Protocol
Extension
Delegation

Protocol

- **协议(protocal)**定义了用于实现某个任务或功能的一系列**方法**、**属性**和其它的种种需求。然后这个协议被**类(class)或结构体(struct)或枚举类(enum)实现**。

Protocol 是什么

- **Protocal中文翻译为协议: 就是双方必需准守的意思。**

- **使用了Protocol的Class在定义时，一定要定义和实现Protocol里面写下的method和Propertity**

Protocol 怎么用

```
protocol MustSaySomeThing {
    func hello() -> Void
    func bye() -> Void
}
```

```
class Roommate: MustSaySomeThing {
    func hello() -> Void {
        //么么哒
    }

    func bye() -> Void{
        //再见
    }
}
```

```
16 class Roommate: MustSaySomeThing {
17
18
19     func bye() -> Void{
20         //再见
21     }
22 }
23
```

Type 'Roommate' does not conform to protocol 'MustSaySomeThing'

Extension

- **扩展**就是为一个**已有的**类、结构体、枚举类型或者协议类型**添加新功能**。这包括在没有权限获取原始源代码的情况下扩展类型的能力（即逆向建模）。

Extension 是什么

- Extension的使用就是能在原有的Class里面追加定义一些method或者propertity。
- 扩展那些你不能接触的类。譬如 UIView 和 UIImage

Extension 怎么用

```
class Roommate: MustSaySomeThing {
    func hello() -> Void {
        //么么哒
    }

    func bye() -> Void{
        //再见
    }
}
```

```
extension Roommate{
    func thankU() -> Void{
        //谢谢你
    }
}

let me = Roommate()
me.thankU()//谢谢你
```

Protocol 高级用法

```
protocol MustSaySomeThing {
    func hello() -> Void
    func bye() -> Void
}

extension MustSaySomeThing{
    func hello() -> Void{
        //么么哒
    }
    func bye() -> Void{
        //再见
    }
}
```

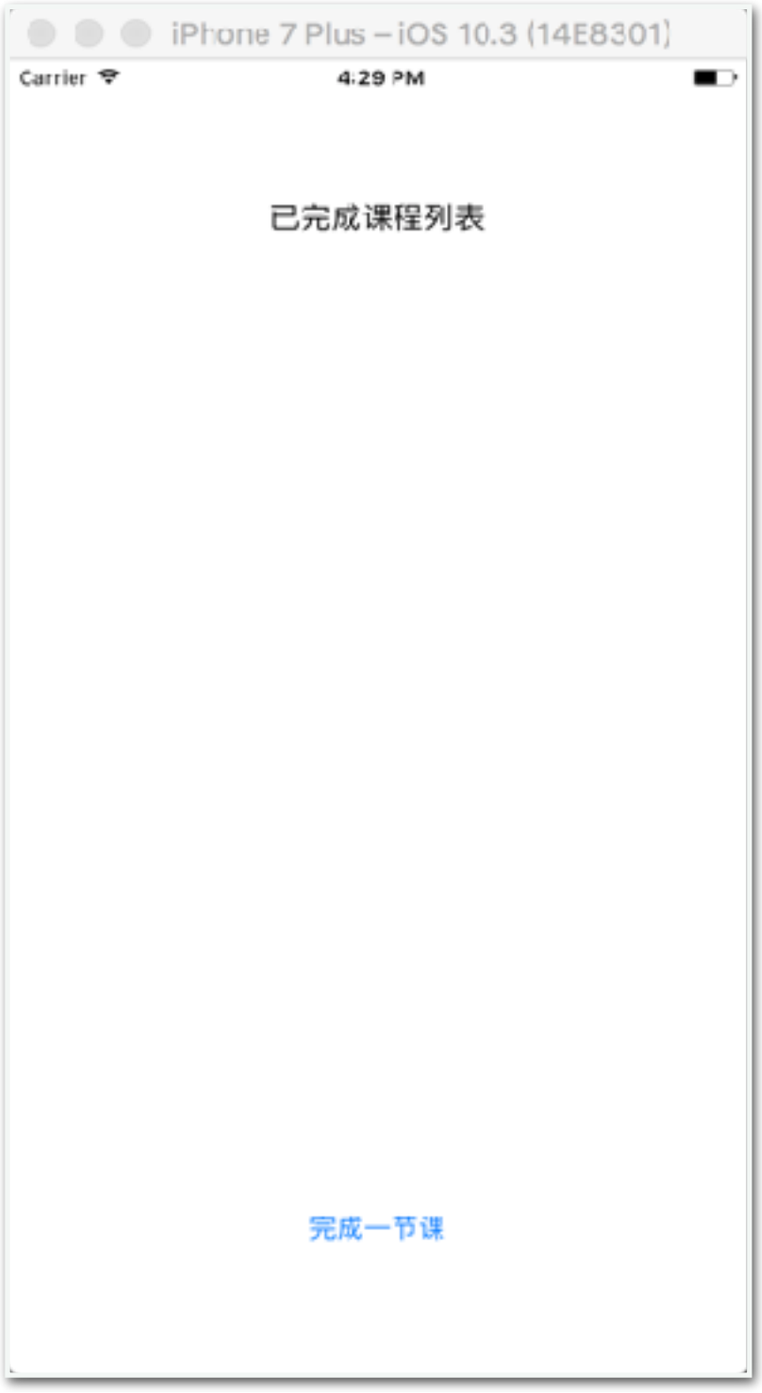
```
class Roommate: MustSaySomeThing {
    func bye() -> Void{
        //good bye 永远不见
    }

}

let me = Roommate()
me.hello() //么么哒
me.bye() //good bye 永远不见
```

