TableView編輯

tableView的儲存格根據isEditing屬性,有一般狀態和編輯狀態。如果是編輯狀態，一般會影響下列的一個或多各外觀：

1.編輯控制項

至少有一個編輯控制，例如有一個Deletion按鈕出現在儲存格左邊。

2.內縮：

通常儲存格內容會內縮來產生一個足夠允許編輯控制項放在左邊的空間。如果沒有編輯控制項，使用delegate's 'S tableView(_:shouldIndentWhileEditingRowAt:)

可以預防儲存格向右內縮。

3.改變一個附屬的View

儲存格的附屬View會根據editingAccessoryType or editingAccessoryView

屬性改變。如果都沒指定將會消失。當在編輯狀態時，附屬的View也會消失。

就像選取，你可以直接設定儲存格的isEditing的屬性。但是你將更喜歡使用tableView來管理tableView的isEditing 屬性，通常是執行tableView的set- Editing(_:animated:)

。然後所有的cell進入編輯狀態。

一個儲存格在編輯模式也可以被選取藉由使用tableView的 allows- SelectionDuringEditing or allowsMultipleSelectionDuringEditing

設定為true。

讓tableView進入編輯模式，通常是由使用者控制。一個傳統的介面是提供一個編輯的按鈕讓使用者控制。在導覽介面上可以加入一個UIBarButton Item。

```
let b = UIBarButtonItem(barButtonSystemItem: .edit,
        target: self, action: #selector(doEdit))
    self.navigationItem.rightBarButtonItem = b
```

使用action處理者來回應，讓tableView進入到編輯模式，最簡單的寫法如下:

```
func doEdit(_ sender: Any?) {
        self.tableView.setEditing(true, animated:true)
}
```

但是這樣不能解決退出編輯模式。所以下面是一個標準的解決方案，解決編輯和完成編輯的切換：

```
func doEdit(_ sender: Any?) {
        var which : UIBarButtonSystemItem
        if !self.tableView.isEditing {
            self.tableView.setEditing(true, animated:true)
            which = .done
        } else {
            self.tableView.setEditing(false, animated:true)
            which = .edit
        }
        let b = UIBarButtonItem(barButtonSystemItem: which,
            target: self, action: #selector(doEdit))
        self.navigationItem.rightBarButtonItem = b
    }
```

另一種簡單的方法是使用UIViewController的editbuttonItem。

```
self.navigationItem.rightBarButtonItem = self.editButtonItem
```

當tableView進入到編輯模式。tableView將執行dataSource和delegate
tableView(_:canEditRowAt:) dataSource
詢問是否可以編輯。

tableView(_:editingStyleForRowAt:) to the delegate
儲存格出現的編輯樣式
.delete
.insert
.none
如果使用者點選一個insert button或一個delete button,tableView將執行
dataSourcetableView(_:commit:forRowAt:)。這裏將真正執行插入和刪除儲存格的動作，另外也必需修改資料來源。tableView有下列修改儲存格的方法:

- insertRows(at:with:)
- deleteRows(at:with:)
- insertSections(_:with:)
- deleteSections(_:with:)
- moveSection(_:toSection:)
- moveRow(at:to:)

自訂Action按鈕

當使用者由左向右滑動儲存格，將會顯示Delete按鈕。使用者按下編輯鈕和點選減號按鈕，一樣可以得到相同的效果。可以手動加入更多在contentView下可以顯示的按鈕。

實作tableView delegate方法tableView(_:editActionsForRowAt:) 和回傳UITableViewRowAction陣列可以修改tableView的rowAction, 由右往左顯示，如果傳出為nil，將會得到預設的Delete按鈕。要建立一個rowAction，可以使用init(style:title:handler:) ,參數如下：

1.style:
UIableViewRowActionStyle,可以是.default或.normal。如果使用default，是一個紅色背景的action按鈕就像delete按鈕。.normal是一個灰色按鈕。可以使用button's backgroundColor來改變背景色。

2 .title:
按鈕的文字
3 .handler:
一個closure，當按鈕被按時，將會執行這個closure。這個closure實作時，必需提供2個參數，一個是rowAction參考，一個是這個儲存格的indexPath。

如果你想要使用者能夠滑動儲存格來顯示這些按鈕，你必需實作tableView(_:commit:forRowAt:),甚至可以實作空內容，只為了可以滑動。
實作範例如下:

```
override func tableView(_ tableView: UITableView,
      editActionsForRowAt indexPath: IndexPath) -> [UITableViewRowAction]? {
      let act = UITableViewRowAction(style: .normal, title: "Mark") {
action, ip in

          print("Mark") // in real life, do something here
      }
      act.backgroundColor = .blue
      let act2 = UITableViewRowAction(style: .default, title: "Delete") {
action, ip in

          self.tableView(self.tableView, commit:.delete, forRowAt:ip)
      }
      return [act2, act]
   }
```

