WebServices Testing Essentials 2013

INTRODUCTION

WHAT ARE WEBSERVICES?

- Interface between one program and the other
- XML Xtensible Markup Language
- SOAP (simple object access protocol)
- WSDL (web services description language)
- REST – Representational State Transfer
- UDDI (Universal description discover & integration)
- The above have multiple versions within themselves e.g XML1.0, SOAP 1.2, etc

```
empcode, empname, grade, salary

CSV
101, Raja, PM, 10000
102, James, TL, 8000
103, Reovert, SA, 5000

TABBED
101   Raja  PM    10000
102   James TL    8000
103   Reovert     SA    5000

XML
<employee>
<empcode>101</empcode>
<empname>Raja</emname>
<grade>PM</grade>
<salary>10000</salary>
</employee>

Note:
<container>
      <node></node>
</container>
```

ROLES AND OPERATIONS

3 Major roles

- Service provider – actual web services
- Registry – place where info is stored and published
- Consumer – calls the webservices

3 Major Operations

- Publish – somebody to say this is the webservice
- Find – Moment you access has to find the webservice
- Bind – call the webservices
- A webserver essentially holds the service (TomCat)
    - Deploy to TomCat

- This operates in request-response mode
    - TomCat receives request and responds
- To call a web service, you need to know the methods and arguments

XML: ADVANTAGES

- XML has specific advantage of being independent of data position (the tags can appear anywhere)
- Webservices are language independent – C# webservice can be easily called from a jsp page
- Since they are independent, they act as plug and play components; hence applications can be easily enhanced

SOA

- **Service Oriented Architecture** heavily depends on webservices
- In an online shopping portal, if we make every operation like get product catalog, get price for product, add to cart, payment gateway etc., we will get the ability to call them the way we want
- This gives an open field for many vendors to build their applications on top of the base product

WHAT IS IN THE MESSAGES?

- XML contains tags and data
- SOAP contains an envelope, header and body
- WSDL contains data types, messages and service ports
- UDDI is mainly for universal business registrations
- All these are again using XML format only
- JSON data format (name-value pairs)
    - JavaScript Object Notation

TESTING A WEBSERVICE

- Think that you are testing a function with many arguments
- What differs here is the way to call the webservice
- Call it from a browser
- Pass parameters
- View results in XML format
- Validate the XML format and data
- Simply send data in the URL to the webservice

WEBSERVICE

```
[WebService (Namespace = "http://silosix.com")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService

[WebMethod]
public String AddEmployeeDetails(String strEmpCode, String strEmpname, Decimal iSal){
…
…
}
```

VIEW THE ENTIRE EMPMGMT/SERVICE CLASS

- http://Localhost:1165/EmpMgmt/Service.asmx

Service

The following operations are supported. For a formal definition, please review the Service Description.

- AddEmployeeDetails
- GetAllEmployeeDetails
- UpdateEmployeeDetails

ADDEMPLOYEEDETAILS

- To call the webservice http://Localhost:1165/EmpMgmt/Service.asmx?op=AddEmployeeDetails
- .NET framework provides the form based off the [WebMethod] attribute/tag
- Aervice.aspx = live page
- Service.asmx = denoting the **webservice** (testing form)
- ?op=AddEmployeeDetails = **operation**
- Invoke button to post form data back to the service

AddEmployeeDetails

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
strEmpCode:	
strEmpName:	
iSal:	

Invoke

Parameter	Value
strEmpCode:	501
strEmpName:	Abdul
iSal:	500

Invoke

- Sends SOAP data back to the ASP server

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /EmpMgmt/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/AddEmployeeDetails"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://ww
  <soap:Body>
    <AddEmployeeDetails xmlns="http://tempuri.org/">
      <strEmpCode>string</strEmpCode>
      <strEmpName>string</strEmpName>
```

- ASP server response

Favorites http://localhost:1165/EmpMgmt/Service.asmx/AddEm

