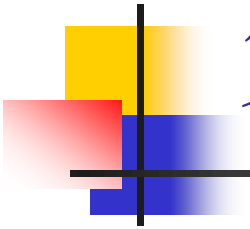
第五部分　控件

授课教师：杨文川

控件（2学时）

- 控件是构建用户界面时的最基本的单元。
- 苹果的开发组件库中提供了大量功能强大的控件，每种控件都能满足特定的应用需求。
- 通过组合这些控件，可以构建出功能强大的复杂应用，并且非常稳定和高效。



第五部分　控件

本章目的在于掌握文本编辑框、文本编辑区等控件以及容器视图控制器的相关知识，并通过多个应用实例来进行讲解。

- 文本编辑框
- 文本编辑区
- 选择控件等
- 进度显示控件
- 警告框控制器

文本编辑框

- 文本编辑框，即：UITextField，是一种图形化的控件，用来搭建用户界面，提供给用户一种可编辑单行文本的控件。
- 用户可通过手机屏幕上的系统内置键盘，在文本编辑框内输入或编辑文本信息。可以根据实际应用的需求，通过配置的方式来定制键盘。
- 文本编辑框通过目标-动作模式来处理交互事件。
  - 即：当某个事件发生的时候，调用其控制器对象中的相应事件处理函数。
- 文本编辑框还采用了委托模式，通过委托对象来报告UITextField在编辑过程中的各种状态变化，并对特定事件进行处理。

使用文本编辑框

- 从可视控件库中找到UITextField
- 将其拖拽到设计面板中的指定位置
- 然后对其进行配置，具体内容包括：
  - 根据需要配置目标-动作对（一个或多个）：即连接关系
  - 定制内置键盘
  - 指定代理（被委托对象）来处理特定的任务（例如：验证文本、处理用户回车事件等）
  - 在相应的控制器对象中创建一个文本编辑框对象的引用。

定制内置键盘

- 当文本编辑框被点击后（获得焦点），系统内置键盘应该自动弹出到屏幕上，并且键盘的输入与该文本编辑框绑定。用户点击文本框的动作可以使其自动获得焦点。
- 当输入完信息后，可以从屏幕上去除内置键盘。
  - 实现去除内置键盘的方法是resignFirstResponder()。
  - 用户点击键盘的回车键或者键盘以外的其它非输入控件时，系统会调用该方法来去除内置键盘。
- 内置键盘的出现和消失会影响文本编辑框对象的编辑状态。
  - 当键盘出现到屏幕上的时候，文本编辑框对象进入到“正在编辑”状态，并且会发送消息给它的代理；
  - 当文本编辑框对象失去焦点的时候，它就结束了“正在编辑”状态，同时发消息通知其代理。
- 内置键盘的特征属性可以通过UITextField对象的属性UITextInputTraits来定制
  - 可以通过编码的方式来设置
  - 也可以通过属性面板来设置，设置的属性包括：键盘的类型（纯文本、数字、URL、电邮等）、自动校验、返回键的类型、自动大小写等等。

指定代理

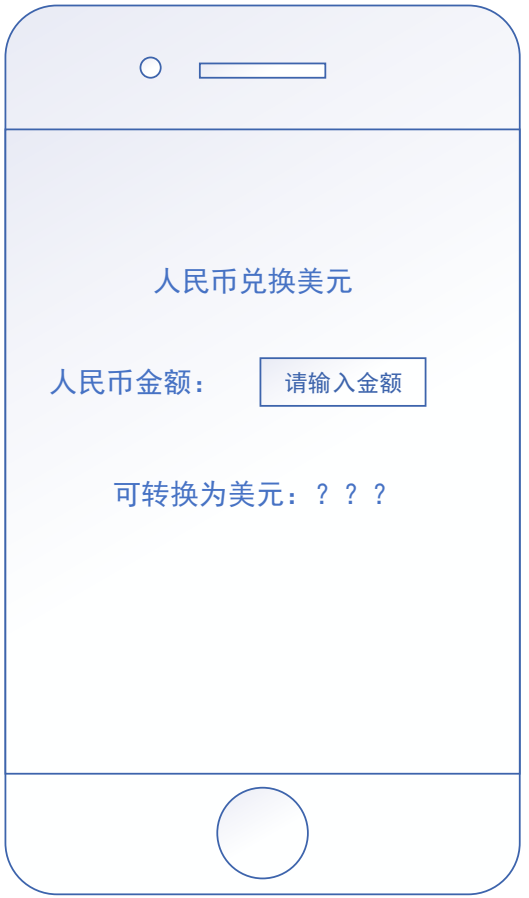
- 在iOS app开发中，委托模式Delegation是使用频率很高的一个设计模式。
- 代理中含有一个指向委托方的引用，并在特定事件发生的时候发消息给委托方。
- 委托方通常为一个Cocoa框架类的对象，而代理则常常为一个控制器对象。
- 要使一个控制器对象变成一个代理，则需要声明该控制器对象遵守相应的代理协议。
  - 例如：要成为UITextField的代理则需要遵守UITextFieldDelegate协议。
- UITextFieldDelegate协议由一系列的可选方法组成，通过这些方法可以实现对文本编辑框的编辑和验证。
- 文本编辑框调用它的代理中的方法来处理特定的事件：
  - 验证用户输入的文本的格式
  - 处理特定的键盘交互事件
  - 控制编辑动作的整个过程等。

UITextFieldDelegate协议中方法的调用过程

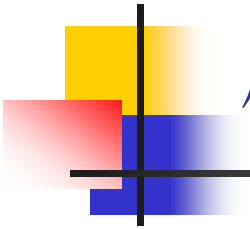
1. 当文本编辑框获得焦点前，文本编辑框会调用它的代理的方法textFieldShouldBeginEditing(_:)，用来许可或禁止对文本编辑框中内容进行改动。
2. 文本编辑框获得焦点。此时，系统会在界面上显示键盘，同时发送消息UIKeyboardWillShow、UIKeyboardDidShow。
3. 文本编辑框调用它的代理的方法textFieldDidBeginEditing(_:)，并发送消息UITextFieldTextDidBeginEditing。
4. 在文本编辑框编辑的过程中，会根据不同动作调用不同的方法来处理：
   - 当文本内容发生变化时，调用方法textField(_:shouldChangeCharactersIn:replacementString:)，并发送消息UITextFieldTextDidChange。
   - 当用户点击内置清除文本的按钮时，调用方法textFieldShouldClear(_:)。
   - 当用户点击键盘的返回按钮后，调用方法textFieldShouldReturn(_:)。
5. 在文本编辑框失去焦点的角色前，调用方法textFieldShouldEndEditing(_:)，用它来验证当前已经输入的文本。
6. 文本编辑框失去焦点。此时，系统隐藏起键盘，同时发送消息UIKeyboardWillHide和UIKeyboardDidHide。
7. 文本编辑框调用它的代理的方法textFieldDidEndEditing(_:)，并发送消息UITextFieldTextDidEndEditing。

汇率转换工具

- 要求设计一个汇率转换的小工具，输入人民币的金额，计算出可兑换的美元数额。
- 该应用只有一个界面，在界面上有三行文字，第一行文字显示标题“人民币兑换美元”。第二行提示用户人民币金额。第三行给出兑换后的美元具体金额。



