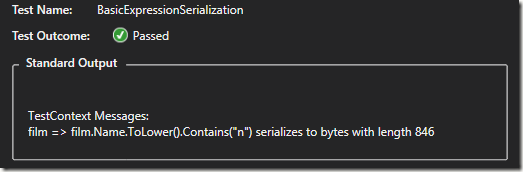
**How to serialize an Expression**

Download (or use NuGet):  <https://github.com/esskar/Serialize.Linq>

using System.Collections.Generic;  
using System.Linq;  
using Microsoft.VisualStudio.TestTools.UnitTesting;  
using Serialize.Linq.Serializers;  
using Serialize.Linq.Tests.Internals;  
using System;  
using System.Linq.Expressions;  
  
namespace Serialize.Linq.Tests  
{  
 public class FilmDto  
 {  
 public string Name { get; set; }  
 public DateTime ReleaseDate { get; set; }  
 }  
  
 [TestClass]  
 public class HowToSerializeAnExpressionTests  
 {  
 public TestContext TestContext { get; set; }  
  
 private static IQueryable<FilmDto> GetFilms()  
 {  
 return new List<FilmDto>  
 {  
 new FilmDto { Name = "Gravity", ReleaseDate = new DateTime(2013, 10, 15) },  
 new FilmDto { Name = "Blade Runner", ReleaseDate = new DateTime(1975, 2, 28) },  
 new FilmDto { Name = "Superman", ReleaseDate = new DateTime(1985, 6, 6) },  
 }  
 .AsQueryable();  
 }  
  
 [TestMethod]  
 public void BasicExpressionSerialization()  
 {  
 var predicate = (Expression<Func<FilmDto, bool>>)(film => film.Name.ToLower().Contains("n"));  
 var serializer = new ExpressionSerializer(new BinarySerializer());  
 var bytes = serializer.SerializeBinary(predicate);  
 var predicateDeserialized = serializer.DeserializeBinary(bytes);  
 this.TestContext.WriteLine("{0} serializes to bytes with length {1}", predicate, bytes.Length);  
   
 ExpressionAssert.AreEqual(predicate, predicateDeserialized);  
  
 var films = GetFilms();  
 var expectedCount = films.Where(predicate).Count();  
   
 // a simple example of executing an expression, leveraging IEnumerable  
 var actualCount = films.Where((Expression<Func<FilmDto, bool>>)predicateDeserialized).Count();  
 Assert.AreEqual(expectedCount, actualCount);  
  
 // constructing the same call using more Expressions  
 Expression whereMethodCall = Expression.Call(  
 typeof(Queryable),   
 "Where",   
 new[] { films.ElementType },   
 films.Expression,   
 Expression.Quote(predicateDeserialized));  
  
 // executing the expression, leveraging IQueryable  
 actualCount = films.Provider.CreateQuery<FilmDto>(whereMethodCall).Count();  
 Assert.AreEqual(expectedCount, actualCount);  
 }  
 }  
   
 // Bear in mind this issue with .NET 45: http://forums.asp.net/t/1864636.aspx  
 // reproducible if you use: (film => film.ReleaseDate < new DateTime(1990, 01, 01))  
 // (which is fixed with .NET 451!)  
 // For finer grain support:  
 // http://msdn.microsoft.com/en-us/library/vstudio/bb882637.aspx  
}

[](http://lh6.ggpht.com/-N4-47qXZ_TQ/Ul7-G3AgyuI/AAAAAAAAEX8/SLt3WnTxlfQ/s1600-h/image%25255B3%25255D.png)

If you want to swap out the BinarySerializer for Json, this would be the payload:

