Cocoa aplikace pro macOS

IZA, Martin Hrubý, FIT VUT, 2018

Historie macOS, OS X, …

- OS X (X— 10, ”ten”), dnes macOS. Vychází z NextSTEPu.
  - Březen 2001. Darwin, BSD, Mach, NextSTEP.
- Foundation. AppKit (Cocoa, Core * knihovny).

Architektura aplikace

- Aplication, AppDelegate.
- NSView, NSWindow (není jako UIWindow),
  - controllers,
  - story board.
- Split Views — vetší rozměr okna, dynamická velikost, rozsáhlejší obsah
  - horizontální, vertikální split view.
- Apple Bindings.

Koncept aplikace v macOS

- Horní lišta s MENU.
- Okna. Extension. Ikonka v horní liště.
- Model-View-Controller znova:
  - aplikace nespojuje činnost s oknem (vlákno a okno).
  - aplikace ani nepotřebuje otevřená okna.
  - činnosti (vlákna, operace) jsou globální prvky.
  - okna s view pouze nahlíží do vnitřního stavu aplikace.
  - extensions, helpers (XPC).

Není tlačítko ”Save”

- Respektovat zvyklost.
- Formuláře, okna. Aplikace ukládá, co uživatel v programu zadá. Není explicitní ”save”.
  - UndoManagement
- Pokud chceme explicitní save, pak okno vytvářet nad dočasnou strukturou, pak provést zápis do dat aplikace.

Stále platná funkcionalita

- Foundation:
  - CoreData, CloudKit, Documents.
  - GCD, DispatchQueue, NSOperation.
  - UserDefaults, FileManager, …
- Knihovny přistupující na uživatelská data (kalendář, kontakty, připomínky, poznámky).
- Sandboxing — je povinný (pro AppStore) celkem krátce.

Stále platná funkcionalita

- Principy z Foundation známe z iOS.
- Cocoa Touch, Cocoa.
  - NSView, souřadný systém je obráceně.
  - Frame, Bounds. SubView/SuperView.
  - StoryBoard. Segue.
- Sloučí se někdy iOS a macOS?

Apple Bindings

- *Foundation* versus *Swift Standard Library*.
- Key-Value Observing (KVO), NSObject.
  - willChangeValue(forKey:)
  - didChangeValue(forKey:)
  - properties — termín s odlišným významem Objective-C/Swift.
  - NSObject, Key-Value Coding.
- Demo.

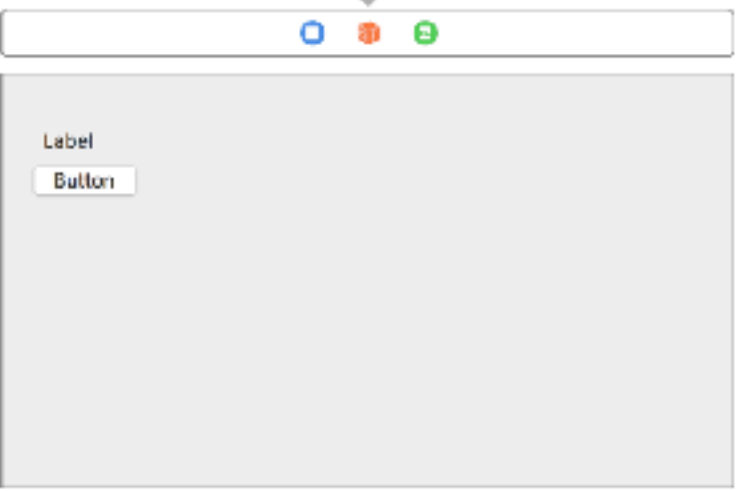
Apple Bindings

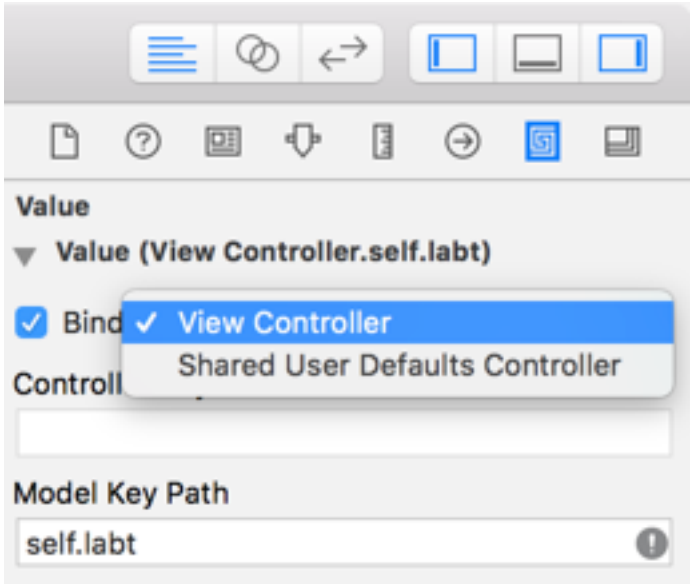
- Objekty Modelu. Objekty View (TextField, apod)
- Potřebujeme přenášet obsah z Model do View a !!!! to i zpětně.
- Manuální/automatizované řešení.
- Cílový objekt implementuje value(forKey)/ setValue(forKey) a dostává zprávy z View

