**import UIKit**

**import Alamofire**

**import SwiftyStoreKit**

**@UIApplicationMain**

**class AppDelegate: UIResponder, UIApplicationDelegate {**

**var window: UIWindow?**

**let manager = NetworkReachabilityManager(host: "https://www.baidu.com")**

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

**// Override point for customization after application launch.**

**//       UserDefaults.standard.set(nil, forKey: "kSelectedProxyKey")**

**DatabaseManager.shared.setup()**

**//注册内购**

**SwiftyStoreKit.completeTransactions(atomically: true) { purchases in**

**for purchase in purchases {**

**switch purchase.transaction.transactionState {**

**case .purchased, .restored:**

**if purchase.needsFinishTransaction {**

**// Deliver content from server, then:**

**SwiftyStoreKit.finishTransaction(purchase.transaction)**

**}**

**case .failed, .purchasing, .deferred:**

**break // do nothing**

**default:**

**break;**

**}**

**}**

**}**

**manager!.listener = { status in**

**var statusStr = ""**

**switch status {**

**case .notReachable:**

**statusStr = "notReachable"**

**print("notReachable")**

**case .unknown:**

**statusStr = "unknown"**

**print("unknown")**

**case .reachable(.ethernetOrWiFi):**

**statusStr = "ethernetOrWiFi"**

**print("ethernetOrWiFi")**

**case .reachable(.wwan):**

**statusStr = "wwan"**

**print("wwan")**

**}**

**NotificationCenter.default.post(Notification.init(name: Notification.Name(rawValue: "netWorkingStatus"), object: nil, userInfo: ["netStatus" : statusStr]))**

**}**

**manager!.startListening()**

**return true**

**}**

**}**

**import UIKit**

**import SwiftyJSON**

**import SVProgressHUD**

**let count = 1**

**class MainTableViewController: UITableViewController,TermsPriDelegate,ServerDelegate,GCDAsyncSocketDelegate {**

**func termsClick() {**

**let vc = PureWebViewController()**

**vc.urlStr = TermsofServerUrl**

**vc.titleStr =  NSLocalizedString("terms\_of\_use", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**vc.block = {**

**weak var weakSelf = self**

**weakSelf?.termsPriView.frame = UIScreen.main.bounds**

**weakSelf?.termsPriView.delegate = weakSelf**

**UIApplication.shared.keyWindow?.addSubview(weakSelf?.termsPriView ?? TermsAndPrivacyView())**

**}**

**UIApplication.shared.keyWindow?.rootViewController?.present(vc, animated: true, completion: nil)**

**}**

**func privacyClick() {**

**let vc = PureWebViewController()**

**vc.urlStr = PrivacyPolicyUrl**

**vc.titleStr = NSLocalizedString("privacy\_policy", comment: "")**

**vc.block = {**

**weak var weakSelf = self**

**weakSelf?.termsPriView.frame = UIScreen.main.bounds**

**weakSelf?.termsPriView.delegate = weakSelf**

**UIApplication.shared.keyWindow?.addSubview(weakSelf?.termsPriView ?? TermsAndPrivacyView())**

**}**

**vc.modalPresentationStyle = .fullScreen**

**UIApplication.shared.keyWindow?.rootViewController?.present(vc, animated: true, completion: nil)**

**}**

**var rightBtn:UIButton?**

**var isAuto = true**

**var proxy: Proxy? {**

**didSet {**

**DispatchQueue.main.async {**

**self.currentLineLab.text = self.proxy?.netArea?.getENWithShort()**

**self.lineLab.text = !self.isAuto ? (self.proxy?.username) : NSLocalizedString("auto", comment: "")**

**self.currentLineImage.image = UIImage(named: self.proxy?.netArea?.uppercased() ?? "")**

**}**

**}**

**}**

**var currentNetWorking:String?{**

**didSet{**

**if currentNetWorking == "ethernetOrWiFi"  {**

**self.netStatusImage.image = UIImage(named: "icon\_home\_wifi")**

**}else if currentNetWorking == "wwan"{**

**self.netStatusImage.image = UIImage(named: "icon\_home\_gsm")**

**}else{**

**self.netStatusImage.image = UIImage(named: "")**

**}**

**}**

**}**

**@IBOutlet weak var statusImage: UIImageView!**

**@IBOutlet weak var statusTitleImage: UIImageView!**

**@IBOutlet weak var statusSubLab: UILabel!**

**@IBOutlet weak var currentLineImage: UIImageView!**

**@IBOutlet weak var currentLineLab: UILabel!**

**@IBOutlet weak var switchBtn: UISwitch!**

**@IBOutlet weak var upLab: UILabel!**

**@IBOutlet weak var downLab: UILabel!**

**@IBOutlet weak var netStatusImage: UIImageView!**

**@IBOutlet weak var lineLab: UILabel!**

**@IBOutlet var termsPriView: TermsAndPrivacyView!**

**@IBOutlet weak var termsTitleLab: UILabel!**

**@IBOutlet weak var bottomRestoreView: UIStackView!**

**var proxyArr:Array<Proxy> = []**

**var index = 0**

**var proxyIndex = 0**

**var bestProxy:Proxy?**

**var pingManager:NENPingManager = NENPingManager()**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**if !UserDefaults.standard.bool(forKey: "isfirstLoad"){**

**UserDefaults.standard.set(true, forKey: "isfirstLoad")**

**termsPriView.frame = UIScreen.main.bounds**

**termsPriView.delegate = self**

**UIApplication.shared.keyWindow?.addSubview(termsPriView)**

**}**

**self.termsTitleLab.text = NSLocalizedString("terandpp", comment: "")**

**if getCurrentLanguage() == "tc"{**

**self.statusTitleImage.image = UIImage(named: "VPNDisconnected\_tc")**

**}else{**

**self.statusTitleImage.image = UIImage(named: "VPNDisconnected!")**

**}**

**makeNav()**

**NotificationCenter.default.addObserver(forName: Notification.Name.CurrentProxyChangeNotification, object: nil, queue: OperationQueue.main) { [weak self] (notification) in**

**guard let `self` = self, let userInfo = notification.userInfo as? [String: Any] else {**

**return**

**}**

**let oldValue = userInfo[kProxyOldValueKey] as? Proxy**

**let newValue = userInfo[kProxyNewValueKey] as? Proxy**

**if oldValue?.rid != newValue?.rid {**

**self.updateConnectingVPN()**

**return**

**}**

**//            self.updateConnectingVPN()**

**}**

**//连接状态改变通知**

**NotificationCenter.default.addObserver(forName: NSNotification.Name.ProxyServiceStatusNotification, object: nil, queue: OperationQueue.main) { [weak self] (\_) in**

