import UIKit

import Foundation

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

var window: UIWindow?

class var sharedInstance : AppDelegate{

guard let single = UIApplication.shared.delegate as? AppDelegate else{

return AppDelegate()

}

return single

}

fileprivate var baseTabBar : LYTabBarController = LYTabBarController()

fileprivate lazy var loginNav : LYNavigationController = {

let loginVC = LoginViewController.spwan()

let loginNav = LYNavigationController(rootViewController: loginVC)

return loginNav

}()

fileprivate lazy var sysMaintainNav : LYNavigationController = {

let sysMaintainVC = SystemMaintainingViewController.spwan()

let sysMaintainNav = LYNavigationController(rootViewController: sysMaintainVC)

return sysMaintainNav

}()

lazy var bgView : UIView = {

let bgView = UIView(frame: UIScreen.main.bounds)

bgView.backgroundColor = UIColor.black

return bgView

}()

fileprivate lazy var scrollView: GuardScrollView = {

let scrollView = GuardScrollView(frame: UIScreen.main.bounds)

scrollView.backgroundColor = UIColor.white

return scrollView

}()

fileprivate var adViewNav : LYNavigationController = {

let adVC = AdViewController()

let adViewNav = LYNavigationController(rootViewController: adVC)

return adViewNav

}()

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

self.window = UIWindow(frame: UIScreen.main.bounds)

self.window?.backgroundColor = UIColor.white

self.window?.makeKeyAndVisible()

// self.window?.rootViewController = loginNav

LocalData.saveYesOrNotValue(value: "0", key: KIsLaunchInfoFromAppWebKey)

//检测网络

if !NetTools.checkNetType(){

LYProgressHUD.showError("未连接网络，请检查网络设置")

}

window?.rootViewController = adViewNav

self.confitUShareSettings()

AMapServices.shared().apiKey = KAmapKey

WXApi.registerApp(KWechatKey)

self.configEasemob(application, launchOptions: launchOptions)

self.setupJpush(launchOptions)

NotificationCenter.default.addObserver(self, selector: #selector(AppDelegate.enterHomeView(noti:)), name: NSNotification.Name(rawValue: KEnterAppNotification), object: nil)

NotificationCenter.default.addObserver(self, selector: #selector(AppDelegate.loginSuccessAction(noti:)), name: NSNotification.Name(rawValue: KLoginSuccessNotification), object: nil)

return true

}

func setUpRootViewController() {

if LocalData.getYesOrNotValue(key: IsSystemMaintaining){

window?.rootViewController = sysMaintainNav

}else{

let isNotFirstOpen = LocalData.getYesOrNotValue(key: IsNotFirstOpen)

if isNotFirstOpen{

let isLogin = LocalData.getYesOrNotValue(key: IsLogin)

if isLogin {

for VC in baseTabBar.childViewControllers {

let navVC = VC as! LYNavigationController

navVC.viewControllers.last?.navigationController?.popToRootViewController(animated: false)

}

window?.rootViewController = baseTabBar

self.goToLaunchDetailVC()

self.checkAppUpdate()

}else{

window?.rootViewController = loginNav

}

}else{

self.showAppGuardView()

}

}

}

func goToLaunchDetailVC(){

let launchInfo = LocalData.getLaunchTypeAndId()

if launchInfo != nil{

let nav = baseTabBar.selectedViewController! as! LYNavigationController

if launchInfo!["type"] == LaunchDetailType.GoodsDetailType.rawValue{

let detailVC = GoodsDetailViewController.spwan()

detailVC.goodsId = launchInfo!["idStr"]!

nav.viewControllers.first!.navigationController?.pushViewController(detailVC, animated: true)

}else if launchInfo!["type"] == LaunchDetailType.ProjectDetailType.rawValue{

let detailVC = TaskReceiveDetailViewController.spwan()

detailVC.task\_id = launchInfo!["idStr"]!

nav.viewControllers.first!.navigationController?.pushViewController(detailVC, animated: true)

}

}

LocalData.setLaunchTypeAndId(type: nil, idStr: nil)

}

func goToLaunchDetailVC2(){

if window?.rootViewController == adViewNav{

LocalData.saveYesOrNotValue(value: "0", key: KIsLaunchInfoFromAppWebKey)

let launchInfo = LocalData.getLaunchTypeAndId()

if launchInfo != nil{

if launchInfo!["type"] == LaunchDetailType.GoodsDetailType.rawValue{

let detailVC = GoodsDetailViewController.spwan()

detailVC.goodsId = launchInfo!["idStr"]!

adViewNav.viewControllers.first!.navigationController?.pushViewController(detailVC, animated: true)

}else if launchInfo!["type"] == LaunchDetailType.ProjectDetailType.rawValue{

let detailVC = TaskReceiveDetailViewController.spwan()

detailVC.task\_id = launchInfo!["idStr"]!

adViewNav.viewControllers.first!.navigationController?.pushViewController(detailVC, animated: true)

}

}

LocalData.setLaunchTypeAndId(type: nil, idStr: nil)

}

}

func isRootViewAdNav() -> Bool {

return window?.rootViewController == adViewNav

}

func confitUShareSettings() {

UMSocialManager.default().openLog(true)

UMSocialManager.default().setPlaform(.wechatSession, appKey: KWechatKey, appSecret: KWechatSecretKey, redirectURL: nil)

UMSocialManager.default().setPlaform(.wechatTimeLine, appKey: KWechatKey, appSecret: KWechatSecretKey, redirectURL: nil)

UMSocialManager.default().setPlaform(.QQ, appKey: KTencentAppId, appSecret: KTencentAppKey, redirectURL: nil)

UMSocialManager.default().setPlaform(.qzone, appKey: KTencentAppId, appSecret: KTencentAppKey, redirectURL: nil)

UMSocialManager.default().setPlaform(.sina, appKey: KSinaAppKey, appSecret: KSinaAppSecret, redirectURL: "http://www.weibo.com")

}

func configEasemob(\_ application: UIApplication, launchOptions: [UIApplicationLaunchOptionsKey: Any]?){

let options = HOptions()

options.apnsCertName = KEasemobCertName

options.tenantId = KEasemobId

options.appkey = KEasemobKey

HChatClient.shared().initializeSDK(with: options)

HChatClient.shared().add(self, delegateQueue: nil)

self.loginEasemob()

}

func setupJpush(\_ launchOptions: [UIApplicationLaunchOptionsKey: Any]?) {

let entity = JPUSHRegisterEntity()

entity.types = Int(JPAuthorizationOptions.alert.rawValue)|Int(JPAuthorizationOptions.badge.rawValue)|Int(JPAuthorizationOptions.sound.rawValue)

JPUSHService.register(forRemoteNotificationConfig: entity, delegate: self)

let advertisingId = ASIdentifierManager.shared().advertisingIdentifier.uuidString

if DeBug{

JPUSHService.setup(withOption: launchOptions, appKey: KJpushKey, channel: "itms-apps://itunes.apple.com/cn/app/id1171281585?mt=8", apsForProduction: false, advertisingIdentifier : advertisingId)

}else{

JPUSHService.setup(withOption: launchOptions, appKey: KJpushKey, channel: "itms-apps://itunes.apple.com/cn/app/id1171281585?mt=8", apsForProduction: true, advertisingIdentifier : advertisingId)

}

JPUSHService.registrationIDCompletionHandler { (resCode, registrationID) in

if resCode == 0{

print("注册极光推送成功---" + registrationID!)

}else{

print("注册极光推送失败---" + String.init(format: "%d", resCode))

}

}

if !LocalData.getUserPhone().isEmpty{

JPUSHService.setAlias(LocalData.getUserPhone(), completion: { (isResCode, alias, seq) in

}, seq:0)

}

}

func application(\_ application: UIApplication, open url: URL, sourceApplication: String?, annotation: Any) -> Bool {

let urlStr = url.absoluteString

//网页打开时跳转详情页

if urlStr.hasPrefix("qixiaofu://"){

if url.host == "goodsDetail"{

//商品详情

if url.query != nil{

let idStr = url.query!.components(separatedBy: "=").last

if idStr != nil{

LocalData.setLaunchTypeAndId(type: LaunchDetailType.GoodsDetailType, idStr: idStr)

}

}

}else if url.host == "projectDetail"{

if url.query != nil{

let idStr = url.query!.components(separatedBy: "=").last

if idStr != nil{

LocalData.setLaunchTypeAndId(type: LaunchDetailType.ProjectDetailType, idStr: idStr)

}

}

}

if window?.rootViewController == baseTabBar{

self.goToLaunchDetailVC()

}else if window?.rootViewController == adViewNav && LocalData.getYesOrNotValue(key: KIsLaunchInfoFromAppWebKey){

self.goToLaunchDetailVC2()

}

}

if urlStr.hasPrefix(KWechatKey){

return WXApi.handleOpen(url, delegate: self)

}

if url.host == "safepay" {

AlipaySDK.defaultService().processOrder(withPaymentResult: url, standbyCallback: { (resultDict) in

NotificationCenter.default.post(name: NSNotification.Name(rawValue: KAliPayNotiName), object: nil, userInfo: resultDict)

})

}

if url.host == "platformapi" {

AlipaySDK.defaultService().processAuthResult(url, standbyCallback: { (resultDict) in

//处理支付结果

NotificationCenter.default.post(name: NSNotification.Name(rawValue: KAliPayNotiName), object: nil, userInfo: resultDict)

})

}

UMSocialManager.default().handleOpen(url, sourceApplication: sourceApplication, annotation: annotation)

return true

}

func application(\_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey : Any] = [:]) -> Bool {

let urlStr = url.absoluteString

if urlStr.hasPrefix("qixiaofu://"){

if url.host == "goodsDetail"{

//商品详情

if url.query != nil{

let idStr = url.query!.components(separatedBy: "=").last

if idStr != nil{

LocalData.setLaunchTypeAndId(type: LaunchDetailType.GoodsDetailType, idStr: idStr)

}

}

}else if url.host == "projectDetail"{

//任务单详情

if url.query != nil{

let idStr = url.query!.components(separatedBy: "=").last

if idStr != nil{

LocalData.setLaunchTypeAndId(type: LaunchDetailType.ProjectDetailType, idStr: idStr)

}

}

}

if window?.rootViewController == baseTabBar{

self.goToLaunchDetailVC()

}else if window?.rootViewController == adViewNav && LocalData.getYesOrNotValue(key: KIsLaunchInfoFromAppWebKey){

self.goToLaunchDetailVC2()

}

}

if urlStr.hasPrefix(KWechatKey){

return WXApi.handleOpen(url, delegate: self)

}

if url.host == "safepay" {

AlipaySDK.defaultService().processOrder(withPaymentResult: url, standbyCallback: { (resultDict) in

NotificationCenter.default.post(name: NSNotification.Name(rawValue: KAliPayNotiName), object: nil, userInfo: resultDict)

})

}

if url.host == "platformapi" {

AlipaySDK.defaultService().processAuthResult(url, standbyCallback: { (resultDict) in

//处理支付结果

NotificationCenter.default.post(name: NSNotification.Name(rawValue: KAliPayNotiName), object: nil, userInfo: resultDict)

})

}

UMSocialManager.default().handleOpen(url, options: options)

return true

}

func applicationDidEnterBackground(\_ application: UIApplication) {

HChatClient.shared().applicationDidEnterBackground(application)

}

func applicationWillEnterForeground(\_ application: UIApplication) {

self.checkAppUpdate()

HChatClient.shared().applicationWillEnterForeground(application)

self.loginEasemob()

application.applicationIconBadgeNumber = 0

application.cancelAllLocalNotifications()

}

func application(\_ application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {

JPUSHService.registerDeviceToken(deviceToken)

}

func application(\_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable : Any]) {

JPUSHService.handleRemoteNotification(userInfo)

}

func application(\_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable : Any], fetchCompletionHandler completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {

JPUSHService.handleRemoteNotification(userInfo)

completionHandler(UIBackgroundFetchResult.newData)

}

}

extension AppDelegate{

fileprivate func showAppGuardView() {

self.window?.addSubview(bgView)

bgView.addSubview(scrollView)

}

@objc func enterHomeView(noti: Notification) {

LocalData.saveYesOrNotValue(value: "1", key: IsNotFirstOpen)

let isLogin = LocalData.getYesOrNotValue(key: IsLogin)

if isLogin {

DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + Double((Int64)(UInt64(2.5) \* NSEC\_PER\_SEC)) / Double(NSEC\_PER\_SEC)) {

UIView.animate(withDuration: 1.0, animations: {

self.bgView.layer.transform = CATransform3DMakeScale(2, 2, 2)

self.bgView.alpha = 0

}, completion: { (completion) -> Void in

UIApplication.shared.isStatusBarHidden = false

self.bgView.removeFromSuperview()

})

}

}else{

self.bgView.removeFromSuperview()

self.setUpRootViewController()

}

}

}

extension AppDelegate{

func checkAppUpdate() {

NetTools.requestData(type: .post, urlString: iOSVersionApi, succeed: { (result, error) in

let localVersion = appVersion().replacingOccurrences(of: ".", with: "").intValue

let netVersion = result["new\_version\_num"].stringValue.intValue

let isForce = result["version\_force"].stringValue.intValue

var message = result["version\_msg"].stringValue

var url = result["version\_url"].stringValue

if localVersion < netVersion{

if isForce == 1{

LYAlertView.show("提示", message,"去更新",{

let urlStr = url

if UIApplication.shared.canOpenURL(URL(string:urlStr)!){

UIApplication.shared.openURL(URL(string:urlStr)!)

}

})

}else{

LYAlertView.show("提示", message,"下次再说","去更新",{

let urlStr = url

if UIApplication.shared.canOpenURL(URL(string:urlStr)!){

UIApplication.shared.openURL(URL(string:urlStr)!)

}

})

}

}

}) { (error) in

}

}

@objc func loginSuccessAction(noti: Notification) {

UIView.animate(withDuration: 1.0, animations: {

self.setUpRootViewController()

}) { (completion) in

UIApplication.shared.isStatusBarHidden = false

}

}

}

extension AppDelegate : WXApiDelegate{

func onResp(\_ resp: BaseResp!) {

if resp.isKind(of: PayResp.self){

var dict = [String:String]()

dict["code"] = "\(resp.errCode)"

dict["error"] = resp.errStr

NotificationCenter.default.post(name: NSNotification.Name(rawValue: KWechatPayNotiName), object: nil, userInfo: dict)

}

}

}

extension AppDelegate : HChatClientDelegate,JPUSHRegisterDelegate{

func loginEasemob() {

if !LocalData.getUserPhone().isEmpty && LocalData.getYesOrNotValue(key: IsLogin){

let loginError = HChatClient.shared().login(withUsername: LocalData.getUserPhone(), password: "11")

}

}

func userAccountDidLoginFromOtherDevice() {

LYAlertView.show("提示", "此账号已在其他设备登陆，如非本人操作，请修改密码","知道了")

LocalData.saveUserId(userId: "")

LocalData.saveYesOrNotValue(value: "0", key: IsLogin)

self.setUpRootViewController()

}

@available(iOS 10.0, \*)

func jpushNotificationCenter(\_ center: UNUserNotificationCenter!, willPresent notification: UNNotification!, withCompletionHandler completionHandler: ((Int) -> Void)!) {

let userInfo = notification.request.content.userInfo

let content = notification.request.content

let badge = content.badge

UIApplication.shared.applicationIconBadgeNumber = badge?.intValue ?? 0

if notification.request.trigger != nil{

if ((notification.request.trigger! as? UNPushNotificationTrigger) != nil){

JPUSHService.handleRemoteNotification(userInfo)

}

}

completionHandler(Int(UNNotificationPresentationOptions.alert.rawValue) | Int(UNNotificationPresentationOptions.badge.rawValue))

}

@available(iOS 10.0, \*)

func jpushNotificationCenter(\_ center: UNUserNotificationCenter!, didReceive response: UNNotificationResponse!, withCompletionHandler completionHandler: (() -> Void)!) {

print(response)

if response.notification.request.trigger != nil{

if ((response.notification.request.trigger! as? UNPushNotificationTrigger) != nil){

JPUSHService.handleRemoteNotification(response.notification.request.content.userInfo)

}

}

completionHandler()

}

}

import UIKit

import SwiftyJSON

import Kingfisher

class HomeViewController: BaseViewController {

class func spwan() -> HomeViewController{

return self.loadFromStoryBoard(storyBoard: "Home") as! HomeViewController

}

@IBOutlet weak var contentViewH: NSLayoutConstraint!

