http://www.apachefriends.org/fr/xampp-windows.html

Table des matières

XAMPP est un kit d'installation d'Apache qui contient MySQL, PHP et Perl. XAMPP est réellement très facile à installer et à utiliser - vous n'avez qu'à le télécharger, le décompresser et le démarrer.

1.8.2-2 version (PHP 5.4 based)
- Updated PHP to 5.4.19 version for Windows, Linux and OS X

1.8.3-1 version (PHP 5.5 based)
- Updated PHP to 5.5.3 version for Windows, Linux and OS X

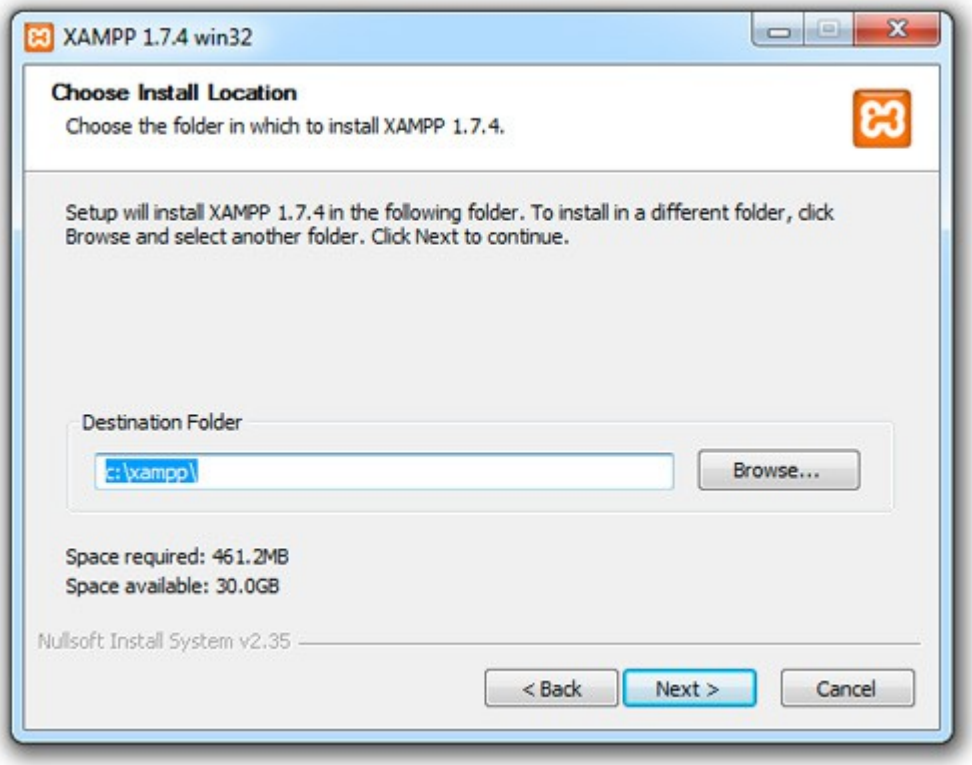
Télécharger simplement le kit de base de XAMPP. Les kits marqués "Ajouts" sont optionnels.

Apache 2.2.17, MySQL 5.5.8 + PBXT engine (currently disabled), PHP 5.3.5, OpenSSL 0.9.8l, phpMyAdmin 3.3.9, XAMPP Control Panel 2.5.8, Webalizer 2.21-02, Mercury Mail Transport System v4.72, FileZilla FTP Server 0.9.37, SQLite 2.8.17, SQLite 3.6.20, ADOdb

Table des matières

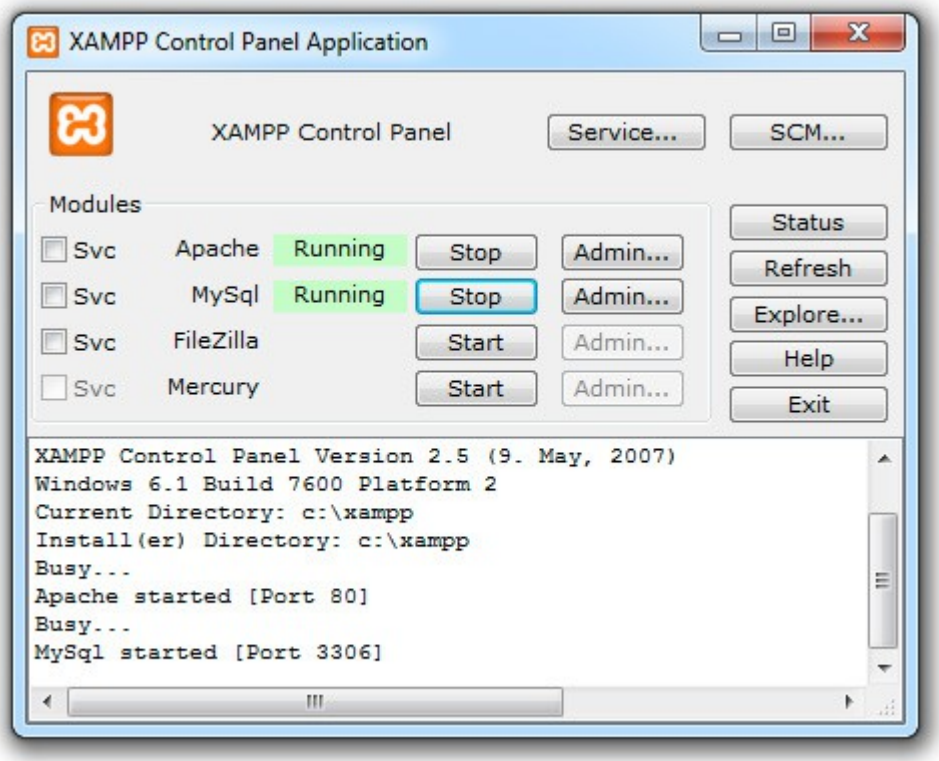
5.11, Xdebug 2.1.0rc1, Tomcat 7.0.3 (with mod_proxy_ajp as connector)
Pour Windows 2000, XP, Vista, 7.
Lisez-moi. Installateur

