Self-Taught

Lessons Learned

I began learning iOS just under 1 year ago.

Approaching this was different from learning many of the languages and technologies I had in the past as I had to learn on my own, with no one having expertise available to help.

This presentation will be about what I learned and what I would do differently. (Or tell my earlier self!)

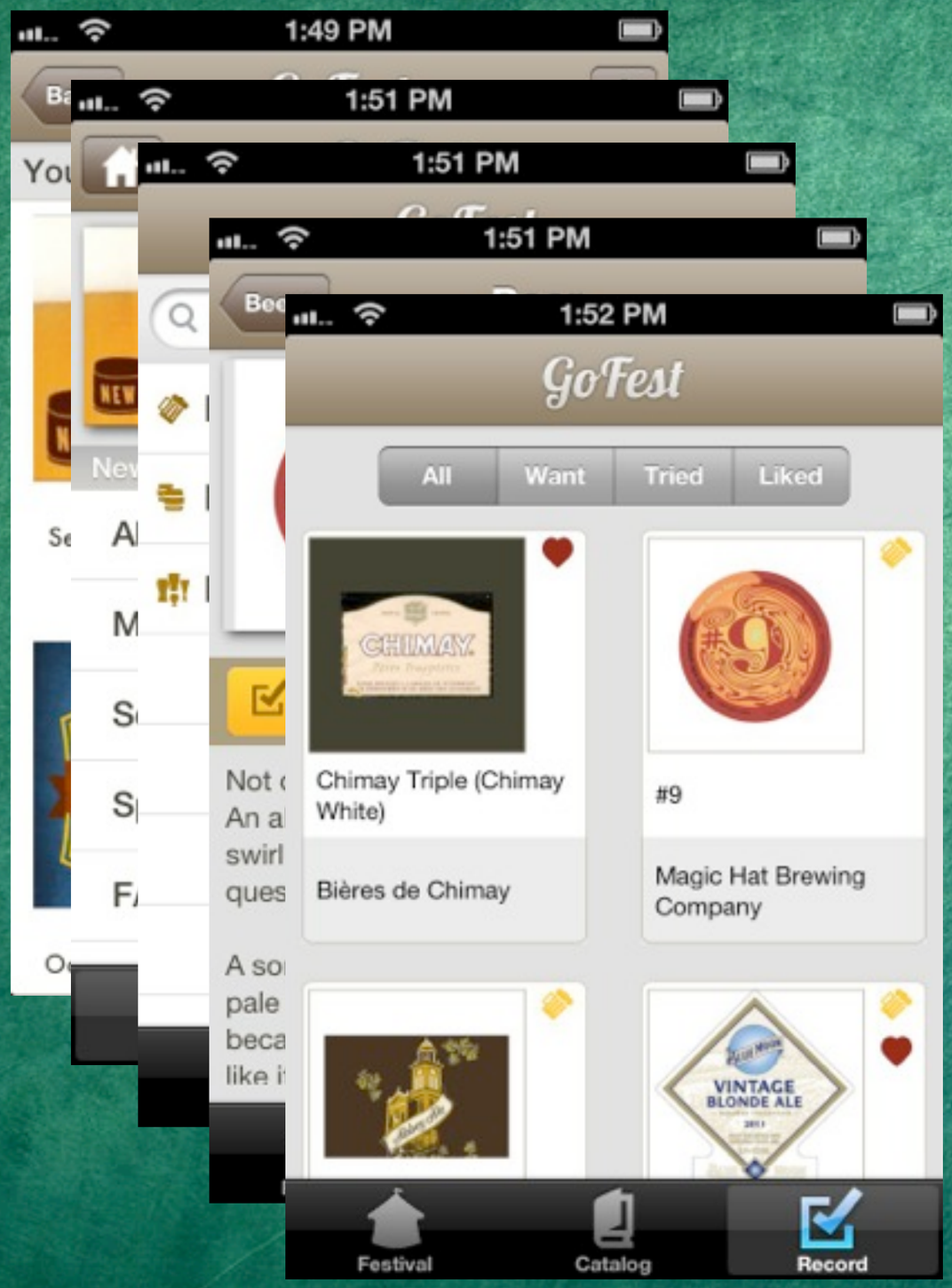
Background

Web Development

My Background is largely in web, Ruby on Rails, integrating systems, etc...

