Setting Up the Core Data Stack

Andrew Bancroft

@andrewcbancroft www.andrewcbancroft.com

The Core Data Stack

Model	Represent
Manipulate	Persist

Summary

Introduce and explain the Core Data stack

Architect for testability and maintainability

Understand application flow and Core Data interaction

Be aware of iOS 10 and macOS Sierra changes

Explaining the Core Data Stack

Explaining the Core Data Stack

Model

NSManagedObjectModel

Manage

NSManagedObject

NSManagedObjectContext

Persist

NSPersistentStoreCoordinator

NSPersistentStore

Explaining the Core Data Stack

Model	Manage	Persist
NSManagedObjectModel	NSManagedObject	NSPersistentStoreCoordinator
	NSManagedObjectContext	NSPersistentStore

Explaining the Core Data Stack

Model

Manage

Persist

NSManagedObjectModel

NSManagedObject

NSManagedObjectContext

NSPersistentStoreCoordinator

NSPersistentStore

ShoutOut NSManagedObjects

ShoutOut

Employee

NSManagedObjectContext

Create	Change	Track Changes
Delete	Query	Save

Explaining the Core Data Stack

Model

NSManagedObjectModel

Manage

NSManagedObject

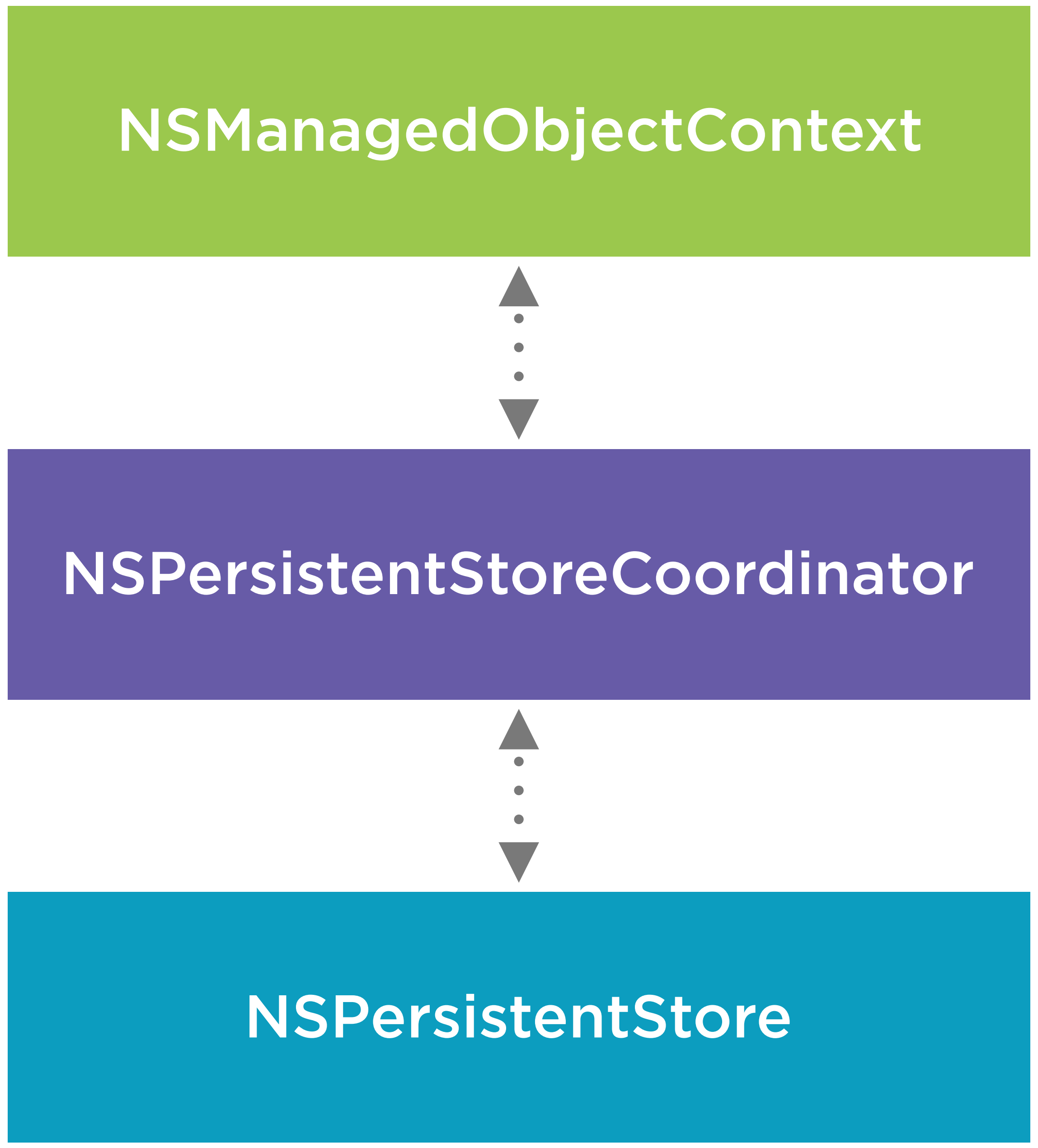
NSManagedObjectContext

Persist

NSPersistentStoreCoordinator

NSPersistentStore

Explaining the Core Data Stack



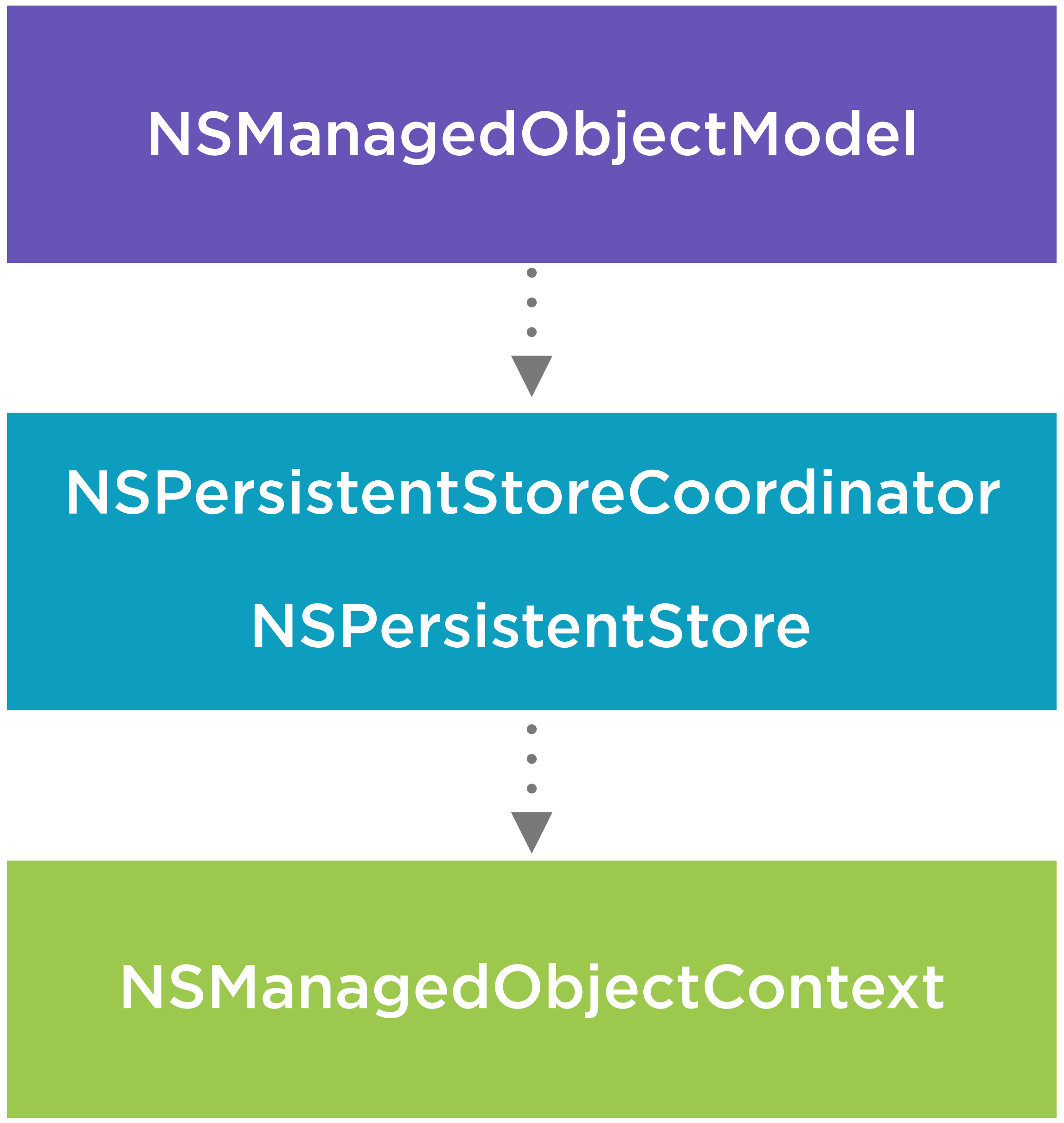
Setting Up the Core Data Stack in Code

Setting Up the Core Data Stack In Code

