Arquitecturas Distribuidas

Tema 4. Tecnologías de La Web dinámica

Contenido del Tema 4

I. Procesado de información en el servidor.
II. Cookies
III. Lenguajes de programación en el servidor: PHP
IV. Lenguajes de *script* en el cliente
V. Bases de datos
VI. La Web en la actualidad
VII. Optimización de sitios web

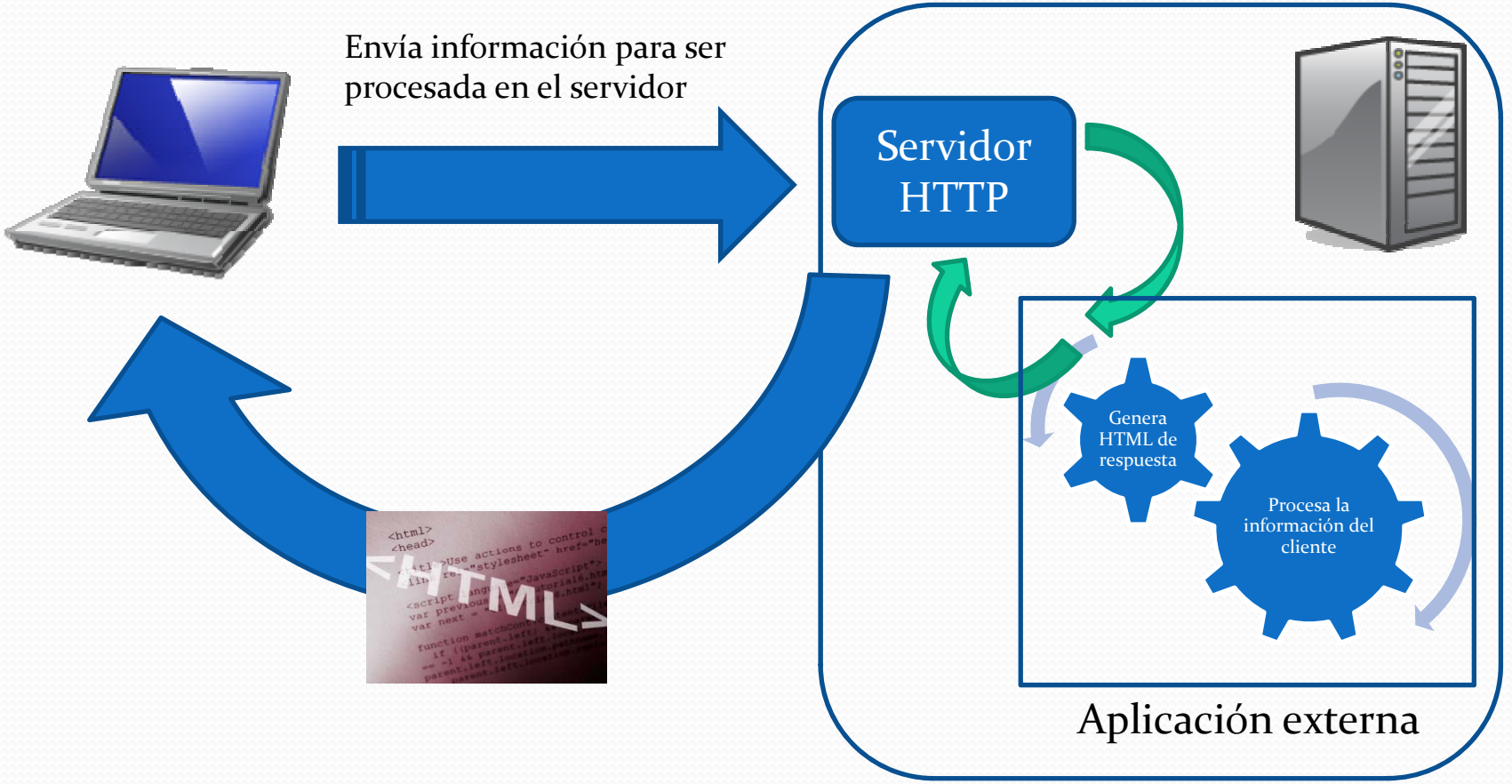
¿Qué es la web dinámica?

