Steven F. Daniel

Xamarin

Tworzenie interfejsów użytkownika

Helion

Packt›

Tytuł oryginału: Mastering Xamarin UI Development

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-283-3948-4

Copyright © Packt Publishing 2017. First published in the English language under the title 'Mastering Xamarin Ul Development - (9781786462008)'

Polish edition copyright © 2018 by Helion SA
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/xamati.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/xamati
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

1

ROZDZIAŁ

Tworzenie macierzystej aplikacji TrackMyWalks

Wraz z pojawieniem się platformy **Xamarin** kilka lat temu programiści cieszą się możliwością tworzenia macierzystych aplikacji mobilnych przeznaczonych dla innych platform niż Microsoftu oraz możliwością wyboru języka C# lub F#, dzięki której mogą przygotowywać produkty zarówno dla systemu iOS, jak i Android.

Z książki tej nauczysz się najlepszych zasad tworzenia wieloplatformowych aplikacji mobilnych i poznasz wzorce projektowe na platformie `Xamarin.Forms` służącej do tworzenia wieloplatformowych interfejsów użytkownika, które mogą być wykorzystywane w systemach Android, iOS oraz Windows Phone.

Ponieważ każdą z tych aplikacji można napisać za pomocą jednego języka programowania, dobrym rozwiązaniem wydaje się utworzenie pojedynczej bazy kodu, przy użyciu której można następnie kompilować i budować osobne aplikacje dla każdej z wymienionych platform.

Na początku tego rozdziału dowiesz się, jak przygotować z wykorzystaniem `Xamarin.Forms` podstawową strukturę aplikacji, która będzie stanowiła podstawę pracy w następnych rozdziałach. Bazując na niej, będziemy rozbudowywać naszą aplikację, stosując w niej coraz to nowsze rozwiązania. W tym rozdziale pokazuję, jak utworzyć wstępną, niezależną od platformy, macierzystą aplikację `Xamarin.Forms` oraz jak dodawać nowe i aktualizować istniejące pakiety w swoim rozwiązaniu.

Nauczysz się tworzyć modele aplikacji w postaci klas C# oraz tworzyć strony z treścią, które będą stanowiły interfejs użytkownika. Na końcu tego rozdziału opisuję różnice w podejściu do tworzenia aplikacji przy użyciu środowisk Xamarin Studio i Microsoft Visual Studio.

Kup książkę Poleć książkę

W tym rozdziale:

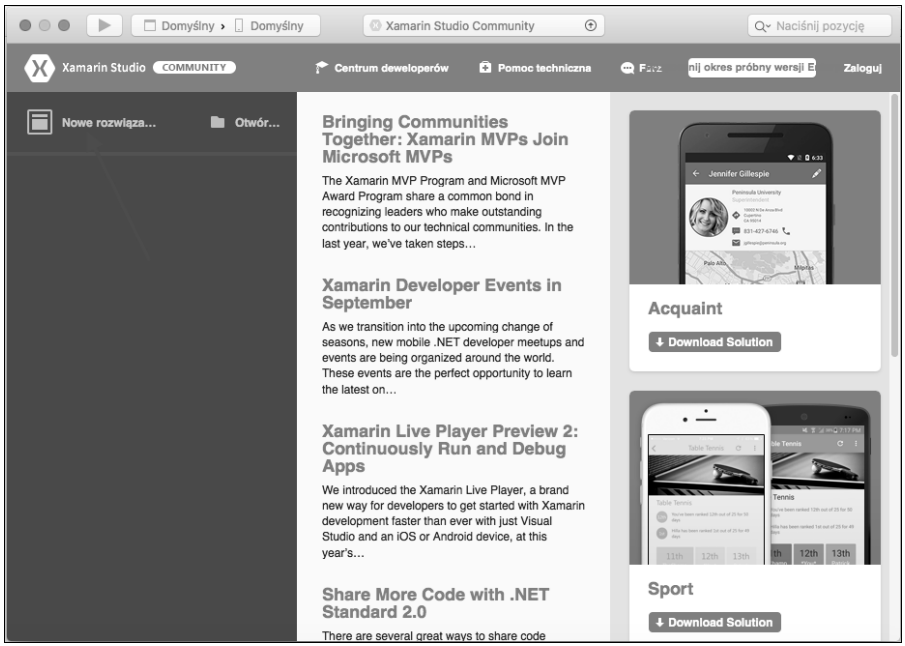
- Tworzenie mobilnej aplikacji `Xamarin.Forms` o nazwie *TrackMyWalks*.
- Sposób aktualizacji pakietów rozwiązania *TrackMyWalks* za pomocą menedżera pakietów *NuGet*.
- Tworzenie modelu danych `TrackMyWalks`.
- Metoda tworzenia stron treści (`ContentPage`) w rozwiązaniu *TrackMyWalks*.
- Różnice między środowiskami Xamarin Studio i Visual Studio.

Tworzenie rozwiązania TrackMyWalks

W tym podrozdziale pokażę Ci, jak utworzyć pierwsze rozwiązanie `Xamarin.Forms`. Zaczniemy od przygotowania podstawowej struktury naszej aplikacji, po czym dodamy niezbędne modele jednostek i zaprojektujemy pliki interfejsu użytkownika.

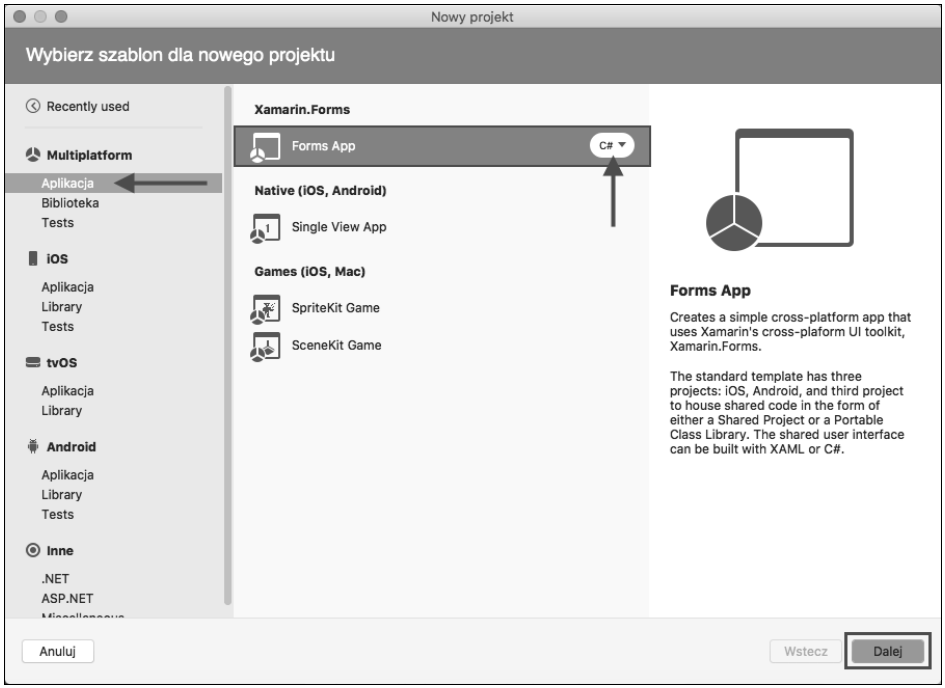
Pierwszą naszą czynnością będzie utworzenie projektu o nazwie *TrackMyWalks*. W Xamarin Studio projekty tworzy się bardzo łatwo. Wystarczy wykonać następujące czynności:

1. Uruchom aplikację Xamarin Studio. Na ekranie pojawi się widoczne poniżej okno:



Kup książkę Poleć książkę

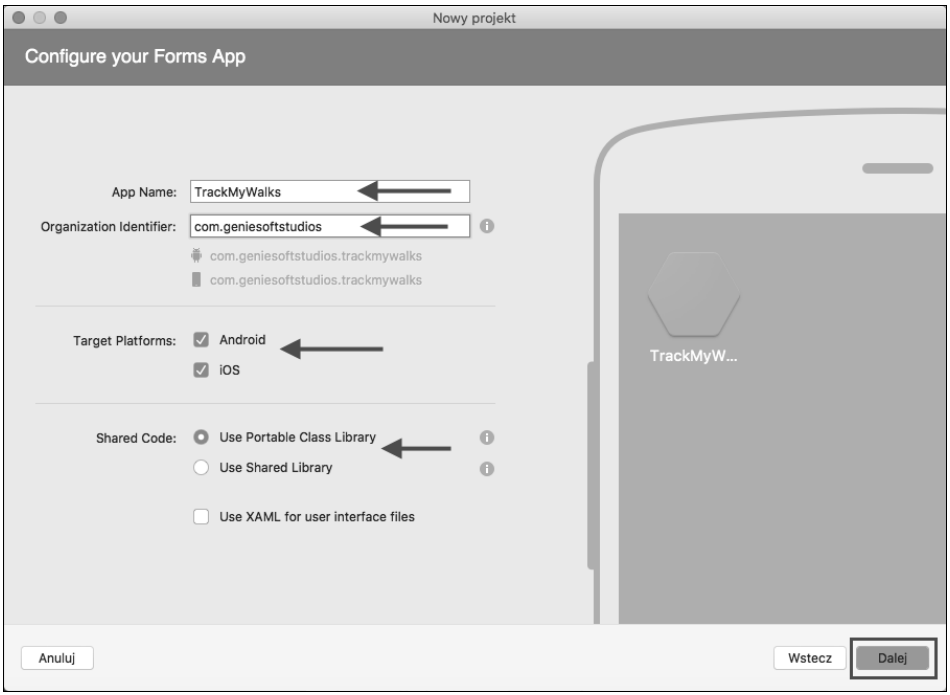
2. Kliknij przycisk *Nowe rozwiązanie* albo naciśnij kombinację klawiszy *Shift+Command+N*.
3. W sekcji *Multiplatform/Aplikacja* kliknij opcję *Forms App* (aplikacja Forms). Z listy rozwijanej widocznej po prawej stronie wybierz język programowania C#.



