Implementing UI Designs in Interface Builder

Session 407

Kevin Cathey Interface Builder Engineer
Tony Ricciardi Interface Builder Engineer
Design Time

Build Time
Run Time
Build Time
Design Time
Demo

Design Time — Best Practices, Tips and Tricks
Takeaways
Best practices
Takeaways

Best practices

Rob, Amber, and Leslie
Epic road trip to Utah for some hiking

Stack Views
Takeaways
Best practices

Stack Views

Dynamic Type
Takeaways
Best practices

Stack Views

Dynamic Type

Designables

Rob, Amber, and Leslie
Epic road trip to Utah for some hiking
Takeaways

Best practices

Stack Views

Dynamic Type

Designables

Inspectables
Takeaways
Best practices

Stack Views
Dynamic Type

Prototype Cells

Designables

Inspectables

Storyboard References
Takeaways
Tips and tricks
Takeaways

Tips and tricks
Takeaways
Tips and tricks

Fast Selection

Canvas Customizations
Takeaways

Tips and tricks

Fast Selection

Canvas Customizations

Advanced Navigation
Takeaways
Tips and tricks

Fast Selection

Canvas Customizations

Advanced Navigation

Multiple Bar Items
Takeaways

Tips and tricks

Fast Selection

Canvas Customizations

Advanced Navigation

Multiple Bar Items

Simulated Metrics
Design Time
XML Documents

Build Time

Run Time
Run Time
Build Time
Design Time
XML Documents
ibtool
Run Time
Run Time
Build Time
Design Time
XML Documents
ibtool
Nib Files
Compiling Storyboards
Compiling Storyboards
Compiling Storyboards
Compiling Storyboards
Compiling Storyboards
Compiling Storyboards
Compiling Storyboards
Loading Storyboards At Run Time
loading storyboards at run time

```
UIStoryboard
init(name:bundle:)
```
Loading Storyboards At Run Time

UIStoryboard

init(name:bundle:)

instantiateInitialViewController()
Loading Storyboards At Run Time

UIStoryboard

init(name:bundle:)

instantiateViewController()
Loading Storyboards At Run Time

UIStoryboard

`instantiateViewControllerWithIdentifier(_:)`

`instantiateViewControllerWithIdentifier(_:)`

`instantiateViewControllerWithIdentifier(_:)`

`init(name:bundle:)`

UIViewController

`view`

`view`
Loading Storyboards At Run Time

UIStoryboard
  instantiateInitialViewController()

UIViewController
  view

UIStoryboard
  instantiateViewControllerWithIdentifier(_:)

UIStoryboard
  init(name:bundle:)

UIStoryboard
Loading Storyboards At Run Time

- **UIKit**
  - `UIStoryboard`
    - `init(name:bundle:)`
    - `instantiateViewControllerWithIdentifier(_:)`
    - `instantiateViewControllerWithIdentifier(_:)`
    - `instantiateViewControllerWithIdentifier(_:)`
    - `dequeueReusableCellWithIdentifier(_:)`

- **UIViewController**
  - `view`
Takeaways
Takeaways

**Performance.** Nib files loaded on demand.
Takeaways

**Performance.** Nib files loaded on demand.

**Reuse.** Nib files enable reuse.
Takeaways

**Performance.** Nib files loaded on demand.

**Reuse.** Nib files enable reuse.

**Life cycle.** Know when objects are created.
Run Time
Build Time
Design Time
Run Time
Design Time

Build Time

Run Time
Interface Builder at Run Time

Tony Ricciardi
Design Time
Build Time
Run Time
Connections API Adaptability
class AccountViewController : UIViewController {

    @IBOutlet var usernameLabel: UILabel!

    override func viewDidLoad() {
        usernameLabel.text = username
    }

    var username: String? {
        didSet {
            usernameLabel?.text = username
        }
    }
}
class AccountViewController : UIViewController {

    @IBOutlet var usernameLabel: UILabel!

    override func viewDidLoad() {
        usernameLabel.text = username
    }

    var username: String? {
        didSet {
            usernameLabel?.text = username
        }
    }
}
class AccountViewController : UIViewController {

    @IBOutlet var usernameLabel: UILabel!

    override func viewDidLoad() {
        usernameLabel.text = username
    }

    var username: String? {
        didSet {
            usernameLabel?.text = username
        }
    }
}
class AccountViewController: UIViewController {

    @IBOutlet var usernameLabel: UILabel!

    override func viewDidLoad() {
        usernameLabel.text = username
    }

    var username: String? {
        didSet {
            usernameLabel?.text = username
        }
    }
}
Connections

class LoginViewController : UIViewController {

    @IBAction func toggledAutoLoginSwitch(sender: UISwitch) {
        UserSettings.autoLogin = sender.on
    }

    @IBAction func tappedLoginButton() {
        if attemptLogin() {
            performSegueWithIdentifier("unwindAfterLogin", sender: nil)
        } else {
            performSegueWithIdentifier("presentLoginError", sender: nil)
        }
    }
}


class LoginViewController: UIViewController {

    @IBAction func toggledAutoLoginSwitch(sender: UISwitch) {
        UserSettings.autoLogin = sender.on
    }

    @IBAction func tappedLoginButton() {
        if attemptLogin() {
            performSegueWithIdentifier("unwindAfterLogin", sender: nil)
        } else {
            performSegueWithIdentifier("presentLoginError", sender: nil)
        }
    }
}

Connections
class LoginViewController: UIViewController {

    @IBAction func toggledAutoLoginSwitch(sender: UISwitch) {
        UserSettings.autoLogin = sender.on
    }

    @IBAction func tappedLoginButton() {
        if attemptLogin() {
            performSegueWithIdentifier("unwindAfterLogin", sender: nil)
        } else {
            performSegueWithIdentifier("presentLoginError", sender: nil)
        }
    }
}

Connections
API

UIStoryboard:
init(name:bundle:)
func instantiateInitialViewController()
func instantiateViewControllerWithIdentifier(_:)

UIViewController:
var storyboard: UIStoryboard? { get }
API

UIViewController:
  func prepareForSegue(_:sender:)
  func performSegueWithIdentifier(_:sender:)
  func shouldPerformSegueWithIdentifier(_:sender:) -> Bool
  func unwindForSegue(_:towardsViewController:)

UIStoryboardSegue:
  func perform()
Adaptability
Adaptability
Adaptability
Adaptability
Adaptability
Adaptability
Adaptability
Demo
Interface Builder at Run Time
Summary
Design a flexible UI with constraints and stack views
Summary

Design a flexible UI with constraints and stack views
Rapidly iterate with designable views
Summary

Design a flexible UI with constraints and stack views
Rapidly iterate with designable views
Modularize your UI with Storyboard References
Summary

Design a flexible UI with constraints and stack views
Rapidly iterate with designable views
Modularize your UI with Storyboard References
Reuse content with the storyboard API

`instantiateViewControllerWithIdentifier(_:)`
Summary

Design a flexible UI with constraints and stack views
Rapidly iterate with designable views
Modularize your UI with Storyboard References
Reuse content with the storyboard API
Make your UI adaptive with Size Classes
More Information

Apple Developer Forums
http://developer.apple.com/forums

Stefan Lesser
Developer Tools Evangelist
slesser@apple.com
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date and Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Storyboards</td>
<td>Mission</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Mysteries of Auto Layout, Part 1</td>
<td>Presidio</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Mysteries of Auto Layout, Part 2</td>
<td>Presidio</td>
<td>Thursday 1:30PM</td>
</tr>
<tr>
<td>Building Adaptive Apps with UIKit</td>
<td></td>
<td>WWDC14</td>
</tr>
<tr>
<td>Taking Control of Auto Layout in Xcode</td>
<td></td>
<td>WWDC13</td>
</tr>
</tbody>
</table>
## Related Labs

<table>
<thead>
<tr>
<th>Interface Builder and Auto Layout</th>
<th>Developer Tools Lab C</th>
<th>Thursday 2:30PM</th>
</tr>
</thead>
</table>
