mod_perl Quick Reference Card

Revision 1.0 for mod_perl version 1.19

Andrew Ford

refcards.com™

mod_perl is an Apache module, created by Doug MacEachern, that embeds a Perl interpreter within the server. It provides a Perl API to Apache and adds a number of Apache configuration directives. Scripts using **mod_perl** should import the *Apache* module, *Apache::Constants*, and other *Apache::* modules. A reference to the request object (denoted below by `$r`) is passed to Perl handlers when they are invoked.

Client request methods

```
$r    = Apache->request();
$str  = $r->args();                     # or %hash = ...
$c    = $r->connection          # see Apache::Connection
$str  = $r->content();                  # or %hash = ...
$str  = $r->filename( [$newval] );
        $r->finfo();
$str  = $r->get_remote_host( [$lookup_type] );
               # use Apache::Constants :remotehost tag
$str  = $r->get_remote_logname();
$str  = $r->header_in( $hdr [, $newval] );
$bool = $r->header_only();
$href = $r->headers_in();               # or %hash = ...
$str  = $r->method( [$newval] );
$num  = $r->method_number([$nv]);   # use :methods tag
$u    = $r->parsed_uri();              # see Apache::URI
$str  = $r->path_info( [$newval] );
$str  = $r->protocol();
$bool = $r->proxyreq( [$newval] );
        $r->read( $buf, $bytes_to_read );
$s    = $r->server                  # see Apache::Server
$str  = $r->the_request();
$str  = $r->uri( [$newval] );
```

Server response methods

```
$num  = $r->bytes_sent();
        $r->cgi_header_out( $hdr [, $newval]);
$str  = $r->content_encoding( [$newval] );
$aref = $r->content_languages( [$newval] );
$str  = $r->content_type( [$newval] );
        $r->custom_response( $code, $uri );
$str  = $r->err_header_out( $hdr [, $newval] );
$href = $r->err_headers_out();          # or %hash = ...
$str  = $r->handler( [$newval] );
$str  = $r->header_out( $hdr [, $newval] );
$href = $r->headers_out();              # or %hash = ...
$bool = $r->no_cache( [$newval] );
$num  = $r->request_time();
$num  = $r->status( [$newval] );
$str  = $r->status_line( [$newval] );
```

Sending data to the client

```
        $r->print( @list );                  # checks $|
        $r->printf( $format, @args );
        $r->rflush();
        $r->send_cgi_header( $str );
$len  = $r->send_fd( $filehandle );
        $r->send_http_header( [$content_type] );
```

Server core functions

```
        $r->chdir_file( $file );
        $r->child_terminate();
        $r->hard_timeout( $msg );
        $r->internal_redirect( $newplace );
        $r->internal_redirect_handler( $newplace );
$bool = $r->is_initial_req();
$bool = $r->is_main();
        $r->kill_timeout();
$str  = $r->location();
$req  = $r->last();
$req  = $r->main();
$req  = $r->next();
$str  = $r->notes($k[,$v]);    # or $tab = $r->notes()
$req  = $r->prev();
        $r->register_cleanup( $code_ref );
        $r->reset_timeout();
        $r->soft_timeout( $msg );
$str  = $r->subprocess_env( [$k [, $v]] );
```

Server configuration methods

```
$str  = $r->dir_config($k); #or $tab=$r->dir_config()
$str  = $r->document_root();
$str  = $r->get_server_name();
$num  = $r->get_server_port();
$str  = $r->server_root_relative( [$obj] );
```

Logging and the Apache::Log class

```
$str  = $r->as_string();
        $r->log_reason( $message, $file );
        $r->log_error( $message );
        $r->warn( $message );
$log  = $r->log();
$log  = $s->log();
        $log->emerg  ( {$str ...|$code_ref} );
        $log->alert  ( {$msg ...|$code_ref} );
        $log->crit   ( {$msg ...|$code_ref} );
        $log->error  ( {$msg ...|$code_ref} );
        $log->warn   ( {$msg ...|$code_ref} );
        $log->notice ( {$msg ...|$code_ref} );
        $log->info   ( {$msg ...|$code_ref} );
        $log->debug  ( {$msg ...|$code_ref} );
```

