

WebP Generator Module



versión 1.0.4
por PrestaChamps

Requisitos	2
general del módulo	3
Configuración del módulo	4
EWWW Cloud Convert	6
Configuración de conversión CWEBP	7
Regenerar WebP	8
Compatibilidad con el navegador WEBP	10
Prestashop 1.6 compatibilidad	11

Requisitos

Asegúrese de que su versión de PHP sea al menos 5.6 del módulo no funcionará (como se menciona en la hoja de requisitos del producto en Complementos PrestaShop)

En el servidor debe configurarse uno de los siguientes módulos PHP:

Cwebp => php exec function enabled

Imagick => Imagick php module con webp compilado

Gmagick => Gmagick php module con webp compilado

Gd => Gd php module con imagewebp, imagecreatefrompng e imagecreatefromjpeg funciones

Ewww => ewww.io Servicios de suscripción y php curl Descripción del



Módulo

Este módulo es un elemento esencial si desea un sitio web rápido.

El módulo WebP Generator, creado para los sitios de Prestashop, permitirá a los usuarios regenerar eficientemente las imágenes de sus productos, categorías y fabricantes, lo que aumentará la velocidad del sitio.

Esta nueva tecnología proporciona una compresión superior sin pérdida y con pérdida para las imágenes de su sitio web. Una vez que comience a generar imágenes a través de nuestro módulo, liberará espacio en su servidor. Además, la calidad de tus imágenes no se verá afectada.

Las imágenes generadas a través del **módulo generador de WebP** son un 26% más pequeñas en tamaño en comparación con las PNG. Las imágenes con pérdida de WebP son un 25-34% más pequeñas que las imágenes JPEG comparables con un índice de calidad SSIM equivalente.

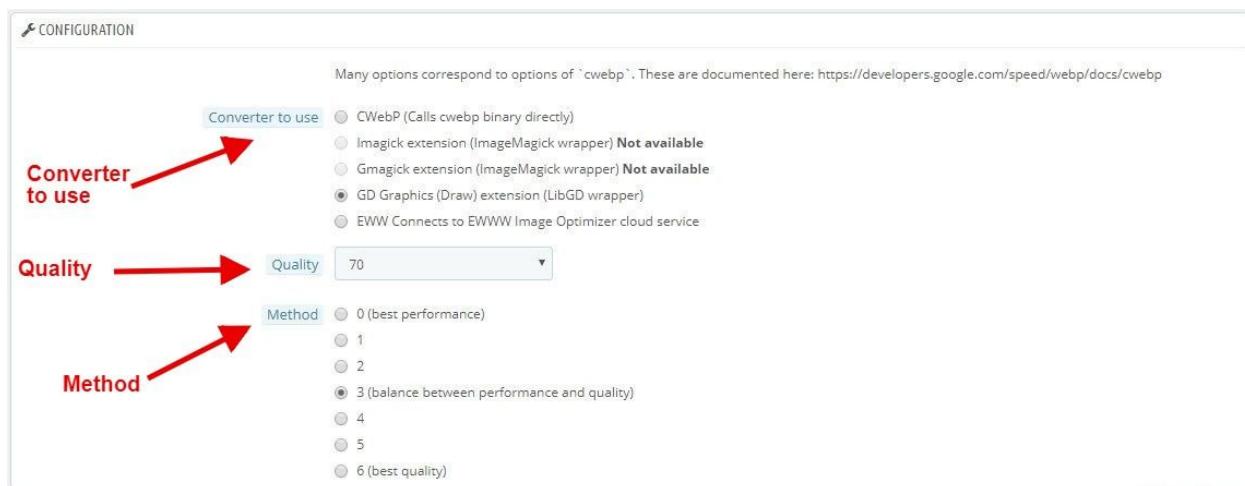
Información: el módulo se puede instalar desde la oficina central del sitio web, al igual que todos los demás módulos.

Configuración del módulo

Después de la instalación se puede acceder al módulo haciendo clic en el **Configurar** botón.

