TableView Adapter in an iOS project

Pirush

Growth Session X (#10) - December 21-22 2017

Intro

- Introduce TableView adapter to RP project
- Refactor Room selection view controller with the adapter
- Assess the usage of the adapter on the production level

Achievements and progress

- Migrated Room selection’s cells to Swift
- Room selection view controller works as before the integration

Pros & Cons

Pros

- Can easily replace implementation of UITableViewDelegate & DataSource
- Can reduce the code and repetitive task
- Can help avoiding developer’s error as it’s kinda typed-safe
- The adapter make table view cell dependent to its view model

Cons

- Not compatible with ObjC
- Good documentation is required
- ReactiveCocoa dependent

How Room selection view controller was implemented

```swift
//MARK:- tableview delegate & datasource

private func registerCell<T: UITableViewCell>(of cellClass: T.Type) {
    let identifier = String(describing: cellClass)
    let nib = UINib(nibName: identifier, bundle: nil)
    tableView.register(nib, forCellReuseIdentifier: identifier)
}

private func cellClass(for object: Any?) -> AnyClass? {
    guard let object = object else { return nil }
    switch object {
    case is RPBreakfastPicker:                      return RPBreakfastPickerCell.self
    case is RPCheckinCheckoutDatesPicker:           return RPCheckinCheckoutDatesPickerCell.self
    case is RPGuestsCountPicker:                    return RPGuestsCountPickerCell.self
    case is RPRoomCountPicker:                      return RPRoomCountPickerCell.self
    case is RPTotalDisplayItem:                     return RPTotalDisplayCell.self
    case is RPAdditionalGuestPriceDisclaimerItem:   return RPAdditionalGuestPriceDisclaimerCell.self
    case is RPSundryItemsPicker:                    return RPAdditionalItemsCell.self
    default:                                        assert(false, "no matching cell found")
    }
    return nil
}

func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return form.numberOfItems
}

func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
    if (form.indexPathsOfHiddenItems.contains { $0 == indexPath }) {
        return 0.0
    }

    let item = form.item(at: indexPath)

    if let heightCalculating = cellClass(for: item) as? RPHeightCalculating.Type {
        return heightCalculating.preferredInstanceHeight()
    }

    return UITableViewAutomaticDimension
}

func tableView(_ tableView: UITableView, estimatedHeightForRowAt indexPath: IndexPath) -> CGFloat {
    if (form.indexPathsOfHiddenItems.contains { $0 == indexPath }) {
        return 0.0
    }

    let item = form.item(at: indexPath)

    if let heightCalculating = cellClass(for: item) as? RPHeightCalculating.Type {
        return heightCalculating.preferredInstanceHeight()
    }

    if let height = heightAtIndexPath[indexPath] {
        return height
    }

    return 95.0 // hotel image starting height
}
```

```swift
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let item = form.item(at: indexPath)
    guard
        let cellClass = cellClass(for: item),
        let cell = tableView.dequeueReusableCell(withIdentifier: String(describing: cellClass))
    else { return UITableViewCell() }

    if let checkInCheckOutCell = cell as? RPCheckinCheckoutDatesPickerCell {
        checkInCheckOutCell.datesSelectionExecutor = self
    }

    if let objectConsumingCell = cell as? RPObjectConsuming {
        objectConsumingCell.apply(item)
    }
    cell.selectionStyle = .none
    return cell
}

func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    if form.item(at: indexPath) is RPSundryItemsPicker {
        didRequestAddingItem?()
    }
}

func tableView(_ tableView: UITableView, willDisplay cell: UITableViewCell, forRowAt indexPath: IndexPath) {
    heightAtIndexPath[indexPath] = cell.frame.height
}
```

```swift
private func setupTableView() {
    tableView.dataSource = self
    tableView.delegate = self
    tableView.rowHeight = UITableViewAutomaticDimension

    registerCell(of: RPBreakfastPickerCell.self)
    registerCell(of: RPCheckinCheckoutDatesPickerCell.self)
    registerCell(of: RPGuestsCountPickerCell.self)
    registerCell(of: RPRoomCountPickerCell.self)
    registerCell(of: RPTotalDisplayCell.self)
    registerCell(of: RPAdditionalGuestPriceDisclaimerCell.self)
    registerCell(of: RPAdditionalItemsCell.self)
}
```

How it is with the adapter

```
private func setupTableViewAdapter() {
    tableViewAdapter = TableViewAdapter(for: tableView, dataAdapter: form.dataAdapter)
    tableViewAdapter.rowHeight = UITableViewAutomaticDimension
    tableViewAdapter.register(CheckInCheckOutDatesTableViewCell.self)
    tableViewAdapter.register(RoomPickerTableViewCell.self)
    tableViewAdapter.register(GuestPickerTableViewCell.self)
    tableViewAdapter.register(BreakfastPickerTableViewCell.self)
    tableViewAdapter.register(TotalDisplayTableViewCell.self)

    tableViewAdapter.didSetItem
        .take(during: tableView.reactive.lifetime)
        .observeValues { [weak self] (cell, _, indexPath) in
            if let cell = cell as? CheckInCheckOutDatesTableViewCell {
                cell.datesSelectionExecutor = self
            }
            cell.selectionStyle = .none
    }

    tableViewAdapter.didSelectCell
        .take(during: tableView.reactive.lifetime)
        .observeValues { (cell, row, _) in
            switch row {
            case let row as Row<RoomPickerTableViewCell>:
                print(">>> room name: \(row.item.room.title ?? "-")")
            default:
                break
            }
    }
}
```

Next Steps

- Some bugs to be fixed
- Best practice & documentation
- Usage of the adapter’s other features
- More test cases needed

Thanks!

Contact Nimbl3

hello@nimbl3.com

399 Sukhumvit Road, Interchange 21
Klongtoey nua, Wattana
Bangkok 10110

20th Floor, Central Tower
28 Queen's Road
Central, Hong Kong

nimbl3.com