Step 1

Step 2

Step 3



Demo

Create data model

Initialize NSManagedObjectModel instance

Configure NSPersistentStoreCoordinator

Add NSPersistentStore to the coordinator

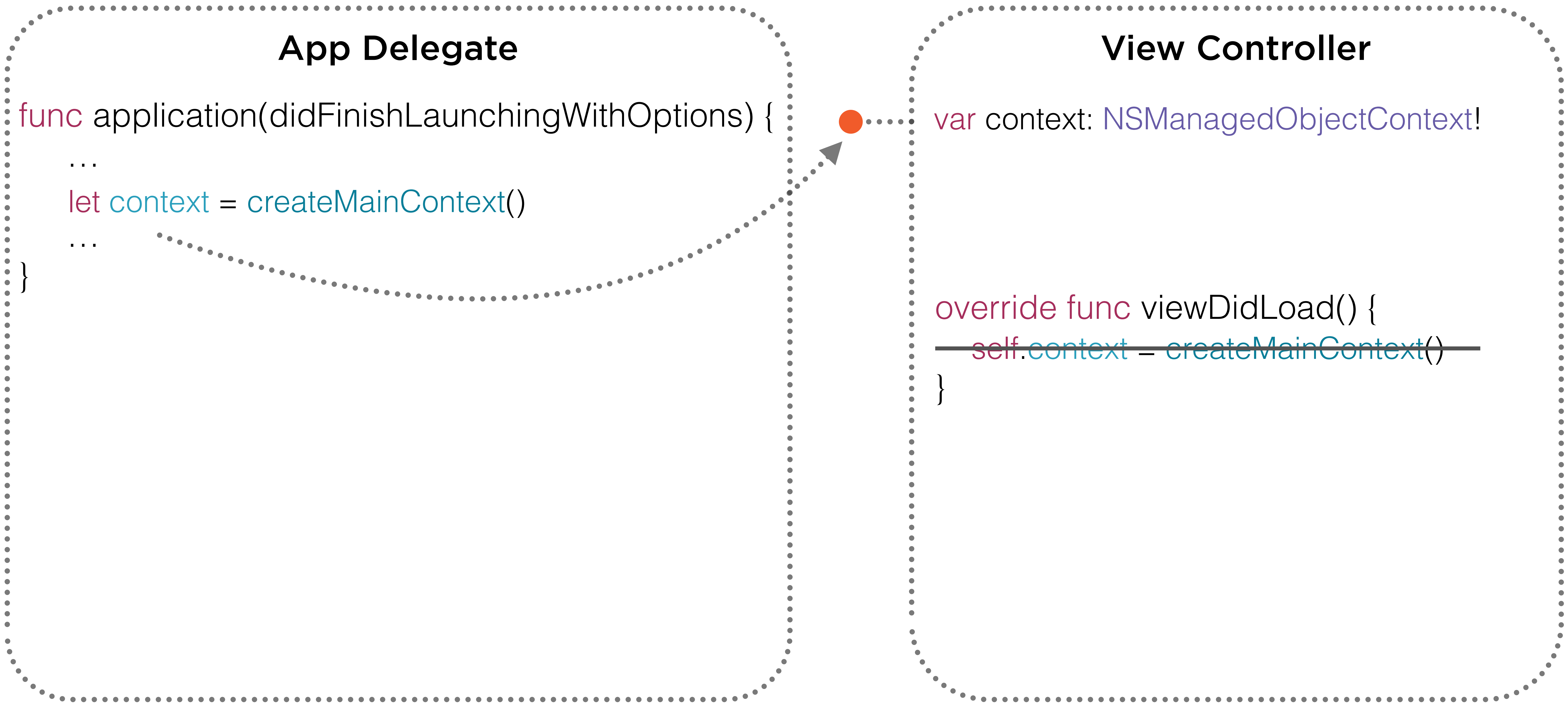
Initialize NSManagedObjectContext

Architecting for Testing and Maintenance

The Key to Unit Testing with Core Data

Dependency Injection

Architecting for Testing and Maintenance



App Delegate

```
func application(didFinishLaunchingWithOptions) {
    ...
    let context = createMainContext()
    ...
}
```

View Controller

```
var context: NSManagedObjectContext!



override func viewDidLoad() {
    ~~self.context = createMainContext()~~
}
```