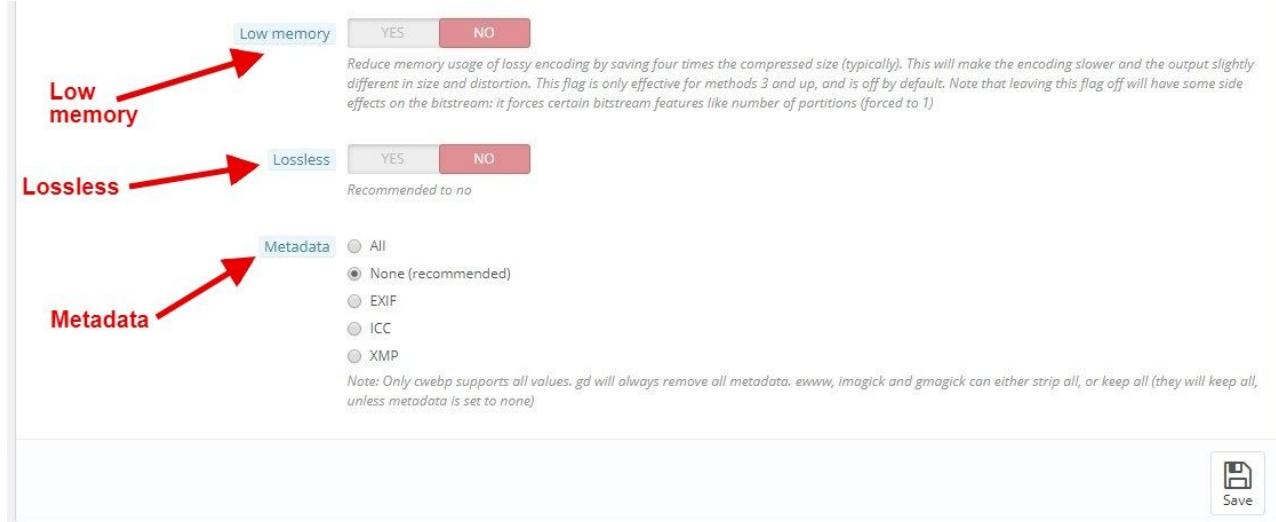


- **Convertidor para usar:** configure los métodos de conversión para usar según la configuración de su servidor.
- **Calidad:** especifique el factor de compresión para los canales RGB entre 0 y 100
- **Método:** este parámetro controla la compensación entre la velocidad de codificación y el tamaño y la calidad del archivo comprimido. Los valores posibles van de 0 a 6. 0 es el más rápido. 6 resultados en la mejor calidad.



- **Memoria baja:** reduzca el uso de memoria de la codificación con pérdida a un costo de aproximadamente un 30% más del tiempo de codificación y un tamaño de salida ligeramente mayor.

- **Sin pérdida:** codifique la imagen sin ninguna pérdida. La opción se ignora para PNG (forzado verdadero)
- **Metadatos-** Metadatos para copiar desde la entrada a la salida, si está presente.



- **Modo de demostración:** habilite el "**modo de demostración**" durante el proceso de generación de imágenes. Se recomienda esto para evitar 404 errores de imágenes que aún no se han generado. Una vez que el proceso se haya completado, puede desactivar esta opción.

