

START DATE

11 OTTOBRE 2021

CORSO DI FORMAZIONE
PROFESSIONALE



ACADEMY

.NET CORE / C#

BENVENUTI



- Presentazione Accademy
.Net Core / C#
Trainer Spalluzzi
Francesco – Certificato
Azure (Az-900 – Dp-900)**



.NET Core c#
Presentazione

Presentazione Accademy .NET Core C#

- ✓ About Training Spalluzzi Francesco
- ✓ About Course .Net Core c#
- ✓ About Platform and IDE (Integrated Development Environment)
- ✓ About First Day of Course

Presentazione Accademy .NET Core C#

✓ About Trainer Spalluzzi Francesco

Sviluppatore C#

Certificato Azure Fundamentals

Azure Data Fundamentals

Microsoft Certified Trainer

Presentazione Accademy .NET Core C#

✓ About Course .Net Core C#

.NET – A unified platform



Un framework ed un unico Runtime per agevolare gli sviluppatori nell'implementazione di soluzioni informatiche utilizzando la piattaforma .NET 5.0

Soluzioni informatiche che saranno cross-platform

Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Prima settimana

Day 1
Lun

Introduction to C#

.Net Framework

What is C#=?

IDE VISUAL STUDIO 2019

IDE VISUAL STUDIO 2019
environment

Console Application

Web Application

Dynamic link libraries

Development – Installation of

Day 2
Mar

Basic C# Syntax

Basic C# Console Applications Structure

Variables

Expressions

Boolean Logic

Branching

Looping

Type Conversion

Complex variable types

String Manipulation



ACADEMY

.NET CORE / C#

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Prima settimana

Day 3 Merc

Module 6 Functions (8 ore)

Defining and using functions

Variable Scope

The Main() Function

Struct Functions

Overloading Functions

Using Delegates

Debugging and Error Handling

Day 4 Giov

Module 7 Introduction to Object-Oriented Programming (8 ore)

Whats is Object-Oriented Programming?

OOP Techniques

Library Applications/

Defining Classes

Class definitions in C#

System.Object

Constructors and Destructors

OOP Tools in Visual Studio

Interfaces versus Abstract classes

Struct types

Shallow Copying versus Deep Copying

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Prima settimana

Day 5 Ven

✓ **Module 8 Defining class members**

Members definitions

Additional Class member topics

Interface implementation

Partial class definitions

Partial method definitions

Example application

The call hierarchy Window

Module 9 Collections, Comparisons, and Conversions (5 ore)

Collections

Comparisons

Conversions

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Seconda settimana

Module 10 Additional C# Techniques (8 ore)

The :: operator and the global namespace qualifier

Custom exceptions

Events

Attributes

Initializers

Type inference

Anonymous types

Dynamic lookup

Advanced Method parameters

Lambda expressions

Day 6 Lun

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Seconda Settimana

Day 7 Mar

Module 11 Cloud and cross-platform Programing (3 ORE)

The Cloud, Cloud Computing Cloud optimized stack

Cloud patterns and best practices

Microsoft Azure C# Libraries to create a storage container

Creating an ASP.Net 4.7 Web site that users the storage container

Creating an ASP.Net Web Api

Deployment and consuming an ASP.Net Web Api on Microsoft Azure

Scaling an ASP.Net Web API on Microsoft Azure



ACADEMY

.NET CORE / C#

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Seconda Settimana

Day 8 Merc

Module .NET Standard and .NET Core (8 ore)

Cross platform basics and Key “Must Know” terms

What is .Net Standard and Why is it Needed?

Referencing and targeting frameworks

Building and packaging a .net standard library

Building a .net core application with Visual Studio

Day 9 Giov

Module ASP.Net and ASP.NET Core (8 ore)

Overview of Web Applications

Which Asp.Net to use and why

Using ASP.Net Web Model Views Controller

Creating ASP.Net Core Web Applications

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Seconda Settimana

Day 10 Ven

Module Data Access (8 ore)

Files

File Classes for Input and Output

Streams

Monitoring the File System

Language Integrated Query

JSON Basics

Databases

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Terza Settimana

Day 11 (Lun)

Module Data Access Exercitations (8 ore)

Day 12 Mar

Module ASP.Net and Asp.Net Core Exercitations (8 ore)

Day 13 Merc

**Module Cloud and cross-platform Programing
Exercitations (8 ore)**

Day 14 Giov

Module Generics (3 ore)

What are generics?

Using Generics

Defining Generic Type

Variance

Module Generics Exercitations (5 ore)

Day 15 Ven

**Module DateTime Methods Example and
Exercitations (3 ore)**

Module Delegate Exercitations (3 ore)

Module Nullable Types (2 ore)

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Quarta Settimana



Day 16 (Lun)

**Module Design Patterns implemented with ASP.Net Core
Web Api Exercitations (8 ore)**

Day 19 (Giov)

Module Exercitations with Entity Framework Core (8 ore)

Day 17 (Mar)

Module Design Patterns in C#

Day 18 (Merc)

Module Async await Task and Thread Examples (8 ore)

Day 20 (Ven)

Module Common String Operations (8 ore)



ACADEMY

.NET CORE / C#

■ Presentazione Accademy .NET Core C#

✓ About Course .Net Core C# - Quinta Settimana

Day 21 (Lun)

**Module Extentions Method in C# and Anonymous Types
(8 ore)**

Day 22 (Mar)

**Module Asp.Net Core Authentication and Performing
HTTP Request (8 ore)**

Day 23 (Merc)

**Data Annotations e Entity Framework Core Examples (8
ore)**

Day 24 (Giov)

**Module Custom Validations with ASP.Net Core App (8
ore)**

Day 25

Final Exercitation (8 ore)

ACADEMY

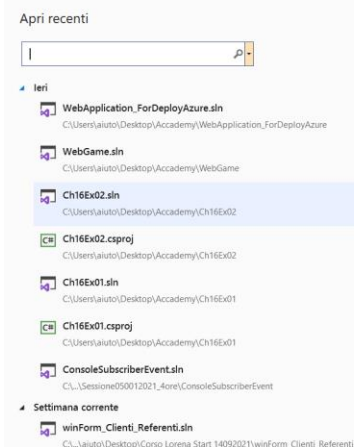
.NET CORE / C#



■ Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

Visual Studio 2019



Visual Studio 2019 – Integrated Development Environment – Ambiente di sviluppo per implementare applicazioni :
.NET FULL FRAMEWORK (Windows)
Cross-platform per Windows – Linux – Mac con .NET Core (Console – Asp.Net Core – Web Api Asp.Net Core – Blazor App)



Visual Studio Code per Linux – Ambiente per sviluppare **direttamente applicazioni .Net Core** con una distro di Ubuntu ad esempio.



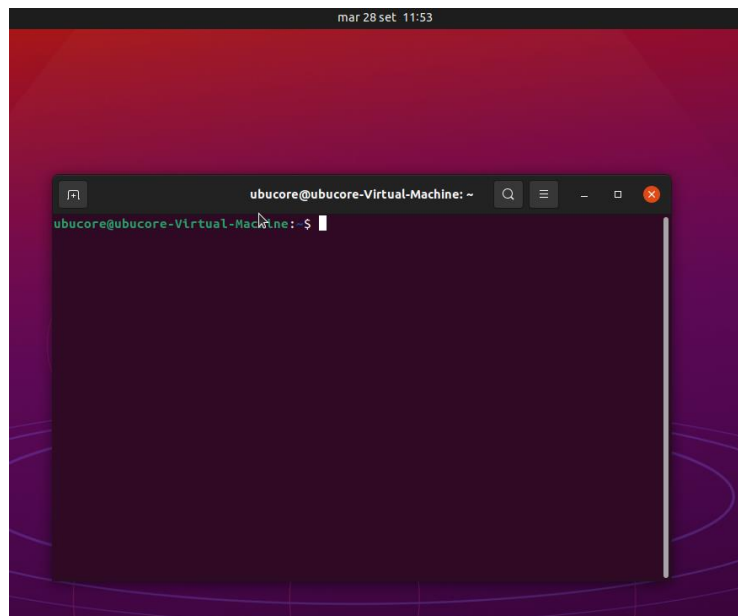
ACADEMY
.NET CORE / C#



Visual Studio for Mac
 Ambiente di sviluppo per Implementare soluzioni
.Net Core con Mac-OS

Presentazione Accademy .NET Core C#

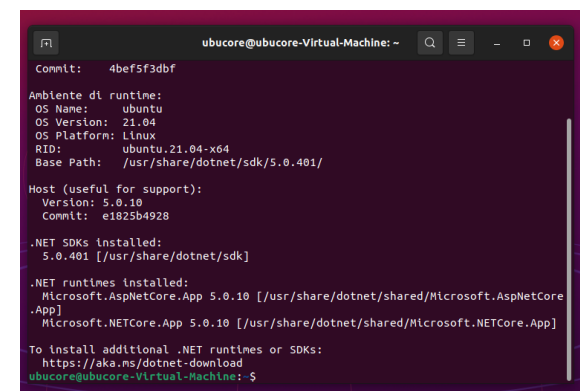
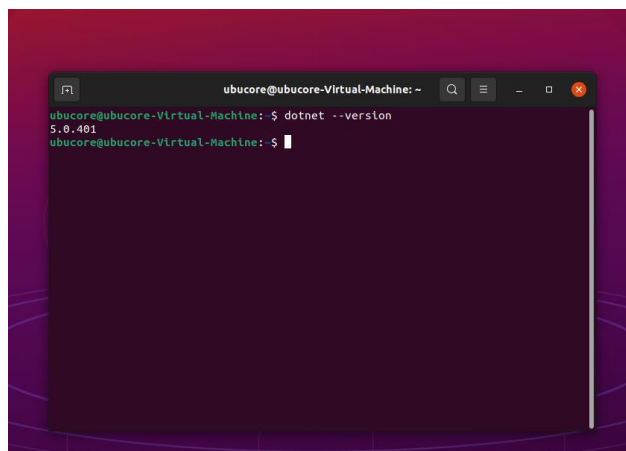
✓ About Platform and IDE (Integrated Development Environment)



Per coloro i quali volessero 'cimentarsi' nello sviluppo di applicazioni C# con .NET Core con una distro Ubuntu, potrebbero partire da questo link:

[How to Install Dotnet Core on Ubuntu 20.04 – TecAdmin](#)

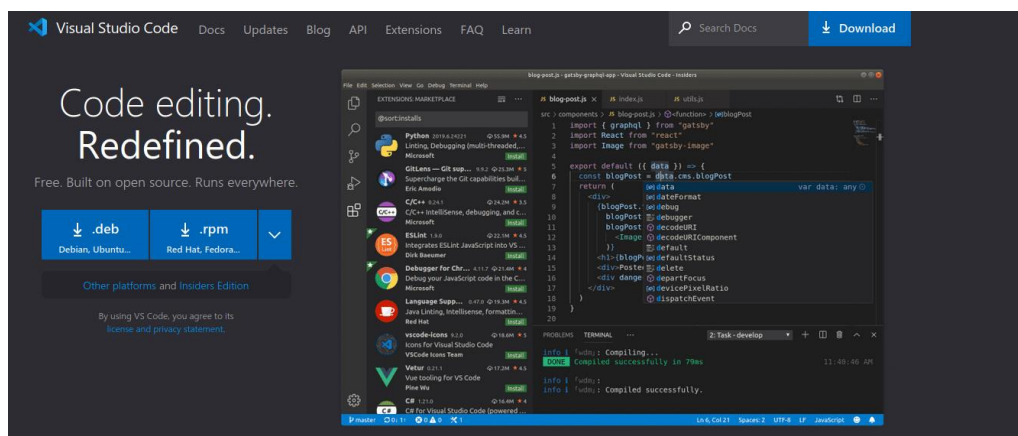
Basterà digitare da una finestra di terminale `dotnet --version` – Significa che l'installazione del framework .NET Core è andato a buon fine.



