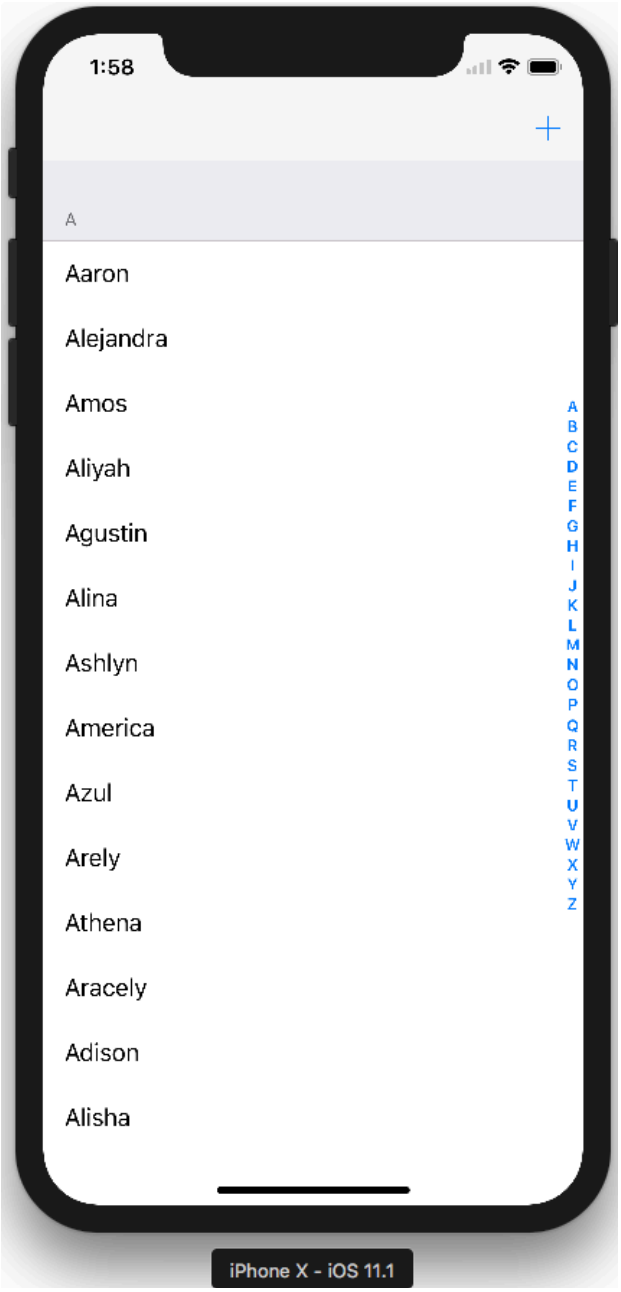
範例3_1Firebase_realtime_dataBase範例說明1



學習目地

- 建立Firebase DataBase
- 上傳大量List的方法
- 使用UITableView顯示多個Section

讀取plist的資料，並將plist資料上傳。

載入Firebase module並在在AppDelegate 建立Firebase 初始化, 在func application(_:, didFinishLaunchingWithOptions:)，加入Firebase初始化

```
import Firebase

FirebaseApp.configure();
```

在AppDelegate的func application(_:, didFinishLaunchingWithOptions:)檢查Firebase資料庫節點是否有資料。

```
//建立iphone2/userName節點參考
let usernameRef = Database.database().reference(withPath: "iphone2/userName");

//將節點參考加上一次監聽，實作Closue，取得dataSnapshot的資料
usernameRef.observeSingleEvent(of: .value) { (dataSnapshot:DataSnapshot) in
    //檢查節點是否沒有資料。
    if !dataSnapshot.hasChildren(){

    }
}
```

