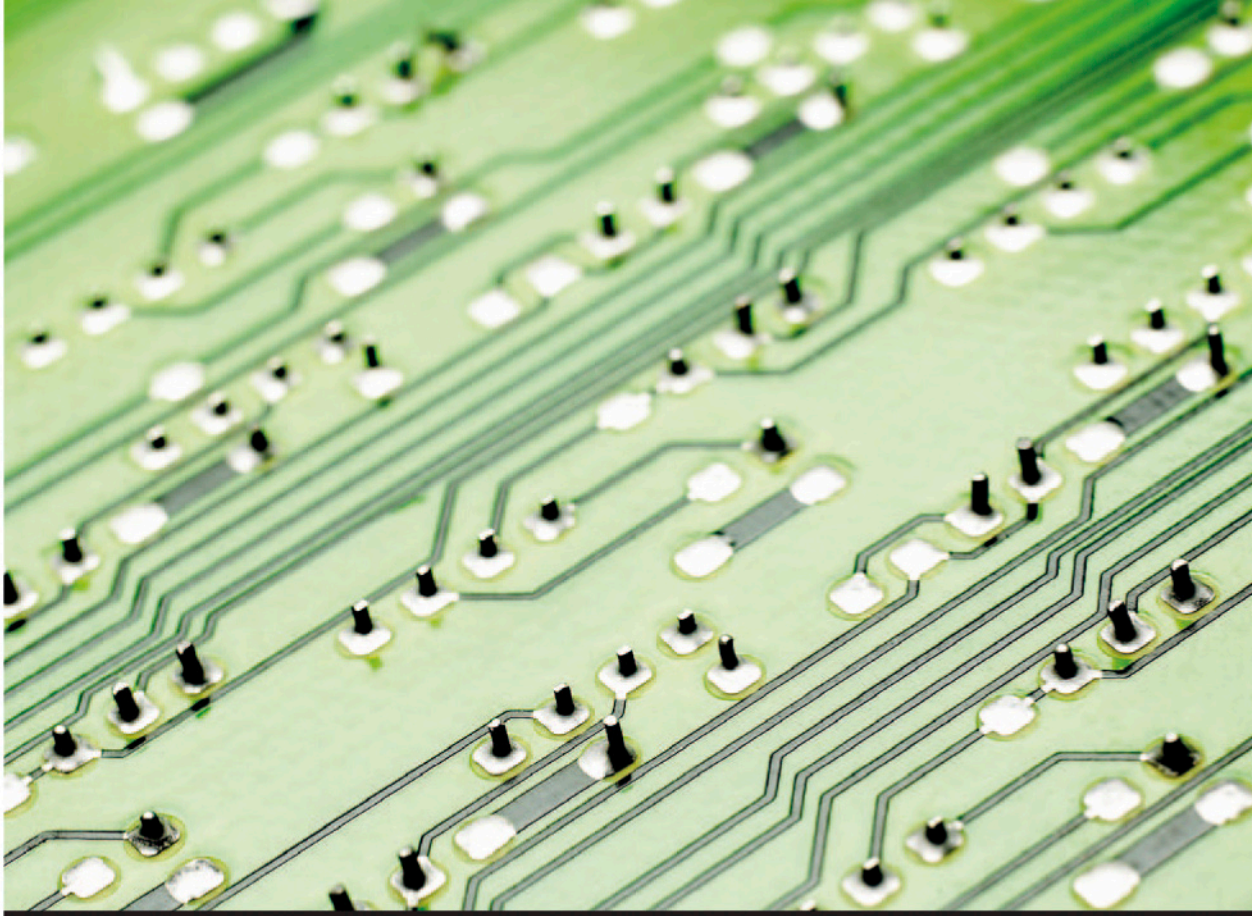


Szybkie rozwiązania typowych problemów

Visual Studio 2012 i .NET 4.5

Poradnik dla zaawansowanych programistów

Ponad 40 przepisów skutecznego łączenia wydajnych możliwości .NET 4.5 i Visual Studio 2012

Abhishek Sur

[PACKT] PUBLISHING

Abhishek Sur

Visual Studio 2012 i .NET 4.5

Poradnik dla zaawansowanych programistów

Ponad 40 przepisów skutecznego łączenia wydajnych możliwości .NET 4.5 i Visual Studio 2012

Przekład: Jakub Niedźwiedź

APN Promise, Warszawa 2013

Visual Studio 2012 i .NET 4.5. Poradnik dla zaawansowanych programistów

Original English language edition © 2013 Packt Publishing

APN PROMISE SA, ul. Kryniczna 2, 03-934 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: mspress@promise.pl

ISBN: 978-83-7541-131-7

Przekład: Jakub Niedźwiedź

Fotografia na okładce: © istockphoto/imagestock

Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWART Marek Włodarz

O autorze

Abhishek Sur jest posiadaczem tytułu Microsoft MVP w dziedzinie programowania aplikacji klienckich od 2011 roku. Jest architektem platformy .NET. Ma dogłębną wiedzę teoretyczną i wieloletnie praktyczne doświadczenie w dziedzinie różnych produktów i języków .NET. Przez lata pomagał programistom na całym świecie wykorzystując swoje doświadczenie i wiedzę. Kieruje w Kalkucie grupą użytkowników Microsoft pod nazwą **KolkataGeeks** i regularnie organizuje spotkania i seminaria w różnych miejscach szerząc wiedzę na temat .NET. Jest uznanym mówcą, pożeraczem książek i znawcą technologii, jego główne zainteresowania dotyczą nowych dziedzin technologii .NET, a także lubi pisać na temat mało zbadanych zagadnień związanych z .NET. Jest związany z listą Microsoft Insider dotyczącą WPF i C#, jest też w stałym kontakcie z zespołami pracującymi nad tymi produktami. Ma tytuł magistra informatyki i zdobył wiele różnych certyfikatów.

W sieci WWW Abhishek działa jako autor, programista i administrator. Jego witryna WWW *abhisheksur.com* pomaga zarówno początkującym, jak i zaawansowanym programistom w zrozumieniu szczegółów związanych z językami i najnowszą technologią. Cieszy się ogromnym poparciem fanów w sieciach społecznościowych. Można się z nim skontaktować pod adresem poczty elektronicznej *books@abhisheksur.com*, przez serwis Facebook lub na Twitterze pod `@abhi2434`.

Podziękowania

Napisanie tej książki nie byłoby możliwe bez pomocy wielu ludzi. Wiedza jest nieocenionym dobrem. Podczas pisania tej książki poświęciłem znaczne ilości czasu na czytanie czasopism, dokumentacji i blogów, żeby wyjaśnić i przedstawić omawiane pojęcia i pomysły w jak najbardziej przejrzysty sposób.

Przede wszystkim jestem niezmiernie wdzięczny całemu zespołowi Packt Publishing za nieustanną pomoc. Chciałbym podziękować Anish Ramchandani i Susmita Panda za wsparcie i pomaganie mi w dotrzymaniu terminów. Szczególne podziękowania kieruję na ręce moich recenzentów: Carlos Hulot, Ahmed Illyas, Sergiy Suchok i Ken Tucker za przejrzenie każdego fragmentu tej książki, co doprowadziło do wprowadzenia wielu zmian na lepsze. Serdeczne podziękowania należą się Amit Ramadas i Jalasha D'costa za redakcję rozdziałów.

Chciałbym również podziękować swoim kolegom Shibatosh, Ranjit, Pallab, Ayan, Malini i innym członkom naszej grupy za ciągłe wsparcie i motywowanie do napisania tej książki. Jestem też wdzięczny za współpracę panu Raj Goswami, prezesowi BuildFusion w Indiach i panu D. K Goswami, dyrektorowi tej firmy. Chciałbym też przekazać serdeczne podziękowania Anoop Madhusudan, Abhijit Jana, Kunal Chowdhury, Dhananjay Kumar, Karthikayan Anabarasan, Lohith, Abhishek Kant i Shivprasad Koirala za ich nieustanną pomoc i motywację.

Chciałbym też podziękować swojej narzeczonej Riya za pomoc i motywowanie do lepszego pisania. Jej chcę zadedykować tę książkę.

Jako fan społeczności pisałem wiele blogów i artykułów, ale po raz pierwszy podjąłem wysiłki związane z przedstawieniem swoich doświadczeń w formie książki. Szczerze dziękuję wszystkim swoim Czytelnikom na całym świecie, którzy lubili moje blogi i mam nadzieję, że polubią też tę książkę.

O recenzentach

Carlos Hulot pracuje w IT od ponad 20 lat w różnych dziedzinach od programowania, zarządzania projektami do marketingu oraz rozwijania produktów. Carlos pracował dla międzynarodowych firm, takich jak Royal Philips Electronics, PricewaterhouseCoopers i Microsoft. Obecnie Carlos pracuje jako niezależny konsultant informatyczny. Carlos wykłada informatykę na dwóch uniwersytetach brazylijskich; ma tytuł doktora informatyki i elektroniki na uniwersytecie Southampton w Wielkiej Brytanii oraz tytuł magistra fizyki na uniwersytecie São Paulo w Brazylii.

Ahmed Ilyas ma tytuł magistra inżynierii na uniwersytecie Napiera w Edynburgu w Szkocji i specjalizuje się w tworzeniu oprogramowania. Ma 15 lat doświadczenia w programowaniu.

Po opuszczeniu firmy Microsoft założył swoją firmę konsultingową oferującą najlepsze rozwiązania dla wielu firm i zapewniającą odpowiedzi na wiele problemów. Wykorzystuje technologie Microsoft do budowania rozwiązań dostarczając klientom najlepsze praktyki, wzorce i oprogramowanie. Zapewnia długoterminową stabilność i zgodność w stale zmieniającej się branży programistycznej oraz pomaga rozwijać się programistom na całym świecie.

Dzięki temu zdobył trzykrotnie tytuł MVP w dziedzinie języka C# zapewniając niezależne rozwiązania rzeczywistych problemów napotykanych przez programistów.

Posiada szeroki zakres wiedzy zdobyty dzięki badaniom oraz pracy w firmie Microsoft oraz czerpie motywację i inspirację z faktu, że 90 procent użytkowników na świecie korzysta z przynajmniej jednej formy technologii Microsoft.

Ahmed Ilyas pracował dla wielu klientów i pracodawców. Dzięki świetnej reputacji zdobył dla swojej firmy konsultingowej Sandler Ltd (UK) mnóstwo klientów pochodzących z różnych branż – od mediów po szpitale. Niektórzy klienci umieścili go na swoich listach „polecanych konsultantów". Należą do nich firmy ICS Solution Ltd (został wymieniony na ich portalu „Drużyna marzeń") oraz CODE Consulting/EPS Software (*www.codemag.com*, z siedzibą w USA).

Ahmed Ilyas brał też w przeszłości udział w recenzowaniu książek dla Packt Publishing i chciałby podziękować za kolejną taką okazję.

> Chciałbym podziękować autorowi/wydawcy tej książki za wielki przywilej recenzowania tej książki.

Sergiy Suchok w 2004 roku ukończył z wyróżnieniem fakultet cybernetyki na Narodowym Uniwersytecie im. Tarasa Szewczenki w Kijowie na Ukrainie i stał się od tej pory fanem technologii informatycznych. Obecnie pracuje w dziedzinie bankowości i ma doktorat z ekonomii. Sergiy jest współautorem ponad 45 artykułów i uczestniczył w ponad 20 naukowych i praktycznych konferencjach poświęconych ekonomii i modelowaniu matematycznemu. Jest członkiem organizacji New Atlantis Youth Public Organization (*newatlantida.org.ua*) i poświęca swój czas wolny na sprawy związane z ochroną środowiska naturalnego, popularyzacją patriotyzmu i szacunkiem dla Ziemi. Pisze też poezję i opowiadania oraz zajmuje się makramą.

Chciałbym podziękować swojej żonie i córce za ich cierpliwość
i zrozumienie podczas recenzowania tej książki.

Ken Tucker jest programistą WWW w Sea World i ma tytuł Microsoft MVP od października 2003. W wolnym czasie zajmuje się pisaniem aplikacji dla Windows Phone i Windows Store.

