1   Apache2::HookRun - Perl API for Invoking Apache HTTP phases

1.1 Synopsis

```
# httpd.conf
PerlProcessConnectionHandler MyApache2::PseudoHTTP::handler

#file:MyApache2/PseudoHTTP.pm
#-------------------------
package MyApache2::PseudoHTTP;

use Apache2::HookRun ();
use Apache2::RequestUtil ();
use Apache2::RequestRec ();

use Apache2::Const -compile => qw(OK DECLINED DONE SERVER_ERROR);

# implement the HTTP protocol cycle in protocol handler
sub handler {
    my $c = shift;
    my $r = Apache2::RequestRec->new($c);

    # register any custom callbacks here, e.g.:
    # $r->push_handlers(PerlAccessHandler => \&my_access);

    $rc = $r->run_post_read_request();
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    $rc = $r->run_translate_name;
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    $rc = $r->run_map_to_storage;
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    # this must be run all a big havoc will happen in the following
    # phases
    $r->location_merge($path);

    $rc = $r->run_header_parser;
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    my $args = $r->args || '';
    if ($args eq 'die') {
        $r->die(Apache2::Const::SERVER_ERROR);

        return Apache2::Const::DONE;
    }

    $rc = $r->run_access_checker;
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    $rc = $r->run_auth_checker;
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    $rc = $r->run_check_user_id;
    return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

    $rc = $r->run_type_checker;
```

```
        return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

        $rc = $r->run_fixups;
        return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

        # $r->run_handler is called internally by $r->invoke_handler,
        # invoke_handler sets all kind of filters, and does a few other
        # things but it's possible to call $r->run_handler, bypassing
        # invoke_handler
        $rc = $r->invoke_handler;
        return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

        $rc = $r->run_log_transaction;
        return $rc unless $rc == Apache2::Const::OK or $rc == Apache2::Const::DECLINED;

        return Apache2::Const::OK;
    }
```

1.2 Description

`Apache2::HookRun` exposes parts of the Apache HTTP protocol implementation, responsible for invoking callbacks for each HTTP Request cycle phase.

Armed with that API, you could run some of the http protocol framework parts when implementing your own protocols. For example see how HTTP AAA (access, auth and authz) hooks are called from a protocol handler, implementing a command server, which has nothing to do with HTTP. Also you can see in Synopsis how to re-implement Apache HTTP cycle in the protocol handler.

Using this API you could probably also change the normal Apache behavior (e.g. invoking some hooks earlier than normal, or later), but before doing that you will probably need to spend some time reading through the Apache C code. That's why some of the methods in this document, point you to the specific functions in the Apache source code. If you just try to use the methods from this module, without understanding them well, don't be surprised if you will get some nasty crashes, from which mod_perl can't protect you.

1.3 API

`Apache2::HookRun` provides the following functions and/or methods:

1.3.1 *die*

Kill the current request

```
  $r->die($type);
```

- **obj: $r (Apache2::RequestRec object)**

    The current request

- **arg1: `$type` (integer)**

  Why the request is dieing. Expects an Apache status constant.

- **ret: no return value**
- **since: 2.0.00**

This method doesn't really abort the request, it just handles the sending of the error response, logging the error and such. You want to take a look at the internals of `ap_die()` in *httpd-2.0/modules/http/http_request.c* for more details.

1.3.2 `invoke_handler`

Run the response phase.

```
$rc = $r->invoke_handler();
```

- **obj: `$r` (`Apache2::RequestRec object`)**

  The current request

- **ret: `$rc` (integer)**

  The status of the current phase run: `Apache2::Const::OK`, `Apache2::HTTP_...`

- **since: 2.0.00**

`invoke_handler()` allows modules to insert filters, sets a default handler if none is set, runs `run_handler()` and handles some errors.

For more details see `ap_invoke_handler()` in *httpd-2.0/server/config.c*.

1.3.3 `run_access_checker`

Run the resource access control phase.

```
$rc = $r->run_access_checker();
```

- **obj: `$r` (`Apache2::RequestRec object`)**

  the current request

- **ret: `$rc` (integer)**

  The status of the current phase run: `Apache2::Const::OK`, `Apache2::Const::DECLINED`, `Apache2::HTTP_....`

- **since: 2.0.00**

This phase runs before a user is authenticated, so this hook is really to apply additional restrictions independent of a user. It also runs independent of 'Require' directive usage.

1.3.4 `run_auth_checker`

Run the authentication phase.

```
$rc = $r->run_auth_checker();
```

- **obj: $r (Apache2::RequestRec object)**

  the current request

- **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK, Apache2::Const::DECLINED, Apache2::HTTP_....

- **since: 2.0.00**

This phase is used to check to see if the resource being requested is available for the authenticated user ($r->user and $r->ap_auth_type).

