Uso de formularios en HTML para enviar y recopilar datos

Piero Berni Millet

PID_00155708

Universitat Oberta de Catalunya
www.uoc.edu

Índice

1. Introducción a CGI y su entorno

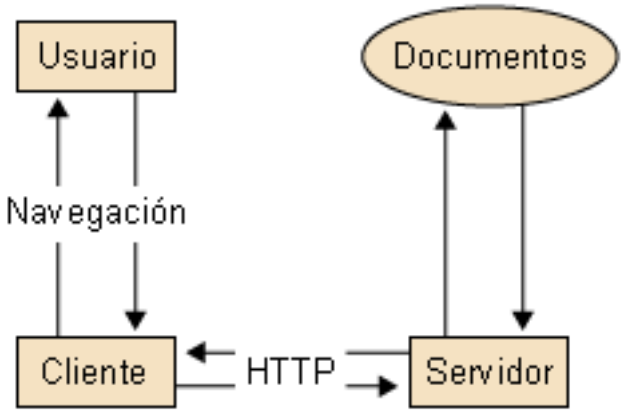
La interfaz de pasarela común (CGI[1]) es la puerta de acceso que hay entre una página web y el servidor de Internet donde reside la página.

(1)CGI es la sigla inglesa de *common gateway interface*.

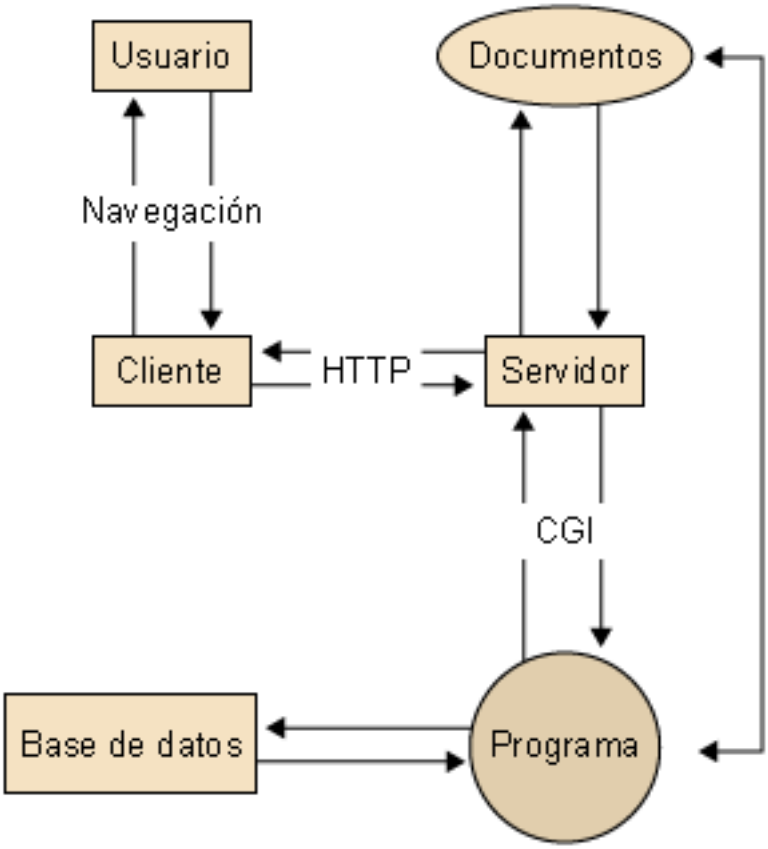
Si habéis utilizado un navegador web, os habréis encontrado con páginas web que permiten interrogar bases de datos para obtener información. ¿Qué está sucediendo detrás de la página? En resumen, el navegador envía solicitudes al servidor y el servidor envía respuestas al navegador. Este intercambio es una cuestión sencilla cuando la solicitud es mostrar otra página web. Pero cuando el navegador desea algo más complejo, como el último parte meteorológico en España, hay grandes posibilidades de que el gestor de la información detrás de la página web sea un programa escrito con un *script* CGI.

Lenguaje de un *script* CGI

Un *script* CGI puede ser escrito en cualquier lenguaje que pueda leer de STDIN (entrada estándar), escribir en STDOUT (la salida estándar) y leer variables de entorno como virtualmente cualquier lenguaje de programación, incluyendo C, Perl, PHP, o incluso *scripts* de Shell de UNIX y LINUX.



