1.3. Estructuras de control: condicionales y bucles

LENGUAJES DE PROGRAMACIÓN NIVEL 4. UNIDAD 1

PHP - LENGUAJE DE PROGRAMACION PARA EL BACKEND

Financiado por la Unión Europea

Plan de Recuperación, Transformación y Resiliencia

Contenido

Introducción

Las estructuras de control son herramientas poderosas en PHP que nos permiten añadir lógica y fluidez a nuestros programas. En esencia, nos permiten dictar cómo y cuándo se ejecutará cierto código en función de condiciones específicas o repetir acciones múltiples veces. Estas estructuras son las que realmente transforman nuestros scripts estáticos en aplicaciones dinámicas e interactivas, permitiéndonos:

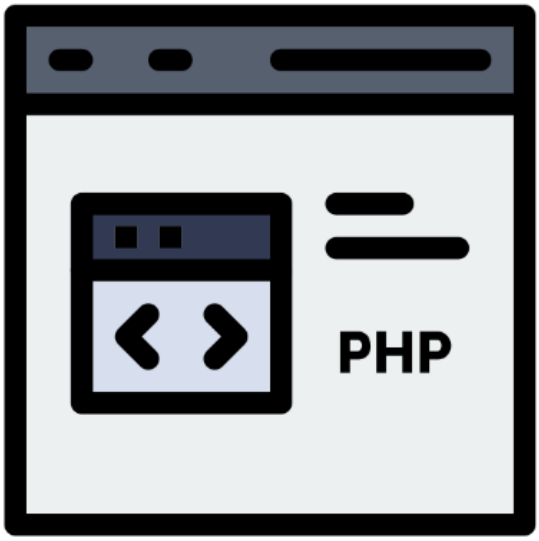
- Tomar Decisiones: Mediante el uso de estructuras condicionales como if, else y elseif, nuestros programas pueden tomar decisiones basadas en la información disponible. Esto es similar a cómo tomamos decisiones en la vida cotidiana: si se cumple una condición, realizamos una acción; de lo contrario, hacemos algo diferente.
- Repetir Acciones: Las estructuras iterativas o bucles, como for, while y foreach, nos permiten ejecutar un bloque de código repetidamente, lo cual es increíblemente útil para tareas como procesar elementos de un array o ejecutar una acción hasta que se cumpla una condición.

Al dominar las estructuras de control, estarás equipado para abordar una amplia gama de problemas de programación y construir aplicaciones web más complejas y funcionales. Estas estructuras son fundamentales para realizar tareas como validar formularios de usuario, manejar datos de bases de datos, construir interfaces de usuario interactivas y mucho más.

En los siguientes módulos de este curso, veremos cómo estas estructuras de control se aplican en escenarios reales, ampliando aún más tus habilidades de desarrollo en PHP.

¡Comencemos!

Estructuras condicionales



En el corazón de casi todos los programas hay decisiones. Las estructuras condicionales en PHP nos permiten tomar esas decisiones dirigiendo el flujo de nuestros programas de manera lógica, ejecutando diferentes bloques de código basados en condiciones específicas.

Este módulo explora las sentencias if, else, elseif, y el operador ternario, herramientas fundamentales para la toma de decisiones en PHP.

Sentencia If

La sentencia if es la forma más básica de control de flujo en PHP. Se utiliza para ejecutar un bloque de código solo si una condición dada es verdadera.

Sintaxis:

```
if (condición) {
    // Código a ejecutar si la condición es verdadera
}
```

[1]

[1] Accesibilidad
if (condición) {
// Código a ejecutar si la condición es verdadera
}

Ejemplo:

```php
$edad = 20;
if ($edad >= 18) {
    echo "Eres mayor de edad.";
}
```

2

Este código verificará si la variable $edad es 18 o más. Si es así, imprimirá "Eres mayor de edad".

Sentencia Else

La sentencia else se usa en combinación con if para ejecutar un bloque de código alternativo cuando la condición if no se cumple.

Sintaxis:

```php
if (condición) {
    // Código a ejecutar si la condición es verdadera
} else {
    // Código a ejecutar si la condición es falsa
}
```

3

[2] Accesibilidad

```
$edad = 20;
if ($edad >= 18) {
    echo "Eres mayor de edad.";
}
```

[3] Accesibilidad

```
if (condición) {
        // Código a ejecutar si la condición es verdadera
} else {
        // Código a ejecutar si la condición es falsa
}
```

Ejemplo:

```php
$edad = 16;
if ($edad >= 18) {
    echo "Eres mayor de edad.";
} else {
    echo "No eres mayor de edad.";
}
```

4

Este código imprimirá "No eres mayor de edad" si $edad es menor que 18.

