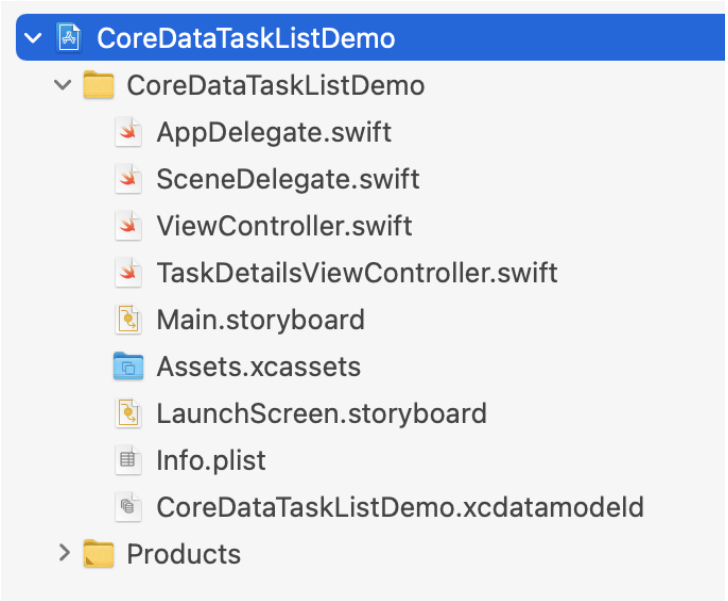
Practice: Using Core Data to Build a Task List

Download and open the starter code

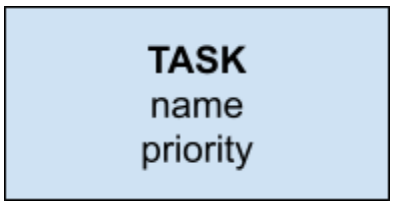
Expected project structure:

- Screen 1 - ViewController.swift
- Screen 2 - TaskDetailsViewController.swift
- Core Data model designer - CoreDataTaskListDemo.xcdatamodeld

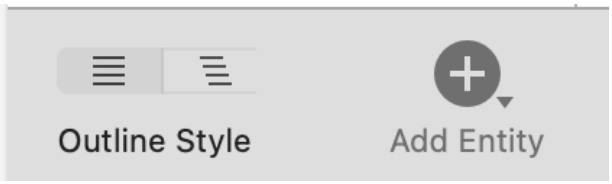


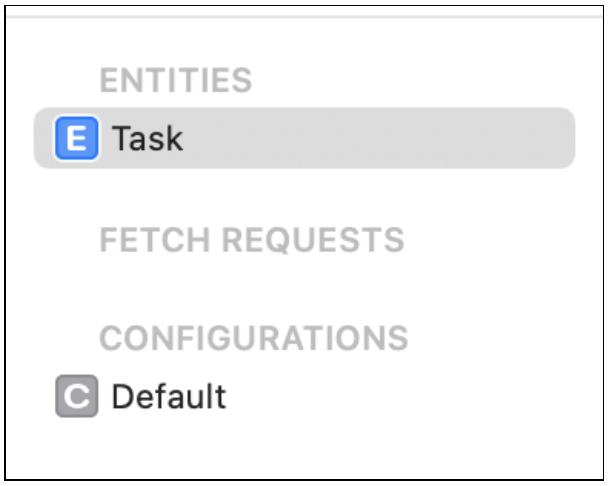
Developing a Task Entity

To represent a task in CoreData, we must create an Core Data *entity* to represent the Task



