CSC 471 / 371 Mobile Application Development for iOS

Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu
@DePaulSWEng

Tabbed Views & Picker Views

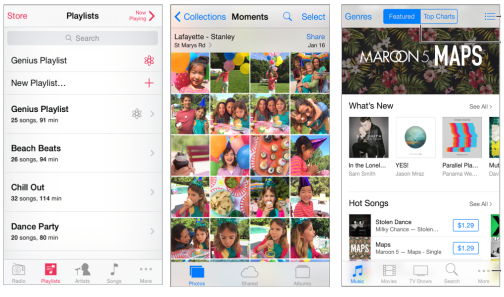
Outline

- Tabbed views
- Picker views

DePaul University 3

Tabbed Views

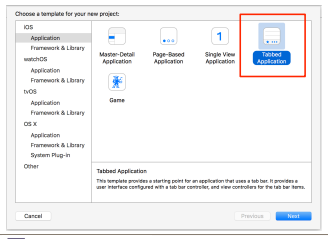
Tabbed Apps



DePaul University 5

Create a Tabbed App

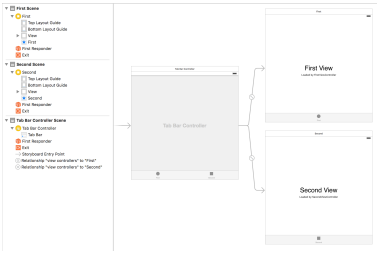
- Use the "*Tabbed Application*" template
  - Create an app with two tabs



DePaul University 6

Storyboard of the Tabbed App

- Ready to run
- Each tab can be customized
- Add tabs by adding
  - view controllers and
  - *relationship* segues



DePaul University 7

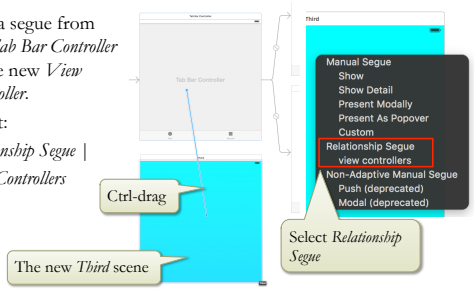
Add a New Tab

- Add a new scene
  - Drag and drop a new *View Controller* in the storyboard
- Add a new Swift class
  - Class: `ThirdViewController`
  - Subclass of: `UIViewController`
  - Uncheck: *Also create XIB file*
- In the storyboard
  - Select the new *View Controller*
  - Use the *Identity Inspector* to change the Class to: `ThirdViewController`

DePaul University 8

Add a Relationship Segue

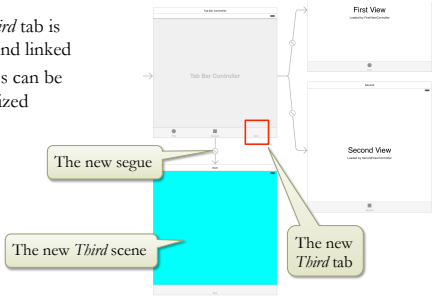
- Add a segue from the *Tab Bar Controller* to the new *View Controller*.
- Select:
  Relationship Segue | *View Controllers*



DePaul University 9

The Storyboard with Three Tabs

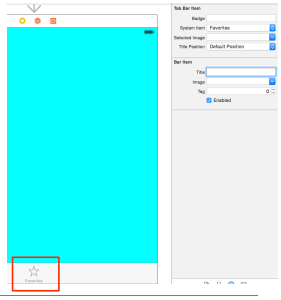
- The *Third* tab is added and linked
- The tabs can be customized



DePaul University 10

Customize the Title and Icon of the New Tab

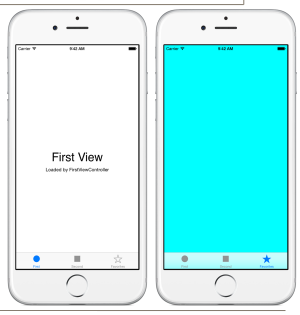
- Select the *tab item* at the bottom of the new scene
  - Use the *Attribute Inspector*
- Option 1:
  - Use *System Items*, e.g., *featured*
- Option 2:
  - Use a *Custom* item
  - Add an icon in `Assets.xcassets`
  - Choose a *title* for the tab item



