NAME

"Future::Exception" − an exception type for failed Futures

SYNOPSIS

```
use Scalar::Util qw( blessed );
use Syntax::Keyword::Try;

try {
   my $f = ...;
   my @result = $f->get;
   ...
}
catch {
   if( blessed($@) and $@->isa( "Future::Exception" ) {
      print STDERR "The ", $@->category, " failed: ", $@->message, "\n";
   }
}
```

DESCRIPTION

The `get` method on a failed Future instance will throw an exception to indicate that the future failed. A failed future can contain a failure category name and other details as well as the failure message, so in this case the exception will be an instance of `Future::Exception` to make these values accessible.

Users should not depend on exact class name matches, but instead rely on inheritence, as a later version of this implementation might dynamically create subclasses whose names are derived from the Future failure category string, to assist with type matching. Note the use of `->isa` in the SYNOPSIS example.

CONSTRUCTOR

from_future

```
$e = Future::Exception->from_future( $f )
```

Constructs a new `Future::Exception` wrapping the given failed future.

ACCESSORS

```
$message  = $e->message
$category = $e->category
@details  = $e->details
```

Additionally, the object will stringify to return the message value, for the common use-case of printing, regexp testing, or other behaviours.

METHODS

throw

```
Future::Exception->throw( $message, $category, @details )
```

Since version 0.41.

Constructs a new exception object and throws it using `die()`. This method will not return, as it raises the exception directly.

If `$message` does not end in a linefeed then the calling file and line number are appended to it, in the same way `die()` does.

as_future

```
$f = $e->as_future
```

Returns a new `Future` object in a failed state matching the exception.

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>