¿Qué es PHP?

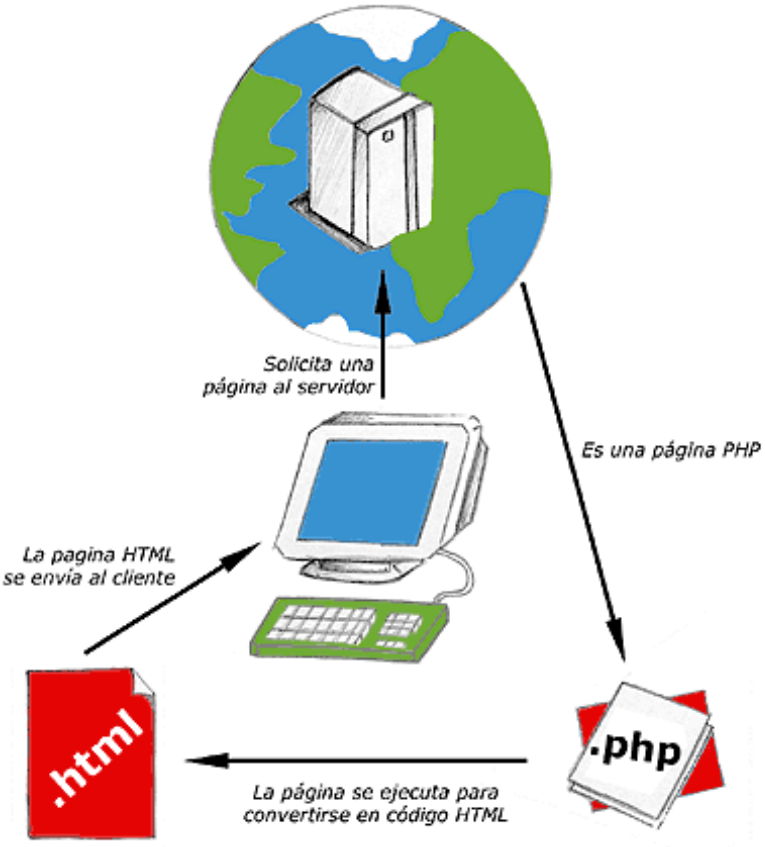
PHP es un acrónimo recursivo que significa *PHP Hypertext Pre-processor*.

Subjetivamente el mejor lenguaje para desarrollar sitios web dinámicos (aunque python intenta ser mi amigo...) :-P

“PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores.

Aplicaciones desarrolladas con PHP: drupal, wordpress, facebook, mediawiki, moodle, etc, etc...

Modelo Cliente - Servidor con PHP



¿Qué sucede detrás del malévolo click?

1. Solicita una Página al servidor: ya sea cuando escribimos www.algo.com.ar o cuando hacemos click en algún enlace, o bien cuando presionamos el botón submit (enviar).
2. ¿Es una pagina .php? Que el compilador de php se arregle.
3. El compilador ejecuta la pagina para convertirse en código **xhtml válido** (*o eso debería*)
4. La página xhtml (o resultado) se envía al cliente como respuesta a su petición (*no les voy a mentir el servidor web intercede en este proceso, aunque no figure en el diagrama*).

Test 1: Hola mundo.

```
<html>
    <head>
        <title>Ejemplo de PHP.</title>
    </head>

    <body>
        <p><?php echo "Hola Mundo!" ?></p>
    </body>
</html>
```

Creo que queda bastante claro lo que hace la "linea" de programa.
Incluso es cuestionable el hecho de el resultado es exactamente el mismo de poner ** Hola Mundo! **.
Pero lo importante es entender que el código PHP esta embebido en el documento html.

Test 2: ¿¡HTML embebido en PHP!?

```
<html>
    <head>
        <title>Ejemplo de PHP.</title>
    </head>

    <body>
        <?php echo "<p>Hola Mundo!</p>" ?>
    </body>
</html>
```

Esta práctica se desaconseja, pero es posible e incluso probable que se use con cierta frecuencia para casos concretos donde es imposible o improductivo embeberlo en un documento xhtml.

Test 3: Algo un poco más complejo.

```
<html>
    <head>
        <title>Ejemplo de PHP.</title>
    </head>

    <body>
        <?php phpinfo() ?>
    </body>
</html>
```

Y ahora?... esa línea si que rinde!!...

Y bastante útil para saber de que tecnologías podemos valernos dentro del servidor que se nos alquila (hosting).

El resultado es el volcado de toda la información disponible del servidor y del compilador de PHP.

