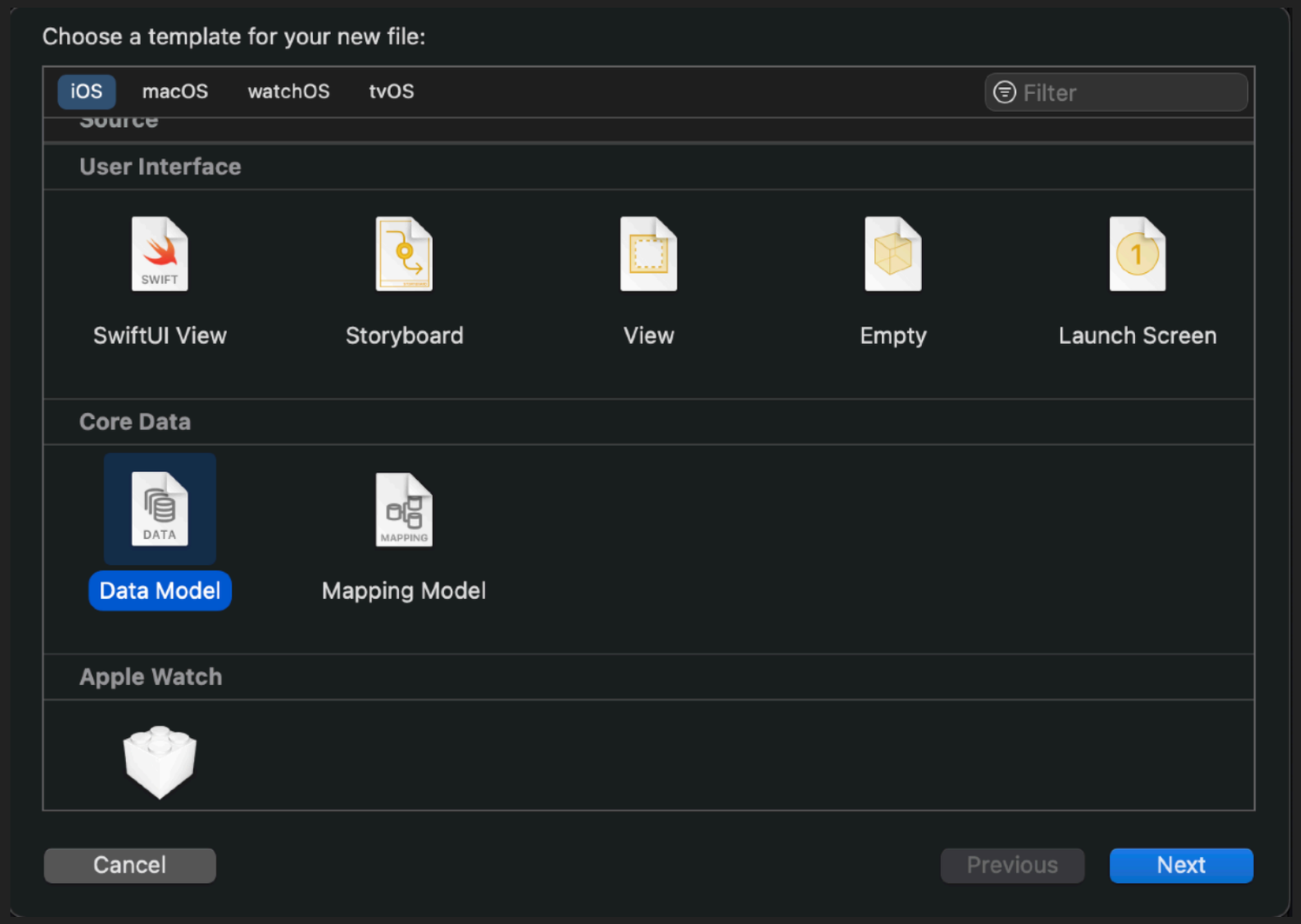
A.STEFANCHUK IT STEP

CORE DATA GUIDE

NATIVE IOS DATABASE

MODEL CREATION (STEP 1 – CREATE MODEL FILE)

