บทที่ 5: ลูกเล่น & แม่ไม้ ในการสร้าง Table View

คงไม่มีใครปฏิเสธว่า Table View เป็นรูปแบบของการแสดงข้อมูลที่มีการใช้งานมากที่สุดในบรรดา View ทุกรูปแบบใน iOS และนำมาซึ่งความปวดหัวมากมายหลายประการของโปรแกรมเมอร์ทั้งมือใหม่และมือเก่า ถึงจะมากประสบการณ์แค่ไหน ก็ต้องทนเสียเวลากับการปรับแต่ง Table View ให้ได้ดังใจ

ปัญหาของการสร้าง Table View

ผมขอวิเคราะห์ถึงปัญหาของการสร้าง Table View ว่ามี 2 เรื่องใหญ่ๆ ดังนี้

1. การสร้าง Table View ที่เป็น Static Data/Cell เช่น การใช้ Table View แทนปุ่มสารพัดปุ่มให้กด ซึ่งเปิดโปรแกรมกี่ครั้งปุ่มเหล่านี้ก็จะคงที่ ไม่มีอะไรเปลี่ยนแปลง
2. การสร้าง Table View ที่ต้องการ Cell นอกเหนือรูปแบบมาตรฐานที่มีมาให้

ในส่วนของประเด็นแรก เนื่องจากจนถึงทุกวันนี้ เราไม่มีวิธีการสร้างเนื้อหาและรับการกด Cell บน Table View ด้วยวิธีการ “Outlet & Action” ผ่าน Interface Builder ได้ว่า Cell ไหนจะมีข้อความว่าอะไร มีองค์ประกอบอะไร และเมื่อกดแล้วจะเกิดอะไรขึ้น แต่เราต้องทำทุกอย่างผ่านโค้ดใน Data Source และ Delegate ซึ่งถูกออกแบบมาสำหรับการแสดง “รายการข้อมูลที่ซ้ำๆ กัน” ทำให้การปรับโค้ดดังกล่าวมาเพื่อสร้าง Static Table View นั้นค่อนข้างยากลำบากและเสียเวลาพอสมควร ระดับที่เรียกได้ว่า “ขี่ช้างจับตั๊กแตน” กันเลยทีเดียว

ส่วนประเด็นที่สองนั้น แม้ว่า TableView Cell ที่มีมาให้ 4 แบบ (Basic, Subtitle, Left Value, Right Value) นั้นจะเพียงพอสำหรับงานทั่วไป ที่เราต้องการแสดงรายการของอะไรบางอย่าง แต่ว่าถ้ามันไม่เพียงพอขึ้นมาล่ะ จะต้องทำอย่างไร?

อันที่จริงแล้ว การสร้าง TableView Cell ในรูปแบบที่ต้องการด้วยตัวเองนั้น เป็นสิ่งที่คู่กับการพัฒนาโปรแกรมบน iOS มาตั้งแต่แรก (ตั้งแต่สมัยเป็น iPhone OS แรกๆ) โดยสมัยนั้น TableView Cell มาตรฐานไม่ได้รองรับการใส่รูปไว้ด้านซ้ายมือของ Cell เหมือนที่เราสามารถทำได้ง่ายๆ อย่างในทุกวันนี้ ผู้ที่ต้องการจะต้องทำเองด้วยการ subclass UITableViewCell และเพิ่มส่วนใส่ภาพด้วยตัวเอง และเมื่อมีคนทำกันมากๆ เข้า (ดีบ้างห่วยบ้าง) ทาง Apple ก็เลยจัดให้กลายมาเป็นฟีเจอร์มาตรฐานของ TableView Cell ในปัจจุบันนี้อย่างที่เห็น

วันนี้ที่รอคอยของนักพัฒนาโปรแกรมบน iOS ก็มาถึง (อีกครั้ง) เพราะ Storyboard ได้ “ยกระดับ” การสร้าง Table View และแก้ปัญหาทั้งสองอย่างนี้ จากสิ่งที่เคยยาก กลายเป็นสิ่งที่ทำได้ง่ายมากขึ้นเยอะ

App: MovieList

ลักษณะ:	1. เป็น iOS App ที่ทำงานในแบบ Navigation-based 2. แสดงรายชื่อภาพยนตร์ 3. เมื่อกดเข้าไปดูภาพยนตร์แต่ละเรื่อง จะพบรายละเอียดต่างๆ เช่น ปีที่เริ่มฉาย คะแนนที่ได้รับ ความยาวของภาพยนตร์ เป็นต้น

เป้าหมายการเรียนรู้:	1. การสร้าง Static TableView Cell โดยใช้ Storyboard 2. การสร้าง Navigation-based Application จาก Single-View Application Template

