Environment Variables in Apache

The Apache HTTP Server provides a mechanism for storing information in named variables that are called *environment variables*. This information can be used to control various operations such as logging or access control. The variables are also used as a mechanism to communicate with external programs such as CGI scripts. This document discusses different ways to manipulate and use these variables.

Although these variables are referred to as *environment variables*, they are not the same as the environment variables controlled by the underlying operating system. Instead, these variables are stored and manipulated in an internal Apache structure. They only become actual operating system environment variables when they are provided to CGI scripts and Server Side Include scripts. If you wish to manipulate the operating system environment under which the server itself runs, you must use the standard environment manipulation mechanisms provided by your operating system shell.

Topics

Setting Environment Variables

Related Modules	Related Directives
mod_env	BrowserMatch
mod_rewrite	BrowserMatchNoCase
mod_setenvif	PassEnv
mod_unique_id	RewriteRule
	SetEnv
	SetEnvIf
	SetEnvIfNoCase
	UnsetEnv

Basic Environment Manipulation

The most basic way to set an environment variable in Apache is using the unconditional `SetEnv` directive. Variables may also be passed from the environment of the shell which started the server using the `PassEnv` directive.

Conditional Per-Request Settings

For additional flexibility, the directives provided by mod_setenvif allow environment variables to be set on a per-request basis, conditional on characteristics of particular requests. For example, a variable could be set only when a specific browser (User-Agent) is making a request, or only when a specific Referer [sic] header is found. Even more flexibility is available through the mod_rewrite's `RewriteRule` which uses the `[E=...]` option to set environment variables.

Unique Identifiers

Finally, mod_unique_id sets the environment variable `UNIQUE_ID` for each request to a value which is guaranteed to be unique across "all" requests under very specific conditions.

Standard CGI Variables

In addition to all environment variables set within the Apache configuration and passed from the shell,

CGI scripts and SSI pages are provided with a set of environment variables containing meta-information about the request as required by the CGI specification[1].

Some Caveats

- It is not possible to override or change the standard CGI variables using the environment manipulation directives.
- When suexec[2] is used to launch CGI scripts, the environment will be cleaned down to a set of *safe* variables before CGI scripts are launched. The list of *safe* variables is defined at compile-time in `suexec.c`.
- For portability reasons, the names of environment variables may contain only letters, numbers, and the underscore character. In addition, the first character may not be a number. Characters which do not match this restriction will be replaced by an underscore when passed to CGI scripts and SSI pages.

Using Environment Variables

Related Modules	Related Directives
mod_authz_host	Allow
mod_cgi	CustomLog
mod_ext_filter	Deny
mod_headers	ExtFilterDefine
mod_include	Header
mod_log_config	LogFormat
mod_rewrite	RewriteCond
	RewriteRule

CGI Scripts

One of the primary uses of environment variables is to communicate information to CGI scripts. As discussed above, the environment passed to CGI scripts includes standard meta-information about the request in addition to any variables set within the Apache configuration. For more details, see the CGI tutorial[3].

SSI Pages

Server-parsed (SSI) documents processed by mod_include's `INCLUDES` filter can print environment variables using the `echo` element, and can use environment variables in flow control elements to makes parts of a page conditional on characteristics of a request. Apache also provides SSI pages with the standard CGI environment variables as discussed above. For more details, see the SSI tutorial[4].

Access Control

Access to the server can be controlled based on the value of environment variables using the `allow from env=` and `deny from env=` directives. In combination with `SetEnvIf`, this allows for flexible control of access to the server based on characteristics of the client. For example, you can use these directives to deny access to a particular browser (User-Agent).

Conditional Logging

Environment variables can be logged in the access log using the `LogFormat` option `%e`. In addition, the decision on whether or not to log requests can be made based on the status of environment variables using the conditional form of the `CustomLog` directive. In combination with `SetEnvIf` this allows for flexible control of which requests are logged. For example, you can choose not to log requests for filenames ending in `gif`, or you can choose to only log requests from clients which are

outside your subnet.

Conditional Response Headers

The `Header` directive can use the presence or absence of an environment variable to determine whether or not a certain HTTP header will be placed in the response to the client. This allows, for example, a certain response header to be sent only if a corresponding header is received in the request from the client.

External Filter Activation

External filters configured by `mod_ext_filter` using the `ExtFilterDefine` directive can by activated conditional on an environment variable using the `disableenv=` and `enableenv=` options.