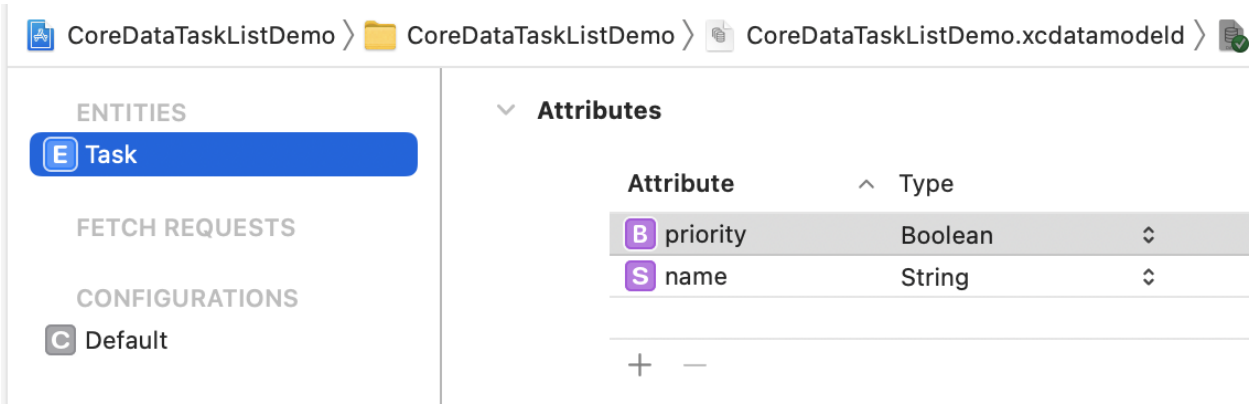
Use the Core Data model editor to add the Task Entity