Spis treści

Wstęp

W ciągu ostatnich kilku dekad platforma .NET przekształciła się z całkowitej nowości do najpopularniejszej technologii informatycznej. Coraz więcej osób sięga po .NET w trakcie swojej kariery, a coraz więcej klientów realizuje swoje marzenia dzięki .NET. Niedawny rozgłos wokół języka C# jako najlepszego języka programowania jeszcze bardziej zachęcił świat programistów do budowania aplikacji .NET. Jako że tłum na arenie programistycznej gęstnieje, istnieje potrzeba jasnego zrozumienia, jak działają różne elementy tej technologii i jak poprawić dzięki nim swoje aplikacje.

Visual Studio 2012 i .NET 4.5: Poradnik dla zaawansowanych programistów skupia się głównie na tym, co programista powinien wiedzieć w poszczególnych sytuacjach. Próbuje też dostarczyć możliwie dużo szczegółów i omawia wiele dziedzin technologicznych ze świata .NET. Książka ta jest napisana w formie przepisów ze wskazówkami pokazującymi krok po kroku dane zagadnienie, a programista może towarzyszyć autorowi w tej cudownej podróży do znanych i dotąd niepoznanych obszarów .NET. Przepisy, które są często poszukiwane przez programistów w Internecie, zostały wybrane do tej książki w taki sposób, żeby praktycznie przedstawiać dane zagadnienie i nie ograniczać programistów, ale ukazywać im inne niezbadane dziedziny związane z tym samym tematem dając nieco szerszą perspektywę. Dla każdego przepisu zamieszczono specjalny dział pod nagłówkiem *Więcej...*, który zawsze skupia się na przekazywaniu dodatkowej wiedzy dotyczącej elementów, które mogłyby umknąć uwadze Czytelnika. Zanim dotrzemy do końca tej podróży, będziemy mogli poczuć ulgę i cieszyć się pewnością siebie, którą uzyskamy dzięki wyraźnemu zrozumieniu zagadnień .NET.

Ta książka jest praktycznym podręcznikiem, który pozwoli optymalnie wykorzystać czas na naukę. Omawia poszczególne tematy oddzielnie i bardzo dokładnie. Wyjaśnia przepisy korzystając z bloków kodu przykładowego, które jasno wyjaśniają

użycie danego zagadnienia i pozwala wykorzystać końcowy kod źródłowy podczas pisania prawdziwych aplikacji. Przykłady wykorzystane w tej książce ułatwią zrozumienie, jak działają poszczególne elementy danego przepisu oraz pozwolą na efektywne i szybkie wykorzystanie tego samego kodu źródłowego w środowisku produkcyjnym. Ta książka jest przeznaczona dla osób, które chcą wykorzystać swój cenny czas na zbadanie wszystkich niezbędnych technologii .NET dostępnych na rynku.

Książka ta skupia się na dostarczeniu:

- Maksymalnych korzyści z czasu poświęconego na naukę
- Dogłębnego omówienia technologii, takich jak Visual Studio 2012, .NET 4.5, ASP.NET, aplikacje Windows 8, Windows Presentation Foundation, HTML5, jQuery, zarządzanie pamięcią, itd.
- Praktycznych przykładów procedur tworzenia rzeczywistych aplikacji
- Przykładów wyjaśniających krok po kroku, jak tworzyć proste aplikacje

Co omawia ta książka

Rozdział 1, Wprowadzenie do funkcji środowiska programistycznego Visual Studio, zaczyna od wprowadzenia do podstaw środowiska programistycznego Visual Studio i daje wgląd w sposoby zwiększenia wydajności programowania dzięki zastosowaniu zestawu narzędzi i funkcji obecnych wewnątrz tego środowiska programistycznego.

Rozdział 2, Podstawy programów .NET i zarządzania pamięcią, przedstawia strukturę programu .NET i jego głównych składników. Zagłębia się w pokazanie infrastruktury .NET ze szczegółowym wyjaśnieniem zarządzania pamięcią i powiązanych technik.

Rozdział 3, Programowanie asynchroniczne w .NET, skupia się na omówieniu wszystkich istniejących technik dotyczących wątków w .NET oraz nowych wzorców, które zastępują dotychczas obowiązujące zasady, z dogłębnym wyjaśnieniem ich działania.

Rozdział 4, Rozszerzenia ASP.NET, stanowi wprowadzenie do najnowszych rozszerzeń ASP.NET 4.5 wykorzystujących HTML5 i jQuery. Pokazuje też sposoby zwiększenia wydajności ASP.NET dostępne w .NET 4.5 i Visual Studio 2012.

Rozdział 5, Rozszerzenia WPF, wprowadza rozszerzenia technologii WPF 4.5 oraz omawia główne składniki WPF. Przedstawia praktyczną implementację aplikacji WPF opartej na wzorcu MVVM omawiając wszystkie aspekty związane z programowaniem w środowisku WPF.

Rozdział 6, Budowanie aplikacji na urządzenia dotykowe z systemem Windows 8, przedstawia nowy model programistyczny do tworzenia aplikacji dla systemu Windows 8.

Krok po kroku wprowadza sposoby programowania aplikacji dla Windows 8 przy użyciu HTML5 i JavaScript, a także WPF i C#.

Rozdział 7, Komunikacja i udostępnianie w Windows 8, skupia się na sposobach implementacji aplikacji sieciowych w Windows 8 pokazując krok po kroku, jak działa udostępnianie i wyszukiwanie w środowisku Windows 8.

Dodatek, Języki .NET i ich konstrukcja, skupia się na dogłębnym przedstawieniu sposobów działania języków na platformie .NET, zwłaszcza języka C# i szczegółowo wyjaśnia na przykładach różne funkcje języka C#.

Co potrzeba do korzystania z tej książki

Podstawowe wymagania programowe dla tej książki są następujące:

- Platforma Microsoft .NET Framework 4.5 lub wyższa
- Microsoft Visual Studio 2012 Express lub wyższe wydanie
- System operacyjny Windows 8 (zwłaszcza do pracy z rozdziałami 6 i 7)
- Najnowsze przeglądarki WWW

Dla kogo jest ta książka

Celem tej książki jest pokazanie gotowych kroków w formie przepisów objaśniających typowe zadania programistyczne często wykorzystywane w rzeczywistych sytuacjach. Książka stara się maksymalnie wypełnić swoje rozdziały zapewniając możliwie dużo szczegółowych informacji pozwalających szybko rozpocząć pracę nad danym zagadnieniem. Książka ta dostarcza też dogłębną analizę niektórych zaawansowanych elementów danego zagadnienia, pozwalając osiągnąć w nim poziom ekspercki. Książka ta jest idealna dla kogoś, kto zaczyna pracę w środowisku programistycznym i chce uzyskać zaawansowane informacje na temat nowych technologii na rynku. Książka ta może być nawet przewodnikiem dla architektów oprogramowania i menedżerów projektów chcących wzbogacić swoją wiedzę.

Książka ta wykorzystuje w przykładach język C# i środowisko Visual Studio 2012 w systemie operacyjnym Windows 8. Choć książka ta nie wymaga na starcie żadnej konkretnej wiedzy, to oczekuje, że Czytelnik będzie miał pewne podstawy teoretyczne i ogólne doświadczenie praktyczne w omawianych zagadnieniach, żeby zrozumieć przedstawiane przepisy. Książka ta stanowi pomost pomiędzy zwykłym programistą a zaawansowanym architektem oprogramowania.

Konwencje

W tej książce znajdziemy wiele stylów tekstu, którymi oznaczane są różne rodzaje informacji. Oto kilka przykładów tych stylów oraz wyjaśnienie ich znaczenia.

Słowa pochodzące z kodu są pokazane w tekście czcionką stałopozycyjną: „Atrybut `requestValidationMode` obiektu `httpRuntime` definiuje, jak sprawdzana jest poprawność żądania do serwera przed wywołaniem `HttpPipeline`.”.

Blok kodu jest formatowany następująco:

```
(function($){
  $.fn.extend({
    Value1 : 20,
    myMethod : function(msg){
      alert(msg + "value : " + this.Value1);
    }
  });
})(JQuery);
```

Wejście lub wyjście wiersza polecenia jest zapisywane następująco:

```
Install-Package Microsoft.Web.Optimization
```

Elementy, które Czytelnik ma wpisać w konsoli, zostały pogrubione.

Nowe terminy i ważne słowa są złożone czcionką bezszeryfową pogrubioną. Słowa, które są widoczne na ekranie, na przykład w menu lub oknach dialogowych, pojawiają się w tekście w następującej formie: „możesz też otworzyć menedżera pakietów Nuget, klikając prawym przyciskiem myszy folder References w projekcie i wybierając **Add Library Package Reference** (Dodaj odwołanie do pakietu biblioteki)”. Czcionką bezszeryfową z kursywą oznaczone są kombinacje klawiszy, np. *Ctrl+C*.

Ostrzeżenia lub ważne uwagi pojawiają się w takiej ramce.

Wskazówki i porady pojawiają się w takiej ramce.

Informacje od Czytelników

Informacje zwrotne od naszych Czytelników są zawsze mile widziane. Prosimy o opinie na temat tej książki – co się w niej podoba, a co nie. Informacje od Czytelników są dla nas bardzo ważne, żebyśmy mogli przygotowywać najbardziej pożądane tytuły.

W celu przekazania ogólnych uwag wystarczy wysłać e-maila na adres *feedback@packtpub.com* podając tytuł książki w temacie wiadomości.

Jeśli ktoś jest ekspertem w jakiejś dziedzinie i chciałby napisać lub uczestniczyć w pracach nad książką, może zajrzeć do naszego przewodnika dla autorów pod adresem *www.packtpub.com/authors*.

Obsługa klienta

Dla dumnego posiadacza książki wydawnictwa Packt mamy wiele rzeczy pomocnych w maksymalnym wykorzystaniu swojego zakupu.

Pobieranie kodu przykładowego

Pliki z kodem przykładowym dla wszystkich zakupionych książek wydawnictwa Packt można pobrać ze swojego konta pod adresem *http://www.packtpub.com*. Jeśli książka została zakupiona gdzie indziej, można odwiedzić adres *http://www.packtpub.com/support*, aby się zarejestrować i otrzymać pliki pocztą elektroniczną*.

Errata

Chociaż podjęliśmy wszelkie starania, aby zapewnić poprawność treści tej książki, błędy czasem się zdarzają. W razie znalezienia błędu w jednej z naszych książek – na przykład błędu w tekście lub w kodzie – bylibyśmy wdzięczni za zgłoszenie nam go. Dzięki temu możemy oszczędzić innym czytelnikom kłopotów i poprawić kolejne wersje tej książki. Wszelkie poprawki prosimy zgłaszać pod adresem *http://www.packtpub.com/submit-errata*, wybierając daną książkę, klikając łącze do formularza zgłaszania błędów i wprowadzając jego szczegóły. Po zatwierdzeniu zgłoszenia, zostanie ono przyjęte i opublikowane na naszej stronie WWW lub dodane do listy istniejących poprawek w części Errata dla danego tytułu. Istniejące erraty można przeglądać wybierając daną książkę ze strony *http://www.packtpub.com/support*.

---

* Podczas rejestracji należy użyć tytułu lub ISBN angielskiego oryginału; zostały one podane na stronie redakcyjnej (przyp. red. wydania polskiego).

Piractwo

Internetowe piractwo materiałów chronionych prawem autorskim jest stałym problemem. W wydawnictwie Packt bardzo poważnie traktujemy ochronę naszych praw autorskich i licencji. W razie natrafienia w Internecie na nielegalne egzemplarze naszych prac w jakiejkolwiek formie prosimy o podanie nam adresu lub nazwy strony WWW.

Prosimy o kontakt pod adresem *copyright@packtpub.com* z łączem do materiału podejrzanego o piractwo.

Pytania

W przypadku problemów związanych z dowolnym aspektem tej książki prosimy o kontakt pod adresem *questions@packtpub.com*, a postaramy się pomóc. W razie problemów technicznych z omawianymi przepisami można się też skontaktować bezpośrednio z autorem pod adresem *books@abhisheksur.com*.

