Pobieranie danych do widoków kolekcji

W poprzedniej części poznałeś wzorzec projektowy Model-View-Controller (MVC) oraz widoki kolekcji. Wróciłeś również do ekranów Eksploruj i Lista restauracji i widziałeś, jak działają widoki kolekcji na obu ekranach. W tym momencie jednak oba ekrany wyświetlają tylko komórki, które nie zawierają żadnych danych. Jak pokazano w przewodniku po aplikacji w części 9 , ekran Eksploruj powinien wyświetlać listę kuchni, a ekran Lista restauracji powinien wyświetlać listę restauracji. W tej części zaimplementujesz obiekty modelu dla ekranu Eksploruj, aby wyświetlić listę kuchni. Zaczniesz od poznania modeli obiektów, których będziesz używać. Następnie dowiesz się o listach usług i zobaczysz, jak są one wykorzystywane do przechowywania danych kuchni, a także utworzysz strukturę Swift, która może przechowywać wystąpienia kuchni. Następnie utworzysz klasę menedżera danych, która odczytuje dane z listy właściwości i wypełni tablicę struktur. Ta tablica struktur będzie następnie używana jako źródło danych dla widoku kolekcji na ekranie Eksploruj. Pod koniec tego rozdziału dowiesz się, w jaki sposób listy właściwości są używane do przechowywania danych, jak tworzyć obiekty modelu, jak tworzyć klasę menedżera danych, która może ładować dane z listy właściwości do tablicy obiektów modelu, jak aby skonfigurować kontrolery widoku w celu udostępnienia obiektów modelu do widoku kolekcji oraz jak skonfigurować widoki kolekcji w celu wyświetlania danych na ekranie.

Omówione zostaną następujące tematy:

• Zrozumienie obiektów modelu

• Zrozumienie plików .plist

• Tworzenie struktury reprezentującej kuchnię

• Implementacja klasy menedżera danych do odczytu danych z pliku .plist

• Wyświetlanie danych w widoku kolekcji

Zrozumienie obiektów modelu

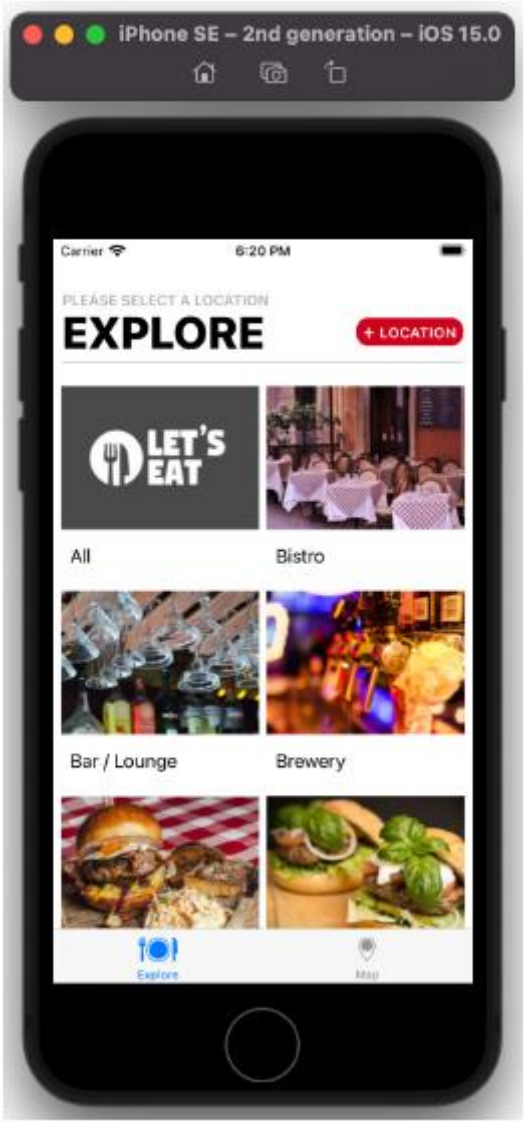
Jak dowiedziałeś się w części 13, Pierwsze kroki z widokami MVC i kolekcji, typowym wzorcem projektowym dla aplikacji na iOS jest Model-View-Controller lub MVC. Podsumowując, MVC dzieli aplikację na trzy różne części:

• Model: Obsługuje zadania związane z przechowywaniem, reprezentacją i przetwarzaniem danych.

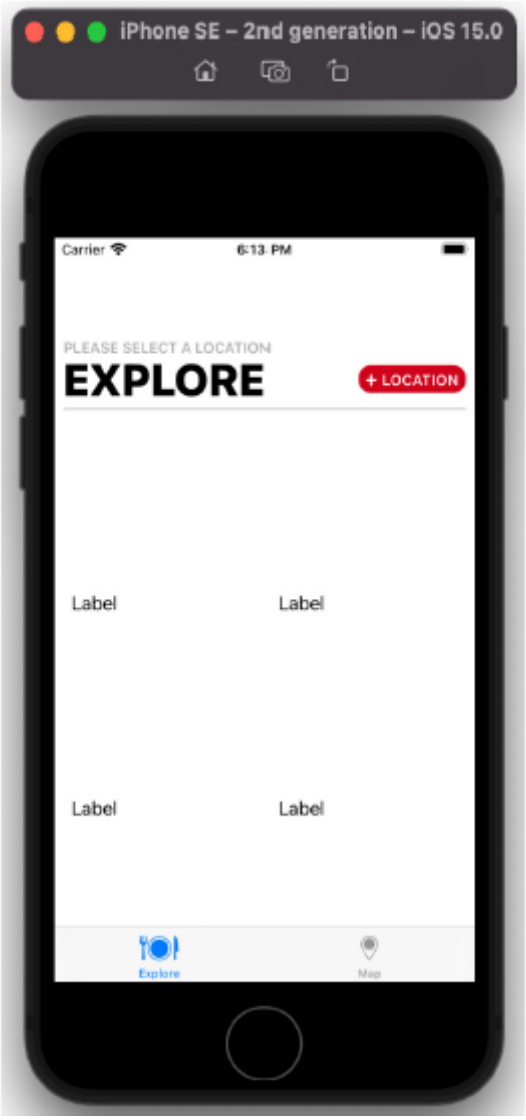
• Widok: to wszystko na ekranie, z którym użytkownik może wchodzić w interakcje.

