1   Apache2::SubRequest - Perl API for Apache subrequests

1.1 Synopsis

```
use Apache2::SubRequest ();

# run internal redirects at once
$r->internal_redirect($new_uri);
$r->internal_redirect_handler($new_uri);

# create internal redirect objects
$subr = $r->lookup_uri("/foo");
$subr = $r->lookup_method_uri("GET", "/tmp/bar")
$subr = $r->lookup_file("/tmp/bar");
# optionally manipulate the output through main request filters
$subr = $r->lookup_uri("/foo", $r->output_filters);
# now run them
my $rc = $subr->run;
```

1.2 Description

`Apache2::SubRequest` contains API for creating and running of Apache sub-requests.

`Apache2::SubRequest` is a sub-class of `Apache2::RequestRec object.`

1.3 API

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

1.3.1 *DESTROY*

Free the memory associated with a sub request:

```
undef $subr; # but normally don't do that
```

- **obj: $subr (`Apache2::SubRequest object`)**

  The sub request to finish

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

`DESTROY` is called automatically when `$subr` goes out of scope.

If you want to free the memory earlier than that (for example if you run several subrequests), you can `undef` the object as:

```
undef $subr;
```

but never call `DESTROY` explicitly, since it'll result in `ap_destroy_sub_req` being called more than once, resulting in multiple brain injuries and certain hair loss.

1.3.2 *internal_redirect*

Redirect the current request to some other uri internally

```
$r->internal_redirect($new_uri);
```

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

  The current request

- **arg1: $new_uri (string)**

  The URI to replace the current request with

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

In case that you want some other request to be served as the top-level request instead of what the client requested directly, call this method from a handler, and then immediately return `Apache2::Const::OK`. The client will be unaware the a different request was served to her behind the scenes.

1.3.3 *internal_redirect_handler*

Identical to `internal_redirect`, plus automatically sets $r->content_type is of the sub-request to be the same as of the main request, if $r->handler is true.

```
$r->internal_redirect_handler($new_uri);
```

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

  The current request

- **arg1: $new_uri (string)**

  The URI to replace the current request with.

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

This function is designed for things like actions or CGI scripts, when using `AddHandler`, and you want to preserve the content type across an internal redirect.

1.3.4 `lookup_file`

Create a subrequest for the given file. This sub request can be inspected to find information about the requested file

```
$ret = $r->lookup_file($new_file);
$ret = $r->lookup_file($new_file, $next_filter);
```

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

  The current request

- **arg1: $new_file (string)**

  The file to lookup

- **opt arg2: $next_filter (Apache2::Filter)**

  See $r->lookup_uri for details.

- **ret: $ret (Apache2::SubRequest object)**

  The sub request record.

- **since: 2.0.00**

See $r->lookup_uri for further discussion.

1.3.5 `lookup_method_uri`

Create a sub request for the given URI using a specific method. This sub request can be inspected to find information about the requested URI

```
$ret = $r->lookup_method_uri($method, $new_uri);
$ret = $r->lookup_method_uri($method, $new_uri, $next_filter);
```

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

  The current request

- **arg1: $method (string)**

  The method to use in the new sub request (e.g. "GET")

- **arg2: $new_uri (string)**

  The URI to lookup

- **opt arg3: `$next_filter` (`Apache2::Filter object`)**

  See `$r->lookup_uri` for details.

- **ret: `$ret` (`Apache2::SubRequest object`)**

  The sub request record.

- **since: 2.0.00**

See `$r->lookup_uri` for further discussion.

1.3.6 `lookup_uri`

Create a sub request from the given URI. This sub request can be inspected to find information about the requested URI.

```
$ret = $r->lookup_uri($new_uri);
$ret = $r->lookup_uri($new_uri, $next_filter);
```

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

  The current request

- **arg1: `$new_uri` (string)**

  The URI to lookup

- **opt arg2: `$next_filter` (`Apache2::Filter object`)**

  The first filter the subrequest should pass the data through. If not specified it defaults to the first connection output filter for the main request `$r->proto_output_filters`. So if the subrequest sends any output it will be filtered only once. If for example you desire to apply the main request's output filters to the sub-request output as well pass `$r->output_filters` as an argument.

- **ret: `$ret` (`Apache2::SubRequest object`)**

  The sub request record

- **since: 2.0.00**

Here is an example of a simple subrequest which serves uri */new_uri*:

```
sub handler {
    my $r = shift;

    my $subr = $r->lookup_uri("/new_uri");
    $subr->run;

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

If let's say you have three request output filters registered to run for the main request:

```
PerlOutputFilterHandler MyApache2::SubReqExample::filterA
PerlOutputFilterHandler MyApache2::SubReqExample::filterB
PerlOutputFilterHandler MyApache2::SubReqExample::filterC
```

and you wish to run them all, the code needs to become:

```
my $subr = $r->lookup_uri("/new_uri", $r->output_filters);
```

and if you wish to run them all, but the first one (`filterA`), the code needs to be adjusted to be:

```
my $subr = $r->lookup_uri("/new_uri", $r->output_filters->next);
```

1.3.7 `run`

Run a sub-request

```
$rc = $subr->run();
```

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

  The sub-request (e.g. returned by `lookup_uri`)

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

  The return code of the handler (`Apache2::Const::OK`, `Apache2::Const::DECLINED`, etc.)

- **since: 2.0.00**

1.4 Unsupported API

`Apache2::SubRequest` also provides auto-generated Perl interface for a few other methods which aren't tested at the moment and therefore their API is a subject to change. These methods will be finalized later as a need arises. If you want to rely on any of the following methods please contact the the mod_perl development mailing list so we can help each other take the steps necessary to shift the method to an officially supported API.

1.4.1 `internal_fast_redirect`

META: Autogenerated - needs to be reviewed/completed

Redirect the current request to a sub_req, merging the pools

```
$r->internal_fast_redirect($sub_req);
```

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

The current request

- **arg1: `$sub_req` (string)**

  A subrequest created from this request

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

META: httpd-2.0/modules/http/http_request.c declares this function as:

```
/* XXX: Is this function is so bogus and fragile that we deep-6 it? */
```

do we really want to expose it to mod_perl users?

1.4.2 lookup_dirent

META: Autogenerated - needs to be reviewed/completed

Create a sub request for the given apr_dir_read result. This sub request can be inspected to find information about the requested file

```
$lr = $r->lookup_dirent($finfo);
$lr = $r->lookup_dirent($finfo, $subtype);
$lr = $r->lookup_dirent($finfo, $subtype, $next_filter);
```

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

  The current request

- **arg1: `$finfo` (APR::Finfo object)**

  The apr_dir_read result to lookup

- **arg2: `$subtype` (integer)**

  What type of subrequest to perform, one of;

  ```
  Apache2::SUBREQ_NO_ARGS     ignore r->args and r->path_info
  Apache2::SUBREQ_MERGE_ARGS  merge  r->args and r->path_info
  ```

- **arg3: `$next_filter` (integer)**

  The first filter the sub_request should use. If this is NULL, it defaults to the first filter for the main request

- **ret: `$lr` (Apache2::RequestRec object)**

  The new request record

- **since: 2.0.00**

META: where do we take the apr_dir_read result from?

1.5 See Also

mod_perl 2.0 documentation.

1.6 Copyright

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

1.7 Authors

The mod_perl development team and numerous contributors.

Table of Contents: