Dawid Borycki

ASP.NET 5. Programowanie nowoczesnych aplikacji internetowych

Wraz ze wzrostem możliwości łączy komunikacyjnych oraz powszechną dostępnością różnego rodzaju urządzeń mobilnych, tradycyjne technologie programowania aplikacji internetowych wymagają ich przeprojektowania. Przykładem takiej technologii jest ASP.NET, która doczekała się niedawno piątej wersji, unifikującej internetowe technologie Microsoftu, takie jak ASP.NET MVC, Web API oraz Web Pages. Takie podejście wymagało sporych zmian w szablonach projektów aplikacji webowych w Visual Studio 2015, które omówię w tym artykule.

WPROWADZENIE

Technologia ASP.NET 5, nazywana również ASP.NET vNext, to zbudowana od podstaw nowa wersja technologii ASP.NET umożliwiająca tworzenie nowoczesnych aplikacji internetowych. ASP.NET 5 jest platformą o otwartym kodzie źródłowym, dystrybuowanym za pomocą systemu kontroli wersji GitHub. Z technologią ASP.NET vNext związanych jest szereg nowych elementów oraz funkcjonalności. Między innymi ASP.NET 5 bazuje na nowym, wieloplatformowym środowisku uruchomieniowym oraz unifikuje modele programowania aplikacji ASP.NET MVC, serwisów Web API oraz technologii Web Pages, co jest również określane mianem ASP.NET MVC 6.

Aplikacje ASP.NET 5 mogą działać w oparciu o trzy różne środowiska CLR (ang. *Common Language Runtime*), skojarzone z odpowiednimi wersjami platformy .NET. Są nimi: Full .NET CLR, Core CLR oraz Cross-Platform CLR. Jak sugerują nazwy, w pierwszym przypadku środowisko uruchomieniowe wspiera wszystkie składniki platformy .NET. Z kolei Core CLR jest zoptymalizowaną, modułową wersją środowiska Full .NET CLR. Natomiast Cross-Platform CLR jest środowiskiem, które w najbliższej przyszłości ma umożliwić uruchamianie aplikacji .NET na systemach operacyjnych z rodziny Linux oraz Mac OS, a także wspierać tworzenie aplikacji .NET dla systemów Android i iOS.

Wprowadzenie nowych kompaktowych, przenośnych wersji biblioteki .NET oraz odpowiadających im środowisk uruchomieniowych było podyktowane zwiększającą się liczbą platform oraz urządzeń, na których uruchamiane są aplikacje wykorzystujące biblioteki .NET. Od momentu wydania pierwszej wersji platformy .NET, dedykowanej desktopowym systemom Windows, Microsoft udostępnił kilka specyficznych wersji swoich bibliotek, przeznaczonych dla innych urządzeń i systemów, na przykład .NET Micro Framework czy .NET Compact Framework. W podobny sposób powstawały również Windows Phone, Windows Store czy Silverlight.

Chociaż niektóre z tych narzędzi nie są już aktywnie rozwijane, to okazało się, że taki model wydzielania funkcjonalności specyficznych dla danej platformy, a następnie ich niezależnego rozwijania nie jest najlepszym rozwiązaniem. Wynika to z faktu, że takie podejście utrudnia współdzielenie kodu źródłowego pomiędzy różnymi platformami. Własność ta jest szczególnie pożądana w dzisiejszych projektach programistycznych, w ramach których powstające aplikacje coraz częściej muszą być kompatybilne z wieloma platformami. Implementacja całkowicie niezależnych wersji aplikacji dla różnych platform byłaby kosztowna i czasochłonna.