1

Wprowadzenie do funkcji środowiska programistycznego Visual Studio

W tym rozdziale zaczniemy od podstawowego wprowadzenia do środowiska programistycznego Visual Studio i zrozumienia, jak możemy zwiększyć wydajność programowania korzystając z niektórych narzędzi i funkcji obecnych w tym środowisku programistycznym. Po przeczytaniu tego rozdziału będziemy rozumieć następujące zagadnienia:

- Identyfikowanie różnych składników środowiska programistycznego Visual Studio
- Praca z narzędziami Solution Explorer (Eksplorator rozwiązania) i Class View (Widok klas)
- Praca z głównym obszarem roboczym środowiska programistycznego
- Nawigowanie w obrębie kodu wewnątrz środowiska programistycznego
- Rozszerzanie szablonów Visual Studio
- Korzystanie z Code Snippets (fragmentów kodu) w Visual Studio
- Korzystanie z funkcji Smart Tags (Inteligentne znaczniki) i Refactor (refaktoryzacja) w Visual Studio

Wprowadzenie

Odkąd firma Microsoft po raz pierwszy ogłosiła technologię .NET 10 lat temu, było wiele szumu wśród społeczności programistów odnośnie kierunków wyznaczanych przez te zmiany. Technologia .NET wprowadzała do kodowania bardziej zaawansowane techniki stosując w programowaniu paradygmaty zorientowane obiektowo i zmieniając cały styl kodowania. Wcześniej wprowadzony przez Microsoft język VB został zmodernizowany w nowym środowisku i przeprojektowany jako VB.NET. Ogłoszono też kilka języków z całkiem inną składnią, takich jak C#, J# i C++. Wszystkie te języki są zbudowane na bazie środowiska uruchomieniowego .NET Runtime (znanego jako **Common Language Runtime** lub **CLR** [wspólne językowe środowisko uruchomieniowe]) i tworzą ten sam efekt pośredni w postaci **MSIL – Microsoft Intermediate Language** (język pośredni firmy Microsoft).

Firma Microsoft wprowadziła środowisko uruchomieniowe .NET definiując standardowe reguły i specyfikacje, które muszą być stosowane przez każdy język wykorzystujący CLR. Całkiem nowy zestaw bibliotek, klas, składni, a nawet sposób kodowania w technologiach Microsoft stworzył ogromne wyzwanie dla społeczności programistów. Wielu programistów zmieniło pracę, a niektórzy naprawdę przestawili się na nowe tory, żeby zrozumieć, jak pracować z tą nową technologią, która była całkiem różna od swoich poprzedników. Społeczność od razu zaczęła sobie zdawać sprawę, że istniejący zestaw narzędzi Microsoft może nie sprostać potrzebom nowej, rozwijającej się technologii. Firma Microsoft musiała stworzyć silny zestaw narzędzi, żeby pomóc programistom w łatwiejszej i lepszej pracy z nową technologią.

Visual Studio stanowi odpowiedź na niektóre z tych potrzeb. Microsoft Visual Studio jest **zintegrowanym środowiskiem programistycznym** (IDE – Integrated Development Environment) do pracy z językami Microsoft. Jest to główne narzędzie dla programistów służące do pracy z technologiami Microsoft. Trzeba jednak zauważyć, że Visual Studio nie jest nowym produktem firmy Microsoft. Środowisko to istniało już od jakiegoś czasu, ale nowy pakiet Visual Studio został zaprojektowany całkiem od nowa i wydany jako Visual Studio 7.0 ze wsparciem dla języków .NET.

Ewolucja Visual Studio

Z biegiem czasu firma Microsoft wydawała coraz nowsze wersje pakietu Visual Studio z dodatkowymi funkcjami i ulepszeniami. Visual Studio może obsługiwać wiele dodatkowych usług w postaci wtyczek i obecnie dostępnych jest wiele zewnętrznych narzędzi i rozszerzeń; pakiet ten stał się integralną częścią codziennej działalności każdego programisty. Visual Studio to nie tylko narzędzie wykorzystywane przez programistów, ale też duża liczba osób, które nie są częścią społeczności programistycznej,

polubiła to zintegrowane środowisko i wykorzystuje je do edycji i zarządzania dokumentami. Szerokie rozpowszechnienie Visual Studio wśród społeczności sprawiło, że produkt ten stał się jeszcze lepszy.

W tym roku firma Microsoft wydała najnowszą wersję Visual Studio. W tym rozdziale przyjrzymy się funkcjom zintegrowanego środowiska programistycznego Visual Studio i omówimy głównie te jego części, które naprawdę mogą pomóc nam w szybszym wykonywaniu swojej pracy.

Identyfikowanie różnych składników zintegrowanego środowiska programistycznego Visual Studio

Visual Studio 2012 wyposażono w mnóstwo nowych rozszerzeń i funkcji. Niektóre z tych funkcji znacznie zwiększają wydajność programowania. Lepsza znajomość zintegrowanego środowiska programistycznego daje zawsze przewagę programiście. W tym przepisie spróbujemy zająć się różnymi funkcjami środowiska programistycznego Visual Studio.

Przygotowanie

Zanim zaczniemy korzystać z Visual Studio, musimy najpierw dokonać wyboru, która wersja będzie nam najbardziej odpowiadać. Przyjrzyjmy się funkcjom wszystkich wersji Visual Studio.

- **Visual Studio Express**: Jeśli ktoś zamierza tworzyć niewielkie lub średnich rozmiarów aplikacje i nie chce wydawać z kieszeni ani grosza, to odpowiednim wyborem będzie Visual Studio Express. Microsoft udostępnia wersję Express za darmo wszystkim, którzy chcą tworzyć oprogramowanie.
- **Visual Studio Professional**: To wydanie Visual Studio jest przeznaczone dla pojedynczego programisty i zawiera większość ważnych narzędzi do debugowania oraz wszystko, co zazwyczaj programiście będzie potrzebne. Będzie to odpowiedni wybór dla kogoś, kto będzie korzystał ze środowiska programistycznego główne do tworzenia kodu. To wydanie ma też rozsądną cenę.
- **Visual Studio Premium**: Wydanie Premium pakietu Visual Studio jest przeznaczone dla osób intensywnie wykorzystujących środowisko programistyczne. Dodaje narzędzia służące do testowania, analizy kodu, debugowania, profilowania, wykrywania typowych błędów kodowania, generowania danych testowych, itd.

- **Visual Studio Ultimate**: Jest to najbardziej rozbudowane wydanie tego produktu ze wszystkimi składnikami dostępnymi w Visual Studio. To wydanie zapewnia zaawansowane możliwości debugowania oraz narzędzia dla projektowania architektury i modelowania oprogramowania.

Pełną listę porównującą wszystkie wersje Visual Studio można znaleźć pod adresem: *http://www.microsoft.com/visualstudio/plk/products/compare*

Po wybraniu wersji najbardziej odpowiadającej naszym wymaganiom możemy zainstalować ją na swoim komputerze i zacząć pracę.

Jeśli otworzymy środowisko programistyczne po raz pierwszy, to pojawi się kilka opcji i pytanie o żądany typ programowania. W większości przypadków polecam wybranie opcji **General Development** (Programowanie ogólne), gdyż daje ona najwygodniejszy układ środowiska Visual Studio.

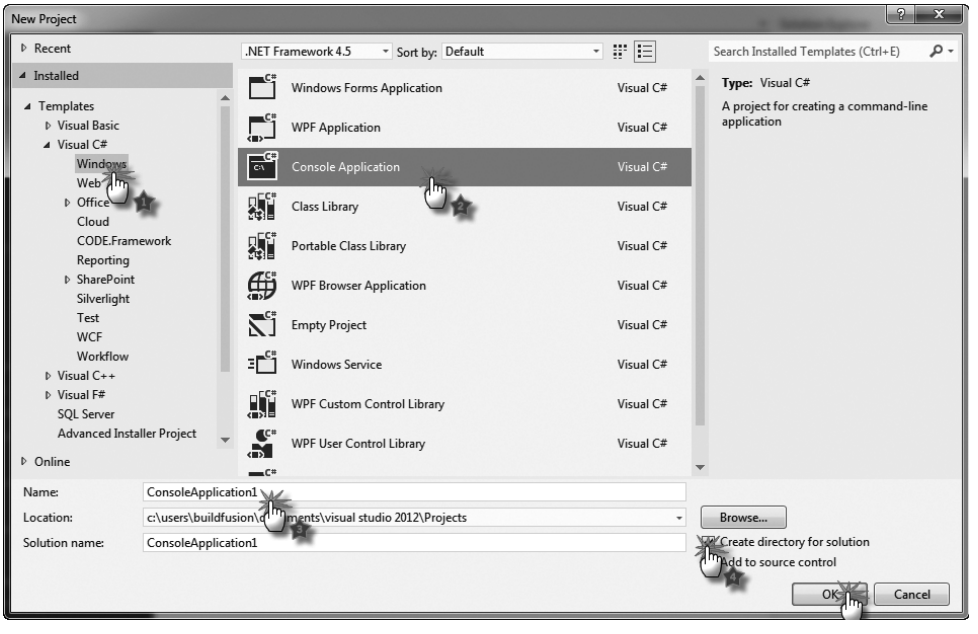
Jak to zrobić…

W tym przepisie zrozumiemy znaczenie różnych elementów środowiska programistycznego Visual Studio 2012.

1. Aby zacząć ten przepis, przejdź do menu **Start** | **Wszystkie programy**, wybierz folder `Visual Studio 2012` i wybierz opcję **Visual Studio 2012**. Uruchomi to zintegrowane środowisko programistyczne Visual Studio.
2. Przez chwilę będzie wyświetlany ekran ładowania środowiska programistycznego, a następnie pojawi się strona **Start** z trzema głównymi opcjami:
   - **Connect to Team foundation server** (Połącz z serwerem Team Foundation)
   - **New Project** (Nowy projekt)
   - **Open Project** (Otwórz projekt)
3. Aby utworzyć nowy projekt, skorzystaj albo z tego łącza **New Project** (Nowy projekt), albo przejdź do menu **File** | **New Project** (Plik | Nowy projekt). To spowoduje pojawienie się okna dialogowego **New Project** (Nowy projekt). W tym oknie dialogowym dostępnych jest wiele opcji w zależności od pakietów aktualnie zainstalowanych wraz ze środowiskiem programistycznym. Z lewej strony okna dialogowego widać drzewo z różnymi elementami zainstalowanymi w środowisku programistycznym. W drzewie tym widać listę zainstalowanych szablonów. Gdy jeden z elementów po lewej stronie zostanie zaznaczony, odpowiednie szablony związane z tą grupą zostaną wymienione w środkowej części okna dialogowego (oznaczonej jako **2** na następnym zrzucie ekranu).

4. Wybierz odpowiednią nazwę dla swojego projektu w polu **Name:** (Nazwa) oznaczonym jako **3** na zrzucie ekranu).

5. Możesz zaznaczyć pole wyboru **Create directory for the solution** (utwórz katalog dla rozwiązania), oznaczone jako **4** na poniższym zrzucie ekranu, aby w wybranej lokalizacji został utworzony nowy folder o właśnie określonej nazwie, zamiast przechowywania tworzonych plików bezpośrednio w podanej lokalizacji.

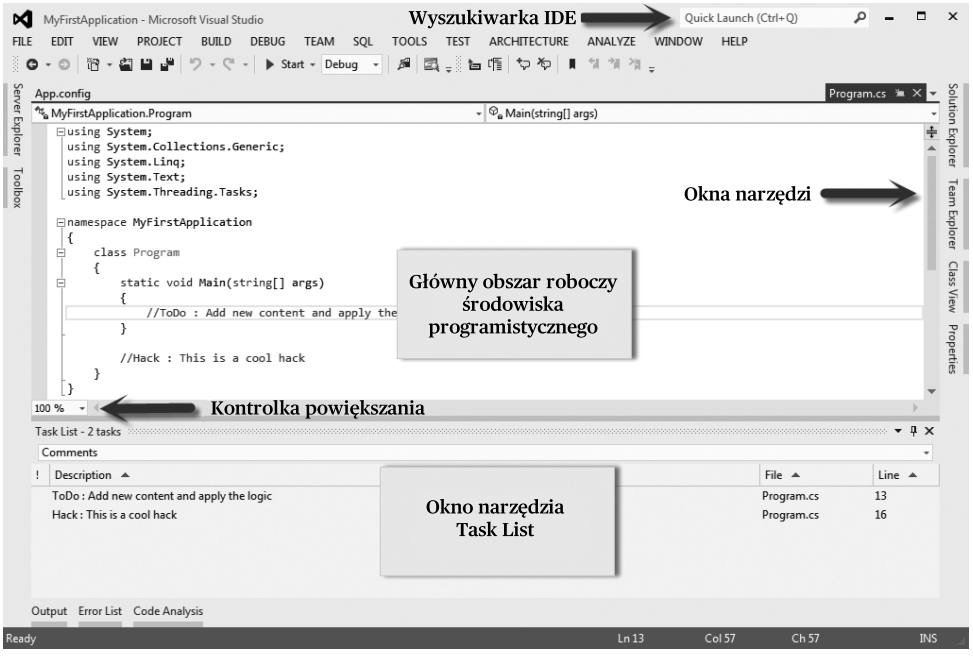


6. Wybierz **Visual C#** w okienku po lewej stronie, a **Console Application** w okienku środkowym; zostawiając domyślną nazwę kliknij **OK**, jak pokazano powyżej. Jeśli wszystko pójdzie dobrze, otwarte zostanie środowisko programistyczne wyglądające jak na zrzucie ekranu na stronie następnej.

