### .Net Framework

نیٹ ورک سے چلنے والی ٹیکنالوجی ہے۔(Net.)

میں ایپلیکیشن کو تیار کرنے، چلانے اور تعینات کرنے کی سہولت فراہم کرتی ہے۔ جو کہ (Net.)

مختلف پلیٹ فارم کے درمیان معلومات کا تبادلہ اور استعمال کرنے کی سہولت فراہم کرتی ہے۔(Net.)

کو مائیکروسافٹ نے تیار کیا ہے۔(Net.)

پلیکیشن تیار کرنے کے لیے ہمیں

کی ضرورت ہےVisual Studio

### What is OOPS?

اسے آبجیکٹ اورینٹیڈ پر وگر امنگ لینگویج کہا جاتا ہے۔

OOPS بنیادی طور پر کلاسز اور آبجیکٹ پر کام کرتا ہے۔

\*\*Important OOPS concepts\*\*

- Class

- Objects

- Inheritance

- Encapsulation

- Abstraction

- Polymorphism

### Class

کلاس اوبجیکٹ بنانے کے لیے ٹیمپلیٹ ہوتی ہے جو کہ ایک لوجیکل اینٹیٹی ہے۔ یہ میموری میں جگہ نہیں لیتی۔ کلاس ایک بلیو پرنٹ کی طرح ہے۔ اگر کلاس سے پہلے ایکسیس موڈیفائر نہ ہو تو کلاس انٹرنل ہوتی ہے۔ بنیادی طور پر، ایک کلاس فیلڈز اور طریقوں (variable,function or method) کو ایک اکائی میں جوڑتا ہے۔

\*\*Example:\*\*

```csharp

// declaring public class

public class Student

{

string Name = "Sanaullah"; // field variable

public int a, b; // member function or method

public void display()

{

Console.WriteLine("Class & Objects in C#");

}

}

```

### Objects

ایک آبجیکٹ بنیادی طور پر کلاس کی ایک مثال ہے۔ یہ فزیکل اینٹیٹی ہے میموری میں جگہ لیتے ہیں۔ آبجیکٹ کلاس کے ممبر ان تک رسائی کے لیے ویری ایبل اور میتھڈ کو میموری میں جگہ مختص کرتا ہے۔

\*\*Example:\*\*

```csharp

class Student

{

string Name = "Sanaullah";

static void Main(string[] args)

{

Student Obj = new Student();

Console.WriteLine(Obj.Name);

}

}

```

### MVC

Model View Controller ہے۔ جو (Net۔) ایپلیکیشنز بنانے کے لئے ایک طریقہ کار ہے

### ViewData

کنٹرولر کا ڈیٹا ویو میں سینڈ کرتا ہے۔ ویوڈیٹا میں ملٹی پل آبجیکٹ بنا سکتے ہیں۔ جو بھی اوبجیکٹ بناتے ہیں وہ ڈیٹا ڈکشنری کلاس سے ہوتا ہے۔ اگر مجھے ایک ویو سے دوسرے ویو میں ڈیٹا سینڈ کرنا ہے تو اس سے نہیں کر سکتے۔

### ViewBag

کنٹرولر کا ڈیٹا ویو میں سینڈ کرتا ہے۔ اگر مجھے ایک ویو سے دوسرے ویو میں ڈیٹا سینڈ کرنا ہے تو اس سے نہیں کر سکتا۔ ویو بیگ میں ٹائپ کاسٹنگ نہیں کرنا پڑتی۔ اس میں اگر کوئی ایرر رہے تو کمپائل ٹائم چیک نہیں ہو گا، رن ٹائم پے چیک ہو گا۔ ویو بیک کے اندر جب ہم سکرپٹ سٹور کرواتے ہیں تو را میتھڈ کے اندر لکھتے ہیں:

```csharp

@Html.Raw(ViewBag.insertdata)

```

### TempData

TempData میں بھی ٹائپ کاسٹنگ کرنا ضروری ہے۔ TempData کا ڈیٹا ایک ایکشن کے ویو سے دوسرے ایکشن کے ویو میں پاس کر سکتے ہیں۔ اگر ایک ایکشن سے دوسرے ایکشن میں جائے گا تو TempData کا ڈیٹا محفوظ رہے گا۔ اگر دوسرے سے تیسرے ایکشن میں جائے گا تو TempData کا ڈیٹا حاصل نہیں ہو گا۔ اگر ہمیں اپنا ڈیٹا رکھنا ہے تو TempData.Keep کی مدد سے رکھ سکتے ہیں۔

### Session

سیشن ایک پراپرٹی ہے کنٹرولر کلاس کی جس کو HttpSessionStateBase سے ڈرائیو کیا گیا ہے۔ سیشن کا جب اوبجیکٹ بنتا ہے وہ بیس منٹ کے لیے ہوتا ہے لیکن اس کو بڑھا بھی سکتے ہیں۔ سیشن کو ٹائپ کاسٹنگ کی ضرورت ہوتی ہے۔

### Strongly Typed View

سٹر ونگ لی ٹائیپ ویو کی مدد سے کنٹرولر سے ویو میں ڈیٹا سینڈ کرتے ہیں۔ ویو کو ماڈل کلاس کے ساتھ بائینڈ کیا جاتا ہے۔ جسے سٹر ونگ لی ٹائیپ ویو کہتے ہیں۔ جو پراپرٹیز ماڈل کلاس میں ہوگی ویو میں وہی ڈسپلے ہو گی اس میں کمپائل ٹائم پے ایرر چیک ہوتے ہیں۔

