

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/340133969>

Seri Belajar Windows Forms: Membangun Aplikasi Desktop berbasis .NET Core 3.1 dengan Visual Studio 2019

Book · March 2020

CITATIONS

0

READS

4,430

2 authors:



Mohammad Reza Faisal
Universitas Lambung Mangkurat

94 PUBLICATIONS 307 CITATIONS

[SEE PROFILE](#)



Erick Kurniawan

27 PUBLICATIONS 36 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Publications [View project](#)



DNA Sequence Classification [View project](#)



Seri Belajar
Windows Forms
Membangun Aplikasi Desktop berbasis
.NET Core 3.1 dengan Visual Studio 2019
M Reza Faisal, Erick Kurniawan

Kata Pengantar

Puji dan syukur diucapkan kepada Allah SWT atas selesainya buku yang berjudul Seri Belajar Windows Forms: Membangun Aplikasi Desktop berbasis .NET Core 3.1 dengan Visual Studio 2019.

Buku ini berisi panduan untuk membuat aplikasi desktop pada sistem operasi Windows. Aplikasi ini dibangun dengan menggunakan framework .NET Core 3.1 dan Tool development yang digunakan pada buku ini adalah Visual Studio 2019.

Panduan diberikan dari dasar yaitu dengan memberikan dasar-dasar pemrograman dengan C# yang diimplementasikan pada aplikasi Console. Kemudian dilanjutkan dengan diberikan dasar-dasar membangun aplikasi desktop. Selanjutnya adalah dasar-dasar koneksi dan operasi ke database MS SQL Server yang akan diimplementasikan dalam bentuk aplikasi desktop. Buku ini akan memberikan langkah-langkah setiap kasus dengan detail sehingga pembaca dapat dengan mudah membangun aplikasi yang dijelaskan pada buku ini.

Harapannya buku ini dapat menjadi panduan bagi software developer untuk membangun aplikasi desktop dengan framework .NET Core 3.1.

Akhir kata, selamat membaca dan semoga buku ini bermanfaat bagi para web developer pemula untuk membuat aplikasi web. Kritik dan saran akan sangat berarti dan dapat ditujukan via email.

Banjarmasin, Maret 2020

Erick Kurniawan

(erick.kurniawan@gmail.com)

M Reza Faisal

(reza.faisal@gmail.com)

Daftar Isi

<i>Kata Pengantar</i>	I
<i>Daftar Isi</i>	II
<i>Daftar Gambar</i>	V
1 Pendahuluan	8
.NET Core.....	8
Aplikasi Desktop	9
Windows Forms	10
Windows Presentation Foundation (WPF).....	10
Universal Windows Platform (UWP).....	11
Development Tool	11
Visual Studio 2019.....	12
Visual Studio Code	13
Database.....	13
MS SQL Server.....	13
Bahan Pendukung.....	14
Buku	14
Source Code	15
2 .NET Core 3.1 SDK & Runtime	16
Installasi.....	16
Uji Coba	16
.NET Core Command Line Tool.....	17
Info & Bantuan	17
Membuat Project	18
Build.....	22
Run	22
3 Visual Studio 2019	25
Installasi.....	25
Antarmuka.....	26
Solution Explorer	28
Editor	28
Toolbox	30

Properties	30
Output	31
Error List	31
Solution & Project.....	31
Solution.....	31
Project	33
Item	37
Build & Debug.....	38
Depedencies	40
NuGet	42
4 Pengantar Pemrograman C#	44
Pendahuluan	44
Persiapan.....	44
Struktur Program Aplikasi Console/Desktop	44
Aturan Penulisan Kode Program.....	47
Tipe & Variable	47
Expression.....	49
Operator Aritmatika	49
Operator Logika	50
Operator Equality	52
Operator Order.....	53
Statement	54
Array	55
Percabangan	57
Statement if	57
Statement switch	58
Pengulangan.....	58
Statement for.....	59
Statement while	60
Method	61
Pemrograman Berbasis Obyek	62
Class & Object.....	62
Property.....	65
Access Modifier	66

5 Pemrograman Visual.....	68
Pengenalan Lingkungan & Control.....	68
Control & Properties.....	69
Struktur Form	70
Program.cs	71
Toolbox.....	74
Kalkulator Sederhana	74
Quisioner.....	79
Multiple Document Interfaces (MDI).....	84
6 Akses Database	101
Pendahuluan	101
Koneksi & Insert Data.....	102
Project	102
Library	102
Database	103
Model.....	105
Antarmuka.....	107
7 Penutup.....	109

Daftar Gambar

Gambar 1. Arsitektur .NET 5	8
Gambar 2. Jadwal pengembangan .NET 5	9
Gambar 3. Daftar file installer .NET Core	9
Gambar 4. Contoh antarmuka aplikasi yang dibangun menggunakan WinForms.	10
Gambar 5. Contoh antarmuka aplikasi yang dibangun menggunakan WPF.	11
Gambar 6. Visual Studio Website.....	12
Gambar 7. Perbandingan Fitur Visual Studio 2019.....	12
Gambar 8. Visual Studio Code.....	13
Gambar 9. SQL Server 2019.....	14
Gambar 10. Buku-buku pendukung.	14
Gambar 11. .NET Core 3.1 – Versi.....	16
Gambar 12. Daftar file dan folder project Console Application.....	21
Gambar 13. Daftar file dan folder project Windows Form (WinForms) Application.	21
Gambar 14. Daftar file dan folder project WPF Application.	22
Gambar 15. dotnet build.....	22
Gambar 16. Menjalankan project Console Application - ConsoleCS.	23
Gambar 17. Menjalankan project Windows Forms (WinForms) Application.....	23
Gambar 18. Menjalankan WPF Application.	24
Gambar 19. Halaman untuk mengunduh Visual Studio 2019.	25
Gambar 20. Visual Studio 2019 - antarmuka installer.	26
Gambar 21. Visual Studio 2019 - Start Page.....	26
Gambar 22. Visual Studio 2019 - Antarmuka tanpa kode yang dipilih.	27
Gambar 23. Visual Studio 2019 - antarmuka dengan kode.....	27
Gambar 24. Visual Studio 2019 – Solution Explorer.	28
Gambar 25. Visual Studio 2019 - Code Editor.	29
Gambar 26. Visual Studio 2019 – Visual Editor.....	29
Gambar 27. Visual Studio - Toolbox.	30
Gambar 28. Visual Studio 2019 – Properties.	30
Gambar 29. Visual Studio 2019 – Output.	31
Gambar 30. Visual Studio 2019 – Error List.	31
Gambar 31. Visual Studio 2019 - Create a new project - Blank Solution.....	32
Gambar 32. Visual Studio 2019 - Configure your new project.	32

Gambar 33. WinFormSolution	33
Gambar 34. Pilihan project berdasarkan language, platform dan project types.....	33
Gambar 35. Membuat project Console App (.NET Core)	34
Gambar 36. Visual Studio 2019 - Membuat project ConsoleCS.	35
Gambar 37. Visual Studio 2019 - Project ConsoleCS pada Solution Explorer.....	35
Gambar 38. Visual Studio 2019 - Membuat project Windows Forms App (.NET Core).	36
Gambar 39. Visual Studio 2019 - memberi nama project WinFormCS.	36
Gambar 40. Visual Studio 2019 - Project WinFormCS pada Solution Explorer.	37
Gambar 41. Visual Studio 2019 - Add New Item.	38
Gambar 42. Visual Studio 2019 – Output proses build.	39
Gambar 43. Visual Studio 2019 – Output proses debug.....	39
Gambar 44. Visual Studio 2019 - Menjalankan project WinFormCS.	40
Gambar 45. Visual Studio 2019 - Depedencies.	40
Gambar 46. Visual Studio 2019 - Reference Manager - COM.....	41
Gambar 47. Visual Studio 2019 - Reference Manager - Projects.....	42
Gambar 48. Visual Studio 2019 - NuGet Package Manager - Installed.	42
Gambar 49. Visual Studio 2019 - NuGet Package Manager - Browse.	43
Gambar 50. Visual Studio 2019 - NuGet pada Solution Explorer.	43
Gambar 51. Folder CSharpBasic pada Solution Explorer.	44
Gambar 52. Project LatihanVariable.	47
Gambar 53. Membuat class MathOperation.cs.....	63
Gambar 54. Project IntroVisual.....	68
Gambar 55. Visual Designer.....	69
Gambar 56. Control & property.....	69
Gambar 57. Membuat MainForm.	72
Gambar 58. MainForm pada Solution Explorer.	73
Gambar 59. Text pada MainForm.....	73
Gambar 60. Property Form.....	74
Gambar 61. Modifikasi property form Simple Calculator.	75
Gambar 62. Property Name pada control Label.....	75
Gambar 63. Property Name pada control TextBox.	76
Gambar 64. Property Name pada control Button.	76
Gambar 65. Antarmuka SimpleQuiz.	79
Gambar 66. RadioBox di dalam GroupBox.	80
Gambar 67. ComboBox untuk Occupation.	80

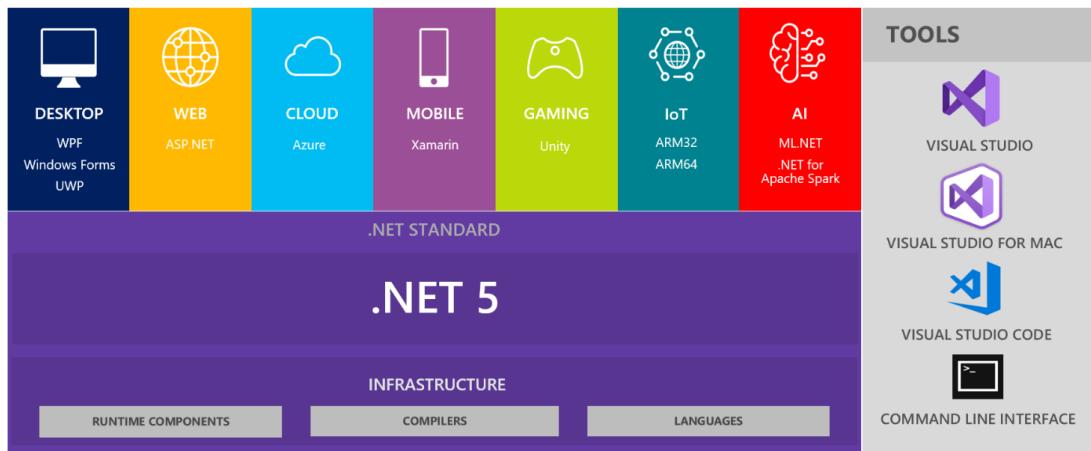
Gambar 68. Edit item occupation.....	80
Gambar 69. CheckBox Programming Skill.....	81
Gambar 70. Label hasil.....	81
Gambar 71. Hasil akhir Simple Quiz.....	83
Gambar 72. Contoh Multiple Document Interfaces (MDI)	84
Gambar 73. Project ControlSamples.....	84
Gambar 74. Membuat form induk MainForm.cs.....	85
Gambar 75. Tampilan MainForm default.....	85
Gambar 76. MainForm menjadi form induk.....	86
Gambar 77. Control MenuStrip.....	86
Gambar 78. Items Collection Editor.....	87
Gambar 79. Menambahkan item anak pada item ToolStripMenuItem pertama.....	87
Gambar 80. MainForm dengan menu.....	88
Gambar 81. Control Toolbar.....	88
Gambar 82. Edit item pada ToolStrip.....	89
Gambar 83. Windows Items Collection Editor untuk toolbar.....	89
Gambar 84. Window Select Resource untuk memilih gambar.....	89
Gambar 85. Combo box untuk memilih control pada area Properties.....	90
Gambar 86. Event pada toolStripMenuItem2.....	91
Gambar 87. Window konfirmasi untuk keluar.....	93
Gambar 88. Membuat AboutForm.cs.....	94
Gambar 89. Control Image pada form About.....	94
Gambar 90. Antarmuka form About yang lengkap.....	95
Gambar 91. Control Example 1.....	98
Gambar 92. Nuget pacage Manager: DatabaeInsert.....	102
Gambar 93. Packages.....	103
Gambar 94. Membuat table GuestBook.....	103
Gambar 95. Simpan table.....	104
Gambar 96. Application Configuration File.....	104
Gambar 97. Antarmuka Database Insert.....	107

Pendahuluan

.NET Core

.NET Framework adalah software framework yang dikembangkan oleh Microsoft pada tahun 2002. .NET Framework terdiri atas aturan kerja, aturan pemrograman dan banyak class library (Framework Class Library) untuk membangun bermacam aplikasi, seperti aplikasi desktop, aplikasi mobile, aplikasi web dan cloud. Sampai saat ini .NET Framework telah mencapai versi 4.8. .NET Framework ini hanya dapat digunakan pada platform atau sistem operasi MS Windows. Aplikasi-aplikasi yang dibangun di atas .NET Framework hanya dapat dijalankan jika pada komputer telah terinstall .NET Framework. Artinya aplikasi-aplikasi itu hanya akan jalannya pada platform MS Windows.

Sejak tahun 2016 Microsoft mengembangkan .NET Core, yaitu “.NET Framework” yang bersifat open-source dan multiplatform. Artinya .NET Core dapat dijalankan pada platform Windows, Linux dan Mac OS. Sehingga aplikasi-aplikasi yang dibangun di atas framework ini juga dapat dijalankan di atas banyak platform. Saat ini .NET Core hanya mendukung bahasa pemrograman C# dan F#. Saat buku ini ditulis .NET Core telah mencapai versi 3.1 yang merupakan versi long-term supported (LTS). Versi ini akan disupport selama tiga tahun. Dan selanjutnya versi ini akan menjadi .NET 5 atau .NET Core vNext pada akhir tahun 2020.



Gambar 1. Arsitektur .NET 5.

Gambar di atas ini adalah arsitektur .NET 5. Pada ilustrasi di atas ini .NET 5 menjadi fondasi untuk membangun aplikasi desktop, web, cloud, mobile, gaming, IoT dan AI. Tipe-tipe aplikasi ini sudah dapat dibangun dengan menggunakan .NET Framework. Dari ilustrasi di atas sudah jelas bahwa .NET 5 akan menggantikan peran .NET Framework.

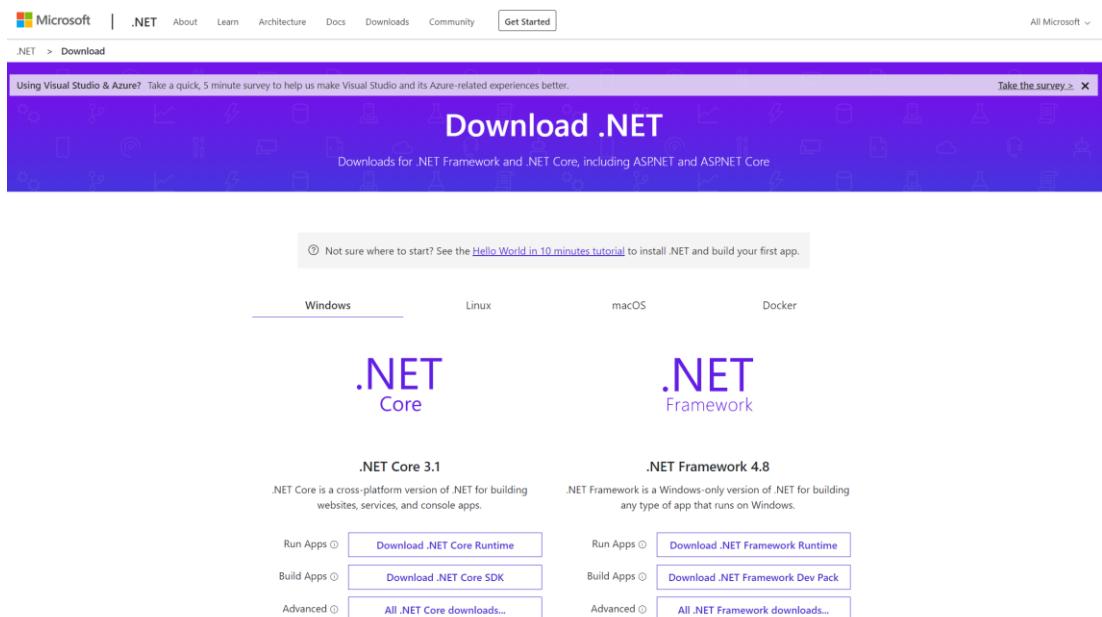
Pada ilustrasi di atas dapat dilihat tool development yang dapat digunakan seperti Visual Studio yang berjalan pada sistem operasi Windows. Visual Studio for Mac yang dapat digunakan pada sistem operasi Mac OS. Visual Studio Code yang merupakan tool code editor ringan yang dapat dijalankan pada multiplatform. Selain itu .NET 5 juga menyediakan perintah command line untuk melakukan debug, kompilasi dan menjalankan program.

Jadwal pengembangan .NET 5 dapat dilihat pada gambar di bawah ini.



Gambar 2. Jadwal pengembangan .NET 5.

Untuk mendapatkan .NET Core dapat diunduh pada link berikut <https://dotnet.microsoft.com/download>. Pada halaman ini tersedia dua tipe framework yang telah disebutkan di atas yaitu .NET Core dan .NET Framework. Kita dapat memilih platform yang diinginkan yaitu Windows, Linux, macOS dan Docker.



Gambar 3. Daftar file installer .NET Core.

Pada halaman tersebut tersedia .NET Core Runtime yang berfungsi untuk menjalankan aplikasi yang dibangun dengan framework ini. sedangkan .NET Core SDK selain berisi runtime juga berisi software development kit (SDK) yang dapat digunakan untuk membangun aplikasi dengan framework ini. Kami sarankan untuk mengunduh .NET Core SDK versi 3.1.

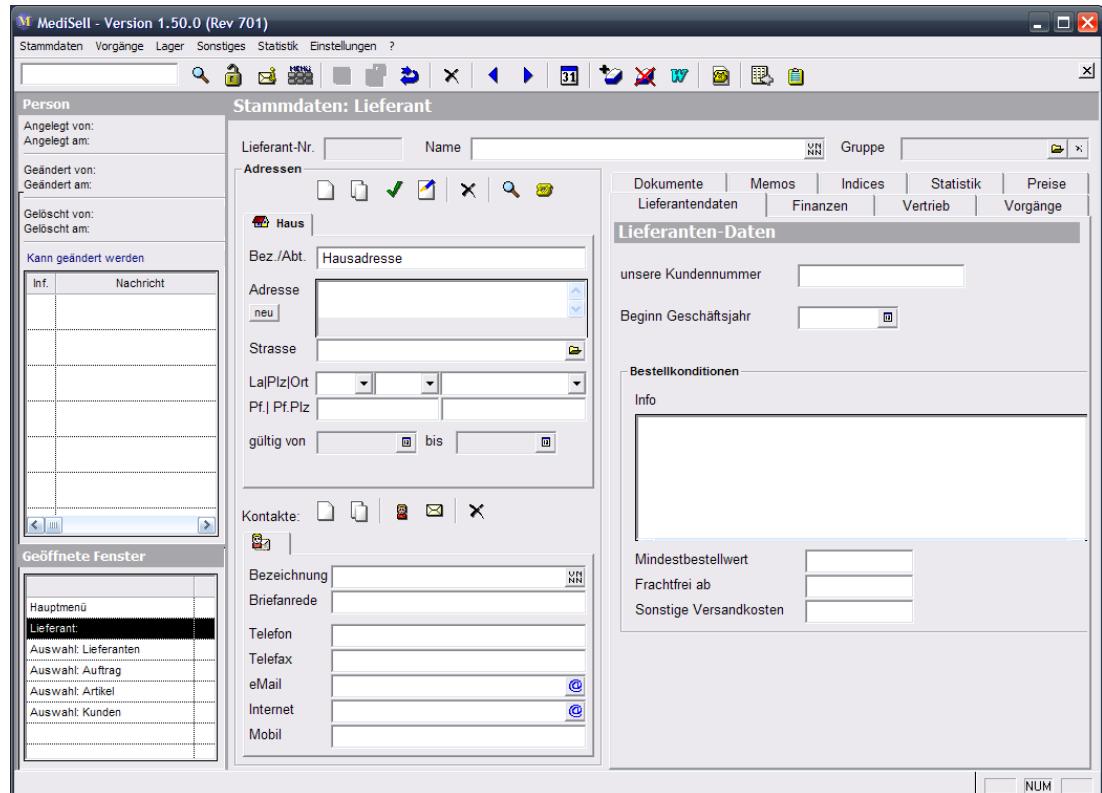
Aplikasi Desktop

Pada Gambar 1, dapat dilihat bahwa .NET Core 3.1 atau nanti dikenal sebagai .NET 5 dapat digunakan untuk membangun banyak tipe aplikasi, salah satunya adalah aplikasi dekstop. Pada framework ini, aplikasi desktop dibedakan menjadi:

- Windows Forms.
- Window Presentation Foundation (WPF).
- Universal Windows Platform (UWP).

Windows Forms

Windows Forms atau dikenal juga sebagai WinForms adalah User Interface (UI) framework untuk membangun aplikasi desktop. WinForms umum digunakan untuk membuat aplikasi desktop karena tampilannya formal dan umum ditemui pada aplikasi desktop. Dan class library ini telah ada sejak .NET Framework dirilis tahun 2002. Dan pada akhir tahun 2018, Microsoft mengumumkan bahwa Windows Forms sebagai proyek open source pada GitHub (<https://github.com/dotnet/winforms>).



Gambar 4. Contoh antarmuka aplikasi yang dibangun menggunakan WinForms.

WinForms adalah .NET wrapper yang berisi Windows beberapa library user interface seperti User32 dan GDI+. Selain itu juga memiliki control dan fungsionalitas yang hanya dimiliki oleh WinForms. WinForms juga memberikan cara yang produktif untuk membangun aplikasi desktop karena memiliki visual designer pada Visual Studio. Hal ini membuat antarmuka aplikasi dapat dibuat dengan cara melakukan drag and drop control visual, ditambah lagi fungsi-fungsi lain sehingga aplikasi desktop dapat dibuat dengan sangat mudah.

Contoh antarmuka aplikasi WinForms dapat dilihat pada gambar di atas. Antarmuka aplikasi di atas terlihat formal dan kaku.

Windows Presentation Foundation (WPF)

Jika Windows Forms memberikan antarmuka yang terlihat formal atau kaku. Maka Windows Presentation Foundation (WPF) dapat memberikan antarmuka yang lebih bebas. Hal ini dikarenakan WPF menggunakan Extensible Application Markup Language (XAML) untuk membuat antarmuka. Secara singkat, XAML mirip dengan HTML. Sehingga antarmuka aplikasi WPF juga dapat didesain seperti halnya halaman web.

Kelebihan WPF lainnya adalah antarmuka yang independek dengan resolusi layar komputer dan dirender dalam format vector sehingga seberapapun besarnya resolusi yang digunakan, antarmuka mengalami penurunan kualitas.

Pada gambar berikut ini adalah contoh antarmuka aplikasi yang dibangun dengan WPF.



Gambar 5. Contoh antarmuka aplikasi yang dibangun menggunakan WPF.

Universal Windows Platform (UWP)

Universal Windows Platform adalah platform yang memungkinkan developer membangun aplikasi lintas perangkat yang menggunakan sistem operasi Windows yaitu perangkat mobile seperti smartphone, PC, Xbox, Surface, HoloLens, IoT dan lain-lain.

Pada buku ini hanya akan dibahas tentang pembangunan aplikasi desktop dengan Windows Forms.

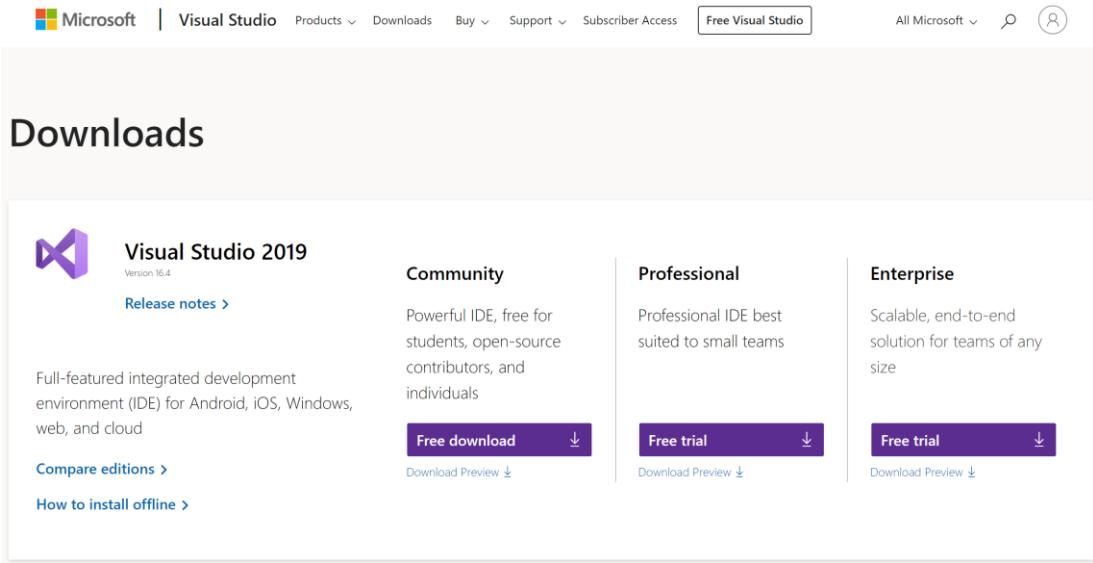
INFORMASI

Walaupun .NET Core disebut sebagai framework yang dapat digunakan pada multiplatform, namun tidak semua aplikasi yang dibangun dapat dijalankan pada multiplatform juga. Sebagai contoh framework untuk membangun aplikasi desktop yang telah dijelaskan di atas (untuk saat ini) hanya dapat dijalankan pada platform Windows saja. Harapannya nanti .NET Core juga mendukung pembangunan aplikasi desktop yang dapat dijalankan pada multiplatform.

Development Tool

Visual Studio adalah sebuah integrated development environment (IDE) yang dikembangkan oleh Microsoft, yang dapat digunakan untuk mengembangkan aplikasi Android, iOS, Windows, web dan cloud. Tool ini tersedia untuk berbagai platform yaitu Windows, Linux dan MacOS.

Visual Studio dapat diunduh di link berikut ini <https://www.visualstudio.com/downloads/>.



Gambar 6. Visual Studio Website.

Visual Studio 2019

Visual Studio 2019 tersedia dalam tiga pilihan lisensi yaitu Enterprise dan Professional yang berbayar. Sedangkan Community bersifat gratis. Berikut adalah perbandingan fitur-fitur yang dimiliki oleh masing-masing versi.

Supported Features	Visual Studio Community	Visual Studio Professional	Visual Studio Enterprise
⊕ Supported Usage Scenarios	● ● ○	● ● ●	● ● ●
Development Platform Support ²	● ● ●	● ● ●	● ● ●
⊕ Integrated Development Environment	● ● ○	● ● ○	● ● ●
⊕ Advanced Debugging and Diagnostics	● ● ○○	● ● ○○	● ● ●
⊕ Testing Tools	● ○○○	● ○○○	● ● ●
⊕ Cross-platform Development	● ● ○○	● ● ○○	● ● ●
⊕ Collaboration Tools and Features	● ● ●	● ● ●	● ● ●

Gambar 7. Perbandingan Fitur Visual Studio 2019.

Saat buku ini ditulis, kami menggunakan Visual Studio 2019 versi 16.6 Preview. Artinya tool ini masih dalam tahap pengembangan. <https://visualstudio.microsoft.com/vs/preview/>. Artinya ada fitur-fitur yang belum dapat digunakan pada tahap produksi, karena fitur-fitur tersebut kemungkinan akan berubah atau hilang. Atau masih terdapat beberapa error yang terjadi jika digunakan pada tahap produksi. Sedangkan Visual Studio 2019 versi 16.4.2 bukan versi Preview lagi dan sudah dapat digunakan pada tahap produksi.

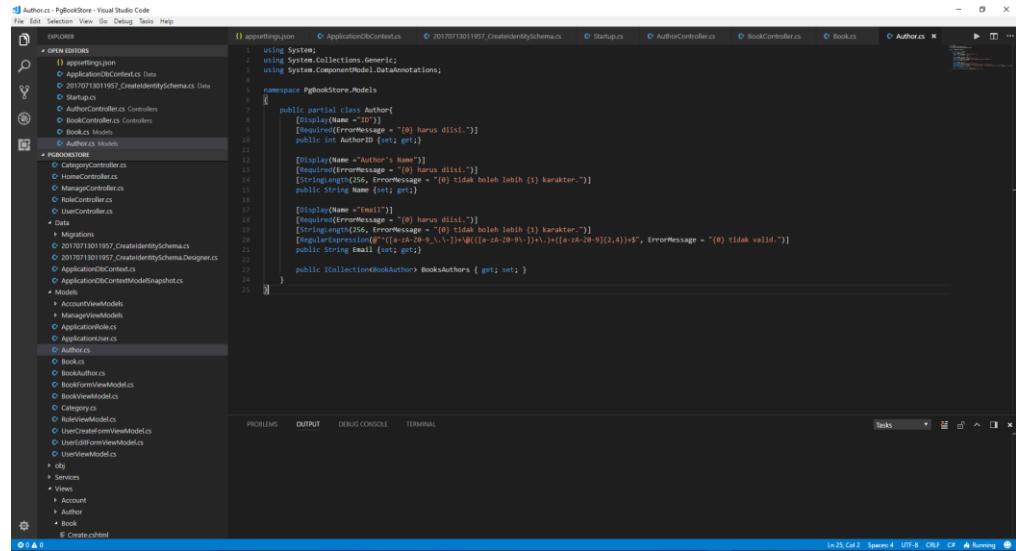
Kami menggunakan versi Visual Studio 2019 versi 16.6 Preview ini agar dapat menunjukkan bagaimana membangun aplikasi desktop dengan WinForms. Karena hanya versi ini yang memiliki fitur-fitur untuk menggunakan WinForms.

Visual Studio Code

Visual Studio Code adalah versi Visual Studio yang ringan tetapi tetap powerful. Versi ini seperti code editor dengan fitur-fitur tambahan untuk mempermudah penulisan kode program.

Visual Studio Code tersedia pada platform Windows, Linux dan MacOS. Visual Studio Code juga mendukung banyak bahasa pemrograman seperti halnya Visual Studio 2015 ditambah bahasa pemrograman PHP, Node.js dan lain-lain.

Berikut adalah tampilan Visual Studio Code.



Gambar 8. Visual Studio Code.

Database

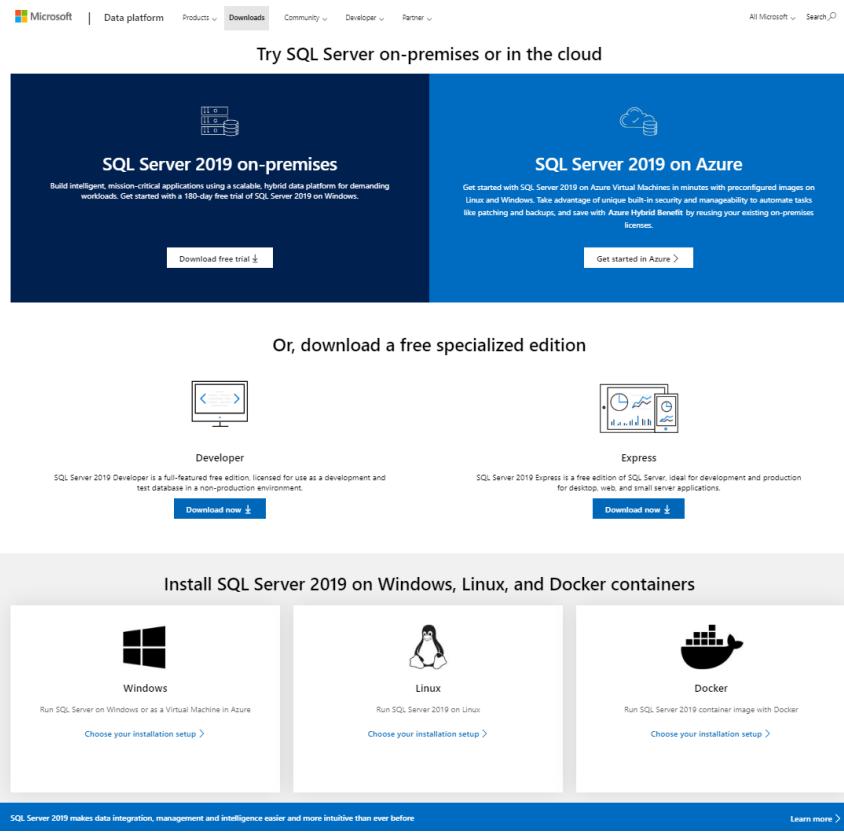
Database adalah koleksi data yang terorganisir. Database yang sering digunakan adalah relational database yang merupakan koleksi dari skema, tabel, query dan view. Software yang memberikan layanan untuk mengelola database dikenal dengan nama Relational Database Management System (RDBMS) atau Database Server. Saat ini telah banyak tersedia database server, beberapa diantaranya adalah MS SQL Server, MySQL, dan PostgreSQL.

MS SQL Server

MS SQL Server adalah RDBMS yang dikembangkan oleh Microsoft. Versi terbaru dari RDBMS yaitu MS SQL Server 2019 telah tersedia tidak hanya pada platform Windows tetapi juga tersedia platform Linux. Jika ingin memanfaatkan RDBMS ini pada platform MacOS maka dapat memanfaatkan Docker.

MS SQL Server tersedia dalam beberapa edisi yaitu Developer and Express Edition. Express Edition dapat digunakan secara gratis dengan beberapa keterbatasan seperti tidak dapat menggunakan kemampuan multi-core dan maksimal ukuran file database yaitu 10GB.