DePaul University 11

Run … the Tabbed App

- The *First* tab
- The *Featured* tab

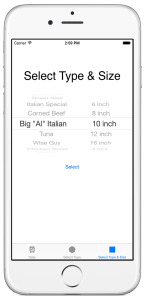


DePaul University 12

"Order My Sub" App Using Pickers

"Order My Sub" App

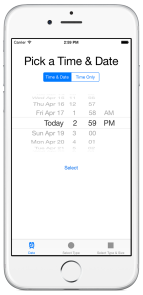
- A three-tab app using pickers
  - A date picker to pick date and time
  - A simple one-component picker to choose the type of sub to order
  - A two-component picker to choose the type and the size of the sub to order
- Using popup alerts and action sheets
  - Using closures for actions
- Sharing data between tabs



DePaul University 14

Using a Date Picker

- Start with the *Tabbed App* template
- Add a third tab with a *Date Picker*
  - Add new *View Controller* class: `DateViewController`
  - Link to the third *View Controller* in the storyboard
- Customize the third tab
  - Change the tab image and title
  - Add a *Date Picker* and other controls in the view



DePaul University 15

The "Pick a Date & Time" Scene

- Connect outlets to
  - The *Segmented Control*
  - The *Date Picker*
- Connect actions to
  - The *Segmented Control*
  - The "Select" *Button*

DePaul University 16

The Date View Controller – the Outlets and Actions

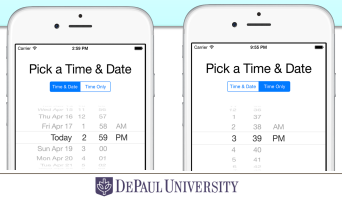
```
class DateViewController: UIViewController {
    @IBOutlet weak var options: UISegmentedControl!
    @IBOutlet weak var datePicker: UIDatePicker!
    @IBAction func optionSelected(sender: UISegmentedControl)
    { … }
    @IBAction func datePicked(sender: UIButton) { … }

    override func viewDidLoad() {
        super.viewDidLoad()
        options.selectedSegmentIndex = 0
    }
    …
}
```

Set the initial state of the *Segmented Control*

DePaul University 17

Set the Date Picker Mode

```
@IBAction func optionSelected(sender: UISegmentedControl) {
    if sender.selectedSegmentIndex == 0 {
        datePicker.datePickerMode = .DateAndTime
    } else if sender.selectedSegmentIndex == 1 {
        datePicker.datePickerMode = .Time
    }
}
```

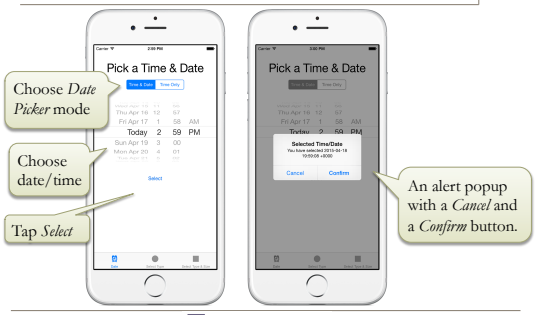


DePaul University 18

Action for Picking Date

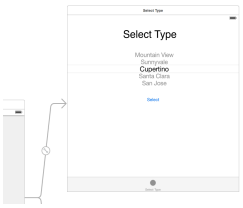
```
@IBAction func datePicked(sender: UIButton) {
    let title = "Selected Time/Date"
    let message = "You have selected \(datePicker.date)"
    let alertController = UIAlertController(title: title,
            message: message, preferredStyle: .Alert)
    let cancelAction = UIAlertAction(title: "Cancel",
            style: .Cancel, handler: nil)
    let okayAction = UIAlertAction(title: "Confirm",
            style: .Default, handler: nil)
    alertController.addAction(cancelAction)
    alertController.addAction(okayAction)
    presentViewController(alertController, animated: true,
            completion: nil)
}
```


Select a Time and a Date



The "Select Type" Scene – Single Component Picker

- Use *View Controller* class: `FirstViewController`
- Add a *Picker View* and a *Button* in the view
- Connect an outlet to the *Picker*
- Connect an actions to the "Select" *Button*



The First View Controller

- The outlet and the action

```
class FirstViewController: UIViewController,
    UIPickerViewDataSource, UIPickerViewDelegate {

    @IBOutlet weak var picker: UIPickerView!
    @IBAction func seleted(sender: UIButton) { … }
    …
}
```

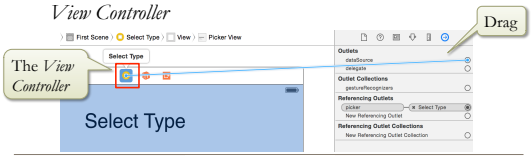

Populate Data in the Picker

- Use the *Delegate* pattern
  - The *Picker View* will delegate the task of populating the data to this *View Controller*.
- This *View Controller* must conform to two protocols
  - `UIPickerViewDelegate`
  - `UIPickerViewDataSource`

```
class FirstViewController: UIViewController,
    UIPickerViewDataSource, UIPickerViewDelegate {
    …
}
```


Connection to the Delegate

- The connection between the *Picker View* and its *data source* and *delegate* must be explicitly set
- In storyboard, select the *Picker View*
- Select the *Connection Inspector*
- Connect the outlets named *dataSource* and *delegate* to the *View Controller*



The Single Component Picker

- Populate the *Picker View* data in the *View Controller*

```
let subs = [
    "Blockbuster",
    "Roast Beef",
    …,
    "American"
]

class FirstViewController: UIViewController,
    UIPickerViewDataSource, UIPickerViewDelegate {
    …
}
```

- An array of strings used to populate the *Picker View*.
- In the *global* scope, i.e., the *internal* scope. Visible inside the project. Shared among view controllers.

Implement the Data Source

- The methods of the `UIPickerViewDataSource` protocol

```
func numberOfComponentsInPickerView(
        pickerView: UIPickerView) -> Int {
    return 1
}

func pickerView(pickerView: UIPickerView,
                numberOfRowsInComponent component: Int)
                -> Int {
    return subs.count
}
```


Implement the Delegate

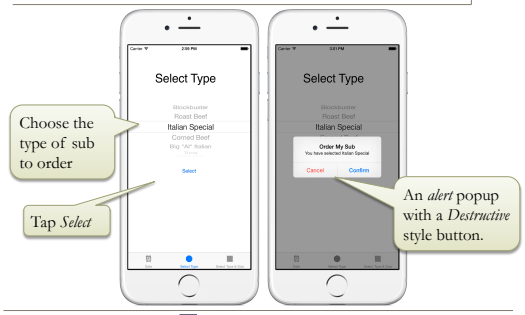
- The methods for the `UIPickerViewDelegate` protocol

```
func pickerView(pickerView: UIPickerView,
                titleForRow row: Int,
                forComponent component: Int)
                -> String? {
    return subs[row]
}
```


The Select Action

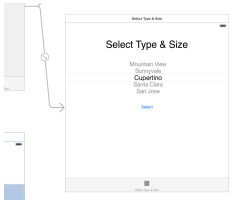
```
@IBAction func seleted(sender: UIButton) {
    let title = "Order My Sub"
    let message = "You have selected
\(subs[picker.selectedRowInComponent(0)])"
    let alertController = UIAlertController(title: title,
        message: message, preferredStyle: .Alert)
    let cancelAction = UIAlertAction(title: "Cancel",
        style: .Destructive, handler: nil)
    let okayAction = UIAlertAction(title: "Confirm",
        style: .Default, handler: nil)
    alertController.addAction(cancelAction)
    alertController.addAction(okayAction)
    presentViewController(alertController, animated: true,
        completion: nil)
}
```


The Test Run …



The "Select Type & Size" Scene – Double Component Picker

- Use *View Controller* class: `SecondViewController`
- Add a *Picker View* and a *Button* in the view
- Connect an outlet to the *Picker*
- Connect an actions to the "Select" *Button*
- Connect the *dataSource* and the *delegate* of the *Picker View*



The Second View Controller

- The outlet and the action
- This *View Controller* also conforms to two protocols
  - `UIPickerViewDelegate`
  - `UIPickerViewDataSource`

```
class SecondViewController: UIViewController,
    UIPickerViewDataSource, UIPickerViewDelegate {

    @IBOutlet weak var picker: UIPickerView!
    @IBAction func seleted(sender: UIButton) { … }
    …
}
```

DePaul University 31

The Double Component Picker

- Populate the *Picker View* data in the *View Controller*

```
let sizes = [
    "6 inch",
    "8 inch",
    "10 inch",
    "12 inch",
    "16 inch",
    "3 foot"
]
class SecondViewController: UIViewController,
    UIPickerViewDataSource, UIPickerViewDelegate {
    …
}
```

An array of strings used to populate the second component of the *Picker View*.

DePaul University 32

Implement the Data Source

- The `UIPickerViewDataSource` protocol

```
func numberOfComponentsInPickerView(
        pickerView: UIPickerView) -> Int {
    return 2
}
func pickerView(pickerView: UIPickerView,
                numberOfRowsInComponent component: Int)
                -> Int {
    if component == 0 {
        return subs.count
    } else {
        return sizes.count
    }
}
```

For component 0

For component 1

DePaul University 33

Implement the Delegate

- The `UIPickerViewDelegate` protocol

```
func pickerView(pickerView: UIPickerView,
                titleForRow row: Int,
                forComponent component: Int)
                -> String? {
    if component == 0 {
        return subs[row]
    } else {
        return sizes[row]
    }
}
```

For component 0

For component 1

DePaul University 34

The Select Action

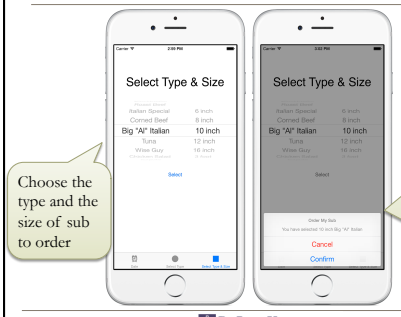
```
@IBAction func selected(sender: UIButton) {
    let title = "Order My Sub"
    let message = "You have selected
\(sizes[picker.selectedRowInComponent(1)])
\(subs[picker.selectedRowInComponent(0)])"
    let alertController = UIAlertController(title: title,
        message: message, preferredStyle: .ActionSheet)
    let cancelAction = UIAlertAction(title: "Cancel",
        style: .Destructive) { action in … }
    let confirmAction = UIAlertAction(title: "Confirm",
        style: .Default) { action in … }
    alertController.addAction(cancelAction)
    alertController.addAction(confirmAction)
    presentViewController(alertController, animated: true,
        completion: nil)
}
```

Action for *Cancel*

Action for *Confirm*

DePaul University 35

The Action Sheet Popup



Choose the type and the size of sub to order

An *action sheet* popup with a *Destructive* style button and a *Default* style button.

DePaul University 36

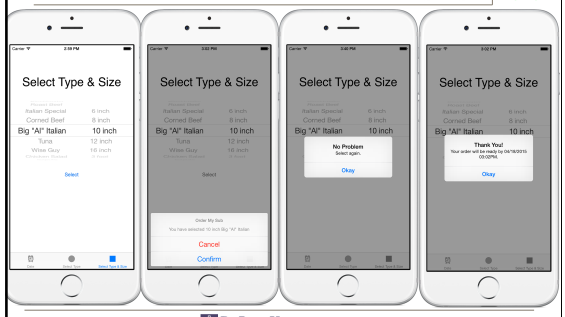
The Select – Cancel Action

```
@IBAction func selected(sender: UIButton) {
    …
    let cancelAction = UIAlertAction(title: "Cancel",
        style: .Destructive) { action in
        let cancelController = UIAlertController(
            title: "No Problem", message: "Select again.",
            preferredStyle: .Alert)
        let okayAction = UIAlertAction(title: "Okay",
            style: .Default, handler: nil)
        cancelController.addAction(okayAction)
        self.presentViewController(cancelController,
            animated: true, completion: nil)
    }
    …
```


