.NET
WPROWADZENIE

Dr hab. inż. Krzysztof Zatwarnicki,
prof. PO

Wprowadzenie

- **Co to jest .NET Framework**
- **Microsoft Visual C#**

.NET Framework

- **Środowisko do budowania i kompilowania aplikacji**

- **Wspólne Środowisko Uruchomieniowe CLR(Common Language Runtime)**

Common language runtime

- **Każdy język podlegający specjalizacji CLS (Common Language Specification) może zostać uruchomiony w CLR.**

- **Aplikacje mogą być napisane w różnych językach programowania i nadal ze sobą współpracować**

Common Language Specification

W .NET Framework Microsoft daje możliwość współpracy między następującymi językami:

- **Microsoft Visual Basic**
- **Microsoft Visual C#**
- **Microsoft Visual C++**
- **Microsoft J#**

Uwzględnia się również możliwość współpracy z innymi językami (według Microsoft jest to około 20 języków)

Visual C# .NET

- **C# to jedyny język programowania Microsoft, który został zaprojektowany od samego początku specjalnie dla platformy .NET** i wspólnego środowiska uruchomieniowego CLR. Chociaż CLR obsługuje wiele języków, to tylko C# był projektowany równolegle z CLR. Te dwie technologie miały na siebie duży wpływ, przez co C# świetnie nadaje się do pisania kodu zarządzanego. Kod zarządzany komponentów platformy .NET, takich jak biblioteki klas i środowisko programistyczne ASP.NET, został napisany właśnie w języku C#.

- **C# jest znacznie prostszym językiem niż C++,** jednak — jak sama nazwa wskazuje — należy do rodziny języków C. Oznacza to, że ma wiele cech wspólnych z C/C++, których nie mają języki takie jak Visual Basic. Na przykład C# rozróżnia wielkie i małe litery, a VB nie. C# wymaga od programistów jawnej konwersji pomiędzy typami danych, a Visual Basic dokonuje niektórych konwersji automatycznie. Składnia języka C# jest podobna do składni języków C++ i Java. C# ma jednak w stosunku do C++ kilka dodatkowych cech obiektowych, takich jak właściwości, atrybuty, delegaty czy zdarzenia.

Visual Basic .NET

- Visual Basic jest uważany za najbardziej popularny język programowania aplikacji dla Windows. W wersji Visual Basic .NET wprowadzono do języka wiele zmian. Zmiany objęły między innymi sposób deklaracji zmiennych i funkcji, sposób tworzenia i usuwania obiektów, domyślny sposób przekazywania parametrów funkcji, sposób wywoływania procedur. Największe zmiany dotyczą chyba sposobu obsługi błędów — została usunięta stosowana do tej pory obsługa błędów, często nazywana „on error goto hell”. Visual Basic .NET w pełni wspiera strukturalną obsługę wyjątków.

- Dzięki możliwości współdziałania platformy .NET Framework ze starszymi technologiami, kod napisany w Visual Basic .NET może wywoływać istniejący kod, napisany w starszych wersjach języka Visual Basic (i odwrotnie), dzięki czemu aplikacje na platformie .NET mogą wykorzystywać starsze moduły aplikacji.

- Na platformę .NET przeniesiono jedynie główną odmianę języka Visual Basic.

- Nie został przeniesiony język skryptowy:

  - Visual Basic Scripting Edition (znany pod nazwą VBScript), wykorzystywany do tworzenia skryptów administracyjnych, stron ASP i dynamicznej zawartości stron internetowych
  - Visual Basic for Applications (VBA) — język skryptowy wykorzystywany do pisania makr dla aplikacji rodziny Office.

Visual C++ .NET

- Visual C++ to bardzo popularny język programowania, nadający się zarówno do tworzenia kodu niskiego poziomu, jak i do pisania aplikacji dla Windows. W podstawowym języku C++ można pisać kod niezarządzany.

- W Visual C++ .NET wprowadzono obsługę nowych słów kluczowych i typów danych (zwanych rozszerzeniami *Managed Extentions* dla Visual C++), które umożliwiają pisanie kodu zarządzanego oraz tworzenie aplikacji w pełni korzystających z możliwości platformy .NET Framework.

- Kompilator Visual C++ .NET nadal jest kompilatorem natywnym, co - wraz z zapewnianymi przez CLR możliwościami łączenia nowego kodu zarządzanego z istniejącym kodem niezarządzanym - pozwala programistom C++ nadal stosować dokładnie ten sam język i środowisko, którego używali do tej pory.

- Dodatkowo w Visual Studio .NET 2003, programiści C++, wykorzystujący rozszerzenia Managed Extensions do tworzenia aplikacji na platformę .NET, mogą stosować te same kreatory interfejsu użytkownika, których używają programiści języków Visual Basic i C#. Wcześniej programiści C++ musieli korzystać z biblioteki MFC lub pisać interfejsy użytkownika ręcznie, co trwało dużo dłużej i było związane z większą liczbą błędów.

Visual J# .NET

- Visual J# .NET to nowy język, umożliwiający programistom języka Visual J++ przejście na platformę programistyczną .NET. J# - o składni charakterystycznej dla języka Java - umożliwia tworzenie programów, które korzystają z bibliotek klas środowiska .NET Framework oraz wspólnego środowiska uruchomieniowego CLR.

- Język J# zapewnia także narzędzia służące do importowania i konwersji kodu źródłowego istniejących aplikacji J++, pozwalające uruchamiać te aplikacje pod kontrolą wspólnego środowiska uruchomieniowego CLR. Dzięki funkcjonalności współpracy z COM wbudowanej w CLR, aplikacje napisane w J# mogą korzystać z istniejących bibliotek J++. Język Visual J# .NET w żaden sposób nie korzysta z technologii Java firmy Sun, tak więc aplikacji napisanych w tym języku nie da się łatwo przenieść na platformę Sun Java.

Inne języki

Inne języki platformy .NET Framework

- APL
- Fortran
- Pascal
- Haskell
- Scheme
- Curriculum
- Mondrian
- Perl
- Python
- COBOL
- Microsoft JScript®
- SmallTalk
- Eiffel
- Oberon
- RPG
- Component Pascal
- Merkury
- Nemerle
- Standard ML
- Forth
- Oz

Środowisko uruchomieniowe CLR (Common Language Runtime)

- ***Wspólne środowisko uruchomieniowe*** (*Common Language Runtime*, w skrócie *CLR*) to podstawa całego systemu .NET Framework.

