WINDOWS WORKFLOW FOUNDATION TUTORIAL

WCF Workflow Services

win technologies
A proud member of the betway group

PRESENTED BY: GLEN THOMAS

INTRODUCTION

OVERVIEW

Windows Workflow Foundation (WF) is a Microsoft technology that provides an API, an in-process workflow engine, and a rehostable designer to implement long-running processes as workflows within .NET applications.

The purpose of this tutorial is to provide an understanding of some basic Windows Workflow Foundation techniques for setting up new workflow services.

We will cover creating a Windows Communication Foundation (WCF) workflow service application that performs some simple tasks.

BENEFITS OF USING WORKFLOW SERVICES

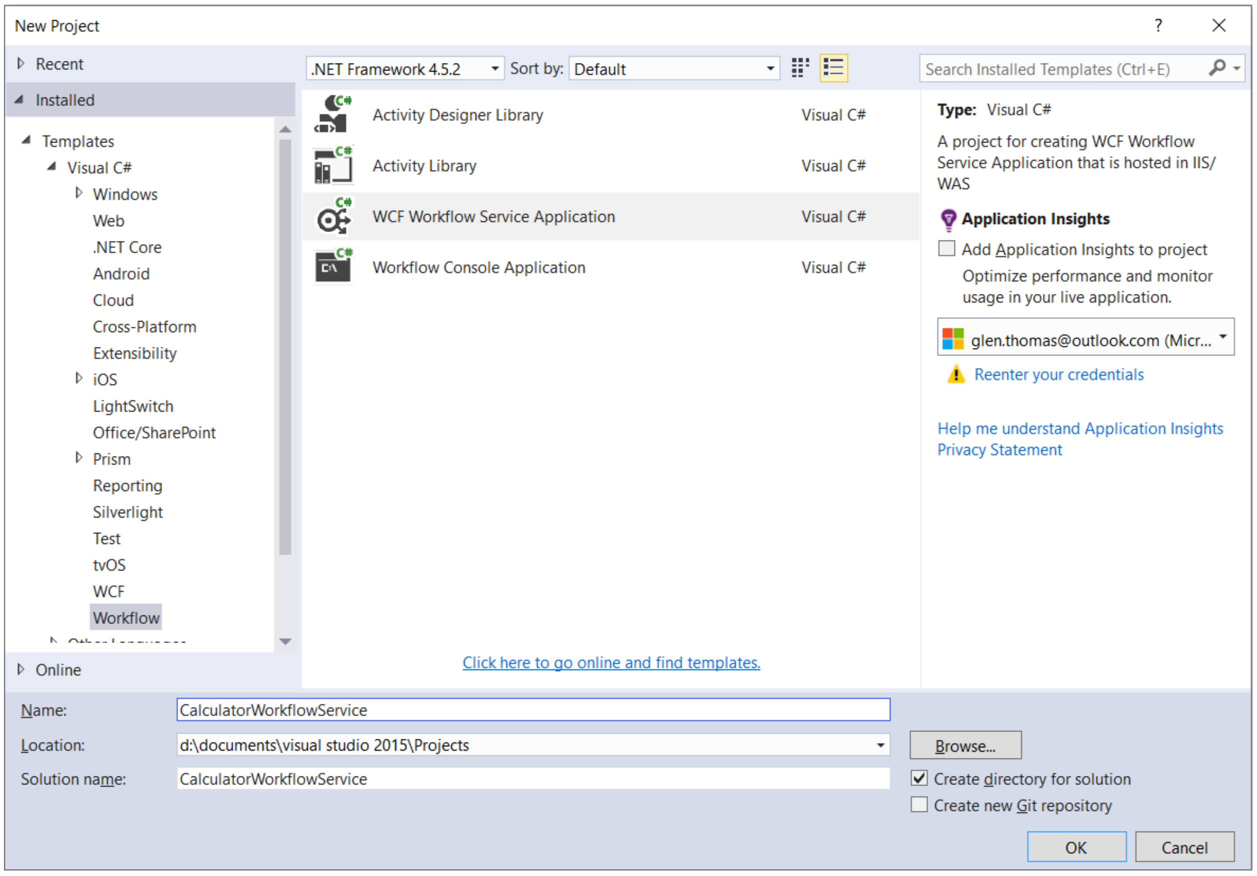
As applications become increasingly distributed, individual services become responsible for calling other services to offload some of the work. Implementing these calls as asynchronous operations introduces some complexity into the code. Error handling adds additional complexity in the form of handling exceptions and providing detailed tracking information. Some services are often long running and can take up valuable system resources while waiting for input. Because of these issues, distributed applications are often very complex and difficult to write and maintain. Workflows are a natural way to express the coordination of asynchronous work, especially calls to external services. Workflows are also effective at representing long-running business processes.

