Impression d'un PDF

DomPDF, PHP, Wordpress

Web Compétences : 3

Il n'y a que deux simples règles à suivre quand l'option d'impression devient une fonctionnalité ou un besoin dans un site web. La première est la conversion de la page en PDF. La deuxième est l'impression du PDF à proprement parlée. Il ne reste plus qu'à choisir une librairie qui fait ce travail. Suivez le guide...

Publié lundi 10 février 2020, 16h47
Modifié lundi 26 août 2024, 10h06

By Olivier Paudex

Introduction

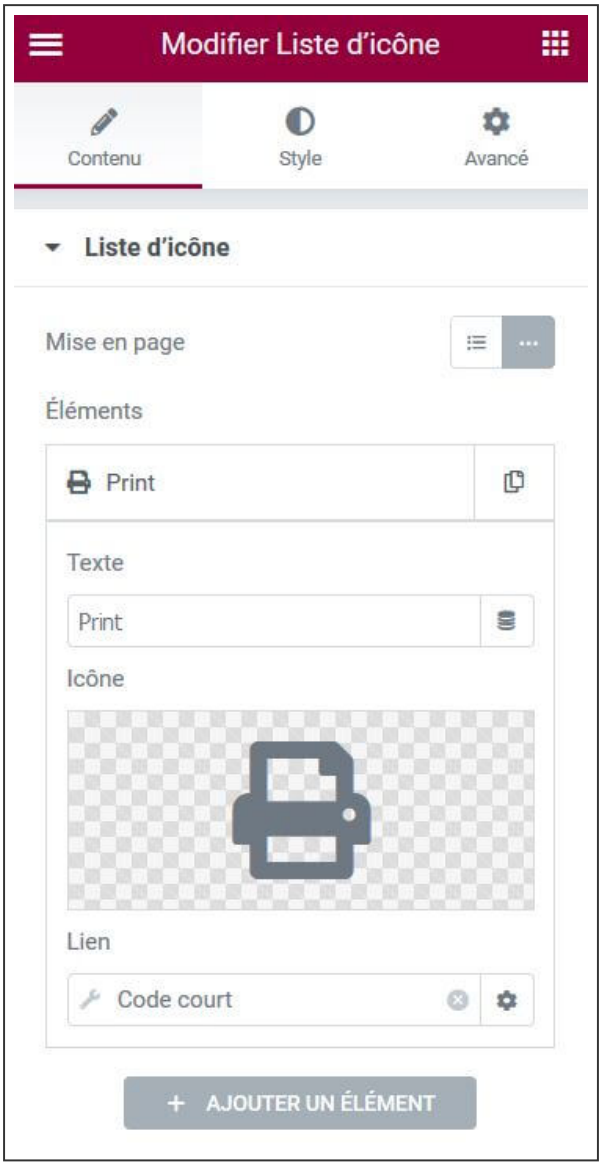
L'impression d'une publication n'est pas une option prévue par WordPress, ni par Elementor. Cela peut paraître logique dans le sens où un site web se doit d'être digital et non sous un format papier. Néanmoins, il est quelquefois bien pratique de pouvoir garder une trace d'une publication avant que celle-ci ne disparaisse. Une autre utilisation de l'impression sera très certainement les formulaires de facturations pour les sites marchands. Quoiqu'il en soit, l'impression à partir d'un site web peut être fastidieuse. Pour rester dans la norme, le mieux est encore de convertir sa page en PDF, puis de l'imprimer. Pour réaliser ceci, WordPress a besoin d'une extension que l'on appelle souvent HTML vers PDF.

Le lien Imprimer

La première tâche à réaliser est très certainement le lien qui va exécuter la création du PDF. Avec Elementor, on peut utiliser un bouton, ou un quelconque widget ayant la possibilité de rediriger vers un URL. Sur ce site, le widget utilisé est la liste d'icône.

Le widget Liste d'icônes

La liste d'icônes permet de créer des liens URL, avec un texte et une petite image issue de la bibliothèque Awesome Font. Comme ce lien va apparaître sur le modèle de publication unique, il se doit d'être dynamique. En d'autres termes, le lien est différent à chaque page. Pour réaliser ceci, une des solutions est de faire appel à un **code court**.