Sentencia Elseif

elseif permite verificar múltiples condiciones en secuencia. Si la condición if no se cumple, PHP verificará la siguiente condición elseif, y así sucesivamente.

Sintaxis:

```php
if (condición1) {
    // Código a ejecutar si condición1 es verdadera
} elseif (condición2) {
    // Código a ejecutar si condición2 es verdadera
} else {
    // Código a ejecutar si ninguna de las condiciones anteriores es verdadera
}
```

5

---

[4] Accesibilidad

```
$edad = 16;
if ($edad >= 18) {
        echo "Eres mayor de edad”;
} else {
        echo "No eres mayor de edad.";
}
```

[5] Accesibilidad

```
if (condicion1) {
        // Código a ejecutar si la Condición1 es verdadera
} elseif (condicion2) {
        // Código a ejecutar si la Condición2 es verdadera
} else {
        // Código a ejecutar cuando ninguna de las condiciones anteriores es verdadera
}
```

Ejemplo:

```php
$calificacion = 85;
if ($calificacion >= 90) {
    echo "Excelente";
} elseif ($calificacion >= 80) {
    echo "Muy bien";
} else {
    echo "Necesitas mejorar";
}
```

6

Este código imprimirá "Muy bien" si $calificacion está entre 80 y 89.

Operador ternario

El operador ternario es una forma compacta de una sentencia if-else. Se utiliza a menudo para asignar valores a variables basadas en una condición.

Sintaxis:

```
condición ? valorSiVerdadero : valorSiFalso;
```

7

[6] Accesibilidad

```
$calificacion = 85;
If ($calificacion >= 90) {
        echo “Excelente”;
} elseif (($calificacion >= 80) {
        echo “Muy bien”;
} else {
        echo “Necesitas mejorar”;

}
```

[7] Accesibilidad

```
condición ? valorSiVerdadero : valorSiFalso;
```

Ejemplo:

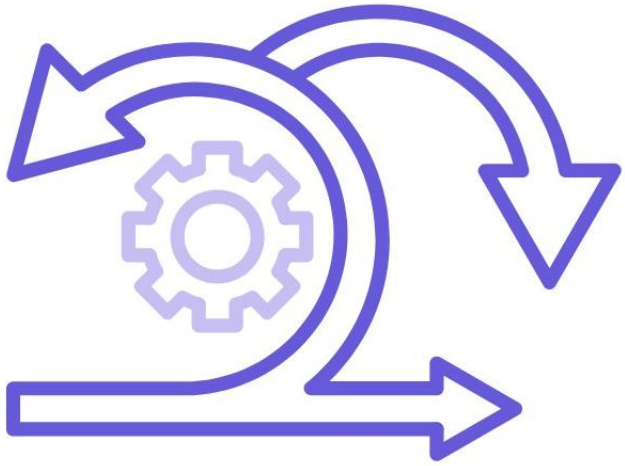
```php
$edad = 20;
$mensaje = $edad >= 18 ? "Eres mayor de edad." : "No eres mayor de edad.";
echo $mensaje;
```

8

Este código asignará "Eres mayor de edad." a $mensaje si $edad es 18 o más; de lo contrario, asignará "No eres mayor de edad."

[8] Accesibilidad
$edad = 20;
$mensaje = $edad >= 18 ? “Eres mayor de edad : “No eres mayor de edad”;
echo $mensaje;

Estructuras iterativas (Bucles)



Tras comprender cómo tomar decisiones en nuestros programas usando estructuras condicionales, ahora nos adentraremos en otro aspecto fundamental de la programación: las iteraciones o bucles.

Los bucles nos permiten ejecutar un bloque de código repetidamente, lo que es esencial para tareas como procesar cada elemento de un array, realizar operaciones matemáticas repetitivas, o simplemente repetir una acción hasta que se cumpla una condición específica.

Este módulo cubrirá los bucles más comunes en PHP: for, while, do-while, y foreach.

Bucle For

El bucle for es ideal para situaciones donde sabes de antemano cuántas veces necesitas repetir una acción.

Sintaxis:

```
for (inicialización; condición; incremento) {
    // Código a ejecutar en cada iteración
}
```

9

Ejemplo:

```
for ($i = 1; $i <= 10; $i++) {
    echo $i . " ";
}
```

10

Este código imprimirá los números del 1 al 10 en la pantalla, cada uno seguido de un espacio.