Z powyższych powodów przeprojektowano platformę .NET i utworzono jej dwie wersje: .NET Framework oraz .NET Core. Pierwsza z nich zawiera pełny zestaw interfejsów programistycznych i zapewnia pełną kompatybilność wsteczną. Natomiast .NET Core została zoptymalizowana dla systemów chmurowych oraz urządzeń przenośnych. Platforma .NET Core składa się z czterech warstw. Najniżej znajduje się warstwa uruchomieniowa, bezpośrednio nad którą umieszczono warstwę adaptacyjną RAL (od ang. *Runtime Adaptation Layer*), a nad nią warstwę zunifikowanych bazowych bibliotek klas BCL (od ang. *Base Class Library*). Najwyższa warstwa zawiera modele aplikacji, czyli wysokopoziomowe elementy, składające się na podstawową strukturę aplikacji danego typu, na przykład widoki, kontrolki czy kontrolery. W .NET Core warstwa ta zawiera modele aplikacji Windows Store App Model oraz ASP.NET 5 App Model. Natomiast podstawowa warstwa uruchomieniowa zawiera dwa środowiska: .NET Native Runtime oraz Core CLR. Pierwsze z nich przeznaczone jest dla aplikacji Windows Store uruchamianych na urządzeniach mobilnych. Z kolei drugie z nich, Core CLR, służy do uruchamiania aplikacji internetowych ASP.NET 5. Warstwa adaptacji RAL przystosowuje biblioteki BCL do danego środowiska uruchomieniowego.

Środowisko .NET Core jest modułowe w tym sensie, że warstwa BCL składa się z bibliotek dystrybuowanych w ramach pakietów NuGet. Dzięki temu dana aplikacja wykorzystuje jedynie te elementy, które są rzeczywiście potrzebne. Dodatkowo, pakiety NuGet ułatwiają implementację aplikacji w oparciu o dużą liczbę istniejących już bibliotek.

Oprócz nowego projektu platformy .NET, drugim z najbardziej istotnych aspektów dotyczących ASP.NET 5 jest unifikacja aplikacji MVC oraz serwisów Web API. W poprzednich wersjach ASP.NET MVC, Web API oraz Web Pages niezależnie implementowały podobne funkcjonalności. Na przykład MVC oraz Web API wykorzystywały podobny mechanizm routowania. W ramach ASP.NET 5, MVC oraz Web API są połączone w jedną strukturę.

ASP.NET vNext wykorzystuje również wzorzec projektowy, określany wstrzykiwaniem zależności (ang. *Dependency Injection /DI/*). W technice DI, usługi, czyli współdzielone obiekty, są instancjonowane w kontenerze, będącym dostawcą serwisów. Dostawca dostarcza (wstrzykuje) usługi do klas klienckich dopiero w momencie ich użycia. Pozwala to jawnie odseparować konstrukcję współdzielonych obiektów od funkcjonalności klas klienckich.

Przybliżę te koncepcje w kolejnych rozdziałach, w których najpierw utworzę projekt aplikacji internetowej ASP.NET 5 i uzupełnię ją o bazę danych, którą utworzę w strategii *Code First*. W kolejnym kroku utworzę serwis Web API. W celu przetestowania poprawności jego działania zaimplementuję również aplikację Windows Store, która będzie wykorzystywała dane z serwisu sieciowego.

Na potrzeby realizacji przykładów wykorzystam środowisko VS 2015 Ultimate Preview, które można pobrać bezpłatnie spod adresu http://www.visualstudio.com/en-us/downloads/visual-studio-2015-downloads-vs.aspx.

Podczas implementacji powyższych przykładów pominę szczegółowy opis poszczególnych aspektów technologii ASP.NET MVC. Tym zagadnieniom była poświęcona seria artykułów, autorstwa Karola Rogowskiego, opublikowanych w „Programiście" (7/2012(07) – 7/2013(14)).

SZABLON PROJEKTU APLIKACJI

Implementację aplikacji ASP.NET vNEXT rozpoczynamy od utworzenia projektu aplikacji ASP.NET Web Application wg szablonu ASP.NET 5 Starter Web. W tym celu:

Zamów w prenumeracie