

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTEL尼亚

FRAGMENTY KSIĄŻEK ONLINE

Visual Basic .Net. Szybki start

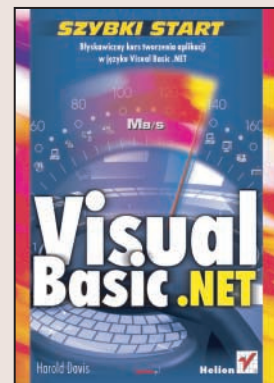
Autor: Harold Davis

Tłumaczenie: Krzysztof Jurczyk, Tomasz Walczak

ISBN: 83-7361-737-X

Tytuł oryginału: [Visual Basic .Net
for Windows Visual Quickstart Guide](#)

Format: B5, stron: 600



Kurs tworzenia aplikacji WWW w języku Visual Basic .NET

Visual Basic od dawna cieszy się uznaniem wśród informatyków. Według ostatnich badań, jest wykorzystywany przez ponad 5 milionów programistów z całego świata. Visual Basic jest niezwykle łatwy do opanowania, a jednocześnie posiada ogromne możliwości i elastyczność. Jego najnowsza wersja – Visual Basic .NET otwiera przed twórcami oprogramowania nowe horyzonty związane z platformą .NET – szybki i wydajny dostęp do baz danych, stosowanie języka XML oraz możliwość tworzenia aplikacji typu Web Services to tylko niektóre z nich. Język VB .NET jest potężnym, eleganckim i w pełni obiektywnym nowoczesnym językiem programowania, w którym jednak została zachowana prostota, która zawsze była cechą charakterystyczną języków z rodziny Basic.

„Visual Basic .NET. Szybki start” to książka dla tych, którzy chcą poznać możliwości najnowszej wersji Visual Basica bez konieczności przebijania się przez tysiące linii przykładowych kodów i setki stron opisów. Każde z omawianych zagadnień przedstawione jest w postaci sekwencji czynności opatrzonej ilustracjami. Nie znajdziesz tu rozważań teoretycznych – poznasz tajniki środowiska .NET, obiektów i programowania poprzez rozwiązywanie przykładowych zagadnień.

- Środowisko Visual Studio .NET
- Projektowanie aplikacji Web Services
- Klasy i obiekty
- Tworzenie formularzy Windows
- Okna komunikatów
- Praca z formularzami MDI
- Obsługa wyjątków i usuwanie błędów
- Stosowanie komponentów
- Dodawanie menu do aplikacji
- Bazy danych oraz technologie ADO.NET i XML
- Aplikacje internetowe ASP.NET

Poznaj język Visual Basic szybko i bez problemów.

Dołącz do ogromnej społeczności programistów VB.NET

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl



Spis treści

	Wprowadzenie	9
Rozdział 1.	Poznajemy Visual Studio .NET	15
	Strona startowa.....	16
	Konfigurowanie środowiska Visual Studio .NET.....	22
	Solucje, projekty i moduły	26
	Narzędzia środowiska Visual Studio.....	36
Rozdział 2.	Tworzenie usług Web Service	43
	Poznajemy usługi Web Service	44
	Tworzenie usługi Web Service.....	47
	Testowanie usług Web Service	52
	Dodawanie metod Web	53
	Testowanie metody Web.....	54
Rozdział 3.	Praca z klasami	55
	Poznajemy obiekty i klasy.....	56
	Dodawanie klasy do usługi Web Service	58
	Dodawanie klasy do aplikacji Windows	67
	Składowe klasy.....	78
	Zasięg	90
	Przeciążanie metody.....	92
	Konstruktory klasy	94
	Dziedziczenie	96
	Dziedziczenie wizualne.....	99
Rozdział 4.	Interfejsy klasy	101
	Abstrakcyjne klasy bazowe	102
	Interfejsy klasy	111
	Tworzenie własnego interfejsu klasy	118
Rozdział 5.	Formularze Windows	125
	Klasa Form	126
	Właściwości formularza	133
	Zmiana kształtu formularza.....	151

