WEB SERVICES & WCF

Class notes

By

Mr . **Narasimha** sir

WebServices

8/11/15

→ What is webservices?

→ webservice is a reusable component (class) across internet.
→ webservices are provides services to other Applications.
→ webservices applications are example of "Services Oriented Application" (SOA).

→ What is SOA ?

→ SOA stands for Services oriented Applications.
→ All SOA based applications are ~~eligible~~ targetting other applications developers.
→ SOA provides reusable functionality for other Applications.
→ Regular applications (console, window, web) are targets the end-user.

→ What are the examples SOA based application technologies?

→ SOA is supported by different technogies.

Examples:-

1) Remoting
2) RMI
3) Dcom
4) Web services
5) WCF
6) Web API

Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

→ What are the technogies can be develop the webservice?

→ web services can be developed by using any of the following technologies.

1) ASP.NET (.Net webservices)
2) J2EE (Java webservices)
3) Oracle (PL/SQL webservices)

→ webservices applications are also called "service providers"

→ what are the technogies can consume the web service?

→ webservice can be consumed by most of technogies in the industry

→ Simply they need internet connection with corresponding programming environment

Examples:-

→ Any .Net Application
→ Java
→ php
→ Oracle
→ IBM
→ SAP
→ Java Script (jQuery / AJS)

Note:- Application that will consume the webservices is called "Service Consumer".

Webservice Related terminology:

→ web service is an example SOA based application which provide service to other applications.

→ In SOA based applications, we are using the following technical terminology to address the programming items

1. Service Provider:

→ Application that will provide the service class and its operations is called "Service Provider".

→ Service provider should deploy on the server so, that we can get URL to access by consumer.

→ Service Provider is responsible for organising the service and required operations.

2) Service class :-

→ Every service is implemented as a class.

→ In .Net/Java, we are creating service by preparing corresponding class.

→ A class that provides required method for other application is called "service class".

3) Service Operations :-

→ Methods that we are creating in service class which can be accessible by consumer is called "service ~~Provider~~ operations".

→ In ASP.Net web service, all public methods does not available for consumer.

→ we need mark the required methods as "service methods" by using corresponding rules.

4) Service Consumer :-

→ The Application that will use the service is called "service consumer".

→ Service consumer will get the service details with help of service URL.

Ex:- http://localhost:1234/WebService1.asmx

→ consumer application may be any application / any technology.

5) Service Proxy :-

→ Service Proxy is a special class that will prepare in service consumer. Application.

→ This class will simulate the service class with dummy methods.

→ Proxy class contains the methods with redirected to logic to service operations.

→ Proxy class methods will get the results from service operation. (from server)

6) Service Protocols :-

→ In order to communicate on the web, we require Http protocol.
→ In order to transfer the objects, we required SOAP protocol.
→ communicate with web service involves Http over SOAP.
→ SOAP stands for "Simple Object Access Protocol"
→ It is a XML based protocol which used to send XML serialized object from service to consumer.

4/11/15

→ Advantages of webservices

Q:- what are the Advantages of webservices ?
Q:- what are the characteristics of webservice?
Q:- what are the features of Webservice ?

→ A webservice is class that contains reusable functionality which can be accessed across the internet.
→ webservice provides the following Advantages:

1) Reusability :-

→ webservice can be reusable across multiple projects/applications.
→ After creating webservice, which can be consumed (accessed) by multiple applications.
→ No need to prepare the same logic in all projects, webservice operations (Methods) provides required functionality.

2) Browsable :-

→ After creating webservice, we need deploy on the web server.
→ So, that other applications can browse using Internet.
→ Due to this feature, webservices can be accessible across the world.

3) Interoperability :-

→ Interoperability is a concept of accessing functionality of one environment from another environment.
→ There are different types of Interoperability.
→ webservice supports "Technology Interoperability".
→ webservice can be accessed from different technology.

Ex:- ASP.NET webservice can be accessed from Java/oracle/IBM applications.

4) Platform Independent :-

→ Web Services uses XML technology to inorder to exchange the data b/w Service application & client application.
→ XML is platform independent, due to this web services is also consider as platform independent.
→ We can access the web services from any platform/technology.

5. Self Description :-

→ web services provides Self Description.
→ It means web services provides metadata about the web services.
→ It includes no.of methods, arguments, argument types, return type etc.,
→ So, that in consumer application we can develop proxy class based on this description.
→ In web Service environment, these details are referred as "WSDL".
→ WSDL stands for Web Service Description Language. It is completely XML based.

6) Provides Security :-

→ Web Services provides security for the resources that are on the web Server (Ex: DataBase, files, etc --)
→ If you allow the other users to access these resources directly, it will causes security issues.

→ Through web services we can overcome those problems.
→ Other applications developers will communicate with web service, WebService will communicate with resources.
→ So, that we can provides security for issues.

5/11/15

Developing Web Services in ASP.NET

→ webservices are used to provide services to other Applications across the internet.
→ We can develop web services by using either ASP.NET (or) J2EE.
→ There are three phases to develop a webservice.

1) Create web service.
2) Deploy the web service.
3) Consume the webservice.

1) Create Web Service :-

→ We need to implement the following steps to create the web service.

1) Create a new ASP.NET Empty web-Application project.
2) Add → New Item → Add web service file (*.asmx) to project.
   Ex:- Webservice1.asmx.
3) Add required methods to webservices class by using [WebMethod] attribute.

Note:-

→ All web Services operations should mark with [web method] attribute.
→ Otherwise ASP.NET Engine doesnot consider those methods as web Service operations.
→ So, the consumers can't access those methods without [web method]

2) Developing Webservice

→ After creating webservice we need to deploy on the webserver.

→ we can use any of the following server.

1) ASP.NET Developmet server.

2) IIS.

Note:-

→ Deploying web service is same as regular ASP.NET webforms.

→ After deployment, we will get URL of webservice. So, that Consumers will accen with that URL.

Ex:- http://localhost:1068/webservice1.asmx.

6/11/15

3) Consume the webservice.

→ After deploying the webservice, we can consume from any .NET application.

Steps:-

1) Create/open any .NET Application.

2) Add service Reference with webservice URL.

3) Import Service Reference namespace after the Reference.

Ex:- using ConsoleApplication1.ServiceReference1

4) Create object for proxy class.

5) Invoke the methods with proxy class object.

Note:-

→ Proxy class will be generated by VS at the time of adding service reference.

Ex:- ClassName: WebServiceNameSoapClient
WebService1SoapClient
LoginServiceSoapClient ...

→ At the time of execution, proxy class doesnot execute methods instead it will communicate with WebService which running in the WebServer.

Example :→

Create webservice to provide operations for consumers using ASP.NET

1. Create new "ASP.NET Empty Web-Application".
2. Add webservice file to the project.
   - Goto Solution Explorer.
   - Right Click on projectname.
   - Add → New Item.
   - Select "WebService"

   Name; WebService1.asmx.
   - Click "Add" Button.
3. Define required operations in webservice.

   fileName : WebService1.asmx.cs.


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace WebApplication1
{
    [WebService]
    public class WebService1 :
    System.Web.Services.WebService  → Base Class.
    {
        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }
```

```
    [WebMethod]
    public int Sum (int x, int y)
    {
      return x+y;
    }
  }
}
```


4- Execute (F5) the WebService.

Note:-

→ Test the webservice in the browser. In order to check ~~methods~~ Operations are working or not.

→ Copy the URL of *.asmx file.

http://localhost:1077/WebService1.asmx.

→ Keep the Service in running mode so that otherapplication can consume.

✱ Create consumer application to consume the above WebService.

1- Create New Console Application. (By open another instance of VS.Net)

2- Add Service URL as Service Reference.

- Goto Solution Explorer
- Right click on "References"
- Choose "Add Service Reference"
- Enter Service URL (which is copied from Browser) & click "Go" Button
- Click OK.

Note:- This step will generate a sub-namespace in your project along with Proxy class & its methods.

Namespace:

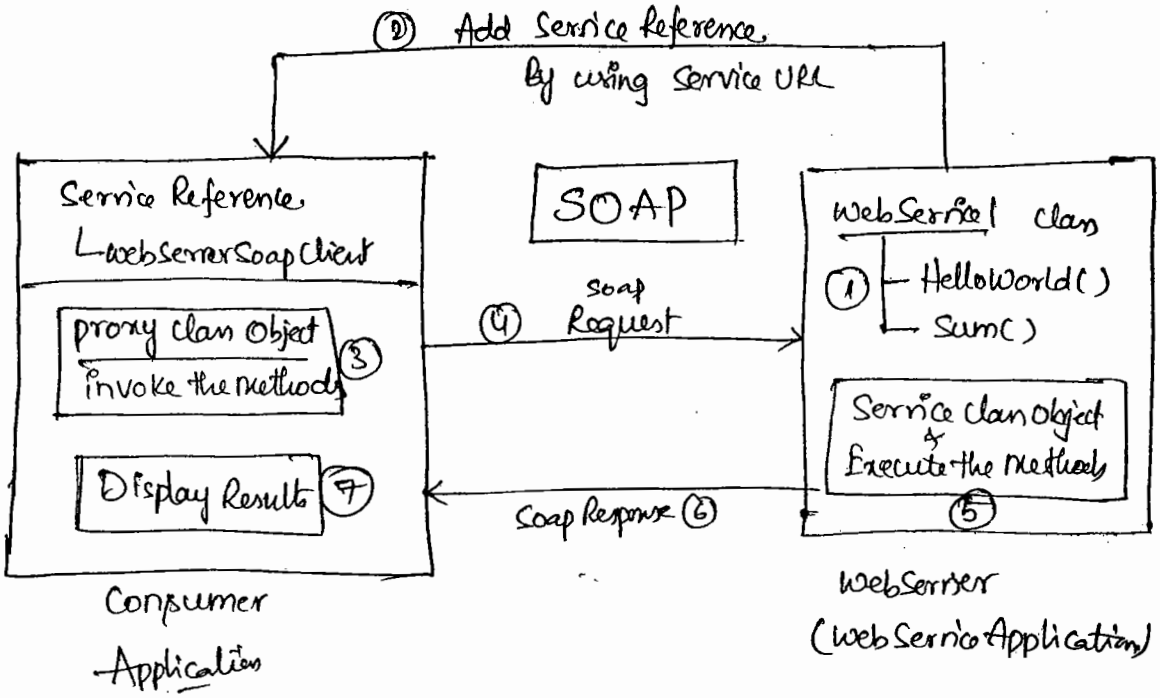
ProjectNamespace.ServiceReference1

ConsoleApplication1.ServiceReference1

3) Write the following code in "Program.cs" file:

```
using ConsoleApplication1.ServiceReference1;

class Program
{
    static void Main()
    {
        WebService1SoapClient proxy = new WebService1SoapClient();
        Console.WriteLine("Results from WebService");
        Console.WriteLine("-----------");
        Console.WriteLine("Hello World Result :" + proxy.HelloWorld());
        Console.WriteLine("sum Result :" + proxy.sum(123,456));
        Console.ReadLine();
    }
}
```

7/7/15

→ create a webservice to develop login service in order to verify user id & password with database:

1) Create new ASP.NET Empty web Application.

2) Add webservice file to project

File Name : Login Service.asmx.

3) Add the required method to Login service class.

```
using System.Data;
using System.Data.SqlClient;
namespace WebApplication
{
  [WebService]
  public class LoginService : WebService
  {
    [WebMethod]
    public bool IsValidUser(string uid, string pwd)
    {
      string b = false;
      string constr = "Server = .; database = "Sathyadb";
                Trusted_Connection = true; ";
      string cmdtext = " select count(*) from userInfo where.
                userid = ' " +uid+ " ' and password = ' " +pwd+ " ' ";
      SqlConnection conn = new SqlConnection(constr);
      SqlCommand cmd = new SqlCommand(cmdtext, conn);
      conn.open();
      int i = (int) cmd.ExecuteScalar();
      Conn.close();
      if(i == 1)
      {
        b = true;
      }
```

```
        else
        {
            b = false;
        }
        return b;
    }
}
```

4) Execute the Login Service & keep it in running mode. So, that other applications can consume & copy that URL from browser.

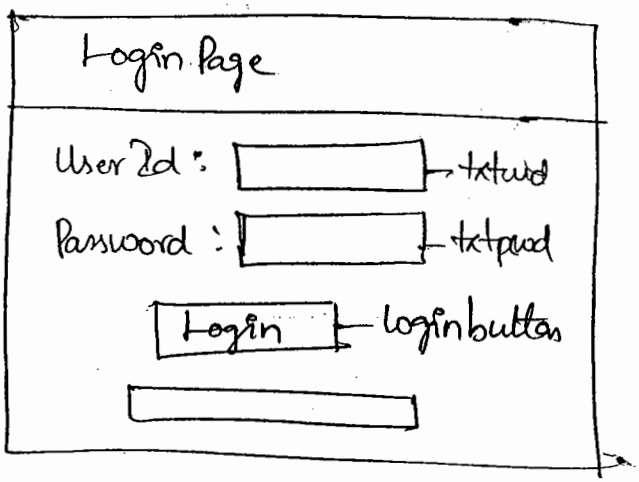
```
http://localhost:1041/LoginService.asmx
```

Consumer Application:-

Create ASP.NET WebApplication to consume the Login Service

1) Create ASP.NET WebApplication.
2) Add the above service URL as Service Reference.
3) Add webform & Design as follows.

a) webform1.aspx



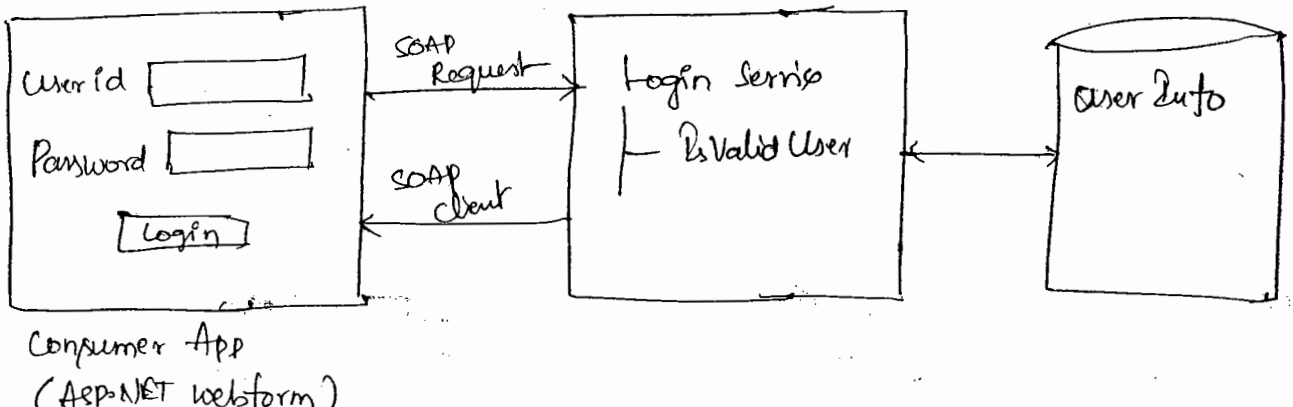
b) webform1.aspx.cs

```
Using System.Drawing;
```

```
public partial class webform1
{
    //proxy class object
    LoginService.SoapClient proxy = new LoginService.SoapClient();
    protected void Login button_Click()
    {
        string uid = txtuid.Text;
        string pwd = txtpwd.Text;
        if(Proxy.IsValidUser(uid, pwd) == true)
        {
            label1.Text = "welcome to" + uid);
            label1.forecolor = color.Green;
        }
        else
        {
            label1.Text = "Invalid userid or password";
            label1.forecolor = Color.Red;
        }
    }
}
```


Note:- Consumer Application doesnot having database & Database related Logic. consumer Application will send the userid & password to webservice & webService will be execute Database Logic by communicating with corresponding database.

Execution procen (or) flow :—



9/11/15

→ WebService that returns complex type data

→ Most of the webservice methods may return scalar types.
→ But it some class cases we need return complex type like class objects.
→ At that time webservice operations return complex types like class objects.
→ To implement this we need create a class which can hold the data.
→ The class that we used to send the data should implement the following rules.

1) Class should be declared as public.
2) properties should be declared as public.

Ex:-
```
public class Emp
{
    public int Empno {get; set;}
    ---
}
```


Note:-
→ Null referenced items cannot be converted as response. It will not be serialized at runtime.

Create a webService that will return Emp details.

↓ Emp-CS

```
public class Emp
{
 public int Empno {get; set;}
 public string Ename {get; set;}
 public string Job {get; set;}.
}
```

2) WebService1.asmx.cs

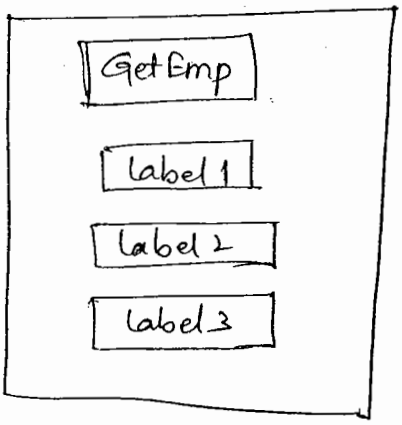
```
public class WebService1 : WebService
{
  [WebMethod]
  public Emp GetEmp()
  {
    Emp obj = new Emp();
    obj.Empno = 1025;
    obj.Ename = "scott";
    obj.Job = "Manager";
    return obj;
  }
}
```


Consumer Application

→ Create windows Application & consume the above Service.

→ Create new windows forms application & add above webservice as Service reference.

1) Form1.Design :-



2) Form1.cs :-

```
using WindowsFormsApplication9.ServiceReference1
public partial class Form1 : Form
{
```

```
private Void form1_Load ( )
{
    label1.Text = string.Empty;
    label2.Text = string.Empty;
    label3.Text = string.Empty;
}
WebService1SoapClient proxy = new WebService1SoapClient();
private void button1_Click ( )
{
    Emp obj = proxy.GetEmp();
    label1.Text = obj.Empno.ToString();
    label2.Text = obj.Ename;
    label3.Text = obj.Job;
}
}
```


Task:-

Database : Student

H.T.No	StudentName	Subject-1	Subject-2	Sub-3	Sub-4	Sub-5	Total

10/11/15

→ Create the webservice that returns Department details in the form of DataTable

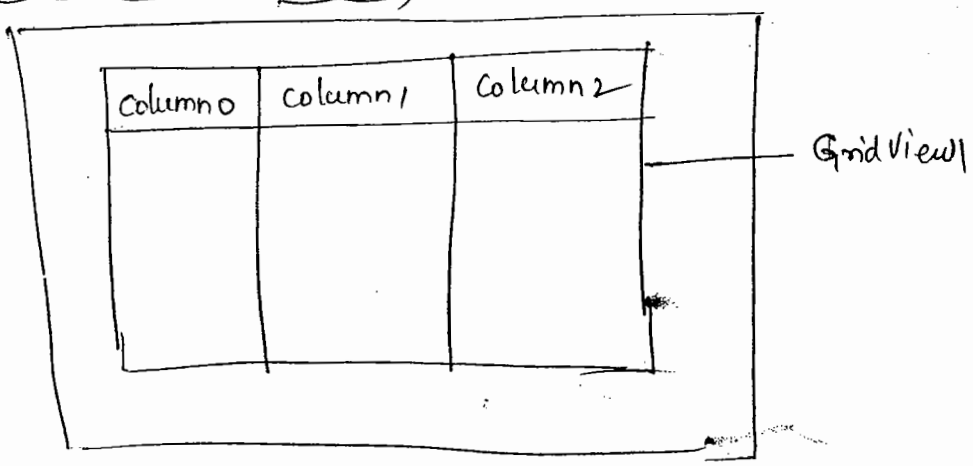
1) webService1.asmx

```
using System.Data;
using System.Data.SqlClient;

[webService]
public class webService1 : WebService
{
    [WebMethod]
    public DataTable GetDepts()
    {
        string connstr = "server=.;Database=Sathya;Trusted_Connection=true;";
        string cmdText = "Select * from Dept";
        SqlDataAdapter da = new SqlDataAdapter(cmdText, connstr);
        DataTable dt = new DataTable("DeptTable");
        da.fill(dt);
        return dt;
    }
}
```


2) Consumer Application (ASP.NET project)

1) webform1.aspx (design)



2) webform1.aspx.cs

```
using System.Data;
using WebApplication1.ServiceReference1;

Void page_load()
{
 WebService1SoapClient proxy = new WebService1SoapClient();
 DataTable t1 = proxy.GetDepts();
 GridView1.DataSource = t1;
 GridView1.DataBind();
}
```


WebService as Technology Interoperability

→ webservices can be consumed by different Technologies.

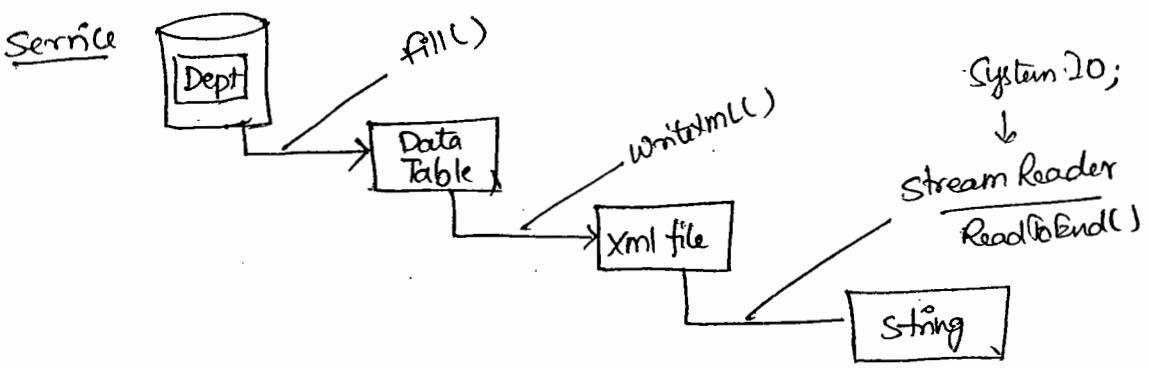
→ Suppose if you return "DataTable" object from ASP.NET WebService operation, .NET application can consume that service.

→ But Java application cannot consume that service, due to "DataTable" class not excists in Java.

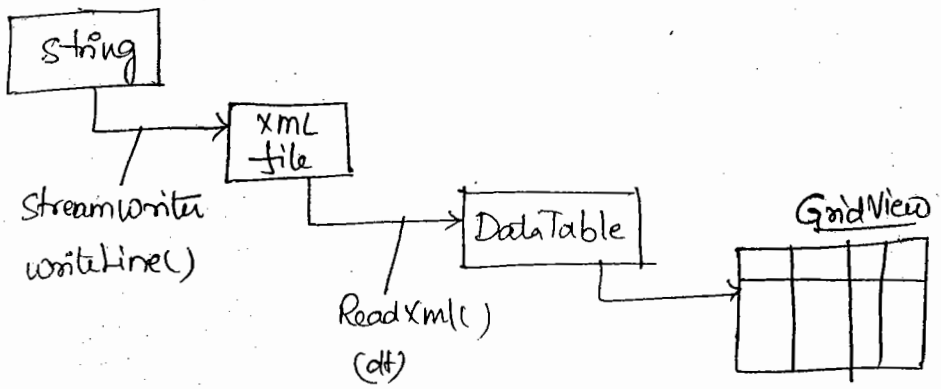
→ Same issues with other applications/Technologies.

→ In Order to support all the technologies, we can send the results in string format which contains xml formatted data.

In Service Application



In Consumer Application



Create a WebService to Send the DataTable values in string format.

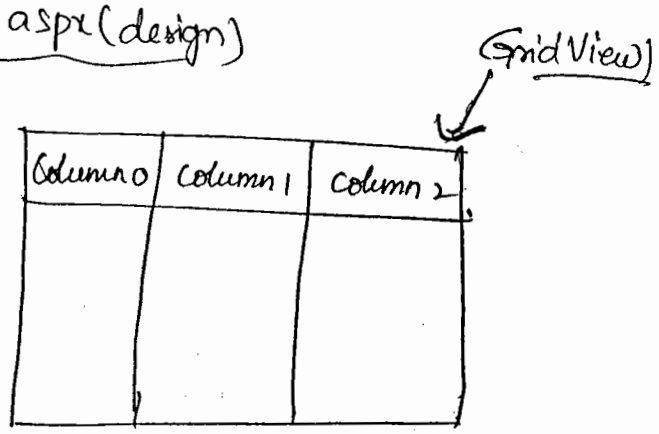
Servic Application

