

DESARROLLO DE APLICACIONES WEB 2

ESTUDIO DE CLASES Y FUNCIONES PHP



15 de noviembre de 2023

DESARROLLO WEB EN ENTORNO SERVIDOR

CARLOS GARCIA CACHON

[CLASES 2](#_Toc151109356)

[DOMDocument 2](#_Toc151109357)

[DATETIME 2](#_Toc151109358)

[PDO 2](#_Toc151109359)

[PDOSTATEMENT 2](#_Toc151109360)

[PDOEXCEPTION 2](#_Toc151109361)

[FUNCIONES 3](#_Toc151109362)

[date() 3](#_Toc151109363)

[json\_encode(): 4](#_Toc151109364)

[json\_decode() 5](#_Toc151109365)

[file\_put\_contents() 5](#_Toc151109366)

[header() 5](#_Toc151109367)

[basename() 5](#_Toc151109368)

CLASES

DOMDocument

Para XML y HTML

<https://www.php.net/manual/es/class.domdocument>

formatOutput()

createElement()

appendChild()

save() “PARA EXPORTAR”

load() “PARA IMPORTAR”

getElementByTagName(‘NOMBRE\_DE\_LA\_COLUMNA\_DE\_LA\_TABLA’) -> item(0) -> nodeValue

DATETIME

PDO

PDOSTATEMENT

PDOEXCEPTION

FUNCIONES

date()

json\_encode()

Funcionalidad y Utilidad:

* Convierte una variable de PHP en su representación JSON.
* Enviar datos desde el servidor al cliente, por ejemplo, en una aplicación web
* Almacenar datos estructurados en archivos o bases de datos en formato JSON.

Parámetros Principales:

Tiene dos parámetros principales

* Datos para codificar ($data): Es el valor que se va a convertir a JSON. Puede ser un array, un objeto, una cadena de texto, números, booleanos y también admite valor NULL.
* Opciones ($options): Es un parámetro opcional que te permite especificar opciones para el proceso de codificación. Puedes usarlo para controlar el formato del JSON resultante. Algunas opciones comunes incluyen JSON\_PRETTY\_PRINT para formatear el JSON de manera legible y JSON\_UNESCAPED\_UNICODE para evitar el escapado de caracteres Unicode.

Ejemplo #1 Un ejemplo de json\_encode()

Texto

Descripción generada automáticamente

Resultado:



Ejemplo #2 Un ejemplo de json\_encode() + JSON\_PRETTY\_PRINT

Texto

Descripción generada automáticamente

Resultado:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ejemplo #3 Un ejemplo de json\_encode() + JSON\_UNESCAPED\_UNICODE

Texto

Descripción generada automáticamente

Resultado Con JSON\_UNESCAPED\_UNICODE:



Resultado Sin JSON\_UNESCAPED\_UNICODE:



Características Principales:

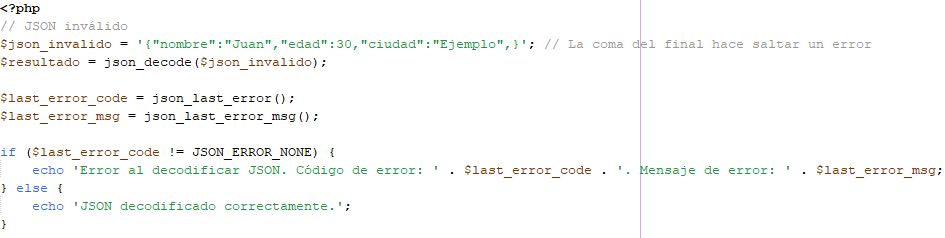
* UTF-8 por defecto: JSON usa UTF-8 como su formato de caracteres predeterminado, y json\_encode() asume que los datos de entrada están en UTF-8.

Manejo de errores:

La función retorna false en caso de error durante la codificación. Puedes usar:

json\_last\_error() : Devuelve un código de error

json\_last\_error\_msg() : Devuelve el mensaje de error correspondiente



En este ejemplo, después de intentar decodificar el JSON inválido, se almacena el código de error en $last\_error\_code y el mensaje de error en $last\_error\_msg. Luego, se verifica si hay algún error ($last\_error\_code != JSON\_ERROR\_NONE) y, en caso afirmativo, se imprime tanto el código como el mensaje de error.

Resultado:



Links:

[https://www.php.net/manual/es/function.json-encode](https://www.php.net/manual/es/function.json-encode.php)

json\_decode()

Funcionalidad y Utilidad:

* Convierte una cadena JSON en una variable de PHP. Esto es especialmente útil cuando recibes datos en formato JSON, como en solicitudes HTTP o al leer archivos JSON.

