

Esercitazione OOP (Object Oriented Programming)

Esercitazioni OOP



Esercitazioni
OOP



ACADEMY

.NET CORE / C#

Esercitazioni OOP (Object Oriented Programming)

Sviluppare un'applicazione orientata agli oggetti per gestire i prestiti che una banca concede ai propri clienti. La banca è caratterizzata da un nome e da un insieme di clienti. I clienti sono caratterizzati da nome, cognome, codice fiscale stipendio. Il prestito concesso al cliente, considerato intestatario del prestito, è caratterizzato da un ammontare, una rata, una data inizio, una data fine. Per i clienti e per i prestiti si vuole stampare un prospetto riassuntivo con tutti i dati che li caratterizzano in un formato di tipo stringa a piacere.

Per la banca deve essere possibile aggiungere, modificare, eliminare e ricercare un cliente. Inoltre, la banca deve poter aggiungere un prestito. La banca deve poter eseguire delle ricerche sui prestiti concessi ad un cliente dato il codice fiscale. La banca vuole anche sapere, dato il codice fiscale di un cliente, l'ammontare totale dei prestiti concessi.

Suggerimento: Non si richiede nessuna connessione ad una base dati ma solo quella di creare una `List<T>` dove `T` è il nome che assegnate alla classe per l'implementazione

Esercitazioni OOP (Object Oriented Programming)

Si vuole progettare un sistema per la gestione di una biblioteca. Gli utenti registrati al sistema, fornendo cognome, nome, email, password, recapito telefonico, possono effettuare dei prestiti sui documenti che sono di vario tipo (libri, DVD). I documenti sono caratterizzati da un codice identificativo di tipo stringa (ISBN per i libri, numero seriale per i DVD), titolo, anno, settore (storia, matematica, economia, ...), stato (In Prestito, Disponibile), uno scaffale in cui è posizionato, un elenco di autori (Nome, Cognome). Per i libri si ha in aggiunta il numero di pagine, mentre per i dvd la durata. L'utente deve poter eseguire delle ricerche per codice o per titolo e, eventualmente, effettuare dei prestiti registrando il periodo (Dal/AI) del prestito e il documento. Il sistema per ogni prestito determina un numero progressivo di tipo alfanumerico. Deve essere possibile effettuare la ricerca dei prestiti dato nome e cognome di un utente.

Suggerimento: Bisognerà rappresentare uno schema con le classi le proprietà e i metodi e soprattutto mettere in enfasi le strutture dati tra quelle di tipo Complex Type (Enum, Structs, Collections tipo List)
Alla fine dopo una attenta analisi sviluppare frammenti di codice per la gestione delle classi annoverate nell'analisi

ACADEMY

.NET CORE / C#

Esercitazioni OOP (Object Oriented Programming)

Progettare la struttura ad oggetti per gestire un carrello di un sito di e-commerce. Un carrello è associato all'utente collegato al sito, e per ogni utente è previsto un solo carrello. È importante risalire dall'utente al carrello e non il viceversa. L'utente è caratterizzato da Username, Password (privato), Nome e Cognome. Ogni carrello può contenere diversi prodotti; per ogni prodotto nel carrello è riportata la quantità, il prezzo, l'eventuale sconto applicato (in termini percentuali). Ogni prodotto è caratterizzato da un codice, una descrizione, un prezzo. Dato un prodotto non è necessario risalire ai carrelli in cui è stato aggiunto.

Sul carrello è possibile eseguire le seguenti operazioni:

- ☐ Aggiungi prodotto
- ☐ Elimina prodotto
- ☐ Modifica di un prodotto già inserito
- ☐ Calcola totale

•Stampa a video del carrello (formato a piacere)

Si noti che nel caso sia inserito un prodotto che già esiste nel carrello questo va a modificare la quantità del prodotto precedentemente inserito.

Esercitazioni OOP (Object Oriented Programming)

Si vuole realizzare un sistema per la gestione di Corsi. Ciascun corso ha un nome e un numero intero che indica l'edizione. Ciascun corso si articola in un certo numero di lezioni. Ogni lezione ha una descrizione, una data, un orario di inizio, una durata, un docente e un'aula assegnata.

I Docenti hanno un nome, un cognome e un titolo di studio.

Ogni Aula ha una capienza, un nome e un elenco di risorse (es. Video Proiettore, PC, Notebook, Tablet, LIM, etc.).

Ogni corso ha un certo numero di studenti partecipanti, iscritti a quella specifica edizione. Ogni studente ha un nome, un cognome e una matricola. Per ogni lezione occorre tenere traccia dei presenti.

Il sistema deve consentire:

- ☐ Aggiungere corso
- ☐ Aggiungere lezioni ad un corso
- ☐ Aggiungere studenti ad un corso
- ☐ Segnare gli assenti ad una lezione

Il sistema deve consentire le seguenti stampe a video:

- Elenco Corsi
- Elenco delle lezioni di un corso
- Elenco degli iscritti a un corso
- Scheda riassuntiva di una lezione
- Elenco dei presenti ad una lezione
- Media dei presenti ad una lezione
- Media dei presenti ad un corso (media tra le lezioni)

ACADEMY

.NET CORE / C#

Esercitazioni OOP (Object Oriented Programming)



Un albergo intende realizzare un software per la gestione delle prenotazioni delle camere. I clienti sono individuati da: codice fiscale, cognome, nome, città, provincia, e-mail, telefono e cellulare. Ogni camera è individuata da un numero, una descrizione e la tipologia (singola, doppia). La prenotazione effettuata da un cliente si riferisce ad una sola camera. Per ogni prenotazione si deve memorizzare la data della prenotazione, un numero progressivo nell'anno, l'anno, il periodo di soggiorno (dal, al), la caparra confirmatoria e la tariffa applicata. La tariffa applicata dipende dal tipo di camera (singola, doppia), dal trattamento (mezza pensione, pensione completa, pernottamento con prima colazione) e dal periodo dell'anno (dal, al).

Durante il soggiorno possono essere richiesti servizi aggiuntivi (Colazione in camera, bevande e cibo nel mini bar, internet, letto aggiuntivo, culla) da caricare sulla prenotazione. Per ogni servizio è memorizzata la data, la quantità e il prezzo. Sui servizi è specificato il prezzo (comprensivo di unità euro oppure euro/giorno), una descrizione.

Il software deve:

- Consentire di caricare una nuova prenotazione
- Ricercare una prenotazione per numero e anno
- Ricercare le prenotazioni per
 - Cognome e/o Nome cliente
 - Data prenotazione
- Per ogni prenotazione stampare una scheda riassuntiva con tutti i dati significativi
- Per ogni prenotazione stampare una scheda riassuntiva dei costi, il totale, la caparra e il saldo finale da pagare



ACADEMY

.NET CORE / C#

Day 7

**Module 11 Cloud and cross-platform
Programming (3 ORE)**

**Module 12 Advanced Cloud Programming and
Deployment (5 ore)**



Day 7

■ Presentazione Accademy .NET Core C#

✓ Day 7 – Module Cloud

The Cloud, Cloud Computing Cloud optimized stack

Cloud patterns and best practices

Microsoft Azure C# Libraries to create a Db SQL Azure

Creating an ASP.Net 4.7 Web site and publishing on Azure Cloud

Creating an ASP.Net Core Web Application and publishing on Azure Cloud

Cloud Computing on Microsoft Azure



Alla base della programmazione C# abbiamo imparato o sappiamo della fattibilità per

