Understanding Core Data Changes in iOS 10 and macOS Sierra

Andrew Bancroft

@andrewcbancroft www.andrewcbancroft.com

Overview

New features in Core Data with Apple’s latest platforms

Generic typing

Simplify Core Data Stack creation with NSPersistentContainer

Wrap up: Additional resources to build on what you’ve learned

Developing with Core Data for Apple’s Latest Platforms

Core Data Release Notes

http://bit.ly/WhatsNewInCoreData

```swift
let shoutOutsFetchRequest = NSFetchRequest<ShoutOut>(
                                entityName: “ShoutOut”)
```

```swift
let frc = NSFetchedResultsController<ShoutOut>(
            fetchRequest: shoutOutFetchRequest,
            managedObjectContext: self.managedObjectContext,
            sectionNameKeyPath: nil,
            cacheName: nil)
```

Parameterized (generic) types

Comes with *Swift*, not iOS/macOS

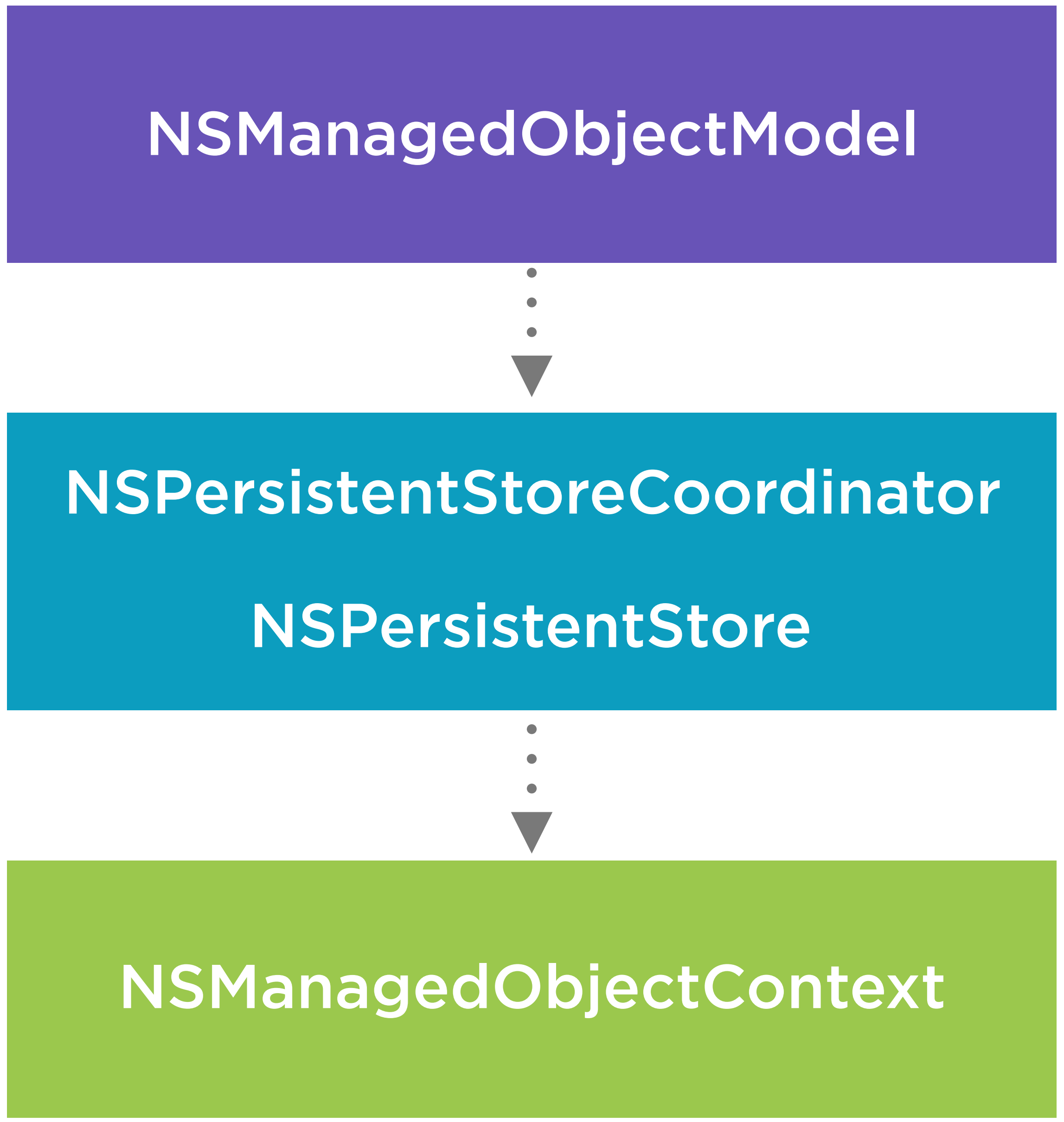
Creating the Core Data Stack with NSPersistentContainer

