

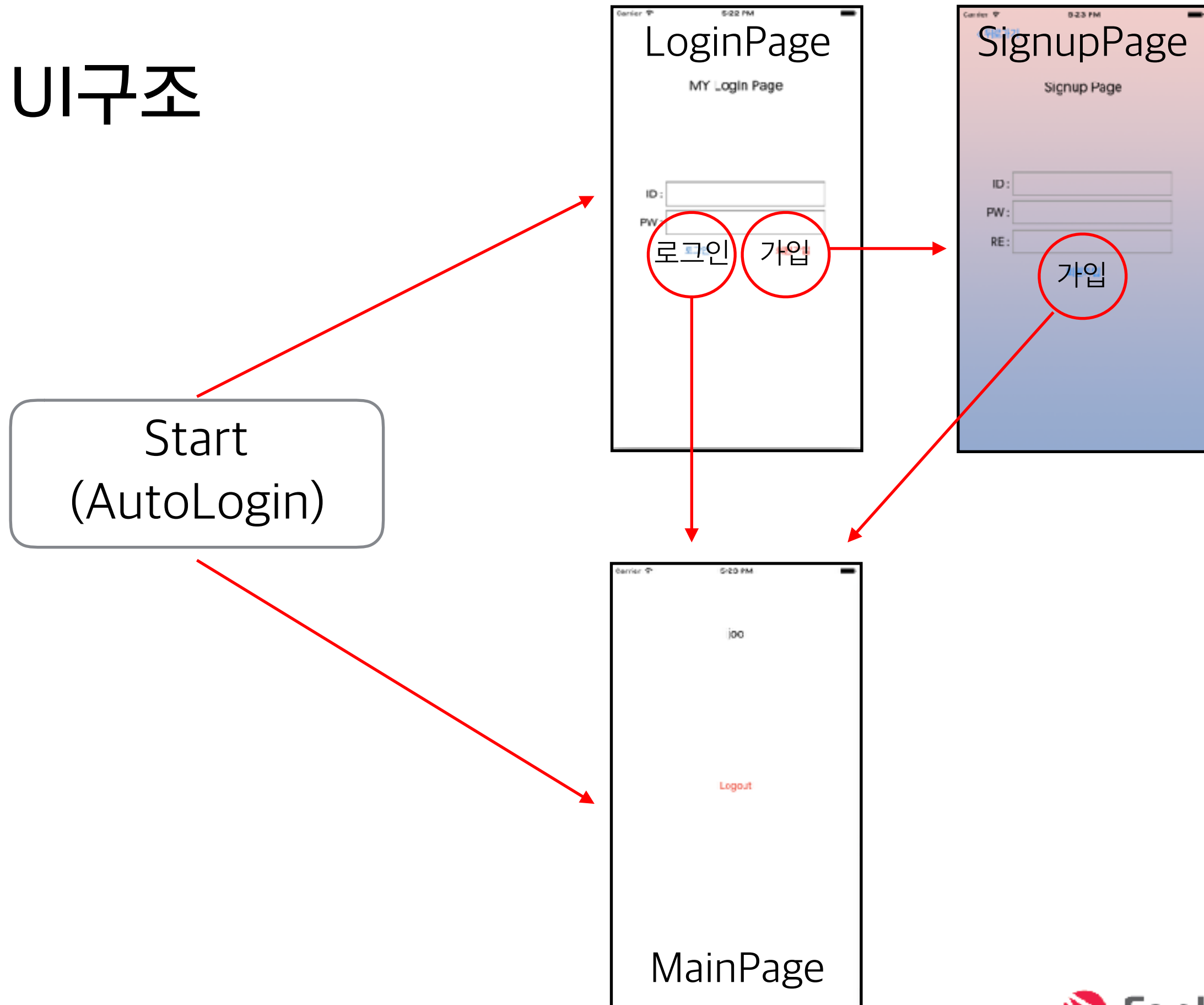
---

# 미니 프로젝트 - LoginPage

---


강사 주영민

# UI구조



# UI구조

## LoginPage



Carrier 10:11 PM

MY Login Page

ID :

PW :

[로그인](#) [회원가입](#)

