Day 18 Assignments

By

Praveen Chakravarthi

16-02-2022

NB Health Care

1. What is the use of XML

- XML is used for Universal Data Transfer mechanism to send data across different platforms
- It is used to create Information and describe Data

2. Write the points discussed about xml in the class

- XML is e**X**tension **M**arkup **L**anguage
- XML is Case Sensitive
- XML is User Defined
- XML has only one Route tag
- XML has two types:

1. Tag based XML
2. Attributes based XML

3. Create a simple ml to illustrate:
a. Tag based xml with 10 products
b. Attribute based xml

Code :

a. Tag based XML

```
<Customers>
  <Customer>
    <ID>01</ID>
    <Name>Ram</Name>
    <Age>21</Age>
  </Customer>
  <Customer>
    <ID>02</ID>
    <Name>Ramesh</Name>
    <Age>22</Age>
  </Customer>
  <Customer>
```

```
    <ID>03</ID>
    <Name>Naveen</Name>
    <Age>20</Age>
  </Customer>
  <Customer>
    <ID>04</ID>
    <Name>Kranthi</Name>
    <Age>24</Age>
  </Customer>
  <Customer>
    <ID>05</ID>
    <Name>Anil</Name>
    <Age>27</Age>
  </Customer>
  <Customer>
    <ID>06</ID>
    <Name>Mahender</Name>
    <Age>25</Age>
  </Customer>
  <Customer>
    <ID>07</ID>
    <Name>Ganesh</Name>
    <Age>28</Age>
  </Customer>
  <Customer>
    <ID>08</ID>
    <Name>Murthi</Name>
    <Age>30</Age>
  </Customer>
  <Customer>
    <ID>09</ID>
    <Name>Prudvi</Name>
    <Age>25</Age>
  </Customer>
  <Customer>
    <ID>10</ID>
    <Name>Arun</Name>
    <Age>24</Age>
  </Customer>
</Customers>
```

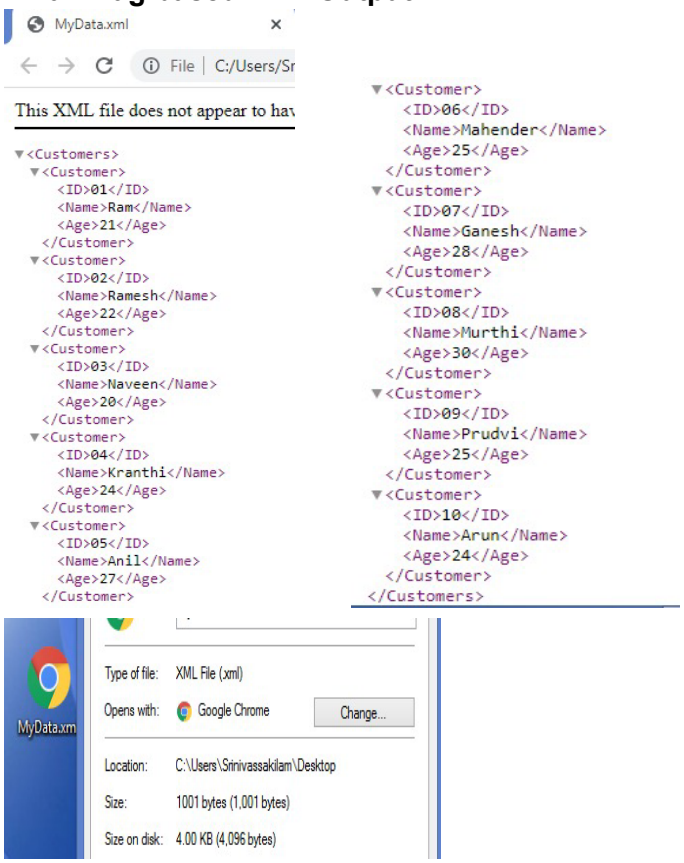
b. Attribute based XML

```
<Customers>
  <Customer ID = "01" Name = "Mahesh" Age = "22" />
  <Customer ID = "02" Name = "Mahender" Age = "29" />
  <Customer ID = "03" Name = "Manish" Age = "28" />
  <Customer ID = "04" Name = "Akhil" Age = "27" />
  <Customer ID = "05" Name = "Bhanu" Age = "21" />
  <Customer ID = "06" Name = "Nikhil" Age = "20" />
  <Customer ID = "07" Name = "Varun" Age = "28" />
  <Customer ID = "08" Name = "Ravi" Age = "26" />
```

