Materia: Programación Web II

Profesor: Hernán Roldán

</>

¿Cómo funciona la web?

- La Web se basa en peticiones o consultas que hace un **cliente** hacia un **servidor web** que se encuentra alojado en algún lugar.

- Ese cliente se comunica con un servidor mediante un **protocolo** que se llama **HTTP** o Protocolo de Transferencia de Hipertexto *(Hypertext Transfer Protocol)*.

- Mediante este protocolo, el cliente realiza consultas (**requests**) y recibe respuestas (**responses**).

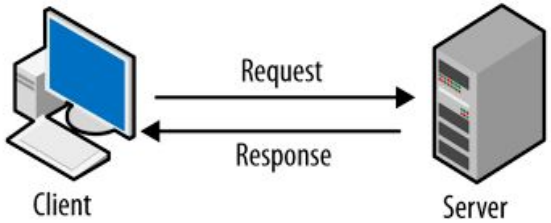
¿Cómo funciona la web? (cont.)

Clientes

En una arquitectura Cliente-Servidor, un navegador es la mejor representación de un cliente que solicita pedidos a un servidor. El cliente comprende las respuestas provenientes del servidor, las interpreta y muestra en pantalla al usuario.

Servidores

Un servidor web o servidor HTTP representa al servidor que se encarga de procesar los pedidos de distintos clientes, como ser usuarios que realizan consultas a través de una aplicación web. Este tiene la capacidad de recibir y administrar pedidos, para determinar la respuesta a enviar.



</>

¿Cómo funciona la web? (cont.)

URL

- Es el Localizador de Recursos Uniforme *(Uniform Resource Locator)*.

- Es la dirección específica que se asigna a cada uno de los recursos disponibles en una red con la finalidad de que estos puedan ser localizados o identificados.

- Cada recurso (sea una página completa, una imagen, un documento, etc.) tiene su propia url.

</>

Estructura de una URL

</>

Códigos de respuesta HTTP

Los códigos de estado de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica. Las respuestas se agrupan en cinco clases.

1. **Respuestas informativas (100-199),**
2. **Respuestas satisfactorias (200-299),**
3. **Redirecciones (300-399),**
4. **Errores de los clientes (400-499),**
5. **y errores de los servidores (500-599).**

Lectura recomendada:

Códigos de estado de respuesta HTTP (MDN Web Docs)

</>

Programación del lado del servidor: ¿Qué es?

La programación del lado del servidor *(Server Side Programming)* es una tecnología que consiste en el procesamiento de una petición de un usuario mediante la interpretación de un script en el servidor web para generar dinámicamente páginas HTML como respuesta.

</>

Programación del lado del servidor: Uso más común

La mayoría de los grandes sitios web usan código de lado servidor para presentar, cuando se necesitan, diferentes datos, generalmente extraídos de una base de datos almacenada en un servidor y enviada al cliente para ser presentada mediante algún código (ejemplo, HTML y JavaScript). Quizás el beneficio más significativo de la codificación del lado servidor es que nos permite confeccionar el contenido del sitio web para usuarios individuales. Los sitios dinámicos pueden resaltar contenido qué es más relevante basándose en las preferencias del usuario y sus hábitos.

</>

Sitios estáticos vs. sitios dinámicos

Sitios estáticos = del lado del cliente.

Un sitio estático es aquél que devuelve desde el servidor el mismo contenido insertado en el código “hard coded” siempre que se solicita un recurso en particular.

Sitios dinámicos = del lado del servidor.

Un sitio dinámico es aquél en que algún contenido de la respuesta está generado dinámicamente sólo cuando se necesita, y puede devolver datos diferentes para una URL basados en la información proporcionada por el usuario o sus preferencias almacenadas.

</>

Lenguajes de programación del lado del servidor

Un lenguaje de programación del lado del servidor es aquel que se ejecuta en el servidor web, **justo antes de que se envíe la página a través de Internet al cliente**.

Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución del script.

Algo importante a saber es que, como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

</>

Esquema de modelo cliente/servidor

Introducción al lenguaje de programación PHP

PHP: ¿Qué es?