4. W polu *App Name* (nazwa aplikacji) wpisz nazwę swojej aplikacji: **`TrackMyWalks`**.
5. Następnie podaj nazwę w polu *Organization Identifier* (identyfikator organizacji).
6. W kolejnym kroku w polach *Target Platforms* (platformy docelowe) zaznacz pola wyboru *Android* i *iOS*, jeśli nie są jeszcze zaznaczone.
7. Następnie sprawdź, czy w sekcji *Shared Code* (wspólny kod) zaznaczona jest opcja *Use Portable Class Library* (użyj biblioteki klas przenośnych), i jeśli nie jest, to ją zaznacz, jak widać na poniższym zrzucie ekranu.

Różnica między Portable Class Library a Shared Library polega głównie na tym, że pierwsza opcja umożliwia napisanie kodu tylko raz i wykorzystywanie go na różnych platformach, takich jak serwisy internetowe czy systemy Android i iOS. Z kolei Shared Library umożliwia skopiowanie wszystkich plików należących do projektu biblioteki do wszystkich projektów znajdujących się w rozwiązaniu podczas kompilacji dla różnych platform, które będą z tej biblioteki korzystać.

Kup książkę Poleć książkę

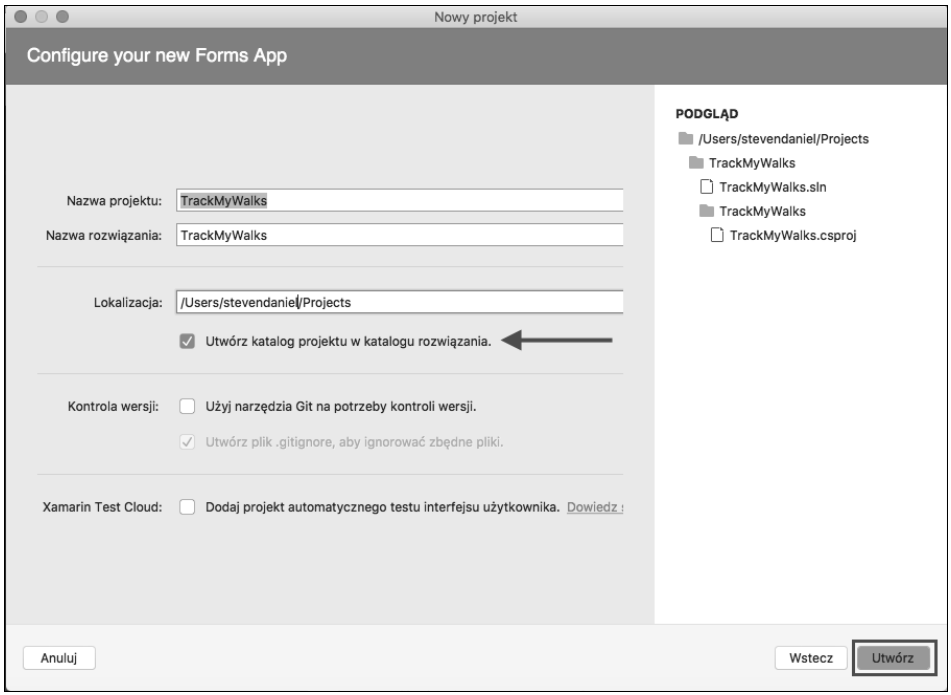


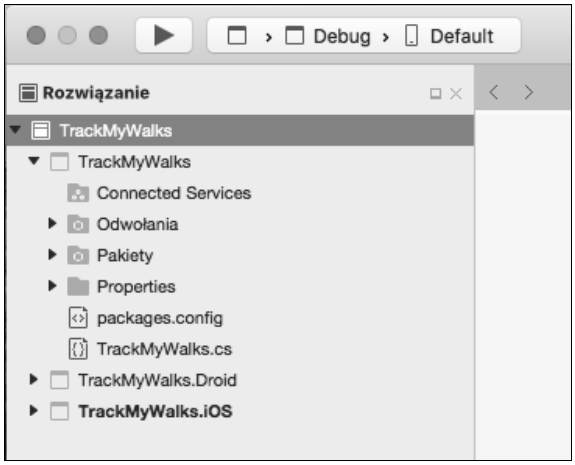
Nazwa wpisana w polu *Organization Identifier* nie może się powtarzać. W Xamarin Studio zaleca się stosowanie nazw odzwierciedlających odwróconą kolejność elementów adresu domeny, np. *com.nazwaDomeny.nazwaAplikacji*.

8. Wyłącz opcję *Use XAML for user interface files* (twórz pliki interfejsu użytkownika w języku XAML).
9. Kliknij przycisk *Dalej*, aby przejść do następnego etapu (zobacz pierwszy rysunek na następnej stronie).
10. Następnie zaznacz opcję *Utwórz katalog projektu w katalogu rozwiązania*.
11. Potem kliknij przycisk *Utwórz*, aby zapisać projekt w wybranym folderze.

Po utworzeniu projektu zostanie wyświetlone okno środowiska programistycznego Xamarin z wykazem kilku domyślnych plików szablonowych (patrz drugi zrzut ekranu na następnej stronie).

Kup książkę Poleć książkę





Jak widać na zrzucie ekranu, rozwiązanie *TrackMyWalks* zostało podzielone na trzy główne obszary. W poniższej tabeli znajduje się zwięzły opis każdego z nich.

Kup książkę Poleć książkę

Nazwa obszaru	Opis
`TrackMyWalks`	To jest projekt PCL (ang. *Portable Class Library*), który będzie odgrywał rolę głównej warstwy architekturalnej rozwiązania *TrackMyWalks*. Projekt ten zawiera całą logikę biznesową, obiekty danych, `Xamarin.FormsPages`, widoki i inny kod niespecyficzny dla żadnej platformy. Kod źródłowy napisany w tym projekcie może być wykorzystywany w wielu różnych projektach specyficznych dla konkretnych platform.
`TrackMyWalks.Droid`	To jest projekt dla systemu Android, a więc zawiera cały kod źródłowy i wszystkie zasoby potrzebne do skompilowania i wdrożenia aplikacji dla systemu Android należącej do rozwiązania. Domyślnie projekt ten odwołuje się do biblioteki PCL `TrackMyWalks`.
`TrackMyWalks.iOS`	To jest projekt dla systemu iOS, a więc zawiera cały kod źródłowy i wszystkie zasoby potrzebne do skompilowania i wdrożenia aplikacji dla systemu iOS należącej do rozwiązania. Domyślnie projekt ten odwołuje się do biblioteki PCL `TrackMyWalks`.

Zwróć uwagę, że nasze rozwiązanie zawiera plik o nazwie *TrackMyWalks.cs*, który stanowi składnik biblioteki PCL `TrackMyWalks`. W pliku tym znajduje się klasa o nazwie `App`, która dziedziczy z hierarchii klas `Xamarin.Forms.Application`, co widać w poniższym kodzie:

```
//
// TrackMyWalks.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using Xamarin.Forms;

namespace TrackMyWalks
{
    public class App : Application
    {
        public App()
        {
            // sprawdzenie systemu docelowego
            if (Device.OS == TargetPlatform.Android)
            {
                MainPage = new SplashPage();
            }
            else
            {
                // strona główna aplikacji
                var navPage = new NavigationPage(new TrackMyWalks.WalksPage()
                {
                    Title = "Track My Walks"
                });
                MainPage = navPage;
            }
        }
```


Kup książkę Poleć książkę

```
        protected override void OnStart()
        {
            // operacje wykonywane podczas uruchamiania aplikacji
        }

        protected override void OnSleep()
        {
            // operacje wykonywane, gdy aplikacja zostaje uśpiona
        }

        protected override void OnResume()
        {
            // operacje wykonywane w chwili wznowienia działania aplikacji
        }
    }
}
```

Konstruktor `App` przypisuje własności `MainPage` nowy egzemplarz obiektu `ContentPage`, który na razie tylko wyświetla pewien napis utworzony domyślnie przez kreator projektu. W dalszej części tego rozdziału pokazuję, jak zbudować początkowe widoki stron interfejsu użytkownika, oraz demonstruję sposoby modyfikacji własności `MainPage` klasy `App` zawartej w pliku *TrackMyWalks.cs*.

Aktualizowanie pakietów rozwiązania TrackMyWalks

W tej sekcji pokazuję, jak można zaktualizować pakiety `Xamarin.Forms` należące do naszego rozwiązania *TrackMyWalks*. Z pewnością udało Ci się już zauważyć, że każdy projekt w naszym rozwiązaniu zawiera folder *Packages*.

Pakiet `Xamarin.Forms` to w istocie pakiet *NuGet*, który został automatycznie dodany do naszego rozwiązania, gdy stwierdziliśmy, że chcemy utworzyć szablon projektu `Xamarin.FormsApp`.