This was beneficial with building the API to back the apps I built.

GoFest



GoFest is the first app I built.

It's a beer festival app platform.
- Users can pull down one or many festivals which are then able to be explored through the app.
- It integrates with a 3rd party data source called brewerydb.com for beer and brewer data.
- I build a backend server in Ruby on Rails to host festival specific information. The app integrates with this.

GoFest

GoFest's first festival will be at the end of the month in New Orleans

GoFest Takeaways

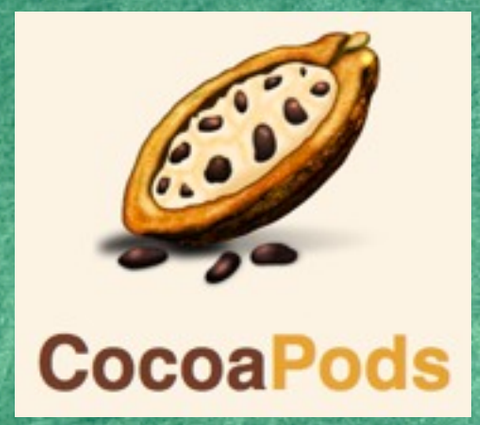
First app
- too big, should have started smaller

B2B
- Communication with organizers takes a lot of time

Technical Complexity
- 3rd party data
- Backend API & CMS
- Cloud storage strategy

UI/UX
- Could use improvements/expertise

Use the right Tools



An engineer is only as good as their tools. It's important to know what tools are available.

I will go over some that I found particularly useful and wish I had known about earlier.

Podfile

```
platform :ios, '6.0'
pod 'AFNetworking', '1.3.1'
pod 'MagicalRecord', '2.1'
pod 'libechonest'
pod 'MBProgressHUD'
pod 'AFOAuth2Client'
pod 'Facebook-iOS-SDK'
pod 'AWSiOSSDK'
pod 'RestKit', '~>  0.20.0'
```

```
Juliettas-MacBook-Air:snapasong_ios juliettayaunches$ pod install
Analyzing dependencies

CocoaPods 0.24.0 is available.

Downloading dependencies
Using AFNetworking (1.3.1)
Using AFOAuth2Client (0.1.1)
Using ASIHTTPRequest (1.8.1)
Using AWSiOSSDK (1.6.0)
Using Facebook-iOS-SDK (3.6.0)
Using MBProgressHUD (0.7)
Using MagicalRecord (2.1)
Using Reachability (3.1.1)
Using RestKit (0.20.3)
Using SBJson (3.2)
Using SOCKit (1.1)
Using TSLibraryImport (0.0.1)
Using TransitionKit (1.1.1)
Using libechonest (1.0.0)
Generating Pods project
Integrating client project
Juliettas-MacBook-Air:snapasong_ios juliettayaunches$
```

Similar to RubyGems

Allows you to manage libraries
- versions, dependencies

I didn’t know this existed until 3 months in.