PART 1 – CREATING A WCF WORKFLOW SERVICE

Open Visual Studio 2015 and select File -> New -> Project.

Under the Visual C# project templates, select Workflow and then choose the WCF Workflow Service Application project type.

In the name box enter “CalculatorWorkflowService” and click OK.



Activity Designer Library	Visual C#
Activity Library	Visual C#
WCF Workflow Service Application	Visual C#
Workflow Console Application	Visual C#

Type: Visual C#

A project for creating WCF Workflow Service Application that is hosted in IIS/ WAS

Application Insights

Add Application Insights to project

Optimize performance and monitor usage in your live application.

glen.thomas@outlook.com (Micr...

Reenter your credentials

Help me understand Application Insights

Privacy Statement

Name: CalculatorWorkflowService

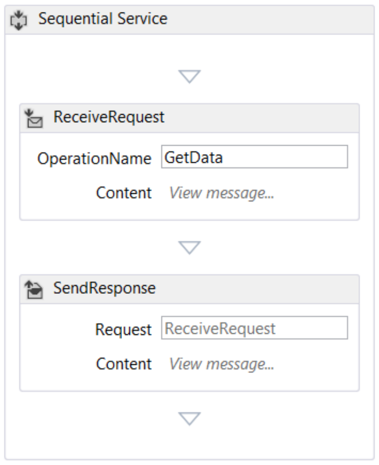
Location: d:\documents\visual studio 2015\Projects

Solution name: CalculatorWorkflowService

✓ Create directory for solution

Create new Git repository

A basic workflow service called “Service1.xamlx” has been created and is displayed in the workflow designer for editing.



OperationName: GetData

Content: View message...

Request: ReceiveRequest

Content: View message...

Right-click the service xamlx file and select Rename. Enter the name “MultiplyWorkflow.xamlx”. Right-click the xamlx file again and select “View Code”. Change the name attribute of the WorkflowService to “MultiplyWorkflow”.

STARTING A WORKFLOW INSTANCE

The default workflow template consists of a Receive activity followed by a SendReply activity. These two activities are added to the workflow to provide a way to begin a new workflow instance. The Receive activity will generate a WCF SOAP operation using contract inference.

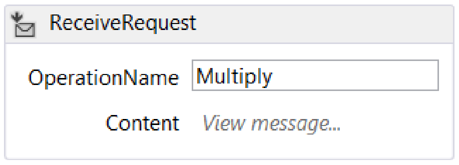
Contract Inference

When a workflow service is hosted using WorkflowServiceHost, the workflow definition is examined and a contract is generated based on the set of messaging activities found in the workflow. The end result of contract inference is a description of the service using the same data structures as WCF service and operation contracts. This information is then used to expose WSDL for the workflow service.

Right-click on the Receive activity and select Properties. In the properties inspector you will see that the CanCreateInstance option is checked. This will cause a new workflow instance to be created when the operation is invoked.

The Receive activity is paired with a SendReply activity to transmit a response object to the caller of the Receive method.

Click in the OperationName text box of the Receive activity in the designer window and change the OperationName to “Multiply”.


ReceiveRequest

OperationName: Multiply

Content: View message...

CREATING WORKFLOW VARIABLES

The Multiply service method will accept two integer parameters and we will need to store the parameters being passed for the rest of the workflow to access them.