- Wszystkie języki środowiska .NET (na przykład C# czy Visual Basic .NET), a także wszystkie biblioteki klas obecne w .NET Framework (ASP.NET, ADO.NET i inne) oparte są na CLR.

- Ponieważ nowe, tworzone przez Microsoft oprogramowanie, także oparte jest na .NET Framework, każdy, kto chce korzystać ze środowiska Microsoft, prędzej czy później będzie musiał zetknąć się z CLR.

Kod zarządzany

- Kompilatory zgodne z CLR zamieniają kod źródłowy aplikacji na kod wykonywalny, zapisany w standardowym języku pośrednim **MSIL (**Microsoft Intermediate Language **)**, oraz na ***metadane*** — informacje na temat kodu wykonywalnego oraz danych wykorzystywanych przez ten kod.

- Niezależnie od języka, w którym napisany jest kod źródłowy aplikacji, kompilator zamienia wszystkie operacje na typach danych, to jest klasach, strukturach, liczbach całkowitych, łańcuchach znaków — na język MSIL i metadane.

Kod zarządzany

- W czasie wykonywania aplikacji, CLR tłumaczy kod MSIL na kod maszynowy (natywny) procesora, na którym wykonywana jest aplikacja.

- Konwersja kodu aplikacji z MSIL na kod maszynowy daje możliwość zarządzania wykonywaniem aplikacji, co pozwala uniknąć wielu problemów — stąd nazwa ***kod zarządzany***.

Microsoft Intermediate Language MSIL

- MSIL to kod dość podobny do zestawu instrukcji procesora. Obecnie nie istnieje jednak żaden sprzęt, który mógłby bezpośrednio wykonywać kod MSIL (nie jest jednak wykluczone, że w przyszłości taki sprzęt powstanie).

- Na razie kod MSIL musi być tłumaczony na język maszynowy procesora, na którym ma być uruchomiony.

MSIL

- Zaletą kodu pośredniego jest potencjalna przenośność aplikacji.

- Przeniesienie całego .NET Framework na inne systemy operacyjne lub inne procesory jest nie lada wyzwaniem, jednak Microsoft stara się, by na innych urządzeniach dostępne było co najmniej środowisko **.NET Compact Framework (.NET CF)**.

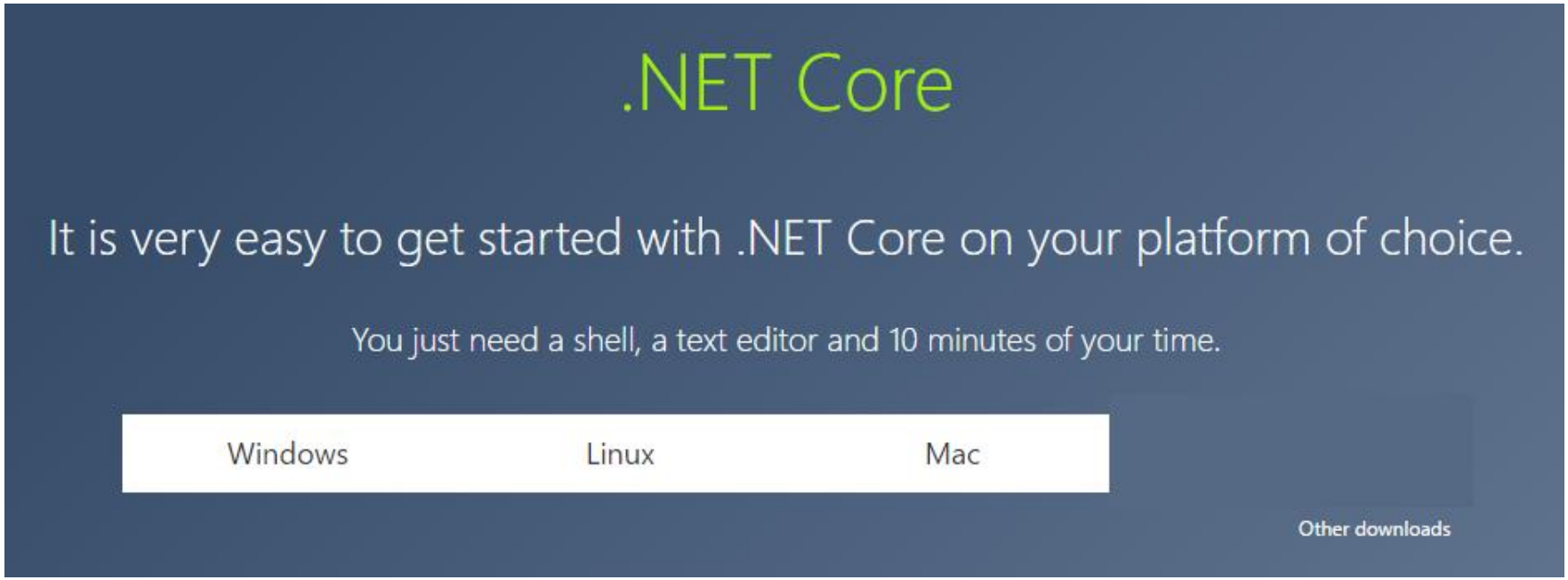
Nadchodzi nowe

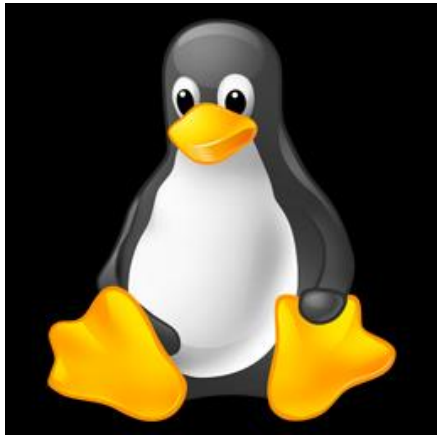
- Na początku maja 2016 pojawił się:

.NET CORE

- To nowa wersja framework'a, która z założenia ma być wieloplatformowa oraz open-source.
- To po prostu pakiet NuGet. Doskonale nadaje się to do aplikacji webowych.
- Wchodząc trochę w szczegóły techniczne, .NET Core składa się z podstawowej biblioteki klas (CoreFx), oraz środowiska uruchomieniowego CoreClr.

