

Inledning

Det här dokumentet är till för att ge en översikt över ASP.NET MVC samt hur WCF Services används från .NET applikationer.

Dokumentet består av två delar där första delen av dokumentet beskriver i detalj hur MVC applikationer fungerar genom att beskriva de olika rollerna för Model, View och Controller i MVC, detta görs genom detaljerade steg från File -> New Project till en färdig applikation.

Den andra delen av dokumentet beskriver hur en WCF Service används från en ASP.NET MVC applikation men sättet är desamma för alla .NET applikationer. Från att hitta en service och lägga till en referens till att konfigurera bindningarna för kommunikationen.

Flertalet av de termer och koncept som förklaras och används i instruktionerna gör sig bäst på engelska varnas känsliga läsare för en stor dos av svengelska i detta dokument, det var mer eller mindre ofrånkomligt.

Innehåll

ASP.NET MVC.....	1
MVC (Model, View och Controller)	1
Nytt projekt	1
Controllers	2
Views	3
Routing	6
Models.....	8
Strongly-typed views & scaffold templates.....	9
Service References	11
Lägg till en Service Reference.....	13
Använda servicen	16
Konfigurera service bindings	18

ASP.NET MVC

Detta kapitel ger en introduktion till design principen MVC och hur principen används i en ASP.NET MVC applikation. Introduktionen utgår från File -> New Project och går sedan steg för steg igenom hur flödet i dessa applikationer fungerar.

MVC (Model, View och Controller)

MVC är en design princip och har en klar uppdelning av ansvarsområden mellan Models, Views och Controllers. Det är viktigt att särskilja och följa dessa ansvarsområden i er implementation av applikationen.

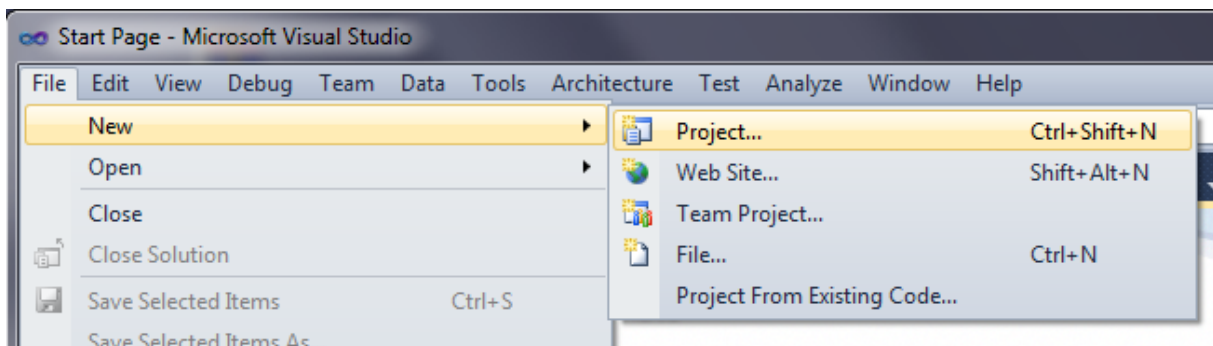
Model – Domän objekten/Logiklagret i applikationen, består av vanliga klasser som representerar de modeller som används i applikationen. I modellerna ska logiken för hur olika objekt fungerar i applikationen samt eventuella affärsregler som ska implementeras.

View – Presentationen av applikationen gentemot användarna. Vyerna är skrivna i HTML och C# för att göra dem dynamiska. Tänk på att inte skriva annan kod än den som behövs för att presentera de objekt som skickats till vyn.

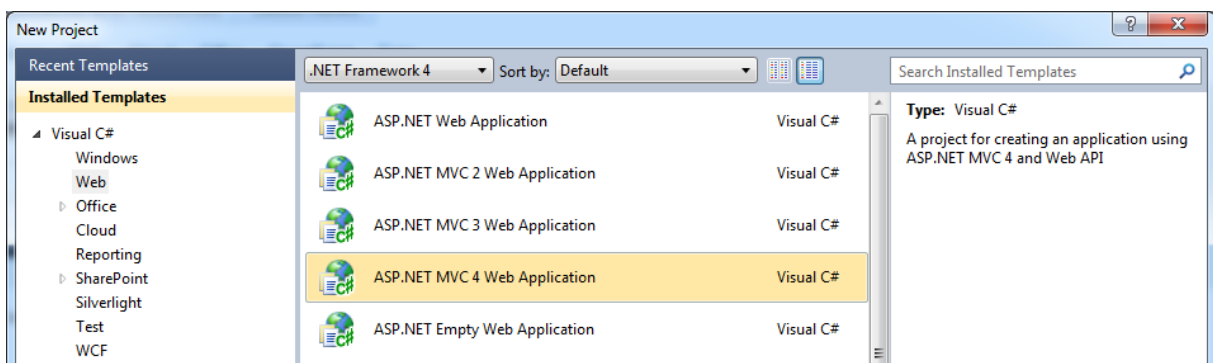
Controller – Hanterar flödet i applikationen genom, t.ex. vad händer när en användare navigerar till en specifik URL på webbsidan, vart ska information i applikationen skickas etc.

Nytt projekt

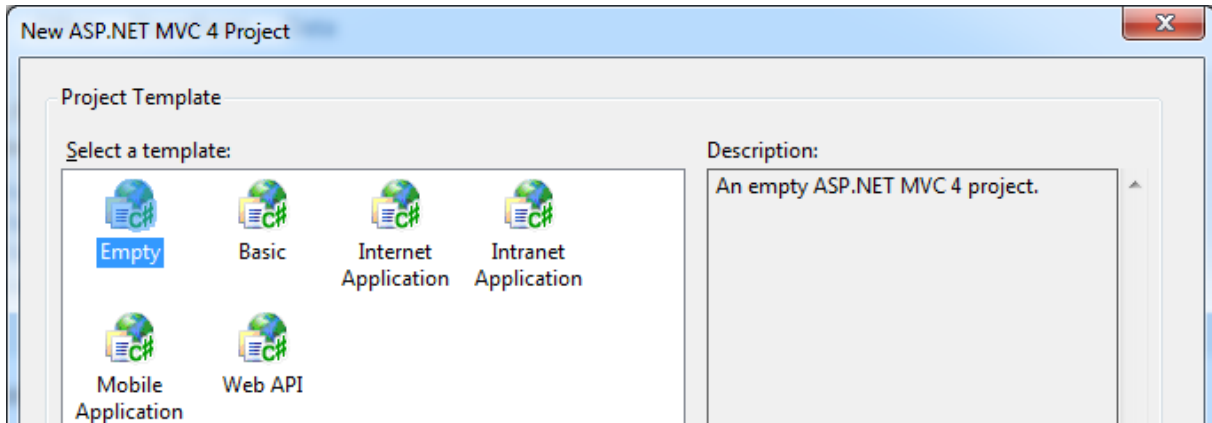
Nedan kommer nu en introduktion till ASP.NET MVC, detta görs genom att börja med ett nytt ASP.NET MVC projekt.



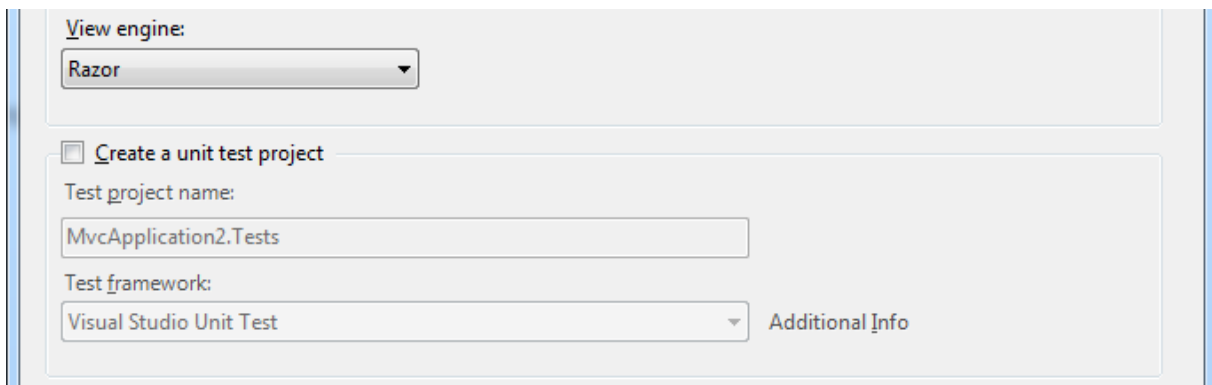
Välj ASP.NET MVC 4 Web Application som finns i Web-fliken.



Välj om du vill starta med ett tomt projekt välj "Empty" om du vill låta Visual Studio skapa ett template projekt för en webbsida välj någon av de andra projekttyperna så får du ett par implementerade Controller/Models/Views/CSS med viss grundfunktionalitet. Dessa instruktioner utgår från en tom applikation.

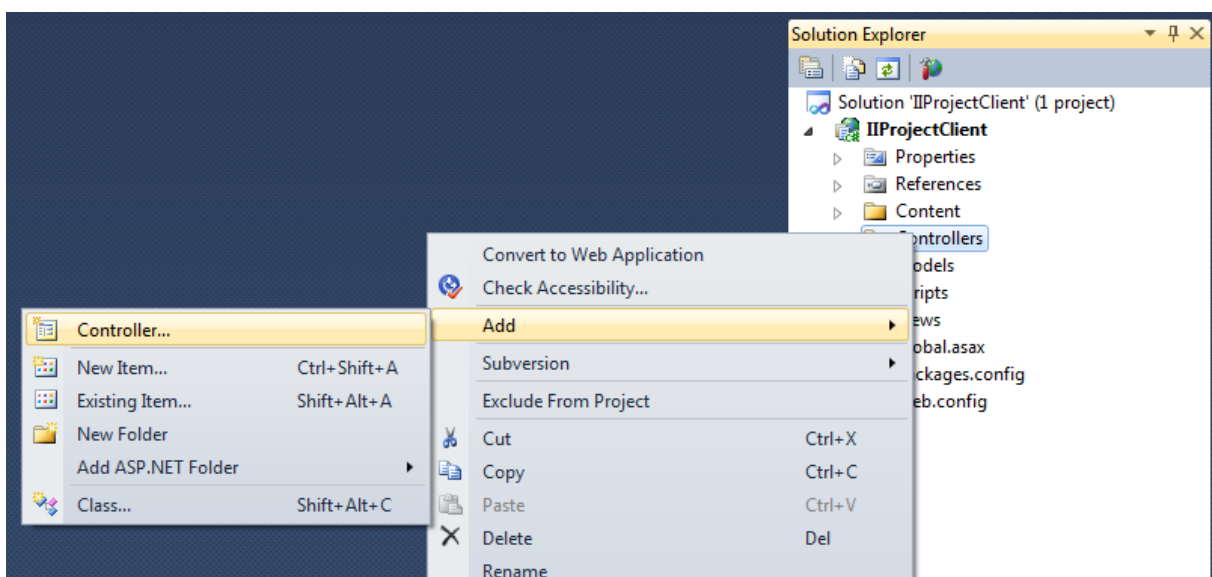


I dessa instruktioner väljs Razor-syntax för vyerna och inget test-projekt.

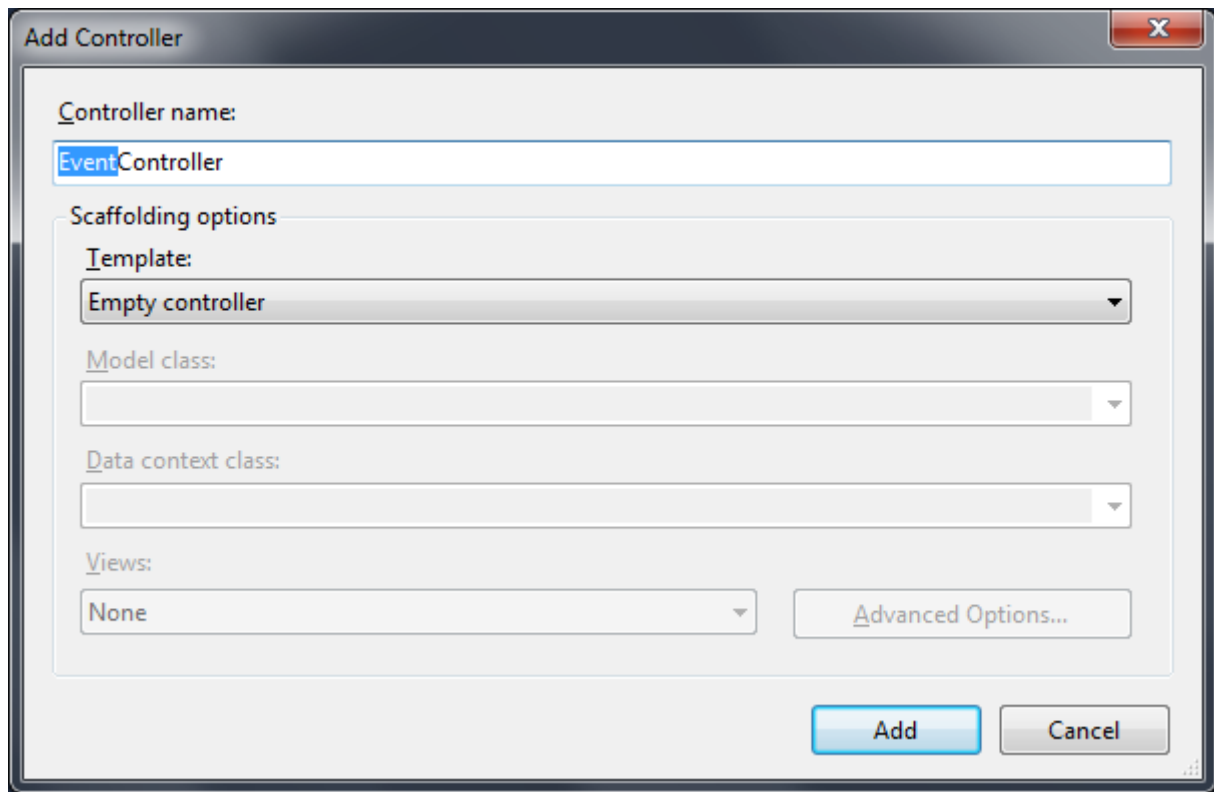


Controllers

C i MVC är Controller och controllers hanterar flödet i applikationen. Dessa placeras i "Controllers" mappen, för att lägga till en ny controller högerklicka på mappen "Controllers" -> Add -> Controller...



Som standard avslutas alla Controller-klasser och filer med just ordet "Controller", nedan följs principen när en EventController skapas. Det finns även templates för att implementera vanliga metoder i controllern men här skapas en tom.



Klassen skapas och en Index-metod skapas för controllern.

```
public class EventController : Controller
{
    //
    // GET: /Event/

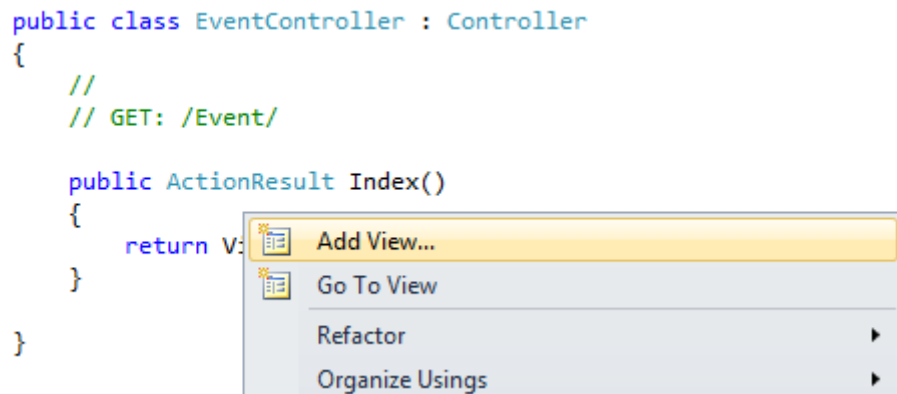
    public ActionResult Index()
    {
        return View();
    }
}
```

Controllern hanterar flödet i applikationen, dvs en viss URL kopplas samman med en metod i en controller. Metoden i sin tur förmedlar senare vidare till vyerna genom anropet "return View();". Mer om denna koppling senare under rubriken "Routing".

Views

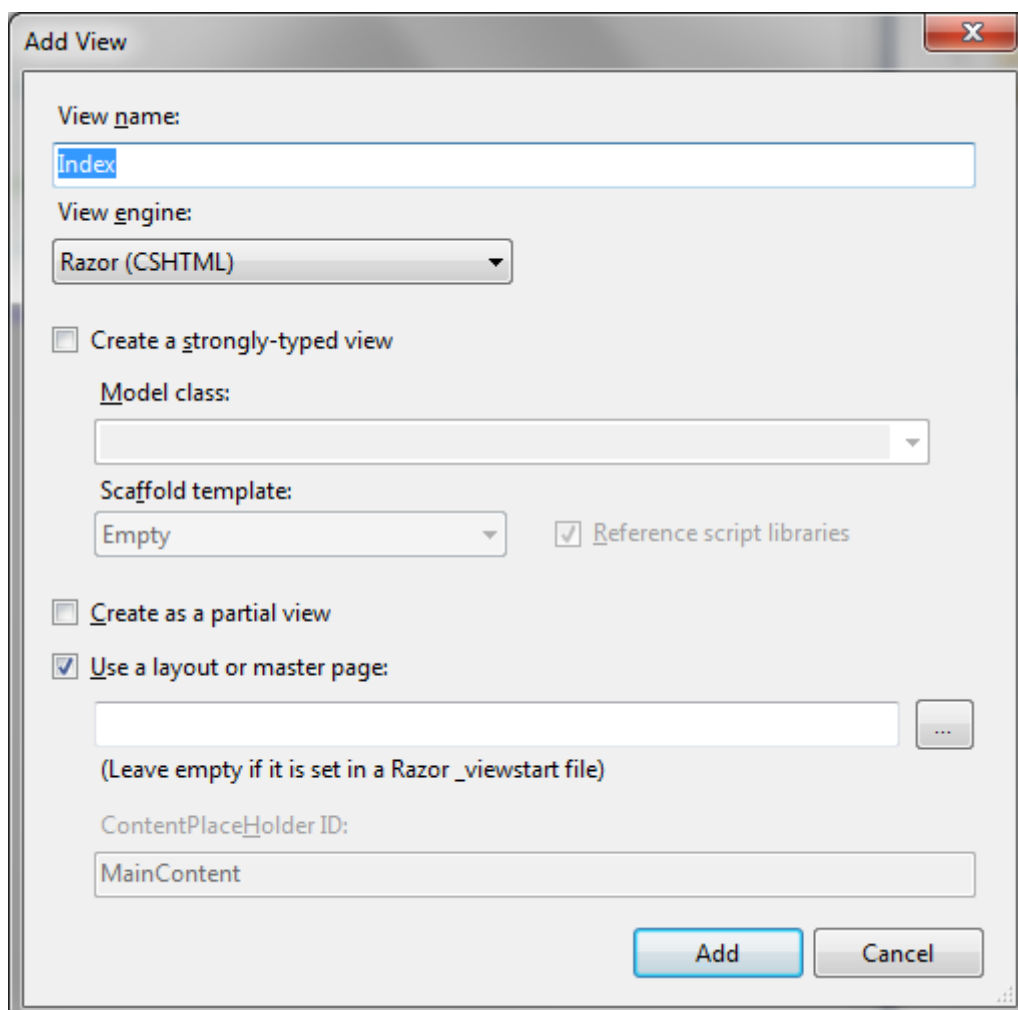
V i MVC står för View, och representerar presentationslagret för applikationen. I ASP.NET MVC görs presentationen i form av HTML-sidor som kan skapas genom att blanda HTML-taggar med kodsuttag med av C#-kod.