建立自訂Function uploadDataToFirebase(usernameRef,plistName)，負責下傳資料

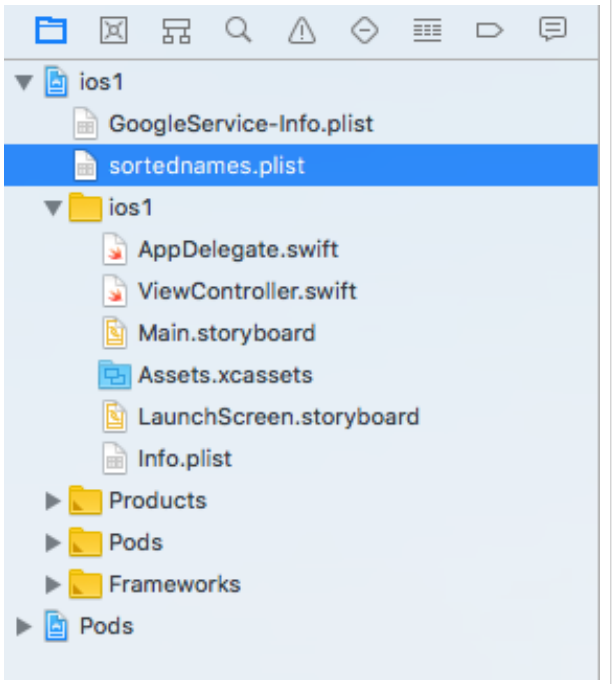
```
func uploadDataToFirebase(usernameRef:DatabaseReference,plistName:String)
{

   }
```

建立plist檔案名常數

```
let fileName = "sortednames.plist";
```

當檢查沒資料時，呼叫自訂function uploadDataToFirebase(usernameRef:,plistName:)，並且將節點的參考和要匯入的plist字串名稱帶入至參數內。

```
self.uploadDataToFirebase(usernameRef: usernameRef, plistName: self.fileName);
```

將sortednames.plist，從資料夾內加入到專案內。	

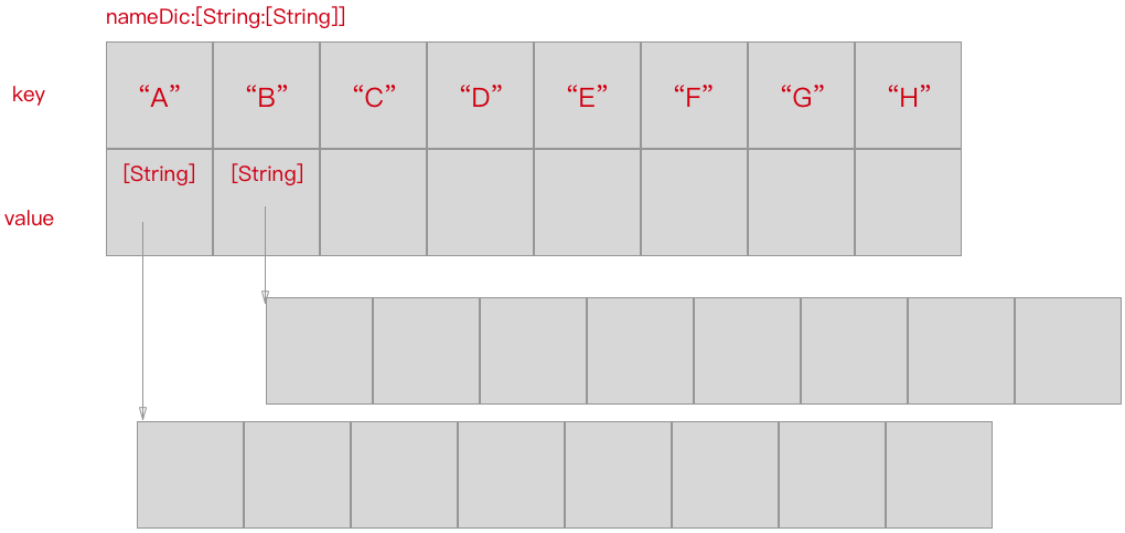
將sortednames.plist轉成Dictionary。

```
func uploadDataToFirebase(usernameRef:DatabaseReference,plistName:String){
    //切割字串為陣列
    let plistNames = plistName.split(separator: ".");

    //將陣列列印出來，確認檔案有如實傳入
    print(plistNames);

    //使用 Bundle的 func url(forResource: String?, withExtension: String?, subdirectory: String?)，取出plist的url路徑。
    //使用guard else檢查是否有錯誤

    guard let plistURL = Bundle.main.url(forResource: String(plistNames[0]), withExtension: String(plistNames[1]))else{
        print("路徑URL有問題");
        return;
    }

    //使用NSDictionary先建立NSDictionary實體，透過 as! 轉型為 Dictionary
    let namesDic = NSDictionary(contentsOf: plistURL) as! [String:[String]];          Initialization of immutable value 'namesDic' was
}
```