**self?.updateConnectStatus()**

**}**

**NotificationCenter.default.addObserver(forName: NSNotification.Name(rawValue: "netWorkingStatus"), object: nil, queue: OperationQueue.main) { [weak self] (notification) in**

**weak var weakSelf = self**

**let status  = notification.userInfo?["netStatus"] as? String ?? "unknown"**

**if status == "ethernetOrWiFi"  {**

**weakSelf?.netStatusImage.image = UIImage(named: "icon\_home\_wifi")**

**NotificationCenter.default.post(name: NSNotification.Name(rawValue: "getConect"), object: nil)**

**}else if status == "wwan"{**

**weakSelf?.netStatusImage.image = UIImage(named: "icon\_home\_gsm")**

**NotificationCenter.default.post(name: NSNotification.Name(rawValue: "getConect"), object: nil)**

**}else if status == "notReachable"{**

**//                VPNManager.shared.disconnect()**

**//                weakSelf?.netStatusImage.image = UIImage(named: "")**

**}else{**

**//                weakSelf?.netStatusImage.image = UIImage(named: "")**

**}**

**}**

**//        netStatus**

**getContentAndBest()**

**let imageView = self.currentLineImage**

**//       imageView?.image = UIImage(named: proxy?.netArea?.uppercased() ?? "")**

**let backgroundView = UIView(frame: imageView!.bounds)**

**backgroundView.backgroundColor = .white**

**imageView?.addSubview(backgroundView)**

**let maskLayer = CAShapeLayer()**

**maskLayer.fillRule = CAShapeLayerFillRule.evenOdd //  奇偶层显示规则**

**let basicPath = UIBezierPath(rect: backgroundView.frame) // 底层**

**let maskPath = UIBezierPath(ovalIn: CGRect(x: backgroundView.frame.width/2 - 16 , y: 0, width: 32, height: 32))**

**basicPath.append(maskPath) // 重叠**

**maskLayer.path = basicPath.cgPath**

**backgroundView.layer.mask = maskLayer**

**}**

**override func viewWillAppear(\_ animated: Bool) {**

**NetTool.share().start { dic in**

**self.upLab.text = dic["oBytes"] as? String**

**self.downLab.text = dic["iBytes"] as? String**

**}**

**}**

**override func viewWillDisappear(\_ animated: Bool) {**

**NetTool.share().fire()**

**}**

**@objc func getContentAndBest() {**

**getConnetData {**

**self.getBestProxy()**

**}**

**}**

**func getConnetData(complate:@escaping()->()){**

**SVProgressHUD.show(withStatus: NSLocalizedString("find\_best\_route", comment: ""))**

**NetworkingTool.getServerList(success: { response in**

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

**if response.code == 200{**

**self.proxyArr.removeAll()**

**let nonce = UserDefaults.standard.string(forKey: "serverNotice") ?? ""**

**let resStr = JSON(parseJSON: response.jsonStr)["reply"].stringValue.fromBase64()**

**let res = resStr?.replacingOccurrences(of: nonce, with: "")**

**let proxyJson = JSON(parseJSON: res ?? "").arrayValue**

**for proxy in proxyJson {**

**let dic = proxy.dictionaryObject**

**guard let idDic = dic?["\_id"] as? Dictionary<String,String> else {**

**return**

**}**

**var r = Proxy()**

**r.type = .shadowsocks**

**r.identifier = idDic["$oid"]**

**r.server = dic?["ip"] as? String ?? ""**

**//                    r.server = "47.106.245.173"**

**r.port = Int(dic?["port"] as? String ?? "8388") ?? 8388**

**r.password = dic?["password"] as? String ?? ""**

**r.encryption = .AES256CFB**

**r.mode = .all**

**r.netArea = (dic?["country"] as? String ?? "").uppercased()**

**r.username = dic?["name"] as? String ?? ""**

**self.proxyArr.append(r)**

**}**

**complate()**

**SVProgressHUD.dismiss()**

**}**

**}) { error in**

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

**NotificationCenter.default.addObserver(self, selector: #selector(self.getContentAndBest), name: NSNotification.Name(rawValue: "getConect"), object: nil)**

**SVProgressHUD.dismiss()**

**}**

**}**

**func getBestProxy(){**

**ProxyManager.shared.deleteAll()**

**let array = proxyArr.compactMap({return $0.server})**

**self.pingManager.getFatestAddress(array) { (host, address) in**

**self.bestProxy = self.proxyArr.first(where: {$0.server == host})**

**self.proxy = self.bestProxy**

**if ProxyManager.shared.save(proxy: self.proxy ?? Proxy()) {**

**//                SVProgressHUD.showSuccess(withStatus: "保存成功！")**

**} else {**

**//                SVProgressHUD.showError(withStatus: "保存失败！")**

**}**

**}**

**}**

**func makeNav() {**

**self.navigationController?.navigationBar.setBackgroundImage(UIImage.Create(size: CGSize(width: 1, height: 1), color: .white), for: UIBarMetrics.default)**

