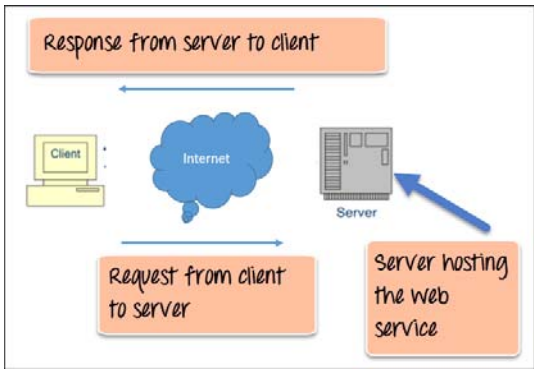
Web services

Overview

- Is available over the Internet or private (intranet) networks
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction.
  - These systems can include programs, objects, messages, or documents.
- Web services are not tied to any one operating system or programming language:
  - Java can talk with Perl; Windows applications can talk with Unix applications.
- Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.

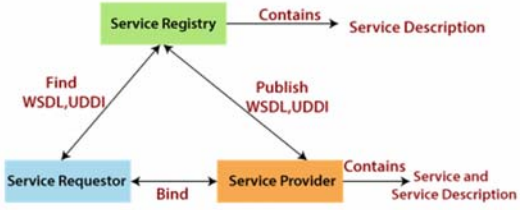
Overview

- Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Uses a standardized XML messaging system:
  - XML is used to encode all communications to a web service.
  - For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response.
  - As all communication is in XML,



Web Service Roles

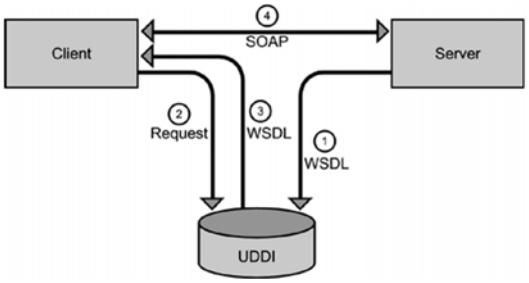
- There are three major roles within the web service architecture −
  - **Service Provider**
    - This is the provider of the web service. The service provider implements the service and makes it available on the Internet.
  - **Service Requestor**
    - This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.
  - **Service Registry**
    - This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.



Web Service Roles, Operations and Artifacts

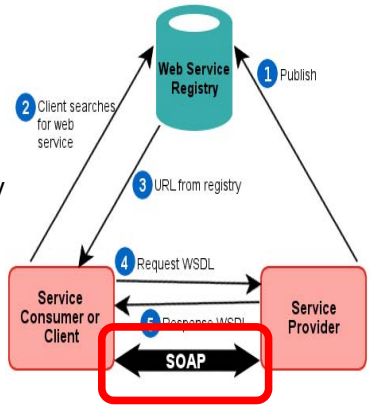
Components of Web Services

- The basic web services platform is XML + HTTP. All the standard web services work using the following components:
  - SOAP (Simple Object Access Protocol):
  - UDDI (Universal Description, Discovery and Integration)
  - WSDL (Web Services Description Language)



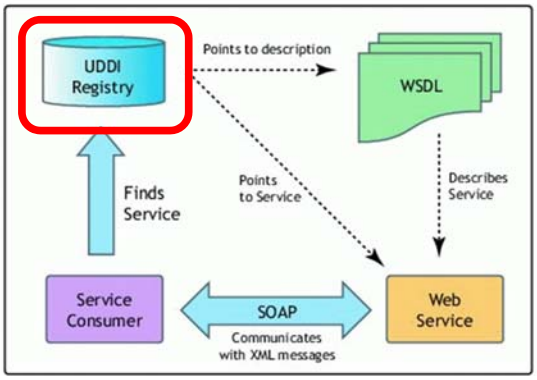
Components of Web Services

- SOAP (Simple Object Access Protocol):
  - Is Communication protocol, for communication between applications.
    - is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks.
  - Is an XML-based protocol for exchanging information:
    - It uses XML Information Set for its message format, and relies on application layer protocols
  - Is designed to communicate via Internet:
    - most often Hypertext Transfer Protocol (HTTP), although some legacy systems communicate over Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.