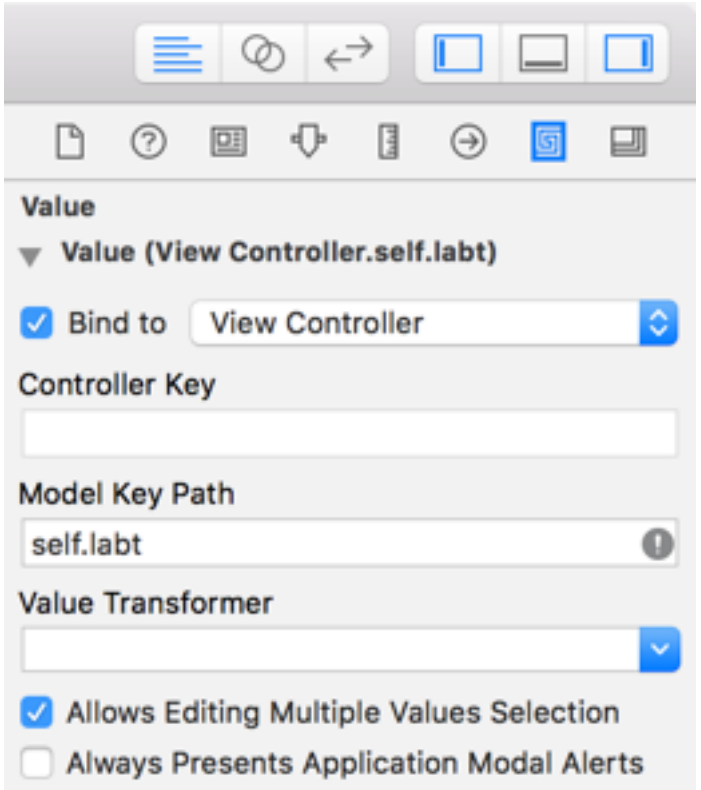
KVC/KVO v Objective-C

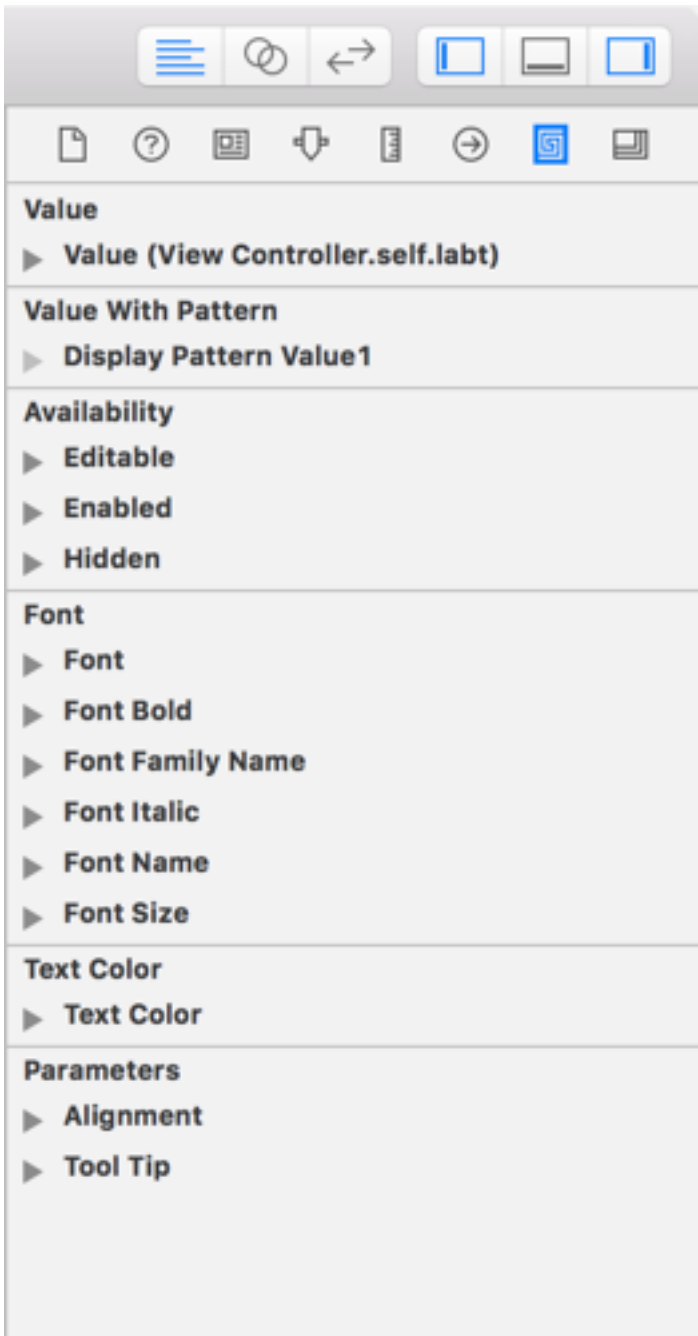
- NSObject. Implementuje KVC/KVO. Implementuje objektový meta-protokol.
- @property, @synthesize, @dynamic
- willChange/didChange:
  - coding — valueForKey, setValueForKey
  - observing.
- Swift — NSObject, @objc
  - @objc var cosi: Type — jinak objekt neprovádí KVC

Apple Bindings, Demo









Apple binding, demo

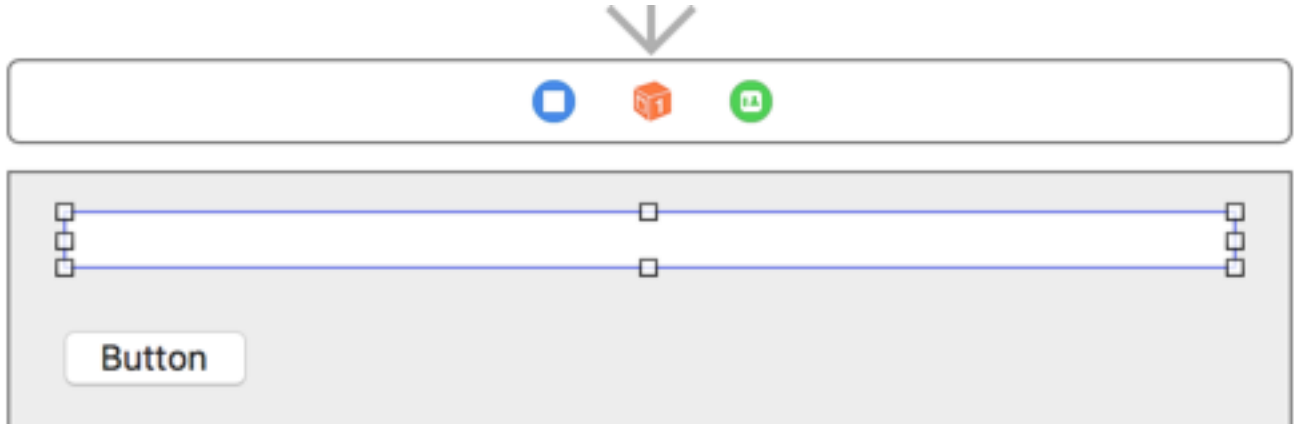
- IBOutlet se stává *observerem view-controlleru,* sleduje jeho key-path (zde ”labt”).
- Událost změny generuje ViewController s danou keyPath.
- V Objective-C se to odehrává automaticky (properties)

Apple bindings, demo

```
//
class ViewController: NSViewController {
    // property musi byt typu @objc
    @objc var labt: String = "dd"

    // nastaveni hodnoty modelu, musi se vsak
    // EXPLICITNE na sledovanem objektu volat will/did
    // Pozn.: v Objective-C automaticky
    @IBAction func butt(_:AnyObject) {
        // self je VC
        self.willChangeValue(forKey: "labt")
        self.labt = "novy Text"
        self.didChangeValue(forKey: "labt")
    }
```