• Kontroler: zarządza przepływem informacji między modelem a widokiem.

Przyjrzyjmy się projektowi ekranu Eksploruj, który widzieliście podczas prezentacji aplikacji, który wygląda tak:



Zbuduj i uruchom swoją aplikację, a ekran Eksploruj będzie wyglądał tak:



Jak widać, wszystkie komórki są obecnie puste. W oparciu o wzorzec projektowy MVC ukończono implementację widoków (nagłówek sekcji widoku kolekcji i widok kolekcji) oraz kontroler (klasa ExploreViewController). Teraz dodasz obiekty modelu, które dostarczą dane do wyświetlenia. Najpierw dodasz do swojego projektu listę właściwości, ExploreData.plist, która zawiera nazwę i nazwę pliku obrazu dla każdej kuchni. Same obrazy kuchni są już obecne w folderze Assets.xcassets. Następnie utworzysz obiekt modelowy ExploreItem, który będzie strukturą o dwóch właściwościach. Jedna właściwość będzie używana do przechowywania nazw kuchni, a druga do przechowywania nazw plików obrazów. Następnie utworzysz klasę menedżera danych, ExploreDataManager, która załaduje dane z pliku ExploreData.plist, umieści je w tablicy instancji ExploreItem i dostarczy tablicę do instancji ExploreViewController. Na koniec zmodyfikujesz klasę ExploreViewController, aby mogła dostarczać dane do wyświetlenia w widoku kolekcji. Aby dowiedzieć się więcej o plikach list właściwości, dodajmy teraz plik ExploreData.plist do projektu i zobaczmy, jak przechowuje on dane kuchni.

Zrozumienie plików .plist

Firma Apple opracowała listy właściwości do przechowywania struktur danych lub stanów obiektów w celu późniejszej rekonstrukcji i transmisji. Są one powszechnie używane do przechowywania preferencji aplikacji. Pliki list właściwości mają rozszerzenie nazwy pliku .plist, dlatego często są nazywane plikami .plist. W swoim projekcie użyjesz pliku .plist zawierającego dane kuchni,

ExploreData.plist. Aby pobrać ExploreData, musisz pobrać pakiet kodu dla tej książki. plik plist. Następnie możesz użyć Xcode, aby wyświetlić jego zawartość. Wykonaj następujące kroki:

1. Jeśli jeszcze tego nie zrobiłeś, pobierz pliki zasobów i ukończ projekt Xcode z tego linku: https://github.com/PacktPublishing/iOS-15-Programming-for-Beginners-Sixth-Edition.

2. Otwórz folder Chapter14 i zajrzyj do folderu zasobów, aby znaleźć plik ExploreData.plist. Ten plik przechowuje nazwy kuchni i nazwy plików obrazów.

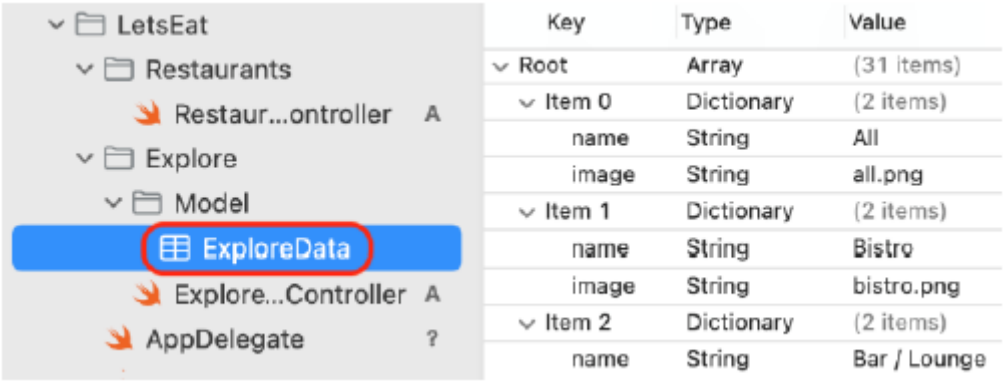
3. Otwórz projekt LetsEat, kliknij prawym przyciskiem myszy folder Eksploruj w nawigatorze projektów i wybierz Nowa grupa.

4. Zmień nazwę nowej grupy, którą właśnie dodałeś Model.

5. Przeciągnij plik ExploreData.plist do tej grupy.

6. Upewnij się, że opcja Kopiuj elementy w razie potrzeby jest zaznaczona i kliknij przycisk Zakończ.

Po kliknięciu ExploreData.plist w nawigatorze projektów zobaczysz tablicę zawierającą słowniki, jak pokazano:



Każdy słownik ma dwa elementy. Pierwszy element ma klucz, nazwę i wartość opisującą rodzaj kuchni. Drugi element ma klucz, obraz i wartość zawierającą nazwę pliku obrazu kuchni. Wszystkie obrazy kuchni są przechowywane w zasobach. xcassets w Twoim projekcie. Aby użyć danych zawartych w ExploreData.plist, musisz utworzyć strukturę do przechowywania ich w aplikacji, aby można było uzyskać do nich dostęp później przez wystąpienie ExploreViewController. Stworzysz go w następnej sekcji.

Tworzenie struktury reprezentującej kuchnię

Aby utworzyć obiekt modelowy, który może reprezentować kuchnię w Twojej aplikacji, dodasz do projektu nowy plik ExploreItem i zadeklarujesz strukturę ExploreItem, która ma właściwości nazwy i obrazu kuchni. Wykonaj następujące kroki:

1. Kliknij prawym przyciskiem myszy folder Model i wybierz Nowy plik.

2. iOS powinien być już wybrany. Wybierz Plik Swift, a następnie kliknij Dalej.

