Знакомство с GCD

хочу чтоб всё и сразу

Roman Stetsenko
evfemist@gmail.com

Grand Central Dispatch

- libdispatch.dylib
- не является частью Cocoa
- Пул потоков на системном уровне
- Интегрированная система обработки событий

Особенности

- Появилась в iOS 4.0
- Более легковесная чем потоки
- Проста в использовании
- Основной элемент - очередь
- Очереди могут быть последовательными и паралельными
- Глобальные очереди и API для создания локальных.
- Даже если проект с ARC, нужно управлять памятью (вызывать dispatch_retain() и dispatch_release() с созданными объектами GCD)
- Группировка задач
- Подписка на события от файловых дескрипторов, системных портов и процессов, таймеров и сигналов, а также пользовательских событий

Очереди

- dispatch_get_global_queue() - возвращает одну из четырёх глобальных паралельных очередей
- dispatch_get_main_queue() - возвращает последовательную очередь которая выполняет задачи в главном потоке приложения
- dispatch_queue_create() - создаёт новую очередь
- dispatch_suspend() - останавливает запуск новых задач на выполнение
- dispatch_resume() - возвращает очередь к жизни

Выполнение задач

- dispatch_async() - ставит задачу в очередь асинхронно
- dispatch_sync() - ставит задачу в очередь синхронно
- dispatch_after() - ставит задачу в очередь через заданный промежуток времени
- dispatch_apply() - ставит задачу в очередь заданное количество раз
- dispatch_once() - ставит задачу в очередь только раз

Примеры

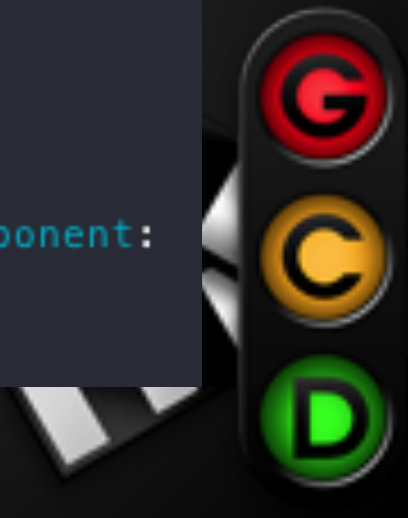
```
UIImageView *imageView = [[UIImageView alloc] initWithFrame:
                                    CGRectMake(10, 10, 300, 300)];
[self.view addSubview:imageView];

dispatch_queue_t myQueue = dispatch_queue_create("myQueue",
                                    DISPATCH_QUEUE_CONCURRENT);
dispatch_async(myQueue, ^{
    NSData *data = [NSData dataWithContentsOfURL:
                                    [NSURL URLWithString:
@"http://images.apple.com/home/images/macbookpro_hero.jpg"]];
    UIImage *image = [UIImage imageWithData:data];
    dispatch_async(dispatch_get_main_queue(), ^{
        imageView.image = image;
    });
});
dispatch_release(myQueue);
```

Примеры

```
for(NSInteger i=0; i<imageArray.count; i++)
{
    UIImage *image = [imageArray objectAtIndex:i];
    NSData *data = UIImagePNGRepresentation(image);
    NSString *fileName = [NSString stringWithFormat:
                          @"image%i", i];
    [data writeToFile:[path stringByAppendingPathComponent:
                       fileName] atomically:YES];
}
```

```
dispatch_apply(imageArray.count,
  dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW, 0),
  ^(size_t i) {
    UIImage *image = [imageArray objectAtIndex:i];
    NSData *data = UIImagePNGRepresentation(image);
    NSString *fileName = [NSString stringWithFormat:
                          @"image%i", i];
    [data writeToFile:[path stringByAppendingPathComponent:
                       fileName] atomically:YES];
});
```



Примеры

```
+(MyClass *)singleton
{
    static MyClass *shared = nil;

    if(shared == nil)
    {
        shared = [[MyClass alloc] init];
    }
    return shared;
}
```

```
+(MyClass *)singleton
{
    static dispatch_once_t pred;
    static MyClass *shared = nil;

    dispatch_once(&pred, ^{
        shared = [[MyClass alloc] init];
    });
    return shared;
}
```

Группировка задач

- dispatch_group_create() - создаёт новую группу
- dispatch_group_async() - добавляет задачу в очередь и привязывает к группе
- dispatch_group_notify() - выполнит указанную задачу после того как все задачи в группе будут выполнены
- dispatch_group_wait() - синхронно ожидает завершения всех задач в группе

Примеры

```
dispatch_queue_t queue = dispatch_get_global_queue(
                 DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
dispatch_group_t group = dispatch_group_create();
for(id obj in array)
{
    dispatch_group_async(group, queue, ^{
        [self doSomethingIntensiveWith:obj];
    });
}

dispatch_group_wait(group, DISPATCH_TIME_FOREVER);
dispatch_release(group);

[self doSomethingWith:array];
```

Барьеры

- dispatch_barrier_async() - отправляет барьер в очередь асинхронно
- dispatch_barrier_sync() - отправляет барьер в очередь синхронно

Примеры

```
-(id)cacheObjectForKey:(id)key
{
    __block id obj;
    dispatch_sync(_queue, ^{
        obj = [_cache objectForKey: key];
    });
    return obj;
}

-(void)setCacheObject:(id)obj forKey:(id)key
{
    dispatch_barrier_async(_queue, ^{
        [_cache setObject:obj forKey:key];
    });
}
```

Разное

- dispatch_debug() - лог для объектов GCD
- dispatch_set_context() - привязывает данные к объекту
- dispatch_get_context() - возвращает данные
- dispatch_queue_set_specific() - записывает значение по ключу в указанную очередь
- dispatch_queue_get_specific() - возвращает значение по ключу в указанную очередь
- dispatch_get_specific() - возвращает значение по ключу в текущей очереди
- GCD не обрабатывает исключений

Спасибо за внимание

Ресурсы:

1. Concurrency Programming Guide
2. Grand Central Dispatch (GCD) Reference
3. "man dispatch" в Терминале
4. http://www.raywenderlich.com/4295/multithreading-and-grand-central-dispatch-on-ios-for-beginners-tutorial
5. http://touchdev.ru/tag/GCD
6. http://idev.by/tag/gcd/

Roman Stetsenko
evfemist@gmail.com