User Interfaces

Lecture 18

Cocoa: Multiple windows

Hamza Bennani
hamza@hamzabennani.com

September 13, 2018

UNIVERSITY of OTAGO
Te Whare Wānanga o Otāgo
NEW ZEALAND

Assignment

```
let bundlePath = Bundle.main.resourcePath
outletImage.image=NSImage(contentsOfFile:
bundlePath!+"/test.png")
if let filepath = Bundle.main.path(forResource: "readme",
ofType: "txt"){
do {
let contents = try String(contentsOfFile: filepath)
outletTextView.string = contents;
outletTextView.isEditable = false ;
} catch {
// contents could not be loaded }
} else {
print ("else");
}
```

Assignment

```
let fileURL = NSURL(fileURLWithPath:
bundlePath!+"/test.mov");
playView = AVPlayer(url: fileURL as URL);
outletVideo.player = playView ;
print("video")
do {
soundPlayer = try AVAudioPlayer(contentsOf:
URL.init(fileURLWithPath:
Bundle.main.path(forResource: "test", ofType: "m4a")!))
soundPlayer.prepareToPlay();
print("playing")
} catch {
}
outletscroll.isHidden=true; outletVideo.isHidden=true;
outletImage.isHidden = true;
```

Last Lecture
Summary

- Window
- NSWindow
- Window Delegate
- Views, content view, custom views
- NSView
- View hierarchy
- View geometry

Window Controller

- The behaviour of each window is controlled by an NSWindowController object
- To create a new window, create a new class that extends (abstract) **NSWindowController**
  - When creating new "Cocoa Class" in Xcode, if you indicate that you are extending
  - NSWindowController, Xcode will provide you with an option to create a new XIB file

NSWindowController

- NSWindowController object contains methods for loading and closing of the associated window

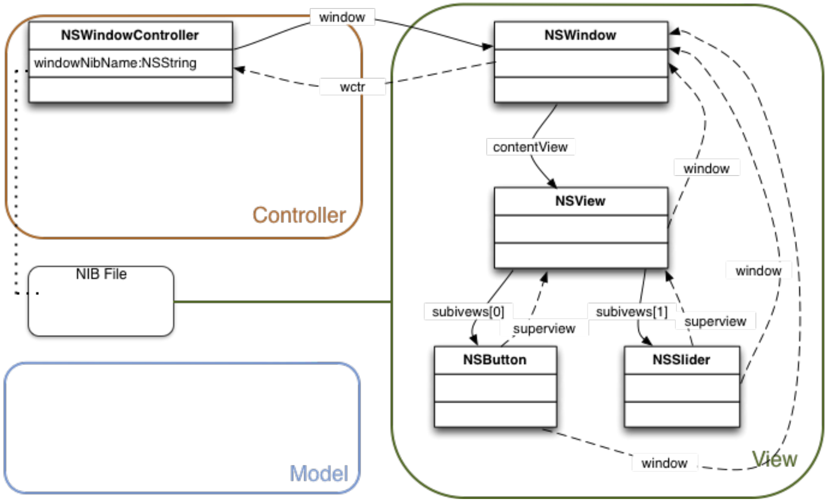
```
let newWinContrl = NSWindowController(windowNibName:
NSNib.Name(rawValue: "TimerWindowController"))
newWinContrl.showWindow(self)
```

- To create a new window, you allocate a new instance of NSWindowController (or its subclass) and associate it with a NIB file
- Window method of NSWindowController returns reference to the NSWindow object associated with the controller

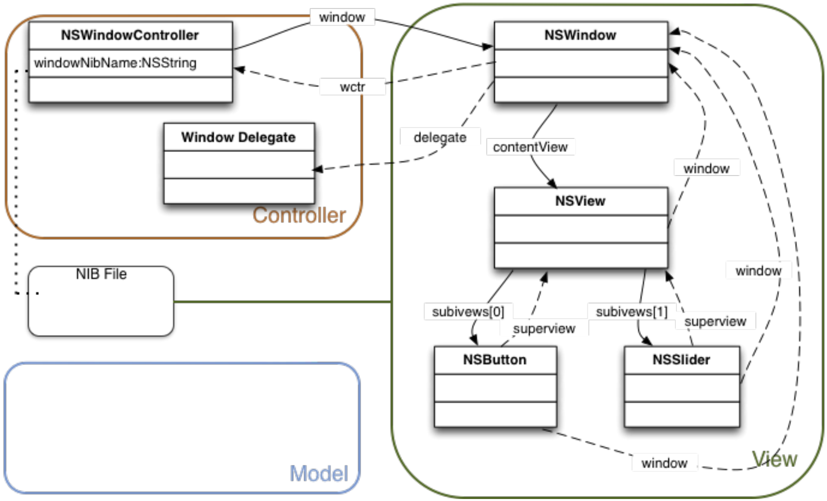
Custom Controller

- It is common to create a custom controller that extends NSWindowController
  - You can override certain methods to customise controller behaviour
    - windowWillLoad - invoked just before window is loaded
    - windowDidLoad - invoked after window is loaded
  - It is common to implement the window delegate methods in this custom controller, and make the controller the window's delegate
- When building a single-window application, the abstraction that NSWindowController provides is less important

Application
Window Controller



Application
Window Delegate



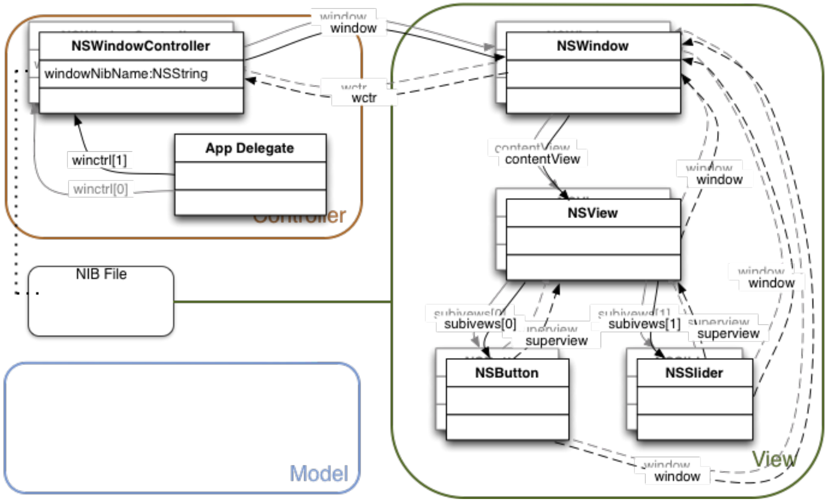
"File's Owner" and NIB files

- The process of loading a NIB file (to load a window) is trigged by an object
- That object is the "file's owner" - referring to ownership of the NIB file
  - Class of "file's owner" is stored in the NIB file: used as a proxy to load other elements on init.
- The file's owner, or its delegate, can store outlets to the elements in the window
  - Trouble creating an outlet? Check if class where you are placing outlet is "file's owner" for XIB file

"File's Owner" single/multi window

- In a single-window application, "file's owner" for main window is the NSApplication
  - Pre Xcode 8.3, a single-window application set up the AppDelegate to be the "file's owner"
- In a multi-window application, you need a "file's owner" that will have a new object instance for each new window
  - NSApplication is a singleton, so it cannot be a "file's owner" for multiple windows
  - NSWindowController is often chosen to be the "file's owner" for windows loaded at run-time

Application
Multi-window



NSViewController

- Library style applications may need to load and manage views in a window dynamically
- NSViewController is used to manage multiple views
  - Has methods for runtime load / display of views
- Storyboard-based GUI design in Xcode provides an NSViewController as default starting point for IBOutlets and IBActions

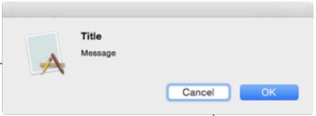
Other windows
NSPanel

- NSPanel is a subclass of NSWindow
- It is a special type of window that functions as an auxiliary window to an existing window
  - Cannot be the main window
  - Lightweight: typically not released upon closing, as it is used often
  - Example: the "open file" or "save file" panel

Other Windows
Dialog box

- Alerts are modal windows: demand user attention
  - "Are you sure you want to erase your hard drive?"
  - Good UI design tries to minimise alerts
- Alerts are easy to create

```
let choice = NSAlert()
choice.messageText = "Title"
choice.informativeText = "Message"
choice.alertStyle = NSAlertStyle.warning
choice.addButton(withTitle: "OK")
choice.addButton(withTitle: "Cancel")
switch(choice.runModal()){
        case NSAlertFirstButtonReturn:  print("OK button")
        case NSAlertSecondButtonReturn: print("Cancel button")
        default: print("another button?!?")
}
```


Title

Message

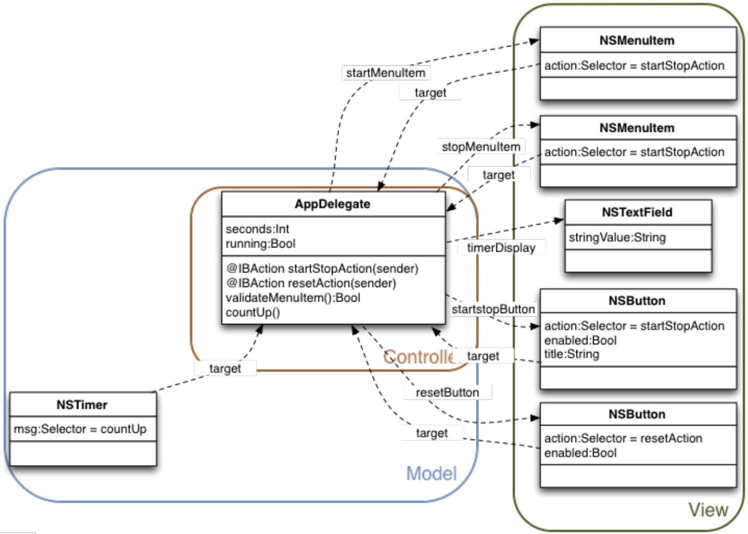
NB: I'd not actually format switch statements like this! It's just so that they fit on the slide better...

Summary

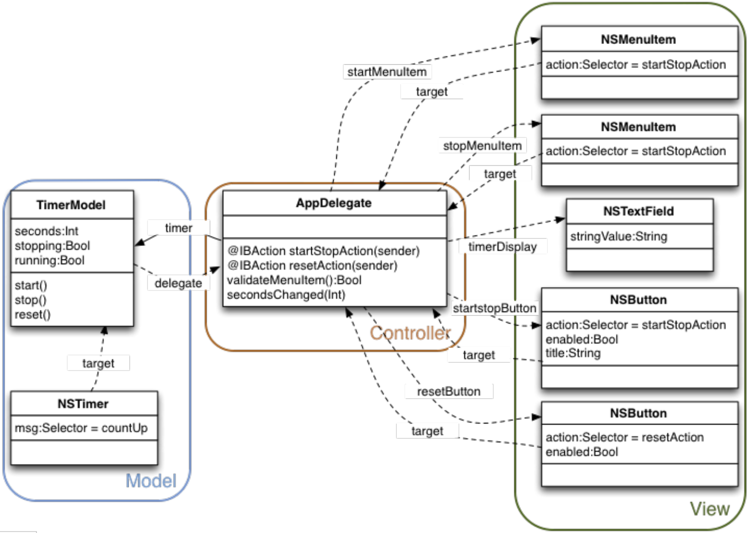
In the last two lectures we covered the windows and view hierarchy that constitute the fundamental aspects of Cocoa's GUI. We have also introduced controllers and delegates that manage the run-time behaviour of multi-window applications.

- **NSWindow** - class controlling the display aspects of a window
- **NSWindowDelegate** - protocol for window's delegate, which reacts to window events
- **NSView** - draws content in a rectangular area, coordinates, geometric operations
- **NSWindowController** - controls runtime window resources
- **NSViewController** (optional) - controls runtime view resources

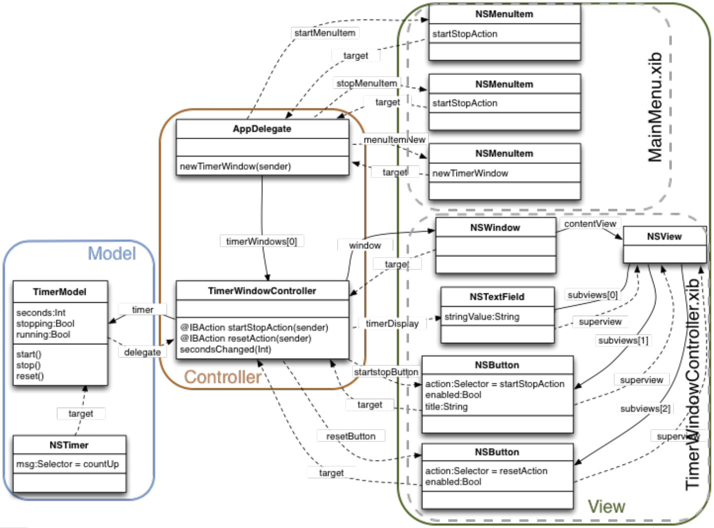
Timer App


NSMenuItem
action:Selector = startStopAction
startMenuItem
target
NSMenuItem
action:Selector = startStopAction
stopMenuItem
target
AppDelegate
seconds:Int
running:Bool
@IBAction startStopAction(sender)
@IBAction resetAction(sender)
validateMenuItem():Bool
countUp()
NSTextField
stringValue:String
timerDisplay
NSButton
action:Selector = startStopAction
enabled:Bool
title:String
startstopButton
target
Controller
target
NSTimer
msg:Selector = countUp
resetButton
NSButton
action:Selector = resetAction
enabled:Bool
target
Model
View

Timer App
MVC



Timer App
Multi-Window



Model
TimerModel
seconds:Int
stopping:Bool
running:Bool
start()
stop()
reset()
timer
delegate
target
NSTimer
msg:Selector = countUp
Controller
AppDelegate
newTimerWindow(sender)
timerWindows[0]
TimerWindowController
@IBAction startStopAction(sender)
@IBAction resetAction(sender)
secondsChanged(Int)
startMenuItem
target
stopMenuItem
target
menuItemNew
target
window
target
timerDisplay
startstopButton
target
resetButton
target
NSMenuItem
startStopAction
NSMenuItem
startStopAction
NSMenuItem
newTimerWindow
MainMenu.xib
NSWindow
contentView
NSView
NSTextField
stringValue:String
subviews[0]
superview
subviews[1]
NSButton
action:Selector = startStopAction
enabled:Bool
title:String
superview
subviews[2]
superview
NSButton
action:Selector = resetAction
enabled:Bool
TimerWindowController.xib
View