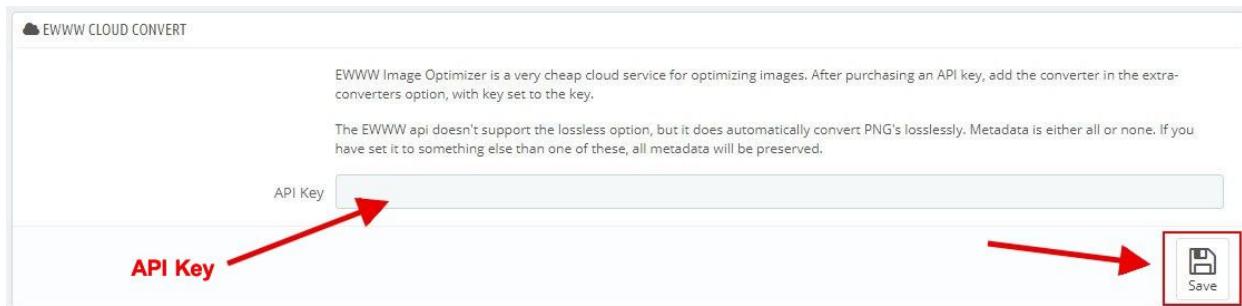


EWWW Cloud Convert

EWWW Image Optimizer es un servicio en la nube muy barato para optimizar imágenes.

Después de comprar una clave API, agregue el convertidor en la opción de convertidores adicionales, con el campo clave que contiene la clave API.

La API EWWW no admite la opción sin pérdida, pero convierte automáticamente los PNG sin pérdida. Los metadatos son todos o ninguno. Si lo ha configurado en otra cosa que no sea uno de estos, todos los metadatos se conservarán.



Configuración de conversión de CWEBP

- **Use el comando** `encuentra en el host, el binario se ejecuta con baja prioridad para ahorrar recursos del sistema.
- **Pruebe las rutas comunes del sistema:** nice`: si el comando` nice` se esté disponible comprueba si cwebp está disponible en una ruta común del sistema (por ejemplo, /usr/bin/cwebp ..)
- **Pruebe el binario suministrado:** si CWebP no está instalado en el servidor, entonces el binario suministrado se selecciona entre Convertidores / Binarios (según el SO) - después de validar la suma de comprobación
- **Activa el filtro automático.** Este algoritmo gastará tiempo adicional optimizando la fuerza de filtrado para alcanzar una calidad bien equilibrada. Desafortunadamente, es extremadamente caro en términos de cómputo. Se tarda aproximadamente 5-10 veces más en hacer una conversión. Una imagen de 1 MB que tal vez demore aproximadamente 2 segundos en convertirse, demorará aproximadamente 15 segundos en convertirse con filtro automático. Así que en la mayoría de los casos, querrá dejar esto en su valor predeterminado, que está desactivado.
- **Opciones de línea de comando:** esto le permite establecer cualquier parámetro disponible para cwebp de la misma manera que lo haría al ejecutar cwebp. Por ejemplo, puede configurarlo en " -sharpness 5 -mt -crop 10 10 40

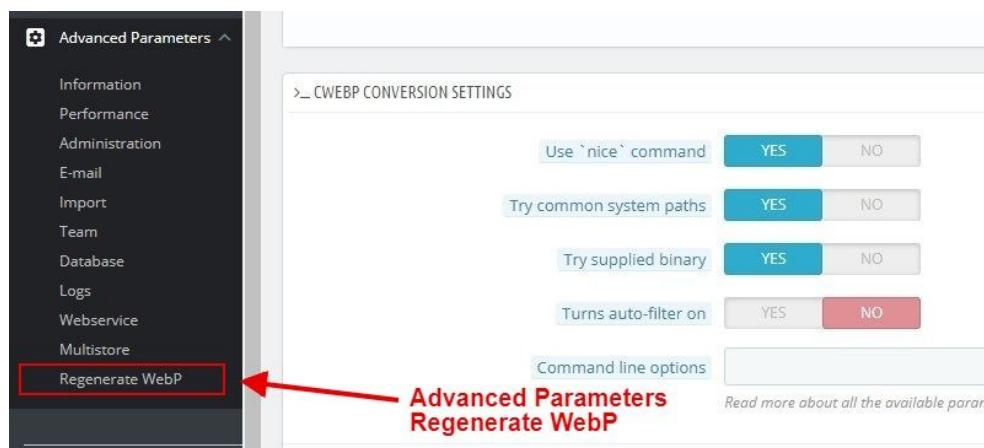


Lea más sobre todos los parámetros disponibles aquí:

https://developers.google.com/speed/webp/docs/cwebp#additional_options

Regenerar WebP

Una vez que haya configurado el módulo, vaya a "**Parámetros avanzados**" en el menú de la izquierda. Elija la "**Regenerar WebP**" pestaña.