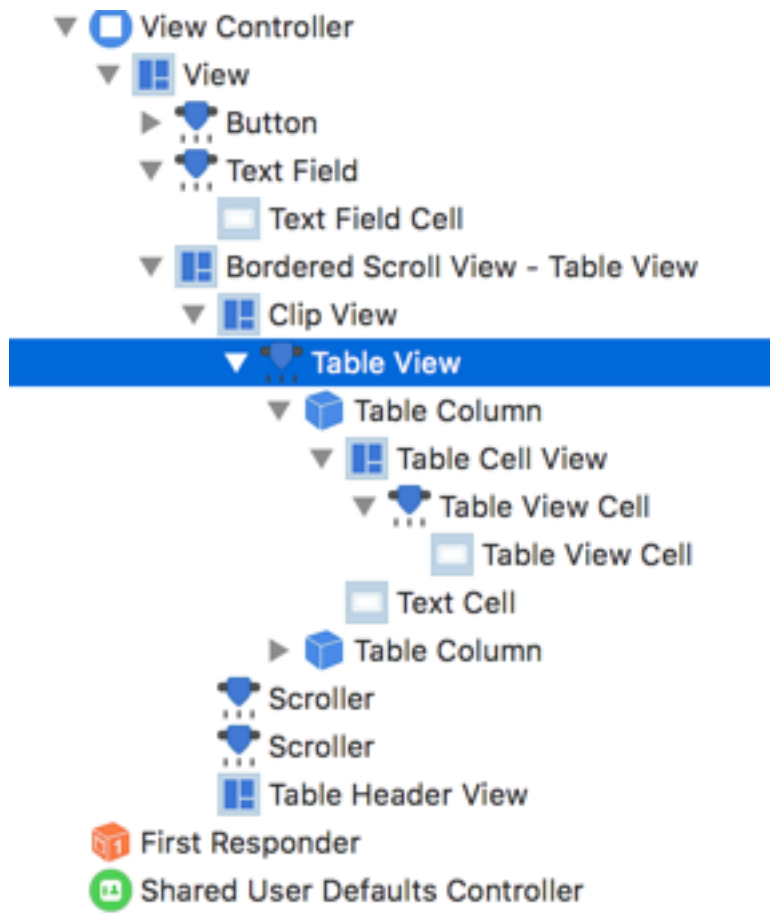
Apple Bindings ve velkém



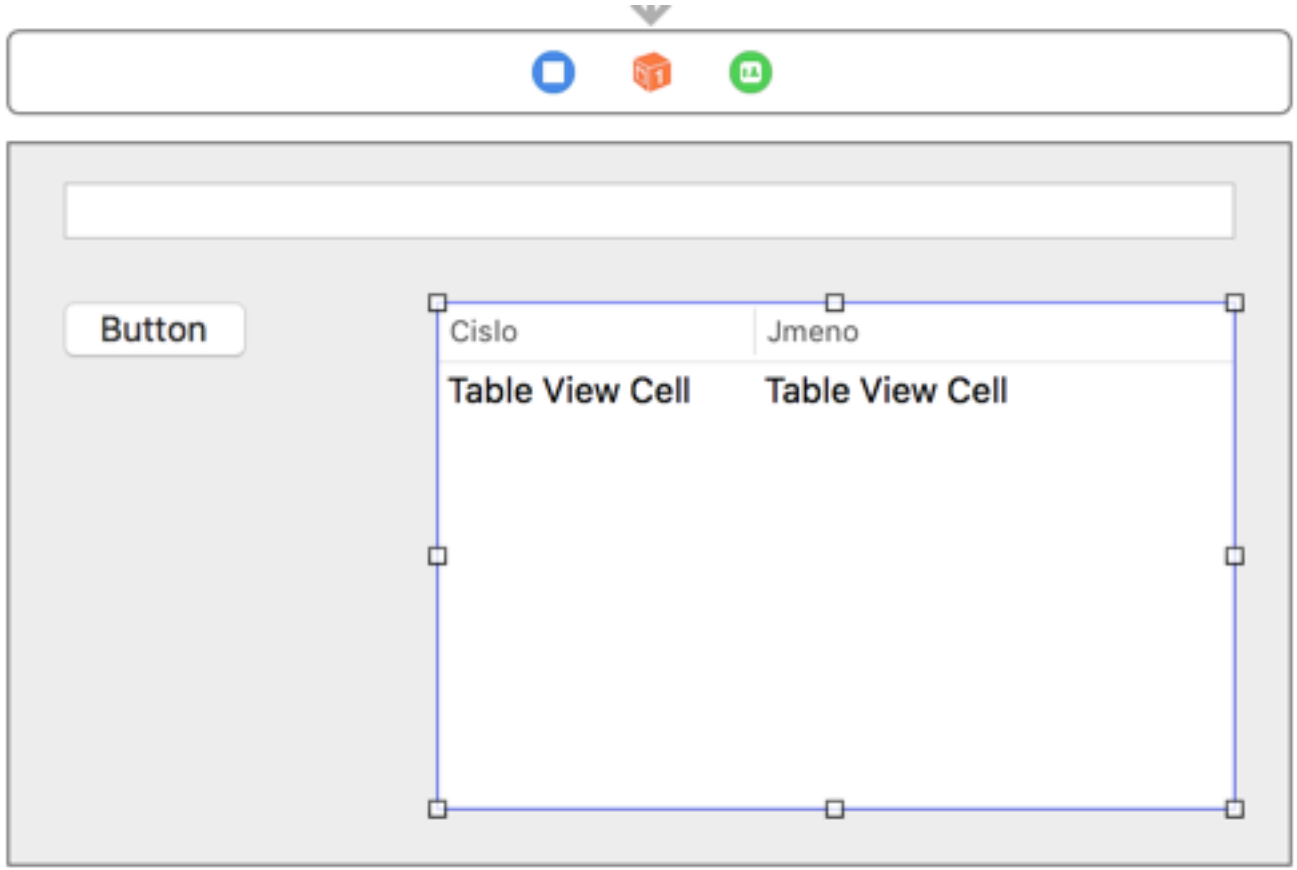
- UI prvek, který je observerem nějakého modelu, chceme obousměrnou komunikaci:
  - z modelu do UI (NSTextField),
  - z UI do modelu.
- Tabulky — dataSource a bindings.
  - NSArrayController

Tabulky

- Mají sloupce, nadpisek sloupce, prototyp buňky pro každý sloupec (formatter, KVO).
- DataSource — manuální, automatizovaný.
- Není explicitní TableViewController.



Column: název, identifier



Metodika VC nad tabulkami

- Potřebujeme definovat dataSource, delegate na tabulku, tj. každá jednotlivá tabulka potřebuje nějaký dedikovaný kontroler.
- Ideálně si přejeme znovupoužitelnost kódu.
- Spojit tabulku (View) s vlastním VC, dále skládat V/VC přes splitView.

DataSource manuální

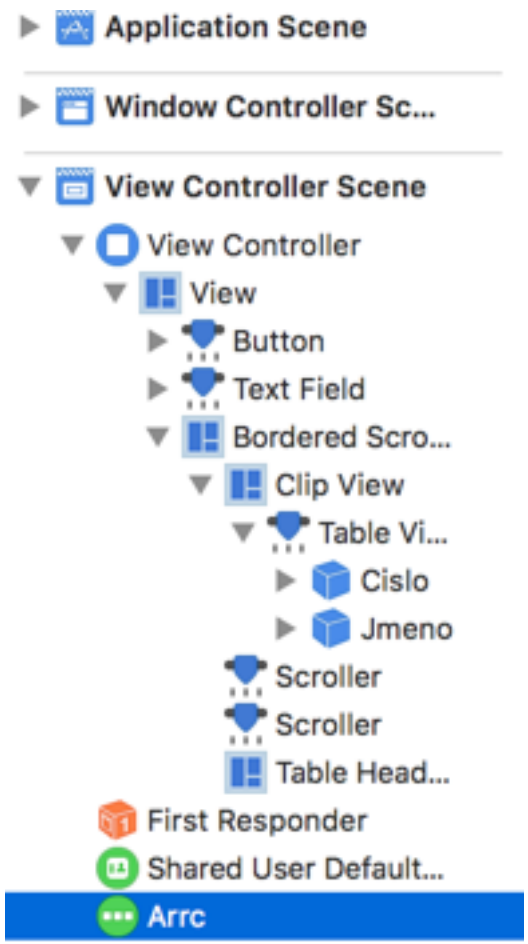
- Počet řádků (sekce implicitně nejsou).
- Varianty:
  - Objekt hodnoty pro řádek/sloupec.
  - Objekt view pro vložení do tableViewCell.
  - Tabulka napojuje dataSource, delegeta

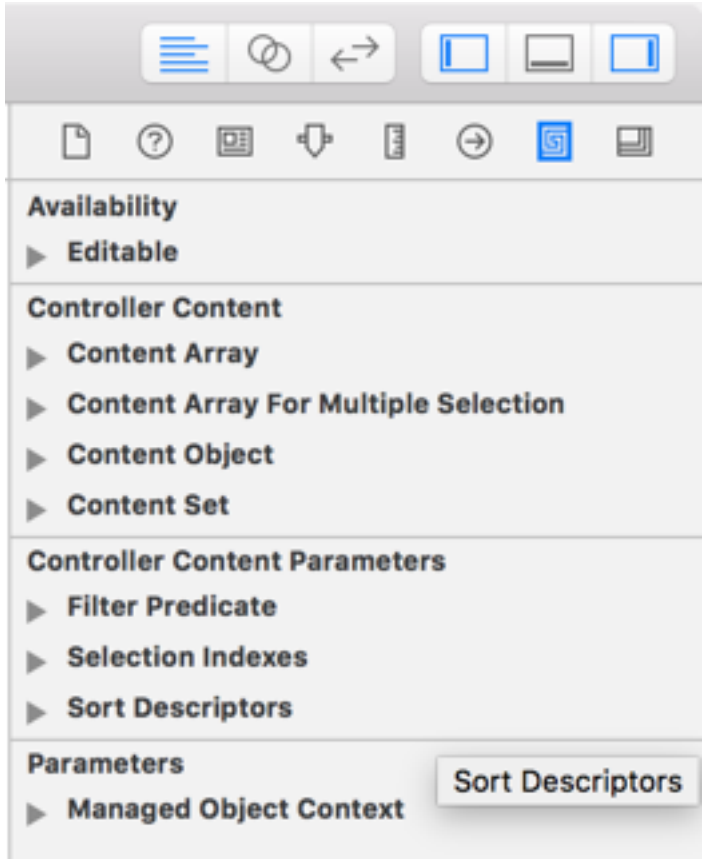
```
// tabulka vyzaduje zadani dataSource i delegate
class ViewController: NSViewController, NSTableViewDataSource,
NSTableViewDelegate
{
    //
    func numberOfRows(in tableView: NSTableView) -> Int {
        //
        return 10
    }
    // driv fungovalo...
    func tableView(_ tableView: NSTableView, objectValueFor tableColumn:
NSTableColumn?, row: Int) -> Any?
    {
        //
        return "hello"
    }
    // funguje
    func tableView(_ tableView: NSTableView,
                   viewFor tableColumn: NSTableColumn?,
                   row: Int) -> NSView?
    {
        let result  = tableView.makeView(withIdentifier:
(tableColumn?.identifier)!, owner: self) as! NSTableCellView
        //
        result.textField?.stringValue = "Hello"
        return result
    }
```