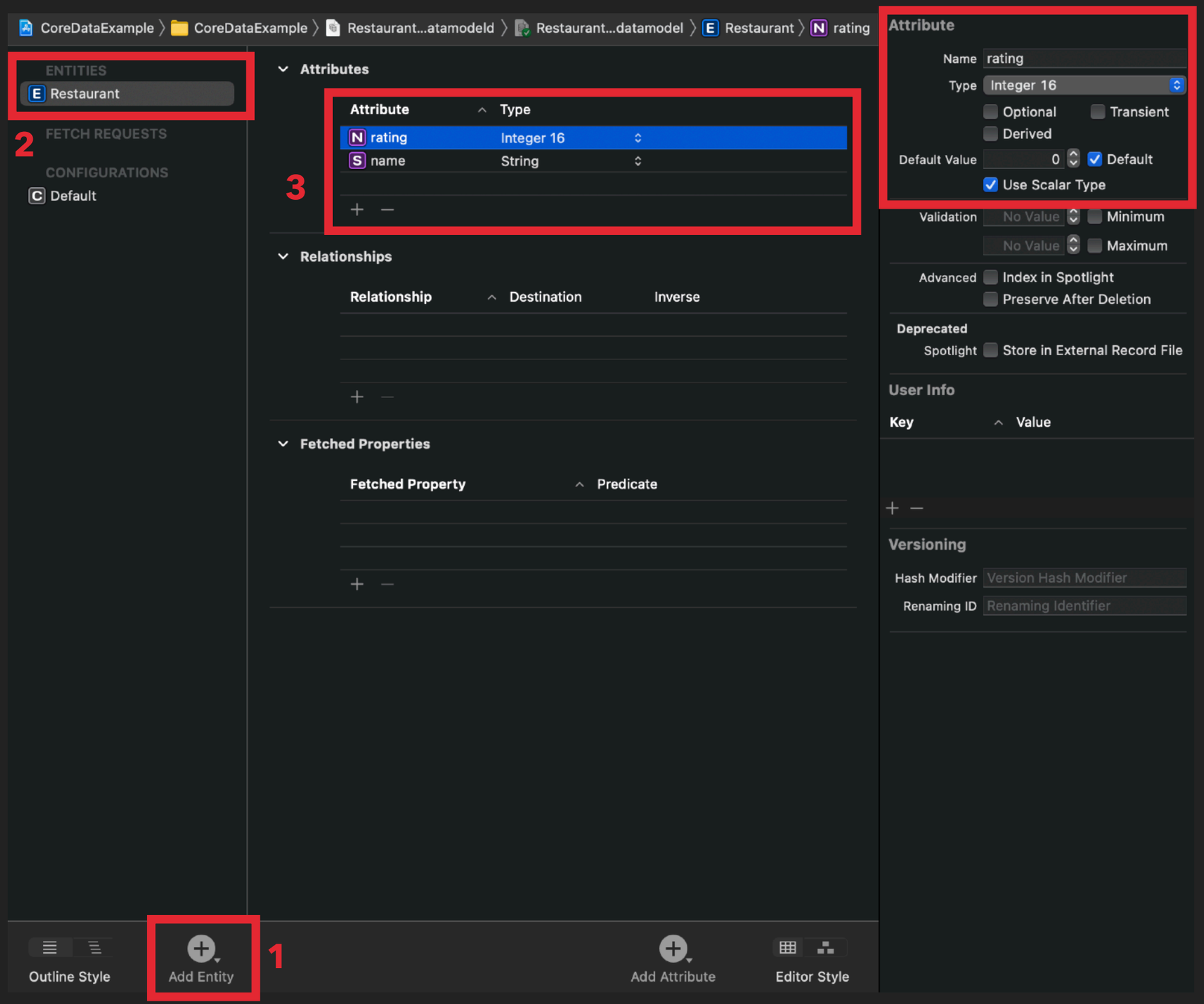
- Start creating new file
- Select “Data Model” in “Core Data” section



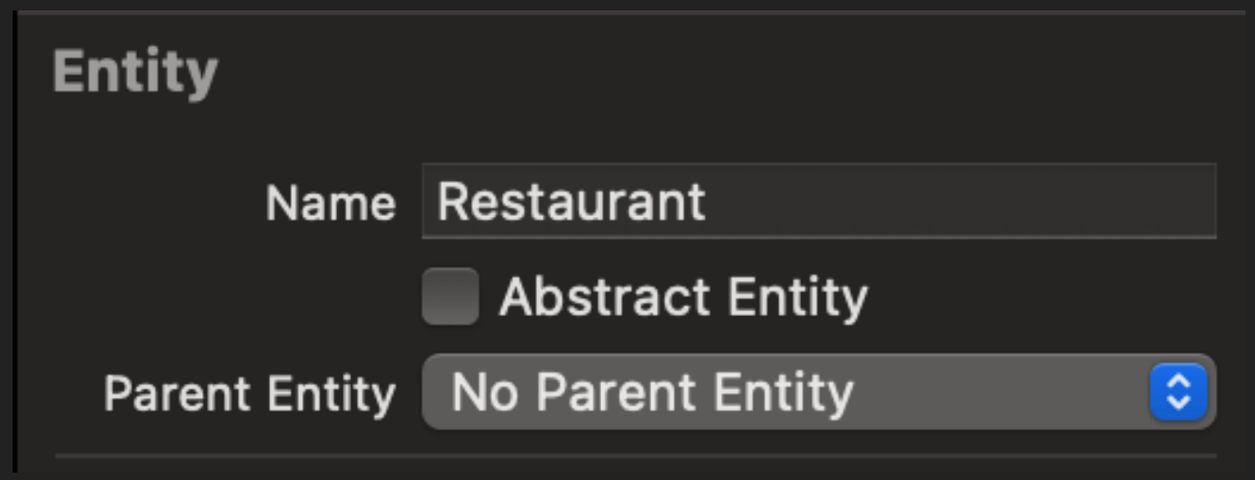
- Name it as you want (like Restaurants, Notes)

MODEL CREATION (STEP 2 – CREATE ENTITIES AND PROPERTIES)

