Trusty, BROUILLON

Configuration de Apache 2

Directive AllowOverride

Active les fichiers .htaccess

Syntaxe :

```
AllowOverride All|None|type directive [type directive] ...
```

Défaut :

```
AllowOverride All
```

Ne peut être utilisée que dans les sections <Directory> définies sans expressions rationnelles.

?AllowOverride All::active les fichiers .htaccess!!
?AllowOverride None::les fichiers .htaccess sont totalement ignorés!!

Directive <Directory>

Syntaxe :

```
<Directory chemin répertoire>
 ...
</Directory>
```

Les balises **<Directory>** et **</Directory>** permettent de regrouper un ensemble de directives qui ne s'appliquent qu'au répertoire précisé, à ses sous-répertoires, et aux fichiers situés dans ces sous-répertoires.

Toute directive autorisée dans un contexte de répertoire peut être utilisée.

?chemin répertoire::chemin absolu d'un répertoire, éventuellement avec des jokers :

? ?::correspond à un caractère quelconque!!
? *::correspond à toute chaîne de caractères.
? []::Les intervalles de caractères sont autorisés.
Aucun caractère générique ne peut remplacer le caractère **/**, si bien que :

- l'expression <Directory /*/public_html> ne conviendra pas pour le chemin * /home/user/public_html,
- alors que <Directory /home/*/public_html> conviendra.

Exemple :

```
<Directory /usr/local/httpd/htdocs>
Options Indexes FollowSymLinks
</Directory>
```

!!

?expressions rationnelles::peuvent aussi être utilisées en ajoutant le caractère ~.
Par exemple :

```
<Directory ~ "^/www/[0-9]{3}">
```

pourra correspondre à tout répertoire situé dans /www/ et dont le nom se compose de trois chiffres.!!

Notez que pour Apache, la politique d'accès par défaut dans les sections <Directory /> est Allow from All. Ceci signifie qu'Apache va servir tout fichier correspondant à une URL. Il est recommandé de modifier cette situation à l'aide d'un bloc du style

```
<Directory />
Order Deny,Allow
Deny from All
</Directory>
```

puis d'affiner la configuration pour les répertoires que vous voulez rendre accessibles.

Directive DocumentRoot

?DocumentRoot chemin_répertoire::Définit le répertoire à partir duquel httpd va servir les fichiers.!!

?chemin_répertoire::ne doit pas comporter de slash terminal ::Le chemin de l'URL sera ajouté par le serveur à la racine des documents. ::Si chemin_répertoire n'est pas un chemin absolu, il est considéré comme relatif au chemin défini par la directive ServerRoot.!!

Par exemple, avec :

```
DocumentRoot /usr/web
```

un accès à **http://www.my.host.com/index.html** se réfère à **/usr/web/index.html**.

Directive Include

Inclut d'autres fichiers de configuration dans un des fichiers de configuration du serveur

Syntaxe

```
Include chemin fichier|chemin répertoire
```

On peut utiliser des jokers dans le nom du fichier ou la partie répertoire pour inclure plusieurs fichiers en une seule fois.

Eviter de pointer vers un répertoire. Utiliser plutôt la syntaxe avec caractères génériques vue ci-dessous pour inclure des fichiers dont le nom correspond à un modèle particulier, comme *.conf par exemple.

Le chemin fichier spécifié peut être soit absolu, soit relatif au répertoire défini par la directive ServerRoot.

Exemples :

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

ou encore, avec des chemins relatifs au répertoire défini par la directive ServerRoot :

```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

Directive Options

La directive Options permet de définir les fonctionnalités de serveur disponibles pour un répertoire particulier.

?All::Toutes les options exceptée MultiViews. (**configuration par défaut**)!!
?None::aucune fonctionnalité spécifique n'est activée!!
?FollowSymLinks::Le serveur va suivre les liens symboliques dans le répertoire concerné.!!
?Indexes::Génère automatiquement un index si aucun n'est défini pour ce répertoire.!!
?MultiViews::vues multiples autorisées.!!
?ExecCGI::L'exécution de scripts CGI à l'aide du module mod_cgi est permise.!!
?Includes::Les inclusions côté serveur (SSI) à l'aide du module mod_include sont autorisées.!!
?IncludesNOEXEC::Les inclusions côté serveur (SSI) sont permises, mais #exec cmd et #exec cgi sont désactivées.!!
?SymLinksIfOwnerMatch::Le serveur ne suivra que les liens symboliques qui renvoient vers un fichier ou un répertoire dont le propriétaire est le même que celui du lien.!!

Les options FollowSymLinks et SymLinksIfOwnerMatch ne fonctionnent que dans les sections <Directory> ou les fichiers .htaccess.

On peut ajouter ou retrancher des options à celles déjà existantes en les faisant précéder de + ou -

Par exemple :

