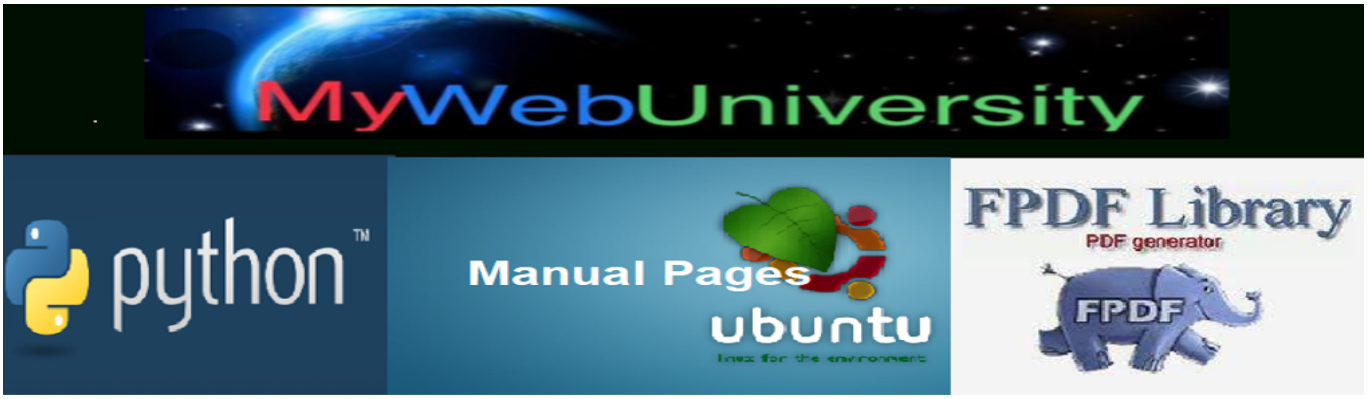

Rocky Enterprise Linux 9.2 Manual Pages on command 'TAP::Parser::Source.3perl'

$ man TAP::Parser::Source.3perl

TAP::Parser::Source(3perl) Perl Programmers Reference Guide TAP::Parser::Source(3perl)

NAME

TAP::Parser::Source - a TAP source & meta data about it

VERSION

Version 3.43

SYNOPSIS

```
use TAP::Parser::Source;
my $source = TAP::Parser::Source->new;
$source->raw( \'reference to raw TAP source' )
    ->config( \%config )
    ->merge( $boolean )
    ->switches( \@switches )
    ->test_args( \@args )
    ->assemble_meta;
do { ... } if $source->meta->{is_file};
# see assemble_meta for a full list of data available
```

DESCRIPTION

A TAP source is something that produces a stream of TAP for the parser to consume, such as an executable file, a text file, an archive, an IO handle, a database, etc.

"TAP::Parser::Source"s encapsulate these raw sources, and provide some useful meta data about them. They are used by TAP::Parser::SourceHandlers, which do whatever is required to produce & capture a stream of TAP from the raw source, and package it up in a TAP::Parser::Iterator for the parser to consume.

Unless you're writing a new TAP::Parser::SourceHandler, a plugin or subclassing TAP::Parser, you probably won't need to use this module directly.

METHODS

Class Methods

"new"

```
my $source = TAP::Parser::Source->new;
```

Returns a new "TAP::Parser::Source" object.

Instance Methods

"raw"

```
my $raw = $source->raw;
$source->raw( $some_value );
```

Chaining getter/setter for the raw TAP source. This is a reference, as it may contain large amounts of data (eg: raw TAP).

"meta"

```
my $meta = $source->meta;
$source->meta({ %some_value });
```

Chaining getter/setter for meta data about the source. This defaults to an empty hashref. See "assemble_meta" for more info.

"has_meta"

True if the source has meta data.

"config"

```
my $config = $source->config;
$source->config({ %some_value });
```

Chaining getter/setter for the source's configuration, if any has been provided by the user. How it's used is up to you. This defaults to an empty hashref. See "config_for" for more info.

"merge"

```
my $merge = $source->merge;
$source->config( $bool );
```

Chaining getter/setter for the flag that dictates whether STDOUT and STDERR should be merged (where appropriate). Defaults to undef.

"switches"

```
my $switches = $source->switches;
```

```
  $source->config([ @switches ]);
```

Chaining getter/setter for the list of command-line switches that should be passed to the source (where appropriate). Defaults to undef.

"test_args"

```
  my $test_args = $source->test_args;
  $source->config([ @test_args ]);
```

Chaining getter/setter for the list of command-line arguments that should be passed to the source (where appropriate). Defaults to undef.

"assemble_meta"

```
  my $meta = $source->assemble_meta;
```

Gathers meta data about the "raw" source, stashes it in "meta" and returns it as a hashref. This is done so that the TAP::Parser::SourceHandlers don't have to repeat common checks. Currently this includes:

```
    is_scalar => $bool,
    is_hash   => $bool,
    is_array  => $bool,
    # for scalars:
    length => $n
    has_newlines => $bool
    # only done if the scalar looks like a filename
    is_file => $bool,
    is_dir  => $bool,
    is_symlink => $bool,
    file => {
        # only done if the scalar looks like a filename
        basename => $string, # including ext
        dir      => $string,
        ext      => $string,
        lc_ext   => $string,
        # system checks
        exists  => $bool,
        stat    => [ ... ], # perldoc -f stat
        empty   => $bool,
```

```
        size    => $n,
        text    => $bool,
        binary  => $bool,
        read    => $bool,
        write   => $bool,
        execute => $bool,
        setuid  => $bool,
        setgid  => $bool,
        sticky  => $bool,
        is_file => $bool,
        is_dir  => $bool,
        is_symlink => $bool,
        # only done if the file's a symlink
        lstat      => [ ... ], # perldoc -f lstat
        # only done if the file's a readable text file
        shebang => $first_line,
      }
    # for arrays:
    size => $n,
```

"shebang"

Get the shebang line for a script file.

```
  my $shebang = TAP::Parser::Source->shebang( $some_script );
```

May be called as a class method

"config_for"

```
  my $config = $source->config_for( $class );
```

Returns "config" for the $class given. Class names may be fully qualified or abbreviated, eg:

```
  # these are equivalent
  $source->config_for( 'Perl' );
  $source->config_for( 'TAP::Parser::SourceHandler::Perl' );
```

If a fully qualified $class is given, its abbreviated version is checked first.

AUTHORS

Steve Purkis.

SEE ALSO

TAP::Object, TAP::Parser, TAP::Parser::IteratorFactory, TAP::Parser::SourceHandler