# LinqToServiceNow End to End Tutorial

Contents

[LinqToServiceNow End to End Tutorial 1](#_Toc376166321)

[Why use LinqToServiceNow 1](#_Toc376166322)

[How to use LinqToServiceNow to consume a ServiceNow Web Service 1](#_Toc376166323)

[Requirements 1](#_Toc376166324)

[Using a Service Reference 1](#_Toc376166325)

[Using a Web Reference 1](#_Toc376166326)

[Writing Linq Queries 1](#_Toc376166327)

[Source Code - Download 1](#_Toc376166328)

# Why use LinqToServiceNow

Currently in order to query ServiceNow in a .NET application, you have two choices: Glide classes with JavaScript or SOAP web services. Each has its strengths and weaknesses and neither really fits in with common .NET design patterns.

Glide classes are Java classes that can be used in JavaScript as if they were native JavaScript classes. There are several Glide classes on both the server-side and the client-side that a developer will need to use in a solution. A few of the classes are: GlideRecord, GlideElement, GlideAggregate, GlideDataTime, and GlideAjax. A GlideRecord is a server-side object that contains records from a single table. A GlideElement is a server-side object that contains fields of the current GlideRecord. GlideAggregate is a class that is used to perform database aggregation queries such as COUNT, SUM, MIN, MAX, and AVG. GlideDateTime is used to perform date-time operations. GlideAjax is a client-side object that can execute server-side code from the client.

As you can see, there are many classes that you will have to deal with and learn in order to use Glide. The syntax for these classes is challenging as well. In order to build a query, a developer will have to call many methods and this can get very complicated as a query gets more complex.

The above syntax does not allow for complex queries that define several sets of logic blocks, for that, a developer using Glide will have to use an encoded query, which contains its own syntax that is not very well documented:

Notice the symbols used to divide the query in the code block above. As a query gets more complex, this string of data would need to be maintained.

Another challenge when using Glide is that you don’t have compile time type checking (although some might find that an advantage) and you need to have intimate knowledge of the data table you are querying in order to provide names of fields as strings to the necessary methods.

Glide works if you are building a web application, but doesn’t work as well if you are building a client application or a native mobile application where JavaScript is not your development language. For that you will need to use ServiceNow SOAP web service calls. These SOAP web services resemble the Glide design and suffer from many of the same challenges. You get compile time type checking, but the syntax doesn’t allow for queries of any complexity, for that you will have to use encoded query syntax just as you would with Glide.

The default syntax for getRecords doesn’t allow for queries of any complexity such as containing an OR operator. You will have to use the \_\_encoded\_query property of the getRecords object in order to include an OR operator in your query. When you start building the \_\_encoded\_query, you lose compile time type checking and you are required to know the names of all of the fields in the data table that you want to include in that query.

Another issue is the naming of the classes in the SOAP web services can be confusing: you have to create an object of type getRecords to set up a query that you then pass to a getRecords method on an instance of the web Service and you will then receive an array of records of type getRecordsResponseGetRecordsResult.

Neither the Glide library nor the SOAP web service allows you to limit the fields that are returned from a web service call. It is up to the developer to loop through the collection returned and decide which fields are desired and to perform any operations on them during the iteration.

Using LinqToServiceNow, a developer will be able to follow a repository pattern and query that repository using Linq and get all of the Visual Studio goodness that he/she is used to such as intellisense. LinqToServiceNow will translate your Linq queries into an encoded query that will be passed to the ServiceNow SOAP web service and return you an IEnumerable of objects that you can then use in your application. You can declare what you want returned and how you want it returned through Linq queries. Both lambda and query expressions are supported.