It is awesome


```objc
#import "AvailableFestival.h"
#import "EnvironmentVars.h"

@interface AvailableFestivalFetcher ()
@property (nonatomic, strong) NSMutableData *responseData;
@end

@implementation AvailableFestivalFetcher

+ (AvailableFestivalFetcher *) festivalRetriever{
    static AvailableFestivalFetcher *requestGenerator = nil;
    if(!requestGenerator)
        requestGenerator = [[AvailableFestivalFetcher alloc] init];

    return requestGenerator;
}

- (void)retrieveFestivals{
    self.responseData = [NSMutableData data];

    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[[EnvironmentVars sharedInst
                                                                cachePolicy:NSURLRequestReloadIgnoringLo
                                                            timeoutInterval:20];

    [request setHTTPMethod: @"GET"];
    NSURLConnection *connection = [[NSURLConnection alloc] initWithRequest:request delegate:self];
    [connection start];
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    NSError *myError = nil;

    NSArray *res = [NSJSONSerialization JSONObjectWithData:self.responseData
                                                   options:NSJSONReadingMutableLeaves
                                                     error:&myError];
    if(myError){
        NSLog(@"Error retrieving available festivals: %@", myError);
        [self.availableFestivalsDelegate couldNotRetrieve];
    }

    NSMutableArray *retrievedFestivals = [[NSMutableArray alloc] init];

    for(NSDictionary* festival in res) {
        [retrievedFestivals addObject: [AvailableFestival initWithInfo:[festival objectForKey:@"festival"]]];
    }
    [self.availableFestivalsDelegate finishLoad:retrievedFestivals];
    [self.networkActivityDelegate finishedActivity];
}

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response {
    [self.responseData setLength:0];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    [self.responseData appendData:data];
}

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {
    NSLog(@"Failed to retrieve festivals by GPS location: %@", error);
    [self.networkActivityDelegate somethingWentWrong];
}
@end
```

```objc
#import "AvailableFestivalFetcher.h"
#import "AvailableFestival.h"
#import "EnvironmentVars.h"
#import "AFNetworking.h"

@implementation AvailableFestivalFetcher

+ (AvailableFestivalFetcher *)festivalRetriever {
    static AvailableFestivalFetcher *requestGenerator = nil;
    if (!requestGenerator)
        requestGenerator = [[AvailableFestivalFetcher alloc] init];

    return requestGenerator;
}

- (void)retrieveFestivals {
    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[[EnvironmentVars sharedInstance] goFestGetFes
                                                                cachePolicy:NSURLRequestReloadIgnoringLocalAndRemoteCacheD
                                                            timeoutInterval:20];

    [request setHTTPMethod:@"GET"];

    AFJSONRequestOperation *operation = [AFJSONRequestOperation JSONRequestOperationWithRequest:request success:^(NSU
        [self processResult:resultJSON];
    }failure:^(NSURLRequest *request1, NSHTTPURLResponse *response, NSError *error, id o) {
        NSLog(@"Failed to retrieve festivals by GPS location: %@", error);
        [self.networkActivityDelegate somethingWentWrong];
    }];

    [operation start];
}

- (void)processResult:(id)resultingJSON {
    NSMutableArray *retrievedFestivals = [[NSMutableArray alloc] init];

    for (NSDictionary *festival in resultingJSON) {
        [retrievedFestivals addObject:[AvailableFestival initWithInfo:[festival objectForKey:@"festival"]]];
    }
    [self.availableFestivalsDelegate finishLoad:retrievedFestivals];
    [self.networkActivityDelegate finishedActivity];
}


@end
```

Less code

AFJSONRequestOperation - lets you specify you're working specifically with json

Block based versus delegate based

MagicalRecord

AppDelegate.m

```objc
- (void)saveContext
{
    NSError *error = nil;
    NSManagedObjectContext *managedObjectContext = self.managedObjectContext;
    if (managedObjectContext != nil) {
        if ([managedObjectContext hasChanges] && ![managedObjectContext save:&error]) {
            // Replace this implementation with code to handle the error appropriately.
            // abort() causes the application to generate a crash log and terminate. You should not us
            NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
            abort();
        }
    }
}

#pragma mark - Core Data stack

// Returns the managed object context for the application.
// If the context doesn't already exist, it is created and bound to the persistent store coordinator f
- (NSManagedObjectContext *)managedObjectContext
{
    if (_managedObjectContext != nil) {
        return _managedObjectContext;
    }

    NSPersistentStoreCoordinator *coordinator = [self persistentStoreCoordina
    if (coordinator != nil) {
        _managedObjectContext = [[NSManagedObjectContext alloc] init];
        [_managedObjectContext setPersistentStoreCoordinator:coordinator];
    }
    return _managedObjectContext;
}

// Returns the managed object model for the application.
// If the model doesn't already exist, it is created from the application's model.
- (NSManagedObjectModel *)managedObjectModel
{
    if (_managedObjectModel != nil) {
        return _managedObjectModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"GoFest" withExtension:@"momd"];
    _managedObjectModel = [[NSManagedObjectModel alloc] initWithContentsOfURL:modelURL];
    return _managedObjectModel;
}

// Returns the persistent store coordinator for the application.
// If the coordinator doesn't already exist, it is created and the application's store added to it.
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator
{
    if (_persistentStoreCoordinator != nil) {
        return _persistentStoreCoordinator;
    }

    NSURL *storeURL = [[self applicationDocumentsDirectory] URLByAppendingPathComponent:@"GoFest.sqlit

    NSError *error = nil;
    _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:[se
    if (![_persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType configuration:nil U
/*
 Replace this implementation with code to handle the error appropriately.
```

```objc
@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    [MagicalRecord setupCoreDataStack];
```