PHP es el acrónimo recursivo de **PHP: Hypertext Preprocessor**, inicialmente se lo conoció como **Personal Home Page Tools**, herramienta para la creación de páginas web personales.

Es un lenguaje de programación de código abierto del lado del servidor, muy popular y especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

Fue creado inicialmente por el programador danés-canadiense Rasmus Lerdorf en 1994.

PHP embebido en HTML ¿ehhh?

Bien, ¿pero qué significa eso de que puede ser incrustado en HTML? Echemos un vistazo.

```
1  <!DOCTYPE html>
2  <html lang="en" dir="ltr">
3    <head>
4      <meta charset="utf-8">
5      <title></title>
6    </head>
7    <body>
8
9      <?php  echo "Hola, soy código PHP embebido en HTML"; ?>
10
11   </body>
12 </html>
```

PHP: ¿Para qué sirve?

PHP es un lenguaje de programación del lado del servidor, una herramienta poderosa para crear páginas web dinámicas e interactivas.

PHP es una alternativa eficiente, gratuita y ampliamente utilizada frente a competidores como ASP de Microsoft.

¿Qué puedo hacer con PHP?

Cualquier cosa. PHP está enfocado principalmente a la programación de scripts del lado del servidor, por lo que se puede hacer cualquier cosa que pueda hacer otro programa CGI, como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Aunque PHP puede hacer mucho más.

Existen tres campos principales donde se usan scripts de PHP, estos son: **scripts del lado del servidor** (y que es el campo más tradicional y el foco principal de dicho lenguaje de programación); **scripts desde la línea de comandos** y también para **escribir aplicaciones de escritorio**.

Repasemos rápidamente cada uno de esos.

¿Qué puedo hacer con PHP? (cont.)

Scripts del lado del servidor.

Este es el campo más tradicional y el foco principal. Son necesarias tres cosas para que esto funcione: **el analizador de PHP** (módulo CGI o servidor), **un servidor web** y un **navegador web**. Es necesario ejecutar el servidor con una instalación de PHP conectada.

¿Qué puedo hacer con PHP? (cont.)

Scripts desde la línea de comandos.

Se puede crear un script de PHP y ejecutarlo sin necesidad de un servidor web o navegador. Para utilizarlo de esta manera solo es necesario el analizador de PHP. Este tipo de uso es ideal para scripts que se ejecuten con regularidad empleando cron (en Linux) o el planificador de tareas (en Windows). Estos scripts también pueden usarse para tareas simples de procesamiento de texto.

¿Qué puedo hacer con PHP? (cont.)

Escribir aplicaciones de escritorio.

Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisieran utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar PHP-GTK para escribir dichos programas.

¿Quiénes usan PHP?

Alguno sitios web globales que usan PHP son:

- **Facebook**
- **Wikipedia**
- **Tumblr**
- **Slack**
- **MailChimp**
- **Etsy**
- **WordPress**
- y muchísimas otros.

¿Qué herramientas necesito para programar en PHP?

PHP: Herramientas

Para programar en PHP, necesitas tener un mínimo de herramientas a disposición, y estas son:

- **Servidor web:** Se necesita un servidor web como Apache. Lo más recomendable es usar herramientas como XAMPP, WAMP, o MAMP para configurarlo fácilmente en la máquina local.
- **PHP:** Instalar PHP en el servidor local.
- **Base de datos:** Generalmente se usa MySQL o MariaDB para almacenar y gestionar datos.
- **Editor de código:** Un buen editor de código como Visual Studio Code, Sublime Text, o PHPStorm y que facilita la escritura de código y el manejo de archivos PHP.

PHP: Herramientas (cont.)

Paquetes de Software (incluye Apache, PHP, MySQL):

XAMPP

WAMP

MAMP

Editores de Código (recomendados)

PHPStorm

Visual Studio Code

PHP: Herramientas (cont.)

Videotutoriales (paso a paso para la instalación de los paquetes de software):

XAMPP

WAMP

MAMP

Sintaxis

PHP: Sintaxis

Como dijimos anteriormente, un script PHP se ejecuta en el servidor y envía como resultado un archivo HTML al cliente (navegador).

