

Sekrety języka C#



**Dlaczego tworzenie
aplikacji w
Visual Studio .NET
2005 jest takie
proste?**

Niniejszy ebook jest **własnością prywatną**.

Niniejsza publikacja, ani żadna jej część, nie może być kopiowana, ani w jakikolwiek inny sposób reprodukowana, powielana, ani odczytywana w środkach publicznego przekazu bez pisemnej zgody wydawcy. Zabrania się jej publicznego udostępniania w Internecie, oraz odsprzedaży zgodnie z [regulaminem Wydawnictwa Złote Myśli](#).

© Copyright for Polish edition by [ZloteMysli.pl](#)

Data: 2.08.2007

Tytuł: Sekrety języka C #

Autor: Andrzej Stefańczyk

Wydanie I

ISBN: 978-83-7521-390-4

Projekt okładki: Marzena Osuchowicz

Korekta: Anna Grabka

Skład: Anna Grabka

Internetowe Wydawnictwo Złote Myśli

Netina Sp. z o. o.

ul. Daszyńskiego 5

44-100 Gliwice

WWW: www.ZloteMysli.pl

EMAIL: kontakt@zlotemysli.pl

Wszelkie prawa zastrzeżone.

All rights reserved.

SPIS TREŚCI

OD AUTORA.....	10
<u>CZEŚĆ I</u>	
<u>PODSTAWY C#.....</u>	11
<u>Rozdział 1.</u>	
<u>Przegląd platformy Microsoft.NET.....</u>	11
<u>Wprowadzenie do platformy .NET.....</u>	11
<u>Przegląd Framework .NET.....</u>	12
<u>Wspólne środowisko uruchomieniowe.....</u>	14
<u>Biblioteka klas .NET Framework.....</u>	16
<u>Obsługa baz danych (ADO.NET).....</u>	19
<u>Usługi webowe (XML Web Services).....</u>	21
<u>Aplikacje webowe (Web Forms).....</u>	21
<u>Aplikacje okienkowe (Windows Forms).....</u>	23
<u>Wspólna specyfikacja językowa (CLS).....</u>	24
<u>Języki programowania w .NET Framework.....</u>	25
<u>Rozdział 2.</u>	
<u>Pierwszy program.....</u>	27
<u>HelloWorld.....</u>	27
<u>Wejście-wyjście.....</u>	28
<u>Kompilacja i uruchomienie.....</u>	32
<u>Komentowanie kodu.....</u>	33
<u>Rozdział 3.</u>	
<u>Wprowadzenie do Microsoft Visual C#.NET 2005.....</u>	36
<u>Środowisko programisty.....</u>	36
<u>Okna narzędziowe.....</u>	37
<u>Generowanie szablonów projektów.....</u>	44
<u>Generowanie szablonu aplikacji konsolowej.....</u>	45
<u>Kompilacja i uruchomienie.....</u>	47
<u>Rozdział 4.</u>	
<u>Typy.....</u>	49
<u>Deklaracja zmiennej.....</u>	49
<u>Inicjacja zmiennej.....</u>	50
<u>Słowa kluczowe.....</u>	51
<u>Typy wartości.....</u>	52
<u>Typy proste.....</u>	52
<u>Typ wyliczeniowy.....</u>	54
<u>Struktura.....</u>	56
<u>Typy referencyjne.....</u>	56
<u>Typ object.....</u>	56
<u>Typ string.....</u>	57
<u>Tablica.....</u>	57
<u>Klasa.....</u>	58
<u>Interfejs.....</u>	58
<u>Delegacja.....</u>	58
<u>Stałe.....</u>	59
<u>Literały.....</u>	59

<u>Konwersje</u>	62
<u>Opakowywanie i rozpakowywanie</u>	65
Rozdział 5.	
<u>Operatory i wyrażenia</u>	66
<u>Wyrażenia</u>	66
<u>Operatory</u>	67
<u>Operatory arytmetyczne</u>	68
<u>Operatory logiczne bitowe</u>	69
<u>Operatory logiczne warunkowe</u>	71
<u>Operator konkatencji</u>	71
<u>Operatory jednostkowego zmniejszania i zwiększania</u>	72
<u>Operatory przesunięcia</u>	73
<u>Operatory relacji</u>	73
<u>Operatory przypisania</u>	74
<u>Operator dostępu do składnika klasy</u>	75
<u>Operator as</u>	77
<u>Znaki ignorowane w wyrażeniach</u>	78
Rozdział 6.	
<u>Instrukcje sterujące</u>	79
<u>Wprowadzenie</u>	79
<u>Instrukcja pusta</u>	79
<u>Blok instrukcji</u>	80
<u>Instrukcje wyboru</u>	80
<u>Instrukcja if</u>	81
<u>Instrukcja switch</u>	86
<u>Instrukcje iteracyjne</u>	92
<u>Instrukcja while</u>	93
<u>Instrukcja do</u>	95
<u>Instrukcja for</u>	98
<u>Instrukcja foreach</u>	101
<u>Instrukcje skoku</u>	102
<u>Instrukcja goto</u>	103
<u>Instrukcja break</u>	104
<u>Instrukcja continue</u>	104
Rozdział 7.	
<u>Klasy i obiekty</u>	106
<u>Podstawowe pojęcia</u>	106
<u>Klasa i obiekt</u>	106
<u>Relacje</u>	107
<u>Hermetyzacja</u>	107
<u>Abstrakcja</u>	107
<u>Kompozycja i dekompozycja</u>	108
<u>Składnik klasy</u>	108
<u>Składnica</u>	108
<u>Definiowanie klas</u>	109
<u>Modyfikatory</u>	110
<u>Modyfikatory dostępu</u>	111
<u>Tworzenie obiektu klasy</u>	111
<u>Pola</u>	112
<u>Konstruktor</u>	113
<u>Konstruktor domyślny</u>	113
<u>Inicjacja pól</u>	114
<u>Lista inicjacyjna</u>	118