Puede elegir qué imágenes desea generar

- Imágenes de productos
- Categoría Imágenes de
- proveedor Imágenes de
- tienda Imágenes de
- fabricante

Una vez que haya elegido, puede comenzar a generar. Puede ver cuántas imágenes hay para cada tipo y la barra de procesamiento mostrará el porcentaje de las imágenes generadas.

The screenshot shows a user interface for generating WebP images. At the top, there's a blue header bar with the text "WebP is a modern image format that provides superior **lossless and lossy** compression for images on the web. Using WebP, webmasters and web developers can create smaller, richer images that make the web faster." Below this, another message states "WebP lossless images are 26% smaller in size compared to PNGs. WebP lossy images are 25-34% smaller than comparable JPEG images at equivalent SSIM quality index." A third message notes that "Lossless WebP **supports transparency** (also known as alpha channel) at a cost of just 22% additional bytes. For cases when lossy RGB compression is acceptable, **lossy WebP also supports transparency**, typically providing 3x smaller file sizes compared to PNG." The main area is titled "REGENARATE WEBP IMAGES". It contains a message: "You can regenerate all your images safely." followed by a list of categories with their current counts and total counts: PRODUCT IMAGES (0/20588), CATEGORY IMAGES (0/0), SUPPLIER IMAGES (0/1), STORE IMAGES (0/5), and MANUFACTURER IMAGES (0/6). To the right of this list is a large red arrow pointing towards a horizontal progress bar consisting of five separate progress indicators, each labeled "0%".

IMPORTANTE: la ventana o pestaña en la que comenzó el proceso de regeneración debe permanecer abierta durante todo el proceso. Cerrarlo interrumpirá el proceso y será necesario reiniciarlo.



Compatibilidad con el navegador WEBP

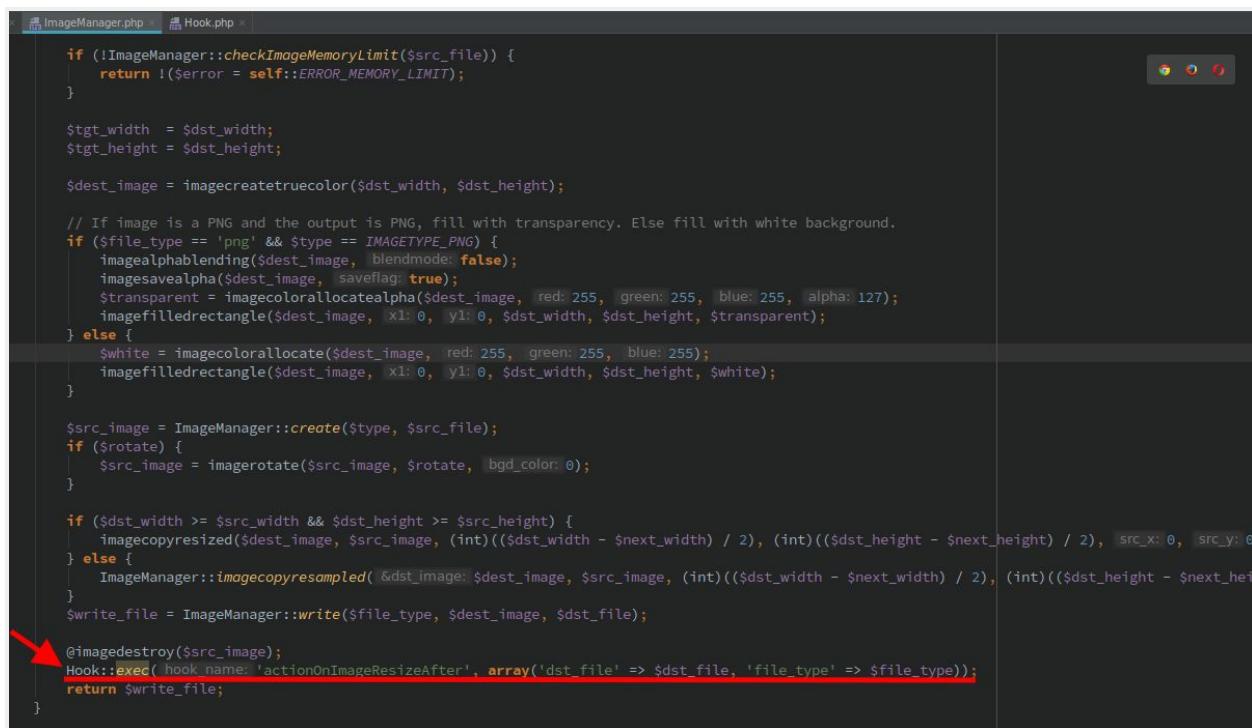
Actualmente solo Google Chrome y Opera admiten imágenes WebP. Aunque otros navegadores, como Firefox, Safari o Internet Explorer actualmente no admiten el formato de imagen de forma nativa, no debe preocuparse. Las imágenes de su sitio no se romperán incluso para los clientes que usan estos navegadores, las imágenes aparecerán en formato PNG o JPG.



