Platforma .NET – Wykład 1
Informacje o wykładzie i platformie .NET

Osoba prowadząca wykład, laboratorium i projekt:
dr hab. inż. Marek Sawerwain, prof. UZ

Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

e-mail : M.Sawerwain@issi.uz.zgora.pl
tel. (praca) : 68 328 2321,
pok. 328a A-2,
ul. Prof. Z.Szafrana 2,
65-246 Zielona Góra

Ostatnia kompilacja pliku: Sunday 17th March, 2024, t: 22:21

Spis treści

Ogólne warunki zaliczenia

Dwa sprawdziany pisemne, na siódmym i czternastym spotkaniu (studia niestacjonarne (SN), sprawdzian na ostatnim spotkaniu):

- 70% – ocena dostateczna (dst),
- 75% – ocena dostateczna plus (dst+),
- 80% – ocena dobra (db),
- 85% – ocena dobra plus (db+),
- 90% – ocena bardzo dobra (bdb),
- >100% – ocena celująca (bdb+).

osoby które nie uzyskały 70% uzyskują prawo do dodatkowego terminu poprawkowego, w trakcie sesji właściwej.
Dodatkowo, dwie prace pisemne (artykuły/eseje do trzydziestu stron, tematy prezentowane będą na wykładach, jeden temat z wykładów 1-6, drugi 7-14), które mogą się wliczać do oceny końcowej:

- pierwsza praca powinna zostać oddana na siódmym lub ósmym wykładzie (tryb SN - piąte spotkanie),
- druga praca powinna zostać oddana na przedostatnim lub ostatnim roboczym wykładzie,

prace po wyznaczonym terminie nie będą przyjmowane!

Zakres tematyczny

- Wprowadzenie do platformy .NET
- Przedstawienie struktury platformy .NET
- Rodzaje i typy platformy .NET
- Zarys środowiska .NET Framework
- Przegląd języków programowania dla platformy .NET
- Charakterystyka pakietu Microsoft Visual Studio
- Prezentacja środowiska programistycznego
- Edycja formularzy
- Uruchamianie aplikacji
- Tworzenie przykładowej aplikacji dla systemu Windows
- środowisko CLR (ang. Common Language Runtime)
- Podstawowe funkcje i usługi CLR
- Zarządzanie pamięcią i zasobami
- Zarządzanie watkami
- Struktura i układ metadanych
- Obsługa błędów przy użyciu wyjątków
- Integracja z bibliotekami Win32 DLL
- Metody interakcji między aplikacjami
- Porównanie środowiska CLR z JVM (ang. Java Virtual Machine)
- Podstawy programowania w języku C#
- Omówienie składni języka: instrukcje, zmienne, operatory i typy danych. Zasady tworzenie klas, metod, konstruktorów oraz obiektów tych klas

Zakres tematyczny

- Użycie tablic
- Przegląd podstawowych narzędzi zawartych w SDK (ang. Software Development Kit)
- Zaawansowane elementy języka C#
- Dyrektywy preprocesora
- Obsługa zdarzeń
- Obsługa błędów za pomocą wyjątków
- Operacje na łańcuchach znaków
- Korzystanie z komponentów interfejsu Windows
- Wykorzystanie wyrażeń regularnych
- Zdalne wywoływanie obiektów
- Dostęp i operacje na plikach
- Watki i ich synchronizacja
- Omówienie BCL (ang. Base Class Library)
- Budowanie komponentów .NET
- Zasady tworzenia, projektowanie, implementacja i testowanie komponentów
- Współpraca z komponentami COM i COM+
- Wykorzystanie języka XML na potrzeby platformy .NET
- Sposoby wymiany informacji z wykorzystanie dokumentów XML, przegląd API do przetwarzania dokumentów XML
- Metody dostępu do baz danych
- Dostęp do danych przy użyciu ADO.NET (ang. ActiveX Data Objects .NET)
- Przegląd obiektów ADO.NET

Zakres tematyczny

- Prezentacja danych z baz danych na witrynach internetowych
- Technologia ASP.NET (ang. Active Server Pages .NET)
- Klasy bazowe i podstawowe obiekty ASP.NET
- Użycie języka XML w połączeniu z ASP.NET
- Tworzenie stron WWW zawierających komponenty ASP.NET
- Tworzenie usług sieciowych przy użyciu „web services”
- Wykorzystanie protokołów SOAP (ang. Simple Object Access Protocol) i UDDI (ang. Universal Description, Discovery and Integration)
- Bezpieczeństwo aplikacji ASP.NET: kontrola dostępu, autoryzacja, szyfrowanie danych
- Programowanie mikrokontrolerów w .NET

Składowe oceny końcowej = wykład: 40% + laboratorium: 30% + projekt: 30%.

Plan wykładu – spotkania tydzień po tygodniu

(1) Informacje o wykładzie, pojęcie platformy, podstawowe informacje o platformie .NET

(2) Składowe platformy .NET: CLR, CTS, języki programowania, biblioteki klas, pojęcie podzespołu (ang. assembly)

(3) Programowanie w C# – środowisko VS, MonoDevelop, syntaktyka C#, wyjątki, współpraca z DLL

(4) Programowanie w C# – model obiektowy, typy uogólnione, lambda wyrażenia

(5) Programowanie w C# – aplikacje „okienkowe", programowanie wielowątkowe

(6) Programowanie w F# – podstawy, przetwarzanie danych tekstowych,

(*) "Klasówka I", czyli egzamin cześć pierwsza

(7) Dostęp do baz danych

Plan wykładu – tydzień po tygodniu

(8) Język zapytań LINQ, Entity Framework
(9) Obsługa standardu XML
(10) Technologia ASP.NET 1/2
(11) Technologia ASP.NET 2/2
(12) Model widok i kontroler – Model View Controller
(13) Tworzenie usług sieciowych SOAP i WCF (komunikacja sieciowa)
(14) Wykład monograficzny .NET 1
(15) Wykład monograficzny .NET 2
(*) "Klasówka II", czyli egzamin cześć druga

Pozycje polskojęzyczne

Literatura podstawowa:

1. Chappell D.: Zrozumieć platformę .NET, Helion, 2007
2. Duffy J.: .NET Framework 2.0, Zaawansowane programowanie, Helion, 2007
3. Michelsen K.: Język C#. Szkoła programowania, Helion, 2007,
4. Esposito D.: Tworzenie aplikacji za pomocą ASP.NET oraz ADO.NET, Wydawnictwo RM, 2002
5. Evjen B., Hanselman S., Rader D.: ASP.NET 4 z wykorzystaniem C# i VB. Zaawansowane programowanie, Helion, 2011
6. Lee W.M.: C# 2008. Warsztat programisty, Helion, 2010.
7. Troelsen A.: Język C# 2010 i platforma .NET 4.0, s.1400, Wydawnictwo Naukowe PWN, 2011.

Literatura uzupełniająca:

1. Halvorson M.: Microsoft Visual Basic.NET, Wydawnictwo RM, Warszawa, 2002
2. Burton K.: .NET CLR. Księga eksperta, Helion, 2002
3. Liberty J.: C# 2005. Wprowadzenie, O'Reilly, 2007
4. Cabrera L. F., Kurt C.: Architektura usług Web i jej specyfikacje. Klucz do zrozumienia WS, Microsoft Press,
5. Hewardt M.: Debugowanie .NET. Zaawansowane techniki diagnostyczne, Helion, 2010

Pozycje anglojęzyczne

Wydawnictwa, gdzie można odszukać wiele cennych i interesujących pozycji książkowych:

- Wydawnictwo A-PRESS (e-book) dział .NET – http://apress.com/
- Wydawnictwo WROX działy .NET/C#/ASP.NET – http://www.wrox.com

Literatura:

1. Novák. I, Velvárt A., Granicz A., Balássy G., Hajdrik A., Sellers M., Hillar G.C., Molnár A., Kanjilal J.: Visual Studio® 2010 and .NET 4 Six-in-One, Wiley Publishing, Inc., 2010,
2. Nash T.: Accelerated C# 2010, A-Press, 2010,
3. Solis D.M.: Illustrated C# 2010, A-Press, 2010,
4. Troelsen A.: Pro C# 2010 and the .NET 4 Platform, 5th Ed., A-Press, 2010,
5. Troelsen A., Japikse P.: C# 6.0 and the .NET 4.6 Framework, 7th Ed., A-Press, 2015,
6. Freeman A. and Rattz J.C. Jr.: Pro LINQ: Language Integrated Query in C# 2010, A-Press, 2010,
7. Pickering R., Eason K.: Beginning F# 4.0, A-Press, 2016,
8. Syme D., Granicz A., Cisternino A.: Expert F# 4.0, 4th Ed., A-Press 2015,
9. Mark J. Price, C# 9 and .NET 5 – Modern Cross-Platform Development. Packt Publishing, 2020.

Informacje w sieci Internet

- Standard ECMA 334 (C#): `http://www.ecma-international.org/publications/standards/ecma-334.htm`
- Standard ECMA 335 (CLI): `http://www.ecma-international.org/publications/standards/ecma-335.htm`
- Konsorcjum WWW – `http://www.w3.org`
- Microsoft .NET – `http://www.microsoft.com/net/`
- Strona poświęcona technologii .NET – `http://www.codeguru.pl`
- Projekt MONO – `http://www.mono-project.com/`
- Projekt DotGNU – `http://dotgnu.org/`
- Środowisko MonoDevelop – `http://monodevelop.com/`
- Środowisko SharpDevelop – `http://sharpdevelop.net/`
- Języki programowania w .NET – `http://DotNetLanguages.net`
- Język programowania Nemerle – `http://nemerle.org/`
- Język F# – `http://msdn.microsoft.com/pl-pl/fsharp`
- Język IronPython – `http://www.ironpython.info/`
- **.NET Blog** – `https://blogs.msdn.microsoft.com/dotnet/`
- **Reimplementacja C# oraz Silver** – `http://www.elementscompiler.com/elements/`

Inne wykłady i materiały

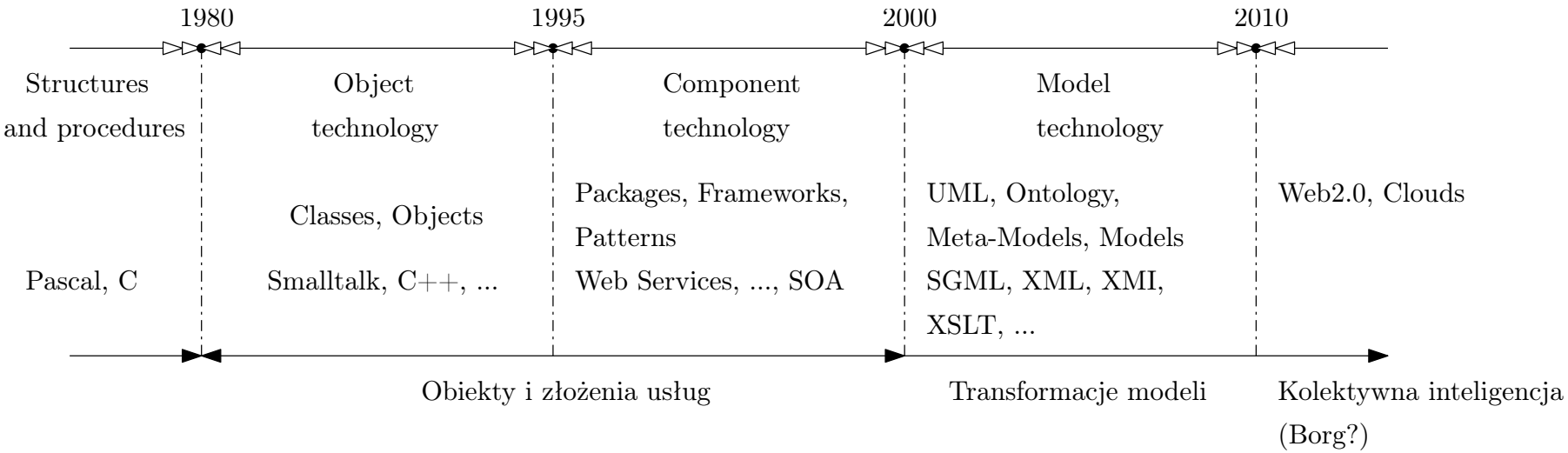
1. Wykład Macieja Piechówki – Politechnika Gdańska,
2. Materiały firmy Microsoft, Piotr Bubacz, ITA-103, Aplikacje Internetowe, zasoby Internetu,
3. Materiały do laboratorium do przedmiotu Platforma .NET.

