WEB SERVICE AND XML LAB - [IT P72]

Academic Year: 2020-21 ODD SEM

EX NO	LIST OF EXPERIMENTS
1	Study of XML
2	XML Document is Well-Formed or not using DOM Parser in Java
3	Reading XML document using DOM Parser in Java
4	XML Document is Well-Formed or not using SAX Parser in Java
5	Reading XML document using SAX Parser in Java
6	Creating or Writing in XML document using DOM Parser in C#.Net
7	Reading XML document using DOM Parser in C#.Net
8	Displaying XML file using CSS
9	Displaying XML file using XSLT
10	Study of EJB
11	Developing Stateless Components Using Ejb Component Technology
12	Developing a Components Using Ejb Component Technology
13	Study of WebService
14	Invoking of EJB Components as Web services
15	Invoking of .Net Components as Web services
16	Developing a J2EE Client to access .Net Web Services
17	Developing a .Net Client to access J2EE Web Services

STAFF INCHARGE HOD PRINCIPAL

2. XML Document is Well-Formed or not using DOM Parser in Java.

Aim:
To write the java program to perform the DOM Parser with XML.

Algorithm:

STEP 1: Start the process.
STEP 2: Open the NetBeans IDE 8.0.2.
STEP 3: Create the new application by click File->New project->java
STEP 4: Import the necessary java package and DOM related package.
STEP 5: Create an object for Document Builder factory ,Document builder and Input source.
STEP 6: Create an XML file or use the already existing xml file to check the xml is well formed or not.
STEP 7: If the object get an expected xml file, data will be displayed in well formed.
STEP 8: If XML file has any error then the data in the Xml file will said to be not well formed.
STEP 9: Stop the process.

SOURCE CODE:

ParsingDomDemo.java

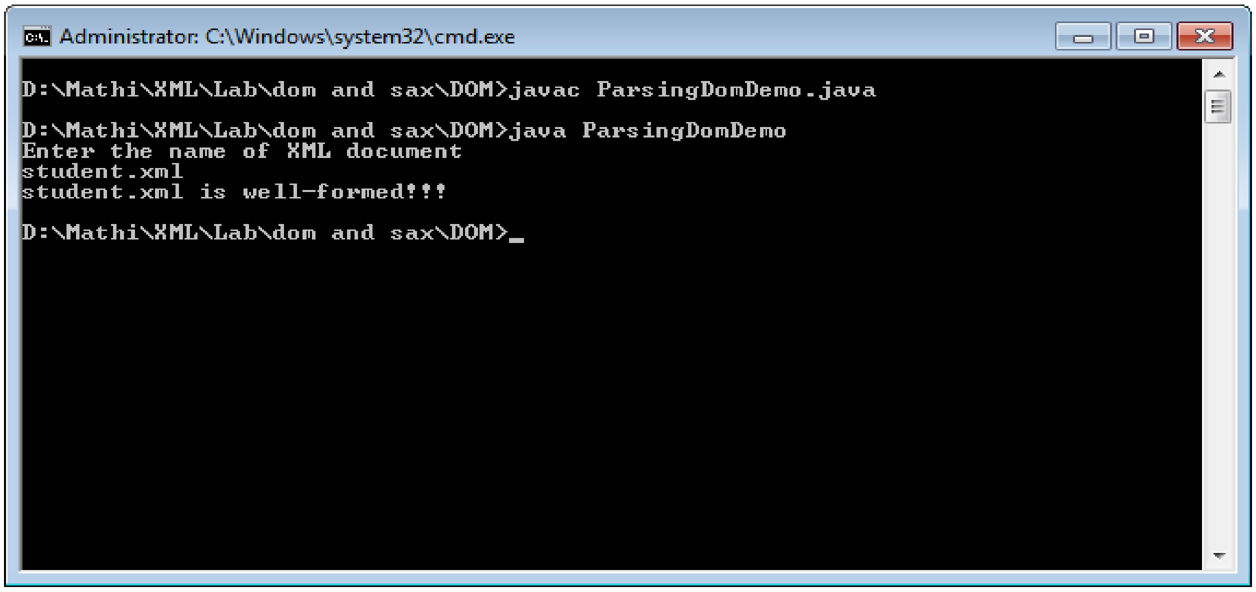
```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import java.util.*;

public class ParsingDomDemo
{
        public static void main(String[] arg)
        {
        try
        {
                System.out.println("Enter the name of XML document");
                Scanner s=new Scanner(System.in);
                String file_name=s.nextLine();
                File fp=new File(file_name);
if(fp.exists())
                {
                try
                {
                DocumentBuilderFactory Factory_obj=DocumentBuilderFactory.newInstance();
                DocumentBuilder builder=Factory_obj.newDocumentBuilder();
                InputSource ip_src=new InputSource(file_name);
                Document doc = builder.parse(ip_src);
                System.out.println(file_name+"is well-formed!!!");
                }
                catch(Exception e)
                {
                        System.out.println(file_name+"isn't well-formed!");
                        System.exit(1);
                }
                }
                else
                {
                        System.out.print("File not found");
                }
        }
        catch(IOException ex)
        {
                ex.printStackTrace();
        }
        }
}
```

Student.xml:-

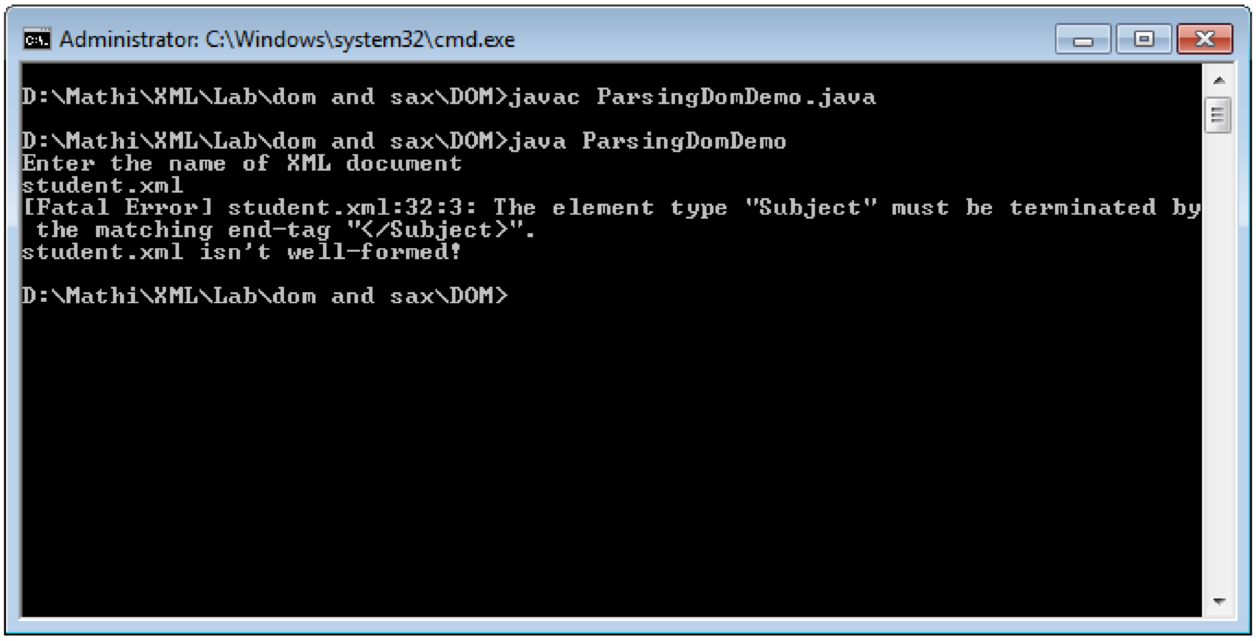
```
<?xml version="1.0"?>
<student>
        <Roll_No>10</Roll_No>
        <Personal_Info>
                <Name>Parth</Name>
                <Address>Pune</Address>
                <Phone>123456</Phone>
        </Personal_Info>
        <Class>Second</Class>
        <Subject>Mathematics</Subject>
        <Marks>100</Marks>

        <Roll_No>20</Roll_No>
        <Personal_Info>
                <Name>Anuradha</Name>
                <Address>Banglore</Address>
                <Phone>156438</Phone>
        </Personal_Info>
        <Class>Fifth</Class>
        <Subject>English</Subject>
        <Marks>90</Marks>

        <Roll_No>30</Roll_No>
        <Personal_Info>
                <Name>Anandh</Name>
                <Address>Mumbai</Address>
                <Phone>7678453</Phone>
        </Personal_Info>
```

```
<Class>Fifth</Class>
<Subject>English</Subject>
<Marks>90</Marks>
</student>
```

Output:-



```
D:\Mathi\XML\Lab\dom and sax\DOM>javac ParsingDomDemo.java

D:\Mathi\XML\Lab\dom and sax\DOM>java ParsingDomDemo
Enter the name of XML document
student.xml
student.xml is well-formed!!!

D:\Mathi\XML\Lab\dom and sax\DOM>_
```



```
D:\Mathi\XML\Lab\dom and sax\DOM>javac ParsingDomDemo.java

D:\Mathi\XML\Lab\dom and sax\DOM>java ParsingDomDemo
Enter the name of XML document
student.xml
[Fatal Error] student.xml:32:3: The element type "Subject" must be terminated by
 the matching end-tag "</Subject>".
student.xml isn't well-formed!

D:\Mathi\XML\Lab\dom and sax\DOM>
```

3. Reading XML document using DOM Parser in Java

AIM:
To write a program to parse XML document using SAX parser.

ALGORITHM:

STEP 1: Start the process.
STEP 2: Open the NetBeans IDE 8.0.2.
STEP 3: Create the new application by click File->New project->java
STEP 4: Right click the application choose New->xml document to create the xml.
STEP 5: Then double click the .java file in the application.
STEP 6: Import the necessary java package and DOM related package.
STEP 7: Create an object for DOM parser to check the XML file
STEP 8: Then check for the specific values or function and the default handler function.
STEP 9: Create an object for the Document Builder factory, Document builder, Node list in order to perform the operation.
STEP 10: Get the Xml file name in order to display.
STEP 11: Check the elements and get the values of each and every elements and the print the result.
STEP 12: Verify the result.
STEP 13: Stop the process.

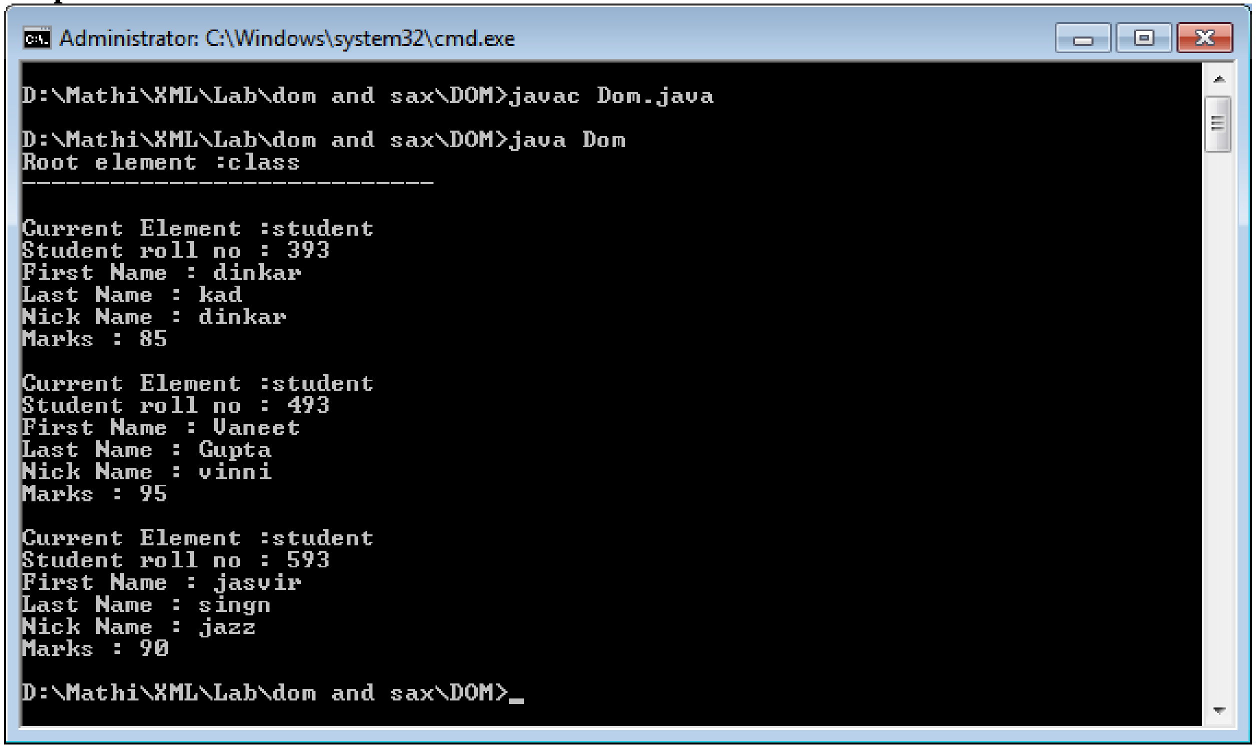
SOURCE CODE:

Dom.java
```
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

public class Dom
{
public static void main(String[] args)
{
try {
File inputFile = new File("XMLdom.xml");
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(inputFile);
doc.getDocumentElement().normalize();
System.out.println("Root element :" + doc.getDocumentElement().getNodeName());
NodeList nList = doc.getElementsByTagName("student");
System.out.println("----------------------------");
for (int temp = 0; temp < nList.getLength(); temp++)
{
Node nNode = nList.item(temp);
System.out.println("\nCurrent Element :" + nNode.getNodeName());
if (nNode.getNodeType() == Node.ELEMENT_NODE)
{
 Element eElement = (Element) nNode;
 System.out.println("Student roll no : " + eElement.getAttribute("rollno"));
 System.out.println("First Name : " +
eElement.getElementsByTagName("firstname").item(0).getTextContent());
System.out.println("Last Name : " +
eElement.getElementsByTagName("lastname").item(0).getTextContent());
System.out.println("Nick Name : " +
eElement.getElementsByTagName("nickname").item(0).getTextContent());
System.out.println("Marks : " +
eElement.getElementsByTagName("marks").item(0).getTextContent());
}
}
}catch (Exception e)
{
e.printStackTrace();
}
}
}
```

XMLdom.xml

```
<?xml version="1.0"?>
<class>
<student rollno="393">
<firstname>dinkar</firstname>
<lastname>kad</lastname>
<nickname>dinkar</nickname>
<marks>85</marks>
</student>
<student rollno="493">
<firstname>Vaneet</firstname>
<lastname>Gupta</lastname>
<nickname>vinni</nickname>
<marks>95</marks>
</student>
<student rollno="593">
<firstname>jasvir</firstname>
<lastname>singn</lastname>
<nickname>jazz</nickname>
<marks>90</marks>
</student>
</class>
```

Output:-


Administrator: C:\Windows\system32\cmd.exe

```
D:\Mathi\XML\Lab\dom and sax\DOM>javac Dom.java

D:\Mathi\XML\Lab\dom and sax\DOM>java Dom
Root element :class
----------------------------

Current Element :student
Student roll no : 393
First Name : dinkar
Last Name : kad
Nick Name : dinkar
Marks : 85

Current Element :student
Student roll no : 493
First Name : Vaneet
Last Name : Gupta
Nick Name : vinni
Marks : 95

Current Element :student
Student roll no : 593
First Name : jasvir
Last Name : singn
Nick Name : jazz
Marks : 90

D:\Mathi\XML\Lab\dom and sax\DOM>_
```

4. XML Document is Well-Formed or not using SAX Parser in Java

Aim:
To check whether the given XML file is well formed or not using the SAX parser in java..

ALGORITHM:

STEP 1: Start the program.
STEP 2: Open the NetBeans IDE 8.0.2.
STEP 3: Create the new application by click File->New project->java
STEP 4: Import the necessary java package and SAX related Package.
STEP 5: Create an object for the class Sax Parse check.
STEP 6: Create an XML reader object by means which get the input file for the process.
STEP 7: Create an Xml file or use the already existing xml file to check the xml is well formed or not.
STEP 8: Check whether the given XML file is well formed or not.
STEP 9: If Xml file has any error then the data in the Xml file will said to be not well formed.
STEP 10: Print the output.
STEP 11: Stop the program.

SOURCE CODE:

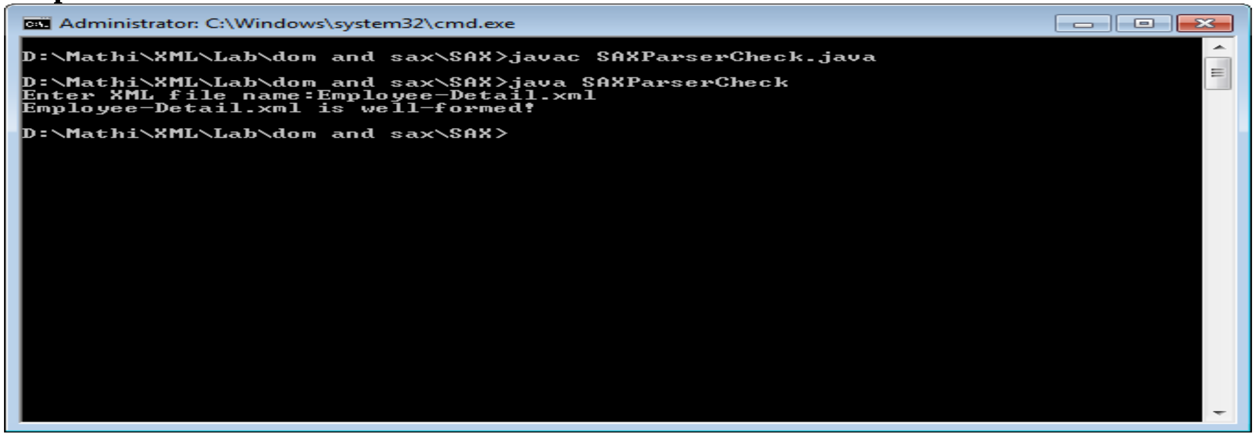
SAXParserCheck.java
```
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;
import java.util.*;

public class SAXParserCheck
{
        public static void main(String[] args) throws IOException
        {
                Scanner s=new Scanner(System.in);
                System.out.print("Enter XML file name:");
                String xmlFile = s.nextLine();
                SAXParserCheck par = new SAXParserCheck(xmlFile);
        }
        public SAXParserCheck(String str)
        {
                try
                {
                        File file = new File(str);
                        if (file.exists())
                        {
                                XMLReader reader = XMLReaderFactory.createXMLReader();
                                reader.parse(str);
                                System.out.println(str + " is well-formed!");
```

```
						}
						else
						{
							System.out.println("File not found: " + str);
						}
				}
				catch (SAXException sax)
				{
						System.out.println(str + " isn't well-formed");
				}
				catch (IOException io)
				{
						System.out.println(io.getMessage());
				}
		}
}
```

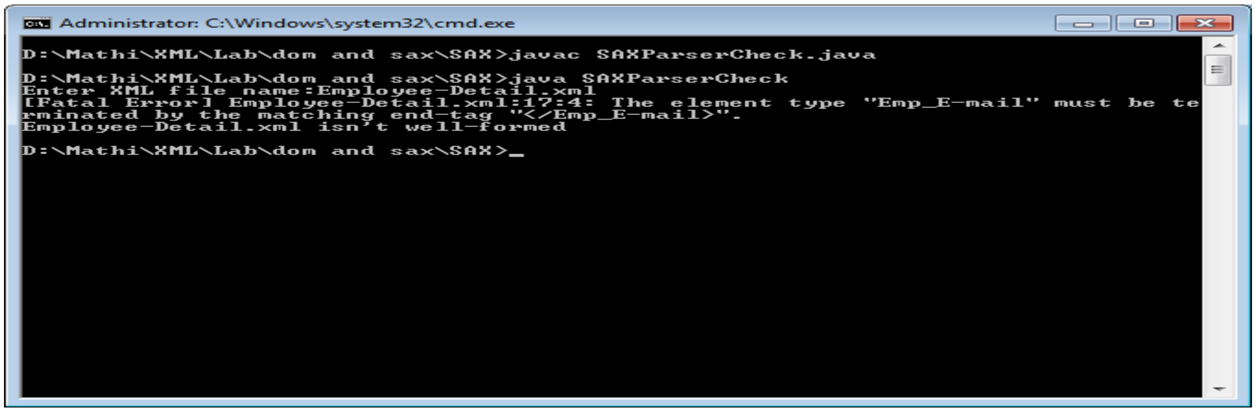
Employee-Detail.xml

```
<?xml version = "1.0" ?>
<Employee-Detail>
	<Employee>
		<Emp_Id> 11032 </Emp_Id>
		<Emp_Name> Hari </Emp_Name>
	<Emp_E-mail> harideivasigamani@gmail.com </Emp_E-mail>
	</Employee>
	<Employee>
		<Emp_Id> 11022 </Emp_Id>
		<Emp_Name> Ashok kumar </Emp_Name>
		<Emp_E-mail> ashokkumar782@gmail.com </Emp_E-mail>
	</Employee>
	<Employee>
		<Emp_Id> 11011 </Emp_Id>
		<Emp_Name> Elavarasan </Emp_Name>
		<Emp_E-mail> ela@pec.in </Emp_E-mail>
	</Employee>
</Employee-Detail>
```

Output:-


Administrator: C:\Windows\system32\cmd.exe

```
D:\Mathi\XML\Lab\dom and sax\SAX>javac SAXParserCheck.java

D:\Mathi\XML\Lab\dom and sax\SAX>java SAXParserCheck
Enter XML file name:Employee-Detail.xml
Employee-Detail.xml is well-formed!

D:\Mathi\XML\Lab\dom and sax\SAX>
```


Administrator: C:\Windows\system32\cmd.exe

```
D:\Mathi\XML\Lab\dom and sax\SAX>javac SAXParserCheck.java

D:\Mathi\XML\Lab\dom and sax\SAX>java SAXParserCheck
Enter XML file name:Employee-Detail.xml
[Fatal Error] Employee-Detail.xml:17:4: The element type "Emp_E-mail" must be te
rminated by the matching end-tag "</Emp_E-mail>".
Employee-Detail.xml isn't well-formed

D:\Mathi\XML\Lab\dom and sax\SAX>_
```

5. Reading XML document using SAX Parser in Java

AIM:
To write a program to parse XML document using SAX parser.

ALGORITHM:

STEP 1: Start the process.
STEP 2: Open the NetBeans IDE 8.0.2.
STEP 3: Create the new application by click File->New project->java
STEP 4: Right click the application choose New->xml document to create the xml.
STEP 5: Then double click the .java file in the application.
STEP 6: Write the java code and save it.
STEP 7: Create an object for the class SAXParserFactory, SAXParser and for the default handler function.
STEP 8:Use the Element_name.equals() function to perform the condition operation.
STEP 8: Save the whole project and run it.
STEP 9: Verify the result.
STEP 10: Stop the process.

SOURCE CODE:

EmployeeDetails.java
```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;
import java.util.*;
public class EmployeeDetails
{
        public static void main(String[] args) throws IOException
        {
                Scanner s=new Scanner(System.in);
                System.out.print("Enter XML file name:");
                String xmlFile = s.nextLine();
                EmployeeDetails detail = new EmployeeDetails(xmlFile);
        }
        public EmployeeDetails(String str)
        {
                try
                {
                        File file = new File(str);
                        if (file.exists())
                        {
                                SAXParserFactory parserFact = SAXParserFactory.newInstance();
                                SAXParser parser = parserFact.newSAXParser();
                                System.out.println("XML Data: ");
```

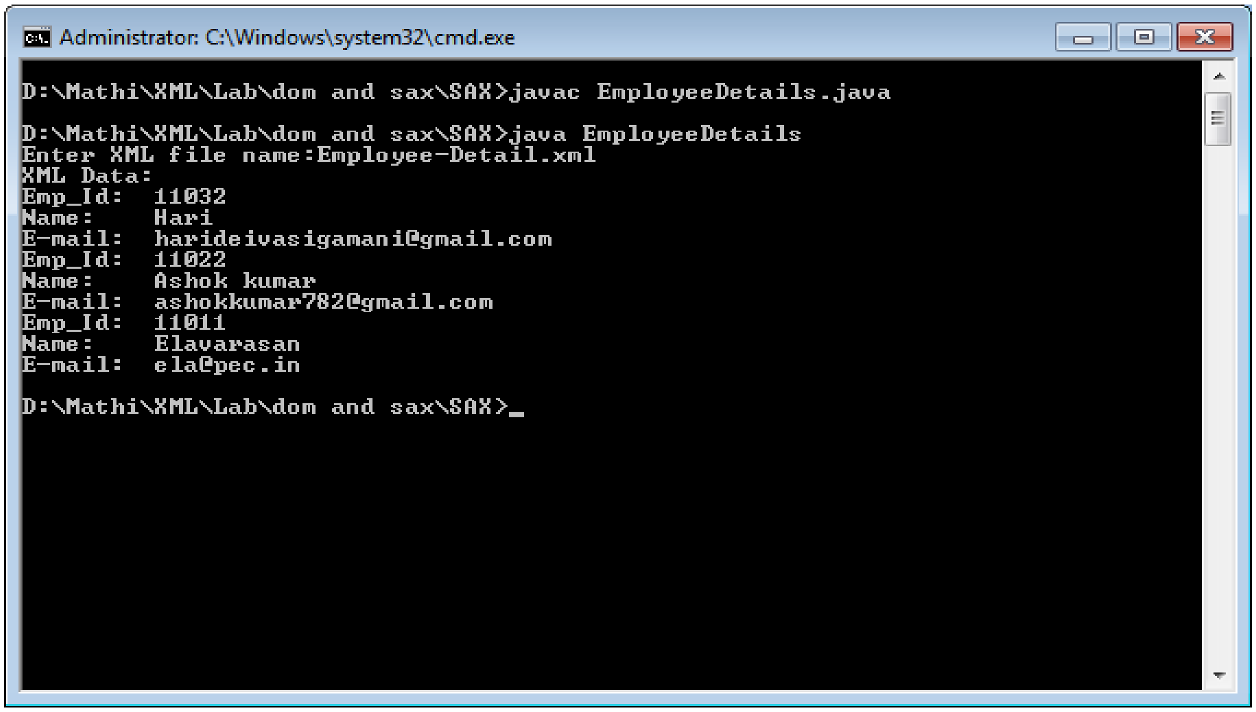
```
		DefaultHandler dHandler = new DefaultHandler()
		{
			boolean id;
			boolean name;
			boolean mail;

			public void startElement(String uri, String localName,
		String element_name, Attributes attributes)throws SAXException
		{
			if (element_name.equals("Emp_Id"))
			{
				id = true;
			}
			if (element_name.equals("Emp_Name"))
			{
				name = true;
			}
			if (element_name.equals("Emp_E-mail"))
			{
				mail = true;
			}
		}

		public void characters(char[] ch, int start, int len) throws
	SAXException
		{
			String str = new String (ch, start, len);
			if (id)
			{
				System.out.println("Emp_Id: "+str);
				id = false;
			}
			if (name)
			{
				System.out.println("Name:   "+str);
				name = false;
			}
			if (mail)
			{
				System.out.println("E-mail: "+str);
				mail = false;
			}
		}
		};

		parser.parse(str, dHandler);
	}
	else
```

```
				{
					System.out.println("File not found!");
				}
			}
			catch (Exception e)
			{
				System.out.println("XML File hasn't any elements");
				e.printStackTrace();
			}
		}
}
```

Employee-Detail.xml

```
<?xml version = "1.0" ?>
<Employee-Detail>
	<Employee>
		<Emp_Id> 11032 </Emp_Id>
		<Emp_Name> Hari </Emp_Name>
	<Emp_E-mail> harideivasigamani@gmail.com </Emp_E-mail>
	</Employee>
	<Employee>
		<Emp_Id> 11022 </Emp_Id>
		<Emp_Name> Ashok kumar </Emp_Name>
		<Emp_E-mail> ashokkumar782@gmail.com </Emp_E-mail>
	</Employee>
	<Employee>
		<Emp_Id> 11011 </Emp_Id>
		<Emp_Name> Elavarasan </Emp_Name>
		<Emp_E-mail> ela@pec.in </Emp_E-mail>
	</Employee>
</Employee-Detail>
```

Output:-



```
D:\Mathi\XML\Lab\dom and sax\SAX>javac EmployeeDetails.java

D:\Mathi\XML\Lab\dom and sax\SAX>java EmployeeDetails
Enter XML file name:Employee-Detail.xml
XML Data:
Emp_Id:  11032
Name:    Hari
E-mail:  harideivasigamani@gmail.com
Emp_Id:  11022
Name:    Ashok kumar
E-mail:  ashokkumar782@gmail.com
Emp_Id:  11011
Name:    Elavarasan
E-mail:  ela@pec.in

D:\Mathi\XML\Lab\dom and sax\SAX>_
```

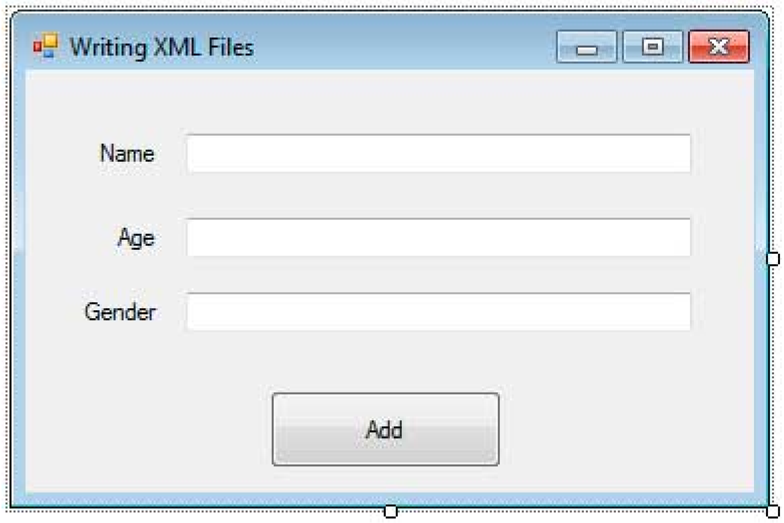
6. Creating or Writing in XML document using DOM Parser in C#.Net

Aim:
To Create or Writing in XML document using DOM Parser in C#.Net .

ALGORITHM:

STEP 1: Start the process.
STEP 2: Open the Microsoft visual studio .
STEP 3: Choose File->New Project->Create Windows Platform.
STEP 4: Design the Form.cs using the Form design process and create the WindowsFormsApplication1.
STEP 5: Initialize Component which are needed like TextBox, TextField, Button.
STEP 6: Create an xml document, If there is no current file.
STEP 7: Create necessary fields and add the values(Name, Age, Gender) for fields.
STEP 8: Save the Form and Run the form.
STEP 9: Enter the values into the Form then click add to the button, Now the entered data in field will be stored in the XML file.
STEP 10: Display the confirmation message by the dialogue box.
STEP 11: Stop the process.

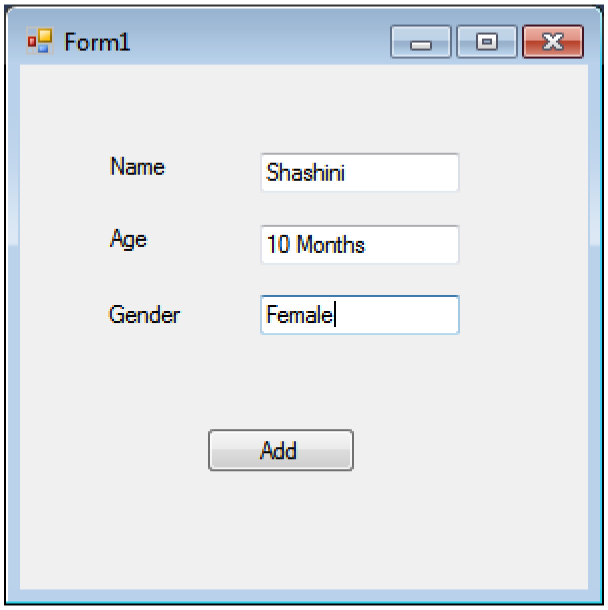
Form.cs[Design]



Form.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;

namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private XmlDocument doc;
      private const string PATH = @"C:\Users\Administrator\Desktop\sample.xml";

      private void button1_Click(object sender, EventArgs e)
      {
         //Create an xml document
         doc = new XmlDocument();

         //If there is no current file, then create a new one
         if (!System.IO.File.Exists(PATH))
         {
            //Create neccessary nodes
            XmlDeclaration declaration = doc.CreateXmlDeclaration("1.0", "UTF-8", "yes");
            XmlComment comment = doc.CreateComment("This is an XML Generated File");
            XmlElement root = doc.CreateElement("Persons");
            XmlElement person = doc.CreateElement("Person");
            XmlAttribute name = doc.CreateAttribute("name");
            XmlElement age = doc.CreateElement("Age");
            XmlElement gender = doc.CreateElement("Gender");

            //Add the values for each nodes
            name.Value=textBox1.Text;
            age.InnerText = textBox2.Text;
            gender.InnerText = textBox3.Text;

            //Construct the document
            doc.AppendChild(declaration);
            doc.AppendChild(comment);
            doc.AppendChild(root);
            root.AppendChild(person);
            person.Attributes.Append(name);
            person.AppendChild(age);
            person.AppendChild(gender);

            doc.Save(PATH);
         }
         else //If there is already a file
         {
            //Load the XML File
            doc.Load(PATH);

            //Get the root element
            XmlElement root = doc.DocumentElement;

            XmlElement person = doc.CreateElement("Person");
            XmlAttribute name = doc.CreateAttribute("name");
            XmlElement age = doc.CreateElement("Age");
            XmlElement gender = doc.CreateElement("Gender");

            //Add the values for each nodes
            name.Value = textBox1.Text;
            age.InnerText = textBox2.Text;
            gender.InnerText = textBox3.Text;

            //Construct the Person element
            person.Attributes.Append(name);
            person.AppendChild(age);
            person.AppendChild(gender);

            //Add the New person element to the end of the root element
            root.AppendChild(person);

            //Save the document
            doc.Save(PATH);
         }

         //Show confirmation message
         MessageBox.Show("Details have been added to the XML File.");

         //Reset text fields for new input
         textBox1.Text = String.Empty;
         textBox2.Text = String.Empty;
         textBox3.Text = String.Empty;
      }
   }
}
```

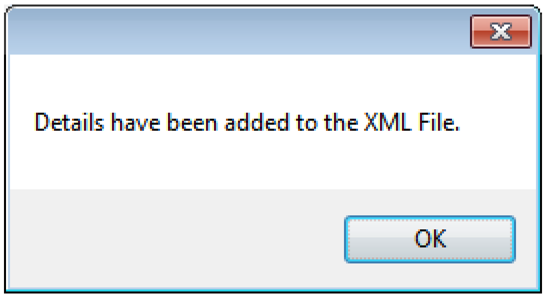
OUTPUT:



Name Shashini

Age 10 Months

Gender Female



Details have been added to the XML File.

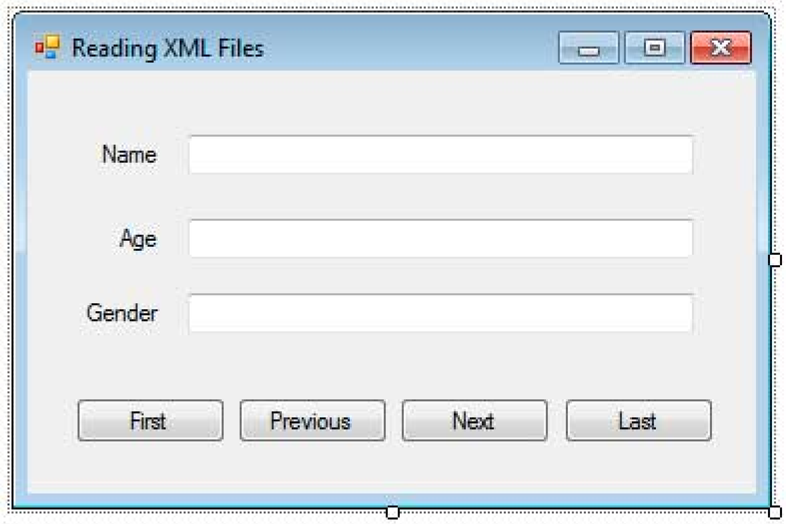
7. Reading XML document using DOM Parser in C#.Net

Aim:
To Read data from the XML document using DOM Parser in C#.Net .

ALGORITHM:

STEP 1: Start the process.
STEP 2: Open the Microsoft visual studio .
STEP 3: Choose File->New Project->Create Windows Platform.
STEP 4: Design the Form.cs using the Form design process and create the WindowsFormsApplication1.
STEP 5: Initialize Component which are needed TextBox, TextField, Button.
STEP 6: Get the XML file path by means of an object and access the data.
STEP 7: Keep an action to every button in the form such as First, Previous, Last, Next.
STEP 8: Use which operation is to be performed the action will be taken place.
STEP 9: The data will be displayed in from which is designed.
STEP 10 :Stop the process.

Form.cs [Design]

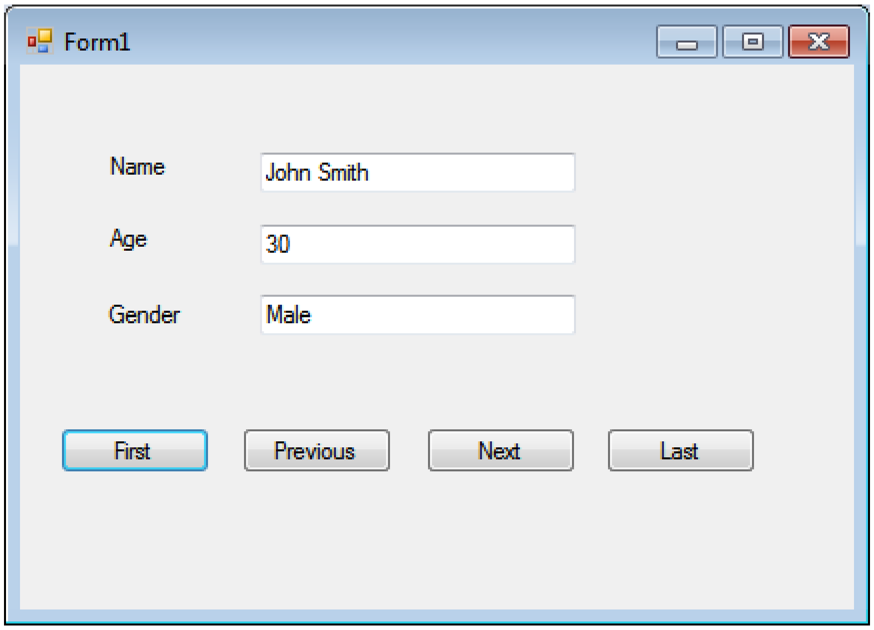


Form.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;
```

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private XmlDocument doc;
        private XmlElement root;
        private XmlElement currentPerson;
        private const string PATH = @"C:\Users\Administrator\Desktop\sample.xml";
        private int current = 0;
        private int max;

        private void ShowDetails(XmlElement currentPerson)
        {
            textBox1.Text = currentPerson.Attributes["name"].Value;
            textBox2.Text = currentPerson.GetElementsByTagName("Age")[0].InnerText;
            textBox3.Text = currentPerson.GetElementsByTagName("Gender")[0].InnerText;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            doc = new XmlDocument();
            doc.Load(PATH);

            //Get root element
            root = doc.DocumentElement;

            //Determine maximum possible index
            max = root.GetElementsByTagName("Person").Count - 1;

            //Get the record at the current index
            currentPerson = (XmlElement)root.ChildNodes[current];