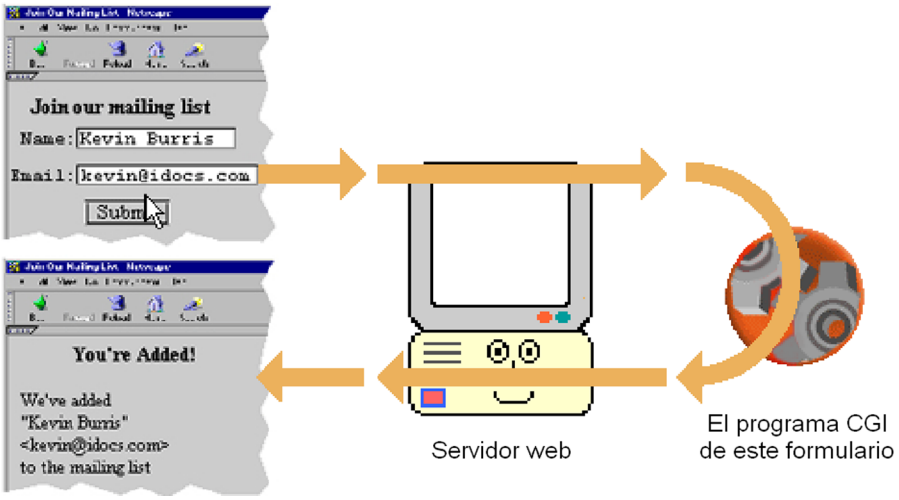
Solicitud http sencilla para mostrar otra página web



Solicitud http combinada con CGI

La programación CGI consiste en el diseño y la escritura de programas que reciben sus órdenes iniciales desde una página web. Los formularios HTML se han convertido en el método preferido para enviar datos a través de Internet debido a su facilidad de uso y de configuración.

La idea básica detrás de un formulario es sencilla, tal y como se ilustra en la siguiente figura. El visitante abre un formulario y lo rellena; el navegador envía los datos al servidor; el programa CGI captura la corriente de datos del formulario y la procesa; el programa CGI devuelve algún tipo de respuesta al visitante a través del servidor.



La idea básica detrás de un formulario

2. Uso de formularios HTML/XHTML

Las etiquetas <form> son la base para el traspaso de datos a los programas CGI en el servidor.

> Un formulario de una página web es una colección de campos de entrada que comunican su contenido al servidor.

En el mundo de Internet y de las páginas web se suelen utilizar para múltiples propósitos: formularios de pedidos de una tienda electrónica, cuestionarios, encuestas, libros de invitados, consultas a bases de datos, etc. Todos ellos devuelven datos al propietario de la página web para que sean procesados posteriormente.

En este apartado vamos a ver sólo el modo de situar un formulario en una página web. Más adelante aprenderemos a procesar los datos que se reciban desde un formulario.

2.1. Tipos de entradas de formularios

Los elementos de los formularios HTML/XHTML ofrecen una variedad de formas de aceptar las entradas de los datos de los visitantes: entradas de líneas de texto, campos de texto formados por múltiples líneas, casillas de verificación, botones de opción, listas de opciones de varios tipos, campos ocultos para administrar la información, etc.

Podéis colocar cualquiera de estos elementos en el cuerpo de un documento HTML/XHTML si se delimitan con la etiqueta <form> y su respectiva etiqueta de fin, </form>. Todos los elementos dentro de estas dos etiquetas componen un único formulario. Los elementos de entrada de datos más comunes son: <input>, <select>, y <textarea>.

2.2. La etiqueta <input>

Los elementos de entrada de un formulario se crean principalmente con la etiqueta <input>. Ésta define el comportamiento y la posición del campo para que acepte datos del visitante en un formulario. Esta etiqueta acepta una serie de atributos como *type*, *name*, *value*, *size*, *maxlength*, etc.

En el siguiente ejemplo se muestra un formulario con tres elementos para introducir texto, “nombre”, “apellidos” y “direccion”:

- Los dos primeros elementos <input> son cuadros de texto de una única línea (type="text") de 30 y 50 caracteres de ancho (*size*); permite al usuario teclear sólo 30 y 50 caracteres (maxlength), respectivamente; el contenido del campo se muestra, por defecto, en blanco al no existir el atributo value.