可編輯儲存格

一個儲存格內可能有可以讓使用者編輯的內容，例如UITextField。現在必需讓儲存格，在一般模式時，文字欄位只可以顯示資料，在編輯模式時，文字欄位可以編輯資料。
想像現在維護一組姓名和電話號碼，當使用者按下編輯按鈕時，不想在編輯模式的儲存格左邊顯示按鈕。

```
override func tableView(_ tableView: UITableView,
        editingStyleForRowAt indexPath: IndexPath)
        -> UITableViewCellEditingStyle {
return .none

}
```

如果UITextField的isEnabled是true，則UITextField是可以編輯的。現在有個最簡單的做法，就是自訂一個UITablbeViewCell class。我命名為MyCell, MyCell將和prototype cell一起設計，在prototype cell內放入一個UITextField，並且使用IBOutlet建立一個名為textField的屬性。並且在MyCell的class內override didTransition(to:):

```
class MyCell : UITableViewCell {
        @IBOutlet weak var textField : UITextField!
        override func didTransition(to state: UITableViewCellStateMask) {
            self.textField.isEnabled = state.contains(.showingEditControlMask)
            super.didTransition(to:state)
        }
}
```

在tableView的dataSource內，建立UITextField時，同時建立代理人是ViewController。

```
override func tableView(_ tableView: UITableView,
        cellForRowAt indexPath: IndexPath) -> UITableViewCell {
}

let cell = tableView.dequeueReusableCell(
    withIdentifier:"Cell", for: indexPath) as! MyCell
switch indexPath.section {
case 0:
    cell.textField.text = self.name
case 1:
    cell.textField.text = self.numbers[indexPath.row]
    cell.textField.keyboardType = .numbersAndPunctuation
default: break
}
cell.textField.delegate = self
return cell
```

在ViewController內實作當使用者按下Return按鈕時，讓textField結束編輯，並且讓keyboard消失。

```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.endEditing(true)
        return false
}
```

由於ViewController是textField的delegate，當textField停止編輯後，將會執行text-FieldDidEndEditing(_:)，我們必需找到這個textField是屬於那一個儲存格的。藉由repeat-while的方式向上尋找UITableViewCell，當找到後更新資料來源的內容。

```
func textFieldDidEndEditing(_ textField: UITextField) {
        // some cell's text field has finished editing; which cell?
        var v : UIView = textField
        repeat { v = v.superview! } while !(v is UITableViewCell)
        let cell = v as! MyCell
        // update data model to match
        let ip = self.tableView.indexPath(for:cell)!
        if ip.section == 1 {
            self.numbers[ip.row] = cell.textField.text!
} else if ip.section == 0 {
            self.name = cell.textField.text!
} }
```

新增儲存格

不可能每一個儲存格都有insert的按鈕，比較有可能是最後一個儲存格有insert的按鈕，其餘的則有刪除的按鈕。實作方法:

```
override func tableView(_ tableView: UITableView,
        editingStyleForRowAt indexPath: IndexPath)
        -> UITableViewCellEditingStyle {
}

if indexPath.section == 1 {
    let ct = self.tableView(
        tableView, numberOfRowsInSection:indexPath.section)
    if ct-1 == indexPath.row {
        return .insert
    }
    return .delete;
}
return .none
```

有姓名的儲存格沒有編輯控制符號，所以不需要向內縮排:

```
override func tableView(_ tableView: UITableView,
   shouldIndentWhileEditingRowAt indexPath: IndexPath) -> Bool {
        if indexPath.section == 1 {
        return true
        }
        return false

}
```

當使用者如果點選編輯控制，我們必需回應。如果刪除，執行刪除動作。如果是新增，新增一個儲存格。並記得要修改資料來源，並重新載入。

```
override func tableView(_ tableView: UITableView,
      commit editingStyle: UITableViewCellEditingStyle,
      forRowAt indexPath: IndexPath) {
         tableView.endEditing(true)
         if editingStyle == .insert {
            self.numbers += [""]
            let ct = self.numbers.count
            tableView.beginUpdates()
            tableView.insertRows(at: [IndexPath(row:ct-1, section:1)],
               with:.automatic)
            tableView.reloadRows(at: [IndexPath(row:ct-2, section:1)],
               with:.automatic)
            tableView.endUpdates()
            // crucial that this next bit be *outside* the updates block
            let cell = self.tableView.cellForRow(
               at: IndexPath(row:ct-1, section:1))
            (cell as! MyCell).textField.becomeFirstResponder()
         }
         if editingStyle == .delete {
         self.numbers.remove(at:indexPath.row)
         tableView.beginUpdates()
         tableView.deleteRows(at:[indexPath], with:.automatic)
         tableView.reloadSections(IndexSet(integer:1), with:.automatic)
         tableView.endUpdates()
         }

}
```