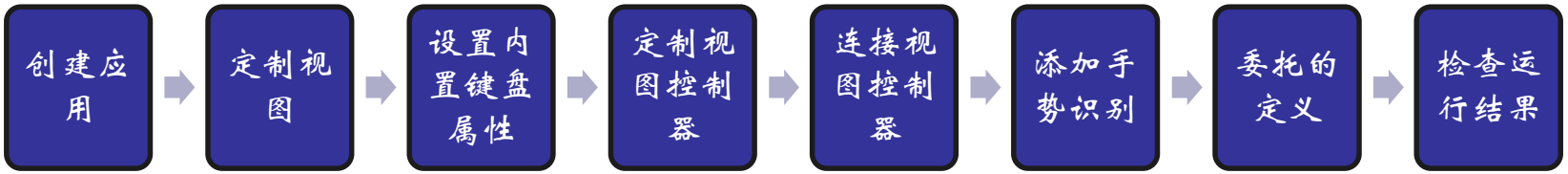
代码参见iosPrj的Chapter04-ExchangeRMBToDollar



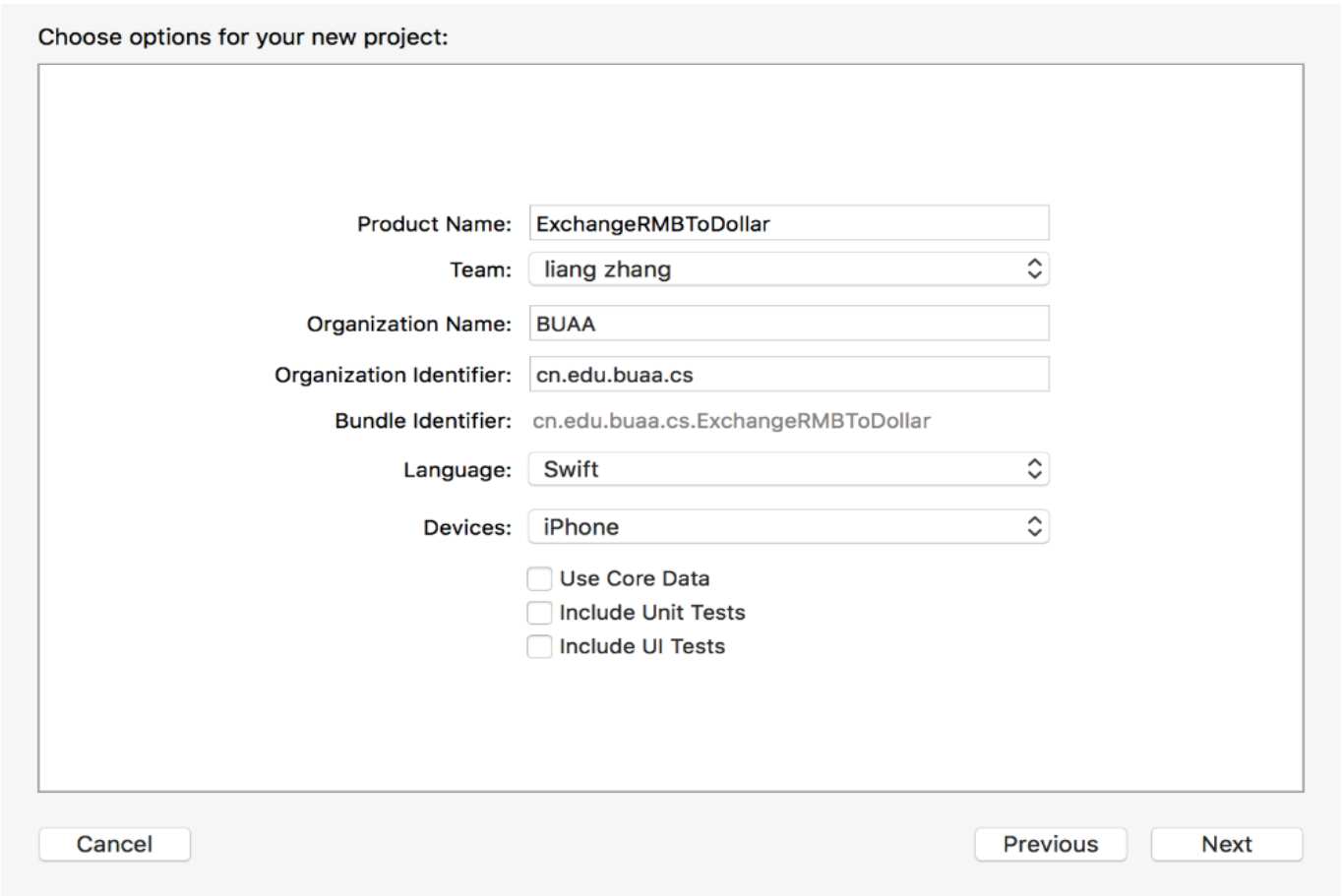
应用设计

- 构建用户界面
  - 用一个UITextField来接收用户输入的人民币金额
  - 用一个UILabel来显示兑换为美元的金额
  - 用三个UILabel来显示固定的提示信息（标题“人民币兑换美元”、提示信息“人民币金额：”、提示信息“可转换为美元：”）。
- 内置键盘的设置
  - 根据应用需要，应该弹出数字键盘，只能纯数字。
  - 在UITextField获得焦点时自动弹出键盘，在失去焦点时自动去除键盘。
- UITextField的委托定义
  - 建立UITextField与ViewController之间的委托关系，并声明ViewController遵守UITextFieldDelegate协议，然后根据应用的需要选择协议中相应的事件处理方法进行重载。