- ✓ Creare applicazioni Console anche con su architettura cross-platfom (non solo Windows – distro Linux tipo Ubuntu);
Si potranno anche attraverso un MAC poter creare esclusivamente applicazioni con la piattaforma .NET Core attraverso
- ✓ un IDE apposito Visual Studio Code oppure Visual Studio su Mac
- ✓ Si potranno creare applicazioni per Linux solo ed esclusivamente con Visusal Studio Code
- ✓ Per la piattaforma .NET Core si potranno creare applicazioni con interfaccia grafica sia per workstation Windows e non solo (WPF – Windows Presentation Foundation)
- ✓ Per la piattaforma .NET Core sarà possibile creare class library (cosiddette librerie di classe o DLL) che sarà possibile condi-
viderle in tutte le soluzioni che supportano tale piattaforma



ACADEMY

.NET CORE / C#

Cloud Computing on Microsoft Azure



Anche se questi sono fattibili e convincenti tecniche di sviluppo, non sono esempi adatti di programmi per ospitare ed eseguire nel cloud. Questi tipi di programmi sono classici distribuito ed eseguito sul computer, tablet o dispositivo mobile di un utente (tipo applicazioni Universal App Platform che da Microsoft sono denominate con l'acronimo UWP

Questi programmi sono compilati in eseguibili o linkati dinamici librerie che hanno dipendenze da software preinstallato come il .NET Framework oppure il runtime di .NET Core ad esempio. Queste dipendenze sono generalmente si presume che siano presenti nel luogo in cui sono installati, o vengono inclusi nella procedura di installazione

Tutte le tipologie di applicazioni diverse dalle precedenti (Applicazioni web con il servizio di ASP.Net o ASP.Net Core oppure Web Api (denominati anche servizi Web) invece sono sicuramente dipendenti dalla presenza di un Web Server tipo IIS – (Internet Information Services), senza del quale si potranno solo eseguire all'interno di un ambiente di sviluppo tipo Visual Studio, se legati fortemente al sistema operativo Windows, oppure per tutte le tipologie di web application realizzabili con la piattaforma .NET Core sarà possibile utilizzare Visual Studio Code su distro Linux. Per Mac OS invece si utilizza una versione di Visual Studio per Mac. Con la stessa piattaforma .NET Core su Windows sarà possibile usare Visual Studio Community e moduli di hosting appositi per IIS al fine di poter distribuire ed eseguire applicazioni Web sotto IIS

ACADEMY

.NET CORE / C#



Cloud Computing on Microsoft Azure



Motivo valido, che queste tipologie di applicazioni Web sono candidate ad essere distribuite, dalla propria Workstation ,sul cloud :

La scalabilità e la possibilità di ottimizzare ed abbattere tutti i costi di gestione dell'amministrazione di un server interno aziendale e tenere a stipendio un amministratore di rete che si dovrà anche occupare della gestione straordinaria tipo sistemare i server a livello hardware distribuire politiche di backup incrementale etc etc.

Si potrà decidere con l'autoscaling di attivare o disattivare i servizi distribuiti sul Cloud a seconda del picco di utilizzo (esempio. Per accessi molto frequenti ad un proprio portale di E-commerce nel fine settimana, si potrà aumentare attivare l'app service web distribuita e nel caso questi accessi dovessero diminuire tenere disattivata l'app service)

Tutte le risorse utilizzate in seno all'attivazione di una sottoscrizione pay to consumer sono effettivamente queste ad essere pagate dall'azienda e quindi fatturate da Microsoft su base mensile sulla base di una calcolatrice che stima nella stessa ragione in base a svariati parametri

La gestione del DataCenter spetta direttamente a Microsoft che si obbliga verso i suoi clienti di garantire l'integrità dei servizi, garantire anche la gestione dei dati sensibili secondo le normative europee vigenti e soprattutto gestire tutta l'infrastruttura e tenere un servizio di supporto h24 per tutti coloro che loro richiedono per disservizi

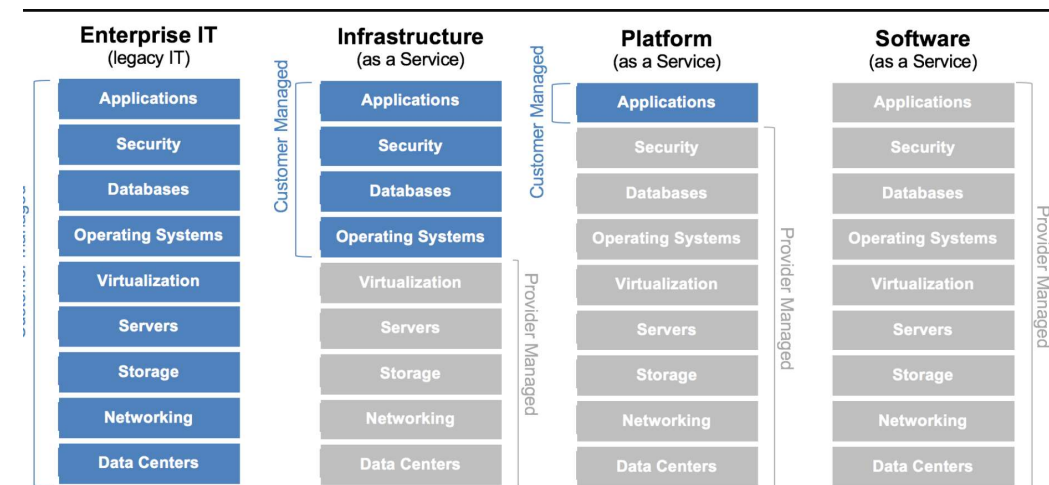
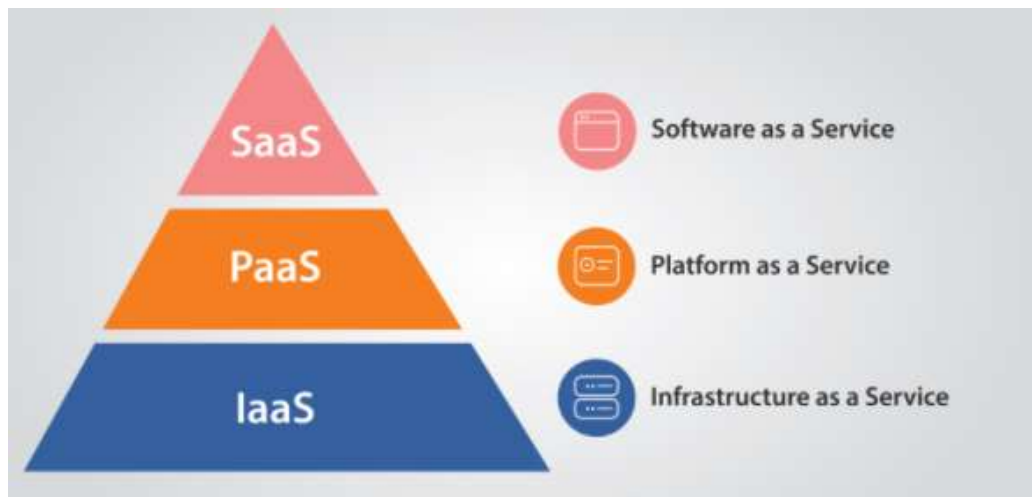
ACADEMY