- Originalmente la Web no era más que un servicio de intercambio de ficheros (estáticos: creados y almacenados en un servidor) enlazados entre sí.
- A medida que va evolucionando, el contenido de la web deja de ser estático, se genera dinámicamente mediante programa para cada petición.
- Documentos web que cambian en función de múltiples factores:
  - Información proporcionada por cliente fundamentalmente
  - Acciones del usuario en el lado del cliente
- Servidores procesan la información y devuelven una respuesta en formato HTML (y muchas veces, además código para ejecutar en el cliente)
- Consecuencia: Internet deja de ser sólo una enorme BBDD distribuida de documentos y se convierte en una plataforma de servicios y aplicaciones

¿Cómo se consigue?

- Mediante aplicaciones en el lado del servidor => procesan las peticiones de los clientes antes de devolver el documento.
- Mediante aplicaciones en el lado cliente => mejoran la interactividad y descargan al servidor de peticiones y procesado simple.
- Con el uso extensivo de almacenamiento de datos estructurados (bases de datos).
- Mediante el intercambio de datos *entre máquinas*.

Generación dinámica de documentos



I. Procesado de información en el servidor.

1. Justificación y ventajas
   1.1. Ventajas de las aplicaciones en el lado del servidor
   1.3. Aplicaciones comunes
2. ¿Cómo se envía la información al servidor?
   2.1 Formularios HTML
   2.2 Envío de la información (métodos GET y POST)
3. *¿Cómo se procesa la información en el servidor?*
   3.1. *Common Gateway Interface* (CGI)
   3.2. Contexto de ejecución de la CGI
   3.4. Peticiones CGI
   2.5. Respuestas CGI
   2.6. Problemas de las CGI

Ventajas de aplicaciones en el lado del servidor

- Soporte multi-plataforma
  - Existen diferencias (en los navegadores) que en ciertos casos evitan el funcionamiento correcto de los *scripts* en el lado del cliente. Esto no ocurre si el *script* está en el lado del servidor, ya que se ejecuta independientemente de la plataforma del cliente.
  - Solamente hay que optimizar el código en una plataforma (la del servidor).

Ventajas de aplicaciones en el lado del servidor

- Mayores opciones para desarrollar aplicaciones
  - Los programas en el lado del servidor no están limitados por razones de seguridad, por lo que pueden acceder a archivos y bases de datos.
- Integridad del código
  - Los clientes no tienen acceso directo al código, ya que no es necesario para que éste se ejecute.
- Mantenimento
  - Las modificaciones sólo se tienen que hacer en un punto

Aplicaciones comunes

- Motores de búsqueda
- Acceso a remoto a aplicaciones corporativas y bases de datos.
- Acceso a boletines de noticias
- Cualquier aplicación que se alimente de datos proporcionados por el usuario.
- Y miles más…

Algunos problemas a resolver

- ¿Cómo se envía la información al servidor?
- ¿Cómo se procesa la información en el servidor? ¿Quién la procesa?
- ¿Cómo se agrupan un conjunto de peticiones relacionadas?

I. Procesado de información en el servidor.

1. Justificación y ventajas
   1.1. Ventajas de las aplicaciones en el lado del servidor
   1.3. Aplicaciones comunes
2. **¿Cómo se envía la información al servidor?**
   2.1 Formularios HTML
   2.2 Envío de la información (métodos GET y POST)
3. *¿Cómo se procesa la información en el servidor?*
   3.1. *Common Gateway Interface* (CGI)
   3.2. Contexto de ejecución de la CGI
   3.4. Peticiones CGI
   2.5. Respuestas CGI
   2.6. Problemas de las CGI

Formularios HTML

- Necesitamos que el documento muestre al usuario la información que va a necesitar el servidor y permitirle que la rellene: nuevas etiquetas HTML.
- Son elementos (etiquetas) que permiten al usuario insertar parámetros y valores que se envían hacia el servidor.
- La etiqueta para declarar un formulario es `<FORM>`.
- El formulario indica además la URL de la aplicación que tiene que procesar la información que se va a enviar

Formularios HTML

```
<html>
<!-- Ejemplo 20 -->

<head>
 <title>Titulo de la pagina</title>
</head>

<body>
    Formulario para seleccionar parametros
    para enviar al servidor.
    <form action="procesar.cgi" method="GET">
            Entrada de texto: <input name="a"
                    type="text"> <br>
        Clave: <input type="password" name="b"
                    maxlenght="8"> <br>
        Checkbox: <input type="checkbox"
                    name="c"> <br>
        Oculto: <input type="hidden" value="a">
            <br>
     </form>
</body>
</html>
```

