Swift DispatchGroup – Doing tasks in parallel

ddas.tech/swift-dispatchgroup-doing-tasks-in-parallel/

April 9, 2023

Created By: Debasis Das (Apr 2023)

Introduction to DispatchGroup.

DispatchGroup is a class in Swift that provides a way to group multiple tasks together and wait for them to complete before continuing with the rest of the program. It allows you to synchronize concurrent tasks in a clean and efficient way.

When we create a `DispatchGroup` object, it starts with a count of zero. We can then use the `enter()` method to increment the count whenever we start a new task, and the `leave()` method to decrement the count when the task is finished.

We can also use the `wait()` method to block the current thread until all the tasks in the group have completed. This method will return immediately if the count of the `DispatchGroup` is already zero, or it will block until the count reaches zero.

Finally, we can use the `notify(queue:work:)` method to execute a closure when all the tasks in the group have completed, even if some of them failed or were cancelled. This method does not block the current thread, but instead executes the closure asynchronously on the specified queue.

Sample Code

```
func performTaskOne(completion: @escaping () -> Void) {
        DispatchQueue.global().async {
            print("Task one started at ",Date())
            sleep(2) // Simulate work
            print("Task one completed at ",Date())
            completion()
        }
    }

    func performTaskTwo(completion: @escaping () -> Void) {
        DispatchQueue.global().async {
            print("Task two started at ",Date())
            sleep(4) // Simulate work
            print("Task two completed at ",Date())
            completion()
        }
    }

    func performTasksAndWait() {
        let dispatchGroup = DispatchGroup()

        dispatchGroup.enter()
        performTaskOne {
            dispatchGroup.leave()
        }

        dispatchGroup.enter()
        performTaskTwo {
            dispatchGroup.leave()
        }

        dispatchGroup.notify(queue: .main) {
            print("All tasks completed")
        }

        //Ensure to wait on the background thread.
        //Waiting on the main thread will make the App UI to Freeze.
        DispatchQueue.global().async {
                dispatchGroup.wait()
                print("Done waiting")
                print("Initiate Remaining tasks that were waiting on Task 1 and Task
2")
        }
    }
```

//Here both the tasks are started.
Task two started at 2023-04-09 19:52:45 +0000
Task one started at 2023-04-09 19:52:45 +0000

//Both the tasks are completed at different time
Task one completed at 2023-04-09 19:52:47 +0000
Task two completed at 2023-04-09 19:52:49 +0000

//Once all the tasks are complete, We get a notification
All tasks completed

Done waiting
Initiate Remaining tasks that were waiting on Task 1 and Task 2