- El tercer campo es otro cuadro de texto con un tamaño de visualización de 75 caracteres y con capacidad máxima para 100 caracteres; establece un valor inicial (value=" Calle ") como encabezamiento del resto de la información que va a rellenar el usuario.

El nombre de la variable que identifica cada campo se indica en el atributo *name*. Este nombre es imprescindible para que el servidor pueda procesar el formulario.

```
<html>
<head><title>Ejemplo de formulario sencillo</title></head>
<body>

<h3>Formulario muy sencillo</h3>

<form action=" formulario.php" method="post">
  Escribe tu nombre:
  <input name="nombre" type="text" value="" size="30" maxlength="30" />
  <br/>
Escribe tus apellidos:
<input name="apellidos" type="text" value="" size="50" maxlength="50" />
  <br/>
Escribe tu dirección particular:
<input name="direccion" type="text" value="Calle " size="75" maxlength="100" />
  <br/>
  <input type="submit" value="Enviar" />
</form>

</body>
</html>
```

El atributo *type* indica el tipo de control que se incluye en el formulario y que puede tener los siguientes valores:

```
"text | password | checkbox | radio | submit | reset | file
| hidden | image | button"
```

A continuación presentamos el significado de los controles en función de su tipo:

1) type="password". Oculta la entrada del usuario reemplazándola por asteriscos en la pantalla, por lo que es ideal para escribir contraseñas y otros datos privados. El navegador los transmite sin encriptar cuando se envían al servidor.

```
Contraseña <br/>
```

<input type=”password” name=”contrasena” value=”” />

2) type=”checkbox”. Las casillas de verificación son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente. Si se quiere mostrar un checkbox seleccionado por defecto, se utiliza el atributo *checked*. Las casillas de verificación no seleccionadas no aparecen en los datos enviados.

```
Puestos de trabajo buscados <br/>
<input name="puesto_directivo" type="checkbox" value="directivo"/> Directivo
<input name="puesto_tecnico" type="checkbox" value="tecnico"/> Técnico
<input name="puesto_empleado" type="checkbox" value="empleado"/> Empleado
```

3) type=”radio”. Botones radiales. Los controles de tipo radiobutton son similares a los controles de tipo checkbox, pero presentan una diferencia muy importante: son mutuamente excluyentes. Son adecuados para aceptar un valor individual de un conjunto de alternativas. Todos los botones radiales de un mismo grupo reciben el mismo nombre. Solamente el botón radial seleccionado del grupo aparece en los datos enviados. Uno de ellos puede estar seleccionado desde el inicio si se le incluye el atributo *checked*.

```
Tipo de inscripción <br/>
<input type=”radio” name=”inscripcion” value=”completa”
checked=”checked” /> Cuota completa: 180 €
<input type=”radio” name=”inscripcion” value=”general” />
Cuota general: 80 €
<input type=”radio” name=”inscripcion” value=”estudiante” />
Cuota estudiante: 50 €
```

4) type=”hidden”. Los campos ocultos se emplean para añadir información oculta en el formulario, por lo que no aparece en la pantalla de la página web, de modo que el usuario desconoce que el formulario los incluye. Se utiliza para comunicar información sobre el estado de la interacción entre el cliente y el servidor, que necesita el servidor pero que no es necesario o no es posible que la establezca el usuario; por ejemplo, un identificador de transacción.

```
<input type="hidden" name="url_previa" value="/articulo/primero.html" />
```

5) type=”submit”. Botón de envío de formulario. La mayoría de los formularios disponen de un botón para enviar al servidor los datos introducidos por el usuario. El valor del atributo *value* es el texto que muestra el botón. Si no se establece el atributo *value*, el navegador muestra el texto predefinido Enviar consulta. Pero si se proporciona un atributo *name*, podremos incluir los atributos *name* y *value* entre los datos enviados. Esta opción es útil cuando se coloca en el formulario más de un botón de envío, ya que brinda un modo de identificar en qué botón del formulario se ejecutó la acción.

```
<input type="submit" name="buscar" value="Buscar" />
```

6) type="reset". Botón de reseteo o restauración del formulario. Se trata de un botón especial que borra todos los datos introducidos por el usuario y devuelve el formulario a su estado original.

```
<input type="reset" name="limpiar" value="Borrar datos del
formulario" />
```

2.3. La etiqueta <select>...</select>

Las casillas de verificación y los botones de selección son una buena solución para crear preguntas y respuestas de selección múltiple en un formulario, pero pueden llegar a convertirse en tediosas y confusas cuando aparecen en pantalla largas listas de elementos. Una manera alternativa y más compacta de representar listas de elementos es la etiqueta select que se representa como un menú desplegable o emergente, y que consta de una o de múltiples opciones. Sus atributos son los siguientes:

- name="un_nombre". El nombre de la variable que identifica este campo.
- size="valor". Número de filas que se muestran de la lista (por defecto sólo se muestra una).
- multiple. La presencia de este atributo indica que se puede seleccionar más de una opción al mismo tiempo.