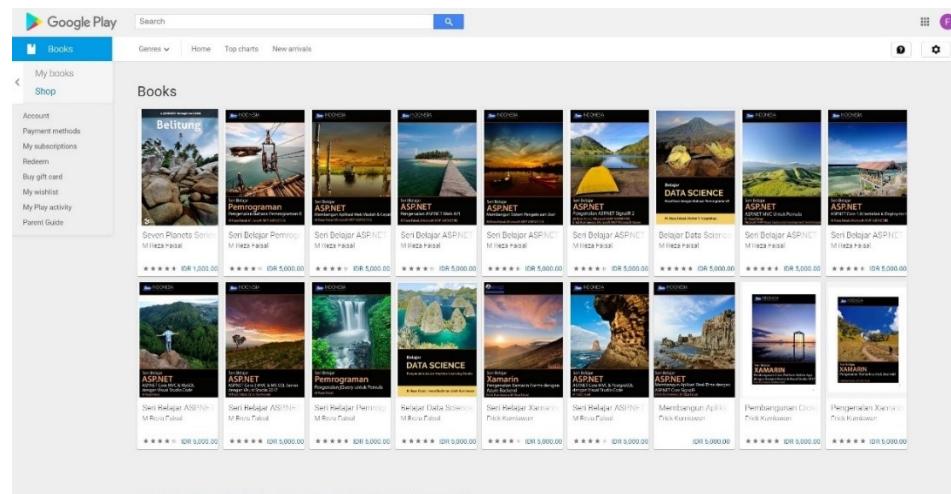
Untuk mencoba MS SQL Server 2019 maka file installer dapat diunduh di sini <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>. Pada buku ini, contoh-contoh akses dan operasi database menggunakan MS SQL Server edisi Express.



Gambar 9. SQL Server 2019.

Bahan Pendukung

Buku



Gambar 10. Buku-buku pendukung.

Buku-buku ini dapat diunduh dengan menggunakan link berikut ini:
<https://play.google.com/store/books/author?id=M%20Reza%20Faisal>

Source Code

Source code contoh-contoh yang dibuat pada buku ini dapat diunduh pada repository pada link berikut ini:

1. <https://github.com/rezafaisal/WinFormNETCoreCodeSamples>.

2

.NET Core 3.1 SDK & Runtime

Pada bab ini akan dijelaskan langkah-langkah untuk melakukan installasi .NET Core 3.1 SDK pada sistem operasi Windows.

Installasi

Saat buku ini ditulis telah tersedia:

- .NET Core SDK v3.1.101.
- .NET Core Runtime v3.1.1

Installer tersebut dapat diunduh pada link berikut <https://dotnet.microsoft.com/download>.

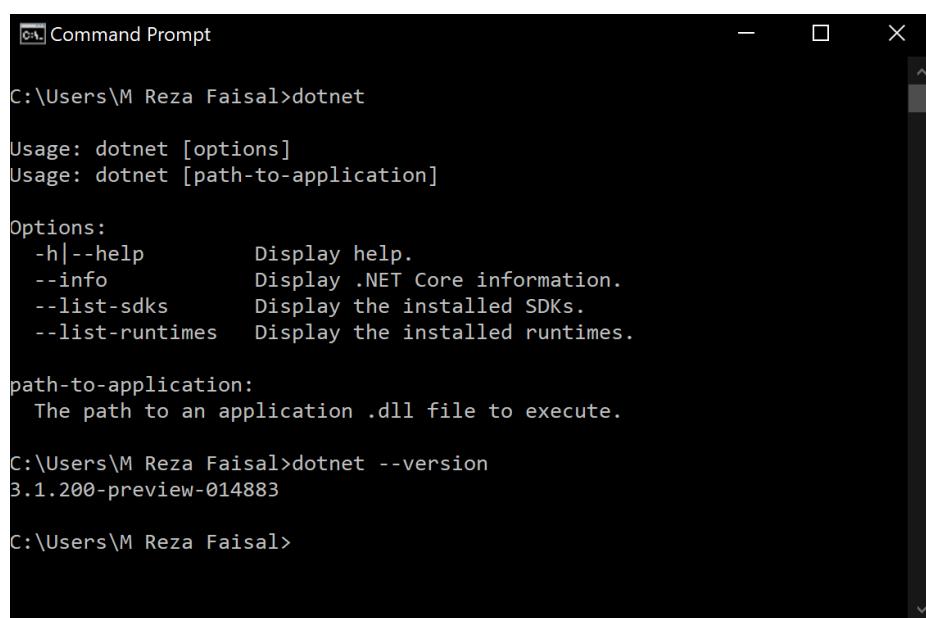
Proses installasi pada platform Windows dan MacOS sangat mudah, hanya dengan cara mengeksekusi file installer kemudian ikuti petunjuk yang diberikan pada window installer.

Uji Coba

Untuk uji coba dapat dilakukan dengan mengetikan perintah berikut ini.

```
dotnet --version
```

Output dari perintah ini adalah sebagai berikut.



```
C:\Users\M Reza Faisal>dotnet
Usage: dotnet [options]
Usage: dotnet [path-to-application]

Options:
  -h|--help      Display help.
  --info         Display .NET Core information.
  --list-sdks    Display the installed SDKs.
  --list-runtimes Display the installed runtimes.

path-to-application:
  The path to an application .dll file to execute.

C:\Users\M Reza Faisal>dotnet --version
3.1.200-preview-014883

C:\Users\M Reza Faisal>
```

Gambar 11. .NET Core 3.1 – Versi.

Dari output dapat dilihat versi 3.1.200-preview-014883, versi ini berasal dari Visual Studio 2019 versi 16.6 Preview. Versi mungkin akan berbeda dengan milik pembaca tergantung versi Visual Studio atau .NET Core SDK yang diinstall.

.NET Core Command Line Tool

Pada sub bab sebelumnya telah dapat dilihat penggunaan perintah “dotnet”. Perintah ini adalah perintah utama yang digunakan untuk melakukan hal-hal penting seperti:

1. Membuat project baru.
2. Melakukan migrasi atau upgrade project ke versi yang lebih baru.
3. Melakukan restore library atau package dan tool yang digunakan project.
4. Melakukan build.
5. Melakukan testing.
6. Menjalankan dan mempublish aplikasi.
7. Dan lain-lain.

Pada sub bab ini akan dijelaskan beberapa fungsi penting perintah “dotnet”.

Info & Bantuan

Untuk mengetahui informasi versi .NET Core Command Line Tool yang digunakan dan informasi lainnya dapat digunakan perintah berikut.

```
dotnet --info
```

Hasilnya adalah sebagai berikut.

```
.NET Core SDK (reflecting any global.json):  
  Version: 3.1.200-preview-014883  
  Commit: 4e2a0ee959  
  
Runtime Environment:  
  OS Name: Windows  
  OS Version: 10.0.18363  
  OS Platform: Windows  
  RID: win10-x64  
  Base Path: C:\Program Files\dotnet\sdk\3.1.200-preview-014883\  
  
Host (useful for support):  
  Version: 3.1.1  
  Commit: a1388f194c
```

Sedangkan untuk mengetahui secara lengkap opsi lengkap yang dapat digunakan pada perintah “dotnet” dapat digunakan perintah berikut.

```
dotnet --help
```

Maka akan dapat dilihat informasi sebagai berikut.

```
.NET Core SDK (3.1.200-preview-014883)  
Usage: dotnet [runtime-options] [path-to-application] [arguments]  
  
Execute a .NET Core application.  
  
runtime-options:
```

```

--additionalprobingpath <path>    Path containing probing policy and
assemblies to probe for.
--additional-deps <path>          Path to additional deps.json file.
--fx-version <version>           Version of the installed Shared Framework
to use to run the application.
--roll-forward <setting>         Roll forward to framework version
(LatestPatch, Minor, LatestMinor, Major, LatestMajor, Disable).

path-to-application:
The path to an application .dll file to execute.

Usage: dotnet [sdk-options] [command] [command-options] [arguments]

Execute a .NET Core SDK command.

sdk-options:
-d|--diagnostics  Enable diagnostic output.
-h|--help          Show command line help.
--info             Display .NET Core information.
--list-runtimes   Display the installed runtimes.
--list-sdks        Display the installed SDKs.
--version          Display .NET Core SDK version in use.

SDK commands:
add                Add a package or reference to a .NET project.
build              Build a .NET project.
build-server       Interact with servers started by a build.
clean              Clean build outputs of a .NET project.
help               Show command line help.
list               List project references of a .NET project.
msbuild            Run Microsoft Build Engine (MSBuild) commands.
new                Create a new .NET project or file.
nuget              Provides additional NuGet commands.
pack               Create a NuGet package.
publish            Publish a .NET project for deployment.
remove              Remove a package or reference from a .NET project.
restore             Restore dependencies specified in a .NET project.
run                Build and run a .NET project output.
sln                Modify Visual Studio solution files.
store              Store the specified assemblies in the runtime package
store.
test               Run unit tests using the test runner specified in a .NET
project.
tool               Install or manage tools that extend the .NET experience.
vstest              Run Microsoft Test Engine (VSTest) commands.

Additional commands from bundled tools:
dev-certs          Create and manage development certificates.
fsi                Start F# Interactive / execute F# scripts.
sql-cache          SQL Server cache command-line tools.
user-secrets       Manage development user secrets.
watch              Start a file watcher that runs a command when files
change.

```

Run 'dotnet [command] --help' for more information on a command.

Membuat Project

Untuk membuat project dengan perintah “dotnet” digunakan opsi “new”. Untuk mengetahui informasi bantuan cara pembuatan project dapat digunakan perintah berikut.

```
dotnet new --help
```

Dan berikut adalah informasi dari perintah di atas.

```
Usage: new [options]

Options:
  -h, --help           Displays help for this command.
  -l, --list            Lists templates containing the specified name. If no name is
specified, lists all templates.
  -n, --name            The name for the output being created. If no name is specified, the
name of the current directory is used.
  -o, --output           Location to place the generated output.
  -i, --install          Installs a source or a template pack.
  -u, --uninstall        Uninstalls a source or a template pack.
  --nuget-source        Specifies a NuGet source to use during install.
  --type                Filters templates based on available types. Predefined values are
"project", "item" or "other".
  --dry-run             Displays a summary of what would happen if the given command line
were run if it would result in a template creation.
  --force               Forces content to be generated even if it would change existing
files.
  -lang, --language      Filters templates based on language and specifies the language of
the template to create.
  --update-check         Check the currently installed template packs for updates.
  --update-apply         Check the currently installed template packs for update, and install
the updates.
```

Templates Tags	Short Name	Language
Console Application	console	[C#], F#, VB
Common/Console		
Class library	classlib	[C#], F#, VB
Common/Library		
WPF Application	wpf	[C#]
Common/WPF		
WPF Class library	wpflib	[C#]
Common/WPF		
WPF Custom Control Library	wpcustomcontrollib	[C#]
Common/WPF		
WPF User Control Library	wpfusercontrollib	[C#]
Common/WPF		
Windows Forms (WinForms) Application	winforms	[C#]
Common/WinForms		
Windows Forms (WinForms) Class library	winformslib	[C#]
Common/WinForms		
Worker Service	worker	[C#]
Common/Worker/Web		
Unit Test Project	mstest	[C#], F#, VB
Test/MSTest		
NUnit 3 Test Project	nunit	[C#], F#, VB
Test/NUnit		
NUnit 3 Test Item	nunit-test	[C#], F#, VB
Test/NUnit		
xUnit Test Project	xunit	[C#], F#, VB
Test/xUnit		
Razor Component	razorcomponent	[C#]
Web/ASP.NET		
Razor Page	page	[C#]
Web/ASP.NET		
MVC ViewImports	viewimports	[C#]
Web/ASP.NET		
MVC ViewStart	viewstart	[C#]
Web/ASP.NET		
Blazor Server App	blazorserver	[C#]
Web/Blazor		
ASP.NET Core Empty	web	[C#], F#
Web/Empty		

ASP.NET Core Web App (Model-View-Controller)	mvc	[C#], F#
Web/MVC		
ASP.NET Core Web App	webapp	[C#]
Web/MVC/Razor Pages		
ASP.NET Core with Angular	angular	[C#]
Web/MVC/SPA		
ASP.NET Core with React.js	react	[C#]
Web/MVC/SPA		
ASP.NET Core with React.js and Redux	reactredux	[C#]
Web/MVC/SPA		
Razor Class Library	razorclasslib	[C#]
Web/Razor/Library/Razor Class Library		
ASP.NET Core Web API	webapi	[C#], F#
Web/WebAPI		
ASP.NET Core gRPC Service	grpc	[C#]
Web/gRPC		
dotnet gitignore file	gitignore	
Config		
global.json file	globaljson	
Config		
NuGet Config	nugetconfig	
Config		
Dotnet local tool manifest file	tool-manifest	
Config		
Web Config	webconfig	
Config		
Solution File	sln	
Solution		
Protocol Buffer File	proto	
Web/gRPC		
Examples:		
dotnet new mvc --auth Individual		
dotnet new mstest		
dotnet new --help		

Dari informasi di atas dapat dilihat daftar template project yang dapat dibuat. Dari daftar tersebut terdapat 7 project dan 1 solution. Ada dua pilihan bahasa pemrograman yang dapat dipergunakan yaitu C# dan F#. Hal yang paling penting dari daftar template di atas adalah bagian "Short Name", karena nilai pada kolom ini yang dipergunakan untuk menentukan tipe project yang akan dibuat nanti.

Untuk melihat seluruh daftar template dapat dilihat perintah berikut ini.

```
dotnet new -all
```

Untuk membuat project digunakan perintah dengan sintaks sebagai berikut.

```
dotnet new [short name] -lang [language] -o [folder name]
```

Sebagai contoh untuk membuat aplikasi console dengan bahasa pemrograman Visual Basic (VB) digunakan perintah berikut ini.

```
dotnet new console -lang VB -o ConsoleVB
```

Jika opsi –lang tidak digunakan, maka akan digunakan nilai default dari opsi ini yaitu bahasa pemrograman C#.

Untuk membuat project aplikasi desktop tersedia 3 template yang dapat digunakan yaitu:

- Console Application, template ini digunakan untuk membuat aplikasi desktop tanpa antarmuka visual. Aplikasi ini adalah aplikasi command line

- WPF Application, template ini digunakan untuk membuat aplikasi desktop Windows Presentation Foundation (WPF).
- Windows Forms (WinForms) Application, template ini digunakan untuk membuat aplikasi desktop Windows Forms.

Selain itu juga disediakan template-template lainnya untuk mendukung pembangunan aplikasi desktop dengan template-template di atas, yaitu:

- Class library.
- WPF Class library.
- WPF Custom Control Library.
- WPF User Control Library.
- Windows Forms (WinForms) Class library.

Untuk membuat aplikasi dengan menggunakan template Console digunakan perintah berikut ini:

```
dotnet new console -o ConsoleCS
```

Hasilnya dapat dilihat di dalam folder ConsoleCS dengan isi sebagai berikut.

Name	Date modified	Type	Size
obj	1/30/2020 9:05 PM	File folder	
ConsoleCS.csproj	1/30/2020 9:05 PM	Visual C# Project file	1 KB
Program.cs	1/30/2020 9:05 PM	Visual C# Source file	1 KB

Gambar 12. Daftar file dan folder project Console Application.

Sedangkan untuk membuat project Windows Forms dapat dilakukan dengan perintah berikut ini.

```
dotnet new winforms -o WinFormCS
```

Hasilnya dapat dilihat di dalam folder WinFormCS dengan isi sebagai berikut ini.

Name	Date modified	Type	Size
obj	1/30/2020 9:23 PM	File folder	
Form1.cs	1/30/2020 9:23 PM	Visual C# Source file	1 KB
Form1.Designer.cs	1/30/2020 9:23 PM	Visual C# Source file	2 KB
Program.cs	1/30/2020 9:23 PM	Visual C# Source file	1 KB
WinFormCS.csproj	1/30/2020 9:23 PM	Visual C# Project file	1 KB

Gambar 13. Daftar file dan folder project Windows Form (WinForms) Application.

Yang terakhir adalah contoh untuk membuat project WPF Application dengan cara menggunakan perintah ini.

```
dotnet new wpf -o WpfCS
```

Hasilnya dapat dilihat pada gambar berikut ini.

Name	Date modified	Type	Size
obj	1/30/2020 9:36 PM	File folder	
App.xaml	1/30/2020 9:36 PM	Windows Markup File	1 KB
App.xaml.cs	1/30/2020 9:36 PM	Visual C# Source file	1 KB
AssemblyInfo.cs	1/30/2020 9:36 PM	Visual C# Source file	1 KB
MainWindow.xaml	1/30/2020 9:36 PM	Windows Markup File	1 KB
MainWindow.xaml.cs	1/30/2020 9:36 PM	Visual C# Source file	1 KB
WpfCS.csproj	1/30/2020 9:36 PM	Visual C# Project file	1 KB

Gambar 14. Daftar file dan folder project WPF Application.

Build

Selanjutnya adalah melakukan proses build atau kompilasi source code yang ada di dalam project. Perintah yang digunakan adalah sebagai berikut.

```
dotnet build
```

```
C:\WINDOWS\system32\cmd.exe

D:\NETCore\ConsoleCS>dotnet build
Microsoft (R) Build Engine version 16.5.0-preview-20064-06+86d9494e4 for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 49.07 ms for D:\NETCore\ConsoleCS\ConsoleCS.csproj.
  You are using a preview version of .NET Core. See: https://aka.ms/dotnet-core-preview
  ConsoleCS -> D:\NETCore\ConsoleCS\bin\Debug\netcoreapp3.1\ConsoleCS.dll

Build succeeded.
  0 Warning(s)
  0 Error(s)

Time Elapsed 00:00:06.01

D:\NETCore\ConsoleCS>
```

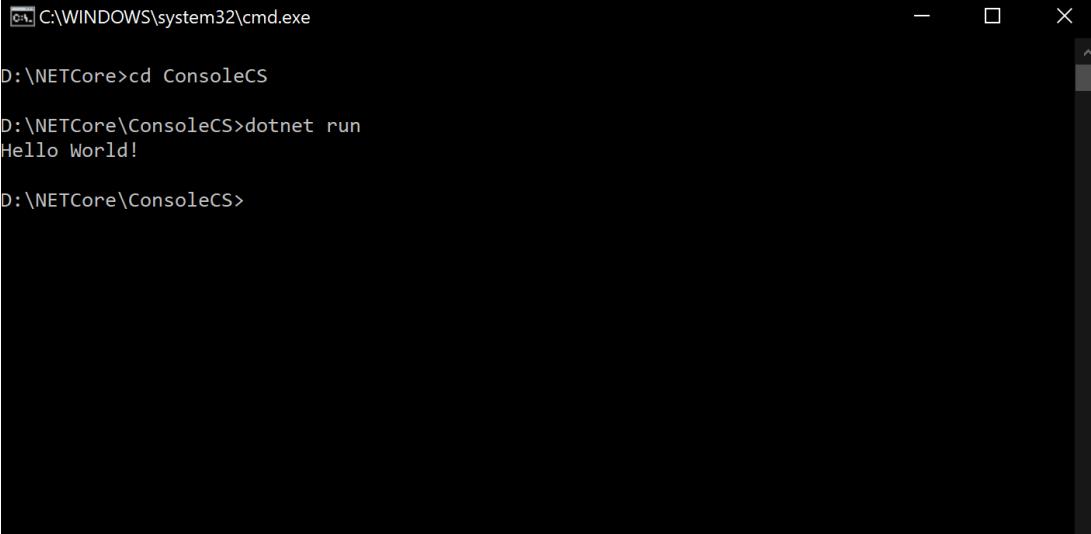
Gambar 15. dotnet build.

Run

Untuk menjalankan aplikasi digunakan perintah ini.

```
dotnet run
```

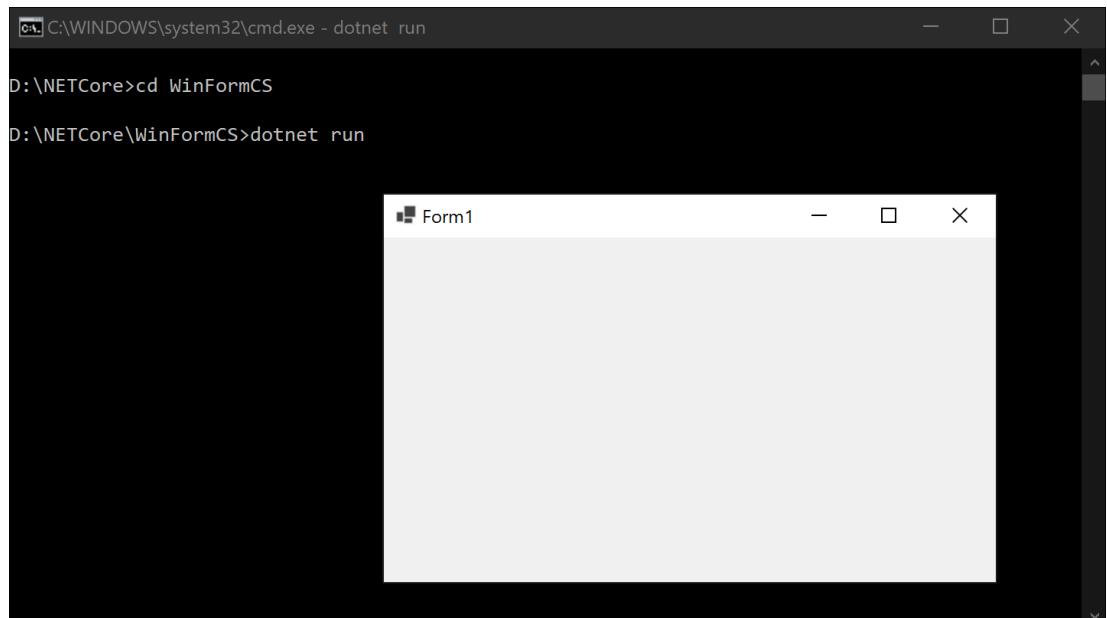
Pada sub bab Membuat Project, telah dibuat 3 project. Pada sub bab ini akan diperlihatkan cara dan hasil ketika masing-masing project tersebut dijalankan. Untuk menjalankan project Console Application pada folder ConsoleCS, maka terlebih dulu masuk ke dalam folder ConsoleCS pada Command Prompt. Kemudian ketikkan perintah di atas untuk menjalankan project ini. hasilnya dapat dilihat pada gambar di bawah ini. Pada gambar dapat dilihat kalimat "Hello World" yang menjadi output dari aplikasi console ini.



```
C:\WINDOWS\system32\cmd.exe
D:\NETCore>cd ConsoleCS
D:\NETCore\ConsoleCS>dotnet run
Hello World!
D:\NETCore\ConsoleCS>
```

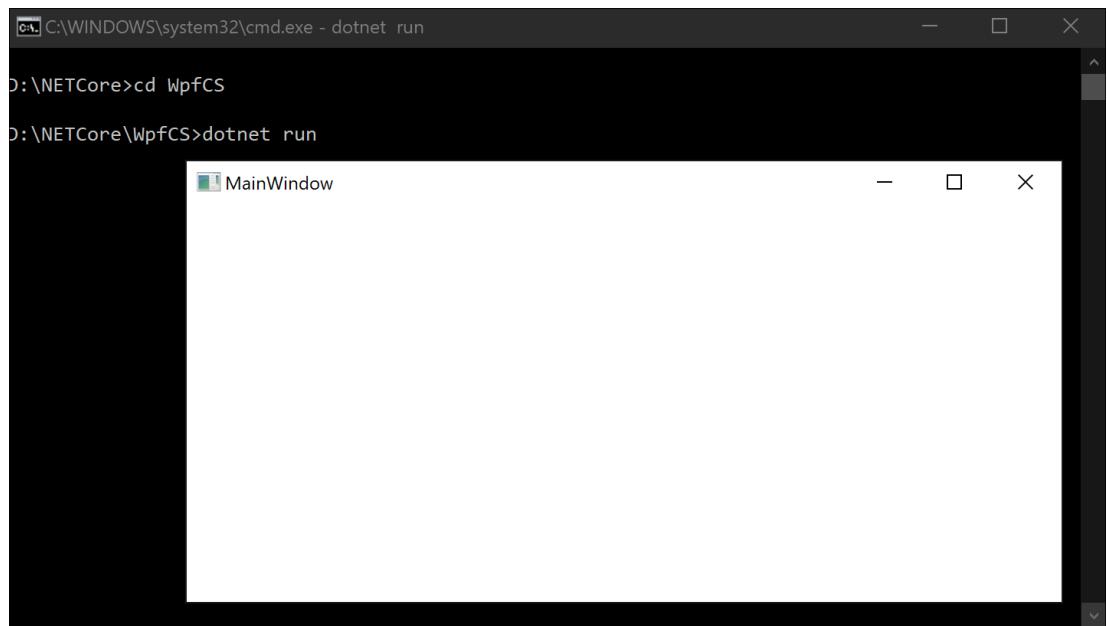
Gambar 16. Menjalankan project Console Application - ConsoleCS.

Sedangkan untuk menjalankan project Windows Forms (WinForms) Application dengan cara masuk ke folder WinFormCS. Kemudian ketikkan perintah “dotnet run” untuk mengeksekusi aplikasi. Hasilnya dapat dilihat pada gambar di bawah ini. Pada gambar ini dapat dilihat aplikasi berupa window baru dengan title Form1.



Gambar 17. Menjalankan project Windows Forms (WinForms) Application.

Sama seperti kedua cara di atas, lakukan hal yang sama untuk menjalankan aplikasi WPF yaitu dengan masuk ke folder tempat project disimpan yaitu WpfCS dan ketika perintah untuk menjalankan aplikasi. Hasilnya dapat dilihat pada gambar di bawah ini. Pada gambar ini dapat dilihat aplikasi berupa window baru dengan title MainWindow.



Gambar 18. Menjalankan WPF Application.

Contoh di atas hanya untuk menunjukkan bahwa .NET Core SDK memiliki tool berupa command line untuk membuat dan menjalankan project. Sedangkan pada bab selanjutnya akan digunakan Visual Studio 2019 untuk membuat dan menjalankan project.

3

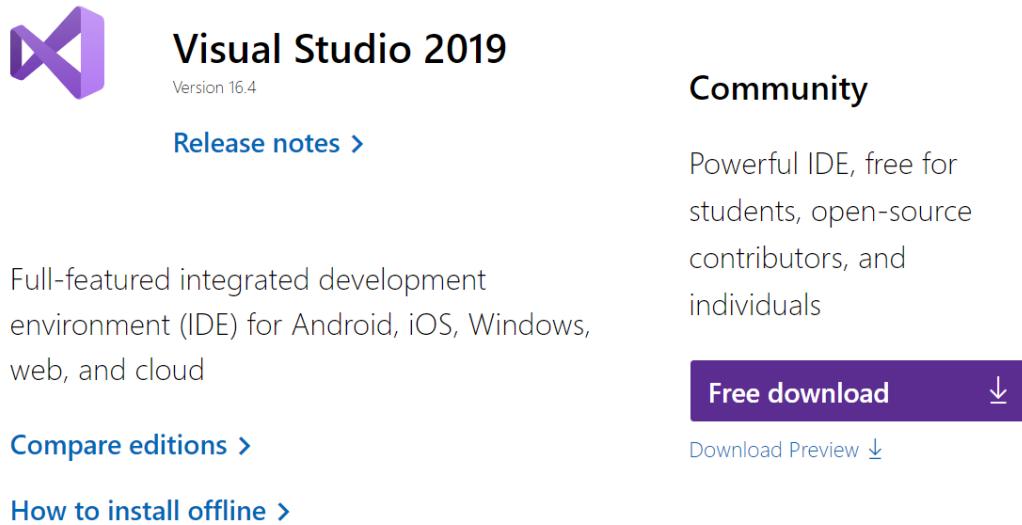
Visual Studio 2019

Pada bab ini diberikan panduan penggunaan Visual Studio dimulai dengan membuat solution dan project. Kemudian dilanjutkan proses eksekusi. Panduan dicontohkan dengan menggunakan Visual Studio 2019.

Installasi

Installer Visual Studio 2019 dapat diunduh di <https://www.visualstudio.com/downloads/>. Bagi pembaca yang belum memiliki lisensinya dapat mengunduh Visual Studio Community 2019 yang bersifat gratis. Saat buku ini ditulis versi Visual Studio 2019 yang stabil adalah 16.4. Namun pada buku ini kami menggunakan versi Preview, yang link downloadnya dapat dilihat di bawah tombol “Free download” di bawah ini.

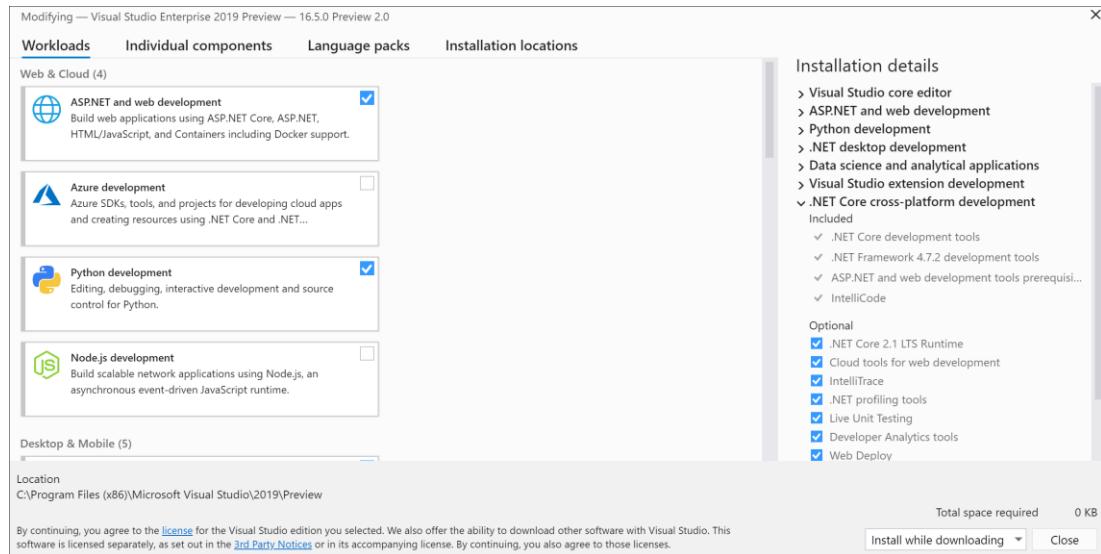
Saat buku ini ditulis (Januari 2020), digunakan Visual Studio 2019 15.6.0 Preview 2.0. Hal ini bertujuan untuk menggunakan fitur .NET Desktop Development untuk Windows Forms Application dan WPF Application yang baru didukung pada versi ini. Namun saat pembaca membaca buku ini mungkin telah keluar versi yang lebih baru dan stabil (bukan Preview) yang telah mendukung pengembangan aplikasi desktop dengan .NET Core.



Gambar 19. Halaman untuk mengunduh Visual Studio 2019.

Setelah file installer diunduh, kemudian diklik double untuk menjalankan. Installasi Visual Studio bersifat online, sehingga file installernya berukuran kecil. File-file yang diperlukan akan diunduh secara online saat installasi. Setelah file-file yang diperlukan selesai diunduh, maka dilanjutkan dengan instalasi file-file tersebut.

Berikut adalah antarmuka installer.



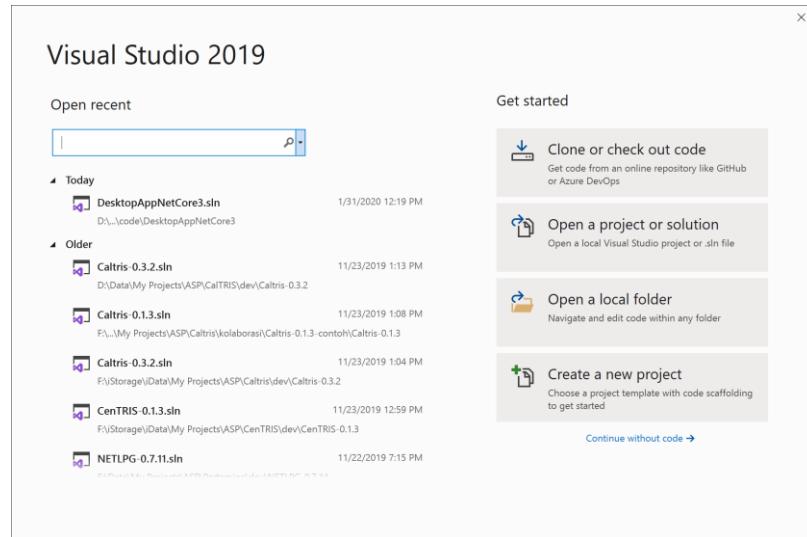
Gambar 20. Visual Studio 2019 - antarmuka installer.

Untuk menggunakan Visual Studio sebagai tool pengembang aplikasi desktop maka paket yang perlu dipilih adalah:

- Dekstop & Mobile > .NET desktop development.
- Other Toolsets > .NET Core cross-platform development.

Antarmuka

Pada sub bab ini memberikan penjelasan tentang antarmuka Visual Studio 2019. Visual Studio akan menampilkan Start Page seperti pada gambar di bawah ini.



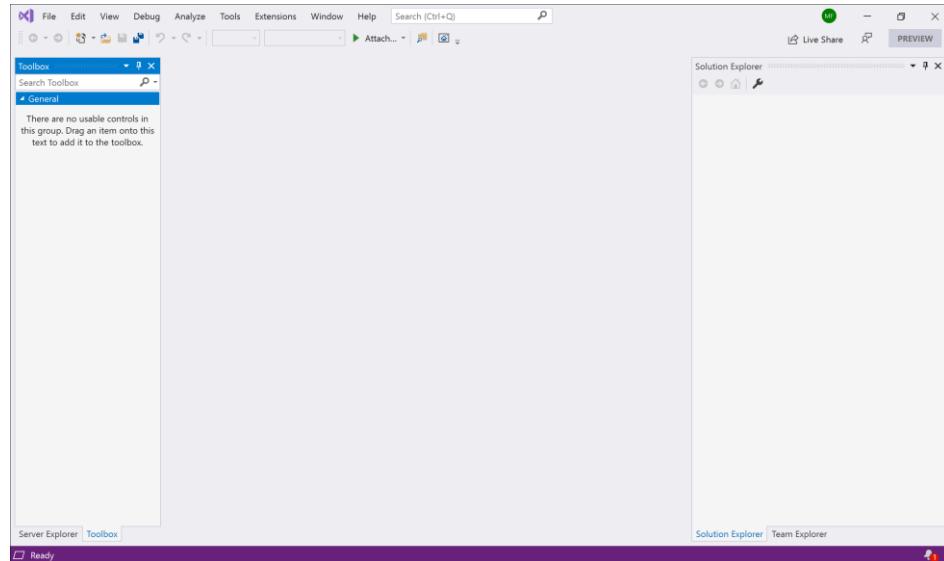
Gambar 21. Visual Studio 2019 - Start Page.

Pada halaman Start Page ini, user dapat melakukan aksi-aksi berikut ini:

- Open recent, membuka project dari daftar history project yang pernah dibuka sebelumnya.