	Praca z egzemplarzami formularza	154
	Uruchamianie aplikacji z Sub Main	167
Rozdział 6.	Używanie usług Web Service	173
	Tworzenie interfejsu użytkownika	174
	Sprawdzanie poprawności danych wprowadzanych przez użytkownika.....	177
	Dodawanie odnośnika do usługi Web Service	180
	Używanie usługi.....	182
Rozdział 7.	Okna komunikatu i instrukcja If	187
	Klasa MessageBox	188
	Tworzenie interfejsu użytkownika	208
	Stosowanie instrukcji If.....	214
	Podłączanie formularza	215
Rozdział 8.	Praca z formularzami MDI	225
	Używanie formularza nadrzędnego	227
	Używanie formularzy potomnych	230
	Używanie tablic.....	237
	Klasy pojemnikowe Collection	245
Rozdział 9.	Wyjątki i usuwanie błędów	247
	Strukturalna obsługa wyjątków	248
	Używanie instrukcji Try	249
	Generowanie wyjątków.....	261
	Praca z obiektami Exception	264
	Używanie okna dialogowego Exceptions.....	265
	Poznajemy różne rodzaje błędów	267
	Korzystanie z metod obiektu Debug	270
	Używanie poleceń Step i Watch.....	273
Rozdział 10.	Kontrolki umożliwiające interakcję z użytkownikiem	277
	Dostosowanie paska narzędzi do własnych wymagań	278
	Polecenia menu Format.....	284
	Praca z grupami kontroltek.....	288
	Praca z kontrolką paska narzędzi ToolBar	294
	Użycie kontrolki ToolTip.....	303
	Obsługa zakładek za pomocą kontrolki TabControl	305
	Korzystanie z kontrolki TrackBar	311
	Zarządzanie listami za pomocą kontrolki ListBox.....	313

Rozdział 11.	Komponent Timer	331
	Zasada działania komponentu Timer	332
	Wyświetlanie bieżącego czasu	337
	Animacja duszka	342
Rozdział 12.	System menu	353
	Dodawanie menu	354
	Definiowanie nazw elementów menu	361
	Doskonalenie systemu menu	364
	Obsługa zdarzeń dotyczących menu	369
	Modyfikacja menu za pomocą kodu aplikacji	377
	Dodawanie menu za pomocą kodu aplikacji	385
	Implementacja menu w aplikacjach MDI	388
Rozdział 13.	Kontrolki Common Dialog	401
	Przykładowa aplikacja	403
	Okno dialogowe Czcionka	408
	Okno dialogowe Kolor	411
	Okna dialogowe File	414
	Korzystanie z obiektu FileStream	423
	Drukowanie	428
Rozdział 14.	Przeglądarka obiektów Object Browser	437
	Otwieranie przeglądarki obiektów	438
	Interfejs przeglądarki obiektów	442
	Pakiety	448
	Przestrzenie nazw	451
	Ważniejsze przestrzenie nazw .NET	458
Rozdział 15.	XML, zbiory danych i ADO.NET	461
	Praca z XML	463
	Praca z bazami danych	483
Rozdział 16.	Aplikacje internetowe ASP.NET	513
	Podstawy ASP.NET	515
	Warunki niezbędne do tworzenia aplikacji ASP.NET	518
	Tworzenie aplikacji internetowych ASP.NET	519
	Praca z kontrolkami	530
Dodatek A	Pomoc w środowisku Visual Studio	557
	Korzystanie z systemu pomocy online	558
	Korzystanie z pomocy dynamicznej	560

	Korzystanie z okna zawartości Contents.....	561
	Korzystanie z okna Index	563
	Korzystanie z okna Search	564
	Filtrowanie	566
Dodatek B	Słowa kluczowe Visual Basic .NET	569
Dodatek C	Typy i konwersja typów	571
	Typy Visual Basic .NET	572
	Opcja kompilatora Option Strict	574
	Podstawy silnego typowania	576
	Korzystanie z konwersji niejawnej.....	579
	Korzystanie z funkcji CType.....	580
	Korzystanie z dołączanych funkcji konwersji typów	581
	Stosowanie innych metod konwersji	582
	Skorowidz	585

Tworzenie usług Web Service

2

Teraz, kiedy wiesz już, jak poruszać się w środowisku Visual Studio .NET, pora, abyś stworzył pierwszy program Visual Studio .NET.

Jedną z najbardziej spektakularnych cech platformy .NET jest łatwe i szybkie tworzenie usług Web Service. Dlatego też pierwszym programem, jaki utworzysz, będzie opisana w tym rozdziale usługa *ASP.NET Web Services*.

Poznajemy usługi Web Service

Czym jest usługa Web Service? W najprostszej postaci usługa Web Service dostarcza metody, które mogą być wywołane przez sieć za pomocą standardowych protokołów, takich jak HTTP (ang. *Hypertext Transfer Protocol*) oraz XML (ang. *eXtensible Markup Language*). **Metoda** jest to funkcja lub procedura, która związana jest z danym obiektem. Innymi słowy, wywołanie metody powoduje, że wykonywane jest określone dla niej działanie.

Usługi Web Service są o tyle ciekawe, że pozwalają tworzyć programy, które są całkowicie międzyplatformowe. Programy te mogą być uruchamiane poprzez sieć internetową niezależnie od użytych do ich tworzenia narzędzi oraz bez względu na system, na jakim mają działać.

