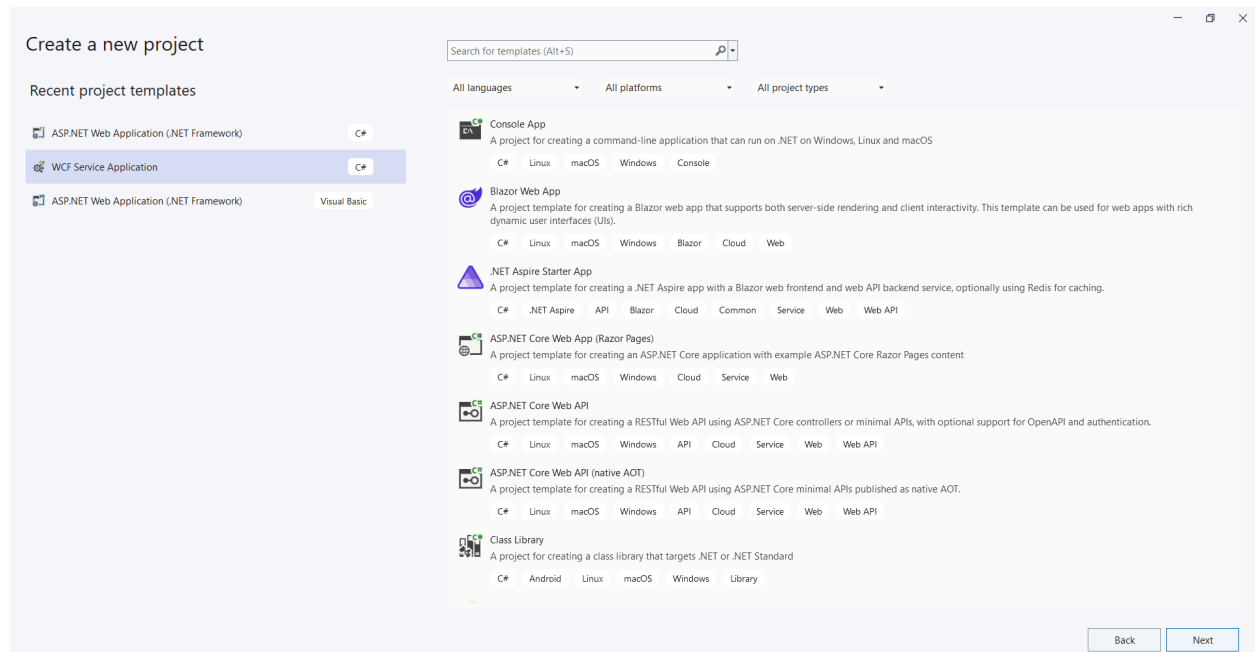
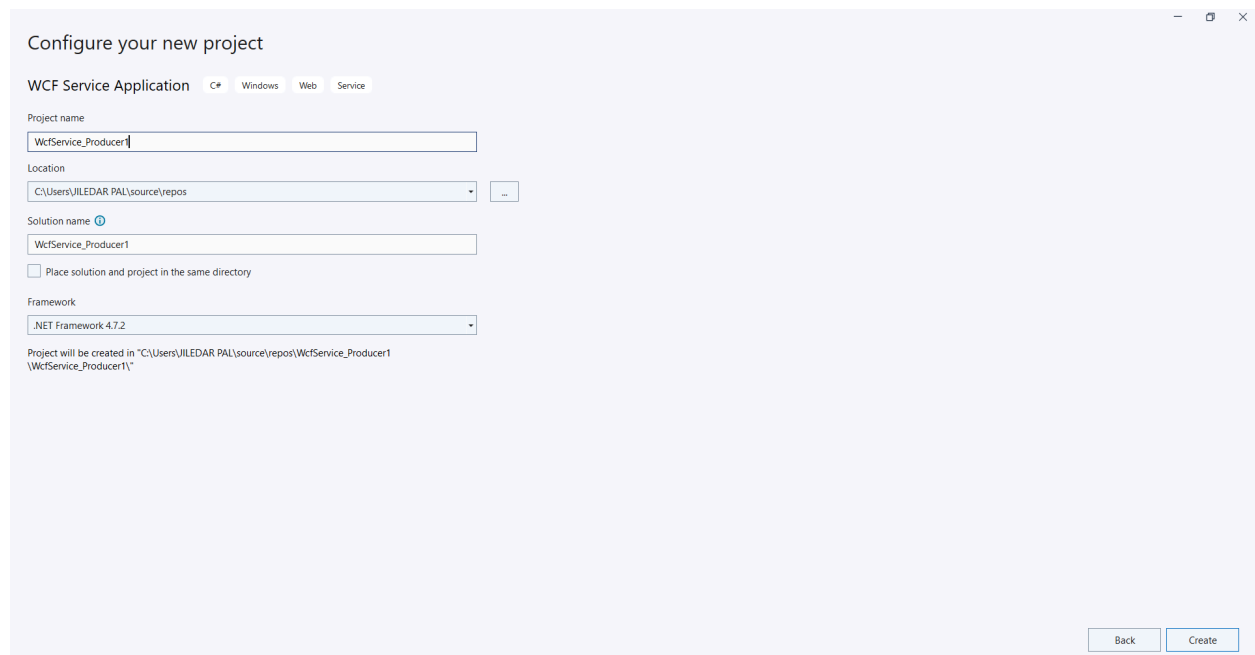