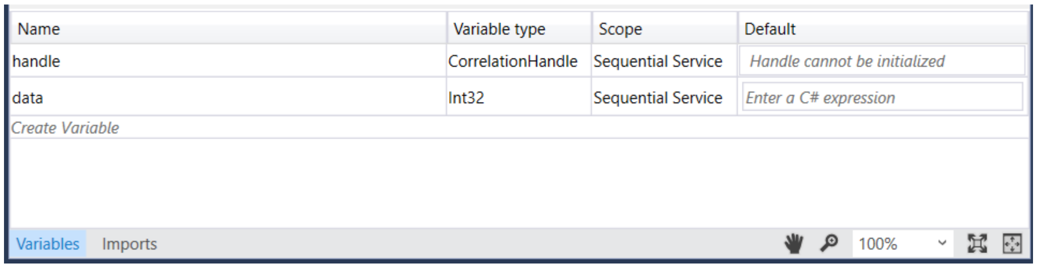
Workflow Variables

Variables are storage locations for data. Variables are declared as part of the definition of a workflow. Variables take on values at runtime and these values are stored as part of the state of a workflow instance. A variable definition specifies the type of the variable and optionally, the name.

Creating a variable in code:

```
Variable<string> var = new Variable<string>
{
    Name = "str"
};
```

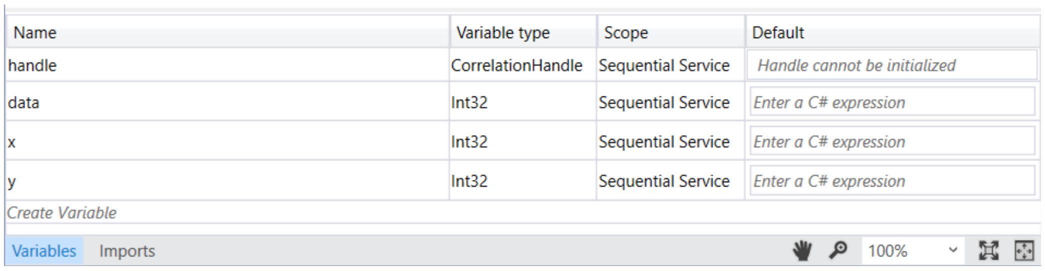
Click the Variables button at the bottom of the workflow designer to access the list of workflow variables and select the “Sequential Service” sequence activity to see all variables within that scope.



Name	Variable type	Scope	Default
handle	CorrelationHandle	Sequential Service	*Handle cannot be initialized*
data	Int32	Sequential Service	*Enter a C# expression*
Create Variable			

There are two variables added by default, which are not required and may be deleted.

Click on the new line (“Create Variable”) and add a new variable with name “x” and type Int32. Then add another called “y”.



Name	Variable type	Scope	Default
handle	CorrelationHandle	Sequential Service	*Handle cannot be initialized*
data	Int32	Sequential Service	*Enter a C# expression*
x	Int32	Sequential Service	*Enter a C# expression*
y	Int32	Sequential Service	*Enter a C# expression*
Create Variable			

Variable Scoping

The lifetime of a variable at runtime is equal to the lifetime of the activity that declares it. When an activity completes, its variables are cleaned up and can no longer be referenced.