- Un script PHP se puede colocar en cualquier parte del documento.
- Un script PHP comienza con **<?php** y termina con **?>**, llamados tags de apertura y cierre.
- La extensión por defecto de un archivo PHP es “**.php**”.

La sintaxis básica de PHP es la siguiente:

```
<?php

	// Todo el código va aquí.

?>
```

Comentarios

PHP: Comentarios

Un comentario en el código PHP es una línea que no se ejecuta como parte del programa, y su único propósito es que lo lea alguien que esté mirando el código. Idealmente, los comentarios sirven de ayuda para entender lo que el código hace.

El principal uso de los comentarios es para:

- Ayudar a que otros que estén leyendo el código entiendan lo que este hace.
- Ayudarnos a recordar lo que hicimos tiempo atrás. Suele suceder que pasado un tiempo nos olvidamos lo que nuestro propio código hace, y por eso es necesario comentarlo.

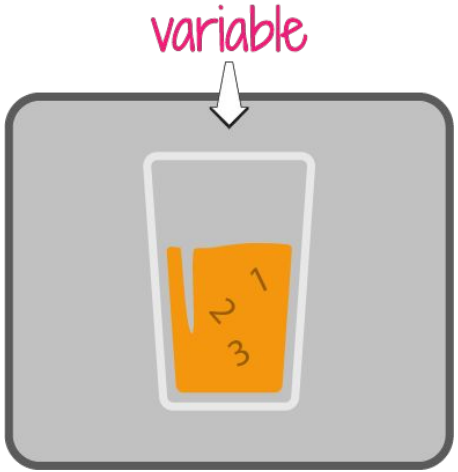
PHP soporta diferentes maneras para comentar el código:

- **//** Comentario de una única línea.
- **#** Comentario de una única línea (con la almohadilla “#”, es el menos usado de los dos).
- **/* */** Comentario de múltiples líneas.

Variables

Variables: ¿Qué son?

Simple! Piensa en las variables como contenedores para almacenar datos.



Reglas para las variables en PHP

Una variable puede tener un nombre corto como “x” e “y” o un nombre más descriptivo como “edad”, “modelo_auto”, “nombre”, “apellido”, “total”, etc).

Reglas de las variables PHP

- Una variable comienza con el signo dólar ($), seguido del nombre de la variable.
- Una variable debe comenzar con una letra o guión bajo.
- Una variable no puede comenzar con un número.
- Una variable solo puede contener caracteres alfanuméricos y guión bajo (A-z, 0-9, y _).
- Los nombres de las variables son sensibles a mayúsculas ($edad y $EDAD son dos variables diferentes).

Alcance de las Variables

Alcance de las variables

En PHP las variables se pueden declarar en cualquier parte del script.

El alcance de la variable (o ámbito de la variable) es la parte del script donde se puede hacer referencia/uso de la variable.

PHP tiene tres alcances de variables diferentes: **local**; **global** y **static**.

Nota: En programación, el ámbito de una variable es la parte del código desde donde se puede acceder a una variable. Básicamente una variable puede ser global (se puede acceder desde cualquier parte del código) o local (sólo se puede acceder desde una porción del código).

Variable de alcance global y local

Variable global:

Una variable declarada fuera de una función tiene un ALCANCE GLOBAL y solo se puede acceder fuera de una función.

Variable local:

Una variable declarada dentro de una función tiene un ALCANCE LOCAL y solo se puede acceder a ella dentro de esa función.

Variable de alcance global y local (cont.)

Variable con mismo nombre en diferentes partes:

También se pueden tener variables locales con el mismo nombre en diferentes funciones, porque las variables locales sólo son reconocidas por la función en la que están declaradas.

Variable de alcance global y local (cont.)

La palabra reservada “global”:

La palabra reservada “**global**” se utiliza para acceder a una variable global desde dentro de una función.

Para hacer esto, la palabra global se debe usar antes de las variables definidas dentro de la función, es decir, las propias de la función.

PHP también almacena todas las variables globales en una matriz llamada **$GLOBALS[*índice*]**. El índice contiene el nombre de la variable.

Variable de alcance global y local (cont.)