MagicalRecord was inspired by Ruby on Rails ORM Active Record

It simulates data fetching. The goals are:

- Clean up Core Data related code
- Allow for clear, simple, one-line fetches

MagicalRecord

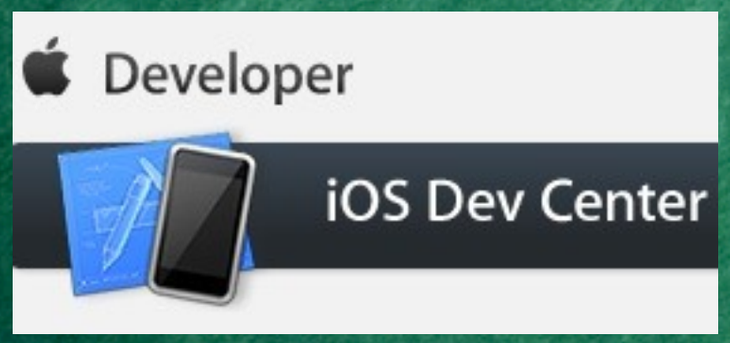
```objc
@implementation UserFestivalsCVC

- (void)viewDidLoad {
    [super viewDidLoad];

    AppDelegate *appDelegate = (AppDelegate *)[[UIApplication sharedApplication] delegate];

    self.managedObjectContext = [appDelegate managedObjectContext];

    NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] init];
    NSEntityDescription *entity = [NSEntityDescription
            entityForName:@"Festival" inManagedObjectContext:self.managedObjectContext];
    [fetchRequest setEntity:entity];
    NSError *error;
    self.savedFestivals = [[self.managedObjectContext executeFetchRequest:fetchRequest error:&error] mutableCopy];
    if(error != nil){
        NSLog(@"Unable to find festivals: %@", error);
    }
}
```

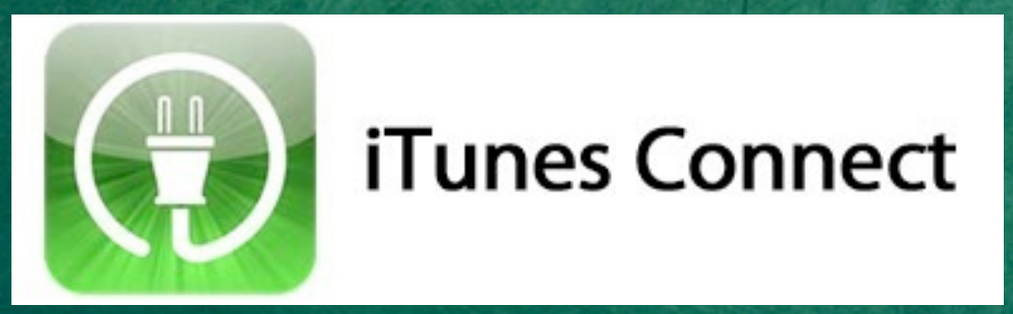
```objc
@implementation UserFestivalsCVC

- (void)viewDidLoad {
    [super viewDidLoad];
    self.savedFestivals = [NSMutableArray arrayWithArray:[Festival findAll]];
}
```

Still allow the modification of the NSFetchRequest when request optimizations are needed

Felt it was good to start using/managing moc myself, then move to MR.

Apple Tools



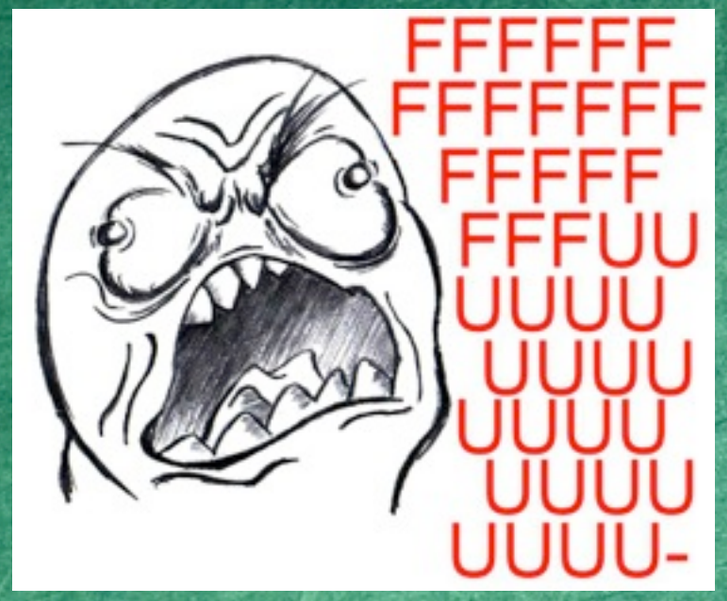


One of the most frustrating things about working in a new technology is learning new toolsets.