Inną charakterystyczną cechą usług Web Service jest to, że dostarczają pełnej informacji o wykonywanych działaniach. Oznacza to, że można sprawdzić, jakie są nazwy metod, których dana usługa dostarcza, jakie argumenty przyjmują poszczególne metody oraz jakie wartości zwracają. Jeśli więc wie się, że dana usługa Web Service istnieje, to można z niej korzystać w tworzonym programie

Celem, dla którego utworzono usługi Web Service, jest dostarczenie komponentów o specyficznych cechach. Komponenty te mogą się ze sobą komunikować niezależnie od języka programowania, w którym zostały napisane oraz bez względu na system operacyjny, na którym są uruchamiane. Metody tych komponentów mogą być używane przez inne programy. Aby użyć usługi Web Service, program wysyła żądanie za pomocą protokołu HTTP, i aby posłużyć się konkretną metodą, wywołuje HTTP GET, HTTP POST lub — najczęściej — używa protokołu *SOAP* (ang. *Simple Object Access Protocol*).

Dzięki platformie .NET programista nie musi wglębiać się w szczegóły kodu protokołu SOAP. Ponieważ jednak usługi Web Service korzystają z żądań protokołu SOAP, warto zapoznać się z istotą jego działania.

We wcześniejszym fragmencie znalazła się informacja, że usługi Web Service dostarczają uniwersalnego mechanizmu, który informuje o używanych metodach. W celu implementacji tego mechanizmu platforma .NET automatycznie tworzy dokument w języku WSDL (ang. *Web Services Description Language*), który może zostać odczytany przez aplikację lub człowieka. Dokument ten pozwala zrozumieć, jak działa dana usługa Web Service. Platforma .NET uwalnia programistę od obowiązku tworzenia dokumentów WSDL, warto jednak wiedzieć, że taki dokument istnieje i do czego służy.

Usługi Web Service korzystają także ze standardu UDDI (ang. *Universal Description, Discovery and Integration*). Standard ten jest czymś w rodzaju *Panoramy Firm* dla usług Web Service. Jeśli chcesz odnaleźć usługę Web Service lub jeśli chcesz, aby dana usługa mogła zostać odnaleziona, musisz skorzystać ze standardu UDDI.

W następnych paragrafach przedstawione zostaną podstawowe cechy trzech głównych komponentów usług Web Service — SOAP, WSDL oraz UDDI. Podane także zostaną niezbędne informacje, potrzebne do utworzenia zaawansowanej usługi Web Service za pomocą platformy .NET.

SOAP

Standard SOAP, kontrolowany przez konsorcjum W3C (ang. *World Wide Web Consortium*), jest oparty na standardzie XML. Składa się z trzech części:

- ◆ Otoczki (ang. *envelope*), która definiuje reguły opisywania wiadomości i ich przetwarzania.
- ◆ Zestawu reguł kodowania (ang. *encoding rules*) struktur danych definiowanych w aplikacjach.
- ◆ Konwencji (ang. *convention*) reprezentowania zdalnych wywołań procedur i ich odpowiedzi.

Jeśli myślisz, że ręczne tworzenie usług Web Service, które będą zgodne ze standardem SOAP, jest skomplikowane i żmudne, to masz całkowitą rację. Na szczęście w trakcie tworzenia usług Web Service w postaci projektu ASP.NET w środowisku Visual Studio nie trzeba wpisywać kodu. Jeśli jednak ktoś chce dowiedzieć się więcej o szczegółach specyfikacji standardu SOAP, może zapoznać się z nimi na stronie <http://www.w3.org/TR/SOAP/>.

WSDL

Standard WSDL używany jest do tworzenia dokumentów, które opisują metody obsługiwane przez usługę Web Service, argumenty przyjmowane przez te metody oraz zwracane przez nie wartości. Innymi słowy dokument WSDL przekazuje programowi lub programiście wszystkie niezbędne informacje, potrzebne do skorzystania z usługi.

Ponieważ dokumenty WSDL odczytywane są zwykle przez komputery, a nie przez ludzi, muszą być formułowane w sposób, który pozwoli na ich przetwarzanie przez oprogramowanie. Dlatego też dokumenty WSDL tworzone są w oparciu o język XML, który służy do opisu formatu innych dokumentów (w tym także i XML).

Przy tworzeniu usługi Web Service za pomocą projektu ASP.NET w środowisku Visual Studio nie trzeba się martwić o tworzenie dokumentu WSDL. Zajmuje się tym środowisko Visual Studio w momencie uruchamiania projektu ASP.NET z usługą Web Service. Więcej informacji o standardzie WSDL można znaleźć na stronie <http://www.w3.org/TR/WSDL>.