[9] Accesibilidad
for (inicialización; condición; incremento) {
// Código a ejecutar en cada interacción
}

[10] Accesibilidad
for ($i =1; $i <= 10; $i++) {
echo $i . " “;
}

Bucle While

Utiliza el bucle while cuando quieres repetir una acción mientras una condición sea verdadera, pero no sabes cuántas veces necesitarás iterar.

Sintaxis:

```
while (condición) {
    // Código a ejecutar mientras la condición sea verdadera
}
```

11

Ejemplo:

```
$i = 1;
while ($i <= 10) {
    echo $i . " ";
    $i++;
}
```

12

Al igual que el bucle for del ejemplo anterior, esto imprimirá los números del 1 al 10.

[11] Accesibilidad

```
while (condición) {
        // Código a ejecutar mientras la condición sea verdadera
}
```

[12] Accesibilidad

```
$i =1;
while ($i <= 10) {
        echo $i ." “ ;
        $i++
}
```

Bucle Do-While

El bucle do-while es similar al while, pero garantiza que el bloque de código se ejecute al menos una vez antes de verificar la condición.

Sintaxis:

```
do {
    // Código a ejecutar
} while (condición);
```

13

Ejemplo:

```
$i = 1;
do {
    echo $i . " ";
    $i++;
} while ($i <= 10);
```

14

Este bucle también imprimirá los números del 1 al 10, asegurando que el cuerpo del bucle se ejecute al menos una vez.

[13] Accesibilidad
do {
// Código a ejecutar
} while (condición);

[14] Accesibilidad
$i = 1;
do {
echo $i ." “ ;
$i++
} while ($i <= 10);

Bucles ForEach

El bucle foreach es específicamente diseñado para iterar sobre los elementos de un array, lo que lo hace extremadamente útil para trabajar con colecciones de datos.

Sintaxis:

```
foreach ($array as $valor) {
    // Código a ejecutar para cada elemento del array
}
```

15

Ejemplo:

```
$colores = ["rojo", "verde", "azul"];
foreach ($colores as $color) {
    echo $color . " ";
}
```

16

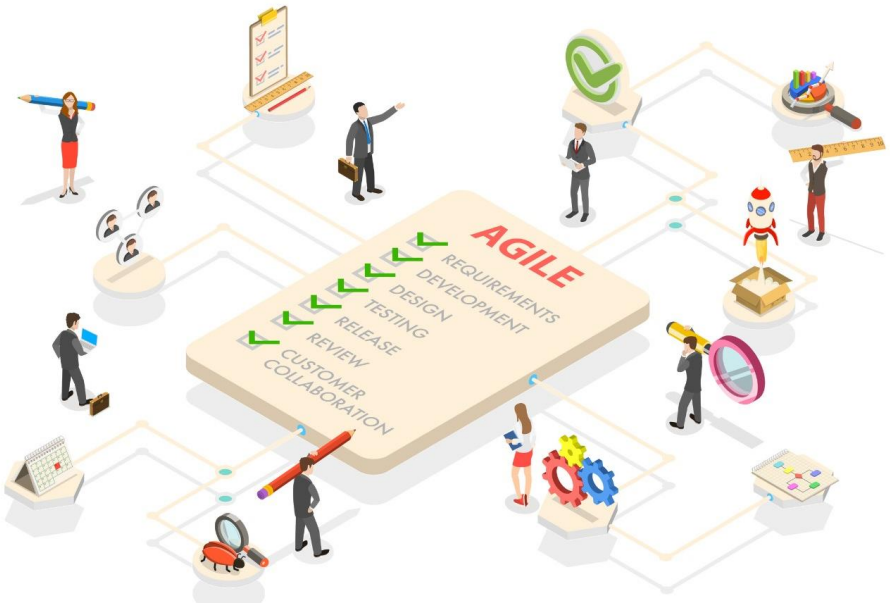
Este código imprimirá cada color en el array $colores, seguido de un espacio.

---

[15] Accesibilidad
foreach (Sarray as $valor) {
// Código a ejecutar para cada elemento del array
}

[16]
Accesibilidad
$colores = ["zojo", "verde", "azul"];
foreach ($colores as $color) {
echo $color . “ ";
}

Control de flujo en bucles



A medida que profundizamos en el uso de estructuras iterativas, es crucial entender cómo podemos manipular y controlar el flujo de ejecución dentro de nuestros bucles. Las sentencias break y continue son herramientas poderosas en PHP que nos permiten modificar el comportamiento predeterminado de los bucles, ya sea terminando prematuramente la iteración o saltando a la siguiente iteración bajo ciertas condiciones.