Ännu finns ingen vy kopplad till Index-metoden i den ovan skapade EventController, nedan läggs den till genom högerklicka i metoden -> Add View...



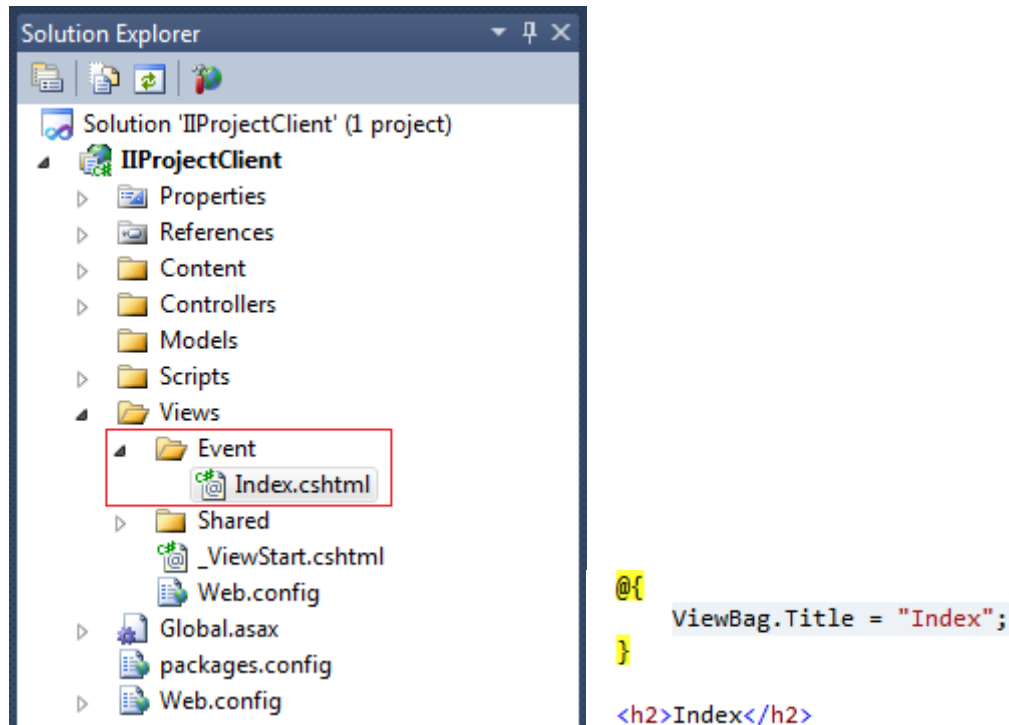
Följande ruta visas, namnet på vyn är redan ifyllt, det är samma som metodnamnet, behåll detta då anropet från Index-metoden kopplas mot en vy med samma namn som metoden. Detta går såklart att ändra om du använder en överlagrad View-metod. Nedan används standardnamnet.

Det går även att använda olika typer av scaffolding templates och typade-vyer, dessa förklaras under rubriken "Strongly-typed views & scaffold templates", vyn som skapas nedan är ej bunden mot någon modell och använder sig inte av någon scaffolding.



Index-vyn som skapats placeras i mappen "Views" och i undermappen "Event", alltså en mapp med samma namn som den Controller som vi skapade vyn ifrån.

I vyn finns en kort kodsnuitt samt lite HTML för att skriva ut titel och en rubrik för vyn, nedan till höger.



ViewBag är en egenskap som finns på både Controller-sidan och vy-sidan, det är en dynamiskt typad kollektion vars värde går att hämta i vyn. Med dynamiskt menas att det går att skriva punkt "." efter ViewBag och sedan skriva en egenskap med valfritt namn, bindningen sker dock först under körningen så det går inte att få IntelliSense i vyn.

Nedan läggs texten "Välkommen!" in i Message för ViewBag, detta värde kan sedan hämtas i vår vy. Nedan visas controller till vänster och vyn till höger.

```
public class EventController : Controller
{
    //
    // GET: /Event/

    public ActionResult Index()
    {
        ViewBag.Message = "Välkommen!";

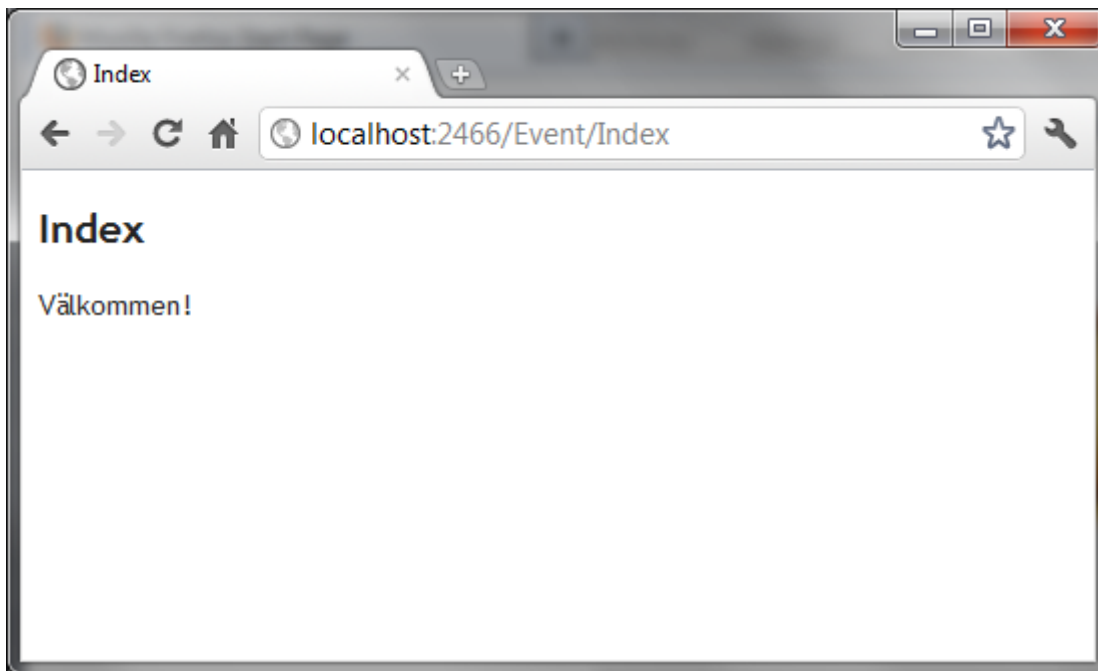
        return View();
    }
}

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>@ViewBag.Message</p>
```

