

C#

Programming Language Fundamentals



Prerequisites

- Computer and its basic knowledge.
- Integrated Development Environment or Code Editor
 - Visual Studio (Recommended)
 - Visual Studio Code

Overview

- Introduction
- Our First C# program
- .NET Overview?
- IDE – Visual Studio quick tour
- C# syntax
- Variables and Data Types
- Type Conversions
- Conditionals
- Loops
- Classes
- Object Oriented Programming
- Value Types vs Reference Types
- Compilation and CLR
- Assemblies and Referencing
- Exceptional Handling

Open Source

Cross Platform



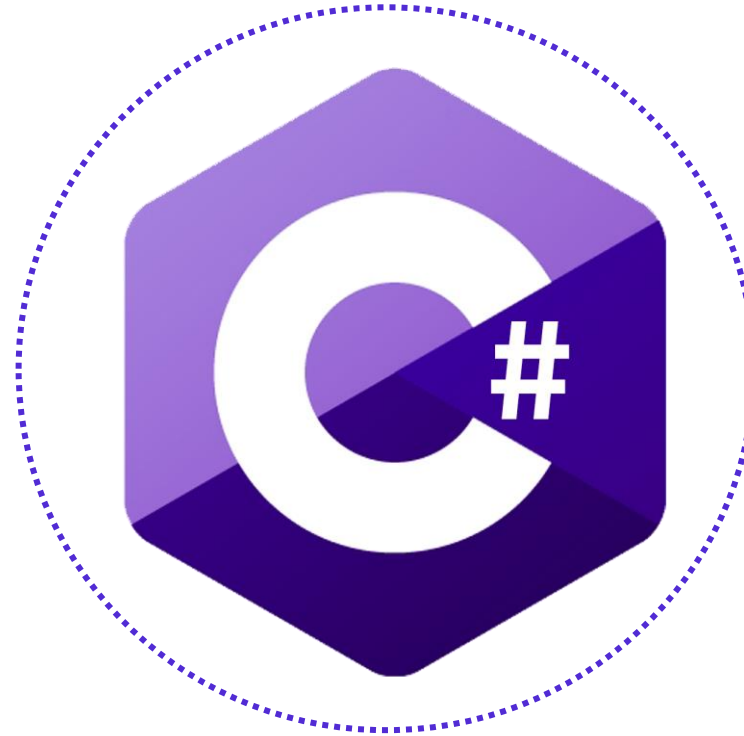
Object Oriented Programming

Type Safety

Automatic Memory Management

Exceptional Handling

Microsoft



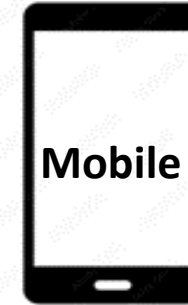
Web



Desktop



Mobile



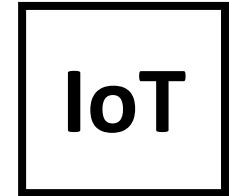