Este módulo se centra en cómo y cuándo utilizar estas sentencias para optimizar la lógica de nuestros bucles y hacer nuestro código más eficiente y legible.

Uso de break

La sentencia break se utiliza para terminar la ejecución del bucle en el que se encuentra. Es especialmente útil en situaciones donde, después de cumplirse una condición específica, no es necesario continuar ejecutando el resto del bucle.

Sintaxis:

```
while (condición) {
    if (condiciónParaTerminar) {
        break;
    }
    // Código a ejecutar
}
```

[17]

Ejemplo:

```
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        break;  // Termina el bucle si $i es igual a 5
    }
    echo $i . " ";
}
```

[18]

[17] Accesibilidad

```
while (condición) {
    if (condiciónParaTerminar) {
        break;
    }
    //Código a ejecutar
}
```

[18] Accesibilidad

```
for ($i =1; $i <= 10; $i++) {
    if $i == 5 {
        break; // Termina el bucle si $i es igual a 5
    }
    echo $i . " ";
}
```

Este código imprimirá los números del 1 al 4. Cuando $i sea igual a 5, la sentencia break terminará el bucle.

Uso de continue

La sentencia continue se utiliza para saltar el resto del código en la iteración actual del bucle y continuar con la siguiente iteración. Es útil cuando, bajo ciertas condiciones, quieres omitir partes del bucle sin terminar completamente su ejecución.

Sintaxis:

```
for ($i = 1; $i <= 10; $i++) {
    if (condiciónParaSaltar) {
        continue;
    }
    // Código a ejecutar si la condición no se cumple
}
```

19

[19] Accesibilidad

```
for ($i =1; $i <= 10; $i++) {
        if (condiciónParaSaltar) {
            continue;
        }
        // Código para ejecutar si la condición no se cumple
}
```

Ejemplo:

```php
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        continue;  // Salta el resto del código en esta iteración si $i es 5
    }
    echo $i . " ";
}
```

20

Este código imprimirá los números del 1 al 10, excepto el 5. Cuando $i sea igual a 5, la sentencia continue saltará al final del bucle y continuará con la siguiente iteración.

Entender cómo utilizar las sentencias break y continue te permitirá tener un mayor control sobre el flujo de ejecución de tus bucles. Estas herramientas son esenciales para manejar lógicas complejas dentro de las iteraciones y pueden hacer que tu código sea más eficiente al evitar ejecuciones innecesarias. Te animo a experimentar con estas sentencias y a integrarlas en tus prácticas de codificación para desarrollar soluciones más refinadas y efectivas.

[20] Accesibilidad

```
for ($i =1; $i <= 10; $i++) {
        if ($i == 5) {
                continue; // Se salta el resto del código si $i es igual o menor que  5
        }
        echo $i . " ";
}
```

Buenas prácticas

A medida que desarrollamos habilidades en la programación con PHP, es crucial adoptar buenas prácticas que no solo nos ayuden a evitar errores comunes, sino que también hagan nuestro código más legible, mantenible y eficiente.

Este módulo se enfoca en las buenas prácticas al trabajar con estructuras de control, con énfasis especial en la prevención de bucles infinitos y la mejora de la legibilidad del código.

Evitar bucles infinitos

- Condiciones de salida claras: Todo bucle debe tener una condición de salida clara. Es fundamental asegurarte de que la condición que controla el bucle pueda cambiar y eventualmente llevar a que el bucle termine.
- Cuidado con las actualizaciones de variables: Verifica que las variables que controlan la salida del bucle se actualicen correctamente dentro del bucle. Un error común es olvidar incrementar un contador o actualizar una condición, lo que puede llevar a un bucle infinito.
- Pruebas y validación: Antes de implementar un bucle en tu código, prueba su lógica con diferentes valores para asegurarte de que siempre tenga una condición de salida válida.

Legibilidad en las estructuras de control

- Indentación adecuada: Utiliza la indentación de manera consistente para estructurar tu código claramente. La indentación ayuda a visualizar la jerarquía del código y facilita la comprensión de las estructuras de control anidadas.
- Uso de comentarios: Añade comentarios para explicar la lógica detrás de decisiones complejas o no obvias en tus estructuras de control. Los comentarios pueden ser invaluablemente útiles para ti o para otros desarrolladores que puedan trabajar con tu código en el futuro.
- Nombres descriptivos para variables: Usa nombres descriptivos para las variables que controlan tus bucles y condiciones. Nombres como contador o esta Activo hacen que tu código sea mucho más comprensible que nombres como i o flag.

