Lecture 6 Demo Code:

FaceIt Multi-MVC & VCL

Objective

Included below is the source code for the demo in lecture. It is provided under the same Creative Commons licensing as the rest of CS193p's course materials. The code for the `FaceViewController` MVC (including `FacialExpression` and `FaceView`) is not included here (see Lecture 5 FaceIt Demo Code document). And here is the complete project.

```
//
//  EmotionsViewController.swift
//  FaceIt
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit

class EmotionsViewController: VCLLoggingViewController
{
    // MARK: - Navigation

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        var destinationViewController = segue.destination
        if let navigationController = destinationViewController as? UINavigationController {
            destinationViewController = navigationController.visibleViewController ?? destinationViewController
        }
        if let faceViewController = destinationViewController as? FaceViewController,
            let identifier = segue.identifier,
            let expression = emotionalFaces[identifier] {
            faceViewController.expression = expression
            faceViewController.navigationItem.title = (sender as? UIButton)?.currentTitle
        }
    }

    private let emotionalFaces: Dictionary<String,FacialExpression> = [
        "sad" : FacialExpression(eyes: .closed, mouth: .frown),
        "happy" : FacialExpression(eyes: .open, mouth: .smile),
        "worried" : FacialExpression(eyes: .open, mouth: .smirk)
    ]
}
```

```
//
//  VCLLoggingViewController.swift
//
//  Created by CS193p Instructor.
//  Copyright © 2015-17 Stanford University. All rights reserved.
//

import UIKit

class VCLLoggingViewController : UIViewController
{
    private struct LogGlobals {
        var prefix = ""
        var instanceCounts = [String:Int]()
        var lastLogTime = Date()
        var indentationInterval: TimeInterval = 1
        var indentationString = "__"
    }

    private static var logGlobals = LogGlobals()

    private static func logPrefix(for className: String) -> String {
        if logGlobals.lastLogTime.timeIntervalSinceNow < -logGlobals.indentationInterval {
            logGlobals.prefix += logGlobals.indentationString
            print("")
        }
        logGlobals.lastLogTime = Date()
        return logGlobals.prefix + className
    }

    private static func bumpInstanceCount(for className: String) -> Int {
        logGlobals.instanceCounts[className] = (logGlobals.instanceCounts[className] ?? 0) + 1
        return logGlobals.instanceCounts[className]!
    }

    private var instanceCount: Int!

    private func logVCL(_ msg: String) {
        let className = String(describing: type(of: self))
        if instanceCount == nil {
            instanceCount =  VCLLoggingViewController.bumpInstanceCount(for: className)
        }
        print("\(VCLLoggingViewController.logPrefix(for: className))(\(instanceCount!)) \(msg)")
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        logVCL("init(coder:) - created via InterfaceBuilder ")
    }

    override init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
        super.init(nibName: nibNameOrNil, bundle: nibBundleOrNil)
        logVCL("init(nibName:bundle:) - create in code")
    }

    deinit {
        logVCL("left the heap")
    }

    override func awakeFromNib() {
        logVCL("awakeFromNib()")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        logVCL("viewDidLoad()")
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        logVCL("viewWillAppear(animated = \(animated))")
    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
        logVCL("viewDidAppear(animated = \(animated))")
    }
```

```
    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)
        logVCL("viewWillDisappear(animated = \(animated))")
    }
    override func viewDidDisappear(_ animated: Bool) {
        super.viewDidDisappear(animated)
        logVCL("viewDidDisappear(animated = \(animated))")
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        logVCL("didReceiveMemoryWarning()")
    }

    override func viewWillLayoutSubviews() {
        super.viewWillLayoutSubviews()
        logVCL("viewWillLayoutSubviews() bounds.size = \(view.bounds.size)")
    }
    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        logVCL("viewDidLayoutSubviews() bounds.size = \(view.bounds.size)")
    }

    override func viewWillTransition(to size: CGSize, with coordinator: UIViewControllerTransitionCoordinator) {
        super.viewWillTransition(to: size, with: coordinator)
        logVCL("viewWillTransition(to: \(size), with: coordinator)")
        coordinator.animate(alongsideTransition: {
        (context: UIViewControllerTransitionCoordinatorContext!) -> Void in
            self.logVCL("begin animate(alongsideTransition:completion:)")
        }, completion: { context -> Void in
            self.logVCL("end animate(alongsideTransition:completion:)")
        })
    }
}
```