عنوان: **استفاده از Lambda Expression در پروژه های مبتنی بر WCF**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۳/۳۰ ۱۹:۱۰
آدرس: www.dotnettips.info
برچسب‌ها: WCF, Serialization, Expression, Delegate

نکته : آشنایی با مفاهیم پایه WCF برای فهم بهتر مفاهیم توصیه می‌شود.

امروزه استفاده از WCF در پروژه‌های SOA بسیار فراگیر شده است. کمتر کسی است که در مورد قدرت تکنولوژی WCF نشنیده باشد یا از این تکنولوژی در پروژه‌های خود استفاده نکرده باشد. WCF مدل برنامه نویسی یکپارچه مایکروسافت برای ساخت نرم‌افزارهای سرویس گرا است و برای توسعه دهندگان امکانی را فراهم می‌کند که راهکارهایی امن، و مبتنی بر تراکنش را تولید نمایند که قابلیت استفاده در بین پلتفرم‌های مختلف را دارند. قبل از WCF توسعه دهندگان پروژه‌های نرم افزاری برای تولید پروژه‌های توزیع شده باید شرایط موجود برای تولید و توسعه را در نظر می‌گرفتند. برای مثال اگر استفاده کننده از سرویس در داخل سازمان و بر پایه دات نت تهیه شده بود از .net remoting استفاده می‌کردند و اگر استفاده کننده سرویس از خارج سازمان یا مثلا بر پایه تکنولوژی J2EE بود از Web Serviceها استفاده می‌شد. با ظهور WCF این تکنولوژی با هم تجمیع شدند(بهتر بگم تبدیل به یک تکنولوژی واحد شدند) و دیگر خبری از net remoting یا web serviceها نیست.

WCF با تمام قدرت و امکاناتی که داراست دارای نقاط ضعفی هم می‌باشد که البته این معایب (یا محدودیت) بیشتر جهت سازگار سازی سرویس‌های نوشته شده با سیستم‌ها و پروتکل‌های مختلف است.
برای انتقال داده‌ها از طریق WCF بین سیستم‌های مختلف باید داده‌های مورد نظر حتما سریالایز شوند که مثال هایی از این دست رو در همین سایت می‌تونید مطالعه کنید:
(^) و (^) و (^)

با توجه به این که داده‌ها سریالایز می‌شوند، در نتیجه امکان انتقال داده هایی که از نوع object هستند در WCF وجود ندارد. بلکه نوع داده باید صراحتا ذکر شود و این نوع باید قابلیت سریالایز شدن را دارا باشد.برای مثال شما نمی‌تونید متدی داشته باشید که پارامتر ورودی آن از نوع delegate باشد یا کلاسی باشد که صفت [Serializable] در بالای اون قرار نداشته باشد یا کلاسی باشد که صفت DataContract برای خود کلاس و صفت DataMember برای خاصیت‌های اون تعریف نشده باشد. حالا سوال مهم این است اگر متدی داشته باشیم که پارامتر ورودی آن حتما باید از نوع delegate باشد چه باید کرد؟

برای تشریح بهتر مسئله یک مثال می‌زنم؟

سرویسی داریم برای اطلاعات کتاب ها. قصد داریم متدی بنوسیم که پارامتر ورودی آن از نوع Lambda Expression است تا Query مورد نظر کاربر از سمت کلاینت به سمت سرور دریافت کند و خروجی مورد نظر را با توجه به Query ورودی به کلاینت برگشت دهد.(متدی متداول در اکثر پروژه ها). به صورت زیر عمل می‌کنیم.

*ابتدا یک Blank Solution ایجاد کنید.