# WEB SERVICES using IService files

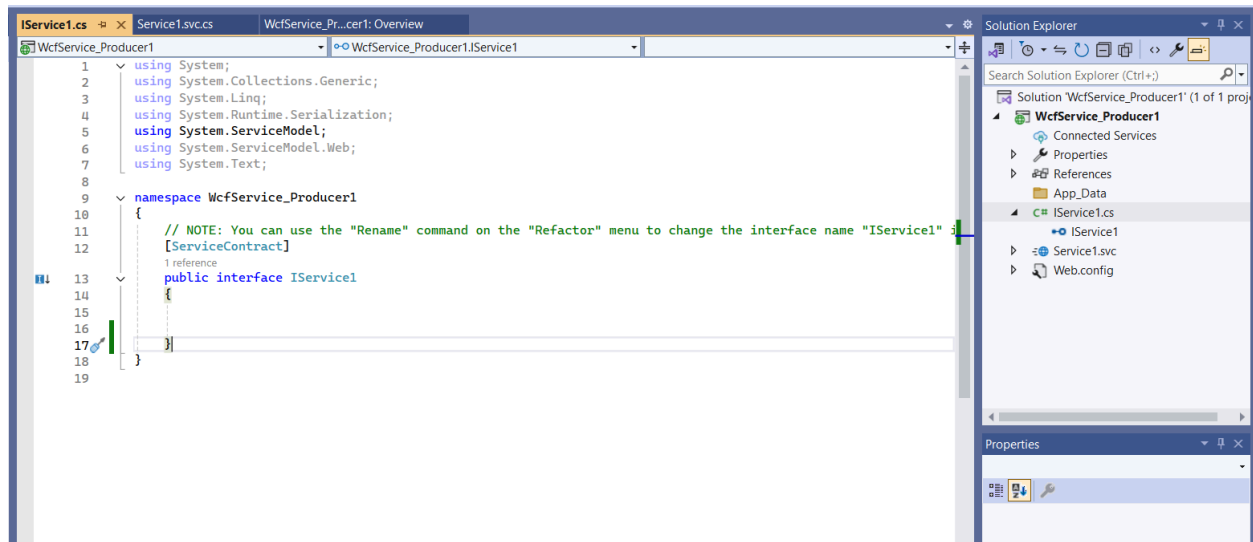
## Step-1



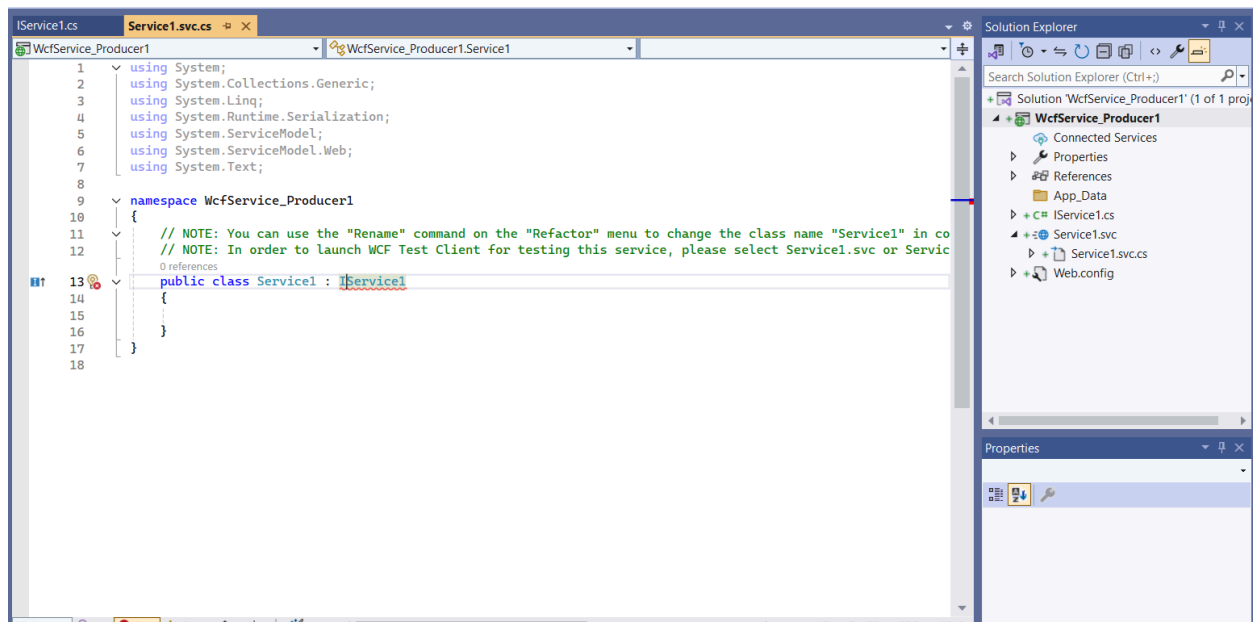
## Step-2



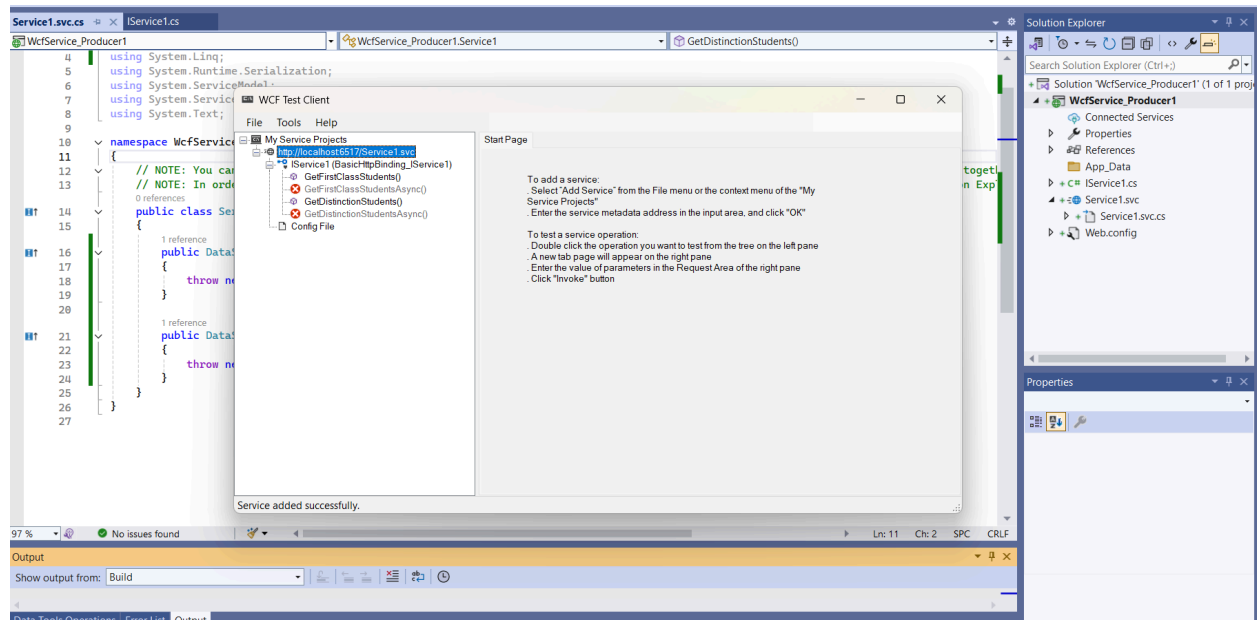
**Step-3: Remove all the code from this file just keep the file in this form and add your logic**



**Step-4: after that Just open the given file that is shown in this image and right click on the line which shows error then you will get option of implement interfaces**



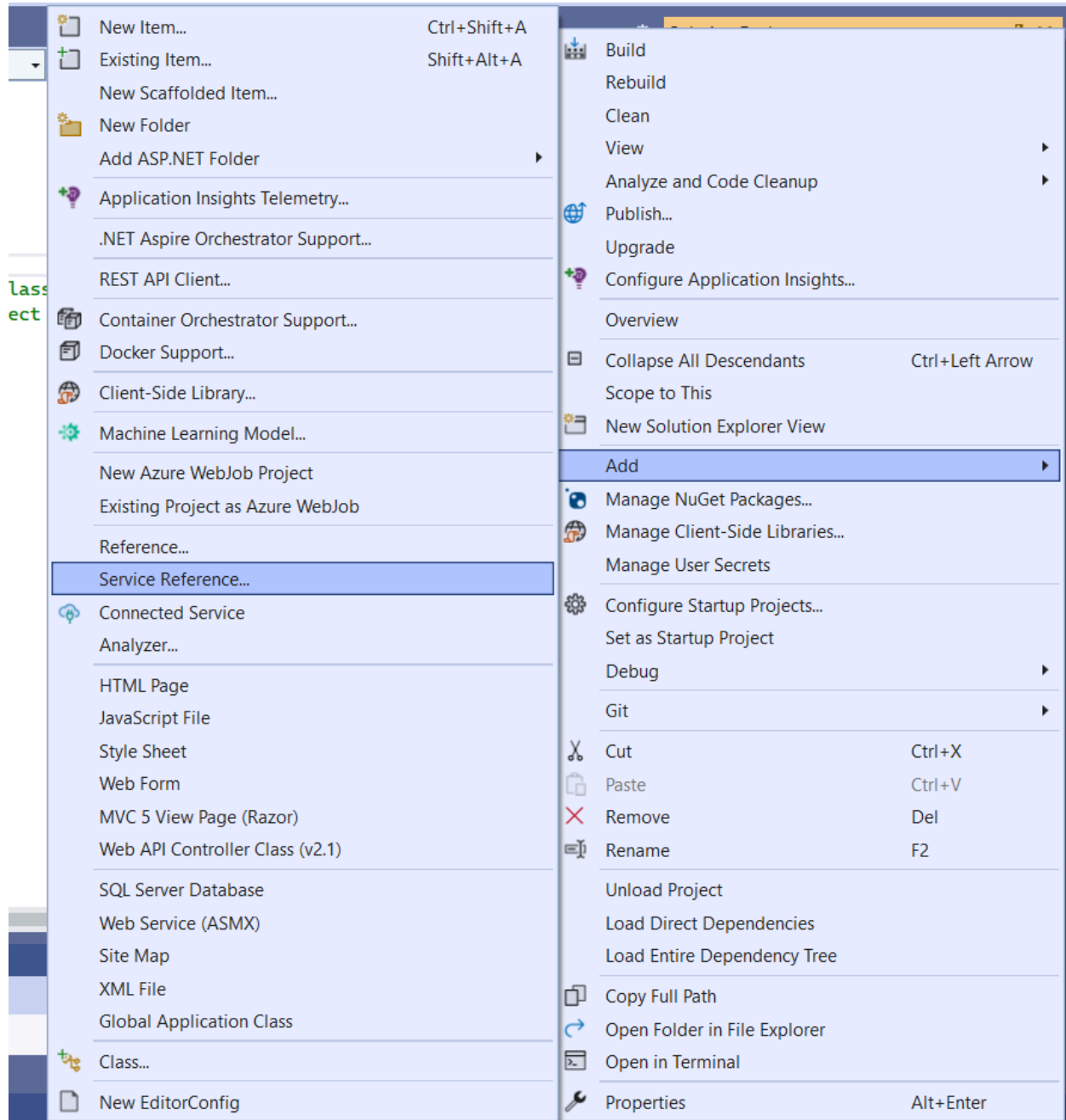
## Step-5: After that run you Application



And copy the selected url as shown

After that just create **WebService\_Consumer** application and on right click just do the following:

**Step-6 : By right clicking on main application file you will get the following just click on “ServiceReference”**



**Step-7 : Just paste that url in address section and click on go**

Add Service Reference?×

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

▼GoDiscover▼

Services:

Operations:

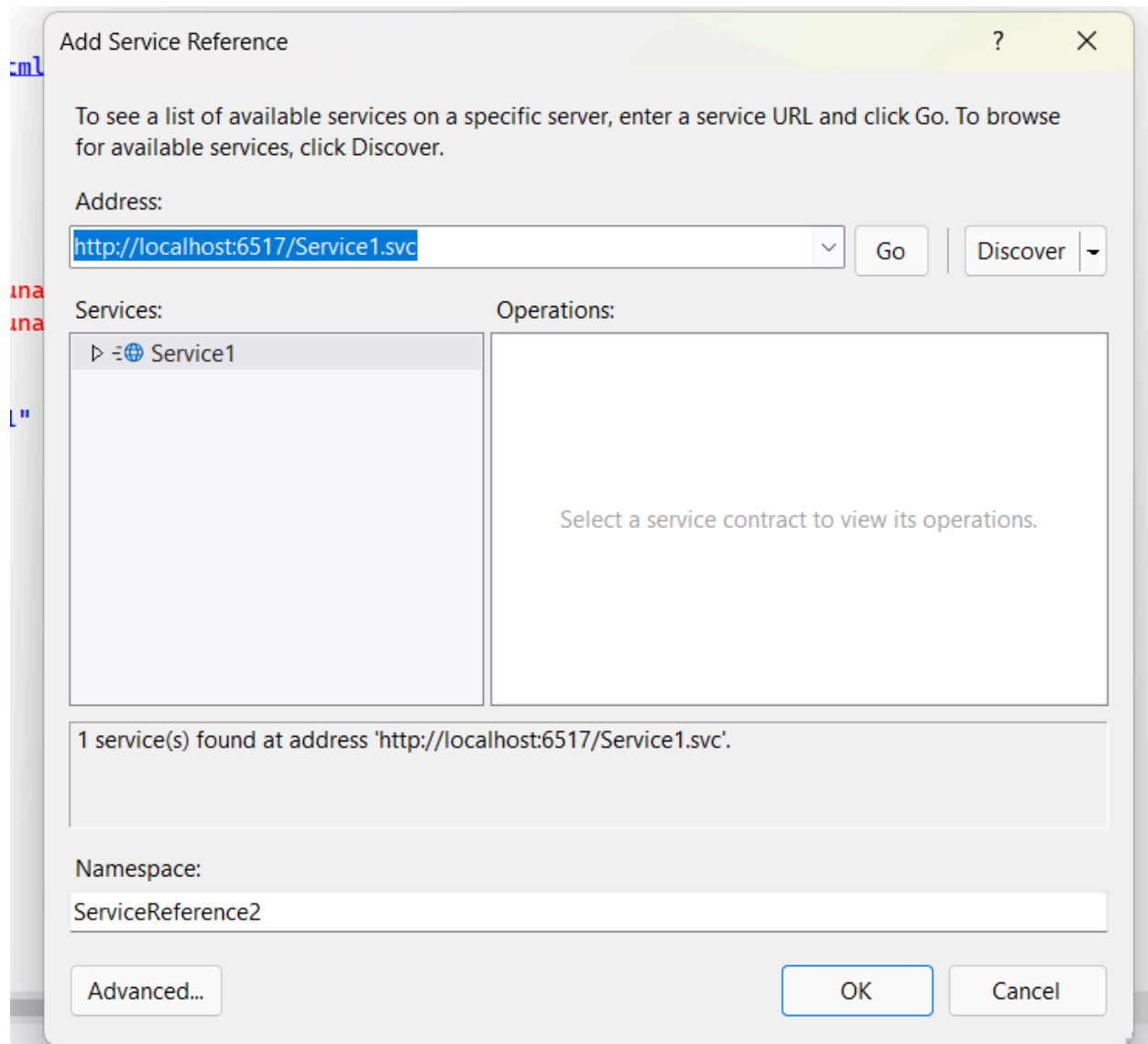
Namespace:

Advanced...