.NET CORE wieloplatformowe

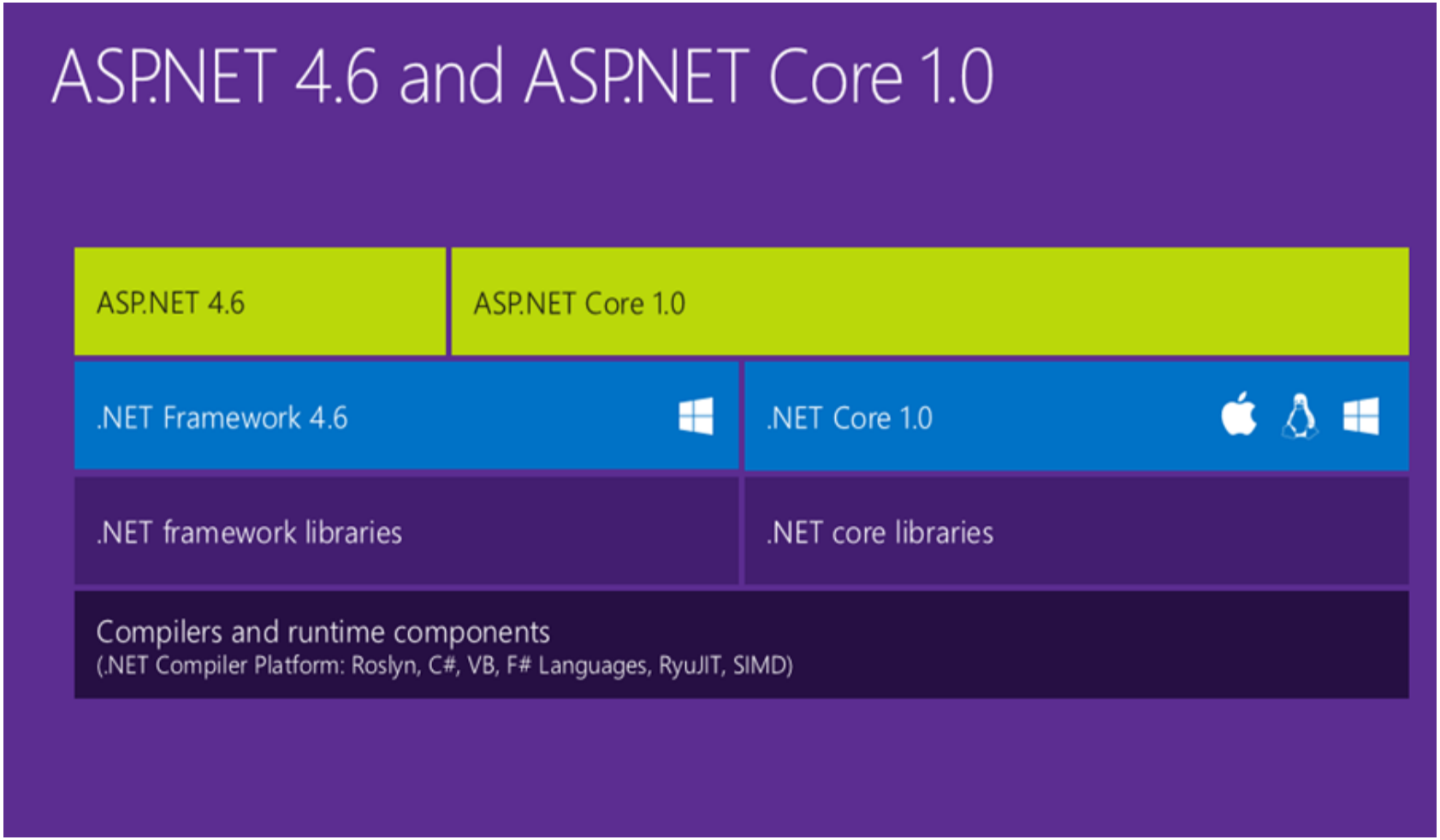




.NET CORE

- Wprowadzenie .NET Core nie oznacza, że nagle będziemy mogli uruchamiać wszystkie aplikacje na Linux.
- .NET Core to jedynie podzbiór .NET Framework.
- Na .NET CORE działać będą aplikacje ASP.NET.
- Aplikacje typu WPF nie będą dostępne.

.NET CORE


ASP.NET 4.6 and ASP.NET Core 1.0
ASP.NET 4.6
ASP.NET Core 1.0
.NET Framework 4.6
.NET Core 1.0
.NET framework libraries
.NET core libraries
Compilers and runtime components
(.NET Compiler Platform: Roslyn, C#, VB, F# Languages, RyuJIT, SIMD)

Metadane

- Kompilacja kodu zarządzanego, oprócz wygenerowania kodu pośredniego MSIL, powoduje utworzenie metadanych opisujących powstały kod.
- Metadane to szczegółowy opis typów zdefiniowanych w kodzie zarządzanym, z którym są związane.
- Opis ten przechowywany jest w tym samym pliku, w którym znajduje się kod MSIL.

Metadane opisują typy znajdujące się w danym fragmencie kodu. Informacje te zawierają:

- nazwę typu,
- zasięg typu (publiczny lub w granicach podzespołu),
- nazwę typu, po którym dziedziczy opisywany typ,
- implementowane interfejsy,
- implementowane metody,
- udostępniane właściwości,
- obsługiwane zdarzenia.

Podzespoły 1

- Na kompletną aplikację często składa się wiele różnych plików. Niektóre z nich zawierają kod wykonywalny, a inne zawierają zasoby (na przykład grafikę lub treść komunikatów).

- W aplikacjach opartych na .NET Framework pliki, które stanowią jeden logiczny moduł, udostępniający określoną funkcjonalność, grupowane są w *podzespoły* (*assembly*).

Podzespoły 2

- Większość podzespołów to pojedyncze pliki DLL.
- Niezależnie od tego, czy podzespół składa się z wielu plików, czy też jest to pojedynczy plik, podzespół jest logiczną, niepodzielną całością. Granice podzespołu wyznaczają granice zasięgu zdefiniowanych w nim typów danych.

Wykonywanie kodu zarządzanego 1

- Gdy uruchamiana jest aplikacja wykorzystująca środowisko .NET Framework, podzespoły, składające się na tę aplikację, muszą zostać odnalezione i załadowane do pamięci.

- Podzespoły nie są ładowane tak długo, aż będą potrzebne — jeśli aplikacja nie wywoła żadnej metody z danego podzespołu, nie zostanie on załadowany do pamięci.

