Configuration Sections

Directives in the configuration files[1] may apply to the entire server, or they may be restricted to apply only to particular directories, files, hosts, or URLs. This document describes how to use configuration section containers or `.htaccess` files to change the scope of other configuration directives.

Topics

Types of Configuration Section Containers

Related Modules	Related Directives
core	<Directory>
mod_proxy	<DirectoryMatch>
	<Files>
	<FilesMatch>
	<IfDefine>
	<IfModule>
	<Location>
	<LocationMatch>
	<Proxy>
	<ProxyMatch>
	<VirtualHost>

There are two basic types of containers. Most containers are evaluated for each request. The enclosed directives are applied only for those requests that match the containers. The <IfDefine> and <IfModule> containers, on the other hand, are evaluated only at server startup and restart. If their conditions are true at startup, then the enclosed directives will apply to all requests. If the conditions are not true, the enclosed directives will be ignored.

The <IfDefine> directive encloses directives that will only be applied if an appropriate parameter is defined on the `httpd` command line. For example, with the following configuration, all requests will be redirected to another site only if the server is started using `httpd -DClosedForNow`:

```
<IfDefine ClosedForNow>
  Redirect / http://otherserver.example.com/
</IfDefine>
```

The <IfModule> directive is very similar, except it encloses directives that will only be applied if a particular module is available in the server. The module must either be statically compiled in the server, or it must be dynamically compiled and its LoadModule line must be earlier in the configuration file. This directive should only be used if you need your configuration file to work whether or not certain modules are installed. It should not be used to enclose directives that you want to work all the time, because it can suppress useful error messages about missing modules.

In the following example, the MimeMagicFiles directive will be applied only if mod_mime_magic is available.

```
<IfModule mod_mime_magic.c>
  MimeMagicFile conf/magic
</IfModule>
```

Both <IfDefine> and <IfModule> can apply negative conditions by preceding their test with "!". Also, these sections can be nested to achieve more complex restrictions.

Filesystem and Webspace

The most commonly used configuration section containers are the ones that change the configuration of particular places in the filesystem or webspace. First, it is important to understand the difference between the two. The filesystem is the view of your disks as seen by your operating system. For example, in a default install, Apache resides at /usr/local/apache2 in the Unix filesystem or "c:/Program Files/Apache Group/Apache2" in the Windows filesystem. (Note that forward slashes should always be used as the path separator in Apache, even for Windows.) In contrast, the webspace is the view of your site as delivered by the web server and seen by the client. So the path /dir/ in the webspace corresponds to the path /usr/local/apache2/htdocs/dir/ in the filesystem of a default Apache install on Unix. The webspace need not map directly to the filesystem, since webpages may be generated dynamically from databases or other locations.

Filesystem Containers

The <Directory> and <Files> directives, along with their regex counterparts, apply directives to parts of the filesystem. Directives enclosed in a <Directory> section apply to the named filesystem directory and all subdirectories of that directory. The same effect can be obtained using .htaccess files [2]. For example, in the following configuration, directory indexes will be enabled for the /var/web/dir1 directory and all subdirectories.

```
<Directory /var/web/dir1>
  Options +Indexes
</Directory>
```

Directives enclosed in a <Files> section apply to any file with the specified name, regardless of what directory it lies in. So for example, the following configuration directives will, when placed in the main section of the configuration file, deny access to any file named private.html regardless of where it is found.

```
<Files private.html>
  Order allow,deny
  Deny from all
</Files>
```

To address files found in a particular part of the filesystem, the <Files> and <Directory> sections can be combined. For example, the following configuration will deny access to /var/web/dir1/private.html, /var/web/dir1/subdir2/private.html, /var/web/dir1/subdir3/private.html, and any other instance of private.html found under the /var/web/dir1/ directory.

```
<Directory /var/web/dir1>
  <Files private.html>
    Order allow,deny
    Deny from all
  </Files>
```

```
</Directory>
```

Webspace Containers

The <Location> directive and its regex counterpart, on the other hand, change the configuration for content in the webspace. For example, the following configuration prevents access to any URL-path that begins in /private. In particular, it will apply to requests for http://yoursite.example.com/private, http://yoursite.example.com/private123, and http://yoursite.example.com/private/dir/file.html as well as any other requests starting with the /private string.

```
<Location /private>
  Order Allow,Deny
  Deny from all
</Location>
```

The <Location> directive need not have anything to do with the filesystem. For example, the following example shows how to map a particular URL to an internal Apache handler provided by mod_status. No file called server-status needs to exist in the filesystem.

```
<Location /server-status>
  SetHandler server-status
</Location>
```

Wildcards and Regular Expressions

The <Directory>, <Files>, and <Location> directives can each use shell-style wildcard characters as in fnmatch from the C standard library. The character "*" matches any sequence of characters, "?" matches any single character, and "[*seq*]" matches any character in *seq*. The "/" character will not be matched by any wildcard; it must be specified explictly.

If even more flexible matching is required, each container has a regular-expression (regex) counterpart <DirectoryMatch>, <FilesMatch>, and <LocationMatch> that allow perl-compatible regular expressions[3] to be used in choosing the matches. But see the section below on configuration merging to find out how using regex sections will change how directives are applied.