Plan wykładu nr 1

1. Pojęcie platformy (platforma technologiczna)
   1. historia i przyszłość,
   2. aspekty tworzenia/projektowania platformy i oprogramowania,
   3. architektura wielowarstwowa,
   4. przykłady platform.
2. Platforma .NET – Podstawowe informacje
   1. źródła oraz cele platformy .NET,
   2. główne elementy platformy .NET,
   3. zalety platformy .NET.

Czym była, jest i będzie platforma informatyczna

Technologie wytwarzania oprogramowania:



Ogólnie technologia to:

- metoda przeprowadzania procesu produkcyjnego lub przetwórczego, ale też całokształt wiedzy potrzebnej do wytworzenia określonego dobra, charakteryzowana przez wiele czynników jak koszt, czas wytworzenia, wygoda, bezpieczeństwo, ...,

Natomiast platforma, czyli dziedzina wspólnego działania, charakteryzuje się min.:

- faktem, iż szczegóły implementacji danej platformy nie są ważne, najważniejsze są funkcjonalności udostępniane przez daną platformę,
- oferuje kompletną infrastrukturę do tworzenia aplikacji.

Na co trzeba zwrócić uwagę

Najważniejsze elementy to min.:

- funkcjonalność,
- wiarygodność,
- efektywność,
- łatwość pielęgnacji,
- elastyczność,
- i inne jak czas, koszt, jakość,
- ... .

$\Leftrightarrow$

Nie istnieją uniwersalne rozwiązania!

- technologia,
- inżynieria dziedziny,
- wzorce projektowe, aplikacji, architektury,
- tworzenie systemu w oparciu o systemy komponentów oraz usług,

Bezwzględnie, najważniejsze aspekty to:

1. Ludzie,
2. Technologia,
3. Organizacja.

Wyzwania projektowe dotyczą dużych i małych projektów

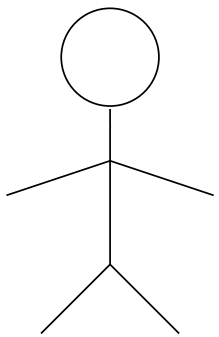
Współdziałanie

Skalowalność

Elastyczność

Ponowne użycie

Pielęgnowalność



Dostępność

Niezawodność

Zewnętrzne i wewnętrzne bezpieczeństwo

Odtwarzalność oraz Wydajność

Umęczony "developer"

Zasady projektowania oprogramowania – 1/2

Obiektowe i nie tylko zasady projektowania oprogramowania:

(1) hermetyzacja albo ukrywanie danych – ukrycie wewnętrznych szczegółów realizacji od aspektów używania danej klasy bądź modułu, inaczej mówiąc o stosowaniu klasy czy też zestawu funkcji trzeba wiedzieć tylko tyle ile trzeba (znajomość szczegółów implementacji nie jest potrzebna),

(2) minimalne powiązania – poszczególne moduły projektu powinny posiadać minimalne zależności, komunikacja pomiędzy modułami również powinna być minimalna

(3) spójność i zwartość – dany moduł/klasa powinna dotyczyć jednego pojęcia lub zespołu wspólnych pojęć,

(4) metaprogramowanie – zwiększenie abstrakcji, poprzez pisanie/tworzenie programów za pomocą komponowania modułów celem otrzymania zakładanej funkcjonalności, również pisanie programu którego zadaniem jest utworzenie innego programu.

Zasady projektowania oprogramowania – 2/2

Obiektowe i nie tylko zasady projektowania oprogramowania:

(5) otwartość i zamkniętość – klasa (moduł) powinna być łatwo rozszerzalna, jednak z drugiej strony musi być zamknięta/zabezpieczona przed modyfikacjami,

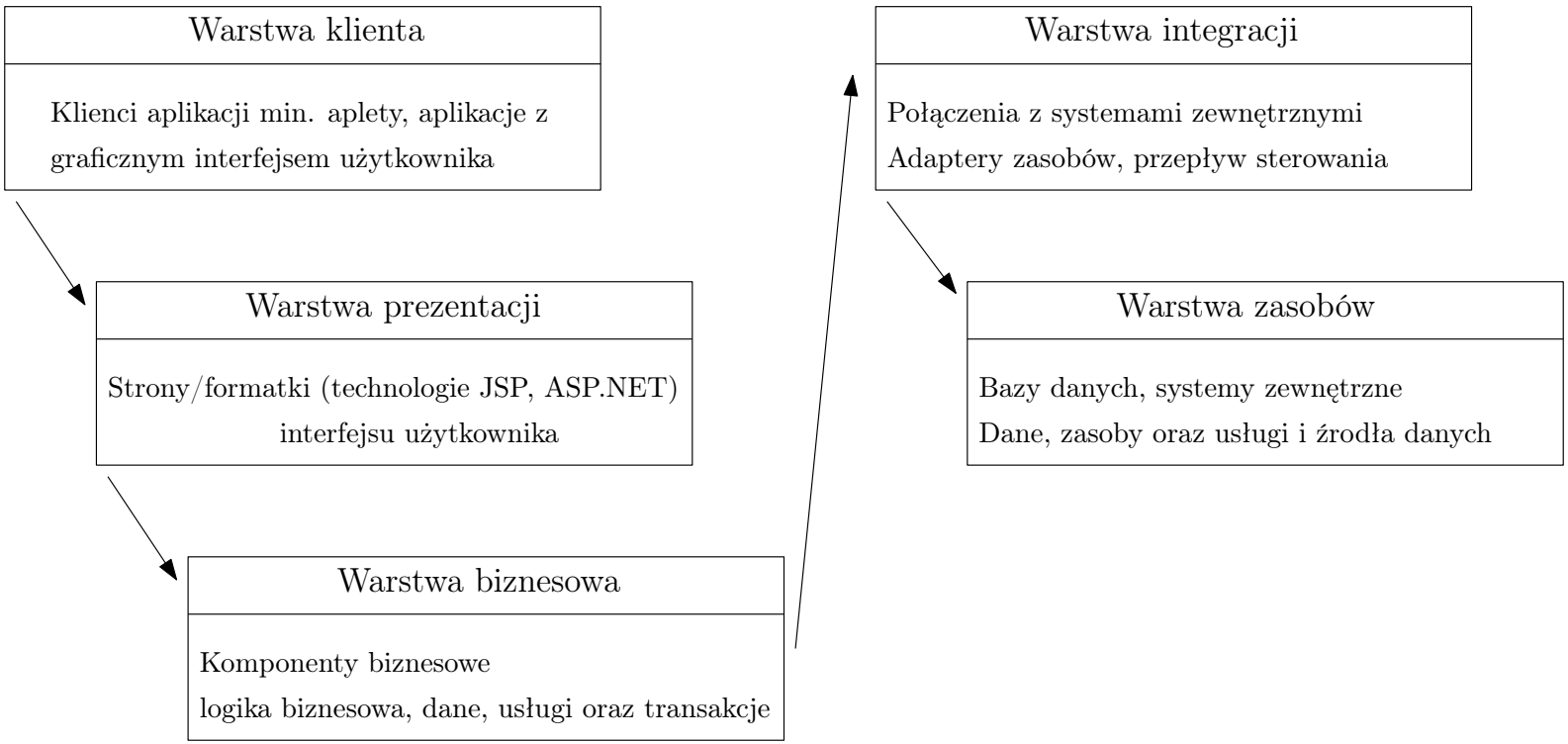
(6) programowanie w oparciu o kontrakty – operacja albo zestaw operacji określa się przez kontrakt, który wprowadza ograniczenia do implementacji:

- warunek wstępny,
- warunek końcowy,
- niezmiennik (inwariant) prawdziwy w trakcie realizacji operacji/zestawu operacji,

(7) rozdzielanie zagadnień w podejściu aspektowym,

- rozdział zagadnień np.: funkcjonalnych jeśli są niezależne,
- rozdział zagadnień technicznych (np.: trwałość danych, komunikacji, bezpieczeństwa, etc.).

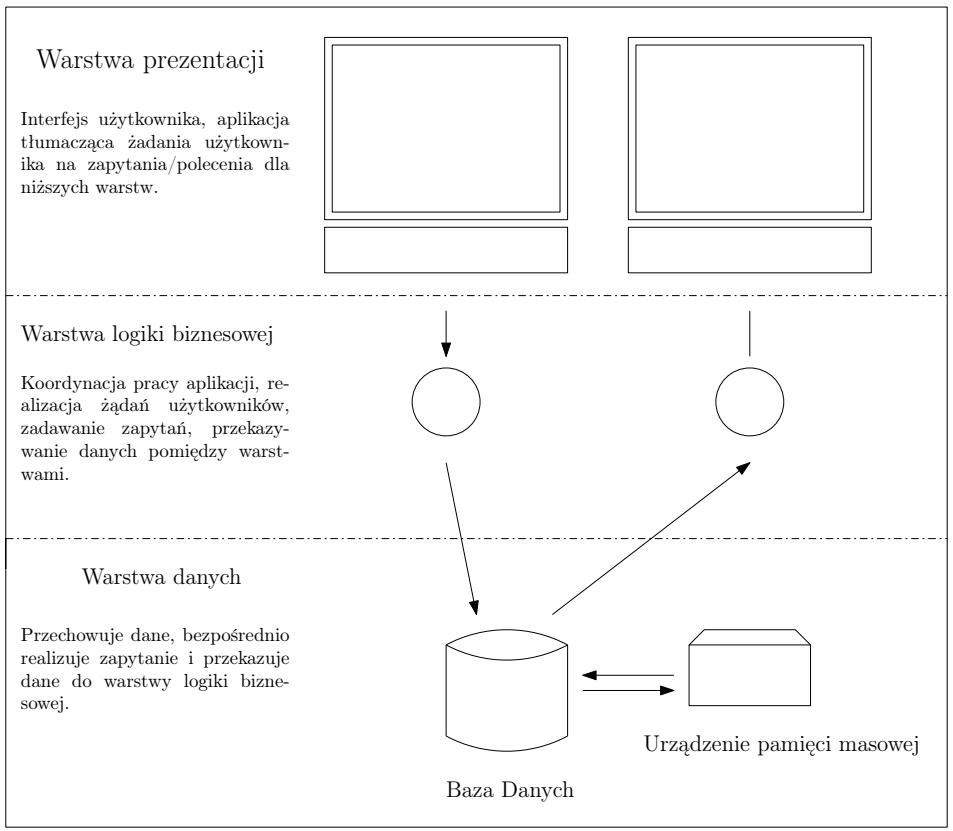
Architektura wielowarstwowa



Architektura wielowarstwowa (ang. multi-tier architecture lub n-tier architecture) to architektura komputerowa typu klient-serwer. Interfejs użytkownika, przetwarzanie i składowanie danych jest rozdzielone na kilka osobnych warstw. Mogą być one rozwijane i aktualizowane niezależnie. Ułatwia to ich utrzymanie i nie wpływa negatywnie na funkcjonowanie pozostałych warstw.