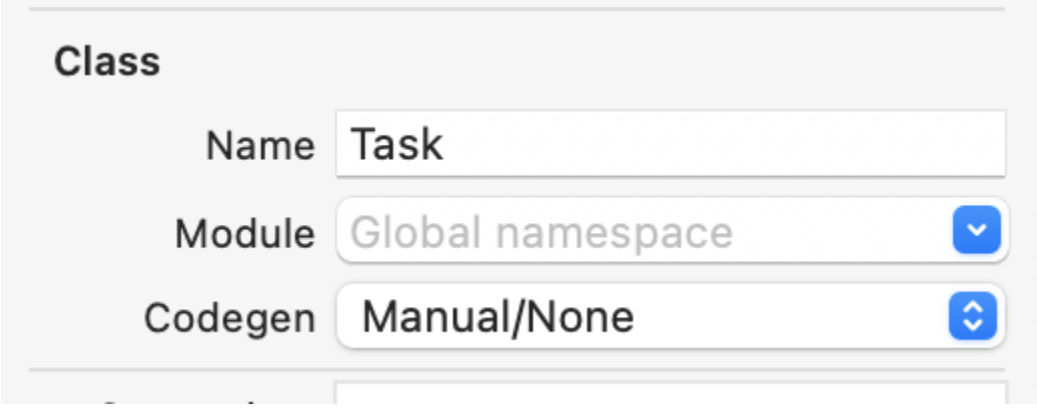


Add attributes for the task

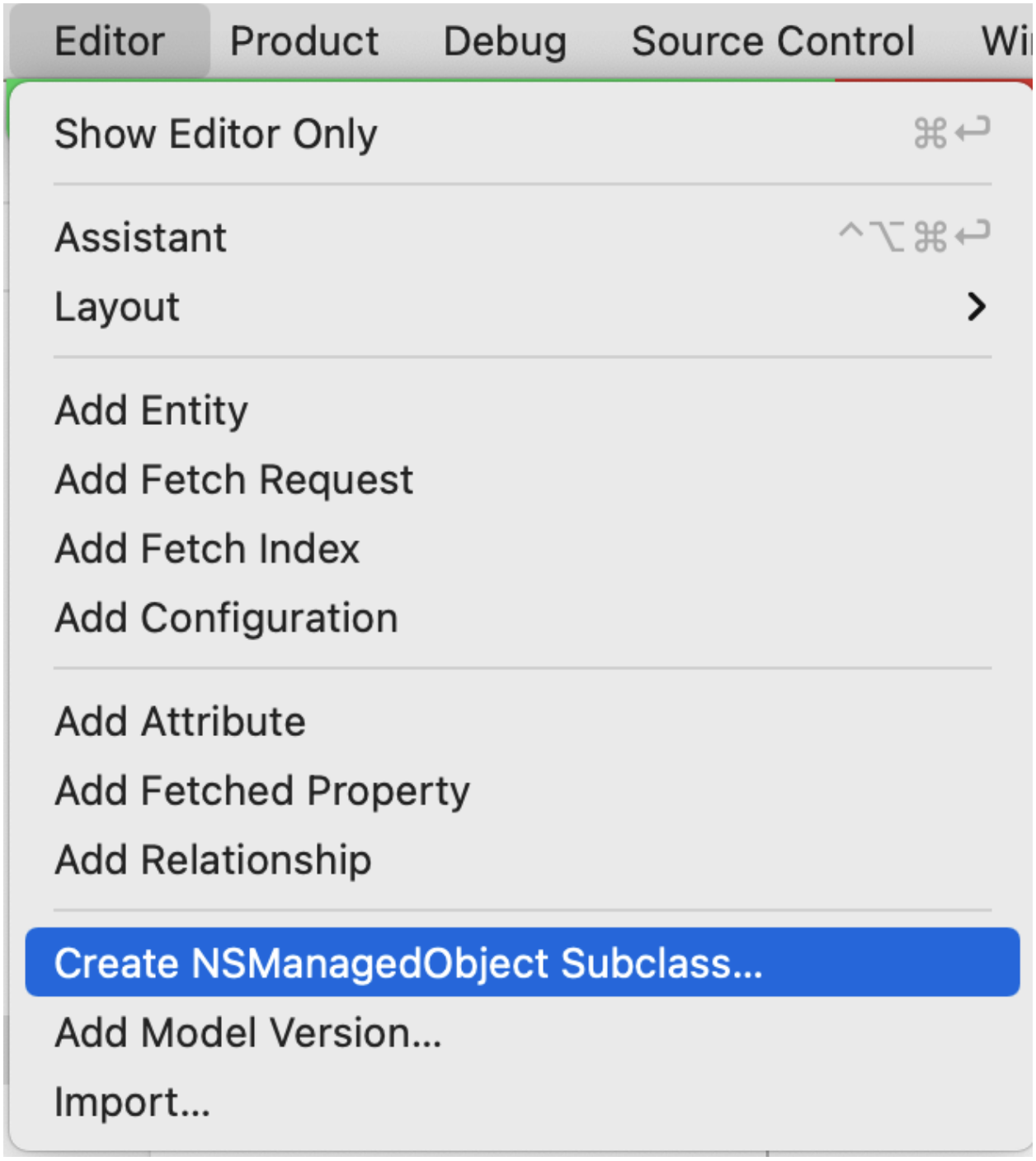
3



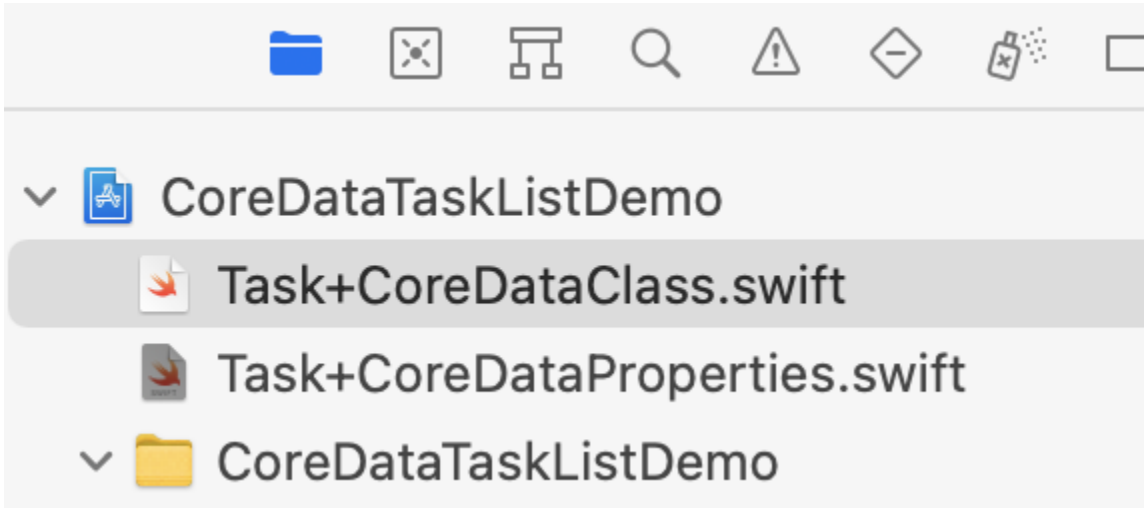
Set CodeGen to Manual/None



Generate the model classes


Editor
Product
Debug
Source Control
Wi
Show Editor Only
Assistant
Layout
Add Entity
Add Fetch Request
Add Fetch Index
Add Configuration
Add Attribute
Add Fetched Property
Add Relationship
Create NSManagedObject Subclass...
Add Model Version...
Import...

Auto generated files:



Screen1 - ViewController.swift

Add a reference to the CoreData context variable:

```
import UIKit
import CoreData     // 1. needed to work with core data

class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
{

    // MARK: Core Data
    let context = (UIApplication.shared.delegate as!
AppDelegate).persistentContainer.viewContext

 …..
}
```

Fetching All Tasks from Core Data

```
// MARK: Lifecycle functions
    override func viewDidLoad() {
        super.viewDidLoad()

        myTableView.delegate = self
        myTableView.dataSource = self


        // Query CoreData for any existing tasks
        // - If tasks exist, then use the list of tasks as your data source
        // for the table view

        // 1. Create a fetch request  (SELECT * FROM Task)
        let request:NSFetchRequest<Task> = Task.fetchRequest()

        // 2. Add any filters to the fetch request (WHERE clause)
        // predicate

        // 3. Initiate the fetch request
        do {
            let results:[Task] = try self.context.fetch(request)
            print("Number of results: \(results.count)")

            // 4. Handle the requests from CoreData
            // TODO: Replace the existing tableview data source with our resuilts from CoreData

            // refresh the table view so it shows our data

        } catch {
            print("Error while fetching data from CoreData")
        }



    }
```

```
Core Data DB Path ::
    /Users/zebra/Library/Dev
    -8EF1-5A374D5145BA/data/
    9885-1B4AEB0F53D1/Librar
Number of results: 0
```

Adding a Task to Core Data

TaskDetailsViewController.swift

```
private func addTask() {
        let t1 = Task(context: self.context)

        t1.name = txtTaskName.text
        t1.isHighPriority = swPriority.isOn

        do {
            try self.context.save()
            print("Task saved!")
            //clear textbox
            txtTaskName.text = ""
        } catch {
            print("Saved failed.")
        }

    }
```

Expected result

```
Button pressed
Trying to add the task to Core Data
Task saved!
```

If you don’t have SQLite browser installed, you should install it!

