

1 Apache::Registry - Run unaltered CGI scrips under mod_perl

1.1 Synopsis

```
#in httpd.conf

Alias /perl/ /perl/apache/scripts/ #optional
PerlModule Apache::Registry

<Location /perl>
    SetHandler perl-script
    PerlHandler Apache::Registry
    Options ExecCGI
</Location>
```

1.2 Description

Apache::Registry is the Apache module allowing you to run CGI scripts very fast under mod_perl, by compiling all scripts once and then caching them in memory.

URIs in the form of *http://www.example.com/perl/file.pl* will be compiled as the body of a perl subroutine and executed. Each server process or 'child' will compile the subroutine once and store it in memory. It will recompile it whenever the file is updated on disk. Think of it as an object oriented server with each script implementing a class loaded at runtime.

The file looks much like a "normal" script, but it is compiled or 'eval'd' into a subroutine.

Here's an example:

```
my $r = Apache->request;
$r->content_type( "text/html" );
$r->send_http_header;
$r->print( "Hi There!" );
```

This module emulates the CGI environment, allowing programmers to write scripts that run under CGI or mod_perl without change. Existing CGI scripts may require some changes, simply because a CGI script has a very short lifetime of one HTTP request, allowing you to get away with "quick and dirty" scripting. Using mod_perl and Apache::Registry requires you to be more careful, but it also gives new meaning to the word "quick"!

Be sure to read all mod_perl related documentation for more details, including instructions for setting up an environment that looks exactly like CGI:

```
print "Content-type: text/html\n\n";
print "Hi There!";
```

Note that each httpd process or "child" must compile each script once, so the first request to one server may seem slow, but each request there after will be faster. If your scripts are large and/or make use of many Perl modules, this difference should be noticeable to the human eye.

1.3 Security

Apache::Registry::handler will preform the same checks as mod_cgi before running the script.

1.4 Environment

The Apache function `exit` overrides the Perl core built-in function.

The environment variable `GATEWAY_INTERFACE` is set to `CGI-Perl/1.1`.

1.5 Command Line Switches on the First Line

Normally when a Perl script is run from the command line or under CGI, arguments on the `#!` line are passed to the perl interpreter for processing.

Apache::Registry currently only honors the `-w` switch and will turn on warnings using the `$^W` global variable. Another common switch used with CGI scripts is `-T` to turn on taint checking. This can only be enabled when the server starts with the configuration directive:

```
PerlTaintCheck On
```

However, if taint checking is not enabled, but the `-T` switch is seen, Apache::Registry will write a warning to the *error_log*.

1.6 Debugging

You may set the debug level with the `$Apache::Registry::Debug` bitmask

```
1 => log recompile in errorlog
2 => Apache::Debug::dump in case of $@
4 => trace pedantically
```

1.7 Caveats

Apache::Registry makes things look just the CGI environment, however, you must understand that this **is not CGI**. Each httpd child will compile your script into memory and keep it there, whereas CGI will run it once, cleaning out the entire process space. Many times you have heard "always use `-w`, always use `-w` and use `strict`". This is more important here than anywhere else!

Your scripts cannot contain the `__END__` or `__DATA__` token to terminate compilation.

1.8 See Also

perl, mod_perl, Apache, Apache::Debug

1.9 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

- **The documentation mailing list**

1.10 Authors

- **Andreas J. Koenig**
- **Doug MacEachern**

Only the major authors are listed above. For contributors see the Changes file.

Table of Contents:

1	Apache::Registry - Run unaltered CGI scrips under mod_perl	1
1.1	Synopsis	2
1.2	Description	2
1.3	Security	3
1.4	Environment	3
1.5	Command Line Switches on the First Line	3
1.6	Debugging	3
1.7	Caveats	3
1.8	See Also	4
1.9	Maintainers	4
1.10	Authors	4