列印出namesDic，確定資料正確。

```
print(namesDic);
```

記住並了解目前namesDic的資料結構,nameDic，key為字串，value為字串陣列。



利用for in 迴圈，執行一次nameDic全部資料的導覽。

```
//導覽nameDic
for (key,values) in namesDic {
    //取出的values為陣列，再一次使用For in導覽value陣列。
    for value in values{
        usernameRef.child(key).childByAutoId().setValue(value);
    }

}
```

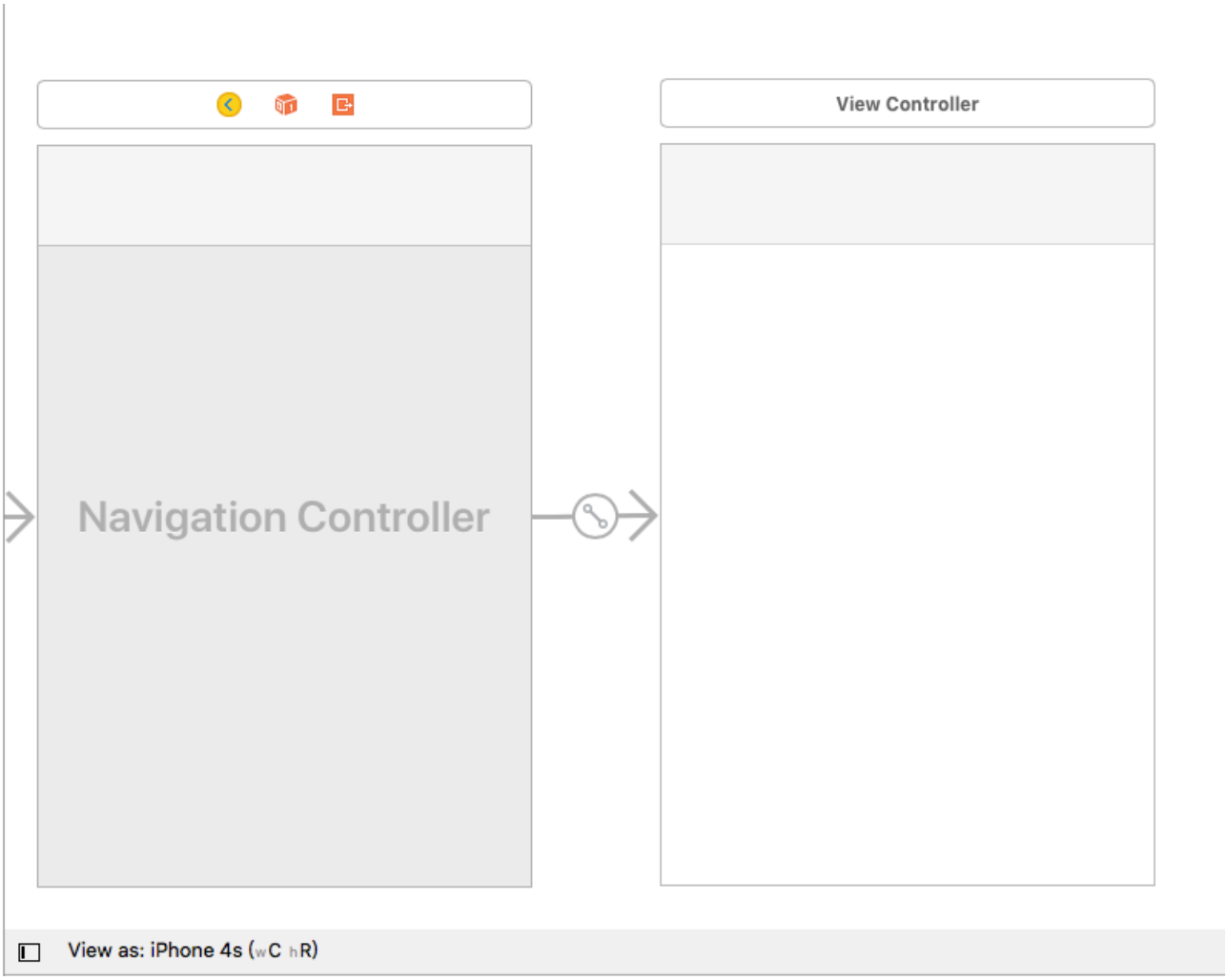
安排在Firebase RealTime DataBase的資料結構，iphone2/userName/key/autoId