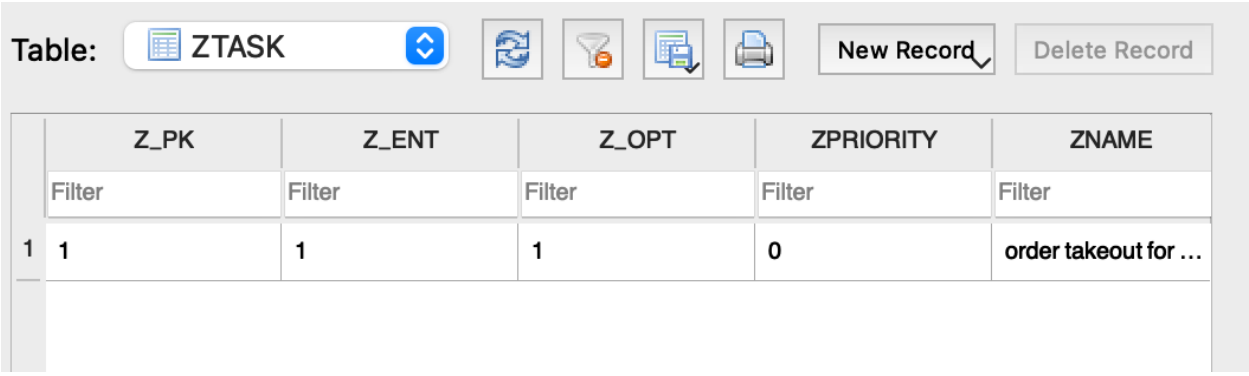


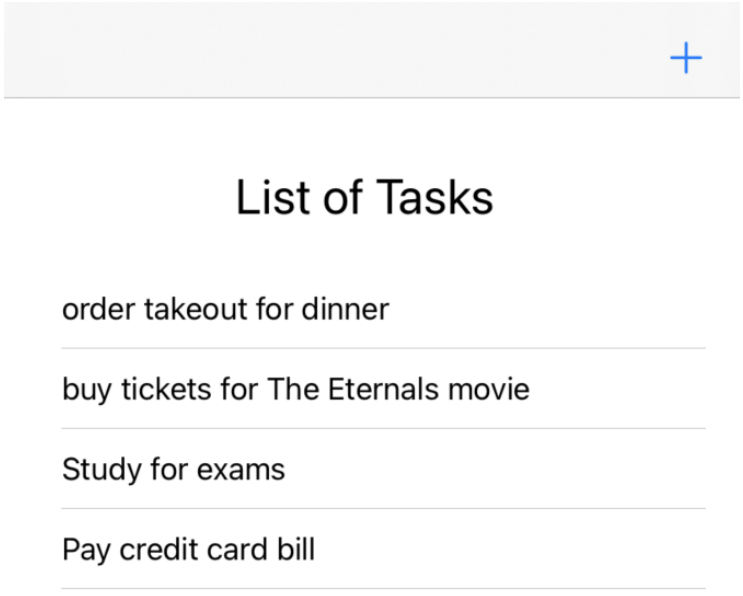
Table:
ZTASK
New Record
Delete Record
Z_PK
Z_ENT
Z_OPT
ZPRIORITY
ZNAME
Filter
Filter
Filter
Filter
Filter
1
1
1
1
0
order takeout for …

Fetch data from Core Data and update table view

```
import UIKit
import CoreData     // 1. needed to work with core data

class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    // MARK: Core Data
    let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext


    // MARK: Outlets
    @IBOutlet weak var myTableView: UITableView!

    // MARK: Table view data source
//    var taskList:[String] = ["Make the bed", "Do homework", "Walk the dog"]

    // Update to an array of Task objects
    var taskList:[Task] = []

    // MARK: Lifecycle functions
    override func viewDidLoad() {
        super.viewDidLoad()

        myTableView.delegate = self
        myTableView.dataSource = self


        // Query CoreData for any existing tasks
        // - If tasks exist, then use the list of tasks as your data source
        // for the table view

        // 1. Create a fetch request  (SELECT * FROM Task)
        let request:NSFetchRequest<Task> = Task.fetchRequest()

        // 2. Add any filters to the fetch request (WHERE clause)
        // predicate

        // 3. Initiate the fetch request
        do {
            let results:[Task] = try self.context.fetch(request)
            print("Number of results: \(results.count)")

            // 4. Handle the requests from CoreData
            // TODO: Replace the existing tableview data source with our resuilts from CoreData
            self.taskList = results

            // refresh the table view so it shows our data

        } catch {
            print("Error while fetching data from CoreData")
        }

    }

    // MARK: Actions
    @IBAction func addButtonPressed(_ sender: Any) { … }

    // MARK: Table View Functions
    func numberOfSections(in tableView: UITableView) -> Int {  … }
```

```
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int { … }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = self.myTableView.dequeueReusableCell(withIdentifier: "myCell", for: indexPath) as UITableViewCell
        // cell.textLabel?.text = taskList[indexPath.row]
        cell.textLabel?.text = taskList[indexPath.row].name
        return cell
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        // navigate to next page
        guard let nextScreen = storyboard?.instantiateViewController(identifier: "taskDetailsScreen") as?
TaskDetailsViewController else {
            print("Cannot find next screen")
            return
        }

        nextScreen.isAddingNewTask = false

        let selectedTask:String = taskList[indexPath.row].name!
        // let selectedTask:Task = taskList[indexPath.row]
        nextScreen.currTask = selectedTask

        self.navigationController?.pushViewController(nextScreen, animated:true)
    }

    // delete a row
    func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath:
IndexPath) {      …  }

}
```