DataSource, ArrayController

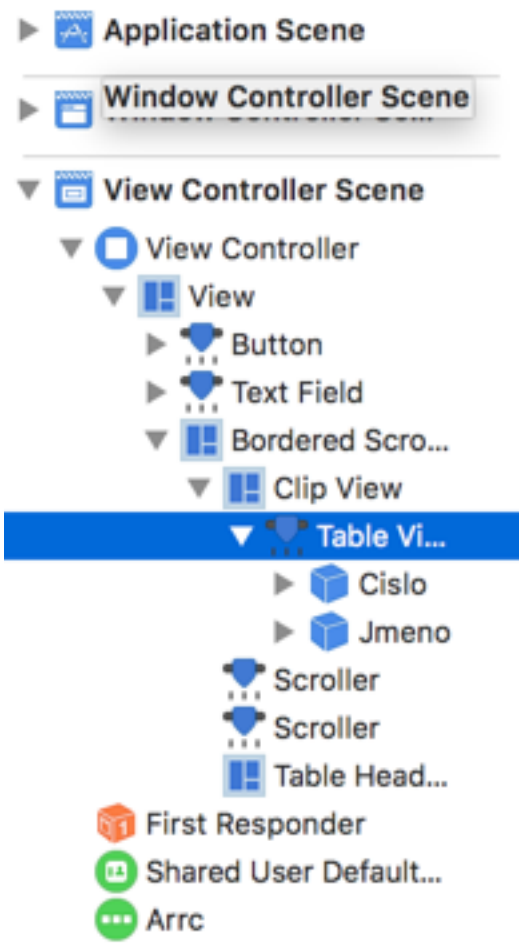
- ArrayController — obdoba/originál NSFetchedResultsController
  - lze vložit libovolná data (včetně dotazu nad CD — Entity, FRC), uspořádání.
- Tabulka nepracuje s dataSource, nýbrž s Apple Bindings.
  - zdroj tabulky se napojí na ARRC (počet řádků, N-tý řádek).
  - s Bindings je tabulka editovatelná (!!) se zápisem do modelu.