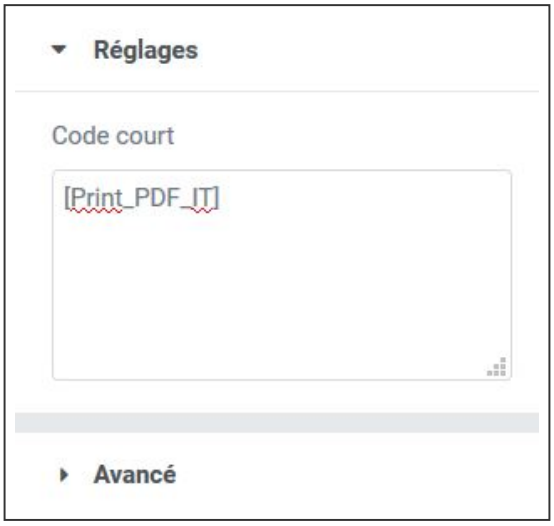


Les réglages du widget

Le code court

Comme énoncé ci-dessus, le code court va remplacer un lien URL et permettre de le rendre dynamique. Le code court n'est rien d'autre qu'un appel vers une fonction PHP, qui va retourner l'URL en fonction du titre de la publication.

L'appel au code court se fait entre crochets, ici **[Print_PDF_IT]**.



L'appel à un code court

Une fois le widget configuré avec le nom du code court, il faut ajouter celui-ci au thème enfant. Pour réaliser ceci, il faut créer un nouveau fichier **PHP** et y insérer le code ci-dessous.

Le lien URL va être créé à partir de l'adresse du site web, puis le dossier **"pdf-it"** et enfin le nom de la publication. Exemple pour une publication dont le titre est **"Le meilleur de WordPress"**, le lien donnerait **"https://www.fuyens.ch/pdf-it/le-meilleur-de-wordpress"**.

Remarquez ici l'utilisation d'une variable globale **$post** qui contient la publication actuelle. Elle permet de récupérer entre autres, le titre.

```
function print_pdf_it() {
  global $post;
  return home_url('pdf-it/') . $post->post_name;
}add_shortcode ('Print_PDF_IT','print_pdf_it');
```

Le modèle WordPress

Créer un modèle WordPress reste tout à fait d'actualités, même si le site est réalisé avec Elementor. Même si ce dernier permet de créer des modèles réutilisables, avec son interface GUI, il est encore possible de le faire à la manière de WordPress, c'est-à-dire par code PHP. La littérature WordPress abonde d'exemples citant la fameuse **"Boucle WordPress"**. Au début de la création de ce site web, il y a un peu plus d'un an, alors que l'outil choisi pour élaborer les pages avait été Elementor, l'utilisation de cette fameuse boucle n'était absolument pas évidente. Et pourtant cela reste très simple. Voici, les trois grosses étapes à suivre.

- Créer un modèle en PHP
- Créer une page vide
- Lier la page et le modèle

Le modèle PHP

Pour créer un modèle PHP, il faut créer un nouveau fichier PHP et lui insérer une entête. Dans l'exemple ci-dessous, le fichier PHP se nomme **"pdf-it.php"** et le nom du modèle est **"IT PDF"**.

La recommandation va de créer un nouveau dossier, dont le nom est **"templates",** à la racine du thème enfant, soit au même niveau hiérarchique que les dossiers **"php, css, js"**, et d'y glisser le fichier PHP.

> La ligne vraiment indispensable est celle-ci : **The template name: IT PDF**.

Les trois premières lignes de code en dessous de l'entête sont ici pour une raison de sécurité. Le modèle ne peut être appelé qu'avec un nom de chemin absolu. Tous les modèles devraient commencer avec ces lignes.

```
<?php
  /**
  * The template name: IT PDF
  * Description: IT PDF Template
  * Author: Olivier Paudex
  * Author Web Site: https://www.fuyens.ch
  */
  if ( ! defined( 'ABSPATH' ) ) {
   exit; // Exit if accessed directly.
  }?>
```

