**11、添加新的城市信息**

**（1）AreaTableViewController**

@IBAction func close(segue: UIStoryboard) {

}

**（2）AddAreaController**

import UIKit

class AddAreaController: UITableViewController,UIImagePickerControllerDelegate,UINavigationControllerDelegate {

@IBOutlet weak var coverImageView: UIImageView!

func imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any]) {

//给出选中的图片类型

coverImageView.image = info[UIImagePickerControllerOriginalImage] as? UIImage

//设定图片的排列模式

coverImageView.contentMode = .scaleAspectFill

//裁边

coverImageView.clipsToBounds = true

//设置与父视图等宽的约束

//constant是关于距离的约束，所以设为0

let coverWithCons = NSLayoutConstraint(item: coverImageView, attribute: .width, relatedBy: .equal, toItem: coverImageView.superview, attribute: .width, multiplier: 1, constant: 0)

//设置与父视图等高的约束

let coverHeightCons = NSLayoutConstraint(item: coverImageView, attribute: .height, relatedBy: .equal, toItem: coverImageView.superview, attribute: .height, multiplier: 1, constant: 0)

//让两个约束生效

coverWithCons.isActive = true

coverHeightCons.isActive = true

//视图退场

dismiss(animated: true, completion: nil)

}

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

if indexPath.row == 0{

//通过guard判定访问相册的权限是否可用，如果不可用，返回

guard UIImagePickerController.isSourceTypeAvailable(.photoLibrary) else {

print("相册不可用")

return

}

//实例化UIImagePickerController

let picker = UIImagePickerController()

//不允许编辑

picker.allowsEditing = false

//来源为相册

picker.sourceType = .photoLibrary

//设置代理为当前控制器

picker.delegate = self

//以模态弹出

self.present(picker,animated: true,completion: nil)

}

//单元格取消选择

tableView.deselectRow(at: indexPath, animated: true)

}

}

**12 CoreData**

**（1）AppDelegate**

import UIKit

import CoreData

// MARK: - Core Data stack

lazy var persistentContainer: NSPersistentContainer = {

//创建一个持久化容器，名称与工程名一致

let container = NSPersistentContainer(name: "Area")

container.loadPersistentStores(completionHandler: { (storeDescription, error) in

if let error = error as NSError? {

fatalError("Unresolved error \(error), \(error.userInfo)")

}

})

return container

}()

// MARK: - Core Data Saving support

func saveContext () {

let context = persistentContainer.viewContext

if context.hasChanges {

do {

try context.save()

} catch {

let nserror = error as NSError

fatalError("Unresolved error \(nserror), \(nserror.userInfo)")

}

}

}

}

**（2）AddAreaController**

class AddAreaController: UITableViewController,UIImagePickerControllerDelegate,UINavigationControllerDelegate {

var area: AreaMO!

var isVisited = false

@IBOutlet weak var tfName: UITextField!

@IBOutlet weak var tfProvince: UITextField!

@IBOutlet weak var tfPart: UITextField!

@IBOutlet weak var labelVisited: UILabel!

@IBOutlet weak var coverImageView: UIImageView!

@IBAction func saveTap(\_ sender: Any) {

//获取appDelegate，强制转换成工程的AppDelegate，可获得对它的引用

let appDelegate = UIApplication.shared.delegate as! AppDelegate

//可以获得CoreData持久化容器中的Context

area = AreaMO(context: appDelegate.persistentContainer.viewContext)

//设置相关的值

area.name = tfName.text

area.province = tfProvince.text

area.part = tfPart.text

area.isVisited = isVisited

//把图像转换为JPEG格式，0.7是图片的清晰度

if let imageData = UIImageJPEGRepresentation(coverImageView.image!, 0.7) {

area.image = NSData(data: imageData)

}

print("正在保存...")