La palabra reservada “static”:

Normalmente, cuando se completa / ejecuta una función, se eliminan todas sus variables. Sin embargo, a veces queremos que NO se elimine una variable local, ya que la necesitamos para otra tarea.

Para hacer esto, se debe usar la palabra reservada **static** cuando se declare la variable por primera vez.

Sentencias Echo y Print

Sentencias Echo y Print

Con PHP existen dos maneras de imprimir (mostrar) texto en pantalla y es mediante el uso de las sentencias **echo** y **print**.

echo y **print** son muy parecidas y ambas pueden usarse para mostrar texto en pantalla, pero tienen pequeñas diferencias.

- **echo** no tiene un valor de retorno, mientras que **print** tiene un valor de retorno de 1, por lo que puede usarse en expresiones.
- **echo** puede tomar varios parámetros, **print** solo puede tomar uno.
- **echo** es ligeramente más rápido que **print**.
- Tanto **echo** como con **print** se pueden usar con o sin paréntesis, pero en el caso de **echo** los paréntesis se pueden usar si y sólo si se pasa un único parámetro.

Nota: **echo** suele ser la sentencia más usada entre ambas.

Tipos de Datos

Algo importante a saber sobre el tipado de datos

PHP es un lenguaje de programación débilmente tipado *(Loosely Typed Language)*. Esto significa que PHP asocia automáticamente un tipo de datos a la variable, dependiendo de su valor. Dado que los tipos de datos no se establecen en un sentido estricto, puede hacer cosas como agregar una cadena a un número entero sin causar un error.

Nota: En PHP 7, se agregaron declaraciones de tipo. Esto da una opción para especificar el tipo de datos esperado al declarar una función, y al habilitar el requisito estricto, arrojará un "Error fatal" en una falta de coincidencia de tipos.

Tipos de Datos en PHP

PHP admite diez tipos de datos primitivos agrupados de la siguiente manera:

Escalares

- Boolean
- Integer
- Float
- String

Compuestos

- Array
- Object
- Callable
- Iterable

Especiales

- Resource
- NULL

Nota: Por el momento nos centraremos en los tipos de datos Boolean, Integer, Float, String y Array.

Ok... ¿pero qué son los tipos de datos primitivos?

Se llama tipo primitivo o tipo elemental a los tipos de datos originales de un lenguaje de programación.

Generalmente ejemplos de datos primitivos son: **Char** (caracter); **Int** (entero); **Float** (coma flotante) y **Boolean** (lógico).

Ejemplos de tipos de datos primitivos en PHP

Vista en el IDE (declaración)

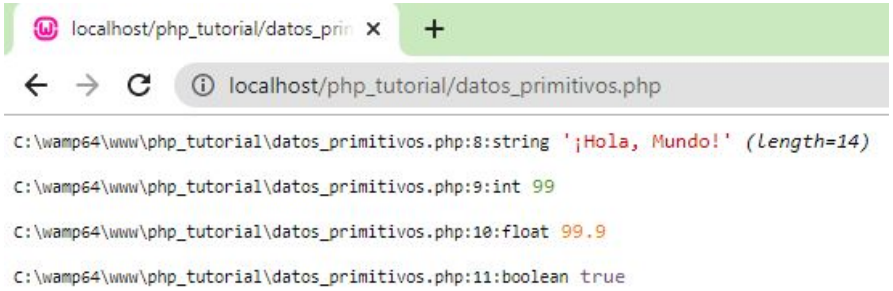
```php
<?php

  $primitivo_string = "¡Hola, Mundo!";
  $primitivo_integer = 99;
  $primitivo_float = 99.9;
  $primitivo_boolean = TRUE;

  var_dump($primitivo_string);
  var_dump($primitivo_integer);
  var_dump($primitivo_float);
  var_dump($primitivo_boolean);

?>
```

Vista en el Navegador (ejecución)



Recomendación: Ver diferencia entre IDE y Editor de Código.

Tipos de datos: String

Una cadena (**string**) es una secuencia de caracteres, como “¡Hola Mundo!”.