            //Show the record information
            ShowDetails(currentPerson);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            current = 0;
            currentPerson = (XmlElement)root.ChildNodes[current];
            ShowDetails(currentPerson);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            current = (current - 1 < 0) ? 0 : current - 1;
            currentPerson = (XmlElement)root.ChildNodes[current];
            ShowDetails(currentPerson);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            current = (current + 1 > max) ? max : current + 1;
            currentPerson = (XmlElement)root.ChildNodes[current];
            ShowDetails(currentPerson);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            current = max;
            currentPerson = (XmlElement)root.ChildNodes[current];
            ShowDetails(currentPerson);
        }
    }
}
```

OUTPUT:



Name John Smith
Age 30
Gender Male

8. Displaying XML file using CSS

Aim:
To Display XML file using the CSS.

ALGORITHM:

STEP 1: Start the program.
STEP 2: Create a XML file with the required attributes and variables.
STEP 3: Use the style sheet by declaring it in the “text/css” at the header.
STEP 4: Create <Cd> node and get the details for the XML file.
STEP 5: Create an external Catalog.Css file in which add the graphic components such as margin: , display: , color: ,font-size.
STEP 6: Embed the External.css file in cd_catalog.xml file using xml-stylesheet element.
STEP 7: Print the output.
STEP 8: Stop the program.

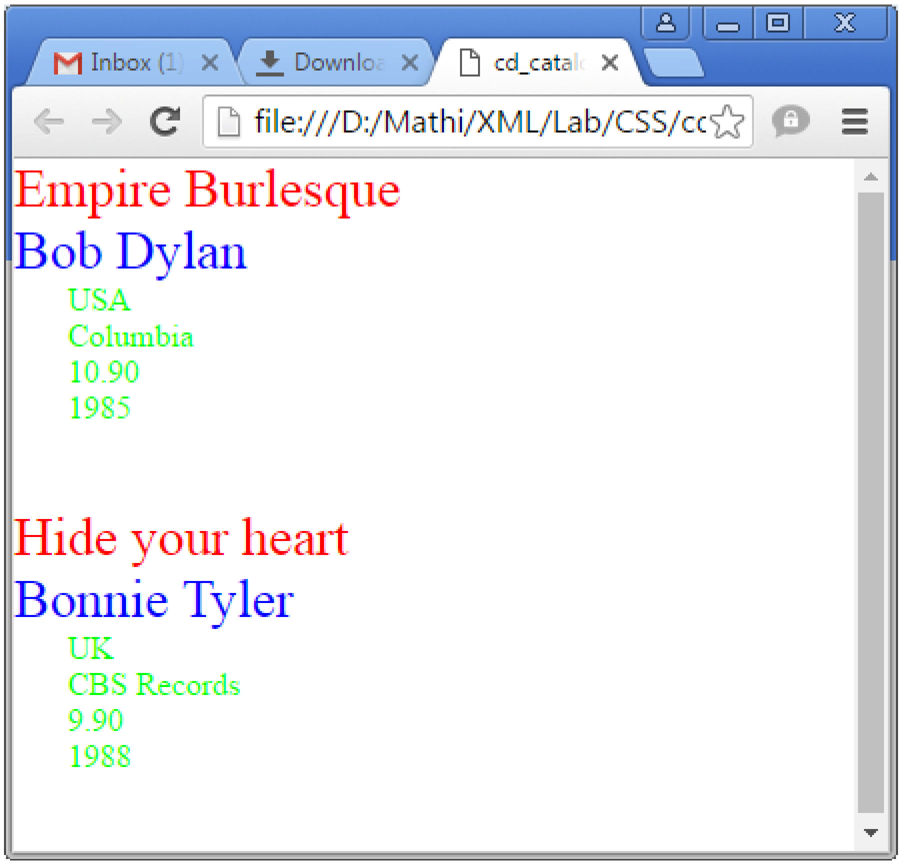
cd_catalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
        <CD>
                <TITLE>Empire Burlesque</TITLE>
                <ARTIST>Bob Dylan</ARTIST>
                <COUNTRY>USA</COUNTRY>
                <COMPANY>Columbia</COMPANY>
                <PRICE>10.90</PRICE>
                <YEAR>1985</YEAR>
        </CD>
        <CD>
                <TITLE>Hide your heart</TITLE>
                <ARTIST>Bonnie Tyler</ARTIST>
                <COUNTRY>UK</COUNTRY>
                <COMPANY>CBS Records</COMPANY>
                <PRICE>9.90</PRICE>
                <YEAR>1988</YEAR>
        </CD>
</CATALOG>
```

cd_catalog.css

```
CATALOG {
   background-color: #ffffff;
   width: 100%;
}
CD {
   display: block;
   margin-bottom: 30pt;
   margin-left: 0;
}
TITLE {
    display: block;
    color: #ff0000;
    font-size: 20pt;
}
ARTIST {
    display: block;
    color: #0000ff;
    font-size: 20pt;
}
COUNTRY, PRICE, YEAR, COMPANY {
    display: block;
    color: #00ff00;
    margin-left: 20pt;
}
```

Output:-



```
Empire Burlesque
Bob Dylan
    USA
    Columbia
    10.90
    1985


Hide your heart
Bonnie Tyler
    UK
    CBS Records
    9.90
    1988
```

9. Displaying XML file using XSLT

Aim:
To Display the XML file using the XSLT.

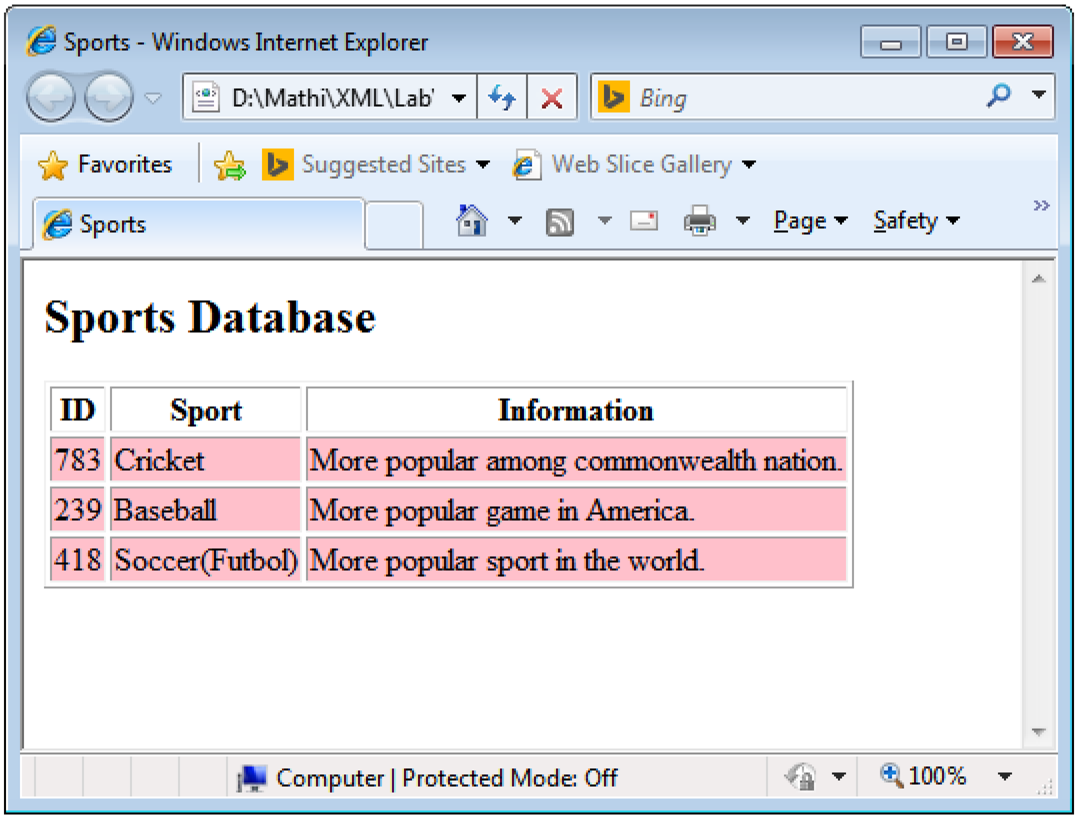
ALGORITHM:

STEP 1: Start the program.
STEP 2: Create a XML file “Sports.xml” with the required attributes name, paragraph and id to get all details about the sport.
STEP 3: Create a “sports.xsl” file in which use the style sheet to provide the necessary design to the xml file.
STEP 4: Embed the sports.xsl file in sport.xml file using xml-stylesheet element.
STEP 5: By creating the another “Sports1.xsl” file with the necessary information of the sport will be displayed.
STEP 6: use the xsl:value-of select=” ” attribute to fetch the id, name and paragraph values.
STEP 6: Print the output.
STEP 7: Stop the program.

sports.xml
```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="sports.xsl"?>
<sports>
        <game id="783">
                <name>Cricket</name>
                <paragraph>
                        More popular among commonwealth nation.
                </paragraph>
        </game>
        <game id="239">
                <name>Baseball</name>
                <paragraph>
                        More popular game in America.
                </paragraph>
        </game>
        <game id="418">
                <name>Soccer(Futbol)</name>
                <paragraph>
                        More popular sport in the world.
                </paragraph>
        </game>
</sports>
```

sports.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="/">
                <html>
                        <head><title>Sports</title></head>
                        <body>
                                <h2>Sports Database</h2>
                                <table border ="1">
                                        <tr>
                                                <th>ID</th>
                                                <th>Sport</th>
                                                <th>Information</th>
                                        </tr>
                                        <xsl:for-each select="sports/game">
                                        <tr bgcolor="pink">
                                                <td><xsl:value-of select="@id"/></td>
                                                <td><xsl:value-of select="name"/></td>
                                                <td><xsl:value-of select="paragraph"/></td>
                                        </tr>
                                        </xsl:for-each>
                                </table>
                        </body>
                </html>
        </xsl:template>
</xsl:stylesheet>
```

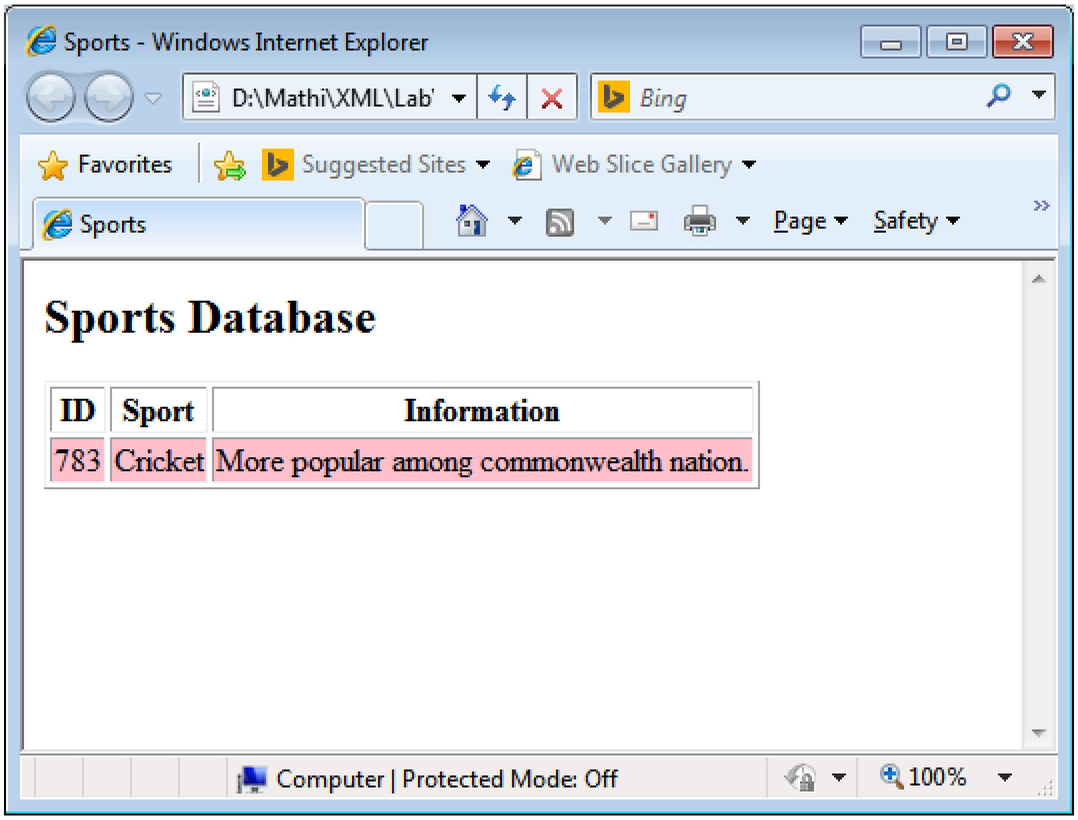


Sports Database

ID	Sport	Information
783	Cricket	More popular among commonwealth nation.
239	Baseball	More popular game in America.
418	Soccer(Futbol)	More popular sport in the world.

sports1.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
        <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
                <xsl:template match="/">
                        <html>
                                <head>
                                        <title>Sports</title>
                                </head>
                                <body>
                                        <h2>Sports Database</h2>
                                        <table border ="1">
                                                <tr>
                                                        <th>ID</th>
                                                        <th>Sport</th>
                                                        <th>Information</th>
                                                </tr>
                                                <xsl:for-each
                                        select="sports/game[name='Cricket']">
                                                <tr bgcolor="pink">
                                                        <td><xsl:value-of select="@id"/></td>
                                                        <td><xsl:value-of select="name"/></td>
                                                        <td><xsl:value-of
                                                select="paragraph"/></td>
                                                </tr>
                                                </xsl:for-each>
                                        </table>
                                </body>
                        </html>
                </xsl:template>
        </xsl:stylesheet>
```



Sports Database

ID	Sport	Information
783	Cricket	More popular among commonwealth nation.

11. Simple EJB Stateless Program for addition of 2 numbers

AIM:

To perform Addition operation using EJB component technology.

ALGORITHM:

1. Start Netbeans and create a new project in Enterprise Application.
2. Create a session bean by choosing stateless session type and inside the session bean create addition methods for invoking EJB Component.
3. Then create servlet page and call the addition methods created in session bean page.
4. Design the initial page in index.jsp and call the servlet file.
5. Then compile and run the index.jsp page.

1. Stateless SessionBean with Remote Interface --- > Java class Library
   [New project—Java EE—EJB Modeule—SessionBean]

```
public class NewSessionBean implements NewSessionBeanRemote {
    @Override
    public int add(int a, int b) {
        return a+b;
    }
}
```

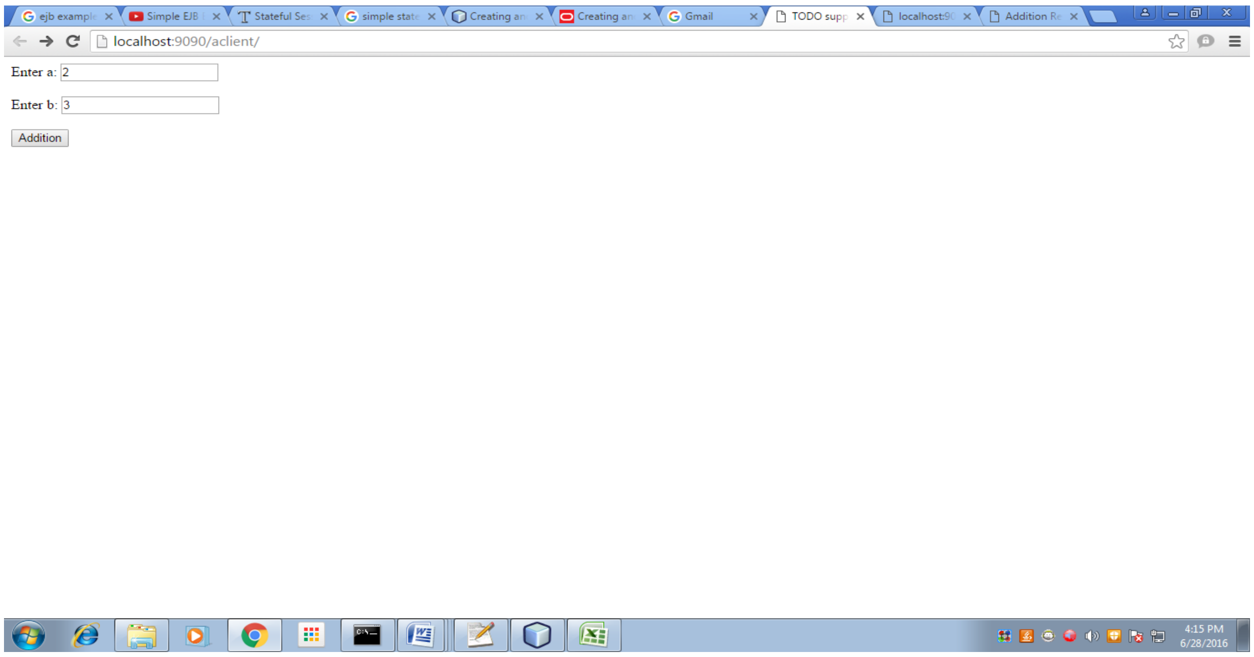
2. Index.html [New Project—Java Web—Web Application]—Client Side

```
<form name="f1" action="a" method="post">
        Enter a: <input type="text" name="t1"><br><br>
Enter b: <input type="text" name="t2"><br><br>
<input type="submit" value="Addition">
</form>
```

3. Under Web Application create Servlet file and also R.Click—Insert code—Call Enterprise Bean and then select the session bean from the file and paste the code in post method.