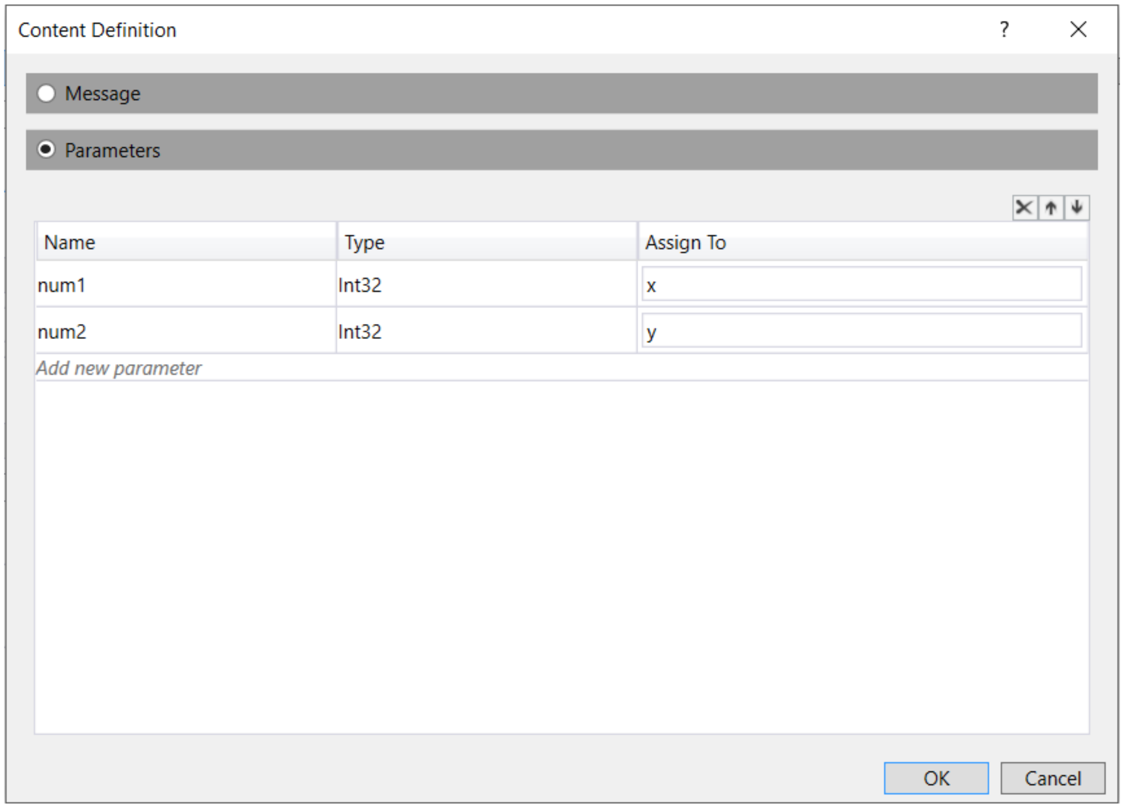
ADDING INPUT PARAMETERS TO THE WORKFLOW

Right-click on the Receive activity and select Properties. In the properties inspector click the “…” button of the Content property. This will open the Content Definition window where we can add parameters to the operation.

Data Contracts

The Receive activity can also accept a data contract object as a single parameter. In this case a data contract class must be created in the workflow service, which is then presented to the WCF client via the WSDL.

Select the Parameters radio button and add two Int32 parameters called “num1” and “num2” and assign to the workflow variables x and y.

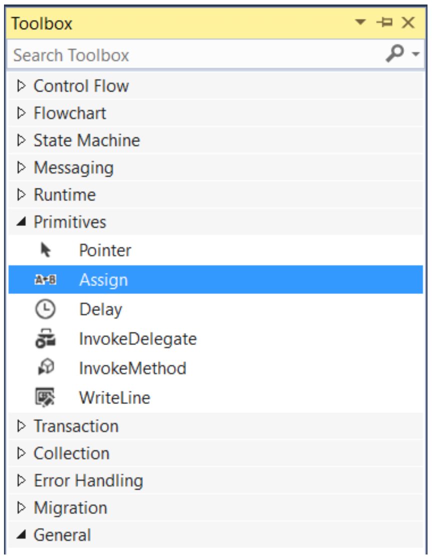


Name	Type	Assign To
num1	Int32	x
num2	Int32	y
Add new parameter		

ADDING A WORKFLOW ACTIVITY

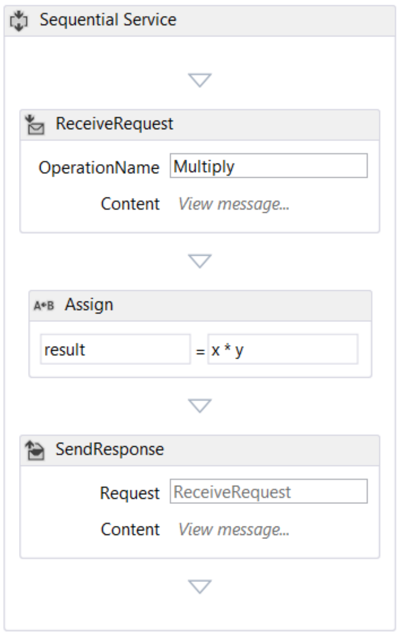
We will now add a new activity to the workflow that will multiply the two variables. Firstly, add a new Int32 workflow variable to the sequence scope called “result”, this will hold the result of the multiplication.