Jeśli będziesz chciał utworzyć program VB .NET, który używa usług Web Service, ale ich nie udostępnia, wtedy środowisko Visual Studio utworzy usługę proxy w oparciu o dokument WSDL dostarczony przez usługę Web Service. Klasa proxy zawiera kod, który pozwala na użycie metod usług Web Service w kodzie języka Visual Basic. Szczegółowy opis znajduje się w rozdziale 6. „Używanie usług Web Services”.

UDDI

Jeśli w lesie upadło drzewo, ale nikt tego nie widział, to czy rzeczywiście upadło? Ta sama zagadka mogłaby odnosić się do usług Web Service. W tym przypadku istnieje jednak standard UDDI, który przekazuje informacje o istniejących usługach.

Jeśli na przykład dysponujesz usługą Web Service, której mogą używać inne osoby, możesz skorzystać z UDDI, aby umieścić usługę na liście. Potencjalni użytkownicy usług mogą ją tam znaleźć. W podobny sposób i Ty możesz znaleźć potrzebną usługę.

W przypadku standardów SOAP i WSDL środowisko .NET sprawia, że wystarczy zrozumieć tylko ogólny sens ich działania. Z kolei standard UDDI wymaga działania programisty już na wczesnych etapach pracy.

Aby znaleźć usługę Web Service:

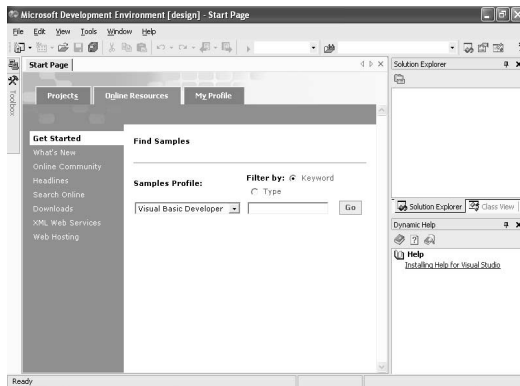
1. Na stronie startowej środowiska Visual Studio na zakładce *Online Resources* kliknij odnośnik *XML Web Services* (rysunek 2.1).
2. Pojawi się zakładka *Find a Service*.
3. Upewnij się, że zaznaczony jest przycisk opcji *UDDI Production Environment*.
4. Wybierz kategorię z listy rozwijanej *Category*, na przykład *Financial*.
5. W polu tekstowym *Search For* wpisz szukane pojęcie lub zostaw pole puste, jeśli chcesz zobaczyć wszystkie dostępne w danej kategorii usługi.

5. Kliknij przycisk *Go*.

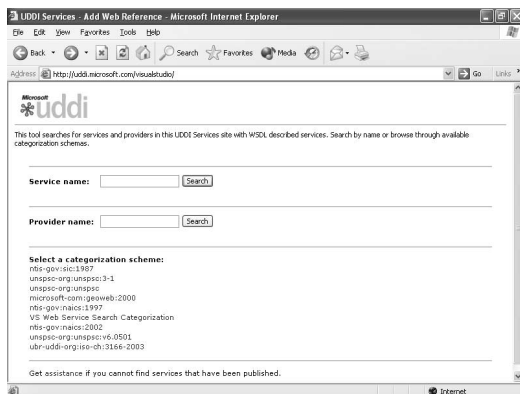
Zostanie wyświetlona lista dostępnych serwisów *Web Services* wraz z ich opisami.

Wskazówki

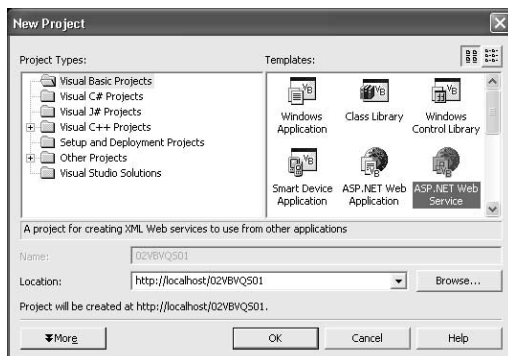
- Więcej informacji o stronie startowej środowiska Visual Studio znajduje się w rozdziale 1.
- Aby uzyskać dostęp do większej ilości opcji, które kontrolują szukanie usług, kliknij przycisk *Advanced*.
- Aby zarejestrować własną usługę Web Service za pomocą UDDI, wybierz zakładkę *Register a Service*, a następnie kliknij odnośnik *Register Your XML Web Service Today*.
- Innym sposobem wyszukiwania usługi Web Service jest użycie Internet Explorera i odwołanie się do strony <http://uddi.microsoft.com/visualstudio> (rysunek 2.2)



Rysunek 2.1. Strona XML Web Service służąca do odnajdywania istniejących usług Web Service, a także do rejestrowania nowych usług, aby inni użytkownicy mogli z nich korzystać



Rysunek 2.2. Strona <http://uddi.microsoft.com/visualstudio> umożliwiająca wyszukiwanie usług Web Service



Rysunek 2.3. Aby utworzyć nowy projekt *Web Service*, wybierz opcję *ASP.NET Web Service* z panelu *Templates*

Tworzenie usługi Web Service

Przy użyciu języka Visual Basic .NET niezwykle łatwo utworzyć usługę Web Service, więc zacznijmy od razu.