- 1. Create new entity
- 2. Edit name of entity
- 3. Add new properties and select it's type (String, Int)
- 4. Edit property values (optional, default value)

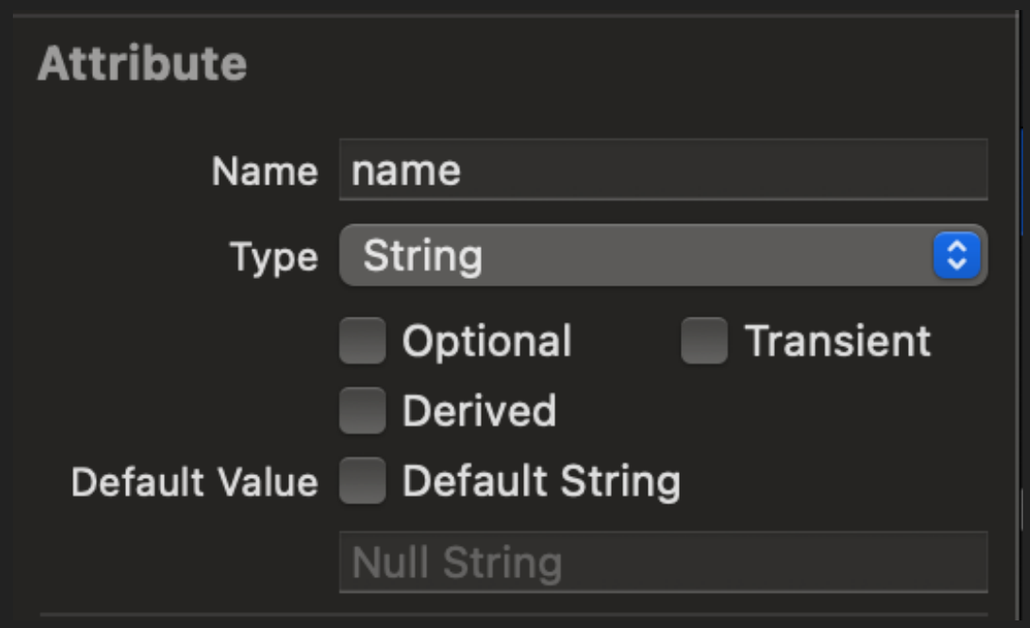


MODEL CREATION (JUST INFO)



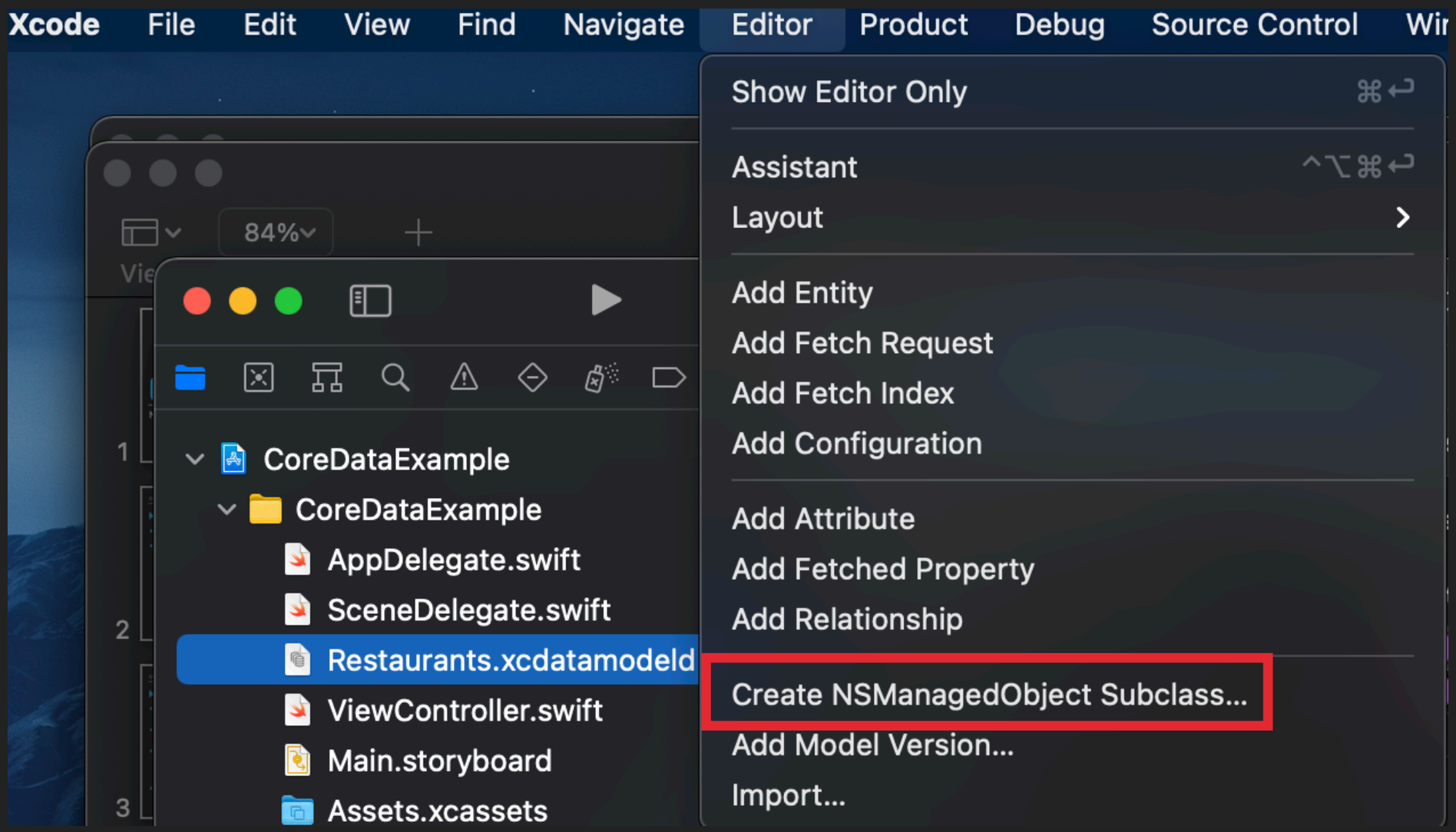
- After entity selection you can:
- 1. Set entity as abstract (no need for you now)
- 2. Set parent entity when using inheritance

