

آموزش گام به گام برنامه نویسی

# iOS

## به زبان Swift

مولف: مهندس افشین رفوآ



Swift



۷	درس ۱ : توسعه ی اپلیکیشن های ios بر اساس Swift
۷	شروع
۷	درباره ی اپلیکیشن
۸	دسترسی به ابزار لازم
۹	درس ۲ : آموزش مبانی و مفاهیم اصلی Swift
۹	آموزش برنامه نویسی IOS بر اساس Swift – آموزش مبانی و مفاهیم اصلی Swift
۹	آنچه خواهید آموخت
۱۰	انواع داده ای (Basic type) در Swift
۱۵	کنترل جریان یا روند اجرای برنامه (Control Flow)
۱۹	تشریح تفاوت بین function و method
۲۱	تشریح مفهوم class و initializer
۲۷	تشریح مفهوم Enumeration ها و Structure ها در Swift
۳۱	شرح مفهوم Protocol (الگوی پیاده سازی)
۳۴	Swift و فریم ورک طراحی رابط کاربری Cocoa Touch
۳۶	درس ۳ : ساخت رابط کاربری در Swift
۳۶	ساخت یک رابط کاربری UI/ساده
۳۷	آنچه خواهید آموخت
۳۸	ایجاد پروژه ی جدید
۴۲	آشنایی بیشتر با محیط کاری Xcode
۴۳	راه اندازی محیط شبیه ساز (Simulator)
۴۸	مشاهده و بررسی Source code پروژه
۴۹	فایل App Delegate
۴۹	فایل AppDelegate.swift دو کاربرد اصلی دارد:
۵۱	فایل View Controller
۵۳	مشاهده ی محتوای فایل Storyboard
۵۵	ساخت یک رابط کاربری UI/ساده
۶۷	پیش مشاهده ی User Interface برنامه
۷۰	استفاده از موتور نمایش Auto Layout
۸۳	درس ۴ : اتصال UI به کد Swift
۸۳	متصل کردن UI (ظاهر یا رابط کاربری برنامه) به کد (رفتارها و عملیاتی که برنامه انجام می دهد)

۸۴	آنچه خواهید آموخت.....
۸۵	متصل کردن UI به کد برنامه (Source Code) .....
۸۸	تعریف Outlet برای المان های UI .....
۹۵	تعریف یک رخداد با استفاده از Action .....
۱۰۱	پردازش ورودی کاربر (برابر قرار دادن مقدار label با مقدار وارد شده توسط کاربر در text field) .....
۱۱۱	درس ۵ : چرخه حیات view controller .....
۱۱۱	آموزش کار با view controller ها .....
۱۱۲	آنچه خواهید آموخت.....
۱۱۳	آشنایی با life cycle/چرخه ی حیات View Controller و مفهوم آن .....
۱۱۶	افزودن یک عکس به UI برنامه .....
۱۲۲	به نمایش گذاشتن یک عکس مکان نگهدار در image view .....
۱۲۶	متصل کردن المان image view به کد مربوطه در فایل ViewController.swift .....
۱۳۰	تعریف یک Gesture Recognizer (اعطای قابلیت های یک control به view) .....
۱۳۳	متصل کردن gesture recognizer به کد مربوطه در فایل ViewController.swift .....
۱۳۵	افزودن یک image picker به پروژه جهت تعامل با کاربر و فراهم آوردن امکان انتخاب عکس .....
۱۴۴	درس ۶ : کنترل اختصاصی (Custom control) در Swift .....
۱۴۴	پیاده سازی یک کنترل اختصاصی (Custom control) .....
۱۴۵	آنچه خواهید آموخت.....
۱۴۶	تعریف یک View اختصاصی .....
۱۵۰	به نمایش گذاشتن view اختصاصی .....
۱۵۲	در حال حاضر، UI برنامه می بایست ظاهری مشابه زیر داشته باشد. ....
۱۵۵	افزودن کنترل Button به View .....
۱۶۸	تنظیم میزان فاصله ی دکمه ها از هم و تعداد آن ها (در قالب دو متغیر/property مجزا) .....
۱۶۹	تعریف یک ثابت جهت تعیین اندازه ی کنترل دکمه .....
۱۷۳	جایگزین کردن عکس ستاره بجای مربع های قرمز رنگ.....
	جایگزین کردن پیاده سازی debugging متد ratingButtonTapped() با پیاده سازی واقعی (پیاده سازی
۱۷۹	action متصل به UIButton) .....
۱۸۴	متصل کردن کنترل امتیاز دهی به کد view controller .....
۱۸۷	پاک سازی کد و اطلاعات غیرضروری از پروژه (Cleanup) .....
۱۹۲	درس ۷ : ساخت Data Model برای اپلیکیشن های IOS .....
۱۹۲	ساخت Data Model برای اپلیکیشن .....
۱۹۲	آنچه خواهید آموخت.....

۱۹۲	..... ساخت data model
۲۰۷	..... درس ۸ : تعریف Table View در Swift
۲۰۷	..... تعریف Table View در Swift- ساخت یک scene با نمای جدولی
۲۰۸	..... آنچه خواهید آموخت
۲۳۱	..... افزودن عکس به پروژه
۲۳۳	..... ایجاد اتصال بین UI و کد به وسیله ی outlet (وصل کردن UI نمای جدولی به کد)
۲۴۱	..... بارگذاری داده های اولیه ی اپلیکیشن
۲۵۵	..... آماده سازی صفحه محتوای افزودن غذای جدید/Meal Scene برای پیمایش (پیاده سازی قابلیت پیمایش در برنامه ی FoodTracker)
۲۶۰	..... درس ۹ : آموزش پیاده سازی قابلیت پیمایش (navigation) در Swift
۲۶۰	..... پیاده سازی قابلیت پیمایش (navigation)
۲۶۱	..... آنچه خواهید آموخت
۲۶۲	..... اضافه کردن یک segue جهت پیمایش به جلو (صفحه ی بعدی)
۲۶۶	..... تنظیم و پیکربندی Navigation Bar برای scene ها
۲۸۲	..... تکمیل ظاهر برنامه (UI) با بهره گیری موتور نمایش/مدیریت چیدمان المان ها Auto Layout
۲۸۷	..... ذخیره نمودن غذاهای جدید در لیست غذاها (Meal List)
۲۹۱	..... تعریف قابلیت پیمایش به عقب (unwind Segue)
۳۰۰	..... غیر فعال نمودن قابلیت ذخیره (دکمه ی Save) زمانی که کاربر اسم آیتم را وارد نکند
۳۰۴	..... پیاده سازی قابلیت لغو فرایند اضافه و ذخیره کردن آیتم جدید (پیاده سازی رفتار مربوط به دکمه ی Cancel)
۳۰۷	..... درس ۱۰ : آموزش Edit و Delete در Swift
۳۰۷	..... پیاده سازی قابلیت حذف و ویرایش
۳۰۷	..... آنچه خواهید آموخت
۳۰۷	..... پیاده سازی امکان ویرایش غذاهای جاری
۳۲۳	..... افزودن قابلیت لغو ویرایش آیتم جاری و بازگشت به صفحه ی لیست غذاها بدون ذخیره ی تغییرات (بروز رسانی کد متصل به دکمه ی Cancel)
۳۲۷	..... پیاده سازی امکان حذف آیتم ها (اضافه کردن دکمه ی Delete)
۳۳۲	..... درس ۱۱ : آموزش Swift Data Persistence
۳۳۲	..... ماندگار سازی یا ذخیره دائمی داده های برنامه (Data Persistence)
۳۳۲	..... آنچه خواهید آموخت
۳۳۳	..... پیاده سازی قابلیت ذخیره ی دائم و بارگذاری آیتم (Save/Load مقدار meal در کلاس Meal)
۳۴۱	..... ذخیره و بارگذاری لیست نمایش غذاها (Meal list)



## مقدمه

زکات علم نشر آن است - حضرت علی (ع)

موسسه آموزشی تحلیل داده ، با حضور جمعی از متخصصین مجرب در زمینه برنامه نویسی در نظر دارد مطالب آموزشی خود را در قالب کتاب های آموزشی و فیلم ، به صورت رایگان در دسترس عموم قرار دهد تا حتی آن دسته از عزیزانی که بنا به دلایل مالی ، مسافت جغرافیایی و یا نداشتن وقت کافی ، امکان شرکت در دوره های حضوری برای آنها میسر نیست ، از یادگیری بی بهره نمانند .

علاوه ب راین علاقه مندان می توانند با ثبت نام در انجمن سایت تحلیل داده ، سواات خود را مطرح نموده و مدرسین آموزشگاه و اعضای انجمن در اسرع وقت ، پاسخ های خود را حتی الامکان به صورت فیلم در دسترس عموم قرار دهند .

لذا از کلیه فعالان در این زمینه دعوت میشود در این حرکت جمعی در کنار ما باشند و با حضور فعال خود در انجمن گام موثری در بهبود سطح علمی جوانان کشور عزیزمان ایران بردارند .

آدرس سایت : <http://www.tahlildadeh.com>

**توجه :**

برای دانلود سورس کد مثال های کتاب ، [اینجا](#) را کلیک کنید .

## درس ۱ : توسعه ی اپلیکیشن های iOS بر اساس Swift

### شروع

آموزش حاضر نقطه ی شروع مناسبی برای ساخت اپلیکیشن هایی است که بر روی iPhone و iPad نصب و اجرا می شوند. می توانید این سری آموزشی گام به گام را به مثابه ی سکوی پرش جهت ساخت اولین اپلیکیشن خود همراه با ابزار، مفاهیم اصلی و بهترین روش ها (best practices) که مسیر شما در برنامه نویسی تحت موبایل برای سیستم عامل) iOS براساس زبان (Swift را آسان می سازد، در نظر بگیرید. هر مبحث یک آموزش جدید را دربرگرفته و مفاهیم لازم برای درک و پیاده سازی مطالب آن را در اختیار شما قرار می دهد. گفتنی است که آموزش ها به هم مرتبط بوده (در واقع هر آموزش ادامه ی آموزش قبلی بوده) و شما را در ساخت یک اپلیکیشن ساده اما کامل iOS به صورت گام به گام راهنمایی می کنند. در طول مطالعه ی مباحث و ساخت اپلیکیشن به صورت مرحله به مرحله، مفاهیم ضروری در توسعه ی برنامه های کاربردی iOS را فراگرفته و درک عمیق تری از زبان برنامه نویسی Swift پیدا می کنید و همچنین با امکانات جالب محیط برنامه نویسی Xcode آشنا می شوید.

### درباره ی اپلیکیشن

در طی مباحث سری آموزشی حاضر، یک اپلیکیشن ساده ی مشاهده و ثبت اطلاعات غذا به نام FoodTracker را خواهید ساخت. این برنامه، همان طور که از نامش پیدا است، فهرستی از غذاها و اطلاعات مربوط به آن ها از جمله اسم غذا، درجه ی کیفی و عکس غذا را نمایش می دهد. کاربر می تواند غذای جدید به لیست اضافه نموده و غذای جاری را ویرایش یا حذف کند. برای افزودن غذای جدید یا ویرایش غذای جاری، کاربران ابتدا به صفحه ی دیگری رفته و سپس یک اسم، عکس و درجه کیفی برای غذای دلخواه وارد می نمایند.

اولین درس صرفاً یک فایل ساده ی Xcode است (playground) که به شما اجازه می دهد با کد تعامل داشته (تغییراتی را در آن ایجاد کنید) و نتیجه را مستقیماً مشاهده

نمایید. فایل مزبور را دانلود کرده، آن را در محیط Xcode باز می کنید و از طریق آن با مفاهیم اصلی زبان Swfit آشنا می شوید. در هر یک از مباحث بعدی، فایل پروژه ی Xcode اپلیکیشن در اختیار شما قرار می گیرد که نمای نهایی از کد و interface (رابط کاربری) را ارائه می دهد. پس از اتمام هر درس، می توانید پروژه را دانلود کرده و کار خود را با نسخه ی مرجع مقایسه نمایید.

### دسترسی به ابزار لازم

جهت توسعه ی اپلیکیشن های IOS با استفاده از آخرین تکنولوژی های تشریح شده در این مباحث، بایستی یک کامپیوتر Mac با سیستم عامل OS X 10.10 یا بالاتر) که آخرین ورژن محیط برنامه نویسی Xcode بر روی آن نصب شده، در اختیار داشته باشید . Xcode دربردارنده ی تمامی امکاناتی است که برای طراحی، توسعه و عیب یابی (debug) اپلیکیشن های IOS لازم دارید. به این محیط برنامه نویسی IOS SDK نیز اضافه می شود که قابلیت های Xcode را افزایش داده و ابزار، کامپایلرها و فریم ورک های مورد نیاز برای توسعه ی اپلیکیشن های IOS را در اختیار برنامه نویس قرار می دهد. می توانید آخرین ویرایش Xcode را رایگان از App Store دانلود کرده و بر روی کامپیوتر Mac خود نصب نمایید.

جهت دانلود آخرین ورژن Xcode :

- اپلیکیشن App Store را بر روی کامپیوتر Mac خود اجرا نمایید (به صورت پیش فرض این اپلیکیشن در نوار Dock قابل دسترسی می باشد) .
  - در فیلد جستجو، مقیم در بالای صفحه سمت راست، واژه ی Xcode را وارد نموده و سپس کلید Return را فشار دهید. برنامه ی Xcode به عنوان اولین نتیجه ی جستجو نمایش داده می شود.
  - ابتدا بر روی Get و سپس جهت نصب برنامه بر روی Install App کلیک نمایید.
  - زمانی که از شما درخواست می شود، Apple ID و گذرواژه ی خود را وارد نمایید .
- Xcode دانلود شده و در پوشه ی Applications / جایگذاری می شود.

- Double : دو خط حاشیه تعریف می کند.
- Groove : یک خط حاشیه ی سه بعدی نمایش می دهد که در آن خطوط بالایی و سمت چپ پررنگ تر از خطوط دیگر هستند.
- Ridge : در این حالت خطوط حاشیه به صورت سه بعدی و با طیف رنگی نمایش داده می شوند.
- Inset : این مقدار یک حاشیه ی داخلی سه بعدی تعریف می کند که در آن معمولاً خطوط بالایی و سمت چپ دارای سایه و تیره تر از خطوط دیگر هستند.
- Outset : حاشیه ی سه بعدی خارجی تعریف می کند که در آن خطوط پایینی و سمت راست دارای سایه و تیره تر از خطوط دیگر هستند.

**نکته :**

توجه داشته باشید که هیچ یک از خاصیت های این آموزش از Xcode 7.0 و SDK 9.0 استفاده می شود. لازم است شما نیز از این نسخه ها استفاده نمایید .

## درس ۲ : آموزش مبانی و مفاهیم اصلی Swift

آموزش برنامه نویسی IOS بر اساس – Swift آموزش مبانی و مفاهیم اصلی Swift

اولین آموزش در قالب یک فایل Swift playground همراه با آموزش و راهنمایی لازم ارائه می شود playground . یک نوع فایل است که به شما اجازه می دهد در محیط برنامه نویسی Xcode با کد تعامل داشته، آن را تغییر دهید و نتایج را مستقیماً مشاهده نمایید. فایل های playground برای یادگیری و آزمایش مطالب بسیار مناسب بوده و فایلی که در این درس ارائه شده به شما امکان می دهد تا با مفاهیم پایه ای Swift آشنا شوید.

## آنچه خواهید آموخت

با مطالعه ی کامل مبحث قادر خواهید بود:

۱. فرق بین ثوابت (constant) و متغیرها را بدانید .

۲. بدانید چه هنگام از اعلان نوع به صورت ضمنی (implicit declaration) و چه هنگام از اعلان نوع به صورت صریح (explicit declaration) استفاده نمایید.
۳. با فایده‌ی استفاده از optional ها و optional binding ارسال پارامترهایی از نوع optional آشنا شوید .
۴. فرق بین optional ها و implicitly unwrapped optional ها را بدانید .
۵. هدف استفاده از حلقه و دستورات شرطی را درک کنید.
۶. با استفاده از switch امکان انتخاب بین چندین گزینه در صورت برقرار بودن شرط خاص را فراهم آورید (ورای شرط های باینری که در صورت برقرار بودن شرط می توان از بین دو گزینه انتخاب انجام داد).
۷. با استفاده از عبارات where ، محدودیت های (constraint) بیشتری در دستورات شرطی اعمال نمایید .
۸. فرق بین تابع (function) ، متد (method) و initializer را تشخیص دهید .
۹. فرق بین class ، structure و enumeration را بدانید .
۱۰. ساختار نگارشی یا سینتکس ساده برای ارث بری (inheritance) از کلاس دیگر و پیروی از الگوی پیاده سازی یا protocol خاص برای ساختارهای class ، structure و enumeration را بدانید.
۱۱. تشخیص نوع هایی که به صورت ضمنی اعلان شده اند و استفاده از امکان option-click جهت کسب اطلاعات بیشتر در محیط کاری Xcode.
۱۲. وارد (import) کردن کتابخانه ی UIKit و استفاده از کلاس های آن در پروژه .

### انواع داده ای (Basic type) در Swift

Constant (ثابت) مقداری است که پس از اعلان، ثابت باقی می ماند. این در حالی است که مقدار variable پس از تعریف قابل تغییر می باشد constant را immutable نیز می گویند، بدین معنی پس از تعریف، امکان تغییر (مقدار) آن وجود ندارد. اما variable

یا متغیر را mutable یا به اصطلاح قابل تغییر می نامند (در بخش های مختلف برنامه مقدارش قابل تغییر است.)

چنانچه از عدم تغییر مقدار در سرتاسر کد خود اطمینان دارید، در آن صورت بهتر است بجای متغیر از ثابت یا constant استفاده نمایید .

جهت تعریف constant از کلیدواژه ی let و به منظور تعریف متغیر از var استفاده کنید:

```
var myVariable = 42
myVariable = 50
let myConstant = 42
```

هر ثابت یا متغیری که در Swift تعریف می کنید، یک نوع دارد. اما لازم نیست همیشه نوع داده ای را به صورت صریح مشخص نمایید. با مقداری که به متغیر یا ثابت تخصیص می دهید، در واقع نوعش را نیز به کامپایلر اعلان می کنید (با انتساب مقدار به ثابت یا متغیر تعریف شده، به کامپایلر اجازه می دهید نوع آن را حدس بزند). در مثال بالا، خط دوم کد، کامپایلر خود به نوع متغیر myVariable پی می برد (و آن را به عنوان integer می شناسد) چرا که مقدار اولیه ی آن، که در خط اول مشخص شده، از نوع integer است. از این قابلیت کامپایلر به عنوان type inference حدس یا استنتاج نوع داده ای) یاد می شود. پس از اعلان نوع ثابت یا متغیر، امکان تغییر نوع آن وجود ندارد .

اگر مقدار اولیه، اطلاعات لازم درباره ی متغیر یا ثابت ارائه نکند (یا اساساً هیچ مقدار اولیه ای وجود نداشته باشد)، می توانید نوع را پس از اسم متغیر و دو نقطه ذکر نمایید .

```
let implicitInteger = 70
let implicitDouble = 70.0
let explicitDouble: Double = 70
```

**امتحان و تجربه کنید**

داخل محیط Xcode ، جهت اطلاع از نوع (تشخیص داده شده توسط کامپایلر)، بر روی اسم ثابت یا متغیر option-click نمایید. می توانید عملیات ذکر شده را بر روی ثوابت فوق اجرا کنید.

مقادیر هیچگاه به صورت ضمنی (از نوعی) به نوع دیگر تبدیل نمی شوند. در صورت نیاز به تبدیل نوع مقدار مورد نظر، کافی است (به صورت صریح) نمونه ای از نوع دلخواه را ایجاد نمایید. در نمونه ی زیر، مقدار یک ثابت را از نوع int به string تبدیل می کنیم:

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

**امتحان و تجربه کنید**

تبدیل نوع به string را از آخرین خط کد حذف نمایید. با چه خطایی مواجه می شوید؟

راه آسان تری نیز برای اضافه کردن مقادیر در رشته ها وجود دارد؛ داخل رشته ی مورد نظریک "\" تایپ کرده، سپس مقادیر دلخواه را از طریق پرانتز به داخل رشته ی مورد نظر تزریق کنید. از این پروسه تحت عنوان string interpolation یا درج مقدار جدید در رشته یاد می شود .

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let fruitSummary = "I have \(apples + oranges) pieces of fruit."
```

برای کار با متغیرهایی که ممکن است مقدار داشته یا نداشته باشد، می توانید از optional ها استفاده نمایید. optional متغیری است که یا مقداری دارد و یا به نشانه ی تهی بودن حاوی nil می باشد، به عبارتی دیگر یا مقدار دارد یا از مقدار تهی می باشد.



جهت اعلان یک متغیر به عنوان optional، کافی است یک علامت "?" پس از نوع مقدار تایپ کنید.

```
let optionalInt: Int? = 9
```

برای استخراج (و دسترسی به) مقدار (اصلی و اولیه ی) یک متغیر optional، لازم است آن را unwrap کنید. در مباحث آینده با نحوه ی unwrap کردن optional ها آشنا خواهید شد، اما در خلاصه باید گفت که برای دسترسی به مقدار optional از عملگر "!" استفاده می شود. لازم به ذکر است که شما تنها زمانی باید از عملگر مزبور استفاده کنید که مطمئن باشید مقدار اصلی یا زیرین nil نیست.

```
let actualInt: Int = optionalInt!
```

Optional ها در Swift کاربرد زیادی دارند و در مواقعی که ممکن است مقداری وجود داشته یا نداشته باشد، می توانند فوق العاده مفید واقع شوند. از optional ها می توان به خصوص برای تبدیل نوع به صورت آزمایشی بهره گرفت.

```
var myString = "7"
var possibleInt = Int(myString)
print(possibleInt)
```

در این کد، مقدار متغیر possibleInt برابر ۷ است چرا که نوع متغیر myString از رشته به عدد صحیح تبدیل شده است و پس از عملیات تبدیل حاوی مقدار تبدیل شده می باشد. اما اگر myString را نوعی تعریف کنید که قابل تبدیل به integer نباشد، آنگاه مقدار متغیر possibleInt، nil یا تهی خواهد بود.

```
myString = "banana"
possibleInt = Int(myString)
print(possibleInt)
```

آرایه یا array نوع داده ای است که مجموعه ای مرتب از آیتم ها را در خود جای می دهد. برای ایجاد آرایه از [] استفاده نمایید و برای دسترسی به المان های داخل آن، اندیس المان مورد نظر را در علامت [] ذکر کنید. اندیس آرایه از ۰ آغاز می شود.

```
var ratingList = ["Poor", "Fine", "Good", "Excellent"]
ratingList[1] = "OK"
ratingList
```

جهت ایجاد آرایه ی تهی، کافی است از initializer استفاده کنید. به زودی به شرح initializer خواهیم پرداخت.

```
// Creates an empty array.
یک آرایه ی خالی تعریف می کند
let emptyArray = [String]()
```

همان طور که مشاهده می کنید، در کد بالا از comment استفاده شده است. Comment یا توضیحات یک قطعه متن (در کد برنامه) است که به عنوان بخشی از برنامه کامپایل یا ترجمه نمی شود اما توضیحاتی و اطلاعات مفیدی را درباره ی بخش های مختلف کد ارائه می دهد. comment ها به دو دسته تقسیم می شوند: ۱. Comment های تک خطی که پس از // درج می شود ۲. Comment های چند خطی که بین /\* و \*/ قرار می گیرند. هر دو نمونه ی نام برده را در بخش های مختلف کد برنامه مشاهده خواهید کرد.

Implicitly unwrapped optional یک متغیر از نوع optional است که می توان از آن مانند یک متغیر nonoptional و عادی استفاده کرد، بدون اینکه لازم باشد هر بار برای دسترسی به مقدارش (مقدار متغیر optional) آن را unwrap کنید. دلیلش این است که یک implicitly unwrapped optional پس از اعلان و تنظیم مقدار اولیه، همیشه حاوی مقدار در نظر گرفته می شود، هرچند مقدار آن می تواند تغییر کند. برای تعریف

متغیرهایی از نوع optional unwrapped implicitly، بایستی بجای "?" از "!" استفاده نمایند.

```
var implicitlyUnwrappedOptionalInt: Int!
```

### کنترل جریان یا روند اجرای برنامه (Control Flow)

Swift از دو نوع دستور (دستورات شرطی و حلقه های تکرار) برای کنترل جریان اجرای برنامه بهره می گیرد. دستورات شرطی یا Conditional statements نظیر if و switch، قبل از اجرای دستور معین، بررسی می کنند آیا یک شرط صحیح/برقرار است یا خیر. بدین معنی که قبل از اجرای دستور معین، ابتدا شرط را ارزیابی می کنند و سپس در صورت درست بودن شرط (ارزیابی شرط به مقدار بولی true)، آن دستور را اجرا می کنند. و دیگری حلقه های تکرار نظیر for-in و while که یک قطعه کد یا مجموعه دستور را مادامی که شرط خاصی برقرار است، تکرار می کنند.

دستور if بررسی می کند آیا شرط خاصی صادق است یا خیر و سپس در صورت درست بودن شرط، if (کد داخل) دستور را بررسی و اجرا می کند. می توان با اضافه کردن دستور else به if، رفتار پیچیده تری تعریف کرد. همچنین می توان از دستور else برای متصل کردن دستورات if به هم استفاده کرد و یا آن را به طور مستقل بکار برد که در آن صورت دستور else، اگر هیچ یک از دستورات if زنجیره ای نتیجه ی صحیح نداشته و صادق نباشند، اجرا خواهد شد.

```
let number = 23
if number < 10 {
    print("The number is small")
} else if number > 100 {
    print("The number is pretty big")
} else {
    print("The number is between 10 and 100")
}
```

**امتحان و تجربه کنید**

مقدار number را به عدد صحیح دیگری تغییر داده و پس از اجرای کد ببینید کدام دستور اجرا می شود (کدام رشته چاپ می شود).

دستورات را می توان جهت تعریف رفتار و عملیات پیچیده، داخل هم قرار داد یا به اصطلاح تودرتو (nest) کرد. در زیر یک دستور if..else را مشاهده می کنید که در بدنه ی یک حلقه ی for-in قرار داده شده است (این حلقه داخل آرایه چرخیده و تک تک آیتم های آن را به ترتیب می خواند).

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0
for score in individualScores {
  if score > 50 {
    teamScore += 3
  } else {
    teamScore += 1
  }
}
print(teamScore)
```

می توانید با بهره گیری از optional binding در دستور if، بررسی کنید آیا تغییری که به صورت optional تعریف شده، حاوی مقدار هست یا خیر. در واقع مقدار متغیر optional را به عنوان پارامتر به دستور داخل ساختمان if ارسال کنید و در صورت برقرار بودن شرط و داشتن مقدار مورد نظر، آن مقدار را به متغیر داخل ساختمان تخصیص دهید.

```
var optionalName: String? = "John Appleseed"
var greeting = "Hello!"
if let name = optionalName {
  greeting = "Hello, \(name)"
}
```

**امتحان و تجربه کنید**

مقدار متغیر optionalName را به nil تغییر داده و کد را اجرا کنید. چه مقداری را در خروجی دریافت می کنید؟ حال یک دستور else اضافه کنید که در صورت nil بدون مقدار optionalName، مقدار متفاوتی را در greeting قرار دهد.

در صورتی که مقدار nil باشد، شرط غلط بوده و دستور داخل {} اجرا نمی شود. در غیر این صورت مقدار متغیر از نوع optional استخراج (unwrap) شده و به ثابت بعد از let (name) تخصیص داده می شود. در نتیجه مقدار استخراج شده داخل قطعه کد مورد نظر برای استفاده در دسترس قرار می گیرد.

می توانید با تنها یک دستور if همزمان چندین مقدار را bind کنید.

می توان با اضافه کردن دستور where به یک case یا مورد انتخاب، قیود بیشتری اعمال نموده و نتیجه را محدودتر کرد. برای مثال در نمونه ی زیر، دستور if تنها زمانی اجرا می شود که binding تمامی مقادیر با موفقیت انجام شده و تمامی شرایط برآورده شوند.

```
var optionalHello: String? = "Hello"
if let hello = optionalHello where hello.hasPrefix("H"), let name = optionalName
{
    greeting = "\(hello), \(name)"
}
```

ساختار Switch در زبان برنامه نویسی Swift از قدرت و امکانات ویژه ای برخوردار است. یک دستور switch از انواع داده ای و همچنین طیف وسیعی از عملیات مقایسه ای پشتیبانی می کند – تنها به نوع داده ای int و بررسی برابری مقادیر محدود نمی شود. در مثال زیر، دستور switch مقدار متغیر vegetable را با مقادیر case ها، مقایسه کرده و سپس دستور مرتبط با مقدار منطبق را اجرا می کند.

```
let vegetable = "red pepper"
switch vegetable {
case "celery":
    let vegetableComment = "Add some raisins and make ants on a log."
case "cucumber", "watercress":
    let vegetableComment = "That would make a good tea sandwich."
```

```

case let x where x.hasSuffix("pepper"):
let vegetableComment = "Is it a spicy \(x)?"
default:
let vegetableComment = "Everything tastes good in soup."
}

```

امتحان و تجربه کنید

دستور default را حذف کنید. با چه خطایی مواجه می شوید؟

ببینید چگونه از let برای تخصیص مقدار case یا مورد منطبق به یک constant استفاده شده است. مشابه دستور if، می توان با افزودن عبارت شرطی where به یک case، قیود بیشتری اعمال کرده و نتیجه را محدودتر نمود.

برخلاف دستور if، یک case می تواند چندین شرط (که توسط ویرگول از هم جدا شده اند) داشته باشد و حتی اگر یکی از شروط آن صادق باشد، آنگاه دستور مرتبط اجرا خواهد شد. پس از اجرای دستور موجود در case منطبق، برنامه از ساختمان switch خارج می شود و اجرای برنامه به case بعدی ادامه نمی یابد، بنابراین لازم نیست پس از هر case یک دستور break درج نمایید (به صورت صریح از قطعه کد switch خارج شوید).

استفاده از دستور default در بیشتر سناریوها ضروری است. با استفاده از default این اطمینان بدست می آید که در صورت برقرار نبودن شرط هیچ یک از case ها (برابر نبودن مقدار داخل پرانتز با مقدار هیچ یک از case ها)، یک دستور در ساختمان switch اجرا می گردد (با استفاده از این دستور می توانید برای مثال یک پیغام مرتبط به صفحه نمایش ارسال نمایید).

در واقع در ساختار چند انتخابی switch، باید از دستور default استفاده کنید، مگر اینکه مطمئن باشید مقدار داخل پرانتز و مورد مقایسه با حداقل یکی از case ها منطبق است.

می توان با استفاده از یک Range، اندیس را بین دو بازه نگه داشته و در آن پیمایش نمود. جهت ایجاد بازه ای از اندیس ها می توان از عملگر "<.." استفاده نمود.

```
var firstForLoop = 0
for i in 0..<4 {
    firstForLoop += i
}
print(firstForLoop)
```

این عملگر عدد بزرگتر را شامل نمی شود، بنابراین بازه از ۰ شروع شده و تا ۳ ادامه می یابد که در کل ۴ بار دستور اجرا می گردد. می توانید با استفاده از "... " یک بازه ایجاد کنید که مقدار دو طرف بازه را دربرمی گیرد (شامل می شود).

```
var secondForLoop = 0
for _ in 0...4 {
    secondForLoop += 1
}
print(secondForLoop)
```

این بازه از ۰ شروع شده و تا عدد ۴ را دربرمی گیرد که در کل به پنج بار اجرای دستور توسط حلقه منتهی می گردد. علامت " \_ " بیانگر یک wildcard است و شما می توانید از آن زمانی استفاده کنید که نیازی به آگاهی از گام جاری تکرار (در حال اجرا) حلقه ندارید (به عبارت دیگر برای شما مهم نیست حلقه چند بار اجرا شده است و در حال حاضر کدام گام حلقه در حال اجرا است).

### تشریح تفاوت بین function و method

Function یا تابع یک تکه کد نام گذاری شده با قابلیت استفاده ی مجدد است که از بخش های مختلف برنامه می توان آن را فراخوانی کرد. به عبارت دیگر عملیاتی که فکر می کنید چندین بار به آن نیاز پیدا خواهید کرد را در قالب یک تابع تعریف کرده، یک نام به آن تخصیص می دهید. سپس هر بار که به این عملیات نیاز داشتید می توانید آن را در برنامه ی خود فراخوانی کنید (دیگر نیازی نیست آن عملیات را هر بار تعریف کنید).



جهت تعریف یک تابع جدید، کافی است از کلیدواژه `func` استفاده نمایید. یک تابع می تواند چندین پارامتر ورودی داشته باشد که به صورت (نوع پارامتر: اسم پارامتر) `name: Type` داخل پرانتزهای آن نوشته می شوند. پارامترها در واقع اطلاعات اضافی هستند که به هنگام فراخوانی تابع باید به آن ارسال شوند. یک تابع می تواند مقدار بازگشتی یا خروجی داشته باشد که در `Swift` پس از علامت `"->"` ذکر می شود. بدین وسیله مشخص می شود تابع پس از اجرا چه مقداری (از چه نوعی) را به عنوان خروجی تولید می کند. پیاده سازی (implementation) تابع نیز داخل بدنه ی آن `{ }` قرار می گیرد.

```
func greet(name: String, day: String) -> String {
    return "Hello \(name), today is \(day)."
}
```

جهت فراخوانی یک تابع کافی است اسم آن را ذکر نموده و سپس آرگومان هایش را داخل پرانتز (روبه روی) آن مشخص نمایید (آرگومان ها مقادیری هستند که به هنگام فراخوانی تابع به آن ارسال می شود و در جایگاه پارامترهای آن تابع می نشینند). زمانی که شما یک تابع را فراخوانی می کنید، اولین مقدار آرگومان را بدون ذکر اسم آن به داخل پرانتز تابع ارسال می کنید و سپس تمامی مقادیر بعدی را همراه با اسم آن ها مشخص می نمایید.

```
greet("Anna", day: "Tuesday")
greet("Bob", day: "Friday")
greet("Charlie", day: "a nice day")
```

توابعی که داخل نوع خاصی (برای مثال کلاس) تعریف می شوند در اصطلاح **method** یا متد خوانده می شوند. متدها صریحا به نوعی که در آن تعریف می شوند وابسته هستند و تنها در آن نوع (یا یکی از زیرمجموعه ها و **subclass** های آن) قابل فراخوانی می باشند. در دستور **switch** مثال بالا، یک متد به نام **hasSuffix()** را می بینید که بر روی نوع **string** تعریف شده است. این مثال در زیر مجددا عنوان شده است:

```
let exampleString = "hello"
if exampleString.hasSuffix("lo") {
    print("ends in lo")
}
```

همان طور که در مثال مشاهده می کنید، متد با عملگر `"."` (بر روی متغیر) فراخوانی شده است. زمانی که یک متد را صدا می زنید، اولین آرگومان را بدون ذکر اسم آن پاس داده و مقادیر آرگومان بعدی را همراه با

اسم آن ها به متد ارسال می کنید. برای مثال، متد `insert()` که بر روی متغیری به نام `array` از نوع آرایه (`Array`) فراخوانی می شود، دو پارامتر می گیرد ولی شما فقط اسم آرگومان دوم را مشخص می کنید.

```
var array = ["apple", "banana", "dragonfruit"]
array.insert("cherry", atIndex: 2)
array
```

### تشریح مفهوم `class` و `initializer`

در برنامه نویسی شی گرا، رفتار یک برنامه تا حد زیادی به تعامل و تبادل اطلاعات میان آبجکت ها (`object`) بستگی دارد. یک `object` در واقع نمونه ای از یک `class` است. کلاس نیز یک طرح کلی یا الگو برای ساخت آبجکت می باشد. در یک کلاس، آبجکت ها اطلاعات مورد نیاز درباره ی خود را در قالب `property` ها ذخیره کرده و رفتار خود را با استفاده از متدها تعریف می کنند.

جهت تعریف یک کلاس جدید، ابتدا کلیدواژه ی `class` و سپس اسم دلخواه را تایپ نمایید. یک `property` درست مانند ثابت یا متغیر تعریف می شود، با این تفاوت که تعریف آن در سطح (`context`) کلاس انجام می شود (`property` داخل کلاس تعریف می شود).

تعریف متد و تابع نیز به همان صورت انجام می شود (متد در بستر کلاس تعریف می شود، اما تابع چنین نیست). مثال زیر یک کلاس به نام `shape`، یک `property` به نام `numberOfSides` و یک متد به نام `simpleDescription()` تعریف می کند.

```
class Shape {
var numberOfSides = 0
func simpleDescription() -> String {
return "A shape with \(numberOfSides) sides."
}
}
```

به منظور ایجاد نمونه ای (یا آبجکت) از یک کلاس، کافی است یک پرانتز باز و بسته بعد از اسم کلاس درج نمایید. حال جهت دسترسی به `property` ها/`method` های نمونه ی ایجاد شده از کلاس، اسم نمونه ی کلاس و به دنبال آن عملگر نقطه و در پایان اسم

property/method دلخواه را تایپ نمایید. در این مثال، shape یک نمونه یا آبجکت از کلاس Shape است که با پیروی از ساختار نگارشی ذکر شده به اعضای آن (متد و پراپرتی) دسترسی پیدا می کنید.

```
var shape = Shape()
shape.numberOfSides = 7
var shapeDescription = shape.simpleDescription()
```

این کلاس به یک عضو دیگر نیاز دارد و آن initializer است. initializer یک متد است که نمونه ای از یک کلاس را برای استفاده آماده ساخته، تک تک property های آن را مقداردهی اولیه می کند و امکان هرگونه تنظیم دیگری را فراهم می آورد. برای ایجاد initializer، کافی است از کلیدواژه ی init استفاده نمایید. این مثال یک کلاس جدید به نام NamedShape تعریف کرده، سپس با استفاده از initializer خود (متد سازنده یا مقدار دهنده ی اولیه متغیرهای کلاس)، یک property به نام name را مقداردهی اولیه می کند.

```
class NamedShape {
    var numberOfSides = 0
    var name: String
    init(name: String) {
        self.name = name
    }
    func simpleDescription() -> String {
        return "A shape with \(numberOfSides) sides."
    }
}
```

با استفاده از کلیدواژه ی self (معادل this در C#)، در حقیقت عملاً بین (property) متغیر name و آرگومان ارسالی به همان نام (آرگومان name) به initializer تمایز قائل می شوید. هر property ای که تعریف می کنید، باید مقداردهی نمایید (حال این مقدار دهی یا در تعریف آن property انجام می شود مانند numberOfSides و یا داخل initializer مانند name).

برای فراخوانی یک `initializer`، ذکر کلیدواژه `init` نامصحیح است. روش صحیح آن عبارت است از ذکر اسم کلاس و سپس درج پرانتز باز، آرگومان های مورد نظر و در نهایت پرانتز بسته. زمانی که شما یک `initializer` را صدا می زنید، تمامی آرگومان ها را همراه با اسمشان به آن (`initializer`) پاس می دهید.

```
let namedShape = NamedShape(name: "my named shape")
```

کلاس ها رفتار خود را از کلاس والد/`parent` به ارث می برند. کلاسی که رفتارش را از کلاس دیگری به ارث می برد، فرزند/زیرمجموعه یا `subclass` آن کلاس شناخته می شود. کلاسی هم که رفتارش را به کلاس دیگری به ارث می دهد، کلاس والد/ارشد یا `superclass` خوانده می شود. در ارث بری، ابتدا اسم کلاس فرزند و پس از دو نقطه، اسم کلاس والد ذکر می شود. به عبارت دیگر جهت ارث بری، ابتدا اسم کلاس `subclass`، سپس دو نقطه ":" و در پایان اسم `superclass` درج می شود. یک کلاس (`subclass`) تنها می تواند از یک `superclass` ارث بری کند و متعاقباً آن `superclass` خود می تواند از یک `superclass` دیگر ارث بری داشته باشد که از آن به عنوان `class` hierarchy (زنجیره ی ارث بری) یاد می شود.

متدهایی که در یک `subclass`، پیاده سازی متد کلاس `superclass` را بازنویسی یا `override` می کنند، با کلیدواژه ی `override` نشانه گذاری می شوند. اگر بدون استفاده از کلیدواژه ی `override`، سعی کنید بدنه یا پیاده سازی متد `superclass` را بازنویسی نمایید، `compiler` بلافاصله ایراد می گیرد و آن را به عنوان خطا تشخیص می دهد. کامپایلر همچنین متدهایی که با `override` علامت گذاری می شوند اما به معنای واقعی بازنویسی در آن ها صورت نمی گیرد را به راحتی شناسایی می کند.

مثال زیر یک کلاس به نام `Square` را تعریف می کند که از `NamedShape` ارث بری دارد (کلاس زیرمجموعه ای از `NamedShape` می باشد).

```

class Square: NamedShape {
var sideLength: Double
init(sideLength: Double, name: String) {
self.sideLength = sideLength
super.init(name: name)
numberOfSides = 4
}
func area() -> Double {
return sideLength * sideLength
}
override func simpleDescription() -> String {
return "A square with sides of length \$(sideLength)."
}
}
let testSquare = Square(sideLength: 5.2, name: "my test square")
testSquare.area()
testSquare.simpleDescription()

```

همان طور که می بینید متد initializer از کلاس Square، سه کار زیر را انجام می دهد:

۱. مقداردهی property هایی که در کلاس Square تعریف شده است.
  ۲. فراخوانی متد initializer از کلاس NamedShape.
  ۳. ویرایش مقدار property های تعریف شده در کلاس (والد) NamedShape.
- تمامی تنظیمات لازم هم که از متدها، getter ها و setter ها استفاده می کنند، باید در این مرحله انجام شود.

گاهی مقداردهی اولیه ی یک آبجکت باید (یا ممکن است) با شکست مواجه شود، مانند زمانی که مقادیر ارائه شده به عنوان آرگومان از یک بازه ی مشخص خارج هستند یا هنگامی که داده های مورد انتظار (اطلاعاتی که باید وجود داشته باشند) در دسترس نیستند. Initializer هایی که در مقداردهی اولیه ی یک آبجکت با شکست مواجه می شوند را در اصطلاح failable initializer می نامند. failable initializer پس از (شکست

در) عملیات مقداردهی اولیه می تواند nil بازگردانی نماید. جهت اعلان یک failable initializer از init? استفاده نمایید:

```
class Circle: NamedShape {
    var radius: Double
    init?(radius: Double, name: String) {
        self.radius = radius
        super.init(name: name)
        numberOfSides = 1
        if radius <= 0 {
            return nil
        }
    }
    override func simpleDescription() -> String {
        return "A circle with a radius of \(radius)."
    }
}

let successfulCircle = Circle(radius: 4.2, name: "successful circle")
let failedCircle = Circle(radius: -7, name: "failed circle")
```

initializer ها نیز کلیدواژه های اختصاصی خود را دارند. یک designated initializer به هیچ کلیدواژه ای نیاز ندارد. designated initializer همان متد init اصلی و اولیه ی کلاس است که می تواند property های غیر optional و فاقد مقدار را مقداردهی اولیه کند). متد نام برده به عنوان یکی از initializer های اولیه و اصلی کلاس ایفای نقش می کنند. تمامی initializer های داخل یک کلاس در نهایت بایستی designated initializer را صدا بزنند.

کلیدواژه ی convenience (در کنار کلمه ی کلیدی init) نشانگر initializer های ثانوی است که برای افزودن رفتار جدید یا سفارشی سازی بکار می روند. convenience initializer ها نیز در نهایت می بایست designated initializer اصلی و اولیه ی کلاس را صدا بزنند.

کلیدواژه ی required اعلان می کند که تمامی کلاس هایی که از کلاس والد ارث بری می کنند و initializer آن را دارند، باید نسخه ی اختصاصی خود از initializer مورد نظر را پیاده سازی کنند (البته در صورتی که کلاس مورد نظر متد init را پیاده سازی کند).

Type casting (تبدیل نوع) روشی است برای بررسی نوع یک آبجکت (یا نمونه از کلاس) و برخورد با آن شی گویی خود یک کلاس والد مجزا است و یا یک کلاس فرزند جای دیگری در زنجیره ی ارث بری (class hierarchy) خود است.

یک ثابت یا متغیر از کلاسی با نوع معین، ممکن است در اصل به یک نمونه از کلاس فرزند اشاره داشته باشد (مقدارش را از آن گرفته باشد). در جایی که فکر می کنید این امر

صدق می کند، می توانید با استفاده از عملگر تبدیل نوع، به کلاس فرزند downcast انجام دهید. به عبارت دیگر، از نمونه کلاس والد به کلاس فرزند تبدیل نوع انجام دهید (downcast: تبدیل آبجکتی به یک نوع از کلاس های فرزند آن. Downcast زمانی امکان پذیر است که متغیری از کلاس والد، مقداری را از کلاس فرزند می گیرد).

از آنجایی که احتمال شکست فرایند downcast وجود دارد، عملگر تبدیل به دو صورت کلی ارائه می شود. شکل optional آن، as?، یک مقدار optional از نوعی که می خواهید به آن تبدیل نوع انجام دهید، در خروجی برمی گرداند. شکل forced آن، as!، سعی می کند downcast را انجام داده و نتیجه را همزمان force-unwrap کند (محتوای نتیجه را به زور استخراج کند).

از "as?" زمانی استفاده کنید که از انجام موفقیت آمیز عملیات downcast اطمینان ندارید. این عملگر در خروجی همیشه مقداری از نوع optional را برمی گرداند. چنانچه downcast امکان پذیر نباشد، مقدار خروجی nil خواهد بود. بدین وسیله شما می توانید بررسی کنید آیا downcast موفقیت آمیز خواهد بود یا خیر.

از عملگر "as!" تنها زمانی استفاده کنید که از اجرای موفقیت آمیز عملیات downcast اطمینان کامل دارید. چنانچه سعی کنید با این عملگر تبدیل نوع به کلاس (class type) نامربوط و نامنتطبق را انجام دهید، با خطای زمان اجرا (runtime) مواجه خواهید شد.



این مثال سعی می کند با بهره گیری از عملگر `as?` بررسی کند آیا یک `shape` (متغیر `shape`) در آرایه ای از `shape` ها از نوع `square` است یا `triangle`. سپس هر بار که حلقه در گام تکرار خود با `shape` یا مورد منطبق برخورد می کند، مقدار متغیرهای `squares` و `triangles` را یک واحد اضافه می کند و در پایان مقادیر مربوطه را از طریق دستور `print` چاپ می نماید.

```
class Triangle: NamedShape {
    init(sideLength: Double, name: String) {
        super.init(name: name)
        numberOfSides = 3
    }
}

let shapesArray = [Triangle(sideLength: 1.5, name: "triangle1"),
    Triangle(sideLength: 4.2, name: "triangle2"), Square(sideLength: 3.2, name:
    "square1"), Square(sideLength: 2.7, name: "square2")]
var squares = 0
var triangles = 0
for shape in shapesArray {
    if let square = shape as? Square {
        squares++
    } else if let triangle = shape as? Triangle {
        triangles++
    }
}
print("\(squares) squares and \(triangles) triangles.")
```

**امتحان و تجربه کنید**

در کد بالا عملگر `as?` را با `as!` جایگزین کنید. چه خطایی رخ می دهد؟

**تشریح مفهوم Enumeration ها و Structure ها در Swift**

کلاس ها تنها راه های تعریف انواع داده ای در زبان شی گرای Swift نیستند (تنها بسترهای تعریف متغیر، `property` و متد نیستند). `Enumeration` ها و `Structure` ها

کاربرد و قابلیت های مشابه به کلاس را دارند، اما در شرایط متفاوتی به کار گرفته می شوند.

Enumerations یک نوع مشترک برای گروهی از مقادیر مرتبط تعریف کرده و به شما امکان می دهد در کد خود با این مقادیر به صورت type-safe کار کنید. (swift یک زبان است که جانب ایمنی نوع یا type safety را رعایت می کند؛ بدین معنی که اگر بخشی از کد برنامه ی شما انتظار مقداری از جنس رشته را داشته باشد، ویژگی type safety مانع از این می شود که به صورت تصادفی مقداری از نوع عدد صحیح را به آن ارسال کنید).

برای مثال می توان به وضعیت اتصال شبکه اشاره کرد که در آن چهار حالت unknown، connected، connecting، disconnected ممکن است. در این سناریو یک enum از نوع int تعریف می شود که مقدار خام آن از ۱ شروع می شود و به همین ترتیب ادامه می یابد.

Enumeration ها در Swift، متد نیز می توانند داشته باشند. جهت تعریف Enumeration کافی است از کلیدواژه ی enum استفاده نمایید:

```
enum Rank: Int {
    case Ace = 1
    case Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten
    case Jack, Queen, King
    func simpleDescription() -> String {
        switch self {
            case .Ace:
                return "ace"
            case .Jack:
                return "jack"
            case .Queen:
                return "queen"
            case .King:
                return "king"
            default:
                return String(self.rawValue)
        }
    }
}
```

```

}
let ace = Rank.Ace
let aceRawValue = ace.rawValue

```

در مثال بالا نوع مقدار خام (مقدار از نوع رشته، عدد صحیح، کاراکتر) و تخصیص داده شده به ساختار enumeration از جنس int است، بنابراین بایستی فقط مقدار اولین عضو را مشخص نمایید. دیگر اعضای این ساختار به ترتیب مقداردهی می شوند (برای مثال اگر مقدار اولین عضو را ۱ بدهید، مقادیر دیگر اعضا به ترتیب ۲، ۳ و ۴ خواهند بود).

نوع enum را می توان از جنس رشته یا عددی ممیز شناور (floating-point) نیز تعریف کرد.

جهت دسترسی به مقدار خام و اولیه ی اعضای ساختار enum، کافی است از property ای به نام rawValue استفاده نمایید.

می توان با استفاده از مقدار یکی از اعضای enum یک نمونه از آن ایجاد کرد. برای ساختن نمونه از enum از روی مقدار یکی از اعضای آن، می بایست متد (rawValue:?) init را بکار ببرید.

```

if let convertedRank = Rank(rawValue: 3) {
let threeDescription = convertedRank.simpleDescription()
}

```

مقادیر عضو یک ساختار enum، مقادیر حقیقی هستند، نه صرفاً روشی دیگر برای نوشتن مقادیر خام/raw value (از نوع اولیه همچون رشته، عدد صحیح و غیره...) آن ها. در حقیقت، در مواردی که مقدار خام معناداری وجود ندارد، شما هم ملزوم به ارائه ی مقدار نیستید.

```

enum Suit {
case Spades, Hearts, Diamonds, Clubs
func simpleDescription() -> String {

```

```

switch self {
case .Spades:
return "spades"
case .Hearts:
return "hearts"
case .Diamonds:
return "diamonds"
case .Clubs:
return "clubs"
}
}
}
let hearts = Suit.Hearts
let heartsDescription = hearts.simpleDescription()

```

در مثال بالا عضوی به نام Hearts به دو روش مختلف مورد اشاره قرار گرفته است: زمانی که به ثابت hearts مقداری اختصاص داده شده، عضو Suit.Hearts، همان طور که می بینید، با نام کاملش مورد اشاره قرار گرفته چرا که نوع ثابت (constant) مورد نظر به صورت صریح مشخص نشده است. اما داخل ساختمان switch، عضو مزبور enumeration با شکل مختصر آن مورد اشاره قرار گرفته زیرا نوع self مشخصاً از جنس suit است (از قبل می دانیم نوع آن suit است).

هرگاه که نوع مقدار از قبل مشخص باشد، می توانید (برای اشاره به آن) از شکل مختصر اسم آن عضو استفاده کنید.

Structure ها بسیاری از رفتارها و امکانات کلاس نظیر متد و initializer را پشتیبانی می کنند. یکی از تفاوت های عمده ی structure با کلاس در این است که به هنگام ارسال structure به بخش های مختلف کد، یک کپی از آن به کد مورد نظر فرستاده می شود. این در حالی است که کلاس با ارجاع یا reference به کد پاس داده می شود (بدین معنی که هرگونه تغییر در کلاس توسط کد، بر روی کلاس اصلی نیز اعمال می شود).

structure ها برای تعریف انواع داده ای سبک که به قابلیت هایی همچون ارث بری و تبدیل نوع نیاز ندارند، بسیار مناسب می باشد. جهت تعریف یک structure جدید کافی است از کلیدواژه ی struct استفاده نمایید:

```
struct Card {
    var rank: Rank
    var suit: Suit
    func simpleDescription() -> String {
        return "The \(rank.simpleDescription()) of \(suit.simpleDescription())"
    }
}

let threeOfSpades = Card(rank: .Three, suit: .Spades)
let threeOfSpadesDescription = threeOfSpades.simpleDescription()
```

شرح مفهوم Protocol (الگوی پیاده سازی)

Protocol یک الگو برای پیاده سازی متدها، property ها و دیگر ملزومات مناسب برای انجام کار یا پیاده سازی قابلیت خاص می باشد. لازم به ذکر است که protocol هیچ پیاده سازی در خصوص متدها، property ها و غیره ... ارائه نمی دهد، بلکه تنها نحوه ی پیاده سازی ملزومات نام برده را تعیین می کند. می توان گفت protocol همان interface در objective-c است.

این الگو سپس برای پیاده سازی اعضای (متدها، پراپرتی ها و غیره ..) یک class، enumeration یا structure بکار گرفته می شود. هر یک از ساختارهای ذکر شده (class و enumeration یا structure) که الگو یا protocol را پیاده سازی کنند، در اصطلاح به آن conform (از آن الگو پیروی) می کنند.

برای تعریف یک protocol، از کلیدواژه ی protocol استفاده می کنیم:

```
protocol ExampleProtocol {
    var simpleDescription: String { get }
```

```
func adjust()
```

```
}
```

**نکته:**

دستور { get }، پس از متغیر simpleDescription در کد بالا، بیانگر این است که مقدار متغیر نام برده، فقط خواندنی است؛ بدین معنی که مقدارش را می توان خواند اما امکان تغییر در آن وجود ندارد.

Protocol ها می توانند ایجاب کنند که ساختارهایی که از آن ها پیروی می کنند (پیاده سازی اعضای خود را بر اساس آن الگو انجام می دهند) دارای property instance، instance method، type method، operator و subscript های خاصی باشند. در بالا نیز گفته شد که protocol یک الگو یا معرفی یک سری متد و property است. اگر برای آبجکت یا کلاسی protocol اعلان کنید، آن کلاس یا آبجکت ملزم به پیاده سازی متدها و property های تعیین شده توسط آن protocol می شود.

Protocol ها همچنین می توانند instance method ها و type method هایی که توسط ساختار مورد نظر پیاده سازی می شوند را تعیین کنند (به عبارت دیگر protocol می تواند ساختار تبعیت کننده را مجاب به پیاده سازی type method ها و instance method های خاصی بکند). این متدها به عنوان بخشی از تعریف protocol و بدون {} یا بدنه مشخص می شوند. Type method: متدهایی که در ساختار خاصی (class، enum یا structure) تعریف می شوند و به آن وابسته و مرتبط هستند. Instance method: متدهایی که به نمونه ی ایجاد شده از ساختار مورد نظر تعلق دارند و تمامی قابلیت های آن ساختار را پشتیبانی می کنند.

Class ها، structure ها و enumeration ها برای پیروی از protocol یا الگوی پیاده سازی خاص، اسم آن را پس از اسم خود و دو نقطه ذکر می کنند. هر ساختار می تواند از بی نهایت protocol پیروی کند. اسم protocol ها به صورت یک لیست تفکیک شده با ویرگول، پس از اسم ساختار مورد نظر عنوان می شوند.

چنانچه یک کلاس از کلاس والدی ارث بری کند، در آن صورت اسم کلاس والد باید پیش از اسم protocol ها لیست شود.

همان طور که می دانید یک ساختار زمانی کاملاً از protocol معین پیروی می کند که تمامی ملزومات و اعضای تعیین شده در آن نظیر متدها، property ها و غیره را پیاده سازی کند.

در زیر، کلاس SimpleClass اعضای خود را براساس protocol یا الگوی ExampleProtocol پیاده سازی می کند (اعضای تعریف شده در الگوی نام برده، property ای به نام simpleDescription و متد adjust() را پیاده سازی می کند).

```
class SimpleClass: ExampleProtocol {
var simpleDescription: String = "A very simple class."
var anotherProperty: Int = 69105
func adjust() {
simpleDescription += " Now 100% adjusted."
}
}
var a = SimpleClass()
a.adjust()
let aDescription = a.simpleDescription
```

protocol ها نیز مانند دیگر نوع ها می توانند به عنوان آرگومان ارسال شوند، خروجی یک تابع باشند یا به متغیر معینی تخصیص داده شوند (به عبارت دیگر، protocol ها first class type هستند و با آن ها مانند دیگر نوع های نام گذاری شده همچون آرایه، رشته و غیره ... برخورد می شود).

برای مثال می توانید یک آرایه ExampleProtocol تعریف کنید و سپس تابع adjust() را برای تک تک نمونه های داخل آن فراخوانی نمایید (تمامی نمونه های داخل آن آرایه متد نام برده را که توسط protocol تعریف شده است، پیاده سازی می کنند).



```

class SimpleClass2: ExampleProtocol {
var simpleDescription: String = "Another very simple class."
func adjust() {
simpleDescription += " Adjusted."
}
}
var protocolArray: [ExampleProtocol] = [SimpleClass(), SimpleClass(),
SimpleClass2()]
for instance in protocolArray {
instance.adjust()
}
protocolArray

```

### Swift و فریم ورک طراحی رابط کاربری Cocoa Touch

طراحی زبان Swift به گونه ای است که به راحتی و بی درد سر می توان از آن همراه با Cocoa Touch استفاده کرد. به عبارت دیگر این زبان شی گرا طوری تعبیه شده که به راحتی می تواند با Cocoa Touch همکاری داشته باشد. Cocoa Touch یک سری framework جهت طراحی اپلیکیشن برای سیستم عامل iOS است. با پیشرفت در مباحث بعدی این سری آموزش، داشتن فهم پایه ای از نحوه ی تعامل Swift با Cocoa Touch، کمک شایانی به شما در یادگیری برنامه نویسی برای iOS و توسعه ی اپلیکیشن می کند.

تا به اینجا ، منحصر از نوع داده هایی که Swift standard library ارائه می دهد، در مثال های عملی خود استفاده کردید. Swift standard library یا کتابخانه ی متعارف زبان Swift، متشکل است از تعدادی نوع داده ای و امکانات اختصاصی که به صورت درون ساخته در این زبان برنامه نویسی تعبیه شده است. نوع های همچون آرایه/Array و رشته/String نمونه هایی از انواع داده ای هستند که در کتابخانه ی متعارف Swift یافت می شود.

```
let sampleString: String = "hello"
let sampleArray: Array = [1, 2, 3.1415, 23, 42]
```

### امتحان و تجربه کنید

با Option-click بر روی نوع مورد نظر در محیط Xcode، می توانید اطلاعات مربوط به انواع داده ای موجود در کتابخانه ی Swift را مشاهده نمایید. بر روی String و Array در کد بالا، داخل محیط Xcode، Option-click کنید. چه اطلاعاتی به نمایش در می آید؟

در طول طراحی اپلیکیشن برای سیستم عامل iOS، از کتابخانه ی متعارف Swift بیشتر استفاده خواهید کرد.

یکی از framework های پرکاربرد در طراحی اپلیکیشن برای iOS، کتابخانه ی UIKit است. این framework کلاس های متعددی را برای کار با لایه ی UI و طراحی رابط کاربری برنامه ارائه می دهد.

جهت دسترسی و استفاده از کلاس های UIKit، کافی است آن را به صورت یک ماژول داخل فایل playground یا فایل Swift دلوخواه وارد/import نمایید.

```
import UIKit
```

پس از وارد کردن UIKit، می توانید انواع داده ای مشخص شده در این framework را همراه با متدها، property های متناظر آن بکار ببرید.

```
let redSquare = UIView(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
redSquare.backgroundColor = UIColor.redColor()
```

بیشتر کلاس هایی که در مثال های عملی این سری آموزشی با آن ها برخورد می کنید از کتابخانه ی UIKit برگرفته می شوند، بنابراین دستور import را به کرات مشاهده خواهید کرد. با کسب این میزان دانش از Swift شما آماده اید که در مبحث بعدی یک اپلیکیشن کامل و کاربردی را طراحی و پیاده سازی کنید.

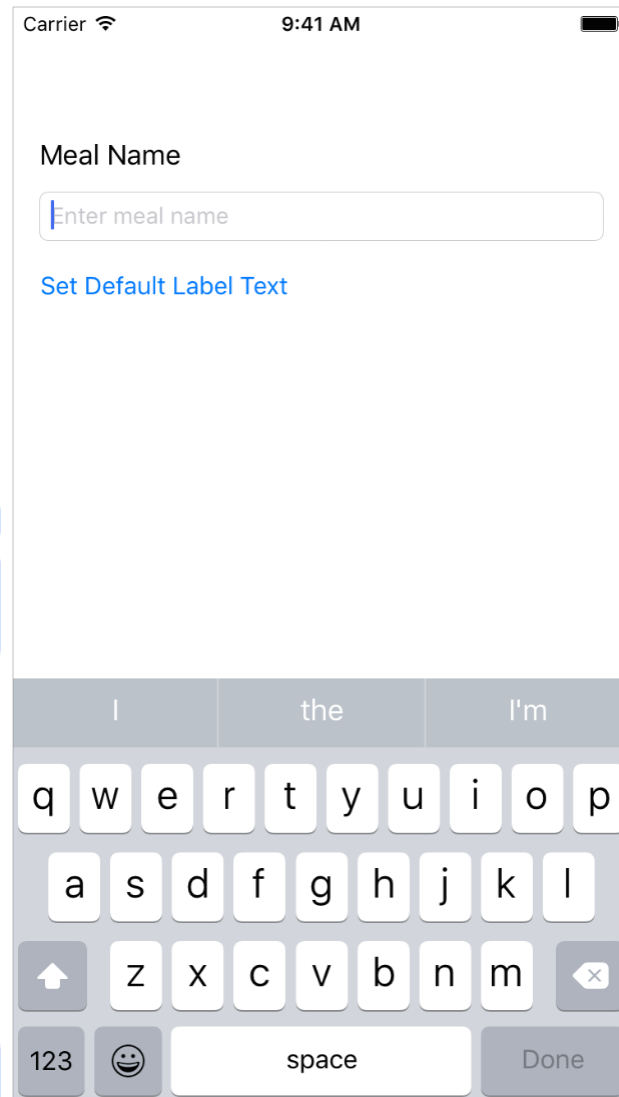
در این مبحث از تنها بخشی از کل قابلیت های فایل playground استفاده شد. ناگفته نماند که فایل های playground کاربرد وسیعی دارند که از جمله می توان به خطایابی، به تصویر کشیدن کدهای پیچیده و ساخت نمونه های اولیه از اپلیکیشن اشاره کرد.

### درس ۳ : ساخت رابط کاربری در Swift

#### ساخت یک رابط کاربری UI/ساده

این مبحث شما را با Xcode، محیط برنامه نویسی و ساخت اپلیکیشن برای سیستم عامل iOS، آشنا می سازد. در این مبحث همچنین با ساختار پروژه در Xcode آشنا شده و نحوه ی پیمایش بین کامپوننت ها و استفاده از آن ها را خواهید آموخت. از ابتدا تا انتهای مبحث، یک رابط کاربری ساده (UI) برای اپلیکیشن FoodTracker طراحی کرده و سپس آن را در محیط شبیه ساز (Simulator) اجرا و مشاهده می کنید. در پایان برنامه ظاهری مشابه زیر خواهد داشت:

آموزشگاه تحلیک داده



آنچه خواهید آموخت

- یک پروژه جدید در محیط کاری Xcode ایجاد نمایید.
- کاربرد فایل های اصلی که به همراه الگو یا قالب پروژه (project template) ایجاد می شوند را بدانید.
- فایل های پروژه را باز کرده و بین آن ها راه گزینی (switch) کنید.
- یک اپلیکیشن را در محیط شبیه ساز (simulator) اجرا نمایید.
- در سطح storyboard المان های UI جدید اضافه کرده، آن ها را جابجا و تغییر اندازه دهید.

- با استفاده از inspector، Attribute، attribute های المان های UI را در storyboard ویرایش نمایید.
- المان های UI را با استفاده از outline view مشاهده و در صورت لزوم مجدداً سازمان دهی نمایید.
- با استفاده از Preview assistant editor، پیش نمایشی از یک storyboard UI (پیش نمایشی از طرح کلی و ظاهر برنامه) داشته باشید.
- UI طراحی کنید که به وسیله ی امکان Auto Layout خود را با اندازه ی دستگاه میزبان (کاربر) تنظیم می کند.

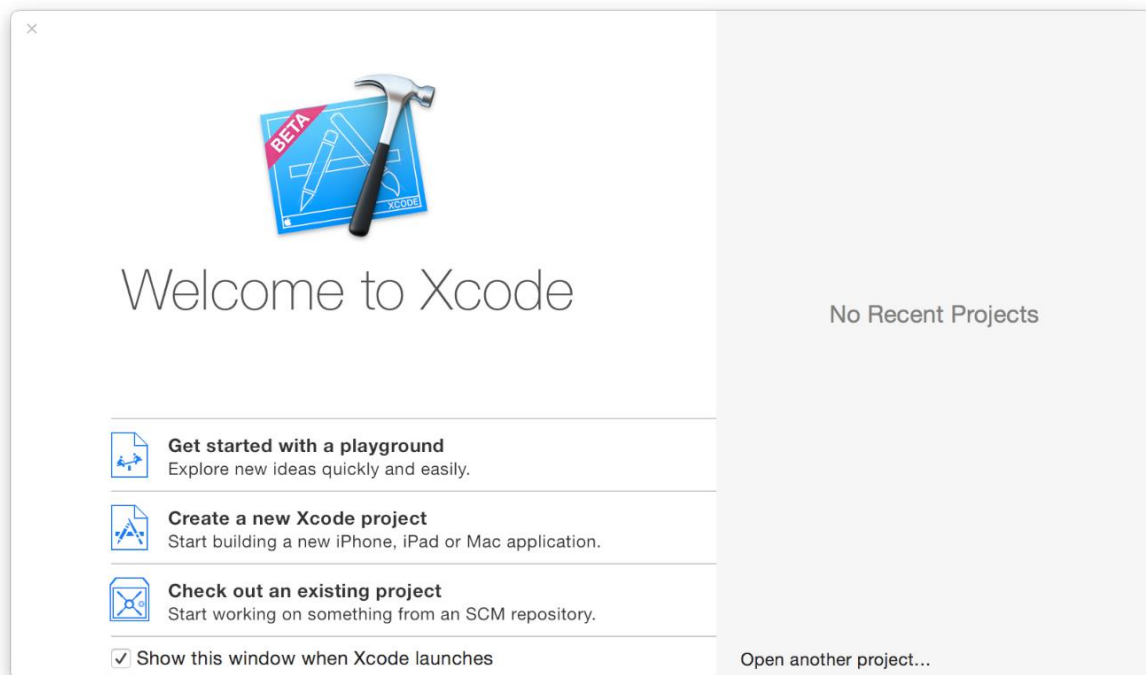
#### ایجاد پروژه ی جدید

Xcode الگو یا template های درون ساخته ی متعددی دارد که ساخت و توسعه ی اپلیکیشن های متعارف IOS همچون بازی ها، برنامه هایی با قابلیت پیمایش با تب (tab-based navigation) و همچنین اپلیکیشن های با نمای جدولی (table-view-based) را آسان می سازد. بیشتر این template ها دارای interface و فایل های source code از پیش تنظیم شده هستند و شما می توانید از آن ها به صورت آماده استفاده کنید.

در آموزش حاضر کار را با ساده ترین نوع template آغاز می کنیم: Single View Application.

برای ایجاد پروژه ی جدید:

۱. Xcode را از پوشه ی Applications / باز کنید. پنجره ی زیر به نمایش در می آید.



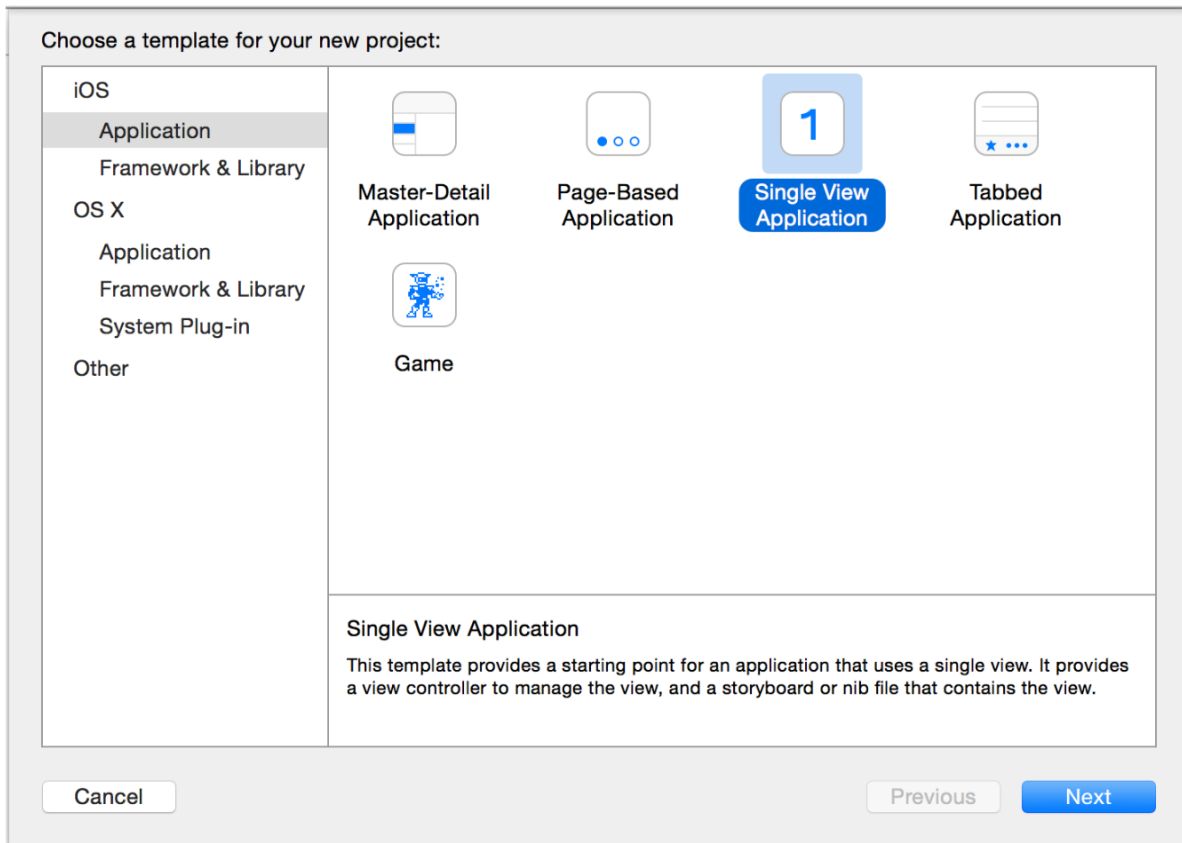
ممکن است بجای پنجره ی فوق، پنجره ی پروژه نمایش داده شود. جای نگرانی نیست، احتمالاً از قبل یک پروژه در محیط Xcode تعریف کرده یا فایل یک پروژه ی آماده را باز کرده اید. کافی است بر روی آیتم مربوطه در منو برای ایجاد پروژه ی جدید کلیک نمایید.

۲. در پنجره ی آغازین (welcome window)، بر روی گزینه ی “Create a new Xcode project” کلیک کرده و یا این گام ها را دنبال نمایید: **File > New > Project**

Xcode یک پنجره ی جدید باز کرده و پنجره ی محاوره ای را نمایش می دهد که می توان در آن template یا الگوی پروژه را انتخاب نمود.

۳. در کادر کناری سمت چپ پنجره، گزینه ی Application را از بخش مربوط به iOS انتخاب نمایید.

۴. داخل ناحیه ی اصلی پنجره ی محاوره ای، آیکون Single View Application را انتخاب کرده و سپس دکمه ی Next را کلیک نمایید.



۵. پنجره ی محاوره ای دیگر پدیدار می شود. مقادیر مشخص شده در زیر را

وارد فیلدهای آن نموده و تنظیمات لازم را انجام دهید.

- **Product Name:** FoodTracker  
همان طور که از اسم آن پیدا است، Xcode از این فیلد برای نام گذاری پروژه و اپلیکیشن شما بهره می گیرد.

- **Organization Name:** اسم شرکت تولید نرم افزار یا اسم خودتان را در این فیلد وارد نمایید. می توانید فیلد نام برده را خالی بگذارید.

- **Organization Identifier:** شناسه ی شرکت خود را وارد نمایید. در اینجا com.example را به عنوان مثال وارد می کنیم.

- **Language:** Swift

- **Devices:** Universal

اپلیکیشن Universal قابلیت اجرا بر روی iPhone و iPad را دارا می باشد.

- **Bundle identifier**: این مقدار به صورت خودکار براساس **organization** و **product name** تولید می شود.
- **Use Core Data: Unselected**
- **Include Unit Tests: Selected**
- **Include UI Tests: Unselected**

Choose options for your new project:

Product Name: FoodTracker

Organization Name: Apple Inc.

Organization Identifier: com.example

Bundle Identifier: com.example.FoodTracker

Language: Swift

Devices: Universal

☐ Use Core Data

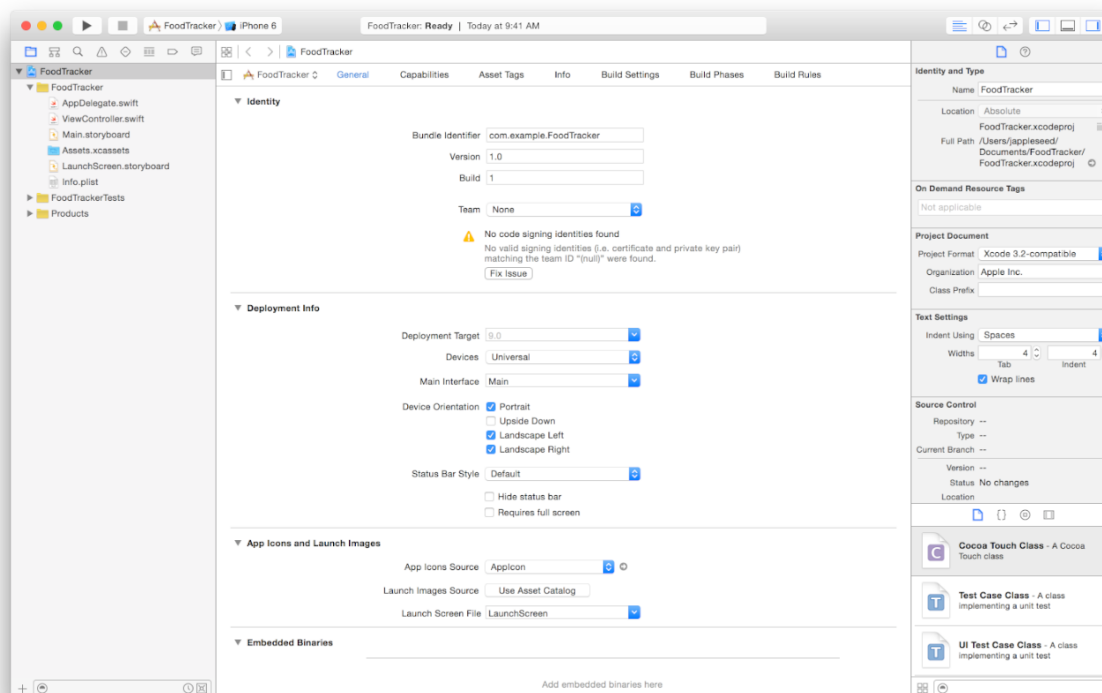
☒ Include Unit Tests

☐ Include UI Tests

Cancel Previous Next

۶. بر روی دکمه ی **Next** کلیک کنید.
۷. در پنجره ی محاوره ای که به نمایش در می آید، آدرس یا محل ذخیره فایل های پروژه ی خود را انتخاب کرده و بر روی **Create** کلیک نمایید. **Xcode**، پروژه ی جدید را داخل **workspace window** باز می کند (workspace = پنجره ی داخل محیط کاری **Xcode** که امکان مدیریت فایل ها و منابع پروژه و پیمایش در آن ها را برای شما فراهم می کند).





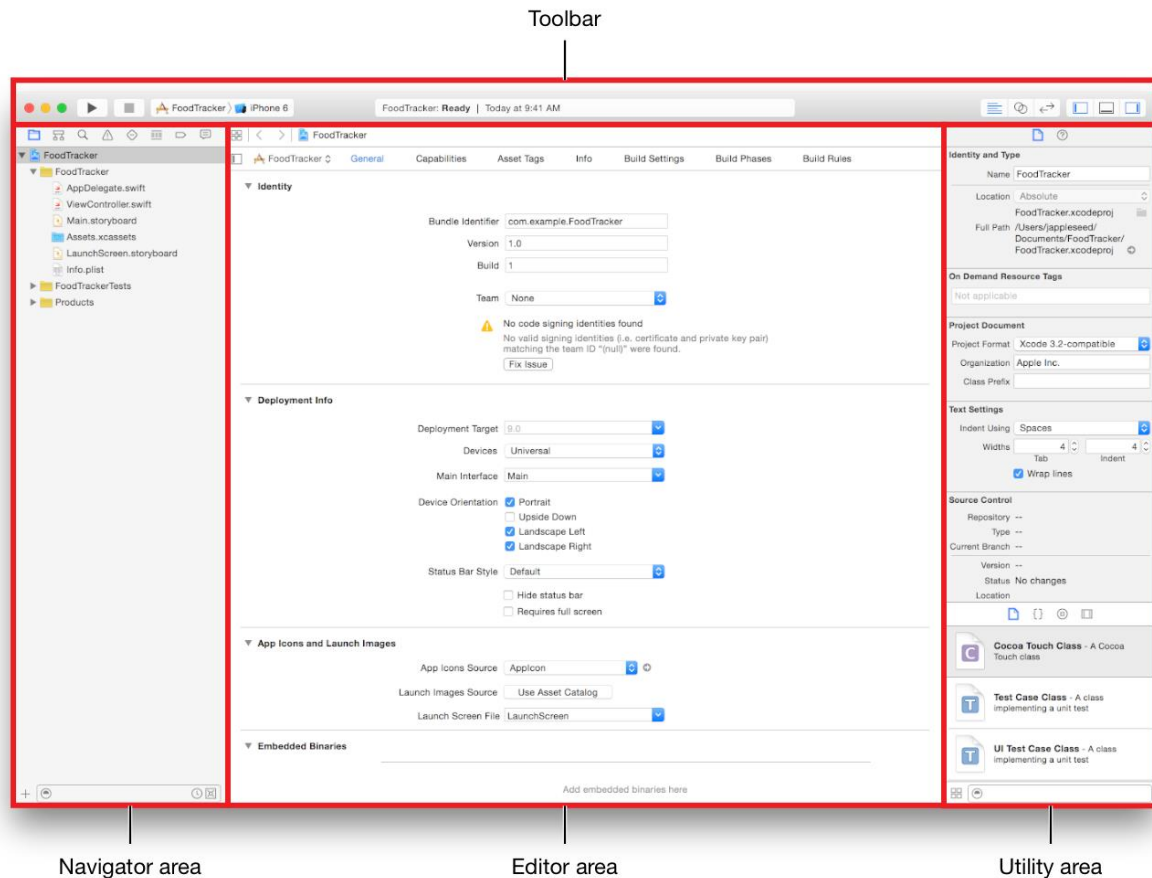
در پنجره‌ی Workspace، ممکن است پیغام هشدار “No code signing identities found.” را مشاهده نمایید. پیغام ذکر شده بیانگر این است که محیط Xcode را برای توسعه اپلیکیشن و برنامه نویسی iOS تنظیم نکرده اید. اما جای هیچ نگرانی نیست، شما می توانید بدون انجام این تنظیمات، پروژه را کامل کنید.

### آشنایی بیشتر با محیط کاری Xcode

هر آنچه برای ساخت یک اپلیکیشن کارآمد به آن نیاز دارید را در محیط Xcode خواهید یافت. این محیط نه تنها فایل های تشکیل دهنده ی پروژه ی را سازمان دهی می کند، ابزارهایی جهت ویرایش کد و المان های UI فراهم نموده و به شما امکان می دهد اپلیکیشن خود را کامپایل و اجرا نمایید. جهت خطایابی و اشکال زدایی، یک debugger درون ساخته در اختیار شما قرار می دهد که با استفاده از آن می توانید خط به خط برنامه را بررسی و debug کنید.

توصیه می کنیم زمانی را به آشنایی با بخش های مختلف پنجره ی Xcode تخصیص دهید. چرا که بخش های نام گذاری شده در پنجره ی زیر را در سرتاسر پروژه های این

سری آموزشی بکار خواهید برد. در نگاه اول ممکن است کمی گیج کننده به نظر برسد، اما لازم نیست نگران باشید. به هنگام استفاده، هر بخش با جزئیات بیشتری شرح داده خواهد شد.



### راه اندازی محیط شبیه ساز (Simulator)

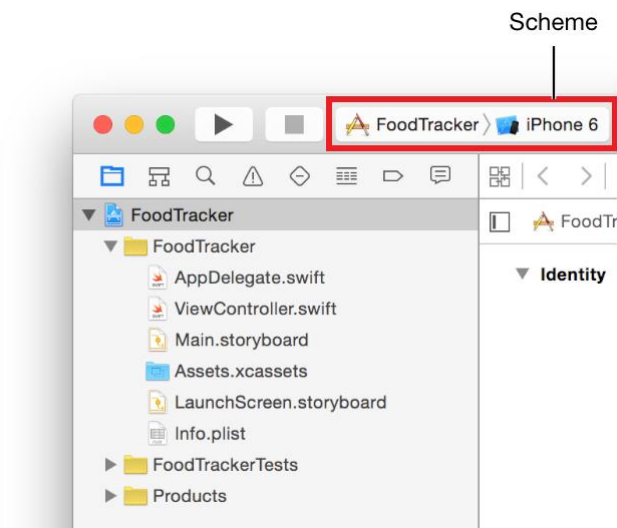
از آنجایی که پروژه ی خود را با استفاده از یک template در Xcode ایجاد کردید (پروژه ی خود را بر مبنای template آماده پی ریزی کرده اید)، محیط اولیه ی اپلیکیشن به صورت خودکار برای شما تنظیم می شود. با اینکه هیچ کدی برای برنامه خود ننوشته اید، می توانید قالب آماده ی Single View Application را بدون هیچگونه تنظیمات اضافی کامپایل (build) و اجرا نمایید.

جهت کامپایل و اجرای اپلیکیشن، می توانید از اپلیکیشن Simulator (یا برنامه ی شبیه ساز IOS) که همراه با محیط Xcode ارائه می شود، استفاده نمایید. Simulator به شما

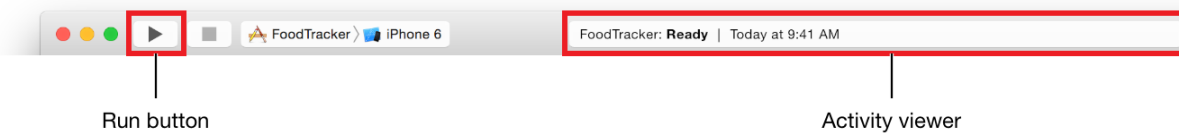
امکان می دهد اپلیکیشن خود را (در محیط شبیه ساز) اجرا کرده و پیش نمایشی از عملکرد و قابلیت های آن در یک دستگاه واقعی IOS را مشاهده نمایید.

Simulator قادر است سخت افزار دستگاه های iPad و iPhone (با صفحه نمایش بزرگ و کوچک) را شبیه سازی کند. این امر به شما امکان می دهد برنامه ی کاربردی خود را بر روی دستگاهی که می خواهید اپلیکیشن بر روی آن اجرا شود (اپلیکیشن را ویژه ی آن طراحی و تولید می کنید)، تست کرده و از عملکرد صحیح آن در دستگاه میزبان اطمینان حاصل نمایید. برای این پروژه simulator را بر روی گزینه ی iPhone 6 تنظیم نمایید. جهت اجرای برنامه در محیط شبیه ساز:

۱. به نوار ابزار Xcode مراجعه نموده، و از منوی pop-up (که Scheme نام دارد) گزینه ی iPhone 6 را انتخاب نمایید. این منوی pop-up به شما اجازه می دهد تا شبیه ساز یا دستگاهی که اپلیکیشن بر روی آن اجرا می شود را انتخاب نمایید. به خاطر داشته باشید که محیط میزبان برنامه را حتما بر روی گزینه ی iPhone 6 Simulator تنظیم نمایید، نه IOS Device.

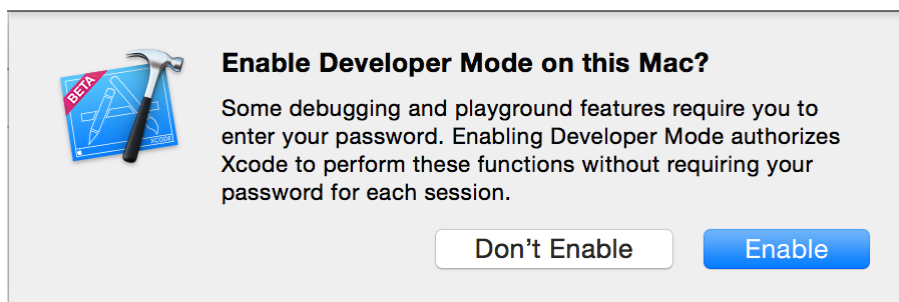


۲. جهت اجرا، بر روی دکمه ی Run در نوار ابزار محیط Xcode (واقع در بالای محیط سمت چپ) کلیک نمایید.



برای اجرای پروژه همچنین می توانید به دو روش رو به رو اقدام نمایید: ۱. **Product > Run**. ۲. کلید **Command-R** را فشار دهید.

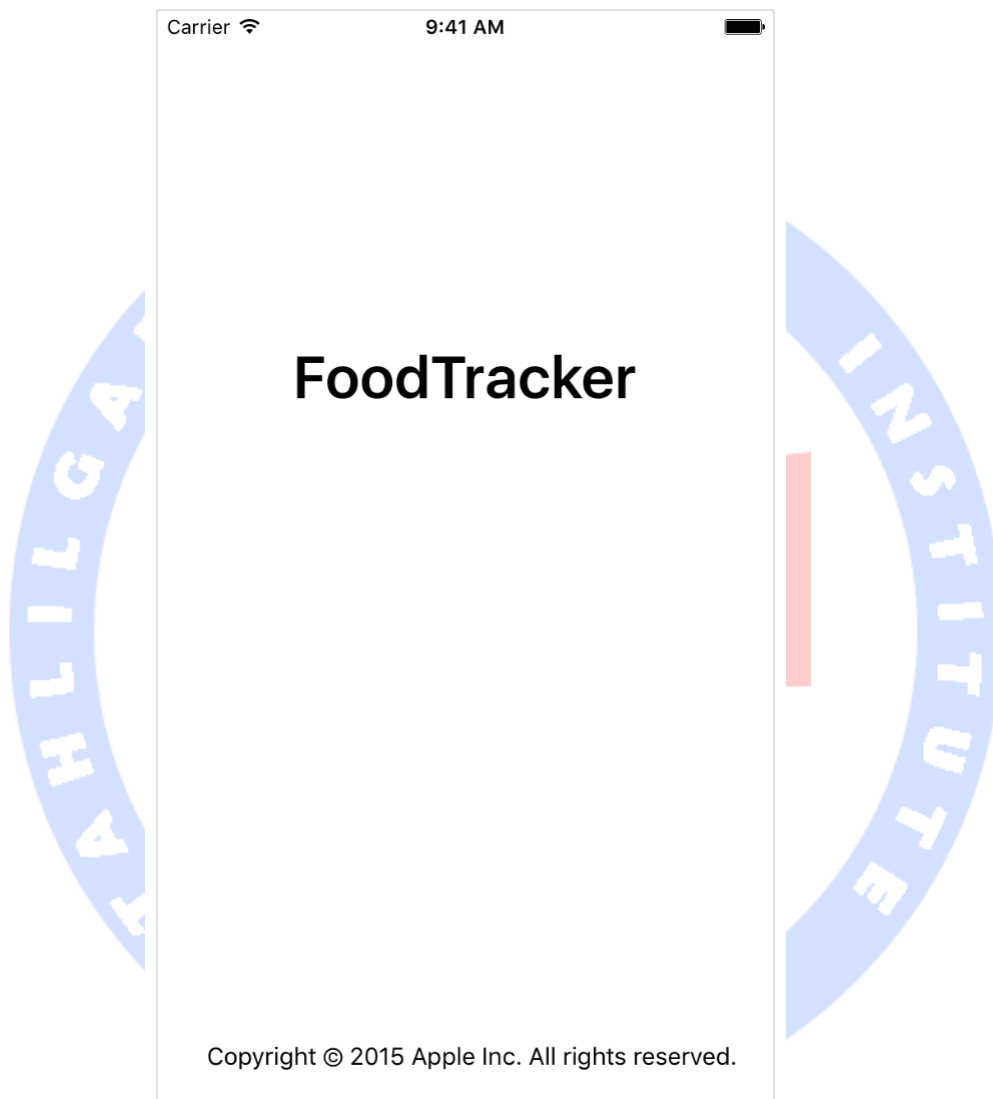
زمانی که برای اولین بار اپلیکیشنی را بر روی محیط **Mac** اجرا می کنید، **Mac** از شما می پرسد آیا مایلید **developer mode** را فعال نمایید یا خیر. این حالت به **Xcode** اجازه می دهد به برخی امکانات ویژه ی عیب یابی و **debugging** دسترسی داشته باشد، بدون اینکه برای هر بار دسترسی شما مجبور شوید گذرواژه ی لازم را وارد نمایید (برای دسترسی به برخی قابلیت های **debugging** و فایل های **playground** شما ملزوم به ارائه ی گذرواژه ی خود هستید. با فعال کردن این مد دیگر برای دسترسی به امکانات نام برده نیازی به ارائه ی پسورد نیست). پس از تصمیم گیری در خصوص فعال کردن/نکردن **developer mode**، بر روی دکمه ی مربوطه کلیک نمایید.



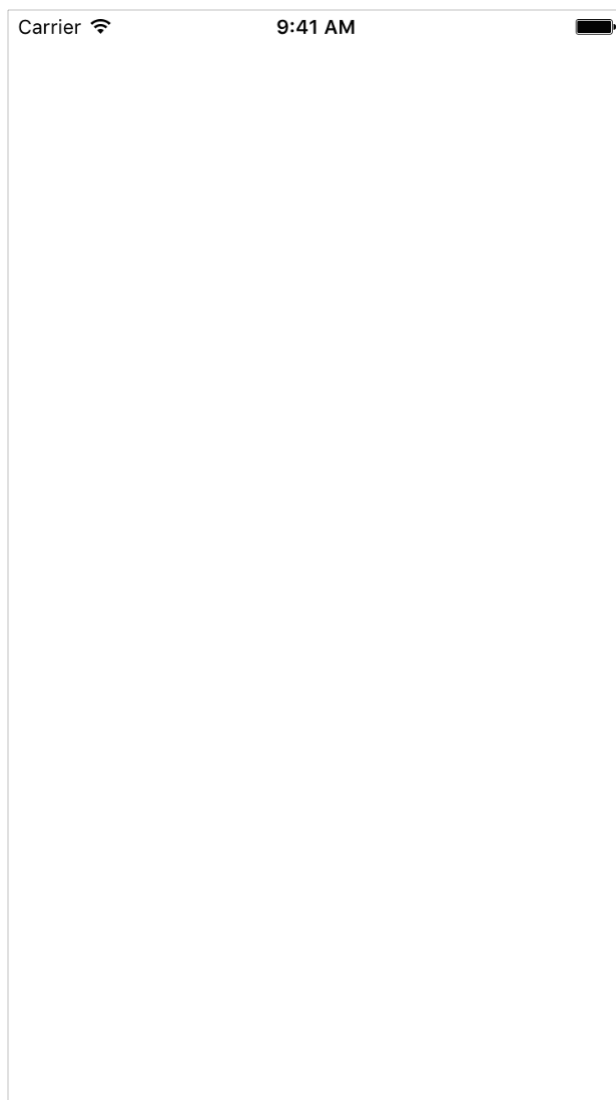
برای پروژه های این سری آموزشی، **developer mode** را فعال نمایید. ۴. همین که فرایند **build** تکمیل می شود، به نوار ابزار **Xcode** دقت کنید. **Xcode** اطلاعاتی را در خصوص فرایند **build**، داخل **activity viewer** نمایش می دهد. **viewer** در وسط نوار ابزار محیط **Xcode** قرار دارد.

پس از اینکه **Xcode** پروژه ی شما را با موفقیت کامپایل کرد، شبیه ساز به صورت خودکار راه اندازی می شود. راه اندازی برای اولین بار ممکن است کمی زمان ببرد. شبیه ساز/**simulator** در حالت **iPhone mode** اجرا شده، سپس اپلیکیشن شما در صفحه نمایش شبیه ساز اجرا و به نمایش گذاشته می شود. اما قبل از اینکه شبیه ساز

اپلیکیشن را به طور کامل راه اندازی (launch) کند، launch screen برنامه به همراه اسم آن (FoodTracker) به مدت چند ثانیه نشان داده شده و سپس خود اپلیکیشن اجرا می شود.



سپس صفحه ی زیر به نمایش در می آید:



در حال حاضر، قالب Single View Application کار خاصی انجام نمی دهد - همان طور که می بینید صرفاً یک صفحه ی سفید را برای کاربر به نمایش می گذارد. سایر Template هایی که در Xcode در اختیار شما قرار می دهد، از قابلیت ها و رفتارهای بسیار پیچیده تری برخوردار هستند.

لازم است قبل از انتخاب یک template و توسعه ی آن برای ساخت اپلیکیشن خود، از موارد استفاده و قابلیت های آن کاملاً مطلع باشید. اجرای برنامه در محیط شبیه ساز، بدون ایجاد تغییرات در آن روشی خوبی برای فهم این مسئله هست.

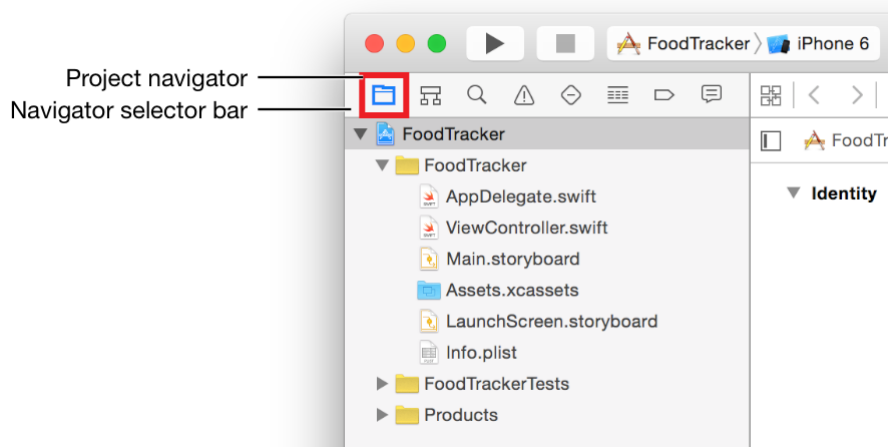
از محیط شبیه ساز خارج شوید (Quit Simulator > Simulator یا کلید Command-Q را فشار دهید).

### مشاهده و بررسی Source code پروژه

به هنگام انتخاب قالب Single View Application، تعدادی فایل source code همراه آن ارائه می شود که این فایل ها وظیفه ی تنظیم محیط اپلیکیشن را بر عهده دارند. ابتدا فایل AppDelegate.swift را مورد بررسی قرار می دهیم.

برای مشاهده ی محتویات فایل AppDelegate.swift:

۱. ابتدا project navigator را در navigator area باز کنید. project navigator تمامی فایل های پروژه را نمایش می دهد. اگر project navigator باز نبود، می توانید آن را با کلیک بر روی دکمه ی سمت چپ در نوار navigator selector، باز نمایید (یا می توانید این مسیر را طی نمایید: View > Navigators > Show Project Navigator).



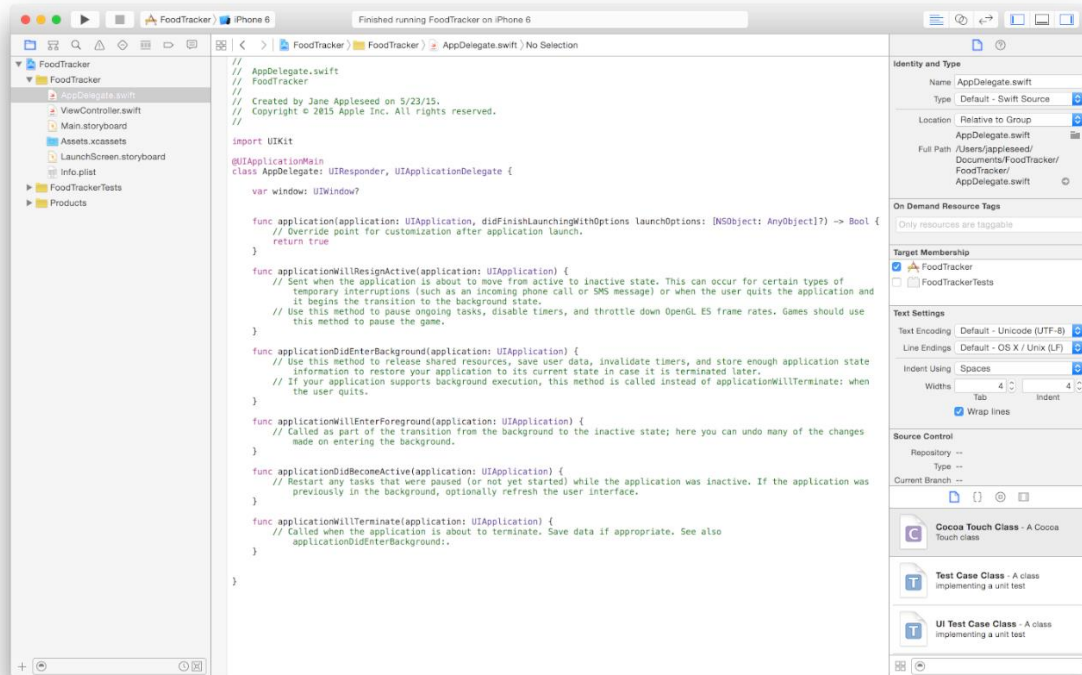
۲. در صورت لزوم، پوشه ی FoodTracker را (در کادر project navigator) با

کلیک بر روی آیکن مثلث کوچک در کنار آن، باز نمایید.

۳. فایل AppDelegate.swift را انتخاب نمایید. Xcode فایل نام برده را داخل

editor area (ناحیه ی ویرایش) پنجره ی محیط باز کرده و محتویات/کدهای آن را

به نمایش می گذارد.



و یا جهت مشاهده ی محتویات فایل در پنجره ی مجزا، دوبار بر روی فایل AppDelegate.swift کلیک نمایید.

فایل AppDelegate

فایل AppDelegate.swift دو کاربرد اصلی دارد:

- علاوه بر entry point یا نقطه ی شروع اجرای برنامه، یک run loop تعریف می کند که وظیفه ی تحویل رخدادهای ورودی به برنامه را دارد (run loop یک حلقه ی پردازش event است که برای زمان بندی کارها و هماهنگ سازی دریافت رخدادهای ورودی به برنامه، مورد استفاده قرار می گیرد. به عبارت دیگر هدف از run loop، فعال نگه داشتن thread به هنگام نیاز/زمانی که عملیاتی و کارهایی باید پردازش و انجام شوند و در حالت sleep قرار دادن آن در زمان بی کاری است). این کار توسط attribute ای به نام UIApplicationMain، درج شده در بالای فایل مزبور، انجام می شود.



UIApplicationMain در واقع یک application object (مسئول مدیریت life cycle برنامه از زمان اجرا تا پایان) و یک آبجکت app delegate تعریف می کند. در زیر به شرح مفهوم app delegate خواهیم پرداخت.

- این فایل همچنین کلاس AppDelegate را تعریف می کند. کلاس مذکور الگو یا نقشه ای برای (ساخت) آبجکت app delegate می باشد. app delegate نیز پنجره ای را می سازد که محتوای اپلیکیشن شما در آن به تصویر کشیده شده و بستری را تعریف می کند که در آن می توان به تغییرات در وضعیت برنامه واکنش نشان داد. کدهای اختصاصی در سطح اپلیکیشن (custom app-level code) داخل کلاس AppDelegate تعریف می شوند.

کلاس AppDelegate یک property بیشتر ندارد و آن window است. app delegate به کمک این property پنجره ای که محتوای برنامه در آن نمایش داده می شود را رصد می کند. window از نوع optional است بدین معنی که می تواند در برهه ای از زمان هیچ مقداری نداشته و به اصطلاح nil باشد.

var window: UIWindow?

کلاس AppDelegate همچنین الگویی آماده برای پیاده سازی متدهای پرکاربرد و مهم ارائه می دهد. این متدهای از پیش تعریف شده به application object امکان می دهند تا با app delegate تعامل داشته و ارتباط برقرار کند.