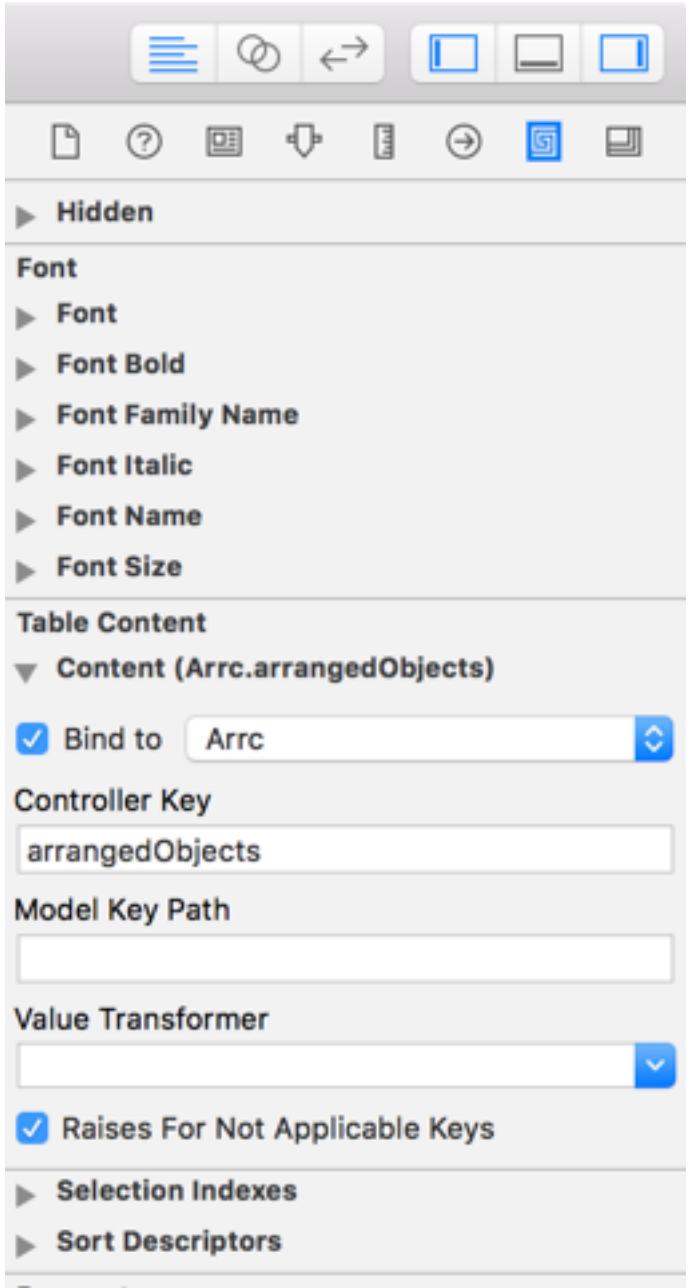
Table, ArrC, demo



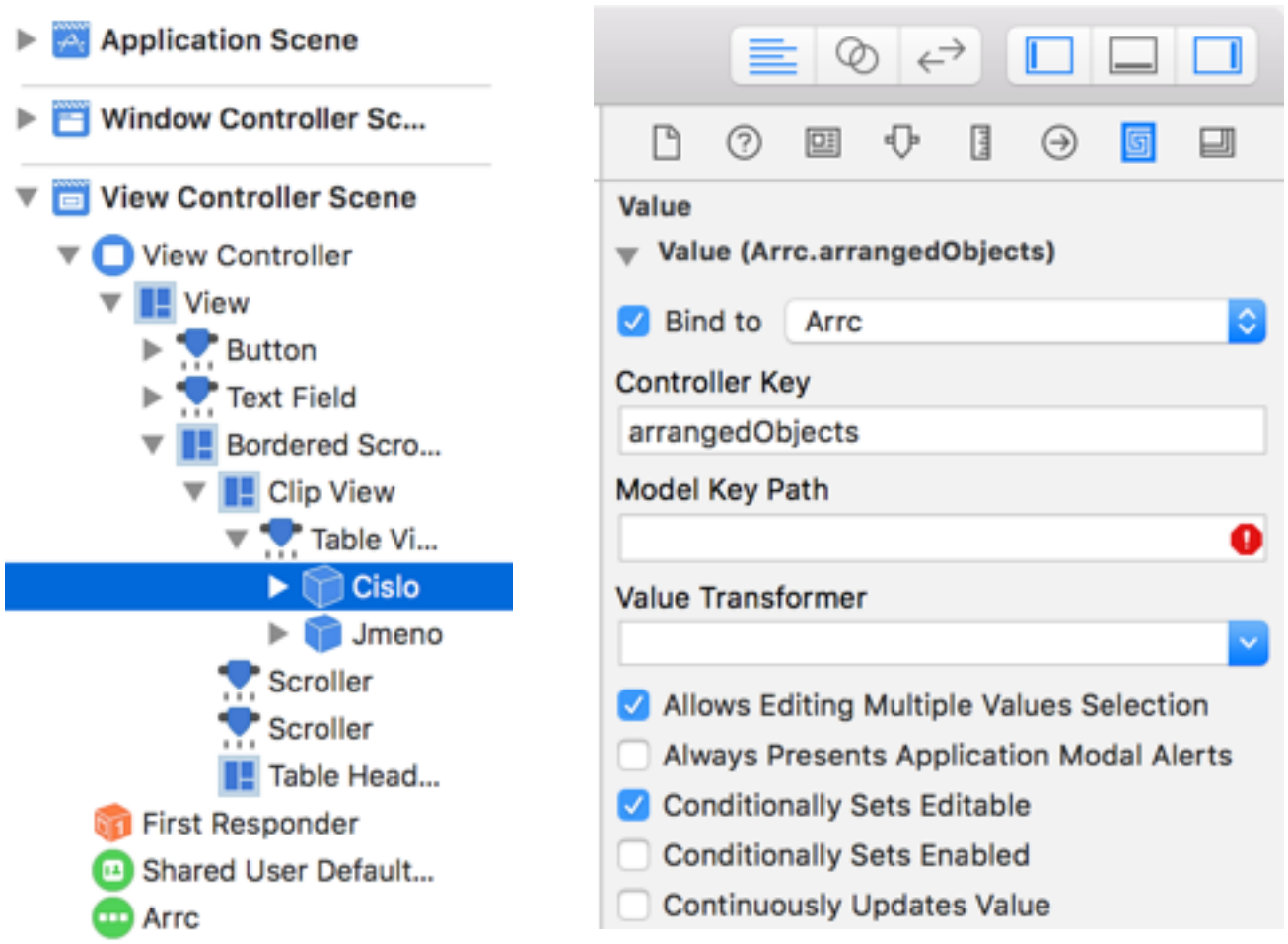


Table, Arrc, demo, tabulka

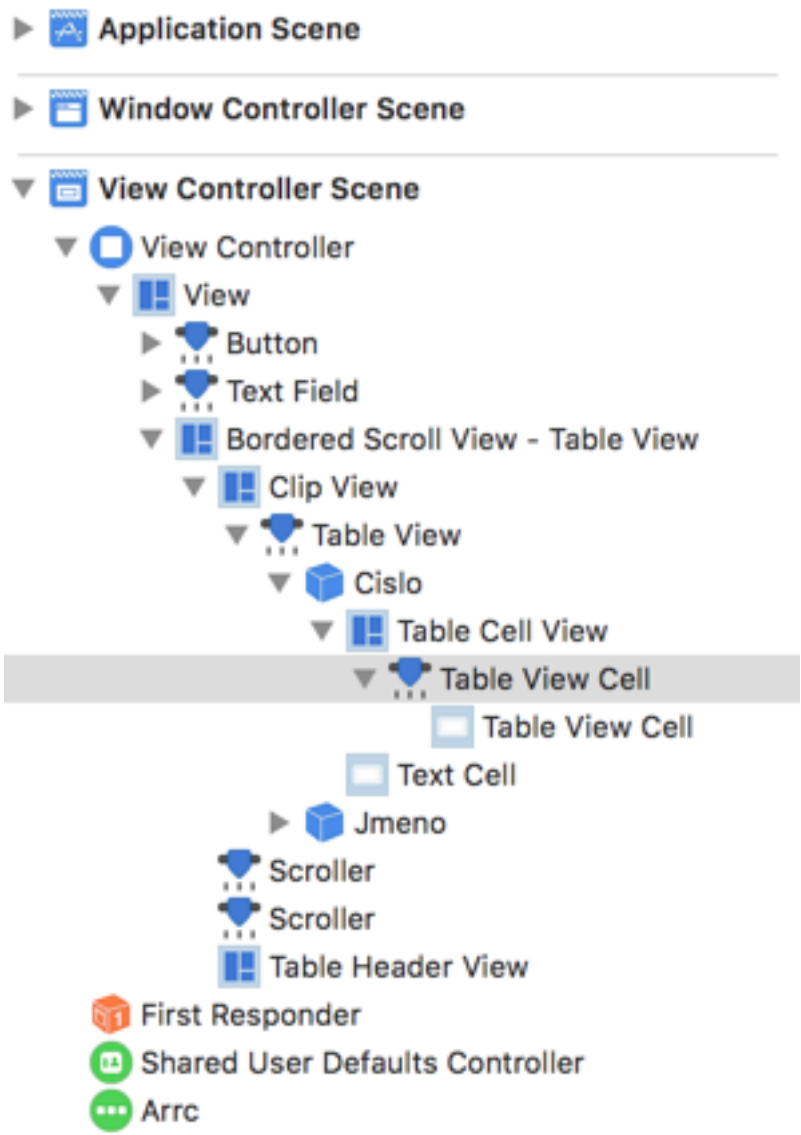


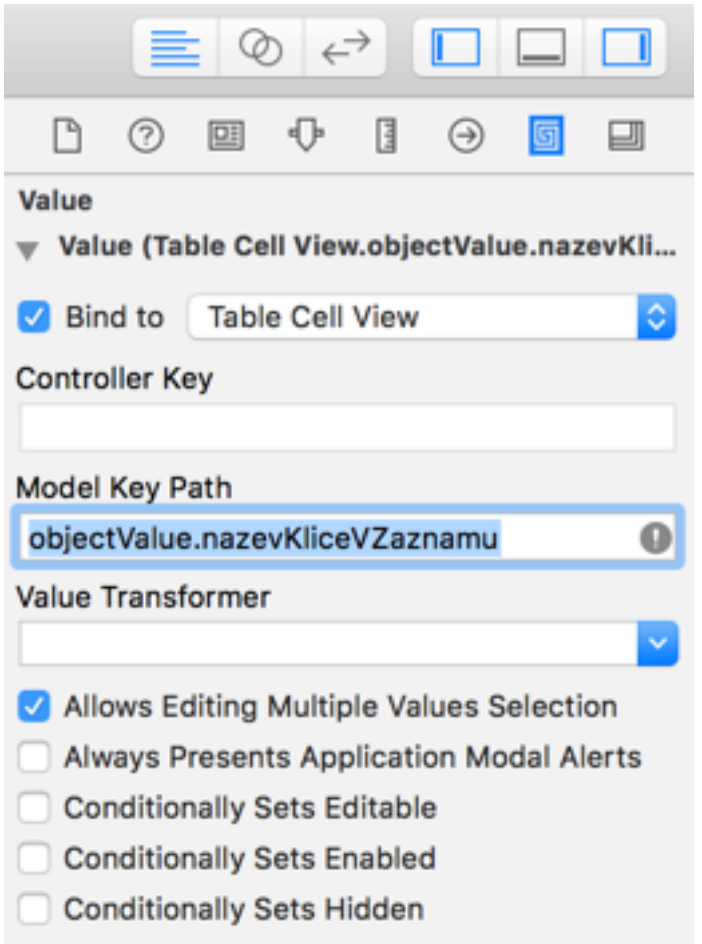


Bindings, buňka



Bindings, sloupec buňky

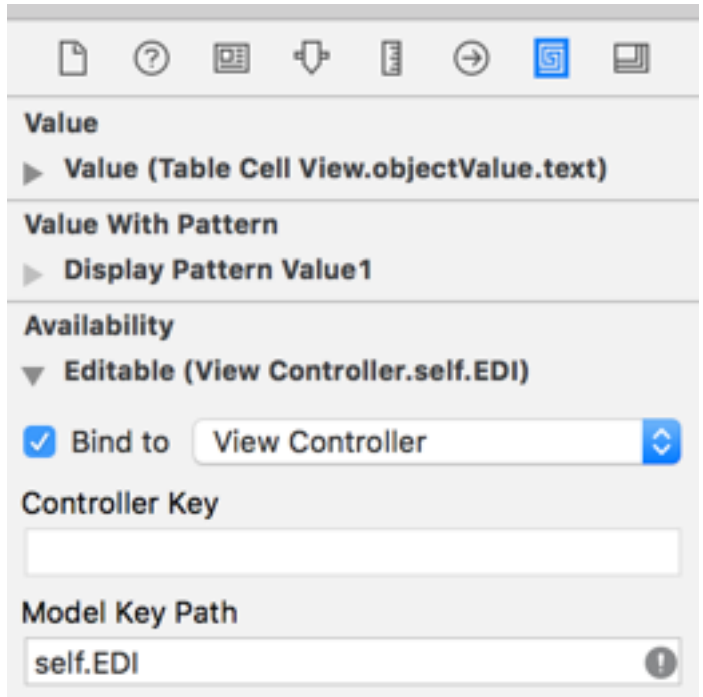




TableView s CoreData

- NSArrayController
  - Binding — MOC. ”Prepare Content”. Entity name.

