Andrzej Stefańczyk

„Przegląd platformy Microsoft.NET”

Data: 05.07.2005

Tytuł: „Przegląd platformy Microsoft.NET”
Fragment utworu „Sekrety języka C#”
Autor: Andrzej Stefańczyk

Korekta techniczna i skład: Anna Grabka

Internetowe Wydawnictwo Złote Myśli
Złote Myśli s.c.
ul. Plebiscytowa 1
44-100 Gliwice
WWW: www.ZloteMysli.pl
EMAIL: kontakt@zlotemysli.pl

SPIS TREŚCI

PODSTAWY C# 4

ROZDZIAŁ 1. PRZEGLĄD PLATFORMY MICROSOFT.NET 4

WPROWADZENIE DO PLATFORMY .NET 4

PRZEGLĄD FRAMEWORK .NET 4

WSPÓLNE ŚRODOWISKO URUCHOMIENIOWE 6

BIBLIOTEKA KLAS .NET FRAMEWORK 7

OBSŁUGA BAZ DANYCH (ADO.NET) 9

USŁUGI WEBOWE (XML WEB SERVICES) 10

APLIKACJE WEBOWE (WEB FORMS) 10

APLIKACJE OKIENKOWE (WINDOWS FORMS) 11

WSPÓLNA SPECYFIKACJA JĘZYKOWA (CLS) 11

JĘZYKI PROGRAMOWANIA W .NET FRAMEWORK 12

JAK SKORZYSTAĆ Z WIEDZY ZAWARTEJ W PEŁNEJ WERSJI EBOOKA? 14

SPIS TREŚCI PEŁNEJ WERSJI EBOOKA

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 przysłaniających 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.

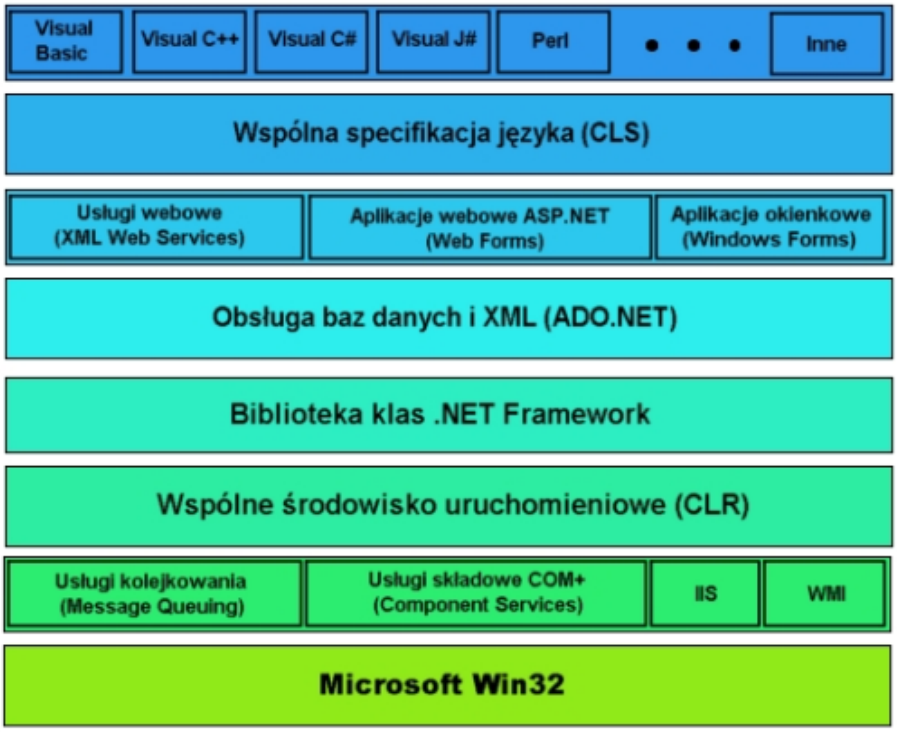
Nowe podejście polega przede wszystkim na zmniejszeniu liczby problemów, z jakimi musi zmagać się programista w czasie żmudnego procesu tworzenia. Do tej pory wiele problemów nastręczało poprawne zarządzanie pamięcią, zapewnianie przenośności kodu między różnymi językami programowania, poprawna orientacja 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 ujednolicony 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
Nadzorca klas	Zarządza metadanymi, wczytuje interfejsy klas.
Kompilator JIT	Konwertuje kod pośredni (IL) na maszynowy.
Nadzorca kodu	Nadzoruje uruchomieniem kodu.
Garbage collector (GC) ***“Kolekcjoner nieużytków”***	Automatycznie zwalnia nieużywaną pamięć dla wszystkich obiektów.
Interfejs bezpieczeństwa	Zapewnia mechanizmy bezpieczeństwa oparte na sprawdzaniu oryginalności kodu.
Interfejs debugowania	Pozwala na wyszukiwanie błędów w aplikacji i śledzenie stanu uruchamianego kodu.
Nadzorca typów	Nadzoruje proces konwersji typów i zabezpiecza przed wykonywaniem niebezpiecznych z punktu widzenia wykonania kodu rzutowań.
Nadzorca wyjątków	Zarządza obsługą wyjątków, informuje o wystąpieniu błędów.
Interfejs wątków	Dostarcza interfejs dla programowania wielowątkowego.

