USE SERVICESTACK WEB SERVICE IN VNEXT VS 2015 CTP WITH ANGULAR JS

Author: David LIzhong Huang

Date: 10, Feb, 2015

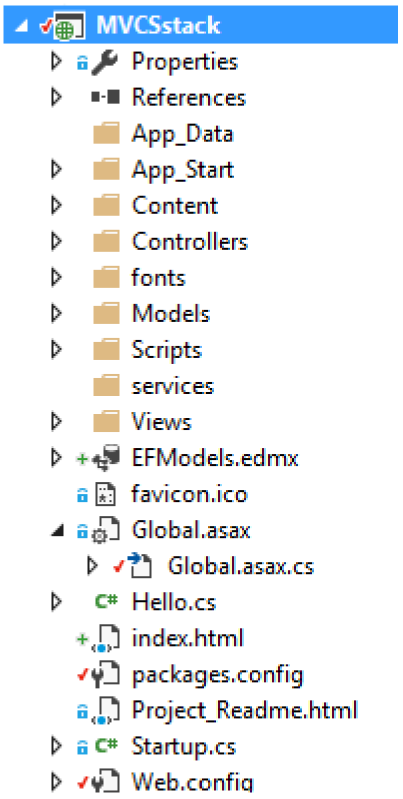
1, INTRODUCTION

Ultimate VS2015 CTP 5 is the latest version of Visual Studio 2015 that will be the next version of Visual Studio IDE. Microsoft merges Node JS into .Net to allow this IDE to develop cross platform applications from front end to backend. It is an exciting platform I love to use. VNext 2015 embeds Asp.Net Web API in MVC. For a standalone project, Web API is enough to be used as a data proxy layer. However, WCF gets survived in enterprise SOA infrastructure. Industry invents a new web service wheel try to merge both together that is ServiceStack. It needs some time to get used to using this new Service Stack service in VNext VS 2015 web application. This document intends to show you how I can use ServiceStack in VNext VS 2015 to bring data from SQL server to HTML page.

2, GET STARTED

1, Install VS 2015 CTP 5 from Microsoft

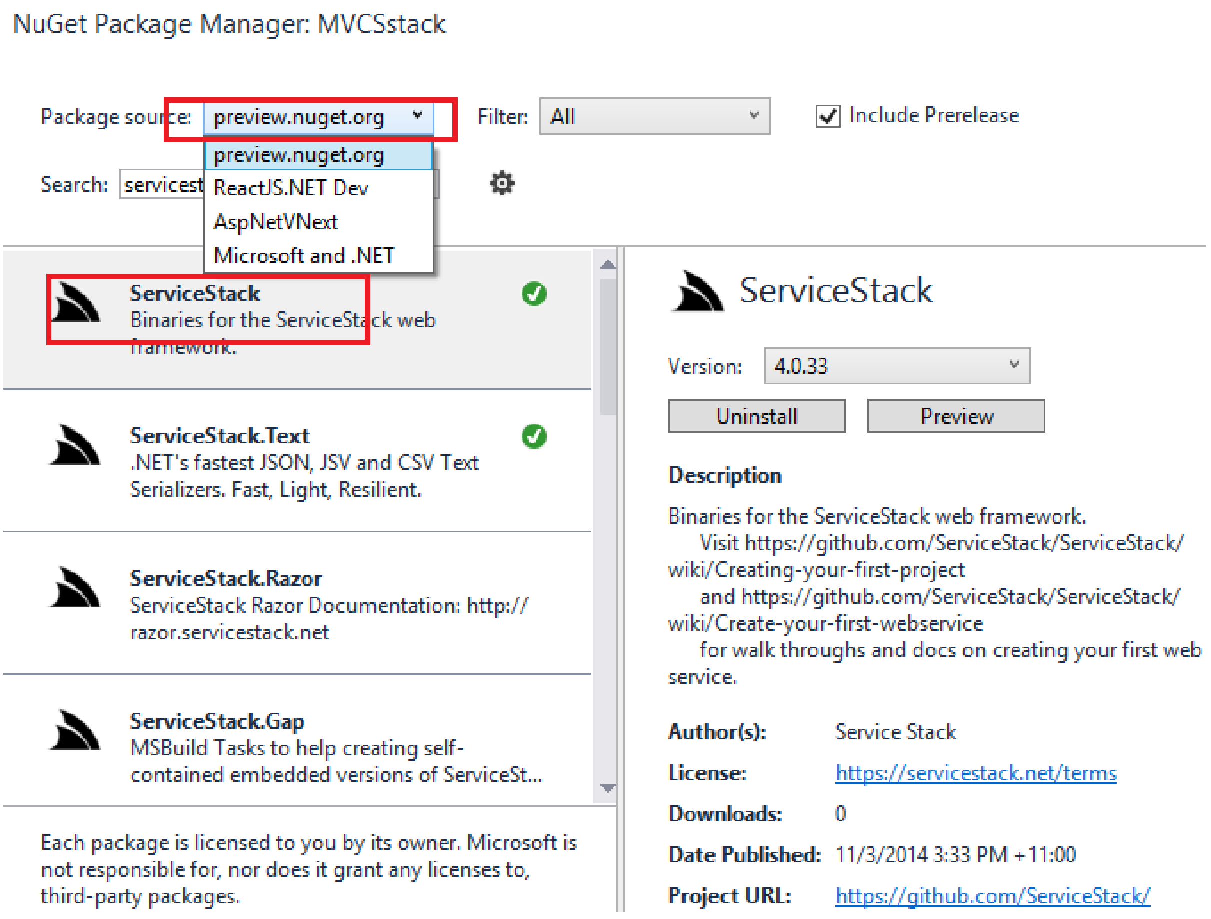
It takes some time about one or two days for me to get this correct installation. Now try to create a new MVC6 web site in VNext as below



