Exécuter PHP en CLI

Table des matières

1. Introduction....2
2. Premier programme....2
3. Exécution en CLI....4
4. Les arguments....5

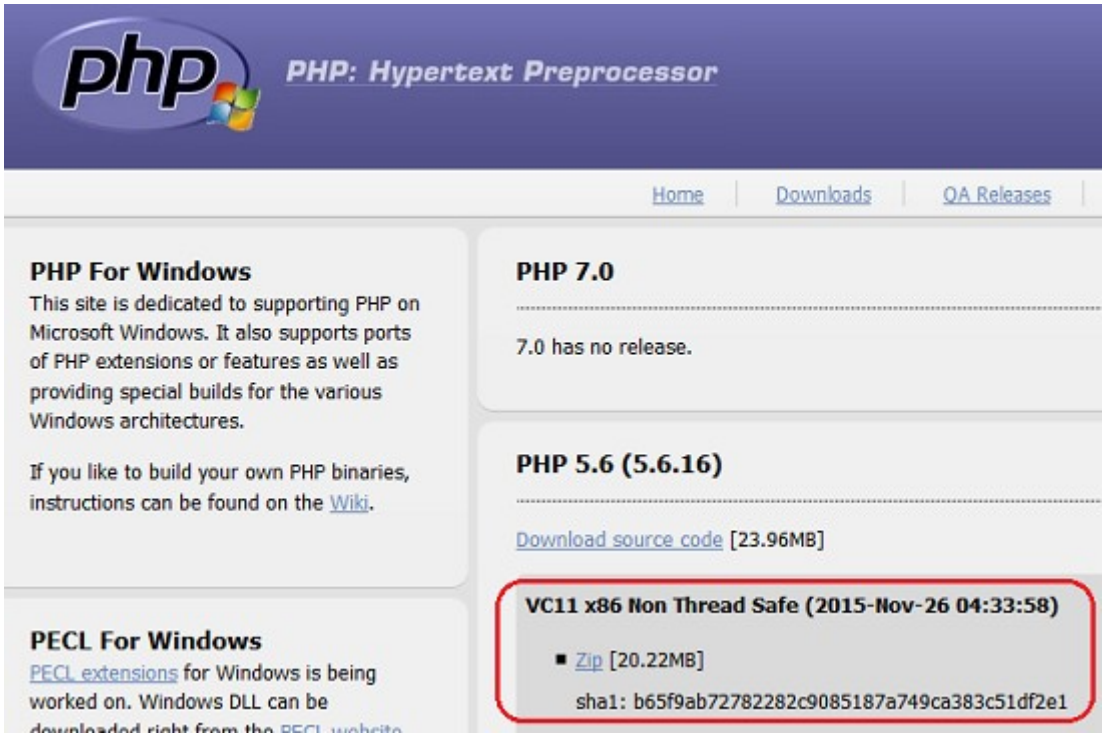
PHP est un langage simple, puissant et modulaire qui peut s'exécuter à partir de la ligne de commande sous GNU/Linux et sous M$-Windows, en mode scripting, à la manière du langage Perl.

1. Introduction

A l'origine PHP était un langage côté serveur, c'est-à-dire qui ne servait surtout qu'à générer des pages en (X)HTML. Cela reste sa principale utilisation. Cependant l'exécution de PHP est possible en CLI[1].

Avant de commencer à tester des scripts PHP, il faut télécharger PHP car l'ordinateur n'est pas capable d'interpréter du PHP. Pour ce faire, rendez-vous sur la page des téléchargements de php.net.

NB : pour des version M$-Windows, utilisez ce lien.



Choisir 'Zip' en dessous de la version la plus récente de PHP

Une fois l'archive téléchargée, créez un dossier 'PHP' dans vos documents personnels sur votre disque dur et extrayez l'archive dans ce répertoire. Puis renommez le fichier "php.ini-development" en "php.ini".

Ce dernier est un fichier texte que vous pouvez éditer pour paramétrer l'exécution de l'interpréteur PHP.

2. Premier programme

Créez un fichier texte avec l'extension .php et ouvrez ce fichier à l'aide de l'éditeur qui vous sert habituellement à scripter en PHP (Bloc-Notes, NotePad++, etc). Ensuite, mettez les tags PHP, et ajoutez un echo ou un print, avec comme paramètre la chaîne "Hello World !".

```php
<?php
	// programme test
	print("Hello World !");
?>
```

1 Command-Line Interface : Interface en Ligne de Commande

Enregistrez et fermez.