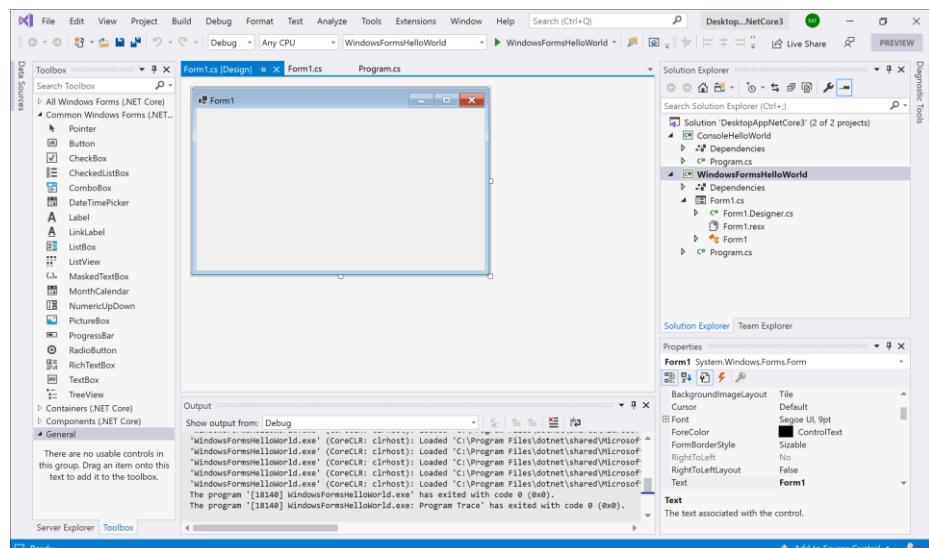
- Clone or check out code, membuka dan mendapatkan source code dari online repository seperti GitHub atau Azure DevOps.
- Open a project or solution, membuka project atau solution yang berada di ruang penyimpanan komputer user.
- Open a local folder, membuat dan mengedit kode yang berada pada suatu folder.
- Create a new project, membuat project dengan menggunakan template project dengan code scaffolding yang telah disediakan.

Pilihan lain adalah “Continue without code”, yaitu melanjutkan ke halaman selanjutnya tanpa pemilihan kode program yang akan dibuka. Pembaca dapat memilih opsi ini dan akan melihat antarmuka seperti pada gambar di bawah ini.



Gambar 22. Visual Studio 2019 - Antarmuka tanpa kode yang dipilih.

Sedangkan jika pengguna memilih kode untuk dibuka saat berada di halaman Start Page, maka antarmukanya adalah seperti pada gambar di bawah ini.



Gambar 23. Visual Studio 2019 - antarmuka dengan kode.

Seperti aplikasi desktop Windows pada umumnya, terdapat fitur yang umum yang ada yaitu:

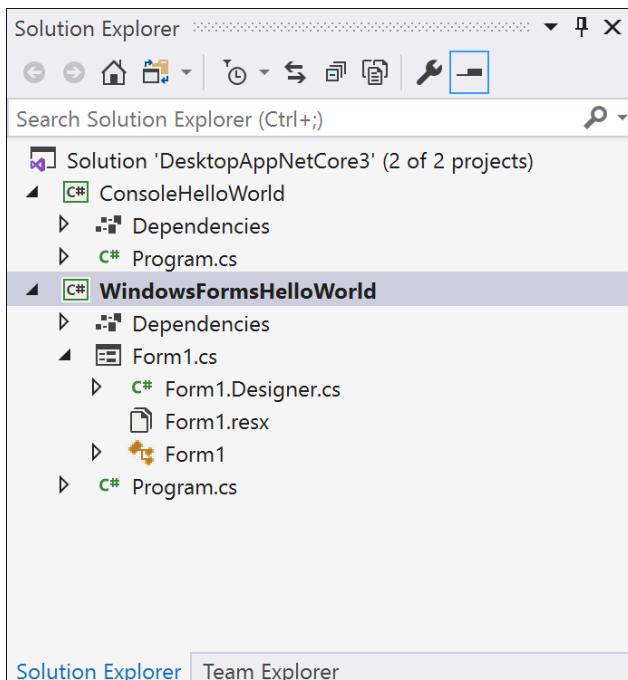
- Menu.
- Toolbar.

Setelah salah satu project dibuka maka akan dilihat antarmuka seperti pada gambar di atas. Antarmuka di atas terbagi atas beberapa bagian yaitu:

- Solution Explorer.
- Editor.
- Toolbox.
- Properties.
- Output.
- Error List

Solution Explorer

Solution Explorer dapat dilihat pada bagian kanan atas pada Visual Studio 2019.



Gambar 24. Visual Studio 2019 – Solution Explorer.

Solution Explorer menampilkan Solution. Solution dapat berisi satu atau lebih Project. User dapat melihat daftar file yang dimiliki oleh project. User dapat memilih file yang ingin dilihat.

Editor

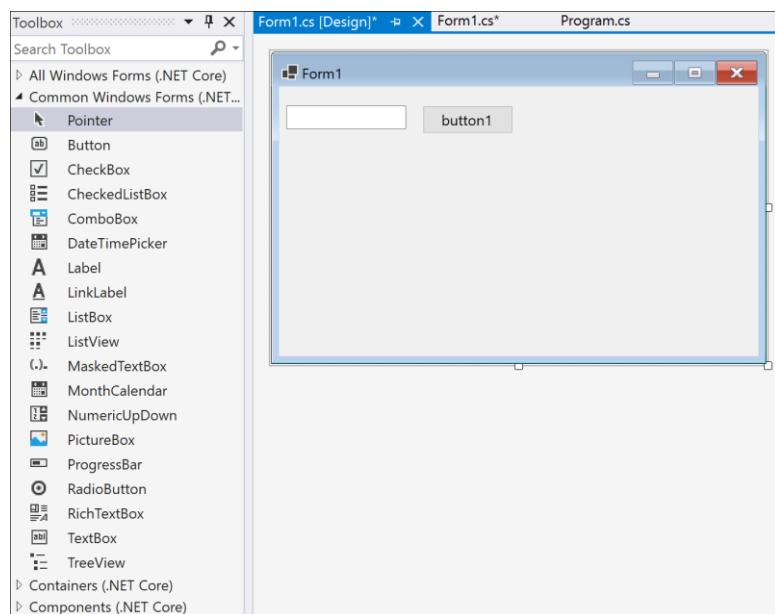
Editor berfungsi untuk menambah atau memodifikasi item atau kode baik secara visual ataupun kode dari file yang dipilih. Pada gambar di bawah ini contoh Code Editor. Code Editor berfungsi untuk mengedit file text.

The screenshot shows the Visual Studio 2019 Code Editor with three tabs: Form1.cs [Design], Form1.cs, and Program.cs. The Program.cs tab is active, displaying the following C# code:

```
1  using System;
2
3  namespace ConsoleHelloWorld
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Hello World!");
10             Console.WriteLine("");
11             Console.ReadKey();
12         }
13     }
14 }
15
```

Gambar 25. Visual Studio 2019 - Code Editor.

Saat file yang dibuka adalah antarmuka dari suatu aplikasi seperti pada contoh di bawah ini, maka Editor akan beralih fungsi sebagai Visual Editor.

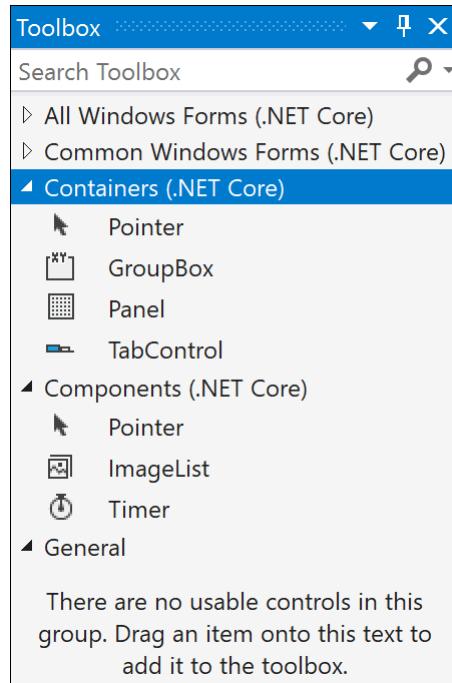


Gambar 26. Visual Studio 2019 – Visual Editor.

Visual Editor memberikan kemampuan pada user untuk melakukan modifikasi antarmuka dengan cara drag-n-drop item atau komponen visual dari Toolbox.

Toolbox

Tool akan menampilkan item-item yang sesuai dengan file yang sedang aktif.

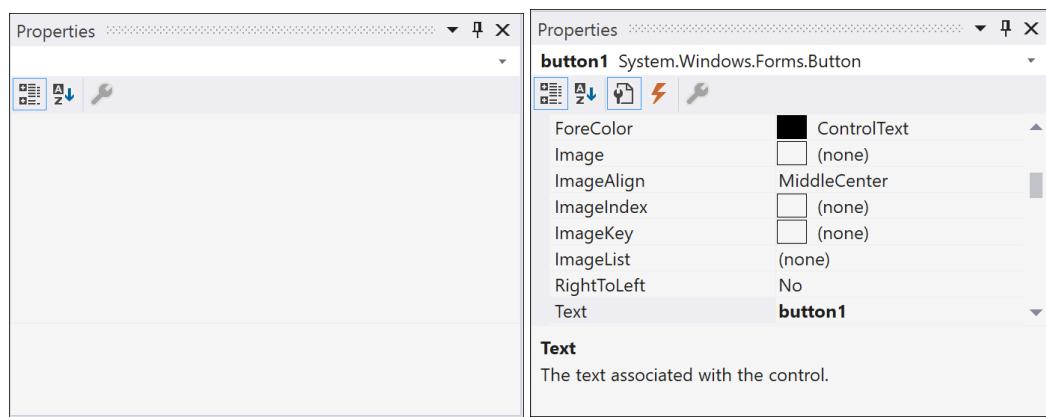


Gambar 27. Visual Studio - Toolbox.

Sebagai contoh, ketika user sedang membuat Windows Form Application maka akan ditampilkan daftar item-item seperti pada gambar di atas. Namun jika user sedang membuat aplikasi lain seperti WPF Application atau ASP.NET Web Application maka Toolbox akan menampilkan daftar item yang berbeda lagi.

Properties

Bagian Properties menampilkan detail informasi dari item yang dipilih pada Editor. Biasanya informasi yang ditampilkan pada bagian ini berkaitan dengan item yang dipilih saat mode Visual Editor.

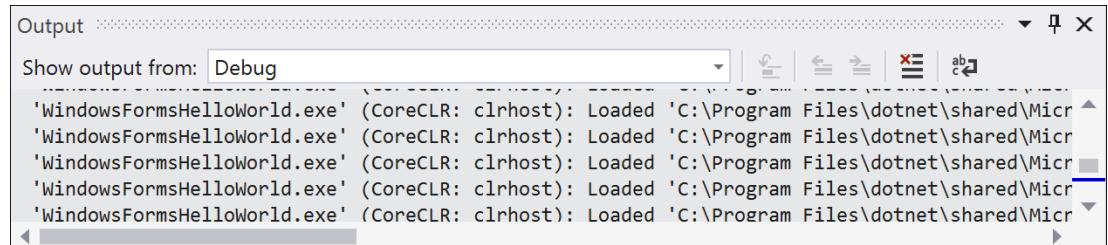


Gambar 28. Visual Studio 2019 – Properties.

Gambar di sebelah kanan tidak menampilkan informasi, hal ini terjadi ketika Editor sedang menampilkan file text. Saat mode Editor Visual, maka dapat dilihat property-property yang dimiliki oleh file yang sedang dibuka.

Output

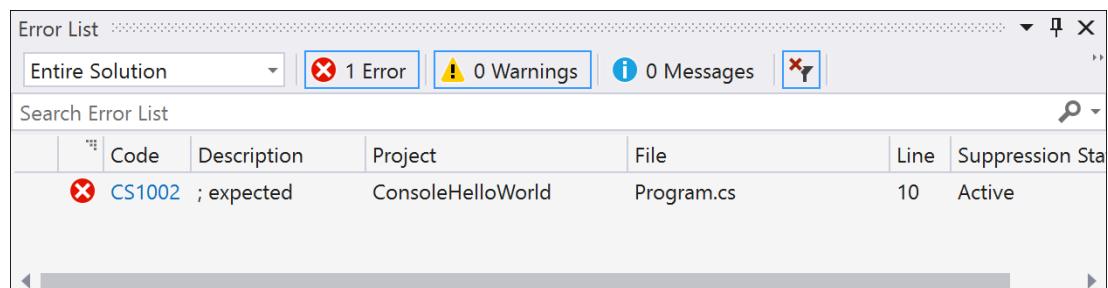
Bagian Output menampilkan keluaran dari proses yang dijalankan pada Visual Studio. Seperti proses build atau kompilasi kode. Selain itu bagian ini juga menampilkan keluaran dari program yang dijalankan, sebagai contoh keluaran dari program berbasis Command Line.



Gambar 29. Visual Studio 2019 – Output.

Error List

Bagian Error List menampilkan error yang terjadi baik dari kode yang ditulis, atau error yang terjadi karena program yang dibuat melakukan kesalahan atau mendapatkan pesan error dari sistem lain. Sebagai ketika program yang dibuat tidak bisa melakukan koneksi ke database.



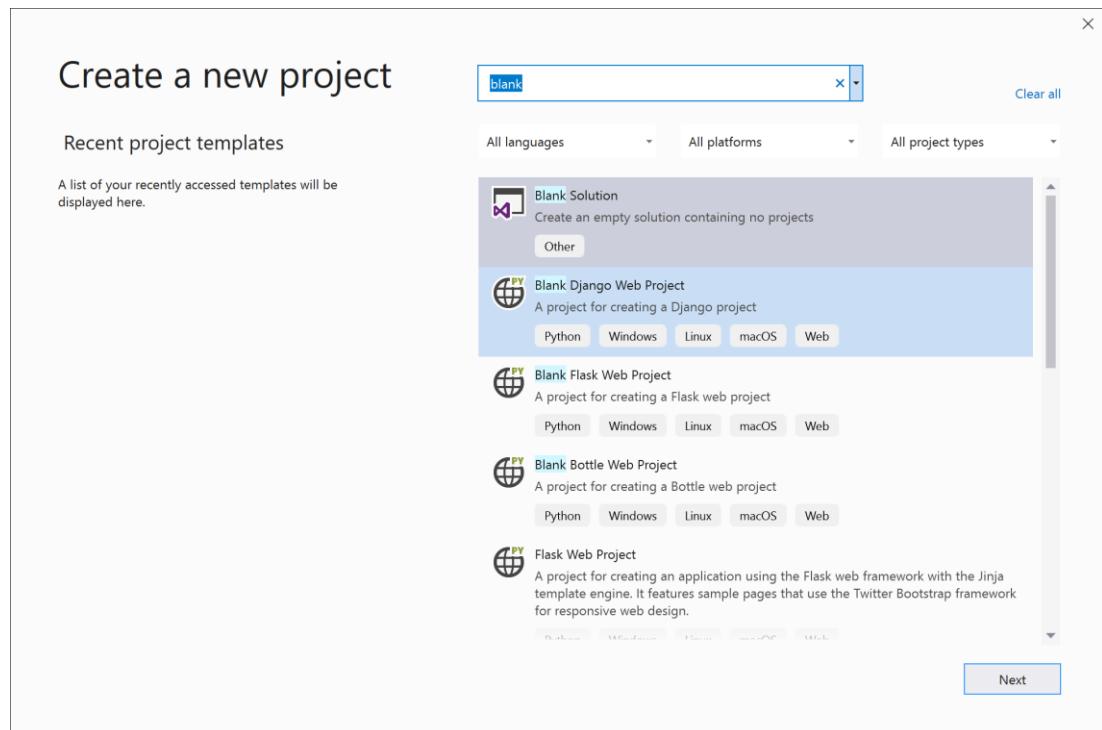
Gambar 30. Visual Studio 2019 – Error List.

Solution & Project

Solution adalah tempat yang dapat berisi banyak project. Setiap solution akan memiliki sebuah file *.sln. Setiap solution akan disimpan ke dalam folder dengan nama yang sama dengan nama solutionnya. Pada Visual Studio, solution dapat dilihat pada bagian Solution Explorer.

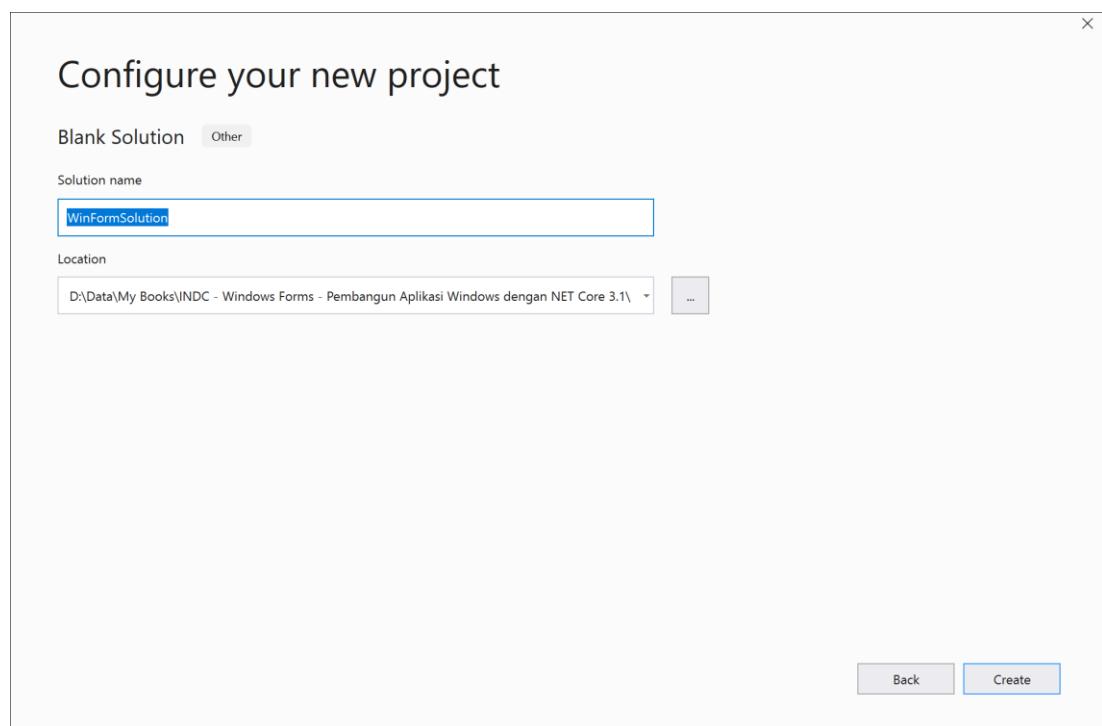
Solution

Pada Visual Studio, solution dapat dibuat dengan dua cara. Yang pertama adalah dengan membuat solution kosong. Langkah pertama adalah dengan membuat project baru dengan cara memilih File > New > Project pada menu. Selanjutnya akan dilihat window seperti pada gambar di bawah ini. Ketik "blank" pada kolom pencarian.



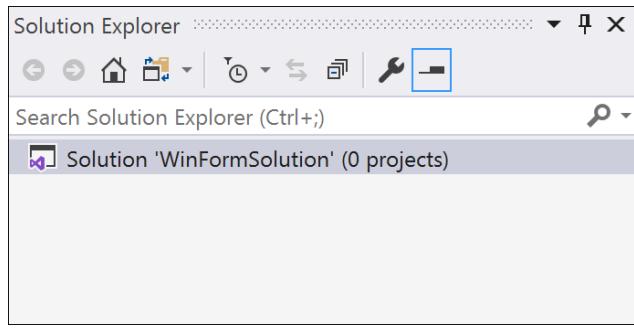
Gambar 31. Visual Studio 2019 - Create a new project - Blank Solution.

Pilihan Blank Solution (Create an empty solution containing no projects), kemudian klik tombol Next. Kemudian ditampilkan window Configure your new project seperti pada gambar di bawah ini.



Gambar 32. Visual Studio 2019 - Configure your new project.

Pada kolom Solution name beri nama solution yang diinginkan. Pada contoh di atas digunakan nama WinFormSolution. Kemudian klik tombol Create. Maka hasilnya dapat dilihat pada Solution Explorer.

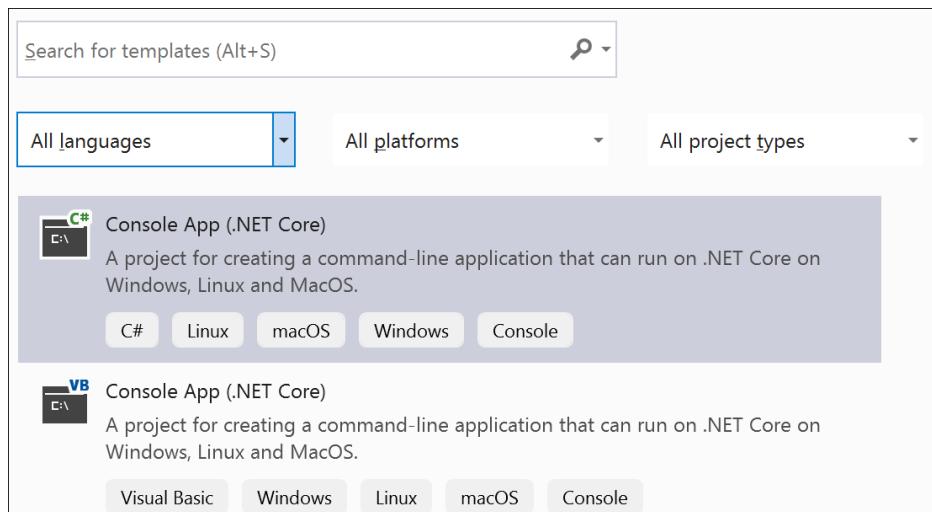


Gambar 33. WinFormSolution.

Cara kedua adalah dengan membuat project baru. Dengan membuat sebuah project baru maka secara otomatis akan dibuat sebuah solution. Namun nama solution dan nama project yang dibuat akan menjadi sama. Tetapi penulis menyarankan untuk menggunakan cara pertama, karena sebaiknya nama solution dibuat berbeda dengan nama project didalamnya.

Project

Visual Studio memiliki banyak template project yang dapat dipilih dan digunakan. Jika diperhatikan pada window Add a new project di Gambar 31. Maka dapat dilihat pilihan project berdasarkan bahasa pemrograman, platform dan tipe project.



Gambar 34. Pilihan project berdasarkan language, platform dan project types.

Jika diklik pada dropdownlist All languages maka dapat dilihat pilihan bahasa pemrograman yang didukung, yaitu:

- C#.
- C++.
- F#.
- Java.
- JavaScript.
- Python.
- Query Language.
- TypeScript.
- Visual Basic.

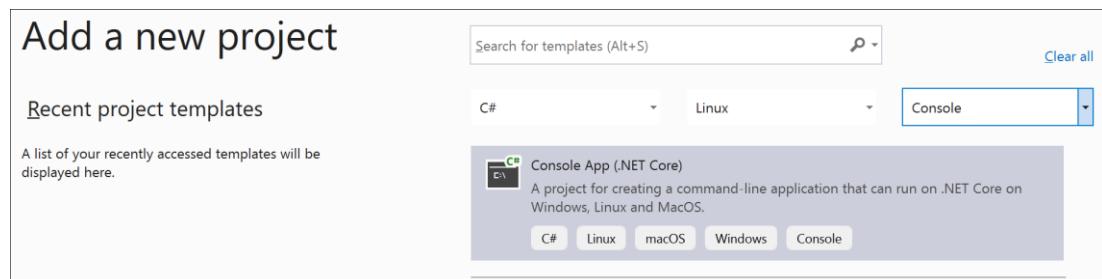
Sedangkan jika dropdownlist All platforms diklik maka platform yang didukung adalah:

- Android.
- Azure.
- iOS.
- Linux.
- macOS.
- tvOS.
- Windows.
- XBox.

Untuk setiap bahasa pemrograman dan platform di atas dapat dibuat project dengan tipe-tipe sebagai berikut, yaitu:

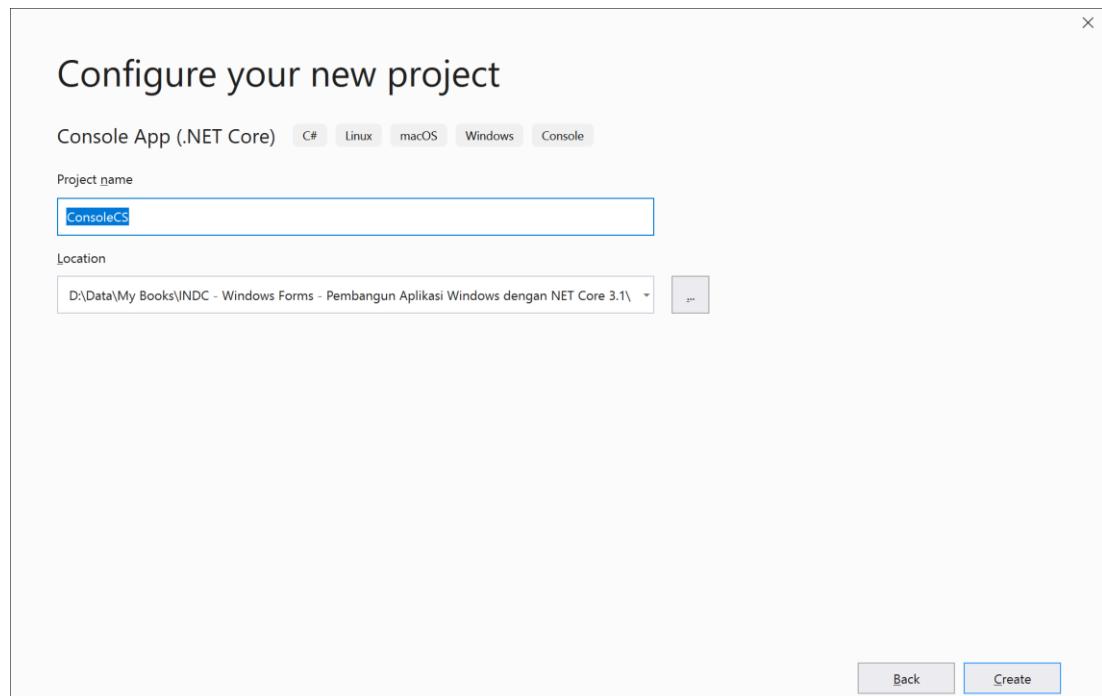
- Add-in.
- Cloud.
- Console.
- Desktop.
- Extensions.
- Games.
- IoT.
- Library.
- Machine Learning.
- Mobile.
- Office.
- Other.
- Plugin.
- Service.
- Test.
- UWP.
- VSSDK.
- Web.

Sebagai contoh akan dibuat project di dalam WinFormSolution yang telah dibuat sebelumnya. Maka pada Solution Explorer, klik kanan pada WinFormSolution kemudian pilih Add > New Project. Kemudian akan ditampilkan window Add a new project. Jika ingin dibuat project Console yang akan dijalankan pada platform Linux dan ditulis dengan bahasa pemrograman C#. Maka user dapat memilih nilai-nilai tersebut seperti pada gambar berikut ini. Pada gambar ini dapat dilihat bahwa template project ini dapat dijalankan pada platform Linux, macOS dan Windows.



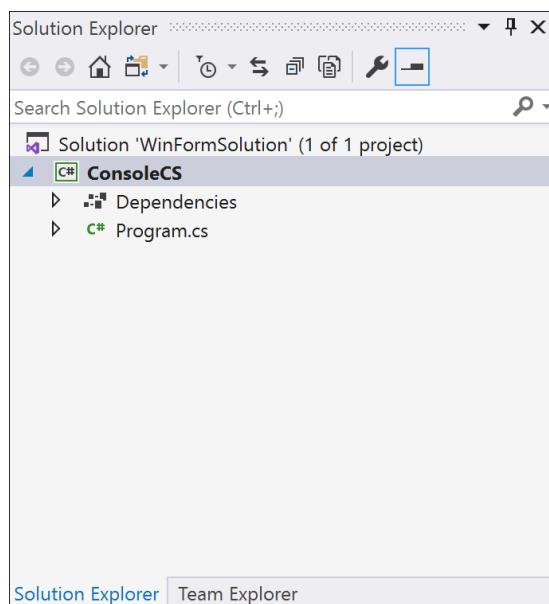
Gambar 35. Membuat project Console App (.NET Core).

Kemudian klik tombol Next. Selanjutnya berikan nama project yang diinginkan. Misalnya diberikan nama ConsoleCS. Kemudian klik tombol Create.



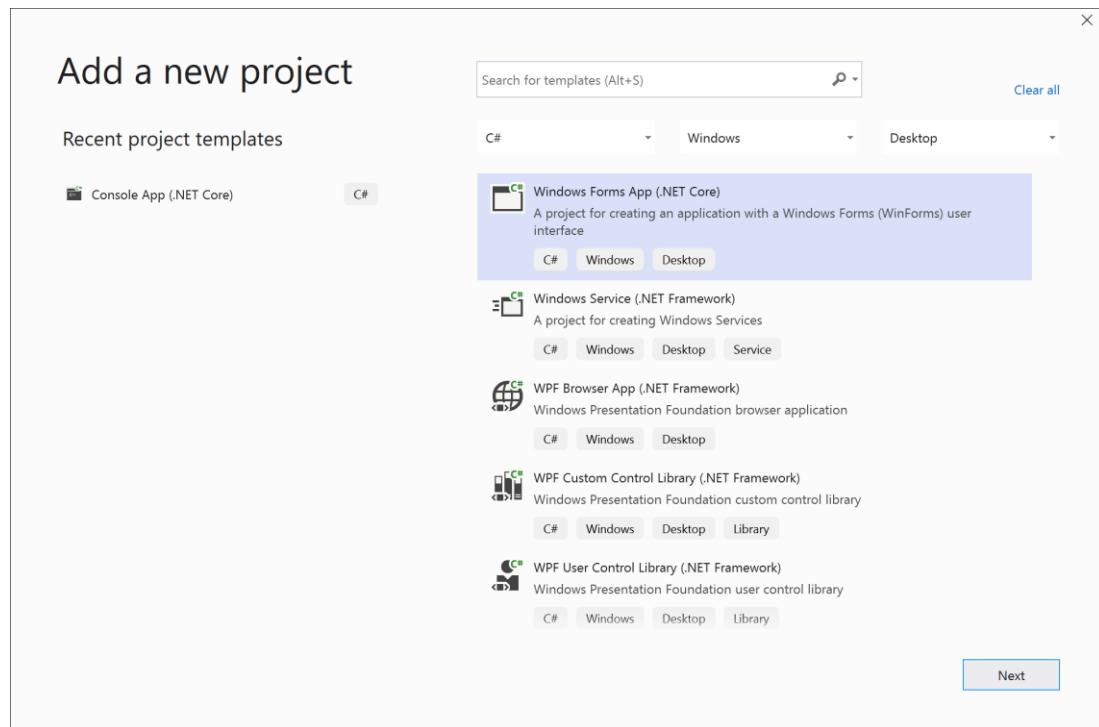
Gambar 36. Visual Studio 2019 - Membuat project ConsoleCS.

Hasilnya dapat dilihat tambahan project ConsoleCS pada WinFormSolution seperti terlihat pada gambar di bawah ini.



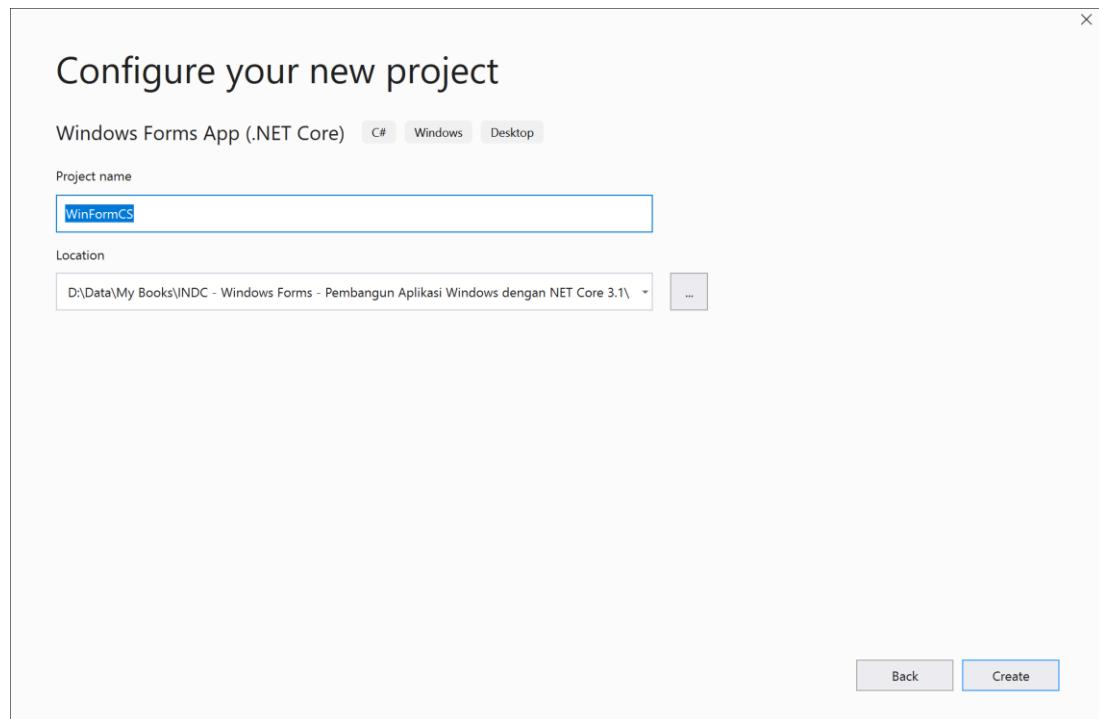
Gambar 37. Visual Studio 2019 - Project ConsoleCS pada Solution Explorer.

Contoh lain akan dibuat project aplikasi desktop dengan bahasa memrograman C#. Maka pilih C# pada dropdown languages dan pilih Desktop pada dropdown project types seperti pada gambar di bawah ini. Kemudian pilih tombol Next. Pada template project yang disorot pada gambar di bawah ini dapat dilihat bahwa template project Windows Forms App (.NET Core) hanya dapat dijalankan pada platform Windows.