Wykonywanie kodu zarządzanego 2

- Potrzebne podzespoły muszą zostać odnalezione w systemie plików.
- Pierwszym miejscem, w którym CLR szuka podzespołów, jest *globalna pamięć podręczna podzespołów* (*Global Assembly Cache — GAC*).
- GAC jest to specjalny katalog, w którym przechowywane są podzespoły wykorzystywane przez więcej niż jedną aplikację.

Wykonywanie kodu zarządzanego 3

- Po załadowaniu potrzebnych podzespołów do pamięci, kod aplikacji nadal jest kodem MSIL, który nie może być bezpośrednio wykonywany przez procesor.

- Potrzebna jest jeszcze jedna kompilacja, która dokona zamiany kodu MSIL na kod maszynowy procesora, na którym uruchomiona będzie aplikacja.

Kompilacja 1
just-in-time compilation — JIT

- Najczęściej stosowaną metodą kompilacji kodu MSIL na kod natywny jest załadowanie przez CLR podzespołu do pamięci, a następnie kompilacja każdej metody w momencie pierwszego jej wywołania. Ponieważ każda metoda kompilowana jest tylko w momencie pierwszego uruchomienia, proces kompilacji nazywa *się kompilacją w samą porę* (*just-in-time compilation — JIT*).

- Kompilacja JIT umożliwia kompilowanie tylko tych metod, które są rzeczywiście wykorzystywane.

Kompilacja 2
Native Image Generator (NGEN)

- Inną metodą kompilacji jest wygenerowanie całego kodu binarnego danego podzespołu z użyciem narzędzia Native Image Generator (NGEN), dostępnego w .NET Framework SDK.

- Narzędzie *ngen.exe* kompiluje cały podzespół i umieszcza jego kod maszynowy w obszarze zwanym *pamięcią podręczną obrazów kodu natywnego (Native Image Cache)*.

Narzędzia programisty

Visual Studio Community

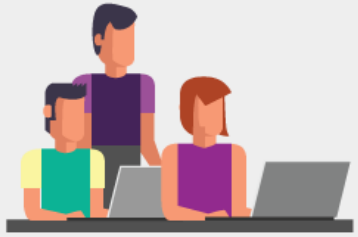
Bezpłatne, w pełni funkcjonalne i rozszerzalne narzędzie dla deweloperów tworzących aplikacje w środowiskach innych niż przedsiębiorstwa

Dowiedz się więcej >

Co myślisz o tej cenie?

Bezpłatnie [1]

Bezpłatna wersja Community



Visual Studio Professional

Profesjonalne narzędzia i usługi dla indywidualnych deweloperów i małych zespołów

Dowiedz się więcej >

Cena od

45 USD/miesiąc

Professional

Visual Studio Enterprise

Rozwiązanie klasy biznesowej z zaawansowanymi funkcjami dla zespołów pracujących nad projektami o dowolnym rozmiarze i poziomie złożoności, łącznie z zaawansowanymi funkcjami testowania i infrastrukturą DevOps

Dowiedz się więcej >

Cena od

250 USD/miesiąc

Enterprise

Narzędzia programisty
Visual Studio Code

Program Visual Studio Code jest lekkim, ale potężnym edytorem kodu źródłowego .NET, który jest dostępny dla systemów Windows, OS X i Linux.

Inne narzędzia programistyczne

- **SharpDevelop**
- is a free and open source IDE for the C#, Visual Basic .NET (VB.NET), Boo, and (starting from version 3.0) F# and IronPython programming languages.
- It is typically used as an alternative to Microsoft's Visual Studio .NET. Early in its development there was a fork to Mono/Gtk# called MonoDevelop which includes multi-platform support.

Inne narzędzia programistyczne

- **DevForce**
- is an application server and development framework for the Microsoft .NET platform. It provides a complete framework for building enterprise applications in .NET.

Inne narzędzia programistyczne

- **Turbo Delphi for .NET Explorer**

- Darmowe środowisko programistyczne oparte na języku Object Pascal (Delphi)

- Narzędzie umożliwia wykorzystanie języka *Object Pascal* (*Delphi*) do tworzenia aplikacji **WinForms**, **VCL.NET**, **ASP.NET** i **usług sieciowych .NET**. Dodatkowo możliwe jest budowanie zaawansowanych rozwiązań klient-serwer *.NET Remoting* oraz nowych komponentów *WinForms*, *VCL.NET* i *ASP.NET*.

Projekty związane z .NET

- **DotGNU Project**
- The DotGNU project aims to be for webservices and for C# programs what GNU/Linux is rapidly becoming for desktop and server applications: the industry leader and provider of Free Software solutions

- **Mono**
- Mono is a software platform designed to allow developers to easily create cross platform applications. Sponsored by Novell (http://www.novell.com/), Mono is an open source implementation of Microsoft's .NET Framework based on the ECMA standards for C# and the Common Language Runtime. A growing family of solutions and an active and enthusiastic contributing community is helping position Mono to become the leading choice for development of Linux applications

Projekty związane z .NET

- **.NET Compact Framework**

- .NET Compact Framework to uproszczona wersja .NET Framework, dostosowana do pracy na urządzeniach przenośnych i innych urządzeniach, działających pod kontrolą systemu operacyjnego Windows Embedded.

- Platforma .NET Compact Framework nie jest dostarczana wraz z .NET Framework SDK, ale jest dostępna w pakiecie Visual Studio .NET 2008. Dzięki niej programiści mogą wykorzystywać jeden zestaw narzędzi oraz bibliotek API do tworzenia aplikacji dla całej gamy urządzeń - począwszy od komputerów kieszonkowych, poprzez stacje typu thin client, po stacje robocze oraz serwery.