Installation



*L'assistant d'installation de XAMPP win32*Quand l'installation est complétée, vous trouverez XAMPP sous Démarrer / Programmes / XAMPP. Grâce au panneau de contrôle XAMPP, vous pouvez démarrer/arrêter chacun des serveurs et installer/désinstaller les services.

Table des matières



Le panneau de contrôle XAMPP pour arrêter/démarrer Apache, MySQL, FileZilla et Mercury ou installer ces serveurs en tant que services.

Présentation de HTTP

Voir aussi:http://doc.ubuntu-fr.org/projets/ecole/apache

- HTTP - HyperText Transfert Protocol - fondé par **Tim Berners** Lee, développé et utilisé par le WWW à partir de 1990.
- Protocole adapté au transfert d'information multimédia.
- Léger et rapide, à coût d'exploitation très bas.
- Introduit la notion d'hypertexte, c'est à dire que l'information de navigation est prise en compte dans le document, mais ne prend pas en charge le procédé complet de navigation.
- Le protocole HTTP sert à la communication entre le client et le serveur. Il s'agit en fait d'un dérivé du protocole FTP. Lors d'une communication, le logiciel client se connecte en TCP sur le serveur et télécharge en FTP le document désigné. Il coupe aussitôt la communication avec le serveur. S'il y a dans le document HTML plusieurs composants (comme des images, la plupart du temps), ce processus se répète autant de fois qu'il y a d'éléments constituant la page.
- L'avantage de ce processus est de limiter au maximum le temps d'occupation du serveur, de façon qu'il n'y ait pas d'engorgement de ce dernier.

Il est extrêmement simple d'étudier son fonctionnement en utilisant le programme telnet. Pour se connecter au serveur web il suffit de faire :

Table des matières

telnet adresse_ip_serveur_web 80

où *adresse_ip_serveur* est l'adresse ip du serveur web cible

Comment récupérer Telnet sur Windows VISTA et Windows Seven ?

Il suffit d'exécuter la commande :

pkgmgr /iu:TelnetClient

La partie principale de la requête en ce qui concerne le client est la commande, placée sur la première ligne de la reqûete. Elle détermine l'action à effectuer. Ces commandes sont parfois accompagnées d'arguments qui précisent l'objet sur lequel porte la commande. L'action la plus courament utilisée est la récupération d'un document en utilisant la commande GET ou POST (GET et POSt sont deux méthodes HTTP différentes pour effectuer la même action). Cette commande prend comme argument la ressource à récupérer, c'est à dire le chemin vers le fichier à récupérer (ce n'est pas un chemin absolu, mais relatif à la racine du serveur). Par exemple :

GET /index.html HTTP/1.0

les clients placent une entête dans leur requête, à la suite de la commande, décrivant par exemple qui ils sont (le nom et le numéro de version pour un navigateur par exemple) et ce qu'ils peuvent accepter (des images, du texte au format HTML pour le même type de logiciel par exemple). Certaines entêtes sont optionelles et d'autres obligatoires
pour utiliser ce type d'enttête, il faut mentionner à la suite de la requête la version du protocole utilisée. (en général HTTP 1.0 ou HTTP/1.0 voire HTTP 1.1 ou HTTP/1.1)

Table des matières

Par exemple :

```
telnet 127.0.0.1 80
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
GET /test.html HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 17 Apr 2008 07:39:21 GMT
Server: Apache/2.2.3 (Debian) mod_jk/1.2.18 mod_python/3.2.10 Python/2.4.4
PHP/5.2.0-8+etch10 mod_ssl/2.2.3 OpenSSL/0.9.8c mod_perl/2.0.2 Perl/v5.8.8
Last-Modified: Thu, 17 Apr 2008 07:16:12 GMT
ETag: "160488-9e-6883c700"
Accept-Ranges: bytes
Content-Length: 158
Connection: close
Content-Type: text/html

<html>

<head>
  <title>tester</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<h1 >tester</h1>

</body>
</html>
Connection closed by foreign host.
```

La requête utilisée dans cet exemple est la plus simple que l'on puisse trouver, elle se compose d'un seule ligne qui comprend trois éléments : la méthode, l'url (elle identifie la ressource, dans la plupart des cas sur Internet il s'agit d'un simple fichier texte ou d'une image) et la version du protocole HTTP utilisé (HTTP/1.0 ou HTTP/1.1).