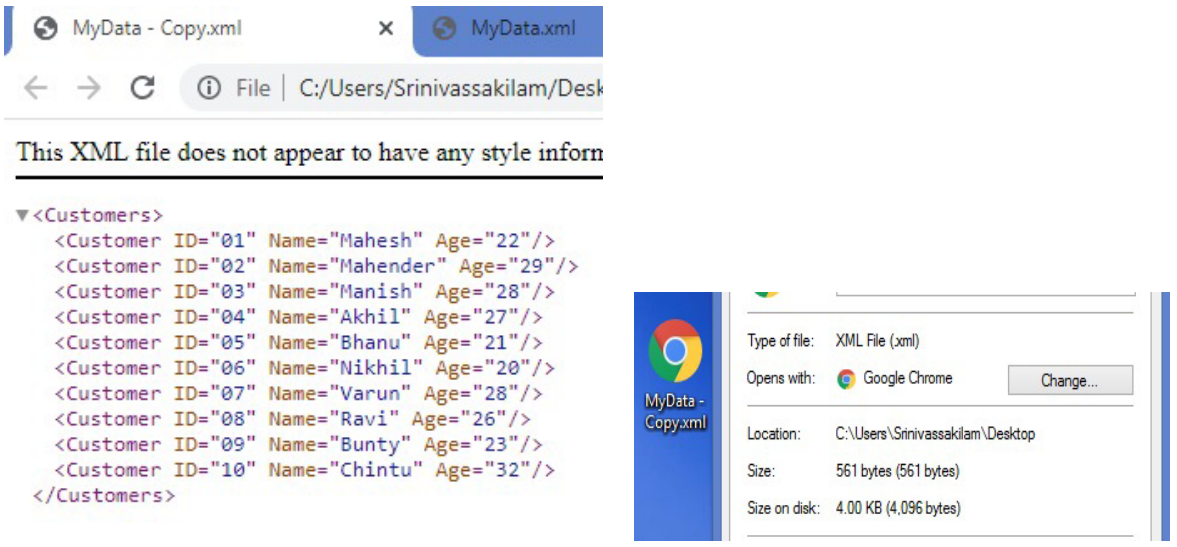
```
  <Customer ID = "09" Name = "Bunty" Age = "23" />
  <Customer ID = "10" Name = "Chintu" Age = "32" />
</Customers>
```

Output :

a. Tag based XML Output



b . Attributes bases XML Output



4. Convert the above xml to JSON and display the JSON data

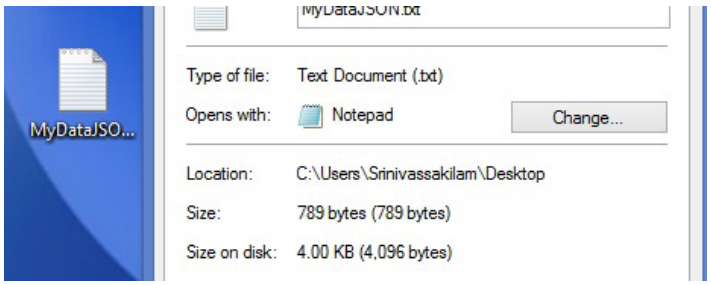
Code :