Parámetros Principales:

Tiene dos parámetros principales:

* Cadena JSON ($json): Es la cadena que contiene la representación JSON que deseas decodificar.
* Asociatividad ($assoc, opcional): Este parámetro es opcional y determina si deseas que el resultado sea un array asociativo (true) o un objeto estándar (false). Por defecto, es false, lo que significa que el resultado será un objeto.

Ejemplo #1 Un ejemplo de json\_decode()

Texto

Descripción generada automáticamente con confianza baja

Resultado:



Ejemplo #2 Un ejemplo de json\_decode() + true

Imagen que contiene Texto

Descripción generada automáticamente

Resultado:



Características Principales:

* Manejo de Objetos y Arrays: Puede generar tanto objetos como arrays dependiendo del valor de $assoc.
* Maneja tipos de datos comunes en JSON: Array, un objeto, una cadena de texto, números, booleanos y también admite valor NULL.

Manejo de errores:

Si hay algún error durante el proceso de decodificación devuelve NULL.

Puedes utilizar:

json\_last\_error() : Devuelve un código de error

json\_last\_error\_msg() : Devuelve el mensaje de error correspondiente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

En este ejemplo con una cadena JSON inválida, al intentar decodificar la con json\_decode(), el resultado será NULL, indicando un fallo en la decodificación. Luego, se verifica si hay un error al comprobar si el código de error es diferente de JSON\_ERROR\_NONE, lo que indica la presencia de un error. En caso afirmativo, se imprime un mensaje detallado que incluye tanto el código como el mensaje de error,

Resultado:



Links:

<https://www.php.net/manual/es/function.json-decode>

file\_put\_contents()

Funcionalidad y Utilidad:

* Sirve para escribir datos en un archivo. Su utilidad principal es simplificar el proceso de escritura de datos en un archivo, ya que realiza varias operaciones de archivo en una sola llamada.

Parámetros Principales:

Tiene dos parámetros principales:

* Nombre del Archivo ($filename): Es el nombre del archivo en el que deseas escribir los datos. Si el archivo no existe, se intentará crear. Si el archivo ya existe, los datos existentes se sobrescribirán, a menos que se especifique lo contrario.
* Datos ($data): Los datos que deseas escribir en el archivo. Esto puede ser una cadena de texto, un array o cualquier otro tipo de datos que pueda convertirse en una cadena.

Ejemplo #1 Un ejemplo de file\_put\_contents()

Texto

Descripción generada automáticamente

Resultado:

En este ejemplo, la cadena "Hola, mundo!" se escribirá en el archivo llamado 'archivo.txt'. Si el archivo ya existe, su contenido se sobrescribirá.

* Flags ($flags, opcional): Este parámetro es opcional y se utiliza para especificar comportamientos adicionales.

Algunos valores comunes son:

* + FILE\_APPEND : Para agregar datos al final del archivo en lugar de sobrescribirlo.
  + LOCK\_EX : Para adquirir un bloqueo exclusivo mientras se escribe en el archivo.

Ejemplo #2 Un ejemplo de file\_put\_contents() + FILE\_APPEND

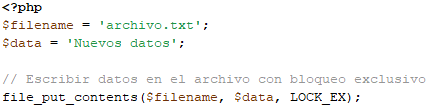
Texto

Descripción generada automáticamente

Resultado:

En este ejemplo, si el archivo 'archivo.txt' ya contiene datos, la nueva cadena ("Datos adicionales") se agregará al final del archivo sin borrar el contenido existente. Si el archivo no existe, se creará y se escribirá la cadena en él.

Ejemplo #3 Un ejemplo de file\_put\_contents() + LOCK\_EX



Resultado:

En este ejemplo, se utiliza file\_put\_contents() con LOCK\_EX para escribir los "Nuevos datos" en el archivo 'archivo.txt'. Durante este proceso, se adquiere un bloqueo exclusivo en el archivo, asegurando que ningún otro proceso pueda escribir en él simultáneamente.

Características Principales:

* Sobreescritura o Adición: Por defecto sobrescribe el contenido del archivo. Sin embargo, si usas la bandera FILE\_APPEND, puedes agregar datos al final del archivo.
* Facilidad de Uso: file\_put\_contents() simplifica la tarea de escribir datos en un archivo en comparación con usar fopen(), fwrite() y fclose() de manera separada.
* Devuelve el número de bytes escritos en el archivo si la operación se realiza con éxito.

Manejo de Errores:

En caso de error retorna false si ocurre un error.

Puedes usar:

error\_get\_last() : devuelve un array asociativo que contiene información sobre el último error que ocurrió en el script. Este array tiene las siguientes claves:

type: El tipo de error.

message: El mensaje de error.

file: La ruta al archivo donde ocurrió el error.

line: El número de línea donde ocurrió el error.

Texto

Descripción generada automáticamente

Resultado:



Links:

<https://www.php.net/manual/es/function.file-put-contents>

header()

basename()

Recibe una URL