En plus de cette ligne on peut trouver un certain nombre de champ (1 par ligne) dont la forme est toujours la même, le nom du champ, suivi de : et d'un espace et la valeur que l'on veut lui donner (toujours suivi des caractères \r et \n). Les caractères \r et \n correspondent respectivement au retour chariot et saut de ligne. Vient ensuite une ligne vide, composée donc seulement des deux caractères \r et \n et le corps de la requête. Une requête a donc la forme suivante :

Méthode url HTTP/1.0\r\n
Champ1 : valeur 1\r\n
Champ2 : valeur 2\r\n
\r\n
Ceci est le corps de ma requête ...
Autre exemple:

```
telnet test.skateinmars.net 80
```

Vous verrez s'afficher :

```
Trying 82.228.105.196...
Connected to skateinmars.net.
```

Table des matières

Vous êtes donc connecté au serveur identifié par le nom `skateinmars.net` et l'adresse IP 82.228.105.196. Le serveur attend une action de votre part. Saisissez :

```
GET / HTTP/1.1
Host: test.skateinmars.net
```

Puis 2 fois <entrée> Vous voyez alors s'afficher du texte HTML.

```
mic@bureau:~$ telnet test.skateinmars.net 80
Trying 88.182.115.250...
Connected to skateinmars.net.
Escape character is '^]'.
GET / HTTP/1.1
Host: test.skateinmars.net


HTTP/1.1 200 OK
Server: nginx/0.6.21
Date: Tue, 02 Feb 2010 08:30:28 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Set-Cookie:
SESS70816c10121542b97e1f8307d31ccd47=f435a23d4ebc0831ae5232be29202ae1;
expires=Thu, 25 Feb 2010 12:03:48 GMT; path=/; domain=.local.ubuntu-fr.org
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Last-Modified: Tue, 02 Feb 2010 08:30:28 GMT
Cache-Control: store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Content-Length: 7965

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr" dir="ltr">

<head>
  <title>Ubuntu-fr | Communauté francophone des utilisateurs d'Ubuntu</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="alternate" type="application/rss+xml" title="Ubuntu-fr RSS" href="http://local.ubuntu-
fr.org/rss.xml" />

<link rel="shortcut icon" href="/misc/favicon.ico" type="image/x-icon" />
  <link type="text/css" rel="stylesheet" media="all" href="/modules/aggregator/aggregator.css?F" />
<link type="text/css" rel="stylesheet" media="all" href="/modules/node/node.css?F" />
<link type="text/css" rel="stylesheet" media="all" href="/modules/system/defaults.css?F" />
<link type="text/css" rel="stylesheet" media="all" href="/modules/system/system.css?F" />
<link type="text/css" rel="stylesheet" media="all" href="/modules/system/system-menus.css?F" />
<link type="text/css" rel="stylesheet" media="all" href="/modules/user/user.css?F" />
    <script type="text/javascript"> </script>
</head>
<body class="front not-logged-in page-node two-sidebars">
  <div id="page">
--------- on coupé ------
  </div> <!-- /page -->

</body>
</html>
```

HTTP/**1.1 200 OK**

Ici on peut voir le protocole utilisé, et le code de retour. Le code de retour est très important, il nous permet de savoir si la requête a reussi ou non. Ici le code est 200, soit OK. Le code 200, vous le recevez à peu prêt tout le temps, il signifie que la requête a réussi.

Les deux méthodes vraiment utilisées sont GET et POST (les amateurs d'html

Table des matières

reconnaîtront des mots clefs familiers à la construction d'un formulaire). La méthode GET est la plus simple car le corps du message dans ce type de requête est vide. La méthode POST permet d'envoyer des informations au serveur dans le corps du message d'une requête HTTP. Lorsque des informations sont envoyées au serveur à l'aide de la méthode GET, elles sont encodées à la suite de la ressource après le symbole '?' dans l'url.

Qu'est-ce qu'un type MIME

Lorsqu'un utilisateur demande une page web au serveur Apache, il arrive que cette page web fasse appel à des fichiers complémentaires : images, sons…
Tous ces fichiers complémentaires ont un format particulier et nécessitent donc des logiciels particuliers pour être affichés.

Afin de permettre au client de savoir de quel type de logiciel il a besoin en fonction du type de données qu'Apache lui envoie, le serveur web associe un type **MIME** (Multipurpose Internet Mail Extensions). Cette information a été créé pour permettre l'envoi de pièces jointes avec un courrier électronique et elle est utilisé également pour le protocole http.

Réglages d'exécution

cachée dans xampp\apache\conf\extra\httpd-default.conf

- **Timeout** 300
  Paramètre important qui fixe le temps (en ms) d'attente maximum du serveur d'une réponse à une requete envoyée à un programme extérieur (comme un gestionnaire de base de données)
- **KeepAlive**on
  MaxKeepAliverequests 100
  KeepAliveTimeout 15
  Autorise les connexions persistantes d'un client, afin de lui permettre l'envoi de plusieurs requetes sans déconnexion, avec un plafond fixé pour un client, pour servir aussi d'éventuels autres clients ! et un temps d'attente maxi de la requete suivante provenant du meme client.
- **MinSpareServers 4 MaxSpareServers** 20
  Nombres maximum et minimum de processus serveurs devant être en permanence disponibles, en attente de nouvelles connexions clientes
- **StartServers** 5
  Nombre de processus serveurs démarrés à l'initialisation, en plus du processus père.
  ps aux|*grep apache* donne le nombre de PID enfant propriété de *www-data*.
- **MaxClients** 20
  Nombre maximum de processus qu'Apache peut lancer et gérer simultanément. Ce nombre ne peut pas excéder 254
- **MaxRequestsPerChild** 500
  Nombre maximum de requetes HTTP traitées par un processus enfant avant qu'il ne soit éliminé.

