Pierwsze kroki z widokami MVC i kolekcji

W poprzedniej Części zmodyfikowałeś komórki na ekranie Eksploruj, na ekranie Lista restauracji i ekranie Lokalizacje, aby dopasować je do przewodnika po aplikacji w Rozdziale 9, Konfiguracja interfejsu użytkownika. Ukończyłeś wstępny interfejs użytkownika aplikacji Let's Eat. Teraz skupisz się na kodzie, dzięki któremu Twoja aplikacja będzie działać. Dowiesz się o wzorcu projektowym Model-View-Controller (MVC) oraz o tym, jak różne części aplikacji współdziałają ze sobą. Następnie programowo zaimplementujesz widok kolekcji (co oznacza wdrożenie go przy użyciu kodu zamiast scenorysów) przy użyciu placu zabaw, aby zrozumieć, jak działają widoki kolekcji. Na koniec wrócisz do widoków kolekcji zaimplementowanych na ekranach Eksploruj i Lista restauracji, aby zobaczyć, jakie są różnice między implementacją ich w scenorysie a implementacją programową. Pod koniec tego rozdziału zrozumiesz wzorzec projektowy MVC, dowiesz się, jak programowo utworzyć kontroler widoku kolekcji oraz jak używać delegatów widoku kolekcji i protokołów źródła danych. Omówione zostaną następujące tematy:

• Zrozumienie wzorca projektowego Model-Widok-Kontroler

• Odkrywanie kontrolerów i klas

Zrozumienie wzorca projektowego Model-View-Controller

Wzorzec projektowy Model-View-Controller (MVC) jest typowym podejściem używanym do tworzenia aplikacji dla systemu iOS. MVC dzieli aplikację na trzy różne części:

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

• Widok: Obejmuje wszystkie rzeczy, które są na ekranie, z którymi użytkownik może wchodzić w interakcje.

• Kontroler: zarządza przepływem informacji między modelem a widokiem.

Jedną z godnych uwagi cech MVC jest to, że widok i model nie współdziałają ze sobą; zamiast tego cała komunikacja jest zarządzana przez kontroler. Na przykład wyobraź sobie, że jesteś w restauracji. Patrzysz na menu i wybierasz coś, co chcesz. Następnie przychodzi kelner, przyjmuje zamówienie i wysyła je kucharzowi. Kucharz przygotowuje zamówienie, a gdy jest gotowe, kelner przyjmuje zamówienie i przynosi je Tobie. W tym scenariuszu menu to widok, kelner to kontroler, a kucharz to model. Pamiętaj też, że wszystkie interakcje między tobą a kuchnią odbywają się tylko przez kelnera; nie ma interakcji między tobą a kucharzem. Aby zobaczyć, jak działa MVC, dowiedzmy się więcej o kontrolerach i klasach. Zobaczysz, jak zaimplementować kontroler widoku, który jest wymagany do zarządzania widokiem kolekcji, który jest używany na ekranie Eksploruj i na ekranie Lista restauracji.

Odkrywanie kontrolerów i klas

Do tej pory zaimplementowano sceny kontrolera widoku w pliku Main storyboard za pomocą Interface Builder. Dodano do projektu ExploreViewController, kontroler widoku, który zarządza widokiem kolekcji na ekranie Eksploruj, oraz RestaurantListViewController, kontroler widoku, który zarządza widokiem kolekcji na ekranie listy restauracji. Jednak nadal nie nauczyłeś się, jak działa kod dodany do każdego kontrolera widoku, więc spójrzmy na to teraz. Po uruchomieniu typowej aplikacji iOS ładowany jest kontroler widoku dla pierwszego wyświetlanego ekranu. Kontroler widoku ma właściwość widoku i automatycznie ładuje wystąpienie widoku przypisane do jego właściwości widoku. Widok ten może mieć podwidoki, które również są ładowane. Jeśli jeden z widoków podrzędnych jest widokiem kolekcji, będzie miał właściwości dataSource i delegata. Właściwość dataSource jest

przypisana do obiektu, który dostarcza dane do widoku kolekcji. Właściwość delegata jest przypisana do obiektu, który obsługuje interakcję użytkownika z widokiem kolekcji. Zazwyczaj kontroler widoku dla widoku kolekcji zostanie również przypisany do właściwości dataSource i delegata widoku kolekcji. Wywołania metod, które widok kolekcji wyśle do swojego kontrolera widoku, są zadeklarowane w protokołach UICollectionViewDataSource i UICollectionViewDelegate. Pamiętaj, że protokoły dostarczają tylko deklaracje metod; implementacja tych wywołań metod znajduje się w kontrolerze widoku. Kontroler widoku następnie dostarczy żądane dane dla widoku kolekcji lub obsłuży interakcję użytkownika. Przyjrzyjmy się bliżej widokom kolekcji i protokołom widoku kolekcji w następnej sekcji.