ซึ่งเมื่อทำเสร็จแล้วจะหน้าตาเป็นแบบนี้

พร้อมแล้วก็เริ่มทำการสร้างโปรเจคใหม่ แต่ครั้งนี้เราจะเริ่มด้วยการเลือก Single-View Application จากนั้นทำการตั้งชื่อว่า MovieList ตั้ง Class Prefix ว่า IDB เลือกใช้ Storyboard และ ARC

ลบสิ่งที่ไม่จำเป็นออกจากโปรเจค

สิ่งแรกที่เราจะทำคือ ลบสิ่งที่ไม่จำเป็นออกจากโปรเจค ซึ่งตอนนี้ให้เราลบ IDBViewController.h/m และลบ View Controller ตัวเดียวที่มีใน Storyboard ออกด้วย (ซึ่งอาจจะมีคนสงสัยว่าทำไมผมไม่เลือก Empty Application เลย ซึ่งเหตุผลก็คือ Empty Application นั้นจะไม่มี Storyboard ซึ่งหากเราต้องการเพิ่มและตั้งค่าเองก็ได้เช่นเดียวกัน แต่ยุ่งยากกว่าการสร้าง Single View แล้วลบออก)

เตรียม Model ของภาพยนตร์

งานนี้เราจะเริ่มจากการเตรียม Model สำหรับเก็บข้อมูลภาพยนตร์ ซึ่งเป็นวัตถุง่ายๆ ดังนี้

Movie.h

```
@interface Movie : NSObject
@property (nonatomic, copy) NSString *name;
@property (nonatomic, copy) NSString *detail;
@property (nonatomic, copy) NSString *director;
@property (nonatomic, copy) NSString *imageName;
@property (nonatomic, unsafe_unretained, readonly) UIImage *image;
```

```
@property (nonatomic, assign) int year;
@property (nonatomic, assign) int length;
@property (nonatomic, assign) float rating;

-(id)initWithName:(NSString *)name
           detail:(NSString *)detail
         director:(NSString *)director
             year:(int)year
           length:(int)length
           rating:(float)rating
        imageName:(NSString *)imageName;
@end
```

และ Implementation ดังนี้

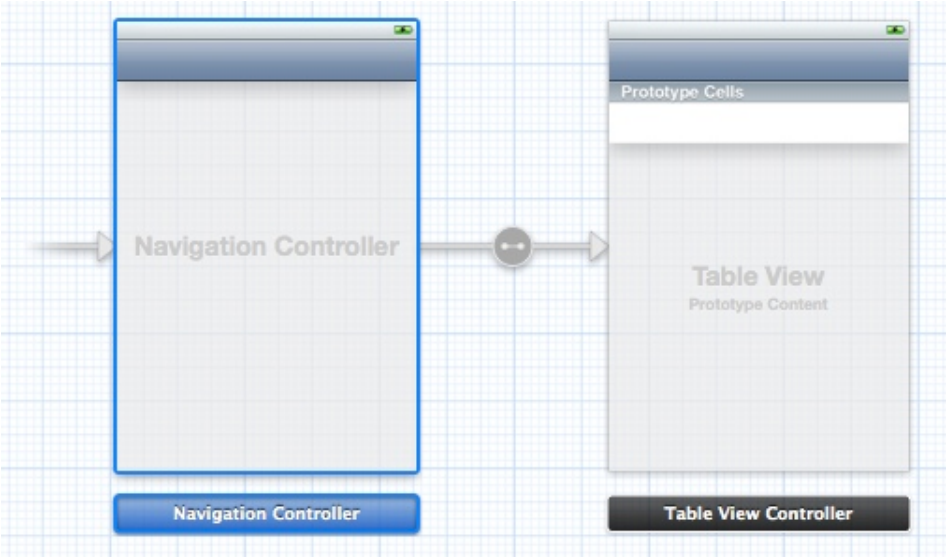
Movie.m

```
@implementation Movie
@synthesize name = _name, detail = _detail, director = _director;
@synthesize year = _year, length = _length, rating = _rating, imageName = _imageName;

-(UIImage*)image {
    return [UIImage imageNamed:self.imageName];
}

-(id)initWithName:(NSString *)name
           detail:(NSString *)detail
         director:(NSString *)director
             year:(int)year
           length:(int)length
           rating:(float)rating
        imageName:(NSString *)imageName
{
    self = [super init];
    if (self) {
        self.name = name;
        self.detail = detail;
        self.director = director;
        self.year = year;
        self.length = length;
        self.rating = rating;
        self.imageName = imageName;
    }
    return self;
}
@end
```