Configuration

dans xampp\apache\conf\httpd.conf

Table des matières

- **ServerRoot "repertoire/de/configuation/apache"**
- **ServerRoot "C:/xampp/apache"**
  Il s'agit du répertoire où le serveur trouvera son répertoire de configuration
- **ServerName** www.monsite.be
  Le nom doit correspondre à une adresse IP, donc être renseigné dans un serveur DNS (car la machine hôte est jointe par son adresse IP)
  Si aucun nom n'est spécifié, alors le serveur tente de déduire un nom en procédant à un "lookup inverse" à partir de l'adresse IP.

- **DocumentRoot** DocumentRoot "C:/xampp/htdocs"
- **ServerRoot** "C:/xampp/apache"
  fixe la racine du serveur Web, c'est-à-dire le répertoire de base où sont cherchées par défaut les pages html, lorsque l'URL se limite au nom du serveur et ne comporte pas de chemin de répertoire.
- **Listen**
  Cette directive est (pour Apache2) indispensable. Elle spécifie les adresses IP des interfaces locales et les ports sur lesquels Apache doit être en écoute (par défaut les requêtes sont acceptées de toutes les interfaces IP, et donc en général seul(s) le(s) numéros de port(s) sont renseignés).
  Pour que le serveur accepte des connexions à la fois sur les ports 80 et 8080

```
Listen 80
Listen 8080
```

  Ou si vous ne permettez l'accès qu'à deux couples d'adresses-ports, alors :

  - Listen 192.70.2.1 :80
  - Listen 192.70.2.2 :8000

- **Directive ServerTokens (full par défaut)**
  cachée dans xampp\apache\conf\extra\httpd-default.conf
  - Description Contrôle le contenu de l'en-tête de réponse.
  - Syntaxe ServerTokens
  - Major|Minor|Min[imal]|Prod[uctOnly]|OS|Full
  - Exemple ServerTokens **Full** (valeur par défaut)

- **User** daemon
- **Group** daemon

Pour xamp

  Apache doit être démarré par l'administrateur mais par sécurité ses processus auront pour propriétaire l'utilisateur *(daemon)*, sans privilège.
- **ServerAdmin** webmaster@localhost
  S'il a un problème, le serveur écrit un message à cette adresse
- **AccessFileName** .htaccess
  Cette clause fixe le nom du fichier à trouver dans un répertoire pour que son accès soit protégé, en imposant à l'utilisateur une authentification par nom et mot de passe. Ces comptes sont spécifiques à Apache et n'interfèrent pas avec les comptes Linux.
- **DirectoryIndex** index.html index.php index.htm ...
  Il est courant d'omettre le nom du fichier de la page d'accueil d'un site ou de l'un de

Table des matières

ses sous-répertoires. Pour ne pas retourner systématiquement une **erreur 404** signalant une adresse erronnée, le serveur posséde une liste standard de noms de fichiers qu'il s'efforce de trouver dans le répertoire.
Cette liste ordonnée est indiquée par la clause `DirectoryIndex`

Gestion des droits

Nous présenterons ici les mesures préventives liées aux fichiers contenus dans l'arborescence du serveur web.

Directory, Files, Location

La gestion des accès est effectuée par le module `mod_access`. On manipule principalement trois catégories d'objets :

- `Directory` désigne un répertoire du serveur ;
- `Location` une arborescence du serveur web ;
- `Files` un fichier.

Mesure défensive

Il est fortement conseillé de tout interdire par défaut :

```
<Directory />
Order deny,allow
Deny from all
</Directory>
```

Ensuite, il ne reste qu'à valider l'accès aux répertoires correspondant aux sites

`Order` indique dans quel ordre les directives `deny` et `allow` sont évaluées. `Deny from all` interdit l'accès depuis partout. On aurait pu indiquer un nom de machine, un nom de domaine, une adresse IP, un couple IP/masque de réseau.

Options, AllowOverride

`Options` contrôle

- le suivi des liens symboliques `FollowSymLinks`/`SymLinksIfOwnerMatch` ;
- l'exécution des scripts CGI `ExecCGI` ;
- les Server Side Includes `Includes` et `IncludesNOEXEC` ;
- la génération de pages d'index `Indexes` en l'absence de celles-ci ;
- ainsi que l'orientation multilingue `MultiViews`.

`All` regroupe les différentes options sauf `MultiViews`, `None` supprime les options.

`MultiViews` redirige une demande pour `index.html` vers `index.html.en` ou `index.html.fr` selon la préférence signalée par le navigateur au serveur web.

Il est important d'être le plus restrictif possible par défaut, je conseille de n'autoriser que le suivi des liens symboliques où liens et destinations ont le même propriétaire :

```
<Directory />
    Options SymLinksIfOwnerMatch
    AllowOverride None
</Directory />
```

Table des matières

Un pirate pouvant écrire dans un répertoire du serveur web, par exemple via un partage NFS, peut en profiter pour accéder au fichier `/etc/passwd` via un lien symbolique si l'option `FollowSymLinks` est présente, `Includes` ou `ExecCGI` permet d'exécuter des programmes... En un mot, soyez prudent.

La directive `AllowOverride` peut prendre n'importe quel paramètre qu'aurait pris `Options`.