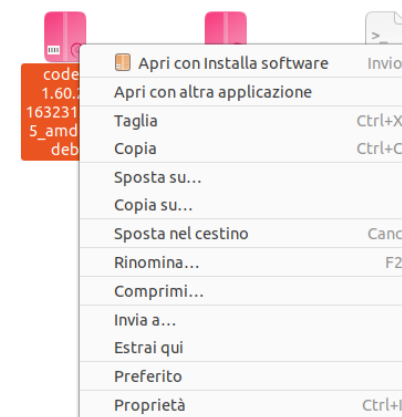
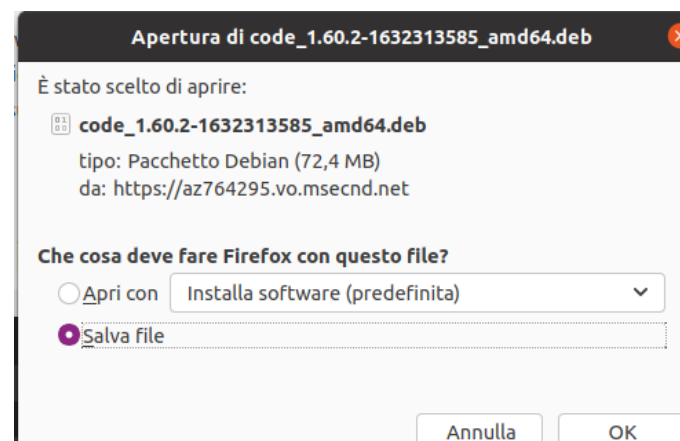
Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

Per continuare il tutorial della slide precedente, **bisognerà installare dal sito Microsoft Visual Studio Code per Linux** Aprendo una scheda del browser di Linux e scrivendo nella barra degli indirizzi <https://code.visualstudio.com> Troverete tutte le istruzioni dettagliate per installare questo ambiente di scripting all'interno della distro corrente di Linux



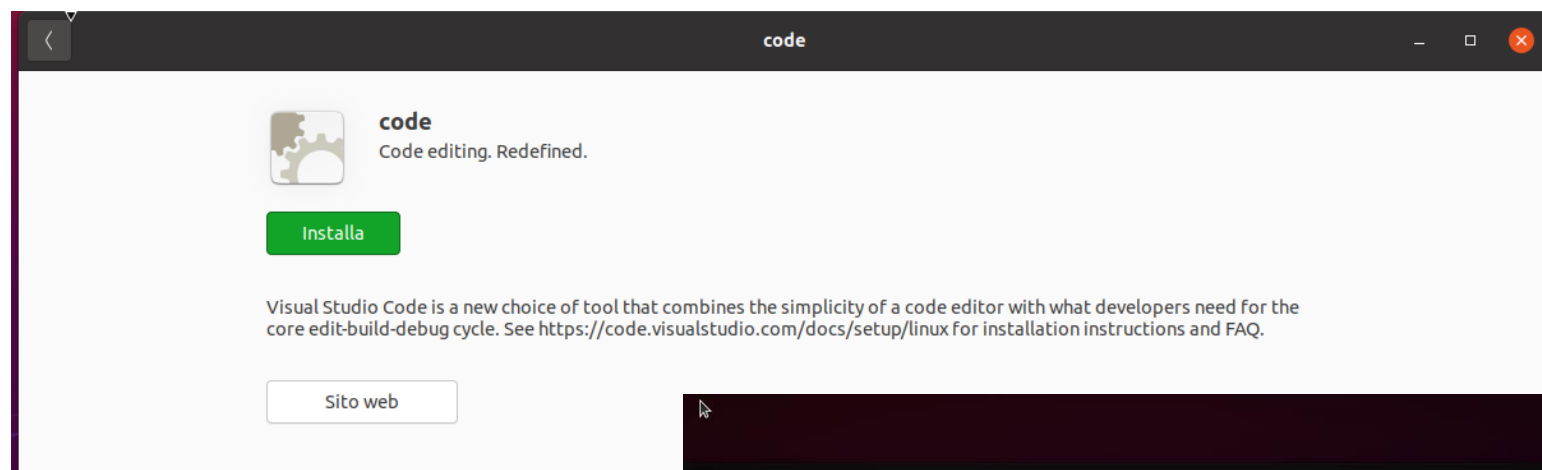
Cliccando sul pulsante .deb partirà il download del setup per Ubuntu, come si nota da questa finestra di dialogo



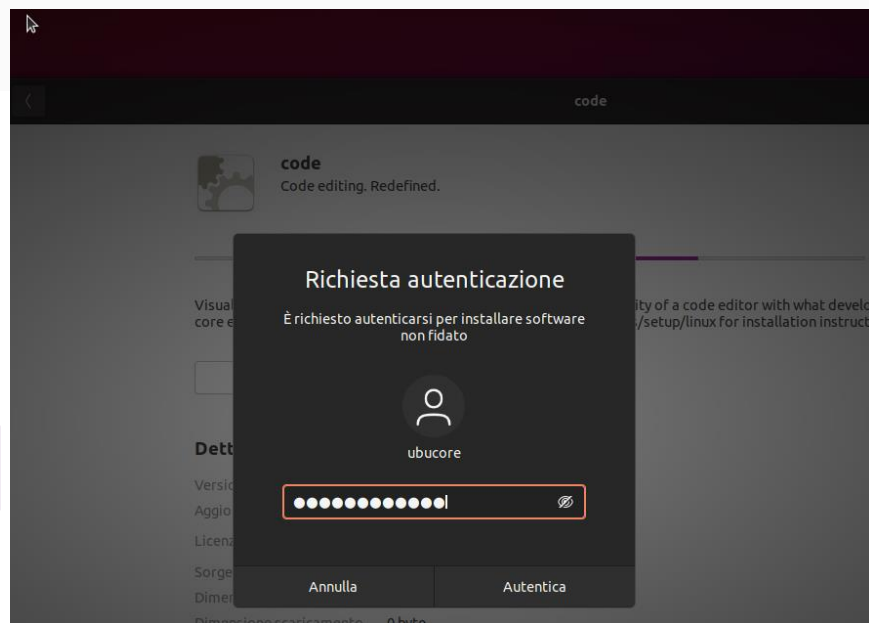
ACADEMY
.NET CORE / C#

Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)



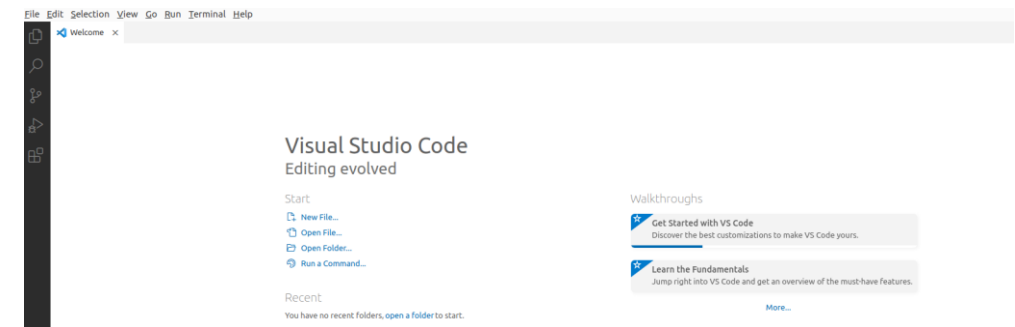
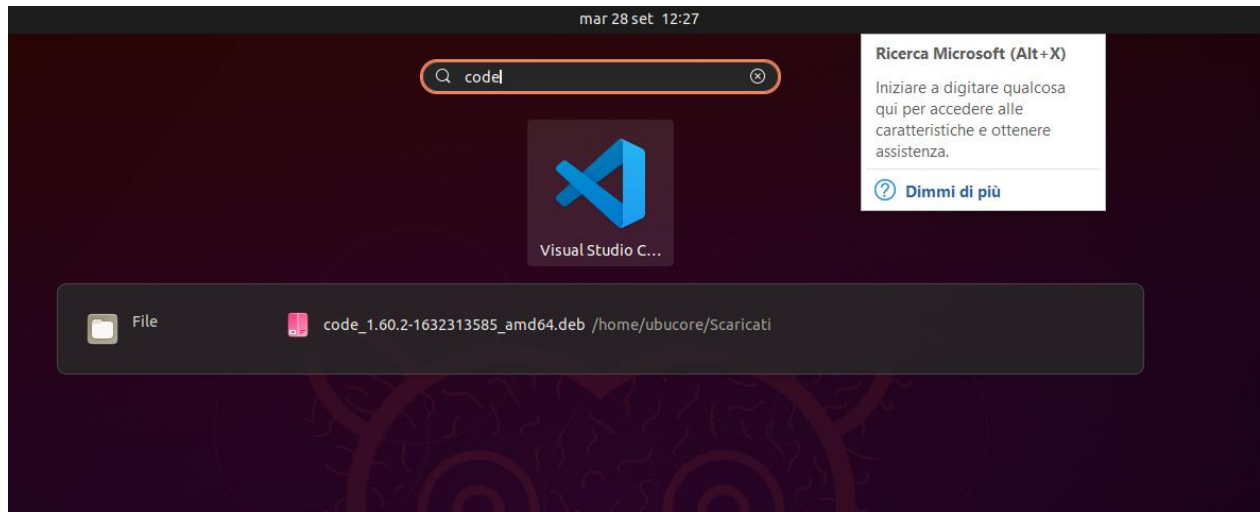
Si provvede a installare Visual Studio Code cliccando sul pulsante **Installa** e poi da **terminale si può digitare code**. Durante l'installazione vi verrà anche una finestra di autenticazione. Bisognerà scrivere la password definita in fase di installazione della distro di Ubuntu.