```
using System.Data;
using System.Data.SqlClient;
using system.IO;

public string GetDepts()
{
 string connstr = "server=.; database='Sathya'; Trusted_Connection=true;";
 string cmdText = "Select * from Dept";
 SqlDataAdapter da=new SqlDataAdapter (cmdText, connstr);
 DataTable dt= new DataTable("DeptTable");
 //Step1
 da.fill(dt);
 //Step2
 string fname = server.MapPath(".")+"\\DeptData.xml";
 dt.WriteXml(fname, XmlWriteMode.WriteSchema);
 //Step3
 StreamReader sr= new StreamReader(fname);
 string str= sr.ReadToEnd();
 sr.close();
} return str;
```

Consumer Application

1) Webform1.aspx(design)



2) webform1.aspx.cs :-

```
using WebApplication1.ServiceReference1;
using System.IO;
using System.Data;

Void page_Load()
{
  WebService2SoapClient proxy=new WebService2SoapClient();

  //Step1
   String Str = proxy.GetDepts();

  //Step2
  string fname = Server.MapPath(".") + "\\DeptData_Consumer.xml";
  StreamWriter sw=new StreamWriter(fname);
  sw.WriteLine(str);
  sw.close();

  //Step3
  DataTable dt=new DataTable();
  dt.ReadXml(fname);

  //Step4
  GridView1.DataSource=dt;
  GridView1.DataBind();
}
```

12/10/15

Q: How to consuming online webservices

Q: Differences b/w aspx & asmx?

Q: what are the differences b/w Webform & webservices?

webform	webservices.
1) Webform files having extension aspx.	1) webservices having extensions asmx.
2) aspx stands for "Active Server Page"	2) asmx stands for "Active Server method"
3) Every webform class is inherits from Page class.	3) Every webservice class is inherits from webservice.
4) The classes related to webforms (or) belongs to "System.web.UI" namespace.	4) classes related to webservices (or) belongs to "System.Web.Services" namespace.
5) Webforms will generate the op in HTML format.	5) webservice will generate the output in xml format.
6) Webforms are accessible and understandable by End user.	6) webservices can not be understandable by Enduserp.
7) webforms cannot be accessed in other Applications.	7) webservices can be accessible in other Applications.
8) Webforms containg Designing and coding.	8) webservices having only coding.
9) At the time of developing Webforms we targets the Enduser.	9) At the time of developing webservices we target the other Applications.
10) No need to perform Serialisation & deserialisation in webform request processing.	10) we need to perform serialisation & deserialisation to process the request of webservice.

⇒ Consuming OnLine WebServices.

→ we can consume online webservices with the help of corresponding service URL.

→ Before consume any webservice, first we need to test the service in browser.

→ At the time of testing service, observe the following points.

- operation provided.
- Arguments
- Return Data(Type)

→ After that we can consume from any .NET Application.

→ couple of websites are providing online free webservices to consume.

Ex:- 1) http://wcf.cdyne.com/weatherWS/weather.asmx.

2) http://www.webservicex.net.

consuming Weather Webservice

1) URL: http://wcf.cdyne.com/weatherWS/weather.asmx.

2) operation
   - → GetCityforecastByZIP()
   - → GetCityWeatherByZIP()
   - → GetWeatherInformation()

3) Classes
   - → weather — Service Class
   - → WeatherReturn — DataClasses
   - → WeatherDescription — Data classes.

Note: The Above webservice will generate two end-points (communication URLs) in web.config/app.config.

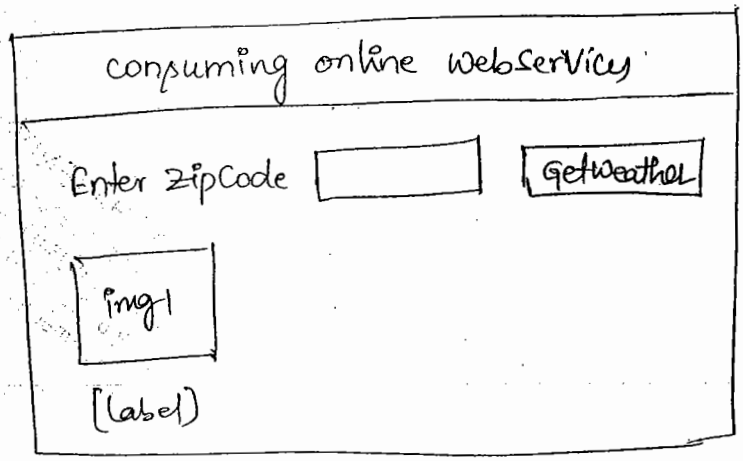
Remove second End-point "customerEndPoint", otherwise URL's,

13/11/15

Example :-

Consuming online webservices by using ASP.NET :-

webform1.aspx — Add ServiceURL to ASP.NET WebApplication



webform1.aspx

```
using WebApplication1.ServiceReference1;

WeatherSoapClient proxy = new WeatherSoapClient();

Void Button1_Click()
{
  WeatherReturn obj = proxy.GetCityWeatherByZip(TextBox1.Text);
  String str = "City:" + obj.City + "<br>";
  Str = Str + "State:" + obj.State + "<br>";
  Str = Str + "Description:" + obj.Description + "<br>";
  Str = Str + "Temperature:" + obj.Temperature + "<br>";
  Str = Str + "RelativeHumidity:" + obj.RelativeHumidity + "<br>";

  WeatherDescription[] ar = proxy.GetWeatherInformation();
  foreach(WeatherDescription item in ar)
  {
    if (item.WeatherID == obj.WeatherID)
    {
      Image1.ImageUrl = item.PictureURL;
    }
    break;
  }
}  → Label.Text = "Str";
```

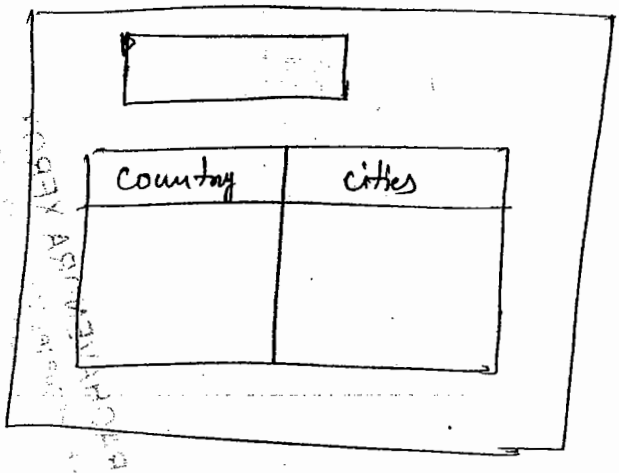
✱ Consuming Online WebServices in ASP.NET WebApplication

Service Url: http://www.webservicex.net/globalweather.asmx.

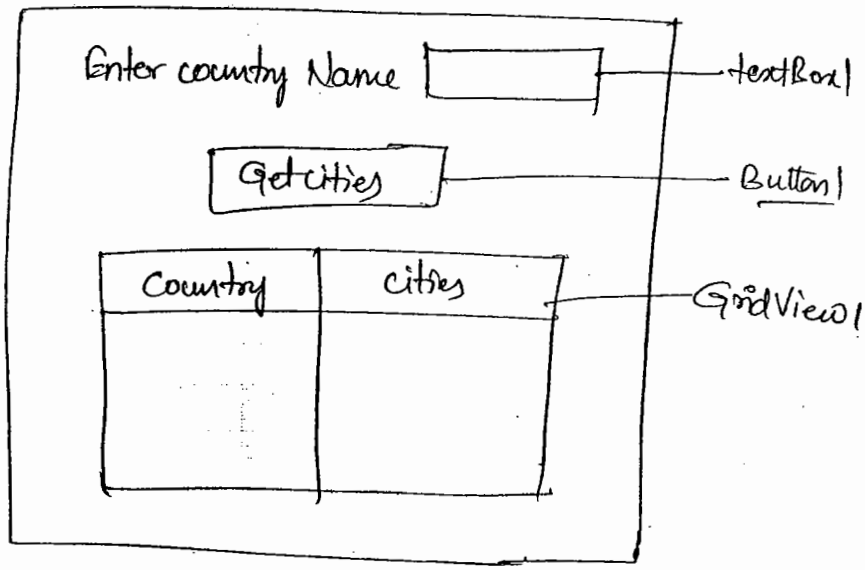
class name: globalweather.asmx.

service operation: GetCitiesByCountry.

webform1.aspx (design)



1) Create new asp.net web application & add the above Servia URL as service Reference.
2) Remove second end-point from web.config file
3) Add webform & design as follows.



webform1.aspx.cs

```
using WebApplication1.ServiceReference1;
using System.IO;
using System.Data;
```

```
GlobalWeatherSoapClient proxy = new GlobalWeatherSoapClient();

Void Button1_click()
{
  //step1
  string str = proxy.GetCitiesByCountry(TextBox1.Text);
  //step2
 string fname = server.MapPath(".") + "//ServiceResults.xml";
  StreamWriter sw = new StreamWriter(fname);
  sw.WriteLine(str);
  sw.Close();
  //step3
  DataSet ds = new DataSet();
  ds.ReadXml(fname);
  //step4
  GridView1.DataSource = ds.Tables[0];
  GridView1.DataBind();
}
```


14/11/15

Interview Questions

1) what is webservices? why do we use Webservice? what is the purpose of webservice?

→ Webservice is a component (reusable class). That will provide services to Heterogenious Applications. which can be accessible across the Internet.

2) What is SOAP ? what is the significance of SOAP in webservice ?

→ SOAP stands for "Simple Object Access Protocol".

→ SOAP protocol is developed based on the XML Technology, which is used to exchange the data blw service provider & Service consumer.

→ SOAP will work on top of http, it can be able to transfer structural data like class objects, Arrays, etc.,

3) What is WSDL ?

→ WSDL stands for "WebService Description Language"

→ It is used to providing self description about Web service. (MetaData).

→ Consumer Application can build (prepare) the proxy classes based on the MetaData.

4) What is UDDI ? www.UDDI-xml.org

→ UDDI stands for "Universal Description Discovery and Integration".

→ It provides directory service (repository) for storing information about webServices.

5) How to provide Description for WebServices Operations ?

