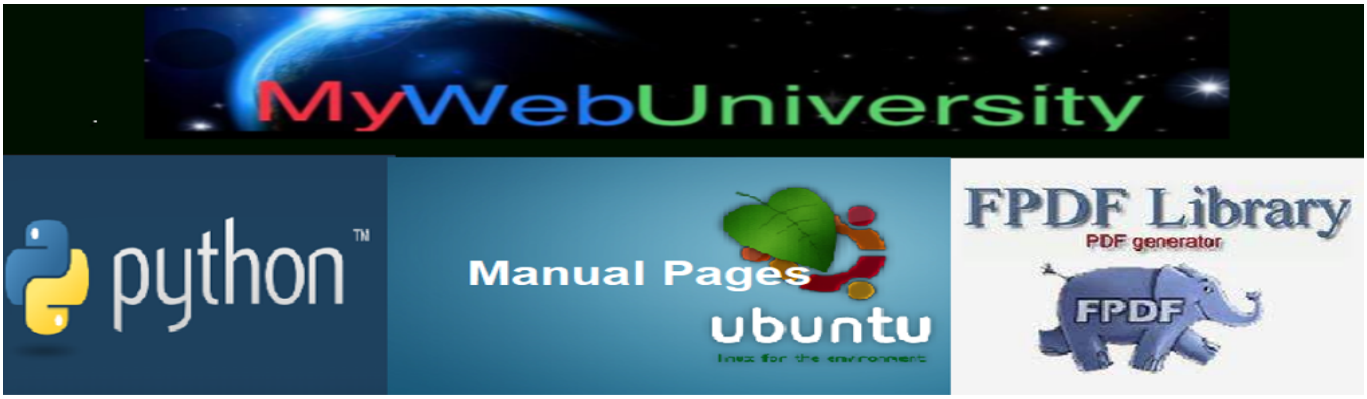

Rocky Enterprise Linux 9.2 Manual Pages on command 'Apache::TestConfig.3pm'

$ man Apache::TestConfig.3pm

Apache::TestConfig(3pm) User Contributed Perl Documentation Apache::TestConfig(3pm)

NAME

Apache::TestConfig -- Test Configuration setup module

SYNOPSIS

```
use Apache::TestConfig;

my $cfg = Apache::TestConfig->new(%args)
my $fh = $cfg->genfile($file);
$cfg->writefile($file, $content);
$cfg->gendir($dir);
...
```

DESCRIPTION

"Apache::TestConfig" is used in creating the "Apache::Test" configuration files.

FUNCTIONS

genwarning()

```
my $warn = $cfg->genwarning($filename)
```

genwarning() returns a warning string as a comment, saying that the file was autogenerated and that it's not a good idea to modify this file. After the warning a

perl trace of calls to this this function is appended. This trace is useful for finding what code has created the file.

```
my $warn = $cfg->genwarning($filename, $from_filename)
```

If $from_filename is specified it'll be used in the warning to tell which file it was generated from.

genwarning() automatically recognizes the comment type based on the file extension. If the extension is not recognized, the default "#" style is used.

Currently it support "<!-- -->", "/* ... */" and "#" styles.

genfile()

```
my $fh = $cfg->genfile($file);
```

genfile() creates a new file $file for writing and returns a file handle.

If parent directories of $file don't exist they will be automagically created.

The file $file and any created parent directories (if found empty) will be automatically removed on cleanup.

A comment with a warning and calls trace is added to the top of this file. See genwarning() for more info about this comment.

```
my $fh = $cfg->genfile($file, $from_file);
```

If $from_filename is specified it'll be used in the warning to tell which file it was generated from.

```
my $fh = $cfg->genfile($file, $from_file, $nowarning);
```

If $nowarning is true, the warning won't be added. If using this optional argument and there is no $from_file you must pass undef as in:

```
my $fh = $cfg->genfile($file, undef, $nowarning);
```

writefile()

```
$cfg->writefile($file, $content, [$nowarning]);
```

writefile() creates a new file $file with the content of $content.

A comment with a warning and calls trace is added to the top of this file unless $nowarnings is passed and set to a true value. See genwarning() for more info about this comment.

If parent directories of $file don't exist they will be automagically created.

The file $file and any created parent directories (if found empty) will be automatically removed on cleanup.

write_perlscript()

```
$cfg->write_perlscript($filename, @lines);
```

Similar to writefile() but creates an executable Perl script with correctly set shebang line.

gendir()

```
$cfg->gendir($dir);
```

gendir() creates a new directory $dir.

If parent directories of $dir don't exist they will be automagically created.

The directory $dir and any created parent directories will be automatically removed on

cleanup if found empty.

Environment Variables

The following environment variables affect the configuration and the run-time of the "Apache::Test" framework:

APACHE_TEST_COLOR

To aid visual control over the configuration process and the run-time phase, "Apache::Test" uses coloured fonts when the environment variable "APACHE_TEST_COLOR" is set to a true value.

APACHE_TEST_LIVE_DEV

When using "Apache::Test" during the project development phase, it's often convenient to have the project/lib (live) directory appearing first in @INC so any changes to the Perl modules, residing in it, immediately affect the server, without a need to rerun "make" to update blib/lib. When the environment variable "APACHE_TEST_LIVE_DEV" is set to a true value during the configuration phase ("t/TEST -config", "Apache::Test" will automatically unshift the project/lib directory into @INC, via the autogenerated t/conf/modperl_inc.pl file.

Special Placeholders

When generating configuration files from the *.in templates, special placeholder variables get substituted. To embed a placeholder use the "@foo@" syntax. For example in extra.conf.in you can write:

```
Include @ServerRoot@/conf/myconfig.conf
```

When extra.conf is generated, "@ServerRoot@" will get replaced with the location of the server root.

Placeholders are case-insensitive.

Available placeholders:

Configuration Options

All configuration variables that can be passed to "t/TEST", such as "MaxClients", "DocumentRoot", "ServerRoot", etc. To see the complete list run:

```
% t/TEST --help
```

and you will find them in the "configuration options" sections.

NextAvailablePort

Every time this placeholder is encountered it'll be replaced with the next available port. This is very useful if you need to allocate a special port, but not hardcode it. Later when running:

```
% t/TEST -port=select
```

it's possible to run several concurrent test suites on the same machine, w/o having port collisions.

AUTHOR

SEE ALSO

perl(1), Apache::Test(3)