*یک ClassLibrary به نام Model ایجاد کنید و کلاسی به نام Book در آن بسازید .(همانطور که میبینید کلاس مورد نظر سریالایز شده است):

```
[DataContract]
    public class Book
    {
        [DataMember]
        public int Code { get; set; }

        [DataMember]
        public string Title { get; set; }
    }
```

* یک WCF Service Application ایجاد کنید

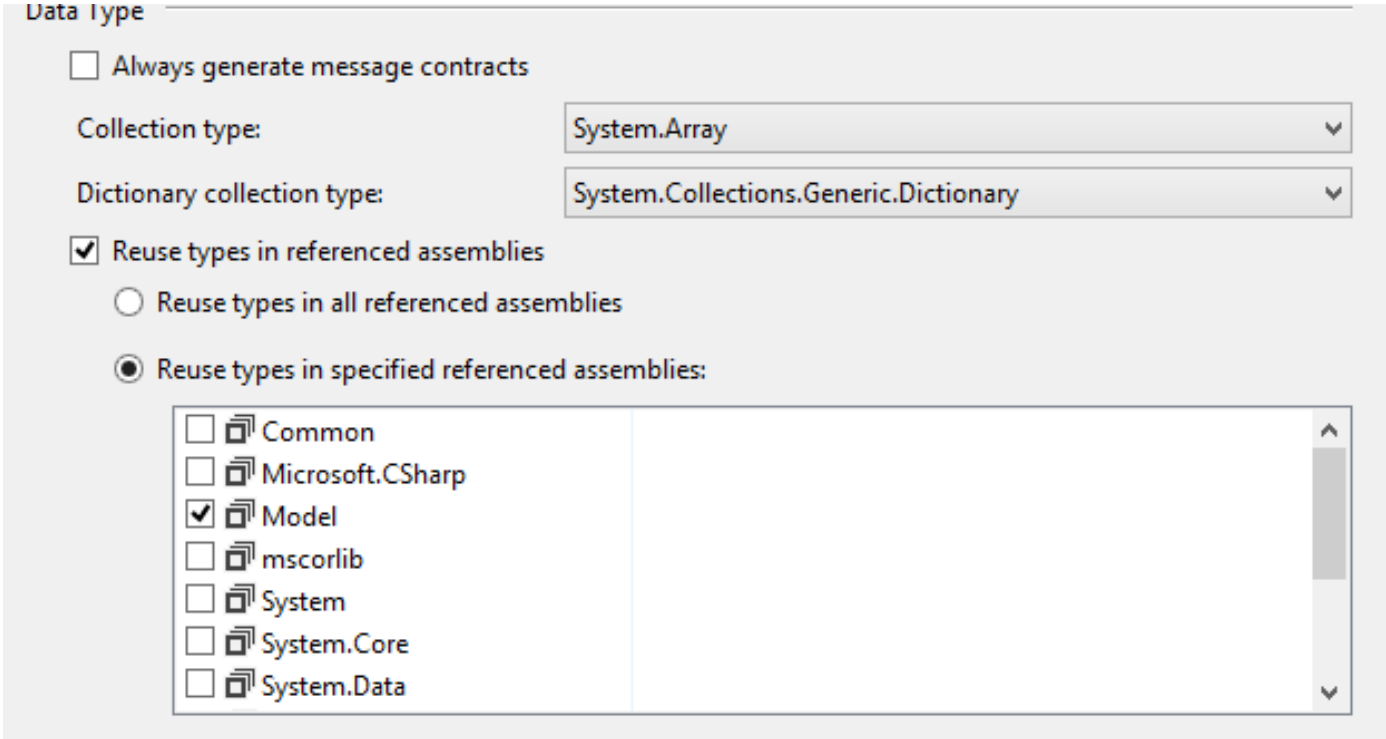
یک Contract برای ارتباط بین سرور و کلاینت می‌سازیم:

```
using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using System.ServiceModel;

namespace WcfLambdaExpression
{
    [ServiceContract]
    public interface IBookService
    {
        [OperationContract]
        IEnumerable<Book> GetByExpression( Expression<Func<Book, bool>> expression );
    }
}
```