QWERTYUIOP

ASDFGHJKL

ZXCVBNM

123 globe microphone space return

---

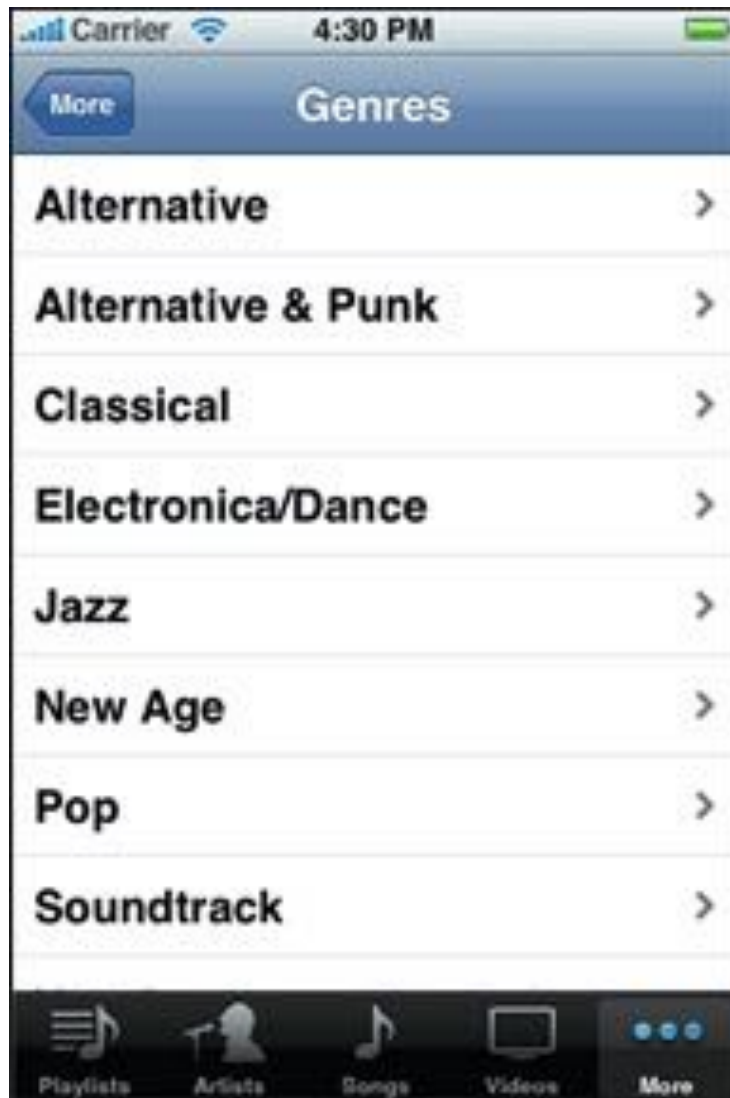
# UITableView

---

강사 주영민

# UITableView

- 리스트의 형태로 정보를 보여주는 View



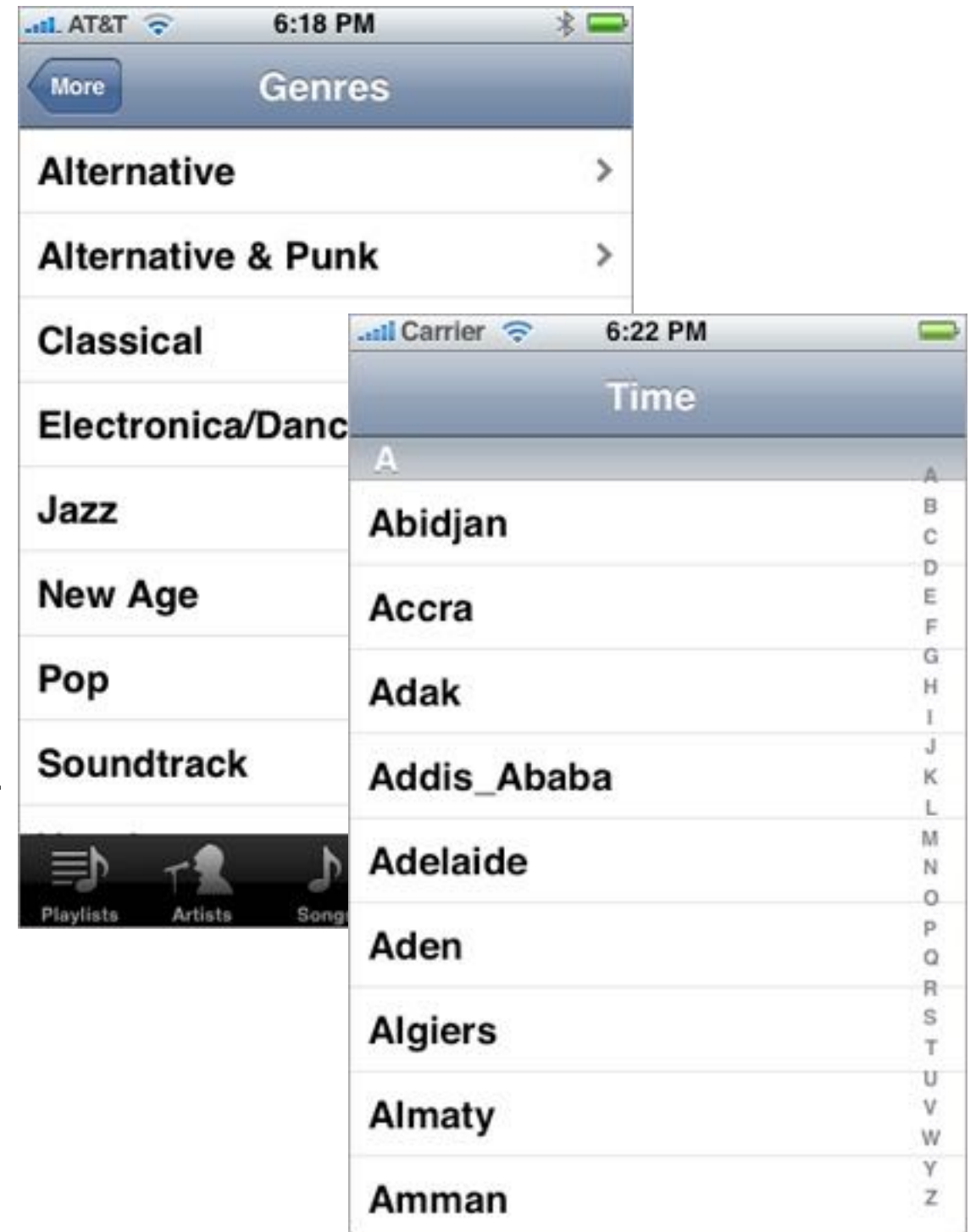
# Style

---

- plain
- Grouped

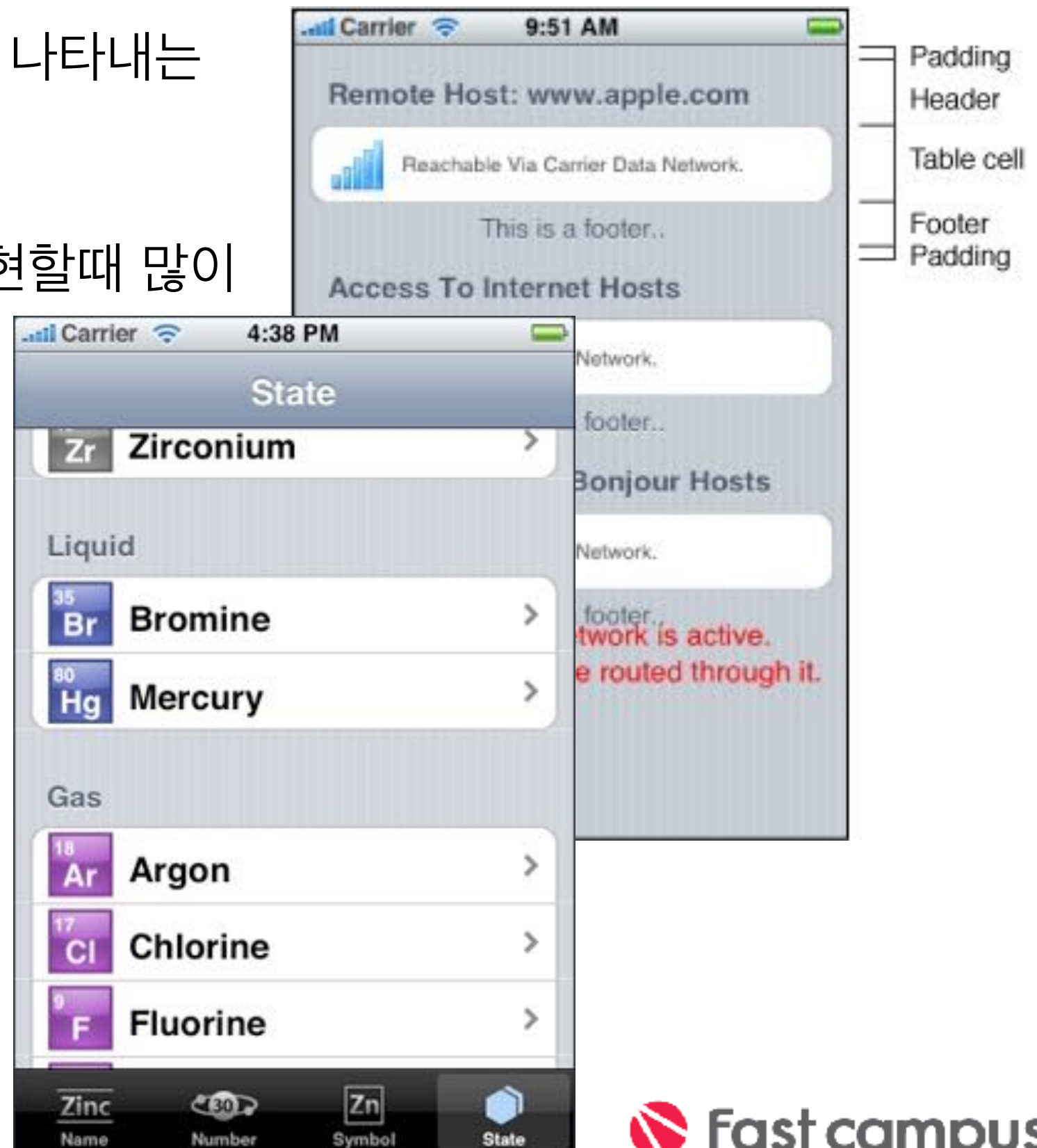
# Plain Table Views

- 기본적인 TableView
- 여러개의 Section을 가질수 있다.
- 한 Section에는 여러개의 Row를 포함하고 있다.
- 각각의 Section에는 Section을 표시하는 header, Footer title을 사용할수 있다.
- Section을 빠르게 검색할수 있는 네비게이터 바를 index list 라고 부른다.



# Grouped Table Views

- 각 Section을 Group의 형태로 나타내는 테이블 뷰
- 데이터의 디테일한 정보를 표현할때 많이 사용된다.