# How to use LinqToServiceNow to consume a ServiceNow Web Service

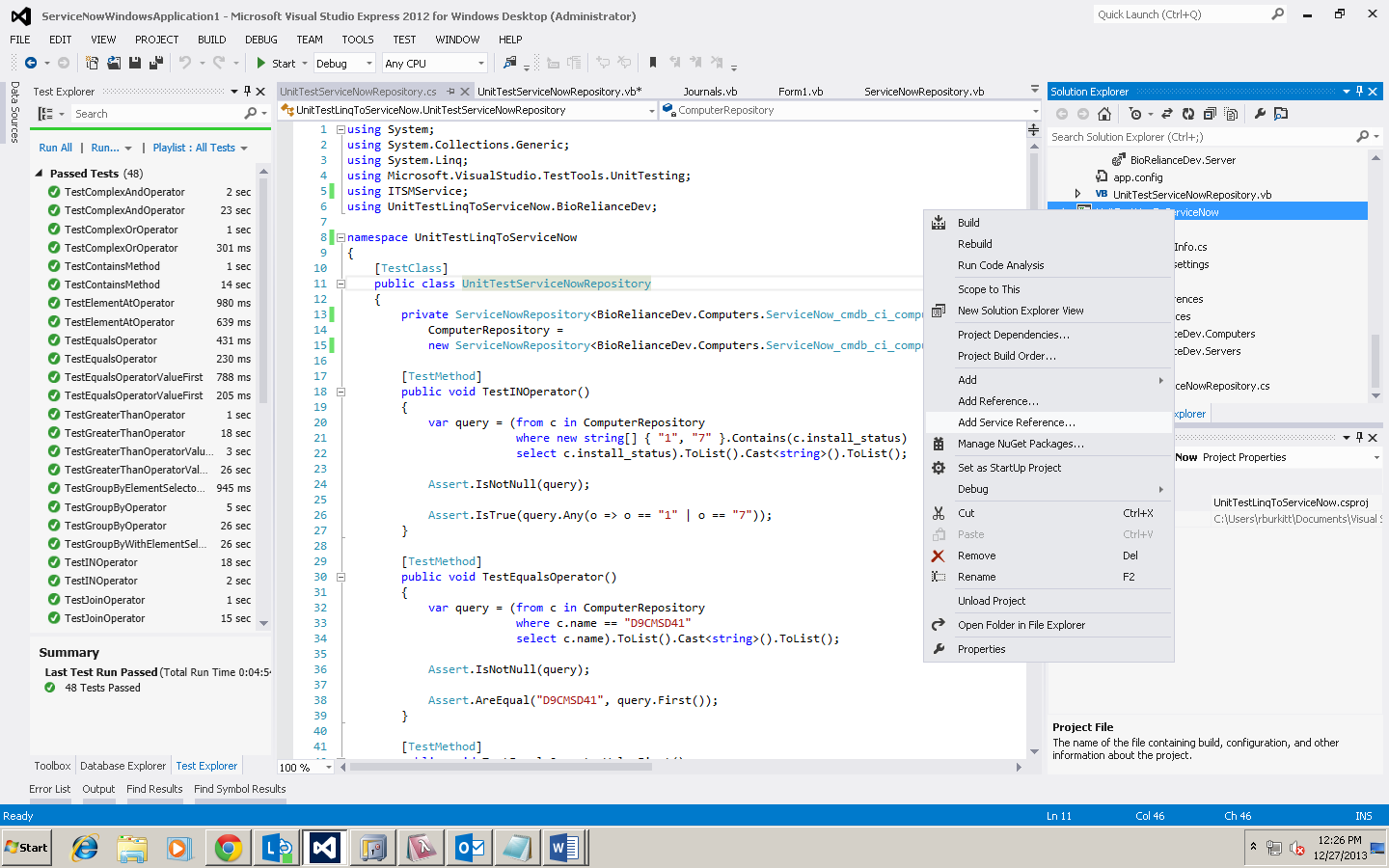
This tutorial will show you how to consume a ServiceNow web service using Visual Basic or C# .NET and the LinqToServiceNow Linq Provider.

