PROGRAMACIÓN WEB EN ENTORNO SERVIDOR UD-4

ARRAYS

ÍNDICE

[INTRODUCCIÓN 3](#_Toc146125486)

[STRING Y ARRAYS 3](#_Toc146125487)

[ARRAYS CON CLAVE 4](#_Toc146125488)

[ARRAYS MULTIDIMENSIONALES 4](#_Toc146125489)

[RECORRIDO CON FOREACH 4](#_Toc146125490)

[OPERACIONES DE COMPARACIÓN 4](#_Toc146125491)

# INTRODUCCIÓN

Los arrays en PHP se pueden usar de dos maneras, como un arrayList de Java, pero tenemos índices, que comienzan en el valor 0, de cada elemento, o de una forma similar a los mapas de Java. Tendremos una clave y un valor.

Los arrays van a poder contener elementos de distintos tipos.

Si quiero inicializar el array con los valores, pongo entre paréntesis, separados por comas los valores del array $strArrNotas = [9, 6, 5…];

Para mostrar un valor del array, lo hago igual que en java accediendo a su índice echo $strArrNotas [5]

Si quiero ver todo el contenido puedo usar la función nativa de PHP var\_dump($strArrNotas); No es necesario usar el echo.

Con la función print\_r($variable), también obtengo información de una variable. No indica el nombre de la variable ni inserta saltos de línea.

Se puede crear un array vacío con corchetes en blanco $arrayVacio = [ ]; o usando la función array() $arrayVacio = array( );

Para añadir elementos, simplemente indicamos la variable de array con los corchetes vacíos y le asignamos el nuevo valor $intArr[] = 6;

Con la función count($array), se cuántos elementos tiene mi array.

Podemos concatenar arrays con la función array\_merge($array1, $array2), que devuelve un array con los elementos del array1 seguidos de los del array2 (similar al método .addAll(coleccion) de java)

Con la función unset($array[posición]), se elimina dicha posición de dicho array y desaparece dicho índice del array. Aunque metamos nuevos datos, dicho índice no se vuelve a restituir. Esto es problemático, dado que si usamos la función array\_merge, quitaría los huecos, pero cambiaría los índices de los elementos según el orden de inserción primero (las modificaciones no cambian el orden)

# STRING Y ARRAYS

Se puede acceder a los elementos de un string y modificarlos o mostrarlos con la nomencladura de los arrays (strCadena[posición]=valor)

Para dejar un string en un array, carácter a carácter, se puede usar la función str\_split(cadena) y si se quieren usar comodines para eliminar ciertos caracteres, por ejemplo, los espacios en blanco, se puede usar spred\_Split(‘comodines’, cadena);

# ARRAYS CON CLAVE

Como ya indicamos al principio del tema, se pueden utilizar arrays similares a los mapas de java. En lugar de índices numéricos, vamos a utilizar cadenas (claves), que deberán ser únicas. El acceso, modificación y borrado funciona de la misma manera que en un array normal. Para inicializar el array lo aremos con $array=[‘clave1’ => valor1,..., ‘clave1N => valorN];

# ARRAYS MULTIDIMENSIONALES

Para definir un array con más dimensiones, lo único que tenemos que hacer, es definir tantos corchetes como nos interese, indicando cada apertura y cierre de corchetes una dimensión $array = [[]]. Para operar con cada elemento indicaremos sus índices en corchetes diferentes $array[1][3].

# RECORRIDO CON FOREACH

Para recorrer un array el bucle más sencillo es foreach. Al igual que en java se indica el array y en que variable lo queremos almacenas por medio de la palabra reservada as foreach($array as $variable).

Si tenemos un array multidimensional, la variable que nos devolverá, será también un array foreach($array1 as $array2).

En caso de que el array sea con claves, en lugar de con índices, tendremos que indicar la clave del array que deseamos recorrer foreach($array[clave] as $array2).

Para cada una de las dimensiones que queramos recorrer deberíamos meter un foreach.

# OPERACIONES DE COMPARACIÓN

* $array1 + $array2: devuelve un array con los elementos de 1 más los de 2, pero eliminando colisiones en función del índice. Si el índice es numérico, solo se copiarán los elementos mayores del segundo array. Si los índices están desordenados, pueden seguir quedando desordenados.
* $array1 == $array2: Indica si los arrays contienen los mismos elementos.
* $array1 === $array2: Indica si los arrays contienen los mismos elementos en el mismo orden.
* $array1 != $array2: Indica si los arrays no contienen los mismos elementos.
* $array1 !== $array2: Indica si los arrays no contienen los mismos elementos, o no están en el mismo orden.
* rand([$intValor1, $intValor2]): saca números aleatorios enteros entre los valores incluidos dichos valores.
* range(intInicial, intFinal[, rlnSalto]): rellena un array con valores entre los indicados con distancia entre un valor y otro según lo indicado en rlnSalto. Los valores inicial y final se pueden cambiar por caracteres y el salto tendrá que ser un número real.
* in\_array(elemento); indica si el elemento está en el array.
* sort($array[, tipoOrden]): ordena el un array. Si se le indica un tipo de orden sigue el específico indicado SORT\_NUMERIC, SORT\_STRING, SORT\_LOCALE\_STRING (idioma local), SORT\_NATURAL (entiende la cadena y si tiene números, esa parte como número) y SORT\_FLAG\_CASE (no casesensitive).
* asort y ksort: igual que sort pero en arrays de clave valor, de tal manera que ordenaremos por el valor con asort, o por la clave con ksort.
* rsort, rasort y rksort: igual que los anteriores, pero para orden inverso
* shuffle($array): ordena un array de manera aleatoria.
* explode($strSeparador, $strCadenaADividir[,nº\_elementos]): saca un array de una cadena en base a un separador.
* list($var1,..., $varN)=$array:asigna los valores de un array a un conjunto de variables.