**CS 590 – PARALLEL AND DISTRIBUTED COMPUTING**

**HOMEWORK 6**

**Name:** Ivan Sangines Escrig

**ID#:** 968606

**Instructor:** Dr. Mahmood

**Date:** October 21, 2018

**TABLE OF CONTENT**

[**INTRODUCTION**](#_Toc446970371) 3

[**SCREENSHOTS:**](#_Toc446970373) 4

[**SOURCE CODE:**](#_Toc446970374) 10

[**CONCLUSION:**](#_Toc446970375) 30

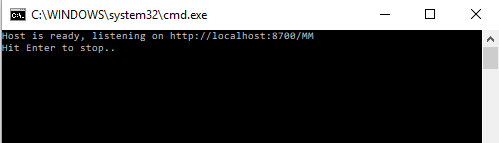
**INTRODUCTION**

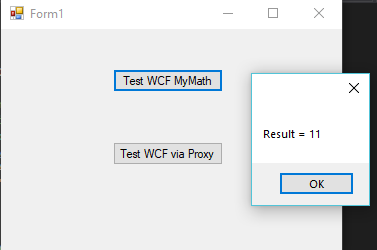
After doing this assignment, I will get familiar with creating Web Services. The objective is to learn how to create, connect and use properly Service Libraries, Hosts and Web Clients. Also, by doing the second part of this assignment I will learn how to use web Services in order to use callbacks and exchange data. This will be achieved by trying all the examples in the hand outs.

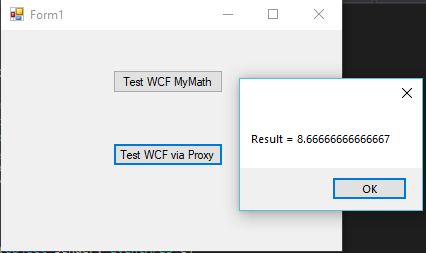
**SCREEN SHOTS:**

**Example 1:**

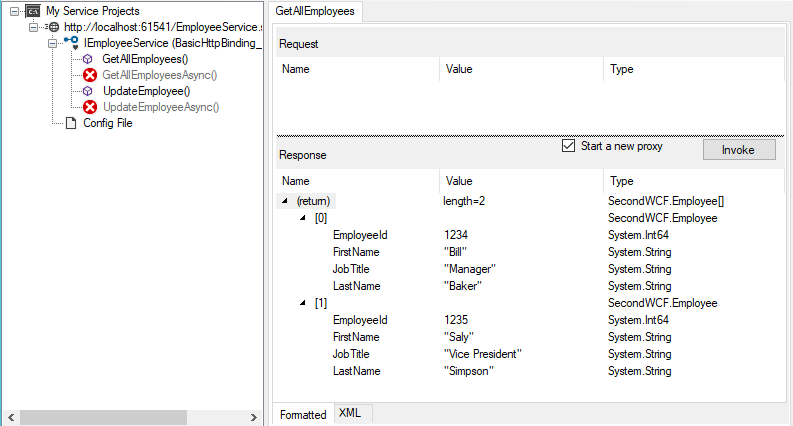
Here we have the Host running and then the outputs for the operations. The first output is used by using a Channel Factory and the second one by using the Proxy.