متد GetByExpression دارای پارامتر ورودی expression است که نوع آن نیز Lambda Expression می‌باشد. حال یک سرویس ایجاد می‌کنیم:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;

namespace WcfLambdaExpression
{
    public class BookService : IBookService
    {
        public BookService()
        {
            ListOfBook = new List<Book>();
        }

        public List<Book> ListOfBook
        {
            get;
            private set;
        }

        public IEnumerable<Book> GetByExpression( Expression<Func<Book, bool>> expression )
        {
            ListOfBook.AddRange( new Book[]
            {
                new Book(){Code = 1 , Title = "Book1"},
                new Book(){Code = 2 , Title = "Book2"},
                new Book(){Code = 3 , Title = "Book3"},
                new Book(){Code = 4 , Title = "Book4"},
                new Book(){Code = 5 , Title = "Book5"},
            } );

            return ListOfBook.AsQueryable().Where( expression );
        }
    }
}
```

بعد از Build پروژه همه چیز سمت سرور آماده است. یک پروژه دیگر از نوع Console ایجاد کنید و از روش AddServiceReference سعی کنید که سرویس مورد نظر را به پروژه اضافه کنید. در هنگام Add Service Reference برای اینکه سرویس سمت سرور و کلاینت هر دو با یک مدل کار کنند باید از یک Reference assembly استفاده کنند و کافی است از قسمت Advanced گزینه Reuse types in referenced assemblies را تیک بزنید و assemblyهای مورد نظر را انتخاب کنید.(در این پروژه باید Model و System.Xml.Linq را انتخاب کنید)



به طور حتم با خطا روبرو خواهید شد. دلیل آن هم این است که امکان سریالایز کردن برای پارامتر ورودی expression میسر نیست.

خطای مربوطه به شکل زیر خواهد بود:

```
Type 'System.Linq.Expressions.Expression`1[System.Func`2[WcfLambdaExpression.Book,System.Boolean]]'
cannot be serialized.
Consider marking it with the DataContractAttribute attribute, and marking all of its members you want
serialized with the DataMemberAttribute attribute.
If the type is a collection, consider marking it with the CollectionDataContractAttribute.
See the Microsoft .NET Framework documentation for other supported types
```

حال چه باید کرد؟

روش‌های زیادی برای بر طرف کردن این محدودیت وجود دارد. اما در این پست روشی رو که خودم از اون استفاده می‌کنم رو براتون شرح می‌دهم.

در این روش باید از XElement استفاده شود که در فضای نام System.Linq.Xml قرار دارد. یعنی آرگومان ورودی سمت کلاینت باید به فرمت Xml سریالایز شود و سمت سرور دوباره دی سریالایز شده و تبدیل به یک Lambda Expression شود. اما سریالایز کردن Lambda Expression واقعا کاری سخت و طاقت فرساست . با توجه به این که در اکثر پروژه‌ها این متدها به صورت Generic نوشته می‌شوند. برای حل این مسئله بعد از مدتی جستجو، کلاسی رو پیدا کردم که این کار رو برام انجام می‌داد. بعد از مطالعه دقیق و مشاهده روش کار کلاس، تغییرات مورد نظرم رو اعمال کردم و الان در اکثر پروژه هام دارم از این کلاس استفاده می‌کنم.

یک مثال از روش استفاده :

برای اینکه از این کلاس در هر دو پروژه (سرور و کلاینت) استفاده می‌کنیم باید یک Class Library جدید به نام Common بسازید و یک ارجاع از اون رو به هر دو پروژه سمت سرور و کلاینت بدید.

سرویس و Contract بالا رو به صورت زیر باز نویسی کنید.