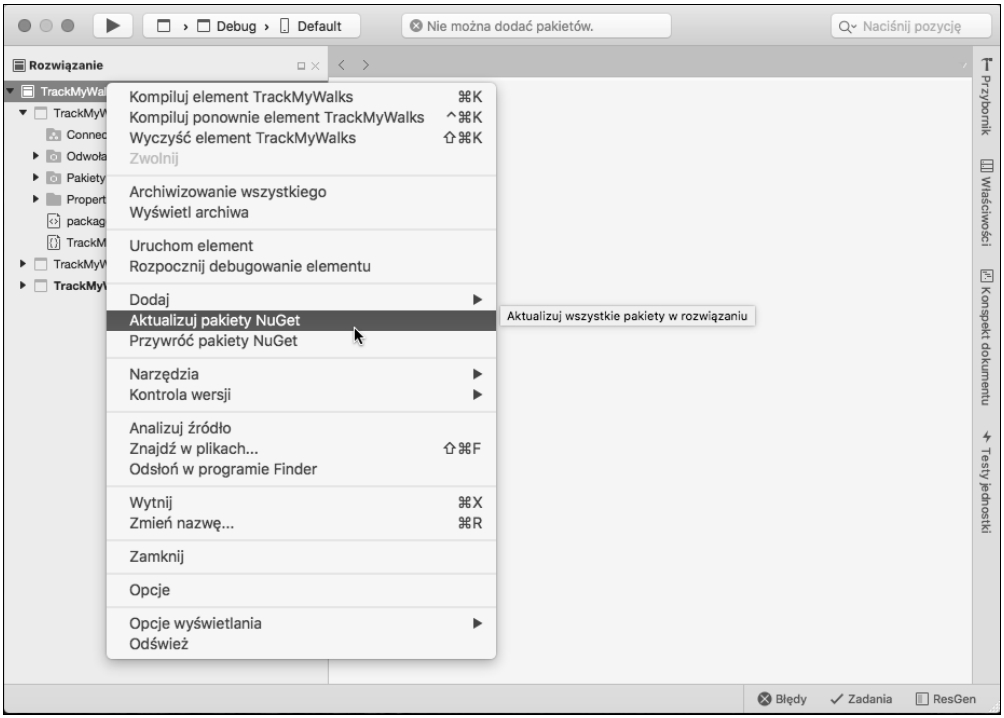
Czasami Xamarin powiadamia, że jakiś pakiet jest przestarzały i trzeba go zaktualizować do najnowszej wersji.

> Pakiet *NuGet* to w istocie menedżer pakietów platformy Microsoft Development Platform zawierający narzędzia klienckie, za pomocą których można tworzyć i eksploatować pakiety .NET.

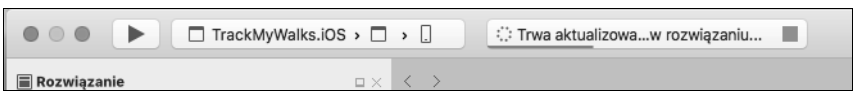
Zobaczmy, jak zaktualizować pakiety *NuGet* w naszym rozwiązaniu *TrackMyWalks*, aby mieć pewność, że posługujemy się najnowszymi pakietami `Xamarin.Forms`. Wykonujemy następujące czynności:

1. Kliknij prawym przyciskiem myszy rozwiązanie *TrackMyWalks* i z menu wybierz polecenie *Aktualizuj pakiety NuGet*, jak pokazano na zrzucie ekranu na następnej stronie.

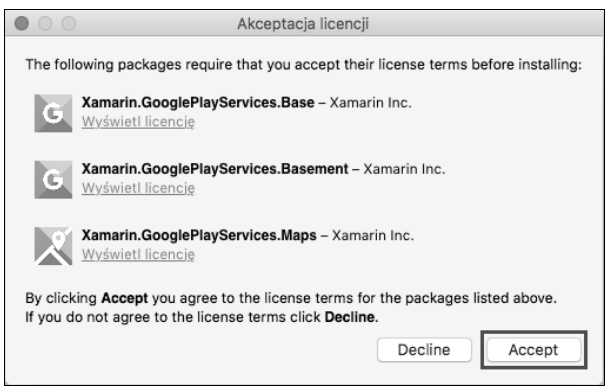
Kup książkę Poleć książkę



Gdy włączysz tę opcję, Xamarin Studio zaktualizuje wszystkie pakiety znajdujące się w rozwiązaniu *TrackMyWalks* w projektach dotyczących wszystkich platform i wyświetli wskaźnik postępu o wyglądzie podobnym do pokazanego poniżej:



Podczas instalowania aktualizacji niektórych pakietów dotyczących konkretnych platform będzie konieczne zaakceptowanie licencji, co widać na poniższym rysunku:



Kup książkę Poleć książkę

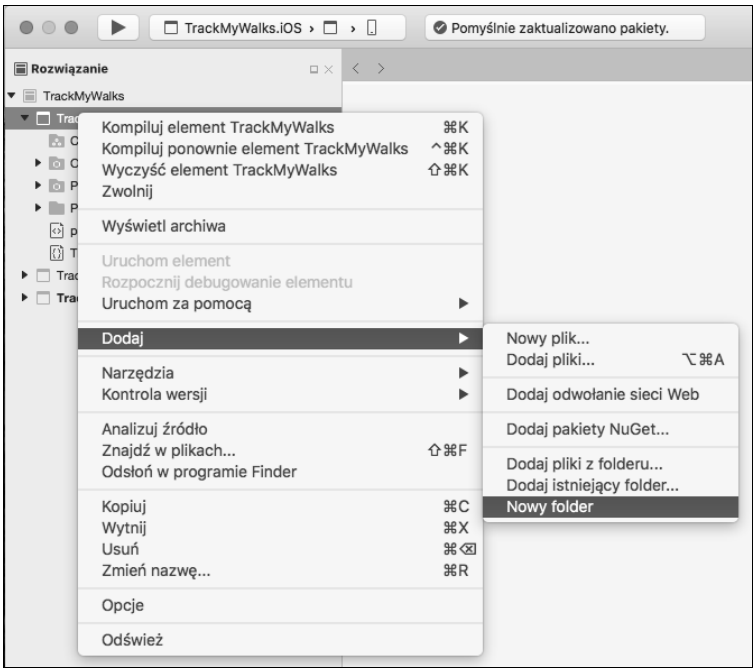
2. Kliknij przycisk *Accept* (akceptuj), aby zaakceptować licencje pakietów wymienionych w oknie dialogowym i je zainstalować (zobacz poprzedni rysunek).

Po zaktualizowaniu pakietów `Xamarin.Forms` w swoim rozwiązaniu możesz przejść do tworzenia plików interfejsu użytkownika w rozwiązaniu *TrackMyWalks*.

Tworzenie modelu TrackMyWalks

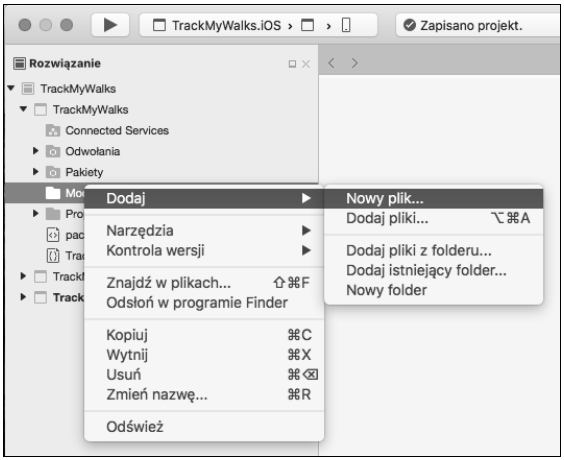
W tym podrozdziale pokażę Ci, jak utworzyć model *TrackMyWalks*, który będzie reprezentował wpisy na temat przebytych przez nas szlaków. Potem dowiesz się, jak przy użyciu tego modelu przygotować i zainicjalizować kilka wpisów dla naszej głównej strony `WalksPage` przy użyciu kontrolki `ListView`, tak aby w każdym wierszu `ListView` można było wyświetlić wpis.
Poniżej przedstawiam listę czynności do wykonania:

1. W projekcie Portable Class Library `TrackMyWalks` utwórz folder o nazwie *Models*, jak pokazano na poniższym zrzucie ekranu:

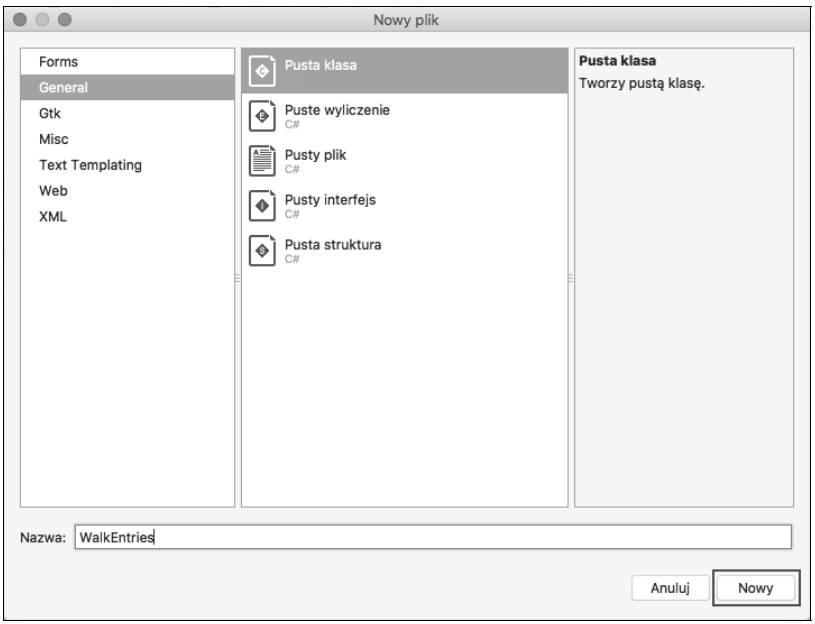


2. Następnie w folderze *Models* utwórz nowy plik klasy zgodnie z tym, co widać na poniższym rysunku:

Kup książkę Poleć książkę



3. W sekcji *General* (ogólne) wybierz opcję *Pusta klasa* i w polu nazwy klasy wpisz nazwę **`WalkEntries`**:



4. Następnie kliknij przycisk *Nowy*, aby pozwolić kreatorowi na utworzenie nowego pliku (zobacz powyższy rysunek).

   Gratulacje! Właśnie został utworzony Twój pierwszy folder i plik klasy C# w rozwiązaniu. Teraz możesz przejść do dodawania deskryptorów własności, które będą definiować nasz model.