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool
func applicationWillResignActive(application: UIApplication)
func applicationDidEnterBackground(application: UIApplication)
func applicationWillEnterForeground(application: UIApplication)
func applicationDidBecomeActive(application: UIApplication)
func applicationWillTerminate(application: UIApplication)
```

در طول تغییر برنامه از یک وضعیت به وضعیت دیگر (state transition) - برای مثال، اجرا و شروع برنامه، تغییر وضعیت به پس زمینه و اتمام آن - application object متد

متناظر در app delegate را صدا زده و به آن امکان می دهد پاسخ یا واکنش مناسب را نشان دهد.

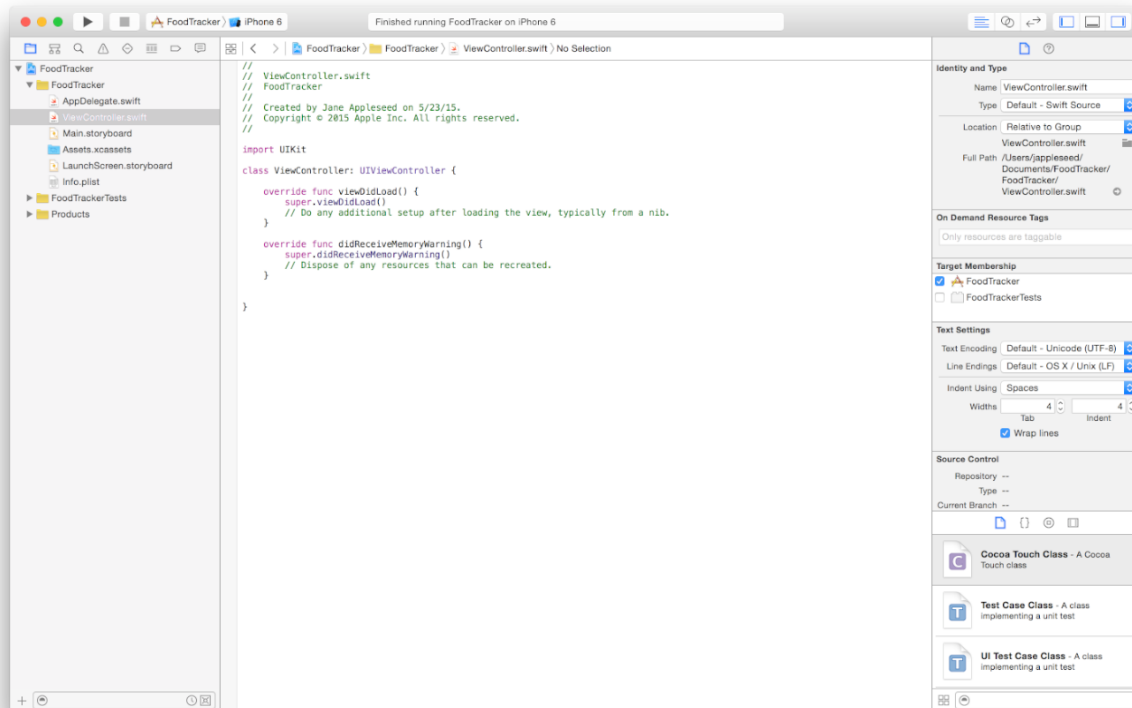
لازم نیست زمان فراخوانی این متدها را خود مدیریت کرده و از اجرای به موقع آن ها اطمینان حاصل کنید. application object این وظیفه را بر عهده می گیرد.

هر یک از این متدها (که به صورت خودکار پیاده سازی می شوند) رفتار پیش فرضی داشته و عملیات خاصی را انجام می دهند. اگر پیاده سازی های الگو (template implementation) را خالی گذاشته یا آن را کلا از کلاس AppDelegate حذف کنید، در آن صورت هرگاه که متد صدا خورده می شود، دقیقاً همان رفتار پیش فرض را از خود نشان خواهد داد. می توانید به این method template ها (الگو پیاده سازی یا قالب آماده ی متدها) کد اختصاصی خود را اضافه کنید تا با فراخوانی آن متد، رفتار دلخواه را به اجرا بگذارند.

در این مبحث هیچ کد اختصاصی به فایل AppDelegate.swift اضافه نخواهیم کرد.

#### فایل View Controller

با انتخاب قالب Application View Single، یک فایل دیگر به نام ViewController.swift ایجاد می شود. برای مشاهده ی محتوای این فایل بر روی ViewController.swift در کادر project navigator کلیک نمایید.



این فایل یک کلاس فرزند/subclass از UIViewController به نام ViewController ایجاد می‌کند. در حال حاضر، این کلاس صرفاً رفتارهای (تعریف شده توسط) کلاس UIViewController را به ارث می‌برد. جهت بازنویسی (override) یا بسط این رفتارها، لازم است متدهای تعریف شده در UIViewController را override نمایید (همان طور که دو متد viewDidLoad() و didReceiveMemoryWarning() در فایل ViewController.swift بازنویسی شده‌اند) و یا متدهای اختصاصی خود را پیاده سازی کنید.

در این فایل متدی به نام didReceiveMemoryWarning() نیز پیاده سازی شده که در حال حاضر با آن کاری نداریم. می‌توانید این متد را از کلاس ViewController حذف کنید.

پس از حذف متد ذکر شده، کد فایل ViewController.swift مشابه زیر خواهد بود:

```
import UIKit
class ViewController: UIViewController {
```

```

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
}
}

```

در ادامه ی مبحث حاضر کدهایی را به این فایل اضافه خواهیم کرد.

### مشاهده ی محتوای فایل Storyboard

در این بخش با فایل storyboard اپلیکیشن خود کار خواهیم کرد.

Storyboard در واقع فایلی است که نمودی (ظاهری) از (کدهای پشت) رابط کاربری اپلیکیشن را ارائه می دهد. به عبارتی روشن تر کدهای نوشته شده در فایل، به صورت مستقیم اجرا شده و می توان در هر لحظه خروجی کدها را مشاهده نمود.

در واقع با استفاده از storyboard جریان برنامه را به تصویر می کشید، آنچه برای برنامه طراحی می کنید را در حین طراحی و در لحظه مشاهده نموده، می توانید بخش هایی که درست کار می کند و بخش هایی که از عملکرد صحیح برخوردار نیستند را به چشم خود ببینید، تغییرات لازم را بر روی ال اعمال نمایید و مستقیماً نتیجه ی آن را مشاهده کنید.

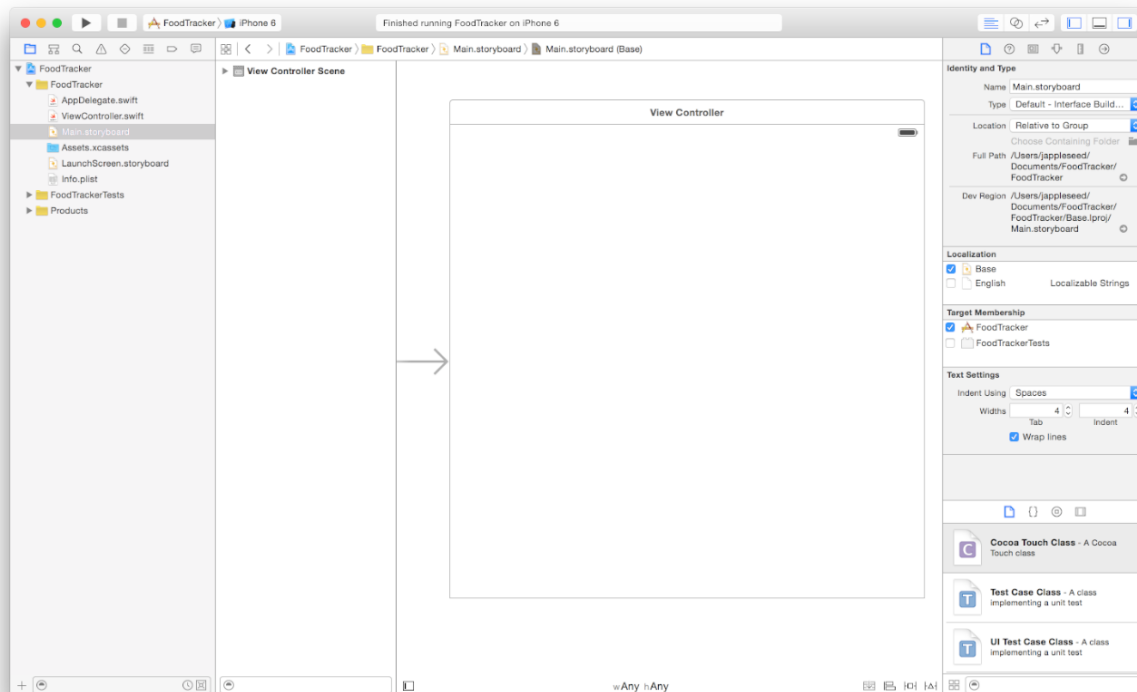
برای مشاهده ی فایل storyboard:

- در کادر project navigator، بر روی فایل Main.storyboard کلیک نمایید.
- Xcode این فایل را داخل Interface Builder، در ناحیه ی ویرایشگر ( editor area ) محیط باز می کند.

به پس زمینه ی storyboard در اصطلاح Canvas گفته می شود. می توانید

Canvas را برای افزودن و مدیریت چیدمان المان های ال بکار ببرید.

Storyboard می بایست ظاهری مشابه زیر داشته باشد:



در این مرحله، storyboard تنها یک scene دارد. به یک صفحه از برنامه که حامل محتوا است در اصطلاح scene گفته می شود. پیکانی که به ضلع سمت چپ صفحه ی محتوا یا scene (بر روی canvas) اشاره می کند، نقطه ی شروع storyboard می باشد. بدین معنی که این scene اولین صفحه ای است که به مجرد اجرای اپلیکیشن بارگذاری می شود (اولین صفحه ی محتوای برنامه که جریان و چرخه ی اپلیکیشن از آنجا شروع می شود).

در حال حاضر، scene جاری (که بر روی canvas یا پس زمینه ی storyboard مشاهده می کنید) دربردارنده ی یک view بیشتر نیست. مدیریت این View یا هر View دیگری بر عهده ی view controller است. به زودی درباره ی مفهوم view controller، view و وظیفه ی هر یک توضیحاتی را ارائه خواهیم داد.

زمانی که اپلیکیشن خود را بر روی محیط شبیه ساز iPhone 6 در Xcode راه اندازی کردید، آنچه با اجرای برنامه برای شما به نمایش گذاشته شد، در واقع همین view یا محتوایی است که هم اکنون در این scene مشاهده می کنید. اما یک نکته ی جالب توجه: با دقت به scene مورد نظر، متوجه می شوید که ابعاد آن با اندازه ی صفحه

نمایش iPhone 6 دقیق همخوانی ندارد. به نظر شما علتش چیست؟ در پاسخ باید گفت که scene قابل مشاهده بر روی پس زمینه یا canvas، یک نمای کلی از رابط کاربری است که برای هر دستگاهی و با هر جهت نمایشی (نمای افقی/عمودی)، قابل استفاده می باشد.

از این نمای کلی برای ایجاد یک adaptive interface یا رابط کاربری سازگار با (اندازه و جهت نمایش) صفحه نمایش دستگاه میزبان استفاده می شود. به عبارت دیگر adaptive interface، همان طور که از نامش پیدا است، یک رابط کاربری انعطاف پذیر است که خود را به صورت اتوماتیک، جهت نمایش دقیق المان های UI، متناسب با صفحه نمایش دستگاه میزبان تنظیم می کند.

#### ساخت یک رابط کاربری/ UI ساده

اکنون به ساخت یک رابط کاربری ساده می پردازیم. ابتدا UI را برای scene (صفحه ی محتوایی) طراحی می کنیم که به ما اجازه می دهد یک آیتم جدید به برنامه ی ثبت و رصد اطلاعات غذا (meal tracking app)، که آن را FoodTracker نام گذاری کردیم، اضافه نماییم.

Xcode کتابخانه ای از اشیای آماده (object library) دارد که می توانید آن ها را به فایل storyboard اضافه نمایید. برخی از این اشیاء، المان های قابل مشاهده هستند که در UI به نمایش درمی آیند. از جمله ی آن های می توان به button و text field اشاره کرد. برخی دیگر مانند view controller ها و gesture recongnizer ها، صرفاً رفتار اپلیکیشن را تعریف می کنند و بر روی صفحه نمایش قابل مشاهده نیستند.

المان هایی که در UI برای کاربر قابل مشاهده هستند در اصطلاح view خوانده می شوند. View ها در واقع محتوای اپلیکیشن را برای کاربر به نمایش می گذارند. می توان آن ها را ابزار اصلی و مصالح تشکیل دهنده ی رابط کاربری تلقی کرد که توسط آن محتوا را به صورت کارآمد و زیبا در صفحه نمایش به تصویر می کشیم.

View ها طیفی وسیعی از رفتارهای ذاتی و درون ساخته دارند که از جمله ی آن ها می توان از نمایاندن خود بر روی صفحه نمایش و واکنش نشان دادن به ورودی کاربر نام برد.

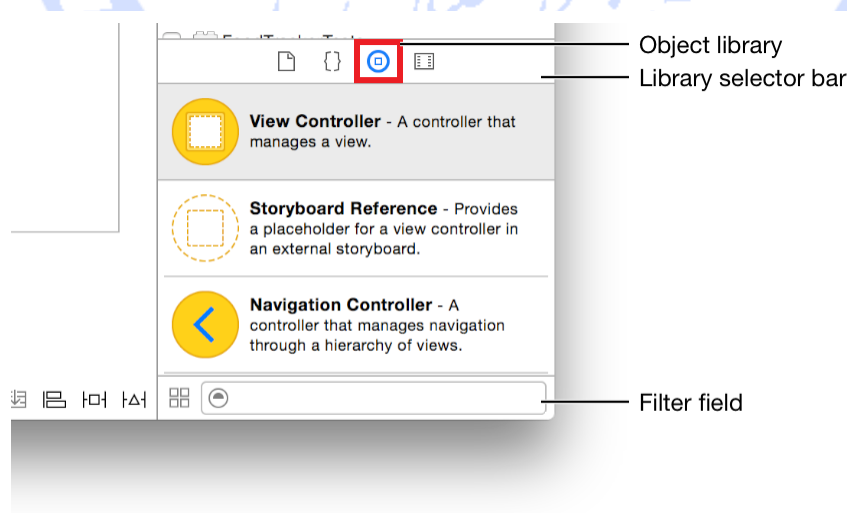
تمامی object view ها در iOS از جنس UIView یا یکی از کلاس های فرزند (subclass) آن هستند. بسیاری از کلاس هایی که از UIView ارث بری می کنند (subclass های آن محسوب می شوند)، در ظاهر و رفتار منحصر بفرد هستند.

برای شروع یک کادر متن/text field (کلاس UITextField که یکی از کلاس های زیرمجموعه ی UIView است) به scene اضافه می کنیم. همان طور که می دانید، text field یک کادر متن است که به کاربر اجازه می دهد، اسم آیتم جدید (در این برنامه نام غذا) را وارد کند.

برای اضافه کردن یک text field به scene:

۱. Object Library را باز نمایید.

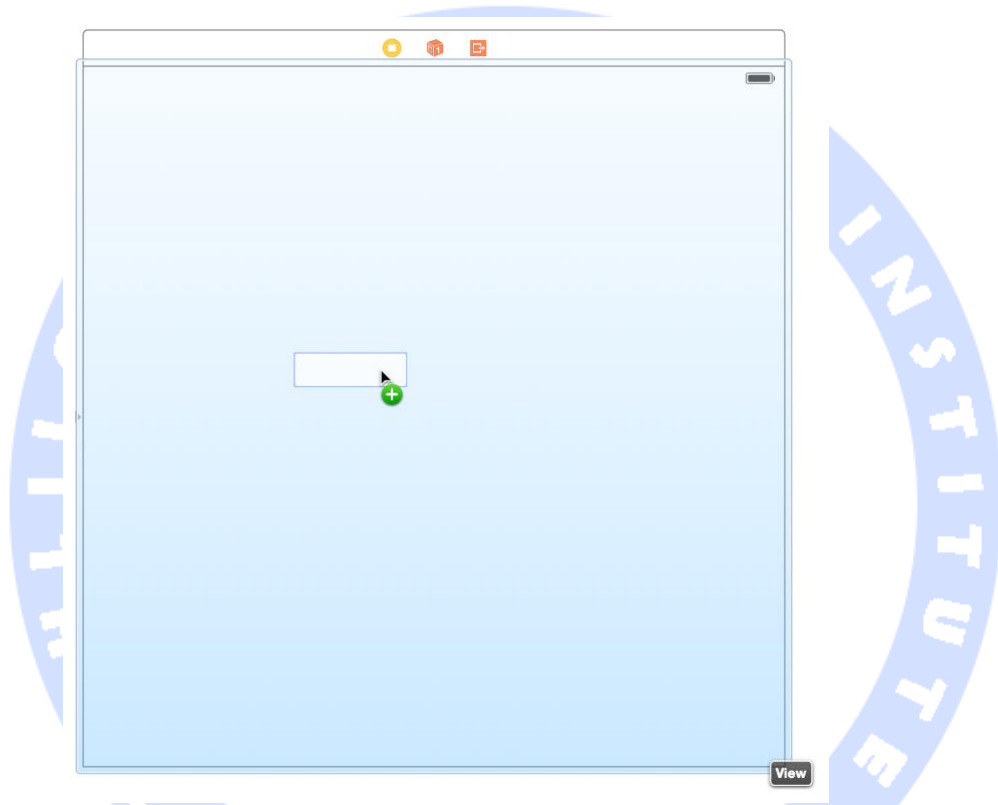
Object library در پایین utility area، سمت راست محیط کاری Xcode قابل دسترسی می باشد. اگر Object library قابل مشاهده نبود، می توانید با کلیک بر روی دکمه سوم از سمت چپ در library selector bar، آن را باز کنید (یا می توانید به این روش اقدام نمایید: View > Utilities > Show Object Library).



یک لیست پدیدار می شود که اسم هر آبجکت را به همراه شرحی درباره ی کاربرد و نمای ظاهری آن نمایش می دهد.

۲. در Object library، اسم آبجکت دلخواه (text field) را داخل کادر جستجو (filter field) وارد نمایید. بدین وسیله شی مورد نظر به سرعت در اختیار شما قرار می گیرد.

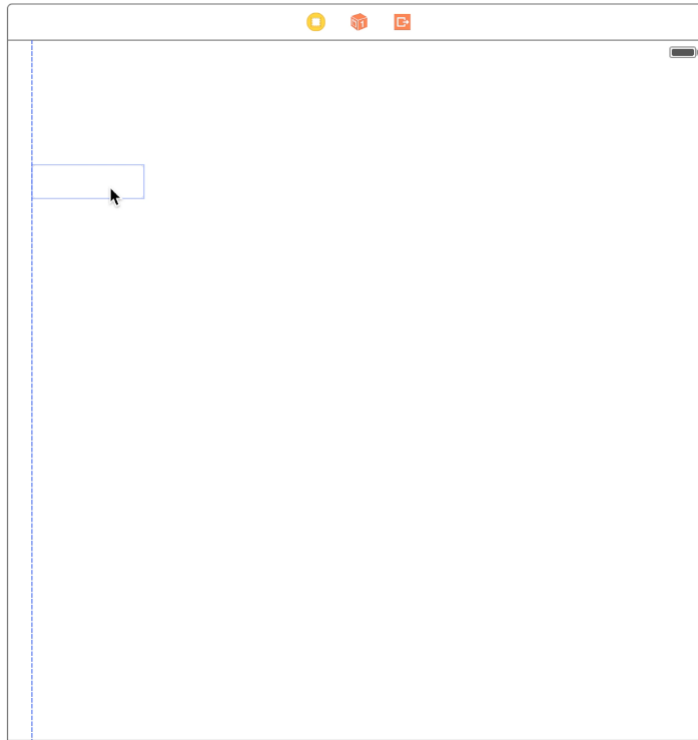
۳. آبجکت Text Field را از Object Library کشیده و بر روی سطح scene جایگذاری کنید.



در صورت لزوم، با طی این مراحل Editor > Canvas > Zoom، زمینه را بزرگ نمایید.

۴. بر روی Text field کلیک کرده و آن را در جهت نیمه ی بالایی scene، هم تراز با حاشیه ی سمت چپ بکشید. هنگامی که خط نقطه چین آبی رنگ مانند زیر نمایان شد، کشیدن را متوقف کرده و آبجکت مزبور را رها کنید.

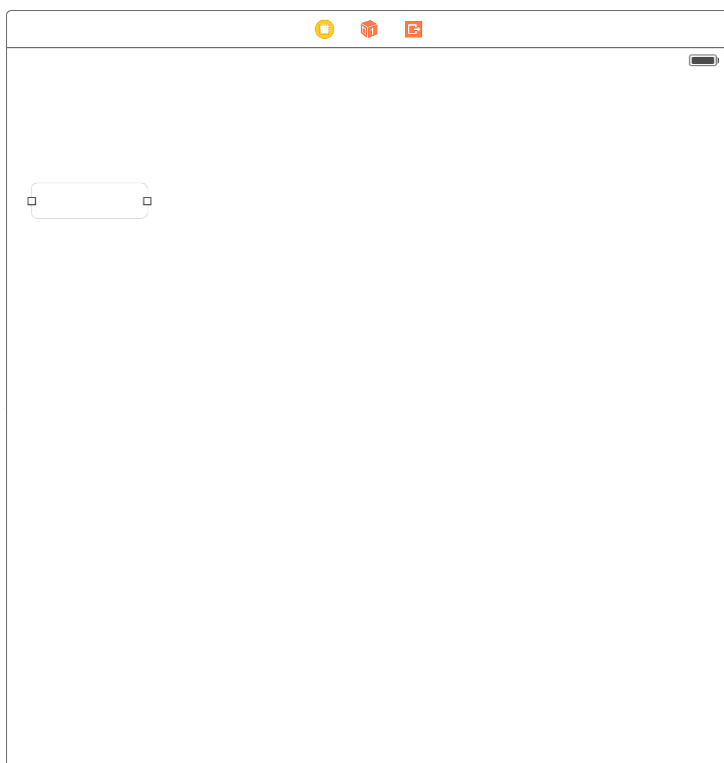




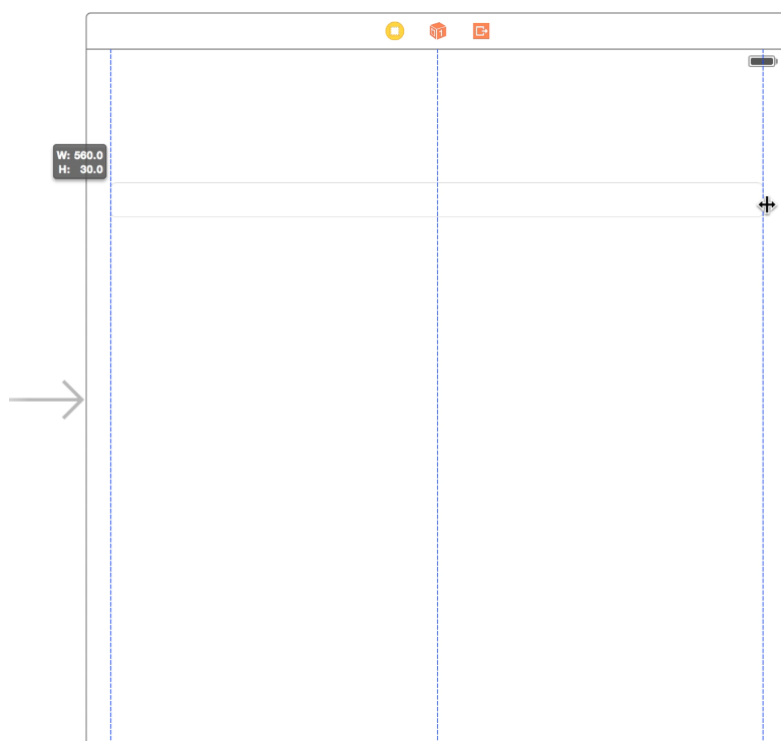
راهنمای آبی رنگ به شما کمک می کند آجکت text field را در جایگاه مناسب آن قرار دهید. در حقیقت این راهنما تنها زمانی پدیدار می شود که آجکت ها را در نزدیکی آن ها جابجا کرده یا تغییر اندازه دهید. هرگاه آجکت را رها کنید، راهنمای آبی رنگ نیز محو می شود.

۵. جهت تنظیم اندازه ی المان مورد نظر، کافی است `resize handle` آن المان را بکشید. `Resize handle` یا دستگیره های تنظیم اندازه، مربع های کوچک سفید رنگی هستند که با کلیک بر روی المان مورد نظر، بر روی لبه های آن المان نمایان می شوند.

همان طور که گفته شده برای نمایان شدن `resize handle`، باید المان را با کلیک بر روی آن انتخاب نمایید. از آنجایی که text field را تازه رها کردیم، المان انتخاب شده و `resize handle` های آن هنوز قابل مشاهده می باشند.



۶. لبه های سمت چپ و راست المان را در جهات مربوطه کشیده تا سه خط عمودی آبی رنگ به صورت زیر نمایان شود.

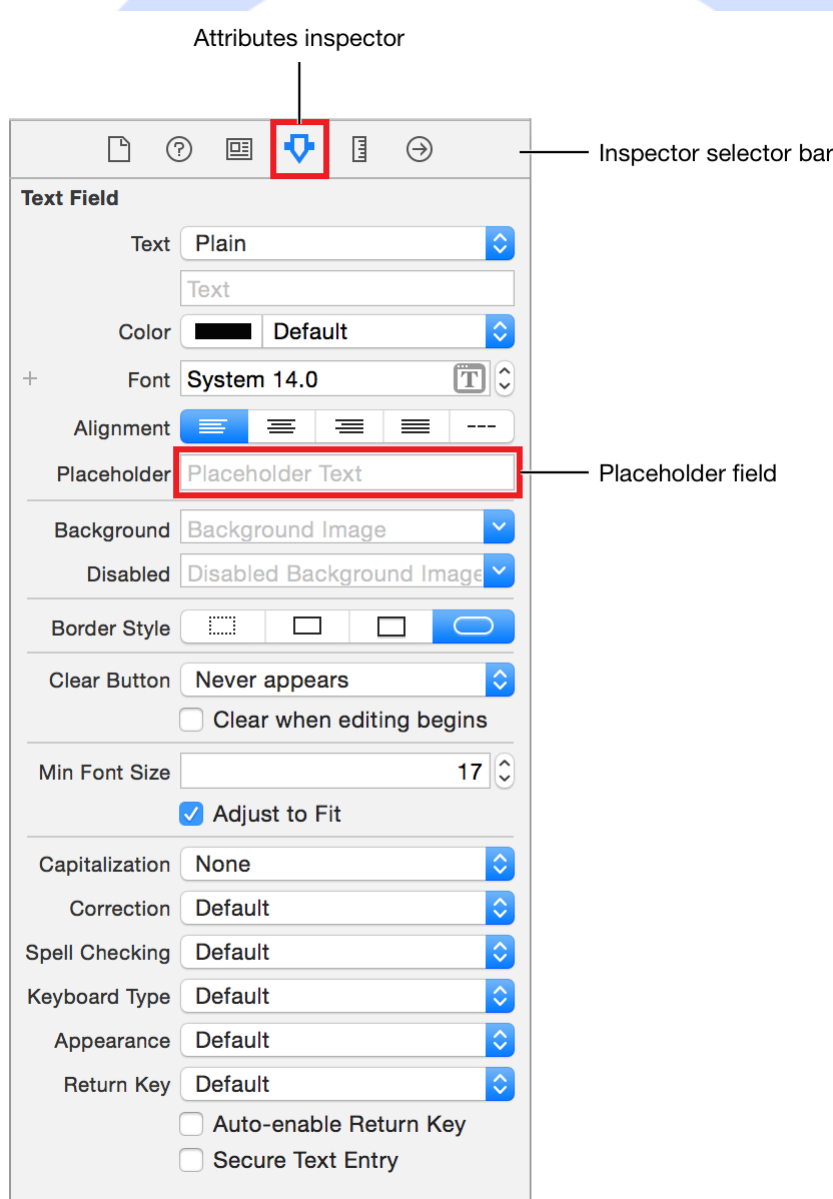


Text field در scene جایگذاری شده، اما هیچ سرنخی که کاربر را در خصوص ورودی مورد انتظار راهنمایی کند، لحاظ نشده است. حال بایستی با استفاده از فیلد

placeholder، ویژگی ی آجکت text field، به کاربر اعلان کنیم که در کادر متن اسم یک غذا را درج کند.

جهت تنظیم مقدار فیلد placeholder مربوط به این آجکت:

۱. پس از انتخاب text field، به utility area داخل محیط کاری Xcode مراجعه کرده و Attributes inspector را (در inspector selector bar) با کلیک بر روی دکمه ی سوم از سمت راست، باز نمایید. Attributes inspector به شما امکان می دهد مقدار property های یک آجکت را در storyboard ویرایش نمایید.



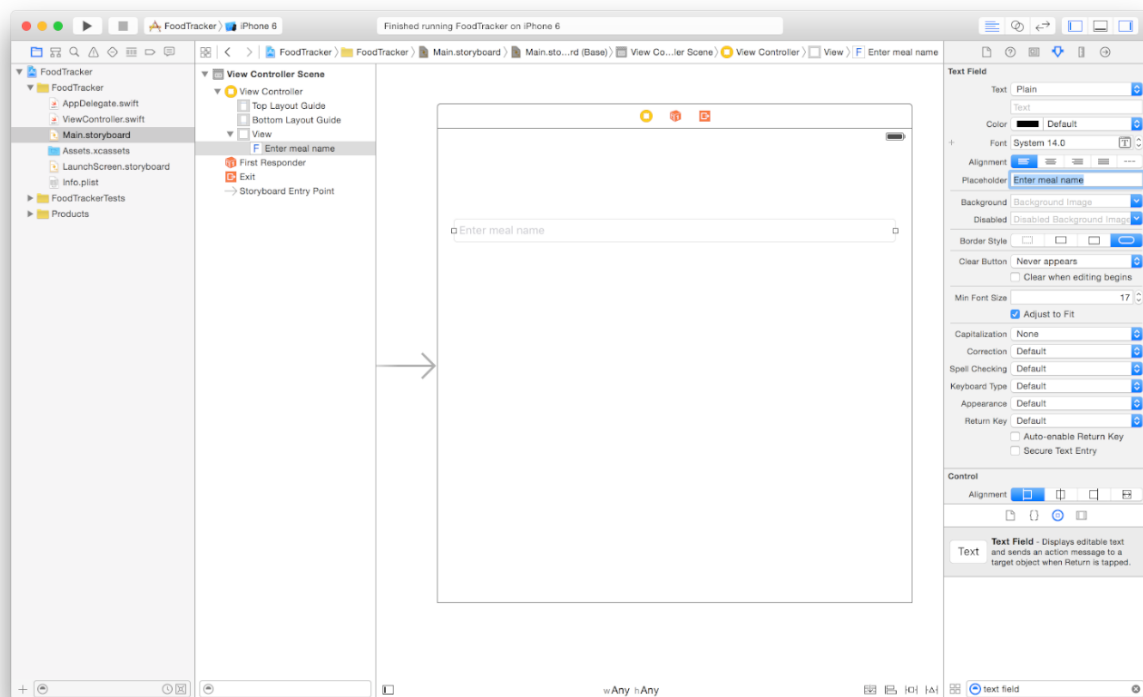
۲. داخل کادر Attributes inspector، فیلدی که Placeholder نام دارد را یافته و

مقدار Enter meal name را در آن وارد نمایید.

۳. کلید Return را فشار داده تا متن مورد نظر داخل آبجکت text field نمایش داده

شود.

پس از اعمال تغییرات نام برده، scene ظاهری مشابه نمونه ی زیر خواهد داشت:



همزمان با ویرایش attribute های آبجکت text field، همچنین می توانید attribute

های صفحه کلیدی که با انتخاب آبجکت ذکر شده نمایان می شود را تغییر دهید.

جهت تنظیم صفحه کلید:

۱. Text field را انتخاب نمایید.

۲. داخل کادر Attributes inspector، فیلدی که Return Key نام دارد را یافته و

مقدار آن را بر روی Done تنظیم کنید. با این کار مقدار پیش فرض Return در

صفحه کلید به Done تغییر کرده و بیشتر مورد توجه کاربر قرار می گیرد.

۳. حال در کادر Attributes inspector، گزینه ی Auto-enable Return Key را

تیک دار نمایید. با فعال نمودن این گزینه، کاربر دیگر قادر نخواهد بود بدون وارد

کردن متن در text field (اسم غذا) کلید Done را فشار دهد.

اکنون یک label (از کلاس UILabel) جهت ارائه ی اطلاعات درباره ی کادر متن

(آبجکت text field) به بالای آن اضافه می کنیم. همان طور که می دانید label یک

المان هوشمند و تعاملی نیست بلکه صرفاً یک متن ساده را برای کاربر نمایش می

دهد. برای اینکه شما بتوانید نحوه ی تعریف تعامل بین المان های مختلف UI را بهتر

درک کنید، این label را طوری تنظیم می کنیم که ورودی کاربر در کادر متن را نمایش

دهد. با این کار می توان اطمینان حاصل نمود که المان مورد نظر ورودی کاربر را به

درستی دریافت و پردازش می کند.

جهت افزودن یک label به scene:

۱. به Object library مراجعه نموده، واژه ی label را در کادر جستجوی (filter

field) آن وارد نمایید تا آبجکت Label به سرعت در اختیار شما قرار گیرد.

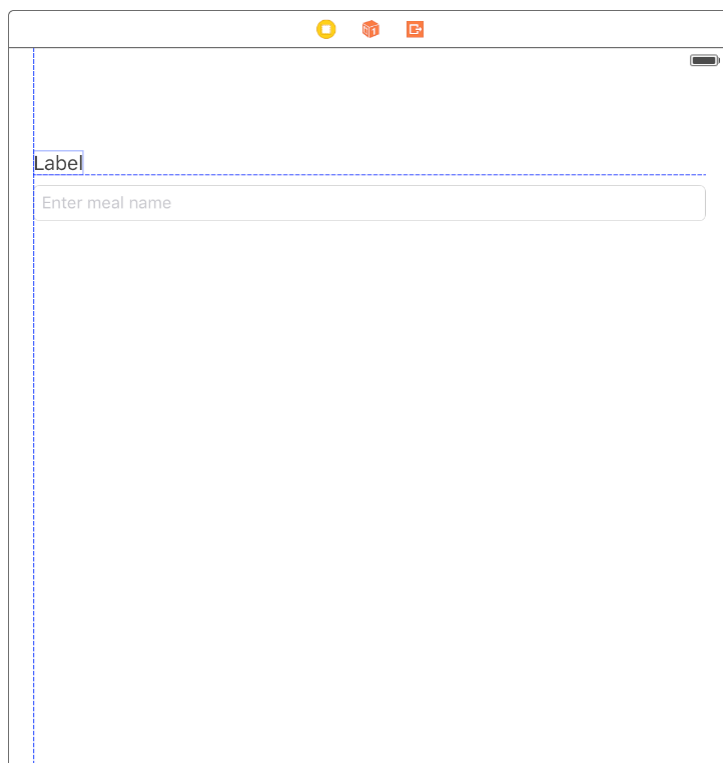
۲. آبجکت Label را از Object library کشیده و بر روی سطح scene جایگذاری

کنید.

۳. آبجکت نام برده را به بالای کادر متن کشیده و آن را همتراز با حاشیه ی سمت

چپ قرار دهید. هنگامی که خط نقطه چین افقی، مانند زیر نمایان می شود،

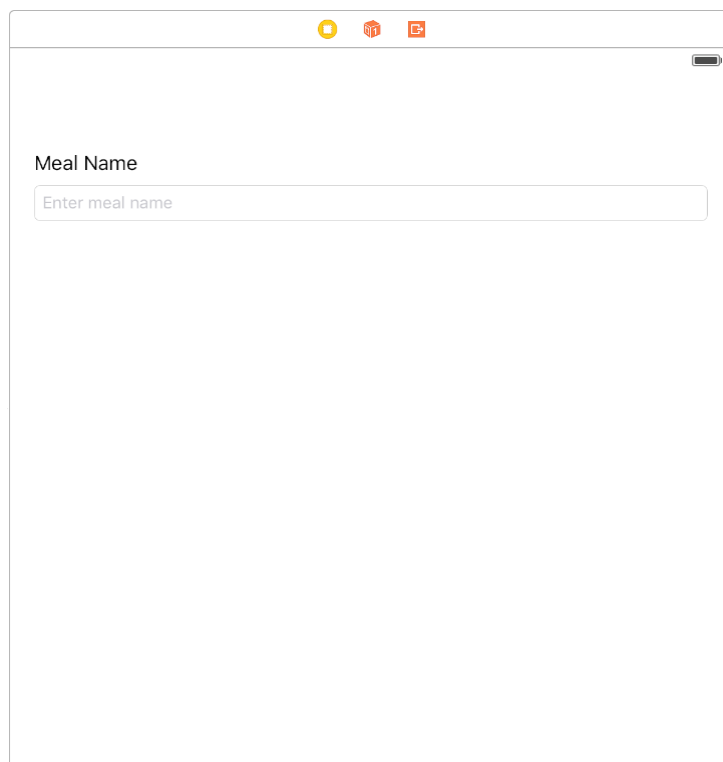
المان را رها کنید.



۴. بر روی label دوبار کلیک کرده و سپس واژه ی Meal Name را وارد نمایید.

۵. کیلد Return را فشار داده تا متن جدید در label نمایش داده شود.

با اضافه شدن یک المان جدید به scene، ظاهر برنامه به این صورت در خواهد آمد:

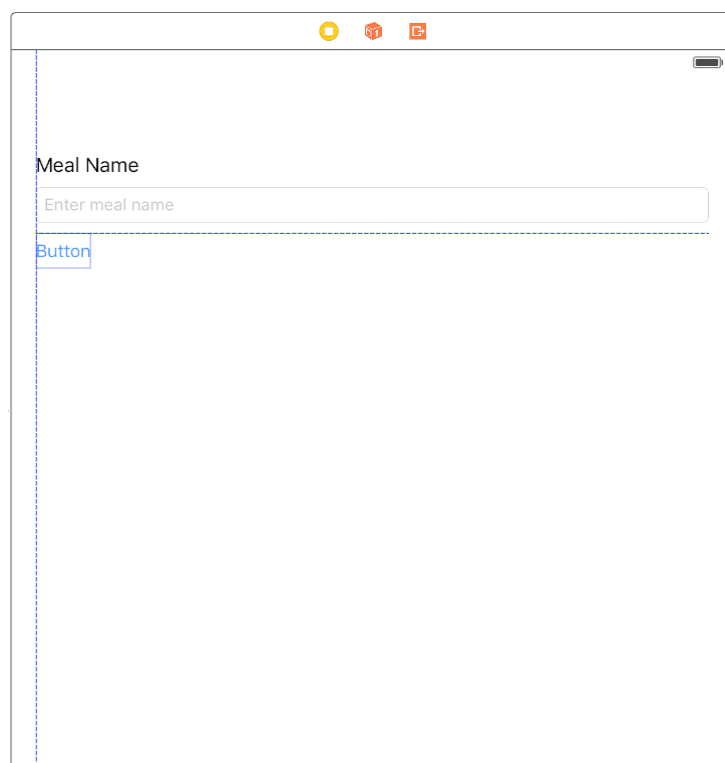


آخرین المانی که به رابط کاربری اضافه می‌کنیم، یک دکمه (از کلاس UIButton) است. دکمه برخلاف label، یک المان تعاملی می‌باشد. بدین معنی که با کلیک بر روی آن، فعل یا عملیات خاصی (که شما تعریف می‌کنید) انجام می‌شود. در همین مبحث رفتاری (action) برای دکمه تعریف می‌کنیم که با کلیک بر روی آن، متن label را به مقدار پیش فرض برمی‌گرداند.

به منظور افزودن المان دکمه به scene:

۱. به library Object مراجعه نمایید و پس از یافتن آبجکت Button، آن را کشیده و در سطح scene جایگذاری کنید.

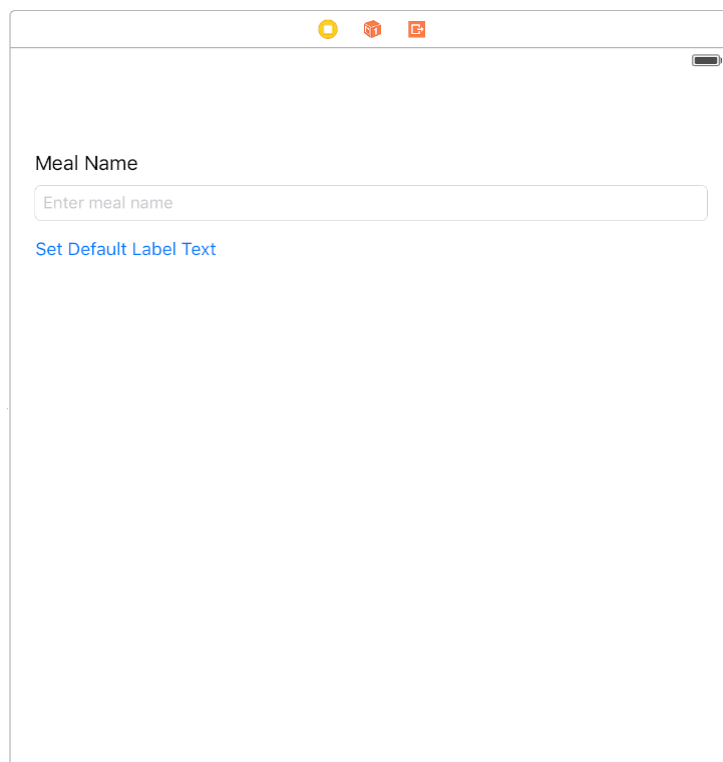
۲. حال المان دکمه را کشیده و در زیر کادر متن قرار دهید. زمانی که دو خط نقطه چین آبی رنگ نمایان شد، المان را رها کنید.



۳. بر روی دکمه دابل کلیک کرده و عبارت Set Default Label Text را وارد نمایید.

۴. کلید Return را فشار دهید. متن جدید بر روی دکمه نمایش داده می‌شود.

## ظاهر جدید برنامه:

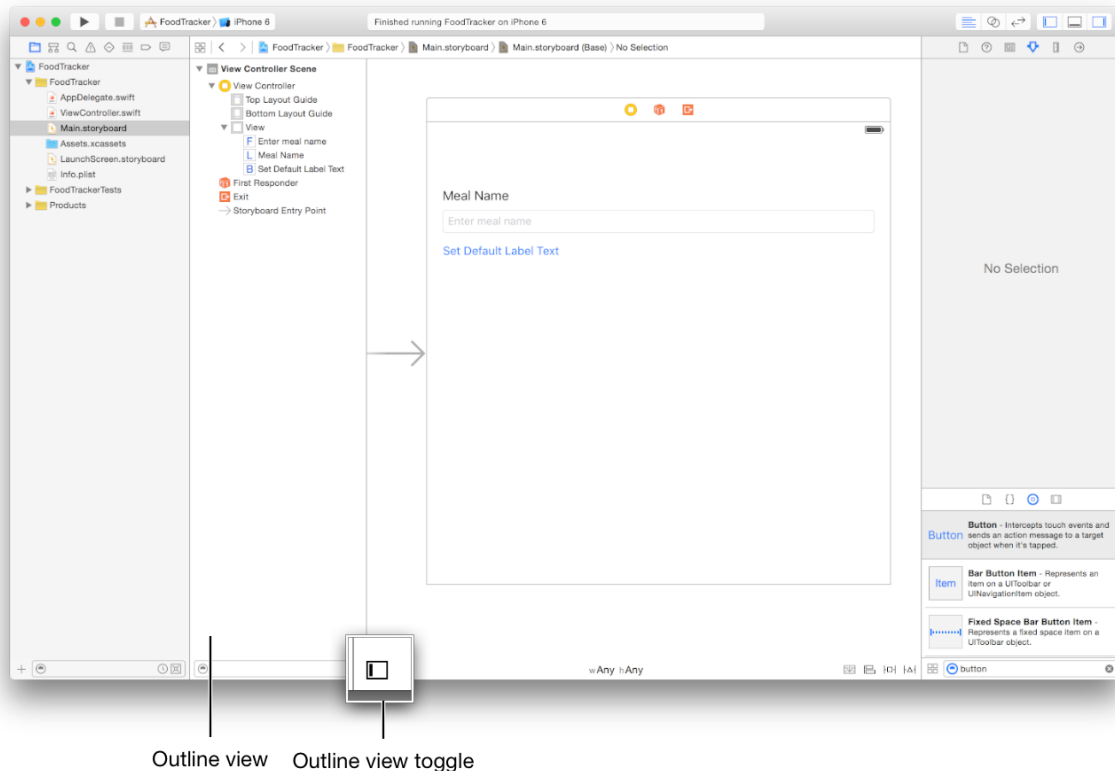


بد نیست پس از افزودن تمامی المان های لازم، چیدمان آن ها در scene را بررسی کنیم. برای این منظور از outline view کمک می گیریم.

جهت مشاهده ی outline view: آموزشگاه خلیفه داده

۱. در Storyboard، دکمه ی toggle (نمایش دادن/پنهان کردن) outline view را پیدا کنید.





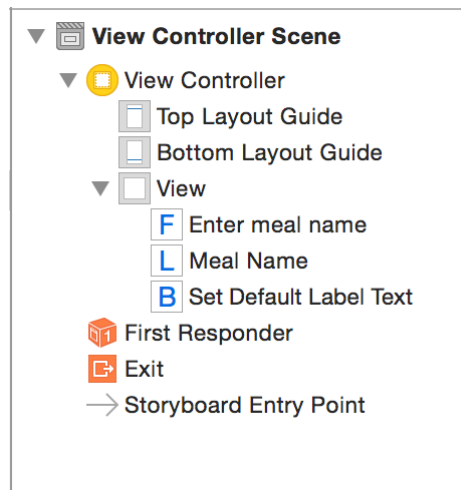
Outline view    Outline view toggle

۲. در صورتی که **view outline** در حالت **collapsed** قرار داشته باشد (پنهان باشد)، بر روی دکمه ی **toggle** کلیک کرده تا کادر **view outline** باز شود. با این دکمه می توانید **view outline** را با توجه به نیاز باز کرده و یا آن را جمع کنید.

**Outline view** – کادری که در سمت چپ **canvas** نمایش داده می شود – به شما اجازه می دهد اشیای موجود در **storyboard** را با نمای درختی (به صورت سلسله مراتب) مشاهده نمایید. به عبارت بهتر در این کادر می توانید المان هایی که به رابط کاربری برنامه خود اضافه کردید را در سلسله مراتب مشاهده نمایید. با دیدن این سلسله مراتب در کادر مذکور، شاید این سوال پیش آید که چرا المان های **UI** در زیر گره ی **View** لیست شده اند (به عبارت دیگر داخل **view** دیگر گنجانده شده اند)؟

**View** ها علاوه بر نمایش محتوا و تعامل با کاربر، می توانند ظرفی برای نگهداری سایر **view** ها باشند. **View** ها در قالب یک سلسله مراتب نمایش داده می شوند که در اصطلاح **view hierarchy** خوانده می شود. **view hierarchy** چیدمان و ترتیب **view** ها را نسبت به دیگر **view** ها مشخص می کند. داخل این سلسله مراتب، **view** هایی که داخل **view** دیگری جای گرفته اند **subview** نامیده و **view** هایی که دربرگیرنده و

میزبان view های دیگر هستند superview خوانده می شوند. یک view می تواند چندین subview داشته باشد، اما خود زیرمجموعه ی تنها یک superview باشد.



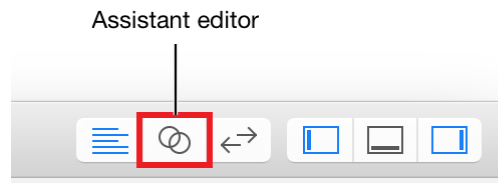
به طور کلی هر scene یک view hierarchy دارد. در بالاترین مرتبه ی هر view hierarchy نیز یک view content مشاهده می شود. برای مثال در scene جاری، View ای که در بالاترین مرتبه قرار دارد، content view نامیده می شود و همان طور که می بینید خود داخل View Controller قرار دارد. View قرار دارد. button، label، text field و هر المان دیگری که داخل این scene قرار می دهید زیرمجموعه ی content view خواهد بود (هرچند این المان ها نیز می توانند view های دیگری به عنوان زیرمجموعه ی خود داشته باشند).

### پیش مشاهده ی User Interface برنامه

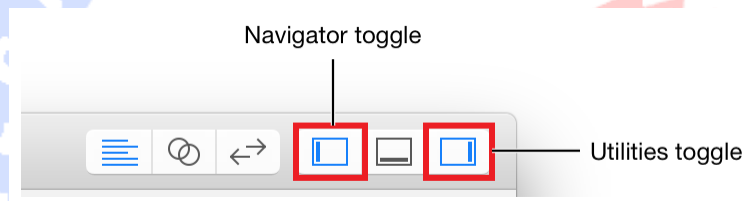
اپلیکیشن خود را در فواصل زمانی معین بررسی کرده و مطمئن شوید همه چیز مورد انتظار و دلخواه شما می باشد. به عنوان مثال جهت پیش مشاهده ی رابط کاربری برنامه ی خود می توانید از ابزاری به نام assistant editor بهره بگیرید. این ابزار یک ویرایشگر کمکی در کنار ویرایشگر اصلی Xcode به نمایش در می آورد که در آن می توانید نمایی از برنامه ی خود را مشاهده نمایید.

جهت پیش مشاهده ی رابط کاربری برنامه ی خود:

۱. بر روی دکمه ی اشاره شده در تصویر زیر کلیک نموده و ابزار assistant editor را باز کنید (این دکمه در نوار ابزار Xcode، بالای محیط گوشه سمت راست قابل مشاهده و دسترسی می باشد).

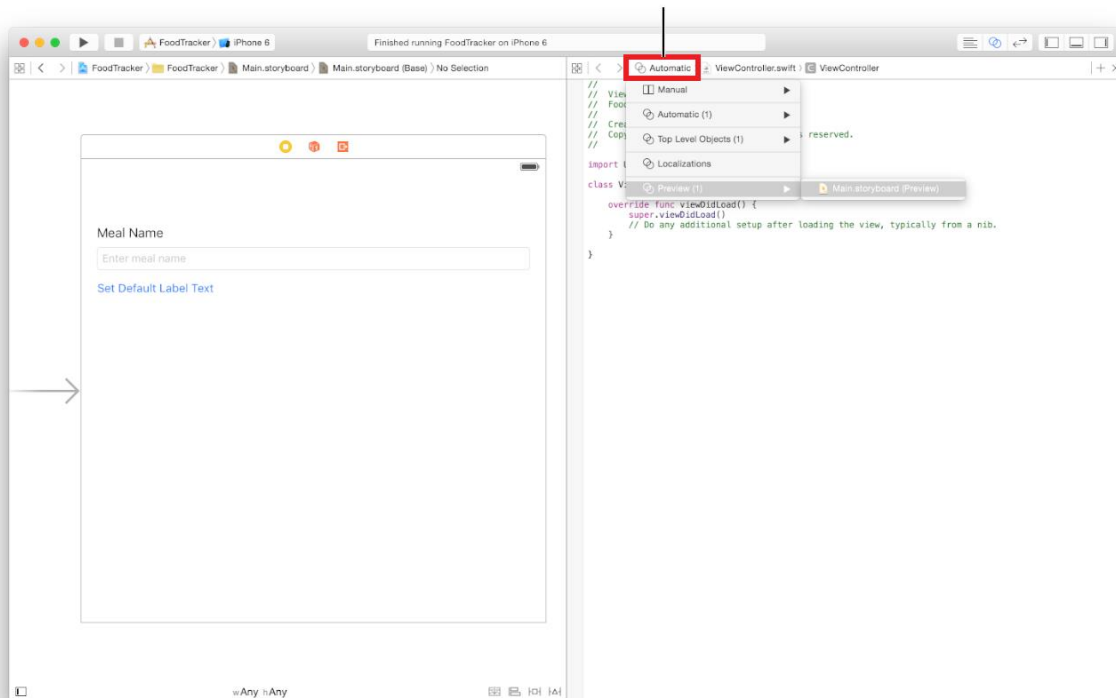


۲. اگر در محیط به فضای کاری بیشتری نیاز دارید، در آن صورت می توانید project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار محیط (اشاره شده در تصویر زیر) کوچک نمایید.

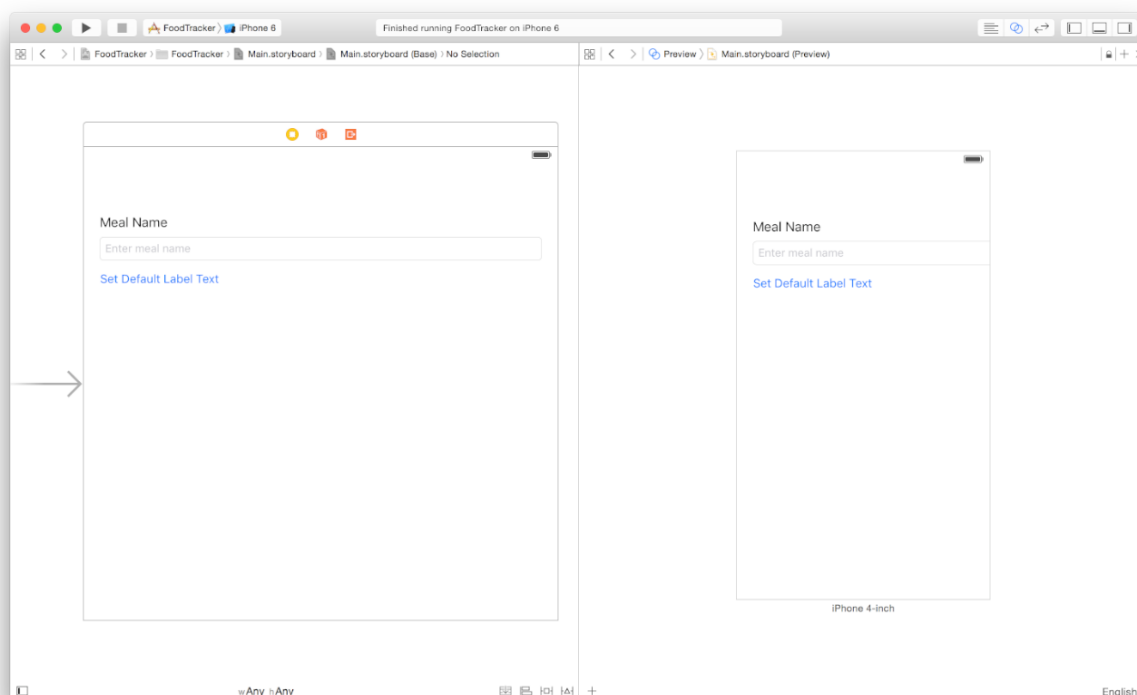


- در صورت نیاز می توانید outline view را نیز جمع نمایید.
۳. در editor selector bar، مقیم در بالای assistant editor (اشاره شده در تصویر زیر)، با طی کردن مراحل زیر، assistant editor را از حالت Automatic بر روی Preview تنظیم نمایید: Preview > Main.storyboard (Preview).

Editor selector bar



همان طور که در assistant editor مشاهده می کنید، طول المان text field از لبه ی صفحه ی جاری بیرون می زند. اما اگر به خاطر داشته باشید، رابط کاربری که در storyboard تعریف کردیم کاملاً بی نقص نمایش داده می شود. پس چرا همین رابط کاربری در حالت preview یا پیش نمایش iPhone متوازن نیست و به طور صحیح مقیاس دهی نمی شود؟



همان طور که قبلاً ذکر شد، رابط کاربری که طراحی می‌کنیم، در واقع یک interface انعطاف پذیر و قابل تنظیم است که متناسب با اندازه‌ی دستگاه میزبان (صفحه نمایش دستگاه‌های iPhone و iPad) مقیاس بندی می‌شود. scene یا صفحه‌ی محتوایی که به صورت پیش فرض در storyboard می‌بینید عملاً یک نما و نسخه‌ی کلی از UI برنامه‌ی شما را به نمایش می‌گذارد. اما در بستر assistant editor می‌بایست نحوه‌ی نمایش و مقیاس interface را برای دستگاه‌های مختلف (فضای موجود و اندازه‌ی صفحه نمایش مربوطه) در نظر گرفته و مشخص نمایید. برای مثال، زمانی که رابط کاربری بایستی خود را برای تطبیق با اندازه‌ی صفحه نمایش دستگاه iPhone کوچک کند، همان text field نیز باید همراه با آن کوچک شود و یا زمانی که می‌بایست جهت تطبیق با صفحه نمایش iPad بزرگ شود، همان مزبور باید متناسب با آن مقیاس بندی شود.

برای تعیین قوانین مقیاس دهی و تنظیم رابط کاربری متناسب با دستگاه، می‌توانید از موتور نمایش Auto Layout استفاده نمایید.

### استفاده از موتور نمایش Auto Layout

Auto Layout یک layout engine یا موتور نمایش و رندرینگ قدرتمند است که به شما امکان می دهد به آسانی layout (قالب های) انعطاف پذیر و قابل تنظیم طراحی نمایید. بدین صورت است که شما خواسته یا ایده ی خود را در خصوص موقعیت دهی المان ها در یک scene توصیف می کنید و سپس به layout engine واگذار می کنید خود تصمیم بگیرد چگونه به بهترین نحو این ایده را پیاده سازی کند. تمایل خود در خصوص چیدمان المان ها را بایستی در قالب constraint ها بیان کنید – constraint ها قوانینی هستند که مشخص می کنند المان ها چگونه باید نسبت به دیگر المان ها موقعیت دهی شوند، چه اندازه ای داشته باشند و در صورت کاهش فضای موجود، کدام المان اول کوچک شود.

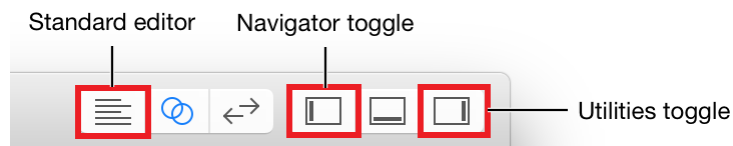
یکی دیگر از ابزار قدرتمندی که همراه با Auto Layout، در راستای مدیریت چیدمان و تنظیم اندازه ی المان متناسب با اندازه ی دستگاه (کار با layout) بکار می رود، stack view (کلاس UIViewStackView) می باشد. stack view یک رابط ساده و پربازده برای نمایش و چیدمان مجموعه ای از view ها در قالب ستون یا سطر ارائه می دهد. این ابزار به شما اجازه می دهد از قدرت Auto Layout بهترین استفاده را کرده و ال هایی طراحی کنید که به صورت پویا و در لحظه خود را با جهت (نمای افقی/عمودی)، اندازه ی صفحه نمایش و سایر تغییراتی که در فضای موجود ایجاد می شود، تطبیق دهد.

شما می توانید به آسانی رابط جاری را در یک stack view گنجانده، سپس با اعمال constraint های لازم اطمینان حاصل نمایید این stack view در دستگاه های مختلف به درستی نمایش داده می شود.

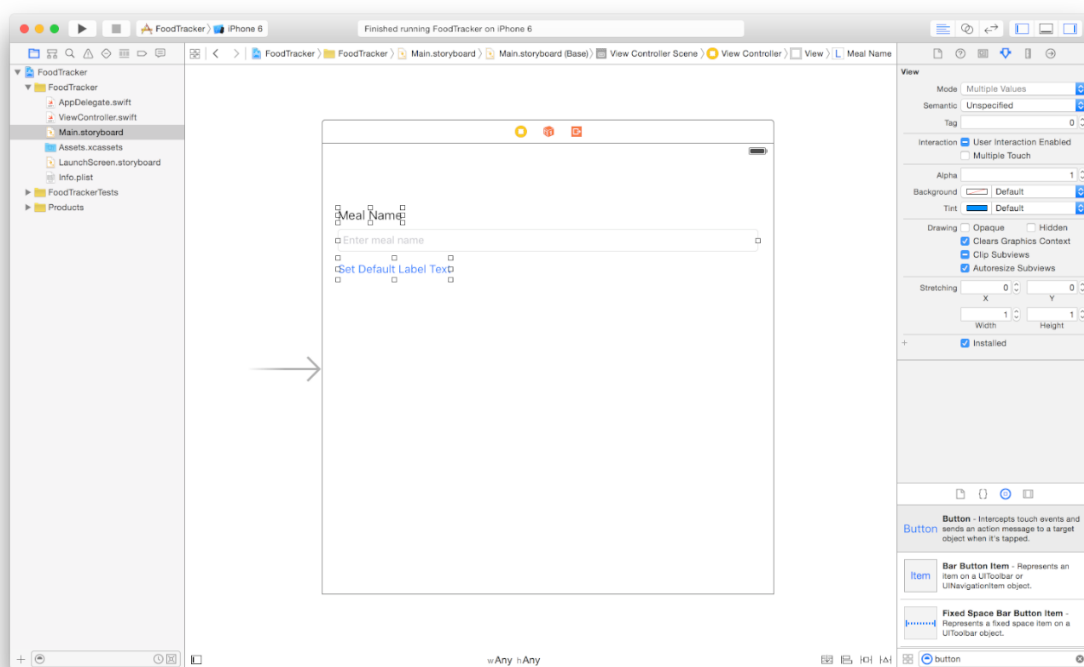
جهت افزودن constraint های مورد نیاز به scene جاری:

۱. ابتدا می بایست با کلیک بر روی دکمه ی Standard، اشاره شده در تصویر زیر،

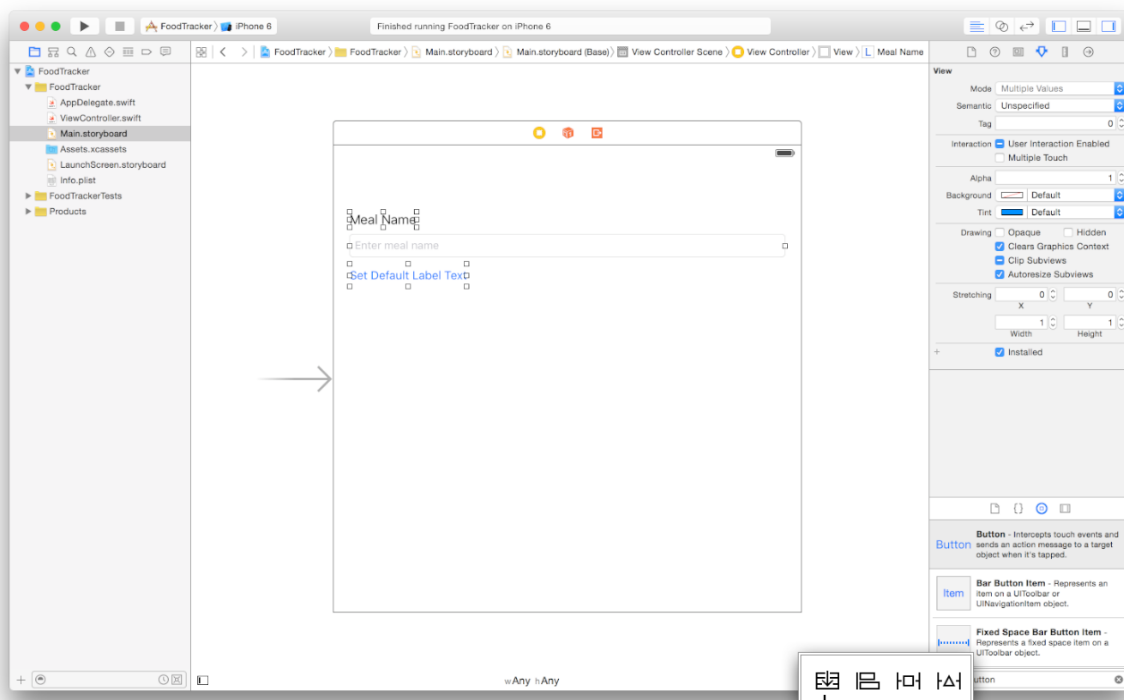
Standard editor را باز نمایید.



- با کلیک بر روی دکمه های Navigator و Utilities در نوار ابزار محیط Xcode، کادر project navigator و utility area را باز نمایید.
۱. کلید Shift را نگه داشته و سپس المان های text field، label و button را انتخاب نمایید.



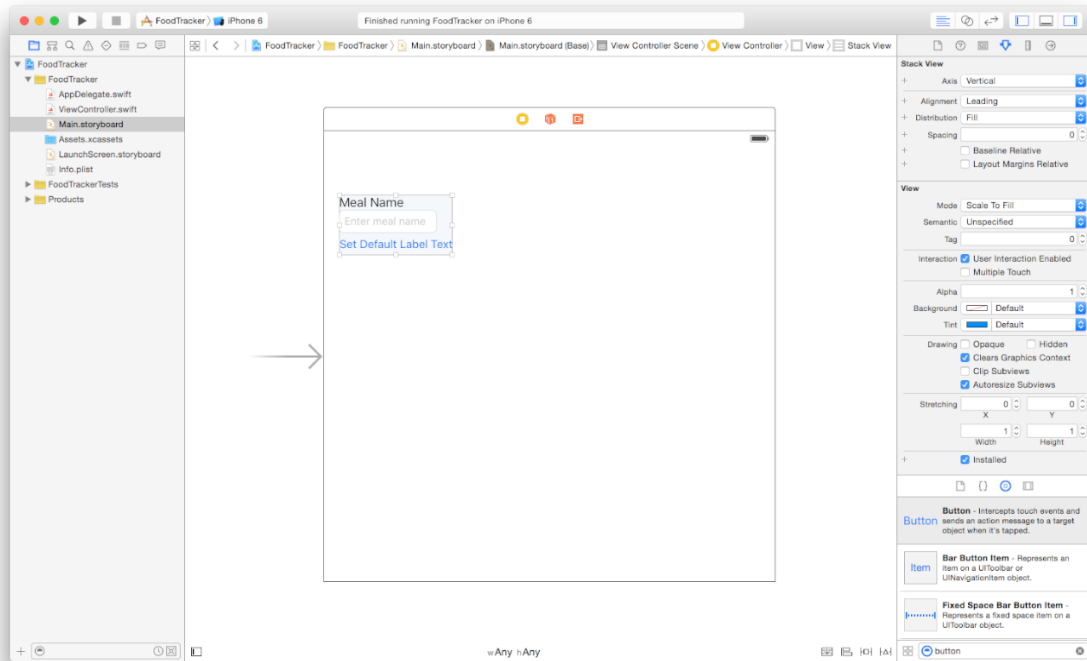
۲. در پایین canvas، سمت راست بر روی دکمه ی Stack کلیک نمایید (و یا این مسیر را طی کنید: Editor > Embed In > Stack View).



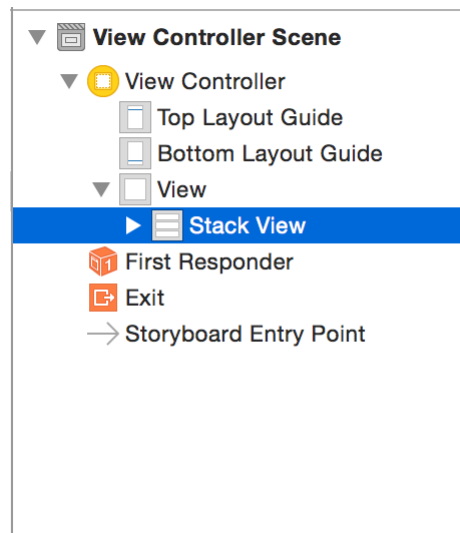
Stack

پس از کلیک بر روی دکمه ی مزبور، Xcode المان های UI را به صورت پشته بر روی هم قرار داده یا به اصطلاح آن ها را در قالب یک stack view می گنجاند (wrap می کند). با بررسی و تحلیل layout یا طرح بندی جاری، Xcode تصمیم می گیرد که المان ها را باید به صورت عمودی بر روی هم انباشته و نمایش دهد (نه به صورت افقی).

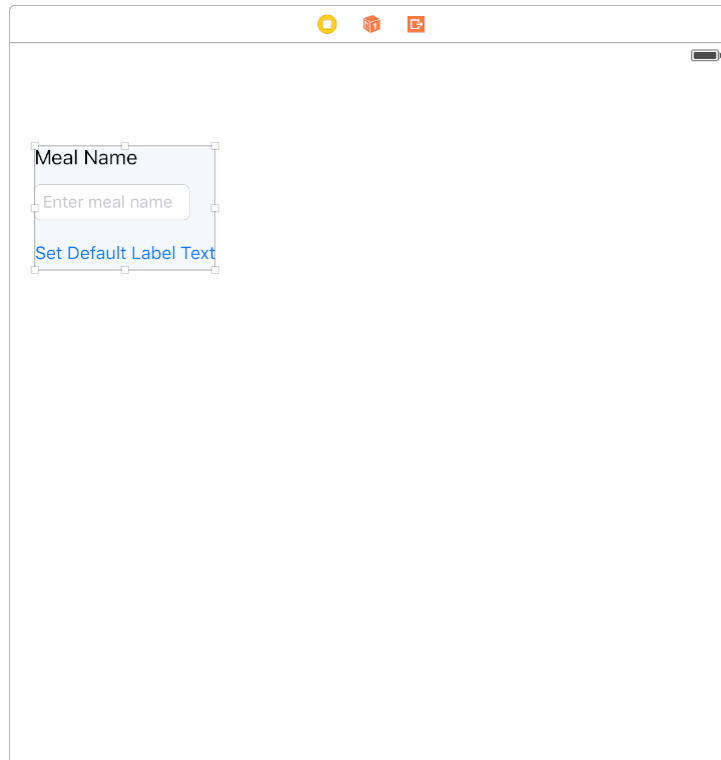




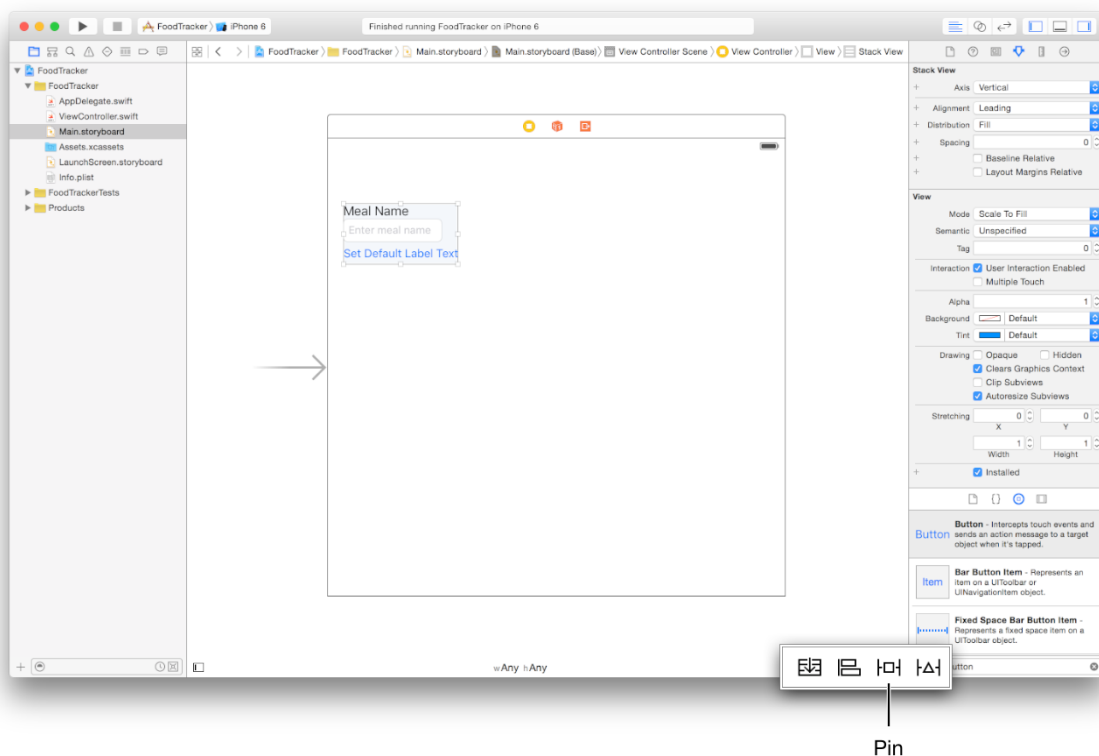
۳. در صورت لزوم، می توانید outline view را باز کرده، سپس آبجکت Stack View را انتخاب نمایید.



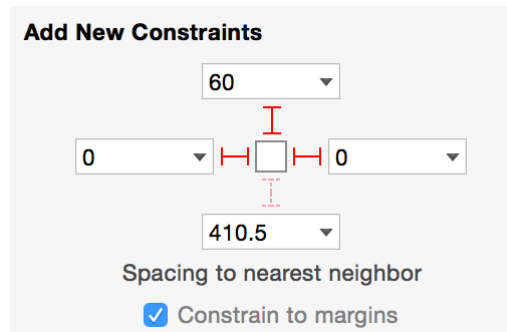
۴. در Attribute inspector، داخل فیلد Spacing مقدار ۱۲ را وارد نمایید. حال کلید Return را فشار دهید. خواهید دید که المان های UI به اندازه ی مقداری که در فیلد نام برده درج کردید، به صورت عمودی از هم فاصله داده شده و همزمان با آن stack view نیز بزرگتر می شود.



۵. در پایین canvas، سمت راست، با کلیک بر روی دکمه ی سوم (از سمت چپ) منوی Pin را باز نمایید.



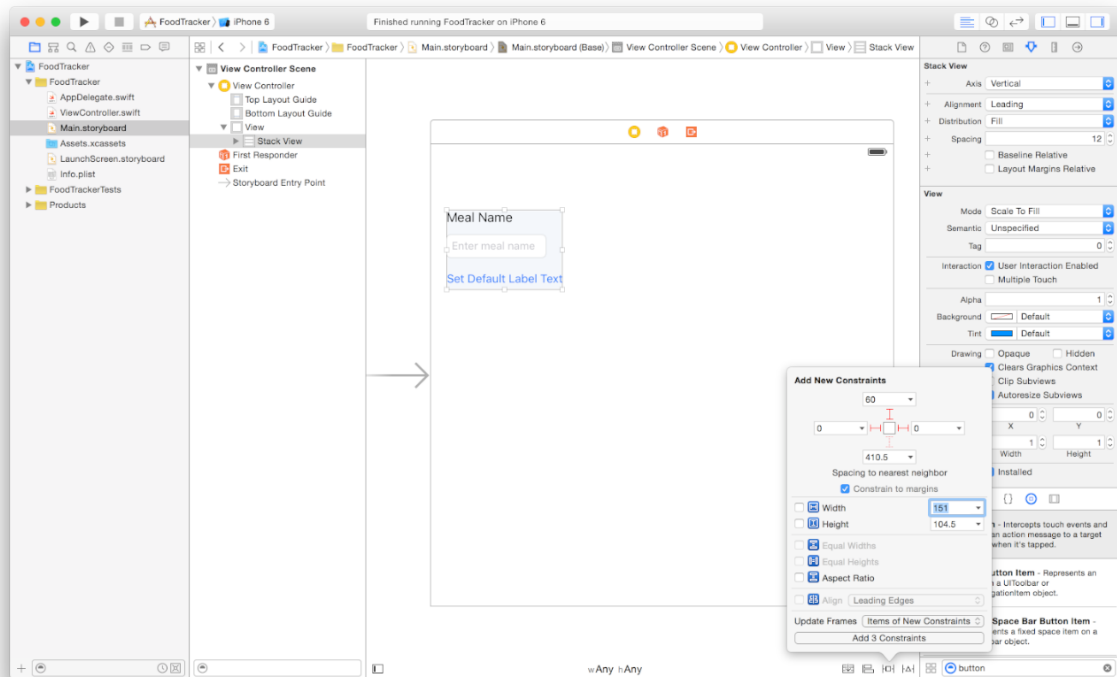
۶. در بالای عبارت “Spacing to nearest neighbor” در تصویر زیر، با کلیک بر روی دو constraint افقی و همچنین constraint عمودی سمت بالا، آن ها را انتخاب نمایید.
- constraint های نام برده پس از انتخاب قرمز رنگ می شوند.



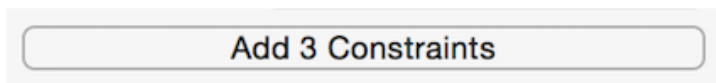
این Constraint ها در واقع نشانگر میزان فاصله المان جاری با المان های مجاور است. در اینجا منظور از اصطلاح nearest neighbor، لبه ی نزدیکترین المان UI به عنصر مورد نظر است که ممکن است superview، peer view یا صرفاً یک margin (حاشیه ی پس زمینه یا canvas) باشد. از آنجایی که گزینه ی “Constrain to margins” انتخاب شده، stack view به حاشیه ی سمت چپ، راست و بالای المان superview چسبانده شده و فاصله ی اندکی از لبه ی سمت چپ صفحه را خالی می گذارد.

۷. در کادرهای سمت چپ و راست مقدار ۰ و در کادر سمت بالا مقدار ۶۰ را وارد نمایید.

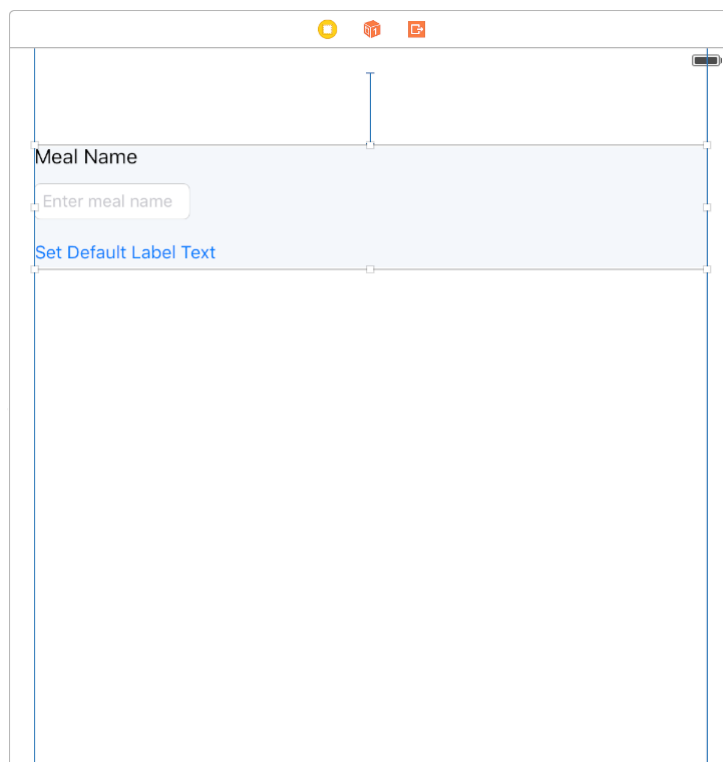
۸. از منوی pop-up جنب عبارت Update Frames، گزینه ی Items of New Constraints را انتخاب نمایید. منوی Pin هم اکنون می بایست ظاهری مشابه زیر داشته باشد:



۹. در منوی Pin، بر روی دکمه ی Add 3 Constraints کلیک نمایید.



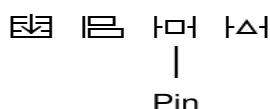
رابط کاربری برنامه هم اکنون باید چنین بنظر برسد:



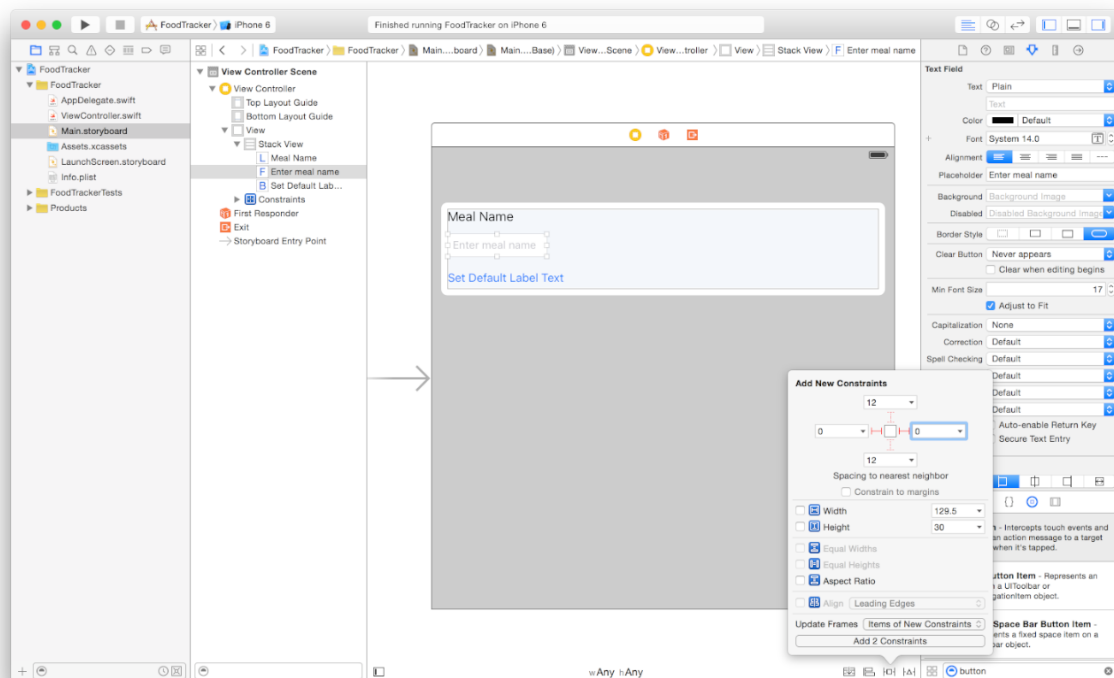
همان طور که می بینید که المان text field مانند قبل تا انتهای scene (لبه ی سمت راست) کشیده نمی شود. در زیر به تنظیم پهنای المان ذکر شده می پردازیم.

جهت تنظیم عرض المان text field در scene جاری:

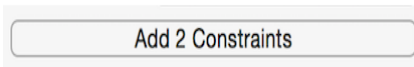
۱. (داخل storyboard) ابتدا المان مورد نظر را در scene فعلی انتخاب نمایید.
۲. در پایین canvas، سمت راست، با کلیک بر روی دکمه ی سوم (از سمت چپ) منوی Pin را باز نمایید.




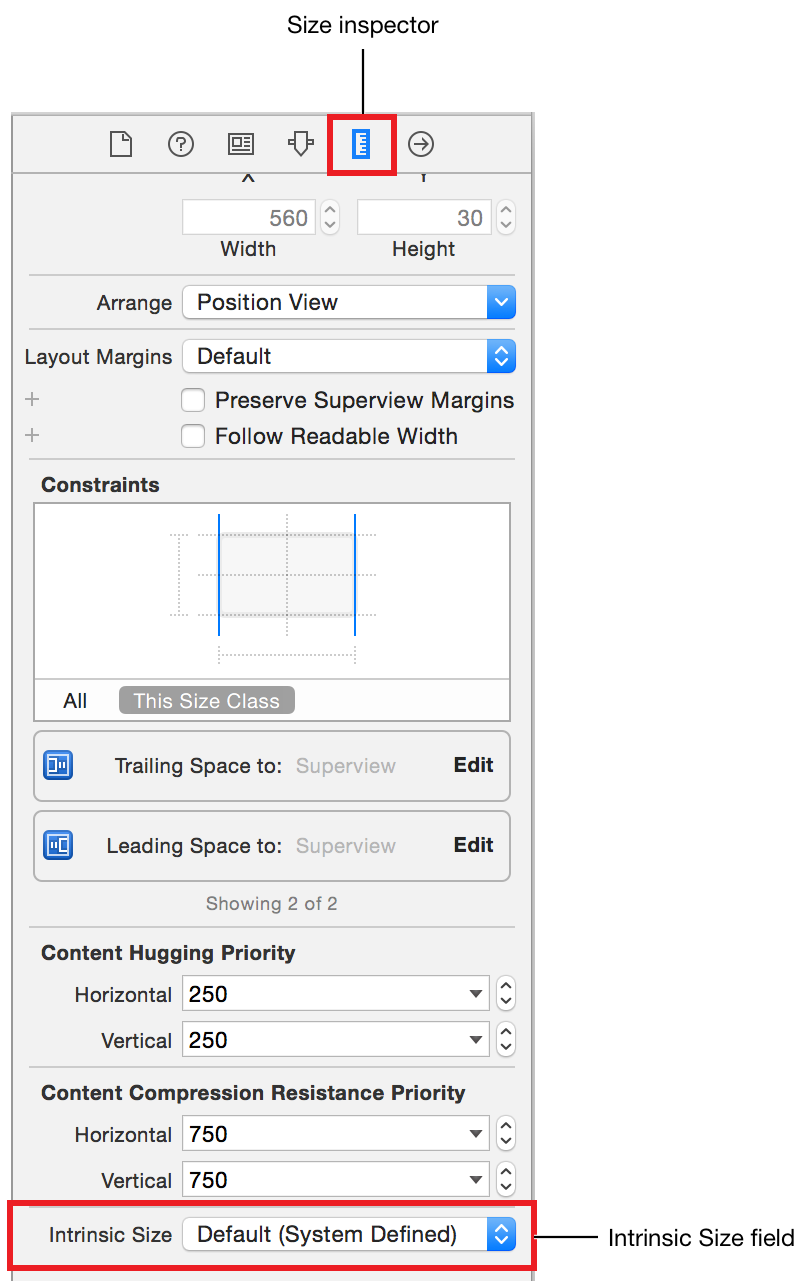
۳. در بالای عبارت "Spacing to nearest neighbor"، با کلیک بر روی دو constraint افقی، آن ها را انتخاب نمایید. همان طور که قبلاً گفته شد، constraint ها پس از انتخاب به رنگ قرمز پر رنگ در می آیند.
۴. داخل کادرهای سمت راست و چپ، مقدار ه را وارد نمایید.
۵. از منوی pop-up، جنب عبارت Update Frames، گزینه ی Items of New Constraints را انتخاب نمایید. در حال حاضر منوی Pin می بایست اینگونه به نظر برسد:



۶. در منوی Pin، بر روی دکمه ی Add 2 Constraints کلیک نمایید.



۷. در حالی که text field را انتخاب کرده اید، بر روی آیکون  در area utility کلیک نموده و Size inspector را باز نمایید. Size inspector به شما امکان می دهد، اندازه و مکان قرارگیری (position) آبجکت مورد نظر را در storyboard ویرایش و تنظیم نمایید.

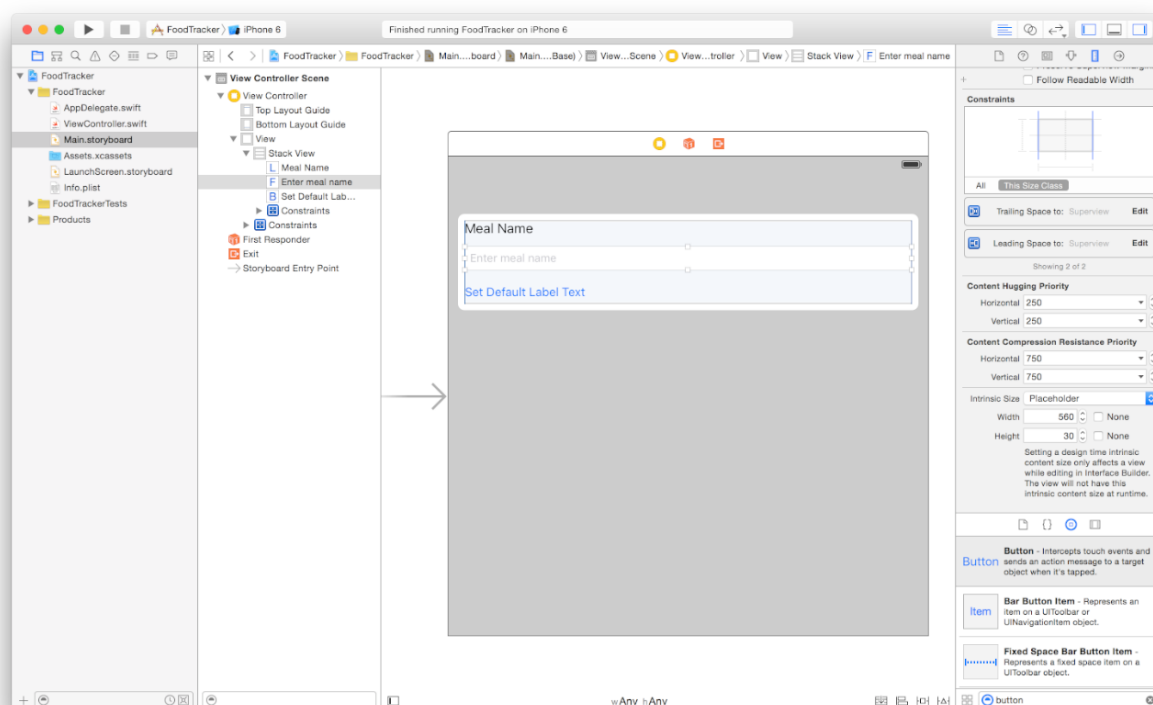


۸. از فیلد **Intrinsic Size**، گزینه **Placeholder** را انتخاب نمایید. (این فیلد در پایین **Size inspector** قرار گرفته، بنابراین جهت دسترسی به آن باید با نوار پیمایش به پایین کادر بروید.)

لازم به ذکر است که **Text field** بر اساس محتوای درونی خود اندازه بندی می شود. اندازه ی این المان بر اساس ویژگی **intrinsic content size** المان مشخص می شود (**intrinsic content size** به حداقل فضای لازم برای نمایش کامل و بی نقص تمامی محتویات **view** اشاره دارد).

در مواقعی که فکر می کنید اندازه ی المان ال ممکن است بزرگتر یا کوچکتر از مورد انتظار (در زمانی طراحی) باشد، می توانید فیلد `intrinsic size` آن را بر روی مقدار `placeholder` تنظیم نمایید. در حال حاضر، تنها محتوای `text field`، یک رشته ی مکان نگهدار (`placeholder string`) بوده که اندازه ی آن احتمالاً از طول رشته ای که کاربر وارد می کند، کم تر است.

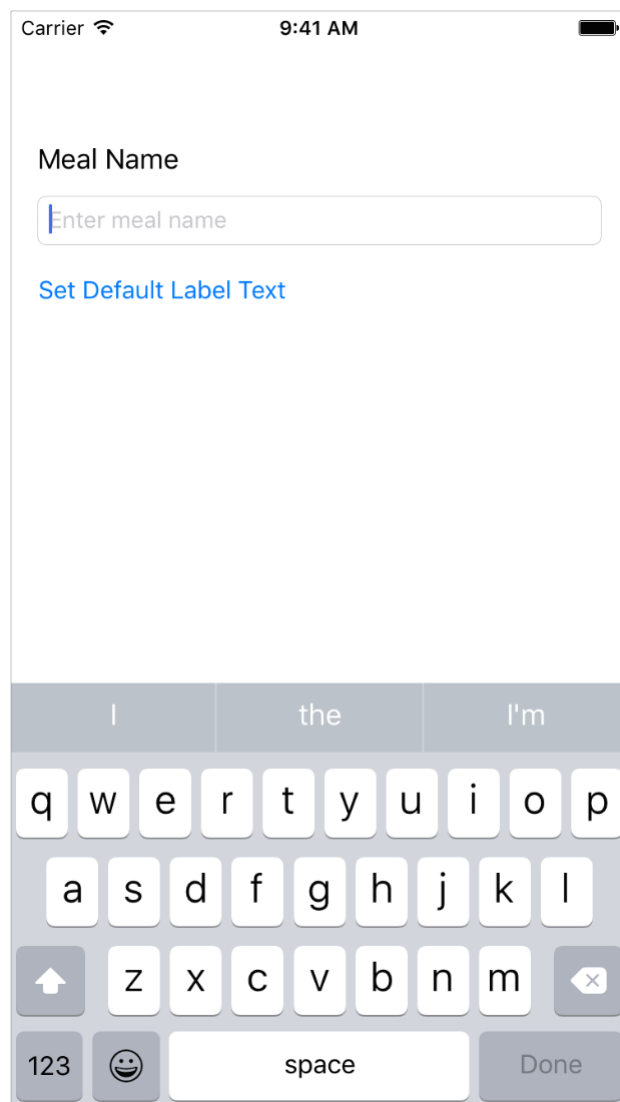
رابط کاربری برنامه در زمان حاضر می بایست مشابه زیر باشد:



اکنون برنامه ی خود را در محیط شبیه سازی Simulator اجرا کنید. المان `text field` قاعدتاً دیگر نباید از چارچوب صفحه ی جاری بیرون بزند. شما بایستی بتوانید داخل `text field` کلیک کرده و متن مورد نظر را با استفاده از صفحه کلید وارد نمایید (در صورت تمایل می توانید صفحه کلید نرم افزار را با زدن `Command-K` به صورت `toggle` نمایش داده/پنهان نمایید). حال اگر دستگاه را بچرخانید (با فشردن کلیدهای `Command-Left Arrow` یا `Command-Right Arrow`) و یا برنامه را بر روی دستگاه دیگری اجرا کنید، خواهید دید که `text field` متناسب با جهت و اندازه ی صفحه نمایش

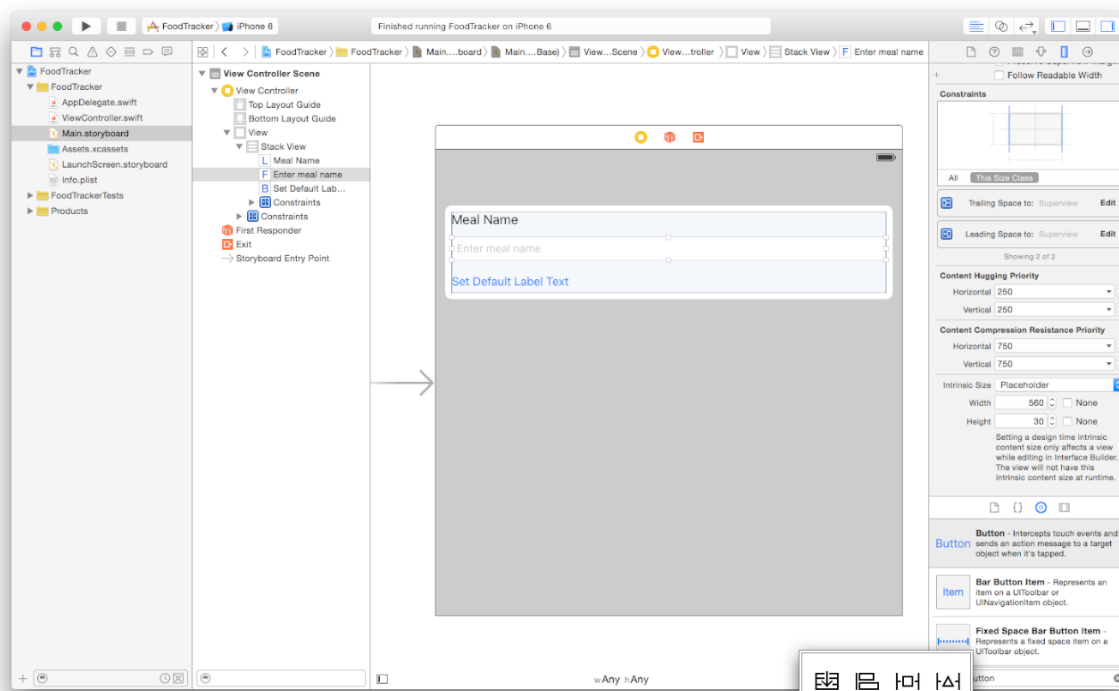


دستگاه میزبان، کوچک/بزرگ می شود. لازم به ذکر است که در نمای افقی (landscape orientation) نوار وضعیت/status bar از نظر کاربر محو می شود.



چنانچه اپلیکیشن رفتار مورد انتظار را ارائه نداد (المان ها را آن طور که باید نمایش نداد)، می توانید از امکانات debugging موتور Auto Layout جهت خطیابی و اشکال زدایی بهره بگیرید. برای این منظور بر روی آیکون Resolve Auto Layout Issues کلیک نموده، سپس گزینه ی Reset to Suggested Constraints را انتخاب نمایید تا Xcode رابط کاربری برنامه ی شما را بروز آوری کرده و constraint های مناسب را اعمال کند.

یا پس از کلیک بر روی آیکن **Resolve Auto Layout Issues**، گزینه **Clear Constraints** را انتخاب نمایید تا تمامی **constraint** های اعمال شده بر روی المان های **UI** حذف گشته و سپس تمامی مراحل فوق را مجدداً دنبال نمایید.



Resolve Auto Layout Issues

**scene** جاری به جز نمایش تعداد محدودی المان **UI**، کار به خصوصی انجام نمی دهد. با این حال نمایی از یک رابط کاربری ساده و کاربردی ارائه می دهد که می تواند شما را در طراحی **UI** برنامه ی خود راهنمایی کند.

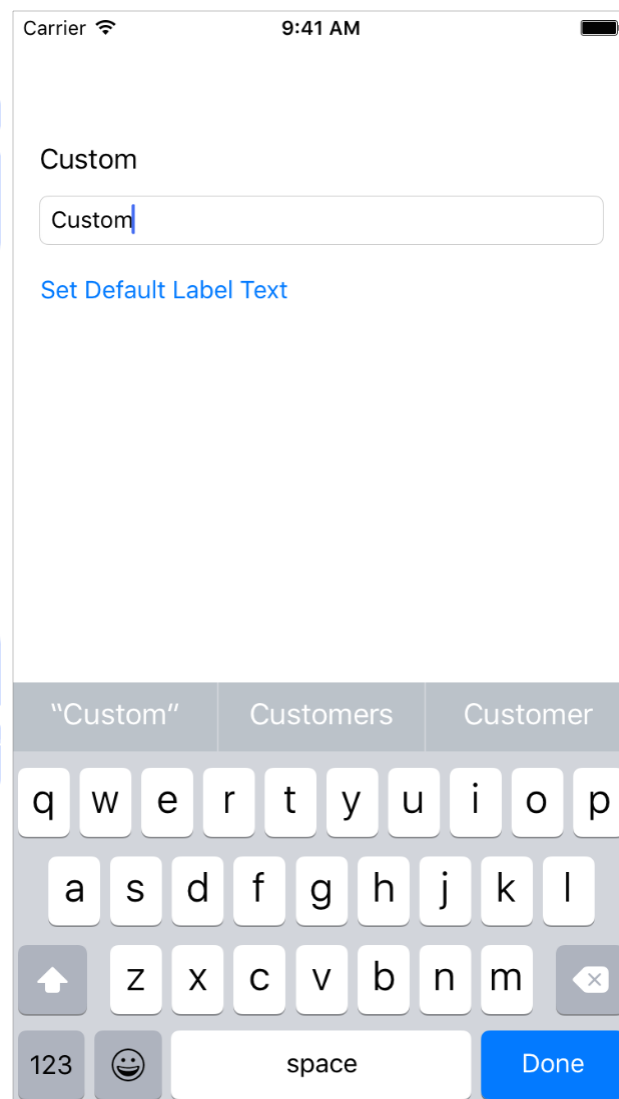
ساخت یک **layout** (قالب و نمای کلی) کارآمد و قابل تنظیم (که متناسب با عرض صفحه ی دستگاه میزبان و جهت نمایش، بزرگ/کوچک می شود) می تواند پایه ی قوی برای برنامه باشد. شما می توانید بر اساس این پایه ی قوی نسبت به ساخت دیگر بخش های برنامه اقدام نمایید.

## درس ۴ : اتصال **UI** به کد **Swift**

متصل کردن **UI** (ظاهر یا رابط کاربری برنامه) به کد (رفتارها و عملیاتی که برنامه انجام می دهد)

در این مبحث، شما UI ساده ی اپلیکیشن FoodTracker (یا ثبت و رصد اطلاعات غذا) را به کد برنامه متصل می نمایید. با استفاده از این کد امکان انجام عملیات خاصی بر روی UI را برای کاربر برنامه ی خود فراهم می کنید.

در پایان برنامه ظاهری مشابه زیر خواهد داشت:



آنچه خواهید آموخت

- رابطه ی بین یک scene در storyboard و view controller زیرین آن را درک کرده و توضیح دهید.

- با استفاده از outlet و action بین المان های UI در سطح storyboard و کد برنامه (source code) ارتباط برقرار کنید.
- ورودی کاربر از یک کادر متن یا text field را خوانده، پردازش کنید و سپس خروجی آن را در UI اپلیکیشن به نمایش بگذارید.
- یک کلاس را بر اساس protocol خاص پیاده سازی کنید (property ها و متدهای تعیین شده توسط آن protocol را در کلاس پیاده سازی کنید).
- با الگوی توسعه ی delegation آشنا شوید.
- در طراحی معماری اپلیکیشن از الگوی توسعه ی target-action پیروی نمایید (الگوی توسعه ی target-action را در معماری اپلیکیشن پیاده سازی کنید).

#### متصل کردن UI به کد برنامه (Source Code)

المان هایی که در storyboard مشاهده می کنید، در واقع هریک به بخشی از source code وصل است. شما باید این رابطه ی بین storyboard با کد برنامه را به خوبی درک کنید.

در Storyboard، یک scene نشانگر صفحه ای از صفحات متعدد برنامه است که بخشی از محتوای برنامه و عموماً یک view controller را شامل می شود. view controller ها رفتار (متدهای) اپلیکیشن شما را پیاده سازی می کنند.

view controller تنها یک content view و view های زیرمجموعه ی آن را اداره می کند (view = content view دربرگیرنده ی تمامی view ها یا آبجکت های یک scene همچون button، text field و label).

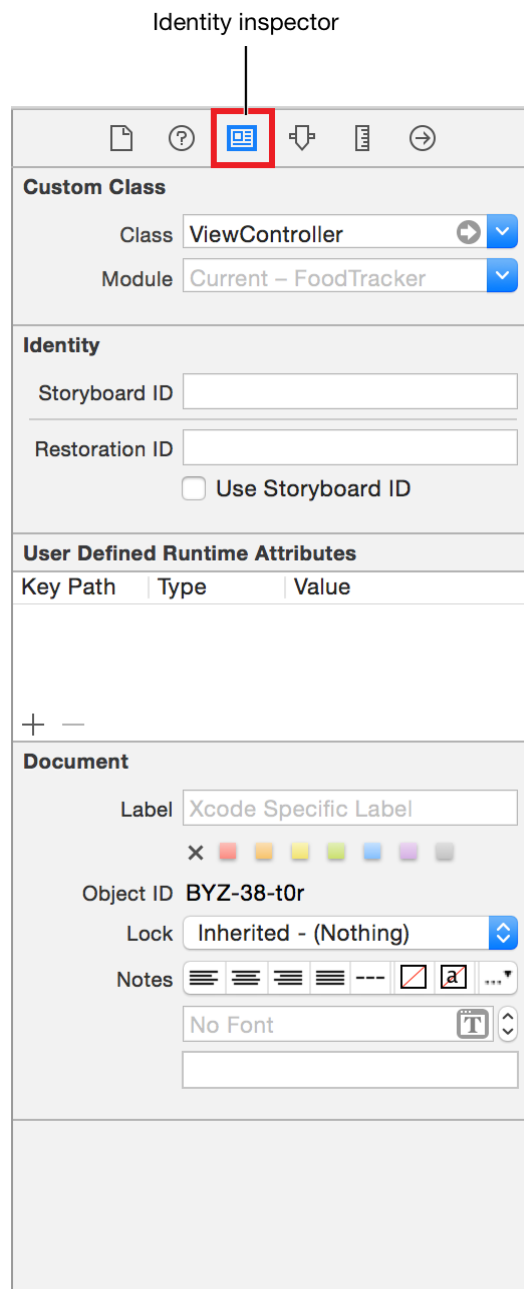
View controller ها: ۱. تعاملات/گردش اطلاعات بین data model برنامه (بخشی که داده های اپلیکیشن را کپسوله سازی می کند) و view ها را (بخش هایی که داده های برنامه را برای کاربر به نمایش می گذارد) مدیریت می کند. ۲. چرخه ی حیات content view های برنامه (view ای که در بالای سلسله مراتب قرار گرفته و نقش ظرف را برای view های فرزند خود همچون button، label ایفا می کند) را از ابتدا تا انتها مدیریت می

نماید ۳. تغییرات در جهت نمایش برنامه را به هنگام چرخش دستگاه اداره می کند ۴. قابلیت پیمایش در برنامه را تعریف کرده ۵. و در پایان رفتاری که به ورودی کاربر واکنش نشان می دهد را پیاده سازی می نماید.

تمامی آبجکت های view controller در iOS، از جنس کلاس UINavigationController یا یکی از کلاس های فرزند آن هستند.

رفتار view controller ها را بایستی در کد با ایجاد و پیاده سازی subclass های اختصاصی از view controller (کلاس های فرزند view controller با پیاده سازی اختصاصی خود)، تعریف نمایید. سپس می توانید یک اتصال بین کلاس های مذکور و scene ها در storyboard ایجاد کرده و در نهایت برنامه ای داشته باشید که رفتار تعریف شده در کد (کلاس های view controller) را همراه با UI مورد نظر در storyboard، ارائه می دهد.

Xcode قبلاً چنین کلاسی را با نام ViewController.swift ایجاد کرده و آن را به scene جاری که هم اکنون در storyboard با آن کار می کنید، متصل کرده است. در آینده، همین که scene های بیشتری را به برنامه اضافه می کنید، این اتصال بین scene و view controller را خود از طریق Identity inspector برقرار می نمایید. Identity inspector این امکان را به شما می دهد تا آن دسته از property های یک آبجکت را که با identity آن مرتبط هستند، مانند اینکه به کدام کلاس تعلق دارد، ویرایش نمایید.



در زمان اجرای برنامه (runtime)، storyboard نمونه ای از کلاس ViewController می سازد (همان subclass اختصاصی view controller). صفحه ای از برنامه را که در storyboard می بینید، رابط کاربری و ظاهر خود را از scene جاری گرفته، و رفتارهایش، عملیاتی که می تواند انجام دهد، را از کدهای فایل ViewController.swift دریافت می کند.

اگرچه scene مورد نظر به کدهای فایل ViewController.swift متصل است، اما این اتصال به تنهایی برای داشتن تعامل بین بخش های مختلف برنامه کافی نیست. برای

تعریف تعامل در برنامه ی خود، کد view controller بایستی بتواند با view های موجود در storyboard (آبجکت های text field، label) تبادل اطلاعات داشته باشد. این کار را با ایجاد connection های بیشتر – که outlet و action خوانده می شوند – بین view ها در storyboard و فایل های view controller انجام می دهید.

### تعریف Outlet برای المان های UI

Outlet روشی تعریف می کند که می توان به وسیله ی آن به آبجکت های رابط کاربری – آبجکت هایی نظیر label، button که به storyboard اضافه شده – از کد فایل های source code اشاره کرد (دسترسی داشت).

برای ایجاد outlet، کافی است از آبجکت مورد نظر در storyboard، به فایل view controller، control-drag کنید. بدین معنی که با کلیک بر روی آن آبجکت، نگه داشتن کلید control و کشیدن آن به فایل view controller، یک اشاره گر از آن آبجکت در فایل view controller ایجاد نمایید.

این عملیات یک (متغیر) property برای آبجکت مورد نظر در فایل view controller ایجاد می کند و به شما اجازه می دهد در زمان اجرا از کد به آن آبجکت دسترسی داشته و آن را دستکاری کنید.

با توجه به آنچه گفته شد، جهت دسترسی یا اشاره به label و text field در لایه ی رابط کاربری، باید برای آن ها در کد outlet تعریف نمایید.

به منظور وصل کردن المان text field به کد فایل ViewController.swift:

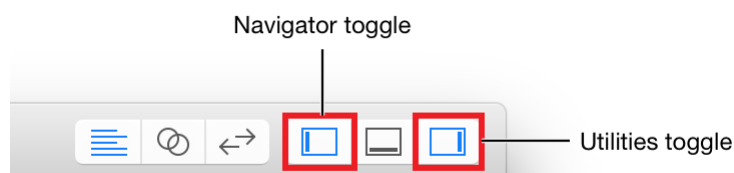
۱. فایل Main.storyboard را باز کنید.

۲. Assistant editor را با کلیک بر روی دکمه ی Assistant، واقع در نوار ابزار Xcode، بالای

محیط گوشه ی سمت راست، باز نمایید.

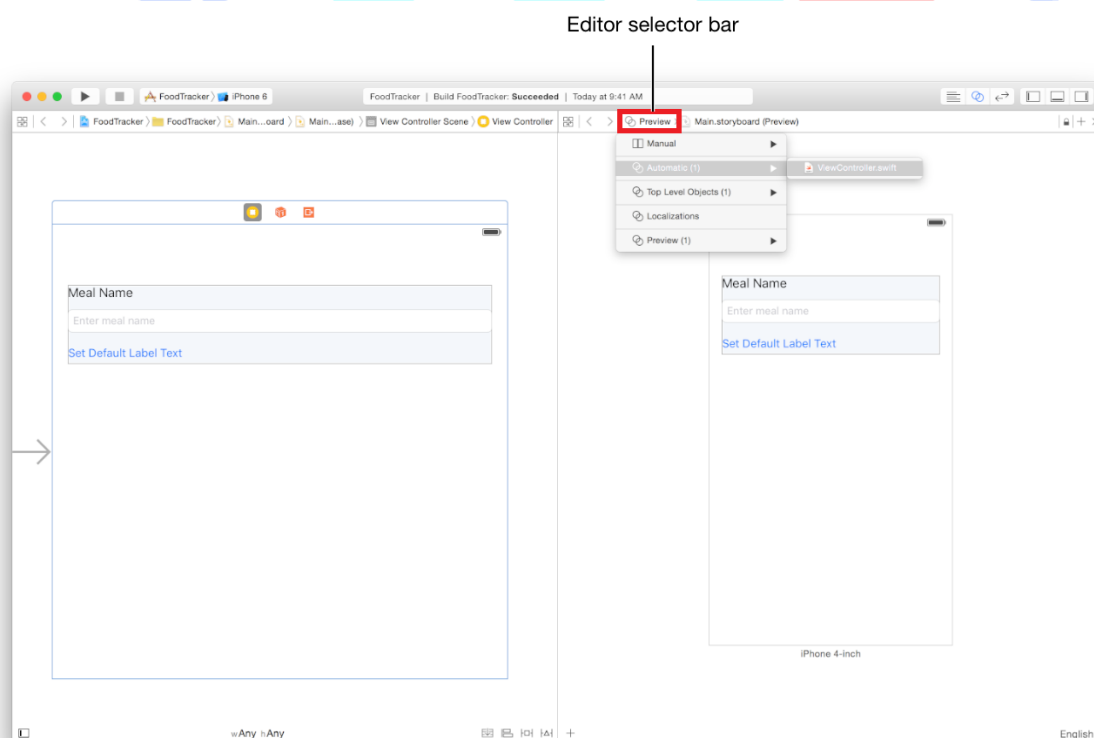


۳. اگر به فضای کاری بیشتری نیاز دارید، می توانید project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار Xcode پنهان نمایید (جمع کنید).



در صورت لزوم می توانید **outline view** را نیز جمع کنید.

۴. در editor selector bar، در بالای assistant editor، این مسیر را طی کنید: - Preview -> Automatic -> viewController.swift



محتویات فایل ViewController.swift در ویرایشگر سمت راست محیط به نمایش در می آید.

۱. در فایل ViewController.swift، خطی که کلیدواژه ی class را دارد، پیدا کنید:

۲. در زیر کد ذکر شده، عبارت ذیل را درج کنید:

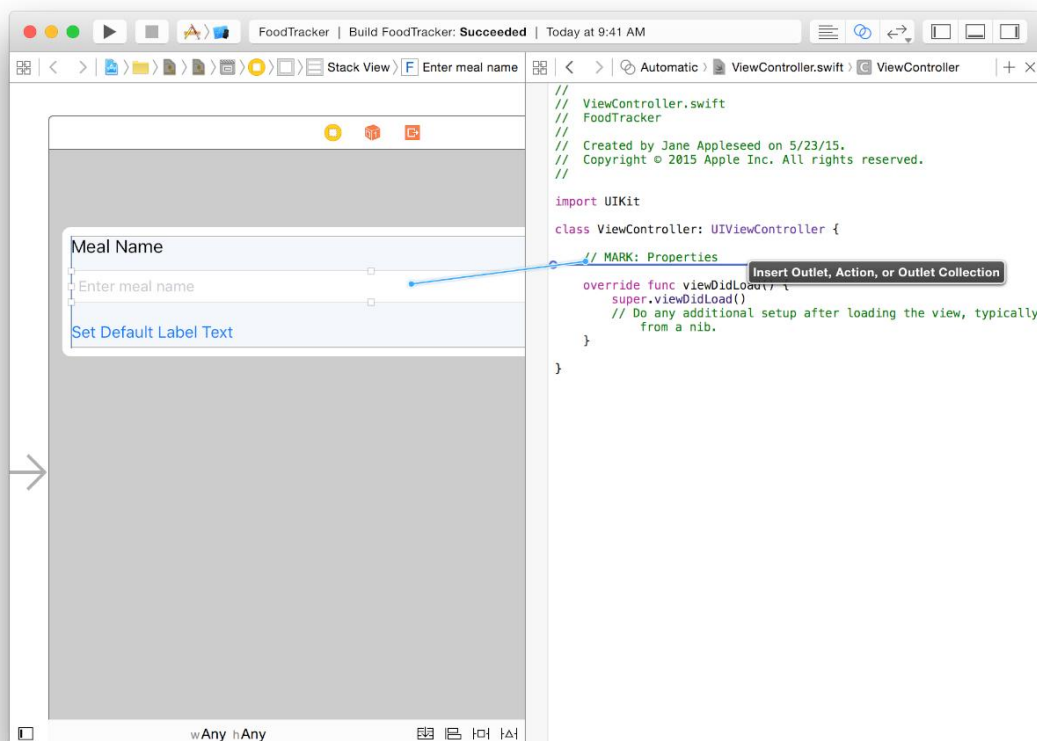
**// MARK: Properties**



با این کار در واقع یک comment (کد درج توضیحات) به source code خود اضافه می کنید. همان طور که قبلاً گفته شد، comment یک تکه متن در فایل source code است که به هنگام کامپایل به عنوان بخشی از دستورات برنامه ترجمه و اجرا نمی شود، اما اطلاعات کاربردی در خصوص بخش های مختلف کد ارائه می دهد.

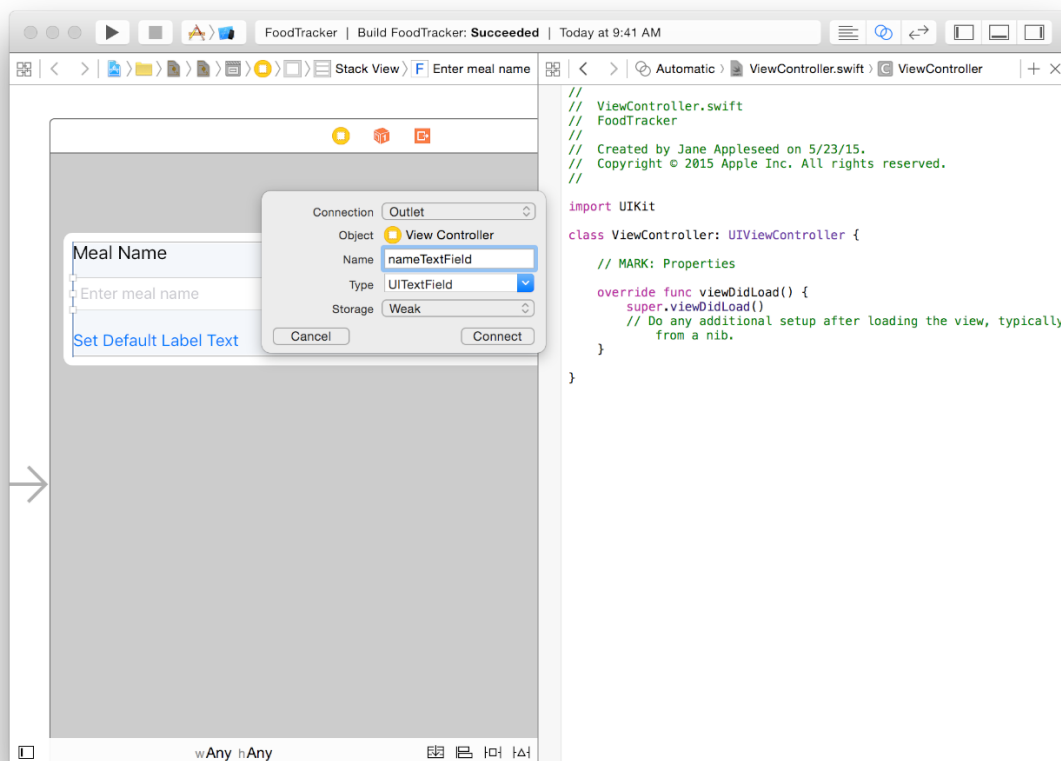
Comment ای که با کاراکترهای MARK: // آغاز می شود، یک نوع خاص comment است که بیشتر به منظور سازمان دهی کد و کمک در فهم کاربرد آن بخش مورد استفاده قرار می گیرد.

- به عنوان نمونه، تکه کدی که شما به عنوان comment به فایل نام برده اضافه کردید، اعلان می کند که در آن بخش property های آبجکت لیست می شوند.
۱. در storyboard، المان text field را انتخاب کنید.
  ۲. بر روی المان مذکور در canvas کلیک کرده، کلید control را نگه دارید، سپس آن را کشیده و در زیر بخش comment در assistant editor جایگذاری نمایید.



۳. یک پنجره ی محاوره ای به نمایش در می آید. در فیلد Name، واژه ی nameTextField را وارد کنید.

لازم نیست دیگر تنظیمات را تغییر دهید.



۴. حال بر روی دکمه ی Connect کلیک نمایید. Xcode کد لازم را به فایل ViewController.swift، جهت ذخیره ی اشاره گری (pointer) به text field اضافه کرده و storyboard را برای برقراری اتصال مورد نیاز بین المان UI و کد تنظیم می کند. به عبارت واضح تر، Xcode یک اشاره گر (در قالب متغیر یا property) به المان UI مورد نظر، در فایل ViewController.swift اضافه کرده و سپس storyboard را جهت برقراری ارتباط بین المان مورد نظر در لایه ی رابط کاربری با کد مربوطه ی آن در فایل مزبور تنظیم می کند.

**@IBOutlet weak var nameTextField: UITextField!**

مدت زمانی را به فهم این خط کد اختصاص دهید.

خصیصه (attribute) IBOutlet این امکان را برای شما فراهم می کند تا در محیط Xcode از interface builder به متغیر (property) nameTextField وصل شوید (پیشوند IB نیز به همین امکان اشاره دارد). کلیدواژه ی weak بیانگر قابلیت nil بودن متغیر در طول عمر آن می باشد (بدین معنی که property مذکور می تواند در برهه ای از زمان هیچ مقداری نداشته باشد). در بقیه ی (تعریف) کد صرفاً یک متغیر از نوع (کلاس) UITextField، به نام nameTextField ایجاد می شود.

به علامت تعجب "!" در انتهای تعریف outlet دقت کنید. اگر بخاطر داشته باشید، برخی از property ها در فایل AppDelegate.swift این علامت را در انتهای خود دارند. علامت مزبور نشانگر این است که متغیر یا property مورد نظر از نوع implicitly unwrapped optional است. implicitly unwrapped optional، یک متغیر از جنس optional است که پس از مقداردهی اولیه همیشه حاوی مقدار خواهد بود (optional متغیری که می تواند مقداری داشته/نداشته باشد).

حال label را مانند المان قبلی (text field) به کد مربوطه ی آن در فایل ViewController.swift متصل کنید.

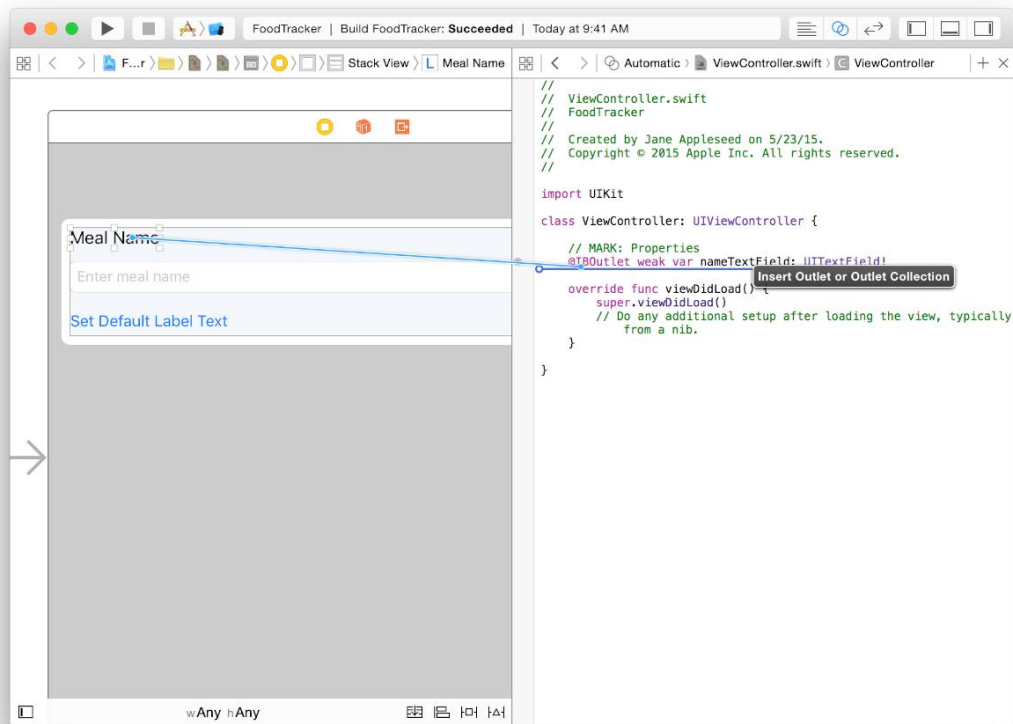
برای وصل کردن المان label به کد مربوطه در فایل ViewController.swift:

۱. Label را در storyboard انتخاب نمایید.

۲. بر روی المان مذکور در canvas کلیک کرده و کلید Control را نگه دارید، سپس

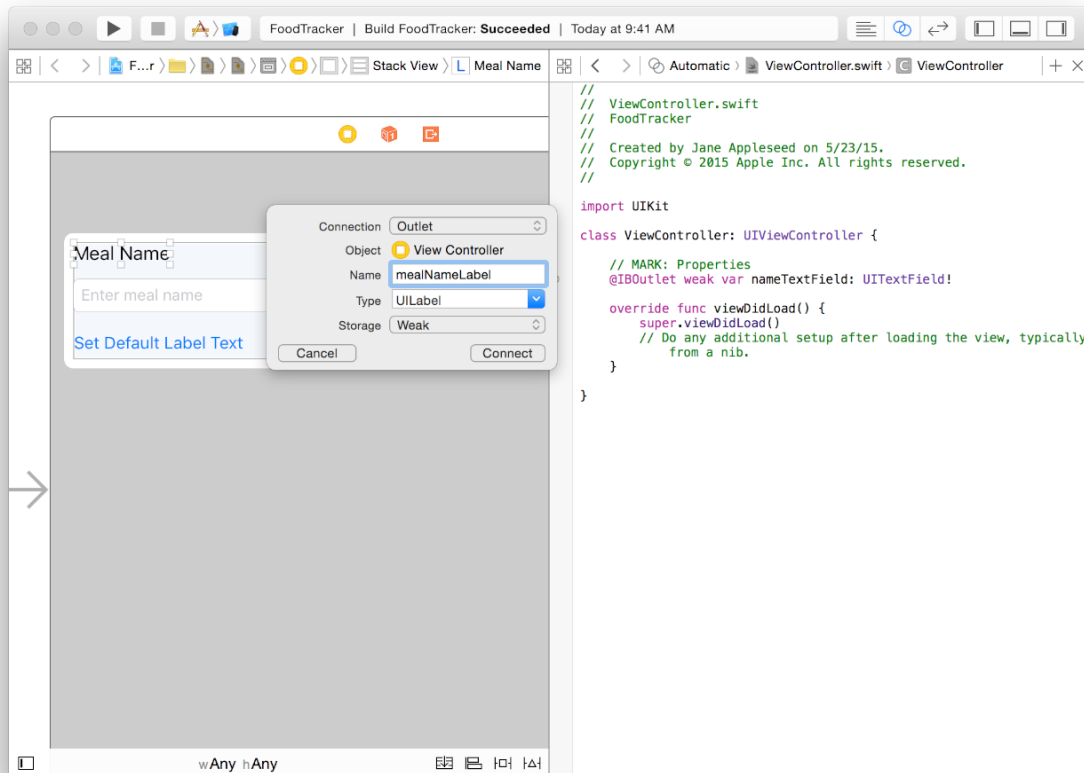
آن را کشیده و در assistant editor، زیر متغیر nameTextField جایگذاری

کنید.



۳. در کادر محاوره ای که نمایان می شود، در فیلد Name، واژه ی mealNameLabel را به عنوان اسم المان وارد نمایید. لازم نیست تنظیمات دیگری انجام دهید.

آموزشگاه تحلیکیر داده ها



#### ۴. بر روی Connect کلیک نمایید.

Xcode یک اشاره گر (pointer) در قالب متغیر به المان Label در فایل ViewController.swift اضافه کرده، سپس storyboard را جهت برقراری ارتباط بین المان UI مورد نظر و کد مربوطه ی آن در فایل مذکور، تنظیم می کند. این outlet به استثنای دو ویژگی نوع و اسم، کاملاً مشابه المان قبلی است (outlet جاری از نوع کلاس UILabel است که با نوع آبجکت در storyboard کاملاً همخوانی دارد).

**@IBOutlet weak var mealNameLabel: UILabel!**

حالا که روشی برای اشاره و دسترسی به المان های UI از داخل کد (در فایل ViewController.swift) دارید، باید یک رخداد که توسط کاربر فعال می شود و زمینه ی تعامل میان این دو المان را فراهم می کند، تعریف نمایید. برای تعریف این رخداد در IOS از action بهره می گیریم.

## تعریف یک رخداد با استفاده از Action

اپلیکیشن های IOS رخداد محور هستند (بر اساس الگوی برنامه نویسی رخداد محور/event-driven طراحی می شوند). بدین معنی که روند اجرای (flow) برنامه توسط event ها: رخدادهای سیستم و action های کاربر (کلیک موس یا فشردن کلیدهای کیبورد یا دکمه ای بر روی صفحه) تعیین می گردد.

کاربر با UI برنامه تعامل برقرار کرده، برای مثال یک دکمه را کلیک می کند، در پی این عمل/action کاربر رخداد خاصی اتفاق می افتد. این رخدادها خود سبب اجرای منطق برنامه و تغییر در داده های آن می شود. سپس واکنش برنامه به عمل کاربر و اثر آن، در UI برنامه منعکس می شود.

همان طور که می دانید در برنامه های رویداد محور، کنترل اجرای بخش هایی از برنامه به دست کاربر است. بدین معنی که زمانی که کاربر با UI برنامه تعامل می کند، بخش های مختلف از کد برنامه در پاسخ به عمل کاربر اجرا می شود. حال از آنجایی که کاربر، نه شما، کنترل اینکه چه زمانی بخش هایی از کد برنامه اجرا شوند را در دست دارد، شما به عنوان برنامه نویس بایستی مشخص کنید کاربر دقیقاً اجازه ی انجام چه عملیاتی را دارد و در پاسخ به آن اعمال دقیقاً چه واکنشی بایستی نشان داده شود (چه اتفاقی رخ دهد).

Action (یا action method) یک تکه کد است که به event یا رخدادی در برنامه وصل بوده و با اتفاق افتادن آن event فراخوانی و اجرا می شود. به عبارتی دیگر زمانی که آن event رخ می دهد، متعاقباً کد یا دستور مرتبط با آن اجرا می گردد.

می توانید با تعریف action، عملیات دلخواه را پیاده سازی کنید. حال این عملیات می تواند دستکاری داده های مشخص، بروز رسانی بخشی از UI و یا هر عملیات دیگری باشد. در واقع شما با استفاده از action جریان یا روند اجرای برنامه را در پاسخ به رخدادهای فعال شده توسط کاربر و سیستم تعیین می کنید.

تعریف action و outlet به یک نحو صورت می گیرد: کافی است بر روی آبجکت مورد نظر کلیک کرده، کلید control را نگه دارید. سپس آن را کشیده و داخل فایل view controller جایگذاری نمایید. به دنبال آن یک متد در فایل view controller تعریف می شود که به مجرد تعامل کاربر با آبجکت متصل (به آن متد)، فراخوانده و اجرا می گردد.

کار را با ایجاد یک action ساده آغاز نمایید: زمانی که کاربر در UI برنامه، دکمه ی Set Default Label Text را فشار می دهد، شما باید label را طوری تنظیم کرده باشید که (به مجرد فشردن دکمه) مقدار پیش فرض Default Text را نمایش دهد. (در بخش بعدی label را طوری تنظیم می کنیم که مقدار وارد شده در text field را عیناً نمایش دهد.)

جهت ایجاد یک action برای بازگردانی مقدار label به Default Text در فایل ViewController.swift، مراحل زیر را گام به گام دنبال نمایید:

۱. در فایل ViewController.swift، به قبل از آخرین "}"، عبارت زیر را اضافه نمایید:

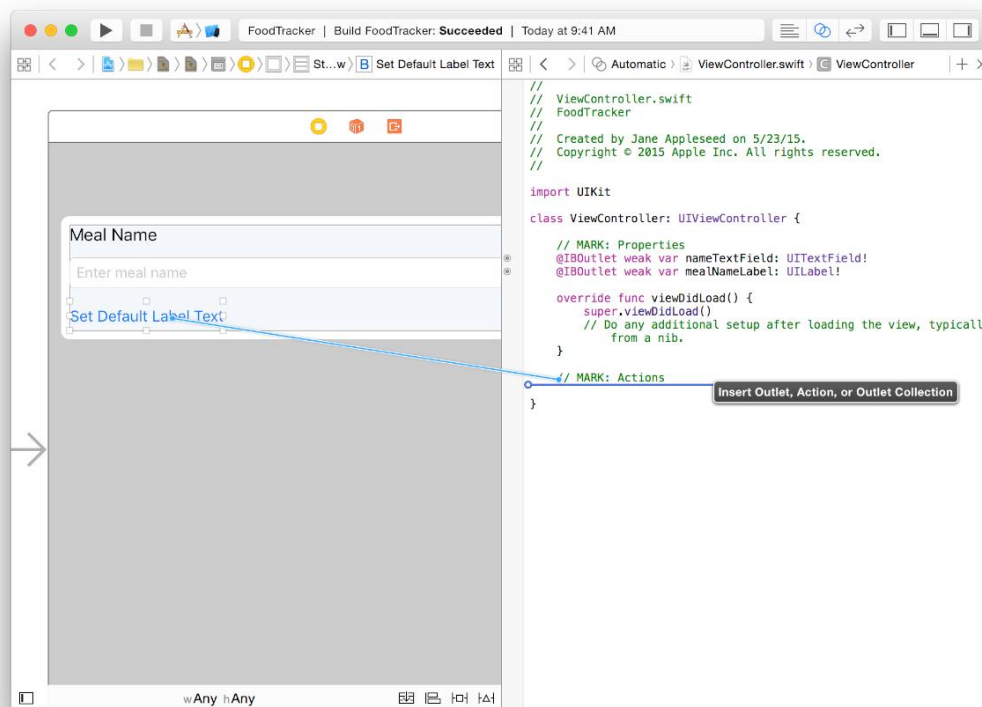
**// MARK: Actions**

comment فوق نشانگر این است که action های کد در این بخش قرار می گیرند.

۲. بر روی دکمه ی Set Default Label Text کلیک کرده، کلید control را نگه دارید. سپس

المان مزبور را کشیده و داخل فایل ViewController.swift، در زیر comment درج شده

(// MARK: Actions)، جایگذاری کنید.

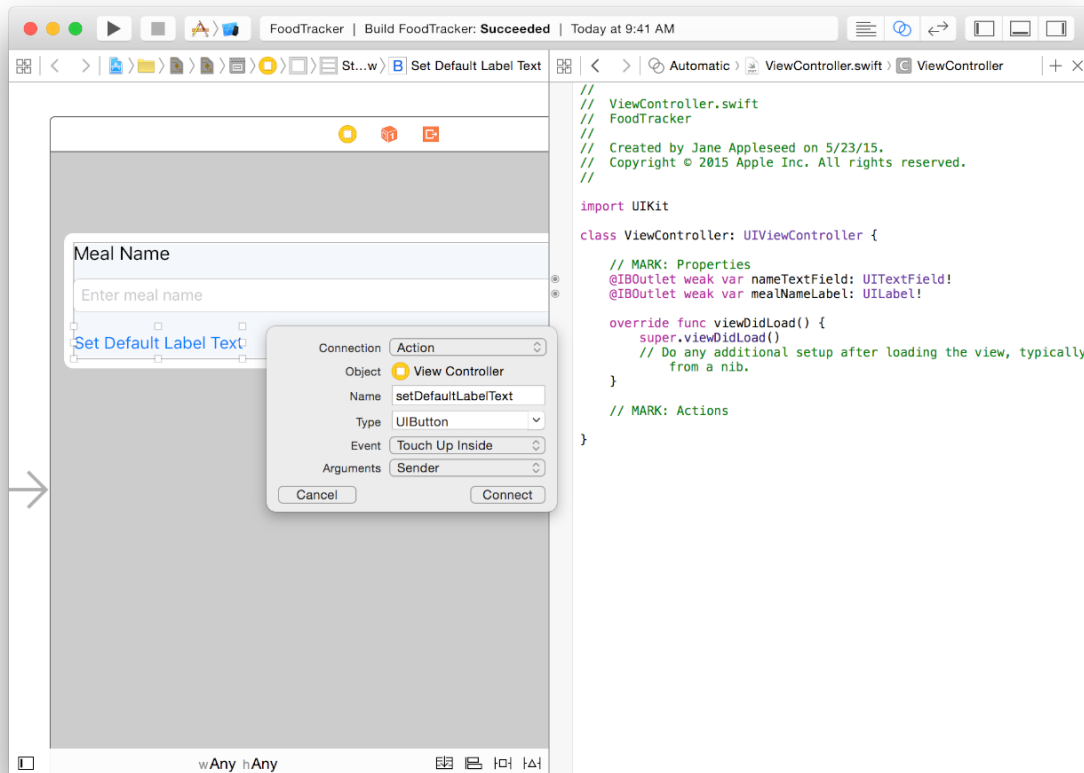


۳. در کادر محاوره ای که پدیدار می شود، در فیلد Connection گزینه ی Action را انتخاب نمایید.

۴. در فیلد Type، گزینه ی UIButton را انتخاب نمایید.

حتما متوجه شدید که مقدار فیلد Type به صورت پیش فرض بر روی AnyObject تنظیم می شود. در زبان Swift، AnyObject یک نوع است و برای توصیف آبجکتی که می تواند به هر کلاسی تعلق داشته باشد، بکار می رود. تنظیم نوع این method action بر روی مقدار UIButton، بیانگر این است که تنها آبجکت های از جنس کلاس UIButton می توانند به این متد یا action وصل شوند. کادر محاوره ای پس از وارد کردن مقادیر مورد نیاز در آن به صورت زیر خواهد بود:





۵. اکنون بر روی دکمه ی Connect کلیک نمایید.

۶. محیط Xcode خود تعریف و کد لازم برای action method را اضافه می کند.