Expected result:



Make tableview refresh when you go back to it

1/ Remove CoreData fetching code from viewDidLoad()
2/ Move it to viewWillAppear()
3/ Add tableView.reloadData() to force the tableview to redraw the rows in the table view

```
override func viewDidLoad() {
        super.viewDidLoad()

        myTableView.delegate = self
        myTableView.dataSource = self

        print("1. Calling viewDidLoad()")

    }

    override func viewWillAppear(_ animated: Bool) {
        print("2. Calling viewWillAppear()")

        // Query CoreData for any existing tasks
        // - If tasks exist, then use the list of tasks as your data source
        // for the table view

        // 1. Create a fetch request  (SELECT * FROM Task)
        let request:NSFetchRequest<Task> = Task.fetchRequest()

        // 2. Add any filters to the fetch request (WHERE clause)
        // predicate

        // 3. Initiate the fetch request
        do {
            let results:[Task] = try self.context.fetch(request)
            print("Number of results: \(results.count)")

            // 4. Handle the requests from CoreData
            // TODO: Replace the existing tableview data source with our resuilts from CoreData
            self.taskList = results

            // refresh the table view so it shows our data
            self.myTableView.reloadData()

        } catch {
            print("Error while fetching data from CoreData")
        }


    }
```

Deleting an Existing Task

- Get a reference to the NSManaged Task object you want to delete
- Use context variable to delete it from CoreData
- Commit your changes
- If commit was successful, delete from the tableview data source & from tableview UI

```
func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle,
forRowAt indexPath: IndexPath) {

    if editingStyle == .delete {

        // 1. get a reference to the Task object you want to delete
        let currTask = self.taskList[indexPath.row]       // of type Task

        // 2. Call the coreData function to delete an item (to be of type NSManagedObject)
        self.context.delete(currTask)          // DELETE FROM Task where ____

        // 3. Save the changes (commit the delete)
        do {
            try self.context.save()
            print("Successfully deleted from CoreData")

            // 4. Remove the object from the tableView data source array
            self.taskList.remove(at:indexPath.row)
            // 5. Remove the object from the tableview UI
            myTableView.deleteRows(at: [indexPath], with: .fade)

        }catch {
            print("Error while deleting data")
        }



    }
}
```

Updating an Existing Task

ViewController.swift

```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {

        // navigate to next page
        guard let nextScreen = storyboard?.instantiateViewController(identifier:
"taskDetailsScreen") as? TaskDetailsViewController else {
            print("Cannot find next screen")
            return
        }

        nextScreen.isAddingNewTask = false

       // let selectedTask:String = taskList[indexPath.row].name!
        let selectedTask:Task = taskList[indexPath.row]
        nextScreen.currTask = selectedTask

        self.navigationController?.pushViewController(nextScreen, animated:true)
    }
```