只要一行語法就可以新增一筆資料。透過迴圈的執行，將所有資料加入至Firebase。進入Firebase realtime Datatime檢查是否有正確加入資料。

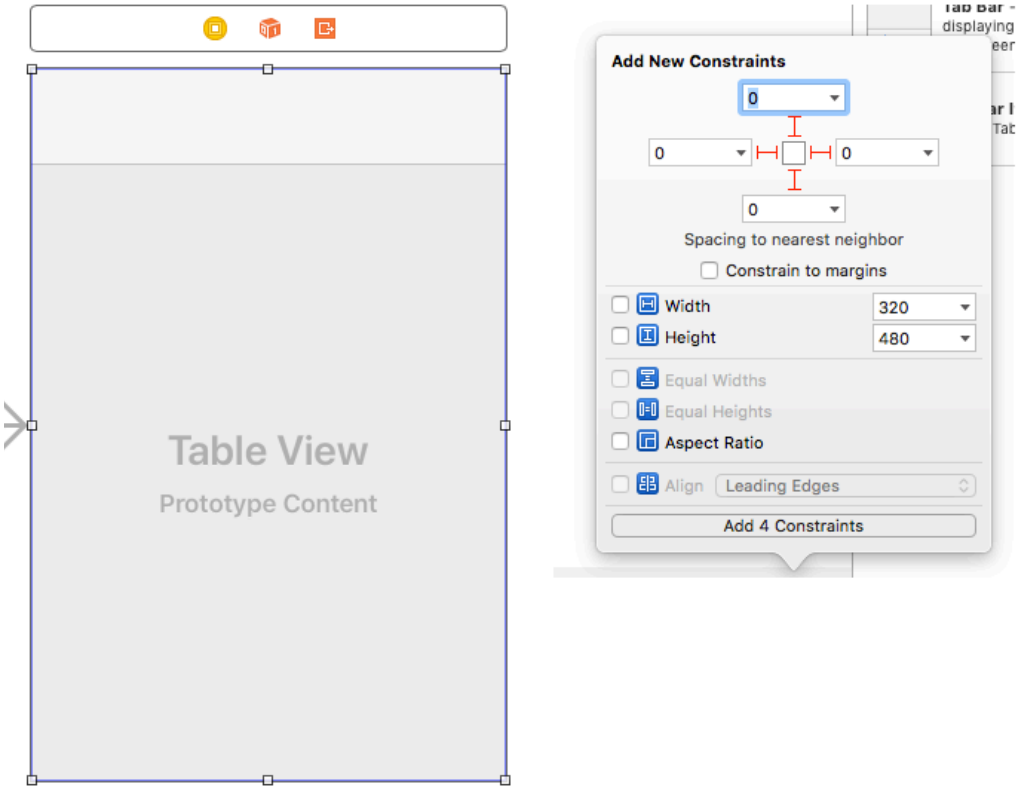
```
usernameRef.child(key).childByAutoId().setValue(value);
```