3. Nazwij plik ExploreItem, a następnie kliknij Utwórz. Pojawi się on w nawigatorze projektu, a jego zawartość pojawi się w obszarze edytora. Jedyny wiersz w tym pliku to instrukcja importu.

4. Dodaj następujący kod do pliku, aby zadeklarować strukturę o nazwie ExploreItem:

```
struct ExploreItem {
```

}

5. Dodaj następujący kod przed ostatnim nawiasem klamrowym, aby dodać dwie opcjonalne właściwości String do struktury ExploreItem:

Struct ExploreItem {

let name: String?

let image: String?

}

Właściwość name będzie przechowywać nazwę kuchni, a właściwość image będzie przechowywać nazwę pliku obrazu z pliku Assets.xcassets.

Struktury automatycznie otrzymują domyślny inicjator. Możesz utworzyć wystąpienie ExploreItem za pomocą następującej instrukcji:

let myExploreItem = ExploreItem(name:"name", image:"image")

Jednak nazwa i nazwa pliku obrazu w ExploreData.plist są przechowywane jako elementy słownika, jak pokazano w poniższym przykładzie:

["name": "All", "image": "all.png"]

Utworzysz niestandardowy inicjator, który przyjmuje słownik jako parametr i przypisuje wartości uzyskane z elementów słownika do właściwości w wystąpieniu ExploreItem. Użyjesz rozszerzenia, aby dodać ten niestandardowy inicjator do struktury ExploreItem.

Wykonaj następujące kroki:

1. Wpisz następujące polecenie po deklaracji struktury ExploreItem, aby dodać rozszerzenie:

extension ExploreItem {

}

Użyjesz tego rozszerzenia, aby dodać metodę inicjatora do struktury ExploreItem.

2. Dodaj następujące informacje między nawiasami klamrowymi rozszerzenia, aby zadeklarować niestandardową metodę inicjatora:

init(dict: [String: String]) {

}

Zignoruj wyświetlany błąd, ponieważ wkrótce go naprawisz. Ten inicjator przyjmuje słownik jako argument. Zauważ, że zarówno klucz, jak i wartość są ciągami.

3. Dodaj następujące informacje między nawiasami klamrowymi inicjatora:

self.name = dict["name"]

self.image = dict["image"]

To przypisuje wartość elementu słownika z kluczem „name” do właściwości name, a wartość elementu słownika z kluczem „image” do właściwości obrazu. Ukończone rozszerzenie powinno wyglądać następująco:

```
extension ExploreItem {

init(dict: [String: String]) {

self.name = dict["name"]

self.image = dict["image"]

}

}
```

W tym momencie masz strukturę ExploreItem z dwiema właściwościami String, name i image. Tworząc instancję tej struktury, przekażesz słownik zawierający elementy z kluczami i wartościami typu String. Wartość klucza nazwy zostanie przypisana do właściwości name, a wartość klucza obrazu zostanie przypisana do właściwości obrazu. W następnej sekcji zaimplementujesz klasę menedżera danych. Spowoduje to odczytanie tablicy słowników z pliku ExploreData.plist i przypisanie wartości elementów słownika do instancji ExploreItem.

Implementacja klasy menedżera danych do odczytu danych z pliku .plist

Do projektu dodano plik .plist ExploreData.plist zawierający dane kuchni i utworzyłeś strukturę ExploreItem, która może przechowywać szczegóły każdej kuchni. Teraz musisz utworzyć nową klasę ExploreDataManager, która może odczytywać dane z pliku .plist i przechowywać je w tablicy instancji ExploreItem. Będziesz odnosić się do tej klasy jako do klasy menedżera danych. Wykonaj następujące kroki:

1. Kliknij prawym przyciskiem myszy folder Model i wybierz Nowy plik.

2. iOS powinien być już wybrany. Wybierz Plik Swift, a następnie kliknij Dalej.

3. Nazwij plik ExploreDataManager, a następnie kliknij Utwórz, aby wyświetlić jego zawartość w obszarze Edytora.

4. Wpisz w pliku następujący kod, aby zadeklarować klasę ExploreDataManager:

```
class ExploreDataManager {

}
```

5. Wpisz następujący kod między nawiasami klamrowymi. Implementuje to metodę loadData(), która odczyta zawartość pliku ExploreData.plist i zwróci tablicę słowników:

```
private func loadData() -> [[String: String]] {

let decoder = PropertyListDecoder()

if let path = Bundle.main.path(forResource:

"ExploreData", ofType: "plist"),

let exploreData = FileManager.default.contents(

atPath: path),

let exploreItems = try? decoder.decode([[String:

String]].self, from: exploreData) {
```

return exploreItems

}

return [[:]]

}

Rozbijmy tę metodę:

private

Słowo kluczowe private oznacza, że metoda może być używana tylko w tej klasie.

func loadData() -> [[String: String]]

Deklaracja metody loadData() nie ma argumentów i zwraca tablicę słowników, a każdy słownik zawiera elementy z kluczami i wartościami typu String.

let decoder = PropertyListDecoder()

Ta instrukcja tworzy instancję dekodera listy właściwości, który będzie używany do dekodowania danych w pliku ExploreData.plist.

if let path = Bundle.main.path(forResource:

"ExploreData", ofType: "plist"),

Wynikiem tworzenia aplikacji jest folder zawierający wszystkie znajdujące się w nim zasoby aplikacji, zwany pakietem aplikacji. ExploreData.plist znajduje się w tym pakiecie. Ta instrukcja próbuje uzyskać ścieżkę do pliku ExploreData.plist i przypisać ją do stałej ścieżki.