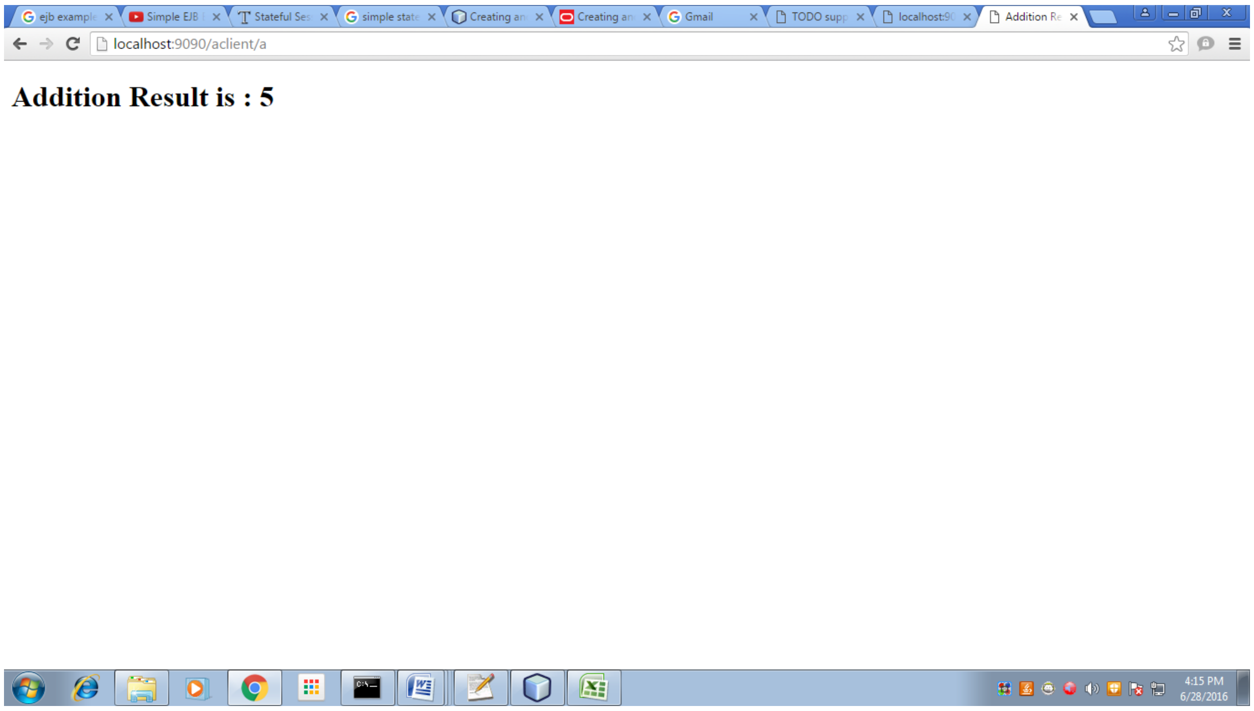
```
int a=Integer.parseInt(request.getParameter("t1"));
int b=Integer.parseInt(request.getParameter("t2"));
int result=newSessionBean.add(a, b);
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter()
try {
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Addition Result</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Addition Result is : "+result+"</h1>");
out.println("</body>");
out.println("</html>"); }
```

Output:-



Enter a: 2

Enter b: 3



Addition Result is : 5

12. DEVELOPING COMPONENTS USING EJB COMPONENT TECHNOLOGY

AIM:

To perform Factorial operation using EJB component technology.

ALGORITHM:

1. Start Netbeans and create a new project in Enterprise Application.
2. Create a session bean and inside the session bean create factorial methods for invoking EJB Component.
3. Then create servlet page and call the factorial methods created in session bean page.
4. Design the initial page in index.jsp and call the servlet file.
5. Then compile and run the index.jsp page.

STEPS TO CREATE SESSION BEAN

1. Start the Netbeans, in File create new project → Java EE→ Enterprise Application.
2. In the Enterprise Application goto Files and click the Enterprise Application-ejb folder.
3. Go to Src→Java, Then right click → New → Session Bean.
4. Give the name for SessionBean as Ejbsessionbean and also provide the name for package as pack, select the Remote interface and remove from Local interface.
5. In the created Ejbsessionbean class right click and create the Business method for performing factorial operation.

Ejbsessionbean

```
package pack;
import javax.ejb.Stateless;
@Stateless
public class ejbsessionbeanBean implements ejbsessionbeanRemote
{
        public double Factorial(final double a)
        {
                int fact = 1;
                for(int x = 1;x<=a;x++)
                {
                        fact=fact*x;
                }
                return fact;
        }
}
```

STEPS TO CREATE SERVLET PAGE

1. In the corresponding Files click the Enterprise Application-war folder.
2. Go to Src→Java, Then right click → New → Servlet.
3. Give the name as Ejbservlet and package name as pack.
4. Enter the following coding in doPost method

Ejbservlet

```
package pack;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ejbservlet extends HttpServlet
{
        protected void processRequest(HttpServletRequest request, HttpServletResponse
        response)throws ServletException, IOException
        {
                response.setContentType("text/html;charset=UTF-8");
                PrintWriter out = response.getWriter();
                try
                {
                }
                finally
                {
                        out.close();
                }
        }
        @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse
    response)               throws ServletException, IOException
        {
                ejbsessionbeanBean obj =new ejbsessionbeanBean();
                response.setContentType("text/html;charset=UTF-8");
                PrintWriter out=response.getWriter();
                String aa=request.getParameter("t1");
                int x=Integer.parseInt(aa);
                out.println("The Factorial Value is:"+obj.Factorial(x));
        }
    }
```

STEPS TO DESIGN Index.jsp PAGE

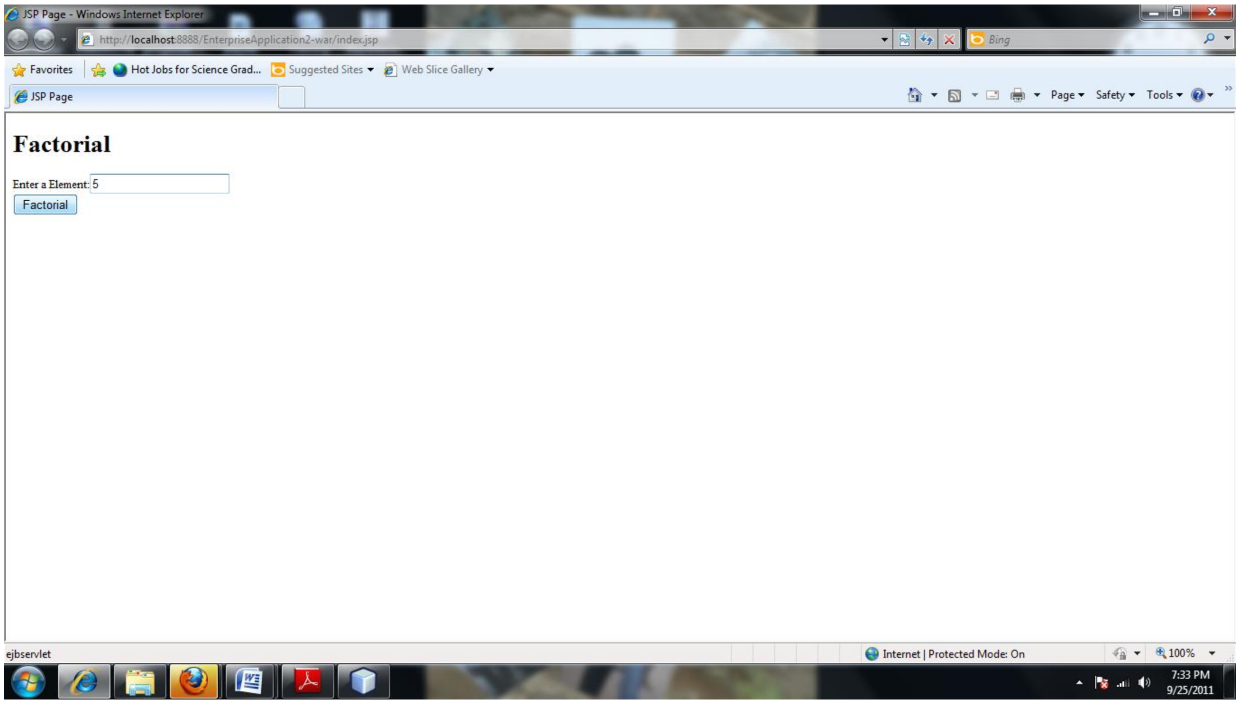
1. Index.jsp will be in Web folder inside the Enterprise Application-war folder.
2. Double click the index.jsp page and write the following code in it.

   Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
            <h1>Hello World!</h1>
            <form name="form1" action="ejbservlet" method="post">
                    Enter a Element:<input type="text" name="t1"/></br>
                    <input type="submit" value="Factorial"/>
            </form>
    </body>
</html>
```

3. Compile the index.jsp page by right click on it. Then run that page.

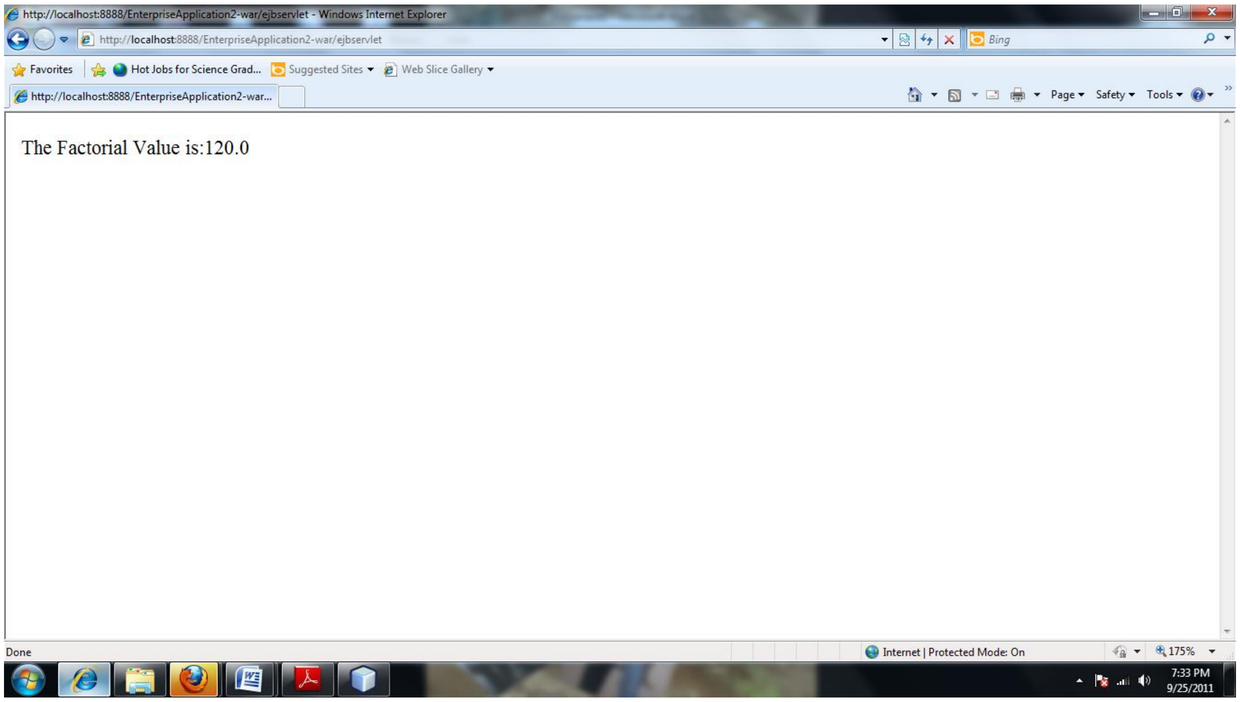
OUTPUT:


JSP Page - Windows Internet Explorer
http://localhost:8888/EnterpriseApplication2-war/index.jsp
Favorites
Hot Jobs for Science Grad...
Suggested Sites
Web Slice Gallery
JSP Page
Page
Safety
Tools
ejbservlet
Internet | Protected Mode: On
100%
7:33 PM
9/25/2011

Factorial

Enter a Element: 5

Factorial


http://localhost:8888/EnterpriseApplication2-war/ejbservlet - Windows Internet Explorer
http://localhost:8888/EnterpriseApplication2-war/ejbservlet
Favorites
Hot Jobs for Science Grad...
Suggested Sites
Web Slice Gallery
http://localhost:8888/EnterpriseApplication2-war...
Page
Safety
Tools
Done
Internet | Protected Mode: On
175%
7:33 PM
9/25/2011

