

# 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



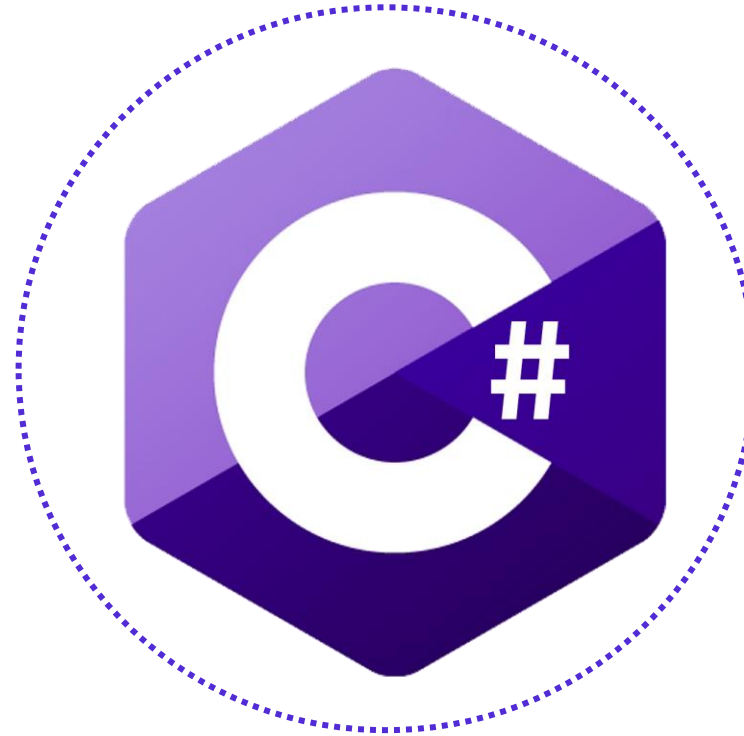
Object Oriented Programming

Type Safety

Automatic Memory Management

Exceptional Handling

# Microsoft



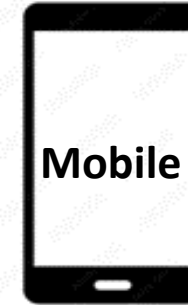
Web



Desktop



Mobile



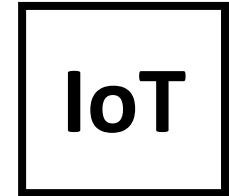
Games



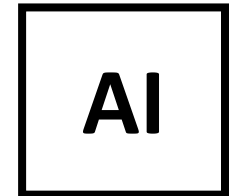
Cloud



IoT



AI

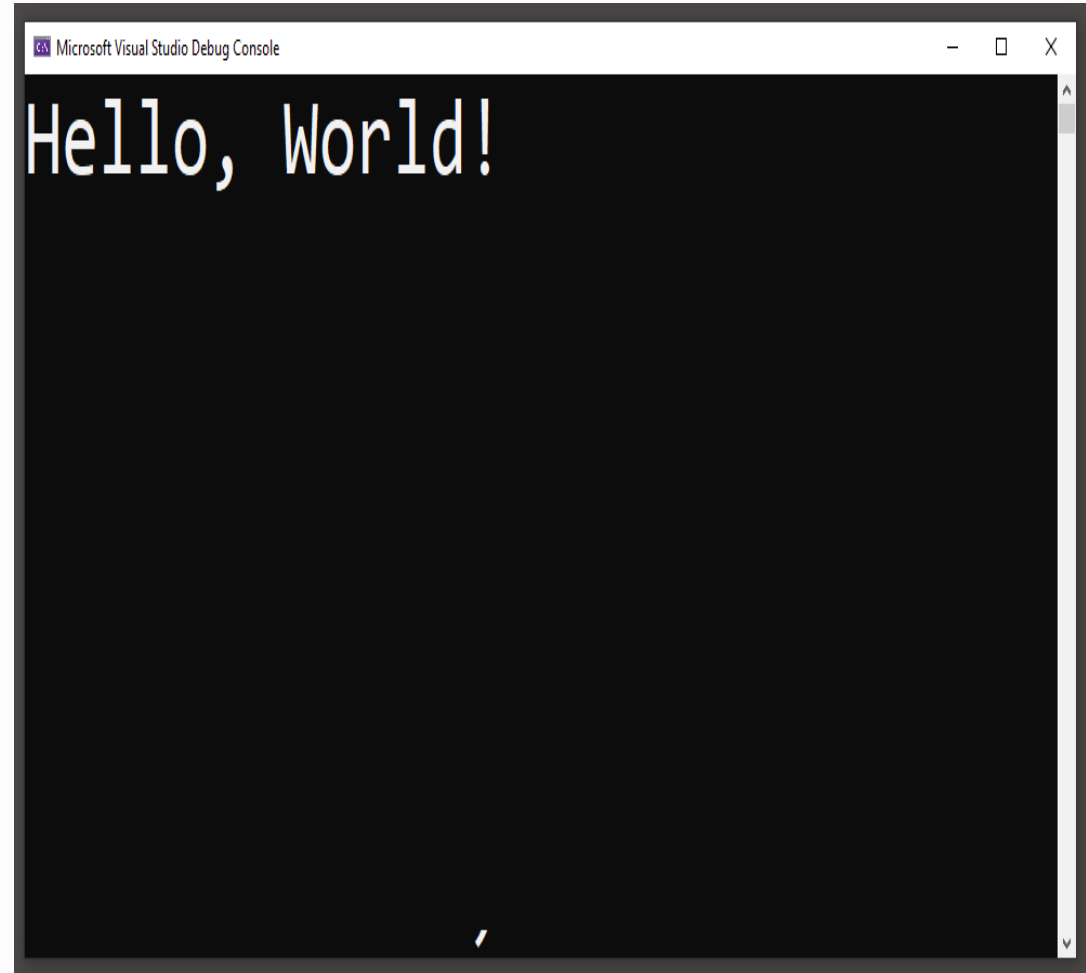


# Our First C# Program



```
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

▷ + C# 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)

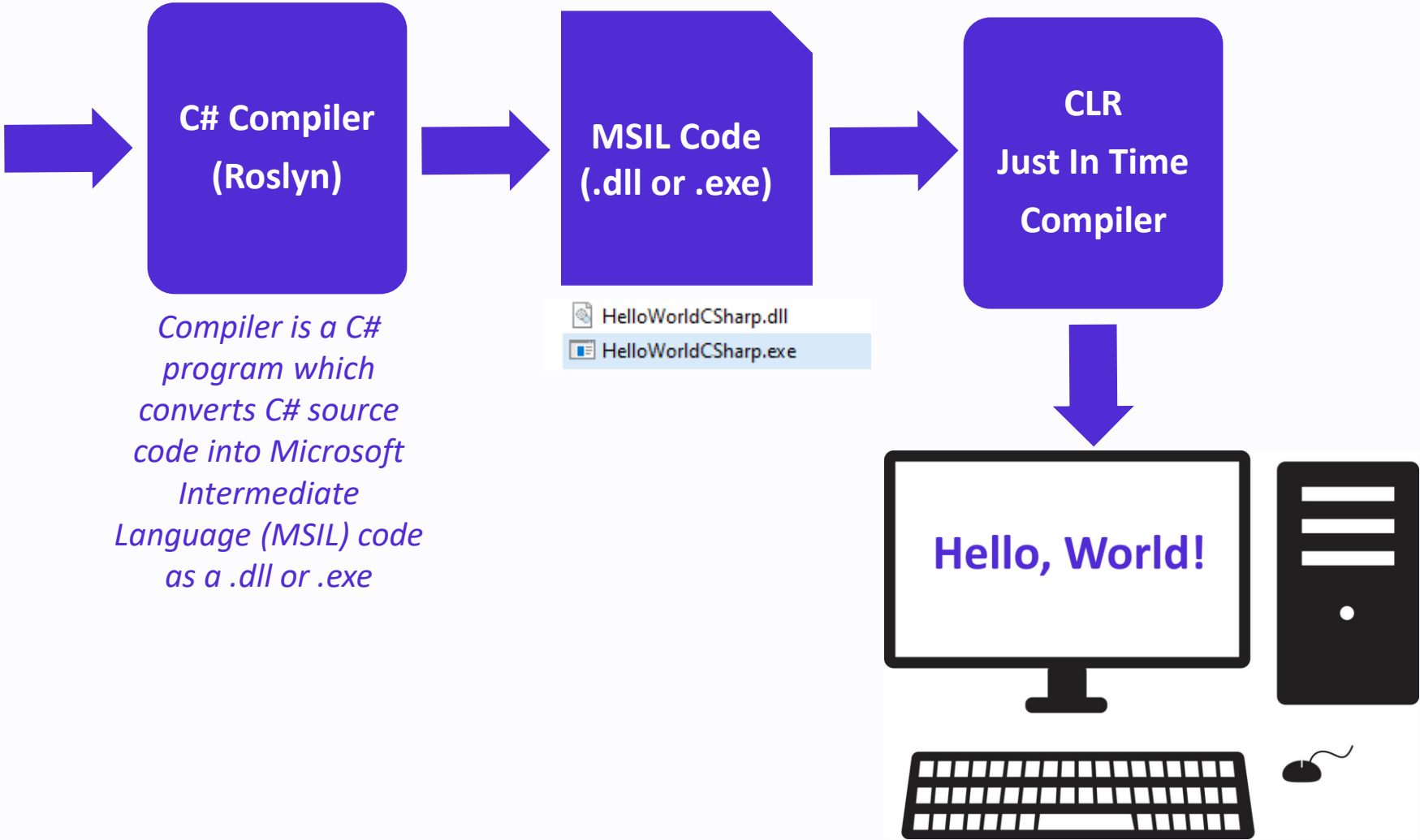
{

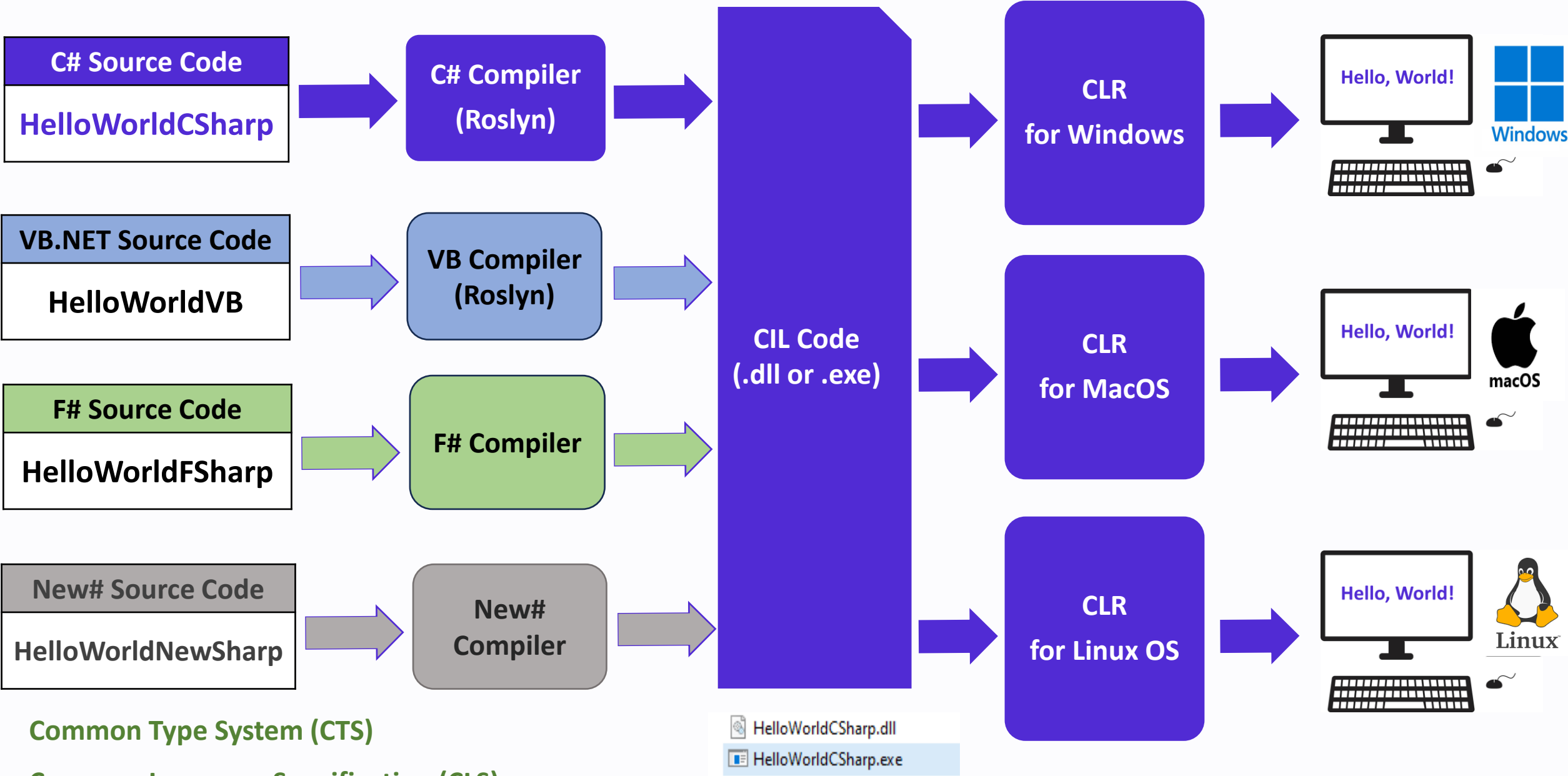
Console.WriteLine("Hello, World!");

}

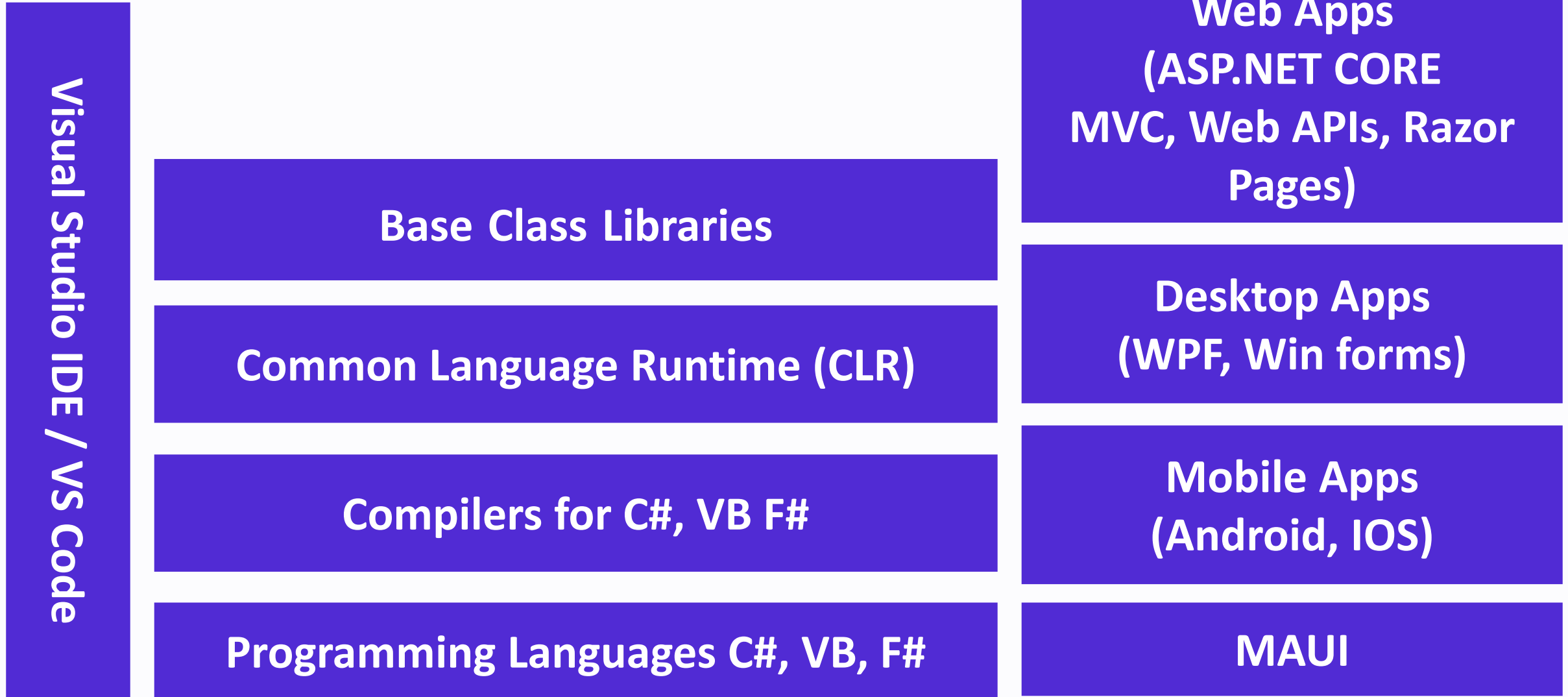
}

Console Application

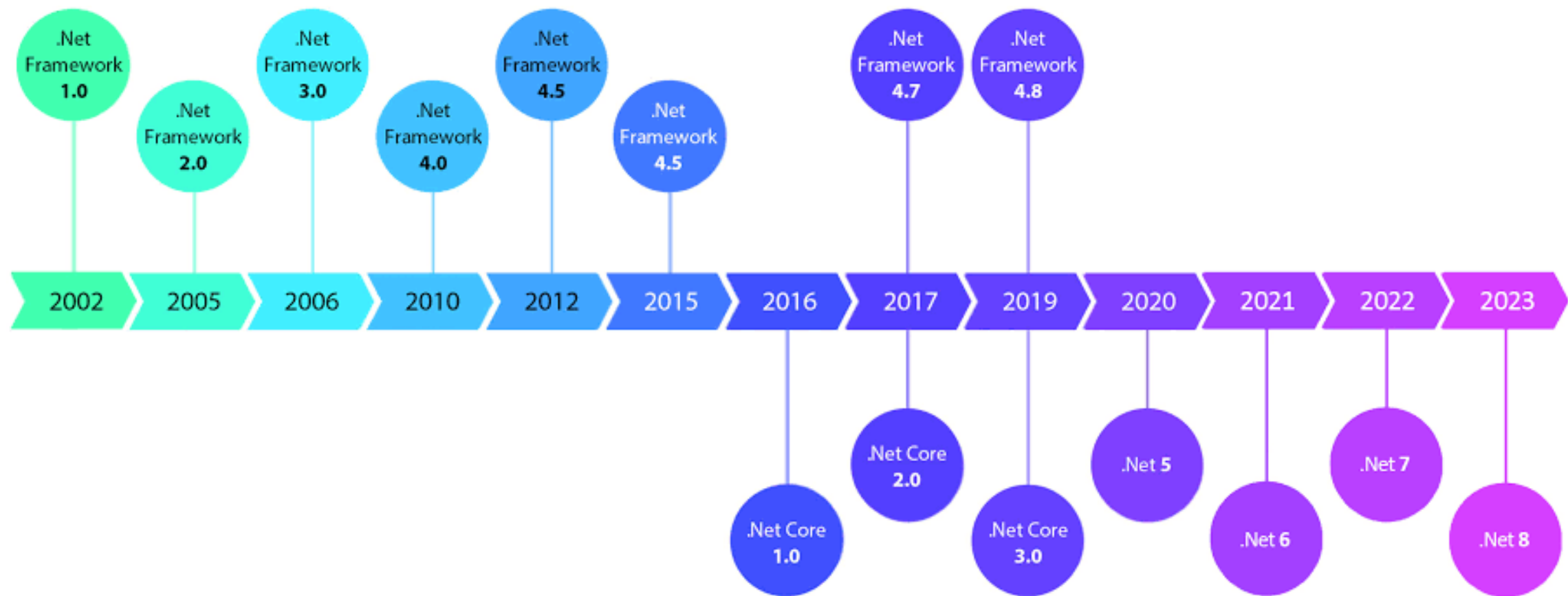




