Dawid Borycki

Programowanie inteligentnych zegarków Apple Watch

W poprzednim artykule (Programista 3/2015) omówiłem technologię Xamarin.Forms, która umożliwia programowanie uniwersalnych aplikacji mobilnych z wykorzystaniem zunifikowanego interfejsu programistycznego oraz jednolitego zestawu komponentów wizualnych. W tym artykule zaprezentuję możliwości bibliotek WatchKit z Xamarin.iOS w kontekście tworzenia aplikacji dla Apple Watch.

Apple Watch to inteligentny zegarek (smartwatch), który oprócz charakterystycznej dla zegarków funkcjonalności odmierzania czasu posiada zestaw wbudowanych czujników. Umożliwiają one między innymi pomiar jego położenia i orientacji w przestrzeni (akcelerometr, żyroskop), rytmu serca użytkownika, siły nacisku na ekran dotykowy, a także ciśnienia atmosferycznego (barometr). Jednostką centralną smartwatcha od Apple jest procesor S1, definiowany przez producenta jako kompletny komputer znajdujący się w jednym układzie scalonym. Apple Watch jest wyposażony w 512 MB pamięci oraz wyświetlacz skonstruowany w oparciu o technologię organicznych diod elektroluminescencyjnych (OLED). Całość kontroluje system operacyjny Watch OS.

Tworzenie aplikacji dla urządzeń ubieralnych Apple Watch jest możliwe dzięki interfejsowi programistycznemu WatchKit. Uruchamianie tak zaimplementowanych aplikacji wymaga jednak wcześniejszego sparowania zegarka z iPhonem 5, 5s, 6 lub 6 plus z systemem iOS w wersji co najmniej 8.2. Wynika to z faktu, że na strukturę aplikacji WatchKit składają się dwa elementy: rozszerzenie standardowego projektu dla iPhone'a (WatchKit extension) oraz moduł zawierający definicję interfejsu użytkownika (UI) aplikacji Apple Watch. Pierwsza z tych aplikacji, działając w tle na urządzeniu nadrzędnym, aktualizuje UI zegarka i obsługuje interakcje z jego użytkownikiem. Natomiast w systemie smartwatcha jest instalowana wyłącznie moduł definiujący UI.

Istnieją trzy główne rodzaje UI dla Apple Watch. Są nimi WatchKit Apps, kontrolery typu Glance oraz Notification. Pierwszy z nich dostarcza pełen, interaktywny UI aplikacji dla Apple Watch. Kontrolery typu Glance umożliwiają prezentację zwięzłych komunikatów i nie obsługują interakcji z użytkownikiem zegarka. Z kolei kontrolery typu Notification służą do wyświetlania komunikatów wypychanych.

Aplikacje dla WatchKit można implementować z wykorzystaniem narzędzi dostarczanych przez Apple, czyli środowiska Xcode w wersji co najmniej 6.2, zestawu narzędzi programistycznych (SDK) dla iOS (8.2 lub nowszej) oraz języków programowania Objective-C lub Swift. Alternatywnie można posłużyć się bibliotekami Xamarin.iOS, które umożliwiają tworzenie natywnych aplikacji dla iOS oraz WatchKit w oparciu o platformę .NET i języki programowania C# lub F#. Dodatkowo, Xamarin dostarcza własne środowisko programistyczne Xamarin Studio.

Biblioteki Xamarin były już omawiane na łamach „Programisty" i z tego powodu nie będę ich tutaj dokładnie opisywał. Zamiast tego skoncentruje się na ich wykorzystaniu do tworzenia aplikacji WatchKit. Opis będzie obejmował zagadnienia podstawowe, polegające na utworzeniu i skonfigurowaniu projektu, implementacji prostej funkcjonalności typu *Witaj, świecie!*, jak również bardziej złożone zagadnienia jak chociażby współpraca z aplikacją nadrzędną.