```
<?xml version="1.0" encoding="utf-8" ?>
<string xmlns="http://tempuri.org/">Record inserted
  successfully</string>
```

SQL INSERT STATEMENT

SELECT * FROM employeemanagement.emploreemaster;

empid	empcode	empname	salary
1	501	Abdul	500
2	502	Bob	200

UPDATEEMPLOYEEDETAILS

- To call the webservice http://Localhost:1165/EmpMgmt/Service.asmx?op=UpdateEmployeeDetails

UpdateEmployeeDetails

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
strEmpCode:	
iSal:	

Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /EmpMgmt/Service.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/UpdateEmployeeDetails"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www
  <soap:Body>
    <UpdateEmployeeDetails xmlns="http://tempuri.org/">
      <strEmpCode>string</strEmpCode>
      <iSal>decimal</iSal>
    </UpdateEmployeeDetails>
  </soap:Body>
```

SOAP MESSAGE SENT TO ASP SERVER

```
POST /EmpMgmt/Service.asmx HTTP/1.1
Host: localhost
Content-type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: http://tempuri.org/UpdateEmployeeDetails
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsihttp://www.w3.org/2001/XMLSchema-instance
xmlsn:xsd=http://www....>
 <soap:Body>
  <UpdateEmployeeDetails xmlns="http://tempuri.org/>"
   <strEmpCode>string</strEmpCode>
   <iSal>decimal</iSal>
  <UpdateEmployeeDetails>
 </soap:Body>
…
…
```

TIPS FOR WEBSERVICE TESTING

- Verify Webservice Works
- Test boundaries
- Missing Data
- Invalid Data
- Negatives
- Invalid Data types
- No Data
- Too Much Data

SIDE:NOTES – ASP FILE TYPES EXPLAINED

aspx = page in WebForms. Contains controls and events

asmx = web services. Modality of sharing information from/to website

ashx = generic handler. Can be used in various ways* as generate pages, sharing information, display pictures... .asax : Global.asax, used for application-level logic

.ascx : Web UserControls: custom controls to be placed onto web pages.

.ashx : custom HTTP handlers.

.asmx : web service pages. From version 2.0 a Code behind page of an asmx file is placed into the app_code folder.

.axd : when enabled in web.config requesting trace.axd outputs application-level tracing. Also used for the special webresource.axd handler which allows control/component developers to package a component/control complete with images, script, css etc. for deployment in a single file (an 'assembly')

.config : web.config is the only file in a specific Web application to use this extension by default (machine.config similarly affects the entire Web server and all applications on it), however ASP.NET provides facilities to create and consume other config files. These are stored in XML format.

.cs/vb : Code files (cs indicates C#, vb indicates Visual Basic). Code behind files (see above) predominantly have the extension ".aspx.cs" or ".aspx.vb" for the two most common languages. Other code files (often containing common "library" classes) can also exist in the web folders with the cs/vb extension. In ASP.NET 2 these should be placed inside the App_Code folder where they are dynamically compiled and available to the whole application.

.dbml : LINQ to SQL data classes file

.master : 2.0 master page file

.resx : resource files for internationalization and localization. Resource files can be global (e.g. messages) or "local" which means specific for a single aspx or ascx file.

.sitemap : sitemap configuration files. Default file name is web.sitemap

.skin : theme skin files.