Now open the Toolbox pane in Visual Studio with the workflow designer open and expand the Primitives group.



Select the assign activity and drag-drop the activity onto the workflow designer between the Receive and SendReply activities.

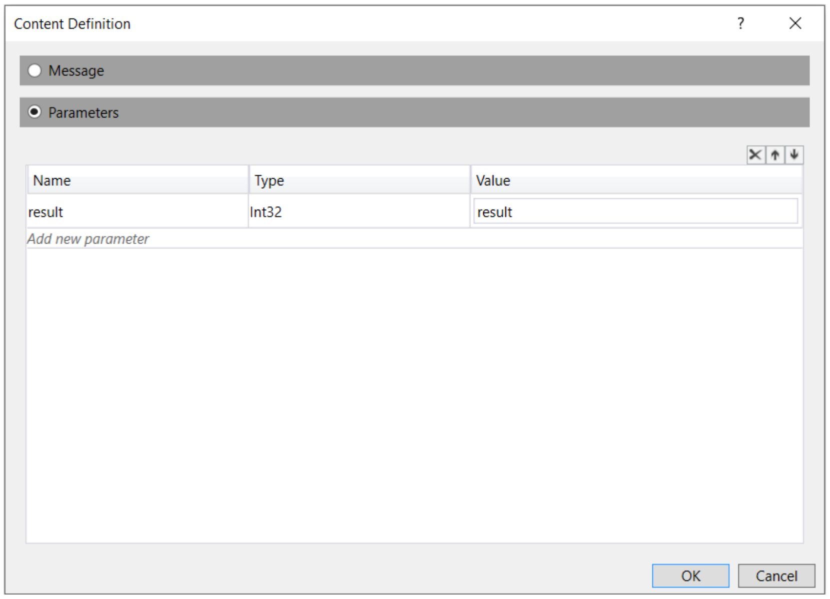
In the To box enter "result" to specify that the result workflow variable will be assigned to. In the C# expression box enter "x * y" so that the result variable will be assigned the multiplication product of the x and y variables.



SEND THE REPONSE

To send the result variable to the caller of the Multiply method, click the "…" button of the Content property of the SendReply activity to open the Content Definition window.

Select the Parameters radio button and add a parameter called result of type Int32, set the value to the result workflow variable and click OK.



Name	Type	Value
result	Int32	result
Add new parameter		

Setting the Content property of the SendReply activity will change the return type of the service method generated through contract inference to the type specified and the value of the return object will be set automatically to what is specified in the value field. This could be a workflow variable, a C# expression or simply a constant value.

PART 2 – HOSTING THE WORKFLOW SERVICE

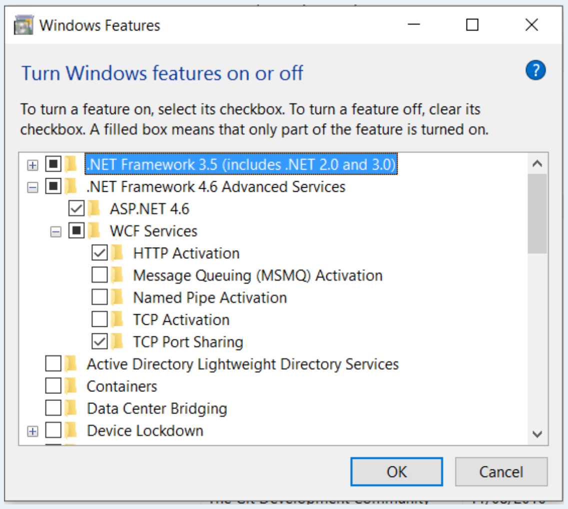
CONFIGURING THE SERVICE PROJECT

Right-click the CalculatorWorkflowService project and select Properties. Open the Web settings and under the Server section select "Local IIS". You will see a prompt asking if you want to configure the virtual directory; click Yes.