La page vide et la liaison avec le modèle

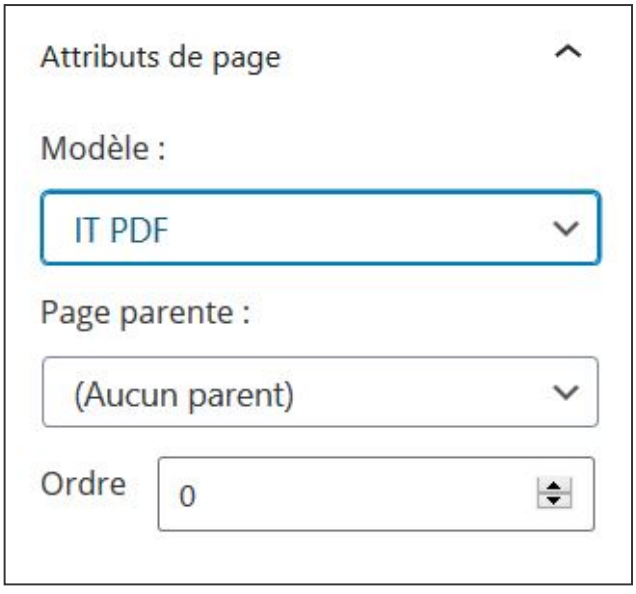
Une fois le modèle créé, il faut créer une page vide dans WordPress, car celui-ci ne peut appeler directement un modèle PHP. Le nom de cette page doit correspondre à celui donné au lien URL. Dans cet exemple, ce sera **"PDF-IT"**.

En réalité ce n'est pas vraiment le nom de la page qui est important, mais son**"slug"**, ainsi que son chemin URL, appelé aussi "**permalien"**. Il est possible de le voir dans les paramètres de la page à droite.

> Dans l'exemple ci-dessous, le permalien comporte encore un dossier **"en"**. Le site a été créé en deux langues, français et anglais. Au moment de la capture d'écran, le dossier de la langue s'est automatiquement ajouté. Ce blog contient un chapitre concernant le multilinguisme, appelé **Polylang**. Il fait la part belle au plugin Polylang avec lequel ce site a été réalisé.

Le dernier réglage à modifier est l'attribut de page. Il faut sélectionner le modèle créé auparavant avec PHP, pour lier la page avec le code.

Le slug URL et le permalien



La liaison avec le modèle PHP

Une fois cette étape réalisé, tous les liens qui commenceront par **"https://www.fuyens.ch/pdf-it/..."**, exécuteront le modèle **"IT PDF"**. Cela reste simple et tout à fait compatible avec les modèles Elementor.

Les librairies PDF

Une fois la structure créée, il faut choisir une librairie qui va convertir le code HTML de la page, en PDF. Pour réaliser ceci, plusieurs outils sont disponibles.

En voici quelques-uns :

FPDF

FPDF est un outil orienté PHP. Il exécute du code PHP pour créer un PDF. Entièrement gratuit et très pratique pour créer des formulaires de facturation avec des tableaux, pour un site marchand, par exemple. Il reste très simple d'emploi, n'utilise pas d'autres librairies en dehors de la sienne. Son site web est très documenté. Je le recommande chaudement pour réaliser des formulaires PDF.

mPDF

mPDF est une librairie qui utilise FPDF et HTML2FPDF. L'avantage par rapport à FPDF est sans aucun doute celui de pouvoir convertir du code HTML en PDF. De plus, il a la possibilité d'utiliser des feuilles de styles CSS pour modifier l'esthétisme de la page. Son installation demande une maîtrise un peu avancé, utilisant l'outil **"Composer"**, qui n'était pas compatible avec l'hébergeur sur lequel ce site a été installé.

TCPDF

TCPDF est certainement l'outil gratuit le plus élaboré à ce jour. Il peut convertir du HTML vers le PDF, créer des entêtes, utiliser des polices d'écritures truetype. C'est à ce jour le seul a pouvoir exécuter du code Javascript. Il possède tellement de possibilités qu'il en est devenu presque trop complexe à mettre en œuvre. A l'instar de mPDF, il est entièrement autonome. Seul bémol à sa pérennité, il n'est plus supporté. Son créateur ayant décidé de se lancer dans un tout nouveau projet, encore en développement, **tc-lib-pdf**
.

DomPDF

DomPDF est une librairie entièrement autonome et gratuite. Elle est très simple à mettre en place dans une infrastructure WordPress. Elle est compatible CSS 2.1 avec quelques fonctions CSS 3.0 (mais pas les flexbox) qui peuvent être des fichiers externes. Elle est compatible avec les polices d'écritures externes et comprend la plupart des syntaxes écrites en HTML 4.0.