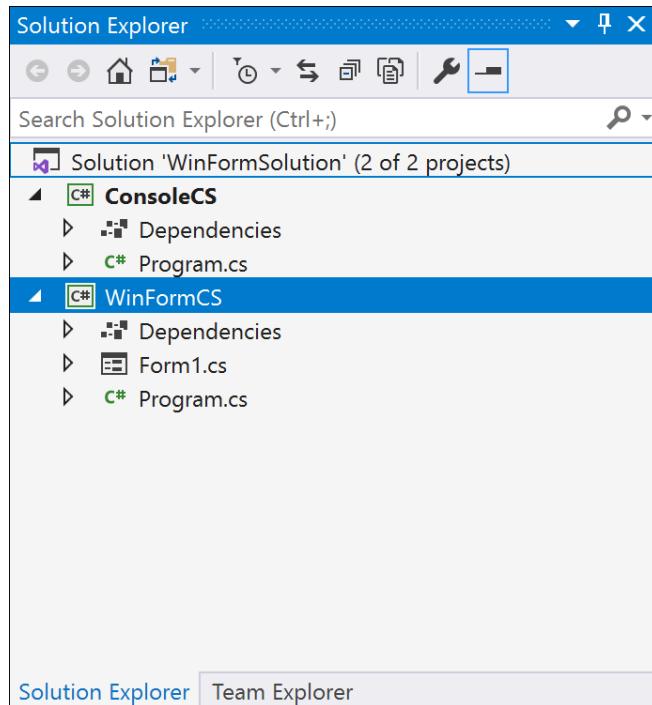
Gambar 38. Visual Studio 2019 - Membuat project Windows Forms App (.NET Core).

Setelah tombol Next diklik maka dapat dilihat window berikut ini. Isikan WinFormCS pada kolom Project name, kemudian klik tombol Create.



Gambar 39. Visual Studio 2019 - memberi nama project WinFormCS.

Kemudian dapat dilihat project WinFormCS pada Solution Explorer seperti pada gambar di bawah ini.



Gambar 40. Visual Studio 2019 - Project WinFormCS pada Solution Explorer.

Dari penjelasan di atas maka sangat dimungkinkan untuk memiliki sebuah solution yang didalamnya berisi banyak project seperti:

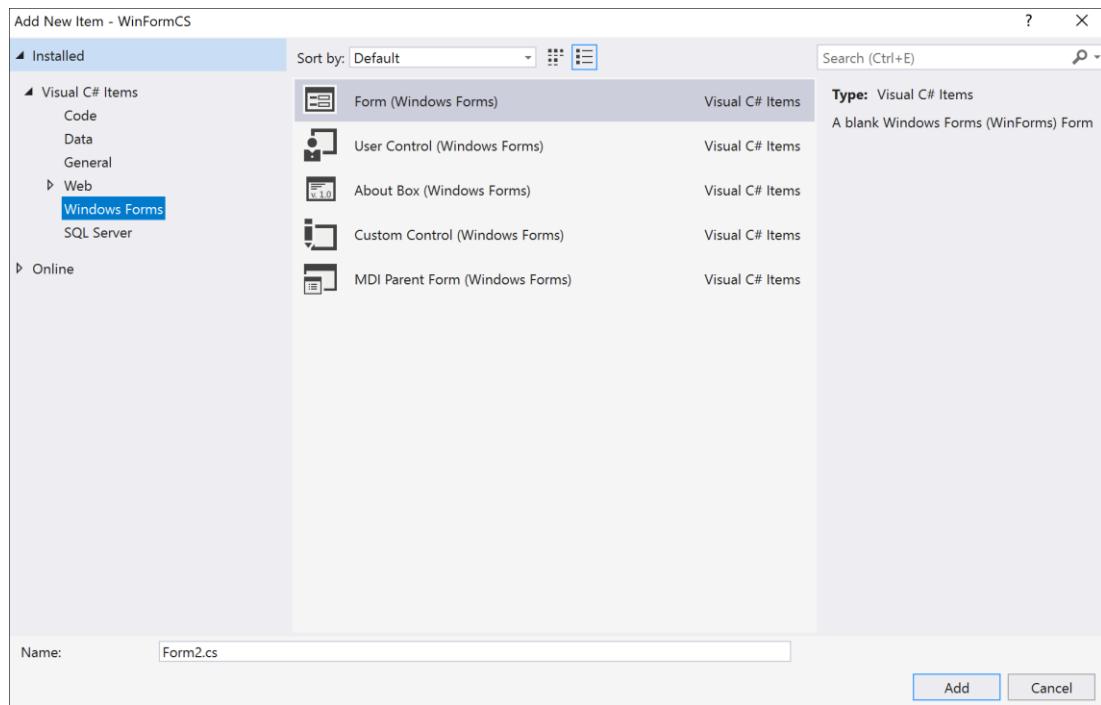
- Project Console App.
- Project Windows Forms App.
- Dan lain-lain.

Item

Item adalah sesuatu yang dapat ditambahkan ke dalam project. Item dapat berupa:

- Class.
- Image.
- Resource.
- Dan lain-lain.

Untuk menambahkan item pada project dapat dilakukan dengan cara klik kanan pada project kemudian pilih Add > New Item.



Gambar 41. Visual Studio 2019 - Add New Item.

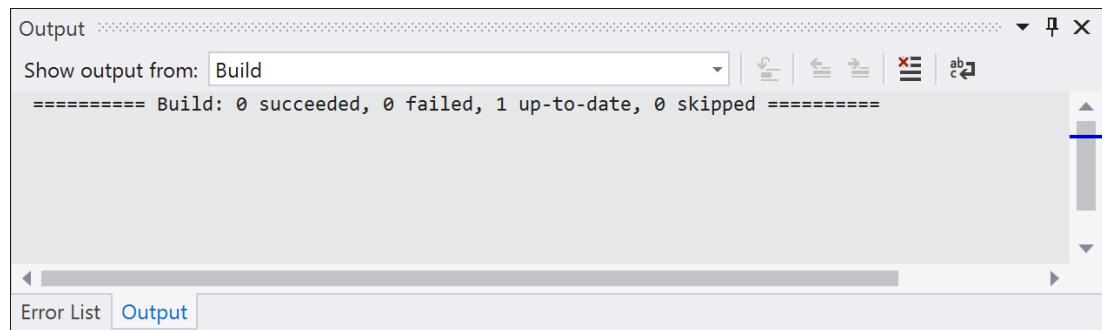
Kemudian pilih template file yang diinginkan, kemudian berikan nama item pada kolom Name. Item yang ditampilkan pada window itu akan sesuai dengan tipe project.

Build & Debug

Pada pembangunan aplikasi desktop atau console, file-file kode program akan dikompilasi menjadi file executable dengan format binary. Untuk aplikasi web, file-file kode program akan dideploy ke web server kemudian setelah web server aktif maka aplikasi web tersebut dapat dijalankan dengan menggunakan web browser yang mengakses alamat web server tersebut.

Proses kompilasi dikenal juga dengan istilah Build. Untuk melakukan proses ini dengan Visual Studio, pengguna dapat melakukannya dengan cara klik kanan pada solution yang ada pada Solution Explorer kemudian pilih Build Solution. Dengan cara ini maka proses build akan dilakukan pada seluruh project yang ada di dalam solution. Jika proses build hanya ingin dilakukan pada salah satu project saja, maka lakukan klik kanan pada project yang diinginkan pada Solution Explorer kemudian pilih Build. Status proses ini dapat dilihat pada bagian Output seperti yang dapat dilihat pada Gambar 42.

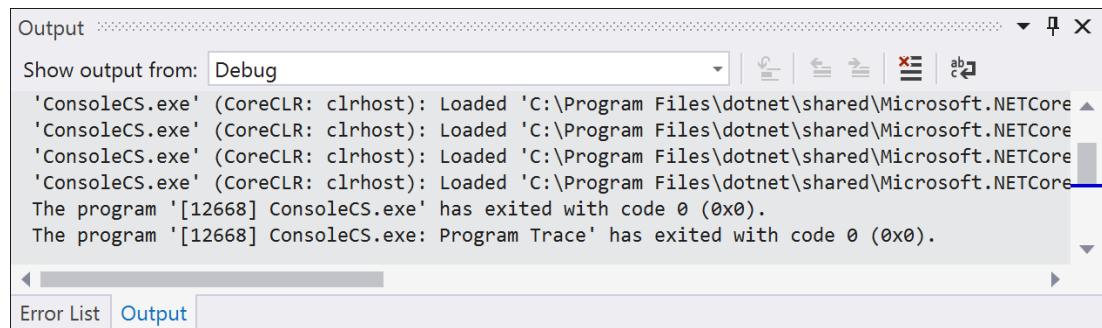
Proses Debug adalah proses untuk menjalankan StartUp project atau project yang diinginkan. StartUp project biasanya adalah project yang pertama kali dibuat pada solution. Pada Solution Explorer, StartUp project dapat dibedakan dengan nama project dengan cetak tebal seperti yang terlihat pada project ConsoleApp1 pada gambar di atas. Jika proses debug dilakukan dengan cara memilih Debug > Start Debugging maka StartUp project akan dijalankan oleh Visual Studio.



Gambar 42. Visual Studio 2019 – Output proses build.

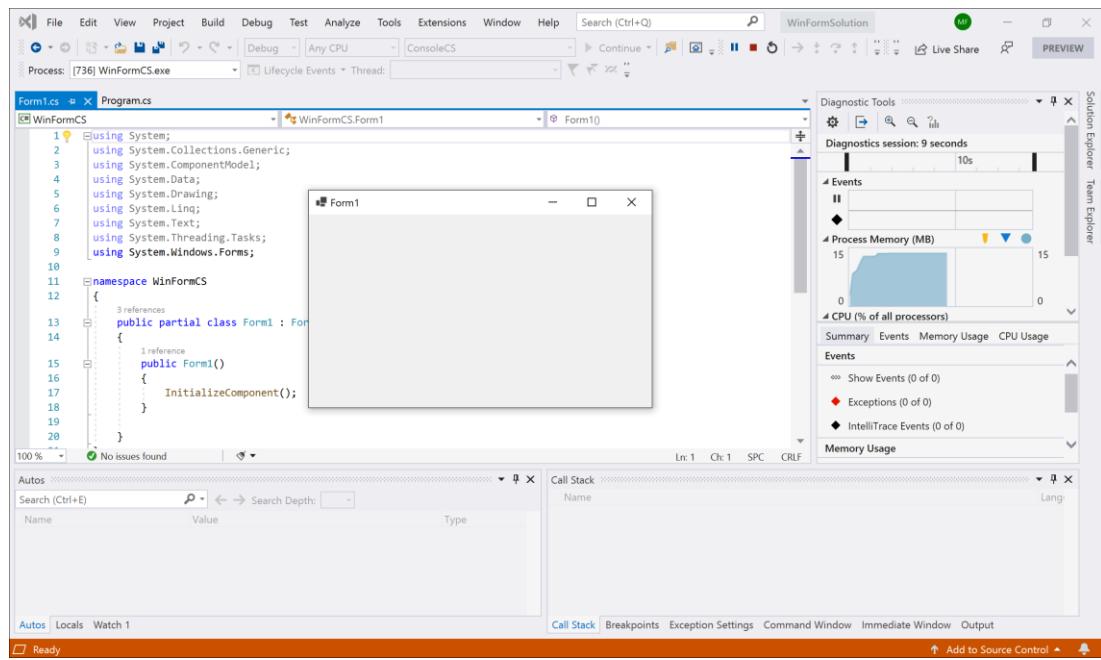
Sedangkan jika ingin melakukan proses debug pada project yang diinginkan saja maka klik kanan pada project yang ada di Solution Explorer kemudian pilih Debug > Start new instance.

Status proses debug akan dapat dilihat pada bagian Output. Sebagai contoh dilakukan proses debug pada project ConsoleCS.



Gambar 43. Visual Studio 2019 – Output proses debug.

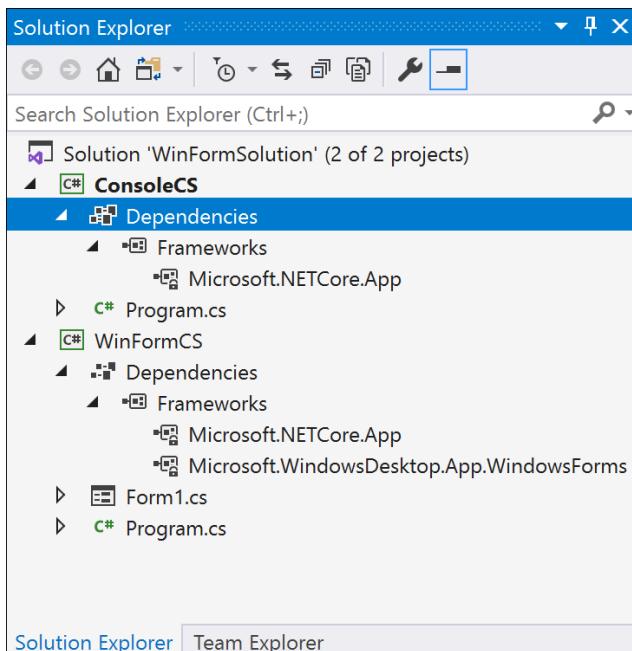
Contoh lain, untuk menjalankan project WinFormCS. Maka klik kanan pada project WinFormCS kemudian pilih Build > Start new instance. Maka akan ditampilkan window Form1 seperti pada Gambar 44. Selain itu user juga dapat melihat tampilan Visual Studio mengalami perubahan. Sekarang terlihat terdapat Diagnostic Tool pada area sebelah kanan yang memberikan informasi penggunaan CPU dan memory. Untuk kembali ke tampilan semula, maka tutup window Form1.



Gambar 44. Visual Studio 2019 - Menjalankan project WinFormCS.

Depedencies

Sebuah project memiliki library default. Library default ini akan berbeda-beda antar setiap tipe project. Sebagai contoh project Console App akan memiliki library yang berbeda jika dibandingkan dengan project Windows Form App. Untuk melihat library apa saja yang dimiliki oleh suatu project, maka dapat dilihat dengan cara membuka Depedencies pada project.



Gambar 45. Visual Studio 2019 - Depedencies.

Pada gambar di atas dapat dilihat bahwa project tipe Console App memiliki library default dengan nama namespace berikut ini:

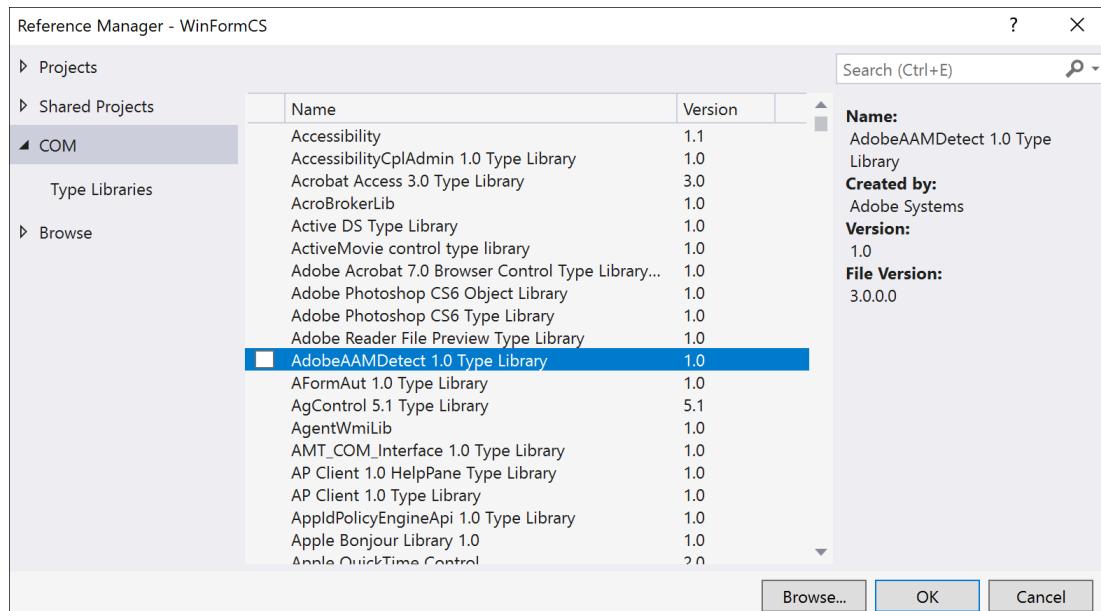
- Microsoft.NETCore.App.

Sedangkan project tipe Windows Forms App memiliki library default dengan nama namespace berikut ini:

- Microsoft.NETCore.App.
- Microsoft.WindowsDesktop.App.WindowsForms.

Selain library default tersebut, pengguna dapat menambahkan library lain yang tersedia atau library dari suatu project yang sedang dibangun.

Untuk menambahkan library pada project dapat dilakukan dengan cara klik kanan pada Dependencies di project yang diinginkan. Kemudian pilih Add Reference.



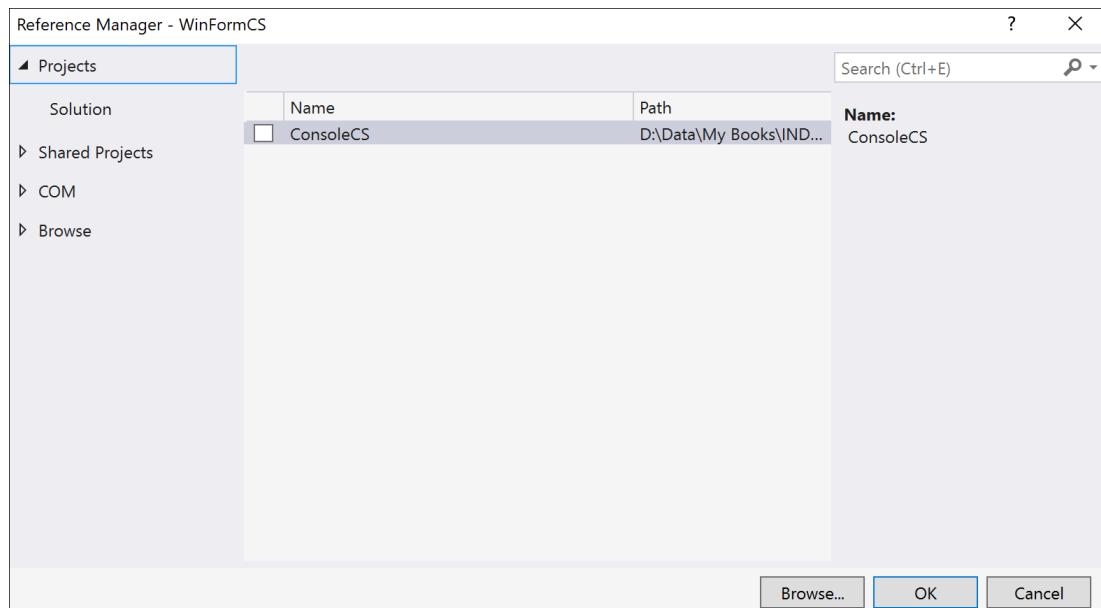
Gambar 46. Visual Studio 2019 - Reference Manager - COM.

User dapat memilih library dari 3 tipe, yaitu:

- Projects
- Shared Projects.
- COM.

Pada gambar di atas adalah contoh ketika tab COM aktif. Dapat dilihat daftar assembly yang tersedia dan juga yang telah digunakan pada project. COM yang telah digunakan pada project memiliki tanda check seperti gambar di atas.

Jika pada solution sedang dikembangkan project tipe class library, maka project tersebut dapat ditambahkan pada project dengan mengklik tab Projects. Maka dapat dilihat daftar project yang ada pada solution seperti pada gambar di bawah ini (daftar project akan berbeda dengan pembaca, dan mungkin kosong jika pada solution hanya ada 1 project utama saja).



Gambar 47. Visual Studio 2019 - Reference Manager - Projects.

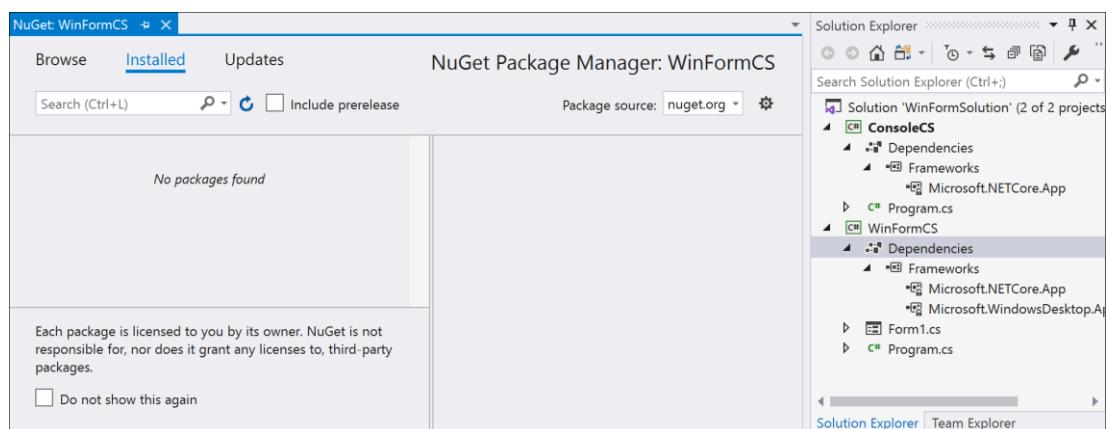
Selain itu library juga bisa ditambahkan dengan cara menambahkan file library *.DLL. Caranya dengan mengklik tombol Browse kemudian pilih file *.DLL yang ingin ditambahkan.

Setelah memilih salah satu assembly atau project maka hasilnya akan dapat dilihat pada daftar bagian Depedencies pada project di bagian Solution Explorer.

NuGet

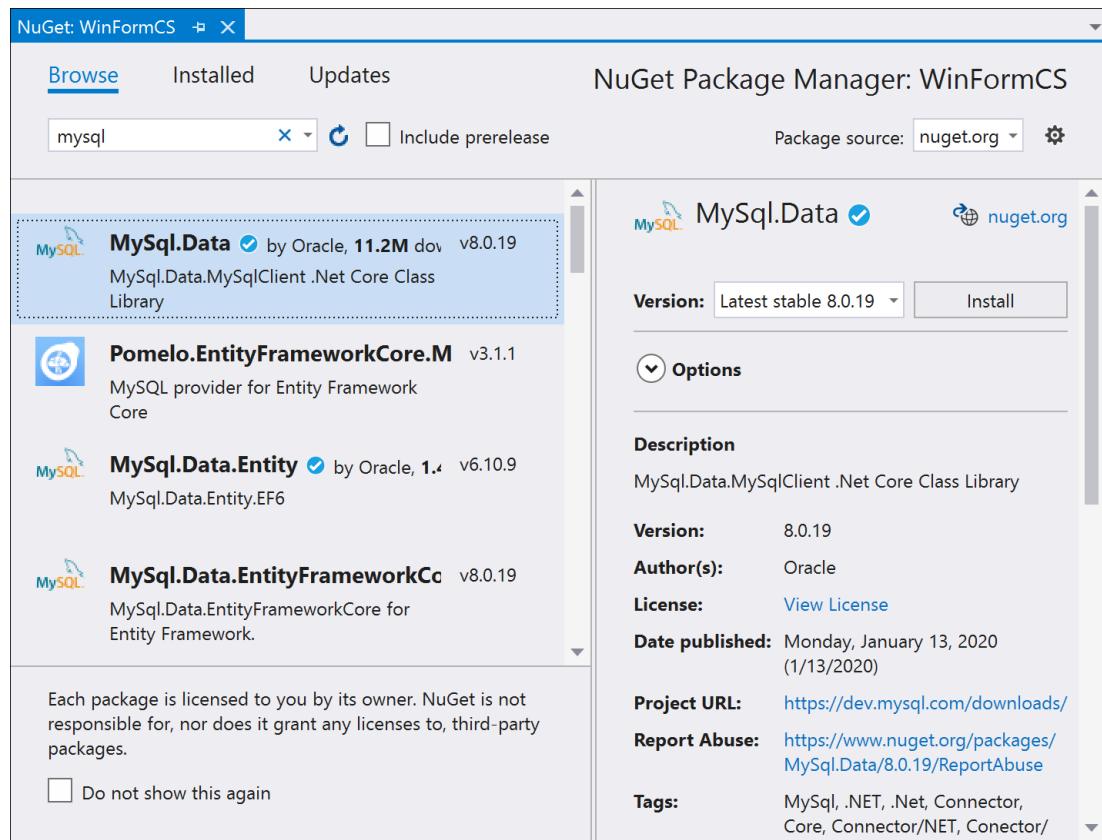
Untuk project yang dibangun dengan .NET Core Framework maka cara lain untuk menambahkan library dengan menggunakan NuGet. Sebagai contoh, pada gambar di bawah ini terdapat project dengan nama WinFormCS yang dibuat dengan menggunakan template project Windows Forms App (.NET Core).

Untuk menambahkan library pada project ini dapat menggunakan NuGet. Caranya klik kanan pada project kemudian pilih Manage NuGet Packages.



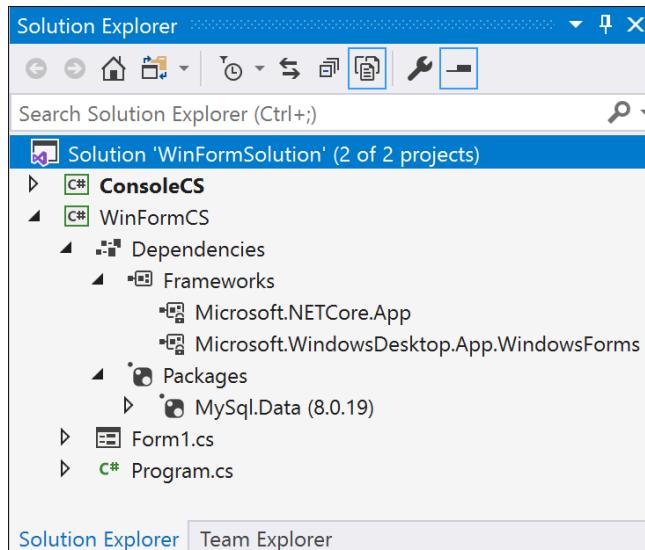
Gambar 48. Visual Studio 2019 - NuGet Package Manager - Installed.

Maka akan ditampilkan daftar library yang telah diinstall. Pada gambar di atas dapat dilihat belum ada library yang diinstall pada project WinFormCS. Jika ingin menginstall library maka klik tab Browse. Maka akan ditampilkan library yang tersedia secara online.



Gambar 49. Visual Studio 2019 - NuGet Package Manager - Browse.

Untuk itu harus dipastikan komputer pengguna terkoneksi dengan internet jika ingin menggunakan fitur ini. Jika ingin mencari library yang diinginkan, maka bisa mengetikkan kata kunci pada kolom Search. Misal library yang diinginkan telah ditemukan adalah MySql.Data, klik library tersebut maka di samping kanan daftar dapat dilihat informasi detail dari library tersebut. Jika ingin menginstallnya maka klik tombol Install. Kemudian ikutan instruksi selanjutnya.



Gambar 50. Visual Studio 2019 - NuGet pada Solution Explorer.

Hasilnya dapat dilihat pada Solution Explorer, pada Dependencies di project terdapat tambahan Packages > MySQL.Data (8.0.19).

4

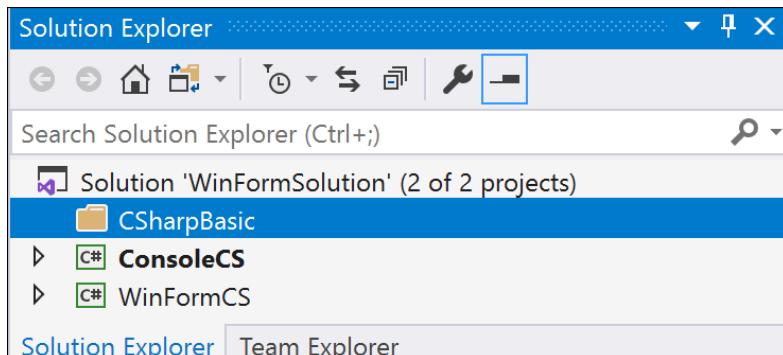
Pengantar Pemrograman C#

Salah satu bahasa pemrograman pada platform .NET adalah C# selain VB.NET, F# dan lain-lain. Saat buku ini ditulis bahasa pemrograman C# telah mencapai versi 8. Pada bab ini akan diberikan dasar-dasar pemrograman dengan bahasa pemrograman C# dengan penjelasan tentang konsep diikuti latihan yang dapat langsung dikerjakan oleh pembaca.

Pendahuluan

Persiapan

Untuk memulai bab ini ada beberapa hal yang harus dipersiapkan seperti membuat solution dan menyiapkan project. Solution yang digunakan adalah WinFormSolution yang sebelumnya telah dibuat pada sub bab 3 Visual Studio 2019 > Solution & Project > Solution pada halaman 31. Selanjutnya adalah membuat folder CSharpBasic pada solution ini dengan cara klik kanan pada solution kemudian pilih Add > New Solution Folder. Selanjutnya beri nama folder tersebut. Jika ingin memberi nama ulang folder tersebut, klik kanan pada folder kemudian pilih Rename pada context menu.



Gambar 51. Folder CSharpBasic pada Solution Explorer.

Selanjutnya project-project latihan pada bab ini akan disimpan pada folder tersebut.

Struktur Program Aplikasi Console/Desktop

C# dibaca sebagai see sharp adalah bahasa pemrograman yang sederhana, modern, object oriented, dan type safe. Bahasa ini merupakan berakar dari bahasa pemrograman C sehingga struktur program dan tata cara penulisannya mirip dengan bahasa pemrograman yang berakar dari C lainnya seperti Java, C++, Javascript dan lain-lain.

Seperti pada dua project yang telah dibuat sebelumnya, program utama disimpan pada file Program.cs. Untuk aplikasi console dapat dilihat file berikut ini.

```
Program.cs
using System;
```

```

namespace ConsoleCS
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}

```

Sedangkan untuk aplikasi desktop dengan WinForm dapat dilihat source code seperti berikut.

```

Program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WinFormCS
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

Dari kedua contoh file Program.cs di atas maka dapat dilihat beberapa bagian.

Bagian pertama adalah blok namespace, blok ini berisi kode untuk memanggil namespace lain yang digunakan pada program yang kita tulis. Untuk memanggil namespace lain digunakan keyword using diikuti nama namespace seperti contoh di bawah ini.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

```

Namespace berisi class yang didalamnya berisi fungsi-fungsi yang dapat digunakan oleh program. Namespace pada suatu file Program.cs dapat berbeda-beda tergantung kebutuhannya.

Blok kedua adalah program yang kita tulis yang mempunyai struktur sebagai berikut.

```

// nama namespace, biasanya sesuai dengan nama project
namespace NAMA_NAMESPACE
{ // mulai namespace
    // nama class
    class NAMA_CLASS
    { // mulai class
        // method utama
        static void Main(string[] args)
    }
}

```

```

    {
}

// method tambahan
static void Method1()
{
}

} // akhir class
} // akhir namespace

```

Nama namespace NAMA_NAMESPACE biasanya sesuai dengan nama project atau nama folder tempat file ini disimpan. Hal ini terjadi secara otomatis jika pembuatan project atau penambahan file class dengan menggunakan Visual Studio. Setelah penulisan namespace diikuti dengan tanda kurung kurawal buka ({) dan kurung kurawal tutup (}) yang menyatakan begin (mulai) dan end (akhir).

Selanjutnya didalamnya terdapat class. Di dalam sebuah namespace dapat berisi beberapa class. Namun pada umumnya sebuah file hanya berisi sebuah class saja. Pada aplikasi console atau desktop nama class utama yang harus ada adalah Program. Nama class umumnya disimpan pada file dengan nama yang sama dengan nama class, oleh sebab itu dapat dilihat class Program disimpan pada file Program.cs. Setelah nama class diikuti dengan tanda kurung kurawal buka ({) dan kurung kurawal tutup (}) yang menyatakan begin (mulai) dan end (akhir).

Di dalam sebuah class berisi method. Dalam sebuah class dapat berisi banyak method. Namun method yang harus ada untuk aplikasi console atau desktop adalah method berikut ini.

```

static void Main(string[] args)
{
}

```

Tapi jika dibuat aplikasi tipe lain seperti web atau mobile maka tidak digunakan method dengan nama seperti di atas sebagai method utama. Method Main ini akan dipanggil pertama kali jika program dijalankan. Sehingga baris-baris kode program akan ditulis didalamnya.

Berikut adalah contoh dimana ditulis beberapa method di dalam sebuah class.

Program.cs
<pre> using System; namespace ConsoleCS { class Program { static void Main(string[] args) { Console.WriteLine("Hello World!"); SayHello(); } static void SayHello() { Console.WriteLine("We say hello from Indonesia"); } } } </pre>

Sedangkan method SayHello adalah method yang dapat ditambahkan sesuai keperluan. Selanjutnya method ini dapat dipanggil dari method utama atau method tambahan lainnya. Cara membuat method tambahan akan dijelaskan pada sub bab selanjutnya.

Referensi

- <https://docs.microsoft.com/id-id/dotnet/csharp/tour-of-csharp/program-structure>
- <https://docs.microsoft.com/id-id/dotnet/csharp/tour-of-csharp/>

Aturan Penulisan Kode Program

Under development

Ada beberapa aturan penting yang harus diketahui untuk menulis kode program dengan benar.

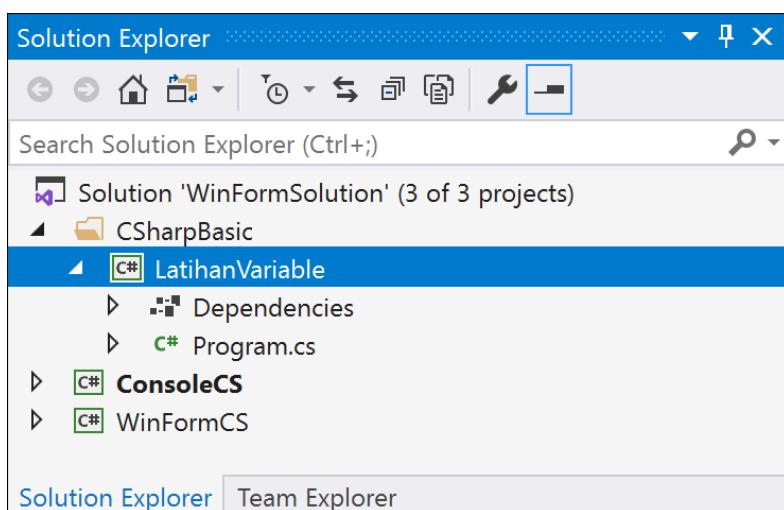
Tipe & Variable

Variable adalah wadah untuk menyimpan suatu nilai. Bahasa C# termasuk bahasa pemrograman yang menentukan tipe data pada suatu variable sebelum diisikan nilai kedinamanya. Ada dua tipe variable yang digunakan yaitu:

1. Value type yaitu tipe yang secara langsung berisi data.
2. Reference type yaitu menyimpan referensi ke data.

Sebagai latihan penggunaan variable dan tipe data maka terlebih dahulu dibuat project baru dengan nama LatihanVariable di dalam solution WinFormSolution > folder CSharpBasic. Caranya dengan klik kanan pada folder tersebut kemudian pilih Add > New Project. Kemudian pada window Add a new project pilih template Console App (.NET Core) dengan bahasa pemrograman C#. Kemudian klik tombol Next. Dan berikan nama project LatihanVariable. Kemudian klik tombol Create.