# Protocol

---

- DataSource
- Delegte

# DataSource

---

- 프로토콜을 사용하여 테이블뷰에서 보여줄 데이터를 관리할 대리인의 역할을 정의해 둔 것
- 역할

@requires

- 테이블 뷰의 각 섹션별 열의 개수를 설정
- Row별 Cell객체

@optional

- 테이블 뷰 섹션의 개수를 설정

등등

# DataSource

---

## @requires

- (NSInteger)tableView:(UITableView \*)tableView  
numberOfRowsInSection:(NSInteger)section;
- (UITableViewCell \*)tableView:(UITableView \*)tableView  
cellForRowAtIndexPath:(NSIndexPath \*)indexPath;

## @optional

- (NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView;  
// Default is 1
- (nullable NSString \*)tableView:(UITableView \*)tableView  
titleForHeaderInSection:(NSInteger)section;
- (nullable NSString \*)tableView:(UITableView \*)tableView  
titleForFooterInSection:(NSInteger)section;

# Delegate

---

- 프로토콜을 사용하여 테이블뷰의 대리자로써의 수행할 수 있는 역할들을 정의해 둔 것
- 역할
  1. 헤더 또는 풋터의 높이를 설정 : Variable height support
  2. 헤더 또는 풋터 뷰를 제공 : Section Informations.
  3. 셀을 선택하였을 때 수행할 동작 관리 : Selection
  4. 셀의 삭제 될 때의 동작 등의 관리 : Editing
  5. 기타 등등

# Delegate

---

- 프로토콜에서 다음 항목들이 어떤 메소드 명으로 정의 되어 있는지 찾아보아요
  1. Cell의 높이를 조정하는 델리게이트 메소드
  2. Header View를 Custom View로 대체하는 메소드
  3. Cell을 선택 했을때 불러오는 델리게이트 메소드

# NSIndexPath Class

---

- 셀의 위치를 나타낼 Data Type으로 Secion정보와 해당 섹션에서의 열(Row) 정보를 가지고 있다.

```
@interface NSIndexPath (UITableView)
```

```
+ (instancetype)indexPathForRow:(NSInteger)row inSection:  
(NSInteger)section;
```

```
@property (nonatomic, readonly) NSInteger section;
```

```
@property (nonatomic, readonly) NSInteger row;
```

```
@end
```

# 테이블 뷰 만들기

---

1. 객체 생성
2. Section 갯수 설정 (option)
3. 각 Section당 Row설정
4. 각 indexPath당 Cell(view)설정

# 예제 - 객체 생성

---

```
UITableView *tableView = [[UITableView alloc] initWithFrame:CGRectMake(0, 0,  
self.view.frame.size.width, self.view.frame.size.height)  
style:UITableViewStylePlain];  
tableView.dataSource = self;  
tableView.delegate = self;  
[self.view addSubview:tableView]
```



# 예제 - Row Count Set

---

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSectionSection:
(NSInteger)section
{
    return 10;
}
```

# 예제 - Cell create And Reusable

---

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"Cell"];
    }

    cell.textLabel.text = [NSString stringWithFormat:@"%zd", indexPath.row];

    return cell;
}
```

# 다양한 테이블 뷰를 만들어 봅시다.

---

- Plain Style TableView vs Grouped Style TableView
- changed section number & row number

# 동물 이름 리스트 만들기

---

- 동물 데이터 만들기(List)
- tableView 만들기
- 텍스트 보여지기

# UITableViewCell

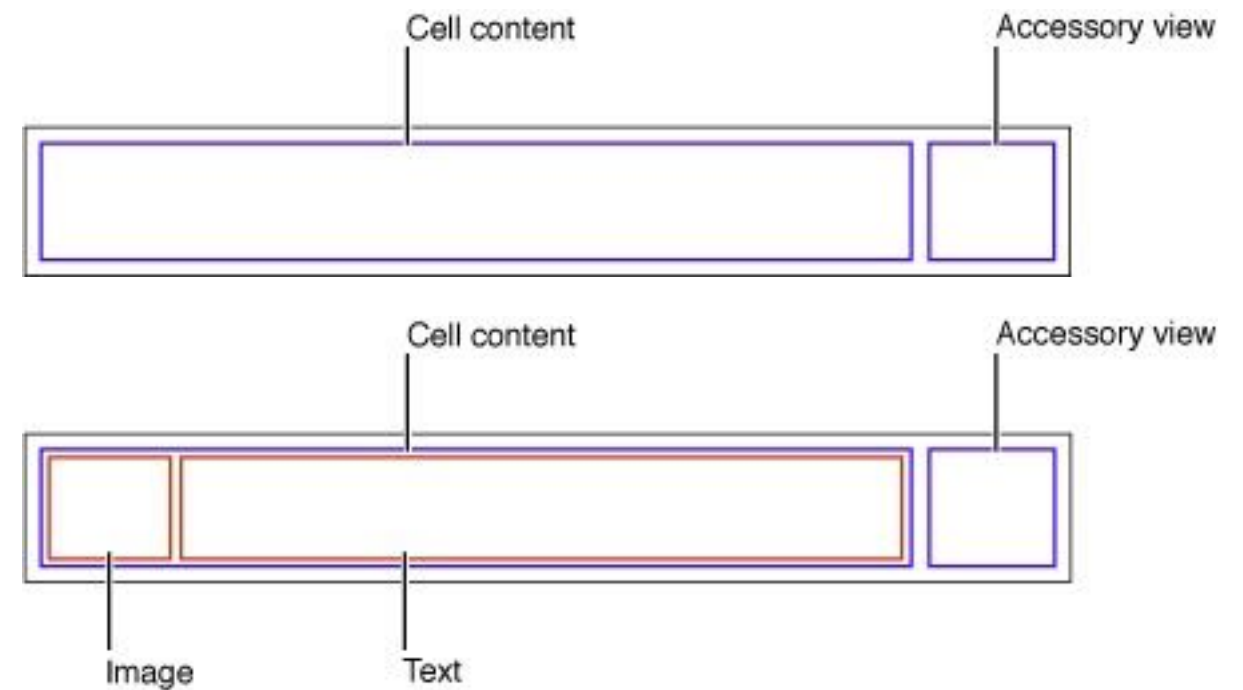
---

- display text, images, or other kinds of content.
- background views for both normal and selected states.
- Cells can also have accessory views,