Setting up the Core Data Stack

Step 1

Step 2

Step 3



Setting up the Core Data Stack

NSPersistentContainer

NSPersistentContainer Default Configuration

SQLiteStoreType

```
let pscOptions = [
                NSMigratePersistentStoresAutomaticallyOption : true,
                NSInferMappingModelAutomaticallyOption : true
        ]

try! psc.addPersistentStore(
               ofType: NSSQLiteStoreType,
               configurationName: nil,
               at: storeURL,
               options: pscOptions)
```

Enabling Lightweight Migrations

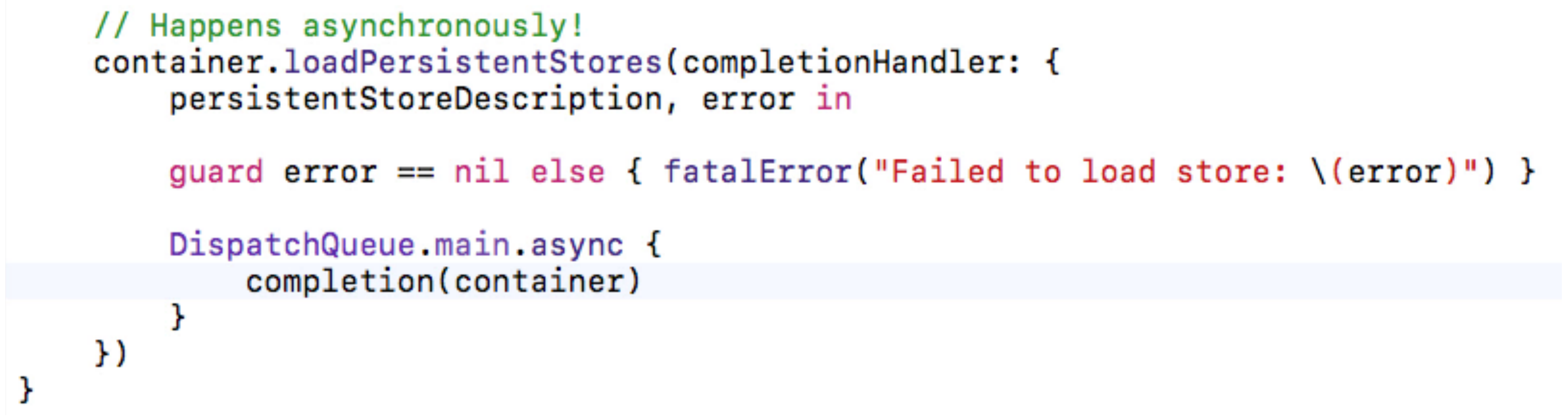
Create an options dictionary to pass to the persistent store coordinator's addPersistentStore method

Add NSMigratePersistentStoresAutomaticallyOption and NSInferMappingModelAutomaticallyOption keys to the dictionary