```

@IBAction func setDefaultLabelText(sender: UIButton) {
}

```

۷. در این کد، پارامتر sender به آبجکتی که action را صدا زده و اجرای آن را سبب می شود – در این سناریو همان آبجکت button – اشاره دارد. خصیصه IBAAction (attribute) نشانگر این است که متد مورد نظریک action بوده که می توان از storyboard در interface builder به آن وصل شد. بقیه ی کد صرفاً یک متد متعارف به نام setDefaultLabelText(\_\_\_\_) را تعریف می کند.

۸. در حال حاضر، تعریف متد هیچ پیاده سازی در بدنه ی خود ندارد. در زیر کدی به بدنه ی متد اضافه می کنیم که مقدار label را به یک مقدار پیش فرض برمی گرداند.

۹. به منظور پیاده سازی action یا متدی که مقدار label را در فایل ViewController به Default Text برمی گرداند، مراحل زیر را طی نمایید:

۱. در فایل ViewController.swift، تعریف متد `setDefaultLabelText` را پیدا کنید.

۲. داخل بدنه ی متد (بین `{}`)، این خط کد را اضافه نمایید:

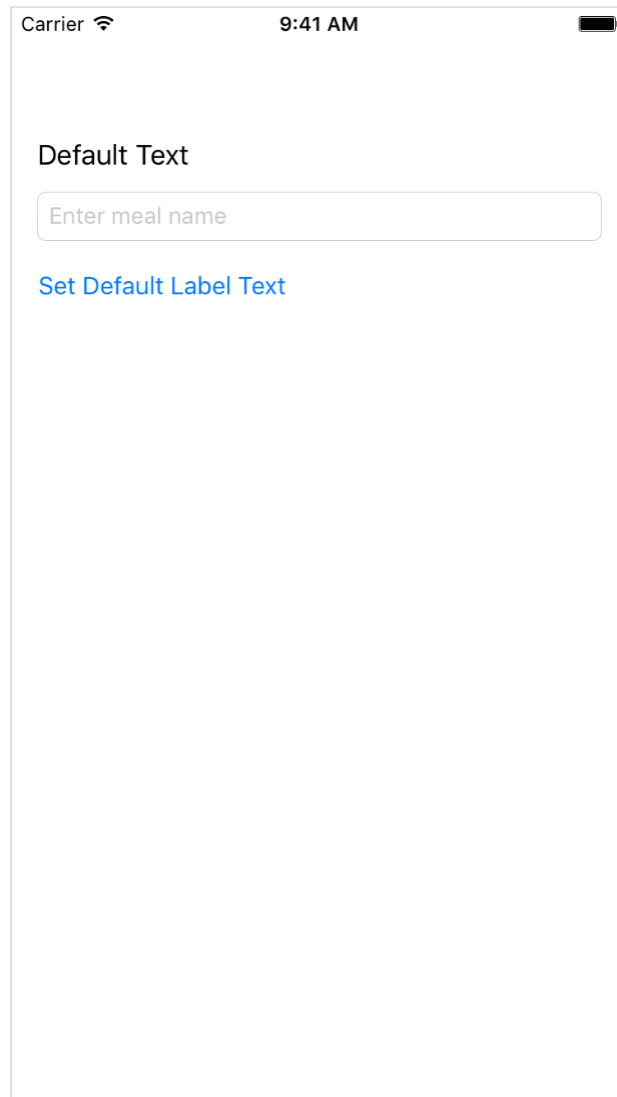
```
mealNameLabel.text = "Default Text"
```

۳. همان طور که حدس می زنید، این کد مقدار text (که یکی از property های label است) را برابر Default Text قرار می دهد. اگر به کد دقت کنید، متوجه می شوید که نوع Default Text را صریحا مشخص نکردیم و نیازی هم به این کار نیست. چراکه زبان Swift قابلیت به نام `type inference` (استنتاج نوع) دارد و با استفاده از آن می تواند درست حدس بزند که مقداری از نوع رشته را به آبجکتی از جنس کلاس NSString تخصیص می دهید.

هم اکنون کد action method شما باید به صورت زیر باشد:

```
@IBAction func setDefaultLabelText(sender: UIButton) {
    mealNameLabel.text = "Default Text"
}
```

**تست کنید:** تغییراتی که تاکنون به کد اعمال کرده اید را با شبیه ساز (Simulator) تست نمایید. زمانی که بر روی دکمه ی Set Default Label Text کلیک می کنید، مقدار label بایستی از Meal Name (مقداری که در storyboard تنظیم کردید) به Default Text (مقداری که در بدنه ی متد یا action تعریف کردید) تغییر کند.



رفتاری که با این مثال پیاده سازی کردید، مصداق و نمونه ای از پیاده سازی الگوی توسعه target-action در طراحی برنامه های iOS است. target-action یک الگوی توسعه یا design pattern است که در آن آبجکت مورد نظر، با اتفاق افتادن یا فعال شدن رخدادی معین، message یا پیغامی را به آبجکت دیگری ارسال می کند. در این سناریو، event کلیک کاربر بر روی دکمه ی Set Default Label Text است، action همان متد setDefaultLabelText است، target همان ViewController، جایی که متد در آن تعریف شده است و در پایان sender (ارسال کننده ی پیغام) دکمه ی Set Default Label Text است.

Message در واقع یک action method است که در code source تعریف شده و target – دریافت کننده ی پیغام – آن آبجکتی است که قادر به اجرای action یا عملیات تعریف شده توسط متد می باشد. آبجکتی که پیغام را ارسال می کند معمولا یک کنترل همچون button، slider یا switch است که در پاسخ به عمل کاربر (لمس صفحه ی نمایش، کشیدن یا تغییر مقدار) رخدادی را فعال می کند (به ورودی کاربر واکنش نشان داده و رخدادی را اجرا می کند).

همان طور که گفته شد این الگو به وفور در برنامه نویسی IOS پیاده سازی شده و شما نمونه های بیشتری از آن را در مباحث بعدی این سری آموزشی مشاهده خواهید کرد.

#### پردازش ورودی کاربر (برابر قرار دادن مقدار label با مقدار وارد شده توسط کاربر در text field)

در حال حاضر، برنامه طوری تنظیم است که با کلیک بر روی دکمه، مقدار label را بر روی Default Text قرار می دهد. اکنون می خواهیم رفتاری برای برنامه تعریف کنیم که با کلیک بر روی دکمه ی آن، مقدار label را برابر مقدار وارد شده توسط کاربر در text field قرار دهد.

در این بخش جهت ساده نگه داشتن برنامه، برای اعلان زمان بروز رسانی مقدار label، متکی به action کاربر که همان کلیک بر روی دکمه ی Return صفحه کلید است، خواهیم بود.

برای پردازش و کار با ورودی کاربر از آبجکت text field، به یک delegate از آبجکت مزبور نیاز دارید. Delegate یک آبجکت است که به نمایندگی از یا با همکاری آبجکت دیگری کاری را انجام می دهد. آبجکتی که کاری را به آبجکت دیگر محول می کند (به آن اشاره می کند) یا با هماهنگی آن آبجکت کاری را انجام می دهد در اینجا delegating object نامیده می شود. آن آبجکتی که کاری از جانب delegating object به آن سپرده می شود، delegate خوانده می شود.

Delegating object (در این مثال text field)، یک اشاره گر (reference) از آبجکت دیگر (delegate) نزد خود نگه داشته و در زمان مناسب، delegating object نام برده یک پیغام به delegate ارسال می کند. حال این پیغام به delegate درباره ی رخدادی خبر می دهد که delegating object در صدد مدیریت آن بوده و یا به تازگی مدیریت کرده است. در پی آن، delegate با بروز رسانی ظاهر/وضعیت خود، آبجکت دیگر در برنامه یا بازگردانی مقداری که تعیین می کند یک رخداد در حال وقوع چگونه اداره شود، به این دست رخدادها واکنش نشان می دهد.

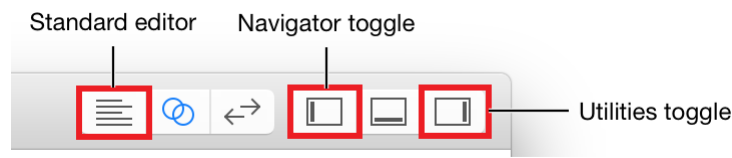
این delegate text field، زمانی که مقدار کنترل مورد نظر (text field) در حال ویرایش است، با آن تعامل داشته و کاملاً اطلاع دارد چه زمانی رخدادهای مهم اتفاق می افتد – مانند اینکه کاربر چه زمانی ویرایش متن را آغاز و کی آن را متوقف می کند. delegate می تواند با استفاده از این اطلاعات، داده ها را در زمان مناسب ذخیره/حذف کند، صفحه کلید را ببندد و غیره ...

هر آبجکتی می تواند به عنوان delegate آبجکتی دیگری ایفای نقش کند مادام اینکه از الگو یا protocol مناسب پیروی نماید. Protocol یا الگویی که delegate کنترل field text را تعریف می کند، UITextFieldDelegate نام دارد (protocol = صرفاً یک الگو و معرفی یک سری متد و property است. اگر برای آبجکت یا کلاسی protocol اعلان کنید، آن کلاس یا آبجکت ملزم به پیاده سازی متدها و property های تعیین شده توسط آن protocol می شود). در این مثال، از آنجایی که ViewController اشاره گر به آبجکت text field را نگهداری می کند، view controller را delegate آن text field انتخاب می کنیم.

ابتدا بایستی ViewController را بر اساس protocol یا الگوی UITextFieldDelegate تنظیم و پیاده سازی کنید. برای استفاده از protocol خاص، کافی است آن را در خط تعریف کلاس لحاظ نمایید.

به منظور پیروی از الگوی پیاده سازی UITextFieldDelegate در view controller و انتساب کلاس نام برده به عنوان دلیگیت text field، مراحل زیر را گام به گام دنبال نمایید:

۱. در صورتی که assistant editor باز است، با کلیک بر روی آخرین دکمه، از سمت چپ، standard editor را باز کنید.



Project navigator و utility area را با کلیک به ترتیب بر روی دکمه های Navigator و Utilities در نوار ابزار محیط Xcode (اشاره شده در تصویر فوق) باز نمایید.

در project navigator، بر روی فایل ViewController.swift کلیک کرده و آن را باز نمایید.

داخل فایل مزبور، خط حاوی کلیدواژه ی class را پیدا کنید:

```
class ViewController: UIViewController {
```

پس از UIViewController، یک ویرگول اضافه نموده و سپس UITextFieldDelegate را درج نمایید. با این کار شما protocol نام برده را adopt می کنید یا به عبارتی الگویی برای معرفی آن کلاس به عنوان delegate ارائه می دهید (protocol = صرفاً یک الگو و معرفی یک سری متد و property است. اگر برای آبجکت یا کلاسی protocol اعلان کنید، آن کلاس یا آبجکت ملزم به پیاده سازی متدها و property های تعیین شده توسط protocol می شود).

```
class ViewController: UIViewController, UITextFieldDelegate {
```

با اضافه نمودن این protocol به خط تعریف کلاس ViewController، به کلاس ذکر شده این قابلیت را دادید که خود را به عنوان UITextFieldDelegate معرفی کند. به عبارت دیگر شما می توانید از این پس، از کلاس نام برده به مثابه ی delegate آجکت text field استفاده کرده، برخی از رفتارهای (متدهای) آن را پیاده سازی و بدین وسیله ورودی کاربر را مدیریت نمایید.

به منظور انتساب کلاس ViewController به عنوان delegate آجکت nameTextField، مراحل زیر را دنبال نمایید:  
داخل فایل ViewController.swift، متد viewDidLoad() را پیدا کنید:

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
}
```

در پیاده سازی این متد یک comment مشاهده می کنید. نیازی به این comment نیست، می توانید آن را حذف کنید.

۱. در زیر دستور super.viewDidLoad()، ابتدا یک خط خالی ایجاد کرده، سپس کد زیر را وارد نمایید:

```
// Handle the text field's user input through delegate callbacks.
nameTextField.delegate = self
```

کلیدواژه ی self به کلاس ViewController اشاره دارد، زیرا آجکت text field در محدوده یا scope تعریف کلاس ViewController مورد اشاره قرار گرفته است. شما می توانید برای فهم کاربرد و استفاده ی هر بخش کد، توضیحات و comment در زیر آن درج نمایید.

پس از اعمال تغییرات فوق، کد شما باید مشابه زیر باشد:

```
override func viewDidLoad() {
```

```
super.viewDidLoad()
```

```
// Handle the text field's user input through delegate callbacks.
```

در این بخش از کد ورودی کاربر در تکست فیلد را از طریق توابع بازفراخوان یا کالک دلیگیت

```
// مدیریت می کنیم
```

```
nameTextField.delegate = self
```

```
}
```

اکنون کلاس ViewController یک delegate برای آجکت nameTextField است. UITextFieldDelegate تعدادی متد اختیاری نیز ارائه می دهد که شما ملزوم به پیاده سازی آن ها نیستید. اما برای اینکه برنامه جاری رفتار و عملیات دلخواه را داشته باشد، بایستی دو متد از مجموع متدهای آن را پیاده سازی کنید:

```
func textFieldShouldReturn(textField: UITextField) -> Bool
```

```
func textFieldDidEndEditing(textField: UITextField)
```

برای درک اینکه چه زمان این متدها فراخوانده شده و چه کارهایی را انجام می دهند، آگاهی از نحوه ی عملکرد و واکنش text field ها به ورودی کاربر لازم و ضروری است. زمانی که کاربر text field را با قابلیت لمس انتخاب می کند، کنترل نام برده (text field) به صورت خودکار first responder می شود. در یک برنامه، first responder اولین آجکتی است که انواع event های برنامه از جمله key event ها (رخداد های مربوط به دکمه)، motion event (رخداد های مربوط به حرکت یا تکان خوردن دستگاه)، action message ها و غیره ... را دریافت می کند. به عبارت دیگر، بسیاری از رخداد های فعال شده بر اثر اعمال کاربر ابتدا به first responder ارجاع داده می شوند.

از آنجایی که text field به صورت پیش فرض اولین المان واکنش دهنده یا first responder می باشد، سیستم عامل IOS صفحه کلید را به نمایش گذاشته و session ویژه ی درج و ویرایش مقدار آن المان را مهیا و راه اندازی می کند. آنچه کاربر با فشردن دکمه های صفحه کلید تایپ می کند، بلا درنگ وارد المان text field می شود.

زمانی که کاربر قصد دارد ویرایش مقدار text field را متوقف کند، text field بایستی از موقعیت خود به عنوان first responder دست بکشد (از این وضعیت خارج شود). به



دنبال این اتفاق، چون که المان text field دیگر آبجکت فعال در برنامه نیست، رخدادهای بعدی می بایست به آبجکت دیگری ارجاع یا سوق داده شوند.

در اینجا است که پیاده سازی متدهای UITextFieldDelegate بکار گرفته و اجرا می شود. شما می بایست اعلان کنید که با اتمام editing session و کلیک کاربر بر روی دکمه ی مربوطه (Return یا Done)، المان text field بایستی از موقعیت خود به عنوان first responder دست بکشد (جایگاه خود به عنوان آبجکت فعال در پنجره و دریافت کننده ی تمامی رخدادهای رخدادها رها کند). این کار را در بدنه ی متد textFieldShouldReturn(:) تعریف و پیاده سازی می کنید. متد مذکور زمانی که کاربر بر روی دکمه ی Return (یا در مثال جاری دکمه ی Done) کلیک می کند، فراخوانده و متعاقباً اجرا می گردد.

جهت پیاده سازی متد textFieldShouldReturn(:) از protocol یا الگوی UITextFieldDelegate، مراحل زیر را دنبال نمایید:

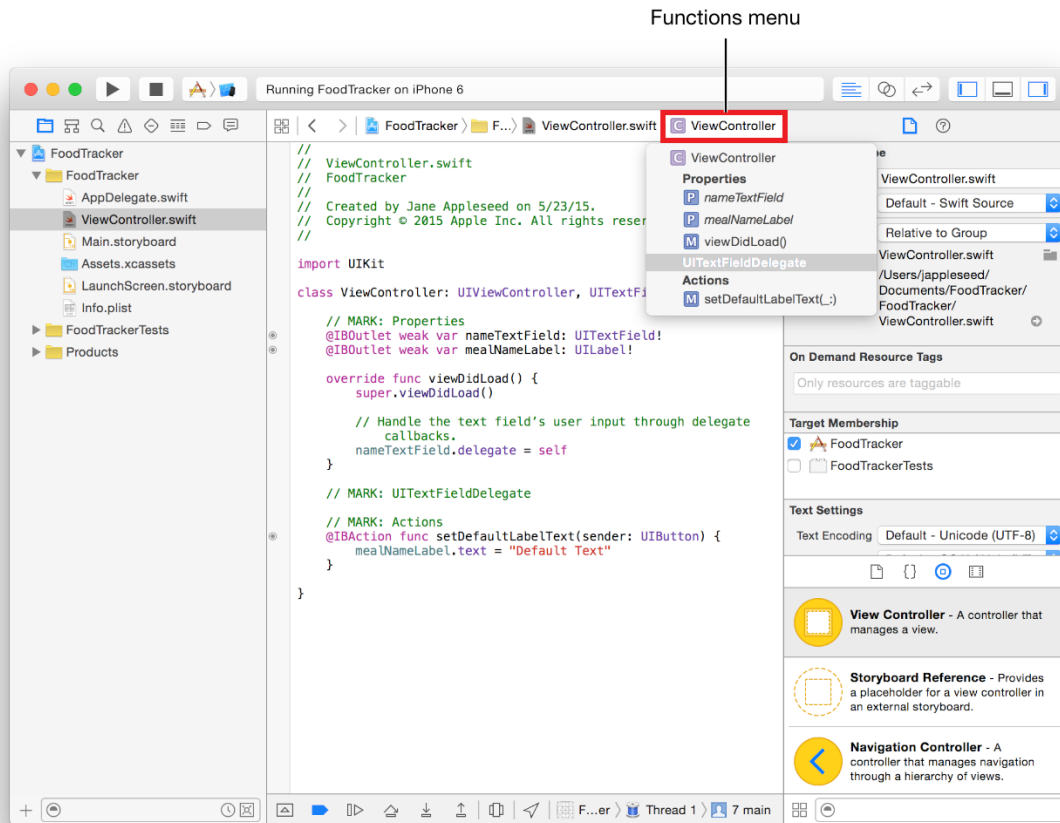
داخل فایل ViewController.swift، در بالای بخش MARK: Actions، کد زیر را وارد نمایید:

```
// MARK: UITextFieldDelegate
```

این comment یا کد درج توضیحات به سازمان دهی کد، خوانایی بیشتر و فهم کاربرد بخش های مختلف آن کمک شایانی می کند.

تاکنون تعداد زیادی از این comment ها را به کد خود اضافه کردید. Xcode هر یک از این comment ها را به صورت یک section title (متن یا عنوان توصیف کننده ی بخشی از سند یا صفحه) در functions menu فایل کد اصلی برنامه (source code) لیست می کند. برای دسترسی به functions menu، کافی است بر روی اسم فایل مربوطه در بالای ناحیه ی ویرایش (editor area) کلیک نمایید. functions menu به شما امکان می دهد تا با کلیک بر روی آیتم های قابل گزینش آن، به بخش های مرتبط در کد خود پیمایش کنید (بپروید). با کمی

دقت می بینید که کلیه ی بخش هایی که قبلاً با MARK // علامت گذاری کردید، در این بخش قابل مشاهده می باشد. می توانید بر روی یکی از section title ها کلیک کرده تا به سرعت به بخش مربوطه در فایل هدایت شوید.



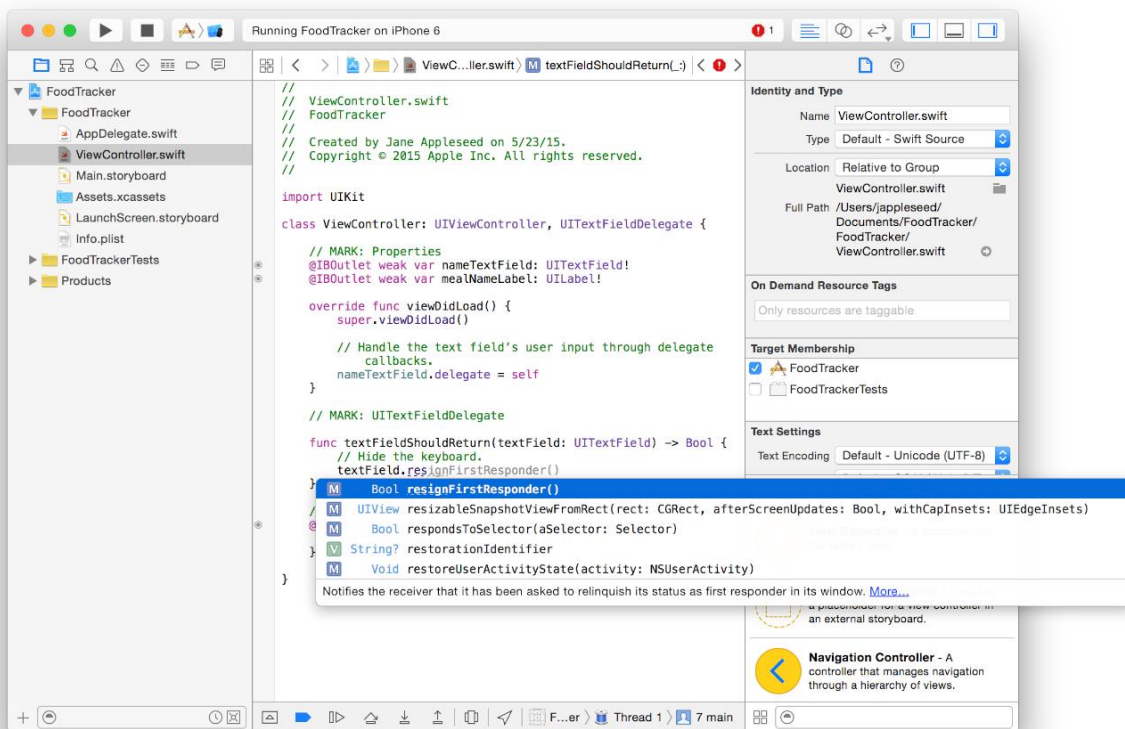
۱. در زیر comment، متد زیر را درج نمایید:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
}
```

۲. در بدنه ی این متد، کد زیر را وارد نموده تا text field از موقعیت خود به عنوان first responder دست بکشد. همچنین یک comment جهت ارائه ی توضیحات در خصوص کاربرد کد، در بالای آن درج نمایید.

```
// Hide the keyboard.
textField.resignFirstResponder()
```

۳. سعی کنید خط دوم کد را (متد `textField.resignFirstResponder()`) را (جای کپی و پیست) به صورت دستی تایپ کنید. در حین کدنویسی متوجه خواهید شد که Xcode سعی می کند بر اساس شرایط و بستر جاری، کدی که قصد وارد کردن آن را دارید، حدس زده و بخش های بعدی آن را به شما پیشنهاد کند. این ویژگی که `code completion` نامیده می شود، تنها یکی از امکانات Xcode است که در افزایش سرعت برنامه نویسی می تواند کمک شایانی به شما بکند. زمانی که Xcode لیست پیشنهادات یا کدهای تکمیل کننده ی خود را به صورت لیست ارائه می دهد، کافی است با نوار پیمایش آن به پایین لیست رفته و پس از یافتن مورد مناسب، کلید `Return` را فشار دهید. خواهید دید که Xcode کل خط را به صورت خودکار برای شما وارد می کند.



۴. در بدنه ی متد جاری، دستور زیر را نیز وارد نمایید:

`return true`

از آنجایی که این متد یک مقدار بولی (Boolean) در خروجی برمی گرداند، بازگردانی مقدار true توسط متد مورد نظر نشانگر این است که text field بایستی در پاسخ به اعمال کلید Return (توسط کاربر) صفحه کلید را ببندد (از موقعیت خود به عنوان first-responder یا آبجکت فعال در پنجره دست بکشد).

متد (:) textFieldShouldReturn هم اکنون می بایست مشابه زیر باشد:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    // Hide the keyboard.
    textField.resignFirstResponder()
    return true
}
```

دومین متدی که باید پیاده سازی کنید، (:) textFieldDidEndEditing، زمانی فراخوانده می شود که text field از موقعیت خود به عنوان اولین آبجکت واکنش دهنده (first-responder) دست بکشد (از این وضعیت خارج شود). این متد پس از اجرای textFieldShouldReturn (متدی که در بالا پیاده سازی کردید)، صدا خورده می شود. تابع textFieldShouldReturn به شما این امکان را می دهد تا اطلاعات وارد شده در text field را خوانده و آن را برای منظور خاصی بکار ببرید. برای مثال، در شرایط فعلی شما مقدار وارد شده در text field را گرفته و از آن برای تغییر مقدار label در رابط کاربری برنامه ی خود استفاده می کنید.

جهت پیاده سازی متد (:) textFieldDidEndEditing از protocol یا الگوی UITextFieldDelegate، کافی است مراحل زیر را گام به گام دنبال نمایید:

۱. داخل فایل ViewController.swift، سپس از متد (:) textFieldShouldReturn، متد زیر را درج نمایید:

```
func textFieldDidEndEditing(textField: UITextField) {
}
```

۲. در بدنه ی این متد، دستور زیر را وارد نمایید:

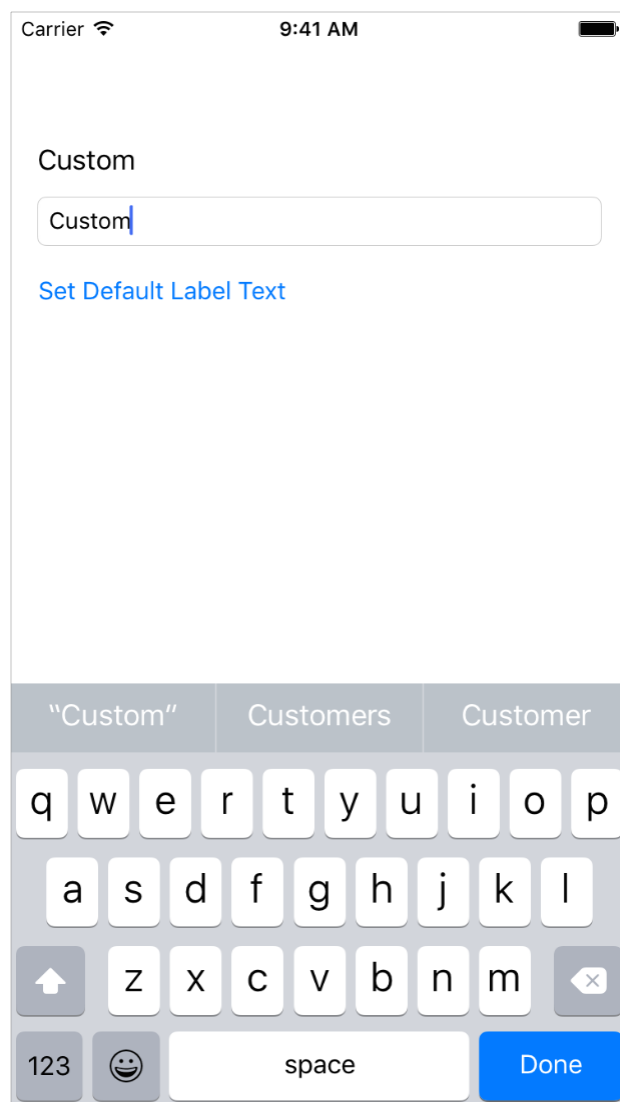
```
mealNameLabel.text = textField.text
```

با افزودن این متد خواهید توانست مقدار وارد شده در text field را جایگزین مقدار label (در بالای کادر متن) کرده و این تغییر را در UI معکس و برای کاربر نمایش دهید. در حال حاضر کد متد (:\_): textFieldDidEndEditing شما باید به صورت زیر باشد:

```
func textFieldDidEndEditing(textField: UITextField) {
    mealNameLabel.text = textField.text
}
```

**تست کنید:** برنامه ی خود را به همراه تمامی تغییراتی که اعمال کرده اید، در محیط شبیه ساز/ Simulator اجرا و تست کنید. در برنامه باید بتوانید text field را انتخاب کرده و مقداری از نوع متن را در آن وارد کنید. زمانی که بر روی دکمه ی Done کلیک می کنید، صفحه کلید باید بسته شده و متن label می بایست مقدار وارد شده در text field را عیناً نمایش دهد. حال اگر بر روی دکمه ی Set Default Label Text کلیک نمایید، مقدار label بایستی تغییر کرده و به مقدار اولیه Default Text (مقداری که قبلاً توسط action تعیین کرده بودید) بازگردد.

آموزشگاه تحلیکیر داده



## درس ۵ : چرخه حیات view controller

آموزش کار با view controller ها

در این مبحث همچنان بر روی scene و UI مربوطه ی آن در برنامه ی ثبت و مشاهده ی اطلاعات غذا، FoodTracker، کار خواهید کرد. ترتیب المان های جاری UI را تغییر داده و با استفاده از image picker یک عکس به رابط کاربری برنامه ی خود اضافه خواهید نمود.

در پایان برنامه ظاهری مشابه زیر خواهد داشت:

Carrier 9:41 AM

Meal Name

Enter meal name

[Set Default Label Text](#)

No photo selected

آنچه خواهید آموخت

۱. با چرخه‌ی حیات `view controller` و مفهوم آن آشنا شده، بدانید توابع `viewWillAppear`، `viewDidLoad`، `viewDidAppear` و `callback` —ای آن، `viewWillAppear` و `viewDidLoad` چه زمانی صدا خورده و اجرا می شوند.

۲. بتوانید بین view controller ها داده رد و بدل نمایید.
۳. یک view controller را از صفحه حذف یا پنهان کنید.
۴. با الحاق کردن آجکت های gesture recognizer به view، قابلیت تعامل با کاربر و واکنش نشان دادن به رخداد/event ها را به آن اعطا نمایید.
۵. بر اساس سلسله مراتب و زنجیره ی ارث بری (class hierarchy) UIView/UIControl، رفتار یک آجکت را پیش بینی کنید.
۶. با استفاده از ابزار asset catalog، محتوا و منابعی همچون عکس را به پروژه اضافه نمایید.

### آشنایی با life cycle/چرخه ی حیات View Controller و مفهوم آن

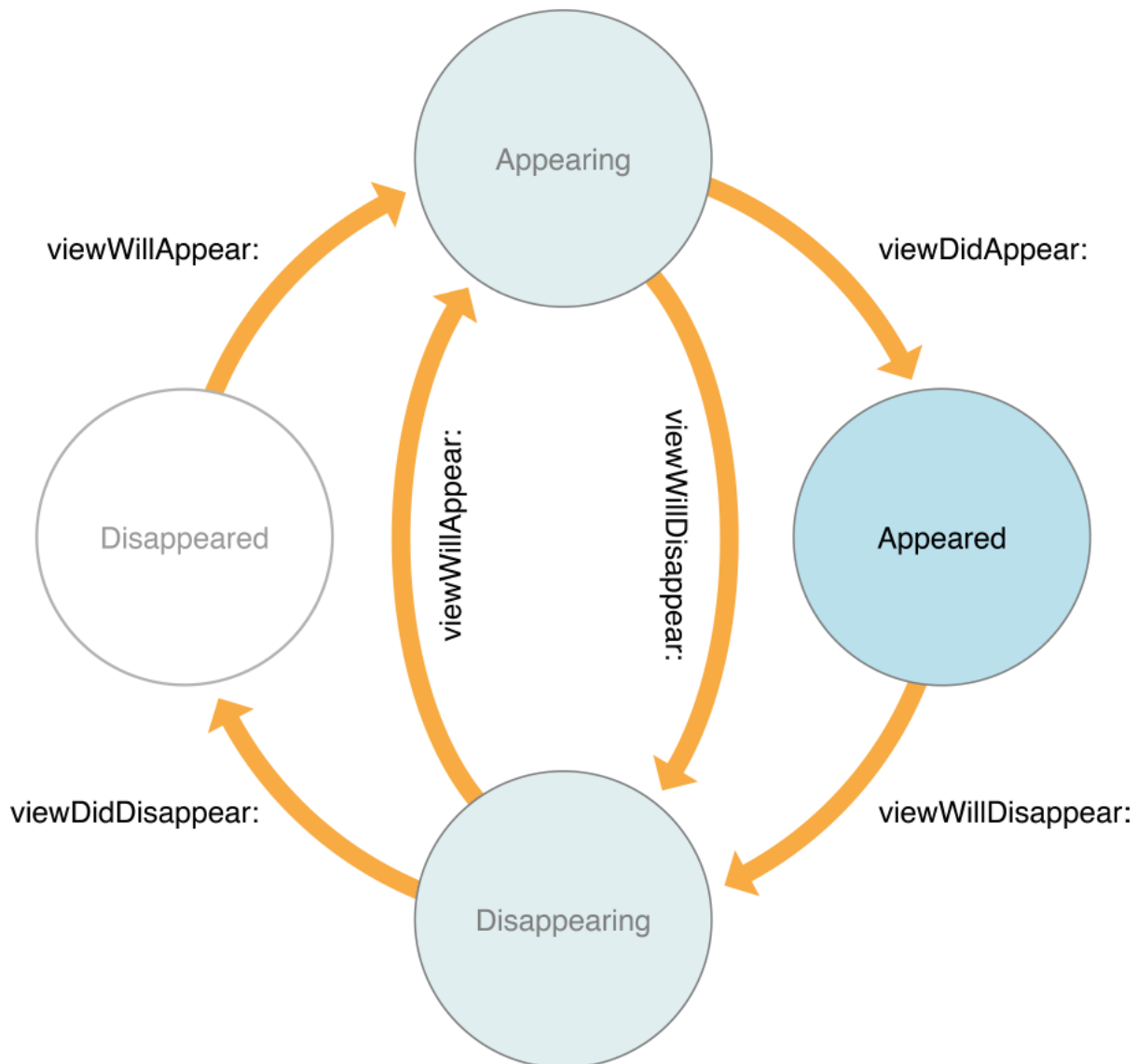
در حال حاضر، برنامه ی FoodTracker از تنها یک scene یا صفحه ی محتوا برخوردار است که UI آن توسط فقط یک view controller مدیریت می شود. با ساخت اپلیکیشن های پیچیده تر، ناگزیر با scene های بیشتری سر و کار داشته و علاوه بر آن می بایست بارگذاری و حذف view ها از حافظه را همین که بر روی صفحه نمایش داده و سپس از آن حذف می شوند، اداره نمایید.

آجکتی که از روی کلاس UINavigationController (و کلاس های فرزند آن) ساخته می شود، تعدادی متد پیش فرض دارد که توسط آن ها view hierarchy (نمای سلسله مراتبی از آجکت ها/view های یک scene همچون دکمه، لیبل) خود را مدیریت می کند. زمانی که یک view controller تغییر وضعیت می دهد (برای مثال در صفحه نمایش داده می شود و سپس از آن حذف می گردد)، iOS خود این متدها را در زمان مناسب فراخوانی می کند.

کلاس فرزندی که از UINavigationController ایجاد می کنید، مانند کلاس ViewController که در برنامه ی جاری با آن کار می کنید، تمامی متدهای کلاس والد (UINavigationController) را به ارث برده و در این حین به شما امکان می دهد تا کد خود را جهت پیاده سازی عملیات و رفتار دلخواه به بدنه ی این متدها اضافه نمایید. به منظور



تنظیم و آماده سازی view ها جهت نمایش در صفحه یا حذف آن ها در زمان مناسب (یا گام مربوطه در فرایند مزبور)، آگاهی از ترتیب و زمان دقیق فراخوانی متدهای نام برده ضروری است. البته این عملیات کمی پیچیده بوده و در مباحث آینده به پیاده سازی آن خواهید پرداخت.



متدهای کلاس UINavigationController به ترتیب شرح داده شده در زیر صدا خورده و اجرا می شوند:

- `viewDidLoad()` – زمانی فراخوانی می شود که `view` content ای که در بالای سلسله مراتب قرار گرفته و چندین `view` دیگر را دربرمی گیرد) کلاس `view controller` ایجاد شده و از storyboard بارگذاری می شود. این متد ویژه

ی تنظیم اولیه تعبیه شده است. لازم به ذکر است که view ها، گاهی به دلیل عدم وجود منابع کافی، از برنامه حذف می شوند و از اینرو هیچ تضمینی وجود ندارد که متد مذکور تنها یکبار فراخوانده شود (ممکن است به دلیل منابع محدود و حذف view ها از صفحه، لازم باشد چندین بار فراخوانی شود).

- `viewWillAppear()` – ویژه ی عملیاتی طراحی شده که می خواهید حتما پیش از به نمایش گذاشتن view برای کاربر، انجام شوند. از آنجایی که امکان دارد قابلیت رویت یک view به دلیل وجود دیگر view ها، بین دو حالت تغییر کند (نمایش داده شده/پنهان شود) یا کلاً تحت الشعاع قرار گیرد، این متد همیشه بلافاصله قبل از نمایان شدن content view بر روی صفحه صدا زده می شود.

- `viewDidAppear()` – عملیاتی که می خواهید بلافاصله پس از نمایان شدن view بر روی صفحه، انجام شوند از جمله واکنشی داده ها و به اجرا گذاشتن یک انیمیشن و غیره ... در بدنه ی این متد تعریف می شوند. از آنجایی که امکان دارد قابلیت رویت یک view به دلیل وجود دیگر view ها، بین دو حالت تغییر کند (نمایش داده شده/پنهان شود) یا کلاً تحت الشعاع قرار گیرد، این متد همیشه بلافاصله پس از نمایان شدن content view بر روی صفحه صدا زده می شود.

همان طور که در نمودار فوق مشاهده می کنید، تعدادی متد مکمل نیز جهت `teardown` یا حذف view از صفحه وجود دارد.

برخی از متدهای ذکر شده را جهت بارگذاری و نمایش داده های view در زمان مناسب، در برنامه ی `FoodTracker` مورد استفاده قرار خواهید داد. در واقع، اگر بخاطر داشته باشید، داخل بدنه ی یکی از این متدها (تابع `viewDidLoad()` از کلاس `ViewController`) کدنویسی کردید:

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Handle the text field's user input through delegate callbacks.
    nameTextField.delegate = self
}
```

این سبک طراحی برنامه که در آن view controller ها به عنوان پل های ارتباطی (pipeline) بین view ها و data model ایفای نقش کرده و زمینه ی تعامل بین آن ها را فراهم می نمایند، در اصطلاح معماری MVC ( الگو توسعه ی مدل-نما-کنترلگر) خوانده می شود. در این الگوی توسعه، model وظیفه ی کار با داده ها (ارتباط با پایگاه داده/هر منبع داده ای همچون آرایه و واکنشی اطلاعات از آنها، ذخیره ی موقت یا تبدیل آن ها به یک شی و همچنین بررسی صحت داده ها) را بر عهده دارد، بخش view ظاهر برنامه/UI را در صفحه به نمایش گذاشته، محتوای قابل مشاهده برای کاربر را ارائه می دهد و در نهایت لایه ی controller اداره ی view ها، مدیریت تعامل با کاربر را عهده دار بوده و در حقیقت واسطی میان model و view محسوب می شود. با واکنش نشان دادن به اعمال یا ورودی کاربر، پر کردن view ها با داده از data model، بخش controller به مثابه ی یک درگاه یا پل ارتباطی بین لایه ی model و view ایفای نقش می کند. در حال حاضر تمامی برنامه های کارآمد و پربازده ی IOS مبتنی بر این الگوی توسعه طراحی می شوند و برنامه ی FoodTracker نیز از این امر مستثنی نیست.

در بخش بعدی UI ساده برنامه را گسترش داده و قالب یا طرح نهایی (layout) آن را مشخص می کنید.



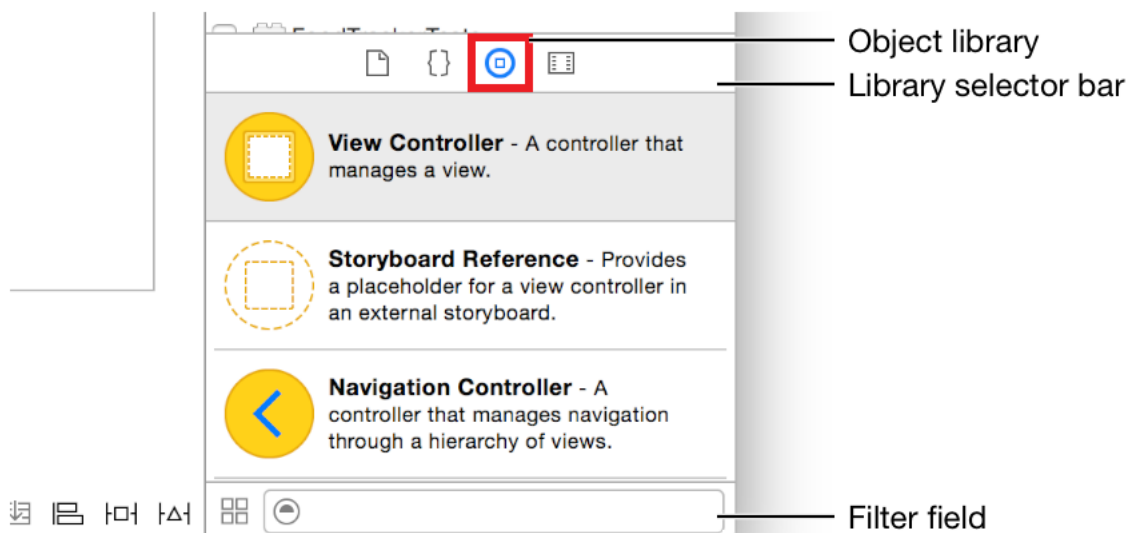
مرحله بعدی که آخرین گام در تکمیل ظاهر برنامه ی Foodtracker است، تعریف مکانیزمی جهت نمایش یک عکس، ویژه ی غذای مورد نظر در UI می باشد. برای این منظور از یک image view (کلاس UIImageView) بهره می گیرد. Image view یک المان UI است که تصویری را برای کاربر بر روی صفحه به نمایش می گذارد.

به منظور افزودن image view به scene فعلی مراحل زیر را گام به گام دنبال نمایید:

۱. فایل storyboard خود، Main.storyboard را از project navigator انتخاب کرده و باز

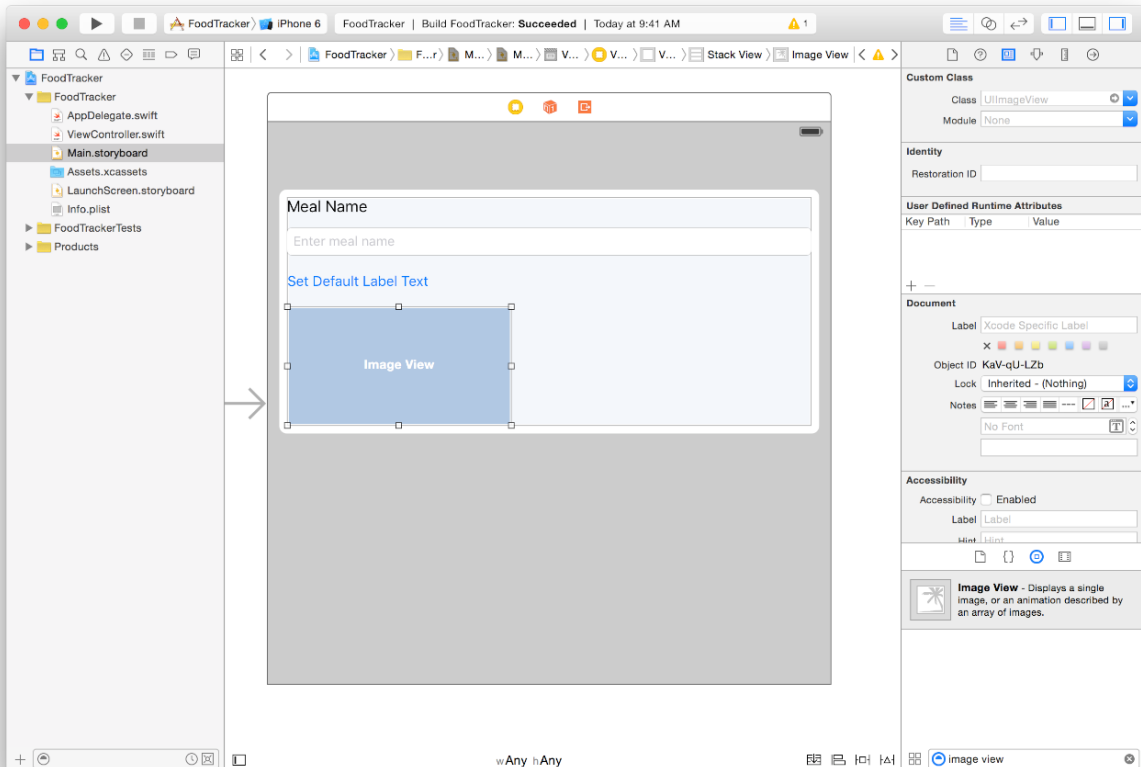
نمایید.

۲. Object library را از utility area باز نمایید (یا این مراحل را دنبال نمایید):  
(View > Utilities > Show Object Library).



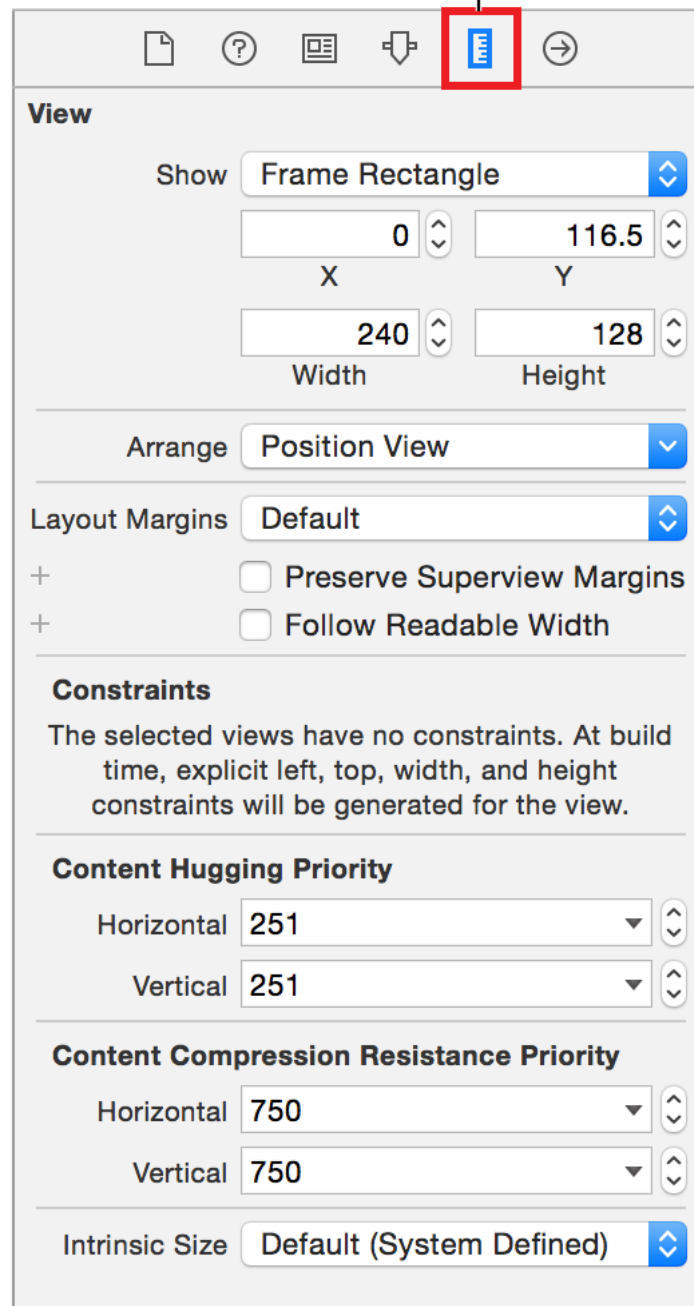
۳. داخل فیلد جستجوی Object library، عبارت image view را وارد نمایید تا Xcode به سرعت آبجکت Image View را یافته و در اختیار شما قرار دهد.

۴. آبجکت نام برده را از Object library کشیده و در سطح scene جاری، زیر کنترل دکمه جایگذاری کنید.



۵. پس از کلیک بر روی المان image view و انتخاب آن، کادر Size inspector را در utility area باز کنید. اگر بخاطر داشته باشید، Size inspector با کلیک بر روی دکمه ی پنجم از سمت چپ در inspector selector bar، به نمایش در می آید. این کادر به شما اجازه ی تنظیم اندازه و مکان قرار گیری آبجکت مورد نظر در storyboard را می دهد.

Size inspector

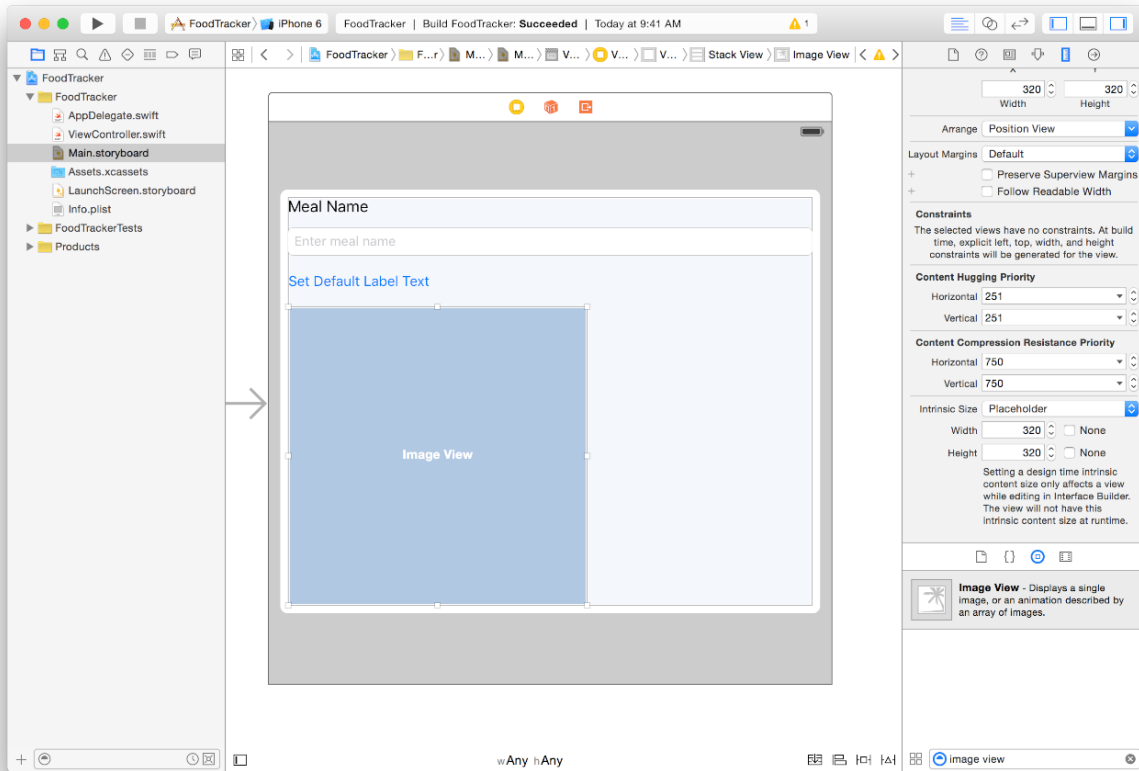


۶. از فیلد Intrinsic Size در پایین کادر، گزینه ی Placeholder را انتخاب نمایید.

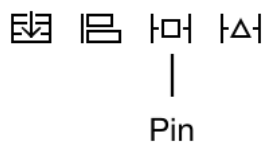
۷. داخل فیلدهای Width و Height، مقدار 320 را وارد کنید. کلید Return را فشار دهید.

به دلیل اینکه در view image حاضر عکسی برای نمایش وجود ندارد، از intrinsic content size آن (حداقل فضای لازم برای نمایش کامل و بی نقص

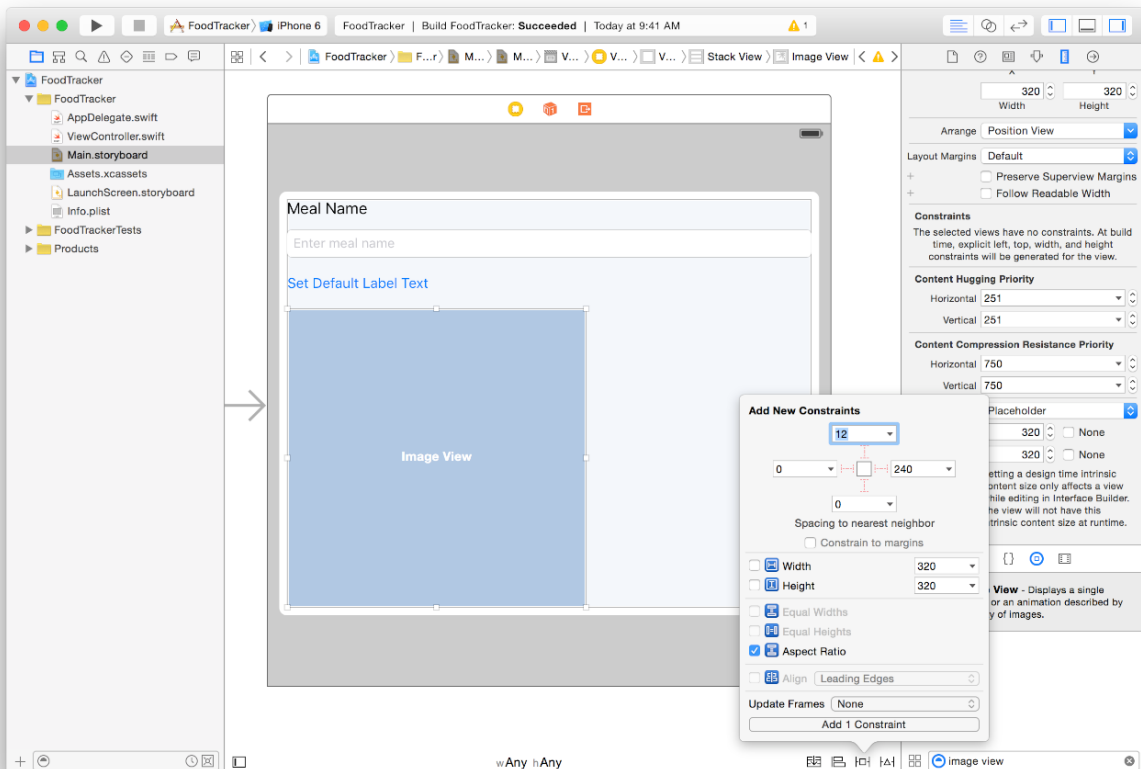
محتوای view) اطلاع کافی نداریم. بنابراین فیلد Intrinsic Size را بر روی گزینه ی placeholder می تنظیم می کنیم تا بعده ها بتوانیم اندازه ی صحیح و constraint های مناسب را در interface builder مشخص کنیم.



۸. در زیر canvas (پس زمینه ی storyboard)، گوشه ی سمت راست محیط، منوی Pin را با کلیک بر روی سومین دکمه از سمت چپ باز نمایید.



۹. در کادر جاری، چک باکس Aspect Ratio را تیک دار نمایید. هم اکنون منوی Pin باید مشابه زیر باشد:



۱۰. در منوی Pin، بر روی دکمه ی Add 1 Constraint کلیک نمایید.



Image view در حال حاضر دارای aspect ratio/نسبت پهنا به ارتفاع 1:1 می باشد. بنابراین یک مربع را برای کاربر به نمایش می گذارد.

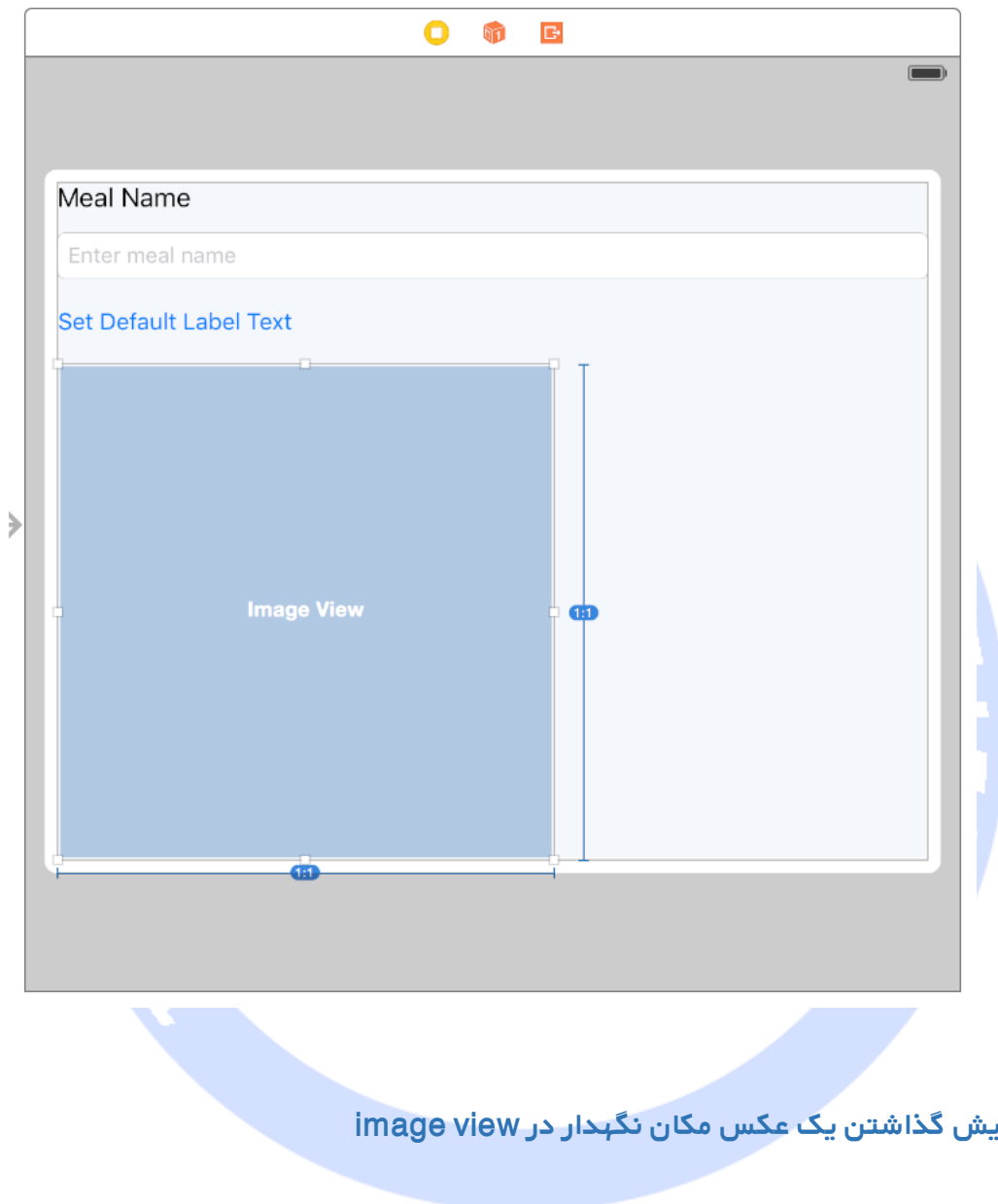
۱۱. پس از کلیک بر روی المان image view و انتخاب آن، کادر Attribute inspector را باز کنید.

۱۲. اگر بخاطر داشته باشید، کادر Attribute inspector با کلیک بر روی دکمه ی چهارم از سمت چپ در inspector selector bar باز می شود. در این کادر شما می توانید property های یک آبجکت را در storyboard خود ویرایش کنید.

۱۳. در کادر Attribute inspector، فیلدی که Interaction نام دارد را پیدا کرده و سپس چک باکس User Interaction Enabled را تیک دار نمایید. این امکان به کاربران اجازه می دهد تا با المان image view تعامل داشته باشند.

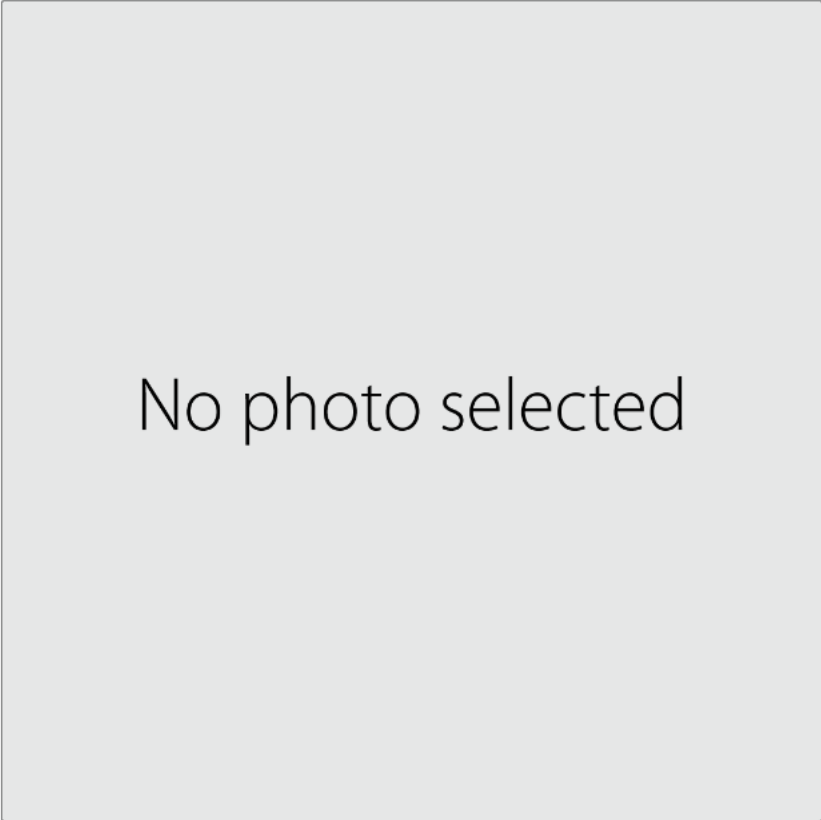


اپلیکیشن شما اکنون باید ظاهری مشابه زیر داشته باشد:



به نمایش گذاشتن یک عکس مکان نگهدار در `image view`

شما باید به طریقی به کاربر اعلان کنید که اجازه دارد با المان `image view` تعامل داشته و از آن یک عکس را انتخاب و مشاهده نماید. برای نیل به این هدف، یک عکس به عنوان مکان نگهدار در این المان به نمایش می گذارید که کاربر با مشاهده ی آن متوجه می شود که می تواند با آن تعامل داشته و تصویری را برای مشاهده انتخاب کند.



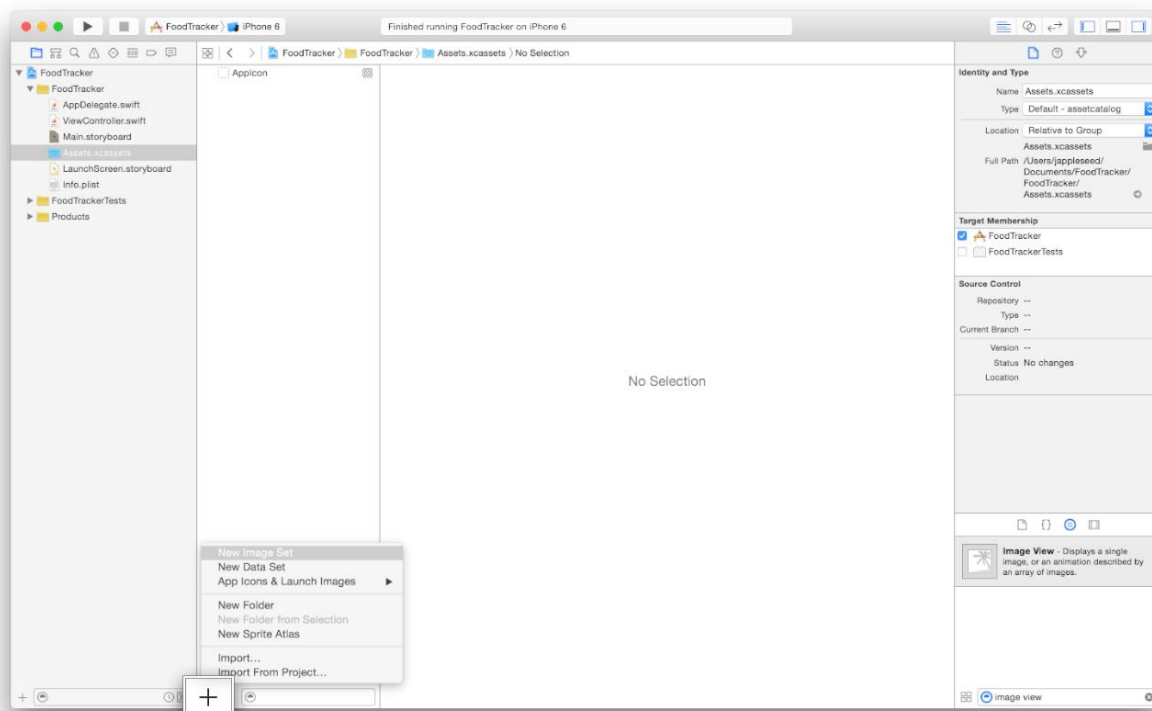
No photo selected

می‌توانید این تصویر را در پوشه ی Images/ از پروژه ی حاضر (پس از دانلود آن) انتخاب کنید یا عکس دلخواه خود را انتخاب نمایید.

به منظور افزودن یک عکس به پروژه ی خود مراحل زیر را دنبال نمایید:

۱. در project navigator، با کلیک بر روی فایل Assets.xcassets، ابزار asset catalog را باز کنید. asset catalog بستری است که در آن asset های پروژه (محتوای برنامه ی خود همچون عکس ها، فایل های تصویری و متنی) را ذخیره و سازمان دهی می کنید.

۲. در پایین محیط کاری Xcode، گوشه ی سمت چپ بر روی دکمه ی (+) کلیک کرده و سپس گزینه ی New Image Set را از منوی pop-up انتخاب نمایید.

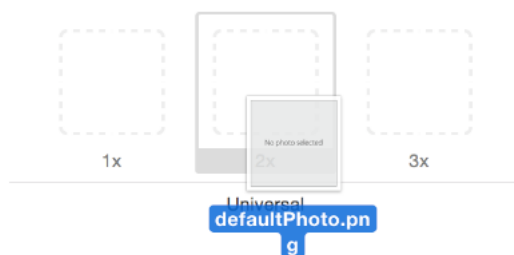


۳. بر روی اسم image set دوبار کلیک کرده و آن را به defaultPhoto تغییر دهید.

۴. عکس دلخواه را از حافظه ی کامپیوتر خود انتخاب کنید.

۵. حال عکس را کشیده و آن را در فضای خالی ۲x در image set جایگذاری کنید.

defaultPhoto



2x، وضوح تصویر نرم افزار شبیه ساز/ iPhone 6 simulator در محیط Xcode

است. از اینرو عکس مورد نظر باید با بهترین کیفیت به نمایش گذاشته شود.

پس از افزودن عکس مکان نگهدار به پروژه، بایستی تنظیمات image view را انجام

دهید تا این المان بتواند عکس مورد نظر را در UI برای کاربر به نمایش بگذارد.

جهت نمایش عکس مکان نگهدار در المان image view، مراحل زیر را گام به گام دنبال نمایید:

۱. فایل storyboard را باز کنید.
  ۲. در storyboard، المان image view را انتخاب نمایید.
  ۳. پس از کلیک بر روی المان مورد نظر، کادر Attributes inspector را از utility area باز کنید.
  ۴. در Attribute inspector، فیلدی که Image نام دارد را پیدا کرده و سپس گزینه ی defaultPhoto را از آن انتخاب نمایید.
- تست کنید:** برنامه را در شبیه ساز اجرا کنید. می بینید که عکس مکان نگهدار در المان image view به نمایش در می آید.

Carrier
9:41 AM

Meal Name

Enter meal name

Set Default Label Text

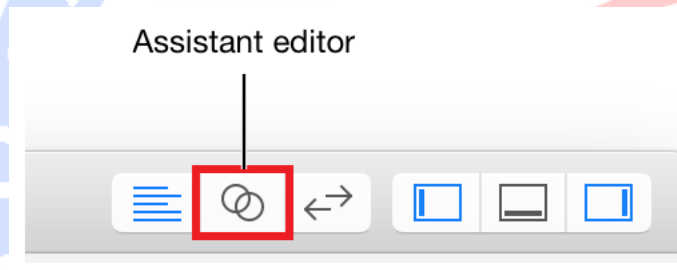
No photo selected

متصل کردن المان image view به کد مربوطه در فایل ViewController.swift

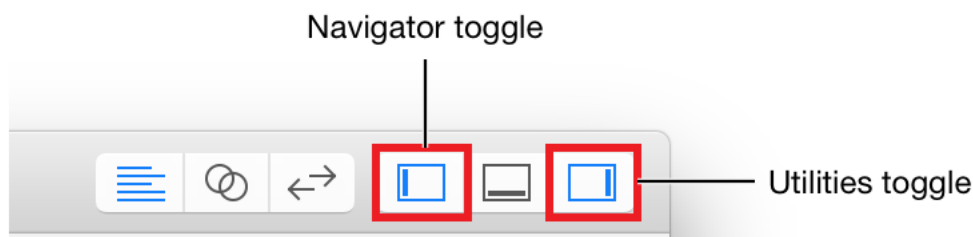
در این بخش بایستی با کدنویسی قابلیت به المان image view اضافه کنید که به شما اجازه می دهد محتوای آن را در زمان اجرای برنامه (runtime)، تغییر دهید. شما باید بتوانید تصویر را از داخل کد (با کدنویسی) تغییر دهید. برای این منظور ابتدا لازم است المان image view را به کد مربوطه در فایل ViewController.swift متصل کنید.

جهت متصل کردن المان image view به کد مرتبط در فایل ViewController.swift، مراحل زیر را گام به گام دنبال نمایید:

۱. بر روی دکمه ی Assistant در نوار ابزار Xcode، بالای محیط گوشه ی سمت راست کلیک نموده و ویرایشگر کمکی (assistant editor) را باز کنید.

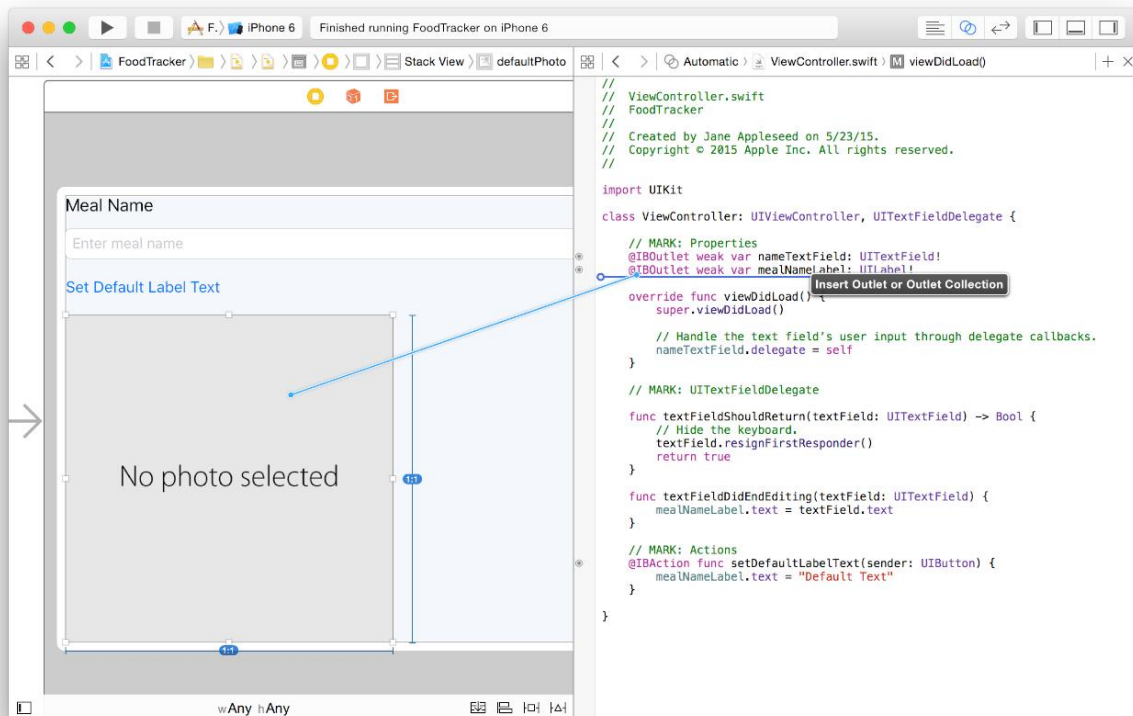


۲. در صورت نیاز به فضای کاری بیشتر، با کلیک بر روی دکمه های navigator toggle و toggle utility در نوار ابزار Xcode، می توانید project navigator و utility area را ببندید.



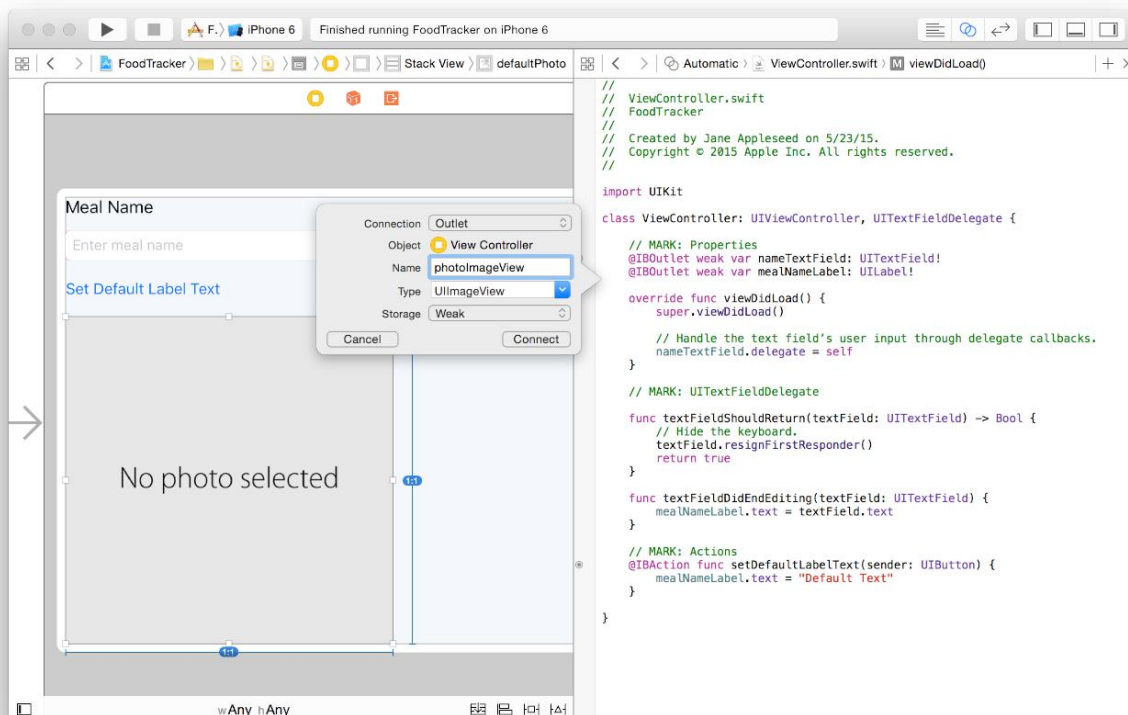
- در صورت تمایل، می توانید کادر outline view را نیز ببندید.
۳. در storyboard، المان image view را انتخاب نمایید.
۴. حال المان نام برده را از سطح scene جاری کشیده و در ناحیه ی ویرایش کد (سمت راست محیط، داخل فایل ViewController.swift) در زیر outlet جاری (متغیر

mealNameLabel) جایگذاری نمایید.



۵. در پنجره ی محاوره ای که به نمایش در می آید، واژه ی `photoImageView` را وارد فیلد `Name` نمایید. لازم نیست سایر تنظیمات را دستکاری کنید. پنجره ی محاوره ای هم اکنون می بایست به صورت زیر باشد:

آموزشگاه تحلیکیر داده



## ۶. حال بر روی دکمه ی Connect کلیک نمایید.

Xcode یک اشاره گر در قالب متغیری با نام photoImageView به المان مزبور در فایل ViewController.swift ایجاد می کند. سپس storyboard را جهت برقراری اتصال بین کد و المان مربوطه در آن تنظیم می نماید.

**@IBOutlet weak var photoImageView: UIImageView!**

حال می توانید از کد به المان image view در آن دسترسی داشته و محتوای آن را تغییر دهید. اما اینجا یک سوال جالب مطرح می شود! از کجا بفهمیم چه زمانی باید عکس یا محتوای المان مزبور را تغییر دهیم؟ شما می بایست روشی تعریف کنید که کاربران از طریق آن بتوانند تمایل خود مبنی بر تغییر محتوای image view را به نحوی بیان کنند، برای مثال با کلیک (قرار دادن انگشت) بر روی image view. پس از آن یک action method تعریف می کنید که به مجرد رخداد tap (کلیک کاربر بر روی المان)، صدا خورده شده و محتوای المان را تغییر می دهد.



گفتنی است که بین view و control یک تفاوت جزئی وجود دارد. همان طور که می دانید control ها نسخه های ویژه از view ها هستند که به شیوه ای خاص به ورودی یا اعمال کاربر واکنش نشان می دهند. در واقع view صرفاً محتوای ساده ای را برای کاربر به نمایش می گذارد، در حالی که یک control امکان ویرایش محتوا (به نحوی خاص) را نیز در اختیار کاربر قرار می دهد.

control (کلاس UIControl) در حقیقت از کلاس UIView ارث بری می کند (یک کلاس فرزند از UIView است). از نمونه های view می توان به label و view image اشاره کرده و از control ها می توان از text field و button نام برد که اگر بخاطر داشته باشید قبلاً با تمامی آن ها در UI برنامه ی خود کار کرده اید.

تعریف یک Gesture Recognizer (اعطای قابلیت های یک control به view) همان طور که در بالا گفته شد، المان image view از نوع control نیست، بنابراین واکنشی که در پاسخ به ورودی کاربر نشان می دهد با یک control – همچون دکمه – یکسان نیست (در اصل طوری طراحی نشده که به عمل کاربر مانند یک کنترل واکنش نشان دهد). به عنوان مثال، نمی توانید به راحتی و بدون هیچ گونه واسطی یک action method تعریف کرده، آن را به image view متصل کنید که به مجرد کلیک کاربر بر روی المان مذکور مستقیماً فراخوانی/اجرا می شود (اگر بر روی المان در interface builder کلیک کرده و آن را به ناحیه ی ویرایش کد بکشید، می بینید که داخل کادر محاوره ای امکان انتخاب گزینه ی Action از فیلد connection وجود ندارد).

خوشبختانه، به راحتی می توانید قابلیت های یک control را به view مورد نظر اعطا کنید. برای این منظور کافی است از gesture recognizer استفاده نمایید. recognizer ها آبجکت هایی هستند که با الماق آن ها به view، به آن ها این امکان داده می شود تا درست مانند یک control به action یا فعل کاربر واکنش نشان دهند.

Gesture recognizer ها حالات مختلف لمس را شناسایی کرده و هر یک را به فعل خاصی تفسیر می کنند. در واقع زمانی که کاربر نمایشگر را با حالت خاصی لمس می کند، iOS آن را مورد بررسی و تحلیل قرار می دهد. سپس آن حالت خاص لمس را معادل یکی

از حالات تعریف شده برای سیستم عامل مورد نظر همچون pinch، swipe یا rotation در نظر می گیرد. در پی آن، متناسب با حالت لمس تفسیر شده، (پس از فراخوانی متد متصل به آن gesture recognizer) عملیات معینی نظیر کوچک/بزرگ کردن صفحه یا چرخاندن تصویر را ترتیب می دهد.

جهت اعطا کردن قابلیت های یک control به view (واکنش نشان دادن به عمل کاربر)، کافی است یک آبجکت gesture recognizer به آن view متصل کنید. سپس یک action method تعریف کنید که به مجرد اینکه gesture recognizer حالت لمس کاربر را شناسایی و تفسیر کرد، آن متد فراخوانی شود.

در این بخش با الصاق یک gesture recognizer به المان image view، قابلیت تعامل با کاربر را به این المان اعطا می کنیم.

ابتدا یک tap gesture recognizer (از کلاس UITapGestureRecognizer) به المان image view الحاق کنید. با این کار شما، زمانی که کاربر با انگشت خود به سطح نمایشگر ضربه می زند، gesture recognizer آن را شناسایی کرده و پس از تفسیر آن (و فراخوانی action method مربوطه) زمینه ی تعامل کاربر با المان مزبور را فراهم می آورد.

جهت افزودن یک tap gesture recognizer به المان image view در storyboard، مراحل زیر را گام به گام دنبال نمایید:

۱. در مرحله ی اول، Object library را باز کنید. (روش سریع برای باز کردن آن:

(. View > Utilities > Show Object Library)

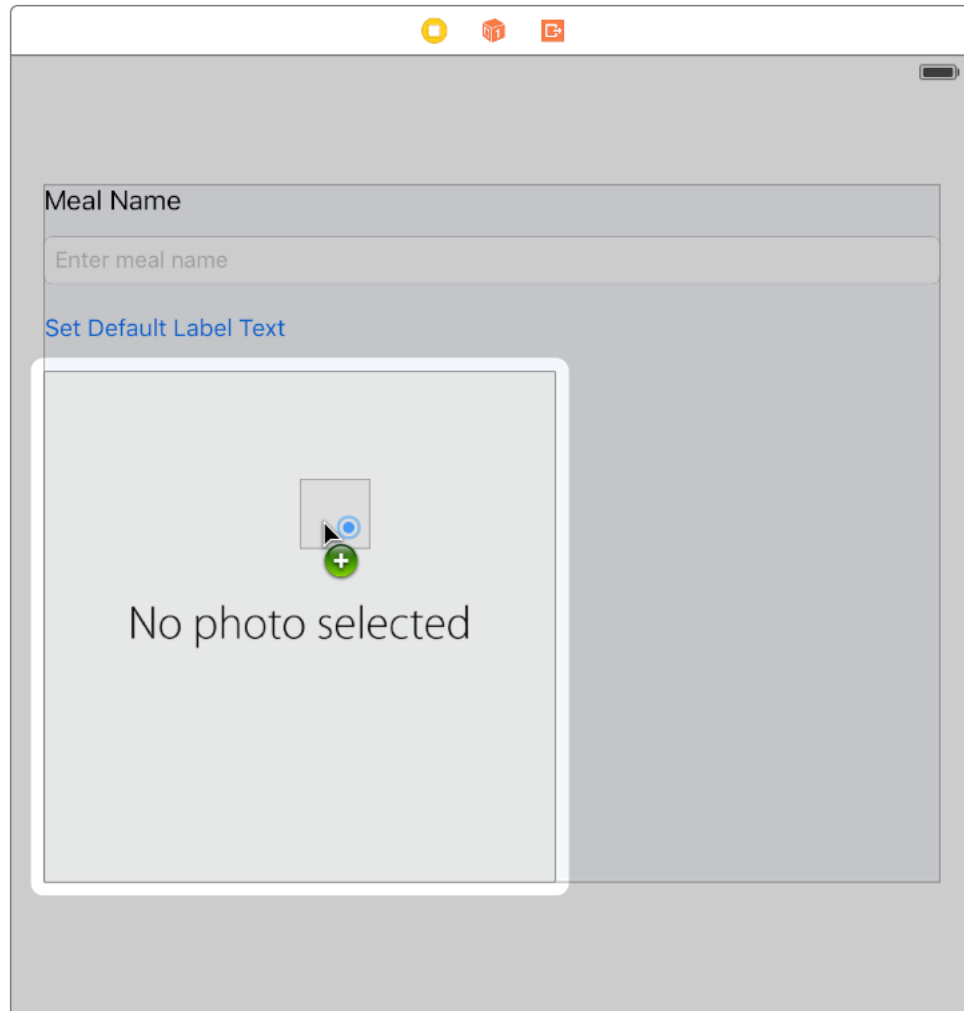
۲. داخل فیلد جستجوی Object library، واژه ی tap gesture را وارد کنید تا محیط

Xcode آبجکت مد نظر را یافته و به سرعت در اختیار شما قرار دهد.

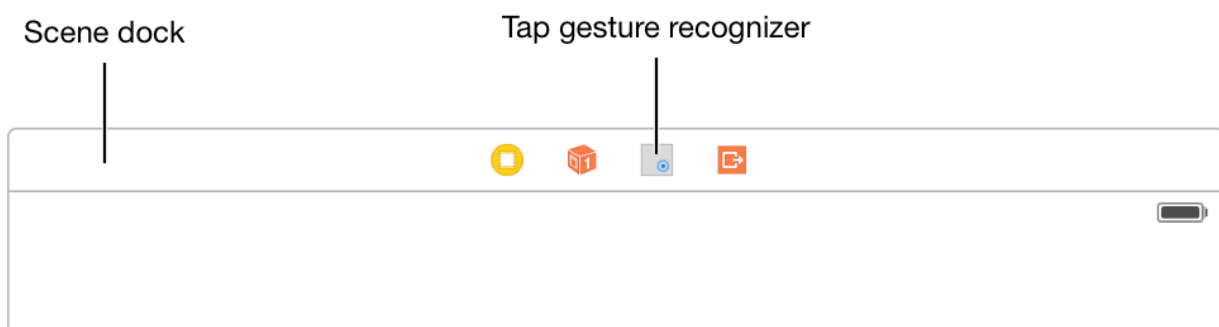
۳. آبجکت Tap Gesture Recognizer را از Object library کشیده و آن را بر روی

المان image view در scene کنونی جایگذاری نمایید.

Attach a tap gesture recognizer (`UITapGestureRecognizer`) to the image view, which will recognize when a user has tapped the image view. You can do this easily in your storyboard.



می بینید که آیکون Tap Gesture Recognizer در scene dock نمایان می شود (نواری در بالای scene که اطلاعاتی را درباره ی آن به نمایش می گذارد).



## متصل کردن gesture recognizer به کد مربوطه در فایل ViewController.swift

اکنون زمان آن فرا رسیده که gesture recognizer را به method action در کد متصل کنید.

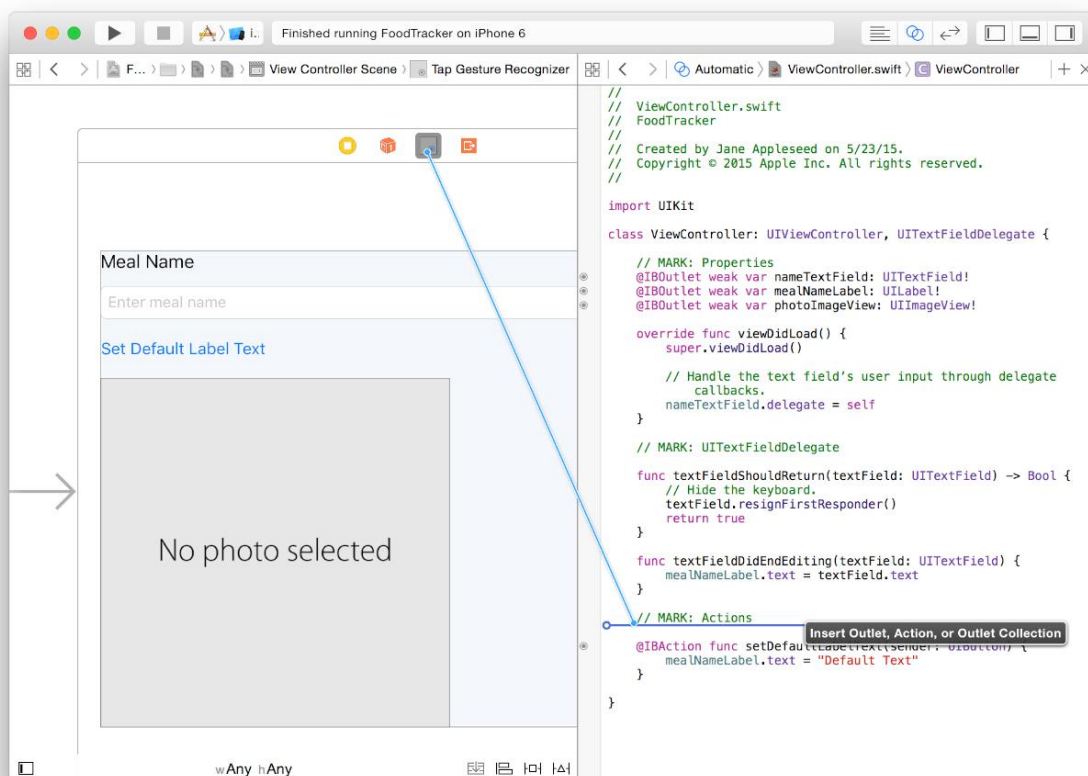
به منظور متصل کردن gesture recognizer به کد مربوطه در فایل ViewController.swift، مراحل زیر را دنبال نمایید:

۱. بر روی آیکنون gesture recognizer در scene dock کلیک کنید، آن را به

ناحیه ی ویرایش کد در سمت راست scene جاری کشیده و در زیر خط: MARK:

(خط توضیحات مربوط به action ها در فایل ViewController.swift) جایگذاری

نمایید:

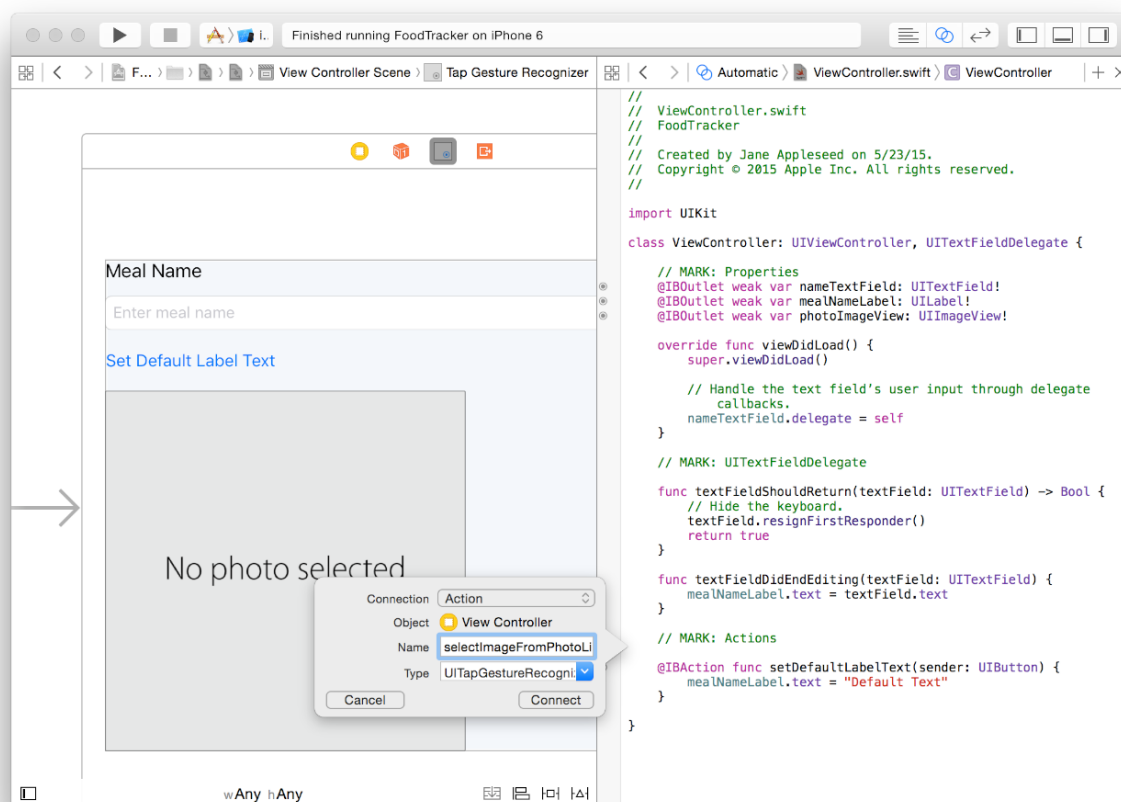


۲. در کادر محاوره ای که نمایان می شود، گزینه ی Action را از فیلد Connection

انتخاب کنید.

۳. در فیلد Name، واژه ی selectImageFromPhotoLibrary را به عنوان اسم متد وارد کنید.

۴. از فیلد Type، گزینه ی UITapGestureRecognizer را انتخاب نمایید. پس از تنظیمات ذکر شده، کادر محاوره ای می بایست به صورت زیر باشد:



۵. حال بر روی دکمه ی Connect کلیک نمایید.

Xcode خود کد مورد نیاز جهت (ایجاد و تنظیم) action method را به فایل ViewController.swift اضافه می کند.