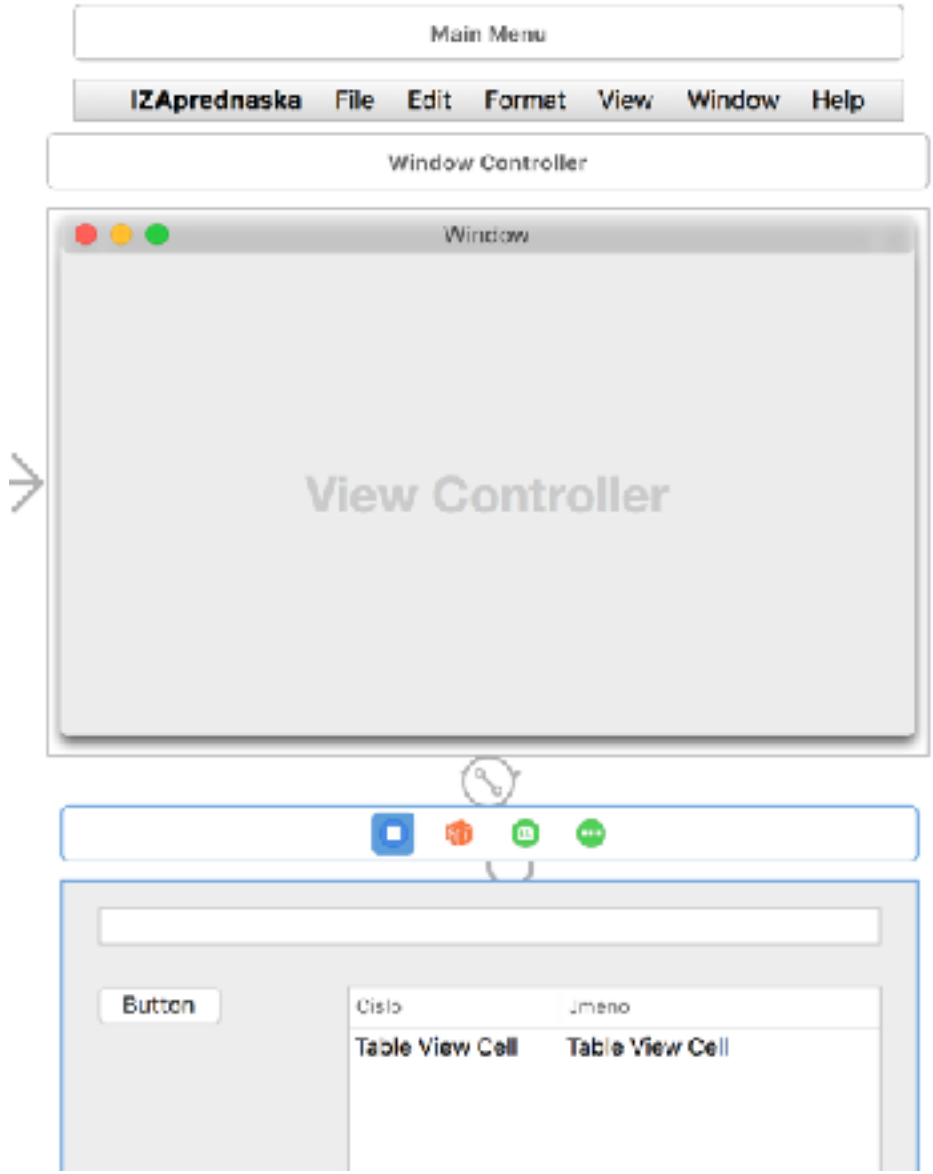
```
class ViewController: NSViewController {
    //
    @objc var MOC: NSManagedObjectContext {
        //
        return (NSApplication.shared.delegate as
AppDelegate).persistentContainer.viewContext
    }
    //
    @objc var EDI : Bool = true
```



Prvky view

- NSTextField — label nebo input line.
- Varianty tlačítek (push, radio, check).
- Strukturované — tabulka, collection view.
- Picker — datum, roletová menu apod.
- NSWindow, NSView, Sheets

NSWindow / NSWindowController



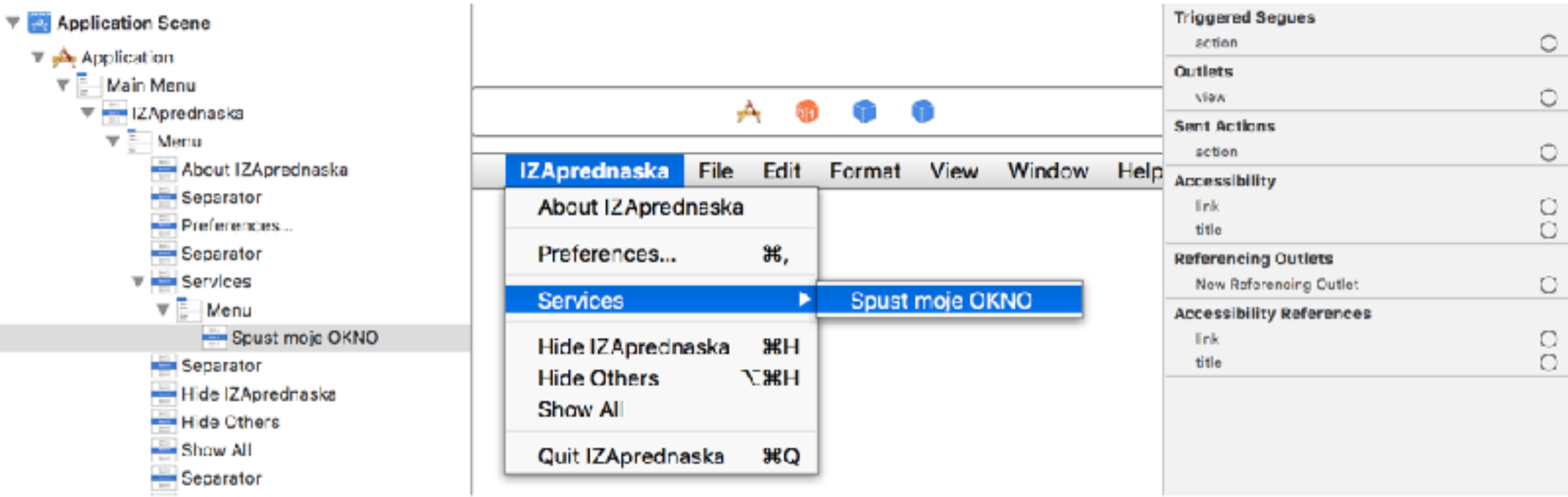
NSWindow / NSWindowController

- NSWindowController vlastní:
  - window — NSWindow
  - window content — NSViewController
- Někdo musí držet reference na NSWC.

Hlavní Controller okna

- Hierarchie VC:
  - NSWindowController — akce z ToolBaru, Menu apod.
  - contentView, NSViewController — obsluha vnitřních View (tlačítek, tabulek)
- Rozšiřování okna o další VC (split, tabbar).