# UITableViewCell











---

- contentView
  - .textLabel
  - detailTextLabel
  - imageView
- accessoryView



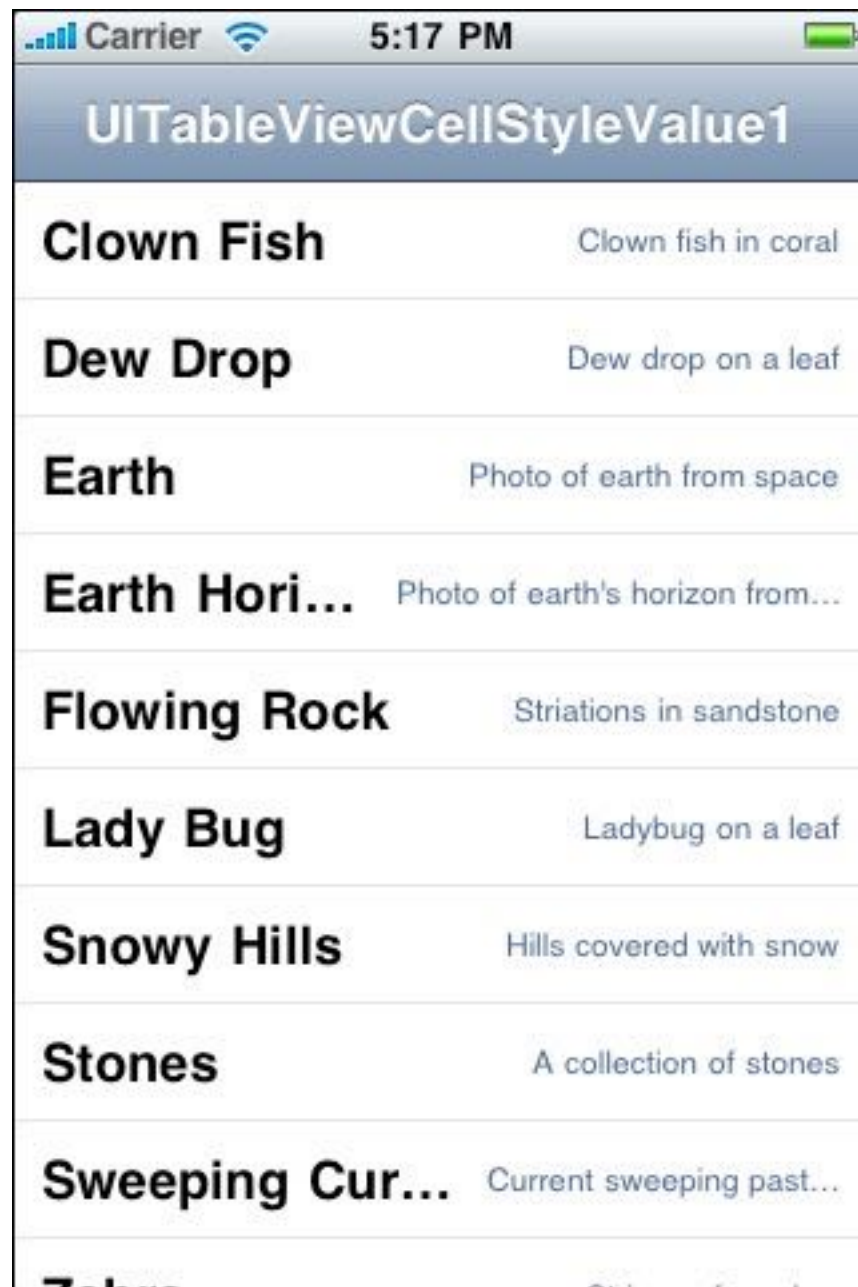
# Table View Cells Style

UITableViewCellStyleDefault	
	<b>Clown Fish</b>
	<b>Dew Drop</b>
	<b>Earth</b>
	<b>Earth Horizon</b>
	<b>Flowing Rock</b>
	<b>Lady Bug</b>
	<b>Snowy Hills</b>
	<b>Stones</b>
	<b>Sweeping Current</b>
	<b>Zebra</b>

UITableViewCellStyleSubtitle	
	<b>Clown Fish</b> Clown fish in coral
	<b>Dew Drop</b> Dew drop on a leaf
	<b>Earth</b> Photo of earth from space
	<b>Earth Horizon</b> Photo of earth's horizon from space
	<b>Flowing Rock</b> Striations in sandstone
	<b>Lady Bug</b> Ladybug on a leaf
	<b>Snowy Hills</b> Hills covered with snow
	<b>Stones</b> A collection of stones
	<b>Sweeping Current</b> Current sweeping past rocks
	<b>Zebra</b>