```
@IBAction func selectImageFromPhotoLibrary(sender:
UITapGestureRecognizer) {
}
```

## افزودن یک image picker به پروژه جهت تعامل با کاربر و فراهم آوردن امکان انتخاب عکس

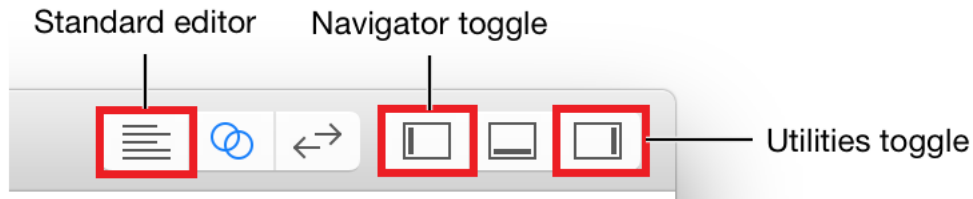
زمانی که کاربر بر روی المان image view در نمایشگر دستگاه ضربه می زند، طبیعتاً باید بتواند یک عکس از مجموعه عکس های آماده در حافظه ی دستگاه انتخاب کند یا عکس دلخواه خود را بگیرد. خوشبختانه، کلاس UIImagePickerControllerController تمامی این رفتارها را به صورت درون ساخته در اختیار دارد.

image picker controller رابط کاربری مورد نیاز جهت گرفتن عکس جدید یا انتخاب از میان عکس های ذخیره شده در حافظه ی دستگاه و استفاده از آن ها در برنامه ی جاری را مدیریت می کند. همان طور که به هنگام کار با المان text field به یک delegate احتیاج دارید، برای کار با image picker نیز، ناگزیر به یک image picker delegate controller نیاز دارید. protocol (الگوی پیاده سازی) ویژه ی این delegate عبارت است از UIImagePickerControllerDelegate. همچنین، آجکتی که به عنوان delegate آجکت image picker منصوب می کنید (آجکتی که با همکاری و هماهنگی آن)، مانند نمونه ی قبلی، ViewController است.

اما پیش از هر چیزی، ViewController می بایست protocol نام برده (UIImagePickerControllerDelegate) را پیاده سازی کند. از آنجایی که ViewController وظیفه ی ارائه ی image picker controller را نیز بر عهده دارد، ناگزیر می بایست پروتکل UINavigationControllerDelegate را هم پیاده سازی کند. این protocol برخی از مسئولیت های پیمایش (navigation) را به view controller واگذار می کند.

به منظور افزودن protocol های نام برده به ViewController، مراحل زیر را دنبال نمایید:

۱. با کلیک بر روی دکمه ی Standard در سمت چپ نوار ابزار Xcode، ویرایشگر اصلی محیط (standard editor) را باز کنید.



حال project navigator و utility area را با کلیک بر روی دکمه های مربوطه ی آن ها (اشاره شده در تصویر فوق) باز نمایید.

۲. از کادر project navigator، فایل ViewController.swift را انتخاب نمایید.

۳. داخل فایل ViewController.swift، خط حامل تعریف کلاس را پیدا کنید:

```
class ViewController: UIViewController, UITextFieldDelegate {
```

۴. بعد از UITextFieldDelegate، یک ویرگول اضافه کرده و سپس UINavigationControllerDelegate را درج کنید.

```
class ViewController: UIViewController, UITextFieldDelegate,
UIImagePickerControllerDelegate {
```

۵. اکنون پس از UIImagePickerControllerDelegate، یک ویرگول اضافه کرده و سپس UINavigationControllerDelegate را تایپ کنید. با این کار کلاس ViewController را ملزم به پیاده سازی متدهای تعریف شده در این دو protocol می کنید:

```
class ViewController: UIViewController, UITextFieldDelegate,
UIImagePickerControllerDelegate, UINavigationControllerDelegate {
```

پس از اضافه کردن این دو protocol به خط تعریف کلاس، می توانید به تکمیل پیاده سازی متد selectImageFromPhotoLibrary(\_\_\_\_) پردازید.

جهت پیاده سازی متد selectImageFromPhotoLibrary(\_\_\_\_)، مراحل زیر را دنبال کنید:

۶. در فایل ViewController.swift، متد

selectImageFromPhotoLibrary(\_\_\_\_) را پیدا کنید. بایستی ظاهری مشابه

زیر داشته باشد:

```
@IBAction func selectImageFromPhotoLibrary(sender:
UITapGestureRecognizer) {
}
```

در بدنه ی متد، بین ({} ) کد زیر را اضافه کنید:

```
// Hide the keyboard.
```

```
nameTextField.resignFirstResponder()
```

این کد سبب می شود اگر کاربر حین نمایان بودن صفحه کلید در نمایشگر، بر روی المان

image view کلیک کرد، صفحه کلید به طور کامل از صفحه حذف شود.

۱. حال کد زیر را جهت ایجاد یک image picker controller، به بدنه ی متد اضافه نمایید:

```
// UIImagePickerController is a view controller that lets a user pick media from
their photo library.
```

```
let imagePickerController = UIImagePickerController()
```

۲. سپس کد زیر را به بدنه ی متد اضافه کنید:

```
// Only allow photos to be picked, not taken.
```

```
imagePickerController.sourceType = .PhotoLibrary
```

این خط کد، محلی که image picker controller فایل های تصویری خود را از آن

وارد المان می کند را مشخص می نماید. پارامتر PhotoLibrary به image picker

اعلان می کند که باید عکس های خود را فقط از مجموعه تصاویر محیط شبیه ساز

(roll camera simulator) وارد کنند. می دانیم که

imagePickerController.sourceType از جنس

UIImagePickerControllerSourceType است که نوع داده ای آن



enumeration می باشد. بنابراین می توانید بجای اینکه مقدار را به طور کامل UIImagePickerControllerSourceType.PhotoLibrary تایپ کنید، آن را به صورت مختصر PhotoLibrary. مورد اشاره قرار دهید. یادآور می شویم که هرگاه نوع مقدار enumeration از قبل مشخص است، می توانید از صورت مختصر آن استفاده نمایید.

۳. با افزودن کد زیر، کلاس حاضر (ViewController) را به عنوان delegate کنترلر image picker انتخاب نمایید:

// Make sure ViewController is notified when the user picks an image.

imagePickerController.delegate = self

۴. حال این کد را در زیر کد فوق، به بدنه ی متد اضافه کنید:

presentViewController(imagePickerController, animated: true, completion: nil)

presentViewController(\_:animated:completion:) یک متد است که در سطح کلاس ViewController فراخوانی می شود. اگرچه این متد صریحا به آبجکت self الحاق نشده، اما از شرایط جاری مشخص است که بر روی آن فراخوانی می شود. متد مذکور به کلاس ViewController اعلان می کند که باید view controller را نمایش دهد.

ارسال مقدار true به پارامتر animated، سبب می شود image picker controller به صورت پویا (با انیمیشن) ارائه شود. پارامتر completion به یک completion handler اشاره دارد. Completion handler یک تابع closure است که با اتمام اجرای متد جاری، صدا زده می شود. از آنجایی که پس از اجرای کامل متد، نیاز به انجام عملیات دیگری نیست، مقدار nil را به این پارامتر پاس می دهیم.

در حال حاضر بدنه ی متد selectImageFromPhotoLibrary(\_\_\_\_\_) بایستی مشابه نمونه ی زیر باشد:

```

@IBAction func selectImageFromPhotoLibrary(sender:
UITapGestureRecognizer) {
// Hide the keyboard.
nameTextField.resignFirstResponder()
// UIImagePickerController is a view controller that lets a user pick media from
their photo library.
let imagePickerController = UIImagePickerController()
// Only allow photos to be picked, not taken.
imagePickerController.sourceType = .PhotoLibrary
// Make sure ViewController is notified when the user picks an image.
imagePickerController.delegate = self
presentViewController(imagePickerController, animated: true, completion: nil)
}

```

پس از اینکه image picker controller به نمایش گذاشته شد، رفتارش (عملیاتی که قادر به اجرای آن ها است) به آبجکت delegate محول می شود.

برای این که به کاربران این امکان را بدهید تا پس از کلیک بر روی المان، یک عکس را انتخاب کنند، لازم است دو متد زیر از پروتکل UIImagePickerControllerDelegate را پیاده سازی نمایید:

```

func imagePickerControllerDidCancel(picker: UIImagePickerController)
func imagePickerController(picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String :
AnyObject])

```

متد اول، imagePickerControllerDidCancel( \_:)، زمانی صدا خورده می شود که کاربر بر روی دکمه ی Cancel از image picker کلیک می کند. این متد در واقع به شما اجازه می دهد تا UIImagePickerController را از صفحه حذف/محو کنید (و در صورت تمایل، کدی اضافه کنید که عملیات cleanup و پاک سازی داده های غیرضروری از حافظه را انجام می دهد).

جهت پیاده سازی متد imagePickerControllerDidCancel( \_:)، مراحل زیر را به ترتیب دنبال نمایید:

۱. داخل فایل ViewController.swift، درست در بالای MARK: Actions، خط زیر را درج نمایید:

```
// MARK: UIImagePickerControllerDelegate
```

این کد صرفاً یک comment است که شما یا هر شخص دیگری که کد برنامه‌ی شما را می‌خواند را در پیمایش در پروژه و فهم کاربرد بخش‌های مختلف آن یاری می‌کند. این comment اعلان می‌کند که این بخش از کد مربوط به پیاده‌سازی متدهای image picker می‌باشد.

۲. حال در زیر comment، متد زیر را وارد نمایید:

```
func imagePickerControllerDidCancel(picker: UIImagePickerController) {
}
```

۳. داخل بدنه‌ی متد دستور زیر را تایپ کنید:

```
// Dismiss the picker if the user canceled.
dismissViewControllerAnimated(true, completion: nil)
```

این کد، فرایند حذف شدن image picker controller از نمایشگر را پویانمایی می‌کند (با انیمیشن آن را نظر کاربر پنهان می‌کند).

بدنه‌ی متد imagePickerControllerDidCancel(\_\_\_\_\_) هم اکنون بایستی مشابه نمونه‌ی زیر باشد:

```
func imagePickerControllerDidCancel(picker: UIImagePickerController) {
// Dismiss the picker if the user canceled.
dismissViewControllerAnimated(true, completion: nil)
}
```

دومین متدی که باید از پروتکل UIImagePickerControllerDelegate پیاده‌سازی کنید، \_\_\_\_\_،

imagePickerController(\_:didFinishPickingMediaWithInfo:\_\_\_\_\_) می‌باشد.

این متد زمانی فرخوانی می‌شود که کاربر عکس دلخواه خود را انتخاب می‌نماید.

با اجرا شدن متد نام برده، این فرصت در اختیار شما قرار می گیرد تا بر روی عکس یا عکس هایی که کاربر انتخاب کرده، عملیات مورد نظر را انجام دهید. در مثال جاری، شما عکس مورد انتخاب کاربر را گرفته و در آل به نمایش می گذارید.

جهت پیاده سازی متد

(\_:\_didFinishPickingMediaWithInfo:)، imagePickerController، مراحل زیر را

به ترتیب دنبال نمایید:

۱. در زیر متد (\_:\_didCancel)، imagePickerController، این کد را درج نمایید:

```
func imagePickerController(picker: UIImagePickerController,
didFinishPickingMediaWithInfo info: [String : AnyObject]) {
}
```

۲. در بدنه ی متد، کد زیر را وارد نمایید:

// The info dictionary contains multiple representations of the image, and this uses the original.

let selectedImage = info[UIImagePickerControllerOriginalImage] as! UIImage

info در کد فوق، یک dictionary است که علاوه بر عکس اولیه و انتخاب شده در image picker،

نسخه ی ویرایش شده ی آن را نیز دربر می گیرد. در اینجا از نسخه ی ساده و ویرایش نشده ی

عکس برای غذای مورد نظر استفاده می کنیم. کد حاضر نیز همین عکس (نسخه ی ویرایش نشده)

را داخل ثابت selectedImage ذخیره می کند.

۳. این کد را به بدنه ی متد اضافه کنید تا عکس مورد نظر در متغیر photoImageView

(یک outlet که قبلاً به فایل اضافه کردید) قرار داده شده و برای کاربر به نمایش درآید.

// Set photoImageView to display the selected image.

photoImageView.image = selectedImage

۴. حال با افزودن کد زیر به متد، image picker را از حالت نمایش خارج کنید:

// Dismiss the picker.

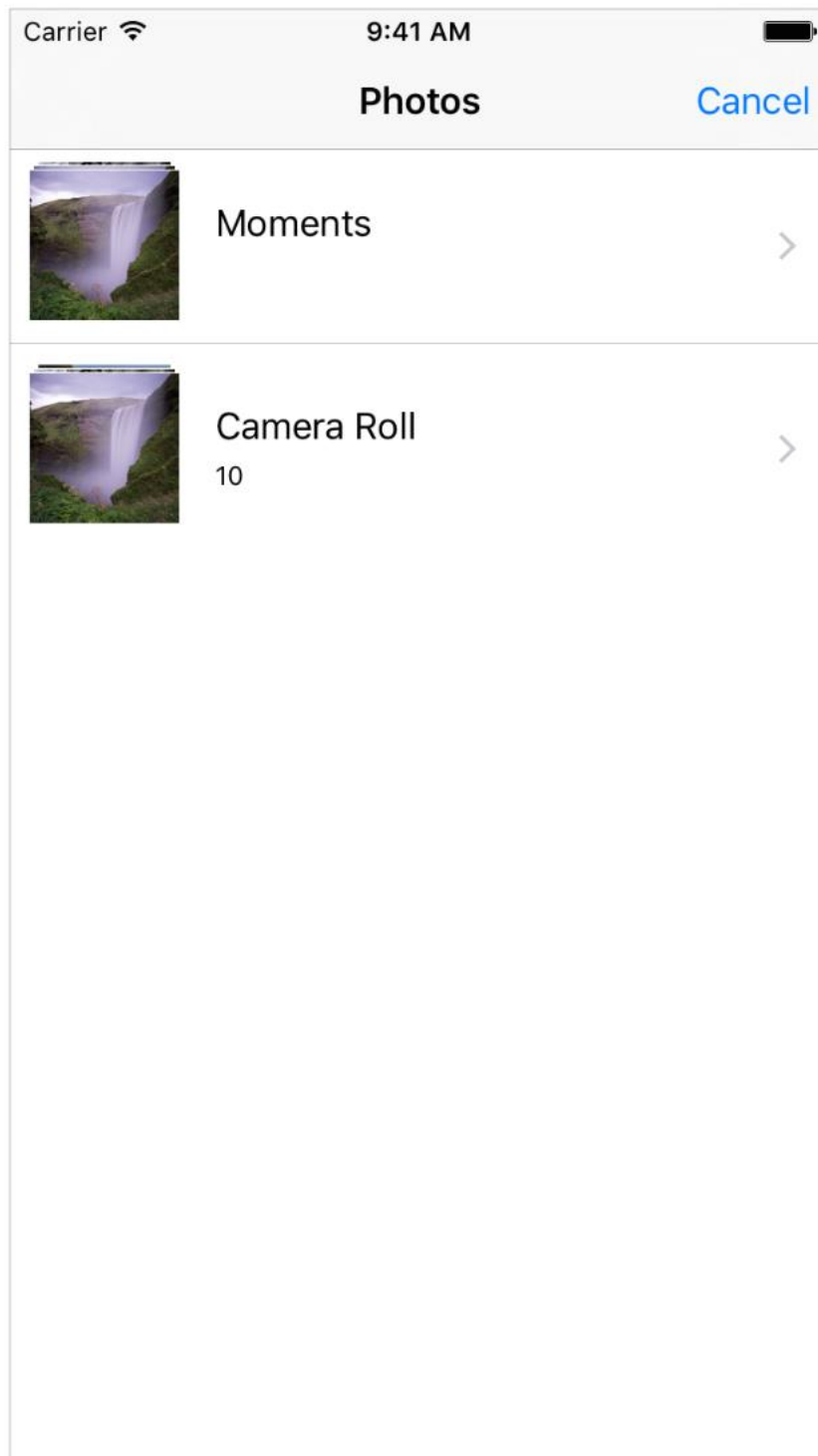
dismissViewControllerAnimated(true, completion: nil)

در حال حاضر بدنه ی متد (imagePickerController(\_:didFinishPickingMediaWithInfo))

شما بایستی مشابه زیر باشد:

```
func imagePickerController(picker: UIImagePickerController,
didFinishPickingMediaWithInfo info: [String : AnyObject]) {
// The info dictionary contains multiple representations of the image, and this
uses the original.
let selectedImage = info[UIImagePickerControllerOriginalImage] as!
UIImage
// Set photoImageView to display the selected image.
photoImageView.image = selectedImage
// Dismiss the picker.
dismissViewControllerAnimated(true, completion: nil)
}
```

**تست کنید:** برنامه را اجرا کنید. بایستی پس از کلیک بر روی image view، یک image picker به نمایش درآمده و به شما اجازه ی انتخاب عکس دلخواه را بدهد. برای اینکه برنامه FoodTracker بتواند به بخش Photos دسترسی داشته و به کاربر اجازه ی انتخاب عکس را بدهد، باید بر روی دکمه ی OK در کادر محاوره ای کلیک نمایید. پس از آن، با توجه به کدی که برای برنامه نوشته شده، باید بتوانید با کلیک بر روی Cancel، انتخابگر عکس (image picker) را ببندید یا Camera Roll را باز کرده، عکس دلخواه را از آن انتخاب نمایید و متعاقباً در image view به نمایش بگذارید.



اگر به عکس های موجود در محیط شبیه ساز (Simulator) نگاه کنید، متوجه می شوید که عکسی در رابطه با غذا در آن وجود ندارد. می توانید عکس های مد نظر خود را مستقیماً وارد محیط شبیه ساز نموده و در `image view` نمایش دهید یا پس از دانلود

پروژه ی این آموزش، به پوشه ی Images/ مراجعه نموده و از عکس نمونه ی داخل این فولدر استفاده کنید.

جهت افزودن عکس جدید به شبیه ساز، مراحل زیر را دنبال نمایید:

۱. ابتدا برنامه را در محیط شبیه ساز اجرا کنید.
۲. عکس دلخواه را انتخاب نمایید.
۳. عکس مورد نظر را با اشاره گر موس کشیده و در محیط شبیه ساز جایگذاری کنید.

## درس ۶ : کنترل اختصاصی (Custom control) در Swift

پیاده سازی یک کنترل اختصاصی (Custom control)

در این مبحث، شما یک کنترل جهت امتیاز دهی (rating control) برای برنامه ی FoodTracker پیاده سازی خواهید کرد. در پایان، برنامه ی شما ظاهری مشابه نمونه زیر خواهد داشت:

آموزشگاه تحلیکیر داده

Carrier 9:41 AM

Meal Name

Enter meal name

No photo selected

★★★★☆

آنچه خواهید آموخت

- ایجاد المان های UI اختصاصی به همراه فایل های source code مرتبط. متصل کردن المان های UI در storyboard به کد آن ها.
- تعریف کلاس اختصاصی.
- پیاده سازی یک متد initializer (مقداردهنده ی اولیه) داخل کلاس اختصاصی مورد نظر.
- استفاده از کلاس UIView به عنوان یک ظرف.



- نحوه ی به نمایش گذاشتن محتوای view ها برای کاربر با کدنویسی (programmatically).

### تعریف یک View اختصاصی

برای اعطا کردن قابلیت امتیاز دهی به غذاهای موجود در برنامه به کاربر، لازم است یک control تعریف کنید که تعدادی ستاره نمایش داده و کاربر برای نظر دادن درباره ی غذا باید این ستاره ها را پر کند. روش های مختلفی برای پیاده سازی این قابلیت وجود دارد. در آموزش حاضر از این روش استفاده خواهیم کرد: یک view اختصاصی با کد تعریف کرده و سپس آن را در storyboard خود مورد استفاده قرار خواهیم داد.

در زیر نمایی از کنترلی که جهت نمایش و تخصیص امتیاز به غذا پیاده سازی خواهیم کرد را مشاهده می کنید:



کنترل نام برده به کاربران این امکان را می دهد تا بر اساس میزان محبوبیت غذا، به آن ستاره تخصیص دهند. همان طور که می بینید این کنترل در کل ۵ ستاره نمایش می دهد که پنجمین ستاره طبیعتاً نشانگر بالاترین امتیاز برای آن غذا می باشد. زمانی که کاربر بر روی یک ستاره ضربه می زند، تمامی ستاره های منتهی به آن (از جمله خود ستاره ی انتخاب شده) از سمت چپ پر می شوند.

برای اقدام به طراحی ظاهر (UI)، قابلیت تعامل با کاربر و رفتار این کنترل، بایستی ابتدا یک کلاس فرزند view با پیاده سازی اختصاصی از UIView ایجاد کنید (یک custom view subclass از کلاس پایه UIView ایجاد نمایید).

جهت ایجاد یک کلاس فرزند از UIView، مراحل زیر را گام به گام دنبال نمایید:

۱. این مسیر را طی کنید: File > New > File یا Command-N را فشار دهید.
۲. یک کادر محاوره ای نمایان می شود. در سمت چپ آن، گزینه ی iOS را انتخاب نمایید.

۳. گزینه ی Cocoa Touch Class را انتخاب نموده و بر روی Next کلیک کنید.

۴. داخل فیلد Class، عبارت RatingControl را وارد نمایید.

۵. حال گزینه ی UIView را از فیلد “Subclass of” انتخاب نمایید.

۶. زبان برنامه نویسی پروژه را بر روی Swift تنظیم نمایید.

Choose options for your new file:

Class: RatingControl

Subclass of: UIView

☐ Also create XIB file

iPhone

Language: Swift

Cancel Previous Next

۷. بر روی دکمه ی Next کلیک کنید.

محل ذخیره ی فایل به صورت پیش فرض بر روی دایرکتوری پروژه ی جاری شما تنظیم می شود.

گزینه ی Group به صورت پیش فرض بر روی اسم برنامه ی فعلی، FoodTracker تنظیم می شود.

در بخش Targets، می بینید که برنامه ی شما انتخاب شده اما تست های مربوط به آن انتخاب نشده اند.

۸. لازم نیست تنظیمات پیش فرض را دستکاری نمایید. کافی است بر روی Create کلیک

کرده تا فایل مورد نظر ایجاد گردد.

Xcode یک فایل جدید، حاوی RatingControl.swift class: RatingControl ایجاد می کند. RatingControl یک کلاس اختصاصی view، ارث بری شده از کلاس پایه ی UIView است (منظور از کلاس فرزند اختصاصی، کلاسی است که متدهای کلاس پایه در آن بازنویسی شده باشند).

۹. در فایل RatingControl.swift، تمامی comment هایی که همراه با قالب آماده

(template) و پیاده سازی الگو (template implementation) ارائه می شوند را حذف نمایید.

در حال حاضر پیاده سازی کلاس بایستی مشابه زیر باشد:

```
import UIKit
class RatingControl: UIView {
}
```

View معمولاً به دو روش زیر ایجاد می شود: روش اول عبارت است مقدار دهی اولیه ی یک view با یک فریم که به شما امکان می دهد view را به صورت دستی به UI اضافه کنید و روش دوم، عبارت است از واگذار کردن بارگذاری view به storyboard. برای هر یک از روش های نام برده یک متد initializer اختصاصی وجود دارد: به منظور مقداردهی اولیه و ایجاد یک فریم برای view از متد init(frame:) و برای روش دوم، محول کردن بارگذاری view به storyboard، از متد initWithCoder() استفاده می کنیم. یادآور می شویم که initializer متدی است که یک نمونه از روی کلاس جاری می سازد، property های آن کلاس را مقداردهی اولیه نموده و در صورت لزوم سایر تنظیمات آغازین را انجام می دهد.

در آموزش حاضر، از روش دوم اقدام به ایجاد view خواهیم نمود (در storyboard با view خود کار خواهیم کرد)، از این رو لازم است ابتدا (پیاده سازی) متد initWithCoder() کلاس والد view را override (بازنویسی) نمایید.

به منظور بازنویسی پیاده سازی متد initializer، مراحل زیر را گام به گام دنبال نمایید:

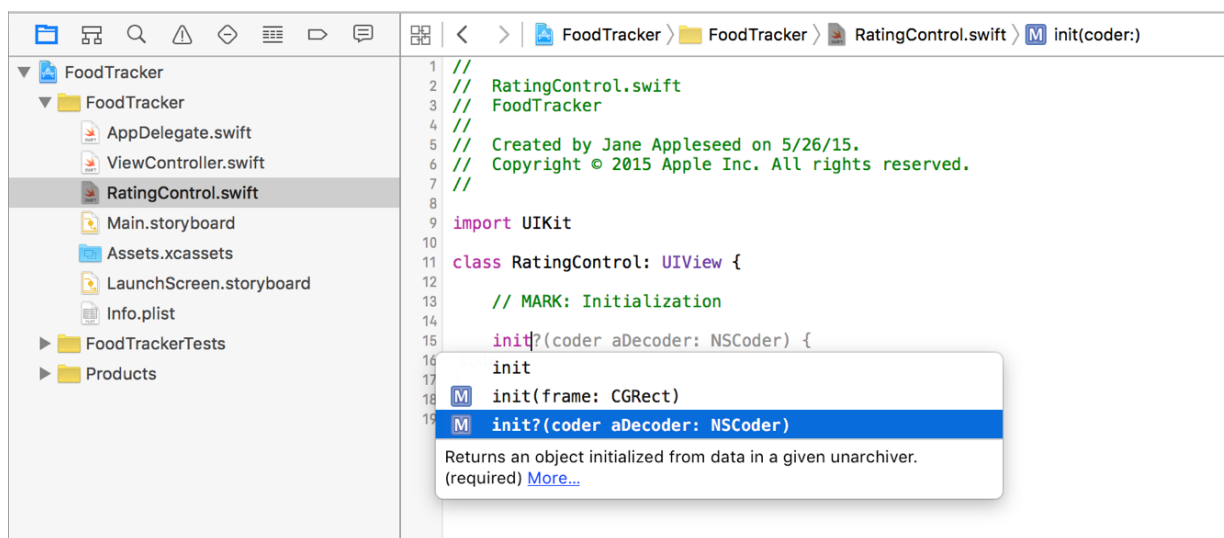
۱. داخل فایل RatingControl.swift، در زیر خط تعریف کلاس، این comment را درج

نمایید:

// MARK: Initialization

۲. حال در زیر این comment، ابتدا واژه ی init را تایپ کنید.

خواهید دید که ابزار پیشبینی و تکمیل کد (code completion) محیط کاری Xcode نمایان می شود.

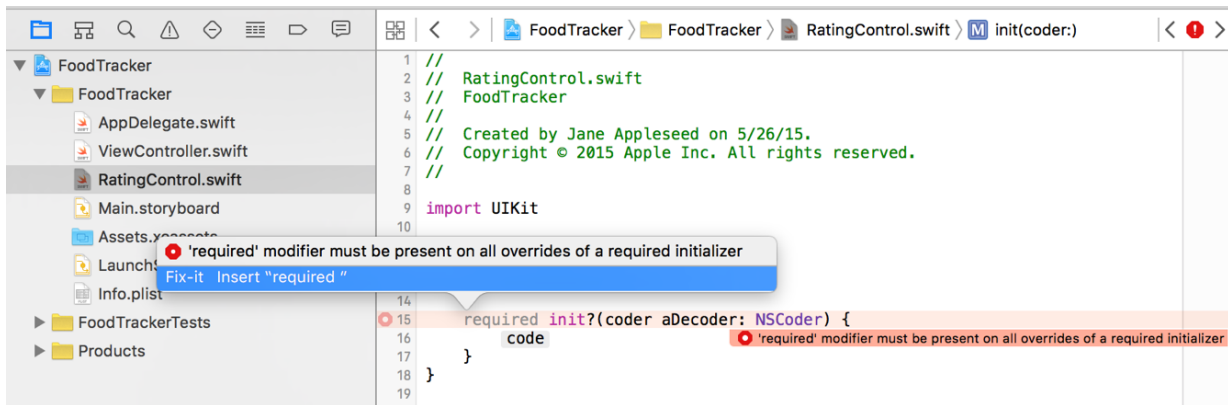


۳. سومین متد را در لیستی که پدیدار می شود (init?(coder:)) انتخاب کرده و سپس کلید Return را فشار دهید.

```
init?(coder aDecoder: NSCoder) {
}
```

Xcode خود اسکلت (کد لازم) متد را برای شما درج می کند.

۴. یک پیغام خطا با آیکون قرمز رنگ به نمایش در می آید. بر روی Fix-it کلیک کرده تا Xcode کلیدواژه ی required را به متد initializer اضافه کند.



```
required init?(coder aDecoder: NSCoder) {
}
```

تمامی کلاس های ارث بری شده از UIView (کلاس های فرزند آن) که متد initializer را پیاده سازی می کنند، می بایست متد init?(coder:) را نیز در بدنه ی خود داشته و پیاده سازی کنند. کامپایلر زبان Swift به این امر واقف بوده و راه حل خود (جهت ویرایش کد) را در قالب fix-it برای شما ارائه می کند. Fix-it ها راه حل هایی هستند که کامپایلر در جواب خطاهای رخ داده به هنگام کدنویسی در اختیار شما قرار می دهد.

۱. کد زیر را جهت فراخوانی متد initializer کلاس والد اضافه نمایید.

```
super.init(coder: aDecoder)
```

۲. هم اکنون بدنه ی متد init?(coder:) بایستی دربردارنده ی پیاده سازی زیر باشد:

```
required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
}
```

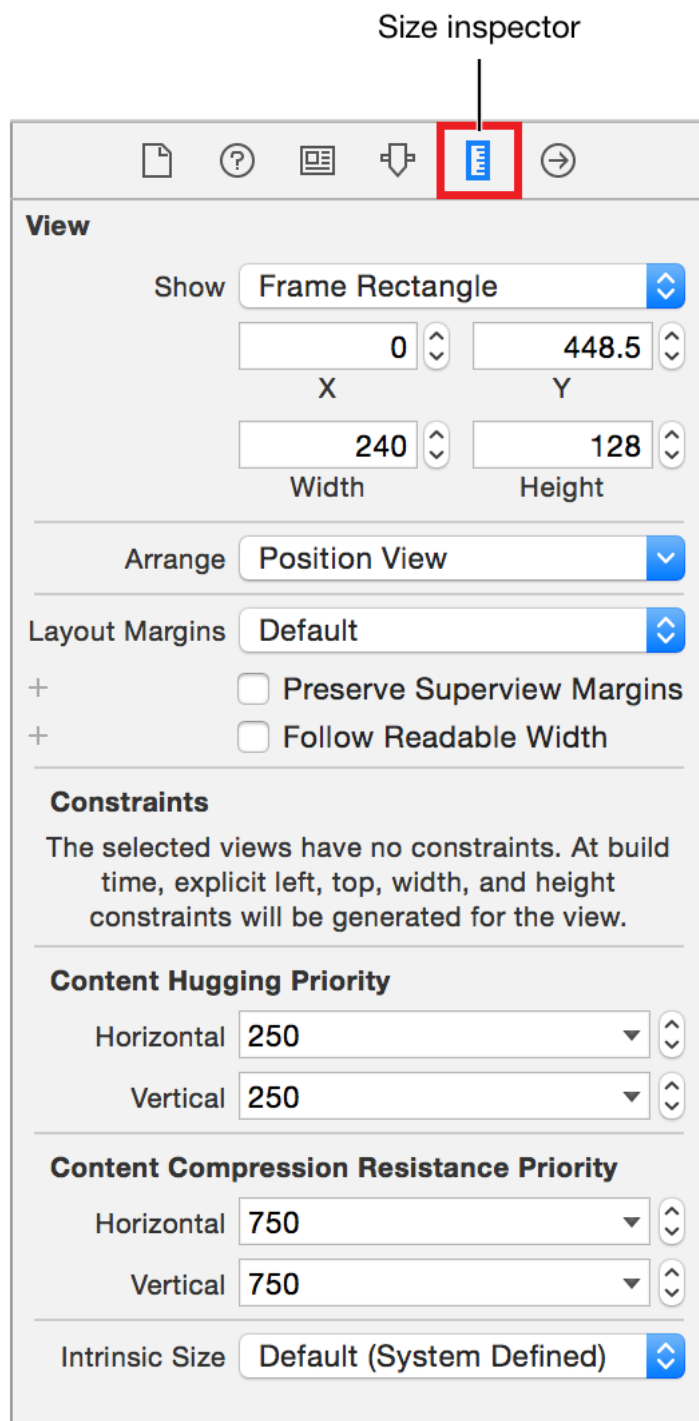
به نمایش گذاشتن view اختصاصی

جهت نمایش view دلخواه، می بایست یک view به UI خود اضافه کرده، سپس

ارتباطی بین view مورد نظر و کد متناظر آن برقرار نمایید.

جهت به نمایش گذاشتن محتوای view اختصاصی خود در رابط کاربری، مراحل زیر را به ترتیب دنبال نمایید:

۱. ابتدا فایل storyboard را باز نمایید.
  ۲. داخل storyboard، یک View object از کادر Object library انتخاب کرده، آن را بکشید و در سطح scene جاری جایگذاری نمایید، به طوری که این المان در قالب stackview و زیر آبجکت image view قرار گیرد.
  ۳. پس از انتخاب view مورد نظر، کادر Size inspector را با کلیک بر روی آیکون ، در utility area باز نمایید.
- یادآور می شویم که Size inspector با کلیک بر روی دکمه ی پنجم از سمت چپ inspector selector bar باز می شود. در این کادر شما می توانید اندازه و مکان قرار گیری آبجکت مورد نظر را ویرایش نمایید.

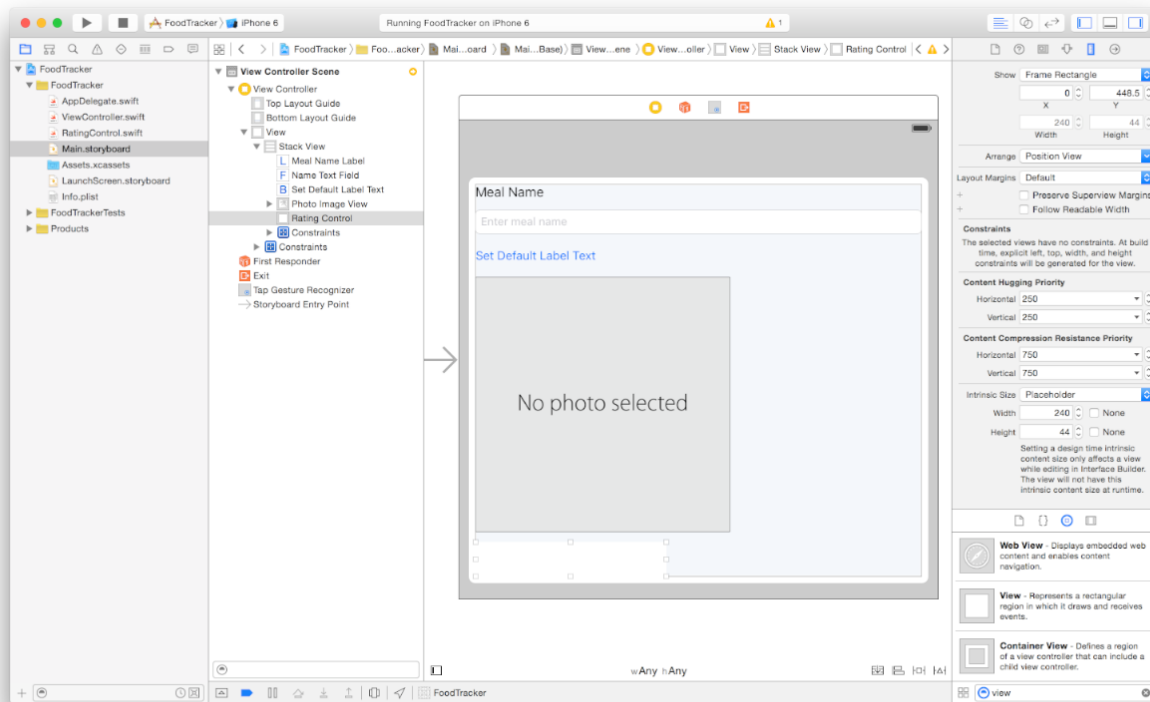


۴. از منوی **intrinsic size**، گزینه **Placeholder** را انتخاب نمایید.

۵. در زیر منوی **Intrinsic Size**، داخل فیلد **Height** مقدار ۴۴ و داخل فیلد **Width**

مقدار عددی ۲۴۰ را وارد کنید. اکنون کلید **Return** را فشار دهید.

در حال حاضر، **UI** برنامه می بایست ظاهری مشابه زیر داشته باشد.

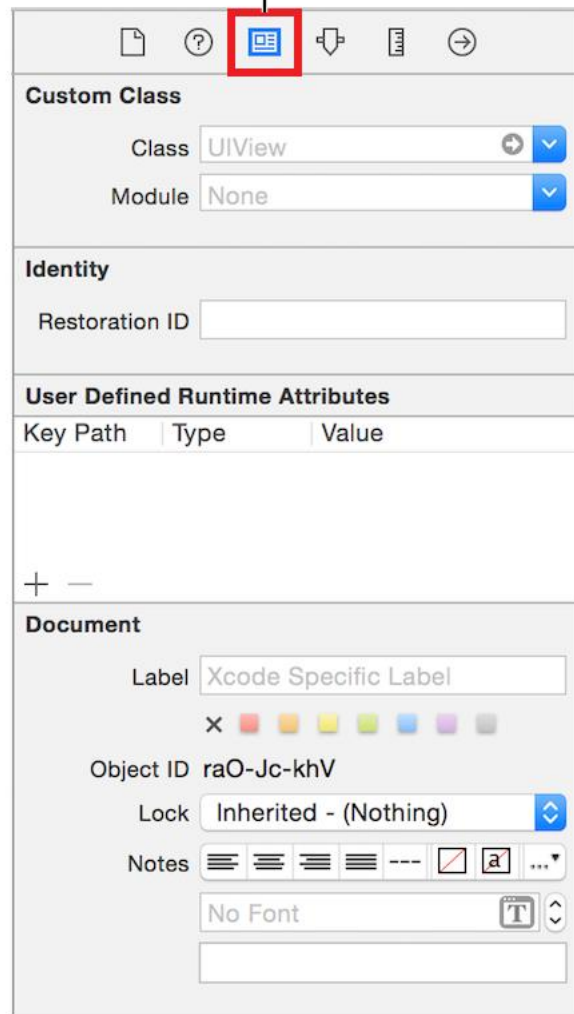


۶. پس از انتخاب view مورد نظر، کادر Identity inspector را باز نمایید.

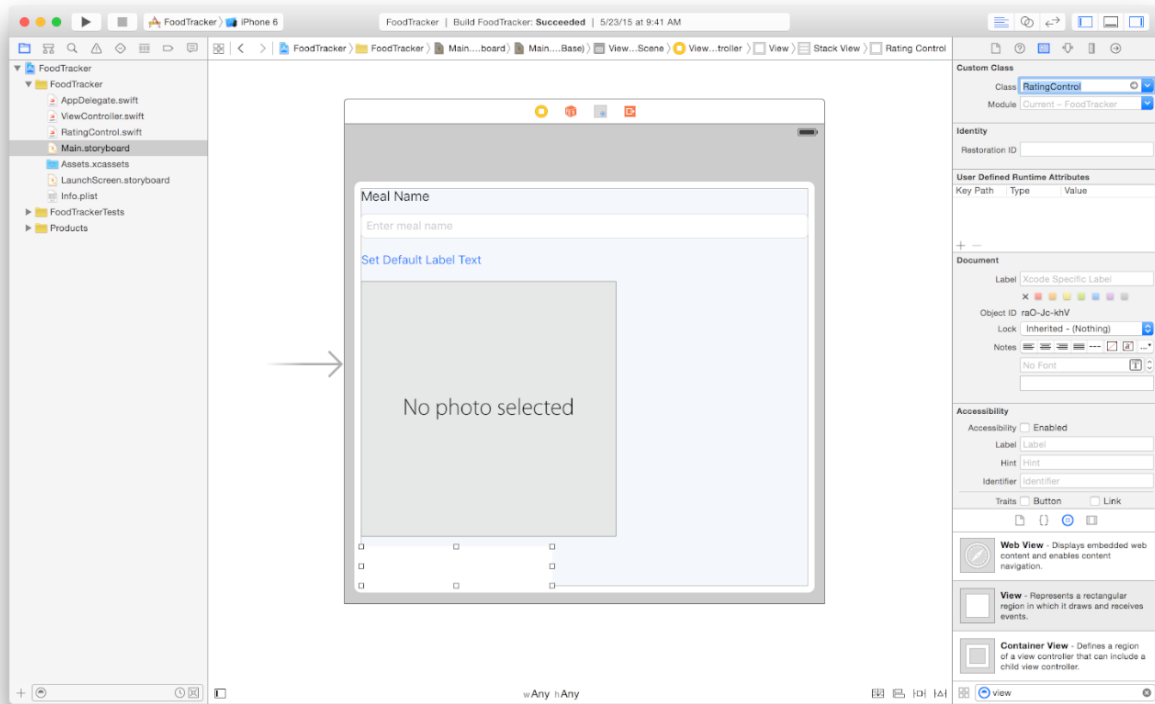
یادآور می شویم که Identity inspector به شما این امکان را می دهد تا آن دسته از property ها و ویژگی های یک آبجکت که با identity آن مرتبط است، نظیر اینکه به کدام کلاس تعلق دارد را ویرایش نمایید.



Identity inspector



۷. داخل این کادر، فیلدی که Class نام دارد را یافته و آن را بر روی RatingControl تنظیم نمایید.



## افزودن کنترل Button به View

تا اینجا، اصول اولیه ی یک کلاس فرزند از UIView با پیاده سازی اختصاصی خود (به نام RatingControl) را پائیزی کردید. حال در این بخش تعدادی دکمه به view دلخواه خود اضافه می کنید که کاربر با استفاده از آن ها می تواند به غذای مورد نظر امتیاز بدهد.

کار را با اضافه کردن یک دکمه ی ساده قرمز رنگ آغاز می کنیم که در view به نمایش در می آید.

جهت ایجاد یک دکمه در view اختصاصی خود، مراحل زیر را دنبال کنید:

۱. داخل بدنه ی متد `init?(coder:)`، دستورات زیر را جهت ایجاد یک دکمه ی قرمز رنگ وارد نمایید:

```
let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
button.backgroundColor = UIColor.redColor()
```

متد `redColor()`، رنگ کنترل دکمه را بر روی قرمز تنظیم می کند. در صورت تمایل می توانید سایر مقادیر پیش فرض `UIColor` نظیر `blueColor()` یا `greenColor()` را بکار ببرید.

۲. در زیر آخرین خط، کد زیر را وارد کنید:

`addSubview(button)`

متد `addSubview()`، دکمه ای که تعریف کردید را به کلاس `RatingControl` اضافه می کند.

پیاده سازی متد `init?(coder:)` در بدنه بایستی به صورت زیر باشد:

```
required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)

    let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
    button.backgroundColor = UIColor.redColor()
    addSubview(button)
}
```

برای اینکه به `stack view` اعلان کنید چگونه و با چه اندازه ای کنترل دکمه را نمایش دهد، لازم است `intrinsic content size` آن را مشخص نمایید. برای نیل به این هدف، کافی است پیاده سازی متد `intrinsicContentSize` را مانند زیر بازنویسی کرده تا پارامترهای تنظیم کننده ی اندازه ی دکمه با اندازه ای که قبلاً در `Interface Builder` (برای کنترل مربوط) مشخص کردید، همخوانی داشته باشد:

```
override func intrinsicContentSize() -> CGSize {
    return CGSize(width: 240, height: 44)
}
```

**تست کنید:** برنامه ی خود را اجرا کنید. برنامه هم اکنون باید یک مربع قرمز رنگ کوچک در `view` به نمایش بگذارد. این مربع قرمز رنگ همان کنترل دکمه است که در بدنه ی متد `initializer` اضافه کردید.

Carrier 9:41 AM

Meal Name

Enter meal name

Set Default Label Text

No photo selected

این دکمه و دکمه های دیگری که به برنامه اضافه می کنید، می بایست با کلیک کاربر، یک action را صدا زده و عملیات خاصی را انجام دهند. آن action تغییر امتیاز یک غذا است.

به منظور اضافه کردن یک action به کنترل دکمه، مراحل زیر را به ترتیب دنبال نمایید:

۱. داخل فایل RatingControl.swift، قبل از آخرین ({})، comment زیر را وارد کنید:

// MARK: Button Action

در زیر comment، تابع زیر را وارد کنید:

```
func ratingButtonTapped(button: UIButton) {
    print("Button pressed 🍷")
}
```

در زمان حاضر، تابع print() را صرفاً جهت کسب اطمینان از متصل بودن متد ratingButtonTapped( \_: ) به دکمه ی مورد نظر فراخوانی می کنیم. این تابع یک پیغام را در خروجی چاپ می کند. در اینجا منظور از خروجی همان ابزار console در پایین محیط کاری Xcode است که به منظور ثبت گزارش (درباره ی خطاها) و اشکال زدایی کد مورد استفاده قرار می گیرد.

به زودی پیاده سازی این متد را که در حال حاضر صرفاً پیغامی را در console چاپ می کند، با یک عملیات (پیاده سازی) واقعی جایگزین خواهید نمود.

۲. متد init?(coder:) را پیدا کنید:

```
required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
    button.backgroundColor = UIColor.redColor()
    addSubview(button)
}
```

۳. قبل از متد addSubview(button)، این کد را درج نمایید:

```
button.addTarget(self, action:
    #selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
    .TouchDown)
```

از قبل با الگوی target-action آشنایی دارید و از آن برای متصل کردن المان های رابط کاربری در storyboard به action method ها در کد استفاده کردید. در بالا همین کار

را انجام می دهید، با این تفاوت که اتصال بین کد و المان UI در storyboard را با کدنویسی برقرار می کنید. بدین صورت که شما متد `(_:ratingButtonTapped)` را به آبجکت `button` متصل می کنید و به مجرد اینکه کاربر بر روی دکمه کلیک می کند، رخداد `TouchDown`. اتفاق افتاده و به دنبال آن متد مذکور اجرا می شود. این رخداد، همان طور که از نامش پیدا است، به برنامه اعلان می کند که کاربر دکمه مورد نظر را فشار داده است.

از آنجایی که `action` را در سطح کلاس `RatingControl` تعریف کردید، به واسطه ی کلیدواژه `self` که به کلاس جاری اشاره دارد، همین کلاس را `target` و دریافت کننده ی پیغام رخداد فشرده شدن دکمه قرار می دهید.

عبارت `#selector` در خروجی مقدار `Selector` را برای متد ارائه شده برمی گرداند. `Selector` یک مقدار `opaque` است که با استفاده از آن متد خاصی را انتخاب می کنید و در واقع آن متد را به عنوان خروجی برمی گردانید (مقدار `opaque` مقداری است که جز نوع، هیچ اطلاعات دیگری درباره ی خود بروز نمی دهد).

API های قدیمی معمولاً از `selector` برای فراخوانی داینامیک متدها در زمان اجرا استفاده می کردند. اگرچه اغلب API های جدید `block` ها را جایگزین `selector` کرده اند، با این حال برخی از متدهای قدیمی همچون `(_:performSelector)` و `(_:addTargetForControlEvents:)` هنوز `selector` را (برای اشاره به متدی خاص) به عنوان آرگومان می گیرند.

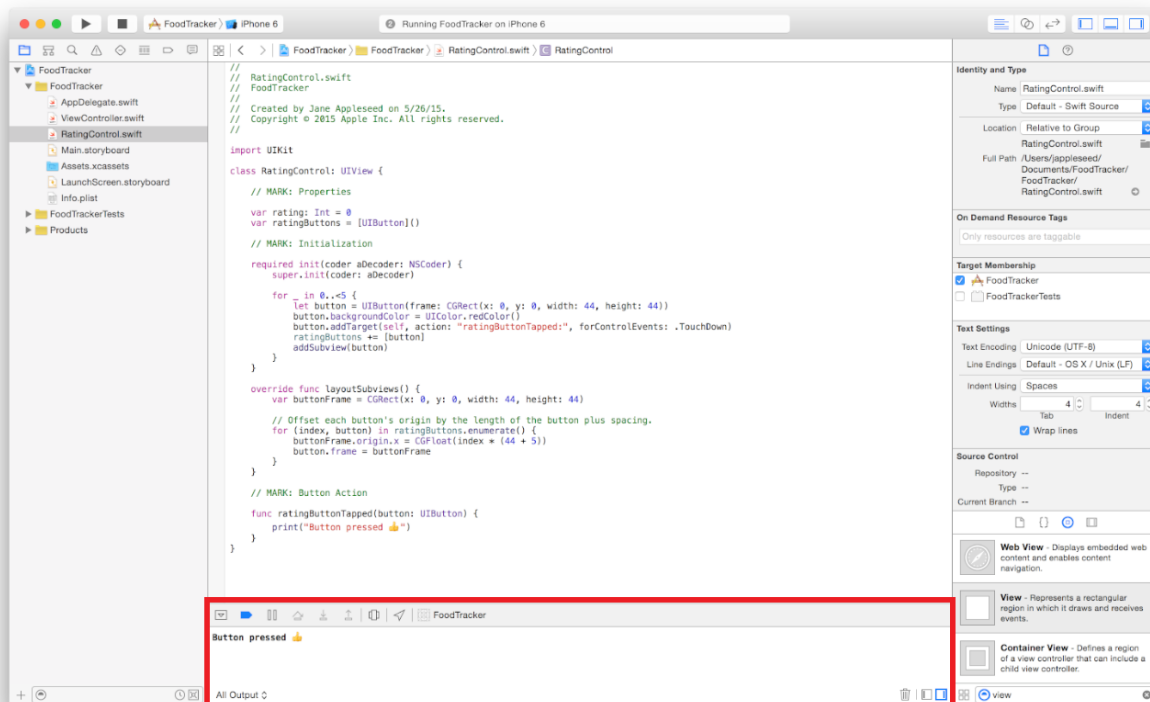
در مثال جاری، عبارت `(_:ratingButtonTapped)(RatingControl.selector)` مقدار `selector` را به عنوان خروجی برای متد `(_:ratingButtonTapped)(RatingControl.selector)` برمی گرداند. با این کار به سیستم اجازه می دهید تا به محض فشرده شدن دکمه، متدی (`action method`) که شما تعریف نموده و به این دکمه متصل کردید را اجرا کند.

از آنجایی که برای ساخت کنترل دکمه از `interface builder` استفاده نکردید، برای تعریف `action method` نیز نیازی به `IBAction` ندارید. می توانید `action` را مانند یک متد معمولی تعریف کنید.

پس از افزودن دستورات فوق، بدنه ی متد `init?(coder:)` می بایست به صورت زیر باشد:

```
required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
    button.backgroundColor = UIColor.redColor()
    button.addTarget(self, action:
    #selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
    .TouchDown)
    addSubview(button)
}
```

**تست کنید:** برنامه ی خود را اجرا نمایید. به محض کلیک بر روی دکمه ی قرمز رنگ، بایستی پیغام "Button pressed" را در Console مشاهده نمایید.



Console

اکنون زمان آن فرا رسیده تا اطلاعات لازم جهت امتیازدهی به غذا و نمایش آن امتیاز برای کاربر را در اختیار کلاس RatingControl قرار دهید. در وهله ی اول بایستی مقدار امتیاز ۰، ۱، ۲، ۳، ۴ یا ۵- را ذخیره کرده و رصد کنید. سپس دکمه هایی که کاربر با فشردن آن ها امتیاز غذا را مشخص می کند، ایجاد نمایید. می توانید مقدار امتیاز را با نوع int و مجموعه دکمه ها را به وسیله ی آرایه ای از آبجکت های UIButton تعریف نمایید.

برای این منظور مراحل زیر را دنبال نمایید:

۱. داخل فایل RatingControl.swift، خط تعریف کلاس را پیدا کنید:

```
class RatingControl: UIView {
```

۲. در زیر این خط، کد زیر را وارد نمایید:

```
// MARK: Properties
```

```
var rating = 0
```

```
var ratingButtons = [UIButton]()
```

در حال حاضر تنها یک دکمه در view خود دارید، در حالی که برای نمایش میزان محبوبیت غذا و امتیاز آن در کل به ۵ دکمه احتیاج دارید. جهت ایجاد این تعداد کنترل دکمه از یک حلقه ی for-in بهره می گیرد. این حلقه در یک دنباله، برای مثال مجموعه ای از اعداد صحیح، پیمایش کرده و یک مجموعه دستور را به دفعات مشخصی تکرار می کند. بجای یک دکمه، این ساختمان کد تعداد کل ۵ دکمه را ایجاد می کند. جهت ایجاد این تعداد کنترل دکمه، مراحل زیر را دنبال نمایید:

۱. در فایل RatingControl.swift، متد init?(coder:) را پیدا کنید:

```
required init?(coder aDecoder: NSCoder) {
```

```
super.init(coder: aDecoder)
```

```
let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
```

```
button.backgroundColor = UIColor.redColor()
```



```
button.addTarget(self, action:
#selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
.TouchDown)
addSubview(button)
}
```

۲. حال آن را به صورت زیر ویرایش نمایید:

```
for _ in 0..<5 {
let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
button.backgroundColor = UIColor.redColor()
button.addTarget(self, action:
#selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
.TouchDown)
addSubview(button)
}
```

برای سازمان دهی کد و اطمینان از توگذاری صحیح تمامی خط های آن، می توانید کل ساختمان کد را انتخاب کرده و سپس از Control-I را فشار دهید.

لازم به ذکر است که عملگر (<..) خود عدد بزرگ را شامل نمی شود، بنابراین این بازه از شروع شده تا ۴ را پیمایش می کند که در کل به ۵ بار اجرای دستورات منتهی می شود. در نتیجه ی بجای یک دکمه، با یک حلقه ۵ دکمه یکجا ایجاد می کنید. در صورتی که نیازی به آگاهی از گام جاری اجرای (گام تکرار) حلقه نیست، می توانید از ( \_ ) استفاده کنید.

۳. در بالای خط addSubview(button)، این دستور را اضافه کنید:

```
ratingButtons += [button]
```

هر کنترل دکمه ای که ایجاد می کنید، جهت ذخیره و رصد مقدارش، آن را به آرایه ی RatingsButtons اضافه می نمایید (تک تک دکمه هایی که با هر بار اجرای حلقه ایجاد می شود را داخل خانه های آرایه مزبور می ریزید).

هم اکنون بدنه ی متد init?(coder:) می بایست مشابه نمونه ی زیر باشد:

```
required init?(coder aDecoder: NSCoder) {
super.init(coder: aDecoder)
for _ in 0..<5 {
```

```

let button = UIButton(frame: CGRect(x: 0, y: 0, width: 44, height: 44))
button.backgroundColor = UIColor.redColor()
button.addTarget(self, action:
#selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
.TouchDown)
ratingButtons += [button]
addSubview(button)
}
}

```

**تست کنید:** حال برنامه ی خود را اجرا نمایید. شاید این طور به نظر برسد که در UI برنامه تنها یک دکمه قابل مشاهده است. اما جالب است بدانید که حلقه ی for-in دکمه های ایجاد شده را مانند پشته بر روی هم قرار داده است. به منظور نمایش صحیح کنترل امتیاز دهی، شما باید چیدمان کنترل های دکمه در view را خود تنظیم نمایید.

Carrier 9:41 AM

Meal Name

Enter meal name

[Set Default Label Text](#)

No photo selected

عملیات لازم جهت تنظیم چیدمان و نمایش صحیح المان ها در UI، داخل بدنه ی متدی به نام `layoutSubviews` از کلاس `UIView` تعریف می شود. سیستم این متد را خود در زمان مناسب فراخوانی کرده و از این طریق به کلاس های ارث بری شده از `UIView` اجازه

می دهد تا subview های مورد نظر را به نحو صحیح و با چیدمان دلخواه برنامه نویس در UI نمایش دهند.

برای تنظیم چیدمان کنترل های دکمه، لازم است پیاده سازی متد نام برده را بازنویسی (override) نمایید.

جهت بازنویسی این متد و نمایش دکمه ها در کنار هم، مراحل زیر را به ترتیب طی نمایید:

۱. داخل فایل RatingControl.swift، در زیر متد `init?(coder:)`، تابع زیر را وارد نمایید:

```
override func layoutSubviews() {
}
```

یادآور می شویم که با استفاده از قابلیت `code completion` محیط کاری Xcode می توانید (اسکلت) این متد را به صورت آماده در کد خود قرار دهید.

۲. داخل بدنه ی این متد، کد زیر را تایپ کنید:

```
var buttonFrame = CGRect(x: 0, y: 0, width: 44, height: 44)
// Offset each button's origin by the length of the button plus spacing.
for (index, button) in ratingButtons.enumerate() {
    buttonFrame.origin.x = CGFloat(index * (44 + 5))
    button.frame = buttonFrame
}
```

این کد frame دکمه ها را ایجاد می کند، سپس با استفاده از حلقه ی `for-in` داخل آرایه ی دکمه ها پیمایش کرده و در هر بار اجرا مقادیر (مربوط به) frame هر یک از کنترل های مزبور را تنظیم می نماید.

متد `enumerate()` در خروجی یک `collection` برمی گرداند که داخل این `collection` تک تک المان های آرایه ی `ratingButtons` همراه با اندیس متناظر آن ها قرار داده شده است. این `collection` در واقع مجموعه ای از `tuple` ها (مقادیر گروهی مثال کلید-مقدار) است و در این مثال، هر `tuple` شامل یک دکمه + اندیس مربوطه ی آن می باشد.

حلقه ی for-in که داخل بدنه ی تابع layoutSubviews مشاهده می کنید، مقدار دکمه و اندیس آن را به ترتیب داخل متغیرهای محلی button و index ذخیره می کند (مقدار آن ها را به متغیرهای محلی ذکر شده bind می کند). با استفاده از متغیر index، محل جدید فریم هر دکمه را محاسبه کرده و بعد آن را داخل متغیر button ذخیره می کند. مکان قرارگیری فریم ها برابر اندازه ی استاندارد دکمه با عرض و طول = ۴۴ + padding 5 = (فاصله ی بین دکمه ها)، ضرب در اندیس (شماره ی مکان قرار گیری) آن دکمه محاسبه و مقداردهی می شود.

در حال حاضر کد موجود در بدنه ی متد layoutSubviews() می بایست مشابه زیر باشد:

```
override func layoutSubviews() {
    var buttonFrame = CGRect(x: 0, y: 0, width: 44, height: 44)
    // Offset each button's origin by the length of the button plus spacing.
    for (index, button) in ratingButtons.enumerate() {
        buttonFrame.origin.x = CGFloat(index * (44 + 5))
        button.frame = buttonFrame
    }
}
```

**تست کنید:** برنامه ی خود را اجرا نمایید. خواهید دید که این بار دکمه ها در کنار هم قرار گرفته اند. یادآور می شویم که کلیک بر روی هر یک از دکمه ها، با توجه به کدی که تا به اینجا برای برنامه ی خود نوشته اید، صرفاً سبب فراخوانی متد ratingButtonTapped(\_\_\_\_\_) و چاپ شدن پیغامی در console می شود.

Carrier 9:41 AM

Meal Name

Enter meal name

Set Default Label Text

No photo selected

می توانید console را برای داشتن فضای کاری بیشتر، پنهان نمایید. برای این منظور کافی است بر روی دکمه ی اشاره شده در تصویر زیر کلیک نمایید:



### تنظیم میزان فاصله ی دکمه ها از هم و تعداد آن ها (در قالب دو متغیر/property مجزا)

اگر به خاطر داشته باشید، برای تنظیم تعداد کنترل های قابل مشاهده دکمه در UI و میزان فاصله ی بین آن ها، تاکنون در کد خود از عدد ۵ استفاده کردید. به طور کلی نگه داشتن مقادیر hardcoded شده به صورت پراکنده در متن برنامه روش مطلوب و بهینه ی کدنویسی تلقی نمی شود، چرا که در آن صورت، اگر بخواهید میزان فاصله ی بین دکمه ها را ویرایش کنید، می بایست هرجایی در کد که این مقدار به نشانی میزان فاصله ی بین المان ها بکار رفته را پیدا و متعاقبا ویرایش نمایید. حال این امر که مقدار مزبور نشانگر دو چیز کاملا متفاوت در کد شما است (یکی فاصله ی بین دکمه ها و دیگری تعداد آن ها) فقط به پیچیدگی هر چه بیشتر شرایط می افزاید.

برای رفع این مشکل، دو متغیر تعریف می کنید که یکی محل نگهداری مقدار تعداد دکمه ها و دیگری ظرف نگهداری فاصله ی بین آن ها است. اکنون زمانی که می خواهید مقدار را تغییر دهید، فقط می بایست آن را در یک مکان واحد ویرایش کنید.

جهت تعریف دو متغیر (property) مجزا برای نگهداری فاصله ی بین دکمه ها و تعداد آن ها، مراحل زیر را دنبال نمایید:

۱. داخل فایل RatingControl.swift، بخش MARK: Properties را پیدا کنید:

```
// MARK: Properties
var rating = 0
var ratingButtons = [UIButton]()
```

می توانید با استفاده از functions menu، به سرعت به بخش دلخواه در کد پیمایش کنید (بپرسید). این منو با کلیک بر روی اسم فایل، در بالای editor area نمایان می شود.

۲. در زیر property های موجود، کد زیر را وارد نمایید:

```
let spacing = 5
```

این ثابت (constant) را جهت تعیین میزان فاصله ی بین دکمه ها در ال، تعریف می کنید.

۳. داخل بدنه ی متد layoutSubviews، مقدار نوشتاری (literal) که برای تعیین میزان

فاصله ی بین دکمه ها استفاده می کردید را با ثابت spacing جایگزین نمایید:

```
buttonFrame.origin.x = CGFloat(index * (44 + spacing))
```

۴. در زیر ثابت spacing، ثابت دیگری به صورت زیر اضافه نمایید:

```
let starCount = 5
```

می توانید این ثابت را جهت تعیین تعداد دکمه هایی (ستاره ها) که کنترل برای کاربر به نمایش می گذارد، بکار ببرید.

۵. اکنون، داخل بدنه ی (init?(coder:))، مقدار نوشتاری (literal) که به نشانه ی تعداد دکمه

ها در متن برنامه استفاده کردید را با ثابت starCount جایگزین نمایید:

```
for _ in 0..

```

**تست کنید:** اپلیکیشن خود را اجرا کنید. رفتار و ظاهر برنامه نباید نسبت به قبل تغییری کرده باشد.

### تعریف یک ثابت جهت تعیین اندازه ی کنترل دکمه

اگر بخاطر داشته باشید، مقدار ۴۴ را به نشانه ی اندازه ی کنترل دکمه، در کد خود بکار بردید. اما همان طور که قبلاً گفته شد، مقادیر hardcoded شده به صورت پراکنده در کد مطلوب نیست و بهتر است این روش کدنویسی را از متن برنامه ی خود حذف کنید. این بار دکمه ها را طوری تنظیم می کنید که خود را با اندازه ی view میزبان تطبیق دهند



(container view ای که در ابتدای مبحث به storyboard خود اضافه کردید). این کار را با بازیابی مقدار طول (height) view میزبان و ذخیره ی آن داخل یک ثابت محلی انجام می دهید، سپس به راحتی و تنها یکبار از داخل هر متد به آن دست پیدا می کنید. جهت تعریف یک ثابت، برای ذخیره و نگهداری اندازه ی دکمه های کنترل امتیازدهی، مراحل زیر طی نمایید:

۱. داخل بدنه ی متد `layoutSubviews()`، این کد را قبل از اولین خط پیاده سازی اضافه نمایید:

```
// Set the button's width and height to a square the size of the frame's height.
let buttonSize = Int(frame.size.height)
```

با افزودن این کد layout را به طور قابل توجهی انعطاف پذیرتر می سازید.

۲. حال ثابت `buttonSize` را جایگزین مقدار ۴۴ در بقیه ی کد نمایید:

```
var buttonFrame = CGRect(x: 0, y: 0, width: buttonSize, height: buttonSize)
// Offset each button's origin by the length of the button plus spacing.
for (index, button) in ratingButtons.enumerate() {
    buttonFrame.origin.x = CGFloat(index * (buttonSize + 5))
    button.frame = buttonFrame
}
```

برای اینکه `stack view` کنترل امتیاز دهی را به صورت دلخواه نمایش دهد، لازم است `intrinsic content size` (اندازه ی درونی) کنترل مورد نظر را بروز رسانی نمایید. این بار محاسبه ی اندازه ی هریک از دکمه ها (ستاره ها) و فاصله ی بین آن ها را به متد `intrinsicContentSize()` واگذار می کنید.

برای این منظور کافی است کد زیر را بکار ببرید:

```
let buttonSize = Int(frame.size.height)
let width = (buttonSize * starCount) + (spacing * (starCount - 1))
return CGSize(width: width, height: buttonSize)
```

در متد `init?(coder:)`، اولین خط کد داخل ساختمان حلقه ی `for-in` را به صورت زیر ویرایش کنید:

```
let button = UIButton()
```

از آنجایی که پراپرتی `frame` دکمه ها را داخل متد `layoutSubviews()` تنظیم می کنید، دیگر در زمان ایجاد دکمه ها نیازی به مقدار دهی آن ها نیست.

کد موجود در بدنه ی متد `layoutSubviews()` اکنون می بایست مشابه نمونه ی زیر باشد:

```
override func layoutSubviews() {
    // Set the button's width and height to a square the size of the frame's height.
    let buttonSize = Int(frame.size.height)
    var buttonFrame = CGRect(x: 0, y: 0, width: buttonSize, height: buttonSize)
    // Offset each button's origin by the length of the button plus some spacing.
    for (index, button) in ratingButtons.enumerate() {
        buttonFrame.origin.x = CGFloat(index * (buttonSize + 5))
        button.frame = buttonFrame
    }
}
```

کد موجود در بدنه ی متد `intrinsicContentSize` می بایست مشابه نمونه ی زیر باشد:

```
override func intrinsicContentSize() -> CGSize {
    let buttonSize = Int(frame.size.height)
    let width = (buttonSize * starCount) + (spacing * (starCount - 1))
    return CGSize(width: width, height: buttonSize)
}
```

بدنه ی متد `init?(coder:)` نیز بایستی مانند زیر باشد:

```
required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    for _ in 0..  
5 {
        let button = UIButton()
        button.backgroundColor = UIColor.redColor()
        button.addTarget(self, action:
            #selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
            .TouchDown)
```

```
ratingButtons += [button]
addSubview(button)
}
}
```

**تست کنید:** برنامه را اجرا کنید. رفتار و ظاهر برنامه نسبت به قبل تغییر خاصی نکرده است. دکمه ها باید در کنار هم قرار گرفته باشند و کلیک بر روی هر کدام از دکمه ها هنوز بایستی سبب فراخوانی متد (ratingButtonTapped(\_:)) شده و پیغامی را در console ثبت کند.



Carrier 9:41 AM

Meal Name

Enter meal name

Set Default Label Text

No photo selected

جایگزین کردن عکس ستاره بجای مربع های قرمز رنگ

در این بخش عکس یک ستاره ی پر شده و خالی را به هریک از دکمه های کنترل اضافه

می کنید:

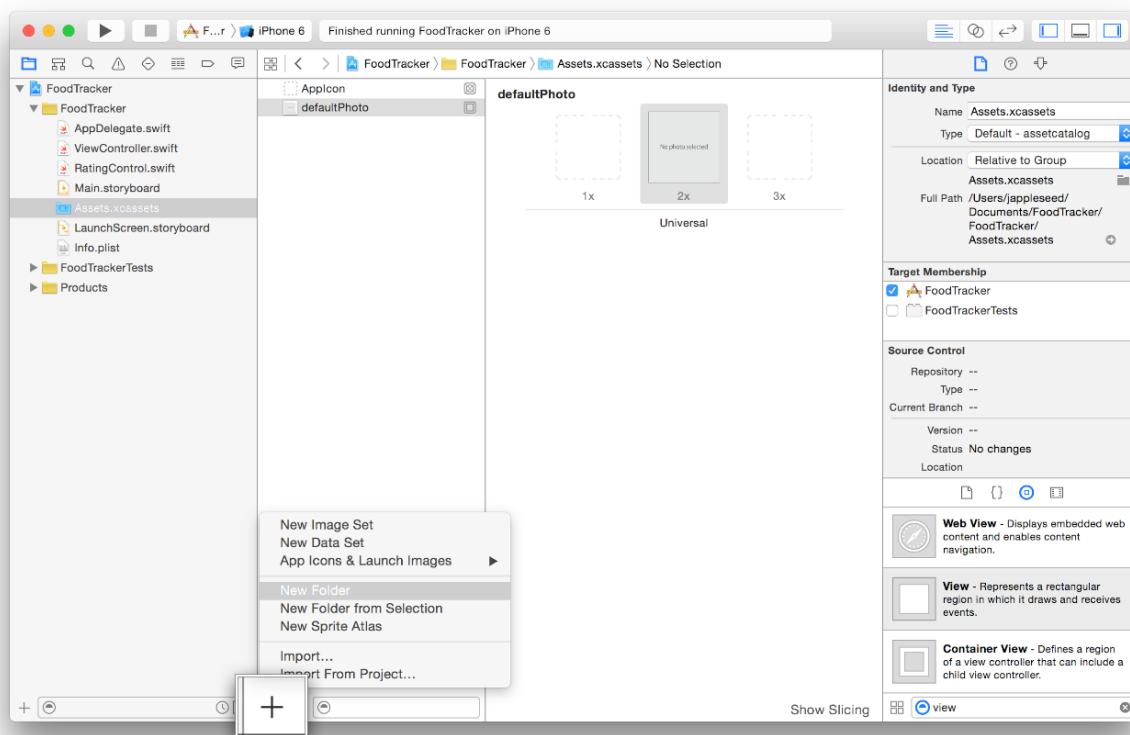


می توانید تصاویر فوق را در پوشه ی Images/ از فایل پروژه ی دانلود شده انتخاب کنید یا در صورت تمایل تصاویر دلخواه خود را بکار ببرید. (کافی است مطمئن شوید اسم عکس های مورد استفاده ی شما با اسم عکس ها در کد کاملاً همخوانی داشته باشد).  
به منظور افزودن تصویر به پروژه ی خود، مراحل زیر را طی نمایید:

۱. داخل project navigator، با کلیک بر روی Assets.xcassets، ابزار asset catalog را باز کنید.

یادآور می شویم که asset catalog مکانی است که در آن محتوای برنامه ی خود نظیر عکس، متن و غیره را مدیریت می کنید.

۲. در گوشه ی سمت چپ، بر روی دکمه ی + کلیک کرده و پوشه ی New Folder را از منوی pop-up انتخاب نمایید.



۳. بر روی اسم پوشه دابل کلیک کرده و آن را به Rating Images تغییر دهید.

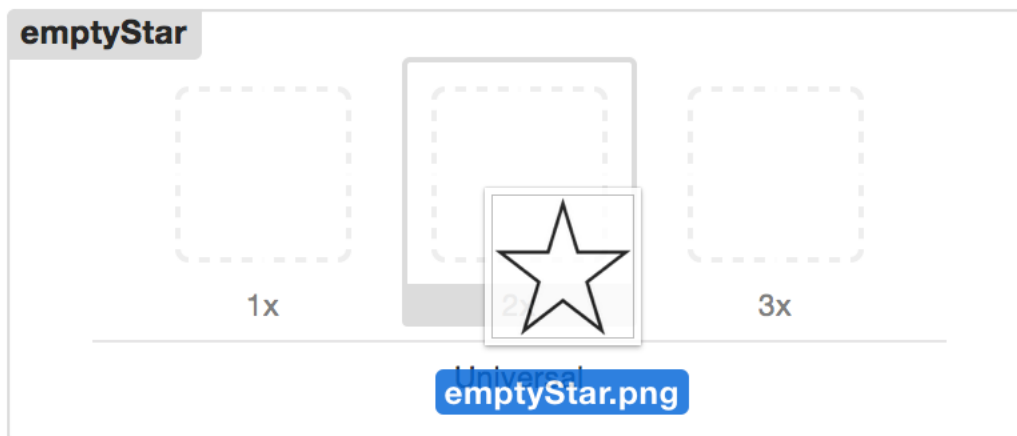
۴. پس از انتخاب پوشه ی مذکور، در گوشه ی سمت چپ، بر روی دکمه ی (+) کلیک کرده و سپس گزینه ی New Image Set را از منوی pop-up انتخاب نمایید.

هر Image set به منزله ی یک image asset واحد است اما می تواند نسخه های مختلفی از یک عکس را در وضوح تصویر متفاوت داشته باشد.

۵. بر روی اسم image set دابل کلیک کرده و سپس آن را به emptyStar تغییر دهید.

۶. حال عکس ستاره ی خالی را از حافظه ی کامپیوتر انتخاب نمایید.

۷. عکس را کشیده و آن را در فضای خالی 2x داخل image set جاری جایگذاری نمایید.



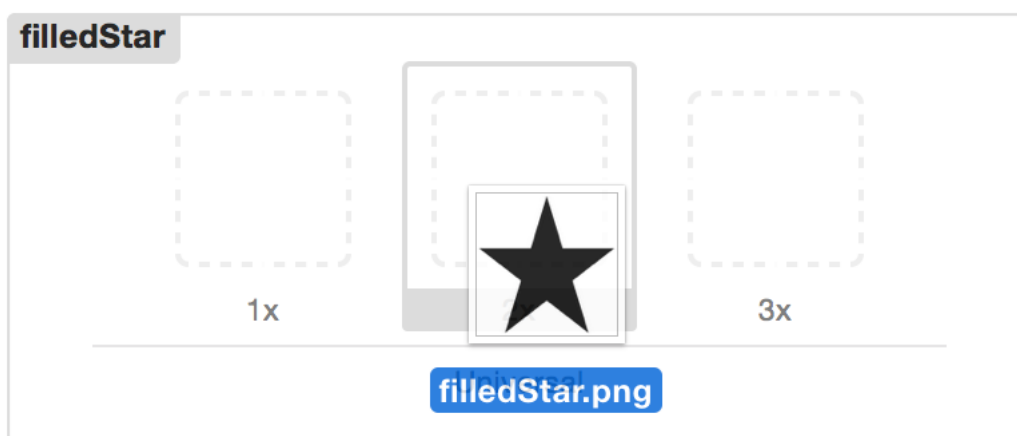
2x وضوح تصویر نمایشگر شبیه ساز iPhone 6 می باشد که برنامه ی خود را در محیط آن تست می کنید. عکس مورد نظر قاعدتا باید در این وضوح با بهترین کیفیت نمایش داده شود.

۸. در پایین محیط، گوشه ی سمت چپ، بر روی دکمه ی (+) کلیک کنید و سپس گزینه ی New Image Set را از منوی pop-up انتخاب نمایید.

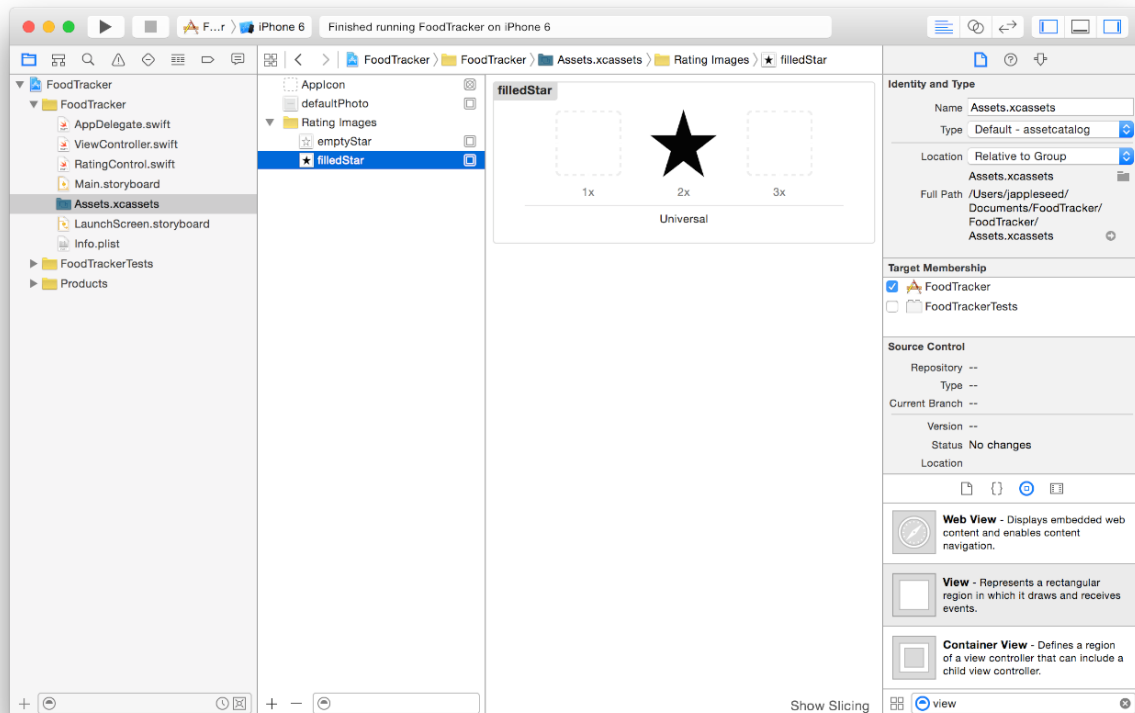
۹. بر روی اسم image set دو بار کلیک کرده و آن را به filledStar تغییر دهید.

۱۰. اکنون عکس ستاره ی پر شده را از حافظه ی کامپیوتر خود انتخاب نمایید.

۱۱. عکس مورد نظر را کشیده و آن را در فضای خالی 2x، داخل image set جاری جایگذاری نمایید.



Asset catalog شما هم اکنون می بایست مشابه زیر باشد:



در گام بعدی کدی می نویسید که عکس مربوطه را در زمان مناسب برای دکمه تنظیم کرده و نمایش می دهد.

جهت تنظیم عکس برای نمایش، مراحل زیر را به ترتیب دنبال نمایید:

۱. فایل `RatingControl.swift` را باز کنید.

۲. داخل بدنه ی متد `init?(coder:)`، دو خط زیر را به قبل از ساختمان حلقه ی `for-` `in` اضافه نمایید:

```
let filledStarImage = UIImage(named: "filledStar")
let emptyStarImage = UIImage(named: "emptyStar")
```

۳. داخل قطعه کد `for-in`، پس از آن خطی از کد که دکمه در آن مقداردهی اولیه `(initialize)` می شود، دستورات زیر را درج کنید:

```
button.setImage(emptyStarImage, forState: .Normal)
button.setImage(filledStarImage, forState: .Selected)
```



```
button.setImage(filledStarImage, forState: [.Highlighted, .Selected])
```

شما برای هر یک از دو حالت دکمه یک عکس مجزا تخصیص می دهید تا بتوانید به راحتی تشخیص دهید چه زمانی دکمه انتخاب شده و چه زمانی کاربر هنوز بر روی دکمه کلیک نکرده است. عکس ستاره ی خالی زمانی نمایش داده می شود که دکمه انتخاب نشده باشد (وضعیت Normal). عکس ستاره ی پر شده زمانی نمایش داده می شود که دکمه انتخاب شده (Selected) و همچنین زمانی که دکمه همزمان انتخاب و هایلایت شده باشد (Selected و Highlighted).  
وضعیتی که کاربر در حال ضربه زدن بر روی دکمه باشد.

۴. آن خطی از برنامه که رنگ پس زمینه ی دکمه را بر روی قرمز تنظیم می کند، حذف نمایید:

```
button.backgroundColor = UIColor.redColor()
```

از آنجایی که دکمه های شما اکنون عکس های خود را دارند، زمان آن رسیده تا رنگ پس زمینه ی دکمه را حذف نمایید.

۵. کد زیر را اضافه نمایید:

```
button.adjustsImageWhenHighlighted = false
```

دستور فوق را به این خاطر اضافه می کنید که عکس به هنگام تغییر بین دو حالت بیش از یکبار هایلایت نشود.

در حال حاضر کد init?(coder:) شما بایستی به صورت زیر باشد:

```
required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    let emptyStarImage = UIImage(named: "emptyStar")
    let filledStarImage = UIImage(named: "filledStar")
    for _ in 0..<5 {
        let button = UIButton()
        button.setImage(emptyStarImage, forState: .Normal)
        button.setImage(filledStarImage, forState: .Selected)
    }
}
```

```

button.setImage(filledStarImage, forState: [.Highlighted, .Selected])
button.adjustsImageWhenHighlighted = false
button.addTarget(self, action:
#selector(RatingControl.ratingButtonTapped(_:)), forControlEvents:
.TouchDown)
ratingButtons += [button]
addSubview(button)
}
}

```

**تست کنید:** برنامه ی خود را اجرا نمایید. این بار باید بجای مربع های قرمز رنگ، عکس ستاره در ال نمایش داده شوند. با کلیک بر روی هر یک از دکمه ها همچنان فقط متد ratingButtonTapped(\_\_\_\_\_) صدا خورده می شود و پیغامی در console چاپ می گردد. در این برهه از زمان، با توجه به کدی که برای برنامه نوشته اید، FoodTracker هنوز قادر به تغییر عکس و هایلایت کردن آن به هنگام کلیک کاربر نیست. این رفتار را در گام بعدی برای اپلیکیشن خود تعریف خواهید کرد.

جایگزین کردن پیاده سازی debugging متد ratingButtonTapped(\_\_\_\_\_) با پیاده سازی واقعی (پیاده سازی action متصل به UIButton)

برنامه ی شما باید به کاربر اجازه دهد تا با کلیک بر روی ستاره، به غذای مورد نظر امتیاز بدهد. برای این منظور باید پیاده سازی متد ratingButtonTapped(\_\_\_\_\_) را که در حال حاضر صرفاً پیغامی را در console ثبت می کند با عملیات واقعی در بدنه ی متد جایگزین کنید.

جهت پیاده سازی عملیات امتیازدهی در بدنه ی متد ذکر شده، مراحل زیر را دنبال نمایید:

۱. داخل فایل RatingControl.swift، متد ratingButtonTapped(\_\_\_\_\_) را پیدا کنید:

```

func ratingButtonTapped(button: UIButton) {
print("Button pressed 🍑")
}

```

۲. کد زیر را در بدنه ی این متد جایگزین دستور print نمایید:

```
rating = ratingButtons.indexOf(button)! + 1
```

متد `indexOf(:)` دکمه ی مورد نظر را بر اساس اندیس آن در آرایه ای از دکمه ها پیدا کرده و سپس اندیس (شماره ی مکان قرارگیری) آن المان در آرایه را به عنوان خروجی برمی گرداند. از آنجایی که احتمال دارد نمونه ی مورد جستجو در مجموعه ی مورد نظر وجود نداشته باشد، این متد طوری نوشته شده که در خروجی یک عدد صحیح (Int) از نوع optional بازگردانی نماید.

شما می دانید تنها دکمه هایی که `action` را فعال و سبب اجرای آن می شود، همان دکمه هایی هستند که خودتان تعریف کرده و به آرایه اضافه نمودید. از اینرو مطمئن هستید که متد `indexOf` به عنوان خروجی شماره ی مکان قرارگیری المان مد نظر را برمی گرداند. می توانید با استفاده از عملگر `!`، اندیس المان را استخراج نموده و به آن دسترسی داشته باشید. حال جهت بدست آوردن امتیاز مربوطه، عدد `1` را به اندیس المان اضافه می کنید.

توجه داشته باشید که اضافه کردن مقدار `1` ضروری است زیرا اندیس آرایه از `0` آغاز می شود.

۳. داخل فایل `RatingControl.swift`، به قبل از آخرین `()`، کد زیر را اضافه نمایید:

```
func updateButtonSelectionStates() {  
}
```

این تابع در واقع یک متد کمکی است که با استفاده از آن وضعیت انتخاب دکمه ها را بروز رسانی می نمایید.

۴. داخل بدنه ی متد `updateButtonSelectionStates()`، این حلقه ی `for-in` را اضافه نمایید:

```
for (index, button) in ratingButtons.enumerate() {
```

// If the index of a button is less than the rating, that button should be selected.

button.selected = index < rating

}

این کد داخل آرایه ای از دکمه ها پیمایش کرده، سپس در هر بار تکرار، وضعیت دکمه ها را بر اساس اینکه آیا مقدار اندیس (متغیر index) آن از امتیاز (مقدار متغیر rating) کوچکتر است یا خیر، تنظیم می کند. در صورت برقرار بودن شرط، بدین معنی که اگر  $index < rating$  نتیجه ی true را برگرداند، حلقه ی for-in وضعیت دکمه را بر روی selected تنظیم نموده و تصویر ستاره ی پر شده را به نمایش می گذارد. در غیر این صورت، دکمه انتخاب نشده و تصویر ستاره ی خالی به نمایش در می آید.

۵. داخل بدنه ی ratingButtonTapped(\_:)، در آخرین خط پیاده سازی،

متد updateButtonSelectionStates() را فراخوانی نمایید:

```
func ratingButtonTapped(button: UIButton) {
    rating = ratingButtons.indexOf(button)! + 1
    updateButtonSelectionStates()
}
```

۶. در بدنه ی متد layoutSubviews()، متد

updateButtonSelectionStates() را در آخرین خط پیاده سازی، فراخوانی

نمایید:

```
override func layoutSubviews() {
    // Set the button's width and height to a square the size of the frame's height.
    let buttonSize = Int(frame.size.height)
    var buttonFrame = CGRect(x: 0, y: 0, width: buttonSize, height: buttonSize)
    // Offset each button's origin by the length of the button plus some spacing.
    for (index, button) in ratingButtons.enumerate() {
        buttonFrame.origin.x = CGFloat(index * (buttonSize + 5))
        button.frame = buttonFrame
    }
    updateButtonSelectionStates()
}
```

لازم است وضعیت انتخاب دکمه هم به هنگام تغییر امتیاز و هم به هنگام بارگذاری view بروز رسانی شود.

۷. در بخش MARK: Properties //، متغیر rating را پیدا کنید:

```
var rating = 0
```

۸. اکنون property observer را به متغیر rating اضافه نمایید. property observer تغییرات در مقدار property را رصد کرده و به آن ها واکنش نشان می دهد.