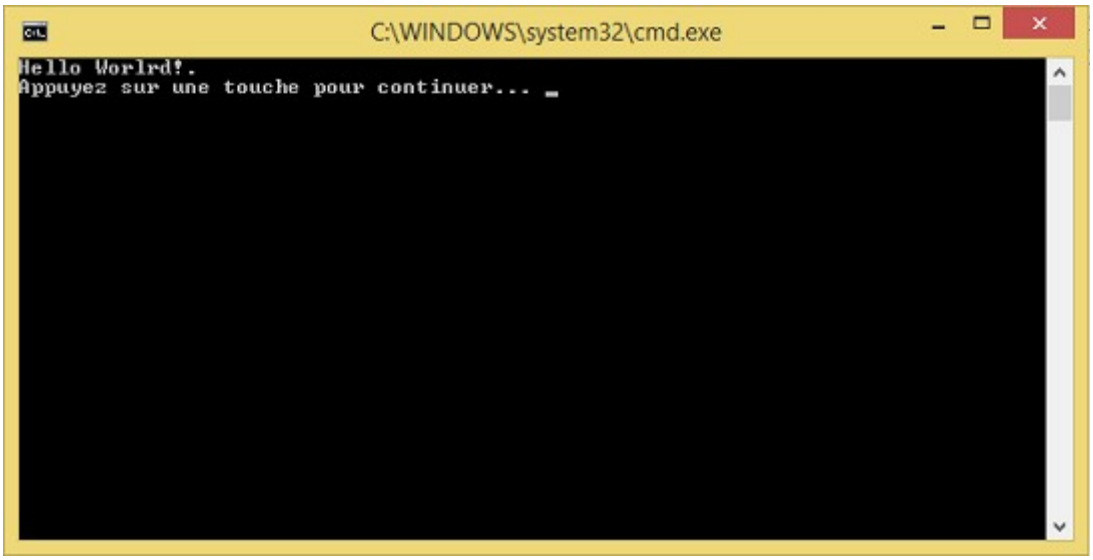
Pour interpréter ce script, il va falloir utiliser l'application php.exe qui se trouve dans le répertoire PHP en lui passant le script en paramètre. Pour que cela soit moins fastidieux, passez par un traitement par lot (fichier batch) comme dans l'exemple ci-desous :

```
@echo off
set file=test.php
set dir=C:\Program Files (x86)\PHP

"%dir%\php.exe" ".\%file%"

echo .
pause
```

1. créer un fichier .bat (test.bat par exemple)
2. copier les commandes ci-dessus
   - la variable file contient le nom du fichier php à interpréter
   - la variable dir contient le répertoire où se trouve les fichiers dézippés de l'archive
3. lancer l'exécution en cliquant sur le fichier batch



Une grande partie des fonctions PHP fonctionnent aussi bien en CLI qu'en CGI[2]. Toutefois, il existe quelques légères différences :

- si vous avez manipulé des headers dans un script pour un site web, ne comptez pas pouvoir faire de même avec PHP en mode CLI, pour des raisons évidentes ; aucun header n'est envoyé nulle part !
- Contrairement à la version CGI, la version CLI de PHP ne transforme pas le répertoire du script en répertoire courant. Si vous manipulez les fichiers de votre ordinateur, il faudra donc taper leur adresse en absolu, ou bien faire du répertoire du script le répertoire courant.
- Lorsqu'une erreur se produit avec un script exécuté en CGI, le message d'erreur est formaté en HTML avant d'être placé dans la source, afin de rendre l'erreur plus facilement

2 Common Gateway Interface : Interface de passerelle commune

lisible. Évidemment, en CLI, formater les erreurs en HTML n'aurait pas de sens, elles sont donc affichées en texte brut.

- Si vous utilisiez l'option implicit_flush dans le fichier php.ini de votre serveur web, sachez que cette option n'est pas activable en CLI.
- Un script exécuté en CGI a un temps d'exécution limité (souvent 30 secondes), afin de ne pas surcharger le serveur web. Cette limite n'ayant pas de sens en CLI, elle n'existe pas lorsque vous exécutez un script de cette manière.
- Comme dans des langages de programmation tels que le C, il peut être utile de pouvoir exécuter un script en mode CLI de plusieurs manières, $argv et $argc existent donc dans ce mode.
- Comme dans des langages de programmation tels que le C, des pointeurs de fichiers sont déjà ouverts dès l'exécution d'un script en CLI. Les constantes STDIN, STDOUT et STDERR peuvent donc être utilisées par des fonctions telles que fgets ou fputs, de la même manière qu'est utilisé un pointeur de fichier renvoyé par des fonctions comme fopen. Il est également inutile de les refermer à l'aide de fclose, ceci étant fait automatiquement à la fin du script.
- La console de Windows ne gère pas les accents ! Faites donc attention lorsque vous utilisez echo ou print...