W rozdziale 1. zostało opisane, że programy opierają się na projektach VB .NET. Tak więc rodzaj tworzonego programu zależy całkowicie od rodzaju projektu, który wybierzesz spośród: *Windows Application*, *ASP.NET Web Application*, *ASP.NET Web Service* oraz innych.

Aby rozpocząć tworzenie usługi Web Service, należy utworzyć nowy projekt ASP.NET Web Service.

Aby utworzyć nowy projekt ASP.NET Web Service:

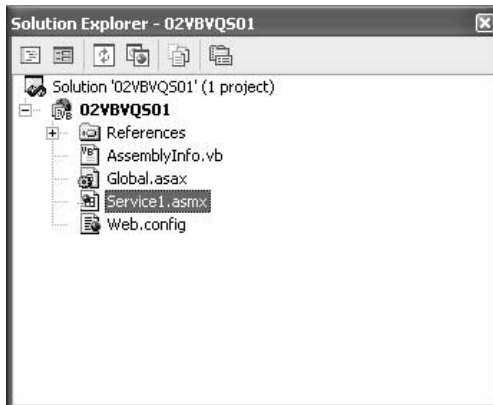
1. Otwórz okno dialogowe *New Project* (rysunek 2.3). Możesz to zrobić wybierając opcję *New Project* z menu *File* lub klikając przycisk *New Project* na stronie startowej.
2. Wybierz opcję *Visual Basic Projects* z panelu *Project Types*.
3. Wybierz jako typ projektu *ASP.NET Web Service* z panelu *Templates*.
4. W polu tekstowym *Location*, po adresie URL *http://localhost/*, wpisz nazwę projektu.
5. Kliknij przycisk *OK*.

Zostanie utworzony nowy projekt *ASP.NET Web Service*.

Wskazówki

- Adres URL (ang. *Uniform Resource Locator*) został uzupełniony w czwartym kroku powyższego zadania przy założeniu, że uruchamiasz i testujesz usługi Web Service na lokalnym serwerze IIS (ang. *Internet Information Services*). Jeśli korzystasz z odległego komputera macierzystego, wpisz odpowiedni adres URL w polu tekstowym *Location*.
- Jeśli w trakcie tworzenia projektu pojawiają się komunikaty o błędach, możesz znaleźć rozwiązania problemów w ramce „Komunikaty o błędach: tworzenie aplikacji sieciowych”, która znajduje się w rozdziale 1.
- Chociaż pole tekstowe *Name* jest nieaktywne, to tekst, który wpiszesz w polu tekstowym *Location* jako uzupełnienie adresu *http://localhost*, zostanie przypisany jako nazwa projektu i pojawi się w polu tekstowym *Name*.
- Domyślnie *http://localhost* wskazuje na katalog *Inetpub/wwwroot*. Tak więc przykładowy projekt o nazwie *02VBVQS01* zostanie umieszczony w katalogu *Inetpub/wwwroot/02VBVQS01*. Możesz jednak użyć panelu administracyjnego *Internet Information Services*, aby umieścić projekt w innym miejscu i utworzyć wirtualny adres URL. Aby uzyskać dostęp do panelu administracyjnego IIS, należy otworzyć grupę *Narzędzia administracyjne (Administrative Tools)* w *Panelu sterowania (Control Panel)* komputera.

Otwórz teraz okno *Solution Explorer* (rysunek 2.4) i sprawdź, jakie pliki zostały dodane do utworzonego projektu. Zauważ, że wśród plików znajduje się moduł Web Service, z domyślną nazwą *Service1.asmx*. Informacje o tym, jak otworzyć okno *Solution Explorer*, znajdują się w rozdziale 1., w części zatytułowanej „Aby otworzyć okno *Solution Explorer*”.