- **Diferentes elementos pueden ir insertados en un formulario:**
  - Elementos de entrada de texto, claves, imágenes, archivos, checkbox, etc.

```
<INPUT  type="TEXT|PASSWORD|..."> 
```

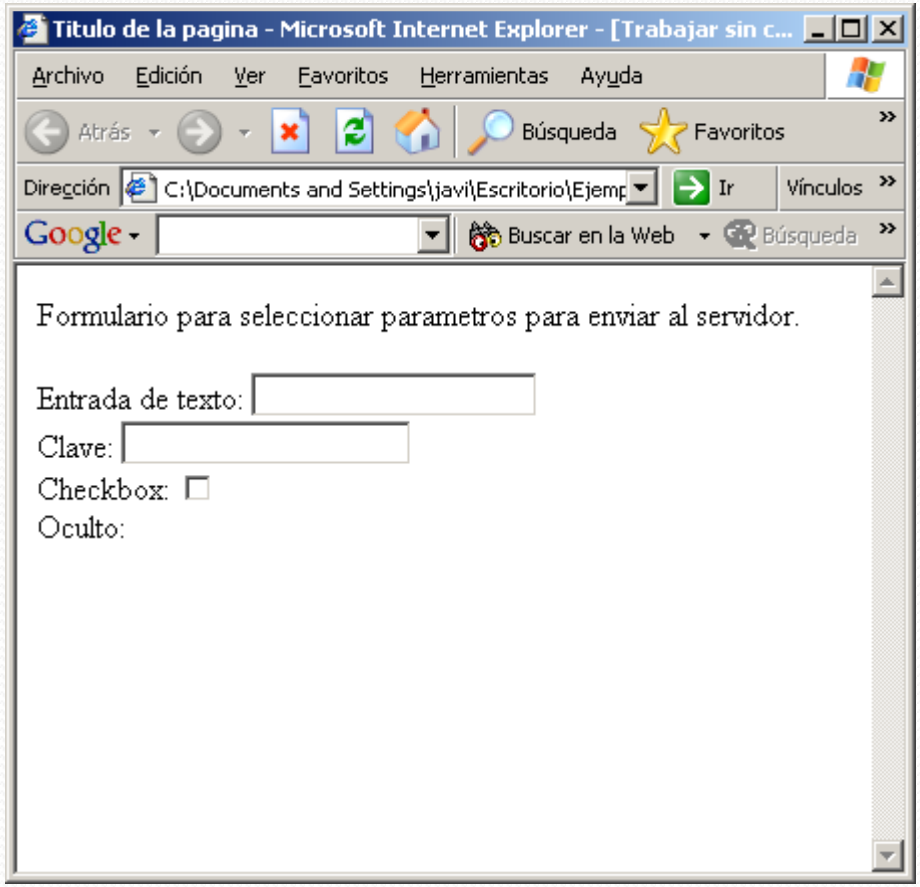
- **Es posible también declarar elementos "ocultos", que no se muestran al usuario, pero sí contienen parámetros que se envían al servidor.**
- **Todos los elementos deben llevar el atributo "name", que indicará como se llama el parámetro.**

Formularios HTML

```
<html>
<!-- Ejemplo 20 -->

<head>
  <title>Titulo de la pagina</title>
</head>

<body>
     Formulario para seleccionar parametros
     para enviar al servidor.
  <form action="procesar.cgi" method="GET">
                Entrada de texto: <input name="a"
                                  type="text"> <br>
          Clave: <input type="password" name="b"
                                  maxlenght="8"> <br>
          Checkbox: <input type="checkbox"
                                  name="c"> <br>
          Oculto: <input type="hidden" value="a">
                <br>
       </form></body>
</html>
```



Formularios HTML

```
<html>
<!-- Ejemplo 21 -->

<head>
  <title>Titulo de la pagina</title>
</head>

<body>
     Formulario para seleccionar parametros
     para enviar al servidor.
 <form action="procesar.cgi" method="GET">
                Selección: <select name="a">
                                <option> Primera
                                <option> Segunda
                                <option> Tercera
                </select> <br> <br>
                Área de texto: <textarea> Escriba aqui
                varias lineas </textarea>
     </form>
</body>
</html>
```

