Implementing a Service Objects Web Service into a C# Sample Form

Introduction

This guide provides a step by step on how to implement a Service Objects DOTS web service into a simple web form using C# .NET. This project will be using the SOAP implementation of Address Validation 3 but will be similar for any SOAP implementation of a Service Objects web service. This guide will be very basic and should help serve as a foundation to integrating one of Service Object's web services into your C#.NET application.

What to Expect:

We will demonstrate the implementation in two parts. The first part will demonstrate the process of calling and displaying the results from a Service Objects web service and the second part will include a more applicable example of how to integrate some useful business logic around the returned validated data from the Service Objects web service.

Audience

Any developer(beginner to expert), that is interested in integrating a DOTS web service into C# Web form.

Requirements

- Visual Studio – *VS 2015 was used in this example, but the steps should be relatively similar for most versions of Visual Studio*
- Basic Working Knowledge of C# and Object Oriented Programming
- A working License Key(trial or production) from Service Objects. Click here to obtain a free trial key for the service you are most interested in.

Contents

Implementing a Service Objects Web Service into a C# Sample Form

Setting Up Your Visual Studio Project

Launch Visual Studio and create a new project. Under the templates section select C#, then select Web and select ASP.NET Web Application for the template as shown in Figure 1. Select an appropriate location and name for the project and solution. Click Ok.

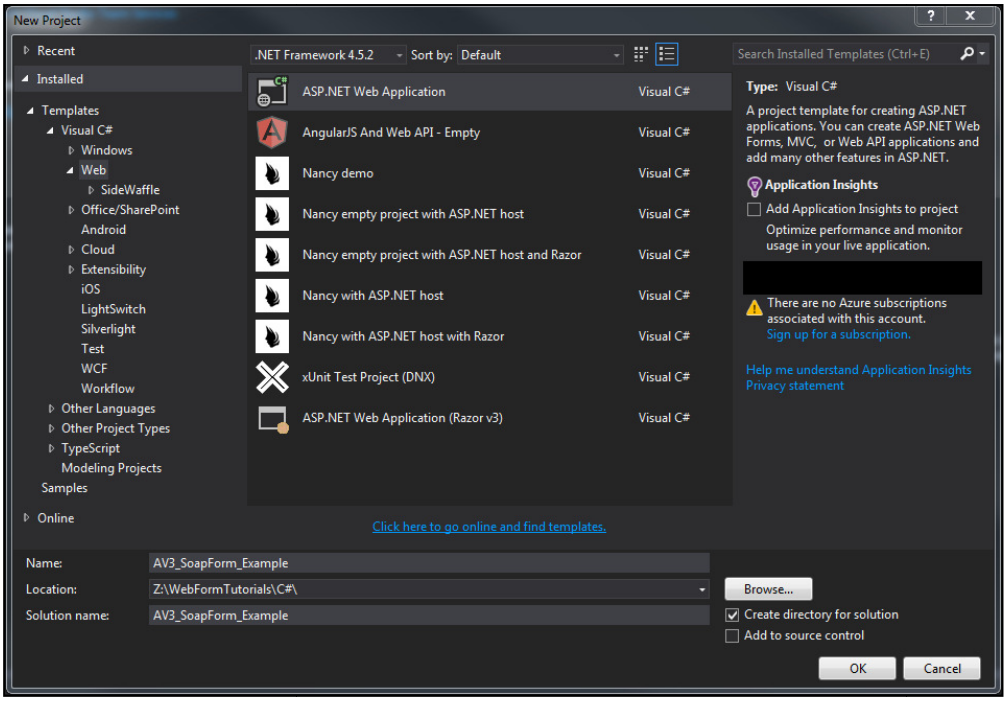

Figure 1

On the next screen, be sure to select “Empty” for the template type. Click ok. Your blank web application has been created. Let’s move onto adding the web page that the user will use to validate their address information.

Implementing a Service Objects Web Service into a C# Sample Form

Creating and Designing Your Web Form

Now that you have successfully created your web application, we need to add a web page to your project. Right click the project solution, go down to "Add" and then highlight and select the "Web Form" option. Other options can work but in our scenario this is what we chose. The page can be seen in figure 2. A dialogue box will appear and ask for a name for your web form. Select and type in an appropriate name. In this example we will call the form "AV3Form". Not very creative but it will get the job done.

After creating the web form you will have an empty aspx page. Click on it in the solution explorer and you should see some basic HTML elements as show in figure 2.