It is a good practice to add the new project into source control such as TFS and Git. This will allow us to roll back the wrong version with good history version.

2, Nuget download ServiceStack

Use Nuget manager to search and download the latest ServiceStack package from preview.Org.Net see below


NuGet Package Manager: MVCSstack

Package source: preview.nuget.org

Filter: All

✓ Include Prerelease

Search: servicest

ServiceStack ✓
Binaries for the ServiceStack web framework.

ServiceStack.Text ✓
.NET's fastest JSON, JSV and CSV Text Serializers. Fast, Light, Resilient.

ServiceStack.Razor
ServiceStack Razor Documentation: http://razor.servicestack.net

ServiceStack.Gap
MSBuild Tasks to help creating self-contained embedded versions of ServiceSt...

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

ServiceStack

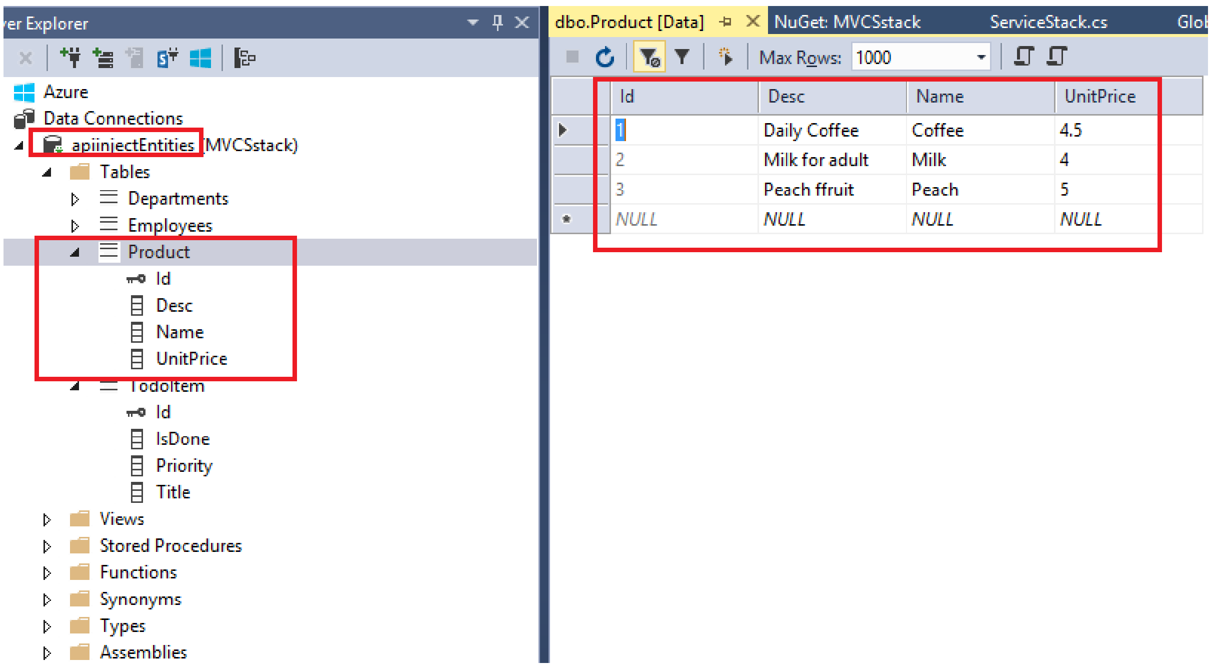
Version: 4.0.33

Description

Binaries for the ServiceStack web framework.
Visit https://github.com/ServiceStack/ServiceStack/wiki/Creating-your-first-project
and https://github.com/ServiceStack/ServiceStack/wiki/Create-your-first-webservice
for walk throughs and docs on creating your first web service.

Author(s): Service Stack
License: https://servicestack.net/terms
Downloads: 0
Date Published: 11/3/2014 3:33 PM +11:00
