PHP

UT4: ESTRUCTURAS DE CONTROL

1. Introducción
2. Estructuras de bifurcación condicional
    - Condicional simple: if
    - Condicional compuesta: if
    - Condicional switch
    - Anidamiento de estructuras condicionales
    - Operador ternario ?:
3. Estructuras de bucle
    - Bucle while
    - Bucle do-while
    - Bucle for
4. Funciones
    - Funciones definidas en PHP
    - Funciones definidas por el usuario
5. include y require

www.php.net/manual/es/

1. Introducción

Las instrucciones de un script PHP se ejecutan secuencialmente una detrás de otra, desde la primera hasta la última, a menos que utilicemos algún tipo de estructura de control.

Las estructuras de control nos permiten variar el flujo de ejecución secuencial y pueden ser de los siguientes tipos:

- Bifurcaciones: Ejecutan un bloque de instrucciones concreto entre varios disponibles en función de con qué valor se evalúe una expresión.
- Bucles: Ejecutan un mismo bloque de instrucciones repetidamente hasta que una expresión se evalúe con cierto valor.
- Funciones: Son un bloque de instrucciones reutilizable que puede ejecutarse en distintas partes del script simplemente mencionando su nombre.
- include y require: Sirven para incluir en un script el código disponible en un archivo, como si hubiera sido escrito directamente en el script original.

2. Estructuras de bifurcación condicional

En PHP disponemos de dos estructuras de bifurcación: `if` y `switch`.

El lenguaje PHP ofrece la sitaxis de llaves y la sintaxis de dos puntos. Permite cambiar la llave de apertura { por dos puntos (:) y la de cierre } por endif; o endswitch;.

Ambas sintaxis producen exactamente los mismos resultados y pueden aplicarse en los mismos contextos; pero por facilidad de lectura suele utilizarse la sintaxis de dos puntos para presentar código HTML.

Pero también existe un operador que podría interpretarse como una bifurcación: el operador ternario o ?:

- # Condicional simple: if

Sintaxis de llaves:

```
if(condición)
   {bloque_de_instrucciones_a_ejecutar_si_condicion_es_TRUE;}
```

Sintaxis de dos puntos:

```
if(condición):
   bloque_de_instrucciones_a_ejecutar_si_condicion_es_TRUE;
endif;
```

- # Condicional compuesta: if-else

Sintaxis de llaves:

```
if(condición)
        {bloque_de_instrucciones_a_ejecutar_si_condicion_es_TRUE;}
else
        {bloque_de_instrucciones_a_ejecutar_si_condicion_es_FALSE;}
```

Sintaxis de dos puntos:

```
if(condición):
        bloque_de_instrucciones_a_ejecutar_si_condicion_es_TRUE;
else:
        bloque_de_instrucciones_a_ejecutar_si_condicion_es_FALSE;
endif;
```

- Condicional switch

Sintaxis de llaves:

```
switch(expresión){
 case valor_caso_1:
        instrucciones_a_ejecutar_si_expresion_coincide_con_el_valor_de_este_caso;
        break;
 case valor_caso_2:
        instrucciones_a_ejecutar_si_expresion_coincide_con_el_valor_de_este_caso;
        break;
 case valor_caso_3:
        instrucciones_a_ejecutar_si_expresion_coincide_con_el_valor_de_este_caso;
        break;
        ......
 default:
    instrucciones_a_ejecutar_si_expresion_no_coincide_con_el_valor_de_otro_caso;
}
```

Sintaxis de dos puntos:

```
switch(expresión):
 case valor_caso_1:
        instrucciones_a_ejecutar_si_expresion_coincide_con_el_valor_de_este_caso;
        break;
 case valor_caso_2:
        instrucciones_a_ejecutar_si_expresion_coincide_con_el_valor_de_este_caso;
        break;
 case valor_caso_3:
        instrucciones_a_ejecutar_si_expresion_coincide_con_el_valor_de_este_caso;
        break;
        ......
 default:
    instrucciones_a_ejecutar_si_expresion_no_coincide_con_el_valor_de_otro_caso;
endswitch;
```

- Anidamiento de estructuras condicionales

Sintaxis de llaves:

```
if(condición_1)
        { bloque_de_instrucciones_a_ejecutar_si_condicion_1_es_TRUE;}
elseif(condición_2)
        { bloque_de_instrucciones_a_ejecutar_si_condicion_2_es_TRUE; }
elseif(condición_3)
        { bloque_de_instrucciones_a_ejecutar_si_condicion_3_es_TRUE;}
.....
else
        { bloque_de_instrucciones_a_ejecutar_si_ninguna_condición_anterior_es_TRUE;}
```

Sintaxis de dos puntos:

```
if(condición_1):
        bloque_de_instrucciones_a_ejecutar_si_condicion_1_es_TRUE;
elseif(condición_2):
        bloque_de_instrucciones_a_ejecutar_si_condicion_2_es_TRUE;
elseif(condición_3):
        bloque_de_instrucciones_a_ejecutar_si_condicion_3_es_TRUE;
.....
else:
        bloque_de_instrucciones_a_ejecutar_si_ninguna_condición_anterior_es_TRUE;
endif;
```

- Operador ternario ?:

```
condición ? resultado_si_condicion_TRUE : resultado_si_condicion_FALSE;
```

3. Estructuras de bucles

En esta sección vamos a estudiar los bucles `while`, `do…while` y `for`. Todos ellos ejecutan un mismo bloque de instrucciones repetidas veces (hasta que se evalúa como FALSE una condición. Cada una de estas ejecuciones se denomina iteración.

Existen dos instrucciones que permiten alterar este funcionamiento predeterminado y que son válidas para todos los tipos de bucles:

- `break`: Abandona el bucle.
- `continue`: Abandona la iteración actual y continua la siguiente.

• Bucle while

Sintaxis de llaves:

```
while (condicion){
        instrucciones_a_ejecutar_en_cada_iteración
}
```

Sintaxis de dos puntos:

```
while (condicion):
        instrucciones_a_ejecutar_en_cada_iteración
endwhile;
```

- Bucle do-while

Sintaxis de llaves:

```
do{
        instrucciones_a_ejecutar_en_cada_iteración
}while (condicion);
```

- Bucle for

Sintaxis de llaves:

```
for(expresion_inicial;expresion_de_continuidad;expresión_tras_cada_iteración){
        instrucciones_a_ejecutar_en_cada_iteración;
}
```

Sintaxis de dos puntos:

```
for(expresion_inicial;expresion_de_continuidad;expresión_tras_cada_iteración):
        instrucciones_a_ejecutar_en_cada_iteración
endfor;
```

Y su ejecución se realiza en los siguientes pasos:

1. Se evalúa la expresión inicial.
2. Si la *expresión_de_continuidad* se evalúa como TRUE se inicia la ejecución de las instrucciones de la iteración, si no se abandona el bucle.
3. Al terminar la iteración se ejecuta la *expresión_tras_cada_iteración* y, a continuación, se vuelve al paso 2.

4. Funciones

- Funciones definidas en PHP

Las funciones definidas por PHP las puedes encontrar en el siguiente enlace:

http://es2.php.net/manual/es/funcref.php

Ejemplo:

number pow (number *$base* , number *$exp*)

- Funciones definidas por el usuario

5. include y require