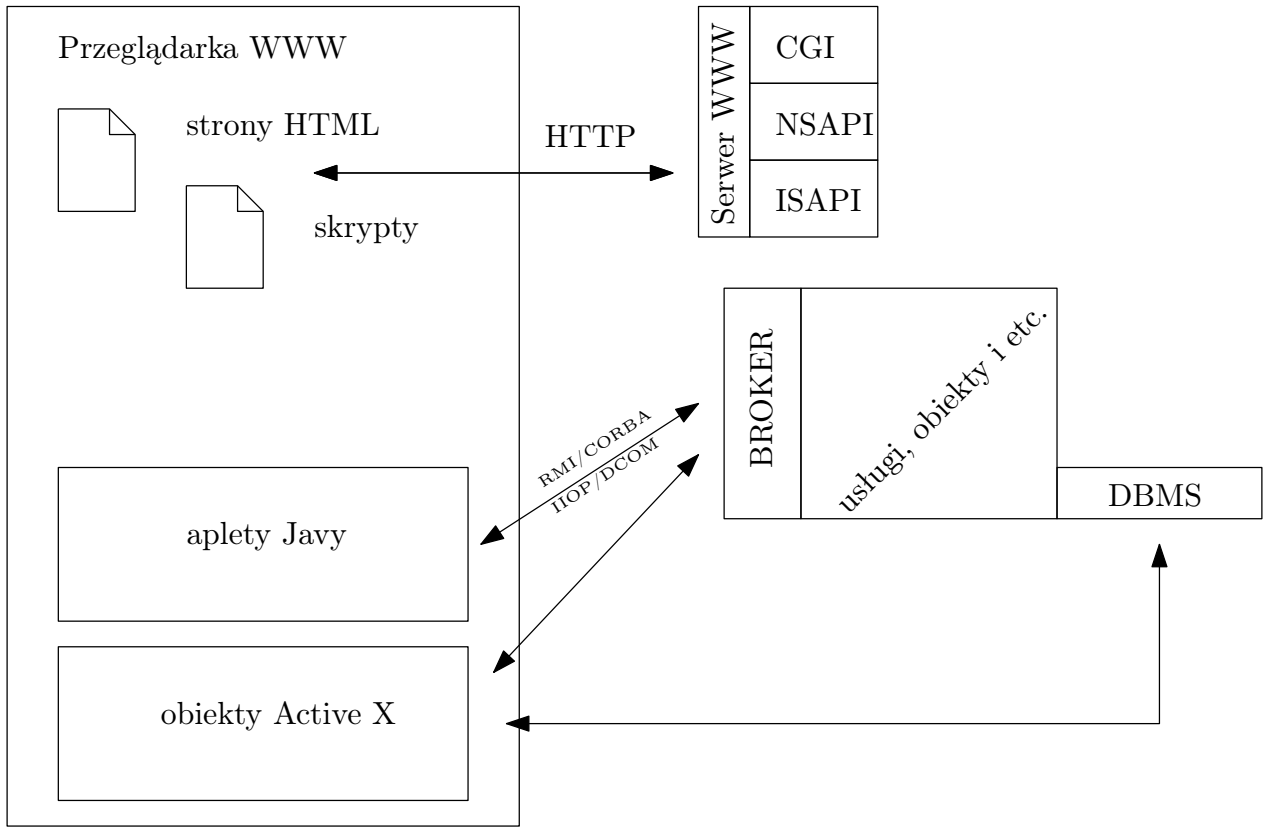
Architektura trójwarstwowa

Najpowszechniej używanym przykładem architektury wielowarstwowej jest architektura trójwarstwowa:



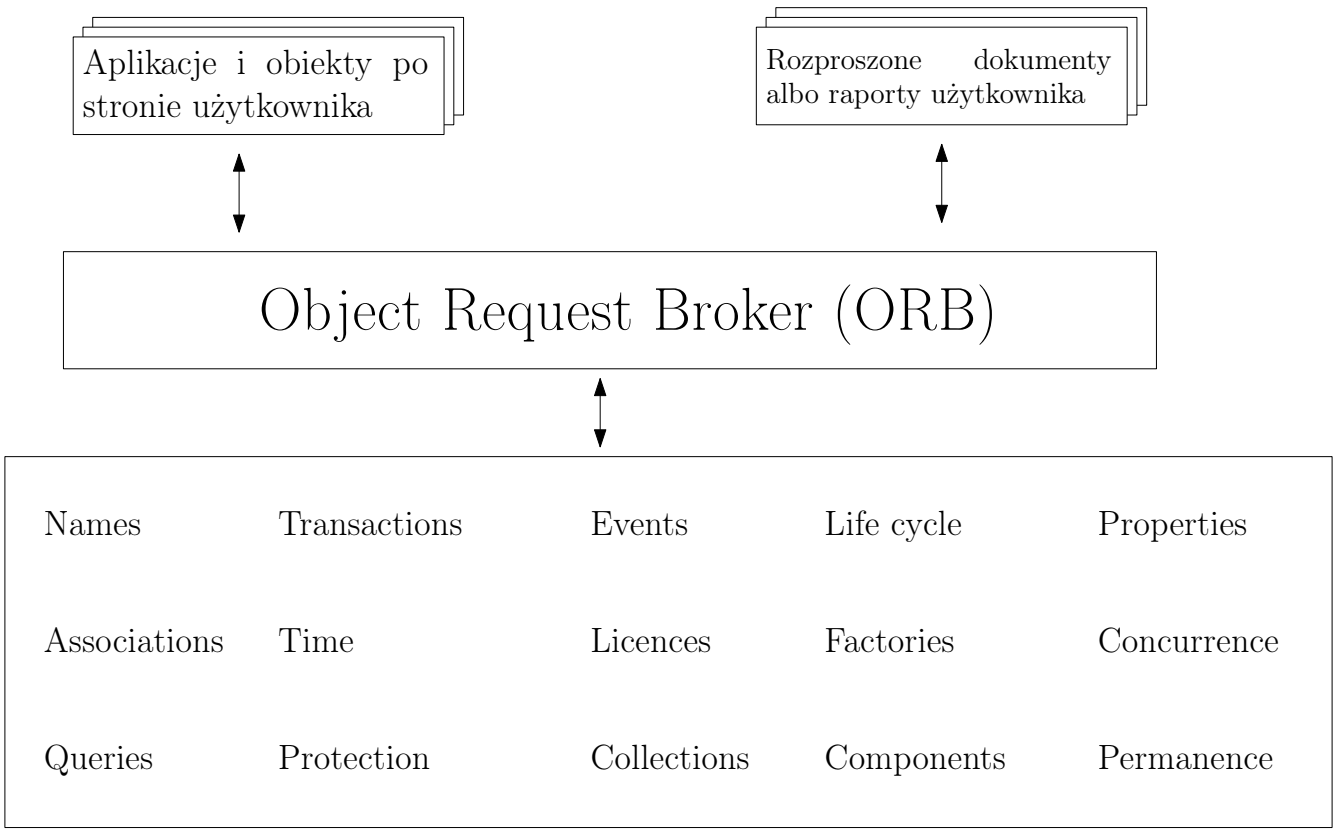
Zmiana szczegółów implementacji w jednej warstwie nie może wpływać na pozostałe warstwy.

Ogólny schemat systemów WEB

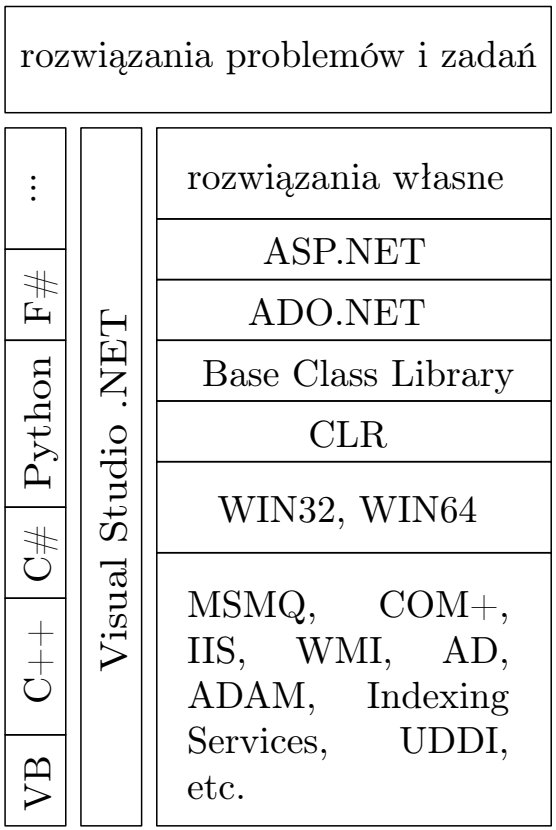


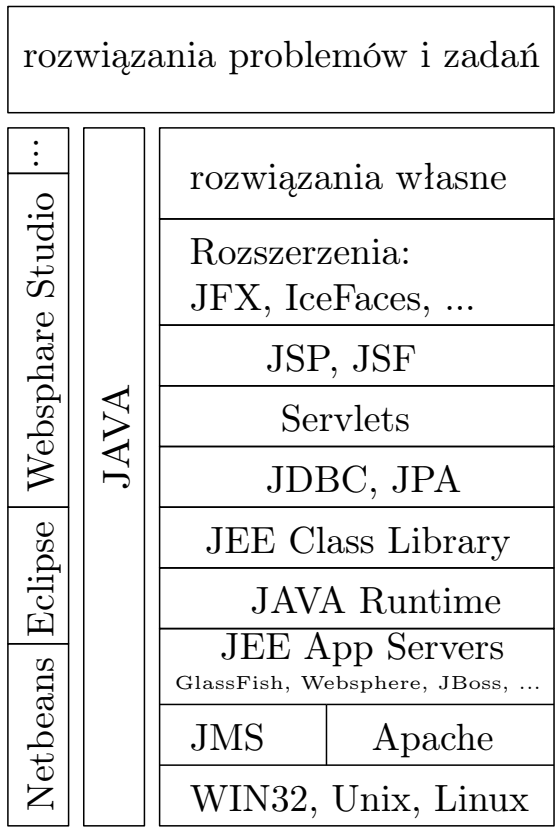
- CGI – Common Gateway Interface
- ISAPI/NSAPI — Internet/Netscape Server API
- RMI – Remote Method Invocation
- CORBA – Common Object Request Broker Architecture
- CORBA/IIOP – Internet Inter ORB Protocol