7. Na tym zrzucie ekranu zaznaczyliśmy kilka elementów środowiska programistycznego, które wymagają specjalnej uwagi. Są one następujące:
   - Pierwszym elementem jest **Wyszukiwarka**, która jest po prostu pustym polem tekstowym służącym do wyszukiwania elementów środowiska programistycznego.
   - **Okna narzędzi** są umieszczone po lewej stronie, po prawej stronie i u dołu ekranu. Gdy okno narzędziowe zostanie otwarte, jak pokazano w przypadku **Okna narzędzia Task List** (Lista zadań) u dołu, pojawi się ono doczepione do brzegu ekranu, a gdy będzie schowane, odwołanie do niego będzie widoczne

w pasku bocznym środowiska programistycznego, jak pokazano po lewej i po prawej stronie okna.

- Główny obszar roboczy środowiska programistycznego zajmuje środkową część okna. Stanowi on podstawową część środowiska programistycznego, a programista aplikacji będzie w nim głównie pisać kod.
- Wewnątrz środowiska programistycznego dostępna jest też specjalna **Kontrolka powiększania**, która pomaga nam powiększać i pomniejszać zawartość edytora.



8. Na koniec możesz zacząć pisać swój kod w głównym obszarze roboczym środowiska programistycznego lub zacząć badać inne opcje dostępne w samym środowisku programistycznym.

Jak to działa…

Kilka rzeczy wymaga uwagi po otwarciu środowiska programistycznego Visual Studio. Visual Studio jest procesem uruchamianym przy wykorzystaniu pliku wykonywalnego o nazwie `devenv`, co oznacza **Developer's Environment** (środowisko programisty). Możemy albo podwójnie kliknąć ikonę Visual Studio w menu **Start** (co robi większość osób), albo przejść do menu **Start**, a następnie wyszukać plik `devenv`, żeby uruchomić

zintegrowane środowisko programistyczne. Środowisko programistyczne jest zwykle wywoływane w trybie uprawnień domyślnych. Czasami ważne jest otwarcie środowiska programistycznego jako **Administrator**, żeby móc korzystać z administracyjnych funkcji środowiska. Żeby zmienić zachowanie domyślne, możemy kliknąć prawym przyciskiem myszy skrót do programu i wybrać opcję **Run as Administrator** (Uruchom jako administrator). Możemy też ustawić na stałe uruchamianie środowiska programistycznego jako administrator korzystając z menu **Properties** (Właściwości).

Po wyświetleniu przez chwilę ekranu uruchamiania Visual Studio, pierwszą rzeczą, jaką zobaczymy, jest strona **Start**. Można przejść do opcji **File | New Project** (Plik | Nowy projekt), aby otworzyć okno dialogowe **New Project** (Nowy projekt). Jak pokazano na pierwszym zrzucie ekranu, po lewej stronie okna (oznaczone jako **1**) widać drzewo ze wszystkimi zainstalowanymi typami projektów pogrupowane w kilku panelach.

Jeśli nie możemy znaleźć odpowiedniego szablonu, możemy skorzystać ze znajdującej się w prawym górnym rogu okna dialogowego opcji **Search Installed Templates** (Wyszukaj zainstalowane szablony), aby wyszukać dowolny szablon według jego nazwy.

Ponieważ więcej niż jedna wersja platformy .NET może współistnieć na tym samym komputerze, okno dialogowe **New Project** (Nowy projekt) jest na tyle sprytne, że pozwala nam wybrać żądaną wersję platformy, której chcemy użyć podczas tworzenia aplikacji. Domyślnie wyświetlana jest wersja platformy .NET 4.0 jako domyślna platforma dla projektu, ale możemy ją zmienić wybierając inną wersję z listy rozwijanej. Całe środowisko się zmieni dając nam tylko te opcje, które są dostępne w przypadku aktualnie wybranej wersji.

Wybraliśmy opcję **Visual C#** w lewym drzewie i zaznaczyliśmy **Console Application** (Aplikacja konsolowa) w środkowym okienku jako szablon projektu. Po wybraniu szablonu jego opis jest wyświetlany po prawej stronie ekranu. Podaje nam on krótkie informacje, czym jest szablon **Console Application** (Aplikacja konsolowa) i co można dzięki niemu zrobić.

U dołu mamy możliwość nazwania projektu i rozwiązania, a także mamy opcję wyboru lokalizacji, w której zostanie utworzony projekt (oznaczona jako **3**). Możemy wybrać swoją własną ścieżkę do folderu do przechowywania plików tworzonych wewnątrz projektu podając odpowiedną ścieżkę w systemie plików.

Dostępne są też dwa pola opcji. Jednym z nich jest pole **Create directory for solution** (Utwórz katalog dla rozwiązania), którego zaznaczenie (domyślnie jest zaznaczone) utworzy katalog wewnątrz wybranej ścieżki i w nim umieści pliki projektu. W przeciwnym razie pliki projektu będą tworzone bezpośrednio w wybranym folderze. Dobrym zwyczajem jest pozostawianie tego pola zaznaczonego.

Na koniec możemy kliknąć **OK**, aby utworzyć projekt z domyślnymi plikami.

Gdy projekt zostanie utworzony, ekran środowiska programistycznego będzie wyglądał jak na zrzucie ekranu w kroku 6. Podzielimy teraz całe zintegrowane środowisko programistyczne na pokazane tam części i będziemy badać to środowisko w kolejnych przepisach.

Usuniemy kod znajdujący się wewnątrz metody `Main` otwierając klasę `Program` i wklejając następujący kod pomiędzy nawiasy klamrowe metody `Main`:

```
string content = "This is the test string to demonstrate Visual Studio IDE
features. ";
string content2 = "This is another string content";
Debug.Assert(content.Equals(content2), "The contents of the two strings are not
same");
Console.WriteLine("Thanks!");
Console.ReadKey(true);
```

Pobieranie kodu przykładowego

Pliki z kodem przykładowym dla wszystkich zakupionych książek wydawnictwa Packt można pobrać ze swojego konta pod adresem *http://www.packtpub.com*. Jeśli książka została zakupiona gdzie indziej, można odwiedzić adres *http://www.packtpub.com/support*, aby się zarejestrować i otrzymać pliki pocztą elektroniczną.

Po wklejeniu tego kodu możemy nacisnąć *F5* na klawiaturze. Zobaczymy okno **Konsoli** wyświetlające komunikat. Możemy nacisnąć klawisz *Enter*, aby zamknąć **Konsolę**.

Więcej…

Istnieje wiele innych opcji, w które wyposażony jest pakiet Visual Studio. Oprócz normalnego otwierania Visual Studio, możemy też skorzystać z pewnych opcji specjalnych przy uruchamianiu pakietu. Przyjrzyjmy się dokładniej innym opcjom, które mogą być przydatne.

Przełączniki polecenia uruchamiającego Visual Studio

Visual Studio będąc normalnym plikiem wykonywalnym działającym w systemie Windows udostępnia też kilka przełączników, które mogą być używane podczas otwierania środowiska programistycznego. Aby uruchomić Visual Studio z tymi przełącznikami, musimy albo skorzystać z wiersza polecenia, albo użyć opcji **Run** (Uruchom), żeby dodać przełączniki do środowiska programistycznego. Aby skorzystać z wiersza

polecenia, możemy po prostu przejść do lokalizacji `%systemdrive%\Program Files\Microsoft Visual Studio 11\VC` i wpisać `devenv /?`, żeby uzyskać listę wszystkich przełączników polecenia dostępnych dla zintegrowanego środowiska programistycznego. Na przykład `devenv /resetsettings`.

Ten przełącznik polecenia wyzeruje wszystkie ustawienia użytkownika, które zostały zastosowane w zintegrowanym środowisku programistycznym. Podczas zerowania ustawień można też podać plik `vssettings`, który zostanie użyty do nadpisania ustawień aktualnie zdefiniowanych w środowisku programistycznym. Podobnie możemy skorzystać z polecenia `devenv /resetSkipPkgs`.

To polecenie wyzeruje ładowanie wszystkich znaczników użytkownika skojarzonych z pakietami, które muszą być załadowane, aby wszystko działało gładko w Visual Studio. Czasami, jeśli środowisko programistyczne ulegnie uszkodzeniu lub znacznie wydłuży się czas jego ładowania, możemy też włączyć tryb ładowania diagnostycznego korzystając z przełącznika `devenv /SafeMode`.

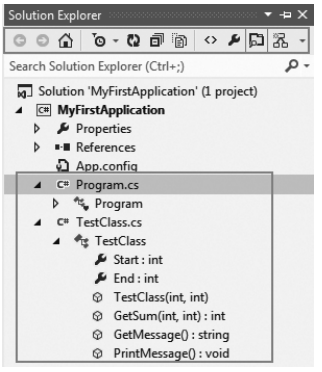
Możemy też spróbować uruchomić projekt w ogóle bez uruchamiania środowiska programistycznego. To znaczy w sytuacji, gdy musimy jedynie otworzyć środowisko programistyczne, uruchomić projekt i wyjść. Najlepszą opcją w tym przypadku jest użycie przełącznika polecenia `devenv /runexit "[ścieżka do rozwiązania/projektu]"`.

Tekst `[ścieżka do rozwiązania/projektu]` należy zastąpić faktyczną ścieżką, w której znajduje się plik `rozwiązania` (z rozszerzeniem `.sln`) lub pliki projektów (pliki `.csproj/.vbproj`).

Aby zobaczyć wszystkie przełączniki polecenia obsługiwane przez plik wykonywalny, możemy spróbować uruchomić polecenie `devenv /?` w wierszu polecenia.

Praca z narzędziami Solution Explorer i Class View

Najważniejszą częścią zintegrowanego środowiska programistycznego, z której będziemy korzystać najczęściej, jest **Solution Explorer** (Eksplorator rozwiązania). **Solution Explorer**, który znajduje się po prawej stronie środowiska programistycznego, jest najszerzej wykorzystywanym narzędziem nawigacyjnym do poruszania się wśród plików i klas. Narzędzie to jest pokazane na zrzucie ekranu na stronie następnej.



Na powyższym zrzucie ekranu widać podstawową strukturę narzędzia **Solution Explorer** w momencie załadowania projektu do zintegrowanego środowiska programistycznego. Okno **Solution Explorer** wyświetla na początku plik `Solution` (rozwiązanie) i ładuje wszystkie projekty związane z tym rozwiązaniem w postaci drzewa poniżej.

Kolejnym węzłem pod plikiem `Solution` (rozwiązanie) jest zwykle plik projektu.

Żeby łatwiej było rozpoznać każdy z tych plików w środowisku programistycznym, obok nazwy wyświetlana jest odpowiednia ikona, a nazwa pliku projektu jest pogrubiona.

Jak to zrobić…

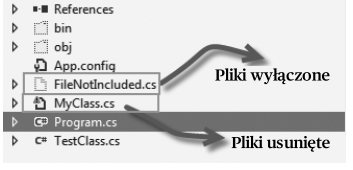
W tym przepisie zbadamy narzędzie **Solution Explorer**.

1. Z prawej strony środowiska programistycznego widać panel **Solution Explorer**, jak pokazano na poprzednim zrzucie ekranu. Jest to główny obszar ekranu, w którym można wykonywać działania na plikach i folderach związanych z projektem. Jeśli nie widać okienka **Solution Explorer**, wybierz z menu **View** | **Solution Explorer** (Widok | Eksplorator rozwiązania) lub naciśnij *Ctrl+W*, *S* na klawiaturze.
2. Gdy pojawi się okienko **Solution Explorer**, powinno ono zawierać drzewo ze wszystkimi plikami i folderami związanymi z projektem. Przełączaj każdy węzeł w tym drzewie, aby zobaczyć powiązane z nim informacje. Na rysunku widać, że **Solution Explorer** może przedstawiać też listę elementów składowych typu, który jest zapisany w danym pliku. Węzeł **TestClass** po otwarciu pokazuje drzewo ze wszystkimi swoimi elementami składowymi w kolejnych węzłach.
3. Nagłówek okienka **Solution Explorer** zawiera szereg przycisków. Przyciski te są poleceniami związanymi z aktualnie zaznaczonym węzłem drzewa.