Las listas desplegables se definen con la etiqueta <select> y cada elemento de la lista se define mediante la etiqueta <option>. El elemento de una lista desplegable tiene dos atributos comunes: *selected* = "selected", indica si el elemento aparece seleccionado por defecto al cargarse la página; *value* = "texto", el valor que se envía al servidor cuando el usuario elige esa opción. A continuación podemos ver un ejemplo donde se crea una lista desplegable para elegir un sistema operativo:

```
<label for="so">Sistema operativo</label> <br/>
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="windows">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>
```

2.4. La etiqueta <textarea>...</textarea>

Como hemos visto, el tipo de entrada <input type="text"> limita la entrada de datos a una sola línea de caracteres. La etiqueta <textarea> da más libertad al usuario al permitir introducir una gran cantidad de texto repartido en una línea de texto. El texto que se extiende hasta la etiqueta de cierre se emplea para inicializar el valor del campo:

```
<form action="proyecto.php" method="post">
   <label for="titulo">Título del proyecto</label> <br/>
   <input type="text" name="titulo" value="" />
  <label for="descripcion">Descripción del proyecto</label>
      <br/>
   <textarea name="descripcion" cols="40" rows="5">
      </textarea>
</form>
```

Los atributos específicos son los siguientes:

- name="un_nombre". El nombre de la variable que identifica el campo.
- rows="numero". Número de líneas que mostrarán en el cuadro de texto.
- cols="numero". Número de caracteres que se muestran en cada fila.

2.5. Distintas maneras de enviar los datos de un formulario (atributo *method*)

La etiqueta <form> de HTML se vale de los atributos especiales *action* y *method* para informar al navegador de cómo codificar los datos y dónde enviarlos:

```
<form action="http://comoras.uoc.edu/aplicacion.php" method="post">
...
</form>
```

El atributo *action* = "url" proporciona la URL absoluta o relativa de la aplicación que va a recibir y procesar los datos del formulario. La aplicación es típicamente un *script* de un programa (escrito usualmente en Perl o PHP), que puede separar los datos y darles formato.

El atributo *method* indica cómo deben enviarse los datos desde el formulario al servidor. La diferencia entre estos dos métodos es la siguiente:

- Con el método GET los parámetros se pasan como parte de la URL que llama a la aplicación para procesar el formulario del lado del servidor. El navegador agrega los datos a la URL asignada en action, separados por el signo de interrogación "?". El método GET tiene varias limitaciones importantes: no puede manejar muchos datos debido a que algunos sistemas operativos limitan el número y la longitud de argumentos de línea de co-

mandos (en general, admite como máximo el envío de unos 500 bytes de información), que pueden pasarse a una aplicación a la vez. Además, GET coloca los parámetros del formulario directamente en la URL, donde pueden ser capturados con facilidad por los usuarios de Internet, poniendo en peligro la seguridad del sistema de envío. La otra gran limitación del método GET es que no permite el envío de archivos adjuntos con el formulario.

- El método POST es el preferido y el más recomendado. Este método también envía los datos como un añadido a una URL, pero los envía después de que se hayan enviado todas las cabeceras de solicitud del servidor, y por lo tanto invisible al usuario. La principal ventaja de POST es la de permitir enviar formularios con muchos campos o con campos de texto extensos.

Si no sabéis qué método elegir para un formulario, existe una regla general que dice que el método GET se debe utilizar en los formularios que no modifican la información (por ejemplo, en un formulario de búsqueda). Por su parte, el método POST se debería utilizar cuando el formulario modifica la información original (insertar, modificar o borrar alguna información).

El atributo *enctype* especifica el tipo de codificación empleada al enviar el formulario al servidor (sólo se indica de manera explícita en los formularios que permiten adjuntar archivos). El valor predeterminado es: application/x-www-form-urlencoded. Aparte de esta codificación, el tipo de codificación para adjuntar un archivo se indica con multipart/form-data.