```
[WebMethod(Description="Text")]
public string MethodName()
{
    ==
}
```

6) How to ~~Catch~~ cache webService Response?

→ By default, caching is not enabled.

→ We can enable by using "cacheDuration" property of "WebMethod Attribute".

usage:-

```
[WebMethod(CacheDuration=20)]
public string HelloWorld()
{
    return "Hello World";
}
```

7) How to enable session state in WebService?

```
[webmethod(EnableSession = true)]
public string HelloWorld()
{
    ≡
}
```

8) what is namespace in which all types related WebServices are available?

System.Web.Services.

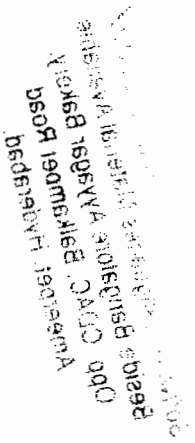
9) How to enable your WebService so, that it can be accesed from JS/jquery/AJS?

System Attribute : [ScriptService]

Namespace : System.Web.Script.Services.

By default ~~WebService~~ JavaScript webServices does not allow ~~webServices/JS~~ JavaScript

we can enable by using Script Service attribute.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

WCF

(Windows Communication foundation)

→ What is Wcf?

→ wcf stands for Windows communication foundation.

→ wcf is a new framework from Microsoft to implement distributed applications across multiple networks and multiple technologies.

→ wcf was introduced in the year 2008.

→ What is distributed Application?

→ An Application that can be used by different users/applications.

→ This Application can be accessed across the network.

→ These applications are developed by using distributed technologies.

→ What are the examples of distributed technologies?

→ There are several distributed technologies developed by Microsoft.

Ex:
1. DCOM (Language dependent)
2. MSMQ
3. Remoting. (Intranet, Hetegogeneous)
4. WebServices.
5. WCF.

→ Why Microsoft is introduced wcf eventhough it contains webservice as a Distributed Technology? (or)

What are the Limitations of WebServices?

→ ASP.NET webServices having the following limitations.

1) Supports only WebHosting :

→ WebServices can be hosted only on web Server.
→ There is no option to host without using any web Server.
→ We have to use either IIS / ASP.NET Development Server / IIS Express to host.
→ Hosting the resources on web server is called "WebHosting".

2) Consumed by only Internet Applications :-

→ WebServices supports only webHosting so that communication is depends on the Http protocol.
→ Http based applications only can consume webServices i,e., Internet based.
→ Other protocols like TCP does not support by webService.

3) Supports Only Transport Level Security :-

→ WebServices can implement security by using Authentication options of ASP.NET.
→ These options can support only transport level security i,e., Check the user is valid user or not.
→ But it does not support Message Level security i,e., Check the data is valid or not.

4) Supports Only SOAP :-

→ WebServices using SOAP protocol to transfer the data between applications.
→ SOAP supports only XML based serialization & de-serialization.
→ It does not supports binary based serialization & de-serialization.

5) Difficulty to Porting :-

→ porting is a concept of changing service application from one network to another.

→ In Webservices porting will much difficult.

→ We need to develop new application to implement porting.

6) WebServices doesnot supports WS-* :-

→ After 2005, there were few standards are adding Webservice like WS-Security, WS-Transport, etc..,

→ These all standards are called as "WS-*" specifications.

→ These standards are not supported in .NET webservices.

18/11/15

⇒ Features of WCF :-

Q: What are the advantages of WCF over WebServices?

Q: What are the features of WCF framework?

→ WCF is a new framework from Microsoft used to develop service based applications.

→ WCF framework provides the following features.

1) Multiple Hosting Support :

→ WCF services can be hosted with multiple options.

→ It supports
- WebHosting.
- SelfHosting (ManagedHosting)

→ We can host WCF service either IIS (or) WAS server to implement webHosting.

→ IIS supports only HTTP.

→ WAS supports HTTP, TCP, etc.,

→ WAS (Windows Proces Activation Services).

2) Multiple Networks :-

→ WCF services can be consumed from multiple Networks.
→ Either Internet (or) Intranet users can access WCF services
→ Allowing additional network is adding addition end-point to service.

3) Multiple Level of Security :-

→ WCF supports multiple levels of security options to services.
→ It includes both message level (or) transport level.
→ So, that our service will be accessible trusted users with trusted messages.

4) Multiple Message formats :-

→ At the time of exchange the data b/w client and Service, data need to serialized / deserialized according to the protocols.
→ WCF supports all formats of serialization techniques.
→ It supports all communication protocols like HTTP, HTTPS, TCP ...

5) Multiple Porting :-

→ In WCF, porting will be easy to implement.
→ Porting is a concept of allowing the users of another network.
→ Adding additional endpoints will be allow the users of another network.
→ Porting is very easy to implement just by updating *.config file.

6) WS-* Specification Support

→ New WebService specification (WS-*) that are introduced after 2005 are called "WSE".
→ These specifications are supported by WCF.

7) Multiple Technologies :-

→ Similar to WebServices, WCF also can be accessed from multiple technologies.

→ You can access from any application .net, java, Oracle, etc.,

→ This feature is called "Technology Interoperability".

⇒ Developing WCF Services steps :-

Service Application	Consumer Application
1) Service Interface	1) Add Service Reference ↳ Service Proxy.
2) Service class	2) Create Proxy class object
3) End point Configuration.	3) Invoke service operations.
4) Hosting	4) Display the results to End-user

19/11/15

⇒ Create WCF Service to provide the operations to consumer Application :

1) Create ASP.NET Empty WebApplication.

2) Add "Wcf Service" file to the project.

3) → Goto Solution Explorer.
→ Right click on Project Name.
→ Add → New Item.
→ Select "Wcf Service" from "Add New Item" dialog Box.
fileName : Service1.svc
→ Click "Add" button.

Note :- This step will add the following files.
1) IService1.cs
2) Service1.svc
3) Service1.svc.cs

3) Defines the required Operations, in the Service interface as follows.

fileName : IService1.cs

```
using System;
using System.Collections.Generics;
Using System.Linq;
Using System.Runtime.Serialization;
Using System.ServiceModel;
Using System.Text;

namespace WebApplication
{
  [ServiceContract]
  public interface IService1
  {
    [OperationContract]
    String Greeting();
  }
    String Testing();
}
```


4) Implement the above interface in Service1.svc.cs file.

```
namespace WebApplication
{  public class Service1 : IService1
  {public string Greeting()
    {
      string str = "Message from WCF service Operation";
      return str;
    }
    public string Testing()
    {
      return "This is Testing Method()";
    }
  }
}
```

5) Run the Service. (Hosting)

→ Goto Solution Explorer.

→ Right click on "Service1.svc" file.

→ Select "View in Browser".

6) Copy the URL service from Brooser.

Ex: http://Localhost:1238/Service1.svc

Note:

→ The Above service URL will be used by consumer Application.

→ keep the Service in running mode.

⇒ Create the consumer Application to access the above WCF Service.

i) Create console Application Add Service URL as Service Reference.

→ Goto Solution Explorer Right click on "Reference".

→ Select "Add" Reference Service Reference" option.

⇒ And paste the URL in the Address bar which is specified with Add Service Reference dialog Box.

→ Click "GO" Button and click "Ok" Button.

ii) Write the following code in program.cs file in consume service operation

```
using ConsoleApp.ServiceReference1;
namespace ConsoleApp
{
  Class Program
  {
    static void Main()
    {
     ServiceClient proxy = new ServiceClient();
     string str = proxy.Greeting();
     C.W.L ("Result" + str);
     C.R.L();
    }
  }
}
```

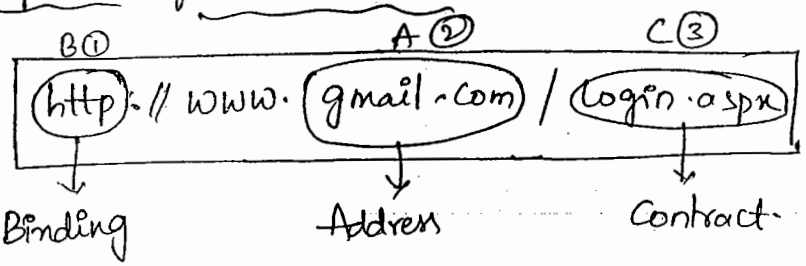
iii) Execute the consumer Application to get the Result..

Note:
→ ServiceClient, it is a class Name the class consider as proxy class. The proxy class is generated based on the "Service Interface" in "consumer Application" at the time as adding "Service Reference".

→ Proxy class cannot Execute the operation in client(or) consumer Application. Instead it will communicate with Service in order to Execute the operations in Service applications.

20/11/15

⇒ End points of WCF Service



Binding :- "How" to communicate with Service & client

Address :- "Where" is exactly service is Execution/Running.

Contract :- "What" Service are there in the operations as the Server.

→ In WCF service application & WCF consumer application end points plays very important role.

→ communication b/w Service & consumer is based on the end-points only.

What is the purpose of the End-points ?

→ End points provides a fecility to client so that they can communicate with service.

→ End points allow the External world to communicate with Service.

⇒ What are the contents of the End points?

→ Each End point contains three important properties.

→ These properties are called "ABC of End-point".

1) A – Address

→ It represents where service running. Generally it is a URL.

Ex:- http://localhost:1234/Service/net-tcp://

2) B – Binding

→ Binding represents the protocols, transactions, security, Serialization techniques etc;

→ In WCF framework, each Binding available as a class.

Ex:- BasicHttpBinding, WSHttpBinding, Net-TCP Binding---

3) C- Contract

→ Contract specifies the service interface name.

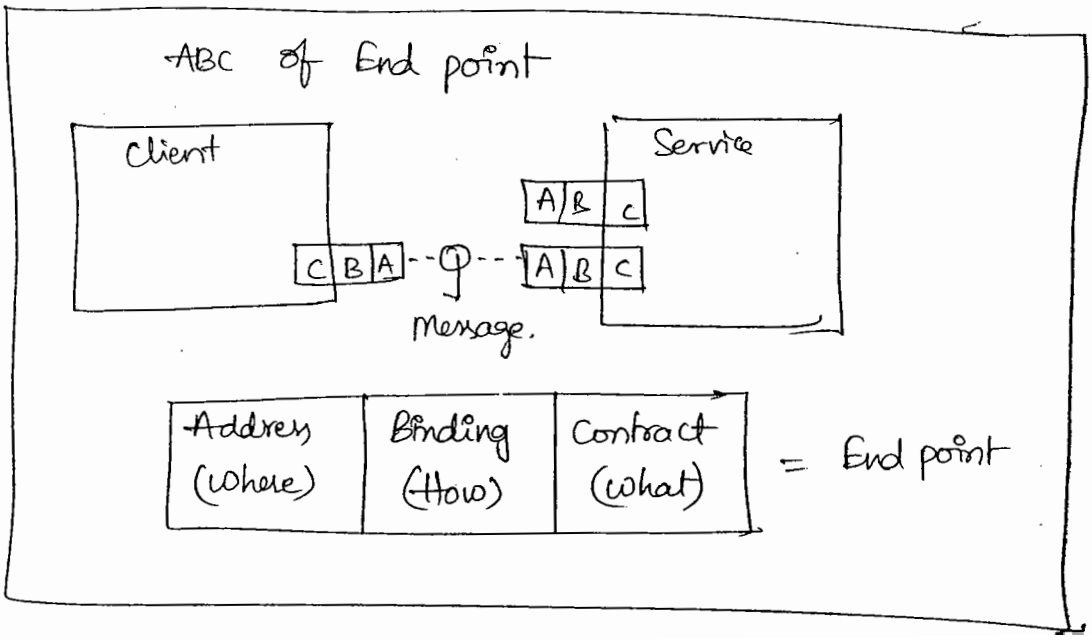
→ It represents what service can be accessed by client through the end-point.

Ex:- IService1, IEmpService, IMathService..

Note:-

→ A Service can provide more than one end-points.

→ But a client side can provide only one end-point at a time.



→ what is mex end-point? why do we need mex end-point in Service?

WCF Service provides two types of End-points.

i) Meta data End-point (mex)

ii) Service End-point

i) mex :- (metaData End-point)

→ It is called as mex (or) metadata End-point

→ It is used to get metadata (data about Service) in order to generate proxy class.

ii) Service End-point :-

→ Service End-points are actually involved at the time of Execution.

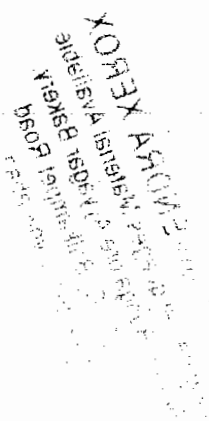
→ These end-points are actually communication end-points to invoke service operations.

→ How to Create End-points?

→ We can Create the End points in 2 ways.

1) Manual Approach.

2) Using WCF configuration Editor.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

21/11/15

→ Create WCF service to host on the IIS directly as webHosting.

1) Create new ASP.NET Empty website.
2) Provide the following URL at the time of creating project.

   http://Localhost/WCFServiceDemo

3) Add Math service file to the project.

   fileName : MathService.svc

4) Define the required service interface and class in the following files.

→ IMathService.cs

```
using System.