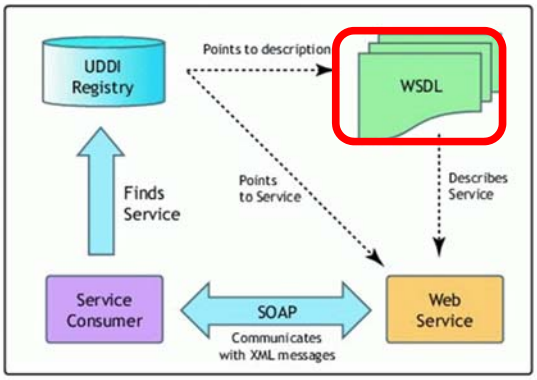
Components of Web Services

- UDDI (Universal Description, Discovery and Integration)
  - Is an XML-based registry for businesses worldwide to list themselves on the Internet.
  - Its ultimate goal is to streamline online transactions by enabling companies to find one another on the Web and make their systems interoperable for e-commerce.
  - Stands for Universal Description, Discovery, and Integration.
  - Is platform independent, open framework.
  - Can communicate via SOAP, CORBA, and Java RMI Protocol.
  - UDDI uses WSDL to describe interfaces to web services, it is seen with SOAP and WSDL as one of the three foundation standards of web services.



Components of Web Services

- WSDL (Web Services Description Language)
  - WSDL is an XML-based protocol for information exchange in decentralized and distributed environments.
  - WSDL definitions describe how to access a web service and what operations it will perform.
  - WSDL is a language for describing how to interface with XML-based services.

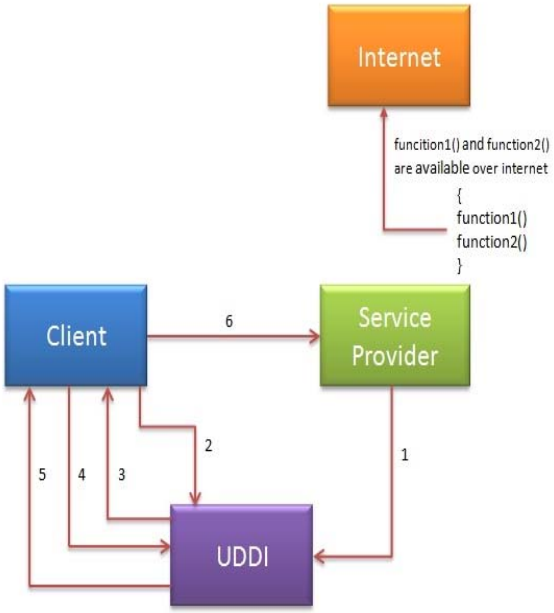


How Does a Web Service Work?

- A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of :
  - XML to tag the data
  - SOAP to transfer a message
  - WSDL to describe the availability of service.
- You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.
- You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

How to access web service

1. Service provider registers with UDDI.
2. Client searches for service in UDDI.
3. UDDI returns all service providers offering that service.
4. Client chooses service provider
5. UDDI returns WSDL of chosen service provider.
6. Using WSDL of service provider,client accesses web service.



Example of the Web Service

- **Web Portal:** Web portal is used to fetch the headline news from the linked web service.
- **Weather Reporting:** For the reporting of weather, we will use Weather Reporting Web Service for displaying the information about the weather on our website.
- **Stock Quote:** The latest information about the share market with the Stock Quote can display on our website.
- **News headline:** By using the news headline Web Service, we can show the latest update of the news on our website

Web Service Cases

- **Amazon Web Services**
  - Amazon, one of the most popular commercial online businesses, offers a Web-service interface that provides a number of interesting features. The possibilities range from simple queries out of the Amazon catalogs to full-fledged e-commerce Web sites that operate in partnership with Amazon through the company's Amazon affiliates program.
-
- **The GoogleSearch API**
  - Google also provides a SOAP-based API for accessing its resources in a model for Web services. The Google API can be used to programmatically access several different services, including executing a search on Google and receiving the results, requesting a spelling suggestion, and fetching a cached page.