Zrozumienie poglądów kolekcji

Widok kolekcji wyświetla uporządkowaną kolekcję komórek widoku kolekcji przy użyciu dostosowywanych układów. Układ widoku kolekcji jest podyktowany przez UICollectionViewFlowLayout. Określa kierunek przepływu i rozmiar elementów w widoku kolekcji. Dane wyświetlane przez widok kolekcji są zwykle dostarczane przez kontroler widoku. Kontroler widoku udostępniający dane dla widoku kolekcji musi być zgodny z protokołem UICollectionViewDataSource. Ten protokół deklaruje listę metod, które informują widok kolekcji, ile komórek wyświetlić i co wyświetlić w każdej komórce. Obejmuje również tworzenie i konfigurację dodatkowych widoków (takich jak nagłówek sekcji widoku kolekcji). Aby zapewnić interakcję z użytkownikiem, kontroler widoku dla widoku kolekcji musi również być zgodny z protokołem UICollectionViewDelegate, który deklaruje listę metod, które są wyzwalane, gdy użytkownik wchodzi w interakcję z widokiem kolekcji. Aby zrozumieć, jak działają widoki kolekcji, zaimplementujesz kontroler widoku, który kontroluje widok kolekcji na Twoim placu zabaw CollectionViewBasics. Następnie porównasz to z implementacją dla kontrolerów widoku na ekranach Eksploruj i Lista restauracji w następnej sekcji. Ponieważ na placu zabaw nie ma scenorysu, nie można dodawać elementów interfejsu użytkownika tak, jak to zrobiłeś w poprzednich rozdziałach. Zamiast tego dodasz je programowo. Zaczniesz od utworzenia klasy CollectionViewExampleController, implementacji kontrolera widoku, który zarządza widokiem kolekcji. Następnie utworzysz wystąpienie CollectionViewExampleController i sprawisz, że będzie wyświetlał widok kolekcji zawierający pojedynczą komórkę widoku kolekcji w widoku na żywo placu zabaw. Wykonaj następujące kroki:

1. Otwórz swój plac zabaw CollectionViewBasics, który utworzyłeś na początku tego rozdziału. Na samej górze placu zabaw usuń instrukcję var i dodaj instrukcję import PlaygroundSupport. Twój plac zabaw powinien teraz zawierać następujące elementy:

```
import UIKit

import PlaygroundSupport
```

Pierwsza instrukcja importu importuje interfejs API do tworzenia aplikacji na iOS. Druga instrukcja umożliwia placu zabaw wyświetlanie podglądu na żywo, którego użyjesz do wyświetlenia widoku kolekcji.

2. Dodaj następujący kod po instrukcjach importu, aby zadeklarować klasę CollectionViewExampleController:

```
class CollectionViewExampleController: UIViewController {

}
```

Ta klasa jest podklasą UIViewController, klasy udostępnianej przez firmę Apple do zarządzania widokami na ekranie.

3. Dodaj następujący kod wewnątrz nawiasów klamrowych, aby dodać opcjonalną właściwość collectionView do klasy CollectionViewExampleController:

var collectionView: UICollectionView?

Instancja widoku kolekcji zostanie później przypisana do tej właściwości.

4. Sprawdź, czy Twój kod wygląda tak:

class CollectionViewExampleController: UIViewController {

var collectionView: UICollectionView?

}

W następnej sekcji dowiesz się, jak ustawić liczbę komórek, które mają być wyświetlane w widoku kolekcji, oraz jak ustawić zawartość każdej komórki.