MODEL CREATION (JUST INFO)



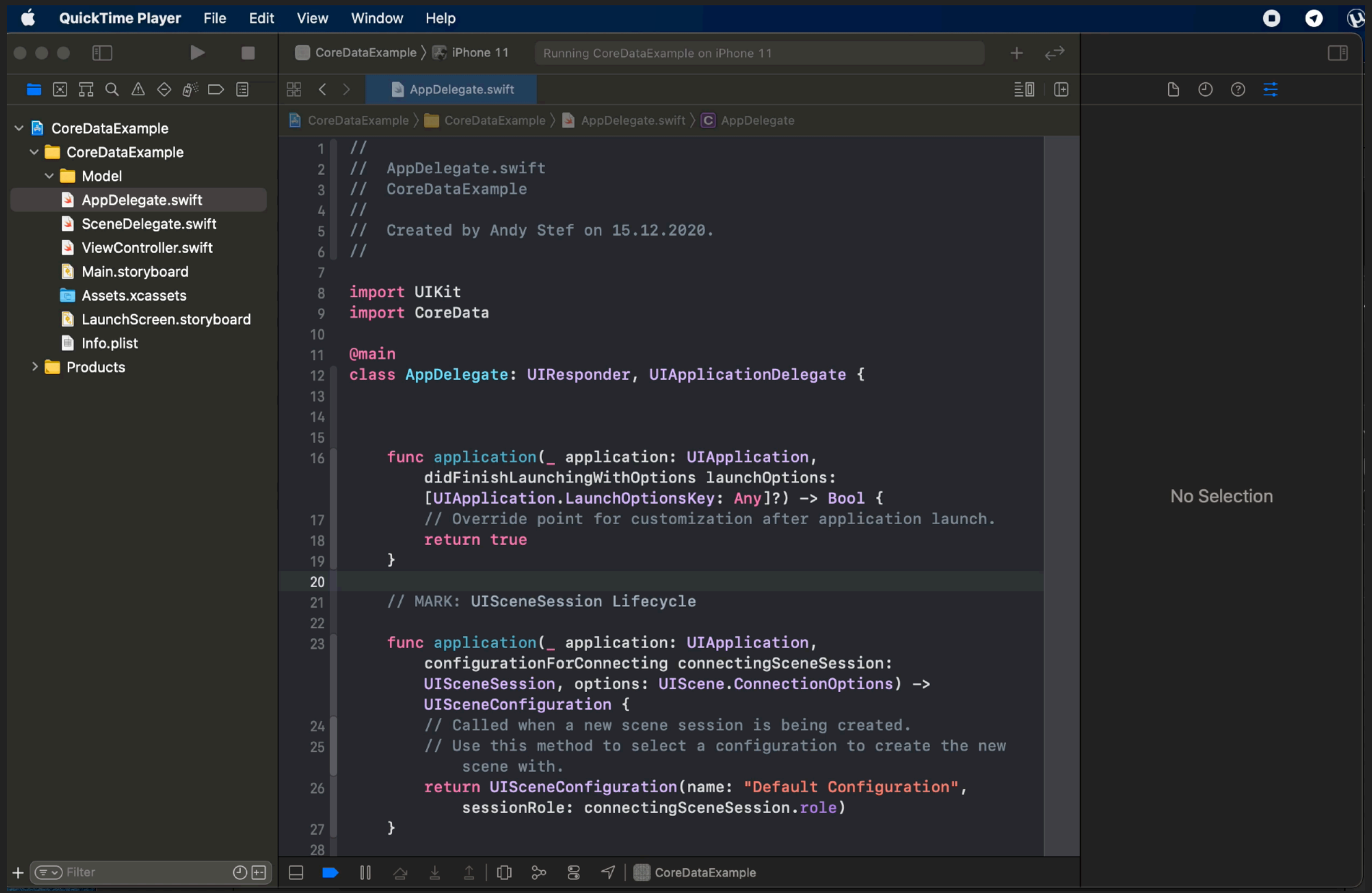
- After attribute selection you can:
- 1. Set attribute as optional
- 2. Set default value
- 3. Transient (it’s like computed property - you don’t need it)