URL Rewriting

The `%{ENV:...}` form of *TestString* in the `RewriteCond` allows mod_rewrite's rewrite engine to make decisions conditional on environment variables. Note that the variables accessible in mod_rewrite without the `ENV:` prefix are not actually environment variables. Rather, they are variables special to mod_rewrite which cannot be accessed from other modules.

Special Purpose Environment Variables

Interoperability problems have led to the introduction of mechanisms to modify the way Apache behaves when talking to particular clients. To make these mechanisms as flexible as possible, they are invoked by defining environment variables, typically with `BrowserMatch`, though `SetEnv` and `PassEnv` could also be used, for example.

downgrade-1.0

This forces the request to be treated as a HTTP/1.0 request even if it was in a later dialect.

force-no-vary

This causes any `Vary` fields to be removed from the response header before it is sent back to the client. Some clients don't interpret this field correctly (see the known client problems[5] page); setting this variable can work around this problem. Setting this variable also implies **force-response-1.0**.

force-response-1.0

This forces an HTTP/1.0 response to clients making an HTTP/1.0 request. It was originally implemented as a result of a problem with AOL's proxies. Some HTTP/1.0 clients may not behave correctly when given an HTTP/1.1 response, and this can be used to interoperate with them.

gzip-only-text/html

When set to a value of "1", this variable disables the DEFLATE output filter provided by `mod_deflate` for content-types other than `text/html`.

no-gzip

When set, the `DEFLATE` filter of `mod_deflate` will be turned off.

nokeepalive

This disables `KeepAlive` when set.

redirect-carefully

This forces the server to be more careful when sending a redirect to the client. This is typically used when a client has a known problem handling redirects. This was originally implemented as a result of a problem with Microsoft's WebFolders software which has a problem handling redirects on directory resources via DAV methods.

suppress-error-charset

Available in versions after 2.0.40

When Apache issues a redirect in response to a client request, the response includes some actual text to be displayed in case the client can't (or doesn't) automatically follow the redirection. Apache ordinarily labels this text according to the character set which it uses, which is ISO-8859-1.

However, if the redirection is to a page that uses a different character set, some broken browser versions will try to use the character set from the redirection text rather than the actual page. This can result in Greek, for instance, being incorrectly rendered.

Setting this environment variable causes Apache to omit the character set for the redirection text, and these broken browsers will then correctly use that of the destination page.

Examples

Changing protocol behavior with misbehaving clients

We recommend that the following lines be included in httpd.conf to deal with known client problems.

```
#
# The following directives modify normal HTTP response behavior.
# The first directive disables keepalive for Netscape 2.x and browsers that
# spoof it. There are known problems with these browser implementations.
# The second directive is for Microsoft Internet Explorer 4.0b2
# which has a broken HTTP/1.1 implementation and does not properly
# support keepalive when it is used on 301 or 302 (redirect) responses.
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

#
# The following directive disables HTTP/1.1 responses to browsers which
# are in violation of the HTTP/1.0 spec by not being able to grok a
# basic 1.1 response.
#
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
```

Do not log requests for images in the access log

This example keeps requests for images from appearing in the access log. It can be easily modified to prevent logging of particular directories, or to prevent logging of requests coming from particular hosts.

```
SetEnvIf Request_URI \.gif image-request
SetEnvIf Request_URI \.jpg image-request
```

```
SetEnvIf Request_URI \.png image-request
CustomLog logs/access_log common env=!image-request
```

Prevent "Image Theft"

This example shows how to keep people not on your server from using images on your server as inline-images on their pages. This is not a recommended configuration, but it can work in limited circumstances. We assume that all your images are in a directory called /web/images.

```
SetEnvIf Referer "^http://www.example.com/" local_referal
# Allow browsers that do not send Referer info
SetEnvIf Referer "^$" local_referal
<Directory /web/images>
   Order Deny,Allow
   Deny from all
   Allow from env=local_referal
</Directory>
```

For more information about this technique, see the ApacheToday tutorial " Keeping Your Images from Adorning Other Sites[6]".

URI References

[1] http://cgi-spec.golux.com/
[2] http://httpd.apache.org/docs-2.1/suexec.html
[3] http://httpd.apache.org/docs-2.1/howto/cgi.html
[4] http://httpd.apache.org/docs-2.1/howto/ssi.html
[5] http://httpd.apache.org/docs-2.1/misc/known_client_problems.html
[6] http://apachetoday.com/news_story.php3?ltsn=2000-06-14-002-01-PS