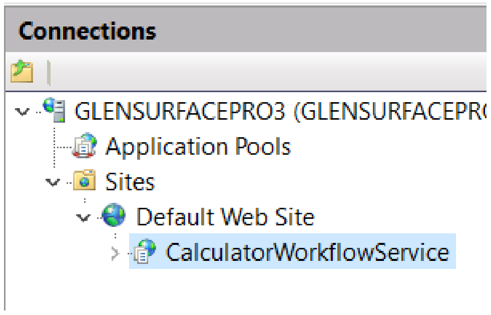
IIS Hosting

IIS features must be enabled in Windows for the Local IIS option to be available and Visual Studio must be run as an administrator in order to create the virtual directory in IIS.

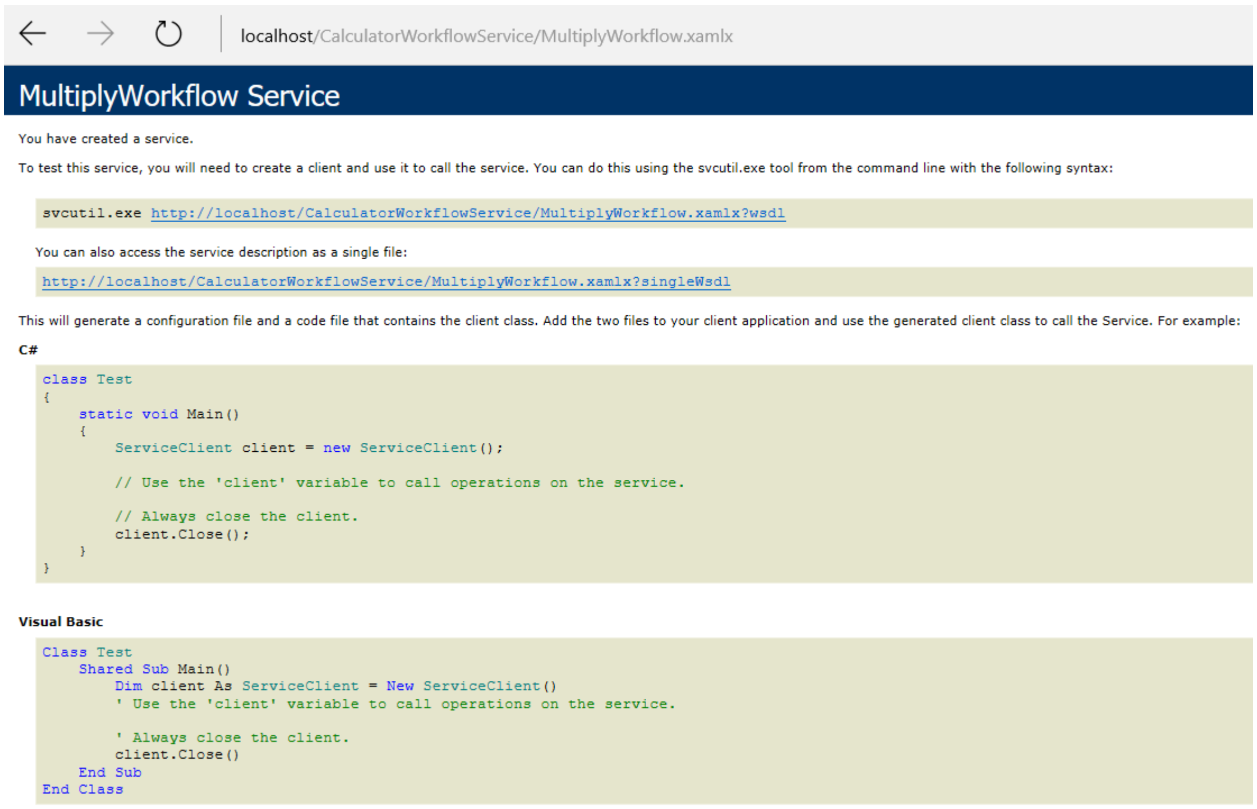
.NET WCF Services HTTP Activation feature must also be enabled


Windows Features
Turn Windows features on or off
To turn a feature on, select its checkbox. To turn a feature off, clear its checkbox. A filled box means that only part of the feature is turned on.
.NET Framework 3.5 (includes .NET 2.0 and 3.0)
.NET Framework 4.6 Advanced Services
ASP.NET 4.6
WCF Services
HTTP Activation
Message Queuing (MSMQ) Activation
Named Pipe Activation
TCP Activation
TCP Port Sharing
Active Directory Lightweight Directory Services
Containers
Data Center Bridging
Device Lockdown
OK
Cancel

Right-click the CalculatorWorkflowService project and select "Build" to build the workflow service application. This will add the CalculatorWorkflowService application to the default web site in IIS.



