

[SwiftData](#) / FetchDescriptor

Structure

FetchDescriptor

A type that describes the criteria, sort order, and any additional configuration to use when performing a fetch.

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+

```
struct FetchDescriptor<T> where T : PersistentModel
```

Mentioned in

 Preserving your app's model data across launches

Overview

Use a fetch descriptor to capture the criteria necessary to select, and optionally sort, a specific collection of models from your app's persistent storage. A fetch descriptor retrieves only a single type of persistent model, and relies on type inference to determine the appropriate type. However, you can configure a fetch descriptor to prefetch related models of different types using the [relationshipKeyPathsForPrefetching](#) property.

To fetch a collection of models, first create a fetch descriptor and specify a predicate and one or more sort descriptors. The predicate describes the attributes to filter by and the constraints to apply to those attributes. If you don't specify a predicate, the fetch returns all models of the inferred type. You can further tweak a fetch by limiting the number of models it returns, or indicating whether the fetch evaluates any unsaved changes when it selects the models to return. After configuring the fetch descriptor, pass it to the model context's [fetch\(: \)](#) method to run the fetch.

```
let descriptor = FetchDescriptor<Recipe>(
    predicate: #Predicate { $0.isFavorite == true },
    sortBy: [
        .init(\.createdAt)
    ]
)
descriptor.fetchLimit = 10

let favoriteRecipes = try modelContext.fetch(descriptor)
```

If you're displaying the fetched models in a SwiftUI view, use the descriptor with the `Query(_: animation:)` macro instead.

```
struct FavoriteRecipesList: View {
    static var fetchDescriptor: FetchDescriptor<Recipe> {
        let descriptor = FetchDescriptor<Recipe>(
            predicate: #Predicate { $0.isFavorite == true },
            sortBy: [
                .init(\.createdAt)
            ]
        )
        descriptor.fetchLimit = 10
        return descriptor
    }

    @Query(FavoriteRecipesList.fetchDescriptor) private var favoriteRecipes: [Recipe]

    var body: some View {
        List(favoriteRecipes) { RecipeRowView($0) }
    }
}
```

Topics

Creating a fetch descriptor

```
init(predicate: Predicate<T>?, sortBy: [SortDescriptor<T>])
```

Creates a fetch descriptor with the specified predicate that, optionally, arranges the fetched models in a particular order.

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

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

```
struct SortDescriptor<Compared>
```

A serializable description of how to sort numerics and strings.

Constraining the fetch

```
var predicate: Predicate<T>?
```

The logical condition that determines whether the fetch includes a specific model in its results.

```
var sortBy: [SortDescriptor<T>]
```

The sort descriptors that tell the fetch how to order its results.

```
var fetchLimit: Int?
```

The maximum number of models the fetch can return.

```
var fetchOffset: Int?
```

The offset of the first matching model to fetch.

```
var includePendingChanges: Bool
```

A Boolean value that indicates whether, when the fetch runs, it matches against currently unsaved changes in the model context.

Specifying the fetched attributes

```
var relationshipKeyPathsForPrefetching: [PartialKeyPath<T>]
```

The key paths that identify any related models to include as part of the fetch.

```
var propertiesToFetch: [PartialKeyPath<T>]
```

The specific subset of attributes to fetch if you don't require them all.

Relationships

Conforms To

Copyable

Equatable
Sendable
SendableMetatype

See Also

Model fetch

`{}` Filtering and sorting persistent data

Manage data store presentation using predicates and dynamic queries.

`macro Query()`

Fetches all instances of the attached model type.

`:≡` Additional query macros

Supplementary macros that enable you to narrow query results and tell SwiftData how to sort and order those results.

`struct Query`

A type that fetches models using the specified criteria, and manages those models so they remain in sync with the underlying data.