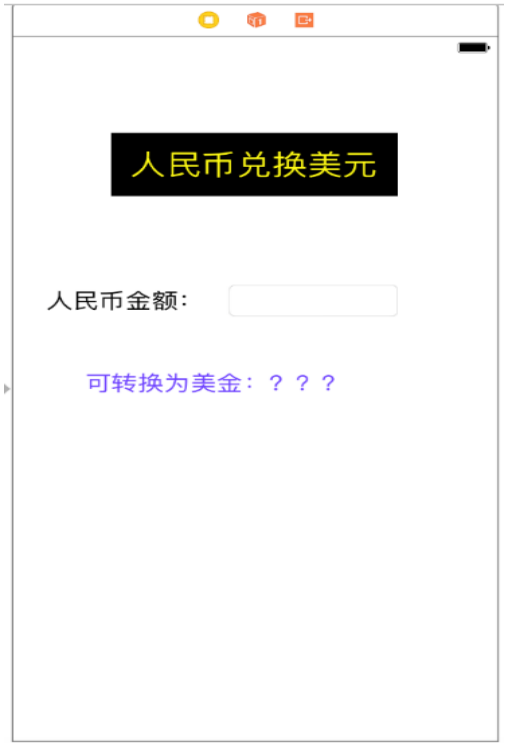
应用开发流程

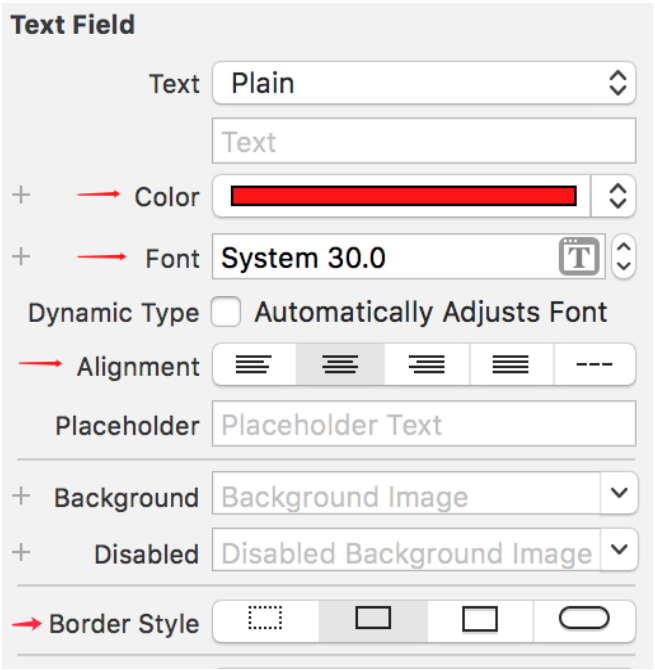


创建应用


Choose options for your new project:
Product Name: ExchangeRMBToDollar
Team: liang zhang
Organization Name: BUAA
Organization Identifier: cn.edu.buaa.cs
Bundle Identifier: cn.edu.buaa.cs.ExchangeRMBToDollar
Language: Swift
Devices: iPhone
Use Core Data
Include Unit Tests
Include UI Tests
Cancel
Previous
Next

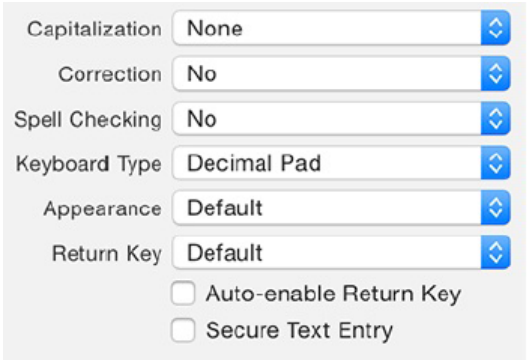
定制视图


人民币兑换美元
人民币金额:
可转换为美金：？？？


Text Field
Text
Plain
Text
Color
Font
System 30.0
Dynamic Type
Automatically Adjusts Font
Alignment
Placeholder
Placeholder Text
Background
Background Image
Disabled
Disabled Background Image
Border Style

设置内置键盘属性

- 应用运行时，点击UITextField获得焦点，此时屏幕中应该弹出一个数字键盘，用户可通过该键盘输入要计算的人民币金额。
- 设置UITextField控件的键盘属性UITextInputTraits来实现。
  - 选中UITextField，打开其属性面板，设置Correction为No、Spell Checking为No、设置Keyboard Type为Decimal Pad。

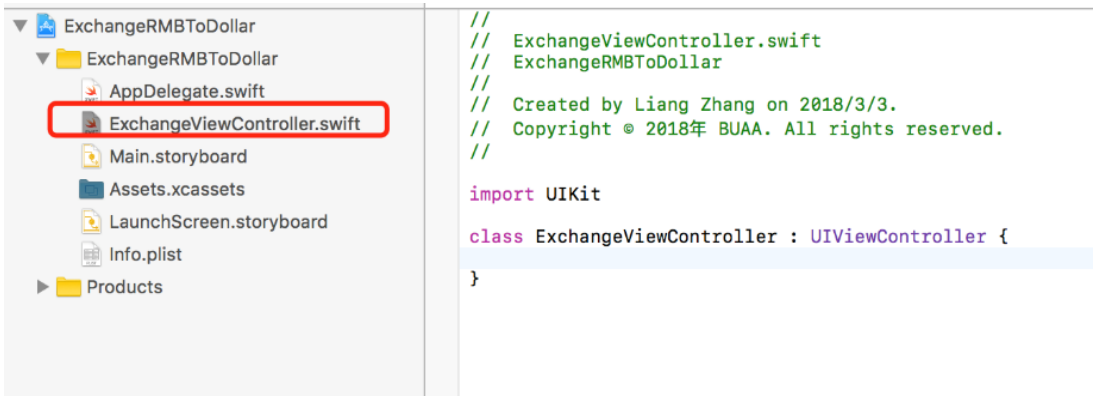


弹出键盘的运行效果


iPhone 7 – iOS 10.3 (14E8301)
Carrier
7:14 PM
人民币兑换美元
人民币：
请输入金额
可转换为美金：？？？
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
.
0

定制视图控制器

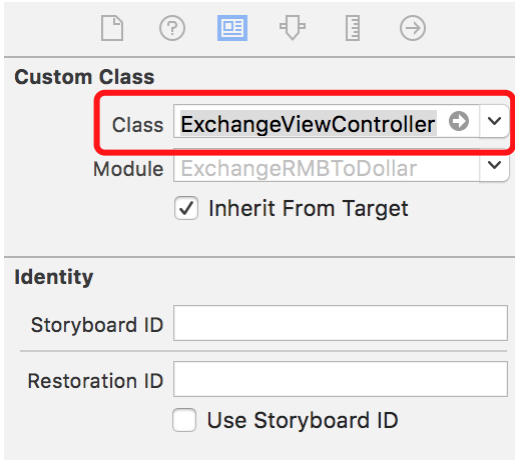
- 对于规模大一些的项目，特别是需要多个视图控制器的时候，需要自己定制视图控制器。
- 定制视图控制器：
  - 在项目文件树中找到ViewController.swift文件，将其删除。
  - 点击菜单栏中的“File”，在子菜单中选择“New File”，打开如图所示的文件选择窗口，选择“Swift File”
  - 按照步骤设置文件属性，最后点击“Create”，定制的视图控制器就创建完毕了。



代码参见iosPrj的Chapter04-ExchangeRMBToDollar

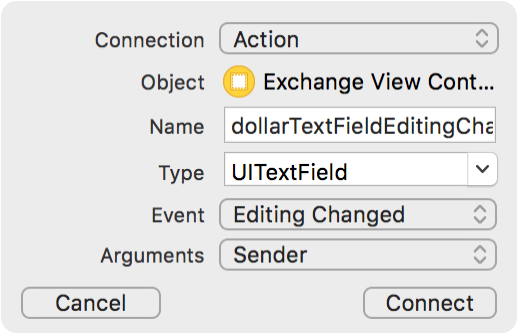
关联视图控制器与故事板

- 由single view application模板生成的缺省视图控制器ViewController和Main.storyboard中的ViewController是自动建立关联的。
- 定制的视图控制器ExchangeViewController与其对应的View需要手动关联：
  - 选中文件Main.storyboard，打开视图面板，选中View Controller，在右侧的属性面板中找到“Custom Class”栏中的“Class”属性，在下拉列表中选择“ExchangeViewController”。