Konstruktor kopiujący	119
Niszczenie obiektu klasy	119
Destruktor	120
Słowo kluczowe this	121
Metody klasy	122
Definiowanie	122
Zwracanie wartości	124
Argumenty	125
Wywoływanie	129
Przeciążanie	131
Statyczne składniki klasy	133
Przeciążanie operatorów	134
Przeciążanie operatorów relacji	136
Przeciążanie operatorów logicznych	138
Przeciążanie operatorów konwersji	139
Przeciążanie operatorów arytmetycznych	140
Właściwości	141
Indeksatory	144
Delegacje	148
Zdarzenia	149
Dziedziczenie	151
Dostęp do składowych klasy bazowej	153
Wywoływanie bazowych konstruktorów	154
Przesłanie metod	156
Ukrywanie metod	160
Klasy ostateczne	163
Klasy abstrakcyjne	164
Bazowa klasa System.Object	166
<u>Rozdział 8.</u>	
<u>Struktury</u>	168
Definiowanie struktur	168
Porównanie z klasami	169
Grupowanie pól	170
<u>Rozdział 9.</u>	
<u>Interfejsy</u>	171
Definiowanie interfejsów	171
Implementacja interfejsów	172
Implementacja metod interfejsów	173
Jawna implementacja metod interfejsów	175
Interfejs IDisposable	177
<u>Rozdział 10.</u>	
<u>Wyjątki</u>	180
Mechanizm wyjątków	180
Bloki try i catch	181
Klasy wyjątków	183
Rzucanie wyjątków	185
Blok finally	187
Przepelnienia arytmetyczne	188
<u>Rozdział 11.</u>	
<u>Przestrzeń nazw</u>	190
Deklarowanie przestrzeni nazw	190
Nazwy kwalifikowane	192
Dyrektywa using	193

<u>Tworzenie aliasów</u>	195
<u>Rozdział 12.</u>	
<u>Tablice</u>.....	197
<u>Deklaracja tablic</u>	197
<u>Wymiary tablic</u>	197
<u>Tworzenie instancji tablic</u>	198
<u>Dostęp do elementów</u>	199
<u>Inicjacja elementów tablic</u>	199
<u>Właściwości tablic</u>	200
<u>Metody operujące na tablicach</u>	201
<u>Zwracanie tablic z metod</u>	203
<u>Przekazywanie tablic do metod</u>	204
<u>Tablica argumentów Main</u>	205
<u>Rozdział 13.</u>	
<u>Łańcuchy</u>.....	206
<u>Klasa String</u>	206
<u>Pola, właściwości i metody klasy String</u>	207
<u>Budowanie łańcuchów – klasa StringBuilder</u>	212
<u>Rozdział 14.</u>	
<u>Kolekcje</u>.....	215
<u>Wprowadzenie</u>	215
<u>Klasa ArrayList</u>	215
<u>Klasa BitArray</u>	220
<u>Klasa Hashtable</u>	222
<u>Klasa Queue</u>	224
<u>Klasa SortedList</u>	226
<u>Klasa Stack</u>	230
<u>Rozdział 15.</u>	
<u>Data i czas</u>.....	232
<u>Rozdział 16.</u>	
<u>Foldery i pliki</u>.....	240
<u>Wprowadzenie</u>	240
<u>Klasa Directory</u>	241
<u>Klasa DirectoryInfo</u>	245
<u>Klasa File</u>	249
<u>Klasa FileInfo</u>	255
<u>Klasa FileStream</u>	260
<u>Klasa StreamReader</u>	262
<u>Klasa StreamWriter</u>	264
<u>Klasa BinaryReader</u>	265
<u>Klasa BinaryWriter</u>	268
<u>Klasa Path</u>	269
<u>Rozdział 17.</u>	
<u>Debugowanie</u>.....	272
<u>Wprowadzenie</u>	272
<u>Pułapki i śledzenie krokowe</u>	272
<u>Okna śledzenia</u>	274
<u>CZEŚĆ II.</u>	
<u>TWORZENIE APLIKACJI OKIENKOWYCH</u>.....	276
<u>Rozdział 1.</u>	
<u>Podstawy Windows Forms</u>.....	276

<u>Wprowadzenie</u>	276
<u>Generowanie aplikacji Windows Forms</u>	276

Rozdział 2.

<u>Praca z formą</u>	278
<u>Tworzenie formy</u>	278
<u>Właściwości formy</u>	280
<u>Obsługa zdarzeń</u>	282
<u>Metody formy</u>	283
<u>Przykładowa aplikacja</u>	284

Rozdział 3.

<u>Korzystanie z prostych kontrol</u>	286
<u>Dodawanie kontrol do formy</u>	286
<u>Organizowanie kontrol na formie</u>	287
<u>Wspólne cechy kontrol</u>	288
<u>Właściwości</u>	288
<u>Zdarzenia</u>	289
<u>Metody</u>	291
<u>Etykieta tekstowa</u>	291
<u>Właściwości</u>	291
<u>Etykieta łącza</u>	292
<u>Właściwości</u>	292
<u>Zdarzenia</u>	293
<u>Przycisk</u>	294
<u>Właściwości</u>	294
<u>Zdarzenia</u>	296
<u>Przykładowa aplikacja</u>	296
<u>Przycisk radiowy</u>	297
<u>Właściwości</u>	297
<u>Przykładowa aplikacja</u>	298
<u>Przycisk selekcji</u>	299
<u>Właściwości</u>	300
<u>Przykładowa aplikacja</u>	301
<u>Pole tekstowe</u>	303
<u>Właściwości</u>	303
<u>Przykładowa aplikacja</u>	305
<u>Pole tekstowe z wzorcem</u>	306
<u>Właściwości</u>	306
<u>Zdarzenia</u>	308
<u>Przykładowa aplikacja</u>	308
<u>Lista prosta</u>	309
<u>Właściwości</u>	309
<u>Zdarzenia</u>	310
<u>Kolekcja elementów</u>	311
<u>Przykładowa aplikacja</u>	311
<u>Lista selekcji</u>	313
<u>Właściwości</u>	313
<u>Zdarzenia</u>	314
<u>Przykładowa aplikacja</u>	315
<u>Lista rozwijana</u>	316
<u>Właściwości</u>	316
<u>Zdarzenia</u>	317
<u>Przykładowa aplikacja</u>	318
<u>Pole grupujące</u>	319
<u>Właściwości</u>	319