สร้าง Screen Flow ของโปรแกรมด้วย Storyboard

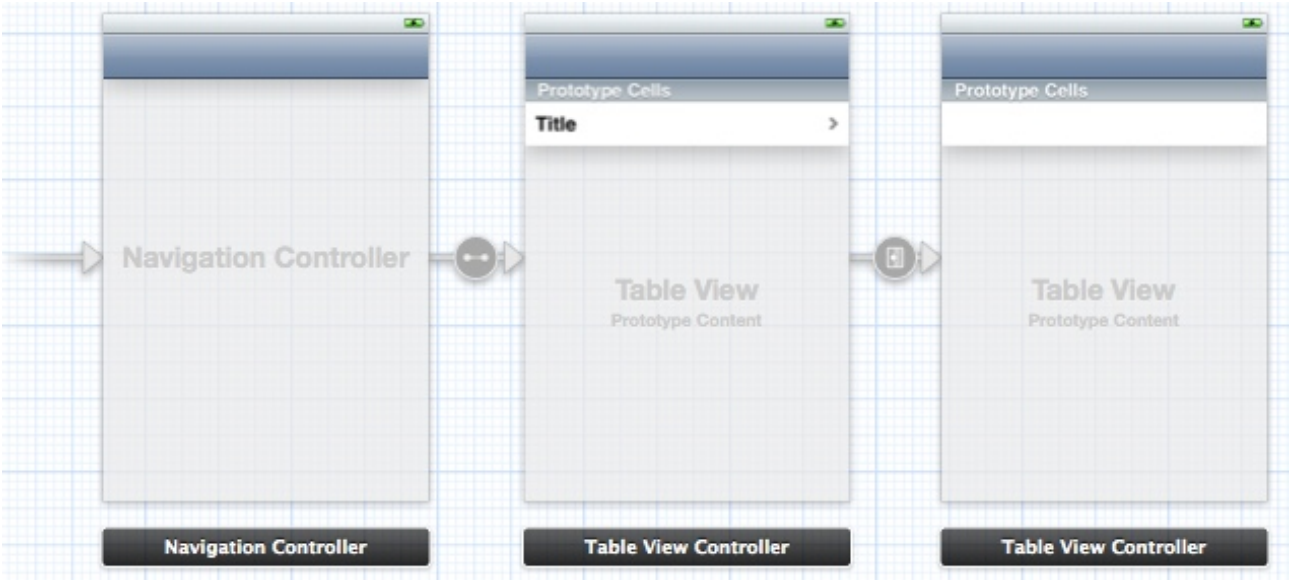
เราเริ่มต้นด้วยการลาก Table View Controller มาวางไว้ใน Storyboard ซึ่ง Storyboard จะกำหนดให้เป็น View แรกในการทำงานทันที

จากนั้นเลือกตัว Table View และเลือกเมนู Editor > Embed In > Navigation Controller ซึ่งจะสร้าง Navigation Controller มาควบคุม Navigation ในโปรแกรม และตั้งค่า Table View Controller ของเราเป็น Root View Controller ของ Navigation อย่างอัตโนมัติ



จะสังเกตว่า Table View ตัวนี้ถูกกำหนดค่าตั้งต้นของ Cell เป็น Prototype Cell มาเรียบร้อยแล้ว ซึ่งตอนนี้ให้เราไปเลือกชนิดของ Cell เป็น Basic ซึ่งเราจะใช้เพื่อแสดงชื่อของภาพยนตร์ และกำหนด AccessoryType เป็น Disclosure Indicator สุดท้ายกำหนด Identifier เป็น MovieCell

จากนั้นเพิ่ม Table View Controller เข้ามาใน Storyboard อีกตัวหนึ่ง และกำหนดให้มี Segue ชื่อ ShowMovieDetail จาก Cell ใน Table View แรกมายัง Table View ตัวที่ใหม่นี้ และให้ทำงานแบบ push ซึ่งเมื่อเรียบร้อยแล้วเราจะได้ภาพรวมของ Storyboard ดังนี้

