Smit Vora

Learnings and Conclusions

Module – 1

Topics

1. **Visual Studio IDE 2019 Overview**

   1.1 **Different types of windows (solution exp.,properties etc.)**

   - Properties window is used to show the properties of objects of UI Elements and from files of class view and solution explorer.
   - It shows properties of selected items.

   1.2 **Solution and Projects**

   - Solution is a container in this IDE which contains a bunch of code related projects for ex- A class library project and test project stored in one solution.
   - We can create solution and we can create and add new projects in it.

   1.3 **Code Editor Features**

   - VS 2019 provides many features like IntelliSense to help with the coding recommending different suggestions, Go to Definition which helps us to go to the definition of an object.
   - It also provides syntax colouring for different keywords for different files, uppercase shortcut – Ctrl + Shift + U, Collapsing a block of code, Line numbers, Structure Visualizer which are dotted line from start to end bracket in a vertical line makes it easier to identify opening and closing brackets.

   1.4 **Keyboard Shortcuts**

   - VS Search – Ctrl + Q
   - Go To Definition – F12
   - Start Debugging – F5
   - Step Over – F10
   - Step Into – Shift + F10
   - Formatting – Ctrl + K ,Ctrl + D
   - Commenting – Ctrl +K ,Ctrl + C

2. Project Types

2.1 Windows App And Class Library

- Class Library in Visual Studio are essentially used for reusability of code when we need same functionalities in any other app or in a different module of the same app.
- We can use a class library in our app or project by passing reference into it and then in the file using it's namespace.
- Windows App is an app that can be run only on windows platform and can be executed by the operating system on the system directly.
- Windows app in installed in windows platform using windows operating system.
- Its interface is shown in the form of windows form.
- It has standard drag and drop of UI components from toolbox and has .cs file to edit code of any UI components

2.2 Web Application

- They are accessible anywhere where internet is available.
- They run on IIS (Internet Information Services).
- It contains Startup.cs file and Program.cs file along with data pages and dependencies.

3. Create First C# Program "Hello World"

3.1 What Is Namespace?

- A namespace is a scope in which classes are bundled and arranged logically. There can be nested namespaces as well.
- Namespace members are called by **using** keyword and writing namespace name.
- This helps us as we don't have to write fully qualified name of a member while calling it.
- There are some standard namespaces as well like System ,System.IO etc.

3.2 What is class?

- Classes are user defined datatypes which show the state and behaviour of an object. There are different **access modifiers** for class members which enables them a particular accessibility level like private, protected, public, Internal.
- Types of classes are abstract, sealed, static, partial.

3.3 Variable And Method Declaration

- Variable Declaration in C# is as follows-
  [modifier] [dataType] [variableName] = variable;
- Here **modifiers** are public, private, protected, internal defining the scope of use of variables in classes.
- **Datatypes** are int ,string ,float ,char which define the type of data to be saved in variable being declared.

- **Method** is a code block that performs a particular task. Its syntax is as follows-

```
<access specifier> <return type> <method name> (parameters)
{
        //code
}
```

- Here **modifiers** are public, private, protected, internal defining the scope of use of variables in classes.
- **return type** – it is the type of the value method will be returning.
- **Parameters** – arguments which are passed during call of the function are received as parameters during defining it.

4. Understanding C# Program

4.1 Program Flow

- The program flow of a program in C# basically contains -

Namespace declaration

A class

Class methods

Class attributes or variables

A Main method

Statements and Expressions

Comments

4.2 Understanding Syntax

- Hello World Class Program

```
using System;
namespace HelloWorld
{
   class Hello {
      static void Main(string[] args)
      {
         /* first program in C# */
         Console.WriteLine("Hello World!");
```

```
    }
  }
}
```

- The first line of the program **using System;** - the using keyword is used to include the System namespace in the program.
- The next line has the **namespace** declaration. A namespace is a collection of classes. The HelloWorld namespace contains the class HelloWorld.
- The next line has a **class** declaration, the class HelloWorld contains the data and method definitions that our program uses. Classes generally contain multiple methods. However, the HelloWorld class has only one method Main.
- The next line defines the **Main** method, which is the **entry point** for all C# programs. The Main method represents what the class does when executed.
- The next line /*...*/ is ignored by the compiler and it is put to add **comments** in the program.
- The Main method specifies its behavior with the statement **Console.WriteLine("Hello World");**

5. Working with code files, projects & solutions

5.1 Understanding structure of solution

- Solution are collection of projects which are related each other and are containing information of dependencies between them.
- A solution can contain multiple projects and a project can be part of multiple solutions.
- A solution contains projects, projects contains source files and solution also contains solutionName.sln file.

5.2 Understanding structure of project (Win app, web app, web api, class library)

- **Structure of Windows forms app**
  - **5.2..1** File name is same as app name and **[appname].csproj** file contains references of other projects and of packages used and also version of project etc.
  - **5.2..2** It has got **dependencies** which has all the server side NuGet Packages and other third party packages required in the project
  - **5.2..3** It has a .cs file which contains code of windows form that we create.
  - **5.2..4** It also contains **Program.cs** file which has Main method as required to run app as console application.

5.2..5 The configuration for app is in **CreateHostBuilder** Method called in Main method.