Protection par mot de passe

Le module `mod_auth` permet de protéger l'accès à un répertoire par mot de passe. En pratique, c'est souvent utiliser pour filtrer les accès à un sous-répertoires d'une page personnelle.

```
<Directory /home/*/public_html>
    AllowOverride AuthConfig
    Options SymLinksIfOwnerMatch
</Directory>
```

ou pour bloquer l'accès à un répertoire déterminé

```
<Location "/private">
Options None
AllowOverride None
AuthName "restricted stuff"
AuthType Basic
AuthUserFile "/etc/httpd/.passwd"
require valid-user
</Location>
```

Dans le cas des pages personnelles `/home/*/public_html` des utilisateurs, l'accès est déterminé par un fichier `.htaccess` que peut utiliser ou non un utilisateur, le nom du fichier même est défini dans la section générale de la configuration du serveur. Ce fichier protége l'accès au répertoire dans lequel il est placé ainsi que l'accès aux sous-répertoires. `AllowOverride AuthConfig` permet à ce fichier d'être pris en compte. Par précaution, il faut empêcher un utilisateur de les récupérer via le web :

```
AccessFileName .htaccess
<Files ~ "^\.ht">
    Order deny,allow
    Deny from all
</Files>
```

Si on veut pouvoir définir explicitement des exceptions pour les fichiers `.htpipo` par exemple, il faut spécifier l'ordre `allow` puis `deny` pour que l'autorisation prime sur l'interdiction.

Le fichier `.htaccess` contient les mêmes champs que pour le répertoire `/private` de l'exemple, c'est-à-dire un nom qui apparaîtra sur la fenêtre de demande d'identification (`AuthName`), la méthode d'identification (`AuthType`), le fichier de mot de passe (`AuthUserFile`) et enfin `require valid-user` :

```
AuthName "my restricted stuff"
AuthType Basic
AuthUserFile "/home/titi/.htpasswd"
require valid-user
```

Table des matières

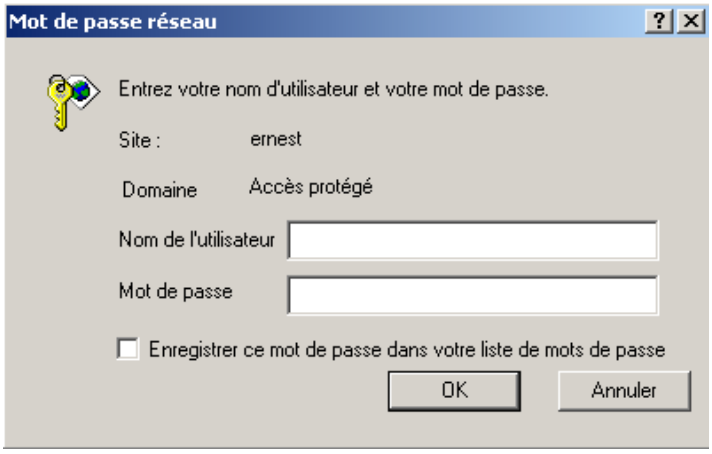
Accès sécurisé .htaccess

Remarque pour xampp

Dans httpd.conf vérifiez

#LoadModule rewrite_module modules/mod_rewrite.so

et enlevez le #



Doc tirée de http://ernest.cheska.net/htaccess/htaccess.shtml

Il existe beaucoup de méthodes pour securiser une partie d'un site web tournant sur Apache. La methode de l'htaccess est tres souplemais ce n'est pas forcement la meilleure.
La méthode Basic
Tiré de http://www.ac-creteil.fr/reseaux/systemes/linux/lamp/apache2-authentification.html

- Directives Voici les directives usuelles et leur signification

Directive	Action
`AuthType basic`	type d'authentification communément adopté (fait circuler les mots de passe en clair)
`AuthName texte`	affichera ce texte comme invite dans une boite de dialogue
`AuthUserFile chemin/fichier`	précise le fichier qui contient les comptes et mots de passe des utilisateurs ayant droit d'accès
`Require valid-user` `Require liste-noms`	l'accès s'applique à tous les comptes du fichiers, ou seulement aux comptes énumérés dans la liste

Nous allons ajouter des directives pour le dossier que nous souhaitons proteger comme suit:

```
<Directory /var/www/html/>
AllowOverride AuthConfig
</Directory>
```

Table des matières

Ici je souhaite que le dossier **/var/www/html** accepte la directive AllowOverride AuthConfig. Cela implique que **tous ces sous-dossiers vont également accepter cette directive**. Que signifie AllowOverride AuthConfig ? Tout simplement qu'Apache daignera lire votre fichier .htaccess si il se trouve dans le repertoire en question (ou un de ses sous-repertoires)

Apache2 modification ?
les fichiers sont cachés par Apache 2 par un .htaccess et finalement, j'ai pas mis le .htaccess j'ai configuré directement dans default-000 comme ça pas de risque et mis le .htpasswd bien ailleurs.