Games



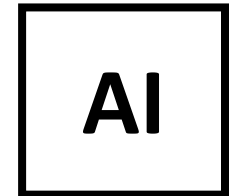
Cloud



IoT



AI

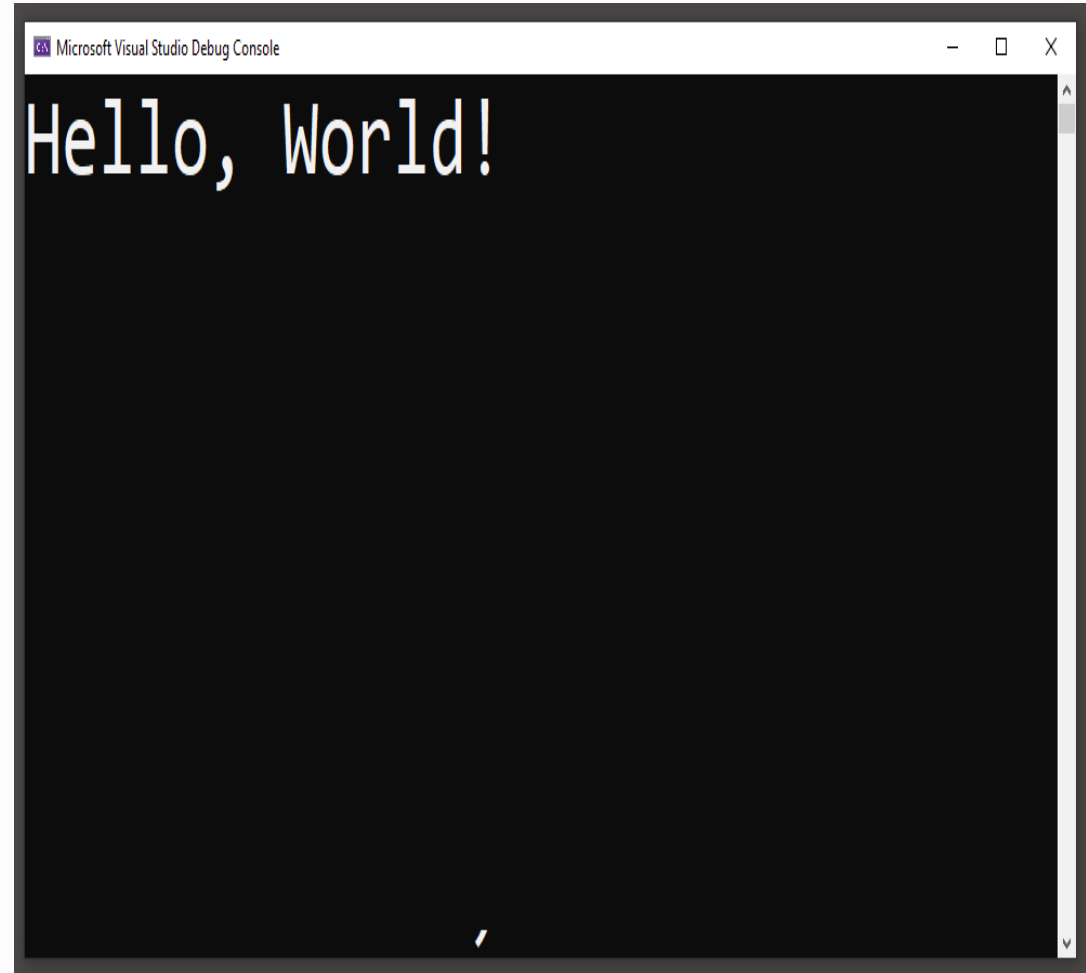


Our First C# Program



```
namespace HelloWorldCSharp
{
    0 references | 0 changes | 0 authors, 0 changes
    internal class Program
    {
        0 references | 0 changes | 0 authors, 0 changes
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

RUN



Console Application --name Hello World

C# Source Code Execution

C# Source Code

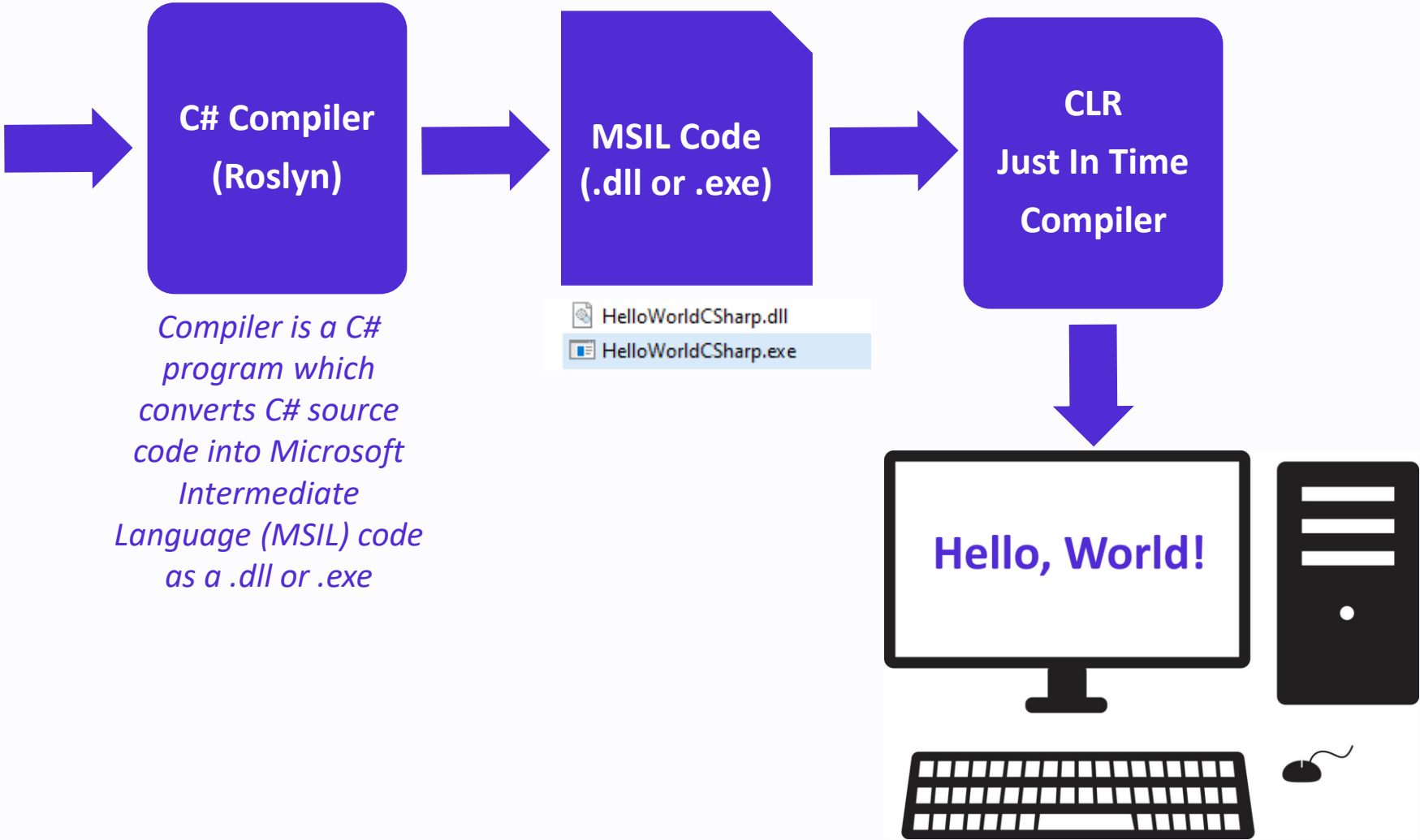
```

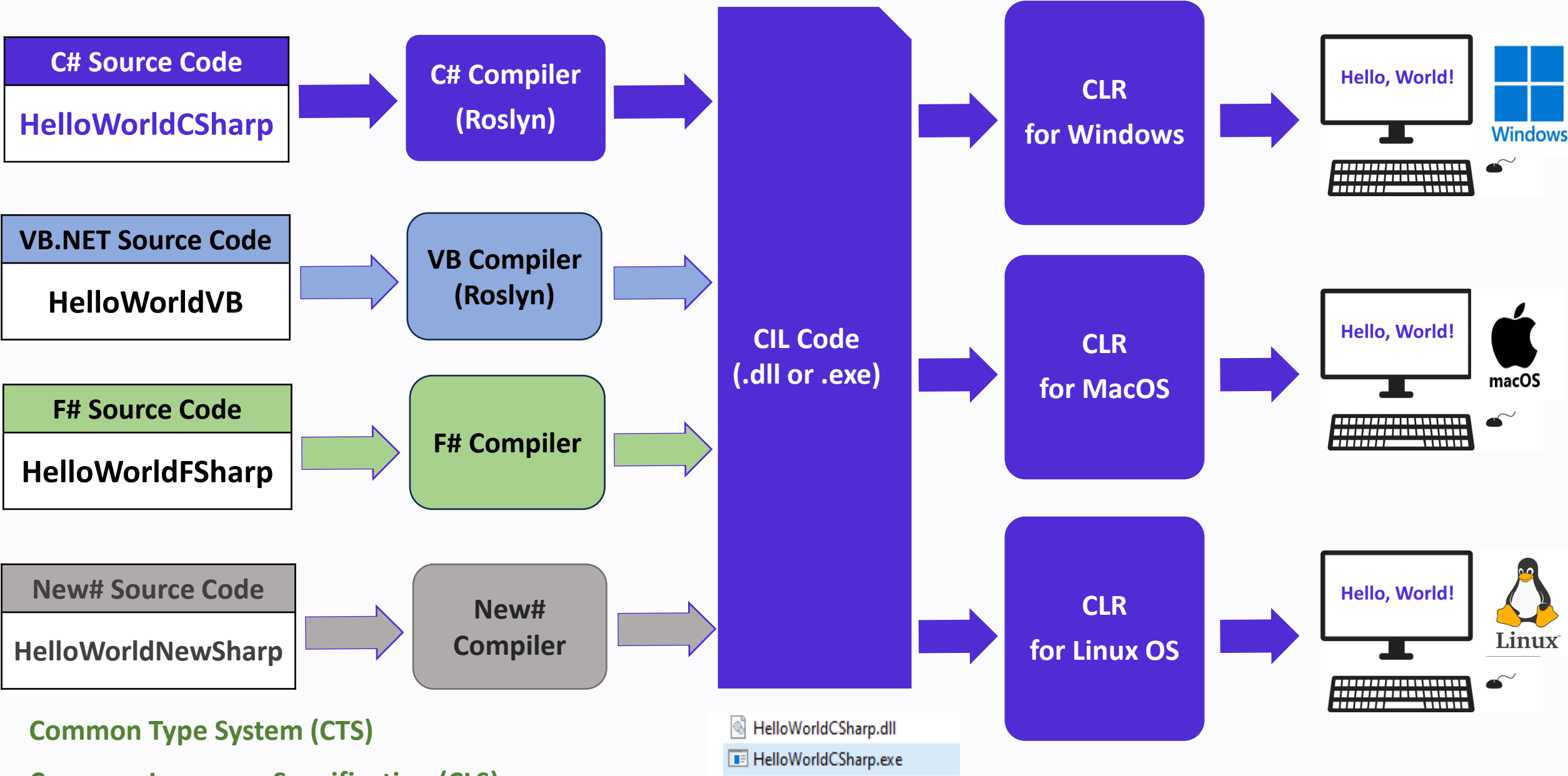
HelloWorldCSharp
└─ Dependencies
  └─ Program.cs

namespace HelloWorldCSharp
{
    0 references | 0 changes | 0 authors, 0 changes
    internal class Program
    {
        0 references | 0 changes | 0 authors, 0 changes
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}