Figure 2

We are going to create an input form that user will use to input an address that will be validated. This form will represent a basic data entry form. We will do this by creating a simple two column table that will hold our textboxes and field labels. This input form will allow us to pass those particular variables to the code behind of this web form. The contents for this input form can be found in the ASPXPage_Contents.txt file that is included with this tutorial.

For this example, we will add all the textboxes that will correspond to the different inputs that the GetBestMatches operation takes. We will be adding a ***TextBox*** for BusinessName, Address1, Address2, City, State, and Zip Code. We will also add some other asp elements that will be used by our code to display results to the user and to show an example integration of a Service Objects web application.

In the input form code, the button ***CallWebServiceBtn*** will be used to trigger an event that will make a call to our web service. Take note of the name that you have given it in the ***OnClick*** field as we will need this for the code behind for this page.

The asp items Label and GridView will be used to display information about the results returned. The label ***resultsLabel*** will be used to provide a heading for the output values that the web service will return. The GridView named ***outputGrid*** will be used to display the output values. The ***Radiobuttonlist*** and ***ChooseAddressButton*** will be used to implement some more sophisticated functionality than simply displaying the results from the service to the user. Now that we have everything we need added to the HTML page let's get to looking at the code behind for this page to make it call the Address Validation 3 web service.

Implementing a Service Objects Web Service into a C# Sample Form

Calling a Service Objects Web Service

After selecting the .cs file associated with the aspx page that we made previously, you will note that the file is relatively empty. We created a button click method that corresponds with onClick field in the ASP element we made previously in the aspx page and added some code to initialize our webpage in the ***Page_Load*** method. The code should be self explanatory. Copy and paste the code in the ***AV3Soap_Contents.txt*** file that is included with this project into your program.

Before we add any code to our ***SubmitBtn_Click*** method we need to add a web service reference to the project so that we can initialize the objects that will be making the SOAP call to the Service Objects Web Service. Add the service reference by right-clicking the References item in the solution explorer, highlight and select "Add Service Reference". On the following screen add in the WSDL location of the web service you would like to use, click "Go" then give it an appropriate name for the Namespace. In this example we paste in the WSDL URL for Address Validation 3 and name the reference AV3. Click OK to create the service reference. Both of these steps can be seen in figure 3:

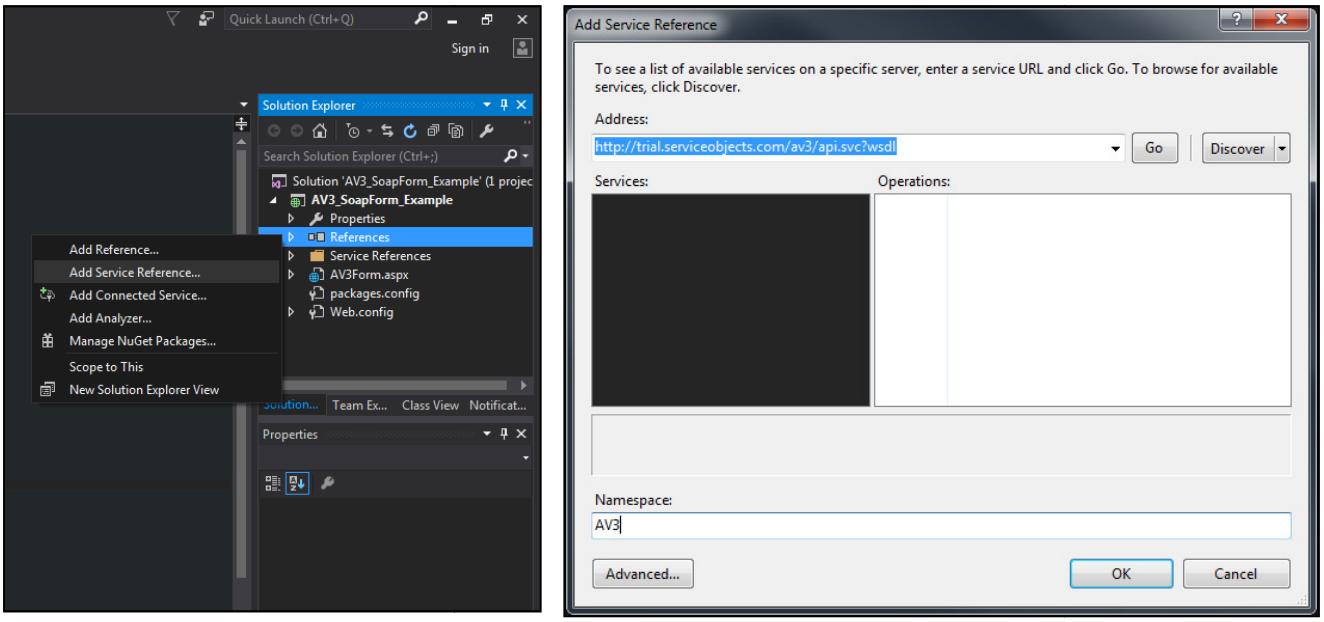

Figure 3

Now that our service reference is set up, we can add the code to make a call to our web service. In Code Snippet#3 below, we instantiate the objects ws and wsbackup with the AddressValidation3Client.

> *In this code both ws and wsbackup are created from the same service reference but once a production key is purchased there should be 2 service references added to the project: one for the primary Service Objects endpoint and one for the backup:*
>
> https://ws.serviceobjects.com/av3/api.svc?wsdl
>
> https://wsbackup.serviceobjects.com/av3/api.svc?wsdl

Implementing a Service Objects Web Service into a C# Sample Form

The GetBestMatches method that we call here will take all the input values that user enters and pass them to the SOAP call and encapsulate it with a try catch block to catch any exceptions or errors that may occur.

```
//Add the backup service reference to enable failover and to access the backup endpoint
AV3.AddressValidation3Client ws = new AV3.AddressValidation3Client();
AV3.AddressValidation3Client wsbackup = new AV3.AddressValidation3Client();

string licenseKey = ConfigurationManager.AppSettings["LicenseKey"];

AV3response.wsresponse = ws.GetBestMatches(BusinessName.Text, Address.Text, Address2.Text, City.Text, State.Text, PostalCode.Text, licenseKey);
```

Figure 4: Soap Client Instantiation

Another thing to note about the above code is that the LicenseKey that is passed to the service is not exposed in the main section of our code. In order to best protect and hide the LicenseKey, it is recommended that key is stored in the app.config or web.config file of the project that you using.

> **Note: For added security you can even go a step forward by encrypting the key in the web.config and decrypting it in code when it is needed. There are many tutorials on the web that can show you how to do that if you are interested. For most purposes simply keeping the key as is in the web.config is enough.*

Implementing a Service Objects Web Service into a C# Sample Form

Failover Configuration

Now that we have the code to call the Address Validation 3 service, we need some code that will check and process the response from the web service. We highly recommend implementing a failover call into your code. This is done by checking if the returned response is null or has an error with a TypeCode of 3; which would indicate a fatal error. In the event that either of these instances occurs, the code will call the backup service. We strongly recommend implementing this fail over configuration into your project. With failover enabled you can ensure that your application will function as normal in the event that the production endpoint is unavailable.

```
if (AV3response.wsresponse == null || AV3response.wsresponse.Error != null && AV3response.wsresponse.Error.TypeCode == 3)
{
    AV3response.wsresponse = wsbackup.GetBestMatches(BusinessName.Text, Address.Text, Address2.Text, City.Text, State.Text, PostalCode.Text, licenseKey);
}
```

Figure 5: Example Failover Configuration

As noted in the screen shot above, if the code detects a null response or TypeCode of 3 it will call the wsbackup endpoint and attempt to get a valid response. After this, the code checks if there was an error response returned; if so, then it will call the ProcessErrorResponse method. If no error response is detected then it will pass the response into the ProcessValidResponse method which we will describe in more detail in the next section.

Implementing a Service Objects Web Service into a C# Sample Form

Display Results to the User

This section will illustrate how to simply display the results from the service to user. If you are interested in seeing possible applications or business logic that Address Validation 3 may be used for, skip to the next section as it will present some possible ways that the AV3 service can be used in your application.

