1 Apache::RequestUtil - Perl API for Apache request record utils

1.1 Synopsis

```
use Apache::RequestUtil ();

# directory level PerlOptions flags lookup
$r->subprocess_env unless $r->is_perl_option_enabled('SetupEnv');
```

META: to be completed

1.2 Description

META: to be completed

1.3 Functions API

1.3.1 `Apache->request()`

```
$request = Apache->request;
```

1.4 Methods API

1.4.1 `default_type`

META: Autogenerated - needs to be reviewed/completed

Retrieve the value of the DefaultType directive, or text/plain if not set

```
$ret = $r->default_type();
```

- **arg1: `$r` (`Apache::RequestRec`)**

  The current request

- **ret: `$ret` (string)**

  The default type

1.4.2 `document_root`

META: Autogenerated - needs to be reviewed/completed

Retrieve the document root for this server

```
$ret = $r->document_root();
```

- **arg1: `$r` (`Apache::RequestRec`)**

  The current request

- **ret: `$ret` (string)**

  The document root

1.4.3 `get_limit_req_body`

META: Autogenerated - needs to be reviewed/completed

Return the limit on bytes in request msg body

```
  $ret = $r->get_limit_req_body();
```

- **arg1: `$r` (`Apache::RequestRec`)**

  The current request

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

  the maximum number of bytes in the request msg body

1.4.4 `get_server_name`

META: Autogenerated - needs to be reviewed/completed

Get the current server name from the request

```
  $ret = $r->get_server_name();
```

- **arg1: `$r` (`Apache::RequestRec`)**

  The current request

- **ret: `$ret` (string)**

  the server name

1.4.5 `get_server_port`

META: Autogenerated - needs to be reviewed/completed

Get the current server port

```
  $ret = $r->get_server_port();
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **ret: `$ret` (integer)**

  The server's port

1.4.6 *`is_initial_req`*

META: Autogenerated - needs to be reviewed/completed

Determine if the current request is the main request or a sub requests

```
  $ret = $r->is_initial_req();
```

- **arg1: `$r` (`Apache::RequestRec`)**

  The current request

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

1.4.7 *`add_config`*

META: Autogenerated - needs to be reviewed/completed

```
  $ret = $r->add_config($lines, $path, $override);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$lines` (ARRAY ref)**
- **opt arg3: `$path` (scalar)**
- **opt arg4: `$override` (string)**
- **ret: `$ret` (string)**

1.4.8 *`location`*

META: Autogenerated - needs to be reviewed/completed

```
  $location = $r->location($location);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **opt arg2: `$location` (string)**
- **ret: `$location` (integer)**

1.4.9 *`location_merge`*

META: Autogenerated - needs to be reviewed/completed

```
  $ret = $r->location_merge($location);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$location` (string)**
- **ret: `$ret` (integer)**

1.4.10 `pnotes`

META: Autogenerated - needs to be reviewed/completed

Notes from one module to another

```
  $pnotes = $r->pnotes();
  $pnotes = $r->pnotes($new_pnotes);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **opt arg2: `$new_pnotes` (`APR::Table`)**
- **ret: `$pnotes` (`APR::Table`)**

Similar to (`Apache::RequestRec`), but values can be any perl variables. That also means that it can be used only between perl modules.

1.4.11 `no_cache`

META: Autogenerated - needs to be reviewed/completed

```
  $ret = $r->no_cache($flag);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$flag` (number)**
- **ret: `$ret` (integer)**

1.4.12 `as_string`

META: Autogenerated - needs to be reviewed/completed

```
  $string = $r->as_string();
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **ret: `$string` (string)**

1.4.13 `get_handlers`

Returns a reference to a list of handlers enabled for a given phase.

```
@handlers = $r->get_handlers($hook_name);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$hook_name` (string)**

  a string representing the phase to handle.

- **ret: `@handlers` (CODE ref or ref to ARRAY of CODE refs)**

  a list of references to the handler subroutines

For example:

```
@handlers = $r->get_handlers('PerlResponseHandler');
```

1.4.14 `push_handlers`

META: Autogenerated - needs to be reviewed/completed

Add one or more handlers to a list of handlers to be called for a given phase.

```
$r->push_handlers($hook_name => \&handler);
$r->push_handlers($hook_name => [\&handler, \&handler2]);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$hook_name` (string)**

  a string representing the phase to handle.

- **arg3: `$handlers` (CODE ref or ref to ARRAY of CODE refs)**

  a reference to a list of references to the handler subroutines, or a single reference to a handler subroutine

- **ret: no return value**

Examples:

```
$r->push_handlers(PerlResponseHandler => \&handler);
$r->push_handlers(PerlResponseHandler => [\&handler, \&handler2]);

# XXX: not implemented yet
$r->push_handlers(PerlResponseHandler => sub {...});
```

1.4.15 `set_handlers`

META: Autogenerated - needs to be reviewed/completed

Set a list of handlers to be called for a given phase.