ACADEMY
.NET CORE / C#

Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)



Una volta installato Visual Studio Code va configurato per gestire progetti in C#. Ma questo aspetto lo vedremo in dettaglio durante le sessioni del percorso formativo

Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
ubucore@ubucore-Virtual-Machine:~$ █

```

Creiamo una cartella dal nome **DemoChsarpCore** (mkdir DemoCsharpCore)

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ dotnet new console

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ dotnet new console
Creazione del modello "Console Application" completata.

Elaborazione post-creazione delle azioni in corso...
Esecuzione di 'dotnet restore ' in /home/ubucore/DemoCsharpCore/DemoCsharpCore.csproj...
Individuazione dei progetti da ripristinare...
/home/ubucore/DemoCsharpCore/DemoCsharpCore.csproj ripristinato (in 67 ms).
Il ripristino è riuscito.

ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ █

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ dotnet new console
:reazione del modello "Console Application" completata.

Elaborazione post-creazione delle azioni in corso...
Esecuzione di 'dotnet restore ' in /home/ubucore/DemoCsharpCore/DemoCsharpCore.csproj...
Individuazione dei progetti da ripristinare...
/home/ubucore/DemoCsharpCore/DemoCsharpCore.csproj ripristinato (in 67 ms).
Il ripristino è riuscito.

ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ ls
DemoCsharpCore.csproj  obj  Program.cs
ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ dotnet restore
Individuazione dei progetti da ripristinare...
Tutti i progetti sono aggiornati per il ripristino.
ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ dotnet run
Hello World!
ubucore@ubucore-Virtual-Machine:~/DemoCsharpCore$ █

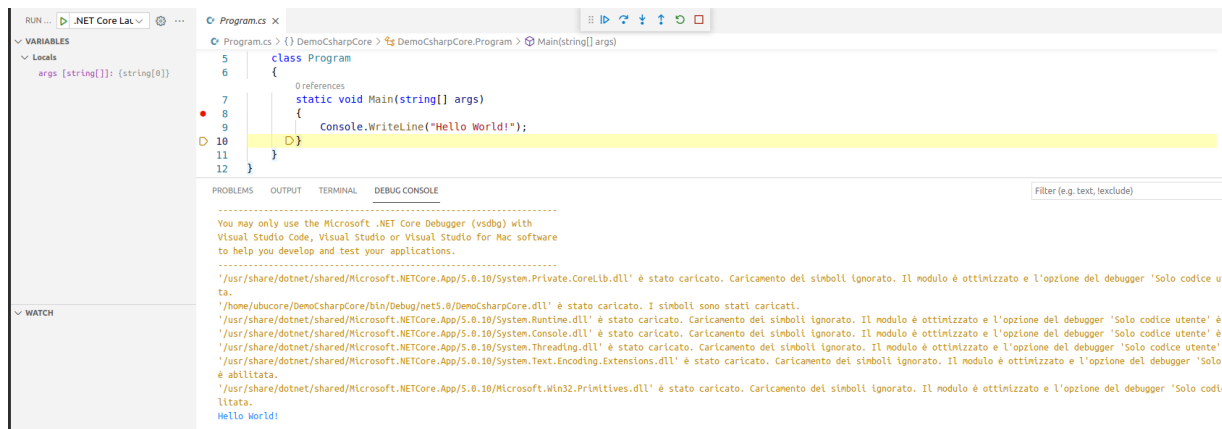
```

ACADEMY

.NET CORE / C#

Presentazione Accademy .NET Core C#

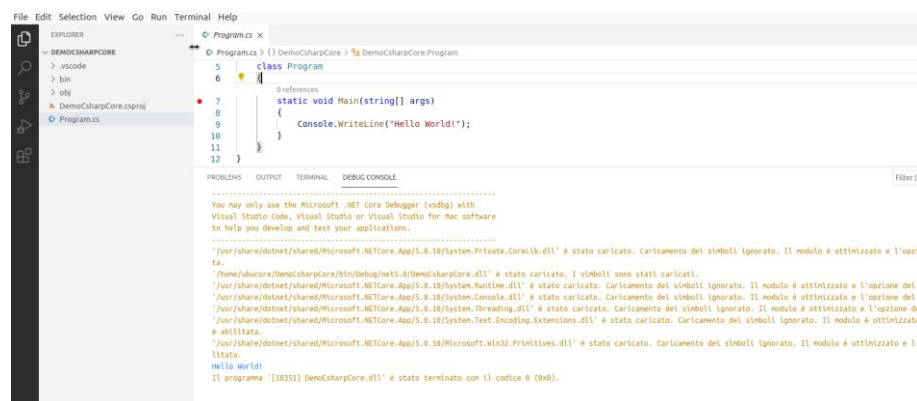
✓ About Platform and IDE (Integrated Development Environment)



Dalla finestra di Explorer aprire il progetto creato nella cartella DemoCsharpCore ed eseguire la prova di Debug così si installa in automatico

Il plugin per gestire progetti

In C#. In questo modo si attiva la gestione degli errori dell'applicazione.



ACADEMY
.NET CORE / C#

Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

SQL Server on Linux

11/04/2019 • 3 minutes to read • 

Applies to:  SQL Server (all supported versions) - Linux

SQL Server 2019 runs on Linux. It's the same SQL Server database engine, with many similar features and services regardless of your operating system. To find out more about this release, see [What's new in SQL Server 2019 for Linux](#).

Install

To get started, install SQL Server on Linux using one of the following quickstarts:

- [Install on Red Hat Enterprise Linux](#)
- [Install on SUSE Linux Enterprise Server](#)
- [Install on Ubuntu](#)
- [Run on Docker](#)
- [Provision a SQL VM in Azure](#)

Per progettare applicazioni database oriented cross-platform con .NET Core bisognerà installare SQL Server on Linux

<https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview>

Una volta installata e configurata correttamente una istanza di SQL Server bisognerà integrare in Visual Studio Code l'accesso a tale istanza per gestire la base dati

[Use the Visual Studio Code mssql extension - SQL Server | Microsoft Docs](#)

ACADEMY

.NET CORE / C#



Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

```
ubucore@ubucore-Virtual-Machine: ~  
ubucore@ubucore-Virtual-Machine:~$ wget -qO- https://packages.microsoft.com/keys  
/microsoft.asc | sudo apt-key add -  
[sudo] password di ubucore:  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (s  
ee apt-key(8)).  
OK  
ubucore@ubucore-Virtual-Machine:~$ sudo add-apt-repository "$(wget -qO- https://  
packages.microsoft.com/config/ubuntu/20.04/mssql-server-2019.list)"  
Repository: 'deb [arch=amd64,armhf,arm64] https://packages.microsoft.com/ubuntu/  
20.04/mssql-server-2019 focal main'  
Description:  
Archive for codename: focal components: main  
More info: https://packages.microsoft.com/ubuntu/20.04/mssql-server-2019  
Adding repository.  
Press [ENTER] to continue or Ctrl-c to cancel.  
Adding deb entry to /etc/apt/sources.list.d/archive_uri-https_packages_microsoft  
_com_ubuntu_20_04_mssql-server-2019-hirsute.list  
Adding disabled deb-src entry to /etc/apt/sources.list.d/archive_uri-https_packa  
ges_microsoft_com_ubuntu_20_04_mssql-server-2019-hirsute.list  
Scaricamento di:1 http://security.ubuntu.com/ubuntu hirsute-security InRelease [  
110 kB]  
Trovato:2 http://it.archive.ubuntu.com/ubuntu hirsute InRelease
```

Si lanciano da terminale i comandi che sono rappresentati in questa finestra.

Si finalizza l'installazione con questi ultimi comandi, sempre da terminale

3. Run the following commands to install SQL Server:

```
Bash Copy  
sudo apt-get update  
sudo apt-get install -y mssql-server
```

4. After the package installation finishes, run **mssql-conf setup** and follow the prompts to set the SA password and choose your edition.

```
Bash Copy  
sudo /opt/mssql/bin/mssql-conf setup
```

Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

```

ubucore@ubucore-Virtual-Machine: ~
-----+-----
Eseguire 'sudo /opt/mssql/bin/mssql-conf setup'
per completare l'installazione di Microsoft SQL Server
-----+-----

Elaborazione dei trigger per man-db (2.9.4-2)...
Elaborazione dei trigger per libc-bin (2.33-0ubuntu5)...
ubucore@ubucore-Virtual-Machine:~$ sudo /opt/mssql/bin/mssql-conf setup
usermod: nessuna modifica
Scegliere un'edizione di SQL Server:
 1) Evaluation (gratuita, nessun diritto sull'utilizzo in ambito di produzione,
limite di 180 giorni)
 2) Developer (gratuita, nessun diritto sull'utilizzo in ambito di produzione)
 3) Express (disponibile)
 4) Web (A PAGAMENTO)
 5) Standard (A PAGAMENTO)
 6) Enterprise (A PAGAMENTO) - Utilizzo di core CPU limitato a 20 core fisici/4
con hyperthreading
 7) Enterprise Core (A PAGAMENTO) - Utilizzo di core CPU fino al massimo del si
stema operativo
 8) Ho acquistato una licenza tramite un canale di vendita al dettaglio e ho un
codice Product Key per l'accesso.

```

Si provvede all'installazione di SQL Server scegliendo con un numero compreso tra 1-8 l'edizione. Noi sceglieremo quella Express e indicheremo il numero tre

