Structure

Predicate

A logical condition used to test a set of input values for searching or filtering.

iOS 17.0+ | iPadOS 17.0+ | Mac Catalyst 17.0+ | macOS 14.0+ | tvOS 17.0+ | visionOS 1.0+ | watchOS 10.0+

```
struct Predicate<each Input>
```

Overview

A predicate is a logical condition that evaluates to a Boolean value (true or false). You use predicates for operations like filtering a collection or searching for matching elements.

To create a predicate, use the `Predicate(_:)` macro. For example:

```
let messagePredicate = #Predicate<Message> { message in
    message.length < 100 && message.sender == "Jeremy"
}
```

In the example above, the closure that contains the predicate's conditions takes one argument — the value being tested. Even though you write the predicate using a closure, the macro transforms that closure into a predicate when you compile. The code in the closure isn't run as part of your program.

In the predicate's definition, you can use the following operations:

- Arithmetic (+, −, ∗, /, %)
- Unary minus (−)
- Range (`...`, `..<`)
- Comparison (<, <=, >, >=, ==, !=)

- Ternary conditional (`?:`)
- Conditional expressions
- Boolean logic (&&, `||`, `!`)
- Swift optionals (?, `??`, `!`, `flatMap(_:)`, `if-let` expressions)
- Types (`as`, `as?`, `as!`, `is`)
- Sequence operations (`allSatisfy()`, `filter()`, `contains()`, `contains(where:)`, `starts(with:)`, `max()`, `min()`)
- Subscript and member access (`[]`, `.`)
- String comparisons (`contains(_:)`, `localizedStandardContains(_:)`, `caseInsensitiveCompare(_:)`, `localizedCompare(_:)`)

A predicate can't contain any nested declarations, use any flow control such as `for` loops, or modify variables from its enclosing scope. However, it can refer to constants that are in scope.

To express more complex queries, you can nest expressions in the predicate:

```
let messagePredicate = #Predicate<Message> { message in
    message.recipients.contains {
        $0.firstName == message.sender.firstName
    }
}
```

You can safely encode and decode predicates, pass predicates across concurrency boundaries, and load a predicate from a file. To define a list of types and key paths that are allowed when reading an archived predicate, use `PredicateCodableConfiguration`.

You can transform a predicate into another representation — for example, to express a predicate in another query language, or to create a modified predicate — using the `expression` property.

Topics

Inspecting and transforming a predicate

`let expression: any StandardPredicateExpression<Bool>`

The component expressions of the predicate.

Initializers

```
init((repeat PredicateExpressions.Variable<each Input>) -> any Standard
PredicateExpression<Bool>)
```

Instance Properties

```
let variable: (repeat PredicateExpressions.Variable<each Input>)
```

Instance Methods

```
func evaluate(repeat each Input) throws -> Bool
```

Type Properties

```
static var `false`: Predicate<repeat each Input>
```

```
static var `true`: Predicate<repeat each Input>
```

Relationships

Conforms To

`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Decodable`
`DecodableWithConfiguration`

Conforms when `each Input` conforms to `Copyable` and `Escapable`.

`Encodable`
`EncodableWithConfiguration`

Conforms when `each Input` conforms to `Copyable` and `Escapable`.

`Sendable`
`SendableMetatype`

See Also

Filltering

`struct PredicateError`

An error thrown while evaluating a predicate.

`struct PredicateCodableConfiguration`

A specification of the expected types and key paths found in an archived predicate.

`protocol PredicateCodableKeyPathProviding`

A type that provides the expected key paths found in an archived predicate.

`protocol PredicateExpression`

A component expression that makes up part of a predicate.

`protocol StandardPredicateExpression`

A component expression that makes up part of a predicate, and that’s supported by the standard predicate type.

`enum PredicateExpressions`

The expressions that make up a predicate.

`struct PredicateBindings`

A mapping from a predicates’s input variables to their values.

`class NSPredicate`

A definition of logical conditions for constraining a search for a fetch or for in-memory filtering.

`class NSExpression`

An expression for use in a comparison predicate.

`class NSComparisonPredicate`

A specialized predicate for comparing expressions.

`class NSCompoundPredicate`

A specialized predicate that evaluates logical combinations of other predicates.