Dynamic WHERE clause in LINQ

Ask Question



What is the best way to assemble a dynamic WHERE clause to a LINQ statement?

51

I have several dozen checkboxes on a form and am passing them back as: Dictionary<string, List<string>> (Dictionary<fieldName,List<values>>) to my LINQ query.



22

edited Feb 18 '15 at 4:46



abatishchev 70.9k 70 267 397

asked May 11 '09 at 14:34



Keith Barrows 14.5k 25 70 120

4 ^ ^

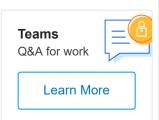




Tags

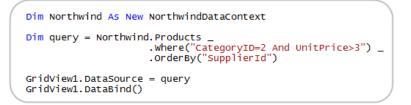
Users

Jobs





52





(source: scottgu.com)

You need something like this? Use the Linq Dynamic Query Library (download includes examples).

Check out ScottGu's blog for more examples.

edited Mar 16 at 8:00



Glorfindel 16.9k 11 52 74

answered May 11 '09 at 14:39



Thomas Stock **6,864** 11 53 76

There is a ported version on github (github.com/kahanu/System.Linq.Dynamic), which I contribute to and help manage. – Ryan Gates Mar 31 '16 at 21:29



You can also use the PredicateBuilder from LinqKit to chain multiple typesafe lambda expressions using Or or And.



13

http://www.albahari.com/nutshell/predicatebuilder.aspx

answered May 13 '09 at 1:08



841 10 2



I have similar scenario where I need to add filters based on the user input and I chain the where clause.

9



Here is the sample code.

```
var votes = db.Votes.Where(r => r.SurveyID == surveyId);
if (fromDate != null)
{
    votes = votes.Where(r => r.VoteDate.Value >= fromDate)
}
if (toDate != null)
{
    votes = votes.Where(r => r.VoteDate.Value <= toDate);
}
votes = votes.Take(LimitRows).OrderByDescending(r => r.VoteDate.Value);
```

answered Jul 30 '14 at 0:28



Best suited for my need and easy to use. Thank you. – user6121177 Aug 24 '17 at 13:56



A simple Approach can be if your Columns are of Simple Type like String

8



public static IEnumerable<MyObject> WhereQuery(IEnumerablecolumnName, string propertyValue)
{
 return source.Where(m => { return m.GetType().GetPropernull).ToString().StartsWith(propertyValue); });

edited Jan 19 '17 at 13:17

answered Mar 20 '13 at 9:34





I came up with a solution that even I can understand... by using the 'Contains' method you can chain as many WHERE's as you like. If the WHERE is an empty string, it's



ignored (or evaluated as a select all). Here is my example of joining 2 tables in LINQ, applying multiple where clauses and populating a model class to be returned to the view. (this is a select all).

```
public ActionResult Index()
   {
        string AssetGroupCode = "";
        string StatusCode = "";
       string SearchString = "";
        var mdl = from a in db.Assets
                  join t in db.Tags on a.ASSETID equals t
                  where a.ASSETGROUPCODE.Contains(AssetGro
                  && a.STATUSCODE.Contains(StatusCode)
                  && (
                  a.PO.Contains(SearchString)
                  | a.MODEL.Contains(SearchString)
                  | a.USERNAME.Contains(SearchString)
                  | a.LOCATION.Contains(SearchString)
                  | t.TAGNUMBER.Contains(SearchString)
                  | t.SERIALNUMBER.Contains(SearchString)
                  select new AssetListView
                      AssetId = a.ASSETID,
                      TagId = t.TAGID,
                      P0 = a.P0,
                      Model = a.MODEL,
                      UserName = a.USERNAME,
```

```
return View(mdl);
}

edited Feb 28 '14 at 23:26

answered Feb 28 '14 at 22:51

mike
51 1 3
```



I had same question (<u>User defined filter for linq</u>), and @tvanfosson told me about Dynamic Linq (http://code.msdn.microsoft.com/csharpsamples).



edited May 23 '17 at 11:46



answered May 11 '09 at 14:40



TcKs

21.3k 6 54 9



You could use the Any() extension method. The following seems to work for me.

1



```
);
Console.WriteLine(root.ToString());
```

Where 'fieldsToSearch' and 'fieldsToReturn' are both List objects.

answered Aug 27 '13 at 17:52



Todd DeLand 1,933 1 17 13



This project on CodePlex have what you want.



System.Linq.Dynamic - http://dynamiclinq.codeplex.com/



Project Description

Extends System.Linq.Dynamic to support Execution of Lambda expressions defined in a string against Entity Framework or any provider that supports IQueryable.

As it is an extension of the source code you can find on Scott Guthrie's Blog it will allow you to do things like this:

And things like this:

```
NorthwindDataContext northwind = new NorthwindDataContext();
Tuberyphile=Products_queryphile=Data = northwindDataContext();
string query = "Products_where(Product => (Product.CategoryID = 3 And (Product.unitPrice > 10))).Take(10)";
var externals = new Dictionarystring, object>();
externals.add("Products", queryphileData);
var experssion = System.inq.DymanicSuppression.Parse(typeof(IQueryphileProduct)), query, new[] { externals });
var result = queryphileData.Provider.CreateQuerycProduct>(expression);
result = TaneryphileData.Provider.CreateQuerycProduct>(expression);
result = TaneryphileData.Provider.CreateQuerycProduct>(expression);
```

answered Sep 21 '13 at 20:03



Zignd

2.747 11 29 53



This is the solution I came up with if anyone is interested.



https://kellyschronicles.wordpress.com/2017/12/16/dynami c-predicate-for-a-ling-query/



First we identify the single element type we need to use (Of TRow As DataRow) and then identify the "source" we are using and tie the identifier to that source ((source As TypedTableBase(Of TRow)). Then we must specify the predicate, or the WHERE clause that is going to be passed (predicate As Func(Of TRow, Boolean)) which will either be returned as true or false. Then we identify how we want the returned information ordered (OrderByField As String). Our function will then return a EnumerableRowCollection(Of TRow), our collection of datarows that have met the conditions of our predicate(EnumerableRowCollection(Of TRow)). This is a basic example. Of course you must make sure your order field doesn't contain nulls, or have handled that situation properly and make sure your column names (if you are using a strongly typed datasource never mind this, it will rename the columns for you) are standard.

edited Dec 17 '17 at 18:59

answered Dec 17 '17 at 18:53



A link to a solution is welcome, but please ensure your answer is useful without it: add context around the link so your fellow users will have some idea what it is and why it's there, then quote the most relevant part of the page you're linking to in case the target page is unavailable. Answers that are little more than a link may be deleted. – FelixSFD Dec 17 '17 at 18:56

I do apologize. I am new here. - KJM Dec 17 '17 at 18:59



It seems much simpler and simpler to use the ternary operator to decide dynamically if a condition is included



List productList = new List();



answered Feb 22 at 0:15



Josué Camacho