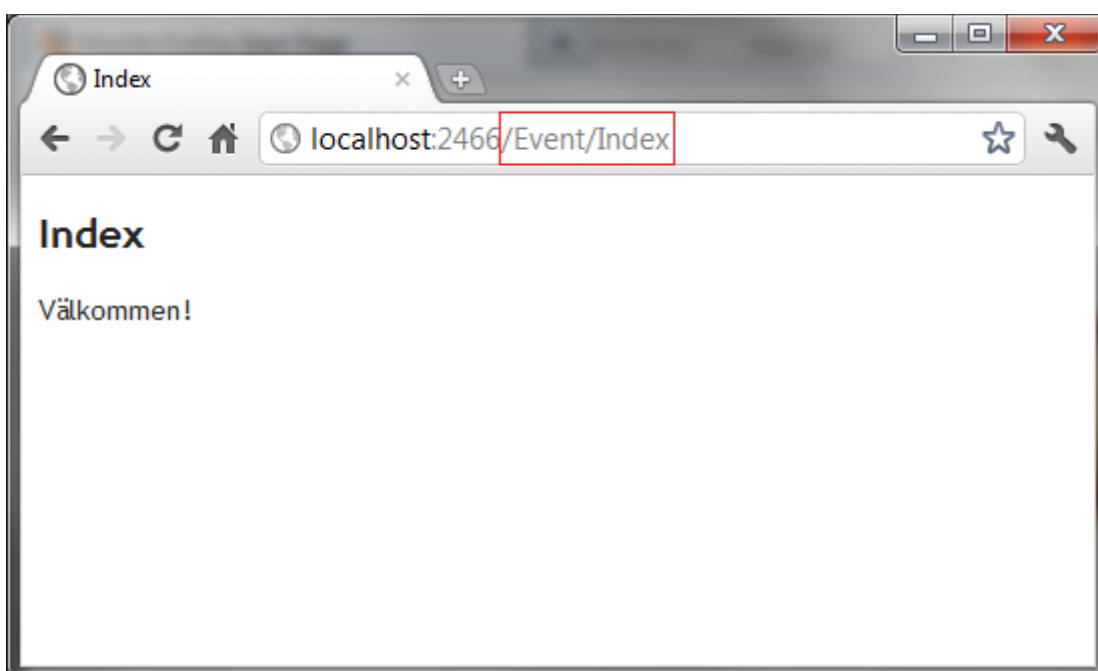
När vi visar upp vyn i browsern visas meddelandet upp:



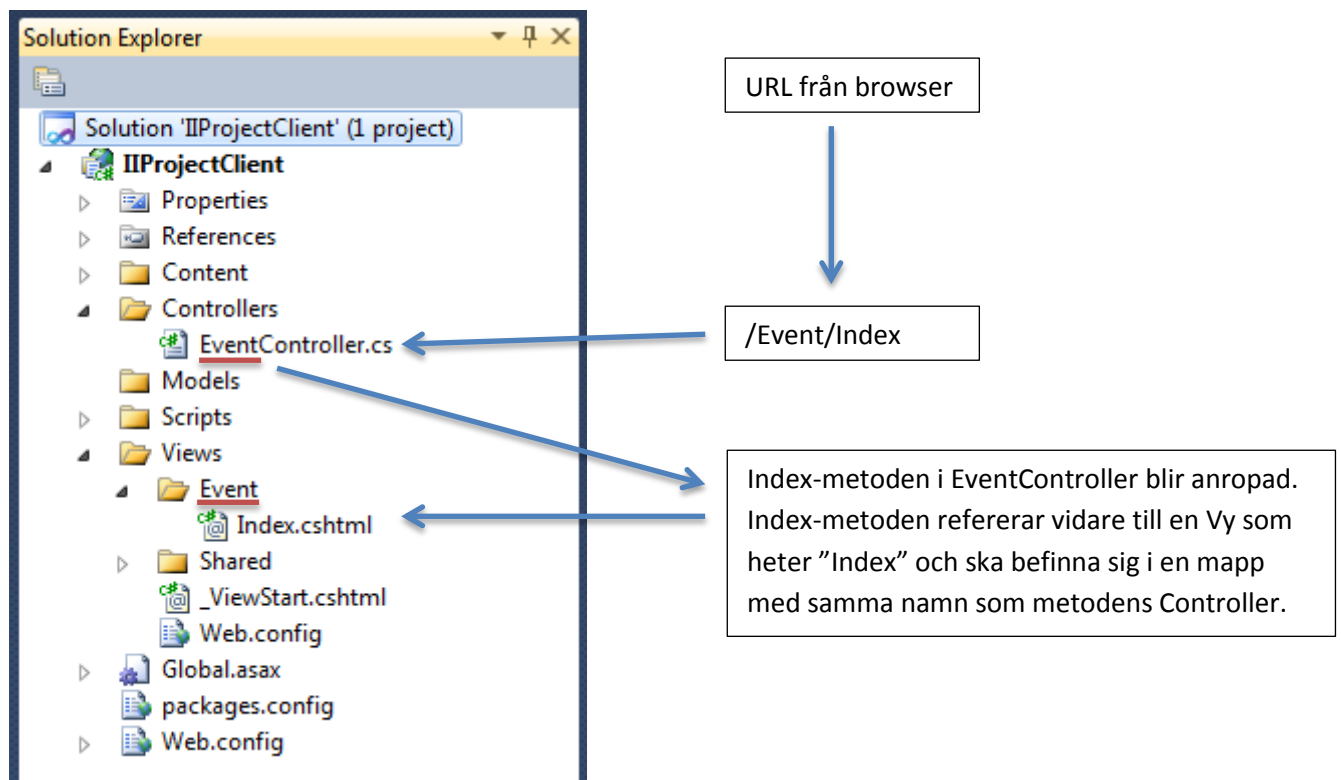
Routing

För traditionella webbsidor refererar browsern till en fysisk fil exempelvis "www.testside.se/Testar/index.html" här ligger" alltså Index.html filen i mappen "Testar" som befinner sig på servern för "www.testside.se".

ASP.NET MVC fungerar lite annorlunda som du kanske redan misstänkt genom att vi använder oss av controllers, när vi från en browser ansluter till en MVC.NET applikation refererar vi inte en fysisk fil utan till en metod i en controller. Vilken controller och vilken metod avgörs av vad som skrivits in som URL i browserns adressfält. Nedan pekas browsern mot localhost som kör MVC.NET applikationen, sedan kommer "/Event/Index" vilket refererar till Index-metoden i EventController.



Nedan visas flödet från anropet från browsern ovan, genom controller -> metod -> vy



Ifall ingen ytterligare information anges efter host i URL'n skickas en förfrågan automatiskt till "/Home/Index", detta är definierat i filen "App_Start/RouteConfig.cs". Nedan definieras detta om till att rikta dessa anrop till "/Event/Index" istället.

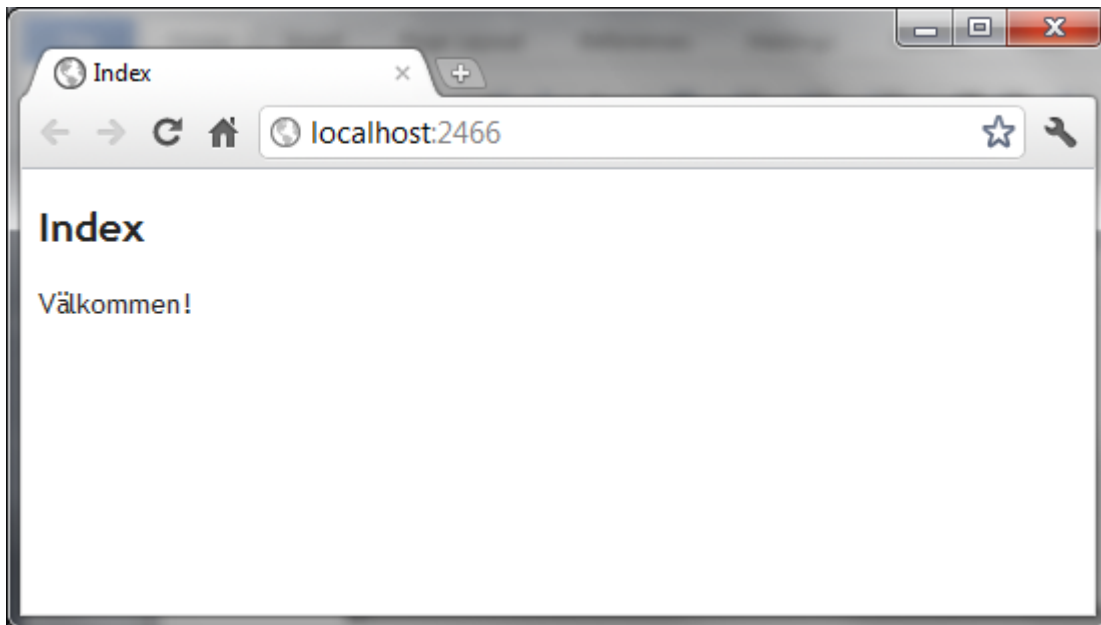
```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        "Default", // Route name
        "{controller}/{action}/{id}", // URL with parameters
        new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter defaults
    );
}

public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

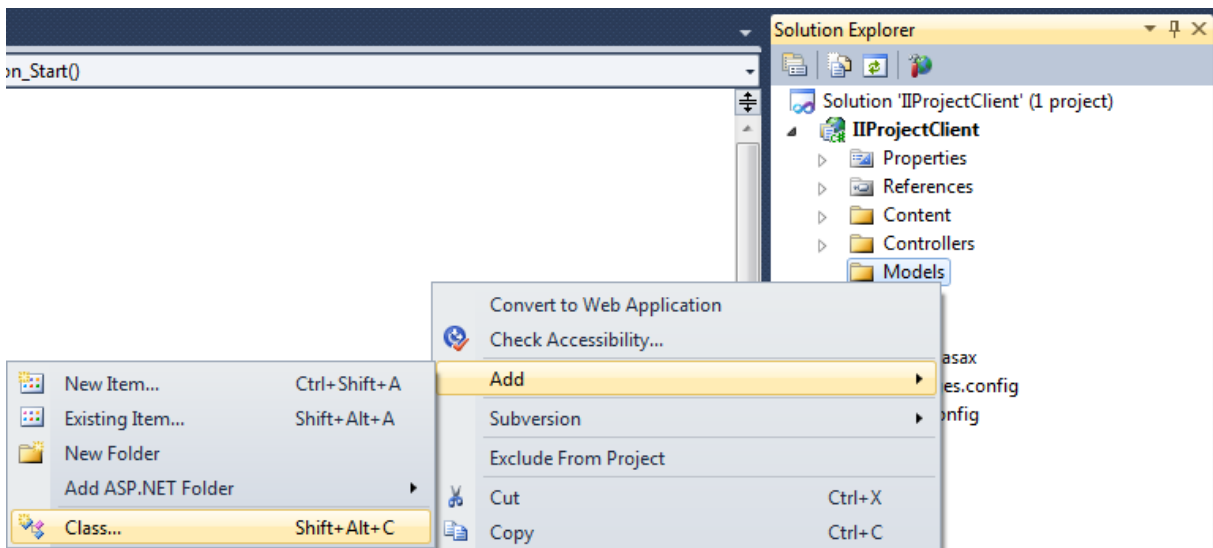
    routes.MapRoute(
        "Default", // Route name
        "{controller}/{action}/{id}", // URL with parameters
        new { controller = "Event", action = "Index", id = UrlParameter.Optional } // Parameter defaults
    );
}
```