- Elemento de selección entre un grupo de opciones:

```
<SELECT> (<OPTION>)+
```

- Áreas de texto (texto con más de una línea):

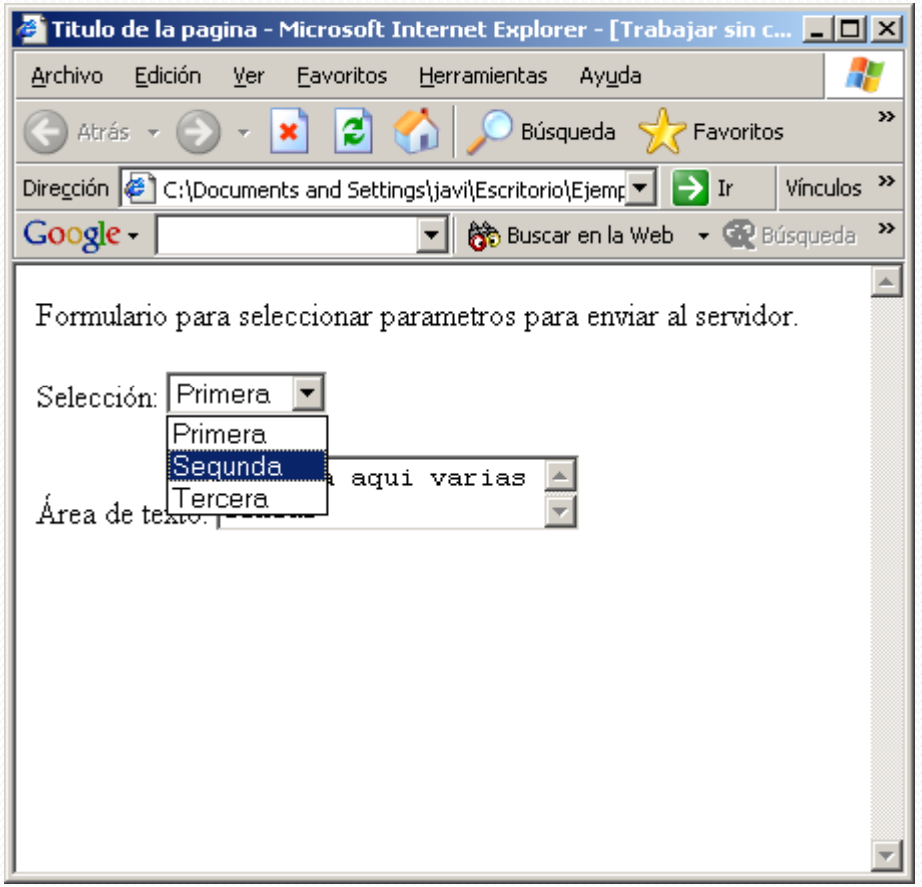
```
<TEXTAREA>
```

Formularios HTML

```
<html>
<!-- Ejemplo 21 -->

<head>
  <title>Titulo de la pagina</title>
</head>

<body>
     Formulario para seleccionar parametros
     para enviar al servidor.
 <form action="procesar.cgi" method="GET">
               Selección: <select name="a">
                              <option> Primera
                              <option> Segunda
                              <option> Tercera
               </select> <br> <br>
               Área de texto: <textarea> Escriba aqui
               varias lineas </textarea>
     </form>
</body>
</html>
```



Formularios HTML

- Existen, además dos elementos <INPUT> especiales:
  - RESET: borra el formulario a su estado original
  - SUBMIT: presenta un botón para enviar el formulario.
- Incluso aunque no exista botón de envío el navegador suele enviar la información al pulsar enter

Formularios HTML

```
<html>
<!-- Ejemplo 21 -->

<head>
  <title>Titulo de la pagina</title>
</head>

<body>
     Formulario para seleccionar parametros
     para enviar al servidor.
      <form action="procesar.cgi" method="GET">
               Selección: <select name="a">
                              <option> Primera
                              <option> Segunda
                              <option> Tercera
               </select> <br> <br>
               Área de texto: <textarea> Escriba aqui
               varias lineas </textarea>
     </form>
</body>
</html>
```

- Al pulsar el botón de envío, se ejecuta una petición al URL indicado en el atributo `action` del `<FORM>`.