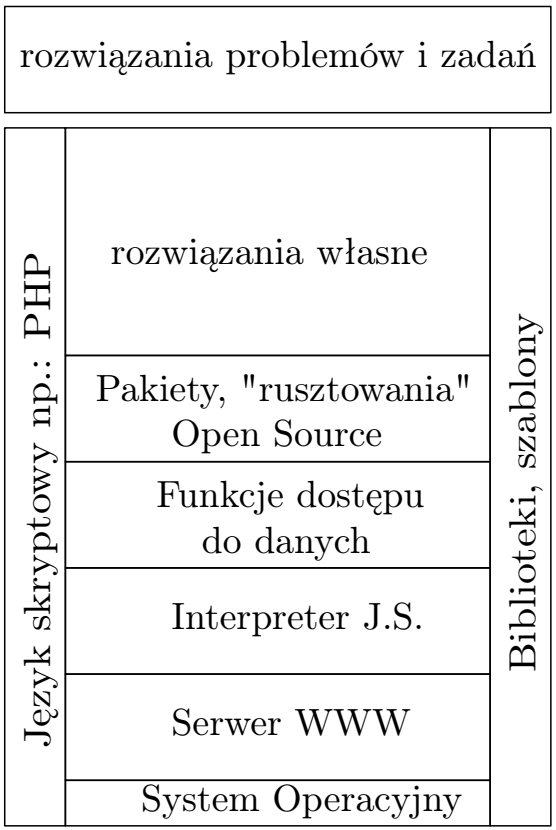
CORBA/OMA – zarządzanie obiektami



Platformy WEB – trzy główne przykłady







Platforma łączy w sobie języki, biblioteki i architektury

Infrastruktura

Infrastruktura łączy zbiór usług dostępnych dla komponentów, co pozwala na ich koordynację i w efekcie budować aplikacje które rozwiązują założone zadania i problemy.

Podstawowe typy/kategorie usług to usługi w rodzaju:

1. pakowania – zapis stanu/właściwości obiektów czy komponentów w aktywnej aplikacji,
2. cykl życia i rozproszenia – tworzenie/aktywacja/dezaktywacja obiektów, zarządzenie zasobami,
3. bezpieczeństwo – uwierzytelnianie/autoryzacja dostępu, szyfrowanie informacji,
4. zarządzanie transakcjami – dbanie o spójność danych,
5. komunikacja synchroniczna oraz asynchroniczna – wymiana informacji, dostęp do usług w trybie pełnej synchronizacji jeśli proces tego wymaga, lub komunikacja asynchroniczna bez konieczności synchronizacji klienta i serwera.

Obecne infrastruktury obiektowo/komponentowe: CORBA, CORBA/IIOP, COM+, .NET, Java BEANS (Java EE).

Zagadnienia omawiane w tej części

Platforma .NET – Podstawowe informacje:

1. źródła oraz cele platformy .NET,
2. główne elementy platformy .NET,
3. kompilacja programów,
4. zalety platformy .NET,
5. suma dwóch liczb oraz zliczanie linii.

Platforma .NET to obecnie zalecana technologia do tworzenia programów dla systemów z rodziny Windows, jednakże nie jest to jedyne rozwiązanie:

1. WIN32, WIN64 C API, tworzenie oprogramowania bezpośrednio na poziomie systemu operacyjnego, droga trudniejsza i bardziej żmudna ale oferuje bardzo dużą wydajność,
2. korzystanie z bibliotek MFC, ATL, VCL – obiektowe (łatwiejsze) programowanie nadal blisko poziomu OS,
3. Visual Basic – popularny język oraz środowisko, który ukrywa szczegóły API WIN32, upraszcza także tworzenie interfejsu użytkownika, tworzenie obiektów/komponentów COM,
4. Delphi – popularny język i środowisko, ogólne zasady podobne do Visual Basica, istnieje również darmowa/OpenSource odmiana środowiska Lazarus, zgodna choć nie w 100% ze środowiskiem Delphi, umożliwia także bezpośredni dostęp do API WIN32,
5. programowanie COM – model obiektów i komponentów który w założeniach jest niezależny od języka programowania, jego położenie również jest przezroczyste, dany obiekt nie musi znajdować się na tym samym komputerze, co oprogramowanie klienta.

Korzystanie z API WIN32/COM wymaga troszczenia się o szczegóły implementacyjne, zastosowanie VB bądź Delphi eliminuje w pewnym sensie ten wymóg, oraz co ważne w przypadku Delphi tworzony jest kod maszynowy.

Główne złożenia oraz cele

Główne właściwości platformy .NET:

1. współpraca z istniejącym kodem (obiekty COM, biblioteki DLL, słowo kluczowe **dynamic** w .NET 4.0),
2. wsparcie dla różnych języków programowani (C#, VB, F#, IronPython, i etc.),
3. wspólne środowisko uruchomieniowe dla języków .NET,
4. całkowita integracja różnych języków programowania na poziomie dziedziczenia klas, przechwytywanie wyjątków, „odpluskwianie" (ang. debugging) kodu,
5. ukrywanie szczegółów implementacji modelu COM, min. interfejsy typu: IClassFactory, IUnknown, IDispatch, IDL, typ wariantowy,
6. uproszczenie modelu rozwoju aplikacji, nie trzeba rejestrować obiektów poprzez rejestr, określona aplikacja może współpracować z wieloma wersjami obiektów w postaci plików DLL.

Ewolucja pakietu .NET

Kalendarium wydań platformy .NET:

Wersja	Data wydania	Nowe funkcje
.NET Framework 1.0		
.NET Framework 1.0 SP1	19 marca 2002	
.NET Framework 1.0 SP2	7 sierpnia 2002	
.NET Framework 1.0 SP3	9 września 2004	
.NET Framework 1.1	10 lipca 2003	mobile ASP.NET, ODBC, .NET Compact Framework, protokół IPv6
.NET Framework 1.1 SP1	9 września 2004	

Ewolucja pakietu .NET

Kalendarium wydań platformy .NET:

Wersja	Data wydania	Nowe funkcje
.NET Framework 2.0	17 luty 2006	wsparcie p. 64-bitowej, .NET Micro Framework, typy uogólnione, klasy częściowe, metody anonimowe
.NET Framework 2.0 SP1	19 listopada 2007	
.NET Framework 2.0 SP2	16 stycznia 2009	
.NET Framework 3.0	21 listopada 2006	Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), Windows Workflow Foundation (WWF), CardSpace

Ewolucja pakietu .NET

Kalendarium wydań platformy .NET:

Wersja	Data wydania	Nowe funkcje
.NET Framework 3.5	9 listopada 2007	platforma bytów, LINQ, metody rozszerzeń, drzewa wyrażeń
.NET Framework 3.5 SP1	11 sierpnia 2008	
.NET Framework 4.0	12 kwietnia 2010	rozszerzenia do programowania równoległego, bezpośrednie wsparcie dla języków IronRuby, IronPython, F#, platforma modelowania OSLO
.NET Framework 4.5	15 sierpnia 2012	wsparcie dla aplikacji Metro, biblioteka klas przenośnych pomiędzy platformami .NET, wiele innych pomniejszych zmian np.: tablice o wielkości większej niż 2GB, kompilacja JIT w tle w środowisku wieloprocesorowym
.NET Framework 4.6/4.6.2	20 lipca 2015 2 sierpnia 2016	wsparcie dla Windows 10, nowe funkcje krypptografi, ADO.NET, WPF, unowocześnienia w profilowaniu, obsługa HTTP/2

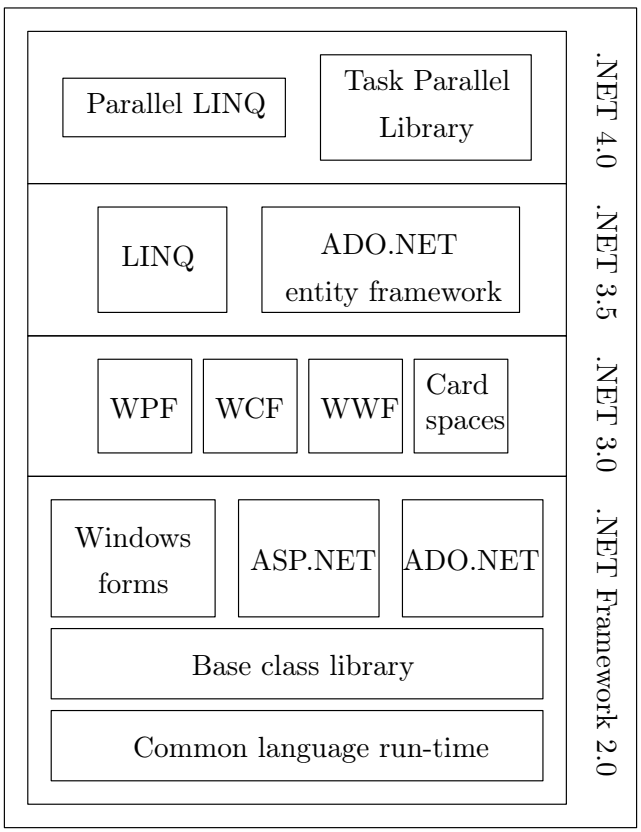
Ewolucja pakietu .NET

Kalendarium wydań platformy .NET:

Wersja	Data wydania	Nowe funkcje
.NET Framework 4.7	5 kwietnia 2017	Integracja z Windows 10 Creator Update, polepszania w obsłudze kryptografi, High-DPI, nowe API do wydruków WPF
.NET Framework 4.7.1/4.7.2	17 października 2017, 30 kwietnia 2018	dalsze usprawnienia i poprawki, wsparcie dla Server 2019
.NET Framework 4.8	18 kwietnia 2019	poprawki, ulepszone wsparcie dla urządzeń o wysokiej rozdzielczości
.NET 5.0	10 listopad 2020	unifikacja platform .NET (tj. Core i Framework), Blazor, wsparcie dla ARM, RyuJIT
.NET 6.0	08 listopad 2021	wersja LTS, nowa wersja środowiska Visual Studio 2022
.NET 7.0	08 listopad 2022	wersja bez długiego wsparcia
.NET 8.0	14 listopad 2023	wersja LTS
.NET 9.0	listopad 2024	wersja poglądowa
.NET 10.0	listopad 2025	wersja LTS

Przyszłe wydania .NET 8 i 9, są planowane na listopad 2023 i 2024.

Ewolucja pakietu .NET – schemat



Wersja 1.X wprowadziła podstawowe elementy platformy .NET, następne wersje dodają sukcesywnie nowe składowe, przy czym jako pierwszą pełnoprawną platformę należy traktować wydanie .NET od wersji 2.0.

.NET Core 1.0

Najnowsza odsłona platformy .NET wspierana przez społeczność oraz firmę Microsoft. Ta odmiana jest multiplatformowa (Windows, MacOS, Linux, Docker) oraz o otwartym kodzie kodzie źródłowym. Najważniejsze elementy to:

- CoreCLR, wieloplatformowe środowisko uruchomieniowe dla CLR, tj. maszyna wirtualna do uruchamiania programów .NET,
- kompilator JIT oraz nazwie RyuJIT,
- CoreFX, biblioteka klas bazowych oparta o podstawową bibliotekę FCL.