Ifall en browser pekar på applikationen med enbart hostnamnet så anropas Index-metoden i EventController:

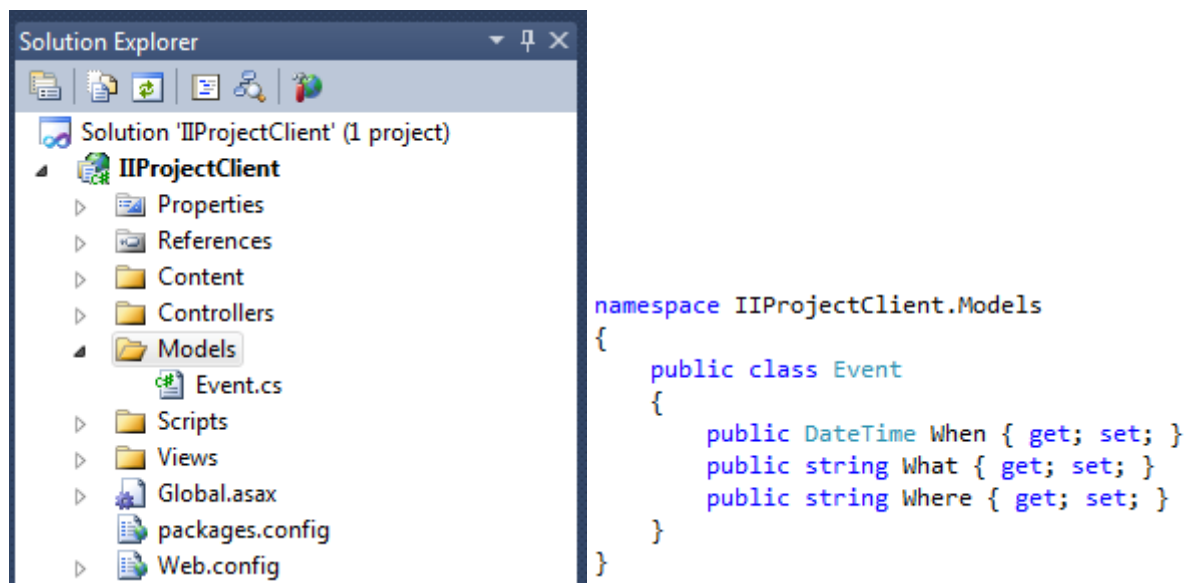


Models

M i MVC står för Model och är logiklagret i applikationen. Hit hör de klasser/komponenter som applikationen behöver för att fungera, nedan skapas en klass genom att högerklicka på mappen "Models" -> Add -> Class...



Klassen Event läggs till i Model-mappen, dess implementation ses nedan till höger.



Det går att skicka en modell till en vy, t.ex. som nedan så skickas en lista av Event-objekt till vyn. För att skicka med en modell till en vy används en överlagring av View-metoden.

```
public ActionResult Events()
{
    List<Event> eventList = new List<Event>()
    {
        new Event() { What = "Train", When = DateTime.Parse("2011/12/7 12:30:00"), Where = "Uppsala" },
        new Event() { What = "Car", When = DateTime.Parse("2011/11/22 17:55:00"), Where = "Stockholm" },
        new Event() { What = "Cat", When = DateTime.Now, Where = "Ekonomikum" }
    };

    return View(eventList);
}
```

▲ 1 of 8 ▼ View() Controller.View()
Creates a System.Web.Mvc.ViewResult object that renders a view to the response.

Strongly-typed views & scaffold templates

När man skapar en vy kan man bestämma om vyn ska vara "strongly-typed", alltså bunden till en viss modell. Nedan skapas vyn "Events" som ska ta emot en lista av Event-objekt (se metoden ovan). Högerklicka i metoden och välj Add View...

```
public ActionResult Events()
{
    List<Event> ev
    {
        Add View...
        Go To View
    }
}
```

Kryssa i rutan för "strongly-typed" och välj den modell som du vill använda i listan. Ifall din modell inte finns i listan se till att göra en "Build" (meny -> Build -> Build Solution) på applikationen som går igenom utan fel, sedan försök igen. Nedan väljs Event som modellen:

☒ Create a strongly-typed view

Model class:

Event (IIProjectClient.Models)

"Scaffolding template" är en teknik för att autogenerera generell kod för olika beteenden, på samma sätt som vi har olika templates för att skapa projekt (Windows Forms applikation kontra en MVC applikation) så går det att använda en template för att skapa vyer i MVC.NET. Vi skickar en lista av Event-objekt till vyn så vi väljer "List" i listan.

Scaffold template:

Empty

Create

Delete

Details

Edit

Empty

List

Fullständiga "Add View"-fönstret:

Add View

View name: Events

View engine: Razor (CSHTML)

☒ Create a strongly-typed view

Model class: Event (IIProjectClient.Models)

Scaffold template: List

☒ Reference script libraries

☐ Create as a partial view

☒ Use a layout or master page:

(Leave empty if it is set in a Razor _viewstart file)

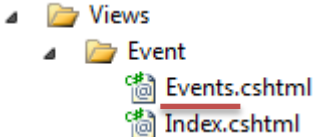
ContentPlaceHolder ID: MainContent

Add Cancel

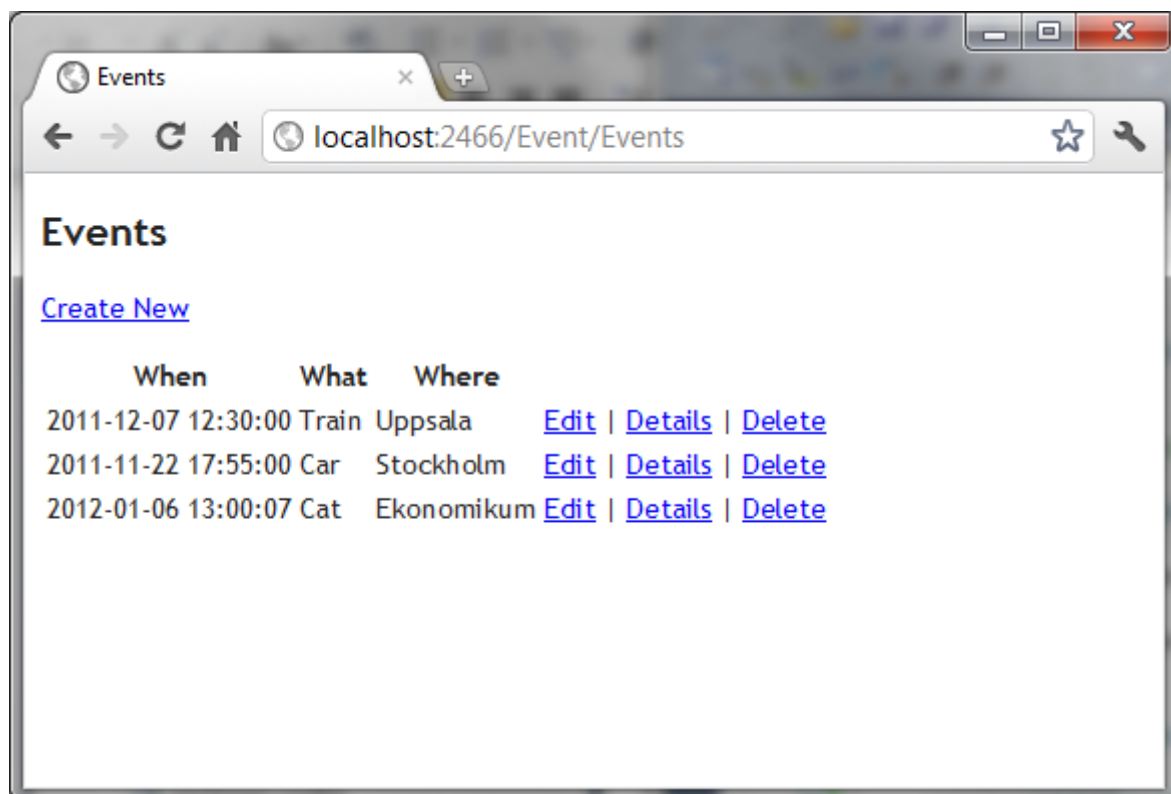
Detta skapar en vy "Events" som placeras i rätt mapp (nedan till höger) som även har autogenererad kod för att visa upp en lista av Event-objekt. Modellen som skickades till vyn var en Lista av Event-objekt därför skapades en foreach-loop som loopar över egenskapen "Model" (Model är en egenskap i en vy som håller det objekt, modellen, som skickats till vyn) och skriver ut information om varje Event-objekt.

```
@model IEnumerable<IIPProjectClient.Models.SimpleEvent>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.When)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.What)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Where)
        </td>
    </tr>
}
```