# .NET Ecosystem and its tools







# 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

A 6x3 grid of 18 purple buttons, each labeled 'Class' in white text. The top-right corner of the grid is missing, revealing a light blue background.

A diagram of a database cylinder, colored blue, containing 15 tables. The tables are arranged in a 5x3 grid, with each table represented by a white rounded rectangle with a black border and the word "Table" in black text.

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

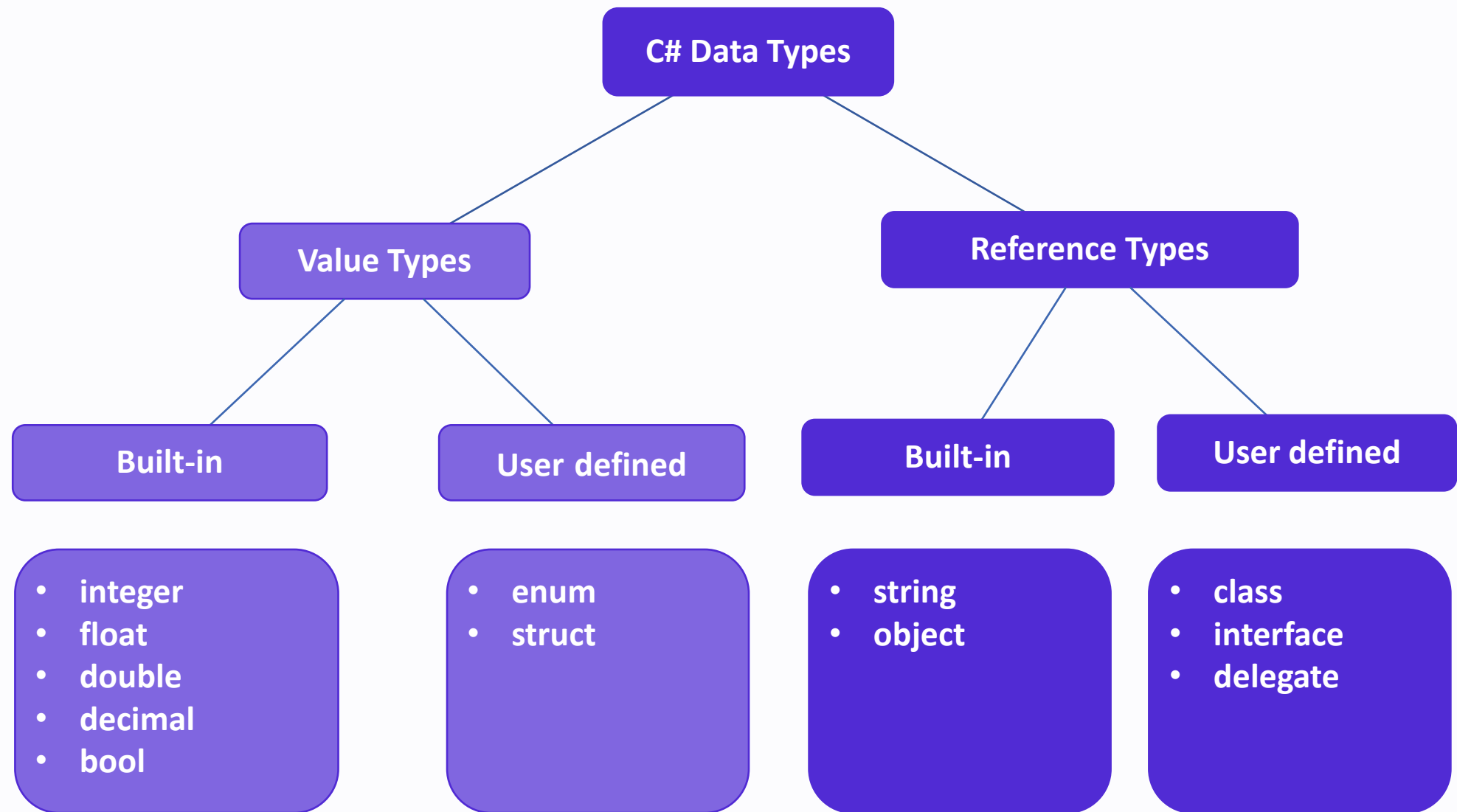
Table

# 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

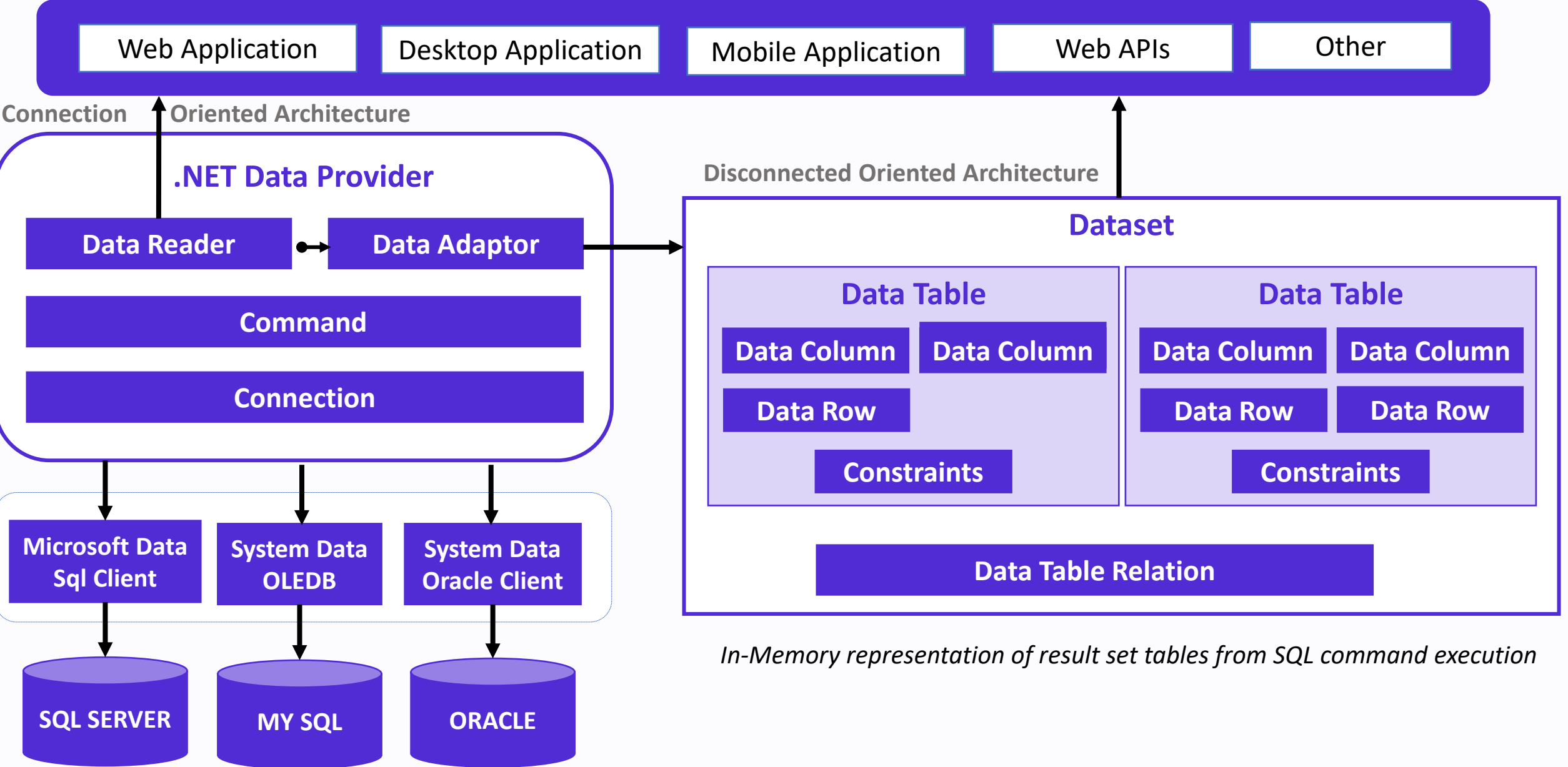
Data Type	Size	Description
Integer		
sbyte	8 bits	$-2^7$ to $2^7-1$
int	32 bits	$-2^{31}$ to $2^{31}-1$ (-2,147,483,648 to 2,147,483,647)
short	16 bits	$-2^{15}$ to $2^{15}-1$
long	64 bits	$-2^{63}$ to $2^{63}-1$ (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
Real Or Floating Point		