JSON


```
[
  {
    "@ID": "01",
    "@Name": "Mahesh",
    "@Age": "22"
  },
  {
    "@ID": "02",
    "@Name": "Mahender",
    "@Age": "29"
  },
  {
    "@ID": "03",
    "@Name": "Manish",
    "@Age": "28"
  },
  {
    "@ID": "04",
    "@Name": "Akhil",
    "@Age": "27"
  },
  {
    "@ID": "05",
    "@Name": "Bhanu",
    "@Age": "21"
  },
  {
    "@ID": "06",
    "@Name": "Nikhil",
    "@Age": "20"
  },
  {
    "@ID": "07",
    "@Name": "Varun",
    "@Age": "28"
  },
  {
    "@ID": "08",
    "@Name": "Ravi",
```

```
    "@Age": "26"
  },
  {
    "@ID": "09",
    "@Name": "Bunty",
    "@Age": "23"
  },
  {
    "@ID": "10",
    "@Name": "Chintu",
    "@Age": "32"
  }
]
```

Output :



5. Research and write the benefits of JSON over XML (2 or 3 points)

- JSON(JavaScript Object Notation) is easier to understand compared to XML
- It has Less Character Count
- JSON is much easier to parse
- JSON takes less memory compared to XML

6. For the below requirement, create a layered architecture project with separate class library for Business logic.
create console application
create windows (or desktop) application
Business Requirement:
FIND FACTORIAL OF A NUMBER:
0 = 1
positive number (upto 7) = factorial answer
>7 = -999 (as answer)
< 0 = -9999 (as answer)

put the screen shots of the output and project (solution explorer) screen shot

Code :

MathLibrary

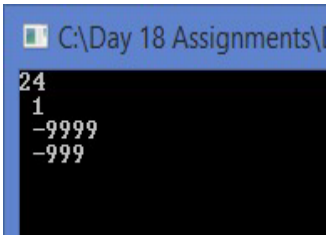
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathLibrary
{
    // Author : Praveen Chakravarthi
    // Purpose : Layered Architecture, Implementing TDD
   public class Algebra
   {
       /// <summary>
       /// This Method returns Factorial of a Number
       /// </summary>
       public static int Factorial(int n)
       {
           if (n== 0)
               return 1;

           else if (n < 0)
               return -9999;

           else if (n > 7)
               return -999;

           int fact = 1;
           for (int i =1; i <= n; i++)
               fact *= i;
           return fact;

       }
   }
}
```

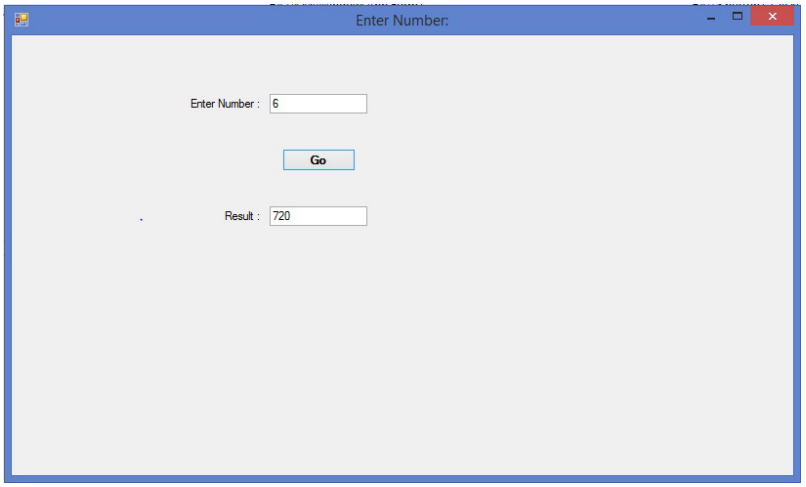
ConsoleApp

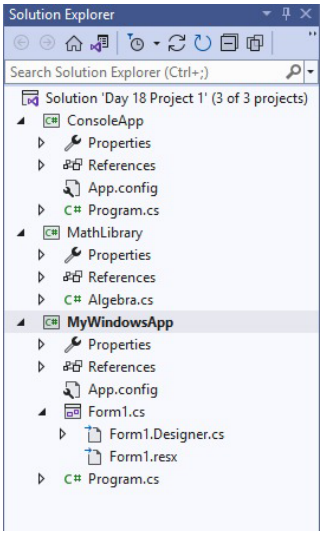
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MathLibrary;

namespace ConsoleApp
{
```

```
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine($"{Algebra.Factorial(4)}\n" +
                              $" {Algebra.Factorial(0)}\n" +
                              $" {Algebra.Factorial(-3)}\n" +
                              $" {Algebra.Factorial(9)}");
            Console.ReadLine();
        }
    }
}
```