MODEL CREATION (STEP 3 – GENERATE CLASSES)



Use this to generate Swift files and use Entities in your code

MODEL CREATION TOGETHER (VIDEO 1)



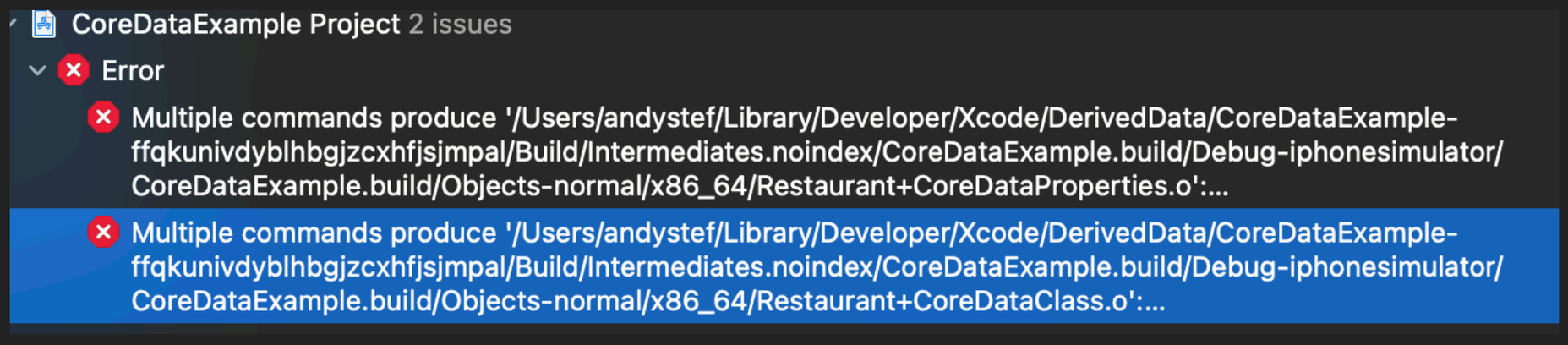
CREATE PERSISTENT CONTAINER

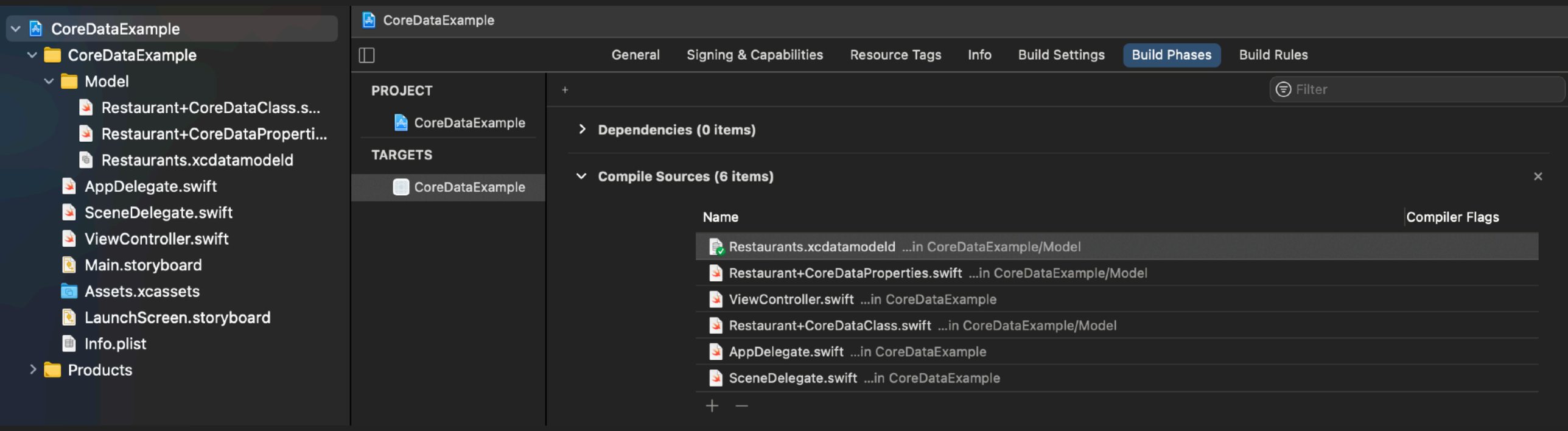
- In AppDelegate file:
- 1. Add import CoreData at the top
- 2. Inside AppDelegate add this code:

```
// MARK: - Core Data Stack -

lazy var persistentContainer: NSPersistentContainer = {
    // name should be the same as name of the created file
    let container = NSPersistentContainer(name: "Restaurants")
    container.loadPersistentStores { storeDescription, error in
        if let error = error {
            print(error.localizedDescription)
        }
    }

    return container
}()
```