Demo

Set up dependency injection architecture for the app

Write a unit test to demonstrate the testability of this architecture

Understanding Application Flow

Understanding Application Flow

ShoutOutDraftsViewController

ShoutOutEditorViewController

ShoutOutDetailsViewController

Understanding Application Flow

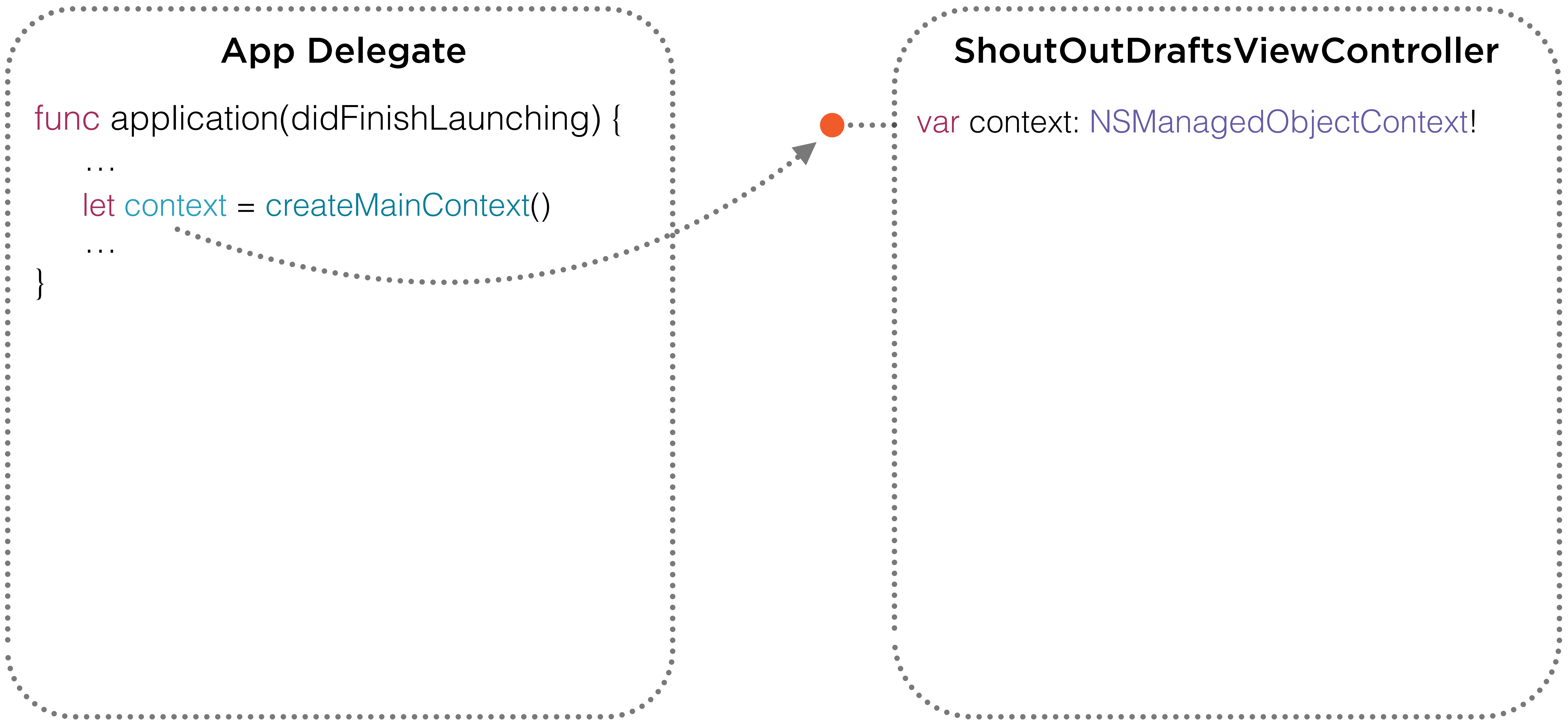
ShoutOutEditorViewController

```
var context: NSManagedObjectContext!



override func viewDidLoad() {
    ~~self.context = createMainContext()~~
}
```