L'heure du choix

A l'heure du choix de la librairie PDF, la préférence a été donnée à**domPDF**. Les raisons sont multiples, mais en voici quelques unes. **FPDF** ne permettait pas de convertir nativement du HTML en PDF. Il reste un produit fiable pour d'autres projets. **mPDF** est compliqué à installer et demande l'utilisation de **"Composer"**, outil que l'hébergeur de mon site refusait d'exécuter. **TCPDF** est certes la librairie la plus avancée, mais plus supportée. En attendant la future version de **"tc-lib-pdf"**. De plus, le code à fournir est relativement complexe à assimiler. Sur les quatre produits testés, restait **domPDF**, qui paraissait, malgré certaines faiblesses, correspondre le plus aux besoins de ce site.

Il en existe bien entendu d'autres, comme **wkhtmltopdf**, qui demande l'installation d'un exécutable sur le serveur. Cette manipulation est strictement interdite chez la plupart des hébergeurs. Et cette liste ne tient pas compte de tous les produits payants, qui, pour la plupart, sont compatible HTML 5.0 et CSS 3.0, ce qui rend la compatibilité totale avec Elementor. Ce serait vraiment génial de voir un jour une librairie comme **DomPDF** intégré le CSS 3.0. A suivre…

Installation de DomPDF (version 0.85)

Comme énoncé ci-dessus, le choix s'est porté sur la librairie DomPDF. Pour l'installer, il faut bien entendu avoir au préalable préparer le terrain et utiliser un thème enfant, maintes fois citer dans ce blog. Ci-dessous une liste des différentes étapes à réaliser.

- Téléchargez la librairie sur **GitHub** ou en cliquant sur le lien au sommet de cette publication
- Ouvrez le fichier ZIP est copiez le dossier **"dompdf"** à la racine du thème enfant

Et c'est tout, simple non ?

Polices d'écriture locales et feuille de styles CSS

Pour étendre les possibilités de DomPDF, il est recommandé de créer deux nouveaux fichiers CSS :

- Custom_Fonts.css
- DomPDF_Styles.css

Il n'est pas nécessaire de mettre la feuille de style pour domPDF en queue, mais il est conseillé de le faire pour les polices, au cas ou celles-ci seraient utilisées ailleurs que dans domPDF.

Pour rappel, insérez le code ci-dessous pour mettre en queue un fichier CSS.

```
// Enqueue Custom CSS Fonts
function Custom_Fonts() {
  wp_enqueue_style('Custom_Fonts', get_stylesheet_directory_uri() . '/css/Custom_Fonts.css', fa
}add_action('wp_enqueue_scripts', 'Custom_Fonts');
```

Premier fichier PDF

Il ne reste plus qu'à compléter le modèle PHP pour créer le premier PDF.

Ajoutez une référence aux fichiers de polices et feuille de styles.

```
// Load Fonts & Styles
$fonts = $dir . '/css/Custom_Fonts.css';$css = $dir . '/css/DomPDF_Styles.css';
```

Ajoutez une référence pour indiquer où se trouve le fichier **"autoload.inc.php"**

```
// Load domPDF library
$dir = get_stylesheet_directory();require_once ( $dir . '/dompdf/autoload.inc.php');
```

Récupérez la publication dans une variable **"$post"**. Voici une astuce pour le faire depuis l'URL grâce à la fonction WordPress **"get_posts"**.

```
// Get current URL
global $wp;
$current_url = home_url(htmlspecialchars($wp->request));

// Get post from slug
$args = array(
  'name'   => substr(strrchr($current_url, "/"),1),
  'post_type'   => 'it',
  'post_status' => 'publish',
  'numberposts' => 1
);$post = get_posts($args)[0];
```

Récupérez le permalien, l'auteur, le titre et le contenu.

```
// Permalink
$permalink = get_permalink($post);
// The author
$authorID = $post->post_author;
$authorName = get_the_author_meta('display_name', $authorID);
// Get title & content
$title = $post->post_title;$content = apply_filters('the_content', $post->post_content);
```

Ecrivez le code HTML qui va être converti en PDF. Le code ci-dessous est un exemple qui va afficher le titre, l'auteur et le contenu de la publication.