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

**let topBorder = CALayer()**

**let borderHeight: CGFloat = 0.33**

**topBorder.borderWidth = borderHeight**

**topBorder.borderColor = UIColor.colorWithHexColorString("000000", alpha: 0.1).cgColor**

**topBorder.frame = CGRect(x: 0, y: 43.67, width: LJScreenWidth, height: borderHeight)**

**self.navigationController?.navigationBar.layer.addSublayer(topBorder)**

**if true {**

**let rightBtn = UIButton(type: .custom)**

**rightBtn.frame = CGRect(x: 0, y: 0, width: 45, height: 35)**

**rightBtn.setTitle(NSLocalizedString("about", comment: ""), for: .normal)**

**rightBtn.titleLabel?.font = UIFont.pingFangSC\_Regular(size: 17)**

**rightBtn.setTitleColor(UIColor.colorWithHexColorString("C6C6C8"), for: .normal)**

**rightBtn.backgroundColor = UIColor.white**

**rightBtn.addTarget(self, action: #selector(goAbout), for: .touchUpInside)**

**self.rightBtn = rightBtn**

**let rightItem = UIBarButtonItem(customView: rightBtn)**

**self.navigationItem.rightBarButtonItems = [rightItem]**

**}**

**}**

**@objc func goAbout(){**

**let vc = AboutTableViewController.loadStoryboard(name: "Main")**

**self.navigationController?.pushViewController(vc, animated: true)**

**}**

**@IBAction func switchChange(\_ sender: UISwitch) {**

**if !PurchaseTool.sharedInstance.getCurVipStatus() {**

**sender.isOn = false**

**let vc = PurchTableViewController.loadStoryboard(name:"Main")**

**self.present(vc, animated: true, completion: nil)**

**}else{**

**starVPN()**

**}**

**}**

**// MARK: - Table view data source**

**@IBAction func restoreClick(\_ sender: UIButton) {**

**PurchaseTool.sharedInstance.restorePurchases { res in**

**if res{**

**}**

**}**

**}**

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

**// #warning Incomplete implementation, return the number of sections**

**return 1**

**}**

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

**// #warning Incomplete implementation, return the number of rows**

**return 4**

**}**

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

**return 0.001**

**}**

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

**return 0.001**

**}**

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

**if indexPath.row == 3 {**

**let vc = ServerLocationTableViewController.loadStoryboard(name: "Main")**

**vc.proxyList = self.proxyArr**

**vc.delegate = self**

**self.navigationController?.pushViewController(vc, animated: true)**

**}**

**}**

**func getServerModel(model: Proxy ,isAuto:Bool) {**

**self.isAuto = isAuto**

**if self.isAuto {**

**self.proxy = self.bestProxy**

**}else{**

**self.proxy = model**

**}**

**if ProxyManager.shared.save(proxy: self.proxy ?? Proxy()) {**

**ProxyManager.shared.currentProxy = self.proxy**

**reconnect()**

**} else {**

**}**

**}**

**}**

**extension MainTableViewController{**

**func starVPN() {**

**if switchBtn.isOn {**

**VPNManager.shared.connect()**

**} else {**

**VPNManager.shared.disconnect()**

**}**

**}**

**private func updateConnectingVPN() {**

**DispatchQueue.main.asyncAfter(deadline: DispatchTime.now() + 0.1) {**

**if VPNManager.shared.status == .on || VPNManager.shared.status == .connecting {**

**let alertVC = UIAlertController(title: NSLocalizedString("update\_config", comment: ""), message: NSLocalizedString("need\_reconnect", comment: ""), preferredStyle: UIAlertController.Style.alert)**

**alertVC.addAction(UIAlertAction(title: NSLocalizedString("cancel", comment: ""), style: .cancel, handler: nil))**

**alertVC.addAction(UIAlertAction(title: NSLocalizedString("ok", comment: ""), style: .default, handler: { [weak self] \_ in**