Jak to działa…

Solution Explorer jest głównym oknem pokazującym zawartość całego rozwiązania załadowanego do zintegrowanego środowiska programistycznego. Daje nam zorganizowany w formie drzewa widok projektów i plików związanych z rozwiązaniem w celu łatwiejszej nawigacji. Najbardziej zewnętrznym węzłem w okienku **Solution Explorer** jest sam węzeł `Solution` (Rozwiązanie), pod nim znajdują się projekty, a następnie pliki/foldery. Plik `Solution` pozwala nam też ładować foldery bezpośrednio do rozwiązania, a nawet przechowywać dokumenty na pierwszym poziomie struktury rozwiązania. Projekt, który jest ustawiony jako startowy, jest oznaczony pogrubieniem.

U góry okna **Solution Explorer** zebranych jest kilka przycisków zwanych przyciskami paska narzędzi, a w zależności od typu pliku zaznaczonego w drzewie będą one dostępne lub wyłączone. Omówimy każdy z nich z osobna:



Drzewo rozwiązania w Visual Studio 2012 ładuje też całą strukturę klas do swoich węzłów. Wystarczy rozwinąć plik `.cs`, aby zobaczyć wszystkie zawarte w nim klasy i ich elementy składowe. Visual Studio ma też okno widoku klas, ale **Solution Explorer** jest wystarczająco inteligentny, żeby wymieniać wszystkie elementy widoku klas wewnątrz swojej hierarchii. Możemy otworzyć narzędzie Class View (Widok klas) korzystając z opcji menu **View** | **ClassView** (Widok | Widok klas) lub naciskając klawisze *Ctrl+W*, *C*, aby zobaczyć tylko widok klasy i jej elementów składowych.

Inną ważną cechą okna **Solution Explorer** jest śledzenie przez niego faktycznego istnienia plików rozwiązania w lokalizacji fizycznej. Podczas ładowania plików może czasem wyświetlać znaki ostrzegawcze, jeśli dany plik nie istnieje w lokalizacji fizycznej.

W powyższym przykładzie plik MyClass, choć jest plikiem `.cs`, to nie wyświetla zwykłej ikony, ale pokazuje znak ostrzegawczy wskazujący, że plik jest dodany do rozwiązania, ale fizycznie plik ten nie istnieje w miejscu docelowym.

Z drugiej strony niektóre pliki są wyświetlane w rozwiązaniu jako pliki puste (w powyższym przykładzie plik `FileNotIncluded.cs` albo foldery takie jak `bin/obj`). Te pliki, choć istnieją w systemie plików, nie są zawarte w pliku rozwiązania.

Każdy z plików wyświetla przycisk **Additional Information** (Informacje dodatkowe) z prawej strony węzła drzewa w rozwiązaniu. Przycisk ten daje nam dodatkowe informacje związane z danym plikiem. Na przykład, jeśli klikniemy przycisk odpowiadający

plikowi `.cs`, to wywołane zostanie menu **Contains** (Zawiera). Udostępni to widok klasy związanej z określonym plikiem w rozwiązaniu. To menu może być dość długie w zależności od elementów, które nie mogą być pokazane w ogólnych przyciskach paska narzędzi. Gdy rozwiązanie ładuje informacje dodatkowe, to dostępne są przyciski Forward (Dalej) i Back (Wstecz), dzięki którym możemy przechodzić pomiędzy widokami w rozwiązaniu.

Więcej…

Oprócz podstawowych opcji narzędzia **Solution Explorer**, istnieje wiele innych ulepszeń tego narzędzia, które zwiększają wydajność i poprawiają interfejs użytkownika zintegrowanego środowiska programistycznego. Zbadajmy je wszystkie po kolei.

Podgląd obrazów w narzędziu Solution Explorer

Solution Explorer pokazuje podgląd obrazu wskazanego myszą bez faktycznego otwierania tego obrazu. Jest to nowe rozszerzenie narzędzia **Solution Explorer**, które nie było dostępne w żadnej z poprzednich wersji środowiska programistycznego Visual Studio.

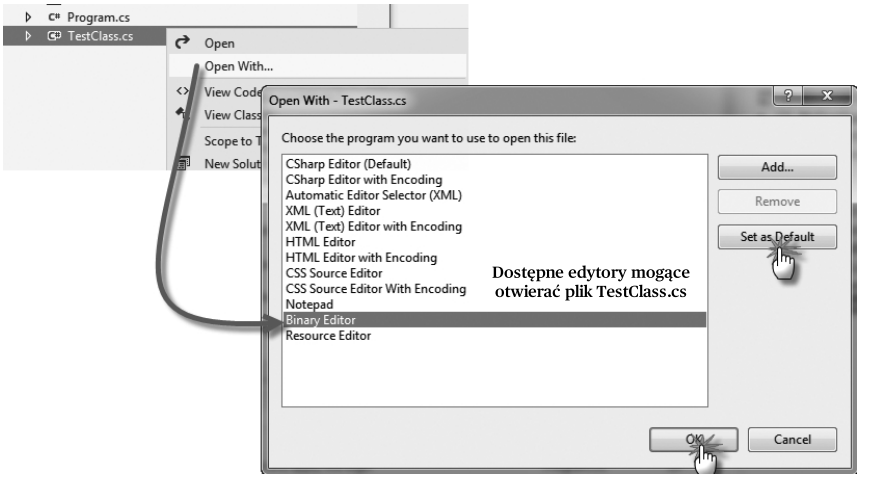
Możemy dodać obraz do rozwiązania klikając projekt prawym przyciskiem myszy i wybierając z menu opcję **Add** | **Add Existing Item** (Dodaj | Dodaj istniejący element), a następnie zaznaczając obraz w systemie plików.

Jeśli zrobimy to poprawnie, obraz zostanie załadowany do drzewa, jak pokazano na zrzucie ekranu. Wystarczy umieścić kursor myszy nad obrazem, a zobaczymy niewielki podgląd obrazu, jak pokazano na poprzednim zrzucie ekranu.

Różne edytory środowiska programistycznego

Visual Studio zawiera wiele edytorów instalowanych w tym pakiecie domyślnie. W zależności od typu pliku odpowiedni edytor jest ładowany do zintegrowanego środowiska programistycznego. Na przykład, jeśli podwójnie klikniemy obraz, zostanie

on otwarty w edytorze obrazów, jeśli z kolei wybierzemy plik `.cs`, zostanie on otwarty w edytorze C#.



Zamiast korzystać z podwójnego kliknięcia, możemy kliknąć prawym przyciskiem myszy dowolny plik wewnątrz węzła `Solution` i wybrać z menu **Open With...** (Otwórz przy pomocy...), aby otworzyć okno dialogowe wymieniające wszystkie dostępne edytory, które mogą załadować zaznaczony plik. Do edytorów dostępnych w środowisku programistycznym należą: C# Editor (Edytor języka C#), C# Editor with Encoding (Edytor języka C# z kodowaniem), Automatic Editor Selector (Automatyczne wybieranie edytora), XML Editor (Edytor XML), HTML Editor (Edytor HTML), Notepad (Notatnik), Binary Editor (Edytor binarny), Resource Editor (Edytor zasobów), itd.

Aby otworzyć plik CS w edytorze binarnym, możemy kliknąć plik CS prawym przyciskiem myszy, wybrać opcję **Open With...** (Otwórz przy pomocy...), zaznaczyć **Binary Editor** (Edytor binarny) i wybrać **OK**. Na poniższym zrzucie ekranu widać, że plik z kodem będzie wyglądał jak ciąg znaków binarnych:

Pierwsza kolumna pokazuje adres bajtów w pliku, druga pokazuje faktyczną zawartość bajtów danych, a trzecia kolumna zawiera tekstowy odpowiednik tych danych.

Możemy też wypróbować inne edytory na tej liście.

Praca z głównym obszarem roboczym środowiska programistycznego

Ten element reprezentuje główny obszar roboczy ekranu. Jest to najważniejsza część środowiska programistycznego Visual Studio, z której programiści będą głównie korzystać. Obszar roboczy zwykle wypełnia całą przestrzeń środowiska programistycznego lub większą część jego okna. Każde z okien, które mogą być ładowane wewnątrz środowiska programistycznego, możemy ukrywać, przeciągać poza obszar środowiska programistycznego, a nawet doczepiać w różnych miejscach okna głównego.

W głównym obszarze roboczym został już załadowany plik z klasą o nazwie `class1`. Edytor skojarzony z plikiem `.cs` jest ładowany na ekranie w celu wyświetlenia tego pliku. Istnieje wiele edytorów dostępnych w Visual Studio, a każdy z nich może być ładowany bezpośrednio w głównym obszarze roboczym. Nasze podstawowe działania programistyczne wykonujemy w tej części środowiska programistycznego.

Jak to zrobić…

Będziemy pracować w głównym obszarze roboczym zintegrowanego środowiska programistycznego wykonując następujące kroki:

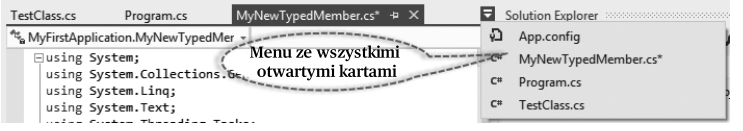
1. Zamknij wszystkie okna otwarte w środowisku programistycznym, aby elementem, który pozostanie na ekranie środowiska programistycznego, był główny obszar roboczy.
2. Otwórz **Solution Explorer** i podwójnie kliknij kilka plików. Każdy otwarty plik doda nową kartę do głównego obszaru roboczego. Nowe karty pojawiają się po lewej stronie stosu kart u góry głównego obszaru roboczego. Otwarcie dużej liczby plików w środowisku programistycznym spowoduje pojawienie się menu w prawym górnym rogu ekranu.
3. Przeciągnij kartę i umieść ją pomiędzy innymi, żeby zmienić jej położenie w stosie kart.
4. Zmień coś w wybranym pliku bez zapisywania zmian. Nagłówek karty wyświetli znak gwiazdki oznaczający niezapisane zmiany. W przypadku otwarcia pliku tylko do odczytu będzie tam wyświetlany znak kłódki.
5. Użyj przycisku **Toggle pin status** (Zmień stan przypięcia) na jednej z kart, aby przypiąć ją na stałe tak, aby nowo otwierane pliki nie zmieniały jej położenia. Jeśli wyświetlana jest karta **Preview** (Podgląd), to pojawi się specjalny przycisk **Promote** (Promuj), który przekształci ją w nowe okno. Obszar roboczy zawiera

edytor stanowiący większą część interfejsu środowiska programistycznego. W edytorze ładowana jest zawartość faktycznego pliku. Zmiany w edytorze są zaznaczane żółtym paskiem (gdy zmiany nie są zapisane) i zielonym paskiem (gdy zawartość jest zapisana).

6. Możesz powiększyć zawartość edytora korzystając z listy rozwijanej Zoom (Powiększ) w lewym dolnym rogu ekranu.

Jak to działa…

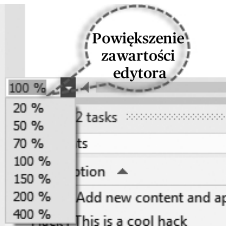
Obszar roboczy ładuje edytory na kartach. Gdy więc wybierzemy dwa węzły z okienka **Solution Explorer**, żeby otworzyć okno z kodem, to Visual Studio zachowa łącza do każdego z plików, które są otwierane w osobnych kartach. Nagłówek każdej karty zawiera kilka stałych zestawów elementów.



