DESARROLLO DE APLICACIONES WEB 2

ESTUDIO DE CLASES Y FUNCIONES PHP

15 DE DICIEMBRE DE 2023

DESARROLLO WEB EN ENTORNO SERVIDOR
OSCAR PASCUAL FERRERO
AUTOR ORIGINAL
CARLOS GARCIA CACHON

CLASES

DOMDocument

Es una clase que permite trabajar con documentos XML permitiendo crear, cargar y manipular este tipo de archivos forma parte de DOM (Modelo de Objeto de Documento)

https://www.php.net/manual/en/class.domdocument.php

Propiedades

version

Indica la versión de XML que se utilizará en el documento.

```php
$dom = new DOMDocument('1.0');
echo $dom->version; // Imprime 1.0
```

encoding

Indica la codificación de caracteres utilizada en el documento.

```php
$dom = new DOMDocument('1.0', 'utf-8');
echo $dom->encoding; // Imprime utf-8
```

formatOutput

Formatea la salida

```php
//formatea la salida
$dom->formatOutput = true;
```

xmlStandalone

Controla si el documento XML debe considerarse independiente o no. Un documento XML independiente es aquel que no hace referencia a un documento DTD. Mediante true, false.

```php
//Indica con true que el documento no esta asoicado a un dtd
$dom->xmlStandalone = true;
```

resolveExternals

Controla si el analizador XML debe intentar resolver y cargar entidades externas, como archivos DTD o entidades externas definidas en el propio documento. Mediante true, false.

```
//Indica si hay que resolver entidades externas o no como puede ser un dtd
$dom->resolveExternals = true;
```

documentElement

Representa el elemento raíz

```
//Representa el elemento raiz del documento
$dom->documentElement;
```

validateOnParse

Valida un documento que tiene un dtd asociado

```
$dom = new DOMDocument;
$dom->validateOnParse = true;
$dom->load('documento.xml');
```

Métodos

createElement()

Crea un nuevo elemento con el nombre especificado.

```
$elemento = $dom->createElement('nombre_etiqueta', 'contenido_de_texto');
```

appendChild()

Agrega un hijo a un elemento

```
$dom->appendChild($elemento);
```

createAttribute()

Crea un nuevo nodo de atributo con el nombre especificado.

```
$atributo = $dom->createAttribute('nombre_atributo');
```

saveXML()

Devuelve una cadena que contiene la representación del documento en XML

```
echo $dom->saveXML();
```

load()

Carga un documento XML desde una cadena o un archivo

```
$dom->load('nombre_archivo.xml');
```

getElementsByTagName()

Devuelve una lista de elementos con un nombre de etiqueta específico

```
$elementos = $dom->getElementsByTagName('nombre_etiqueta');
```

getAttribute()

Obtiene el valor de un atributo específico de un elemento

```
$valor = $elemento->getAttribute('nombre_atributo');
```

setAttribute()

Establece el valor de un atributo para un elemento.

```
$elemento->setAttribute('nombre_atributo', 'nuevo_valor');
```

Ejemplo de uso

En PHP

```php
//Crea un fichero XML
$dom = new DOMDocument('1.0', 'utf-8');

// Crear elemento raíz
$raiz = $dom->createElement('raiz');
$dom->appendChild($raiz);

// Crear elemento hijo
$hijo = $dom->createElement('hijo', '¡Hola, XML!');
$raiz->appendChild($hijo);

// Generar salida XML
echo $dom->saveXML();
```

Esto genera

```xml
<?xml version="1.0" encoding="utf-8"?>
<raiz>
    <hijo>¡Hola, XML!</hijo>
</raiz>
```

Manejo de errores

Sirve para guardar los errores relacionados con el documento xml provocados durante la ejecución

```php
$dom = new DOMDocument;
libxml_use_internal_errors(true);  // Habilitar el manejador de errores interno de libxml
$dom->load('documento_erroneo.xml');
$errores = libxml_get_errors(); // Guarda los errores ocurridos durante la ejecucion en $errores
libxml_clear_errors(); //limpia el listado de errores
```

Autor: Ismael Ferreras García

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()

```php
<?php
$data = array(
    "nombre" => "Juan",
    "edad" => 30,
    "ciudad" => "Ejemplo"
);

$json_data = json_encode($data);
```

Resultado:

```
{"nombre":"Juan","edad":30,"ciudad":"Ejemplo"}
```

Ejemplo #2 Un ejemplo de json_encode() + JSON_PRETTY_PRINT

```php
<?php
$data = array(
    "nombre" => "Juan",
    "edad" => 30,
    "ciudad" => "Ejemplo");
$json_data = json_encode($data, JSON_PRETTY_PRINT);
```