@IBOutlet weak var headerView: UIView!

@IBOutlet weak var headerViewH: NSLayoutConstraint!

@IBOutlet weak var recommendView: UIView!

@IBOutlet weak var collectionView: UICollectionView!

fileprivate lazy var bannerView1 : LYAnimateBannerView = {

let bannerView = LYAnimateBannerView.init(frame: CGRect(x: 0, y: 0, width: kScreenW, height: kScreenW \* 320 / 750), delegate: self)

bannerView.backgroundColor = UIColor.white

bannerView.showPageControl = true

self.headerView.addSubview(bannerView)

return bannerView

}()

fileprivate lazy var bannerView2 : LYUpScrollBannerView = {

let bannerView = LYUpScrollBannerView.init(frame: CGRect(x: 80, y: 0, width: kScreenW - 80 - 60, height: 49), delegate: self)

bannerView.backgroundColor = UIColor.white

self.recommendView.addSubview(bannerView)

return bannerView

}()

fileprivate var class\_list : JSON = []

fileprivate var bill\_list : JSON = []

fileprivate var member\_list : JSON = []

fileprivate var eng\_banner\_list : JSON = []

fileprivate var banner\_list : JSON = []

fileprivate var rightItem = UIBarButtonItem()

override func viewDidLoad() {

super.viewDidLoad()

self.setUpMainView()

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

self.loadMainData()

}

override func viewWillDisappear(\_ animated: Bool) {

super.viewWillDisappear(animated)

self.bannerView1.timer?.invalidate()

self.bannerView2.timer?.invalidate()

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

}

@IBAction func moreRecommendAction() {

if self.rightItem.title == "接单"{

let taskVC = TaskListViewController()

taskVC.isHomeMore = true

self.navigationController?.pushViewController(taskVC, animated: true)

}else{

let engineerVC = EngineerListViewController()

engineerVC.isHomeMore = true

self.navigationController?.pushViewController(engineerVC, animated: true)

}

}

}

extension HomeViewController{

override func setUpMainView() {

super.setUpMainView()

self.headerViewH.constant = kScreenW \* 320 / 750 + 50

self.setUpCollectionFlowLayout()

self.setUpTitleView()

self.navigationItem.leftBarButtonItem = UIBarButtonItem.init(image: imageLiteral(resourceName: "scan\_icon"), target: self, action: #selector(HomeViewController.leftItemAction))

}

func setUpCollectionFlowLayout() {

self.collectionView.register(UINib.init(nibName: "HomeCollectionViewCell", bundle: Bundle.main), forCellWithReuseIdentifier: "HomeCollectionViewCell")

let layout = UICollectionViewFlowLayout()

layout.itemSize = CGSize(width:80, height:90)

layout.minimumLineSpacing = 10

layout.minimumInteritemSpacing = 10

layout.sectionInset = UIEdgeInsets(top: 10,left: 10,bottom: 10,right: 10)

self.collectionView.setCollectionViewLayout(layout, animated: true)

}

func setUpTitleView() {

let lySwitch = LYSwitch(frame:CGRect(x:0, y:0, width:100, height:30), tag:123, delegate: self)

lySwitch.rightTitle = "发单"

lySwitch.leftTitle = "接单"

self.navigationItem.titleView = lySwitch

rightItem = UIBarButtonItem(title: "接单", style: .plain, target: self, action: #selector(HomeViewController.rightItemAction))

rightItem.tintColor = UIColor.RGBS(s: 33)

rightItem.setTitleTextAttributes([NSAttributedStringKey.font:UIFont.systemFont(ofSize: 15)], for: .normal)

self.navigationItem.rightBarButtonItem = self.rightItem

}

func setUpBannerView() {

var bannerArray3 = Array<String>()

var bannerArray4 = Array<String>()

if self.rightItem.title == "接单"{

for subJson : JSON in self.eng\_banner\_list.arrayValue {

bannerArray3.append(subJson.stringValue)

}

for subJson : JSON in self.bill\_list.arrayValue {

bannerArray4.append(subJson["title"].stringValue)

}

}else{

for subJson : JSON in self.banner\_list.arrayValue {

bannerArray3.append(subJson.stringValue)

}

for subJson : JSON in self.member\_list.arrayValue {

bannerArray4.append(subJson["title"].stringValue)

}

}

self.bannerView1.imageUrlArray = bannerArray3

self.bannerView2.titleArray = bannerArray4

}

func loadMainData() {

NetTools.requestData(type: .post, urlString: HomeMainApi, succeed: { (resultDict, error) in

self.loadDataFinished()

self.class\_list = resultDict["class\_list"]

self.bill\_list = resultDict["bill\_list"]

self.member\_list = resultDict["member\_list"]

self.eng\_banner\_list = resultDict["eng\_banner\_list"]

self.banner\_list = resultDict["banner\_list"]

self.setUpBannerView()

self.collectionView.reloadData()

var collecViewH : CGFloat = 0

if kScreenW == 320{

if self.class\_list.arrayValue.count % 3 == 0{

collecViewH = CGFloat(self.class\_list.arrayValue.count / 3 \* 100)

}else{

collecViewH = CGFloat(self.class\_list.arrayValue.count / 3 \* 100 + 100)

}

}else{

if self.class\_list.arrayValue.count % 4 == 0{

collecViewH = CGFloat(self.class\_list.arrayValue.count / 4 \* 100)

}else{

collecViewH = CGFloat(self.class\_list.arrayValue.count / 4 \* 100 + 100)

}

}

self.contentViewH.constant = self.headerViewH.constant + collecViewH + 20

}) { (error) in

self.loadDataFinished()

LYProgressHUD.showError(error!)

}

}

}

extension HomeViewController : LYSwitchDelegate{

func lySwitchChangeIndex(lySwitch:LYSwitch, index: NSInteger) {

debugPrint(index,lySwitch.tag)

var rightItemTitle = String()

if lySwitch.tag == 123 && index == 1 {

rightItemTitle = "接单"

}else if lySwitch.tag == 123 && index == 2 {

rightItemTitle = "发单"

}

self.rightItem.title = rightItemTitle

self.setUpBannerView()

}

@objc func rightItemAction(){

if self.rightItem.title == "接单"{

let taskVC = TaskListViewController()

taskVC.isHomeAllTaskList = true

self.navigationController?.pushViewController(taskVC, animated: true)

}else{

UserViewModel.haveTrueName(parentVC: self, {

let sendTaskVC = SendTaskViewController.spwan()

self.navigationController?.pushViewController(sendTaskVC, animated: true)

})

}

}

}

extension HomeViewController : LYAnimateBannerViewDelegate, LYUpScrollBannerViewDelegate{

func LY\_UpScrollBannerViewClick(index: NSInteger) {

if self.rightItem.title == "接单"{

if self.bill\_list.arrayValue.count > index{

let detailVC = TaskReceiveDetailViewController.spwan()

detailVC.task\_id = self.bill\_list.arrayValue[index]["id"].stringValue

self.navigationController?.pushViewController(detailVC, animated: true)

}

}else{

if self.member\_list.arrayValue.count > index{

let engineerDetailVC = EngineerDetailViewController()

engineerDetailVC.member\_id = self.member\_list.arrayValue[index]["id"].stringValue

self.navigationController?.pushViewController(engineerDetailVC, animated: true)

}

}

}

func LY\_AnimateBannerViewClick(banner:LYAnimateBannerView, index: NSInteger) {

if self.rightItem.title == "接单"{

if self.eng\_banner\_list.count > index{

let subJson = self.eng\_banner\_list[index]

if subJson["url"].stringValue.trim.isEmpty{

return

}

let webVC = BaseWebViewController.spwan()

webVC.urlStr = subJson["url"].stringValue

webVC.titleStr = subJson["title"].stringValue

self.navigationController?.pushViewController(webVC, animated: true)

}

}else{

if self.banner\_list.count > index{

let subJson = self.banner\_list[index]

if subJson["url"].stringValue.trim.isEmpty{

return

}

let webVC = BaseWebViewController.spwan()

webVC.urlStr = subJson["url"].stringValue

webVC.titleStr = subJson["title"].stringValue

self.navigationController?.pushViewController(webVC, animated: true)

}

}

}

}

extension HomeViewController : UICollectionViewDelegate,UICollectionViewDataSource{

func collectionView(\_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {

return self.class\_list.arrayValue.count

}

func collectionView(\_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

let item : HomeCollectionViewCell = collectionView.dequeueReusableCell(withReuseIdentifier: "HomeCollectionViewCell", for: indexPath) as! HomeCollectionViewCell

let json = self.class\_list.arrayValue[indexPath.row]

item.titleLbl.text = json["gc\_name"].stringValue

item.iconImgV.setImageUrlStr(json["gc\_image"].stringValue)

return item

}

func collectionView(\_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {

collectionView.deselectItem(at: indexPath, animated: false)

debugPrint(indexPath.row)

if self.class\_list.arrayValue.count > indexPath.row{

let json = self.class\_list.arrayValue[indexPath.row]

if self.rightItem.title == "接单"{

let taskVC = TaskListViewController()

taskVC.gc\_id = json["gc\_id"].stringValue

taskVC.gc\_name = json["gc\_name"].stringValue

taskVC.categoryArray = json["list"].arrayValue

self.navigationController?.pushViewController(taskVC, animated: true)

}else{

let engineerVC = EngineerListViewController()

engineerVC.gc\_id = json["gc\_id"].stringValue

engineerVC.gc\_name = json["gc\_name"].stringValue

self.navigationController?.pushViewController(engineerVC, animated: true)

}

}

}

}

extension HomeViewController{

@objc func leftItemAction() {

let scanVC = ScanActionViewController()

scanVC.scanResultBlock = {(result) in

let url = URL(string:result)

if url != nil{

if UIApplication.shared.canOpenURL(url!){

UIApplication.shared.openURL(url!)

}

}

}

self.navigationController?.pushViewController(scanVC, animated: true)

}

}

import UIKit

import SwiftyJSON

class SendTaskViewController: BaseTableViewController {

class func spwan() -> SendTaskViewController{

return self.loadFromStoryBoard(storyBoard: "Home") as! SendTaskViewController

}

var isRepairOrder = false//是否为补单

var isRedoOrder = false//是否为重新发布

var orderId = ""

@IBOutlet weak var taskNameTF: UITextField!

@IBOutlet weak var serverControlLbl: UILabel!

@IBOutlet weak var serverTypeLbl: UILabel!

@IBOutlet weak var serverAreaLbl: UILabel!

@IBOutlet weak var serverStartTimeLbl: UILabel!

@IBOutlet weak var serverEndTimeLbl: UILabel!

@IBOutlet weak var serverRangeLbl: UILabel!

@IBOutlet weak var goodsBrandLbl: UILabel!

@IBOutlet weak var countTF: UITextField!

@IBOutlet weak var unitTF: UITextField!

@IBOutlet weak var priceTF: UITextField!

@IBOutlet weak var goodsBrandTF: UITextField!

fileprivate var sTime : Date?

fileprivate var eTime : Date?

fileprivate var lat : String = ""

fileprivate var lon : String = ""

fileprivate var params : [String : Any] = [:]//参数

fileprivate var controlJson : JSON = []

fileprivate var rangeJson : JSON = []

fileprivate var paymentJson : JSON = []

fileprivate var serverTypeJson : JSON = []

fileprivate var top\_price : String = "100"

fileprivate var redoOrderDataJson : JSON = []//重新发单时的订单信息

fileprivate var service\_form : String = ""//服务形式

fileprivate var service\_type : String = ""//服务类型

fileprivate var service\_sector : String = ""//服务领域

fileprivate var selectedServerRangeIndex : Array<String>?

override func viewDidLoad() {

super.viewDidLoad()

self.loadEssentialData()

if self.isRedoOrder{

self.loadRedoOrderData()

self.navigationItem.title = "发单"

}else if self.isRepairOrder{

self.navigationItem.title = "补单"

}else{

self.navigationItem.title = "发单"

}

self.navigationItem.rightBarButtonItem = UIBarButtonItem(title:"下一步" ,target:self,action:#selector(SendTaskViewController.rightItemAction))

self.navigationItem.leftBarButtonItem = UIBarButtonItem.init(image: UIImage(named:"btn\_back"), target: self, action: #selector(SendTaskViewController.backClick))

}

func loadRedoOrderData() {

let redoParams : [String : Any] = ["id" : self.orderId]

LYProgressHUD.showLoading()

NetTools.requestData(type: .post, urlString: RedoOrderDataApi, parameters: redoParams, succeed: { (result, msg) in

self.taskNameTF.text = result["entry\_name"].stringValue

self.taskNameTF.isEnabled = false

self.serverControlLbl.text = result["service\_form"].stringValue

self.serverTypeLbl.text = result["service\_type"].stringValue

self.serverAreaLbl.text = result["service\_address"].stringValue

self.serverRangeLbl.text = result["service\_sector"].stringValue

self.goodsBrandTF.text = result["service\_brand"].stringValue

self.goodsBrandTF.isEnabled = false

self.countTF.text = result["number"].stringValue

self.unitTF.text = result["number\_unit"].stringValue

self.unitTF.isEnabled = false

self.priceTF.text = result["service\_price"].stringValue

self.priceTF.isEnabled = false

self.redoOrderDataJson = result

LYProgressHUD.dismiss()

}) { (error) in

LYProgressHUD.showError(error!)