Dopo l'installazione si controllerà lo stato del servizio

```

ubucore@ubucore-Virtual-Machine: ~
ubucore@ubucore-Virtual-Machine:~$ systemctl status mssql-server --no-pager
● mssql-server.service - Microsoft SQL Server Database Engine
   Loaded: loaded (/lib/systemd/system/mssql-server.service; enabled; vendor p
  reset: enabled)
   Active: active (running) since Tue 2021-09-28 14:51:19 CEST; 52s ago
     Docs: https://docs.microsoft.com/en-us/sql/linux
    Main PID: 9942 (sqlservr)
       Tasks: 123
      Memory: 586.9M
      CGroup: /system.slice/mssql-server.service
              └─9942 /opt/mssql/bin/sqlservr
                  └─9967 /opt/mssql/bin/sqlservr

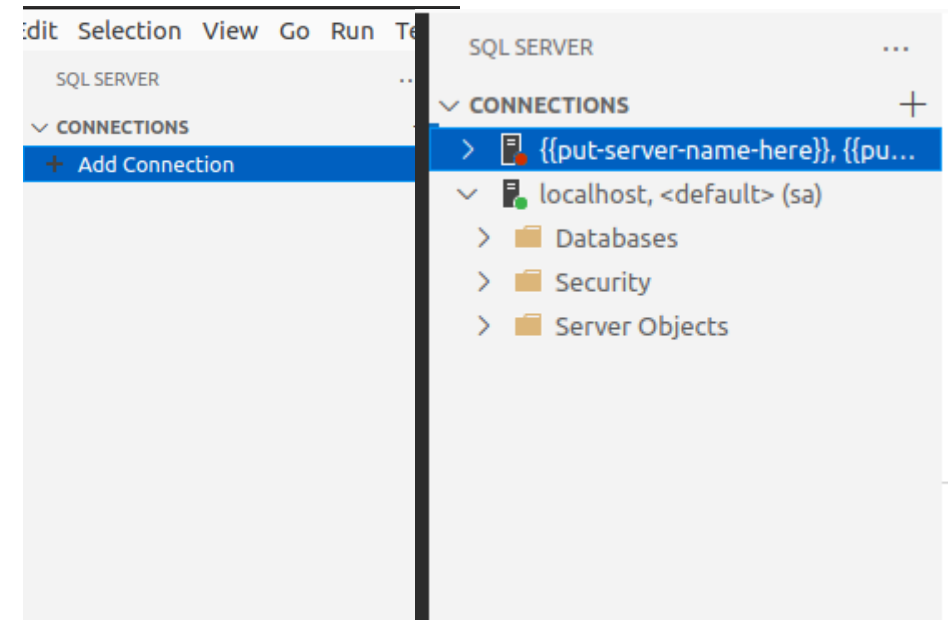
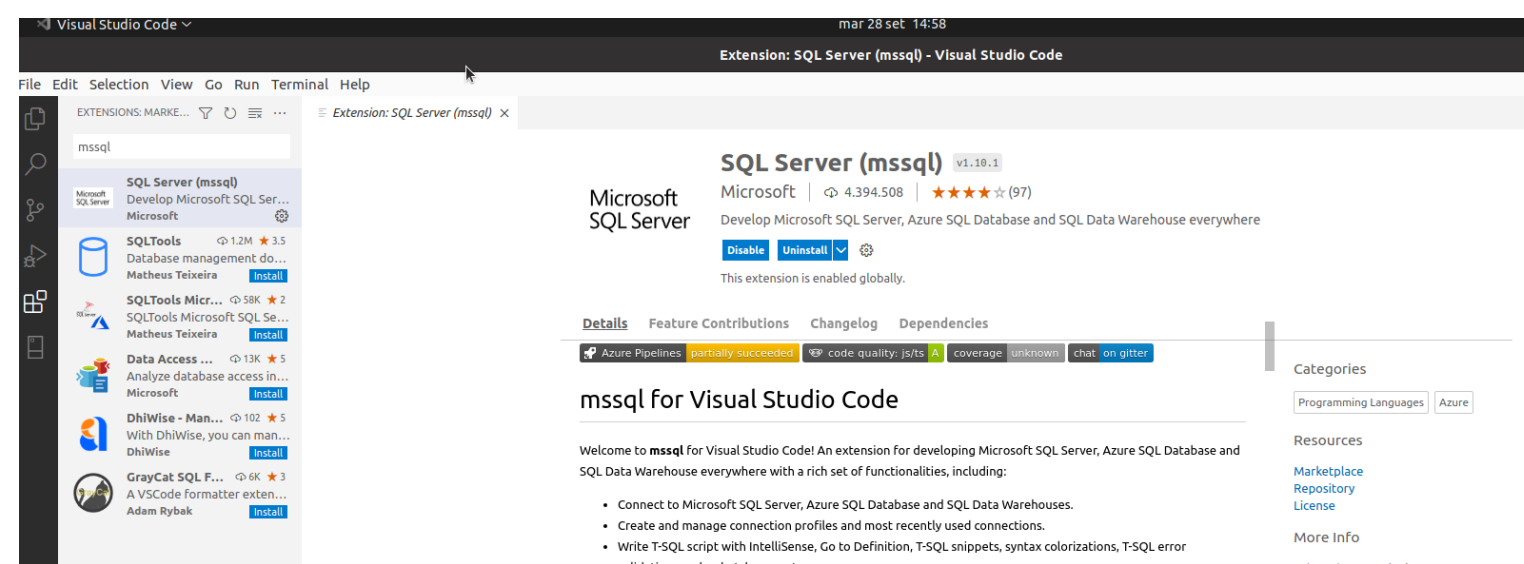
set 28 14:51:22 ubucore-Virtual-Machine sqlservr[9967]: [318B blob data]
set 28 14:51:22 ubucore-Virtual-Machine sqlservr[9967]: [78B blob data]
set 28 14:51:22 ubucore-Virtual-Machine sqlservr[9967]: [84B blob data]
set 28 14:51:22 ubucore-Virtual-Machine sqlservr[9967]: [145B blob data]
set 28 14:51:23 ubucore-Virtual-Machine sqlservr[9967]: [96B blob data]
set 28 14:51:23 ubucore-Virtual-Machine sqlservr[9967]: [66B blob data]
set 28 14:51:23 ubucore-Virtual-Machine sqlservr[9967]: [96B blob data]
set 28 14:51:23 ubucore-Virtual-Machine sqlservr[9967]: [100B blob data]
set 28 14:51:23 ubucore-Virtual-Machine sqlservr[9967]: [71B blob data]
set 28 14:51:23 ubucore-Virtual-Machine sqlservr[9967]: [124B blob data]
ubucore@ubucore-Virtual-Machine:~$

```

Presentazione Accademy .NET Core C#

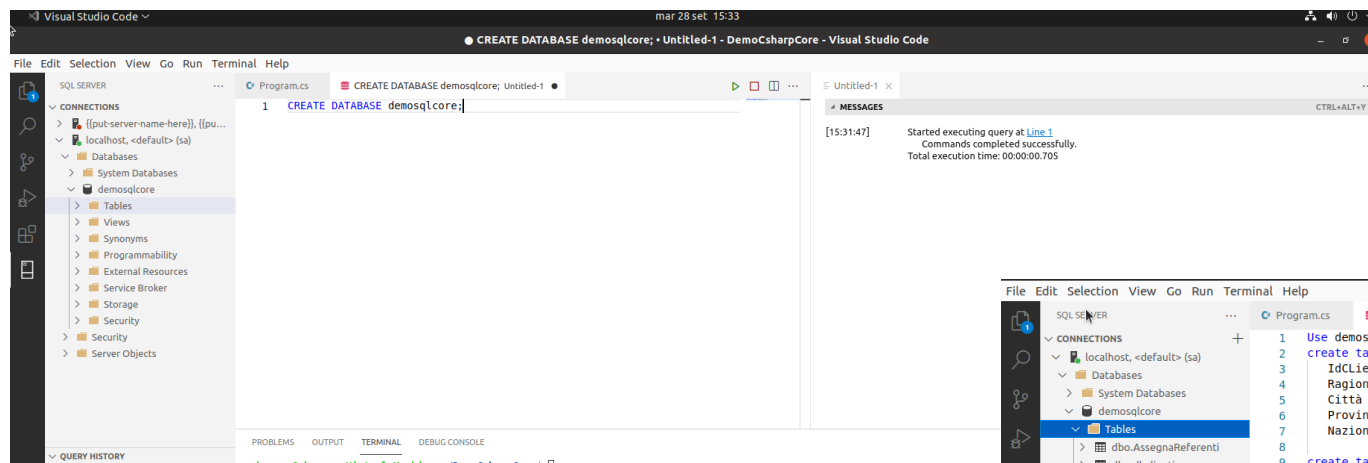
✓ About Platform and IDE (Integrated Development Environment)

Si integra la funzionalità in Visual Studio Code di esplorazione database su una Istanza di SQL Server installata su Linux
Si clicca su AddConnection per creare un Profilo di connessione

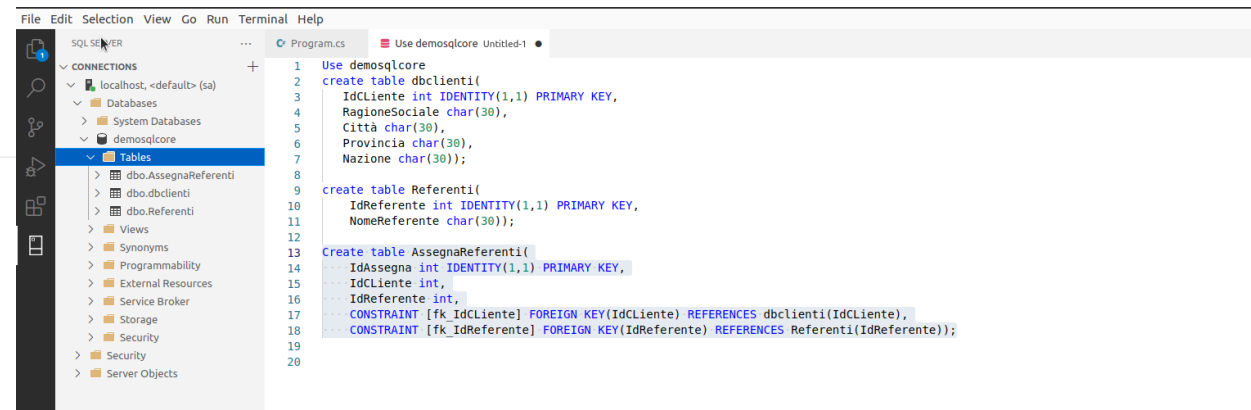


Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)



Con lo strumento di Command Palette ci sono tutti una serie di comandi che si possono eseguire tramite attività di scripting sull'istanza di SQL configurata



ACADEMY
.NET CORE / C#

In progetti .NET Core Solution database oriented

 System.Data.SqlClient 4.8.3 

Requires NuGet 2.12 or higher.

Package Manager  PackageReference Paket CLI Script & Interactive Cake

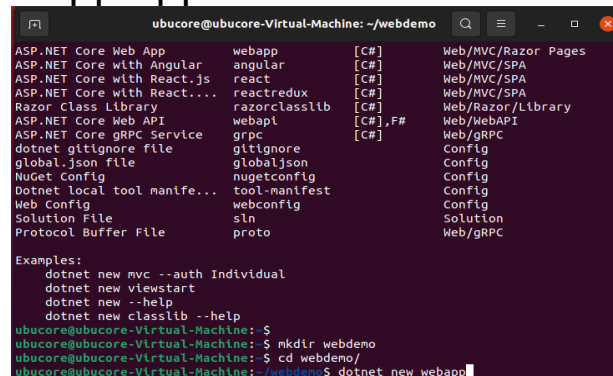
> dotnet add package System.Data.SqlClient --version 4.8.3

■ Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)

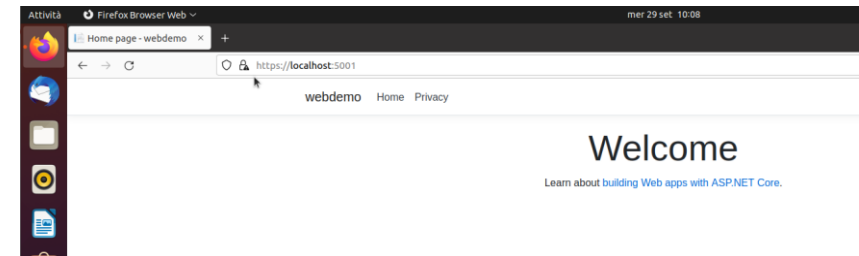
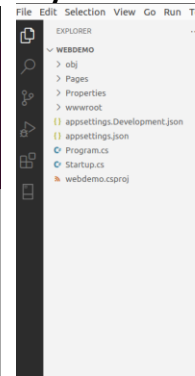
Con l'architettura .NET Core è possibile quale che sia l'architettura del Sistema Operativo (Windows, Linux, Mac OS) di creare tre tipologie di applicazioni:

- Applicazioni Web con l'architettura di ASP.Net Core [ASP.NET Core – Wikipedia](#) disegnate architetturalmente con il paradigma MVC (Model View Controller)
- Web API RestFull o volgarmente denominati «Web Services» [ASP.NET Web APIs | Rest APIs with .NET and C# \(microsoft.com\)](#)
- Da alcune versioni di .NET Core è possibile creare web apps con C# and HTML con un framework denominato Blazor (Blazor Server App oppure Blazor WebAssembly o denominate Single Page apps) ([Blazor – Wikipedia](#))