.svc : Windows Communication Foundation service file

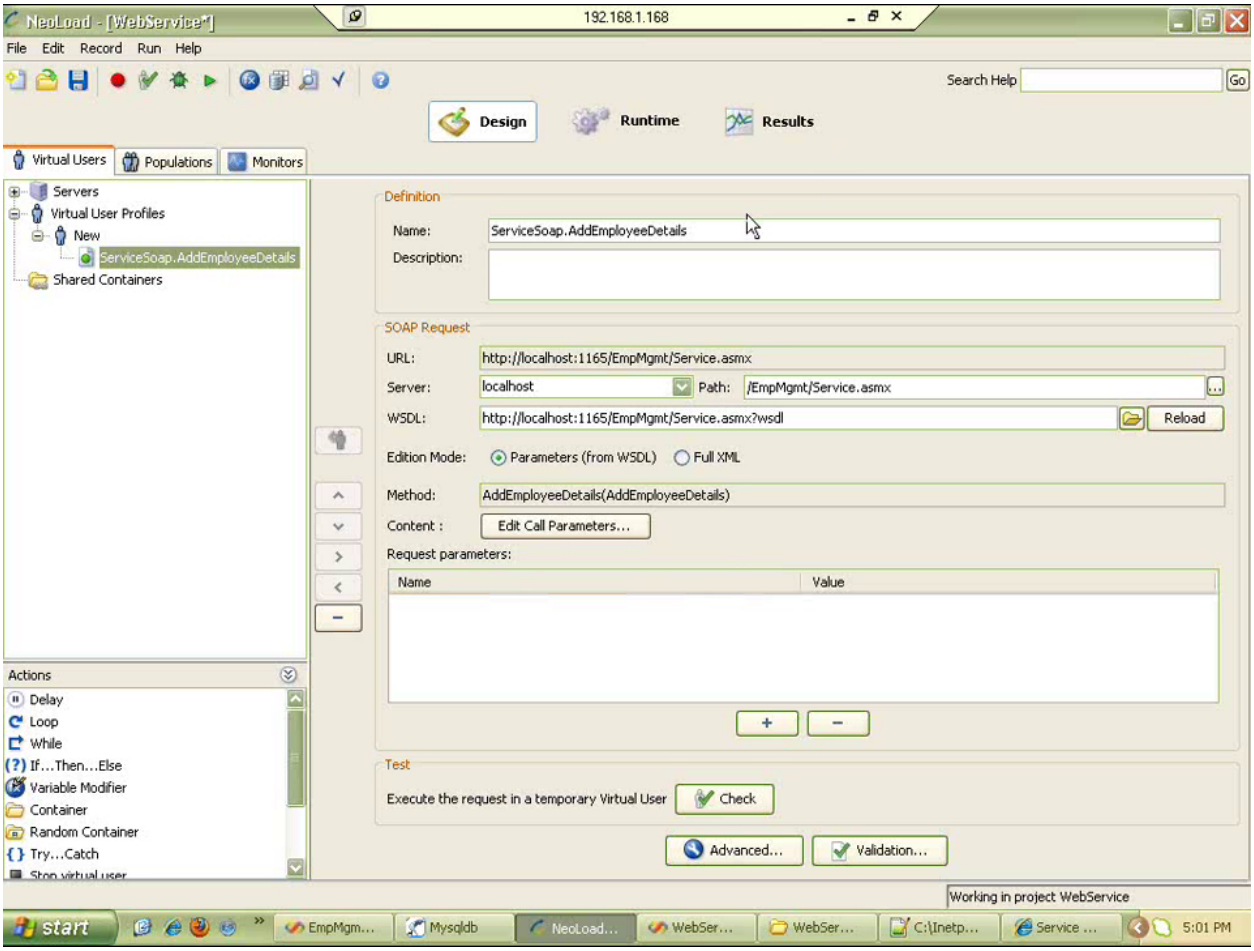
browser : browser capabilities files stored in XML format; introduced in version 2.0. ASP.NET 2 includes many of these by default, to support common web browsers. These specify which browsers have which capabilities, so that ASP.NET 2 can automatically customize and optimize its output accordingly. Special .browser files are available for free download to handle, for instance, the W3C Validator, so that it properly shows standards-compliant pages as being standards-compliant. Replaces the harder-to-use BrowserCaps section that was in machine.config and could be overridden in web.config in ASP.NET 1.x.