}

}

@objc func backClick() {

self.view.endEditing(true)

let \_ = self.setUpParams(false)

if self.params.keys.count > 0 && !self.isRedoOrder{

LYAlertView.show("提示", "是否保存草稿，保存后下次打开将自动填充", "放弃", "保存", {

if self.params.keys.count > 0{

LocalData.saveSendTaskData(dict: self.params)

}

self.navigationController?.popViewController(animated: true)

},{

LocalData.saveSendTaskData(dict: [:])

self.navigationController?.popViewController(animated: true)

})

}else{

self.navigationController?.popViewController(animated: true)

}

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

NotificationCenter.default.addObserver(forName: NSNotification.Name(rawValue: ""), object: self, queue: OperationQueue.main) { (noti) in

let \_ = self.setUpParams(false)

if self.params.keys.count > 0{

LocalData.saveSendTaskData(dict: self.params)

}

}

}

override func viewWillDisappear(\_ animated: Bool) {

super.viewWillDisappear(animated)

NotificationCenter.default.removeObserver(self, name: NSNotification.Name(rawValue: ""), object: nil)

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

}

func loadEssentialData() {

NetTools.requestData(type: .post, urlString: SendTaskEssentialDataApi, succeed: { (resultJson, error) in

self.controlJson = resultJson["service\_form"]

self.rangeJson = resultJson["service\_sector"]

self.paymentJson = resultJson["payment\_list"]

self.serverTypeJson = resultJson["service\_type"]

self.top\_price = resultJson["top\_price"].stringValue

if !self.isRedoOrder{

//根据草稿填充信息

self.params = LocalData.getSendTaskData()

self.prepareLocalData()

}

}) { (error) in

}

}

//根据草稿填充信息

func prepareLocalData() {

if params.keys.count == 0{

return

}

if params.keys.contains("project\_name"){

self.taskNameTF.text = (params["project\_name"] as! String)

}

if params.keys.contains("service\_form"){

service\_form = params["service\_form"] as! String

for subJson in self.controlJson.arrayValue {

if subJson["field\_value"].stringValue.intValue == (params["service\_form"] as! String).intValue{

self.serverControlLbl.text = subJson["field\_name"].stringValue

}

}

}

if params.keys.contains("service\_type"){

service\_type = params["service\_type"] as! String

for subJson in self.serverTypeJson.arrayValue {

if subJson["field\_value"].stringValue.intValue == (params["service\_type"] as! String).intValue{

self.serverTypeLbl.text = subJson["field\_name"].stringValue

}

}

}

if params.keys.contains("service\_address") && params.keys.contains("lng") && params.keys.contains("lat"){

self.serverAreaLbl.text = (params["service\_address"] as! String)

self.lat = params["lat"] as! String

self.lon = params["lng"] as! String

}

if params.keys.contains("service\_stime"){

sTime = Date.init(timeIntervalSince1970: TimeInterval(params["service\_stime"] as! String)!)

self.serverStartTimeLbl.text = Date.dateStringFromDate(format: Date.datesChineseFormatString(), timeStamps: (params["service\_stime"] as! String))

}

if params.keys.contains("service\_etime"){

eTime = Date.init(timeIntervalSince1970: TimeInterval(params["service\_etime"] as! String)!)

self.serverEndTimeLbl.text = Date.dateStringFromDate(format: Date.datesChineseFormatString(), timeStamps: (params["service\_etime"] as! String))

}

if params.keys.contains("service\_sector"){

service\_sector = params["service\_sector"] as! String

let str = (params["service\_sector"] as! String)

let arr = str.components(separatedBy: ",")

var arrM = Array<String>()

self.selectedServerRangeIndex = arr

for subJson in self.rangeJson.arrayValue {

for str2 in arr {

if str2.intValue == subJson["gc\_id"].stringValue.intValue{

arrM.append(subJson["gc\_name"].stringValue)

}

}

}

self.serverRangeLbl.text = arrM.joined(separator: ",")

}

if params.keys.contains("service\_brand"){

self.goodsBrandTF.text = (params["service\_brand"] as! String)

}

if params.keys.contains("number"){

self.countTF.text = (params["number"] as! String)

}

if params.keys.contains("number\_unit"){

self.unitTF.text = (params["number\_unit"] as! String)

}

if params.keys.contains("service\_price"){

self.priceTF.text = (params["service\_price"] as! String)

}

}

func setUpParams(\_ showError:Bool) -> Bool{

let taskName = self.taskNameTF.text!

let serverArea = self.serverAreaLbl.text!

let serverRangeStr = self.serverRangeLbl.text!

let brand = self.goodsBrandTF.text!

let count = self.countTF.text!

let unit = self.unitTF.text!

let price = self.priceTF.text!

if taskName.isEmpty{

if showError {

LYProgressHUD.showError("请输入项目名称")

}

return false

}

params["project\_name"] = taskName

if service\_form.isEmpty{

if showError {

LYProgressHUD.showError("请选择服务形式")

}

return false

}

params["service\_form"] = service\_form

if service\_type.isEmpty{

if showError {

LYProgressHUD.showError("请选择服务类型")

}

return false

}

params["service\_type"] = service\_type

if serverArea.isEmpty{

if showError {

LYProgressHUD.showError("请选择服务区域")

}

return false

}

params["service\_address"] = serverArea

params["lng"] = self.lon

params["lat"] = self.lat

if (self.sTime?.phpTimestamp().isEmpty)! {

if showError {

LYProgressHUD.showError("请选择开始时间")

}

return false

}

params["service\_stime"] = self.sTime?.phpTimestamp()

if (self.eTime?.phpTimestamp().isEmpty)! {

if showError {

LYProgressHUD.showError("请选择结束时间")

}

return false

}

params["service\_etime"] = self.eTime?.phpTimestamp()

if showError{

if (sTime?.isLaterThanDate(aDate: eTime!))!{

LYProgressHUD.showError("开始时间不可晚于结束时间！")

return false

}

}

if service\_sector.isEmpty{

if showError {

LYProgressHUD.showError("请选择服务领域")

}

return false

}

params["service\_sector"] = service\_sector

if brand.isEmpty{

if showError {

LYProgressHUD.showError("请输入品牌型号")

}

return false

}

params["service\_brand"] = brand

if count.isEmpty && count.intValue > 0{

if showError {

LYProgressHUD.showError("请输入数量")

}

return false

}

params["number"] = count

if unit.isEmpty{

if showError {

LYProgressHUD.showError("请输入单位")

}

return false

}

params["number\_unit"] = unit

if price.isEmpty{

if showError {

LYProgressHUD.showError("请输入价格")

}

return false

}

if price.doubleValue <= 0{

if showError {

LYProgressHUD.showError("请输入标准价格")

}

return false

}

params["service\_price"] = price

params["title"] = serverRangeStr

return true

}

@objc func rightItemAction() {

if self.isRedoOrder{

var redoParams : [String : Any] = [:]

redoParams["id"] = self.orderId

let count = self.countTF.text!

if (self.sTime?.phpTimestamp().isEmpty)! {

LYProgressHUD.showError("请选择开始时间")

return

}

redoParams["service\_stime"] = self.sTime?.phpTimestamp()

if (self.eTime?.phpTimestamp().isEmpty)! {

LYProgressHUD.showError("请选择结束时间")

return

}

redoParams["service\_etime"] = self.eTime?.phpTimestamp()

if (sTime?.isLaterThanDate(aDate: eTime!))!{

LYProgressHUD.showError("开始时间不可晚于结束时间！")

return

}

if count.isEmpty && count.intValue > 0 {

LYProgressHUD.showError("请输入数量")

return

}

redoParams["number"] = count

let sendVC = SendTaskSureViewController.spwan()

sendVC.params = redoParams

sendVC.redoOrderDataJson = self.redoOrderDataJson

sendVC.isRedoOrder = self.isRedoOrder

self.navigationController?.pushViewController(sendVC, animated: true)

}else{

if self.setUpParams(true){

if self.isRepairOrder{

params["is\_compe"] = "1"//表示补单

let payVC = PaySendTaskViewController.spwan()

payVC.paymentJson = paymentJson

payVC.isRepairOrder = true

payVC.params = params

self.navigationController?.pushViewController(payVC, animated: true)

}else{

let sendVC = SendTaskSureViewController.spwan()

sendVC.params = params

sendVC.paymentJson = paymentJson

self.navigationController?.pushViewController(sendVC, animated: true)

}

}

}

}

}

extension SendTaskViewController{

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

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

self.view.endEditing(true)

if self.isRedoOrder{

if indexPath.row != 4 && indexPath.row != 5{

return

}

}

switch indexPath.section {

case 0:

switch indexPath.row {

case 1:

var arrM = Array<String>()

if self.controlJson.arrayValue.count > 0{

for subJson in self.controlJson.arrayValue {

arrM.append(subJson["field\_name"].stringValue)

}

}else{

arrM = ["现场服务","培训服务","远程服务"]

}

LYPickerView.show(titles: arrM, selectBlock: {[weak self] (title,index) in

self?.serverControlLbl.text = title

self?.service\_form = "\(index + 1)"

})

case 2:

var arrM = Array<String>()

if self.serverTypeJson.arrayValue.count > 0{

for subJson in self.serverTypeJson.arrayValue {

arrM.append(subJson["field\_name"].stringValue)

}

}else{

arrM = ["安装","巡检","调试","故障","方案","售前","培训","驻场","搬迁","咨询"]

}

LYPickerView.show(titles: arrM, selectBlock: {[weak self] (title,index) in

self?.serverTypeLbl.text = title

self?.service\_type = "\(index + 1)"

})

case 3:

let mapSearchVC = MapSearchViewController()

mapSearchVC.selectedLocation = {(poi) in

self.serverAreaLbl.text = poi.name

self.lon = "\(poi.location.longitude)"

self.lat = "\(poi.location.latitude)"

}

self.navigationController?.pushViewController(mapSearchVC, animated: true)

case 4:

//预约开始时间

let datePicker = LYDatePicker.init(component: 4)

datePicker.ly\_datepickerWithFourComponent = {(date,year,month,day,hour) in

self.sTime = date

self.serverStartTimeLbl.text = "\(year)年\(month)月\(day)日 \(hour)时"

}

datePicker.show()

case 5:

let datePicker = LYDatePicker.init(component: 4)

datePicker.ly\_datepickerWithFourComponent = {(date,year,month,day,hour) in

self.eTime = date

self.serverEndTimeLbl.text = "\(year)年\(month)月\(day)日 \(hour)时"

}

datePicker.show()

default:

break

}

case 1:

switch indexPath.row {

case 0:

let serverRangeVC = ServerRangeViewController.spwan()

serverRangeVC.serverRangeBlock = {[weak self] (selectedDictArray,titles,ids) in

self?.serverRangeLbl.text = titles.joined(separator: ";")

self?.selectedServerRangeIndex = ids

self?.service\_sector = ids.joined(separator: ",")

}

serverRangeVC.dataArray = self.rangeJson.arrayValue

self.navigationController?.pushViewController(serverRangeVC, animated: true)

case 1:

print(indexPath.row)

default:

break

}

default:

break

}

}

override func tableView(\_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {

return 10

}

override func scrollViewWillBeginDragging(\_ scrollView: UIScrollView) {

if self.taskNameTF.isFirstResponder{

self.view.endEditing(true)

}

}

}

extension SendTaskViewController : UITextFieldDelegate{

func textFieldShouldBeginEditing(\_ textField: UITextField) -> Bool {

textField.textAlignment = .left

return true

}

func textFieldDidEndEditing(\_ textField: UITextField) {

textField.textAlignment = .right

}

}

import UIKit

import SwiftyJSON

class ShopViewController: BaseViewController ,WXApiDelegate{

class func spwan() -> ShopViewController{

return self.loadFromStoryBoard(storyBoard: "Shop") as! ShopViewController

}

@IBOutlet weak var topView: UIView!

@IBOutlet weak var topViewH: NSLayoutConstraint!

@IBOutlet weak var leftTableView: UITableView!

@IBOutlet weak var rightTableView: UITableView!

@IBOutlet weak var suspendTableView: UITableView!

@IBOutlet weak var suspendTopDis: NSLayoutConstraint!

@IBOutlet weak var suspendTableH: NSLayoutConstraint!

@IBOutlet weak var blackBgBtn: UIButton!

@IBOutlet weak var emptyImgV: UIView!

fileprivate let searchBar : UISearchBar = UISearchBar()

fileprivate var leftDataArray : JSON = []

fileprivate var suspendDataArray : JSON = []

fileprivate var rightDataArray : Array<JSON> = Array<JSON>()

fileprivate var pluginDataArray : Array<JSON> = Array<JSON>()

fileprivate var bannerListArray : Array<JSON> = Array<JSON>()

fileprivate var curPage = 1

fileprivate var gcId = "0"

fileprivate var selectedLeftRow = 0

fileprivate var isPlugInShow = false

fileprivate lazy var bannerView : LYAnimateBannerView = {

let bannerView = LYAnimateBannerView(frame: CGRect(x: 0, y: 0, width: kScreenW, height: kScreenW \* 320 / 750), delegate: self)

bannerView.backgroundColor = UIColor.white

bannerView.showPageControl = true

self.topView.addSubview(bannerView)

return bannerView

}()

fileprivate var photoView : LYPhotoBrowseView!

override func viewDidLoad() {

super.viewDidLoad()

if #available(iOS 11.0, \*){

self.leftTableView.contentInsetAdjustmentBehavior = .never

self.rightTableView.contentInsetAdjustmentBehavior = .never

self.suspendTableView.contentInsetAdjustmentBehavior = .never

}else{

self.automaticallyAdjustsScrollViewInsets = false

}

self.topViewH.constant = kScreenW \* 320 / 750

self.rightTableView.register(UINib.init(nibName: "CollectGoodsCell", bundle: Bundle.main), forCellReuseIdentifier: "CollectGoodsCell")

self.leftTableView.register(UINib.init(nibName: "LeftCategoryCell", bundle: Bundle.main), forCellReuseIdentifier: "LeftCategoryCell")

self.suspendTableView.register(UINib.init(nibName: "LeftCategoryCell", bundle: Bundle.main), forCellReuseIdentifier: "LeftCategoryCell")

self.addRefresh()

self.setUpSearchNavView()

}

func setUpSearchNavView() {

searchBar.placeholder = "请输入品牌、名称、类别等搜索"

searchBar.searchBarStyle = .minimal

searchBar.delegate = self

self.navigationItem.titleView = searchBar

}

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

self.loadRightData()

self.loadLeftData()

}

override func viewWillDisappear(\_ animated: Bool) {

super.viewWillDisappear(animated)

self.hideSuspendTable()

self.bannerView.timer?.invalidate()

}

@IBAction func hideSuspendTable() {

self.suspendTableView.isHidden = true

self.blackBgBtn.isHidden = true

}

func showSuspendTable() {

self.suspendTableView.isHidden = false

self.blackBgBtn.isHidden = false

if self.suspendDataArray.count \* 55 > 220{

self.suspendTableH.constant = 220

}else{

self.suspendTableH.constant = CGFloat(self.suspendDataArray.count \* 55)

}

self.leftTableView.contentSize = CGSize.init(width: self.leftTableView.contentSize.width, height: CGFloat(55 \* (self.leftDataArray.count + self.suspendDataArray.count)))

}

}

extension ShopViewController {

func loadLeftData() {

var params : [String : Any] = [:]

params["store\_id"] = "1"

params["gc\_id"] = "0"

NetTools.requestData(type: .post, urlString: ShopCategoryDataApi,parameters: params, succeed: { (resultJson, msg) in

var bannerUrls : Array<String> = Array()

self.bannerListArray.removeAll()

for subJson in resultJson["banner\_list"].arrayValue{

bannerUrls.append(subJson["banner\_image"].stringValue)

self.bannerListArray.append(subJson)

}

self.bannerView.imageUrlArray = bannerUrls

self.leftDataArray = resultJson["class\_list"]

if self.leftDataArray.arrayValue.count > 0{

self.leftTableView.reloadData()

}

LYProgressHUD.dismiss()

}) { (error) in

LYProgressHUD.showError(error!)