規畫顯示介面

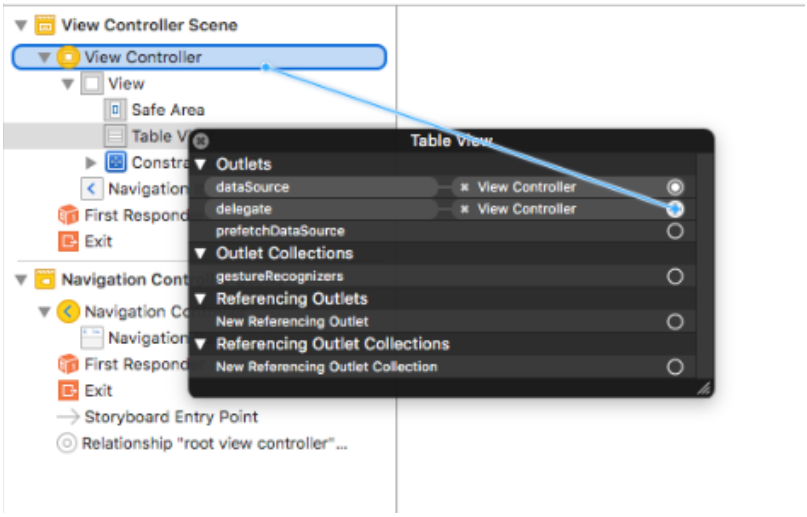
進入storyboard，將ViewController加入UINavigationController。並將顯示介面設為iphone4S。



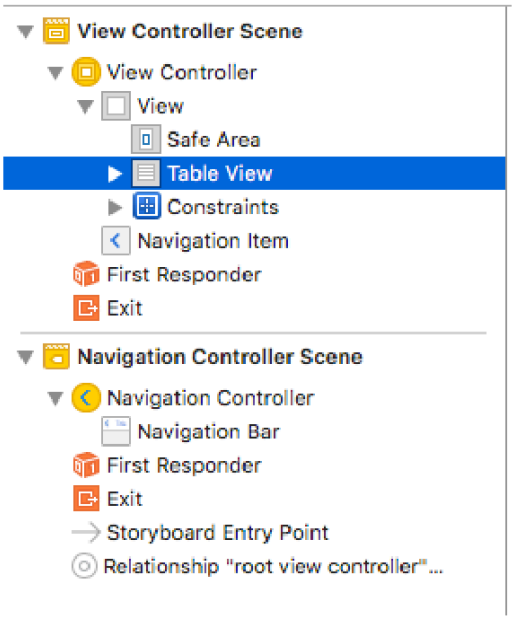
從Library內，加入UITableView，並且設定Constraints為上:0， 下:0，左:0，右:0