Kup książkę Poleć książkę

5. Otwórz plik *WalkEntries.cs*, znajdź w nim konstruktor klasy `WalkEntries` i wpisz kod zaznaczony pogrubieniem na listingu:

```
//
// WalkEntries.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016.
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//

namespace TrackMyWalks.Models
{
    public class WalkEntries
    {
        public string Title { get; set; }
        public string Notes { get; set; }
        public double Longitude { get; set; }
        public double Latitude { get; set; }
        public double Kilometers { get; set; }
        public string Difficulty { get; set; }
        public double Distance { get; set; }
        public string ImageUrl { get; set; }
    }
}
```

W kodzie tym zdefiniowaliśmy model, który będzie nam służył do reprezentowania wpisów dotyczących naszych szlaków. W następnym podrozdziale przy użyciu tego modelu zainicjalizujemy kilka wpisów dla głównej strony `WalksPage` za pomocą kontrolki `ListView`, a następnie przy użyciu obiektu `DataTemplate` opiszemy sposób prezentacji danych tego modelu w każdym wierszu kontrolki `ListView`.

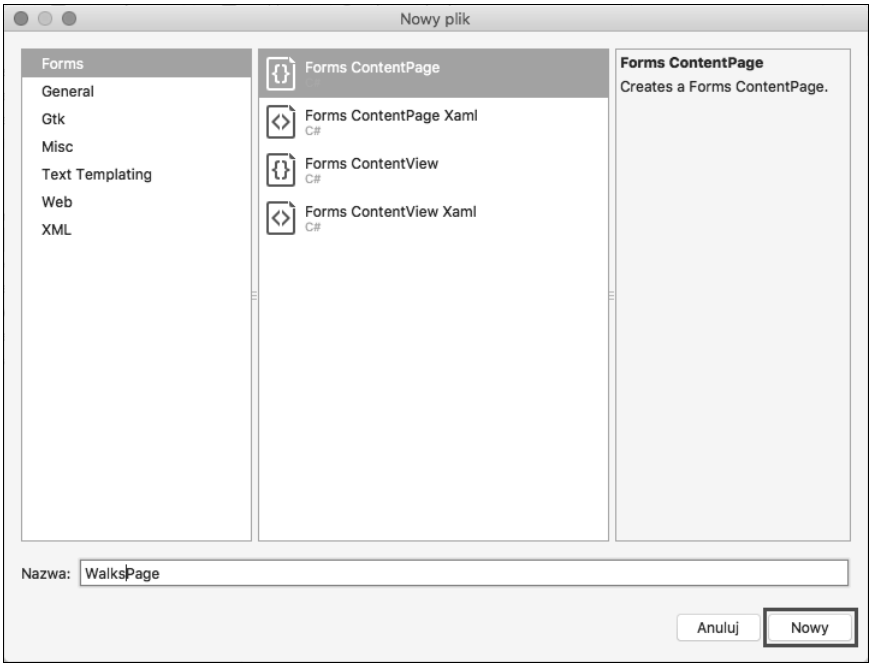
Tworzenie strony głównej prezentacji szlaków

Jak napisałem wcześniej, `WalksPage` będzie nam służyć jako główny punkt wejściowy do naszej aplikacji. Przy użyciu modelu `WalkEntries` przekażemy trochę statycznych danych na temat szlaków i wyświetlimy je w kontrolce `ListView` za pomocą obiektu `DataTemplate`. Oto lista czynności, które należy wykonać:

1. W rozwiązaniu Portable Class Library *TrackMyWalks* utwórz folder o nazwie *Pages* w taki sam sposób jak poprzednio.
2. Na ekranie *Nowy plik* w lewym okienku otwórz sekcję *Forms* (formularze).
3. W prawym okienku zaznacz opcję *Forms ContentPage*.
4. W polu *Nazwa* nowo tworzonej klasy wpisz **`WalksPage`**.

Kup książkę Poleć książkę

5. Na zakończenie kliknij przycisk *Nowy* (zobacz poniższy rysunek).



6. Następnie otwórz plik *WalksPage.cs* w edytorze kodu i wpisz zaznaczone pogrubieniem fragmenty poniższego kodu.

```
//
// WalksPage.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using System.Collections.Generic;
using Xamarin.Forms;
using TrackMyWalks.Models;

namespace TrackMyWalks
{
    public class WalksPage : ContentPage
    {
        public WalksPage()
        {
            var newWalkItem = new ToolbarItem
            {
                Text = "Dodaj szlak"
            };
            newWalkItem.Clicked += (sender, e) =>
```


Kup książkę Poleć książkę

```
{
    Navigation.PushAsync(new WalkEntryPage());
};

ToolbarItems.Add(newWalkItem);

var walkItems = new List<WalkEntries>
{
    new WalkEntries {
        Title  = "10-milowy szlak wzdłuż strumienia, Margaret River",
        Notes  = "10-milowy szlak wzdłuż strumienia zaczyna się
        ↪w Rotary Park w pobliżu Old Kate, czyli starej lokomotywy
        ↪stojącej w północnej części Margaret River.",
        Latitude    = -33.9727604,
        Longitude   = 115.0861599,
        Kilometers  = 7.5,
        Distance    = 0,
        Difficulty  = "Średni",
        ImageUrl    = "http://trailswa.com.au/media/cache/media/images/
        ↪trails/_mid/FullSizeRender1_600_480_c1.jpg"
    },
    new WalkEntries {
        Title  = "Szlak Ancient Empire, Dolina Gigantów",
        Notes  = "Ancient Empire to 450-metrowy szlak pośród
        ↪gigantycznych drzew, wśród których znajdują się popularne
        ↪sękate olbrzymy zwane Grandma Tingle.",
        Latitude  = -34.9749188,
        Longitude   = 117.3560796,
        Kilometers = 450,
        Distance   = 0,
        Difficulty = "Wysoki",
        ImageUrl   = "http://trailswa.com.au/media/cache/media/images/
        ↪trails/_mid/Ancient_Empire_534_480_c1.jpg"
    },
};

var itemTemplate = new DataTemplate(typeof(ImageCell));
itemTemplate.SetBinding(TextCell.TextProperty, "Title");
itemTemplate.SetBinding(TextCell.DetailProperty, "Notes");
itemTemplate.SetBinding(ImageCell.ImageSourceProperty, "ImageUrl");

var walksList = new ListView
{
    HasUnevenRows = true,
    ItemTemplate = itemTemplate,
    ItemsSource = walkItems,
    SeparatorColor = Color.FromHex("#ddd"),
};
```


Kup książkę Poleć książkę

```
                // konfiguracja procedury obsługi zdarzeń
                walksList.ItemTapped += (object sender, ItemTappedEventArgs e) =>
                {
                    var item = (WalkEntries)e.Item;
                    if (item == null) return;
                    Navigation.PushAsync(new WalkTrailPage(item));
                    item = null;
                };

                Content = walksList;
            }
        }
    }
```

Na początku zadeklarowaliśmy zmienną `newWalkItem` typu klasy `ToolbarItem`. Za pomocą tej zmiennej wiążemy nowy przycisk *Dodaj szlak* z głównym paskiem narzędzi podstawowej kolekcji `ContentPage.ToolbarItems`, aby umożliwić użytkownikom aplikacji dodawanie informacji o nowych szlakach.

Następnie tworzymy zdarzenie dla obiektu `newWalkItem`, wykorzystując zdarzenie `Clicked` klasy `ToolbarItem`, za pomocą którego będzie można przejść do nowej strony `WalksEntryPage`.

Kolejną naszą czynnością jest zadeklarowanie nowej zmiennej o nazwie `walkItems`, która jest kolekcją elementów listy do przechowywania wszystkich naszych wpisów w modelu. Za pomocą klasy `DataTemplate` opisujemy żądany sposób prezentowania danych modelu w każdym z wierszy zadeklarowanych w `ListView`.

Na koniec definiujemy procedurę obsługi zdarzeń dla naszego widoku `ListView`, która pozwoli nam przejść do strony `WalksTrailPage` w celu wyświetlenia informacji o wybranym elemencie.

Tworzenie strony treści nowego wpisu

W tym podrozdziale zbudujemy interfejs użytkownika na naszej stronie `WalkEntryPage`, służącej do dodawania informacji o nowym szlaku do aplikacji. Jej wywołanie nastąpi, gdy użytkownik naciśnie przycisk *Dodaj szlak* na stronie głównej.

Stronę do wprowadzania nowych danych można przedstawić na wiele sposobów. W naszej aplikacji skorzystamy z widoku `TableView`, ale równie dobrze moglibyśmy posłużyć się widokiem `StackLayout` i zaprezentować informacje w postaci serii obiektów `Label` i `EntryCell`.