Dodatkowo .NET Core wspiera technologię ASP.NET Core oraz aplikacje uniwersalne (Universal Windows Platform), obecnie nie ma wsparcia dla Windows Forms (to zostało naprawione w wersji 5.0 .NET) oraz WPF. Pierwsza wersja .NET Core 1.0 została wydana 27 czerwca 2016.

Kolejne wydanie .NET Core

Kolejne główne wydania:

- NET Core 2.0 została wydana razem z Visual Studio 2017 15.3, i zawiera technologię ASP.NET Core 2.0, oraz Entity Framework Core 2.0. Kolejne uaktualnienia to .NET Core 2.1 oraz NET Core 2.2.
- .NET Core 3 wydana podczas konferencji Microsoft Build. Najważniejszy element w .NET Core 3 to wsparcie rozwoju aplikacji desktopowych, API dot. sztucznej inteligencji, uczenia maszynowego, oraz aplikacji IoT.
- kolejne planowane wydanie to .NET 5, 6.0 LTS. 7.0, 8.0 LTS, w cyklu rocznym.

Spis wydań .NET Core:

	Data wydania	Wersja VS	Ostatnia aktualizacja	Koniec wsparcia
.NET Core 1.0	2016-06-27	VS 2015 U3	1.0.16 2019-05-14	27.06.2019
.NET Core 1.1	2016-11-16	VS 2017 V15.0	1.1.13 2019-05-14	27.06.2019
.NET Core 2.0	2017-08-14	VS 2017 V15.3	2.0.9 2018-07-10	01.10.2018
.NET Core 2.1	2018-05-30	VS 2017 V15.7	2.1.16 (LTS) 2020-02-18	21.08.2021
.NET Core 2.2	2018-12-04	VS 2019 V16.0	2.2.8 2019-11-19	23.12.2019
.NET Core 3.0	2019-09-23	VS 2019 V16.3	3.0.3 2020-02-18	03.03.2020
.NET Core 3.1	2019-12-03	VS 2019 V16.4	3.1.2 (LTS) 2020-02-18	03.12.2022
.NET 5	2020-11-10	VS 2019 V16.8	5.0.3 2021-02-19	$\approx$ 02.2022

Trzy główne odmiany platformy .NET na rok 2013

- .NET Framework 4.5
- .NET Compact Framework
- .NET Micro Framework

.NET Framework 4.5

Główna platforma do rozwoju aplikacji dla systemów z rodziny Windows. Pozwala na budowę bezpiecznych programów z bogatym interfejsem użytkownika, wspiera także szeroki obszar technologii biznesowych.

Trzy główne odmiany platformy .NET na rok 2013

- .NET Framework 4.5
- .NET Compact Framework
- .NET Micro Framework

.NET Compact Framework

W przypadku urządzeń takich jak nowoczesne telefony komórkowe, urządzenia PDA, zasadniczym ograniczeniem są niewielkie zasoby. Środowisko .NET dla tego typu urządzeń jest niezależne od użytej platformy sprzętowej, ogólnie architektura jest identyczna z pełną wersją .NET. Ograniczenia to mniejsza ilość klas oraz obecność klas wyspecjalizowanych charakterystycznych dla urządzeń z ograniczonymi zasobami.

Trzy główne odmiany platformy .NET na rok 2013

- .NET Framework 4.5
- .NET Compact Framework
- .NET Micro Framework

.NET Micro Framework

Platforma MF została zaprojektowana specjalnie do urządzeń z ograniczonymi zasobami. Można ją uruchamiać na sprzęcie bez systemu operacyjnego, bowiem posiada dwa następujące poziomy:

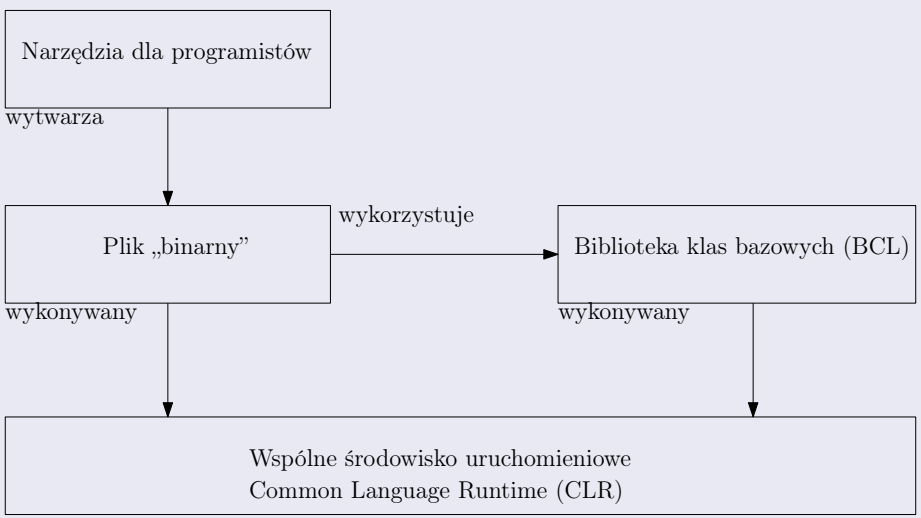
- Hardware Abstraction Layer (HAL) – ukrywa własności sprzętu,
- Platform Abstraction Layer (PAL) – wprowadza brakującą funkcjonalność w zależności od zastosowanego sprzętu,

Inne elementy to: CLR, biblioteki, aplikacje użytkownika.
Typowe wymagania dla MF to 200 – 500 KB (dla porównania CF wymaga 12MB), tego typu platforma znajduje zastosowanie w różnego rodzaju kontrolerach i innych małych urządzeniach.

Główne składowe platformy .NET

Trzy główne elementy platformy .NET:

- narzędzia dla programistów (Visual Studio, MonoDevelop, SharpDevelop),
- biblioteka klas bazowych (ang. Base Class Library – BCL),
- wspólne środowisko uruchomieniowe (ang. Common Language Runtime – CLR).

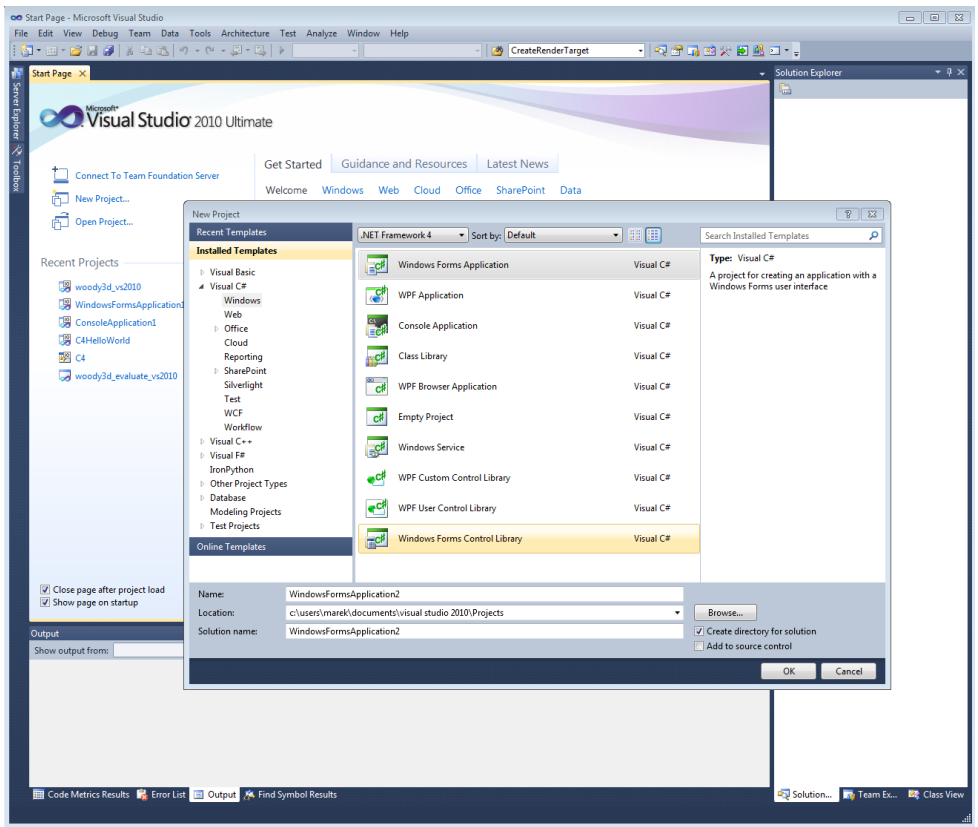


Microsoft Visual Studio w roku 2012

Trzy podstawowe odmiany środowiska Visual Studio to:

1. Microsoft Visual Studio 2012 Professional with MSDN,
2. Microsoft Visual Studio 2012 Premium with MSDN,
3. Microsoft Visual Studio 2012 Ultimate with MSDN.

a także Test Professional oraz Team Foundation Server. Główna zaleta to pełna i naturalna integracja z Platformą .NET.

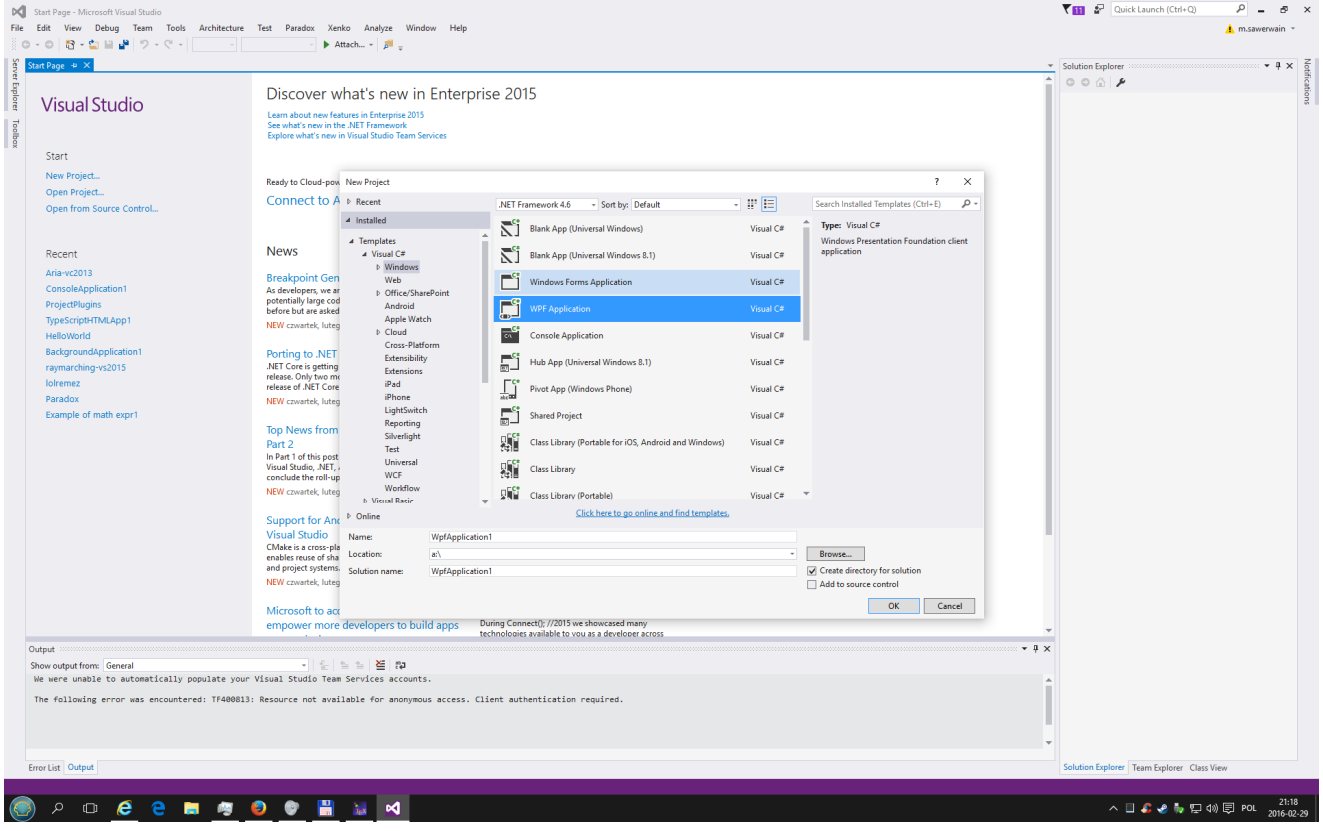


Microsoft Visual Studio w roku 2015/2016

Trzy podstawowe odmiany środowiska Visual Studio to:

1. Microsoft Visual Studio 2015 Community,
2. Microsoft Visual Studio 2015 Professional,
3. Microsoft Visual Studio 2015 Enterprise.

a także Test Professional oraz Team Foundation Server. Gówna zaleta to pełna i naturalna integracja z Platformą .NET, ale nie tylko np. wsparcie dla Androida.