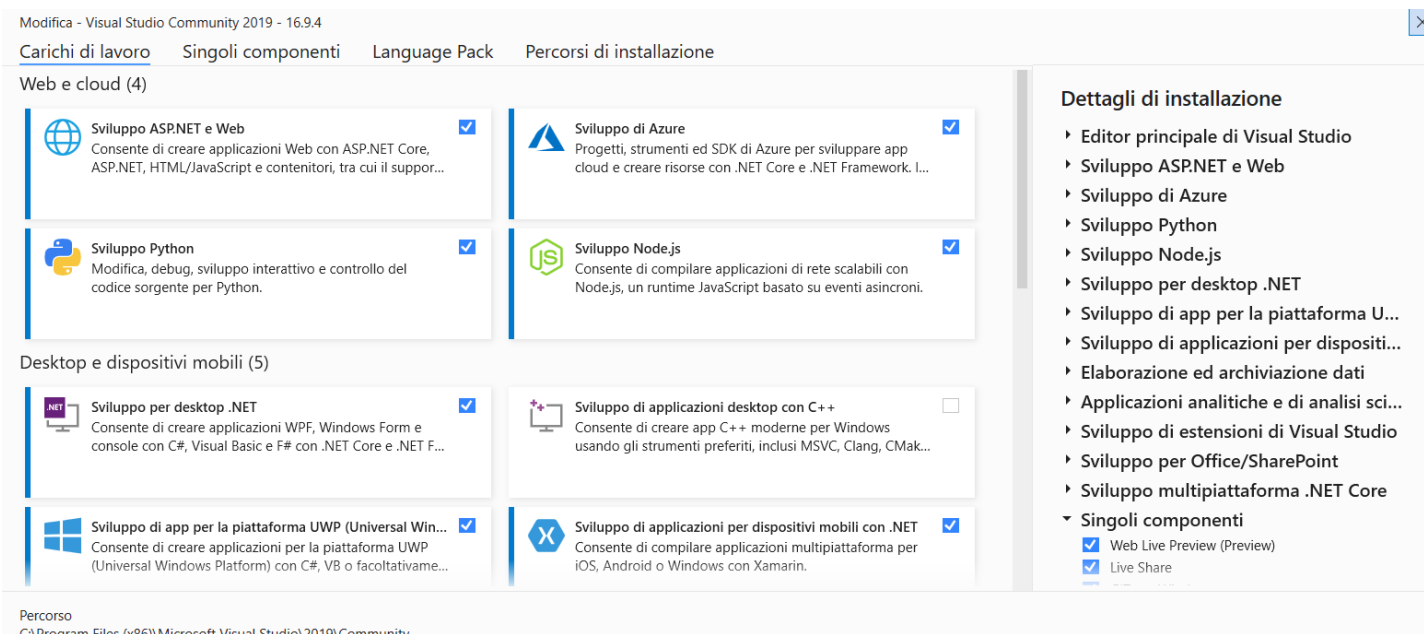
```
ubucore@ubucore-Virtual-Machine: ~/webdemo
ASP.NET Core Web App      webapp      [C#]      Web/MVC/Razor Pages
ASP.NET Core with Angular angular     [C#]      Web/MVC/SPA
ASP.NET Core with React.js react       [C#]      Web/MVC/SPA
ASP.NET Core with React... reactredux  [C#]      Web/MVC/SPA
Razor Class Library       razorclasslib [C#]      Web/Razor/Library
ASP.NET Core Web API      webapi      [C#],F#    Web/WebAPI
ASP.NET Core gRPC Service grpc        [C#]      Web/gRPC
dotnet global tool file   gitignore   Config
global.json file         nugetconfig Config
NuGet Config              tool-manifest Config
Dotnet local tool manife... webconfig   Config
Web Config                sln         Solution
Protocol Buffer File       proto       Web/gRPC

Examples:
dotnet new mvc --auth Individual
dotnet new viewstart
dotnet new --help
dotnet new classlib --help
ubucore@ubucore-Virtual-Machine: $
ubucore@ubucore-Virtual-Machine: $ mkdir webdemo
ubucore@ubucore-Virtual-Machine: $ cd webdemo/
ubucore@ubucore-Virtual-Machine: ~/webdemo $ dotnet new webapp
```



Presentazione Accademy .NET Core C#

✓ About Platform and IDE (Integrated Development Environment)



In ambiente Windows, così si presenta
Il programma di manutenzione della
Versione di Visual Studio 2019 Community
con i workloads necessari per consentirvi
di seguire tutte le tematiche del corso.

**Come abbiamo visto nelle precedenti slides
l'ambiente di sviluppo per Ubuntu è deno-
minato Visual Studio Code**

**In ambiente Windows, si può installare anche
SQL Server (sqlLocalDB per la gestione di
Istanze). Questo tools lo troviamo integrato
nell'istallazione della versione community**

ACADEMY
.NET CORE / C#

■ Day 1

- Module 1 C# Language (3 hours)
- Modulo 2 Writing C# Program (5 hours)



Day 1

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

C# è un linguaggio di programmazione utilizzato all'interno del framework .NET Core denominato .NET 5 attualmente per sviluppare svariate tipologie di applicazioni:

- ☐ **Console Application;**
- ☐ **Class Library** (o Librerie a collegamento dinamico o dll)
- ☐ **Web Application** con architettura Model View Controller incapsulando anche framework fronte-end tipo React o Angular.
- ☐ **Web Application con architettura Model View Controller** per sviluppare Web Api RESTFull (o volgarmente denominati servizi web)
- ☐ **Applicazioni web con framework denominato Blazor** (<https://en.wikipedia.org/wiki/Blazor>)

Da terminale dotnet new senza aggiungere altro si ottengono tutte le tipologie di applicazioni realizzabili con .Net Core e il linguaggio C#

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

```

Amministratore: Prompt dei comandi
.NET SDK (che rispecchia un qualsiasi file global.json):
Version: 5.0.202
Commit: db7cc87d51

Ambiente di runtime:
OS Name: Windows
OS Version: 10.0.19042
OS Platform: Windows
RID: win10-x64
Base Path: C:\Program Files\dotnet\sdk\5.0.202\

Host (useful for support):
Version: 5.0.5
Commit: 2f740adc14

.NET SDKs installed:
5.0.101 [C:\Program Files\dotnet\sdk]
5.0.102 [C:\Program Files\dotnet\sdk]
5.0.202 [C:\Program Files\dotnet\sdk]

.NET runtimes installed:
Microsoft.AspNetCore.All 2.1.27 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.All]
Microsoft.AspNetCore.App 2.1.27 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 3.1.10 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 3.1.14 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 5.0.5 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.NETCore.App 2.1.27 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 3.1.14 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 5.0.5 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.WindowsDesktop.App 3.1.14 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]

```

La parola chiave dotnet –info ci offre utili informazioni circa il sistema operativo la versione l’architettura e la lista di .NET sdks e le versioni .Net runtimes installed

Sono utilissime queste informazioni per eseguire correttamente attività di troubleshooting al fine di risolvere problematiche relative ad applicazioni con tale architettura.

ACADEMY

.NET CORE / C#



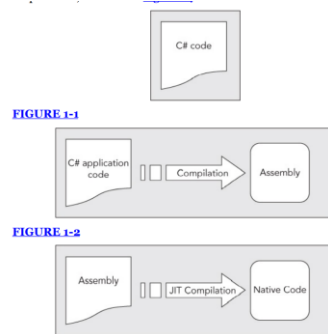
Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

C# risulta essere un compilatore che con il .NET Framework (attualmente con la versione 4.7) permette di realizzare applicazioni legate al sistema operativo (Windows per intenderci) con l'utilizzo di ambienti di sviluppo (Visual Studio oppure Visual Studio Code)

.NET Framework è basato su una serie di tipi basici (il tipo è un modo di rappresentare le informazioni) facilitando l'interoperabilità tra i diversi linguaggi che usano tale framework. **Questo modulo è denominato CTS (Common Type System).**

Un'applicazione una volta sviluppata e compilata correttamente con questo linguaggio c'è altro modulo denominato CLR (Common Language Runtime), modulo responsabile di gestire tutti gli aspetti legati all'esecuzione



➤ **Desktop applications-**

Windows Store applications

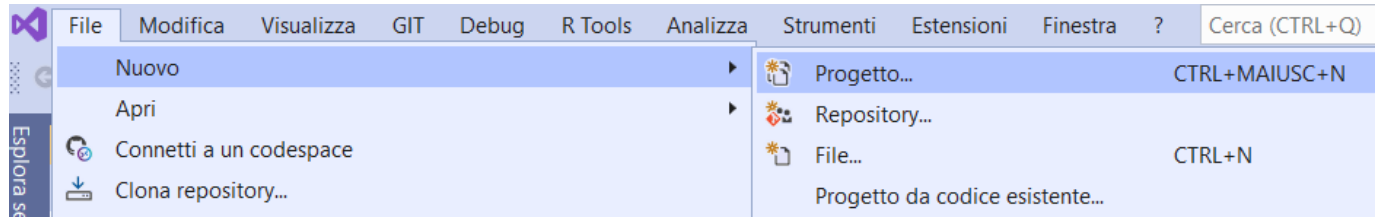
Cloud/Web applications

Web APIs

Any of these types might also require some form of database access, which can be achieved using the ADO.NET (Active Data Objects .NET) section of the .NET Framework, through the Entity Framework, or through the LINQ (Language Integrated Query) capabilities of C#. For .NET Core applications requiring database access you would use the Entity Framework Core library. Many other resources can be drawn on, such as tools for creating networking components, outputting graphics,

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program



Crea un nuovo progetto

Modelli di progetto recenti

- Applicazione Web ASP.NET (.NET Framework) C#
- App Windows Forms (.NET Framework) C#
- API Web ASP.NET Core C#
- App Web ASP.NET Core C#
- App console (.NET Framework) C#

Configura il nuovo progetto

App console (.NET Framework) C# Windows Console

Nome del progetto

ConsoleApplication

Percorso

C:\Users\aiuto\Desktop\Accademy\

Nome soluzione

ConsoleApplication

☐ Inserisci soluzione e progetto nella stessa directory

Framework

.NET Framework 4.7.1

```
Program.cs  x Attività pubblicazione sul Web
ConsoleApplication  ConsoleApplication.Program  Main(string[] args)
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApplication
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("AESYS CONSOLE APPLICATION");
14             Console.ReadKey();
15         }
16     }
17 }
18
```

C:\Users\aiuto\Desktop\Accademy\ConsoleApplication\ConsoleApplication\bin\Debug\ConsoleApplication.exe

AESYS CONSOLE APPLICATION

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

La Console Application ha una struttura molto basica e permette di implementare utility che eseguono interazioni o operazioni batch in modalità text-mode e senza interfaccia grafica.

Tecnicamente ha un entry point con **metodo static void** e una firma ossia un'array di string che funge da command line parameters da gestire durante l'esecuzione. La firma dell'array di stringhe non può essere cambiato in alcun modo e così dicasi per nome del metodo ossia Main che non può chiamarsi Main1.

IL metodo **WriteLine** permette di scrivere all'interno della console

Il metodo **ReadLine** permette di acquisire qualsiasi informazione **e memorizzarla in variabili**

Il metodo **ReadKey** permette di interrompere temporaneamente l'esecuzione e continuerà solo dopo aver premuto qualsiasi tasto sulla tastiera.