Menu aplikace



Otevření okna, programem

```
// evidence otevrenych oken (AppDelegate)
    var _myWindows = Set<NSWindowController>()
    //
    @IBAction func menuOkno(_:AnyObject) {
        //
        let SB = NSStoryboard.main
        // ze SB si vytahneme instanci VC
        let win = SB!.instantiateController(withIdentifier:
NSStoryboard.SceneIdentifier(rawValue: "mainWIN")) as!
NSWindowController
        //
        _myWindows.insert(win)

        //
        win.showWindow(nil)
    }
```

Zrušení okna, programem

```
// AppDelegate
        NotificationCenter.default.addObserver(self, selector:
#selector(windowClosing(notif:)), name: NSWindow.willCloseNotification,
object: nil)



        //
            @objc func windowClosing(notif:Notification) {
                //
                guard
                    let _window = notif.object as? NSWindow,
                    let _wcont = _window.windowController,
                    _myWindows.contains(_wcont)
                    else { return }

                //
                _myWindows.remove(_wcont)
            }
```

Správa oken v aplikaci

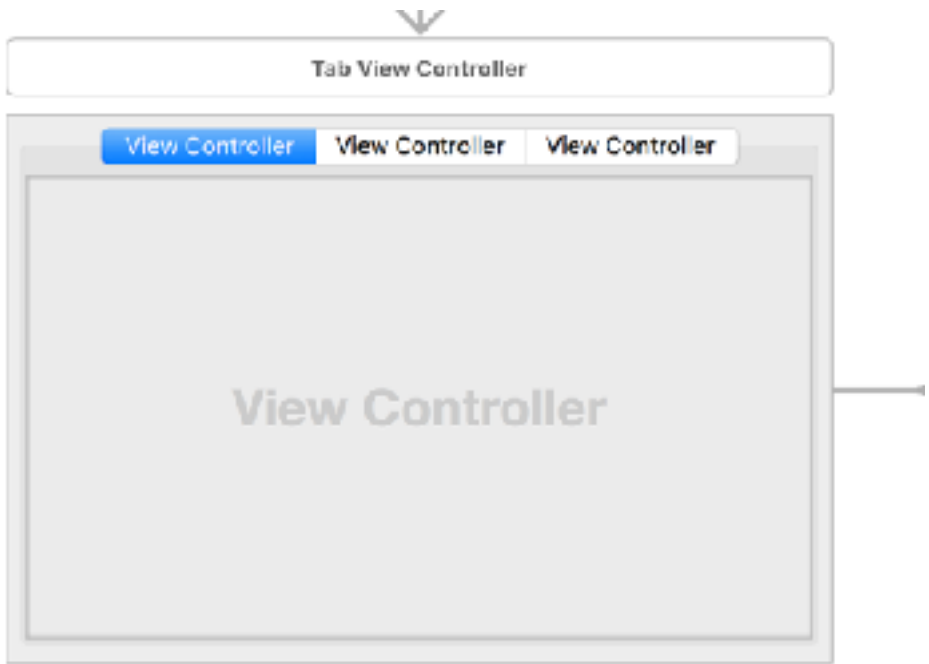
- Zavření okna (červený křížek) nemusí být v konceptu aplikací macOS bráno jako dealokace okna
- Odchytávat události, rozhodnout podle případu.
- Nastavení: ukončit aplikaci, pokud počet oken klesne na nulu

Obsah okna

- Tool Bar.
- TabBar — viditelný / skrytý.
- Sheets.
- Content view — další rozklad na subviews (horní lišta, dolní lišta, tabbar)

TabBar

- Programování Cocoa aplikací je o správném návrhu controllerů, které se dále skládají do větších (splitView, Tabbar, popover, sheet).
  - Programujeme NSViewController(-y).



Styly TabBar Controlleru

- Tlačítka s taby nahoře, dole.
- … v ToolBaru.
- skrytá — přepínání tabů se děje jinak přes API.
- TabBar — multi VC na jednom View. Často app mění i velikost okna při přepínání pohledů.

Navigation Controller

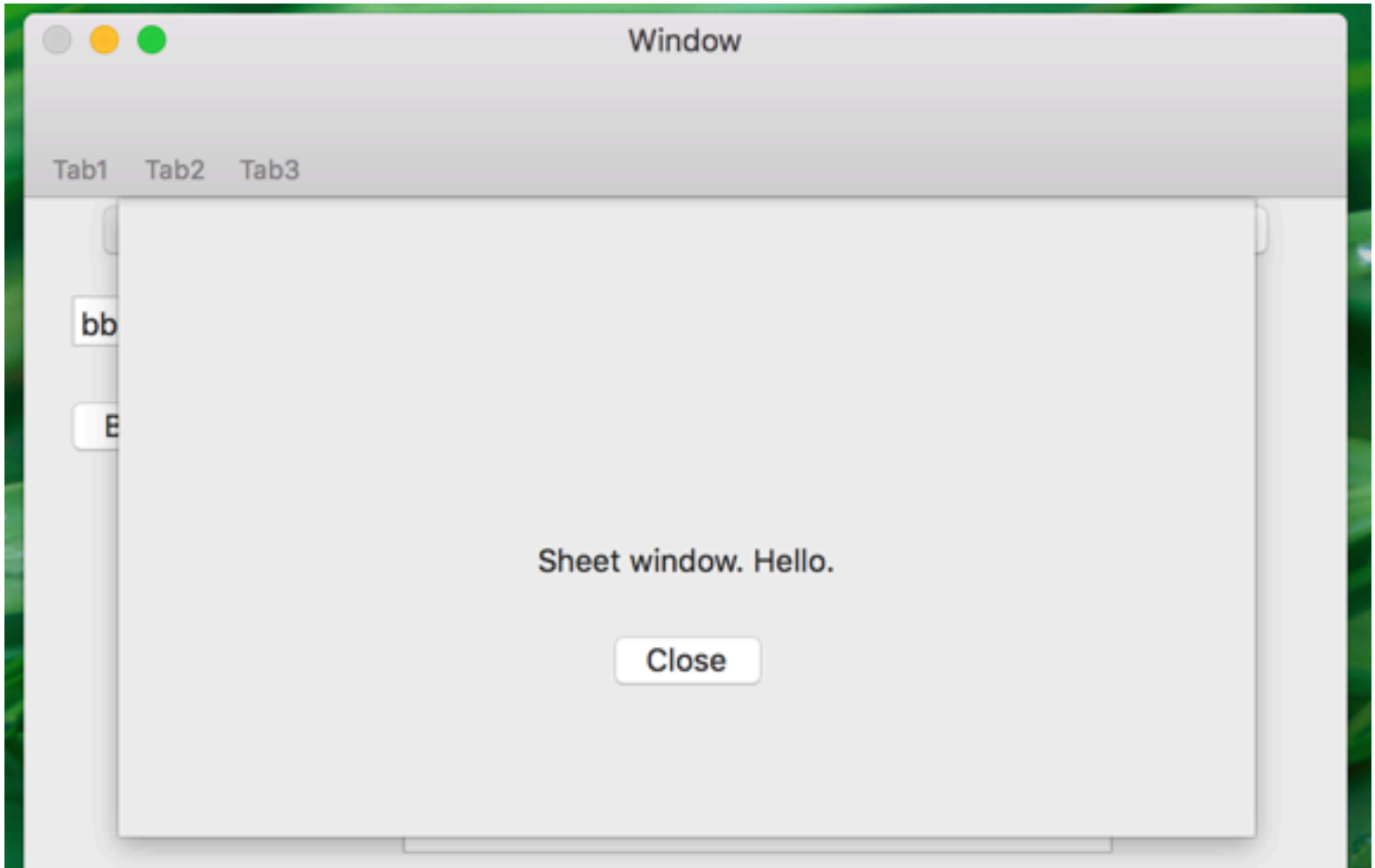
- V Cocoa NC není.
- Lze doprogramovat.
  - Vložení subView.
  - Přepnutí kontrolerů.

