

Hoja de Referencia Rápida: PHP (Unidades 0-3)

Módulo: Desarrollo Web en Entorno Servidor

UD 0: Conceptos Clave (La Teoría)

- **Arquitectura Cliente-Servidor:**

- **Cliente:** Es el navegador web (Chrome, Firefox). Su trabajo es **PEDIR** páginas.
- **Servidor:** Es nuestro software (Apache en XAMPP). Su trabajo es **ESCUCHAR** peticiones, procesarlas (ejecutar PHP) y **RESPONDER** enviando HTML puro.

- **Lenguaje Interpretado (PHP):**

- PHP es "interpretado", lo que significa que el servidor Apache lee el código "línea a línea" y lo ejecuta *en el momento* cada vez que un usuario pide la página. No se crea un fichero '.exe' antes.

- **Protocolo HTTP:**

- Es el "idioma" oficial que usan el Cliente y el Servidor para hablar entre ellos. El cliente hace una "petición HTTP" y el servidor da una "respuesta HTTP".

UD 1: Entorno y Comandos (XAMPP y Git)

XAMPP (Tu Servidor Local)

Para programar en PHP, necesitas un entorno que simule un servidor. Eso es XAMPP.

1. **Panel de Control:** Antes de programar, **DEBES** iniciar el servicio (darle al "Start") de **Apache**. Apache es el software que actúa como servidor web. (MySQL lo usaremos más adelante para bases de datos).

2. **La Carpeta Mágica (htdocs):**

- Todos tus proyectos PHP deben guardarse obligatoriamente dentro de la carpeta C:\xampp\htdocs\
- Esta carpeta es la "raíz" pública de tu servidor.

3. **Acceso en el Navegador:**

- Para ver tus ficheros, NUNCA hagas doble clic en el archivo.
- Debes abrir tu navegador (Chrome) y escribir: `http://localhost/`
- **Ejemplo:** Si guardas tu fichero en C:\xampp\htdocs\mi_proyecto\index.php, lo abres en el navegador escribiendo `http://localhost/mi_proyecto/index.php`

Git (Control de Versiones) - El Flujo de Trabajo

Git sirve para "guardar partidas" de tu código. El flujo de trabajo básico (local -> remoto) es:

```
# PASO 1: (Opcional) Ver el estado de tus ficheros
git status

# PASO 2: Preparar los ficheros que quieras guardar (Añadirlos al "sobre")
# Añade un fichero específico
git add index.php
# O añade TODO lo que haya en la carpeta actual
git add .

# PASO 3: Guardar los cambios en tu repositorio LOCAL (Cerrar el "sobre" y
#           ponerle etiqueta)
# El mensaje es obligatorio y debe describir el cambio
git commit -m "He añadido la página de inicio"

# PASO 4: Subir tus cambios guardados al repositorio REMOTO (GitHub)
# 'origin' es el nombre por defecto de tu conexión a GitHub
# 'master' (o 'main') es la rama principal
git push origin master
```

UD 2: Sintaxis Básica (HTML + PHP)

Estructura HTML5 Mínima (Comentada)

Este es el "esqueleto" de cualquier página web.

```
<!DOCTYPE html> <!> Le dice al navegador que es HTML5 <>
<html lang="es"> <!> El inicio del documento, en español <>
<head>
    <meta charset="UTF-8"> <!> Permite que se vean bien las tildes y ñ <>
        <title>Título en la Pestaña</title> <!> Esto se ve en la pestaña del navegador
</head>
<body>
    <!>
        El BODY es la parte visible de la página.
        Todo tu contenido (y tu PHP) va aquí dentro.
    <>
</body>
</html>
```

Embeber (Insertar) PHP en HTML (Comentado)

El servidor **solo** ejecuta el código que está dentro de las etiquetas `<?php ... ?>`. El resto lo trata como HTML normal.

```

<!DOCTYPE html>
<html>
<head>
    <title>Mi Primer PHP</title>
</head>
<body>

    <!-- Esto es HTML normal. El servidor no lo toca, solo lo envía. -->
    <h1>Esto es HTML normal</h1>

    <?php
        // INICIO DEL BLOQUE PHP

        // 'echo' es la instrucción para imprimir texto o HTML en la página.
        // Fíjate que imprimimos una etiqueta HTML <p> desde PHP.
        echo "<p>¡Hola Mundo! Esto se imprime desde PHP.</p>";

        // FIN DEL BLOQUE PHP
    ?>

    <!-- Esto vuelve a ser HTML normal. -->
    <p>Esto vuelve a ser HTML normal.</p>

</body>
</html>

```

UD 3: Fundamentos de Programación PHP

1. Variables (Comentadas)

- Siempre empiezan con el símbolo \$
- Son sensibles a mayúsculas: \$miVariable es **diferente** de \$mivariable.

```

<?php
    // --- Tipos de Datos Escalares ---

    // String (Cadena de texto): Siempre entre comillas
    $nombre = "Juan";

    // Integer (Número entero): Sin comillas
    $edad = 25;

    // Float (Número decimal): Se usa un punto, no una coma
    $precio = 19.95;

    // Boolean (Verdadero o Falso): Sin comillas
    $es_valido = true;
    $esta_listo = false;
?>