```
[ServiceContract]
    public interface IBookService
    {
        [OperationContract]
        IEnumerable<Book> GetByExpression( XElement expression );
    }
```

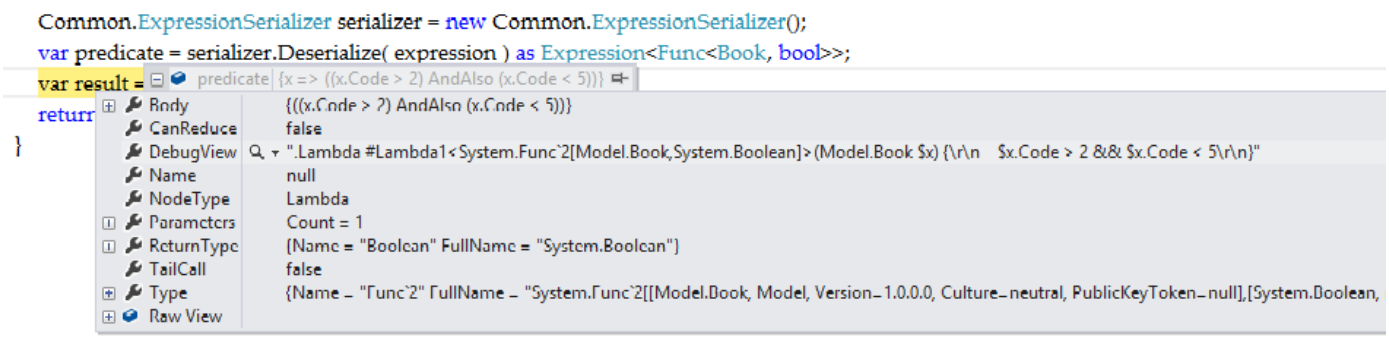
و سرویس :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Xml.Linq;

namespace WcfLambdaExpression
{
    public class BookService : IBookService
    {
        public BookService()
        {
            ListOfBook = new List<Book>();
        }

        public List<Book> ListOfBook
        {
            get;
            private set;
        }

        public IEnumerable<Book> GetByExpression( XElement expression )
        {
            ListOfBook.AddRange( new Book[]
            {
                new Book(){Code = 1 , Title = "Book1"},
                new Book(){Code = 2 , Title = "Book2"},
                new Book(){Code = 3 , Title = "Book3"},
                new Book(){Code = 4 , Title = "Book4"},
                new Book(){Code = 5 , Title = "Book5"},
            } );

             Common.ExpressionSerializer serializer = new Common.ExpressionSerializer();

            return ListOfBook.AsQueryable().Where( serializer.Deserialize( expression ) as
Expression<Func<Book, bool>> );
        }
    }
```

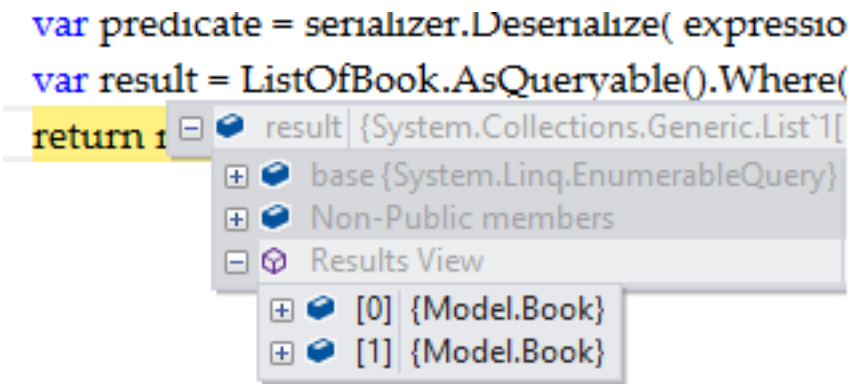
بعد از Build پروژه از روش Add Service Reference استفاده کنید و می‌بینید که بدون هیچ گونه مشکلی سرویس مورد نظر به پروژه Console اضافه شد. برای استفاده سمت کلاینت به صورت زیر عمل کنید.

```
using System;
using System.Linq.Expressions;
using TestExpression.MyBookService;

namespace TestExpression
{
    class Program
    {
        static void Main( string[] args )
        {
            BookServiceClient bookService = new BookServiceClient();

            Expression<Func<Book, bool>> expression = x => x.Code > 2 && x.Code < 5;

            Common.ExpressionSerializer serializer = new Common.ExpressionSerializer();

            bookService.GetByExpression( serializer.Serialize( expression ) );
        }
    }
}
```

بعد از اجرای پروژه، در سمت سرور خروجی‌های زیر رو مشاهده می‌کنیم.



