Article

Retrieving events and reminders

Fetch events and reminders from the Calendar database.

Overview

To search for and return events and reminders from the Calendar database, you must connect to the event store, as discussed in “Accessing the event store.”

There are two ways to retrieve events and reminders. You can fetch via:

1. *Predicate*, or *search query*, and return zero or more events that match a given query.
2. *Unique identifier* and return a single event that corresponds to the given identifier.

> Note
>
> Retrieving events from the Calendar database does not necessarily return events in chronological order. To sort an array of `EKEvent` objects by date, call `sortedArray(using:)` on the array, providing the selector for the `compareStartDate(with:)` method.

Search with Predicates

Predicates return events and reminders that match a search query.

Fetch Events

It’s common to fetch events and reminders that fall within a date range. The `EKEventStore` method `events(matching:)`: fetches all events that fall within the date range specified in the predicate you provide. The following listing demonstrates how to fetch all events that occur between one day before and one year after the current date.

> **Note**
>
> Although the `events(matching:)` method accepts a parameter of type `NSPredicate`, you must supply a predicate created with the `EKEventStore` method `predicateForEvents(withStart:end:calendars:)`.

Listing 1. Fetching events with a predicate

```
// Get the appropriate calendar.
let calendar = Calendar.current

// Create the start date components
var oneDayAgoComponents = DateComponents()
oneDayAgoComponents.day = -1
let oneDayAgo = calendar.date(byAdding: oneDayAgoComponents, to: Date(), wrappingCom

// Create the end date components.
var oneYearFromNowComponents = DateComponents()
oneYearFromNowComponents.year = 1
var oneYearFromNow = calendar.date(byAdding: oneYearFromNowComponents, to: Date(), w

// Create the predicate from the event store's instance method.
var predicate: NSPredicate? = nil
if let anAgo = oneDayAgo, let aNow = oneYearFromNow {
    predicate = store.predicateForEvents(withStart: anAgo, end: aNow, calendars: nil
}

// Fetch all events that match the predicate.
var events: [EKEvent]? = nil
if let aPredicate = predicate {
    events = store.events(matching: aPredicate)
}
```

You can specify a subset of calendars to search by passing an array of `EKCalendar` objects as the calendars parameter of the `predicateForEvents(withStart:end:calendars:)` method. You can get the user's calendars from the event store's `calendars(for:)` method. Passing `nil` tells the method to fetch from all of the user's calendars.

Because the events(matching:) method is synchronous, you may not want to run it on your app’s main thread. For asynchronous behavior, run the method on another thread with the dispatch_async function or with an Operation object.

Fetch Reminders

You can call fetchReminders(matching:completion:) to access multiple reminders that match a predicate. Pass a predicate returned by one of the following methods:

- predicateForIncompleteReminders(withDueDateStarting:ending:calendars:) finds incomplete reminders within an optional time period.
- predicateForCompletedReminders(withCompletionDateStarting:ending:calendars:) finds completed reminders within an optional time period.
- predicateForReminders(in:) finds all reminders.

You can iterate across matched reminders by passing a block to the completion argument, as shown in the listing below.

```
var predicate: NSPredicate? = store.predicateForReminders(in: nil)
if let aPredicate = predicate {
    store.fetchReminders(matching: aPredicate, completion: {(_ reminders: [Any]?) ->
        for reminder: EKReminder? in reminders as? [EKReminder?] ?? [EKReminder?]()
            // Do something for each reminder.
        }
    })
}
```

Unlike fetching events via predicate, you can fetch reminders via predicate asynchronously without dispatching to another thread.

If you want to abort your fetch request by predicate, call cancelFetchRequest(_:) while passing the identifier as returned by fetchReminders(matching:completion:).

Search with Unique Identifiers

If you know the event’s unique identifier because you fetched it previously with a predicate, you can use the EKEventStore method event(withIdentifier:) to fetch the event. If it is a recurring event, this method will return the first occurrence of the event. You can get an event’s unique identifier with the eventIdentifier property.

Similarly, if you know a specific reminder’s unique identifier from previously fetching it with a predicate, you can call the calendarItem(withIdentifier:) instance method. calendar

`Item(withIdentifier:)` can fetch any calendar item (reminders and events), whereas `event(withIdentifier:)` fetches only events.

See Also

Events and reminders

Creating events and reminders

Create and modify events and reminders in a person’s database.

Updating with notifications

Register for notifications about changes and keep your app up to date.

Managing location-based reminders

Access reminders set up with geofence-enabled alarms on a person’s calendars.

`class EKEvent`

A class that represents an event in a calendar.

`class EKReminder`

A class that represents a reminder in a calendar.