**self?.reconnect()**

**}))**

**SVProgressHUD.dismiss()**

**self.present(alertVC, animated: true, completion: nil)**

**}**

**}**

**}**

**private func reconnect() {**

**SVProgressHUD.show(withStatus: NSLocalizedString("build\_config", comment: ""))**

**VPNManager.shared.disconnect()**

**SVProgressHUD.dismiss(withDelay: 0.7)**

**DispatchQueue.main.asyncAfter(deadline: .now() + 0.9) {**

**VPNManager.shared.connect()**

**}**

**}**

**private func updateConnectStatus() {**

**let status = VPNManager.shared.status**

**switch status {**

**case .connecting:**

**break**

**case .on:**

**self.switchBtn.isOn = true**

**self.statusImage.image = UIImage(named: "img\_home\_logo\_light")**

**if getCurrentLanguage() == "tc"{**

**self.statusTitleImage.image = UIImage(named: "VPNConnected\_tc")**

**}else{**

**self.statusTitleImage.image = UIImage(named: "VPNConnected!")**

**}**

**self.statusSubLab.text = NSLocalizedString("use\_vpn\_online", comment: "")**

**self.bottomRestoreView.isHidden = true**

**case .disconnecting:**

**self.switchBtn.isOn = false**

**self.statusImage.image = UIImage(named: "img\_home\_logo\_dark")**

**if getCurrentLanguage() == "tc"{**

**self.statusTitleImage.image = UIImage(named: "VPNDisconnected\_tc")**

**}else{**

**self.statusTitleImage.image = UIImage(named: "VPNDisconnected!")**

**}**

**self.statusSubLab.text = NSLocalizedString("check\_switch", comment: "")**

**self.bottomRestoreView.isHidden = false**

**break**

**case .off:**

**self.switchBtn.isOn = false**

**self.statusImage.image = UIImage(named: "img\_home\_logo\_dark")**

**if getCurrentLanguage() == "tc"{**

**self.statusTitleImage.image = UIImage(named: "VPNDisconnected\_tc")**

**}else{**

**self.statusTitleImage.image = UIImage(named: "VPNDisconnected!")**

**}**

**self.statusSubLab.text = NSLocalizedString("check\_switch", comment: "")**

**self.bottomRestoreView.isHidden = false**

**break**

**}**

**}**

**}**

**import UIKit**

**protocol ServerDelegate {**

**func getServerModel(model:Proxy,isAuto:Bool)**

**}**

**class ServerLocationTableViewController: UITableViewController,StoryboardLoadable {**

**var delegate:ServerDelegate?**

**var proxyList:Array<Proxy> = []{**

**didSet{**

**keys = []**

**dataArr = [:]**

**serverListToModel(list: proxyList)**

**}**

**}**

**var keys:Array<String> = []**

**var dataArr:Dictionary<String,Array<Proxy>> = [:]**

**func serverListToModel(list:Array<Proxy>) {**

**for item in list {**

**let country = item.netArea?.getENWithShort() ?? ""**

**if country.count > 0 {**

**let firstLetterString = String(describing: country.prefix(1))**

**if keys.contains(firstLetterString) {**

**dataArr[firstLetterString]?.append(item)**

**}else{**

**keys.append(firstLetterString)**

**dataArr[firstLetterString] = [item]**

**}**

**}**

**}**

**keys = keys.sorted(by: <)**

**self.tableView.reloadData()**

**}**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**self.tableView.sectionIndexBackgroundColor = .white**

**// Uncomment the following line to preserve selection between presentations**

**// self.clearsSelectionOnViewWillAppear = false**

**// Uncomment the following line to display an Edit button in the navigation bar for this view controller.**

**// self.navigationItem.rightBarButtonItem = self.editButtonItem**

**}**

**// MARK: - Table view data source**

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

**// #warning Incomplete implementation, return the number of sections**

**return keys.count + 1**

**}**

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

**// #warning Incomplete implementation, return the number of rows**

**if section == 0 {**

**return 1**

**}**

**let index = section - 1**

**return dataArr[keys[index]]?.count ?? 0**

**}**

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

**if indexPath.section == 0 {**

**let cell = tableView.dequeueReusableCell(withIdentifier: "autoCell", for: indexPath)**

**return cell**

**}else{**

**let proxy = dataArr[keys[indexPath.section - 1]]?[indexPath.row]**

**let cell = tableView.dequeueReusableCell(withIdentifier: "countryCell", for: indexPath)**

**let imageView = cell.viewWithTag(201) as? UIImageView**

**imageView?.image = UIImage(named: proxy?.netArea?.uppercased() ?? "")**

**let backgroundView = UIView(frame: imageView!.bounds)**

**backgroundView.backgroundColor = .white**

**imageView?.addSubview(backgroundView)**

**let maskLayer = CAShapeLayer()**

**maskLayer.fillRule = CAShapeLayerFillRule.evenOdd //  奇偶层显示规则**

**let basicPath = UIBezierPath(rect: backgroundView.frame) // 底层**

**let maskPath = UIBezierPath(ovalIn: CGRect(x: backgroundView.frame.width/2 - 12 , y: 0, width: 24, height: 24))**

**basicPath.append(maskPath) // 重叠**

**maskLayer.path = basicPath.cgPath**

**backgroundView.layer.mask = maskLayer**

**(cell.viewWithTag(202) as? UILabel)?.text = proxy?.username**

**(cell.viewWithTag(203) as? UILabel)?.text = ""**

**if (cell.viewWithTag(203) as? UILabel)?.text == "" {**

**cell.accessoryType = .none**

**}**

**return cell**

**}**

**}**

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

**if indexPath.section == 0 {**

**if dataArr.count > 0 {**

**let proxy = dataArr[keys[0]]?[indexPath.row]**

**delegate?.getServerModel(model:proxy ?? Proxy(), isAuto: true)**

**self.navigationController?.popViewController(animated: true)**

**}**

**}else{**

**let proxy = dataArr[keys[indexPath.section - 1]]?[indexPath.row]**

**delegate?.getServerModel(model:proxy ?? Proxy(), isAuto: false)**

**self.navigationController?.popViewController(animated: true)**

**}**

**}**

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

**if indexPath.section == 0 {**

**return 68**

**}else{**

**return 44**

**}**

**}**

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

**if section == 0 {**

**return 38**

**}else{**

**return 28**

**}**

**}**

**override func tableView(\_ tableView: UITableView, viewForFooterInSection section: Int) -> UIView? {**

**return UIView()**

**}**

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

**return 0.001**

**}**

**override func sectionIndexTitles(for tableView: UITableView) -> [String]? {**

**return keys**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView(frame: CGRect(x: 0, y: 0, width: LJScreenWidth, height: 38))**

**view.backgroundColor = UIColor.colorWithHexColorString("F8F8F8", alpha: 0.82)**

**let lab = UILabel(frame: CGRect(x: 16, y: section == 0 ? 10 : 0, width: LJScreenWidth, height: 28))**

**if section == 0 {**

**lab.text = NSLocalizedString("recommend", comment: "")**

**}else{**

**lab.text =  keys[section - 1]**

**}**

**lab.font = UIFont.pingFangSC\_Semibold(size: 17)**

**lab.textColor = .black**

**lab.backgroundColor = .clear**

**view.addSubview(lab)**

**return view**

**}**

**override func tableView(\_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {**

**if section == 0 {**

**return NSLocalizedString("recommend", comment: "")**

**}else{**

**return keys[section - 1]**

**}**

**}**

**override func scrollViewDidChangeAdjustedContentInset(\_ scrollView: UIScrollView) {**

**}**

**override func tableView(\_ tableView: UITableView, sectionForSectionIndexTitle title: String, at index: Int) -> Int {**

**return index**

**}**

**}**

**class ServerListModel: NSObject {**

**var title = ""**

**var subList:Array<Dictionary<String,Proxy>> = []**

**}**

**import WCDBSwift**

**import CocoaLumberjackSwift**

**protocol BaseModel: TableCodable {**

**static var prefix: String { get }**

**static var tableName: String { get }**

**static func createTable(database: Database?)**

**}**

**extension BaseModel {**

**static var prefix: String {**

**return "lt\_"**

**}**

**static var tableName: String {**

**return prefix + "\(self)".lowercased()**

**}**

**static func createTable(database: Database?) {**

**guard let database = database else { return }**

**let table = tableName**

**do {**

**let exist = try database.isTableExists(table)**

**if !exist {**

**try database.create(table: table, of: self)**

**}**

**} catch {**

**DDLogError("Create table \(tableName) error: \(error.localizedDescription)")**

**}**

**}**

**}**

**import WCDBSwift**

**import CocoaLumberjackSwift**

**public class DatabaseManager {**

**public static let shared = DatabaseManager()**

**public var database: Database?**

**private let databaseName = "leiter.db"**

**private lazy var path: String? = {**

**guard let documentPath =  NSSearchPathForDirectoriesInDomains(FileManager.SearchPathDirectory.documentDirectory, FileManager.SearchPathDomainMask.userDomainMask, true).last,**

**let bundleIdentifier = Bundle.main.bundleIdentifier else {**

**return nil**

**}**

**let p = documentPath + "/" + bundleIdentifier + "/" + databaseName**

**return p**

**}()**

**private init() {**

**if let path = path {**

**self.database = Database(withPath: path)**

**DDLogInfo("Database create: \(self.database.debugDescription)")**

**}**

**}**

**public func setup() {**

**createAllTables()**

**}**

**private func createAllTables() {**

**do {**

**try self.database?.run(transaction: {**

**Proxy.createTable(database: self.database)**

**})**

**} catch {**

**DDLogWarn("Database create table transaction error:\(error.localizedDescription)")**

**}**

**}**

**}**

**import UIKit**

**protocol TermsPriDelegate {**

**func termsClick()**

**func privacyClick()**

**}**

**class TermsAndPrivacyView: UIView,UITextViewDelegate,NibLoadable {**

**@IBOutlet weak var textView: UITextView!**

**var delegate:TermsPriDelegate?**

**override func awakeFromNib() {**

**setTextView()**

**}**

**func setTextView() {**

**textView.delegate = self**

**let str1 = NSLocalizedString("privacy\_1", comment: "")**

**let str2 = NSLocalizedString("terms\_of\_use", comment: "")**

**let str3 = NSLocalizedString("and", comment: "")**

**let str4 = NSLocalizedString("privacy\_policy", comment: "")**

**let str5 = NSLocalizedString("privacy\_2", comment: "")**

**let str:NSString = (str1 + str2 + str3 + str4 + str5) as NSString**

**let range1 = str.range(of: str2)**

**let range2 = str.range(of: str4)**

**let parStyle = NSMutableParagraphStyle()**

**parStyle.lineSpacing = 0.5**

**let att = NSMutableAttributedString(string: str as String, attributes: [NSAttributedString.Key.font:UIFont.pingFangSC\_Regular(size: 14),NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("000000", alpha: 0.7),NSAttributedString.Key.paragraphStyle:parStyle])**