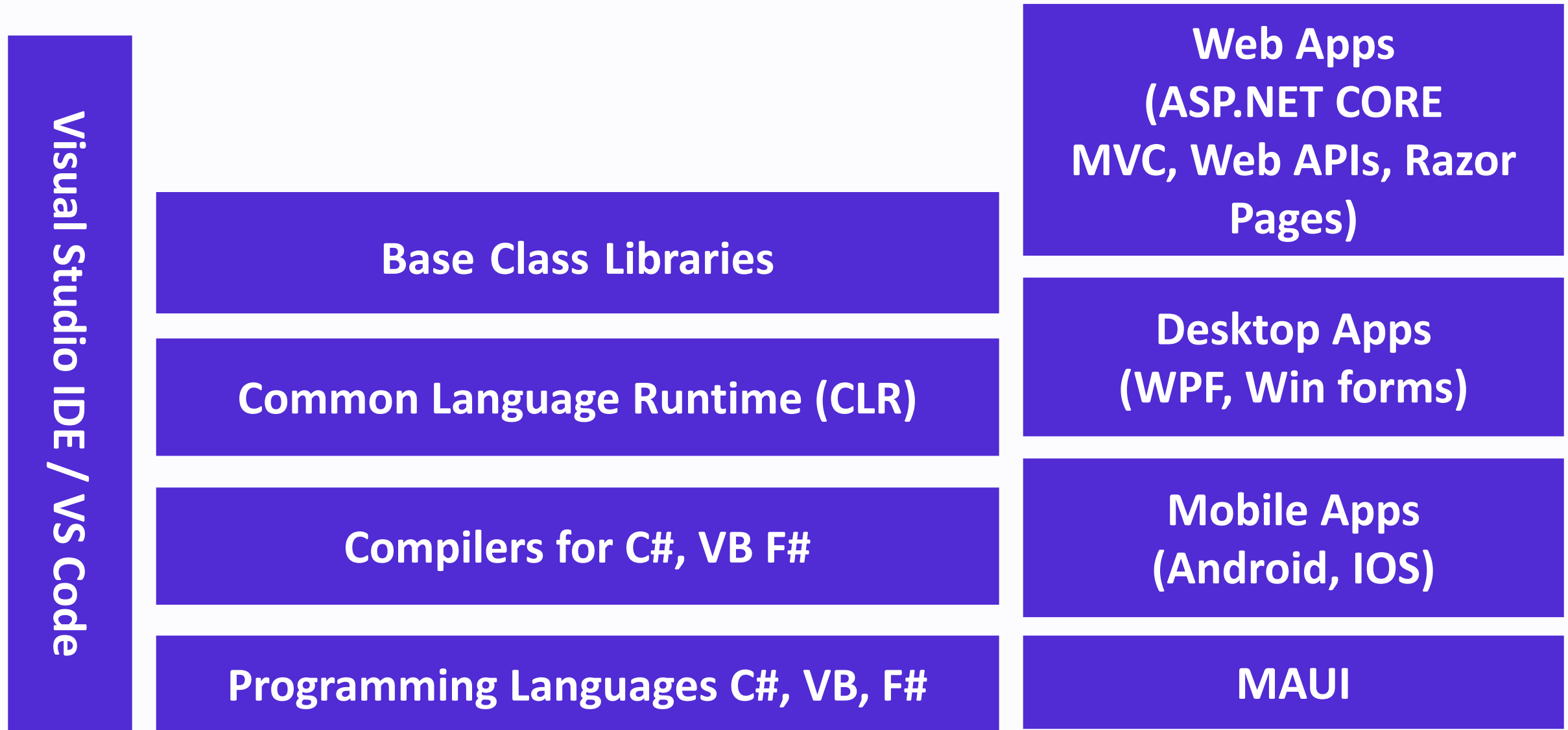
```

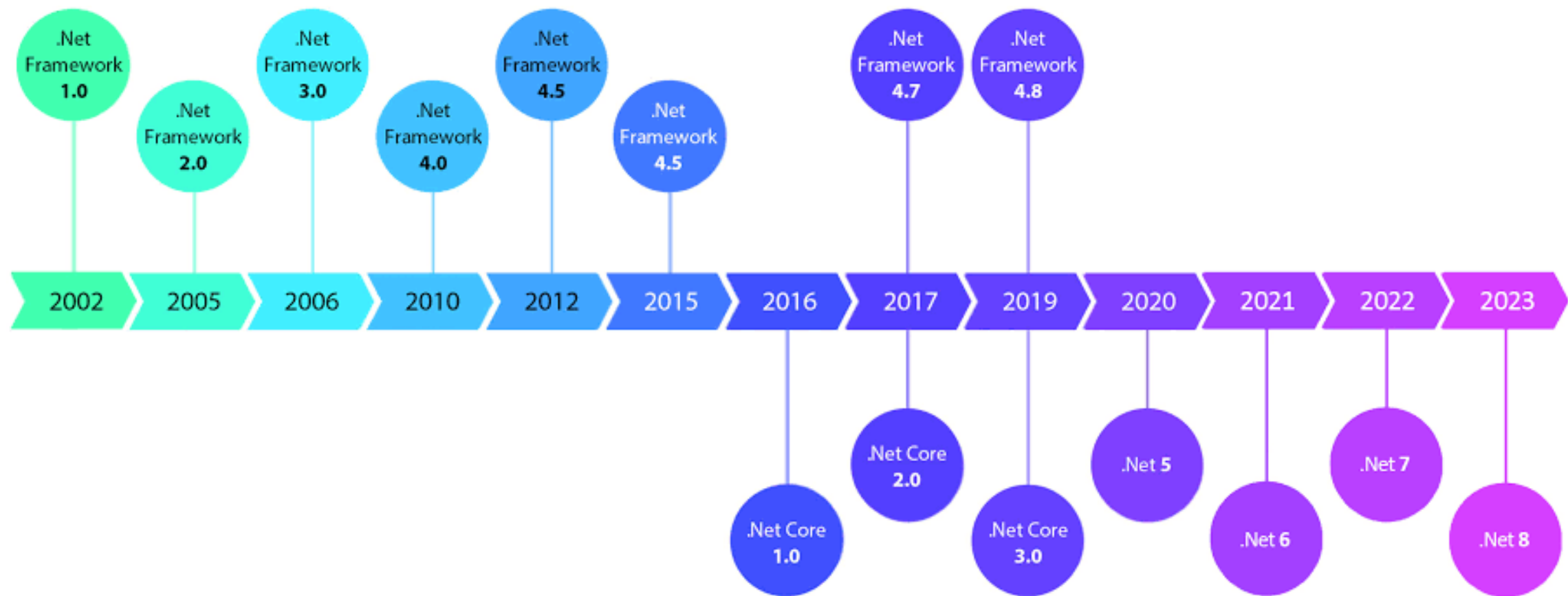
Console Application





.NET Ecosystem and its tools





Version History

| Target | Version | C# language version default |
|----------------|---------|-----------------------------|
| .NET | 8.x | C# 12 |
| .NET | 7.x | C# 11 |
| .NET | 6.x | C# 10 |
| .NET | 5.x | C# 9.0 |
| .NET Core | 3.x | C# 8.0 |
| .NET Core | 2.x | C# 7.3 |
| .NET Standard | 2.1 | C# 8.0 |
| .NET Standard | 2 | C# 7.3 |
| .NET Standard | 1.x | C# 7.3 |
| .NET Framework | all | C# 7.3 |

[illegible]

A diagram of a database cylinder, colored blue with a lighter blue top and bottom. Inside the cylinder, there are 15 white rounded rectangular boxes, each containing the word "Table" in blue text. The boxes are arranged in a 5x3 grid.

| | | |
|-------|-------|-------|
| Table | Table | Table |
| Table | Table | Table |
| Table | Table | Table |
| Table | Table | Table |
| Table | Table | Table |

C# Assembly

namespace: MyApp.User

Class

Class

Class

Class

Class

Class

namespace: MyApp.Admin

Class

Class

Class

Class

Class

Class

MS SQL Server Database

Schema: dbo User

Table

Table

Table

Table

Table

Table

Schema: Admin

Table

Table

Table

Table

Table

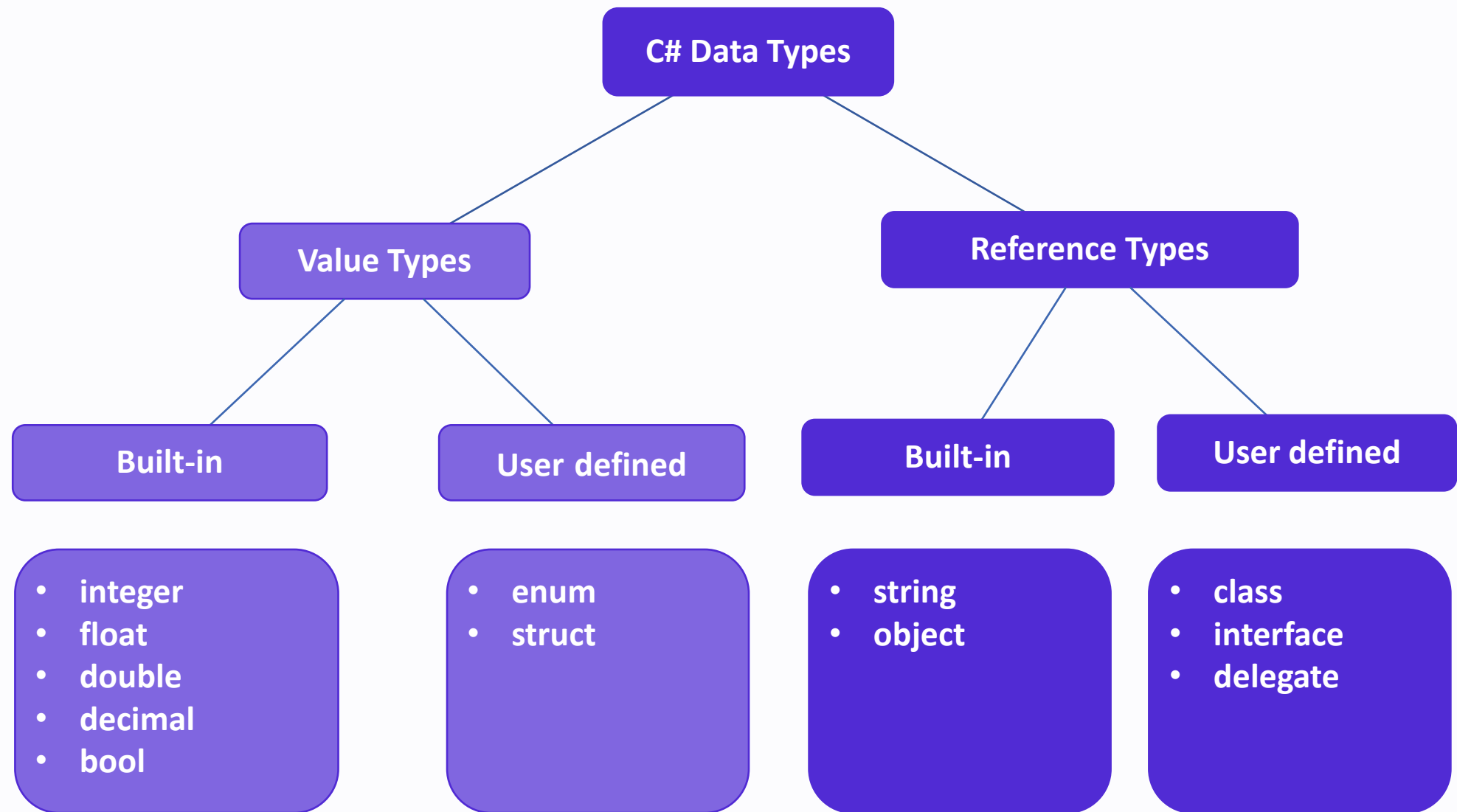
Table

Keywords

| | | | | | |
|----------|----------|-----------|------------|-----------|----------|
| abstract | delegate | if | override | struct | volatile |
| as | do | implicit | params | switch | while |
| base | double | in | private | this | |
| bool | else | int | protected | throw | |
| break | enum | interface | public | true | |
| byte | event | internal | readonly | try | |
| case | explicit | is | ref | typeof | |
| catch | extern | lock | return | uint | |
| char | false | long | sbyte | ulong | |
| checked | finally | namespace | sealed | unchecked | |
| class | fixed | new | short | unsafe | |
| const | float | null | sizeof | ushort | |
| continue | for | object | stackalloc | using | |
| decimal | foreach | operator | static | virtual | |
| default | goto | out | string | void | |

Contextual Keywords

| | | |
|------------|---------|-----------|
| add | group | record |
| and | init | remove |
| alias | into | required |
| ascending | join | scoped |
| args | let | select |
| async | managed | set |
| await | nameof | unmanaged |
| by | nint | value |
| descending | not | var |
| dynamic | notnull | when |
| equals | nuint | where |
| File | on | with |
| from | or | yield |
| get | orderby | |
