NAME

      "IO::Async::Timer" − base class for Notifiers that use timed delays

DESCRIPTION

      This module provides a subclass of IO::Async::Notifier for implementing notifiers that use timed delays. For specific implementations, see one of the subclasses:

- IO::Async::Timer::Absolute − event callback at a fixed future time

- IO::Async::Timer::Countdown − event callback after a fixed delay

- IO::Async::Timer::Periodic − event callback at regular intervals

CONSTRUCTOR

  new

```
$timer = IO::Async::Timer->new( %args )
```

      Constructs a particular subclass of `IO::Async::Timer` object, and returns it. This constructor is provided for backward compatibility to older code which doesn't use the subclasses. New code should directly construct a subclass instead.

      mode => STRING

            The type of timer to create. Currently the only allowed mode is `countdown` but more types may be added in the future.

      Once constructed, the `Timer` will need to be added to the `Loop` before it will work. It will also need to be started by the `start` method.

METHODS

  is_running

```
$running = $timer->is_running
```

      Returns true if the Timer has been started, and has not yet expired, or been stopped.

  start

```
$timer->start
```

      Starts the Timer. Throws an error if it was already running.

      If the Timer is not yet in a Loop, the actual start will be deferred until it is added. Once added, it will be running, and will expire at the given duration after the time it was added.

      As a convenience, $timer is returned. This may be useful for starting timers at construction time:

```
 $loop->add( IO::Async::Timer->new( ... )->start );
```

  stop

```
$timer->stop
```

      Stops the Timer if it is running. If it has not yet been added to the `Loop` but there is a start pending, this will cancel it.

AUTHOR

      Paul Evans <leonerd@leonerd.org.uk>