float	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
<b>SqlConnection</b>	Establishes a connection to a database.
<b>SqlCommand</b>	Represents an individual SQL statement or stored procedure that can be executed against the database connected.
<b>SqlDataReader</b>	Provides read-only, forward-only access to the data in a database.
<b>SqlDataAdapter</b>	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

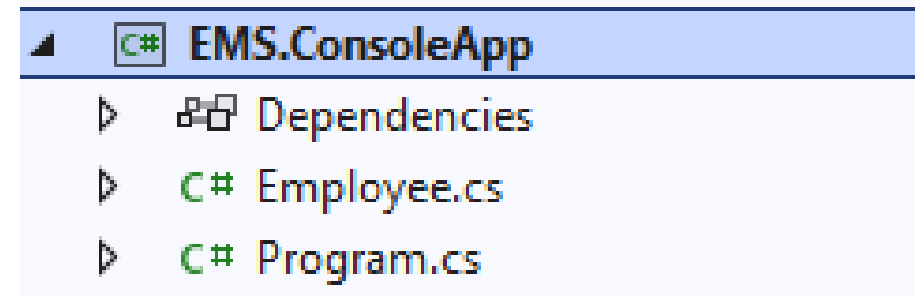
+ Fill ( ) : int

+ 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	

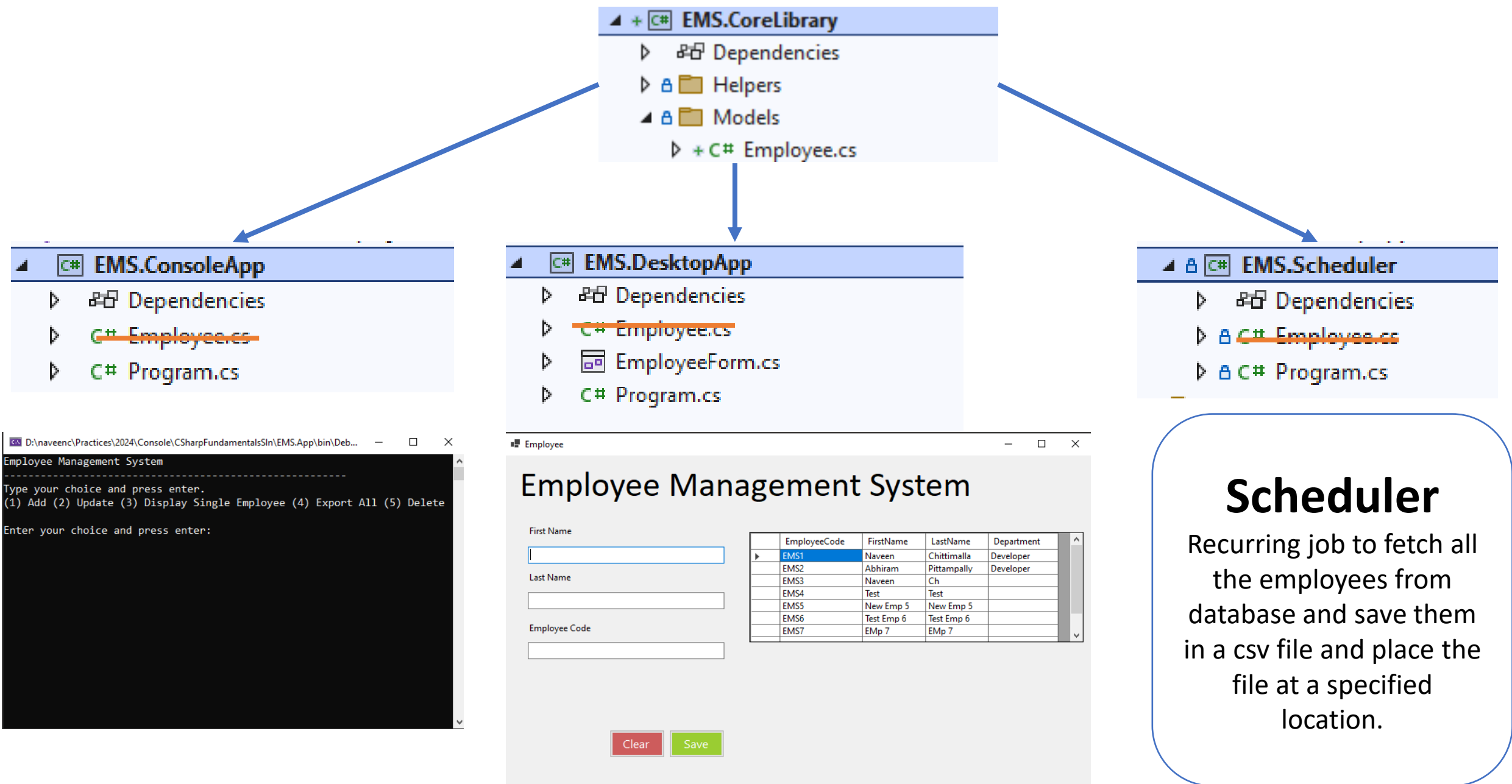
Clear

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





# 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

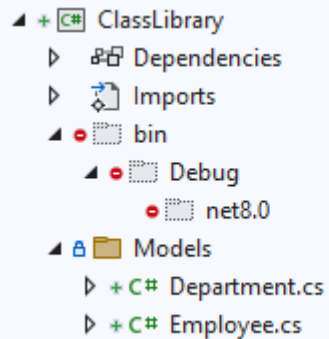
Class

interface

struct

enum

## Class Library

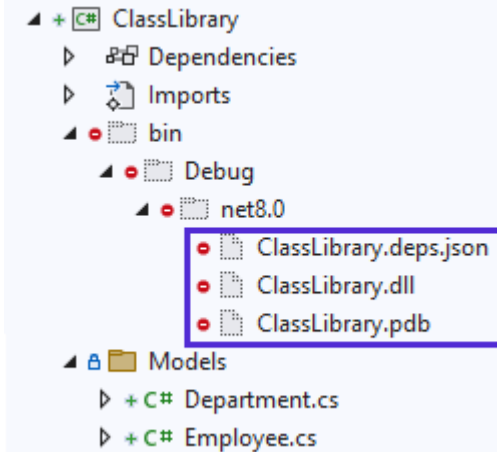


