Day 18 Assignment

By

B.P.N.V.S. Sudheer

16-02-22

|  |
| --- |
| 1. What is the use of XML |
| * Xml stands for Extensible markup language it is used to send data across different platform * XML is one of the most widely-used formats for sharing structured information today: between programs, between people, between computers and people, both locally and across networks. If you are already familiar with HTML, you can see that XML is very similar |
| 2. Write the points discussed about xml in the class |
| * Xml is used for universal data transfer mechanism to send data across different platforms * Xml will have user – defined tag * Xml can have one root tag * Xml is case sensitive |
| 3. Create a simple xml to illustrate: a. Tag based xml with 10 products b. Attribute based xml |
| (a)Tag based xml with 10 products |
| <products>  <product>  <id>1</id>  <name>Realme</name>  <price>100</price>  </product>  <product>  <id>2</id>  <name>Realme 8</name>  <price>200</price>  </product>  <product>  <id>3</id>  <name>Realme 9</name>  <price>300</price>  </product>  <product>  <id>4</id>  <name>Mi 4</name>  <price>400</price>  </product>  <product>  <id>5</id>  <name>Mi 5</name>  <price>500</price>  </product>  <product>  <id>6</id>  <name>Mi 6</name>  <price>600</price>  </product>  <product>  <id>7</id>  <name>Mi 7</name>  <price>700</price>  </product>  <product>  <id>8</id>  <name>Mi 8</name>  <price>800</price>  </product>  <product>  <id>9</id>  <name>Mi 9</name>  <price>900</price>  </product>  <product>  <id>10</id>  <name>Mi 10</name>  <price>100</price>  </product>  </products> |
| Browser Output |

|  |
| --- |
|  |
| (b) Attribute based xml |
| <products>  <product id = "1" name = "mi4" price = "1000"> </product>  <product id = "2" name = "mi5" price = "2000"> </product>  <product id = "3" name = "mi6" price = "3000"> </product>  <product id = "4" name = "mi7" price = "5000"></product>  <product id = "5" name = "mi8" price = "6000"> </product>  <product id = "6" name = "mi9" price = "7000"> </product>  <product id = "7" name = "mi10" price = "8000"> </product>  <product id = "8" name = "mi11" price = "9000"> </product>  <product id = "9" name = "mipro" price = "10000"> </product>  <product id = "10" name = "mi11pro" price = "11000"> </product>  <product id = "11" name = "mimax" price = "12000"> </product>  </products> |
| Browser Output: |
|  |
| 4. Convert the above xml to JSON and display the JSON data |
| [ { "id": "1", "name": "Realme", "price": "100" }, { "id": "2", "name": "Realme 8", "price": "200" }, { "id": "3", "name": "Realme 9", "price": "300" }, { "id": "4", "name": "Mi 4", "price": "400" }, { "id": "5", "name": "Mi 5", "price": "500" }, { "id": "6", "name": "Mi 6", "price": "600" }, { "id": "7", "name": "Mi 7", "price": "700" }, { "id": "8", "name": "Mi 8", "price": "800" }, { "id": "9", "name": "Mi 9", "price": "900" }, { "id": "10", "name": "Mi 10", "price": "100" }] |

|  |
| --- |
| [ { "@id": "1", "@name": "mi4", "@price": "1000" }, { "@id": "2", "@name": "mi5", "@price": "2000" }, { "@id": "3", "@name": "mi6", "@price": "3000" }, { "@id": "4", "@name": "mi7", "@price": "5000" }, { "@id": "5", "@name": "mi8", "@price": "6000" }, { "@id": "6", "@name": "mi9", "@price": "7000" }, { "@id": "7", "@name": "mi10", "@price": "8000" }, { "@id": "8", "@name": "mi11", "@price": "9000" }, { "@id": "9", "@name": "mipro", "@price": "10000" }, { "@id": "10", "@name": "mi11pro", "@price": "11000" }, { "@id": "11", "@name": "mimax", "@price": "12000" }] |
| 5. Research and write the benefits of JSON over XML ( 2 or 3 points ) |
| * In most scenarios, JSON is undoubtedly easier to read in its expanded form than XML. * JSON can have a substantially lower character count reducing the overhead in data transfers. * JSON is much easier to parse |
| 6. For the below requirement, create a layered  architectureproject with seperate 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 andproject (solution explorer) screen shot |
| Console Application |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace mathsLibrary  {  public class Algebra  {  public static int Factorial(int n)  {  int fact = 1;  if (n == 0)  return 1;  else if (n > 7)  return -999;  else if (n < 0)  return -9999;  else  {  for (int i = 1; i <= n; i++)    fact = fact \* i;    return fact;    }  }  }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using mathsLibrary;    namespace \_18dayproject1  {  internal class Program  {  static void Main(string[] args)  {  int n;  Console.WriteLine("enter a number");  n = Convert.ToInt32(Console.ReadLine());  Console.WriteLine(Algebra.Factorial(n));  Console.ReadLine();  }  }  } |
| Output: |

|  |
| --- |
|  |
| Desktop Application |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;    using mathsLibrary;    namespace app  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }    private void Form1\_Load(object sender, EventArgs e)  {    }    private void button1\_Click(object sender, EventArgs e)  {  int n = Convert.ToInt32(textBox1.Text);  int result = Algebra.Factorial(n);    textBox2.Text = result.ToString();    }    private void textBox1\_TextChanged(object sender, EventArgs e)  {    }  }  } |
| 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 :  using Microsoft.VisualStudio.TestTools.UnitTesting;  using mathsLibrary;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace mathsLibrary.Tests  {  [TestClass()]  public class AlgebraTests  {  [TestMethod()]  public void FactorialTest\_zero\_input()  {  //Arrange  int n = 0;  int expected = 1;    //Act  int actual = Algebra.Factorial(n);    //Assert  Assert.AreEqual(expected, actual);    }  [TestMethod()]  public void FactorialTest\_one\_to\_seven\_input()  {  //Arrange  int n = 5;  int expected = 120;    //Act  int actual = Algebra.Factorial(n);    //Assert  Assert.AreEqual(expected, actual);    }  [TestMethod()]  public void FactorialTest\_Negative\_input()  {  //Arrange  int n = -1;  int expected = -9999;    //Act  int actual = Algebra.Factorial(n);    //Assert  Assert.AreEqual(expected, actual);    }  [TestMethod()]  public void FactorialTest\_greater\_than\_seven()  {  //Arrange  int n = 8;  int expected = -999;    //Act  int actual = Algebra.Factorial(n);    //Assert  Assert.AreEqual(expected, actual);    }  }  } |
|  |
|  |