### Requirements

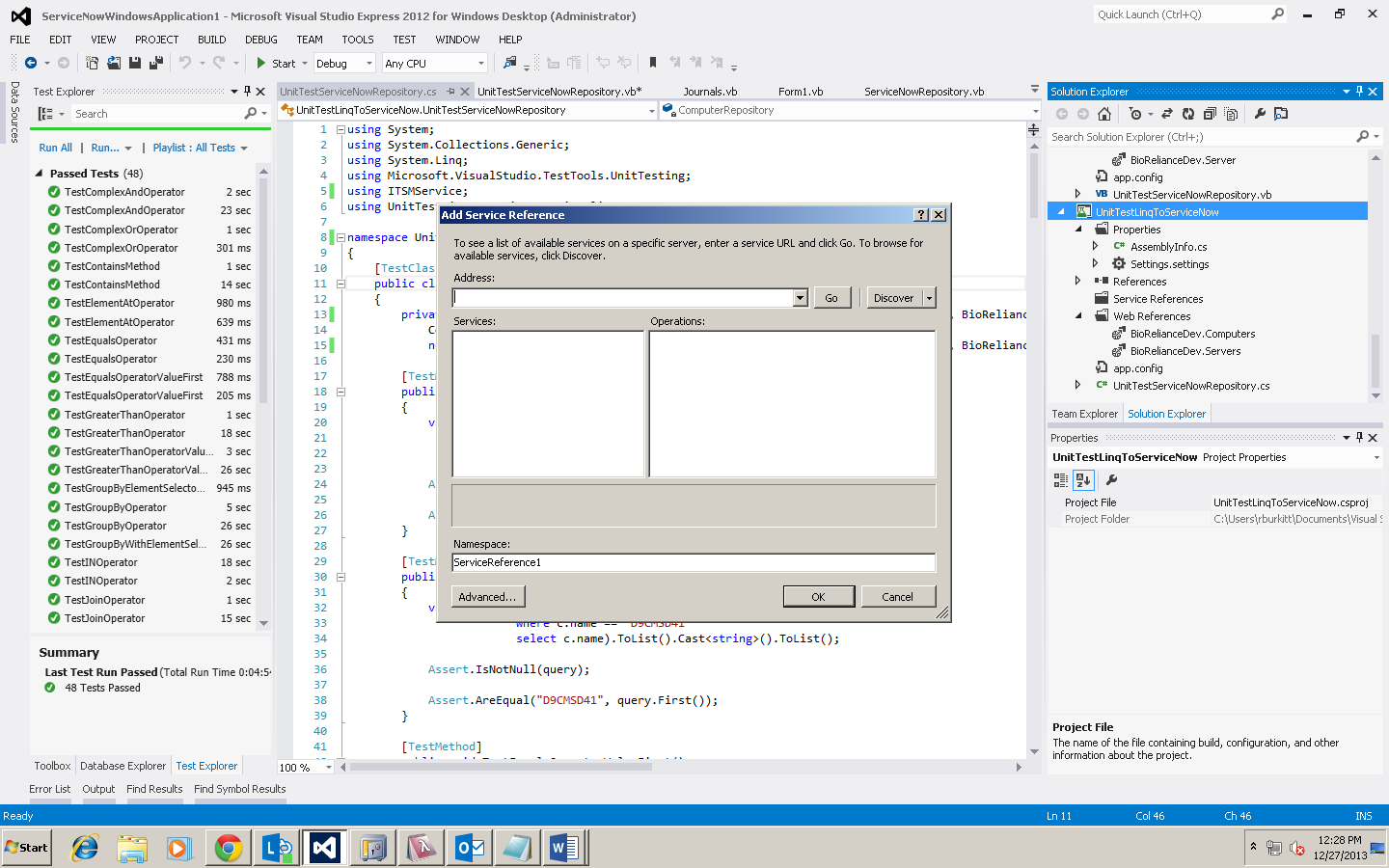
.NET 4.5 or later

### Using a Service Reference

Go to the Solutions Explorer and select Service References->Add Service Reference.



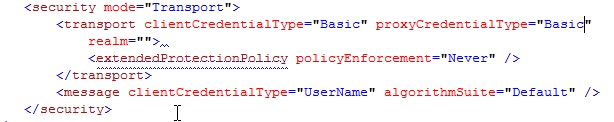
A wizard will appear asking for an address.



Use: https://<instance name>.service-now.com/cmdb\_ci\_computer.do?WSDL and name the service reference, 'Computer'.

Accept the defaults for the rest of the wizard.

Open the app.config file and change the Security mode to "Transport" and the clientCredentialType and proxyCredentialType to "Basic"



Then set the maxBufferPoolSize, maxBufferSize, maxReceivedMessageSize, as well as the maxDepth, maxStringContentLength and maxArrayLength.