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

```
Prompt dei comandi
Numero di serie del volume: 5C6A-7A1D

Directory di C:\Users\aiuto\Desktop\Accademy

01/10/2021 11:03 <DIR> .
01/10/2021 11:03 <DIR> ..
24/09/2021 14:50 4.081.600 AccademyC#NetCore.rar
01/10/2021 11:03 8.202.042 aesitys_PPT_net_Edition_1.pptx
23/09/2021 10:59 4.424.256 aesitys_PPT_net_pr1.pptx
29/09/2021 14:38 147.927 Appunti .NET Core vari ed eventuali.docx
28/09/2021 14:08 879.862 Azure Storage Account.docx
22/09/2021 14:59 15.982.108 Beginning C# 7 Programming With Visual Studio 2017.pdf
27/09/2021 14:22 <DIR> Ch16Ex01
27/09/2021 15:03 <DIR> Ch16Ex02
01/10/2021 10:49 <DIR> ConsoleApplication
22/09/2021 17:10 6.688.075 csharp-language.pdf
27/09/2021 10:02 16.824 Esercitazioni_Soluzioni.docx
03/12/2020 13:50 13.730.993 MANUALE_DOCS_MICROSOFT.pdf
28/09/2021 09:23 58.446 Programma Accademy C# .Net Core.docx
27/09/2021 10:01 <DIR> Soluzioni
27/09/2021 16:08 <DIR> WebApplication_ForDeployAzure
27/09/2021 15:17 <DIR> WebGame
10 File 54.212.133 byte
8 Directory 66.507.251.712 byte disponibili

C:\Users\aiuto\Desktop\Accademy>md HelloConsoleCore

C:\Users\aiuto\Desktop\Accademy>cd HelloConsoleCore

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>dotnet new console
```

```
Prompt dei comandi
28/09/2021 14:08 879.862 Azure Storage Account.docx
22/09/2021 14:59 15.982.108 Beginning C# 7 Programming With Visual Studio 2017.pdf
27/09/2021 14:22 <DIR> Ch16Ex01
27/09/2021 15:03 <DIR> Ch16Ex02
01/10/2021 10:49 <DIR> ConsoleApplication
22/09/2021 17:10 6.688.075 csharp-language.pdf
27/09/2021 10:02 16.824 Esercitazioni_Soluzioni.docx
03/12/2020 13:50 13.730.993 MANUALE_DOCS_MICROSOFT.pdf
28/09/2021 09:23 58.446 Programma Accademy C# .Net Core.docx
27/09/2021 10:01 <DIR> Soluzioni
27/09/2021 16:08 <DIR> WebApplication_ForDeployAzure
27/09/2021 15:17 <DIR> WebGame
10 File 54.212.133 byte
8 Directory 66.507.251.712 byte disponibili

C:\Users\aiuto\Desktop\Accademy>md HelloConsoleCore

C:\Users\aiuto\Desktop\Accademy>cd HelloConsoleCore

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore\HelloConsoleCore.csproj...
Individuazione dei progetti da ripristinare...
C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore\HelloConsoleCore.csproj ripristinato (in 67 ms).
Restore succeeded.

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>
```

```
Prompt dei comandi
27/09/2021 15:03 <DIR> Ch16Ex02
01/10/2021 10:49 <DIR> ConsoleApplication
22/09/2021 17:10 6.688.075 csharp-language.pdf
27/09/2021 10:02 16.824 Esercitazioni_Soluzioni.docx
03/12/2020 13:50 13.730.993 MANUALE_DOCS_MICROSOFT.pdf
28/09/2021 09:23 58.446 Programma Accademy C# .Net Core.docx
27/09/2021 10:01 <DIR> Soluzioni
27/09/2021 16:08 <DIR> WebApplication_ForDeployAzure
27/09/2021 15:17 <DIR> WebGame
10 File 54.212.133 byte
8 Directory 66.507.251.712 byte disponibili

C:\Users\aiuto\Desktop\Accademy>md HelloConsoleCore

C:\Users\aiuto\Desktop\Accademy>cd HelloConsoleCore

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore\HelloConsoleCore.csproj...
Individuazione dei progetti da ripristinare...
C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore\HelloConsoleCore.csproj ripristinato (in 67 ms).
Restore succeeded.

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>dotnet run
Hello World!

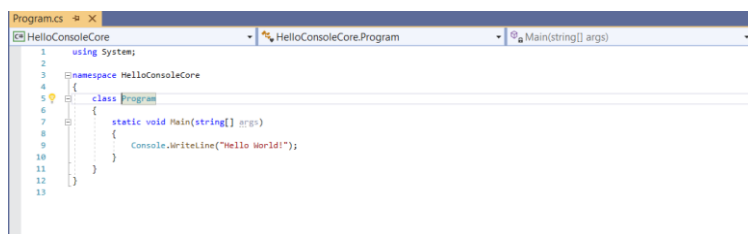
C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>
```

ACADEMY
.NET CORE / C#

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

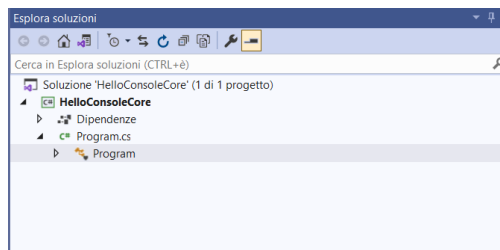
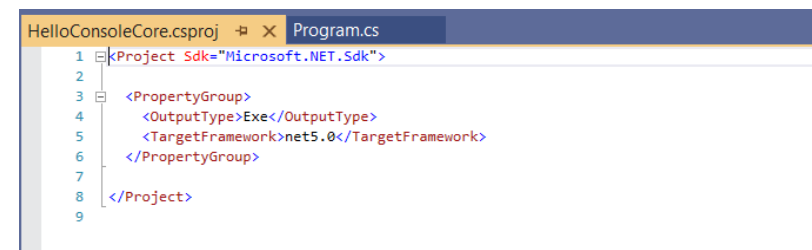
Nella slide precedente abbiamo rappresentato gli steps per creare con l'ausilio di .NET Core un'applicazione console utilizzando .NET Cli mode. Poi per poterla modificare e ricompilarla basterà avvalersi dell'ambiente Visual Studio oppure Visual Studio Code.



```

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

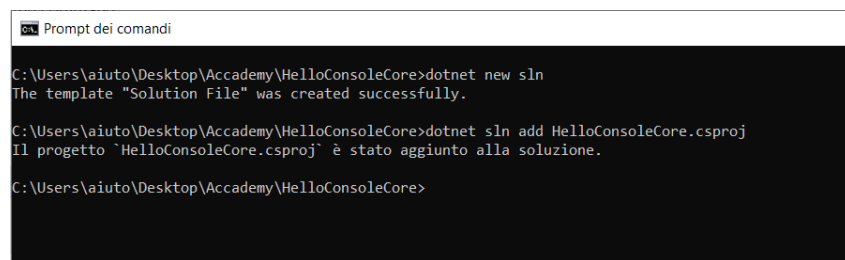
```

```

1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>net5.0</TargetFramework>
6   </PropertyGroup>
7
8 </Project>
9

```



```

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>dotnet new sln
The template "Solution File" was created successfully.

C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>dotnet sln add HelloConsoleCore.csproj
Il progetto 'HelloConsoleCore.csproj' è stato aggiunto alla soluzione.

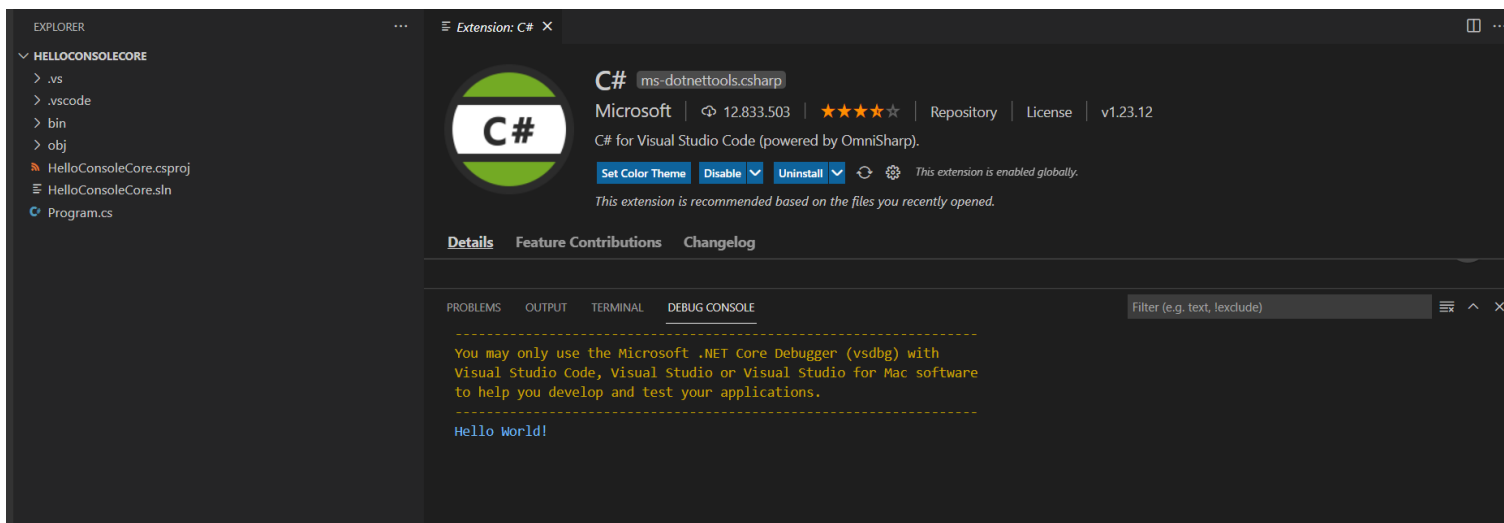
C:\Users\aiuto\Desktop\Accademy\HelloConsoleCore>

```

In questa immagine praticamente creiamo con il comando dotnet un file sln che porta il nome della cartella (nome solution). A questo file di solution aggiungiamo un riferimento al file csproj

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program



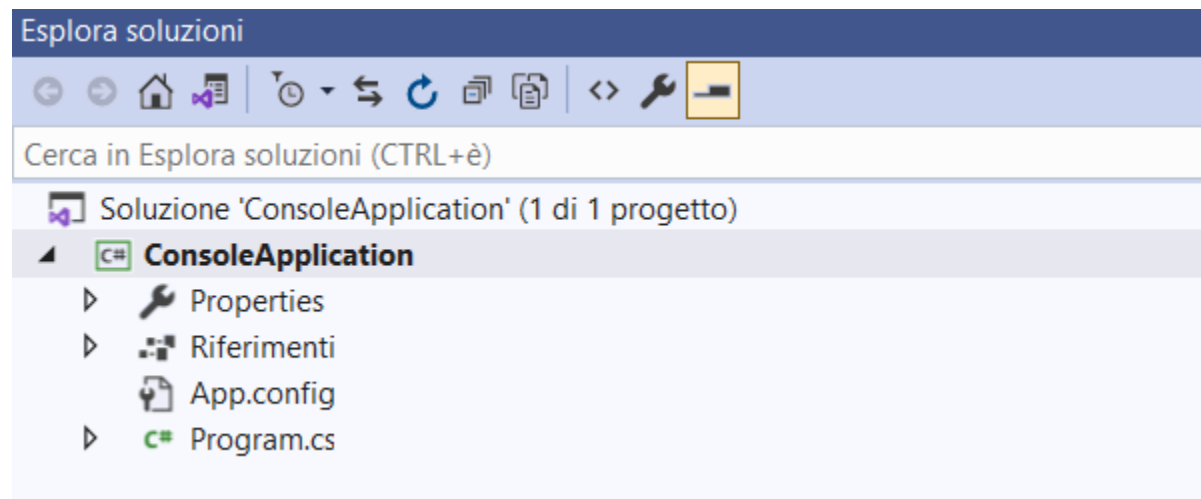
Dalla finestra di terminale di Windows e precisamente all'interno della cartella contenente la soluzione Core della console application digitando Code ed eseguendo l'applicazione senza Debug otteniamo in output nella finestra di terminale integrata Hello World

Dalle slides successive partiamo con le caratteristiche sintattiche di programmazione con questo linguaggio