BUILD PROJECT

If you have such error

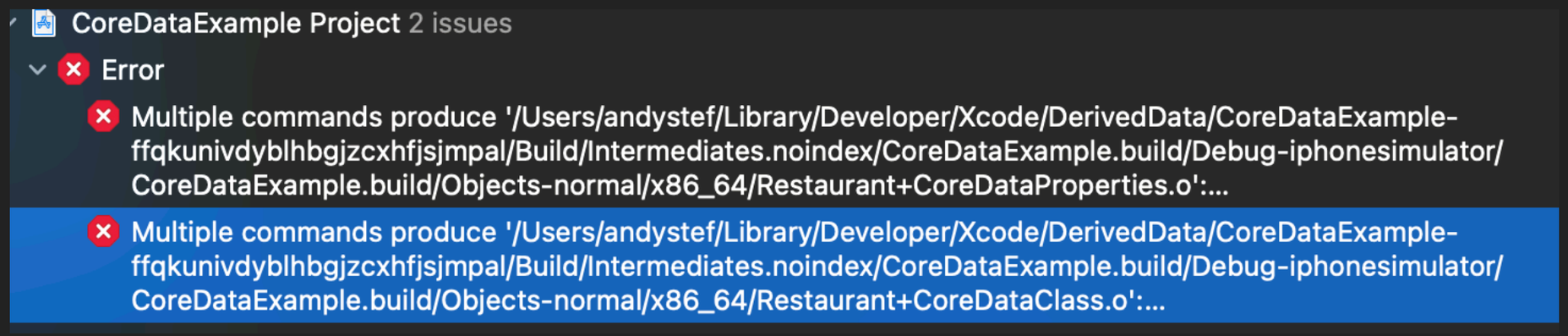


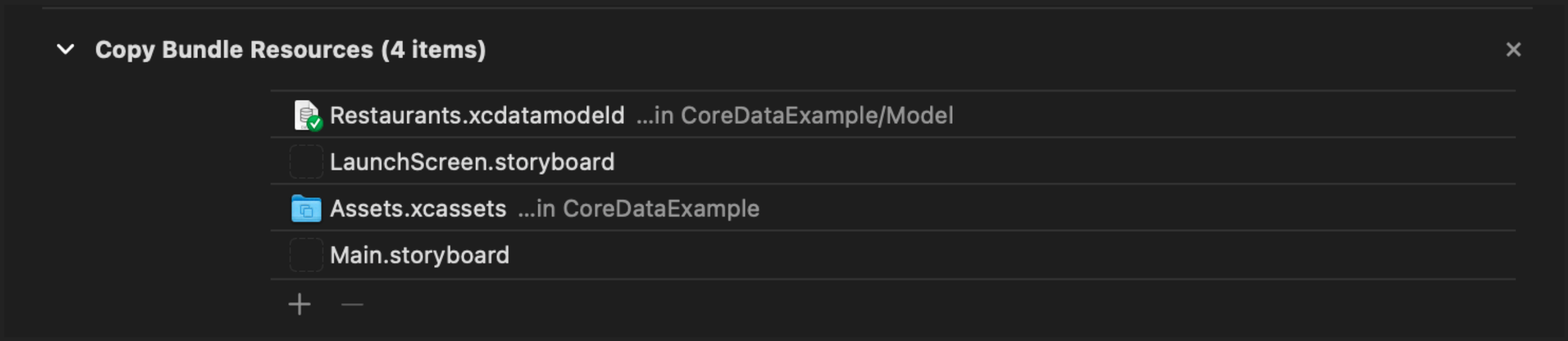


Press - when Restaurants.xcdatamodel selected

BUILD PROJECT

If you have such error





Then -> add this file to Copy bundle resources

CREATE DATABASE PROTOCOL

```
protocol Database {
    func create<T: DatabaseObject>(type: T.Type) -> T?
    func fetchObjects<T: DatabaseObject>(of type: T.Type) -> [T]
    func delete(_ object: DatabaseObject)
    func saveChanges()
}
```

+

```
import Foundation
import CoreData

protocol DatabaseObject {}

extension NSManagedObject: DatabaseObject {}
```

IMPLEMENT CORE DATA SERVICE

```
// Helper func for getting the current context.
private func getContext() -> NSManagedObjectContext? {
    guard let appDelegate = UIApplication.shared.delegate as? AppDelegate else { return nil }
    return appDelegate.persistentContainer.viewContext
}

func create<T>(type: T.Type) -> T? {
    guard let context = getContext() else { return nil }
    guard let model = type as? NSManagedObject.Type else { return nil }
    guard let newObject = NSManagedObject(entity: model.entity(), insertInto: context) as? T else {
        return nil
    }

    return newObject
}
```

IMPLEMENT CORE DATA SERVICE

```
func fetchObjects<T>(of type: T.Type) -> [T] {
    guard let context = getContext() else { return [] }
    guard let model = type as? NSManagedObject.Type else { return [] }

    let fetchRequest = NSFetchRequest<NSFetchRequestResult>()
    fetchRequest.entity = model.entity()
    let objects = try? context.fetch(fetchRequest) as? [T]

    return objects ?? []
}

func delete(_ object: DatabaseObject) {
    guard let context = getContext() else { return }
    guard let object = object as? NSManagedObject else { return }
    context.delete(object)
    saveChanges()
}

func saveChanges() {
    try? getContext()?.save()
}
```

