Dynamic Search with LINQ, the Entity Framework and Silverlight – Part 2

After my previous blog post, I realized that using SQL strings is not a great way to do things. Sometimes we start blogging too quick and then realize our mistakes after. But, no big deal, live and learn... I am going to now rewrite this application and use some lambda expressions to solve the problems inherit with concatenating strings to SQL statements; namely escaping a single quote and SQL Injection attacks. I am going to use the same search screen shown in Figure 1.

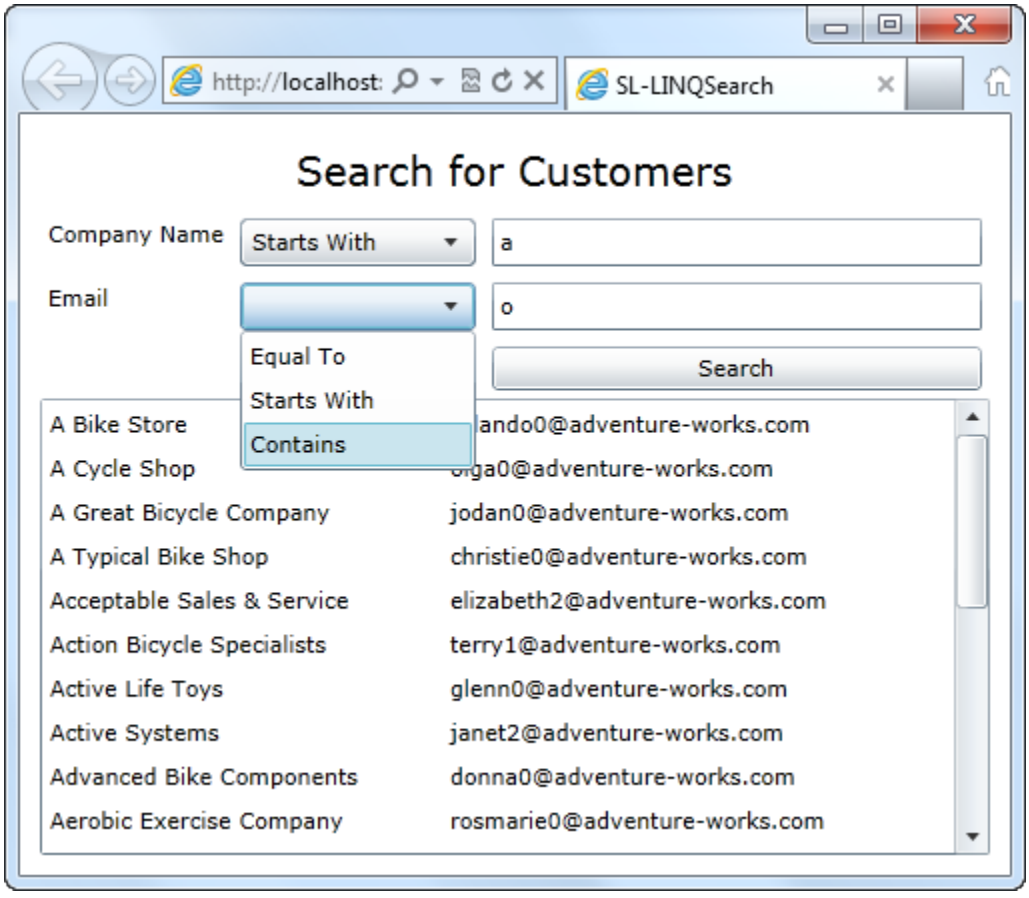

Figure 1: A search screen where the user can select an operation and a value for the searching on multiple fields.

Read the first blog post to see the calls to the WCF service. However, I want to know show the revised code to build a dynamic WHERE clause.

```
C#
public List<Customer> GetCustomers(string cname, string
cnameOperator, string email, string emailOperator)
{
  AdventureWorksLTEntities db =
    new AdventureWorksLTEntities();

  var query = from cust in db.Customers select cust;

  if (string.IsNullOrEmpty(cname) == false)
  {
    switch (cnameOperator.ToLower())
    {
      case "equal to":
        query = query.Where(cust =>
                cust.CompanyName.Equals(cname));
        break;
      case "starts with":
        query = query.Where(cust =>
                cust.CompanyName.StartsWith(cname));
        break;
      case "contains":
        query = query.Where(cust =>
                cust.CompanyName.Contains(cname));
        break;
    }
  }
  if (string.IsNullOrEmpty(email) == false)
  {
    switch (emailOperator.ToLower())
    {
      case "equal to":
        query = query.Where(cust =>
                cust.EmailAddress.Equals(email));
        break;
      case "starts with":
        query = query.Where(cust =>
                cust.EmailAddress.StartsWith(email));
        break;
      case "contains":
        query = query.Where(cust =>
                cust.EmailAddress.Contains(email));
        break;
    }
  }

  query = query.OrderBy(cust => cust.CompanyName);

  return query.ToList();
}

VB
Public Function GetCustomers(cname As String, _
 cnameOperator As String, email As String, _
 emailOperator As String) As List(Of Customer) _
 Implements ICustomerSearch.GetCustomers
```

```
  Dim db As New AdventureWorksLTEntities

  Dim query = From cust In db.Customers Select cust

  If String.IsNullOrEmpty(cname) = False Then
    Select Case cnameOperator.ToLower()
      Case "equal to"
        query = query.Where(Function(cust) _
                cust.CompanyName.Equals(cname))

      Case "starts with"
        query = query.Where(Function(cust) _
                cust.CompanyName.StartsWith(cname))

      Case "contains"
        query = query.Where(Function(cust) _
                cust.CompanyName.Contains(cname))

    End Select
  End If
  If String.IsNullOrEmpty(email) = False Then
    Select Case emailOperator.ToLower()
      Case "equal to"
        query = query.Where(Function(cust) _
                cust.EmailAddress.Equals(email))

      Case "starts with"
        query = query.Where(Function(cust) _
                cust.EmailAddress.StartsWith(email))

      Case "contains"
        query = query.Where(Function(cust) _
                cust.EmailAddress.Contains(email))

    End Select
  End If

  query = query.OrderBy(Function(cust) cust.CompanyName)

  Return query.ToList()
End Function
```

As you can see in the above code you can simply use the Where() function on your IQueryable query to add WHERE clauses that get submitted to the back end database. It is always a good idea to turn on your SQL Profiler and check out the SQL that gets submitted to the back end database.

Summary

The advantage to this approach is now you are relying on the Entity Framework to handle escaping single quotes and avoiding the SQL injection attacks that you would otherwise have to handle. I hope this shows you something useful that you can use in your applications.

NOTE: You can download the sample code for this article by visiting my website at http://www.pdsa.com/downloads. Select “Tips & Tricks”, then select “Dynamic Search with LINQ, the Entity Framework and Silverlight – Part 2” from the drop down list.