Przykładowa aplikacja.....	320
Pole obrazkowe.....	322
Właściwości.....	323
Zdarzenia.....	324
Przykładowa aplikacja.....	324
Panel.....	326
Właściwości.....	326
Przykładowa aplikacja.....	327
Pasek postępu.....	328
Przykładowa aplikacja.....	329
Suwak.....	330
Właściwości.....	330
Zdarzenia.....	331
Przykładowa aplikacja.....	331
Kalendarz.....	332
Właściwości.....	332
Zdarzenia.....	333
Przykładowa aplikacja.....	333
Pole numeryczne.....	334
Właściwości.....	335
Zdarzenia.....	335
Przykładowa aplikacja.....	336
Lista obrazów.....	337
Właściwości.....	337
Kolekcja obrazów.....	338

Rozdział 4.

<u>Korzystanie z zaawansowanych kontrolek.....</u>	340
Zakładki.....	340
Właściwości.....	340
Zdarzenia.....	341
Kolekcja stron.....	341
Przykładowa aplikacja.....	342
Drzewo.....	344
Właściwości.....	344
Zdarzenia.....	345
Kolekcja elementów drzewa.....	346
Przykładowa aplikacja.....	347
Lista złożona.....	349
Właściwości.....	349
Zdarzenia.....	352
Kolumny w widoku szczegółowym.....	353
Kolekcja elementów listy.....	353
Przykładowa aplikacja.....	354
Kontener podzielnika obszarów.....	356
Właściwości.....	356
Zdarzenia.....	357
Przykładowa aplikacja.....	357
Czasomierz.....	360
Właściwości.....	360
Zdarzenia.....	360
Przykładowa aplikacja.....	360

Rozdział 5.

<u>Interakcja z użytkownikiem.....</u>	362
Współpraca z myszą.....	362

<u>Parametry zdarzenia</u>	362
<u>Przykładowa aplikacja</u>	363
<u>Współpraca z klawiaturą</u>	364
<u>Parametry zdarzenia</u>	364
<u>Przykładowa aplikacja</u>	365
<u>Korzystanie z menu, paska narzędzi i paska stanu</u>	366
<u>Właściwości</u>	367
<u>Zdarzenia</u>	368
<u>Kolekcja elementów</u>	368
<u>Zarządzanie kolekcją elementów</u>	369
<u>Przykładowa aplikacja</u>	371
<u>Rozdział 6.</u>	
<u>Korzystanie z okien dialogowych</u>.....	373
<u>Tworzenie okien dialogowych</u>	373
<u>Przykładowa aplikacja</u>	373
<u>Wspólne okna dialogowe</u>	375
<u>Okna wyboru pliku</u>	375
<u>Okno wyboru folderu</u>	377
<u>Okno wyboru koloru</u>	378
<u>Okno wyboru czcionki</u>	378
<u>Przykładowa aplikacja</u>	380
<u>Rozdział 7.</u>	
<u>Tworzenie aplikacji MDI</u>.....	383
<u>Tworzenie aplikacji MDI</u>	383
<u>Przykładowa aplikacja</u>	384
<u>ŹRÓDŁA</u>.....	389

Od autora

Pisząc tą książkę starałem się przekazywać wiedzę stopniowo krok po kroku od prostych do bardziej złożonych zagadnień.

Opierając się na tej zasadzie postanowiłem podzielić książkę na dwie odrębne i różniące się nieco konwencją części.

Pierwsza część przedstawia składnię języka C# prostym i zrozumiałym dla każdego językiem z dużą ilością przykładów, wykorzystujących omówione w danym rozdziale elementy składni języka.

Druga część książki pokazuje w jaki sposób tworzy się aplikacje okienkowe *Windows Forms*, opisując najważniejsze komponenty *.NET Framework 2.0*. Zdecydowałem się opisać elementy najnowszej wersji *.NET Framework* ze względu na duże zmiany, jakie wprowadzono w stosunku do wersji poprzednich. To samo dotyczy nowej wersji środowiska *Microsoft Visual Studio .NET 2005* (w trakcie pisania książki zarówno *.NET Framework 2.0* jak nowe środowisko *Microsoft Visual Studio .NET 2005* dostępne były w wersji Beta).

Drogi czytelniku, mam nadzieję, że książka, którą napisałem pomoże Ci w poszerzaniu Twojej wiedzy z dziedziny programowania. Życzę Ci zatem wielu sukcesów i miłej lektury.

Pozdrawiam,
Andrzej Stefańczyk



Część I

Podstawy C#

Rozdział 1.

Przegląd platformy Microsoft.NET

Wprowadzenie do platformy .NET

Zanim zaczniemy naszą przygodę z językiem C# powinniśmy przyjrzeć się platformie, w jakiej uruchamiane są aplikacje stworzone w tym języku. Przed pojawieniem się platformy .NET, programista korzystający ze środowisk programistycznych *Microsoft* zmuszony był do korzystania z funkcji *Windows API* lub też nie do końca dobrze przemyślanych klas przysyłających te funkcje. Ze względu na dużą złożoność i coraz liczniej pojawiające się błędy spowodowane seriami poprawek i rozszerzeń, *Microsoft* zdecydował się na całkowitą zmianę koncepcji tworzenia aplikacji.

Nowe podejście polega przede wszystkim na zmniejszeniu liczby problemów, z jakimi musi zmagać się programista w czasie żmudnego procesu tworzenia. Do tej pory wiele problemów nastroczało poprawne zarządzanie pamięcią, zapewnianie przenośności kodu między różnymi językami programowania, poprawna orientacja w ogromnej liczbie funkcji *API*, obsługa błędów oraz brak mechanizmów kontroli. Wraz z pojawieniem się *platformy .NET* powyższe problemy przestały istnieć, dzięki czemu programista może skupić się nad tym, co ważne, czyli logiką aplikacji.