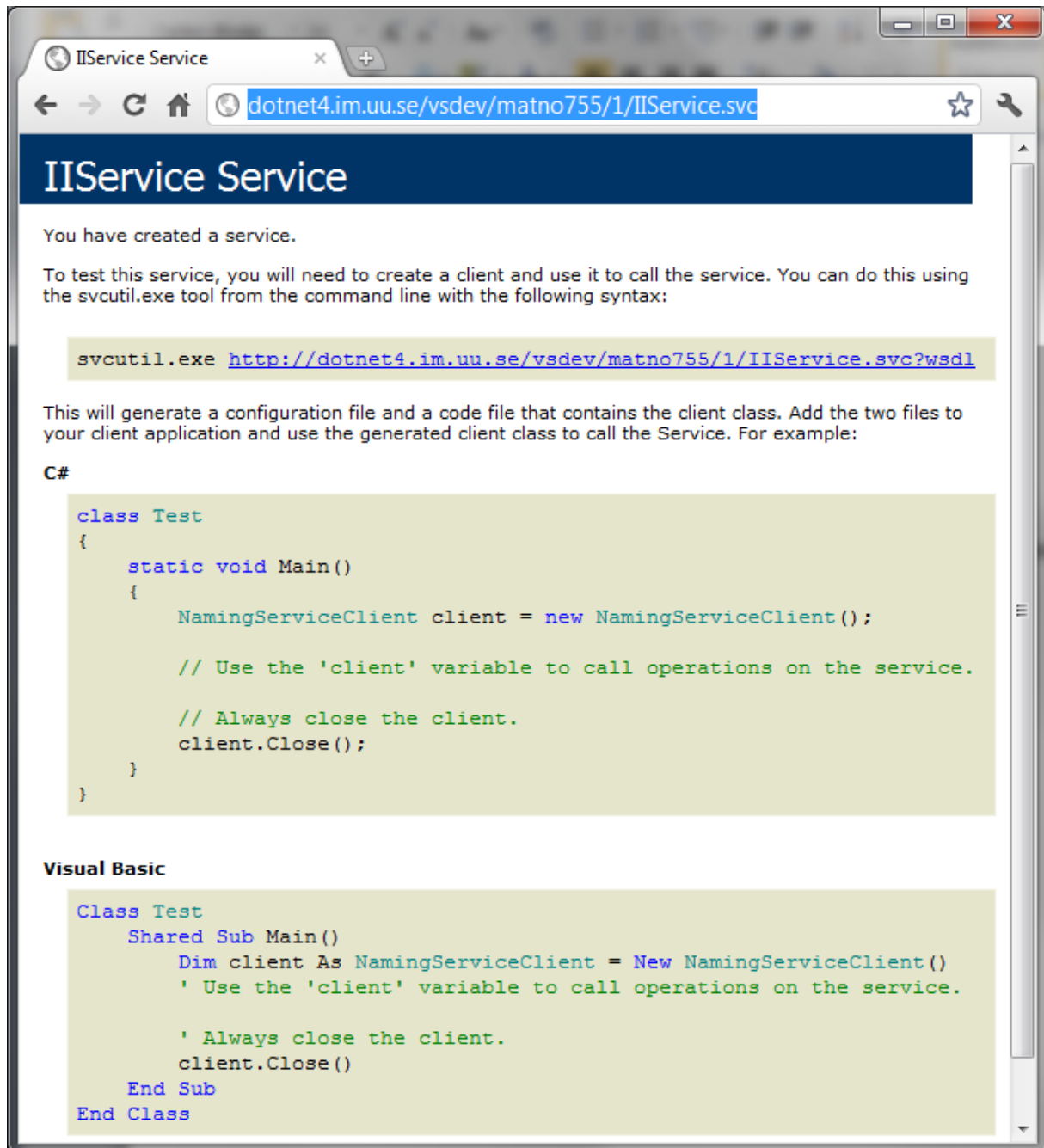
Ifall vi pekar browsern mot URL /Event/Events skapas listan med Event-objekt i kontrollern som sedan skickas till vyn där den loopas över och skriver ut informationen, nedan visas resultatet:



Service References

För att använda sig av en WCF Service från en .NET applikation (i dessa instruktionerna en ASP.NET MVC applikation) lägger man till en "Service Reference". För att göra detta behöver man först veta adressen till den service som ska användas, samt att den är igång. Nedan pekar browsern på en WCF

Service som är igång, att en service är igång ser man på att det visas information om hur tjänsten används och en länk till dess WSDL.



IIService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://dotnet4.im.uu.se/vsdev/matno755/1/IIService.svc?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        NamingServiceClient client = new NamingServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

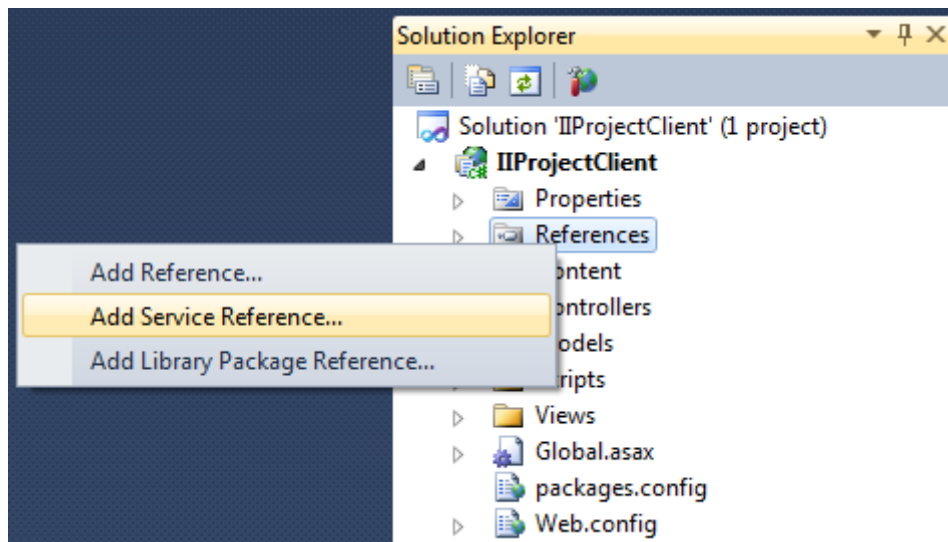
Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As NamingServiceClient = New NamingServiceClient()
        ' Use the 'client' variable to call operations on the service.