SIDE:NOTES – Q&A

LOAD TESTING

- Unit Testing…only one user doing one thing at a time
- Load Testing = concurrent users
- Webservices, being independent and open, must definitely be load tested
- Webservices can be called by thousands of clients simultaneously so will be under stress
- Webservice, per se, is not light; it depends on what logic you had placed in the service
- Many tools also provide ways to test web services
- Neoload can record and run webservice calls
- Softsmith has got its own set of custom scripts when logic becomes complicated

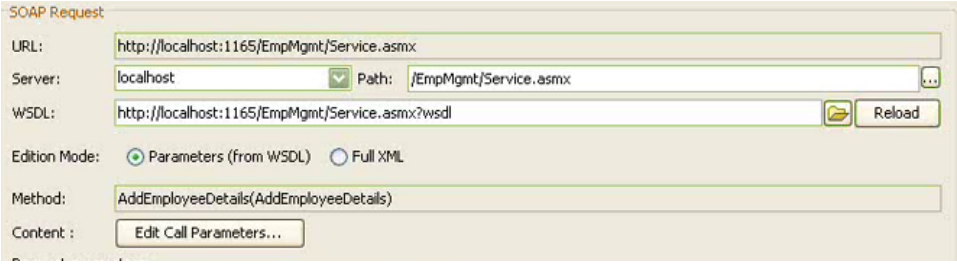
NEOLOAD (LOAD TESTER)

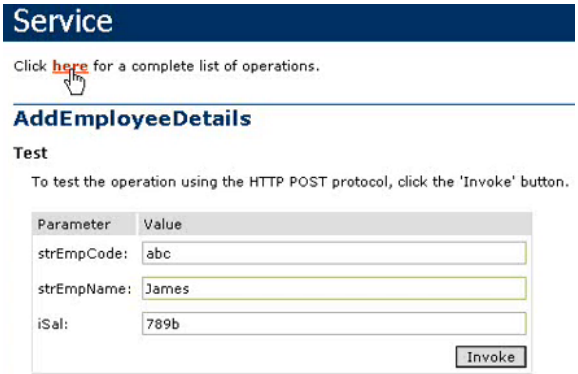


SOAP REQUEST FRAME

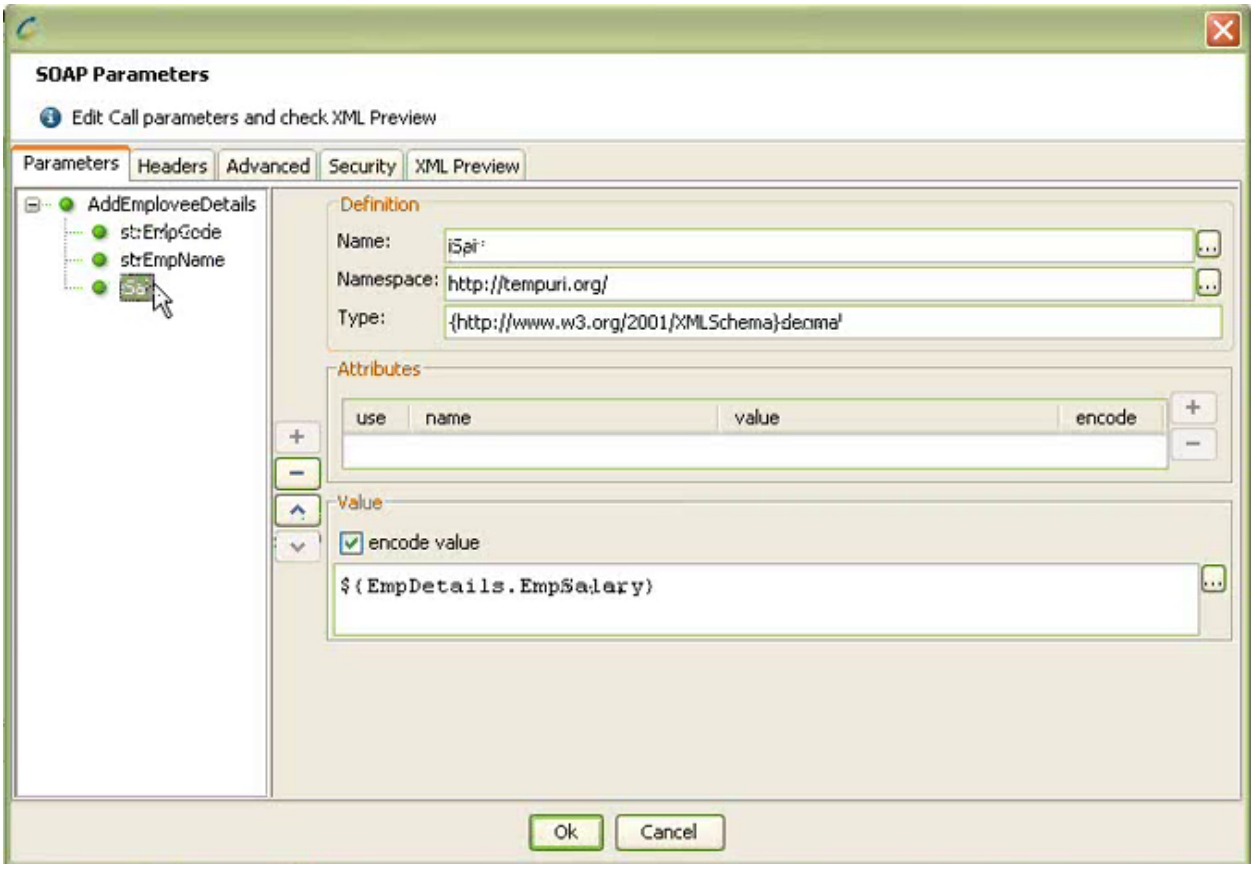
- URL: http://localhost:1165/EmpMgmt/Service.asmx
- Server: localhost
- Path: /EmpMgmt/Service.asmx
- WSDL: http:localhost:1165/EmpMgmt/Service.asmx?wsdl
- Edition Mode: **Parameters(from WSDL)** / FulXML
- Method: AddEmployeeDetails(AddEmployeeDetails)