Platforma .NET to coś więcej niż środowisko do tworzenia aplikacji, to ogromny zbiór języków programowania funkcjonujących we wspólnym środowisku (*Visual Basic, Visual C++, Visual C#, Visual J#*), usług oferowanych przez serwery *.NET Enterprise (Microsoft Exchange Server, Microsoft SQL Server, Microsoft BizTalk Server, Microsoft Commerce Server*, itd.), usług dystrybuowanych (usługi dostępne przez Internet, wykorzystujące *XML* oraz *SOAP*) oraz usług dla urządzeń przenośnych (palmtopów, telefonów komórkowych, konsol, itp.).

Przegląd Framework .NET

Framework .NET jest pakietem komponentów do budowy i uruchamiania aplikacji opartych na technologii .NET. Został on zaprojektowany w taki sposób, aby:

- Zapewniał zgodność z istniejącymi standardami.

Wsparcie dla istniejących technologii takich, jak: *HTML, XML, SOAP, XSLT, XPath*

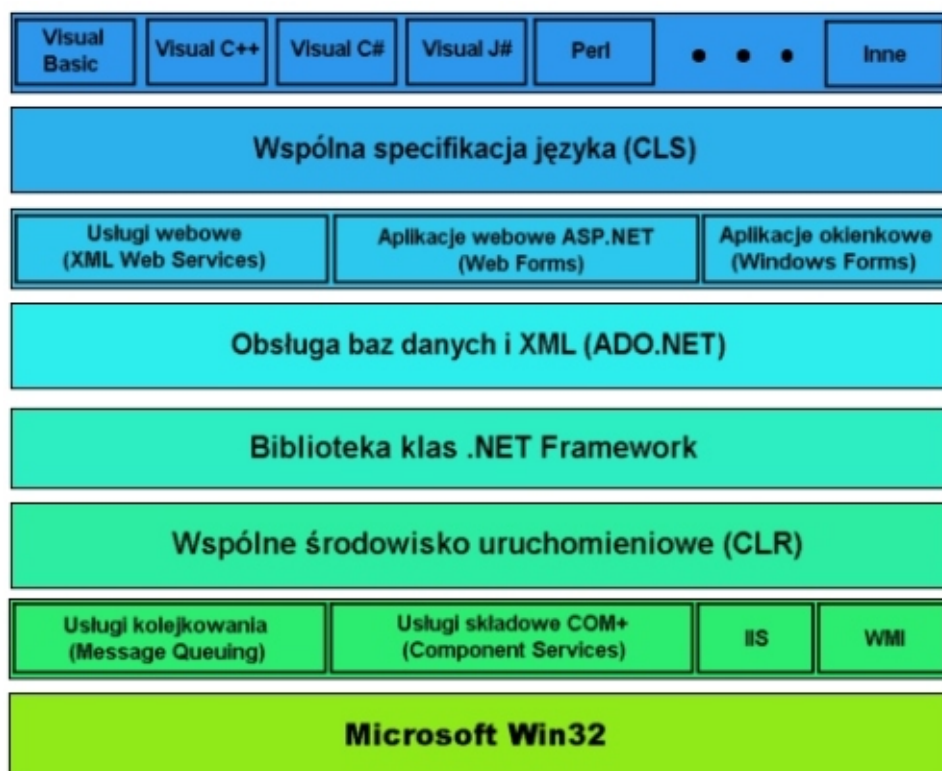
- Był prosty w użyciu.

Kod zorganizowany jest hierarchicznie poprzez przestrzenie nazw oraz klasy. Istnieje wspólna baza typów dla wszystkich języków. Każdy element języka jest obiektem.

- Pozwalał na rozszerzanie istniejącego zbioru klas.

Hierarchia przestrzeni nazw i klas nie jest ukryta. Programista może rozszerzać funkcjonalność każdej klasy poprzez mechanizm dziedziczenia (wyjątek stanowią jedynie klasy typu *sealed*).

- Zapewniał taką samą funkcjonalność niezależnie od języka programowania.



Rysunek 1. Architektura .NET Framework

Przyjrzyjmy się bliżej powyższemu rysunkowi. Na samym dole znajduje się system operacyjny, na którym działa framework.

Dalej mamy usługi aplikacyjne (dostępne poprzez klasy biblioteki klas): *Message Queuing* – kolejkowanie wiadomości, *Component Services* – usługi składowe (*COM+*), *IIS* (Internet Information Server) oraz *WMI* (Windows Management Instrumentation).

Kolejna warstwa to wspólne środowisko uruchomieniowe *CLR* (Common Language Runtime), które upraszcza proces tworzenia aplikacji, poprzez zapewnienie odseparowanego i zabezpieczonego środowiska uruchomieniowego, obsługę wielu języków oraz mechanizmów dystrybucji i zarządzania aplikacjami.

Następna warstwa to biblioteka klas, która zapewnia funkcjonalność (w postaci licznego zbioru klas) niezbędną do implementacji każdej aplikacji.

Kolejna warstwa to *ADO.NET*, zapewniająca szeroki dostęp do baz danych oraz wsparcie dla standardu *XML*.

Następna warstwa zawiera:

- *Web services* (usługi webowe, czyli komponenty, które mogą być współdzielone poprzez Internet),
- *Web forms* (aplikacje webowe, czyli oparte o *ASP.NET* aplikacje dostępne poprzez dynamicznie zmieniające się webowe interfejsy użytkownika),
- *Windows Forms* (aplikacje okienkowe, czyli aplikacje klienckie systemu Windows).

Kolejna warstwa to wspólna specyfikacja językowa *CLS* (Common Language Specification), czyli ujednoczony zbiór reguł dla każdego z dostępnych języków platformy .NET.

Ostatnia warstwa zawiera zbiór aktualnie dostępnych języków programowania platformy .NET.

Wspólne środowisko uruchomieniowe

Wspólne środowisko uruchomieniowe *CLR* (Common Language Runtime) znacznie ułatwia proces tworzenia aplikacji poprzez zapewnienie usług nadzorujących proces wykonywania kodu.