# Table View Cells Style

---





# TableView Cell 만들기

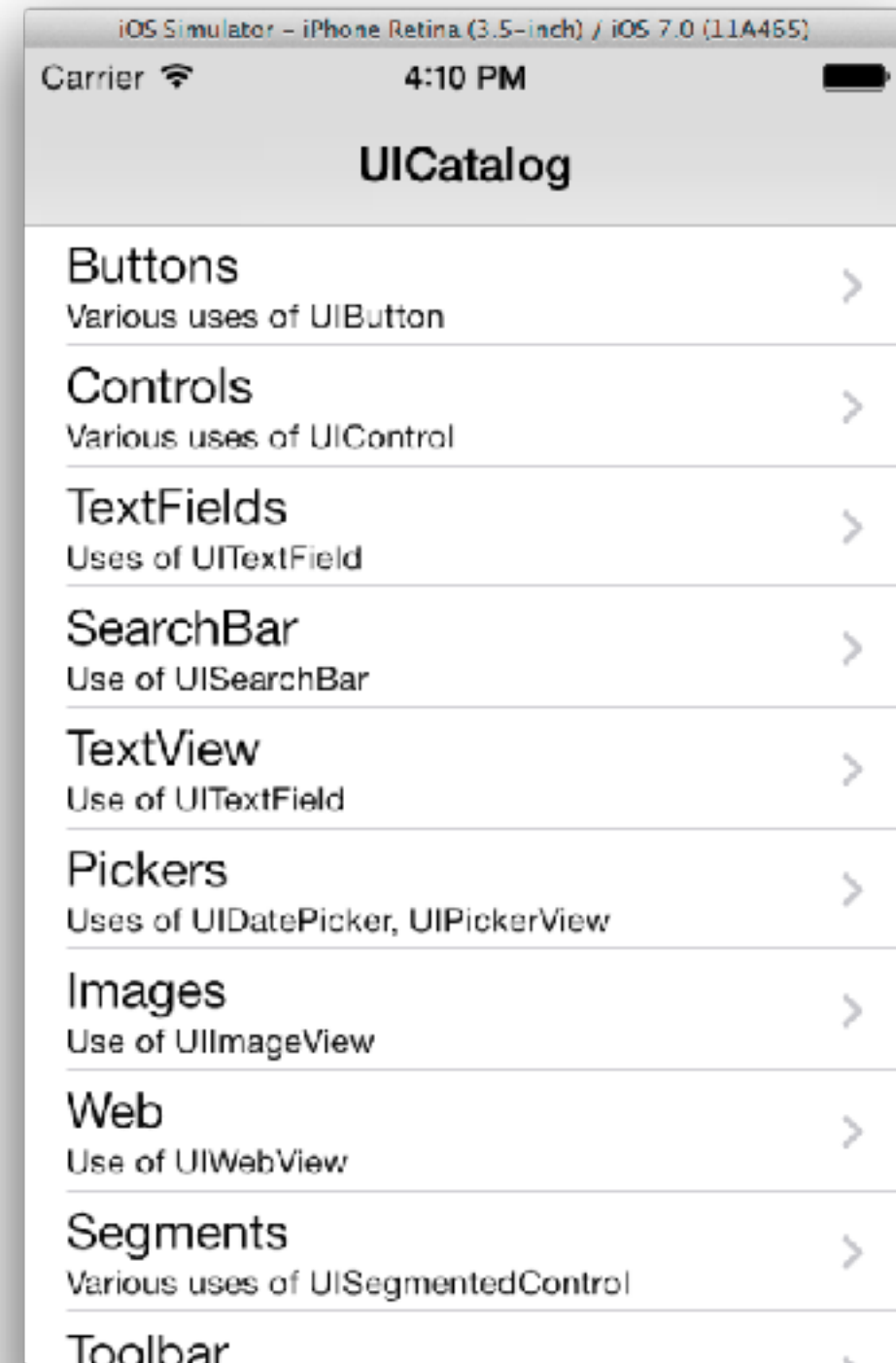
---

- 다양한 타입의 Cell을 만들어 봅시다.

# 재사용(reuse)

---

- 테이블 뷰는 한 줄 한 줄이 굉장히 비슷한 모습을 가지고 있다.
- 수백개의 테이블 리스트가 있다고 가정할 때...



# 재사용(reuse)

---

- 공통적인 레이아웃을 사용하여 여러번 화면을 보여줄 필요가 있는 경우 뷰를 재사용

# 재사용(reuse)

---

- 두 메소드의 차이는 뭘까요?

```
- (UITableViewCell *)dequeueReusableCellWithIdentifier:(NSString *)identifier;  
  
// Used by the delegate to acquire an already allocated cell, in lieu of  
// allocating a new one.  
  
- (UITableViewCell *)dequeueReusableCellWithIdentifier:(NSString *)identifier  
                                     forIndexPath:(NSIndexPath *)indexPath  
// newer dequeue method guarantees a cell is returned and resized properly,  
// assuming identifier is registered
```

# 재사용(reuse)

---

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"];

    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:@"Cell"];
    }

    cell.textLabel.text = [NSString stringWithFormat:@"%zd", indexPath.row];

    return cell;
}
```

---

VS

---

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell"
forIndexPath:indexPath];

    cell.textLabel.text = [NSString stringWithFormat:@"%zd", indexPath.row];

    return cell;
}
```

---

# UITableView 실습

---

# TableView 가지고 놀기(실습) : Selected Cell Action




---

- Cell을 클릭했을때 어떤 델리게이트 메소드가 실행될까요?
- deselect는 어떻게 시킬까요?
- 선택된 Cell의 타이틀 정보를 가져와서 알럿으로 띄어 봅시다.