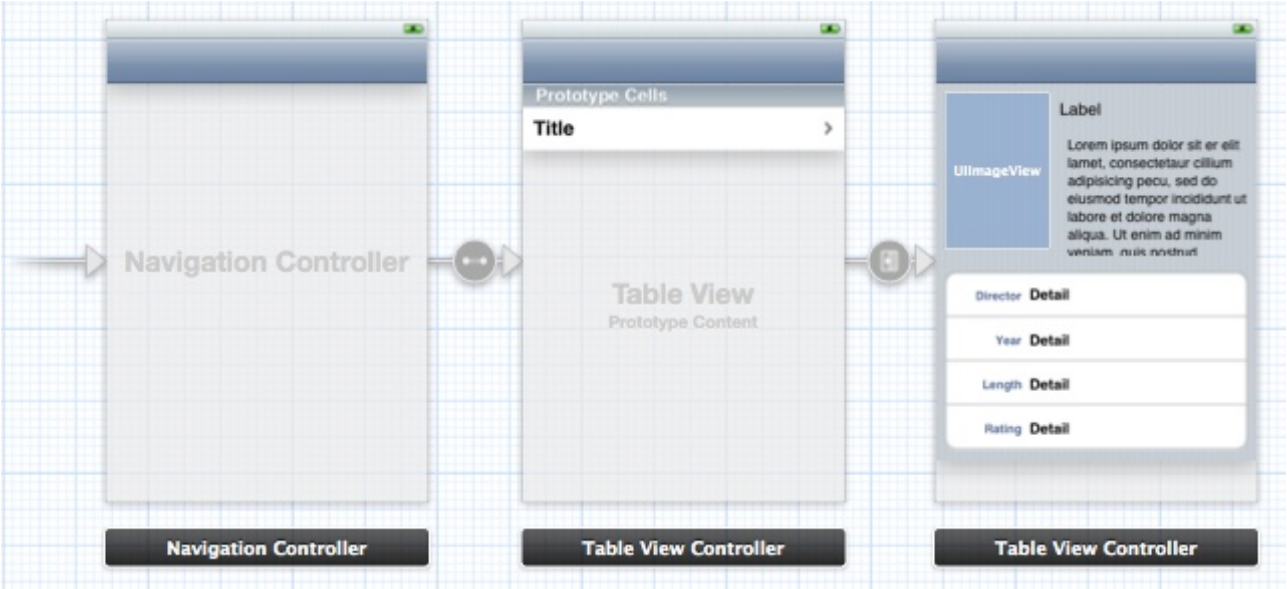


จากนั้นทำการสร้างรูปแบบของ Detail View ดังนี้

1. ปรับ Content ของ Table View ให้เป็นแบบ Static Cells ในทั้งหมด 1 Section และมีลักษณะแบบ Grouped
2. เพิ่ม View ลงไปในส่วนหัวของ Table View ซึ่งจะทำหน้าที่เป็น Header View ให้กับ Table View กำหนดสีของฉากหลังเป็น Clear Color
3. เพิ่ม Image View, Label, Text View ลงในส่วน Header View

4. กำหนดให้ Table View มีทั้งหมด 4 แถว แต่ละแถวมีลักษณะเป็น Left Detail และมี Label ด้านซ้ายเป็น Director, Year, Length, Rating ตามลำดับ

เมื่อเรียบร้อยแล้ว Storyboard ของเราจะมีโครงสร้างและหน้าตาแบบนี้



สร้าง Movie List View Controller

เพิ่ม UITableViewController subclass ชื่อ MovieListViewController เข้าไปในโปรเจค กำหนดให้เป็น Custom Class ให้กับ Table View Controller Placeholder ของ Table View ตัวแรก จากนั้นเขียนโค้ดดังนี้

1. ประกาศ NSArray ชื่อ movies สำหรับเก็บรายการภาพยนตร์ไว้ในส่วน Implementation Data
2. เขียน Method สำหรับสร้างข้อมูลภาพยนตร์ให้กับ movies โดยอาศัยข้อมูลจากเว็บไซต์ที่ให้ข้อมูลภาพยนตร์ทั่วไป เช่น imdb.com เป็นต้น และเรียกใช้ Method นี้ใน viewDidLoad
3. เขียน TableView DataSource เพื่อสร้าง Content ให้กับ Table View โดยเอาข้อมูล name จาก Movie จากแต่ละตำแหน่งมาแสดงใน textLabel

เมื่อเรา Build & Run เราจะได้โปรเจคที่ทำงานได้ และสามารถกดเข้าไปดูในรายละเอียดได้ แต่ตอนนี้ยังไม่ได้เชื่อมโยง View ซึ่งแสดงรายละเอียด ดังนั้นจึงไม่มีข้อมูลอะไร

ดังนั้นงานที่เหลือของเราก็คือ สร้าง View Controller สำหรับแสดงรายละเอียด สร้างความเชื่อใยง และเขียน Segue Method เพื่ออ้างอิงข้อมูลให้ถูกต้อง

สร้าง Movie Detail View Controller

เพิ่ม UITableViewController subclass ชื่อ MovieDetailViewController เข้ามาในโปรเจค จากนั้นตั้งให้เป็น Custom Class ของ Table View Controller ซึ่งจะต้องทำหน้าที่แสดงรายละเอียดของภาพยนตร์แต่ละเรื่อง เมื่อกำหนด Custom Class เรียบร้อยแล้ว ก็มาถึงกระบวนการที่เราคุ้นเคย ก็คือ “การสร้าง Connection” ว่าแล้วก็ลากจาก View ไปยัง Controller Placeholder และสร้าง Outlet ตามนี้เลย (รวมถึงการอ้างอิงวัตถุ Movie ด้วย)