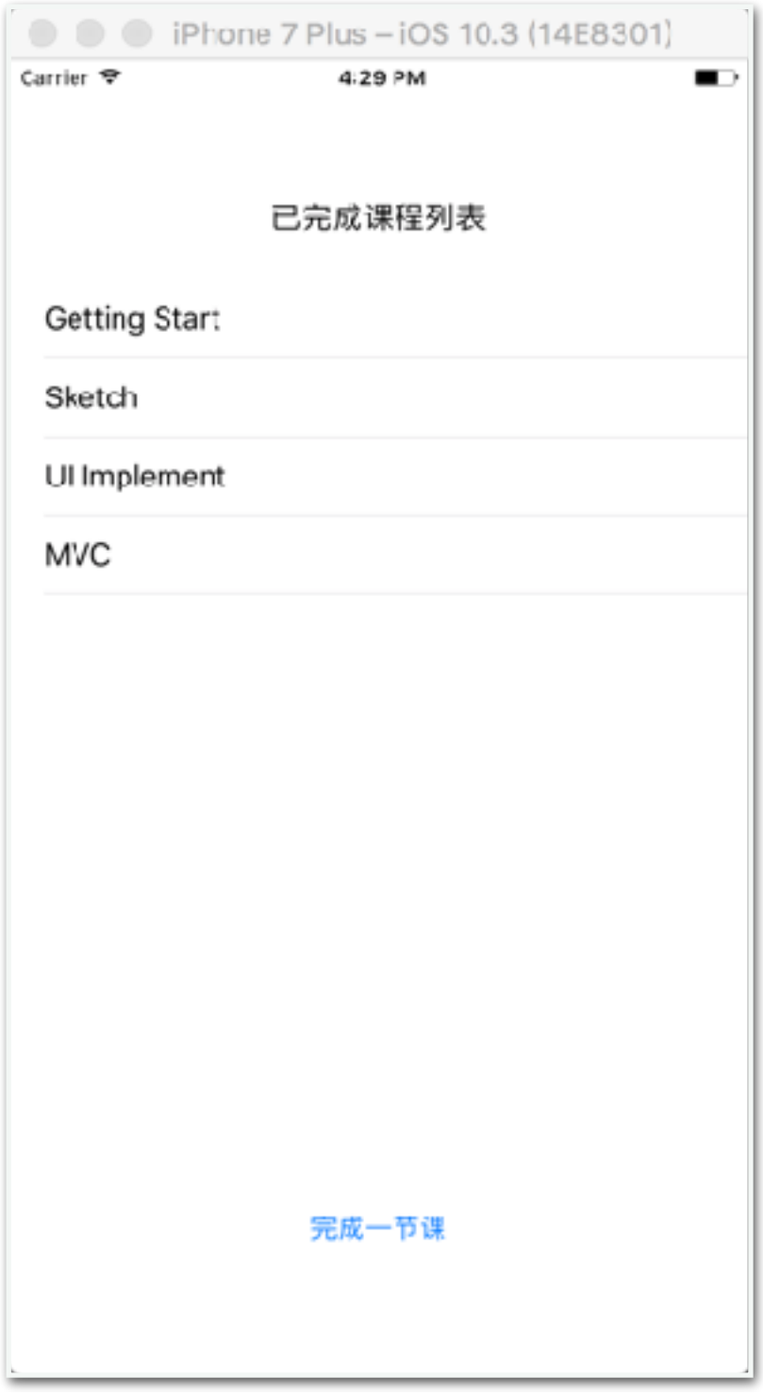
Delegation

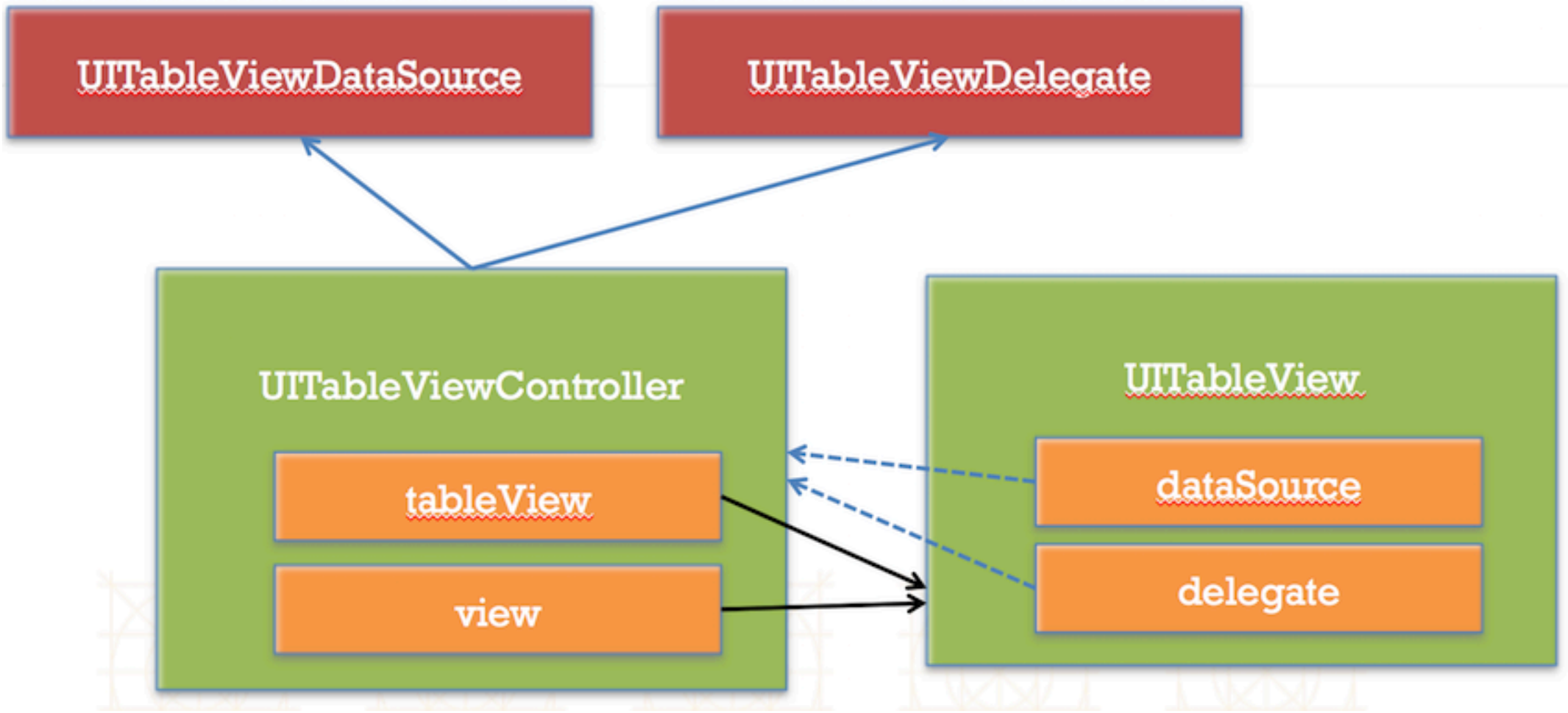
- **委托（delegation）**是一种支持**类或结构体**将一**部分职责**委托给**另一种类型的实例**来完成的设计模式。这种设计模式是通过定义一个**封装了委托职责的协议**来实现的。然后被委托的实例通过遵从该协议来确保完成了委托的任务。