Inne narzędzia

- .NET DiscUtils
- **A**
- Aggiorno
- **B**
- Baltie
- Base One Foundation Component Library
- BugTracker.NET
- **C**
- CLR Profiler
- Castle Project
- CodeRush
- CodeIt.Right
- **D**
- DataObjects.NET
- Designbox
- DevForce
- Dotfuscator
- **E**
- EntitySpaces
- **E cont.**
- Euss
- **F**
- FxCop
- **G**
- Gemini (issue tracking system)
- **L**
- Linqpad
- **M**
- MonoDevelop
- Moq
- MyGeneration
- **N**
- NAnt
- NClass
- NDepend
- NHibernate
- Neo (object-relational toolset)
- **O**
- OpenRasta
- **P**
- Phalanger (compiler)
- **R**
- ReSharper
- RealmForge
- .NET Reflector
- Runtime Intelligence Service
- **S**
- Saturn Disk Image
- SharpDevelop
- StyleCop
- SubSonic (software)
- **V**
- VTD-XML
- Visual Assist X
- **X**
- XAMLPad
- XSP (softwar

.NET Biblioteka klas

Poza podstawowymi klasami, biblioteka klas .NET oferuje kilka unikalnych klas jak:

- **ADO.NET**
- **ASP.NET**
- **XML Web Services**
- **Interfejsy użytkownika**

ADO.NET

- **Jest to nowa generacja ActiveX**
- **Klasa ADO.NET pozwala na interakcję z danymi pozyskanymi przez interfejsy OLE DB, ODBC, Oracle, i SQL Server w formie XML. Klasy XML pozwalają na manipulacje danymi, przeglądanie ich i tłumaczenie.**

ASP.NET

Jest używana do budowania internetowych aplikacji

- **Do tworzenia dynamicznych interfejsów użytkownika.**

XML Web services

- **Programowalne komponenty sieciowe, które mogą być współużywane przez aplikacje w internecie, bądź w intranecie.**

Interfejsy użytkownika

Jest kilka rodzajów interfejsów użytkownika w .NET Framework:

- **ASP.NET: Web Forms, Web Sites**
- **Windows Forms,**
- **Windows WPF (Windows Presentation Foundation)**
- **Windows UWP (Universal Windows Platform)**
- **Aplikacje konsolowe,**

C# w praktyce

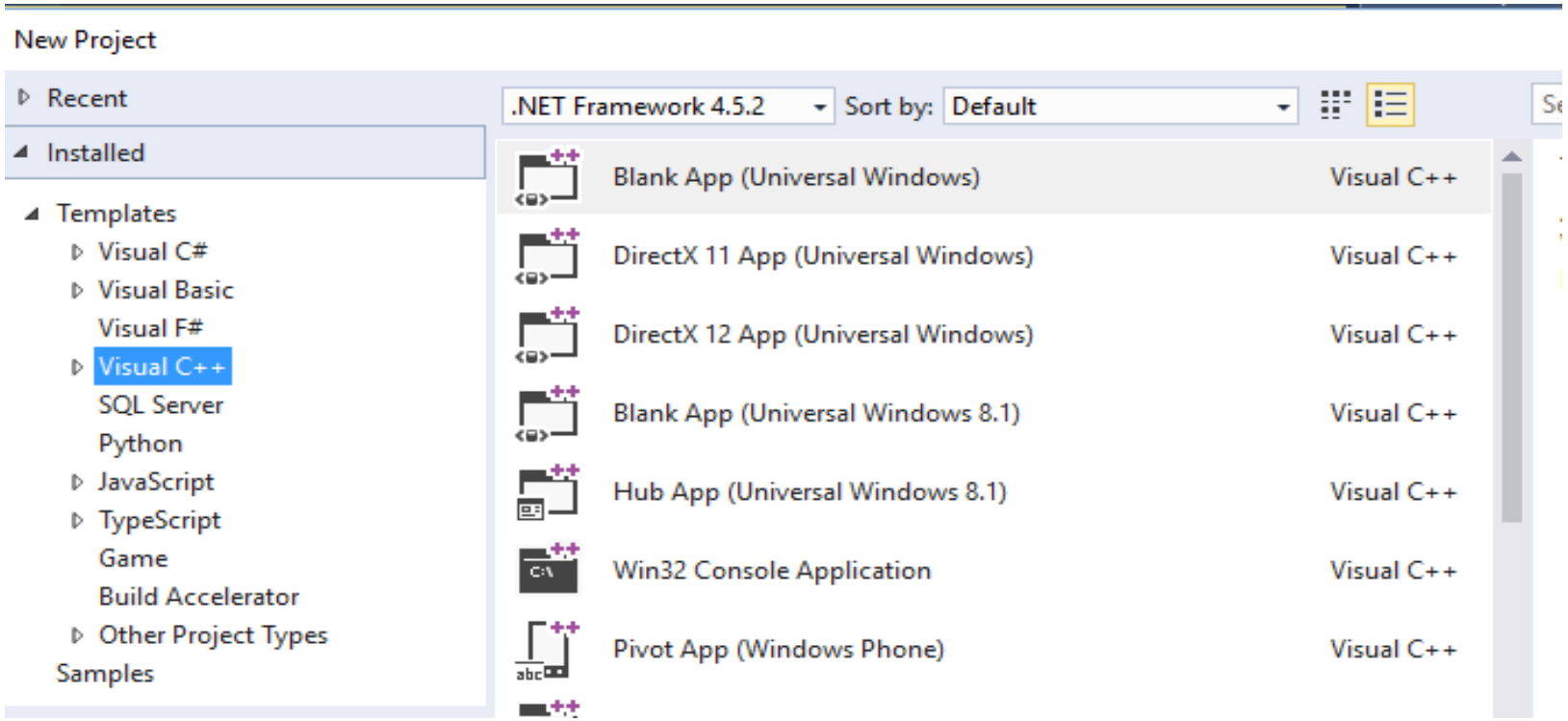
- **Wprowadzono namespaces**
- **Duże podobieństwo do Javy. Uważa się, że C# jest odpowiedzią Microsoft na Javę. (Między innymi Garbage Collector, klasy dziedziczą po klasie Object, wielodziedziczenie interfejsów). Podobno jest kilkukrotnie szybszy od Javy**
- **Uproszczone tworzenie GUI**

C# podstawy

- **Using (jak include lub import)**
- **Typy danych podobne do Java i C++**
- **Operatory podobne do Java i C++**