MovieDetailViewController.h

```
@class Movie;
@interface MovieDetailViewController : UITableViewController
@property (weak, nonatomic) IBOutlet UIImageView *imageView;
@property (weak, nonatomic) IBOutlet UILabel *nameLabel;
@property (weak, nonatomic) IBOutlet UITextView *detailTextView;
@property (weak, nonatomic) IBOutlet UILabel *directorNameLabel;
@property (weak, nonatomic) IBOutlet UILabel *yearLabel;
@property (weak, nonatomic) IBOutlet UILabel *lengthLabel;
@property (weak, nonatomic) IBOutlet UILabel *rateLabel;

@property (weak, nonatomic) Movie *movie;
@end
```

จากนั้นก็เขียน Method สำหรับนำข้อมูลจาก Movie เข้ามาใส่ในแต่ละส่วนของ View ซึ่งจะถูกเรียกใช้ใน viewDidLoad

MovieDetailViewController.m

```
- (void)setupView {
    self.nameLabel.text = self.movie.name;
```

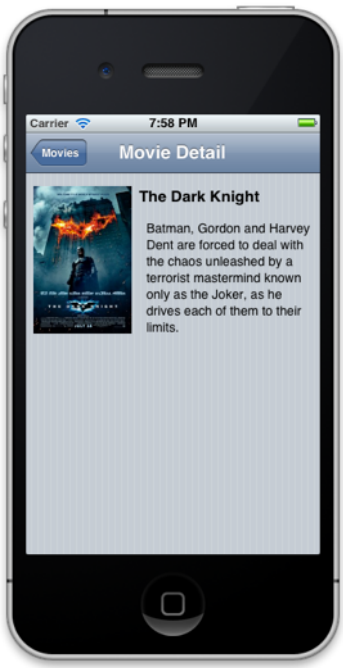
```
    self.detailTextView.text = self.movie.detail;
    self.imageView.image = self.movie.image;
    self.directorNameLabel.text = self.movie.director;
    self.yearLabel.text = [NSString stringWithFormat:@"%d", self.movie.year];
    self.lengthLabel.text = [NSString stringWithFormat:@"%d minutes", self.movie.length];
    self.rateLabel.text = [NSString stringWithFormat:@"%.1f (IMDB)", self.movie.rating];
}
```

เมื่อเรียบร้อยแล้ว ก็เขียน Method เพื่อเตรียม Segue ให้ Movie Detail View Controller อ้างอิง Movie

MovieListViewController.m

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if ([segue.identifier isEqualToString:@"ShowMovieDetail"]) {
        NSIndexPath *indexPath = [self.tableView indexPathForCell:sender];

