Macro

Transient()

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

iOS 17.0+ | iPadOS 17.0+ | Mac Catalyst 17.0+ | macOS 14.0+ | tvOS 17.0+ | visionOS 1.0+ | watchOS 10.0+ | Swift 5.9+

```
@attached(peer)
macro Transient()
```

Mentioned in

Preserving your app's model data across launches

Overview

If your model class has one or more stored properties that you want to omit from writes to the persistent storage, annotate each of those properties with the `@Transient` macro.

> **Note**
>
> By default, SwiftData considers any computed properties to be transient. You don't need to explicitly annotate those properties.

```
@Model
class RemoteImage {
    var sourceURL: URL
    var data: Data
```

```
    @Transient
    var isDownloading = false

    init(sourceURL: URL, data: Data = Data(), isDownloading: Bool) {
        self.sourceURL = sourceURL
        self.data = data
        self.isDownloading = isDownloading
    }
}
```

Unless the type of the annotated property is an optional, the `@Transient` macro requires you to provide a default value. This constraint enables SwiftData to successfully materialize instances of the enclosing model class when running fetches.

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 Unique<T>([PartialKeyPath<T>]...)`

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

`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.