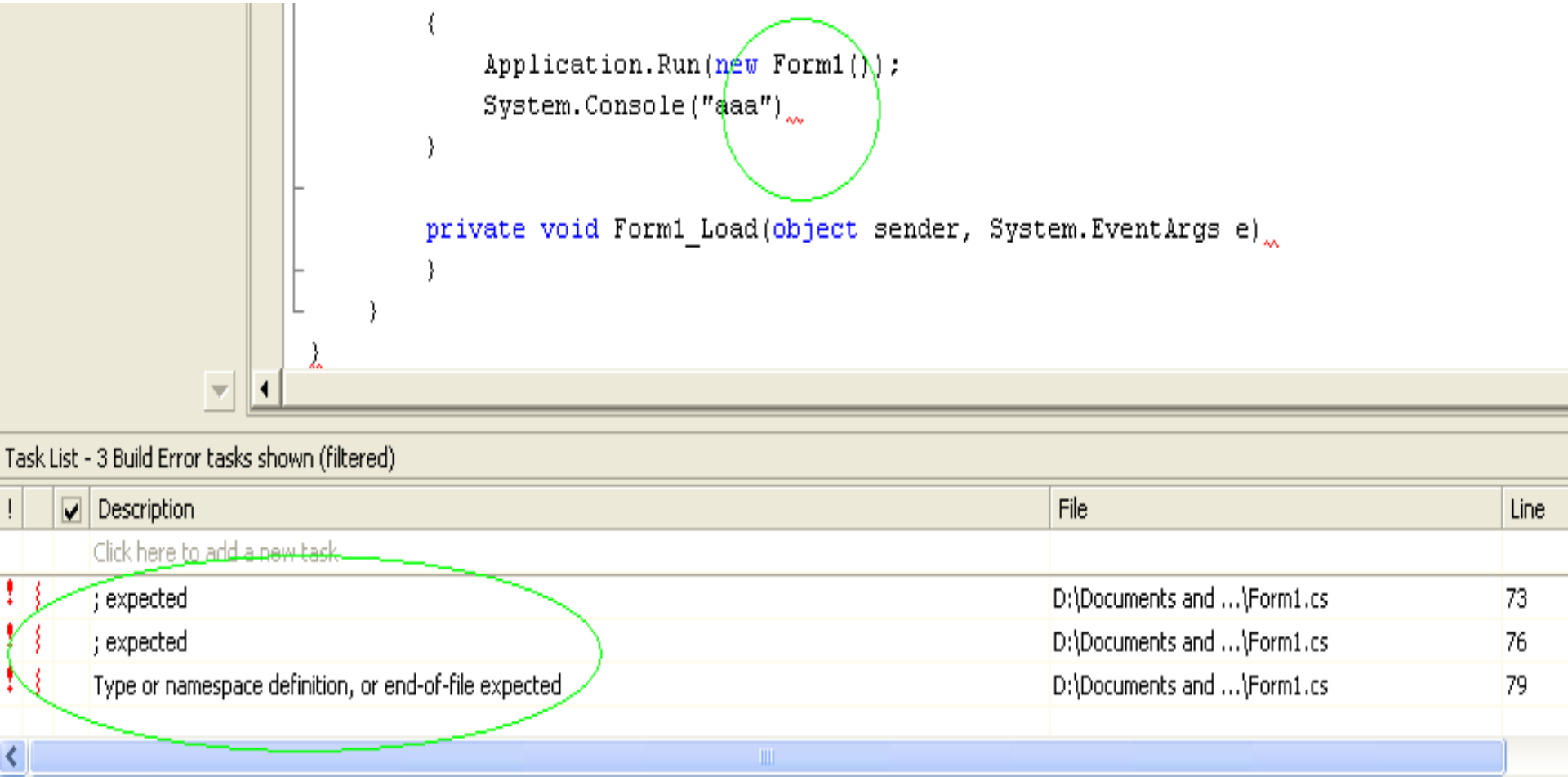
Templates

- **Rozpoczynając nowy projekt wybieramy język w którym będziemy go tworzyć i template**
- **Możemy oczywiście wybrać pusty template**



.NET Framework pomoc 1

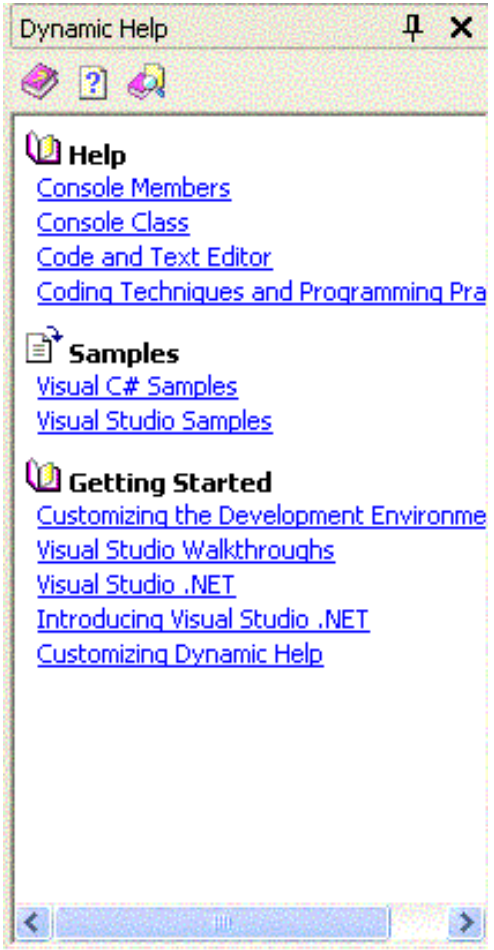
Błędy są wychwytywane podczas programowania