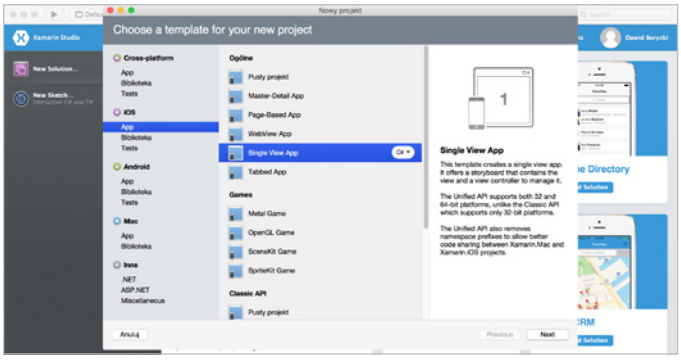
TWORZENIE I KONFIGURACJA PROJEKTU

WatchKit wymaga systemu operacyjnego OS X w wersji Mavericks lub Yosemite oraz zintegrowanego środowiska programistycznego Xcode w wersji 6.2 lub wyższej. Instalacja tych narzędzi, jak również bibliotek Xamarin i środowiska Xamarin Studio przebiega automatycznie. Z tego powodu nie będę tutaj dokładnie opisywał tej procedury. Po zainstalowaniu wszystkich narzędzi uruchamiamy Xamarin Studio, a następnie za jego pomocą tworzymy nowy projekt (*Plik/Nowy/Solution*). Spowoduje to uruchomienie kreatora *Nowy projekt* (Rysunek 1), w którym:

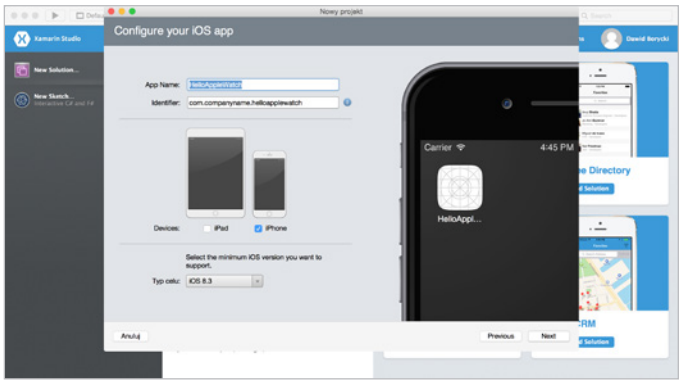
1. Z grupy iOS/App wybieramy szablon *Single View App* i klikamy przycisk z etykietą *Next*.
2. W kolejnym kroku kreatora (Rysunek 2) zmieniamy nazwę aplikacji na *HelloAppleWatch*, z grupy dostępnych urządzeń zaznaczamy *iPhone*, a następnie ustalamy typ celu (wersji SDK dla iOS) na 8.2 lub wyższą.
3. W ostatnim etapie konfiguracji projektu potwierdzamy jego nazwę i lokalizację, a następnie klikamy przycisk z etykietą *Create* (Rysunek 3).

Utworzony projekt obejmuje obecnie wyłącznie kod źródłowy aplikacji nadrzędnej uruchamianej na urządzeniach iOS. Jednakże aby móc tworzyć aplikacje dla zegarków Apple Watch, należy jeszcze uzupełnić główny projekt aplikacji *HelloAppleWatch* o projekt-rozszerzenie dla WatchKit. W tym celu w oknie *Solution (Rozwiązanie)* Xamarin Studio z menu podręcznego projektu *HelloAppleWatch* wybieramy opcję *Add/New Project*. Następnie w kreatorze tworzenia nowego projektu, zaznaczamy pozycję *Extension*, znajdującą się w grupie iOS i z listy dostępnych szablonów wybieramy *WatchKit App* (Rysunek 4). W efekcie zostanie uruchomiony kreator tworzenia aplikacji Apple Watch (Rysunek 5), w którym z listy rozwijanej *Project* wybieramy *HelloWorldAppleWatch* i klikamy przycisk z etykietą *Next*, a w kolejnym kroku kreatora przycisk z etykietą *Create*.