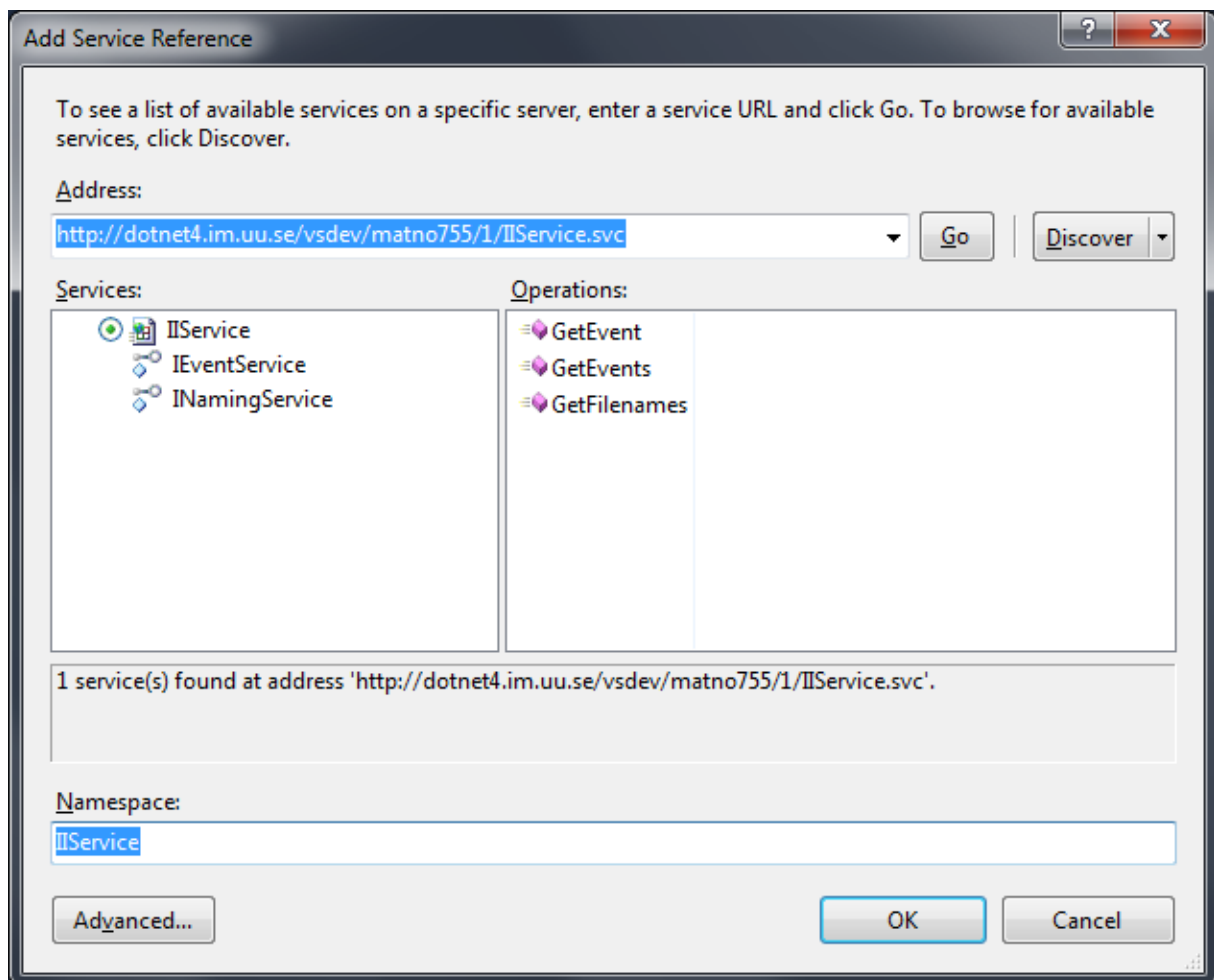
        ' Always close the client.
        client.Close()
    End Sub
End Class
```

Lägg till en Service Reference

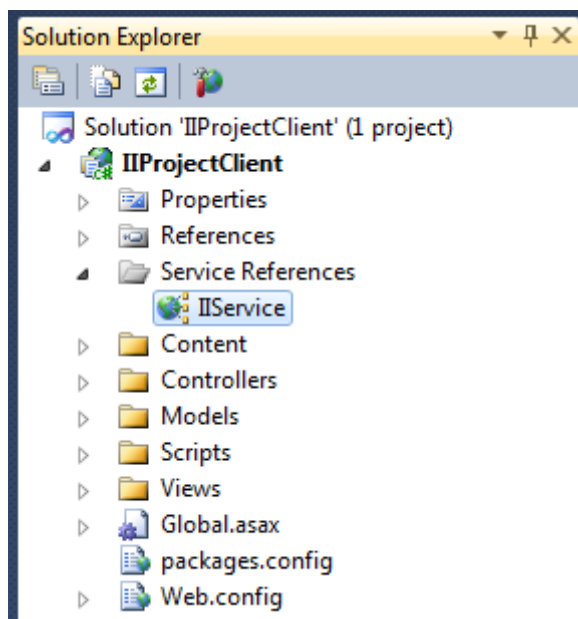
För att använda servicen behövs en referens till denna, detta görs genom att högerklicka på References-mappen -> Add Service Reference...



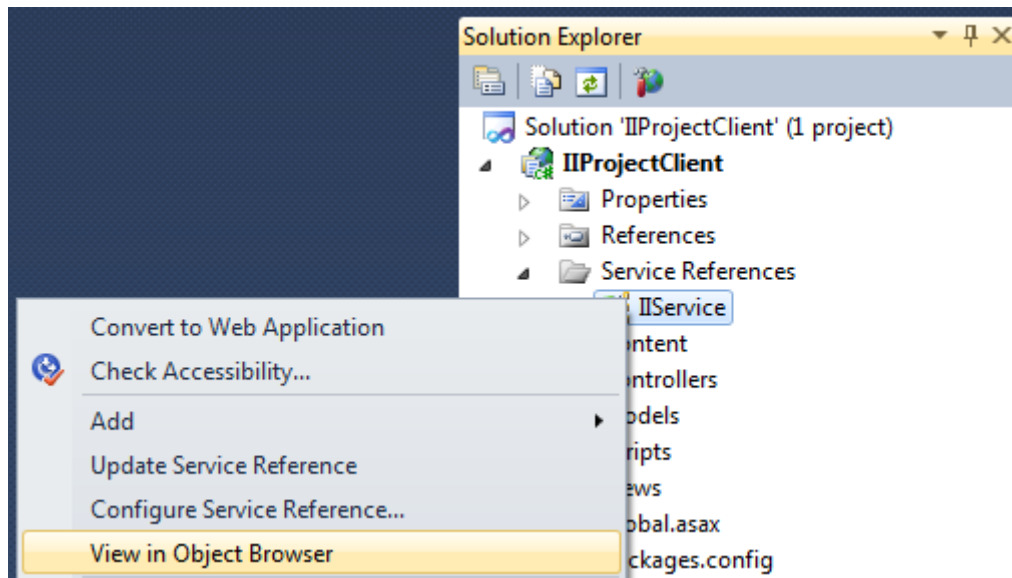
Rutan nedan visas, kopiera in adressen till servicen i adressfältet och tryck på "Go", ifall servicen finns i samma projekt som klienten går det bra att trycka på knappen "Discover" för att hitta den servicen. Nedan kopierades ovanstående adress in och en service med namnet IService hittades. Ange ett namespace som de klasser som Visual Studio skapar för att kommunicera med servicen ska skapas inom.



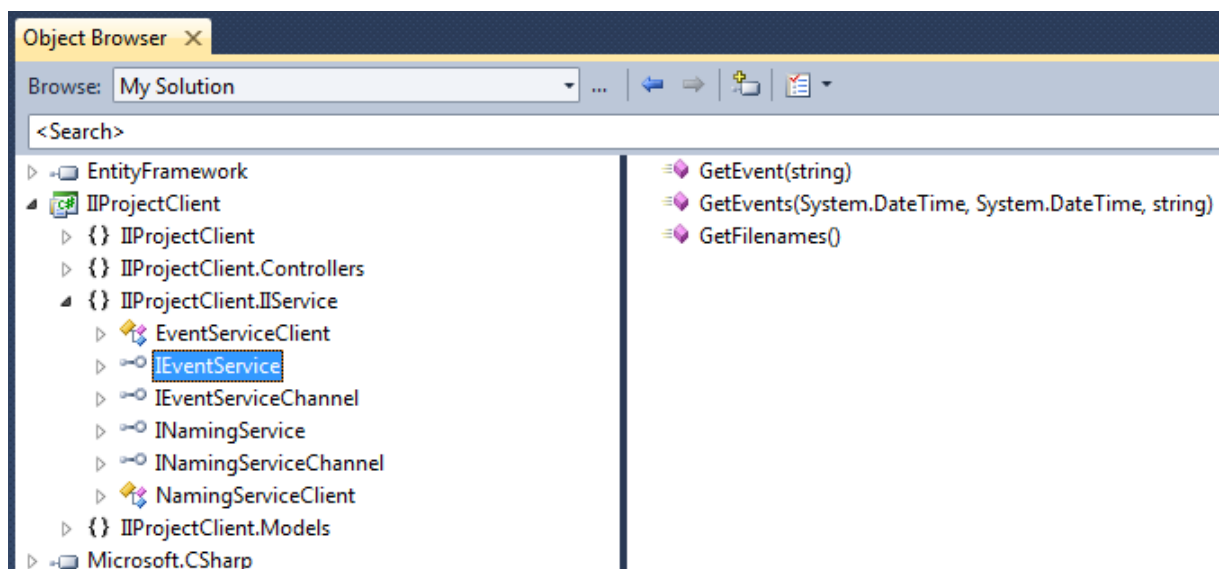
En mapp "Service References" skapas och referensen till servicen läggs till.

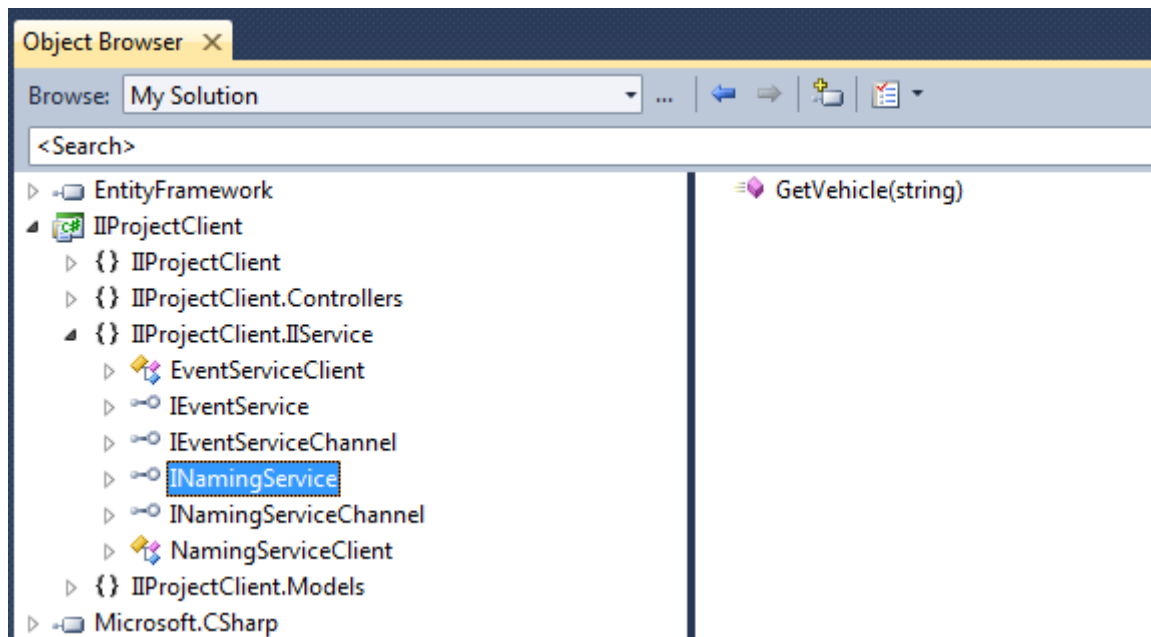


För att närmare undersöka vad för metoder, typer och annat som förmedlas genom servicen högerklicka -> View in Object Browser.



Under det tidigare valda namespace "IService" har det skapats ett antal klasser och interface, dessa skapas utifrån WSDL för servicen och gör det enkelt att kommunicera med servicen. Den här tjänsten har två kontrakt, de representeras genom interfacen IEventService och INamingService, de visas nedan med sina respektive metoder.





Använda servicen

Den kod som skapats ger möjligheten att skapa objekt från en proxy-klass som hanterar kommunikationen med servicen. En proxy-klass skapas för varje service-kontrakt som finns, alltså i det här fallet skapas både EventServiceClient och NamingServiceClient. Den autoskapade koden gör det lika enkelt att anropa servicen som det är att anropa en metod för vilken annan klass som helst. Nedan används EventServiceClient för att anropa metoden GetFileNames som inte tar några argument och returnerar en array av String-objekt.