Hasilnya dapat dilihat seperti pada Gambar 52.



Gambar 52. Project LatihanVariable.

Selanjutnya adalah mengubah kode program Program.cs menjadi seperti berikut ini.

```
using System;

namespace LatihanVariable
{
    class Program
    {
        static void Main(string[] args)
        {
            char huruf = 'a';
            char hurufBesar = 'A';
            Console.WriteLine(hurufBesar);

            int bilangan1;
            bilangan1 = 13;
            Console.WriteLine(bilangan1);

            bool status;
            status = true;
            Console.WriteLine(status);
            status = false;
            Console.WriteLine(status);

            string message = "hello world";
            Console.WriteLine(message);
        }
    }
}
```

Dari contoh terlihat pola atau sintaks untuk melakukan deklarasi variable. Sintaks untuk membuat variable atau dikenal juga dengan istilah deklarasi variable adalah sebagai berikut.

```
TIPE_DATA namaVariable;
```

Penggunaan sintaks ini dapat dilihat pada potongan kode berikut.

```
int bilangan1;
...
bool status;
```

Jika saat deklarasi variable langsung diberikan nilai maka dapat menggunakan sintaks sebagai berikut.

```
TIPE_DATA namaVariable = NILAI_INISIAL;
```

Penggunaan sintaks ini dapat dilihat pada potongan kode berikut.

```
char huruf = 'a';
char hurufBesar = 'A';
...
string message = "hello world";
```

Setelah deklarasi variable dilakukan dengan salah satu cara di atas maka variable dapat digunakan. Variable dapat diisi dengan nilai yang sesuai dengan tipe datanya seperti contoh kode di bawah ini.

```
int bilangan1;
bilangan1 = 13;
```

Setelah variable diisi suatu nilai, variable dapat dipanggil atau nilainya diisi dengan nilai yang baru seperti contoh kode di bawah ini.

```
bool status;  
status = true;  
Console.WriteLine(status);  
status = false;  
Console.WriteLine(status);
```

Pada kode di atas variable yang dibuat adalah status dengan tipe data adalah boolean (bool). Kemudian variable status diisi dengan nilai. Untuk mengakses nilai tersebut dapat dilakukan dengan memanggil nama variable status. Salah satu caranya adalah dengan menggunakan method WriteLine yang berfungsi untuk menulis nilai ke layar.

Setelah tipe data di atas masih sangat banyak tipe data yang dapat disediakan oleh C#. secara lengkap dapat dilihat pada referensi di bawah ini.

Referensi

<https://docs.microsoft.com/id-id/dotnet/csharp/tour-of-csharp/types-and-variables>

Expression

Expression atau ekspresi adalah gabungan antara operan dan operator. Operan dapat berupa nilai seperti angka, karakter, string dan lain-lain. Operan juga dapat berupa variable. Sedangkan operator adalah tanda untuk melakukan operasi. Sebagai contoh tanda atau operator + untuk menjumlah. Ada beberapa jenis operator yang dapat digunakan yaitu:

- Operator aritmatika.
- Operator logika.
- Operator equality.
- Operator order.

Operator Aritmatika

Operator aritmatika dapat dibedakan menjadi dua jenis yaitu:

- Unary, operator ini dapat digunakan oleh satu operan. Contoh dari operator ini adalah ++ yang berfungsi untuk meningkatkan nilai operan sebanyak 1, -- berfungsi untuk menurunkan nilai operan sebanyak 1, + untuk tanda positif dan - untuk tanda negatif.
- Binary, operator ini digunakan oleh minamal dua operan. Contoh operator ini adalah * untuk perkalian, / untuk pembagian, % untuk mendapatkan sisa pembagian, + untuk penjumlahan dan - untuk pengurangan.

Sebelum menulis kode program contoh penggunaan operator aritmatika, buat project dengan nama LatihanOperatorAritmatika. Berikut adalah contoh kode program penggunaan beberapa operator yang telah dijelaskan di atas.

```
using System;  
  
namespace LatihanOperatorAritmatika  
{  
    class Program  
    {
```

```

static void Main(string[] args)
{
    // unary -- start
    int i = 3;
    Console.WriteLine(i);    // output: 3
    Console.WriteLine(i++); // output: 3
    Console.WriteLine(i);    // output: 4

    double a = 1.5;
    Console.WriteLine(a);    // output: 1.5
    Console.WriteLine(++a); // output: 2.5
    Console.WriteLine(a);    // output: 2.5

    Console.WriteLine(+4);    // output: 4
    Console.WriteLine(-4);    // output: -4
    Console.WriteLine(-(-4)); // output: 4
    // unary -- end

    //binary -- start
    Console.WriteLine(13 / 5.0);      // output: 2.6

    int x = 13;
    int y = 5;
    Console.WriteLine((double)x / y); // output: 2.6

    Console.WriteLine(5 % 4);    // output: 1
    Console.WriteLine(5 % -4);   // output: 1
    Console.WriteLine(-5 % 4);   // output: -1
    Console.WriteLine(-5 % -4);  // output: -1
    //binary -- end
}
}

```

Operator Logika

Seperti halnya operator aritmatika, operator ini juga terdiri atas dua tipe yaitu unary dan binary. Dan seperti diketahui bahwa nilai dari operan pada operasi logika hanyalah dua nilai yaitu true dan false.

Operator logika unary adalah tanda seru !. operator ini berfungsi sebagai negasi, artinya jika suatu operan bernilai true maka operator ini akan membuat nilai sebaliknya. Sedangkan untuk operator logika binary adalah:

- &, contoh penggunaanya adalah sebagai berikut a & b. Pernyataan itu benar jika nilai a dan b adalah true. Namun pemeriksaan tetap dilakukan kepada dua operan atau variable tersebut.
- &&, contoh penggunaanya sebagai berikut a && b. Seperti operator di atas, pernyataan benar jika operan a dan b adalah true. Namun cara memeriksa operan yang berbeda. Dengan menggunakan operator ini, jika a bernilai false maka b tidak perlu diperiksa lagi.
- | adalah operator OR yang artinya pernyataan benar jika kedua operan atau salah satu operan bernilai benar.

- || juga merupakan operator OR. Yang membedakannya dengan operator | adalah jika operan pertama sudah bernilai benar maka operan kedua tidak perlu diperiksa lagi.
- ^ adalah operator XOR atau exclusive OR yang artinya salah satu dari kedua operan adalah true dan satunya lagi adalah false dan sebaliknya.

Sebelum menulis kode program contoh penggunaan operator logika, buat project dengan nama LatihanOperatorLogika. Berikut adalah contoh kode penggunaan operator-operator ini.

```
using System;

namespace LatihanOpeartorLogika
{
    class Program
    {
        static void Main(string[] args)
        {
            // negasi -- start
            bool passed = false;
            Console.WriteLine(!passed); // output: True
            Console.WriteLine(!true); // output: False
            // negasi -- end

            // operator &
            bool SecondOperand()
            {
                Console.WriteLine("Second operand is evaluated.");
                return true;
            }

            bool a = false & SecondOperand();
            Console.WriteLine(a);
            // Output:
            // Second operand is evaluated.
            // False

            bool b = true & SecondOperand();
            Console.WriteLine(b);
            // Output:
            // Second operand is evaluated.
            // True

            // operator &&
            a = false && SecondOperand();
            Console.WriteLine(a);
            // Output:
            // False

            b = true && SecondOperand();
            Console.WriteLine(b);
            // Output:
            // Second operand is evaluated.
            // True

            // operator ^
            Console.WriteLine(true ^ true); // output: False
            Console.WriteLine(true ^ false); // output: True
            Console.WriteLine(false ^ true); // output: True
            Console.WriteLine(false ^ false); // output: False

            // operator |
        }
    }
}
```

```

        a = true | SecondOperand();
Console.WriteLine(a);
// Output:
// Second operand is evaluated.
// True

        b = false | SecondOperand();
Console.WriteLine(b);
// Output:
// Second operand is evaluated.
// True

        // operator ||
a = true || SecondOperand();
Console.WriteLine(a);
// Output:
// True

        b = false || SecondOperand();
Console.WriteLine(b);
// Output:
// Second operand is evaluated.
// True
    }
}
}

```

Operator Equality

Opeator ini bertujuan untuk membandingkan kesamaan dari dua operan. Untuk memeriksa apakah dua operan sama maka digunakan operator ==. Sedangkan untuk memeriksa apakah dua operan berbeda maka digunakan operator !=.

Sebelum menulis kode program contoh penggunaan operator equality, buat project dengan nama LatihanOperatorEquality. Contoh kode penggunaan operator-operator ini adalah sebagai berikut.

```

using System;

namespace LatihanOperatorEquality
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 1 + 2 + 3;
            int b = 6;
            Console.WriteLine(a == b); // output: True

            char c1 = 'a';
            char c2 = 'A';
            Console.WriteLine(c1 == c2); // output: False
            Console.WriteLine(c1 == char.ToLower(c2)); // output: True

            a = 1 + 1 + 2 + 3;
            b = 6;
            Console.WriteLine(a != b); // output: True

            string s1 = "Hello";
        }
    }
}

```

```

        string s2 = "Hello";
        Console.WriteLine(s1 != s2); // output: False

        object o1 = 1;
        object o2 = 1;
        Console.WriteLine(o1 != o2); // output: True
    }
}

```

Operator Order

Operator order dan operator equality yang telah dijelaskan di atas dapat disebut sebagai operator comparison yang bertujuan untuk membandingkan dua operator. Operator order bertujuan untuk mengetahui nilai operan mana yang lebih besar atau lebih kecil dari nilai operan lainnya.

Operator yang digunakan adalah:

- > untuk memeriksa apakah operan di sebelah kiri operator lebih besar daripada nilai operan di sebelah kanan operator.
- >= untuk memeriksa apakah operan di sebelah kiri operator lebih besar atau sama dengan dengan nilai operan di sebelah kanan operator.
- < untuk memeriksa apakah operan di sebelah kiri operator lebih kecil daripada nilai operan di sebelah kanan operator.
- <= untuk memeriksa apakah operan di sebelah kiri operator lebih kecil atau sama dengan nilai operan di sebelah kanan operator.

Sebelum menulis kode program contoh penggunaan operator order, buat project dengan nama LatihanOperatorOrder. Berikut adalah contoh kode penggunaan operator-operator order ini.

```

using System;

namespace LatihanOperatorOrder
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(7.0 < 5.1); // output: False
            Console.WriteLine(5.1 < 5.1); // output: False
            Console.WriteLine(0.0 < 5.1); // output: True

            Console.WriteLine(7.0 > 5.1); // output: True
            Console.WriteLine(5.1 > 5.1); // output: False
            Console.WriteLine(0.0 > 5.1); // output: False

            Console.WriteLine(7.0 <= 5.1); // output: False
            Console.WriteLine(5.1 <= 5.1); // output: True
            Console.WriteLine(0.0 <= 5.1); // output: True

            Console.WriteLine(7.0 >= 5.1); // output: True
            Console.WriteLine(5.1 >= 5.1); // output: True
            Console.WriteLine(0.0 >= 5.1); // output: False
        }
    }
}

```

Referensi

Statement

Pada sub bab-sub bab sebelumnya telah digunakan penggunaan statement. Penjelasan singkat statement adalah baris yang berada di antara tanda { dan }. Aturan umum statement adalah setiap statement diakhiri dengan tanda semicolon ;.

```
statement1;  
statement2;
```

Statement dapat dibagi menjadi beberapa jenis sesuai fungsinya, yaitu:

- Declaration statement untuk mendeklarasikan variable dan konstanta lokal .
- Expression statement untuk membuat obyek dan operasi-operasi yang menggunakan operator dan operan.
- Selection statement digunakan pada percabangan dengan kata kunci if dan switch.
- Iteration statement digunakan pada pengulangan dengan menggunakan kata kunci for, while dan foreach.
- Jump statement digunakan untuk kontrol yang digunakan jika menggunakan kata kunci break, continue, goto, throw, return dan yield.

Sebelum menulis kode program contoh penggunaan statement-statement, buat project dengan nama LatihanStatement. Berikut adalah sebagian kode yang menggunakan statement-statement di atas.

```
using System;  
  
namespace LatihanStatement  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            Declarations();  
            ConstantDeclarations();  
            Expressions();  
            IfStatement();  
        }  
  
        static void Declarations()  
        {  
            Console.WriteLine("Declarations");  
            int a;  
            int b = 2, c = 3;  
            a = 1;  
            Console.WriteLine(a + b + c);  
        }  
  
        static void ConstantDeclarations()  
        {  
            Console.WriteLine("ConstantDeclarations");  
            const float pi = 3.1415927f;  
            const int r = 25;  
            Console.WriteLine(pi * r * r);  
        }  
    }  
}
```

```

    }

    static void Expressions()
    {
        Console.WriteLine("Expressions");
        int i;
        i = 123;           // Expression statement
        Console.WriteLine(i); // Expression statement
        i++;              // Expression statement
        Console.WriteLine(i); // Expression statement
    }

    static void IfStatement()
    {
        int number = 1;
        Console.WriteLine("IfStatement");
        if (number == 1)
        {
            Console.WriteLine("Number is 1");
        }
        else
        {
            Console.WriteLine("Number is not 1");
        }
    }

    static void WhileStatement()
    {
        int i = 0;
        int number = 13;
        while (i < number)
        {
            Console.WriteLine(i);
            i++;
        }
    }
}

```

Referensi

- <https://docs.microsoft.com/id-id/dotnet/csharp/tour-of-csharp/statements>

Array

Pada sub bab tipe dan variable telah diberikan cara untuk menyimpan sebuah nilai. Jika ingin menampung atau menyimpan sekumpulan variable atau nilai maka diperlukan wadah. Pada bahasa pemrograman umumnya wadah tersebut dikenal dengan istilah array. Array hanya dapat menampung nilai dengan tipe data yang sama.

Berikut ini adalah sintaks untuk melakukan deklarasi array.

```
TIPE_DATA[] namaVariable = new TIPE_DATA[JUMLAH_DATA];
```

Pada sintaks di atas dapat dilihat ada JUMLAH_DATA yang menyatakan pemesanan jumlah tempat untuk nilai. Sebagai contoh penggunaannya seperti pada potongan kode di bawah ini..

```
int[] a1 = new int[10];
```

Artinya adalah variable a1 merupakan array yang akan menampung nilai yang tipe datanya adalah integer. Jumlah data yang dapat ditampung adalah maksimal 10 data atau nilai.

Cara deklarasi array yang lain adalah seperti contoh berikut ini. Pada kedua contoh ini tidak ada ada penentuan jumlah data. Namun variable a1 ataupun a2 telah diberikan nilai sebanyak 3 nilai.

```
int[] a1 = new int[] { 1, 2, 3 };
int[] a2 = { 1, 2, 3 };
```

Ini artinya telah dipesan 3 tempat. Sehingga variable a1 dan a2 tidak bisa diberikan nilai lebih dari 4.

Sebelum menulis kode program contoh penggunaan array, buat project dengan nama LatihanArray. Berikut adalah contoh pemanfaatan array.

```
using System;

namespace LatihanArray
{
    class Program
    {
        static void Main(string[] args)
        {
            // deklarasi
            int[] a = new int[3];

            // memberikan nilai
            a[0] = 1;
            a[1] = 2;
            a[2] = 3;

            // akses nilai array
            Console.WriteLine(a[0]);

            a[0] = 13;
            Console.WriteLine(a[0]);
        }
    }
}
```

Selain dengan cara di atas, seluruh nilai array juga bisa diakses secara otomatis dengan cara pengulangan yang akan dibahas pada sub bab selanjutnya.

Pada contoh-contoh di atas hanya dibahas array satu dimensi, selain itu dimensi array dapat berjumlah lebih dari satu yaitu dua atau tiga. Untuk membuat array dengan dimensi dua atau tiga dapat dilihat contoh berikut ini.

```
int[,] a2 = new int[10, 5];
int[, ,] a3 = new int[10, 5, 2];
```

Referensi

- <https://docs.microsoft.com/id-id/dotnet/csharp/tour-of-csharp/arrays>

Percabangan

Pada sub bab statement telah disebutkan tentang selection statement yang didalamnya menyebutkan tentang percabangan dengan menggunakan kata kunci if dan switch. Statement ini memungkinkan kita untuk melakukan percabangan sesuai dengan kondisi tertentu.

Statement if

Statement if bertujuan untuk menjalankan suatu statement jika suatu kondisi terpenuhi. Sintaks percabangan dengan if adalah sebagai berikut.

```
if (condition) {  
    statement1;  
    statement2;  
    ...  
}
```

Jika diartikan ke dalam bahasa yang digunakan manusia dalam berkomunikasi maka sintaks di atas dapat ditulis sebagai berikut.

```
jika (kondisi benar atau terpenuhi) maka kerjakan:  
    - statement1  
    - statement2  
    - dan seterusnya
```

Selain itu juga ada sintaks yang lebih lengkap yaitu sebagai berikut:

```
if (condition1) {  
    statement1;  
    ...  
} else if (condition2) {  
    statement2;  
    ...  
} else {  
    statement3;  
    ...  
}
```

Dengan sintaks di atas dapat terlihat seperti ada 3 percabangan. Alur pemeriksaan kondisi adalah mengalir turun dari atas ke bawah. Pertama adalah memeriksa condition1, jika bernilai benar atau syarat terpenuhi maka statement1 akan dijalankan. Jika condition1 tidak terpenuhi maka akan diperiksa condition2. Bila terpenuhi maka statement2 akan dijalankan. Namun jika condition1 dan condition2 tidak terpenuhi maka statement3 akan dijalankan.

Sebelum menulis kode program contoh penggunaan percabangan if, buat project dengan nama LatihanPercabanganIf. Berikut ini adalah contoh dari penggunaan percabangan dengan if.

```
using System;  
  
namespace LatihanPercabanganIf  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int a, b;  
            a = 13;  
            b = 23;
```

```

        if (a > b)
        {
            Console.WriteLine("a > b");
        } else if (a < b)
        {
            Console.WriteLine("a < b");
        } else
        {
            Console.WriteLine("a = b");
        }
    }
}

```

Statement switch

Statement ini mencocokkan sebuah nilai dengan nilai dari daftar pilihan yang ada. Untuk mengerti statement ini maka dapat dilihat pada contoh penggunaan kata kunci switch di bawah ini. Sebelum menulis kode program contoh penggunaan percabangan switch, buat project dengan nama LatihanPercabanganSwitch.

```

using System;

namespace LatihanPercabanganSwitch
{
    class Program
    {
        static void Main(string[] args)
        {
            int caseSwitch = 1;

            switch (caseSwitch)
            {
                case 1:
                    Console.WriteLine("Case 1");
                    break;
                case 2:
                    Console.WriteLine("Case 2");
                    break;
                default:
                    Console.WriteLine("Default case");
                    break;
            }
        }
    }
}

```

Pada contoh di atas dimiliki variable caseSwitch, dengan menggunakan kata kunci switch maka nilai variable ini akan dibandingkan nilai nilai-nilai yang berada setelah kata kunci case. Karena nilai variable caseSwitch adalah 1 maka cocok dengan case pertama sehingga statement didalamnya akan dijalankan. statement di dalam blok ini dapat terdiri atas satu atau lebih statement. Akhir dari blok ini adalah break yang bertujuan agar jalannya program keluar dari blok ini switch. Artinya case kedua dan default tidak akan diperiksa lagi.

Pengulangan

Pada kehidupan nyata, kita sering melakukan hal yang sama secara berulang-ulang. Sebagai contoh menulis "Hello World" sebanyak 10 kali maka ditulis kode sebagai berikut.

```

using System;

namespace LatihanPengulanganFor
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.WriteLine("Hello World!");
        }
    }
}

```

Bayangkan jika ingin ditulis sebanyak 100 atau 1000 kali, artinya akan ditulis 100 atau 1000 baris seperti contoh di atas. Untuk itu diperlukan kota yang singkat namun melakukan hasil yang seperti contoh di atas.

Pada sub bab ini akan dijelaskan pengulangan dengan menggunakan kata kunci for dan while saja. C# masih memiliki kata kunci yang lain yang dapat digunakan untuk melakukan pengulangan.

Statement for

Statement for memiliki sintaks sebagai berikut.

```

for(inisialisasi; kondisi; pencacah)
{
    statement1;
    . . .
}

```

Statement inisialisasi bertujuan untuk menentukan nilai awal dari suatu variable. Kemudian statement kondisi bertujuan untuk memeriksa nilai variable jika masih pada kondisi benar maka statement-statement di dalam tanda { dan } dikerjakan. Setelah statement-statement selesai dikerjakan, maka statement pencacah akan dikerjakan.

Sebelum menulis kode program contoh penggunaan pengulangan for, buat project dengan nama LatihanPengulanganFor. Berikut adalah contoh penggunaan statement for.

```

using System;

namespace LatihanPengulanganFor
{
    class Program
    {
        static void Main(string[] args)
        {
            int max_count = 100;
            for(int i=1; i < max_count; i++)
            {
                Console.WriteLine("Hello World");
            }
        }
    }
}

```

```
    }
}
```

Pada contoh di atas dapat dilihat pada statement inisialisasi dilakukan deklarasi variable i dan memberikan nilai awal yaitu 1. Statement inisialisasi ini hanya dijalankan sekali saja. Pada statement kondisi variable ini diperiksa, jika nilai variable i lebih kecil dari nilai variable max_count maka artinya kondisi masih bernilai benar. Statement kondisi ini dijalankan setiap kali sampai proses pengulangan berhenti. Kemudian statement di dalam tanda { dan } dijalankan. setelah itu statement pencacah dijalankan, artinya nilai i akan ditambah 1 menjadi 2. Selanjutnya statement kondisi dijalankan lagi, dengan nilai i adalah 2 maka nilai ini masih lebih kecil dari 100. Artinya kondisi ini masih benar dan statement diantara tanda { dan } kembali dijalankan. Perulangan akan berhenti ketika nilai i = 100.

Statement while

Sintaks pengulangan dengan statement while adalah sebagai berikut.

```
while(kondisi)
{
    statement1;
    ...
}
```

Statement-statement diantara tanda { dan } akan diulang selama kondisi benar.

Sebelum menulis kode program contoh penggunaan pengulangan while, buat project dengan nama LatihanPengulanganWhile. Berikut adalah contoh kode penggunaan sintaks di atas.

```
using System;

namespace LatihanPengulanganWhile
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, max_count;
            i = 1;
            max_count = 100;
            while(i < max_count)
            {
                Console.WriteLine("Hello World!");
                i++;
            }
        }
    }
}
```

Contoh di atas memiliki keluaran yang sama dengan pengulangan dengan menggunakan statement for. Pada kode di atas tetap dilakukan inisialisasi variable, pada contoh di atas adalah variable i yang diberi nilai inisial 1. Kemudian pengulangan akan dilakukan selama kondisi terpenuhi. Di dalam statement yang berada di antara tanda { dan } terdapat statement untuk mengubah nilai i.

Method

Method dapat diartikan sebagai program kecil yang berada di dalam program utama. Dimana program ini dapat dipanggil lebih dari satu kali sesuai dengan kebutuhan.

Method dapat dibedakan menjadi dua jenis, yaitu:

- Method yang tidak mengembalikan nilai.
- Method yang mengembalikan nilai.

Pada contoh-contoh project yang telah disebutkan di atas, telah diperkenalkan method dengan nama Main. Method seperti itu adalah jenis method yang tidak mengembalikan nilai. Sintaks dari method jenis ini adalah sebagai berikut.

```
void NAMA_METHOD(TIPE_DATA PARAMETER)
{
}
```

Untuk membuat kode method ini maka terlebih dahulu dibuat project Console App (.NET Core) di dalam folder CSharpBasic dengan nama LatihanProsedur.

```
using System;

namespace LatihanProsedur
{
    class Program
    {
        static void Main(string[] args)
        {
            TulisHello();
            Tulis("Hello World");
        }

        static void TulisHello()
        {
            Console.WriteLine("Hello World!");
        }

        static void Tulis(string message)
        {
            Console.WriteLine(message);
        }
    }
}
```

Pada contoh di atas dapat dilihat dua contoh method di atas. Sedangkan untuk method jenis yang mengembalikan nilai memiliki sintaks sebagai berikut.

```
TIPE_DATA NAMA_METHOD()
{
    return NILAI_ATAU_VARIABLE_YANG_BERISI_NILAI;
}
```

Dan berikut adalah contoh method ini yang ditulis pada project LatihanFungsi.

```
using System;

namespace LatihanFungsi
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

```

    }

    static string Hello()
    {
        string message = "Hello World";
        return message;
    }

    static int Tambah(int bil1, int bil2)
    {
        int hasil = bil1 + bil2;
        return hasil;
    }
}

```

Pemrograman Berbasis Obyek

C# adalah bahasa pemrograman berorientasi obyek atau object oriented programming (OOP). Maka oleh sebab itu itu pembahasan tentang class dan object penting. Selain itu pembahasan lainnya adalah method. Penjelasan tentang class, object dan method dimaksudkan agar pembaca mengerti cara membuat atau menggunakan class-class yang telah disediakan oleh framework Windows Form.

Untuk latihan pemrograman berbasis obyek ini akan dibuat project-project yang akan disimpan pada folder CSharpOOP. Oleh sebab itu buat folder ini didalam solution WinFormSolution.

Class & Object

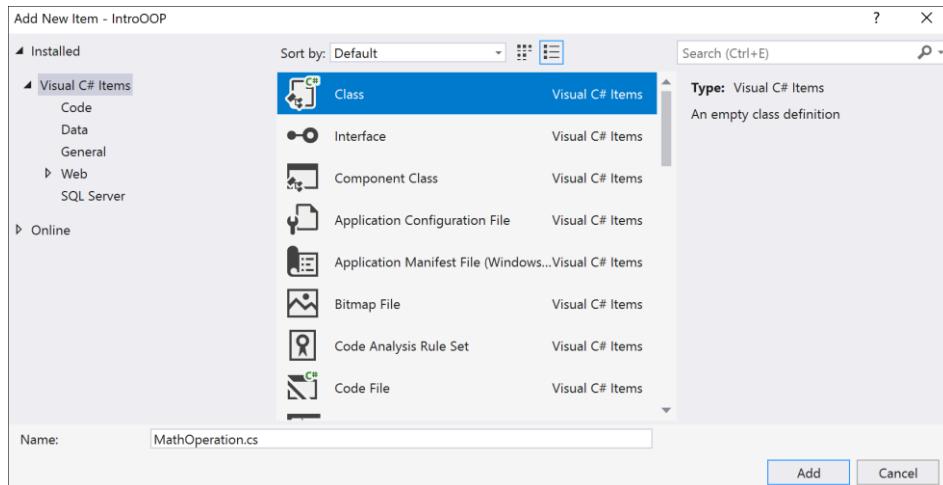
Secara sederhana class dapat diartikan sebagai template untuk membuat object. Untuk melihat secara nyata yang dimaksudnya maka terlebih dahulu dibuat project Console App (.NET Core) pada folder CSharpOOP. Nama project adalah IntroOOP. Pada proejct ini terdapat class yang disimpan pada file Program.cs.

Program.cs
<pre> using System; namespace IntroOOP { class Program { static void Main(string[] args) { Console.WriteLine("Hello World!"); } } } </pre>

Dari contoh di atas dapat dilihat sintaks untuk membuat class adalah sebagai berikut.

<pre> class NAMA_CLASS { } </pre>

Sebuah class yang dibuat umumnya disimpan ke dalam sebuah file. Dalam sebuah project dapat dibuat class baru. Sebagai contoh untuk menambahkan class pada project IntroOOP dapat dilakukan dengan cara klik kanan pada project kemudian pilih Add > Class. Kemudian akan ditampilkan window Add New Item seperti pada Gambar 53. Ketik nama file pada kolom Name dengan nama MathOperation.cs. Kemudian klik tombol Add.



Gambar 53. Membuat class MathOperation.cs.

Hasilnya dapat dilihat pada kode di bawah ini.

```
MathOperation.cs
using System;
using System.Collections.Generic;
using System.Text;

namespace IntroOOP
{
    class MathOperation
    {
    }
}
```

Selain itu class juga bisa dibuat di dalam file class yang sudah ada. Sebagai contoh ditambahkan class StringOperation di dalam file Program.cs seperti contoh berikut ini.

```
Program.cs
using System;

namespace IntroOOP
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }

        class StringOperation
        {
        }
    }
}
```

Pada kedua contoh class di atas hanyalah class kosong. Umumnya class memiliki cara mendefinisikan data dan method untuk memanipulasi dan mengakses data.

Untuk menambahkan method pada class dapat dilihat kembali pada sub bab tentang method. Pada kode berikut akan diperlihatkan penambahan method pada class MathOperation.

```
MathOperation.cs
using System;
using System.Collections.Generic;
using System.Text;

namespace IntroOOP
{
    public class MathOperation
    {
        public int Tambah(int bil1, int bil2)
        {
            int hasil = bil1 + bil2;
            return hasil;
        }

        public int Kurang(int bil1, int bil2)
        {
            int hasil = bil1 - bil2;
            return hasil;
        }

        public int Kali(int bil1, int bil2)
        {
            int hasil = bil1 * bil2;
            return hasil;
        }

        public void TulisHasil(object nilai)
        {
            Console.WriteLine(nilai);
        }
    }
}
```

Selanjutnya akan diperlihatkan cara untuk membuat obyek dari class MathOperation. Berikut adalah sintaks yang digunakan.

```
NAMA_CLASS NAMA_OBJECT = new NAMA_CLASS();
```

Maka berikut adalah contoh program lengkapnya.

```
Program.cs
using System;

namespace IntroOOP
{
    class Program
    {
        static void Main(string[] args)
        {
            MathOperation mo = new MathOperation();
            int hasilTambah = mo.Tambah(1, 3);
            int hasilKurang = mo.Kurang(5, 2);

            StringOperation.Tulis("Hasil 1 + 3 = " + hasilTambah);
            StringOperation.Tulis("Hasil 5 + 2 = " + hasilKurang);
        }
    }
}
```

```

public static class StringOperation
{
    public static void Tulis(object value)
    {
        Console.WriteLine(value);
    }
}

```

Pada contoh di atas dapat dilihat contoh membuat object dari class MathOperation. Dan bagaimana cara mengakses method-method yang dimiliki oleh class MathOperation.

```

MathOperation mo = new MathOperation();
int hasilTambah = mo.Tambah(1, 3);
int hasilKurang = mo.Kurang(5, 2);

```

Pada Program.cs juga dapat dilihat class static StringOperation. Untuk class statis dan method static tidak perlu melakukan pembuatan object seperti contoh MathOperation. Namun dapat langsung digunakan dengan memanggil nama class diikuti dengan nama methodnya seperti potongan kode berikut ini.

```

StringOperation.Tulis("Hasil 1 + 3 = " + hasilTambah);
StringOperation.Tulis("Hasil 5 + 2 = " + hasilKurang);

```

Property

Selain memiliki method class juga dapat memiliki property. Jika method berfungsi untuk memproses data. Maka property berfungsi untuk menyimpan data. Berikut ini adalah kode class MathOperation yang telah dimodifikasi.

```

MathOperation.cs
using System;
using System.Collections.Generic;
using System.Text;

namespace IntroOOP
{
    public class MathOperation
    {
        public int bilangan1 { set; get; }
        public int bilangan2 { set; get; }
        public int Tambah(int bil1, int bil2)
        {
            int hasil = bil1 + bil2;
            return hasil;
        }

        public int Tambah()
        {
            int hasil = bilangan1 + bilangan2;
            return hasil;
        }

        public int Kurang(int bil1, int bil2)
        {
            int hasil = bil1 - bil2;
            return hasil;
        }

        public int Kali(int bil1, int bil2)
        {
            int hasil = bil1 * bil2;
        }
    }
}

```

```

        return hasil;
    }

    public void TulisHasil(object nilai)
    {
        Console.WriteLine(nilai);
    }
}

```

Pada kode di atas dapat dilihat property bilangan1 dan bilangan2. Dan dapat dilihat juga tambahan method Tambah yang tidak memiliki parameter input. Dan pada kode di bawah ini dapat dilihat bagaimana mengisi data pada property-property tersebut.