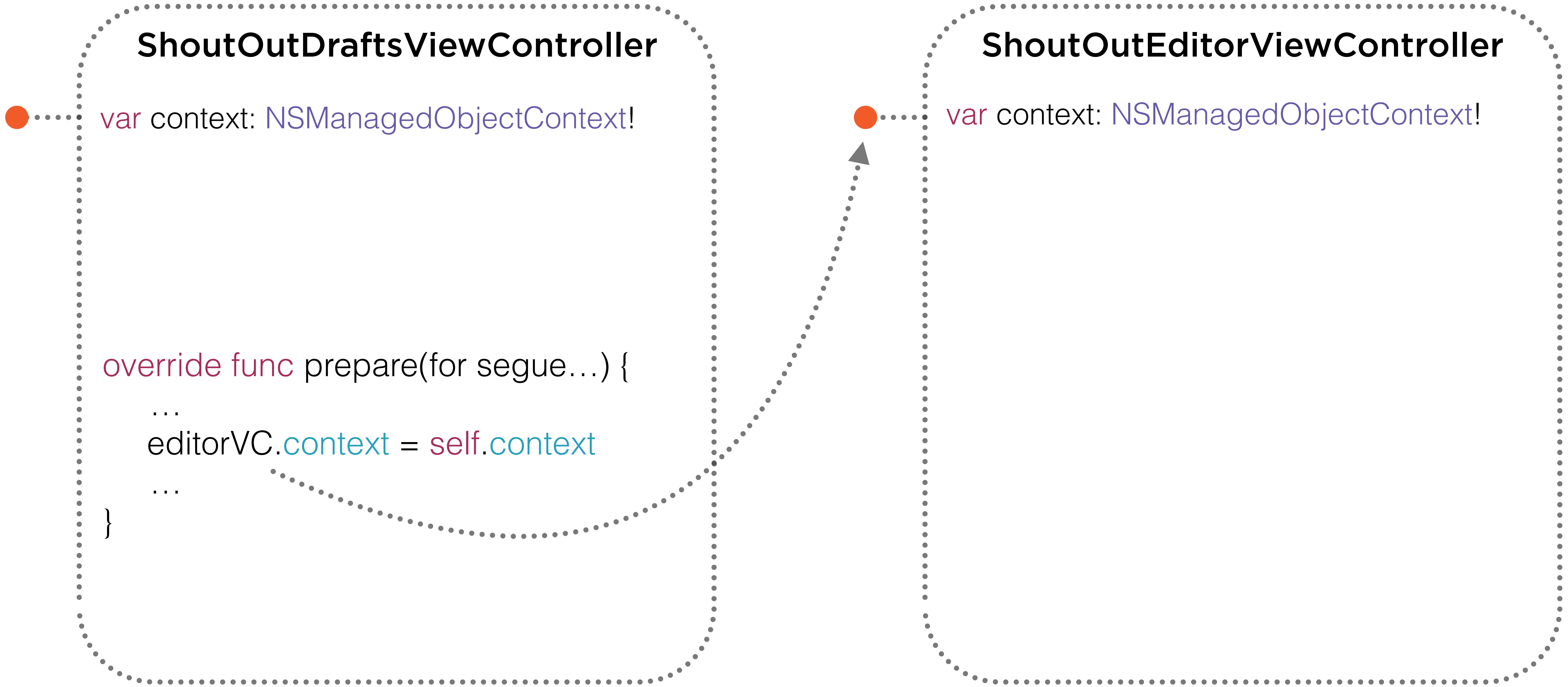
Understanding Application Flow



```
func application(didFinishLaunching) {
     . . .
     let context = createMainContext()
     . . .
}
```

```
var context: NSManagedObjectContext!
```

Understanding Application Flow



```
var context: NSManagedObjectContext!

override func prepare(for segue…) {
      …
      editorVC.context = self.context
      …
}
```

```
var context: NSManagedObjectContext!
```

Understanding Application Flow

ShoutOutDraftsViewController

```
class shoutOutDraftsViewController: UIViewController {

    var context: NSManagedObjectContext!


}
```

ManagedObjectContextDependentType Protocol

Understanding Application Flow

ShoutOutDraftsViewController

```
class shoutOutDraftsViewController: UIViewController, ManagedObjectContextDependentType {

        var context: NSManagedObjectContext!



}
```

Understanding Application Flow

ShoutOutDraftsViewController

```
class shoutOutDraftsViewController: UIViewController, ManagedObjectContextDependentType {



}
```

Understanding Application Flow

ShoutOutDraftsViewController

```
class shoutOutDraftsViewController: UIViewController, ManagedObjectContextDependentType {

✓       var context: NSManagedObjectContext!



}
```

Demo

Implement and use protocol: ManagedObjectContextDependentType

Use prepare(for segue) to pass NSManagedObjectContext instance between view controllers

Summary

Implemented Core Data stack

Architected for testability
and maintainability

Utilized prepare(for segue) to pass NSManagedObjectContext instance from one view controller to another

Clarified application flow with Swift protocol

Coming up… Creating a data model