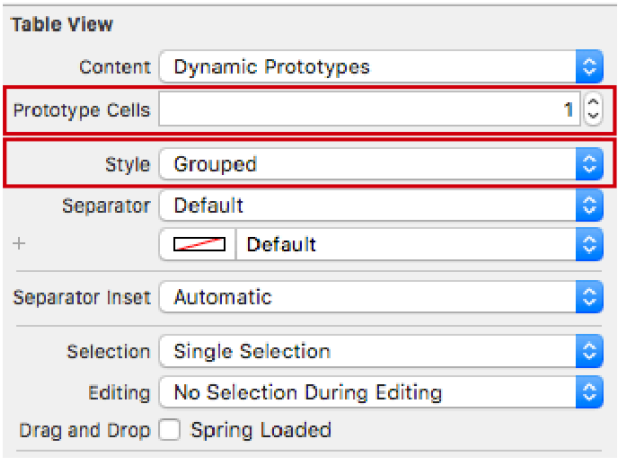


點選tableView按右鍵，將tableViewdataSourced和tableViewDelegate拉指向ViewController。

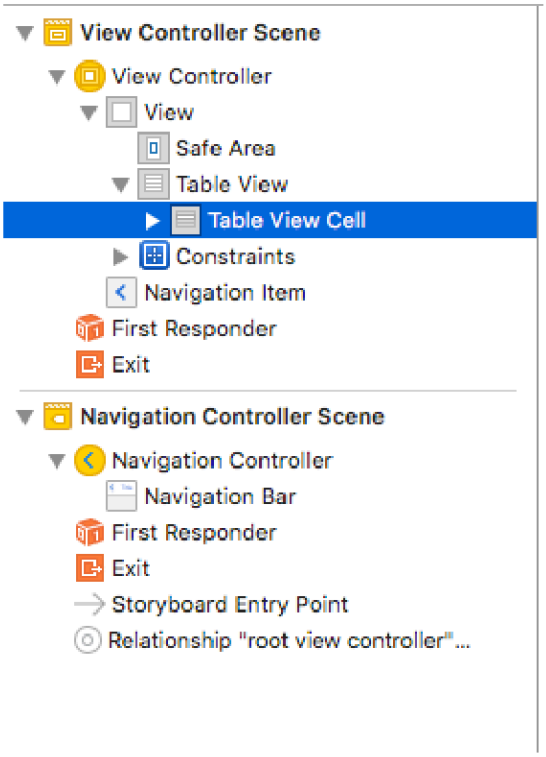


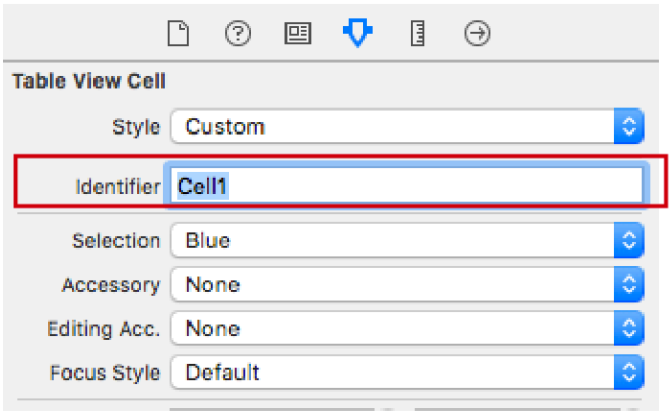
點選tableView，設定屬性prototype cells為1，style設為:Group



