

Foundation / ParseStrategy

Protocol

ParseStrategy

A type that parses an input representation, such as a formatted string, into a provided data type.

iOS 15.0+ | iPadOS 15.0+ | Mac Catalyst 15.0+ | macOS 12.0+ | tvOS 15.0+ | visionOS 1.0+ | watchOS 8.0+

```
protocol ParseStrategy : Decodable, Encodable, Hashable
```

Overview

A [ParseStrategy](#) allows you to convert a formatted representation into a data type, using one of two approaches:

- Initialize the data type by calling an initializer of that type that takes a formatted instance and a parse strategy as parameters. For example, you can create a `Decimal` from a formatted string with the initializer `init(:format: lenient:)`.
 - Create a parse strategy and call its `parse(:)` method on one or more formatted instances.

`ParseStrategy` is closely related to `FormatStyle`, which provides the opposite conversion: from data type to formatted representation. To use a parse strategy, you create a `FormatStyle` to define the representation you expect, then access the style's `parseStrategy` property to get a strategy instance.

The following example creates a `DecimalFormatStyle.Currency` format style that uses US dollars and US English number-formatting conventions. It then creates a `Decimal` instance by providing a formatted string to parse and the format style's Decimal/Format Style/Currency/parseStrategy.

```
let parsed = try? Decimal("$12,345.67",  
    strategy: style.parseStrategy) // 12345.67
```

Topics

Performing parsing

```
func parse(Self.ParseInput) throws -> Self.ParseOutput
```

Parses a value, using this strategy.

Required

Commonly-used parsers

Use the static accessors in this section to get parse strategies for common input types like dates and URLs.

```
static func fixed(format: Date.FormatString, timeZone: TimeZone, locale: Locale?) -> Self
```

A fixed-format date parse strategy.

```
static var url: URL.ParseStrategy
```

A parse strategy for URLs.

```
static var name: PersonNameComponents.ParseStrategy
```

A parse strategy for person name components.

Commonly-used format styles

```
static var dateTime: Date.FormatStyle
```

A default format style for formatting dates.

Supporting types

```
associatedtype ParseInput
```

The input type parsed by this strategy.

Required

```
associatedtype ParseOutput
```

The output type returned by this strategy.

Required

Type Properties

```
static var http: Date.HTTPFormatStyle  
static var http: DateComponents.HTTPFormatStyle  
static var iso8601: DateComponents.ISO8601FormatStyle  
static var iso8601: Date.ISO8601FormatStyle
```

Relationships

Inherits From

Decodable
Encodable
Equatable
Hashable

Conforming Types

Date.FormatStyle
Date.HTTPFormatStyle
Date.ISO8601FormatStyle
Date.ParseStrategy
DateComponents.HTTPFormatStyle
DateComponents.ISO8601FormatStyle
Decimal.ParseStrategy
FloatingPointParseStrategy

Conforms when Format conforms to FormatStyle and Format.FormatInput conforms to BinaryFloatingPoint.

IntegerParseStrategy

Conforms when Format conforms to FormatStyle and Format.FormatInput conforms to BinaryInteger.

PersonNameComponents.ParseStrategy
URL.ParseStrategy

See Also

Data parsing in Swift

`protocol ParseableFormatStyle`

A type that can convert a given input data type into a representation in an output type.

`struct IntegerParseStrategy`

A parse strategy for creating integer values from formatted strings.

`struct FloatingPointParseStrategy`

A parse strategy for creating floating-point values from formatted strings.

`struct ParseStrategy`

A parse strategy for creating decimal values from formatted strings.