Aby możliwe było wykorzystanie usług wspólnego środowiska uruchomieniowego, kod należy skompilować przy pomocy współpracującego ze środowiskiem kompilatora. Kod wykonywalny przygotowany pod *CLR* nazywa się „*kodem nadzorowanym*”.

Wspólne środowisko uruchomieniowe zostało zaprojektowane z myślą o zintegrowaniu wielu języków programowania. Dzięki temu można przykładowo odziedziczyć w jednym języku klasę, która została napisana w innym języku. Poza tym, środowisko zapewnia mechanizm zarządzania

pamięcią, więc programista nie musi dłużej przejmować się tym, że zapomni zwolnić pamięć. Inną zaletą środowiska jest mechanizm kontroli wersji, który dba o to, aby do aplikacji dołączono wszystkie potrzebne komponenty. Środowisko posiada również jednolity mechanizm obsługi wyjątków (współpracujący z różnymi językami) oraz mechanizm kontroli typów danych.

Aby wykorzystać wszystkie mechanizmy środowiska, kompilator z nim współpracujący oprócz kodu nadzorowanego, musi przygotować tzw. *metadane*. Metadane służą do opisu typów danych używanych w kodzie programu i są przechowywane wraz z kodem w przenośnym pliku wykonywalnym (w skrócie *PE* – Portable Executable). Kod nadzorowany, o którym wspominałem, nie jest jednak kodem maszynowym, tylko kodem pośrednim zapisanym przy pomocy *języka pośredniego (IL* – Intermediate Language).

Język IL jest całkowicie niezależny od procesora, na którym zostanie wykonany kod, więc aby możliwe było jego wykonanie na środowisku docelowym musi istnieć kompilator, który przekształci kod IL w kod maszynowy. Kompilator przekształcający kod IL w kod maszynowy nazywany jest kompilatorem *JIT* (Just In Time).

Przyjrzyjmy się teraz elementom środowiska CLR:

Element środowiska	Opis
<i>Nadzorca klas</i>	Zarządza metadanymi, wczytuje interfejsy klas.
<i>Kompilator JIT</i>	Konwertuje kod pośredni (IL) na maszynowy.
<i>Nadzorca kodu</i>	Nadzoruje uruchomieniem kodu.
<i>Garbage collector (GC)</i> <i>“Kolekcjoner nieużytków”</i>	Automatycznie zwalnia nieużywaną pamięć dla wszystkich obiektów.
<i>Interfejs bezpieczeństwa</i>	Zapewnia mechanizmy bezpieczeństwa oparte na sprawdzaniu oryginalności kodu.
<i>Interfejs debugowania</i>	Pozwala na wyszukiwanie błędów w aplikacji i śledzenie stanu uruchamianego kodu.
<i>Nadzorca typów</i>	Nadzoruje proces konwersji typów i zabezpiecza przed wykonywaniem niebezpiecznych z punktu widzenia wykonania kodu rzutowań.

<i>Nadzorca wyjątków</i>	Zarządza obsługą wyjątków, informuje o wystąpieniu błędów.
<i>Interfejs wątków</i>	Dostarcza interfejs dla programowania wielowątkowego.
<i>Marszaler COM</i>	Dostarcza mechanizmy marszalingu do i z COM.
<i>Bazowa biblioteka klas</i>	Integruje kod z biblioteką klas .NET Framework.

Biblioteka klas .NET Framework

Biblioteka klas .NET Framework zawiera bardzo bogatą funkcjonalność w postaci dużej ilości klas. Klasy zorganizowane są hierarchicznie poprzez przestrzenie nazw. Podstawową przestrzenią nazw jest **System**, która zawiera bazowe klasy definiujące typy danych, zdarzenia i uchwytów zdarzeń, interfejsy, atrybuty oraz obsługę wyjątków. Pozostałe klasy zapewniają funkcjonalność: konwersji danych, manipulacji parametrami, operacji arytmetycznych i logicznych, zarządzania środowiskiem programu, nadzorowania zarządzanego i nie zarządzanego programu.

Znajomość podstawowych przestrzeni nazw .NET Framework jest bardzo istotna, gdyż dzięki tej znajomości można w łatwy sposób odnaleźć zestaw klas realizujących wymaganą przez nas w danej chwili funkcjonalność.

Przestrzeń nazw	Funkcjonalność
<i>Microsoft.CSharp</i>	Zawiera klasy wspierające proces kompilacji i generacji kodu dla języka C#.
<i>Microsoft.Win32</i>	Zawiera dwa rodzaje klas: obsługujące zdarzenia generowane przez system i obsługujące rejestr systemowy.
<i>System</i>	Zawiera bazowe klasy definiujące typy danych, zdarzenia i uchwytów zdarzeń, interfejsy, atrybuty oraz obsługę wyjątków.
<i>System.CodeDom</i>	Zawiera klasy, które można użyć do reprezentacji elementów i struktur kodu źródłowego dokumentu.
<i>System.Collections</i>	Zawiera interfejsy i klasy definiujące kolekcje obiektów takich jak: listy, kolejki, tablice bitowe, słowniki oraz hash-tablice.

