CoreData

Persistence Store

-DNREDDi

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

Data Persistence

Data Persistence is a process of saving the application data between the application launches.

- Data in the iOS applications' can be saved in different ways.
- Variables (No retention of data once after application terminates)
- NSUserDefaults
- PropertyLists
- SQLite
- CoreData
- Realm

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com

CoreData is a **Persistence Store,** used to save the application data between application launches.

It is actually a framework that lets developers store (or retrieve) data in database in an **object-oriented** way.

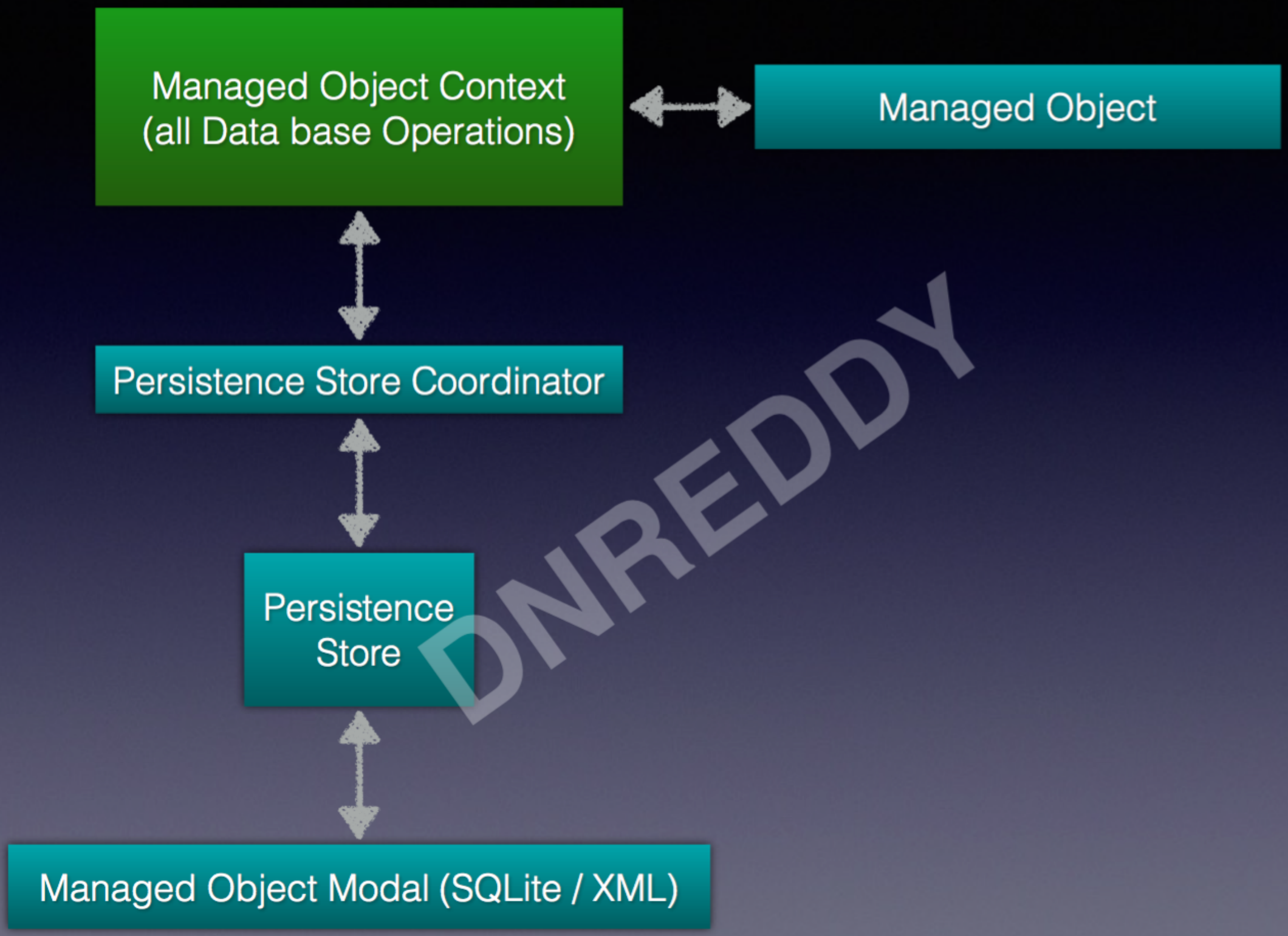
With Core Data, you can easily map the objects in your apps to the table records in the database without even knowing any SQL.