连接UITextField与视图控制器

- 将界面中显示美元数额的UILabel和输入人民币数额的UITextField连接到视图控制器ExchangeViewController中。
- 还需要为UITextField建立一个Action Connection




```swift
//
//  ExchangeViewController.swift
//  ExchangeRMBToDollar
//
//  Created by Liang Zhang on 2018/3/3.
//  Copyright © 2018年 BUAA. All rights reserved.
//

import UIKit

class ExchangeViewController : UIViewController {


    @IBOutlet var dollarTextField: UITextField!

    @IBOutlet var dollarLabel: UILabel!


    @IBAction func dollarTextFieldEditingChanged(_ sender: UITextField) {

        if let input = dollarTextField.text, !input.isEmpty {
           dollarLabel.text = dollarTextField.text
        } else {
            dollarLabel.text = "???"
        }
    }

}
```

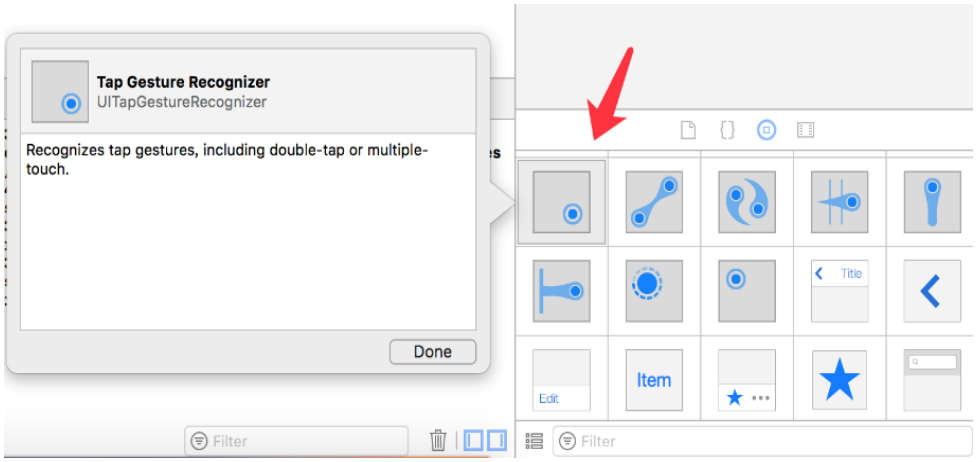
代码参见iosPrj的Chapter04-ExchangeRMBToDollar

手势识别器

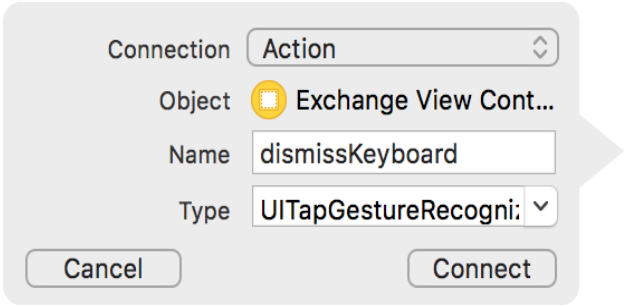
- 手势识别器可以检测交互界面中的触摸动作（包括：触碰、滑动、长按等等），并将其发送到相应的事件处理对象中，由对象中的相应函数来进行处理。
- 在本应用中，只需要检测文本编辑框以外区域的用户触碰动作。

添加手势识别

- 在运行过程中，存在一个bug：
  - 在UITextField中输入完人民币金额后，点击空白区域时，数字键盘一直在屏幕的下部，无法消除。
  - 正常情况下，用户输入信息时（即输入控件获得焦点），要立即弹出键盘，在输入完毕后，要自动消除，这样可以节省屏幕的显示空间。
- 为了实现这个功能，首先需要：
  - 向视图面板中加入一个手势识别器UIGestureRecognizer，用来捕捉视图中所有的手势。如图，在控件库中找到手势识别器UIGestureRecognizer，然后将其拖拽到视图面板中。
  - 连接手势识别器与视图控制器



连接手势识别器与视图控制器

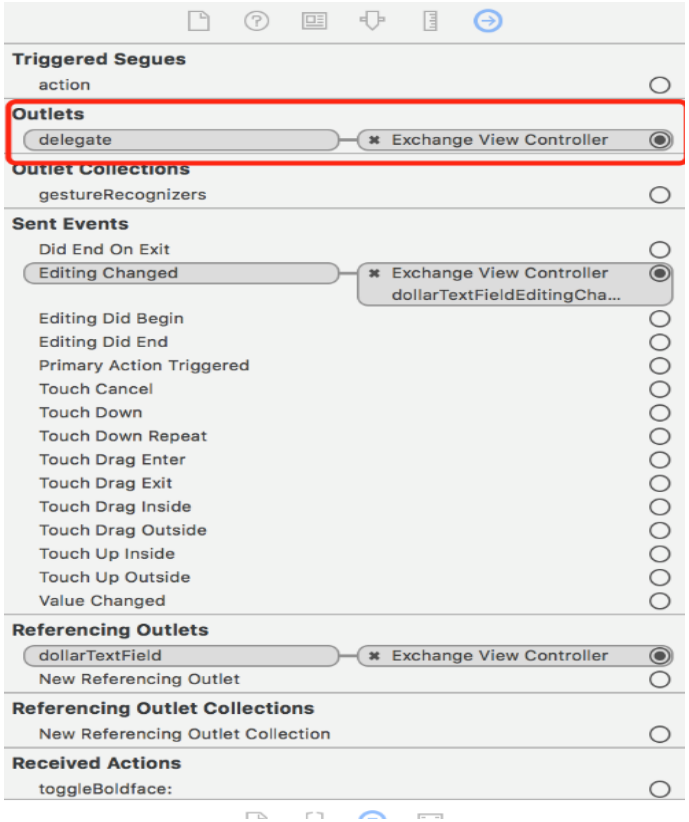



```
import UIKit

class ExchangeViewController : UIViewController {

    @IBAction func dismissKeyboard(_ sender: UITapGestureRecognizer) {

        dollarTextField.resignFirstResponder()

    }

    @IBOutlet var dollarTextField: UITextField!

    @IBOutlet var dollarLabel: UILabel!


    @IBAction func dollarTextFieldEditingChanged(_ sender: UITextField) {

        if let input = dollarTextField.text, !input.isEmpty {
           dollarLabel.text = dollarTextField.text
        } else {
            dollarLabel.text = "???"
        }
    }

}
```

代码参见iosPrj的Chapter04-ExchangeRMBToDollar

委托的定义

- 首先，指定ExchangeViewController为UITextField的delegate。
- 具体操作：
  - 在Main.storyboard中，选中UITextField，同时点击Ctrl键和鼠标左边不放，拖动连线到ExchangeViewController中，在弹出框中选择delegate。



委托的定义

- 接着，

要声明类ExchangeViewController遵守UITextFieldDelegate协议

```
import UIKit

class ExchangeViewController : UIViewController,UITextFieldDelegate {

    @IBAction func dismissKeyboard(_ sender: UITapGestureRecognizer) {

        dollarTextField.resignFirstResponder()

    }

    @IBOutlet var dollarTextField: UITextField!

    @IBOutlet var dollarLabel: UILabel!


    @IBAction func dollarTextFieldEditingChanged(_ sender: UITextField) {

        if let input = dollarTextField.text, let value = Double(input) {
           dollarLabel.text = "\(value*0.1577)"
        } else {
            dollarLabel.text = "???"
        }
    }

}
```

代码参见iosPrj的Chapter04-ExchangeRMBToDollar

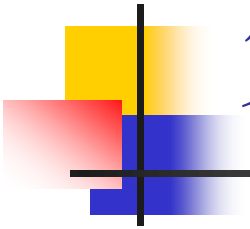
委托的定义

- 最后，需要在UITextFieldDelegate协议中找到相关的处理方法，并在类ExchangeViewController中进行重载。
  - 在本例中，为了禁止用户输入第二个小数点，可以选择协议中的方法textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool。
  - 该方法将自动获取textField中的当前显示的字符串和新输入即将显示的字符串。
  - 通过分别检查这两个字符串是否均含有小数点“.”，来判断当前输入是否合法。如果输入合法，则允许输入显示，并返回true。
  - 如果当前字符串中已含有小数点的情况下，新输入了一个小数点，则返回false。

```
import UIKit

class ExchangeViewController : UIViewController, UITextFieldDelegate {

    @IBAction func dismissKeyboard(_ sender: UITapGestureRecognizer) {

        dollarTextField.resignFirstResponder()

    }

    @IBOutlet var dollarTextField: UITextField!

    @IBOutlet var dollarLabel: UILabel!


    @IBAction func dollarTextFieldEditingChanged(_ sender: UITextField) {

        if let input = dollarTextField.text, let value = Double(input) {
           dollarLabel.text = "\(value*0.1577)"
        } else {
            dollarLabel.text = "???"
        }
    }

    func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String) -> Bool {

        let currentText = dollarTextField.text?.range(of: ".")
        let addingText = string.range(of: ".")

        if currentText != nil && addingText != nil {
            return false
        } else {
            return true
        }
    }
}
```

运行结果

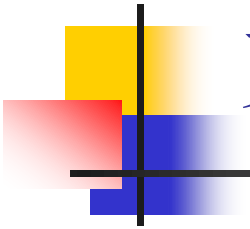

iPhone 7 – iOS 10.3 (14E8301)
Carrier
8:12 PM
人民币兑换美元
人民币：
1.22
可转换为美金：
0.192394
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
.
0



第五部分　控件

本章目的在于掌握文本编辑框、文本编辑区等控件以及容器视图控制器的相关知识，并通过多个应用实例来进行讲解。

- 文本编辑框
- 文本编辑区
- 选择控件等
- 进度显示控件
- 警告框控制器



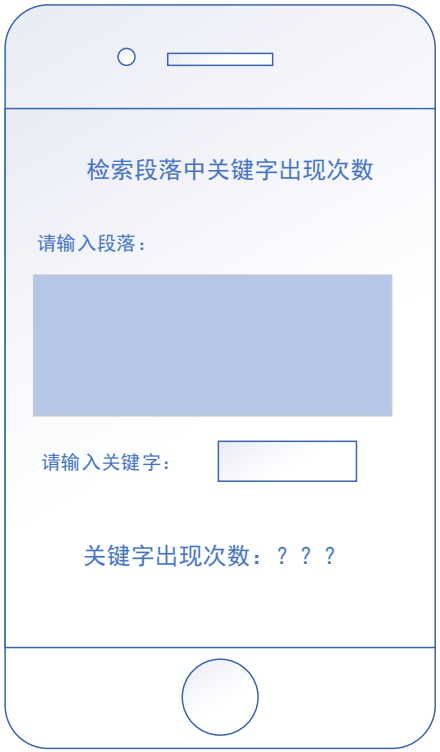
文本编辑区

- 文本编辑区，即：UITextView，是搭建用户界面的一种可编辑文本区。
- 它和UITextField很类似，区别在于UITextField为单行文本输入，而UITextView为多行文本输入。
- 这两种文本编辑控件有不同的应用场景需要：
  - UITextField可用于输入“用户民”、“密码”、“账号”、“邮箱”等信息。
  - UITextView可用于输入一段文字（多行的），如简介、说明性文字等。

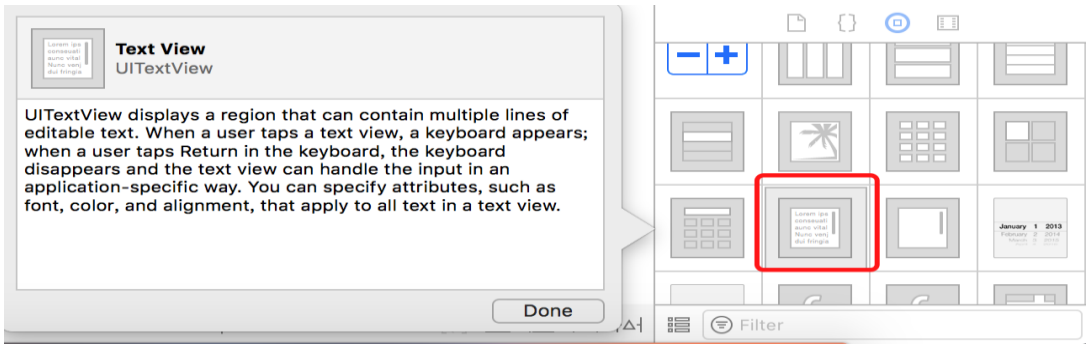
检索关键字应用

- 要求设计一个检索关键字的小工具，输入一个段落和待检索的关键字，计算出关键字在段落中出现的次数。
- 该应用的界面设计草图如图所示。
- 要求在UITextView中输入完段落后，回车可以直接切换到UITextField中继续输入关键字，然后回车后就给出检索结果。

代码参见iosPrj的Chapter04-SearchKeyword



定义视图


Text View
UITextView
UITextView displays a region that can contain multiple lines of editable text. When a user taps a text view, a keyboard appears; when a user taps Return in the keyboard, the keyboard disappears and the text view can handle the input in an application-specific way. You can specify attributes, such as font, color, and alignment, that apply to all text in a text view.
Done
Filter


检索段落中关键词出现次数
请输入段落：
请输入关键字：
关键字出现次数：？？？

建立连接关系

```
import UIKit

class SearchViewController: UIViewController {


    @IBOutlet var paragraphTextView: UITextView!

    @IBOutlet var keywordTextField: UITextField!

    @IBOutlet var resultLabel: UILabel!

    var keyword: String = " "

    var paragraph: String = " "


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
            typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

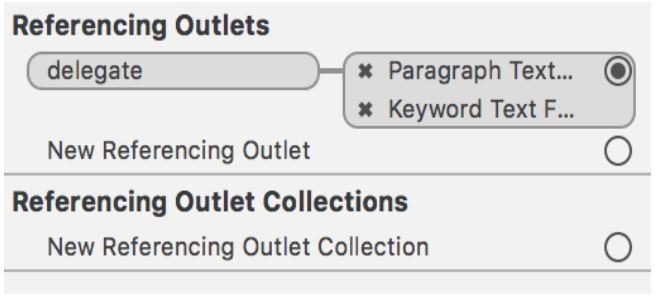
代码参见iosPrj的Chapter04-SearchKeyword

查找关键字的函数

- 在控制器类中编写一个输入段落和关键字，返回关键字出现次数的函数。
- 这里用到了系统函数components(separatedBy separator: String)，具体的用法可查阅苹果官方文档。

```
func searchKeyword(theString : String, theSubS : String) -> Int {

    let countsOfKeyword = theString.components(separatedBy: theSubS).count - 1
    return countsOfKeyword

}
```

定义委托




```
import UIKit

class SearchViewController: UIViewController, UITextFieldDelegate, UITextViewDelegate {

    @IBOutlet var paragraphTextView: UITextView!
    @IBOutlet var keywordTextField: UITextField!
    @IBOutlet var resultLabel: UILabel!

    var keyword: String = " "

    var paragraph: String = " "
```

重载委托协议中的方法

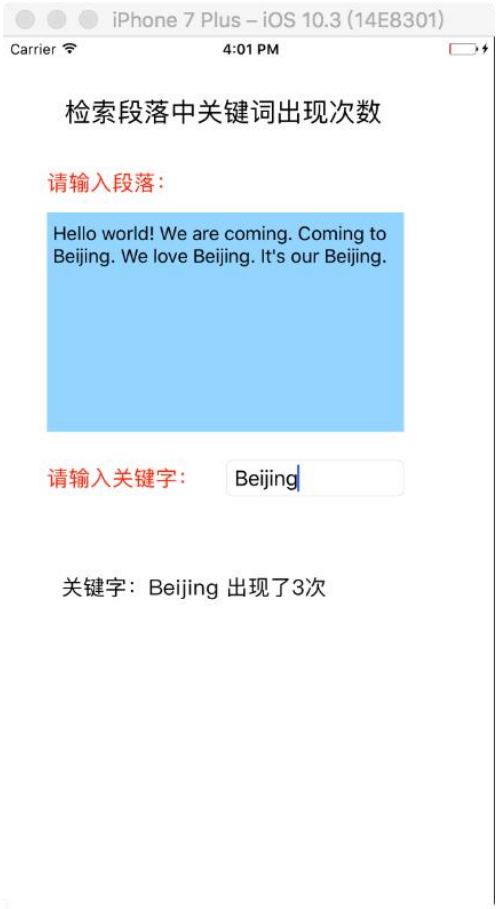
- 重载方法textView(_ textView: UITextView, shouldChangeTextIn range: NSRange, replacementText text: String) -> Bool
- 每次用户在UITextView中输入新的字符后，都会触发该方法。
- 当用户输入回车后，将焦点切换到关键字输入框。

```
import UIKit
class SearchViewController: UIViewController, UITextFieldDelegate, UITextViewDelegate {

    @IBOutlet var paragraphTextView: UITextView!
    @IBOutlet var keywordTextField: UITextField!
    @IBOutlet var resultLabel: UILabel!
    var keyword: String = " "
    var paragraph: String = " "
```

- 重载方法textFieldShouldReturn(_ textField: UITextField) -> Bool
- 当用户在UITextField中输入回车键，就会触发该方法。
- 先将当前的段落和关键字保存到变量keyword、paragraph中，再调用函数searchKeyword计算关键字在段落中出现的次数，最后将结果拼接成字符串赋值给resultLabel.text

```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    keyword = keywordTextField.text!
    paragraph = paragraphTextView.text!
    let counts = searchKeyword(theString: paragraph, theSubS: keyword)
    resultLabel.text = " 关键字: \(keyword) 出现了\(counts)次"
    return true
}
```

运行效果


iPhone 7 Plus – iOS 10.3 (14E8301)
Carrier
4:01 PM
检索段落中关键词出现次数
请输入段落：
Hello world! We are coming. Coming to Beijing. We love Beijing. It's our Beijing.
请输入关键字：
Beijing
关键字：Beijing 出现了3次

第五部分　控件

本章目的在于掌握文本编辑框、文本编辑区等控件以及容器视图控制器的相关知识，并通过多个应用实例来进行讲解。

- 文本编辑框
- 文本编辑区
- 选择控件等
- 进度显示控件
- 警告框控制器

选择控件

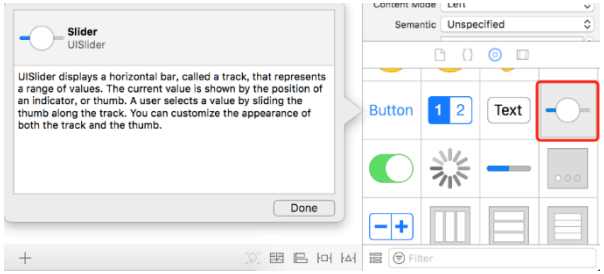
- 选择控件包括：UISwitch、UISegmentedControl、UISlider。
- UISwitch是一个提供二元选择的控件，即：On/Off。该控件有一个重要的属性“isOn”，用来表征二元状态。当用户在“On/Off”中做出选择后，会创建事“valueChanged”。可以编写相应的事件处理函数来相应该动作。
- UISegmentedControl是一个离散的多段选择控件。每一段都有一个对应的序号，序号是从0开始的整数。当用户在选择某个段，会创建事件“valueChanged”，同样可以编写相应的事件处理函数来相应该动作。
- UISlider是一个可以从连续的值中选择的控件。在控件使用时需要设置取值范围，即：最大值和最小值。另外，还要设置当前值，即滑块的水平缺省位置。当用户在控件中滑动滑块时，会不断产生 “valueChanged”事件，可以编写相应的事件处理函数来捕获该动作。

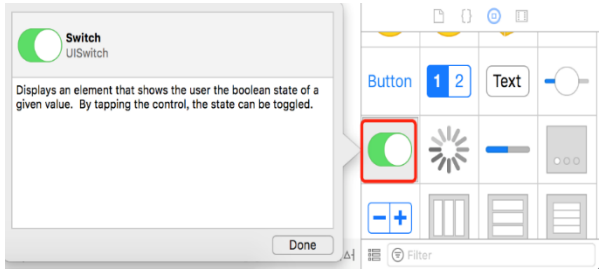
实例：选择控件的状态

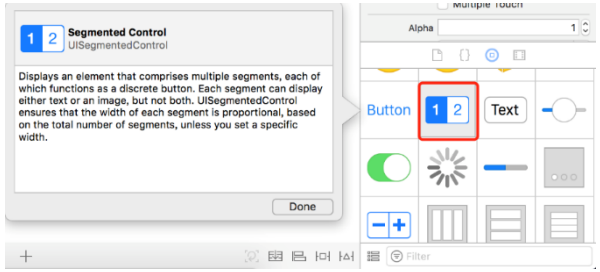
- 本例将同时运用三种选择控件：UISwitch、UISegmentedControl、UISlider。
- 要求：
  - 在一个应用中同时用到三个选择控件，并在控件后面显示该控件的当前状态。

代码参见iosPrj的Chapter04-ThreeControllers

在控件库中找到控件


Slider
UISlider
UISlider displays a horizontal bar, called a track, that represents a range of values. The current value is shown by the position of an indicator, or thumb. A user selects a value by sliding the thumb along the track. You can customize the appearance of both the track and the thumb.
Done
Semantic
Unspecified
Button
1 2
Text
Filter


Switch
UISwitch
Displays an element that shows the user the boolean state of a given value. By tapping the control, the state can be toggled.
Done
Button
1 2
Text
Filter


1 2
Segmented Control
UISegmentedControl
Displays an element that comprises multiple segments, each of which functions as a discrete button. Each segment can display either text or an image, but not both. UISegmentedControl ensures that the width of each segment is proportional, based on the total number of segments, unless you set a specific width.
Done
Alpha
1
Button
1 2
Text
Filter

定义视图

连接控件与视图控制器