}

}

func loadSuspendDataWithGcId(gcId : String) {

var params : [String : Any] = [:]

params["store\_id"] = "1"

params["gc\_id"] = gcId

LYProgressHUD.showLoading()

NetTools.requestData(type: .post, urlString: ShopCategoryDataApi,parameters: params, succeed: { (resultJson, msg) in

LYProgressHUD.dismiss()

self.suspendDataArray = resultJson["class\_list"]

if self.suspendDataArray.arrayValue.count > 0{

self.scrollViewDidScroll(self.leftTableView)

self.showSuspendTable()

if self.suspendDataArray.arrayValue.count > 0{

self.suspendTableView.reloadData()

}

}else{

self.hideSuspendTable()

}

}) { (error) in

LYProgressHUD.showError(error!)

}

}

func addRefresh() {

self.rightTableView.es.addPullToRefresh {

[weak self] in

self?.curPage = 1

if (self?.isPlugInShow)!{

self?.loadPluginData()

}else{

self?.loadRightData()

}

}

self.rightTableView.es.addInfiniteScrolling {

[weak self] in

self?.curPage += 1

if (self?.isPlugInShow)!{

self?.loadPluginData()

}else{

self?.loadRightData()

}

}

}

func loadRightData() {

var params : [String : Any] = [:]

params["store\_id"] = "1"//店铺ID

params["gc\_id"] = self.gcId//商品分类ID

params["key"] = "4"// 排序类型【1:销量】【2:人气（访问量）】【3:价格】【4:新品】

params["order"] = "2"//排序方式【1:升序】【2:降序】

params["curpage"] = "\(self.curPage)"//页数

NetTools.requestData(type: .post, urlString: ShopGoodsListApi, parameters: params, succeed: { (resultJson, msg) in

if self.curPage == 1{

self.rightTableView.es.stopPullToRefresh()

self.rightDataArray.removeAll()

}else{

self.rightTableView.es.stopLoadingMore()

}

for subJson in resultJson["goods\_list"].arrayValue{

self.rightDataArray.append(subJson)

}

if resultJson["goods\_list"].arrayValue.count < 10{

self.rightTableView.es.noticeNoMoreData()

}else{

self.rightTableView.es.resetNoMoreData()

}

if self.rightDataArray.count > 0{

self.emptyImgV.isHidden = true

self.rightTableView.reloadData()

}else{

self.emptyImgV.isHidden = false

}

LYProgressHUD.dismiss()

}) { (error) in

if self.curPage == 1{

self.rightTableView.es.stopPullToRefresh()

}else{

self.rightTableView.es.stopLoadingMore()

}

LYProgressHUD.showError(error!)

}

}

func loadPluginData() {

var params : [String : Any] = [:]

params["curpage"] = "\(self.curPage)"//页数

NetTools.requestData(type: .post, urlString: PluginListApi, parameters: params, succeed: { (result, msg) in

if self.curPage == 1{

self.rightTableView.es.stopPullToRefresh()

self.pluginDataArray.removeAll()

}else{

self.rightTableView.es.stopLoadingMore()

}

for subJson in result.arrayValue{

self.pluginDataArray.append(subJson)

}

if result.arrayValue.count < 10{

self.rightTableView.es.noticeNoMoreData()

}else{

self.rightTableView.es.resetNoMoreData()

}

if self.pluginDataArray.count > 0{

self.emptyImgV.isHidden = true

self.rightTableView.reloadData()

}else{

self.emptyImgV.isHidden = false

}

}) { (error) in

if self.curPage == 1{

self.rightTableView.es.stopPullToRefresh()

}else{

self.rightTableView.es.stopLoadingMore()

}

LYProgressHUD.showError(error!)

}

}

}

extension ShopViewController : LYAnimateBannerViewDelegate,UISearchBarDelegate{

func LY\_AnimateBannerViewClick(banner: LYAnimateBannerView, index: NSInteger) {

let subJson = self.bannerListArray[index]

if subJson["banner\_url"].stringValue.trim.isEmpty{

return

}

let webVC = BaseWebViewController.spwan()

webVC.urlStr = subJson["banner\_url"].stringValue

webVC.titleStr = subJson["banner\_title"].stringValue

self.navigationController?.pushViewController(webVC, animated: true)

}

func searchBarShouldBeginEditing(\_ searchBar: UISearchBar) -> Bool {

let searchVC = GoodsSearchListViewController.spwan()

self.navigationController?.pushViewController(searchVC, animated: true)

return false

}

}

extension ShopViewController : UITableViewDelegate,UITableViewDataSource{

func numberOfSections(in tableView: UITableView) -> Int {

return 1

}

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

if tableView == self.leftTableView{

return self.leftDataArray.arrayValue.count + 2

}else if tableView == self.rightTableView{

if self.isPlugInShow{

return self.pluginDataArray.count

}

return self.rightDataArray.count

}else if tableView == self.suspendTableView{

return self.suspendDataArray.arrayValue.count

}

return 0

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

if tableView == self.leftTableView || tableView == self.suspendTableView{

let cell = tableView.dequeueReusableCell(withIdentifier: "LeftCategoryCell", for: indexPath) as! LeftCategoryCell

if tableView == self.leftTableView{

if indexPath.row == 0{

cell.nameLbl.text = "全部商品"

}else{

if self.leftDataArray.arrayValue.count > indexPath.row - 1 {

let subJson = self.leftDataArray.arrayValue[indexPath.row - 1]

cell.nameLbl.text = subJson["gc\_name"].stringValue

}else{

cell.nameLbl.text = "附加软件"

}

}

}else{

if self.suspendDataArray.arrayValue.count > indexPath.row {

let subJson = self.suspendDataArray.arrayValue[indexPath.row]

cell.nameLbl.text = subJson["gc\_name"].stringValue

cell.nameLbl.backgroundColor = UIColor.white

}

}

return cell

}else if tableView == self.rightTableView{

if self.isPlugInShow{

var cell = tableView.dequeueReusableCell(withIdentifier: "PlugListCell")

if cell == nil{

cell = UITableViewCell.init(style: .value1, reuseIdentifier: "PlugListCell")

}

if self.pluginDataArray.count > indexPath.row{

let json = self.pluginDataArray[indexPath.row]

cell!.textLabel?.text = json["plugname"].stringValue

cell!.detailTextLabel?.text = "销量:" + json["paynum"].stringValue

cell?.textLabel?.textColor = Text\_Color

cell?.textLabel?.font = UIFont.systemFont(ofSize: 14.0)

cell?.detailTextLabel?.textColor = Text\_Color

cell?.detailTextLabel?.font = UIFont.systemFont(ofSize: 14.0)

cell?.accessoryType = .disclosureIndicator

let line = UIView(frame:CGRect.init(x: 0, y: 49, width: self.rightTableView.w, height: 1))

line.backgroundColor = BG\_Color

cell!.addSubview(line)

}

return cell!

}else{

let cell = tableView.dequeueReusableCell(withIdentifier: "CollectGoodsCell", for: indexPath) as! CollectGoodsCell

if self.rightDataArray.count > indexPath.row{

let subJson = self.rightDataArray[indexPath.row]

cell.subJson = subJson

}

return cell

}

}

return UITableViewCell()

}

func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

if tableView == self.rightTableView{

if self.isPlugInShow{

return 50

}

return 100

}else {

return 55

}

}

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

if tableView == self.leftTableView{

self.isPlugInShow = false

if indexPath.row == 0{

if self.leftDataArray.arrayValue.count > indexPath.row {

self.gcId = "0"

self.loadRightData()

self.hideSuspendTable()

}

}else{

if self.leftDataArray.arrayValue.count > indexPath.row - 1 {

let subJson = self.leftDataArray.arrayValue[indexPath.row - 1]

self.loadSuspendDataWithGcId(gcId: subJson["gc\_id"].stringValue)

}else{

self.hideSuspendTable()

self.isPlugInShow = true

self.curPage = 1

self.loadPluginData()

}

}

self.suspendTopDis.constant = CGFloat(indexPath.row \* 55)

if self.selectedLeftRow == 0{

let cell1 = tableView.cellForRow(at: IndexPath.init(row: self.selectedLeftRow, section: 0)) as? LeftCategoryCell

if cell1 != nil{

cell1?.nameLbl.backgroundColor = UIColor.RGBS(s: 240)

}

}

self.selectedLeftRow = indexPath.row

let cell = tableView.cellForRow(at: indexPath) as! LeftCategoryCell

cell.nameLbl.backgroundColor = UIColor.white

}else if tableView == self.suspendTableView{

self.hideSuspendTable()

if self.suspendDataArray.arrayValue.count > indexPath.row {

self.curPage = 1

let subJson = self.suspendDataArray.arrayValue[indexPath.row]

self.gcId = subJson["gc\_id"].stringValue

self.rightTableView.setContentOffset(CGPoint.zero, animated: false)

self.loadRightData()

}

}else if tableView == self.rightTableView{

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

if self.isPlugInShow{

if self.pluginDataArray.count > indexPath.row{

let plugInDetailVC = PlugInDetailViewController.spwan()

plugInDetailVC.plugId = self.pluginDataArray[indexPath.row]["plugid"].stringValue

self.navigationController?.pushViewController(plugInDetailVC, animated: true)

}

}else{

if self.rightDataArray.count > indexPath.row{

let subJson = self.rightDataArray[indexPath.row]

let detailVC = GoodsDetailViewController.spwan()

detailVC.goodsId = subJson["goods\_id"].stringValue

self.navigationController?.pushViewController(detailVC, animated: true)

}

}

}

}

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

if tableView == self.leftTableView{

guard let cell = tableView.cellForRow(at: indexPath) as? LeftCategoryCell else{

return

}

cell.nameLbl.backgroundColor = UIColor.RGBS(s: 240)

}

}

func tableView(\_ tableView: UITableView, willDisplay cell: UITableViewCell, forRowAt indexPath: IndexPath) {

if tableView == self.leftTableView {

let ce = cell as! LeftCategoryCell

if indexPath.row == self.selectedLeftRow {

ce.nameLbl.backgroundColor = UIColor.white

}else{

ce.nameLbl.backgroundColor = UIColor.RGBS(s: 240)

}

}

}

func scrollViewDidScroll(\_ scrollView: UIScrollView) {

let tempTable = scrollView as? UITableView

if self.leftTableView.isDecelerating || self.leftTableView.isDragging || (tempTable != nil && tempTable == self.leftTableView){

self.suspendTopDis.constant = CGFloat(self.selectedLeftRow \* 55) - self.leftTableView.contentOffset.y

}

}

}

import UIKit

import SwiftyJSON

class GoodsDetailViewController: BaseViewController {

class func spwan() -> GoodsDetailViewController{

return self.loadFromStoryBoard(storyBoard: "Shop") as! GoodsDetailViewController

}

var goodsId = ""

fileprivate var resultJson : JSON = []

fileprivate var showGoodsTable = false

fileprivate var isCollected = false

fileprivate var isBuying = true

fileprivate var goodsDescH : CGFloat = 0

fileprivate var goodsParamsH : CGFloat = 0

fileprivate var leftNavBtn = UIButton()

fileprivate var rightNavBtn = UIButton()

fileprivate var navLine = UIView()

@IBOutlet weak var buyView: UIView!

@IBOutlet weak var buyCountLbl: UILabel!

@IBOutlet weak var tableView: UITableView!

@IBOutlet weak var chatBuyBtn: UIButton!

lazy var bannerView : LYAnimateBannerView = {

let bannerView = LYAnimateBannerView(frame: CGRect(x: 0, y: 0, width: kScreenW, height: kScreenW), delegate: self)

bannerView.backgroundColor = UIColor.white

bannerView.showPageControl = true

return bannerView

}()

override func viewDidLoad() {

super.viewDidLoad()

self.tableView.register(UINib.init(nibName: "ShopInventoryCell", bundle: Bundle.main), forCellReuseIdentifier: "ShopInventoryCell")

self.tableView.register(UINib.init(nibName: "WebViewCell", bundle: Bundle.main), forCellReuseIdentifier: "WebViewCell")

self.tableView.register(UINib.init(nibName: "GoodsDetailPriceCell", bundle: Bundle.main), forCellReuseIdentifier: "GoodsDetailPriceCell")

if #available(iOS 11.0, \*){

self.tableView.contentInsetAdjustmentBehavior = .never

self.tableView.contentInset = UIEdgeInsets.init(top: 0, left: 0, bottom: 0, right: 0)

self.tableView.scrollIndicatorInsets = self.tableView.contentInset

}else{

self.automaticallyAdjustsScrollViewInsets = false

}

self.tableView.estimatedRowHeight = 0

self.tableView.estimatedSectionFooterHeight = 0

self.tableView.estimatedSectionHeaderHeight = 0

self.loadDetailData()

self.setUpNavView()

self.buyCountLbl.text = "1"

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

}

func setUpNavView() {

let view = UIView(frame:CGRect.init(x: 0, y: 0, width: 120, height: 40))

self.leftNavBtn = UIButton(frame:CGRect.init(x: 0, y: 0, width: 60, height: 40))

leftNavBtn.setTitle("商品", for: .normal)

leftNavBtn.setTitleColor(Normal\_Color, for: .normal)

leftNavBtn.addTarget(self, action: #selector(GoodsDetailViewController.leftNavAction), for: .touchUpInside)

self.rightNavBtn = UIButton(frame:CGRect.init(x: 60, y: 0, width: 60, height: 40))

rightNavBtn.setTitle("详情", for: .normal)

rightNavBtn.setTitleColor(UIColor.RGBS(s: 33), for: .normal)

rightNavBtn.addTarget(self, action: #selector(GoodsDetailViewController.rightNavAction), for: .touchUpInside)

self.navLine = UIView(frame:CGRect.init(x: 0, y: 39, width: 60, height: 1.5))

navLine.backgroundColor = Normal\_Color

view.addSubview(leftNavBtn)

view.addSubview(rightNavBtn)

view.addSubview(navLine)

self.navigationItem.titleView = view

self.setUpRightItems()

}

@objc func leftNavAction() {

if self.navLine.x != 0{

self.navLine.x = 0

self.leftNavBtn.setTitleColor(Normal\_Color, for: .normal)

self.rightNavBtn.setTitleColor(UIColor.RGBS(s: 33), for: .normal)

self.showGoodsTable = false

self.tableView.reloadData()

}

}

@objc func rightNavAction() {

if self.navLine.x != 60{

self.navLine.x = 60

self.leftNavBtn.setTitleColor(UIColor.RGBS(s: 33), for: .normal)

self.rightNavBtn.setTitleColor(Normal\_Color, for: .normal)

self.showGoodsTable = true

self.tableView.reloadData()

}

}

func setUpRightItems() {

var searchItem = UIBarButtonItem()

if self.isCollected{

searchItem = UIBarButtonItem(image: imageLiteral(resourceName: "icon\_collect\_red"), target: self, action: #selector(GoodsDetailViewController.collectAction))

}else{

searchItem = UIBarButtonItem(image: imageLiteral(resourceName: "icon\_collect"), target: self, action: #selector(GoodsDetailViewController.collectAction))

}

let filtrateItem = UIBarButtonItem(image: imageLiteral(resourceName: "icon\_share"), target: self, action: #selector(GoodsDetailViewController.shareAction))

self.navigationItem.rightBarButtonItems = [filtrateItem,searchItem]

}

@IBAction func hideBuyView() {

self.buyView.isHidden = true

self.buyCountLbl.text = "1"

}

@IBAction func sureBuyView() {

let count = self.buyCountLbl.text?.intValue

if isBuying{

let goods : JSON = ["name" : resultJson["goods\_info"]["goods\_name"].stringValue,

"icon" : resultJson["goods\_image"].arrayValue[0],

"price" : resultJson["goods\_info"]["goods\_price"].stringValue,

"count" : "\(count ?? 1)"

]

let payVC = PayGoodsViewController.spwan()

payVC.cartId = self.goodsId + "|" + "\(count ?? 1)"

payVC.isFromShopCar = false

payVC.goodsArray = [goods]

self.navigationController?.pushViewController(payVC, animated: true)

self.hideBuyView()

}else{

var params : [String : Any] = [:]

params["store\_id"] = "1";

params["goods\_id"] = self.goodsId

params["quantity"] = "\(count ?? 1)"

NetTools.requestData(type: .post, urlString: GoodsAddToCarApi, parameters: params, succeed: { (resultJson, msg) in

LYProgressHUD.showSuccess(msg ?? "添加购物车成功！")

self.hideBuyView()

}, failure: { (error) in

LYProgressHUD.showError(error!)