With the application selected, click the "Content View" button to see the files in the virtual directory. Right-click the MultiplyWorkflow.xamlx file and select "Browse" to load the service in the default web browser and confirm that all is set up correctly. You will see the default WCF service page


localhost/CalculatorWorkflowService/MultiplyWorkflow.xamlx

MultiplyWorkflow Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://localhost/CalculatorWorkflowService/MultiplyWorkflow.xamlx?wsdl
```

You can also access the service description as a single file:

```
http://localhost/CalculatorWorkflowService/MultiplyWorkflow.xamlx?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        ServiceClient client = new ServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As ServiceClient = New ServiceClient()
        ' Use the 'client' variable to call operations on the service.

        ' Always close the client.
        client.Close()
    End Sub
End Class
```

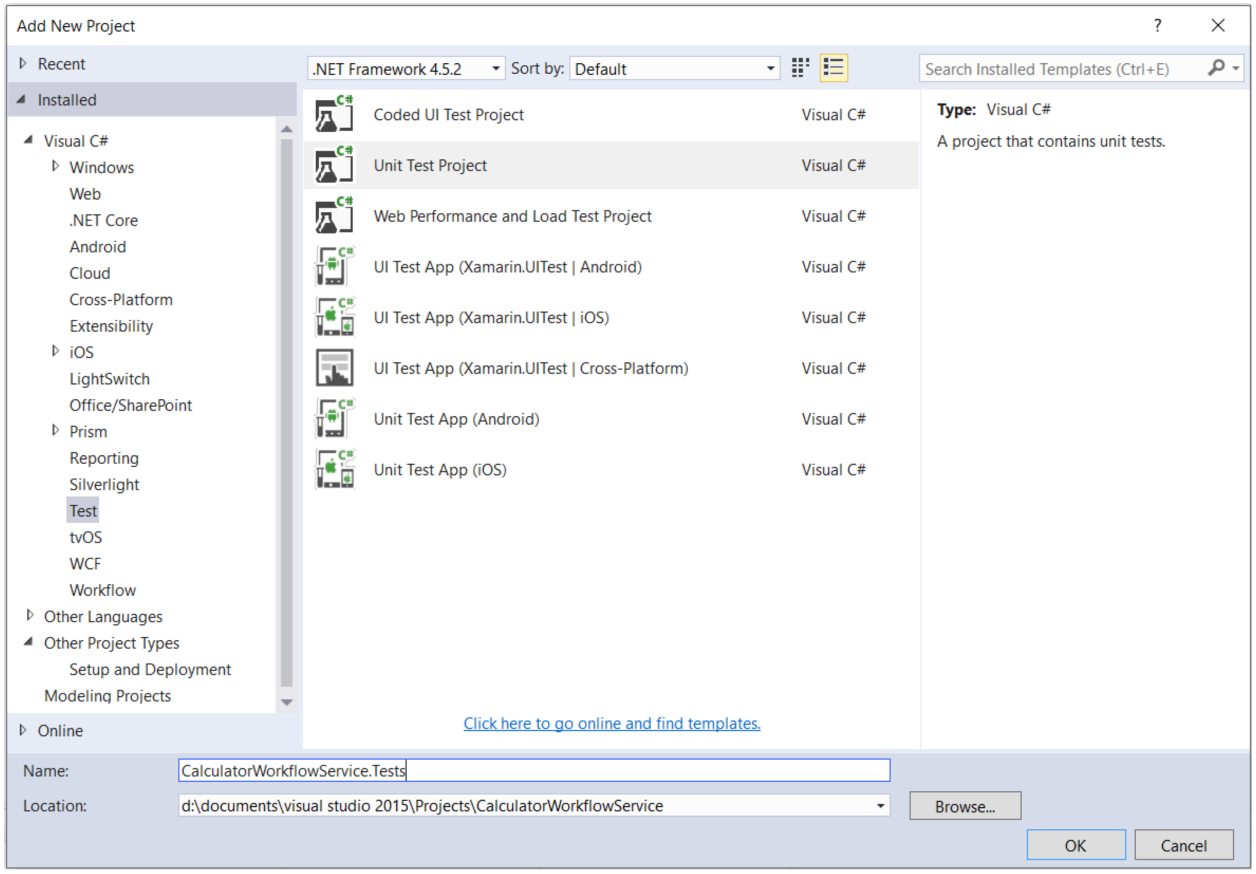
PART 3 – TESTING THE WORKFLOW SERVICE

CREATING A TEST PROJECT

Right-click the CalculatorWorkflowService solution and select Add -> New Project.

Select the Test section under Visual C# and choose the Unit Test Project template.

Enter “CalculatorWorkflowService.Tests” for the project name and click OK.



Add New Project

- Recent
- Installed
  - Visual C#
    - Windows
    - Web
    - .NET Core
    - Android
    - Cloud
    - Cross-Platform
    - Extensibility
    - iOS
    - LightSwitch
    - Office/SharePoint
    - Prism
    - Reporting
    - Silverlight
    - Test
    - tvOS
    - WCF
    - Workflow
  - Other Languages
  - Other Project Types
    - Setup and Deployment
  - Modeling Projects
- Online

.NET Framework 4.5.2 | Sort by: Default | Search Installed Templates (Ctrl+E)

Coded UI Test Project	Visual C#
Unit Test Project	Visual C#
Web Performance and Load Test Project	Visual C#
UI Test App (Xamarin.UITest | Android)	Visual C#
UI Test App (Xamarin.UITest | iOS)	Visual C#
UI Test App (Xamarin.UITest | Cross-Platform)	Visual C#
Unit Test App (Android)	Visual C#
Unit Test App (iOS)	Visual C#

Type: Visual C#

A project that contains unit tests.

Click here to go online and find templates.