[ServiceContract]
public interface IMathService
{
  [OperationContract]
  int Add(int x, int y)

  [OperationContract]
  int power(int x, int y)

  [OperationContract]
  int factorial (int x)
}
```

→ MathService.cs

```
public class Mathservice: IMathService.
{
   public int Add(int x, int y)
   {
      int z = x+y;
      return z;
   }
```

```
    public int Power(int x, int y)
    {
        int z=1;
        for(int i=1; i<=y; i++)
        {
            z = x * z;
        } return z;
    }
    public int factorial(int x)
    {
        int z=1;
        for(int i=0; i<=x; i++)
        {
            z = z * i;
        }
        return z;
    }
}
```


5) Execute the *.svc file and copy the URL from browser.

Note:

→ The Above project is deployed permenantly on IIS. No need to keep the visual studio.

→ you may the close the visual Studio. IIS will running in the background so, that your service is accessible always.

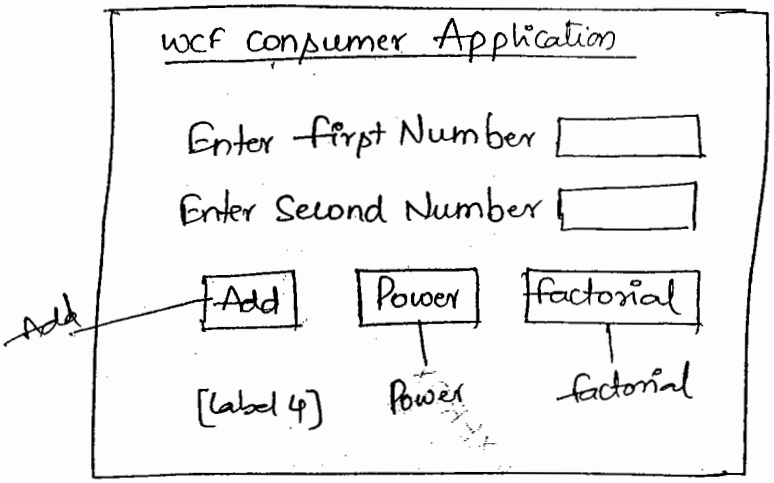
⇒ Developing the Windows application to consume the above wcf Service.

Service URL:

http://localhost/WcfServiceDemo/MathService.svc

→ Create Windows application & add the service Reference

→ form1.cs [Design]



→ form1.cs [code]

```
using WindowsApplication1.ServiceReference1;

public partial class form1 : form
{
  MathServiceClient proxy = new MathServiceClient();

  private void form1_Load( )
  {
    Lable4.Text = " ";
  }
  private void Add
  private void ~~button1~~_click( )
  {
    int x = int.Parse (textBox1.Text);
    int y = int.Parse (textBox2.Text);
    int result = proxy.Add (x,y);
    Label4.Text = " Result is :" + result ;
  }
  private void Power_click( )
  {
    int x = int.Parse (textBox1.Text);
    int y = int.Parse (textBox2.Text);
    int result = proxy.Power (x,y);
    Label4.Text = " Result is :" + result;
  }
```

```
private void factorial_click()
{
    int x = int.Parse(textBox1.Text);
    int result = proxy.factorial(x);
    Label4.Text = "Result is:" + result;
}
}
```

23/11/15

⇒ Self-Hosting:-

→ WCF Services can hosting by using a option called "self-Hosting".
→ In this option, we are no need to depends on any webServer.
→ By writing .Net program we are hosting the service.
→ Self Hosting is also called as "Managed-Hosting".
→ To implement this option, we need to use a special class called "ServiceHost".

Library Info:

Library : System.ServiceModel.dll.
Namespace : System.ServiceModel.
Class Name : ServiceHost.

Methods:
open() – start the service.
close() – Stop the service.

Note:-
→ Provide type details of Service class as a argument to ServiceHost constructor.

```
ServiceHost host = new ServiceHost(typeof(Service1));
```

* Create console Application to host wcf service by using self Hosting option.

1) create console application and wcf service file.

filename : Service1.cs

(*) This step will add required *.cs files, app.config and Libraries.

2) IService1.cs

```
[ServiceContract]
public interface IService1
{
  [OperationContract]
  bool IsValidUser(string uid, string pwd)
}
```

3) Service1.cs


```
using System.Data.SqlClient;

public class Service1 : IService1
{
  public bool IsValidUser(string uid, string pwd)
  {
    string constr = "Server=.; Database=SathyaDb; Trusted_Connection=
                                                    true;";
    string cmdText = "SELECT COUNT(*) from UserInfo where
                  UserId = '" + uid + "' and Password = '" + pwd + "'";

    SqlConnection conn = new SqlConnection(conStr);
    SqlCommand cmd = new SqlCommand(cmdText, conStr);
    conn.open();
    int i = (int) cmd.ExecuteScalar();
    conn.close();
    if(i>0)
    {
      return true;
    }
```

```
    else
    {
      return false;
    }
  }
}
```

4) Program.cs

```
using System.ServiceModel;

class Program
{
  static void Main()
  {
    ServiceHost host = new ServiceHost(type(service1));
    host.Open();
    Console.WriteLine("Service is started. Press any key to stop the service");
    Console.ReadLine();
    host.Close();
  }
}
```


5) End-point configure

Hint: use WCF configuration editor to prepare end points.

Tools → WCF Service Configuration Editor.

→ Provide end-point details in app.config file.
→ Right click on app.config file in solution Explorer.
→ Select "Edit WCF Configuration".

→ Services → ConsoleApplication1.Service1 → Host.(select)
→ It will show "BaseAddress" in right side panel.
→ Click "Edit" Button & provide the following Address.

http://localhost:5566/Service1

→ Save & close the Editor.

6) Execute the application to host the service.

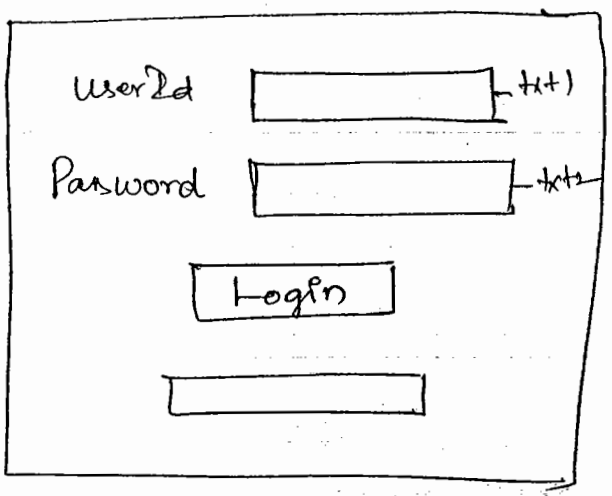
Note :- Open the Visual Studio in "Administrator" mode.

Create windows Application to consume the above service.

Service Url :- http://localhost:5566/Service1.

1) Create the windows Application & Add the service reference

form1.cs Design.



form1.cs (Code)

```
using WindowsformApplication1.ServiceReference1;

public partial class form1:form
{
   Service1Client proxy = new Service1Client();
   private void button1_Click()
   {
     bool b = proxy.IsValidUser(text1.Text, text2.Text);
     if(b==true)
     {
       label3.Text = "Valid User";
     }
     else
     {
       label3.Text = "InValid User";
     }
}
```

24/11/15

⇒ update previous service application by adding another end-point for TCP protocol.

1) Open service application and add new service end point in app.config.

Steps

→ Right click on app.config.
→ Select "Edit WCF Configuration"
→ In left side configuration panel find the Services option.
→ Expand Services → consoleapplication40.Service1 → [End points]
→ Right click on [End points] folder.
→ "New Service Endpoint"
→ Provide the required details for new end point in left side panel.

Name : endpoint2
Address : net.tcp://localhost:5577/Service1
Binding : netTCP Binding.
Contract : consoleapplication40.IService1.

Note:-

→ Select Http end point and update name property.
Name : EndPoint1
→ Save and close.

2) Execute the project to start the service.

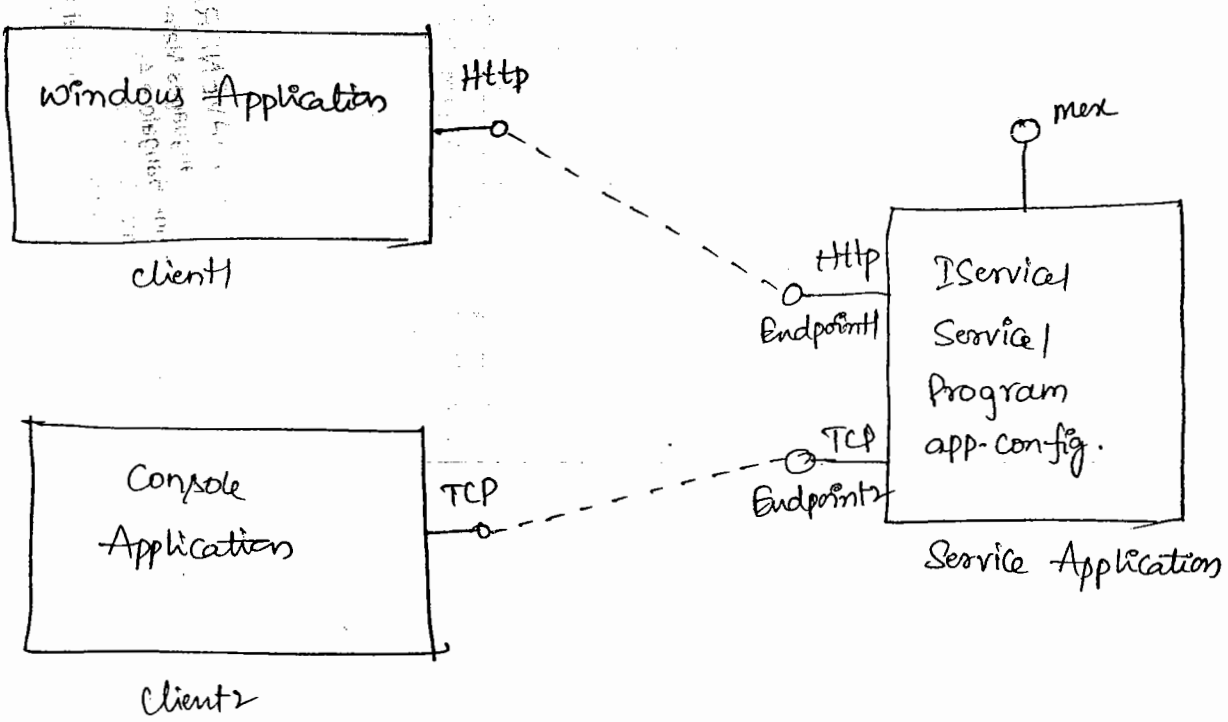
Note:- Now the above service application provides 2 end-points.

1) for http. 2) for tcp

3) Create the console application to consume the above service by using TCP End point.

using consoleapplication1.ServiceReference1;