A screenshot of a Visual Studio solution explorer for a project named 'ClassLibrary'. The tree view shows the following structure: a 'Dependencies' node, an 'Imports' node, a 'bin' folder containing a 'Debug' sub-folder with a 'net8.0' sub-folder, and a 'Models' folder. Under 'Models', there are two C# files: 'Department.cs' and 'Employee.cs'.

## C# Compiler

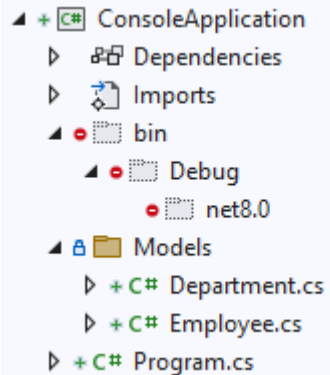
Build the Class Library project, then C# compiler converts C# code into MSIL code and packages all the types into .dll file

## .dll (Dynamic Link Library)



A screenshot of a Visual Studio solution explorer for a project named 'ClassLibrary'. The tree view shows the same structure as the previous screenshot, but with additional files in the 'bin\Debug\net8.0' folder. These files are 'ClassLibrary.deps.json', 'ClassLibrary.dll', and 'ClassLibrary.pdb', which are highlighted with a red box.

## Console Application

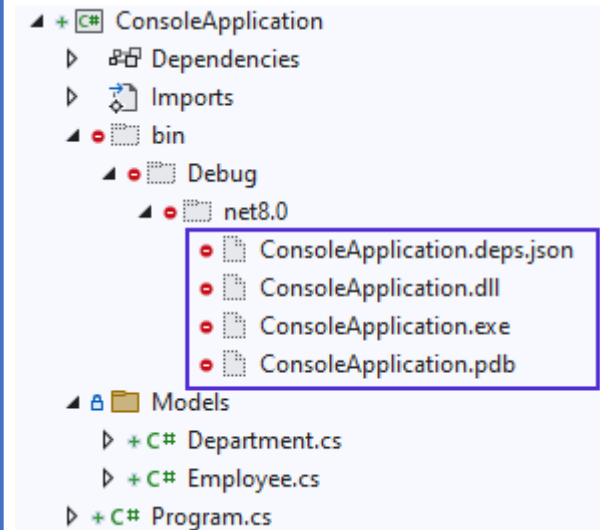


A screenshot of a Visual Studio solution explorer for a project named 'ConsoleApplication'. The tree view shows the following structure: a 'Dependencies' node, an 'Imports' node, a 'bin' folder containing a 'Debug' sub-folder with a 'net8.0' sub-folder, and a 'Models' folder. Under 'Models', there are three C# files: 'Department.cs', 'Employee.cs', and '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)



A screenshot of a Visual Studio solution explorer for a project named 'ConsoleApplication'. The tree view shows the same structure as the previous screenshot, but with additional files in the 'bin\Debug\net8.0' folder. These files are 'ConsoleApplication.deps.json', 'ConsoleApplication.dll', 'ConsoleApplication.exe', and 'ConsoleApplication.pdb', which are highlighted with a red box.

Let's Write Code

# EMS Project v3