Rysunek 2.4. W oknie *Solution Explorer* można zauważyć moduł usługi Web Service o rozszerzeniu *.asmx*



Rysunek 2.5. Projektant klasy Web Service może służyć do dodawania komponentów w trybie graficznym

Po utworzeniu nowego projektu powinien się w nim otworzyć moduł Web Service. Dzieje się tak, o ile nie zostaną zmienione domyślne ustawienia. Jeśli projekt nie jest otwarty, wystarczy dwukrotnie kliknąć jego nazwę w oknie *Solution Explorer*.

Projektant usług Web Service może służyć do dodawania komponentów do modułu klasy w trybie graficznym (rysunek 2.5). O ile projektowanie formularzy Windows w trybie graficznym jest sensowne, a większość formularzy Web posiada istotne elementy graficzne, o tyle projektowanie modułów Web Service w ten sposób przeważnie nie jest zasadne. Dzieje się tak, ponieważ usługi Web Service składają się z samego kodu w postaci metod (lub funkcji). Używanie narzędzi, które umożliwiają projektowanie graficzne, nie daje żadnych korzyści. Więcej informacji o używaniu projektantów do przyspieszenia procesu tworzenia aplikacji Windows i Web znajduje się w rozdziale 5. „Formularze Windows” oraz w rozdziale 16. „Aplikacje internetowe ASP.NET”.

Kiedy nowy projekt *ASP.NET Web Service* jest już otwarty, należy otworzyć edytor kodu, aby zapoznać się ze szczegółami modułu *Web Service*.

Aby otworzyć edytor kodu:

1. W oknie *Solution Explorer* wybierz moduł *Web Service*, a następnie opcję *Code* z menu *View*.

lub

Kliknij odnośnik *Switch to Code View* w projektancie *Web Service*.

Kiedy analizuje się moduł *Web Service* w edytorze kodu (rysunek 2.6), można zauważyć, że program rozpoczyna się od instrukcji *Imports*:

```
Imports System.Web.Services
```



Rysunek 2.6. W edytorze kodu można zauważyć oznaczoną jako komentarz metodę (*WebMethod*) *HelloWorld*

Instrukcja ta przesyła do kompilatora informację, że w kodzie mogą być używane klasy oraz elementy klas z przestrzeni nazw `System.Web.Services`.

Następnie umieszczone są linie kodu, które w rzeczywistości stanowią części jednej instrukcji:

```
<WebService(Namespace:=  
    "http://tempuri.org")>  
    Public Class Service1  
        Inherits System.Web.Services.WebService
```

Pierwsza część instrukcji, znacznik `<WebService>`, informuje środowisko Visual Basic, że kod stanowi usługę Web Service.

W dalszej części instrukcji znajduje się nazwa klasy (`Service1`) oraz określenie klasy, z której dziedziczy (`System.Web.Services.WebServices`). Więcej informacji o dziedziczeniu klas zawiera rozdział 3. „Praca z klasami”.

Wskazówka

- W następnej linii edytora kodu znajduje się szary znak + (plus), po którym można zobaczyć napis `Web Services Designer Generated Code`. Po kliknięciu tego znaku kod, który został utworzony automatycznie przez projektanta, rozwinie się i będzie widoczny. Kod ten ułatwia rozmieszczenie komponentów w projektancie usług Web Service.

W dalszej części kodu znajduje się przykładowa usługa Web Service: metoda o nazwie `HelloWorld`, która zwraca łańcuch tekstowy `"Hello World"`.

Na rysunku 2.6 fragment ten jest oznaczony jako komentarz za pomocą pojedynczych apostrofów (`'`). Po usunięciu symboli komentarza dostępny jest szkielec metody umieszczony w usłudze Web Service.

Kod kończy się instrukcją:

```
End Class
```

Kiedy wiadomo już, co znajduje się w module *Web Service* w konfiguracji standardowej, można przystąpić do pracy nad własną usługą. Na początek zmień domyślny adres URI.

Adres Uniform Resource Identifier

Domyślnie każda nowa usługa Web Service w środowisku Visual Studio posiada atrybut *Namespace* w postaci `http://tempuri.org`. Jest to adres URI (ang. *Uniform Resource Identifier*), który powinien wskazywać na lokalizację będącą pod kontrolą programisty. W przeciwieństwie do adresu URL, użytkownik nie musi mieć możliwości dostania się do tej lokalizacji.

Jeśli pozostawisz domyślny adres URI, każde uruchomienie strony testowej usługi Web Service spowoduje, że wyświetli się propozycja zmiany adresu.

Aby zmienić domyślny adres URI:

1. Przypisz adresowi URI w znaczniku WebService nową wartość:

```
<WebService(Namespace:= _
"http://peachpit.com/webservices")> _
Public Class Service1
Inherits System.Web.Services.WebService
```

Następnie dodaj metodę Web do usługi.