```
<Directory /web/docs>
Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
Options +Includes -Indexes
</Directory>
```

Dans ce cas,, les options FollowSymLinks et Includes seront prises en compte pour le répertoire /web/docs/spec.

Directive Require

Cette option remplace les anciennes options *order* et *allow*!!

Require [all|local]

?Require all granted::Accès autorisé dans tous les cas!!
?Require all denied::Accès refusé dans tous les cas!!
?Require local::Ne permet l'accès au serveur qu'aux clients locaux, c'est-à-dire que :

- l'adresse IP du client correspond à 127.0.0.0/8
- ou à ::1
- ou les adresses IP du client et du serveur sont identiques!!

Require [host|ip]

Contrôle l'accès au serveur en fonction du nom d'hôte ou de l'adresse IP du client distant.

?Require host <nom d'hôte>::Accès autorisé pour ce nom d'hôte

- exemple : Require host example.org!!

?Require host <nom de domaine>Accès autorisé pour ce domaine (éventuellement partiel)

- exemples :
  - Require host .net
  - Require host .net example.edu!!

?Require ip <ip complète>::Accès autorisé pour cette adresse

- exemple : Require ip 10 172.20 192.168.2!!

?Require ip <ip partielle>::Accès autorisé pour cette plage d'adresses!!
?Require ip <paire réseau/masque de sous-réseau>::Accès autorisé pour ce réseau!!

Require [user|group|valid-user]

?Require user identifiant utilisateur [identifiant utilisateur] ...::Seuls les utilisateurs spécifiés sont autorisés!!
?Require group nom groupe [nom groupe] ...::Seuls les utilisateurs appartenant aux groupes spécifiés sont autorisés!!
?Require valid-user::Tous les utilisateurs valides sont autorisés!!

Directive ServerAdmin

Adresse électronique que le serveur inclut dans les messages d'erreur envoyés au client

Syntaxe :

```
ServerAdmin adresse_électronique|URL
```

Eviter d'utiliser une URL, préférer une adresse mail.

Directive ServerAlias

Autres noms d'un serveur utilisables pour atteindre des serveurs virtuels à base de nom

Syntaxe :

```
ServerAlias nom serveur [nom serveur] ...
```

La directive ServerAlias peut contenir des caractères génériques, si nécessaire.

```
<VirtualHost *:80>
ServerName serveur.domaine.com
ServerAlias serveur serveur2.domaine.com serveur2
ServerAlias *.example.com
```

```
UseCanonicalName Off
# ...
</VirtualHost>
```

Directive ServerName

Nom d'hôte et port que le serveur utilise pour s'authentifier lui-même

Syntaxe :

```
ServerName [protocole://]nom de domaine entièrement qualifié[:port]
```

Directive ServerRoot

Racine du répertoire d'installation du serveur

Syntaxe :

```
ServerRoot chemin de répertoire
```

Directive <VirtualHost>

Contient des directives qui ne s'appliquent qu'à un nom d'hôte spécifique ou à une adresse IP

Syntaxe :

```
<VirtualHost adresse_IP[:port] [adresse_IP[:port]] ...> ...
</VirtualHost>
```

?<VirtualHost adresse_IP>::adresse IP du serveur virtuel!!
?<VirtualHost *>::en combinaison avec Name, intercepte toutes les adresses IP!!
?<VirtualHost _default_>::intercepte les adresses IP qui ne correspondent à aucun serveur virtuel.!!
?:port::port du serveur virtuel. S'il n'est pas spécifié, sa valeur par défaut correspond à celle qui est définie par la dernière directive Listen du serveur principal.!!

?:*::accepte tous les ports associés à l'adresse du serveur virtuel!!

Exemple

```
<VirtualHost 10.1.2.3:80>
ServerAdmin webmaster@host.example.com
DocumentRoot /www/docs/host.example.com
ServerName host.example.com
ErrorLog logs/host.example.com-error_log
```

```
TransferLog logs/host.example.com-
access_log
</VirtualHost>
```

Les adresses IPv6 doivent être entourées de crochets car dans le cas contraire, un éventuel port optionnel ne pourrait pas être déterminé. Voici un exemple de serveur virtuel avec adresse IPv6 :

```
<VirtualHost
[2001:db8::a00:20ff:fea7:ccea]:80>
ServerAdmin webmaster@host.example.com
DocumentRoot /www/docs/host.example.com
ServerName host.example.com
ErrorLog logs/host.example.com-error_log
TransferLog logs/host.example.com-
access_log
</VirtualHost>
```

Tout bloc **<VirtualHost>** doit comporter une directive **ServerName**. Dans le cas contraire, le serveur virtuel héritera de la valeur de la directive **ServerName** issue de la configuration du serveur principal.

Désinstallation

Voir aussi

- **(fr)** http://httpd.apache.org/docs/2.2/fr/mod/core.html#options
- **(fr)** http://httpd.apache.org/docs/trunk/fr/mod/mod_authz_host.html

---

Contributeurs principaux : Jamaique.

Basé sur Documentation du Serveur HTTP Apache Version 2.5 par The Apache Software Foundation.