3. Exécution en CLI

Les scripts en CLI ont souvent une utilité bien différente de celle des scripts en CGI. On les utilise parfois pour accomplir des tâches brèves, ou des tâches "automatisées". Un des principaux défauts des scripts en CLI est qu'on ne peut pas interagir directement avec l'utilisateur autrement qu'avec STDIN, qui se révèle limité.

Il existe des solutions face à ce problème, comme PHP-GTK qui permet de scripter de manière événementielle (le script est "en attente" d'un événement, provenant souvent de l'utilisateur) et de manier des widgets, mais elles sont souvent complexes.

Toutefois, le plus grand atout de PHP en CLI est l'absence de limite de temps d'exécution de la plupart des serveurs web.

Pour interagir de façon simple avec l'utilisateur en mode console, voici un exemple de code :

```
<?php
      $reponse = fgets(STDIN);
?>
```

Lorsque PHP atteindra cette ligne, le script se "bloquera". L'utilisateur pourra alors entrer du texte dans la console, et pourra l'envoyer dans $reponse à l'aide de la touche Entrée.

On peut utiliser PHP en CLI pour gérer des fichiers et / ou des dossiers. Les fonctions "Système de Fichiers" sont donc très utiles dans ce cas-là. Par exemple, si vous avez plusieurs fichiers à renommer, l'utilisation de PHP en CLI peut être judicieuse.

Pour renommer des fichiers .html en fichiers .php, qui seraient situés dans le dossier "fichiers" situé dans un dossier (contenant le script) à la racine de votre disque dur, il suffit de taper un script pour scanner le répertoire "fichiers", et renommer tous les fichiers .htm(l) s'y trouvant.

Exemple de script :

```php
<?php
  if (($array = scandir(getcwd()."/fichiers")) !== false )
    foreach($array as $key => $content)
      if ( $content != "." && $content != ".." && preg_match("`.html?$`",
$content) == 1 ) {
        $newname = preg_replace("`.html?$`", ".php", $content);

        rename(getcwd()."/fichiers/".$content, getcwd()."/fichiers/".$newname);

        print($content." renomme par ".$newname."\n");
        }
?>
```

Pour manipuler des pages web distantes (en lecture seule, bien sûr), quelque soit le mode de fonctionnement de PHP, l'une des meilleures solutions est d'utiliser cURL.

Si vous voulez utiliser toutes les fonctionnalités de PHP, comme les sockets, MySQL, cURL, etc, activez les extensions correspondantes en ôtant le point-virgule (";") devant leur DLL, dans php.ini. Pour trouver facilement la zone correspondante dans ce fichier, faite simplement une recherche sur "cURL" avec l'outil de recherche de l'éditeur de texte. Les DLL à activer devraient être à proximité de la première occurrence.

Si vous n'activez pas une fonctionnalité, les fonctions PHP correspondantes ne seront pas définies.

4. Les arguments

De la même façon qu'en langage C, on peut passer une liste d'arguments au script. Celui-ci peut ensuite les gérer.

Comme le fichier .bat est de loin le plus pratique pour lancer le script rapidement, je ne m'attarderai que sur lui.

Pour passer des arguments au script à partir du fichier .bat, il suffit de les ajouter à la suite de la ligne déjà remplie, en séparant chaque argument par un espace. Exemple :

```
@echo off
set file=test.php
set arg=un deux trois quatre
set dir=C:\Program Files (x86)\PHP

"%dir%\php.exe" ".\%file%" %arg%

echo .
pause
```

La variable de la ligne 3 contient les arguments passés au script PHP (ici 4).

Il faut ensuite gérer ces arguments côté script : les arguments sont stockés dans la variable superglobale, $_SERVER, plus précisément dans l'array $_SERVER['argv']. Le premier argument sera $_SERVER['argv'][0], le second sera $_SERVER['argv'][1], etc.

Le nombre d'arguments est stocké, dans $_SERVER['argc'].

NB : $_SERVER['argv'][0] contient le nom du script (ici test.php).