El formato multipart/form-data se utiliza para los formularios que contienen campos para seleccionar archivos que el usuario puede obtener.

El formato text/plain se puede utilizar junto con la URL de tipo *mailto* en el atributo *action* para enviar el formulario a una dirección de correo electrónico. Por ejemplo, para enviar los datos de un formulario por correo electrónico, sin ser procesados por un servidor, escribiremos:

```
<form method="post" action="mailto:user@server.com" enctype="text/ plain">
```

2.6. Codificación estándar *application/x-www-form-unlencoded*

Los formularios lanzan sus datos con la codificación estándar de la URL. Las reglas principales de este formato de datos se explican a continuación.

Todos los datos que se han introducido desde un formulario se envían al servidor o al programa CGI como pares de nombre/valor. Los pares de nombre/valor se pasan siempre al servidor como name=value y cada nuevo par se separa por medio de un signo &, name1=value1&name2=value2.

Por ejemplo, observad la siguiente entrada:

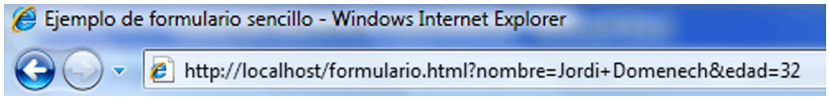
```
<h3>Formulario muy sencillo</h3>
<form method="get">
  Escribe tu nombre:
  <input type="text" name="nombre" value="" />
  <br/>
  Escribe tu edad:
  <input type="text" name="edad" value="" />
  <br/>
  <input type="submit" value="Enviar" />
</form>
```

Formulario muy sencillo

Escribe tu nombre: Jordi Domenech
Escribe tu edad: 32
Enviar

Al enviar los datos del ejemplo ilustrado con el método GET, los datos aparecen adjuntos a la URL y codificados con el siguiente formato:

```
nombre=Jordi+Domenech&edad=32
```



Los datos de los dos campos del ejemplo aparecen detrás del signo de interrogación como pares de nombre=valor. El signo & separa los pares entre sí. Dentro de cada par, el valor de la variable se indica mediante un signo de igualdad (=). Los espacios en blanco se sustituyen por un signo más (+).

Los caracteres especiales de los datos que se introducen en los campos de un formulario se codifican al ser incluidos en la cadena de la URL. Por ejemplo, los caracteres reservados a la URL (?, &, =, #, %, /) se deben codificar si no se quiere que el servidor los interprete según sus significados especiales. Los caracteres no alfanuméricos se codifican convirtiéndolos a sus valores hexadecimales, con el signo de porcentaje (%) delante del número HEX. Por ejemplo, el signo interrogación (?) se codifica como %3F.

Por ejemplo, la siguiente entrada:

Formulario muy sencillo

Escribe tu nombre: María Fernández
Escribe tu edad: 33
Enviar

producirá esta codificación de los datos:

```
nombre=Mar%C3%ADa+Fern%C3%A1ndez&edad=33
```

El lenguaje PHP proporciona una manera sencilla de manejar formularios y permite crear programas interactivos que sepan responder según los datos que el usuario suministre. Al diseñar un formulario, deberemos indicar el fichero PHP que lo procesará, así como el método por el que se le pasará la información del formulario.

3. Leer datos de un formulario con PHP

Antes de ver cómo se manejan los datos desde el lado del servidor con PHP, vamos a construir un sencillo formulario de ejemplo sumando algunos de los conceptos y controles de formularios explicados hasta ahora.

Antes, copiaremos el siguiente bloque de código PHP en un archivo de texto y le pondremos el nombre formulario.php.

```
<?php
/* Muestra los datos enviados a través de un formulario HTML
a través del estado actual de PHP */
phpinfo( );
?>
```