Access control methods

```
$opts = $r->allow_options();        # use :options tag
$str  = $r->auth_name( [$newval] );
$str  = $r->auth_type();
($rc, $pw) = $r->get_basic_auth_pw();
        $r->note_basic_auth_failure();
$aref = $r->requires();
$flag = $r->satisfies();          # use :satisfies tag
$bool = $r->some_auth_required();
```

mod_perl specific methods

```
$str  = $r->current_callback();
$bool = $r->define( $name );
        Apache->exit( [$code] );
$fh   = Apache->gensym();
$aref = $r->get_handlers( $str );
        Apache->httpd_conf( $str );
$bool = $r->module( $name );
$bool = Apache->perl_hook( $name );
        $r->post_connection( $code_ref );
        $r->push_handlers( $str => $code_ref );
$r    = Apache->request( [$r] );
        $r->set_handlers( $str => $aref );
```

Apache::SubRequest class

```
$subr = $r->lookup_uri($uri);
$subr = $r->lookup_file($filename);
$rc   = $subr->run();
```

Apache::Server class

```
$s    = Apache->server                   # or $r->server
$bool = $s->is_virtual();
        $s->log_error();
$aref = $s->names();
$s    = $s->next();
$num  = $s->port();
$str  = $s->server_admin();
$str  = $s->server_hostname();
$num  = $s->timeout( [$newval] );
        $s->warn();
```

Apache::Connection class

```
$bool = $c->aborted();
$str  = $c->auth_type();
$addr = $c->local_addr();
$addr = $c->remote_addr( [$addr] );
$str  = $c->remote_host();
$str  = $c->remote_ip( [$ip] );
$str  = $c->remote_logname();
$str  = $c->user( [$username] );
```

Apache::Table class

```
$tab  = Apache::Table->new( $r [, $size] );
        $tab->add( $key, $str_or_aref );
        $tab->clear();
        $tab->do( $code_ref );
        $tab->merge( $key, $str_or_aref );
        $tab->set( $key, $str );
$str  = $tab->get( $key );             # or @list = ...
        $tab->unset($key);
```

Apache::URI class

```
$uri  = Apache::URI->parse( $r [, $string_uri] );
$str  = $uri->unparse();
$str  = $uri->component( [$newval] );
```

(where *component* is one of: `fragment`, `hostinfo`, `hostname`, `password`, `path_info`, `path`, `port`, `query`, `rpath`, `scheme`, `user`)

Apache::Util class

```
$str  = Apache::Util::escape_html( $html );
$str  = Apache::Util::escape_uri( $uri );
$str  = Apache::Util::ht_time($time[, $fmt[, $bool]]);
$secs = Apache::Util::parsedate( $date_str );
$num  = Apache::Util::size_string( $num );
$str  = Apache::Util::unescape_uri( $uri );
$str  = Apache::Util::unescape_uri_info( $uri );
```

`Apache::Constants` class

The following export tag groups are defined (HTTP status code synonyms are given in brackets):

:common: OK, DECLINED, DONE, NOT_FOUND, FORBIDDEN, AUTH_REQUIRED (HTTP_UNAUTHORIZED), SERVER_ERROR

:response: DOCUMENT_FOLLOWS (HTTP_OK), MOVED (HTTP_MOVED_PERMANENTLY), REDIRECT (HTTP_MOVED_TEMPORARILY), USE_LOCAL_COPY (HTTP_NOT_MODIFIED), BAD_REQUEST, BAD_GATEWAY, NOT_IMPLEMENTED, CONTINUE, NOT_AUTHORITATIVE