```
var rating = 0 {
  didSet {
    setNeedsLayout()
  }
}
```

property observer همان طور که در بالا نیز ذکر شد، یک تکه کد است که به property اضافه شده و تغییرات اعمال شده در آن را دنبال می کند. سپس هر بار که مقدار property تنظیم می شود، فراخوانی شده و می توان از آن برای اجرای عملیات بلافاصله قبل/بعد از تغییر مقدار استفاده نمود. به عبارت دقیق تر، didSet یک property observer است که بلافاصله پس از تنظیم مقدار متغیر صدا خورده می شود. در مثال جاری، متد setNeedsLayout() را داخل بدنه ی property observer فراخوانی می کنید که با هر بار تغییر در مقدار متغیر rating، ظاهر/layout را در UI تغییر می دهد. با این کار، اطمینان حاصل می شود که UI همیشه مقدار دقیق متغیر rating را منعکس می کند.

بدنه ی متد updateButtonSelectionStates() هم اکنون بایستی مشابه نمونه ی زیر باشد:

```
func updateButtonSelectionStates() {
  for (index, button) in ratingButtons.enumerate() {
```

// If the index of a button is less than the rating, that button shouldn't be selected.

button.selected = index < rating

}

}

**تست کنید:** برنامه ی خود را اجرا کنید. برنامه بایستی ۵ ستاره به نمایش گذاشته و شما

باید بتوانی با استفاده از آن ها به غذای مورد نظر امتیاز بدهید. به منظور تست، بر روی

ستاره ی سوم کلیک کرده تا امتیاز بر روی ۳ تنظیم شود.



Carrier 9:41 AM

Meal Name

Enter meal name

[Set Default Label Text](#)

No photo selected

★★★★☆

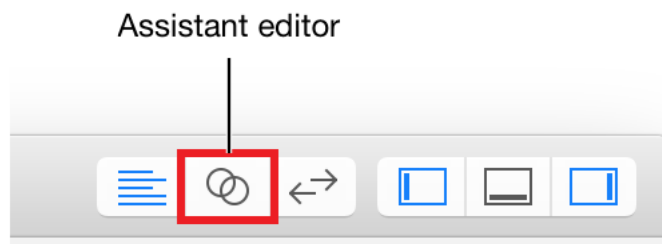
متصل کردن کنترل امتیاز دهی به کد `view controller`

آخرین کاری که باید برای تکمیل و راه اندازی کنترل امتیازدهی انجام دهید، این است که یک اشاره گر (در قالب `outlet`) به این کنترل در کلاس `view controller` تعریف نمایید.

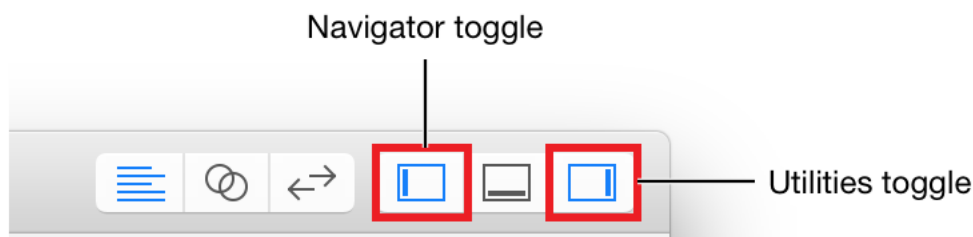
جهت متصل کردن outlet کنترل امتیازدهی به `ViewController.swift`، مراحل زیر را به ترتیب دنبال نمایید:

۱. Storyboard خود را باز کنید.

۲. جهت باز کردن ویرایشگر کمکی، بر روی دکمه ی Assistant در نوار ابزار محیط کاری Xcode کلیک نمایید:



۳. در صورت نیاز به فضای کاری بیشتر، می توانید `project navigator` و `utility area` را با کلیک بر روی دکمه های مربوطه در نوار ابزار محیط Xcode، ببندید.



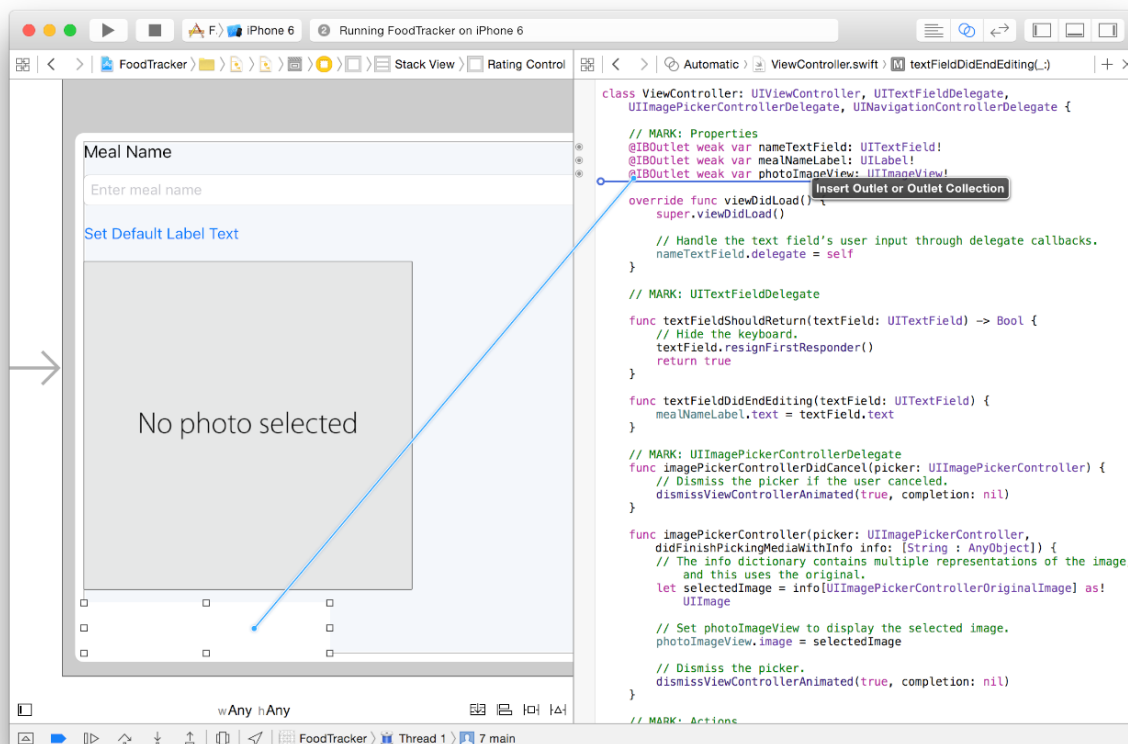
در صورت لزوم می توانید کادر `outline view` را نیز پنهان نمایید.

۴. کنترل امتیازدهی را انتخاب نمایید.

محتویات فایل `ViewController.swift` داخل ویرایشگر کمکی ( `assistant editor` ) در سمت راست محیط به نمایش در می آید (اگر محتوای فایل نام برده به نمایش گذاشته نشد، آنگاه می بایست در `editor selector bar` مراحل روبرو را طی نمایید: `Automatic > ViewController.swift`).

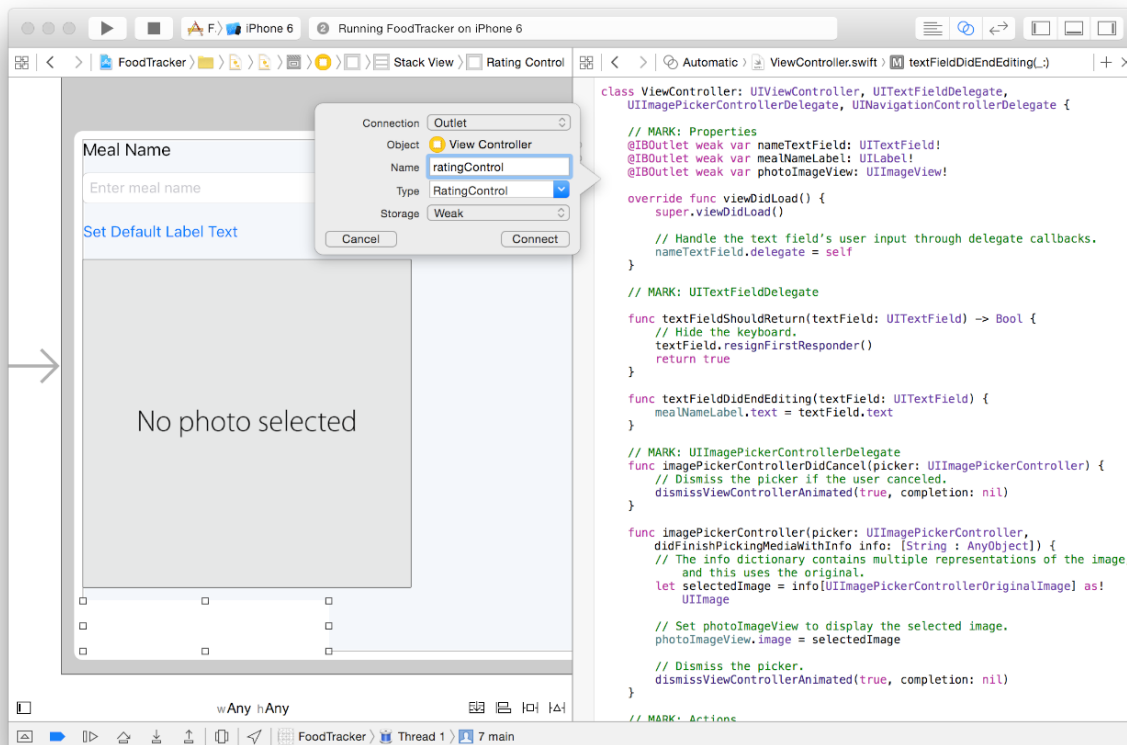
۵. بر روی کنترل امتیازدهی کلیک کرده، آن را با اشاره گر موس از `canvas` به داخل ناحیه ی ویرایشگر بکشید و سپس کنترل مورد نظر را در زیر متغیر `photoImageView` (داخل فایل `ViewController.swift`) جایگذاری نمایید.





۶. داخل کادر محاوره ای که نمایان می شود، در فیلد Name آن، مقدار ratingControl را وارد نمایید.

لازم نیست به دیگر تنظیمات در این کادر محاوره ای دست بزنید. کادر محاوره ای شما هم اکنون می بایست ظاهری مشابه نمونه ی زیر داشته باشد:



۷. دکمه ی Connect را کلیک نمایید.

اکنون کنترل امتیازدهی در storyboard یک اشاره گر به خود در کلاس ViewController دارد.

پاک سازی کد و اطلاعات غیرضروری از پروژه (Cleanup)

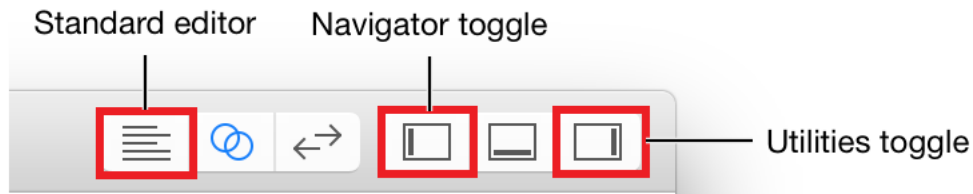
به مراحل نهایی و تکمیل scene جاری در رابط کاربری برنامه ی FoodTracker نزدیک می شوید. اما پیش از تکمیل این صفحه ی محتوا می بایست نسبت به پاک سازی پروژه اقدام نمایید. در حال حاضر برنامه ی FoodTracker رفتار به مراتب پیچیده تری را پیاده سازی کرده و ال متفاوت تری را نسبت به مباحث قبلی به نمایش می گذارد. از اینرو توصیه می شود کدهای غیر ضروری و بلااستفاده را از پروژه حذف نمایید.

در این بخش همچنین المان های رابط کاربری را در stack view وسط چین می کنید تا ال توازن بیشتری پیدا کند.

جهت پاک سازی لایه ی UI برنامه از گدهای غیرضروری، مراحل زیر را دنبال نمایید:

۱. با کلیک بر روی دکمه ی Standard، ویرایشگر اصلی محیط Xcode (standard editor) را

باز نمایید:



۲. حال project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار

محیط کاری Xcode باز نمایید.

۳. Storyboard را باز نمایید.

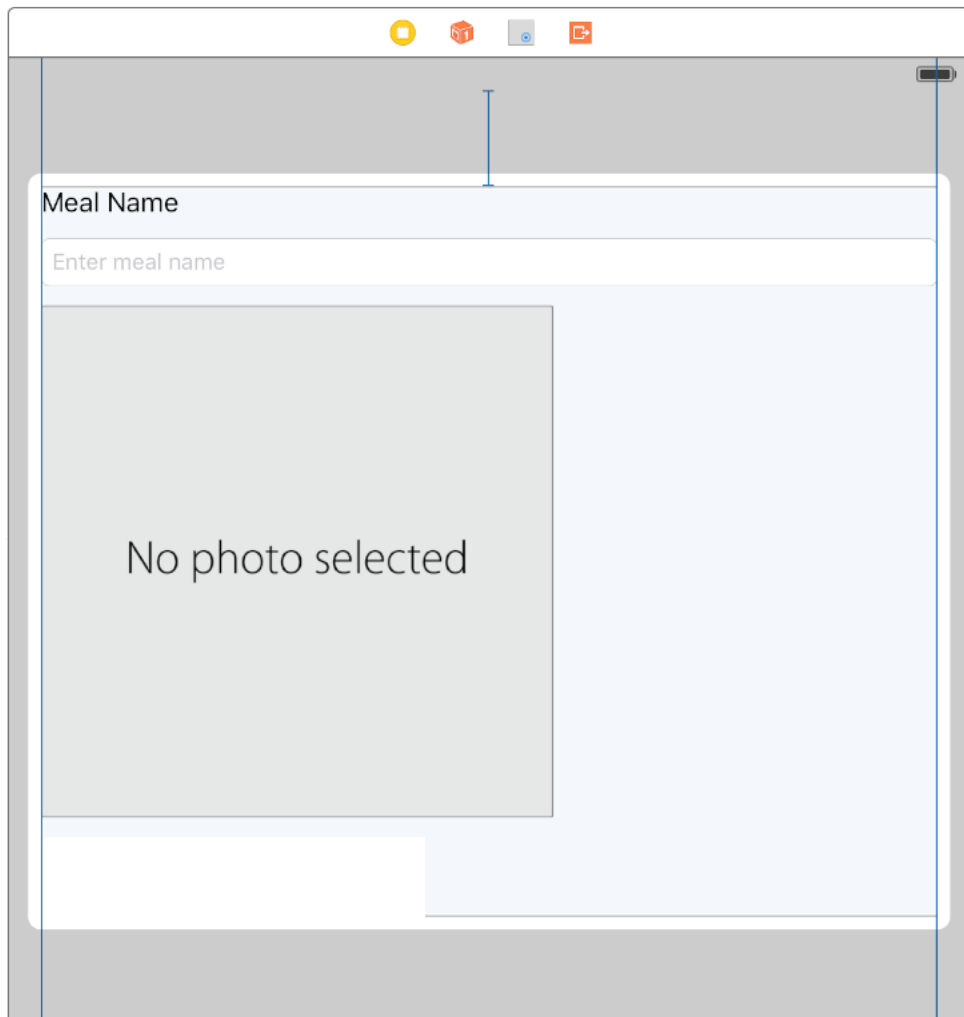
۴. دکمه ی Set Default Label Text را انتخاب کنید، سپس با فشردن دکمه ی Delete آن

را حذف نمایید.

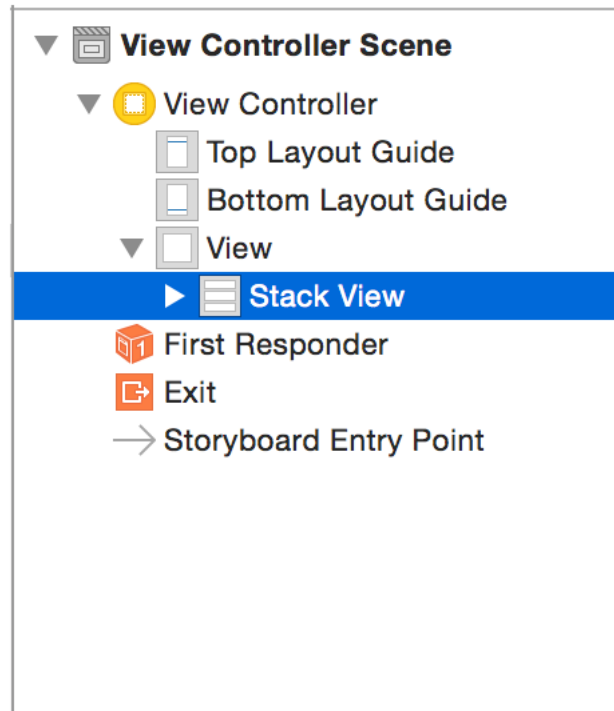
Stack view با تنظیم مجدد چیدمان المان های UI، فاصله ی خالی که در اثر حذف دکمه

ی مورد نظر ایجاد شده را پر می کند.

آموزشگاه تحلیکرو داده



۵. در صورت لزوم outline view را باز کرده و آبجکت Stack View را از آن انتخاب نمایید.

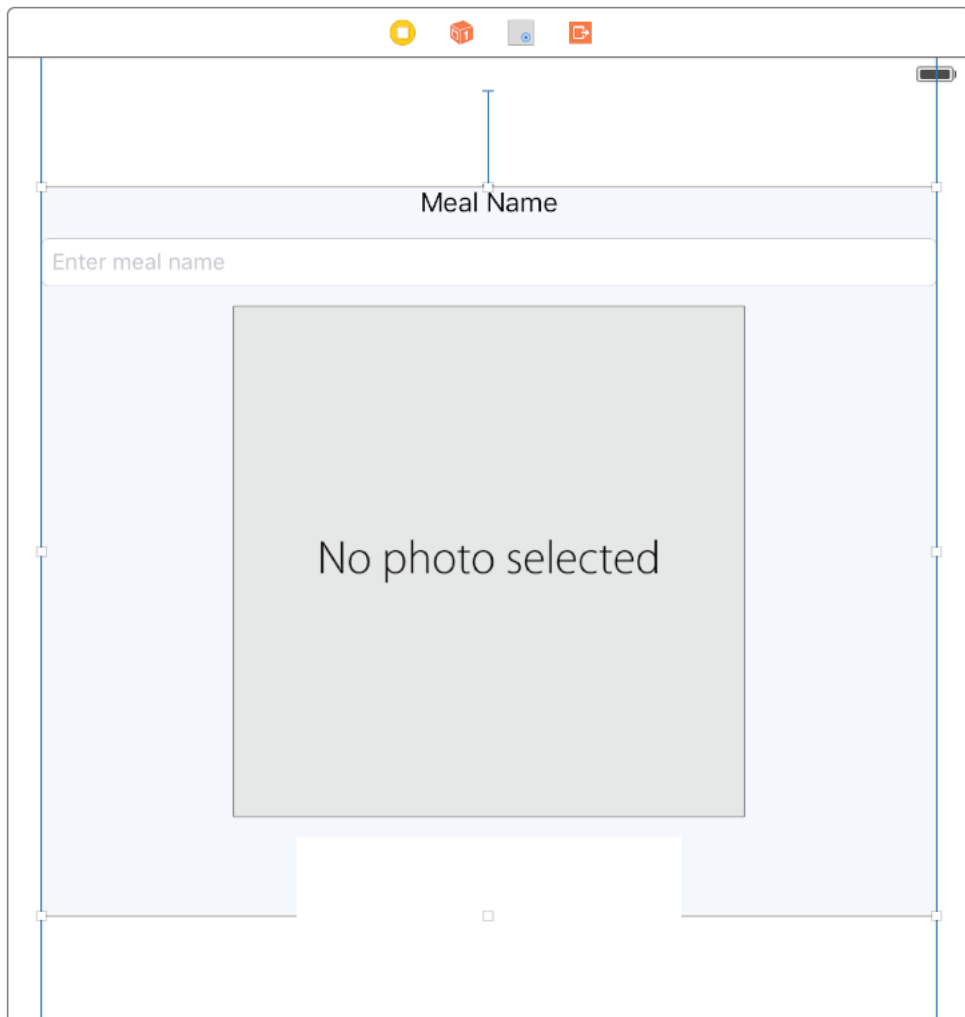


۶. کادر Attribute inspector را باز کنید.

۷. در Attribute inspector، فیلد Alignment را پیدا کرده و سپس گزینه ی Center را

انتخاب نمایید.

المان های موجود در stack view به صورت افقی وسط چین می شوند.



حال action method متصل به دکمه ی حذف شده را از کد پاک می کنید.

برای این منظور (پاک سازی پروژۀ از کدهای بلااستفاده)، مراحل زیر را دنبال نمایید:

۱. فایل ViewController.swift را باز کنید.

۲. متد `setDefaultLabelText(_:)` را از فایل ذکر شده حذف نمایید.

```
@IBAction func setDefaultLabelText(sender: UIButton) {
mealNameLabel.text = "Default Text"
}
```

در حال حاضر حذف همین قطعه کد کفایت می کند. در مبحث بعدی تغییراتی را نیز به

outlet لیبل اعمال خواهید کرد.

**تست کنید:** برنامه ی خود را اجرا نمایید. رفتار برنامه نباید نسبت به قبل تغییر کرده باشد. اما این بار دکمه ی Set Default Label Text دیگر وجود ندارد و المان های UI به صورت افقی در وسط صفحه بر روی هم قرار گرفته اند. علاوه بر آن دکمه ها بایستی در یک سطر و کنار هم چیده شده باشند. با کلیک بر روی هر یک از دکمه ها، همچنان متد (ratingButtonTapped(\_:)) صدا خورده می شود اما با توجه به تغییراتی که در کد برنامه اعمال کردید، این بار عکس دکمه ها به دنبال اجرای متد نام برده، تغییر می کنند.

**مهم:** چنانچه در زمان build/کامپایل با مشکل مواجه شدید، می توانید کلیدهای Command-Shift-K را جهت پاک سازی پروژه فشار دهید.

## درس ۷ : ساخت Data Model برای اپلیکیشن های IOS

### ساخت Data Model برای اپلیکیشن

در این مبحث، یک data model برای اپلیکیشن FoodTracker تعریف خواهید کرد. Data model داده های اپلیکیشن را در خود ذخیره کرده و نمای درختی از آن ها را ارائه می دهد.

### آنچه خواهید آموخت

- یک data model برای اپلیکیشن خود ایجاد نمایید.
- پیاده سازی failable initializer در کلاس اختصاصی خود.
- تفاوت مفهومی بین initializer های failable و nonfailable را تشریح کنید.
- Data model را با (نوشتن و اجرای) unit test آزمایش کرده و از کارکرد صحیح آن اطمینان حاصل نمایید.

### ساخت data model

اکنون یک data model ایجاد می کنید که داده های اپلیکیشن را در خود ذخیره کرده و اطلاعاتی که scene (صفحه ی محتوای برنامه) جاری برای کاربر در UI به نمایش می

گذارند را فراهم می نماید. برای این منظور، در ابتدا یک کلاس با سه property جهت ذخیره اطلاعات مربوط به اسم، عکس و امتیاز غذا تعریف می کنید.

به منظور ایجاد یک کلاس data model جهت ذخیره ی اطلاعات برنامه، مراحل زیر را دنبال نمایید:

۱. ابتدا این مسیر را طی نمایید: File > New > File یا کلیدهای Command-N را همزمان فشار دهید تا فایل جدید ایجاد شود.

۲. در سمت چپ کادر محاوره ای که نمایان می شود، گزینه ی Source را از زیر iOS انتخاب نمایید.

۳. Swift File را انتخاب نموده و بر روی دکمه ی Next کلیک نمایید.

فرایندی که برای ایجاد کلاس data model طی می کنید از پروسه ی ایجاد کلاسی که برای پیاده سازی کنترل امتیازدهی (کلاس RatingControl) دنبال کردید ( iOS > Cocoa Touch Class > Source) کاملاً متفاوت است چرا که data model یک کلاس پایه است که از کلاس دیگری ارث بری ندارد. اما اگر بخاطر داشته باشید کلاس RatingControl از کلاس UIView مشتق می شد.

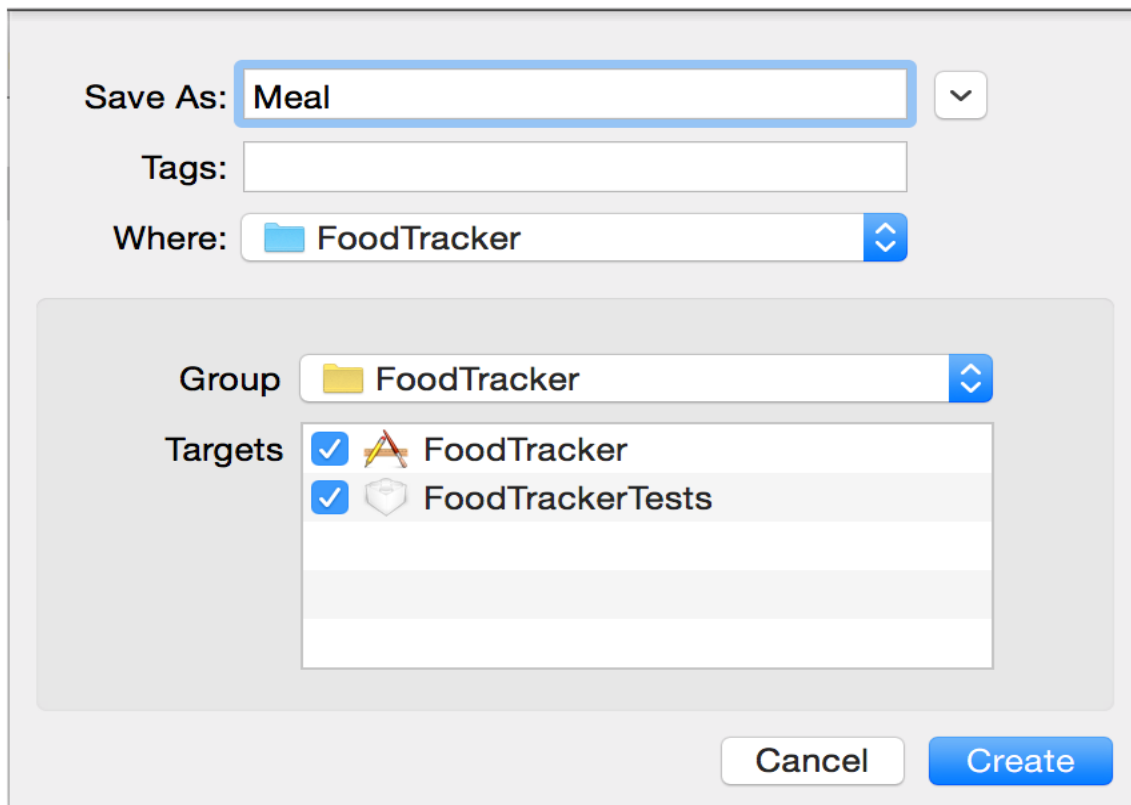
۱. داخل فیلد Save As، واژه ی Meal را به عنوان اسم فایل جدید انتخاب نمایید.

۲. فایل مورد نظر به صورت پیش فرض بر روی پوشه ی پروژه ذخیره می شود ( save location به صورت پیش فرض بر روی دایرکتوری پروژه تنظیم می شود).

فیلد Group به صورت پیش فرض بر روی اسم اپلیکیشن، FoodTracker، تنظیم می شود.

در بخش Targets، لازم است هر دو گزینه را فعال نمایید (کادر تیک دوم مربوط به تست های برنامه می باشد که باید برای این مبحث آن را انتخاب نمایید).





۳. بر روی دکمه ی Create کلیک نمایید.

Xcode یک فایل به نام Meal.swift ایجاد می کند.

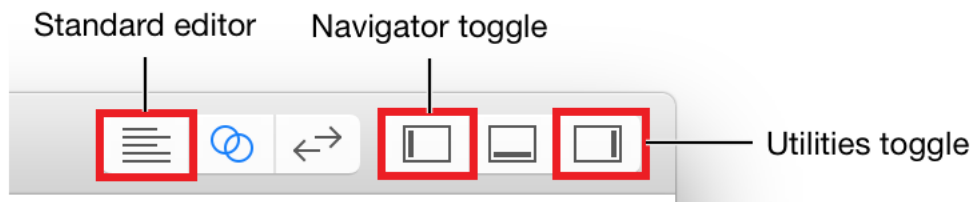
اسم غذا را می توانید در یک متغیر از نوع String، عکس را در قالب یک متغیر از جنس کلاس UIImage و امتیاز غذا را با استفاده از یک عدد صحیح یا Int ذخیره نمایید. از آنجایی که یک غذا همیشه یک اسم و امتیاز دارد، اما ممکن است عکس نداشته باشد، می توانید متغیر سوم را به وسیله ی عملگر "?"، optional تعریف کنید (بدین معنی که در صورت عدم وجود عکس بتواند مقدار nil بپذیرد).

جهت تعریف data model برای ذخیره و نگهداری اطلاعات مربوط به غذا، مراحل زیر را به ترتیب دنبال نمایید:

۱. در صورت باز بودن assistant editor، با کلیک بر روی دکمه ی Standard

editor (اشاره شده در تصویر زیر)، ویرایشگر اصلی محیط کاری Xcode را با

نمایید:



۲. حال فایل Meal.swift را باز نمایید.

۳. دستور import را جهت وارد کردن کتابخانه ی UIKit به صورت زیر ویرایش نمایید:

```
import UIKit
```

به صورت پیش فرض، یک فایل Swift چارچوب نرم افزاری/framework Foundation را وارد پروژه می کند تا شما بتوانید با ساختارهای داده ای نظیر آرایه کار کنید. در این مبحث از کلاس های چارچوب نرم افزاری UIKit استفاده خواهید کرد، بنابراین لازم است UIKit را در دستور import در بالای فایل لحاظ نمایید. لازم به ذکر است که UIKit، علاوه بر کلاس های خود، امکان دسترسی به کلاس های Foundation را نیز فراهم می نماید (از این جهت بهتر است دستور اضافی import Foundation را از فایل حذف کنید).

۴. در زیر دستور import، کد زیر را درج نمایید:

```
class Meal {
// MARK: Properties
var name: String
var photo: UIImage?
var rating: Int
}
```

این کد property هایی (متغیر) تعریف می کند که نقش ظرف را برای داده های مورد نیاز شما ایفا کرده و آن ها را در خود نگه می دارند. همان طور که می بینید برای تعریف property ها، بجای let از کلیدواژه ی var استفاده شده چرا که ممکن

است اطلاعات ذخیره شده در آن، در آینده تغییر کند (یا به عبارتی طی چرخه ی حیات آبجکت Meal مقدارشان عوض شود).

۵. اکنون در زیر property های مزبور، این کد را جهت تعریف یک initializer (متد سازنده/مقداردهنده ی اولیه) اضافه نمایید:

```
// MARK: Initialization
```

```
init(name: String, photo: UIImage?, rating: Int) {  
}
```

یادآور می شویم که initializer یک نمونه از روی کلاس جاری ساخته، آن را برای استفاده آماده می کند که مقداردهی اولیه ی property های کلاس و انجام دیگر تنظیمات لازم جزئی از این آماده سازی، محسوب می شود.

۶. بدنه ی این متد را به صورت زیر پیاده سازی نمایید (مقدار پارامترهای ارسال شده به متد سازنده را برابر property های کلاس قرار دهید):

```
// Initialize stored properties.
```

```
self.name = name
```

```
self.photo = photo
```

```
self.rating = rating
```

اینجا یک سوال جالب مطرح می شود: اگر سعی کنید یک آبجکت Meal با مقادیر غیر مجاز نظیر امتیاز منفی ایجاد کنید یا فیلد name را خالی بگذارید، چه اتفاقی می افتد؟ در چنین سناریویی بایستی مقدار nil را به نشانه ی اینکه امکان ایجاد آیتم مورد نظر وجود نداشته و فیلدها به ناچار با همان مقادیر پیش فرض مقدار دهی شده اند، از متد سازنده (initializer) برگردانید.

برای نیل به این هدف می بایست کدی (یک متد سازنده با دستور شرطی if در بدنه ی آن) اضافه کنید که ابتدا این دست سناریوها را بررسی کرده و در صورتی که در مقداردهی اولیه ی property های آبجکت با شکست مواجه شد، مقدار nil را در خروجی برگرداند.

۷. در انتهای پیاده سازی متد سازنده (initializer)، یک دستور شرطی if اضافه می کنید که مقادیر را بررسی کرده و در صورت برخورد با مقدار غیرمجاز، nil را به عنوان خروجی بازمی گرداند.

// Initialization should fail if there is no name or if the rating is negative.

اگر مقداری برای فیلد نیم ارائه نشده باشد یا مقدار امتیاز برای آبجکت منفی وارد شود، مقداردهی اولیه قاعدتا باید با شکست مواجه شود

```
if name.isEmpty || rating < 0 {
```

```
return nil
```

```
}
```

از آنجایی که متد سازنده (initializer) ممکن است در فرایند مقداردهی اولیه با

شکست مواجه شده و مقدار nil را در خروجی برگرداند، این امر را باید در

signature (خط تعریف) متد سازنده مشخص کنید.

۸. با کلیک بر روی آیکون قرمز رنگ fix-it (راه حلی که کامپایلر در پاسخ به خطا در

کدنویسی شما ارائه می دهد) عملگر "?" را به تعریف initializer اضافه نمایید.



```
init?(name: String, photo: UIImage?, rating: Int) {
```

متد سازنده ای (initializer) که به این صورت (با علامت سوال) نگارش می شود

در اصطلاح failable initializer نام دارد. این متد قادر است در صورت عدم

موفقیت در فرایند مقداردهی اولیه property های کلاس، nil را در خروجی

برگرداند.

در حال حاضر، متد سازنده ی `init?(name:photo:rating:)` می بایست ظاهری مشابه زیر داشته باشد:

// MARK: Initialization

`init?(name: String, photo: UIImage?, rating: Int) {`

// Initialize stored properties.

// مقداردهی اولیه ی متغیرهای کلاس

`self.name = name`

`self.photo = photo`

`self.rating = rating`

// Initialization should fail if there is no name or if the rating is negative.

در صورتی که مقداردهی اولیه با شکست مواجه شد، برای مثال مقدار ریتینگ منفی بود، باید در

خروجی مقدار نیل برگرداند/مقداردهی اولیه با شکست مواجه شود

`if name.isEmpty || rating < 0 {`

`return nil`

`}`

`}`

**تست کنید:** پروژه ی خود را کامپایل نمایید (`Product > Build`) یا فشردن همزمان

کلیدهای (`Command-B`). در حال حاضر کلاس `Meal` را برای منظور خاصی بکار نمی

برید، بلکه با `build` آن، صرفاً کامپایلر را از عدم وجود خطا در کد خود مطمئن می سازید

(برای مثال به کامپایلر اعلان می کنید که `initializer` را با درج ؟ از نوع `failable` تعریف

کردید و در کد خطای نگارشی وجود ندارد). اگر در کدنویسی شما خطایی وجود داشت، در

آن صورت می بایست هشدارهایی که کامپایلر در خصوص بخش های مختلف کد ارائه می

دهد را خوانده، سپس دستورالعمل های تشریح شده در مبحث جاری را مرور نمایید و کد

را مطابق با آن ویرایش کنید.

### تست کردن داده ها با استفاده از `unit test`

گرچه کد `data model` شما با موفقیت کامپایل می شود، اما هنوز آن را به طور کامل در

اپلیکیشن خود جاسازی نکرده اید. در نتیجه، نمی توان با اطمینان کامل گفت که همه چیز

را به درستی پیاده سازی کردید. همچنین ممکن است در زمان اجرا با مشکلات و شرایطی

مواجه شوید که انتظارش را ندارید (به عنوان مثال ورودی کاربر از حداکثر مقدار مجاز

تعیین شده توسط برنامه بیشتر یا کمتر باشد. مانند زمانی که به یک غذا مقدار منفی داده می شود).

برای مدیریت این عدم قطعیت و کسب اطمینان از کارکرد صحیح برنامه، می توانید برای کد خود unit test بنویسید. Unit test عبارت است از آزمایش قطعه کدهای کوچک و مستقل برنامه (مجزا از دیگر بخش های برنامه) جهت اطمینان حاصل نمودن از عملکرد صحیح آن ها. کلاس Meal گزینه ی مناسبی برای اجرای unit test می باشد.

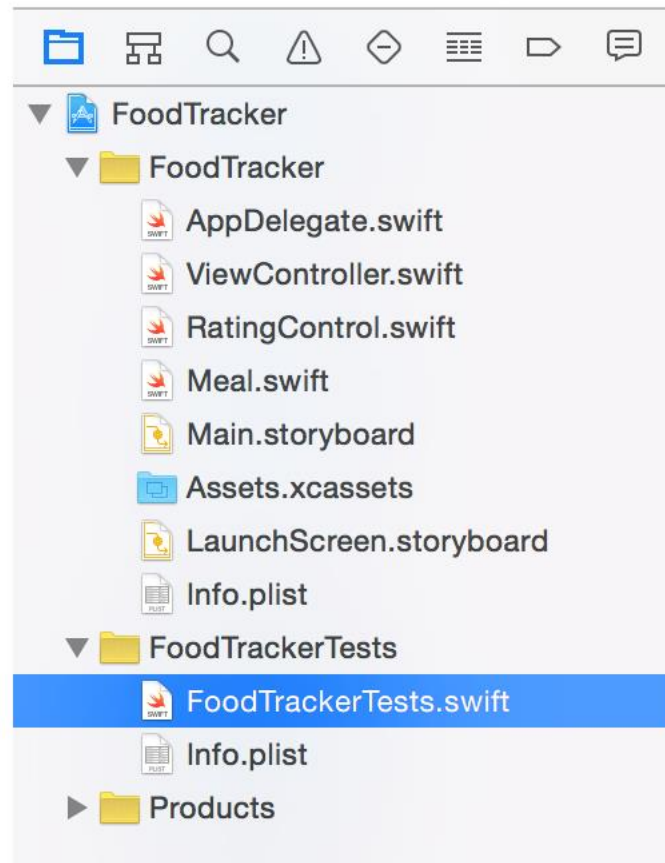
محیط Xcode خود یک فایل unit test به عنوان بخشی از فایل الگو (template) Single View Application ایجاد کرده و به صورت آماده در اختیار شما قرار می دهد.

به منظور مشاهده و دسترسی به فایل unit test جهت اپلیکیشن FoodTracker، مراحل زیر را گام به گام دنبال نمایید:

۱. پوشه ی FoodTrackerTests را در project navigator با کلیک بر روی آیکن

مثلی که در کنار آن به نمایش گذاشته شده، باز نمایید:





۲. فایل FoodTrackerTests.swift را باز نمایید.

حال زمان مختصری را به درک کد موجود در این فایل تخصیص دهید.

```
import UIKit
import XCTest
class FoodTrackerTests: XCTestCase {
    override func setUp() {
        super.setUp()
        // Put setup code here. This method is called before the invocation of each test
        method in the class.
    }
    override func tearDown() {
        // Put teardown code here. This method is called after the invocation of each test
        method in the class.
        super.tearDown()
    }
    func testExample() {
```

// This is an example of a functional test case.

```
XCTAssert(true, "Pass")
```

```
}
```

```
func testPerformanceExample() {
```

// This is an example of a performance test case.

```
self.measureBlock() {
```

// Put the code you want to measure the time of here.

```
}
```

```
}
```

```
}
```

Xcode یک فریم ورک تست گیری به نام XCTest دارد که به واسطه ی دستور import XCTest، این فریم ورک تست گیری را جهت آزمایش بخش های مختلف برنامه در فایل جاری وارد می کنید. unit test ها (توابع تست گیری) در سطح کلاسی به نام FoodTrackerTests تعریف می شوند که این کلاس خود اعضایش را از کلاس XCTestCase به ارث می برد. همان طور که در کد بالا مشاهده می کنید، comment هایی در بالای دو متد setUp() و tearDown() درج شده که توضیحاتی درباره ی کاربرد آن ها ارائه می دهد.

در کل دو نوع unit test وجود دارد: ۱. تست های functional که جهت تست قسمت های مختلف برنامه به صورت مجزا، نوشته می شوند (به طوری که تغییر در بخش های دیگر را به دنبال نداشته باشد و به بخش های دیگر وابسته نباشد). به عبارت دیگر، به واسطه ی این نوع تست عملکرد کامپوننت های مختلف نرم افزار را آزمایش می کنید و مطمئن می شوید رفتار تعریف شده و عملیات مورد انتظار را پیاده سازی می کند یا خیر ۲. تست های performance که جهت بررسی سرعت اجرا و کارایی کد تنظیم می شوند (کد با سرعت مورد انتظار اجرا می شود یا خیر).

تا اینجا آموزش هیچ عملیات سنگینی ننوشته ایم که نیاز به تست سرعت اجرا و کارایی آن داشته باشیم، از اینرو فعلا فقط به نوشتن تست های functional بسنده می کنیم.



بتر است عنوان متدهای تست گیری خود را با واژه ی “test” آغاز کرده و باقی آن را نیز طوری انتخاب کنید که فهم و شناسایی کاربرد آن بعده ها برای شما آسان باشد. به عنوان مثال، می توانید یک تست ساده بنویسید که بررسی می کند آیا property های کلاس Meal به درستی مقداردهی اولیه می شوند یا خیر و اسم این تست را testMealInitialization انتخاب نمایید.

برای نوشتن تستی که صحت مقداردهی اولیه ی property های آبجکت Meal را بررسی می کند، مراحل زیر را گام به گام دنبال نمایید:

۱. از داخل فایل FoodTrackerTests.swift، تست های الگو (template) را حذف نمایید.

```
import UIKit
import XCTest
class FoodTrackerTests: XCTestCase {
}
// برای تست عملکرد کدهای این برنامه به هیچ یک از پیاده سازی های الگویی ( template
// implementation) که محیط Xcode به صورت آماده در اختیار شما قرار می دهد، نیاز
// ندارید و می توانید آن ها را حذف نمایید.
```

۲. به قبل از آخرین ({}), کد زیر را اضافه نمایید:

```
// MARK: FoodTracker Tests
```

۳. این خط صرفاً یک comment است که به وسیله ی آن شرح مختصری درباره ی کاربرد قسمت های مختلف تست و اینکه هر تست برای آزمایش کدام بخش از برنامه نوشته شده است، ارائه می دهید.

۴. حال در زیر این comment، یک unit test جدید به صورت زیر اضافه نمایید:

```
// Tests to confirm that the Meal initializer returns when no name or a negative
// rating is provided.
func testMealInitialization() {
}
```

۵. ابتدا یک مورد تست (test case) اضافه نمایید که مطمئنید با موفقیت اجرا می شود. comment و دستورات زیر را به متد testMealInitialization() اضافه نمایید.

// Success case.

```
let potentialItem = Meal(name: "Newest meal", photo: nil, rating: 5)
XCTAssertNotNil(potentialItem)
```

۶. XCTAssertNotNil تست کرده و مطمئن می شود که آبجکت Meal پس از پروسه ی مقداردهی اولیه، nil نباشد. این بدین معنی است که متد سازنده (initializer) توانسته با موفقیت یک آبجکت Meal با مقادیر ارائه شده به عنوان پارامتر، ایجاد کند.

۷. اکنون یک مورد تست اضافه نمایید که در آن مقداردهی اولیه ی آبجکت Meal مطمئناً با شکست مواجه می شود. comment و دستورات زیر را به متد (تست گیری) testMealInitialization() اضافه نمایید:

// Failure cases.

```
let noName = Meal(name: "", photo: nil, rating: 0)
XCTAssertNil(noName, "Empty name is invalid")
```

XCTAssertNil انتظار دارد (assert) که آبجکت noName دارای مقدار nil باشد، بدین معنی که پروسه ی مقداردهی اولیه که توسط متد initializer باید انجام شود با شکست مواجه می شود. به عبارت دیگر شما انتظار دارید که مقداردهی اولیه ناموفق باشد چرا که پارامتر name یک رشته ی تهی است.

۸. اکنون یک مورد تست دیگر اضافه کنید که در آن همواره آبجکت Meal در مقداردهی اولیه با شکست مواجه می شود، اما این بار (با استفاده از متد XCTAssertNotNil) assert کنید که مقداردهی اولیه باید با موفقیت انجام شود. برای این منظور کد زیر را به متد تست گیری testMealInitialization() اضافه نمایید:

```
let badRating = Meal(name: "Really bad rating", photo: nil, rating: -1)
XCTAssertNotNil(badRating)
```

شما انتظار دارید که این مورد تست نیز ناموفق باشد زیرا مقدار پارامتر rating منفی است.

در حال حاضر بدنه ی متد تست گیری ()testMealInitialization می بایست شبیه نمونه ی زیر باشد:

```
// Tests to confirm that the Meal initializer returns when no name or a negative
rating is provided.
```

```
func testMealInitialization() {
    // Success case.
    let potentialItem = Meal(name: "Newest meal", photo: nil, rating: 5)
    XCTAssertNotNil(potentialItem)
    // Failure cases.
    let noName = Meal(name: "", photo: nil, rating: 0)
    XCTAssertNil(noName, "Empty name is invalid")
    let badRating = Meal(name: "Really bad rating", photo: nil, rating: -1)
    XCTAssertNotNil(badRating)
}
```

می توانید با فشردن کلیدهای Command-U تمامی تست های خود را به طور همزمان اجرا نمایید و یا هر یک را به صورت مجزا اجرا کنید. آخرین تست باید با شکست مواجه شود چرا که آبجکت در اصل دارای مقدار منفی است و اگر بخاطر داشته باشید با استفاده از متد XCTAssertNotNil انتظار داشتید که مقدار nil نباشد، در حالی که در اصل nil و فاقد مقدار مجاز است (در initializer failable شرطی اضافه کردیم که در آن مقدار متغیر rating نمی توانست منفی باشد).

به منظور اجرای متد تست گیری ()testMealInitialization، مراحل زیر را دنبال نمایید:

۱. در فایل FoodTrackerTests.swift، متد ()testMealInitialization را پیدا کنید.
۲. در سمت چپ اسم متد (تست گیری)، شکل لوزی را پیدا کنید.

```
import XCTest

class FoodTrackerTests: XCTestCase {

    // MARK: FoodTracker Tests

    // Tests to confirm that the Meal initializer returns nil when no name or a negative rating is provided.
    func testMealInitialization() {
        // Success case.
        let potentialItem = Meal(name: "Newest meal", photo: nil, rating: 5)
        XCTAssertNotNil(potentialItem)

        // Failure cases.
        let noName = Meal(name: "", photo: nil, rating: 0)
        XCTAssertNil(noName, "Empty name is invalid")

        let badRating = Meal(name: "Really bad rating", photo: nil, rating: -1)
        XCTAssertNotNil(badRating)
    }
}
```

۳. اشاره گر موس را بر روی شکل لوزی معلق نگه داشته تا دکمه ی اجرا به نمایش در آید.

```
import XCTest

class FoodTrackerTests: XCTestCase {

    // MARK: FoodTracker Tests

    // Tests to confirm that the Meal initializer returns nil when no name or a negative rating is provided.
    func testMealInitialization() {
        // Success case.
        let potentialItem = Meal(name: "Newest meal", photo: nil, rating: 5)
        XCTAssertNotNil(potentialItem)

        // Failure cases.
        let noName = Meal(name: "", photo: nil, rating: 0)
        XCTAssertNil(noName, "Empty name is invalid")

        let badRating = Meal(name: "Really bad rating", photo: nil, rating: -1)
        XCTAssertNotNil(badRating)
    }
}
```

۴. بر روی دکمه ی اجرا کلیک نمایید تا unit test اجرا شود.

**تست کنید:** اپلیکیشن را تست کنید. برنامه باید با تستی که شما نوشته اید، اجرا شود. انتظار می رود که دو نمونه ی اول با موفقیت انجام (pass) شوند، اما سومین باید ناموفق باشد (fail).

```
import XCTest

class FoodTrackerTests: XCTestCase {
    // MARK: FoodTracker Tests

    // Tests to confirm that the Meal initializer returns nil when no name or a negative rating is provided.
    func testMealInitialization() {
        // Success case.
        let potentialItem = Meal(name: "Newest meal", photo: nil, rating: 5)
        XCTAssertNotNil(potentialItem)

        // Failure cases.
        let noName = Meal(name: "", photo: nil, rating: 0)
        XCTAssertNil(noName, "Empty name is invalid")

        let badRating = Meal(name: "Really bad rating", photo: nil, rating: -1)
        XCTAssertNotNil(badRating)
    }
}
```

همان طور که می بینید unit test به شما کمک می کند خطاها را در کد خود پیدا و متعاقباً مدیریت کنید. اگر واقعاً انتظار داشتید که آبجکت Meal دارای مقدار یا non-nil باشد، آنگاه این خطا را در طول فرایند تست گیری از برنامه ی خود پیدا می کردید (در اینجا از قصد یک مورد تست غلط نوشتید، به همین دلیل به بخش مربوطه مراجعه کرده و به سادگی آن را برطرف می کنید).

حال جهت برطرف کردن خطای مورد تست، مراحل زیر را دنبال نمایید:

۱. در فایل FoodTrackerTests.swift، متد تست گیری testMealInitialization() را پیدا کنید.
۲. آخرین خط کد را به صورت زیر ویرایش کنید:

`XCTAssertNil(badRating, "Negative ratings are invalid, be positive")`

۳. متد تست گیری testMealInitialization() اکنون می بایست دارای پیاده سازی زیر باشد:

```
// Tests to confirm that the Meal initializer returns when no name or a negative
rating is provided.
func testMealInitialization() {
    // Success case.
    let potentialItem = Meal(name: "Newest meal", photo: nil, rating: 5)
    XCTAssertNotNil(potentialItem)
    // Failure cases.
```