Project URL: https://github.com/ServiceStack/

3, Create a new SQL database in Visual Studio 2015 as below



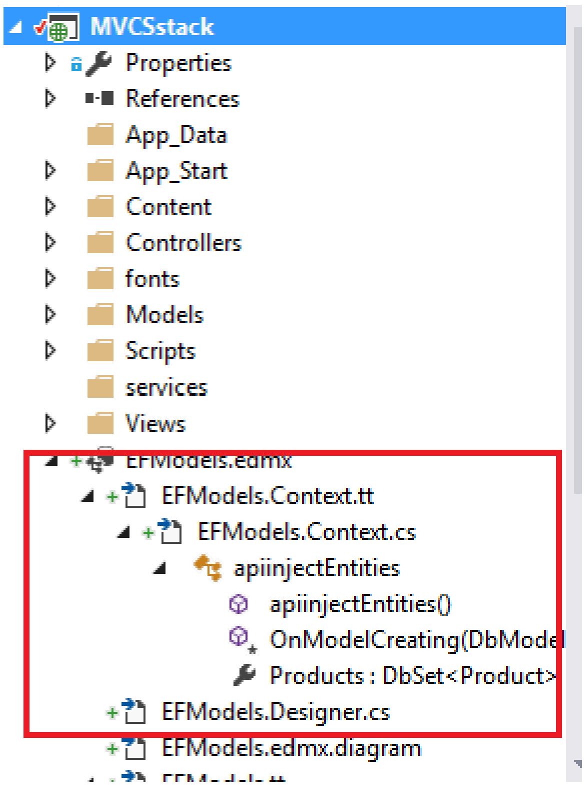
Id	Desc	Name	UnitPrice
1	Daily Coffee	Coffee	4.5
2	Milk for adult	Milk	4
3	Peach ffruit	Peach	5
NULL	NULL	NULL	NULL

4, Update SQL database connection string and ServiceStack configuration in Web.config as below

```
<add name="apiinjectEntities"
connectionString="metadata=res://*/EFModels.csdl|res://*/EFModels.ssdl|res://*/EFModel
s.msl;provider=System.Data.SqlClient;provider connection string=&quot;data
source=.\sqlexpress;initial catalog=apiinject;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;"
providerName="System.Data.EntityClient" /></connectionStrings>

<httpHandlers>
      <add path="*" type="ServiceStack.HttpHandlerFactory, ServiceStack" verb="*" />
</httpHandlers>
```

Now this web application has been configured as a ServiceStack enabled web application we can create ans consume this service in application without Web API and WCF.

5, Create ADO.Net Entity Data Model and select "generate model from database" option to create an Entity ServiceStack will use as below



6, This Entity Data Model has been ready for ServiceStack to consume. Now it is time to create exciting ServiceStack web service here.

