实验编号： 5 **四川师大《IOS》实验报告 2018** 年 **10** 月 **10** 日

### **计算机科学学院** 2016 级 4 班 实验名称： 纯代码版本Hello World \_

姓名：\_樊琳\_\_ 学号：\_2016110408\_ 指导老师：\_\_李贵洋\_\_ 实验成绩:\_\_\_\_\_

**实验\_五\_ \_\_\_\_\_\_**纯代码版本Hello World **\_\_\_\_\_\_\_\_**

1. 实验目的及要求
2. 通过实现一款纯代码版Hello World程序，深入理解iOS程序的运行机制；
3. 实验内容
4. 纯代码界面
   1. 删除storyboard和控制器代码；
   2. 新建控制器的子类，并完成相关代码；
      1. 代码中生成label（outlet）和button；
      2. 将label和button加入根view中；
      3. button添加像self（target）发射action（selector）的操作；
      4. 实现clicked响应代码（selector）
   3. 在app delegate中完成window和controller的创建；
   4. 自己制作一款Icon并添加Icon到程序中；
   5. 添加图片到程序中，并在程序中显示图片(可以Bundle定位)；
   6. 将自定制的视图添加到界面并显示；
5. 实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）
6. 纯代码界面
   1. 删除storyboard和控制器代码；
   2. 新建控制器的子类，并完成相关代码；
      1. 代码中生成label（outlet）和button；
      2. 将label和button加入根view中；
      3. button添加像self（target）发射action（selector）的操作；
      4. 实现clicked响应代码（selector）
   3. 在app delegate中完成window和controller的创建；
   4. 自己制作一款Icon并添加Icon到程序中；
   5. 添加图片到程序中，并在程序中显示图片(可以Bundle定位)；
   6. 将自定制的视图添加到界面并显示；

* 程序代码：

//

// MyViewController.swift

// 2

//

// Created by student on 2018/12/13.

// Copyright © 2018年 fl. All rights reserved.

//

import UIKit

class MyViewController: UIViewController {

//声明UILabel对象

var label: UILabel!

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view.

//设置根视图视图背景色

self.view.backgroundColor = UIColor.white

//

// let myView = MyView(frame: CGRect(x: 0, y: 20, width: self.view.frame.width, height: self.view.frame.height - 20))

// myView.drawCurve { (x) -> CGFloat in

// return x \* x

// }

// self.view.addSubview(myView)

//创建UILabel对象

label = UILabel(frame: CGRect(x: 100, y: 100, width: 100, height: 40))

//设置label显示文字

label.text = "hello world"

//将label加入到视图控制器中

self.view.addSubview(label)

//创建UIButton对象

let btn = UIButton(frame: CGRect(x: 100, y: 250, width: 100, height: 40))

//设置正常状态下按钮的标题

btn.setTitle("点我", for: .normal)

//设置按钮边框宽度

btn.layer.borderWidth = 1

//设置正常状态下标题的颜色

btn.setTitleColor(UIColor.black, for: .normal)

//设置高亮状态下（被点击且未释放鼠标）标题的颜色

btn.setTitleColor(UIColor.brown, for: .highlighted)

//给按钮添加target-action

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

self.view.addSubview(btn)

//创建UIImageView对象

let imageView = UIImageView(frame: CGRect(x: 10, y: 400, width: self.view.frame.width - 20, height: (self.view.frame.width - 20) \* 0.618))

//以该路径创建图片

let image = UIImage(named: "1")

//将图片加入到imageView中

imageView.image = image

//将imageView加入到控制器中

self.view.addSubview(imageView)

}

/// 按钮点击事件，改变label显示的文字

@objc func clicked() {

if label.text == "hello world" {

label.text = "I'm clicked"

} else {

label.text = "hello world"

}

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

// Dispose of any resources that can be recreated.

}

/\*

// 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.

}

\*/

}

//

// AppDelegate.swift

// 2

//

// Created by student on 2018/12/13.

// Copyright © 2018年 fl. All rights reserved.

//

import UIKit

import CoreData

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

var window: UIWindow?

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

// Override point for customization after application launch.

//创建UIWindow对象

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

//设置window的根视图控制器为MyViewController

window?.rootViewController = MyViewController()

//显示window

window?.makeKeyAndVisible()

return true

}

func applicationWillResignActive(\_ application: UIApplication) {

// Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.

// Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.

}

func applicationDidEnterBackground(\_ application: UIApplication) {

// Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.

// If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

}

func applicationWillEnterForeground(\_ application: UIApplication) {

// Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.

}

func applicationDidBecomeActive(\_ application: UIApplication) {

// Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

}

func applicationWillTerminate(\_ application: UIApplication) {

// Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.

// Saves changes in the application's managed object context before the application terminates.

self.saveContext()

}

// MARK: - Core Data stack

lazy var persistentContainer: NSPersistentContainer = {

/\*

The persistent container for the application. This implementation

creates and returns a container, having loaded the store for the

application to it. This property is optional since there are legitimate

error conditions that could cause the creation of the store to fail.

\*/

let container = NSPersistentContainer(name: "\_")

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

if let error = error as NSError? {

// Replace this implementation with code to handle the error appropriately.

// fatalError() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

/\*

Typical reasons for an error here include:

\* The parent directory does not exist, cannot be created, or disallows writing.

\* The persistent store is not accessible, due to permissions or data protection when the device is locked.

\* The device is out of space.

\* The store could not be migrated to the current model version.

Check the error message to determine what the actual problem was.

\*/

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

}

})

return container

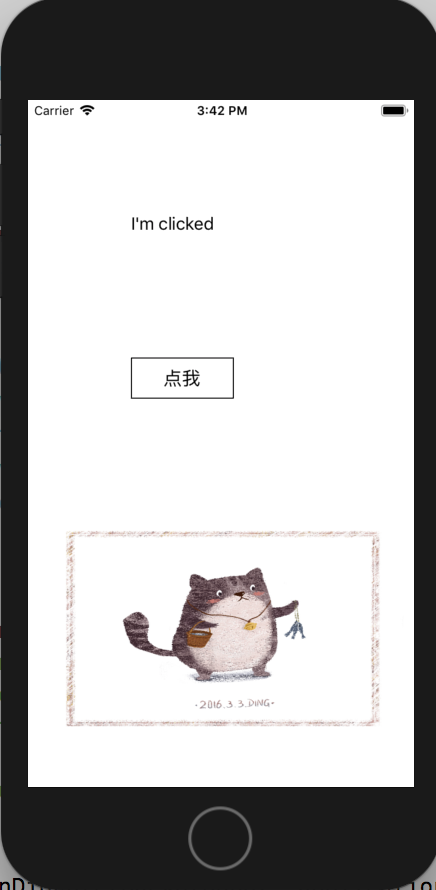
}()

// MARK: - Core Data Saving support

func saveContext () {

let context = persistentContainer.viewContext

* 运行结果：



1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

这次实验要求我们通过实现一款纯代码版Hello World程序，深入理解iOS程序的运行机制。其实在老师的PPT中关于iOS程序的运行机制已经讲述的非常清楚了。这次实验对于我来说不是很难。

Github地址：

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。