Now we need to define the methods to process the response from the web service so that it may be output to the user. In this example we have 2 methods: ProcessValidResponse and ProcessErrorResponse. Each of these methods will process their respective data from the web service into a DataTable and then bind that Data to our output Grid.

```
1 reference
protected void ProcessValidResponse(AV3.BestMatchesResponse validResponse)
{
    DataTable dtResponse = new DataTable();
    dtResponse.Columns.Add(new DataColumn("Output", typeof(string)));
    dtResponse.Columns.Add(new DataColumn("Values", typeof(string)));

    /* setting visible the table to hold the valid address
        * details*/
    dtResponse.Rows.Add("Address", validResponse.Addresses[0].Address1);
    dtResponse.Rows.Add("City", validResponse.Addresses[0].City);
    dtResponse.Rows.Add("State", validResponse.Addresses[0].State);
    dtResponse.Rows.Add("Zip", validResponse.Addresses[0].Zip);
    dtResponse.Rows.Add("Address2", validResponse.Addresses[0].Address2);
    dtResponse.Rows.Add("BarcodeDigits", validResponse.Addresses[0].BarcodeDigits);
    dtResponse.Rows.Add("CarrierRoute", validResponse.Addresses[0].CarrierRoute);
    dtResponse.Rows.Add("CongressCode", validResponse.Addresses[0].CongressCode);
    dtResponse.Rows.Add("CountyCode", validResponse.Addresses[0].CountyCode);
    dtResponse.Rows.Add("CountyName", validResponse.Addresses[0].CountyName);
    dtResponse.Rows.Add("FragmentHouse", validResponse.Addresses[0].FragmentHouse);
    dtResponse.Rows.Add("FragmentPreDir", validResponse.Addresses[0].FragmentPreDir);
    dtResponse.Rows.Add("FragmentStreet", validResponse.Addresses[0].FragmentStreet);
    dtResponse.Rows.Add("FragmentSuffix", validResponse.Addresses[0].FragmentSuffix);
    dtResponse.Rows.Add("FragmentPostDir", validResponse.Addresses[0].FragmentPostDir);
    dtResponse.Rows.Add("FragmentUnit", validResponse.Addresses[0].FragmentUnit);
    dtResponse.Rows.Add("Fragment", validResponse.Addresses[0].Fragment);
    dtResponse.Rows.Add("FragmentPMBPrefix", validResponse.Addresses[0].FragmentPMBPrefix);
    dtResponse.Rows.Add("FragmentPMBNumber", validResponse.Addresses[0].FragmentPMBNumber);
    dtResponse.Rows.Add("DPV", validResponse.Addresses[0].DPV);
    dtResponse.Rows.Add("DPVDesc", validResponse.Addresses[0].DPVDesc);
    dtResponse.Rows.Add("DPVNotes", validResponse.Addresses[0].DPVNotes);
    dtResponse.Rows.Add("DPVNotesDesc", validResponse.Addresses[0].DPVNotesDesc);
    dtResponse.Rows.Add("Corrections", validResponse.Addresses[0].Corrections);
    dtResponse.Rows.Add("CorrectionsDesc", validResponse.Addresses[0].CorrectionsDesc);
    dtResponse.Rows.Add("IsCASS", validResponse.IsCASS);

    outputGrid.Visible = true;
    resultsLabel.Visible = true;
    outputGrid.DataSource = new DataView(dtResponse);
    outputGrid.DataBind();
}
```

Figure 6: ProcessValidResponse Method

Implementing a Service Objects Web Service into a C# Sample Form

```
1 reference
protected void ProcessErrorResponse(AV3.AV3Error errorResponse)
{
    DataTable dtError = new DataTable();
    dtError.Columns.Add(new DataColumn("Output", typeof(string)));
    dtError.Columns.Add(new DataColumn("Values", typeof(string)));

    dtError.Rows.Add("Type", errorResponse.Type);
    dtError.Rows.Add("TypeCode", errorResponse.TypeCode);
    dtError.Rows.Add("Desc", errorResponse.Desc);
    dtError.Rows.Add("Desc", errorResponse.DescCode);

    outputGrid.Visible = true;
    resultsLabel.Visible = true;
    outputGrid.DataSource = new DataView(dtError);
    outputGrid.DataBind();
}
```

Figure 7: ProcessErrorResponse Method

One thing to note about the Address Validation 3 and about other Service Objects web service responses, is that some may return an array of responses for any given input. In this example, we are only displaying the first value in the array of addresses that is returned. It may suit your business needs to use or account for the other values returned. An example of how it can be handled comes in the following section.

Our code is ready to run. After running the solution we will see that we have a functioning web form that will take our address and return a validated response. In the below screen shot, we have used the Service Objects address as an example input to our web service. If you input the same values with your respective license key you should see the same results.