Tip: Comentarios

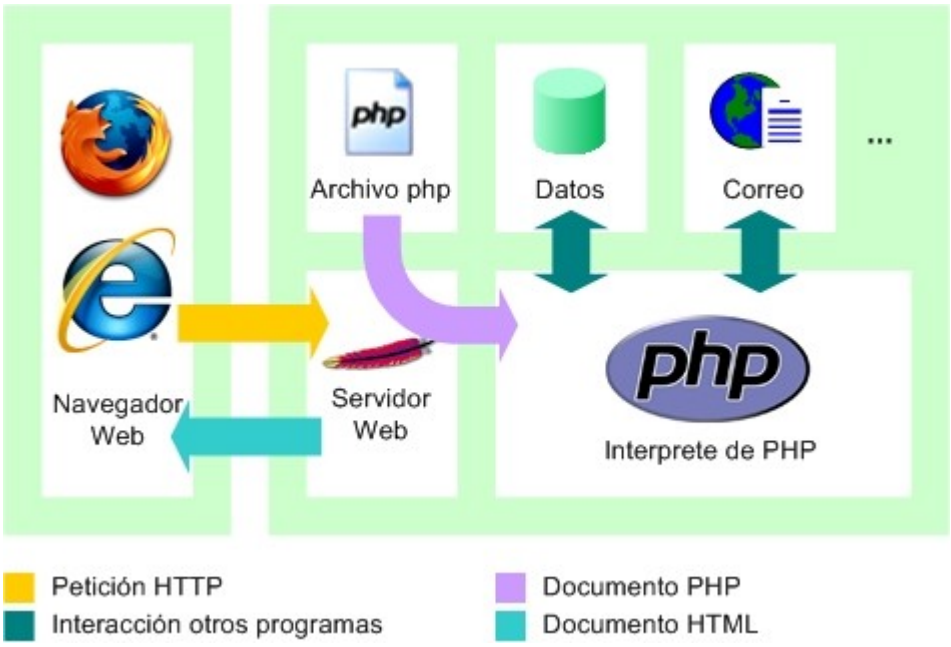
```
<?php

/* Este es un comentario de una sola línea. */

/* Este es un testamente de comentario
 * con varias líneas... */

?>
```

Estructura del Servidor web con algunos servicios básicos.



Este esquema esta más completo y es más preciso que el anterior. Solo queda mencionar que un servidor web pude tener muchos interpretes, uno de php, otro de python, otro de asp.net, etc, etc, por eso es muy necesario incluir en la etiqueta el nombre del lenguaje <?php … ?>.

Envíos de datos al servidor.

Envíos mediante POST.

La etiqueta para armar un formulario en xhtml que será enviado al servidor mediante el método post es:

<form action=”guardar.php” method=”post”>

</form>

La ventaja de enviar los datos mediante post es que viajan ocultos dentro de la cabecera y no son leíbles a simple vista. Los formularios por post son usados obligatoriamente por ej, en los formularios de login.

Envíos mediante GET.

La etiqueta para armar un formulario en xhtml que envié datos al servidor mediante el método get es:

<form action=”guardar.php” method=”get”>

</form>

El problema (o ventaja, depende) que tiene el método GET es que los datos ingresados en el formulario viajan en la URL, por lo cual pueden ser leídos en la barra de navegación del navegador de internet.

Las URL con los datos se concatenan de la siguiente manera:

www.misitio.com/script.php?nombre=Marcos&apellido=Henning

Como ven los datos son fácilmente leíbles por ello es totalmente desaconsejable utilizar GET como método para transferir datos confidenciales como las contraseñas.

Qué pasa cuando tenemos acentos o signos de interrogación en nuestros datos a enviar por get?. Hay toda una norma para convertir esos caracteres especiales a URL. A nuestros fines, utilizaremos un comando en php llamado urlencode($url); que hará el trabajo de codificar la url escrita "a lo bestia" a los caracteres correspondientes para una URL.

Envíos mixtos, POST y GET.

Hay ocasiones en donde necesitamos enviar datos por GET y por POST. Ej:

```
<form action="despachante.php?accion=guardar&usuario=juan" method="post">
</form>
```

En este caso estaríamos enviando la variable: accion, usuario más todos los campos que se especifiquen dentro del formulario.

Es un método práctico muchas veces para no tener que recurrir a los campos de texto ocultos en un formulario.

¿Como definir los nombres de campos?

Importante!, el atributo name de los input, select, textarea y demás es el identificador que se utilizará para recuperar los datos que ingreso el usuario. No hay que definir atributos name iguales dentro del mismo formulario ni olvidarse de definirlos.

```
<form action="guardar.php" method="get">
        <input type="text" name="nombre" />
        <input type="text" name="apellido" />
        <input type="submit" />
</form>
```

Para explicar como llegan los datos al servidor tomaremos como ejemplo que el usuario escribio en el formulario "Silvio" "Gonzales" en los campos.

¿Como llegan los datos al servidor?

El array $_GET.

Los datos que recopila el servidor son:

$_GET['nombre'] → "Silvio"

$_GET['apellido'] → "Gonzales"

El Array $_POST.

En caso de que hubiéramos especificado el formulario con método post. El array sería el siguiente.