<i>System.ComponentModel</i>	Zawiera klasy do manipulacji zachowaniem komponentów i kontrolek.
<i>System.Configuration</i>	Zawiera klasy oraz interfejsy pozwalające na programowalny dostęp do ustawień konfiguracyjnych .NET Framework oraz przechwytywania błędów związanych z obsługą plików konfiguracyjnych (plików z rozszerzeniem .config).
<i>System.Data</i>	Zawiera klasy wspierające architekturę <i>ADO.NET</i> (które pozwalają na łatwą manipulację danymi oraz źródłami danych).
<i>System.Diagnostics</i>	Zawiera klasy pozwalające na interakcję z procesami systemowymi, logami zdarzeń oraz licznikami wydajności.
<i>System.DirectoryServices</i>	Zapewnia łatwy dostęp do <i>Active Directory</i> .
<i>System.Drawing</i>	Zapewnia dostęp do podstawowej funkcjonalności obsługi grafiki <i>GDI+</i> .
<i>System.EnterpriseServices</i>	Zapewnia infrastrukturę dla aplikacji enterprise.
<i>System.Globalization</i>	Zawiera klasy definiujące informacje specyficzne dla danego kraju (język, strefa czasowa, format wyświetlania: daty, pieniędzy, liczb, etc.). Klasy są wykorzystywane do tworzenia aplikacji wielojęzycznych.
<i>System.IO</i>	Zawiera klasy pozwalające na manipulację plikami i strumieniami oraz zarządzanie plikami i folderami.
<i>System.Management</i>	Zapewnia dostęp do zbioru informacji administracyjnych oraz zdarzeń pochodzących z systemu, urządzeń oraz aplikacji. Przykładowo można uzyskać informacje dotyczące ilości dostępnego miejsca na dysku, aktualnego obciążenia procesora i wielu innych).
<i>System.Messaging</i>	Zawiera klasy pozwalające na łączenie się, monitorowanie oraz zarządzanie kolejkami wiadomości w sieci (wysyłanie, odbieranie i przetwarzanie wiadomości)
<i>System.Net</i>	Zapewnia prosty interfejs dla wielu protokołów sieciowych.
<i>System.Reflection</i>	Zawiera klasy i interfejsy pozwalające na dynamiczne tworzenie i wybieranie typów.

<i>System.Resources</i>	Zawiera klasy i interfejsy pozwalające na tworzenie, przechowywanie i zarządzanie zasobami specyficznymi dla danego języka.
<i>System.Runtime.Compiler Services</i>	Zapewnia funkcjonalność pozwalającą na zarządzanie atrybutami i metadanymi w czasie działania programu w środowisku CLR .
<i>System.Runtime.InteropServices</i>	Zapewnia wsparcie dla wywołań COM .
<i>System.Runtime.Remoting</i>	Zawiera klasy i interfejsy pozwalające na tworzenie i konfigurację dystrybuowanych aplikacji.
<i>System.Runtime.Serialization</i>	Zapewnia wsparcie dla mechanizmów serializacji obiektów.
<i>System.Security</i>	Zapewnia dostęp do systemu bezpieczeństwa środowiska CLR wraz z podstawowymi klasami do zarządzania prawami dostępu.
<i>System.ServiceProcess</i>	Zawiera klasy pozwalające na implementację, instalację oraz kontrolę usług systemu Windows.
<i>System.Text</i>	Zawiera klasy reprezentujące standardy kodowania znaków takie jak: ASCII , Unicode , UTF-7 oraz UTF-8 . Dodatkowo można tu znaleźć bardzo użyteczną klasę wspomagającą proces manipulacji łańcuchami znaków (StringBuilder).
<i>System.Threading</i>	Zawiera klasy i interfejsy wspierające programowanie wielowątkowe.
<i>System.Web</i>	Zawiera klasy i interfejsy zapewniające komunikację z serwerem Webowym.
<i>System.Windows.Forms</i>	Zawiera klasy i interfejsy do tworzenia aplikacji okienkowych. Zbiór zawiera pełną gamę komponentów niezbędnych do tworzenia interfejsu użytkownika.
<i>System.Xml</i>	Zawiera standardy obsługi i przetwarzania XML . Wspierane są następujące standardy: XML wersja 1.0: http://www.w3.org/TR/1998/REC-xml-19980210 (z uwzględnieniem wsparcia DTD) Przestrzenie nazw XML : http://www.w3.org/TR/REC-xml-names/ (zarówno poziom strumienia jak i DOM). Schematy XSD :

	http://www.w3.org/2001/XMLSchema Wyrażenia XPath : http://www.w3.org/TR/xpath Transformacje XSLT : http://www.w3.org/TR/xslt DOM Poziom 1: http://www.w3.org/TR/REC-DOM-Level-1/ DOM Poziom 2: http://www.w3.org/TR/DOM-Level-2/
--	--

Obsługa baz danych (ADO.NET)

ADO.NET to następca technologii *ADO (Microsoft ActiveX Data Objects)* zawierająca wsparcie dla *ADO.NET* to następca technologii *ADO (Microsoft ActiveX Data Objects)* zawierająca wsparcie dla obsługi baz danych oraz formatu *XML*. Technologię tą można wykorzystać zarówno do tworzenia zwykłych aplikacji klienckich Windows, jak i aplikacji przeznaczonych dla Internetu.

ADO.NET wspiera następujące rodzaje typów przechowywania danych:

- bez określonej struktury (dane nie posiadają logicznego uporządkowania np. proste notatki),
- o niehierarchicznej strukturze (dane podzielone są na odseparowane od siebie i uporządkowane jednostki np.: pliki tekstowe z danymi separowanymi przez znaki tabulacji, arkusze *Microsoft Excel*, pliki *Microsoft Active Directory*, itp.),
- hierarchiczne (dane przechowywane są w postaci struktur drzewiastych np. dokumenty *XML*),
- relacyjne (dane przechowywane są w tabelach zawierających kolumny o określonym typie danych i wiersze z danymi, tablice mogą być ze sobą logicznie połączone poprzez kolumny z identycznymi danymi czyli tzw. relacje, np.: baza *Microsoft SQL Server*, baza *Oracle*, itp.),
- obiektowe (dane przechowywane są w postaci obiektów np. obiektowe bazy danych)

Poniższe przestrzenie nazw zawierają pełną funkcjonalność **ADO.NET** znajdującą się w .NET Framework:

Przestrzeń nazw	Funkcjonalność
<i>System.Data</i>	Zawiera bazowe klasy do obsługi baz danych (przykładowo klasy zbioru danych <i>DataSet</i>).
<i>System.Data.Common</i>	Zawiera klasy i interfejsy z których dziedziczą „dostawcy danych” .NET (.NET data providers).
<i>System.Data.SqlClient</i>	Dostawca danych .NET <i>SQL Server</i>
<i>System.Data.OleDb</i>	Dostawca danych .NET <i>OLE DB</i>
<i>System.Data.Odbc</i>	Dostawca danych .NET <i>ODBC</i>
<i>System.Data.OracleClient</i>	Dostawca danych .NET <i>Oracle</i>
<i>System.Data.SqlTypes</i>	Zawiera klasy i struktury typów danych bazy <i>SQL Server</i> . Stanowi szybszą i bezpieczniejszą alternatywę dla pozostałych typów danych.
<i>System.Data.SqlServerCe</i>	Dostawca danych .NET <i>SQL Server CE</i> .
<i>System.Xml</i>	Zawiera klasy, interfejsy i typy wyliczeniowe zapewniające wsparcie dla przetwarzania dokumentów <i>XML</i> (np. klasa <i>XmlDataDocument</i>).
<i>System.Xml.Schema</i>	Zawiera wsparcie dla schematów <i>XML</i> (Schemas definition language - <i>XSD</i>). Wspierane są: Schematy dla struktur <i>XML</i> : http://www.w3.org/TR/xmlschema-1/ (wsparcie dla mapowania i walidacji schematu) Schematy dla typów danych <i>XML</i> : http://www.w3.org/TR/xmlschema-2/ (wsparcie dla typów danych XML)
<i>System.Xml.Serialization</i>	Zawiera wsparcie dla serializacji obiektów w postaci dokumentów <i>XML</i> .
<i>System.Xml.XPath</i>	Zawiera parser <i>XPath</i> . Wspiera: Język ścieżek <i>W3C XML (XPath)</i> w wersji 1.0 (www.w3.org/TR/xpath)
<i>System.Xml.Xsl</i>	Zawiera wsparcie dla <i>XSLT</i> (Extensible Stylesheet Transformation). Wspiera: Transformacje <i>W3C XSL (XSLT)</i> wersja 1.0 (www.w3.org/TR/xslt).

Usługi webowe (XML Web Services)

Usługi webowe (*XML Web Services*) to dostępne przez sieć usługi (realizujące określoną funkcjonalność), które można wykorzystać jako komponenty do budowy aplikacji rozproszonych. Usługi webowe oparte są na otwartych standardach Internetowych takich jak *HTTP*, *SOAP* oraz *XML*. Usługi webowe mogą dostarczać różną funkcjonalność począwszy od prostych komponentów informujących o cenach akcji publikowanych przez jakiś dom maklerski do złożonych komponentów pełniących funkcję aplikacji finansowych. Praktycznie nie ma ograniczeń, jeżeli chodzi o rozproszenie komponentów w sieci. Poza tym każdy komponent może wykorzystywać funkcjonalność innych komponentów rozproszonych w celu dostarczenia bardziej złożonej funkcjonalności.

Jedną z podstawowych cech usług webowych jest wysoki stopień abstrakcji istniejący między implementacją a użytkowaniem. Dzięki wykorzystaniu mechanizmu wymiany danych przez standard *XML* klient usługi jak i jej dostawca są zwolnieni z potrzeby informowania się nawzajem o formacie wejścia/wyjścia czy też położeniu.

Poniższe przestrzenie nazw są wykorzystywane przy tworzeniu usług webowych w .NET Framework:

Przestrzeń nazw	Funkcjonalność
<i>System.Web</i>	Dostarcza bazową funkcjonalność dla usług webowych.
<i>System.Web.Caching</i>	Dostarcza funkcjonalność związaną z przechowywaniem kopii często używanych danych (keshowaniem) z serwera.
<i>System.Web.Configuration</i>	Dostarcza funkcjonalność związaną z konfiguracją.
<i>System.Web.Security</i>	Dostarcza funkcjonalność związaną z bezpieczeństwem.
<i>System.Web.Services</i>	Dostarcza funkcjonalność niezbędną do tworzenia usług webowych.
<i>System.Web.SessionState</i>	Przechowuje dane dotyczące sesji użytkownika.

Aplikacje webowe (Web Forms)

Aplikacje webowe (*Web Forms*) są aplikacjami opartymi o *ASP.NET*, które dostępne są poprzez Internet. Tworzenie aplikacji webowej przypomina

tworzenie zwykłej aplikacji okienkowej. Podobnie jak to miało miejsce przy **ASP (Active Server Pages)**, **ASP.NET** działa na serwerze webowym, dzięki czemu pozwala na rozwijanie spersonalizowanych oraz dynamicznie zmieniających się sajtów webowych. Aplikacje webowe oparte na **ASP.NET** są całkowicie niezależne od typu przeglądarki po stronie klienta oraz używanego przez niego systemu operacyjnego.

Aplikacja webowa składa się z różnych współpracujących ze sobą elementów:

- stron webowych .aspx (dostarczają dynamiczny interfejs dla aplikacji webowej);
- kodu ukrytego za stroną webową (kod niewidoczny dla klienta, który jest skojarzony ze stroną webową i zawiera funkcjonalność aplikacji znajdującej się po stronie serwera);
- plików konfiguracyjnych (pliki te są plikami **XML** i zawierają domyślne ustawienia dla aplikacji webowej oraz serwera webowego, każda aplikacja webowa zawiera jeden plik konfiguracyjny Web.config, dodatkowo serwer webowy zawiera swój plik konfiguracyjny machine.config);
- pliku **global.aspx** (zawiera kod niezbędny do obsługi zdarzeń aplikacji zgłaszanych przez **ASP.NET**);
- odsyłaczy do usług webowych (odsyłacze pozwalają na wysyłanie i odbieranie danych z i do usługi webowej);
- połączenia z bazą (pozwalające na wymianę danych ze źródłem danych);
- kieszowania (pozwalające aplikacji webowej na szybszą odpowiedź po pierwszym żądaniu).

Poniższe przestrzenie nazw są wykorzystywane przy tworzeniu aplikacji webowych w .NET Framework:

Przestrzeń nazw	Funkcjonalność
System.Web	Dostarcza bazową funkcjonalność dla usług webowych.