| global | partial | |



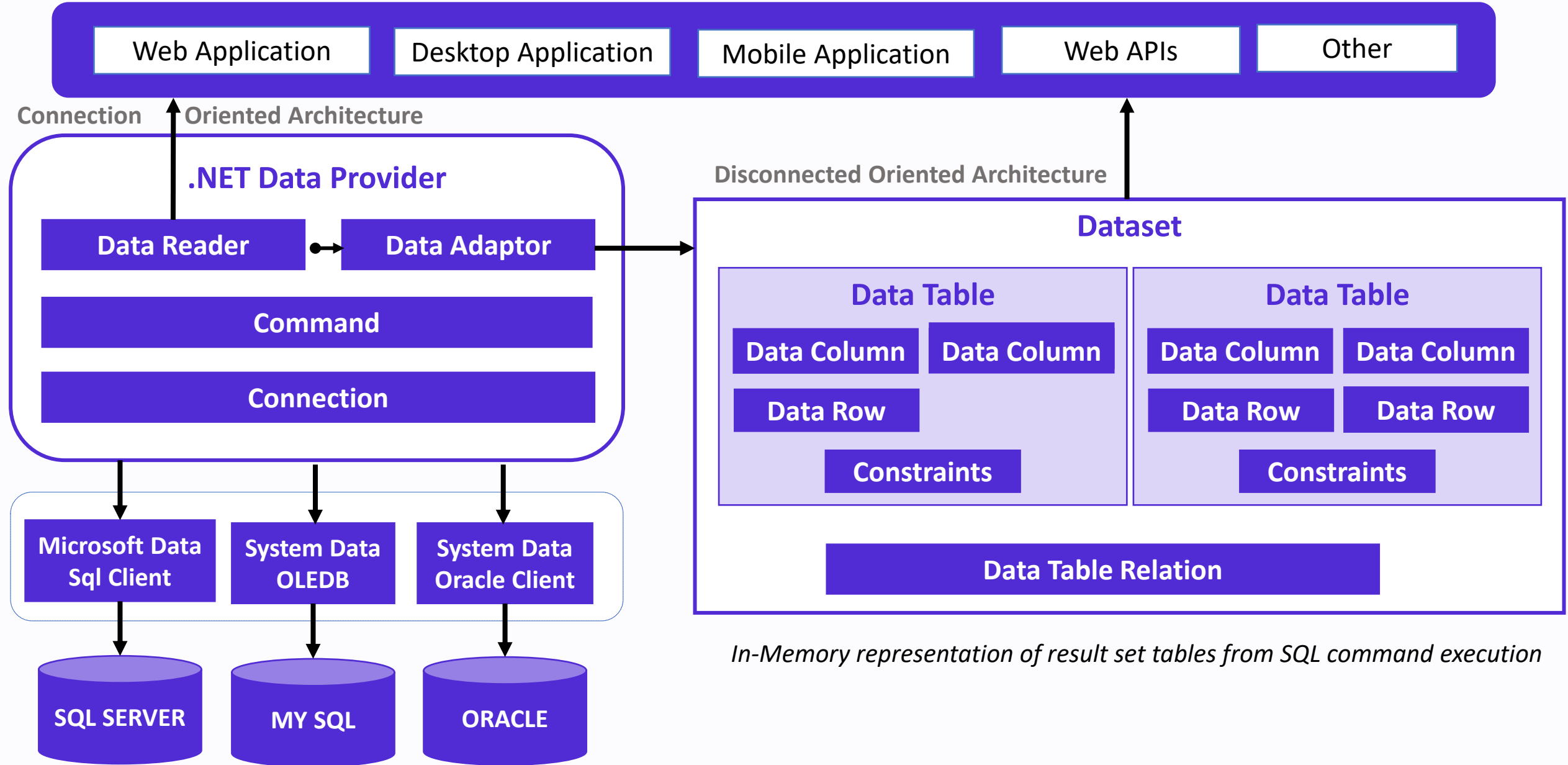
Data Types

| Data Type | Size | Description |
|-------------------------------|-----------------------|--|
| Integer | | |
| sbyte | 8 bits | -2^7 to 2^7-1 |
| int | 32 bits | -2^{31} to $2^{31}-1$ (-2,147,483,648 to 2,147,483,647) |
| short | 16 bits | -2^{15} to $2^{15}-1$ |
| long | 64 bits | -2^{63} to $2^{63}-1$ (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807) |
| Real Or Floating Point | | |
| float | 32 bits | $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ (7 decimal digits). Suffix: f |
| double | 64 bits | $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ (15 decimal digits). Suffix: D |
| decimal | 128 bits | $\pm 1.0 \times 10^{-28}$ to $\pm 7.9228 \times 10^{28}$. Suffix: M |
| bool | 1 bit | true or false |
| Free Text | | |
| char | 2 bytes | '@' |
| string | 2 bytes per character | "Hello". Size = (2* no of characters in the sequence) bytes. |

ADO.NET – Data Access API

“ADO.NET is a data access api in .NET Platform to interact with different data sources such as databases (sql server, oracle, etc.), xml, Microsoft access, and other in a standard, and structured approach.”

ADO.NET Architecture



MS SQL SERVER Data Access

- Add below DLL as project reference through manage NuGet package manager.
 - **Microsoft.Data.SqlClient**
 - **System.Data.SqlClient (Legacy library)**
- SQL Server data provider provides the following classes to interact with database.

| Class | Description |
|-----------------------|---|
| SqlConnection | Establishes a connection to a database. |
| SqlCommand | Represents an individual SQL statement or stored procedure that can be executed against the database connected. |
| SqlDataReader | Provides read-only, forward-only access to the data in a database. |
| SqlDataAdapter | Acts as a bridge between the command and connection objects and a dataset |

UML Diagram

SqlConnection

+ConnectionString : string

+ Open () : void

+ Close () : void

SqlCommand

+ Connection : SqlConnection

+ CommandType : Text or SP

+ CommandText : Query or SP

+ Parameters : SqlParameter[]

+ ExecuteNonQuery () : int

+ ExecuteScalar () : object

+ ExecuteReader () : data reader

SqlDataReader

+ indexer : object

+ FieldCount : int

+ Read () : bool

+ Close () : void

SqlDataAdapter

+ SelectCommand : SqlCommand

+ InsertCommand : SqlCommand

+ UpdateCommand : SqlCommand

+ DeleteCommand : SqlCommand

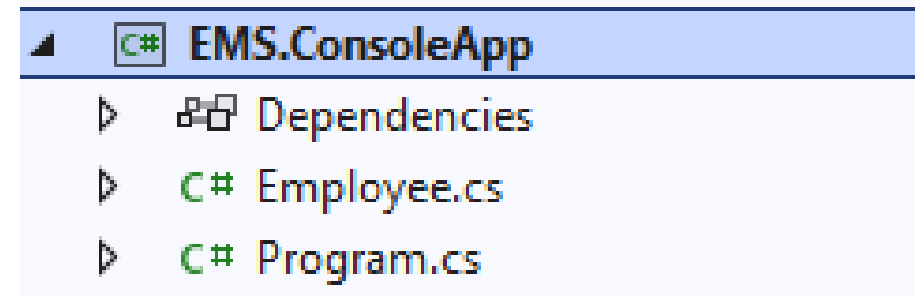
+ Fill () : int

+ update () : int

EMS Project v1