Envío de la información

- Puesto que se utiliza el protocolo HTTP, estamos limitados por su una interfaz: sólo se puede utilizar alguno de los comandos del protocolo.
- Se utilizan normalmente dos comandos del protocolo: GET o POST
- Dos tipos diferentes de peticiones, según atributo `method` del `<FORM>`.
  - Peticiones GET (método GET de HTTP)
  - Peticiones POST (método POST de HTTP)
- Al pulsar el botón de envío el navegador construye la petición adecuada

Peticiones GET

- Peticiones GET: (método GET de HTTP)
  - Los parámetros se añaden a la URL de la petición tras un signo de ?
  - El formato es *nombre_par=valor* y separados por &

```
http://www.upct.es/cgi?name1=value1&name2=value2&name3=value3
```

  - Los caracteres especiales son trasladados a otros símbolos: Por ejemplo, los espacios se traducen a "+", y los caracteres del ASCII extendido se envían con el formato %NNN (NNN: número del código ASCII extendido).

Peticiones POST (I)

- Peticiones POST: (método POST de HTTP)
  - Los parámetros se envían en **el paquete HTTP**, tras la línea de solicitud y las cabeceras de envío. Los parámetros son, por tanto, la entidad u objeto codificado en la petición.
  - No se añade nada a la URL de la petición.
  - Como en el método GET, los caracteres especiales se traducen a sus extensiones ASCII.
  - Es necesario indicar el tipo de codificación en el <form> con el atributo *enctype*
    - **application/x-www-form-urlencoded .** (Por defecto). Similar a GET. NO PERMITE ENVIAR ARCHIVOS
    - **multipart/form-data**. Separa los parámetros mediante una marca (*boundary*) aleatoria. PERMITE ENVIAR ARCHIVOS

Peticiones POST (II), ejemplo

- Paquete HTTP de la petición

POST /pago.php HTTP/1.1

Host: ait.upct.es

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; es-ES; rv:1.9.0.15) Gecko/2009101601 Firefox/3.0.15 (.NET CLR 3.5.30729)

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language:es-es,es;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.7

Keep-Alive: 300

Connection: keep-alive

Referer:http://ait.upct.es/asignaturas/ad/post.html

Cache-Control: max-age=0

Enviar=Enviar&nombre=pepe&cuenta=212415

GET vs POST

- Problemas GET
  - No se puede enviar información binaria (archivos, imágenes, etc.) => necesario el método POST
- Problemas POST
  - Rompe la funcionalidad del botón "Atrás" del navegador
  - El botón actualizar repite la operación

- Principios generales
  - GET implica "obtener" información. **Operaciones *idempotentes***
  - POST implica "realizar" una acción con un "efecto secundario". **Operaciones no *idempotentes***
- Ejemplos
  - Buscar usuarios o contenidos: GET
  - Registrar usuarios o actualizar un perfil: POST

I. Procesado de información en el servidor.

1. Justificación y ventajas
   1.1. Ventajas de las aplicaciones en el lado del servidor
   1.3. Aplicaciones comunes
2. ¿Cómo se envía la información al servidor?
   2.1 Formularios HTML
   2.2 Envío de la información (métodos GET y POST)
3. ***¿Cómo se procesa la información en el servidor?***
   3.1. *Common Gateway Interface* (CGI)
   3.2. Contexto de ejecución de la CGI
   3.4. Peticiones CGI
   2.5. Respuestas CGI
   2.6. Problemas de las CGI

¿Cómo se procesa la información en el servidor?

- El servidor HTTP en principio sólo recibe peticiones, recupera el documento indicado en el *path* de la URL y lo devuelve al cliente
- Si es necesario procesar información del cliente, el servidor debe encargarse de pasar dicha información a una aplicación externa que la procese y genere una respuesta
  - ¿Cómo puede el servidor ejecutar un aplicación externa?
  - ¿Cómo le pasa la información (parámetros GET o POST) que ha recibido a dicha aplicación?
  - ¿Existe algún mecanismo estándar?