let exploreData = FileManager.default.contents(atPath: path),

Ta instrukcja próbuje pobrać plik ExploreData.plist przechowywany w ścieżce i przypisać go do stałej exploreData.

let exploreItems = try? decoder.decode([[String:

String]].self, from: exploreData) {

Jeśli klikniesz ExploreData.plist w swoim projekcie, zauważ, że obiekt poziomu głównego jest tablicą, a każdy element tablicy jest słownikiem, jak pokazano:

Key	Type	Value
⌄ Root	Array	(31 items)
⌄ Item 0	Dictionary	(2 items)
name	String	All
image	String	all.png

Ta instrukcja próbuje utworzyć tablicę z zawartości ExploreData. plist i przypisz go do stałej, exploreItems:

return exploreItems

}

Jeśli opcjonalne wiązanie się powiedzie, ta instrukcja zwraca exploreItems jako tablicę słowników, a każdy słownik jest typu

the [String: String] type:

return [[:]]

}

Jeśli opcjonalne wiązanie nie powiedzie się, zwracana jest pusta tablica słowników. W tym momencie masz klasę menedżera danych, ExploreDataManager, zawierającą metodę, która ładuje dane z pliku ExploreData.plist i przypisuje je do tablicy exploreItems. W następnej sekcji przyjrzymy się, jak używać słowników w tej tablicy do inicjowania instancji ExploreItem, które ostatecznie zostaną przekazane do instancji ExploreViewController zarządzającej ekranem Explore.

Używanie menedżera danych do inicjowania instancji ExploreItem

Obecnie masz klasę menedżera danych, ExploreDataManager. Ta klasa zawiera metodę loadData(), która odczytuje dane z pliku ExploreData.plist i zwraca tablicę słowników. Dodasz metodę, która tworzy i inicjuje wystąpienia ExploreItem ze słownikami w tej tablicy. Wykonaj następujące kroki:

1. W definicji klasy ExploreDataManager dodaj następujący kod przed metodą loadData(), aby zaimplementować metodę fetch():

func fetch() {

for data in loadData() {

print(data)

}

}

Ta metoda wywoła metodę loadData(), która zwróci tablicę słowników. Pętla for jest następnie używana do drukowania zawartości każdego słownika w tablicy w obszarze Debug, aby upewnić się, że plik ExploreData.plist został pomyślnie odczytany.

2. Kliknij plik ExploreViewController w nawigatorze projektów. Dodaj następujący kod do metody viewDidLoad(). Spowoduje to utworzenie instancji ExploreDataManager i wywołanie jej metody fetch(), gdy instancja ExploreViewController ładuje swój widok:

override func viewDidLoad() {

super.viewDidLoad()

let manager = ExploreDataManager()

manager.fetch()

}

Zbuduj i uruchom swoją aplikację. Zawartość pliku ExploreData.plist jest odczytywana i drukowana w obszarze Debug, jak pokazano:

Teraz przypiszesz ciągi nazw i obrazów z każdego słownika w tablicy do wystąpienia ExploreItem.

3. Kliknij plik ExploreDataManager. Dodaj deklarację właściwości, aby przechowywać tablicę instancji ExploreItem tuż przed metodą fetch() :

private var exploreItems: [ExploreItem] = []

Spowoduje to dodanie właściwości exploreItems do klasy ExploreDataManager i przypisanie do niej pustej tablicy.

4. Wewnątrz metody fetch() zastąp instrukcję print() następującą instrukcją, która inicjuje instancje ExploreItem i dołącza je do tablicy exploreItems:

exploreItems.append(ExploreItem(dict: data))

Niestandardowy inicjator w klasie ExploreItem przypisuje ciągi nazwy i obrazu w każdym słowniku odczytanym z pliku ExploreData.plist do właściwości nazwy i obrazu wystąpienia ExploreData.

5. Sprawdź, czy zawartość pliku ExploreDataManager wygląda tak:

```
import Foundation

class ExploreDataManager {

private var exploreItems: [ExploreItem] = []

func fetch() {

for data in loadData() {

exploreItems.append(ExploreItem(dict:

data))

}

}

private func loadData() -> [[String: String]] {

let decoder = PropertyListDecoder()

if let path = Bundle.main.path(forResource:

"ExploreData", ofType: "plist"),

let exploreData =
```

```
FileManager.default.contents(atPath: path),

let exploreItems = try?

decoder.decode([[String: String]].self,

from: exploreData) {

return exploreItems

}

return [[:]]

}

}
```

W tym momencie masz klasę ExploreDataManager, która odczytuje dane z pliku ExploreData.plist i przechowuje je w exploreItems, tablicy wystąpień ExploreItem. Ta tablica będzie źródłem danych dla widoku kolekcji zarządzanego przez wystąpienie ExploreViewController. Zawarte w nim informacje o kuchni zostaną ostatecznie wyświetlone na ekranie Eksploruj.

Wyświetlanie danych w widoku kolekcji

Zaimplementowano klasę menedżera danych, ExploreDataManager, która odczytuje dane kuchni z pliku ExploreData.plist i przechowuje je w tablicy wystąpień ExploreItem. Teraz zmodyfikujesz klasę ExploreViewController, aby używała tej tablicy jako źródła danych dla widoku kolekcji na ekranie Eksploruj. Obecnie widok kolekcji na ekranie Eksploruj wyświetla 20 komórek widoku kolekcji, przy czym każda komórka zawiera pusty widok obrazu i etykietę. Potrzebujesz sposobu na ustawienie wartości dla widoku obrazu i etykiety w komórkach, więc w tym celu utworzysz nową klasę ExploreCell. Następnie można skonfigurować kontroler widoku dla widoku kolekcji ExploreViewController, aby uzyskać szczegóły kuchni z wystąpienia ExploreDataManager i przekazać je do widoku kolekcji do wyświetlenia. Aby utworzyć ExploreCell, wykonaj następujące kroki:

1. Kliknij prawym przyciskiem myszy folder Eksploruj w nawigatorze projektów i wybierz Nowa grupa.

2. Zmień nazwę nowej grupy Widok.

3. Kliknij prawym przyciskiem myszy folder Widok i wybierz Nowy plik.

4. iOS powinien być już wybrany. Wybierz opcję Cocoa Touch Class, a następnie kliknij przycisk Dalej.