Compatibilidad con Prestashop 1.6

Dado que Prestashop 1.6 no ejecuta el `actionOnImageResizeAfter` enganche, debe copiar el siguiente fragmento de código en la `ImageManager` clase redimensionar el método, antes de la declaración de retorno de `$ write_file`:

```
Hook :: exec ('actionOnImageResizeAfter', array ('dst_file' => $ dst_file, 'file_type'  
=> $ file_type));
```



```
if (!ImageManager::checkImageMemoryLimit($src_file)) {  
    return !($error = self::ERROR_MEMORY_LIMIT);  
}  
  
$tgt_width = $dst_width;  
$tgt_height = $dst_height;  
  
$dest_image = imagecreatetruecolor($dst_width, $dst_height);  
  
// If image is a PNG and the output is PNG, fill with transparency. Else fill with white background.  
if ($file_type == 'png' && $type == IMAGETYPE_PNG) {  
    imagealphablending($dest_image, blendmode: false);  
    imagesavealpha($dest_image, saveflag: true);  
    $transparent = imagecolorallocatealpha($dest_image, red: 255, green: 255, blue: 255, alpha: 127);  
    imagefilledrectangle($dest_image, x1: 0, y1: 0, $dst_width, $dst_height, $transparent);  
} else {  
    $white = imagecolorallocate($dest_image, red: 255, green: 255, blue: 255);  
    imagefilledrectangle($dest_image, x1: 0, y1: 0, $dst_width, $dst_height, $white);  
}  
  
$src_image = ImageManager::create($type, $src_file);  
if ($rotate) {  
    $src_image = imagerotate($src_image, $rotate, bgd_color: 0);  
}  
  
if ($dst_width >= $src_width && $dst_height >= $src_height) {  
    imagecopyresized($dest_image, $src_image, (int)((($dst_width - $next_width) / 2), (int)((($dst_height - $next_height) / 2), src_x: 0, src_y: 0));  
} else {  
    ImageManager::imagecopyresampled(&$dest_image, $dest_image, $src_image, (int)((($dst_width - $next_width) / 2), (int)((($dst_height - $next_height) / 2), src_x: 0, src_y: 0));  
}  
$write_file = ImageManager::write($file_type, $dest_image, $dst_file);  
  
@imagedestroy($src_image);  
Hook::exec( hook_name: 'actionOnImageResizeAfter', array('dst_file' => $dst_file, 'file_type' => $file_type));  
return $write_file;
```