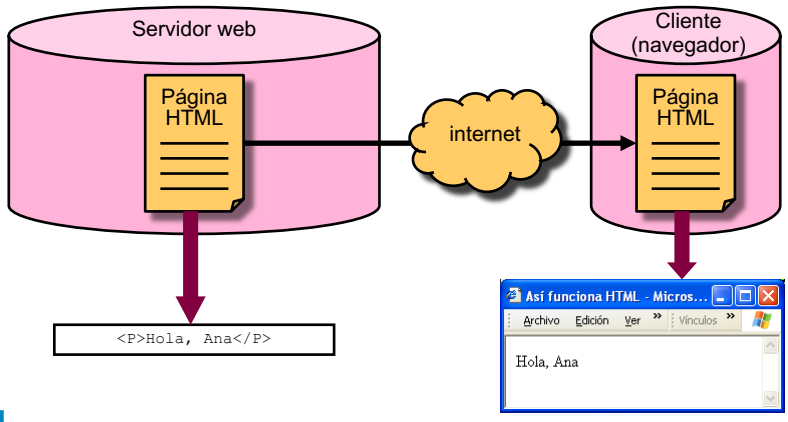
Ejemplo PHP

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Ejemplo.**php**

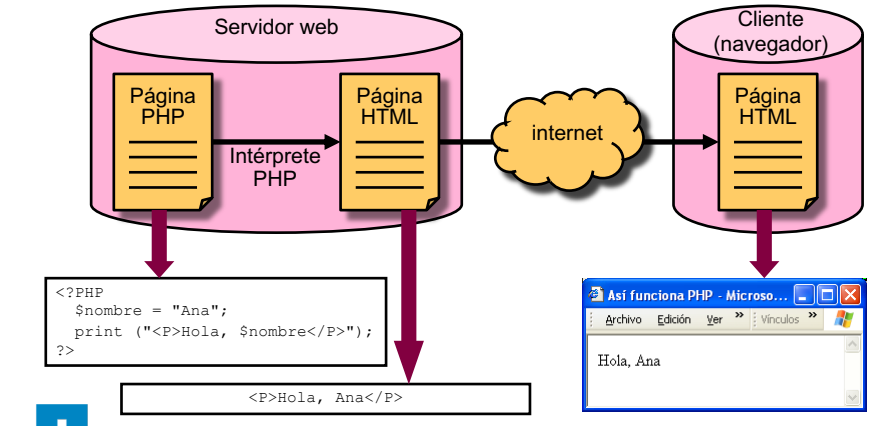
Introducción a PHP

- **¿Cómo funciona PHP? (1)**



Introducción a PHP

- ¿Cómo funciona PHP? (2)



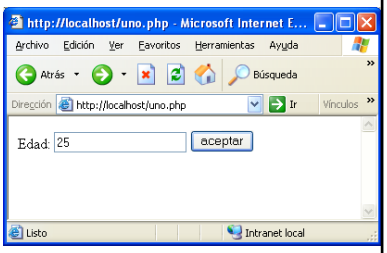
La función $_REQUEST

```
Welcome <?php echo $_REQUEST["fname"]; ?>!<br />
You are <?php echo $_REQUEST["age"]; ?> years old.
```

Acceso a formularios desde PHP

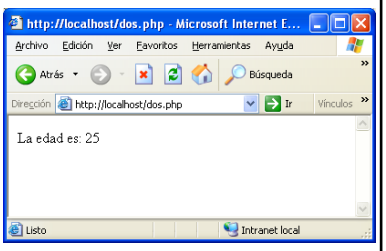
uno.html

```
<HTML>
<BODY>
<FORM ACTION="dos.php" METHOD="POST">
  Edad: <INPUT TYPE="text" NAME="edad">
  <INPUT TYPE="submit" VALUE="aceptar">
</FORM>
</BODY>
</HTML>
```



dos.php

```
<HTML>
<BODY>
<?PHP
 $edad = $_REQUEST['edad'];
  print ("La edad es: $edad");
?>
</BODY>
</HTML>
```



El formulario de PHP

- La forma habitual de trabajar con formularios en PHP es utilizar un único programa que procese el formulario o lo muestre según haya sido o no enviado, respectivamente
- Ventajas:
  - Disminuye el número de ficheros
  - Permite validar los datos del formulario en el propio formulario
- Procedimiento:

```
si se ha enviado el formulario:
   Procesar formulario
si no:
   Mostrar formulario
fsi
```

Sistemas Informáticos

Cookies y Sesiones

Cookies

- Una forma de identificar al usuario.
- Es un pequeño documento que el servidor (con el permiso del navegador) introduce en el ordenador del cliente.
- Cada vez que el navegador solicite una nueva página (a ese servidor) envía también la cookie.

Cookies: ejemplo sintaxis PHP

- Antes de <html>
  - setcookie(name, value, expire, path, domain);
- Ejemplos
  - <?php
    setcookie("user", ”Bob Sponge", time() + 60 * 60);
    ?>

    <html>
    .....

Cookies: sintaxis (ii)

- Para recuperar los valores
  - <?php
    // Imprimir una cookie
    echo $_COOKIE["user"];

    // Para ver todas las cookies
    print_r($_COOKIE);
    ?>

Cookies: sintaxis (iii)

- Verificar si existe
  - <html>
    <body>

    <?php
    if (isset($_COOKIE["user"]))
      echo "Welcome " . $_COOKIE["user"] . "!
";
    else
      echo "Welcome guest!
";
    ?>

    </body>
    </html>

Sesiones

- Una sesión la relación que se establece entre un cliente y un servidor durante un tiempo finito. El típico ejemplo en Internet es un “carrito de la compra” en el que el cliente (navegador) visita diferentes lugares del supermercado (servidor).
- El servidor debe mantener información a lo largo de todo el proceso. Para ello se utilizan las sesiones.
- En PHP se manejan a través de la variable $_SESSION, que se utiliza como cualquier otra variable

Sesiones

```
<?php
session_start(); // start session
$_SESSION['name'] = 'Alvaro';
?>
<html>
…
```

```
<?php session_start(); ?>
…
<html>
<body>
<?php
echo 'Hello '.$_SESSION['name'];
?>
…
```

```
unset($_SESSION['name']);
```

```
session_destroy();
```