Structure

CodableConfiguration

A property wrapper that makes a type codable, by supplying a configuration that provides additional information for serialization.

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

```
@propertyWrapper
struct CodableConfiguration<T, ConfigurationProvider> where T : DecodableWith
Configuration, T : EncodableWithConfiguration, ConfigurationProvider : Decoding
ConfigurationProviding, ConfigurationProvider : EncodingConfiguration
Providing, T.DecodingConfiguration == ConfigurationProvider.Decoding
Configuration, T.EncodingConfiguration == ConfigurationProvider.Encoding
Configuration
```

Overview

`CodableConfiguration` allows you to create `Codable` types whose members don't all conform to `Codable`. For types that can't support encoding and decoding by themselves but could become encodable and decodable with some statically-defined information, use the `@CodableConfiguration` wrapper. This lets you assign a configuration provider — a type that conforms to both `EncodingConfigurationProviding` and `DecodingConfiguration Providing` — to supply this data.

Limiting the `CodableConfiguration` to statically-defined information protects clients from loading unexpected data, similar to the protection provided by `NSSecureCoding`.

In the following example, the `Message` type uses `@CodableConfiguration` for an `AttributedString` property called `content`. While `AttributedString` does conform to `Codable`, it can only encode its known attributes — those declared by the platform SDK — as part of this conformance. By adding a `CodableConfiguration` for the custom `MyAttributes`

type, `Message` uses `encode(to:configuration:)` when encoding `content`, which preserves the custom attributes.

```
struct Message: Codable {
    let date: Date
    let sender: Person
    @CodableConfiguration(from: MyAttributes.self) var content = AttributedString()
}
```

Topics

Creating a Codable Configuration

`init(wrappedValue: T)`

Creates a codable configuration wrapper for the given value.

`init(wrappedValue: T, from: ConfigurationProvider.Type)`

Creates a codable configuration wrapper for the given value, using the given configuration provider type.

`init(wrappedValue: T, from: KeyPath<AttributeScopes, Configuration Provider.Type>)`

Creates a codable configuration wrapper for the given value, using given configuration provider type identified by key path.

Accessing the Wrapped Value

`var wrappedValue: T`

The underlying value to make codable, using data from the configuration provider.

Relationships

Conforms To

`Copyable`
`Decodable`
`Encodable`

`Equatable`
`Hashable`
`Sendable`
`SendableMetatype`

See Also

Serializing Arbitrary Payloads

`typealias` `CodableWithConfiguration`

A type that can convert itself into and out of an external representation with the help of a configuration that handles encoding contained types.

`protocol` `DecodableWithConfiguration`

A protocol for types that support decoding when supplied with an additional configuration type.

`protocol` `DecodingConfigurationProviding`

A protocol whose conformers provide a configuration instance to help decode types that don’t support encoding by themselves.

`protocol` `EncodableWithConfiguration`

A protocol for types that support encoding when supplied with an additional configuration type.

`protocol` `EncodingConfigurationProviding`

A protocol whose conformers provide a configuration instance to help encode types that don’t support encoding by themselves.