NAME

TAP::Parser::Source - Stream output from some source

VERSION

Version 3.17

SYNOPSIS

```
use TAP::Parser::Source;
my $source = TAP::Parser::Source->new;
my $stream = $source->source(['/usr/bin/ruby', 'mytest.rb'])->get_stream;
```

DESCRIPTION

Takes a command and hopefully returns a stream from it.

METHODS

Class Methods

new

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

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

Instance Methods

source

```
my $source = $source->source;
$source->source(['./some_prog some_test_file']);

# or
$source->source(['/usr/bin/ruby', 't/ruby_test.rb']);
```

Getter/setter for the source. The source should generally consist of an array reference of strings which, when executed via *&IPC::Open3::open3*, should return a filehandle which returns successive rows of TAP. `croaks` if it doesn't get an arrayref.

get_stream

```
my $stream = $source->get_stream;
```

Returns a *TAP::Parser::Iterator* stream of the output generated by executing `source`. `croaks` if there was no command found.

Must be passed an object that implements a `make_iterator` method. Typically this is a TAP::Parser instance.

merge

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

Sets or returns the flag that dictates whether STDOUT and STDERR are merged.

SUBCLASSING

Please see *"SUBCLASSING" in TAP::Parser* for a subclassing overview.

Example

```
package MyRubySource;
```

```
use strict;
use vars '@ISA';

use Carp qw( croak );
use TAP::Parser::Source;

@ISA = qw( TAP::Parser::Source );

# expect $source->(['mytest.rb', 'cmdline', 'args']);
sub source {
  my ($self, $args) = @_;
  my ($rb_file) = @$args;
  croak("error: Ruby file '$rb_file' not found!") unless (-f $rb_file);
  return $self->SUPER::source(['/usr/bin/ruby', @$args]);
}
```

SEE ALSO

TAP::Object, *TAP::Parser*, *TAP::Parser::Source::Perl*,