WkHtmlToPDF

Présentation :

Les exports en PDF lors de la création d'une application web font parties des fonctionnalités récurrentes. Pourtant, malgré le nombre de libraries dédiées (DomPDF, FPDF, API de conversion, …), la mise en œuvre est toujours fastidieuse, et incomplète !

Le problème de la plupart de ces bibliothèques, est de devoir générer un document PDF « from scratch », via la création manuelle du contenu du document ainsi que sa mise en page avec des outils (programmiques) qu'il n'est pas aisé de manipuler…

Enfin, dans le cas des API de conversion, il est encore moins pratique de maitriser la transformation que celles-ci feront de votre page web. De plus, cette conversion est rarement transparente pour l'utilisateur. (Pop-ups, …)

Les principales difficultés sont donc :

- D'obtenir une mise en page et le rendu graphique souhaité.
- Devoir créer un template dédié avec des outils peu pratiques.
- L'impossibilité pour la plupart des bibliothèques d'intégrer des fonctionnalités CSS3.
- Maîtriser l'appartenance de votre document.
- Générer rapidement le document. Gérer un système de cache.
- Et bien d'autres ! …

Ce sont pour toutes ces raisons que je vous propose d'installer un utilitaire appelé **WkHtmlToPDF** pour - vous ne le devinerez jamais – générer des PDF ! Et ce, quel que soit le langage serveur que vous utiliserez.

« Oui mais, tu ne viens pas de nous dire que ces bibliothèques sont loin d'être adaptées pour le développement fonctionnel et rapide d'une application web ? »

Si, si ! Cependant, WkHtmlToPDF n'est pas une simple library PHP ou autre. Il s'agit d'un utilitaire qui tournera sur votre serveur auquel vous pourrez faire appel pour générer votre PDF.
Amazing thing : L'utilitaire ne prend qu'un seul paramètre : l'URL !

Vous pouvez donc génèrer un PDF pour n'importe quelle URL, avec un rendu parfait, identique à celui d'un navigateur utilisant le moteur Webkit (Google Chrome, Safari, …).
Hé oui, WkhtmlToPDF, c'est pour « Webkit Html To PDF », autrement dit, la génération de PDF depuis une URL via l'utilisation du moteur de rendu Webkit !

Ce qui veut dire :

- Aucune utilisation d'outil fastidieux pour générer votre document.
- Pas besoin de template particulier. (Pensez tout de même à créer une feuille de style supplémentaire afin d'épurer vos pages si le PDF est destiné à l'impression.)
- Rendu tel que dans le navigateur : Prise en charge de toutes les propriétés CSS3 gérées par webkit pour un rendu parfait, sans se compliquer la vie.

- La gestion possible des sauts de pages en CSS via les attributs page-break-after/before : http://www.w3.org/TR/css3-page/
- La génération est extrêmement performante. Il n'appartient plus qu'à vous de stocker vos documents générés dans un dossier de cache, plutôt que de régénérer la page à chaque fois.
- Amazing feature : il est possible d'attendre le chargement des scripts JavaScript avant de générer la page. Très utile en cas de requêtes AJAX !
- Il vous en faut d'autres ? …

Téléchargement :

Pour télécharger Wkhtmltopdf, rendez-vous à cette adresse :
https://code.google.com/p/wkhtmltopdf/

Il existe une version Windows, tout comme pour Mac et Linux.

Ou plus simplement, pour debian :

```
apt-get install wkhtmltopdf
```

La liste des arguments disponibles : http://madalgo.au.dk/~jakobt/wkhtmltoxdoc/wkhtmltopdf-0.9.9-doc.html

Pour l'utiliser, c'est très simple :

```
wkhtmltopdf http://google.com test.pdf
```

Et là, c'est le drame.
Vous êtes sur votre environnement de production, à 90% je parie Debian, et vous obtenez l'erreur suivante :

```
wkhtmltopdf: cannot connect to X server
```

Problématique

Si vous obtenez cette erreur, alors que tout fonctionne parfaitement sur votre environnement de développement local, c'est parce que votre serveur distant ne possède pas de Server X. Server X nécessaire à la génération du rendu via webkit.

-« Hein ?! Mais ça ne sert à rien alors ! Je ne vais pas installer un environnement graphique sur mon serveur de prod, c'est useless ! »

Du calme ! Il existe heureusement plusieurs solutions :

- Emuler un server X.
- Recompiler la bibliothèque Qt ainsi que wkhtmltopdf avec une version patchée.

N'ayez crainte…Nous allons recompiler.

« WHAT ?! Pourquoi ne pas émuler ? »

Parce que c'est bien trop simple :p
Et cela n'offre pas les mêmes possibilités.
De plus, si vous utilisez une version non patchée, la plupart des bibliothèques PHP utilisant wkhtmltopdf ne fonctionneront pas.

