1 Apache::ServerUtil -- Methods for work with Apache::Server object

1 Apache::ServerUtil -- Methods for work with Apache::Server object

1.1 SYNOPSIS

```
use Apache::ServerUtil;

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

# get 'conf/' dir path using $r
my $conf_dir = Apache::server_root_relative('conf', $r->pool);

# get 'log/' dir path using default server startup pool
my $log_dir = Apache::server_root_relative('log');
```

1.2 DESCRIPTION

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

META: complete

1.3 API

Function arguments (if any) and return values are shown in the function's synopsis.

1.3.1 CONSTANTS

- **server_root**

  returns the value set by the `ServerRoot` directive.

1.3.2 FUNCTIONS

- **server_root_relative()**

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

  ```
  Apache::server_root_relative($pool, $fname);
  ```

  `$fname` is appended to the value of `ServerRoot` and return it. e.g.:

  ```
  my $log_dir = Apache::server_root_relative($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 `server_root` constant.

1.3.3 METHODS

- **server()**

  The main server's object can be retrieved with:

  ```
  $s = Apache->server;
  ```

  Gets the `Apache::Server` object for the main server.

- **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.

  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.

- **push_handlers()**

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

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

- **add_handlers()**
- **get_handlers()**

Table of Contents: