//

// ScrollViewExtension.swift

// SwiftPullToRefresh

//

// Created by Leo Zhou on 2017/12/19.

// Copyright © 2017年 Wiredcraft. All rights reserved.

//

import UIKit

private var headerKey: UInt8 = 0

private var footerKey: UInt8 = 0

private var tempFooterKey: UInt8 = 0

extension UIScrollView {

private var spr\_header: RefreshView? {

get {

return objc\_getAssociatedObject(self, &headerKey) as? RefreshView

}

set {

spr\_header?.removeFromSuperview()

objc\_setAssociatedObject(self, &headerKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

newValue.map { insertSubview($0, at: 0) }

}

}

private var spr\_footer: RefreshView? {

get {

return objc\_getAssociatedObject(self, &footerKey) as? RefreshView

}

set {

spr\_footer?.removeFromSuperview()

objc\_setAssociatedObject(self, &footerKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

newValue.map { insertSubview($0, at: 0) }

}

}

private var spr\_tempFooter: RefreshView? {

get {

return objc\_getAssociatedObject(self, &tempFooterKey) as? RefreshView

}

set {

objc\_setAssociatedObject(self, &tempFooterKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

}

}

/// Indicator header

///

/// - Parameters:

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setIndicatorHeader(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = IndicatorView(isHeader: true, height: height, action: action)

}

/// Indicator + Text header

///

/// - Parameters:

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setTextHeader(refreshText: RefreshText = headerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = TextView(isHeader: true, refreshText: refreshText, height: height, action: action)

}

/// GIF header

///

/// - Parameters:

/// - data: data for the GIF file

/// - isBig: whether the GIF is displayed with full screen width

/// - height: refresh view height and also the trigger requirement

/// - action: refresh action

public func spr\_setGIFHeader(data: Data,

isBig: Bool = false,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = GIFHeader(data: data, isBig: isBig, height: height, action: action)

}

/// GIF + Text header

///

/// - Parameters:

/// - data: data for the GIF file

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setGIFTextHeader(data: Data,

refreshText: RefreshText = headerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = GIFTextHeader(data: data, refreshText: refreshText, height: height, action: action)

}

/// Custom header

/// Inherit from RefreshView

/// Update the presentation in 'didUpdateState(\_:)' and 'didUpdateProgress(\_:)' methods

///

/// - Parameter header: your custom header inherited from RefreshView

public func spr\_setCustomHeader(\_ header: RefreshView) {

self.spr\_header = header

}

/// Custom footer

/// Inherit from RefreshView

/// Update the presentation in 'didUpdateState(\_:)' and 'didUpdateProgress(\_:)' methods

///

/// - Parameter footer: your custom footer inherited from RefreshView

public func spr\_setCustomFooter(\_ footer: RefreshView) {

self.spr\_footer = footer

}

/// Begin refreshing with header

public func spr\_beginRefreshing() {

spr\_header?.beginRefreshing()

}

/// End refreshing with both header and footer

public func spr\_endRefreshing() {

spr\_header?.endRefreshing()

spr\_footer?.endRefreshing()

}

/// End refreshing with footer and remove it

public func spr\_endRefreshingWithNoMoreData() {

spr\_tempFooter = spr\_footer

spr\_footer?.endRefreshing { [weak self] in

self?.spr\_footer = nil

}

}

/// Reset footer which is set to no more data

public func spr\_resetNoMoreData() {

if spr\_footer == nil {

spr\_footer = spr\_tempFooter

}

}

/// Indicator footer

///

/// - Parameters:

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setIndicatorFooter(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = IndicatorView(isHeader: false, height: height, action: action)

}

/// Indicator + Text footer

///

/// - Parameters:

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setTextFooter(refreshText: RefreshText = footerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = TextView(isHeader: false, refreshText: refreshText, height: height, action: action)

}

/// Indicator auto refresh footer (auto triggered when scroll down to the bottom of the content)

///

/// - Parameters:

/// - height: refresh view height, default is 60

/// - action: refresh action

public func spr\_setIndicatorAutoFooter(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = IndicatorAutoFooter(height: height, action: action)

}

/// Indicator + Text auto refresh footer (auto triggered when scroll down to the bottom of the content)

///

/// - Parameters:

/// - loadingText: text display for refreshing

/// - height: refresh view height, default is 60

/// - action: refresh action

public func spr\_setTextAutoFooter(loadingText: String = loadingText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = TextAutoFooter(loadingText: loadingText, height: height, action: action)

}

/// Clear the header

public func spr\_clearHeader() {

spr\_header = nil

}

/// Clear the footer

public func spr\_clearFooter() {

spr\_footer = nil

}

}

/// Text display for different states

public struct RefreshText {

let loadingText: String

let pullingText: String

let releaseText: String

/// Initialization method

///

/// - Parameters:

/// - loadingText: text display for refreshing

/// - pullingText: text display for dragging when don't reach the trigger

/// - releaseText: text display for dragging when reach the trigger

public init(loadingText: String, pullingText: String, releaseText: String) {

self.loadingText = loadingText

self.pullingText = pullingText

self.releaseText = releaseText

}

}

private let isChinese = Locale.preferredLanguages[0].contains("zh-Han")

public let loadingText = isChinese ? "正在加载..." : "Loading..."

public let headerText = RefreshText(

loadingText: loadingText,

pullingText: isChinese ? "下拉刷新" : "Pull down to refresh",

releaseText: isChinese ? "释放刷新" : "Release to refresh"

)

public let footerText = RefreshText(

loadingText: loadingText,

pullingText: isChinese ? "上拉加载" : "Pull up to load more",

releaseText: isChinese ? "释放加载" : "Release to load more"

)

//

// ScrollViewExtension.swift

// SwiftPullToRefresh

//

// Created by Leo Zhou on 2017/12/19.

// Copyright © 2017年 Wiredcraft. All rights reserved.

//

import UIKit

private var headerKey: UInt8 = 0

private var footerKey: UInt8 = 0

private var tempFooterKey: UInt8 = 0

extension UIScrollView {

private var spr\_header: RefreshView? {

get {

return objc\_getAssociatedObject(self, &headerKey) as? RefreshView

}

set {

spr\_header?.removeFromSuperview()

objc\_setAssociatedObject(self, &headerKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

newValue.map { insertSubview($0, at: 0) }

}

}

private var spr\_footer: RefreshView? {

get {

return objc\_getAssociatedObject(self, &footerKey) as? RefreshView

}

set {

spr\_footer?.removeFromSuperview()

objc\_setAssociatedObject(self, &footerKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

newValue.map { insertSubview($0, at: 0) }

}

}

private var spr\_tempFooter: RefreshView? {

get {

return objc\_getAssociatedObject(self, &tempFooterKey) as? RefreshView

}

set {

objc\_setAssociatedObject(self, &tempFooterKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

}

}

/// Indicator header

///

/// - Parameters:

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setIndicatorHeader(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = IndicatorView(isHeader: true, height: height, action: action)

}

/// Indicator + Text header

///

/// - Parameters:

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setTextHeader(refreshText: RefreshText = headerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = TextView(isHeader: true, refreshText: refreshText, height: height, action: action)

}

/// GIF header

///

/// - Parameters:

/// - data: data for the GIF file

/// - isBig: whether the GIF is displayed with full screen width

/// - height: refresh view height and also the trigger requirement

/// - action: refresh action

public func spr\_setGIFHeader(data: Data,

isBig: Bool = false,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = GIFHeader(data: data, isBig: isBig, height: height, action: action)

}

/// GIF + Text header

///

/// - Parameters:

/// - data: data for the GIF file

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setGIFTextHeader(data: Data,

refreshText: RefreshText = headerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = GIFTextHeader(data: data, refreshText: refreshText, height: height, action: action)

}

/// Custom header

/// Inherit from RefreshView

/// Update the presentation in 'didUpdateState(\_:)' and 'didUpdateProgress(\_:)' methods

///

/// - Parameter header: your custom header inherited from RefreshView

public func spr\_setCustomHeader(\_ header: RefreshView) {

self.spr\_header = header

}

/// Custom footer

/// Inherit from RefreshView

/// Update the presentation in 'didUpdateState(\_:)' and 'didUpdateProgress(\_:)' methods

///

/// - Parameter footer: your custom footer inherited from RefreshView

public func spr\_setCustomFooter(\_ footer: RefreshView) {

self.spr\_footer = footer

}

/// Begin refreshing with header

public func spr\_beginRefreshing() {

spr\_header?.beginRefreshing()

}

/// End refreshing with both header and footer

public func spr\_endRefreshing() {

spr\_header?.endRefreshing()

spr\_footer?.endRefreshing()

}

/// End refreshing with footer and remove it

public func spr\_endRefreshingWithNoMoreData() {

spr\_tempFooter = spr\_footer

spr\_footer?.endRefreshing { [weak self] in

self?.spr\_footer = nil

}

}

/// Reset footer which is set to no more data

public func spr\_resetNoMoreData() {

if spr\_footer == nil {

spr\_footer = spr\_tempFooter

}

}

/// Indicator footer

///

/// - Parameters:

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setIndicatorFooter(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = IndicatorView(isHeader: false, height: height, action: action)

}

/// Indicator + Text footer

///

/// - Parameters:

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setTextFooter(refreshText: RefreshText = footerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = TextView(isHeader: false, refreshText: refreshText, height: height, action: action)

}

/// Indicator auto refresh footer (auto triggered when scroll down to the bottom of the content)

///

/// - Parameters:

/// - height: refresh view height, default is 60

/// - action: refresh action

public func spr\_setIndicatorAutoFooter(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = IndicatorAutoFooter(height: height, action: action)

}

/// Indicator + Text auto refresh footer (auto triggered when scroll down to the bottom of the content)

///

/// - Parameters:

/// - loadingText: text display for refreshing

/// - height: refresh view height, default is 60

/// - action: refresh action

public func spr\_setTextAutoFooter(loadingText: String = loadingText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = TextAutoFooter(loadingText: loadingText, height: height, action: action)

}

/// Clear the header

public func spr\_clearHeader() {

spr\_header = nil

}

/// Clear the footer

public func spr\_clearFooter() {

spr\_footer = nil

}

}

/// Text display for different states

public struct RefreshText {

let loadingText: String

let pullingText: String

let releaseText: String

/// Initialization method

///

/// - Parameters:

/// - loadingText: text display for refreshing

/// - pullingText: text display for dragging when don't reach the trigger

/// - releaseText: text display for dragging when reach the trigger

public init(loadingText: String, pullingText: String, releaseText: String) {

self.loadingText = loadingText

self.pullingText = pullingText

self.releaseText = releaseText

}

}

private let isChinese = Locale.preferredLanguages[0].contains("zh-Han")

public let loadingText = isChinese ? "正在加载..." : "Loading..."

public let headerText = RefreshText(

loadingText: loadingText,

pullingText: isChinese ? "下拉刷新" : "Pull down to refresh",

releaseText: isChinese ? "释放刷新" : "Release to refresh"

)

public let footerText = RefreshText(

loadingText: loadingText,

pullingText: isChinese ? "上拉加载" : "Pull up to load more",

releaseText: isChinese ? "释放加载" : "Release to load more"

)

//

// ScrollViewExtension.swift

// SwiftPullToRefresh

//

// Created by Leo Zhou on 2017/12/19.

// Copyright © 2017年 Wiredcraft. All rights reserved.

//

import UIKit

private var headerKey: UInt8 = 0

private var footerKey: UInt8 = 0

private var tempFooterKey: UInt8 = 0

extension UIScrollView {

private var spr\_header: RefreshView? {

get {

return objc\_getAssociatedObject(self, &headerKey) as? RefreshView

}

set {

spr\_header?.removeFromSuperview()

objc\_setAssociatedObject(self, &headerKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

newValue.map { insertSubview($0, at: 0) }

}

}

private var spr\_footer: RefreshView? {

get {

return objc\_getAssociatedObject(self, &footerKey) as? RefreshView

}

set {

spr\_footer?.removeFromSuperview()

objc\_setAssociatedObject(self, &footerKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

newValue.map { insertSubview($0, at: 0) }

}

}

private var spr\_tempFooter: RefreshView? {

get {

return objc\_getAssociatedObject(self, &tempFooterKey) as? RefreshView

}

set {

objc\_setAssociatedObject(self, &tempFooterKey, newValue, .OBJC\_ASSOCIATION\_RETAIN\_NONATOMIC)

}

}

/// Indicator header

///

/// - Parameters:

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setIndicatorHeader(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = IndicatorView(isHeader: true, height: height, action: action)

}

/// Indicator + Text header

///

/// - Parameters:

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setTextHeader(refreshText: RefreshText = headerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = TextView(isHeader: true, refreshText: refreshText, height: height, action: action)

}

/// GIF header

///

/// - Parameters:

/// - data: data for the GIF file

/// - isBig: whether the GIF is displayed with full screen width

/// - height: refresh view height and also the trigger requirement

/// - action: refresh action

public func spr\_setGIFHeader(data: Data,

isBig: Bool = false,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = GIFHeader(data: data, isBig: isBig, height: height, action: action)

}

/// GIF + Text header

///

/// - Parameters:

/// - data: data for the GIF file

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setGIFTextHeader(data: Data,

refreshText: RefreshText = headerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_header = GIFTextHeader(data: data, refreshText: refreshText, height: height, action: action)

}

/// Custom header

/// Inherit from RefreshView

/// Update the presentation in 'didUpdateState(\_:)' and 'didUpdateProgress(\_:)' methods

///

/// - Parameter header: your custom header inherited from RefreshView

public func spr\_setCustomHeader(\_ header: RefreshView) {

self.spr\_header = header

}

/// Custom footer

/// Inherit from RefreshView

/// Update the presentation in 'didUpdateState(\_:)' and 'didUpdateProgress(\_:)' methods

///

/// - Parameter footer: your custom footer inherited from RefreshView

public func spr\_setCustomFooter(\_ footer: RefreshView) {

self.spr\_footer = footer

}

/// Begin refreshing with header

public func spr\_beginRefreshing() {

spr\_header?.beginRefreshing()

}

/// End refreshing with both header and footer

public func spr\_endRefreshing() {

spr\_header?.endRefreshing()

spr\_footer?.endRefreshing()

}

/// End refreshing with footer and remove it

public func spr\_endRefreshingWithNoMoreData() {

spr\_tempFooter = spr\_footer

spr\_footer?.endRefreshing { [weak self] in

self?.spr\_footer = nil

}

}

/// Reset footer which is set to no more data

public func spr\_resetNoMoreData() {

if spr\_footer == nil {

spr\_footer = spr\_tempFooter

}

}

/// Indicator footer

///

/// - Parameters:

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setIndicatorFooter(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = IndicatorView(isHeader: false, height: height, action: action)

}

/// Indicator + Text footer

///

/// - Parameters:

/// - refreshText: text display for different states

/// - height: refresh view height and also the trigger requirement, default is 60

/// - action: refresh action

public func spr\_setTextFooter(refreshText: RefreshText = footerText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = TextView(isHeader: false, refreshText: refreshText, height: height, action: action)

}

/// Indicator auto refresh footer (auto triggered when scroll down to the bottom of the content)

///

/// - Parameters:

/// - height: refresh view height, default is 60

/// - action: refresh action

public func spr\_setIndicatorAutoFooter(height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = IndicatorAutoFooter(height: height, action: action)

}

/// Indicator + Text auto refresh footer (auto triggered when scroll down to the bottom of the content)

///

/// - Parameters:

/// - loadingText: text display for refreshing

/// - height: refresh view height, default is 60

/// - action: refresh action

public func spr\_setTextAutoFooter(loadingText: String = loadingText,

height: CGFloat = 60,

action: @escaping () -> Void) {

spr\_footer = TextAutoFooter(loadingText: loadingText, height: height, action: action)

}

/// Clear the header

public func spr\_clearHeader() {

spr\_header = nil

}

/// Clear the footer

public func spr\_clearFooter() {

spr\_footer = nil

}

}

/// Text display for different states

public struct RefreshText {

let loadingText: String

let pullingText: String

let releaseText: String

/// Initialization method

///

/// - Parameters:

/// - loadingText: text display for refreshing

/// - pullingText: text display for dragging when don't reach the trigger

/// - releaseText: text display for dragging when reach the trigger

public init(loadingText: String, pullingText: String, releaseText: String) {

self.loadingText = loadingText

self.pullingText = pullingText

self.releaseText = releaseText

}

}

private let isChinese = Locale.preferredLanguages[0].contains("zh-Han")

public let loadingText = isChinese ? "正在加载..." : "Loading..."

public let headerText = RefreshText(

loadingText: loadingText,

pullingText: isChinese ? "下拉刷新" : "Pull down to refresh",

releaseText: isChinese ? "释放刷新" : "Release to refresh"

)

public let footerText = RefreshText(

loadingText: loadingText,

pullingText: isChinese ? "上拉加载" : "Pull up to load more",

releaseText: isChinese ? "释放加载" : "Release to load more"

)