<i>System.Web.Caching</i>	Dostarcza funkcjonalność związaną z przechowywaniem kopii często używanych danych (kaszowaniem) z serwera.
<i>System.Web.Configuration</i>	Dostarcza funkcjonalność związaną z konfiguracją.
<i>System.Web.Security</i>	Dostarcza funkcjonalność związaną z bezpieczeństwem.
<i>System.Web.Services</i>	Dostarcza funkcjonalność niezbędną do tworzenia usług webowych.
<i>System.Web.SessionState</i>	Przechowuje dane dotyczące sesji użytkownika.
<i>System.Web.UI</i>	Dostarcza funkcjonalność pozwalającą na kontrolę zachowania kontrolek i stron, które pojawiają się w aplikacji webowej w postaci interfejsu strony webowej.

Aplikacje okienkowe (Windows Forms)

Aplikacje okienkowe (*Windows Forms*) to klienckie aplikacje systemu Windows oparte o standardowy graficzny interfejs użytkownika. Z poziomu aplikacji klienckiej mamy do dyspozycji pełną gamę komponentów zawartych w .NET Framework

Poniższe przestrzenie nazw są wykorzystywane przy tworzeniu interfejsu użytkownika w aplikacjach okienkowych .NET Framework:

Przestrzeń nazw	Funkcjonalność
<i>System.Windows.Forms</i>	Dostarcza komponenty do budowy okienkowych aplikacji w standardzie Windows (okna, wspólne dialogi, paski narzędzi, menu, paski statusu, przyciski, listy rozwijane, napisy, pola edycyjne, itd.).
<i>System.Drawing</i>	Pozwala na dostęp do podstawowej funkcjonalności GDI+ .
<i>System.Drawing.Drawing2D</i>	Zapewnia obsługę grafiki wektorowej i 2D.
<i>System.Drawing.Imaging</i>	Zapewnia zaawansowaną funkcjonalność GDI+ wsparcia procesu przetwarzania obrazów.
<i>System.Drawing.Printing</i>	Dostarcza usługi związane z drukowaniem grafiki.
<i>System.Drawing.Text</i>	Zapewnia zaawansowaną funkcjonalność GDI+ wsparcia rysowania tekstu (pozwala na korzystanie z kolekcji czcionek).

Wspólna specyfikacja językowa (CLS)

Wspólna specyfikacja językowa **CLS** definiuje zespół typów oraz zasady posługiwania się nimi dla różnych języków w celu zapewnienia kompatybilności między nimi. Przestrzegając zasad dotyczących zgodności typów, autor biblioteki klas ma gwarancję, że jego biblioteka będzie mogła zostać użyta w dowolnym języku zgodnym z **CLS**. Dzięki takiemu podejściu, możemy przykładowo rozwinąć bazową funkcjonalność komponentu w C#, odziedziczyć po nim w Visual Basic i rozszerzyć jego możliwości, a następnie jeszcze raz odziedziczyć w C# i znowu rozszerzyć jego funkcjonalność.

Poniżej znajdują się niektóre zasady dotyczące typów we wspólnej specyfikacji językowej:

- typy proste należące do specyfikacji **CLS**: **bool, char, short, int, long, float, double, decimal, string, object**;
- numeracja tablicy rozpoczyna się od 0;
- rozmiar tablicy musi być znany i większy lub równy 1;
- typy mogą być abstrakcyjne lub ostateczne;
- typ może być pochodnym tylko dla jednego typu;
- typ nie musi mieć składowych;
- wszystkie typy wartości muszą dziedziczyć z obiektu **System.ValueType** (wyjątek stanowi typ wyliczeniowy – musi dziedziczyć z **System.Enum**);
- typy parametrów przekazywanych i zwracanych muszą być typami zgodnymi z **CLS**;
- można przeciążać konstruktory, metody i właściwości;
- metody statyczne muszą zawierać implementację a metody składowe i wirtualne mogą być metodami abstrakcyjnymi;
- metoda może być statyczna, wirtualna lub składowa;
- globalne statyczne metody i pola nie są dozwolone;

- pola statyczne mogą być stałymi lub polami do zainicjowania;
- typ wyniku zwracanego przez metody `get` i `set` danej właściwości musi być zgodny;
- właściwości mogą być indeksowane;
- dla typów wyliczeniowych dozwolone są jedynie następujące typy całkowite: ***byte***, ***short***, ***int*** lub ***long***;
- wyjątki zgłaszane przez program muszą dziedziczyć z ***System.Exception***;
- identyfikatory muszą się różnić między sobą czymś więcej niż tylko wielkością liter w nazwie (niektóre języki nie rozróżniają identyfikatorów na podstawie wielkości liter).

Języki programowania w .NET Framework

Microsoft.NET Framework dostarcza pełne wsparcie dla kilku różnych języków programowania:

- **Język C#** – został zaprojektowany na platformę .NET i jest pierwszym nowoczesnym językiem komponentowym w linii języków C/C++. W skład języka wchodzi: klasy, interfejsy, delegacje, mechanizmy opakowywania i odpakowywania, przestrzenie nazw, właściwości, indeksatory, zdarzenia, operatory przeciążania, mechanizmy wersjonowania, atrybuty, wsparcie dla wywołań kodu niezabezpieczonego i mechanizmy generacji dokumentacji ***XML***.
- **Rozszerzenia dla zarządzanego języka C++** – zarządzany C++ stanowi rozszerzenie dla języka C++. To rozszerzenie zapewnia programiście dostęp do .NET Framework.
- **Visual Basic .NET** – stanowi innowację w stosunku do poprzedniej wersji ***Visual Basic***. Wspiera mechanizmy dziedziczenia i polimorfizmu, przeciążania konstruktorów, obsługi wyjątków, sprawdzania typów i wiele innych.
- **Visual J# .NET** – przeznaczony dla programistów języka ***Java***, którzy

chcą korzystać ze wsparcia technologii .NET.

- ***JScript .NET.***
- Pozostałe (***APL, COBOL, Pascal, Eiffel, Haskell, ML, Oberon, Perl, Python, Scheme i Smalltalk.***)