ACADEMY
.NET CORE / C#

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program



Entriamo nel vivo della struttura sintattica di questa tipologia applicativa:

- C'è un file di configurazione **App.Config**
- **Riferimenti** (si aggiungono eventuali riferimenti ad Assembly mancanti del framework per particolari esigenze applicative)
- **Properties**
- **Program.cs** -> Tecnicamente è una classe che vediamo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("AESYS CONSOLE APPLICATION");
            Console.ReadKey();
        }
    }
}
```

- ❑ **Parte di dichiarazione dei riferimenti** con la parola riservata del linguaggio using
- ❑ **Suddivisione del codice sorgente in namespace**
- ❑ **Class Program**
- ❑ **Entry point statico metodo Void Main** con una commandLine array di stringhe

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

L'organizzazione all'interno dei metodi avviene con la parola riservata #region #end region

Selezionando il codice che si vuole commentare **basta selezionarlo e usare short keys CTRL+K,CTRL+C**

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApplication
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             // Method Body
14         }
15     }
16 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApplication
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             // Method Body
14             #region Method Body
15             Console.WriteLine("AESYS CONSOLE APPLICATION");
16             Console.ReadKey();
17             #endregion
18         }
19     }
20 }
21
```

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

Trattiamo circa la visibilità delle parti del codice (campi, proprietà, classi, metodi) e visibilità delle parti di programma tra Namespace in solution di Console Application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication
{
    class Program
    {
        public string valoreA { get; set; }
        public string valoreB { get; set; }

        static void Main(string[] args)
        {
            #region Method Body
            Console.WriteLine("AESYS CONSOLE APPLICATION");
            Console.ReadKey();
            #endregion
        }
    }
}
```

Regola 1 – Referenziare questa solution di Console Application in altro progetto avviene correttamente ma non sarà visibile la classe Program perché è privata. Se non è presente il modificatore di accesso public davanti a class la classe è privata

Regola 2 Le proprietà così dichiarate non potranno essere usate nel metodo void Main semplicemente perché il metodo è statico. Quando il metodo è dichiarato con il modificatore di accesso statico semplicemente si dice che è esso non è di istanza e quindi accessibile con il nome di Program.<nameOfVariable> oppure Program.<nameOfMethod> Quindi bisognerà aggiungere a tali proprietà il modificatore static prima del tipo della variabile.

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

Dalla sintesi di due regole di Visibilità a riguardo delle Console Application , ecco qui uno specchietto molto chiarificatore a riguardo di tutti i modificatori di accesso e una loro definizione esplicativa.

Modificatore di C#	Definizione
pubblico	Il tipo o il membro è accessibile da altro codice nello stesso assembly o in un altro assembly che vi fa riferimento.
privata	Il tipo o il membro è accessibile solo dal codice nella stessa classe.
protetto	Il tipo o il membro è accessibile solo dal codice nella stessa classe o in una classe derivata.
interno	Il tipo o il membro è accessibile dal codice nello stesso assembly ma non da un altro assembly.
protected internal	Il tipo o il membro è accessibile dal codice nello stesso assembly o da una classe derivata in un altro assembly.
protetto privato	Il tipo o membro è accessibile solo dal codice nella stessa classe o in una classe derivata all'interno dell'assembly della classe di base.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApplication
8  {
9      class Program
10     {
11         public static string valoreA { get; set; }
12         public static string valoreB { get; set; }
13
14         static void Main(string[] args)
15         {
16             Program.
17             #region Console
18             Console.
19             Console.
20             #endregion
21         }
22     }
23 }

```

Equals
ReferenceEquals
valoreA
valoreB

bool object.Equals(object objA, object objB)
 Determines whether the specified object instances are considered equal.
 Nota: premere due volte TAB per inserire il frammento di codice 'Equals'.

Applicazione di una regola di visibilità per la Console Application. Il metodo di Entry Point della console deve essere sempre statico e l'argomento deve essere sempre un array di stringhe

ACADEMY
.NET CORE / C#

Presentazione Accademy .NET Core C#

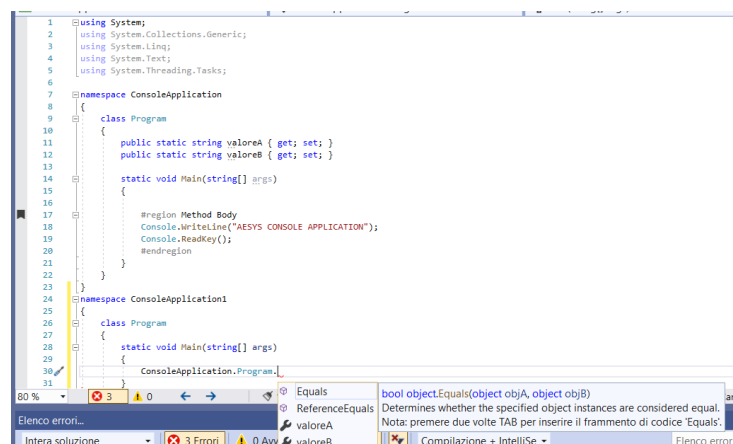
✓ Module 1 C# Language – Module 2 Writing C# Program

In questo esempio praticamente nella stesso sorgente Program.cs abbiamo due namespace:

Dal secondo namespace possiamo assicurarci della visibilità delle proprietà non di istanza, in quanto avendo il metodo Entry Point statico la classe è statica.

La visibilità è data proprio dal fatto che sono state dichiarate con modificatore public

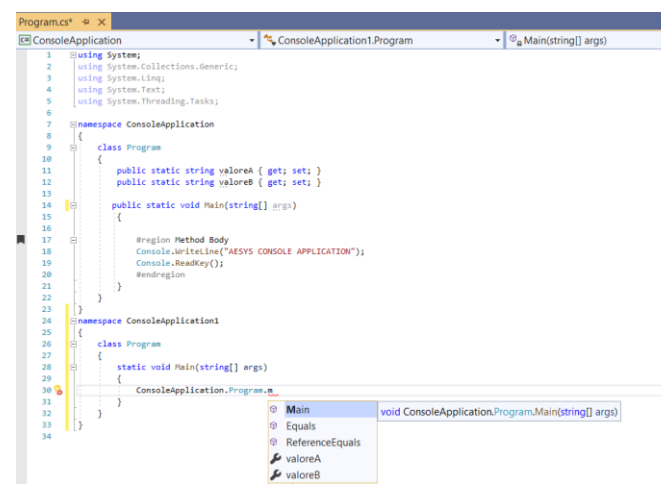
Non vediamo l'entry Point Main semplicemente perché, se manca un modificatore, di default è private e quindi non visibile.



```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApplication
8 {
9     class Program
10     {
11         public static string valoreA { get; set; }
12         public static string valoreB { get; set; }
13
14         static void Main(string[] args)
15         {
16             #region Method Body
17             Console.WriteLine("AESYS CONSOLE APPLICATION");
18             Console.ReadKey();
19             #endregion
20         }
21     }
22 }
23
24 namespace ConsoleApplication1
25 {
26     class Program
27     {
28         static void Main(string[] args)
29         {
30             ConsoleApplication.Program.Main(args);
31         }
32     }
33 }

```



```

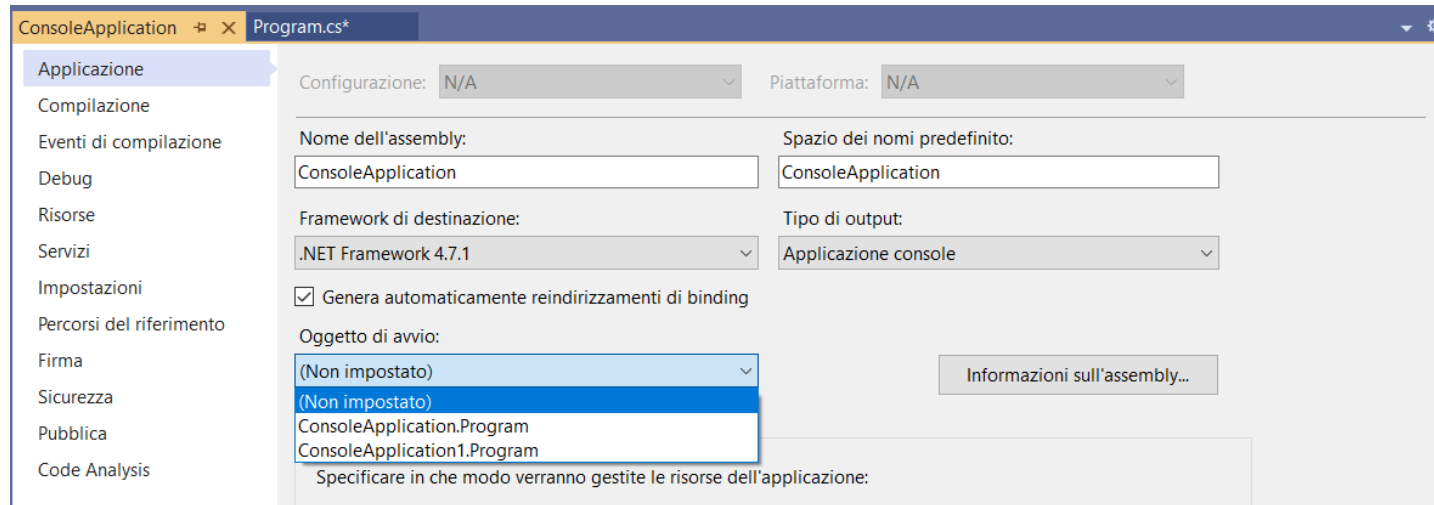
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApplication
8 {
9     class Program
10     {
11         public static string valoreA { get; set; }
12         public static string valoreB { get; set; }
13
14         public static void Main(string[] args)
15         {
16             #region Method Body
17             Console.WriteLine("AESYS CONSOLE APPLICATION");
18             Console.ReadKey();
19             #endregion
20         }
21     }
22 }
23
24 namespace ConsoleApplication1
25 {
26     class Program
27     {
28         static void Main(string[] args)
29         {
30             ConsoleApplication.Program.Main(args);
31         }
32     }
33 }

```

E' bastato aggiungere il modificatore public e vediamo visibile dall'altro namespace l'entry point Main

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

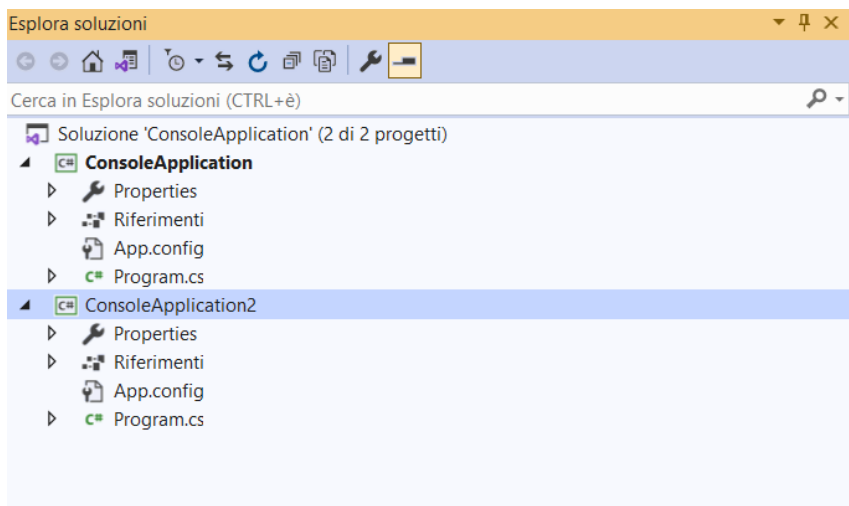


Altra regola collegata alle Console Application che in un namespace e quindi in una classe Program ci può essere solo un entry Point Main altrimenti il compilatore genera una eccezione Bloccante

