NAME

TAP::Parser::Iterator - Base class for TAP source iterators

VERSION

Version 3.36

SYNOPSIS

```
 # to subclass:
 use TAP::Parser::Iterator ();
 use base 'TAP::Parser::Iterator';
 sub _initialize {
 # see TAP::Object...
 }

 sub next_raw { ... }
 sub wait { ... }
 sub exit { ... }
```

DESCRIPTION

This is a simple iterator base class that defines TAP::Parser's iterator API. Iterators are typically created from TAP::Parser::SourceHandlers.

METHODS

Class Methods

new

Create an iterator. Provided by TAP::Object.

Instance Methods

next

```
 while ( my $item = $iter->next ) { ... }
```

Iterate through it, of course.

next_raw

Note: this method is abstract and should be overridden.

```
 while ( my $item = $iter->next_raw ) { ... }
```

Iterate raw input without applying any fixes for quirky input syntax.

handle_unicode

If necessary switch the input stream to handle unicode. This only has any effect for I/O handle based streams.

The default implementation does nothing.

get_select_handles

Return a list of filehandles that may be used upstream in a *select()* call to signal that this Iterator is ready. Iterators that are not handle-based should return an empty list.

The default implementation does nothing.

wait

Note: this method is abstract and should be overridden.

```
 my $wait_status = $iter->wait;
```

Return the `wait` status for this iterator.

exit

Note: this method is abstract and should be overridden.

```
my $wait_status = $iter->exit;
```

Return the `exit` status for this iterator.

SUBCLASSING

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

You must override the abstract methods as noted above.

Example

TAP::Parser::Iterator::Array is probably the easiest example to follow. There's not much point repeating it here.

SEE ALSO

TAP::Object, TAP::Parser, TAP::Parser::Iterator::Array, TAP::Parser::Iterator::Stream, TAP::Parser::Iterator::Process,