Class

NSFetchRequest

A description of search criteria used to retrieve data from a persistent store.

iOS 3.0+ | iPadOS 3.0+ | Mac Catalyst 13.1+ | macOS 10.4+ | tvOS | visionOS 1.0+ | watchOS 2.0+

```
class NSFetchRequest<ResultType> where ResultType : NSFetchRequestResult
```

Overview

An instance of `NSFetchRequest` collects the criteria needed to select and optionally to sort a group of NSManagedObject managed objects held in an `NSPersistentStore` persistent store. A fetch request contains an `NSEntityDescription` or an entity name that specifies which entity to search. It frequently also contains:

- An `NSPredicate` predicate that specifies which properties to filter by and the constraints on selection, such as, `“last name begins with a ‘J’”`. If you don’t specify a predicate, then the system fetches all instances of the entity that you specified, subject to other constraints. For more information, see `fetch(_:)`.
- An array of `NSSortDescriptor` sort descriptors that specify how to order the returned objects, such as ascending by last name and then by first name.

You can also specify other aspects of a fetch request:

`fetchLimit`
: The maximum number of objects that a request returns

`fetchOffset`
: The number of objects to skip

`affectedStores`
: Which data stores the request accesses

`resultType`
: Whether the fetch returns managed objects, object IDs, dictionaries, or a count

`includesPropertyValues` and
: Whether objects are fully populated with their properties

`returnsObjectsAsFaults`
: Whether the objects are faults

`includesSubentities`
: Whether the fetch includes subentities of the fetched entity

`propertiesToFetch`
: Which properties to fetch

`includesPendingChanges`
: Whether to include unsaved changes

Use `execute()` to perform the fetch directly on the managed object context that's associated with the current queue. Or use one of the `NSManagedObjectContext` methods such as `perform(_:)` to execute the fetch.

> Note
>
> When you execute an instance of `NSFetchRequest`, it always accesses the underlying persistent stores to retrieve the latest results.

In SwiftUI, you can use a `FetchRequest` property wrapper to execute the fetch and assign the results to a property. First, create the request:

```
let request: NSFetchRequest = {
    // Create a fetch request.
    let request = ShoppingItem.fetchRequest()

    // Limit the maximum number of items that the request returns.
    request.fetchLimit = 100

    // Filter the request results, such as to only return unchecked items.
    request.predicate = NSPredicate(format: "isChecked = false")

    // Sort the fetched results, such as ascending by name.
    request.sortDescriptors = [NSSortDescriptor(keyPath: \ShoppingItem.name, ascendi

    return request
}()
```

Then use a FetchRequest property wrapper with the request to declare a property that receives the objects that the fetch returns:

```
// Use a `FetchRequest` property wrapper to fetch the managed objects
// and assign the result.
@FetchRequest(fetchRequest: request) private var items: FetchedResults<ShoppingItem>
```

Tip

If you don’t need to specify multiple properties of the fetch, you can avoid creating the fetch request separately and declare it in the property wrapper instead. See FetchRequest for more information.

You often predefine fetch requests in an NSManagedObjectModel managed object model to provide an API to retrieve a stored fetch request by name. Stored fetch requests can include placeholders for variable substitution, and serve as templates for later completion. Fetch request templates allow you to predefine queries with variables to substitute at runtime.

Topics

Managing the Fetch Request’s Entity

`init()`

Creates a new fetch request.

`convenience init(entityName: String)`

Initializes a fetch request configured with a given entity name.

`var entityName: String?`

The name of the entity the request is configured to fetch.

`var entity: NSEntityDescription?`

The entity specified for the fetch request.

`var includesSubentities: Bool`

A Boolean value that indicates whether the fetch request includes subentities in the results.

`struct NSFetchRequestResultType`

Constants that specify the possible result types a fetch request can return.

Specifying Fetch Constraints

`var predicate: NSPredicate?`

The predicate of the fetch request.

`var fetchLimit: Int`

The fetch limit of the fetch request.

`var fetchOffset: Int`

The fetch offset of the fetch request.

`var fetchBatchSize: Int`

The batch size of the objects specified in the fetch request.

`var affectedStores: [NSPersistentStore]?`

An array of persistent stores specified for the fetch request.

`class NSFetchRequestExpression`

An expression that evaluates the result of a fetch request on a managed object context.

`class NSExpressionDescription`

An object that describes an expression to include with a fetch request.

`class NSFetchedPropertyDescription`

A description object used to define which properties are fetched from Core Data.

Sorting the Results

`var sortDescriptors: [NSSortDescriptor]?`

The sort descriptors of the fetch request.

Prefetching Related Objects

`var relationshipKeyPathsForPrefetching: [String]?`

The relationship key paths to prefetch along with the entity for the request.

Managing How Results Are Returned

`var resultType: NSFetchRequestResultType`

The result type of the fetch request.

`var includesPendingChanges: Bool`

A Boolean value that indicates whether, when the fetch is executed, it matches against currently unsaved changes in the managed object context.

`var propertiesToFetch: [Any]?`

A collection of either property descriptions or string property names that specify which properties should be returned by the fetch.

`var returnsDistinctResults: Bool`

A Boolean value that indicates whether the fetch request returns only distinct values for the fields specified by `propertiesToFetch`.

`var includesPropertyValues: Bool`

A Boolean value that indicates whether, when the fetch is executed, property data is obtained from the persistent store.

`var shouldRefreshRefetchedObjects: Bool`

A Boolean value that indicates whether the property values of fetched objects will be updated with the current values in the persistent store.

`var returnsObjectsAsFaults: Bool`

A Boolean value that indicates whether the objects resulting from a fetch request are faults.

`struct NSFetchRequestResultType`

Constants that specify the possible result types a fetch request can return.

`protocol NSFetchRequestResult`

An abstract protocol used with parameterized fetch requests.

Grouping and Filtering Dictionary Results

`var propertiesToGroupBy: [Any]?`

An array of objects that indicates how data should be grouped before a select statement is run in a SQL database.

`var havingPredicate: NSPredicate?`

The predicate used to filter rows being returned by a query containing a GROUP BY directive.

Executing a Fetch Request Directly

`func execute() throws -> [ResultType]`

Executes the fetch request against the managed object context that is associated with the current queue.

Relationships

Inherits From

`NSPersistentStoreRequest`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSCoding`
`NSCopying`
`NSObjectProtocol`

See Also

Fetch requests

`class NSAsynchronousFetchRequest`

A fetch request that retrieves results asynchronously and supports progress notification.

`class NSAsynchronousFetchResult`

A fetch result object that encompasses the response from an executed asynchronous fetch request.

```
class NSFetchedResultsController
```

A controller that you use to manage the results of a Core Data fetch request and to display data to the user.