SERVICE USAGE

```
var dataSource: [Restaurant] = []
let databaseService: Database = CoreDataService()
```

```
private func setupDataSource() {
    dataSource = databaseService.fetchObjects(of: Restaurant.self)
}
```

```
// object creation
guard let newRestaurant = databaseService.create(type: Restaurant.self) else {
    return
}

newRestaurant.name = name

databaseService.saveChanges()
dataSource.append(newRestaurant)
```

```
// object delete
databaseService.delete(dataSource[indexPath.row])
dataSource.remove(at: indexPath.row)
```

ADD SORTING ORDER

```
struct Sorted {
    var key: String
    var ascending: Bool = true
}
```

```
func fetchObjects<T: DatabaseObject>(of type: T.Type, sortDescriptor: Sorted?) -> [T]
```

```
if let sortDescriptor = sortDescriptor {
    let nsSortDescriptor = NSSortDescriptor(key: sortDescriptor.key, ascending: sortDescriptor.ascending)
    fetchRequest.sortDescriptors = [nsSortDescriptor]
}
```

```
private func setupDataSource() {
    dataSource = databaseService.fetchObjects(of: Restaurant.self,
                                              sortDescriptor: Sorted(key: "rating", ascending: false))
}
```

ADD FILTERING OF RESULTS

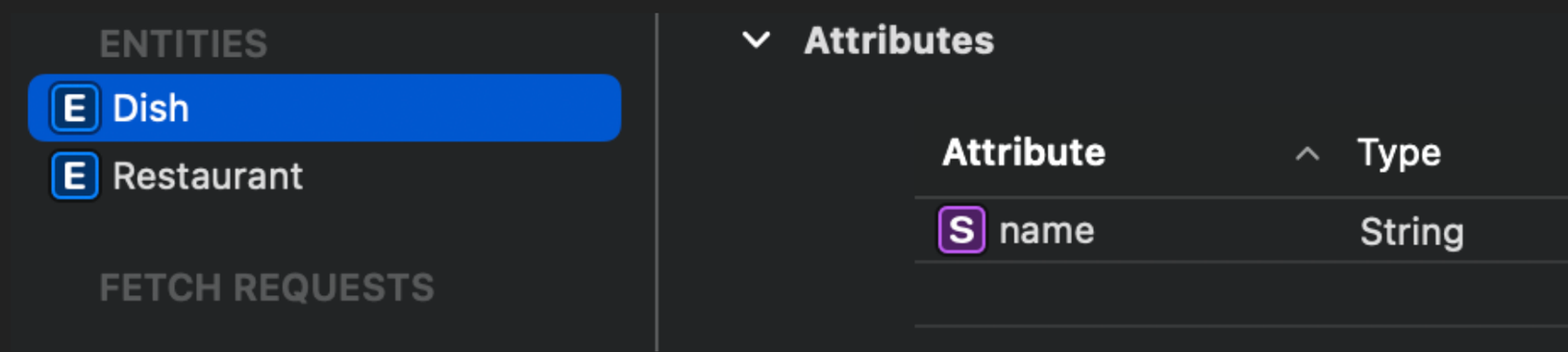
```
func fetchObjects<T: DatabaseObject>(of type: T.Type, sortDescriptor: Sorted?, predicate: NSPredicate?) -> [T]
```

```
if let predicate = predicate {
    fetchRequest.predicate = predicate
}
```

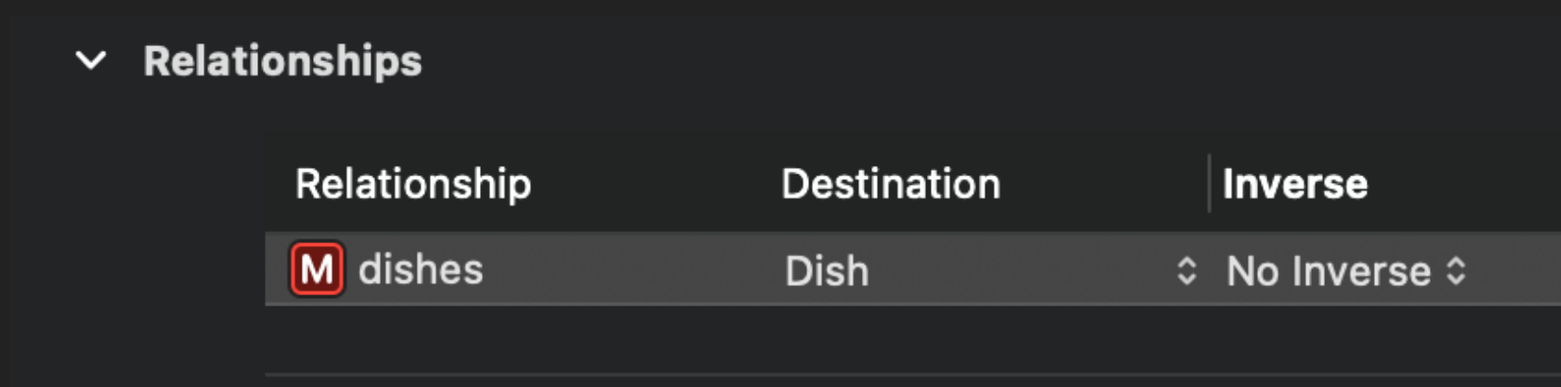
```
private func setupDataSource() {
    dataSource = databaseService.fetchObjects(of: Restaurant.self,
                                              sortDescriptor: Sorted(key: "rating", ascending: false),
                                              predicate: NSPredicate(format: "rating > 3"))
}
```