Zgodny z protokołem UICollectionViewDataSource

Widok kolekcji wyświetla na ekranie siatkę komórek widoku kolekcji. Jednak zanim to zrobi, musi wiedzieć, ile komórek ma wyświetlić i co umieścić w każdej komórce. Aby udostępnić te informacje w widoku kolekcji, należy sprawić, by klasa CollectionViewExampleController była zgodna z protokołem UICollectionViewDataSource. Ten protokół ma dwie wymagane metody:

• collectionView(_:numberOfItemsInSection:) jest wywoływana przez widok kolekcji w celu określenia, ile komórek widoku kolekcji ma być wyświetlanych.

• collectionView(_:cellForItemAt:) jest wywoływana przez widok kolekcji w celu określenia, co ma być wyświetlane w każdej komórce widoku kolekcji.

Dodajmy trochę kodu, aby CollectionViewExampleController był zgodny z protokołem UICollectionViewDataSource. Wykonaj następujące kroki:

1. Aby CollectionViewExampleController przyjął protokół UICollectionViewDataSource, wpisz przecinek po deklaracji nadklasy, a następnie wpisz UICollectionViewDataSource. Kiedy skończysz, twój kod powinien wyglądać następująco:

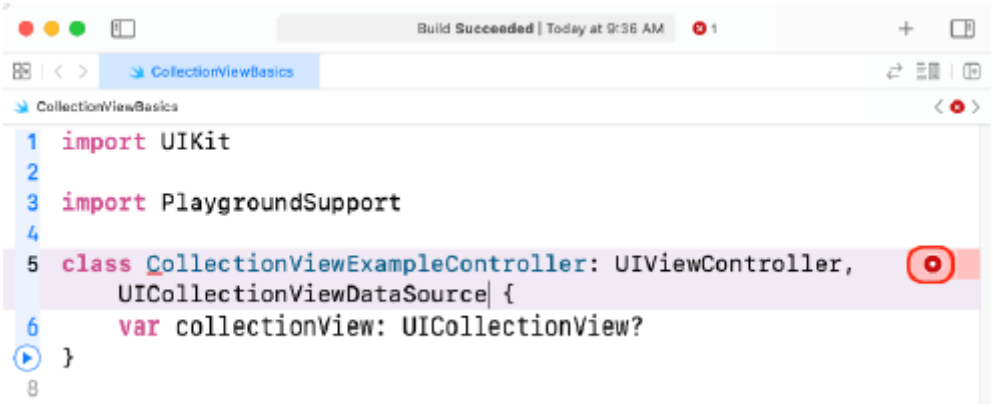
class CollectionViewExampleController: UIViewController,

UICollectionViewDataSource {

var collectionView:UICollectionView?

}

2. Pojawi się błąd, ponieważ nie zaimplementowałeś jeszcze dwóch wymaganych metod. Kliknij ikonę błędu:



3. Komunikat o błędzie informuje, że brakuje wymaganych metod dla protokołu UICollectionViewDataSource. Kliknij przycisk Napraw, aby dodać wymagane metody:

4. Sprawdź, czy Twój kod wygląda tak:

class CollectionViewExampleController: UIViewController,

UICollectionViewDataSource {

func collectionView(_ collectionView:

UICollectionView, numberOfItemsInSection

section: Int) -> Int {

code

}

func collectionView(_ collectionView:

UICollectionView, cellForItemAt indexPath:

IndexPath) -> UICollectionViewCell{

code

}

var collectionView:UICollectionView?

}

5. W definicji klasy konwencja nakazuje, że właściwości są deklarowane na początku klasy, przed deklaracjami metod. Zmień układ kodu tak, aby deklaracja właściwości collectionView znajdowała się na górze, w następujący sposób:

class CollectionViewExampleController: UIViewController,

UICollectionViewDataSource {

var collectionView:UICollectionView?

func collectionView(_ collectionView:

UICollectionView, numberOfItemsInSection

section: Int) -> Int {

code

}

6. W collectionView(_:numberOfItemsInSection:), kliknij słowo kod i typ 1. Ukończona metoda powinna wyglądać następująco:

func collectionView(_ collectionView:

UICollectionView, numberOfItemsInSection

section: Int) -> Int {

1

}

Dzięki temu instancja collectionView wyświetli pojedynczą komórkę widoku kolekcji. Zazwyczaj liczba wyświetlanych komórek zostanie podana przez obiekt modelu.