```
public ActionResult FilesFromService()
{
    EventServiceClient client = new EventServiceClient();

    IEnumerable<string> filenames = client.GetFileNames()
    string[] EventServiceClient.GetFileNames()
}
}
```

Fullständig metod i EventController som skickar hela listan med filnamn till vyn. Kom ihåg att stänga proxy-klienten när anropet är klart. (Den här koden är inte helt fullständig och hanterar inte några fel som kan ske i själva anropet, exempel på detta ges nedan)

```
public ActionResult FilesFromService()
{
    EventServiceClient client = new EventServiceClient();

    IEnumerable<string> filenames = client.GetFileNames();

    client.Close();
    return View(filenames);
}
```

Stäng proxy-klienten när anropet är klart.

En vy som är bunden mot `IEnumerable<string>` dvs en lista av `String`-objekt och som för varje sådan sträng skriver ut den.

```
@model IEnumerable<string>
```

Definierar att modellen är av typen `IEnumerable<string>`

```
@{  
    ViewBag.Title = "FilesFromService";  
}
```

```
<h2>FilesFromService</h2>
```

```
@foreach (string filename in Model)  
{  
    <p>@filename</p>  
}
```

För varje `String`-objekt skriv ut texten i en paragraf-tag.

Nu testkörs metoden och uppenbarligen blev det ett fel, närmare sagt ett `CommunicationException` på grund av att det var för mycket information som skickades till klienten från servicen. Det är inte det underliggande protokollet som inte kan skicka informationen utan det är en standardinställning i `Web.config/App.config` (beroende på applikationstyp) som den gränsen är satt.

```
public ActionResult FilesFromService()  
{  
    EventServiceClient client = new EventServiceClient();  
    IEnumerable<string> filenames = client.GetFilesNames();  
    client.Close();  
    return View(filenames);  
}
```

CommunicationException was unhandled by user code ✕

The maximum message size quota for incoming messages (65536) has been exceeded. To increase the quota, use the `MaxReceivedMessageSize` property on the appropriate binding element.

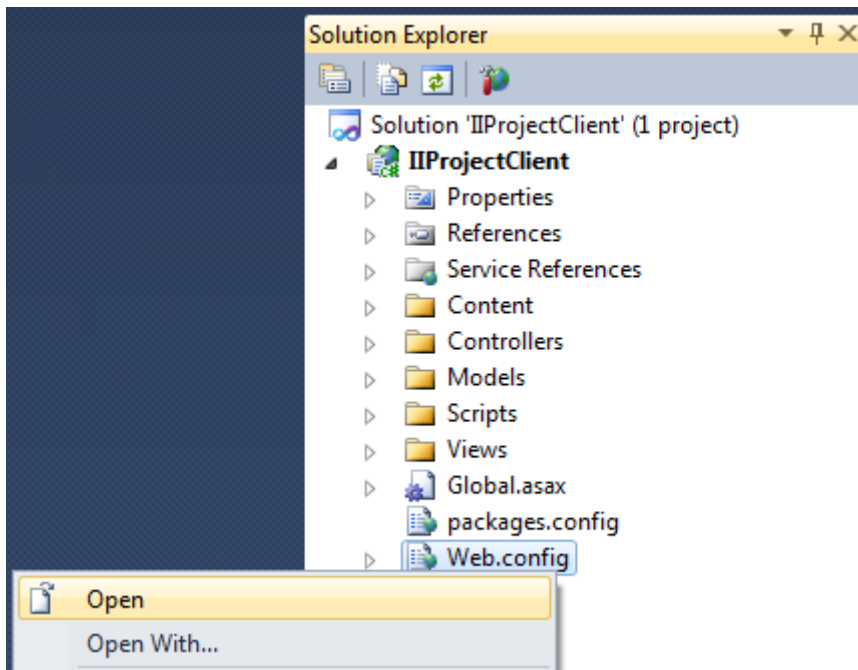
Troubleshooting tips:
[Get general help for this exception.](#)
[Get general help for the inner exception.](#)

[Search for more Help Online...](#)

Actions:
[View Detail...](#)
[Enable editing](#)
[Copy exception detail to the clipboard](#)

Konfigurera service bindings

För att konfigurera att klienten får ta emot tillräckligt mycket information för att undslippa ovanstående fel så öppna Web.config-filen som finns i rot-nivå i klienten (App.config för icke webbapplikationer).



XML konfigureringsfilen för applikationen och även kommunikationen till servicen öppnas, ändra värdet för noden "maxReceivedMessageSize" i bindningen med namnet "WSHttpBinding_IEventService" från standardvärdet på 65536 till ett högre värde, för den här tjänsten går det bra med att öka till 6553600.

```

<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding name="WSHttpBinding_INamingService" closeTimeout="00:01:00"
        openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        bypassProxyOnLocal="false" transactionFlow="false" hostNameComparisonMode="StrongWildcard"
        maxBufferPoolSize="524288" maxReceivedMessageSize="65536" messageEncoding="Text"
        textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">
        <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
          maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        <reliableSession ordered="true" inactivityTimeout="00:10:00"
          enabled="false" />
        <security mode="None">
          <transport clientCredentialType="Windows" proxyCredentialType="None"
            realm="" />
          <message clientCredentialType="Windows" negotiateServiceCredential="true" />
        </security>
      </binding>
      <binding name="WSHttpBinding_IEventService" closeTimeout="00:01:00"
        openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        bypassProxyOnLocal="false" transactionFlow="false" hostNameComparisonMode="StrongWildcard"
        maxBufferPoolSize="524288" maxReceivedMessageSize="65536" messageEncoding="Text"
        textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">
        <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
          maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        <reliableSession ordered="true" inactivityTimeout="00:10:00"
          enabled="false" />
        <security mode="None">
          <transport clientCredentialType="Windows" proxyCredentialType="None"
            realm="" />
          <message clientCredentialType="Windows" negotiateServiceCredential="true" />
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
  <client>
    <endpoint address="http://dotnet4.im.uu.se/vsdev/matno755/1/IISvc.svc/NamingService"
      binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_INamingService"
      contract="IISvc.INamingService" name="WSHttpBinding_INamingService">
      <identity>
        <dns value="localhost" />
      </identity>
    </endpoint>
    <endpoint address="http://dotnet4.im.uu.se/vsdev/matno755/1/IISvc.svc/EventService"
      binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_IEventService"
      contract="IISvc.IEventService" name="WSHttpBinding_IEventService">
      <identity>
        <dns value="localhost" />
      </identity>
    </endpoint>
  </client>
</system.serviceModel>

```

Ändra värdet för noden maxReceivedMessageSize till 6553600:

```

<binding name="WSHttpBinding_IEventService" closeTimeout="00:01:00"
  openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
  bypassProxyOnLocal="false" transactionFlow="false" hostNameComparisonMode="StrongWildcard"
  maxBufferPoolSize="524288" maxReceivedMessageSize="6553600" messageEncoding="Text"
  textEncoding="utf-8" useDefaultWebProxy="true" allowCookies="false">

```

Nedan visas resultatet av anropet till servicen som gjordes på sida 17.