Una cadena puede ser cualquier texto entre comillas, que puede estar entre comillas simples o dobles.

```php
<?php
    $ejemplo1 = “¡Hola Mundo!”;
    $ejemplo2 = ‘¡Hola Mundo!’;
?>
```

Tipos de datos: Integer

Un tipo de dato entero (**integer**) es un número no decimal comprendido entre -2,147,483,648 y 2,147,483,647.

Reglas para enteros:

- **Un número entero debe tener al menos un dígito.**
- **Un número entero no debe tener un punto decimal.**
- **Un número entero puede ser positivo o negativo.**
- **Los números enteros se pueden especificar en notación decimal (base 10), hexadecimal (base 16), octal (base 8) o binaria (base 2).**

```php
<?php

    $num = 1001;

?>
```

Tipos de datos: Float

Un flotante (**float**) es un número con un punto decimal o un número en forma exponencial.

```php
<?php
    $num_decimal = 965.50;
?>
```

Tipos de datos: Boolean

Un booleano (**boolean**) representa dos posibles estados: **VERDADERO** o **FALSO.**

```php
<?php
    $x = true;
    $y = false;
?>
```

Nota: Los booleanos se suelen usar en pruebas condicionales.

Tipos de datos: Array

Los arreglos (**arrays**) son matrices que almacenan muchos datos en una misma variable.

```php
<?php
    $marvel_superheroes = array(“Spiderman”, “Captain America”);
    $dc_superheroes = array(“Superman”, “Batman”);
?>
```

Tipos de Arrays

PHP soporta 3 (tres) tipos de Arrays y son los siguientes:

- **Indexado o numérico:** se accede a los elementos mediante su posición index (número). En estos Arrays la primera posición es 0 (cero),por lo que un Array con 3 (tres) elementos tendrá las siguientes posiciones: (0) para el primer elemento, (1) para el segundo elemento y (2) para el tercer elemento.
- **Asociativo:** se accede a los elementos mediante la llave que los referencia, estos Arrays están compuestos por lo que se conoce como "Llave/Valor".
- **Multidimensional:** son Arrays que contienen otros Arrays, y a su vez estos Arrays contenidos pueden ser numéricos, multidimensionales o ambos.

Tipos de datos: Object

Las clases y los objetos son los dos aspectos principales de la programación orientada a objetos. Una clase es un molde que representa por ejemplo a un objeto de la vida real, como ser, un auto, una persona, un animal, etc., y un objeto es una instancia de una clase.

Nota: Abordaremos este tipo de datos cuando estemos más familiarizados con el paradigma de programación orientado a objetos (POO).

Tipos de datos: Null

Nulo (**null**) es un tipo de datos especial que solo puede tener un valor: **null**.

Una variable de tipo de datos **null** es una variable que no tiene ningún valor asignado.

Nota: Si se crea una variable sin un valor, se le asigna automáticamente un valor de NULL.

Las variables también se pueden vaciar estableciendo el valor en NULL:

```php
<?php
    $variable = ¡Hola Mundo!;
    $variable = null;
    var_dump($variable) // Muestra null, ya que ahora $variable no contiene datos.
?>
```

Constantes

PHP: Constantes

Una constante es un identificador (nombre) de un valor simple. El valor no se puede cambiar durante la ejecución del script.

Un nombre válido de una constante comienza con una letra o un guión bajo sin el signo dólar ($) antes del nombre de la constante.

Nota: A diferencia de las variables, las constantes son automáticamente globales en todo el script.

PHP: Constantes (cont.)

Para crear una constante se usa la función **define()**, y su sintaxis es:

define(*name, value, case-insensitive*)

Parámetros:

- ***name:*** *Especifica el nombre de la constante.*
- ***value:*** *Especifica el valor de la constante.*
- ***case-insensitive:*** *Especifica cuando el nombre de la variable debe ser sensible a mayúsculas (case sensitive). Por defecto es false.*

PHP: Constantes (cont.)

Ejemplo constante case sensitive.

```
<?php
	define("FRASE", "Bienvenidos a la materia Programación II.");
	echo FRASE; // Muestra "Bienvenidos a la materia Programación II.").
	echo frase; // Muestra “frase”.
?>
```

Nota: Para hacer que la constante no sea sensible a mayúsculas es necesario establecer el parámetro “case-sensitive” en true.