})

}

}

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

if sender.tag == 11{

let count = (self.buyCountLbl.text?.intValue)! + 1

if count > resultJson["goods\_info"]["goods\_storage"].stringValue.intValue{

self.buyCountLbl.text = "\(count - 1)"

}else{

self.buyCountLbl.text = "\(count)"

}

}else{

let count = (self.buyCountLbl.text?.intValue)! - 1

if count < 1{

self.buyCountLbl.text = "\(1)"

}else{

self.buyCountLbl.text = "\(count)"

}

}

}

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

if sender.tag == 11{

if resultJson["goods\_info"]["goods\_storage"].stringValue.intValue > 0{

self.buyView.isHidden = false

self.isBuying = false

}else{

LYProgressHUD.showInfo("商品库存不足！")

}

}else if sender.tag == 22{

if resultJson["goods\_info"]["goods\_storage"].stringValue.intValue > 0{

self.buyView.isHidden = false

self.isBuying = true

}else{

LYProgressHUD.showInfo("商品库存不足！")

}

}else{

var goodsInfo : [AnyHashable : Any] = [AnyHashable : Any]()

goodsInfo["type"] = "track"

goodsInfo["title"] = resultJson["goods\_info"]["goods\_name"].stringValue

if resultJson["goods\_image"].arrayValue.count > 0{

goodsInfo["desc"] = resultJson["goods\_image"].arrayValue[0].stringValue

}else{

goodsInfo["desc"] = ""

}

goodsInfo["price"] = resultJson["goods\_info"]["goods\_price"].stringValue

goodsInfo["img\_url"] = resultJson["goods\_info"]["goods\_storage"].stringValue

goodsInfo["item\_url"] = resultJson["goods\_info"]["share"].stringValue

if !HChatClient.shared().isLoggedInBefore{

HChatClient.shared().login(withUsername: LocalData.getUserPhone(), password: "11")

}

let chatVC = HDChatViewController.init(conversationChatter: "kefu1")

chatVC?.commodityInfo = goodsInfo

self.navigationController?.pushViewController(chatVC!, animated: true)

}

}

func loadDetailData() {

var params : [String : Any] = [:]

params["goods\_id"] = self.goodsId

LYProgressHUD.showLoading()

NetTools.requestData(type: .post, urlString: ShopGoodsDetailApi,parameters: params, succeed: { (resultJson, msg) in

LYProgressHUD.dismiss()

var arrM = Array<String>()

for subJson in resultJson["goods\_image"].arrayValue{

arrM.append(subJson.stringValue)

}

self.bannerView.imageUrlArray = arrM

if arrM.count < 2{

self.bannerView.showPageControl = false

}

self.resultJson = resultJson

self.tableView.reloadData()

if resultJson["goods\_info"]["goods\_storage"].stringValue.intValue > 0{

self.chatBuyBtn.isHidden = true

}else{

self.chatBuyBtn.isHidden = false

}

if resultJson["is\_fav"].intValue == 1{

self.isCollected = true

self.setUpRightItems()

}

}) { (error) in

LYProgressHUD.dismiss()

}

}

}

extension GoodsDetailViewController : LYAnimateBannerViewDelegate, UIWebViewDelegate{

func LY\_AnimateBannerViewClick(banner: LYAnimateBannerView, index: NSInteger) {

}

func webViewDidFinishLoad(\_ webView: UIWebView) {

self.tableView.reloadData()

}

@objc func collectAction() {

var params : [String : Any] = [:]

params["store\_id"] = "1"

var url = ""

if self.isCollected{

params["fav\_id"] = self.goodsId

url = CancelCollectGoodsApi

}else{

params["goods\_id"] = self.goodsId

url = CollectGoodsApi

}

LYProgressHUD.showLoading()

NetTools.requestData(type: .post, urlString: url,parameters: params, succeed: { (resultJson, msg) in

if self.isCollected{

LYProgressHUD.showSuccess("取消成功！")

}else{

LYProgressHUD.showSuccess("收藏成功！")

}

self.isCollected = !self.isCollected

self.setUpRightItems()

}) { (error) in

LYProgressHUD.showError(error!)

}

}

@objc func shareAction() {

func shareWithImage(image:UIImage?){

ShareView.show(url: self.resultJson["goods\_info"]["share"].stringValue, title: self.resultJson["goods\_info"]["goods\_name"].stringValue, desc: "七小服备件商品", image:image, viewController: self)

}

if resultJson["goods\_image"].arrayValue.count > 0{

let imgV = UIImageView.init(frame: CGRect.init(origin: CGPoint.zero, size: CGSize.init(width: 100, height: 100)))

imgV.kf.setImage(with: URL(string:(resultJson["goods\_image"].arrayValue.first?.stringValue)!), placeholder: nil, options: nil, progressBlock: nil, completionHandler: { (img, error, memory, imgUrl) in

shareWithImage(image: img)

})

}else{

shareWithImage(image: nil)

}

}

}

extension GoodsDetailViewController : UITableViewDelegate,UITableViewDataSource{

func numberOfSections(in tableView: UITableView) -> Int {

return 3

}

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

if section == 1{

return 5

}

return 1

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

if indexPath.section == 0{

let cell = tableView.dequeueReusableCell(withIdentifier: "WebViewCell", for: indexPath) as! WebViewCell

cell.setHtmlStr(resultJson["goods\_info"]["goods\_table"].stringValue)

cell.webCellHeightBlock = {[weak self] (height , webView) in

if self?.goodsParamsH != height{

self?.goodsParamsH = height

self?.tableView.reloadData()

}

}

if self.showGoodsTable{

cell.isHidden = false

}else{

cell.isHidden = true

}

return cell

}else if indexPath.section == 1{

if indexPath.row == 0{

let cell = UITableViewCell()

cell.addSubview(self.bannerView)

if self.showGoodsTable{

cell.isHidden = true

}else{

cell.isHidden = false

}

return cell

}else if indexPath.row == 1{

let cell = tableView.dequeueReusableCell(withIdentifier: "GoodsDetailPriceCell", for: indexPath) as! GoodsDetailPriceCell

cell.subJson = resultJson

if resultJson["goods\_info"]["is\_discount"].stringValue.intValue == 1{

cell.isHidden = false

}else{

cell.isHidden = true

}

return cell

}else if indexPath.row == 2{

let cell = UITableViewCell()

cell.textLabel?.text = resultJson["goods\_info"]["goods\_name"].stringValue

cell.textLabel?.textColor = UIColor.RGBS(s: 33)

if self.showGoodsTable{

cell.isHidden = true

}else{

cell.isHidden = false

}

let line = UIView(frame:CGRect.init(x: 0, y: 0, width: kScreenW, height: 1))

line.backgroundColor = BG\_Color

cell.addSubview(line)

return cell

}else if indexPath.row == 3{

let cell = tableView.dequeueReusableCell(withIdentifier: "WebViewCell", for: indexPath) as! WebViewCell

cell.setHtmlStr(resultJson["goods\_info"]["mobile\_body"].stringValue)

cell.webCellHeightBlock = {[weak self] (height , webView) in

if self?.goodsDescH != height{

self?.goodsDescH = height

self?.tableView.reloadData()

}

}

if self.showGoodsTable{

cell.isHidden = true

}else{

cell.isHidden = false

}

return cell

}else if indexPath.row == 4{

let cell = UITableViewCell()

cell.textLabel?.text = "¥ " + resultJson["goods\_info"]["goods\_price"].stringValue

cell.textLabel?.textColor = Normal\_Color

if self.showGoodsTable{

cell.isHidden = true

}else{

cell.isHidden = false

}

let line = UIView(frame:CGRect.init(x: 0, y: 0, width: kScreenW, height: 1))

line.backgroundColor = BG\_Color

cell.addSubview(line)

if resultJson["goods\_info"]["is\_discount"].stringValue.intValue == 1{

cell.isHidden = true

}else{

cell.isHidden = false

}

return cell

}

}else if indexPath.section == 2{

let cell = tableView.dequeueReusableCell(withIdentifier: "ShopInventoryCell", for: indexPath) as! ShopInventoryCell

cell.shopInventoryLbl.text = resultJson["goods\_info"]["goods\_storage"].stringValue

cell.engineerInventoryLbl.text = resultJson["goods\_info"]["engineer\_storage"].stringValue

if self.showGoodsTable{

cell.isHidden = true

}else{

cell.isHidden = false

}

return cell

}

return UITableViewCell()

}

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

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

if indexPath.section == 2{

let goodsLocationVC = GoodsLocationViewController()

goodsLocationVC.goodsId = self.goodsId

self.navigationController?.pushViewController(goodsLocationVC, animated: true)

}

}

func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

if self.showGoodsTable{

if indexPath.section == 0{

return self.goodsParamsH

}

}else{

if indexPath.section == 1{

if indexPath.row == 0{

return kScreenW

}else if indexPath.row == 1{

if resultJson["goods\_info"]["is\_discount"].stringValue.intValue == 1{

return 60

}else{

return 0

}

}else if indexPath.row == 2{

return 44

}else if indexPath.row == 3{

return self.goodsDescH

}else if indexPath.row == 4{

if resultJson["goods\_info"]["is\_discount"].stringValue.intValue == 1{

return 0

}else{

return 44

}

}

}else if indexPath.section == 2{

return 72

}

}

return 0

}

}

import UIKit

import SwiftyJSON

class DiscoverViewController: BaseViewController {

class func spwan() -> DiscoverViewController{

return self.loadFromStoryBoard(storyBoard: "Knowledge") as! DiscoverViewController

}

@IBOutlet weak var scrollView: UIScrollView!

@IBOutlet weak var leftTableView: UITableView!

@IBOutlet weak var middleTableView: UITableView!

@IBOutlet weak var rightTableView: UITableView!

@IBOutlet weak var sortCollectionView: UICollectionView!

@IBOutlet weak var sortLeftView: UIView!

@IBOutlet weak var scrollViewW: NSLayoutConstraint!

@IBOutlet weak var sortViewH: NSLayoutConstraint!

@IBOutlet weak var courseBtn: UIButton!

@IBOutlet weak var videoBtn: UIButton!

@IBOutlet weak var knowledgeBtn: UIButton!

@IBOutlet weak var topScrolView: UIView!

@IBOutlet weak var topScrolViewW: NSLayoutConstraint!

@IBOutlet weak var knowledgeSortView: UIView!

@IBOutlet weak var newBtn: UIButton!

@IBOutlet weak var hotBtn: UIButton!

fileprivate var videoArray : Array<JSON> = Array<JSON>()

fileprivate var courseArray : Array<JSON> = Array<JSON>()

fileprivate var knowledgeArray : Array<JSON> = Array<JSON>()

fileprivate var knowledgeSortArray : Array<JSON> = Array<JSON>()

fileprivate let secOneLbl = UILabel()

fileprivate let secTwoLbl = UILabel()

fileprivate let btnLine = UIView()

fileprivate var curpage = 1

fileprivate var keyWord = ""

fileprivate var typeId = "0"

fileprivate var sortId = "0"

fileprivate var selectedSecOneIndex = 0

fileprivate let searchBar : UISearchBar = UISearchBar()

override func viewDidLoad() {

super.viewDidLoad()

self.addRefresh()

self.setUPTableConfig()

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

self.showSortView(show: false)

}

func setUPTableConfig() {

self.scrollViewW.constant = kScreenW \* 3

self.topScrolViewW.constant = kScreenW

self.middleTableView.register(UINib.init(nibName: "DiscoverVideoCell", bundle: Bundle.main), forCellReuseIdentifier: "DiscoverVideoCell")

self.leftTableView.register(UINib.init(nibName: "DiscoverCourseCell", bundle: Bundle.main), forCellReuseIdentifier: "DiscoverCourseCell")

self.rightTableView.register(UINib.init(nibName: "KnowledgeListCell", bundle: Bundle.main), forCellReuseIdentifier: "KnowledgeListCell")

self.sortCollectionView.register(UICollectionViewCell.self, forCellWithReuseIdentifier: "SortCollectionCell")

self.secOneLbl.text = "品牌:"

self.secOneLbl.font = UIFont.systemFont(ofSize: 14.0)

self.secOneLbl.textColor = UIColor.colorHex(hex: "575757")

self.secOneLbl.textAlignment = .center

self.sortLeftView.addSubview(secOneLbl)

self.secTwoLbl.text = "系列:"

self.secTwoLbl.font = UIFont.systemFont(ofSize: 14.0)

self.secTwoLbl.textColor = UIColor.colorHex(hex: "575757")

self.secTwoLbl.textAlignment = .center

self.sortLeftView.addSubview(secTwoLbl)

self.btnLine.backgroundColor = Normal\_Color

self.btnLine.frame = CGRect.init(x: 0, y: self.topScrolView.h-3, width: 50, height: 2)

self.topScrolView.addSubview(self.btnLine)

self.courseBtn.w = kScreenW / 3.0

self.topBtnAction(self.courseBtn)

self.newBtn.setTitleColor(Normal\_Color, for: .normal)

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

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

if sender.tag == 111{

self.hotBtn.setTitleColor(UIColor.RGBS(s: 145), for: .normal)

self.newBtn.setTitleColor(Normal\_Color, for: .normal)

self.sortId = "1"

self.loadKnowledgeData()

self.showSortView(show: false)

}else if sender.tag == 222{

self.newBtn.setTitleColor(UIColor.RGBS(s: 145), for: .normal)

self.hotBtn.setTitleColor(Normal\_Color, for: .normal)

self.sortId = "2"

self.loadKnowledgeData()

self.showSortView(show: false)

}else if sender.tag == 333{

if self.knowledgeSortView.isHidden {

self.loadKnowledgeSort()

}else{

self.showSortView(show: false)

}

}

}

func showSortView(show : Bool) {

if show && self.knowledgeSortView.isHidden{

self.knowledgeSortView.isHidden = false

}else{

self.knowledgeSortView.isHidden = true

}

}

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

if sender.isSelected == true {return}

self.courseBtn.isSelected = false

self.videoBtn.isSelected = false

self.knowledgeBtn.isSelected = false

sender.isSelected = true

UIView.animate(withDuration: 0.25) {

self.btnLine.x = sender.centerX - 25

}

if sender.tag == 11{

self.loadCourseData()

self.scrollView.scrollRectToVisible(CGRect.init(x:0, y: 0, width: self.scrollView.w, height: self.scrollView.h), animated: true)

self.cancelAction()

self.setUpRightItem(show: false)

}else if sender.tag == 22{

self.loadVideoData()

self.scrollView.scrollRectToVisible(CGRect.init(x:self.scrollView.w, y: 0, width: self.scrollView.w, height: self.scrollView.h), animated: true)

if self.navigationItem.titleView == nil{

self.setUpRightItem(show: true)

}

}else if sender.tag == 33{

self.loadKnowledgeData()

self.scrollView.scrollRectToVisible(CGRect.init(x:self.scrollView.w \* 2, y: 0, width: self.scrollView.w, height: self.scrollView.h), animated: true)

if self.navigationItem.titleView == nil{

self.setUpRightItem(show: true)

}

}

}

}

extension DiscoverViewController {

func addRefresh() {

self.leftTableView.es.addPullToRefresh {

self.curpage = 1

self.loadCourseData()

}

self.leftTableView.es.addInfiniteScrolling {

self.curpage += 1

self.loadCourseData()

}

self.middleTableView.es.addPullToRefresh {

self.curpage = 1

self.loadVideoData()

}

self.middleTableView.es.addInfiniteScrolling {

self.curpage += 1

self.loadVideoData()

}

self.rightTableView.es.addPullToRefresh {

self.curpage = 1

self.loadKnowledgeData()

}

self.rightTableView.es.addInfiniteScrolling {

self.curpage += 1

self.loadKnowledgeData()

}

}

func loadCourseData() {

var params : [String : Any] = [:]

params["curpage"] = self.curpage

NetTools.requestData(type: .post, urlString: KCourseListApi, parameters: params, succeed: { (result, msg) in

if self.curpage == 1{

self.leftTableView.es.stopPullToRefresh()

self.courseArray.removeAll()

}else{

self.leftTableView.es.stopLoadingMore()

}

if result.arrayValue.count < 10{

self.leftTableView.es.noticeNoMoreData()

}else{

self.leftTableView.es.resetNoMoreData()

}

for subJson in result.arrayValue{

self.courseArray.append(subJson)

}

if self.courseArray.count > 0{

self.hideEmptyView()

}else{

self.showEmptyView()

}

self.leftTableView.reloadData()

}) { (error) in

LYProgressHUD.showError(error!)