5. Skonfiguruj klasę, jak pokazano tutaj:

Class: ExploreCell

Subclass: UICollectionViewCell

Also create XIB: Unchecked

Language: Swift

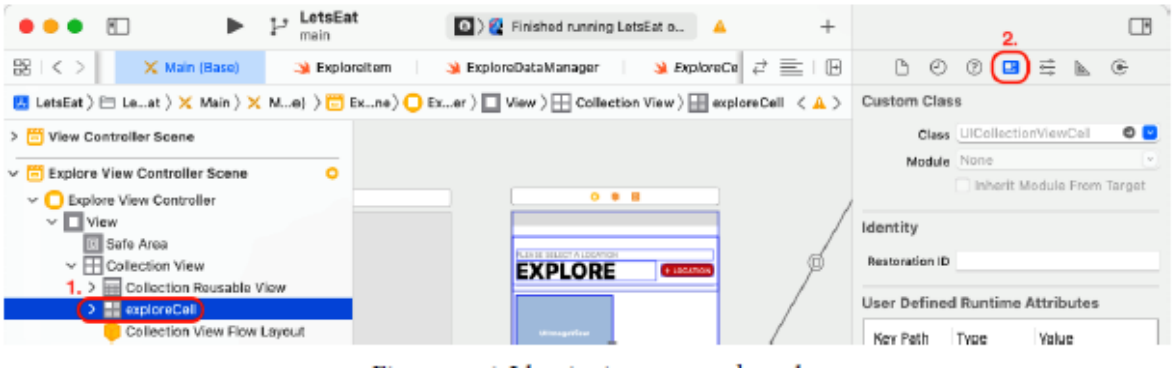
Click Next.

6. Kliknij Utwórz.

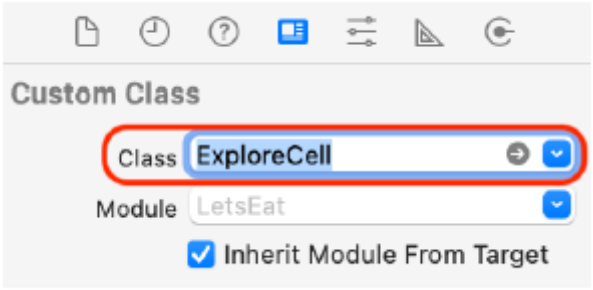
7. Nowy plik, ExploreCell, zostanie dodany do twojego projektu. Wewnątrz zobaczysz następujące elementy:

```
import UIKit

class ExploreCell: UICollectionViewCell {

}
```

8. Teraz przypiszesz klasę ExploreCell jako tożsamość komórki widoku kolekcji exploreCell. Kliknij plik głównego scenorysu w nawigatorze projektów i kliknij exploreCell wewnątrz sceny Eksploruj kontroler widoku w konspekcie dokumentu. Kliknij przycisk Inspektora tożsamości:



9. W sekcji Custom Class ustaw Class na ExploreCell. Ustawia to wystąpienie ExploreCell jako menedżera exploreCell. Po zakończeniu naciśnij Return:



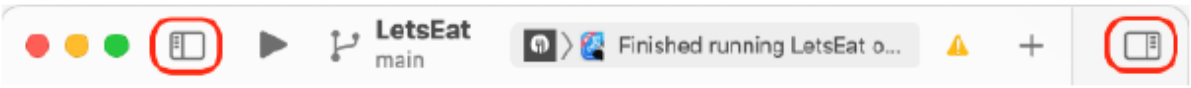
Właśnie zadeklarowałeś i zdefiniowałeś klasę ExploreCell i przypisałeś ją jako menedżera komórki widoku kolekcji exploreCell. Teraz utworzysz punkty sprzedaży w tej klasie, które będą połączone z widokiem obrazu i etykietą w komórce widoku kolekcji exploreCell, dzięki czemu możesz kontrolować, co wyświetlają.

Podłączanie gniazdek w exploreCell

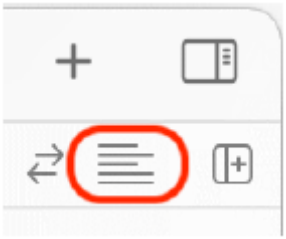
Aby zarządzać tym, co jest wyświetlane przez komórki widoku kolekcji w

na ekranie Eksploruj połączysz widok obrazu i etykietę w komórce widoku kolekcji exploreCell z gniazdami w klasie ExploreCell. Użyjesz do tego edytora asystenta. Wykonaj następujące kroki:

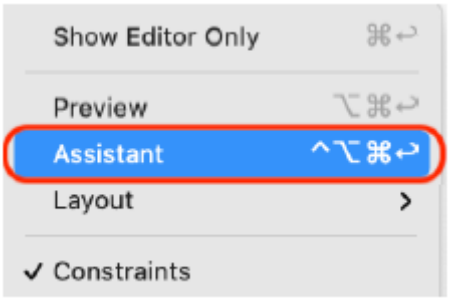
1. Kliknij przyciski Nawigator i Inspektor, aby ukryć obszary Nawigator i Inspektor. To da ci więcej miejsca do pracy:



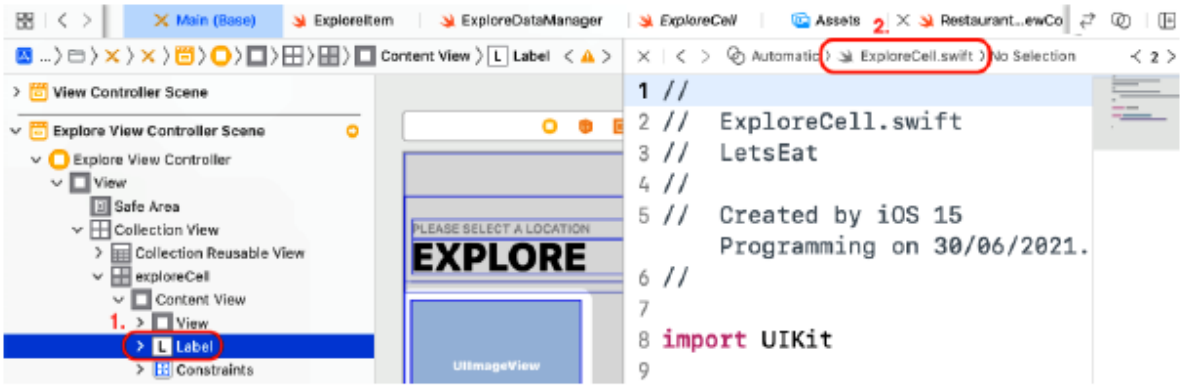
2. Kliknij przycisk Dostosuj opcje edytora, aby wyświetlić menu:



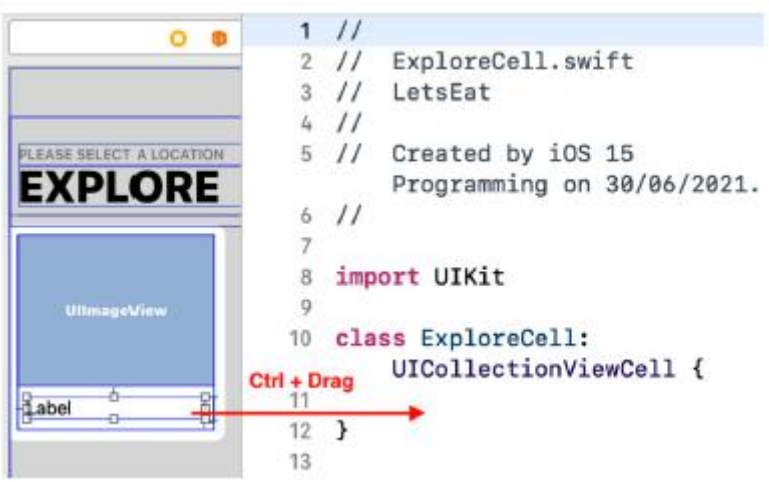
3. Wybierz Asystenta z menu, aby wyświetlić edytor asystenta:



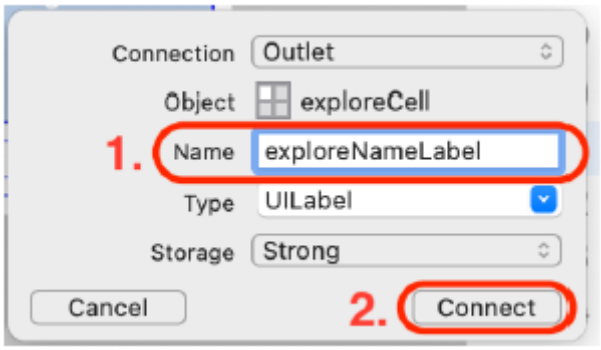
4. Aby utworzyć gniazda dla klasy ExploreCell, ścieżka edytora asystenta powinna być ustawiona na Automatyczne | PrzeglądajCell.swift. Jeśli nie widzisz ExploreCell. swift w ścieżce, wybierz etykietę komórki widoku kolekcji exploreCell w konspekcie dokumentu i wybierz opcję ExploreCell.swift z menu rozwijanego ścieżki edytora asystenta:



5. Ctrl + przeciągnij z elementu Label w komórce widoku kolekcji do przestrzeni między nawiasami klamrowymi, jak pokazano. Stwarza to dla niego ujście:

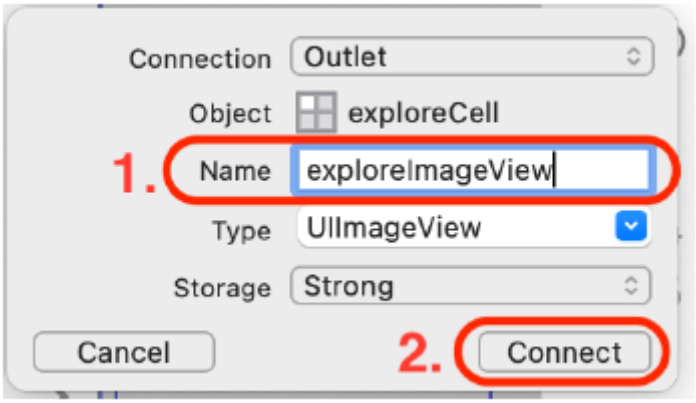


6. W wyskakującym oknie dialogowym wpisz exploreNameLabel w polu Nazwa, aby ustawić nazwę gniazdka:



7. Ctrl + przeciągnij z elementu UIImageView w komórce widoku kolekcji do miejsca tuż za właściwością exploreNameLabel. Stwarza to dla niego ujście:

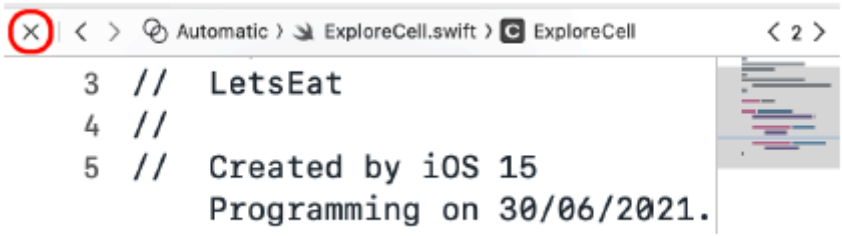
8. W wyskakującym oknie dialogowym wpisz exploreImageView w polu Nazwa, aby ustawić nazwę gniazda:



