1 Apache::ServerUtil - Perl API for XXX

1.1 Synopsis

```
use Apache::ServerUtil ();

$s = Apache->server;
my $srv_cfg = $s->dir_config;

# get 'conf/' dir path using $s
my $conf_dir = $s->server_root_relative('conf');

# server level PerlOptions flags lookup
$s->push_handlers(ChildExit => \&child_exit)
    if $s->is_perl_option_enabled('ChildExit');
```

META: to be completed

1.2 Description

`Apache::ServerUtil` provides the Perl API for Apache server object.

META: to be completed

1.3 Constants

1.3.1 `Apache::Server::server_root`

returns the value set by the `ServerRoot` directive.

1.4 Functions API

1.4.1 `add_version_component`

META: Autogenerated - needs to be reviewed/completed

Add a component to the version string

```
add_version_component($pconf_pool, $component);
```

- **arg1: `$pconf` (`APR::Pool`)**

  The pool to allocate the component from (should really be a `$pconf_pool`)

- **arg2: `$component` (string)**

  The string to add

- **ret: no return value**

1.4.2 `exists_config_define`

Check for a definition from the server command line

```
$result = Apache::Server::exists_config_define($name);
```

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

  The define to check for

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

  true if defined, false otherwise

For example:

```
print "this is mp2" if Apache::Server::exists_config_define('MODPERL2');
```

1.4.3 `get_server_built`

META: Autogenerated - needs to be reviewed/completed

Get the date and time that the server was built

```
$when_built = Apache::Server::get_server_built();
```

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

  The server build time string

1.4.4 `get_server_version`

Get the server version string

```
Apache::Server::get_server_version();
```

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

  The server version string

1.5 Methods API

`Apache::ServerUtil` provides the following functions and/or methods:

1.5.1 server_root_relative()

Returns the canonical form of the filename made absolute to `ServerRoot`:

```
$path = $s->server_root_relative($fname);
```

- **arg1: `$s` (`Apache::Server`)**
- **opt arg2: `$fname` (string)**
- **ret: `$path` (string)**

`$fname` is appended to the value of `ServerRoot` and returned. For example:

```
my $log_dir = Apache::Server::server_root_relative($r->pool, 'logs');
```

If `$fname` is not specified, the value of `ServerRoot` is returned with a trailing `/`. (it's the same as using `''` as `$fname`'s value).

Also see the `Apache::Server::server_root` constant.

1.5.2 error_log2stderr

META: Autogenerated - needs to be reviewed/completed

Convert stderr to the error log

```
$s->error_log2stderr();
```

- **arg1: `$s` (`Apache::Server`)**

  The current server

- **ret: no return value**

1.5.3 psignature

META: Autogenerated - needs to be reviewed/completed

Get HTML describing the address and (optionally) admin of the server.

```
$sig = $r->psignature($prefix);
```

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

  Text which is prepended to the return value

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

HTML describing the server

1.5.4 dir_config

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

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

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

Depends on the passed arguments, see further discussion

The keys are case-insensitive.

```
$t = $s->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 *APR::Table*.

```
@values = $s->dir_config($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 = $s->dir_config($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.

```
$s->dir_config($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.

```
$s->dir_config($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.

1.5.5 is_perl_option_enabled

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

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

- **arg1: `$s` (`Apache::Server`)**
- **arg2: `$flag` (string)**
- **ret: `$result` (integer)**

For example to check whether the `ChildExit` hook is enabled (which can be disabled with `PerlOptions -ChildExit`) and configure some handlers to run if enabled:

```
$s->push_handlers(ChildExit => \&child_exit)
    if $s->is_perl_option_enabled('ChildExit');
```

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

1.5.6 get_handlers

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

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

- **arg1: `$s` (`Apache::Server`)**
- **arg2: `$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 = $s->get_handlers('PerlResponseHandler');
```

1.5.7 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.

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

- **arg1: `$s` (`Apache::Server`)**
- **arg2: `$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:

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

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

1.5.8 `set_handlers`

META: Autogenerated - needs to be reviewed/completed

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

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

- **arg1: `$s` (`Apache::Server`)**
- **arg2: `$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:

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

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

1.5.9 method_register

META: Autogenerated - needs to be reviewed/completed

Register a new request method, and return the offset that will be associated with that method.

```
$ret = $p->method_register($methname);
```

- **arg1: `$p` (`APR::Pool`)**

  The pool to create registered method numbers from.

- **arg2: `$methname` (string)**

  The name of the new method to register.

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

  Ab int value representing an offset into a bitmask.

1.5.10 get_status_line

META: Autogenerated - needs to be reviewed/completed

Return the Status-Line for a given status code (excluding the HTTP-Version field). If an invalid or unknown status code is passed, "500 Internal Server Error" will be returned.

```
$ret = get_status_line($status);
```

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

  The HTTP status code

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

  The Status-Line

1.5.11 server

Get the main server's object

```
$main_s = Apache->server();
```

- **arg1: `Apache` (class name)**
- **ret: `$main_s` (`Apache::Server`)**

1.6 See Also

mod_perl 2.0 documentation.

1.7 Copyright

1.8 Authors

The mod_perl development team and numerous contributors.

Table of Contents: