Protocol

# EntityPropertyQuery

**An interface for locating entities by matching values against one or more of their properties.**

iOS 16.0+ | iPadOS 16.0+ | Mac Catalyst | macOS 13.0+ | tvOS 16.0+ | visionOS | watchOS 9.0+

```
protocol EntityPropertyQuery : EntityQuery
```

## Mentioned in

Integrating custom data types into your intents

## Overview

`EntityPropertyQuery` provides an `EntityQuery` with the ability to query instances according to its traits as they are modeled through query properties and their corresponding comparators.

At runtime, `entities(matching:mode:sortedBy:limit:)` receives an array of `ComparatorMappingType` instances - a type of your choice that represents the different comparator mappings given by the user - and is responsible for fetching entities matching those comparators.

The `properties` property on a `EntityPropertyQuery` contains the set of `EntityQuery Property`s the query accepts. When declaring each property's set of supported `EntityQuery Comparator`s, you will supply a closure that will transform a runtime user-supplied value into an instance of `ComparatorMappingType`.

The AppIntents runtime will invoke query comparator mapping closures for each comparator included in the user request, gather their output `ComparatorMappingType` values, and then invoke the `entities` function. It is then up to you to retrieve entities matching those

comparators, using whichever backend (in-memory lookup, CoreData, remote network call, ...) suits your application. For example, consider the following simplified `AppEntity` for a photo:

```
class MyPhoto: AppEntity {
    @Property(title: "Date taken")
    let takenAt: Date

    @Property(title: "Tags")
    let tags: [String]
}
```

The following example shows how you would build a `EntityPropertyQuery` that allows the user to query for photos taken:

- Before a certain date
- After a certain date
- Containing a given tag

In `entities` you want to retrieve photos through a network request, so you supply mapping closures returning `URLQueryItem` instances for each `EntityQueryComparator`. Then you can use those `URLQueryItem` in `entities` to construct the correct URL to fetch entities.

```
struct MyPhotoQuery: EntityPropertyQuery {
   static var properties = QueryProperties {
       Property(\.$takenAt) {
           LessThanMapping { URLQueryItem(name: "takenBefore", value: $0) }
           GreaterThanMapping { URLQueryItem(name: "takenAfter": value: $0) }
       }
       Property(\.$tags) {
           ContainsComparator { URLQueryItem(name: "tagsContains", value: $0) }
       }
   }

   func entities(
       matching comparators: [URLQueryItem],
       mode: ComparatorMode,
       sortedBy: [Sort<MyPhoto>],
       limit: Int?
   ) async throws -> [MyPhoto] {
       let components = URLComponents()
       components.queryItems = comparators
```

```
        let url = components.url(relativeTo: "https://myphotosbackend.com/photos")

        return try await PhotosBackend.fetch(url: url)
    }
```

# Topics

## Specifying the queryable properties

`static var properties: Self.QueryProperties`

The set of query properties supported by this query.

**Required**

`typealias QueryProperties`

`typealias Property`

`associatedtype ComparatorMappingType`

Type produced by `EntityQueryComparator` mapping closures and supplied as input to `results`.

**Required**

## Sorting the results

`static var sortingOptions: Self.SortingOptions`

The set of sorting orders supported by this query.

**Required**

`typealias SortingOptions`

`typealias SortableBy`

## Searching for entities

`func entities(matching: [Self.ComparatorMappingType], mode: Self.ComparatorMode, sortedBy: [EntityQuerySort<Self.Entity>], limit: Int?) async throws -> Self.Result`

Retrieves instances matching the supplied comparators.

**Required**

`typealias Sort`

`typealias` ComparatorMode

`enum` EntityQueryComparatorMode

Modes that determine how to apply a query's comparators.

### Type Properties

`static var` findIntentDescription: IntentDescription?

Defines how the generated 'Find' Shortcuts action of this query type is displayed to the user.

**Required** Default implementation provided.

---

## Relationships

### Inherits From

DynamicOptionsProvider
EntityQuery
PersistentlyIdentifiable
Sendable
SendableMetatype

---

## See Also

### Property-matched queries

`struct` EntityQueryProperties

A type that provides the properties to include in a property-matched query.

`class` EntityQueryProperty

An object that provides the supported comparators you use to describe the different ways users can query against a property of an app entity.

Property comparators

Specify the type of comparison to perform during a property-matched query.

`struct` EntityQuerySortingOptions

The potential properties you can use to sort the results of a query.

`struct` `EntityQuerySortableByProperty`

Details about a specific property you use to sort the query results.

`struct` `EntityQuerySort`

The properties to use to sort the results when the query runs.