NSPersistentContainer Default Configuration

SQLiteStoreType

Lightweight Migrations

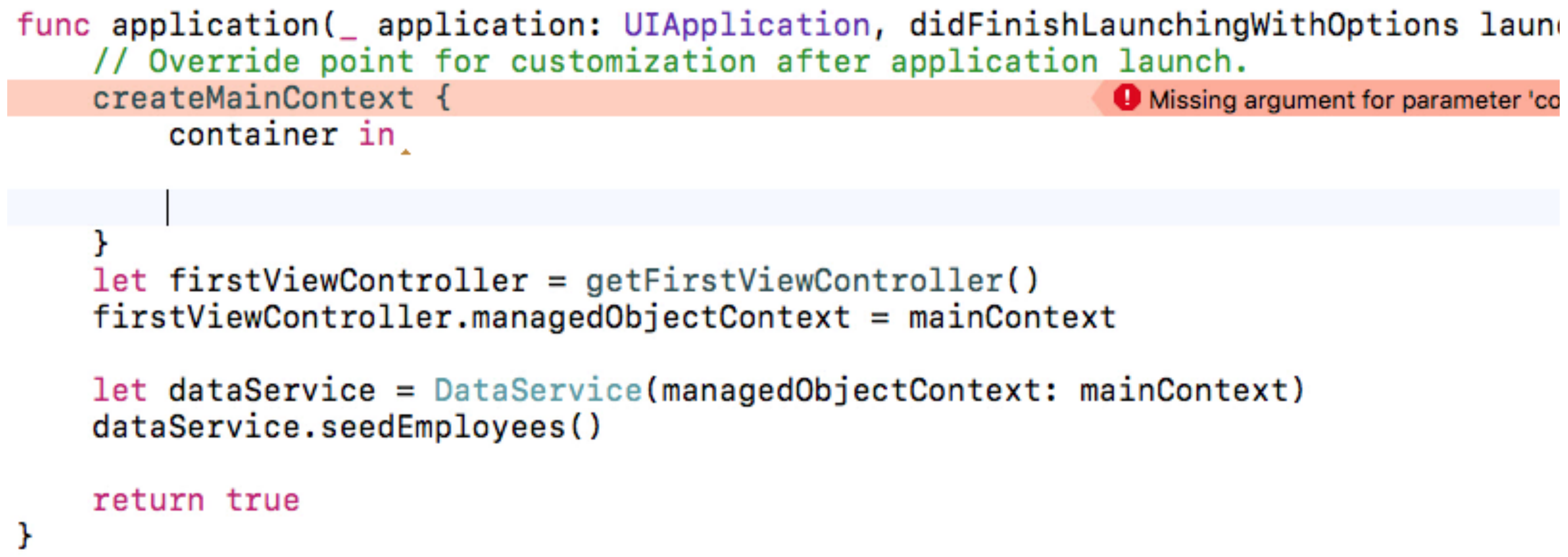


```
    // Happens asynchronously!
    container.loadPersistentStores(completionHandler: {
        persistentStoreDescription, error in

        guard error == nil else { fatalError("Failed to load store: \(error)") }

        DispatchQueue.main.async {
            completion(container)
        }
    })
}
```