```
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 Management System

First Name

Last Name

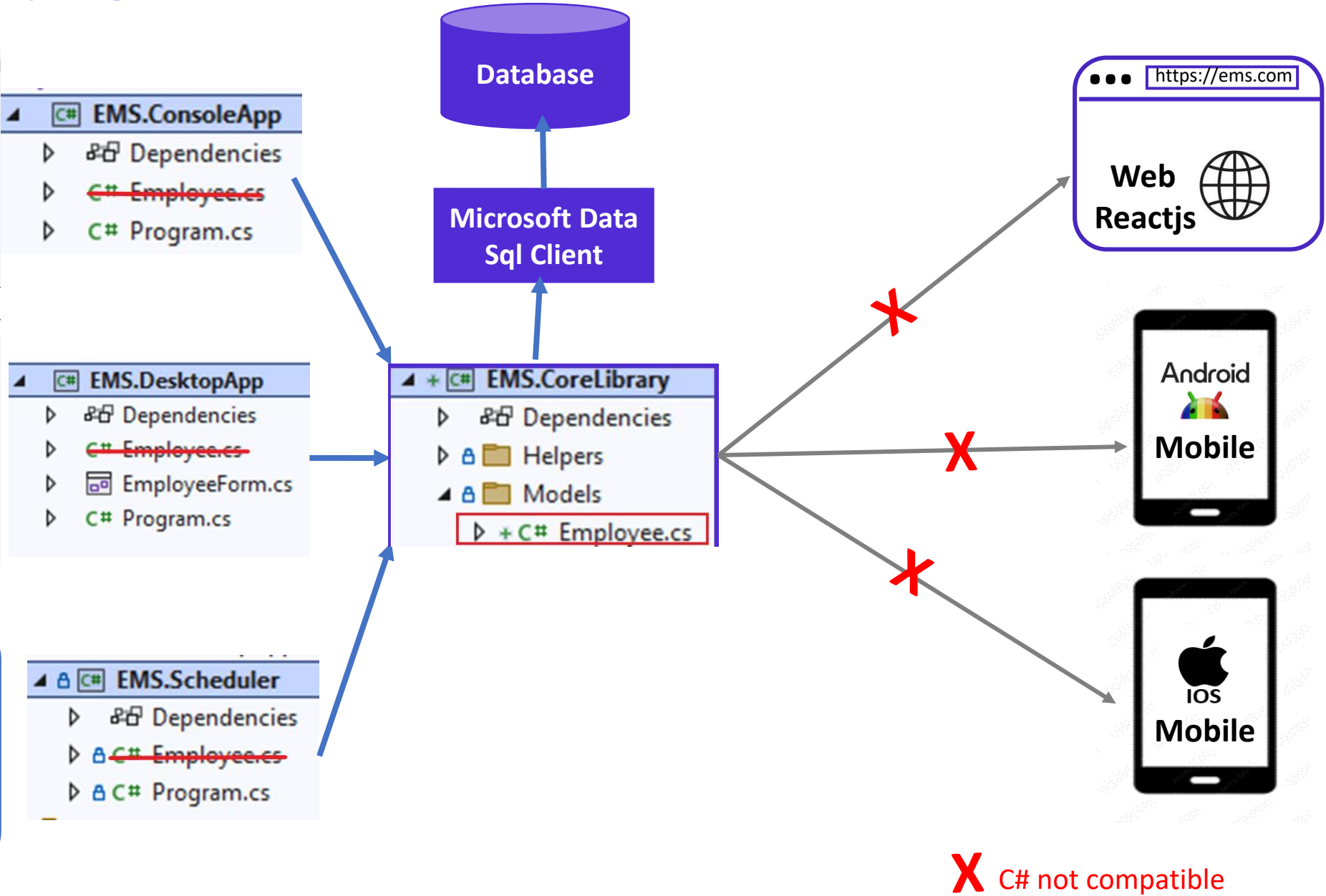
Employee Code

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

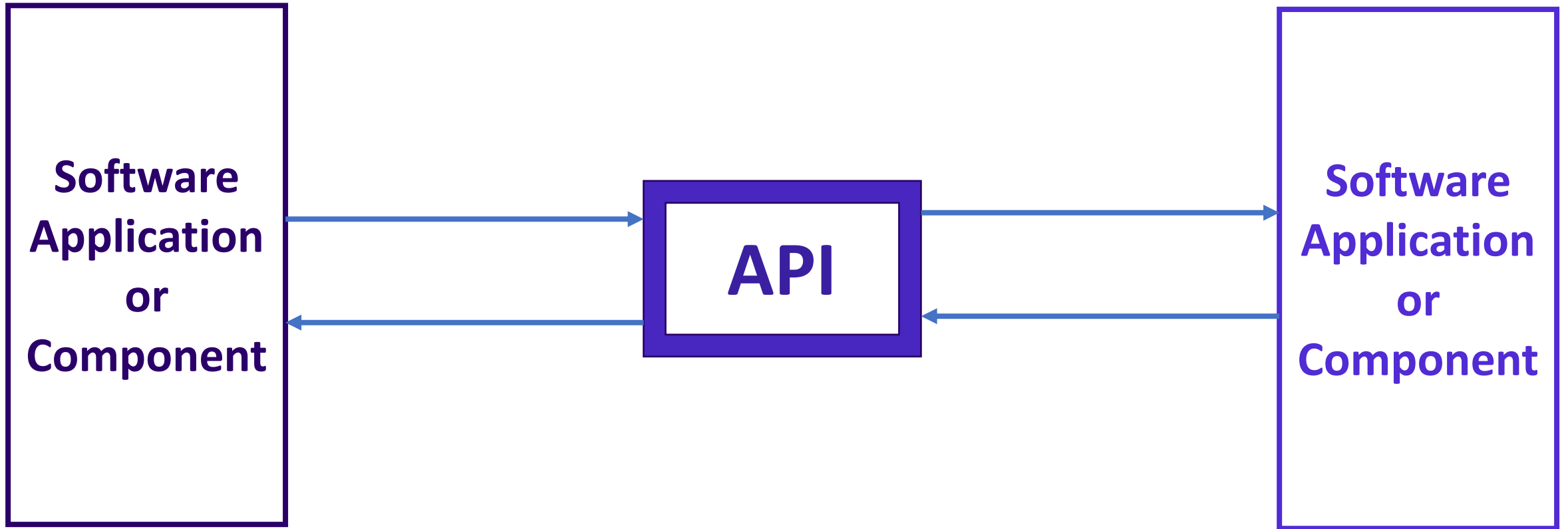
Scheduler

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



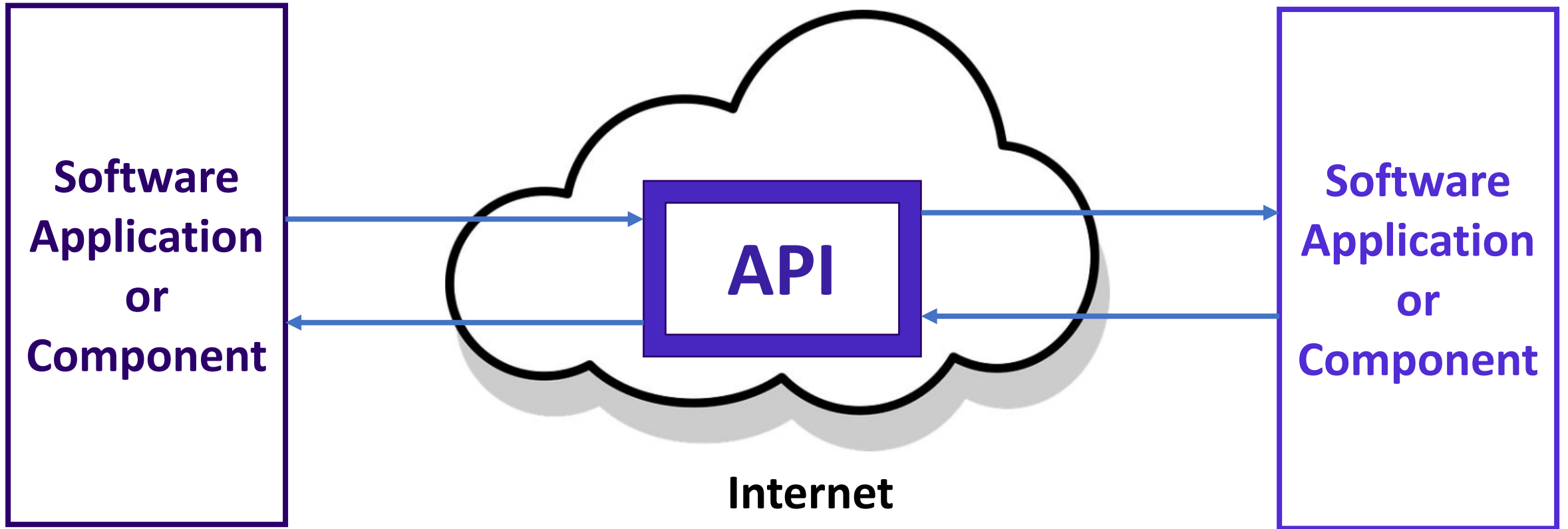
# What is API ?

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

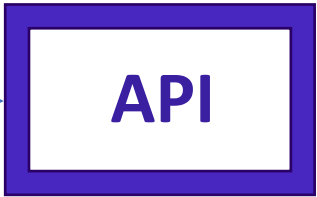
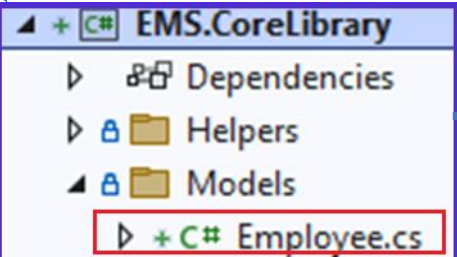
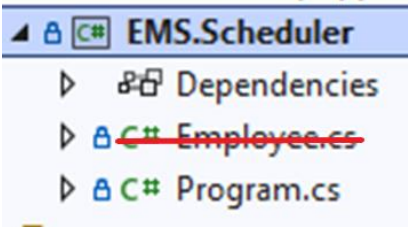
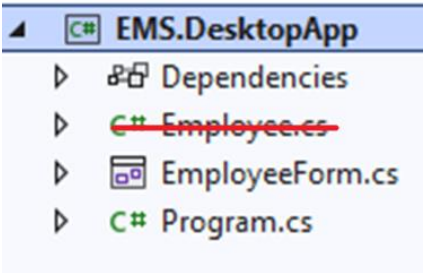
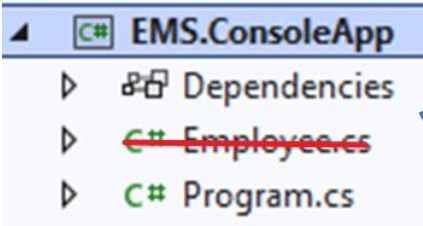
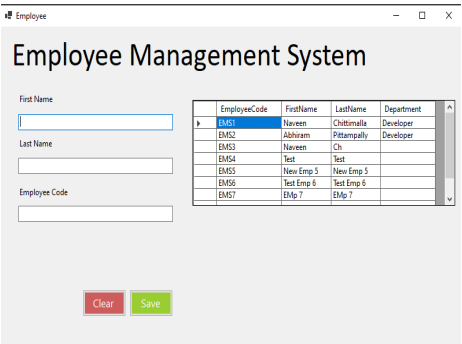
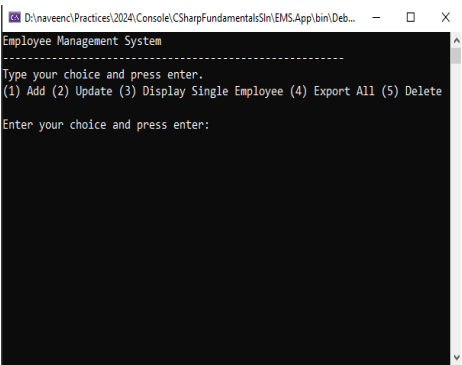