OK

Cancel

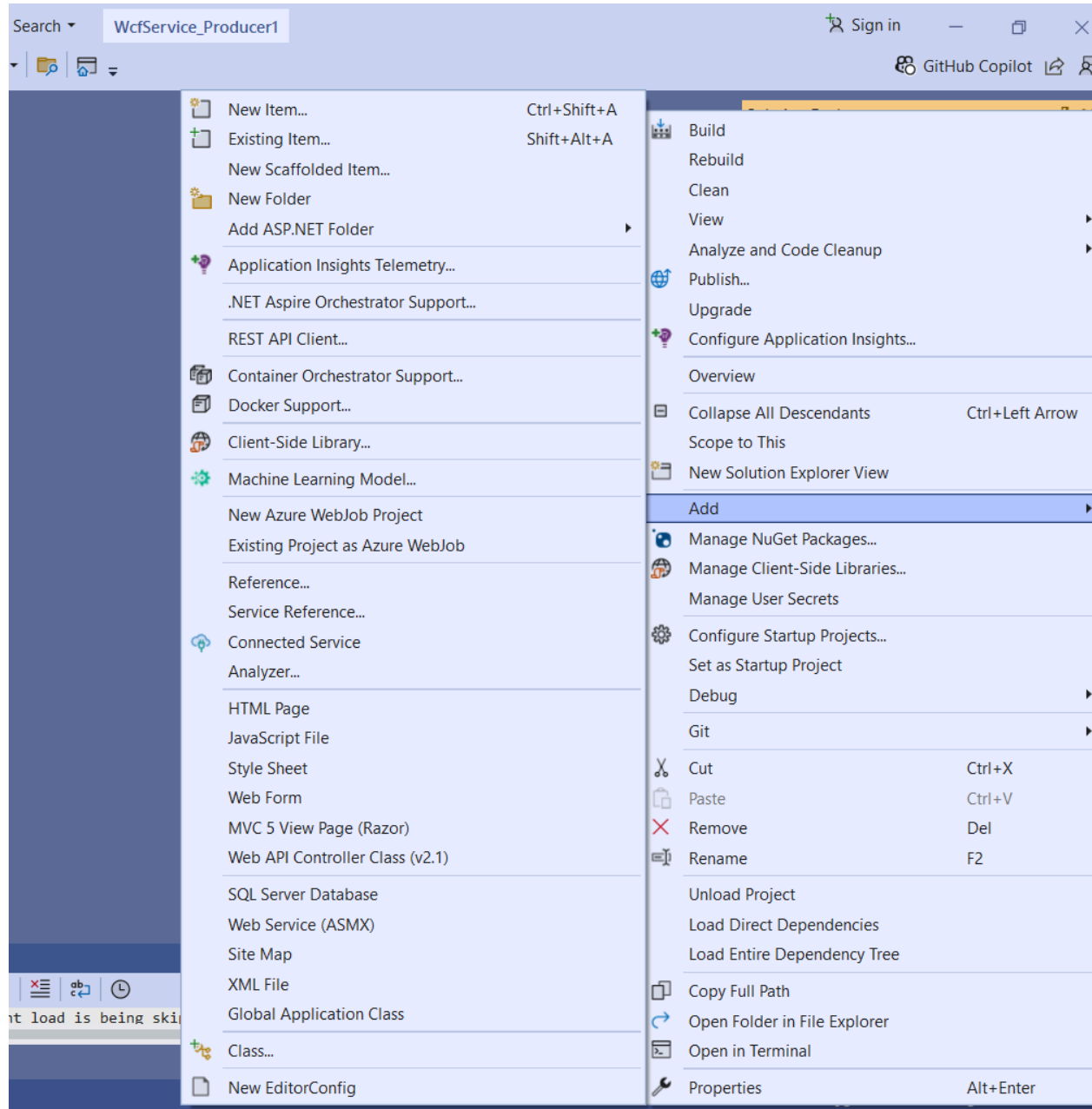
**Step-8: Then u will get the below interface**



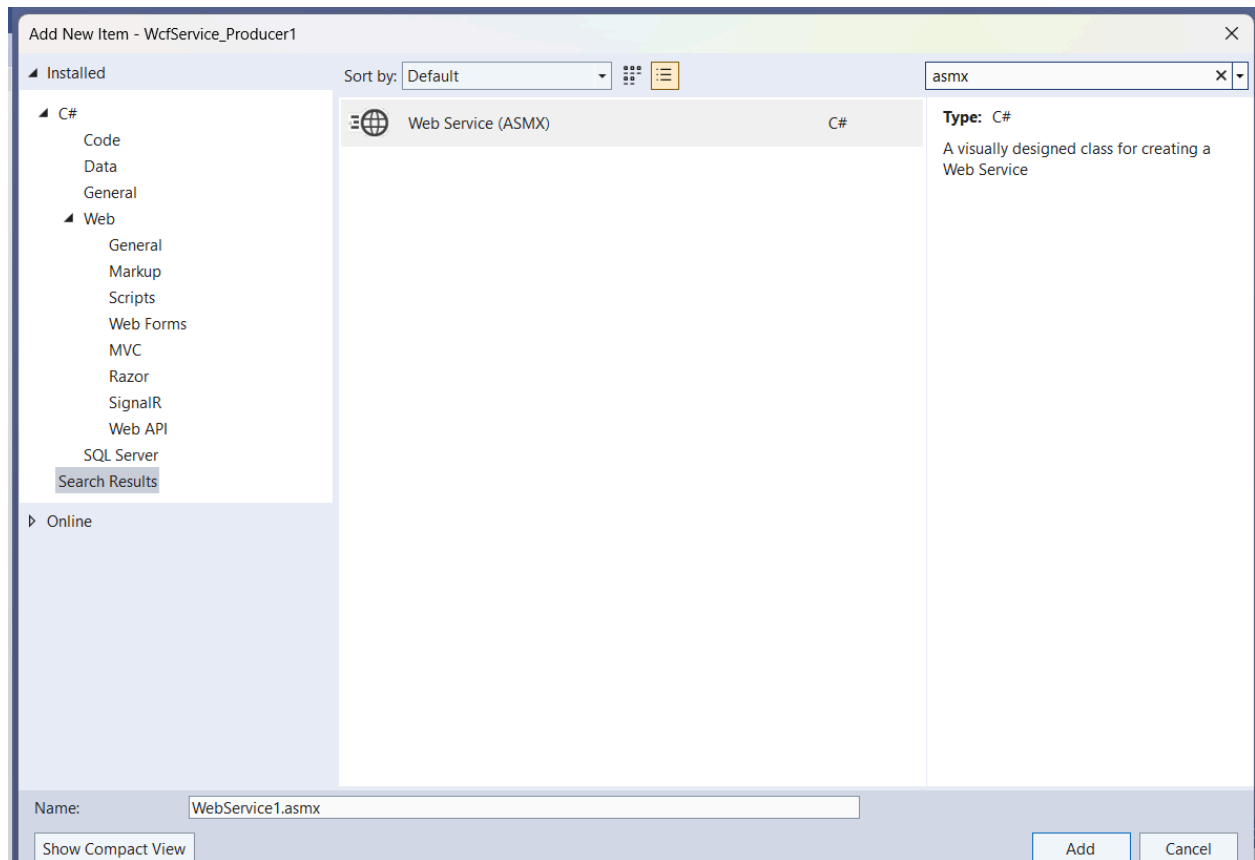
**After that just add web forms and add the code.**

