**IGListKit**

Migration guide: <https://github.com/Instagram/IGListKit/blob/master/Guides/Migration.md>

**Generating your models using remodel**

Implement IGListDiffable sẽ tự động implement hash, isEqual, description, diffIdentifier và isEqualToDiffableObject. Remodel (?)

**Get started**

**Create section controller**

Section class phải implement **IGListSectionController** (và override **cellForItemAtIndex** và **sizeForItemAtIndex**)

**class** LabelSectionController: ListSectionController {

**override** **func** sizeForItem(at index: Int) **->** CGSize {

**return** CGSize(width: collectionContext**!.**containerSize**.**width, height: 55)

}

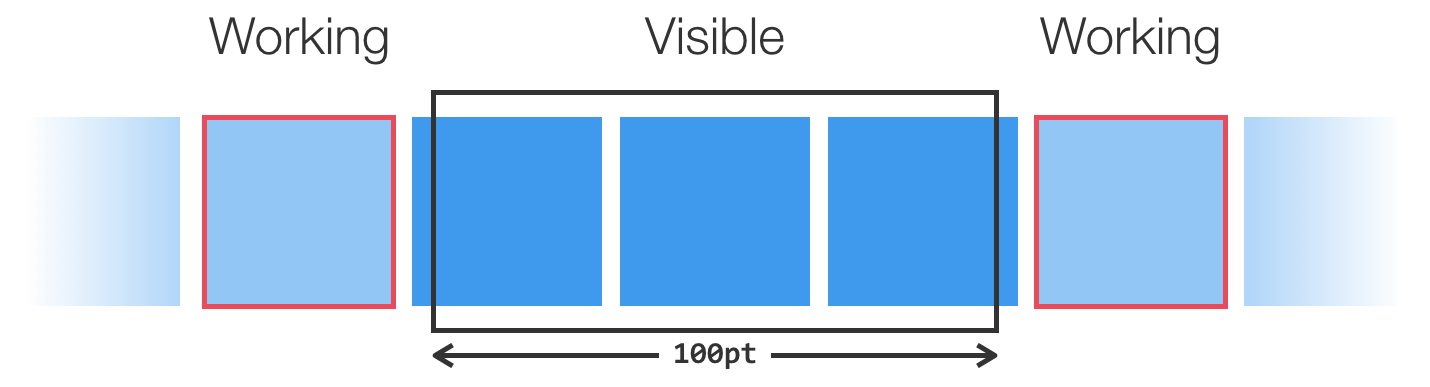
**override** **func** cellForItem(at index: Int) **->** UICollectionViewCell {

**return** collectionContext**!.**dequeueReusableCell(of: MyCell**.self**, for: **self**, at: index)

}

}

**Creating the UI**

**WorkingRangeSize**: là range của section controller chưa được hiển thị nhưng nó gần với vùng hiển thị. Nó cho phép xử lý trước phần content trước khi nó được hiển thị (vd: download images). Nó có thể là multable của **visible** width hoặc height tuỳ vào scroll-direction

**Connecting the data source**

**func** objects(**for** listAdapter: ListAdapter) **->** [ListDiffable] {

*// this can be anything!*

**return** [ "Foo", "Bar", 42, "Biz" ]

}

**func** listAdapter(\_ listAdapter: ListAdapter, sectionControllerFor object: Any) **->** ListSectionController {

**if** object **is** String {

**return** LabelSectionController()

} **else** {

**return** NumberSectionController()

}

}

**func** emptyView(**for** listAdapter: ListAdapter) **->** UIView? {

**return** **nil**

}

Kết nối data source

adapter**.**dataSource **=** **self**

**Immutability**

Ta nên gán dữ liệu với let (không thay đổi được). Nếu ta trả về dữ liệu với var và sau đó ta thay đổi dữ liệu thì IGListKit sẽ không thể xác định được thay đổi dữ liệu và dữ liệu thay đổi sẽ bị mất. Ta nên gán 1 instance mới cho dữ liệu khi muốn thay đổi dữ liệu.

**Diffing**

IGListKit sử dụng thuật toán trong bài viết [A technique for isolating differences between files](http://dl.acm.org/citation.cfm?id=359467&dl=ACM&coll=DL). Thuật toán này sử dụng công nghệ Longest common subsequence để tìm ra điểm khác biệt nhỏ nhất giữa các collection để tìm ta tất cả các **inserts, delete, update, moves.**

Để thêm custom, diffable models, ta phải implement ListDiffable với diffIdentifier() và isEqual(). DiffIdentifier() không nên được thay đổi

**class** User {

**let** primaryKey: Int

**let** name: String

*// implementation, etc*

}

Giả sử server trả về user với nội dung

**let** shayne **=** User(primaryKey: 2, name: "Shayne")

nhưng sau đó user lại đổi tên thành Ann

**let** ann **=** User(primaryKey: 2, name: "Ann")

lúc này cả Ann và Shayne đều có primaryKey = 2 nhưng tên lại khác nhau nên 2 object này là không equal.