.NET CORE / C#



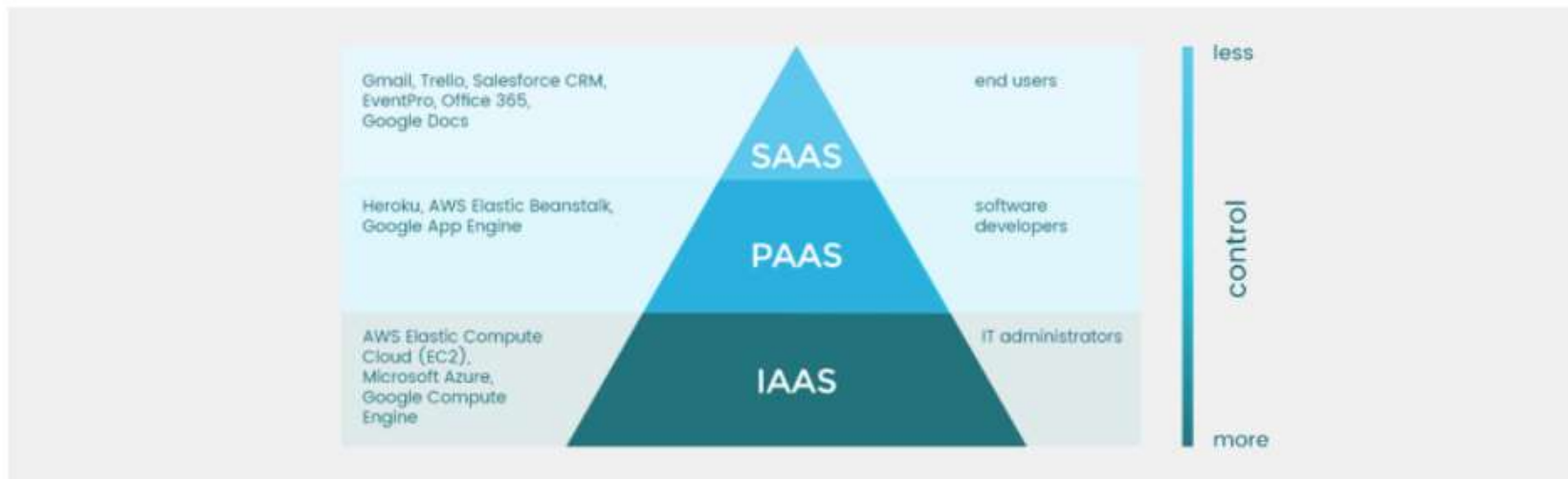
Cloud Computing on Microsoft Azure

Si aggiunge altra motivazione che dovrebbe stimolare tutte le aziende ad utilizzare i servizi di cloud computing anche per implementare soluzioni di business intelligence con database e soluzioni del modello relazionale e non (quindi tutti i programmi noti per tale funzione e che molte aziende utilizzano in stile enterprise tipo SQL Server, Oracle, MySQL, etc. etc) saranno utilizzati in ottica cloud con la caratteristica dell'iper-scalabilità e performance sotto forma di servizi contro il pagamento mensile di una sottoscrizione sempre in rapporto al grado di utilizzo delle risorse in ragione mensile



Cloud Computing on Microsoft Azure

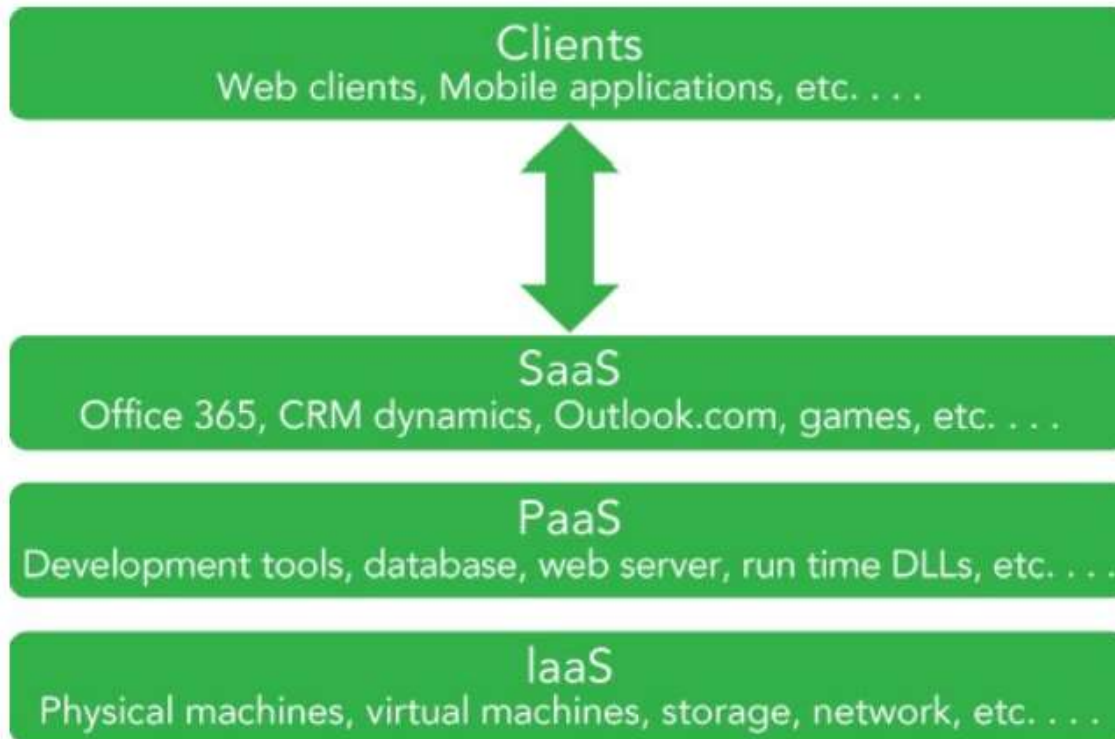
IaaS, PaaS, and SaaS – how do they differ?



Cloud Computing on Microsoft Azure

- **Public cloud** is shared computer hardware and infrastructure owned and operated by a cloud provider like Microsoft Azure, Amazon AWS, Rackspace, or IBM Cloud. This cloud type is ideal for small and medium businesses that need to manage fluctuations in customer and user demands.
- **Private cloud** is dedicated computer hardware and infrastructure that exists onsite or in an outsourced data center. This cloud type is ideal for larger companies or those that must deliver a higher level of data security or government compliance.
- **Hybrid cloud** is a combination of both public and private cloud types whereby you choose which segments of your IT solution run on the private cloud and which run on the public cloud. The ideal solution is to run your businesses-critical programs that require a greater level of security in the private cloud and run non-sensitive, possibly spiking tasks in the public cloud.

Cloud Computing on Microsoft Azure



- **Infrastructure as a Service (IaaS)**—You are responsible from the operating system upward. You are not responsible for the hardware or network infrastructure; however, you are responsible for operating system patches and third-party dependent libraries.
- **Platform as a Service (PaaS)**—You are responsible only for your program running on the chosen operating system and its dependencies. You are not responsible for operating system maintenance, hardware, or network infrastructure.
- **Software as a Service (SaaS)**—A software program or service used from a device that is accessed via the Internet. For example, O365, Salesforce, OneDrive or Box, all of which are accessible from anywhere with an Internet connection and do not require software to be installed on the client to function. You are only responsible for the software running on the platform and nothing else.

Cloud Computing on Microsoft Azure



Lo stack ottimizzato per il cloud è un concetto utilizzato per fare riferimento al codice che può gestire un'elevata produttività, ha un ingombro ridotto, può funzionare fianco a fianco con altre applicazioni sullo stesso server ed è multiplatforma abilitato. Un ingombro ridotto si riferisce al confezionamento nel tuo programma cloud solo dei componenti per i quali esiste una dipendenza, rendendo la loro dimensione di distribuzione più piccola possibile.

In ottica Cloud computing un programma potrebbe richiedere l'intero .NET Framework per funzionare. Se no, allora invece di impacchettare l'intero .NET Framework, si va ad includere solo le librerie necessarie per eseguire il tuo programma cloud e quindi compilare il tuo programma cloud in un'applicazione autonoma per supportare side-by-side l'esecuzione.