It runs after the access_checker and check_user_id hooks.

Note that it will only be called if Apache determines that access control has been applied to this resource (through a 'Require' directive).

1.3.5 `run_check_user_id`

Run the authorization phase.

```
$rc = $r->run_check_user_id();
```

- **obj: $r (Apache2::RequestRec object)**

  The current request

- **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK, Apache2::Const::DECLINED, Apache2::HTTP_....

- **since: 2.0.00**

This hook is used to analyze the request headers, authenticate the user, and set the user information in the request record ($r->user and $r->ap_auth_type).

This hook is only run when Apache determines that authentication/authorization is required for this resource (as determined by the 'Require' directive).

It runs after the access_checker hook, and before the auth_checker hook.

1.3.6 *run_fixups*

Run the fixup phase.

```
$rc = $r->run_fixups();
```

* **obj: $r (Apache2::RequestRec object)**

  The current request

* **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK, Apache2::Const::DECLINED, Apache2::HTTP_....

* **since: 2.0.00**

This phase allows modules to perform module-specific fixing of HTTP header fields. This is invoked just before the response phase.

1.3.7 *run_handler*

Run the response phase.

```
$rc = $r->run_handler();
```

* **obj: $r (Apache2::RequestRec object)**

  The request_rec

* **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK, Apache2::Const::DECLINED, Apache2::HTTP_....

* **since: 2.0.00**

run_handler() is called internally by invoke_handler(). Use run_handler() only if you want to bypass the extra functionality provided by invoke_handler().

1.3.8 run_header_parser

Run the header parser phase.

```
$rc = $r->run_header_parser();
```

- **obj: $r (Apache2::RequestRec object)**

  The current request

- **ret: $rc (integer)**

  Apache2::Const::OK or Apache2::Const::DECLINED.

- **since: 2.0.00**

1.3.9 run_log_transaction

Run the logging phase.

```
$rc = $r->run_log_transaction();
```

- **obj: $r (Apache2::RequestRec object)**

  The current request

- **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK, Apache2::Const::DECLINED,
  Apache2::HTTP_...

- **since: 2.0.00**

This hook allows modules to perform any module-specific logging activities over and above the normal
server things.

1.3.10 run_map_to_storage

Run the map_to_storage phase.

```
$rc = $r->run_map_to_storage();
```

- **obj: $r (Apache2::RequestRec object)**

  The current request

- **ret: $rc (integer)**

Apache2::Const::DONE (or Apache2::HTTP_*) if this contextless request was just fulfilled (such as TRACE), Apache2::Const::OK if this is not a file, and Apache2::Const::DECLINED if this is a file. The core map_to_storage (Apache2::HOOK_RUN_LAST) will directory_walk() and file_walk() the $r->filename (all internal C functions).

- **since: 2.0.00**

This phase allows modules to set the per_dir_config based on their own context (such as <Proxy> sections) and responds to contextless requests such as TRACE that need no security or filesystem mapping based on the filesystem.

1.3.11 run_post_read_request

Run the post_read_request phase.

```
$rc = $r->run_post_read_request();
```

- **obj: $r (Apache2::RequestRec object)**

  The current request

- **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK or Apache2::Const::DECLINED.

- **since: 2.0.00**

This phase is run right after read_request() or internal_redirect(), and not run during any subrequests. This hook allows modules to affect the request immediately after the request has been read, and before any other phases have been processes. This allows modules to make decisions based upon the input header fields

1.3.12 run_translate_name

Run the translate phase.

```
$rc = $r->run_translate_name();
```

- **obj: $r (Apache2::RequestRec object)**

  The current request

- **ret: $rc (integer)**

  The status of the current phase run: Apache2::Const::OK, Apache2::Const::DECLINED, Apache2::HTTP_....

- **since: 2.0.00**

This phase gives modules an opportunity to translate the URI into an actual filename. If no modules do anything special, the server's default rules will be applied.

1.3.13 `run_type_checker`

Run the type_checker phase.

```
$rc = $r->run_type_checker();
```

- **obj: `$r` (`Apache2::RequestRec object`)**

  the current request

- **ret: `$rc` (integer)**

  The status of the current phase run: `Apache2::Const::OK`, `Apache2::Const::DECLINED`, `Apache2::HTTP_....`

- **since: 2.0.00**

This phase is used to determine and/or set the various document type information bits, like `Content-type` (via `$r->content_type`), language, etc.

1.4 See Also

mod_perl 2.0 documentation.

1.5 Copyright

mod_perl 2.0 and its core modules are copyrighted under The Apache Software License, Version 2.0.

1.6 Authors

The mod_perl development team and numerous contributors.

Table of Contents: