goto; conference

TDD by Controlling Dependencies

Jorge D. Ortiz-Fuentes



Join the conversation #gotocph

goto; conference





Click ‘engage’ to rate sessions and ask questions



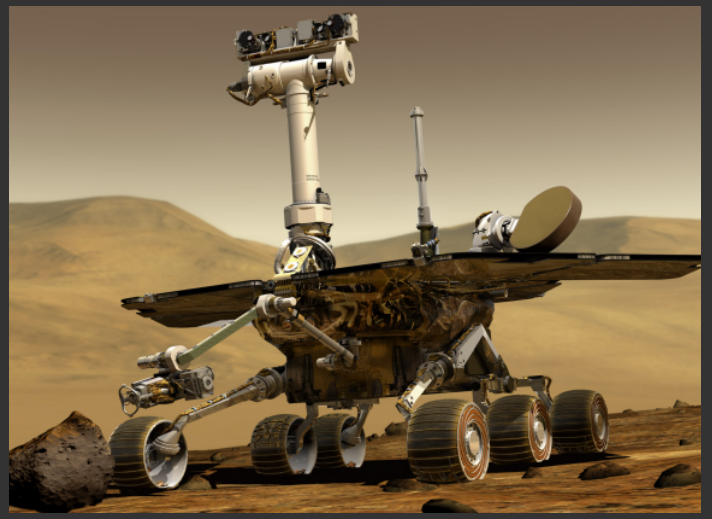
Join the conversation #gotocph

You Go First









TDD by Controlling Dependencies



Jorge D. Ortiz-Fuentes
@jdortiz

#Swift2TDD

A Canonical Examples production

#Swift2TDD



Agenda

- ★ TDD
- ★ Real Life
- ★ Swift



#Swift2TDD

TDD



TDD



Progress as a developer

- ★ It does things!
- ★ It does the things I want!
- ★ I can prove that it does what I say!
- ★ I am going to make it do exactly what i say!

Nerdvana



#Swift2TDD

The Rules of Testing

- ★ We only test our code
- ★ Only a level of abstraction
- ★ Only public methods
- ★ Only one assertion per test
- ★ Tests are independent of sequence or state



#Swift2TDD

The TDD Dance



Refactor





Red



Green

Pros

- ★ 100% coverage
- ★ Detect edge cases and unforeseen scenarios
- ★ Measurable progress
- ★ Easier to introduce changes
- ★ Less debug

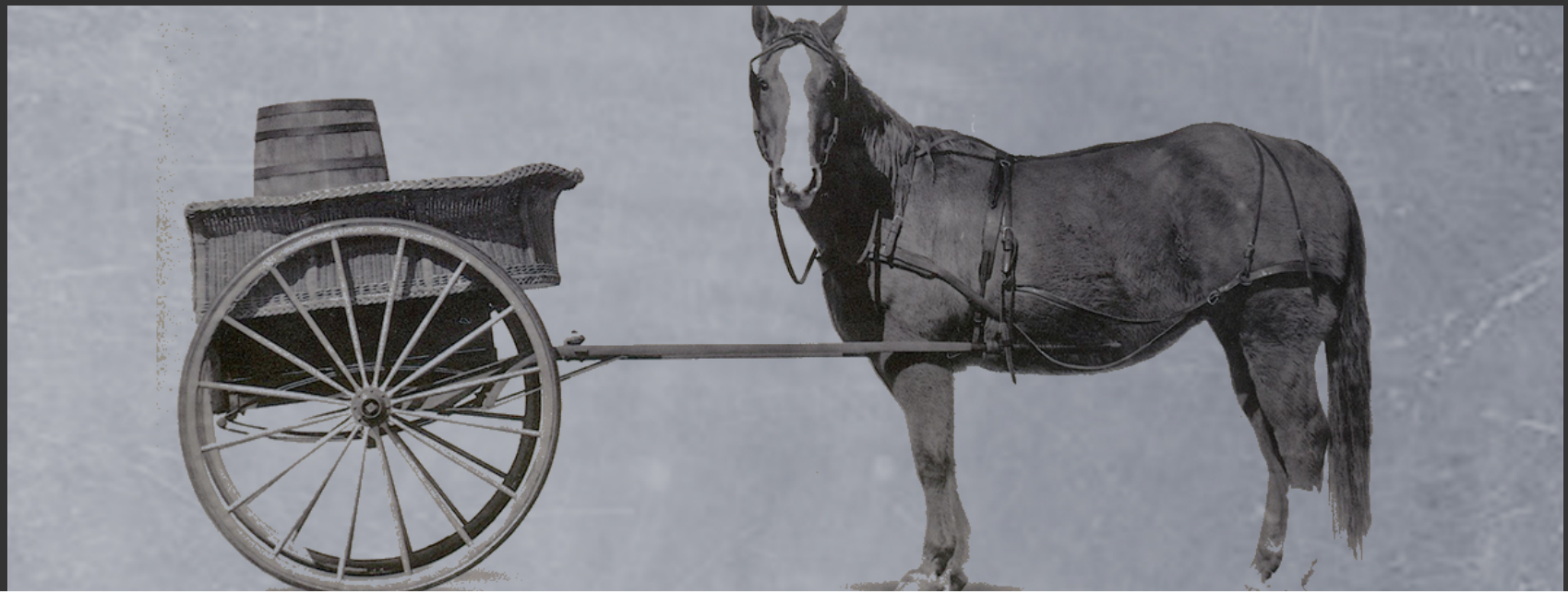
- ★ Decoupled code
- ★ Explicit dependencies
- ★ Single responsibility
- ★ Autopilot programming
- ★ Simpler design

Easier



#Swift2TDD

TDD Cons?



Cons

- ★ Put the horse before the cart
- ★ Takes longer
- ★ Initial investment. Slower and more work in QA
- ★ Design without the big picture
- ★ Tests must also be rewritten when requirements change
- ★ Other tests are needed too
- ★ Tests are as bad as code



#Swift2TDD

"Success is stumbling from failure to failure
with no loss of enthusiasm."

– Winston Churchill

Real Life



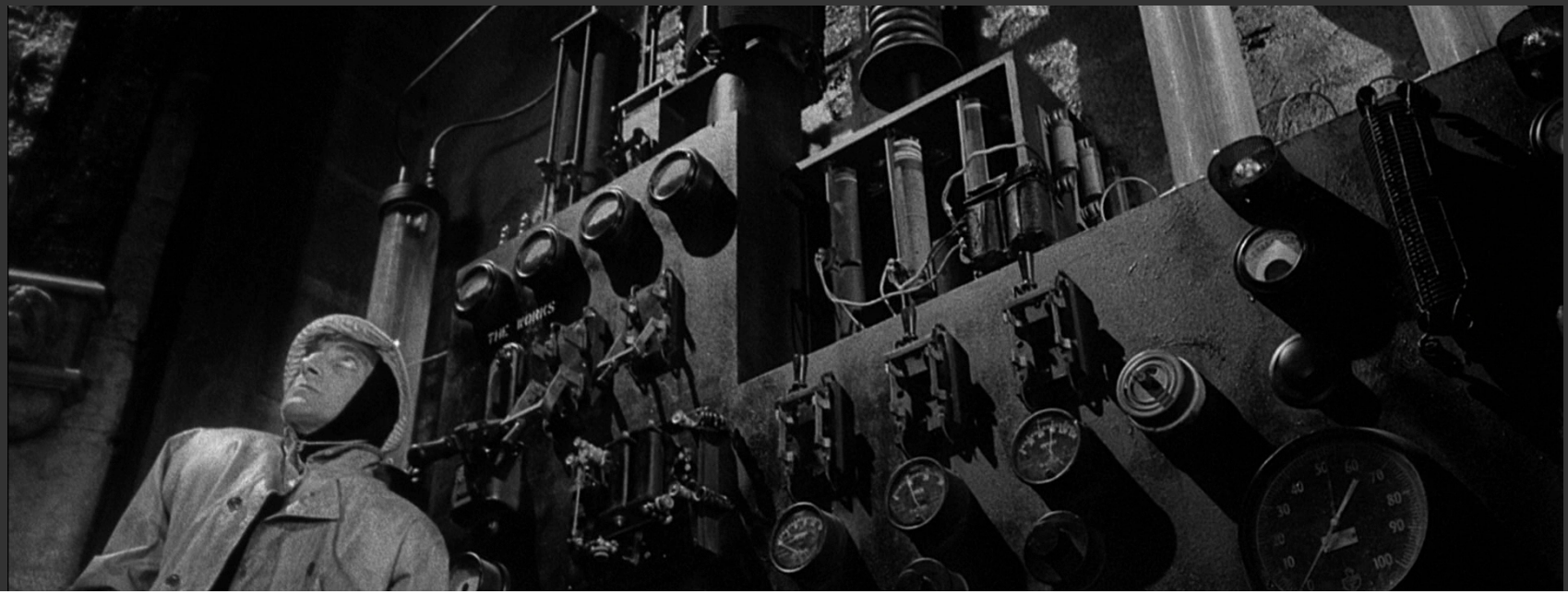
UI Testing



Persistence



Traits



TDD
+
Swift 2



Test the UI

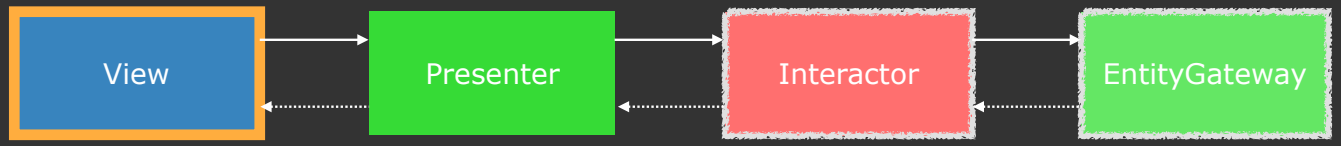
UI Testing

- ★ Xcode UI Testing ≠ Unit Testing
- ★ Nor is Kif, Calabash, or Cucumber (Integration)
- ★ Unit test is about isolation



#Swift2TDD

View



Make the view dumb

```
class SpeakerTableViewCell: UITableViewCell {

    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var nameLabel: UILabel!
    @IBOutlet weak var dateLabel: UILabel!
}


extension SpeakerTableViewCell: SpeakerCellProtocol {
    func displayName(name: String) {
        nameLabel.text = name
    }


    func displayTitle(title: String) {
        titleLabel.text = title
    }


    func displayDateSubmitted(date: String){
        dateLabel.text = date
    }
}
```



Testing a label

```
func testDisplayDateChangesLabelTextToAltValue() {
    let labelMock = LabelMock()
    sut.dateLabel = labelMock
    sut.displayDateSubmitted(speakerAltDateSubmitted)
    XCTAssertTrue(labelMock.setTextWasInvoked,
        "displayDate must set the text of the label.")
    XCTAssertEqual(labelMock.text, speakerAltDateSubmitted,
        "displayDate must set the text of the label to the provided text.")
}


// MARK: - Stubs & Mocks

class LabelMock: UILabel {

    // MARK: - Properties

    var setTextWasInvoked = false
    override var text: String? {
        didSet {
            setTextWasInvoked = true
        }
    }
}
```



Test despite the persistence

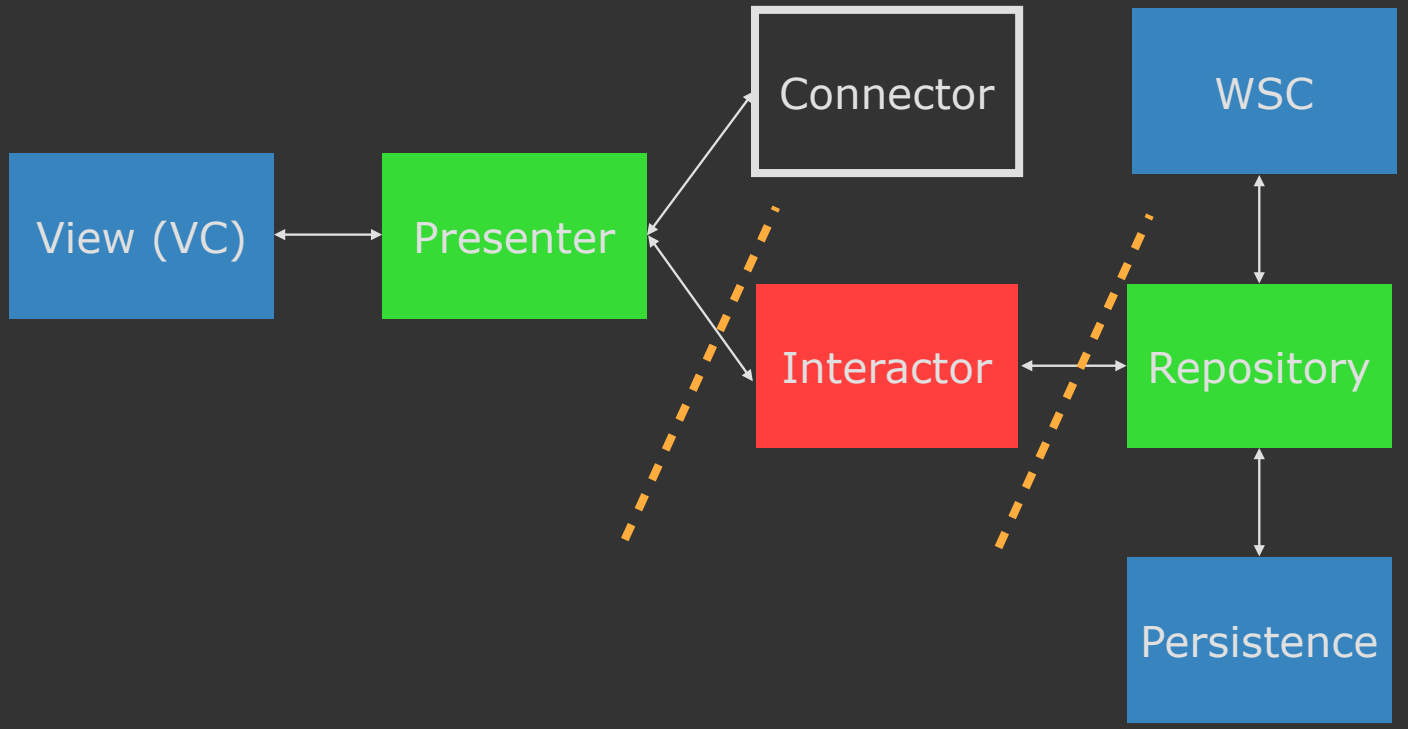
Testing Business Logic

- ★ Persistence seems to be a requirement for testing the logic
- ★ But it can be abstracted
- ★ And the data from the entity gateway



#Swift2TDD

Clean Architecture



Isolate the user story

```
class ShowAllSpeakersInteractor {

    // MARK: - Properties

    let entityGateway: EntityGatewayProtocol
    weak var presenter: SpeakersListPresenterProtocol?


    // MARK: - Initializers

    init(entityGateway: EntityGatewayProtocol) {
        self.entityGateway =  entityGateway
    }
}


extension ShowAllSpeakersInteractor: ShowAllSpeakersInteractorProtocol {
    func showAllSpeakers() {
        let entities = entityGateway.fetchAllSpeakers()
        let displayData = entities.map({entity in
            return SpeakerDisplayData(speaker: entity)})
        presenter?.presentAllSpeakers(displayData)
    }
}
```



Testing a use case

```
func testEntityIstransformedIntoDisplayData() {
    entityGateway.entities = [Speaker(name: speakerMainName,
        title: speakerMainTitle,
        synopsis: speakerMainSynopsis,
        dateSubmitted: speakerMainDateSubmitted)]

    sut.showAllSpeakers()

    guard presenter.displayData.count == entityGateway.entities.count
       else { XCTFail("There must be one and only one display data per each
   entity of the repo.")
        return }

    let speakerData = presenter.displayData[0]
    XCTAssertEqual(speakerData.name, speakerMainName,
        "Speaker name must be copied onto the display data.")
    XCTAssertEqual(speakerData.title, speakerMainTitle,
        "Speaker title must be copied onto the display data.")
    XCTAssertEqual(speakerData.dateSubmitted, speakerMainDateSubmitted,
        "Speaker submitedDate must be copied onto the display data.")
}
```



Traits

Protocol extensions

```
protocol SegueHandlerTypeProtocol {
    typealias SegueIdentifier: RawRepresentable
}


extension SegueHandlerTypeProtocol where
    Self: UIViewController,
SegueIdentifier.RawValue == String {

    func performSegueWithIdentifier(segueIdentifier: SegueIdentifier, sender:
AnyObject?) {
        performSegueWithIdentifier(segueIdentifier.rawValue, sender: sender)
    }


    func segueIdentifierForSegue(segue: UIStoryboardSegue) -> SegueIdentifier {
        guard let identifier = segue.identifier,
            segueIdentifier = SegueIdentifier(rawValue: identifier)
            else { fatalError("Invalid segue identifier \(segue.identifier).") }
        return segueIdentifier
    }
}
```



As explained in WWDC 2015 Swift in Practice

Testing a Trait

```
    var sut: SegueHandlerTypeViewControlerMock!

    class SegueHandlerTypeViewControlerMock: UIViewController,
SegueHandlerTypeProtocol {

        // MARK: - Properties
        var identifierToPerformSegue: String?
        var senderPerformingSegue: AnyObject?

        enum SegueIdentifier: String {
            case FirstSegue = "FirstSegueString"
            case SecondSegue = "SecondSegueString"
        }


        // MARK: - Mocks & Stubs

        override func performSegueWithIdentifier(identifier: String, sender:
AnyObject?) {
            identifierToPerformSegue = identifier
            senderPerformingSegue = sender
        }
    }
```



Testing a trait (2)

```
    func testPerformSegueWithIdentifierInvokesClassicVersion() {

sut.performSegueWithIdentifier(SegueHandlerTypeViewControlerMock.SegueIdentifier.FirstSeg
ue, sender: nil)
        XCTAssertNotNil(sut.identifierToPerformSegue,
            "Classic version of performSegueWithIdentifier must be invoked by the one in
the protocol extension.")
    }


    func testPerformSegueWithIdentifierPassesRawValueOfFirstIdentifier() {

sut.performSegueWithIdentifier(SegueHandlerTypeViewControlerMock.SegueIdentifier.FirstSeg
ue, sender: nil)
        XCTAssertEqual(sut.identifierToPerformSegue, firstSegueString,
            "Identifier must correspond to the raw value of the enum case.")
    }


    func testPerformSegueWithIdentifierPassesRawValueOfSecondIdentifier() {

sut.performSegueWithIdentifier(SegueHandlerTypeViewControlerMock.SegueIdentifier.SecondSe
gue, sender: nil)
        XCTAssertEqual(sut.identifierToPerformSegue, secondSegueString,
            "Identifier must correspond to the raw value of the enum case.")
    }
```



"I have not failed. I've just found 10,000 ways that won't work."

– Thomas A. Edison

Thank you!



@jdortiz

#Swift2TDD



goto; conference



Please

Remember to rate this session

Thank you!

Join the conversation #gotocph