```
The Factorial Value is:120.0
```

14. INVOKING OF EJB COMPONENTS AS WEB SERVICES

AIM:

To create a web service for adding few numbers using NetBeans and write client side code to invoke the web service.

ALGORITHM:

1. Using the Netbeans API create a project of the type web application.
2. Create a web service in the project.
3. Click on the Design tab and design the prototype of the arithmetic web service.
4. Click on source tab and modify the application logic of the web service.
5. Save the project.
6. Right click on the project and click on deploy and undeploy.
7. Then test the web service.
8. Create another web application project and create a jsp file.
9. Right click on project and click on create web service client.
10. Browse and choose the web service created i.e wsdl url in Project tab.
11. Drag and drop the web service reference to the source code window.
12. Then pass the appropriate parameters to the web service client and invoke the web service.

STEPS TO CREATE WEB SERVICE:

1. Create the new project and give Project name->addserver…then click finish
2. The addserver project will be created in right side. Right click it and choose the web service and name it as addweb.
3. After this in left side ,the design window choose the add operation
4. Give the name for operation and declare the parameters.
5. Go to source view and change the code in corresponding methods.

Addweb → WebService

```
package pack;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
@WebService()
public class NewWebService {
    @WebMethod(operationName = "add")
    public int add(@WebParam(name = "a")
    final int a, @WebParam(name = "b")
    final int b) {
        return a+b;
    }
@WebMethod(operationName = "sub")
    public int sub(@WebParam(name = "a")
    final int a, @WebParam(name = "b")
    final int b) {
        return a-b;
    }
}
```

STEPS TO CREATE CLIENT SIDE PROJECT:

1. Create the new project as above and give the name as addclient.
2. In addclient project will be created. right click it and choose the Webservice client.
3. Then browse and choose the addweb wsdl file
4. Then choose the following and add the source code in index.jsp and save it.

Index.jsp source code

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
```

```
<h1>Hello World!</h1>
<form name="" action="actionn.jsp" method="post">
 Enter 1st No:<input  name="fst" type="text" /><br/>
Enter 2nd No:<input  name="snd" type="text" /><br/>
<input  name="ok" type="submit" value="Add" />
</form>
</body>
</html>
```

5. Then create an action.jsp as follows.

Right click web page in addclient and choose new->jsp

Name:action

Click finish

6. click on the actionn.jsp page..then right click in it and choose web service client reference ->call web service

7. The invoke the add service.

8. Add the following code in the **action.jsp:**

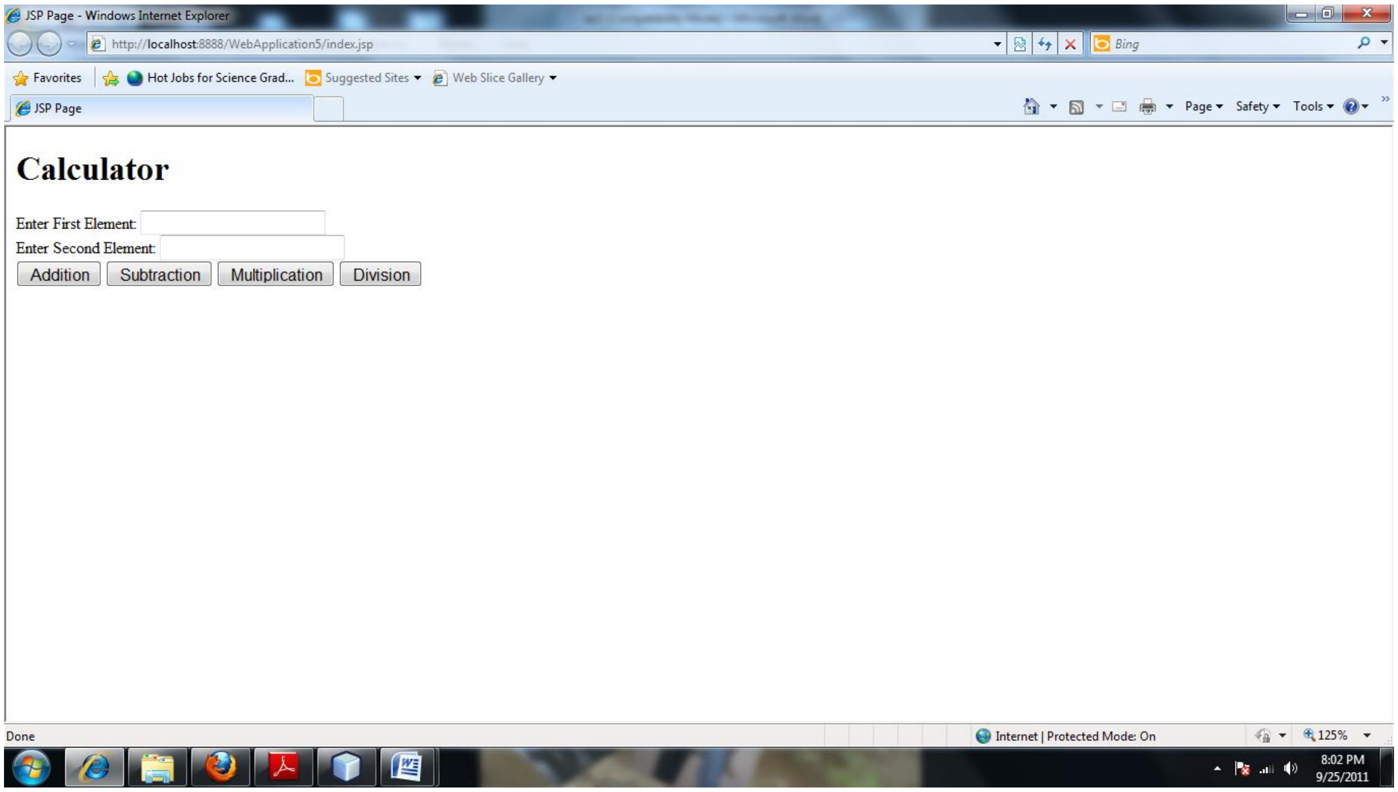
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<h1>Hello World!</h1>
<%
 String a1=request.getParameter("fst");
String b1=request.getParameter("snd");
int aa=Integer.parseInt(a1);
int bb=Integer.parseInt(b1);
  %>
```

```
<%-- start web service invocation --%><hr/>
<%
    try {
            org.AddwebService service = new org.AddwebService();
            org.Addweb port = service.getAddwebPort();
             // TODO initialize WS operation arguments here
            int a = aa;
            int b = bb;
            // TODO process result here
            int result = port.add(a, b);
            out.println("Addition = "+result);
    } catch (Exception ex) {
            // TODO handle custom exceptions here
    }
    %>
<%
    try {
            org.AddwebService service = new org.AddwebService();
            org.Addweb port = service.getAddwebPort();
             // TODO initialize WS operation arguments here
            int a = aa;
            int b = bb;
            // TODO process result here
            int result = port.sub(a, b);
            out.println("Subtraction = "+result);
    } catch (Exception ex) {
            // TODO handle custom exceptions here
    }
    %>

<%-- end web service invocation --%><hr/>
</body>
```

</html>

8. Finally undeploy and deploy the addclient and run it.

OUTPUT:



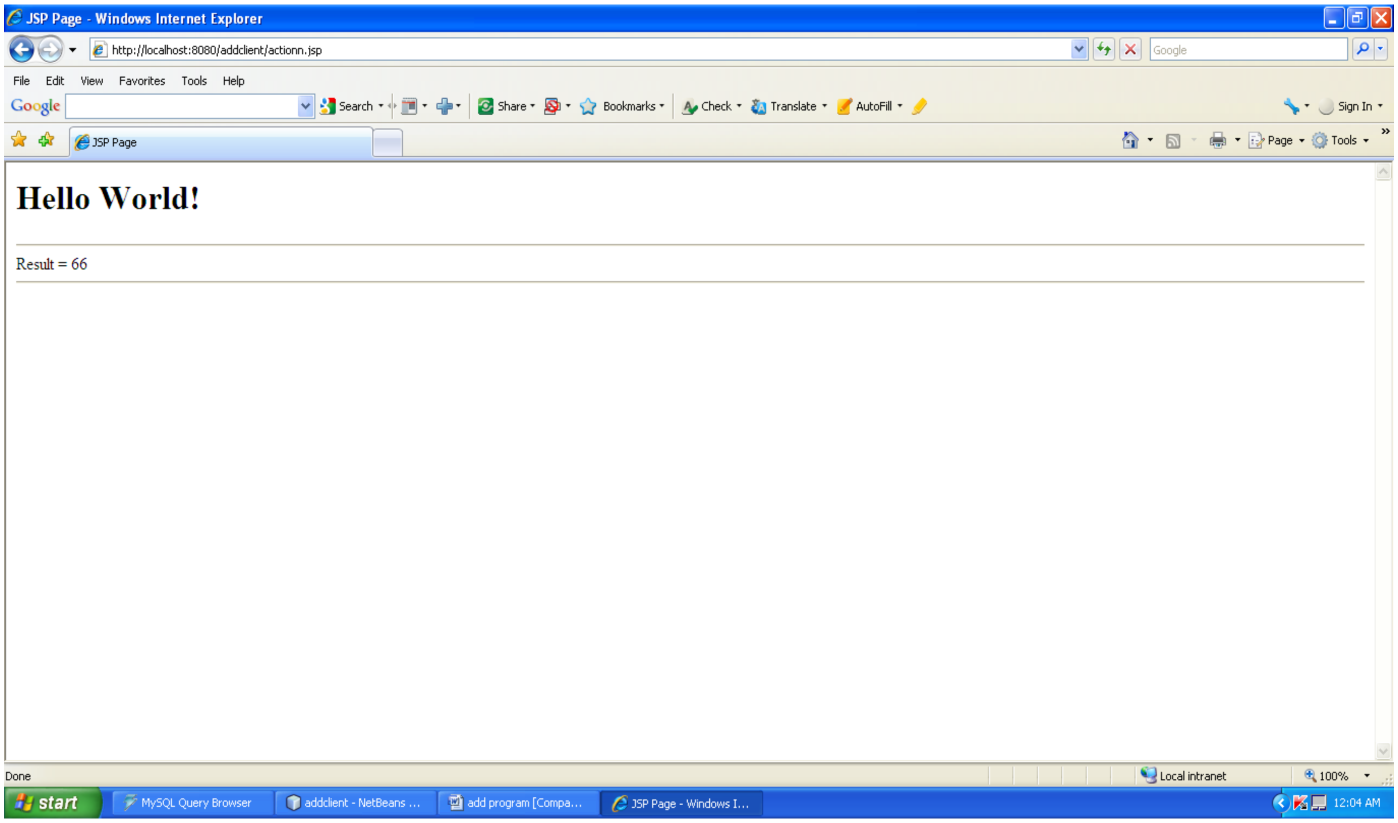
JSP Page - Windows Internet Explorer

http://localhost:8888/WebApplication5/index.jsp

Calculator

Enter First Element:

Enter Second Element:



Hello World!

```
Result = 66
```

15. INVOKING OF .NET COMPONENTS AS WEB SERVICES
(Temperature Conversion)

AIM:

To invoke .NET component as Web Services(Temperature Conversion).

ALGORITHM:

The various steps that are involved in creating a Web Service Component using C# and the .NET Framework are as follows,

1. Create a Visual C# - ASP.NET Web Service project.
2. In the Web Service project create WebMethod for the Temperature conversion.
3. Build and run the Web Service to test it whether the WebMethod are properly given.
4. Then create a new Website with some Control forms.
5. Add the Web Service with the newly created Website by using Add WebReference from the Project menu.
6. Build and run the website.

Steps to Create a Web service:

1. Open Visual Web Developer.
2. On the Filemenu, click New Web Site. The New Web Site dialog box appears.
3. Under Visual Studio installed templates, click ASP.NET Web Service.
4. Type the Web Service name as TemperatureWebService.
5. In the TemperatureWebService create a WebMethod for the Temperature conversion.
6. Type the following code:

TemperatureWebService