Englobez tout le code HTML dans une variable **"$html"**.

Notez ici que deux classes, **"author"** et **"content"** ont été utilisées pour permettre de modifier ces éléments.

```
// HTML
$html = '
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" media="screen" type="text/css" title="CSS" href="' . $css . '"/>
    <link rel="stylesheet" media="screen" type="text/css" title="CSS" href="' . $fonts . '"/>
    <title>' . $title . '</title>
  </head>
  <body>
    <div>
      <!-- Title -->
      <h1>' . $title . '</h1>
      <div class="author">' . 'By' . ' ' . $authorName . '</div>
      <!--Content -->
      <div class="content">' . $content . '</div>
    </div>
  </body>
</html>
';
```

Dernière étape, référencez et instancier un nouveau document PDF, puis :

- Passez le code HTML.
- Choisissez l'orientation du papier.
- Utilisez l'analyseur de syntaxe **"isHtml5ParserEnabled"**, qui va simplifier le code et surtout le rendre lisible.
- Activez la prise à distance avec **"isRemoteEnabled"**, pour permettre le téléchargement de fichiers externes comme les polices d'écriture, le CSS ou même les images.
- Choisissez une résolution en DPI. Ici, le PDF sera affiché en **98dpi**, ce qui correspond à une page de **810 x 1146 pixels**.
- Enfin, affichez le contenu à l'écran avec les fonctions **"render"** et **"stream"**. Ici le titre de la publication est utilisé comme titre pour le document PDF et sera directement affiché dans le navigateur.

```
// Reference the Dompdf namespace
  use Dompdf\Dompdf;

  // Instantiate and use the DomPDF class
  $document = new Dompdf();
  $document->loadHtml($html);
  // Setup the paper size and orientation
  $document->setPaper('A4', 'portrait');
  // Options
  $document->set_option('isHtml5ParserEnabled', true);
  $document->set_option('isRemoteEnabled', true);
  $document->set_option('dpi', '98');
  // Render the HTML as PDF
  $document->render();
  // Preview the generated PDF to Browser
  $document->stream($title . ".pdf", array("Attachment"=>0));
```

Ci-dessous, le code complet du modèle.

```
<?php
  /**
  * The template name: IT PDF
  * Description: IT PDF Template
  * Author: Olivier Paudex
  * Author Web Site: https://www.fuyens.ch
  */
  if ( ! defined( 'ABSPATH' ) ) {
   exit; // Exit if accessed directly.
  }
  // Load Fonts & Styles
  $fonts = $dir . '/css/Custom_Fonts.css';
  $css = $dir . '/css/DomPDF_Styles.css';
  // Load domPDF library
  $dir = get_stylesheet_directory();
  require_once ( $dir . '/dompdf/autoload.inc.php');
  // Get current URL
  global $wp;
  $current_url = home_url(htmlspecialchars($wp->request));

  // Get post from slug
  $args = array(
    'name'   => substr(strrchr($current_url, "/"),1),
    'post_type'   => 'it',
    'post_status' => 'publish',
    'numberposts' => 1
  );
  $post = get_posts($args)[0];
  // Permalink
  $permalink = get_permalink($post);
  // The author
  $authorID = $post->post_author;
  $authorName = get_the_author_meta('display_name', $authorID);
  // Get title & content
  $title = $post->post_title;
  $content = apply_filters('the_content', $post->post_content);
  // HTML
  $html = '
  <!DOCTYPE html>
  <html>
    <head>
      <meta charset="utf-8" />
      <link rel="stylesheet" media="screen" type="text/css" title="CSS" href="' . $css . '"/>
      <link rel="stylesheet" media="screen" type="text/css" title="CSS" href="' . $fonts . '"/>
      <title>' . $title . '</title>
    </head>
    <body>
      <div>
        <!-- Title -->
        <h1>' . $title . '</h1>
        <div class="author">' . 'By' . ' ' . $authorName . '</div>
        <!--Content -->
        <div class="content">' . $content . '</div>
      </div>
    </body>
  </html>
  ';
  /*************************************************************************************************
  // Reference the Dompdf namespace
  use Dompdf\Dompdf;

  // Instantiate and use the DomPDF class
  $document = new Dompdf();
  $document->loadHtml($html);
  // Setup the paper size and orientation
  $document->setPaper('A4', 'portrait');
```

```
  // Options
  $document->set_option('isHtml5ParserEnabled', true);
  $document->set_option('isRemoteEnabled', true);
  $document->set_option('dpi', '98');
  // Render the HTML as PDF
  $document->render();
  // Preview the generated PDF to Browser
  $document->stream($title . ".pdf", array("Attachment"=>0));?>
```

La feuille de style CSS

Comme cité ci-dessus, il est possible de formater le PDF avec le CSS.