# What is Web APIs ?

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



# EMS Project v3



*API which is accessible and understandable by all application*

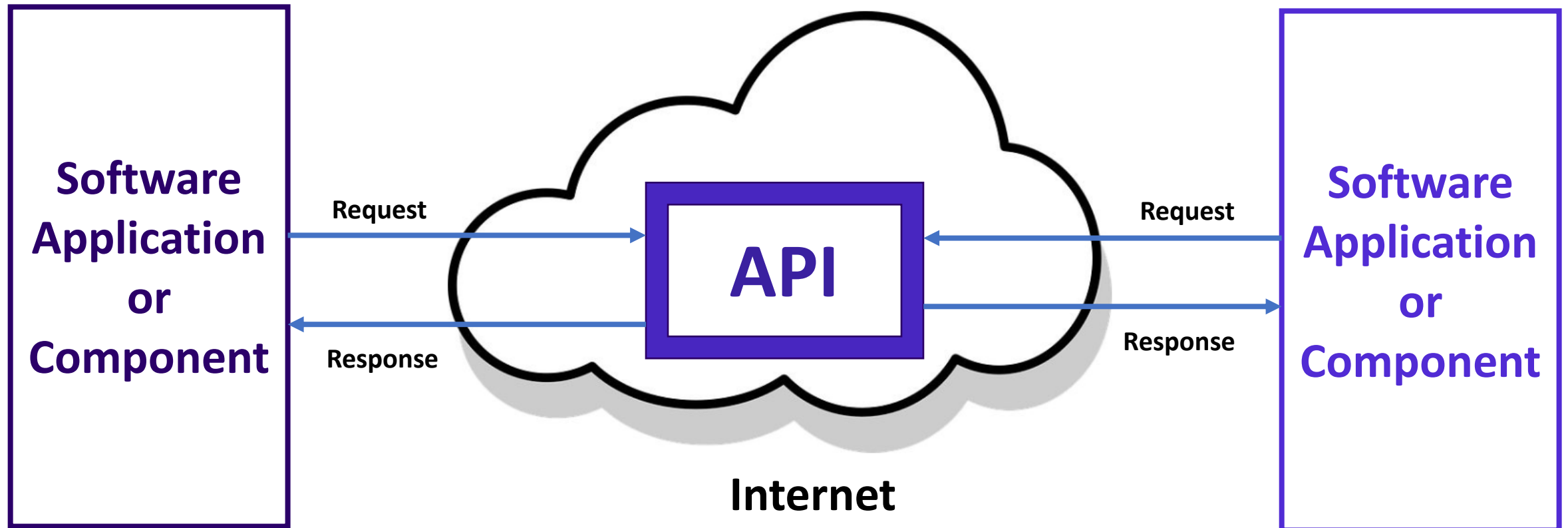


## Scheduler

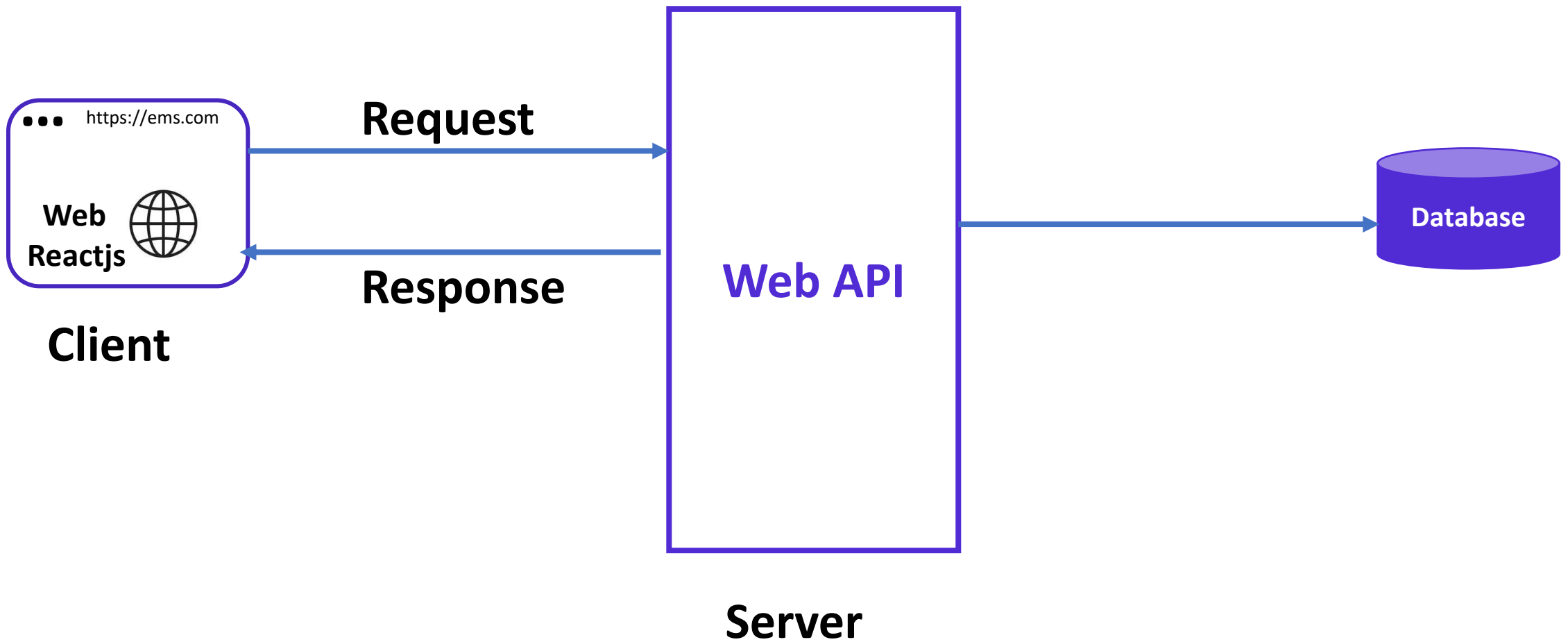
Generate a csv file and place the file at a specified location.

# What is Web APIs ?

- Application Programming interface, is a set of routines, protocols, and tools for building software applications and communicate with each other.
- Web APIs are simply the APIs that communicate over the internet.
- Follows Client-Server model with request-response pattern

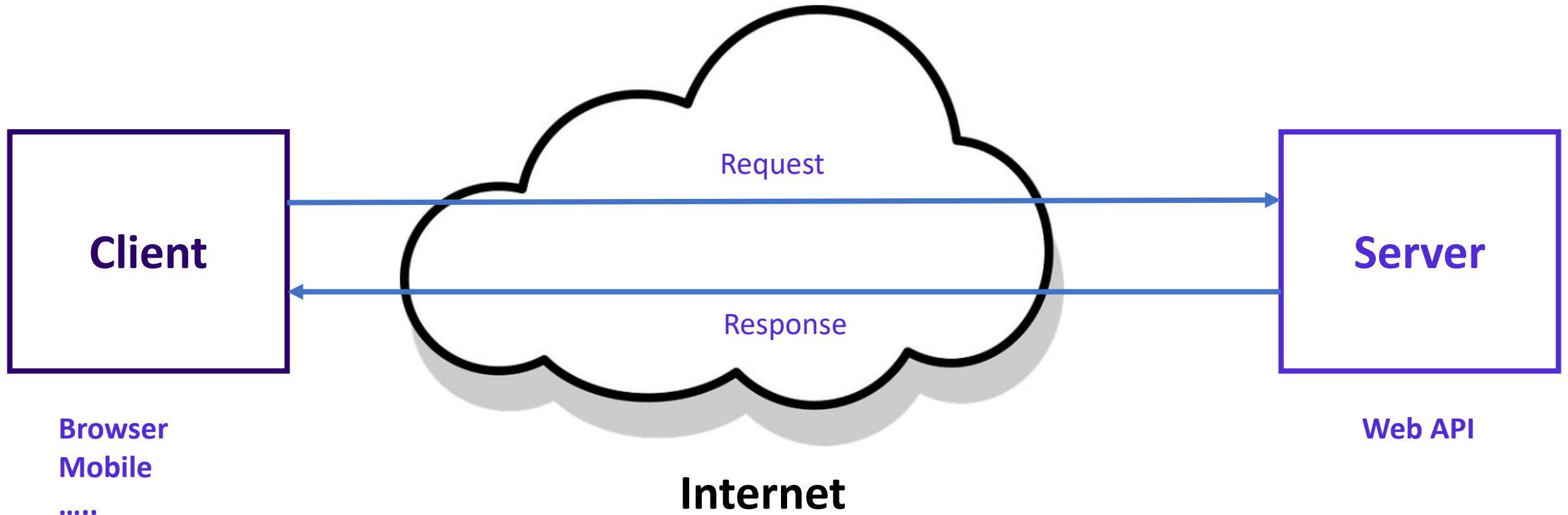






# HTTP

- Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML or data such as json, xml etc.
- Follows Client-Server model with request-response pattern
- Stateless protocol means that the server does not keep any data (state) between two requests. Simply, every request is a new request.



# HTTP Request

- HTTP Request is a message sent by client to communicate with server over internet.
- HTTP Request consists of following elements.

Element	Description	Format
Request Line	Specifies HTTP Method, path, HTTP protocol version.	{HTTP Method} {Path} {HTTP Version}
Headers	Client to pass additional information about request to server.	<ul style="list-style-type: none"><li>• Host</li><li>• User-agent</li><li>• Connection</li><li>• Accept</li><li>• Content-type</li></ul>
Empty Line	To separate body and other elements	
Body	Client to pass data to server for few HTTP methods such as POST/PUT/PATCH	Json, xml, html, text etc.

# HTTP Response

- HTTP Response is a message sent by server to communicate with client over internet.
- HTTP Response consists of following elements.

Element	Description	Format
Response Line	Specifies HTTP Method, path, HTTP protocol version.	{HTTP Version} {Status Code} {Status Text}
Headers	Server to pass additional information about response to client.	<ul style="list-style-type: none"><li>• Date</li><li>• Server</li><li>• Connection</li><li>• Content-type</li></ul>
Empty Line	To separate body and other elements	
Body	Not all responses have one: Responses with a status code that sufficiently answers the request without the need for corresponding payload (like <u>201</u> Created or <u>204</u> No Content) usually don't.	Json, xml, html, text etc.

