Article

Using Core Data in the background

Use Core Data in both a single-threaded and multithreaded app.

Overview

Core Data works in a multithreaded environment. However, not every object under the Core Data framework is thread safe. To use Core Data in a multithreaded environment, ensure that:

- Bind managed object contexts to the thread (queue) that they're initialization on.
- Bind managed objects that you retrieve from a context to the same queue as the context.

Comparing Main Queue and Private Queue Contexts

There are two types of managed object contexts: main queue and private queue. You define the type of context when you initialize it.

A main queue context (as defined by a `NSManagedObjectContextConcurrencyType.mainQueueConcurrencyType`) is specifically for use with your application interface. Only use it on the main queue of your app.

A private queue context (as defined by a `NSManagedObjectContextConcurrencyType.privateQueueConcurrencyType`) creates its own queue upon initialization. Only use it on that queue. Because the queue is private and internal to the `NSManagedObjectContext` instance, you can only access it through the `perform(_:)` and the `performAndWait(_:)` methods.

Initializing and configuring a context

Use `init(concurrencyType:)` to create a new context. For example, to create a private queue context:

```
// Create a private queue context.
let context = NSManagedObjectContext(.privateQueue)
```

The parameter you pass during initialization determines what type of `NSManagedObjectContext` you receive.

When you use the `NSPersistentContainer`, you configure the `viewContext` property as a main queue (`NSManagedObjectContextConcurrencyType.mainQueueConcurrencyType`) context, and configure the contexts associated with `performBackgroundTask(_:)` and `newBackgroundContext()` as a private queue (`NSManagedObjectContextConcurrencyType.privateQueueConcurrencyType`).

Avoiding problems

In general, avoid doing data processing on the main queue that's not user-related. Data processing can be CPU-intensive, and if it's performed on the main queue, it can result in unresponsiveness in the user interface. If your application processes data, such as importing data into Core Data from JSON, create a private queue context and perform the import on the private context.

Don't pass managed object instances between queues. Doing so can result in corruption of the data and termination of the app. When it's necessary to hand off a managed object reference from one queue to another, use `NSManagedObjectID` instances.

You retrieve the managed object ID of a managed object by calling the `objectID` accessor on the `NSManagedObject` instance.

See Also

Background tasks

{} Loading and displaying a large data feed

Consume data in the background, and lower memory use by batching imports and preventing duplicate records.

Conflict resolution

Detect and resolve conflicts that occur when data is changed on multiple threads.

Batch processing

Use batch processes to manage large data changes.