.NET Framework pomoc 2

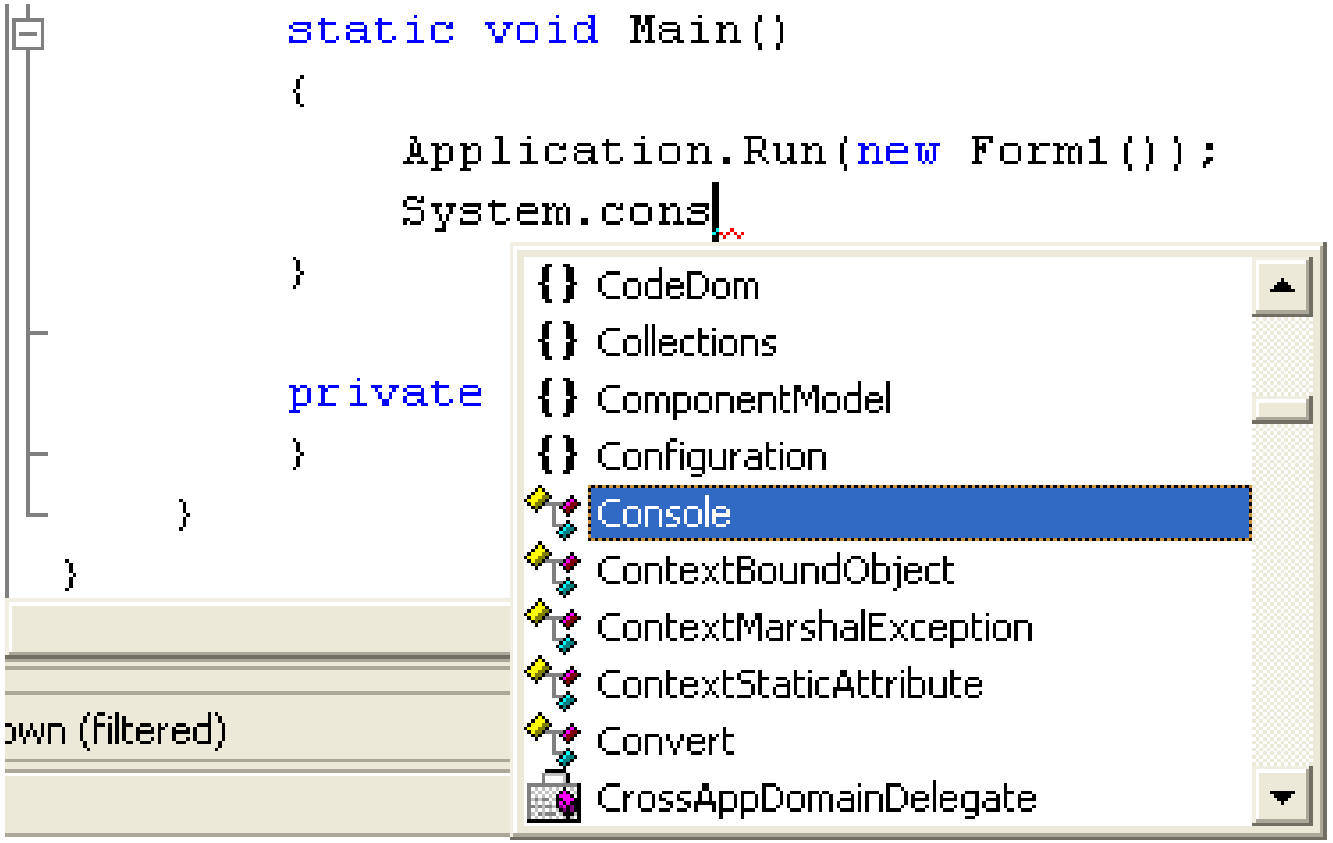
Dynamic help:

- **Podczas programowania okno dynamicznej pomocy pokazuje pomocne informacje związane z wpisywanym kodem.**



.NET Framework pomoc 3

Gdy używamy metod z klas znajdujących się w bibliotece klas, .NET służy „auto-uzupełnianiem”



.NET Framework pomoc 4

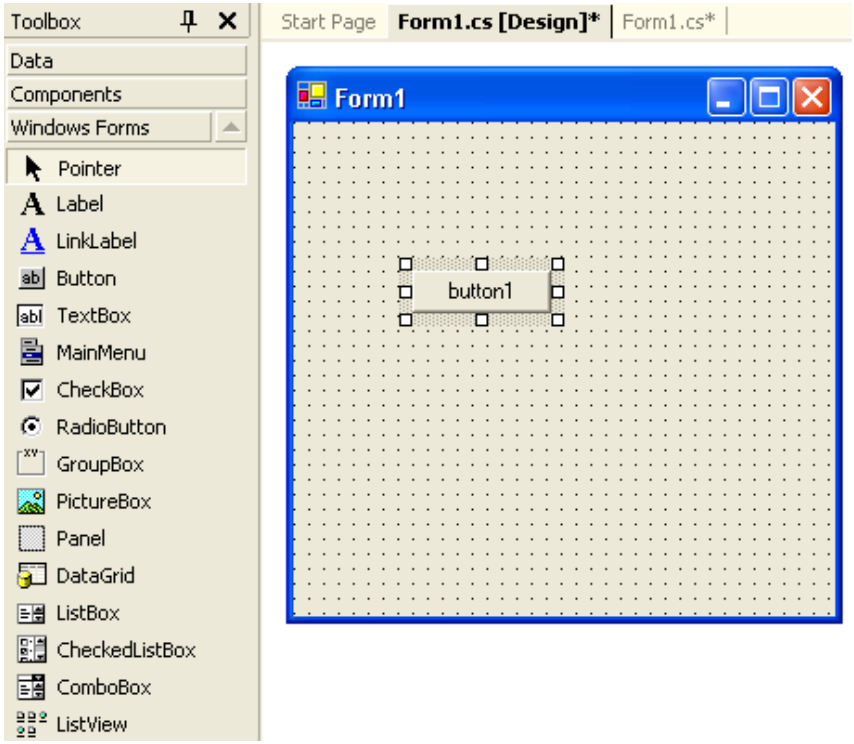
Trzymając kursor myszy nad interesującą nas zmienną, metodą lub nazwą klasy, Framework uruchamia okienko „pop-up” z krótką informacją o interesującym nas obiekcie.

```
static void Main()
{
    Application.Run(new Form1());
    System.Console("aaa");
    String aaa;
}
```

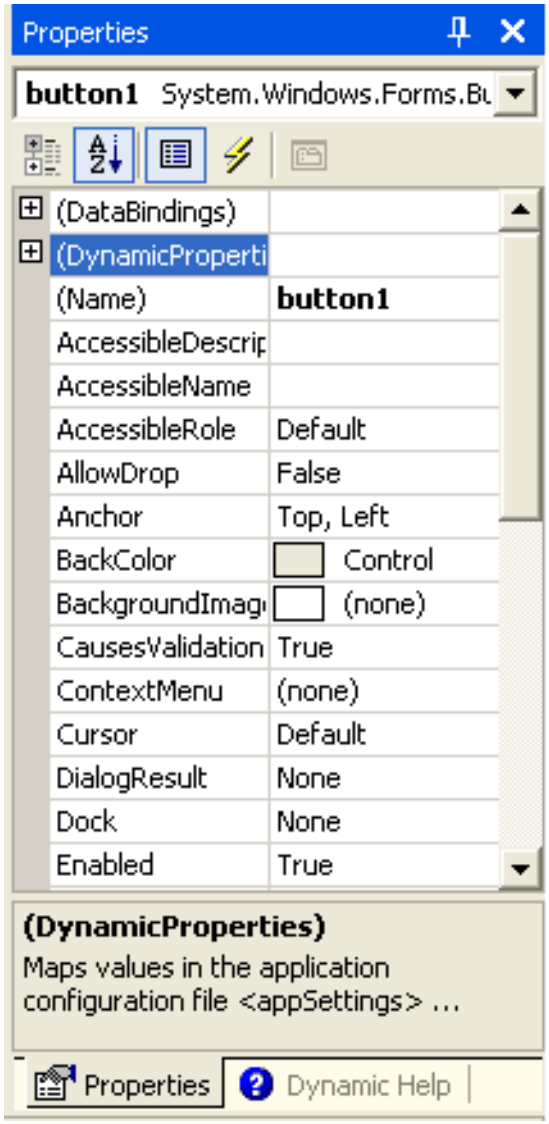
class System.String
Represents an immutable series of characters.