9. Wyloty exploreNameLabel i exploreImageView zostały dodane do klasy ExploreCell i połączone z widokiem obrazu i etykietą komórki widoku kolekcji exploreCell, jak pokazano:

```
7
8  import UIKit
9
10 class ExploreCell: UICollectionViewCell {
11
       @IBOutlet var exploreNameLabel:
           UILabel!
13
       @IBOutlet var exploreImageView:
           UIImageView!
15
```

10. Kliknij przycisk x, aby zamknąć edytor asystentów:



Komórka widoku kolekcji exploreCell w pliku głównego scenorysu została teraz skonfigurowana z klasą ExploreCell. Utworzono i przypisano również punkty sprzedaży widoku obrazu i etykiety komórki widoku kolekcji. Teraz możesz ustawić punkty sprzedaży exploreNameLabel i exploreImageView w instancji ExploreCell, aby wyświetlać obraz i nazwę kuchni w każdej komórce, gdy aplikacja jest uruchomiona.

Wskazówka : Możesz sprawdzić, czy gniazda są prawidłowo podłączone w Inspektorze połączeń.

W następnej sekcji dodasz kod do ExploreDataManager, aby podać liczbę komórek, które mają być wyświetlane przez collectionView, i udostępnisz wystąpienie ExploreItem, którego właściwości będą używane do określenia, jaki obraz i etykieta będzie wyświetlana w komórce.

Wdrażanie dodatkowych metod zarządzania danymi

Jak dowiedziałeś się w części 13, Wprowadzenie do widoków MVC i kolekcji, widok kolekcji musi wiedzieć, ile komórek ma zostać wyświetlonych i co umieścić w każdej komórce. Dodasz dwie metody do klasy ExploreDataManager, które zapewnią liczbę wystąpień ExploreItem w tablicy exploreItems i zwrócą wystąpienie ExploreItem w określonym indeksie tablicy. Kliknij plik ExploreDataManager w nawigatorze projektów i dodaj te dwie metody po metodzie loadData():

```
func numberOfExploreItems() -> Int {

exploreItems.count

}

func exploreItem(at index: Int) -> ExploreItem {

exploreItems[index]

}
```

Pierwsza metoda, numberOfExploreItems(), określi liczbę komórek wyświetlanych w widoku kolekcji. Druga metoda, exploreItem(at:), zwróci instancję ExploreItem, która odpowiada pozycji komórki w widoku kolekcji. W następnej sekcji zaktualizujesz metody źródła danych widoku kolekcji w klasie ExploreViewController, aby podać liczbę komórek do wyświetlenia w widoku kolekcji oraz podać nazwę kuchni i obraz dla każdej komórki.

Aktualizacja metod źródła danych w ExploreViewController

Metody źródła danych w klasie ExploreViewController są obecnie ustawione na wyświetlanie 20 komórek, z których każda zawiera pusty widok obrazu i etykietę. Zaktualizujesz je, aby uzyskać liczbę komórek do wyświetlenia i dane do umieszczenia w każdej komórce z instancji ExploreDataManager. Wykonaj następujące kroki:

1. Kliknij plik ExploreViewController i znajdź metodę viewDidLoad(). To powinno wyglądać tak:

```
override func viewDidLoad() {

super.viewDidLoad()

let manager = ExploreDataManager()

manager.fetch()

}
```

Oznacza to, że instancja ExploreDataManager jest dostępna tylko w ramach metody viewDidLoad(). Musisz udostępnić go całej klasie.

2. Przenieś wiersz let manager = ExploreDataManager() tuż za deklaracją właściwości collectionView, aby instancja ExploreDataManager była dostępna dla wszystkich metod w klasie ExploreViewController, jak pokazano:

```
@IBOutlet var collectionView: UICollectionView!

let manager = ExploreDataManager()
```

3. Zaktualizuj collectionView(_:numberOfItemsInSection:), jak pokazano. Spowoduje to, że widok kolekcji wyświetli komórkę dla każdego elementu w tablicy items:

```
func collectionView(_ collectionView: UICollectionView,
Sekcja numberOfItemsInSection: Int) -> Int {
manager.numberOfExploreItems()
}
```

4. Zaktualizuj collectionView(_:cellForItemAt:), jak pokazano, aby ustawić widok obrazu i etykietę dla każdej komórki przy użyciu danych z odpowiedniego elementu w tablicy elementów:

```
func collectionView(_ collectionView: UICollectionView,
cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
let cell = collectionView.dequeueReusableCell(
withReuseIdentifier: "exploreCell", for:
indexPath) as! ExploreCell
let exploreItem = manager.exploreItem(at:
indexPath.row)
cell.exploreNameLabel.text = exploreItem.name
cell.exploreImageView.image = UIImage(named:
exploreItem.image!)
return cell
}
```

Rozbijmy to:

```
let cell = collectionView.dequeueReusableCell(
withReuseIdentifier:"exploreCell", for: indexPath)
as! ExploreCell
```

Określa, że komórka, która jest usuwana z kolejki, jest wystąpieniem ExploreCell.

```
let exploreItem = manager.exploreItem(at: indexPath.row)
```