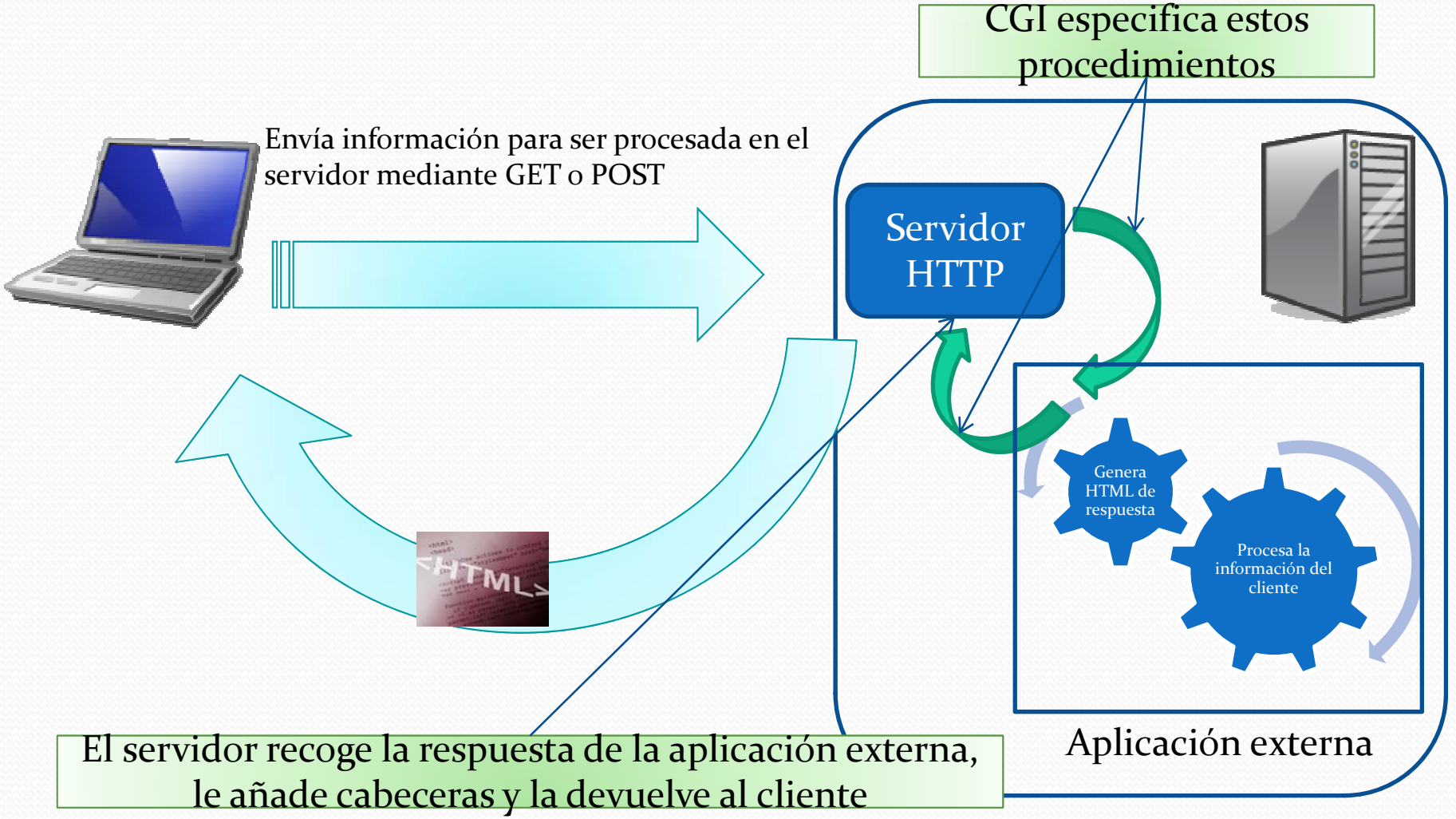
Common Gateway Interface (I)

- *Common Gateway Interface*
- Procedimiento **estándar** para comunicar servidores HTTP y aplicaciones externas para generar documentos de forma dinámica.
- El URL de la petición identifica un programa ejecutable externo (el servidor web debe poder identificarlo como tal con algún criterio).
- El servidor ejecutará el programa externo y le pasará los parámetros

Common Gateway Interface (II)