```
using System;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;
```

```
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
        public Service ()
        {
        }
        [WebMethod]
        public string HelloWorld()
        {
                return "Hello World";
        }
        [WebMethod]
        public double FahrenheitToCelsius(double Fahrenheit)
        {
                return ((Fahrenheit - 32) * 5) / 9;
        }
        [WebMethod]
        public double CelsiusToFahrenheit(double Celsius)
        {
                return ((Celsius * 9) / 5) + 32;
        }
}
```

7. Build the TemperatureWebService and test it.

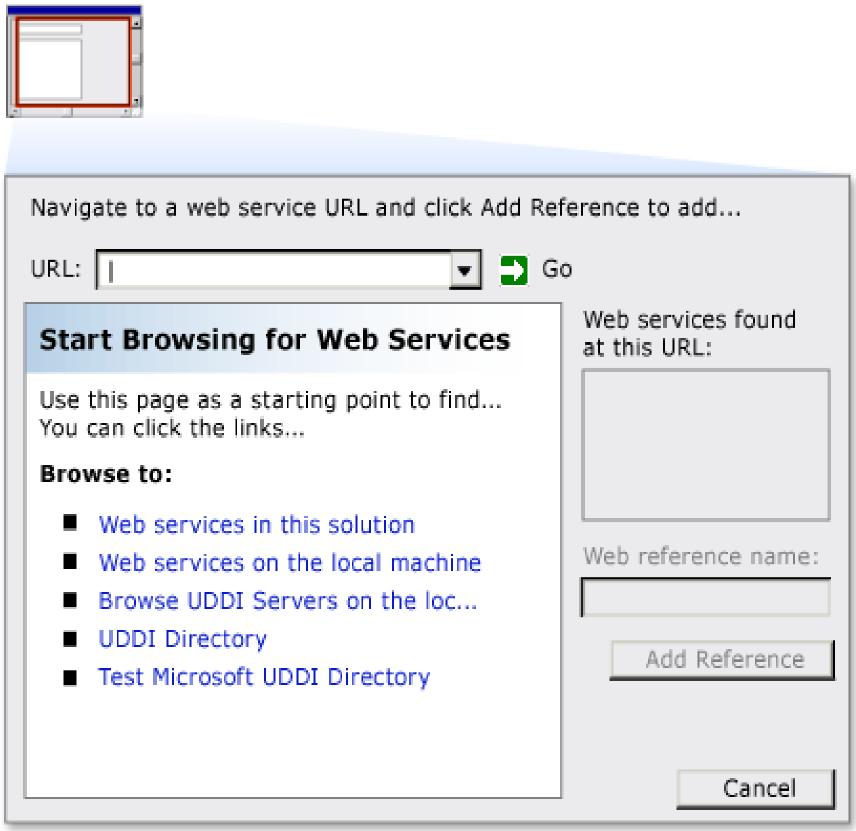
Steps to Create a Web site:

1. Open Visual Web Developer.
2. On the File menu, click New Web Site.
3. The New Web Site dialog box appears.
4. Under Visual Studio installed templates, click ASP.NET Web Site.
5. Click the Design view in the WebSite page and design the WebSite by dragging the Textboxes and Button from the ToolBox.
6. Then refer the TemperatureWebService in the WebSite.

Steps to Create a reference to the Web service:

1. On the Web Site menu, click **Add Web Reference**.
2. The **Add Web Reference**dialog box appears, as shown in the following screen shot.

Add Web Reference dialog box


Navigate to a web service URL and click Add Reference to add...
URL:
Go
Start Browsing for Web Services
Use this page as a starting point to find...
You can click the links...
Browse to:
Web services in this solution
Web services on the local machine
Browse UDDI Servers on the loc...
UDDI Directory
Test Microsoft UDDI Directory
Web services found at this URL:
Web reference name:
Add Reference
Cancel

3. In the **URL** list, enter the following URL for the Web service, and then click **Go**:

   http://localhost/TemperatureWebService/Convert.asmx

   When Visual Web Developer finds the Web service, information about the Web service appears in the **Add Web References** dialog box.
4. Click one of the method links. The test page for the method appears.
5. If you want to change the webservice name means change in text box or give default name as localhost. Then Click **Add Reference**.

   Visual Web Developer creates an App_WebReferences folder and adds a folder to it for the new Web reference. By default, Web references are assigned a namespace corresponding to their server name (in this case, localhost).

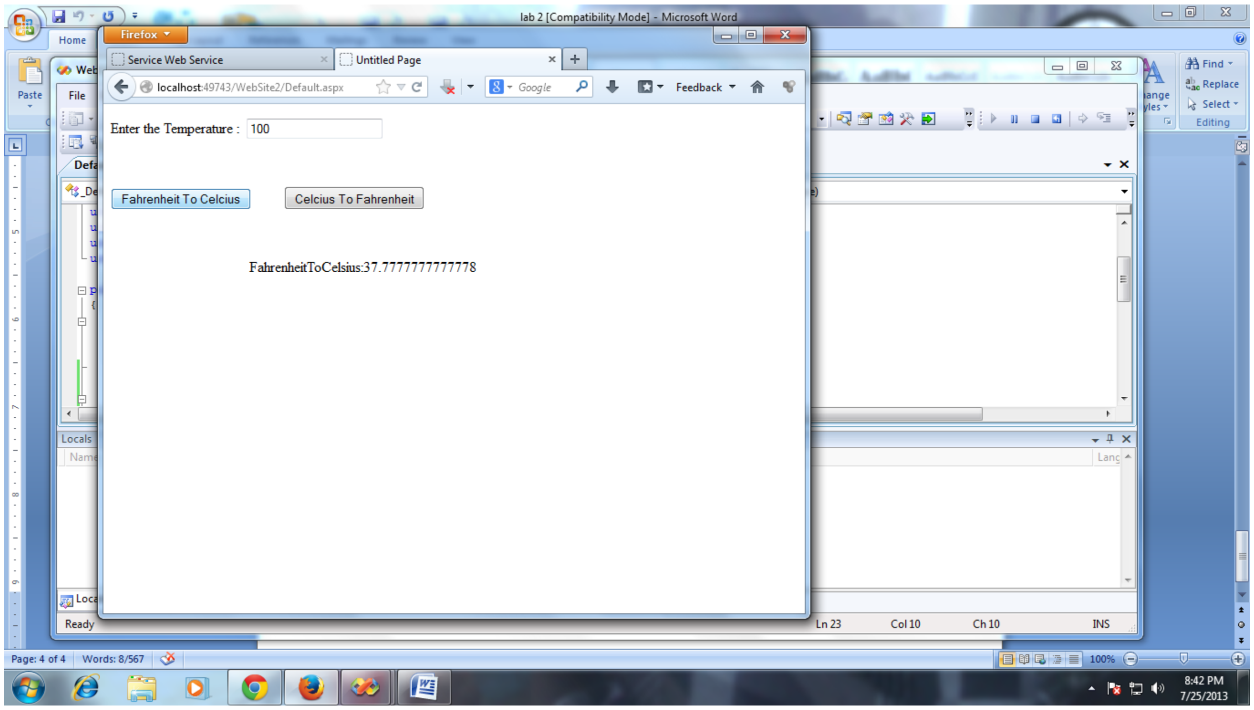
7. Type the following code in the source view of the WebSite in concern Button event.

TemperatureWebSite

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
public partial class _Default : System.Web.UI.Page
{
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        localhost.Service ob = new localhost.Service();
        protected void Button1_Click(object sender, EventArgs e)
        {
                double temp = System.Convert.ToDouble(TextBox1.Text);
                Label2.Text = "FahrenheitToCelsius:"
                                +ob.FahrenheitToCelsius(temp).ToString();
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
                double temp = System.Convert.ToDouble(TextBox1.Text);
                Label2.Text = "CelsiusToFahrenheit:"
                                +ob.CelsiusToFahrenheit(temp).ToString();
        }
}
```

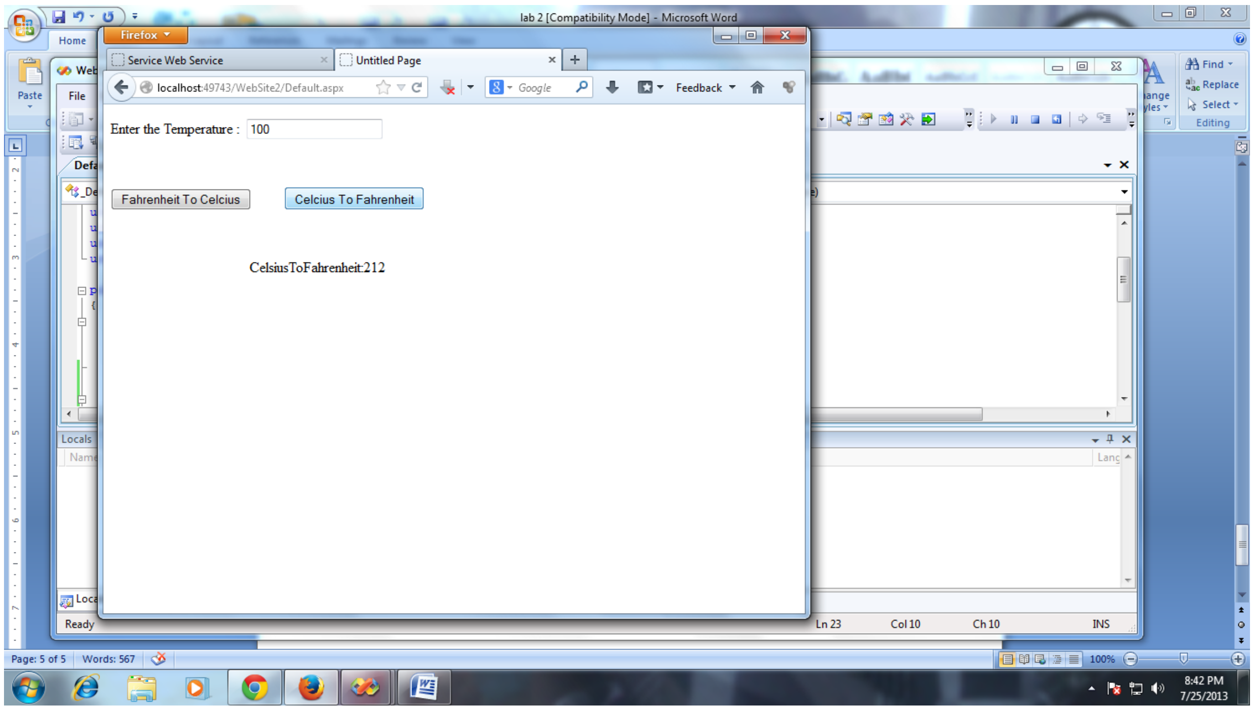
8. Build and run the WebSite.

OUTPUT:


Enter the Temperature : 100

Fahrenheit To Celcius   Celcius To Fahrenheit

FahrenheitToCelsius:37.7777777777778


Enter the Temperature : 100

Fahrenheit To Celcius   Celcius To Fahrenheit

CelsiusToFahrenheit:212

16. DEVELOP A J2EE CLIENT TO ACCESS .NET WEB SERVICES

AIM:

To develop a J2EE Client to access a .Net Web Services.

ALGORITHM:

1. Create a Visual C# - ASP.NET Web Service project.
2. In the Web Service project create WebMethod for the Temperature conversion.
3. Build and run the Web Service to test it whether the WebMethod are properly given.
4. Then create a new Website in J2EE i.e., in Netbeans.
5. Add the Web Service with the newly created Website by using Add WebReference from the Project menu.
6. Build and run the website.

Steps to Create a Web service:

7. Open Visual Web Developer.
8. On the Filemenu, click New Web Site. The New Web Site dialog box appears.
9. Under Visual Studio installed templates, click ASP.NET Web Service.
10. Type the Web Service name as TemperatureWebService.
11. In the TemperatureWebService create a WebMethod for the Temperature conversion.
12. Type the following code:

```
using System;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Linq;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment
the following line.
// [System.Web.Script.Services.ScriptService]
```

```
public class Service : System.Web.Services.WebService
{
    public Service () {

        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public double FahrenheitToCelsius(double Fahrenheit)
    {
        return ((Fahrenheit - 32) * 5) / 9;
    }
    [WebMethod]
    public double CelsiusToFahrenheit(double Celsius)
    {
        return ((Celsius * 9) / 5) + 32;
    }

    }
```

13. Build the TemperatureWebService and test it.

Steps to Create a Web site:

1. Create the new project in Netbeans and give the name as addclient.
2. In addclient project will be created. right click it and choose the Web service client.
3. Copy the wsdl url created by the webservic using ASP .NET .
3. Then paste the wsdl url inside the WSDL Url textbox and then click finish.
4. Then in the index.jsp page write the following code:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<form name="form1" action="act.jsp" method="post">
        Enter the Temperature:<input type="text" name="t1"/></br>
<input type="submit" value="Temperature Conversion"/>
</form>
</body>
</html>
```

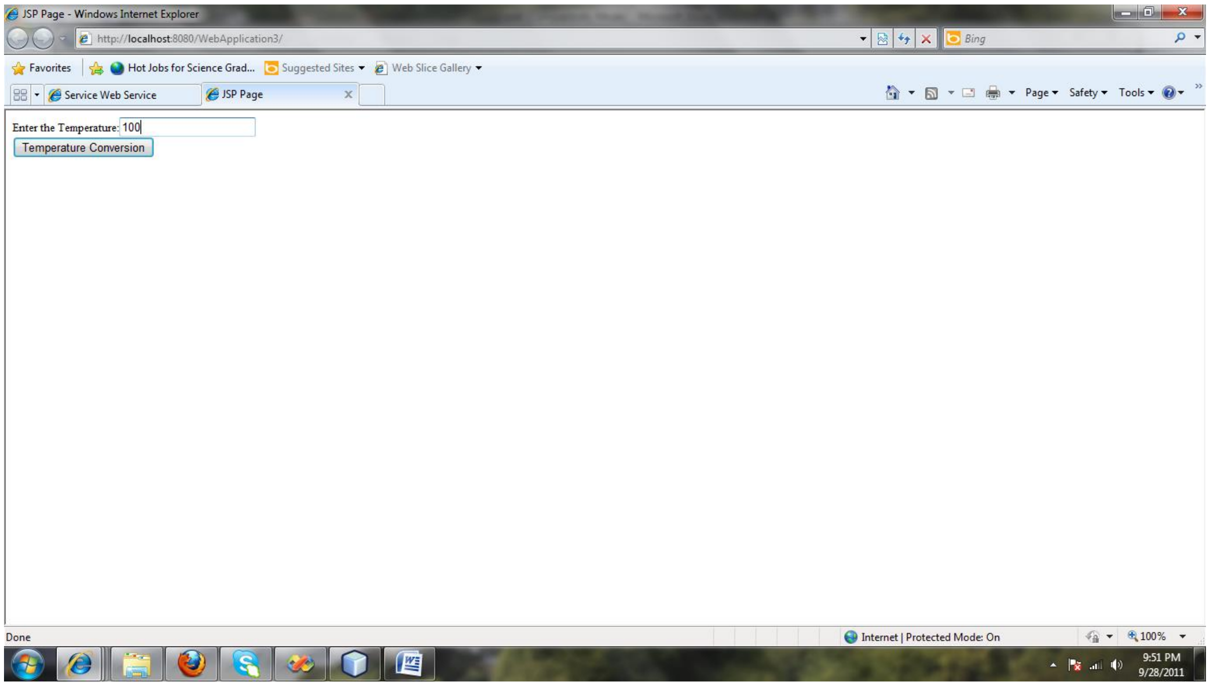
5. Then create another jsp page(act.jsp) and right click in it and click WebService client resources→ cal webservice operations.

6. The following code will generate and alter some codings in it.

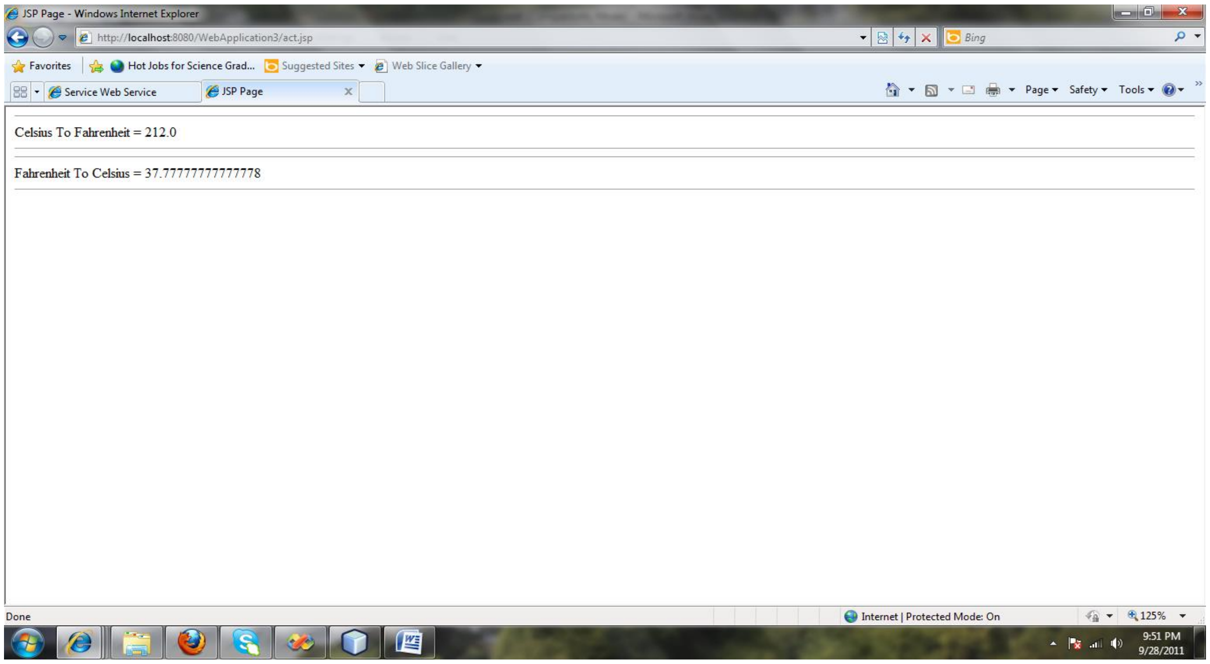
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<%
      String aa=request.getParameter("t1");
      double a1=Double.parseDouble(aa);
      %>
<%-- start web service invocation --%><hr/>
<%
   try {
        org.tempuri.Service service = new org.tempuri.Service();
        org.tempuri.ServiceSoap port = service.getServiceSoap();
        double celsius = a1;
```

```
            double result = port.celsiusToFahrenheit(celsius);
            out.println("Celsius To Fahrenheit = "+result);
    } catch (Exception ex) {
            // TODO handle custom exceptions here
    }
    %>
<%-- end web service invocation --%><hr/>
<%-- start web service invocation --%><hr/>
<%
    try {
            org.tempuri.Service service = new org.tempuri.Service();
            org.tempuri.ServiceSoap port = service.getServiceSoap();
            double fahrenheit = a1;
            double result = port.fahrenheitToCelsius(fahrenheit);
            out.println("Fahrenheit To Celsius = "+result);
    } catch (Exception ex) {
    }
    %>
<%-- end web service invocation --%><hr/>
</body>
</html>
```