```
let noName = Meal(name: "", photo: nil, rating: 0)
XCTAssertNil(noName, "Empty name is invalid")
let badRating = Meal(name: "Really bad rating", photo: nil, rating: -1)
XCTAssertNil(badRating, "Negative ratings are invalid, be positive")
}
```

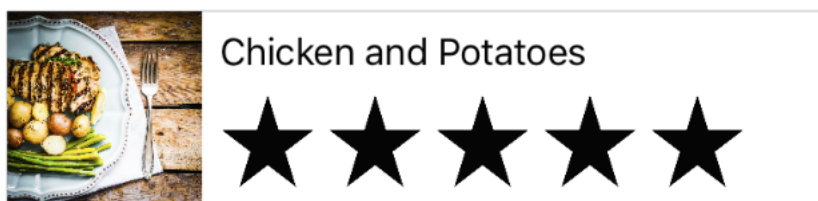
**تست کنید:** برنامه ی شما با تستی که هم اکنون نوشتید اجرا می شود. این بار تمامی موارد تست باید با موفقیت اجرا شوند (پاس شوند).

می توان گفت که Unit test یک بخش اساسی و جدایی ناپذیر از کدنویسی و روش های صحیح ساخت اپلیکیشن است چرا که به شما کمک می کند خطاهایی که ممکن است در صورت عدم اجرای unit test نادیده بگیرید را در متن برنامه ی (source code) خود پیدا کرده و مدیریت نمایید. همان طور که از نامش پیدا است، unit test باید مازولار باشد. به عبارت دیگر، بهتر است برای هر بخش از کد خود یک unit test مجزا تنظیم نمایید. در واقع هر تست می بایست رفتار فقط یک بخش از برنامه را تست کند. اگر unit test هایی بنویسید که بیش از حد طولانی یا پیچیده باشند، آنگاه یافتن خطا و اینکه کجای برنامه اشکال دارد، به مراتب دشوارتر می شود.

## درس ۸ : تعریف Table View در Swift

### تعریف Table View در Swift ساخت یک scene با نمای جدولی

در این مبحث شما به ساخت صفحه ی اصلی اپلیکیشن FoodTracker خواهید پرداخت. پس از آن یک صفحه ی محتوا (scene) با نمای جدولی یا table view-based خواهید ساخت که غذاهای اضافه شده توسط کاربر را در خانه های جدول به نمایش می گذارد. همچنین خانه های جدول را با تنظیمات دلخواه خود پیاده سازی خواهید نمود که هر یک از غذاها در آن به صورت زیر نمایش داده می شود:



## آنچه خواهید آموخت

۱. یک scene (صفحه ی محتوا) دیگر به فایل storyboard خود اضافه نمایید.
۲. با اجزا و کامپوننت های اصلی یک table view (نمای جدولی) آشنا شوید.
۳. یک خانه ی جدول با تنظیمات دلخواه خود طراحی و ایجاد نمایید.
۴. با نقش delegate ها، حین کار با table view و منابع داده ای (data source) به خوبی آشنا شوید.
۵. داده های مورد نیاز را در قالب یک آرایه ذخیره کرده و با آن ها کار کنید.
۶. داده ها را به صورت داینامیک (در زمان اجرا) در نمای جدولی (table view) به نمایش بگذارید.

## ساخت scene آغازین (اولین صفحه ی محتوا) اپلیکیشن FoodTracker

تا به اینجا، برنامه ی کاربردی FoodTracker فقط یک صفحه ی محتوا (scene) دارد که مدیریت آن همان طور که انتظار می رود بر عهده ی یک view controller است. این view controller نمایشگر یک صفحه است که به کاربر امکان می دهد غذای جدید به لیست غذاهای جاری اضافه نموده و همچنین به غذای مورد نظر امتیاز دهد. در این مبحث یک صفحه ی محتوای دیگر ایجاد خواهید کرد که لیست تمامی غذاها را برای کاربر به نمایش می گذارد. خوشبختانه، در iOS کلاس درون ساخته ای تعبیه شده که امکان ایجاد لیست قابل پیمایش جهت نمایش فهرستی طولانی از آیتم ها (به همراه نوار پیمایش) را به راحتی برای شما فراهم می سازد. این کلاس table view (UITableView) نام دارد.

هر آبجکت table view به تبع توسط یک table view controller (UITableViewController) مدیریت می شود. table view controller یک کلاس مشتق یا ارث بری شده از UIViewController است که ویژه ی اداره ی منطق مربوط به table view طراحی و تنظیم شده است. صفحه محتوای (scene) جدید برنامه ی خود را بر اساس یک table view controller پی ریزی و طراحی خواهید کرد.

به منظور افزودن یک scene با نمای جدولی (table view) به فایل storyboard برنامه ی جاری خود، مراحل زیر را گام به گام دنبال نمایید:

۱. ابتدا storyboard خود، فایل Main.storyboard را باز کنید.
۲. کادر Object library را در utility area محیط کاری Xcode باز نمایید. (یا این مسیر را طی نمایید: View > Utilities > Show Object Library)
۳. در Object library، آبجکت Table View Controller را پیدا کرده و انتخاب نمایید.

۴. آبجکت Table View Controller را با اشاره گر موس از Object library کشیده و در سطح canvas، موجود در سمت چپ scene جاری (Meal scene) جایگذاری نمایید.

اگر یک table view به همراه محتوا مشاهده می کنید که به هنگام کشیدن به ناحیه ی canvas عکس العملی از خود نشان نمی دهد، احتمالاً به این خاطر است که بجای table view controller، یک table view را بر روی سطح canvas کشیده اید. جالب است بدانید که table view زیرمجموعه ی کلاس table view controller است و تنها یکی از مولفه هایی است که توسط این کلاس مدیریت شده و در نهایت در اختیار شما قرار می گیرد. شما برای اپلیکیشن خود به کلاس table view controller که یک پکیج کامل است احتیاج دارید. به همین خاطر بایستی خود table view controller را یافته و در سطح canvas جایگذاری نمایید.

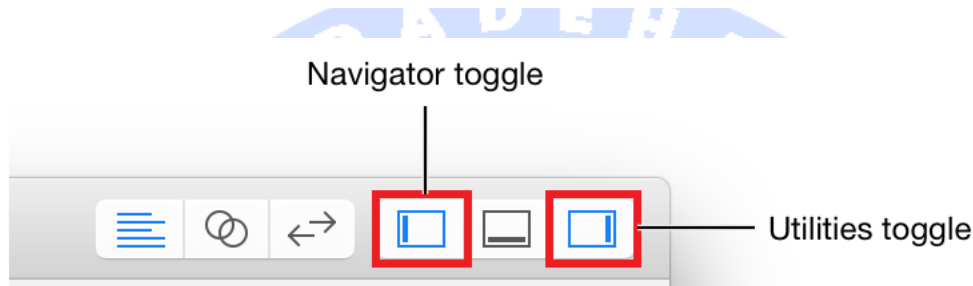
شما در حال حاضر دو صفحه ی محتوا دارید، یکی برای نمایش لیستی از غذا ها و دیگری برای افزودن غذای جدید به لیست غذاهای جاری.

بهتر است اولین صفحه ای که کاربر با اجرای برنامه ی شما مشاهده می کند یک لیست غذا (meal list) باشد. برای این منظور می بایست صفحه ی نمایش لیست غذاها را به وسیله ی ابزاری که محیط کاری Xcode در اختیار شما قرار می دهد، به عنوان صفحه اصلی برنامه ی خود انتخاب نمایید.



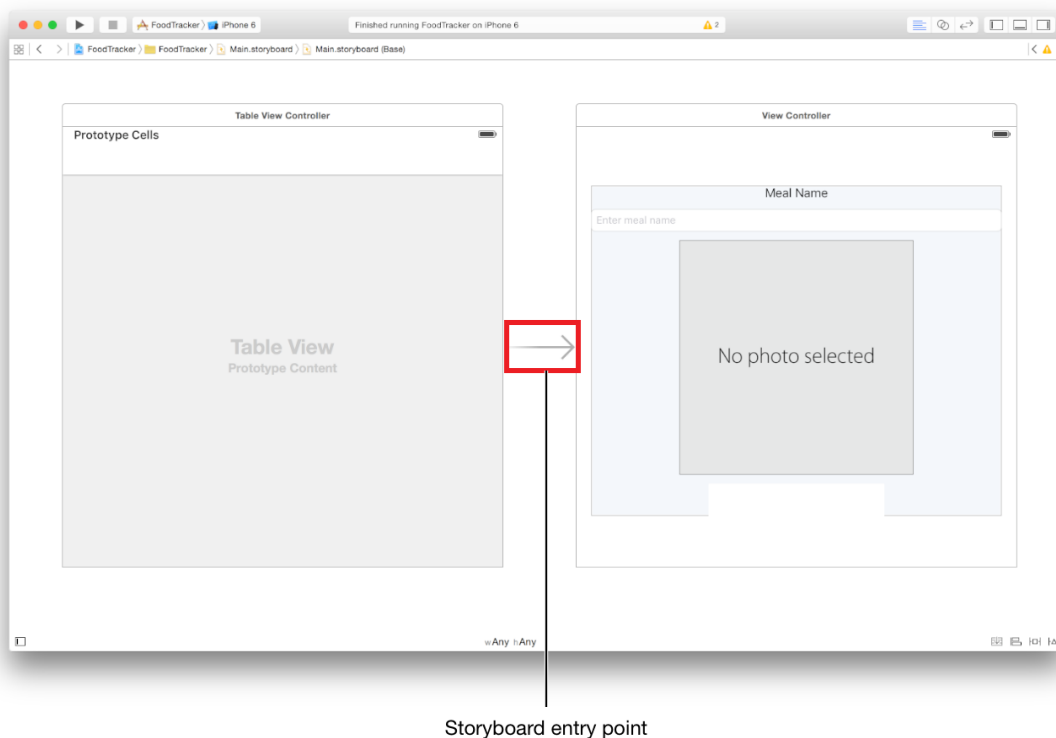
به منظور تنظیم table view controller به عنوان اولین صفحه ی محتوا برنامه ی خود، مراحل زیر را گام به گام دنبال نمایید:

۱. اگر به فضای کاری بیشتری نیاز دارید، می توانید project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار محیط کاری Xcode (اشاره شده در تصویر زیر) پنهان نمایید:

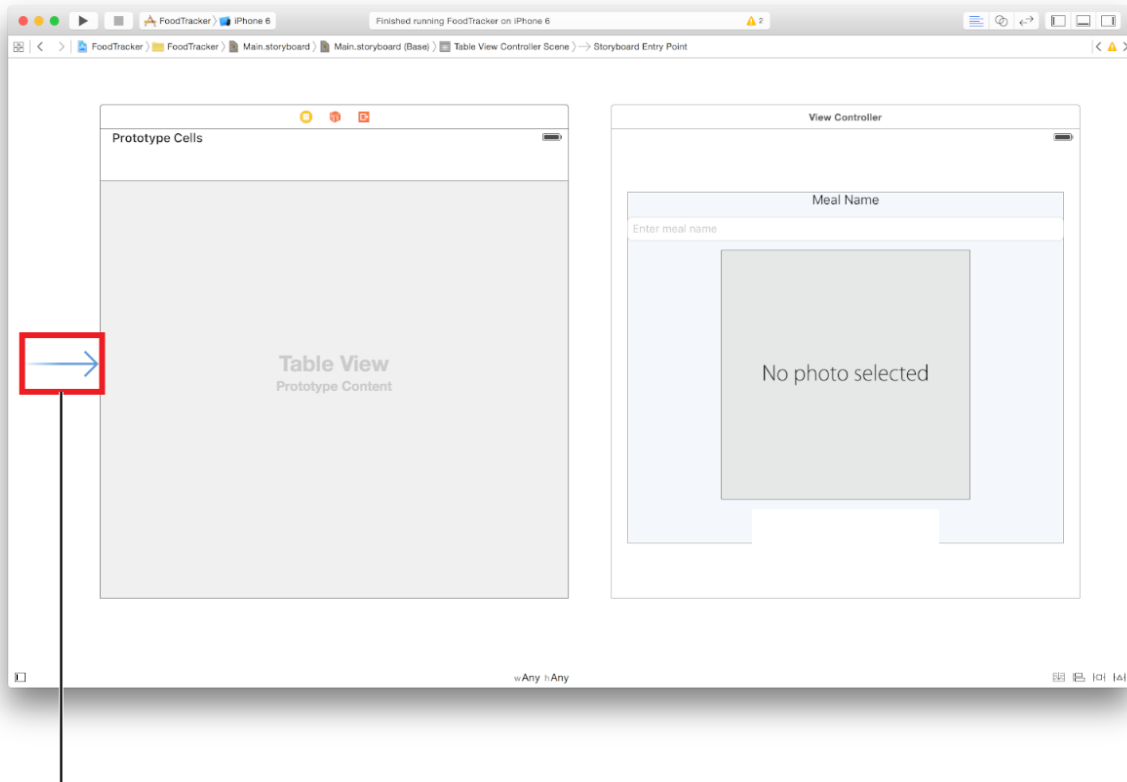


۲. در صورت لزوم، می توانید کادر outline view را نیز پنهان نمایید.

۳. اولین صفحه محتوای برنامه (storyboard entry point) را از سطح meal scene (صفحه ای که امکان افزودن غذای جدید را به کاربر می دهد) کشیده و در table view controller جایگذاری نمایید.

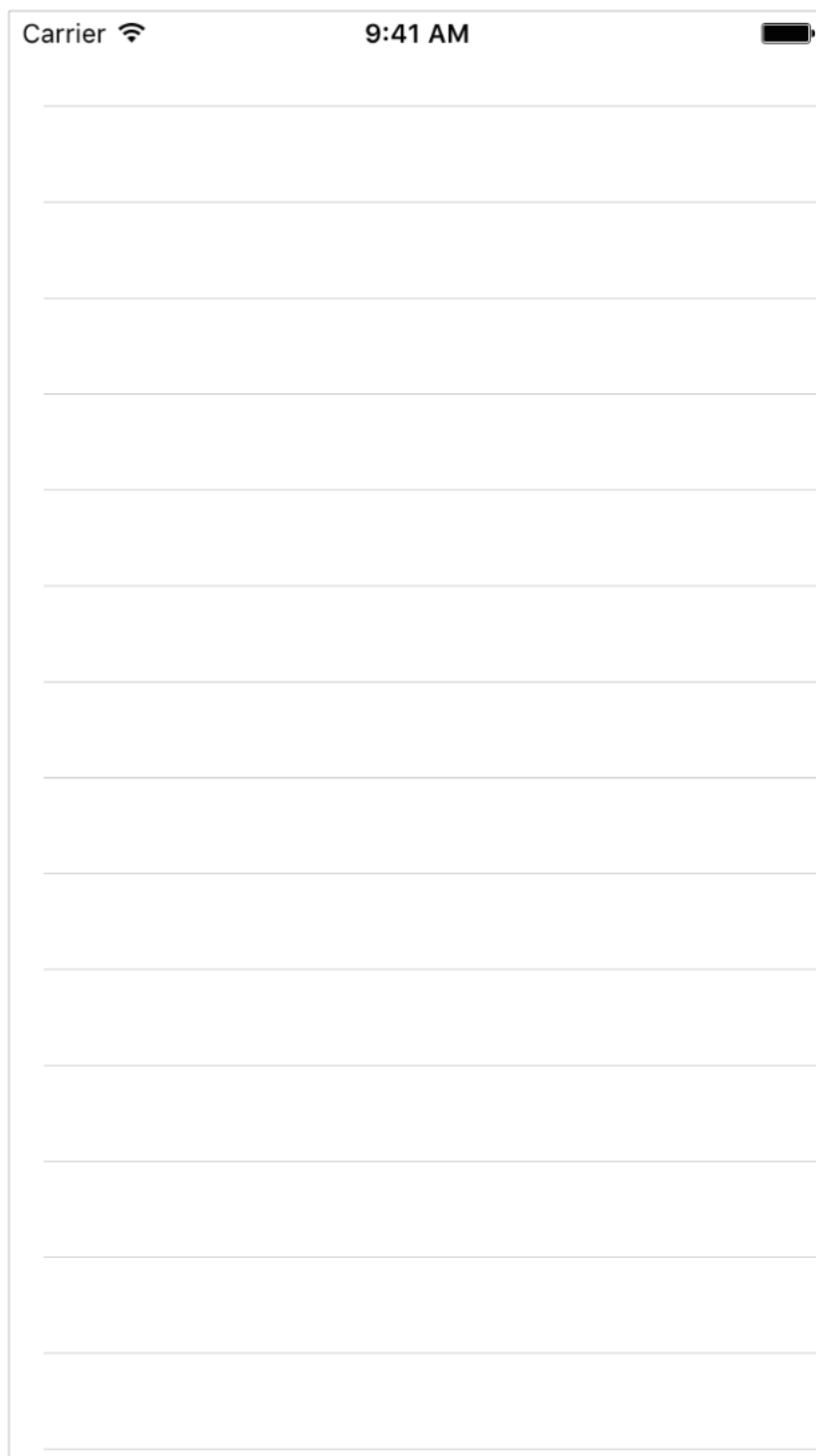


با این کار table view controller به عنوان اولین view controller در storyboard انتخاب و تنظیم می شود، بدین معنی که به مجرد اجرای برنامه، اولین صفحه ای که برای کاربر به نمایش در می آید، table view controller و محتوای آن (لیست غذاها) است.



Storyboard entry point

**تست کنید:** اپلیکیشن خود را اجرا کنید. بجای meal scene (صفحه ی افزودن غذای جدید) که المان هایی نظیر text field، image view و rating control را در UI به نمایش می گذارد، بایستی یک table view (نمای جدولی) خالی از محتوا – صفحه ای با خطوط افقی که سطرهایی فاقد محتوا را ایجاد کرده اند – مشاهده نمایید.



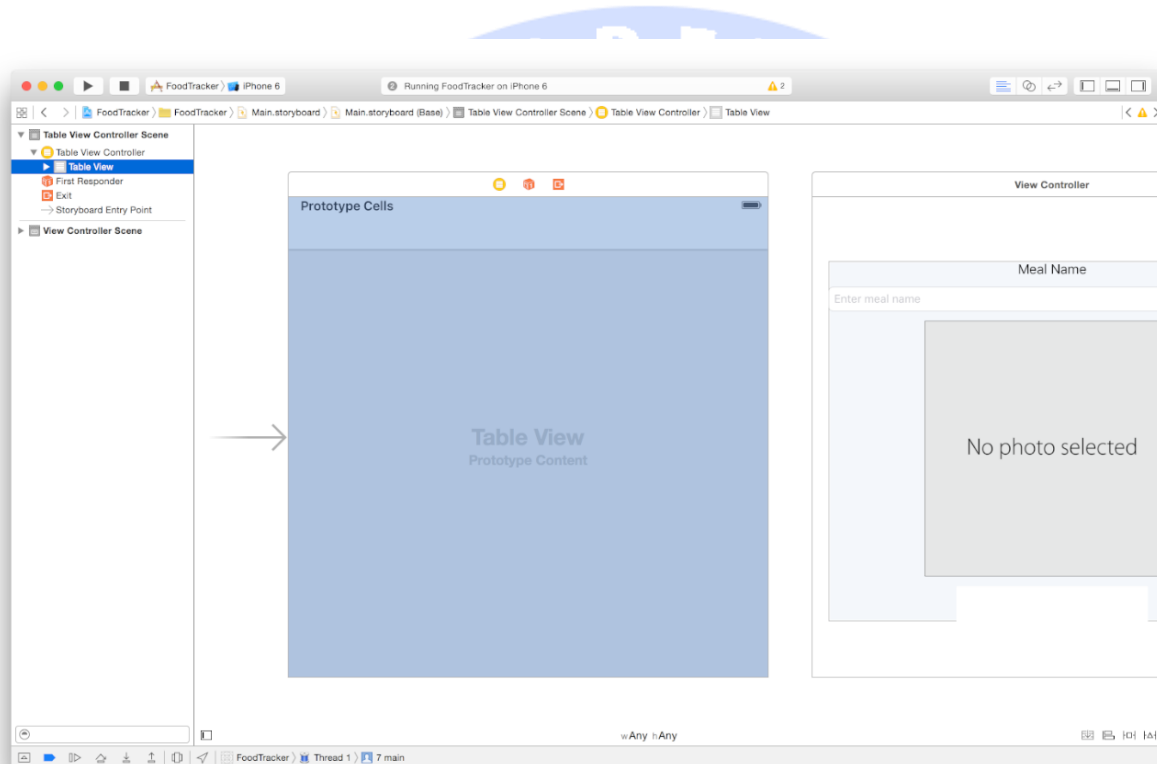
برای اینکه **table view** فعلی دقیقاً مناسب کار شما باشد و بتوانید آن را در برنامه ی خود استفاده کنید، لازم است تغییراتی در تنظیمات آن اعمال نمایید.

**به منظور تنظیم table view، مراحل زیر را به ترتیب دنبال نمایید:**

۱. در فایل storyboard خود، outline view را باز کنید.

۲. در کادر outline view، بر روی Table View کلیک نمایید.

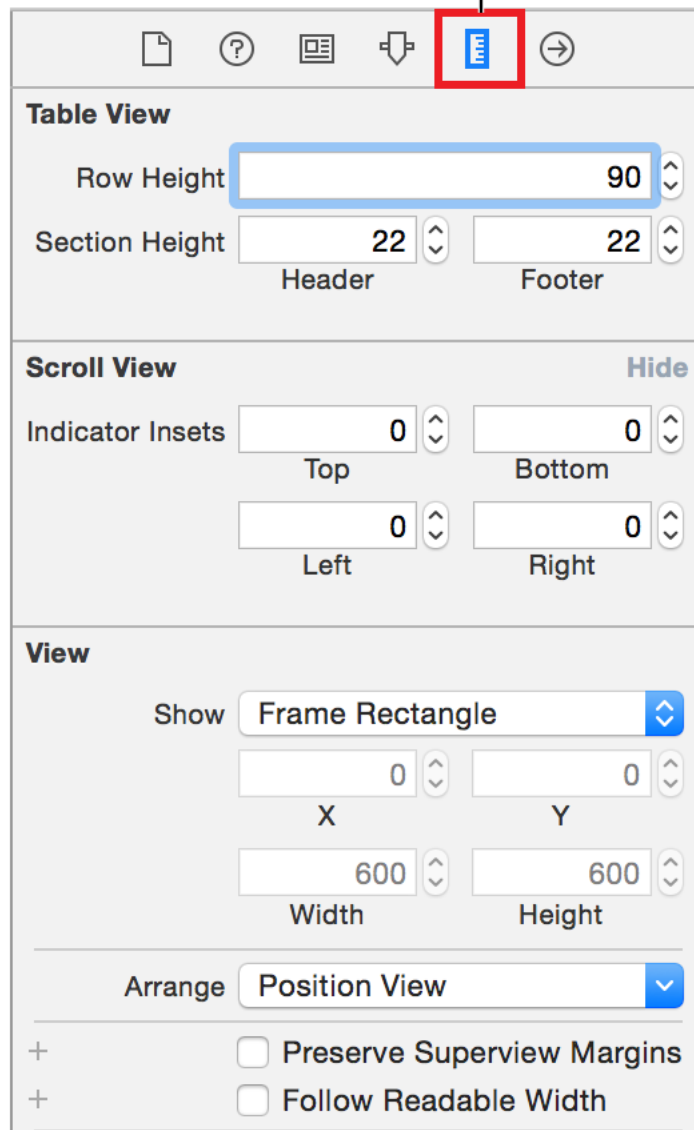
Table view زیرمجموعه ی Table View Controller Scene > Table View می باشد. برای مشاهده ی table view، کافی است بر روی آیکن Controller مثلث شکلی که در کنار آبجکت های نام برده نمایش داده می شود، کلیک نمایید.



۳. پس از کلیک بر روی table view و انتخاب آن، کادر Size inspector را از utility area باز کنید.

یادآور می شویم که برای باز کردن کادر Size inspector، می بایست بر روی دکمه ی پنجم از سمت چپ در inspector selector bar کلیک نمایید. در این کادر می توانید اندازه و مکان قرارگیری آبجکت مورد نظر را در storyboard مطابق نیاز تنظیم نمایید.

Size inspector



۴. داخل کادر Size inspector، اولین فیلد که Row Height نام دارد را یافته و مقدار ۹۰ را در آن وارد نمایید. کلید Return را فشار دهید.

فعلاً به این حد از تنظیمات table view بسنده می‌کنیم. پس از طراحی رابط کاربری table view (خانه‌های جدول) مجدداً به تنظیمات این آبجکت در Size inspector خواهیم پرداخت.

## طراحی اختصاصی خانه های جدول

برای مدیریت ظاهر سطرهای جدول در المان `table view` بایستی یک کلاس فرزند از کلاس `UITableViewCell` که وظیفه ی مدیریت خانه های جدول و محتوای درونی آن ها را بر عهده دارد ایجاد نمایید. این کلاس تعدادی رفتار از پیش تعریف شده و استایل (style) پیش فرض دارد. استایل پیش فرضی که این کلاس برای خانه های جدول ارائه می دهد در بیشتر موارد نیاز شما را برطرف می کند. اما اگر محتوایی که می خواهید در هر خانه ی جدول به نمایش بگذارید به فضای بیشتری نیاز دارد، آنگاه می بایست استایل خانه های جدول خود را تنظیم نمایید.

جهت ایجاد یک کلاس فرزند از `UITableViewCell`، مراحل زیر را دنبال کنید:

۱. ابتدا این مسیر را طی نمایید: `File > New > File` یا کلیدهای `Command-N` را فشار دهید.
  ۲. در سمت چپ کادر محاوره ای که نمایان می شود، `source` را از زیر `IOS` انتخاب نمایید.
  ۳. `Cocoa Touch Class` را انتخاب کرده و دکمه ی `Next` را کلیک نمایید.
  ۴. در فیلد `Class`، واژه ی `Meal` را درج نمایید.
  ۵. در فیلد `"Subclass of"`، کلاس `UITableViewCell` را انتخاب نمایید.
- عنوان کلاس به `MealTableViewCell` تغییر می کند. `Xcode` با تعیین اسم مرتبط صراحتاً مشخص می کند که کلاس ایجاد شده از جنس `UITableViewCell` بوده و برای تنظیم دلخواه خانه های جدول می باشد. این اسم را تغییر ندهید.
۶. در فیلد `Language`، زبان پروژه را بر روی `Swift` تنظیم نمایید.
  ۷. بر روی `Next` کلیک کنید.
- محل ذخیره سازی فایل کلاس (`save location`) به صورت پیش فرض بر روی پوشه ی پروژه تنظیم می شود.
- گزینه ی `Group` به صورت پیش فرض با اسم اپلیکیشن شما، `FoodTracker`، مقداردهی می شود.

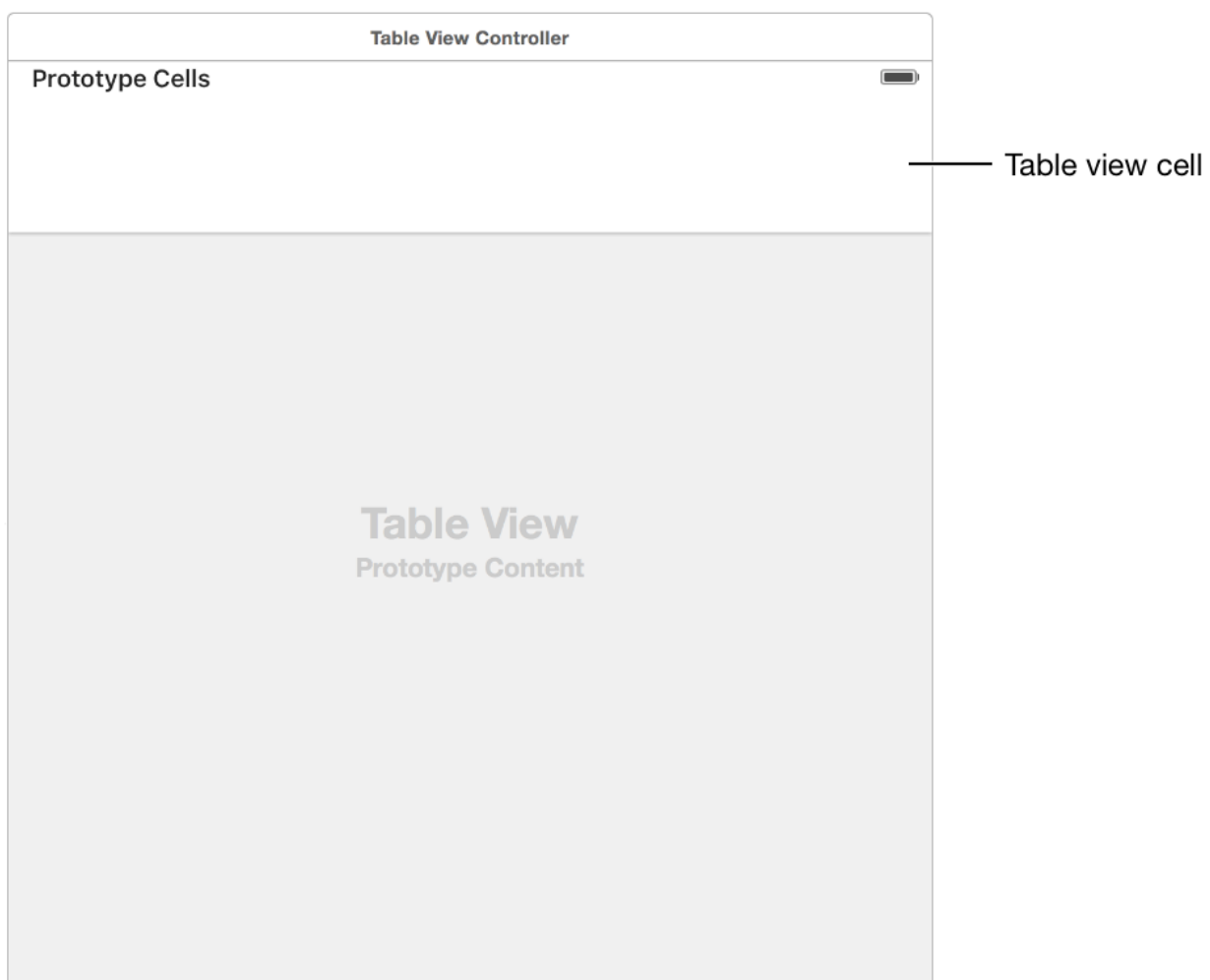
در بخش Targets، فقط کادر تیک اول باید انتخاب شده باشد.

۸. لازم نیست سایر گزینه هایی که به صورت پیش فرض مقداردهی شده اند را تنظیم نمایید.

Xcode فایل با مشخصات ارائه شده ایجاد می کند: MealTableViewCell.swift.

حال فایل Storyboard خود را باز نمایید.

اگر به scene دربردارنده ی table view controller در storyboard خود توجه کنید، می بینید که فقط یک خانه از جدول به نمایش می گذارد.



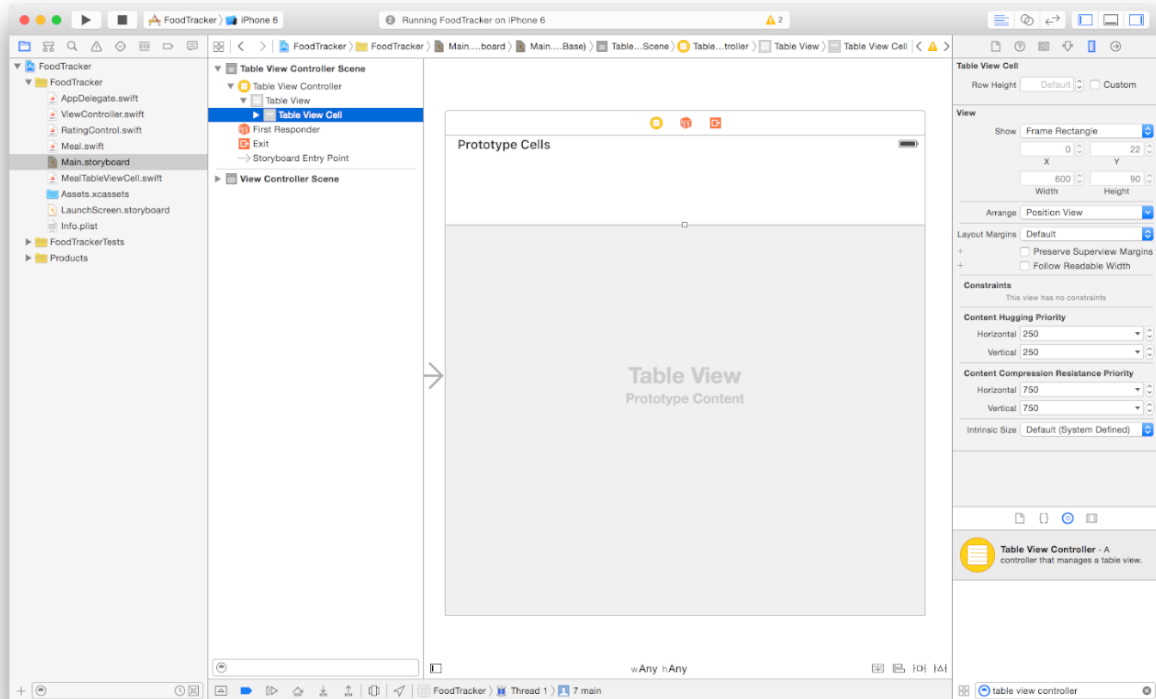
این خانه به منزله ی یک نمونه ی اولیه یا الگو جهت ساخت دیگر خانه های جدول محسوب می شود. ظاهر و رفتاری که برای این خانه از جدول تعیین می کنید، توسط دیگر خانه های جدول در table view ارث بری می شود. در وهله ی اول بایستی خانه ی نمونه را به صورت دلخواه تنظیم نمایید.

برای این منظور، لازم است خانه ی المان table view را در scene به کلاس اختصاصی خود که رفتار و ظاهر خانه ی جدول را می خواهید در آن پیاده سازی کنید، متصل نمایید.

به منظور تنظیم خانه ی جدول مطابق نیاز خود، مراحل زیر را دنبال نمایید:

۱. در کادر outline view، بر روی Table View Cell کلیک نمایید.

خانه ی جدول مورد نظر زیرمجموعه ی گره های Table View Controller Scene محسوب می شود. برای مشاهده و دسترسی به خانه ی جدول (table view cell)، لازم است بر روی آیکن های مثلثی شکل مربوطه کلیک نمایید.



۲. پس از کلیک بر روی table view cell و انتخاب آن، کادر Attributes inspector را از utility area باز نمایید.

۳. در کادر Attribute inspector، فیلدی که Identifier نام دارد را پیدا کرده و MealTableViewCell را وارد نمایید. حال کلید Return را فشار دهید. انجام کامل این مرحله ضروری است. علت آن را بعداً متوجه خواهید شد.



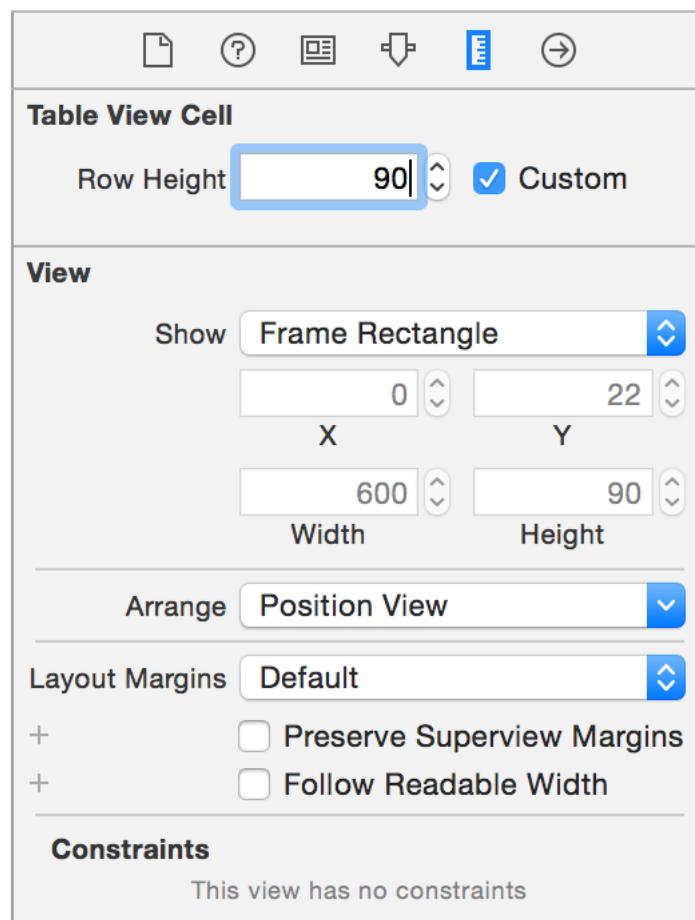
۴. در **Attribute inspector**، فیلدی که **Selection** نام دارد را پیدا کرده و گزینه **None** را از آن انتخاب نمایید.

با انتخاب این گزینه، زمانی که کاربر بر روی خانه ی جدول کلیک می کند، خانه ی جدول هایلایت نمی شود.

۵. اکنون کادر **Size inspector** را باز نمایید.

۶. در **Size inspector**، فیلدی که **Row Height** نام دارد را پیدا کرده، مقدار ۹۰ را وارد نمایید.

۷. در کنار این فیلد یک کادر تیک به نام **Custom** وجود دارد. آن را فعال نمایید:



کلید **Return** را فشار داده تا خانه ی جدول با مقدار طول جدید تنظیم شده و نمایش داده شود.

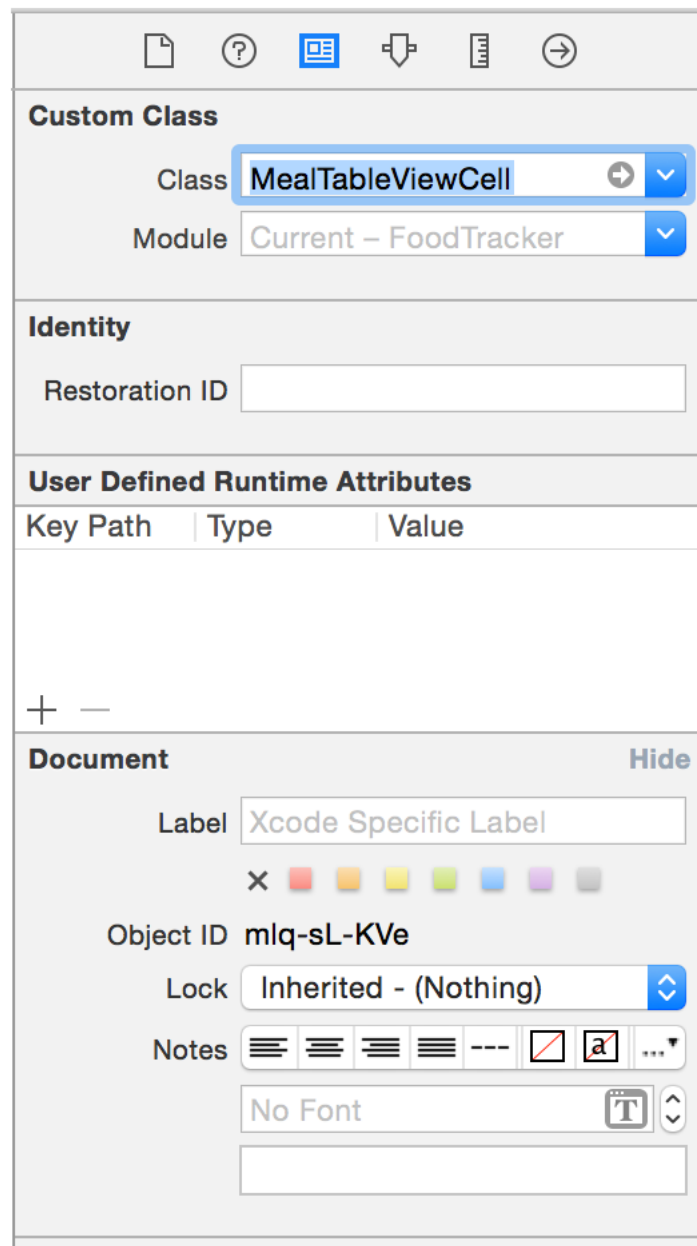
۸. کادر **Identity inspector** را باز نمایید.

یادآور می شویم که **identity inspector** به شما اجازه می دهد تا **property** های

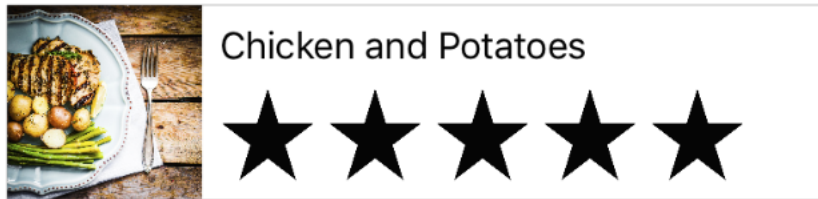
یک آبجکت را که مربوط به **identity** آن می باشند، نظیر این که آبجکت مورد نظر از

روی کدام کلاس ساخته شده یا به کدام کلاس تعلق دارد را ویرایش نمایید.

۹. در کادر Identity inspector، فیلدی با نام Class را پیدا کرده و از منوی آن، آیتم MealTableViewCell را انتخاب نمایید.



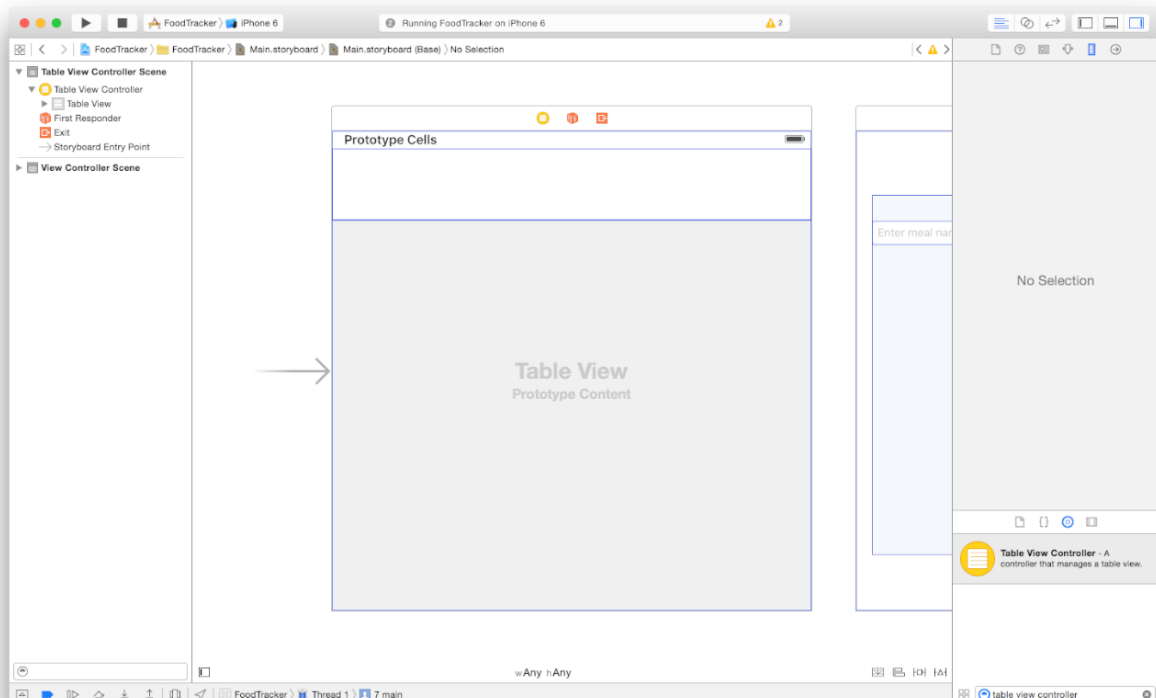
پس از انجام تنظیمات مربوطه به این خانه از جدول، می توانید ظاهر و ال دلخواه خود را برای آن مستقیماً در storyboard طراحی کنید. در این خانه از جدول اسم، عکس و امتیاز غذا به نمایش گذاشته شده است. ظاهر خانه ی جدول مشابه زیر خواهد بود:



همان طور که در تصویر فوق مشاهده می کنید، به یک label، image view و rating control نیاز دارید. می توانید کنترل امتیازدهی که در مباحث قبلی ایجاد کرده و آماده دارید را مجدداً برای این بخش مورد استفاده قرار دهید.

جهت طراحی UI خانه ی جدول با ظاهر اختصاصی و دلخواه خود، می بایست مراحل زیر را دنبال نمایید:

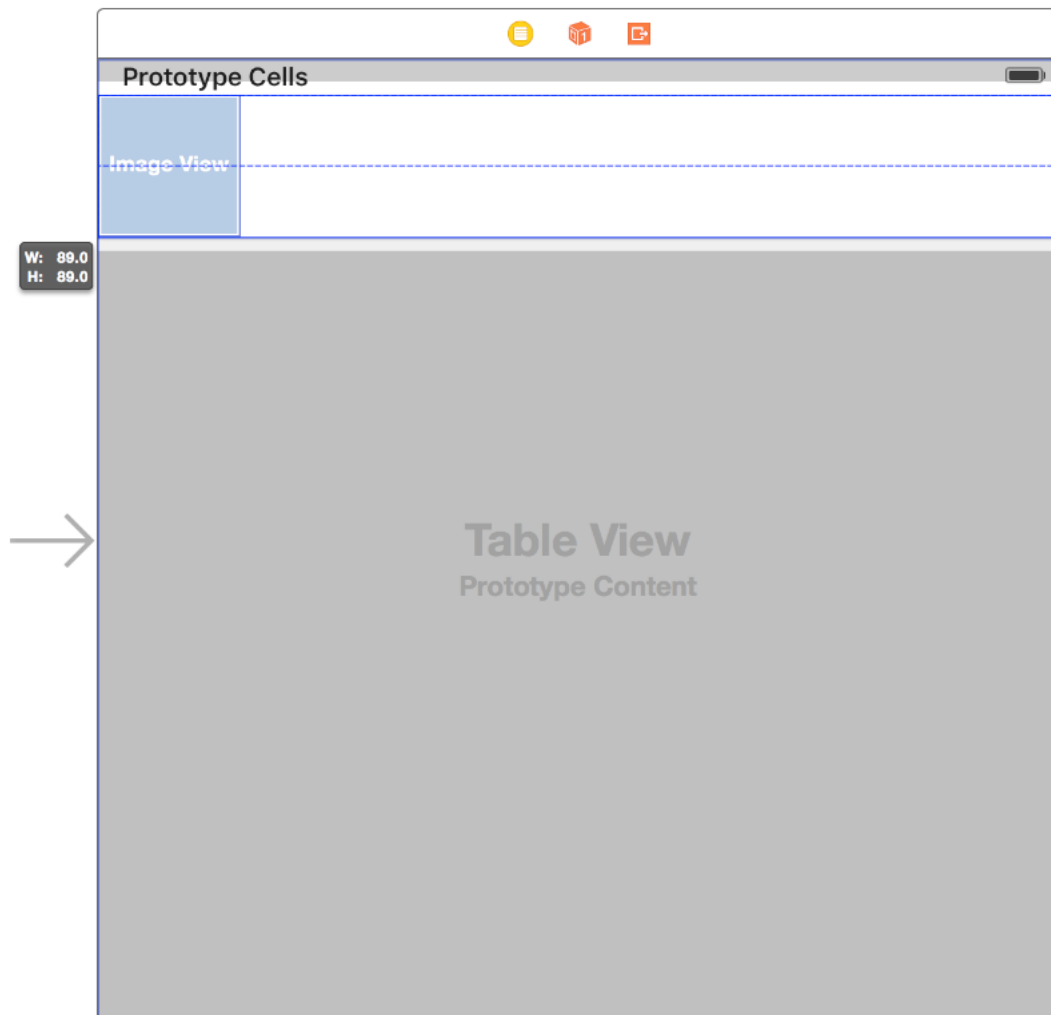
۱. ابتدا این مسیر را طی کرده: Editor > Canvas > Show Bounds Rectangles تا لبه های المان مورد نظر در UI دقیقاً مشخص گشته، ترازبندی و چیدمان آیتم ها در خانه ی جدول مورد نظر به مراتب آسان تر شود.



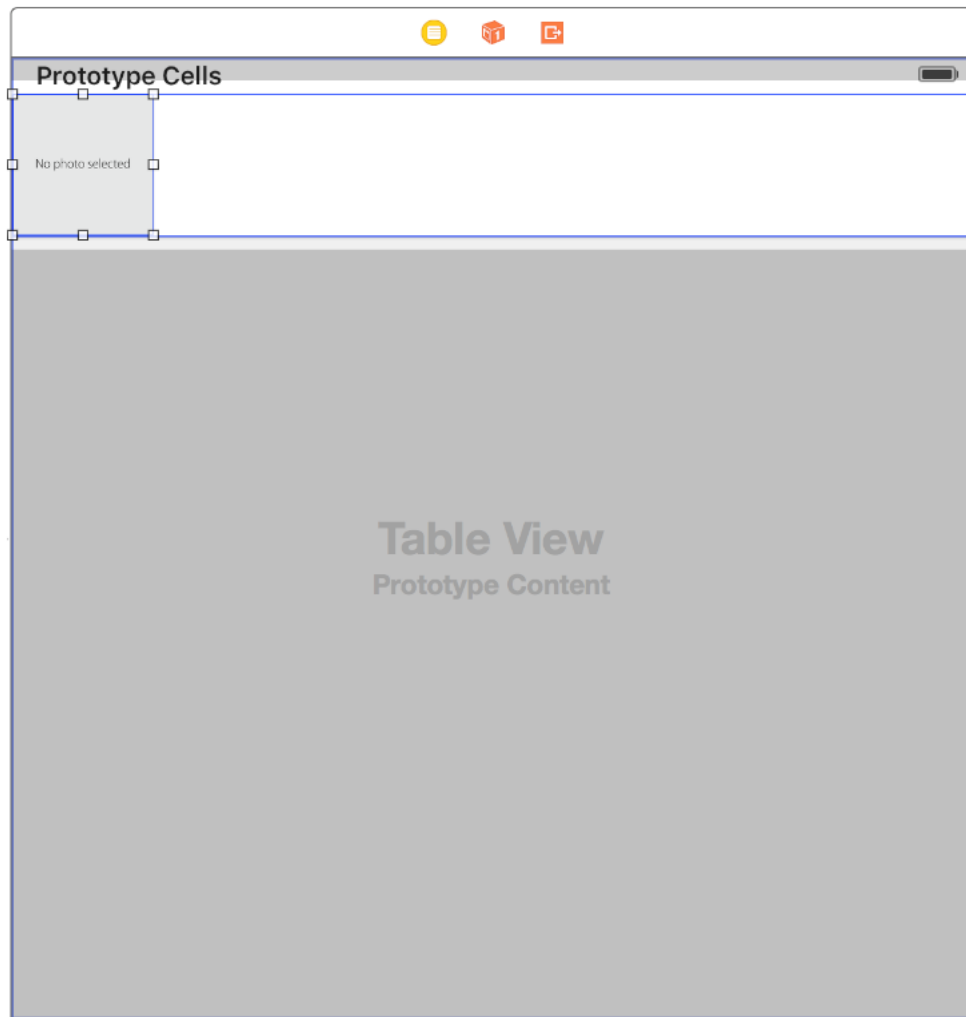
۲. آبجکت Image View را از Object library انتخاب کرده و آن را با استفاده از

اشاره گر موس بر روی خانه ی جدول جایگذاری نمایید.

۳. Image view را با اشاره گر موس کشیده و آن را طوری تنظیم کنید قالب یک مربع را به خود بگیرد و هم سطح با لبه ی سمت چپ، بالا و پایین خانه ی جدول قرار گیرد.

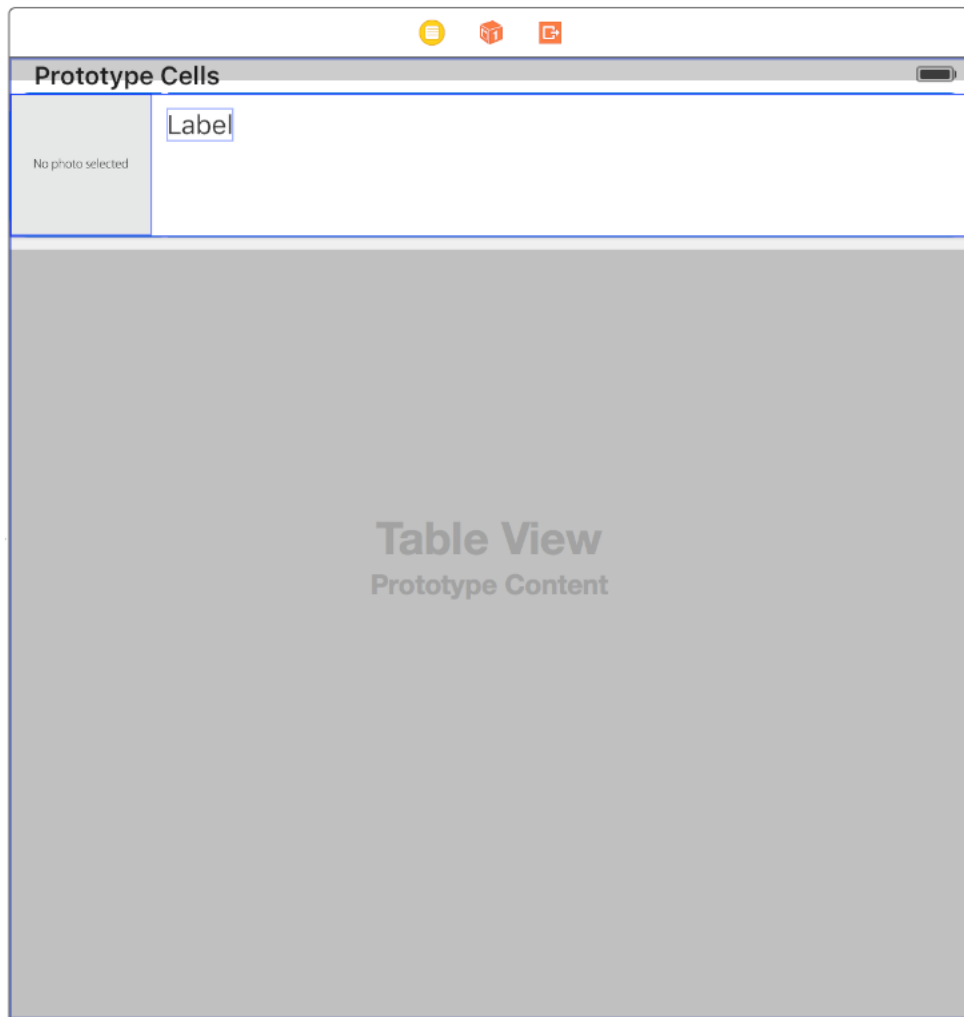


۴. اگر در مبحث قبلی عکس پیش فرض به پروژه خود اضافه نکردید، اکنون آن را به همان image view الحاق نمایید.
۵. پس از انتخاب image view، کادر Attribute inspector را در utility area محیط کاری Xcode باز نمایید.
۶. داخل Attribute inspector، فیلدی که Image نام دارد را یافته و گزینه ی defaultPhoto را در آن انتخاب نمایید.

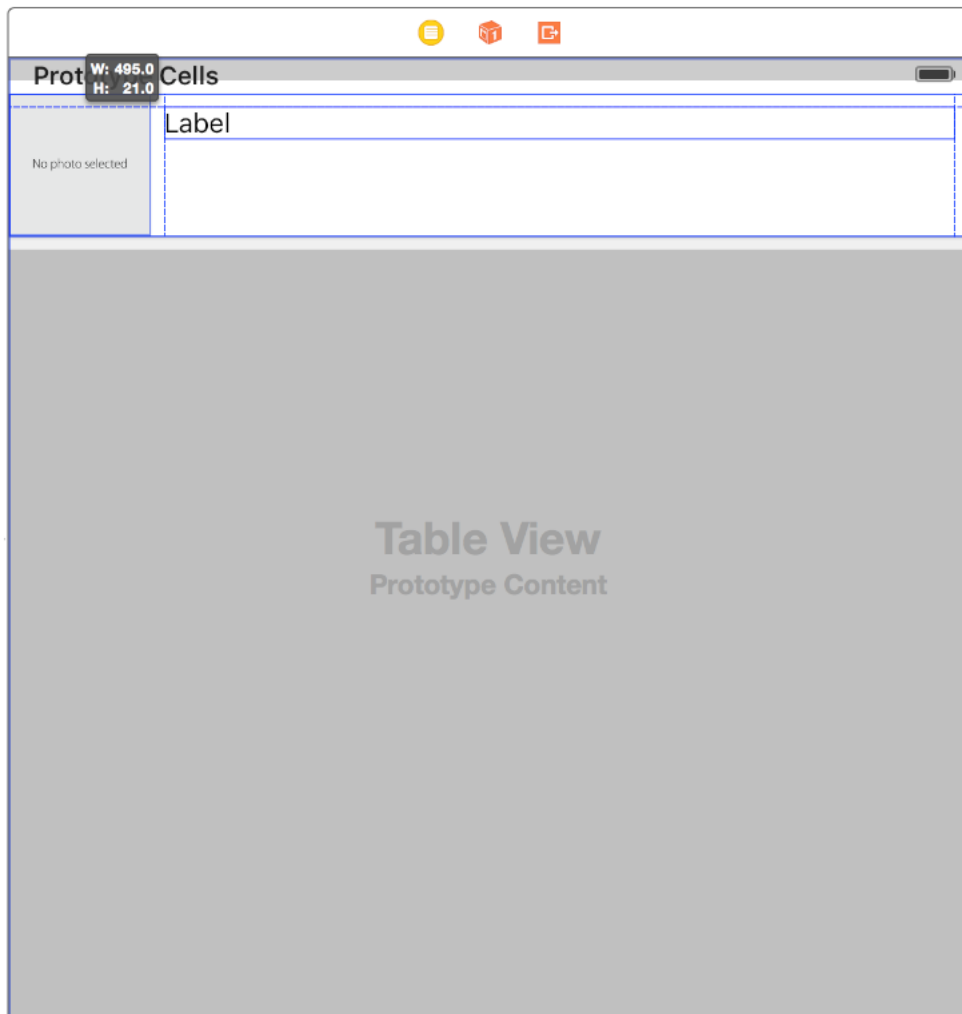


۷. آبجکت label را از library Object کشیده و آن را در سطح خانه ی جدول مورد نظر جایگذاری کنید.

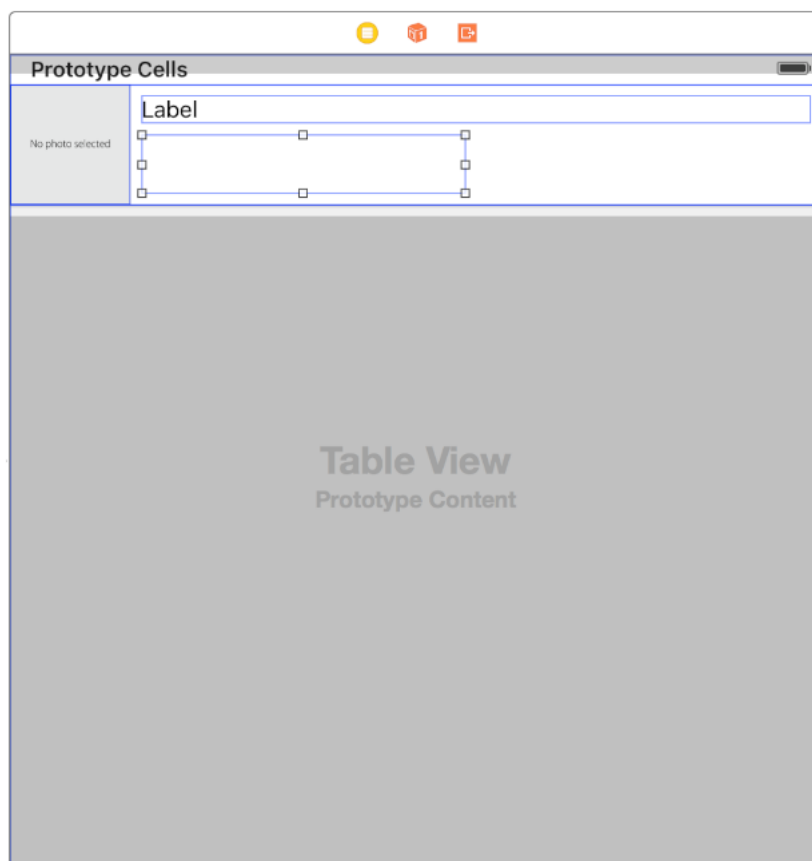
۸. اکنون آبجکت مزبور را با اشاره گر موس کشیده و آن را هم تراز با حاشیه ی (margin) بالا خانه ی جدول، سمت راست المان image view قرار دهید.



۹. لبه ی سمت راست المان Label را با اشاره گر کشیده و تا حاشیه ی سمت چپ خانه ی جدول آن را بکشید.



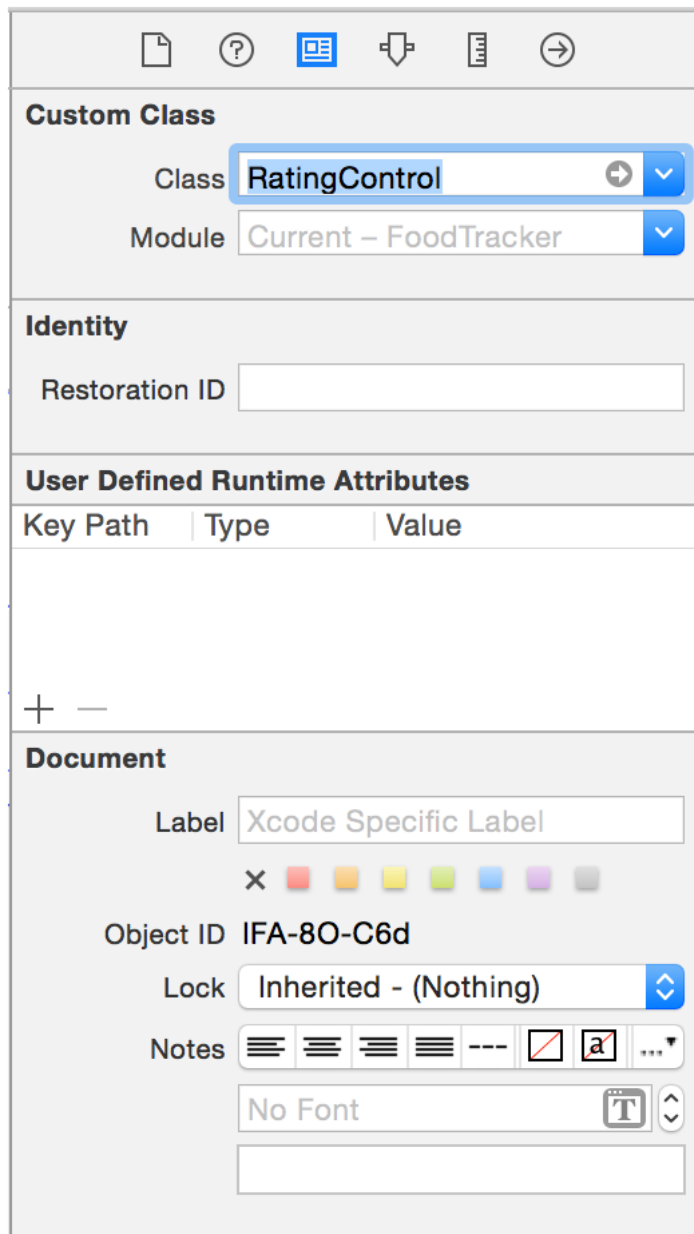
۱۰. حال زمان آن رسیده که یک آبجکت ساده ی View را از Object library انتخاب کرده و آن را در سطح خانه ی جدول جایگذاری نمایید.
۱۱. پس از انتخاب View نام برده، کادر Size inspector را جهت تنظیم اندازه ی المان مورد نظر در utility area محیط کاری Xcode باز نمایید.
۱۲. داخل کادر Size inspector، مقدار ۴۴ را جهت تنظیم طول المان در فیلد Height و ۲۴۰ را جهت تنظیم پهنا در فیلد Width وارد نمایید. پ
۱۳. View را کشیده و در زیر المان label، همتراز با حاشیه ی سمت چپ آن قرار دهید.



۱۴. پس از انتخاب view، کادر Identity inspector را باز نمایید.

۱۵. داخل کادر Identity inspector، فیلدی که Class نام دارد را یافته و RatingControl را از آن انتخاب نمایید.





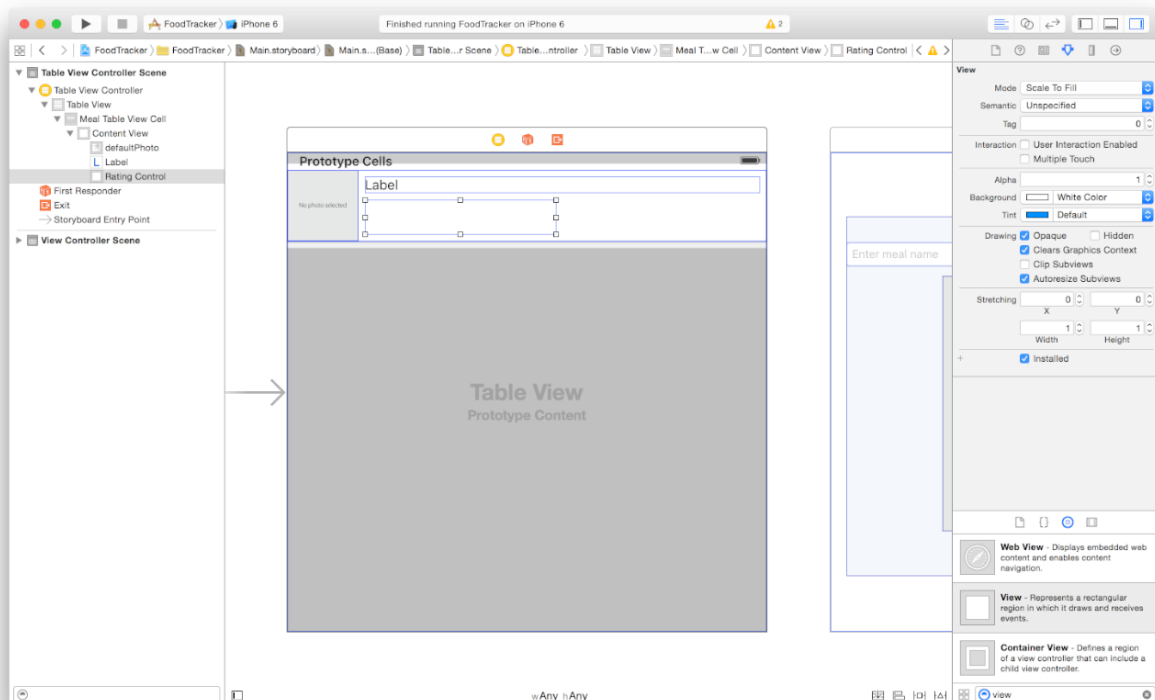
در صورتی که کلاس RatingControl به عنوان آیتم قابل گزینش در منوی pop-up در دسترس نبود، آنگاه بایستی به canvas مراجعه نموده و المان مربوطه را انتخاب نمایید (المانی که در تصویر قبلی، دستگیره های تنظیم اندازه بر روی لبه های آن نمایان می باشد).

۱۶. در حالی که view را انتخاب کرده، کادر Attribute inspector را باز نمایید.

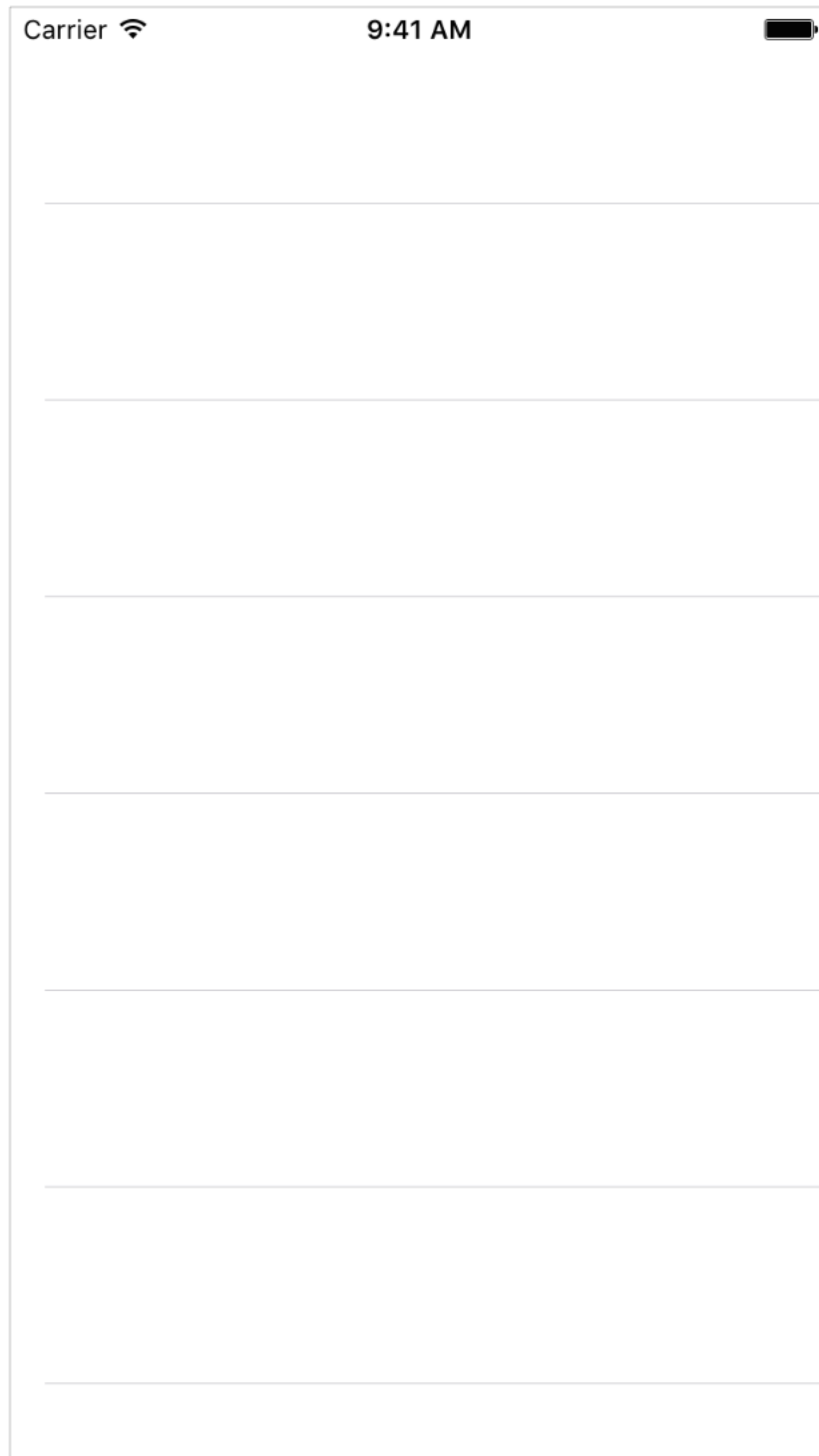
۱۷. داخل کادر Attribute inspector، فیلدی که Interaction نام دارد را یافته و تیک چک باکس User Interaction را بردارید.

اگر بخاطر داشته باشید کلاس rating control اختصاصی را طوری تنظیم کردید که قابلیت تعامل با کاربر را داشته باشد. اما زمانی که این rating control (کنترل امتیازدهی) را در خانه ی جدول به نمایش می گذارید، دیگر نیازی به تعامل با کاربر ندارد. لازم است قابلیت تعامل با کاربر را از این المان، در بستر جاری (داخل خانه ی جدول)، سلب نمایید.

در حال حاضر ال اپلیکیشن شما بایستی ظاهری مشابه زیر داشته باشد:



**تست کنید:** برنامه ی خود را اجرا نمایید. دو خط بالا و پایین سطرهای (خانه ها) table view شما بایستی با فاصله ی بیشتری از هم نمایش داده شوند (ارتفاع هر خانه می بایست افزایش یافته باشد). اگر خوب دقت کنید، متوجه می شوید که اگر چه تمامی المان های مورد نیاز ال را اضافه کرده اید، با این وجود سطرهای جدول، درست مانند قبل، همگی خالی از محتوا نمایش داده می شوند. علت آن چیست؟



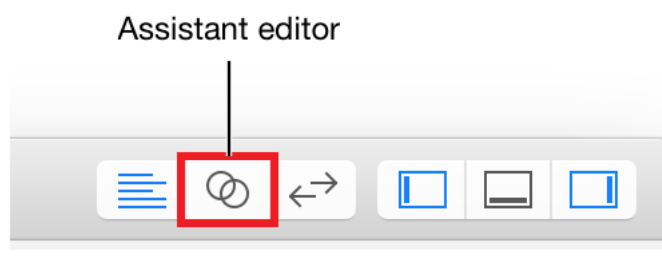
در storyboard، می‌توان یک table view را طوری تنظیم کرد که داده‌ها را به صورت static نمایش دهد (داده‌هایی که از داخل رابط کاربری storyboard به جدول اضافه می‌شود را به کاربر نشان دهد) یا آن‌ها را به صورت داینامیک از data model خوانده و در زمان اجرا به نمایش بگذارد (داده‌ها را از منطق table view controller بگیرد).

Table view به صورت پیش فرض داده های داینامیک را نمایش می دهد و از آنجایی که باید اطلاعات را از طریق کد در حافظه بارگذاری کنید، دقیقاً همین کار را هم از همان ذکر شده انتظار دارید – اما یک جای کار مشکل دارد: رفتار دلخواه برای این منظور را هنوز پیاده سازی نکرده اید. به عبارت دیگر داده های static که در storyboard به جدول اضافه کردید، در زمان اجرای اپلیکیشن نمایش داده نمی شود – مگر اینکه data model که داده های اپلیکیشن را کپسوله سازی کرده و در خود نگه می دارد را پیاده سازی نمایید.

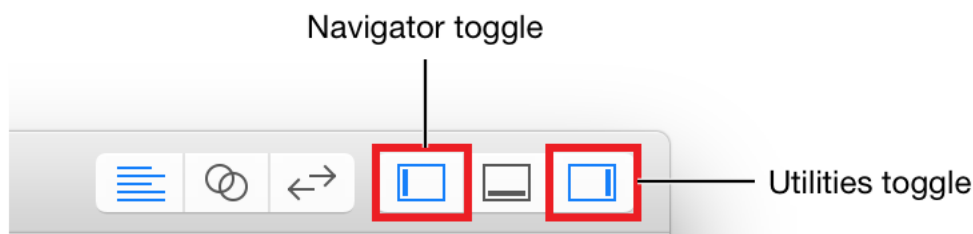
فعلاً، با استفاده از assistant editor (ویرایشگر کمکی محیط کاری Xcode)، نمای موقتی UI خود را مشاهده نمایید.

جهت پیش مشاهده ی UI اپلیکیشن، مراحل زیر را دنبال نمایید:

۱. با کلیک بر روی دکمه ی Assistant در نوار ابزار Xcode، ویرایشگر کمکی محیط را باز کنید.

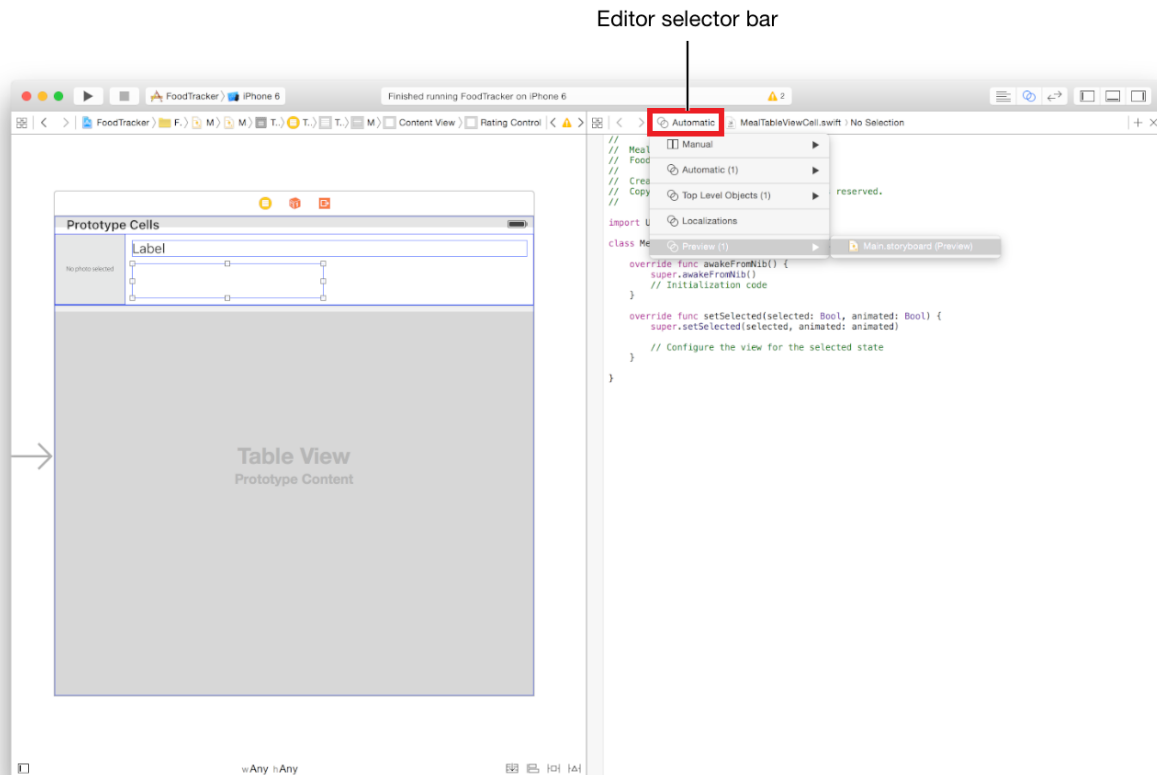


۲. اگر به فضای کاری بیشتری نیاز دارید، می توانید project navigator و utility area را با کلیک بر روی دکمه های مربوطه ی آن در نوار ابزار محیط Xcode ببندید.

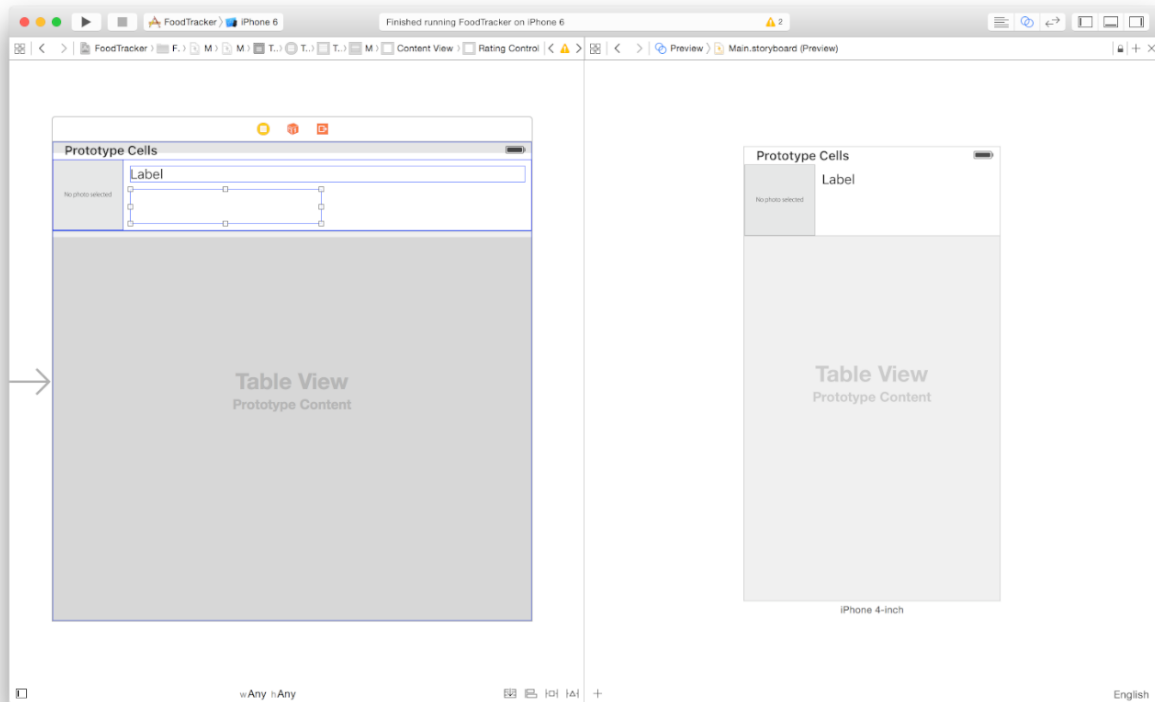


در صورت نیاز به فضای بیشتر، می توانید کادر outline view را نیز ببندید.

در **editor selector bar** که در بالای ویرایشگر کمکی محیط نمایش داده می شود، با انجام این عملیات: **Automatic to Preview > Main.storyboard** (Preview)، ویرایشگر کمکی را از حالت **Automatic** بر روی **Preview** تنظیم نمایید.



پنجره ی محیط توسعه ی Xcode هم اکنون می بایست مشابه زیر باشد:



همان طور که می بینید، نمای preview ویرایشگر کمکی، UI را دقیقاً به صورت مورد انتظار نمایش می دهد. پیش نمونه ی خانه جدول/المان table view در UI اکنون دیگر کامل به نظر می رسد.

نکته: اگر صفحه محتوا/scene نامربوط را در حالت UI preview (نمای موقتی) مشاهده نمودید، در آن صورت می توانید با کلیک بر روی scene dock مربوطه ی آن، صفحه محتوای نمایش جدول (table view scene حاوی المان table view) را انتخاب نمایید.

### افزودن عکس به پروژه

در گام بعدی، بایستی تعدادی عکس ساده به پروژه ی خود اضافه نمایید. این عکس ها را دقیقاً زمانی که داده های اولیه ی meal را در اپلیکیشن بارگذاری می کنید، مورد استفاده قرار خواهید داد.

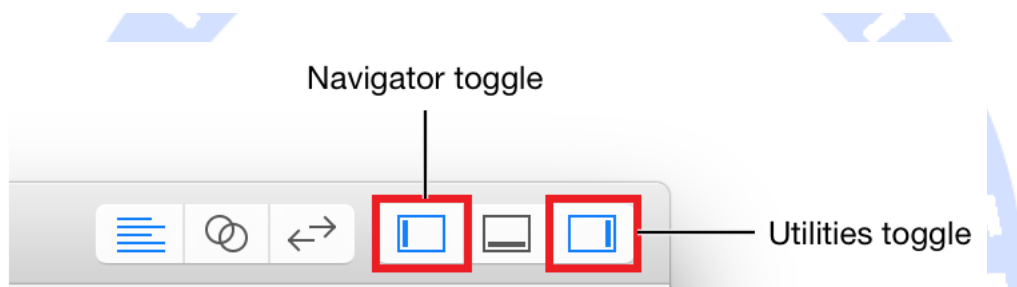
می توانید برای عکس به پوشه ی Images/ که پس از دانلود فایل پروژه در اختیار شما قرار می گیرد، مراجعه نمایید. یا از عکس های دلخواه خود استفاده کنید (فقط به یاد

داشته باشید که اسم عکس های مورد استفاده باید با اسم عکس ها در کد همخوانی داشته باشد).

جهت افزودن عکس به پروژه، مراحل زیر را دنبال نمایید:

۱. در صورت باز بودن ویرایشگر کمکی محیط، با کلیک بر روی دکمه ی Standard در نوار ابزار Xcode، ویرایشگر اصلی محیط را باز کنید.

حال project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار Xcode، باز نمایید.



۲. در project navigator، با کلیک بر روی فایل Assets.xcassets، ابزار asset catalog را جهت مشاهده ی محتوای آن باز نمایید.

اگر بخاطر داشته باشید، asset catalog بستری است که در آن فایل های محتوای برنامه خود نظیر عکس را ذخیره یا مدیریت می کنید.

۳. در پایین، گوشه ی سمت چپ، بر روی علامت (+) کلیک نمایید و سپس New Folder را از منوی pop-up که نمایان می شود، انتخاب نمایید.

۴. بر روی اسم پوشه دوبار کلیک کرده و سپس آن را به Sample Images تغییر دهید.

۵. پس از انتخاب پوشه، در پایین، گوشه ی سمت چپ، بر روی علامت (+) کلیک نمایید و سپس New Image Set را از منوی pop-up گزینش نمایید.

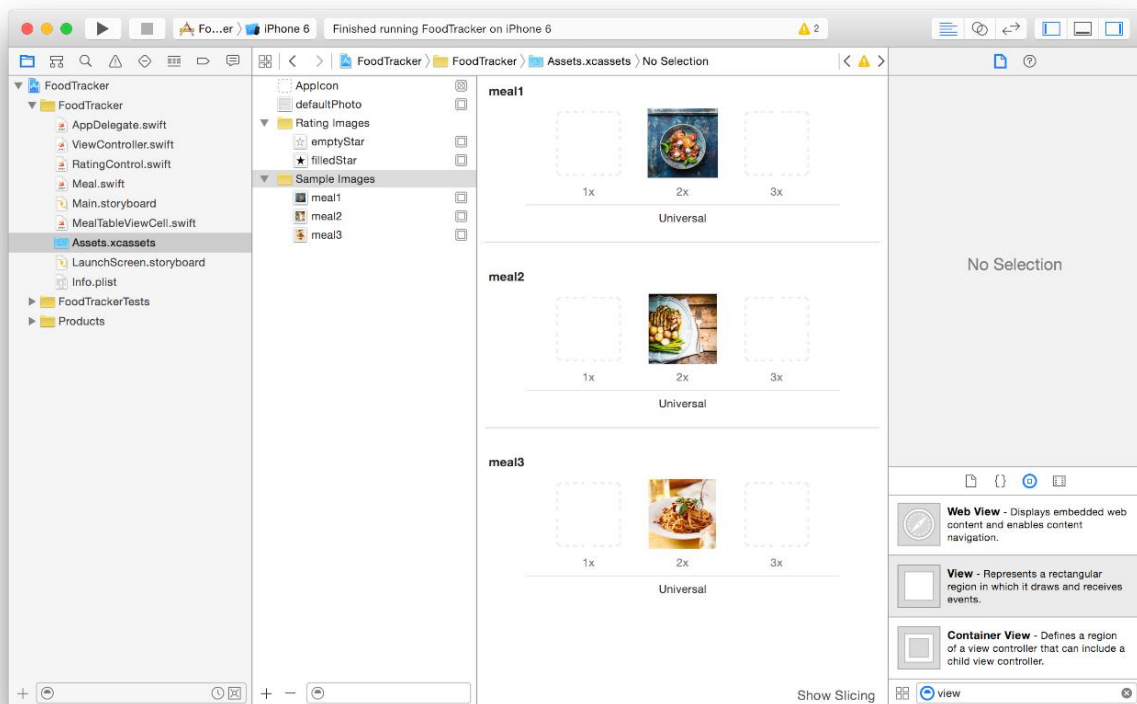
۶. بر روی اسم image set دوبار کلیک نمایید. اسم image set را طوری انتخاب کنید که به هنگام ذکر آن در کدنویسی با مشکل مواجه نشوید.

۷. عکس مورد نظر را از حافظه ی کامپیوتر انتخاب نمایید.

۸. عکس را با اشاره گر موس کشیده و آن را در جایگاه خالی 2x، داخل image set

جایگذاری نمایید.

مراحل ۵ تا ۸ را برای افزودن هر تعداد عکس که لازم دارید، تکرار نمایید. برای این مبحث سه عکس مختلف از کامپیوتر خود انتخاب نموده و به پروژه اضافه کنید.



ایجاد اتصال بین ال و کد به وسیله ی outlet (وصل کردن ال نمای جدولی به کد)

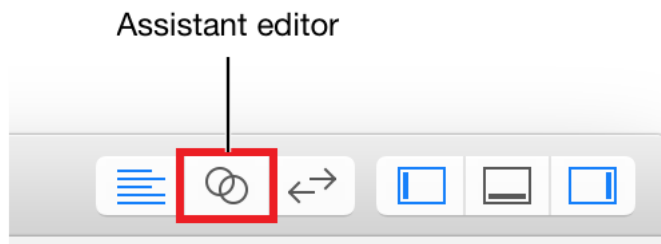
برای اینکه بتوانید داده های خود را داخل خانه های جدول در زمان اجرای اپلیکیشن به صورت داینامیک نمایش دهید، بایستی با استفاده از outlet بین view ها در storyboard و کد کلاس table view cell در فایل MealTableViewCell.swift اتصال ایجاد نمایید.

جهت متصل کردن view ها به کد موجود در فایل MealTableViewCell.swift، مراحل زیر دنبال نمایید:

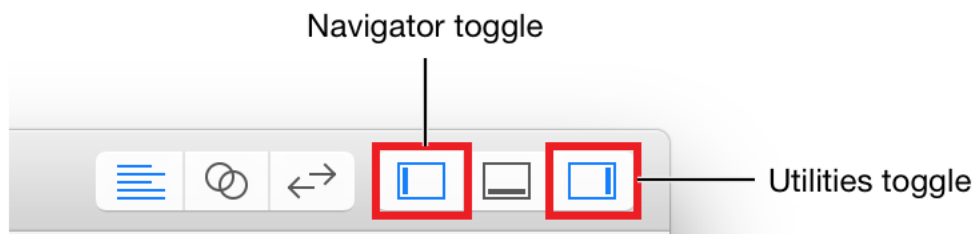
۱. داخل storyboard، آبجکت label را در table view cell (خانه جدول) انتخاب نمایید.



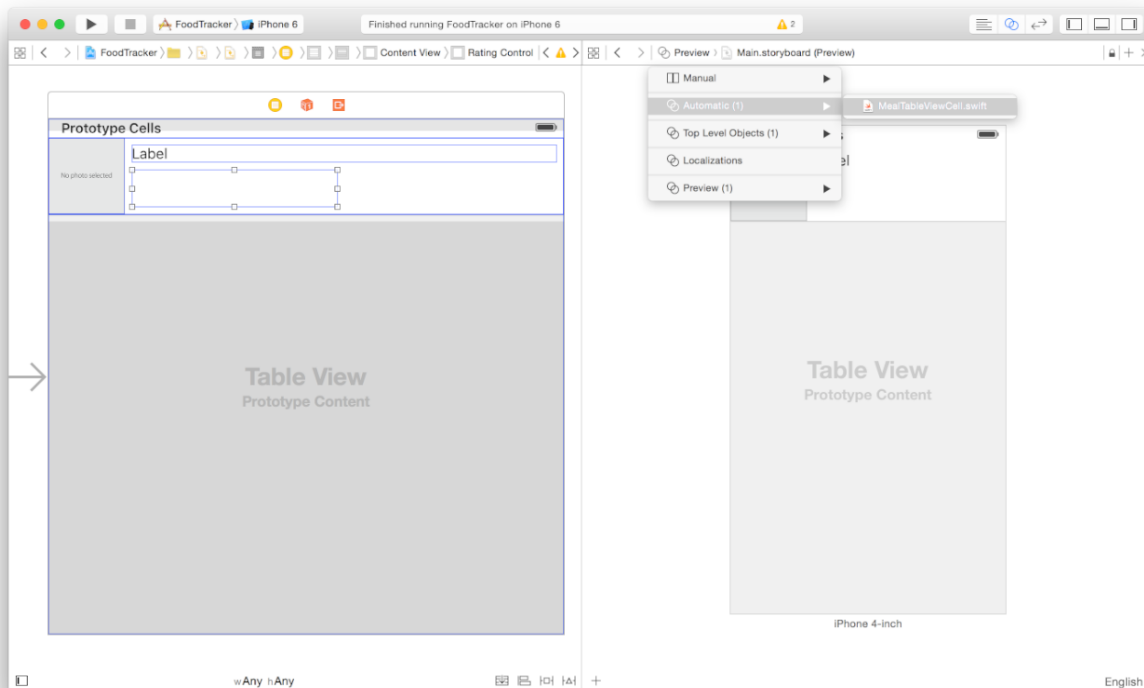
۲. بر روی دکمه ی Assistant در نوار ابزار Xcode کلیک نموده و ویرایشگر کمکی محیط را باز نمایید.



۳. در صورت نیاز به فضای کاری بیشتر، project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار محیط کاری Xcode پنهان نمایید.



۴. در editor selector bar (نوار ابزاری که در بالای ویرایشگر کمکی محیط به نمایش در می آید)، حالت assistant editor را از Preview به Automatic تغییر دهید (Preview > Automatic > MealTableViewCell.swift).



محتوای فایل MealTableViewCell.swift در ویرایشگر کمکی، سمت راست محیط کاری Xcode به نمایش در می آید.

۵. داخل کدهای فایل MealTableViewCell.swift، خط تعریف کلاس را پیدا کنید:

```
class MealTableViewCell: UITableViewCell {
```

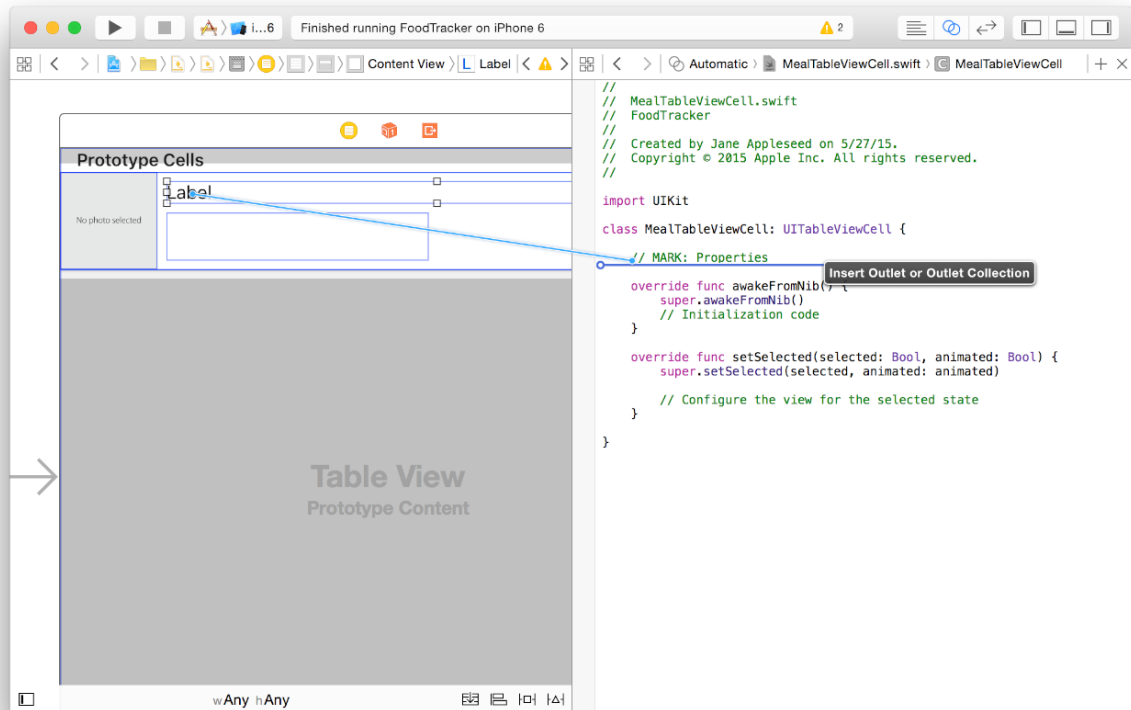
۶. در زیر خط تعریف کلاس، یک comment به صورت زیر درج نمایید:

```
// MARK: Properties
```

۷. بر روی آبجکت label در canvas کلیک کنید. سپس آن را به ناحیه ی ویرایش کد در سمت

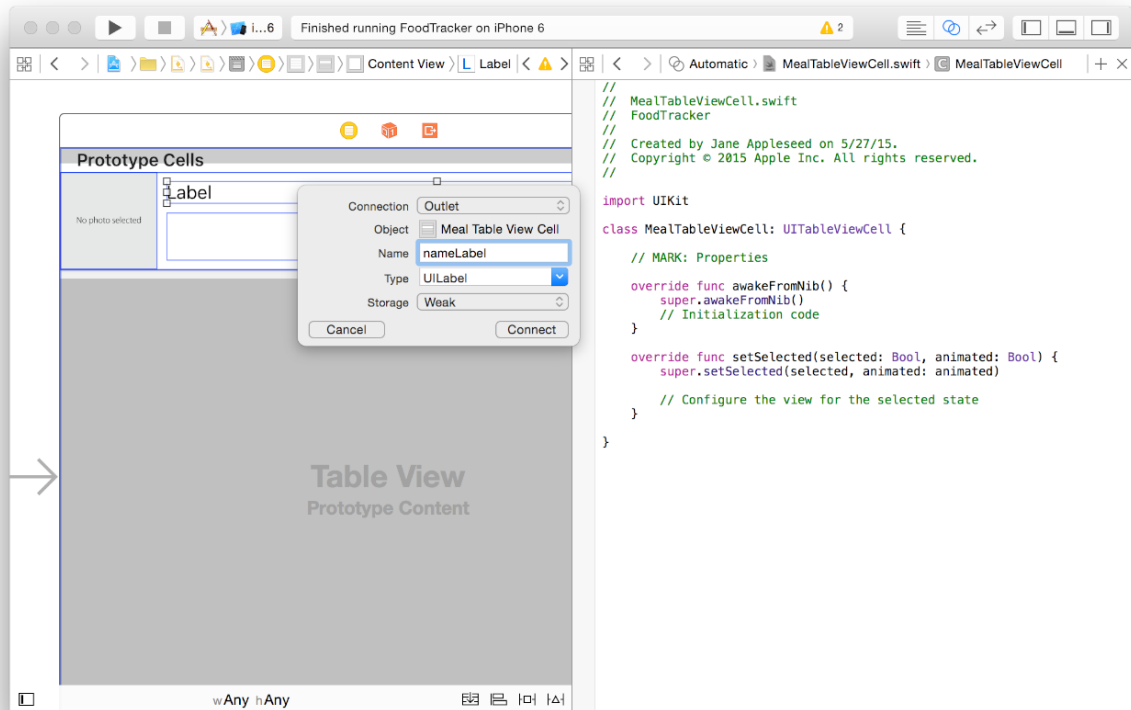
راست محیط کشیده و در زیر خط comment درج شده، داخل فایل

MealTableViewCell.swift جایگذاری نمایید.



۸. در کادر محاوره ای که نمایان می شود، داخل فیلد Name واژه ی nameLabel را به عنوان اسم outlet وارد نمایید.

به تنظیمات دیگری نیاز نیست. هم اکنون کادر محاوره ای اتصال از ال به کد، بایستی به صورت زیر مقداردهی شده باشد:



۹. دکمه ی Connect را کلیک کنید.

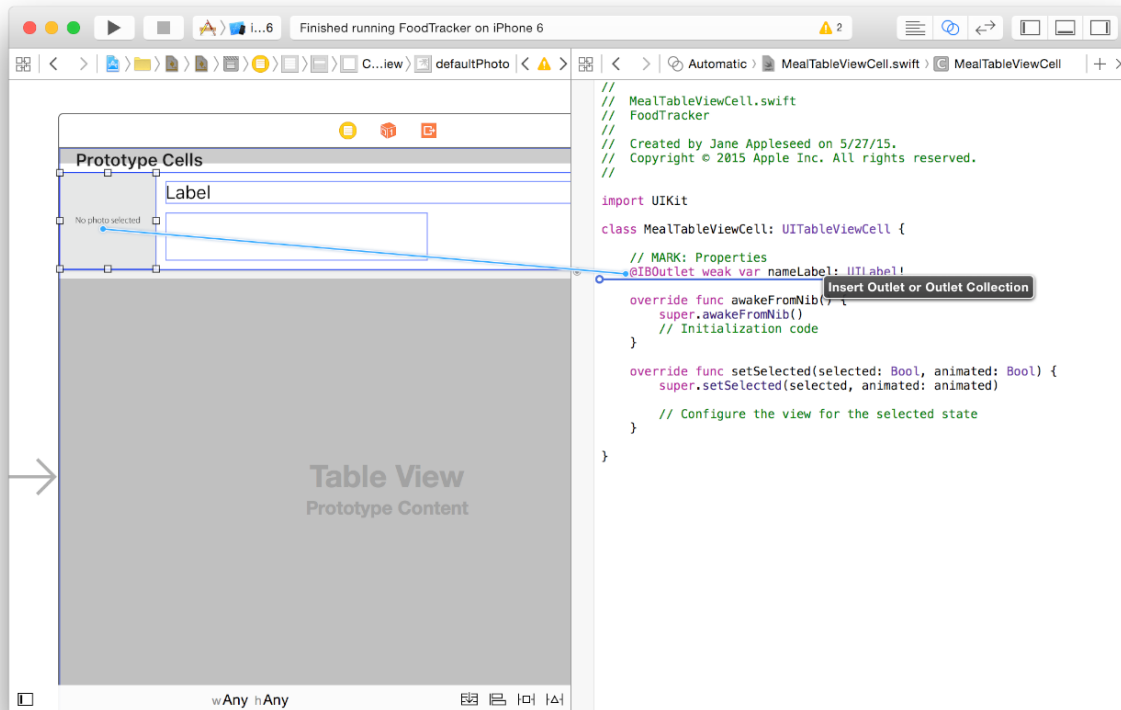
۱۰. در storyboard، بر روی المان image view داخل خانه ی جدول (table view cell)

کلیک نموده و آن را انتخاب کنید.

۱۱. حال المان image view را از سطح canvas با اشاره گر موس کشیده و داخل ناحیه ی

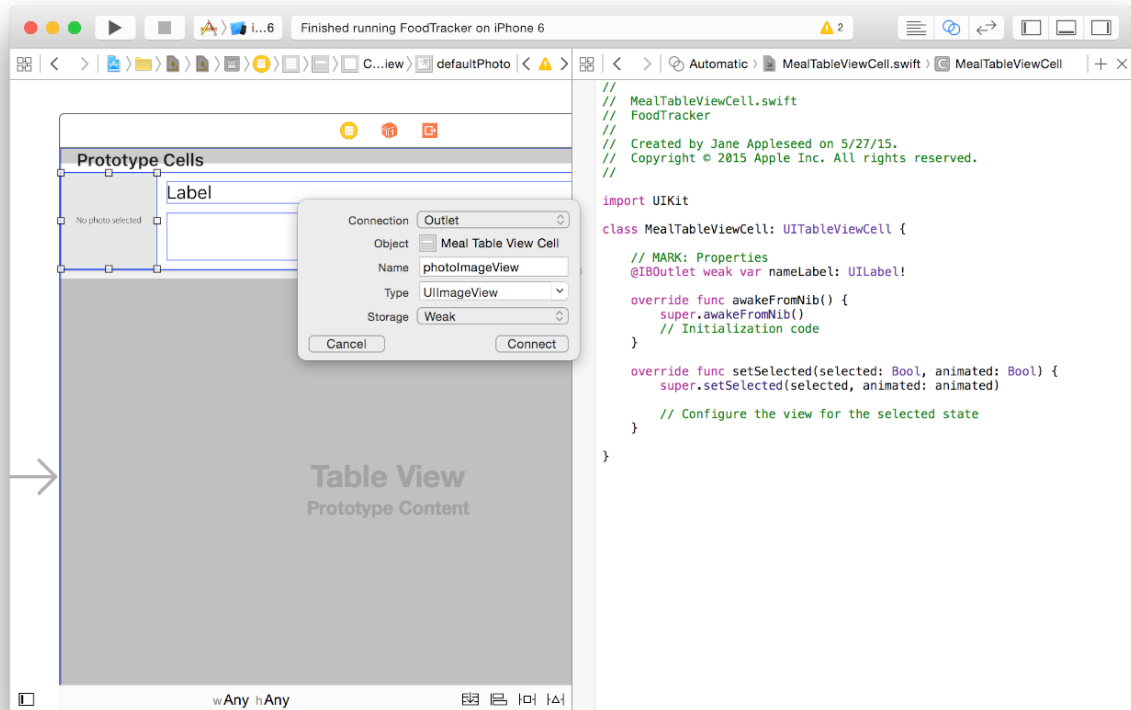
ویرایش کد، سمت راست محیط در زیر متغیر (property) nameLabel (در فایل

MealTableViewCell.swift) جایگذاری نمایید.

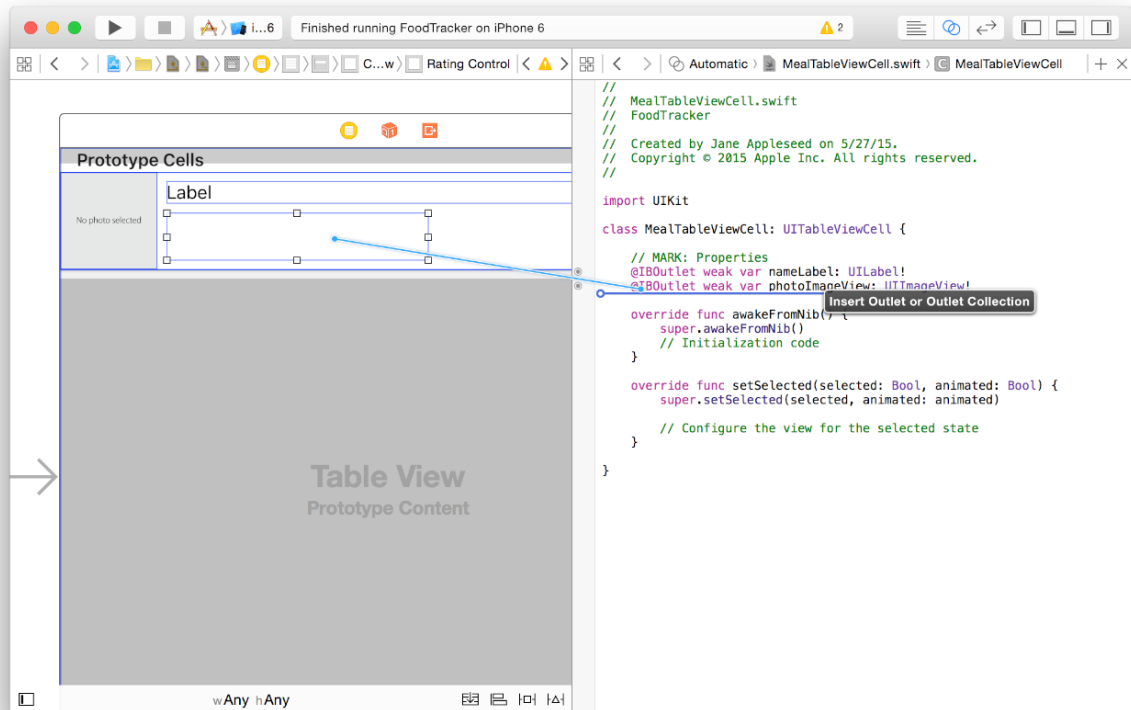


۱۲. در کادر محاوره ای که نمایان می شود، داخل فیلد Name، واژه ی photoImageView را به عنوان اسم جدید وارد نمایید.

لازم به انجام تنظیمات دیگری نیست. کافی است بر روی دکمه ی Connect جهت اتصال برقرار کردن بین UI و کد کلیک نمایید.

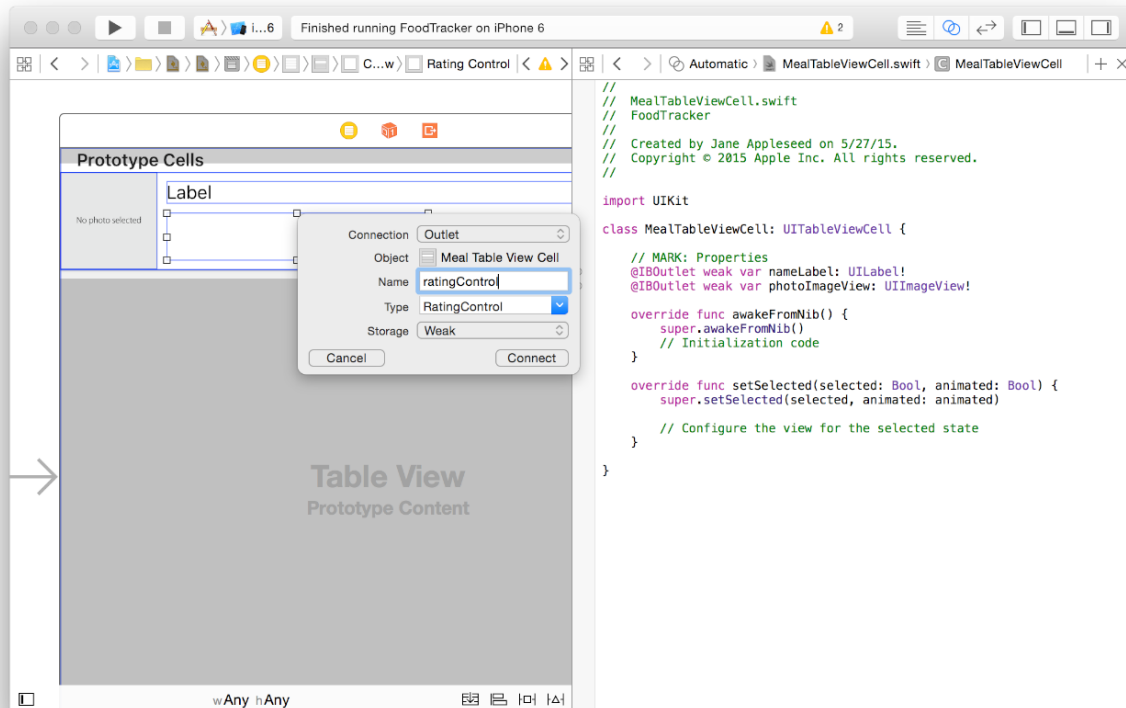


۱۳. این بار در storyboard، بر روی کنترل امتیازدهی (rating control) داخل خانه ی جدول (table view cell) کلیک نمایید.
۱۴. کنترل نام برده را از سطح canvas با موس کشیده و در ناحیه ی ویرایش کد در سمت راست محیط، زیر متغیر (property) photoImageView جایگذاری نمایید (در فایل MealTableViewCell.swift).



۱۵. در کادر محاوره ای اتصال از UI به کد، داخل فیلد Name، واژه ی ratingControl را وارد نمایید.

۱۶. به تنظیمات دیگر احتیاجی نیست. کافی است بر روی دکمه ی Connect جهت برقراری اتصال کلیک نمایید.



در حال حاضر متغیرهایی که در قالب outlet داخل فایل MealTableViewCell.swift تعریف کردید، می بایست به ترتیب زیر باشد:

```
@IBOutlet weak var nameLabel: UILabel!
@IBOutlet weak var photoImageView: UIImageView!
@IBOutlet weak var ratingControl: RatingControl!
```

بارگذاری داده های اولیه ی اپلیکیشن

جهت نمایش داده های مورد نظر در خانه ی های جدول (table view)، لازم است کدی بنویسید که این داده ها را بارگذاری کند. اگر به خاطر داشته باشید در بحث قبلی یک data model برای اپلیکیشن خود طراحی کردید که داده های برنامه را موقتاً ذخیره کرده و در اختیار آن قرار می دهد؛ اسم این data model، کلاس Meal بود. همچنین شما برای برنامه ی کاربردی خود به یک لیست احتیاج دارید که اطلاعات مربوط به غذاها را در آن نگه دارید. ناگفته پیداست که مکان مشخص برای نگهداری و رصد این اطلاعات یک



کلاس فرزند view controller با پیاده سازی اختصاصی ( custom view controller subclass ) است که به meal list ( لیست نگه داری غذاها ) متصل می باشد.

کلاس view controller ای که از آن سخن به میان آمد، وظیفه ی مدیریت view ای را بر عهده دارد که لیست غذاها را برای کاربر در ال به نمایش می گذارد و علاوه بر آن اشاره گری (reference) به data model دارد که اطلاعات مورد نیاز ال را فراهم می کند.

ابتدا می بایست یک کلاس فرزند از کلاس table view controller با پیاده سازی اختصاصی خود ایجاد نمایید که مدیریت صفحه محتوایی که لیست غذاها را نمایش می دهد ( meal list scene ) بر عهده دارد.

به منظور ایجاد یک subclass از کلاس UITableViewController، مراحل زیر را دنبال نمایید:

۱. این مسیر را طی نمایید: File > New > File یا کلیدهای Command-N را همزمان فشار دهید.

۲. کادر محاوره ای نمایان می شود. در سمت چپ این کادر محاوره ای، گزینه ی Source را از زیر iOS انتخاب نموده، سپس Cocoa Touch Class را انتخاب کنید.

۳. حال دکمه ی Next را کلیک نمایید.

۴. در فیلد Class، واژه ی Meal را وارد نمایید.

۵. از فیلد "Subclass of field"، گزینه ی UITableViewController را انتخاب نمایید.

عنوان کلاس به MealTableViewController تغییر می کند. لازم نیست در آن تغییری ایجاد کنید.

۶. گزینه ی "Also create XIB file" را از حالت انتخاب خارج نمایید.

۷. زبان پروژه را بر روی Swift تنظیم نمایید (گزینه ی Language می بایست بر روی Swift تنظیم باشد).

۸. بر روی دکمه ی Next کلیک نمایید.

محل ذخیره سازی (save location) فایل ها به صورت پیش فرض بر روی پوشه

ی پروژه (project directory) تنظیم می شود.

گزینه ی Group به صورت پیش فرض بر روی اسم اپلیکیشن، FoodTracker،

تنظیم می شود.

در بخش Targets، کادر تیک برای اپلیکیشن شما باید انتخاب شده باشد و کادر

تیک دوم که مربوط به unit test برای برنامه ی شما است باید غیرفعال باشد.

۹. گزینه هایی که با مقادیر پیش فرض تنظیم شده اند را تغییر ندهید. اکنون دکمه

ی Create را فشار دهید.

محیط Xcode فایل MealTableViewCell.swift را ایجاد می کند. این فایل

دربدارنده ی کدی است که کلاس فرزند از UITableViewController با پیاده سازی

اختصاصی را تعریف می کند.

در این subclass که دارای پیاده سازی اختصاصی شما است، می توانید یک متغیر

(property) تعریف کنید که لیستی از آبجکت های Meal را در خود نگه می دارد.

کتابخانه ی متعارف Swift standard library/Swift دربردارنده ی ساختاری به نام

Array هست که ویژه ی ذخیره ی لیستی مرتب از مقادیر تعبیه شده.

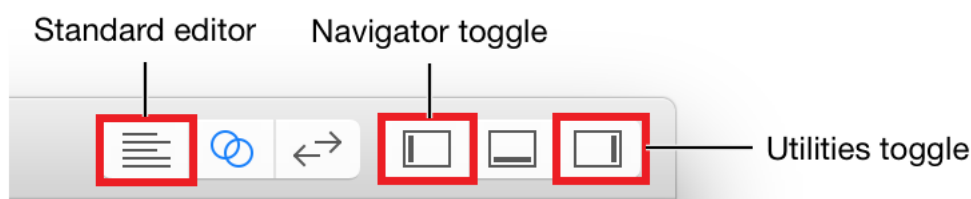
برای بارگذاری داده های اولیه ی اپلیکیشن، مراحل زیر را دنبال نمایید:

۱. در صورت باز بودن ویرایشگر کمکی محیط (assistant editor)، با کلیک بر روی

دکمه ی Standard، ویرایشگر اصلی Xcode را باز نمایید.

Project navigator و utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار محیط

کاری Xcode، باز نمایید.



۲. فایل MealTableViewController.swift را باز نمایید.

۳. در زیر خط تعریف کلاس داخل فایل مذکور، کد زیر را وارد نمایید:

```
// MARK: Properties
```

```
var meals = [Meal]()
```

توسط این کد یک متغیر (property) به نام meals از نوع آرایه تعریف کرده و آن را با یک مقدار پیش فرض (آرایه ی تهی از آبجکت های meal) مقدار دهی اولیه می کنید. همان طور که می بینید property بجای let (به صورت ثابت) با کلیدواژه ی var و از نوع متغیر تعریف شده چرا که با استفاده از کلیدواژه ی var، در واقع شما آرایه ی مورد نظر را به صورت mutable و تغییر پذیر تعریف می کنید. بدین معنی که بعده ها (پس از مقداردهی اولیه) می توانید آیتم هایی به آن اضافه کرده یا از آن حذف نمایید.

۴. در فایل MealTableViewController.swift، پس از تعریف متد

viewDidLoad()، تابع زیر را اضافه نمایید:

```
func loadSampleMeals() {  
}
```

این متد در حقیقت یک تابع کمکی است که داده های نمونه را در اپلیکیشن بارگذاری می کند.

۵. در بدنه ی متد loadSampleMeals()، با اضافه کردن این کد تعدادی آبجکت

Meal ایجاد نمایید. می توانید این غذاهای نمونه را خود هر طور که مایلید نام

گذاری و امتیازدهی نمایید. در زیر نمونه هایی ارائه شده که در صورت تمایل می

توانید از آن ها استفاده کنید.

```
let photo1 = UIImage(named: "meal1")!
```

```
let meal1 = Meal(name: "Caprese Salad", photo: photo1, rating: 4)!
```

```
let photo2 = UIImage(named: "meal2")!
```

```
let meal2 = Meal(name: "Chicken and Potatoes", photo: photo2, rating: 5)!
```

```
let photo3 = UIImage(named: "meal3")!
```

```
let meal3 = Meal(name: "Pasta with Meatballs", photo: photo3, rating: 3)!
```

دقت داشته باشید که اسم تصاویر در پروژه بایستی با اسمی که در کد می نویسید همخوانی داشته باشد.

۶. پس از ایجاد آبجکت های Meal، آن ها را با استفاده از این کد به آرایه ی meals اضافه نمایید:

```
meals += [meal1, meal2, meal3]
```

متد viewDidLoad() را پیدا کنید. پیاده سازی که به عنوان الگو و آماده برای این متد ارائه می شود (template implementation)، به صورت زیر می باشد:

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Uncomment the following line to preserve selection between presentations خط
    // زیر را از حالت کامنت خارج نموده تا انتخاب بین نمایش ها ثابت بماند.
    // self.clearsSelectionOnViewWillAppear = false
    // Uncomment the following line to display an Edit button in the navigation bar for
    // this view controller. این کد را از حالت کامنت خارج نموده تا یک دکمه ی ویرایش در
    // نوارپیمایش به نمایش در آید.
    // self.navigationItem.rightBarButtonItem = self.editButtonItem()
}
```

محیط توسعه ی Xcode به هنگام ایجاد فایل MealTableViewController.swift، پیاده سازی هایی را به صورت آماده و در قالب comment همراه این متد ارائه می دهد. کدهایی که به صورت comment مانند نمونه ی فوق در اختیار شما قرار می گیرد، اطلاعات و سرخ های بسیار مفیدی مرتبط با شرایط جاری برای شما فراهم می کند. برای مبحث فعلی نیازی به این پیاده سازی ها نیست، می توانید آن ها را حذف نمایید.

۷. Comment های موجود در بدنه ی متد viewDidLoad() را حذف نموده و بجای آن، کد زیر را پس از خط فراخوانی متد super.viewDidLoad()، جهت بارگذاری داده های نمونه درج نمایید:

// Load the sample data.

loadSampleMeals()

این کد، متد کمکی (helper method) را که شما جهت بارگذاری داده های نمونه اعلان نمودید، همزمان با بارگذاری view فراخوانی می کند. این متد را به صورت مجزا و در کد مختص به خود قرار دادید تا بدین وسیله متن برنامه (source code) خوانا تر شده و اپلیکیشن شما دارای طراحی ماژولار باشد.

متد viewDidLoad() هم اکنون می بایست دارای پیاده سازی زیر باشد:

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Load the sample data.
    loadSampleMeals()
}
```

و بدنه ی متد loadSampleMeals() شما باید مشابه زیر باشد:

```
func loadSampleMeals() {
    let photo1 = UIImage(named: "meal1")!
    let meal1 = Meal(name: "Caprese Salad", photo: photo1, rating: 4)!
    let photo2 = UIImage(named: "meal2")!
    let meal2 = Meal(name: "Chicken and Potatoes", photo: photo2, rating: 5)!
    let photo3 = UIImage(named: "meal3")!
    let meal3 = Meal(name: "Pasta with Meatballs", photo: photo3, rating: 3)!
    meals += [meal1, meal2, meal3]
}
```

**تست کنید:** اکنون پروژه ی خود را با طی کردن مسیر رو به رو کامپایل نمایید: Product > Build. پروژه بایستی بدون خطا کامپایل شود.

**نکته:** در صورت برخورد با خطاهای زمان کامپایل پروژه، ابتدا باید اسم تصاویر در پروژه را با اسم تصاویر در کد مطابقت داده و از همخوانی آن ها اطمینان حاصل نمایید.

### به نمایش گذاشتن داده ها در UI

در حال حاضر، کلاس ارث بری شده از کلاس UITableViewController که پیاده سازی آن را خود به صورت اختصاصی انجام دادید، دارای یک آرایه ی تغییر پذیر (mutable

array) است که با داده های نمونه پر شده. اکنون زمان آن رسیده که این داده های نمونه را در رابط کاربری برنامه به نمایش بگذارید.

برای اینکه یک table view بتواند داده ها را به صورت dynamic و در زمان اجرای اپلیکیشن نمایش دهد، به همکاری دو مولفه رو به رو احتیاج دارد: ۱. Data source. ۲. Delegate. همان طور که از اسم آن پیداست، منبعی است که table view داده های خود را از آن می گیرد و نمایش می دهد. Delegate به table view کمک می کند مواردی همچون انتخاب خانه ی جدول، تنظیم فاصله بین دو خط یک سطر (row height) و دیگر جنبه های مربوط به نمایش داده های اپلیکیشن را مدیریت کند. به صورت پیش فرض، کلاس UITableViewController و کلاس های مشتق شده از آن (subclass های آن)، protocol های مورد نیاز را پیاده سازی کرده (این protocol ها را به خط تعریف کلاس اضافه کرده) و این کلاس را همزمان به یک source (UITableViewDataSource) و data (UITableViewDelegate) delegate برای table view مربوطه تبدیل می کند. وظیفه ی شما این است که متدهای مورد نیاز را از پروتکل های نام برده پیاده سازی کنید تا کلاس فرزند table view شما رفتار مورد نظر را از خود نشان دهد.

یک table view برای کارکرد صحیح خود در کل به پیاده سازی سه متد از پروتکل UITableViewDataSource نیاز دارد. این متدها به شرح زیر می باشند:

```
func numberOfSectionsInTableView(tableView: UITableView) -> Int
func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell
```

اولین متدی که باید پیاده سازی شود، numberOfSectionsInTableView( ) است که به table view اعلان می کند که چند بخش (section) را باید نمایش دهد. بخش یا

section ها گروه هایی از خانه های قابل مشاهده در table view هستند که برای منظور نمایش انبوه داده بسیار مناسب می باشند. برای table view ساده ی اپلیکیشن FoodTracker نمایش یک بخش کفایت می کند. از اینرو پیاده سازی متد UITableViewDataSource(numberOfSectionsInTableView(\_):) از پروتکل ————— آسان خواهد بود.

به منظور نمایش یک بخش (section) در table view مراحل زیر را گام به گام دنبال نمایید:

۱. داخل فایل MealTableViewController.swift، متد —————

numberOfSectionsInTableView(\_:) را پیدا کنید. پیاده سازی که به صورت آماده توسط Xcode ارائه می شود، به صورت زیر می باشد:

```
override func numberOfSectionsInTableView(tableView: UITableView) -> Int {
// #warning Incomplete implementation, return the number of sections
return 0
}
```

۲. مقدار بازگشتی را از ۰ به ۱ تغییر داده، سپس comment را حذف نمایید.

```
override func numberOfSectionsInTableView(tableView: UITableView) -> Int {
return 1
}
```

این کد به table view اعلان می دارد که باید بجای ۰، یک بخش (section) نمایش دهد. Comment را به این خاطر حذف کردید که پیاده سازی متد را با تغییراتی که در بالا اعمال کردید، تکمیل نموده و دیگر نیازی به comment نیست.

متد بعدی که از پروتکل UITableViewDataSource پیاده سازی خواهید کرد، تابع tableView(\_:numberOfRowsInSection:) است. این تابع به table view اعلان می دارد که در هر بخش می بایست چند سطر نمایش دهد.

Table view در حالت پیش فرض تنها یک بخش نمایش می دهد. برای اپلیکیشن FoodTracker نیز دقیقاً به یک بخش نیاز دارید. هر غذا باید داخل بخش مورد نظر، در

سطر مجزا نمایش داده شود. به عبارت دیگر، تعداد سطرها باید با تعداد آبجکت های Meal در آرایه ی meals برابر باشد.

به منظور بازگردانی و مشاهده ی تعداد سطرهای موجود در table view، مراحل زیر را دنبال نمایید:

۱. داخل فایل MealTableViewController.swift، متد

tableView(\_:numberOfRowsInSection:) را پیدا کنید. پیاده سازی که به صورت آماده ارائه می شود، به صورت زیر می باشد:

```
override func tableView(tableView: UITableView, numberOfRowsInSectionSection: Int) -> Int {
    // #warning Incomplete implementation, return the number of rows
    return 0
}
```

می خواهید تعداد meal های موجود در آرایه را در خروجی برگردانید. نوع داده ای Array یک property به نام count دارد که تعداد آیتم های موجود در آرایه را به عنوان خروجی برمی گرداند. بنابراین تعداد سطرهای موجود در آرایه meals را با درج دستور meals.count می توانید در خروجی داشته باشید.

۲. متد tableView(\_:numberOfRowsInSection:) را طوری ویرایش کنید که تعداد صحیح سطرها را در خروجی برگرداند. همچنین comment ای که همراه این متد ارائه می شود را حذف نمایید.

```
override func tableView(tableView: UITableView, numberOfRowsInSectionSection: Int) -> Int {
    return meals.count
}
```

آخرین متد از پروتکل UITableViewDataSource که باید پیاده سازی کنید، تابع tableView(\_:cellForRowAtIndexPath:) است که یک خانه از جدول را پیکربندی کرده و در سطر مورد نظر نمایش می دهد. به ازای هر سطر در table view یک خانه وجود دارد و آن خانه محتوایی که در سطر جدول نمایش داده شده



و نحوه ی چیدمان آن را تعیین می کند. در یک table view که تعداد اندکی سطر دارد، تمامی سطرها احتمالاً یکجا در صفحه نمایش داده می شوند، از اینرو این متد به ازای هر سطر در جدول فراخوانی می گردد. اما در جداولی که تعدادی زیادی سطر دارند، قضیه کمی فرق می کند. به عبارت دیگر جداول بزرگ با سطرهای متعدد در یک برهه ی زمانی مشخص تنها تعداد محدودی از کل آیتم های خود را در صفحه به نمایش می گذارند. در چنین مواردی بهتر است table view درخواست نمایش آن خانه هایی از جدول را بدهد که سطرهای آن در حال نمایش هستند. جالب است بدانید که متد tableView(\_:cellForRowAtIndexPath:) را برای شما فراهم می آورد.

خانه ی هر سطر جدول (table view) را با واکنشی المان Meal متناظر (مربوطه) در آرایه ی meals و سپس برابر قرار دادن property های خانه ی جدول با مقادیر از کلاس Meal تنظیم و پیکربندی می کنید. جهت تنظیم و نمایش خانه های جدول در table view، مراحل زیر را به ترتیب دنبال نمایید:

۱. داخل فایل MealTableViewCell.swift، متد

tableView(\_:cellForRowAtIndexPath:) را پیاده کرده و از حالت comment خارج نمایید. برای خارج نمودن متد از حالت comment، کافی است `/*` و `*/` را از اطراف آن حذف نمایید. پس از حذف دو کاراکتر مزبور، پیاده سازی الگو (template Implementation) متد به صورت زیر خواهد بود:

```
override func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {
let cell = tableView.dequeueReusableCellWithIdentifier("reuseIdentifier",
forIndexPath: indexPath)
// Configure the cell...
return cell
}
```

پیاده سازی الگو (template implementation) وظایف متعددی را انجام می دهد که از جمله ی آن می توان به موارد زیر اشاره کرد: از table view می خواهد آن خانه ای که دارای یک placeholder identifier هست را در اختیارش قرار دهد، یک comment در خصوص اینکه کد تنظیم کننده ی خانه ی جدول کجا باید قرار گیرد، اضافه نموده و سپس خانه ی مورد نظر را از جدول (table view) برمی گرداند. برای اینکه کد طبق نیاز برای اپلیکیشن شما کار کند، می بایست placeholder identifier را به آنچه قبلا در storyboard برای خانه ی نمونه (MealTableViewCell) تنظیم کرده بودید، تغییر دهید. سپس کدی اضافه نمایید که خانه ی جدول را تنظیم می کند.

۲. کد زیر را به ابتدای بدنه ی متد مانند زیر اضافه نمایید:

```
// Table view cells are reused and should be dequeued using a cell identifier.  
let cellIdentifier = "MealTableViewCell"
```

این کد یک ثابت (constant) با شناسه ای که در storyboard تنظیم کرده بودید، ایجاد می کند. ۳. Placeholder identifier (اسمی که به عنوان مکان نگهدار تعریف کرده بودید) را به آن identifier ای که در storyboard تعریف کرده بودید، تغییر دهید. دومین خط کد در بدنه ی متد بایستی مشابه زیر باشد:

```
let cell = tableView.dequeueReusableCellWithIdentifier(cellIdentifier,  
forIndexPath: indexPath)
```

از آنجایی که قبلا یک کلاس با پیاده سازی سفارشی از UITableViewCell ایجاد کردید، این امکان برای شما وجود دارد که نوع کلاس ایجاد شده را به کلاس فرزند از UITableViewCell با پیاده سازی اختصاصی، MealTableViewCell، تبدیل

(downcast) نمایید. دومین خط از بدنه ی متد هم اکنون می بایست به صورت زیر باشد:

```
let cell = tableView.dequeueReusableCellWithIdentifier(cellIdentifier,
forIndexPath: indexPath) as! MealTableViewCell
```

۴. در زیر خط قبلی، کد زیر را اضافه نمایید:

```
// Fetches the appropriate meal for the data source layout.
let meal = meals[indexPath.row]
```

این کد آیتم مربوطه ی meal را از آرایه ی meals واکشی می کند.

۵. خط Configure the cell // را حذف نموده و بجای آن این کد را درج نمایید:

```
cell.nameLabel.text = meal.name
cell.photoImageView.image = meal.photo
cell.ratingControl.rating = meal.rating
```

این خط کد تک تک view های موجود در خانه ی جدول را طوری تنظیم می کند که داده های مربوطه را از ثابت meal نمایش دهد.

متد (tableView(\_:cellForRowAtIndexPath:)) هم اکنون بایستی مشابه زیر باشد:

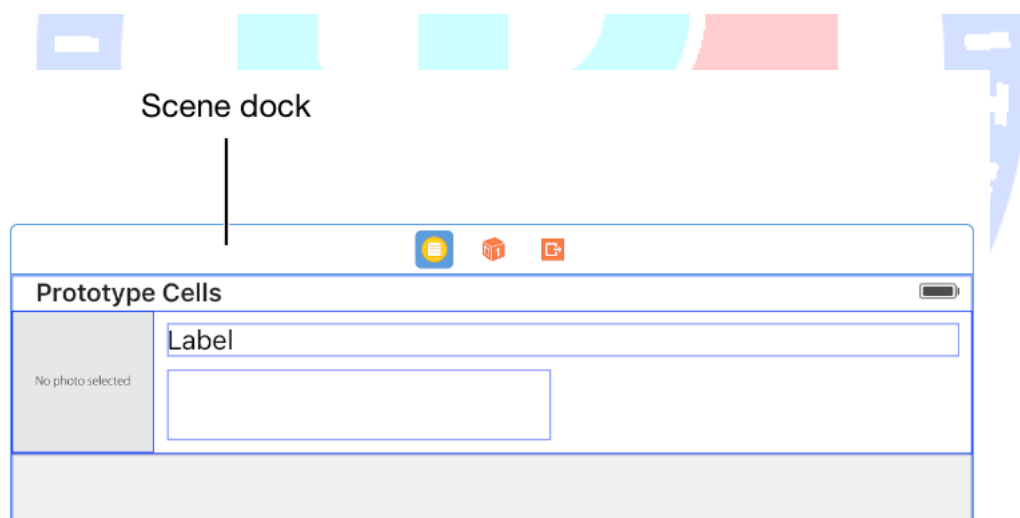
```
override func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {
// Table view cells are reused and should be dequeued using a cell identifier.
let cellIdentifier = "MealTableViewCell"
let cell = tableView.dequeueReusableCellWithIdentifier(cellIdentifier,
forIndexPath: indexPath) as! MealTableViewCell
// Fetches the appropriate meal for the data source layout.
let meal = meals[indexPath.row]
cell.nameLabel.text = meal.name
cell.photoImageView.image = meal.photo
cell.ratingControl.rating = meal.rating
return cell
}
```

آخرین مرحله برای نمایش داده ها در UI اپلیکیشن FoodTracker این است که کد موجود در فایل MealTableViewController.swift را به صفحه محتوای نمایش لیست غذاها (meal list storyboard scene) متصل نمایید.

برای اینکه صفحه محتوای نمای جدولی لیست غذاها (table view controller scene) به کد موجود در فایل متصل شده و به آن اشاره داشته باشد، مراحل زیر را دنبال نمایید:

۱. ابتدا storyboard را باز نمایید.

۲. با کلیک بر روی scene dock/نوار بالایی صفحه محتوای نمایش لیست غذاها (table view controller scene) آن را انتخاب کنید به طوری که یک خط آبی رنگ پیرامون scene جاری را فرا بگیرد.



۳. حال کادر identity inspector را باز نمایید.

۴. داخل کادر identity inspector، فیلدی که Class نام دارد را پیدا کرده و سپس گزینه ی MealTableViewController را انتخاب کنید.

**Custom Class**

Class:

Module:

**Identity**

Storyboard ID:

Restoration ID:

☐ Use Storyboard ID

**User Defined Runtime Attributes**

Key Path	Type	Value
+ -		

**Document**

Label:




Object ID:

Lock:

Notes:

No Font

**تست کنید:** برنامه را اجرا نمایید. لیست آیتم هایی که به متد `viewDidLoad()` اضافه نمودید، باید به صورت خانه های جدول در `table view` به نمایش در آیند. با کمی دقت متوجه می شوید که خانه های جدول با `status bar` (نوار نمایشگر وضعیت کلی دستگاه) کمی همپوشانی دارد (نوار وضعیت به مقدار اندکی بر روی خانه ی پایینی جدول قرار گرفته است). در مبحث بعدی به برطرف کردن این همپوشانی خواهید پرداخت.

	<div>Carrier 9:41 AM</div> <div>Caprese Salad</div> <div>★★★★☆</div>
	<div>Chicken and Potatoes</div> <div>★★★★★</div>
	<div>Pasta with Meatballs</div> <div>★★★☆☆</div>
<hr/>	
<hr/>	
<hr/>	
<hr/>	
<hr/>	

آماده سازی صفحه محتوای افزودن غذای جدید/Meal Scene برای پیمایش (پیاده سازی قابلیت پیمایش در برنامه ی FoodTracker)

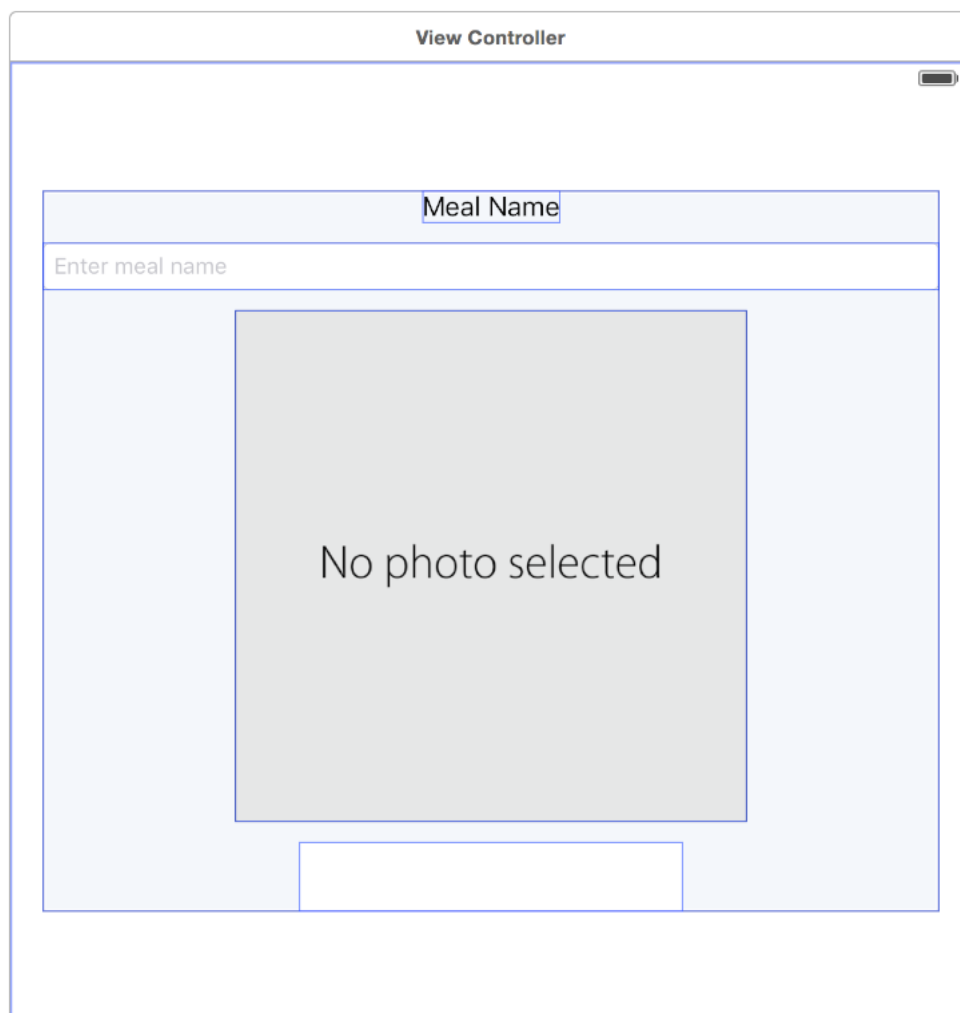
برای اینکه قابلیت پیمایش (navigation) را در اپلیکیشن FoodTracker پیاده سازی کنید، لازم است تعدادی خط کد را از متن برنامه حذف کرده و آن را با کدهای دیگر جایگزین نمایید و یا بخش هایی از UI که با پیاده سازی navigation در آن بلااستفاده می شود، حذف/جایگزین کنید.

جهت حذف بخش های بلااستفاده از پروژه، مراحل زیر را دنبال نمایید:

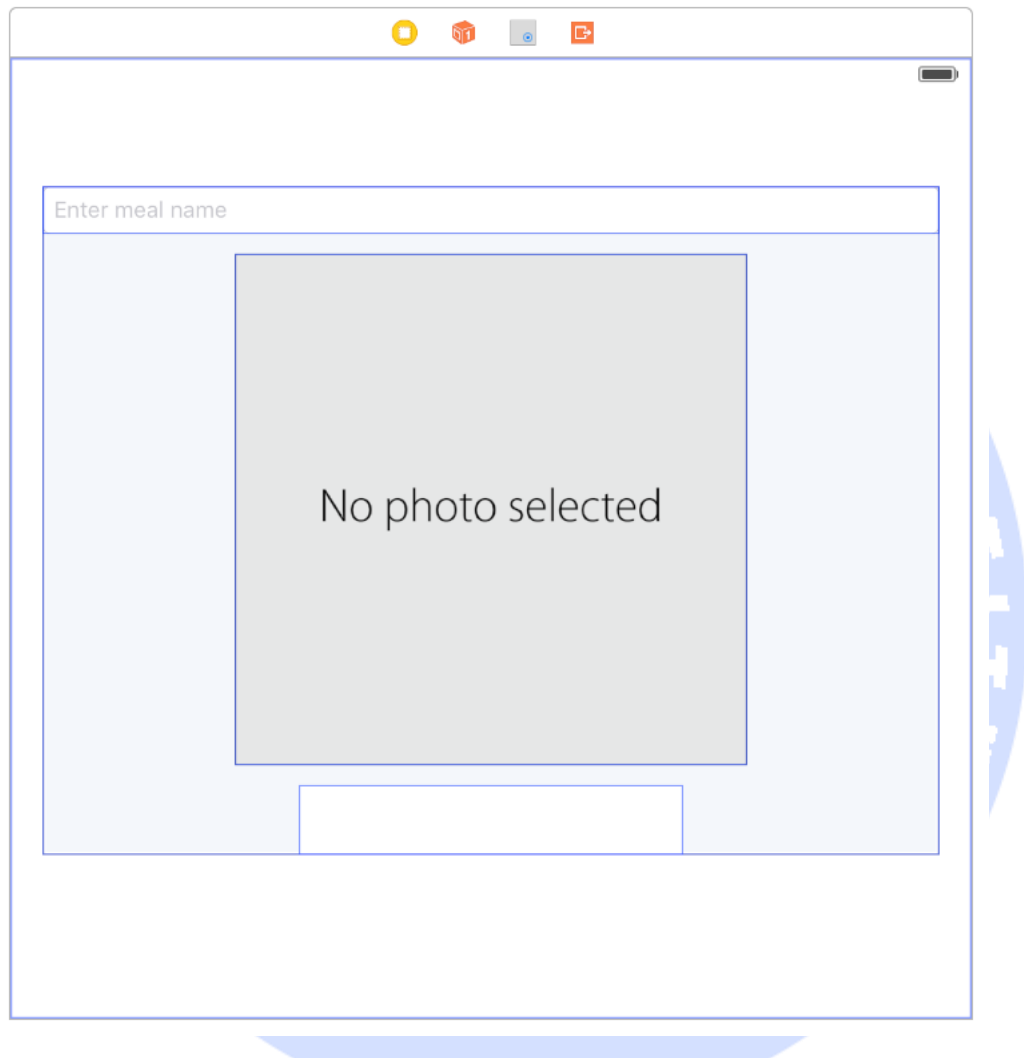
۱. فایل storyboard را باز نموده و صفحه محتوای افزودن غذای جدید (scene)

(meal) را مشاهده نمایید.

صفحه محتوا نمایش meal در حال حاضر می بایست ظاهری مشابه زیر داشته باشد:



۲. در scene جاری، المان Meal Name را انتخاب نموده و با فشردن کلید Delete، آن را حذف کنید. لازم نیست نگران چیدمان المان ها باشید. باقی المان ها موقعیت خود را به طور مناسب در stack view تنظیم می کنند.



۳. فایل ViewController.swift را باز نمایید.

۴. در فایل ViewController.swift، متد `textFieldDidEndEditing(_:)` را پیدا کنید.

```
func textFieldDidEndEditing(textField: UITextField) {
    mealNameLabel.text = textField.text
}
```

۵. آن خطی که مقدار خصوصیت `text` (property) از المان label را مقداردهی می کند، حذف نمایید.



```
mealNameLabel.text = textField.text
```

۶. به زودی داخل بدنه ی این متد، دستور دیگری درج خواهید نمود.

۷. در فایل ViewController.swift، متغیر (یا outlet) mealNameLabel را پیدا کرده و آن را حذف نمایید.