```
class ViewController: UIViewController {
    @IBOutlet var theSwitch: UISwitch!
    @IBOutlet var theSegmented: UISegmentedControl!
    @IBOutlet var theSlider: UISlider!
    @IBOutlet var switchLabel: UILabel!
    @IBOutlet var segmentedLabel: UILabel!
    @IBOutlet var sliderLabel: UILabel!
```

代码参见iosPrj的Chapter04-ThreeControllers

创建UISwitch的动作处理函数

```
@IBAction func switchValueChanged(_ sender: Any) {

    let switchControl = sender as! UISwitch

    let status = switchControl.isOn

    if status {
        switchLabel.text = "开"
    } else {
        switchLabel.text = "关"
    }

}
```

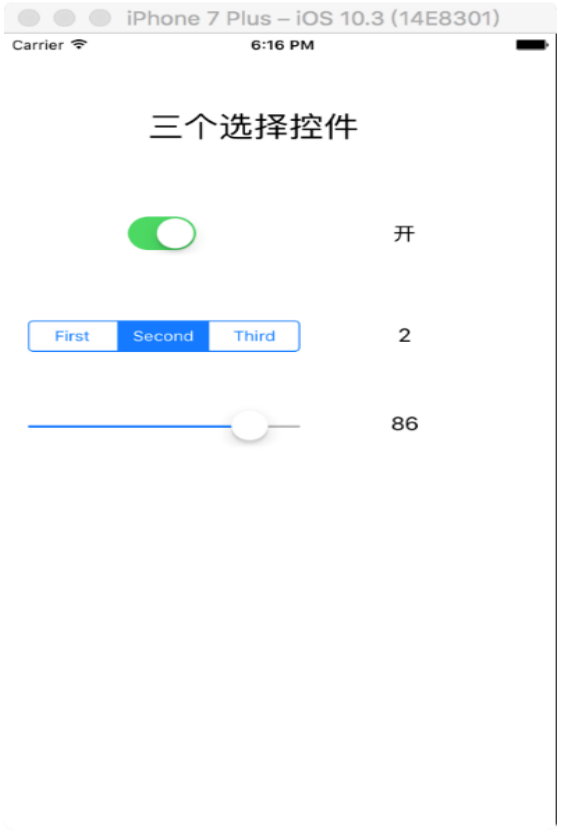
创建UISlider的动作处理函数

```
@IBAction func sliderValueChanged(_ sender: Any) {
    let sliderControl = sender as! UISlider
    let value = Int(sliderControl.value)
    sliderLabel.text = String(format: "%d", value)
}
```

创建UISegmentedControl的动作处理函数

```
@IBAction func segmentedValueChanged(_ sender: Any) {
    let segmentedControl = sender as! UISegmentedControl
    segmentedLabel.text = String(segmentedControl.selectedSegmentIndex + 1)
}
```

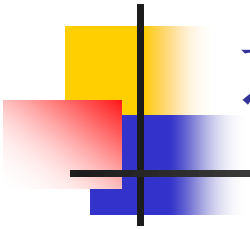
运行效果


iPhone 7 Plus – iOS 10.3 (14E8301)
Carrier
6:16 PM
三个选择控件
开
First
Second
Third
2
86

第五部分　控件

本章目的在于掌握文本编辑框、文本编辑区等控件以及容器视图控制器的相关知识，并通过多个应用实例来进行讲解。

- 文本编辑框
- 文本编辑区
- 选择控件等
- 进度显示控件
- 警告框控制器



进度显示控件

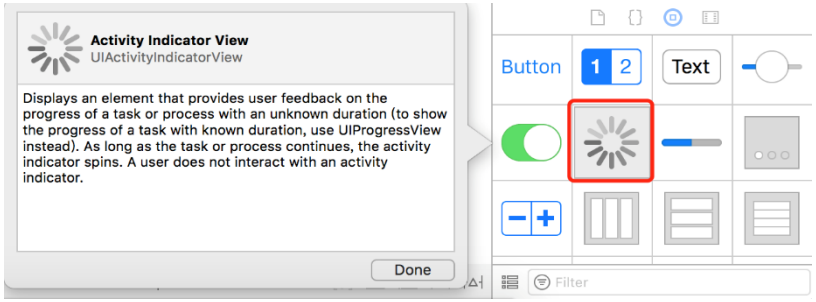
- 进度显示控件包括：进度条显示器UIProgressView和活动指示器UIActivityIndicatorView。
- 这两个控件都是用来显示后台有正在运行的程序。
- 活动指示器用于显示执行时间较短的程序正在执行，并不显示还有多长时间才能执行完。
- 进度条显示器一般会根据后台程序的执行情况，用动态进度条进展情况来表示还有多长时间执行完毕。

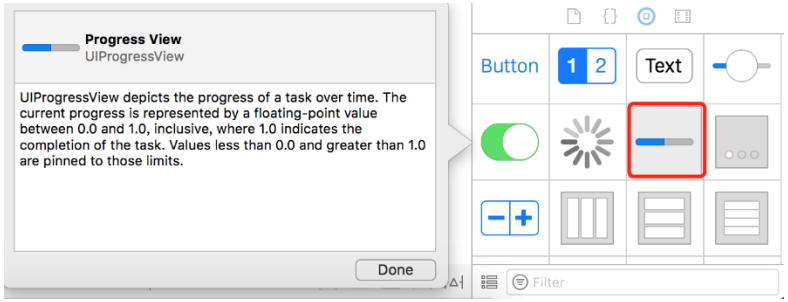
实例：等待时刻

- 本例将同时运用两种进度显示控件：UIActivityIndicatorView和UIProgressView。
- 要求：
  - 点击启动按钮后，两个显示进度的控件开始运转，运行一段时间后，停止运行。点击清零按钮后清零进度条，可以重新运转进度显示控件。

代码参见iosPrj的Chapter04-Waiting

在控件库中找到控件


Activity Indicator View
UIActivityIndicatorView
Displays an element that provides user feedback on the progress of a task or process with an unknown duration (to show the progress of a task with known duration, use UIProgressView instead). As long as the task or process continues, the activity indicator spins. A user does not interact with an activity indicator.
Done
Button
1
2
Text
Filter


Progress View
UIProgressView
UIProgressView depicts the progress of a task over time. The current progress is represented by a floating-point value between 0.0 and 1.0, inclusive, where 1.0 indicates the completion of the task. Values less than 0.0 and greater than 1.0 are pinned to those limits.
Done
Button
1
2
Text
Filter

定义视图

连接控件与视图控制器

```
import UIKit

class ViewController: UIViewController {


    @IBOutlet var theActivityIndicator: UIActivityIndicatorView!