        MovieDetailViewController *detailViewController = segue.destinationViewController;
        detailViewController.movie = [movies objectAtIndex:indexPath.row];
    }
}
```

จากนั้นเราลอง Build & Run และลองใช้งานโปรแกรม จะพบว่า



Movie Detail View สามารถแสดงรายละเอียดของภาพยนตร์ได้ แต่ว่าเกิดอะไรขึ้น? ทำไม Table View ด้านล่างถึงหายไป? ทั้งๆ ที่ก่อนหน้านี้ตอนที่ยังไม่ได้สร้าง MovieDetailViewController มาเป็น Custom Class ก็แสดงผลได้อย่างถูกต้องแท้ๆ

คำตอบคือ เมื่อเราสร้าง UITableViewController subclass นั้น จะมีโค้ดในส่วนของ Table View DataSource อยู่ด้วย ซึ่ง DataSource นี้จะเป็นตัวที่กำหนดว่า Table View จะมีกี่ส่วนกี่แถว แต่ละแถวมีหน้าตาเป็นอย่างไร และ

การตั้งค่าในส่วนของการออกแบบ ไม่ว่าจะเป็น Interface Builder หรือ Storyboard ก็จะถูกการตั้งค่าในโค้ดตั้งค่าทับเสมอ ดังนั้นส่วน Table View DataSource ใน MovieDetailViewController จึงทับส่วนที่เราออกแบบไว้

ดังนั้นปัญหานี้จึงแก้ไม่ยาก เพราะว่า Static Table View จะไม่ต้องการ Data Source แต่อย่างใด จึงสามารถลบโค้ดส่วนนี้ออกจากไฟล์ได้เลย หรือจะยกเลิกการเชื่อมโยงให้ MovieDetailViewController เป็น DataSource ของ TableView ก็ได้เช่นเดียวกัน (สำหรับผม ผมเลือกที่จะลบโค้ดส่วนนี้ทิ้งครับ)

สร้าง Custom TableView Cell

รู้สึกบ้างไหมครับ ว่า Basic TableView Cell ในหน้าแสดงรายการภาพยนตร์ทั้งหมดมันดูธรรมดา และนานๆ ไปก็ดูน่าเบื่อพอสมควร กับโปรแกรมแสดงรายภาพยนตร์ เราทำอะไรได้บ้างในการทำให้มันมีข้อมูลมากขึ้น ดูดีมากขึ้น

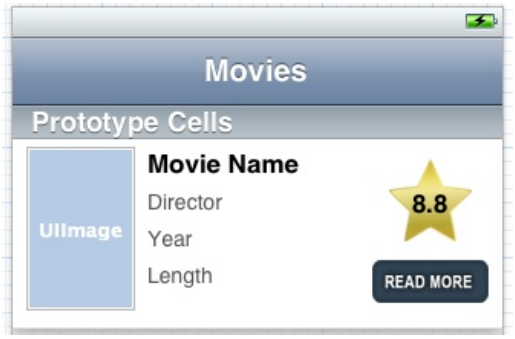
แล้วหน้าตามันควรจะเป็นอย่างไรดีล่ะ? สมมติว่าเรานำหน้าตาเว็บไซต์อย่างเช่น IMDB (Internet Movie Database)

มาเป็นแรงบันดาลใจในการออกแบบ Cell เราอาจจะได้หน้าตาของ Cell ในหน้าแรกแบบนี้

แต่จะต้องทำอย่างไรล่ะ?

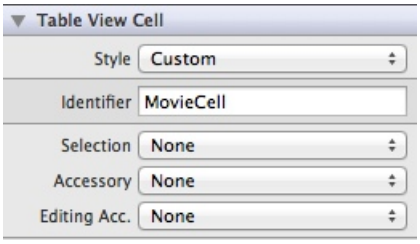
สร้าง Prototype Cell สำหรับ Movie List View

เปิด Storyboard และเลือก Movie List View จากนั้นเปลี่ยนชนิดของ Cell จาก Basic เป็น Custom จากนั้นก็ทำการ “ละเลง” แบบที่เราต้องการ



เริ่มจากการกำหนดความสูงอย่างที่เราต้องการเสียก่อน จากนั้นเพิ่มองค์ประกอบทางส่วนติดต่อกับผู้ใช้เหมือนกับการออกแบบหน้าตาผู้ใช้ในส่วนอื่นๆ โดยเราอาจต้องหากราฟิกส์เพิ่มเติม (เช่นรูปดาว และกราฟิกส์สำหรับปุ่ม Read More)

และเนื่องจากเราต้องการให้กดปุ่ม Read More แทนที่จะเป็นการกดทั้ง Cell ดังนั้นเราจึงเลือกรูปแบบของ Selection เป็น None ด้วย



แล้วอะไรจะเป็นตัวจัดการ TableView Cell ที่หน้าตาประหลาดขนาดนี้? UITableViewCell ธรรมดาไม่น่าจะใช้ได้ ดังนั้นก็ถึงเวลาที่เราต้อง subclass มันเสียแล้ว

สร้าง Movie TableView Cell

เพิ่มไฟล์ใหม่ในโปรเจค โดยเลือกเป็น Objective-C class แต่ให้เลือกเป็น subclass ของ UITableViewCell แทนที่จะเป็น NSObject อย่างที่เราคุ้นเคย ตั้งชื่อว่า MovieTableViewCell

จากนั้นกลับไปใน Storyboard และเลือก Custom Class ให้กับ Prototype Cell เป็น MovieTableViewCell โดยยังคง Identifier เดิมเอาไว้ และเนื่องจากเราจะใช้การกดปุ่ม Read More แทนการแตะทั้ง Cell เพื่อไปยังรายละเอียด เราจึงลบ Segue ตัวเก่าทิ้งไป และสร้าง Segue ตัวใหม่ จากปุ่ม Read More ไปยัง Detail View โดยยังคง Identifier ชื่อเดิมเอาไว้

และเนื่องจาก Storyboard ไม่สามารถที่จะเชื่อมสร้าง Outlet/Action จากภายใน Prototype Cell ไปยังภายในไฟล์ MovieTableViewCell ได้เหมือนที่เราคุ้นเคย ดังนั้นคราวนี้เราจึงจำเป็นต้องเขียนโค้ดก่อน แล้วค่อยเชื่อมทีหลัง

MovieTableViewCell.h