7. W collectionView(_:cellForItemAt:) kliknij kod słowa i zmodyfikuj metodę w następujący sposób:

func collectionView(_ collectionView:

UICollectionView, cellForItemAt indexPath:

IndexPath) -> UICollectionViewCell{

let cell = collectionView.dequeueReusableCell

(withReuseIdentifier: "BoxCell", for: indexPath)

cell.backgroundColor = .red

return cell

}

Oto jak działa ta metoda. Wyobraź sobie, że masz 1000 przedmiotów do wyświetlenia w widoku kolekcji. Nie potrzebujesz 1000 komórek widoku kolekcji; wystarczy tylko tyle, aby wypełnić ekran.

Komórki widoku kolekcji, które przewijają się u góry ekranu, mogą być ponownie używane do wyświetlania elementów, które pojawiają się u dołu ekranu. Aby upewnić się, że używasz właściwego typu komórki, używasz identyfikatora ponownego użycia do identyfikacji typu komórki. Identyfikator ponownego wykorzystania musi być zarejestrowany w widoku kolekcji, co zrobisz później. Kolejny wiersz kodu ustawia kolor tła komórki na czerwony, a kolejny wiersz zwraca komórkę, która jest następnie wyświetlana na ekranie. Proces powtarza się dla liczby komórek podanej w pierwszej metodzie, która w tym przypadku wynosi 1.

8. Sprawdź, czy klasa CollectionViewExampleController wygląda następująco:

```
class CollectionViewExampleController:

UIViewController, UICollectionViewDataSource {

var collectionView: UICollectionView?

func collectionView(_ collectionView:

UICollectionView, numberOfItemsInSection

section: Int) -> Int {

1

}

func collectionView(_ collectionView:

UICollectionView, cellForItemAt indexPath:

IndexPath) -> UICollectionViewCell{

let cell = collectionView.dequeueReusableCell

(withReuseIdentifier: "BoxCell", for: indexPath)

cell.backgroundColor = .red

return cell

}

}
```

Zakończyłeś implementację klasy CollectionViewExampleController. W następnej sekcji dowiesz się, jak utworzyć instancję tej klasy.

Tworzenie instancji CollectionViewExampleController

Teraz, po zadeklarowaniu i zdefiniowaniu klasy CollectionViewExampleController, napiszesz metodę, aby utworzyć jej instancję. Wykonaj następujące kroki:

1. Wpisz następujący kod po deklaracji zmiennej collectionView, aby zadeklarować nową metodę:

```
func createCollectionView() {

}
```

To deklaruje nową metodę, createCollectionView(), której użyjesz do utworzenia wystąpienia widoku kolekcji i przypisania go do właściwości collectionView.

2. Wpisz następujący kod po otwierającym nawiasie klamrowym, aby zdefiniować treść tej metody:

self.collectionView = UICollectionView(

frame: CGRect(x: 0, y: 0, width:

self.view.frame.width, height:

self.view.frame.height),

collectionViewLayout: UICollectionViewFlowLayout())

Spowoduje to utworzenie nowego wystąpienia widoku kolekcji i przypisanie go do collectionView. Wymiary tego widoku kolekcji są dokładnie takie same, jak jego widok otaczający, z domyślnym układem przepływu. Układ przepływu określa kolejność wyświetlania komórek widoku kolekcji, od lewej do prawej.

3. Przejdź do następnego wiersza, a następnie wpisz następujący kod, aby ustawić właściwość dataSource widoku kolekcji na instancję CollectionViewExampleController:

self.collectionView?.dataSource = self

Właściwość dataSource widoku kolekcji określi, który obiekt zawiera implementację wymaganych metod UIViewControllerDataSource.

4. Przejdź do następnej linii, a następnie wpisz następujący kod, aby ustawić kolor tła widoku kolekcji na biały:

self.collectionView?.backgroundColor = .white

5. Przejdź do następnego wiersza, a następnie wpisz następujący kod, aby ustawić identyfikator komórek widoku kolekcji w widoku kolekcji na BoxCell:

self.collectionView?.register(UICollectionViewCell.self,

forCellWithReuseIdentifier:"BoxCell")

Ten identyfikator będzie używany w metodzie collectionView(_:cellForItemAt:) do identyfikowania typu komórek widoku kolekcji do ponownego użycia.