Core Data by default uses **sqlite** database for storing data.

Core Data is not a database like oracle,SQL etc..

Features

CoreData is an objective-c-based wrapper around SQLite.
Faster
Proper memory management
Improves responsive time
Supported Datatypes: Integer16, 32, 64, String, Decimal, Double, Boolean, Binary Data, Transformable.



CoreData Stack

Managed Object Modal

It describes the **schema** that you use in the app. If you have a database background, think of this as the database schema.

However, the schema is represented by a collection of objects (also known as entities). In Xcode, the Managed Object Model is defined in a file with the extension **.xcdatamodeld**. You can use the visual editor to define the entities and their attributes, as well as, relationships.

Persistence Store Coordinator

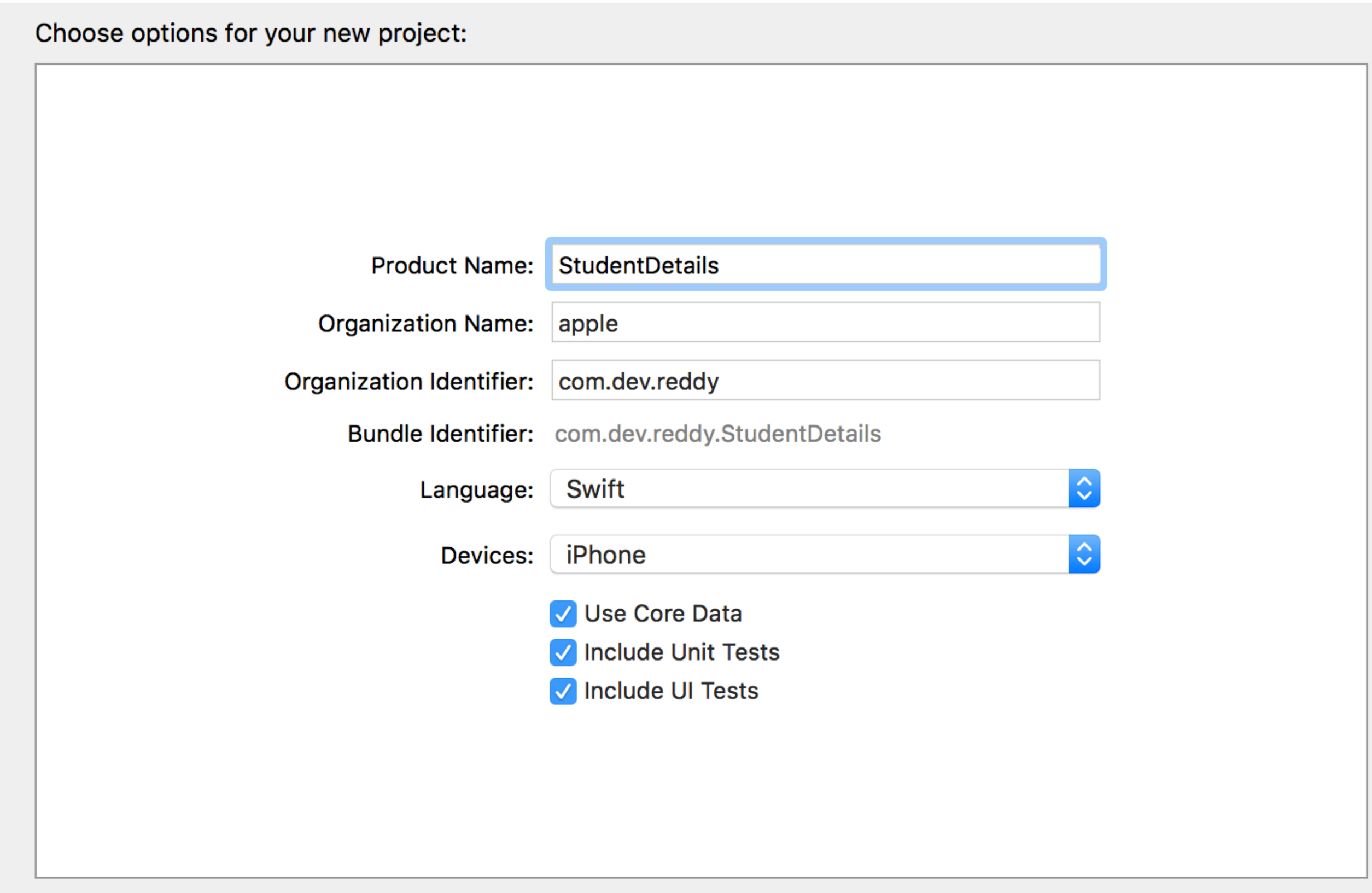
SQLite is the default persistent store in iOS. However, Core Data allows developers to setup multiple stores containing different entities. The Persistent Store Coordinator is the party responsible to manage different persistent object stores and save the objects to the stores. Forget about it you don't understand what it is. You'll not interact with Persistent Store Coordinator directly when using Core Data.