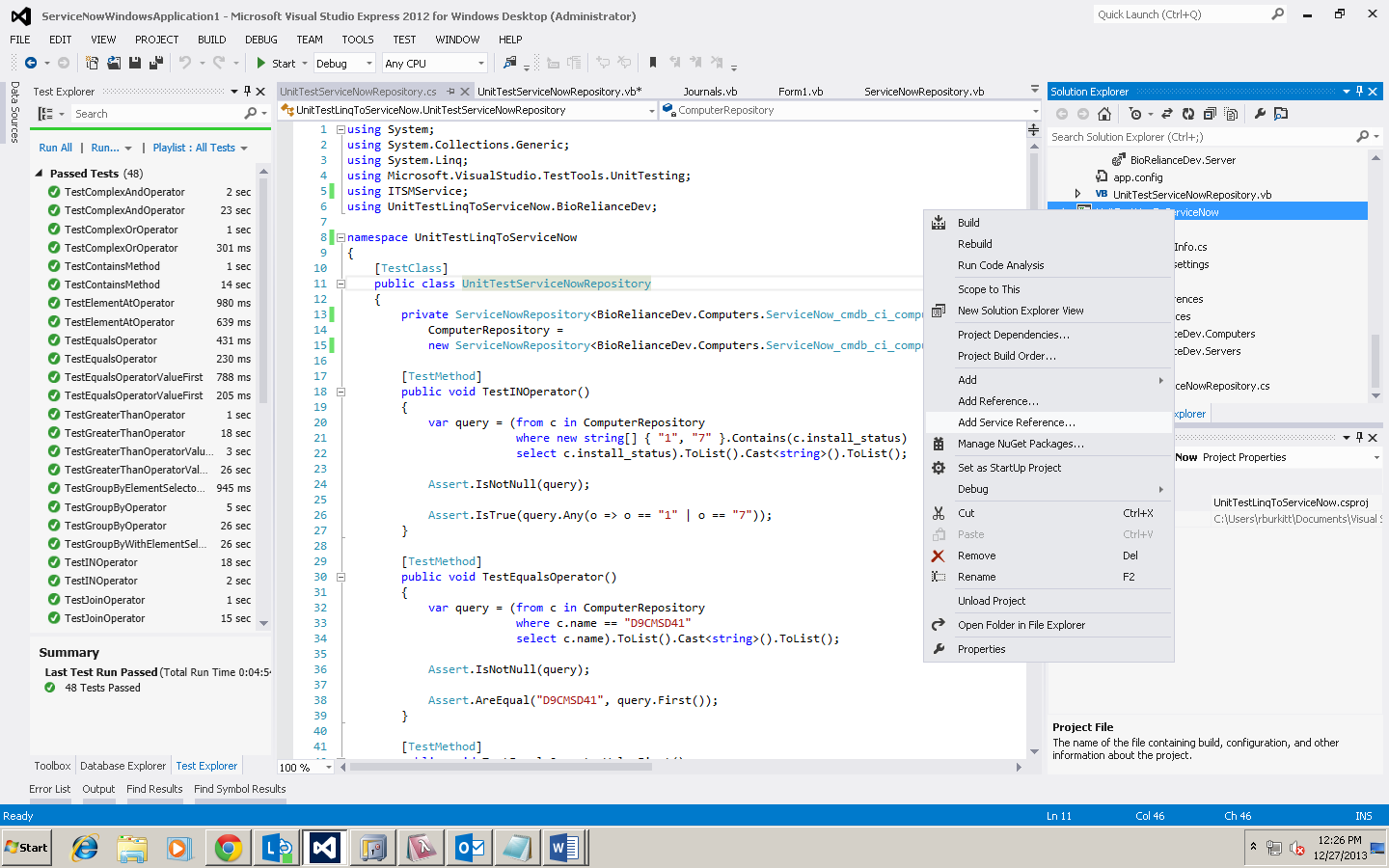
You can then declare a variable of ServiceNowRepository of the following types:



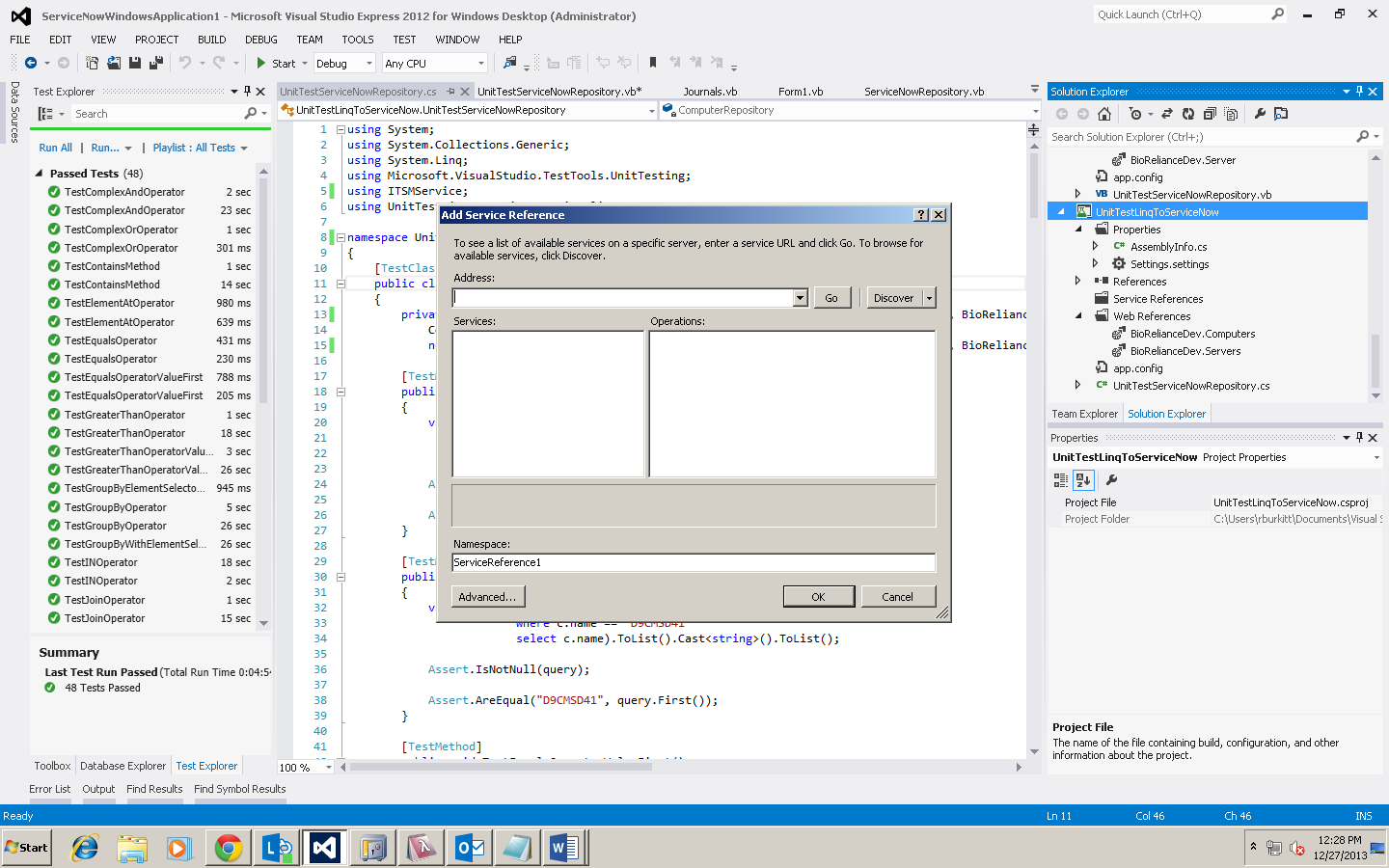


### Using a Web Reference

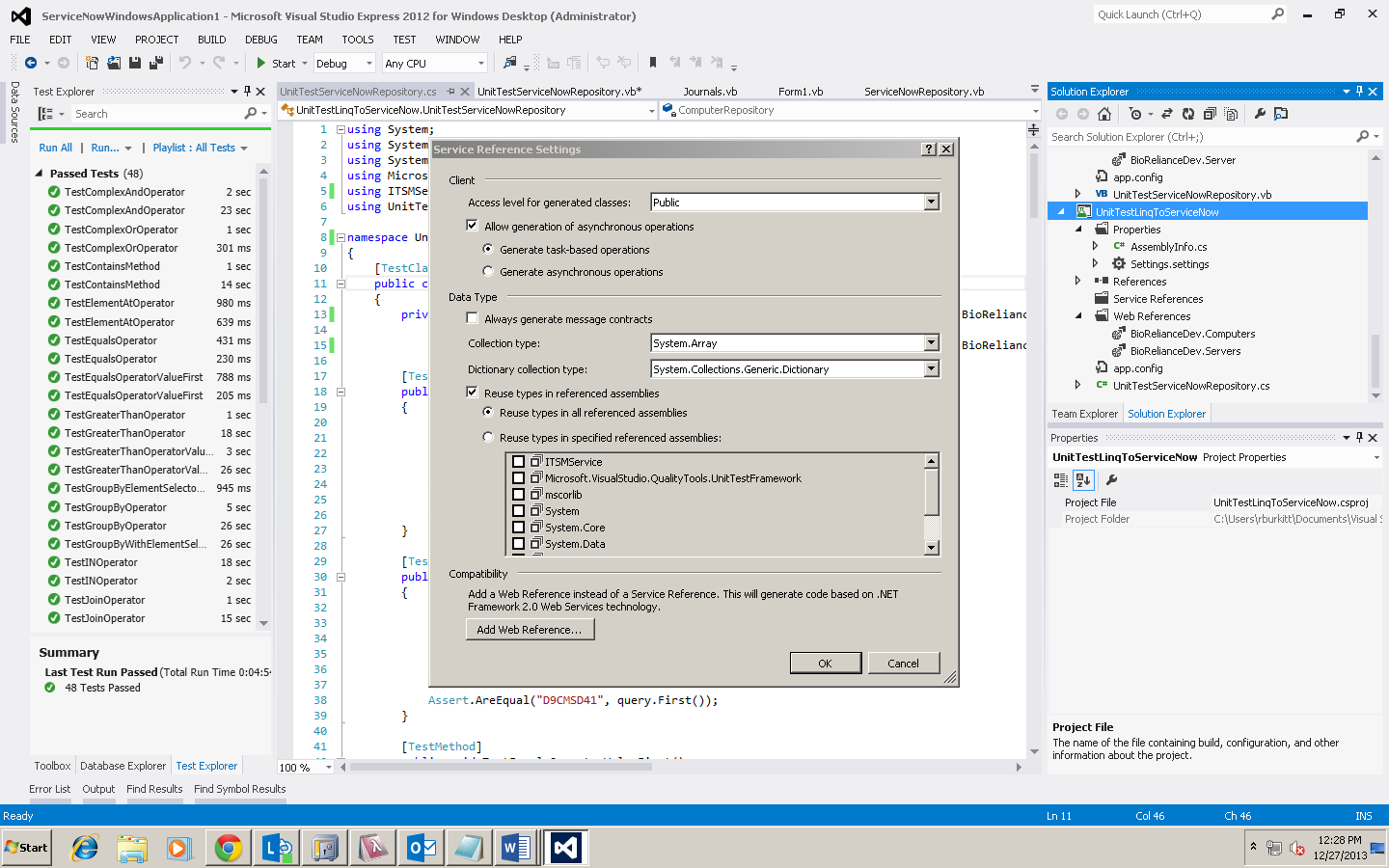
Go to the Solutions Explorer and select Service References->Add Service Reference.