```
class Program
{
  static void Main()
  {
    ServiceClient proxy = new ServiceClient("EndPoint2");
    Console.WriteLine("using NetTCP End point");
    bool b= proxy.IsValidUser("admin","admin123");
    if(b == true)
    {
      C.WL("Valid User");
    }
    else
    {
      Console.WriteLine("Invalid user");
    }
    Console.ReadLine();
  }
}
```

DataContract in WCF

→ WCF framework provides different types of contracts as attributes to organise the service

→ These attributes provides required information to wcf framework so, that at the time of execution wcf framework will makes the decission.

Attributes :-

1) ServiceContract : This attribute is applicable to Service interface.

2) OperationContrat : This attribute is applicable to service members.

3) DataContract : This attribute is applicable to class that is used to exchange the data.

4) DataMember : This attribute is applicable to members of the DataContract class.

5) FaultContract : This attribute is applicable to exception handling related members.

Usage :-

```
[DataContract]
publi class Emp
{
   [DataMember]
   public int Empno { get ; set ; }
}
```

* Create Employee Service that returns Employee details by using DataContract concept.

1) IEmpService . c s

```
namespace consoleApplication1
```

```
{
    [ServiceContract]
    public interface IEmpService
    {
        [OperationContract]
        Emp GetEmp(int eno);
    }
    [DataContract]
    public class Emp
    {
        [DataMember]
        public int Empno {get; set;}
        [DataMember]
        public string Ename {get; set;}
        [DataMember]
        public string Job {get; set;}
        [DataMember]
        public int Salary {get; set;}
        [DataMember]
        public int Deptno {get; set;}
    }
}
```

2) EmpService.cs

```
using System.Data.SqlClient;
public class EmpService : IEmpService
{
    public Emp GetEmp(int eno)
    {
        Emp obj = new Emp();
        SqlConnection conn = new SqlConnection("Server=.; database=sathydb;
                                               Trusted_Connection=true;");
```

```
SqlCommand cmd = new SqlCommand("Select * from Emp", conn);
SqlDataReader dr = cmd.ExecuteReader();  //where Empno = "+eno, conn);
if(dr.HasRows)
{
   dr.Read()
   obj.Empno = (int) dr["Empno"];
   obj.Ename = (string) dr["Ename"];
   obj.Salary = (int) dr["Salary"];
   obj.Job = (string) dr["Job"];
   obj.Deptno = (int) dr["Deptno"];
}
dr.close();
conn.close();
return obj;
}
}
```


3) Program.cs

```
using System.ServiceModel;

class Program
{
   ServiceHost host = new ServiceHost(typeof(EmpService));
   host.Open();
   Console.WriteLine("Employee Service is started. Press any key to stop
                      the Service");

   Console.ReadLine();
} host.Close();
```

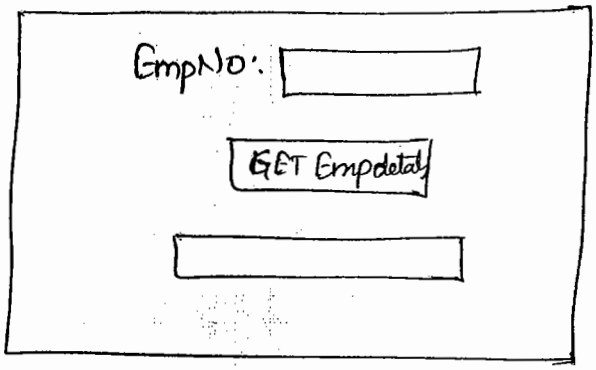
→ update the base address in app.config. as follows.

// http : localhost:5588 / EmpService /.

Consumer Application :-

→ Create ASP.Net Web Application to consume the above service.

1) webform1.aspx (design)



2) webform1.aspx.cs

```
using WebApplication1.ServiceReference1;

class webform1
{
    EmpServiceClient proxy = new EmpServiceClient();

    void button1_Click()
    {
        int n = int.parse(TextBox1.Text);
        Emp obj = proxy.GetEmp(n);
        if(obj.Empno != 0)
        {
            string str = "EmpNo:" + obj.Empno + "<br>";
            str = str + "Ename:" + obj.Ename + "<br>";
            str = str + "Job:" + obj.Job + "<br>";
            str = str + "Salary:" + obj.Salary + "<br>";
            str = str + "Deptno:" + obj.Deptno + "<br>";
            label1.Text = str;
        }
        else
        {
            label1.Text = "InValid Empno";
        }
    }
```

Task 1:- ~~Add~~ perform those operations for previous Example.

1) string AddEmp(Emp obj)
2) string DeleteEmp(~~Emp ob~~ int eno)

25/11/15

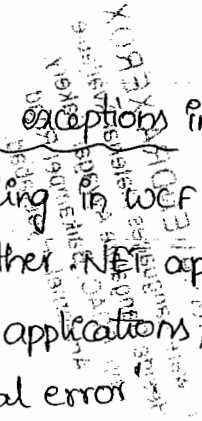
Exception Handling in WCF

→ In .Net Applications exceptions are handled by CLR as a default execution handling.

→ If CLR handles the exception, it will throw the exception details and stop the execution.

→ In order to execute application smoothly without closing, we need to handle the exception.

→ In regular application execution, CLR will provide proper details about exception.



How to handle the exceptions in WCF ?

→ Exception handling in WCF application is slightly different from exception handling in other .NET applications.

→ In WCF client applications, CLR will throw only one exception saying "Server internal error"

→ It will not provide actual details about exception in client application.

→ If you want to provide proper detailed information to client application about exception, we should handle in WCF exception handling approach.

→ By default, client application will get an exception called "FaultException"

Implementation :-

1) Provide [FaultContract] attribute to methods in service interface.
2) Define seperate class to carry the details through FaultException by using [DataContract].

3) Throw the faultException in service class methods with DataContract class object.

MathServiceFault

4) Catch the object in client application.

→ Create WCF application to implement Exception Handling.

1) IMathService1.cs

```
namespace ConsoleApplication1
{
  [ServiceContract]
  public interface IMathService1
  {
    [FaultContract(typeof(MathServiceFault))]
    [OperationContract]
    int Division(int x, int y);
  }
  [DataContract]
  public class MathServiceFault
  {
    [DataMember]
    public string Message {get; set;}
    [DataMember]
    public string Operation {get; set;}
    [DataMember]
    public string Reason {get; set;}
  }
}
```


2) MathService1.cs

```
public class MathService1 : IMathService1
{
  public int Division(int x, int y)
  {
    int z = 0;
```

```
try
{
  z = x/y;
}
catch(Exception exp)
{
  MathServiceFault obj = new MathServiceFault();
  obj.Message = exp.Message;
  obj.Operation = "Exception occur in Division Operation";
  obj.Reason = "Due to division of a number with zero this exception happens";
  FaultException<MathServiceFault> faultobj =
                new FaultException<MathServiceFault>(obj);
  throw faultobj;
}
return z;
}
}
```


3) Program.cs

```
using System.ServiceModel;
class Program
{
  static void Main()
  {
    ServiceHost host = new ServiceHost(typeof(MathService1));
    host.Open();
    Console.WriteLine("Service is started. Press any key to stop service");
    Console.ReadLine();
    host.Close();
  }
}
```

4) update the Base-Address in App.config file as follows.

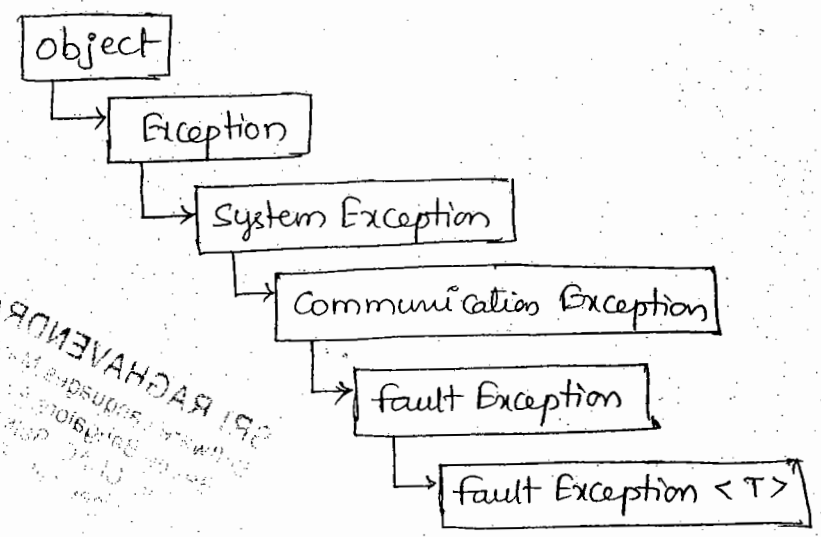
http://localhost:5599/MathService1/.

Client Application :- (Console)

```
using ConsoleApplication2.ServiceReference1;
using System.ServiceModel;

class Program
{
  static void Main()
  {
    MathService1Client proxy = new MathService1Client();
    try
    {
      int n= proxy.Division(10,0);
      Console.WriteLine("Result:"+n);
    }
    catch (FaultException<MathServiceFault> exp)
    {
      Console.WriteLine("Exception occured in Service");
      Console.WriteLine("Message:" + exp.Detail.Message);
      Console.WriteLine("Reason:" + exp.Detail.Reason);
      Console.Writeline("operation:"+ exp.Detail.Operation);
    }
    Console.ReadLine();
  }
}.
```

⇒ Exception class hierarchy :-



26/11/15

⇒ Instance Management in WCF :-

→ Instance Management in WCF means how to organize Service class object in service application.

→ This concept is also called as Service state Management.

→ WCF framework provides multiple options to organize service object creation process based on the requirement.

→ These options are called Instance modes.

→ There are three instance modes in WCF.

1) Per-Call instance mode.
2) Per-Session instance mode.
3) Singleton instance mode.

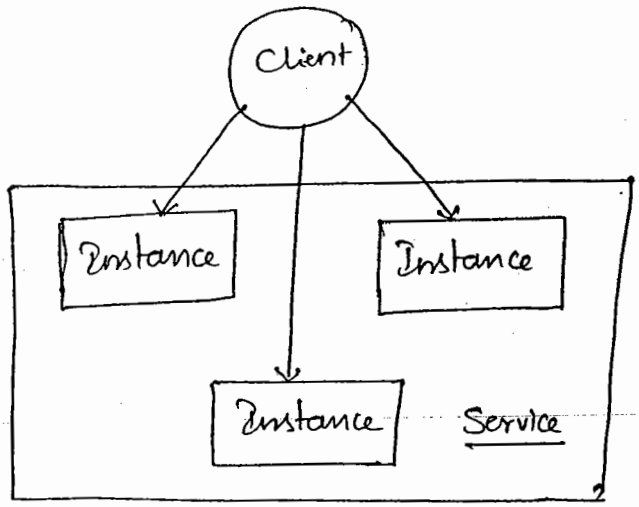
1) Per-Call instance mode :-

→ If you use this option, WCF will create seperate Service class object for every call (request).

→ If a client application invoke a method 3 times, WCF generate 3 service class objects in service application.

→ This option will create new object for every request, process the request, destroy the object after sending response.

Per-call instance Mode.



Note:- Service instances will be disposed after response is sent back to client.

2) Per-Session instance Mode:-

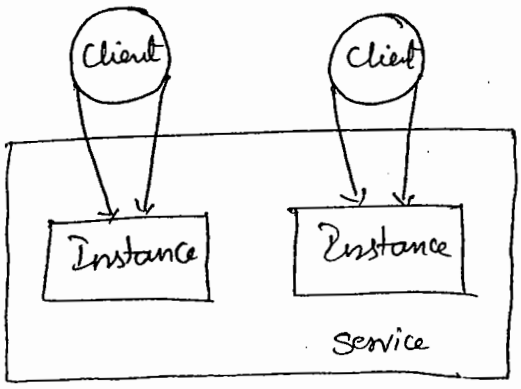
→ If you use this option, wcf will create seperate service class object for every client application.

→ If a client application invoke a method 3 times, it generate one object to address.

→ If 2 clients are communicating, two objects will be prepared.

→ This option will create new object for every client & destroy that object after client application closes.

Per-Session instance mode



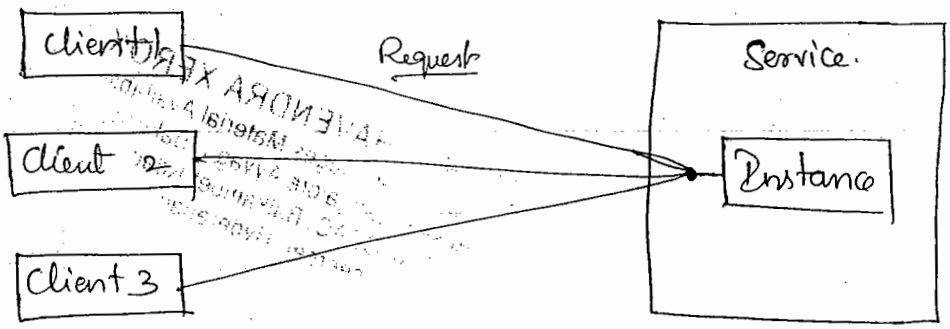
3) Singleton instance mode :-

→ If you use this option, WCF will create only one service object for all clients and all requests.

→ It is implemented irrespective of no.of clients & no.of requests.

→ In this option, service object will be destroyed at the time of closing the service.

Singleton Instance Mode



Implementation :-

