Structure

URL.FormatStyle

A structure that converts between URL instances and their textual representations.

iOS 16.0+ | iPadOS 16.0+ | Mac Catalyst 16.0+ | macOS 13.0+ | tvOS 16.0+ | visionOS 1.0+ | watchOS 9.0+

```
struct FormatStyle
```

Overview

Instances of `URL.FormatStyle` create localized, human-readable text from `URL` instances and parse string representations of URLs into instances of `URL`.

Formatting URLs

Use the `formatted()` method to create a string representation of a URL using the default `URL.FormatStyle` configuration. As seen in the following example, the default style creates a string with the scheme, host, and path, but not the port or query.

```
let url = URL(string:"https://www.example.com:8080/path/to/endpoint?key=value")!
let formatted = url.formatted() // "https://www.example.com/path/to/endpoint"
```

You can specify a format style by providing an argument to the `format(_:)` method. The following example uses the previous URL, but preserves only the host and path.

```
let url = URL(string:"https://www.example.com:8080/path/to/endpoint?key=value")!
let style = URL.FormatStyle(scheme: .never,
                            user: .never,
```

```
                                password: .never,
                                host: .always,
                                port: .never,
                                path: .always,
                                query: .never,
                                fragment: .never)
let formatted = style.format(url) // "www.example.com/path/to/endpoint"
```

Instantiate a style when you want to format multiple URL instances with the same style. For one-time access to a default style, you can use the static accessor `url` at call points that expect the `URL.FormatStyle` type, such as the `format(_:)` method. This means you can write the example above as follows:

```
let url = URL(string:"https://www.example.com:8080/path/to/endpoint?key=value")!
let formatted = url.formatted(.url
    .scheme(.never)
    .host(.always)
    .port(.never)
    .path(.always)
    .query(.never)) // "www.example.com/path/to/endpoint"
```

This example works by taking the default style provided by `url`, then customizing it with calls to the style modifiers in Customizing style behavior.

Parsing URLs

You can use `URL.FormatStyle` to parse strings into URL values. To do this, create a `URL.ParseStrategy` from a format style, then call the strategy's `parse(_:)` method.

```
let style = URL.FormatStyle(scheme: .always,
                                user: .never,
                                password: .never,
                                host: .always,
                                port: .always,
                                path: .always,
                                query: .always,
                                fragment: .never)
let urlString = "https://www.example.com:8080/path/to/endpoint?key=value"
let url = try? style.parseStrategy.parse(urlString)
```

Matching regular expressions

Along with parsing URL values in strings, you can use the regular expression domain-specific language provided by Swift to match and capture URL substrings. The following example scans source input that's expected to contain a timestamp, some whitespace, and a URL.

```
import RegexBuilder
let source = "7/31/2022, 5:15:12 AM  https://www.example.com/productList?query=slush
let matcher = Regex {
    One(.dateTime(date: .numeric,
                  time: .standard,
                  locale: Locale(identifier: "en_US"),
                  timeZone: TimeZone(identifier: "PST")!))
    OneOrMore(.horizontalWhitespace)
    Capture {
        One(.url(scheme: .required,
                 user: .optional,
                 password: .optional,
                 host: .required,
                 port: .defaultValue(8088),
                 path: .optional,
                 query: .optional,
                 fragment: .optional))
    }
}
guard let match = source.firstMatch(of: matcher) else { return }
let url = match.1 // url = https://www.example.com:8088/productList?query=slushie
```

Topics

Creating a URL format style

`init(scheme: URL.FormatStyle.ComponentDisplayOption, user: URL.FormatStyle.ComponentDisplayOption, password: URL.FormatStyle.ComponentDisplayOption, host: URL.FormatStyle.HostDisplayOption, port: URL.FormatStyle.ComponentDisplayOption, path: URL.FormatStyle.ComponentDisplayOption, query: URL.FormatStyle.ComponentDisplayOption, fragment: URL.FormatStyle.ComponentDisplayOption)`

Creates a URL format style with the given display options.

`struct ComponentDisplayOption`

A type that indicates whether a formatted URL should include a component.

`struct HostDisplayOption`

A type that indicates whether a formatted URL should include the host component.

Customizing style behavior

`func scheme(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's scheme component in accordance with the provided option.

`func user(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's user component in accordance with the provided option.

`func password(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's password component in accordance with the provided option.

`func host(URL.FormatStyle.HostDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's host component in accordance with the provided option.

`struct HostDisplayOption`

A type that indicates whether a formatted URL should include the host component.

`func port(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's port component in accordance with the provided option.

`func path(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's path component in accordance with the provided option.

`func query(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's query component in accordance with the provided option.

`func fragment(URL.FormatStyle.ComponentDisplayOption) -> URL.FormatStyle`

Modifies a format style to display a URL's fragment component in accordance with the provided option.

`struct` `ComponentDisplayOption`

A type that indicates whether a formatted URL should include a component.

Parsing URLs

`struct` `ParseStrategy`

A parse strategy for creating URLs from formatted strings.

Enumerations

`enum` `Component`

An enumeration of the components of a URL, for use in creating format style options that depend on a component's value.

Relationships

Conforms To

`Copyable`
`Decodable`
`Encodable`
`Equatable`
`FormatStyle`
`Hashable`
`ParseableFormatStyle`
`Sendable`
`SendableMetatype`

See Also

Data formatting in Swift

{} Language Introspector

Converts data into human-readable text using formatters and locales.

`protocol` `FormatStyle`

A type that converts a given data type into a representation in another type, such as a string.

`struct` `IntegerFormatStyle`

A structure that converts between integer values and their textual representations.

`struct` `FloatingPointFormatStyle`

A structure that converts between floating-point values and their textual representations.

`struct` `FormatStyle`

A structure that converts between decimal values and their textual representations.

`struct` `ListFormatStyle`

A type that formats lists of items with a separator and conjunction appropriate for a given locale.

`struct` `StringStyle`

`struct` `FormatStyleCapitalizationContext`

The capitalization formatting context used when formatting dates and times.

Format Style Configurations

Behaviors for traits like numeric precision, rounding, and scale, used for formatting and parsing numeric values.