Aby dodać metodę Web do usługi:

1. Dodaj znacznik atrybutu <WebMethod>.

Funkcja oznaczona tym znacznikiem będzie metodą Web.

2. Utwórz funkcję.

Funkcja ta jest metodą Web.

Na listingu 2.1 pokazany jest cały kod usługi Web Service wraz z nową metodą Web, która zwraca łańcuch znaków.

Wskazówka

- Instrukcja

```
HelloVQS = "Visual Basic .NET jest super!"
```

przypisuje łańcuch znaków "Visual Basic .NET jest super!" do funkcji, która staje się wartością zwracaną przez metodę Web.

Listing 2.1. Usługa Web Service z metodą, która zwraca łańcuch znaków

```
Listing
Imports System.Web.Services

<WebService(Namespace:="http://peachpit.com/webservices/")> _
Public Class Service1
    Inherits System.Web.Services.WebService
    <WebMethod()> Public Function HelloVQS() As String
        HelloVQS = "Visual Basic .NET jest super!"
    End Function
End Class
```

Testowanie usług Web Service

Można łatwo testować usługi Web Service, jak też związane z nimi metody, za pomocą środowiska Visual Studio, które tworzy strony testowe.

Aby przetestować usługę Web Service i jej metody:

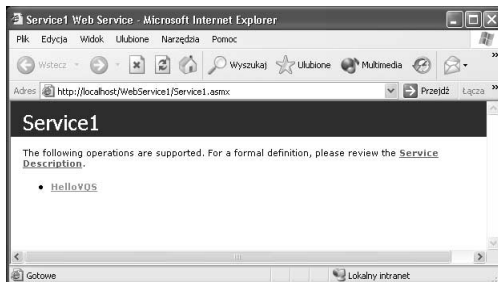
1. Uruchom projekt poprzez wybranie opcji *Start* z menu *Debug* lub kliknięcie strzałki *Start*.

Powinna otworzyć się strona testowa (rysunek 2.7).

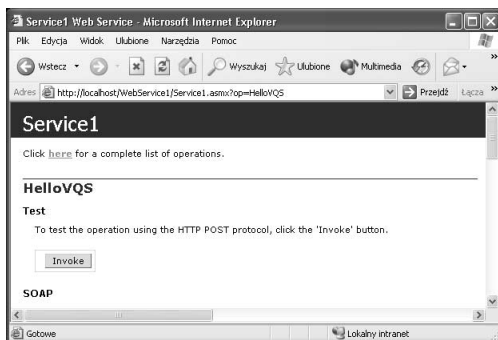
2. Kliknij odnośnik metody Web, który znajduje się na stronie testowej usługi. Otworzy się strona testowa dla konkretnej metody Web (rysunek 2.8).

3. Kliknij przycisk *Invoke*.

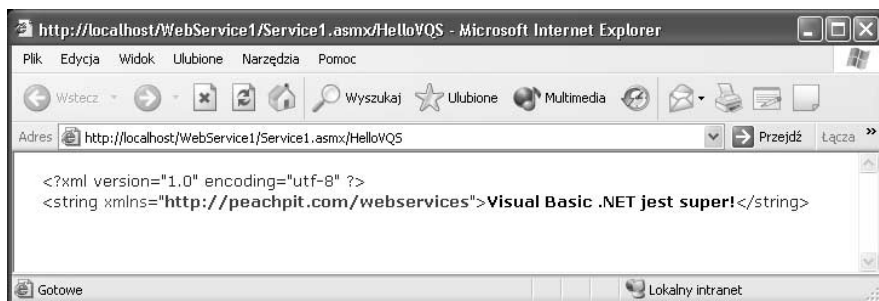
Program zwróci stronę w formacie XML, na której znajduje się wartość zwracana przez metodę Web oraz typ danych — w tym przypadku łańcuch znaków String (rysunek 2.9).



Rysunek 2.7. Strona testowa, wygenerowana dla usługi Web Service przez środowisko Visual Studio, zawiera odnośniki do metod, które ta usługa udostępnia



Rysunek 2.8. Aby przetestować metodę Web, kliknij przycisk *Invoke*



Rysunek 2.9. Wartość zwracana przez metodę Web wyświetlana jest w formacie XML

Dodawanie metod Web

Usługa Web Service może zawierać wiele metod Web. Żeby robić coś naprawdę pożytecznego, metoda powinna wykonywać coś więcej, niż tylko zwracanie łańcucha znaków. Stworzysz teraz metodę Web, która może przyjmować różne dane wejściowe. Zacznij od dodania metody Web, która oblicza sumę dwóch liczb całkowitych typu `Long`.

Aby dodać metodę Web, która sumuje dwie liczby całkowite:

1. Dodaj znacznik atrybutu `<WebMethod>`.

Funkcja oznaczona tym znacznikiem będzie metodą Web.