```
[ServiceBehaviour(InstanceContextMode = InstanceContextMode.PerCall)]
public class DemoService : IDemoService
{
    //
}
```


Note :- InstanceContextMode Enumeration provides the following options:
percall, PerSession, Single.

→ By default WCF Service instance modes are organized by using "PerSession instance mode".

* Create WCF Service to demonstrate usage of Instance Modes of Service class Object

* Service Application *

1) IDemoService.cs

```
namespace ConsoleApplication1
{
  [ServiceContract]
  public interface IDemoService.
  {
    [OperationContract]
    int GetCount();
    [OperationContract]
    void UpdateCount(int n);
  }
}
```


2) DemoService.cs

```
namespace ConsoleApplication1
{
  [ServiceBehaviour(InstanceContextMode=InstanceContextMode.Single)]
  public class DemoService : IDemoService
  {
    private int count;
    public DemoService()
    {
      count = 0;
      Console.WriteLine("Service Instance is Created");
    }
    public int GetCount()
    {
      return count;
    }
    public void UpdateCount(int n)
    {
      count = count+n;
    }
  }
}
```

3) Program.cs

```
using System.ServiceModel;

class Program
{
    static void Main()
    {
        ServiceHost host = new ServiceHost(typeof(DemoService));
        host.Open();
        Console.WriteLine("Service is started. Press any key to stop service");
        Console.ReadLine();
        host.Close();
    }
}
```

4) update the Base Address as follows in app.config

```
http://localhost:5511/DemoService/
```

5) Run the Program.

→ Client Application ←

6) Program.cs

```
using ConsoleApplication2.ServiceReference1;
class Program
{
    static void Main()
    {
        DemoServiceClient proxy = new DemoServiceClient();
        proxy.UpdateCount(5);
        Console.WriteLine("Count:" + proxy.GetCount());
        proxy.updateCount(10);
        Console.WriteLine("Count:" + proxy.GetCount());
        Console.ReadLine();
    }
}
```

Output:-

1) Per-Call:-

Execution-1

Client 1
count : 0
count : 0
count : 0

Execution-2

Client-2
count : 0
count : 0
count : 0

2) PerSession:-

Client1
Count : 5
Count : 15
Count : 30

Client2
count : 5
count : 15
count : 30

3) Singleton:-

Client1
count : 5
count : 15
count : 30

Client 2
count : 35
count : 45
count : 60

27/4/15

Consuming WCF Operations Asynchronously.

→ In general, the operations that we invoke from client application will execute in synchronously.

→ In synchronous process, client application need to wait for the response.

→ Client application cannot do any other operations until it will get the response from Service.

→ WCF framework provides Asynchronous communicate with service. so, that client application no need to wait for the response.

→ Client application calls the operation, after some time response handling code will execute once response ready (operation completed).

→ To implement this Asynchronous calling we need to prepare corresponding code in consumer Application.

→ Programming items related to Asynchronously calling in client application:

1) MethodNameAsync() – method

   Ex:- GetDataAsync()

2) MethodNameCompleted – Event

   Ex:- GetDataCompleted.

3) MethodNameCompletedEventArgs – class.

   Ex:- GetDataCompletedEventArgs.

Note:-

→ The Above programming items will be auto-generated by Visual Studio in client Application.

→ GetEmpDetailsCompletedEventArgs provides a property called "Result". This property datatype is depends on the return type of WCF operation.

* create WCF Service application to implement Asynchronous Communication,

* Service Application

1) IDemoService.cs

```
namespace ConsoleApplication1
{
    [ServiceContract]
    public interface IDemoService
    {
        [OperationContract]
        int GetEmpCount(int dno);
    }
}
```

2) DemoService.cs

```
using System.Data.SqlClient;

public class DemoService : IDemoService
{
    public int GetEmpCount(int dno)
    {
        int Count = 0;
        SqlConnection conn = new SqlConnection("Server=.; Database =
                                    Sathy Db; Trusted_Connection = true;");
        SqlCommand cmd = new SqlCommand("Select count(*) from emp
                                    where deptno = " + dno, conn);
        conn.open();
        Count = (int) cmd.ExecuteScalar();
        conn.close();
        System.Threading.Thread.Sleep(5000);
        return count;
    }
}
```

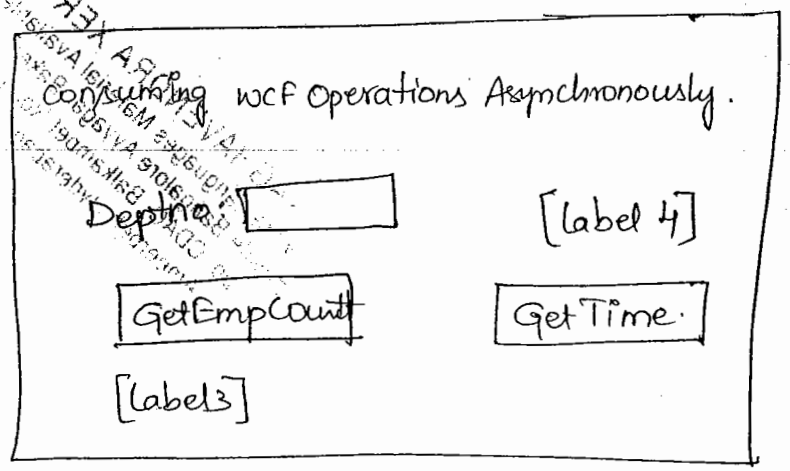
3) Program.cs

// Same as Above program.

* Client Application (windows)

1) Create windows Application to consume the above Service.
2) Add the above service base address as Service reference.
   → At the time of adding Service reference, enable asynchronous calling.
   → Click "Advanced" button.
   → Check "Generate Asynchronous Operations" CheckBox.
   → Click OK.

3) form1.cs (design)



4) form1.cs (code)

```
using System.Windows.forms;
using System.WindowsApplication1.ServiceReference1;

public partial class form1 : form
{
   DemoServiceClient proxy=new DemoServiceClient();
```

```
private void button2_Click()
{
  label4.Text = "DateTime.Now.ToString("T");
}
private void button1_Click()
{
  proxy.GetEmpCountCompleted += new EventHandler<GetEmpCountCompleted
                                EventHandler>(ResultHandle);
  proxy.GetEmpCountAsync(Int.Parse(textBox1.Text));
}
public void ResultHandle(object Sender, GetEmpCountCompletedEventArgs e)
{
  label3.Text = "EmpCount:" + e.Result;
}
}
```

28/11/15

WCF Self-Hosting Using Windows Services.

→ Self-Hosting can be implemented in different ways.
→ It may be using Console, Windows (or) Windows Service.
→ If you use Windows Service to prepare Self-Hosting, your wcf will start when OS is loaded (automatic)
→ Due to this, your wcf Service is always in running mode. No need to start the service manually.

Implementation :-

1) Create wcf Service interface, Service class.
2) Create Windows Service to host the wcf Service.
3) Install Windows Service on the OS.
4) Create Consumer project to use wcf Service.

Example :-

* Create Windows Service Application to implement Self-Hosting.

1) Create New Windows Service Application.
2) It generate Service1.cs file to prepare Windows Service.
3) Add Installers to Service1.cs
4) Set the properties for the components in "ProjectInstaller.cs".

ServiceProcessInstaller1

Account = LocalSystem.

ServiceInstaller1

DisplayName = "wcf Self-Hosting Service".
Description : "This Service will start the wcf Service so, that clients can consume service operations".

5) Add "WCF Service" to project & prepare required files:

Service Application

1) IDemoService.cs

```
namespace WindowsService1
{
  [ServiceContract]
  public interface IDemoService
  {
    [OperationContract]
    string AddDept(Dept obj);
  }
  public class Dept
  {
    [DataMembers]
    public int Deptno {get; set;}
    [DataMembers]
    public string Dname {get; set;}
    [DataMembers]
    public string Loc {get; set;}
  }
}
```

2) DemoService.cs

```
using System.IO;
using System.Data.SqlClient;
public class DemoService : IDemoService
{
  public string AddDept(Dept obj)
  {
```

```
SqlConnection conn = new SqlConnection("Server=.;database="Sathya Db";
                                Trusted-Connection=true;");
SqlCommand cmd = new SqlCommand("Insert into Dept Values
                        ("+obj.Deptno+", '"+Dbj.Dname+"', '"+obj.Loc"')",
                                                        conn);
conn.open();
cmd.ExecuteNonQuery();
Conn.close();
cmd.Dispose();
string str = "Dept details are added in DB";
return str;
}
}
```


3) update the Base Address in App.config as follows.