1. El servidor recibe una petición con una serie de parámetros
2. El servidor reconoce que la URL no es un documento sino una aplicación externa
3. El servidor ejecuta la aplicación externa en un *contexto especial*
    1. Establece el valor de una serie de variables de entorno
    2. Redirecciona la salida estándar (stdout) de la aplicación externa hacia él mismo
4. La información que recibe de la salida estándar la envía al cliente
    1. Añade cabeceras previas si es necesario

Common Gateway Interface (III)



Resumen CGI

- CGI especifica como pasar parámetros y ejecutar las aplicaciones externas
- La aplicación externa puede estar programada en cualquier lenguaje, siempre que éste sea capaz de aceptar la entrada según especifica CGI, y de crear una respuesta adecuada.
- Lenguajes más habituales: PERL, PYTHON, C, TCSH, BASH, Java.

¿Cómo pasar los parámetros e información adicional?

- Problema de **comunicación entre procesos** (repasar Tema 1):
  - Servidor y aplicación externa son dos procesos independientes
  - Múltiples soluciones. Se escoge la menos costosa computacionalmente
  - Se utilizan variables de entorno: el SO las pone a disposición de cualquier proceso en ejecución
- En el contexto de ejecución, el servidor HTTP, establece las variables de entorno:
  - CONTENT_LENGTH, CONTENT_TYPE, REMOTE_HOST, REMOTE_USER, REQUEST_METHOD, SERVER_NAME, QUERY_STRING, GATEWAY_INTERFACE, HTTP_*
  - La aplicación externa puede leer el valor de las que necesite, por ejemplo, en REMOTE_HOST el servidor escribe la dirección IP del cliente que ha hecho la petición

Peticiones CGI

- Dos formas diferentes. Depende del método con el que se envíen los parámetros, GET o POST
- Para peticiones GET la variable QUERY_STRING toma el valor de los parámetros, tal y como aparecen en la URL.
  - La aplicación externa lee el valor de la variable
- Para peticiones POST los parámetros se pasan por la entrada estándar (*stdin*).
  - El servidor redirecciona hacia el mismo la entrada estándar de la aplicación externa al ejecutarla e introduce los parámetros
  - La aplicación externa debe leer la entrada estándar.

Respuestas CGI

- La CGI escribe la respuesta en el *stdout*. El servidor la lee, y se la envía al cliente.
- Según el tipo de servidor, la CGI debe:
  - Servidor NPH (*No Parse Header*): Escribir la respuesta completa (incluyendo cabeceras)
  - Servidor PH (*Parse Headers*): Escribir la entidad respuesta, y pasar al servidor indicaciones sobre cómo formar las cabeceras.
- Lo habitual es que el servidor sea tipo NPH.

Respuestas CGI

- Para servidores NPH, una CGI debe generar la salida:

```
(Cabecera_CGI|Cabecera_HTTP)
[LÍNEA EN BLANCO]
[Cuerpo_Entidad]
```

La cabecera_CGI es una de las siguientes:

- Content-type: Obligatoria si existe `Cuerpo_Entidad`
- Location: devuelve un puntero (URL) en lugar de la información
- Status: resultado de la operación, si no se incluye el cliente sobreentiende "200 OK".

Problemas de las CGI

- CGI se ha quedado obsoleto
- En la práctica las aplicaciones que procesan las peticiones se integran **modularmente** en el propio servidor HTTP
- Integración → *Servlets*, SSI, PHP
- Portabilidad → *Servlets*, PHP
- Seguridad → *Servlets*, PHP
- Escalabilidad → *Servlets*, PHP, ASP
  - Cada vez que se recibe una petición hay que crear un nuevo proceso -> Es un proceso computacionalmente costoso
  - Solución: se utilizan lenguajes de procesado integrados en la implementación del propio servidor (módulos adicionales). Ej.: *Servlets*, PHP, ASP

Referencias y bibliografía

- CGI
  - http://ait.upct.es/asignaturas/ad/manuales/CGI/cgi.html → Manual sobre uso de CGI
  - http://ait.upct.es/asignaturas/ad/manuales/CGI/env.html → Documento que discute el entorno de ejecución de una CGI (en inglés)
  - http://ait.upct.es/asignaturas/ad/manuales/CGI/especificacion.html → Especificación CGI (en inglés)
- Formularios HTML
  - http://html.conclase.net/w3c/html401-es/interact/forms.html
- GET y POST
  - Pruebe a buscar GET vs POST en Google