Infine, utilizzando una versione open source di Mono, .NET Core o ASP.NET Core il programma cloud può essere impacchettato, compilato e distribuito su sistemi operativi diversi da Microsoft, ad esempio Mac OS X, iOS o Linux



ACADEMY





.NET CORE / C#

Cloud Computing on Microsoft Azure

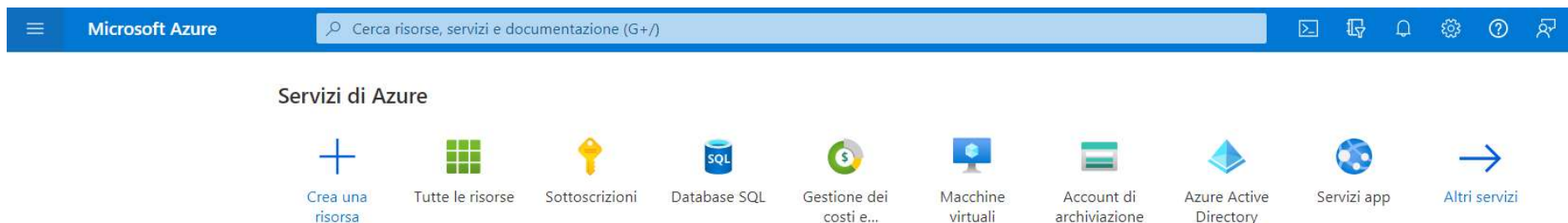
Ci sono diversi e svariati canali per utilizzare tutti i servizi messi a disposizione della piattaforma cloud di Microsoft

- ❑ Se studente, sviluppatore o azienda puoi sempre provare una sottoscrizione free di un mese di prova e nella gamma messa a disposizione da Microsoft sempre si possono utilizzare in modo free anche dopo altri secondo uno schema presente a questo url: [Free Services | Microsoft Azure](#)

Azure services that are free for 12 months
When you try Azure free with a USD200 credit and then move to pay as you go

COMPUTE	COMPUTE	STORAGE	STORAGE
			
Azure Virtual Machines—Linux	Azure Virtual Machines—Windows	Azure Managed Disks	Azure Blob Storage
750 hours B1s burstable virtual machines	750 hours B1s burstable virtual machines	2 64 GB (P6) solid state drives SSD storage, plus 1 GB snapshot and 2 million I/O operations	5 GB locally redundant storage (LRS) hot block with 20,000 read and 10,000 write operations
Create Linux virtual machines with on-demand capacity in seconds.	Create Windows virtual machines with on-demand capacity in seconds.	Get high performance, durable block storage for Azure Virtual Machines with simplified management.	Use massively scalable object storage for any type of unstructured data.

Cloud Computing on Microsoft Azure



Attraverso una sottoscrizione free oppure pay to consume si potranno:

- ☐ Creare gruppi di risorse per creare storage container al fine di contenere BLOB (Binary Large Object) in Cloud, o secret key o certificati (Key Vault Container)
- ☐ Creare database SQL
- ☐ Creare VM (Virtual Machine seguendo opportuni template pronti o configurare la creazione secondo le proprie esigenze

[Servizio App | Microsoft Azure](#)

ACADEMY
.NET CORE / C#

Guide introduttive di 5 minuti

Crea la tua prima app Web in Windows o Linux con:

[Node.js](#)

[ASP.NET](#) o [.NET Core](#)

[Python](#)

[Java](#)

Moduli di apprendimento

Indicazioni dettagliate su come creare e ospitare app Web da Microsoft Learn:

[Distribuisci ed esegui un'app Web containerizzata](#)

[Ospita un'app Web nel Servizio app](#)

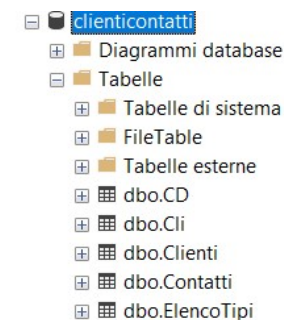
Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe



La libreria di classe con .NET Core dovrà permettere di:

- ☐ di interagire con uno spazio di database SQL per gestire le seguenti entità
- ☐ Si dovrà prevedere la gestione di storedProcedure CRUD (Create Read Update Delete) per gestire tali entità ed integrare dei metodi all'interno della libreria con il DataAccess ADO.Net al fine di provvedere alla sua esecuzione a richiesta
- ☐ Integrare dei metodi parametrici al fine di ottenere la serializzazione in XML/JSON/CSV/XLS delle suddette entità (valutare di adottare la libreria IRONXL)
- ☐ Si fornisce al fine di lavorare uno script SQL
- ☐ Integrare la libreria implementata all'interno una console Application .NET Core con la gestione di un menu nella shell dei comandi per l'esecuzione dei metodi implementati



ACADEMY

.NET CORE / C#

Cloud Computing on Microsoft Azure

Esercitazione – Integrazione della libreria

Implementata anche in un app ASP.Net Core

Sviluppare un'applicazione ASP.Net Core per integrare la libreria di classe di cui si hanno le specifiche descritte nella slides precedente

Suggerimento -> Per coloro i quali non dovessero avere credito/o attivato una sottoscrizione si può prevedere la pubblicazione di tale applicazione direttamente sul web server IIS.

Per coloro i quali dovessero avere anche una sottoscrizione attiva su Azure pubblicare l'app , ma prima testarla con l'ambiente di sviluppo fornendo un link per l'accesso via browser al docente ai fini della valutazione.

Tale suggerimento vale anche per le specifiche previste per l'attività descritta nella slides precedente

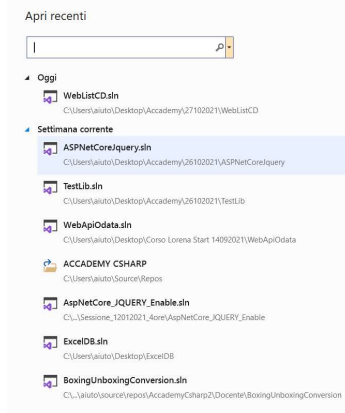
Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

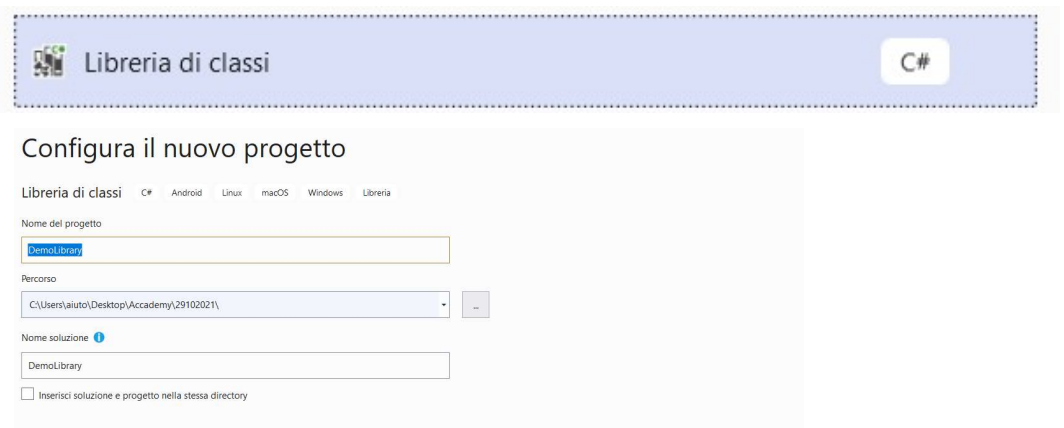


Ricordo in sintesi le attività che vi saranno di notevole aiuto all'esecuzione delle attività principali previste nelle slides precedenti:

Visual Studio 2019



- ☐ Cliccare su Creare Nuovo Progetto
- ☐ Scegliere



Informazioni aggiuntive

Libreria di classi C# Android Linux macOS Windows Libreria

Framework di destinazione

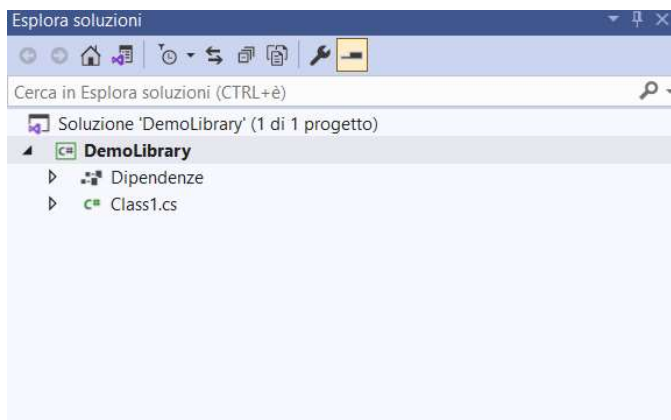
.NET 5.0 (Corrente)

ACADEMY
.NET CORE / C#

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

Raccolte tutte le informazioni di configurazione Visual Studio genererà il progetto seguente:



```
1 using System;
2
3 namespace DemoLibrary
4 {
5     public class Class1
6     {
7     }
8 }
9
```

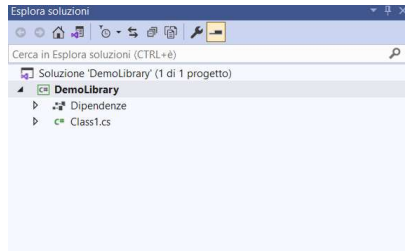
Note operative si dovranno aggiungere nel file .csproj i seguenti namespace strumentali a interagire con il file appsettings.json (file di configurazione)

Il namespace System.Data.SqlClient

Nella slides successiva si potrà capire la procedura corretta

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



Si clicca sul nome del **progetto DemoLibrary** e si avrà e prima del tag di chiusura **Project** si dovranno indicare le seguenti reference:

<ItemGroup>

```
<PackageReference Include="Microsoft.Extensions.Configuration" Version="5.0.0" />
<PackageReference Include="Microsoft.Extensions.Configuration.FileExtensions" Version="5.0.0" />
<PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="5.0.0" />
<PackageReference Include="Microsoft.Extensions.Configuration.UserSecrets" Version="5.0.0" />
<PackageReference Include="Microsoft.Extensions.DependencyInjection" Version="5.0.2" />
<PackageReference Include="System.Data.SqlClient" Version="4.8.3" />
```

</ItemGroup>

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

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



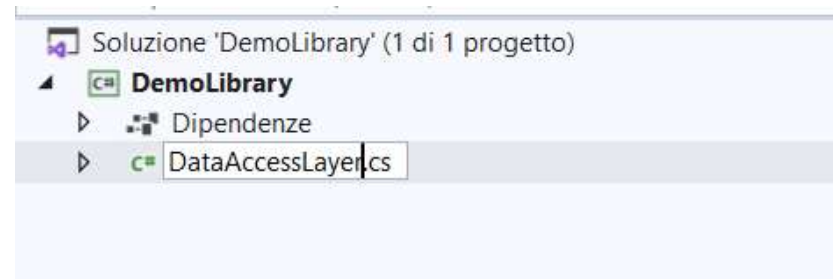
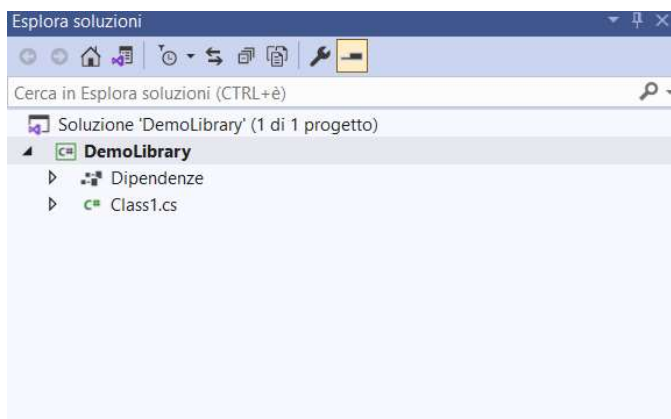
Il file csproj all'atto di salvataggio dovrà presentarsi in questo modo:

```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <TargetFramework>net5.0</TargetFramework>
5   </PropertyGroup>
6
7   <ItemGroup>
8
9     <PackageReference Include="Microsoft.Extensions.Configuration" Version="5.0.0" />
10    <PackageReference Include="Microsoft.Extensions.Configuration.FileExtensions" Version="5.0.0" />
11    <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="5.0.0" />
12    <PackageReference Include="Microsoft.Extensions.Configuration.UserSecrets" Version="5.0.0" />
13    <PackageReference Include="Microsoft.Extensions.DependencyInjection" Version="5.0.2" />
14    <PackageReference Include="System.Data.SqlClient" Version="4.8.3" />
15  </ItemGroup>
16
17
18 </Project>
19
```

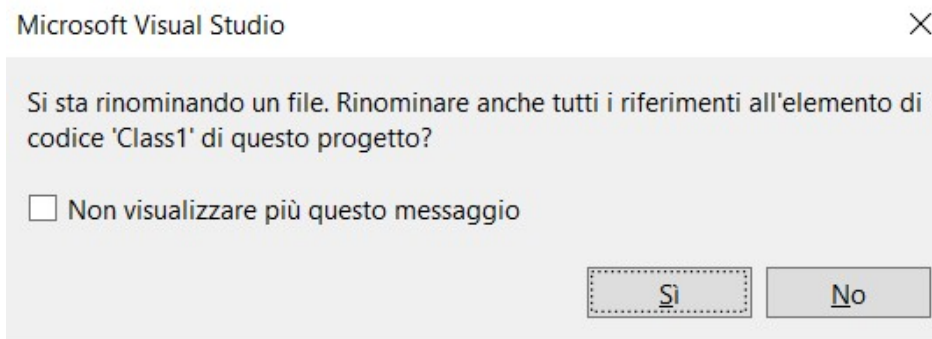

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

Provvedere a rinominare il nome dell'entry point della classe Class1 in DataAccess.cs



Visual Studio provvederà a visualizzare questa finestra di dialogo: Confermare Con un click sul pulsante Si



Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



```
1 using System;
2
3 namespace DemoLibrary
4 {
5     public class DataAccessLayer
6     {
7     }
8 }
9
```

```
1 using System;
2
3 namespace DemoLibrary
4 {
5     public class DataAccessLayer
6     {
7         public string Metodo1()
8         {
9             return "Metodo1 eseguito";
10        }
11
12        public string Metodo2()
13        {
14            return "Metodo2 eseguito";
15        }
16
17        public string Metodo3(string messaggio)
18        {
19            return "Messaggio inviato al metodo:" + messaggio;
20        }
21    }
22 }
23
```

All'interno del blocco di codice Public Class DataAccessLayer implementare tutti i metodi previsti per la gestione delle attività previste

Una volta compilata la libreria di classe e si vorrà eseguirla essa non la si potrà eseguire finché non viene referenziata
In un progetto Console Application, Web Application di qualsiasi Tipo ASP.net Core ASP.Net Core Model View Controller – Web Api