}

}

func loadVideoData() {

var params : [String : Any] = [:]

params["curpage"] = self.curpage

params["keywords"] = self.keyWord

NetTools.requestData(type: .post, urlString: KVideoListApi, parameters: params, succeed: { (result, msg) in

if self.curpage == 1{

self.middleTableView.es.stopPullToRefresh()

self.videoArray.removeAll()

}else{

self.middleTableView.es.stopLoadingMore()

}

if result.arrayValue.count < 10{

self.middleTableView.es.noticeNoMoreData()

}else{

self.middleTableView.es.resetNoMoreData()

}

for subJson in result.arrayValue{

self.videoArray.append(subJson)

}

if self.videoArray.count > 0{

self.hideEmptyView()

}else{

self.showEmptyView()

}

self.middleTableView.reloadData()

}) { (error) in

LYProgressHUD.showError(error!)

}

}

func loadKnowledgeData() {

var params : [String : Any] = [:]

params["type\_id"] = self.typeId

params["sortid"] = self.sortId

params["curpage"] = self.curpage

params["keyword"] = self.keyWord

NetTools.requestData(type: .post, urlString: KnowledgeListApi, parameters: params, succeed: { (result, msg) in

if self.curpage == 1{

self.rightTableView.es.stopPullToRefresh()

self.knowledgeArray.removeAll()

}else{

self.rightTableView.es.stopLoadingMore()

}

if result.arrayValue.count < 10{

self.rightTableView.es.noticeNoMoreData()

}else{

self.rightTableView.es.resetNoMoreData()

}

for subJson in result.arrayValue{

self.knowledgeArray.append(subJson)

}

if self.knowledgeArray.count > 0{

self.hideEmptyView()

}else{

self.showEmptyView()

}

self.rightTableView.reloadData()

}) { (error) in

LYProgressHUD.showError(error!)

}

}

func loadKnowledgeSort() {

self.knowledgeSortArray.removeAll()

NetTools.requestData(type: .post, urlString: KnowledgeCategoryApi, succeed: { (result, msg) in

let json = JSON(["id" : "0", "name" : "所有品牌", "smallList" : [["id" : "0", "name" : "所有系列"]]])

self.knowledgeSortArray.append(json)

for subJson in result.arrayValue{

self.knowledgeSortArray.append(subJson)

}

self.sortCollectionView.reloadData()

self.showSortView(show: true)

}) { (error) in

LYProgressHUD.showError("获取筛选条件失败！")

}

}

}

extension DiscoverViewController : UITableViewDelegate,UITableViewDataSource{

func numberOfSections(in tableView: UITableView) -> Int {

return 1

}

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

if tableView == self.leftTableView{

return self.courseArray.count

}else if tableView == self.middleTableView{

return self.videoArray.count

}else if tableView == self.rightTableView{

return self.knowledgeArray.count

}

return 0

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

if tableView == self.leftTableView{

let cell = tableView.dequeueReusableCell(withIdentifier: "DiscoverCourseCell", for: indexPath) as! DiscoverCourseCell

if self.courseArray.count > indexPath.row{

let subJson = self.courseArray[indexPath.row]

cell.subJson = subJson

}

return cell

}else if tableView == self.middleTableView{

let cell = tableView.dequeueReusableCell(withIdentifier: "DiscoverVideoCell", for: indexPath) as! DiscoverVideoCell

if self.videoArray.count > indexPath.row{

let subJson = self.videoArray[indexPath.row]

cell.subJson = subJson

}

return cell

}else if tableView == self.rightTableView{

let cell = tableView.dequeueReusableCell(withIdentifier: "KnowledgeListCell", for: indexPath) as! KnowledgeListCell

if self.knowledgeArray.count > indexPath.row{

let subJson = self.knowledgeArray[indexPath.row]

cell.subJson = subJson

}

return cell

}

return UITableViewCell()

}

func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

if tableView == self.leftTableView{

return (kScreenW - 16) / 2.0 + 20

}else if tableView == self.middleTableView{

return (kScreenW - 16) / 2.0 + 20

}else if tableView == self.rightTableView{

return 125

}

return 0

}

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

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

if tableView == self.leftTableView{

if self.courseArray.count > indexPath.row{

var subJson = self.courseArray[indexPath.row]

let courseDetailVC = CourseDetailViewController.spwan()

courseDetailVC.courseId = subJson["lession\_id"].stringValue

courseDetailVC.status = subJson["lession\_enroll\_state"].stringValue

courseDetailVC.coursePaySuccessBlock = {() in

subJson["lession\_enroll\_state"] = JSON("1")

self.courseArray.remove(at: indexPath.row)

self.courseArray.insert(subJson, at: indexPath.row)

self.leftTableView.reloadRows(at: [indexPath], with: .none)

}

self.navigationController?.pushViewController(courseDetailVC, animated: true)

}

}else if tableView == self.middleTableView{

if self.videoArray.count > indexPath.row{

let subJson = self.videoArray[indexPath.row]

let videoPlayVC = KnowledgeVideoPlayViewController.spwan()

videoPlayVC.videoUrl = subJson["mv\_link"].stringValue

self.navigationController?.pushViewController(videoPlayVC, animated: true)

}

}else if tableView == self.rightTableView{

if self.knowledgeArray.count > indexPath.row{

let subJson = self.knowledgeArray[indexPath.row]

let detailVC = KnowledgeDetailViewController.spwan()

detailVC.knowledgeId = subJson["post\_id"].stringValue

detailVC.dataChangeBlock = {[weak self] (result : JSON) in

self?.knowledgeArray.remove(at: indexPath.row)

self?.knowledgeArray.insert(result, at: indexPath.row)

self?.rightTableView.reloadRows(at: [indexPath], with: .automatic)

}

self.navigationController?.pushViewController(detailVC, animated: true)

}

}

}

func scrollViewDidScroll(\_ scrollView: UIScrollView) {

if scrollView == self.scrollView{

if scrollView.contentOffset.x == 0{

self.topBtnAction(self.courseBtn)

}else if scrollView.contentOffset.x == scrollView.w{

self.topBtnAction(self.videoBtn)

}else if scrollView.contentOffset.x == scrollView.w \* 2{

self.topBtnAction(self.knowledgeBtn)

}

}else if scrollView as? UITableView == self.rightTableView{

self.showSortView(show: false)

self.searchBar.resignFirstResponder()

}else if scrollView as? UITableView == self.middleTableView{

self.searchBar.resignFirstResponder()

}

}

}

extension DiscoverViewController : UICollectionViewDelegate,UICollectionViewDataSource,UICollectionViewDelegateFlowLayout{

func numberOfSections(in collectionView: UICollectionView) -> Int {

self.sortViewH.constant = 200

return 2

}

func collectionView(\_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {

if section == 0{

return self.knowledgeSortArray.count

}else{

if self.knowledgeSortArray.count > self.selectedSecOneIndex{

let smallList = self.knowledgeSortArray[self.selectedSecOneIndex]["smallList"].arrayValue

return smallList.count

}

}

return 0

}

func collectionView(\_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

let item = collectionView.dequeueReusableCell(withReuseIdentifier: "SortCollectionCell", for: indexPath)

for view in item.subviews{

view.removeFromSuperview()

}

var str = ""

if indexPath.section == 0{

if self.knowledgeSortArray.count > indexPath.row{

str = self.knowledgeSortArray[indexPath.row]["name"].stringValue

}

}else{

if self.knowledgeSortArray.count > self.selectedSecOneIndex{

let smallList = self.knowledgeSortArray[self.selectedSecOneIndex]["smallList"].arrayValue

if smallList.count > indexPath.row{

str = smallList[indexPath.row]["name"].stringValue

}

}

}

let lbl = UILabel(frame:item.bounds)

lbl.textAlignment = .center

lbl.font = UIFont.systemFont(ofSize: 14.0)

lbl.textColor = UIColor.RGBS(s: 33)

lbl.text = str

item.addSubview(lbl)

if indexPath.row == 0{

if indexPath.section == 0{

self.secOneLbl.frame = CGRect.init(x: 0, y: item.y, width: self.sortLeftView.w, height: 25)

}else{

self.secTwoLbl.frame = CGRect.init(x: 0, y: item.y, width: self.sortLeftView.w, height: 25)

}

}

if indexPath.section == 1{

let smallList = self.knowledgeSortArray[self.selectedSecOneIndex]["smallList"].arrayValue

if indexPath.row == smallList.count - 1{

self.sortViewH.constant = item.frame.maxY + 30

}

}

return item

}

func collectionView(\_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {

collectionView.deselectItem(at: indexPath, animated: false)

if indexPath.section == 0{

if self.knowledgeSortArray.count > indexPath.row{

self.typeId = self.knowledgeSortArray[indexPath.row]["id"].stringValue

self.selectedSecOneIndex = indexPath.row

collectionView.reloadData()

}

}else{

if self.knowledgeSortArray.count > self.selectedSecOneIndex{

let smallList = self.knowledgeSortArray[self.selectedSecOneIndex].arrayValue

if smallList.count > indexPath.row{

self.typeId = smallList[indexPath.row]["id"].stringValue

}

self.showSortView(show: false)

}

}

self.loadKnowledgeData()

}

func collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {

return CGSize.init(width: 70, height: 25)

}

func collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int) -> CGFloat {

return 1

}

func collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int) -> CGFloat {

return 2

}

func collectionView(\_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, referenceSizeForHeaderInSection section: Int) -> CGSize {

return CGSize.init(width: collectionView.w, height: 15)

}

}

extension DiscoverViewController : UISearchBarDelegate{

func setUpRightItem(show : Bool) {

if show{

self.navigationItem.rightBarButtonItem = UIBarButtonItem.init(image: imageLiteral(resourceName: "icon\_search"), target: self, action: #selector(DiscoverViewController.setUpSearchNavView))

}else{

self.navigationItem.rightBarButtonItem = nil

}

}

func cancelAction() {

self.navigationItem.titleView = nil

self.navigationItem.rightBarButtonItems = nil

self.navigationItem.rightBarButtonItem = UIBarButtonItem.init(image: imageLiteral(resourceName: "icon\_search"), target: self, action: #selector(DiscoverViewController.setUpSearchNavView))

self.navigationItem.title = "发现"

if self.searchBar.isFirstResponder{

self.searchBar.resignFirstResponder()

}

searchBar.text = ""

self.keyWord = ""

self.curpage = 1

if self.videoBtn.isSelected{

self.loadVideoData()

}else if self.knowledgeBtn.isSelected{

self.loadKnowledgeData()

}

}

@objc func setUpSearchNavView() {

searchBar.placeholder = "请输入关键字搜索"

searchBar.searchBarStyle = .minimal

searchBar.delegate = self

searchBar.showsCancelButton = true

searchBar.becomeFirstResponder()

self.navigationItem.titleView = searchBar

self.navigationItem.rightBarButtonItem = nil

}

func searchBarSearchButtonClicked(\_ searchBar: UISearchBar) {

if !(searchBar.text?.isEmpty)!{

self.keyWord = searchBar.text!

self.curpage = 1

if self.videoBtn.isSelected{

self.loadVideoData()

}else if self.knowledgeBtn.isSelected{

self.loadKnowledgeData()

}

}

}

func searchBar(\_ searchBar: UISearchBar, textDidChange searchText: String) {

print(searchText)

}

func searchBarCancelButtonClicked(\_ searchBar: UISearchBar) {

self.cancelAction()

}

}

import UIKit

import SwiftyJSON

class PersonalViewController: BaseTableViewController {

class func spwan() -> PersonalViewController{

return self.loadFromStoryBoard(storyBoard: "Personal") as! PersonalViewController

}

@IBOutlet weak var navViewH: NSLayoutConstraint!

@IBOutlet weak var iconImgV : UIImageView!

@IBOutlet weak var nameLbl : UILabel!

@IBOutlet weak var aouthBtn : UIButton!

@IBOutlet weak var aouthLbl: UILabel!

@IBOutlet weak var invoteLbl : UILabel!

@IBOutlet weak var signLbl : UILabel!

@IBOutlet weak var creditsLbl : UILabel!

@IBOutlet weak var couponLbl : UILabel!

@IBOutlet weak var walletLbl : UILabel!

@IBOutlet weak var billLbl : UILabel!

@IBOutlet weak var orderLbl : UILabel!

@IBOutlet weak var creditsView : UIView!

@IBOutlet weak var signView : UIView!

@IBOutlet weak var couponView : UIView!

@IBOutlet weak var messageNumLbl: UILabel!

@IBOutlet weak var levelView: UIView!

@IBOutlet weak var levelImgV1: UIImageView!

@IBOutlet weak var levelImgV2: UIImageView!

@IBOutlet weak var levelImgV3: UIImageView!

@IBOutlet weak var levelLbl: UILabel!