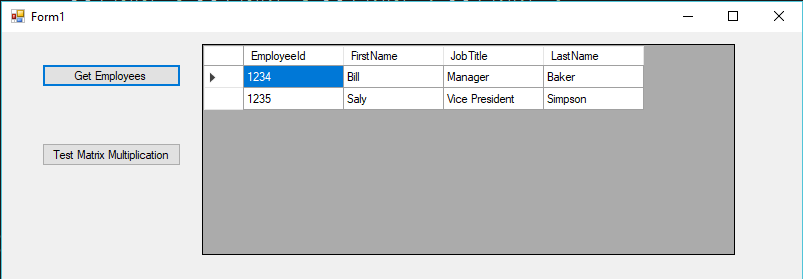
****

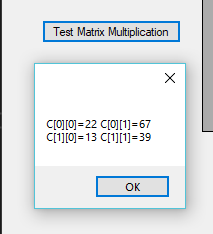




**Example 2:**

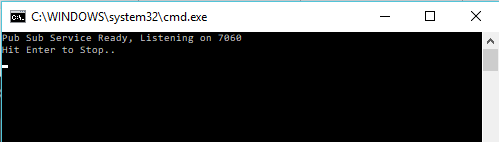


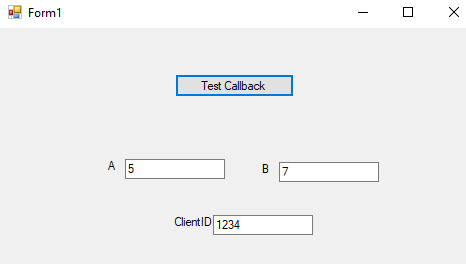
****

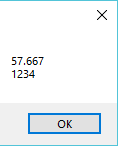
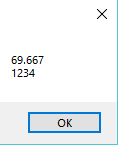
****

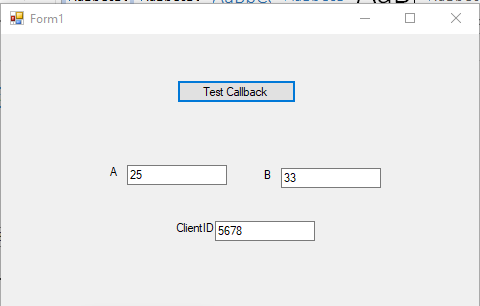
**Example 3:**

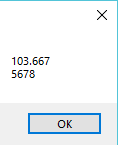
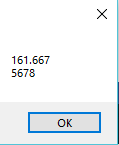
Here we can see the Host running and two Client Forms running at the same time. We can see how the clients operations are done separately since in both outputs we can see which client is responsible for each operation.

****

****

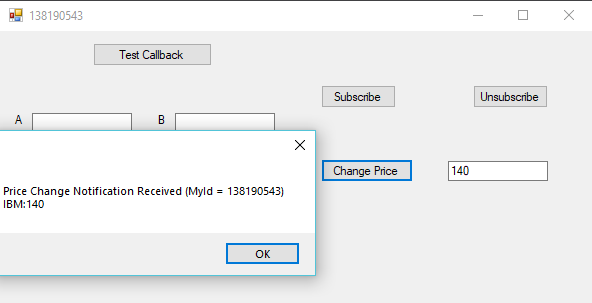
** **

****

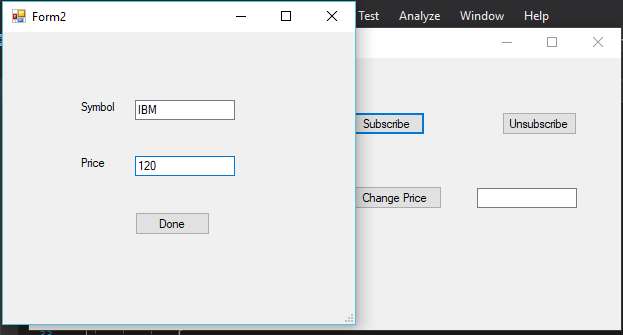
** **

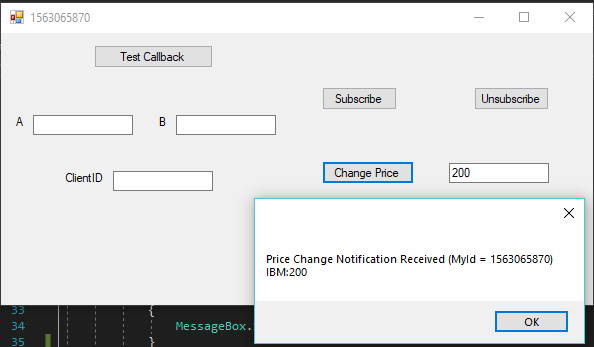
**Example 4:**

Here we can see the first part of this assignment where we first need to subscribe an item (IBM) clicking the button subscribe, this is done automatically by just pressing subscribe. Once we have the item subscribed, we can change the price, if we change the price above 120, a message like the one on the screenshot will show up with the id of the “person” who changed the price and the new price. We can unsubscribe the item IBM by just closing the form since I have indicated that on closing form, execute the method unsubscribe. However, we can also unsubscribe the IBM item but clicking to unsubscribe.

****

Same exercise but using a Form to introduce the Symbol and Price to subscribe.