Web services advantages

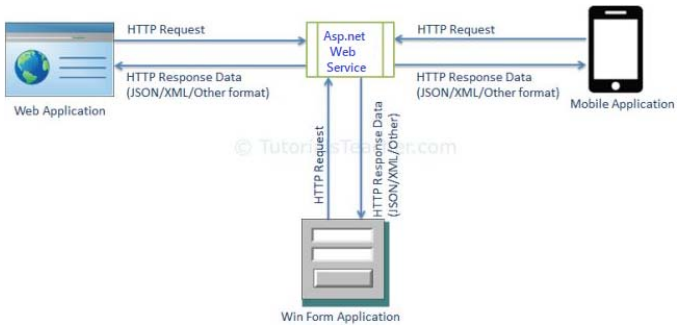
- Use open, text-based standards, which enable components written in various languages and for different platforms to communicate.
- Promote a modular approach to programming, so multiple organizations can communicate with the same Web service.
- Comparatively easy and inexpensive to implement, because they employ an existing infrastructure and because most applications can be repackaged as Web services.
- Significantly reduce the costs of enterprise application (EAI) integration and B2B communications.

Web Services Limitations

- Too slow for use in high-performance situations.
- Increase traffic on networks.
- The lack of security standards for Web services.
- The standards that drive Web services are still in draft form (always will be in refinement).

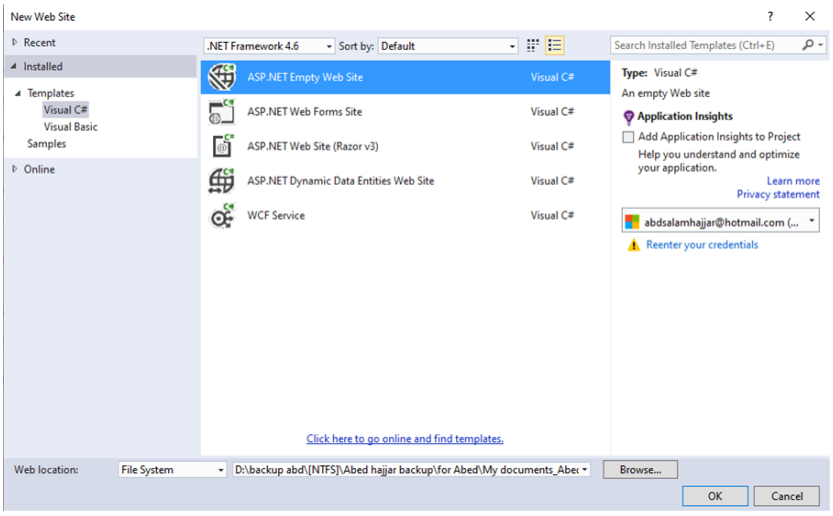
ASP.NET Web Service

- We can now use ASP.NET to create Web Services based on industrial standards including XML, SOAP, and WSDL.
  - that can be accessed in different applications on different platforms such as web, windows, mobile etc.
- ASP.net web service is built on top of ASP.NET and supports ASP.NET request/response pipeline.
  - can be hosted in IIS