خروجی هم به صورت زیر خواهد بود:



دریافت سورس کامل Expression-Serialization

نظرات خوانندگان

نویسنده: سایلنت
تاریخ: ۱۳۹۲/۰۸/۰۲ ۱۵:۱۱

بسیار عالیه . تازه شروع کردم به یادگیری WCF از مقالات شما نهایت استفاده رو بردم .

نویسنده: محمد
تاریخ: ۱۳۹۲/۰۹/۱۹ ۱۷:۱۶

سلام و ممنون از مقاله خوبتون، اما متاسفانه کلاس شما رو نمیشه برای JSON استفاده نمود.

```
string json = JsonConvert.SerializeObject(serializer.Serialize(predicate3));
predicate3 = JsonConvert.DeserializeObject<Expression<Func<Entity, bool>>>(json);
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۹ ۲۲:۵۸

- اینکار اضافی است. چون xml را تبدیل به json می‌کنید؛ بعد json را تبدیل به xml.

+ خروجی serializer.Serialize از نوع XElement است. بنابراین در قسمت آرگومان جنریک JsonConvert.DeserializeObject باید XElement ذکر شود. مرحله بعدی آن فراخوانی serializer.Deserialize روی این خروجی است.

```
Expression<Func<Book, bool>> expression = x => x.Code > 2 && x.Code < 5;
var expressionSerializer = new Common.ExpressionSerializer();
var xml = expressionSerializer.Serialize(expression);
var xmlToJson = JsonConvert.SerializeObject(xml);
var xmlObject = JsonConvert.DeserializeObject<XElement>(xmlToJson);
var exp2 = expressionSerializer.Deserialize(xmlObject) as Expression<Func<Book, bool>>;
```

عنوان: **فراخوانی متد Parent Page از User-Control**
نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۲/۱۰/۲۶ ۲۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Delegate, User Control

در ASP.Net، ما user-control سفارشی را جهت استفاده مجدد و مستقل در صفحات ASPX ایجاد می‌کنیم. هر user-control دارای properties عمومی، متدها و یا delegateهای خاص خود است و زمانی که user-control در یک صفحه وب جاسازی (embedded) یا فرخوانی (load) می‌شود بوسیله صفحه وب قابل استفاده است.
بعد از درج user-control در صفحه وب و فراخوانی آن، ممکن است نیاز باشد مثلاً باکلیک بر روی دکمه‌ای از user-control متدی از صفحه اجرا شود. اما یک مشکل، زمانی که در حال ایجاد user-control هستید هیچ اطلاعی از صفحه ای که قرار است user-control در آن قرار بگیرد ندارید پس چگونه می‌توانیم به متدهای آن دسترسی داشته باشیم؟!
در کلاس Delegate، متدی بنام DynamicInvoke وجود دارد که برای فراخوانی (Invoke) متد اشاره شده در delegate استفاده می‌شود. ما از این متد برای صدا زدن یک متد صفحه وبی که user-control در آن قرار دارد استفاده می‌کنیم.
مثال:

```
public partial class CustomUserCtrl : System.Web.UI.UserControl
{
     private System.Delegate _delWithParam;
     private System.Delegate _delNoParam;

     // برای فراخوانی متدهایی از صفحه که دارای پارامتر هستند
     public Delegate PageMethodWithParamRef
     {
        set { _delWithParam = value; }
     }