Address Validation 3 US - SOAP Example

Business Name:	Service Objects
Address:	27 E Cota St
Address2:	Suite 500
City:	Santa Barbara
State:	CALIFORNIA
Postal Code:	93101

Get Results

AV3 Results

Output	Values
Address	27 E Cota St Ste 500
City	Santa Barbara
State	CA
Zip	93101-7602
Address2	Suite 500
BarcodeDigits	931017602254
CarrierRoute	C006
CongressCode	24
CountyCode	083
CountyName	Santa Barbara
FragmentHouse	27
FragmentPreDir	E
FragmentStreet	Cota
FragmentSuffix	St
FragmentPostDir	
FragmentUnit	Ste
Fragment	500
FragmentPMBPrefix	
FragmentPMBNumber	
DPV	1
DPVDesc	Yes, the input record is a valid mailing address
DPVNotes	24,26,28,39
DPVNotesDesc	SuiteLink did not find Suite or Unit data to append to the address,The input address matched the ZIP+4 record,The input address matched the DPV record,Highrise apartment/office building address
Corrections	
CorrectionsDesc	
IsCASS	True

Figure 8: Valid Response Example

Implementing a Service Objects Web Service into a C# Sample Form

Implementing Business Logic into Your Web Application.

Simply displaying the results to the end user isn't a necessarily a good use of the Service Objects web service, so in this section we will go over an example of what it may look like to integrate some business logic around the results that web service retruns. For example, let's presume that a user will be entering address to complete a purchase. For this example we will want to ensure that the user's entered address will be standardized and corrected version of the address to ensure that whatever is delivered to the client will be sent to the correct address. The code is set up will also allow the client to choose the best addresses in the event that multiple addresses are returned from the service.

In this section we will add a method to our project called "CheckDeliverability". This method will check the response from the DPV value from the AV3 web service and relay to the user whether or not the address is valid. The updated code for the codebehind file can be found in the ***Updated_AV3Soap_Contents.txt*** file included with this tutorial.

This new code will integrate some logic into our application that will allow us to use the response from the web service to ensure the accuracy of a user entered address data.

The first check the code makes is to check for the existence of the Error Object in the response. If an error object is found, this code will display the resulting error information to the user in order to adjust their input accordingly.

If the code finds that error object is empty it will move onto the next set of code that will display the standardized addresses that was returned from the service along with information regarding the deliverability of that address.

Now, we programmatically add a **ListItem** to the radio button list. The first radio button will be the original user entered address and will have a list value of "original". The rest of the radio buttons will be determined by the DPV score result of the original address.

One thing to note about the GetBestMatches operation is that it has the ability to produce multiple addresses based on a single input. We will see an example of an address that will produce multiple results later. But for now, we cycle through all the addresses from the service. Within the for loop, there will be 2 conditions that will check for different values of the DPV field that is returned from the web service. In each of these statements there will be a code that will add the array index as the value of the **ListItem**.

The first condition will check to see if the DPV field is equal to 1. This means, that an address is considered deliverable by the USPS. If this happens, the standardized address will be added as a List Item and displayed to the user with a message indicating that it is the corrected address. Below is a screen shot of this instance.

Implementing a Service Objects Web Service into a C# Sample Form

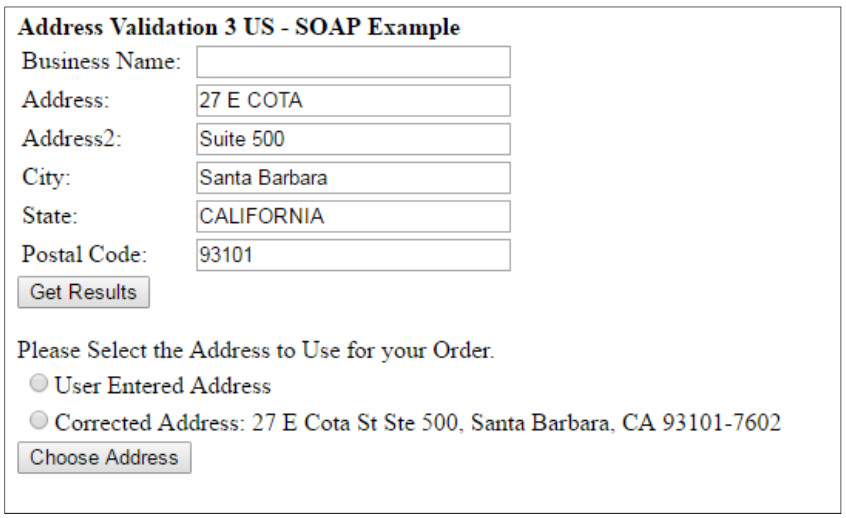

Figure 9

The last check will be for a DPV of 3 or 4. A DPV indicates of 3 indicates that the given house number for an address is valid but the given apartment number is not valid. A DPV of 4 indicates that the address is missing a valid apartment number. For this example we will concatenate them into a single check as they will have a similar message. If you would like more specificity in your program, separate checks can be easily implemented along with separate status messages. An Example of this is noted in the screen shot below.