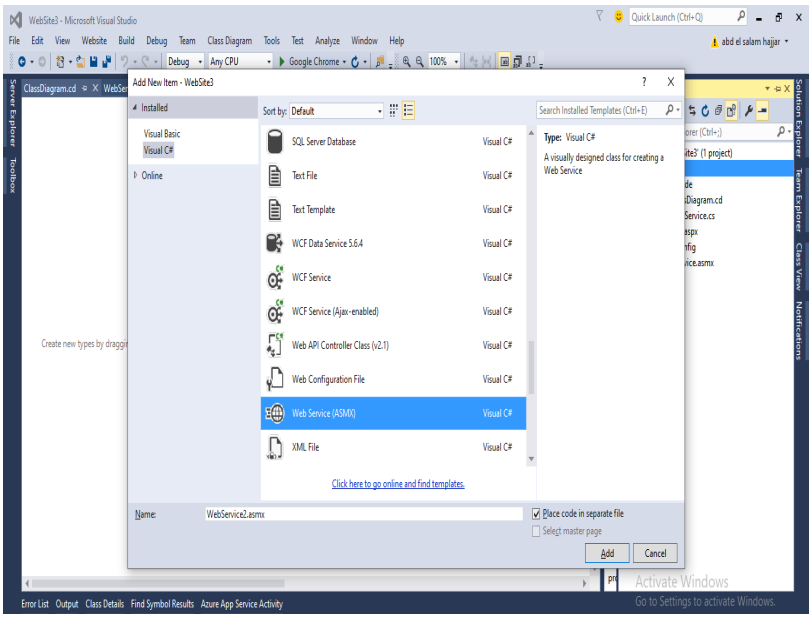
ASP.NET Web Service

- **Create the ASP.NET Web Service Source File**
  - Open Visual Studio and create a new web site:
    - Select .Net Framework 4.6 ->Select “ASP.NET Empty Web Site” -> Then, you have to give the name of your web site "**mySite**". Then Click the ok Button.



ASP.NET Web Service

- **Right-click on the project name: “Add”→”Add New Item”, and select “Web Service(ASMX),** Then, you have to give the name of your web service "**service1**". Then Click the Add Button.



ASP.NET Web Service

- you will see the following window:

- Note that there is predefined method "**HelloWorld**" which returns the string "**Hello World**".
- You can use your own method and can perform various operations. Here I made a simple method which returns the “**multiplication**” & “**sum**” of two numbers using the code.

```
App_Code/Service.cs
Service
Service()
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJ
// [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 string HelloWorld() {
        return "Hello World";
    }

[WebMethod]
public int Multiplication(int a, int b)
{
    return (a * b);
}
[WebMethod]
public int Sum(int a, int b)
{
    return (a + b);
}
```

ASP.NET Web Service

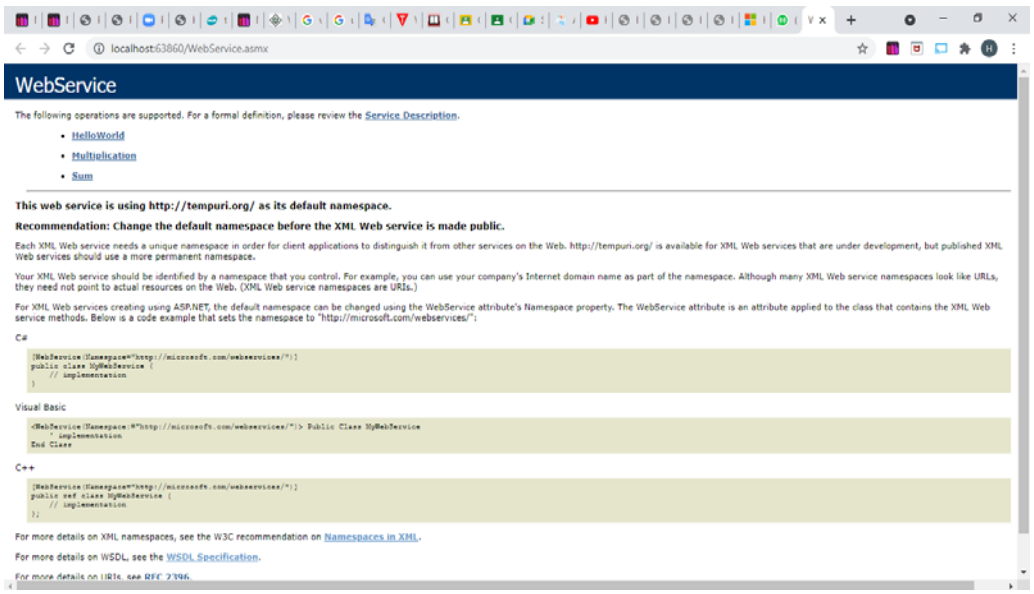
- Service.asmx- which contains the following code:

WebService.asmx | WebService.cs

```
<%@ WebService Language="C#" CodeBehind="~/App_Code/WebService.cs" Class="WebService" %>
```