```

2. Imprimir: echo (Comentado)

- echo es una construcción del lenguaje, no una función (normalmente no necesita paréntesis).
- Se usa para "imprimir" texto o HTML en la página.

```

<?php
    // Imprime texto simple
    echo "Hola Mundo";

    // Imprime una etiqueta HTML
    echo "<h1>Soy un Título</h1>";
?>

```

3. Concatenación (Unir cadenas) (Comentado)

- Se usa el **punto (.)** para unir cadenas y variables.

```

<?php
    $usuario = "Ana";

    // Se usa el punto (.) para unir "Bienvenida, " con el valor de
    // $usuario y con "."
    echo "Bienvenida, " . $usuario . ".";
    // Imprime en pantalla: Bienvenida, Ana.
?>

```

4. ¡La Trampa! Comillas Simples vs. Dobles (Comentado)

- **Comillas Simples ('):** Tratan todo como texto **literal**. NO interpretan variables.
- **Comillas Dobles (""):** **Interpretan** (evalúan) las variables que están dentro.

```

<?php
$fruta = "Manzana";

// Con comillas simples:
// Imprime literalmente: Mi fruta es: $fruta
echo 'Mi fruta es: $fruta';

echo "<br>"; // Un salto de línea HTML

// Con comillas dobles:
// Reemplaza $fruta por su valor ("Manzana")
// Imprime: Mi fruta es: Manzana
echo "Mi fruta es: $fruta";
?>
\end{LSTlisting}
\end{codebox}

\subsection*{5. Operadores Esenciales}
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{p{2.5cm} p{2.5cm} p{9cm}}
\toprule
\textbf{Tipo} & \textbf{Operador} & \textbf{Significado y Ejemplo} \\
\midrule
Asignación & \texttt{=} & Asigna un valor (ej: \texttt{\$a = 5}) \\
Comparación & \texttt{==} & ¿Son iguales en *valor*? (ej: \texttt{5 == "5"} es \textbf{TRUE}) \\
Comparación & \texttt{==>} & ¿Son idénticos en *valor Y tipo*? (ej: \texttt{5 == "5"} es \textbf{FALSE}) \\
Comparación & \texttt{!=} & ¿Son diferentes en valor? \\
Comparación & \texttt{>} & Mayor que \\
Comparación & \texttt{<} & Menor que \\
Lógico & \texttt{&&} & \textbf{AND} (ej: \texttt{if (\$edad > 18 && \$pais == "España")}) \\
Lógico & \texttt{||} & \textbf{OR} (ej: \texttt{if (\$edad == 1 || \$b == 1)}) \\
Lógico & \texttt{!} & \textbf{NOT} (ej: \texttt{if (!$es_valido)}) \\
\bottomrule
\end{tabular}
\end{codebox}

\subsection*{6. Estructuras de Control (Decisión) (Comentadas)}

\textbf{if / elseif / else}:
Se usa para tomar decisiones. El código se ejecuta de arriba abajo y entra en el \textit{primer} bloque que sea verdadero.
\begin{codebox}
\begin{lstlisting}[style=phpstyle]
<?php
$edad = 17;

if ($edad >= 18) {
    // Este bloque se ejecuta si $edad es 18 o más
    echo "Eres mayor de edad.";
} elseif ($edad >= 13) {
    // Este bloque solo se comprueba si el 'if' de arriba fue falso
    // Se ejecuta si $edad está entre 13 y 17
    echo "Eres un adolescente.";
} else {
    // Este bloque se ejecuta si NINGUNO de los anteriores fue
    // verdadero
    echo "Eres un niño.";
}
?>

```

switch: Útil para comprobar una *única* variable contra muchos valores exactos.

```
<?php
    $color = "verde";

    switch ($color) {
        case "rojo":
            echo "El color es rojo.";
            break; // 'break' es crucial, evita que "caiga" al siguiente
        case "verde":
            echo "El color es verde.";
            break;
        case "azul":
            echo "El color es azul.";
            break;
        default:
            // Se ejecuta si ninguno de los 'case' anteriores coincide
            echo "Es otro color.";
            break;
    }
?>
```

7. Estructuras de Control (Repetición) (Comentadas)

while: Repite un bloque de código MIENTRAS la condición sea verdadera.

```
<?php
    // 1. Inicializar un contador (¡fuera del bucle!)
    $i = 1;

    // 2. Condición: El bucle se ejecutará mientras $i sea 5 o menos
    while ($i <= 5) {
        // 3. Acción a repetir:
        echo "El número es: " . $i . "<br>";

        // 4. ¡¡MUY IMPORTANTE!! Incrementar el contador
        // Si olvidas esta línea, $i siempre será 1 y tendrás un BUCLE
        // INFINITO
        $i++;
    }
    // Cuando $i llega a 6, la condición ($i <= 5) es falsa y el bucle
    // termina.
?>
```

Aplicación Práctica (UD3): while con HTML Así se usa un bucle para generar HTML dinámicamente.

```
<select name="numeros">
<?php
    $i = 1;
    while ($i <= 10) {
        // Imprimimos una etiqueta <option> por cada vuelta
        // Usamos comillas dobles para que $i se interprete
        echo "<option value='$i'>$i</option>";

        // Incrementamos el contador
        $i++;
    }
?>
</select>
```