## WEB Services using ASMX file

**Step-1: Right click on main application file (WcfService\_Producer) and click on new item**

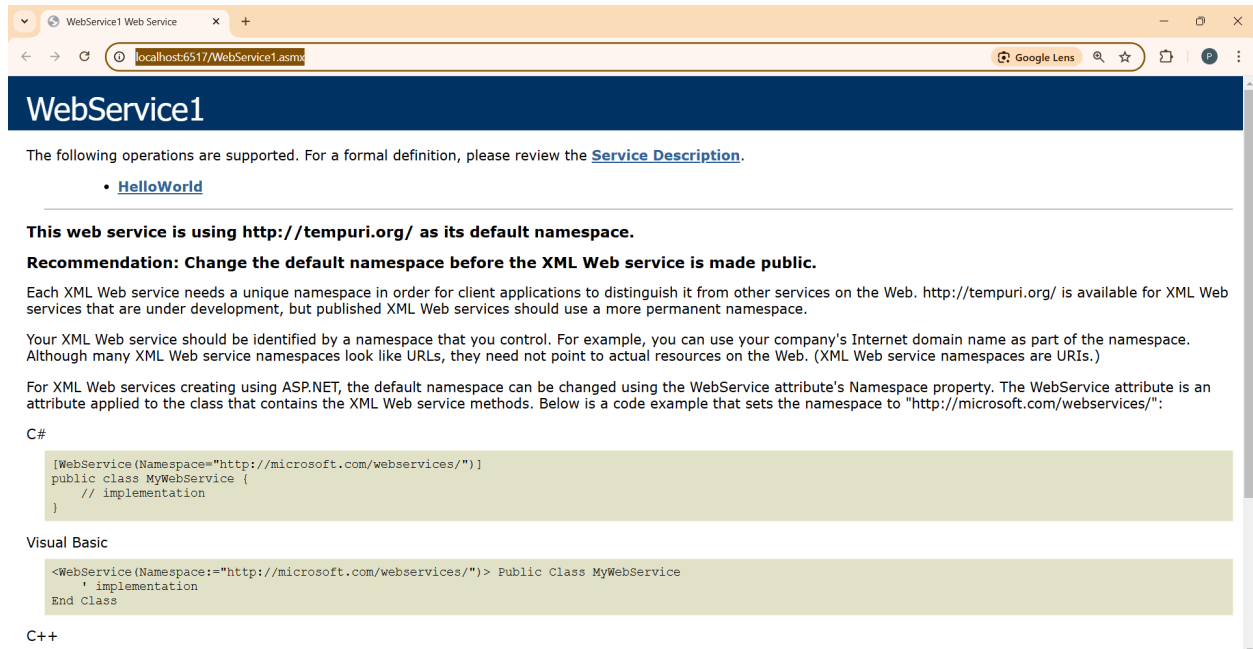


## Step-2: Search asmx file





### Step-3: After the adding the logic in this page just run this page and copy the url



WebService1

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [HelloWorld](#)

**This web service is using <http://tempuri.org/> as its default namespace.**

**Recommendation: Change the default namespace before the XML Web service is made public.**

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. <http://tempuri.org/> is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "<http://microsoft.com/webservices/>":