TaskDetailsViewController.swift

```
class TaskDetailsViewController: UIViewController {

    // MARK: Core Data context variable
    let context = (UIApplication.shared.delegate as!
AppDelegate).persistentContainer.viewContext

    // MARK: Data from Screen 1
    //  var currTask:String?
    var currTask:Task?
    var isAddingNewTask:Bool = false

    // MARK: Outlets
    @IBOutlet weak var txtTaskName: UITextField!
    @IBOutlet weak var swPriority: UISwitch!
    @IBOutlet weak var btnSave: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        // check if the Task is not nil?
        if let currTask = currTask {
            print("Current task is NOT nil")
            self.txtTaskName.text = currTask.name
            swPriority.isOn = currTask.priority
        }

        if (isAddingNewTask == true) {
            btnSave.setTitle("Add New Task", for:.normal)
        }
        else {
            btnSave.setTitle("Update Task", for:.normal)
        }
    }
```

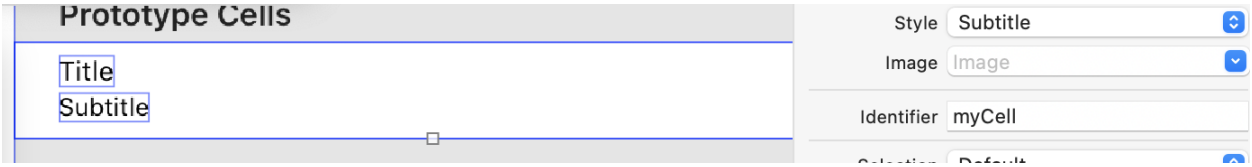
If the task is of type NSManagedObject, and you already have a reference to an task that was retrieved from CoreData, you can reuse this task to do updates

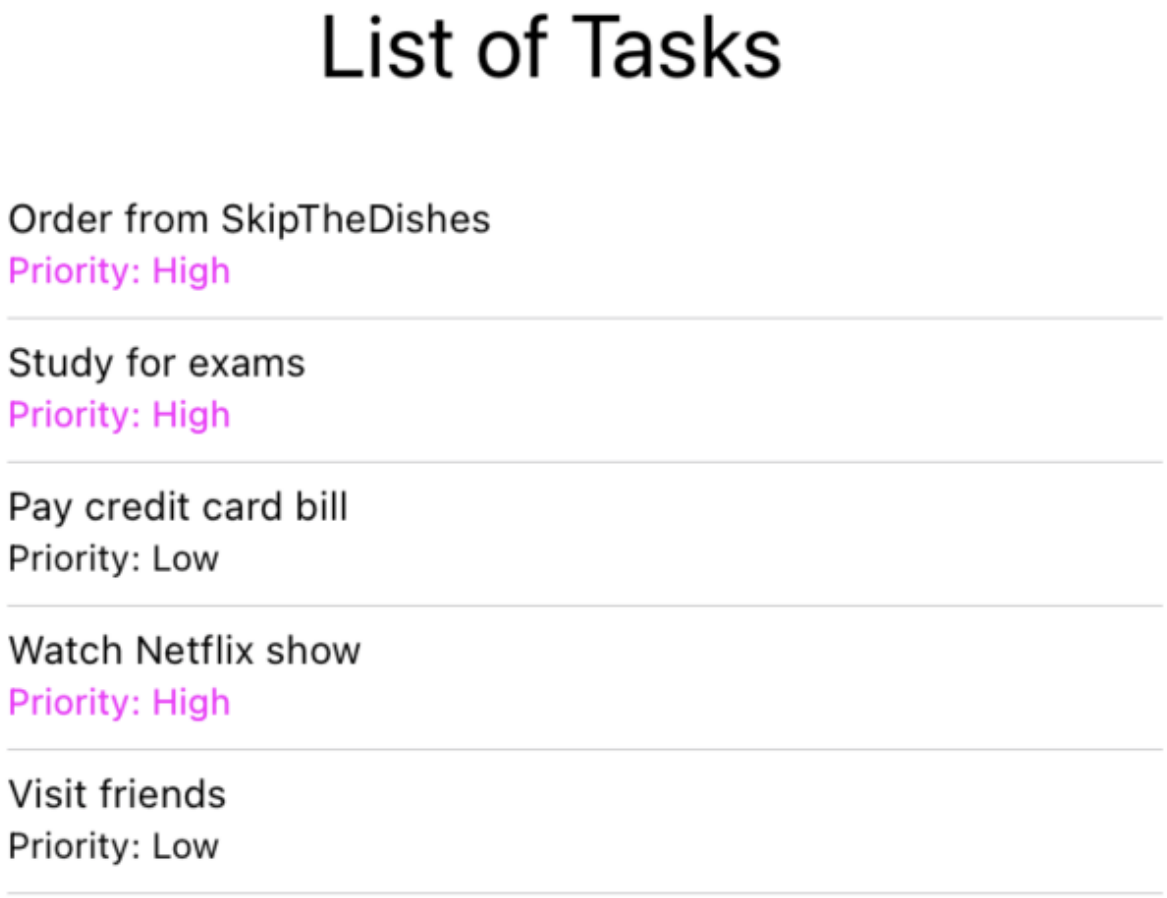
You don’t have to “requery” the database for the task.

```
private func updateTask() {
        //TODO
        print("Trying to update existing task")

        // 1. Get a reference to the NSManagedObject Task that you want to update
        guard let taskToUpdate = self.currTask else {
            print("Current task is nil, cannot proceed")
            return
        }

        // if you are here, then task is not nil

        // 2. Update the properties of the task
        taskToUpdate.name = txtTaskName.text!
        taskToUpdate.priority = swPriority.isOn

        // 3. Commit your changes using self.context.save()
        do {
            try self.context.save()
            print("Update success!")
        }
        catch {
            print("Error while updating!")
        }
        // 4. Done!
    }
```