Resultado:

```
{
    "nombre": "Juan",
    "edad": 30,
    "ciudad": "Ejemplo"
}
```

Ejemplo #3 Un ejemplo de json_encode() + JSON_UNESCAPED_UNICODE

```php
<?php
$data = array("nombre" => "Juan", "ciudad" => "東京");
$json_data = json_encode($data, JSON_UNESCAPED_UNICODE);
```

Resultado Con JSON_UNESCAPED_UNICODE:

```
{"nombre":"Juan","ciudad":"東京"};
```

Resultado Sin JSON_UNESCAPED_UNICODE:

```
{"nombre":"Juan","ciudad":"\u6771\u4EAC"}
```

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

```php
<?php
// JSON inválido
$json_invalido = '{"nombre":"Juan","edad":30,"ciudad":"Ejemplo",}'; // La coma del final hace saltar un error
$resultado = json_decode($json_invalido);

$last_error_code = json_last_error();
$last_error_msg = json_last_error_msg();

if ($last_error_code != JSON_ERROR_NONE) {
    echo 'Error al decodificar JSON. Código de error: ' . $last_error_code . '. Mensaje de error: ' . $last_error_msg;
} else {
    echo 'JSON decodificado correctamente.';
}
```

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:

Error al decodificar JSON. Código de error: 4. Mensaje de error: Syntax error

Links:

https://www.php.net/manual/es/function.json-encode

Autor: Carlos García Cachón

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()

```php
<?php
$json_data = '{"nombre": "Juan", "edad": 30, "ciudad": "Ejemplo"}';

// Decodificar la cadena JSON
$resultado = json_decode($json_data);

// Imprimir el resultado
var_dump($resultado);
```

Resultado:

```
object(stdClass)#1 (3) { ["nombre"]=> string(4) "Juan" ["edad"]=> int(30) ["ciudad"]=> string(7) "Ejemplo" }
```

Ejemplo #2 Un ejemplo de json_decode() + true

```php
<?php
$json_data = '{"nombre": "Juan", "edad": 30, "ciudad": "Ejemplo"}';

// Decodificar la cadena JSON
$resultado = json_decode($json_data,true);

// Imprimir el resultado
var_dump($resultado);
```

Resultado:

```
array(3) { ["nombre"]=> string(4) "Juan" ["edad"]=> int(30) ["ciudad"]=> string(7) "Ejemplo" }
```

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

```php
<?php
$json_invalido = '{"nombre": "Juan", "edad": 30, "ciudad": "Ejemplo",}'; // La coma del final hace saltar un error
$resultado_invalido = json_decode($json_invalido);

if ($resultado_invalido === null && json_last_error() !== JSON_ERROR_NONE) {
    echo 'Error al decodificar JSON. Código de error: ' . json_last_error() . '. Mensaje de error: ' . json_last_error_msg();
} else {
    var_dump($resultado_invalido);
}
```

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:

```
Error al decodificar JSON. Código de error: 4. Mensaje de error: Syntax error
```

Links:

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

Autor: Carlos García Cachón

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()

```php
<?php
$filename = 'archivo.txt';
$data = 'Hola, mundo!';

// Escribir datos en el archivo
file_put_contents($filename, $data);
?>
```

  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

```php
<?php
$filename = 'archivo.txt';
$data = 'Datos adicionales';

// Agregar datos al final del archivo
file_put_contents($filename, $data, FILE_APPEND);
```

  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

```php
<?php
$filename = 'archivo.txt';
$data = 'Nuevos datos';

// Escribir datos en el archivo con bloqueo exclusivo
file_put_contents($filename, $data, 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.

```php
<?php
$filename = 'archivo_no_permisos.txt';
$data = 'Datos para escribir';

// Intentar escribir en un archivo sin permisos
$resultado = file_put_contents($filename, $data);

// Verificar si hay un error
if ($resultado === false) {
    $error = error_get_last();
    echo 'Error: ' . $error['message'] . ' en ' . $error['file'] . ' en la línea ' . $error['line'];
} else {
    echo 'Operación de escritura exitosa. Bytes escritos: ' . $resultado;
}
```

Resultado:

Error: file_put_contents(archivo_no_permisos.txt): failed to open stream: Permission denied en /home/EIRGo6/prog.php en la línea 10

Links:

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

Autor: Carlos García Cachón

header()

basename()

Recibe una URL