Conociendo de qué tipo es la variable

PHP propone dos funciones muy útiles para conocer de qué tipo es la variable y qué información contiene ésta.

var_dump()

// Esta función obtiene el tipo de la variable y la información contenida en ésta.

gettype()

// Obtiene únicamente el tipo de la variable.

Nota: La función var_dump es la más usada y es ideal para debuguear código, ya que además de obtener el tipo de dato de la variable trae la información que esta contiene.

Funciones isset() y empty()

Función isset()

La función **isset()** comprueba si una variable está establecida, lo que significa que debe estar declarada y no ser NULL.

Esta función devuelve verdadero si la variable existe y no es NULL, de lo contrario, devuelve falso.

Función empty()

La función **empty()** comprueba si una variable está vacía o no.

Esta función devuelve falso si la variable existe y no está vacía, de lo contrario, devuelve verdadero.

include y require

declaraciones include y require

La declaración **include** o **require** toma todo el texto/código/marcado que existe en el archivo especificado y lo copia en el archivo que usa la declaración include/requiere.

Las declaraciones **include** y **require** son idénticas, excepto en caso de falla:

- **include** solo producirá una advertencia (E_WARNING) y el script continuará.
- **require** producirá un error fatal (E_COMPILE_ERROR) y detendrá el script.

Variables Súper globales

Variables Súper globales

Las variables súper globales son variables internas que están disponibles siempre en todos los ámbitos.

Dicho de otra manera, PHP tiene incorporadas algunas variables predefinidas que son “súper globales”, esto significa que están disponibles en todos los ámbitos a lo largo del script. No es necesario emplear **GLOBALS[$variable]** para acceder a ellas dentro de las funciones o métodos.

Variables Superglobales (cont.)

Las variables superglobales son:

- `$GLOBALS`
- `$_SERVER`
- `$_GET`
- `$_POST`
- `$_FILES`
- `$_COOKIE`
- `$_SESSION`
- `$_REQUEST`
- `$_ENV`

Variables Superglobales (cont.)

Variable $GLOBALS

$GLOBALS es una variable súper global de PHP que se utiliza para acceder a variables globales desde cualquier lugar del script PHP (también desde dentro de funciones o métodos).

PHP almacena todas las variables globales en una matriz llamada $GLOBALS[índice]. El índice contiene el nombre de la variable.

Variables Superglobales (cont.)

Variable $_SERVER

$_SERVER $_SERVER es una variable súper global de PHP que contiene información sobre encabezados, rutas y ubicaciones de scripts.

Variables Superglobales (cont.)

Variable $_POST

$_GET es una variable súper global de PHP que se utiliza para recopilar datos de formularios después de enviar un formulario HTML con method = "get". **$ _GET** también puede recopilar datos enviados en la URL.

Variables Superglobales (cont.)

Variable $_POST

$_POST es una variable súper global de PHP que se utiliza para recopilar datos después de enviar un formulario HTML con method = “post”. **$ _POST** también se usa ampliamente para pasar variables.

Variables Superglobales (cont.)

Variable $_REQUEST

$_REQUEST es una variable súper global de PHP que se utiliza para recopilar datos después de enviar un formulario HTML.

Variables Superglobales (cont.)

Variable $_SESSION

$_SESSION es una variable súper global de PHP que se utiliza para almacenar información que se utilizará en varias páginas.

A diferencia de una cookie, la información **no se almacena** en la computadora del usuario.

Ok, ¿pero qué es una sesión en PHP?

Por ejemplo, cuando trabajamos con una aplicación sin importar cuál sea, la abrimos, eventualmente realizamos algunos cambios y por último la cerramos. Esto que hicimos es muy parecido a una sesión, es decir, nuestra computadora sabe quiénes somos, sabe cuándo inicia la aplicación y cuándo finaliza. Pero en Internet hay un problema, el servidor web no sabe quiénes somos ni qué hacemos, porque la dirección HTTP no mantiene el estado.

Entonces, podemos decir que las variables de sesión resuelven este problema almacenando información del usuario.

</>

Ok, momento de ejercitar lo aprendido. Manos a la obra!