:methods: M_CONNECT, M_COPY, M_DELETE, M_GET, M_INVALID, M_LOCK, M_MKCOL, M_MOVE, M_OPTIONS, M_PATCH, M_POST, M_PROPFIND, M_PROPPATCH, M_PUT, M_TRACE, M_UNLOCK, METHODS

:options: OPT_ALL OPT_NONE, OPT_INDEXES, OPT_INCLUDES, OPT_SYM_LINKS, OPT_EXECCGI, OPT_UNSET, OPT_INCNOEXEC, OPT_SYM_OWNER, OPT_MULTI,

:satisfies: SATISFY_ALL, SATISFY_ANY, SATISFY_NOSPEC

:server: MODULE_MAGIC_NUMBER, SERVER_BUILT, SERVER_VERSION

:remotehost: REMOTE_HOST, REMOTE_NAME, REMOTE_NOLOOKUP, REMOTE_DOUBLE_REV

:http includes only those HTTP status code constants shown below in bold type (other HTTP constants may be imported explicitly):

100	HTTP_CONTINUE	405	**HTTP_METHOD_NOT_ALLOWED**
101	HTTP_SWITCHING_PROTOCOLS	406	**HTTP_NOT_ACCEPTABLE**
200	**HTTP_OK**	407	HTTP_PROXY_AUTHENTICATION_REQUIRED
201	HTTP_CREATED		
202	HTTP_ACCEPTED	408	HTTP_REQUEST_TIMEOUT
203	HTTP_NON_AUTHORITATIVE	409	HTTP_CONFLICT
204	**HTTP_NO_CONTENT**	410	HTTP_GONE
205	HTTP_RESET_CONTENT	411	HTTP_LENGTH REQUIRED
206	HTTP_PARTIAL_CONTENT	412	**HTTP_PRECONDITION_FAILED**
300	HTTP_MULTIPLE_CHOICES	413	HTTP_REQUEST_ENTITY_TOO_LARGE
301	**HTTP_MOVED_PERMANENTLY**	414	HTTP_REQUEST_URI_TOO_LARGE
302	**HTTP_MOVED_TEMPORARILY**	415	HTTP_UNSUPPORTED_MEDIA_TYPE
303	HTTP_SEE_OTHER	500	HTTP_INTERNAL_SERVER_ERROR
304	**HTTP_NOT_MODIFIED**	501	HTTP_NOT IMPLEMENTED
305	HTTP_USE_PROXY	502	HTTP_BAD_GATEWAY
400	**HTTP_BAD_REQUEST**	503	**HTTP_SERVICE_UNAVAILABLE**
401	**HTTP_UNAUTHORIZED**	504	HTTP_GATEWAY_TIME_OUT
402	HTTP_PAYMENT_REQUIRED	505	HTTP_VERSION_NOT_SUPPORTED
403	**HTTP_FORBIDDEN**	506	**HTTP_VARIANT_ALSO_VARIES**
404	**HTTP_NOT_FOUND**		

Magic global variables

$0, $^X, $|, $/, %@, %SIG, @INC, %INC, %ENV{MOD_PERL}, %ENV{GATEWAY_INTERFACE}, %ENV{PERL_SEND_HEADER}

Special package globals

$Apache::Server::CWD	$Apache::Server::SaveConfig
$Apache::Server::ReStarting	$Apache::Server::Starting