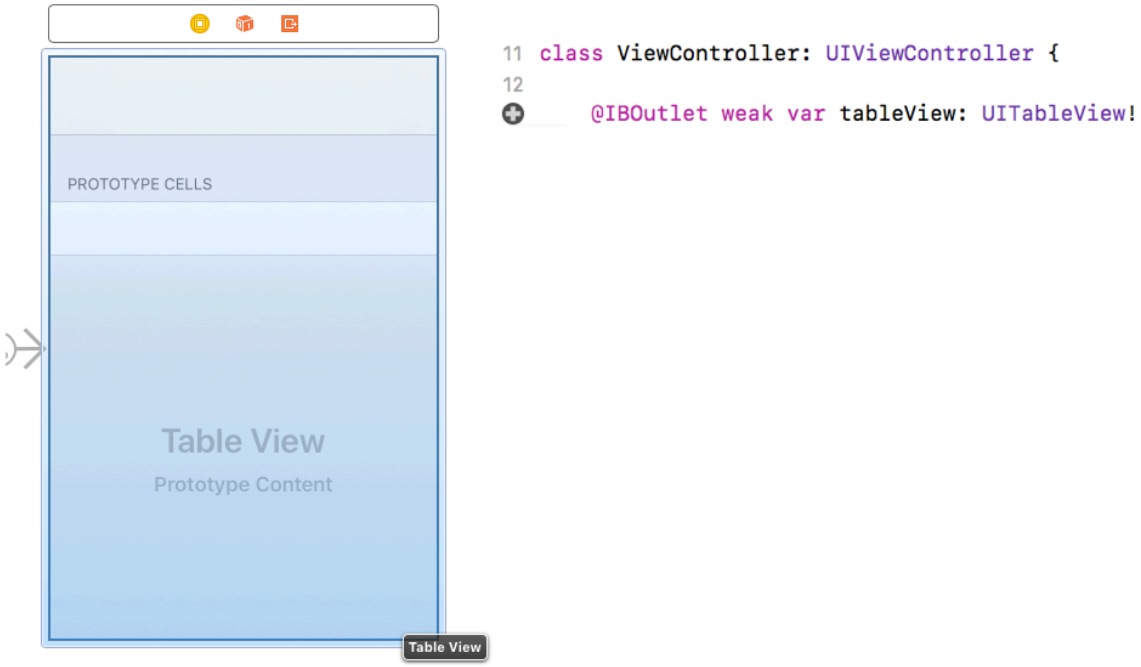


點選TableViewCell，並將identified設為Cell1。



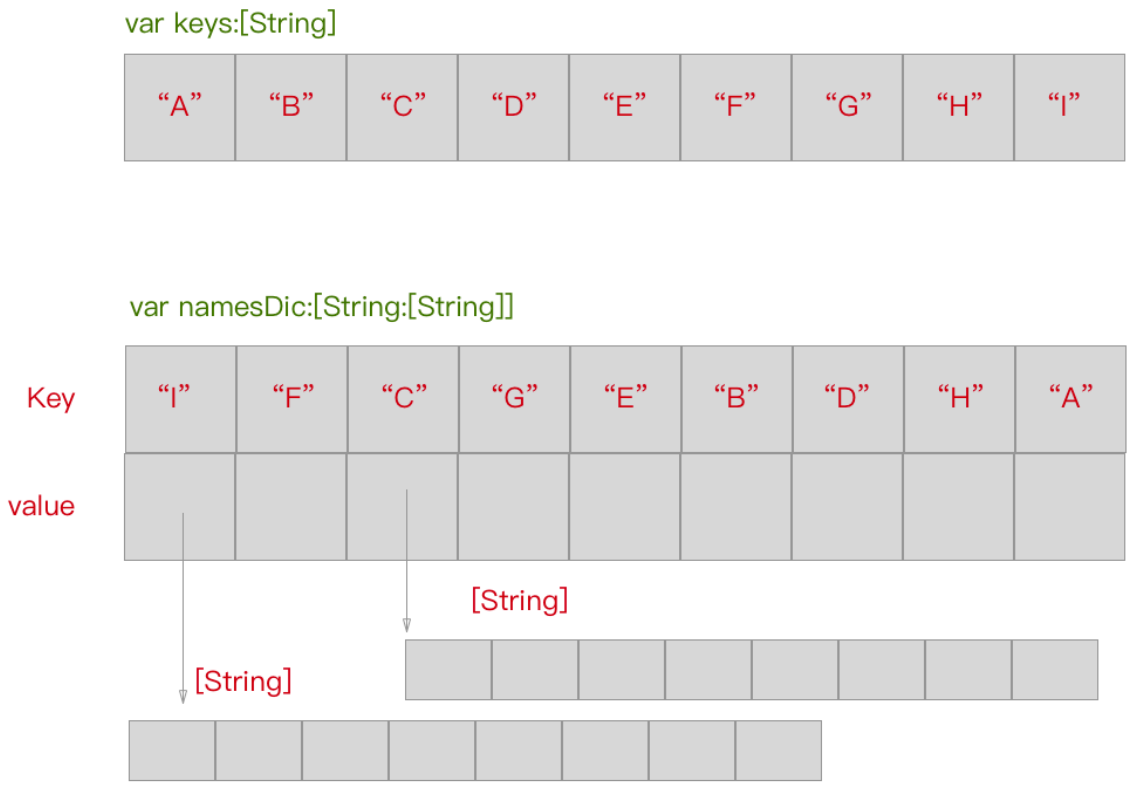


將tableView建立IBOutlet tableView參考。



規畫本地端的資料架構

使用這資料架構可以建立出多Section的TableView，每個Section 可以擁有各自不同數量的row。規畫的架構如下:
建立keys陣列，負責section的數量和title。
建立namesDic，負責每個Section內rows的資料。



在ViewController建立規畫的本地端資料來源的陣列和Dictionary

```
//建立本地端tableView的資料來源，負責Section的數量和title
//一開始建立空的陣列
var keys:[String] = [];

//建立本地端tableView的資料來源，負責ros的數量和內容
//一開始建立空的Dictionary
var namesDic:[String:[String]] = [:];
```

在ViewController import module Firebase 和在 viewDidLoad時間點，註冊Firebase realtime 內 iphone2/userName節點的監聽，並且取得資料。

```
import Firebase
```

```
override func viewDidLoad() {
        super.viewDidLoad();
        //取得節點
        let userNameRef = Database.database().reference(withPath: "iphone2/
userName");
        //監聽並取得資料
        userNameRef.observe(.value) { (userNameSnapshot:DataSnapshot) in

        }
    }
```