Tworzenie GUI 1

Metodą przeciągania i upuszczania elementów z toolbox'a (narzędnika?)



Tworzenie GUI 2



Parametry poszczególnych elementów możemy zmieniać, w sposób podobny jak w językach 4GL, przy pomocy „Property window” (Okna właściwości)

Tworzenie GUI 3

Klikając dwa razy na stworzony obiekt dostajemy się bezpośrednio do miejsca w kodzie, gdzie jest umieszczona metoda opisująca domyślną akcję skojarzoną z danym obiektem.

```
static void Main()
{
    Application.Run(new Form1());
    System.Console("aaa");
    String aaa;
}

private void Form1_Load(object sender, System.EventArgs e)
}

private void button1_Click() {

}
```

Ukrywanie kodu 1

Używając słów kluczowych #region i #endregion .NET pozwala łatwo ukrywać kod zawarty pomiędzy nimi.

```
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    //
    // Form1
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(292, 266);
    this.Name = "Form1";
    this.Text = "Form1";
    this.Load += new System.EventHandler(this.Form1_Load);

}
#endregion
```

Ukrywanie kodu 2

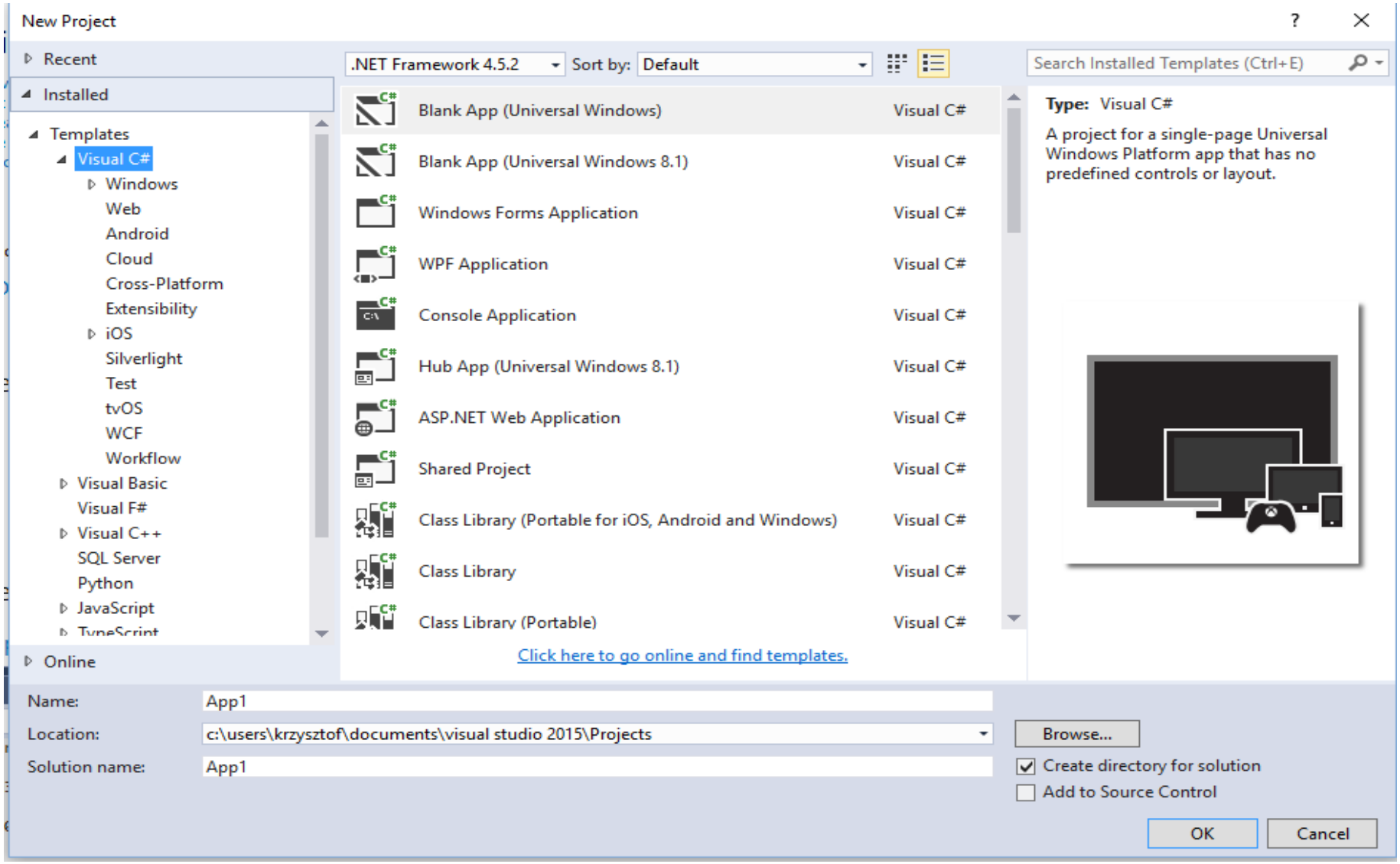
Również można ukrywać metody.

```
Windows Form Designer generated code

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
    System.Console("aaa");
    String aaa;
}

private void Form1_Load(object sender, System.EventArgs e) {
}
```

Pierwszy program 1



Pierwszy program 2

Program w języku C# rozpoczyna się od metody Main, czyli od miejsca w pliku źródłowym:

static void Main(string[] args)

lub

static void Main()

Pierwszy program 3

```
using System;
class Przywitanie
{
    static void Main()
    {
        Console.WriteLine("Witamy w świecie programowania!");
    }
}
```