Split View Controller

- Horizontální, vertikální.
- Dynamické nastavení velikosti vnořených subview.
- SVC je delegátem SplitView.
- NSSplitViewItem -> NSViewController.
- Autoresize, layout, constraints.

Sheet Window

- Okno modálně prezentuje jiné okno (contentView).



Sheet, demo

```
if let sb = NSStoryboard.main {
            //
            sheet = sb.instantiateController(withIdentifier:
NSStoryboard.SceneIdentifier(rawValue: "sheet")) as! NSWindowController

            //
            view.window!.beginSheet(sheet.window!, completionHandler:
{ (resp) in
                //
            })
        }
```

Popovers

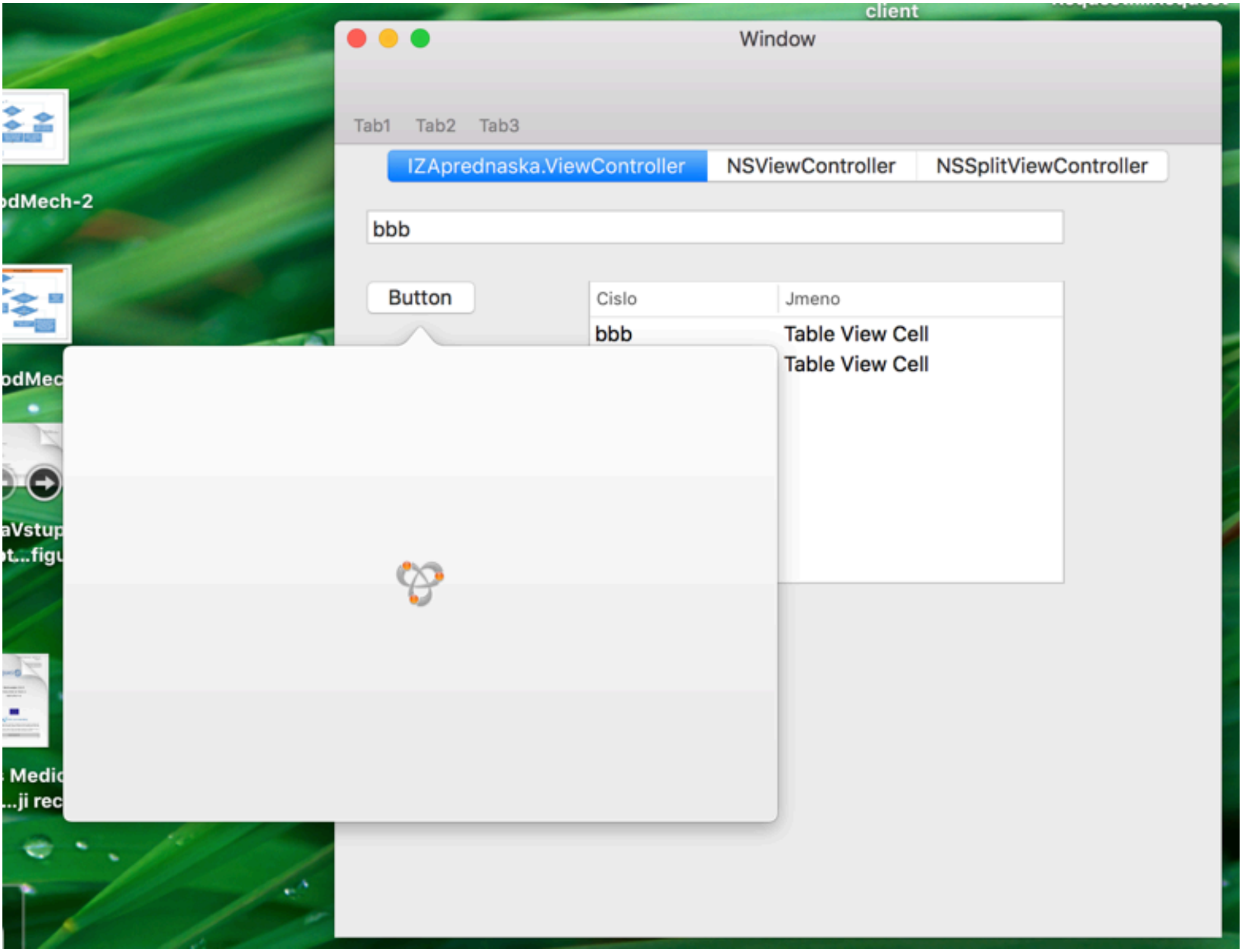
- Dočasná View plovoucí nad zadaným oknem.
  - zadaným view, které je contentView nějakého window.
- Lze je následně obalit do okna.
- NSPopover — responder
  - delegate, contentViewController, metadata, stav

Popover

```
//
        let SB = NSStoryboard.main
        // ze SB si vytahneme instanci VC
        popik = SB!.instantiateController(withIdentifier:
NSStoryboard.SceneIdentifier(rawValue: "popik")) as! NSViewController
        //
        pop = NSPopover()
        ///
        pop.contentViewController = popik
        pop.appearance = NSAppearance(named: NSAppearance.Name.aqua)
        pop.behavior = NSPopover.Behavior.transient



 // pripevni POP k zadanemu tlacitku
         let nsrect = self.button.frame
         let withinView = self.view.window!.contentView!
         //
         pop.show(relativeTo: nsrect, of: withinView, preferredEdge:
 NSRectEdge.minY)
```


client
Window
Tab1
Tab2
Tab3
IZAprednaska.ViewController
NSViewController
NSSplitViewController
bbb
Button
Cislo
Jmeno
bbb
Table View Cell
Table View Cell

XPC

- IPC — Inter-Process Communication.
- XPC — technologie značně používaná interně Applem, užív. aplikacemi a dost možná i v iOS.
- Aplikace a ”helper”.
  - client-server. Helper je proces s vnitřním stavem.
  - App Groups. Sandboxing.
  - Protokol. Helper implementuje protokol.

XPC princip

- Klient (aplikace) — Server (helper, XPC).
  - klient invokuje proceduru serveru (posílá zprávu),
  - server může invokovat closure (v kontextu klienta).

```
// protokol pro Client-Server
@objc protocol MujXPC {
    // "resp" je blok spusteny z opacne strany
    func hello(with: String, resp: (String)->())
}
```

XPC, strana klienta

```
@NSApplicationMain
class AppDelegate: NSObject, NSApplicationDelegate {
    // knihovni objekt pro ustanoveni spojeni
    var xpc: NSXPCConnection!
    // objekt posilajici zpravy do XPC
    var _agent: MujXPC!
    //
    func applicationDidFinishLaunching(_ aNotification: Notification) {
        // ...
        xpc = NSXPCConnection(serviceName: "MyXPC.hello.hell")
        //
        xpc.remoteObjectInterface = NSXPCInterface(with: MujXPC.self)
        // musi se provest. Implicitne je susspended.
        xpc.resume()
        //
        _agent = xpc.remoteObjectProxyWithErrorHandler({ (error) in
            //
            print("Agent: \(error)")
        }) as! MujXPC
    }
```

XPC, strana serveru

```
// Implementace MujXPC, tj. funkcionalita Serveru
class Agent: NSObject, NSXPCListenerDelegate, MujXPC {
    // ...
    func hello(with: String, resp: (String)->()) {
        // ted volam metodu v kontextu klienta
        resp("Saying \(with)")
    }
}
// pojmenovani sluzby
let listener = NSXPCListener(machServiceName: "MyXPC.hello.hell")
let agent = Agent()
//
listener.delegate = agent
// start serveru
listener.resume()
//
RunLoop.current.run()
```

Závěr

- Probrali jsme základy Objective-C a Swiftu.
- Programování aplikací pro iOS (důraz), tvOS, watchOS, macOS.
  - Důraz na společné prvky (Foundation).