```
@IBOutlet weak var mealNameLabel: UILabel!
```

شما اکنون دو view controller در پروژه ی خود دارید. بنابراین بهتر است اسم فایل ViewController.swift را تغییر داده و نام معنی دار تری برای آن انتخاب نمایید. جهت تغییر اسم فایل ViewController.swift، مراحل زیر را دنبال نمایید:

۱. در کادر project navigator، بر روی فایل ViewController.swift کلیک کرده و سپس کلید Return را فشار دهید. Xcode به شما این امکان را می دهد تا برای فایل اسم جدیدی تایپ کنید.

۲. فایل MealViewController.swift را تغییر اسم داده و سپس کلید Return را فشار دهید.

۳. در فایل MealViewController.swift، خط تعریف کلاس (حاوی کلید واژه ی class) را پیدا کنید:

```
class ViewController: UIViewController, UITextFieldDelegate, UIImagePickerControllerDelegate, UINavigationControllerDelegate {
```

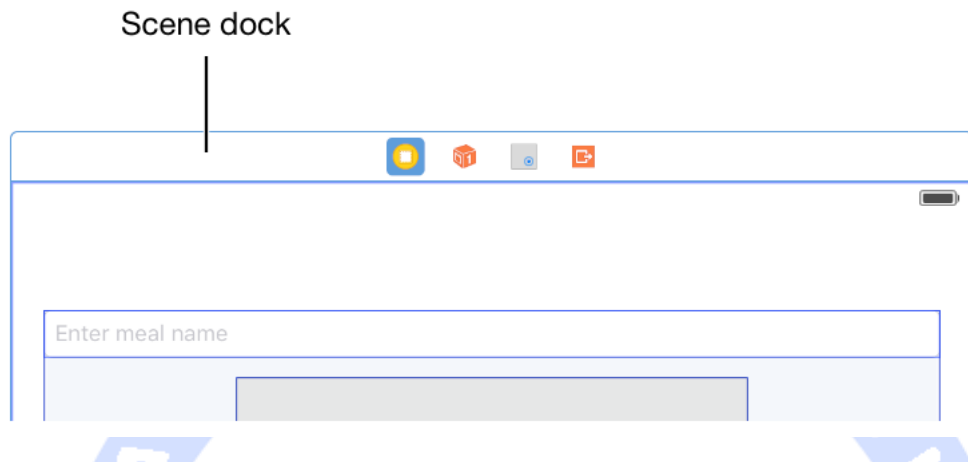
۴. اسم کلاس را به MealViewController تغییر دهید.

```
class MealViewController: UIViewController, UITextFieldDelegate, UIImagePickerControllerDelegate, UINavigationControllerDelegate {
```

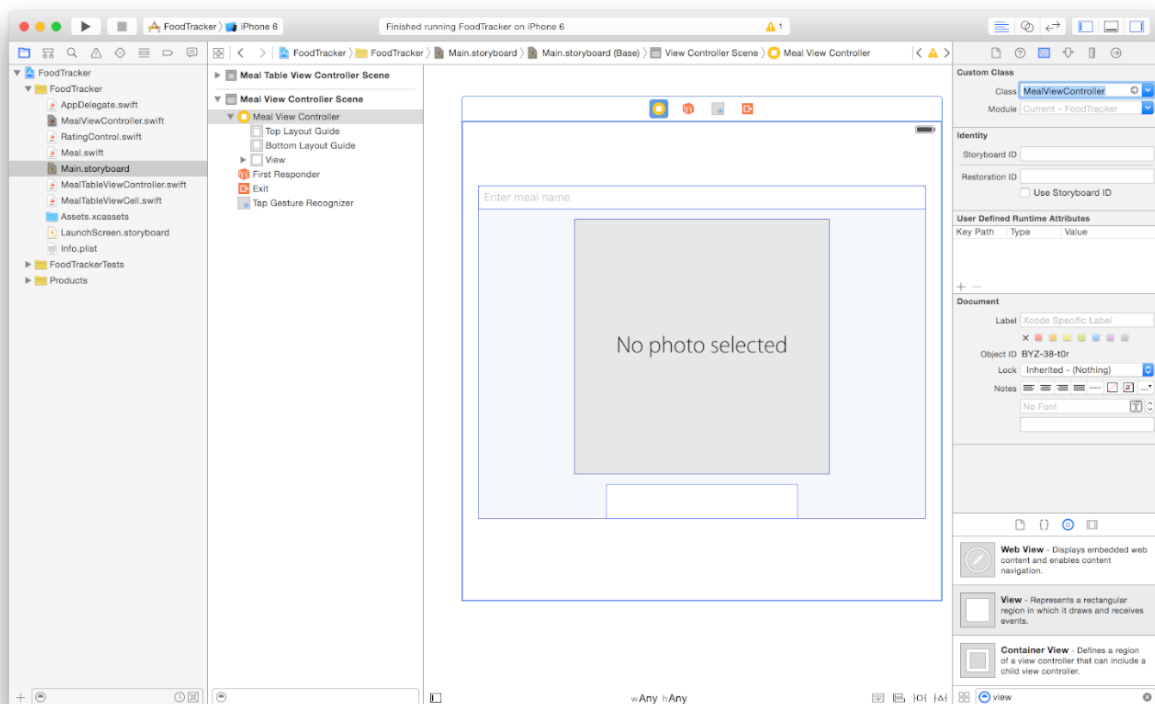
۵. در comment موجود در بالای فایل نیز اسم فایل را از ViewController.swift به MealViewController.swift تغییر دهید.

۶. اکنون فایل storyboard را باز کنید.

۷. صفحه محتوای افزودن غذای جدید (meal scene) را با کلیک بر روی نوار scene dock انتخاب نمایید.



۸. پس از انتخاب scene مذکور، کادر Identity inspector را باز نمایید.  
 ۹. در این کادر، به فیلد Class مراجعه کرده و گزینه ی انتخاب شده را از ViewController به MealViewController تغییر دهید.



**تست کنید:** اپلیکیشن خود را کامپایل/اجرا نمایید. همه چیز باید مانند قبل به درستی کار کند. در این لحظه احتمالاً یک پیغام هشدار به نمایش در می آید. در آن شرح داده شده

که هیچ راهی برای دسترسی به صفحه محتوای افزودن غذای جدید به برنامه FoodTracker (meal scene) وجود ندارد. جای نگرانی نیست. این رفتار (قابلیت پیمایش و دسترسی به meal scene) را در مبحث بعدی پیاده سازی خواهید کرد.

## درس ۹ : آموزش پیاده سازی قابلیت پیمایش (navigation) در Swift

### پیاده سازی قابلیت پیمایش (navigation)


در این مبحث، با بهره گیری از ابزاری نظیر navigation controller و segue ها قابلیت پیمایش را در اپلیکیشن FoodTracker پیاده سازی خواهید نمود. در پایان مبحث، قادر خواهید به راحتی بین صفحات مختلف برنامه راهبری نموده و با آن ها تعامل داشته باشید.

ظاهر برنامه در پایان به صورت زیر خواهد بود:




Carrier
9:41 AM

Your Meals
+




Caprese Salad

★ ★ ★ ★ ☆



Chicken and Potatoes

★ ★ ★ ★ ★



Pasta with Meatballs

★ ★ ★ ☆ ☆

آنچه خواهید آموخت

- یک view controller از قبل موجود را داخل یک navigation controller در storyboard جاسازی نمایید/بگنجانید.
- جهت انتقال یا پیمایش بین دو view controller از segue ها بهره بگیرید.
- از طریق کادر attribute inspector، خصیصه ها (attribute های) یک segue را در storyboard ویرایش نمایید.
- با بهره گیری از متد prepareForSegue(\_:sender:) داده ها را بین view controller ها رد و بدل نمایید.
- بتوانید یک unwind segue اجرا نمایید (قابلیت پیمایش به عقب پیاده سازی کنید).
- با استفاده از stack view، قالب های (layout) کارآمد، انعطاف پذیر و قدرتمند ایجاد نمایید.

اضافه کردن یک segue جهت پیمایش به جلو (صفحه ی بعدی)

در حالی که داده های اپلیکیشن به صورت مورد انتظار در UI برنامه به نمایش گذاشته می شوند، اکنون زمان آن رسیده که امکان پیمایش از صفحه نمایش لیست غذاها به صفحه محتوا افزودن غذای جدید (meal scene) را فراهم نمایید. در برنامه نویسی iOS، به انتقال از یک صفحه محتوا (scene) به صفحه محتوا دیگر در اصطلاح segue گفته می شود. پیش از ایجاد یک segue، لازم است scene ها یا همان صفحات حاوی محتوای برنامه ی خود را تنظیم و پیکربندی نمایید.

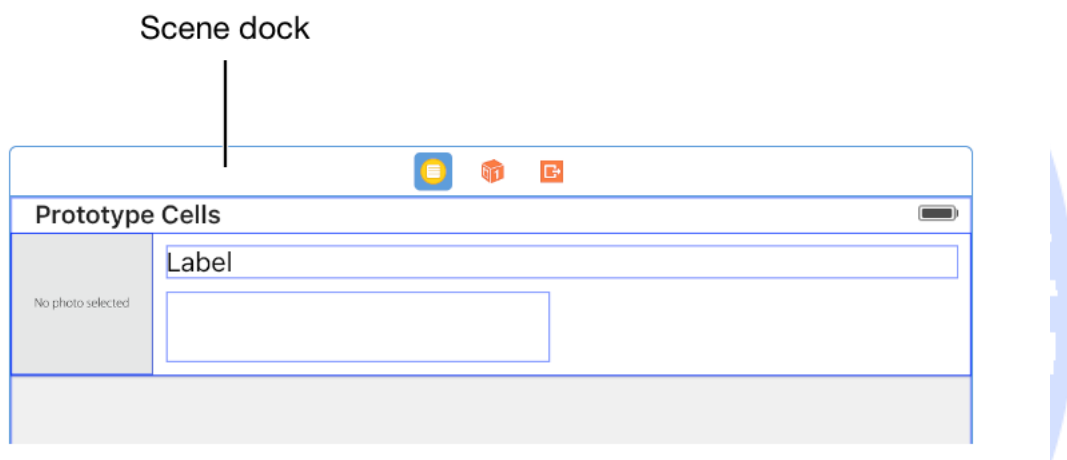
در مرحله ی اول می بایست یک view controller را داخل navigation controller جاسازی نمایید. حال این سوال پیش می آید که navigation controller چیست؟ navigation controller یک کلاس فرزند و مشتق شده ی view controller با پیاده سازی اختصاصی است که امکان پیمایش به جلو و عقب بین مجموعه ای از view controller ها را مهیا می سازد. به مجموعه ای از view controller ها که توسط یک navigation controller مدیریت می شوند، navigation stack گفته می شود. اولین

آیتمی که به پشته اضافه می شود، root view controller خوانده شده (view controller آغازین) و هیچگاه از روی navigation stack حذف (pop off) نمی شود.

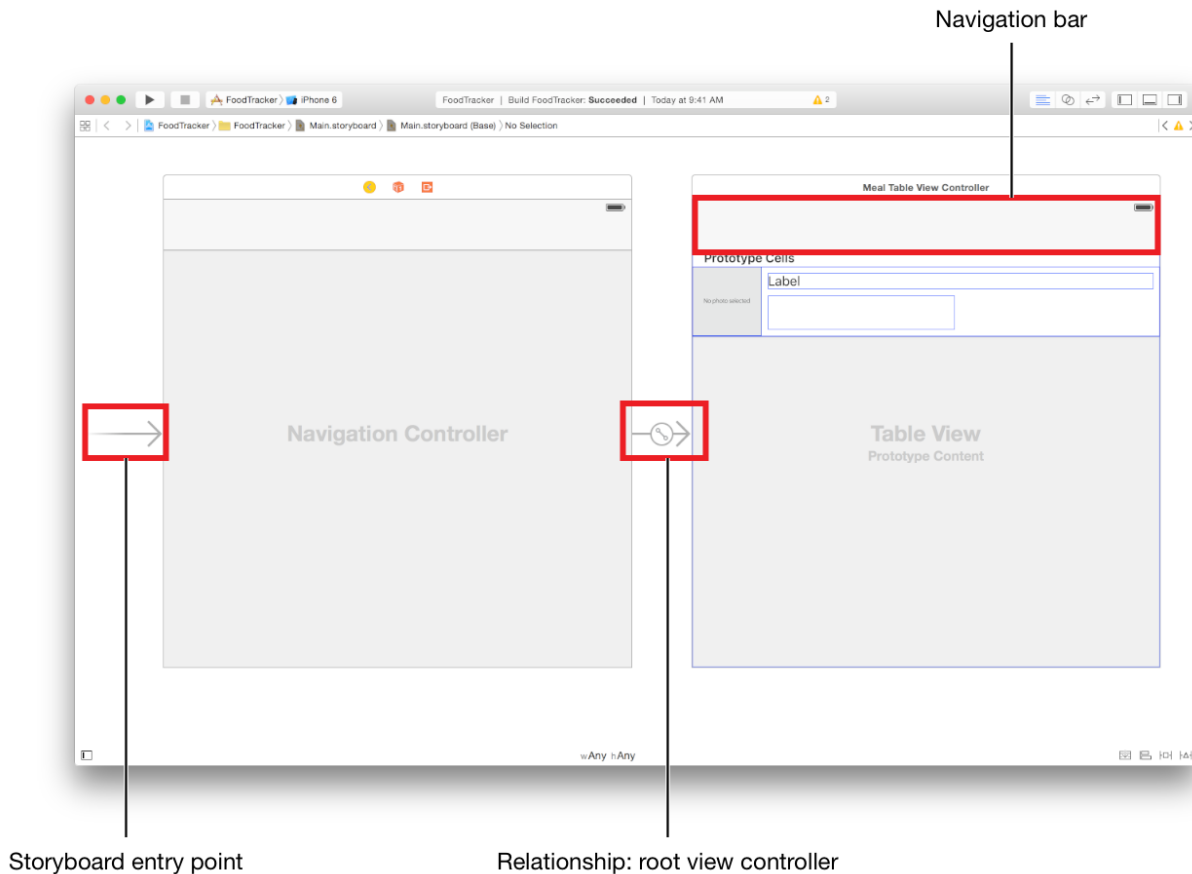
جهت افزودن یک navigation controller به meal list scene (صفحه ی نمایش لیست غذاها با نمای جدولی)، مراحل زیر را دنبال نمایید:

۱. ابتدا فایل storyboard خود، Main.storyboard را باز نمایید.

۲. Table view controller (نمای جدولی) را با کلیک بر روی نوار scene dock آن، انتخاب نمایید.



۳. پس از انتخاب table view controller، این مسیر را طی نمایید: Editor > Embed In > Navigation Controller. محیط کاری Xcode یک navigation controller جدید به storyboard جاری اضافه کرده، آن را به عنوان صفحه محتوای آغازین این storyboard انتخاب می کند (آن را به عنوان entry point منصوب کرده) و سپس یک رابطه بین navigation controller جدید و table view controller فعلی ایجاد می کند.

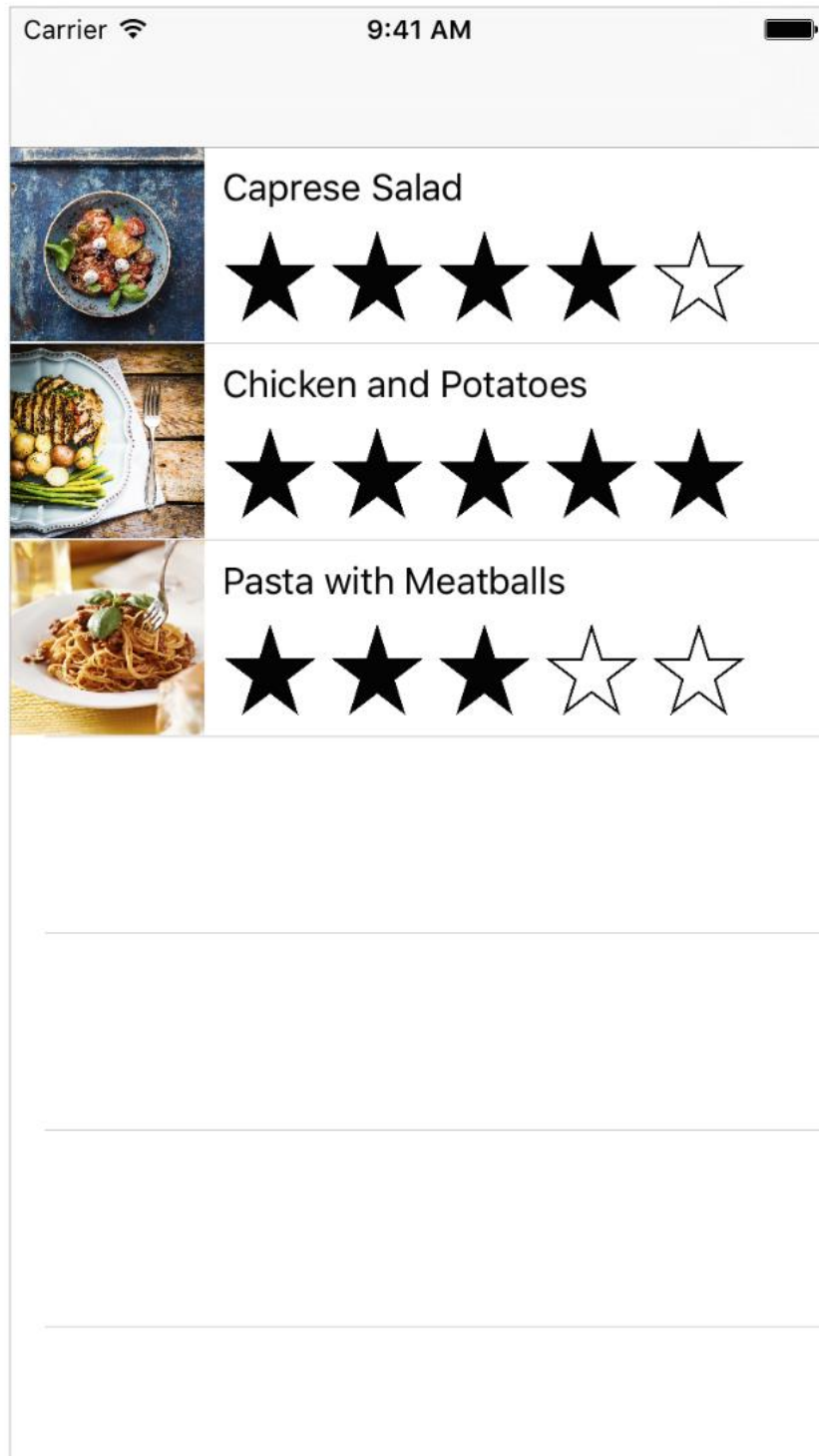


در سطح canvas (پس زمینه ی storyboard)، آیکونی که controller ها را به هم متصل می کند، یک رابطه است (root view controller relationship). Table view controller همان root view controller (view آغازین) navigation controller است که لیست غذاها را نمایش می دهد. storyboard entry point به این خاطر به navigation controller تخصیص داده شده که navigation controller نام برده در حال حاضر نقش یک ظرف (container) را برای table view controller ایفا می کند. با کمی دقت متوجه نواری در بالای نمای جدولی خود (table view) می شوید. این نوار در واقع نوار پیمایش (navigation bar) جهت راهبری در اپلیکیشن شما است. به تک تک controller های موجود در پشته یک نوار پیمایش اختصاص داده می شود که دارای دو control برای پیمایش به جلو و عقب می باشد. در گام بعدی یک دکمه به نوار پیمایش جهت راهبری به صفحه افزودن غذای جدید meal scene اضافه خواهید نمود.

**تست کنید:** برنامه ی خود را اجرا کنید. در بالای table view، فضای اضافی قابل مشاهده می باشد. این فضای اضافی همان navigation bar یا نوار پیمایش است که توسط navigation controller جهت راهبری در صفحات متعدد اپلیکیشن در اختیار شما قرار می گیرد. Navigation bar پس زمینه ی خود را به بالای status bar (نوار نمایشگر وضعیت کلی دستگاه) وام می دهد تا بدین وسیله status bar دیگر با محتوای صفحه همپوشانی نداشته باشد (لبه ی آن بر روی سطر بالایی table view قرار نگیرد).







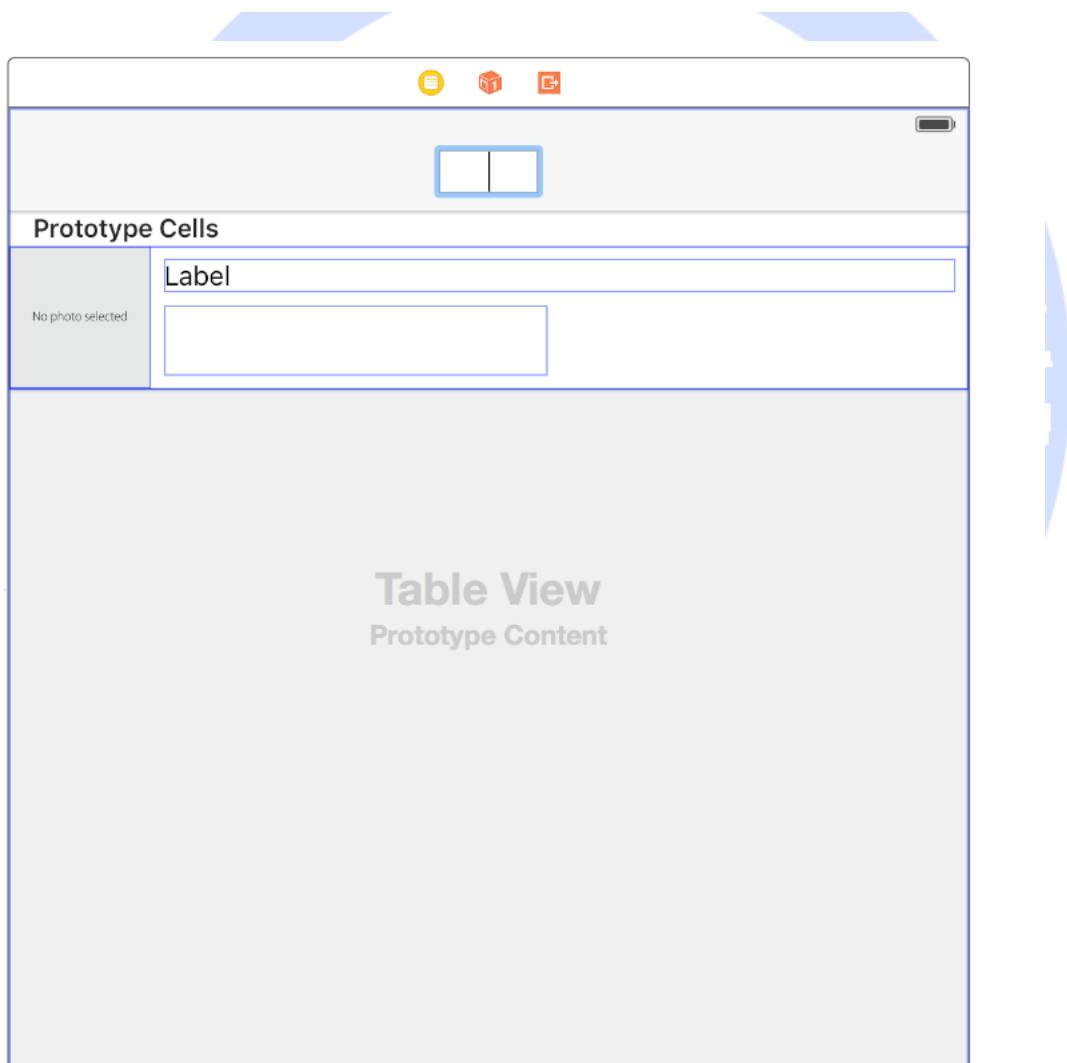
### تنظیم و پیکربندی Navigation Bar برای scene ها

اکنون یک عنوان (به meal list) و یک دکمه (جهت افزودن غذاهای جدید) به نوار پیمایش اضافه خواهید نمود. Navigation bar ها عنوان خود را از آن view controller ای که navigation controller در حال حاضر نمایش می دهد، می گیرند - گفتنی است که Navigation bar ها به خودی خود عنوان ندارند. عنوان مربوطه را بجای اینکه

مستقیماً بر روی navigation bar تنظیم کنید، با استفاده از آیتم navigation از meal list (صفحه نمایش لیست غذاها / table view controller) تنظیم خواهید نمود.

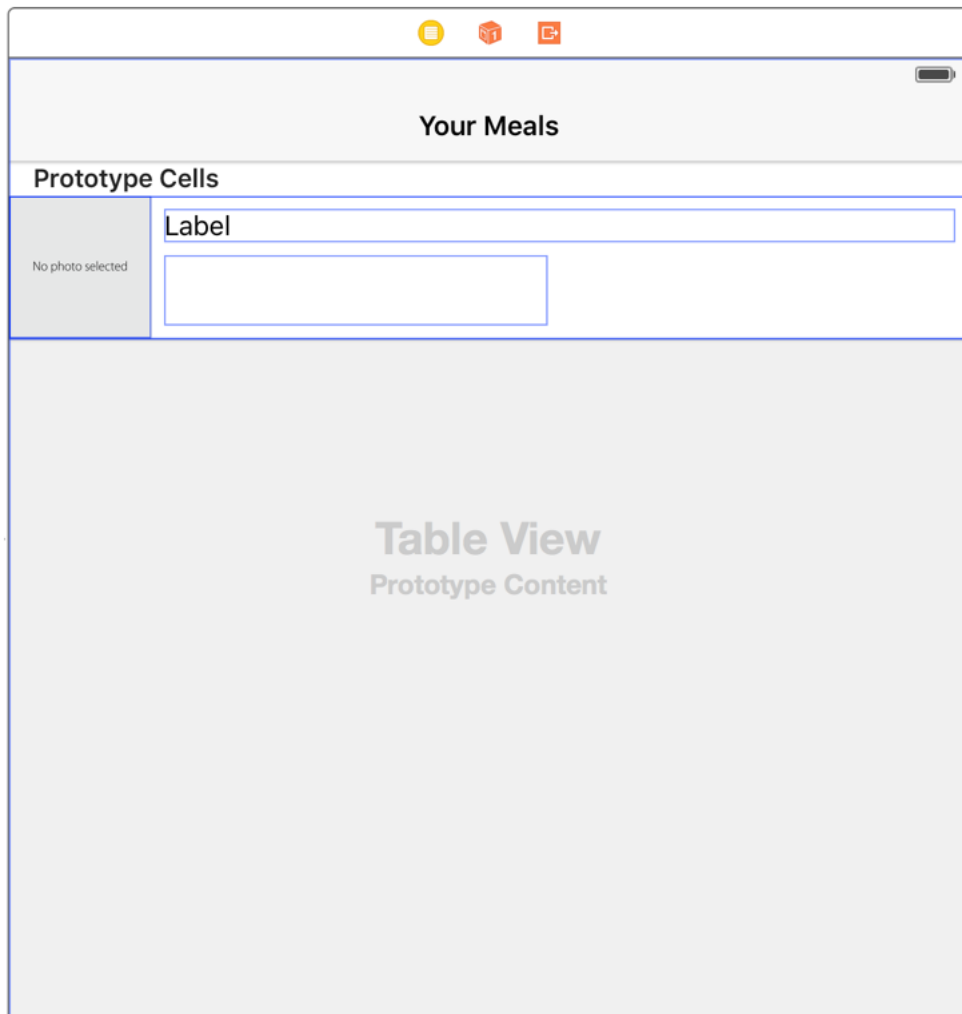
جهت تنظیم navigation bar در meal list (یا لیست نمایش غذاهای جدول)، مراحل زیر را به ترتیب دنبال نمایید:

۱. بر روی navigation bar در صفحه محتوای نمایش لیست غذاها (meal list scene) دوبار کلیک نمایید.



یک مکان نما به نمایش درآمده و به شما اجازه ی درج متن جدید را می دهد.

۲. واژه ی Your Meals را وارد نموده، سپس کلید Return را جهت ذخیره سازی تغییرات فشار دهید.

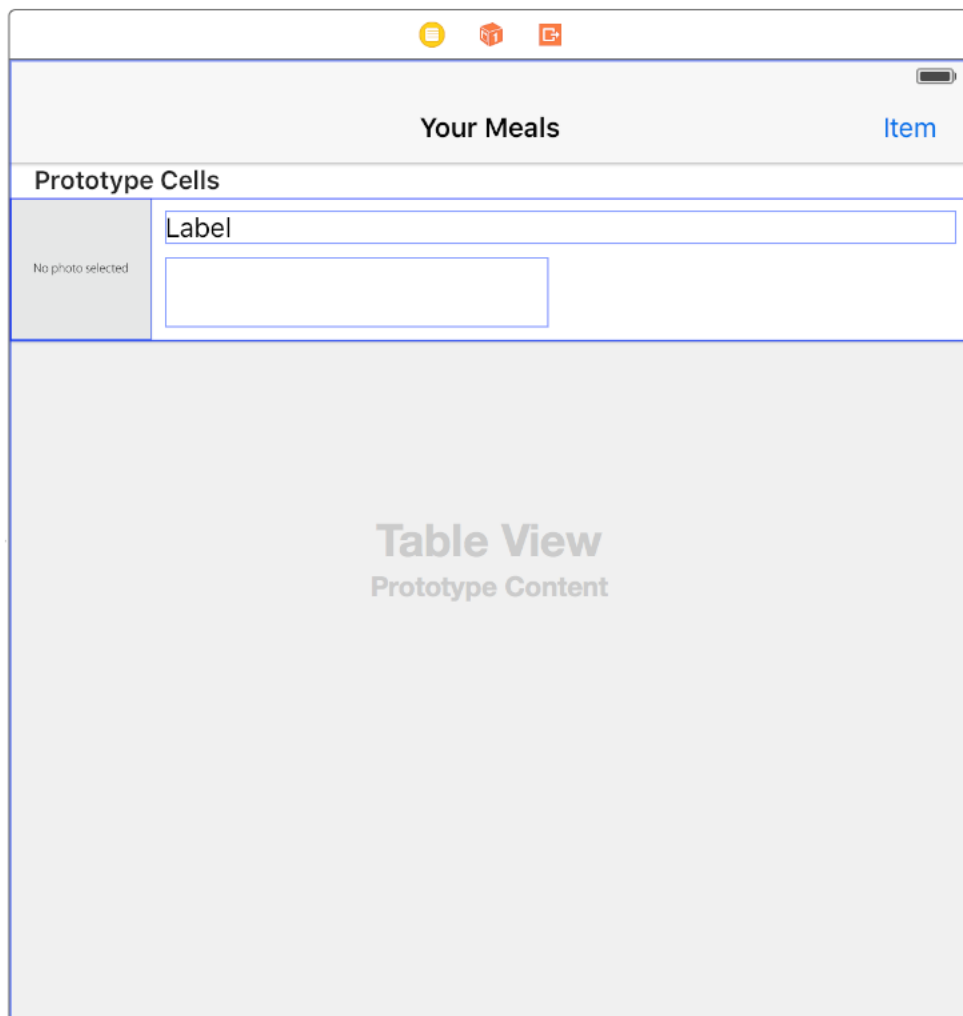


۳. کادر Object library را باز نمایید (برای این منظور کافی است مراحل روبرو را دنبال نمایید: View > Utilities > Show Object Library).

۴. در کادر object library، آبجکت Bar Button Item را جستجو نمایید.

۵. حال آبجکت مزبور را از کادر object library کشیده و در سمت راست نوار پیمایش در meal list scene جایگذاری نمایید.

۶. یک دکمه به نام item در جایی که آبجکت Bar Button را در آن قرار دادید نمایان می شود.

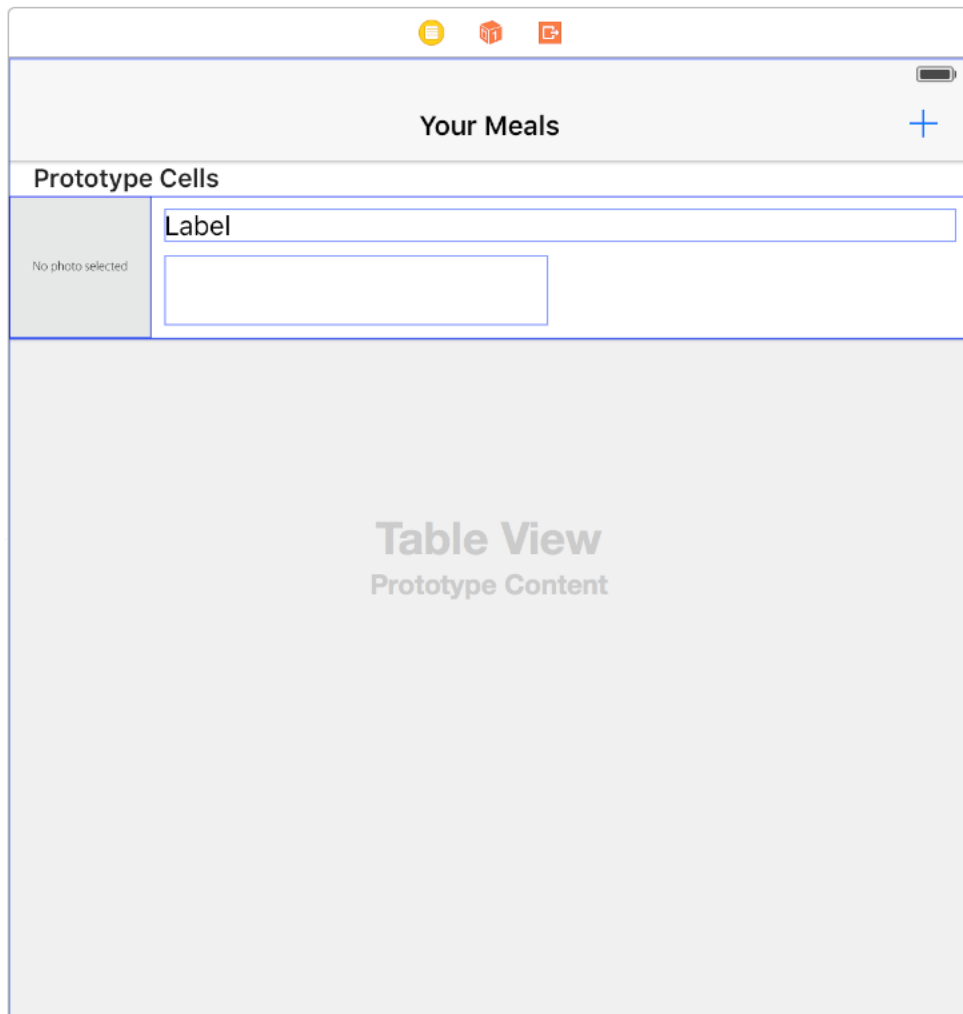


۷. دکمه ی item را انتخاب نموده، سپس کادر Attribute inspector را باز نمایید.

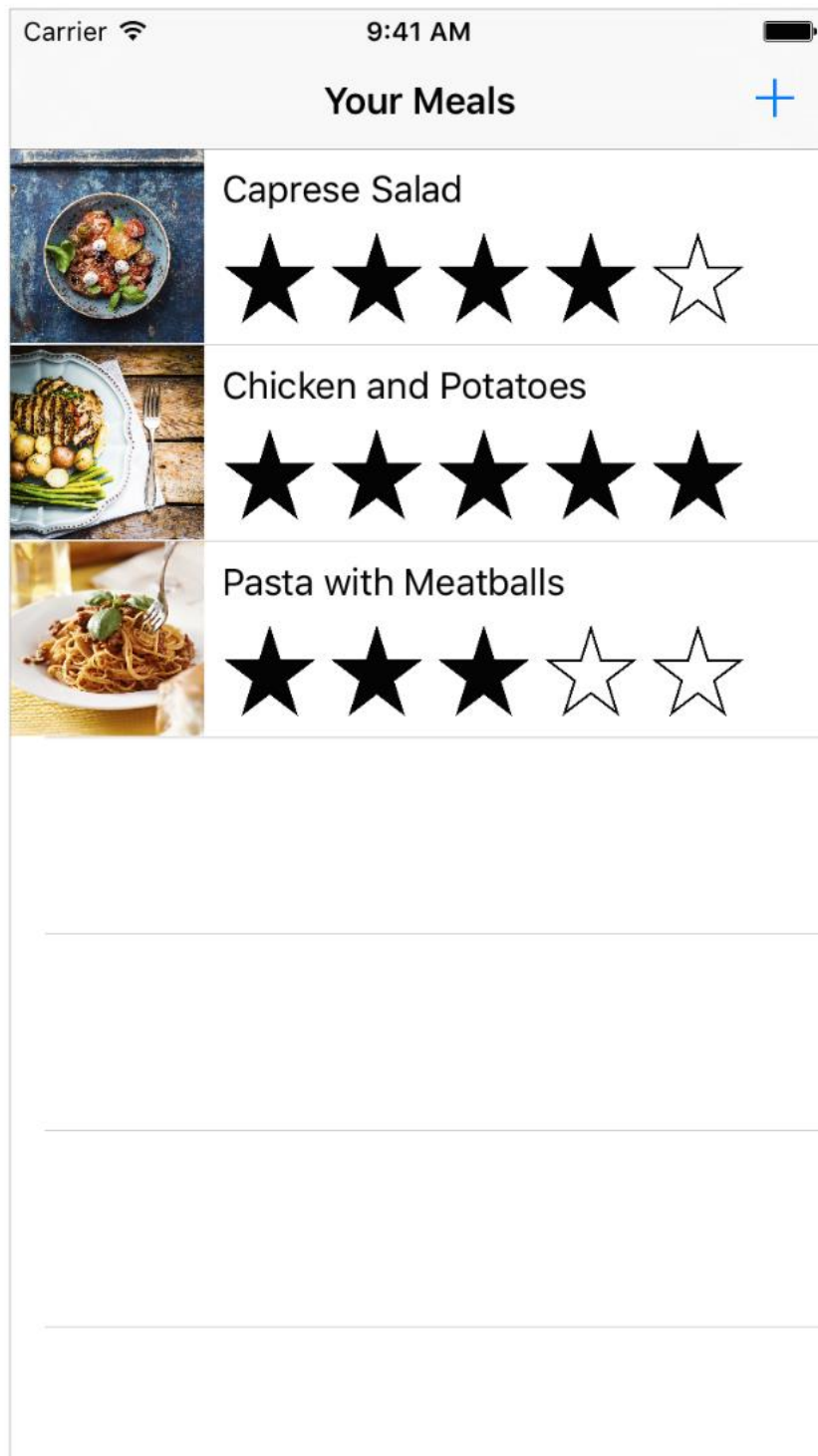
۸. در کادر Attribute inspector، گزینه ی Add را از منوی pop-up جنب آپشن

System item انتخاب نمایید. خواهید دید که ظاهر دکمه به یک علامت (+)

تغییر می کند.



**تست کنید:** برنامه را اجرا کنید. در حال حاضر نوار پیمایش می بایست یک عنوان داشته و یک دکمه ی (+) برای کاربر به نمایش بگذارد. البته دکمه ی ذکر شده در زمان حاضر کار خاصی انجام نمی دهد. در گام بعدی رفتار آن را پیاده سازی خواهید نمود.

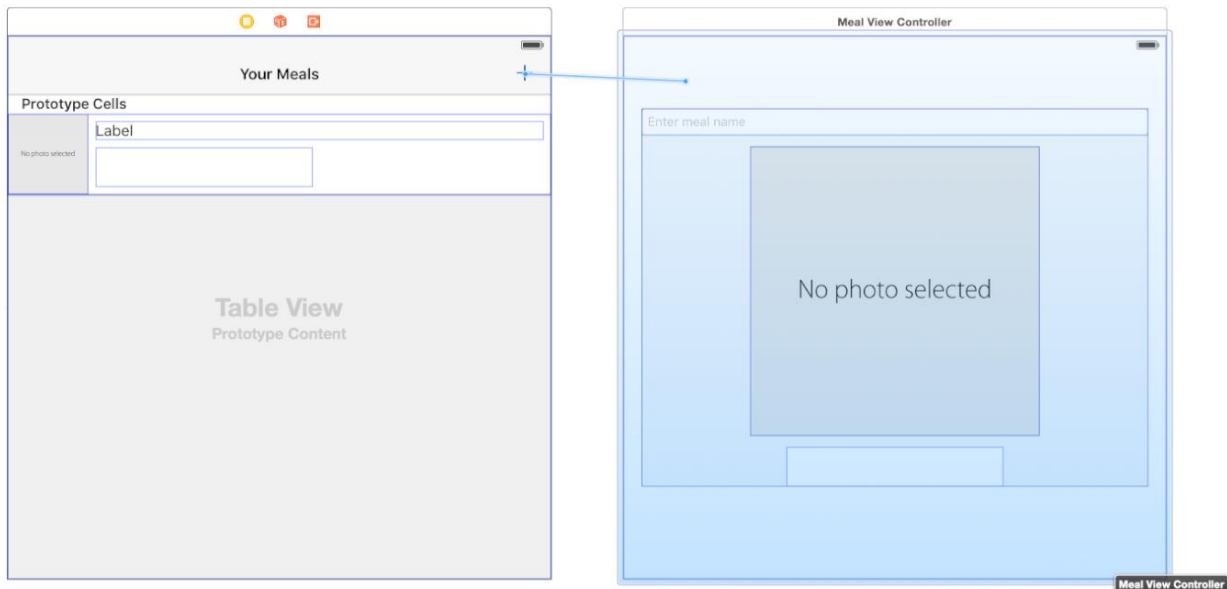


شما می خواهید دکمه ی (+)، صفحه ی افزودن غذای جدید به لیست غذاهای جاری را فراخوانی کرده و برای کاربر به نمایش بگذارد. برای این منظور، رفتاری برای دکمه تعریف می کنید که طی آن یک انتقال به صفحه ی دیگر از برنامه رخ می دهد (با کلیک بر روی دکمه یک segue رخ داده و صفحه ی افزودن غذای جدید پدیدار شود).

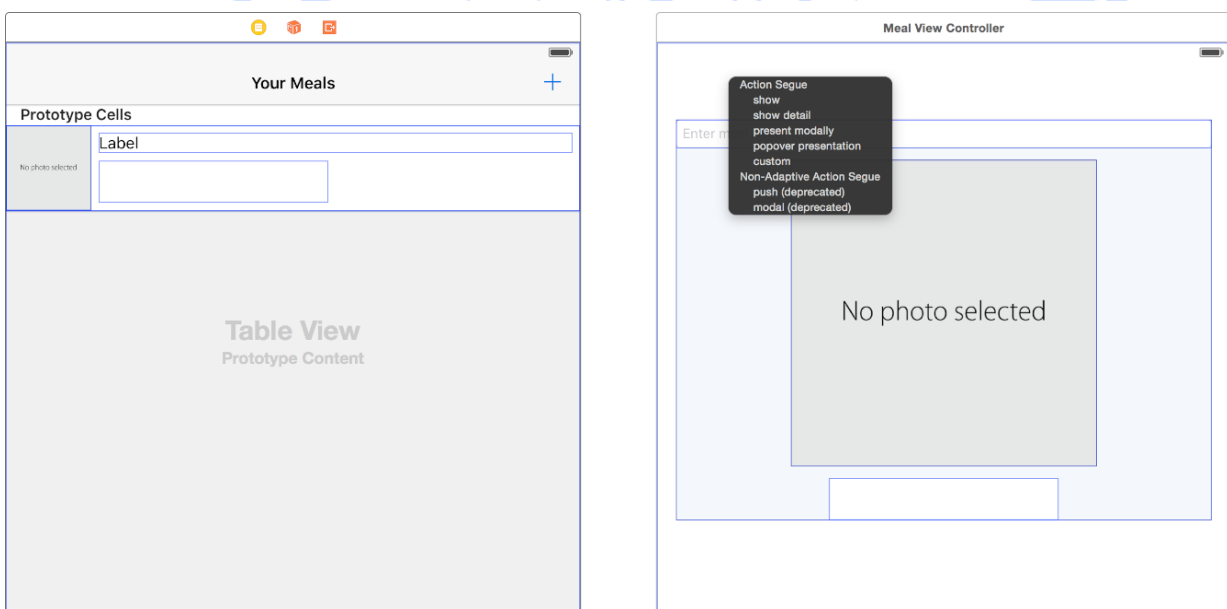
جهت تنظیم و پیکربندی دکمه ی Add در meal scene مراحل زیر را دنبال نمایید:

۱. در سطح canvas دکمه ی (+) را انتخاب نمایید.

۲. دکمه را با اشاره گر موس کشیده و در سطح meal scene جایگذاری نمایید.



یک منوی میانبهر با عنوان Action Segue در مکانی که کشیدن آیتم را در آن متوقف کردید، پدیدار می شود.

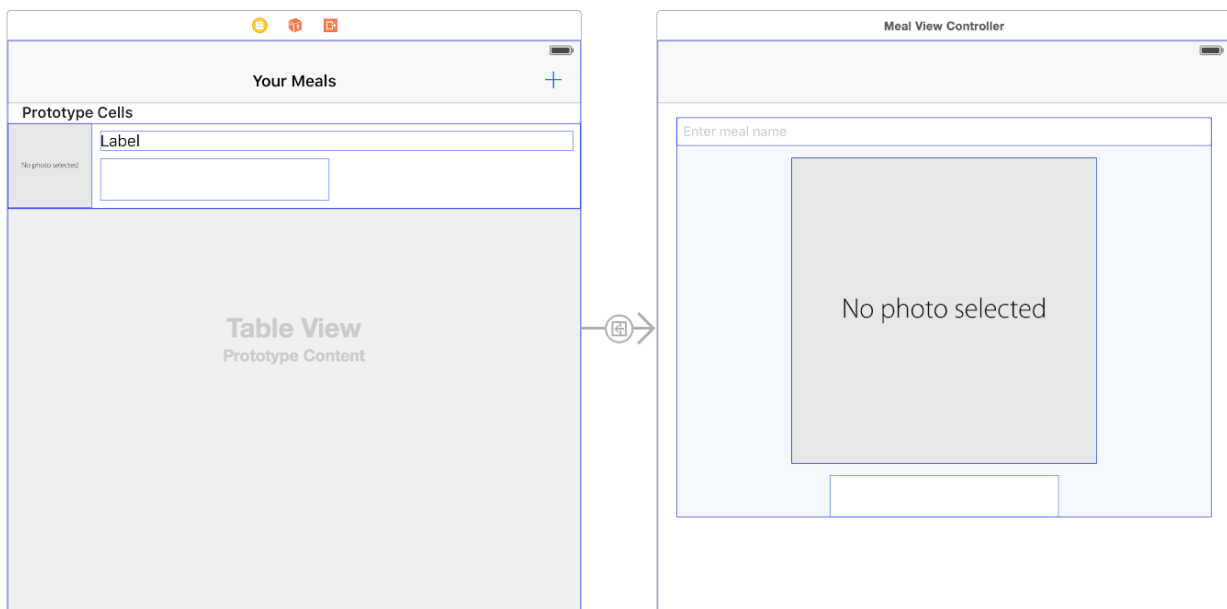


منوی Action Segue حاوی تعدادی گزینه است که از طریق آن ها می توانید نوع segue (انتقال) از صفحه نمایش لیست غذاها (meal list) به صفحه ی افزودن غذای

جدید (new meal view controller) را زمانی که کاربر بر روی دکمه ی مورد نظر کلیک می کند، تعیین نمایید.

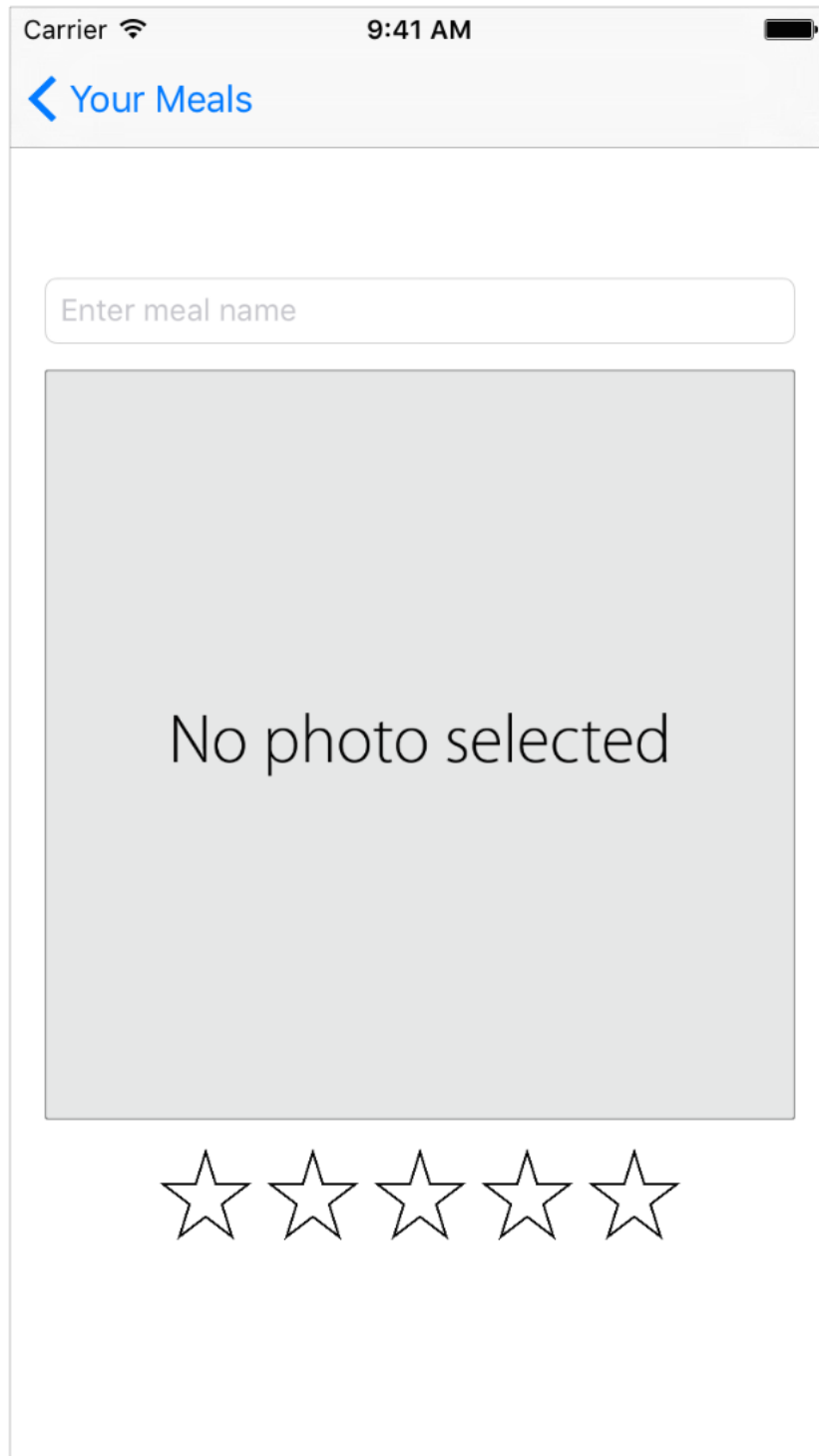
۳. گزینه ی show را از منوی Action Segue انتخاب نمایید.

محیط کاری Xcode انتقال از صفحه ی نمایش لیست غذاها (meal list) به صفحه ی افزودن غذای جدید (meal scene) را به سبک انتخاب شده (show segue) پیگیربندی نموده، تنظیمات لازم جهت نمایش meal scene در یک navigation controller را انجام می دهد – navigation bar را در Interface Builder می بینید.



**تست کنید:** برنامه را اجرا نمایید. با توجه به کدی که برای برنامه نوشته اید، باید بتوانید بر روی دکمه ی (+) کلیک کرده و به صفحه محتوای افزودن غذای جدید پیمایش نمایید. به این خاطر که برای انتقال از یک صفحه محتوا به صفحه محتوای دیگر از show segue استفاده می کنید، پیمایش به عقب (backward navigation) به صورت آماده پیاده سازی شده است و دکمه ی پیمایش به عقب جهت بازگشت به صفحه نمایش لیست غذاها به صورت خودکار در meal scene (صفحه ی افزودن غذای جدید) به نمایش در می آید.





پیمایش به سبک push (قرار دادن محتوای جدید بر روی پشته ای از view controller های جاری) که با انتخاب show segue در اختیار شما قرار می گیرد، به صورت مورد انتظار کار می کند – اما این سبک پیمایش به صفحه ی دیگر آنچه که شما به هنگام افزودن آیتم جدید برای برنامه ی جاری نیاز دارید، نیست. پیمایش به سبک push بیشتر مناسب رابط های کاربری drill-down (راهبری یا

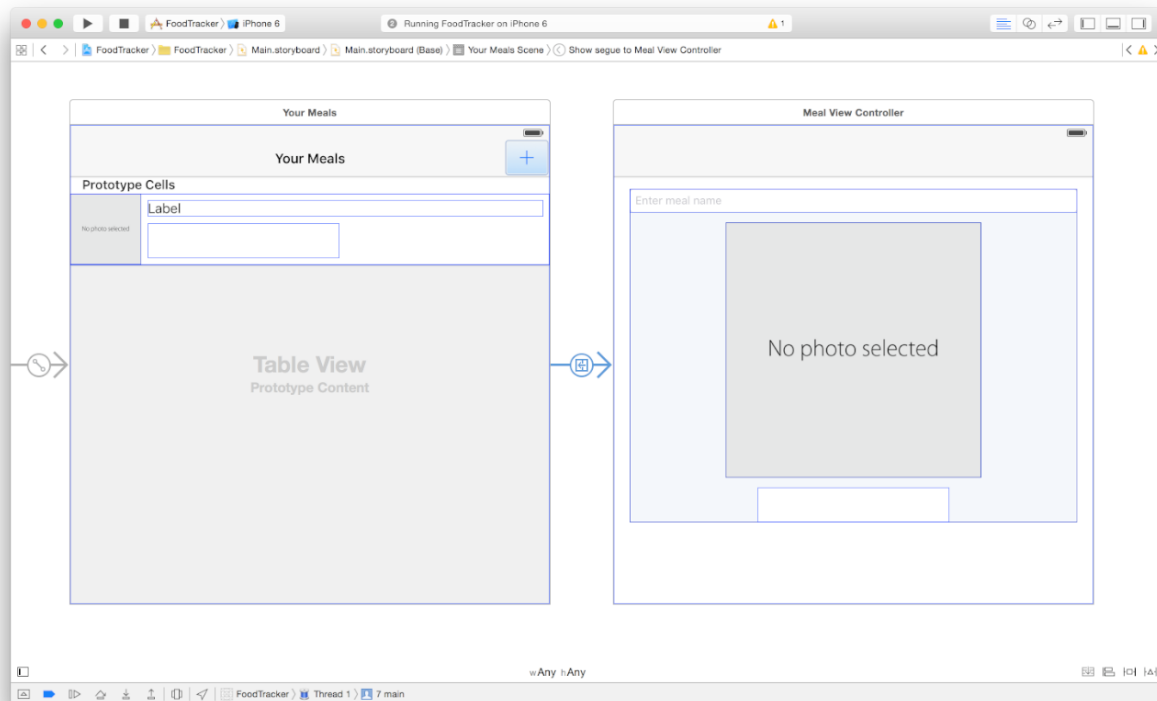
پیمایش از بالا به پایین) می باشد که در آن می خواهید اطلاعاتی را در خصوص انتخاب های کاربر در UI نمایش دهید. برای اضافه کردن آیتم جدید معمولا باید یک نوع عملیات modal انجام شود – صفحه ای مجزا به صورت شناور باز می شود و کاربر در آن عمل خاصی را انجام می دهد، متعاقبا از آن scene یا صفحه به navigation اصلی باز می گردد. روش صحیح و مناسب برای این نوع نمایش یک modal segue است. در این نوع segue یک view controller، view controller دیگری را به عنوان فرزند خود به نمایش گذاشته و از کاربر انتظار دارد که فعل خاصی را بر روی آن انجام دهد (برای مثال متنی را وارد کند)، پس از دریافت واکنش مورد نظر از جانب کاربر، او را به جریان عادی برنامه بر می گرداند.

بجای حذف segue جاری و ایجاد یک segue جدید، کافی است سبک انتقال (segue style) از صفحه به صفحه ی دیگر را از طریق کادر Attribute inspector تغییر دهید. می توانید attribute های یک segue را مانند دیگر آیتم های قابل گزینش در storyboard، از طریق کادر Attribute inspector ویرایش نمایید.

جهت تغییر سبک انتقال از یک صفحه به صفحه ی دیگر (segue style)، مراحل زیر را دنبال نمایید:

۱. ابتدا segue (جهت انتقال) از صفحه نمایش لیست غذاها (meal list scene) به صفحه ی افزودن غذای جدید (meal scene) را انتخاب نمایید.





۲. در کادر Attribute inspector، گزینه ی Modally Present را از منوی pop-up،

جنب گزینه ی Segue انتخاب نمایید.

۳. داخل کادر Attribute inspector، واژه ی AddItem را در فیلد Identifier تایپ

نمایید. بعده ها به identifier یا اسم انتخاب شده برای segue مورد نظر جهت

دسترسی و اشاره به آن احتیاج خواهید داشت.

با انجام مراحل عنوان شده در بالا، modal view controller به navigation stack

اضافه نمی شود. به همین خاطر هم این آیتم نوار پیمایش (navigation bar) خود را از

navigation controller مربوط به meal list وام نمی گیرد. اما شما لازم دارید که با

اضافه شدن نوار پیمایش برای کاربر قابلیت پیمایش بین صفحات را فراهم نموده و در

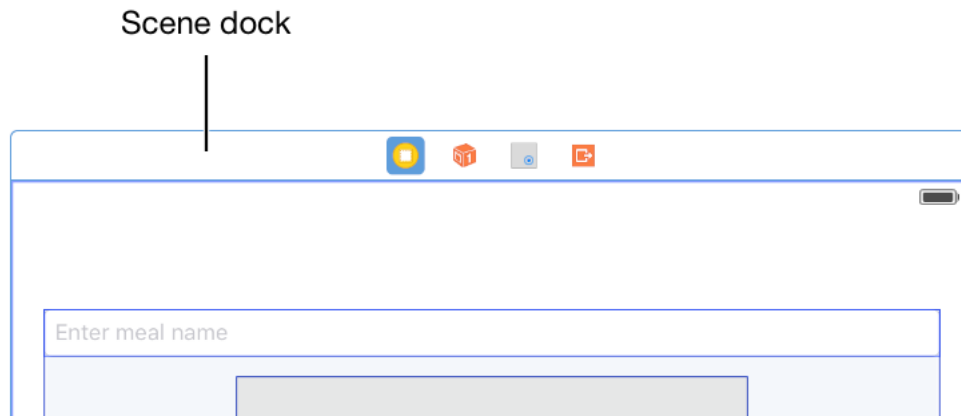
واقع یک نوع پیوستگی بصری را در برنامه ی خود بوجود بیاورید.

به منظور اینکه meal list به هنگام نمایش به صورت modal دارای نوار پیمایش باشد،

کافی است آن را در navigation controller ویژه ی خود جاسازی نمایید (بگنجانید).

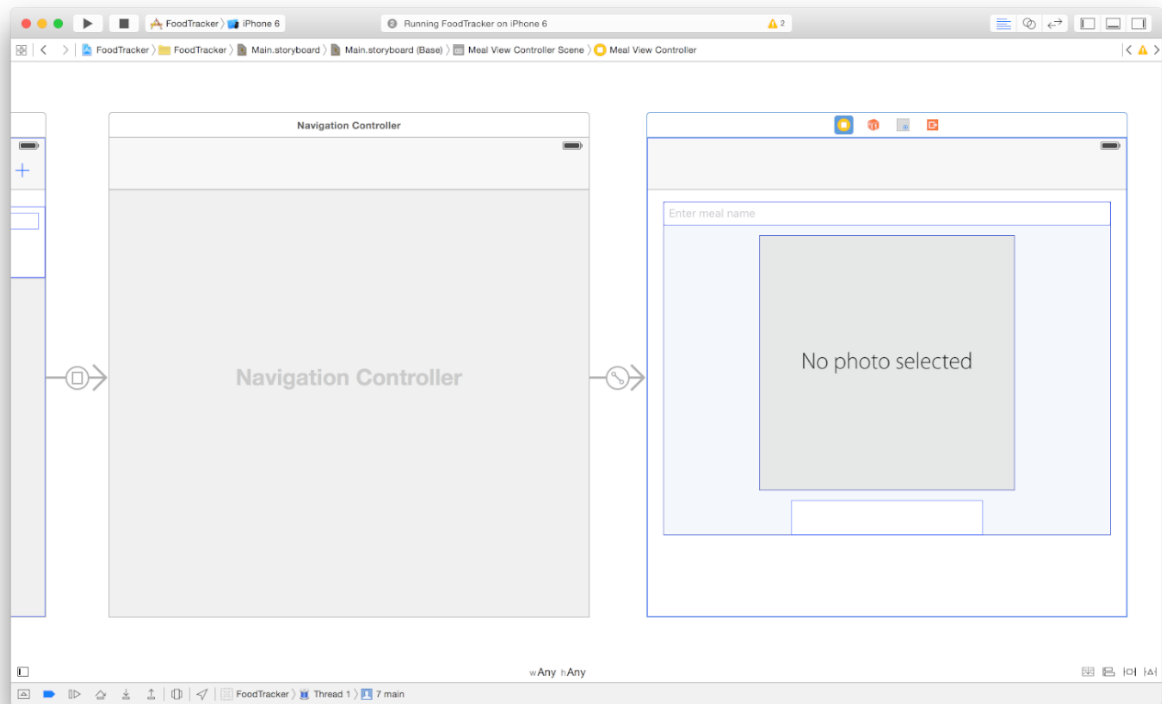
به منظور افزودن navigation controller به صفحه ی افزودن غذای جدید ( meal scene )، کافی است مراحل زیر را دنبال نمایید:

۱. Meal scene را با کلیک بر روی scene dock مربوطه ی آن، انتخاب نمایید.



۲. پس از انتخاب view controller، مراحل رو به رو را دنبال نمایید: Editor > Embed In > Navigation Controller.

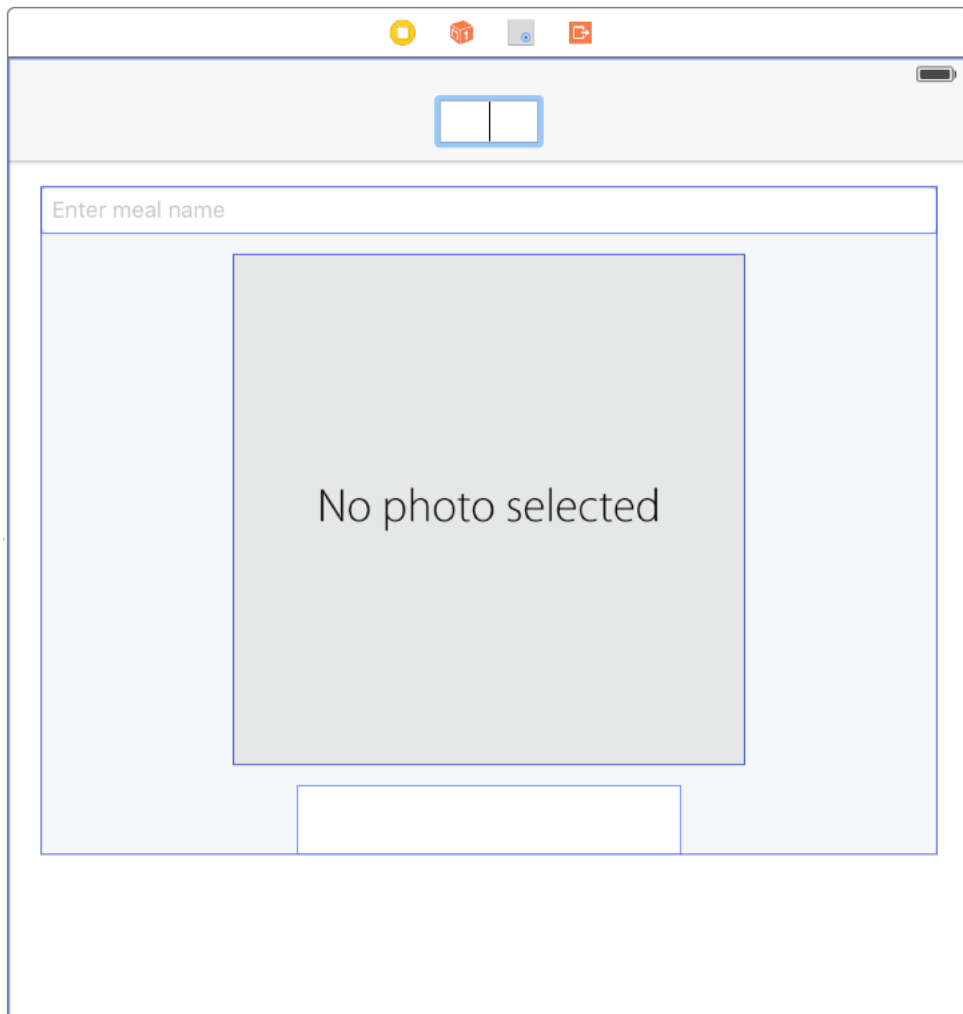
درست مانند قبل، Xcode یک navigation controller اضافه نموده و نوار پیمایش را در بالای meal scene (صفحه ی افزودن غذای جدید) به نمایش می گذارد. در گام بعدی، این نوار را طوری تنظیم می کنید که علاوه بر یک عنوان، دو دکمه نیز جهت Save و Cancel به صفحه محتوای جاری (scene) اضافه کند. بعده ها این دو کنترلر دکمه را به action های مربوطه (متدها یا عملیاتی که به محض کلیک بر روی دکمه های نام برده فراخوانی و اجرا می شوند) متصل خواهید نمود.



جهت ایجاد و تنظیم نوار پیمایش در meal scene، مراحل زیر را به ترتیب دنبال نمایید:

۱. بر روی navigation bar یا نوار پیمایش در meal scene دابل کلیک کنید.

آموزشگاه تحلیگر داده

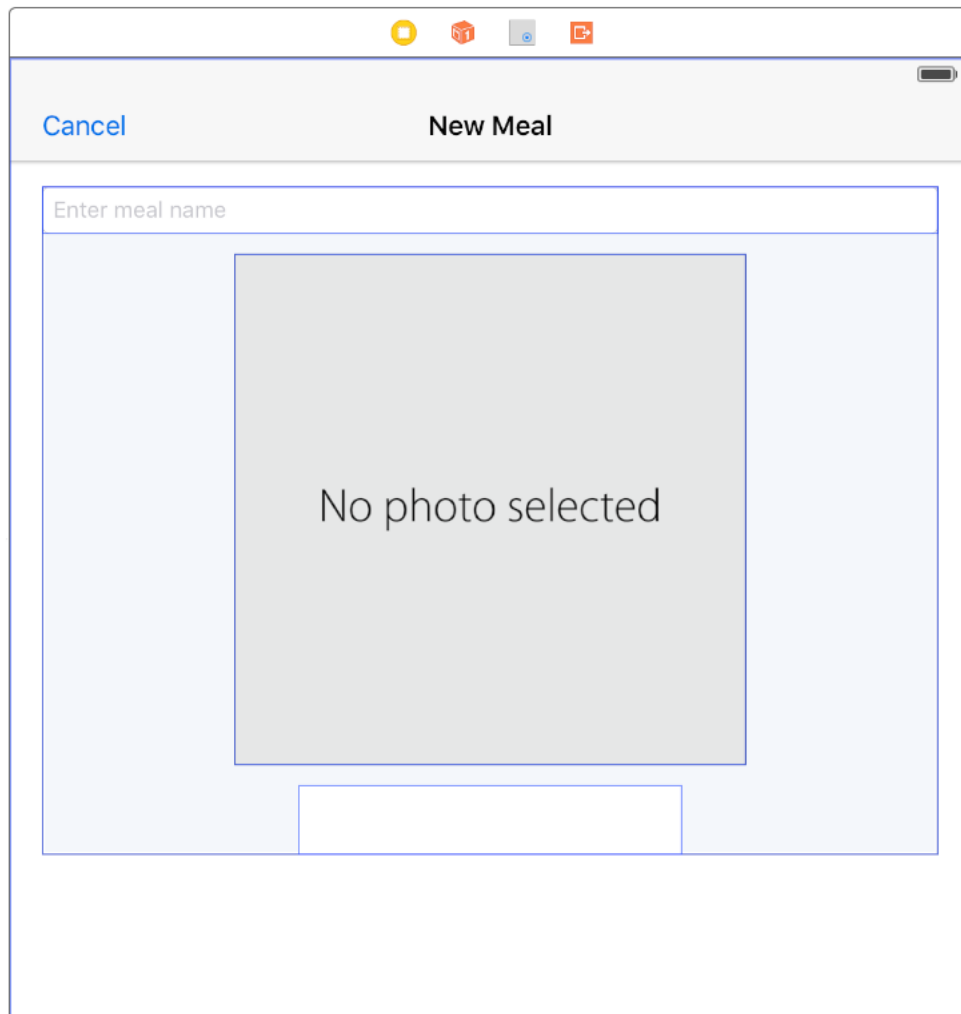


با کلیک بر روی نوار پیمایش، یک مکان نما پدیدار شده که به شما اجازه ی درج متن را می دهد.

۲. واژه ی New Meal را تایپ نموده و سپس کلید Return را فشار دهید.

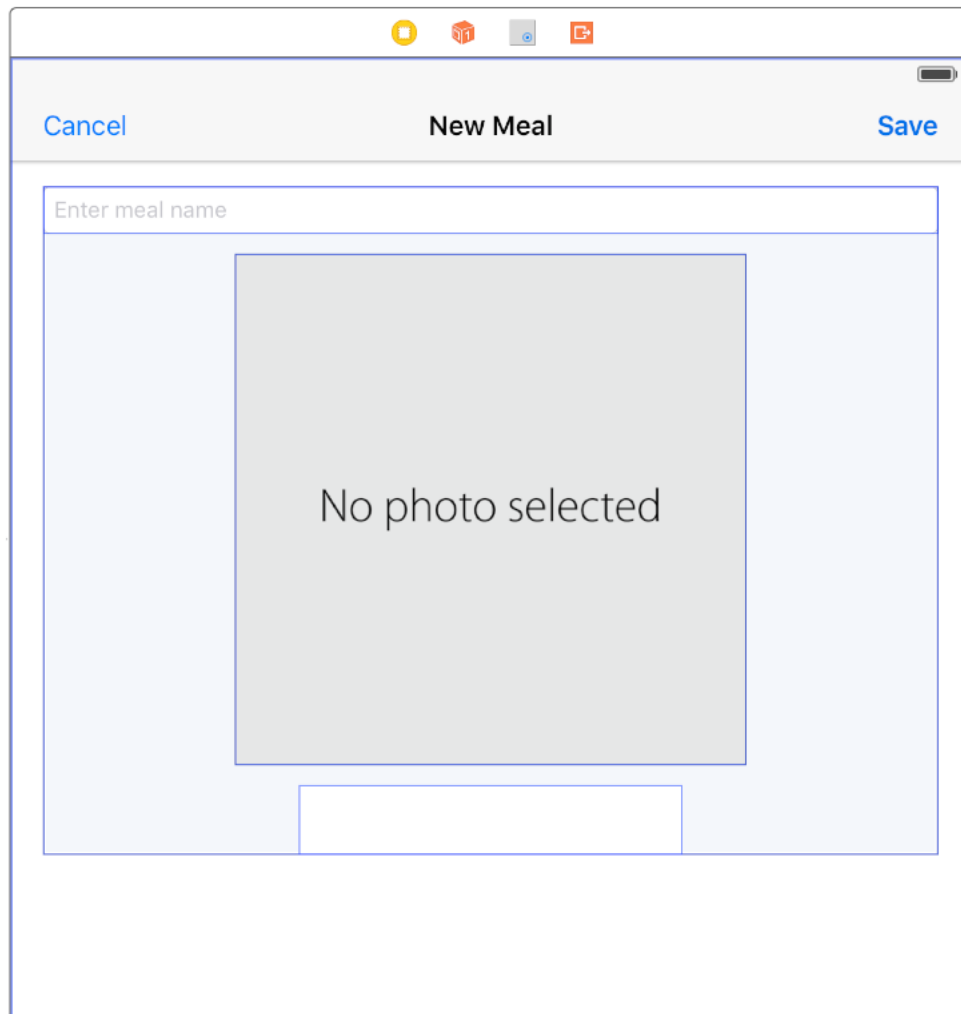
۳. حال یک آبجکت Bar Button Item از کادر Object Library انتخاب نمایید و آن را با اشاره گر موس به ناحیه ی سمت چپ نوار پیمایش مورد نظر کشیده (داخل meal scene)، در آنجا جایگذاری کنید.

۴. داخل کادر Attribute inspector، از فیلد System Item، گزینه ی Cancel را انتخاب نمایید. می بینید که متن دکمه به Cancel تغییر می کند.



۵. اکنون یک **Item** **Bar Button** دیگر از کادر **Object library** انتخاب کرده و به ناحیه ی سمت راست نوار پیمایش مورد نظر در **meal scene** بکشید و در آنجا جایگذاری کنید.

۶. داخل کادر **Attribute inspector**، از فیلد **System Item**، گزینه ی **Save** را انتخاب نمایید. می بینید که متن دکمه ی مورد نظر به **Save** تغییر می کند.



**تست کنید:** برنامه ی خود را اجرا نمایید. حال بر روی دکمه ی Add کلیک کنید. می بینید که همچنان meal scene در حال نمایش می باشد، اما این بار دیگر دکمه ای جهت بازگشت به meal list وجود ندارد – بجای آن دو دکمه ی Cancel و Save را مشاهده می کنید. همان طور که می دانید دو دکمه نام برده هنوز به هیچ action و عملیاتی وصل نیستند، از اینرو با کلیک بر روی آن ها اتفاق خاصی رخ نمی دهد. به زودی به این دکمه ها قابلیت ذخیره/عدم ذخیره ی غذای جدید و همچنین بازگرداندن کاربر به صفحه ی نمایش لیست غذاها (meal list scene) را اعطا خواهید کرد.



Carrier 9:41 AM

Cancel New Meal Save

Enter meal name

No photo selected

★ ★ ★ ★ ★

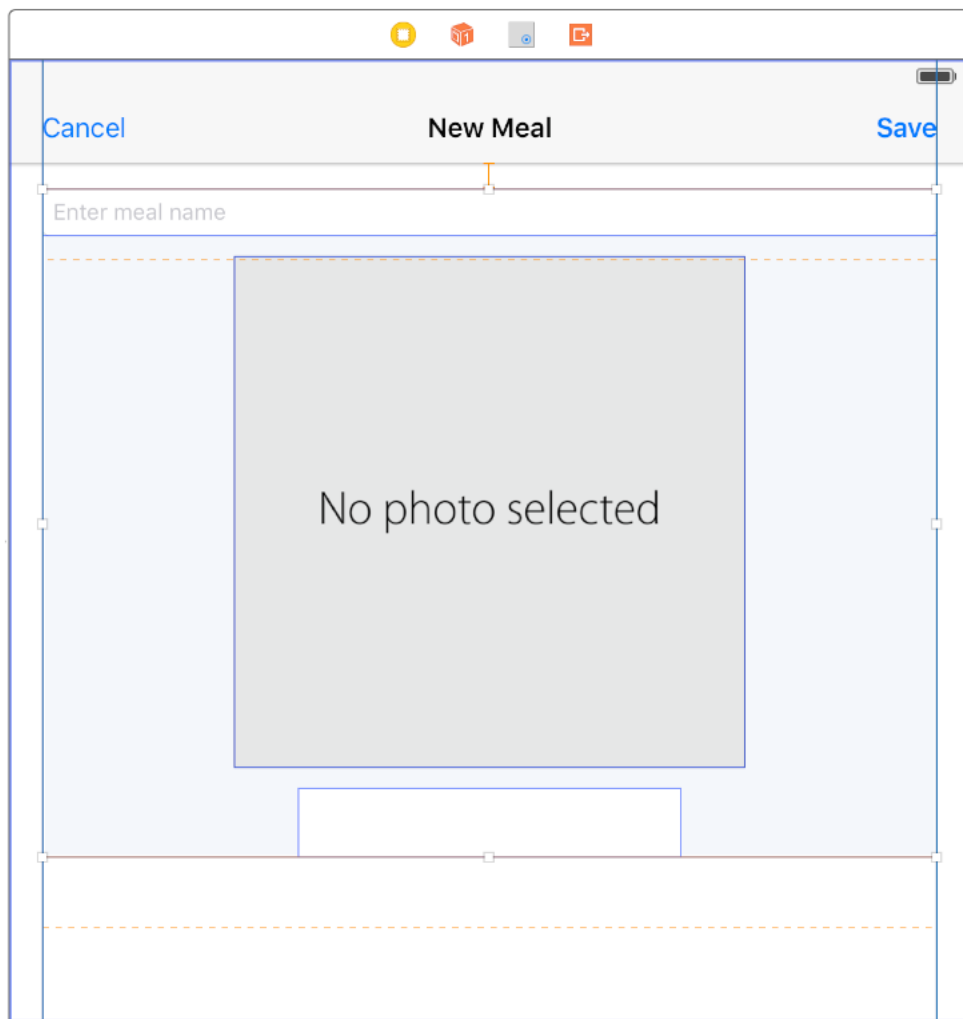
تکمیل ظاهر برنامه (UI) با بهره گیری موتور نمایش/مدیریت چیدمان المان ها Auto Layout

UI اصلی و اولیه ی اپلیکیشن خود را چندی پیش طراحی کردید. اما از زمان ایجاد ظاهر اصلی برنامه تا کنون، همان طور که طی آموزش های پیشین شاهد آن بودید، تغییرات زیادی رخ داده. در اینجا تغییرات بیشتری را به layout (قالب و طرح کلی) برنامه وارد خواهید کرد. بلکه صرفاً با بهره گیری از موتور نمایش و مدیریت چیدمان المان های UI، ابزار Auto layout محیط کاری Xcode، المان های رابط کاربری که به صورت پشته بر روی هم چیده شده اند (stack view) را تنظیم نموده و از نمایش صحیح آن ها برای کاربر اطمینان حاصل می کنید.

جهت بروز رسانی layout و چیدمان کلی stack view، مراحل زیر را گام به گام دنبال نمایید:

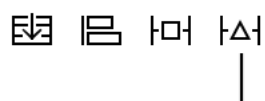
۱. داخل meal scene، با کلیک بر روی ناحیه ی آبی کم رنگ، stack view جاری را انتخاب نمایید.





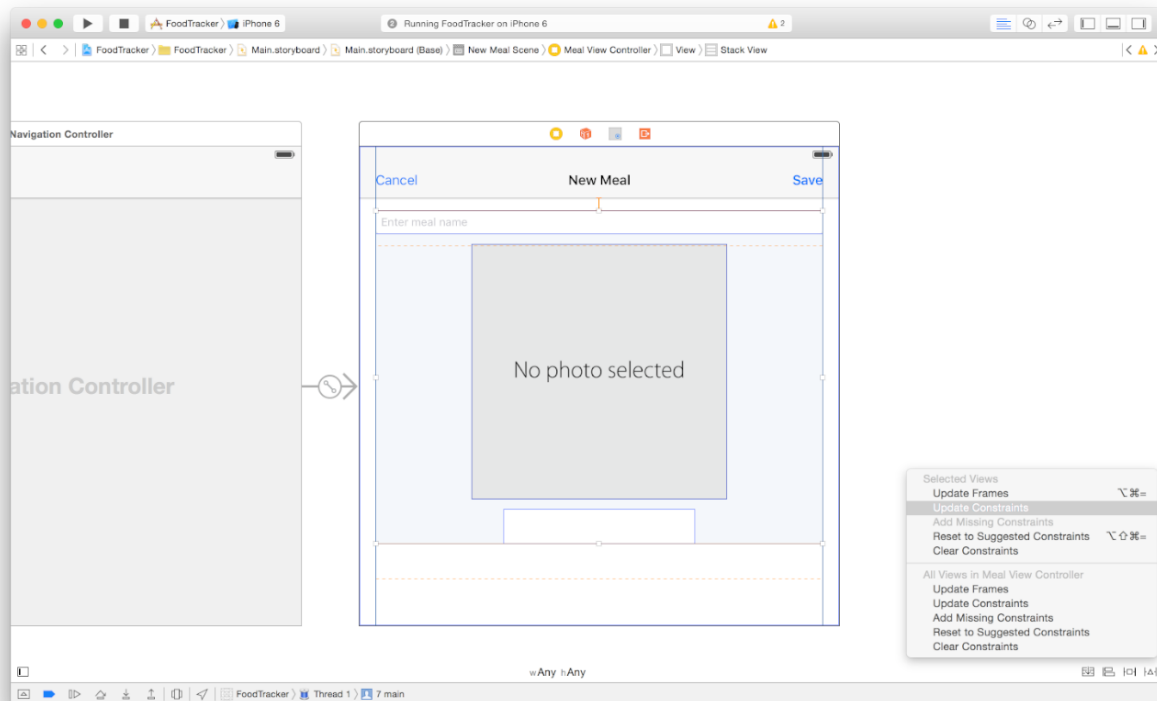
یا می توانید Stack View را از کادر Outline view انتخاب نمایید.

۲. در گوشه ی پایین، سمت راست canvas، منوی Resolve Auto Layout Issues را باز نمایید.



Resolve Auto Layout Issues

۳. از منوی مذکور، گزینه ی Update Constraints را انتخاب نمایید.



۴. المان ها در جای اولیه ی خود باقی می مانند، اما stack view اکنون بجای خط حاشیه ی بالای صفحه (top margin)، به نوار پیمایش view مورد نظر متصل شده است.

Constraint های meal scene و UI اکنون ظاهری مانند زیر خواهد داشت:

Cancel New Meal Save

Enter meal name

No photo selected

**تست کنید:** برنامه ی خود را اجرا کنید. فیلد درج متن (text field) و کنترل امتیازدهی به غذا ( rating control) هم اکنون می بایست نزدیکتر به نوار پیمایش باشند.

Carrier
9:41 AM

Cancel
New Meal
Save

Enter meal name

No photo selected

### ذخیره نمودن غذاهای جدید در لیست غذاها (Meal List)

آدرس آموزشگاه : تهران - خیابان شریعتی - بالاتر از خیابان ملک - جنب بانک صادرات - پلاک ۵۶۱ واحد ۷  
 شماره تماس : ۸۸۱۴۶۳۲۳ - ۸۸۴۴۶۷۸۰ - ۸۸۱۴۶۳۳۰

گام بعدی در افزودن قابلیت به برنامه ی FoodTracker و تکمیل امکانات آن، پیاده سازی امکان افزودن غذای جدید به لیست غذاهای برنامه است. به طور مشخص شما در نظر دارید، زمانی که کاربر اسم غذا، امتیاز و عکس دلخواه برای آن را در meal scene وارد کرده و دکمه ی Save را کلیک کرد، MealViewController (صفحه محتوای افزودن غذای جدید/meal scene) یک آبجکت Meal با اطلاعات مربوطه ایجاد (configure) کند و سپس آن اطلاعات را به MealTableViewController جهت نمایش در صفحه ی لیست غذاها (meal list) پاس دهد.

برای نیل به این هدف، ابتدا یک property به نام Meal در MealViewController تعریف می کنید.

به منظور افزودن متغیر Meal به MealViewController، مراحل زیر را دنبال نمایید:

۱. فایل MealViewController.swift را باز کنید.

۲. در زیر متغیر (outlet) ratingControl در فایل MealViewController.swift، متغیر

(property) زیر را وارد نمایید:

```
/*
This value is either passed by `MealTableViewController` in
`prepareForSegue(_:sender:)`
or constructed as part of adding a new meal.
*/
var meal: Meal?
```

این دستور یک property در MealViewController تعریف می کند که این

property یک Meal از نوع optional است. بدین معنی که می تواند در هر برهه

ای از زمان هیچ مقداری نداشته و به اصطلاح nil باشد.

قاعداً شما تنها زمانی می خواهید آبجکت Meal ساخته شده و پاس داده شود که دکمه

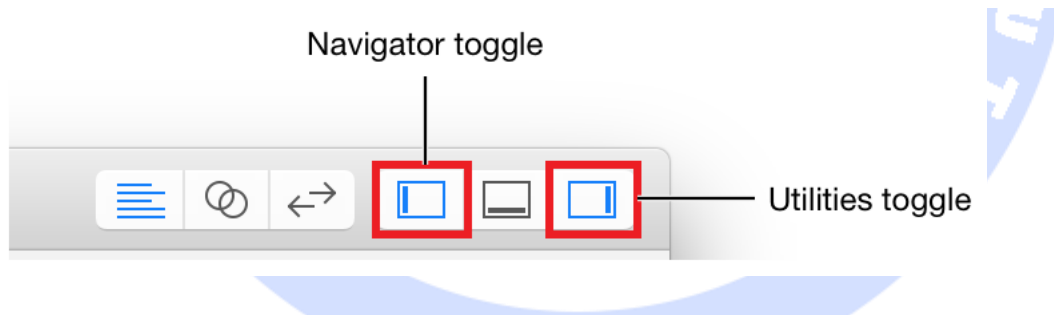
ی Save کلیک شده باشد. برای اینکه بتوان تشخیص داد چه زمانی این اتفاق رخ داده

(دکمه توسط کاربر فشرده شده)، دکمه ی Save را در قالب یک outlet (با افزودن خصیصه ی @IBAction) در فایل MealViewController.swift تعریف می کنید. به منظور متصل کردن دکمه ی Save به کد موجود در MealViewController، مسیر زیر را طی نمایید:

۱. فایل storyboard را باز نمایید.

۲. بر روی دکمه ی Assistant در نوار ابزار Xcode کلیک نموده تا ویرایشگر کمکی (assistant editor) محیط به نمایش درآید.

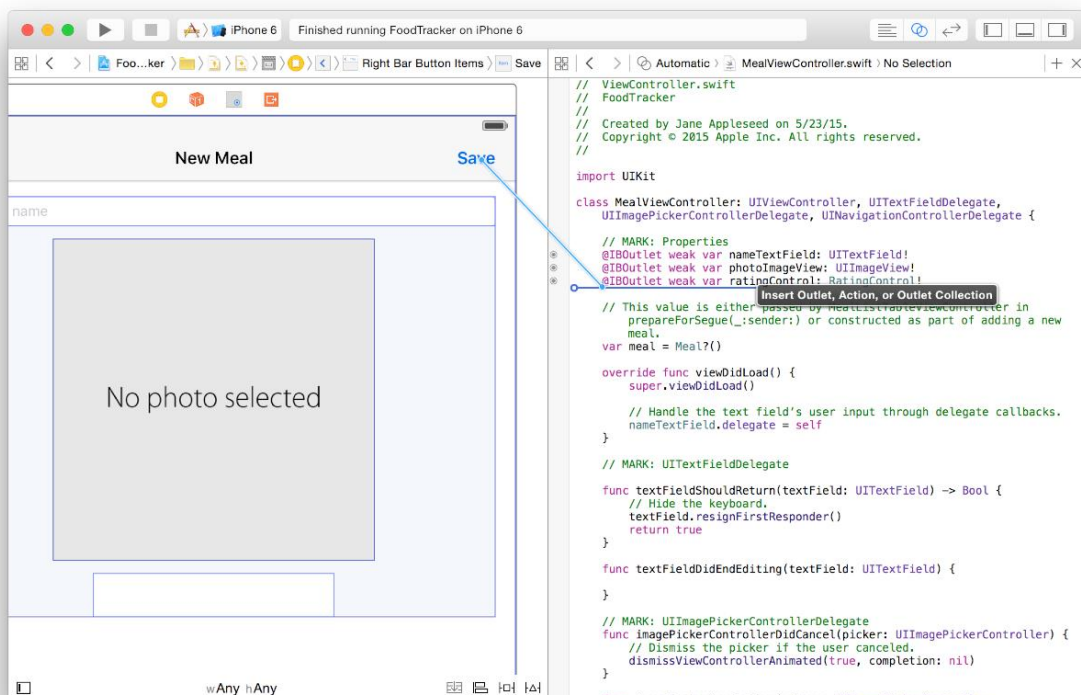
۳. در صورت نیاز به فضای کاری بیشتر، کادر Project navigator و Utility area را با کلیک بر روی دکمه های مربوطه در نوار ابزار Xcode، پنهان نمایید.



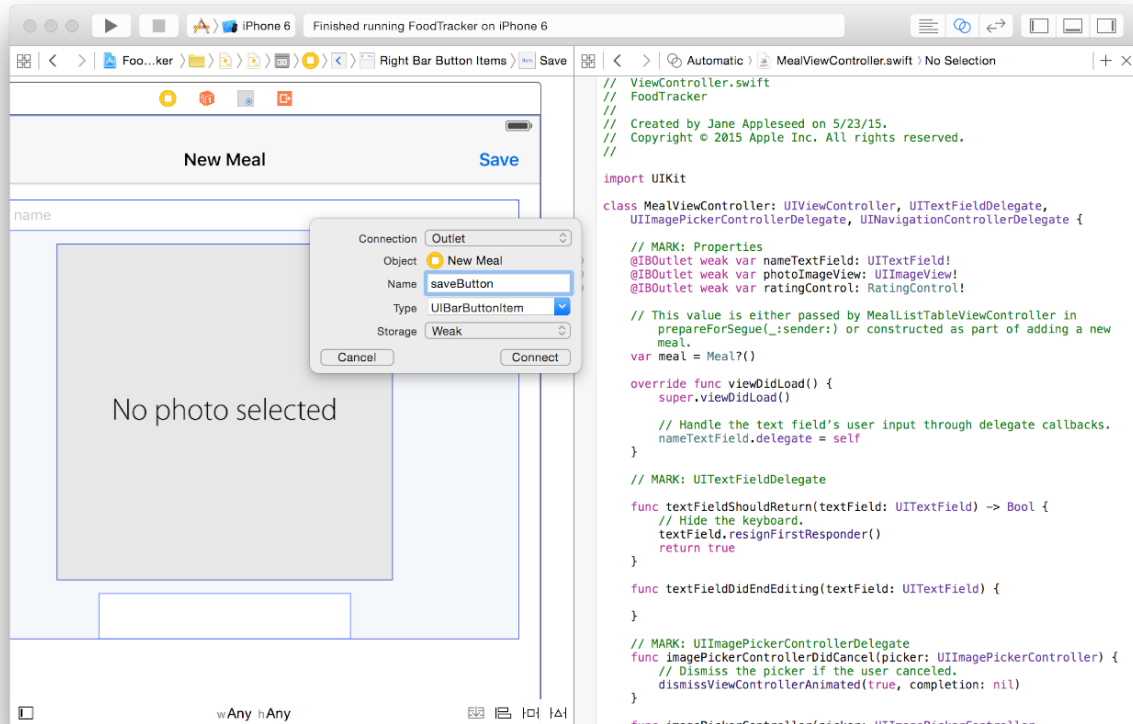
۴. در Storyboard، دکمه ی Save را انتخاب نمایید.

۵. دکمه ی Save را با اشاره گر موس (و فشردن کلید control) بر روی ناحیه ی ویرایش کد (داخل ویرایشگر کمکی و فایل MealViewController.swift) کشیده و آن را در زیر متغیر ratingControl جایگذاری نمایید.





۶. داخل کادر محاوره ای که نمایان می شود، واژه ی saveButton را در کادر Name وارد نمایید. لازم به انجام تنظیمات دیگری نیست.



۷. حال دکمه ی Connect را کلیک کنید.

تعریف قابلیت پیمایش به عقب (unwind Segue)

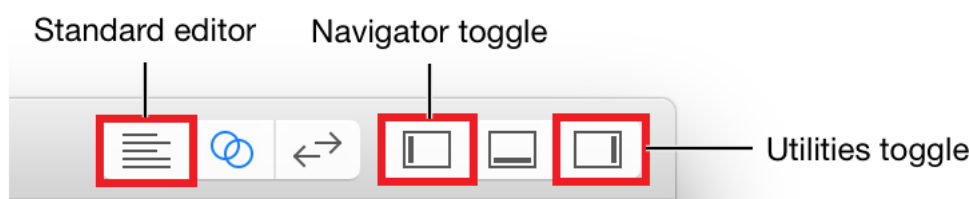
اکنون کاری که باید انجام دهید عبارت است از ارسال آبجکت Meal به MealTableViewController، زمانی که کاربر دکمه ی Save را فشار می دهد و دور انداختن یا حذف آن به هنگام کلیک کاربر بر روی دکمه ی Cancel. در هر دو مورد صفحه ی مورد نمایش باید از meal scene (صفحه ی افزودن غذای جدید) به meal list scene (صفحه نمایش لیست غذاها) تغییر کند.

برای انجام این کار می بایست از یک unwind segue بهره بگیرید. Unwind segue یک یا چند segue را به عقب پیمایش کرده و کاربر را به یک نمونه از view controller باز می گرداند. به عبارت دیگر شما به وسیله ی unwind segue، پیمایش به عقب (reverse navigation) یا صفحات قبلی برنامه را پیاده سازی می کنید.

هر زمان که یک segue فعال می شود، به دنبال آن مکانی در قالب یک متد برای شما ایجاد می شود که می توانید در بدنه ی آن کد خود را اضافه نموده و انتظار اجرای آن را داشته باشید. اسم این متد `prepareForSegue(_:sender:)` بوده و این امکان را به شما می دهد تا داده های مورد نیاز را ذخیره کنید و هر گونه عملیات `cleanup` لازم را در `source view controller` (view controller مبدا) که segue از آن آغاز می شود، انجام دهید. برای رسیدن به این هدف، متد ذکر شده را در `MealViewController` پیاده سازی خواهید کرد.

به منظور پیاده سازی متد `prepareForSegue(_:sender:)` در `MealViewController`، مراحل زیر را دنبال نمایید:

۱. با کلیک بر روی دکمه ی Standard (اشاره شده در تصویر زیر)، ویرایشگر اصلی (Standard editor) محیط را باز نمایید.



۲. فایل `MealViewController.swift` را باز کنید.
۳. در فایل `MealViewController.swift`، بالای بخش `Actions` MARK: // کد زیر را درج نمایید:

**// MARK: Navigation**

۴. این خط صرفاً یک `comment` است که شرحی در خصوص مورد کاربرد کد ارائه داده و به شما (یا هر شخص دیگری که کد شما را می خواند) اعلان می کند که متد درج شده در زیر آن مربوط به قابلیت پیمایش (navigation flow) در برنامه می باشد.

**// MARK: Navigation**

۵. در زیر `comment`، تعریف متد را به صورت زیر وارد نمایید:

```
// This method lets you configure a view controller before it's presented.
override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
}
```

۶. داخل بدنه ی متد (prepareForSegue(\_:sender:)) دستور شرطی زیر را درج نمایید:

```
if saveButton === sender {
}
```

این کد با استفاده از عملگر "===" بررسی می کند آیا آبجکت مورد اشاره ی متغیر saveButton (outlet) همان instance object /نمونه کلاس sender (فراخواننده ی action) است یا خیر (هر دو به یک نمونه کلاس اشاره دارند یا خیر). در صورتی که این امر صادق بود، دستور if اجرا خواهد شد. داخل ساختمان if، کد زیر را وارد نمایید:

```
let name = nameTextField.text ?? ""
let photo = photoImageView.image
let rating = ratingControl.rating
```

این کد سه ثابت از property یا خصوصیت های selected image، text و rating موجود در scene مورد نظر ایجاد می کند. حال توجه خود را به عملگر "???" در خط name جلب نمایید. این عملگر برای این استفاده شده که بررسی کند اگر متغیر از نوع optional مقداری داشت، آن مقدار را برگرداند و در غیر این صورت یک مقدار پیش فرض را بازگردانی نماید. nameTextField.text (یک متغیر optional است چرا که text field می تواند متن داشته یا نداشته باشد) یک مقدار برمی گرداند. در اینجا عملگر "???", مقدار بازگشتی توسط nameTextField.text را استخراج نموده (unwrap) و در دسترس قرار می دهد. اگر مقدار بازگشتی یک رشته ی مجاز بود، آن را برمی گرداند و در غیر این

صورت، مقدار پیش فرض که همان رشته ی تهی ("") است را در خروجی بازگردانی می کند.

۷. داخل ساختمان دستور شرطی if، کد زیر را وارد نمایید:

```
// Set the meal to be passed to MealTableViewController after the unwind segue.
```

```
meal = Meal(name: name, photo: photo, rating: rating)
```

این کد، متغیر meal را پیش از اجرای segue، با مقادیر مناسب تنظیم می کند.

در حال حاضر، پیاده سازی متد prepareForSegue(\_:sender:) شما باید به صورت زیر باشد:

```
// This method lets you configure a view controller before it's presented.
```

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
```

```
if saveButton === sender {
```

```
let name = nameTextField.text ?? ""
```

```
let photo = photoImageView.image
```

```
let rating = ratingControl.rating
```

```
// Set the meal to be passed to MealTableViewController after the unwind segue.
```

```
meal = Meal(name: name, photo: photo, rating: rating)
```

```
}
```

```
}
```

گام بعدی در پیاده سازی پیمایش به عقب (ایجاد یک unwind segue)، افزودن یک action method به destination view controller (view controller مقصد) است. این متد باید با attribute یا خصیصه ی IBAAction علامت گذاری شده و به عنوان پارامتر خود یک segue (UIStoryboardSegue) را بپذیرد. از آنجایی که شما می خواهید به صفحه ی نمایش لیست غذاها برگردید (meal list scene)، بایستی متدی با این فرمت را به فایل MealTableViewController.swift اضافه نمایید.

داخل بدنه ی این متد، منطقی را که غذای جدید (ارسال شده از MealViewController که source view controller هست) را به داده های جاری لیست غذاها افزوده و سپس

یک سطر جدید به نمای جدولی (table view) در صفحه نمایش لیست غذاها (meal list scene) اضافه می کند، پیاده سازی خواهید نمود.

به منظور افزودن action method به کد موجود در فایل MealTableViewController، مراحل زیر را دنبال نمایید:

۱. فایل MealTableViewController.swift را باز کنید.

۲. در فایل MealTableViewController.swift، درست قبل از آخرین ({}), دستور زیر را

اضافه نمایید:

```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
}
```

۳. داخل بدنه ی متد unwindToMealList(\_:) دستور شرطی if را اضافه نمایید:

```
if let sourceViewController = sender.sourceViewController as?
MealViewController, meal = sourceViewController.meal {
}
```

در شرط تعریف شده توسط این دستور if، اتفاقات زیادی می افتد. این کد از عملگر downcast (as?) و تبدیل آزمایشی source view controller از segue مورد نظر به نوع MealViewController بهره می گیرد. انجام downcast ضروری است چرا که sender.sourceViewController از جنس کلاس UINavigationController است، اما شما مجبورید که با MealViewController کار کنید (downcast عبارت است از تبدیل یک آبجکت به یکی از کلاس های مشتق شده از آن).

عملگر مورد نظر یک مقدار optional برمی گرداند که در صورت ناموفق بودن downcast آن مقدار nil خواهد بود. اما چنانچه downcast با موفقیت صورت پذیرفت، آنگاه کد view controller مورد نظر را به ثابت (constant) محلی sourceViewController تخصیص می دهد و همچنین بررسی می کند که متغیر

meal در sourceViewController دارای مقدار nil هست یا خیر. اگر متغیر meal دارای مقدار و به اصطلاح non-nil بود، کد مورد نظر مقدار آن متغیر (property) را به ثابت محلی meal اختصاص داده و دستور داخل بدنه ی if را اجرا می کند.

در صورتی که downcast با شکست مواجه شده یا متغیر تعریف شده در sourceViewController دارای مقدار nil بود، آنگاه شرط صادق نبوده (نتیجه ی آن false بوده) و به تبع آن دستور شرطی if اجرا نمی شود.

۴. در بدنه ی دستور if، کد زیر را درج نمایید:

// Add a new meal.

```
let newIndexPath = NSIndexPath(forRow: meals.count, inSection: 0)
```

این کد محل جدید قرار گیری خانه ی جدول که نمایشگر (حاوی اطلاعات) غذای جدید است را در نمای جدولی (table view) محاسبه نموده و سپس آن را در ثابت (constant) محلی به نام newIndexPath ذخیره می کند.

۵. داخل بدنه ی if، در زیر خط قبلی، کد زیر را اضافه نمایید:

```
meals.append(meal)
```

این دستور، غذای جدید را به لیست غذاهای جاری، کپسوله سازی (ذخیره) شده در data model، اضافه می کند.

۶. حال دستور زیر را در زیر خط قبلی، داخل ساختمان if، درج نمایید:

```
tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation: .Bottom)
```

این دستور فرایند اضافه شدن سطر جدید به نمای جدولی (table view) را در خانه ای که حاوی اطلاعات مربوط به غذای جدید می باشد، با انیمیشن به اجرا در می آورد. پارامتر Bottom. سبب می شود که سطر جدید از پایین صفحه به آن اضافه شود.