Moteur… Action !

Commençons par supprimer le paquet précédent :

```
apt-get remove wkhtmltopdf –purge
```

Puis, passons aux choses sérieuses…

> WARNING : Cette étape est assez longue dû aux téléchargements et surtout à la compilation de l'application et Qt lui-même avec le patch.

Commencez par récupérer les packets essentiels au fonctionnement de wkhtmltopdf ainsi que pour compiler Qt :

```
sudo apt-get build-dep libqt4-gui libqt4-network libqt4-webkit
sudo apt-get install openssl build-essential xorg git-core git-doc libssl-dev
```

Placez-vous dans le répertoire souhaité, puis créez un nouveau dossier :

```
mkdir ~/sources
cd ~/sources
```

Récupérez les dernières sources de WkhtmltoPdf depuis github :

```
git clone git://github.com/antialize/wkhtmltopdf.git wkhtmltopdf
```

Bien sûr, vous devez avoir git installé pour exécuter cette commande.

```
sudo apt-get install git-core
```

Ensuite, il va falloir récupérer la version modifiée de Qt.
Anciennement, le repo git était à l'adresse suivante :
git://gitorious.org/+wkhtml2pdf/qt/wkhtmltopdf-qt.git wkhtmltopdf-qt

Cependant le repository a été déplacé. Exécutez donc la commande suivante :

```
git clone git://gitorious.org/~antialize/qt/antializes-qt.git wkhtmltopdf-qt
cd wkhtmltopdf-qt
git checkout 4.8.4
QTDIR=. ./bin/syncqt
```

Comme vous pouvez le constater, le téléchargement est assez long (432Mo).
Vous pouvez bien évidemment partir prendre un café.

Qt doit être configuré. Voici comment le faire rapidement avec la configuration de base proposée :

```
cat ../wkhtmltopdf/static_qt_conf_base ../wkhtmltopdf/static_qt_conf_linux | sed -re
's/#.*//'
./configure -nomake tools,examples,demos,docs,translations -opensource -prefix "../wkqt"
```

Pensez à accepter la licence avant de partir prendre votre second café.

Passons à la première phase de compilation : Qt.
C'est la partie la plus longue. Vaquez à vos occupations, allez manger, prenez 2-3 cafés. Selon votre serveur, il vous faudra entre 1h et 2h...

```
make -j3 && make install
cd ..
```

Enfin, nous allons pouvoir compiler et installer notre version modifiée de WkhtmltoPdf, et profiter ainsi des fonctionnalités supplémentaires. Il ne sera alors plus nécessaire que votre serveur soit doté d'un server X pour utiliser la commande :

```
cd wkhtmltopdf
../wkqt/bin/qmake
make && make install
```

Vous pouvez dorénavant utiliser la commande wkhtmltopdf sans erreur.

Voici un exemple d'utilisation en PHP :

```php
//On construit la commande :
$output = array();
$returnVar = null;
$command= 'wkhtmltopdf -T 5mm -B 5mm -L 5mm -R 5mm';
$command.= " ".$url;
$command.= " ".$pdfPath;

//Exécution de la commande et récupération des erreurs et code de sortie :
$out = exec(
            $command,
            $output,
            $returnVar
 );
```

Avec émulation

Enfin, si vous ne souhaitez pas vous embêter avec toutes ces phases de compilation, sachez qu'il est également possible d'utiliser la version simple de wkhtmltopdf en simulant un environnement doté d'un serveur X :

1. Télécharger wkhtmltopdf : http://code.google.com/p/wkhtmltopdf/downloads/list
   L'extraire dans /usr/bin/

2. Renommer wkhtmltopdf afin d'obtenir un exécutable à l'adresse **/usr/bin/wkhtmltopdf**
   N'oubliez pas de donner la permission pour l'exécution :
   ```
   sudo chmod a+x /usr/bin/wkhtmltopdf
   ```
   Ni d'installer les packets requis :
   ```
   sudo apt-get install openssl build-essential xorg libssl-dev
   ```

3. Vérifier que tout fonctionne :
   ```
   wkhtmltopdf http://www.google.com test.pdf
   ```
   Vous devriez maintenant obtenir l'erreur "Cannot connect to X server" si vous êtes sur votre serveur de production, sans server X. Si non, vous avez fait tout cela pour rien et auriez pu faire un simple : apt-get install wkhtmltopdf ...