ADDING RELATIONSHIP

Add new entity



In entity that contains new type(Restaurant) add:



ADDING RELATIONSHIP

Add reverse connection(in Dish)

Relationships

Relationship	Destination	Inverse
O restaurant	Restaurant	No Inverse

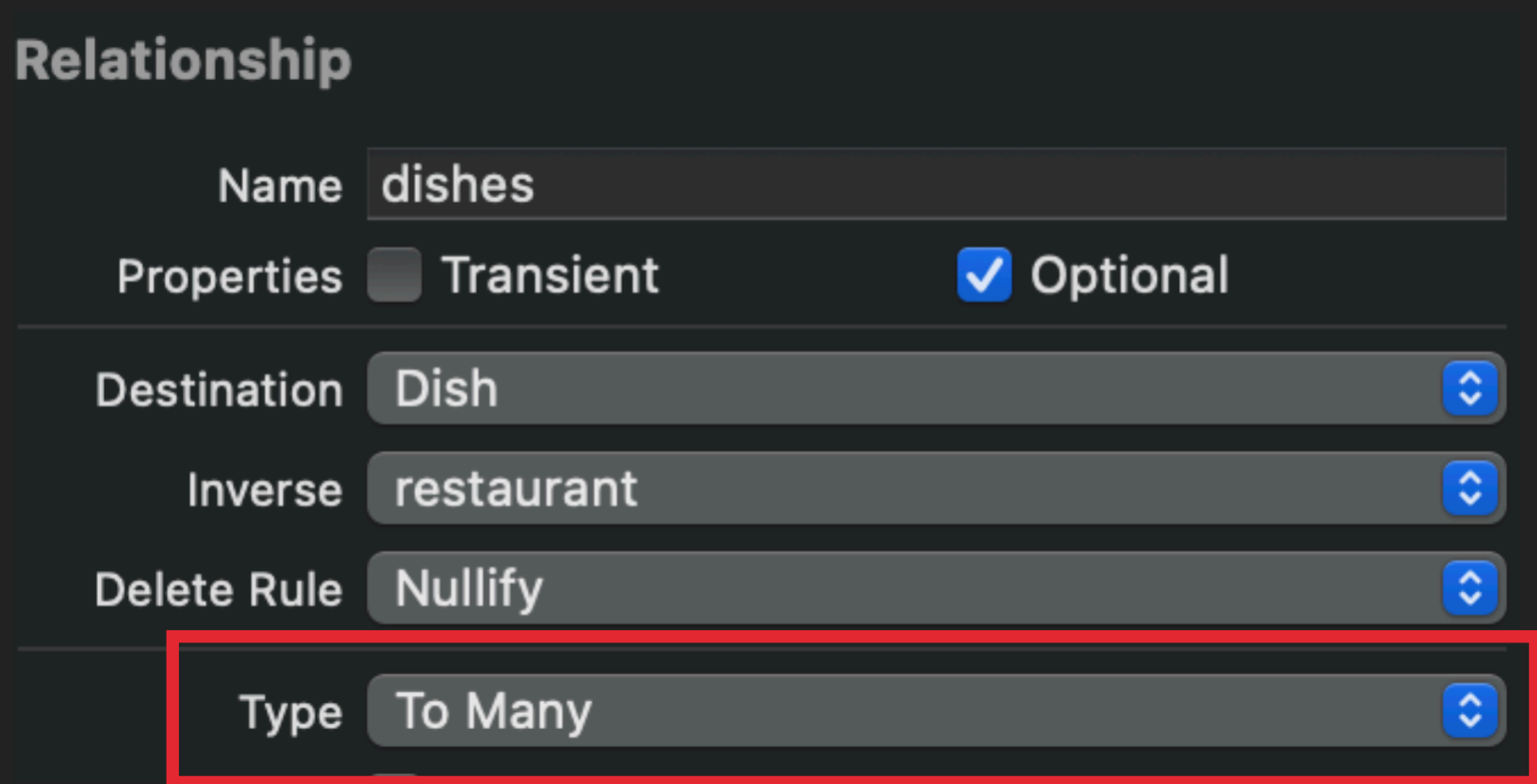
Update inverse value (in Restaurant)

Relationships

Relationship	Destination	Inverse
M dishes	Dish	restaurant

ADDING RELATIONSHIP

Set Relation Type to “To Many”



Generate updated classes

TIPS

- If you add some new entities or new attributes to your model file, you should:
- Generate class again
- Delete app on simulator

LINKS

- Tutorial: https://www.raywenderlich.com/7569-getting-started-with-core-data-tutorial
- Core Data vs Realm: https://agilie.com/en/blog/coredata-vs-realm-what-to-choose-as-a-database-for-ios-apps