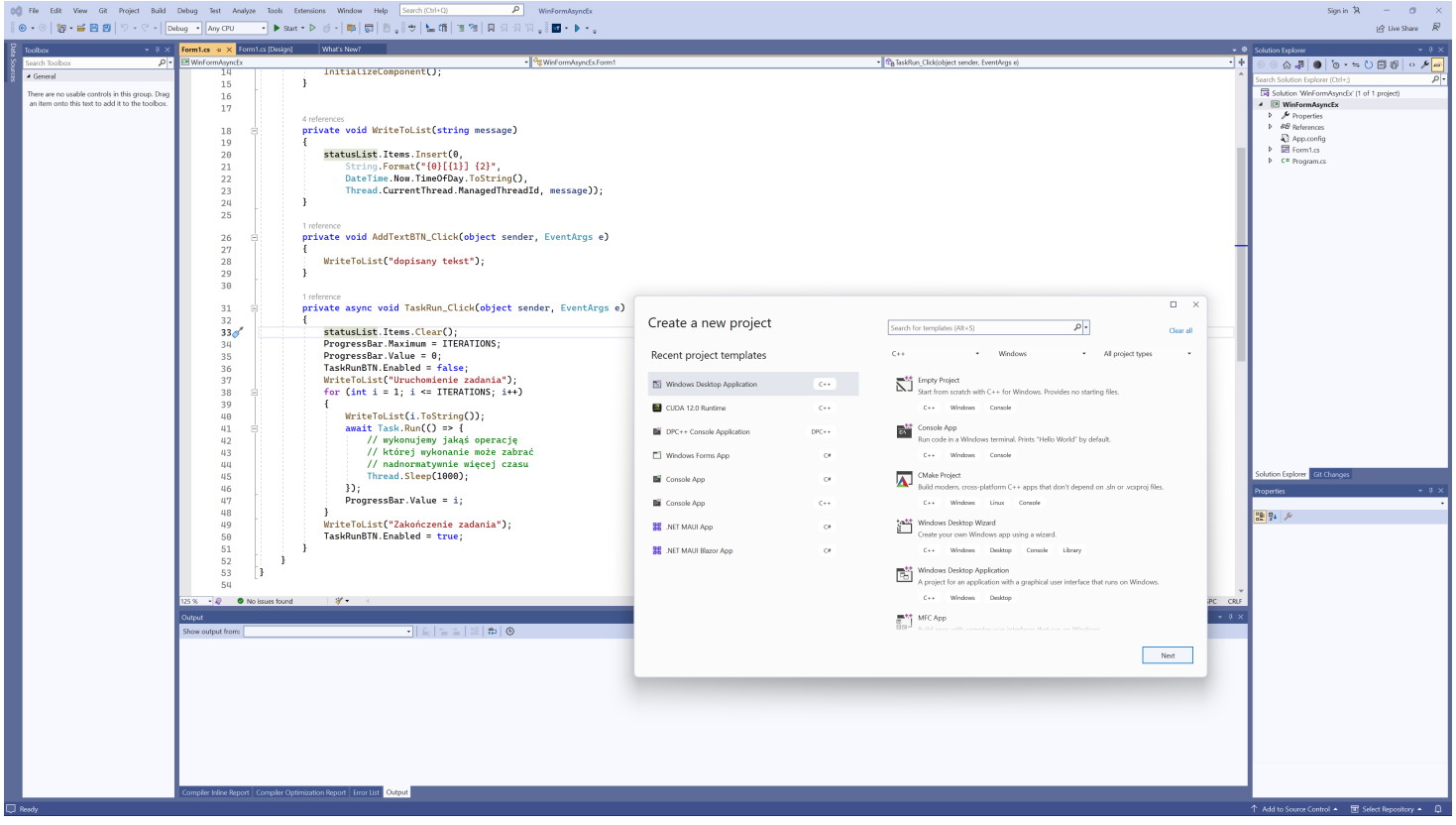


Microsoft Visual Studio w roku 2022/2023

Nadal obowiązują trzy podstawowe odmiany środowiska Visual Studio:

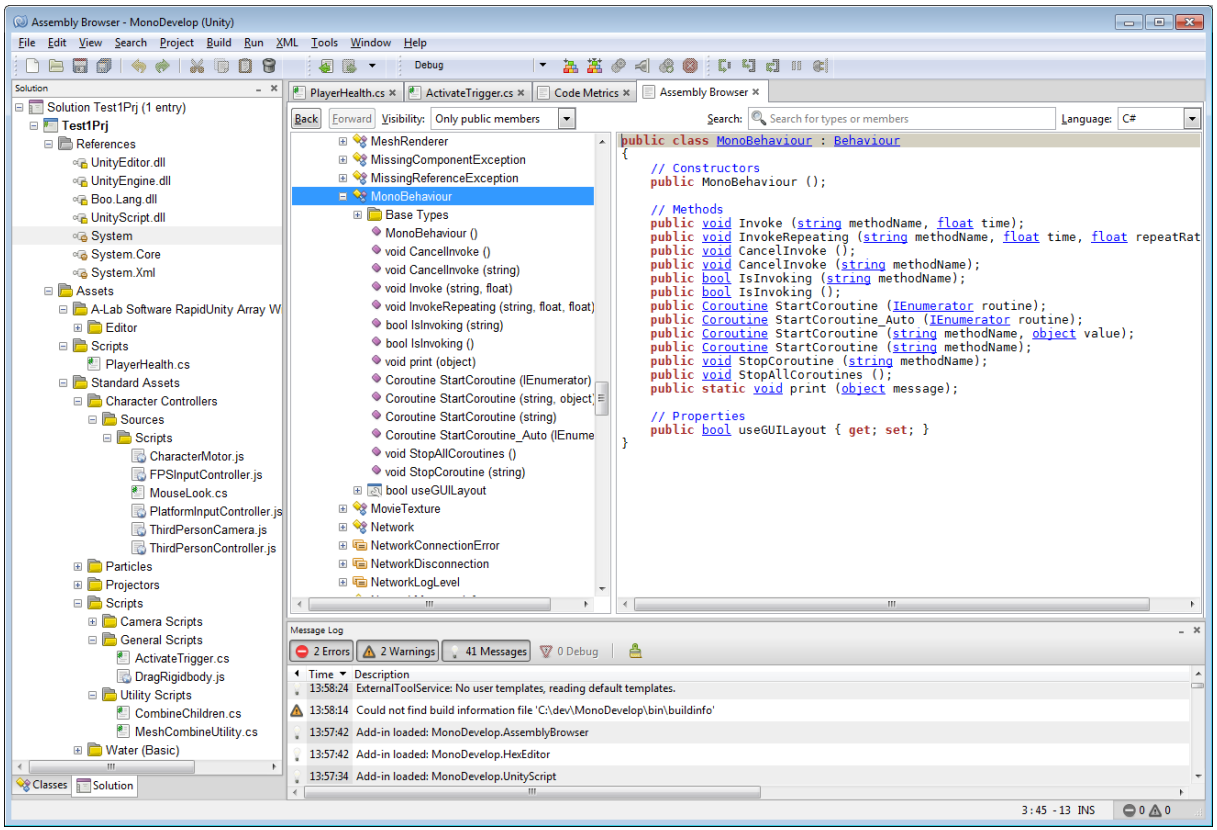
1. Microsoft Visual Studio 2022 Community,
2. Microsoft Visual Studio 2022 Professional,
3. Microsoft Visual Studio 2022 Enterprise.

Środowisko VS wspiera także inne technologie jak np. Python, system Linux, Android, IOS, technologie WEB.



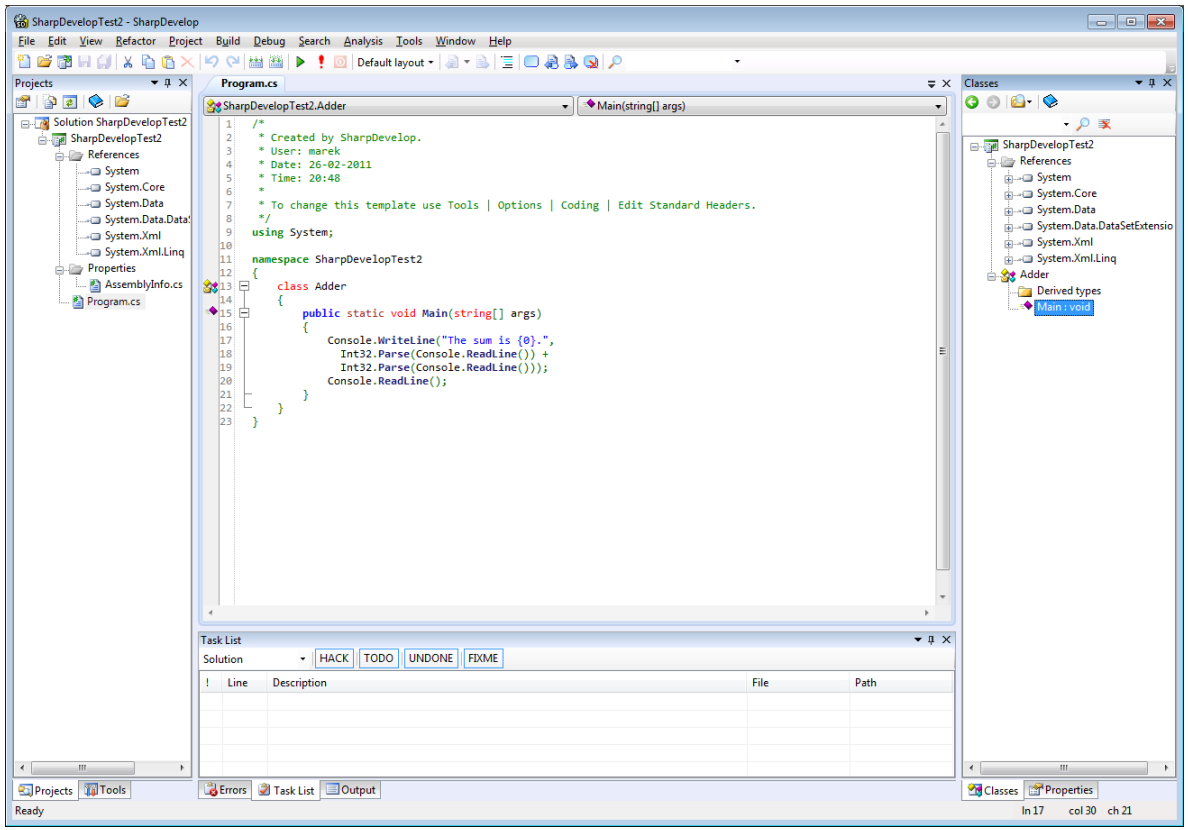
MonoDevelop

Narzędzie OpenSource, współpracujące z platformą MONO, zaletą jest wieloplatformowość, projekt można tworzyć i przenosić pomiędzy systemami Windows, Linux, MacOS.