A non-regex wildcard section that changes the configuration of all user directories could look as follows:

```
<Directory /home/*/public_html>
  Options Indexes
</Directory>
```

Using regex sections, we can deny access to many types of image files at once:

```
<FilesMatch \.(?i:gif|jpe?g|png)$>
  Order allow,deny
  Deny from all
</FilesMatch>
```

What to use When

Choosing between filesystem containers and webspace containers is actually quite easy. When applying directives to objects that reside in the filesystem always use <Directory> or <Files>.

When applying directives to objects that do not reside in the filesystem (such as a webpage generated from a database), use <Location>.

It is important to never use <Location> when trying to restrict access to objects in the filesystem. This is because many different webspace locations (URLs) could map to the same filesystem location, allowing your restrictions to be circumvented. For example, consider the following configuration:

```
<Location /dir/>
  Order allow,deny
  Deny from all
</Location>
```

This works fine if the request is for http://yoursite.example.com/dir/. But what if you are on a case-insensitive filesystem? Then your restriction could be easily circumvented by requesting http://yoursite.example.com/DIR/. The <Directory> directive, in contrast, will apply to any content served from that location, regardless of how it is called. (An exception is filesystem links. The same directory can be placed in more than one part of the filesystem using symbolic links. The <Directory> directive will follow the symbolic link without resetting the pathname. Therefore, for the highest level of security, symbolic links should be disabled with the appropriate Options directive.)

If you are, perhaps, thinking that none of this applies to you because you use a case-sensitive filesystem, remember that there are many other ways to map multiple webspace locations to the same filesystem location. Therefore you should always use the filesystem containers when you can. There is, however, one exception to this rule. Putting configuration restrictions in a <Location /> section is perfectly safe because this section will apply to all requests regardless of the specific URL.

Virtual Hosts

The <VirtualHost> container encloses directives that apply to specific hosts. This is useful when serving multiple hosts from the same machine with a different configuration for each. For more information, see the Virtual Host Documentation[4].

Proxy

The <Proxy> and <ProxyMatch> containers apply enclosed configuration directives only to sites accessed through mod_proxy's proxy server that match the specified URL. For example, the following configuration will prevent the proxy server from being used to access the cnn.com website.

```
<Proxy http://cnn.com/*>
  Order allow,deny
  Deny from all
</Proxy>
```

What Directives are Allowed?

To find out what directives are allowed in what types of configuration sections, check the Context[5] of the directive. Everything that is allowed in <Directory> sections is also syntactically allowed in <DirectoryMatch>, <Files>, <FilesMatch>, <Location>, <LocationMatch>, <Proxy>, and <ProxyMatch> sections. There are some exceptions, however.

- The `AllowOverride` directive works only in `<Directory>` sections.
- The `FollowSymLinks` and `SymLinksIfOwnerMatch` `Options` work only in `<Directory>` sections or `.htaccess` files.
- The `Options` directive cannot be used in `<Files>` and `<FilesMatch>` sections.

How the sections are merged

The configuration sections are applied in a very particular order. Since this can have important effects on how configuration directives are interpreted, it is important to understand how this works.

The order of merging is:

1. `<Directory>` (except regular expressions) and .htaccess done simultaneously (with .htaccess, if allowed, overriding `<Directory>`)
2. `<DirectoryMatch>` (and `<Directory ~>`)
3. `<Files>` and `<FilesMatch>` done simultaneously
4. `<Location>` and `<LocationMatch>` done simultaneously

Apart from `<Directory>`, each group is processed in the order that they appear in the configuration files. `<Directory>` (group 1 above) is processed in the order shortest directory component to longest. So for example, `<Directory /var/web/dir>` will be processed before `<Directory /var/web/dir/subdir>`. If multiple `<Directory>` sections apply to the same directory they are processed in the configuration file order. Configurations included via the `Include` directive will be treated as if they were inside the including file at the location of the `Include` directive.

Sections inside `<VirtualHost>` sections are applied *after* the corresponding sections outside the virtual host definition. This allows virtual hosts to override the main server configuration.

Later sections override earlier ones.

Technical Note

There is actually a `<Location>`/`<LocationMatch>` sequence performed just before the name translation phase (where `Aliases` and `DocumentRoots` are used to map URLs to filenames). The results of this sequence are completely thrown away after the translation has completed.

Some Examples

Below is an artificial example to show the order of merging. Assuming they all apply to the request, the directives in this example will be applied in the order A > B > C > D > E.

```
<Location />
  E
</Location>

<Files f.html>
  D
</Files>

<VirtualHost *>
  <Directory /a/b>
    B
  </Directory>
</VirtualHost>
```

```
<DirectoryMatch "^.*b$">
  C
</DirectoryMatch>

<Directory /a/b>
  A
</Directory>
```

For a more concrete example, consider the following. Regardless of any access restrictions placed in `<Directory>` sections, the `<Location>` section will be evaluated last and will allow unrestricted access to the server. In other words, order of merging is important, so be careful!

```
<Location />
  Order deny,allow
  Allow from all
</Location>

# Woops! This <Directory> section will have no effect
<Directory />
  Order allow,deny
  Allow from all
  Deny from badguy.example.com
</Directory>
```

URI References

[1] http://httpd.apache.org/docs-2.1/configuring.html
[2] http://httpd.apache.org/docs-2.1/howto/htaccess.html
[3] http://httpd.apache.org/docs-2.1/glossary.html#regex
[4] http://httpd.apache.org/docs-2.1/vhosts/
[5] http://httpd.apache.org/docs-2.1/mod/directive-dict.html#Context