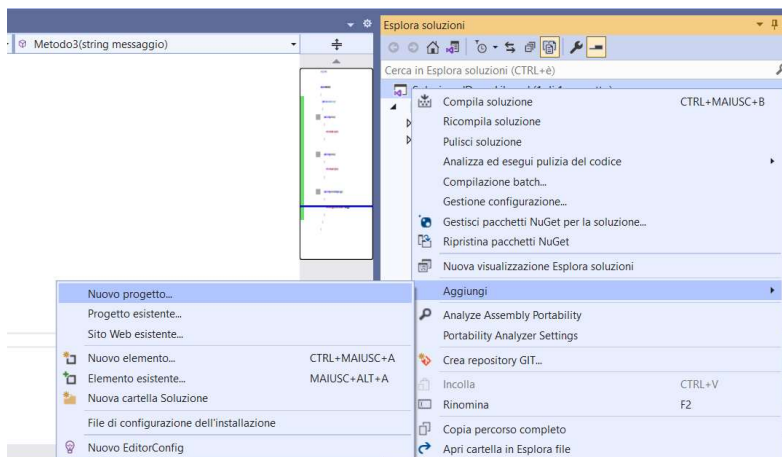
```
Output
Mostra output di: Compilazione
Compilazione avviata...
1>----- Inizio compilazione: Progetto: DemoLibrary, Configurazione: Debug Any CPU -----
1>DemoLibrary -> C:\Users\aiuto\Desktop\Accademy\29102021\DemoLibrary\DemoLibrary\bin\Debug\net5.0\DemoLibrary.dll
===== Compilazione: 1 completate, 0 non riuscite, 0 aggiornate, 0 ignorate =====
```

ACADEMY
.NET CORE / C#

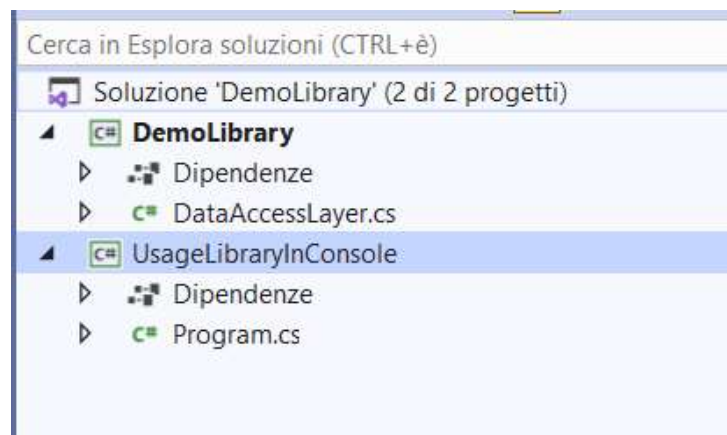


Cloud Computing on Microsoft Azure Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

Se si dovesse creare una console Application all'interno della stessa soluzione, bisognerà aggiungere un riferimento alla libreria compilata



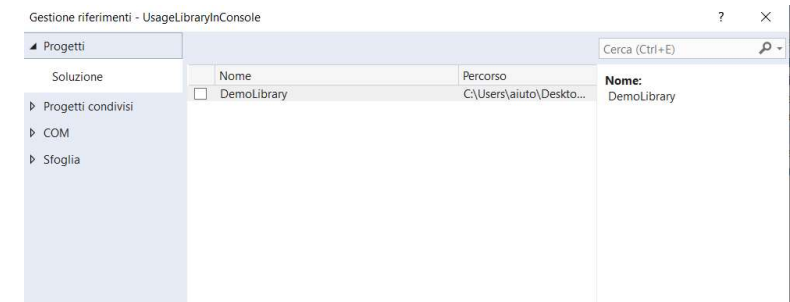
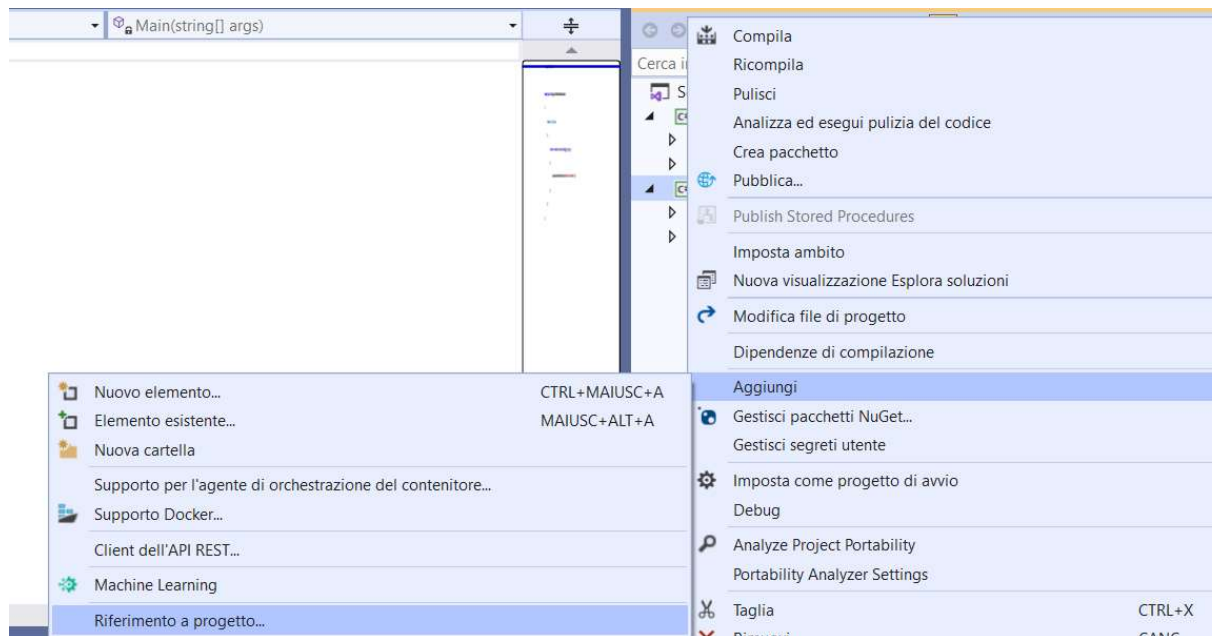
Si dovrà scegliere di creare una Console Application .NET Core



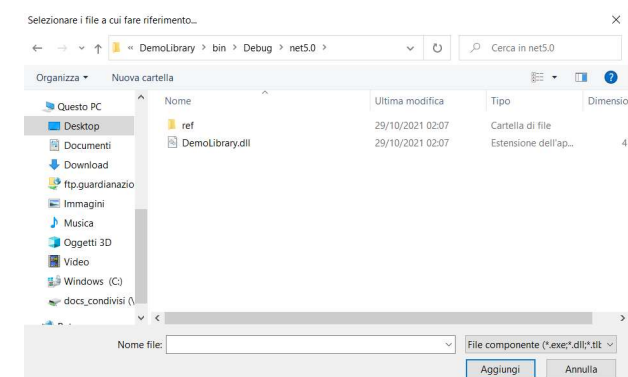
Nella slides successiva si selezionerà il progetto appena creato (Console Application UsageLibraryInConsole) e si selezionerà con l'uso del tasto destro su di esso la voce Aggiungi->Aggiungi riferimento

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



Si potrà selezionare direttamente dalla finestra DemoLibrary, oppure con il pulsante sfoglia andare a selezionare la libreria DLL compilata