Información Adicional

Glosario de términos usados en los slides

Término	Descripción
IDE	Entorno de Desarrollo Integrado.
Browser	Navegador Web.
Server-Side Programming	Programación del lado del servidor.
Case-sensitive	Cuando un programa puede identificar si una letra está escrita en minúscula o mayúscula.
cron	Planificador de Tareas en Linux.
Task Scheduler	Planificador de Tareas en Windows.

Glosario de términos más usados (cont.)

Término	Descripción
CGI	CGI es un método mediante el cual un servidor web puede obtener datos de (o enviar datos a) bases de datos, documentos y otros programas, y presentar esos datos a los usuarios a través de la web. En otras palabras, CGI es un programa destinado a ejecutarse en la web.
PHP-GTK	Extensión para PHP para desarrollar aplicaciones de escritorio.

Diferencia entre IDE y Editor de Código

Básicamente, un entorno de desarrollo integrado o **IDE** es un paquete autónomo que permite escribir, compilar, ejecutar y depurar código en el mismo lugar. Por otro lado, un **editor de código** es un editor de texto con varias características que facilitan el proceso de escritura de código, ya sea a través de capacidades nativas o mediante complementos opcionales.

Generalmente, un **IDE** se centra en un solo lenguaje de programación y contiene el compilador / intérprete y el depurador específicos de ese lenguaje en cuestión. Por el contrario, los **editores de código** tienen capacidades más generales y pueden trabajar con varios lenguajes de programación. Los **editores de código** se limitan a escribir código y no van más allá de esta tarea.

¿Qué es CGI?

CGI significa Common Gateway Interface ("Interfaz de Entrada Común"), o lo que es lo mismo, Application Programming Interface. **CGI** no es ningún lenguaje de programación, sino una API de servidor web. Se trata de un sistema de comunicación que le dice al servidor web cómo enviar y recibir datos de una aplicación de servidor a un cliente. Esto permite a los servidores usar aplicaciones de servidor para realizar funciones concretas que añaden mayor interactividad a los sitios web, como formularios, acceso a bases de datos, login de usuarios, chats, etc. Fue una de las primeras formas para realizar sitios web dinámicos en Internet (actualmente se utiliza una variante mucho más rápida, **Fast-CGI**).

La mayoría (o todos) los servidores web se pueden configurar para ejecutar una aplicación o programa **CGI**. Esto significa que al recibir una petición, el servidor reenviará los datos a un programa **CGI** específico, configurando algunas variables de entorno y ordenando los parámetros para que el programa pueda saber dónde y qué buscar. Los programas **CGI** se ejecutan en el servidor web en cualquier lenguaje que pueda procesar variables de entorno y standard input (stdin) y escribir en standard output (stdout).

Lenguajes de programación de tipado fuerte

Un lenguaje de programación es fuertemente tipado si no se permiten violaciones de los tipos de datos, es decir, dado el valor de una variable de un tipo concreto, no se puede usar como si fuera de otro tipo distinto a menos que se haga una conversión. Cabe aclarar que no existe una única definición de este término. Un lenguaje que se dice que no está tipado se refiere a que no está fuertemente tipado.

Fuente: Wikipedia

Lenguajes de programación de tipado débil

Los lenguajes de programación no tipados o débilmente tipados no controlan los tipos de las variables que declaran, de este modo, es posible usar variables de cualquier tipo en un mismo escenario. Por ejemplo, una función puede recibir como parámetro un valor entero, cadena de caracteres, flotante, etc.

Nota: No hay que confundir el término con los lenguajes de tipos dinámicos, en los que los tipos de las variables se deciden en tiempo de ejecución, si bien es cierto que muchos lenguajes de programación de tipos dinámicos (en los que no se declaran los tipos de datos) son también no tipados.

Fuente: Wikipedia

Algunos ejemplos de IDEs y Editores de Código

IDEs

- PHPStorm
- Netbeans
- Aptana Studio
- Eclipse
- Visual Studio (with Xamarin)
- ZendStudio

y muchos más.

Editores de Código

- Sublime Text
- Visual Studio Code
- Atom
- Notepad++
- Coda
- Brackets
- Komodo Edit

y muchos más.