**Partial classes in C# - Part 61**

**In this video we will discuss about**  
**1.** What are partial classes?  
**2.** What are the advantages of using partial classes?  
**3.** Where are partial classes used?  
  
**Partial classes allow us to split a class into 2 or more files.**  All these parts are then combined into a single class, when the application is compiled. The partial keyword can also be used to split a struct or an interface over two or more files.

**Let's understand partial classes with an example.**Create an asp.net web application project. Add a class file, with name **Customer.cs** to the project. Copy and paste the following code in the **customer.cs** file. This is a very simple customer class, with 2 private fields, 2 public properties and a public method.  
public class Customer  
{  
    private string \_firstName;  
    private string \_lastName;  
  
    public string FirstName  
    {  
        get { return \_firstName; }  
        set { \_firstName = value; }  
    }  
  
    public string LastName  
    {  
        get { return \_lastName; }  
        set { \_lastName = value; }  
    }  
  
    public string GetFullName()  
    {  
        return \_firstName + ", " + \_lastName;  
    }  
}

**Now, let us split this class into 2 files.**One file is going to contain, the private fields and public properties, and the other file is going to contain the public method. Right click on the web application project, and add a class file, with name **PartialCustomerOne.cs**. Notice, that the **PartialCustomer** class is marked with the **partial** keyword and it contains, only, the 2 private fields and the public properties.   
public partial class PartialCustomer  
{  
    private string \_firstName;  
    private string \_lastName;  
  
    public string FirstName  
    {  
        get { return \_firstName; }  
        set { \_firstName = value; }  
    }  
  
    public string LastName  
    {  
        get { return \_lastName; }  
        set { \_lastName = value; }  
    }  
}  
  
**Now, add another class file with name, PartialCustomerTwo.cs**. Notice that, the **PartialCustomer** class, in this file is also marked as a **partial** class, and contains only the public method - **GetFullName()**. We are able to access the private fields, **\_firstName** and **\_lastName**, that are defined in **PartialCustomerOne.cs** file.  
public partial class PartialCustomer  
{  
    public string GetFullName()  
    {  
        return \_firstName + ", " + \_lastName;  
    }  
}  
  
**Copy and paste the following code in the Page\_Load() event of the webform1.** Though, the **PartialCustomer** class is split across 2 files(PartialCustomerOne.cs and PartialCustomerTwo.cs), we are able to use it the same way as the Customer class.  
Customer c1 = new Customer();  
c1.FirstName = "Pragim";  
c1.LastName = "Technologies";  
  
string FullName1 = c1.GetFullName();  
Response.Write("Full Name = " + FullName1 + "<br/>");  
  
PartialCustomer c2 = new PartialCustomer();  
c2.FirstName = "Pragim";  
c2.LastName = "Tech";  
  
string FullName2 = c2.GetFullName();  
Response.Write("Full Name = " + FullName2 + "<br/>");  
  
**Advantages of partial classes**  
**1.** The main advantage is that, visual studio uses partial classes to separate, automatically generated system code from the developer's code. For example, when you add a webform, two .CS files are generated  
**a) WebForm1.aspx.cs -**Contains the developer code  
**b) WebForm1.aspx.designer.cs -** Contains the system generated code. For example, declarations for the controls that you drag and drop on the webform.  
  
**2.** When working on large projects, spreading a class over separate files allows multiple programmers to work on it simultaneously. Though, microsoft claims this as an advantage, I haven't really seen anywhere, people using partial classes, just to work on them simultaneously.