Aby utworzyć nową stronę `WalkEntryPage`, wykonaj następujące czynności:

1. Utwórz nową stronę `ContentPage` o nazwie `WalkEntryPage` w sposób opisany w poprzednim podrozdziale.

Kup książkę Poleć książkę

2. Otwórz w edytorze kodu plik *WalkEntryPage.cs* i wpisz fragmenty kodu zaznaczone poniżej pogrubieniem:

```
//
// WalkEntryPage.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using Xamarin.Forms;
using TrackMyWalks.Models;
using System.Collections.Generic;

namespace TrackMyWalks
{
    public class WalkEntryPage : ContentPage
    {
        public WalkEntryPage()
        {
            // tytuł strony
            Title = "Nowy wpis";

            // pola do wypełnienia
            var walkTitle = new EntryCell
            {
                Label = "Tytuł:",
                Placeholder = "Nazwa szlaku"
            };

            var walkNotes = new EntryCell
            {
                Label = "Uwagi:",
                Placeholder = "Opis"
            };

            var walkLatitude = new EntryCell
            {
                Label = "Szerokość geograficzna:",
                Placeholder = "Szerokość",
                Keyboard = Keyboard.Numeric
            };

            var walkLongitude = new EntryCell
            {
                Label = "Długość geograficzna:",
                Placeholder = "Długość",
                Keyboard = Keyboard.Numeric
            };

            var walkKilometers = new EntryCell
            {
```


Kup książkę Poleć książkę

```
                Label = "Liczba kilometrów:",
                Placeholder = "Liczba kilometrów",
                Keyboard = Keyboard.Numeric
            };

            var walkDifficulty = new EntryCell
            {
                Label = "Poziom trudności:",
                Placeholder = "Poziom trudności szlaku"
            };

            var walkImageUrl = new EntryCell
            {
                Label = "URL obrazu:",
                Placeholder = "URL obrazu"
            };

            // definicja widoku TableView
            Content = new TableView
            {
                Intent = TableIntent.Form,
                Root = new TableRoot
                {
                    new TableSection()
                    {
                        walkTitle,
                        walkNotes,
                        walkLatitude,
                        walkLongitude,
                        walkKilometers,
                        walkDifficulty,
                        walkImageUrl
                    }
                }
            };

            var saveWalkItem = new ToolbarItem
            {
                Text = "Zapisz"
            };
            saveWalkItem.Clicked += (sender, e) =>
            {
                Navigation.PopToRootAsync(true);
            };

            ToolbarItems.Add(saveWalkItem);
        }
    }
}
```


Kup książkę Poleć książkę

Najpierw zadeklarowaliśmy etykiety `EntryCell` dla naszego interfejsu użytkownika, za pomocą którego będziemy przyjmować wprowadzane informacje — *Tytuł*, *Uwagi*, *Szerokość geograficzna*, *Długość geograficzna*, *Liczba kilometrów*, *Poziom trudności* oraz *URL obrazu*. Potem pokażę Ci, jak modyfikować wygląd i styl komórek `EntryCell` przez utworzenie specyficznego dla danej platformy elementu do wybierania ustawień dotyczących poziomu trudności i liczby kilometrów.

Następnie zdefiniowaliśmy widok `TableView` i dodaliśmy każde z naszych pól `EntryCell` do własności `TableSection` kontrolki `TableView`. Każda własność `TableSection` zdefiniowana w `TableView` zawiera nagłówek i jedną lub więcej komórek `ViewCell`, które w naszym przypadku są polami `EntryCell`.

Na koniec zadeklarowaliśmy i dodaliśmy do kolekcji `ContentPageToolbarItems` obiekt `ToolbarItem` o nazwie `saveWalkItem`, a następnie utworzyliśmy zdarzenie, które w reakcji na kliknięcie powoduje zapisanie wprowadzonych informacji dotyczących szlaku na stronie głównej. Oczywiście później jeszcze wielokrotnie będziemy zmieniać kod strony `WalkEntryPage`, która w reakcji na naciśnięcie przycisku *Zapisz* będzie wysyłała informacje do serwera za pośrednictwem REST-owego interfejsu API, a następnie będzie odświeżała stronę główną *TrackMyWalks*.

Strona opisu szlaku

W tym podrozdziale rozpoczniemy budowę interfejsu użytkownika dla naszej strony `WalksTrailPage`. Strona ta będzie wywoływana kliknięciem wpisu w widoku `ListView` na głównej stronie treści *TrackMyWalks* i będzie prezentowała informacje dotyczące wybranego szlaku.

1. Utwórz stronę `ContentPage` o nazwie `WalkTrailPage` w sposób opisany w jednym z wcześniejszych podrozdziałów.
2. Otwórz w edytorze kodu plik *WalkTrailPage.cs* i wpisz fragmenty kodu zaznaczone poniżej pogrubieniem:

```
//
// WalkTrailPage.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016.
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using Xamarin.Forms;
using TrackMyWalks.Models;

namespace TrackMyWalks
{
    public class WalkTrailPage : ContentPage
    {
        public WalkTrailPage(WalkEntries walkItem)
        {
            Title = "Szlak";
```


Kup książkę Poleć książkę

```
var beginTrailWalk = new Button
{
    BackgroundColor = Color.FromHex("#008080"),
    TextColor = Color.White,
    Text = "Rozpocznij ten szlak"
};

// definicja procedury obsługi zdarzeń
beginTrailWalk.Clicked += (sender, e) =>
{
    if (walkItem == null) return;
    Navigation.PushAsync(new DistanceTravelledPage(walkItem));
    Navigation.RemovePage(this);
    walkItem = null;
};

var walkTrailImage = new Image()
{
    Aspect = Aspect.AspectFill,
    Source = walkItem.ImageUrl
};

var trailNameLabel = new Label()
{
    FontSize = 28,
    FontAttributes = FontAttributes.Bold,
    TextColor = Color.Black,
    Text = walkItem.Title
};

var trailKilometersLabel = new Label()
{
    FontAttributes = FontAttributes.Bold,
    FontSize = 12,
    TextColor = Color.Black,
    Text = $"Długość: { walkItem.Kilometers } km"
};

var trailDifficultyLabel = new Label()
{
    FontAttributes = FontAttributes.Bold,
    FontSize = 12,
    TextColor = Color.Black,
    Text = $"Poziom trudności: { walkItem.Difficulty } "
};

var trailFullDescription = new Label()
{
    FontSize = 11,
    TextColor = Color.Black,
    Text = $"{ walkItem.Notes }",
    HorizontalOptions = LayoutOptions.FillAndExpand
};
```


Kup książkę Poleć książkę

```
            this.Content = new ScrollView
            {
                Padding = 10,
                Content = new StackLayout
                {
                    Orientation = StackOrientation.Vertical,
                    HorizontalOptions = LayoutOptions.FillAndExpand,
                    Children =
                    {
                    walkTrailImage,
                    trailNameLabel,
                    trailKilometersLabel,
                    trailDifficultyLabel,
                    trailFullDescription,
                    beginTrailWalk
                    }
                }
            };
        }
    }
}
```

Najpierw importujemy klasę `TrackMyWalks.Models`, przy użyciu której będziemy pobierać informacje przekazywane ze strony `WalksPage`.

Następnie deklarujemy zmienną `beginTrailWalk` dziedziczącą po klasie `Button`. Potem definiujemy zdarzenie `Clicked` klasy `Button`, które pozwoli nam przejść do strony treści `DistanceTravelledPage` w celu wyświetlenia informacji o naszym szlaku po usunięciu strony treści szlaku z hierarchii `NavigationPage`.

W następnym kroku deklarujemy zmienną obrazu `walkTrailImage` i ustawiamy jego własność źródła `Source` na obraz wybranego elementu `walkItem` z `ListView`. W dalszej kolejności deklarujemy i inicjalizujemy kilka obiektów etykiet, które będą zawierać informacje `walkItem` przekazane z kontrolki `ListView` strony treści `WalksPage` i które zostaną wyświetlone.

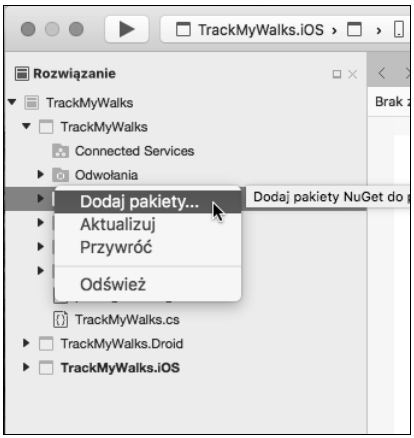
Potem definiujemy kontrolkę `ScrollView`, która jest częścią bazowej klasy `Xamarin.Forms.Core` i dodajemy wszystkie nasze pola `Image` i `Label` formularza do kontrolki `StackLayout`. Kontrolka `ScrollView` to fantastyczny element umożliwiający przewijanie treści strony `ContentPage`, jeśli ta nie mieści się na pojedynczym ekranie urządzenia.

Dodawanie pakietu NuGet Xamarin.Forms.Maps

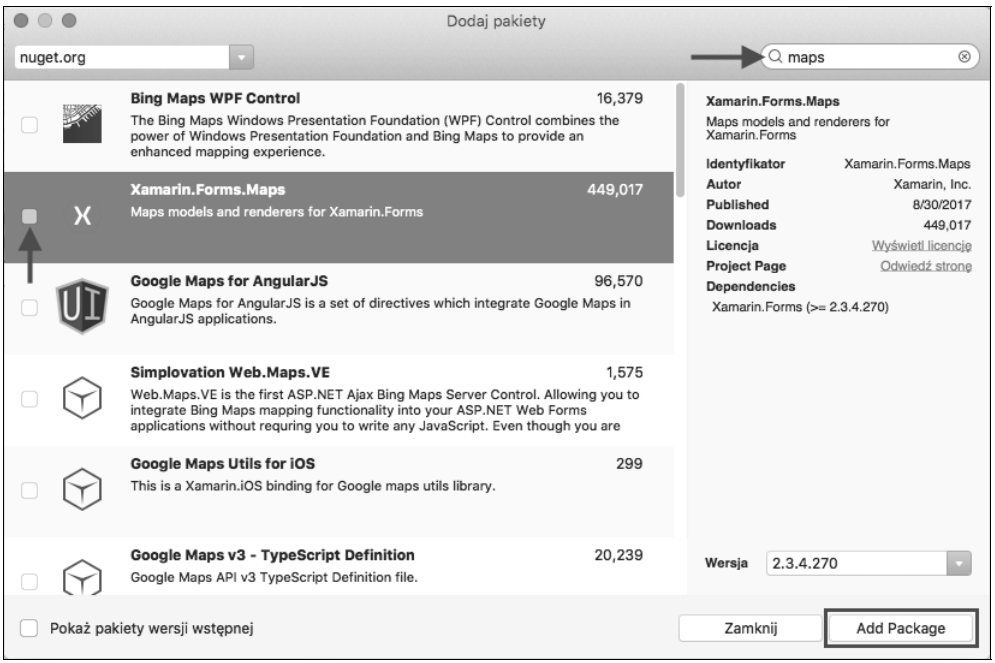
W tej sekcji dodamy pakiet NuGet `Xamarin.Forms.Maps` do naszego projektu głównego i projektów specyficznych dla platform iOS i Android. Jest on potrzebny do tego, abyśmy mogli korzystać z kontrolki `Xamarin.FormsMap` na stronie treści `DistanceTravelled`, którą utworzymy w następnym podrozdziale.