@IBOutlet weak var levelLblLeftDis: NSLayoutConstraint!

fileprivate var personalInfo : JSON = []

override func viewDidLoad() {

super.viewDidLoad()

self.iconImgV.layer.cornerRadius = 35

self.messageNumLbl.layer.cornerRadius = 9;

self.iconImgV.addTapActionBlock {

self.imgvClickAction()

}

self.signView.addTapActionBlock {

let signVC = SignInViewController.spwan()

self.navigationController?.pushViewController(signVC, animated: true)

}

self.creditsView.addTapActionBlock {

let creditsVC = CreditsViewController.spwan()

self.navigationController?.pushViewController(creditsVC, animated: true)

}

self.couponView.addTapActionBlock {

let couponVC = CouponViewController()

self.navigationController?.pushViewController(couponVC, animated: true)

}

self.levelView.addTapActionBlock {

let levelInfoVC = LevelInfoViewController.spwan()

self.navigationController?.pushViewController(levelInfoVC, animated: true)

}

self.scrollViewDidScroll(self.tableView)

if iphoneType() == "iPhone X"{

self.navViewH.constant = 88

}else{

self.navViewH.constant = 64

}

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

self.navigationItem.title = ""

self.navigationController?.navigationBar.isTranslucent = true

self.navigationController?.navigationBar.setBackgroundImage(UIImage(), for: .default)

self.navigationController?.navigationBar.shadowImage = UIImage()

self.edgesForExtendedLayout = UIRectEdge.top

}

override func viewDidAppear(\_ animated: Bool) {

super.viewDidAppear(animated)

self.navigationController?.setNavigationBarHidden(true, animated: false)

UIApplication.shared.statusBarStyle = .lightContent

if !LocalData.getUserId().isEmpty{

self.loadMineInfoData()

self.loadMessageNum()

}

}

override func viewWillDisappear(\_ animated: Bool) {

super.viewWillDisappear(animated)

self.navigationController?.setNavigationBarHidden(false, animated: false)

UIApplication.shared.statusBarStyle = .default

self.edgesForExtendedLayout = []

self.navigationController?.navigationBar.isTranslucent = false

self.navigationController?.navigationBar.setBackgroundImage(nil, for: .default)

self.navigationController?.navigationBar.shadowImage = nil

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

}

func loadMineInfoData() {

NetTools.requestData(type: .post, urlString: PersonalInfoApi, succeed: { (resultJson, msg) in

self.personalInfo = resultJson

if resultJson["is\_real"].stringValue == "1"{

LocalData.saveYesOrNotValue(value: "1", key: IsTrueName)

self.addRedits(type: "2")

self.aouthBtn.setTitle("已实名认证", for: .normal)

self.aouthLbl.text = " (已实名认证)"

}else{

LocalData.saveYesOrNotValue(value: "0", key: IsTrueName)

self.aouthBtn.setTitle("请实名认证", for: .normal)

self.aouthLbl.text = " (请实名认证)"

}

LocalData.saveUserName(userName: resultJson["member\_nik\_name"].stringValue)

LocalData.saveTrueUserName(userName: resultJson["member\_truename"].stringValue)

if resultJson["member\_level"].stringValue == "A" || resultJson["member\_level"].stringValue == "DA"{

LocalData.saveYesOrNotValue(value: "1", key: IsALevelUser)

}else{

LocalData.saveYesOrNotValue(value: "0", key: IsALevelUser)

}

LocalData.saveUserInviteCode(phone: resultJson["iv\_code"].stringValue)

if resultJson["count\_bill"].stringValue == "1" && resultJson["count\_bill\_integral"].stringValue == "0"{

self.addRedits(type: "3")

}

self.iconImgV.setHeadImageUrlStr(resultJson["member\_avatar"].stringValue)

if resultJson["member\_nik\_name"].stringValue.isEmpty{

self.nameLbl.text = "请设置昵称"

}else{

self.nameLbl.text = resultJson["member\_nik\_name"].stringValue

}

if resultJson["member\_level"].stringValue == "DA"{

self.invoteLbl.text = ""

}else{

self.invoteLbl.text = "邀请码:" + resultJson["iv\_code"].stringValue

}

self.signLbl.text = resultJson["sign\_day"].stringValue

self.creditsLbl.text = resultJson["jifen"].stringValue

self.couponLbl.text = resultJson["ticket"].stringValue

self.walletLbl.text = resultJson["balance"].stringValue + "元"

self.billLbl.text = resultJson["send\_count\_bill"].stringValue + "单"

self.orderLbl.text = resultJson["take\_count\_bill"].stringValue + "单"

let level = resultJson["level"].stringValue

UIImageView.setLevelImageView(imgV1: self.levelImgV1, imgV2: self.levelImgV2, imgV3: self.levelImgV3, level: level)

self.levelLbl.text = level + "级"

if level.intValue % 3 == 0{

self.levelLblLeftDis.constant = CGFloat( 5 + 3 \* 18)

}else{

self.levelLblLeftDis.constant = CGFloat( 5 + (level.intValue % 3) \* 18)

}

}) { (error) in

}

}

func addRedits(type:String) {

var params : [String : Any] = [:]

params["type"] = type

NetTools.requestData(type: .post, urlString: AddReditsApi, parameters: params, succeed: { (json, msg) in

}) { (error) in

}

}

func imgvClickAction() {

if self.iconImgV.image != nil{

let photoBrowseVC = LYPhotoBrowseViewController()

photoBrowseVC.imgArray = [self.iconImgV.image!]

photoBrowseVC.showDeleteBtn = false

UIApplication.shared.keyWindow?.addSubview(photoBrowseVC.view)

photoBrowseVC.imgSingleTapBlock = {() in

UIView.animate(withDuration: 0.25, animations: {

photoBrowseVC.view.layer.transform = CATransform3DMakeScale(0.1, 0.1, 0.1)

}, completion: { (completion) in

photoBrowseVC.view.removeFromSuperview()

})

}

}

}

@IBAction func personalInfoAction() {

let personalInfoVC = PersonalInfoTableViewController.spwan()

self.navigationController?.pushViewController(personalInfoVC, animated: true)

}

@IBAction func settingAction() {

let messageVC = MessageViewController()

self.navigationController?.pushViewController(messageVC, animated: true)

}

@IBAction func qrCodeAction() {

let urlStr = self.personalInfo["invite\_url"].stringValue

let qrcodeView = CreateQrcodeView.init(frame: nil, urlStr: urlStr, image: self.iconImgV.image)

qrcodeView.show()

qrcodeView.shareBlock = {() in

ShareView.show(url: urlStr, title: "加入七小服", desc: "注册成为七小服用户", viewController: self)

}

}

func loadMessageNum() {

let params : [String : Any] = [

"op" : "message\_list",

"act" : "member\_index",

"store\_id" : "1"

]

NetTools.requestData(type: .post, urlString: SysTermMessageApi, parameters: params, succeed: { (result, msg) in

for sub in result.arrayValue[0]["countnum\_all\_count"].arrayValue {

if sub["msg\_type"].stringValue.intValue == 0{

if sub["countnum"].stringValue.intValue > 0{

self.messageNumLbl.text = sub["countnum"].stringValue

self.messageNumLbl.isHidden = false

}else{

self.messageNumLbl.isHidden = true

}

}

}

}) { (error) in

}

}

}

extension PersonalViewController{

override func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

if indexPath.row == 0{

if iphoneType() == "iPhone X"{

return 252

}

return 230

}

return 44

}

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

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

switch (indexPath.row) {

case 1:

let walletVC = WalletViewController.spwan()

self.navigationController?.pushViewController(walletVC, animated: true)

case 2:

let shopCarVC = ShopCarListViewController.spwan()

self.navigationController?.pushViewController(shopCarVC, animated: true)

case 3:

let collectVC = CollectListViewController()

collectVC.isCollect = true

self.navigationController?.pushViewController(collectVC, animated: true)

case 4:

let mySendVC = MySendOrderListViewController.spwan()

mySendVC.titleArray = ["报名中","已接单","已完成","已取消","已失效"]

mySendVC.stateArray = [1,2,3,5,4]

self.navigationController?.pushViewController(mySendVC, animated: true)

case 5:

let myReceiveVC = MySendOrderListViewController.spwan()

myReceiveVC.titleArray = ["报名中","已接单","已完成","已取消","调价中","补单"]

myReceiveVC.stateArray = [1,2,3,5,6,7]

myReceiveVC.isMyReceive = true

self.navigationController?.pushViewController(myReceiveVC, animated: true)

case 6:

let freeTimeVC = FreeTimeListViewController()

self.navigationController?.pushViewController(freeTimeVC, animated: true)

case 7:

let shopOrderVC = ShopOrderListViewController.spwan()

self.navigationController?.pushViewController(shopOrderVC, animated: true)

case 8:

let pluginVC = PluginHistoryViewController()

self.navigationController?.pushViewController(pluginVC, animated: true)

case 9:

let inventoryVC = InventoryListViewController()

self.navigationController?.pushViewController(inventoryVC, animated: true)

case 10:

if !HChatClient.shared().isLoggedInBefore{

HChatClient.shared().login(withUsername: LocalData.getUserPhone(), password: "11")

}

let chatVC = HDChatViewController.init(conversationChatter: "kefu1")

self.navigationController?.pushViewController(chatVC!, animated: true)

case 11:

let associationVC = AssociationViewController()

self.navigationController?.pushViewController(associationVC, animated: true)

case 12:

let settingVC = SettingTableViewController()

if self.personalInfo["is\_paypwd"].stringValue.intValue == 1{

settingVC.isSetPayPwd = true

}

if self.personalInfo["is\_real"].stringValue.intValue == 1{

settingVC.isReal = true

}

self.navigationController?.pushViewController(settingVC, animated: true)

default:

break;

}

}

}

import UIKit

import SwiftyJSON

import AVFoundation

import Photos

class PersonalInfoTableViewController: BaseTableViewController {

class func spwan() -> PersonalInfoTableViewController{

return self.loadFromStoryBoard(storyBoard: "Personal") as! PersonalInfoTableViewController

}

var personalInfo : JSON = []

@IBOutlet weak var iconImgV: UIImageView!

@IBOutlet weak var nickNameTF: UITextField!

@IBOutlet weak var realNameLbl: UILabel!

@IBOutlet weak var realNameArrowImgV: UIImageView!

@IBOutlet weak var workYearLbl: UILabel!

@IBOutlet weak var certView: UIView!

@IBOutlet weak var techRangeLbl: UILabel!

@IBOutlet weak var adaptBrandTF: UITextField!

@IBOutlet weak var invoteCodeLbl: UILabel!

@IBOutlet weak var levelCodeLbl: UILabel!

@IBOutlet weak var levelImgV1: UIImageView!

@IBOutlet weak var levelImgV2: UIImageView!

@IBOutlet weak var levelImgV3: UIImageView!

fileprivate var photoView = LYPhotoBrowseView()

fileprivate var serverRangeJson : JSON = []

fileprivate var workYear : Date?//从业年限

fileprivate var nickName : String = ""//昵称

fileprivate var techRangeArray : Array<String> = Array<String>()//技术领域

fileprivate var adaptBrand : String = ""//擅长品牌

fileprivate var cerImgs : String = ""//职业证书

override func viewDidLoad() {

super.viewDidLoad()

self.navigationItem.title = "个人信息"

self.iconImgV.layer.cornerRadius = 20

self.navigationItem.rightBarButtonItem = UIBarButtonItem.init(image: imageLiteral(resourceName: "reminder\_icon"), target: self, action: #selector(PersonalInfoTableViewController.rightItemAction))

self.prepareData()

}

override func viewWillAppear(\_ animated: Bool) {

super.viewWillAppear(animated)

self.loadMineInfoData()

}

func loadMineInfoData() {

NetTools.requestData(type: .post, urlString: PersonalInfoApi, succeed: { (resultJson, msg) in

self.personalInfo = resultJson

self.setUpUIData()

self.tableView.reloadRows(at: [IndexPath.init(row: 8, section: 0)], with: .none)

}) { (error) in

}

}

override func viewWillDisappear(\_ animated: Bool) {

super.viewWillDisappear(animated)

self.view.endEditing(true)

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

@objc func rightItemAction() {

let remindView = PersonalRemindView.init(frame: CGRect.zero)

remindView.show()

}

func prepareData() {

NetTools.requestData(type: .post, urlString: ServerRangeListApi, succeed: { (result, msg) in

self.serverRangeJson = result

}) { (error) in

}

}

func setUpUIData() {

self.iconImgV.setHeadImageUrlStr(self.personalInfo["member\_avatar"].stringValue)

if !self.personalInfo["member\_nik\_name"].stringValue.isEmpty{

self.nickNameTF.text = self.personalInfo["member\_nik\_name"].stringValue

}

if self.personalInfo["is\_real"].stringValue.intValue == 1{

self.realNameLbl.text = "已认证"

self.realNameArrowImgV.isHidden = true

}

if !self.personalInfo["working\_time"].stringValue.isEmpty{

self.workYearLbl.text = Date.dateStringFromDate(format: Date.yearFormatString(), timeStamps: self.personalInfo["working\_time"].stringValue)

}

self.photoView = LYPhotoBrowseView.init(frame: CGRect.init(x: 0, y: 0, width: kScreenW - 16, height: 50),superVC:self)

self.photoView.backgroundColor = UIColor.white

self.photoView.showLogoImgV = true

self.photoView.maxPhotoNum = 5

self.photoView.canTakePhoto = true

self.photoView.showDeleteBtn = false

self.photoView.customBlock = {() in

let addCertVC = AddCertificateViewController.spwan()

addCertVC.depth = "\(self.personalInfo["cer\_images"].arrayValue.count + 1)"

self.navigationController?.pushViewController(addCertVC, animated: true)

}

self.certView.addSubview(self.photoView)

if self.personalInfo["cer\_images"].arrayValue.count > 0{

var imgUrlArray : Array<String> = Array<String>()

var imgDescArray : Array<String> = Array<String>()

var imgLogoArray : Array<String> = Array<String>()

for subJson in self.personalInfo["cer\_images"].arrayValue {

imgUrlArray.append(subJson["cer\_image"].stringValue)

imgDescArray.append(subJson["cer\_image\_name"].stringValue)

imgLogoArray.append(subJson["cer\_image\_type"].stringValue)

}

self.photoView.showImgUrlArray = imgUrlArray

self.photoView.imgDescArray = imgDescArray

self.photoView.imgLogoArray = imgLogoArray

self.photoView.longPressBlock = {(index) in

let addCertVC = AddCertificateViewController.spwan()

addCertVC.certImg = self.photoView.imgArray[index]

addCertVC.certName = imgDescArray[index]

addCertVC.imgUrl = imgUrlArray[index]

addCertVC.depth = "\(index + 1)"

addCertVC.certId = self.personalInfo["cer\_images"].arrayValue[index]["cer\_id"].stringValue

self.navigationController?.pushViewController(addCertVC, animated: true)

}

}

if self.personalInfo["service\_sector"].arrayValue.count > 0{

var sectorArray : Array<String> = Array<String>()

for subJson in self.personalInfo["service\_sector"].arrayValue {

sectorArray.append(subJson["gc\_name"].stringValue)

}

self.techRangeLbl.text = sectorArray.joined(separator: ",")

}

if !self.personalInfo["service\_brand"].stringValue.isEmpty{

self.adaptBrandTF.text = self.personalInfo["service\_brand"].stringValue

}

if !self.personalInfo["iv\_code"].stringValue.isEmpty{

self.invoteCodeLbl.text = self.personalInfo["iv\_code"].stringValue

}

self.levelCodeLbl.text = self.personalInfo["levelscore"].stringValue + "分"

UIImageView.setLevelImageView(imgV1: self.levelImgV3, imgV2: self.levelImgV2, imgV3: self.levelImgV1, level: self.personalInfo["level"].stringValue)

}

func changePersonalInfo() {

self.view.endEditing(true)

var params : [String : Any] = [:]

if self.workYear != nil{

params["working\_time"] = self.workYear?.phpTimestamp()

}

if !self.nickName.isEmpty{

params["nik\_name"] = self.nickName

}

if !self.techRangeArray.isEmpty{

params["service\_sector"] = self.techRangeArray.joined(separator: ",")

}

if !self.adaptBrand.isEmpty{

params["service\_brand"] = self.adaptBrand

}

if !self.cerImgs.isEmpty{

params["cer\_images"] = self.cerImgs

}

NetTools.requestData(type: .post, urlString: ChangePersonalInfoApi, parameters: params, succeed: { (result, msg) in

LYProgressHUD.showSuccess("保存成功！")

}) { (error) in

LYProgressHUD.showError(error!)