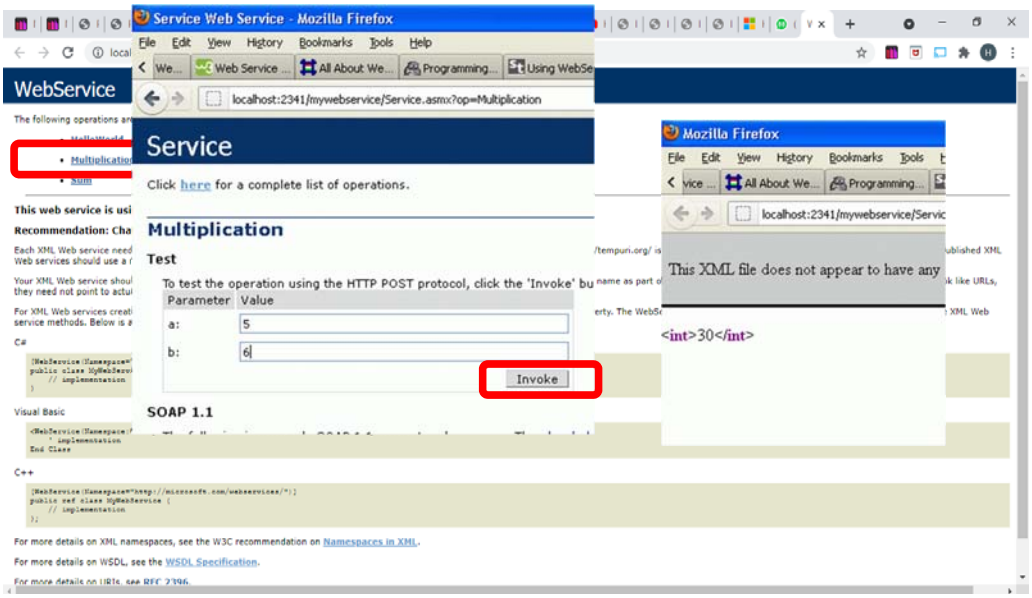
ASP.NET Web Service

- Build Web Service and Run the Web Service for testing by pressing F5 function key.



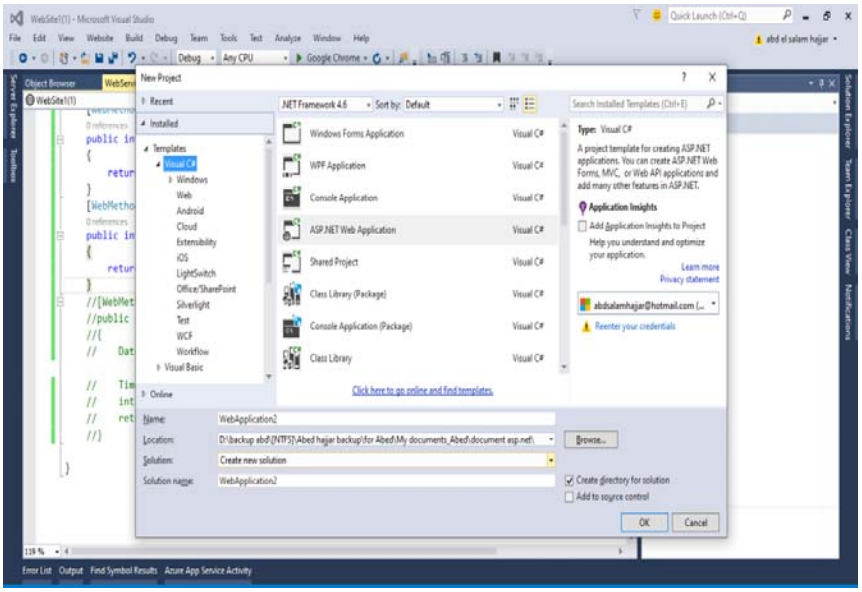
ASP.NET Web Service

- Click on the Multiplication button to test the web service.