Kup książkę Poleć książkę

1. Kliknij prawym przyciskiem myszy rozwiązanie *TrackMyWalks* i w menu kliknij opcję *Dodaj pakiety*, co zilustrowano na poniższym zrzucie ekranu:



2. Zostanie wyświetlone okno dialogowe *Dodaj pakiety*. W polu wyszukiwania wpisz słowo **`maps`**, a następnie na liście, która się pojawi, kliknij opcję *Xamarin.Forms.Maps*:



3. Na koniec kliknij przycisk *Add Package* (dodaj pakiet), aby dodać pakiet NuGet `Xamarin.Forms.Maps` do rdzenia rozwiązania *TrackMyWalks*.

Kup książkę Poleć książkę

4. Powtórz te same czynności, aby dodać pakiet NuGet `Xamarin.Forms.Maps` do projektów iOS i Android znajdujących się w rozwiązaniu *TrackMyWalks*.

Mając pakiet NuGet `Xamarin.Forms` z kontrolką `Map`, możemy zacząć jej używać na stronie treści `DistanceTravelled`, którą opisuję w następnym podrozdziale.

Tworzenie strony treści DistanceTravelledPage

W tym podrozdziale rozpoczynamy prace nad interfejsem użytkownika strony `DistanceTravel ↪ledPage`. Strona ta będzie wywoływana, gdy użytkownik kliknie przycisk *Rozpocznij ten szlak* na stronie treści `WalksTrailPage`, która będzie służyła do prezentowania informacji na temat wybranego szlaku, jak również wbijania szpilki na kontrolce `Xamarin.Forms.Maps` oraz obliczania przebytego dystansu i czasu podróży:

1. Utwórz nową stronę `ContentPage` o nazwie `DistanceTravelledPage` w sposób opisany w poprzednim podrozdziale.
2. Otwórz plik *DistanceTravelledPage.cs* w edytorze kodu i wpisz pogrubione poniżej fragmenty kodu:

```
//
// DistanceTravelledPage.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using Xamarin.Forms;
using Xamarin.Forms.Maps;
using TrackMyWalks.Models;

namespace TrackMyWalks
{
    public class DistanceTravelledPage : ContentPage
    {
```

3. Następnie do konstruktora `DistanceTravelledPage` dodaj parametr `walkItem` określający wybrany szlak, co zaznaczono poniżej pogrubieniem:

```
public DistanceTravelledPage(WalkEntries walkItem)
{
    Title = "Przebyty dystans";
```

4. Następnie zadeklaruj zmienną `trailMap`, która będzie wskazywała egzemplarz kontrolki `Xamarin.Forms.Maps` do utworzenia szpilki na mapie. Posługując się współrzędnymi geograficznymi, wpisz w pliku poniższe pogrubione fragmenty kodu:

Kup książkę Poleć książkę

```
// utworzenie egzemplarza obiektu mapy
var trailMap = new Map();
// wbicie szpilki w mapę dla wybranego szlaku
trailMap.Pins.Add(new Pin
{
  Type = PinType.Place,
  Label = walkItem.Title,
  Position = new Position(walkItem.Latitude, walkItem.Longitude)
});
// wyśrodkowanie mapy na początku szlaku
trailMap.MoveToRegion(MapSpan.FromCenterAndRadius(new
  Position(walkItem.Latitude, walkItem.Longitude),
  Distance.FromKilometers(1.0)));
```

5. Następnie zadeklaruj kilka obiektów `Label` do prezentowania informacji z obiektu `walkItem` przekazanego ze strony treści `WalkTrailPage`. Wpisz poniższe pogrubione fragmenty kodu:

```
var trailNameLabel = new Label()
{
    FontSize = 18,
    FontAttributes = FontAttributes.Bold,
    TextColor = Color.Black,
    Text = walkItem.Title
};
var trailDistanceTravelledLabel = new Label()
{
    FontAttributes = FontAttributes.Bold,
    FontSize = 20,
    TextColor = Color.Black,
    Text = "Przebyty dystans:",
    HorizontalTextAlignment = TextAlignment.Center
};
var totalDistanceTaken = new Label()
{
    FontAttributes = FontAttributes.Bold,
    FontSize = 20,
    TextColor = Color.Black,
    Text = $"{ walkItem.Distance } km",
    HorizontalTextAlignment = TextAlignment.Center
};
var totalTimeTakenLabel = new Label()
{
    FontAttributes = FontAttributes.Bold,
    FontSize = 20,
    TextColor = Color.Black,
    Text = "Czas:",
    HorizontalTextAlignment = TextAlignment.Center
};
var totalTimeTaken = new Label()
{
    FontAttributes = FontAttributes.Bold,
```


Kup książkę Poleć książkę

```
        FontSize = 20,
        TextColor = Color.Black,
        Text = "0 h 0 m 0 s",
        HorizontalTextAlignment = TextAlignment.Center
    };
```

6. Następnie zadeklaruj zmienną `walksHomeButton`, która dziedziczy po klasie `Button`, i przejdź do definicji procedury obsługi kliknięć, która w przypadku wykrycia kliknięcia będzie przenosić użytkownika do głównej strony treści `WalksPage`. Wpisz poniższe fragmenty kodu:

```
var walksHomeButton = new Button
{
    BackgroundColor = Color.FromHex("#008080"),
    TextColor = Color.White,
    Text = "Zakończ ten szlak"
};
 // definicja procedury obsługi zdarzeń
 walksHomeButton.Clicked += (sender, e) =>
 {
    if (walkItem == null) return;
    Navigation.PopToRootAsync(true);
    walkItem = null;
};
```

7. Następnie zdefiniuj kontrolkę `ScrollView` wchodzącą w skład klasy bazowej `Xamarin.Forms.Core` oraz dodaj wszystkie pola formularza `Button` i `Label` do kontrolki `StackLayout`. Wpisz poniższy fragment kodu:

```
 this.Content = new ScrollView
{
    Padding = 10,
    Content = new StackLayout
    {
        Orientation = StackOrientation.Vertical,
        HorizontalOptions = LayoutOptions.FillAndExpand,
        Children = {
            trailMap,
            trailNameLabel,
            trailDistanceTravelledLabel,
            totalDistanceTaken,
            totalTimeTakenLabel,
            totalTimeTaken,
            walksHomeButton
        }
    }
   };
  }
 }
}
```


Kup książkę Poleć książkę

Pracę zaczęliśmy od zaimportowania klasy `TrackMyWalks.Models`, ponieważ za jej pomocą będziemy pobierać informacje przekazywane ze strony `WalksPage`. Pakiet NuGet `Xamarin.Forms.Maps` jest wieloplatformową biblioteką, za pomocą której programista może wyświetlać i oznaczać informacje na mapie. My użyjemy tej kontrolki do wyświetlenia szpilki na mapie wraz z dodatkowymi informacjami dotyczącymi danego szlaku.

Następnie deklarujemy zmienną `trailMap` wskazującą egzemplarz kontrolki `Xamarin.Forms.Maps` i tworzymy znacznik w postaci szpilki na mapie zawierającej dane dotyczące wybranego szlaku, po czym centrujemy go na mapie, aby ukazać cały obszar objęty przez szlak na podstawie współrzędnych szerokości i długości geograficznej. Potem deklarujemy i inicjalizujemy kilka obiektów `Label` zawierających informacje `walkItem`, które zostały przekazane ze strony treści `WalkTrailPage`, po czym deklarujemy zmienną `walksHomeButton`, która dziedziczy po klasie `Button`, i konfigurujemy zdarzenie `Clicked` dla klasy `Button`, przy użyciu którego będziemy wracać do *TrackMyWalks*.

Na koniec definiujemy kontrolkę `ScrollView`, która wchodzi w skład klasy bazowej `Xamarin.Forms.Core`, i dodajemy wszystkie pola formularza `Button` i `Label` do kontrolki `StackLayout`.

W następnej kolejności musimy zainicjalizować pakiet NuGet `Xamarin.Forms.Maps` w klasach startowych dotyczących każdej z naszych konkretnych platform (np. *AppDelegate.cs* w przypadku systemu iOS i *MainActivity.cs* w przypadku systemu Android). Oto opis, jak to zrobić krok po kroku.