**//        att.addAttributes([NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("FF7F24")], range: range1)**

**//        att.addAttributes([NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("FF7F24")], range: range2)**

**let valueStr1 = "TermsofService://\(str2)".addingPercentEncoding(withAllowedCharacters: CharacterSet.urlFragmentAllowed)**

**let valueStr2 = "PrivacyPolicy://\(str4)".addingPercentEncoding(withAllowedCharacters: CharacterSet.urlFragmentAllowed)**

**att.addAttributes([NSAttributedString.Key.link : valueStr1 as Any,], range: range1)**

**att.addAttributes([NSAttributedString.Key.link : valueStr2 as Any], range: range2)**

**textView.linkTextAttributes = [NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("FF7F24")]**

**textView.attributedText = att**

**}**

**func textView(\_ textView: UITextView, shouldInteractWith URL: URL, in characterRange: NSRange) -> Bool {**

**if URL.scheme == "TermsofService" {**

**self.clickTerms()**

**return false**

**}else if URL.scheme == "PrivacyPolicy"{**

**self.clickPrivacy()**

**return false**

**}**

**return true**

**}**

**func clickTerms() {**

**print("TermsofService")**

**self.delegate?.termsClick()**

**self.removeFromSuperview()**

**}**

**func clickPrivacy() {**

**print("PrivacyPolicy")**

**self.delegate?.privacyClick()**

**self.removeFromSuperview()**

**}**

**@IBAction func acceptBtn(\_ sender: UIButton) {**

**self.removeFromSuperview()**

**}**

**}**

**import UIKit**

**enum ProductId:String {**

**case days7 = "com.yk.ultraspeed.product.7days"**

**case days30 = "com.yk.ultraspeed.product.30days"**

**case days90 = "com.yk.ultraspeed.product.90days"**

**case days180 = "com.yk.ultraspeed.product.180days"**

**case days360 = "com.yk.ultraspeed.product.360days"**

**}**

**class PurchTableViewController: UITableViewController,StoryboardLoadable,UITextViewDelegate {**

**@IBOutlet weak var titleImageView: UIImageView!**

**@IBOutlet weak var textView: UITextView!**

**var currentId:ProductId = .days360**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**setTextView()**

**tableView.selectRow(at: IndexPath(row: 0, section: 0), animated: true, scrollPosition: .none)**

**NotificationCenter.default.addObserver(forName: NSNotification.Name("purchaseRes"), object: nil, queue: nil) { notifi in**

**if notifi.userInfo?["res"] as? Bool ?? false {**

**self.dismiss(animated: true, completion: nil)**

**}**

**}**

**if getCurrentLanguage() == "tc"{**

**titleImageView.image = UIImage(named: "ChooseYourPlan\_tc")**

**}else{**

**titleImageView.image = UIImage(named: "ChooseYourPlan")**

**}**

**}**

**// MARK: - Table view data source**

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

**// #warning Incomplete implementation, return the number of sections**

**return 1**

**}**

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

**// #warning Incomplete implementation, return the number of rows**

**return 5**

**}**

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

**if indexPath.row != 0 {**

**let cell = tableView.cellForRow(at: IndexPath(row: 0, section: 0))**

**guard let imageView = cell?.viewWithTag(101) else{**

**return**

**}**

**(imageView as! UIImageView).image = UIImage(named: "sub")**

**}**

**let cell = tableView.cellForRow(at: indexPath)**

**guard let imageView = cell?.viewWithTag(101) else{**

**return**

**}**

**(imageView as! UIImageView).image = UIImage(named: "sub-sel")**

**if indexPath.row == 0 {**

**currentId = .days360**

**}else if indexPath.row == 1{**

**currentId = .days180**

**}else if indexPath.row == 2{**

**currentId = .days90**

**}else if indexPath.row == 3{**

**currentId = .days30**

**}else if indexPath.row == 4{**

**currentId = .days7**

**}**

**}**

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

**let cell = tableView.cellForRow(at: indexPath)**

**guard let imageView = cell?.viewWithTag(101) else{**

**return**

**}**

**(imageView as! UIImageView).image = UIImage(named: "sub")**

**}**

**@IBAction func reStoreClick(\_ sender: UIButton) {**

**PurchaseTool.sharedInstance.restorePurchases { res in**

**if res{**

**DispatchQueue.main.async {**

**self.dismiss(animated: true, completion: nil)**

**}**

**}**

**}**

**}**

**@IBAction func closeClick(\_ sender: UIButton) {**

**self.dismiss(animated: true, completion: nil)**

**}**

**@IBAction func continueClick(\_ sender: UIButton) {**

**let prodi = currentId.rawValue**

**PurchaseTool.sharedInstance.payWithProduct(productID: prodi) { res  in**

**if res {**

**print("success")**

**}else{**

**print("fail")**

**}**

**}**

**}**

**func setTextView() {**

**let str1 = NSLocalizedString("pircha\_1", comment: "")**

**let str2 = NSLocalizedString("terms\_of\_use", comment: "")**

**let str3 = NSLocalizedString("purcha\_2", comment: "")**

**let str4 = NSLocalizedString("privacy\_policy", comment: "")**

**let str5 = NSLocalizedString("purcha\_3", comment: "")**

**let str:NSString = (str1 + str2 + str3 + str4 + str5) as NSString**

**let range1 = str.range(of: str2)**

**let range2 = str.range(of: str4)**

**let parStyle = NSMutableParagraphStyle()**

**parStyle.lineSpacing = 0.5**

**let att = NSMutableAttributedString(string: str as String, attributes: [NSAttributedString.Key.font:UIFont.pingFangSC\_Regular(size: 10),NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("000000", alpha: 0.7),NSAttributedString.Key.paragraphStyle:parStyle])**

**let valueStr1 = "TermsofService://\(str2)".addingPercentEncoding(withAllowedCharacters: CharacterSet.urlFragmentAllowed)**

**let valueStr2 = "PrivacyPolicy://\(str4)".addingPercentEncoding(withAllowedCharacters: CharacterSet.urlFragmentAllowed)**