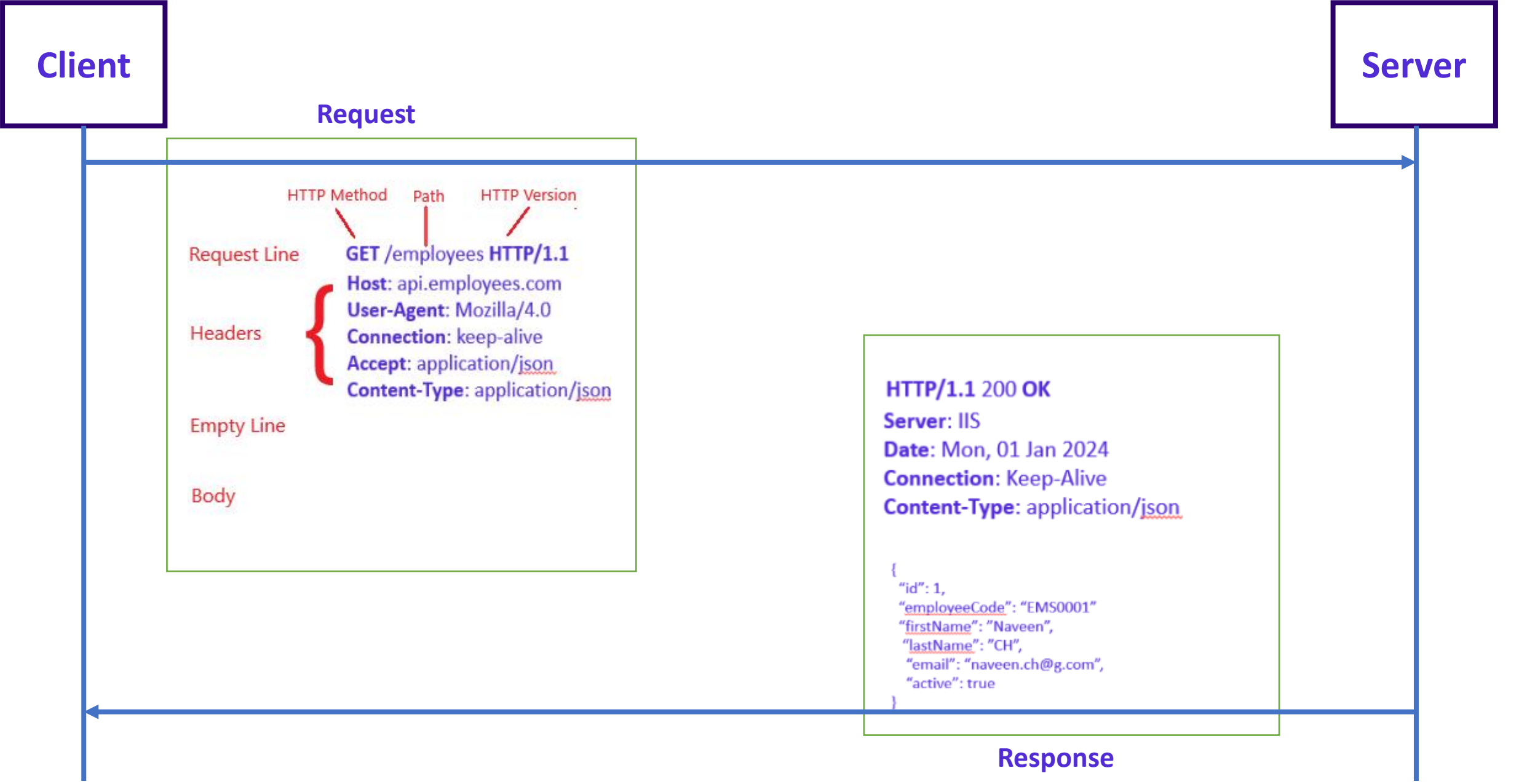
# HTTP GET Request-Response

Element	Request	Response
Request Line	GET /employees HTTP/1.1	HTTP/1.1 200 OK
Headers	Host: api.employees.com User-Agent: Mozilla/4.0 Accept: application/json Content-Type: application/json	Server: IIS Date: Mon, 01 Jan 2024 Connection: Keep-Alive Content-Type: application/json
Empty Line		
Body		{ "id": 1, "employeeCode": "EMS0001" "firstName": "Naveen", "lastName": "CH", "email": "naveen.ch@g.com", "active": true }

# HTTP POST Request-Response

Element	Request	Response
Request Line	POST /employees HTTP/1.1	HTTP/1,1 200 OK
Headers	Host: api.employees.com User-Agent: Mozilla/4.0 Accept: application/json Content-Type: application/json	Server: IIS Date: Thu, 09 May 2024 07:07:51 GMT Content-Type: application/json Location: <a href="http://localhost:5247/api/Employees/1">http://localhost:5247/api/Employees/1</a>
Empty Line		
Body	{ "firstName": "Naveen", "lastName": "CH", "email": "naveen.ch@g.com", "active": true }	{ "id": 1, "employeeCode": "EMS0001" "firstName": "Naveen", "lastName": "CH", "email": "naveen.ch@g.com", "active": true }

# HTTP Request Response



# HTTP Request Methods

- HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.
- Request methods are sometimes referred to as **HTTP verbs**.
- Request method can be safe, idempotent, or cacheable

Method	Action	Safe	Idempotent	Cacheable
GET	Retrieve a specified resource. Read-Only	YES	YES	YES
POST	Create a resource	NO	NO	NO
PUT	Update a specified resource	NO	YES	NO/{YES (Rare)}
DELETE	Delete a specified resource	NO	YES	NO/{YES (Rare)}
PATCH	Partial updates a specific resource	NO	YES	NO

- **Safe:** Doesn't alter the state on the server. Simply it is a read-only operation.
- **Idempotent:** intended effect on the server of making a single request is the same as the effect of making several identical requests, but response returned by each request may differ. Ex: first delete call will return 200 while successive ones will likely return 404.
- **Cacheable:** HTTP Response data can be stored and used later, saving a new request to server.



# HTTP Status Code Ranges

Code Range	Description
1xx (100-199)	Informational responses.
2xx (200-299)	Successful responses.
3xx (300-399)	Redirects
4xx (400-499)	Client errors
5xx (500-599)	Server errors

**Note:** If you receive a response that is not in this list, it is a non-standard response, which is custom to the server's software.

# HTTP Status Codes

Code	Text	Description
200	OK	The request has succeeded.
201	Created	The request has succeeded, and a new resource has been created as a result.
204	No Content	There is no content to send for this request.
301	Moved Permanently	The URL of the requested resource has been changed permanently. The new URL is given in the response.
400	Bad Request	The request has not processed due to invalid or incomplete data.
401	Unauthorized	"unauthenticated". The client's identity is unknown
403	Forbidden	“unauthorized”. The client's identity is known but does not have access rights to the content.
404	Not Found	The requested resource not found
405	Method Not Allowed	The method has been disabled and cannot be used.
406	Not Acceptable	Content not supported.
408	Request Timeout	Request Timeout
415	Unsupported Media Type	The media format of the requested data is not supported
500	Internal Server Error	The server has encountered a situation it doesn't know how to handle.

Thank You!

# ApiController attribute

- **Attribute Routing Requirement:** With ApiController, attribute routing becomes a requirement. This means you must use [Route], [HttpGet], [HttpPost], etc., to define the routes for your actions.
- **Automatic HTTP 400 Responses:** It automatically handles model validation errors by providing a standardized response. If the request model does not satisfy the model validation rules, the API responds with a 400 Bad Request without writing additional code.
- **Binding Source Parameter:** The attribute infers the source of parameters for actions. For instance, it assumes [FromBody] for complex type parameters and [FromQuery] for simple types in GET requests. This reduces the need to specify the source of the parameter explicitly.
- **Enhanced Swagger Support:** When used in conjunction with tools like Swashbuckle, it provides better metadata for API documentation. This results in more descriptive Swagger UIs.

# ControllerBase class

- Designed for building Web APIs or RESTful services focusing on returning data (JSON, XML) rather than rendering HTML views.
- It provides access to several properties and methods useful for handling HTTP requests, such as
  - Request
  - Response
  - ModelState
  - Various methods for returning data (Ok, NotFound, BadRequest, etc.).
- It does not include support for views or rendering HTML.

# 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;";
```

```
        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