    @IBOutlet var theProgress: UIProgressView!

    @IBAction func startButton(_ sender: Any) {

    }
```

代码参见iosPrj的Chapter04-Waiting

编写动作处理函数

- 为按钮“Go！”的点击动作编写处理函数

- selector调用的函数，每次执行增加progress的值0.01，当progress的值达到1时，停止计时器，然后停止活动指示器。

- 为按钮“Clear”的点击动作编写处理函数

```
var timer : Timer!

@IBAction func startButton(_ sender: Any) {

    if theProgress.progress == 0 {

        theActivityIndicator.startAnimating()

        timer = Timer.scheduledTimer(timeInterval: 0.05, target: self,
            selector: #selector(ViewController.going), userInfo: nil,
            repeats: true)

    }
}
```

```
func going() {

    theProgress.progress = theProgress.progress + 0.01

    if (theProgress.progress == 1.0) {

        timer.invalidate()

        theActivityIndicator.stopAnimating()

    }
}
```

```
  @IBAction func clearButton(_ sender: Any) {

      theProgress.progress = 0

  }
```

运行效果


iPhone 7 Plus – iOS 10.3 (14E8301)
Carrier
8:27 PM
等待时刻
活动状态显示
进度条显示
Go!
Clear

第五部分　控件

本章目的在于掌握文本编辑框、文本编辑区等控件以及容器视图控制器的相关知识，并通过多个应用实例来进行讲解。

- 文本编辑框
- 文本编辑区
- 选择控件等
- 进度显示控件
- 警告框控制器

警告框控制器

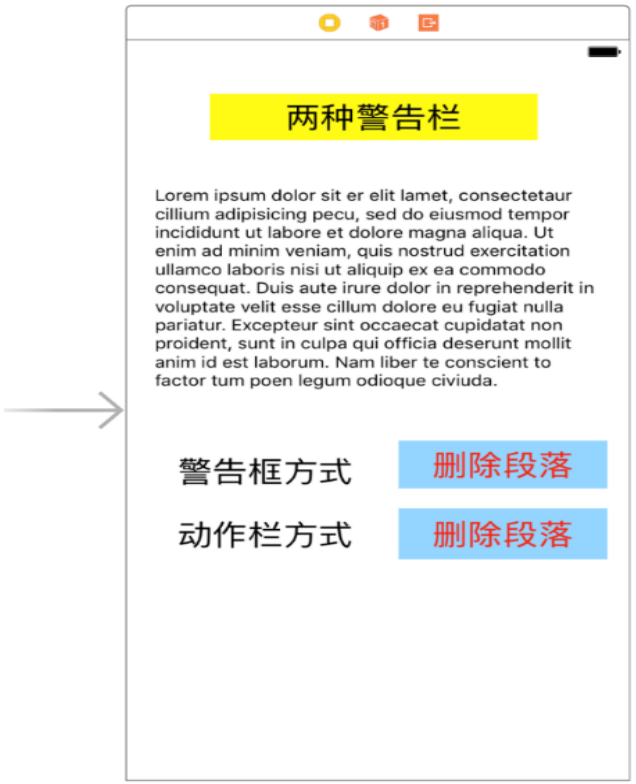
- 当应用需要向用户提供重要的信息，并由用户做出决定时，通常会使用警告控制器来实现。
- 警告控制器提供两种风格的警告信息显示：警告框（.alert）和动作栏（.actionSheet），可以通过警告控制器的属性preferedStyle来设置。
- 警告框和动作栏都是模态的显示信息确认框
  - 警告框是弹出的一个信息框，一般在应用界面的中间；
  - 动作栏则是由应用界面的底部向上弹出一个选项列表。
- 可以通过警告框或动作栏中的选项与具体的动作处理关联起来。当用户选择具体选项时就会调用相应的动作处理函数。可以通过方法addAction(_:)向警告控制器中添加动作。

实例：等待时刻

- 本例将同时运用两种警告风格：.alert和.actionSheet。
- 要求：
  - 应用的界面上有一个文字编辑框，缺省时有一段文字。有两个按钮分别用两种警告风格提示用户是否删除编辑框中的文字。

代码参见iosPrj的Chapter04-MyAlert

定义视图


两种警告栏
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.
警告框方式
删除段落
动作栏方式
删除段落

连接控件与视图控制器

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet var theTextView: UITextView!

    @IBAction func DeleteByAlert(_ sender: UIButton) {

    }

    @IBAction func DeleteByActionSheet(_ sender: UIButton) {

    }


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from
            a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

代码参见iosPrj的Chapter04-MyAlert

警告框动作处理函数

- 定制一个UIAlertController，设置属性：警告栏标题title、警告栏显示的信息message、以及显示风格UIAlertControllerStyle.alert（即：警告框风格）。
- 定制两个UIAlertAction，cancelAction表示“取消”和deleteAction表示“确定”。其中，deleteAction中要编写清空编辑框的代码。
- 将cancelAction和deleteAction添加到alertController中，然后显示该警告框。

```
@IBAction func DeleteByAlert(_ sender: UIButton) {

    let alertController: UIAlertController = UIAlertController(title: "警告
        框", message: "您确定要删除上面段落中的文字吗？", preferredStyle:
        UIAlertControllerStyle.alert)

    let cancelAction = UIAlertAction(title: "取消", style: .cancel)

    let deleteAction = UIAlertAction(title: "确定", style: .default) {
        (alertAction) -> Void in

        self.theTextView.text = ""

    }

    alertController.addAction(cancelAction)

    alertController.addAction(deleteAction)

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

}
```

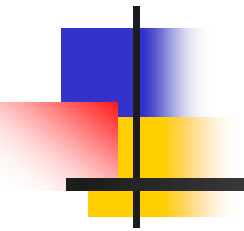
动作栏的动作处理函数

- 定制UIAlertController时将显示风格设置为UIAlertControllerStyle.actionSheet（即：动作栏风格）。

```
@IBAction func DeleteByActionSheet(_ sender: UIButton) {

    let actionSheetController = UIAlertController(title: "警告框", message:
        "您确定要删除上面段落中的文字吗？", preferredStyle:
        UIAlertControllerStyle.actionSheet)

    let cancelAction = UIAlertAction(title: "取消", style:
        UIAlertActionStyle.cancel)

    let deleteAction = UIAlertAction(title: "删除段落", style:
        UIAlertActionStyle.destructive) { (alertAction) -> Void in

        self.theTextView.text = ""

    }

    actionSheetController.addAction(cancelAction)

    actionSheetController.addAction(deleteAction)

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

}
```

运行效果


iPhone 7 Plus – iOS 10.3 (14E8301)
Carrier
4:54 PM
两种警告栏
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.
警告框方式
删除段落
动作栏方式
删除段落
警告框
您确定要删除上面段落中的文字吗?
取消
确定
删除段落
取消



谢　　谢