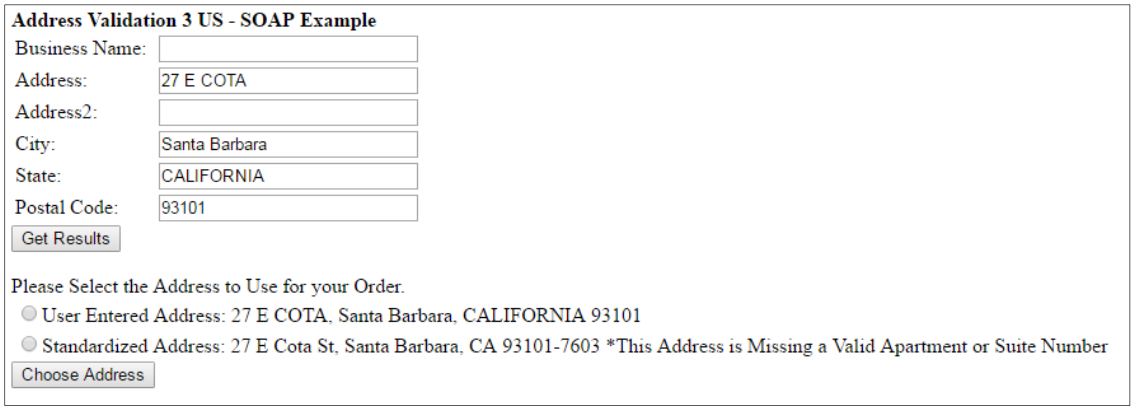

Figure 10

After the for loop has completed, the code will check to see if the ***ChooseAddressButton*** is visible or not. If it is visible, then by the previous logic set up in our code, this would indicate that the AV3 service returned a valid mailing address to the user. If it is not visible, then there was no deliverable address returned by the service. In this event, the code will change the ***resultsLabel*** to indicate to the user to enter a different address and try it again. This can be seen in figure 11.

Implementing a Service Objects Web Service into a C# Sample Form

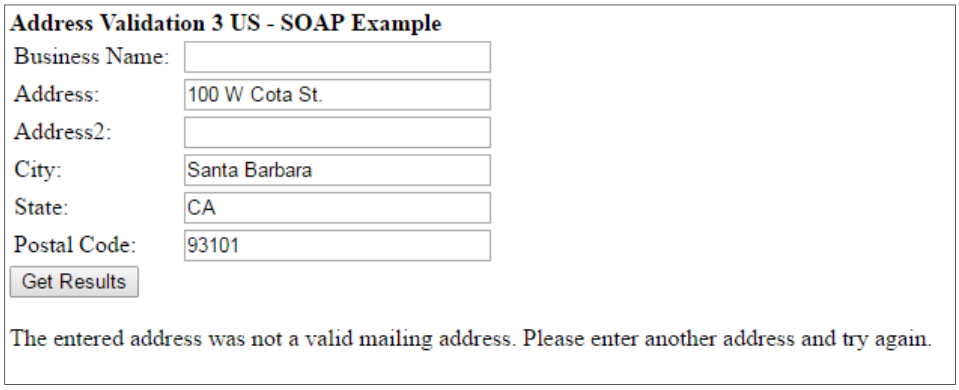

Figure 11

Now that we have the logic set up to filter through our addresses, we will want to implement a click event for the user to select the address of their choice. Below is the click event method that will allow user to confirm their choice of address and pass it along for further processing. In this example we will simply be displaying a message to the user that their chosen address was selected and used.

If the user does not a select an address, the **resultsLabel** will ask the user to try again, and no address will be chosen. If the user chose the original address, the user entered address will be used to display to the user. If the user selected one of the corrected addresses, the array index that was saved earlier will be used to access the correct address from the array of addresses returned from the web service..

Implementing a Service Objects Web Service into a C# Sample Form

Conclusion

That is it for our tutorial on how to create a web form that uses a SOAP call in C#. If you have any questions please email support@serviceobjects.com and we would gladly answer any questions you may have.