- **Structure of Web app**
  - **5.2..1** File name is same as app name and **[appname].csproj** file contains references of other projects and of packages used and also version of project etc.
  - **5.2..2** It contains **Connected Services** folder which is used to connect project to services like Azure and is basically used during deployment.
  - **5.2..3** It has got **dependencies** which has all the server side NuGet Packages and other third party packages required in the project
  - **5.2..4** Then there is properties folder which contains launchSettings.json which contains debug settings
  - **5.2..5** Then it contains **Pages** folder which contains some demo interface pages.
  - **5.2..6** **wwwroot** folder contains static files like images, css, JavaScript, etc. These are the only files which are served over http request.
  - **5.2..7** It also contains **Program.cs** file which has Main method as required to run app as console application.
  - **5.2..8** It also contains **Startup.cs** file which runs always first when the project is executed and it contains configurations of services used in project
  - **5.2..9** **appsettings.json** is an application configuration file and contains configurations like database settings, any global variables for whole application.
- **Structure of Class Library**
  - **5.2..1** File name is same as app name and **[appname].csproj** file contains references of other projects and of packages used and also version of project etc.
  - **5.2..2** It has got **dependencies** which has all the server side NuGet Packages and other third party code files required in the project
  - **5.2..3** It has a .cs file which contains code of a particular class that we define in it.
- **Structure of Web Api Project**
  - **5.2..1** File name is same as app name and **[appname].csproj** file contains references of other projects and of packages used and also version of project etc.
  - **5.2..2** It contains **Connected Services** folder which is used to connect project to services like Azure and is basically used during deployment.

5.2..3 It has got **dependencies** which has all the server side NuGet Packages and other third party packages required in the project

5.2..4 Then there is properties folder which contains launchSettings.json which contains debug settings

5.2..5 Then it contains **Controller** folder which contains controllers which has code of controlling or manipulating an api.

5.2..6 **wwwroot** folder contains static files like images, css, JavaScript, etc. These are the only files which are served over http request.

5.2..7 It also contains **Program.cs** file which has Main method as required to run app as console application.

5.2..8 It also contains **Startup.cs** file which runs always first when the project is executed and it contains configurations of services used in project

5.2..9 **appsettings.json** is an application configuration file and contains configurations like database settings, any global variables for whole application.

5.3 **Familiar with different type of file extensions**

- .**sln** – It contains information of the projects contained in the solution as it is the solution file.
- .**csproj** – It contains the references of other projects and of packages used and also version of project etc.
- .**cs** – class file of C#.
- .**json** – JavaScript Object Notation file which stores simple objects and data structures.

6. Understanding datatypes & variables with conversion

6.1 **Base Datatypes**

- **Value Data Type** – It is a Datatype to store value of variable in memory in signed or unsigned form.
  - **6.1..1** **Predefined Data Type** – These are the already defined datatypes in C#
    - **6.1..1.1** Integer – int, Decimal – decimal, Float – float, Character – char, Double – double, etc.
  - **6.1..2** **User Defined Data Type –** It is a Datatype which is defined and used by the users.
    - **6.1..2.1** Structure – struct, Enumerations – enum

- **Reference Data Type** – It is a Datatype which contains reference of the data of the variable.
    - **6.1..1 Predefined Data Type** – These are the already defined datatypes in C#
        - **6.1..1.1** Object, String
    - **6.1..2 User Defined Data Type –** It is a Datatype which is defined and used by the users.
        - **6.1..2.1** Classes, Interfaces
- **Pointer Data Type** - It is a Datatype which contains memory address of the data of the variable.

6.2 **Datatype Conversion**

- **Implicit Datatype Conversion** – This conversion takes place automatically when we apply value of one datatype to other. It takes place under 2 conditions –
    - **6.2..1** These two datatypes are compatible, Ex – both should be numerical if one is numerical and other is Boolean it doesn't work
    - **6.2..2** The smaller datatype should be assigned to bigger datatype Ex- int to long not vice versa.
- **Explicit Datatype Conversion** – This conversion is useful when we have to do conversion between two incompatible datatypes.
    - **6.2..1** These conversion can be lossy.
    - **6.2..2** The target datatype is the desired datatype we want to convert our value to.
- **Inbuilt Methods** – ToDecimal(), ToInt32(), ToString(), ToDouble() etc.

6.3 **Boxing And Unboxing**

- **Boxing** – It is the implicit conversion from value type to reference type. In Boxing we create an object instance and then value is copied into it.
- **Unboxing –** It is the explicit conversion from reference type object to value type. In unboxing we check if the value is boxed in an object instance and then value is copied from them.

7. Understanding Decision making & statements

7.1 **if…else**

- **if** Statement contains a Boolean Condition which if it is true than the code inside its block is executed.

- **else if** Statement contains a Boolean Condition which if it is true than the code inside its block is executed but also the condition above this statement have to be false for it to happen.
- **else** Statement code block is executed automatically after all the above conditions are found to be false
- **Syntax -**

```
if (condition 1)
    {
      // code block to be executed when if condition1 evaluates to true
    }
    else if(condition2)
    {
      // code block to be executed when
      //condition1 evaluates to false
      //condition2 evaluates to true
    }
    else if(condition3)
    {
      // code block to be executed when
      // condition1 evaluates to false
      // condition2 evaluates to false
      //condition3 evaluates to true
    }
    else
    {
       // code block to be executed when
       // condition1 evaluates to false
       // condition2 evaluates to false
       // condition3 evaluates to false
       // any other condition is True
    }
```

- **Switch case** – It is an alternative used for if…else statement and there will be an expression or variable in switch bracket the result of which will be tested by value in each case and wherever it is equal to the condition the corresponding case's code block will be executed and then it will break. It also has a default case which performs similarly to else statement.
- **Syntax :**

```
switch(match expression/variable)
{
  case value:
    statement(s) to be executed;
    break;
  default:
    statement(s) to be executed;
```

```
        break;
    }
```