**SOURCE CODE:**

**EXAMPLE 1:**

**FirstWcfService:**

IMyMath:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace FirstWcfService

{

[ServiceContract]

public interface IMyMath

{

[OperationContract]

double ComputeAvg(int a, int b, int c);

[OperationContract]

int FindMin(int a, int b, int c);

}

}

My Math:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace FirstWcfService

{

public class MyMath : IMyMath

{

public double ComputeAvg(int a, int b, int c)

{

return (a + b + c) / 3.0;

}

public int FindMin(int a, int b, int c)

{

int min = a;

if (b < min)

min = b;

if (c < min)

min = c;

return min;

}

}

}

**NewWCFHost:**

Program.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace NewWCFHost

{

class Program

{

static void Main(string[] args)

{

// Host always hosts the main class for the service

// Host needs to expose end points so that a client can communicate

// to it. Each end point is a unique combination of A B C

// A = address, B = binding i.e., comm. Prot., C = Contract i.e. interface

ServiceHost sh = new ServiceHost(typeof(FirstWcfService.MyMath));

// rest of the configuration for exposing end points will come

// from WCF Configuration Editor

sh.Open(); // host is ready

Console.WriteLine("Host is ready, listening on " +

sh.Description.Endpoints[0].Address.ToString());

Console.WriteLine("Hit Enter to stop..");

Console.ReadLine();

sh.Close();

}

}

}

**FirstWCFClient:**

using FirstWCFClient.FRWCF;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.ServiceModel;

using System.ServiceModel.Channels;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FirstWCFClient

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnTestWCFMyMath\_Click(object sender, EventArgs e)

{

// We can use ChannelFactory to communicate with the Service

// We do need to know the interface

IMyMath im = new ChannelFactory<IMyMath>(new BasicHttpBinding(),

new EndpointAddress("http://localhost:8700/MM")).CreateChannel();

double result = im.ComputeAvg(7, 12, 14);

MessageBox.Show("Result = " + result.ToString());

((IChannel)im).Close();

}

private void btnWCFProxy\_Click(object sender, EventArgs e)

{

// For each endpoint, the proxy creates a class which starts

// with name of the interface (without I) followed by Client

FRWCF.MyMathClient mmc = new FRWCF.MyMathClient();

double res = mmc.ComputeAvg(5, 8, 13);

MessageBox.Show("Result = " + res.ToString());

}

}

}

**EXAMPLE 2:**

**SecondWCF:**

Employee.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Web;

namespace SecondWCF

{

[DataContract]

public class Employee

{

[DataMember]

public long EmployeeId { get; set; }

[DataMember]

public string FirstName { get; set; }

[DataMember]

public string LastName { get; set; }

[DataMember]

public string JobTitle { get; set; }

}

}

EmployeeService.svc.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace SecondWCF

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "EmployeeService" in code, svc and config file together.

// NOTE: In order to launch WCF Test Client for testing this service, please select EmployeeService.svc or EmployeeService.svc.cs at the Solution Explorer and start debugging.

public class EmployeeService : IEmployeeService

{

public List<Employee> GetAllEmployees()

{

List<Employee> EList = new List<Employee>();

Employee e1 = new Employee

{

EmployeeId = 1234,

FirstName = "Bill",

JobTitle = "Manager",

LastName = "Baker"

};

EList.Add(e1);

Employee e2 = new Employee

{

EmployeeId = 1235,

FirstName = "Saly",

JobTitle = "Vice President",

LastName = "Simpson"

};

EList.Add(e2);

return EList;

}

public bool UpdateEmployee(Employee emp)

{

// code to update employee in database

return true;

}

}

}

IEmployeeService:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace SecondWCF

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IEmployeeService" in both code and config file together.

[ServiceContract]

public interface IEmployeeService

{

[OperationContract]

List<Employee> GetAllEmployees();

[OperationContract]

bool UpdateEmployee(Employee emp);

}

}

IMath2:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace SecondWCF

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface name "IMyMath2" in both code and config file together.

[ServiceContract]

public interface IMyMath2

{

[OperationContract]

Matrix MultiplyMatrix(Matrix A, Matrix B);

[OperationContract]

Matrix InitMatrix(int rows, int cols);

}

}

Matrix.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Web;

namespace SecondWCF

{

[DataContract]

public class Matrix

{

[DataMember]

public int Rows { get; set; }

[DataMember]

public int Cols { get; set; }

[DataMember]

public double[][] Data = null; // jagged array

// [,] 2-d array cannot be exposed as data in a SOAP environment

public Matrix(int rows, int cols)

{

this.Rows = rows;

this.Cols = cols;

Data = new double[rows][]; // create array of arrays like C++

for (int i = 0; i < rows; i++)

Data[i] = new double[cols];

}

public double this[int i, int j] // indexer to make it easier to read data

{

get

{

return Data[i][j];

}

set

{

Data[i][j] = value;

}

}

}

}

MyMath2.svc.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

namespace SecondWCF

{

// NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "MyMath2" in code, svc and config file together.