```
@class Movie;
@interface MovieTableViewCell : UITableViewCell
@property (weak, nonatomic) IBOutlet UIImageView *movieImageView;
@property (weak, nonatomic) IBOutlet UILabel *movieNameLabel;
@property (weak, nonatomic) IBOutlet UILabel *directorNameLabel;
@property (weak, nonatomic) IBOutlet UILabel *yearLabel;
@property (weak, nonatomic) IBOutlet UILabel *lengthLabel;
@property (weak, nonatomic) IBOutlet UILabel *ratingLabel;

-(void)configureWithMovie:(Movie *)movie;
@end
```

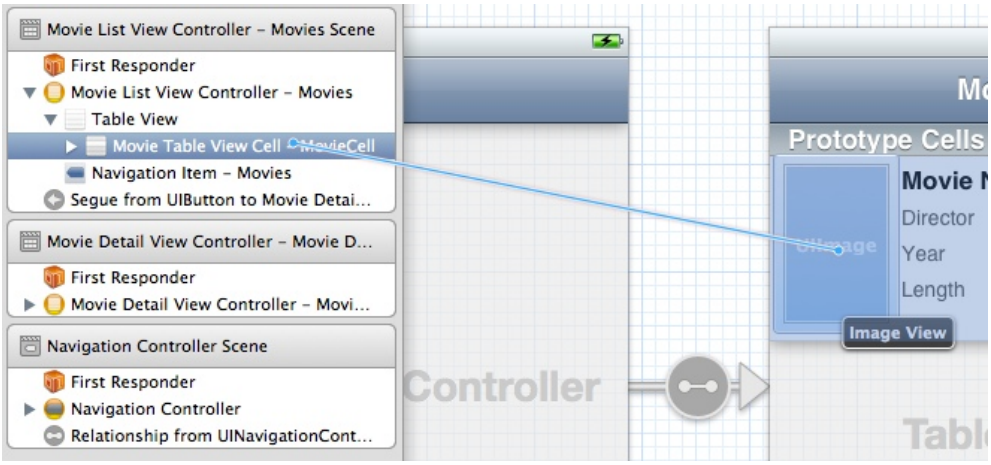
สังเกตว่าผมได้เตรียม Method สำหรับตั้งค่า Cell เอาไว้ใน MovieTableViewCell นี้ด้วย แทนที่จะตั้งค่าใน MovieListViewController (อาจจะใน Method tableView:cellForRowAtIndexPath: หรือ Method อื่นๆ ที่สร้างแยกออกมาและถูกเรียกโดย tableVIew:cellForRowAtIndexPath: อีกทีหนึ่งก็ตาม) ทั้งนี้ก็เพราะว่า ไหนๆ เราก็สร้างวัตถุซึ่งจำเพาะเจาะจงขึ้นมาเองแล้ว วัตถุตัวนี้ก็ควรจะรู้วิธีการตั้งค่าให้กับสิ่งที่มันต้องดูแล

จากนั้นทำการ Implement MovieTableViewCell ซึ่งเราต้องทำการ Implement เพียง Method เดียวเท่านั้น

MovieTableViewCell.m

```
-(void)configureWithMovie:(Movie *)movie {
    self.imageView.image = movie.image;
    self.movieNameLabel.text = movie.name;
    self.directorNameLabel.text = [NSString stringWithFormat:@"By: %@", movie.director];
    self.yearLabel.text = [NSString stringWithFormat:@"Released: %d", movie.year];
    self.lengthLabel.text = [NSString stringWithFormat:@"Length: %d mins", movie.length];

    self.ratingLabel.text = [NSString stringWithFormat:@"%.1f", movie.rating];
}
```

จากนั้นเชื่อม Outlet ทั้งหมดผ่าน Storyboard ซึ่งเราจะต้องเปิด Object Browser ด้านซ้ายของ Storyboard เสียก่อน และลากจาก Movie TableView Cell มายังองค์ประกอบต่างๆ ใน Prototype Cell



จากนั้นกลับไปแก้ Movie List View Controller เพื่อให้สร้าง MovieTableViewCell แทนที่จะเป็น UITableViewCell ซึ่งเราจะใช้ Method configureWithMovie: เพื่อให้การสร้าง Cell เป็นไปโดยง่าย

MovieListViewController.m

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

    Movie *movie = [movies objectAtIndex:indexPath.row];
    [cell configureWithMovie:movie];

    return cell;
}
```

เมื่อเราลอง Build & Run จะเห็นว่า MovieTableViewCell ถูกนำมาใช้งานในการแสดงผลเรียบร้อย เมื่อเรากดลงไปบนตัว Cell ก็ไม่เกิดอะไรขึ้น (Segue ไม่ทำงาน) และเมื่อเรากดปุ่ม Read More ก็แสดงรายละเอียดของภาพยนตร์ใน Detail View .... แต่

แก้ไขการทำงานของ Segue

เราพบว่าไม่ว่าจะกดปุ่ม Read More บน Cell ไหน ก็ยังแสดงผลภาพยนตร์จาก Cell แรกเสมอ เรื่องนี้เกิดขึ้นได้อย่างไร?