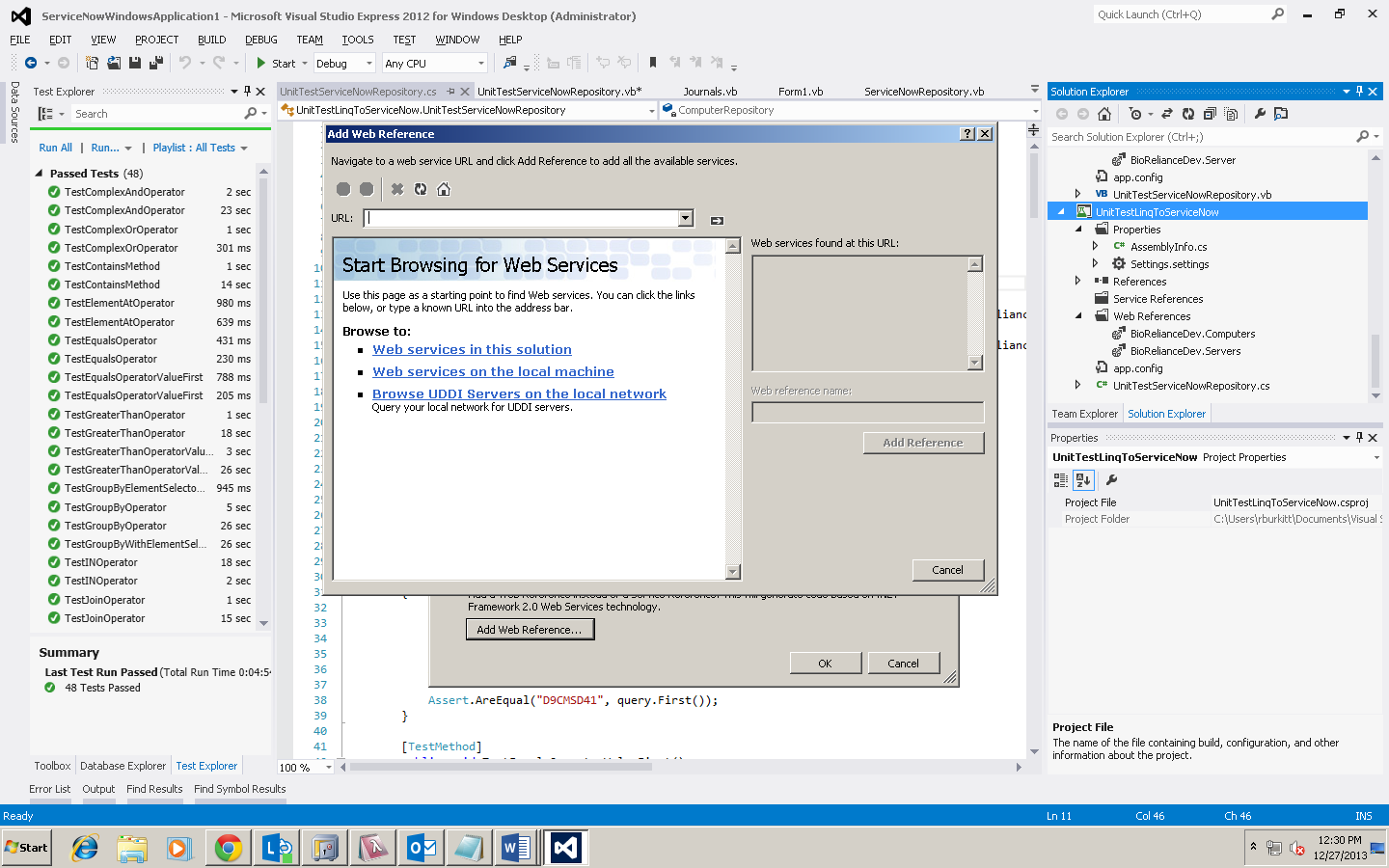
A wizard will appear.  At the bottom of the form, there is an Advanced button.



Click on it and click on the "Add Web Reference" button at the bottom of the new wizard page.



This will start the Web Reference wizard.



For the URL, use: https://<instance name>.service-now.com/cmdb\_ci\_computer.do?WSDL and name the web reference, 'Computer'.  Accept the defaults for the rest of the wizard.

You can then declare a variable of ServiceNowRepository of the following types:





### Writing Linq Queries

You can use most of the common Linq operators directly on the ServiceNowRepository instance.

* Select
* Where
* OrderBy
* ThenBy
* OrderByDescending
* ThenByDescending
* GroupBy
* Join
* Take
* TakeWhile
* Skip
* SkipWhile
* ElementAt
* IN \*\*
* Like \*
* Contains \*\*\*
* StartsWith \*\*\*
* EndsWith \*\*\*
* Not, !
* Or, |
* And, &
* ToArray
* ToList
* ToDictionary

\* Visual Basic .NET only

\*\* This is actually a SQL command that can be mimicked in LinqToServiceNow, see example below

\*\*\* These are string methods that can be called on individual fields inside of a query expression of lambda statement

Below are example of queries that you can write using the LinqToServiceNow Linq Provider:

Simple Linq query:





Linq query emulating SQL "IN" clause:





Linq query evaluating if a field contains a string of data:





Linq query using the NOT keyword:





Linq query using the OR keyword:





Linq query using the greater than keyword:





Linq query using the Take method:





Linq query using the OrderBy method:





Linq query using the OrderBy method with multiple operators:



Linq query using the Skip method:





Linq query using the ElementAt method:





Linq query using the Visual Basic Like operator:



Linq query using the GroupBy method:





Linq query using the GroupBy method and a selector operator:





Linq query using the Join method:





Linq queries with multiple where methods:





Linq queries with where method containing multiple sets:









Linq query with ToDictionary method:





Linq query with ToDictionary method and element selector:





Linq query using the SkipWhile method:





Linq query using the TakeWhile method:





## Source Code - Download

You can download and look at the source code for LinqToServiceNow:

GitHub address: <https://github.com/rburkitt/LinqToServiceNow>

GitHub page: <http://rburkitt.github.io/LinqToServiceNow/>