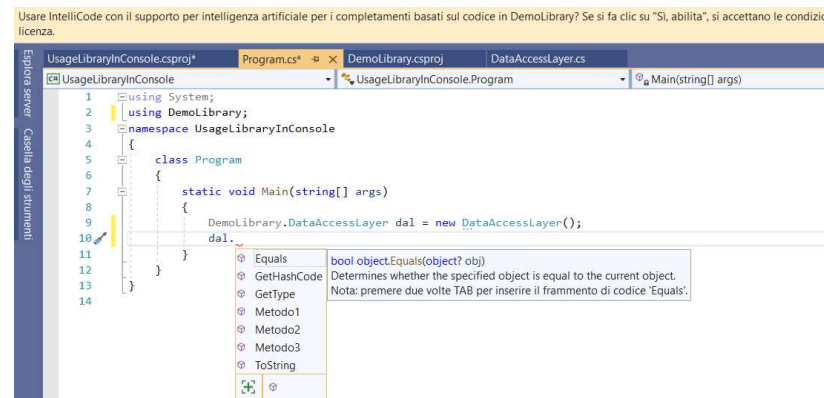


Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

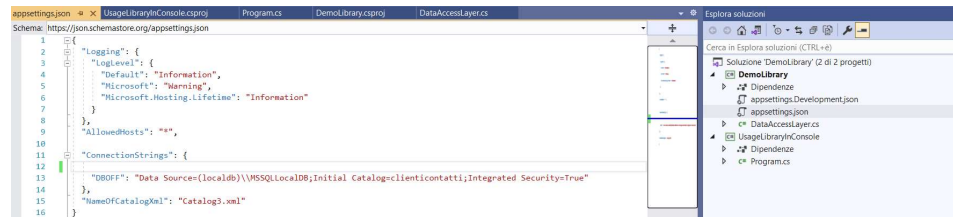


All'aggiunta di un riferimento alla libreria si potrà verificare come viene modificato Il file .csproj



Cloud Computing on Microsoft Azure Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

Si aggiunge un file .json alla libreria di classe



Per la corretta esecuzione del metodo della libreria bisogna vedere la prossima slides per referenziare correttamente nel .csproj della console application alcuni namespace importanti

```
public string ReadKeyFromJson(string key)
{
    var configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("appsettings.json", optional: false)
        .Build();

    var stringa = "ConnectionStrings:" + key;

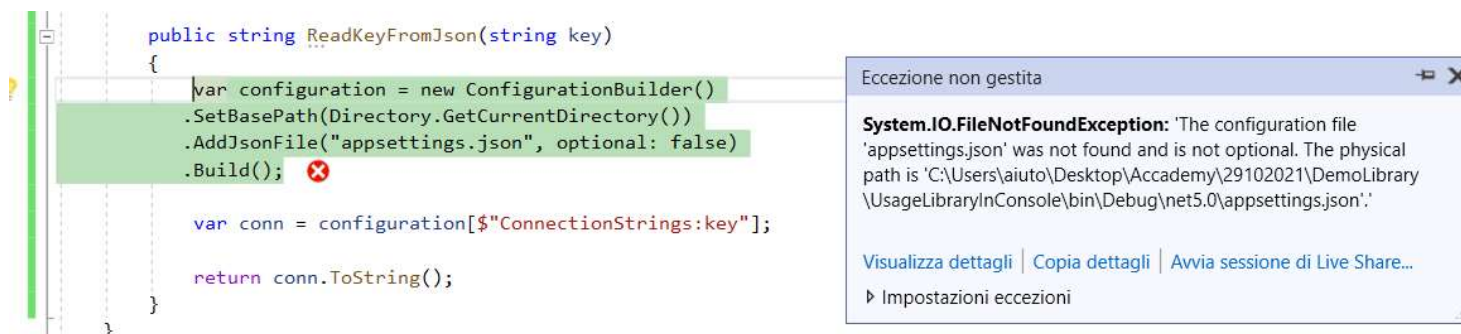
    var conn = configuration[stringa];

    return conn.ToString();
}
```


Cloud Computing on Microsoft Azure Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

```
<ItemGroup>
  <PackageReference Include="Microsoft.Extensions.Configuration.Abstractions" Version="5.0.0" />
  <PackageReference Include="Microsoft.Extensions.Configuration" Version="5.0.0" />
  <PackageReference Include="Microsoft.Extensions.Configuration.FileExtensions" Version="5.0.0" />
  <PackageReference Include="Microsoft.Extensions.Configuration.Json" Version="5.0.0" />
  <PackageReference Include="Microsoft.Extensions.Configuration.UserSecrets" Version="5.0.0" />
  <PackageReference Include="Microsoft.Extensions.DependencyInjection" Version="5.0.2" />
</ItemGroup>
```

Avviando la console Application si potrà verificare che dovrà risolversi ancora un errore in ultima analisi:



ACADEMY
.NET CORE / C#

```
Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=clienticontatti;Integrated Security=True
```

Dopo la soluzione dell'eccezione ecco l'output di successo
esecuzione

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



In alternativa alla pubblicazione in cloud, provare una volta terminata l'implementazione della libreria e aver creato un'applicazione ASP.Net Core, referenziando anche qui la libreria in questione, distribuirla sul Web Server IIS (Internet Information Services)

Modelli di progetto recenti

Applicazione console

C#

Libreria di classi

C#

App Web ASP.NET Core

C#

C#

Tutte le piattaforme

Desktop



App Windows Forms

Modello di progetto per la creazione di un'app Windows Forms (WinForms) .NET.

C# Windows Desktop



Applicazione WPF

Progetto per la creazione di un'applicazione WPF .NET Core

App Web ASP.NET Core

C#

Linux

macOS

Windows

Cloud

Servizio

Web

Nome del progetto

WebAspCoreUsageLibrary

Percorso

C:\Users\aiuto\Desktop\Accademy\29102021\DemoLibrary

App Web ASP.NET Core

C#

Linux

macOS

Windows

Cloud

Servizio

Web

Framework di destinazione

.NET 5.0 (Corrente)

Tipo di autenticazione

Nessuna

☒ Configura per HTTPS

☐ Abilita Docker

Sistema operativo Docker

Linux

☐ Enable Razor runtime compilation



ACADEMY
.NET CORE / C#

Cloud Computing on Microsoft Azure Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



La procedura di reference della libreria implementata dopo aver creato il progetto ASP.Net Core è la stessa vista per quella della console Application

Vi dovrete trovare nel file .csproj del progetto ASP.Net Core questa definizione

```
<Project Sdk="Microsoft.NET.Sdk.Web">
```

```
<PropertyGroup>
```

```
<TargetFramework>net5.0</TargetFramework>
```

```
</PropertyGroup>
```

```
<ItemGroup>
```

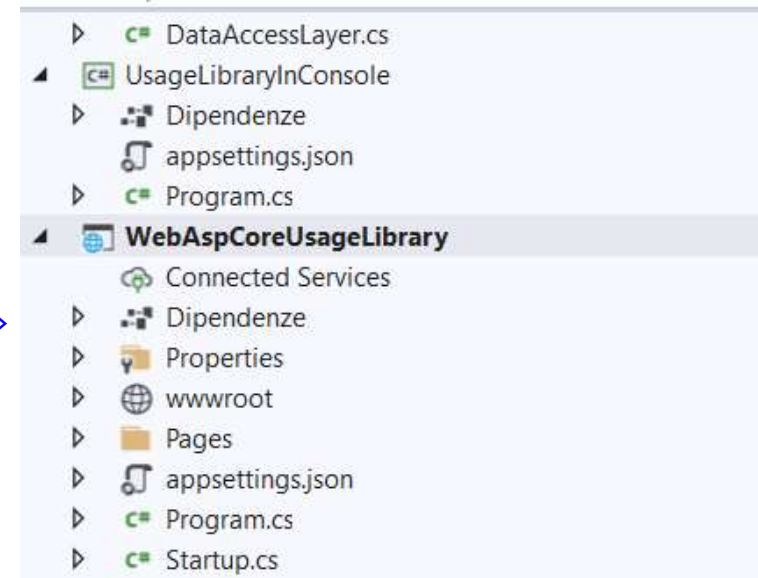
```
<Reference Include="DemoLibrary">
```

```
<HintPath>..\DemoLibrary\bin\Debug\net5.0\DemoLibrary.dll</HintPath>
```

```
</Reference>
```

```
</ItemGroup>
```

```
</Project>
```



ACADEMY

.NET CORE / C#

Cloud Computing on Microsoft Azure Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