{  
 "\_\_type":"L:#Serialize.Linq.Nodes",  
 "NT":18,  
 "T":{  
 "G":[  
 {  
 "N":"Serialize.Linq.Tests.FilmDto"  
 },  
 {  
 "N":"System.Boolean"  
 }  
 ],  
 "N":"System.Func`2"  
 },  
 "B":{  
 "\_\_type":"MC:#Serialize.Linq.Nodes",  
 "NT":6,  
 "T":{  
 "N":"System.Boolean"  
 },  
 "A":[  
 {  
 "\_\_type":"C:#Serialize.Linq.Nodes",  
 "NT":9,  
 "T":{  
 "N":"System.String"  
 },  
 "V":"n"  
 }  
 ],  
 "M":{  
 "D":{  
 "N":"System.String"  
 },  
 "M":8,  
 "S":"Boolean Contains(System.String)"  
 },  
 "O":{  
 "\_\_type":"MC:#Serialize.Linq.Nodes",  
 "NT":6,  
 "T":{  
 "N":"System.String"  
 },  
 "A":[  
  
 ],  
 "M":{  
 "D":{  
 "N":"System.String"  
 },  
 "M":8,  
 "S":"System.String ToLower()"  
 },  
 "O":{  
 "\_\_type":"M:#Serialize.Linq.Nodes",  
 "NT":23,  
 "T":{  
 "N":"System.String"  
 },  
 "E":{  
 "\_\_type":"P:#Serialize.Linq.Nodes",  
 "NT":38,  
 "T":{  
 "N":"Serialize.Linq.Tests.FilmDto"  
 },  
 "N":"film"  
 },  
 "M":{  
 "D":{  
 "N":"Serialize.Linq.Tests.FilmDto"  
 },  
 "M":16,  
 "S":"System.String Name"  
 }  
 }  
 }  
 },  
 "P":[  
 {  
 "\_\_type":"P:#Serialize.Linq.Nodes",  
 "NT":38,  
 "T":{  
 "N":"Serialize.Linq.Tests.FilmDto"  
 },  
 "N":"film"  
 }  
 ]  
}

If you want to swap it out for the XmlSerializer, this would be the payload:

<?xml version="1.0" encoding="UTF-8"?>  
<E xmlns="http://schemas.datacontract.org/2004/07/Serialize.Linq.Nodes" xmlns:i="http://www.w3.org/2001/XMLSchema-instance" i:type="L">  
 <NT>Lambda</NT>  
 <T>  
 <G>  
 <T>  
 <N>Serialize.Linq.Tests.FilmDto</N>  
 </T>  
 <T>  
 <N>System.Boolean</N>  
 </T>  
 </G>  
 <N>System.Func`2</N>  
 </T>  
 <B i:type="MC">  
 <NT>Call</NT>  
 <T>  
 <N>System.Boolean</N>  
 </T>  
 <A>  
 <E i:type="C">  
 <NT>Constant</NT>  
 <T>  
 <N>System.String</N>  
 </T>  
 <V xmlns:a="http://www.w3.org/2001/XMLSchema" i:type="a:string">n</V>  
 </E>  
 </A>  
 <M>  
 <D>  
 <N>System.String</N>  
 </D>  
 <M>Method</M>  
 <S>Boolean Contains(System.String)</S>  
 </M>  
 <O i:type="MC">  
 <NT>Call</NT>  
 <T>  
 <N>System.String</N>  
 </T>  
 <A />  
 <M>  
 <D>  
 <N>System.String</N>  
 </D>  
 <M>Method</M>  
 <S>System.String ToLower()</S>  
 </M>  
 <O i:type="M">  
 <NT>MemberAccess</NT>  
 <T>  
 <N>System.String</N>  
 </T>  
 <E i:type="P">  
 <NT>Parameter</NT>  
 <T>  
 <N>Serialize.Linq.Tests.FilmDto</N>  
 </T>  
 <N>film</N>  
 </E>  
 <M>  
 <D>  
 <N>Serialize.Linq.Tests.FilmDto</N>  
 </D>  
 <M>Property</M>  
 <S>System.String Name</S>  
 </M>  
 </O>  
 </O>  
 </B>  
 <P>  
 <E i:type="P">  
 <NT>Parameter</NT>  
 <T>  
 <N>Serialize.Linq.Tests.FilmDto</N>  
 </T>  
 <N>film</N>  
 </E>  
 </P>  
</E>