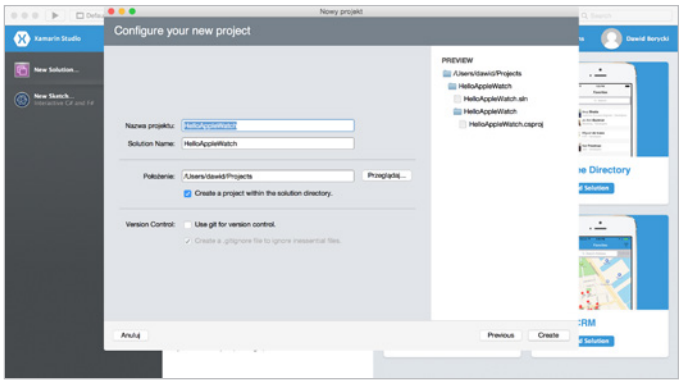
Struktura projektu przygotowanego według powyższego przepisu obejmuje teraz następujące elementy: *HelloAppleWatch*, *HelloApplewatch.UITests*, *HelloAppleWatchWatchKitApp* oraz *HelloAppleWachWatchKitExtension*. Pierwszy z nich to projekt aplikacji głównej dla iPhone'a. Drugi obejmuje natomiast testy jednostkowe interfejsu użytkownika tej aplikacji. Trzeci z wymienionych projektów, *HelloAppleWatchWatchKitApp*, implementuje interfejs użytkownika aplikacji Apple Watch. Z kolei ostatni z projektów, *HelloAppleWachWatchKitExtension*, implementuje logikę aplikacji dla Apple Watch w oparciu o zestaw odpowiednich kontrolerów. Domyślna struktura tego projektu zawiera jeden plik implementujący kontroler. Jest nim *InterfaceController.cs*.



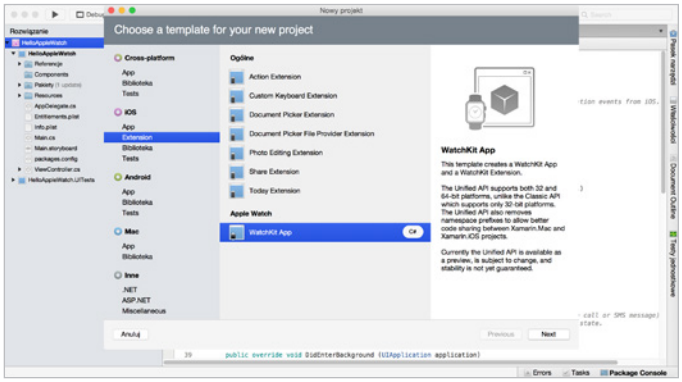
Rysunek 1. Kreator nowego projektu w Xamarin Studio



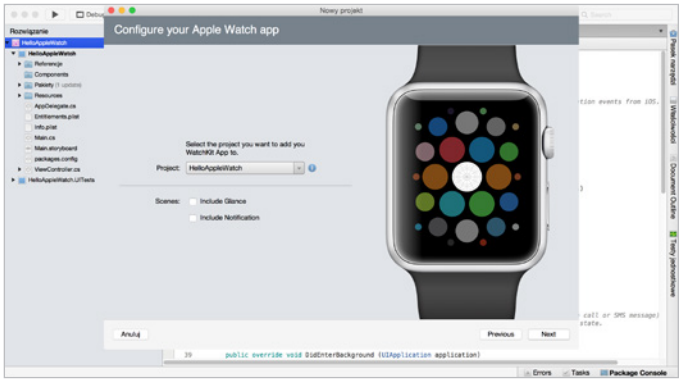
Rysunek 2. Konfiguracja aplikacji



Rysunek 3. Konfiguracja lokalizacji projektu



Rysunek 4. Kreator nowego projektu Xamarin Studio z zaznaczonym szablonem WatchKit App