loadPersistentStores finishes

completionHandler executes

calls completion passes container

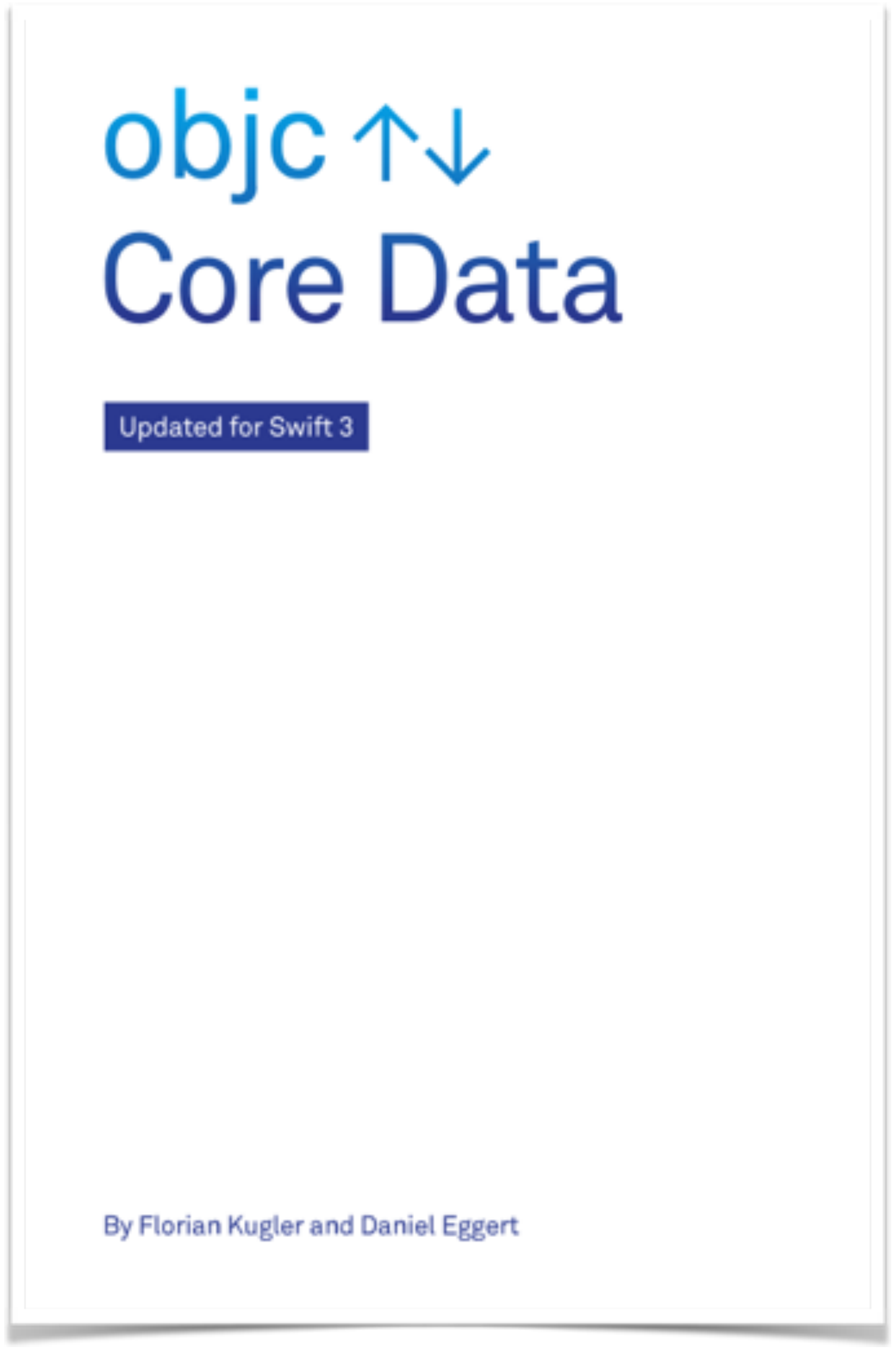


```
func application(_ application: UIApplication, didFinishLaunchingWithOptions laun
    // Override point for customization after application launch.
    createMainContext {                                   Missing argument for parameter 'co
        container in

        |
    }
    let firstViewController = getFirstViewController()
    firstViewController.managedObjectContext = mainContext

    let dataService = DataService(managedObjectContext: mainContext)
    dataService.seedEmployees()

    return true
}
```

logic inside closure passed to createMainContext executes

Conclusion - Where to Go from Here

Where to Go from Here


objc ↑↓
Core Data

Updated for Swift 3

By Florian Kugler and Daniel Eggert

- Explains more advanced topics
- Syncing with web services
- Performance & profiling tips
- Sample code