```
$r->set_handlers($hook_name => \&handler);
$r->set_handlers($hook_name => [\&handler, \&handler2]);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$hook_name` (string)**

  a string representing the phase to handle.

- **arg3: `$handlers` (CODE ref or ref to ARRAY of CODE refs)**

  a reference to a list of references to the handler subroutines, or a single reference to a handler subroutine

- **ret: no return value**

Examples:

```
$r->set_handlers(PerlResponseHandler => \&handler);
$r->set_handlers(PerlResponseHandler => [\&handler, \&handler2]);

# XXX: not implemented yet
$r->set_handlers(PerlResponseHandler => sub {...});
```

1.4.16 `set_basic_credentials`

META: Autogenerated - needs to be reviewed/completed

```
$r->set_basic_credentials($username, $password);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$username` (string)**
- **arg3: `$password` (string)**
- **ret: no return value**

1.4.17 `slurp_filename`

META: Autogenerated - needs to be reviewed/completed

Return a reference to contents of `$r->filename`.

```
$content = $r->slurp_filename($tainted);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$tainted` (number)**

By default the returned data is tainted (if run under `-T`). If an optional `$tainted` flag is set to zero, the data will be marked as non-tainted. Do not set this flag to zero unless you know what you are doing, you may create a security hole in your program if you do. For more information see the *perlsec* manpage. If you wonder why this option is available, it is used internally by the `ModPerl::Registry` handler and friends, because the CGI scripts that it reads are considered safe (you could just as well `require()` them).

- **ret: `$content` (scalar)**

1.4.18 *`is_perl_option_enabled`*

check whether a directory level PerlOptions flag is enabled or not.

```
$result = $r->is_perl_option_enabled($flag);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **arg2: `$flag` (string)**
- **ret: `$result` (integer)**

For example to check whether the `SetupEnv` option is enabled for the current request (which can be disabled with `PerlOptions -SetupEnv`) and populate the environment variables table if disabled:

```
$r->subprocess_env unless $r->is_perl_option_enabled('SetupEnv');
```

See also: PerlOptions and the equivalent function for server level PerlOptions flags.

1.4.19 *`dir_config`*

dir_config() provides an interface for the per-directory variable specified by the `PerlSetVar` and `PerlAddVar` directives, and also can be manipulated via the `APR::Table` methods.

```
$table  = $r->dir_config();
$value  = $r->dir_config($key);
@values = $r->dir_config($key);
$r->dir_config($key, $val);
```

- **arg1: `$r` (`Apache::RequestRec`)**
- **opt arg2: `$key` (string)**
- **opt arg3: `$val` (string)**
- **ret: `$ret` (scalar)**

  Depends on the passed arguments, see further discussion

The keys are case-insensitive.

```
$apr_table = $r->dir_config();
```

dir_config() called in a scalar context without the `$key` argument returns a *HASH* reference blessed into the `APR::Table` class. This object can be manipulated via the `APR::Table` methods. For available methods see the `APR::Table` manpage.

```
@values = $r->dir_config($key);
```

If the `$key` argument is passed in the list context a list of all matching values will be returned. This method is ineffective for big tables, as it does a linear search of the table. Thefore avoid using this way of calling dir_config() unless you know that there could be more than one value for the wanted key and all the values are wanted.

```
$value = $r->dir_config($key);
```

If the `$key` argument is passed in the scalar context only a single value will be returned. Since the table preserves the insertion order, if there is more than one value for the same key, the oldest value assosiated with the desired key is returned. Calling in the scalar context is also much faster, as it'll stop searching the table as soon as the first match happens.

```
$r->dir_config($key => $val);
```

If the `$key` and the `$val` arguments are used, the set() operation will happen: all existing values associated with the key `$key` (and the key itself) will be deleted and `$value` will be placed instead.

```
$r->dir_config($key => undef);
```

If `$val` is *undef* the unset() operation will happen: all existing values associated with the key `$key` (and the key itself) will be deleted.

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 1.1.

1.7 Authors

The mod_perl development team and numerous contributors.

Table of Contents: