Class

DateIntervalFormatter

A formatter that creates string representations of time intervals.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.1+ | macOS 10.10+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
class DateIntervalFormatter
```

Overview

A `DateIntervalFormatter` object creates user-readable strings from pairs of dates. Use a date interval formatter to create user-readable strings of the form *<start>* – *<end>* for your app's interface, where *<start>* and *<end>* are date values that you supply. The formatter uses locale and language information, along with custom formatting options, to define the content of the resulting string. You can specify different styles for the date and time information in each date value.

To use this class, create an instance, configure its properties, and call the `string(from:to:)` method to generate a string. The properties of this class let you configure the calendar and specify the style to apply to date and time values. Given a current date of January 16, 2015, Configuring the Formatter Options shows how to configure a formatter object and generate the string "1/16/15 - 1/17/15".

Configuring a formatter object

Swift | Objective-C

```
let formatter = DateIntervalFormatter()
formatter.dateStyle = .short
formatter.timeStyle = .none

// Create two dates that are exactly 1 day apart.
```

```
let startDate = Date()
let endDate = Date(timeInterval: 86400, since: startDate)

// Use the configured formatter to generate the string.
let outputString = formatter.string(from: startDate, to: endDate)
```

> **Note**
>
> Always set to the `dateStyle` and `timeStyle` properties to appropriate values before generating any strings.

The `string(from:to:)` method may be called safely from any thread of your app. It is also safe to share a single instance of this class from multiple threads, with the caveat that you should not change the configuration of the object while another thread is using it to generate a string.

> **Tip**
>
> In Swift, you can use `Date.IntervalFormatStyle` rather than `DateIntervalFormatter`. The `FormatStyle` API offers a declarative idiom for customizing the formatting of various types. Also, Foundation caches identical `FormatStyle` instances, so you don't need to pass them around your app, or risk wasting memory with duplicate formatters.

Topics

Formatting a String

`func string(from: Date, to: Date) -> String`

Returns a formatted string based on the specified start and end dates.

Configuring the Formatter Options

`var dateStyle: DateIntervalFormatter.Style`

The style to use when formatting day, month, and year information.

`var timeStyle: DateIntervalFormatter.Style`

The style to use when formatting hour, minute, and second information.

`var dateTemplate: String!`

The template for formatting one date and time value.

`var calendar: Calendar!`

The calendar to use for date values.

`var locale: Locale!`

The locale to use when formatting date and time values.

`var timeZone: TimeZone!`

The time zone with which to specify time values.

Constants

`enum Style`

Formatting styles for individual date and time values.

Instance Methods

`func string(from: DateInterval) -> String?`

Relationships

Inherits From

`Formatter`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCoding`
`NSCopying`
`NSObjectProtocol`
`Sendable`
`SendableMetatype`

See Also

Dates and times

`class DateFormatter`

A formatter that converts between dates and their textual representations.

`class DateComponentsFormatter`

A formatter that creates string representations of quantities of time.

`class RelativeDateTimeFormatter`

A formatter that creates locale-aware string representations of a relative date or time.

`class ISO8601DateFormatter`

A formatter that converts between dates and their ISO 8601 string representations.