**att.addAttributes([NSAttributedString.Key.link : valueStr1 as Any,], range: range1)**

**att.addAttributes([NSAttributedString.Key.link : valueStr2 as Any], range: range2)**

**textView.linkTextAttributes = [NSAttributedString.Key.foregroundColor:UIColor.colorWithHexColorString("FF7F24")]**

**textView.attributedText = att**

**}**

**func textView(\_ textView: UITextView, shouldInteractWith URL: URL, in characterRange: NSRange) -> Bool {**

**if URL.scheme == "TermsofService" {**

**self.clickTerms()**

**return false**

**}else if URL.scheme == "PrivacyPolicy"{**

**self.clickPrivacy()**

**return false**

**}**

**return true**

**}**

**func clickTerms() {**

**print("TermsofService")**

**let vc = PureWebViewController()**

**vc.urlStr = TermsofServerUrl**

**vc.titleStr = NSLocalizedString("terms\_of\_use", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}**

**func clickPrivacy() {**

**print("PrivacyPolicy")**

**let vc = PureWebViewController()**

**vc.urlStr = PrivacyPolicyUrl**

**vc.titleStr = NSLocalizedString("privacy\_policy", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**self.present(vc, animated: true, completion: nil)**

**}**

**}**

**import UIKit**

**import SwiftyStoreKit**

**import StoreKit**

**import SVProgressHUD**

**typealias ApplePayBlock = (Bool) ->()**

**class PurchaseTool: NSObject {**

**static let sharedInstance: PurchaseTool = {**

**let instance = PurchaseTool()**

**return instance**

**}()**

**var success:ApplePayBlock?**

**var fail:ApplePayBlock?**

**func payWithProduct(productID: String,payResult: ((Bool)->())? = nil) {**

**SVProgressHUD.showWithNativeType()**

**//        SVProgressHUD.showStatus(string: "Plase Wait...")**

**SwiftyStoreKit.purchaseProduct(productID) { result in**

**switch result {**

**case .success(let product):**

**// 购买正式验证**

**self.verifyPurchase(forceupdate:true,result: { result in**

**payResult?(result)**

**})**

**//                self.verifyReceipt(service: .production)**

**if product.needsFinishTransaction {**

**SwiftyStoreKit.finishTransaction(product.transaction)**

**}**

**print("Purchase Success: \(product.productId)")**

**case .error(let error):**

**SVProgressHUD.dismiss()**

**switch error.code {**

**case .unknown:**

**print("Unknown error. Please contact support")**

**case .clientInvalid:**

**print("Not allowed to make the payment")**

**case .paymentCancelled:**

**break**

**case .paymentInvalid:**

**print("The purchase identifier was invalid")**

**case .paymentNotAllowed:**

**print("The device is not allowed to make the payment")**

**case .storeProductNotAvailable:**

**print("The product is not available in the current storefront")**

**case .cloudServicePermissionDenied:**

**print("Access to cloud service information is not allowed")**

**case .cloudServiceNetworkConnectionFailed:**

**print("Could not connect to the network")**

**case .cloudServiceRevoked:**

**print("User has revoked permission to use this cloud service")**

**default:**

**print((error as NSError).localizedDescription)**

**}**

**SVProgressHUD.showError(withStatus: (error as NSError).localizedDescription)**

**}**

**}**

**}**

**///服务器验证内购**

**func verifyPurchase(forceupdate:Bool?=false,isRest:Bool? = false,result:((Bool)->())? = nil) {**

**let receipUrl = Bundle.main.appStoreReceiptURL**

**let receiptData = try? Data(contentsOf: receipUrl!)**

**let receiptString = receiptData?.base64EncodedString(options: .endLineWithLineFeed) ?? ""**

**///服务器验证请求**

**NetworkingTool.newVerifyReceipt(receiptStr: receiptString, forceupdate: forceupdate, success: { response in**

**weak var weakSelf = self**

**if forceupdate == false && response.code == 403{**

**result?(false)**

**SVProgressHUD.dismiss()**

**}else if response.code == 200{**

**print("\(response.message)---\(response.expires)")**

**if response.message == "ok"{**

**UserDefaults.standard.set(response.product\_id, forKey: "product\_id")**

**weakSelf?.updateExpireTime(time: Int(response.expires) ?? 0)**

**if weakSelf?.getCurVipStatus() ?? false {**

**result?(true)**

**if forceupdate ?? false && isRest == false{**

**SVProgressHUD.showSuccess(withStatus: NSLocalizedString("purchase\_success", comment: ""))**

**}**

**}else{**

**result?(false)**

**if forceupdate ?? false && isRest == false{**

**SVProgressHUD.showError(withStatus: NSLocalizedString("purchase\_failed", comment: ""))**

**}**

**}**

**}else{**

**result?(false)**

**if forceupdate ?? false  && isRest == false{**

**SVProgressHUD.showError(withStatus: NSLocalizedString("purchase\_failed", comment: ""))**

**}**

**}**

**}else{**

**result?(false)**

**SVProgressHUD.showError(withStatus: NSLocalizedString("purchase\_failed", comment: ""))**

**print("服务器错误")**

**}**

**}) { error in**

**SVProgressHUD.dismiss()**

**print(error)**

**}**

**//        NetworkingTool.verifyReceipt(receiptStr: receiptString,forceupdate: forceupdate, success: { response in**

**//            weak var weakSelf = self**

**//            if forceupdate == false && response.code == 403{**

**//                result?(false)**

**//                SVProgressHUD.dismiss()**

**//            }else if response.code == 200{**

**//                print("\(response.message)---\(response.expires)")**

**//                           if response.message == "ok"{**

**//                            UserDefaults.standard.set(response.product\_id, forKey: "product\_id")**

**//                               weakSelf?.updateExpireTime(time: Int(response.expires) ?? 0)**

**//                               if weakSelf?.getCurVipStatus() ?? false {**

**//                                result?(true)**

**//                                if forceupdate ?? false {**

**//                                     SVProgressHUD.showSuccess(withStatus: "Purchase Success")**

**//                                }**

**//                               }else{**

**//                                result?(false)**

**//                                if forceupdate ?? false {**

**//                                    SVProgressHUD.showError(withStatus: "Purchase Failed")**

**//                                }**

**//                               }**

**//                           }else{**

**//                            result?(false)**

**//                            if forceupdate ?? false {**

**//                                SVProgressHUD.showError(withStatus: "Purchase Failed")**

**//                            }**

**//                           }**

**//            }else{**

**//                result?