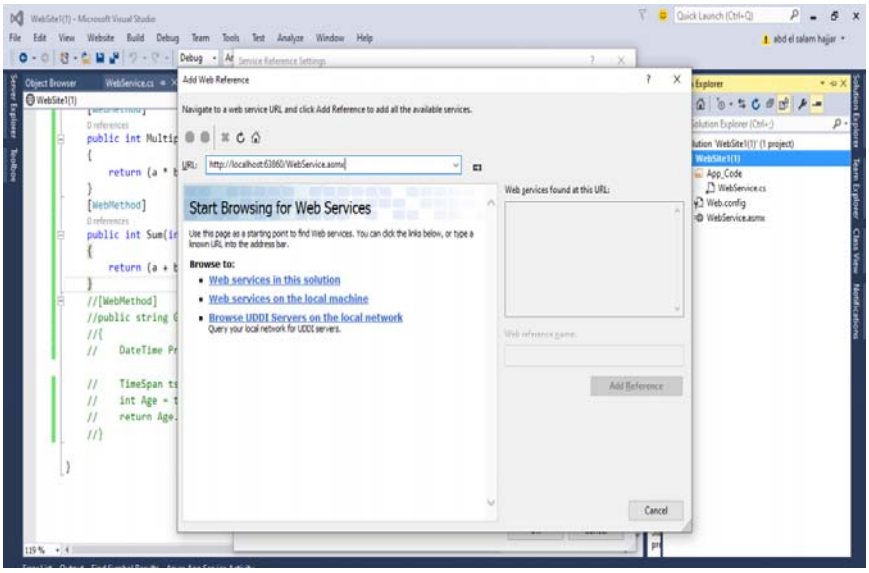
ASP.NET Web Service

- **Testing Web Service**
  - Create a new project by File > New > Project :



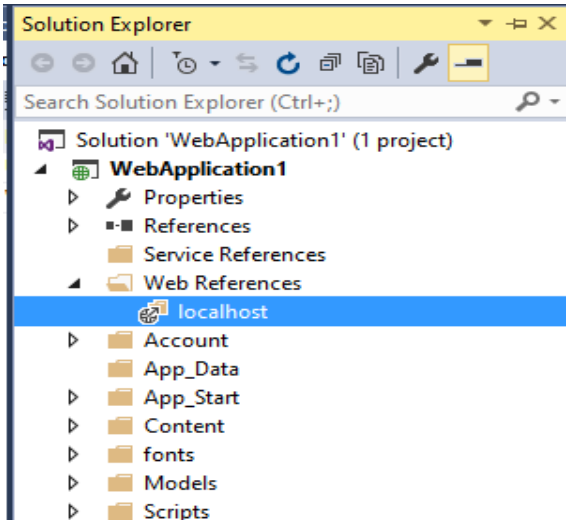
ASP.NET Web Service

- **Testing Web Service**
  - Right click on the project name: **Add→ Service Reference→ Advanced → Add Web Reference**
  - Paste the URL of the created web service and click on **'Go'** button and then **'Add Reference'**



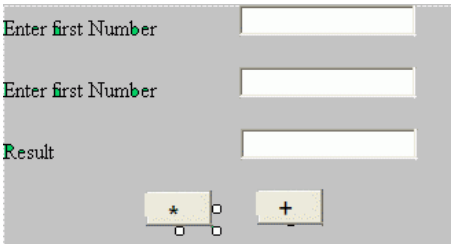
ASP.NET Web Service

- **Testing Web Service**
  - Now your web service is ready to use in the Solution Explorer you will see a “localhost” folder in the Web References” which contain the added web service.



ASP.NET Web Service

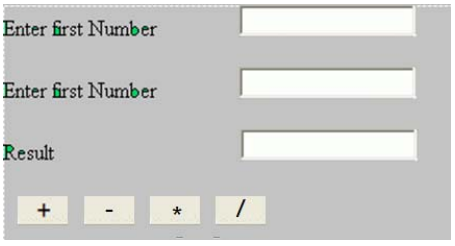
- **Testing Web Service**
  - Add new Form named “Form1.aspx” with the following design




```
1. protected void BtnMultiplication_Click(object sender, EventArgs e)
2. {
3.     localhost.Service mys = new localhost.Service();
4.     int a = Convert.ToInt32(TextBox1.Text);
5.     int b = Convert.ToInt32(TextBox2.Text);
6.     int c = mys.Multiplication(a, b);
7.     TextBox3.Text = c.ToString();
8. }
```