Output :







7. For the above method, Implement TDD and write 4 test cases and put the code in word document.
put the screen shot of all test cases failing. make the test cases pass. put the screen shot

Code :

MathLibrary

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathLibrary
{
    // Author : Praveen Chakravarthi
    // Purpose : Layered Architecture, Implementing TDD
   public class Algebra
   {
      /// <summary>
      /// This Method returns Factorial of a Number
      /// </summary>
      public static int Factorial(int n)
      {
         if (n== 0)
            return 1;

         else if (n < 0)
            return -9999;

         else if (n > 7)
            return -999;

         int fact = 1;
         for (int i =1; i <= n; i++)
            fact *= i;
         return fact;

      }
   }
}
```

LibraryTests

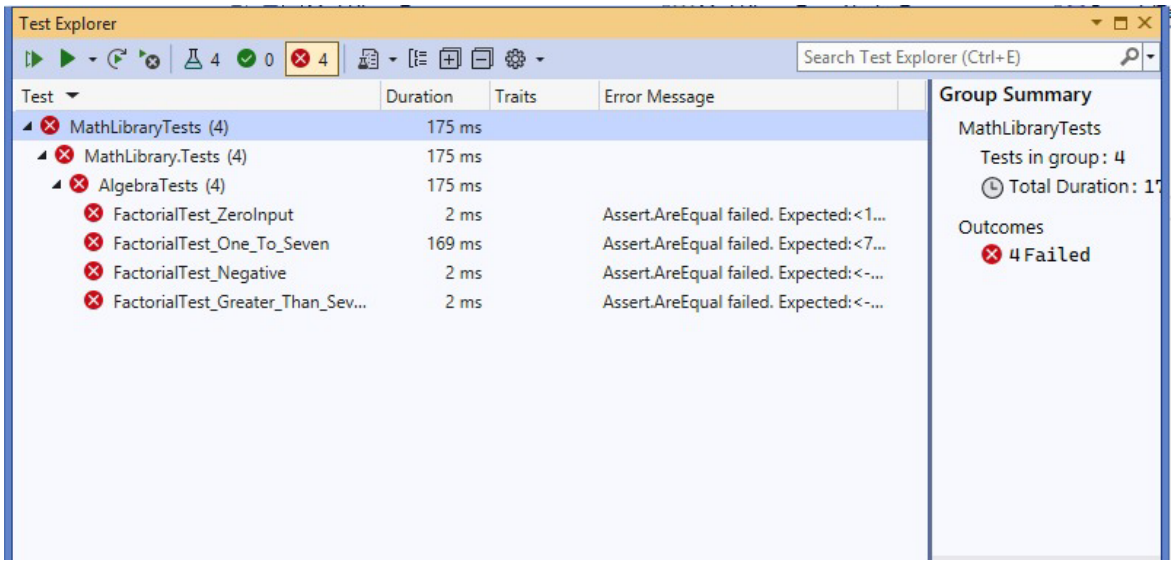
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MathLibrary;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MathLibrary.Tests
{
    [TestClass()]
    public class AlgebraTests
    {
        [TestMethod()]
        public void FactorialTest_ZeroInput()
        {
            // Arrange
            int n = 0;
            int expected = 1;

            // Act
            int actual = Algebra.Factorial(n);


            // Assert
            Assert.AreEqual(expected, actual);
        }

        [TestMethod()]
        public void FactorialTest_Negative_()
        {
            // Arrange
            int n = -2;
            int expected = -9999;

            // Act
            int actual = Algebra.Factorial(n);


            // Assert
            Assert.AreEqual(expected, actual);
        }

        [TestMethod()]
        public void FactorialTest_Greater_Than_Seven()
        {
            // Arrange
            int n = 9;
            int expected = -999;
```

```
        // Act
        int actual = Algebra.Factorial(n);


        // Assert
        Assert.AreEqual(expected, actual);
    }

    [TestMethod()]
    public void FactorialTest_One_To_Seven()
    {
        // Arrange
        int n = 6;
        int expected = 720;

        // Act
        int actual = Algebra.Factorial(n);


        // Assert
        Assert.AreEqual(expected, actual);
    }

    [TestMethod()]
    public void PalindromeTest()
    {
        // Arrange
        int n = 212;
        string expected = "Palindrome";


        // Act
        string actual = "Palindrome";


        // Assert
        Assert.AreEqual(expected,actual);
    }
    [TestMethod()]
    public void Not_PalindromeTest()
    {
        // Arrange
        int n = 213;
        string expected = "Not Palindrome";


        // Act
        string actual = "Not Palindrome";


        // Assert
```