Xcode definitely delivers here.

- target configurations
- use instruments tools
- setup build schemes

Apple Tools
(how-to)



RAYWENDERLICH

Tutorials for iPhone / iOS Developers and Gamers

Objectively Speaking: A Crash Course in Objective-C for iOS 6

Linda

How to Submit Your App to Apple: From No Account to App Store, Part 1

Gustavo Ambrozio on January 26, 2012

Tweet 74 Like

Demystifying iOS Application Crash Logs

Sohei Azarpour on January 3, 2013

Tweet 147 Like

Ray Wenderlichs tutorials I found to be the most useful in getting more comfortable with these tools

Apple Tools

(But know about alternatives)

AppCode

I also love AppCode. I use it for debugging, refactoring, and easy class/file/method navigation.

Resources

Coding Together: Developing Apps for iPhone and iPad (Winter 2013)

by Stanford

View More from This Institution

To subscribe to an iTunes U course, click View in iTunes.

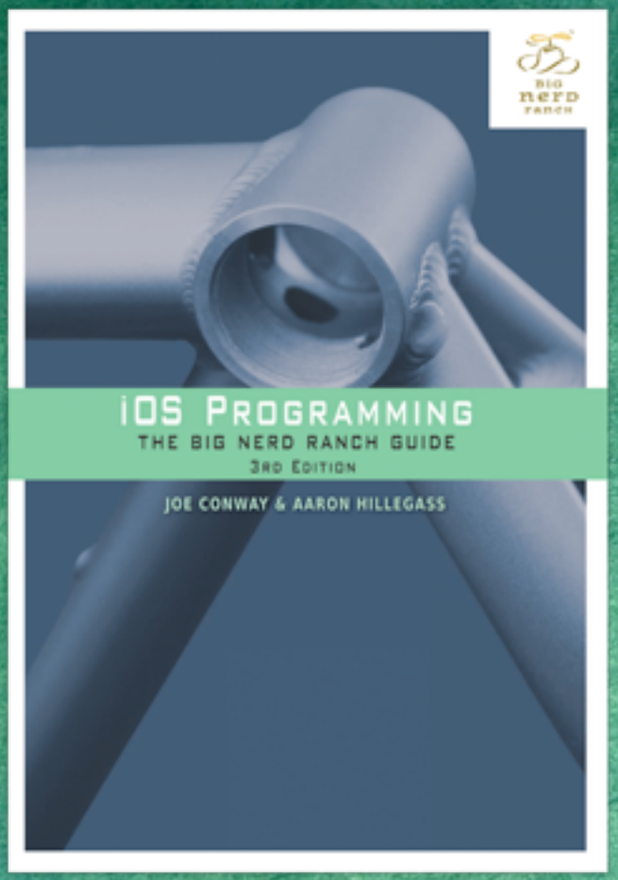
Course Description

Updated for iOS 6. Tools and APIs required to build applications for the iPhone and iPad platform using the iOS SDK. User interface designs for mobile devices and unique user interactions using multi-touch technologies. Object-oriented design using model-view-controller paradigm, memory management, Objective-C programming language. Other topics include: object-oriented database API, animation, multi-threading and performance considerations.

Prerequisites: C language and programming experience at the level of 106B (Programming Abstractions) or X. Recommended: UNIX, object-oriented programming, graphical toolkits

Offered by Stanford's School of Engineering, the course will run from January 22 through March 28. Sign-up begins January 14 and will end on February 1. Released with a Creative Commons BY-NC-SA license. If you have disability-

...More



NSScreencast

Bite-sized videos on iOS development.

RAYWENDERLICH

Tutorials for iPhone / iOS Developers and Gamers

stackoverflow

Obviously, Apple resources are the starting point, but these are some of my favorite external resources.

Discussion