Invece se dovessimo avere due namespace differenti allora dalla finestra delle proprietà potremmo scegliere un oggetto di avvio (ossia l'entry point che verrà eseguito per prima)

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program



Abbiamo due progetti Console Application e possiamo verificare nel concreto due Assembly che hanno due namespace diversi (ConsoleApplication.Exe) – (ConsoleApplication2.Exe)

Vorremmo farli comunicare e di qui nasce l'esercizio da svolgere in autonomia dove bisogna referenziare nel secondo Assembly (ConsoleApplication2.Exe) il primo e poter richiamare l'entry point Main

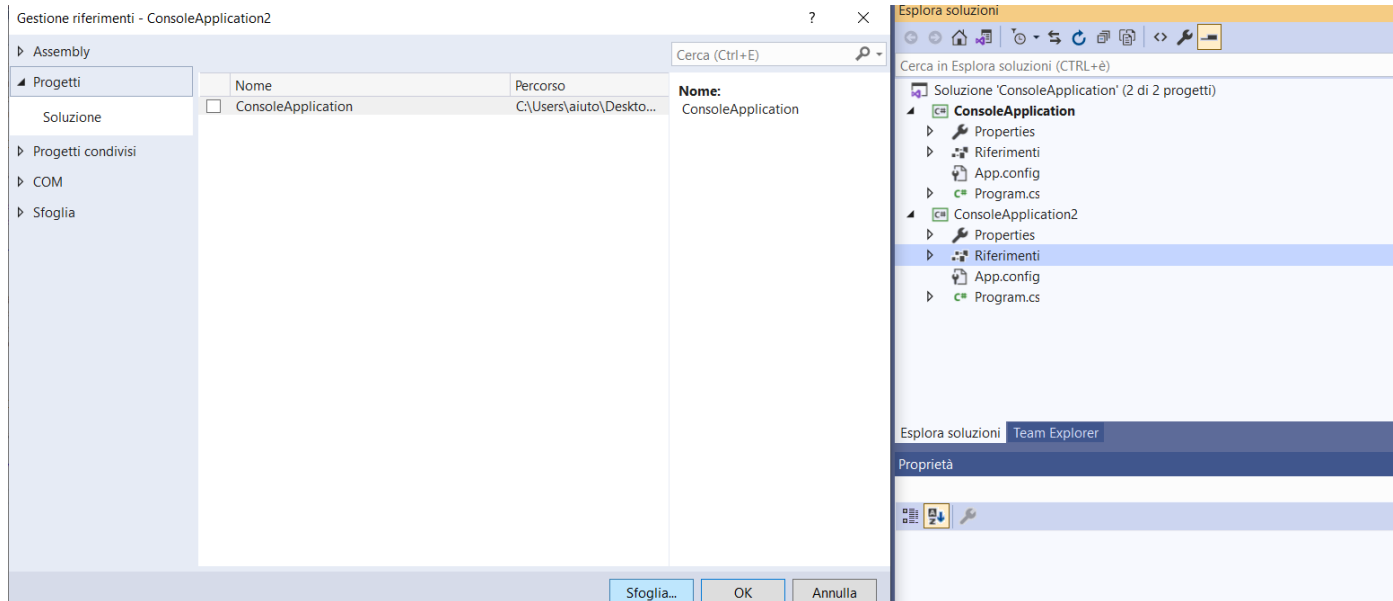
```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApplication
8  {
9      class Program
10     {
11         public static string valoreA { get; set; }
12         public static string valoreB { get; set; }
13
14         public static void Main(string[] args)
15         {
16
17             #region Method Body
18             Console.WriteLine("AESYS CONSOLE APPLICATION");
19             Console.ReadKey();
20             #endregion
21         }
22     }
23 }
24 namespace ConsoleApplication1
25 {
26     class Program
27     {
28         static void Main(string[] args)
29         {
30
31         }
32     }
33 }
34

```

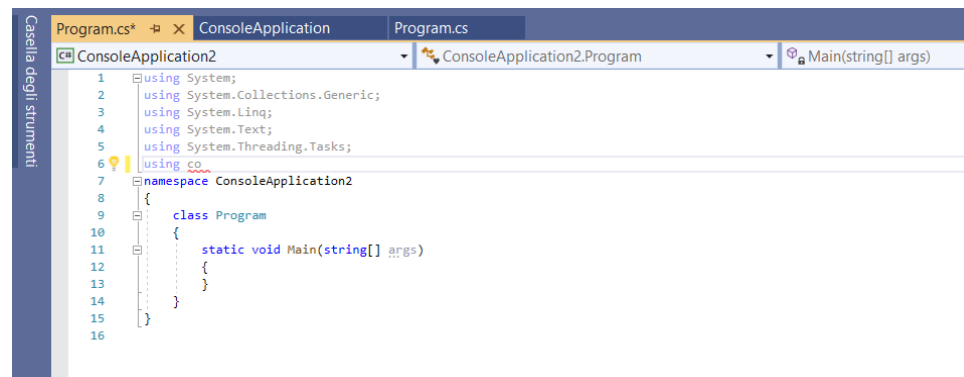
Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program



**Si provvede a fare test destro del mouse
Su Riferimenti e scegliere la categoria
Soluzione e referenziare il progetto
ConsoleApplication. Così si crea un legame
tra i due progetti di questa soluzione**

**Il progetto nell'invocare una
Direttiva using non vede il
Progetto referenziato
(ConsoleApplication)
Nella slide successiva spie-
gheremo la motivazione**



Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

Motivazione -> Le due classi contenenti i due entry Point della ConsoleApplication (static void Main...) non hanno la classe Program con modificatore di accesso Program

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApplication
8 {
9     class Program
10     {
11         public static string valoreA { get; set; }
12         public static string valoreB { get; set; }
13
14         public static void Main(string[] args)
15         {
16             #region Method Body
17             Console.WriteLine("AESYS CONSOLE APPLICATION");
18             Console.ReadKey();
19             #endregion
20         }
21     }
22 }
23
24 namespace ConsoleApplication1
25 {
26     class Program
27     {
28         static void Main(string[] args)
29         {
30         }
31     }
32 }
33
34

```

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using ConsoleApplication;
7 using ConsoleApplication1;
8 namespace ConsoleApplication2
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             ConsoleApplication.Program.L
15         }
16     }
17 }

```

Motivazione valida e testata

Dal progetto con la reference a ConsoleApplication, è visibile il Secondo namespace ConsoleApplication1 E la classe Program ma non si vede L'entry point Main

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using ConsoleApplication;
7 using ConsoleApplication1;
8 namespace ConsoleApplication2
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             ConsoleApplication1.Program.L
15         }
16     }
17 }

```

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

Uno sguardo generale ai Tipi che è possibile dichiarare:

- ❑ A livello di classe come property
- ❑ A livello locale come variabili di metodo o di costrutti di programmazione strutturata (che definiremo nella slide successiva)
- ❑ Classificabili in diverse categorie (**Primitive types** di tipo integer o Values Type ad esempio, **reference type**, **boolean and String types**, **floating point types**)

TYPE	ALIAS FOR	ALLOWED VALUES
sbyte	System.SByte	Integer between -128 and 127
byte	System.Byte	Integer between 0 and 255
short	System.Int16	Integer between -32768 and 32767
ushort	System.UInt16	Integer between 0 and 65535
int	System.Int32	Integer between -2147483648 and 2147483647
uint	System.UInt32	Integer between 0 and 4294967295
long	System.Int64	Integer between -9223372036854775808 and 9223372036854775807
ulong	System.UInt64	Integer between 0 and 18446744073709551615

I tipi elencati nella prima colonna sono mappati Con i tipi di .NET (framework) dove la classe progenitore è System

ACADEMY

.NET CORE / C#

■ Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

TYPE	ALIAS FOR	MIN M	MAX M	MIN E	MAX E	APPROX MIN VALUE	APPROX MAX VALUE
float	System.Single	0	22^4	-149	104	1.5×10^{-45}	3.4×10^{38}
double	System.Double	0	25^3	-1075	970	5.0×10^{-324}	1.7×10^{308}
decimal	System.Decimal	0	29^6	-28	0	1.0×10^{-28}	7.9×10^{28}

TYPE	ALIAS FOR	ALLOWED VALUES
char	System.Char	Single Unicode character, stored as an integer between 0 and 65535
bool	System.Boolean	Boolean value, true or false
string	System.String	A sequence of characters

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

Simple Mathematical Operator

OPERATOR	CATEGORY	EXAMPLE EXPRESSION	RESULT
+	Binary	var1 = var2 + var3;	var1 is assigned the value that is the sum of var2 and var3.
-	Binary	var1 = var2 - var3;	var1 is assigned the value that is the value of var3 subtracted from the value of var2.
*	Binary	var1 = var2 * var3;	var1 is assigned the value that is the product of var2 and var3.

/	Binary	var1 = var2 / var3;	var1 is assigned the value that is the result of dividing var2 by var3.
%	Binary	var1 = var2 % var3;	var1 is assigned the value that is the remainder when var2 is divided by var3.
+	Unary	var1 = +var2;	var1 is assigned the value of var2.
-	Unary	var1 = -var2;	var1 is assigned the value of var2 multiplied by -1.

ACADEMY

.NET CORE / C#

OPERATOR	CATEGORY	EXAMPLE EXPRESSION	RESULT
+	Binary	var1 = var2 + var3;	var1 is assigned the value that is the concatenation of the two strings stored in var2 and var3.

Presentazione Accademy .NET Core C#

✓ Module 1 C# Language – Module 2 Writing C# Program

Simple Mathematical Operator

OPERATOR	CATEGORY	EXAMPLE EXPRESSION	RESULT
++	Unary	var1 = ++var2;	var1 is assigned the value of var2 + 1. var2 is incremented by 1.
--	Unary	var1 = --var2;	var1 is assigned the value of var2 - 1. var2 is decremented by 1.
++	Unary	var1 = var2++;	var1 is assigned the value of var2. var2 is incremented by 1.
--	Unary	var1 = var2--;	var1 is assigned the value of var2. var2 is decremented by 1.

Assignments Operators

OPERATOR	CATEGORY	EXAMPLE EXPRESSION	RESULT
=	Binary	var1 = var2;	var1 is assigned the value of var2.
+=	Binary	var1 += var2;	var1 is assigned the value that is the sum of var1 and var2.
-=	Binary	var1 -= var2;	var1 is assigned the value that is the value of var2 subtracted from the value of var1.
*=	Binary	var1 *= var2;	var1 is assigned the value that is the product of var1 and var2.
/=	Binary	var1 /= var2;	var1 is assigned the value that is the result of dividing var1 by var2.
%=	Binary	var1 %= var2;	var1 is assigned the value that is the remainder when var1 is divided by var2.

■ Day 1 – Final Module

Laboratorio svolto dal Trainer



Day 1
Laboratorio
svolto dal
Trainer



ACADEMY

.NET CORE / C#