Macro

Unique(_:)

Specifies the key-paths that SwiftData uses to enforce the uniqueness of model instances.

iOS 18.0+ | iPadOS 18.0+ | Mac Catalyst 18.0+ | macOS 15.0+ | tvOS 18.0+ | visionOS 1.0+ | watchOS 11.0+ | Swift 5.9+

```
@freestanding(declaration)
macro Unique<T>(_ constraints: [PartialKeyPath<T>]...) where T : PersistentModel
```

Parameters

`constraints`
: Arrays of model key-paths that form the unique constraints to apply to the enclosing model.

Overview

If a model class contains attributes that you require to be unique across all persisted instances of that model, add the `Unique` macro to that model's definition. You can specify a constraint on a single attribute, a compound constraint across multiple attributes, or any combination of the two.

> **Important**
>
> For relationship attributes, SwiftData only supports unique constraints on those that reference a single persistent model, rather than an array of persistent models.

The following example declares that every instance of `Person` has a unique `id`, and that no two instances of `Person` have the same `givenName` and `familyName`:

```
@Model
final class Person {
    // Declare any unique constraints as part of the model definition.
    #Unique<Person>([\.id], [\.givenName, \.familyName])

    var id: UUID
    var givenName: String
    var familyName: String

    init(id: UUID, givenName: String, familyName: String) {
        self.id = id
        self.givenName = givenName
        self.familyName = familyName
    }
}
```

See Also

Model definition

`macro Model()`

Converts a Swift class into a stored model that's managed by SwiftData.

`macro Attribute(Schema.Attribute.Option..., originalName: String?, hashModifier: String?)`

Specifies the custom behavior that SwiftData applies to the annotated property when managing the owning class.

`macro Index<T>([PartialKeyPath<T>]...)`

Specifies the key-paths that SwiftData uses to create one or more binary indices for the associated model.

`macro Index<T>(Schema.Index<T>.Types<T>...)`

Specifies the key-paths that SwiftData uses to create one or more indicies for the associated model, where each index is either binary or R-tree.

{} Defining data relationships with enumerations and model classes

Create relationships for static and dynamic data stored in your app.

```
macro Relationship(Schema.Relationship.Option..., deleteRule: Schema.
Relationship.DeleteRule, minimumModelCount: Int?, maximumModelCount:
Int?, originalName: String?, inverse: AnyKeyPath?, hashModifier: String
?)
```

Specifies the options that SwiftData needs to manage the annotated property as a relationship between two models.

```
macro Transient()
```

Tells SwiftData not to persist the annotated property when managing the owning class.