// NOTE: In order to launch WCF Test Client for testing this service, please select MyMath2.svc or MyMath2.svc.cs at the Solution Explorer and start debugging.

public class MyMath2 : IMyMath2

{

public Matrix InitMatrix(int rows, int cols)

{ // constructors are not exposed via service

Matrix mat = new Matrix(rows, cols);

return mat;

}

public Matrix MultiplyMatrix(Matrix A, Matrix B)

{

Matrix C = new Matrix(A.Rows, B.Cols);

for (int i = 0; i < A.Rows; i++)

for (int j = 0; j < B.Cols; j++)

for (int k = 0; k < A.Cols; k++)

C[i, j] = C[i, j] + A[i, k] \* B[k, j];

return C;

}

}

}

**SecondWCFClient:**

Form1.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SecondWCFClient

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnMatrix\_Click(object sender, EventArgs e)

{

MM2.MyMath2Client mmc = new MM2.MyMath2Client(); // proxy

MM2.Matrix A = mmc.InitMatrix(2, 2);

A.Data[0][0] = 5; A.Data[0][1] = 7; A.Data[1][0] = 3; A.Data[1][1] = 4;

MM2.Matrix B = mmc.InitMatrix(2, 2);

B.Data[0][0] = 3; B.Data[0][1] = 5; B.Data[1][0] = 1; B.Data[1][1] = 6;

MM2.Matrix C = mmc.MultiplyMatrix(A, B);

string out1 = "";

for (int i = 0; i < C.Rows; i++)

{

for (int j = 0; j < C.Cols; j++)

{

out1 += "C[" + i.ToString() + "][" + j.ToString() + "]=" +

C.Data[i][j].ToString() + " ";

}

out1 += "\n";

}

MessageBox.Show(out1);

}

private void btnEmployees\_Click(object sender, EventArgs e)

{

ES.EmployeeServiceClient esc = new ES.EmployeeServiceClient();

gv1.DataSource = esc.GetAllEmployees();

gv1.Refresh();

}

}

}

**Example 3 and Example 4:**

**PubSubLib:**

ComputeResult.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace PubSubLib

{

[DataContract]

public class ComputeResult

{

double result;

[DataMember]

public double Result

{

get { return result; }

set { result = value; }

}

DateTime resultTime;

[DataMember]

public DateTime ResultTime

{

get { return resultTime; }

set { resultTime = value; }

}

string clientID;

[DataMember]

public string ClientID

{

get { return clientID; }

set { clientID = value; }

}

}

}

ILongCompute.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace PubSubLib

{

interface IComputeCallback

{

[OperationContract(IsOneWay = true)]

void OnComputeResult(ComputeResult res);

}

[ServiceContract(CallbackContract = typeof(IComputeCallback))]

public interface ILongCompute

{

[OperationContract(IsOneWay = true)]

void Compute(int a, int b, string clientId);

}

}

IPriceChange.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace PubSubLib

{

[ServiceContract]

public interface IPriceChange

{

[OperationContract()]

bool ChangeStockPrice(string symbol, double newprice);

}

}

IStock.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace PubSubLib

{

interface IStockCallback

{

[OperationContract(IsOneWay = true)]

void OnPriceChange(StockInfo sinfo);

}

[ServiceContract(CallbackContract = typeof(IStockCallback))]

public interface IStocks

{

[OperationContract]

bool SubscribeToStockPrice(string stocksym, double

triggerPrice);

[OperationContract]

bool UnSubscribeToStockPrice(string stocksym);

}

}

PubSub.cs:

namespace PubSubLib

{

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession, ConcurrencyMode = ConcurrencyMode.Multiple)]

public class PubSub : ILongCompute

{

ComputeResult cr = new ComputeResult();

Object olock = new object();

#region ILongCompute Members

public void Compute(int a, int b, string clientId)

{

Thread.Sleep(5000);// asssume 5 secs to produce result

lock (olock)

{

cr.Result = 45.667 + a + b;

cr.ResultTime = DateTime.Now;

cr.ClientID = clientId;