Na poprzednim zrzucie ekranu widać, że nagłówek karty zawierający nazwę pliku (`MyNewTypedMember.cs`) wyświetlanego w tej karcie zawiera znak `*`, gdy dany plik nie został zapisany. Ma on też przycisk do przełączania przypięcia karty (który powoduje przyczepienie karty do lewej strony stosu kart) oraz przycisk do zamykania pliku. Tytuł czasem zawiera dodatkowe informacje o jego stanie – na przykład, gdy plik jest zablokowany, wyświetlana jest ikona kłódki, a gdy obiekt jest ładowany z metadanych, to jego nazwa jest wyświetlana w nawiasach kwadratowych. W miarę otwierania kolejnych plików są one dołączane do stosu kart wypełniającego obszar roboczy. Gdy całe miejsce na nagłówki kart będzie zajęte, w prawym górnym rogu obszaru roboczego zostanie utworzone menu zawierające listę wszystkich otwartych plików, których nazwy nie zmieściły się na ekranie. Możemy wybierać z tego menu plik, który chcemy wyświetlić. Można też użyć kombinacji klawiszy *Ctrl+Tab* do przełączania się pomiędzy kartami, które są już załadowane w obszarze roboczym.

Pod tytułem karty, ponad głównym obszarem roboczym wyświetlane są dwie listy rozwijane. W lewej załadowana jest lista klas otwartych w środowisku programistycznym, a w prawej załadowane są wszystkie elementy składowe wybranej klasy. Te listy rozwijane pomagają łatwiej nawigować po zawartości pliku wymieniając po lewej stronie wszystkie klasy zawarte w bieżącym pliku. Lista po prawej stronie w sposób kontekstowy wymienia wszystkie elementy składowe znajdujące się w klasie, która została wybrana z listy po lewej stronie. Te dwie listy rozwijane są automatycznie aktualizowane, gdy nowy kod jest dodawany w edytorze.

Główny obszar roboczy jest otoczony dwoma paskami przewijania, które pozwalają dotrzeć do niewidocznych w danym momencie części dokumentu. Ponad pionowym paskiem narzędzi znajduje się specjalny przycisk do dzielenia okna na dwie części.



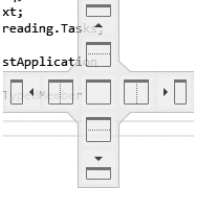
Poziomy pasek przewijania zawiera z kolei jeszcze jedną listę rozwijaną, która wyświetla aktualny stopień powiększenia zawartości edytora. Visual Studio pozwala teraz skalować zawartość edytora przy użyciu żądanego poziomu powiększenia. Skrótem do listy rozwijanej powiększania jest *Ctrl + kółko przewijania myszy*.

Więcej…

Po dokonaniu ogólnego przeglądu środowiska programistycznego Visual Studio 2012 przyjrzyjmy się dokładniej kilku innym funkcjom, które są dostępne wewnątrz środowiska programistycznego.

Przypinanie okien wewnątrz obszaru roboczego środowiska programistycznego

Otworzymy kilka okien w zintegrowanym środowisku programistycznym. Możemy zacząć od opcji dostępnych w menu **View** (Widok) środowiska programistycznego. Po otwarciu wielu okien narzędzi potrzebna nam będzie możliwość łatwego rozmieszczania tych okien. Środowisko programistyczne Visual Studio złożone jest z kilku obszarów dokowania zarówno wewnątrz, jak i na zewnątrz głównego obszaru środowiska programistycznego. Możemy przenieść okno przeciągając jego pasek tytułu, jak to robimy z normalnymi oknami. Podczas przeciągania panelu wewnątrz Visual Studio pojawia się zestaw kontrolek pokazanych na następnym zrzucie ekranu, które wskazują dostępne pozycje dokowania dla bieżącego okna.



Gdy najedziemy myszą na wybraną pozycję dokowania, pojawi się podpowiedź, gdzie znajdzie się okno, jeśli zostanie przyczepione w danej pozycji. Visual Studio automatycznie dopasuje stos elementów przypiętych w tej samej pozycji dokowania.

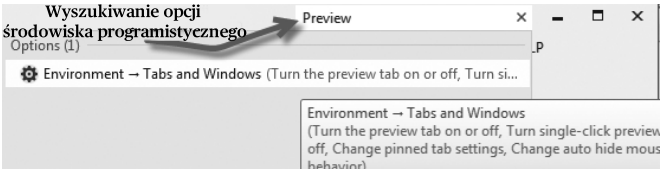
Całe środowisko programistyczne zawiera po obu stronach swojego obszaru roboczego stosy kart dla różnych okien. Jak już wspominałem, dla każdego z tych okien można włączyć funkcję automatycznego ukrywania. Gdy okno jest ukryte, wyświetlane jest tylko łącze do jego karty, natomiast gdy jest otwarte, to jest umieszczane w zbiorze kart.

Do innych elementów interfejsu środowiska programistycznego należą pasek menu, standardowy pasek narzędzi i pasek stanu. Każdy z tych elementów służy do wydawania poleceń środowisku programistycznemu.

Funkcje wyszukiwania opcji środowiska programistycznego

W prawym górnym rogu ekranu można znaleźć nowe pole wyszukiwania. Jest to pole wyszukiwania opcji środowiska programistycznego. Zintegrowane środowisko programistyczne Visual Studio jest ogromne. Wewnątrz tego środowiska są dostępne tysiące opcji, które możemy konfigurować. Czasami ciężko jest znaleźć konkretną opcję, która jest nam potrzebna. Funkcja wyszukiwania opcji środowiska programistycznego pomoże nam znaleźć potrzebną opcję.

Jak pokazano w poprzednim przepisie, jeśli zapomniałem na przykład, gdzie jest dostępna opcja dla karty **Preview File** (Podgląd pliku), mogę wpisać `Preview` w polu wyszukiwania.



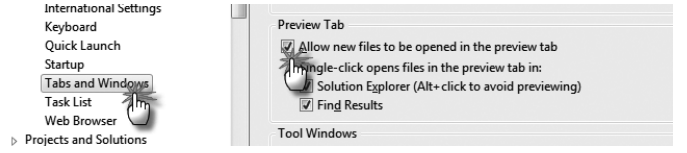
Jak pokazano na zrzucie ekranu, Visual Studio otworzy listę wszystkich opcji, gdzie dostępny jest tekst `Preview`. Po wybraniu pierwszej opcji przejdziemy do odpowiedniego miejsca interfejsu.

Podgląd plików w środowisku programistycznym

Środowisko programistyczne Visual Studio ma funkcję podglądu plików bez otwierania ich. W przypadku pojedynczego kliknięcia pliku zostanie on otwarty w głównym obszarze roboczym w trybie podglądu. Jak już widzieliśmy, karta podglądanego pliku zwykle pojawia się po prawej stronie stosu kart, aby wybranie innego pliku zastąpiło poprzednio podglądany. Jest to więc tymczasowy podgląd pliku. Plik można

wypromować (otworzyć w normalnym trybie) albo pracując bezpośrednio wewnątrz podglądu, albo korzystając ze specjalnego przycisku na pasku tytułu karty podglądu.

Zwykle opcja ta jest domyślnie wyłączona. Możemy przejść do menu **Tools** | **Options** (Narzędzia | Opcje), a następnie w lewym oknie otworzyć w drzewie kategorię **Environment** | **Tabs and Windows** (Środowisko | Karty i okna) i zaznaczyć opcję **Allow new files to be opened in preview Tab** (Zezwól na otwieranie nowych plików w karcie podglądu).



Opcja **Preview Tab** (Karta podglądu) jest zwykle przydatna, gdy pracujemy z dużą liczbą plików załadowanych w środowisku programistycznym.

Nawigowanie w obrębie kodu wewnątrz środowiska programistycznego

Visual Studio zawiera wiele przydatnych opcji nawigowania po kodzie. Przede wszystkim wypróbujemy funkcję **Code Highlighting** (Podświetlanie kodu) w Visual Studio.

Jak to zrobić…

Aby wypróbować funkcję Code Highlighting (Podświetlanie kodu) w Visual Studio, wykonaj następujące kroki:

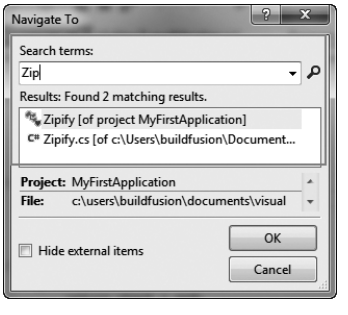
1. Podwójnie kliknij identyfikator w pliku C#, aby podświetlić każde wystąpienie tego samego identyfikatora w kodzie.
2. Naciskaj *Ctrl+Shift+strzałka w dół/górę*, aby przechodzić pomiędzy podświetlonymi wystąpieniami.
3. Naciśnij *Ctrl+,* (przecinek), aby otworzyć okno dialogowe **Navigate to** (Nawiguj do) wyświetlające listę wszystkich opcji nawigacyjnych dostępnych dla aktualnie zaznaczonego elementu (lub elementu znajdującego się w pozycji kursora).
4. Kliknij prawym przyciskiem myszy dowolną metodę lub typ i wybierz opcję **Go To Definition** (Przejdź do definicji), aby przejść do definicji tej metody lub typu.

5. Kliknij prawym przyciskiem myszy dowolny typ i wybierz opcję **Find All References** (Znajdź wszystkie wystąpienia), aby wypisać wszystkie wystąpienia tego typu w nowym oknie narzędziowym.
6. Możesz wybrać opcję **View Call Hierarchy** (Zobacz hierarchię wywołań) z menu kontekstowego wyświetlanego kliknięciem prawym przyciskiem myszy, aby wypisać wszystkie wywołania do i z zaznaczonego elementu.
7. Możesz zmienić nazwę elementu w środowisku programistycznym i nacisnąć *Ctrl+.* (kropka), aby otworzyć menu pozwalające zmienić nazwę wszystkich wystąpień tego elementu.

Jak to działa…

Nawigowanie pomiędzy plikami lub typami jest ważnym elementem pracy nad dowolnym projektem. Środowisko programistyczne Visual Studio daje nam narzędzia i funkcje pomagające w łatwym nawigowaniu pomiędzy różnymi miejscami.

Możemy zaznaczyć dowolny identyfikator w dokumencie projektu klikając go podwójnie, co spowoduje podświetlenie wszystkich wystąpień tego samego kodu w pliku. Możemy naciskać *Ctrl+Shift+strzałka w dół/górę*, aby nawigować pomiędzy podświetlonymi odwołaniami:

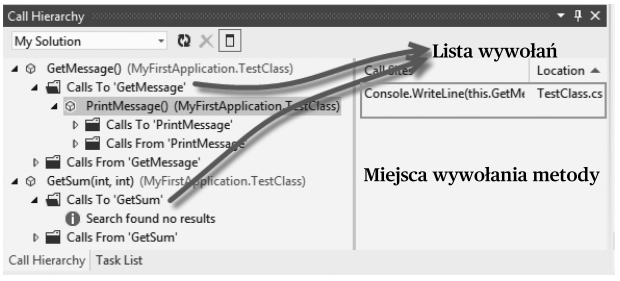


W celu uzyskania zaawansowanych opcji nawigowania po kodzie możemy nacisnąć *Ctrl+,* (przecinek), żeby otworzyć nowe okno o nazwie **Navigate To** (Nawiguj do), które szybko wyszukuje elementy składowe oraz pliki i wypisuje je na liście.

Okno dialogowe **Navigate To** (Nawiguj do) pokazuje też plik, z którego okno jest wywoływane, a także nazwę projektu.

Innym ważnym narzędziem do nawigowania po kodzie jest okno **Call Hierarchy** (Hierarchia wywołań). Możemy otworzyć to okno korzystając z kliknięcia prawym przyciskiem myszy i wybrania opcji **View Call Hierarchy** (Zobacz hierarchię wywołań). Okno **Call Hierarchy** (Hierarchia wywołań) pokazuje nam wszystkie odwołania wykorzystujące dany kod. Możemy też zobaczyć przeciążenia danej metody. Prawa

strona okna wymienia też wszystkie wywołania podając dokładny numer wiersza i nazwę pliku, gdzie znajduje się dane wywołanie.