Update row ui

1/ Change the design of the cell to Subtitle





```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let cell = self.myTableView.dequeueReusableCell(withIdentifier: "myCell", for:
indexPath) as UITableViewCell
        // cell.textLabel?.text = taskList[indexPath.row]
        cell.textLabel?.text = taskList[indexPath.row].name
        if (taskList[indexPath.row].priority == true) {
            cell.detailTextLabel?.text = "Priority: High"
            cell.detailTextLabel?.textColor = UIColor.magenta
        }
        else {
            cell.detailTextLabel?.text = "Priority: Low"
            cell.detailTextLabel?.textColor = UIColor.black
        }

        return cell
    }
```

Reloading the Table From CoreData

ViewController.swift

```
override func viewWillAppear(_ animated: Bool) {

    let request:NSFetchRequest<Task> = Task.fetchRequest()

    // 2. Initiate the fetch request
    do {
      let results:[Task] = try self.context.fetch(request)

      print("Fetch succeeded")
      print("Number of results: \(results.count)")

       // replace the existing task list with results from CoreData
       self.taskList = results

       // refresh tableview
       myTableView.reloadData()

    } catch {
      print("Error while fetching")
    }

  }
```

Deleting Existing Task

ViewController.swift

```
func tableView(_ tableView: UITableView, commit editingStyle:
UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {

        if editingStyle == .delete {

            // 1. get the current task
            let currTask = self.taskList[indexPath.row]

            // 2. remove it from CoreData
            self.context.delete(currTask)

            // 3. commit changes
            do {
                try self.context.save()
                print("Delete success")

                // if save success, them remove from tableview
                self.taskList.remove(at:indexPath.row)
                myTableView.deleteRows(at: [indexPath], with: .fade)

            }
            catch {
                print("Error deleting task")
            }
            // 3. remove it from CoreData



        }
    }
```