### Partial View

مین ویو کے اندر جو ڈیو ہوتا ہے اس کو پارشل ویو کہتے ہیں۔ پارشل ویو کو کبھی بھی سنگل استعمال نہیں کر سکتے، دوسرے ویوز کے ساتھ اٹیچ ہوتا ہے۔

### Sealed classes

ایسی کلاسز جن کو انہیرٹ نہ کیا جا سکے، سیلڈ کلاسز کہلاتی ہیں۔ جن کی چائلڈ کلاسز نہیں بنائی جا سکتیں۔ سیلڈ کلاس بنانے کے لئے کلاس سے پہلے "sealed" کا کی ورڈ استعمال کرتے ہیں۔

### Constructor

کنسٹرکٹر اوبجیکٹ کی انیشلائزیشن کے وقت کال ہونے والا میتھڈ ہے۔ کنسٹرکٹر کا وہی نام ہو تا ہے جو ہماری کلاس کا نام ہو تا ہے۔ کنسٹرکٹر میں کوئی بھی ٹائپ نہیں ہوتی۔

### Data Annotation

ماڈل کلاس کے اندر جو پراپرٹیز بناتے ہیں ان پر ڈیٹا اینوٹیشن لگائی جاتی ہیں۔

### Cookies

کوکیز میں انفارمیشن اسٹور کرواتے ہیں بعد میں استعمال کرنے کے لیے۔ کوکیز ہمارے بروزر میں سیو ہوتی ہیں۔

### Interface

انٹر فیس بھی کلاس کی طرح میتھڈ اور پراپرٹیز کی کلیکشن ہوتی ہے۔ لیکن اس میں میتھڈ کی ڈیکلریشن دیتے ہیں میتھڈ کو انیشلائزیشن نہیں کروا سکتے۔ انٹر فیس کلاس کا لے آؤٹ بنانے اور ملٹیپل انہیرٹینس ایچیو کرنے کے لئے استعمال کرتے ہیں۔ انٹرفیس میں کسی بھی میتھڈ کی باڈی نہیں ہوتی۔ انٹر فیس ڈائریکٹلی اوبجیکٹ نہیں بنا سکتے۔ انٹر فیس کو استعمال کرنے کے لئے پہلے کوئی کلاس انہیرٹ کرے گی پھر اس کلاس کو اوبجیکٹ بنایا جائے گا۔

### Abstract Class

ایبسٹر یکٹ کلاس میں باڈی اور بغیر باڈی کے میتھڈ بنا سکتے ہیں۔ ایبسٹریکٹ کلاس ڈائریکٹلی اوبجیکٹ نہیں بناسکتی۔ ایبسٹر یکٹ کلاس کو استعمال کرنے کے لئے پہلے کوئی کلاس انہیریٹ کرے گی پھر اس کلاس کو اوبجیکٹ بنایا جائے گا۔

### Abstraction

ابسٹریکشن کا مطلب یہ ہے کہ ضروری چیزیں دکھائی جائیں اور غیر ضروری چیزوں کو یوزر سے چھپا لیا جائے۔

### Encapsulation

کلاس کے اندر ویری ایبل اور میتھڈ کو کمبائن کرنا انکیپسولیشن کہلاتا ہے۔

### Validation

یوزر سے انپوٹ لیتے ہوئے جو چیک اور کنڈیشن لگائی جاتی ہیں انہیں ویلیڈیشن کہتے ہیں۔ ویلیڈیشن اس لئے ضروری ہیں کہ ڈوپلیکیٹ یا غلط ڈیٹا نہ جائے۔

- \*\*کلائنٹ سائیڈ ویلیڈیشن\*\*: کلائنٹ سائیڈ میں فرنٹ اینڈ پر چیک لگتے ہیں۔

- \*\*سرور سائیڈ ویلیڈیشن\*\*: کنڈیشن کو بیک اینڈ پر ویلیڈیٹ کرواتے ہیں۔

### Static Class

اسٹیٹک کلاس ہم اس وقت بناتے ہیں جب ہمیں اپنے ڈیٹا کو کونسٹنٹ رکھنا ہوتا ہے۔ سٹیٹک کلاس سے پہلے "static" کا کی ورڈ استعمال کرتے ہیں۔ سٹیٹک کلاس کے اندر صرف سٹیٹک ممبر ہی ہو سکتے ہیں۔ سٹیٹک کلاس کا اوبجیکٹ نہیں بنا سکتے۔ سٹیٹک کلاس کو کسی دوسری کلاس سے انہیرٹ بھی نہیں کروا سکتے۔

\*\*Example:\*\*

```csharp

public static class Student

{

public static string name;

public static int cube(int n) { return n \* n \* n; }

}

```

### Destructors

ڈی کنسٹرکٹر ایک خاص قسم کا میتھڈ ہے جس کا نام وہی ہو گا جو کلاس کا نام ہو گا۔ ڈی کنسٹرکٹر سے پہلے ہم کوئی ایکسس موڈیفائر استعمال نہیں کرتے۔ ڈی کنسٹرکٹر کے نام سے پہلے ٹلڈ (~) کا کریکٹر ہوگا۔ ڈی کنسٹرکٹر اوبجیکٹ کی میموری کو ڈسٹرائے کر دیتا ہے جب اوبجیکٹ کا کام ختم ہو جائے۔ ڈی کنسٹرکٹر اور لوڈ نہیں ہو سکتے اور ان کو انہیرٹ بھی نہیں کر سکتے۔ ڈی کنسٹرکٹر کو خود سے کال نہیں کر سکتے، جب اوبجیکٹ کا کام ختم ہو جائے گا تو اوبجیکٹ کی ساری میموری کو ڈسٹرائے کرے گا۔