# TableView 가지고 놀기(실습) : Cell Accessory

---

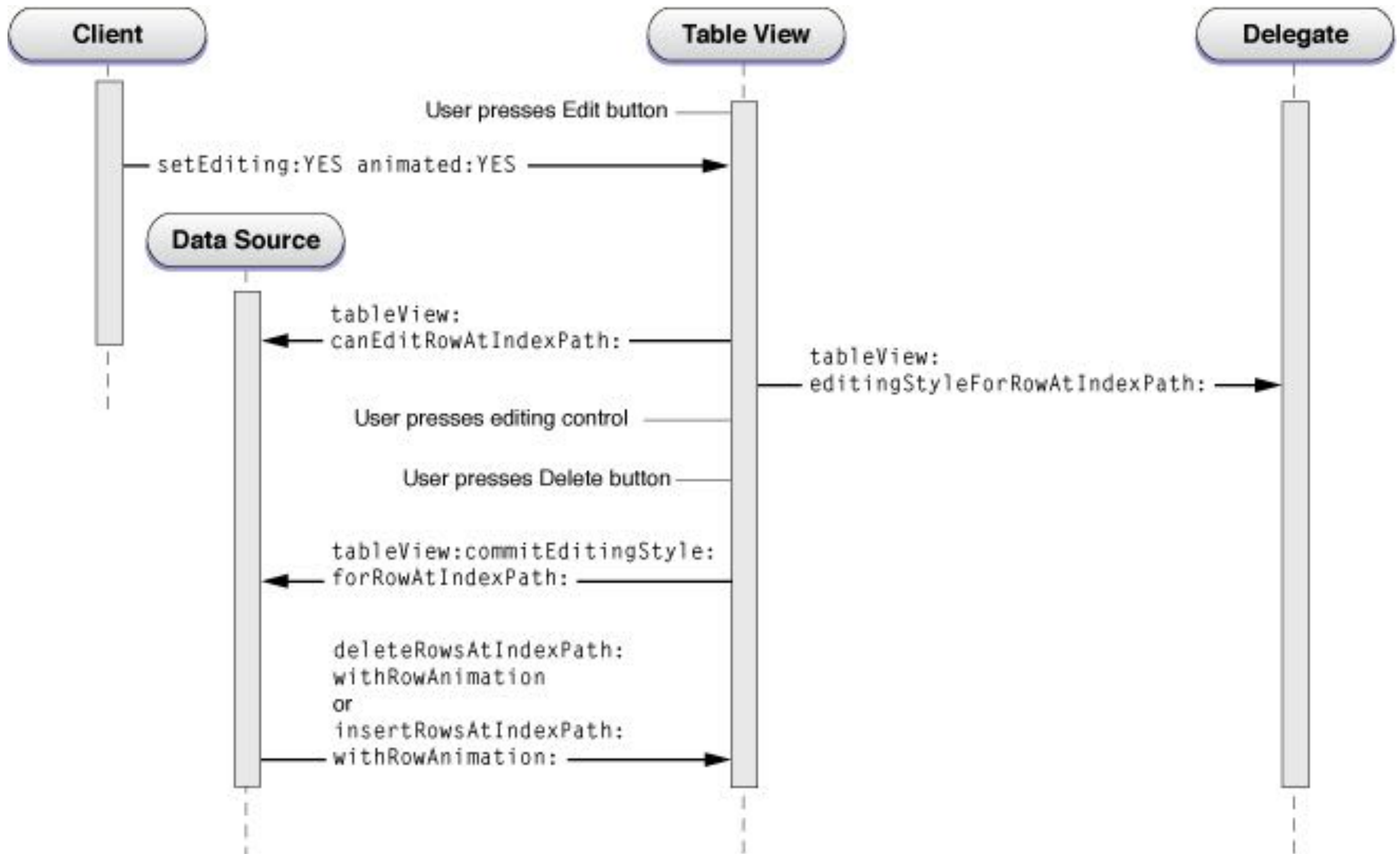
- cell.accessoryType = UITableViewCellAccessoryCheckmark;

Standard accessory views	Description
	<b>Disclosure indicator</b> — <code>UITableViewCellAccessoryDisclosureIndicator</code> . You use the disclosure indicator when selecting a cell results in the display of another table view reflecting the next level in the data model hierarchy.
	<b>Detail disclosure button</b> — <code>UITableViewCellAccessoryDetailDisclosureButton</code> . You use the detail disclosure button when selecting a cell results in a detail view of that item (which may or may not be a table view).
	<b>Checkmark</b> — <code>UITableViewCellAccessoryCheckmark</code> . You use a checkmark when a touch on a row results in the selection of that item. This kind of table view is known as a selection list, and it is analogous to a pop-up list. Selection lists can limit selections to one row, or they can allow multiple rows with checkmarks.

- 다른 Accessory View는 어떻게 추가 할수 있을까요?