The Select – Confirm Action

```
    let confirmAction = UIAlertAction(title: "Confirm",
        style: .Default) { action in
        let okayController = UIAlertController(
            title: "Thank You!", message: "Your order ….",
            preferredStyle: .Alert)
        let okayAction = UIAlertAction(title: "Okay",
            style: .Default, handler: nil)
        okayController.addAction(okayAction)
        self.presentViewController(okayController,
            animated: true, completion: nil)
    }
    alertController.addAction(cancelAction)
    alertController.addAction(confirmAction)
    …
}
```


The Cancel and Conform Popup



One More Refinement

- Use the date/time selected at "Pick a Time & Data" tab as the pick-up time.
- Format the confirm message of the "Pick Type & Size" tab using the pick-up time.
- To pass data between tabs, we use a global variable

```
var date = NSDate()

class DateViewController: UIViewController {
    …
}
```

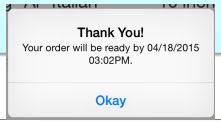
A global variable

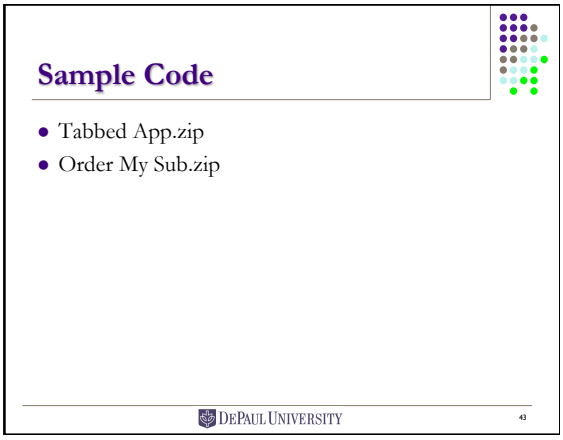
Pick a Time and Date – Handle Confirm Action

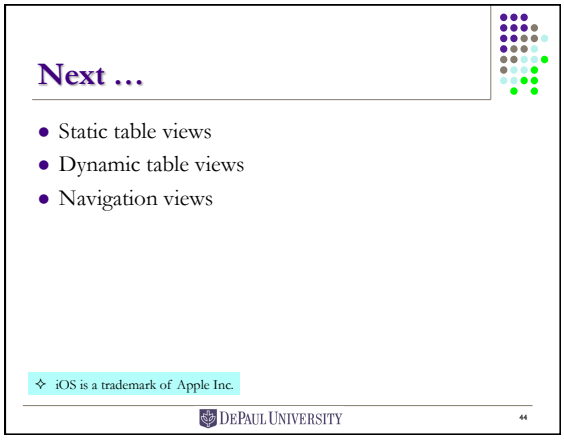
```
@IBAction func datePicked(sender: UIButton) {
    let title = "Selected Time/Date"
    let message = "You have selected \(datePicker.date)"
    let alertController = UIAlertController(title: title,
            message: message, preferredStyle: .Alert)
    let cancelAction = UIAlertAction(title: "Cancel",
            style: .Cancel, handler: nil)
    let okayAction = UIAlertAction(title: "Confirm",
            style: .Default) {
        action in date = self.datePicker.date
    }
    alertController.addAction(cancelAction)
    alertController.addAction(okayAction)
    presentViewController(alertController, animated: true,
                          completion: nil)
}
```

Pick Type & Size – The Select – Confirm Action

```
let confirmAction = UIAlertAction(title: "Confirm",
    style: .Default) { action in
    let dateFormat = NSDateFormatter()
    dateFormat.dateFormat = "MM/dd/yyyy hh:mma"
    let dateString = dateFormat.stringFromDate(date)
    let okayController = UIAlertController(
      title: "Thank You!",
      message: "Your order will be ready by \(dateString).",
      preferredStyle: .Alert)
    …
}
```




Sample Code
Tabbed App.zip
Order My Sub.zip
DePaul University
43


Next …
Static table views
Dynamic table views
Navigation views
iOS is a trademark of Apple Inc.
DePaul University
44