$_POST['nombre'] → "Silvio"

$_POST['apellido'] → "Gonzales"

Mezclando POST y GET.

```
<form action=”despachante.php?accion=guardar&usuario=juan” method=”post”>
        <input type=”text” name=”nombre” />
        <input type=”text” name=”apellido” />
        <input type=”submit” />
</form>
```

En $_GET tendríamos:

$_GET['accion'] → “guardar”

$_GET['usuario'] → “juan”

Y en $_POST tendríamos:

$_POST['nombre'] → “Silvio”

$_POST['apellido'] → “Gonzales”

Otros arrays interesantes son $_SERVER, $_COOKIE, $_SESSION y $_FILES, los veremos más adelante.

NOTA: Enviando archivos al servidor:

```
<form action=”guardarArchivo.php” method=”post” enctype="multipart/form-data" >
<input type=”file” name=”archivo” />
</form>
```

Hay que especificar el encode type a multipartes. O no va a llegar nuestro archivo porque no entra en la cabecera de transferencia del xhtml.

Manejando Sesiones.

Las sesiones son un método seguro y eficaz de guardar y mantener datos del usuario durante toda su visita. Podemos guardar por ejemplo sus datos una vez que se autentifico en el sitio.

IMPORTANTE: El manejo de sesiones debe hacerse antes de enviar cualquier carácter al navegador del cliente. Cualquier trozo de html o algún echo perdido hará que sea imposible modificar la cabecera donde se guardan los valores de sesión.

Guardar Variables de Sesión.

```
<?php
session_start();
$_SESSION[´usuario´] = “Silvio Gonzales”;
?>
```

Restaurar Variables de Sesión.

```
<?php
session_start();
echo $_SESSION[´usuario´];
?>
```

Muchas Más Info: http://us3.php.net/manual/en/book.session.php

Manejo de Cookies.

El uso de cookies es muy frecuente para cosas como recordar la contraseña de una área de usuarios, saber cuantas visitas hemos recibido de un mismo usuario o para distinguir a ese usuario entre otros muchos.

IMPORTANTE: El manejo de cookies debe hacerse antes de enviar cualquier carácter al navegador del cliente. Cualquier trozo de html o algún echo perdido hará que sea imposible modificar la cabecera donde se guardan los valores de los cookies.

Para asignar un cookie utilizamos la siguiente función setcookie(nombre, valor, tiempo_expiración) Ej:

```
<?php
setcookie("usuario",”Silvio Gonzales”,time()+30*24*60*60);
?>
```

Con esta sentencia pondremos una cookie llamada usuario, con el valor “Silvio Gonzales” y que expirará dentro de 30 días. También tenemos que tener en cuenta que la cookie no la tendremos disponible hasta que el usuario recargue la página.

Para leer una cookie lo podemos hacer con la variable $_COOKIE['nombrecookie']. Por ejemplo:

```
<?php
echo $_COOKIE['usuario'];
?>
```

Mucha Más Info: http://us3.php.net/manual/en/features.cookies.php

Archivos enviados por el usuario.

Para manejar los archivos en php vamos a recurrir al array $_FILES que contiene todos los datos del archivo/s en cuestión. Primero empecemos creando el formulario, enviar_archivo.html

```
<html>
<body>
   <form action="cargar_archivo.php" method="post” enctype="multipart/form-data">
      <label for="file">Archivo:</label>
      <input type="file" name="archivo" id="archivo" />
      <br />
      <input type="submit" name="submit" value="Submit" />
   </form>
</body>
</html>
```

Manejando el archivo enviado en PHP.

El formulario anterior llama a cargar_archivo.php para que post-procese la información, nuestro archivo PHP entonces podría ser algo similar a esto:

```
<?php
   if ($_FILES["archivo"]["error"] > 0){
         echo "Ups!...Error: " . $_FILES["archivo"]["error"] . "<br />";
   }
   else{
      echo "Nombre Original del Archivo: " . $_FILES["archivo"]["name"] . "<br />";
      echo "Tipo de archivo (MIME-TYPE): " . $_FILES["archivo"]["type"] . "<br />";
      echo "Tamaño: " . ($_FILES["archivo"]["size"] / 1024) . " Kb<br />";
      echo "Almacenado temporalmente en: " . $_FILES["archivo"]["tmp_name"];
   }
?>
```

Como muchos usuarios pueden estar levantando simultáneamente archivos con el mismo nombre, PHP asigna un nombre único dentro del directorio temporal del servidor web. Es trabajo del programador llevar ese archivo a la posición correcta dentro del directorio que se desee o bien volcarlo a una base de datos.

Con toda la información que tenemos sobre el archivo podemos crear los validadores, por ejemplo si type == “image/jpeg” sabemos que es un archivo de imagen y mas precisamente un jpg. Con size podemos limitar el tamaño de los archivos que nuestro sitio esa recibiendo, igualmente, el servidor tiene un límite que suele ser de 2mb como máximo. Si se desea cambiar esto hay que modificar el archivo de configuración de php el php.ini dentro de la sección recursos dice upload_max_filesize = 2M.

Si necesitan copiar, mover, recorrer, etc, etc un archivo o un directorio, vean las funciones de “File System” de php: http://ar2.php.net/manual/en/ref.filesystem.php o bien de la api que les deje :-D