     // برای فراخوانی متدهایی از صفحه که بدون پارامتر هستند
     public Delegate PageMethodWithNoParamRef
     {
        set { _delNoParam = value; }
     }

     protected void Page_Load(object sender, EventArgs e)
     {
     }

     protected void BtnMethodWithParam_Click(object sender, System.EventArgs e)
     {
        //Parameter to a method is being made ready
        object[] obj = new object[1];
        obj[0] = “Parameter Value” as object;
        _delWithParam.DynamicInvoke(obj);
     }

     protected void BtnMethowWithoutParam_Click(object sender, System.EventArgs e)
     {
        //Invoke a method with no parameter
        _delNoParam.DynamicInvoke();
     }
}
```

فرض کنید در user-control بالا، دو دکمه وجود دارد که متد BtnMethodWithParam_Click را به رویداد کلیک یک دکمه، و متد BtnMethowWithoutParam_Click به رویداد کلیک دکمه دیگر منتسب می‌کنیم، سپس دو عامل خصوصی (Private) را تعریف می‌کنیم و متد DynamicInvoke این عامل‌های خصوصی را در متدهای BtnMethodWithParam_Click و BtnMethowWithoutParam_Click فراخوانی می‌کنیم حال کافیست عامل‌هایی در صفحه تعریف کنیم که این عامل‌ها به متدهای مورد نظر صفحه اشاره کنند و این عامل‌های صفحه را در عامل‌های عمومی user-control قرار دهیم. در ادامه به پیاده سازی صفحه می‌پردازیم:
ابتدا دو عامل تعریف می‌کنیم:

```
public partial class _Default : System.Web.UI.Page
{
     delegate void DelMethodWithParam(string strParam);
     delegate void DelMethodWithoutParam();
```

در رویداد Page_Load، یک وهله از هر کدام از عامل‌های بالا که به متد (**توجه:** امضاء متدها با امضاء عامل‌ها یکسان است) مورد نظر ما در صفحه اشاره می‌کند ایجاد می‌کنیم:

```
protected void Page_Load(object sender, EventArgs e)
     {
        DelMethodWithParam delParam = new DelMethodWithParam(MethodWithParam);

        عامل صفحه را به عامل عمومی تعریف شده در یوزر کنترل تخصیص می‌دهیم //
        this.UserCtrl.PageMethodWithParamRef = delParam;
        DelMethodWithoutParam delNoParam = new DelMethodWithoutParam(MethodWithNoParam);

        عامل صفحه را به عامل عمومی تعریف شده در یوزر کنترل تخصیص می‌دهیم //
        this.UserCtrl.PageMethodWithNoParamRef = delNoParam;
     }
```

در زیر متدهایی خصوصی که در صفحه وجود دارند و قرار است با کلیک بر روی دکمه‌های user-control فراخوانی شوند را مشاهده می‌کنید:

```
متد دارای پارامتری که قرار است در کنترل فراخوانی شود //
     private void MethodWithParam(string strParam)
     {
        Response.Write(“It has parameter: ” + strParam);
     }

     متد بدون پارامتری که قرار است در کنترل فراخوانی شود //
     private void MethodWithNoParam()
     {
        Response.Write(“It has no parameter.”);
     }
```

و در نهایت کد پیاده سازی نهایی صفحه ما بشکل زیر خواهد شد:

```
public partial class _Default : System.Web.UI.Page
{
     delegate void DelMethodWithParam(string strParam);
     delegate void DelMethodWithoutParam();

     protected void Page_Load(object sender, EventArgs e)
     {
        DelMethodWithParam delParam = new DelMethodWithParam(MethodWithParam);

        عامل صفحه را به عامل عمومی تعریف شده در یوزر کنترل تخصیص می‌دهیم //
        this.UserCtrl.PageMethodWithParamRef = delParam;
        DelMethodWithoutParam delNoParam = new DelMethodWithoutParam(MethodWithNoParam);

        عامل صفحه را به عامل عمومی تعریف شده در یوزر کنترل تخصیص می‌دهیم //
        this.UserCtrl.PageMethodWithNoParamRef = delNoParam;
     }

     متد دارای پارامتری که قرار است در کنترل فراخوانی شود //
     private void MethodWithParam(string strParam)
     {
        Response.Write(“It has parameter: ” + strParam);
     }