Dans default-000

Code:

```
<Directory /var/www/XXX>
    AuthName "Accés Password"
AuthUserFile "/XXX/.htpasswd"
AuthType Basic
AllowOverride All
<limit GET POST>
    require valid-user
</Limit>
</Directory>
```

Un autre exemple:
Exemple de procédure Supposons que l'espace privé soit situé dans le répertoire `/var/www/prive` et son accès réservé à un ensemble d'utilisateurs : `admin, webmaster et toto`
Directives dans le fichier *default*

```
<Directory "/var/www/prive">
AuthType Basic
AuthUserFile /etc/apache/users
AuthName "Accès privé"
# autres clauses
# AuthGroupFile /etc/apache/groups

<limit GET>
# ATTENTION : GET en majuscules !
require valid-user
# require user toto dupond
# require group profs
</limit>
</Directory>
```

Table des matières

La procédure de création du fichier .htaccess :

Cela se fait tres simplement avec un editeur et vous le placez dans le repertoire que vous souhaitez proteger. Voici ce que vous mettez dedans:
Lorsque vous souhaitez créer une zone d'identification sur votre site web via les fichiers **.htaccess** et **.htpasswd**, le contenu d'un fichier htaccess ressemble à ceci :

AuthUserFile /repertoire/mesmotsdepasse
AuthGroupFile /dev/null
AuthName "Accès protégé"
AuthType Basic
<limit GET POST>
require valid-user
</Limit>

Attention au chemin sur la première ligne.

Ce chemin indique l'endroit où vous allez caser votre fichier .passwd (ce fichier contiendra les logins et passwords autorisés à voir les pages). On conseille generalement de le placer hors du site web en lui meme.

l'ajout ou le retrait d'un fichier `.htacess` de l'arborescence est pris en compte dynamiquement : à chaque nouvel accès au répertoire, Apache relit le contenu du répertoire.
En cas de problèmes, si la mise en place du fichier `.htaccess` ne change rien : le répertoire est peut être sous le contrôle du directive `AllowOverride None`. Pour vérifier, vous pouvez intentionnellement ajouter une ligne erronnée dans le `.htaccess` et recharger la page. Si vous n'obtenez par la page "Internal Server Error", c'est que la cause du problème réside bien dans la directive `AllowOverride`.

La procédure de création du fichier .htpasswd :

Lorsque vous désirez ajouter ou créer un utilisateur, vous devez savoir crypter le mot de passe que vous désirez lui attribuer puisque mettre un mot de passe en clair dans le fichier ne fonctionnera pas.
Il faut placer ce fichier là ou vous avez indiqué à .htaccess de le chercher. Vous le créez tout simplement avec un editeur de texte. Il contient une ligne par utilisateur autorisé qui se décompose comme suit:

login:password
login2:password2

Ces mots de passe doivent être cryptés.

Bien sur dans cet exemple on ne voit pas les mots de passe mais j'ai bien tapé supertoto deux fois de suite. Il y a maintenant un fichier .htpasswd dans /var/www avec le user toto et son mot de passe crypté.
Rappelons que ce fichier de configuration ne supplante (override) les clauses placées dans le fichier *apache2.conf*, que si la clause suivante est présente dans le paragraphe gérant ce même répertoire:

Table des matières

```
<Directory /var/www/prive>
...........
Allowoverride All
</Directory>
```

- Remarques - l'expérience montre qu'il est parfois indispensable de redémarrer le navigateur client pour que les modifications soient prises en compte.
  - Pour empêcher l'affichage du contenu de ce fichier *.htaccess*, bien vérifier la présence dans httpd.conf de la directive :

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

Crypter les mots de passe

Dans la console de xampp

cd C:\xampplite\apache\bin

htpasswd -n toto

Automatically using MD5 format.