```
C# D:\naveenc\Practices\2024\Console\CSharpFundamentalsSln\EMS.App\bin\Deb...
Employee Management System
-----
Type your choice and press enter.
(1) Add (2) Update (3) Display Single Employee (4) Export All (5) Delete

Enter your choice and press enter:
```



EMS Project v2

```
D:\naveenc\Practices\2024\Console\CSharpFundamentalsSln\EMS.App\bin\Deb...
Employee Management System
-----
Type your choice and press enter.
(1) Add (2) Update (3) Display Single Employee (4) Export All (5) Delete

Enter your choice and press enter:
```

Employee

Employee Management System

First Name

Last Name

Employee Code

| | EmployeeCode | FirstName | LastName | Department |
|---|--------------|------------|-------------|------------|
| ▶ | EMS1 | Naveen | Chittimalla | Developer |
| | EMS2 | Abhiram | Pittampally | Developer |
| | EMS3 | Naveen | Ch | |
| | EMS4 | Test | Test | |
| | EMS5 | New Emp 5 | New Emp 5 | |
| | EMS6 | Test Emp 6 | Test Emp 6 | |
| | EMS7 | EMp 7 | EMp 7 | |

Clear

Save

Scheduler

Recurring job to fetch all the employees from database and save them in a csv file and place the file at a specified location.

EMS Project v2 Solution 1

EMS.ConsoleApp

- Dependencies
- C# Employee.cs
- C# Program.cs

EMS.DesktopApp

- Dependencies
- C# Employee.cs
- EmployeeForm.cs
- C# Program.cs

EMS.Scheduler

- Dependencies
- C# Employee.cs
- C# Program.cs

D:\naveenc\Practices\2024\Console\CSHarpFundamentalsSln\EMS.App\bin\Deb...

```
Employee Management System
-----
Type your choice and press enter.
(1) Add (2) Update (3) Display Single Employee (4) Export All (5) Delete