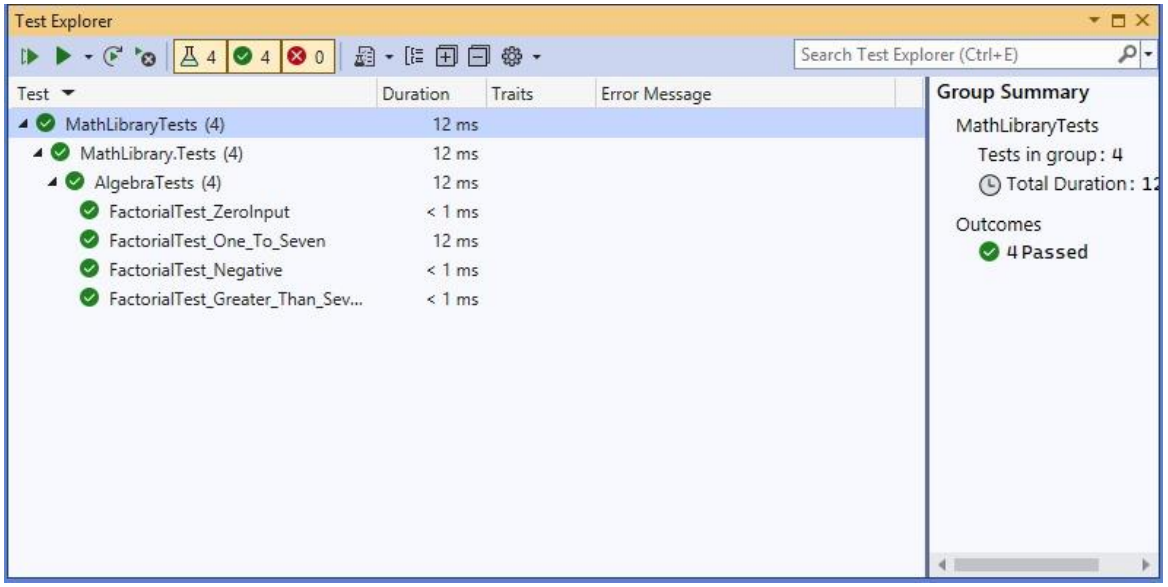
```
            Assert.AreEqual(expected, actual);
        }
    }
}
```

Output :

Failed : RED



Passed : GREEN



8. Add one more method to check if the number is palindrome or not in the above

Algebra class and write test case for the same

Code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MathLibrary
{
    // Author : Praveen Chakravarthi
    // Purpose : Layered Architecture, Implementing TDD
   public class Algebra
   {
      /// <summary>
      /// This Method returns Factorial of a Number
      /// </summary>
      public static int Factorial(int n)
      {
         if (n== 0)
            return 1;

         else if (n < 0)
            return -9999;

         else if (n > 7)
            return -999;

         int fact = 1;
         for (int i =1; i <= n; i++)
            fact *= i;
         return fact;

      }
      public string Palindrome(int n)
      {
         int m, rem, rev = 0;
         m = n;
         while (m>0)
         {
            rem = m % 10;
            rev = (rev * 10) +rem;
            m = m / 10;
         }
         if (n == rev)
            return "Palindrome";
         else
            return "Not Palindrome";
      }
   }
```

```
}
```

Output :

Passed : GREEN