Name: CalculatorWorkflowService.Tests

Location: d:\documents\visual studio 2015\Projects\CalculatorWorkflowService

Browse... | OK | Cancel

Rename the default test class UnitTest1 to “MultiplyTests” and open it. Rename the default test method TestMethod1 to “MultiplyTest”.

ADDING A SERVICE REFERENCE TO THE WORKFLOW SERVICE

Right-click the CalculatorWorkflowService.Tests project and select Add -> Service Reference.

In the Address field enter the URL to the MultiplyWorkflow xamlx (http://localhost/CalculatorWorkflowService/MultiplyWorkflow.xamlx) and click Go. The operations from the workflow WCF service contract will be displayed. Change the namespace to “MultiplyWorkflowService” and click OK. A Service References folder containing the operation contract will be added to the test project.

The WCF configuration for the service client is automatically added to the test project’s app.config file to be loaded at runtime when creating an instance of the service client.

```xml
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="BasicHttpBinding_IService" />
    </basicHttpBinding>
  </bindings>
  <client>
    <endpoint address="http://localhost/CalculatorWorkflowService/MultiplyWorkflow.xamlx"
          binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IService"
          contract="MultiplyWorkflowService.IService" name="BasicHttpBinding_IService" />
  </client>
</system.serviceModel>
```

The address of the endpoint can be modified in the config file to suit the location of the workflow service, or this can be specified as a constructor parameter when instantiating the service client in code.

WRITING THE WORKFLOW TEST

Add the following code to the MultiplyTest method:

```csharp
var num1 = 2;
var num2 = 5;
var expectedResult = 10;
int actualResult;

using (var multiplyClient = new MultiplyWorkflowService.ServiceClient())
{
    actualResult = multiplyClient.Multiply(num1, num2);
}

Assert.AreEqual(expectedResult, actualResult);
```

The test creates a WCF service client for the MultiplyWorkflow service and invokes the multiply method passing two integers as parameters and receiving an integer result.

Run the test and check that the assertion passes.