4. Nous devons maintenant émuler un server X. Nous ferons cela avec un package nommé xvfb:
   ```
   sudo apt-get install xvfb
   ```

5. Il faut maintenant écrire un petit script pour encapsuler wkhtmltopdf dans xvfb.
   Créez donc un fichier wkhtmltopdf.sh dans lequel vous mettrez le code suivant:
   ```
   xvfb-run -a -s "-screen 0 640x480x16" wkhtmltopdf $*
   ```

6. Déplacez ce script dans **/usr/bin**, et n'oubliez pas de donner les permissions pour l'exécution.

7. Enfin, faites un lien symbolique vers ce script là où vous le souhaiter, afin d'utiliser facilement le service depuis PHP par exemple : /var/www/libs/whtmltopdf/.
   ***RAPPEL** : Pour faire un lien symbolique :*
   ```
   ln -s /usr/bin/wkhtmltopdf.sh /var/www/libs/wkhtmltopdf
   ```

Cette méthode présente quelques inconvénients, comme le fait de ne pas pouvoir profiter des extensions fournies à la library grâce au patch Qt, ainsi que le fait de devoir émuler un server X via xvfb.
Il se peut également que la plupart des bibliothèques encapsulant la commande ne fonctionnent pas.
Cependant, la procédure est bien plus rapide.

Wrappers

Vous trouverez également des "Wrappers" (Encapsuleurs) Php et Symfony 2, pour intégrer facilement wkhtmltopdf, sans devoir faire appel de façon brut au script :

Wrapper PHP : http://mikehaertl.github.io/phpwkhtmltopdf/
Wrapper Symfony 2 : https://github.com/KnpLabs/KnpSnappyBundle

Exemple d'utilisation avec le wrapper Symfony 2 :

```
1.  /** Création des dossiers si non-existants :
2.   *  Bien sûr, à  vous de créer les variables nécessaires
3.   */
4.  if (!is_dir($uploadDirRoot))     mkdir($uploadDirRoot);
5.  if (!is_dir($uploadDir))         mkdir($uploadDir);
6.
7.  /* Si le fichier n'existe pas, ou n'est pas assez récent,
8.   * on régénère le PDF : */
9.   if(!is_file($pdfPath)){
10.
11.         $openDir=opendir($uploadDir);
12.         while ($file=readdir($openDir)) {
13.                 if($file!=in_array($file, array(".","..")){
14.                         unlink($uploadDir.$file);
15.                 }
16.         }
17.         closedir($openDir);
18.
19.         $baseUrl = $this->getRequest()->getHost();
20.         $url = 'http://'.$baseUrl.$this->generateUrl('my_route');
21.
22.         /** Exemple de generation avec configuration à la volée : **/
23.         $this->get('knp_snappy.pdf')->generate($url, $pdfPath, array(
24.                         'lowquality'                    => false,
25.                         'page-size'                     => 'A3',
26.                         'title'                         => "Votre-Titre.pdf",
27.                         'enable-javascript'             => true,
28.                         'javascript-delay'              => 1500,
29.                         'margin-bottom'           => '5mm',
30.                         'margin-left'                   => '5mm',
31.                         'margin-right'                  => '5mm',
32.                         'margin-top'                    => '5mm',
33.                         'image-quality'                 => '100',
34.                         'no-stop-slow-scripts'          => true,
35.                     ));
36.         }
37. /* *
38. *  Retourne la réponse avec les bons headers :
39. *  Content-Disposition permet de définir le nom sous lequel
40. *  l'utilisateur enregistrera le PDF par défaut.
41. * */
42. return new Response(
43.         file_get_contents($pdfPath),
44.         200,
45.         array(
46.                 'Content-Type'        => 'application/pdf',
47.                 'Content-Disposition' => 'filename="'.$pdfDisplayedName.'"',
48.         )
49. );
```

Dans votre app/config.yml, indiquez le chemin vers votre exécutable :

```
#Pdf Generation
knp_snappy:
    pdf:
        enabled:    true
        binary:     /usr/bin/wkhtmltopdf
        options:    []
    image:
        enabled:    false
        binary:     /usr/local/bin/wkhtmltoimage
        options:    []
```

Sachez qu'il est également possible de générer des images de la même façon avec wkhtmltoimage.

Conclusion

WebKit Html to PDF est un outil performant, bien que compliqué à mettre en place dans certains environnements. Il est cependant extrêmement simple d'utilisation, contrairement aux bibliothèques dédiées permettant la création de PDF « from scratch » avec des outils peu adaptés à l'esprit web.

L'outil permet tout simplement de générer à partir d'une URL un rendu PDF identique à la page HTML tel que l'afficherait un navigateur equipé du moteur webkit ; avec tous les avantages que cela implique.

De nombreux wrappers existent déjà, notamment en PHP et pour le framework Symfony 2 afin d'en faciliter l'usage.