     متد بدون پارامتری که قرار است در کنترل فراخوانی شود //
     private void MethodWithNoParam()
     {
        Response.Write(“It has no parameter.”);
     }
}
```

برداشتی آزاد از این مقاله .

عنوان: **فراخوانی متدهای Controllerها در Viewهای ASP.NET MVC**
نویسنده: تایماز قبادی
تاریخ: ۱۳۹۳/۱۱/۰۶ ۱۵:۱۵
آدرس: www.dotnettips.info
گروه‌ها: C#, MVC, Delegate, Func

شاید در ابتدا فراخوانی متدی از یک کنترلر در یک View کار سختی به نظر برسد، ولی در واقع با استفاده از مفاهیم Lambda expressions و Delegateها این کار بسیار راحت خواهد بود.

برای این کار میتوانیم متد مورد نظر را به صورت یک delegate تعریف کرده و به view ارسال کنیم. فرض کنیم متدی داریم برای برگرداندن مجموع 2 عدد به صورت string:

```
public string Sum(int a,int b)
{
    return (a + b).ToString();
}
```

حال برای اینکه بتوانیم این متد را بصورت یک delegate به view ارسال کنیم لازم است تا یک delegate را بصورت public و در خارج از تعریف کلاسها و درون یک namespace مشخصی تعریف کنیم. در اینجا برای راحتی در همان MvcTest.Controllers namespace یک delegate را بصورت زیر (MvcTest نام پروژه است) تعریف می‌کنیم:

```
public delegate string SumOf2Number(int a, int b);
```

حال میتوانیم بصورت زیر این متد را از طریق ViewBag به View ارسال کنیم:

```
SumOf2Number sum2numbers = Sum;
ViewBag.SumFunc3 = sum2numbers;
```

در روش دوم، می‌توانیم متد مورد نظر را بصورت Func به View ارسال کنیم. این کار را میتوانیم به دو صورت انجام دهیم، که هر دو را در تکه کد زیر خواهید دید:

```
ViewBag.SumFunc =(Func<int,int,string>) Sum;//way 1
ViewBag.SumFunc2 = (Func<int, int, string>)((int a, int b) => { return (a + b).ToString(); });//way 2
```

همانطور که متوجه شدید، در روش اول تنها کاری که کردیم متد Sum را از طریق TypeCasting به یک Func تبدیل کردیم و در روش دوم هم یک Lambda expression را بصورت مستقیم به Func تبدیل کرده و استفاده کردیم.

میتوانیم یک Lambda expression را به یک متغیر delegate نیز ربط دهیم؛ به این صورت:

```
SumOf2Number sum2numbers2 = (int a, int b) => { return (a + b).ToString(); };
```

در نهایت کد بخش کنترلر کلاً به اینصورت خواهد بود:

```
namespace MvcTest.Controllers
{
    public  delegate string SumOf2Number(int a, int b);
```

```
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            SumOf2Number sum2numbers = Sum;
            SumOf2Number sum2numbers2 = (int a, int b) => { return (a + b).ToString(); };

            ViewBag.SumFunc =(Func<int,int,string>) Sum;
            ViewBag.SumFunc2 = (Func<int, int, string>)((int a, int b) => { return (a + b).ToString();
});
            ViewBag.SumFunc3 = sum2numbers;
            ViewBag.SumFunc4 = sum2numbers2;
            return View();
        }
        public string Sum(int a,int b)
        {
            return (a + b).ToString();
        }
        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
}
```

و در Index View خواهیم داشت: (البته اصولاً استفاده از controller namespace در سمت view کار درستی نیست، منتها اینجا فقط یک مثال کاربردی ساده است)

```
@using MvcTest.Controllers;
@{
    ViewBag.Title = "Home Page";
}

<h1>
    @ViewBag.SumFunc(7,8)
</h1>
<h1>
    @ViewBag.SumFunc2(9, 10)
</h1>
<h1>
    @ViewBag.SumFunc3(5, 1)
</h1>
<h1>
    @ViewBag.SumFunc4(2, 3)
</h1>
```