Więcej…

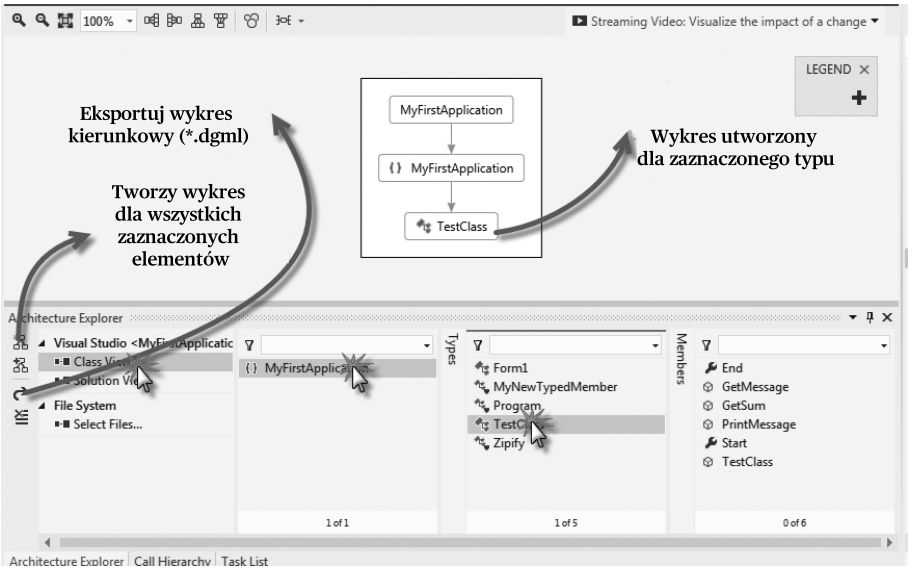
Oprócz podstawowych narzędzi nawigacyjnych, w środowisku programistycznym dostępne jest też kilka opcji zaawansowanych pomagających w szybkim i skutecznym nawigowaniu po interfejsie użytkownika. Przyjrzymy się teraz kilku innym dostępnym opcjom.

Architecture Explorer (Eksplorator architektury)

Jednym z najciekawszych nowych dodatków do Visual Studio jest **Architecture Explorer** (Eksplorator architektury). Narzędzie to pomaga nam w łatwym nawigowaniu po aktywach danego rozwiązania. Przejdźmy do opcji menu **View | Architecture Explorer** (Widok | Eksplorator architektury), aby wyświetlić okno podobne do pokazanego na kolejnym zrzucie ekranu.

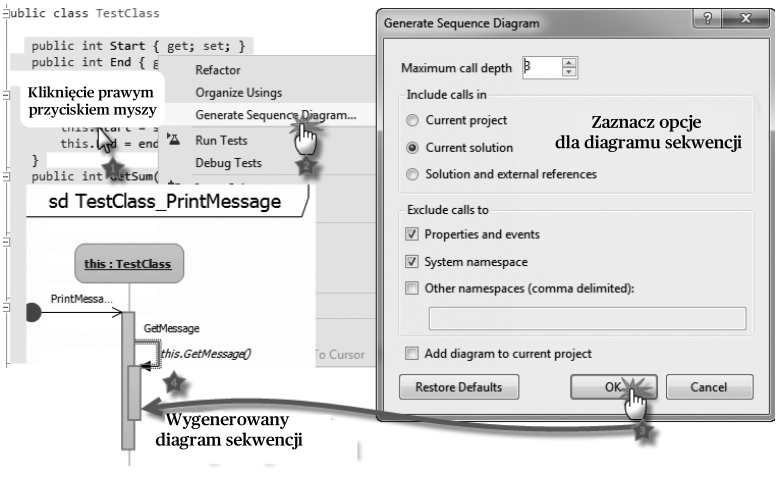
W oknie **Architecture Explorer** (Eksplorator architektury) możemy przeglądać i nawigować po zawartości dowolnej klasy, przestrzeni nazw lub metody. Visual Studio ma też funkcję eksportu okna **Architecture Explorer** (Eksplorator architektury) do dokumentu graficznego. Możemy wybrać elementy, które chcemy pokazać na rysunku i nacisnąć przycisk **Create New Graph Document** (Utwórz nowy dokument graficzny) w lewym górnym rogu okna **Architecture Explorer** (Eksplorator architektury). Pojawi się wykres przydatny do analizowania elementów kodu, wyszukiwania odwołań cyklicznych, węzłów, do których nie ma żadnych odwołań, itd. oraz do utworzenia obrazowego przedstawienia całej biblioteki. Możemy nawet wyeksportować dokument z wykresem korzystając z przycisku **Export Directed graph** (Eksportuj wykres kierunkowy) w eksploratorze.

Wykres z okna **Architecture Explorer** (Eksplorator architektury) można też wyeksportować do dokumentu XPS do wykorzystania w przyszłości.



Diagramy sekwencji

Możemy generować diagramy sekwencji bezpośrednio z poziomu środowiska programistycznego Visual Studio klikając prawym przyciskiem myszy jakąś metodę i wybierając opcje **Generate Sequence Diagram** (Generuj diagram sekwencji). Ta opcja jest dostępna tylko w wersji Ultimate pakietu Visual Studio. **Diagram sekwencji** przedstawi całą zawartość metody w postaci diagramu.



W tej metodzie widać, że użyłem kilku klas i obiektów, które zostały pokazane na diagramie.

Lista zadań

Visual Studio może służyć do śledzenia listy zadań związanych z projektem. Możemy użyć opcji **Task List** (Lista zadań) z menu **View** (Widok), aby wypisać wszystkie zaległe zadania w danym projekcie.

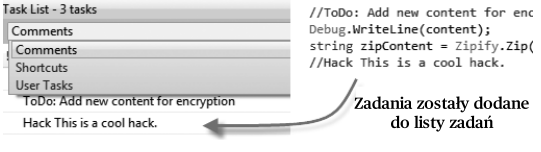
Istnieją różne opcje, które możemy wykorzystać do tworzenia zadań w projekcie.

- Gdy dodamy w kodzie komentarz z identyfikatorem rozpoznawanym przez okno **Task List** (Lista zadań), to nasze zadanie będzie wymienione w części **Comments** (Komentarze). Na przykład:

```
// ToDo To jest zadanie zaległe albo
// Hack To trzeba zmienić
```

Zadanie **ToDo** (Do zrobienia) będzie wymienione w oknie narzędziowym **Task List** (Lista zadań).

- Możemy też tworzyć zadania bezpośrednio wewnątrz okna zadań. W tym celu należy wybrać opcję **User Tasks** (Zadania użytkownika) i kliknąć przycisk znajdujący się zaraz obok tej listy rozwijanej. Możemy też określić priorytet zadania.



Zadania pochodzące z komentarzy można podwójnie kliknąć, aby przejść do odpowiedniego wiersza kodu z komentarzem.

Menu Bookmark (Zakładka)

Innym ważnym oknem do zarządzania kodem jest okno **Bookmarks** (Zakładki). Z zakładek też możemy korzystać do nawigowania po kodzie. Aby dodać zakładkę, możemy przejść do wiersza, w którym chcemy zastosować zakładkę i wybrać opcję **Toggle Bookmark** (Przełącz zakładkę) z menu **Edit** | **Bookmarks** (Edycja | Zakładki) lub nacisnąć *Ctrl+B*, *T*. Symbol zakładki pojawi się obok tego wiersza. Możemy przechodzić do następnej lub poprzedniej zakładki ustawionej w kodzie korzystając odpowiednio z kombinacji klawiszy *Ctrl+B*, *N* i *Ctrl+B*, *P*. Możemy wyczyścić wszystkie zakładki albo korzystając z menu, albo po prostu naciskając *Ctrl+B*, *C* na klawiaturze. Możemy też otworzyć okno narzędziowe **Bookmarks** (Zakładki), aby łatwiej nawigować pomiędzy zakładkami.

Okno narzędziowe **Bookmarks** (Zakładki) można otworzyć korzystając z kombinacji klawiszy *Ctrl+W, B* albo wybierając opcję menu **View | Other Windows | Bookmarks** (Widok | Inne okna | Zakładki). W oknie tym możemy manipulować zakładkami.

Okno Code Definition (Definicja kodu)

Jest to obecny w środowisku programistycznym edytor tylko do odczytu, który wyświetla definicje typów i metod, podczas gdy użytkownik nawiguje po kodzie w edytorze. Gdy ustawimy kursor myszy nad elementem kodu lub zmienimy zaznaczenie w oknie Class View (Widok klas), Object Browser (Przeglądarka obiektów) albo Call Browser (Przeglądarka wywołań), zawartość okna **Code Definition** (Definicja kodu) automatycznie zostanie zaktualizowana przez faktyczny kod pochodzący z aplikacji albo zawartość metadanych dla zaznaczonego typu.

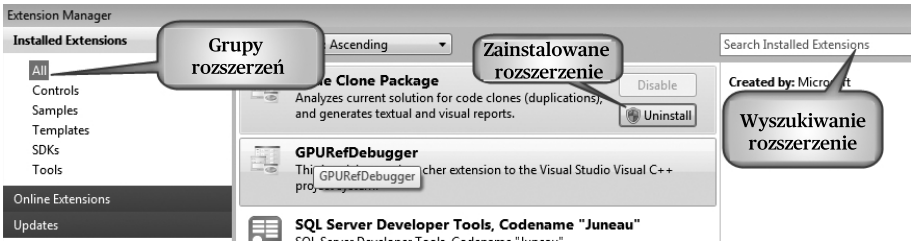
Aby otworzyć okno **Code Definition** (Definicja kodu), należy skorzystać z opcji menu **View | Code Definition** (Definicja kodu).

Okno **Code Definition** (Definicja kodu) nie tylko wyświetla definicję kodu podczas nawigowania, ale pozwala też nam kopiować kod, edytować definicję dzięki opcji **Edit Definition** (Edytuj definicję), wstawiać punkty przerwań, itd. Okno **Code Definition** (Definicja kodu) jest bardzo przydatne w pewnych przypadkach podczas pracy z dużymi aplikacjami.

Extension Manager (Menedżer rozszerzeń)

Visual Studio obsługuje rozszerzenia. Wiele składników środowiska programistycznego Visual Studio można obecnie rozszerzać. **Extension Manager** (Menedżer rozszerzeń) jest specjalnym obszarem, który pozwala nam przeglądać, obsługiwać lub odinstalowywać wszelkie rozszerzenia związane z Visual Studio.

Korzystając z opcji menu **Tools | Extension Manager** (Narzędzia | Menedżer rozszerzeń) możemy otworzyć to okno. Jeśli wybierzemy opcję **Online Extension** (Rozszerzenie online) z lewej strony, to połączymy się z sieciową galerią rozszerzeń. Możemy wybrać rozszerzenie z listy, pobrać je i zainstalować.



Gdy rozszerzenie będzie zainstalowane, wyświetlana będzie dla niego opcja wyłączenia lub odinstalowania rozszerzenia.

Czym jest narzędzie MSBuild i jak z niego korzystać?

Microsoft Build Engine jest platformą upraszczającą proces budowania, gdy mamy wiele plików, które muszą zostać razem skompilowane. Proces budowania w Visual Studio wykorzystuje środowisko MSBuild do zapewniania w jednym miejscu przeźroczystego dla użytkownika trybu budowania wszystkich plików i folderów. Cały proces budowania wymaga pliku projektu, który jest plikiem opartym na XML zapewniającym podstawową strukturę biblioteki.

Visual Studio wykorzystuje plik projektu do utrzymywania wszystkich elementów zawartych w projekcie, a plik ten jest później przekazywany do interfejsu narzędzia MSBuild w celu wywołania procesu budowania. W przypadku zaawansowanych scenariuszy, gdy nie mamy dostępnego pakietu Visual Studio, ale musimy zbudować hierarchię struktury projektów, MSBuild można wywoływać ręcznie pisząc samemu plik projektu.

Zbadajmy każdą część pliku projektu.

- **Item** (Element): Reprezentuje elementy składające się na proces budowania. Są one zgrupowane razem w kolekcji definiowanej przez użytkownika.

```
<ItemGroup>
     <Compile Include = "Program.cs" />
     <Compile Include = "TestClass.cs" />
</ItemGroup>
```

- **Properties** (Właściwości): Przedstawia parę klucz/wartość dla wszystkich właściwości budowania.

```
<PropertyGroup>
     <BuildDir>Build</Build>
</PropertyGroup>
```

- **Task** (Zadanie): Są to zadania, które trzeba wykonać podczas działania procesu budowania.

```
<Target Name"MakeBuildDirectory">
  <MakeDir Directories="$(BuildDir)"/>
</Target>
```

- **Targets** (Cele): Stanowią punkty wejściowe dla procesu budowania. Cele są zgrupowane w poszczególnych procesach budowania.

```
<Target Name="Compile">
   <Csc Sources="@(Compile)" />
</Target>
```

Części te zapewniają cenne informacje na temat tego, jak należy zbudować projekt. Po udanym utworzeniu pliku projektu możemy skorzystać z niego do utworzenia pliku wykonywalnego:

```
MSBuild.exe TestApplication.proj /property:Configuration=Release
```

W ten sposób projekt zostanie zbudowany w trybie produkcyjnym.