```
http://localhost:6611/DemoService/
```

4) write the Hosting code in Windows Service file;

Service1.cs.

5) Service1.cs

```
using System.ServiceModel;
using System.IO;
public partial class Service1 : ServiceBase
{
  public Service1()
  {
    InitializeComponent();
  }
  ServiceHost host = null;
```

```
protected override void OnStart(string[] args)
{
    if(host != null)
    {
        host.Close();
    }
    host = new ServiceHost(typeOf(DemoService));
    host.Open();
    StreamWriter sw = new StreamWriter("D:\\WcfServiceLog.txt", true);
    sw.WriteLine("Wcf Service is Started at :" + DateTime.Now.ToString());
    sw.Close();
}
protected override void OnStop()
{
    if(host != null)
    {
        host.Close();
        host = null;
        StreamWriter sw = new StreamWriter("D:\\WcfServiceLog.txt", true);
        sw.WriteLine("Wcf Service is stopped at :" + DateTime.Now.ToString());
        sw.WriteLine("----------");
        sw.Close();
    }
}
```


6) Build the Project

Install the windows service in OS.

1) Open the VS command prompt in Admin mode.

2) Type the following command at project/service bin/Debug folder path.

installutil.exe -i WindowsService1.exe.

3) Goto Services and start your Windows Service.

ControlPanel → Administrative Tools → Services.

Service Name : WCF SelfHosting Service.

Client Application

Create Console Application to consume the service.

→ Add ServiceReference by using BaseAddress.

Program.cs

```
using ConsoleApp.ServiceReference1;
namespace ConsoleApp
{
  class Program
  {
    void Main()
    {
      DemoServiceClient proxy = new DemoServiceClient();
      Dept obj = new Dept();
      obj.Deptno = 70;
      obj.Dname = "Testing";
      obj.Loc = "Hyd";
      string Status = proxy.AddDept(obj);
      Console.WriteLine(status);
      Console.ReadLine();
    }
  }
}
```

30/11/15

Create WCF Service to Perform CRUD Operation on EmpData

Hint: use Entity framework to perform DB operations.

1) Create Console Application to develop the webService.

2) Add Emp table to your project by using Entity framework.
   → Right click on project name → Add New item.
   → Visual C# → Data → Select "ADO.NET Entity Model".
   → Click "Add" Button.
   → follow the wizard to provide required details.
      – Connection details.
      – table details
   → Click "finish" Button
   → Build the Project.

3) Add WCF Service to your project.
   Name : EmpService.cs.

1) IEmpService.cs :-

```
[ServiceContract]
public interface IEmpService.
{
  [OperationContract]
  List<Emp> GetEmps();
  [OperationContract]
  Emp GetEmp(int eno);
```

```
    [OperationContract]
    Void AddEmp(Emp obj);
    [OperationContract]
    Void DeleteEmp(int eno);
}
```

2) EmpService.cs

```
public class EmpService : IEmpService.
{
    SathyaDBEntities db = new SathyaDbEntities();
    public List<Emp> GetEmps()
    {
        List<Emp> emplist = db.Emps.ToList();
        return emplist;
    }
    public Emp GetEmp(int eno)
    {
        Emp obj = db.Emps.SingleOrDefault(x => x.Empno == eno);
        return obj;
    }
    public void AddEmp(Emp obj)
    {
        db.Emps.AddObject(obj);
        db.SaveChanges();
    }
    public void DeleteEmp(int eno)
    {
        Emp obj = db.Emps.SingleOrDefault(x => x.Empno == eno);
        db.Emps.DeleteObject(obj);
        db.SaveChanges();
    }
}
```

3) Program.cs

// Same as previous Example.

Note: update the class Name as `typeof(EmpService)`

4) update the BaseAddress in App.config as follows.

`http://localhost:7766/EmpService/`

5) Run the project.

Client Application

→ Create the ASP.NET mvc Application to consume the Above EmpService.

1) Create new ASP.NET mvc project and add the service Reference
   → Build the project.

2) Add HomeController and define required action methods.

Home.Controller.cs

```
using mvcAppl.ServiceReference1;
namespace mvcAppl.Controllers
{
  public class HomeController : Controller
  {
    EmpServiceClient proxy = new EmpServiceClient();
    public ActionResult Index()
    {
      Emp[] emplist = proxy.GetEmps();
      return View(emplist);
    }
```

```
public ActionResult Details(int id)
{
    Emp obj = proxy.GetEmp(id);
    return View(obj);
}
public ActionResult Greate()
{
    return View();
}
[HttpPost]
public ActionResult Create(Emp obj)
{
    proxy.AddEmp(obj);
    return RedirectToAction("Index");
}
public ActionResult Delete(int id)
{
    Emp obj = proxy.GetEmp(id);
    return View(obj);
}
[HttpPost]
public ActionResult Delete(string id)
{
    int n = int.Parse(id);
    proxy.DeleteEmp(n);
    return RedirectToAction("Index");
}
}
```


3) Add views for above application action methods by using Scaffolding Templetes.

4) Run and Test.

Note:- In the above Service project "Emp" class is autogenerated by EF.

→ This "Emp" class is Serializable, so no need to add additional attributes -

→ In ClientApplication, List collection is generated as Array.
If you want to get List format, update those settings in "Configure Service Reference" window (Advanced Button).

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1/12/15

Transactions in WCF

→ By default WCF service operations are executing individually.
→ Every call from proxy will execute seperately
→ If you want to make multiple calls as single transactions, you have to implement transaction concept in WCF.

Implementation:-

1) Provide required attribute at operation contract level.

```
[TransactionFlow(TransactionFlowOption.Allowed)]
```

2) Enable Transaction at Service class method level:

```
[OperationBehaviour(TransactionScopeRequired = true)]
```

3) Enable Transactionflow option in config file for your end-point.

4) [Client App]
→ use TransactionScope class object to commit or roleback in client application.

* Create WCF Service to implement transaction concept.

Service Application

1) IDemoService1.cs

```
namespace ConsoleApplication1
{
    [ServiceContract]
    public Interface IDemoService
    {
        [OperationContract]
        [TransactionFlow(TransactionFlowOption.Allowed)]
        Void AddDept(int dno, string dname, string Loc);
    }
}
```

2) DemoService1.cs

```
using System.Data.SqlClient;

public class DemoService : IDemoService
{
 [OperationBehaviour (TransactionScopeRequired = True)]
  public void AddDept (int dno, string dname, string Loc)
  {
    SqlConnection conn = new SqlConnection ("Server = .; database = SathyaDb;
                                  Trusted_connection = true; ");
    SqlCommand cmd = new SqlCommand ("Insert into Dept Values
                                  (@dno, @dname, @loc)", conn);
    Cmd.Parameters.AddWithValue ("@dno", dno);
    Cmd.Parameters.AddWithValue ("@dname", dname);
    Cmd.Parameters.AddWithValue ("@loc", Loc);
    conn.open();
    Cmd.ExecuteNonQuery();
    conn.close();
  }
}
```


3) program.cs

```
// Same as Above.
```

4) Add the following tag in Config file:

```
<System.ServiceModel>
  <bindings>
    <wsHttpBinding>
      <binding name = "TransactionBinding" transactionFlow = "true" />
    </wsHttpBinding>
  </bindings>

</System.ServiceModel>
```

→ update the above binding name in wshttp endpoint as follows

```
<endpoint
    binding = "wsHttpBinding"
    bindingConfiguration = "TransactionBinding" >
```

→ update the base Address as follows:

```
http://localhost:6644/DemoService
```


Client Application

1) Program.cs :-

```
using System.ServiceModel;
using System.Transactions;
using ~~System~~.ConsoleApplication2.ServiceReference1;

class Program
{
  using (TransactionScope ts = new TransactionScope(TransactionScopeOption.
                                                     RequiresNew))
  {
    try
    {
      DemoServiceClient proxy = new DemoServiceClient();
      proxy.AddDept(30, "Operations", "Hyd");
      proxy.AddDept(30, "IT", "Hyd");
      ts.Complete(); // Commit
      Console.WriteLine("Dept details are added to Database");
      Console.ReadLine();
      proxy.Close();
    }
    catch (Exception exp)
    {
      ts.Dispose(); //rollback.
      Console.WriteLine("Transaction is failed, Rollback the operations");
    }
  }
  Console.ReadLine();
}
```

2/12/15

Interview Questions

1) What is WCF?

→ WCF is a framework used to developing, configure and deploying services which can be accessible across multiple networks

2) What are the Advantages of WCF?

3) Why do we need to host the Service?

4) What are the various ways of hosting a WCF Service?

1) IIS (or) WebHosting.
2) SelfHosting
3) WAS (Windows Process Activation Service)

5) What are the advantages of Hosting WCF in IIS?

→ Automatically IIS will be available to users, when OS is starting.
→ IIS will take care all execution Environments issues.

6) What is "Automatic Activation" in WCF?

→ When request will come objects are created automatically.
→ AutomAtic Activation means service is not necessary running in advance.
→ In WebHosting Service will be started automatically based on the request.
→ In SelfHosting we should start the service in advance.
→ WebHosting supports Automatic Activation process. i.e., activating the Service based on request.

7) What are the advantages of Hosting wcf in WAS?

→ WAS Hosting supports all types of Protocols (Http, tcp)

→ WAS server will start automatically along with Operating System.

→ WAS Hosting contains advantages of SelfHosting & IIS Hosting.

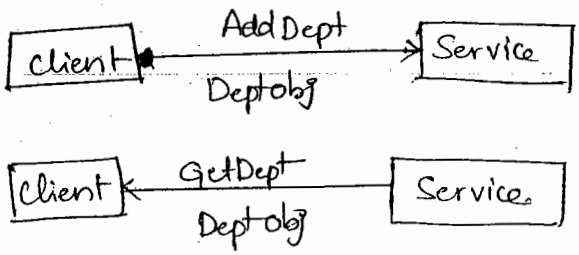
8) How to implement methods overloading in WCF Service?

```
[OperationContract (Name = "AddIntegers")]
int Add (int a, int b)

[OperationContract (Name = "AddDoubles")]
double Add (double a, double b)
```

9) What are the purpose of "DataContract" and "MemberContract" attributes?

→ These attributes are applying for the class i.e., used to exchanging the data b/w Service & client



10) What is the use of "IsRequired" property in DataContracts?

```
[DataMember(IsRequired = true)]
public string CustomerId
```

11) what is the need of Serialization in WCF?

→ Serialization is a process of convert the objects from one format to another format so that we send across networks.

→ In WCF, in order to exchange the data b/w client and Service, Serialization is required.

→ for TCP protocol – Binary Serialization.

→ for Http Protocol – SOAP Serialization.

12) which namespace is involved in wcf to perform Serialization and De-Serialization?

```
using System.Runtime.Serialization;
```

13) what is the proxy for wcf service? Is Proxy will execute the

14) operations completely?

→ class that we prepared in client Application to communicate with service.

→ No, Proxy does not execute the operations, it will redirect the request to service.

15) what is MEP?

16) what are the MEP's supported by WCF?

→ MEP stands for Message Exchange Pattern.

→ MEP will specify. how the message will be exchange between client & service.

→ wcf supports the following MEP's:

1) Request – Response(Request – Reply)
2) OneWay.
3) Duplex

Note:- Default MEP is "Request – Response".

17) what was the code name for WCF?

- Indigo

18) why do we use [ServiceBehaviour] attribute in WCF?

- It is used for Service class

→ Service attribute is specify contextmode of instance.

19) How many types of instance modes supported by WCF?

3 types.

1) Percall
2) PerSession
3) Singleton.

20) which mode is suitable to maintain common operations for all users?

- Singleton.

Note:- Basic Http Binding doesnot support perSession.

21) what are the technologies unites by WCF?

1) NET remoting.
2) MSMQ
3) WebServices.
4) com+

22) In WCF, the type of contract used to define a class (or) an Interface.

ServiceContract.

23) In WCF, the type of contract used to define the method of a Service is?

operation Contract

24) what is the WCF contract to handle the errors in WCF Service?

fault Contract.

25) In wcf, the type of contract used to define the custome. datatypes passing on a service is?

- DataContract

26) In wcf, the default type of binding is?

- wsHttp Binding.

27) In wcf, which ~~machine~~ is binding is used for cross machine. communication (or) wcf to wcf communication?

- NETTCP Binding.

28) what is the difference b/w standard void operation & a one-way. operation?

- Standard void operation client will have to wait for response. until complete the task.
- In one-way. client will not wait for Response.

29) Difference b/w wcf & Web Services?

30) what are the options of transaction flow attributes.

TransactionFlowOption - NotAllowed
TransactionFlowOption - Allowed
TransactionFlowOption. Mandatory.