C#

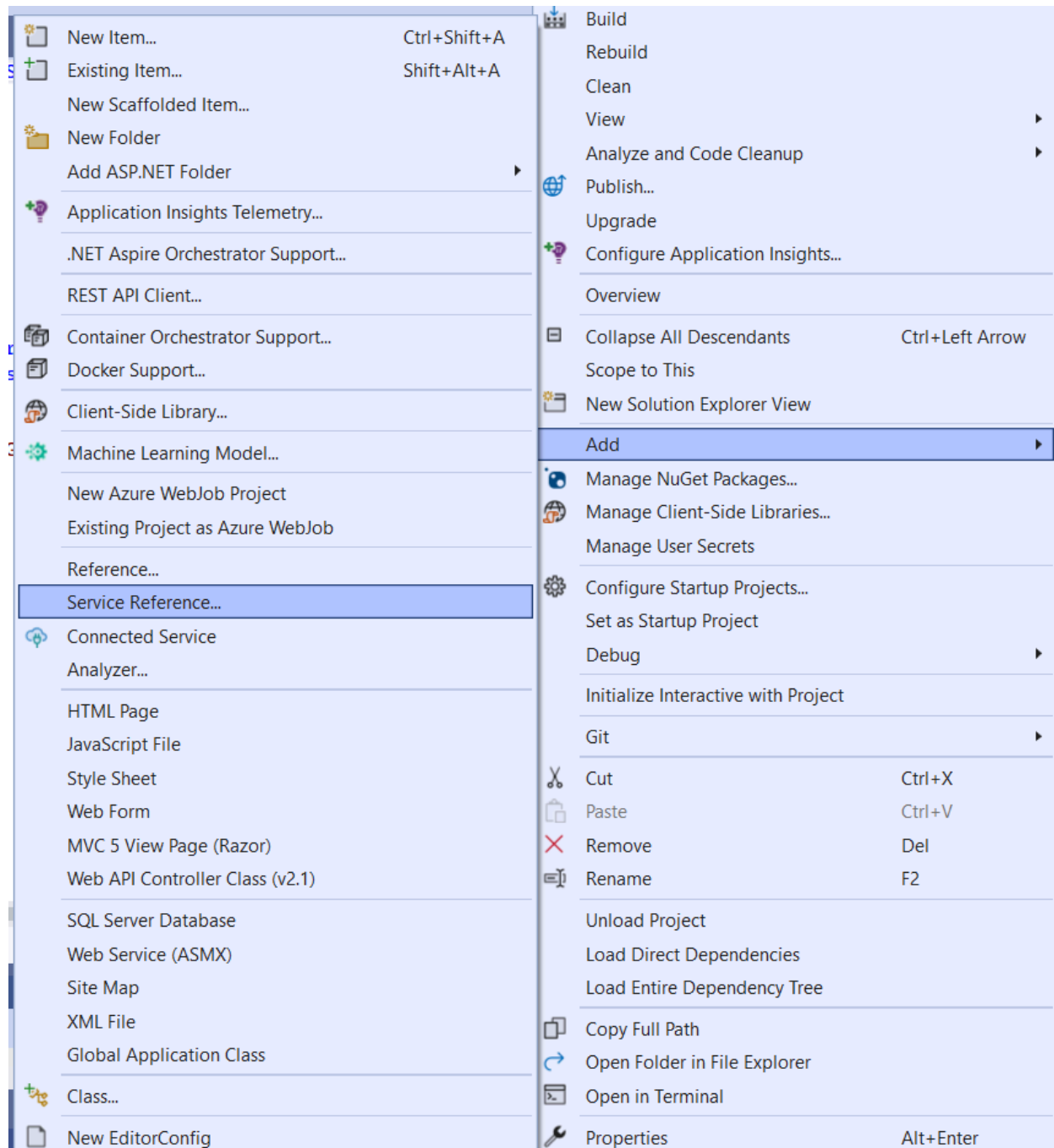
```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

**Step-4: Then just Create another WCFService\_Consumer application and by right clicking on that application you will get below interface just click on Service References:**



**Step-5: You will get the below interface just click on advanced option**

Add Service Reference?×

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

Go

Discover

Services:

Operations:

Namespace:

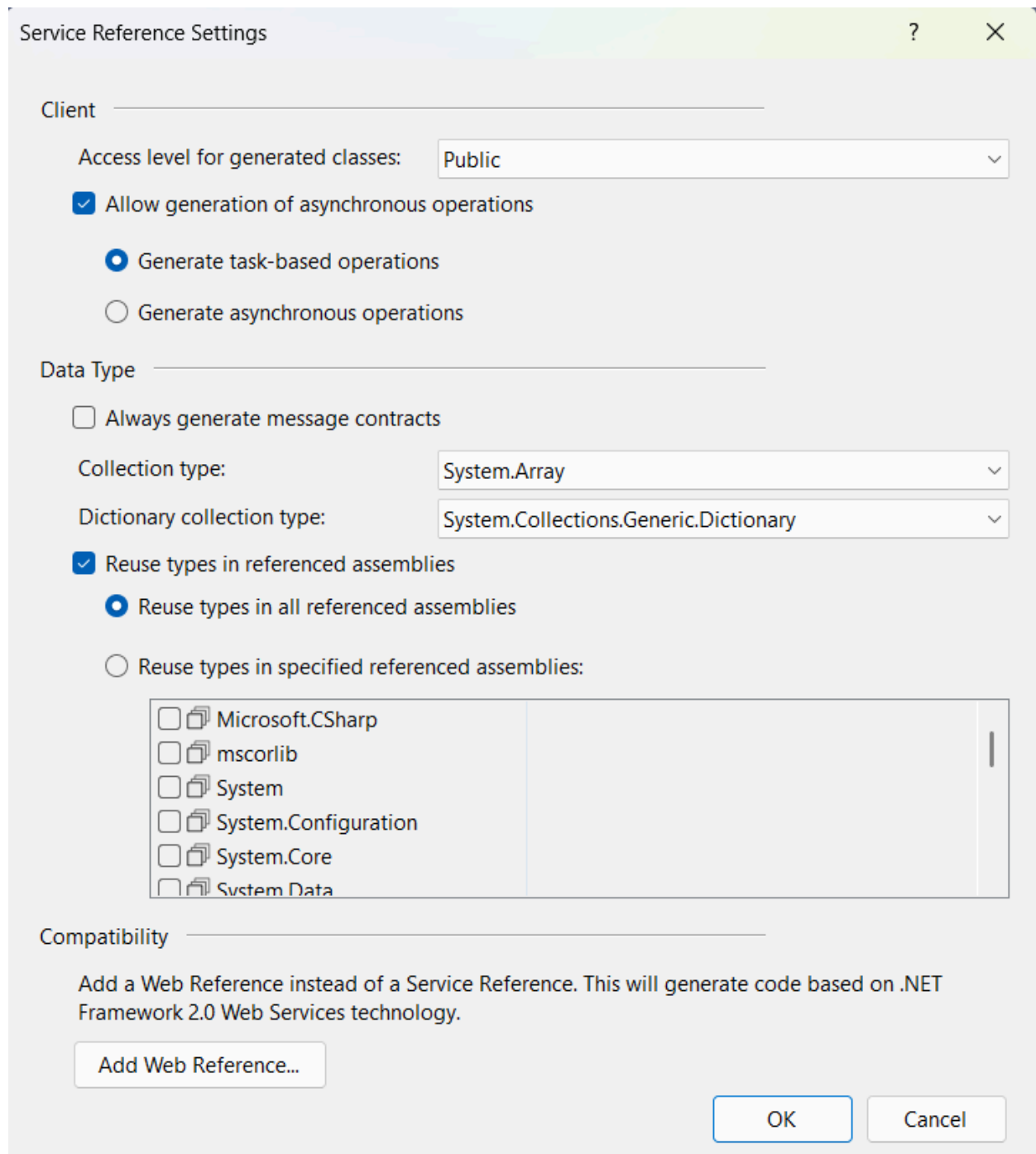
ServiceReference2

Advanced...

OK

Cancel

**Step-6 : After that u will get below interface just click on “Add Web Referene”**



The image shows a 'Service Reference Settings' dialog box with a title bar containing a question mark and a close button. The dialog is divided into three main sections: 'Client', 'Data Type', and 'Compatibility'. The 'Client' section has a text field for 'Client' and a dropdown for 'Access level for generated classes' set to 'Public'. It also features a checked checkbox for 'Allow generation of asynchronous operations' and two radio buttons: 'Generate task-based operations' (selected) and 'Generate asynchronous operations'. The 'Data Type' section includes a checkbox for 'Always generate message contracts' (unchecked), a dropdown for 'Collection type' set to 'System.Array', and another dropdown for 'Dictionary collection type' set to 'System.Collections.Generic.Dictionary'. It also has a checked checkbox for 'Reuse types in referenced assemblies' and two radio buttons: 'Reuse types in all referenced assemblies' (selected) and 