New password: ******

Re-type new password: ******

toto:$apr1$Krg9oGD4$MDIGIPDTIDNjJYZ8ycWV5/

et recopiez cette ligne

Creer le mot de passe depuis

http://www.htaccesstools.com/htpasswd-generator/

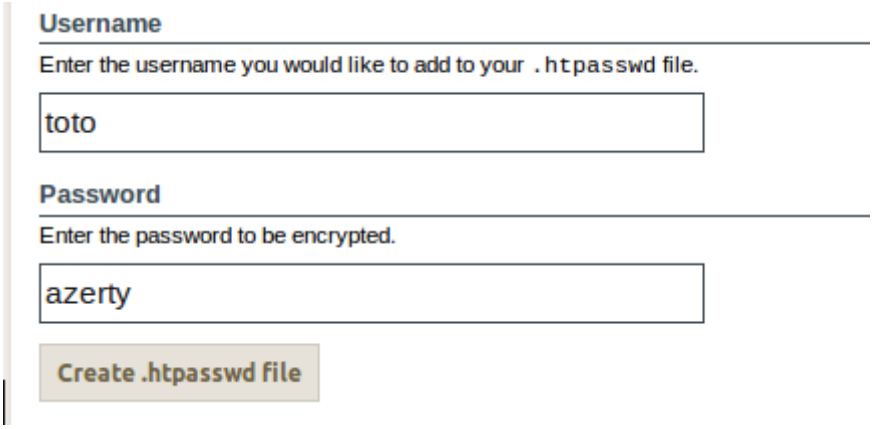
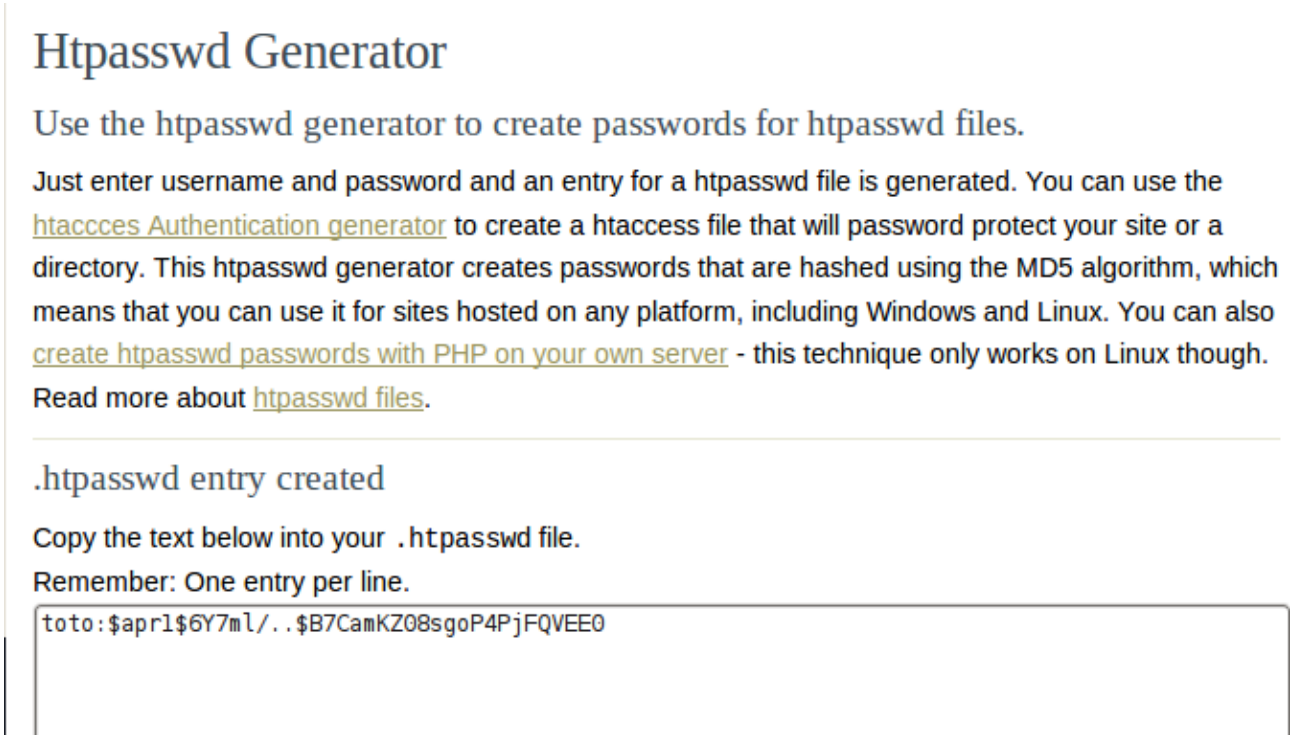


Table des matières



Fonction PHP : **crypt.**

```
Vous lui donnez un mot de passe et, ne cherchez pas à savoir comment, ça vous le crypte
Par exemple, si mon mot de passe est "kangourou", voici le code PHP que je devrai écrire pour l'obtenir en version cryptée :
<?php echo crypt('kangourou'); ?>
```

Créer le .htaccess

voir :
http://www.siteduzero.com/tuto-3-152-1-proteger-un-dossier-avec-un-htaccess.html

La première étape est de créer sur votre disque dur un fichier appelé ".htaccess". Mais là, vous allez certainement avoir un problème.
En effet, Windows n'aime pas les fichiers qui commencent par un point. Pour tous les autres systèmes d'exploitation (Mac OS, Linux) vous n'aurez aucun problème. Mais Windows lui il veut pas, allez savoir pourquoi !

On va utiliser une astuce : on va dans un premier temps créer un fichier appelé htaccess.txt, et plus tard avec notre logiciel FTP on le renommera en .htaccess (et là ça marchera !).

autre possibilité : une ligne de php **rename()**

```
<?php
rename("htaccess", ".htaccess");
?>
```

Table des matières

Commencez donc par ouvrir Bloc-Notes par exemple :

Là dedans, on va rentrer des informations qui n'ont rien à voir avec du HTML ou du PHP : ce sont des instructions pour le serveur. Elles vont expliquer au serveur que seules certaines personnes sont autorisées à accéder au dossier. Copiez-y ce code :

Code : Apache
```
AuthName "Page d'administration protégée"
AuthType Basic
AuthUserFile "/home/sdz/www/gestion/admin/.htpasswd"
Require valid-user
```

Parmi ces 4 lignes, il y en a 2 que vous allez devoir changer :

- AuthName : c'est le texte qui invitera l'utilisateur à inscrire son login / mot de passe. Vous pouvez personnaliser ce texte comme bon vous semble.
- AuthUserFile : là c'est plus délicat, c'est le chemin absolu vers le fichier .htpasswd (que vous mettrez dans le même répertoire que le .htaccess).

Comment trouver ce chemin absolu ?

En effet, c'est la plupart du temps délicat à trouver. Heureusement, il existe une fonction PHP qui va beaucoup nous aider : **realpath().**
Cette fonction donne le chemin absolu vers le fichier que vous indiquez. Vous allez donc faire comme ceci pour trouver le chemin absolu :

1. Créez un fichier appelé "chemin.php".
2. Mettez juste cette ligne de code dedans :
```
<? echo realpath('chemin.php'); ?>
```

Table des matières

3. Envoyez ce fichier sur votre serveur avec votre logiciel FTP. Placez-le dans le dossier que vous voulez protéger.
4. Ouvrez votre navigateur et allez voir ce fichier PHP. Il vous donne le chemin absolu, par exemple dans mon cas : /home/sdz/www/gestion/admin/chemin.php
5. Copiez ce chemin dans votre .htaccess, et remplacez le "chemin.php" par ".htpasswd", ce qui nous donne au final par exemple : /home/sdz/www/gestion/admin/.htpasswd
6. Supprimez le fichier "chemin.php" de votre serveur, il ne nous sert plus à rien maintenant qu'il nous a donné le chemin absolu :)

La ligne AuthUserFile indique donc où se trouve le fichier .htpasswd qui contient les mots de passe.