6. Przejdź do następnej linii, a następnie wpisz poniższy kod, aby dodać widok kolekcji jako widok podrzędny do widoku instancji CollectionViewExampleController:

self.view.addSubview(self.collectionView!)

Gdy instancja kontrolera widoku jest ładowana do pamięci, jego widok jest również ładowany wraz z wszelkimi widokami podrzędnymi. W takim przypadku instancja CollectionViewExampleController automatycznie załaduje swój widok, a ponieważ widok kolekcji jest widokiem podrzędnym jego widoku, zostanie również załadowany widok kolekcji.

7. Sprawdź, czy ukończona metoda wygląda następująco:

func createCollectionView() {

self.collectionView = UICollectionView(frame:

```
CGRect(x: 0, y: 0, width: self.view.frame.width,

height: self.view.frame.height),

collectionViewLayout:

UICollectionViewFlowLayout())

self.collectionView?.dataSource = self

self.collectionView?.backgroundColor = .white

self.collectionView?.register(

UICollectionViewCell.self,

forCellWithReuseIdentifier: "BoxCell")

self.view.addSubview(self.collectionView!)

}
```

Teraz potrzebujesz odpowiedniego miejsca do wywołania tej metody. Kontrolery widoku mają właściwość widoku. Widok przypisany do właściwości widoku zostanie automatycznie załadowany po załadowaniu kontrolera widoku. Po pomyślnym załadowaniu widoku zostanie wywołana metoda viewDidLoad() kontrolera widoku. Zastąpisz metodę viewDidLoad() w klasie CollectionViewControllerExample, aby wywołać metodę createCollectionView(). Wykonaj następujące kroki:

1. Wpisz następujący kod tuż przed metodą createCollectionView():

```
override func viewDidLoad(){

super.viewDidLoad()

self.view.bounds = CGRect(x: 0, y: 0, width: 375,

height: 667)

createCollectionView()

}
```

Spowoduje to ustawienie rozmiaru widoku na żywo, utworzenie wystąpienia widoku kolekcji, przypisanie go do collectionView i dodanie go jako widoku podrzędnego do widoku wystąpienia CollectionViewExampleController. Widok kolekcji następnie wywołuje metody źródła danych, aby określić, ile komórek widoku kolekcji należy wyświetlić i co wyświetlić w każdej komórce. collectionView(_:numberOfItemsInSection:) zwraca 1, więc zostanie wyświetlona pojedyncza komórka widoku kolekcji. collectionView(_:cellForItemAt:) tworzy komórkę, ustawia kolor tła komórki na czerwony i zwraca ją do wyświetlenia.

2. Sprawdź, czy ukończony plac zabaw wygląda tak:

```
import UIKit

import PlaygroundSupport

class CollectionViewExampleController:
```

```
UIViewController, UICollectionViewDataSource{
var collectionView: UICollectionView?
override func viewDidLoad(){
super.viewDidLoad()
self.view.bounds = CGRect(x: 0, y: 0,
width: 375, height: 667)
createCollectionView()
}
func createCollectionView(){
self.collectionView = UICollectionView(frame:
CGRect(x: 0, y: 0, width: self.view.frame.width,
height: self.view.frame.height),
collectionViewLayout:
UICollectionViewFlowLayout())
self.collectionView?.dataSource = self
self.collectionView?.backgroundColor = .white
self.collectionView?.register(
UICollectionViewCell.self,
forCellWithReuseIdentifier: "BoxCell")
self.view.addSubview(self.collectionView!)
}
func collectionView(_ collectionView:
UICollectionView, numberOfItemsInSection
section: Int) -> Int {
1
}
func collectionView(_ collectionView:
UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell {
let cell = collectionView.dequeueReusableCell(
withReuseIdentifier: "BoxCell", for: indexPath)
```

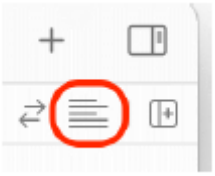
```
cell.backgroundColor = .red

return cell

}

}
```