1. Otwórz plik *AppDelegate.cs* w edytorze kodu i dodaj do niego fragment kodu zaznaczony pogrubieniem:

```
//
// AppDelegate.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using Foundation;
using UIKit;

namespace TrackMyWalks.iOS
{
    [Register("AppDelegate")]
    public partial class AppDelegate :
global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
    {
        public override bool FinishedLaunching(UIApplication app,
        ↪NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();
            Xamarin.FormsMaps.Init();
            LoadApplication(new App());
            return base.FinishedLaunching(app, options);
        }
    }
}
```


Kup książkę Poleć książkę

Najpierw zaznaczyliśmy, że nasza klasa `AppDelegate` ma korzystać z biblioteki `Xamarin.Forms.Maps`, dodając wywołanie metody `Xamarin.FormsMaps.Init`, która inicjalizuje platformę `Xamarin.Forms`, dzięki czemu nasze rozwiązanie *TrackMyWalks* może wykorzystywać mapy. Gdyby zabrakło tego wywołania w tym miejscu, na stronie treści `DistanceTravelledPage` nie pojawiałaby się mapa i strona ta nie działałaby zgodnie z oczekiwaniami.

> Więcej informacji na temat biblioteki `Xamarin.Forms.Maps` i różnych rodzajów dostępnych klas można znaleźć w dokumentacji Xamarin pod adresem *https://developer.xamarin.com/api/namespace/Xamarin.Forms.Maps/*.

Tworzenie strony ekranu startowego

W tym podrozdziale rozpoczniemy budowę interfejsu użytkownika dla naszej strony startowej, która będzie wykorzystywana tylko na platformie Android, ponieważ w systemie iOS jest dostępna odpowiednia metoda. Na razie nasza strona startowa (ang. *splash page*) będzie zawierała jedynie domyślną ikonę Xamarin, ale później zmienimy ją na coś bardziej odpowiedniego dla naszej aplikacji.

1. Utwórz nową stronę `ContentPage` o nazwie `SplashPage` w sposób opisany w poprzednim podrozdziale.
2. Otwórz plik *SplashPage.cs* w edytorze kodu i wpisz fragment kodu zaznaczony pogrubieniem:

```
//
// SplashPage.cs
// TrackMyWalks
//
// Autor: Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using System;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace TrackMyWalks
{
    public class SplashPage : ContentPage
    {
Następnie znajdź konstruktor SplashPage i wpisz poniższy pogrubiony kod:
public SplashPage()
{
  AbsoluteLayout splashLayout = new AbsoluteLayout
 {
  HeightRequest = 600
 };
  var image = new Image()
{
```


Kup książkę Poleć książkę

```
    Source = ImageSource.FromFile("icon.png"),
    Aspect = Aspect.AspectFill,
};

AbsoluteLayout.SetLayoutFlags(image, AbsoluteLayoutFlags.All);
AbsoluteLayout.SetLayoutBounds(image, new Rectangle(0f, 0f,
    1f, 1f));

splashLayout.Children.Add(image);

Content = new StackLayout()
{
    Children = { splashLayout }
};
}
```

3. W dalszej kolejności znajdź metodę `OnAppearing` i wpisz kod:

```
protected override async void OnAppearing()
{
    base.OnAppearing();

 // kilkusekundowe opóźnienie wyłączenia ekranu startowego
 await Task.Delay(3000);

 // utworzenie egzemplarza NavigationPage i przypisanie go do MainPage
 var navPage = new NavigationPage(new WalksPage()
  {
    Title = "Track My Walks"
  });
    Application.Current.MainPage = navPage;
    }
  }
}
```

Najpierw importujemy klasę `System.Threading.Tasks`, za pomocą której będziemy mogli zdefiniować krótkie opóźnienie pozwalające użytkownikom przez chwilę oglądać ekran startowy zgodnie z definicją w metodzie klasowej `OnAppearing`.

Następnie deklarujemy zmienną `splashLayout` typu `AbsoluteLayout`, przy użyciu której proporcjonalnie ustawiamy położenie i rozmiar każdego elementu potomnego w naszym widoku, a potem ustawiamy własność `HeightRequest`.

Później deklarujemy zmienną `image` dziedziczącą po klasie `Image`, własności `Source` przypisujemy obraz, którego chcielibyśmy użyć, a własność `Aspect` ustawiamy tak, aby obraz ten wypełniał cały widok.

Na zakończenie przypisujemy wartości własnościom `LayoutFlags` i `LayoutBounds` w klasie `AbsoluteLayout` w taki sposób, aby obraz zmieniał rozmiar wraz z widokiem. Następnie dodajemy `splashLayout` do strony treści za pomocą kontrolki `StackLayout`. Przesłaniamy też metodę

Kup książkę Poleć książkę

OnAppearing cyklu życia strony Xamarin.Forms.Page, która to metoda jest wywoływana bezpośrednio przed pojawieniem się strony na ekranie, i definiujemy opóźnienie długości trzech sekund przed utworzeniem nowego egzemplarza strony NavigationPage, która wyznaczy naszą stronę WalksPage jako główną stronę treści.

Modyfikacja klasy App Xamarin.Forms

W tej sekcji zainicjalizujemy klasę App z biblioteki Xamarin.Forms, aby wywoływała naszą stronę SplashPage i ustawiała stronę główną aplikacji, jeśli wykryje urządzenie z systemem operacyjnym Android.

Oto lista czynności do wykonania:

1. Otwórz plik *TrackMyWalks.cs* znajdujący się w grupie *TrackMyWalks* w edytorze kodu.
2. Znajdź metodę App i wpisz poniższy pogrubiony fragment kodu, jak pokazano na listingu:

```
//
// TrackMyWalks.cs
// TrackMyWalks
//
// Autor Steven F. Daniel, 04.08.2016
// Copyright © 2016 GENIESOFT STUDIOS. All rights reserved.
//
using Xamarin.Forms;

namespace TrackMyWalks
{
  public class App : Application
  {
    public App()
    {
      // sprawdzenie platformy
      if (Device.OS == TargetPlatform.Android)
      {
        MainPage = new SplashPage();
      }
      else
      {
         // strona główna aplikacji
        var navPage = new NavigationPage(new TrackMyWalks.WalksPage()
        {
          Title = "Track My Walks"
        });
        MainPage = navPage;
      }
    }
  }
}
```


Kup książkę Poleć książkę

W kodzie tym najpierw za pomocą klasy `Device.OS` sprawdziliśmy, na czym działa nasza aplikacja `Xamarin.Forms`, następnie przy użyciu klasy `TargetPlatform` sprawdziliśmy, czy działa ona w systemie operacyjnym Google Android. Jeśli tak, to własności `MainPage` konstruktora `App` przypisujemy nowy egzemplarz strony `ContentPage`, który będzie wykorzystywał stronę `SplashPage` jako stronę główną. Jeśli wynik testu jest negatywny, tworzymy nowy egzemplarz klasy `NavigationPage`, który ustawiamy na naszą stronę `WalksPage`, po czym ustawiamy go jako główną stronę treści.

Różnice między Xamarin Studio i Visual Studio

Jeżeli ktoś chce tworzyć aplikacje mobilne, to obecnie ma do wyboru dwa środowiska programistyczne: Xamarin Studio i Microsoft Visual Studio. Podkreślę, że choć wszystkie zrzuty ekranu i przykłady kodu pokazywane w tej książce zostały utworzone w Xamarin Studio na komputerze Apple Macintosh, to kod powinien bez problemu dać się skompilować także w systemie Windows w środowisku Microsoft Visual Studio 2015.

Zanim jednak na poważnie zagłębisz się w tajniki tworzenia aplikacji mobilnych, musisz wiedzieć, że te dwa środowiska dzielą istotne różnice. Jeśli korzystasz z Xamarin Studio w systemie Windows, to przy tworzeniu nowego rozwiązania `Xamarin.Forms` zawsze automatycznie otrzymasz rozwiązanie Android.

Jeśli chcesz integrować i tworzyć aplikacje dla systemu Windows Phone, musisz posługiwać się środowiskiem Microsoft Visual Studio w systemie Windows. Aby tworzyć aplikacje dla systemu iOS, musisz przygotować Maca do roli hosta kompilacji Xamarin, włączając *zdalne logowanie* w preferencjach systemowych i wybierając Maca jako host kompilacji w środowisku Microsoft Visual Studio na komputerze z systemem Windows.

> Więcej informacji na temat przygotowania Maca do roli hosta kompilacji Xamarin znajduje się w dokumentacji środowiska Xamarin na stronie *https://developer.xamarin.com/guides/ios/getting_started/installation/windows/connecting-to-mac/*.

Znając podstawowe różnice między Xamarin Studio i Microsoft Visual Studio, możemy przejść do kompilacji, budowy i uruchomienia naszej aplikacji *TrackMyWalks* w symulatorze systemu iOS.