Managed Object Context

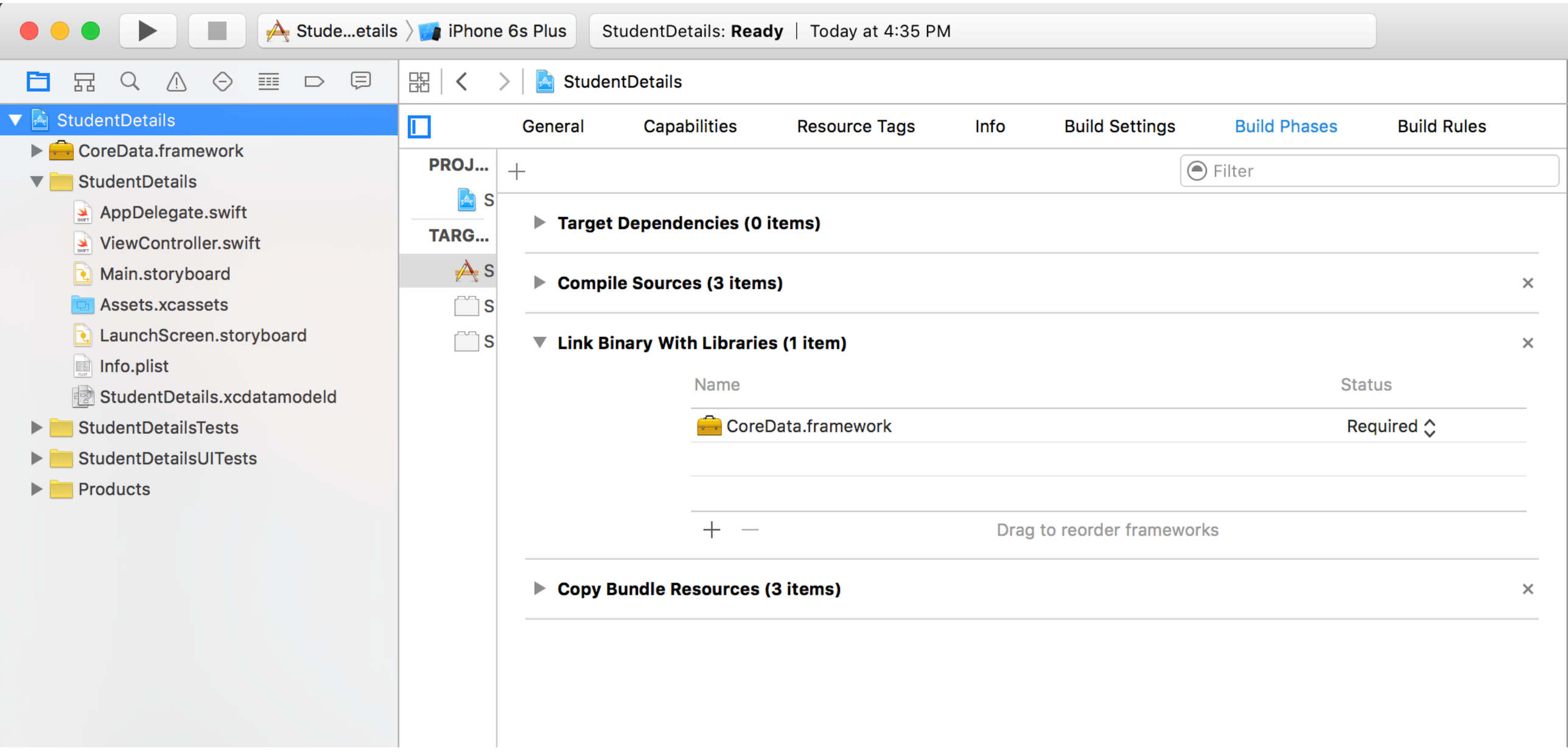
Think of it as a "**scratch pad**" containing objects that interacts with data in persistent store. Its job is to manage objects created and returned using Core Data. Among the components in the Core Data Stack, the Managed Object Context is the one you'll work with for most of the time. In general, whenever you need to fetch and save objects in persistent store, the context is the first component you'll talk to.