Marszaler COM	Dostarcza mechanizmy marszalingu do i z COM.
Bazowa biblioteka klas	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 uchwyty zdarzeń, interfejsy, atrybuty oraz obsługę wyjątków. Pozostałe klasy zapewniają funkcjonalność: konwersji danych, manipulacji parametrów, 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
Microsoft.CSharp	Zawiera klasy wspierające proces kompilacji i generacji kodu dla języka C#.
Microsoft.Win32	Zawiera dwa rodzaje klas: obsługujące zdarzenia generowane przez system i obsługujące rejestr systemowy.
System	Zawiera bazowe klasy definiujące typy danych, zdarzenia i uchwyty zdarzeń, interfejsy, atrybuty oraz obsługę wyjątków.
System.CodeDom	Zawiera klasy, które można użyć do reprezentacji elementów i struktur kodu źródłowego dokumentu.
System.Collections	Zawiera interfejsy i klasy definiujące kolekcje obiektów takich jak: listy, kolejki, tablice bitowe, słowniki oraz hash-tablice.
System.ComponentModel	Zawiera klasy do manipulacji zachowaniem komponentów i kontrolek.
System.Configuration	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).
System.Data	Zawiera klasy wspierające architekturę ***ADO.NET*** (które pozwalają na łatwą manipulację danymi oraz źródłami danych).
System.Diagnostics	Zawiera klasy pozwalające na interakcję z procesami systemowymi, logami zdarzeń oraz licznikami wydajności.
System.DirectoryServices	Zapewnia łatwy dostęp do ***Active Directory***.
System.Drawing	Zapewnia dostęp do podstawowej funkcjonalności obsługi grafiki ***GDI+***.
System.EnterpriseServices	Zapewnia infrastrukturę dla aplikacji enterprise.
System.Globalization	Zawiera klasy definiujące informacje specyficzne dla danego kraju (język, strefa czasowa, format wyświetlania: daty, pieniędzy, liczb, etc.). Klasy są wykorzystywane do tworzenia aplikacji wielojęzycznych.
System.IO	Zawiera klasy pozwalające na manipulację plikami i strumieniami oraz zarządzanie plikami i folderami.
System.Management	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).
System.Messaging	Zawiera klasy pozwalające na łączenie się, monitorowanie oraz zarządzanie kolejkami wiadomości w sieci (wysyłanie, odbieranie i przetwarzanie wiadomości)
System.Net	Zapewnia prosty interfejs dla wielu protokołów sieciowych.
System.Reflection	Zawiera klasy i interfejsy pozwalające na dynamiczne tworzenie i wybieranie typów.
System.Resources	Zawiera klasy i interfejsy pozwalające na tworzenie, przechowywanie i zarządzanie zasobami specyficznymi dla danego języka.
System.Runtime.CompilerServices	Zapewnia funkcjonalność pozwalającą na zarządzanie atrybutami i metadanymi w czasie działania programu w środowisku ***CLR***.
System.Runtime.InteropServices	Zapewnia wsparcie dla wywołań ***COM***.
System.Runtime.Remoting	Zawiera klasy i interfejsy pozwalające na tworzenie i konfigurację dystrybuowanych aplikacji.
System.Runtime.Serialization	Zapewnia wsparcie dla mechanizmów serializacji obiektów.
System.Security	Zapewnia dostęp do systemu bezpieczeństwa środowiska ***CLR*** wraz z podstawowymi klasami do zarządzania prawami dostępu.
System.ServiceProcess	Zawiera klasy pozwalające na implementację, instalację oraz kontrolę usług systemu Windows.
System.Text	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***).
System.Threading	Zawiera klasy i interfejsy wspierające programowanie wielowątkowe.
System.Web	Zawiera klasy i interfejsy zapewniające komunikację z serwerem Webowym.
System.Windows.Forms	Zawiera klasy i interfejsy do tworzenia aplikacji okienkowych. Zbiór zawiera pełną gamę komponentów niezbędnych do tworzenia interfejsu użytkownika.
System.Xml	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 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**
System.Data	Zawiera bazowe klasy do obsługi baz danych (przykładowo klasy zbioru danych ***DataSet***).
System.Data.Common	Zawiera klasy i interfejsy z których dziedziczą „dostawcy danych” .NET (.NET data providers).
System.Data.SqlClient	Dostawca danych .NET ***SQL Server***
System.Data.OleDb	Dostawca danych .NET ***OLE DB***
System.Data.Odbc	Dostawca danych .NET ***ODBC***
System.Data.OracleClient	Dostawca danych .NET ***Oracle***
System.Data.SqlTypes	Zawiera klasy i struktury typów danych bazy ***SQL Server***. Stanowi szybszą i bezpieczniejszą alternatywę dla pozostałych typów danych.
System.Data.SqlServerCe	Dostawca danych .NET ***SQL Server CE***.
System.Xml	Zawiera klasy, interfejsy i typy wyliczeniowe zapewniające wsparcie dla przetwarzania dokumentów ***XML*** (np. klasa ***XmlDataDocument***).
System.Xml.Schema	Zawiera wsparcie dla schematów ***XML*** (Schemas definition language - ***XSD***). Wspierane są: Schematy dla struktur ***XML***: http://www.w3.org/TR/xmlschema-1/ (wsparcie dla mapowania i walidacji schematu) Schematy dla typów danych ***XML***: http://www.w3.org/TR/xmlschema-2/ (wparcie dla typów danych XML)
System.Xml.Serialization	Zawiera wsparcie dla serializacji obiektów w postaci dokumentów ***XML***.
System.Xml.XPath	Zawiera parser ***XPath***.