Copiamos el siguiente código HTML/XHTML en un archivo de texto y le ponemos el nombre formulario.html.

```
<html>
<head><title>Ejemplo de formulario sencillo</title></head>
   <body>
   <h3>Formulario muy sencillo</h3>
     <form action="formulario.php" method="post">
     <input type="hidden" name="codigo_solicitud" value="X76345P09" />
     Escribe tu nombre:
     <input type="text" name="nombre" value="" />
     <br/>
     Escribe tu edad:
     <input type="text" name="edad" value="" />
     <br/>


     Escribe una contraseña:
     <input type="password" name="contrasena" value="" />
     <br/>
     Sexo:
     <input type="radio" name="sexo" value="hombre" checked="checked" />
       Hombre
     <input type="radio" name="sexo" value="mujer" /> Mujer
     <br/>
     Puestos de trabajo buscados:
     <input name="puesto_directivo" type="checkbox" value="directivo"/>
       Directivo
     <input name="puesto_tecnico" type="checkbox" value="tecnico"/>
       Técnico
     <input name="puesto_empleado" type="checkbox" value="empleado"/>
       Empleado
     <br/>
```

```
    <label for="so">Sistema operativo preferido</label>
    <select id="so" name="so">
      <option value="" selected="selected">- selecciona -</option>
      <option value="windows">Windows</option>
      <option value="mac">Mac</option>
      <option value="linux">Linux</option>
      <option value="otro">Otro</option>
    </select>
    <br/>
    <label for="observaciones">Observaciones</label> <br/>
    <textarea id="observaciones" name="observaciones" cols="40"
      rows="5"></textarea>
    <br/>
    <input type="submit" value="Enviar" />
    </form>
  </body>
</html>
```

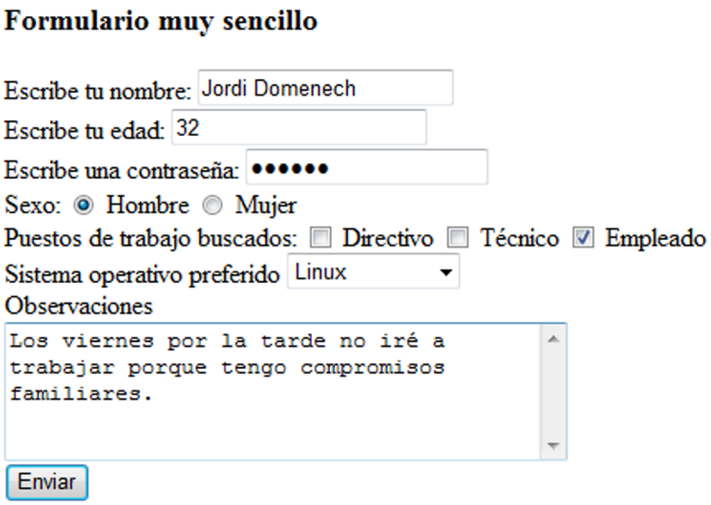
Guardamos los dos ficheros en el directorio raíz del servidor web de Apache:

```
C:\GMMD\apache\htdocs\
```

El método de envío es POST y action proporciona la URL de la aplicación que va a recibir y a procesar los datos, "formulario.php"; el atributo *enctype* no ha sido especificado y, por lo tanto, el formulario lanzará sus datos con la codificación estándar de la URL.

```
<form action="formulario.php" method="post">
```

Podemos visualizar el formulario en la pantalla del navegador introduciendo http://localhost/form.html. A continuación, rellenamos los datos del formulario tal y como aparecen colocados en la siguiente ilustración:



Al pulsar el botón “Enviar”, el contenido del formulario es enviado a la página que indicamos en el atributo *action* de la etiqueta <form>. Las variables de dicho formulario pasan a estar automáticamente disponibles en el *script* gracias al servidor Apache y a PHP.

PHP Variables

Variable	Value
_REQUEST["codigo_solicitud"]	X76345P09
_REQUEST["nombre"]	Jordi Domenech
_REQUEST["edad"]	32
_REQUEST["contrasena"]	secret
_REQUEST["sexo"]	hombre
_REQUEST ["puesto_empleado"]	empleado
_REQUEST["so"]	linux
_REQUEST["observaciones"]	Los viernes por la tarde no iré a trabajar porque tengo compromisos familiares.
_REQUEST["__utmc"]	1
_REQUEST["__utma"]	1.1215844192.1231071363.1231404711.1231409323.20
_REQUEST["__utmz"]	1.1231071363.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none)
_POST["codigo_solicitud"]	X76345P09
_POST["nombre"]	Jordi Domenech
_POST["edad"]	32
_POST["contrasena"]	secret
_POST["sexo"]	hombre
_POST["puesto_empleado"]	empleado
_POST["so"]	linux
_POST["observaciones"]	Los viernes por la tarde no iré a trabajar porque tengo compromisos familiares.