https://www.objc.io/books/core-data/

Where to Go from Here

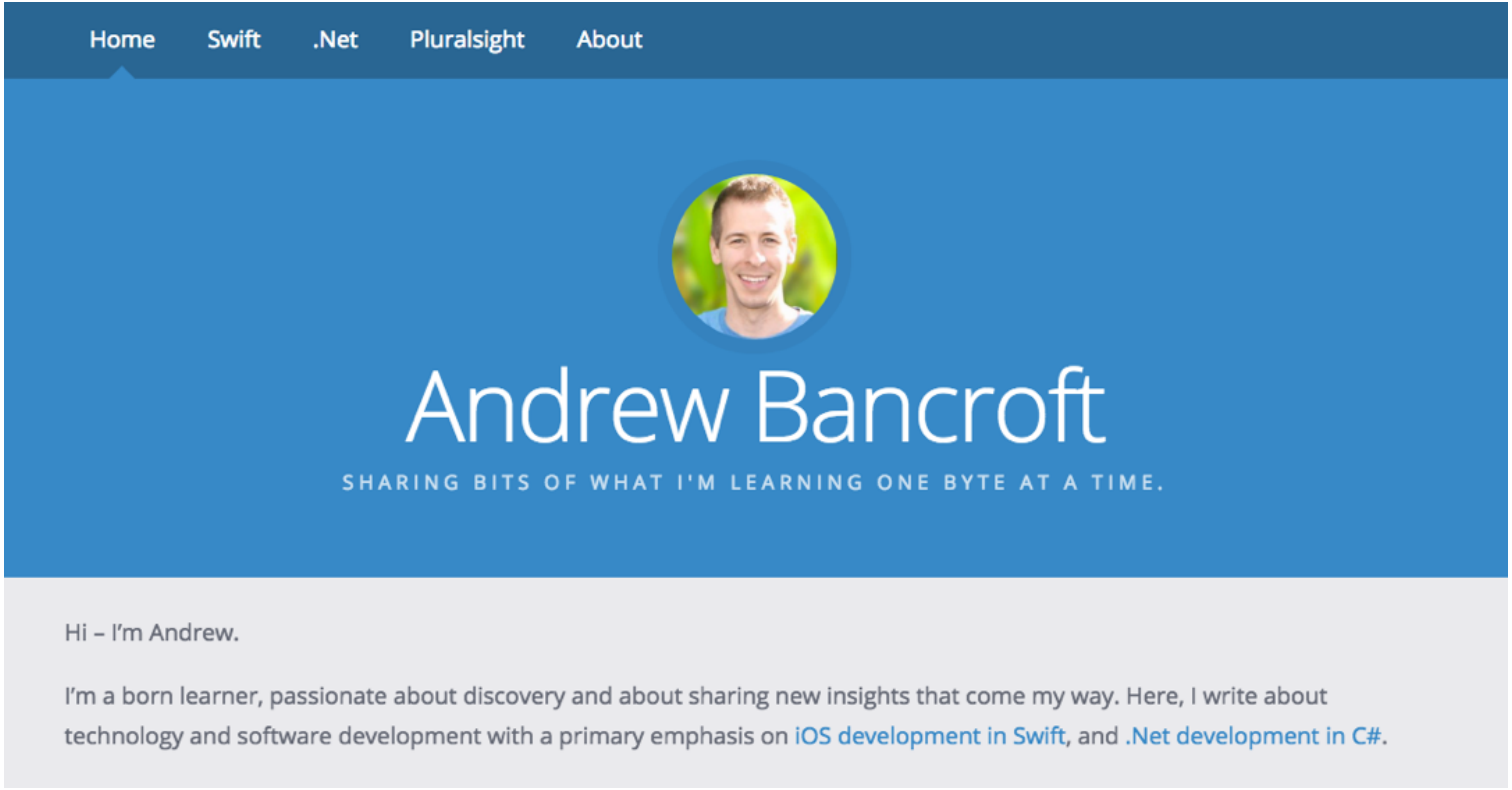
WWDC Sessions

http://bit.ly/CoreDataWWDC16

Where to Go from Here

Core Data Programming Guide

http://bit.ly/AppleCoreDataProgrammingGuide



Andrew Bancroft

SHARING BITS OF WHAT I'M LEARNING ONE BYTE AT A TIME.

Hi – I'm Andrew.

I'm a born learner, passionate about discovery and about sharing new insights that come my way. Here, I write about technology and software development with a primary emphasis on iOS development in Swift, and .Net development in C#.

www.andrewcbancroft.com

@andrewcbancroft