3. Teraz nadszedł czas, aby zobaczyć to w akcji. Wpisz następujący kod po wszystkich innych kodach na placu zabaw:

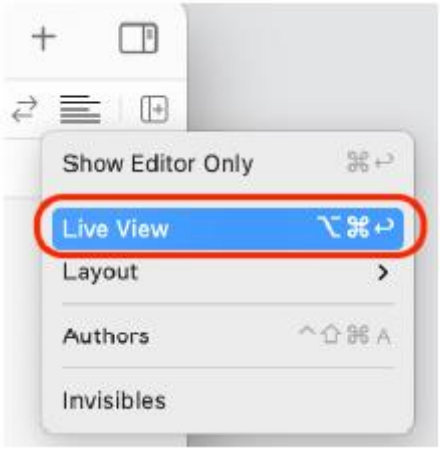
```
PlaygroundPage.current.liveView =

CollectionViewExampleController()
```

To polecenie tworzy wystąpienie CollectionViewExampleController i wyświetla jego widok w widoku na żywo placu zabaw. Metoda createCollectionView() utworzy widok kolekcji i doda go jako widok podrzędny do widoku instancji CollectionViewExampleController i pojawi się na ekranie.

4. Uruchom plac zabaw. Jeśli nie widzisz na ekranie reprezentacji widoku kolekcji, musisz włączyć widok na żywo na placu zabaw. Kliknij przycisk Dostosuj opcje edytora:



5. Upewnij się, że wybrany jest podgląd na żywo:



6. Zobaczysz widok kolekcji wyświetlający jedną czerwoną komórkę widoku kolekcji w widoku na żywo:

Właśnie utworzyłeś kontroler widoku dla widoku kolekcji, utworzyłeś jego instancję i wyświetliłeś widok kolekcji w widoku na żywo placu zabaw. Dobra robota!

Ponowne odwiedzanie ekranów Eksploruj i Lista restauracji

Pamiętasz klasę ExploreViewController dodaną wcześniej i klasę RestaurantListViewController? Obie są przykładami kontrolerów widoku, które zarządzają widokiem kolekcji. Zauważ, że kod w obu z nich jest bardzo podobny do tego na twoim placu zabaw. Różnice są następujące:

• Kolor tła komórki ustawia się programowo w kolekcjiView(_:cellForItemAt:), zamiast ustawiać go w Inspektorze atrybutów.

• Utworzono i przypisano widok kolekcji do właściwości collectionView w CollectionViewExampleController w sposób programowy.

• Wymiary widoku kolekcji ustawia się programowo w UICollectionView(frame: collectionViewLayout:), zamiast używać Inspektora rozmiaru.

• Połączyłeś wyjście źródła danych z kontrolerem widoku programowo, zamiast używać Inspektora połączeń.

• Ustawiasz kolor tła widoku kolekcji programowo, zamiast używać Inspektora atrybutów.

• Ustawiasz identyfikator ponownego użycia dla komórki widoku kolekcji programowo, zamiast używać Inspektora atrybutów.

• Dodano widok kolekcji jako widok podrzędny widoku dla CollectionViewExampleController zamiast przeciągania obiektu widoku kolekcji z biblioteki.

Podsumowanie

Szczegółowo poznałeś wzorzec projektowy MVC i kontrolery widoku kolekcji. Następnie ponownie odwiedziłeś widoki kolekcji używane na ekranach Eksploruj i Lista restauracji i dowiedziałeś się, jak one działają. Powinieneś teraz zrozumieć wzorzec projektowy MVC, jak utworzyć kontroler widoku kolekcji i jak używać protokołu źródła danych widoku kolekcji. Umożliwi to zaimplementowanie kontrolerów widoku kolekcji dla własnych aplikacji. Do tego momentu skonfigurowałeś widoki i kontrolery widoków dla ekranów Eksploruj i Lista restauracji, ale ekran Eksploruj wyświetla tylko siatkę komórek, a ekran Lista restauracji wyświetla pojedynczą komórkę z obrazem zastępczym. W następnym rozdziale zaimplementujesz obiekty modelu dla ekranu Eksploruj, aby wyświetlić listę kuchni. W tym celu odczytasz dane z pliku przechowywanego na urządzeniu z systemem iOS, utworzysz struktury do

przechowywania tych danych, a na koniec przekażesz je do instancji ExploreViewController, aby mogły być wyświetlane w widoku kolekcji na ekranie Eksploruj.