ASP.NET Web Service

- **Ex1:**
  - To create an online calculator:
    - Create 4 web services for the 4 main operations(+,-,*,/)
    - Create the following form.aspx in a new project and call the created calculation services



ASP.NET Web Service

- **Ex1(solution):**
  - in web service :

```
[WebMethod]
    public int Multiplication(int a, int b)
    {
        return (a * b);
    }
    [WebMethod]
    public int Sum(int a, int b)
    {
        return (a + b);
    }
    [WebMethod]
    public int Division(int a,int b)
    { return a / b;
    }
    [WebMethod ]
    public int substraction(int a, int b)
    {
        return a - b;
    }
```

ASP.NET Web Service

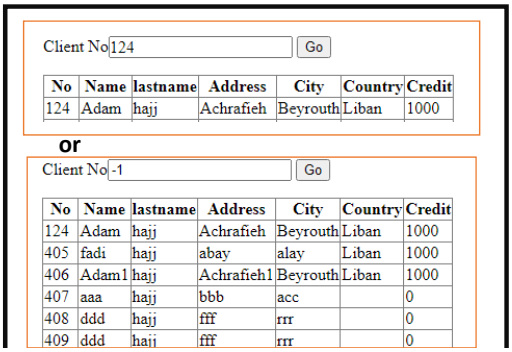
- **Ex1(solution):**
  - in the web form :

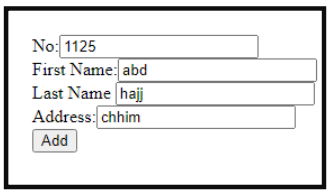
```
protected void btnMul_Click(object sender, EventArgs e)
        {

            localhost.WebService ws = new localhost.WebService();
            int a = Convert.ToInt32(TextBox1.Text);
            int b = Convert.ToInt32(TextBox2.Text);
            int c = ws.Multiplication(a, b);
            TextBox3.Text = c.ToString();
        }
```

Same for others buttons……

- **Ex2:**
  - we have the database name “DB1.mdb” contain the following table:
    - Client (No, Fname, Lname, Address)
  - Create the following 2 web services :
    - ***SelectClient***: allow to return a datatable containing client information, according to the client no that is passed to this web service as parameter,
      - N.B: if the parameter is -1 this service return all clients
    - ***InsertClient***: allow to add a new client, It take client no, first name, last name, and address as parameters.
  - In a new web project create 2 webforms allow to access the created web services as follow:

WebForm1





WebForm2

- **Ex2 (sol):**

Web Services

```
public DataTable   selectClient(int no)
    {
        OleDbConnection cn = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\\dbProducts.mdb");

        if (cn.State==ConnectionState.Closed)
        { cn.Open(); }
        OleDbDataAdapter adp;
        if (no==-1)
            adp =     new OleDbDataAdapter("select * from clients", cn);
          else
        adp= new OleDbDataAdapter("select * from clients where [No]=" + no, cn);
        DataTable dt = new DataTable("c");
        adp.Fill(dt);
        return dt;
    }

    [WebMethod]
    public void insertClient(int no, string fname, string lname, string address)
    {
        OleDbConnection cn = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\\dbProducts.mdb");

        if (cn.State == ConnectionState.Closed)
        { cn.Open(); }
        OleDbCommand cmd = new OleDbCommand("insert into clients([No],[Name],lastname, Address )
values("+no+",'"+fname +"','"+lname+"','"+address +"')", cn);
        cmd.ExecuteNonQuery();
    }
```

- **Ex2 (sol):**

Web form

```
 protected void Button2_Click(object sender, EventArgs e)
        {
            localhost.WebService ws = new localhost.WebService();
            DataTable dt = ws.selectClient(System.Convert.ToInt16(TextBox4.Text));
            GridView1.DataSource = dt;
            GridView1.DataBind();


        }

        protected void btnadd_Click(object sender, EventArgs e)
        {
            localhost.WebService ws = new localhost.WebService();
            ws.insertClient(System.Convert.ToInt16(txtno.Text ), txtfname.Text , txtlname.Text , txtaddress.Text
);

        }
```