Enter your choice and press enter:
```

Employee

Employee Management System

First Name

Last Name

Employee Code

| | EmployeeCode | FirstName | LastName | Department |
|---|--------------|------------|-------------|------------|
| ▶ | EMS1 | Naveen | Chittimalla | Developer |
| | EMS2 | Abhiram | Pittampally | Developer |
| | EMS3 | Naveen | Ch | |
| | EMS4 | Test | Test | |
| | EMS5 | New Emp 5 | New Emp 5 | |
| | EMS6 | Test Emp 6 | Test Emp 6 | |
| | EMS7 | EMp 7 | EMp 7 | |

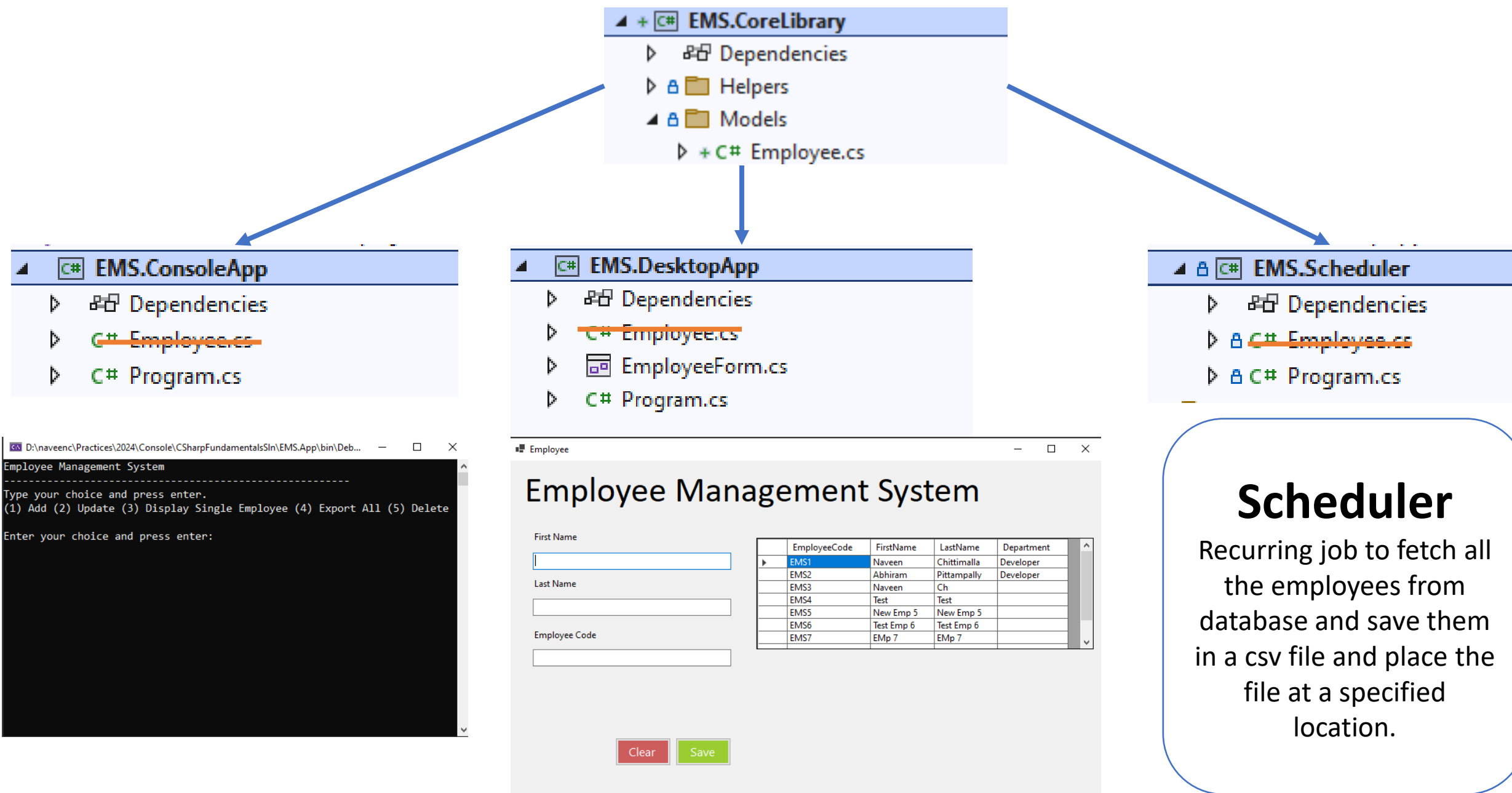
Clear

Save

Scheduler

Recurring job to fetch all the employees from database and save them in a csv file and place the file at a specified location.

EMS Project v2 Solution 2



Scheduler

Recurring job to fetch all the employees from database and save them in a csv file and place the file at a specified location.

C# Assembly

- Assembly is a collection of types such as namespaces, classes, interfaces, enums and resources that are built to work together and form a logical unit of functionality.
- Assembly can be a DLL or EXE based on the project type template that we choose.
 - **Class Library** is a collection of classes and namespaces in C# without any entry point method like Main. **Output Type is .dll**
 - **Console App** is an application that takes input and displays output at a command line console and behaves as an app host to run .dll and which has an entry point method like Main. **Output Type is .exe and .dll file**

namespace: MyApp.User

Class

Class

Class

interface

struct

enum

namespace: MyApp.Admin

Class

Class

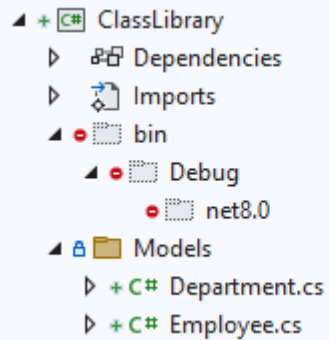
Class

interface

struct

enum

Class Library



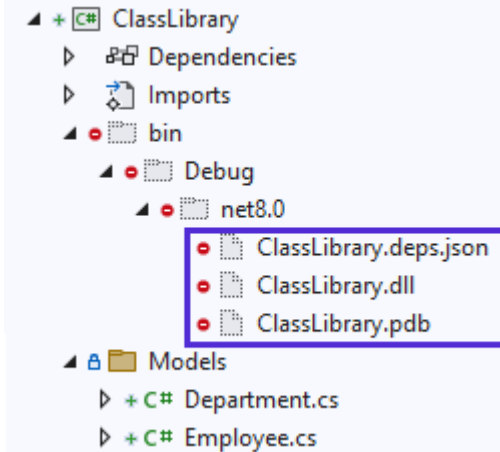
ClassLibrary

- Dependencies
- Imports
- bin
 - Debug
 - net8.0
- Models
 - Department.cs
 - Employee.cs

C# Compiler

Build the Class Library project, then C# compiler converts C# code into MSIL code and packages all the types into .dll file

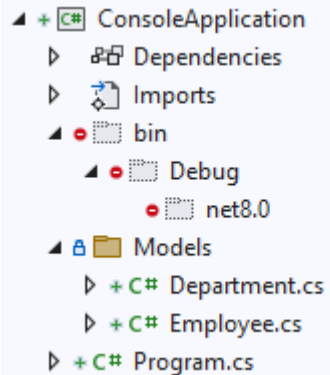
.dll (Dynamic Link Library)



ClassLibrary

- Dependencies
- Imports
- bin
 - Debug
 - net8.0
 - ClassLibrary.deps.json
 - ClassLibrary.dll
 - ClassLibrary.pdb
- Models
 - Department.cs
 - Employee.cs

Console Application



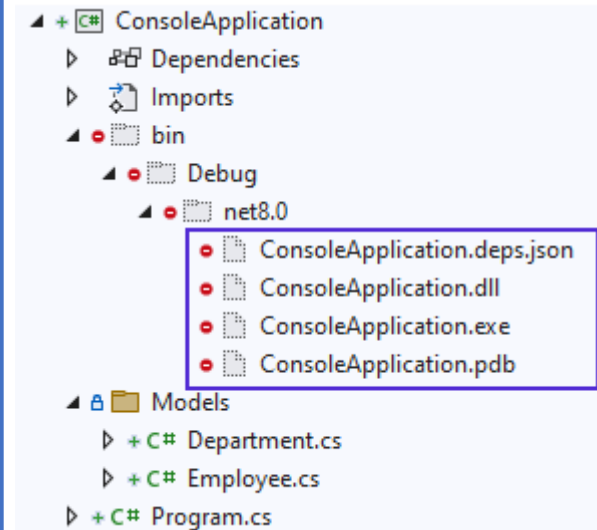
ConsoleApplication

- Dependencies
- Imports
- bin
 - Debug
 - net8.0
- Models
 - Department.cs
 - Employee.cs
 - Program.cs

C# Compiler

Build or Run the Console App project, then C# compiler converts C# code into MSIL code and generates .exe and .dll files

.exe (Executable)



ConsoleApplication

- Dependencies
- Imports
- bin
 - Debug
 - net8.0
 - ConsoleApplication.deps.json
 - ConsoleApplication.dll
 - ConsoleApplication.exe
 - ConsoleApplication.pdb
- Models
 - Department.cs
 - Employee.cs
 - Program.cs

Let's Write Code

EMS Project v3