- 委托可以用于回应某一个特定的动作（action），或者从不知其内部实现的外部源来获得数据。

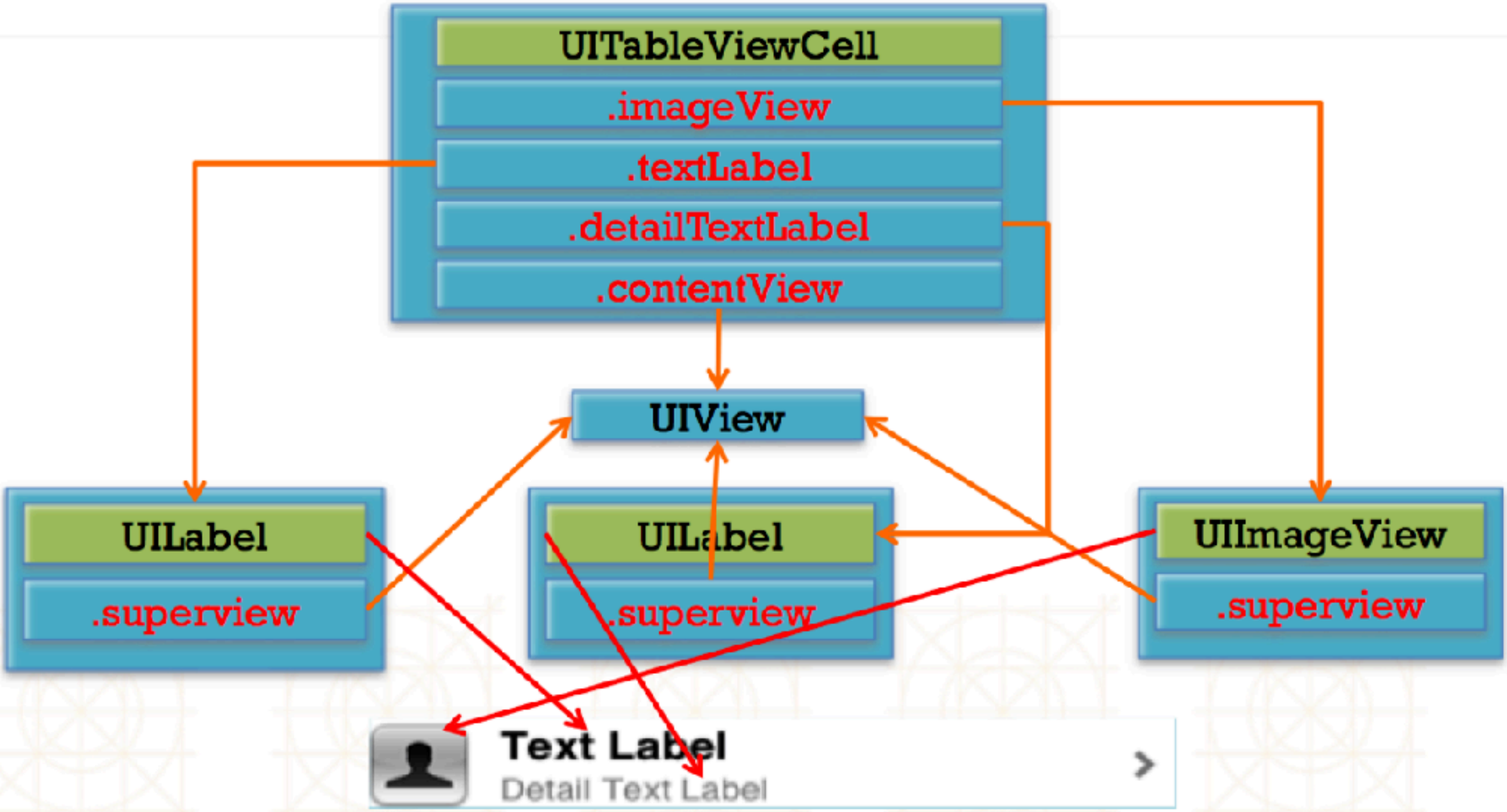
已完成课程列表

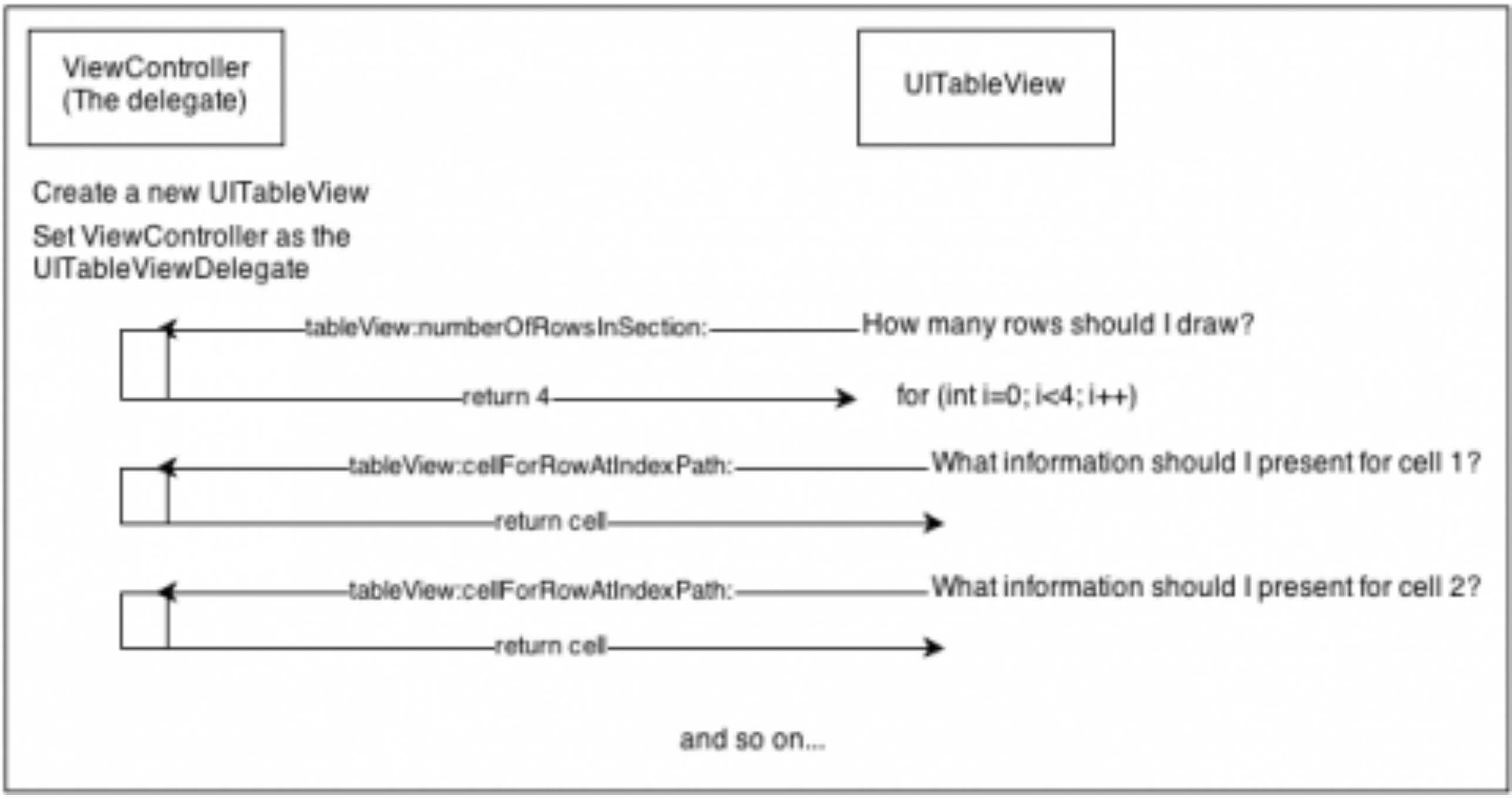

iPhone 7 Plus – iOS 10.3 (14E8301)
Carrier
4:29 PM
已完成课程列表
完成一节课


iPhone 7 Plus – iOS 10.3 (14E8301)
Carrier
4:29 PM
已完成课程列表
Getting Start
Sketch
UI Implement
MVC
完成一节课