```

Program.cs
using System;

namespace IntroOOP
{
    class Program
    {
        static void Main(string[] args)
        {
            MathOperation mo = new MathOperation();
            int hasilTambah = mo.Tambah(1, 3);
            int hasilKurang = mo.Kurang(5, 2);

            StringOperation.Tulis("Hasil 1 + 3 = " + hasilTambah);
            StringOperation.Tulis("Hasil 5 + 2 = " + hasilKurang);

            mo.bilangan1 = 4;
            mo.bilangan2 = 4;
            int hasilTambah2 = mo.Tambah();
            StringOperation.Tulis("Hasil 4 + 4 = " + hasilTambah2);
        }
    }

    public static class StringOperation
    {
        public static void Tulis(object value)
        {
            Console.WriteLine(value);
        }
    }
}

```

Access Modifier

Pada contoh di atas dapat dilihat penggunaan kata kunci public pada class. Kata kunci tersebut dikenal dengan istilah access modifier yang berfungsi untuk mengatur tata cara mengakses property dan method di dalam class. Berikut adalah keterangan dari masing-masing access modifier yang ada:

- Public, membuat method dan property dapat dilihat dan digunakan pada kode lain atau class lain.
- Protected, membuat method dan property hanya dapat dilihat dan digunakan oleh class yang turunannya.

- Private, membuat method dan property hanya dapat dilihat dan digunakan oleh class itu sendiri.

Dan masih banyak lagi access modifier yang dimiliki oleh bahasa pemrograman ini. Selain itu static juga dapat disebut sebagai modifier, yang berfungsi untuk membuat method atau property dapat diakses tanpa perlu membuat object dari class tersebut.

5

Pemrograman Visual

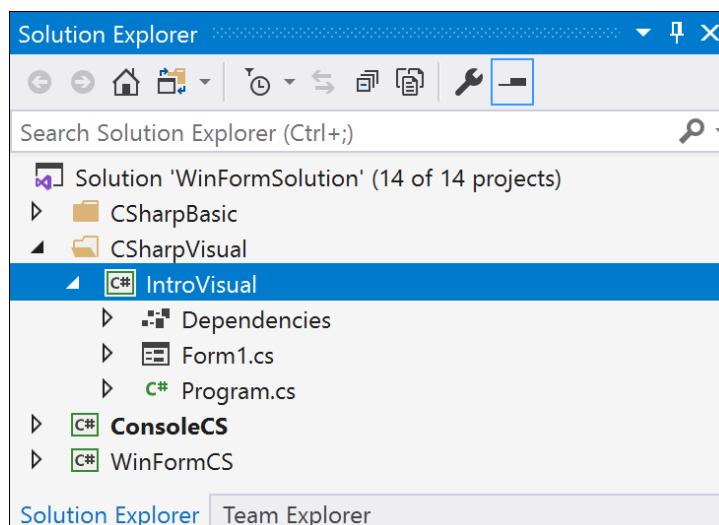
Pada bab ini dijelaskan dasar-dasar pemrograman visual dengan framework Windows Forms.

Pengenalan Lingkungan & Control

Untuk berkenalan dengan lingkungan dan control-control yang disediakan framework Windows Form untuk membuat aplikasi desktop maka terlebih dahulu disiapkan project. Project ini ditambahkan ke dalam WinFormSolution yang sebelumnya telah diterangkan cara pembuatannya. Kemudian tambahkan folder CSharpVisual pada solution ini dengan cara klik kanan WinFormSolution pada Solution Explorer kemudian pilih Add > New Solution Folder. Kemudian klik kanan pada folder CSharpVisual kemudian pilih Add > New Project.

Kemudian pada window Add a new project pilih Windows Form App (.NET Core). Kemudian pada window Configure your new project berikan nama IntroVisual pada kolom Project name. Dan diakhiri dengan mengklik tombol Create.

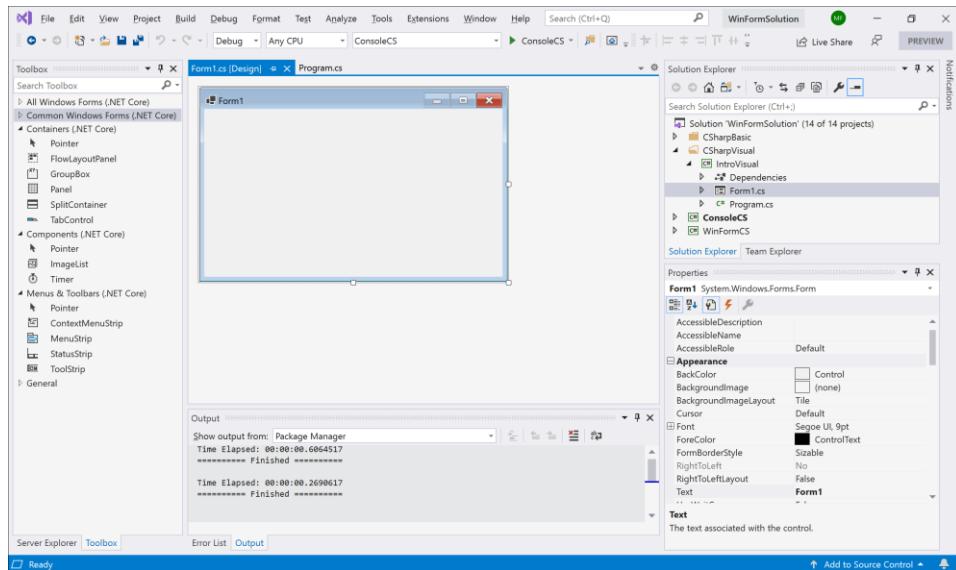
Hasilnya dapat dilihat pada gambar berikut.



Gambar 54. Project IntroVisual.

Pada project Console file Program.cs adalah file utama untuk menulis kode program. Pada project Windows Form App juga terdapat file ini yang berfungsi untuk menjalankan atau memanggil aplikasi desktop yang disimpan pada file Form1.cs. File ini akan berisi kode untuk membangkitkan control-control yang ditambahkan dan juga event handler dan lain-lain.

Pada Visual Studio, jika Form1.cs diklik double maka akan ditampilkan desain secara visual seperti pada gambar berikut ini.

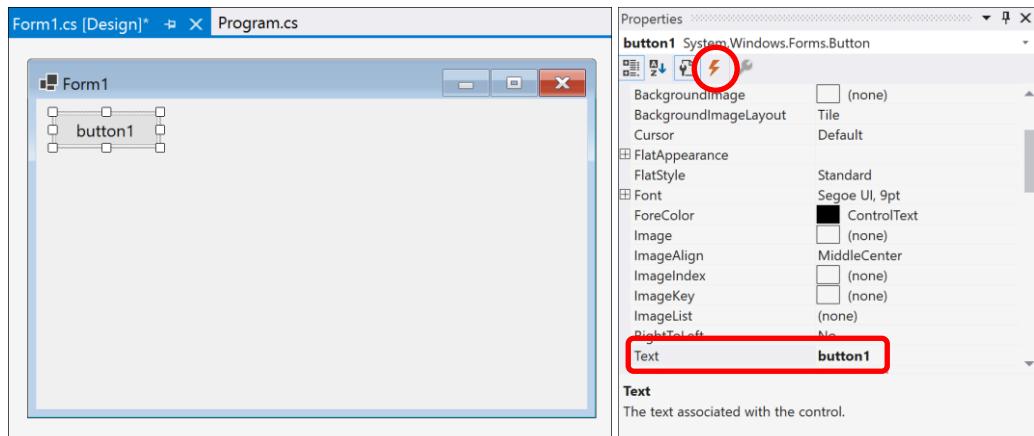


Gambar 55. Visual Designer.

Area design berada di tengah berupa window aplikasi desktop pada umumnya. Pada sisi sebelah kiri adalah area Toolbox yang berisi control-control. Sedangkan pada sisi kanan bawah terdapat area Properties. Setiap sebuah control dipilih pada area design maka dapat dilihat property-property yang dimiliki oleh control tersebut pada area Properties.

Control & Properties

Untuk menambahkan control pada window utama maka dapat dilakukan dengan menarik control yang ada pada Toolbox kemudian diletakkan pada window Form1. Sebagai contoh diletakkan control Button pada window Form1. Posisi atau letak control Button ini dapat diatur posisinya dengan cara menarik dan lepas (drag and drop) pada lokasi yang diinginkan.



Gambar 56. Control & property.

Pada gambar dapat dilihat ketika control Button dipilih maka area Properties akan berubah dan menampilkan property-property milik control ini. Dengan mengakses area Properties maka teks button1 dapat diubah yaitu dengan cara mengubah nilai property Text seperti yang terlihat pada kotak pada gambar. Pada Properties juga dapat digunakan untuk mengetahui atau mengakses event pada control yaitu dengan cara mengklik tombol petir seperti pada lingkaran di atas.

Property dan event yang dimiliki oleh masing-masing control akan dibahas secara lebih dalam pada sub bab selanjutnya.

Struktur Form

Tampilan visual Form yang ditampilkan pada area design terdiri atas beberapa file. Pada project di atas adalah Form1 artinya terdapat beberapa file yaitu:

- Form1.Designer.cs, file ini berisi kode-kode yang berisi instansiasi object control kemudian menyimpan property-propertynya. Sebagai contoh jika ditambahkan control Button pada window form pada area design maka file ini akan mengalami perubahan dengan membuat object control tersebut. Kemudian jika posisi control button juga disimpan pada file ini. Isi file ini dapat dilihat pada source code Form1.Designer.cs di bawah ini. Jika banyak control ditambahkan pada form maka secara otomatis isi file ini akan semakin banyak.
- Form1.cs, file ini dapat dibilang sebagai file utama form yang berfungsi untuk menyimpan logika dari event-event yang diberikan pada form. Sebagai contoh ketika tombol diklik, logika apa yang akan dikerjakan akan disimpan dalam bentuk sebuah method. Contohnya dapat dilihat pada file Form1.cs di bawah ini.

```
Form1.Designer.cs
namespace IntroVisual
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #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()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            // 
            // button1
            // 
```

```

//  

this.button1.Location = new System.Drawing.Point(27, 22);  

this.button1.Name = "button1";  

this.button1.Size = new System.Drawing.Size(150, 46);  

this.button1.TabIndex = 0;  

this.button1.Text = "button1";  

this.button1.UseVisualStyleBackColor = true;  

this.button1.Click += new  

System.EventHandler(this.button1_Click);  

//  

// Form1  

//  

this.AutoScaleDimensions = new System.Drawing.SizeF(13F, 32F);  

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  

this.ClientSize = new System.Drawing.Size(800, 450);  

this.Controls.Add(this.button1);  

this.Name = "Form1";  

this.Text = "Form1";  

this.ResumeLayout(false);  

}  

#endregion  

private System.Windows.Forms.Button button1;  

}
}

```

Form1.cs
<pre> using System; using System.Collections.Generic; using System.ComponentModel; using System.Data; using System.Drawing; using System.Linq; using System.Text; using System.Threading.Tasks; using System.Windows.Forms; namespace IntroVisual { public partial class Form1 : Form { public Form1() { InitializeComponent(); } private void button1_Click(object sender, EventArgs e) { MessageBox.Show("Hello World"); } } } </pre>

Program.cs

File ini dapat dianggap file utama program yang akan dijalankan pertama kali. Dalam sebuah aplikasi desktop yang dibangun dengan framework Windows Form didalamnya dapat

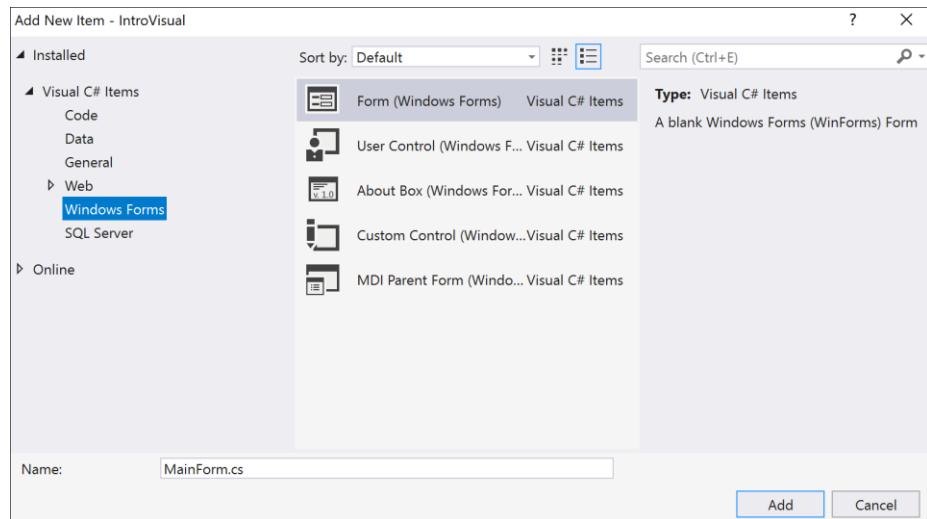
terdapat form-form yang telah disebutkan di atas. Fungsi Program.cs adalah memanggil form mana yang akan dijalankan pertama kali. Selanjutnya dari form tersebut dapat diaktifkan form-form lainnya.

Berikut adalah isi file Program.cs.

```
Program.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

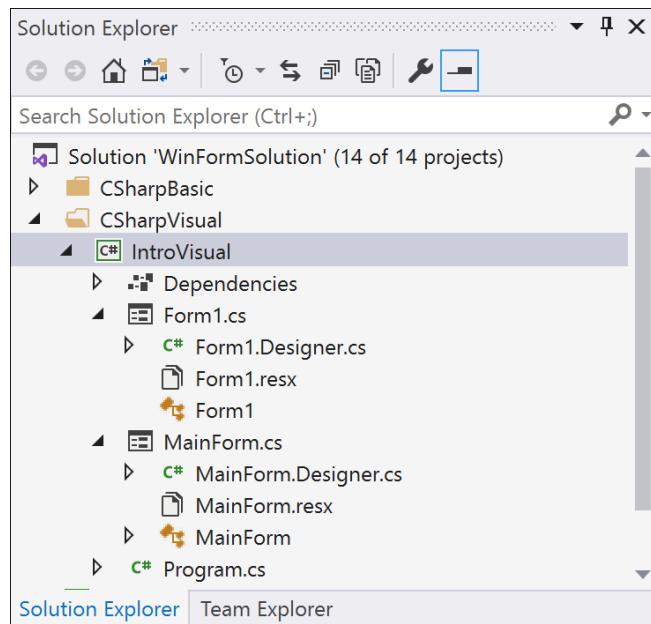
namespace IntroVisual
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.SetHighDpiMode(HighDpiMode.SystemAware);
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Pada baris yang dicetak tebal dapat dilihat form yang pertama dipanggil adalah Form1. Misal dibuat form baru pada project IntroVisual dengan cara klik kanan pada project kemudian pilih Add > New Item kemudian pada window Add New Item pilih Windows Forms. Dan kemudian pilih Form (Windows Forms). Pada kolom Name tuliskan nama form misalnya adalah MainForm.cs seperti pada gambar di bawah ini. Kemudian klik tombol Add.



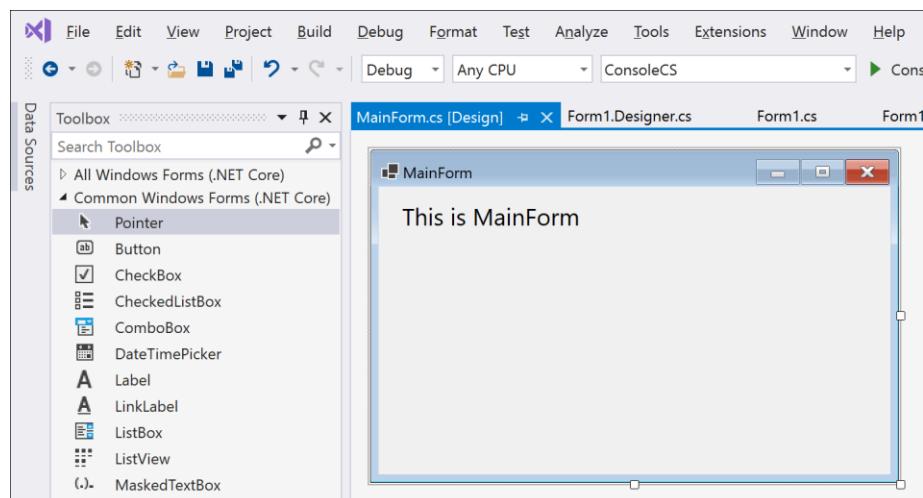
Gambar 57. Membuat MainForm.

Hasilnya dapat dilihat pada gambar di bawah ini.



Gambar 58. MainForm pada Solution Explorer.

Pada MainForm tambahkan Label kemudian atur property Text dan Font agar tulisannya menjadi seperti gambar berikut.



Gambar 59. Text pada MainForm.

Selanjutnya modifikasi file Program.cs dengan mengganti baris berikut:

```
Application.Run(new Form1());
```

Menjadi seperti berikut ini.

```
Application.Run(new MainForm());
```

Untuk melihat apakah benar MainForm yang akan dijalankan pertama kali, jalankan project dengan mengklik kanan project kemudian pilih Debug > Start New Instance.

Toolbox

Saat ini pada area Toolbox terdapat beberapa kelompok yaitu:

- Common Windows Forms.
- Container
- Component
- Menu & Toolbars

Pada sub bab ini akan dibahas control-control pada masing-masing kelompok tersebut. Pembahasan control-control akan diberikan dalam bentuk contoh kasus.

Kalkulator Sederhana

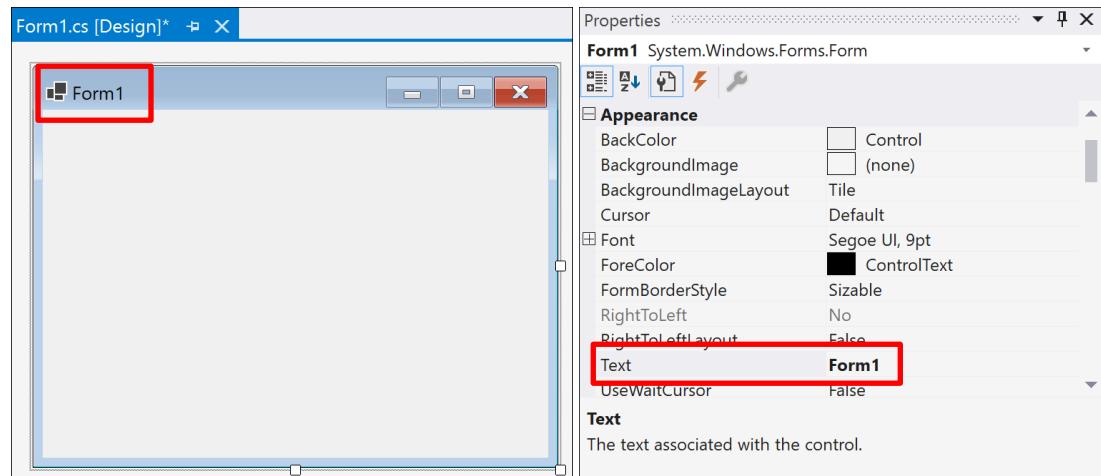
Pada contoh kasus ini akan diberikan penjelasan beberapa control yaitu:

- Button.
- Label.
- TextBox.

Untuk keperluan di atas dibuat project Windows Forms App (.NET Core) dengan nama SimpleCalc. Selanjutnya pembuatan aplikasi akan berdasarkan langkah-langkah berikut ini.

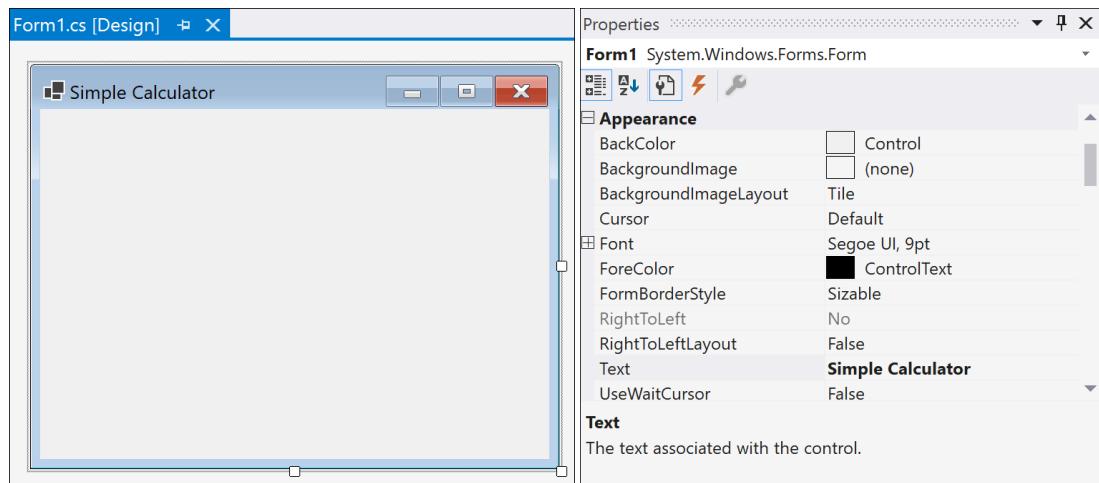
Menyiapkan Form

Setelah project dibuat maka ditampilkan sebuah Form pada visual designer. Jika Form tersebut diklik kemudian perhatikan pada area Properties maka dapat dilihat seperti pada Gambar 60.



Gambar 60. Property Form.

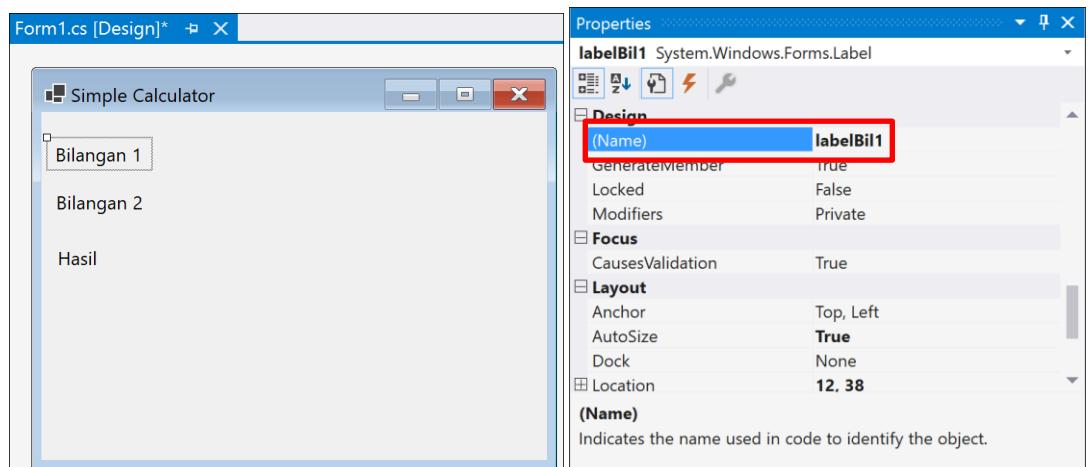
Persiapan form yang dilakukan di sini hanya mengganti title form menjadi Simple Calculator dengan cara mengganti nilai property Text hasilnya dapat dilihat pada Gambar 61.



Gambar 61. Modifikasi property form Simple Calculator.

Menyiapkan Antarmuka

Selanjutnya adalah menyiapkan antarmuka untuk kalkulator sederhana yaitu dengan menambahkan control-control ke atas form tersebut. Control yang ditambahkan adalah Label seperti pada Gambar 62. Pada control ini terdapat property yang penting yaitu Text untuk mengubah teks yang terlihat pada form. Property yang lain adalah Name, property ini tidak saja penting pada control Label namun penting untuk control-control lainnya.



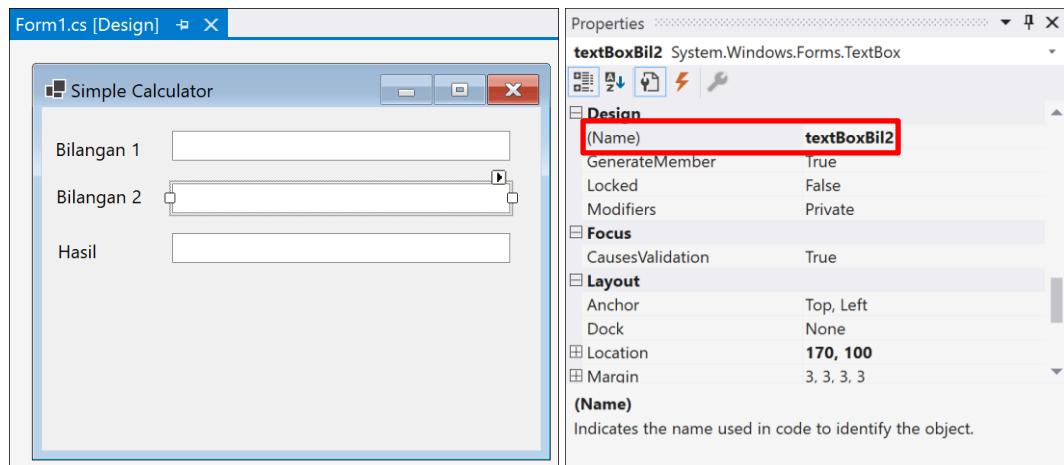
Gambar 62. Property Name pada control Label.

Pada antarmuka di atas ditambahkan dua control Label, ubah property Name untuk kedua control tersebut menjadi:

- labelBil1 untuk control Label pertama.
- labelBil2 untuk control Label kedua.
- labelHasil untuk control Label ketiga.

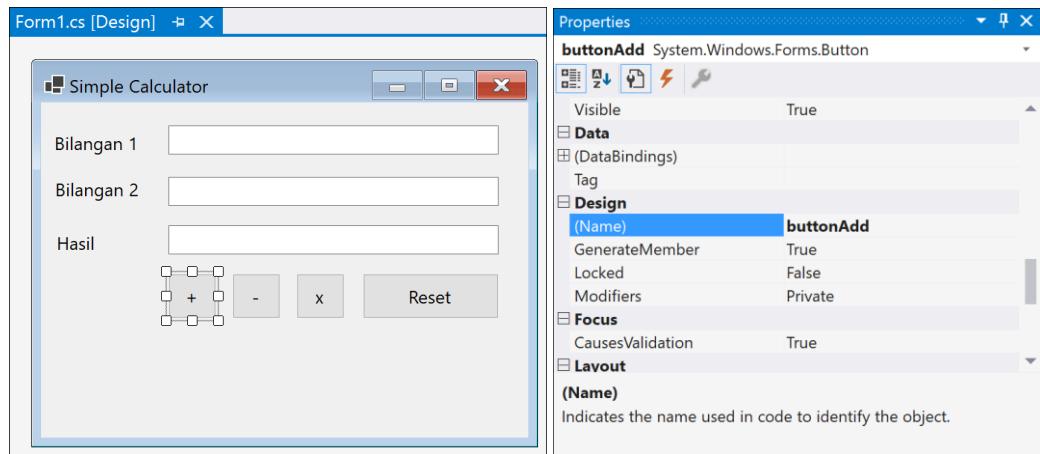
Selanjutnya adalah menambahkan control TextBox seperti pada Gambar 63. Dan ubah property Name kedua control TextBox menjadi:

- textBoxBil1 untuk control TextBox pertama.
- textBoxBil2 untuk control TextBox kedua.
- textBoxHasil untuk control TextBox ketiga.



Gambar 63. Property Name pada control TextBox.

Pemberian nilai property Name pada control ini akan berguna saat pemrograman dilakukan. Selanjutnya adalah menambahkan control Button seperti pada Gambar 64.



Gambar 64. Property Name pada control Button.

Pada antarmuka hanya ditambahkan 3 tombol, ubah property Text masing-masing tombol agar terlihat seperti gambar di atas. Kemudian ubah property Name pada control-control tersebut menjadi:

- buttonAdd untuk control Button yang pertama.
- buttonSub untuk control Button yang kedua.
- buttonMul untuk control Button yang ketiga.
- buttonReset untuk control Button yang keempat.

Menyiapkan Event

Selanjutnya klik double untuk setiap tombol tersebut. Cara tersebut adalah untuk memberikan action atau event kepada control Button tersebut. Hasilnya akan dapat dilihat perubahan kode pada file Form1.cs sebagai berikut ini.

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```

using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SimpleCalc
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            int bil1 = Int32.Parse(textBoxBil1.Text);
            int bil2 = Int32.Parse(textBoxBil2.Text);
            int hasil = bil1 + bil2;

            textBoxHasil.Text = hasil.ToString();
        }

        private void buttonSub_Click(object sender, EventArgs e)
        {

        }

        private void buttonMul_Click(object sender, EventArgs e)
        {

        }

        private void buttonReset_Click(object sender, EventArgs e)
        {

        }
    }
}

```

Selanjutnya akan ditambahkan kode untuk menangani operasi penjumlahan yang akan disimpan ke dalam method buttonAdd_Click. Berikut ada potongan kode logika untuk method ini.

```

private void buttonAdd_Click(object sender, EventArgs e)
{
    int bil1 = Int32.Parse(textBoxBil1.Text);
    int bil2 = Int32.Parse(textBoxBil2.Text);
    int hasil = bil1 + bil2;

    textBoxHasil.Text = hasil.ToString();
}

```

Dengan berasumsi nilai yang dimasukkan hanyalah bilangan bulat maka tipe data variable untuk menampung nilai dari textBoxBil1 dan textBoxBil2 adalah int. selanjutnya untuk mengakses nilai kedua control tersebut dengan mengambil nilai property Text. Karena tipe data dari property ini adalah string maka nilai perlu dikonversi dengan method Parse dari class Int3. Kemudian hasil konversi disimpan ke dalam variable bil1 dan bil2.

Selanjutnya dideklarasikan variable hasil untuk menampung operasi penjumlahan nilai dari variable bil1 dan bil2. Selanjutnya nilai variable hasil diberikan ke property Text dari control

textBoxHasil seperti contoh di atas. Namun karena tipe data property Text adalah string maka variable hasil diubah menjadi string dengan method ToString seperti contoh di atas.

Dengan paparan di atas maka telah dapat dibuat logika untuk method buttonSub_Click dan buttonMul_Click seperti kode berikut ini.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SimpleCalc
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            int bil1 = Int32.Parse(textBoxBil1.Text);
            int bil2 = Int32.Parse(textBoxBil2.Text);
            int hasil = bil1 + bil2;

            textBoxHasil.Text = hasil.ToString();
        }

        private void buttonSub_Click(object sender, EventArgs e)
        {
            int bil1 = Int32.Parse(textBoxBil1.Text);
            int bil2 = Int32.Parse(textBoxBil2.Text);
            int hasil = bil1 - bil2;

            textBoxHasil.Text = hasil.ToString();
        }

        private void buttonMul_Click(object sender, EventArgs e)
        {
            int bil1 = Int32.Parse(textBoxBil1.Text);
            int bil2 = Int32.Parse(textBoxBil2.Text);
            int hasil = bil1 * bil2;

            textBoxHasil.Text = hasil.ToString();
        }

        private void buttonReset_Click(object sender, EventArgs e)
        {
            textBoxBil1.Text = "";
            textBoxBil2.Text = "";
        }
    }
}
```

Pada method buttonReset_Click berfungsi untuk mengosongkan nilai pada control textBoxBil1 dan textBoxBil2 dengan cara di atas.

Quisioner

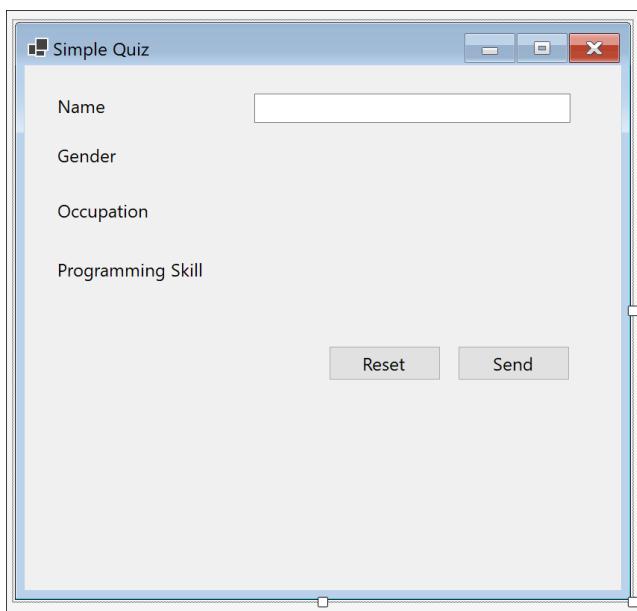
Contoh kasus kedua adalah membuat quisioner sederhana. Pada kasus selain menggunakan control-control yang telah dijelaskan di atas juga diperkenalan control-control baru yaitu:

- RadioButton.
- GroupBox yang bisa dilihat pada kelompok Container (.NET Core).
- CheckBox.
- ComboBox.

Untuk kasus ini dibuat project Windows Froms Apps (.NET Core) dengan nama SimpleQuiz. Kemudian beri title form menjadi Simple Quiz agar terlihat seperti pada Gambar 65.

Menyiapkan Antarmuka

Kemudian atur antarmuka seperti pada gambar di bawah ini.



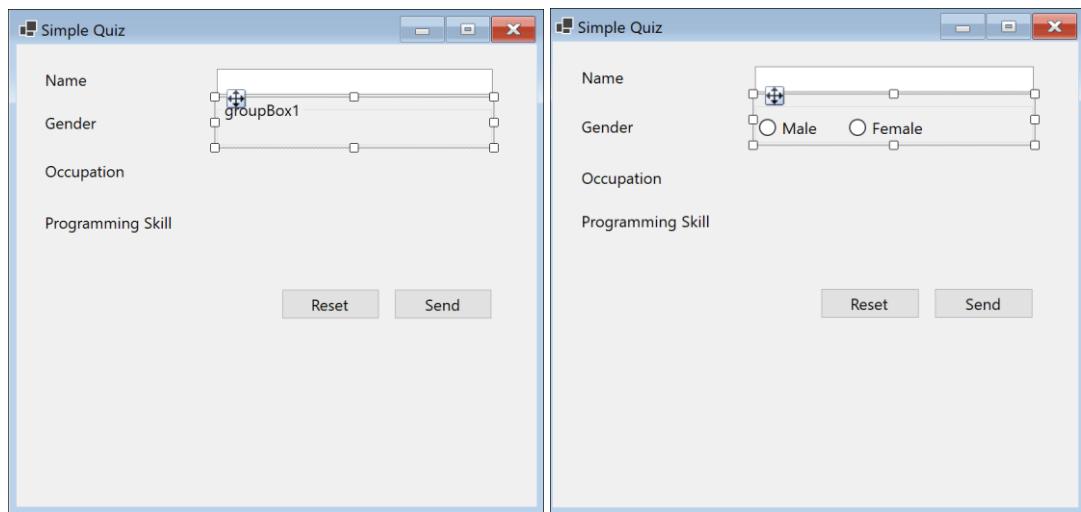
Gambar 65. Antarmuka SimpleQuiz.

Ubah nilai property Name untuk control-control berikut ini:

- Control TextBox menggunakan nilai property Name adalah textBoxName.
- Control Button Send menggunakan nilai property Name adalah textBoxSend.
- Control Button Reset menggunakan nilai property Name adalah textBoxReset.