}

}

}

extension PersonalInfoTableViewController : UITextFieldDelegate{

func textFieldShouldReturn(\_ textField: UITextField) -> Bool {

if textField == self.nickNameTF{

self.nickName = textField.text!

if !self.nickName.isEmpty{

self.changePersonalInfo()

}

}else if textField == self.adaptBrandTF{

self.adaptBrand = textField.text!

if !self.adaptBrand.isEmpty{

self.changePersonalInfo()

}

}

return true

}

override func scrollViewDidScroll(\_ scrollView: UIScrollView) {

self.view.endEditing(true)

}

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

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

switch indexPath.row {

case 0:

self.addPhotoAction()

case 2:

let levelInfoVC = LevelInfoViewController.spwan()

self.navigationController?.pushViewController(levelInfoVC, animated: true)

case 3:

if !self.realNameArrowImgV.isHidden{

let idVC = IdentityViewController.spwan()

self.navigationController?.pushViewController(idVC, animated: true)

}

case 4:

let datePicker = LYDatePicker.init(component: 1)

datePicker.ly\_datepickerWithOneComponent = {(date,year) in

self.workYear = date

self.workYearLbl.text = "\(year)年"

self.changePersonalInfo()

}

datePicker.show()

case 6:

let serverRangeVC = ServerRangeViewController.spwan()

serverRangeVC.serverRangeBlock = {(selectedDictArray,titles,ids) in

self.techRangeLbl.text = titles.joined(separator: ";")

self.techRangeArray = ids

self.changePersonalInfo()

}

serverRangeVC.dataArray = self.serverRangeJson.arrayValue

self.navigationController?.pushViewController(serverRangeVC, animated: true)

case 9:

let commentVC = CommentListViewController()

commentVC.member\_id = self.personalInfo["member\_id"].stringValue

commentVC.isFromPersonalInfo = true

self.navigationController?.pushViewController(commentVC, animated: true)

default:

print(indexPath.row)

}

}

override func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

if indexPath.row == 0{

return 71

}else if indexPath.row == 5{

return 88

}else if indexPath.row == 8{

if self.personalInfo["member\_level"].stringValue == "DA"{

return 0

}else{

return 44

}

}

return 44;

}

}

extension PersonalInfoTableViewController : UIImagePickerControllerDelegate,UINavigationControllerDelegate,UIActionSheetDelegate {

func addPhotoAction() {

let sheet = UIActionSheet.init(title: "添加图片", delegate: self, cancelButtonTitle: "cancel", destructiveButtonTitle: nil, otherButtonTitles: "相册", "拍照")

sheet.show(in: self.view)

}

func actionSheet(\_ actionSheet: UIActionSheet, clickedButtonAt buttonIndex: Int) {

}

func actionSheet(\_ actionSheet: UIActionSheet, didDismissWithButtonIndex buttonIndex: Int) {

if buttonIndex == 1{

self.photoAlbum()

}else if buttonIndex == 2{

self.camera()

}

}

func camera() {

switch AVCaptureDevice.authorizationStatus(for: AVMediaType.video) {

case .restricted,.denied:

LYAlertView.show("提示", "请允许App使用相机权限", "取消", "去设置", {

let url = URL(string:UIApplicationOpenSettingsURLString)

if UIApplication.shared.canOpenURL(url!){

UIApplication.shared.openURL(url!)

}

})

return

case .authorized,.notDetermined:

break

}

if !UIImagePickerController.isSourceTypeAvailable(.camera){

LYProgressHUD.showError("此设备无拍照功能!!!")

return

}

if !UIImagePickerController.isCameraDeviceAvailable(UIImagePickerControllerCameraDevice.rear) && !UIImagePickerController.isCameraDeviceAvailable(UIImagePickerControllerCameraDevice.front){

LYProgressHUD.showError("相机不可用!!!")

return

}

let picker : UIImagePickerController = UIImagePickerController()

picker.sourceType = .camera

picker.delegate = self

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

}

func photoAlbum() {

switch PHPhotoLibrary.authorizationStatus() {

case .restricted,.denied:

LYAlertView.show("提示", "请允许App访问相册", "取消", "去设置", {

let url = URL(string:UIApplicationOpenSettingsURLString)

if UIApplication.shared.canOpenURL(url!){

UIApplication.shared.openURL(url!)

}

})

return

case .authorized,.notDetermined:

break

}

if UIImagePickerController.isSourceTypeAvailable(.photoLibrary) {

let picker : UIImagePickerController = UIImagePickerController()

picker.sourceType = .photoLibrary

picker.delegate = self

picker.navigationBar.tintColor = UIColor.RGBS(s: 33)

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

}else{

LYProgressHUD.showError("不允许访问相册")

}

}

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

picker.dismiss(animated: true, completion: nil)

let img = info[UIImagePickerControllerOriginalImage] as! UIImage

var params : [String : String] = [:]

params["store\_id"] = "1"

LYProgressHUD.showLoading()

NetTools.upLoadCustomImage(urlString: ChangePersonalIconApi, imgArray: [img], parameters: params, success: { (result) in

self.iconImgV.image = img

LYProgressHUD.dismiss()

}) { (error) in

LYProgressHUD.showError("请重新选择！")

}

}

func imagePickerControllerDidCancel(\_ picker: UIImagePickerController) {

picker.dismiss(animated: true, completion: nil)

}

}

import UIKit

import SwiftyJSON

class MySendOrderListViewController: BaseViewController {

class func spwan() -> MySendOrderListViewController{

return self.loadFromStoryBoard(storyBoard: "Personal") as! MySendOrderListViewController

}

var isMyReceive = false

@IBOutlet weak var topScrollView: UIScrollView!

@IBOutlet weak var scrollContentView: UIView!

@IBOutlet weak var scrollContentVW: NSLayoutConstraint!

@IBOutlet weak var tableView: UITableView!

var titleArray : Array<String> = Array<String>()

var stateArray : Array<Int> = Array<Int>()

fileprivate var btnView = UIView()

fileprivate var line = UIView()

fileprivate var dataArray : Array<JSON> = Array<JSON>()

fileprivate var curpage = 1

fileprivate var billState = 1//发单状态【0 撤销】【1 待接单】【2 已接单】【3 已完成】【4 已过期 or 已失效】【5 已取消】【6 调价中】【7 补单】

override func viewDidLoad() {

super.viewDidLoad()

self.setUpTopNav()

self.tableView.register(UINib.init(nibName: "MySendOrderCell", bundle: Bundle.main), forCellReuseIdentifier: "MySendOrderCell")

self.loadData()

self.addRefresh()

}

func setUpTopNav() {

if self.isMyReceive{

self.navigationItem.title = "我的接单"

self.navigationItem.rightBarButtonItem = UIBarButtonItem.init(title: "补单", target: self, action: #selector(MySendOrderListViewController.rightItemAction))

}else{

self.navigationItem.title = "我的发单"

self.navigationItem.rightBarButtonItem = UIBarButtonItem.init(title: "发单", target: self, action: #selector(MySendOrderListViewController.rightItemAction))

}

let merge = 10

let btnW = 80

self.scrollContentVW.constant = CGFloat((btnW + merge) \* self.titleArray.count + merge)

self.btnView = UIView.init(frame: CGRect.init(x: 0, y: 0, width: self.scrollContentVW.constant, height: 42))

self.scrollContentView.addSubview(btnView)

for (i,str) in self.titleArray.enumerated() {

let btn = UIButton(frame:CGRect.init(x: merge + (btnW + merge) \* i, y: 0, width: btnW, height: 42))

btn.setTitle(str, for: .normal)

btn.setTitleColor(Normal\_Color, for: .selected)

btn.setTitleColor(Text\_Color, for: .normal)

btn.titleLabel?.font = UIFont.systemFont(ofSize: 14.0)

btn.addTarget(self, action: #selector(MySendOrderListViewController.topBtnAction(btn:)), for: .touchUpInside)

btn.tag = i

if i == 0{

btn.isSelected = true

}

self.btnView.addSubview(btn)

}

self.line = UIView.init(frame: CGRect.init(x: CGFloat(merge), y: 40, width: CGFloat(btnW), height: 1.5))

self.line.backgroundColor = Normal\_Color

self.scrollContentView?.addSubview(self.line)

}

@objc func topBtnAction(btn : UIButton) {

for view in self.btnView.subviews {

let btn1 = view as! UIButton

btn1.isSelected = false

}

btn.isSelected = true

self.line.x = btn.x

self.billState = self.stateArray[btn.tag]

self.curpage = 1

self.loadData()

if btn.x > kScreenW / 2.0{

if (self.scrollContentVW.constant - btn.x - btn.w / 2.0) > kScreenW / 2.0{

self.topScrollView.contentOffset = CGPoint.init(x: btn.x - kScreenW / 2.0 + btn.w / 2.0 , y: 0)

}else{

self.topScrollView.contentOffset = CGPoint.init(x: self.scrollContentVW.constant - kScreenW, y: 0)

}

}

if (btn.x + btn.w) - self.topScrollView.contentOffset.x < kScreenW / 2.0{

if kScreenW / 2.0 - btn.x - btn.w / 2.0 < 0 {

self.topScrollView.contentOffset = CGPoint.init(x: btn.x + btn.w / 2.0 - kScreenW / 2.0 , y: 0)

}else{

self.topScrollView.contentOffset = CGPoint.zero

}

}

}

@objc func rightItemAction() {

if self.isMyReceive{

if LocalData.getYesOrNotValue(key: IsALevelUser){

LYProgressHUD.showInfo("当前用户为A用户，不可补单！")

return

}

let redoOrderVC = SendTaskViewController.spwan()

redoOrderVC.isRepairOrder = true

self.navigationController?.pushViewController(redoOrderVC, animated: true)

}else{

let sendTaskVC = SendTaskViewController.spwan()

self.navigationController?.pushViewController(sendTaskVC, animated: true)

}

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

func loadData() {

var params : [String : Any] = [:]

params["curpage"] = self.curpage

var url = ""

if self.isMyReceive{

if self.billState == 1{

url = MyReceiveEnrollListApi

}else{

url = MyReceiveOrderListApi

params["bill\_statu"] = (self.billState)

}

}else{

if self.billState == 1{

url = MySendEnrollListApi

}else{

url = MySendOrderListApi

params["bill\_statu"] = (self.billState)

}

}

NetTools.requestData(type: .post, urlString: url, parameters: params, succeed: { (result, msg) in

self.curpage == 1 ? self.tableView.es.stopPullToRefresh() : self.tableView.es.stopLoadingMore()

if result.arrayValue.count < 10{

self.tableView.es.noticeNoMoreData()

}else{

self.tableView.es.resetNoMoreData()

}

if self.curpage == 1{

self.dataArray.removeAll()

}

for subJson in result.arrayValue{

self.dataArray.append(subJson)

}

if self.dataArray.count == 0{

self.emptyView.frame = self.tableView.frame

self.showEmptyView()

}else{

self.hideEmptyView()

}

self.tableView.reloadData()

}) { (error) in

self.curpage == 1 ? self.tableView.es.stopPullToRefresh() : self.tableView.es.stopLoadingMore()

LYProgressHUD.showError(error!)

}

}

func addRefresh() {

self.tableView.es.addPullToRefresh {

[weak self] in

self?.curpage = 1

self?.loadData()

}

self.tableView.es.addInfiniteScrolling {

[weak self] in

self?.curpage += 1

self?.loadData()

}

}

}

extension MySendOrderListViewController : UITableViewDelegate,UITableViewDataSource{

func numberOfSections(in tableView: UITableView) -> Int {

return 1

}

func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

return self.dataArray.count

}

func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

let cell = tableView.dequeueReusableCell(withIdentifier: "MySendOrderCell", for: indexPath) as! MySendOrderCell

if self.dataArray.count > indexPath.row{

var subJson = self.dataArray[indexPath.row]

if self.isMyReceive{

subJson["isMyReceive"] = "1"

}

cell.subJson = subJson

cell.parentVC = self

cell.refreshBlock = {(type) in

if type == 1{

if self.dataArray.count > indexPath.row{

self.dataArray.remove(at: indexPath.row)

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

}

}else if type == 2{

self.changeDetailData(indexpath: indexPath, json: subJson)

}else{

if self.dataArray.count > indexPath.row{

self.dataArray.remove(at: indexPath.row)

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

}

}

}

}

return cell

}

func tableView(\_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {

return 190

}

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

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

if self.dataArray.count > indexPath.row{

let orderDetailVC = MySendOrderDetailViewController.spwan()

orderDetailVC.orderId = self.dataArray[indexPath.row]["id"].stringValue

orderDetailVC.isMyReceive = self.isMyReceive

if self.isMyReceive{

orderDetailVC.moveState = self.dataArray[indexPath.row]["move\_state"].stringValue

}else{

orderDetailVC.enrollNum = self.dataArray[indexPath.row]["num"].stringValue

}

orderDetailVC.refreshBlock = {(type) in

if type == 1{

if self.dataArray.count > indexPath.row{

self.dataArray.remove(at: indexPath.row)

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

}

}else if type == 2{

self.changeDetailData(indexpath: indexPath, json: self.dataArray[indexPath.row])

}else{

if self.dataArray.count > indexPath.row{

self.dataArray.remove(at: indexPath.row)

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

}

}

}

self.navigationController?.pushViewController(orderDetailVC, animated: true)

}

}

func changeDetailData(indexpath : IndexPath,json : JSON) {

if self.isMyReceive{

var params : [String : Any] = [:]

params["id"] = json["id"].stringValue

params["move\_state"] = json["move\_state"].stringValue

LYProgressHUD.showLoading()

NetTools.requestData(type: .post, urlString: ReceiveDetailDataApi, parameters: params, succeed: { (result, msg) in

LYProgressHUD.dismiss()

if self.dataArray.count > indexpath.row{

self.dataArray.remove(at: indexpath.row)

self.dataArray.insert(result, at: indexpath.row)

self.tableView.reloadData()

}

}) { (error) in

LYProgressHUD.showError(error!)

}

}else{

var params : [String : Any] = [:]

params["id"] = json["id"].stringValue

LYProgressHUD.showLoading()

NetTools.requestData(type: .post, urlString: MySendOrderDetailApi, parameters: params, succeed: { (result, msg) in

LYProgressHUD.dismiss()

if self.dataArray.count > indexpath.row{

self.dataArray.remove(at: indexpath.row)

self.dataArray.insert(result, at: indexpath.row)

self.tableView.reloadData()

}

}) { (error) in

LYProgressHUD.showError(error!)

}

}

}

}