Steps to use Core Data

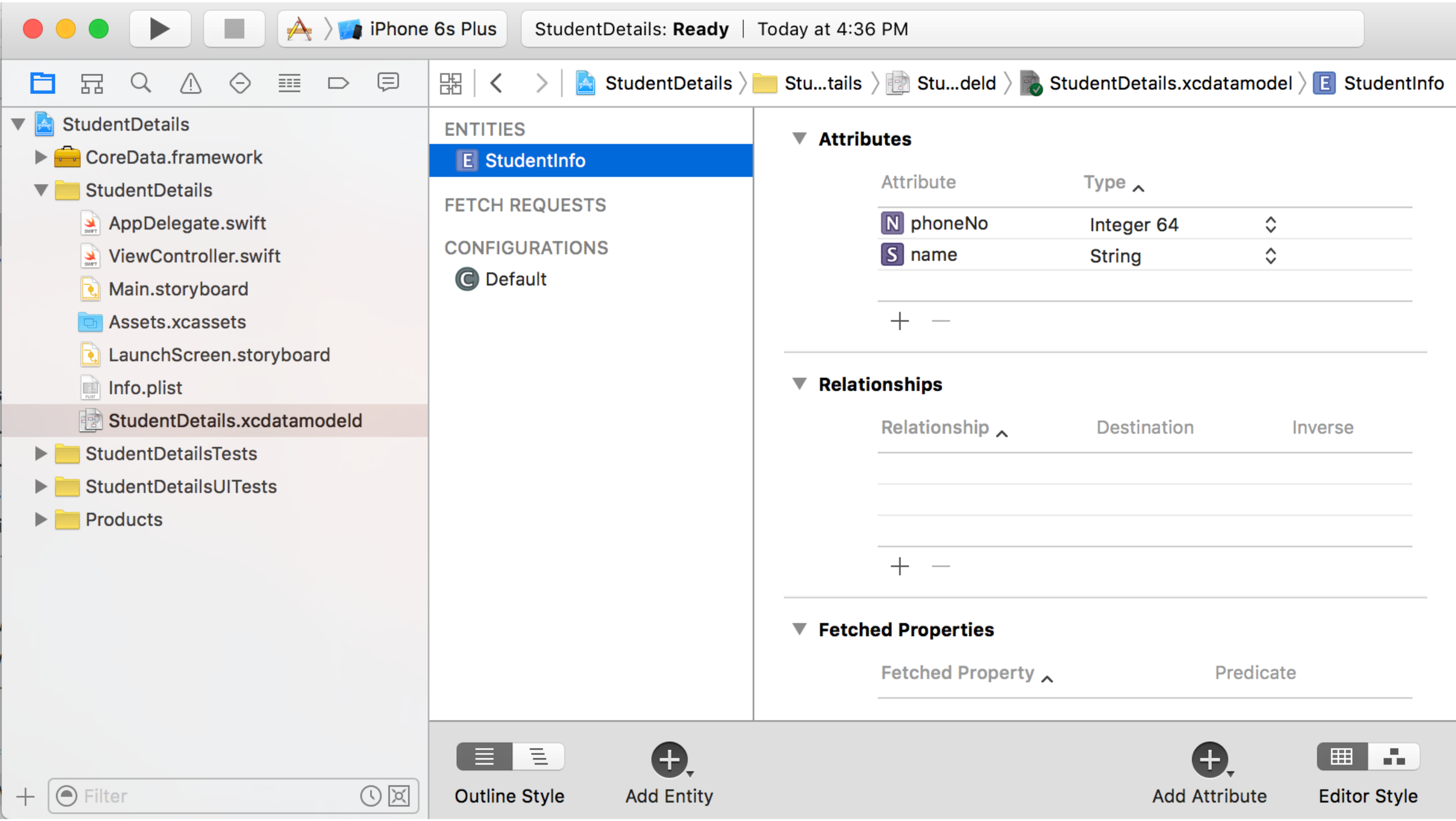
Enable Core Data while creating project.



Add Coredata.framework to the project by selecting Build Phases -> Link Binary with Libraries.