```
D:\naveenc\Practices\2024\Console\CSharpFundamentals\Sln\EMS.App\bin\Deb...
Employee Management System
-----
Type your choice and press enter.
(1) Add (2) Update (3) Display Single Employee (4) Export All (5) Delete
Enter your choice and press enter:
```

Employee Management System

First Name

Last Name

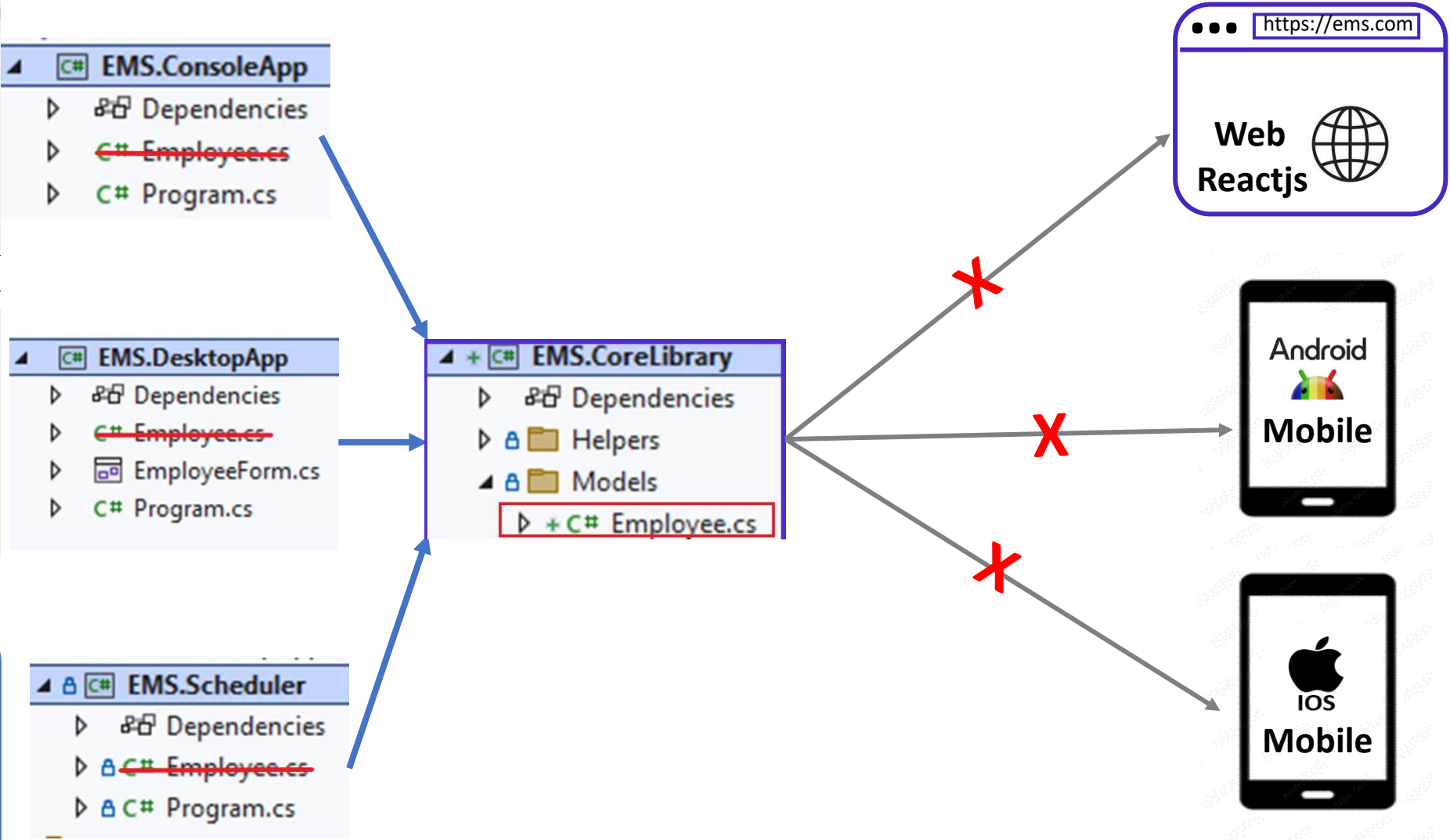
Employee Code

| EmployeeCode | FirstName | LastName | Department |
|--------------|------------|-------------|------------|
| EMS1 | Naveen | Chittimalla | Developer |
| EMS2 | Abhiram | Pittampally | Developer |
| EMS3 | Naveen | Ch | |
| EMS4 | Test | Test | |
| EMS5 | New Emp 5 | New Emp 5 | |
| EMS6 | Test Emp 6 | Test Emp 6 | |
| EMS7 | Emp 7 | Emp 7 | |

Clear Save

Scheduler

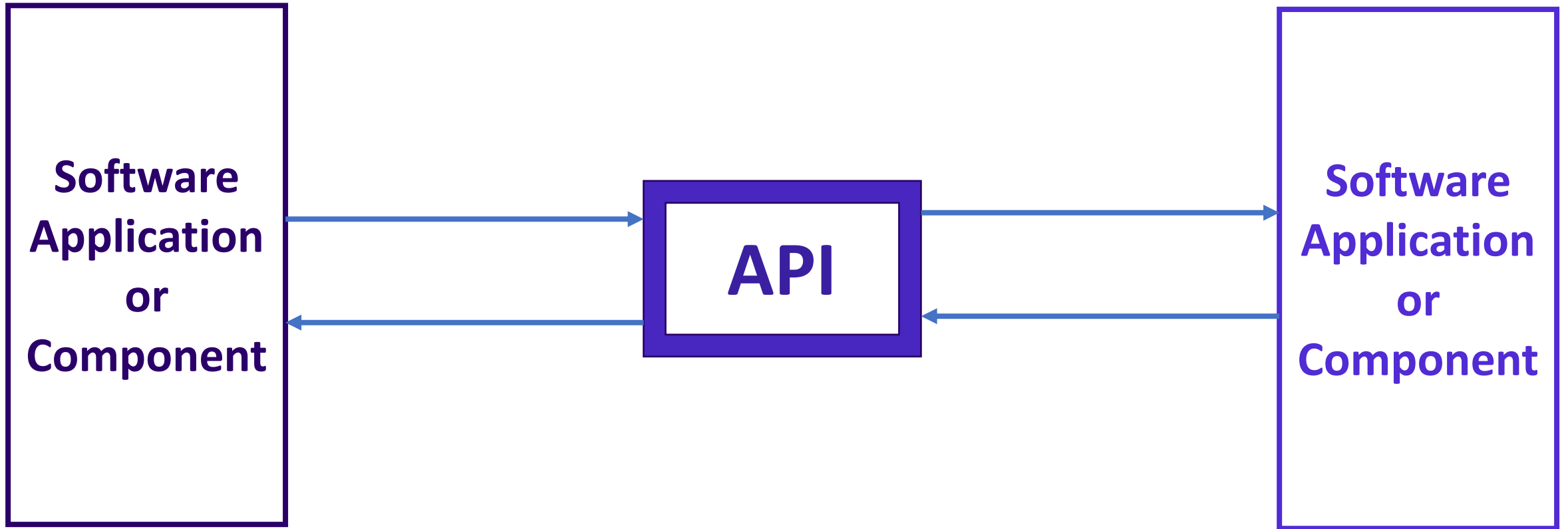
Generate a csv file and place the file at a specified location.



X C# not compatible

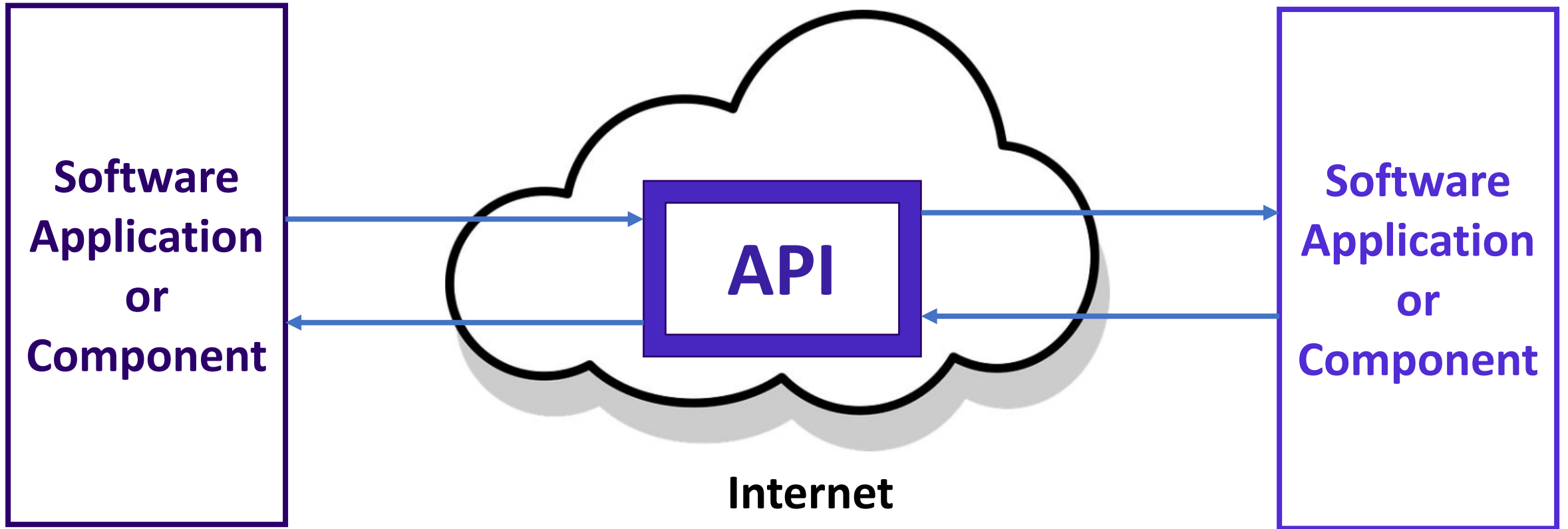
What is API ?

“API stands for Application Programming interface, is a set of routines, protocols, and tools for building software applications and communicate with each other.”

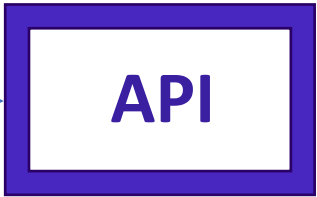
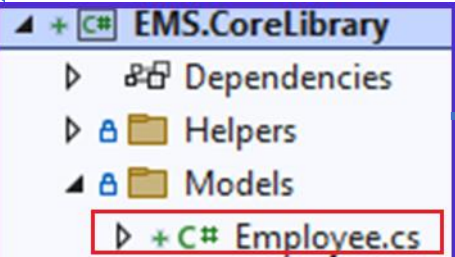
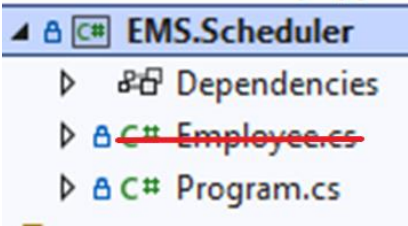
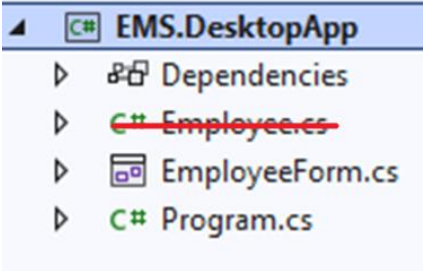
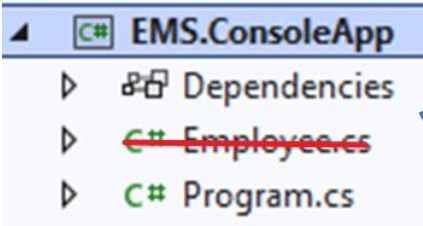
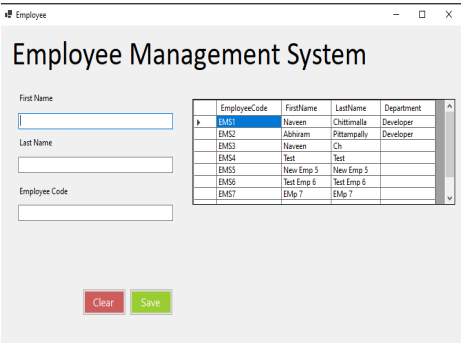
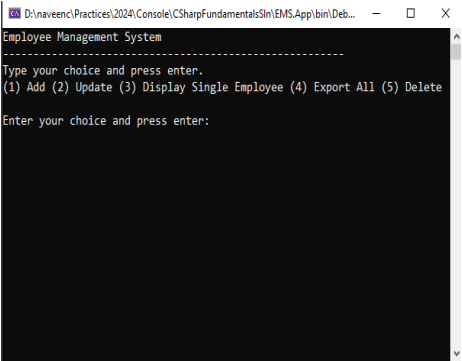


What is Web APIs ?

“Web APIs are simply the APIs that communicate over the internet.”



EMS Project v3



API which is accessible and understandable by all application



Scheduler

Generate a csv file and place the file at a specified location.

Thank You!

using statement

- **using** block can be used on the objects or instances of the classes which inherits from IDisposable class and implements Dispose method.
- **using** block does ensure that the object Dispose method will always be invoked, no matter if an exception is thrown or not.
- Dispose is a method used to clean up resources. In the case of a DB connection, the connection is released or closed, which is important.

Note: The equivalent of using is a try finally, which includes a call to Dispose within the finally block.