به زودی این متد را با پیاده سازی پیچیده تری خواهید نوشت. در حال حاضر بدنه ی متد `unwindToMealList(_)` می بایست به شکل زیر باشد:

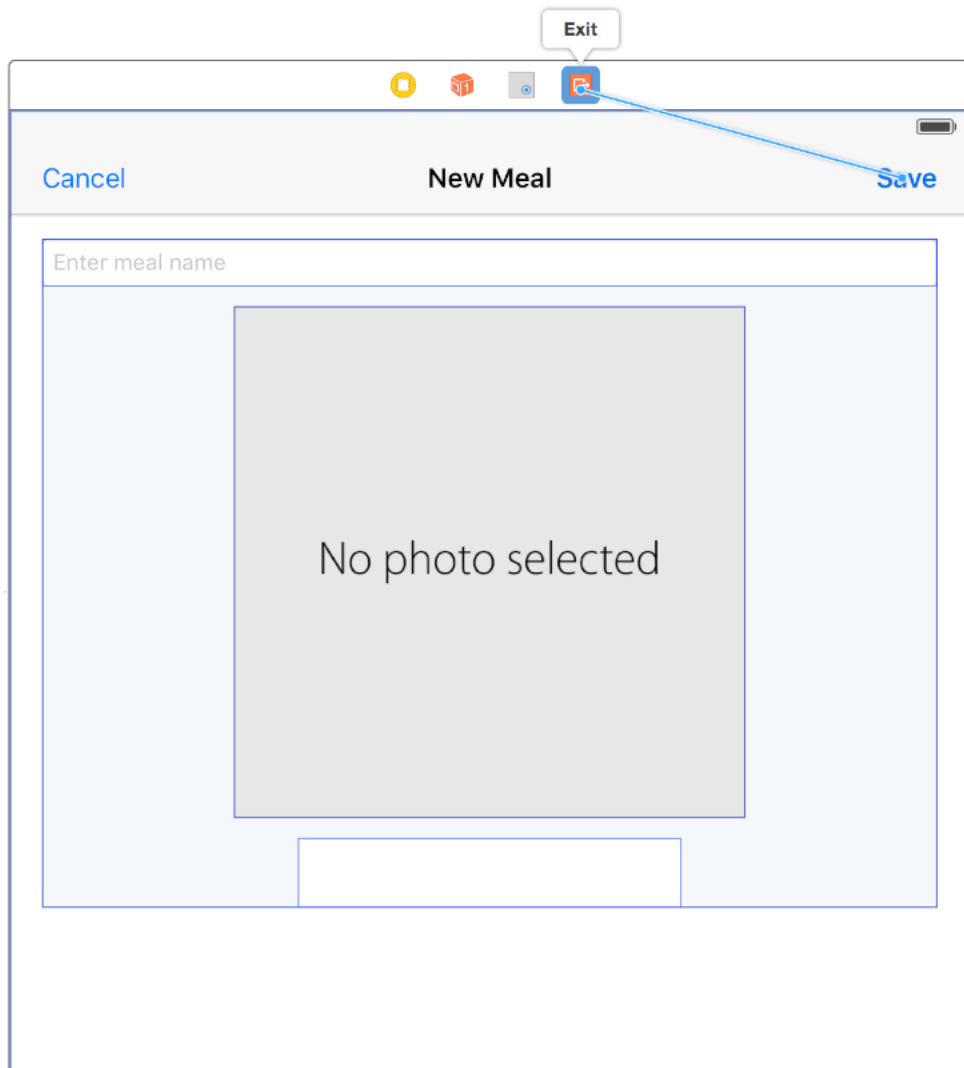
```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
    if let sourceViewController = sender.sourceViewController as?
    MealViewController, meal = sourceViewController.meal {
        // Add a new meal.
        let newPath = NSIndexPath(forRow: meals.count, inSection: 0)
        meals.append(meal)
        tableView.insertRowsAtIndexPaths([newIndexPath],
        withRowAnimation: .Bottom)
    }
}
```

اکنون می بایست segue `unwind` را که قرار است action مورد نظر را فراخوانی کرده و سبب اجرای آن شود، ایجاد نمایید.

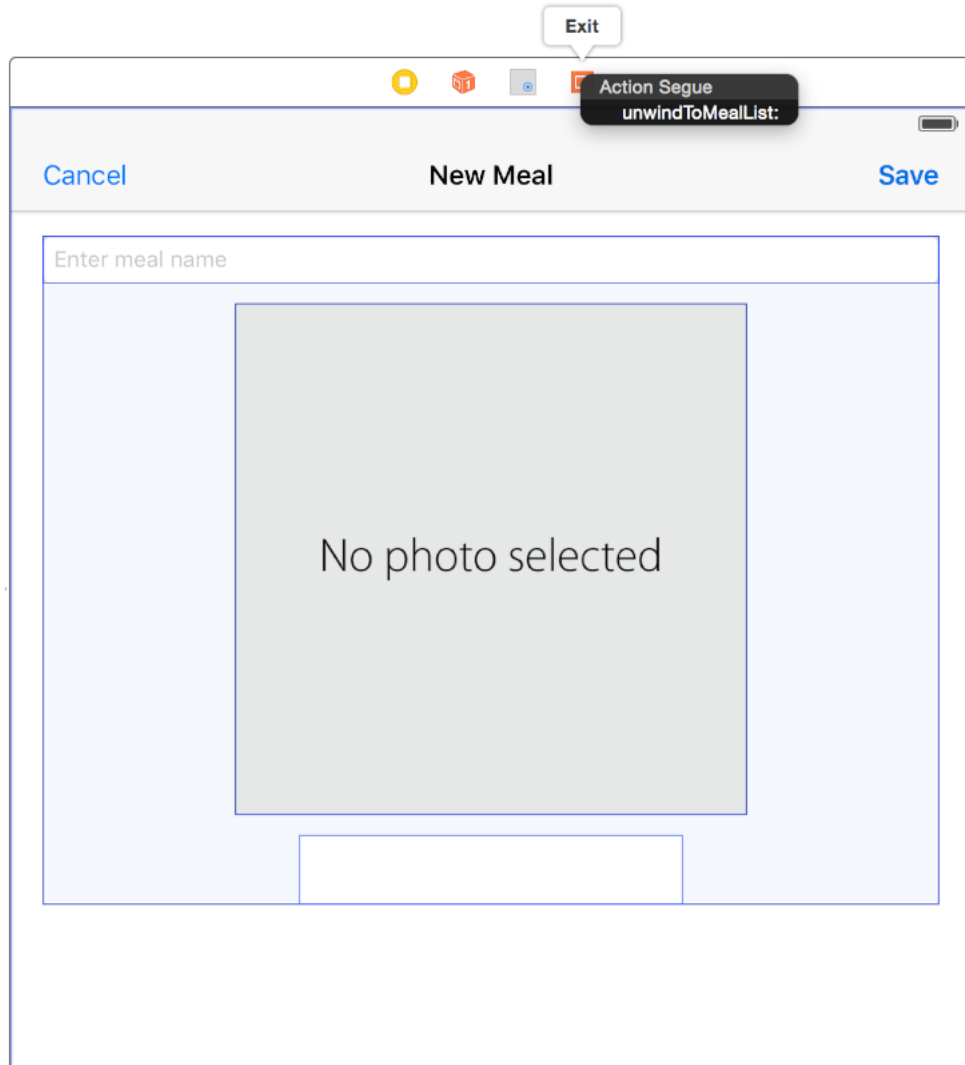
جهت متصل کردن دکمه ی Save به متد `unwindToMealList`، مراحل زیر را دنبال نمایید:

۱. ابتدا Storyboard را باز نمایید.
۲. در سطح canvas، کلید control را نگه داشته و با اشاره گر موس دکمه ی Save را به سمت آیتم Exit در بالای صفحه محتوای جاری (meal scene) بکشید و در آنجا جایگذاری نمایید.





یک منو در مکانی که دکمه ی Save را در آن جایگذاری نمودید، پدیدار می شود.



۳. گزینه ی `unwindToMealList:` را از منوی میانبر انتخاب نمایید.

اکنون، زمانی که کاربران بر روی دکمه ی `Save` کلیک می کنند، برنامه کاربر را به صفحه نمایش لیست غذاها باز می گرداند. این کار با فراخوانی متد `unwindToMealList(_:)` صورت می پذیرد.

**تست کنید:** برنامه را اجرا نمایید. اکنون زمانی که بر روی دکمه ی `(+)` کلیک می کنید، سپس یک غذای جدید ایجاد نموده و به دنبال آن دکمه ی `Save` را فشار می دهید، باید غذای جدید را در لیست نمایش غذاها مشاهده نمایید.

مهم: در صورت عدم مشاهده ی متد `unwindToMealList` در منوی میانبر، باید اطمینان حاصل نمایید که signature (اسم، تعداد و نوع پارامترهای ورودی) متد کاملاً صحیح است.

@IBAction func unwindToMealList(sender: UIStoryboardSegue)

غیر فعال نمودن قابلیت ذخیره (دکمه ی Save) زمانی که کاربر اسم آیتم را وارد نکند حالا اگر کاربر سعی کند بدون وارد کردن اسم غذا، آن را ذخیره کند، چه اتفاقی رخ می دهد؟ از آنجایی که متغیر meal در `MealViewController` از نوع optional می باشد و همچنین شما متد سازنده (initializer) خود را طوری تنظیم کردید که در صورت عدم وجود اسم برای غذا (در مقداری اولیه) با شکست مواجه شود، آبجکت `Meal` ایجاد نشده و به لیست غذاها اضافه نمی شود – شما هم دقیقاً همین را انتظار دارید. شما می توانید این کد را کمی پیچیده تر نمایید: برای مثال می توانید با غیر فعال کردن دکمه ی Save، هنگامی که کاربر در حال وارد کردن اسم برای غذای جدید است، مانع از این شوید که کاربر سهواً بدون وارد کردن اسم غذا، آن را ذخیره کند یا بررسی کنید اسم وارد شده برای غذا کاملاً صحیح است و بعد اجازه ی بستن صفحه کلید را به کاربر بدهید. جهت غیر فعال سازی دکمه ی Save، زمانی که اسمی برای آن وارد نشده، مراحل زیر را دنبال نمایید:

۱. در فایل `MealViewController.swift`، ابتدا بخش MARK: //

`UITextFieldDelegate` را پیدا کنید.

یادآور می شویم که با استفاده از ابزار `menu functions` می توانید به سرعت به بخش مورد نظر در کد برنامه پیمایش کنید (بپرید). برای دسترسی به این ابزار کافی است بر روی اسم فایل در بالای ناحیه ی ویرایش (`editor area`) کلیک نمایید.

۲. در این بخش، یک متد `UITextFieldDelegate` دیگر اضافه نمایید.

`func textFieldDidBeginEditing(textField: UITextField) {`

// Disable the Save button while editing.

```
saveButton.enabled = false
```

```
}
```

متد `textFieldDidBeginEditing` زمانی صدا خورده می شود که وضعیت ویرایش شروع شده (`editing session`) یا زمانی که صفحه کلید به نمایش در می آید. این کد دکمه ی `Save` را هنگامی که کاربر در حال ویرایش متن داخل `text field` است، غیر فعال می سازد.

۳. در زیر متد `(_:) textFieldDidBeginEditing`، متد دیگری به صورت زیر اضافه

نمایید:

```
func checkValidMealName() {
```

// Disable the Save button if the text field is empty.

```
let text = nameTextField.text ?? ""
```

```
saveButton.enabled = !text.isEmpty
```

```
}
```

این متد، یک تابع کمکی است که در صورت تهی بودن `text field` دکمه ی `Save` را غیر فعال می سازد.

۴. متد `(_:) textFieldDidEndEditing` را پیدا کنید:

```
func textFieldDidEndEditing(textField: UITextField) {
```

```
}
```

بدنه ی این متد در حال حاضر می بایست تهی باشد.

۵. دستورات زیر را در آن وارد نمایید:

```
checkValidMealName()
```

```
navigationItem.title = textField.text
```

اولین خط با فراخوانی متد `checkValidMealName()` بررسی می کند آیا `text field` داخل خود متنی دارد یا خیر و در صورت وجود متن در فیلد مزبور، دکمه ی `Save` را فعال می سازد. خط دوم عنوان صفحه (`title` صفحه محتوا یا `scene` جاری) را برابر متن وارد شده در `text field` قرار می دهد.

۶. حال متد `(_) viewDidLoad` را در کد پیدا کنید:

```
override func viewDidLoad() {
```

```
super.viewDidLoad()
// Handle the text field's user input through delegate callbacks.
nameTextField.delegate = self
}
```

۷. متد `checkValidMealName()` را فراخوانی نموده تا مطمئن شوید دکمه ی Save تا زمانی که کاربر یک اسم مجاز در text field وارد نکرده، غیر فعال می ماند:

```
// Enable the Save button only if the text field has a valid Meal name.
checkValidMealName()
می بایست ظاهری مشابه زیر داشته باشد: viewDidLoad() در حال حاضر بدنه ی متد
override func viewDidLoad() {
super.viewDidLoad()
// Handle the text field's user input through delegate callbacks.
nameTextField.delegate = self
// Enable the Save button only if the text field has a valid Meal name.
checkValidMealName()
}
```

متد `(_: textFieldDidEndEditing)` نیز می بایست دارای پیاده سازی زیر باشد:

```
func textFieldDidEndEditing(textField: UITextField) {
checkValidMealName()
navigationItem.title = textField.text
}
```

**تست کنید:** برنامه ی خود را اجرا کنید. حال هنگامی که شما بر روی دکمه ی (+) کلیک می کنید،

دکمه ی Save غیر فعال شده و تا زمانی که یک اسم مجاز در text field وارد نکرده و صفحه کلید را معاف نکرده باشید، این دکمه غیر فعال می ماند.

Carrier 9:41 AM

Cancel New Meal Save

Enter meal name

No photo selected

q w e r t y u i o p  
a s d f g h j k l  
↑ z x c v b n m ↵  
123 😊 space Done

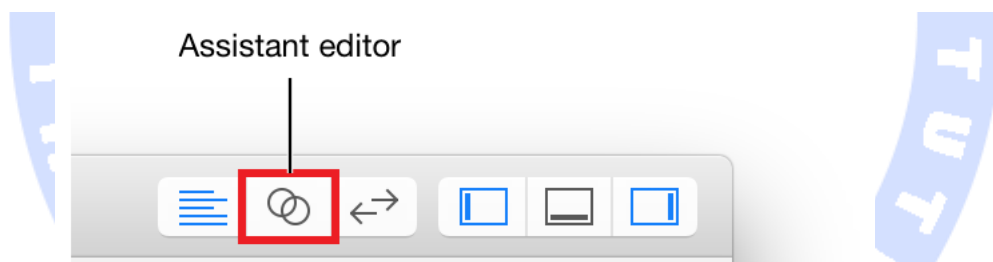
## پیاده سازی قابلیت لغو فرایند اضافه و ذخیره کردن آیتم جدید (پیاده سازی رفتار مربوط به دکمه ی Cancel)

یک کاربر ممکن است مایل باشد نظر خود را در مورد افزودن یک غذای جدید به لیست تغییر داده و آن را لغو کند. در واقع بدون افزودن آیتم جدید و ذخیره ی آن، به صفحه ی meal list (لیست غذاها) بازگردد. برای این منظور می بایست رفتار دکمه ی Cancel را پیاده سازی نمایید.

به منظور ایجاد و پیاده سازی یک action method که به دکمه ی Cancel متصل باشد، مراحل زیر را دنبال نمایید:

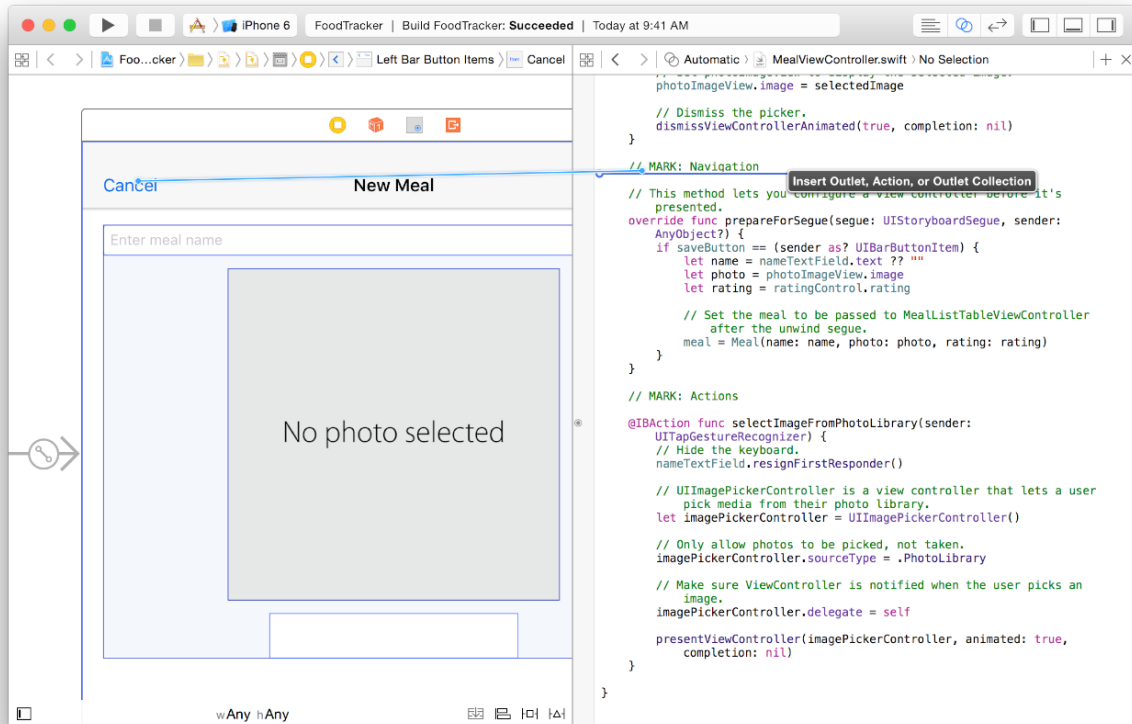
۱. ابتدا storyboard را باز نمایید.

۲. بر روی دکمه ی Assistant در نوار ابزار Xcode کلیک کرده تا ویرایشگر کمکی محیط (assistant editor) باز شود.



۳. داخل storyboard، دکمه ی Cancel را انتخاب نمایید.

۴. بر روی دکمه ی Cancel کلیک کرده، کلید control را نگه دارید. سپس آن را از سطح canvas به ناحیه ی ویرایش کد در سمت راست محیط (داخل فایل MealViewController.swift)، زیر خط MARK: // بکشید و در آنجا جایگذاری نمایید.



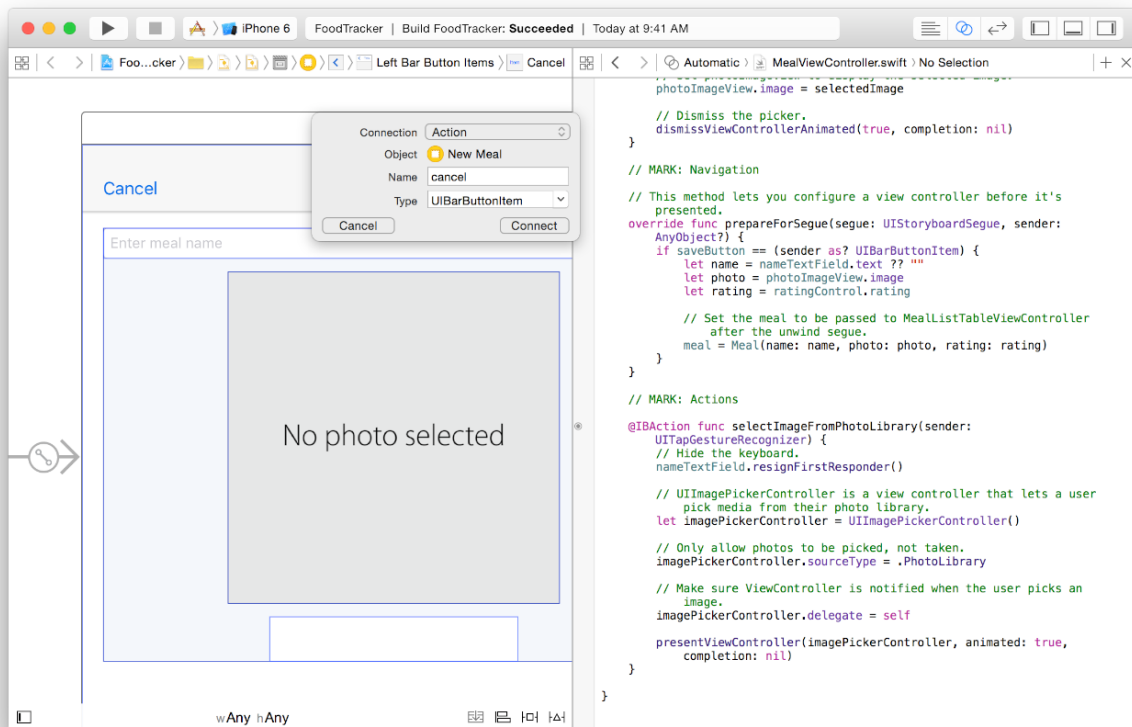
۵. در کادر محاوره ای که نمایان می شود، از فیلد Connection، گزینه ی Action را انتخاب نمایید.

۶. داخل فیلد Name، واژه ی cancel را وارد نمایید.

۷. از فیلد Type، گزینه ی UIBarButtonItem را انتخاب نمایید.

لازم به اعمال تغییرات دیگری در این کادر محاوره ای نیست. در حال حاضر این کادر می بایست ظاهری مشابه زیر داشته باشد:





#### ۸. دکمه ی Connect را کلیک نمایید.

Xcode خود کد لازم را ویژه ی action مورد نظر به فایل MealViewController.swift اضافه می کند.

```
@IBAction func cancel(sender: UIBarButtonItem) {
}
```

#### ۹. داخل بدنه ی متد cancel(\_:)، این دستور را اضافه نمایید:

```
dismissViewControllerAnimated(true, completion: nil)
```

این کد صفحه ی افزودن غذای جدید (meal scene) را بدون ذخیره ی اطلاعات جدید، بسته و کاربر را به صفحه ی نمایش لیست غذاها (meal list scene) هدایت می کند.

کد موجود در ساختمان متد cancel(\_:) هم اکنون می بایست به صورت زیر باشد:

```
@IBAction func cancel(sender: UIBarButtonItem) {
    dismissViewControllerAnimated(true, completion: nil)
}
```

**تست کنید:** برنامه ی خود را اجرا نمایید. اکنون زمانی که شما دکمه ی (+) را فشار داده و سپس بجای دکمه ی Save بر روی Cancel کلیک می کنید، برنامه می بایست بدون ذخیره ی اطلاعات یک آیتم جدید، شما را به صفحه ی meal list هدایت کند.

## درس ۱۰ : آموزش Edit و Delete در Swift

### پیاده سازی قابلیت حذف و ویرایش

در این بحث رفتاری به برنامه اضافه می کنید که به کاربر اجازه می دهد اطلاعات مربوط به غذاهای فهرست شده در برنامه ی FoodTracker را ویرایش کرده و در صورت لزوم حذف نماید.

### آنچه خواهید آموخت

۱. فرق بین پیمایش (navigation) به صورت push و پیمایش به صورت modal را درک کنید.

۲. View controller ها را با توجه به سبک نمایش آن ها (presentation style)، از نمایشگر پنهان نمایید.

۳. بدانید برای downcast (تبدیل نوع از کلاس والد به یکی از کلاس های مشتق شده از آن) چه موقع از کدام عملگر تبدیل نوع (type cast operator) بهره بگیرید.

۴. با بهره گیری از optional binding شرایط پیچیده را بررسی کنید.

۵. با استفاده از اسم تخصیص داده شده به segue ها (segue identifier)، تشخیص دهید کدام segue در حال اجرا است.

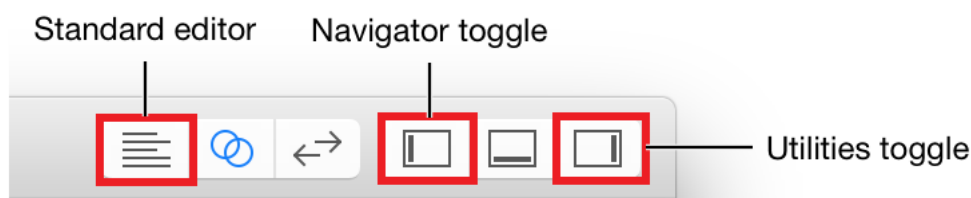
### پیاده سازی امکان ویرایش غذاهای جاری

در حال حاضر، برنامه ی FoodTracker به کاربران خود این امکان را می دهد تا یک آیتم جدید را به لیست غذاها اضافه کنند. در مرحله ی بعدی، قابلیت ویرایش اطلاعات غذای جاری را به کاربر اعطا خواهید نمود.

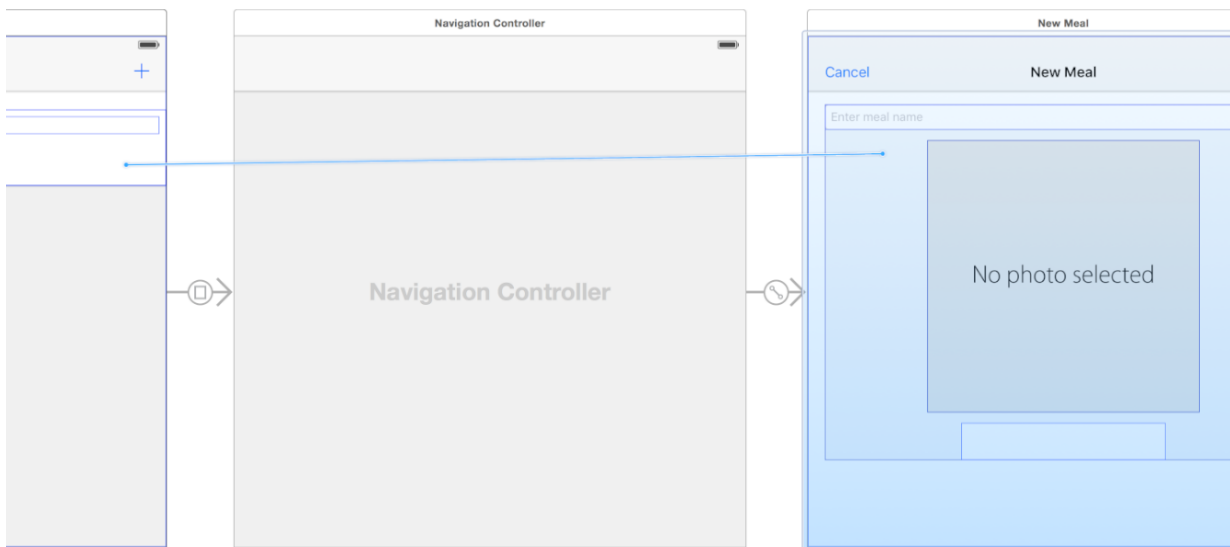
در این بخش شما به کاربر این امکان را می دهید تا بر روی یک خانه از جدول کلیک کرده و به مجرد کلیک نسخه ای از صفحه محتوای افزودن غذای جدید (meal scene) را مشاهده کند که از قبل با اطلاعات مربوط به یک غذا پر شده است. کاربر تغییراتی را اعمال کرده، سپس بر روی دکمه ی Save جهت بروز رسانی و بازنویسی اطلاعات آیتم جاری در لیست غذاها، کلیک می کند.

جهت تنظیم خانه ی جدول برای ویرایش و بروز رسانی اطلاعات، مراحل زیر را دنبال نمایید:

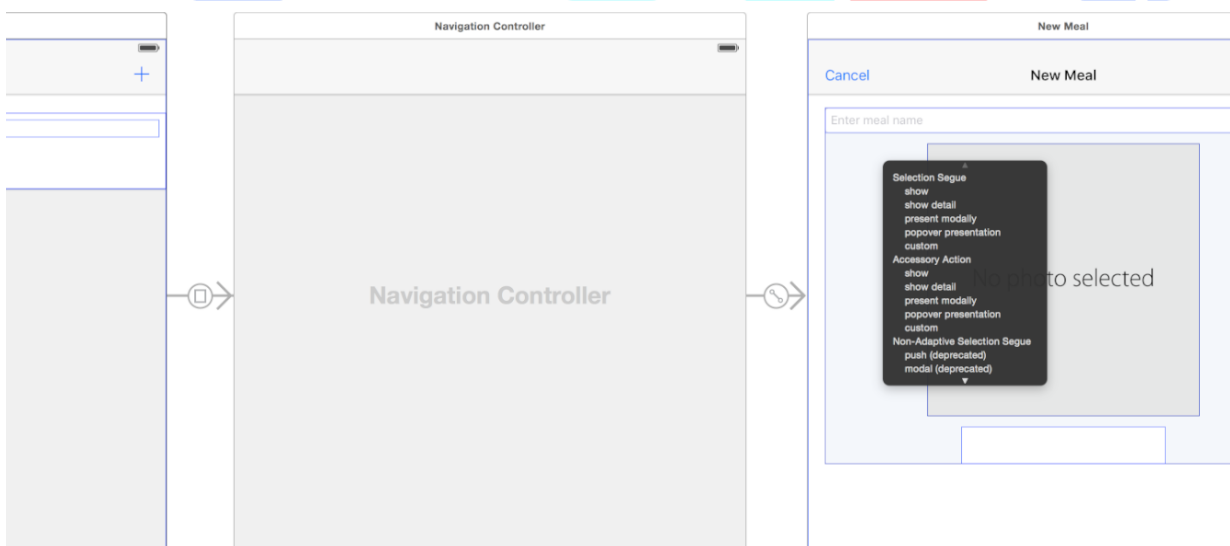
۱. ابتدا با کلیک بر روی دکمه ی Standard (اشاره در تصویر زیر)، ویرایشگر اصلی (standard editor) محیط کاری Xcode را باز نمایید.



۲. حال storyboard، فایل Main.storyboard را باز نمایید.
۳. در سطح canvas، خانه ی جدول (table view cell) را با کلیک بر روی آن، انتخاب نمایید.
۴. کلید control را نگه دارید و سپس با اشاره گر موس خانه ی جدول را به meal scene یا صفحه ی افزودن غذای جدید بکشید.



یک منوی میانبر (shortcut menu) با عنوان Selection Segue در مکانی که آیتم مورد نظر را در آنجا جایگذاری کردید، پدیدار می شود.

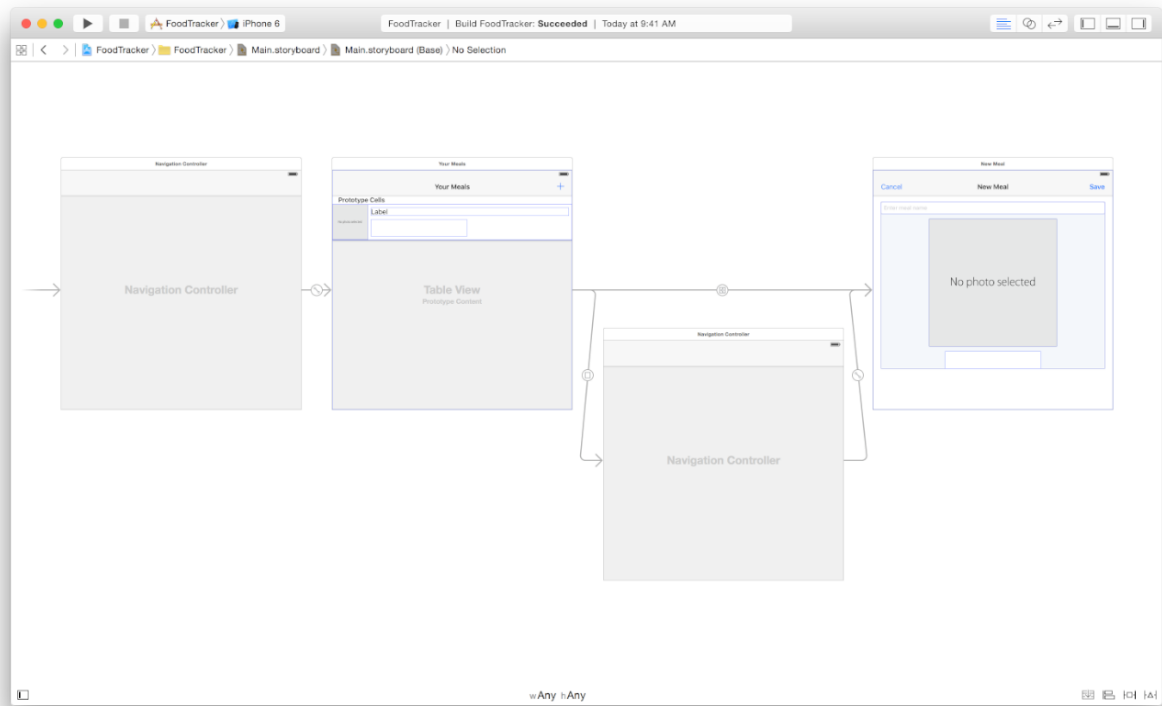


۵. از منوی مزبور، گزینه ی show را انتخاب نمایید.

۶. Navigation controller موجود بین دو صفحه محتوای meal list (نمایش لیست

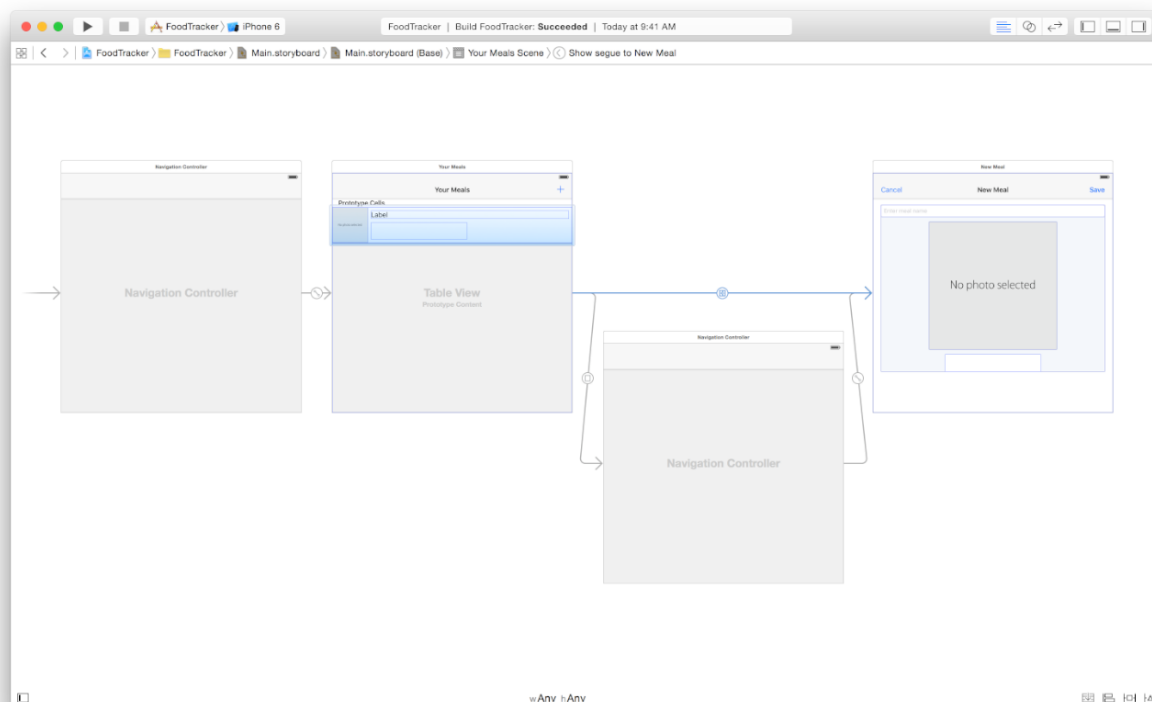
غذاها) و meal scene (صفحه محتوای افزودن غذای جدید) به سمت پایین

کشیده تا segue (انتقال) بین دو scene را مشاهده نمایید.



در صورت تمایل، می‌توانید با اعمال همزمان کلیدهای Command و (-) تصویر را کوچک نمایید.

۷. در سطح canvas، با کلیک بر روی segue جدید، آن را انتخاب نمایید.



۸. داخل کادر inspector Attribute، واژه ی ShowDetail را به عنوان اسم و شناسه ی segue مورد نظر در فیلد identifier وارد نمایید. کلید Return را فشار دهید.

The screenshot shows the 'Storyboard Segue' inspector in Xcode. The 'Identifier' field is set to 'ShowDetail'. The 'Segue Class' is set to 'UIStoryboardSegue'. The 'Segue Module' is set to 'None'. The 'Segue' type is set to 'Show (e.g. Push)'. The 'Animates' checkbox is checked.

اکنون زمانی که این segue اجرا می شود، همزمان view controller مربوط به meal scene (صفحه ی افزودن آیتم جدید) بر روی همان navigation stack یا مجموعه view controller هایی که meal list (صفحه محتوای نمایش لیست غذاها) در حال حاضر عضو آن است، به سبک push قرار می گیرد.

**تست کنید:** برنامه ی خود را اجرا کنید. در صفحه ی نمایش لیست غذاها باید بتوانید با کلیک بر روی یک خانه از جدول، به صفحه ی افزودن آیتم جدید راه پیدا کنید. پس از هدایت به این scene، با صفحه ی خالی مواجه می شوید که امکان افزودن غذای جدید را فراهم می کند. اما شما می خواهید اطلاعات مربوط به آیتم جاری را ویرایش کنید. در زیر به پیاده سازی این قابلیت خواهید پرداخت.

در زمان حاضر، شما دو segue دارید که هر دوی آن ها کاربر را به scene یکسان (صفحه ی افزودن غذای جدید) هدایت می کنند. اما اگر بخاطر داشته باشید، در این مبحث می خواهید امکان ویرایش اطلاعات آیتم جاری را به کاربر اعطا نمایید. حال این سوال مطرح می شود: از کجا می توان فهمید چه زمان کاربر می خواهد یک غذای جدید به برنامه اضافه کند و چه زمان می خواهد اطلاعات مربوط به غذای جاری را ویرایش کند؟ بنابراین باید روشی تعبیه کنید که به وسیله ی آن بتوانید تشخیص دهید چه زمان کاربر می خواهد غذای جدید اضافه کند و چه زمان مایل به ویرایش یا بروز رسانی اطلاعات مربوط به غذای جاری می باشد.

یادآور می شویم که هرگاه segue اجرا می شود، حتما قبل از آن متد `prepareForSegue(_:sender:)` فراخوانی می گردد. شما می توانید داخل بدنه ی این متد کدی تعریف کرده و به وسیله ی آن تشخیص دهید کدام segue در حال اجرا است و به تبع آن اطلاعات مربوطه را در meal scene نمایش دهید. اگر به خاطر داشته باشید قبلا برای این segue ها نام مشخصی تعیین کردید. حال با استفاده از نام آن ها، می توانید به راحتی بین segue ها تمایز قائل شده و دقیقا تشخیص دهید کدام

segue در حال اجرا است. دو segue به همراه اسم آن ها: ۱. AddItem (modal segue) ۲. ShowDetail (show segue).

برای تشخیص اینکه کدام segue در حال اجرا است، مراحل زیر را به ترتیب دنبال نمایید:

۱. ابتدا فایل MealTableViewController.swift را باز کنید.
۲. داخل فایل MealTableViewController.swift، متد `prepareForSegue(_:sender:)` method را یافته و آن را از حالت comment خارج نمایید. (جهت خارج نمودن متد ذکر شده از حالت comment، کافی است کاراکترهای `/*` و `*/` را از طرفین آن حذف نمایید.) پس از انجام این کار، پیاده سازی که به صورت آماده و به عنوان الگو توسط محیط Xcode در اختیار شما قرار گرفته (template implementation)، به صورت زیر خواهد بود:

```
// MARK: - Navigation
// In a storyboard-based application, you will often want to do a little preparation before navigation
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    // Get the new view controller using segue.destinationViewController.
    // Pass the selected object to the new view controller.
}
```

از آنجایی که MealTableViewController یک کلاس مشتق شده از UITableViewController است، پیاده سازی الگو همراه با اسکلت متد `prepareForSegue(_:sender:)` توسط محیط به صورت آماده در اختیار شما قرار می گیرد.

۳. دو خط comment را حذف نموده، سپس دستورات شرطی `if` و `else` را جایگزین آن نمایید:



```
if segue.identifier == "ShowDetail" {
}
else if segue.identifier == "AddItem" {
}
```

این کد خصیصه‌ی اسم یا identifier هر یک از دو segue را با اسم تخصیص داده شده به آن‌ها در کادر Attribute inspector مقایسه می‌کند.

۴. داخل ساختمان اولین دستور شرطی، کد زیر را درج نمایید (این دستور تنها زمانی اجرا می‌شود که آیتم یا غذای مورد نظر توسط کاربر در حال ویرایش باشد):

```
let mealDetailViewController = segue.destinationViewController as!
MealViewController
```

این کد با استفاده از عملگر "as!" سعی می‌کند، view controller مقصد (destination view controller) از segue مورد نظر را به MealViewController که یک کلاس ارث‌بری شده از آن هست، تبدیل نوع (downcast) نماید. با کمی دقت متوجه می‌شوید که در انتهای این عملیات بجای "?" از کاراکتر "!" استفاده شده است. معنی استفاده از کاراکتر مزبور در انتهای عملیات جاری این است که کد تلاش می‌کند به زور عملیات تبدیل را انجام دهد، به طوری که اگر پروسه‌ی تبدیل با موفقیت صورت پذیرد، مقدار segue.destinationViewController به ثابت محلی mealDetailViewController اختصاص می‌یابد و در غیر این صورت اپلیکیشن در زمان اجرا به دلیل ناموفق بودن تبدیل به طور ناگهانی از کار می‌افتد. به عبارت دیگر تبدیل در هر صورت به طور تحمیلی انجام می‌شود، حتی اگر اصلاً امکان آن وجود نداشته باشد که در صورت ناموفق بودن سبب رخداد خطای مهلک در runtime و عدم اجرای برنامه می‌شود.

با توجه به توضیحات فوق، تنها در صورتی باید از عملگر "as!" استفاده کنید که از امکان پذیر بودن عملیات تبدیل و موفقیت آمیز بودن آن اطمینان کامل دارید.

چنانچه مطمئن نیستید که تبدیل به طور حتم با موفقیت انجام می پذیرد، بهتر است از عملگر "as?" استفاده نمایید.

۵. در زیر خط قبلی، یک دستور if دیگر به صورت زیر درج نمایید (این دستور در واقع به صورت تودرتو داخل بدنه ی دستور شرطی اول گنجانده می شود):

```
// Get the cell that generated this segue.
if let selectedMealCell = sender as? MealTableViewCell {
}
```

این کد سعی می کند با استفاده از عملگر "as?" آبجکت sender را به MealCell تبدیل (downcast) نماید. اگر پروسه ی تبدیل با موفقیت انجام شود، مقدار sender که به MealTableViewCell تبدیل شده، در ثابت محلی selectedMealCell ذخیره می گردد و در پی آن دستور تعریف شده در بدنه ی if اجرا می شود. اما چنانچه تبدیل با شکست مواجه شود، عبارت شرطی در واقع برابر nil بوده و دستور مربوطه به مرحله ی اجرا نمی رسد.

۶. داخل ساختمان if، مجموعه دستورات زیر را وارد نمایید:

```
let indexPath = tableView.indexPathForCell(selectedMealCell)!
let selectedMeal = meals[indexPath.row]
mealDetailViewController.meal = selectedMeal
```

این کد آبجکت Meal که متناظر یا مربوط به خانه ی انتخاب شده از جدول (table view) می باشد را واکنشی می کند. سپس مقدار آبجکت Meal را به متغیر meal (property) از mealDetailViewController مقصد تخصیص می دهد که نمونه ای از کلاس MealViewController می باشد (mealDetailViewController.view controller destination یا mealDetailViewController.view controller صفحه ای که segue نهایتاً به آن کاربر را هدایت می کند؛ صفحه محتوایی که در انتهای segue مشاهده می شود). شما MealViewController را طوری تنظیم خواهید کرد که به هنگام بارگذاری اطلاعات را از متغیر meal خوانده و نمایش دهد.

۷. داخل ساختمان دستور شرطی else، تابع print را با پارامتر ورودی زیر درج نمایید:

```
print("Adding new meal.")
```

اگرچه شما در صورت اضافه نمودن آیتم جدید به برنامه اصلاً نیازی به این متد ندارید (استفاده از این متد کاملاً غیر ضروری می شود)، با این حال بد نیست گزارشی از آنچه در حال رخدادن هست را برای کاربر در UI چاپ نمایید.

بدنه ی متد (sender:) prepareForSegue هم اکنون می بایست دارای پیاده سازی زیر باشد:

```
override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
    if segue.identifier == "ShowDetail" {
        let mealDetailViewController = segue.destinationViewController as!
        MealViewController
        // Get the cell that generated this segue.
        if let selectedMealCell = sender as? MealTableViewCell {
            let indexPath = tableView.indexPathForCell(selectedMealCell)!
            let selectedMeal = meals[indexPath.row]
            mealDetailViewController.meal = selectedMeal
        }
    }
    else if segue.identifier == "AddItem" {
        print("Adding new meal.")
    }
}
```

پس از پیاده سازی منطق لازم، می بایست با نوشتن رفتار و کد مناسب در فایل MealViewController.swift، اطمینان حاصل نمایید که UI اپلیکیشن به درستی بروز آوری شده و محتوای مد نظر را برای کاربر به نمایش می گذارد. به عبارتی دقیق تر، زمانی که نمونه یا آبجکتی از کلاس MealViewController (صفحه ی افزودن غذای جدید) ساخته می شود، view های آن (المان های رابط کاربری) بایستی با داده های مناسب از متغیر meal پر شوند (البته در صورت

وجود داده‌ی مورد نظر). اگر بخاطر داشته باشید، مکان مناسب برای انجام این

نوع تنظیمات، بدنه‌ی متد `viewDidLoad()` است.

به منظور بروز رسانی پیاده سازی متد `viewDidLoad`، مراحل زیر را طی نمایید:

۱. فایل `MealViewController.swift` را باز نمایید.

۲. داخل این فایل، متد `viewDidLoad()` را پیدا کنید.

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Handle the text field's user input via delegate callbacks.
    nameTextField.delegate = self
    // Enable the Save button only if the text field has a valid Meal name.
    checkValidMealName()
}
```

۳. در زیر خط `nameTextField.delegate`، این کد را اضافه نمایید:

```
// Set up views if editing an existing Meal.
```

```
if let meal = meal {
    navigationItem.title = meal.name
    nameTextField.text = meal.name
    photoImageView.image = meal.photo
    ratingControl.rating = meal.rating
}
```

این کد سبب می‌شود، در صورت `non-nil` بودن متغیر `meal`، تمامی آبجکت‌های

`view` در فایل `MealViewController` داده‌های مربوطه را از متغیر `meal`

خوانده و نمایش دهند. این رخداد تنها زمانی اتفاق می‌افتد که غذا یا آیتم جاری

توسط کاربر در حال ویرایش باشد.

در حال حاضر متد `viewDidLoad()` شما می‌بایست دارای پیاده سازی زیر باشد:

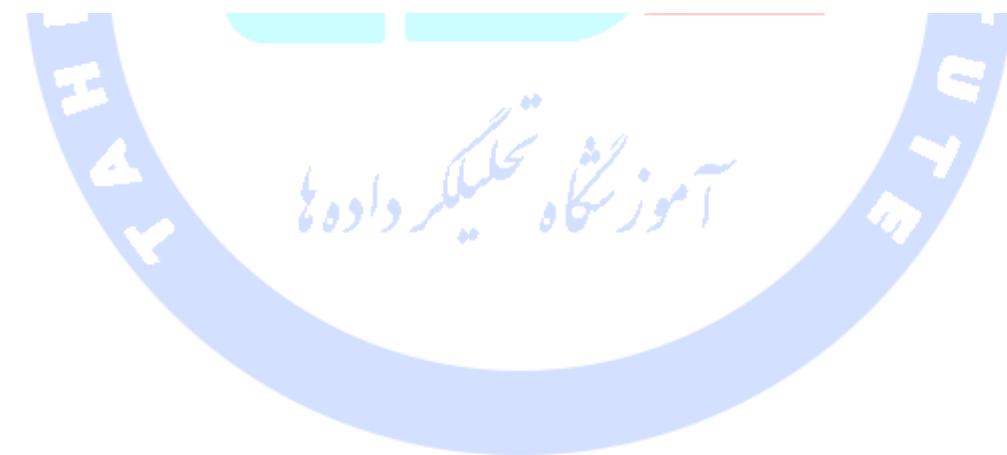
```
override func viewDidLoad() {
    super.viewDidLoad()
    // Handle the text field's user input via delegate callbacks.
    nameTextField.delegate = self
    // Set up views if editing an existing Meal.
```

```

if let meal = meal {
    navigationItem.title = meal.name
    nameTextField.text = meal.name
    photoImageView.image = meal.photo
    ratingControl.rating = meal.rating
}
// Enable the Save button only if the text field has a valid Meal name.
checkValidMealName()
}

```

**تست کنید:** برنامه ی خود را اجرا نمایید. بایستی بتوانید با کلیک بر روی یکی از خانه های جدول (table view cell)، به صفحه ی افزودن غذای جدید (meal scene) راه پیدا کرده و آن را در حالی که از قبل با داده های مربوط با meal پر شده، مشاهده نمایید. اما هنوز یک جای کار مشکل دارد: با کلیک بر روی دکمه ی Save، بجای اینکه اطلاعات جدید بر روی اطلاعات قبلی غذا نوشته شود، برنامه یک آیتم جدید به لیست غذاها اضافه می کند. در زیر به پیاده سازی رفتار مناسب برای این بخش خواهید پرداخت.





جهت بازنویسی اطلاعات مربوط به یک غذا در لیست، می بایست اکشن متد `unwindToMealList(_)` را طوری ویرایش و بروز رسانی نمایید که بتواند دو سناریو کاملاً متفاوت را مدیریت کند. این متد در هر سناریو بایستی بتواند کار متفاوتی انجام دهد: در یکی غذای جدید اضافه کند و در دیگری اطلاعات مربوط به غذای جاری را

بازنویسی نماید. یادآور می شویم که این متد تنها زمانی صدا خورده می شود که کاربر بر روی دکمه ی Save کلیک کرده باشد. بنابراین نیازی به اندیشیدن تمهیدات برای دکمه ی Cancel و کد متصل به آن در بدنه ی این متد نیست.

به منظور بروز رسانی پیاده سازی متد unwindToMealList(\_\_\_\_) جهت افزودن غذای جدید یا بازنویسی غذای جاری، مراحل زیر را طی نمایید:

۱. فایل MealTableViewController.swift را باز نمایید.

۲. داخل این فایل، متد unwindToMealList(\_\_\_\_) را پیدا کنید:

```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
    if let sourceViewController = sender.sourceViewController as?
    MealViewController, meal = sourceViewController.meal {
        // Add a new meal.
        let newIndexPath = NSIndexPath(forRow: meals.count, inSection: 0)
        meals.append(meal)
        tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation:
        .Bottom)
    }
}
```

۳. در ابتدای ساختمان دستور شرطی if، این کد را اضافه نمایید:

```
if let selectedIndexPath = tableView.indexPathForSelectedRow {
}
```

این کد بررسی می کند آیا سطری از جدول (table view) انتخاب شده است یا خیر. در صورتی که سطر از جدول انتخاب شده باشد، معنیش این است که کاربر به منظور ویرایش اطلاعات یک آیتم، بر روی سطری از جدول کلیک کرده است. به عبارت روشن تر، دستور نام برده زمانی اجرا می شود که یک آیتم از لیست در حال ویرایش باشد.

۴. داخل بدنه ی این دستور شرطی if، کد زیر را درج نمایید:



// Update an existing meal.

```
meals[selectedIndexPath.row] = meal
```

```
tableView.reloadRowsAtIndexPaths([selectedIndexPath], withRowAnimation:
.None)
```

اولین خط از این مجموعه دستور، آیتم مربوطه در meals را جهت بازنویسی و ذخیره ی اطلاعات جدید یا ویرایش شده ی آن، بروز رسانی می کند. دومین خط، سطر مورد نظر را مجدداً در جدول بارگذاری نموده و اطلاعات جدید غذا یا آیتم مربوطه را نمایش می دهد.

۵. پس از دستور if، یک دستور else اضافه نموده و آن را در چهار خط پایانی متد، به صورت

زیر جای دهید:

```
else {
```

// Add a new meal.

```
let newIndexPath = NSIndexPath(forRow: meals.count, inSection: 0)
```

```
meals.append(meal)
```

```
tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation:
.Bottom)
```

```
}
```

جهت اطمینان از فاصله گذاری صحیح از سر هر سطر و رعایت توگذاری، کافی است تمامی دستورات را انتخاب نموده و سپس کلیدهای Control-ا را همزمان فشار دهید.

دستور else زمانی اجرا می شود که هیچ سطر از جدول (table view) انتخاب نشده باشد، بدین معنی که کاربر با کلیک بر روی دکمه ی Add، به صفحه ی افزودن غذای جدید هدایت شده. به عبارت دقیق تر، این دستور شرطی زمانی اجرا می شود که آیتم جدیدی به لیست غذاها اضافه می شود.

در حال حاضر، پیاده سازی متد unwindToMealList(\_\_\_\_) می بایست به صورت زیر باشد:

```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
```

```
if let sourceViewController = sender.sourceViewController as?
```

```
MealViewController, meal = sourceViewController.meal {
```

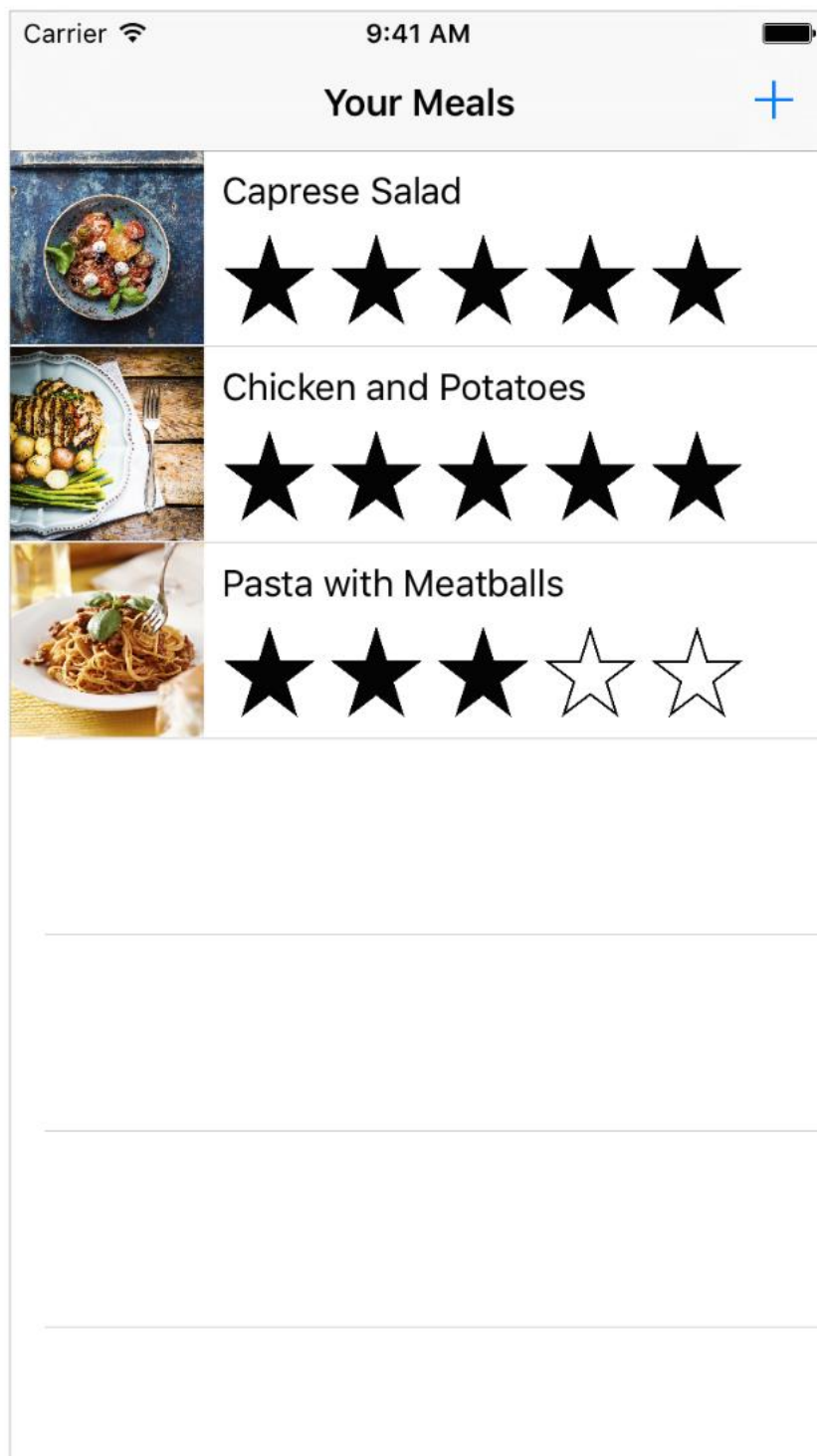


```

if let selectedIndexPath = tableView.indexPathForSelectedRow {
    // Update an existing meal.
    meals[selectedIndexPath.row] = meal
    tableView.reloadRowsAtIndexPaths([selectedIndexPath], withRowAnimation:
        .None)
}
else {
    // Add a new meal.
    let newIndexPath = NSIndexPath(forRow: meals.count, inSection: 0)
    meals.append(meal)
    tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation:
        .Bottom)
}
}
}

```

**تست کنید:** اپلیکیشن خود را اجرا نمایید. بایستی بتوانید با کلیک بر روی یک سطر از جدول به صفحه ی افزودن غذای جدید (meal scene) راه پیدا کرده و آن را در حالی که از قبل با داده های مرتبط با meal مورد نظر پر شده، مشاهده نمایید. در صورت کلیک بر روی دکمه ی Save، تغییراتی که اخیرا اعمال کردید باید بر روی غذای جاری در لیست بازنویسی شود (جایگزین آن گردد).



افزودن قابلیت لغو ویرایش آیتم جاری و بازگشت به صفحه ی لیست غذاها بدون ذخیره ی تغییرات  
(بروز رسانی کد متصل به دکمه ی Cancel)

کاربر اپلیکیشن شما شاید مایل باشد تغییراتی که به یک meal وارد کرده را لغو و به دنبال آن بدون اینکه تغییرات مزبور را ذخیره کرده باشد، به صفحه ی نمایش لیست

غذاها بازگردد. برای این منظور، رفتار دکمه ی Cancel را در کد برنامه طوری ویرایش خواهید کرد که scene مورد نظر را از نمایشگر پنهان نماید. به عبارتی دیگر، آن صفحه را بسته، کاربر را به صفحه ی دیگری هدایت کند.

نحوه ی بستن صفحه و حذف از آن نمایشگر، به نوع ارائه و نمایش آن بستگی دارد. در این بخش شما روشی تعبیه خواهید کرد که طی آن بتوانید تشخیص دهید scene حاضر، چگونه به هنگام کلیک کاربر بر روی دکمه ی Cancel، نمایش داده شده است. اگر به صورت شناور یا modal (در پی کلیک کاربر بر روی دکمه ی Add) بر روی نمایشگر ظاهر شده باشد، متعاقباً به وسیله ی متد dismissViewControllerAnimated(completion:\_) از نمایشگر حذف می شود. حال اگر به سبک push (به واسطه ی یک خانه از جدول یا table view cell) بر روی صفحه به نمایش درآمده باشد، آنگاه توسط همان کلاس navigation controller ای که بر روی نمایشگر پدیدار شد، از صفحه حذف می شود.

به منظور ویرایش و بروز رسانی پیاده سازی متد متصل به دکمه ی Cancel، مراحل زیر را دنبال نمایید:

۱. فایل MealViewController.swift را باز نمایید.

۲. داخل فایل نام برده، متد cancel(\_:) را پیدا کنید.

```
@IBAction func cancel(sender: UIBarButtonItem) {
dismissViewControllerAnimated(true, completion: nil)
}
```

این پیاده سازی در حال حاضر تنها از متد dismissViewControllerAnimated

برای بستن meal scene (صفحه ی افزودن غذای جدید) بهره می گیرد چرا که تا

به اینجا تنها می بایست تمهیدات لازم برای دکمه ی Add را در نظر می گرفتید.

۳. داخل ساختمان متد cancel(\_:)، قبل از کد جاری، دستور زیر را اضافه نمایید.

// Depending on style of presentation (modal or push presentation), this view controller needs to be dismissed in two different ways.

```
let isPresentingInAddMealMode = presentingViewController is
    UINavigationController
```

دستوری که هم اکنون به قطعه کد اضافه کردید، یک مقدار بولی تعریف کرده و طی آن مشخص می کند آیا view controller ای که این صفحه محتوا (scene) را به نمایش می گذارد، از جنس کلاس UINavigationController هست یا خیر. از اسم ثابت isPresentingInAddMealMode پیدا است که meal scene به وسیله ی دکمه ی Add بر روی نمایشگر ارائه می شود. دلیلش این است که meal scene در حالی به این سبک به نمایش در می آید که داخل navigation controller خود جاسازی شده، بدین معنی که این کلاس navigation controller است که صفحه افزودن غذای جدید را در نمایشگر به صورت شناور (modal) ارائه می دهد.

۴. حال دستور شرطی زیر را اضافه نموده و خطی که تابع dismissViewControllerAnimated را صدا می زند، در داخل بدنه ی این دستور شرطی جایگذاری نمایید:

```
if isPresentingInAddMealMode {
    dismissViewControllerAnimated(true, completion: nil)
}
```

اگر بخاطر داشته باشید، قبلاً متد dismissViewControllerAnimated همیشه بلافاصله پس از صدا خوردن تابع (cancel(\_)) فراخوانده می شد. اما اکنون تنها زمانی اجرا می شود که شرط isPresentingInAddMealMode صادق یا true باشد.

۵. بلافاصله پس از دستور شرطی if، عبارت else را به صورت زیر اضافه نمایید:

```
else {
    navigationController!.popViewControllerAnimated(true)
}
```

در زمان حاضر، این قطعه کد یک دستور if با عبارت شرطی else را در برمی گیرد. دستور موجود در بدنه ی if تنها زمانی اجرا می شود که شرط isPresentingInAddMealMode برقرار یا صحیح باشد و در غیر این صورت دستور داخل ساختمان else اجرا خواهد شد. در حقیقت دستور else زمانی اجرا می شود که meal scene (صفحه ی افزودن غذای جدید) بر روی navigation stack (مجموعه view controller ها) که meal list scene (صفحه ی نمایش لیست غذاها) عضوی از آن است، قرار داده (push) شود. دستور موجود در بدنه ی else یک تابع به نام popViewControllerAnimated را اجرا می کند که view controller جاری (meal scene) را از روی navigation stack کلاس navigationController برداشته (pop off) و همزمان با این کار انتقال یا transition را با انیمیشن به اجرا در می آورد.

هم اکنون پیاده سازی متد (cancel(\_)) می بایست به صورت زیر باشد:

```
@IBAction func cancel(sender: UIBarButtonItem) {
    // Depending on style of presentation (modal or push presentation), this view
    // controller needs to be dismissed in two different ways.
    let isPresentingInAddMealMode = presentingViewController is
    UINavigationController
    if isPresentingInAddMealMode {
        dismissViewControllerAnimated(true, completion: nil)
    }
    else {
        navigationController!.popViewControllerAnimated(true)
    }
}
```

**تست کنید:** برنامه ی خود را اجرا نمایید. اکنون زمانی که بر روی دکمه ی (+) کلیک کرده و بعد بجای Save، دکمه ی Cancel را فشار می دهید، برنامه باید بدون ذخیره ی اطلاعات و افزودن آیتم جدید، شما را به صفحه ی نمایش لیست غذاها (meal list scene) هدایت کند.

## پیاده سازی امکان حذف آیتم ها (اضافه کردن دکمه ی Delete)

بد نیست به کاربران اپلیکیشن خود این قابلیت را بدهید تا آیتم دلخواه خود را از لیست حذف کنند. برای این منظور می بایست روشی تعبیه نمایید که طی آن کاربران بتوانند table view (جدول) را در وضعیت ویرایش (editing mode) قرار داده و سپس از آنجا آیتم مد نظرشان را حذف کنند. در این راستا، یک دکمه ی Edit به نوار پیمایش table view اضافه خواهید نمود.

به منظور افزودن دکمه ی Edit به table view، مراحل زیر را دنبال نمایید:

۱. فایل MealTableViewController.swift را باز نمایید.

۲. داخل فایل ذکر شده، متد viewDidLoad() را پیدا کنید.

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Load the sample data.
    loadSampleMeals()
}
```

در زیر خط super.viewDidLoad()، کد زیر را اضافه نمایید:

// Use the edit button item provided by the table view controller.

```
navigationItem.leftBarButtonItem = editButtonItem()
```

این کد یک نوع bar button item ایجاد کرده که قابلیت ویرایش را به صورت درون ساخته در خود دارد. سپس این دکمه را به سمت چپ نوار پیمایش در صفحه ی نمایش لیست غذاها (meal list scene) اضافه می کند.

متد viewDidLoad() اکنون بایستی دارای پیاده سازی زیر باشد:

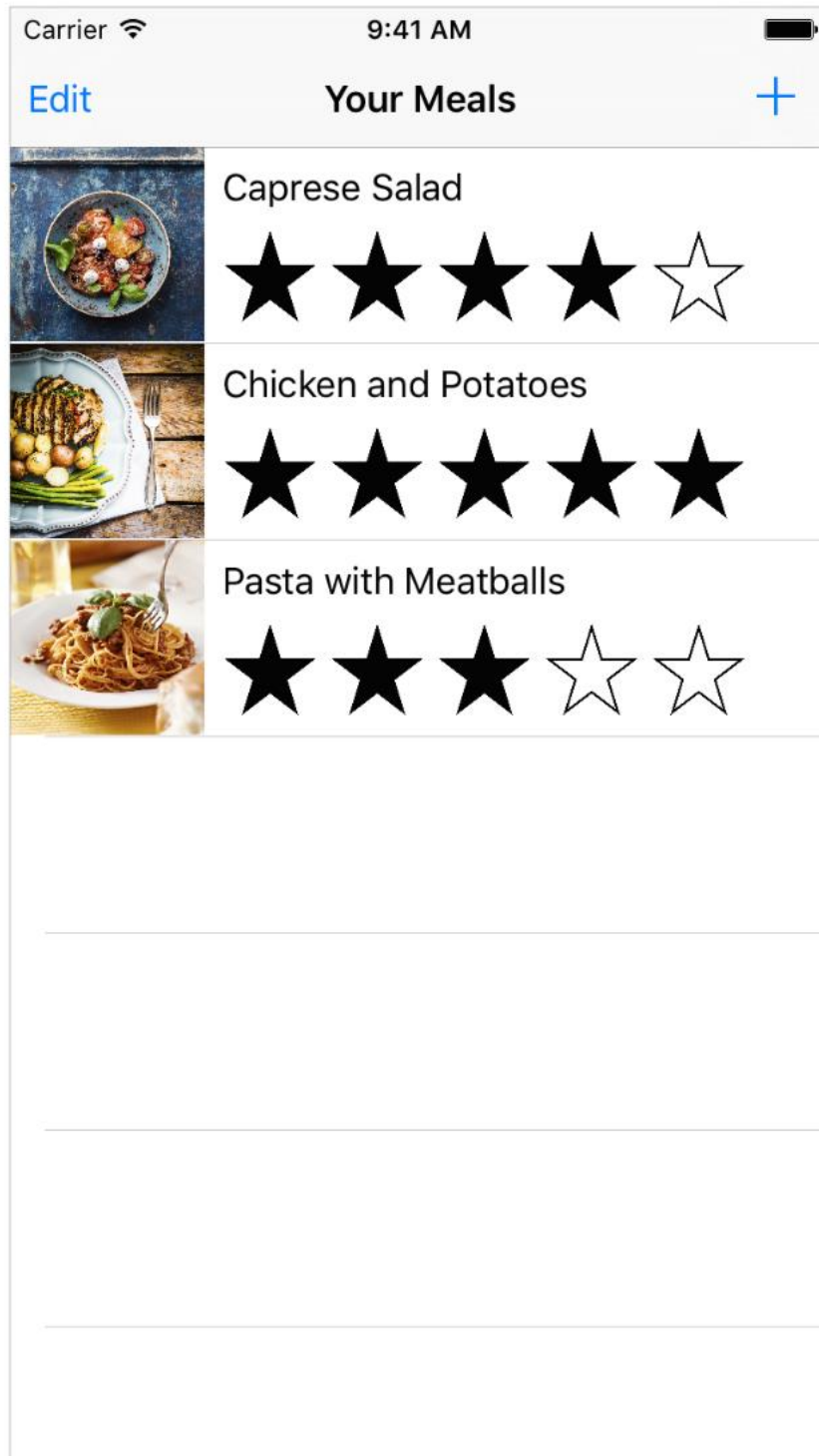
```
override func viewDidLoad() {
    super.viewDidLoad()
    // Use the edit button item provided by the table view controller.
    navigationItem.leftBarButtonItem = editButtonItem()
    // Load the sample data.
```

```
loadSampleMeals()
```

```
}
```

**تست کنید:** برنامه را اجرا نمایید. همان طور که می بینید یک دکمه ی Edit در سمت چپ نوار پیمایش table view قابل مشاهده می باشد. اگر بر روی دکمه ی Edit کلیک نمایید، table view وارد وضعیت ویرایش می شود – اما از آنجایی که هنوز کد لازم برای حذف را پیاده نکرده اید، فعلا نمی توانید سطری را از جدول حذف نمایید.





جهت انجام هر گونه عملیات ویرایش بر روی table view، شما می بایست یکی از متدهای Delegate آن به نام `tableView(_:commitEditingStyle:forRowAtIndexPath:)` را پیاده سازی نمایید. این متد وظیفه ی مدیریت سطرهای جدول، زمانی که این سطرها در وضعیت



ویرایش قرار دارند (توسط کاربر در حال ویرایش هستند) را بر عهده دارد. علاوه بر آن، لازم است متد `tableView(_:canEditRowAtIndexPath:)` را از حالت `comment` خارج نمایید. این متد همان طور که از پارامتر ورودی آن مشخص است، امکان ویرایش را فراهم می کند.

جهت حذف یک meal یا آیتم از لیست، مراحل زیر را دنبال نمایید:

۱. در فایل `MealTableViewController.swift`،

متد `tableView(_:commitEditingStyle:forRowAtIndexPath:)` را یافته و با حذف کاراکترهای `/*` و `*/` از طرفین این متد، آن را از حالت `comment` خارج نمایید. پس وارد کردن این تغییرات، پیاده سازی که به صورت آماده توسط Xcode در اختیار شما قرار می گیرد، به صورت زیر خواهد بود:

```
// Override to support editing the table view.
override func tableView(tableView: UITableView, commitEditingStyle
editingStyle: UITableViewCellEditingStyle, forRowAtIndexPath indexPath:
NSIndexPath) {
if editingStyle == .Delete {
// Delete the row from the data source
tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)
} else if editingStyle == .Insert {
// Create a new instance of the appropriate class, insert it into the array, and add
a new row to the table view
}
}
```

در زیر خط توضیح، `Delete the row from the data source`، این دستور را اضافه نمایید:

```
meals.removeAtIndex(indexPath.row)
```

این کد آبجکت Meal که قرار است از آرایه ی meals پاک شود را حذف می کند. خط پس از آن که بخشی از پیاده سازی ارائه شده توسط Xcode هست، سطر مربوطه را از table view (جدول) حذف می کند.

۲. در فایل `MealTableViewController.swift`، متد

`tableView(_:canEditRowAtIndexPath:)` را یافته و از حالت `comment`

خارج نمایید. پس از وارد کردن این تغییر، پیاده سازی که به صورت آماده توسط

Xcode در اختیار شما قرار می گیرد، به صورت زیر خواهد بود:

`// Override to support conditional editing of the table view.`

```
override func tableView(tableView: UITableView, canEditRowAtIndexPath:
indexPath: NSIndexPath) -> Bool {
```

`// Return false if you do not want the specified item to be editable.`

`return true`

`}`

در حال حاضر، متد `tableView(_:commitEditingStyle:forRowAtIndexPath:)`

شما می بایست دارای پیاده سازی زیر باشد:

`// Override to support editing the table view.`

```
override func tableView(tableView: UITableView, commitEditingStyle
EditingStyle: UITableViewCellEditingStyle, forRowAtIndexPath indexPath:
NSIndexPath) {
```

`if editingStyle == .Delete {`

`// Delete the row from the data source`

`meals.removeAtIndex(indexPath.row)`

`tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)`

`} else if editingStyle == .Insert {`

`// Create a new instance of the appropriate class, insert it into the array, and add a new row to the table view`

`}`

`}`

**تست کنید:** برنامه ی خود را اجرا نمایید. با کلیک بر روی دکمه ی Edit، جدول (table

view) در وضعیت ویرایش قرار می گیرد. می توانید با کلیک بر روی آیکن قرمز رنگ که

در سمت چپ سطر مربوطه قابل مشاهده می باشد، آن سطر را برای حذف انتخاب و آماده

نمایید. سپس جهت حذف دائمی آن سطر، دکمه ی قرمز رنگ Delete که در سمت راست

سطر مورد نظر به نمایش در آمده را کلیک نمایید.

و یا در صورت تمایل می توانید انگشت خود را بر روی سطر دلخواه در سطح نمایشگر قرار داده آن را به سمت چپ بکشید (swipe)، می بینید که دکمه ی Delete نمایان می شود. این رفتار به صورت درون ساخته و از پیش تعریف شده در آبجکت table view وجود دارد. پس از کلیک بر روی دکمه ی Delete، سطر متناظر به طور دائمی از جدول حذف می شود.

## درس ۱۱ : آموزش Data Persistence در Swift

ماندگار سازی یا ذخیره دائمی داده های برنامه (Data Persistence)

این مبحث به شرح چگونگی ماندگار سازی و ذخیره دائمی meal list (لیست نمایش غذاها) در تمامی session های اپلیکیشن FoodTracker می پردازد. ذخیره سازی دائمی اطلاعات یکی از معمول ترین مشکلات در برنامه سازی برای سیستم عامل IOS می باشد که بسیاری از توسعه دهندگان با آن مواجه می شوند. سیستم عامل IOS راه حل های مختلفی ویژه ی ماندگار سازی داده ها ارائه می دهد. در آموزش حاضر، شما از NSCoder به عنوان مکانیزم ذخیره سازی دائمی داده ها در برنامه ی کاربردی FoodTracker بهره خواهید گرفت. NSCoder یک protocol یا الگوی پیاده سازی است که امکان ذخیره و ماندگار سازی آبجکت ها و دیگر ساختارهای ذخیره ی اطلاعات (structure) را به صورت بهینه فراهم می آورد. آبجکت های کدگذاری شده (archived) را می توان بر روی دیسک ذخیره کرده و بعده ها از آن واکنشی کرد.

### آنچه خواهید آموخت

۱. می توانید یک structure تعریف کنید.

۲. تفاوت میان instance property (متغیرهای متعلق به نمونه ی کلاس یا به اصطلاح non-static) و static property (متغیرهایی که به صورت static تعریف شده و متعلق به خود کلاس هستند) را تشخیص داده و شرح دهید.

۳. با استفاده از پروتکل NSCodering داده هایی را از دیسک خوانده و بر روی آن ذخیره (write) نمایید.

### پیاده سازی قابلیت ذخیره ی دائم و بارگذاری آیتم (Save/Load مقدار meal در کلاس Meal)

در این بخش، رفتار یا قابلیت در کلاس Meal پیاده سازی خواهید کرد که اطلاعات مربوط به آیتم جدید (meal) را ذخیره کرده و بارگذاری می کند.

کلاس Meal با پیروی از الگوی پیاده سازی NSCodering (پیاده سازی متدهایی که این protocol معرفی می کند) قابلیت و وظایف مربوط به ذخیره سازی و لود هر یک از property ها را نیز به عهده می گیرد.

کلاس یاد شده داده های خود را با تخصیص مقدار هر یک از property ها به key مشخص، ذخیره نموده، سپس این داده ها را با جستجو و یافتن اطلاعات key مربوطه بارگذاری می کند.

Key صرفاً یک مقدار رشته ای (string) است که به مثابه ی اسم یک property ایفای نقش می کند. مطلوب است اسم key را مرتبط با مقداری که قرار است نگه دارد، انتخاب نمایید. به عنوان مثال، بهتر است برای متغیری که مقدار name را نگه می دارد، key را name انتخاب کنید.

برای اینکه مشخص شود کدام قطعه داده به کدام key تعلق دارد، توصیه می شود یک structure تعریف کرده و key ها (اسم متغیرها) را در آن ذخیره نمایید. با این کار هر زمان که ملزوم به ذکر key در بخش های مختلف کد برنامه ی خود می شوید، می توانید بجای تایپ مجدد مقادیر رشته ای که احتمال رخداد خطا را بالا می برد، اسم تخصیص داده شده به ثوابت (constant) ها یا همان key که اکنون در قالب structure ریخته شده) را بکار ببرید.

به منظور پیاده سازی یک structure جهت ذخیره key ها و تعریف property مراحل زیر را دنبال نمایید:

۱. فایل Meal.swift را باز کنید.

۲. داخل فایل ذکر شده، در زیر بخش MARK: Properties، این structure را اضافه

نمایید:

// MARK: Types

```
struct PropertyKey {  
}
```

۳. داخل بدنه ی این structure (PropertyKey)، property ها (ثوابت) زیر را اضافه

نمایید:

```
static let nameKey = "name"  
static let photoKey = "photo"  
static let ratingKey = "rating"
```

ثوابت فوق هر یک به ترتیب با یکی از سه property اعلان شده در کلاس Meal متناظر هستند. کلیدواژه ی static نشانگر این است که ثابت مورد نظر متعلق به خود structure جاری است و نه نمونه ای از آن. این مقادیر در قالب ثوابت ریخته شده اند (با کلیدواژه ی let تعریف شده اند)، به همین دلیل هیچگاه تغییر نمی کنند.

Structure نام برده هم اکنون می بایست دارای پیاده سازی زیر باشد:

```
struct PropertyKey {  
    static let nameKey = "name"  
    static let photoKey = "photo"  
    static let ratingKey = "rating"  
}
```

برای اینکه کلاس Meal بتواند خود و property های عضویش را encode و decode کند (قابلیت رمز گذاری و رمز گشایی را داشته باشد)، بایستی متدهای پروتکل NSCoder را پیاده سازی کند (به اصطلاح از این الگوی پیاده سازی پیروی کند؛ متدها و property های آن را پیاده سازی کند). برای این منظور، متغیر Meal می بایست از

کلاس NSObject ارث بری کند. NSObject یک کلاس پایه است که پل ارتباطی (interface) ساده ای به سیستم runtime می سازد.

به منظور ارث بری از کلاس NSObject (ایجاد یک کلاس فرزند از آن) و پیاده سازی متدهای پروتکل NSCoder، مراحل زیر را دنبال نمایید:

۱. داخل فایل Meal.swift، خط تعریف کلاس (حاوی کلیدواژه ی class) را پیدا کنید:

```
class Meal {
```

۲. پس از واژه ی Meal، عملگر دو نقطه ":" و NSObject را جهت ارث بری از آن ذکر نمایید:

```
class Meal: NSObject {
```

۳. پس از NSObject، یک ویترگول اضافه نموده و سپس واژه ی NSCoder را جهت پیاده سازی توابع این protocol، درج نمایید:

```
class Meal: NSObject, NSCoder {
```

پروتکل NSCoder در کل دو متد ارائه می دهد. تمامی کلاس هایی که این protocol را پیاده سازی می کنند ناگزیر می بایست دو متد آن را نیز پیاده سازی نمایند تا نمونه های کلاس مورد نظر قابلیت encode و decode را داشته باشند (بتوانند رمزگذاری و رمزگشایی شوند):

```
func encodeWithCoder(aCoder: NSCoder)
```

```
init(coder aDecoder: NSCoder)
```

متد encodeWithCoder(;) اطلاعات کلاس را برای کدگذاری و ذخیره بر روی دیسک (archive) آماده ساخته و متد سازنده (initializer) نیز این داده ها را به هنگام ایجاد کلاس از حالت archive در می آورد. شما می بایست هر دو متد را پیاده سازی کرده تا بدین وسیله داده ها بتوانند به درستی ذخیره و بارگذاری شوند.

به منظور پیاده سازی متد encodeWithCoder از پروتکل NSCoder، مراحل زیر را طی نمایید:

۱. در فایل Meal.swift، قبل از آخرین ({})، کد زیر را اضافه نمایید:

## // MARK: NSCoding

خط بالا یک comment است که اعلان می دارد بخش حاضر مربوط به ماندگار سازی و ذخیره ی دائمی داده ها است.

۲. در زیر خط comment این متد را اضافه نمایید:

```
func encodeWithCoder(aCoder: NSCoder) {
}
```

۳. داخل بدنه ی متد encodeWithCoder( \_:)، کد زیر را وارد نمایید:

```
aCoder.encodeObject(name, forKey: PropertyKey.nameKey)
aCoder.encodeObject(photo, forKey: PropertyKey.photoKey)
aCoder.encodeInteger(rating, forKey: PropertyKey.ratingKey)
```

متد encodeObject( \_:forKey:) قادر است هر نوع آبجکتی را کد گذاری کند، در حالی که متد encodeInteger( \_:forKey:) می تواند صرفاً یک داده از نوع integer را کدگذاری نماید. این سه خط کد مقدار property های کلاس Meal را کدگذاری نموده و سپس آن ها را همراه با key متناظر (مربوطه) ذخیره می کند.

متد encodeWithCoder( \_:) می بایست دارای پیاده سازی زیر باشد:

```
func encodeWithCoder(aCoder: NSCoder) {
aCoder.encodeObject(name, forKey: PropertyKey.nameKey)
aCoder.encodeObject(photo, forKey: PropertyKey.photoKey)
aCoder.encodeInteger(rating, forKey: PropertyKey.ratingKey)
}
```

پس از تنظیم متدی که عملیات کدگذاری را انجام می دهند، نوبت به پیاده سازی متد سازنده (initializer) می رسد که داده های کدگذاری شده را از حالت encoded خارج ساخته و به اصطلاح رمز گشایی (decode) می کند.

به منظور پیاده سازی متد سازنده (initializer) جهت بارگذاری آیتم مورد نظر (meal)، مراحل زیر را دنبال نمایید:

۱. در زیر متد encodeWithCoder( \_:)، متد سازنده ی زیر را اضافه نمایید:

```
required convenience init?(coder aDecoder: NSCoder) {
}
```

کلیدواژه `required` ی اعلان می کند که این متد سازنده بایستی در تمامی کلاس های مشتق شده از آن کلاس (کلاسی که متد سازنده به همراه این کلیدواژه در آن تعریف شده) پیاده سازی شود.

کلیدواژه `convenience` نشانگر این است که متد سازنده ی حاضر یک `initializer` فرعی و پشتیبان بوده و در نهایت بایستی `initializer` اصلی یا به اصطلاح `designated initializer` متد سازنده ی اولیه و اصلی کلاس است که تمامی `property` های آن را مقداردهی اولیه کرده، سپس `initializer` کلاس پدر (`superclass`) را صدا می زند تا پروسه ی مقداردهی اولیه ی `property` ها همین طور تا بالای زنجیره ی ارث بری (`superclass chain`) ادامه یابد. متد سازنده ی جاری را به این علت با کلیدواژه `convenience` تعریف یا علامت گذاری کردید که منحصر در صورت وجود داده های ذخیره شده برای بارگذاری صدا خورده و اعمال می شود.

علامت "?" بیانگر این است متد سازنده ی مورد نظر از نوع `failable` است و ممکن است در مقداردهی اولیه با شکست مواجه شده، در خروجی مقدار `nil` را برگرداند.

۲. خط زیر را اضافه نمایید:

```
let name = aDecoder.decodeObjectForKey(PropertyKey.nameKey) as! String
```

متد `decodeObjectForKey(_)`، اطلاعات ذخیره شده در رابطه با آبجکت مورد نظر را از حالت `archive` (کد گذاری و ذخیره شده بر روی دیسک) خارج می سازد. خروجی این متد `AnyObject` است که شما به واسطه ی عملیات `downcast` در کد بالا، به یک `String` تبدیل نموده و بعد داخل ثابت `name` قرار می دهید.

با دقت در کد بالا متوجه می شوید که مقدار بازگشتی توسط عملگر `as!` تبدیل می شود. علت استفاده از عملگر تبدیل مزبور این است که اگر آبجکت قابل تبدیل به `String` نبوده



یا دارای مقدار nil بود، آنگاه طبق انتظار، شما می دانید که خطایی رخ داده و برنامه باید در زمان اجرا به طور ناگهانی از کار بیافتد (runtime crash).

۳. در زیر خط قبلی، کد زیر را درج نمایید:

۴.

// Because photo is an optional property of Meal, use conditional cast.

```
let photo = aDecoder.decodeObjectForKey(PropertyKey.photoKey) as?
UIImage
```

این دستور خروجی متد (decodeObjectForKey(\_:)) را به کلاس UIImage تبدیل (downcast) کرده و متعاقباً مقدار تبدیل شده را در ثابت photo ذخیره می کند. همان طور که در کد مشاهده می کنید، این بار تبدیل با عملگر "as?" صورت پذیرفته است چرا که photo یک متغیر از نوع optional است. بدین معنی که می تواند حاوی مقدار از جنس کلاس UIImage باشد یا اصلاً مقداری نداشته و nil باشد. شما می بایست هر دو سناریو را در نظر گرفته و تمهیدات لازم را برای آن ها بیاندیشید.

۵. در زیر دستور قبلی، کد زیر را وارد نمایید:

```
let rating = aDecoder.decodeIntegerForKey(PropertyKey.ratingKey)
```

متد (decodeIntegerForKey(\_:)) یک مقدار عدد صحیح (integer) را از حالت archive و رمزنگاری شده خارج می سازد. از آنجایی که خروجی تابع مذکور از نوع عدد صحیح یا int است، لزومی ندارد این خروجی را که اکنون مقدار رمزگشایی شده است، downcast نمایید.

در انتهای پیاده سازی، کد زیر را وارد نمایید:

// Must call designated initializer.

```
self.init(name: name, photo: photo, rating: rating)
```

به عنوان یک convenience initializer، این متد سازنده می بایست در انتهای پیاده سازی، یکی از initializer های اصلی کلاس (designated initializer) را صدا بزند. مقادیر ثوابتی که به هنگام archive داده های ذخیره شده (تبدیل

آبجکت به فرمتی که قابل ذخیره سازی و انتقال بین اپلیکیشن ها باشد) ایجاد نمودید را اکنون به عنوان آرگومان ورودی به متد سازنده (initializer) پاس دهید.

متد سازنده ی (coder:) init? اکنون می بایست دارای پیاده سازی زیر باشد:

```
required convenience init?(coder aDecoder: NSCoder) {
    let name = aDecoder.decodeObjectForKey(PropertyKey.nameKey) as! String
    // Because photo is an optional property of Meal, use conditional cast.
    let photo = aDecoder.decodeObjectForKey(PropertyKey.photoKey) as?
    UIImage
    let rating = aDecoder.decodeIntegerForKey(PropertyKey.ratingKey)
    // Must call designated initializer.
    self.init(name: name, photo: photo, rating: rating)
}
```

از آنجایی که متد سازنده ی دیگری (initializer) که در سطح کلاس Meal تعریف کردید، (designated initializer) init?(name:photo:rating:)، یک متد سازنده ی اصلی است و با کلیدواژه ی required تعریف شده، این متد می بایست داخل بدنه ی خود متد سازنده ی کلاس والد (superclass) را نیز صدا بزند.

جهت فراخوانی متد سازنده ی کلاس پدر (superclass initializer) در بدنه ی متد سازنده ی جاری، مراحل زیر را دنبال نمایید:

```
init?(name: String, photo: UIImage?, rating: Int) {
    // Initialize stored properties.
    self.name = name
    self.photo = photo
    self.rating = rating
    // Initialization should fail if there is no name or if the rating is negative.
    if name.isEmpty || rating < 0 {
        return nil
    }
}
```

۱. در زیر خط `self.rating = rating`، متد سازنده ی کلاس پدر را به صورت زیر فراخوانی نمایید:

`super.init()`

متد (name:photo:rating:) init? اکنون می بایست دارای پیاده سازی زیر باشد:

```
init?(name: String, photo: UIImage?, rating: Int) {
```

```
// Initialize stored properties.
```

```
self.name = name
```

```
self.photo = photo
```

```
self.rating = rating
```

```
super.init()
```

```
// Initialization should fail if there is no name or if the rating is negative.
```

```
if name.isEmpty || rating < 0 {
```

```
return nil
```

```
}
```

```
}
```

در گام بعدی، شما به محلی ثابت و ماندگار (persistent path) بر روی file system نیاز دارید که داده ها در آن ذخیره شده و بعده ها از آن بارگذاری شود. بدین وسیله شما می دانید دقیقاً در کجا داده های مورد نظر را جستجو کرده و واکشی نمایید.

به منظور ایجاد محل ثابت و مشخص جهت ذخیره ی داده ها (file path)، مراحل زیر را دنبال نمایید.

- داخل فایل Meal.swift، در زیر بخش MARK: Properties، کد زیر را اضافه نمایید:

```
// MARK: Archiving Paths
```

```
static let DocumentsDirectory =
```

```
NSFileManager().URLsForDirectory(.DocumentDirectory, inDomains:
```

```
.UserDomainMask).first!
```

```
static let ArchiveURL =
```

```
DocumentsDirectory.URLByAppendingPathComponent("meals")
```

با دقت در مثال بالا متوجه می شوید که این ثوابت به همراه کلیدواژه ی static

تعریف و علامت گذاری شده اند. کلیدواژه ی ذکر شده بیانگر این است که ثوابت

فوق متعلق به خود کلاس جاری هستند، نه نمونه ای از آن کلاس. به منظور دسترسی به محل ذخیره سازی داده های Meal از بیرون این کلاس، از ساختار دستوری `Meal.ArchiveURL.path!` استفاده می شود (= اسم کلاس، عملگر نقطه، اسم متغیر، عملگر نقطه، خصیصه ی `path`).

**تست کنید:** با فشردن کلیدهای Command-B برنامه ی خود را کامپایل (build) نمایید. اپلیکیشن می بایست طبق انتظار و بدون هیچ مشکلی کامپایل شود.

### ذخیره و بارگذاری لیست نمایش غذاها (Meal list)

با افزودن قابلیت ذخیره و بارگذاری غذای جدید به اپلیکیشن FoodTracker، ناگزیر می بایست زمانی که کاربر یک آیتم جدید اضافه یا آیتم جاری را ویرایش/حذف می کند، لیست غذاها (meal list) را نیز همزمان ذخیره نماید.

به منظور پیاده سازی متد لازم جهت ذخیره سازی لیست غذاها، مراحل زیر را دنبال نمایید:

۱. فایل `MealTableViewController.swift` را باز نمایید.

۲. داخل فایل مزبور، خط زیر را به قبل از آخرین `}`، اضافه نمایید:

`// MARK: NSCoder`

این خط صرفاً یک `comment` است و به کسی کد برنامه ی شما را می خواند اعلان می دارد که این بخش مربوط به ذخیره سازی دائمی و ماندگار سازی داده های اپلیکیشن (data persistence) می باشد.

۳. حال در زیر `comment`، متد زیر را اضافه نمایید:

```
func saveMeals() {
}
```

۴. داخل بدنه ی متد `saveMeals()`، کد زیر را اضافه نمایید:

```
let isSuccessfulSave = NSKeyedArchiver.archiveRootObject(meals, toFile:
Meal.ArchiveURL.path!)
```

متد حاضر سعی دارد آرایه ی meals را بر روی مکان مشخصی در دیسک ذخیره کرده (archive)، سپس مقدار بولی true را در صورت امکان پذیر بودن عملیات بازگرداند. این دستور آدرس یا محل دقیق ذخیره ی داده های مورد نظر بر روی دیسک را از ثابت Meal.ArchiveURL که قبلاً در سطح کلاس Meal تعریف کردید، می خواند.

اکنون این سوال مطرح می شود: از کجا می توان اطمینان حاصل کرد داده ها با موفقیت ذخیره شده اند یا خیر؟ برای این منظور کافی است با استفاده از دستور print پیغامی را در console چاپ نمایید. به عنوان مثال، چنانچه عملیات ذخیره سازی داده ی مورد نظر با شکست مواجه شد، یک پیغام خطا به نشانه ی عدم موفقیت عملیات ذخیره سازی، در console چاپ می کنید.

۵. در زیر خط قبلی، دستور شرطی if زیر را اضافه نمایید.

```
if !isSuccessfulSave {
    print("Failed to save meals...")
}
```

پس از اعمال تغییرات فوق، اگر عملیات ذخیره با شکست مواجه شود، یک پیغام خطا در console چاپ می گردد.

در زمان حاضر متد saveMeals() می بایست دارای پیاده سازی زیر باشد:

```
func saveMeals() {
    let isSuccessfulSave = NSKeyedArchiver.archiveRootObject(meals, toFile:
    Meal.ArchiveURL.path!)
    if !isSuccessfulSave {
        print("Failed to save meals...")
    }
}
```

حال می بایست یک متد پیاده سازی کنید که داده های ذخیره شده را از روی دیسک خوانده و در اپلیکیشن FoodTracker بارگذاری نماید.

به منظور پیاده سازی متدی جهت بارگذاری و نمایش در لیست غذاها، مراحل زیر را طی نمایید:

۱. داخل فایل MealTableViewController.swift، قبل از آخرین ({}، متد زیر را درج

نمایید:

```
func loadMeals() -> [Meal]? {  
}
```

همان طور که می بینید خروجی این متد آرایه ای از آبجکت های Meal از نوع optional است (نوع بازگشتی آن optional array است). به عبارت بهتر، متد حاضر ممکن است در خروجی آرایه ای از آبجکت های Meal یا مقدار nil را بازگردانی نماید (= هیچ مقدار را برنگرداند).

۲. داخل بدنه ی متد loadMeals()، خط زیر را اضافه نمایید:

```
return NSKeyedUnarchiver.unarchiveObjectWithFile(Meal.ArchiveURL.path!)  
as? [Meal]
```

این متد سعی دارد آبجکت ذخیره شده در آدرس یا محل Meal.ArchiveURL.path! را از حالت archive خارج نموده، سپس آن آبجکت را به آرایه ای از آبجکت های Meal تبدیل (downcast) نماید. همان طور که در کد مشاهده می کنید، برای انجام عملیات تبدیل نوع از عملگر as? استفاده شده است. کد جاری در حقیقت با استفاده از این عملگر می تواند در صورت امکان پذیر نبودن تبدیل، مقدار nil را برگرداند. از آنجایی که آرایه ممکن است با موفقیت ذخیره شده/نشده باشد، فرایند downcast می تواند با شکست مواجه گردد که در آن صورت متد قاعدتا باید nil را بازگردانی نماید.

متد loadMeals() هم اکنون می بایست دارای پیاده سازی زیر باشد:

```
func loadMeals() -> [Meal]? {  
    return NSKeyedUnarchiver.unarchiveObjectWithFile(Meal.ArchiveURL.path!)  
    as? [Meal]  
}
```

پس از پیاده سازی متدهای فوق، کدی اضافه می کنید که به مجرد اینکه کاربر یک آیتم را اضافه، حذف یا ویرایش کرد، لیست غذاها را ذخیره و بار گذاری نماید.

برای اینکه به مجرد اضافه/حذف شدن یا ویرایش یک آیتم توسط کاربر، متعاقبا تغییرات اعمال شده در لیست غذاها ذخیره و ماندگار گردد، مراحل زیر را طی نمایید:

۱. داخل فایل MealTableViewController.swift، اکشن متد unwindToMealList(\_\_\_\_) را

را نیز فراخوانی نمایید:

```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
    if let sourceViewController = sender.sourceViewController as?
    MealViewController, meal = sourceViewController.meal {
        if let selectedIndexPath = tableView.indexPathForSelectedRow {
            // Update an existing meal.
            meals[selectedIndexPath.row] = meal
            tableView.reloadRowsAtIndexPaths([selectedIndexPath], withRowAnimation:
            .None)
        }
        else {
            // Add a new meal.
            let newIndexPath = NSIndexPath(forRow: meals.count, inSection: 0)
            meals.append(meal)
            tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation:
            .Bottom)
        }
    }
}
```

۲. درست پس از دستور شرطی else، کد زیر را اضافه نمایید:

```
// Save the meals.
saveMeals()
```

این کد، سبب می شود زمانی که کاربر یک آیتم جدید اضافه نموده یا آیتم جاری را ویرایش می کند، آرایه ی meals به تبع آن بر روی دیسک ذخیره گردد. این دستور می بایست داخل بدنه ی دستور if اصلی ({} ) که دیگر دستورات در دل آن گنجانده شده اند، درج شود.

۳. داخل فایل MealTableViewController.swift، متد

tableView(\_\_\_\_:commitEditingStyle:forRowAtIndexPath:\_\_\_\_) را پیدا کنید:

```
// Override to support editing the table view.
override func tableView(tableView: UITableView, commitEditingStyle
editingStyle: UITableViewCellStyle, forRowAtIndexPath indexPath:
NSIndexPath) {
if editingStyle == .Delete {
// Delete the row from the data source
meals.removeAtIndex(indexPath.row)
tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)
} else if editingStyle == .Insert {
// Create a new instance of the appropriate class, insert it into the array, and add
a new row to the table view
}
}
```

۴. بعد از خط `meals.removeAtIndex(indexPath.row)`، خط زیر را اضافه نمایید:

```
saveMeals()
```

این کد باعث می شود به مجرد اینکه یک آیتم از لیست غذاها حذف شد، آرایه ی `meals` (به همراه تغییراتی که به آن اعمال شده) بلافاصله ذخیره گردد.

بدنه ی متد `(: unwindToMealList)` در حال حاضر می بایست به شکل زیر باشد:

```
@IBAction func unwindToMealList(sender: UIStoryboardSegue) {
if let sourceViewController = sender.sourceViewController as?
MealViewController, meal = sourceViewController.meal {
if let selectedIndexPath = tableView.indexPathForSelectedRow {
// Update an existing meal.
meals[selectedIndexPath.row] = meal
tableView.reloadRowsAtIndexPaths([selectedIndexPath], withRowAnimation:
.None)
}
else {
// Add a new meal.
let newIndexPath = NSIndexPath(forRow: meals.count, inSection: 0)
meals.append(meal)
tableView.insertRowsAtIndexPaths([newIndexPath], withRowAnimation:
.Bottom)
}
}
```



// Save the meals.

```
saveMeals()
}
}
```

پیاده سازی متد tableView(\_:commitEditingStyle:forRowAtIndexPath:) نیز می بایست به صورت زیر باشد:

```
// Override to support editing the table view.
override func tableView(tableView: UITableView, commitEditingStyle
editingStyle: UITableViewCellEditingStyle, forRowAtIndexPath indexPath:
NSIndexPath) {
    if editingStyle == .Delete {
        // Delete the row from the data source
        meals.removeAtIndex(indexPath.row)
        saveMeals()
        tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)
    } else if editingStyle == .Insert {
        // Create a new instance of the appropriate class, insert it into the array, and add
        a new row to the table view
    }
}
```

پس از نوشتن کد لازم جهت ذخیره ی آیتم های مورد نظر در زمان های مناسب، می بایست اطمینان حاصل کنید که غذاها به موقع (در زمان مناسب) بارگذاری می شوند. این اتفاق می بایست هر زمان که meal list scene (صفحه محتوای لیست نمایش غذاها) لود شده و برای کاربر در آل نمایش داده می شود، رخ دهد. به عبارت روشن تر، مکان مناسب برای بارگذاری داده های ذخیره شده بر روی دیسک، داخل بدنه ی متد viewDidLoad می باشد.

جهت بارگذاری صفحه محتوای نمایش لیست غذاها (meal list) در زمان مربوطه، مراحل زیر را طی نمایید:

۱. داخل فایل MealTableViewController.swift، متد viewDidLoad() را پیدا کنید:

```
override func viewDidLoad() {
```

```

super.viewDidLoad()
// Use the edit button item provided by the table view controller.
navigationItem.leftBarButtonItem = editButtonItem()
// Load the sample data.
loadSampleMeals()
}

```

۲. در زیر خط `navigationItem.leftBarButtonItem = editButtonItem()` دستور شرطی زیر را وارد نمایید:

```

// Load any saved meals, otherwise load sample data.
if let savedMeals = loadMeals() {
    meals += savedMeals
}

```

چنانچه متد `loadMeals()` به طور موفقیت آمیز اجرا شده و در خروجی خود آرایه ای از آبجکت های Meal را بازگردانی نماید، این شرط برقرار بوده؛ به اصطلاح `true` می باشد و طبیعتاً دستور داخل بدنه ی `if` اجرا می شود. حال اگر متد مذکور `nil` را به عنوان خروجی برگرداند، مشخص می شود که هیچ آیتمی برای بارگذاری وجود نداشته و متعاقباً دستور درج شده داخل ساختمان `if` اجرا نمی شود. کد حاضر در واقع تمامی آیتم هایی که به طور کامل و با موفقیت بارگذاری شدند را به آرایه ی `meals` اضافه می کند.

۳. پس از دستور `if`، یک عبارت شرطی `else` اضافه نموده و به دنبال آن، خط فراخوانی متد `loadSampleMeals()` را به داخل بدنه ی `else` جابجا نمایید.

```

else {
// Load the sample data.
loadSampleMeals()
}

```

این کد تمامی آیتم هایی که بارگذاری شدند را به آرایه ی `meals` اضافه می کند.

متد `viewDidLoad()` هم اکنون می بایست دارای پیاده سازی زیر باشد:

```

override func viewDidLoad() {
super.viewDidLoad()
// Use the edit button item provided by the table view controller.

```

```

navigationItem.leftBarButtonItem = editButtonItem()
// Load any saved meals, otherwise load sample data.
if let savedMeals = loadMeals() {
    meals += savedMeals
} else {
    // Load the sample data.
    loadSampleMeals()
}
}

```

**تست کنید:** برنامه ی خود را اجرا کنید. حال اگر تعدادی آیتم جدید اضافه نموده، از اپلیکیشن خارج شوید و سپس دوباره برنامه را اجرا نمایید، آیتم هایی که قبلا اضافه کرده بودید، باید در لیست غذاها قابل مشاهده و دسترسی باشند.