//保存数据

appDelegate.saveContext()

//退出，返回到首页

performSegue(withIdentifier: "unwindToHomeList", sender: self)

}

@IBAction func isVisited(\_ sender: UIButton) {

if sender.tag == 8001 {

isVisited = true

labelVisited.text = "我来过"

} else {

isVisited = false

labelVisited.text = "没去过"

}

}

override func viewDidLoad() {

super.viewDidLoad()

}

}

**（3）AreaTableViewController**

import UIKit

import CoreData

class AreaTableViewController: UITableViewController,NSFetchedResultsControllerDelegate {

var areas: [AreaMO] = [ ]

var fc: NSFetchedResultsController<AreaMO>!

override func viewDidLoad() {

super.viewDidLoad()

fetchAllData2()

}

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

//调用获取数据的方法

//fetchAllData()

//整体刷新数据

//tableView.reloadData()

}

//当控制器开始处理内容变化时

func controllerWillChangeContent(\_ controller: NSFetchedResultsController<NSFetchRequestResult>) {

tableView.beginUpdates()

}

//当控制器已经处理完内容变更时

func controllerDidChangeContent(\_ controller: NSFetchedResultsController<NSFetchRequestResult>) {

tableView.endUpdates()

}

//内容发生变更时

func controller(\_ controller: NSFetchedResultsController<NSFetchRequestResult>, didChange anObject: Any, at indexPath: IndexPath?, for type: NSFetchedResultsChangeType, newIndexPath: IndexPath?) {

//变更类型筛选

switch type {

//删除数据

case .delete:

tableView.deleteRows(at: [indexPath!], with: .automatic)

//插入数据

case .insert:

tableView.insertRows(at: [newIndexPath!], with: .automatic)

//更新数据

case .update:

tableView.reloadRows(at: [indexPath!], with: .automatic)

default:

tableView.reloadData()

}

//数据已经发生变化，同步到数组

if let object = controller.fetchedObjects{

areas = object as! [AreaMO]

}

}

func fetchAllData2() {

//请求结果类型是AreaMO

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

//NSSortDescriptor指定请求结果如何排序

let sd = NSSortDescriptor(key: "name", ascending: true)

request.sortDescriptors = [sd]

let appDelegate = UIApplication.shared.delegate as! AppDelegate

let context = appDelegate.persistentContainer.viewContext

//NSFetchedResultsController初始化后

fc = NSFetchedResultsController(fetchRequest: request, managedObjectContext: context, sectionNameKeyPath: nil, cacheName: nil)

//指定代理

fc.delegate = self

do {

try fc.performFetch()

//执行查询，指定代理

if let object = fc.fetchedObjects{

areas = object

}

} catch {

print(error)

}

}

//获取数据

/\*func fetchAllData(){

let appDelegate = UIApplication.shared.delegate as! AppDelegate

do {

//获取AreaMO此Enitity的所有条目

areas = try appDelegate.persistentContainer.viewContext.fetch(AreaMO.fetchRequest())

} catch {

print(error)

}

}\*/

// MARK: - Table view dalegate

override func tableView(\_ tableView: UITableView, editActionsForRowAt indexPath: IndexPath) -> [UITableViewRowAction]? {

let actionDel = UITableViewRowAction(style: .destructive, title: "删除") { (\_, indexPath) in

//从数据源中删除相应的行

//self.areas.remove(at: indexPath.row)

let appDelegate = UIApplication.shared.delegate as! AppDelegate

let context = appDelegate.persistentContainer.viewContext

//删除当前获取到的指定行的数据

context.delete(self.fc.object(at: indexPath))

//保存

appDelegate.saveContext()

//删除某一行刷新

//tableView.deleteRows(at: [indexPath], with: .fade)

//整体刷新

//tableView.reloadData()

}

}

**（4）DetailTableViewController**

import UIKit

class DetailTableViewController: UITableViewController {

@IBAction func close(segue: UIStoryboardSegue){

//反场前取回源控制器

let reviewVC = segue.source as! ReviewViewController

//得到源控制中rating中的值，并判断其中是否有值

if let rating = reviewVC.rating {

//更新模型中的rating值

self.area.rating = rating

//将当前ratingBtn中的图片设置为rating

self.ratingBtn.setImage(UIImage(named: rating), for: .normal)

}

let appDelegate = UIApplication.shared.delegate as! AppDelegate

appDelegate.saveContext()

}

override func viewDidLoad() {

super.viewDidLoad()

//预计行高

tableView.estimatedRowHeight = 40

//实际行高为自动适应

tableView.rowHeight = UITableViewAutomaticDimension

if let rating = area.rating{

self.ratingBtn.setImage(UIImage(named: rating), for: .normal)

}

}