ย้อนกลับไปดู Method ที่จะต้องเตรียมข้อมูลให้กับ Segue จะพบว่าตอนนี้การทำงานเป็นแบบนี้

MovieListViewController.m

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if ([segue.identifier isEqualToString:@"ShowMovieDetail"]) {
        NSIndexPath *indexPath = [self.tableView indexPathForCell:sender];

        MovieDetailViewController *detailViewController = segue.destinationViewController;
        detailViewController.movie = [movies objectAtIndex:indexPath.row];
    }
}
```

จะพบว่าสาเหตุของปัญหาเกิดขึ้นจากการอ้างอิงดัชนีของ NSArray จาก indexPath ของ Cell ที่ถูกกด เนื่องจากตอนที่เราเขียนครั้งแรกนั้น Cell จะเป็น sender ของ Segue แต่ตอนนี้ sender นั้นเปลี่ยนไปเป็น UIButton ซึ่งอยู่ใน Cell ไม่ใช่ Cell ดังนั้นจึงต้องแก้ไขตรงนี้ให้อ้างอิง Cell ได้อย่างถูกต้อง

แล้วเราจะอ้างอิงไปยัง Cell ได้อย่างไร?

เนื่องจาก UIButton นี้จะอยู่ใน UITableViewCellContentView ซึ่งจะอยู่ใน MovieTableViewCell อีกทีหนึ่ง ดังนั้นการที่จะอ้างอิงกลับไปยัง Cell ตัวที่มี UIButton ที่ถูกกดอยู่นั้น ก็ทำได้โดยอ้างอิงไปหา Super View เป็นทอดๆ ซึ่งทำให้เราแก้โค้ดดังนี้

MovieListViewController.m

```
-(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if ([segue.identifier isEqualToString:@"ShowMovieDetail"]) {
        MovieTableViewCell *selectedCell = (MovieTableViewCell *)[[sender superview] superview];

        NSIndexPath *indexPath = [self.tableView indexPathForCell:selectedCell];

        MovieDetailViewController *detailViewController = segue.destinationViewController;
        detailViewController.movie = [movies objectAtIndex:indexPath.row];
    }
}
```

เมื่อเราลอง Build & Run ดูอีกครั้ง จะพบว่าโปรแกรมของเราทำงานได้อย่างถูกต้องเรียบร้อยแล้ว

สรุปท้ายบท

บทนี้เราได้เรียนรู้เรื่องที่สำคัญมากในการพัฒนา iOS App ในปัจจุบัน เพราะว่า Table View แบบมาตรฐานๆ มันมักไม่พอเสียแล้ว เท่านั้นไม่พอ หลายครั้งเรายังอยากประยุกต์ใช้ Table View ในการแสดงผลอะไรต่ออะไร ที่ไม่ใช่รายการซ้ำๆ กัน ซึ่งจะทำได้ค่อนข้างยาก เพราะ DataSource Method ของ TableView ไม่ได้ถูกออกแบบมาแบบนั้น โดยสรุปสิ่งที่เราได้ทำคือ

- สร้าง iOS App ที่มีการใช้งานทั้ง Custom Table View Cell และ Static Table View (แต่เนื้อหาข้างในอาจจะเปลี่ยนตามข้อมูล)
- เรียนรู้การสร้าง Table View Controller สำหรับ Static Table View
- เรียนรู้การสร้าง UITableViewCell subclass สำหรับการควบคุม Custom Cell
- เรียนรู้วิธีการตั้งค่าต่างๆ ให้กับ Custom Cell
- เรียนรู้วิธีการทำงานกับ Storyboard ในการสร้าง Table View Application ระดับสูงขึ้น
- เรียนรู้วิธีการทำงานกับ Segue มากยิ่งขึ้น

สำหรับสิ่งใหม่ๆ ใน Xcode 4.2 ที่เราได้พบกันมา 4 บทติดๆ กัน ก็คงจะจบแต่เพียงเท่านี้ และบทต่อไปเราจะหักลำ 180 องศา ตรงข้ามกับ Interface Builder, Storyboard โดยสิ้นเชิง เพราะว่ามันจะเป็นการ “สร้าง View ด้วยการเขียนโค้ดล้วนๆ” ซึ่งในการทำงานจริง สุดท้ายเรามักจะ “หนีไม่พ้น” ครับ