Enregistrez le fichier avec le nom "htaccess.txt" pour le moment, on le renommera en ".htaccess" plus tard.
Voilà, on a fini de créer le .htaccess, on peut maintenant passer au .htpasswd

---

Créer le .htpasswd

Créez maintenant un nouveau fichier avec Bloc-Notes.

Le .htpasswd contient la liste des personnes autorisées à accéder aux pages du dossier. On met une personne par ligne, sous cette forme :

Code:

utilisateur1:motdepasse1
utilisateur2:motdepasse2

Au final, votre fichier .htpasswd devrait ressembler à ceci :

Code : Apache
mateo21:1MEqT//cb$hAVid.qmmSGFW/wDIIfQ81
darkeden:1/lgP8dYa$sQNXcCP47KhP1sneRIZoO0
IAN:1lT7nqnsg$cVtoPfe0IgrjES7Ushmoy.
Leon:1h4oVHp3O$X7Ejpn.uuOhJRkT3qnw3i0

Dans cet exemple, il y a 4 personnes autorisées à accéder au dossier : ce sont mateo21, darkeden, IAN, et Leon.

avec XAMPP

voila comment obtenir le bon mot de passe:
ouvrir le terminal et taper :
htpasswd -n votreLogin
Le programme vous demande de taper votre mot de passe donc tapez le
ensuite il faut le confirmer donc retapez le
et une fois la confirmation effectuée vous allez voir apparaitre un ligne de ce type :

Table des matières

votreLogin:L072M07efFRzg
Et voilà vous n'avez plus qu'à copier cette ligne dans votre .htpasswd

Table des matières

Remarque :

J'ai constaté que le même mot de passe peut générer des cles différentes !!!

toto:$apr1$FEwpYbMw$tjc2sIsGX9HSttuNVfvf11

xx:$apr1$jLPXrxe/$idEsL3dVpOWDkv4egvL7h.

tata:$apr1$tfeeI18o$RtBQ5hvx0bQOgnuPhuZtO0

Ces 3 utilisateurs ont le mot de passe toto !! et cela fonctionne

test du générateur avec le mot de passe toto

htpasswd -n toto

New password:

Re-type new password:

toto:**$apr1$18eQSpdf$P4e21U5Zpub37VJbAYy0A0**

htpasswd -n toto

New password:

Re-type new password:

toto:**$apr1$FEwpYbMw$tjc2sIsGX9HSttuNVfvf11**

OVH - Cryptage - Fichier .htpasswd

Voir : http://www.wordetweb.com/word-et-web/OVH-cryptage-mot-passe-dans-htaccess-htpasswd-FR.htm

Outil de cryptage du mot de passe *très malexpliqué !* :

http://www.ovh.com/fr/support/outils/crypt_password.pl

Exemple de cryptage :

Contenu du fichier .htpasswd : michel:zzA/GxehCIrWk

Accès au dossier protégé :

Table des matières

Login : michel

Mot de passe : admin

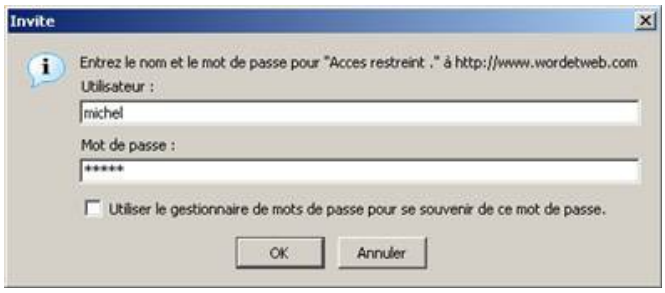


Table des matières

Utilisation d'alias de répertoires

Il peut être utile de remplacer un chemin de répertoires par un nom symbolique. Ces répertoires alias peuvent être paramétrés comme les autres.
Exemple significatif :
Il s'agit d'**accéder par l'alias doc aux doc HTML du serveur** Linux et de ses différentes applications et services installés, qui sont regroupées dans **/usr/share/doc**.
On réserve cette consultation aux machines du réseau local.
La stratégie consiste ici à interdire d'abord à tous (deny from all), puis on énumère les exceptions (allow from ..,..)

```
# pour accéder à la doc directement avec l'url http://Serveur/doc
Alias /doc /usr/share/doc
```

autorisations:

```
<Directory /usr/share/doc>
order deny,allow
deny from all
# permission à partir de localhost
allow from localhost, 127.0.0.1
# permission à partir des stations du sous-domaine de l'établissement
allow from .ipeps.be
        Options Indexes FollowSymLinks
</Directory>
```

exemple dans xamp

```
Alias /site "C:/Users/Michel/Documents/site"
<Directory "C:/Users/Michel/Documents/site">
  Options Indexes FollowSymLinks ExecCGI
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>
```

Table des matières