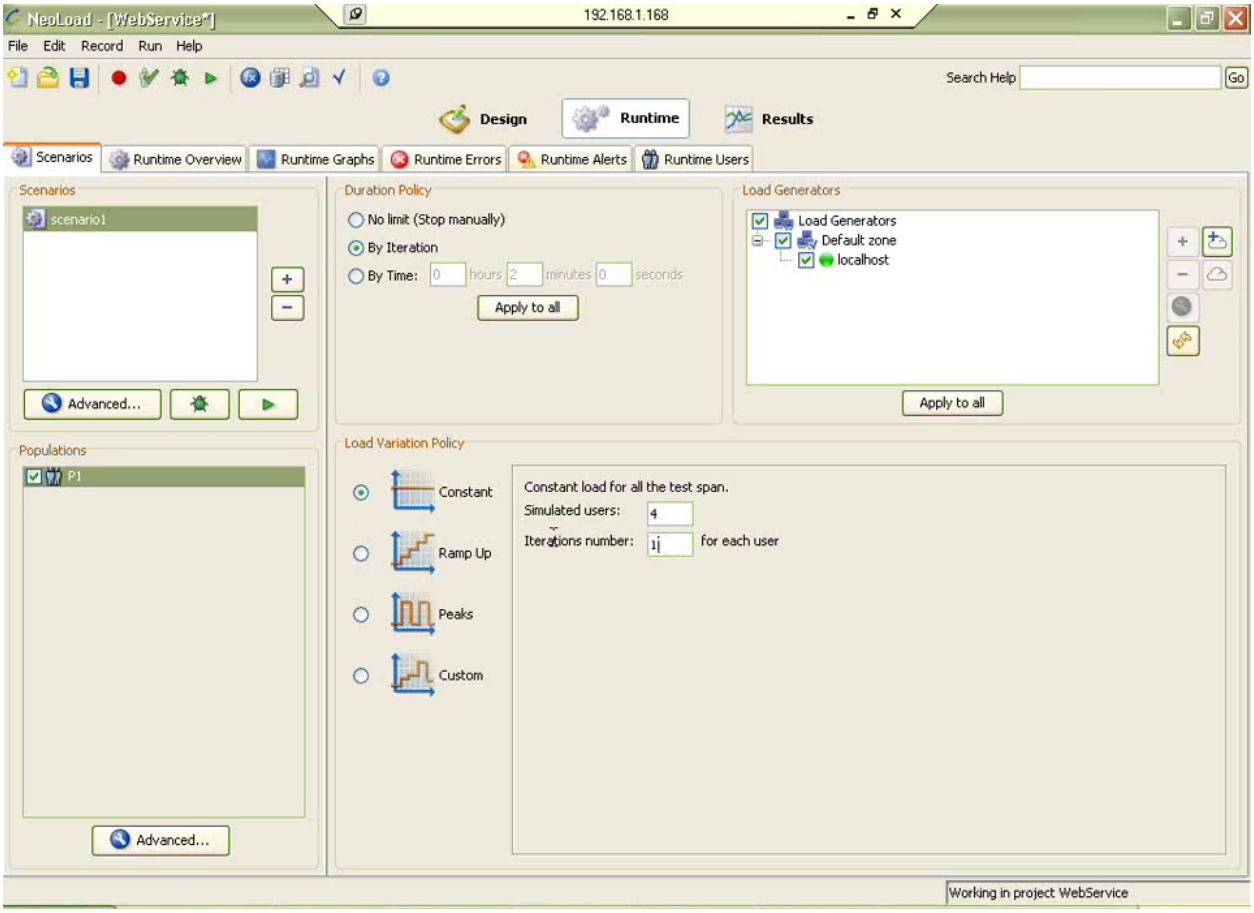
- ‘Edit Call Parameters’ -> Variables Manager