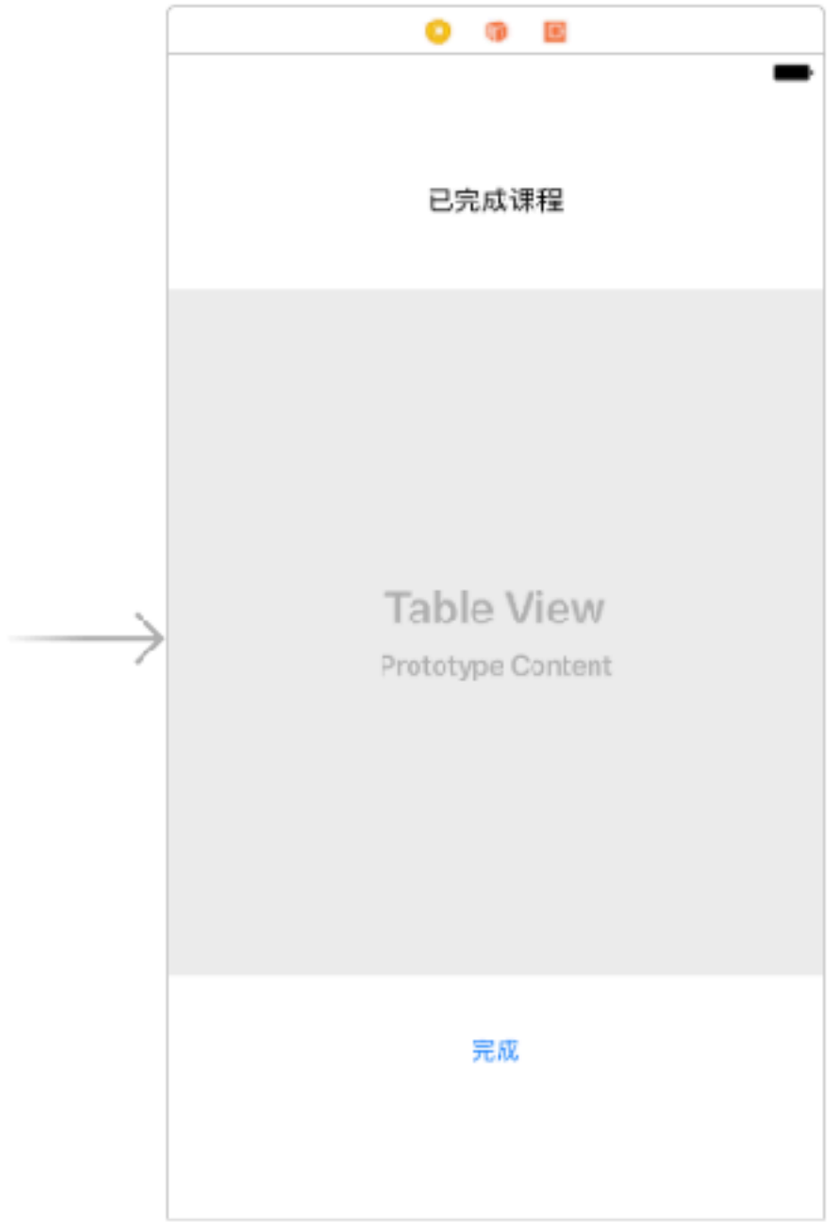


- **UITableViewDataSource:** 作为数据源，可以查询一共有多少行数据以及每一行显示什么数据等。
- **UITableViewDelegate:** 在UITableView触发某事件时做出相应的处理，比如选中了某一行。


UITableViewCell
.imageView
.textLabel
.detailTextLabel
.contentView
UIView
UILabel
.superview
UILabel
.superview
UIImageView
.superview
Text Label
Detail Text Label


ViewController
(The delegate)
UITableView
Create a new UITableView
Set ViewController as the
UITableViewDelegate
tableView:numberOfRowsInSection:
How many rows should I draw?
return 4
for (int i=0; i<4; i++)
tableView:cellForRowAtIndexPath:
What information should I present for cell 1?
return cell
tableView:cellForRowAtIndexPath:
What information should I present for cell 2?
return cell
and so on...

已完成课程列表—storyboard



- 在storyboard中添加title、Table View、添加课程的button

- 将button(**action**)和Table View(**outlet**)链接到view controller中

- 运行（如果和storyboard中排版不同，请检查机型）

UITableViewDelegate需要实现的两个方法

按住command，点击UITableViewDataSource即可查看源代码

- tableView(_:numberOfRowsInSection:)方法返回每个Section中显示多少行内容
- tableView(_:cellForRowAtIndexPath:)方法会逐个创建每行的内容，包括专辑标题和它的值。

已完成课程列表—ViewController中代码实现

- 实现UITableViewDataSource协议中必须实现的两个方法
  - let cell = UITableViewCell(style: .default, reuseIdentifier: “1”)
  - cell.textLabel!.text = "\(lecture[indexPath.row])"
- classNo:记录当前已完成课程数量
  - var lecture = ["Getting Start", "Sketch", "UI Implement", "MVC"] （记录课程数量，也即是所有可以展示的行数）
  - 编写点击button后应该发生的逻辑（classNo加一，刷新界面(lessonView.reloadData())，健壮性(classNo<lecture.count)）
  - lessonView.dataSource = self：使得tableview从当前controller中读取数据(写在viewdidload中)

THANKS