	Wspiera: Język ścieżek ***W3C XML (XPath)*** w wersji 1.0 (www.w3.org/TR/xpath)
System.Xml.Xsl	Zawiera wsparcie dla ***XSLT*** (Extensible Stylesheet Transformation). Wspiera: Transformacje ***W3C XSL (XSLT)*** 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
System.Web	Dostarcza bazową funkcjonalność dla usług webowych.
System.Web.Caching	Dostarcza funkcjonalność związaną z przechowywaniem kopii często używanych danych (keszowaniem) z serwera.
System.Web.Configuration	Dostarcza funkcjonalność związaną z konfiguracją.
System.Web.Security	Dostarcza funkcjonalność związaną z bezpieczeństwem.
System.Web.Services	Dostarcza funkcjonalność niezbędną do tworzenia usług webowych.
System.Web.SessionState	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);

• keszowania (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.
System.Web.Caching	Dostarcza funkcjonalność związaną z przechowywaniem kopii często używanych danych (keszowaniem) z serwera.
System.Web.Configuration	Dostarcza funkcjonalność związaną z konfiguracją.
System.Web.Security	Dostarcza funkcjonalność związaną z bezpieczeństwem.
System.Web.Services	Dostarcza funkcjonalność niezbędną do tworzenia usług webowych.
System.Web.SessionState	Przechowuje dane dotyczące sesji użytkownika.
System.Web.UI	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
System.Windows.Forms	Dostarcza komponenty do budowy okienkowych aplikacji w stardarcie Windows (okna, wspólne dialogi, paski narzędzi, menu, paski statusu, przyciski, listy rozwijane, napisy, pola edycyjne, itd.).
System.Drawing	Pozwala na dostęp do podstawowej funkcjonalności ***GDI+***.
System.Drawing.Drawing2D	Zapewnia obsługę grafiki wektorowej i 2D.
System.Drawing.Imaging	Zapewnia zaawansowaną funkcjonalność ***GDI+*** wsparcia procesu przetwarzania obrazów.
System.Drawing.Printing	Dostarcza usługi związane z drukowaniem grafiki.
System.Drawing.Text	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 wchodzą: 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).

Jak skorzystać z wiedzy zawartej w pełnej wersji ebooka?

Interesuje Cię tematyka programistyczna? Chciałbyś poznać sekrety języka C# i nauczyć się nim sprawnie posługiwać? Teraz także Ty, możesz poznać wiedzę programisty jednej z największych firm informatycznych w Polsce (Prokom S.A.), udostępnioną w formie praktycznego kursu języka C#.

To pierwsza tego typu publikacja elektroniczna w Polsce.

Zobacz szczegóły, na stronie: http://c-sharp.zlotemysli.pl/