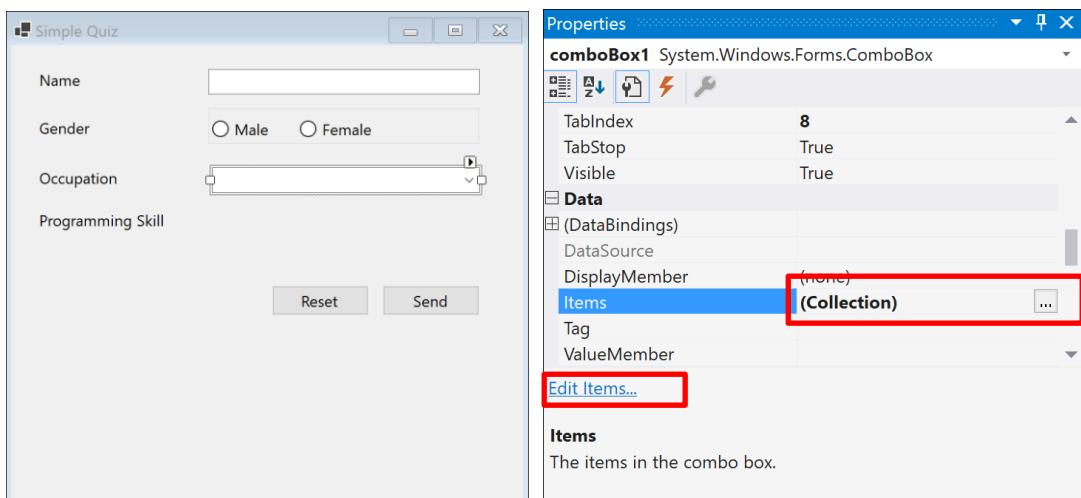
Selanjutnya adalah menambahkan pilihan untuk gender yaitu male dan female. Karena ini adalah pilihan salah satu saja maka digunakan control RadioButton. Untuk kebutuhan tersebut maka diperlukan control GroupBox. Maka hal pertama adalah tambahkan control tersebut di bawah control TextBox seperti pada Gambar 66 bagian sebelah kiri. Kemudian kosongkan nilai property Text. Selanjutnya tambahkan dua control RadioButton ke dalam control GroupBox. Ubah property Text masing-masing control RadioButton agar terlihat seperti pada Gambar 66 bagian sebelah kanan. Tujuan mengelompokkan control RadioButton dengan menggunakan control GroupBox adalah agar user hanya dapat memilih salah satu saja, Male atau Female. Untuk masing-masing control RadioButton ganti nilai property Name menjadi:

- radioButtonMale.
- radioButtonFemale.



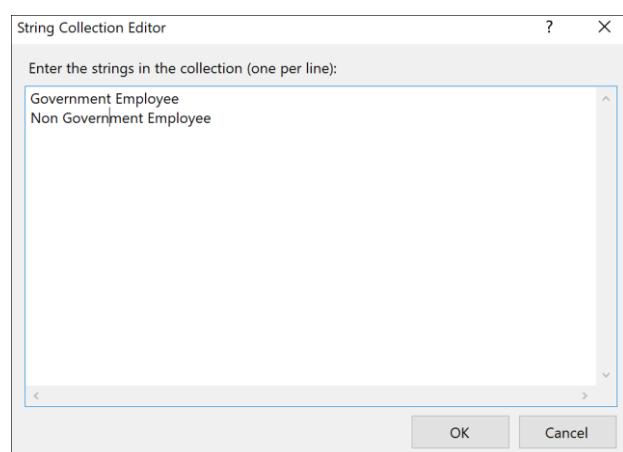
Gambar 66. RadioBox di dalam GroupBox.

Selanjutnya adalah menambahkan contro ComboBox untuk memilih nilai Occupation. Untuk menambahkan item pilihan pada control ini dapat dilakukan dengan cara klik link Edit Items atau tombol disamping tulisan (Collection).



Gambar 67. ComboBox untuk Occupation.

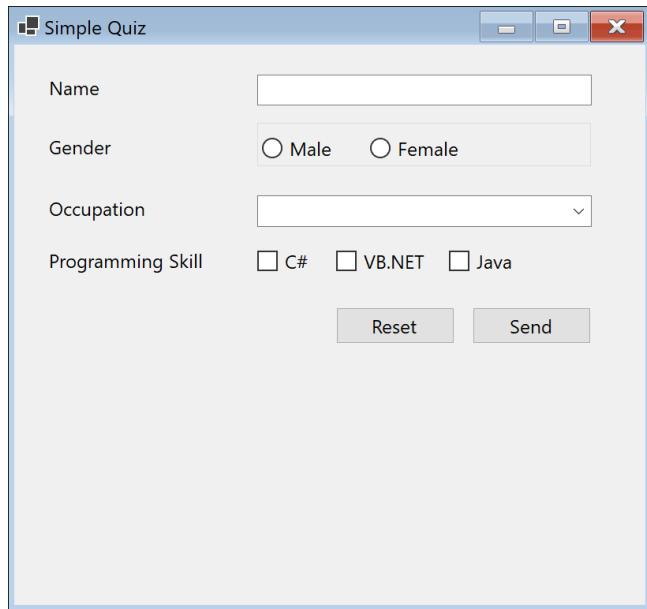
Setelah link Edit Items diklik maka akan ditampilkan window berikut ini.



Gambar 68. Edit item occupation.

Tambahkan nilai pekerjaan diinginkan sebagai contoh adalah Government Employee dan Non Government Employee. Kemudian klik tombol OK. Kemudian berikan nilai comboBoxOccupation kepada property Name.

Selanjutnya adalah menambahkan beberapa control CheckBox untuk pilihan Programming Skill. Pemilihan penggunaan control ini karena user dapat memilih lebih dari satu pilihan yang ada. Berikut adalah mengubah property Text agar terlihat seperti pada Gambar 69.

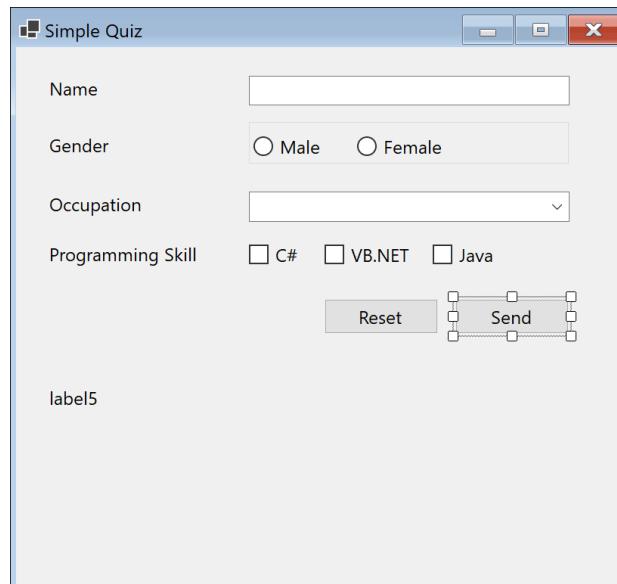


Gambar 69. CheckBox Programming Skill.

kemudian untuk setiap control tersebut ubah property Name menjadi sebagai berikut:

- checkBoxCSharp.
- checkBoxVBNET.
- checkBoxJava.

Yang terakhir adalah menambahkan control Label dengan posisi seperti pada Gambar 70.



Gambar 70. Label hasil.

Berikan nilai Name adalah labelHasil untuk control tersebut. Label ini digunakan untuk menampilkan hasil ketika tombol Send diklik.

Menyiapkan Event

Event diberikan kepada dua control Button di atas yaitu untuk tombol Send dan Reset. Caranya sama seperti yang telah diberikan di sub bab sebelumnya yaitu dengan klik double pada kedua tombol tersebut.

Selanjutnya ketik kode berikut ini.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SimpleQuiz
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            labelHasil.Visible = false;
        }

        private void buttonSend_Click(object sender, EventArgs e)
        {
            string hasil = "";
            string name = textBoxName.Text;

            string gender = "";
            if (radioButtonMale.Checked)
            {
                gender = "Male";
            } else if (radioButtonFemale.Checked)
            {
                gender = "Female";
            } else
            {
                MessageBox.Show("Please choose your gender.");
            }

            string occupation = comboBoxOccupation.Text;
            if (String.IsNullOrEmpty(occupation))
            {
                MessageBox.Show("Please choose your current occupation.");
            }

            string skill = "";
            if (checkBoxCSharp.Checked)
            {
                skill += "C#, ";
            }
        }
    }
}
```

```

        if (checkBoxVBNET.Checked)
        {
            skill += "VB, ";
        }

        if (checkBoxJava.Checked)
        {
            skill += "Java";
        }

        if (String.IsNullOrEmpty(skill))
        {
            MessageBox.Show("Please choose your programming skill.");
        }

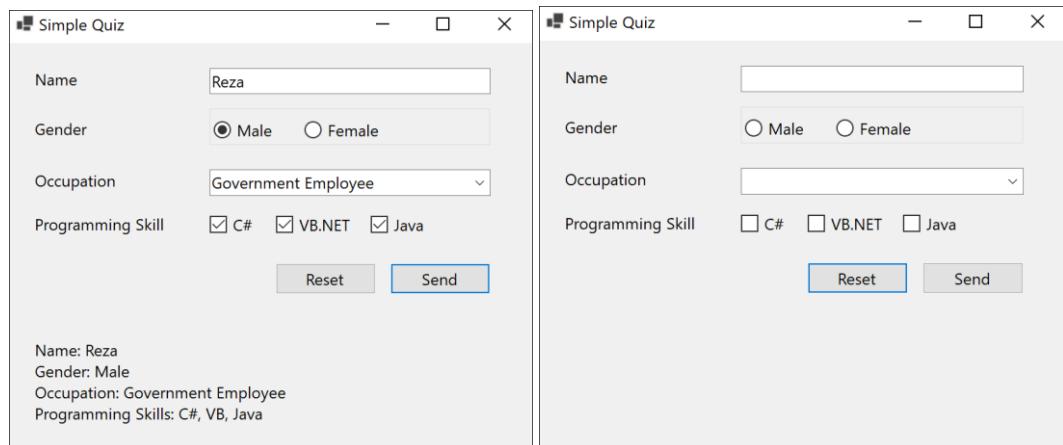
        if(!String.IsNullOrEmpty(name) || !String.IsNullOrEmpty(gender)
        || !String.IsNullOrEmpty(occupation) || !String.IsNullOrEmpty(skill))
        {
            hasil = "Name: " + name + "\n";
            hasil += "Gender: " + gender + "\n";
            hasil += "Occupation: " + occupation + "\n";
            hasil += "Programming Skills: " + skill + "\n";

            labelHasil.Visible = true;
            labelHasil.Text = hasil;
        }
    }

    private void buttonReset_Click(object sender, EventArgs e)
    {
        textBoxName.Text = "";
        radioButtonMale.Checked = false;
        radioButtonFemale.Checked = true;
        comboBoxOccupation.SelectedIndex = -1;
        checkBoxCSharp.Checked = false;
        checkBoxVBNET.Checked = false;
        checkBoxJava.Checked = false;
        labelHasil.Visible = false;
    }
}
}

```

Jika semua nilai diisi maka akan dapat dilihat hasilnya seperti pada gambar berikut.

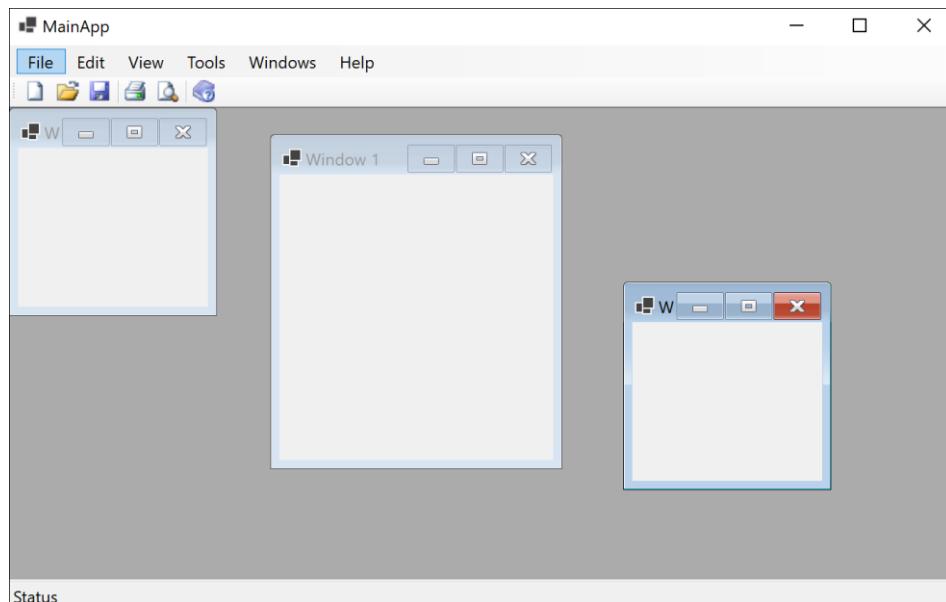


Gambar 71. Hasil akhir Simple Quiz.

Hasil ketika form diisi dapat dilihat pada gambar kiri. Hal itu terjadi ketika tombol Send diklik. Sedangkan ketika tombol Reset diklik akan dilihat hasilnya seperti pada gambar sebelah kanan.

Multiple Document Interfaces (MDI)

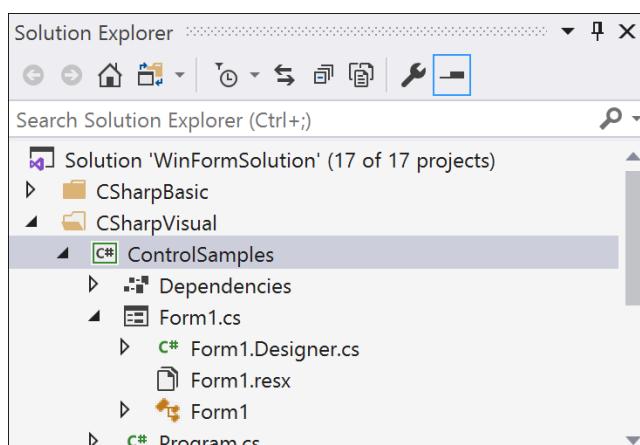
Secara sederhana MDI dapat diartikan sebuah aplikasi seperti pada Gambar 72. Pada gambar dapat dilihat sebuah aplikasi dengan form utama yang memiliki menu dan toolbar. Form ini dapat disebut sebagai form induk Kemudian didalamnya terdapat form-form lain seperti terlihat pada gambar terdapat tiga form. Form-form ini dapat disebut sebagai form anak.



Gambar 72. Contoh Multiple Document Interfaces (MDI)

Sub bab ini memberikan langkah-langkah untuk membuat seperti aplikasi di atas. untuk setiap form anak akan digunakan untuk berkenalan dengan control-control yang belum dibahas.

Untuk itu perlu dibuat project Windows Forms App (.NET Core) baru. Nama project untuk aplikasi ini adalah ControlSamples.

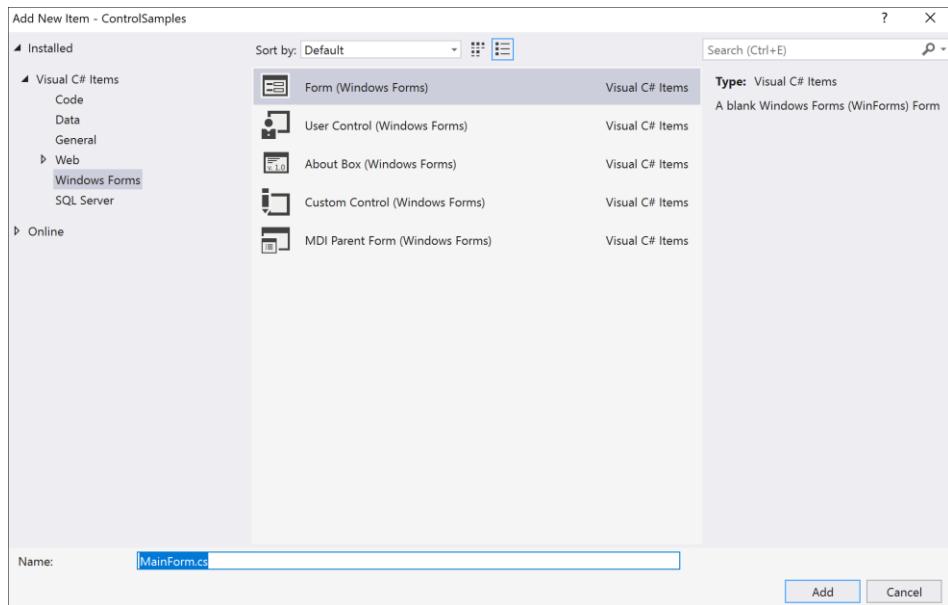


Gambar 73. Project ControlSamples.

Menyiapkan Form Induk

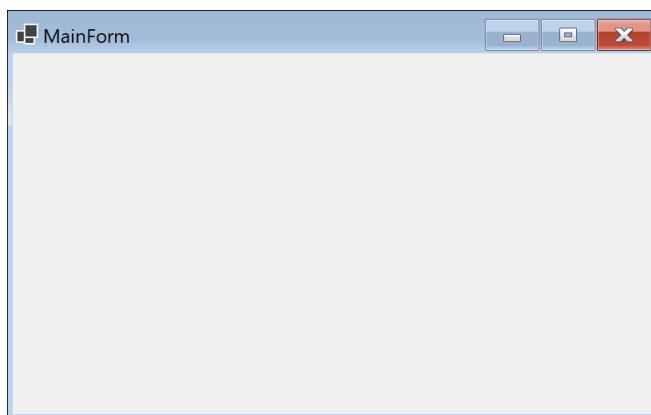
Saat project baru dibuat maka secara default telah ada sebuah form dengan nama Form1 seperti yang terlihat pada Gambar 73. Pada project ini akan ditambahkan form baru yang akan menjadi form induk. Untuk membuat form baru lakukan langkah-langkah berikut ini.

Yang pertama adalah klik kanan pada project ControlSamples kemudian pilih Add > New Item. Pada window Add New Item pilih Windows Forms > Form (Windows Forms) kemudian pada kolom Name isikan nama form yang diinginkan. Pada contoh ini digunakan nama MainForm.cs. Kemudian klik tombol Add.



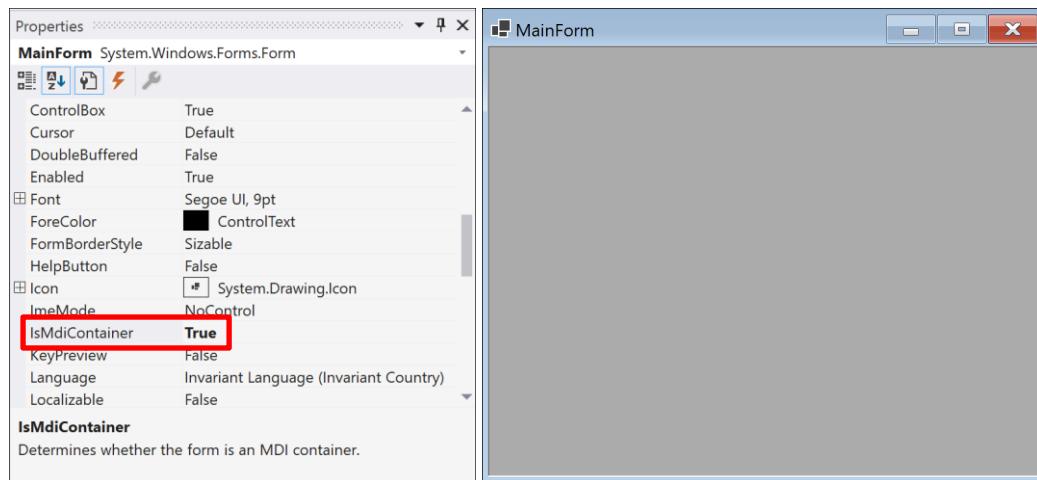
Gambar 74. Membuat form induk MainForm.cs.

Saat ini MainForm ini hanyalah form biasa seperti Form1 yang telah ada dengan tampilan seperti gambar di bawah ini.



Gambar 75. Tampilan MainForm default.

Untuk membuat form ini menjadi form induk maka perlu menganti nilai property IsMdiContainer menjadi True seperti pada Gambar 76 di bagian kiri. Hasilnya bentuk form induk menjadi berbeda seperti yang dapat dilihat pada bagian kiri gambar di bawah ini.



Gambar 76. MainForm menjadi form induk.

Dan ubah property Text dengan title yang diinginkan. Pada contoh ini digunakan nilai Control Samples sebagai nilai property Text. Jika ingin membuat agar ukuran windows form induk ini sesuai dengan ukuran layar maka dapat diubah nilai property WindowState menjadi Maximized.

Selanjutnya edit file Program.cs dengan mengganti baris berikut ini.

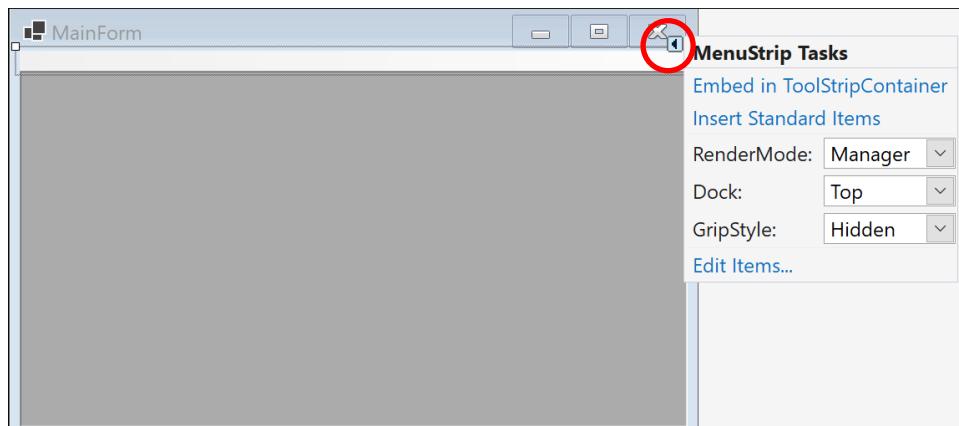
```
Application.Run(new Form1());
```

Menjadi seperti baris ini.

```
Application.Run(new MainForm());
```

Menu

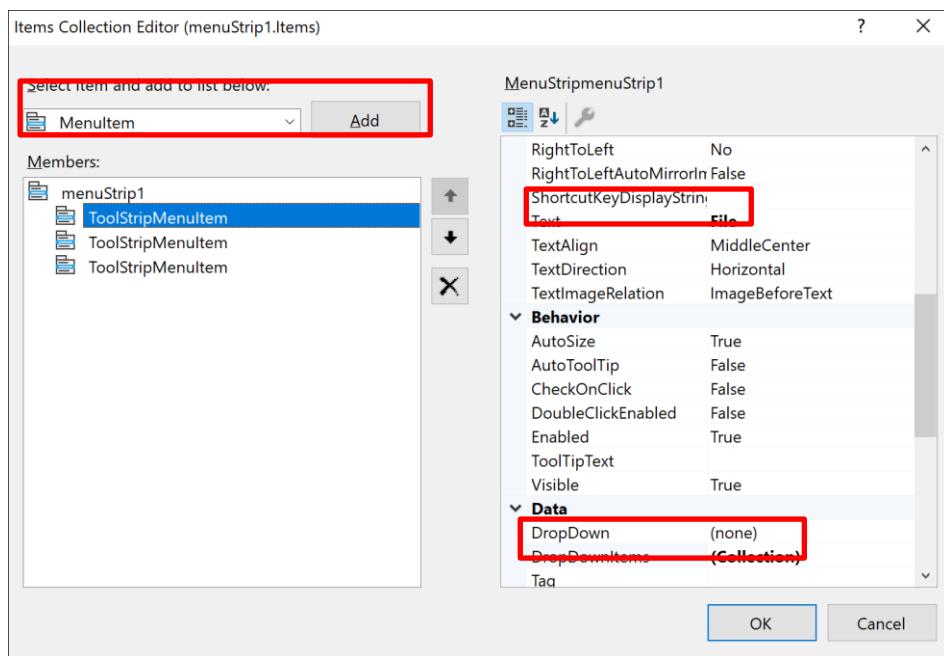
Setelah form induk telah siap, selanjutnya adalah menambahkan menu. Control yang digunakan untuk membuat menu adalah MenuStrip yang terdapat pada kelompok Menus & Toolbars di Toolbox. Tarik control tersebut dan letakkan pada form induk. Secara default lokasi control ini adalah pada posisi atas seperti yang dilihat pada Gambar 77.



Gambar 77. Control MenuStrip.

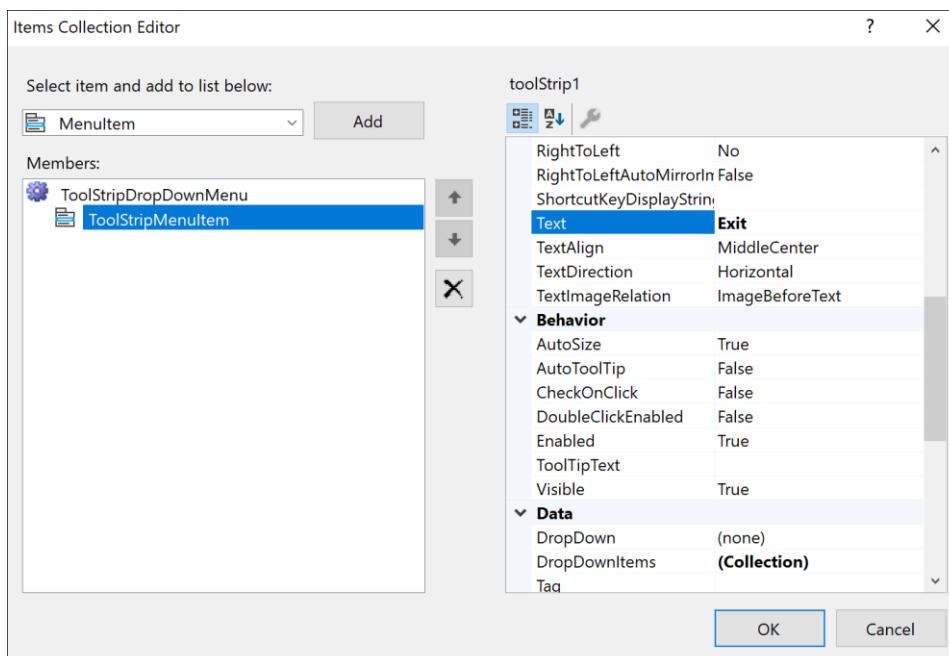
Pada gambar terlihat menu masih kosong. Untuk menambahkan item-item pada menu klik tombol segitiga yang ada pada pojok atas kanan dari control tersebut maka akan keluar context menu seperti pada gambar di atas. kemudian klik link Edit Items. Maka akan ditampilkan window Items Collection Editor seperti pada Gambar 78.

Untuk menambahkan item menu, pilih MenuItem pada combo box kemudian klik tombol Add. Tambahkan tiga item seperti terlihat pada gambar. Kemudian masing-masing item ubah property Text menjadi File, Controls dan Help.



Gambar 78. Items Collection Editor.

Selanjutnya ditambahkan item anak untuk setiap item tersebut dengan cara mengklik tombol [...] di bagian (Collection) pada Property DropDownListItems. Pada contoh ini ditambahkan item anak pada ToolStripMenuItem yang pertama. Setelah tombol [...] diklik ditampilkan window seperti pada gambar di bawah ini. Pilih MenuItem dan klik tombol Add. Kemudian klik tombol OK.



Gambar 79. Menambahkan item anak pada item ToolStripMenuItem pertama.

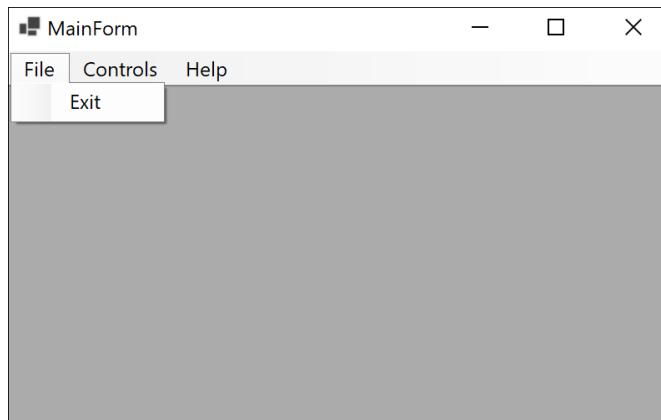
Kemudian tambahkan item anak pada ToolStripMenuItem yang kedua dengan menambahkan sebuah item anak dengan property Text adalah Example 1. Dan item anak

pada item yang ketiga yaitu item anak dengan property Text adalah About. Setelah selesai, klik tombol OK.

Maka secara keseluruhan dapat dilihat hierarki menu sebagai berikut:

- File (Name=toolStripMenuItem1)
 - o Exit (Name=toolStripMenuItem2)
- Controls (Name=toolStripMenuItem3)
 - o Example 1 (Name=toolStripMenuItem4)
- Help (Name=toolStripMenuItem5)
 - o About (Name=toolStripMenuItem6)

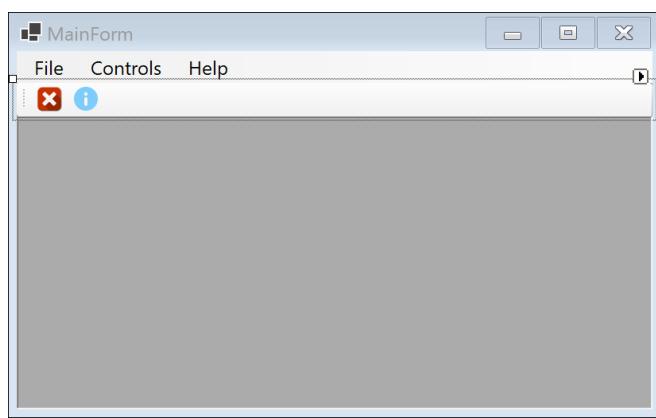
Jika ingin melihat hasilnya, jalankan project dengan klik kanan pada project kemudian pilih Debug > Start New Instance. Hasilnya dapat dilihat pada Gambar 80.



Gambar 80. MainForm dengan menu.

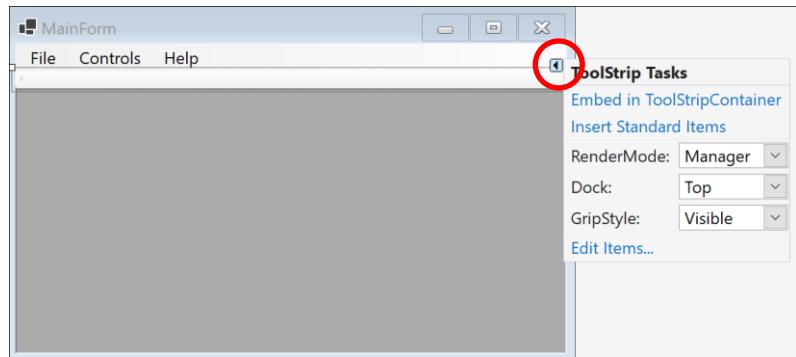
Toolbar

Toolbar adalah control berupa tombol bergambar yang biasanya posisinya berada di bawah menu. Pada Gambar 81 berikut ini adalah contoh toolbar dengan dua buah item berupa tombol.



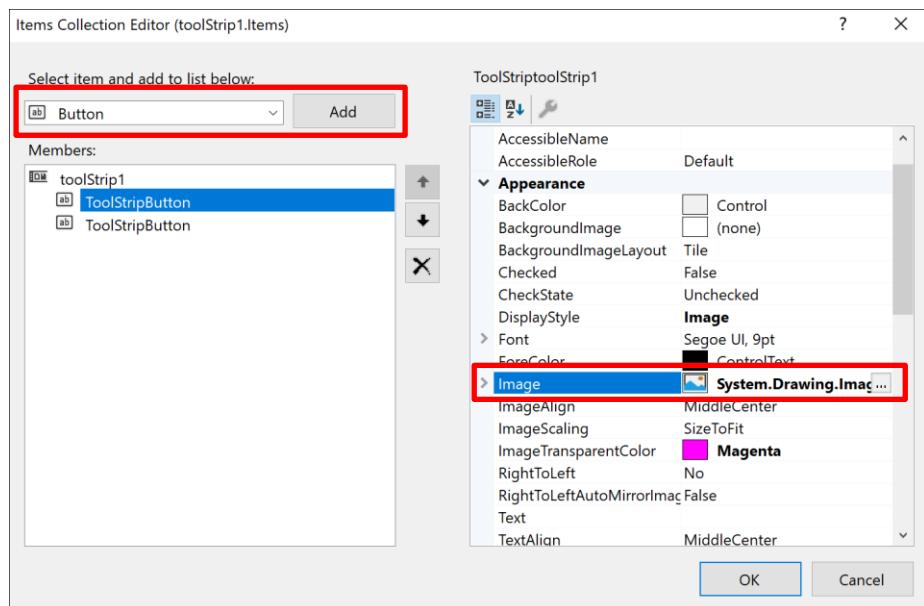
Gambar 81. Control Toolbar.

Untuk menambahkan control ini adalah dengan cara menambahkan control ToolStrip pada form di atas. Setelah control ditambahkan, maka cara untuk menambahkan item sama seperti cara menambahkan item pada menu yang telah dibahas pada sub bab sebelumnya.



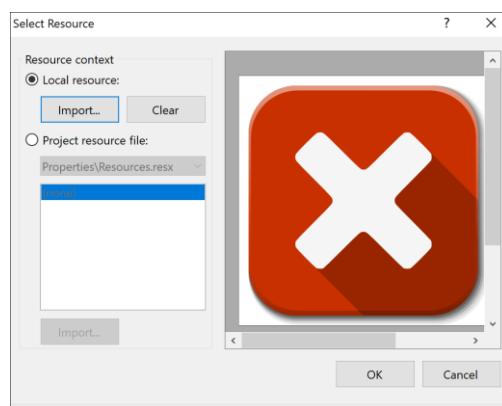
Gambar 82. Edit item pada ToolStrip.

Setelah control ditambahkan pada form induk, klik setiga yang ada pada lingkaran pada gambar di atas. Kemudian pilih link Edit Items. Selanjutnya tambahkan dua ToolStripButton dengan mengklik tombol Add pada kotak kiri atas.



Gambar 83. Windows Items Collection Editor untuk toolbar.

Kemudian ubah property Image (kotak pada sebelah kanan) dengan mengklik tombol [...] untuk mengubah atau memilih gambar yang akan digunakan sebagai tombol. Setelah tombol [...] diklik maka akan ditampilkan window berikut ini.



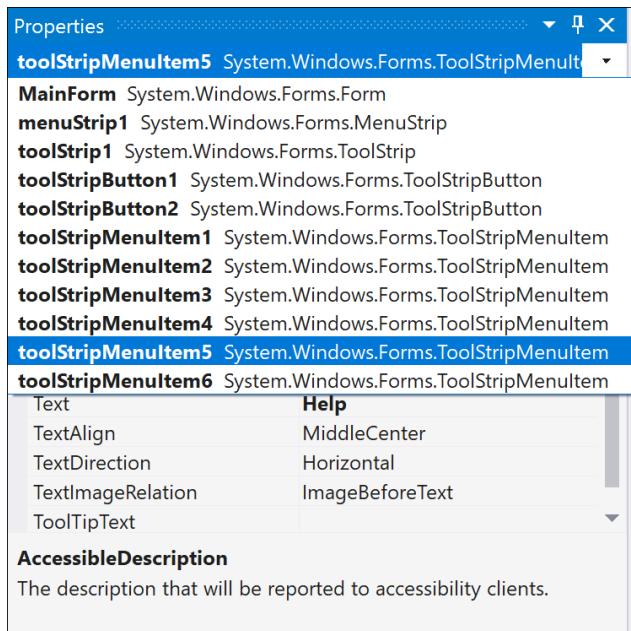
Gambar 84. Window Select Resource untuk memilih gambar.

Pada window Select Resource pilih Local resource dan klik tombol Import kemudian pilih gambar yang diinginkan. Kemudian klik tombol OK. Lakukan hal yang sama untuk item kedua.

Setelah selesai maka klik tombol OK pada Gambar 83. Dan hasilnya dapat dilihat seperti pada Gambar 82.

Menyiapkan Event

Pada sub bab ini akan diberikan cara lain untuk memberikan event kepada control. Pada area Properties terdapat combo box untuk memilih control yang ingin diubah property dan event (Gambar 85).

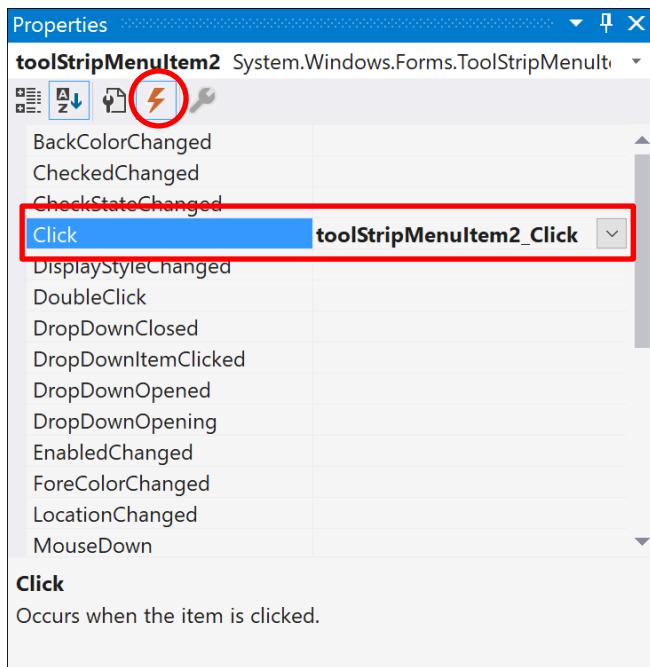


Gambar 85. Combo box untuk memilih control pada area Properties.

Item-item yang akan diberikan event adalah:

- Item menu dengan property Text=Exit dalam hal ini adalah toolStripMenuItem2 (nama ini mungkin berbeda untuk setiap pembaca, tergantung urutan pembuatannya).
- Item menu dengan property Text=Example 1 yaitu toolStripMenuItem4.
- Item menu dengan property Text=About yaitu toolStripMenuItem6.
- Item toolbar untuk Exit yaitu toolStripButton1.
- Item toolbar untuk About yaitu toolStripButton2.

Langkah selanjutnya pilih toolStripMenuItem2 pada combo box di area Properties. Seperti pada Gambar 86, kemudian klik tombol dengan icon petir seperti pada lingkaran. Kemudian klik double pada bagian Click, maka secara otomatis akan ditambahkan event untuk control ini. Lakukan hal yang sama untuk control-control di atas.



Gambar 86. Event pada toolStripMenuItem2.

Sehingga didapat kode sebagai berikut.