(false)**

**//                SVProgressHUD.showError(withStatus: "Purchase Failed")**

**//                print("服务器错误")**

**//            }**

**//**

**//        }) { error in**

**//            SVProgressHUD.dismiss()**

**//            print(error)**

**//        }**

**}**

**///内购本地验证**

**func verifyReceipt(service:AppleReceiptValidator.VerifyReceiptURLType,res:((Bool)->())? = nil) {**

**let receiptValidator = AppleReceiptValidator(service: service, sharedSecret: "4879940749f94984abf6ce4befa187f3")**

**SwiftyStoreKit.verifyReceipt(using: receiptValidator) { (result) in**

**switch result {**

**case .success (let receipt):**

**let status: Int = receipt["status"] as! Int**

**if status == 21007 {**

**// sandbox验证**

**self.verifyReceipt(service: .sandbox)**

**}**

**print("receipt：\(receipt)")**

**let dic = receipt["receipt"]**

**let dicInApp = dic?["in\_app"] as! Array<Dictionary<String,Any>>**

**print(dicInApp as Any)**

**var expireTime = 0;**

**for iapItem in dicInApp{**

**let iapTime = Int(iapItem["expires\_date\_ms"] as? String ?? "0") ?? 0**

**print(expireTime)**

**if expireTime < iapTime {**

**expireTime = iapTime**

**}**

**}**

**print("lastTime:\(expireTime)")**

**weak var weakSelf = self**

**weakSelf?.updateExpireTime(time: expireTime)**

**break**

**case .error(let error):**

**print("error：\(error)")**

**break**

**}**

**}**

**}**

**func updateExpireTime(time:Int){**

**let defaults = UserDefaults.standard**

**defaults.set(time, forKey: "product\_tag")**

**let now = self.getNowStringMilliStamp()**

**let timeStr = "\(time)"**

**let nowStr = "\(now)"**

**print("timeStr:\(Date.timeStampToYYMMDDHHmmSS(timeStr))-----------now:\(Date.timeStampToYYMMDDHHmmSS(nowStr))")**

**if time > Int(now) ?? 0 {**

**defaults.set(true, forKey: "cur\_status")**

**NotificationCenter.default.post(Notification(name: Notification.Name(rawValue: "purchaseRes"), object: nil, userInfo: ["res":true]))**

**}else{**

**NotificationCenter.default.post(Notification(name: Notification.Name(rawValue: "purchaseRes"), object: nil, userInfo: ["res":false]))**

**}**

**SVProgressHUD.dismiss()**

**}**

**func restorePurchases(success:((\_ resulet:Bool)->())?) {**

**SVProgressHUD.showStatus(string: NSLocalizedString("restoreing", comment: ""))**

**let signal = DispatchSemaphore(value: 0)**

**var isSuccess = false**

**DispatchQueue.global().async {**

**SwiftyStoreKit.restorePurchases(atomically: true) { results in**

**SVProgressHUD.dismiss()**

**if results.restoreFailedPurchases.count > 0 {**

**print("Restore Failed: \(results.restoreFailedPurchases)")**

**isSuccess = false**

**signal.signal()**

**SVProgressHUD.showSuccess(withStatus: NSLocalizedString("restore\_failed", comment: ""))**

**}**

**else if results.restoredPurchases.count > 0 {**

**print("Restore Success: \(results.restoredPurchases)")**

**//                            self.verifyReceipt(service: .production)**

**self.verifyPurchase(isRest: true) { res in**

**isSuccess = res**

**signal.signal()**

**if res{**

**SVProgressHUD.showSuccess(withStatus: NSLocalizedString("restore\_success", comment: ""))**

**}else{**

**SVProgressHUD.showSuccess(withStatus: NSLocalizedString("restore\_failed", comment: ""))**

**}**

**}**

**}**

**else {**

**SVProgressHUD.showSuccess(withStatus: NSLocalizedString("nothing\_restore", comment: ""))**

**print("Nothing to Restore")**

**isSuccess = false**

**signal.signal()**

**}**

**}**

**signal.wait()**

**if success != nil{**

**success!(isSuccess)**

**}**

**}**

**}**

**func getCurVipStatus() -> Bool{**

**// MARK: -SSS -验证内购返回成功**

**return true**

**let defaults = UserDefaults.standard**

**let expTime = defaults.integer(forKey: "product\_tag")**

**let now = self.getNowStringMilliStamp()**

**if expTime > Int(now) ?? 0 {**

**defaults.set(true, forKey: "cur\_status")**

**return true**

**}else{**

**defaults.set(false, forKey: "cur\_status")**

**return false**

**}**

**}**

**func getNowStringMilliStamp()->String{**

**//获取当前时间戳**

**let date = Date()**

**let timeInterval:TimeInterval = TimeInterval(date.timeIntervalSince1970)**

**let millisecond = CLongLong(round(timeInterval \* 1000))**

**return "\(millisecond)"**

**}**

**func goInAppPurchaseView() {**

**//        let vc = InAppPurchaseTableViewController.loadStoryboard(name: "Main")**

**//        vc.modalPresentationStyle = .fullScreen**

**//        UIApplication.shared.keyWindow?.rootViewController?.present(vc, animated: true, completion: nil)**

**}**

**}**

**import UIKit**

**import WebKit**

**import SnapKit**

**class PureWebViewController: UIViewController,WKNavigationDelegate {**

**var webView:WKWebView?**

**var urlStr:String = ""**

**var titleStr:String = ""**

**var block:(()->())?**

**lazy var progressView:UIProgressView = {**

**let progressView = UIProgressView()**

**progressView.trackTintColor = UIColor.white**

**progressView.progressTintColor = UIColor(red: 66/255.0, green: 192/255.0, blue: 46/255.0, alpha: 1)**

**return progressView**

**}()**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**initToolbar()**

**initWebView()**

**loadWebView()**

**// Do any additional setup after loading the view.**

**}**

**func initToolbar(){**

**let toolBar = UIView(frame: CGRect(x: 0, y: 0, width: LJScreenWidth, height: LJ\_Navi\_Height))**

**toolBar.backgroundColor = UIColor.colorWithHexColorString("0xFFFFFF")**

**let lineView = UIView(frame: CGRect(x: 0, y: LJ\_Navi\_Height - 1, width: LJScreenWidth, height: 1))**

**let goBackButton = UIButton(type: .custom)**

**goBackButton.setImage(UIImage(named: "navback"), for: .normal)**

**goBackButton.addTarget(self, action: #selector(backClicked), for: .touchUpInside)**

**let titleLab = UILabel()**

**if self.titleStr != "" {**

**titleLab.text = self.titleStr**

**}**

**titleLab.textColor = UIColor.black**

**titleLab.font = UIFont.boldSystemFont(ofSize: 17)**

**toolBar.addSubview(goBackButton)**

**toolBar.addSubview(titleLab)**

**toolBar.addSubview(lineView)**

**self.view.addSubview(toolBar)**

**goBackButton.snp.makeConstraints { make in**

**make.left.equalTo(toolBar.snp\_leftMargin).offset(6)**

**make.bottom.equalTo(toolBar.snp\_bottomMargin).offset(-4)**

**make.width.height.equalTo(30)**

**}**

**titleLab.snp.makeConstraints { make in**

**make.centerX.equalTo(toolBar.snp\_centerXWithinMargins)**

**make.bottom.equalTo(toolBar.snp\_bottomMargin).offset(-7)**

**make.height.equalTo(22)**

**}**

**}**

**func initWebView() {**

**self.progressView.frame = CGRect(x: 0, y: LJ\_Navi\_Height, width: LJScreenWidth, height: 2)**

**self.view.addSubview(progressView)**

**self.view.backgroundColor = UIColor.white**

**self.webView = WKWebView(frame: CGRect(x: 16, y: LJ\_Navi\_Height + 12, width: LJScreenWidth - 32, height: LJScreenHeight - LJ\_Navi\_Height - 12))**