Debugowanie aplikacji

Następnym krokiem po udanym utworzeniu aplikacji jest uruchomienie aplikacji z poziomu środowiska programistycznego i jej debugowanie. Debugowanie aplikacji w środowisku programistycznym Visual Studio jest przyjemnością. Możemy wykonywać kod aplikacji krok po kroku, aby dokładnie zrozumieć wykonywany kod, a także identyfikować wszelkie problemy, które mogą pojawić się podczas wykonywania kodu.

Aby debugować aplikację, możemy albo kliknąć **Start** w pasku narzędzi, albo wybrać opcję **Debug | Start Debugging** (Debuguj | Rozpocznij debugowanie) z menu. Klawiszem skrótu do rozpoczęcia debugowania aktualnej aplikacji jest *F5*.

Gdy aplikacja działa wewnątrz środowiska programistycznego, każdy krok wykonywany przez aplikację jest monitorowany przez środowisko programistyczne, a wszelkie zmiany dokonywane w aplikacji bezpośrednio przechodzą przez silnik wykonawczy Visual Studio. Punkty przerwań są specjalnymi wskaźnikami w kodzie, które pozwalają na zatrzymanie wykonywania programu w określonym punkcie. Gdy aplikacja zatrzyma się na punkcie przerwania, program przestanie się wykonywać, a cały stan programu będzie odzwierciedlony w środowisku.

Aby przechodzić przez program wiersz po wierszu, możemy wybrać z menu opcję **Debug | Step over** (Debuguj | Przekrocz nad) lub nacisnąć *F10* w środowisku programistycznym. Natomiast, aby wejść do wnętrza definicji w kodzie, możemy wybrać

z menu opcję **Debug** | **Step into** (Debuguj | Wejdź do) lub nacisnąć *F11* w środowisku programistycznym.

Visual Studio zapewnia nowe środowisko do wykonywania programu podczas debugowania. Środowisko to obsługuje też wiele narzędzi pomagających w dokładnym zidentyfikowaniu, co się dzieje w określonym punkcie aplikacji.

Zobacz też

Dodatkowe narzędzia zwiększające wydajność pracy w Visual Studio 2012 i inne rozszerzenia środowiska programistycznego można znaleźć pod adresem *http://bit.ly/ProductivityPowerTools*.

Rozszerzanie szablonów Visual Studio

Pliki Visual Studio są tworzone z wykorzystaniem szablonów. Istnieje wiele szablonów dla Visual Studio dostarczanych wraz ze środowiskiem programistycznym i automatycznie kopiowanych podczas tworzenia plików i projektów w aplikacji. Pliki te są zwane **szablonami** w Visual Studio.

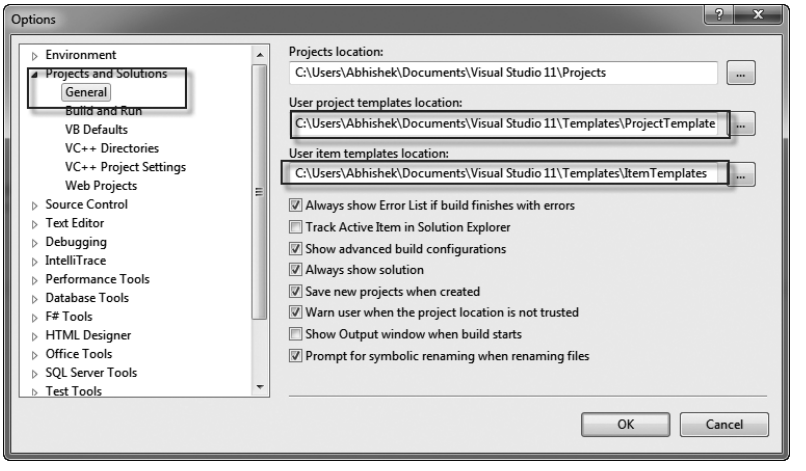
Możemy korzystać z Visual Studio do tworzenia niestandardowych szablonów mogących automatyzować tworzenie nowych plików korzystając z predefiniowanej struktury dla pliku projektu tak, że za każdym razem, gdy będziemy rozpoczynać nowy projekt albo dodawać nowy element do swojego projektu, większość tych elementów zostanie dla nas wygenerowana.

W tym przepisie omówimy sposoby korzystania ze wszystkich wstępnie zdefiniowanych szablonów, które są instalowane w środowisku programistycznym Visual Studio, a także powiemy, jak utworzyć swój własny szablon.

Istnieją dwa typy szablonów:

- **Szablony projektów**: Te pliki są związane z projektami i są używane, gdy nowy projekt jest tworzony lub dodawany do rozwiązania. Szablony, na podstawie których tworzymy projekty, są wymienione wewnątrz okna dialogowego **New Project** (Nowy projekt) w Visual Studio.
- **Szablony elementów**: Są to pliki elementów, które są wymienione w oknie dialogowym **New Item** (Nowy element). Gdy więc dodajemy nowy element do projektu, to wykorzystywane są **Szablony elementów** wymienione w tym oknie dialogowym.

Domyślna lokalizacja folderów dla szablonów projektów i elementów jest określona w oknie dialogowym **Options** (Opcje) pakietu Visual Studio, jak pokazano na poniższym zrzucie ekranu:



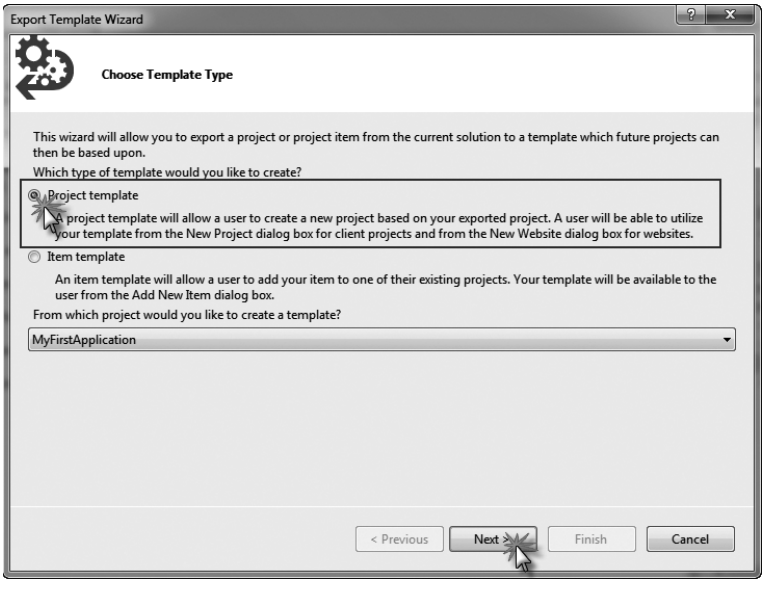
Na tym zrzucie ekranu widać, że szablony projektów i elementów znajdują się w podanych ścieżkach dostępu. Szablony są właściwie skompresowanymi archiwami, które możemy umieścić w tej lokalizacji, żeby były wymieniane na listach w oknach dialogowych dla nowych projektów i elementów.

Domyślne **Szablony projektów** i **Szablony elementów** znajdują się odpowiednio w folderach `Program Files\[Katalog instalacyjny Microsoft Visual Studio]\Common7\IDE\Project Templates` oraz `Program Files\[Katalog instalacyjny Microsoft Visual Studio]\Common7\IDE\ItemTemplates`.

Jak to zrobić…

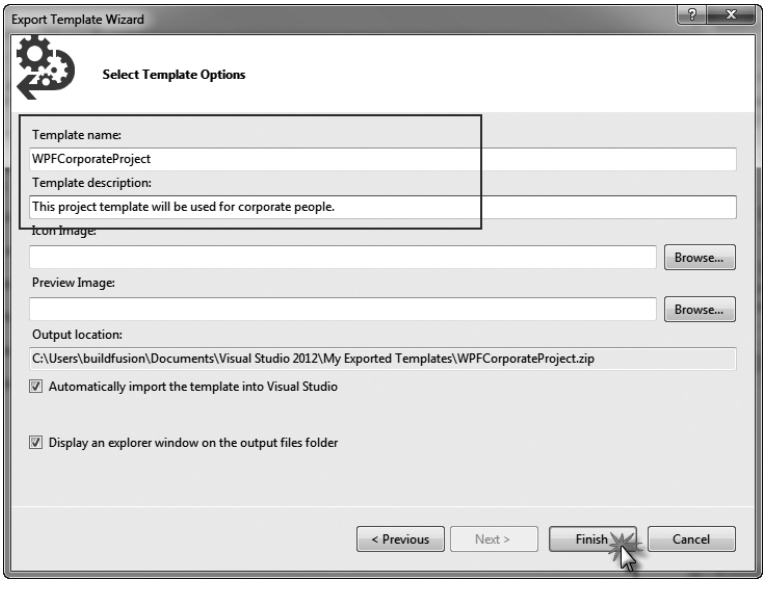
W tym przepisie utworzymy nowy szablon projektu i nowy szablon elementu oraz przyjrzymy się ich szczegółom.

1. Utwórz projekt niestandardowy pisząc kod, który będzie potrzebny, gdy ktoś użyje tego szablonu.
2. Dodaj odwołania do podzespołów, które będą potrzebne w ostatecznym projekcie.
3. Wybierz opcję **File** | **Export Template** (Plik | Eksportuj szablon). Otwarty zostanie nowy kreator, w którym można wybrać albo **Project Template** (Szablon projektu), albo **Item Template** (Szablon elementu). Wybierz opcję **Project Template** (Szablon projektu), tak aby cały projekt został wyeksportowany.

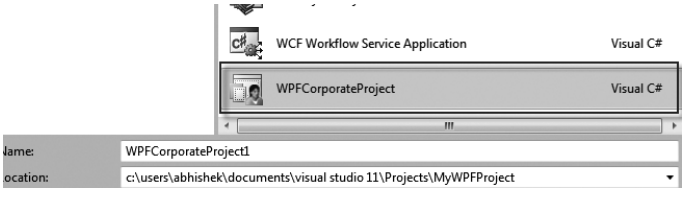


Jeśli w tym samym rozwiązaniu jest wiele projektów, to można wybrać dowolny z nich z listy rozwijanej u dołu ekranu.

4. Na następnym ekranie trzeba podać nazwę szablonu, która będzie wyświetlana w oknie dialogowym **New Project** (Nowy projekt) oraz opis szablonu. Ta informacja będzie widoczna w oknie dialogowym **New Project Template** (Nowy szablon projektu) w etykiecie podpowiedzi albo w osobnym okienku z opisem.
5. Okno **Template Options** (Opcje szablonu) pozwala też wybrać plik ikony oraz obraz podglądu dla szablonu. W naszym przypadku będą one puste.
6. Zwróć uwagę, że lokalizacja wyjściowa jest pokazana w polu tekstowym pod tymi opcjami i widać, że plik ZIP zostanie utworzony w podanej lokalizacji.
7. Gdy wszystko będzie gotowe, kliknij **Finish** (Zakończ). Utworzony zostanie plik `WPFCorporateProject.zip`. Plik ten jest właściwie eksportowany do lokalizacji `My Exported Template`. Żeby był widoczny na liście w oknie dialogowym **New Project** (Nowy projekt), trzeba skopiować ten plik do lokalizacji przeznaczonej na szablony projektów.



8. Wreszcie, gdy otworzymy okno dialogowe **New Project** (Nowy projekt), pojawi się w nim nasz plik **WPFCorporateProject**, jak pokazano na poniższym zrzucie ekranu:



9. Gdy wybierzemy typ projektu, nowy projekt automatycznie ustawi nazwę przestrzeni nazw, nazwy plików itp.

W ten sam sposób możemy wyeksportować **Szablon elementu** w rozwiązaniu. Główna różnica pomiędzy **Szablonem projektu** a **Szablonem elementu** jest taka, że szablon projektu ma za zadanie definiować cały projekt, natomiast szablon elementu ma za zadanie definiować pojedynczy element projektu.

Jak to działa…

Visual Studio definiuje zestaw reguł, które trzeba zastosować wobec zawartości pliku ZIP, gdy tworzymy jego nowe wystąpienie. Podczas eksportowania szablonu Visual Studio analizuje wszystkie pliki, które trzeba wyeksportować wraz z projektem, tak