NAME

TAP::Parser::Iterator - Internal base class for TAP::Parser Iterators

VERSION

Version 3.17

SYNOPSIS

```
  # see TAP::Parser::IteratorFactory for general usage

  # to subclass:
  use vars qw(@ISA);
  use TAP::Parser::Iterator ();
  @ISA = qw(TAP::Parser::Iterator);
  sub _initialize {
    # see TAP::Object...
  }
```

DESCRIPTION

This is a simple iterator base class that defines *TAP::Parser*'s iterator API. See `TAP::Parser::IteratorFactory` for the preferred way of creating iterators.

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::IteratorFactory*, *TAP::Parser::Iterator::Array*, *TAP::Parser::Iterator::Stream*, *TAP::Parser::Iterator::Process*,