

NAME

TAP::Parser::SourceHandler - Base class for different TAP source handlers

VERSION

Version 3.36

SYNOPSIS

```
# abstract class - don't use directly!
# see TAP::Parser::IteratorFactory for general usage

# must be sub-classed for use
package MySourceHandler;
use base 'TAP::Parser::SourceHandler';
sub can_handle { return $confidence_level }
sub make_iterator { return $iterator }

# see example below for more details
```

DESCRIPTION

This is an abstract base class for [TAP::Parser::Source](#) handlers / handlers.

A [TAP::Parser::SourceHandler](#) does whatever is necessary 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.

SourceHandlers must implement the *source detection & handling* interface used by [TAP::Parser::IteratorFactory](#). At 2 methods, the interface is pretty simple: “can_handle” and “make_source”.

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**

can_handle

Abstract method.

```
my $vote = $class->can_handle( $source );
```

\$source is a [TAP::Parser::Source](#).

Returns a number between 0 & 1 reflecting how confidently the raw source can be handled. For example, 0 means the source cannot handle it, 0.5 means it may be able to, and 1 means it definitely can. See “detect_source” in [TAP::Parser::IteratorFactory](#) for details on how this is used.

make_iterator

Abstract method.

```
my $iterator = $class->make_iterator( $source );
```

\$source is a [TAP::Parser::Source](#).

Returns a new [TAP::Parser::Iterator](#) object for use by the [TAP::Parser](#). croaks on error.

SUBCLASSING

Please see “SUBCLASSING” in [TAP::Parser](#) for a subclassing overview, and any of the subclasses that ship with this module as an example. What follows is a quick overview.

Start by familiarizing yourself with [TAP::Parser::Source](#) and [TAP::Parser::IteratorFactory](#). [TAP::Parser::SourceHandler::RawTAP](#) is the easiest sub-class to use as an example.

It’s important to point out that if you want your subclass to be automatically used by [TAP::Parser](#) you’ll have to and make sure it gets loaded somehow. If you’re using [prove](#) you can write an [App::Prove](#) plugin. If you’re using [TAP::Parser](#) or [TAP::Harness](#) directly (e.g. through a custom script, [ExtUtils::MakeMaker](#), or [Module::Build](#)) you can use the `config` option which will cause “load_sources” in

[TAP::Parser::IteratorFactory](#) to load your subclass).

Don't forget to register your class with "register_handler" in TAP::Parser::IteratorFactory.

Example

```
package MySourceHandler;

use strict;

use MySourceHandler; # see TAP::Parser::SourceHandler
use TAP::Parser::IteratorFactory;

use base 'TAP::Parser::SourceHandler';

TAP::Parser::IteratorFactory->register_handler( __PACKAGE__ );

sub can_handle {
    my ( $class, $src ) = @_;
    my $meta = $src->meta;
    my $config = $src->config_for( $class );

    if ( $config->{accept_all} ) {
        return 1.0;
    } elsif ( my $file = $meta->{file} ) {
        return 0.0 unless $file->{exists};
        return 1.0 if $file->{lc_ext} eq '.tap';
        return 0.9 if $file->{shebang} && $file->{shebang} =~ /^#!.+tap/;
        return 0.5 if $file->{text};
        return 0.1 if $file->{binary};
    } elsif ( $meta->{scalar} ) {
        return 0.8 if $$raw_source_ref =~ /\d\.\.\d/;
        return 0.6 if $meta->{has_newlines};
    } elsif ( $meta->{array} ) {
        return 0.8 if $meta->{size} < 5;
        return 0.6 if $raw_source_ref->[0] =~ /foo/;
        return 0.5;
    } elsif ( $meta->{hash} ) {
        return 0.6 if $raw_source_ref->{foo};
        return 0.2;
    }

    return 0;
}

sub make_iterator {
    my ( $class, $source ) = @_;
    # this is where you manipulate the source and
    # capture the stream of TAP in an iterator
    # either pick a TAP::Parser::Iterator::* or write your own...
    my $iterator = TAP::Parser::Iterator::Array->new([ 'foo', 'bar' ]);
    return $iterator;
}

1;
```

AUTHORS

TAPx Developers.

Source detection stuff added by Steve Purkis

SEE ALSO

[TAP::Object](#), [TAP::Parser](#), [TAP::Parser::Source](#), [TAP::Parser::Iterator](#), [TAP::Parser::IteratorFactory](#),
[TAP::Parser::SourceHandler::Executable](#), [TAP::Parser::SourceHandler::Perl](#),
[TAP::Parser::SourceHandler::File](#), [TAP::Parser::SourceHandler::Handle](#),
[TAP::Parser::SourceHandler::RawTAP](#)