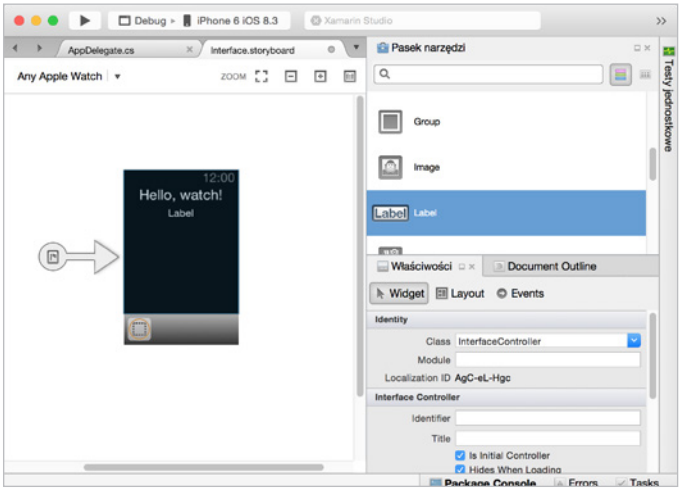


Rysunek 5. Konfiguracja aplikacji-rozszerzenia Apple Watch

IMPLEMENTACJA FUNKCJONALNOŚCI TYPU "WITAJ, ŚWIECIE!"

Interfejs użytkownika aplikacji WatchKit można, podobnie jak aplikacji iOS, zdefiniować w trybie wizualnym poprzez edycję scenorysu (ang. *storyboard*). Projekt *HelloAppleWatchWatchKitApp*, który utworzyliśmy w poprzednim podrozdziale, zawiera jeden domyślny plik z definicją scenorysu, *Interface.storyboard*. Dwukrotne kliknięcie tego pliku spowoduje uaktywnienie trybu wizualnego projektowania UI (Rysunek 6). Wykorzystamy ten tryb do uzupełnienia widoku aplikacji Apple Watch o statyczny komunikat o treści *Hello, watch!*. Dodatkowo, utworzymy etykietę, która będzie prezentowała aktualny czas systemowy. Realizacja tego zadania obejmuje wykonanie następujących czynności:

1. W trybie wizualnej edycji scenorysu przechodzimy do paska narzędzi, na którym odnajdujemy etykietę (kontrolka *Label*), która jest zlokalizowana w grupie *Controls*.
2. W obrębie prostokąta ilustrującego ekran zegarka Apple umieszczamy dwie etykiety.
3. Właściwości *Name* oraz *Text* pierwszej z nich zmieniamy odpowiednio na *helloLabel* oraz na *Hello, watch!* W tym celu wykorzystujemy okno właściwości, które domyślnie znajdują się bezpośrednio pod paskiem narzędzi.
4. Przechodzimy do edycji właściwości drugiej z etykiet i w polu *Name* wpisujemy *timeLabel*, a właściwość *Font* zmieniamy na *Caption1*.
5. Właściwość *Position.Horizontal* obu etykiet ustawiamy na *Center*.
6. Zapisujemy wykonane zmiany i przechodzimy do edycji pliku *InterfaceController.cs* z projektu *HelloAppleWatchWatchKitExtension* i uzupełniamy jego zawartość o polecenia wyróżnione na Listingu 1.



Rysunek 6. Widok wizualnego projektowania interfejsu użytkownika aplikacji WatchKit

Listing 1. Implementacja kontrolera aplikacji Apple Watch

```
using System;
using WatchKit;
using Foundation;
using System.Timers;

namespace HelloAppleWatchWatchKitExtension
{
  public partial class InterfaceController :
  WKInterfaceController
  {
    private Timer timer;

    public InterfaceController(IntPtr handle)
      : base(handle)
    {
      ConfigureTimer();
    }

    public override void Awake(NSObject context)
    {
```