Per provare il funzionamento della libreria ho creato nella Pages Index.cshtml

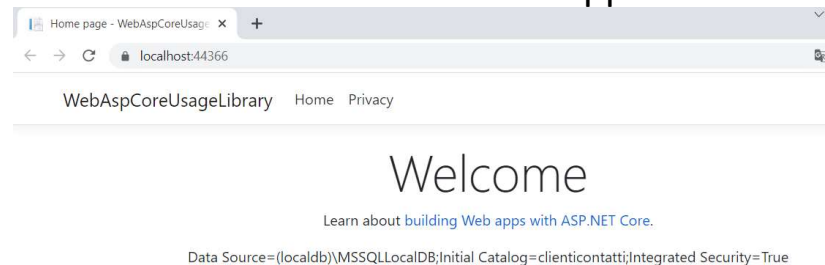
```
@page
@model IndexModel
@using DemoLibrary;
@{
    ViewData["Title"] = "Home page";

    DemoLibrary.DataAccessLayer dal = new DemoLibrary.DataAccessLayer();
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

```
1 @page
2 @model IndexModel
3 @using DemoLibrary;
4 @{
5
6     ViewData["Title"] = "Home page";
7
8     DemoLibrary.DataAccessLayer dal = new DemoLibrary.DataAccessLayer();
9
10    var conn = dal.ReadKeyFromJson("DBOFF");
11
12 }
13
14 <div class="text-center">
15     <h1 class="display-4">Welcome</h1>
16     <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building W
17     <p>@conn</p>
18 </div>
19
```

Domanda Per ottenere correttamente il seguente output, secondo voi bisognerebbe copiare il contenuto del file Appsettings.json anche nel file presente nativamente nella soluzione dell'applicazione ASP.Net Core?

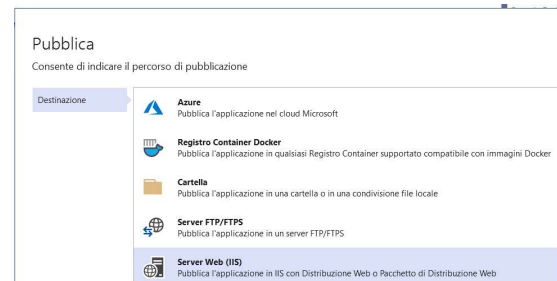
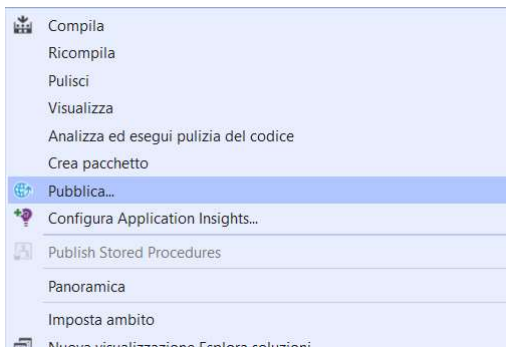


Cloud Computing on Microsoft Azure Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



La pubblicazione dell'applicazione ASP.Net Core ,creata nella slides precedente e che referencia la libreria di classe progettata per la piattaforma .NET Core, avviene seguendo questi passaggi:

- ❑ Lanciare l'ambiente di sviluppo come Amministratore
- ❑ Selezionare il progetto ASP.Net Core creato ed eseguire il wizard che viene lanciato con la funzione di Pubblica dal menu contestuale che appare al click del tasto sinistro del mouse. Seguire correttamente i suoi passaggi. Assicurarsi di eseguire la corretta installazione dei moduli di hosting condiviso sulla cartella GoogleDrive di Randstad



ACADEMY
.NET CORE / C#

Nella prossima slides vediamo i parametri di configurazione per una corretta pubblicazione sul web server

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

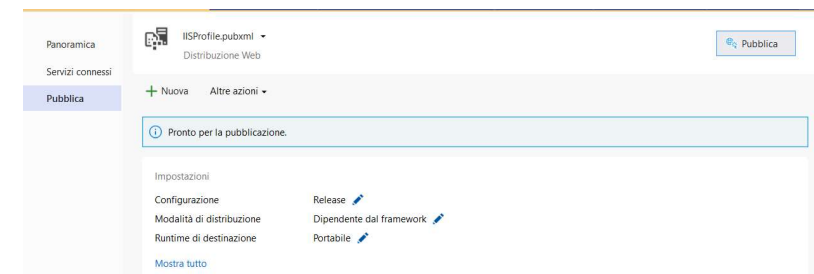


Pubblica

Configura la connessione al server Web (IIS)

Destinazione	Server
Destinazione specifica	localhost
Connessione IIS	Nome del sito Default Web Site/ASPCORELIB
	URL di destinazione http://localhost/ASPCORELIB
	Nome utente <input type="text"/>
	Password <input type="password"/>
	<input type="checkbox"/> Salva password

Si cliccherà con il pulsante Fine e si salverà un profile nella soluzione



Si seleziona Altre azioni->Modifica per assicurarsi che l'utility MS Deploy funziona correttamente su IIS

ACADEMY
.NET CORE / C#

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio

Pubblica

Connessione

Impostazioni

IISProfile

Metodo di pubblicazione: Distribuzione Web

Server: localhost

Nome sito: Default Web Site/ASPCORELIB

Nome utente:

Password:

☐ Salva password

URL di destinazione: http://localhost/ASPCORELIB

Convalida connessione

Pubblica

Connessione

Impostazioni

IISProfile

Configurazione: Release

Framework di destinazione: net5.0

Modalità di distribuzione: Dipendente dal framework

[Informazioni sulle modalità di distribuzione](#)

Runtime di destinazione: Portabile

Opzioni pubblicazione file

Database

DBOFF

☐ Usa questa stringa di connessione in fase di esecuzione

Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=clienticontatti;Integrated Security=SSPI

Migrazioni Entity Framework

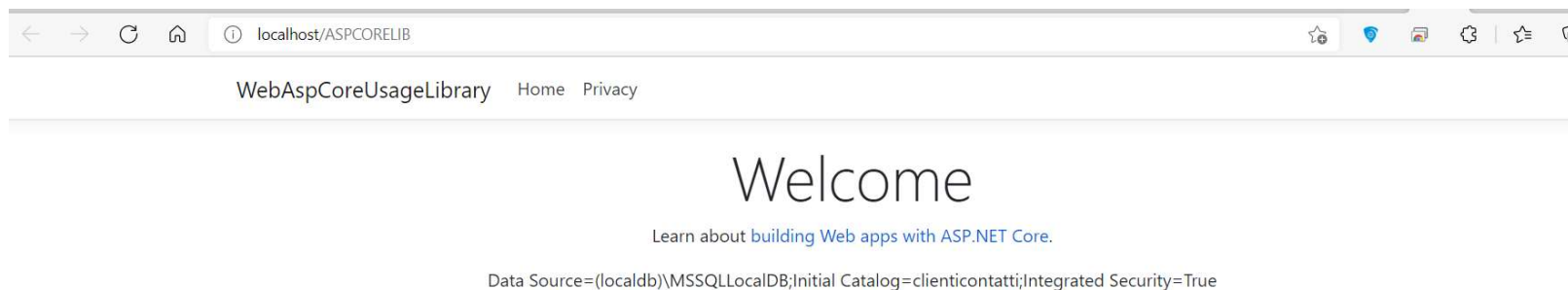
Se si dovesse fare il detect nel file appsettings.json una connection strings
La pubblicazione lo rileva e ci da dei comportamenti che vedremo analiticamente
nel corso delle ultime sessioni

Cloud Computing on Microsoft Azure

Esercitazione – Creazione di una libreria di classe – Demo con Visual Studio



Si clicca finalmente sul pulsante Pubblicazione per avviare la pubblicazione del progetto su IIS:



Con questa immagine di output dovrete essere in grado di svolgere le attività di implementazione dell'esercitazione dalla slide numero 156



ACADEMY
.NET CORE / C#