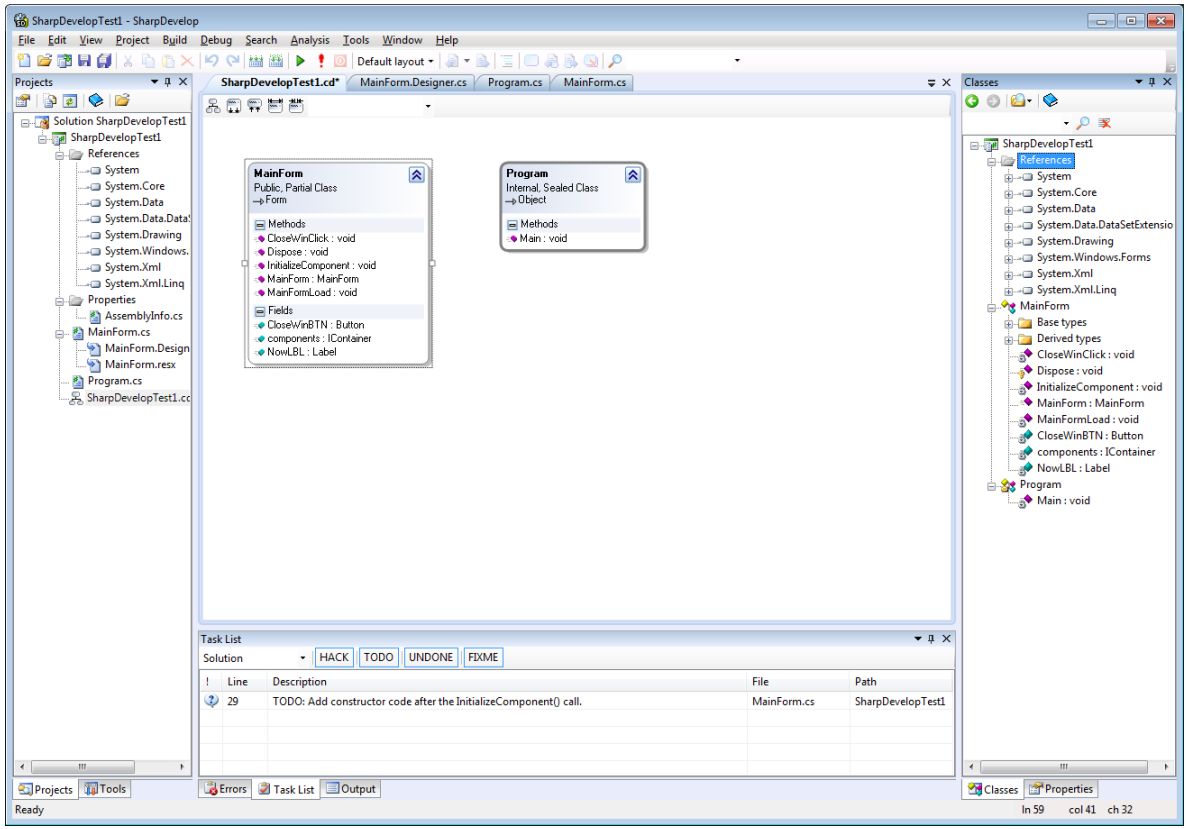
SharpDevelop

Darmowa (OpenSource) alternatywa dla Visual Studio, w przypadku nieco mniejszych projektów może konkurować z Visual Studio.



SharpDevelop

Darmowa (OpenSource) alternatywa dla Visual Studio, w przypadku nieco mniejszych projektów może konkurować z Visual Studio.



Biblioteka klas bazowych

Biblioteka klas bazowych: BCL – Base Class Library albo FCL – Framework Class Library:

1. klasy ogólnego zastosowania – obsługa plików, manipulacja ciągami znaków, szyfrowanie i bezpieczeństwo,
2. kolekcje – implementacja list, słowników, tablic otwartych i tablice bitów,
3. obsługa XML – tworzenie plików w standardzie XML, odczyt oraz zapis dokumentów XML, manipulacje zawartością dokumentów XML.

Poszczególne elementy biblioteki klas bazowych zostały pogrupowanie w trzech głównych przestrzeniach nazw:

1. **Accessibility** – ułatwia dostęp do COM,
2. **System** – zawiera podstawowe/fundamentalne klasy reprezentujące typy i dane, zdarzenia, wyjątki i inne konstrukcje charakterystyczne dla platformy .NET,
3. **Microsoft** – obsługa elementów charakterystycznych dla systemu Microsoft.

Graficzna ilustracja biblioteki klas bazowych

System.Web

Services		UI	
	Decription		HtmlControls
	Discovery		MobileControls
	Protocols		WebControls
Caching		Security	
Configuration		SessionState	

System.Windows.Forms

Design	Component Model

System.Drawing

Drawing2D	Printing
Imaging	Text

System.Data

DataSet	Lineq.SqlClient
DataTable	Objects

System.Xml

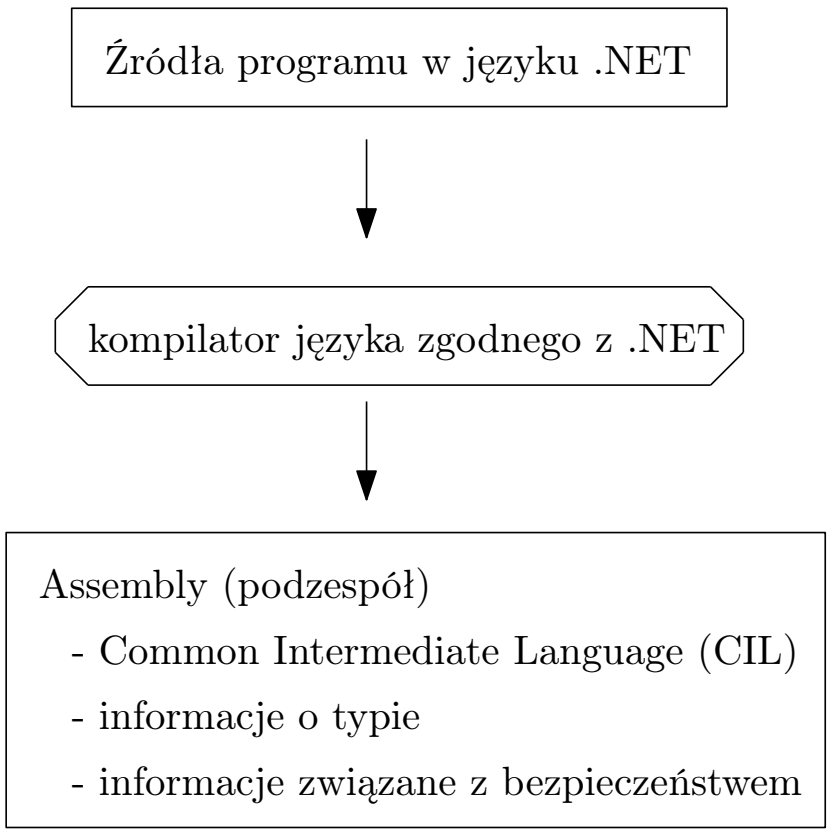
XmlReader	XmlDictionary
XmlWriter	XmlException

Collections	IO	Security	Runtime
Configuration	Net	ServicesProcess	InteropServices
Diagnostics	Reflection	Text	Remoting
Globalization	Resources	Threading	Serialization

Kompilacja do kodu pośredniego

Kompilacja do kodu pośredniego:

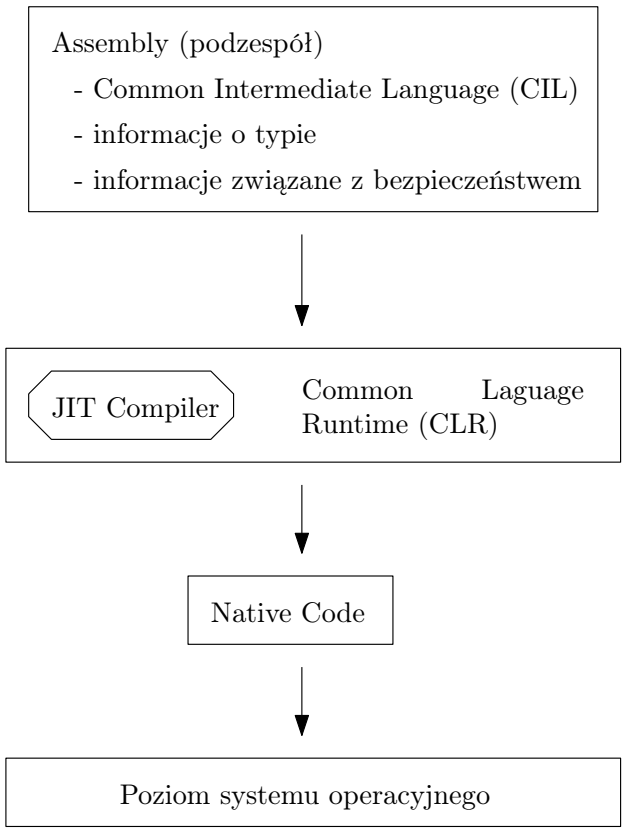
1. podzespół (assembly) to pliki typu EXE oraz DLL,
2. kod zapisany w podzespole nie jest kodem natywnym ale kodem pośrednim (CIL),
3. podzespół zawiera trzy główne elementy:
   1. kod CIL,
   2. metadane o typach,
   3. metadane o użytych innych podzespołach:



Kompilacja do kodu maszynowego

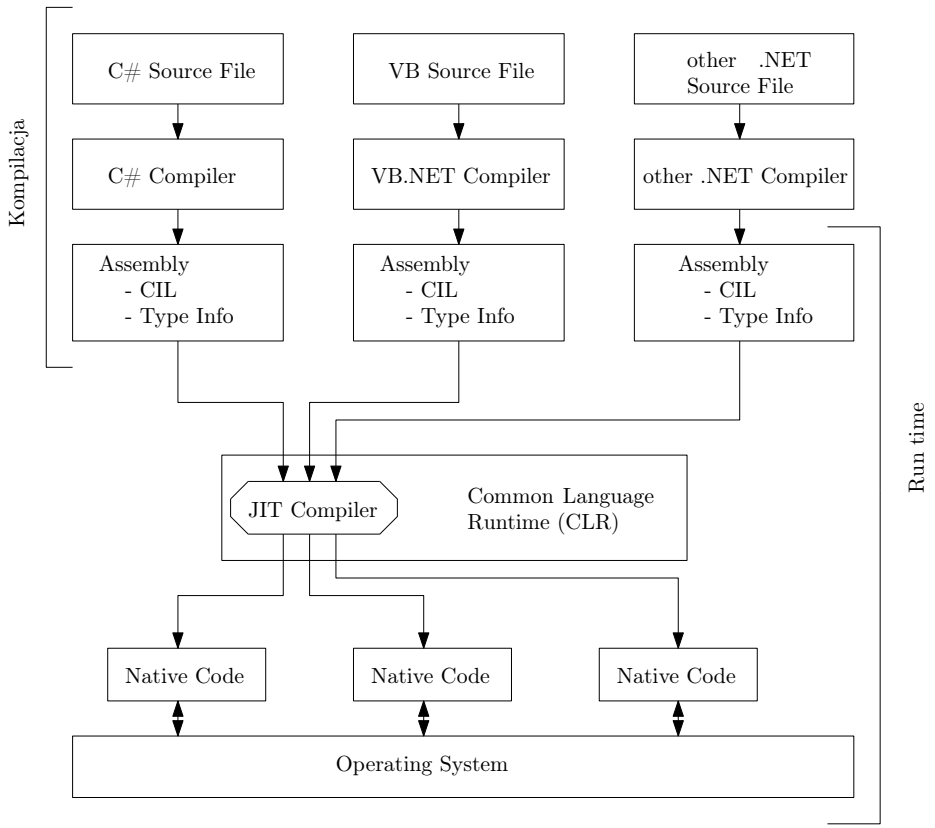
Kompilacja do kodu maszynowego nie odbywa się podczas procesu kompilacji ale podczas uruchamiania podzespołu:

1. sprawdzane są warunki bezpieczeństwa wykonania podzespołu,
2. alokacja pamięci,
3. kod CIL jest przekazywany do kompilatora JIT



Obecność kodu JIT oraz CIL oznacza istnienie dwóch pojęć: kod zarządzany (managed code) wykonywany przez wspólne środowisko uruchomieniowe oraz kod niezarządzany (unmanaged code) odnoszący się do bezpośrednio do systemu operacyjnego.
Istnieje także narzędzie Native Image Generator (ngen), tworzący kod natywny, wtedy kompilator JIT nie jest stosowany (w nowszych odsłonach .NET ngen to usługa kompilacji kodu CIL).

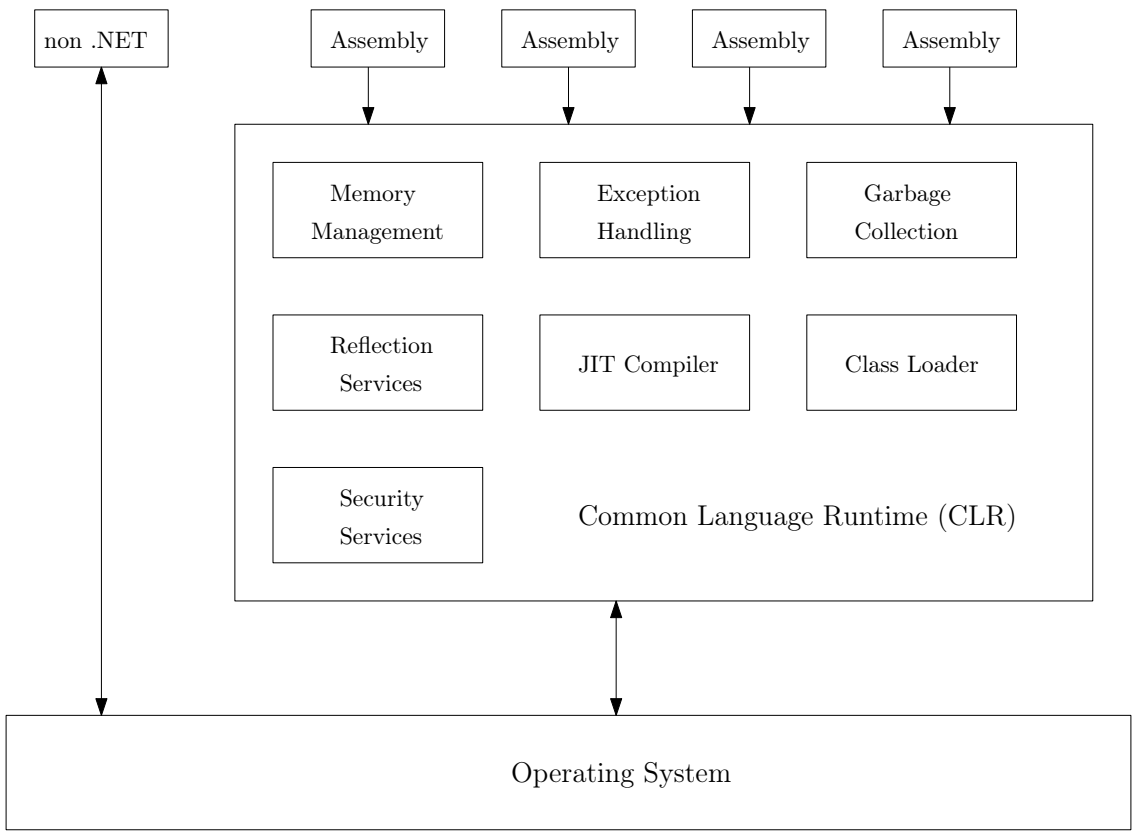
Wspólne środowisko wykonawcze



Środowisko uruchomieniowe dla różnych języków .NET

Wspólne środowisko uruchomieniowe .NET, to główny komponent platformy .NET oferuje trzy główne usługi:

1. automatyczne zarządzanie pamięcią,
2. bezpieczeństwo,
3. wsparcie dla biblioteki klas bazowych, usług sieciowych, usług bazodanowych.



Wspólna infrastruktura językowa

Common Language Infrastructure (CLI), czyli wspólna infrastruktura językowa, to zbiór standardów pozwalających na połączenie komponentów .NET we wspólną i spójną całość, bez względu na stosowany język programowania:

Common Language Runtime (CLR)	Common Language Specification (CLS)
Base Class Library (BCL)	Metadata Definition and Semantics
Common Type System (CTS)	Common Intermediate Language (CIL) Instruction Set

Główne usługi dostępne w .NET

(1) ASP.NET – obsługa aplikacji WEB, od strony interfejsu użytkownika po logikę biznesową,

(2) ADO.NET – dostęp do danych oraz usług bazodanowych,

(3) CardSpace – zabezpiecza oraz składuje cyfrowe identyfikatory,

(4) Entity Framework – zarządzanie bytami, czyli bardziej abstrakcyjne podejście do zarządzania danymi,

(5) WEB Services – tworzenie usług których funkcjonalność może być łatwo udostępniona poprzez sieć,

(6) Windows Forms – formularze, okna dialogowe, elementy graficznego interfejsu użytkownika,

(7) Windows Communication Framework (WCF) – wprowadza możliwość komunikacji za pomocą komunikatów przekazywanych pomiędzy komponentami,

(8) Windows Presentation Framework (WPF) – obsługa interfejsu użytkownika, wprowadzono nową metodologię rozwoju, rozdzielając zadania programistów od zadań projektantów interfejsu użytkownika,

(9) Workflow Foundation (WF) – ogólna obsługa procesów sterowania, a w szczególności procesów sekwencyjnych oraz procesów wyrażonych w postaci maszyny stanów.

Wydaje się, że główne zalety platformy .NET to min.:

1. bezpieczna wielojęzykowa platforma rozwoju aplikacji,
2. wsparcie dla nowoczesnych technologii budowy interfejsu użytkownika (WPF, Silverlight, i nowe rozwiązania jak Blazor),
3. bogate wsparcie dla aplikacji WEB (ASP.NET),
4. wspieranie tworzenia usług WEB, AppFabric,
5. obsługa procesów biznesowych (WF),
6. elastyczny dostęp do danych ADO.NET.

Dwa przykłady

Dwa przykłady prostych programów dla konsoli opracowane w językach: C# oraz Nemerle:

1. suma dwóch liczb całkowitych,
2. zliczanie linii w plikach tekstowych,
3. funkcja silnia.

Suma liczb całkowitych – C#

```
using System;
class Adder {
    public static void Main(string[] args) {
         Console.WriteLine("The sum is {0}.",
            Int32.Parse(Console.ReadLine()) +
            Int32.Parse(Console.ReadLine()));
        Console.ReadLine();
    }
}
```

Suma liczb całkowitych – Nemerle

```
using System;
public class Adder {
    public static Main () : void {
        Console.WriteLine ("The sum is {0}.",
            Int32.Parse (Console.ReadLine ()) +
            Int32.Parse (Console.ReadLine ()));
        _ = Console.ReadLine();
 }
}
```

Zliczanie linii – C#

```
class RecurenceLineCounter {
    static int line_no;
    static System.IO.StreamReader sr;

    static void Main(string[] args) {
        line_no = 0;
        sr = new System.IO.StreamReader("plik.txt");

        read_lines();
        System.Console.WriteLine("Line count: {0}", line_no);
    }

    static void read_lines() {
        String line = sr.ReadLine();
        if(line != null) {
            System.Console.WriteLine( line );
            line_no = line_no + 1;
            read_lines();
        }
    }
}
```

Zliczanie linii – Nemerle

```
class LineCounterWithoutLoop {
  public static Main () : void {
    def sr = System.IO.StreamReader ("file-name.txt");
    mutable line_no = 0;

    def read_lines () : void {
      def line = sr.ReadLine ();
      when (line != null) {
        System.Console.WriteLine (line);
        line_no = line_no + 1;
        read_lines ()
      }
    };

    read_lines ();
    System.Console.WriteLine ("Line count: {0}", line_no);
  }
```

Znana i lubiana funkcja silnia

```
using Nemerle.Collections;
using Nemerle.Text;
using Nemerle.Utility;
using System;
using System.Collections.Generic;
using System.Console;
using System.Linq;

module Program {

  Main() : void {
    def FactorialWithAcc(n, acc) {
        | (0, _)
        | (1, _) => acc
        | _      => FactorialWithAcc(n - 1, n * acc)
    }

    def Factorial = FactorialWithAcc(_, 1);

    WriteLine("5! = {0}", Factorial(5) );
    _ = ReadLine();
  }
}
```

W następnym tygodniu miedzy innymi

Wykład: Składowe platformy .NET: CLR, CTS, języki programowania, biblioteki klas, pojęcie podzespołu (ang. assembly),

1. analiza składowych platformy .NET,
2. przegląd języków programowania .NET,
3. przedstawienie CLR, CTS,
4. analiza biblioteki klas,
5. budowa podzespołu,
6. język pośredni (ang. Intermediate Language – IL).

Proponowane tematy prac pisemnych:

1. platforma informatyczna wczoraj, dziś i jutro (i pojutrze),
2. porównanie platformy .NET oraz JAVA,
3. porównanie maszyny wirtualnej platformy .NET oraz platformy JAVA.

Dziękuje za uwagę!!!