Open ProjectName.xcdatamodeld file to create Data Modals
Add Entities (here: StudentInfo), attributes (PhoneNo, Name etc) and relationships if required



- Choose the entity and create new file for that entity by choosing File-> New File -> CoreData -> ManagedObject Class
- Make sure you select swift as language
- Write a method on ViewController.swift to access the Managed Object Context object easily.

```
func getManagedObjectContext() -> NSManagedObjectContext
    {
        return ((UIApplication.sharedApplication().delegate)
as! AppDelegate).managedObjectContext;
    }
```

Core Data Operations

- **Save**
- **Fetch**
- **Update**
- **Delete**

Steps to save data in Managed Object

- Get an Object of Managed Object Context form AppDelegate
- Create a new Entity Object using NSEntityDescription
- Set all attribute values
- Save the Context

Save Student details

```
self.saveAStudentDetails(student: "Devendra", phoneNumber: 0123456987, rollNo: 1)

func saveAStudentDetails(student name: String, phoneNumber: Int, rollNo:
Int)
    {
        let context: NSManagedObjectContext =
self.getManagedObjectContext()

        let newStudent: StudentInfo =
NSEntityDescription.insertNewObjectForEntityForName("StudentInfo",
inManagedObjectContext: context) as! StudentInfo;

        newStudent.setValue(name, forKey: "name");
        newStudent.setValue(NSNumber(integer: phoneNumber), forKey:
"phoneNo");
        newStudent.setValue(NSNumber(integer: rollNo), forKey: "rollNo");

        do {
            try context.save()
        }
        catch let error as NSError
        {
            print("Could not save \(error), \(error.userInfo)")
        }
    }
```

Fetch Student Details

- Get an Object of Managed Object Context form AppDelegate
- Create a NSFetchRequest object.
- Create a query using predicate according to your requirements
- Execute the query using ManagedObjectContext Object.
- Use results

Code Snippet

```
func fetchAllStudents()
    {
        var listOfStudents: [StudentInfo]!
        let context = self.getManagedObjectContext()

        let fetchRequest = NSFetchRequest(entityName:
"StudentInfo")
        do
        {
            listOfStudents = try
context.executeFetchRequest(fetchRequest) as! [StudentInfo];
        }
        catch let error as NSError
        {
            print("Failed to fetch info", error);
        }
        for aStudent in listOfStudents
        {
            print(aStudent.name!);
            print(aStudent.phoneNo!);
            print(aStudent.rollNo!);
        }
}
```

Using Predicate

```
let fetchRequest = NSFetchRequest(entityName:
"StudentInfo")

fetchRequest.predicate = NSPredicate(format:
"phoneNo = 1234567890“, argumentArray: nil)
```

Deleting and Updating the Objects in Core Data

- Deletion: use deleteObject(aObject) method to delete the objects from Core Data

- Use setValue(newValue, forKey: attribute) method to update an existing object in core data.

Deleting Objects in Core Data

```
func deleteObjects()
    {
        var listOfStudents: [StudentInfo]!
        let context = self.getManagedObjectContext()
        let fetchRequest = NSFetchRequest(entityName: "StudentInfo")
        do
        {
            listOfStudents = try context.executeFetchRequest(fetchRequest) as!
[StudentInfo];
        }
        catch let error as NSError
        {
            print("Failed to fetch info", error);
        }
        for aStudent in listOfStudents
        {
            if aStudent.rollNo == 1
            {
                context.deleteObject(aStudent)
            }
        }

        do{
            try context.save()
        }catch let error as NSError
        {
            print("Failed to save after deleting elements: \(error)")
        }
    }
```

Updating Objects in Core Data

```
    func updateObjectInCoreData()
    {
        var listOfStudents: [StudentInfo]!
        let context = self.getManagedObjectContext()
        let fetchRequest = NSFetchRequest(entityName: "StudentInfo")
        do
        {
            listOfStudents = try context.executeFetchRequest(fetchRequest) as!
[StudentInfo];
        }
        catch let error as NSError
        {
            print("Failed to fetch info", error);
        }
        for aStudent in listOfStudents
        {
            if aStudent.name == "Devendra"
            {
                aStudent.setValue("Ravindra", forKey: "name")
            }
        }

        do{
            try context.save()
        }
        catch let error as NSError
        {
            print("Failed to save after deleting elements: \(error)")
        }
    }
```

Thank You

For Offline/ Online Training, reach me@ iPhoneDev1990@gmail.com