**self.webView?.backgroundColor = UIColor.clear**

**self.webView?.navigationDelegate = self**

**self.webView?.addObserver(self, forKeyPath: "estimatedProgress", options: .new, context: nil)**

**self.view.addSubview(self.webView!);**

**}**

**func loadWebView() {**

**if self.urlStr != "" {**

**let request = URLRequest(url: URL(string: self.urlStr)!)**

**self.webView?.load(request)**

**}**

**}**

**@objc func backClicked(){**

**block?()**

**self.dismiss(animated: true, completion: nil)**

**}**

**func webView(\_ webView: WKWebView, decidePolicyFor navigationAction: WKNavigationAction, decisionHandler: @escaping (WKNavigationActionPolicy) -> Void) {**

**progressView.setProgress(0.0, animated: false)**

**if navigationAction.targetFrame == nil {**

**webView .load(navigationAction.request)**

**}**

**decisionHandler(WKNavigationActionPolicy.allow)**

**}**

**func webView(\_ webView: WKWebView, didFinish navigation: WKNavigation!) {**

**progressView.setProgress(0.0, animated: false)**

**}**

**override func observeValue(forKeyPath keyPath: String?, of object: Any?, change: [NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {**

**//        SVProgressHUD.dismiss()**

**if (keyPath == "estimatedProgress") {**

**progressView.isHidden = self.webView?.estimatedProgress == 1**

**progressView.setProgress(Float(self.webView!.estimatedProgress), animated: true)**

**}**

**}**

**func webViewWebContentProcessDidTerminate(\_ webView: WKWebView) {**

**self.webView?.reload()**

**}**

**/\***

**// MARK: - Navigation**

**// In a storyboard-based application, you will often want to do a little preparation before navigation**

**override func prepare(for segue: UIStoryboardSegue, sender: Any?) {**

**// Get the new view controller using segue.destination.**

**// Pass the selected object to the new view controller.**

**}**

**\*/**

**}**

**import UIKit**

**class AboutTableViewController: UITableViewController,StoryboardLoadable {**

**@IBOutlet weak var versionLab: UILabel!**

**override func viewDidLoad() {**

**super.viewDidLoad()**

**versionLab.text = "\(Bundle.main.infoDictionary!["CFBundleShortVersionString"] as! String)"**

**// Uncomment the following line to preserve selection between presentations**

**// self.clearsSelectionOnViewWillAppear = false**

**// Uncomment the following line to display an Edit button in the navigation bar for this view controller.**

**// self.navigationItem.rightBarButtonItem = self.editButtonItem**

**}**

**// MARK: - Table view data source**

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

**// #warning Incomplete implementation, return the number of sections**

**return 2**

**}**

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

**// #warning Incomplete implementation, return the number of rows**

**return 3**

**}**

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

**return 10**

**}**

**override func tableView(\_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {**

**let view = UIView()**

**view.backgroundColor = UIColor.colorWithHexColorString("F9F9F9")**

**return view**

**}**

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

**if indexPath.section == 0 {**

**if indexPath.row == 0 {**

**let vc = PureWebViewController()**

**vc.urlStr = AboutUsUrl**

**vc.titleStr = NSLocalizedString("about", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**UIApplication.shared.keyWindow?.rootViewController?.present(vc, animated: true, completion: nil)**

**}else if indexPath.row == 1{**

**let image = UIImage(named: "img\_subscribed\_logo")**

**let url = URL(string: "https://apps.apple.com/cn/app/id\(Appid)")!**

**///  applicationActivities 可以没有元素，系统会自动选择合适的平台**

**let activityController = UIActivityViewController(activityItems: ["Ultraspeed VPN", image!,url], applicationActivities: [])**

**/// 可以排除一些不希望的平台 UIActivityType**

**//             activityController.excludedActivityTypes = [.postToTencentWeibo, .airDrop, .openInIBooks, .postToVimeo, .postToFlickr, .addToReadingList, .saveToCameraRoll, .assignToContact, .copyToPasteboard, .print, .mail]**

**/// 分享完成后可以看到分享平台的相关字符串**

**//部分打印内容**

**/// UIActivityType(\_rawValue: com.apple.UIKit.activity.PostToWeibo)**

**/// UIActivityType(\_rawValue: com.tencent.mqq.ShareExtension) // qq**

**/// UIActivityType(\_rawValue: com.apple.mobilenotes.SharingExtension) 笔记**

**/// UIActivityType(\_rawValue: com.tencent.xin.sharetimeline) weixin**

**/// com.apple.UIKit.activity.PostToFlickr**

**/// com.apple.UIKit.activity.PostToVimeo**

**activityController.completionWithItemsHandler = {**

**(type, flag, array, error) -> Swift.Void in**

**print(type ?? "")**

**}**

**present(activityController, animated: true) { }**

**}else if indexPath.row == 2{**

**let appUrl = "https://itunes.apple.com/cn/app/id\(Appid)?action=write-review"**

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

**if #available(iOS 10.0, \*) {**

**UIApplication.shared.open(URL(string: appUrl)!, options: [:], completionHandler: nil)**

**} else {**

**// Fallback on earlier versions**

**}**

**}**

**}**

**}**

**if indexPath.section == 1 {**

**if indexPath.row == 0 {**

**let vc = PureWebViewController()**

**vc.urlStr = PrivacyPolicyUrl**

**vc.titleStr = NSLocalizedString("privacy\_policy", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**UIApplication.shared.keyWindow?.rootViewController?.present(vc, animated: true, completion: nil)**

**}else if indexPath.row == 1{**

**let vc = PureWebViewController()**

**vc.urlStr = TermsofServerUrl**

**vc.titleStr = NSLocalizedString("terms\_of\_use", comment: "")**

**vc.modalPresentationStyle = .fullScreen**

**UIApplication.shared.keyWindow?.rootViewController?.present(vc, animated: true, completion: nil)**

**}**

**}**

**}**

**}**