3.1. Variables predefinidas de PHP y su relación con los formularios

Los datos del formulario enviados con los métodos POST o GET se encuentran disponibles en las variables predefinidas de PHP. Estas variables se conocen también como superglobales porque siempre están disponibles en todos los ámbitos a lo largo de un *script*. PHP ofrece un conjunto de variables o matrices predefinidas que contienen variables del servidor web, el entorno y entradas del usuario. Las variables superglobales de PHP son: $GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST, $_ENV.

Las variables superglobales que nos interesan analizar al procesar un formulario son: $_SERVER, $_REQUEST, $_GET, $_POST. Todas ellas son del tipo *array* asociativo:

Array asociativo

Un *array* asociativo es un tipo abstracto de dato formado por una colección de claves únicas y una colección de valores, con una asociación uno a uno. En vez de estar organizados con índices numéricos en función de su posición dentro del *array* (*arrays* escalares), lo están por claves (*key*) que pueden ser cadenas de texto. Con ello, tenemos la posibilidad de poner cualquier tipo de dato para especificar el índice.

- $_SERVER: Información del servidor y el entorno de ejecución.
- $_REQUEST: Variables de petición HTTP. Un valor tipo *array* asociativo que contiene de manera predeterminada los datos de $_GET, $_POST, y $_COOKIE.
- $_GET: Variables HTTP GET.
- $_POST: Variables HTTP POST.

A continuación vamos a modificar el *script* formulario.php para recoger y mostrar en la pantalla del navegador los datos enviados desde el formulario HTML:

```
<?php
   echo "Método HTTP empleado al enviar el formulario fue:
   ",$HTTP_SERVER_VARS['REQUEST_METHOD'],"<br>";
   echo "Código de solicitud: ".
   $HTTP_POST_VARS['codigo_solicitud']."<br/>";
   echo "Nombre: ". $HTTP_POST_VARS['nombre'],"<br/>";
   echo "Edad: ". $HTTP_POST_VARS['edad'],"<br/>";
   echo "Contraseña: ". $HTTP_POST_VARS['contrasena'],"<br/>";
   echo "Sexo: ". $HTTP_POST_VARS['sexo'],"<br/>";
   echo "Puesto de trabajo buscado: ".
   $HTTP_POST_VARS['puesto_empleado'],"<br/>";
   echo "Sistema operativo preferido: ". $HTTP_POST_VARS['so'],"<br/>";
   echo "Observaciones: ". $HTTP_POST_VARS['observaciones'],"<br/>";
?>
```

El resultado en pantalla será:

```
Método HTTP empleado al enviar el formulario fue: POST
Código de solicitud: X76345P09
Nombre: Jordi Domenech
Edad: 32
Contraseña: secret
Sexo: hombre
Puesto de trabajo buscado: empleado
Sistema operativo preferido: Linux
Observaciones: Los viernes por la tarde no iré a trabajar
porque tengo compromisos familiares.
```

Los *arrays* asociativos globales $_GET, $_POST y $_REQUEST se pueden utilizar conjuntamente con las funciones de PHP, each() y list(), para recorrer la matriz, y obtener así los nombres de cada variable del formulario con su valor

asignado. Estas variables se comportan en realidad como un *array* asociativo. Para cada elemento del formulario se crea un vínculo entre la clave y el valor asociado.

Se pueden utilizar conjuntamente las funciones de PHP, each() y list(), para recorrer la matriz $ _POST, y obtener así los nombres de cada variable del formulario con su valor asignado.

A continuación vamos a reescribir el *script* formulario.php para recorrer la matriz $_POST y visualizar las variables pasadas con el método POST. Haremos el recorrido mediante la función each():

```
<?php
  echo "<p>Los valores enviados con el método POST:<p/>";
    while(list($nombre,$valor) = each($_POST)){
  echo $nombre." = ".$valor."<br/>"; }
?>
```

La función each() va recuperando valores del array $_POST y avanza el puntero. La función list() asigna en cada operación los valores del *array* a la lista de variables ($nombre, $valor) pasadas como argumento.

El resultado en pantalla será:

```
Los valores enviados con el método POST:
codigo_solicitud = X76345P09
nombre = Jordi Domenech
edad = 32
contrasena = secret
sexo = hombre
puesto_empleado = empleado
so = Linux
observaciones = Los viernes por la tarde no iré a trabajar
porque tengo compromisos familiares.
```