```
MainForm.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ControlSamples
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void toolStripMenuItem2_Click(object sender, EventArgs e)
        {

        }

        private void toolStripMenuItem4_Click(object sender, EventArgs e)
        {

        }

        private void toolStripMenuItem6_Click(object sender, EventArgs e)
        {

        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
    }
```

```

        }

        private void toolStripButton2_Click(object sender, EventArgs e)
        {
        }
    }
}

```

Event Menutup Aplikasi

Untuk menutup aplikasi maka perlu ditambahkan logika yang akan dijalankan ketika tombol dan menu Exit diklik. Maka perlu diedit file MainForm.cs menjadi sebagai berikut.

```

MainForm.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ControlSamples
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        // event untuk Exit
        private void toolStripMenuItem2_Click(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to exit?",
                                              "Exit",
                                              MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                Application.Exit();
            }
        }

        // event untuk memanggil form anak
        private void toolStripMenuItem4_Click(object sender, EventArgs e)
        {

        }

        // event untuk about
        private void toolStripMenuItem6_Click(object sender, EventArgs e)
        {

        }

        // event untuk exit
        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to exit?",
                                              "Exit",
                                              MessageBoxButtons.YesNo);
        }
    }
}

```

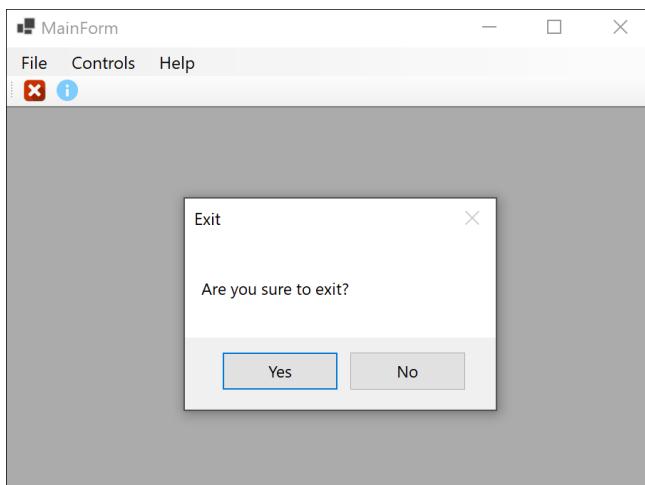
```

        MessageBoxButtons.YesNo);
    if (confirmResult == DialogResult.Yes)
    {
        Application.Exit();
    }
}

// event untuk about
private void toolStripButton2_Click(object sender, EventArgs e)
{
}
}
}

```

Pada kode di atas dapat dilihat ketika tombol dan menu Exit diklik akan ditampilkan window untuk konfirmasi. Untuk window konfirmasi dapat menggunakan method MessageBox.Show. Jika tombol Yes yang diklik maka akan dieksekusi method Application.Exit untuk menutup aplikasi.

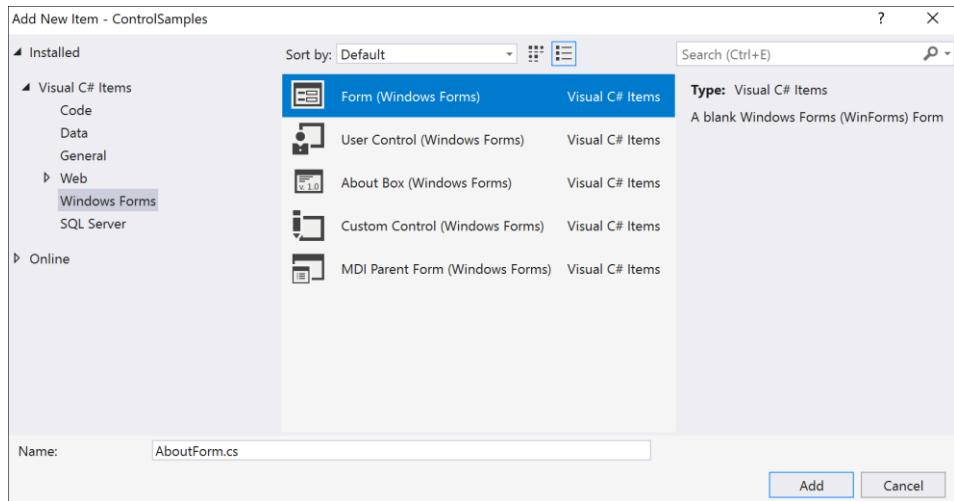


Gambar 87. Window konfirmasi untuk keluar.

Event Menampilkan Window About

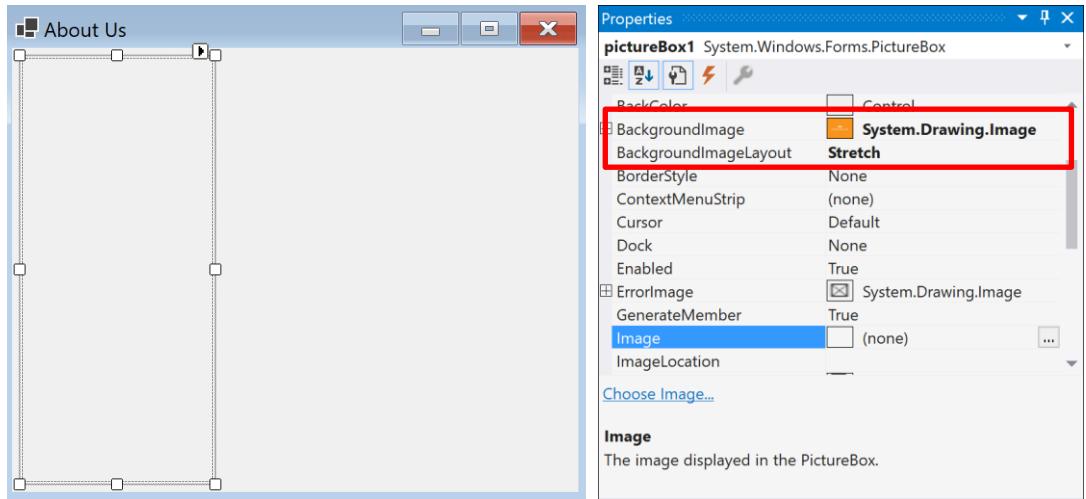
Langkah pertama adalah membuat form anak yang akan dipanggil oleh form induk. Untuk membuat form anak adalah klik kanan pada project ControlSamples kemudian pilih Add > New Item.

Pada window Add New Item pilih Form (Windows Forms) kemudian berikan nama form pada kolom Name dengan nama AboutForm.cs. Kemudian klik tombol Add.



Gambar 88. Membuat AboutForm.cs.

Selanjutnya adalah mendesain form ini agar terlihat sebagai form about pada umumnya. Pertama tambahkan control Image pada form seperti pada Gambar 89.

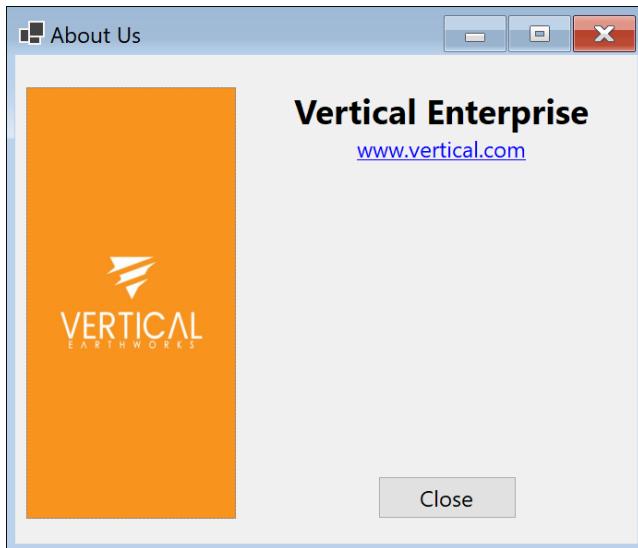


Gambar 89. Control Image pada form About.

Kemudian klik tombol [...] pada property BackgroundImage untuk memilih gambar yang ingin digunakan sebagai logo. Dan pada BackgroundImageLayout pilih Stretch.

Selanjutnya dapat ditambahkan control Label, LinkLabel dan Button untuk membuat antarmuka seperti Gambar 90. Untuk membuat agar teks label terlihat tebal dan ukurannya lebih besar dapat dilakukan dengan mengubah property Font.

Selanjutnya adalah memberi event kepada control LinkLabel dan Button dengan cara klik double pada kedua control tersebut.



Gambar 90. Antarmuka form About yang lengkap.

Kemudian tambahkan kode logika agar isi file AboutForm.cs seperti berikut ini.

```
AboutForm.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ControlSamples
{
    public partial class AboutForm : Form
    {
        public AboutForm()
        {
            InitializeComponent();
        }

        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            System.Diagnostics.Process.Start("http://www.vertical.com");
        }

        private void buttonClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Selanjutnya adalah memodifikasi file MainForm.cs agar ketika menu atau tombol About diklik akan menampilkan AboutForm ini.

```
MainForm.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
```

```

using System.Windows.Forms;

namespace ControlSamples
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        // event untuk Exit
        private void toolStripMenuItem2_Click(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to exit?",
                "Exit",
                MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                Application.Exit();
            }
        }

        // event untuk memanggil form anak
        private void toolStripMenuItem4_Click(object sender, EventArgs e)
        {

        }

        // event untuk about
        private void toolStripMenuItem6_Click(object sender, EventArgs e)
        {
            ShowAboutForm();
        }

        // event untuk exit
        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to exit?",
                "Exit",
                MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                Application.Exit();
            }
        }

        // event untuk about
        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            ShowAboutForm();
        }

        // method untuk menampilkan AboutForm
        private void ShowAboutForm()
        {
            bool isOpen = false;
            foreach (Form f in Application.OpenForms)
            {
                if (f.Text == "About Us")
                {
                    if (!isOpen)
                    {
                        f.Activate();
                        isOpen = true;
                    }
                }
            }
        }
    }
}

```

```

        isOpen = true;
        f.Focus();
        break;
    }
}

if (!isOpen)
{
    AboutForm form = new AboutForm();
    form.MdiParent = this;
    form.Show();
}
}
}
}

```

Pada kode di atas diperkenalkan penggunaan method. Method yang dibuat adalah ShowAboutForm. Method ini memiliki logika sebagai berikut:

- Memeriksa keberadaan form About sudah ditampilkan berdasarkan property Text.
- Jika form dengan nilai Text="About Us" ada maka nilai variable isOpen adalah true.
- Jika tidak ditemukan form dengan nilai Text="About Us" maka nilai variable isOpen adalah false.
- Kemudian dilakukan jika nilai isOpen adalah false maka dilakukan langkah untuk menampilkan form tersebut dengan cara menjalankan statement-statement berikut ini.

```

if (!isOpen)
{
    AboutForm form = new AboutForm();
    form.MdiParent = this;
    form.Show();
}

```

Yang dilakukan oleh kode di atas adalah:

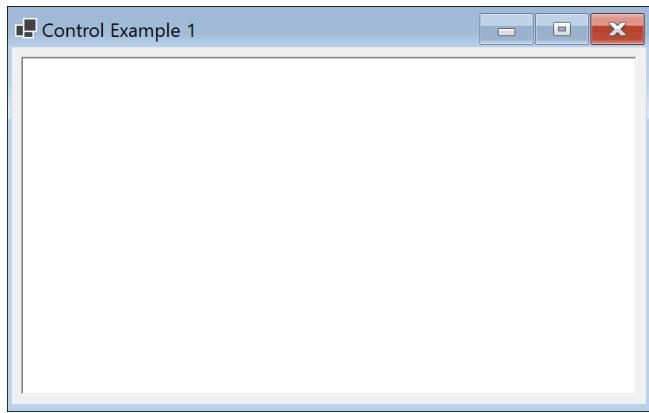
- Membuat object form dari class AboutForm.
- Selanjutnya menentukan induk dari object form.
- Kemudian menampilkan object form.

Setelah method ShowAboutForm dibuat, maka method ini dipanggil oleh event:

- toolStripMenuItem6_Click.
- toolStripButton2_Click.

Event Menampilkan Example 1

Langkah pertama yang umumnya dilakukan adalah menambahkan Form (Windows Forms). Nama file yang digunakan adalah ControlExample1.cs. Ubah property Text form ini menjadi "Control Example 1". Kemudian tambahkan control RichTextBox ke dalam form.



Gambar 91. Control Example 1.

Selanjutnya adalah menambahkan kode pada MainForm.cs agar ketika menu Example 1 diklik akan ditampilkan form di atas. Kode ditambahkan pada event toolStripMenuItem4_Click. Dengan kode seperti berikut ini.

```
MainForm.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ControlSamples
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        // event untuk Exit
        private void toolStripMenuItem2_Click(object sender, EventArgs e)
        {
            var confirmResult = MessageBox.Show("Are you sure to exit?", "Exit", MessageBoxButtons.YesNo);
            if (confirmResult == DialogResult.Yes)
            {
                Application.Exit();
            }
        }

        // event untuk memanggil form anak
        private void toolStripMenuItem4_Click(object sender, EventArgs e)
        {
            bool isOpen = false;
            foreach (Form f in Application.OpenForms)
            {
                if (f.Text == "Control Example 1")
                {
                    isOpen = true;
                    f.Focus();
                    break;
                }
            }
        }
}
```

```

        }

        if (!isOpen)
        {
            ControlExample1 form = new ControlExample1();
            form.MdiParent = this;
            form.Show();
        }
    }

    // event untuk about
private void toolStripMenuItem6_Click(object sender, EventArgs e)
{
    ShowAboutForm();
}

// event untuk exit
private void toolStripButton1_Click(object sender, EventArgs e)
{
    var confirmResult = MessageBox.Show("Are you sure to exit?",
                                         "Exit",
                                         MessageBoxButtons.YesNo);
    if (confirmResult == DialogResult.Yes)
    {
        Application.Exit();
    }
}

// event untuk about
private void toolStripButton2_Click(object sender, EventArgs e)
{
    ShowAboutForm();
}

// method untuk menampilkan AboutForm
private void ShowAboutForm()
{
    bool isOpen = false;
    foreach (Form f in Application.OpenForms)
    {
        if (f.Text == "About Us")
        {
            isOpen = true;
            f.Focus();
            break;
        }
    }

    if (!isOpen)
    {
        AboutForm form = new AboutForm();
        form.MdiParent = this;
        form.Show();
    }
}
}

```

Kesimpulan

Dari pembahasan pada sub bab ini telah dipelajari beberapa cara membuat aplikasi desktop dengan konsep MDI. Selain itu juga dipelajari control-control sebagai berikut:

- MenuStrip.
- ToolStrip.
- RichTextBox.
- Image.
- LinkLabel.

6

Akses Database

Pada saat buku ini ditulis masih belum tersedia control-control untuk akses data seperti Data Grid dan lain-lain. control-control tersebut direncanakan akan ada pada bulan Mei 2020. Namun untuk mengakses dan operasi database telah tersedia Entity Framework Core. Maka pada bab ini dijelaskan bagaimana cara melakukan koneksi dan operasi ke database MS SQL Server dari aplikasi desktop.

Pendahuluan

Entity Framework adalah framework untuk mempermudah mengakses database. Framework ini awalnya dibangun sebagai bagian dari .NET framework yang hanya dapat digunakan pada platform Microsoft. Tetapi dengan dikembangkannya .NET Core yang bersifat multiplatform, maka Entity Framework Core juga dapat digunakan pada berbagai platform.

Entity Framework Core atau disingkat EF Core adalah object-relational mapper (OR/M) yang memungkinkan software developer dapat bekerja dengan database dengan object .NET. Hal ini mengurangi kode program untuk mengakses database, karena digantikan oleh class dan method yang telah disediakan oleh framework ini.

EF Core mendukung berbagai macam database, tetapi tergantung ketersediaan provider database. Saat buku ini ditulis telah tersedia provider database sebagai berikut:

1. MS SQL Server.
2. MS SQL Server Compact Edition.
3. SQLite.
4. MySQL, tersedia tiga provider untuk database MySQL yaitu:
 - o MySQL Official., provider
 - o MySQL Pomelo.
 - o MySQL Sapient Guardian.
5. PostgreSQL.
6. Oracle dan lain-lain.

Tetapi tidak semua provider yang disebutkan diatas adalah gratis, ada beberapa provider database yang bersifat berbayar. Untuk mendapatkan update informasi terbaru tentang provider database dapat mengunjungi alamat berikut <https://docs.microsoft.com/en-us/ef/core/providers/>.

EF Core mendukung dua pendekatan dalam mengembangkan aplikasi, yaitu:

1. Database First, pendekatan ini umum dilakukan dimana database dan tabel-tabel di dalamnya telah terlebih dahulu dibuat. Kemudian dibuat class model berdasarkan tabel-tabel di dalam database tersebut
2. Code First, pada pendekatan ini yang dibuat terlebih dahulu adalah class-class model kemudian tabel-tabel pada database akan secara otomatis dibuat saat pertama kali aplikasi dijalankan.

Koneksi & Insert Data

Pada sub bab ini diberikan langkah-langkah untuk melakukan implementasi Entity Framework Core untuk melakukan koneksi ke database MS SQL dan menimput data dengan antarmuka yang sederhana.

Project

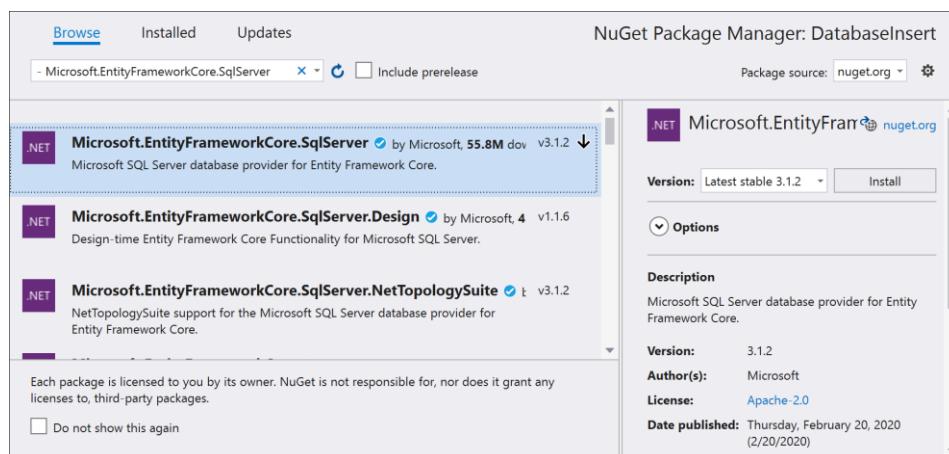
Langkah pertama adalah buat project Windows Forms App (.NET Core) dengan nama project adalah DatabaseInsert.

Library

Pada sub bab ini akan dilakukan persiapan project agar project dapat digunakan untuk melakukan koneksi ke database MS SQL. Library-library yang akan ditambahkan adalah sebagai berikut:

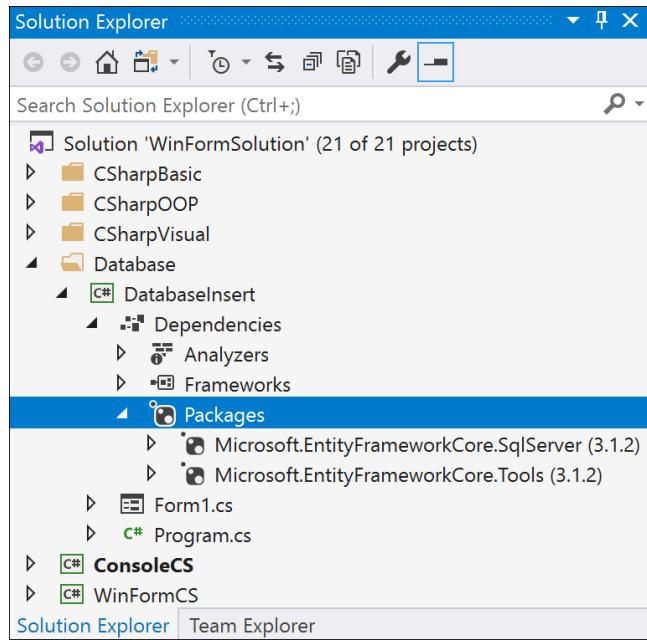
- **Microsoft.EntityFrameworkCore.SqlServer**, adalah library EF Core untuk melakukan koneksi ke database MS SQL Server.
- **Microsoft.EntityFrameworkCore.Tools**, adalah tool untuk membuat model dari database.

Untuk menginstall kedua library ini dapat dapat menggunakan NuGet Package Manager, kemudian cari ketiga library tersebut dengan cara klik kanan pada project DatabaseInsert kemudian pilih Manage NuGet Packages. Untuk menggunakan fitur ini pastikan komputer terkoneksi dengan internet. Kemudian klik tab Browse dan masukkan kata kunci sesuai nama library di atas (cetak tebal) pada kolom keyword.



Gambar 92. Nuget pacage Manager: DatabaeInsert.

Setelah ditemukan library sesuai kata kunci, pilih dengan cara diklik kemudian klik tombol Install pada sisi sebelah kanan. Kemudian ikutan langkah-langkah yang diberikan oleh installer package tersebut. Hasilnya dapat dilihat pada bagian Packages di project DatabaseInsert pada area Solution Explorer seperti terlihat pada Gambar 93. Packages..



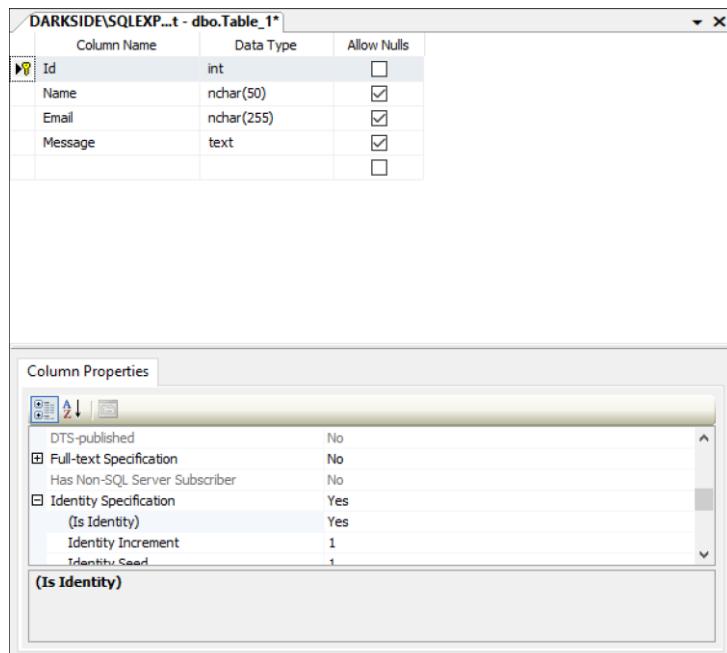
Gambar 93. Packages.

Database

Langkah berikutnya adalah membuat database pada MS SQL Server dan membuat konfigurasi connecting string.

Untuk membuat database pada MS SQL Server, dapat digunakan Microsoft SQL Server Management Studio yang dapat diunduh di <https://go.microsoft.com/fwlink/?linkid=867670>.

Langkah pertama adalah membuat database dengan Microsoft SQL Server Management Studio. Klik kanan pada Databases kemudian pilih New Database. Kemudian ketikkan nama database pada window New Database seperti pada gambar di atas. Nama database yang digunakan adalah SqlServerDbFirst. Kemudian klik OK.



Gambar 94. Membuat table GuestBook.

Langkah kedua adalah membuat tabel. Pada database SqlServerDbFirst, klik kanan pada Tables kemudian pilih Table.

Tabel memiliki field atau kolom sebagai berikut:

- Id, kolom ini adalah primary key. Tipe data kolom ini adalah integer. Agar nilai dari kolom ini dapat terisi dan nilai yang sesuai dengan urutan data yang masuk maka modifikasi nilai Identity Specification > {Is Identity} menjadi Yes.
- Name.
- Email.
- Message.

Kemudian klik tombol Save Table yang ada pada menu bar di bawah menu.

Column Name	Data Type	Allow Nulls
Id	int	<input type="checkbox"/>
Name	nchar(50)	<input checked="" type="checkbox"/>
Email	nchar(255)	<input checked="" type="checkbox"/>
Message	text	<input checked="" type="checkbox"/>

Gambar 95. Simpan table.

Connection string adalah string yang berisi nilai-nilai yang digunakan untuk melakukan koneksi ke database. Informasi ini disimpan ke dalam file App.config. Untuk menambahkan file ini, klik kanan pada project kemudian pilih Add > New Item. Pada window Add New Item pilih General > Application Configuration File. Kemudian klik tombol Add.

Gambar 96. Application Configuration File.

Edit file App.config untuk menyimpan informasi koneksi ke database seperti berikut.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <connectionStrings>
        <add name="default" connectionString="Server=.\SQLEXPRESS;
Database=SqlServerDbFirst;User
Id=sa;Password=MRezaFaisal;Trusted_Connection=True" />
    </connectionStrings>
</configuration>
```

Model

Langkah selanjutnya adalah membuat class model dan class data context. Pertama adalah membuat class entity model dengan nama GuestBook.cs dengan cara klik kanan pada project kemudian pilih Add > Class. Setelah class dibuat isi class dengan kode berikut ini.

```
GuestBook.cs
using System;
using System.Collections.Generic;
using System.Text;

namespace DatabaseInsert
{
    public partial class GuestBook
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public string Email { set; get; }
        public string Message { set; get; }
    }
}
```

Setelah itu membuat class data context dengan nama file GuestBookDataContext.cs dengan isi sebagai berikut.

```
GuestBookDataContext.cs
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
using System.Configuration;
using System.Data.SqlClient;

namespace DatabaseInsert
{
    public partial class GuestBookDataContext : DbContext
    {
        public virtual DbSet<GuestBook> GuestBook { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {

optionsBuilder.UseSqlServer(ConfigurationManager.ConnectionStrings["default"]
.ConnectionStrings);
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<GuestBook>(entity =>
            {
                entity.Property(e => e.Email).HasColumnType("nchar(255)");

                entity.Property(e => e.Message).HasColumnType("text");

                entity.Property(e => e.Name).HasColumnType("nchar(50)");
            });
        }
    }
}
```

Untuk membuat class ini diperlukan library ini Microsoft.EntityFrameworkCore, library ini adalah untuk implementasi Entity Framework.

Nama class adalah bebas, tetapi class harus merupakan turunan dari class DbContext.

```
public partial class GuestBookDataContext : DbContext
{
}
```

Langkah selanjutnya adalah membuat agar class entity GuestBook dapat digunakan untuk melakukan operasi menambah, membaca, mengedit dan menghapus data. Caranya adalah dengan membuat property GuestBooks dari class data context yang mana tipe propertnya itu dibentuk dari class DbSet.

```
public virtual DbSet<GuestBook> GuestBook { get; set; }
```

Sintaks dari kode di atas adalah sebagai berikut.

```
public virtual DbSet<NAMA_CLASS> NAMA_PROPERTY { get; set; }
```

Untuk pemetaan antara class entity model dengan tabel dapat dilihat pada method OnModelCreating. Pemetaan antara tabel ini biasanya dilakukan jika nama class entity model berbeda dengan nama table yang ada di database. Jika nama keduanya sama maka tidak perlu dilakukan pemetaan.

Untuk contoh kasus nama class entity yang berbeda dengan nama table, misal nama class entity adalah GuestBook dan nama table di database adalah buku_tamu maka perlu dilakukan pemetaan sebagai berikut ini.

```
modelBuilder.Entity<GuestBook>().ToTable("buku_tamu");
```

Sintaks umum kode di atas adalah sebagai berikut.

```
modelBuilder.Entity<GuestBook>().ToTable("NAMA_TABLE");
```

Sedangkan untuk memetakan antara property-property yang dimiliki oleh class entity model dengan atribut-atribut yang dimiliki oleh tabel dengan menggunakan kode berikut ini.

```
modelBuilder.Entity<GuestBook>(entity =>
{
    entity.Property(e => e.Email).HasColumnType("nchar(255)");
    entity.Property(e => e.Message).HasColumnType("text");
    entity.Property(e => e.Name).HasColumnType("nchar(50)");
});
```

Kasus di atas juga dilakukan jika nama property pada class entity model sama dengan nama field pada table.

Jika nama property dan field berbeda maka dapat dilakukan pemetaan dengan sintaks umum sebagai berikut ini.

```
entity.Property(e => e.Message).HasColumnName("NAMA_ATTRIBUT");
```

Kemudian untuk memetakan property yang akan menangani atribut primary key dari tabel digunakan kode berikut ini.

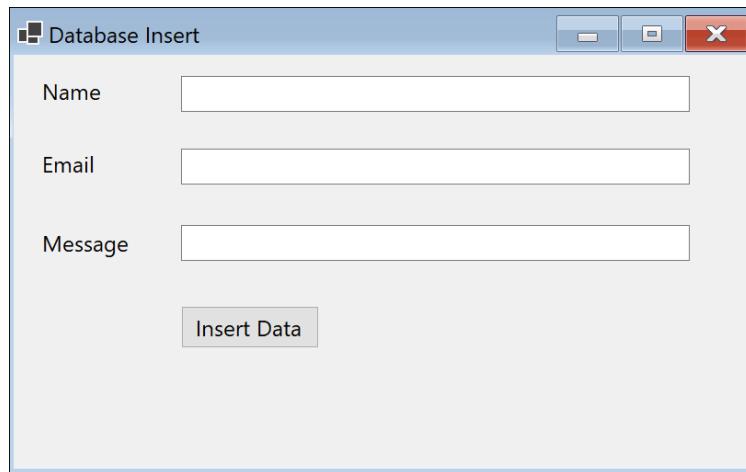
```
entity.Property(e => e.Id).HasColumnName("Id");
```

Sintaks umum kode di atas adalah sebagai berikut.

```
modelBuilder.Entity<NAMA_CLASS>().HasKey(e => new { e.NAMA_PROPERTY });
```

Antarmuka

Selanjutnya adalah membuat form untuk input dengan menambahkan control-control pada Form1 sehingga antarmukanya menjadi seperti berikut ini.



Gambar 97. Antarmuka Database Insert.

Dan berikut adalah logika untuk event saat tombol Insert Data diklik.

```
Form1.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Configuration;
using System.Data.SqlClient;

namespace DatabaseInsert
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void buttonInsertData_Click(object sender, EventArgs e)
        {
            var name = textBoxName.Text;
            var email = textBoxEmail.Text;
            var message = textBoxMessage.Text;

            try
```

```
{  
    var db = new GuestBookDataContext();  
    var item = new GuestBook { Name = name, Email = email,  
Message = message };  
    db.GuestBook.Add(item);  
    db.SaveChanges();  
  
    MessageBox.Show("Data berhasil disimpan.");  
}  
catch(Exception ex)  
{  
    MessageBox.Show("Data gagal disimpan. \n" + ex.ToString());  
}  
}  
}  
}
```

7

Penutup

Sebagai penutup, buku ini ditulis untuk para software developer yang ingin membangun aplikasi desktop dengan framework Windows Forms di atas .NET Core 3.1 dengan menggunakan tool development Visual Studio 2019. Buku ini masih dalam tahap pengembangan yang sejalan dengan pengembangan framework Windows Forms pada .NET Core. Buku akan diupdate begitu ada penambahan control dan fungsi.

Akhir kalimat, jika ada kritik dan saran dapat ditujukan langsung via email ke alamat reza.faisal [at] gmail.com.