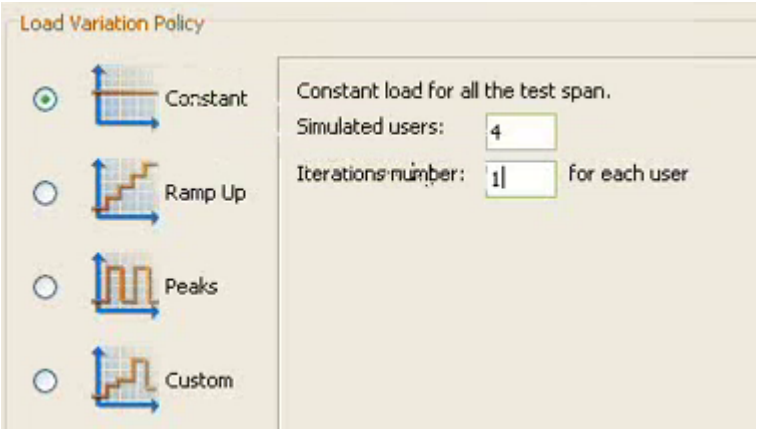


- Once URL given, NeoLoad will pickup the parameters from the WebService ASMX page

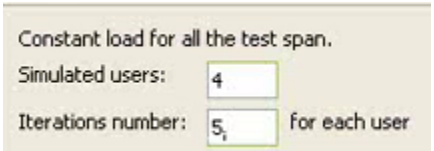




CONCURRENT USERS



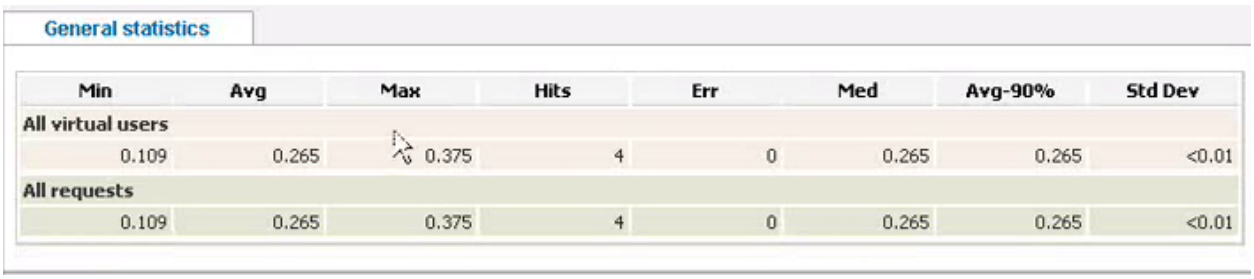
CONCURRENT USERS WITH ITERATIONS



- Need lots of data for load testing

RUN

RESULTS OF TEST



General statistics

Min	Avg	Max	Hits	Err	Med	Avg-90%	Std Dev
All virtual users							
0.109	0.265	0.375	4	0	0.265	0.265	<0.01
All requests							
0.109	0.265	0.375	4	0	0.265	0.265	<0.01

VS2010 UNIT TESTING

Caller <-> UnitTest Code <-> Webservice

- Create another project for unit tests of code
- How to assign test to the unit test?
    - Service References
        - ServiceReferenceEmpMgmt
- This will cause syntax highlight and code hinting to work against that webservice (AddEmployee())..

```
ServiceSoapClient objEmpMgmt = new ServiceSopClient();
try{
      objEmpMgmt.AddEmloyeeDetails("emp1","Ram",100);
} catch(Exception ex) {
      // Log Error…
}
```

VS2012:

- Caller: Program.cs
- UnitTest: UnitTest.cs

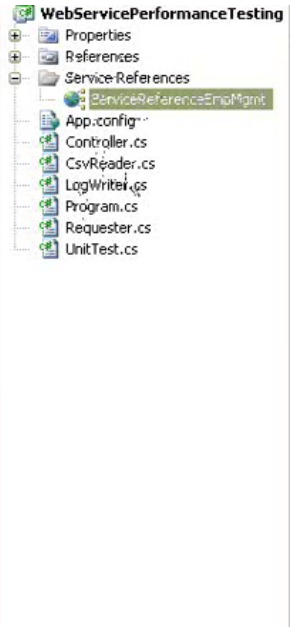
- WebService: AddEmployee.asmx

WEBSERVICEPERFORMANCETESTING.CS

- UnitTest object with Methods for the actual UnitTest

```
using System;
using WebServicePerformanceTesting.ServiceReferenceEmpMgmt;

namespace WebServicePerformanceTesting
{
    class UnitTest
    {
        public void UnitTestWebService()
        {
            ServiceSoapClient objEmpMgmt = new ServiceSoapClient();
            LogWriter objLog = new LogWriter();
            try
            {
                objEmpMgmt.AddEmployeeDetails("emp1", "Ram", 100);
            }
            catch (Exception ex)
            {
                objLog.WriteToErrLog(ex.Message);
            }
            finally
            {
                objEmpMgmt = null;
                objLog = null;
            }
        }
    }
}
```



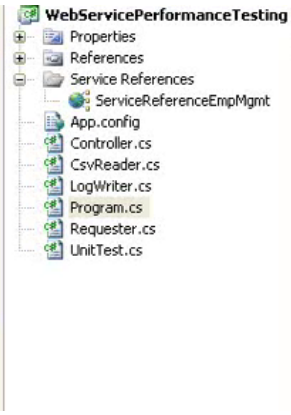
PROGRAM.CS

- Creates the UnitTest object and calls needed methods.
- Probably only ONE UnitTest per 'Cass Under Test'

```
using System;
using System.Threading;

namespace WebServicePerformanceTesting
{
    class Program
    {
        private int vUserCount = 5;
        private int iterations = 1;

        static void Main(string[] args)
        {
            //For Unit Test
            UnitTest objUT = new UnitTest();
            objUT.UnitTestWebService_001();
            objUT.UnitTestWebService_002();
            objUT.UnitTestWebService_001();
```



DON'T HARD CODE DATA

- External text file: CSV
- Excel Sheet
- XML
- Database