```
4 references | 0 changes | 0 authors, 0 changes
public class MyDisposableClass : IDisposable
{
    1 reference | 0 changes | 0 authors, 0 changes
    public void Dispose()
    {
        //Clean up any unmanagable resources
    }
}
```

```
0 references | 0 changes | 0 authors, 0 changes
public class DisposeDemo
{
    0 references | 0 changes | 0 authors, 0 changes
    public void UsingStatementDemoRun()
    {
        using (MyDisposableClass myDisposableClassObjectInstance = new())
        {
            // Implement logic
        }
        // using statement ensures to invoke Dispose method of MyDisposableClass,
        // even if an exception is thrown.
    }
}
```

using statement

4 references | 0 changes | 0 authors, 0 changes

```
public class MyDisposableClass : IDisposable
{
    1 reference | 0 changes | 0 authors, 0 changes
    public void Dispose()
    {
        //Clean up any unmanagable resources
    }
}
```

0 references | 0 changes | 0 authors, 0 changes

```
public class DisposeDemo
{
    0 references | 0 changes | 0 authors, 0 changes
    public void UsingStatementDemoRun()
    {
        using (MyDisposableClass myDisposableClassObjectInstance = new())
        {
            // Implement logic
        } // using statement ensures to invoke Dispose method of MyDisposableClass,
        // even if an exception is thrown.
    }
}
```

0 references | 0 changes | 0 authors, 0 changes

```
public void TryCatchFinallyDemoRun()
{
    MyDisposableClass myDisposableClassObjectInstance = new();
    try
    {
        // logic to be Implemented
    }
    catch (Exception)
    {
        throw;
    }
    finally
    {
        myDisposableClassObjectInstance.Dispose();
    }
}
```

using statement example

0 references | 0 changes | 0 authors, 0 changes

```
public class SqlConnectionDisposeDemo
```

```
{
```

0 references | 0 changes | 0 authors, 0 changes

```
public void UsingStatementDemoRun()
```

```
{
```

```
    using (Microsoft.Data.SqlClient.SqlConnection connection = new())
```

```
    {
```

```
        connection.ConnectionString = "Data Source=.\sqlexpress;Initial Catalog=DisposeDemo;Integrated Security=True;I
```

```
        connection.Open();
```

```
    } // using statement ensures to invoke Dispose method of SqlConnection,
```

```
    // which internally invokes Close() method of SqlConnection object,
```

```
    // even if an exception is thrown.
```

```
}
```

0 references | 0 changes | 0 authors, 0 changes

```
public void TryCatchFinallyDemoRun()
```

```
{
```

```
    Microsoft.Data.SqlClient.SqlConnection connection = new();
```

```
    try
```

```
    {
```

```
        // Implement logic
```

```
    }
```

```
    catch (Exception)
```

```
    {
```

```
        throw;
```

```
    }
```

```
    finally
```

```
    {
```

```
        if (connection.State == ConnectionState.Open)
```

```
        {
```

```
            connection.Close();
```

```
            //connection.Dispose(); |
```

```
        }
```

```
    }
```

```
}
```

```
}
```

Normal Template #512BD4

Normal Template #512BD4

Dotnet Deck Template #3A20A0

Dotnet Deck Template #3A20A0

Dotnet Deck Template Font #4826C0

Dotnet Deck Template Font #4826C0

C# Logo #2A0066

C# Logo #2A0066