// trigger callback in client

IComputeCallback callbackChannel =

OperationContext.Current.GetCallbackChannel<IComputeCallback>();

if (((ICommunicationObject)callbackChannel).State ==

CommunicationState.Opened)

callbackChannel.OnComputeResult(cr);

}

Thread.Sleep(5000); // 5 more secs to produce updated res

lock (olock)

{

cr.Result = cr.Result + a + b;

cr.ResultTime = DateTime.Now;

cr.ClientID = clientId;

//trigger callback in client

IComputeCallback callbackChannel =

OperationContext.Current.GetCallbackChannel<IComputeCallback>();

if (((ICommunicationObject)callbackChannel).State ==

CommunicationState.Opened)

callbackChannel.OnComputeResult(cr);

}

}

#endregion

}

}

PubSub.cs: (Second Part)

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace PubSubLib

{

//IMPLEMENTING ISTOCK AND IPRICECHANGE

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single, ConcurrencyMode = ConcurrencyMode.Multiple)]

public class PubSub : ILongCompute, IStocks, IPriceChange

{

static List<IStockCallback> SubscriberList = new List<IStockCallback>();

ComputeResult cr = new ComputeResult();

Object olock = new object();

#region ILongCompute Members

public void Compute(int a, int b, string clientId)

{

Thread.Sleep(5000);

lock (olock)

{

cr.Result = 45.667 + a + b;

cr.ResultTime = DateTime.Now;

cr.ClientID = clientId;

// trigger callback in client

IComputeCallback callbackChannel = OperationContext.Current.GetCallbackChannel<IComputeCallback>();

if (((ICommunicationObject)callbackChannel).State == CommunicationState.Opened)

callbackChannel.OnComputeResult(cr);

}

Thread.Sleep(5000);

lock (olock)

{

cr.Result = cr.Result + a + b;

cr.ResultTime = DateTime.Now;

cr.ClientID = clientId;

//trigger callback in client

IComputeCallback callbackChannel = OperationContext.Current.GetCallbackChannel<IComputeCallback>();

if (((ICommunicationObject)callbackChannel).State == CommunicationState.Opened)

callbackChannel.OnComputeResult(cr);

}

}

#endregion

#region IStocks Members

public bool SubscribeToStockPrice(string stocksym, double triggerPrice)

{

try

{

IStockCallback callbackChannel = OperationContext.Current.GetCallbackChannel<IStockCallback>();

if (SubscriberList.Contains(callbackChannel) == false)

{

SubscriberList.Add(callbackChannel);

}

}

catch (Exception ex)

{

throw new FaultException(ex.Message, new FaultCode("Subscription Error"));

}

return true;

}

public bool UnSubscribeToStockPrice(string stocksym)

{

try

{

IStockCallback callbackChannel = OperationContext.Current.GetCallbackChannel<IStockCallback>();

if (SubscriberList.Contains(callbackChannel) == true)

{

SubscriberList.Remove(callbackChannel);

}

}

catch (Exception ex)

{

throw new FaultException(ex.Message, new FaultCode("UnSubscription Error"));

}

return true;

}

#endregion

#region IPriceChange Members

public bool ChangeStockPrice(string symbol, double newprice)

{

try

{

if ((symbol == "IBM") && (newprice > 120))

{

// trigger call to the subscribers

foreach (IStockCallback icbChannel in SubscriberList)

{

StockInfo si = new StockInfo();

si.Price = newprice;

si.Symbol = symbol;

si.STime = DateTime.Now;

if (((ICommunicationObject)icbChannel).State == CommunicationState.Opened)

icbChannel.OnPriceChange(si);

}

}

}

catch (Exception ex)

{

throw new FaultException(ex.Message, new FaultCode("Change Price Error"));

}

return true;

}

#endregion

}

}

StockInfo.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace PubSubLib

{

[DataContract]

public class StockInfo

{

string symbol;

[DataMember]

public string Symbol

{

get { return symbol; }

set { symbol = value; }

}

double price;

[DataMember]

public double Price

{

get { return price; }

set { price = value; }

}

DateTime sTime;

[DataMember]

public DateTime STime

{

get { return sTime; }

set { sTime = value; }

}

}

}

**PubSubHost:**

Program.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.ServiceModel;

using System.Text;

using System.Threading.Tasks;

namespace PubSubHost

{

class Program

{

static void Main(string[] args)