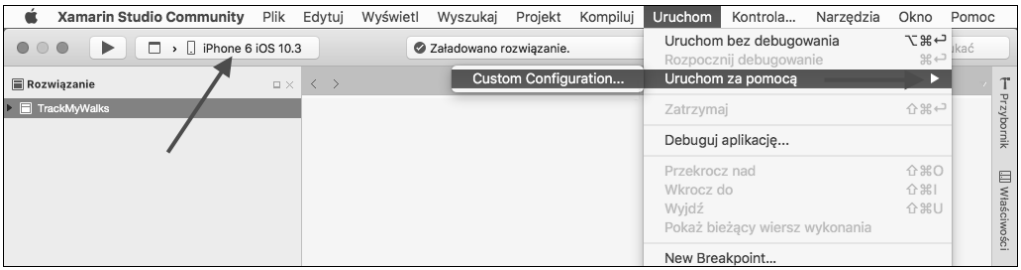
Kup książkę Poleć książkę

Uruchamianie aplikacji TrackMyWalks w symulatorze

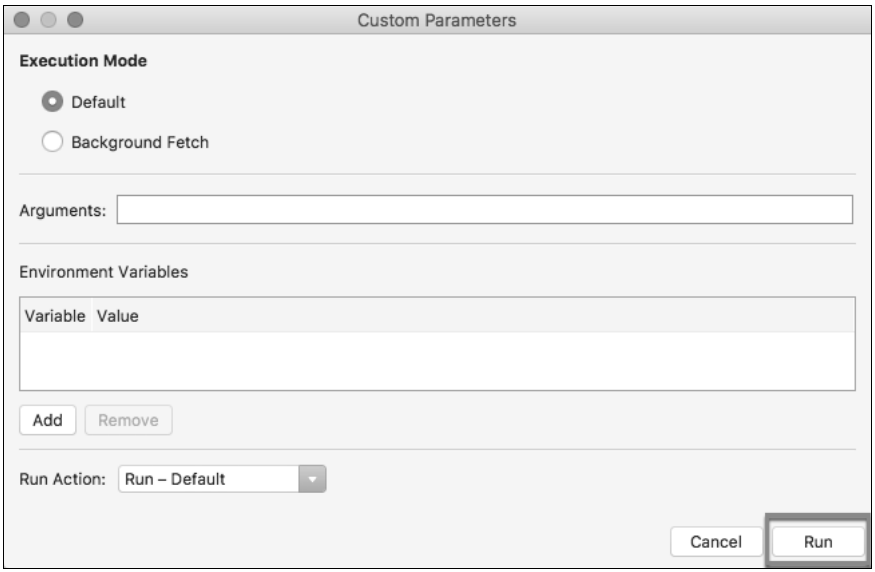
W tym podrozdziale dowiesz się, jak skompilować i uruchomić aplikację *TrackMyWalks*. Możesz ją uruchomić w prawdziwym urządzeniu lub wybrać jeden z kilku dostępnych symulatorów urządzeń z systemem iOS.

Numer wersji symulatora zależy od wersji pakietu SDK systemu iOS zainstalowanego w komputerze. Wykonaj następujące czynności:

1. Aby uruchomić aplikację, wybierz urządzenie z listy dostępnych symulatorów systemu iOS i kliknij opcję menu *Uruchom*.



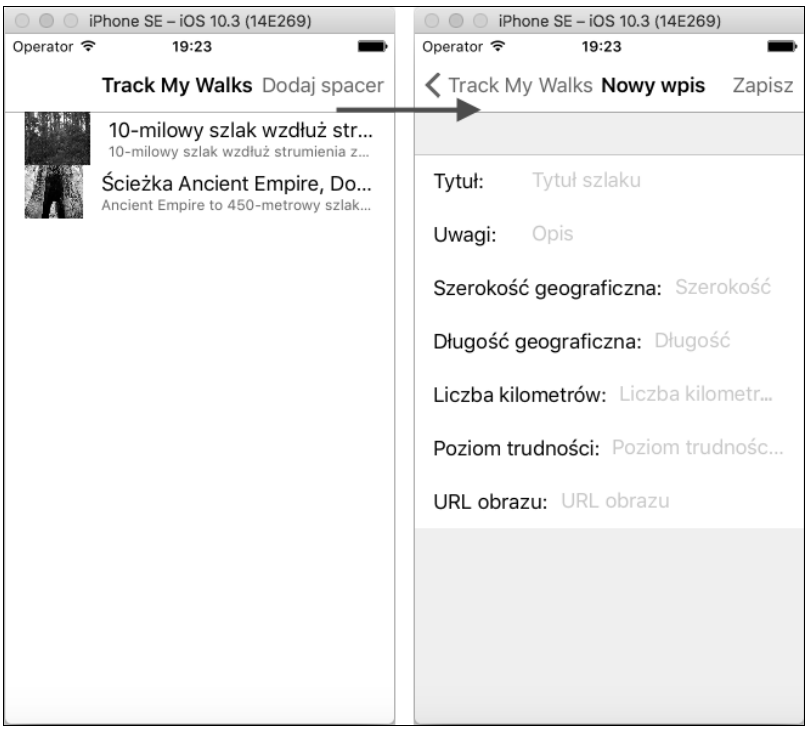
2. Następnie otwórz podmenu *Uruchom za pomocą*, kliknij opcję *Custom Configuration…* (konfiguracja użytkownika), po czym kliknij przycisk *Run* (uruchom) w oknie dialogowym *Custom Parameters* (parametry użytkownika).



Kup książkę Poleć książkę

3. Ewentualnie możesz skompilować i uruchomić aplikację *TrackMyWalks* naciśnięciem kombinacji klawiszy *Command+Return*.

Po zakończeniu kompilacji pojawi się okno symulatora systemu iOS z uruchomioną aplikacją *TrackMyWalks*, co przedstawiono na poniższym zrzucie ekranu:



Jak widać na powyższym rysunku, obecnie aplikacja wyświetla listę statycznych wpisów na temat szlaków, które to wpisy są prezentowane w widoku `ListView`. Gdy użytkownik kliknie przycisk *Dodaj szlak*, zostanie wyświetlona strona treści `WalkEntry`, na której można wprowadzić dane dotyczące kolejnego szlaku.

Obecnie strona ta jeszcze nie zapisuje wprowadzonych informacji, ale w następnych rozdziałach dodamy odpowiednie mechanizmy. Kliknięcie przycisku *Zapisz* będzie powodowało powrót do głównej strony aplikacji *TrackMyWalks* (zobacz rysunek na następnej stronie).

Na zrzucie ekranu przedstawiono sposób nawigacji między stronami, gdy użytkownik wybierze z listy szlak. Na ostatnim ekranie pokazany jest przebyty dystans wraz ze szpilką wskazującą miejsce szlaku na mapie.

Kup książkę Poleć książkę

Gratulacje! Właśnie ukończyłeś prace nad podstawową strukturą aplikacji *TrackMyWalks* i interfejsem użytkownika wszystkich stron treści, które będą wykorzystywane w tej aplikacji. W następnych rozdziałach zastosujemy w naszej aplikacji lepsze wzorce projektowe, zdefiniujemy ładniejsze elementy interfejsu użytkownika oraz dodamy mechanizm synchronizacji danych na bieżąco oparty na REST-owych interfejsach API usług sieciowych.

Podsumowanie

W tym rozdziale pokazałem Ci, jak utworzyć wieloplatformową aplikację `Xamarin.Forms` dla systemów iOS i Android. Potem opisałem kilka stron treści ze statycznymi danymi.

W następnym rozdziale przyjrzymy się, jak za pomocą domyślnych interfejsów API nawigacji `Xamarin.Forms` można przechodzić między stronami z treścią (które w znacznym stopniu omówimy w rozdziale 3.) z zastosowaniem w nich elastyczniejszej usługi nawigacyjnej, łatwiej poddającej się konfiguracji. Na koniec omówiłem najważniejsze różnice między środowiskami programistycznymi Xamarin Studio i Microsoft Visual Studio oraz pokazałem, jak uruchomić aplikację *TrackMyWalks* w symulatorze.

W kolejnym rozdziale poznasz wzorzec architekturalny MVVM (ang. *Model-View-ViewModel* — „model – widok – model widoku") i dowiesz się, jak go zaimplementować w swojej aplikacji, oraz nauczysz się dodawać nowe modele widoków do swojego rozwiązania Xamarin.

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Skorowidz

Kup książkę Poleć książkę

G

I

J

K

Kup książkę Poleć książkę

L

M

Kup książkę Poleć książkę

N

O

P

Kup książkę Poleć książkę

Kup książkę Poleć książkę

W

X

Z

Ż

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

Oto Xamarin: funkcjonalność, elastyczność, niezależność od platformy!

Platforma Xamarin jest wszechstronnym, nowoczesnym narzędziem do budowania aplikacji działających w wielu systemach. Pozwala na pisanie kodu w języku C# i ułatwia tworzenie niestandardowych widoków, układów i kontrolek. Interfejsy graficzne tworzone w Xamarin urzekają swoją estetyką. Platformy Xamarin i Xamarin.Forms zdobywają coraz większą popularność i właściwie stają się standardową technologią pisania oprogramowania na urządzenia mobilne.

Dzięki tej książce dowiesz się, jak zaimplementować struktury i układy interfejsu użytkownika, a także jak tworzyć własne elementy i pisać służące do ich obsługi skrypty C#. Poznasz architekturę MVVM i nauczysz się implementować ją w swoich aplikacjach. Zrozumiesz trudniejsze zagadnienia, takie jak włączanie do aplikacji funkcji specyficznych dla danej platformy mobilnej, współpraca z usługami Microsoft Azure App, korzystanie z pakietu SDK Facebooka oraz obsługa zewnętrznych bibliotek, takich jak Razor. Dowiesz się też, jak zaimplementować kluczowe techniki wiązania danych i efektów wizualnych w interfejsach użytkownika za pomocą własnych rendererów i interfejsu API PlatformEffects.

Najważniejsze zagadnienia:

- model aplikacji w postaci klas C#
- wzorzec MVVM i implementacja architektury zgodnej z tym wzorcem
- funkcje zależne od lokalizacji
- komponenty współpracujące z bazą SQLite
- testy aplikacji za pomocą narzędzi NUnit i UITest

Steven F. Daniel — jest prezesem australijskiej firmy programistycznej GENIESOFT STUDIOS. Od niemal 20 lat tworzy oprogramowanie na komputery PC i urządzenia mobilne. Jest znany z tego, że chętnie dzieli się swoim bogatym doświadczeniem z programistami, a środowisko Xamarin to jedno z jego ulubionych narzędzi. Daniel należy do SQLSIG (SQL Server Special Interest Group), Melbourne CocoaHeads i Java Community.

Helion
helion.pl
0 801 339900
0 601 339900
INFORMATYKA W NAJLEPSZYM WYDANIU
Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
WWW.SZKOLENIA.HELION.PL
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-3948-4
9 788328 339484
Cena: 67,00 zł
Packt