**extension** User: ListDiffable {

**func** diffIdentifier() **->** NSObjectProtocol {

**return** primaryKey

}

**func** isEqual(toDiffableObject object: Any?) **->** Bool {

**if** **let** object **=** object **as?** User {

**return** name **==** object**.**name

}

**return** **false**

}

}

Cách implement trên sẽ không update lại UI nếu object có cùng *primaryKey* và *name* kể cả khi nó là 2 instances khác nhau. Nó giúp tránh việc update UI không cần thiết khi ta cung cấp instance mới cho collection (giả sử mỗi lần update nội dung của collection mà ta khởi tạo các instance mới thì sẽ tránh việc update trùng lặp)

**Note**: nếu muốn reload lại cell của 1 object thì cần return false ở hàm isEqual()

**Diffing outside of IGListKit**

Ta có thể sử dụng Diffing mà không cần phụ thuộc vào ListAdapter hay UICollectionView

let results = ListDiff(oldArray: self.dataArray, newArray: newArray, option: .equality)

self.dataArray = newArray // set new data into exiting array before updating tableview

tableView.beginUpdates()

tableView.deleteRows(at: results.deletes, with: .fade)

tableView.insertRows(at: results.inserts, with: .automatic)

tableView.reloadRows(at: results.updates, with: .none)

tableView.endUpdates()

ListDiff giúp tìm những điểm khác biệt của 2 array từ đó ta có thể tự insert, reload, delete các cells mong muốn

**Display delegate**

Cho phép nhận events về section controller và các cells

**Custom Updaters**

Implement IGListUpdatingDelegate

**IGListDiffable and Equality**

Mục đích của diffIdentifier là để phân biệt các object với nhau, isEqual() để phục vụ việc display object có nên reload hay không.

Ta có thể sử dụng chính object như identifier và sử dụng isEqual của superclass??

**-** (id**<**NSObject**>**)diffIdentifier {

**return** self;

}

**-** (BOOL)isEqualToDiffableObject**:**(id**<**IGListDiffable**>**)object {

**return** [self isEqual:object];

}

**Modeling and binding**

Chung: (**ListAdapterDataSource** có thể sử dụng **ListBindingSectionController** hoặc **ListSectionController** để setup section)

* Ta phải tạo 1 **ListAdapter** để gán **collectionView** với **dataSource** của **ListAdapter**, CollectionView của ViewController với CollectionView của **ListAdapter**
* View Controller sẽ implement **ListAdapterDataSource**, với objects(for) là dữ liệu của colllectionView, **listAdapter(\_, sectionControllerFor)** là section controller ứng với object, **emptyView** là view hiển thị khi không có section nào
* **Header** và **footer** view được khai báo qua **supplementaryViewSource**, **supportedElementKinds()** có thể return array gồm **UICollectionView.elementKindSectionHeader** và **UICollectionView.elementKindSectionFooter**

Binding dữ liệu từ **ListAdapterDataSource** đến **ListBindingSectionController<Element>**

Tóm tắm:

* Các section controller sẽ implement từ **ListBindingSectionController** và gán **dataSource** bằng cách implement **ListBindingSectionControllerDatSource**
* Với mỗi kiểu cell ta sẽ tạo 1 object View Model tương ứng, các View Model phải implement từ **ListDiffable**
* Các cell class phải implement **ListBindable** để có thể binding dữ liệu
* Dữ liệu khi được bắn qua section controller sẽ vào **sectionController(\_, viewModelsFor)** ta dùng hàm này để bóc tách dữ liệu thành các View Model (dùng để binding đến các cell)
* Dữ liệu sau khi được bóc tách sẽ truyền qua hàm **sectionController(\_, cellForViewModel)** khởi tạo cell và bindind dữ liệu đến các cell

<https://instagram.github.io/IGListKit/modeling-and-binding.html>

**Sử dụng ListSectionController**

Section controller sẽ implement từ **ListSectionController**

Tóm tắt:

* Ta sẽ phải tạo 1 property để lưu lại dữ liệu được bắn từ **ListAdapterDataSource** qua, dữ liệu sẽ được truyền qua func **didUpdate(to object:)**

|  |  |
| --- | --- |
| **ListSectionController** | **ListBindingSectionController** |
|  | Các cell phải implement **ListBindable** và các cell được binding thông qua viewModel (các viewModel này phải implement **ListDriffable**) |
| Dữ liệu sẽ xử lý trực tiếp trên ListSectionController | Dữ liệu được binding đến Section sẽ được truyền qua dataSource |
| Số lượng cell trong section được khai báo qua hàm **numberOfItems()** | Số lượng cell sẽ phụ thuộc vào số lượng viewModel được return trong hàm **sectionController(\_, viewModelsFor)** nếu như không override **numberOfItems()** |
|  | Kế thừa từ **ListSectionController** và override **didUpdate(object)** |
| Khi muốn move cell từ vị trí A đến B thì chỉ cần remove dữ liệu của vị trí A rồi insert vào vị trí B (dữ liệu để rend cell trong SectionController) | Khi muốn move cell từ vị trí A đến B ta dùng **super.moveObject(from: A, to: B)** và ***data*.swapAt()** để move cell. Chú ý: khi move object ra 1 section khác bị crash |

**ListWorkingRangeDelegate**

Nhận các event khi sectionController **WillEnterWorkingRange** và **DidExitWorkingRange** (Working range được set khi khởi tạo **ListAdapter**), ta có thể dùng để xử lý dữ liệu trước khi nó hiển thị. Vd: load ảnh

**ListScrollDelegate**

didEndDragging, willBeginDragging, didScroll

**ListSingleSectionControllerDelegate**: mỗi section có 1 cell, có thể bất được event didselect ngay trong viewController

Link:

* [https://instagram.github.io/IGListKit](https://instagram.github.io/IGListKit/getting-started.html)