# TableView 가지고 놀기(실습) : Editing



# TableView 가지고 놀기(실습) : Editing

---

```
- (BOOL)tableView:(UITableView *)tableView
    canEditRowAtIndexPath:(NSIndexPath *)indexPath{
    return YES;
}

- (UITableViewCellEditingStyle)tableView:(UITableView *)tableView
    editingStyleForRowAtIndexPath:(NSIndexPath *)indexPath
{
    //edit style설정
    return UITableViewCellEditingStyleDelete;
}

- (void)tableView:(UITableView *)tableView
    commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
    forRowAtIndexPath:(NSIndexPath *)indexPath
{
    //삭제 되는 데이터 지우고
    [self.dataList removeObjectAtIndex:indexPath.row];
    //테이블에서 Cell 지우기
    [tableView deleteRowsAtIndexPaths:@[indexPath]
                    withRowAnimation:UITableViewRowAnimationFade];
}
```

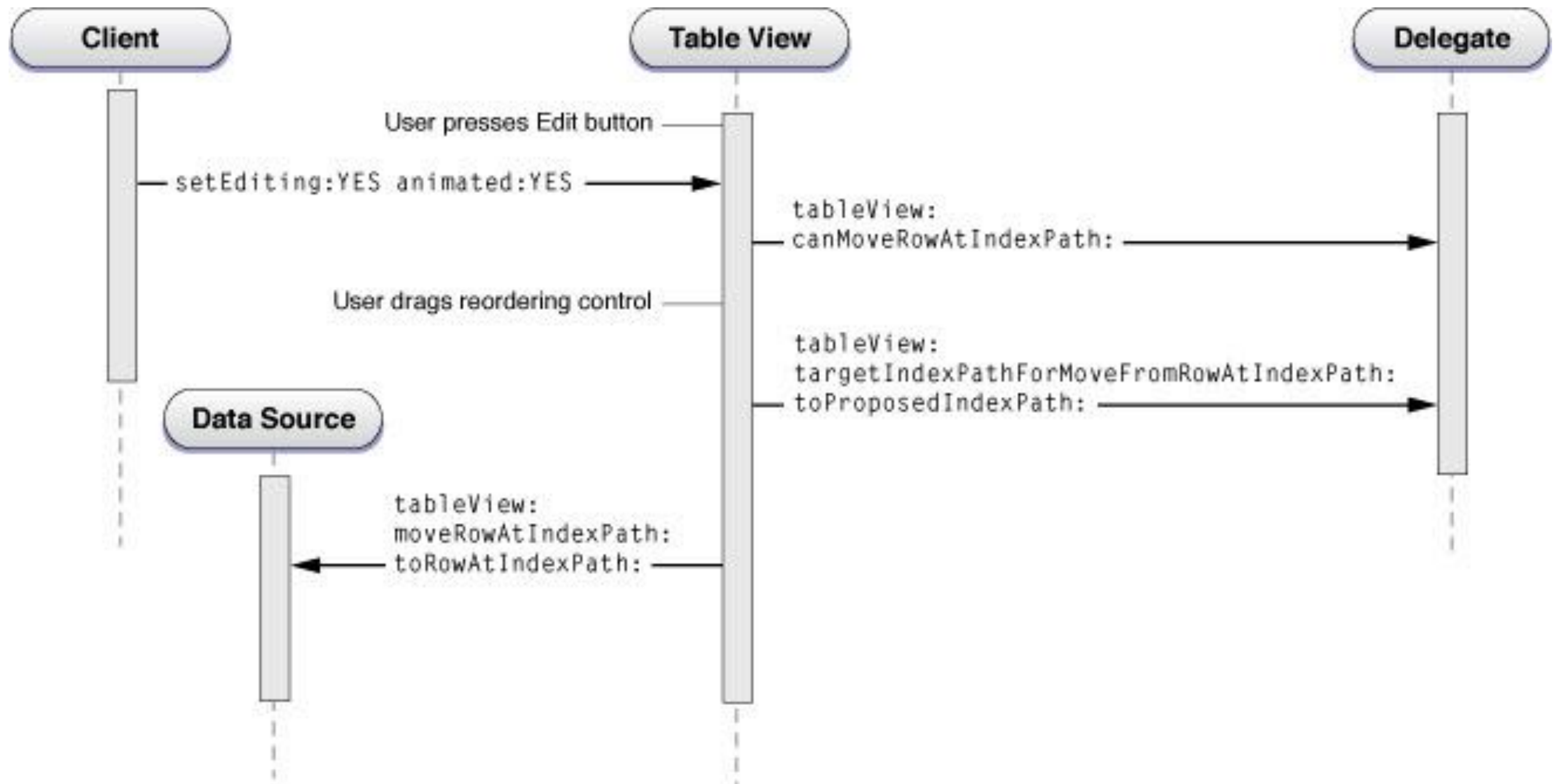
# Edit Button Action Example

---

```
- (void)addORDoneRows
{
    if(self.editing)
    {
        self.editing = NO;
        [self.tableView setEditing:NO animated:NO];
    }
    else
    {
        self.editing = YES;
        [self.tableView setEditing:YES animated:YES];
    }
}
```

# TableView 가지고 놀기(실습) : Moving

- cell moveing



# TableView 가지고 놀기(실습) : Moving

---

```
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:
(NSIndexPath *)indexPath
{
    return YES;
}

- (void)tableView:(UITableView *)tableView
moveRowAtIndexPath:(nonnull NSIndexPath *)sourceIndexPath
toIndexPath:(nonnull NSIndexPath *)destinationIndexPath
{
    //데이터 변경
    NSString *stringToMove = [self.carName
objectAtIndex:sourceIndexPath.row];
    [self.carName removeObjectAtIndex:sourceIndexPath.row];
    [self.carName insertObject:stringToMove
atIndex:destinationIndexPath.row];
}
```

# TableView 가지고 놀기(실습) : Custom Cell 만들기

---

BackGround

Title

Cover