OUTPUT:


JSP Page - Windows Internet Explorer
http://localhost:8080/WebApplication3/
Temperature Conversion

Enter the Temperature: 100


JSP Page - Windows Internet Explorer
http://localhost:8080/WebApplication3/act.jsp

```
Celsius To Fahrenheit = 212.0
Fahrenheit To Celsius = 37.77777777777778
```

17. DEVELOP A .NET CLIENT TO ACCESS J2EE WEB SERVICES

AIM:

To develop a .NET client to access J2EE web services

ALGORITHM:

1. Create a Netbean Web Service project.
2. In the Web Service project create WebMethod for the Addition, Subtraction,etc.
3. Build and run the Web Service to test it whether the WebMethod are properly given.
4. Then create a new Website in .Net Application.
5. Add the Web Service with the newly created Website by using Add WebReference from the Project menu.
6. Build and run the website.

STEPS TO CREATE A WEB SERVICE:

7. Create the new project and give Project name->addserver…then click finish
8. The addserver project will be created in right side. Right click it and choose the web service and name it as addweb.
9. After this in left side ,the design window choose the add operation
10. Give the name for operation and declare the parameters.
11. Go to source view and change the code in corresponding methods.

```
package pack;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
@WebService()
public class NewWebService {
   @WebMethod(operationName = "add")
   public int add(@WebParam(name = "a")
   final int a, @WebParam(name = "b")
```

```
    final int b) {
        return a+b;
    }
@WebMethod(operationName = "sub")
    public int sub(@WebParam(name = "a")
    final int a, @WebParam(name = "b")
    final int b) {
        return a-b;
    }
}
```

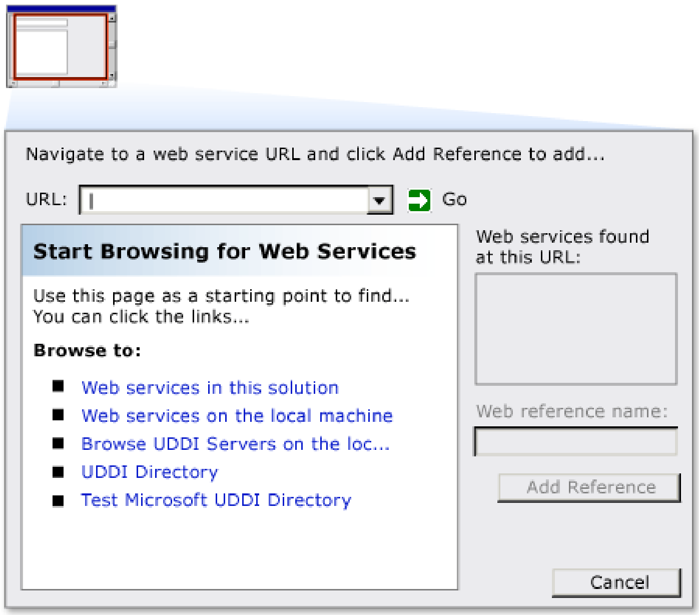
STEPS TO CREATE A WEBSITE:

12. Open Visual Web Developer.
13. On the File menu, click New Web Site.
14. The New Web Site dialog box appears.
15. Under Visual Studio installed templates, click ASP.NET Web Site.
16. Click the Design view in the WebSite page and design the WebSite by dragging the Textboxes and Button from the ToolBox.
17. Then refer the WebService addweb created in Netbean in the WebSite.

    Steps to Create a reference to the Web service:

    1. On the Web Site menu, click **Add Web Reference**.
    2. The **Add Web Reference**dialog box appears, as shown in the following screen shot.

Add Web Reference dialog box


Navigate to a web service URL and click Add Reference to add...
URL:
Go
Start Browsing for Web Services
Use this page as a starting point to find...
You can click the links...
Browse to:
Web services in this solution
Web services on the local machine
Browse UDDI Servers on the loc...
UDDI Directory
Test Microsoft UDDI Directory
Web services found at this URL:
Web reference name:
Add Reference
Cancel

3. In the **URL** list, enter the following URL for the Web service, and then click **Go**:

   http://localhost:8888/WebApplication4/NewWebServiceService?Tester

   When Visual Web Developer finds the Web service, information about the Web service appears in the **Add Web References** dialog box.

4. Click one of the method links. The test page for the method appears.
5. Click **Add Reference**.

Visual Web Developer creates an App_WebReferences folder and adds a folder to it for the new Web reference. By default, Web references are assigned a namespace corresponding to their server name (in this case, localhost).

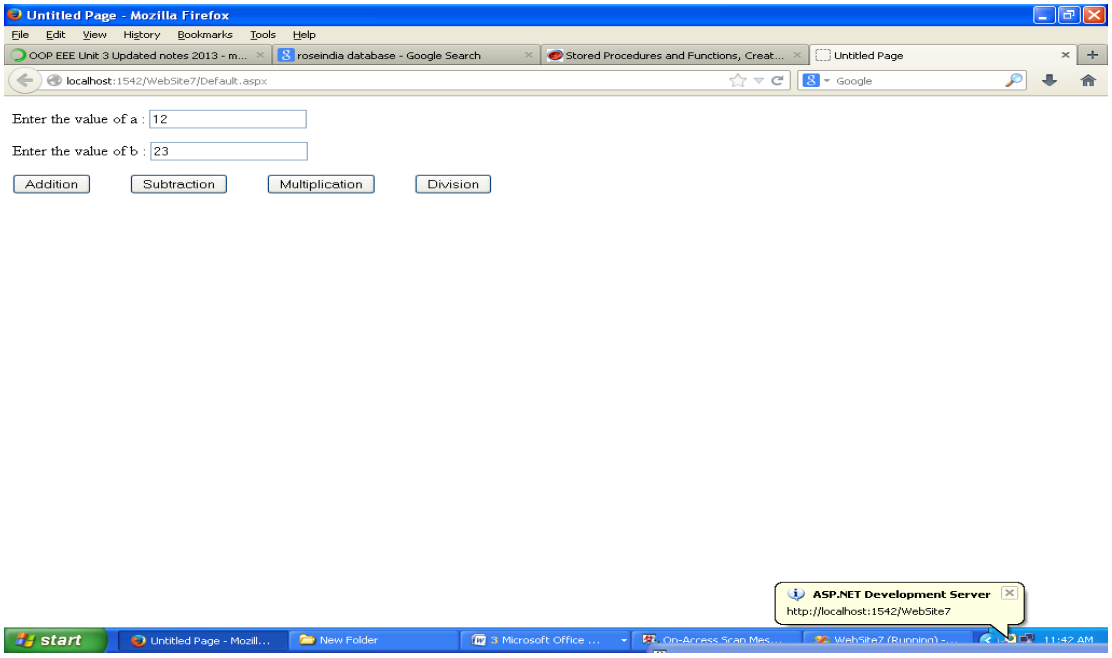
18. Type the following code in the source view of the WebSite in concern Button event.

```
using System;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
```

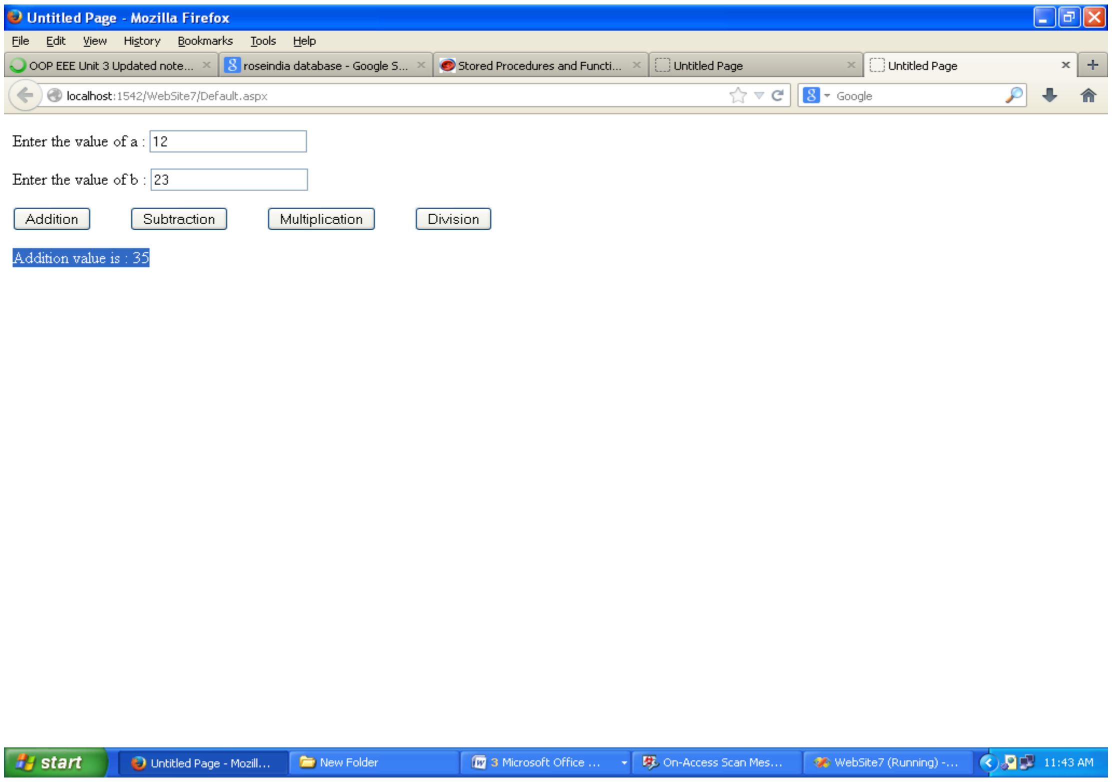
```
public partial class _Default : System.Web.UI.Page

{
    localhost.NewWebServiceService obj = new localhost.NewWebServiceService();
    protected void Button1_Click(object sender, EventArgs e)
    {
        int a = System.Convert.ToInt16(TextBox2.Text);
        int b = System.Convert.ToInt16(TextBox3.Text);
        Label1.Text = "Addition value is:" + obj.add(a, b).ToString();
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        int a = System.Convert.ToInt16(TextBox2.Text);
        int b = System.Convert.ToInt16(TextBox3.Text);
        Label1.Text = "Subtraction value is:" + obj.sub(a,b).ToString();
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        int a = System.Convert.ToInt16(TextBox2.Text);
        int b = System.Convert.ToInt16(TextBox3.Text);
        Label1.Text = "Division value is:" + obj.div(a, b).ToString();
    }
    protected void Button4_Click(object sender, EventArgs e)
    {
        int a = System.Convert.ToInt16(TextBox2.Text);
        int b = System.Convert.ToInt16(TextBox3.Text);
 Label1.Text = "Multiplication value is:" + obj.mul(a,b).ToString();
    }

}
```

19. Build and run the website.

OUTPUT:


Enter the value of a : 12

Enter the value of b : 23


Enter the value of a : 12

Enter the value of b : 23

Addition value is : 35