'Reuse types in specified referenced assemblies:'. Below the second radio button is a list box containing several assemblies with checkboxes: 'Microsoft.CSharp', 'mscorlib', 'System', 'System.Configuration', 'System.Core', and 'System Data'. The 'Compatibility' section contains a text label: 'Add a Web Reference instead of a Service Reference. This will generate code based on .NET Framework 2.0 Web Services technology.' At the bottom, there is a button labeled 'Add Web Reference...', and at the bottom right, there are 'OK' and 'Cancel' buttons.

Service Reference Settings

Client

Access level for generated classes: Public

☒ Allow generation of asynchronous operations

☒ Generate task-based operations

☐ Generate asynchronous operations

Data Type

☐ Always generate message contracts

Collection type: System.Array

Dictionary collection type: System.Collections.Generic.Dictionary

☒ Reuse types in referenced assemblies

☒ Reuse types in all referenced assemblies

☐ Reuse types in specified referenced assemblies:

- ☐ Microsoft.CSharp
- ☐ mscorlib
- ☐ System
- ☐ System.Configuration
- ☐ System.Core
- ☐ System Data

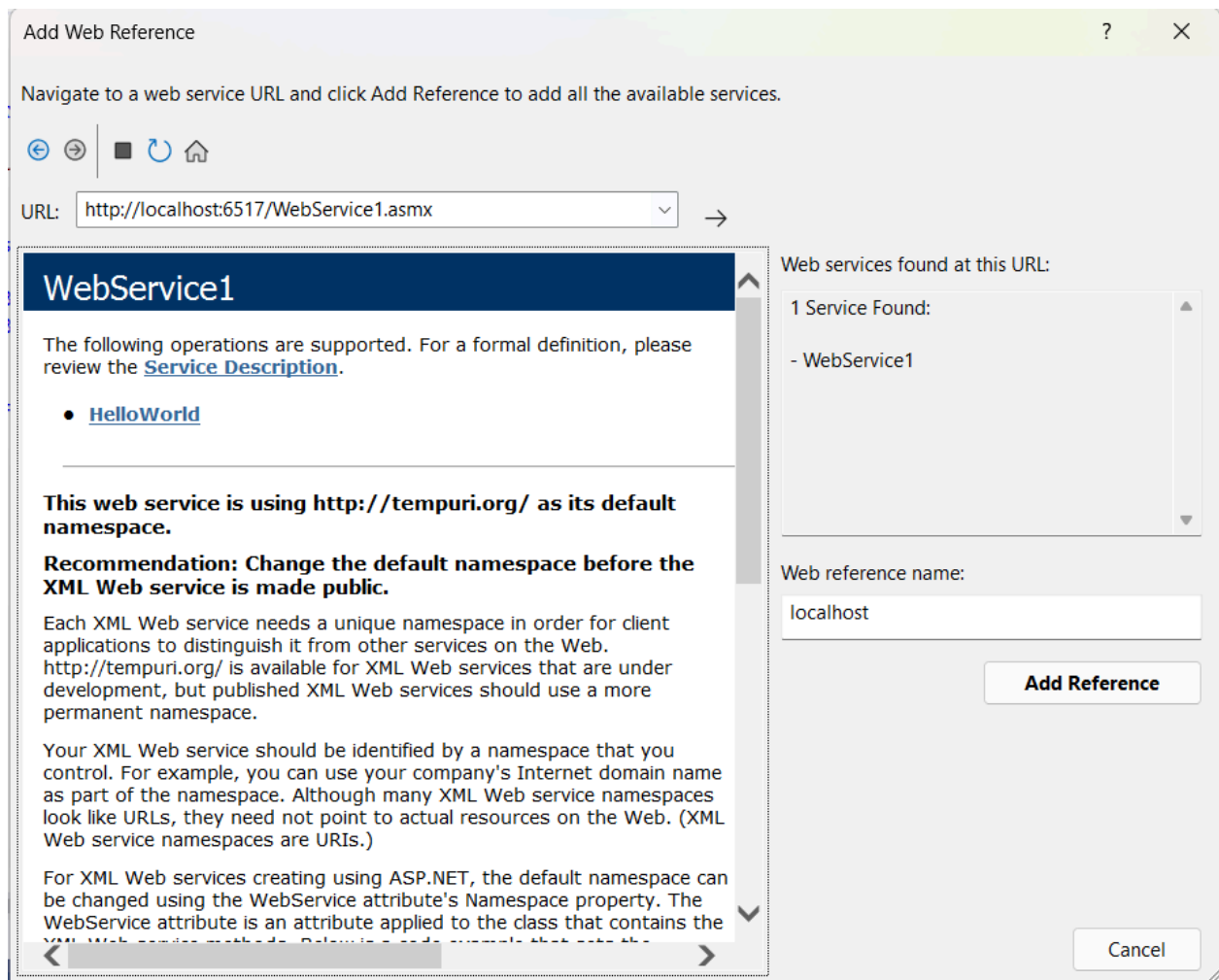
Compatibility

Add a Web Reference instead of a Service Reference. This will generate code based on .NET Framework 2.0 Web Services technology.

Add Web Reference...

OK Cancel

**Step-7: You will get the below interface, Just paste the url of the asmx file of WebService\_Producer's page and click on arrow after that just click on "Add Reference"**



**Step-8: After doing this just add web forms and add logic.**