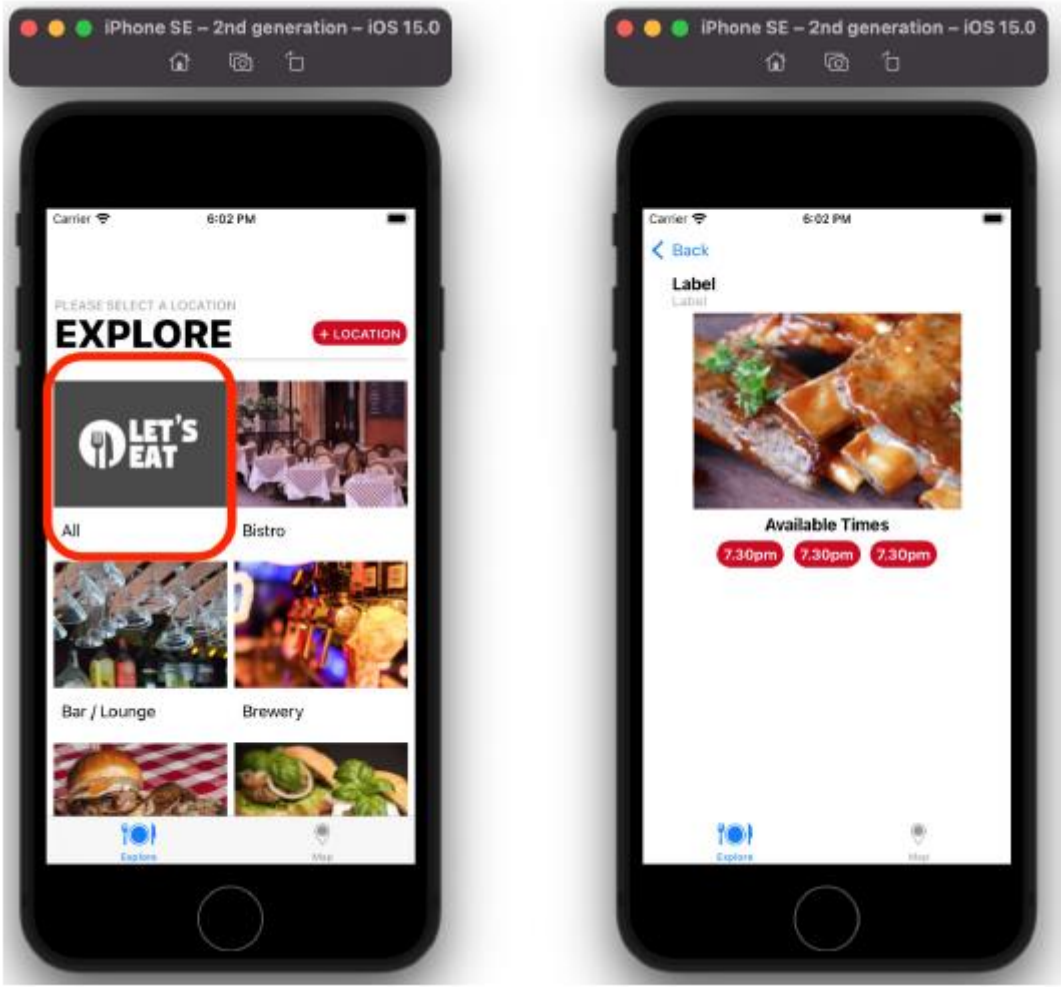
Pobiera wystąpienie ExploreItem odpowiadające bieżącej komórce w widoku kolekcji. Innymi słowy, pierwsza komórka w widoku kolekcji odpowiada pierwszemu wystąpieniu ExploreItem w tablicy exploreItems, druga komórka odpowiada drugiemu wystąpieniu ExploreItem i tak dalej.

```
cell.exploreNameLabel.text = exploreItem.name
```

Ustawia właściwość tekstową nameLabel komórki na nazwę wystąpienia ExploreItem.

```
cell.exploreImageView.image = UIImage(named: exploreItem.image!)
```

Pobiera ciąg obrazu z wystąpienia ExploreItem, pobiera odpowiedni obraz z pliku Assets.xcassets i przypisuje go do obrazu właściwości imgExplore komórki. Zbuduj i uruchom aplikację. Zobaczysz widok kolekcji na ekranie Eksploruj, wyświetlając obrazy i teksty różnych kuchni. Dotknięcie komórki widoku kolekcji spowoduje wyświetlenie ekranu Lista restauracji:



W tym momencie ekran Eksploruj wyświetla obrazy i teksty różnych kuchni na podstawie danych uzyskanych z pliku ExploreData.plist. W rozdziale 17, Pierwsze kroki z plikami JSON, zmodyfikujesz klasę RestaurantListViewController, aby ekran listy restauracji wyświetlał listę restauracji oferujących wybraną kuchnię. Ale zanim to zrobisz, musisz ustawić lokalizację na ekranie Lokalizacje, który wyświetli listę wszystkich dostępnych restauracji w tej lokalizacji**.**

Podsumowanie

Dodałeś do swojego projektu plik listy właściwości, ExploreData.plist. Zaimplementowano strukturę ExploreItem, obiekty modelu dla ekranu Eksploruj. Utworzono klasę menedżera danych, ExploreDataManager, do odczytywania danych z pliku ExploreData.plist, umieszczania danych w tablicy wystąpień ExploreItem i dostarczania ich do ExploreViewController. Utworzyłeś klasę dla komórki widoku kolekcji exploreCell. Na koniec skonfigurowałeś metody źródła danych w klasie ExploreViewController, aby użyć danych z tablicy wystąpień ExploreItem do wypełnienia widoku kolekcji, a ekran Eksploruj wyświetla teraz listę kuchni. Dobra robota! Wiesz już, jak dostarczać dane do aplikacji przy użyciu plików .plist, tworzyć obiekty modelu, tworzyć klasy menedżera danych, które ładują pliki .plist do obiektów modelu, konfigurować widoki kolekcji w celu wyświetlania załadowanych danych i konfigurować kontrolery widoku dla widoków kolekcji . Będzie to przydatne, jeśli chcesz

tworzyć własne aplikacje korzystające z widoków kolekcji. W następnym rozdziale przyjrzysz się widokom tabeli, które są pod pewnymi względami podobne do widoków kolekcji, i skonfigurujesz ekran Lokalizacje, aby wyświetlał listę lokalizacji w widoku tabeli po dotknięciu przycisku LOKALIZACJA na ekranie Eksploruj.