在closure內，取得資料，並且導覽所有資料，將資料分配給keys和namesDic。完成後，代表提供tableView的資料已經準備完成。

```
//監聽並取得資料
userNameRef.observe(.value) { (userNameSnapshot:DataSnapshot) in

    //取得firebase userName節點的所有資料
    if let userNamesValue = userNameSnapshot.value as? [String:[String:String]]{
        //取出userNamesValue內，所有的keys和排序內容
        self.keys = Array( userNamesValue.keys).sorted();

        //使用for..in導覽所有的keys
        for key in self.keys{

            //建立names字串陣列空陣列，將收集每一個內的所有userName
            var names:[String] = [];

            //取得Firebase內，單一key內所有的values([String:String])
            let keyGroup = userNamesValue[key]!;

            //取出所有單一key內的userName，並且將一個一個的userName加入至names陣列。
            for (_,userName) in keyGroup{
                let name = userName;
                names += [name];
            }

            //將key和names加入至namesDic內
            self.namesDic[key] = names;
        }

    }
}
```

列印keys內的資料和nameDic的資料。確保程式邏輯和資料是正確的。(先將storyboard內，tableView至UITableViewDataSource的連結移除，輸出後再重新連結DataSource)

```
print(self.keys);
print(self.namesDic);
```

```
["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T",
"U", "V", "W", "X", "Y", "Z"]
["H": ["Hassan", "Hezekiah", "Hugo", "Hunter", "Holly", "Holden", "Hadley", "Hayley", "Haley", "Hailie",
"Henry", "Harley", "Hannah", "Hallie", "Haiden", "Heaven", "Hazel", "Hanna", "Heather", "Hope",
"Humberto", "Hailey", "Haven", "Hamza", "Hayden", "Hudson", "Helena", "Haden", "Harley", "Hugh",
"Harper", "Hana", "Hailee", "Harry", "Hayden", "Halle", "Hayleigh", "Harper", "Harold", "Houston",
"Haylie", "Haylee", "Heath", "Helen", "Howard", "Hector", "Harmony", "Haleigh", "Heidi", "Hillary",
"Harrison"], "X": ["Ximena", "Xander", "Xiomara", "Xzavier", "Xavier"], "D": ["Dimitri", "Desiree",
```

顯示有Section和Row的TableView

實作UITableViewDataSource。產生有Section的TableView。

```
//建立extension，並採納protocol UITableViewDataSource
extension ViewController:UITableViewDataSource{

    //回傳tableView有多少個Section
    func numberOfSections(in tableView: UITableView) -> Int{
        return self.keys.count;
    }

    //回傳每個Section有多少個Rows
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
        //取出每個Section的namse, 並傳出names的數量
        let names = namesDic[keys[section]]!;
        return names.count;
    }

    //回傳每個Row需要的Cell
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
        //取出section的names
        let names = namesDic[keys[indexPath.section]]!;
        //取出應section內對應row的name
        let name = names[indexPath.row];
        //建立Cell
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell1", for: indexPath);
        //將name顯示在cell上
        cell.textLabel?.text = name;
        return cell;
    }

}
```

資料下載完成後，必需要求tableView重新載入資料。

```
self.tableView.reloadData();
```

實作func tableView(_: UITableView, titleForHeaderInSection: Int) -> String?，建立每個Section的title。

```
//建立每個Section的title
    func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int)
-> String?{
        return keys[section];
    }
```

實作func sectionIndexTitles(for: UITableView) -> [String]?，建立索引的indexTitle。

```
//建立索引的indexTitle
    func sectionIndexTitles(for tableView: UITableView) -> [String]?{
        return keys;
    }
```

由於未來會有刪除的動作。nameDic的value不可以只有單純的儲存姓名的字串陣列，所以建立一個UserName的結構，裏面有2個store property，1個儲存AutoKeyId另一個儲存name。建立UserName.swift

```
struct UserName{
    var AutoKeyId:String;
    var name:String;
}
```

修改ViewController內的資料結構。改為使用UserName結構。修改如下

```
//一開始建立空的Dictionary
var namesDic:[String:[UserName]] = [:];

//建立names字串陣列空陣列，將收集每一個內的所有userName
var names:[UserName] = [];

//取出所有單一key內的userName，並且將一個一個的userName加入至names陣列。
                for (key,userName) in keyGroup{
                    let name = UserName(AutoKeyId: key, name: userName);
                    names += [name];
                }

//取出應section內對應row的name
let userName = names[indexPath.row];

//將name顯示在cell上
cell.textLabel?.text = userName.name;
```