{

try

{

ServiceHost sh = new

ServiceHost(typeof(PubSubLib.PubSub));

Console.WriteLine("Pub Sub Service Ready, Listening on 7060");

Console.WriteLine("Hit Enter to Stop..");

sh.Open();

Console.ReadLine();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

**PubSubClient:**

CBClient:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PubSubClient

{

class CBClient : PS.ILongComputeCallback, IDisposable

{

PS.LongComputeClient proxy = null;

public void CallLongCompute(int x, int y, string clientID)

{

try

{

proxy = new PS.LongComputeClient(new System.ServiceModel.InstanceContext(this),"PSEP");

proxy.Compute(x, y, clientID);

}

catch (Exception ex)

{

throw ex;

}

}

#region ILongComputeCallback Members

public void OnComputeResult(PS.ComputeResult res)

{

MessageBox.Show(res.Result.ToString() + "\n" + res.ClientID);

}

#endregion

#region IDisposable Members

public void Dispose()

{

proxy.Close();

}

#endregion

}

}

STKClient:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PubSubClient

{

class STKClient : PS.IStocksCallback, IDisposable

{

PS.StocksClient stkProxy = null;

int myId = 0;

public void SubscribeToPriceChange(string sym, double triggerPrice, int id)

{

try

{

myId = id;

stkProxy = new PS.StocksClient(new System.ServiceModel.InstanceContext(this), "SPEP"); //SEP is the endpoint name for IStocks interface ep

stkProxy.SubscribeToStockPrice(sym, triggerPrice);

}

catch (Exception ex)

{

throw ex;

}

}

public void UnSubscribeToPriceChange(string sym)

{

try

{

if (stkProxy != null)

stkProxy.UnSubscribeToStockPrice(sym);

}

catch (Exception ex)

{

throw ex;

}

}

#region IStocksCallback Members

public void OnPriceChange(PS.StockInfo sinfo)

{

MessageBox.Show("Price Change Notification Received (MyId = " + myId.ToString() + ")\n" +

sinfo.Symbol + ":" + sinfo.Price.ToString());

}

#endregion

#region IDisposable Members

public void Dispose()

{

stkProxy.Close();

}

#endregion

}

}

Form1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PubSubClient

{

public partial class Form1 : Form

{

STKClient stkC = new STKClient();

int myId = 0;

public Form1()

{

InitializeComponent();

}

private void btnTestCallback\_Click(object sender, EventArgs e)

{

try

{

CBClient cbc = new CBClient();

cbc.CallLongCompute(int.Parse(txtA.Text),

int.Parse(txtB.Text), txtClientID.Text);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnSubscribe\_Click(object sender, EventArgs e)

{

try

{

//stkC.SubscribeToPriceChange("IBM", 128, myId);

//Using Forms2 Information

Form2 form2 = new Form2();

form2.ShowDialog();

stkC.SubscribeToPriceChange(form2.Symbol, form2.Price, myId);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnChangePrice\_Click(object sender, EventArgs e)

{

PS.PriceChangeClient pcc = new PS.PriceChangeClient();

pcc.ChangeStockPrice("IBM", double.Parse(txtNewPrice.Text));

}

private void Form1\_Load(object sender, EventArgs e)

{

myId = new Random((int)DateTime.Now.Ticks).Next();

this.Text = myId.ToString();

}

private void btnUnsubscribe\_Click(object sender, EventArgs e)

{

try

{

stkC.UnSubscribeToPriceChange("IBM");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

try

{

stkC.UnSubscribeToPriceChange("IBM");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

Form2:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PubSubClient

{

public partial class Form2 : Form

{

public String Symbol { get; set; }

public double Price { get; set; }

public Form2()

{

InitializeComponent();

}

private void btnDone\_Click(object sender, EventArgs e)

{

try

{

if (txtPrice.Text != null && txtSymbol.Text != null )

{

Symbol = txtSymbol.Text;

Price = double.Parse(txtPrice.Text);

this.Close();

}

else

{

MessageBox.Show("No empty fields allowed");

}

}

catch (Exception ex)

{

MessageBox.Show("Invalid Input");

}

}

}

}

**Conclusion:**

Before doing this assignment, I did not have any experience with any of the Web Services functions. After completing this assignment, I have a better idea how everything works in order to make a Web Service work by connecting/creating end points between the different components of the Web Service. Also, I have a better idea how callbacks and sharing data work while using Web Services.