Ajoutez le code ci-dessous au fichier **"DomPDF_Styles.css"**. Il va formater le tag HTML **"H1"** et les classes **"author"** et **"content"**.

```
/* Heading */
h1 {
  font-family: "Open Sans", Sans-Serif;
  font-size: 35px;
  font-weight: 800;
  letter-spacing: -1.5px;
  text-align: center;
  color: #333333;
  line-height: 1.3em;
  margin: -10px 0px 20px 0px;
}
/* Content */
.content {
  font-family: "Open Sans", Sans-Serif;
  font-weight: 400;
  font-size: 15px;
  letter-spacing: -0.5px;
  line-height: 1em;
  margin: -10px 0px 20px 0px;
}
/* Author */
.author {
  font-family: "Open Sans", Sans-Serif;
  font-weight: 300;
  font-style: italic;
  font-size: 15px;
  letter-spacing: -0.5px;
  line-height: 1em;}
```

Les polices d'écriture en locale

En ce qui concerne les polices d'écriture, on peut choisir de les charger directement à partir du site web ou en CDN (Content Delivery Network). L'exemple ci-dessous montre la version locale.

Il faut savoir que pour chaque police qui se décline en **gras**, **italique**, etc..., il faut un fichier différent. On peut trouver tous les fichiers sur **Google Fonts**.

Ajoutez le code ci-dessous au fichier **"Custom_Fonts.css"**. Il va appliquer la police choisie au texte grâce à la méthode CSS **"font-family"**.

```
/* Open Sans */
@font-face {
  font-family: "Open Sans";
  src:  url('/wp-content/themes/hello-theme-child-master/fonts/opensans/OpenSans-Regular.ttf') format('
  font-weight: 400;
}
@font-face {
  font-family: "Open Sans";
  src: url('/wp-content/themes/hello-theme-child-master/fonts/opensans/OpenSans-LightItalic.tt
  font-weight: 300;
  font-style: italic;}
```

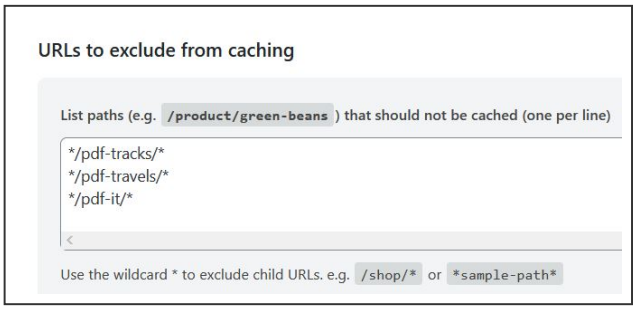
La mise en cache

Mettre à disposition des visiteurs des pages préchargées ou mise en cache rends le site bien plus rapide et l'installation d'un plugin de mise en cache est tout naturellement à effectuer sur WordPress. Ce site ne fait pas exception à la règle et utilise le plugin **Wp-Optimize**.

Créer des pages PDF signifient pourtant une limitation au préchargement des pages. Comme celles-ci sont créées à la volée depuis le code HTML de la page, elles ne peuvent être mise en cache. Il faut donc les exclure.

Avec **Wp-Optimize**, ceci se fait dans l'onglet avancé. On peut y exclure les URL qui ne doivent pas être pris en compte.

Le plugin WP-Optimize



L'exclusion des dossiers contenant les PDF

Le mot de la fin

Le chapitre sur l'impression des PDF était un gros challenge pour parvenir d'un côté à créer des PDF imprimables, mais surtout de pouvoir rendre le contenu dynamique grâce à WordPress. Avec tous ces aspects en tête, les exemples ci-dessus restent réutilisables dans tous les cas. Cela donne un bon point de départ sur la façon d'imprimer une publication créé avec WordPress et Elementor, ce qui était l'objectif.