Adoptar buenas prácticas al trabajar con estructuras de control no solo te ayudará a evitar errores comunes como los bucles infinitos, sino que también hará que tu código sea más claro, legible y fácil de mantener. A medida que continúes practicando y desarrollando tus habilidades en PHP, estas prácticas se convertirán en parte integral de tu enfoque de programación, llevando a un código más eficiente y efectivo.

Conclusión

Las estructuras de control son el alma de la programación en PHP, otorgando a los desarrolladores el poder de crear código que no solo es dinámico y adaptable, sino también inteligente y eficiente. Hemos explorado cómo las sentencias condicionales y los bucles nos permiten tomar decisiones lógicas y repetir acciones de manera controlada, convirtiendo simples scripts en soluciones programáticas complejas y útiles. Estas herramientas fundamentales son las que diferencian un código estático de una aplicación web interactiva y reactiva, capaz de responder a las entradas del usuario y procesar datos de manera efectiva.

Las habilidades que has adquirido en este módulo forman la base sobre la cual construirás conocimientos más avanzados. A medida que avanzamos hacia temas más complejos, como la definición de funciones, la manipulación de formularios y datos, y la interacción con bases de datos, descubrirás cómo las estructuras de control se integran y amplifican en estos contextos. Las funciones te permitirán encapsular lógicas que puedes reutilizar y combinar con estructuras de control para crear código modular y eficiente. Al trabajar con bases de datos, las estructuras de control te ayudarán a procesar y reaccionar a los datos recuperados, facilitando la creación de aplicaciones web dinámicas y ricas en datos.

A medida que cierras este capítulo de tu viaje de aprendizaje en PHP, lleva contigo no solo el conocimiento de cómo usar las estructuras de control, sino también la comprensión de su poder para transformar líneas de código en soluciones programáticas significativas. Sigue practicando, sigue cuestionando, y más importante aún, sigue codificando. El mundo de PHP y el desarrollo web te esperan con infinitas posibilidades para explorar y crear. ***¡Adelante en tu camino hacia convertirte en un desarrollador PHP experto!***

Recursos adicionales

Aquí tienes una lista de recursos útiles que incluyen enlaces a documentación oficial y plataformas de práctica interactiva. Estos recursos están diseñados para complementar y reforzar tu aprendizaje, proporcionando oportunidades para profundizar en tu comprensión y aplicar lo que has aprendido en ejercicios prácticos.

Documentación oficial de PHP

- Estructuras de Control en PHP: La documentación oficial de PHP ofrece una visión exhaustiva de todas las estructuras de control disponibles en PHP, incluidas las sentencias condicionales y los bucles. Aquí encontrarás detalles sobre la sintaxis, ejemplos de uso y consideraciones especiales para cada estructura.
  https://www.php.net/manual/es/language.control-structures.php

Recursos educativos para principiantes

- W3Schools PHP Tutorial: W3Schools ofrece tutoriales introductorios que son fáciles de seguir y cubren una amplia gama de temas en PHP, perfectos para principiantes.
  https://www.w3schools.com/php/
- Codecademy's PHP Course: Codecademy ofrece un curso interactivo de PHP que te guía a través de los conceptos básicos con ejercicios prácticos, ideal para consolidar lo aprendido.
  https://www.codecademy.com/catalog/language/php4
- PHP: The Right Way: PHP: The Right Way es una guía de buenas prácticas popular que ofrece una visión general de PHP, incluyendo recomendaciones sobre estilos de codificación, prácticas recomendadas y enlaces a recursos de calidad.
  https://phptherightway.com/

Comunidades y foros

- Reddit r/PHP: El subreddit r/PHP es un buen lugar para mantenerse al día con las noticias de PHP, hacer preguntas y participar en discusiones con otros desarrolladores PHP.
  https://www.reddit.com/r/PHP/?rdt=45690

- PHP Freaks: PHP Freaks es un foro donde puedes encontrar respuestas a tus preguntas, participar en discusiones y conectarte con otros desarrolladores PHP.
  http://www.phpfreaks.com/

Estos recursos adicionales están aquí para ayudarte a profundizar en tu comprensión de las estructuras de control en PHP y para proporcionarte las herramientas necesarias para practicar y perfeccionar tus habilidades. Aprovecha la documentación oficial para obtener una base sólida, sumérgete en ejercicios prácticos para reforzar tu aprendizaje y participa en comunidades para compartir tus experiencias y aprender de otros. ***Con dedicación y práctica, te convertirás en un experto en manejar el flujo de tus programas PHP.***