2. Utwórz funkcję o nazwie `AddTwoNums`, która będzie przyjmować jako argumenty dwie liczby całkowite typu `Long`. Zwracaną wartością będzie liczba tego samego typu.
3. Przypisz wynik dodawania do funkcji, jak przedstawiono to na listingu 2.2.

Wskazówka

- Przedstawiony kod nie sprawdza, czy użytkownik rzeczywiście wpisał liczby całkowite. Jeśli użytkownik wprowadzi na stronie testowej tekst zamiast liczby, to program zgłosi wyjątek. W rozdziale 6. „Używanie usług Web Service” znajduje się przykład, jak sprawdzać poprawność wprowadzanych przez użytkownika danych. Aby zrozumieć, jak działają wyjątki, należy zapoznać się z rozdziałem 9. „Wyjątki i usuwanie błędów”.

Listing 2.2. Metoda, która dodaje dwie liczby całkowite typu `long`

```
Listing
<WebMethod()> Public Function AddTwoNums(ByVal numOne As Long, _
    ByVal numTwo As Long) As Long
    AddTwoNums = numOne + numTwo
End Function
```

Testowanie metody Web

Nową metodę Web można łatwo testować. Wystarczy uruchomić usługę, która zawiera tę metodę.

Aby przetestować metodę, która dodaje dwie liczby:

1. Uruchom projekt poprzez wybranie opcji *Start* z menu *Debug* lub kliknięcie strzałki *Start*.

Otworzy się strona testowa usługi, na której widać obie metody Web (rysunek 2.10).

2. Na stronie testowej kliknij odnośnik do metody o nazwie *AddTwoNums*.

Otworzy się strona testowa metody *AddTwoNums*, na której znajdują się miejsca, gdzie użytkownik może wpisać wartości argumentów przyjmowanych przez tę metodę (rysunek 2.11).

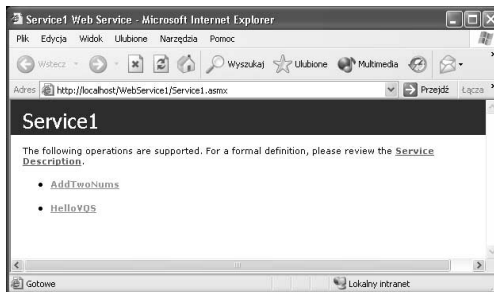
3. Wpisz wartości dla argumentów *numOne* oraz *numTwo*, a następnie kliknij przycisk *Invoke*.

Program wyświetli stronę w formacie XML. Na stronie tej powinna znaleźć się wartość zwracana przez metodę (suma dwóch wpisanych liczb) oraz typ danych (*long*) (rysunek 2.12).

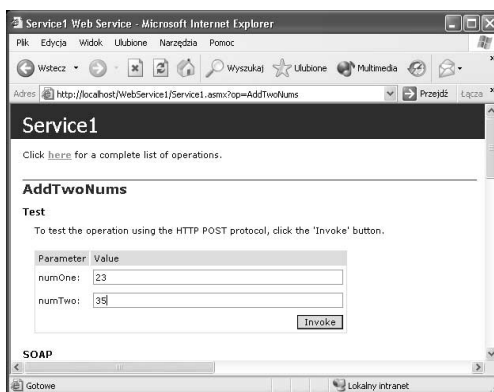
Podsumowanie

W rozdziale nauczyłeś się:

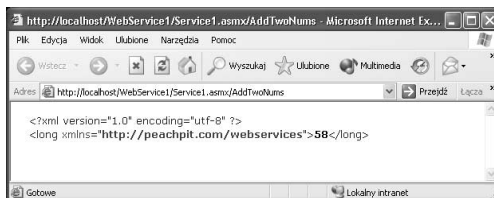
- ◆ opisywać usługi Web Service, a także związane z nimi standardy, takie jak SOAP, WSDL i UDDI;
- ◆ tworzyć projekt usługi Web Service;
- ◆ pracować z modułem *Web Service* w edytorze kodu;
- ◆ dodawać metodę do usługi Web Service;
- ◆ testować usługi i metody za pomocą strony testowej środowiska Visual Studio;
- ◆ dodawać i testować metodę Web, która przyjmuje dane wprowadzane przez użytkownika.



Rysunek 2.10. Nowa metoda Web pojawia się w postaci odnośnika na stronie testowej usługi Web Service



Rysunek 2.11. Strona testowa udostępnia pole do wpisywania argumentów przez użytkownika



Rysunek 2.12. Wynik działania metody Web na argumentach, które wprowadził użytkownik, wyświetla się w formacie XML