HTTP 1.1 headers

Syntax	Category
Accept: *media-types*[;q=*qvalue*] [, ...]	REQUEST
Accept-Charset: *charset*[;q=*qvalue*] [, ...]	REQUEST
Accept-Encoding: *encoding*[;q=*qvalue*] [, ...]	REQUEST
Accept-Language: *lang*[;q=*qvalue*] [, ...]	REQUEST
Accept-Ranges: {bytes|none}	RESPONSE
Age: *seconds*	RESPONSE
Allow: *method* [, ...]	ENTITY
Authorization: *scheme credentials*	REQUEST
Cache-Control: *directive*	GENERAL
Connection: close	GENERAL
Content-Base: *uri*	ENTITY
Content-Encoding: *enc*	ENTITY
Content-Language: *lang*	ENTITY
Content-Length: *len*	ENTITY
Content-MD5: *digest*	ENTITY
Content-Range: bytes *range*/*length*	ENTITY
Content-Type: *media-type*	ENTITY
Cookie: *name*=*value* [; ...]	REQUEST
Date: *date*	GENERAL
ETag: *entity-tag*	RESPONSE
Expect: *expectation*	REQUEST
Expires: *date*	ENTITY
From: *email-address*	REQUEST
Host: *hostname*[:*port*]	REQUEST
If-Match: *entity-tag*	REQUEST
If-Modified-Since: *date*	REQUEST
If-None-Match: *entity-tag*	REQUEST
If-Range: {*entity tag*|*date*}	REQUEST
If-Unmodified-Since: *date*	REQUEST
Last-Modified: *date*	ENTITY
Location: *uri*	RESPONSE
MIME-Version: *version*	GENERAL
Max-Forwards: *number*	REQUEST
Pragma: {no-cache|*extension-pragma*}	GENERAL
Proxy-Authenticate: *challenge*	RESPONSE
Proxy-Authorization: *credentials*	REQUEST
Public: *method* ...	RESPONSE
Range: bytes=*n*[-*m*] [, ...]	REQUEST
Referer: *url*	REQUEST
Retry-After: {*date*|*seconds*}	RESPONSE
Server: *string*	RESPONSE
Set-Cookie: *name*=*value*[; *options*]	RESPONSE
TE: *coding*	REQUEST
Trailer: *header*	GENERAL
Transfer-Encoding: *coding*	GENERAL
Upgrade: *protocol* [, ...]	GENERAL
User-Agent: *string*	REQUEST
Vary: *header* [, ...]	RESPONSE
Via: [*protocol*/]*version* [(*comment*)] [, ...]	GENERAL
WWW-Authenticate: *scheme realm*	RESPONSE
Warning: *code agent* "*text*" [*date*]	GENERAL

Apache `mod_perl` configuration directives

`mod_perl` enables Apache to be configured using Perl statements that are contained within `<Perl>...</Perl>` sections and adds the Apache configuration directives listed below. Each directive is given with its arguments; defaults are given where appropriate in parentheses at the end of the line, followed by the symbol ❖ to mark directives only valid in a directory section or `.htaccess` file.

PerlAccessHandler *handler*		
PerlAuthenHandler *handler*		
PerlAuthzHandler *handler*		
PerlChildExitHandler *handler*		❖
PerlChildInitHandler *handler*		❖
PerlCleanupHandler *handler*		
PerlDispatchHandler *handler*		
PerlFixupHandler *handler*		
PerlFreshRestart {On|Off}	(On)	❖
PerlHandler *handler*		
PerlHeaderParserHandler *handler*		
PerlInitHandler *handler*		
PerlLogHandler *handler*		
PerlModule		
PerlPassEnv *name* ...		❖
PerlPostReadRequestHandler *handler*		❖
PerlRequire *script-file*		
PerlSendHeader {On|Off}	(Off)	
PerlSetEnv *name value*		
PerlSetVar *name value*		
PerlSetupEnv {On|Off}	(Off)	
PerlTaintCheck {On|Off}	(Off)	❖
PerlTransHandler *handler*		❖
PerlTypeHandler *handler*		
PerlWarn {On|Off}	(Off)	❖

Resources

http://perl.apache.org	The Apache/Perl Integration Project
http://www.modperl.com	*Writing Apache Modules* home page
http://www.apache.org	Apache home page
http://www.perl.com	Perl home page
http://www.refcards.com	Quick reference cards

mod_perl Quick Reference Card
A **refcards.com**™ quick reference card
Revision 1.0 for **mod_perl** version 1.19 [May 1999]