Create a product class as a service contract as WCF

```
    [Route("/product")]
    public class ProductEF
    {
        public int Id { get; set; }
        public string Desc { get; set; }
        public string Name { get; set; }
        public double UnitPrice { get; set; }

    }
```

ServiceStack Route engine will find out this contract request when we update the URL in client calls. E.g. http://localhost/servicestackexample/product URL. The new thing ServiceStack invented here is Response class as below

```
  public class ProductReponse
    {
        public List<ProductEF> products { get; set; }

    }
```

This class will return results we needed such as whole product lists here. Therefore, the concept here is really handy and straight forward. We create request and will send this request to servicestack, servicestack will response via this response class object. However, How can we start this service before sending request and returning response? We create a service hosting class see example below to take this responsibility.

```
   public class ProductService : IService
    {
        apiinjectEntities db = new apiinjectEntities();
        public object Any(ProductEF request)
        {
            var allProducts = db.Products;

            var allProductList = new List<ProductEF>();

            allProductList.Clear();

            foreach (var item in allProducts)
            {
                request = new ProductEF();
                request.Desc = item.Desc;
                request.Id = item.Id;
                request.Name = item.Name;
                request.UnitPrice = item.UnitPrice;
                allProductList.Add(request);
            };

            return new ProductReponse
            {
                products = allProductList
            };
        }

    }
```

This ProductService is inherited from IService service stack and bring Entity Data Model in to connect this service to SQL database and will return all products in database back to caller. Because ProductEF is the service model class, Therefore, reformatting is necessary. This service now is ready for client to call.

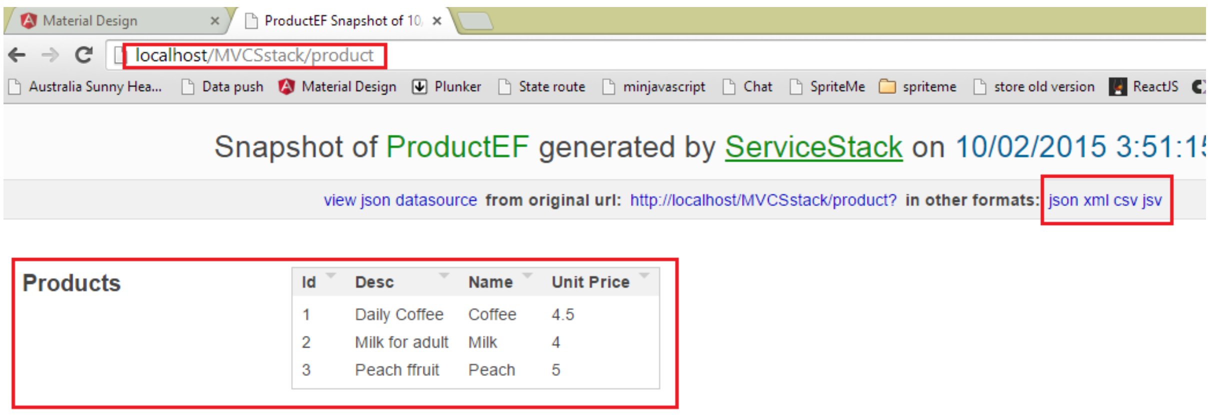
7, Update Global.aspx.cs to allow this service gets started in application level. Global.aspx.cs is updated as below

```
    public class MvcApplication : System.Web.HttpApplication
    {
        public class AppHost : AppHostBase
        {
          public AppHost() : base("Product Web Services",
typeof(ProductService).Assembly) { }

        public override void Configure(Funq.Container container){
                //register any dependencies your services use, e.g:
                //container.Register<ICacheClient>(new MemoryCacheClient());
            }
        }

        protected void Application_Start()
        {
            //AreaRegistration.RegisterAllAreas();
            //FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
          //  RouteConfig.RegisterRoutes(RouteTable.Routes);
            //BundleConfig.RegisterBundles(BundleTable.Bundles);

            new AppHost().Init();  //servicestack will passedby those config
        }
    }
```

We need to disable some configuration in Application_Start() that may conflict with Route() mechanism in ServiceStack. Another option is initiate this AppHost class in StartUp.cs class in the project as below

```
    public class AppHost : AppHostBase
    {
       public AppHost() : base("Product Web Services",
typeof(ProductService).Assembly) { }

        public override void Configure(Funq.Container container)
        {
        }
    }



 public partial class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            ConfigureAuth(app);
            new AppHost().Init();

        }

    }
```

Now this service gets started when application runs. The result is shown as below



Snapshot of ProductEF generated by ServiceStack on 10/02/2015 3:51:1

view json datasource from original url: http://localhost/MVCSstack/product? in other formats: json xml csv jsv

Products

Id	Desc	Name	Unit Price
1	Daily Coffee	Coffee	4.5
2	Milk for adult	Milk	4
3	Peach ffruit	Peach	5

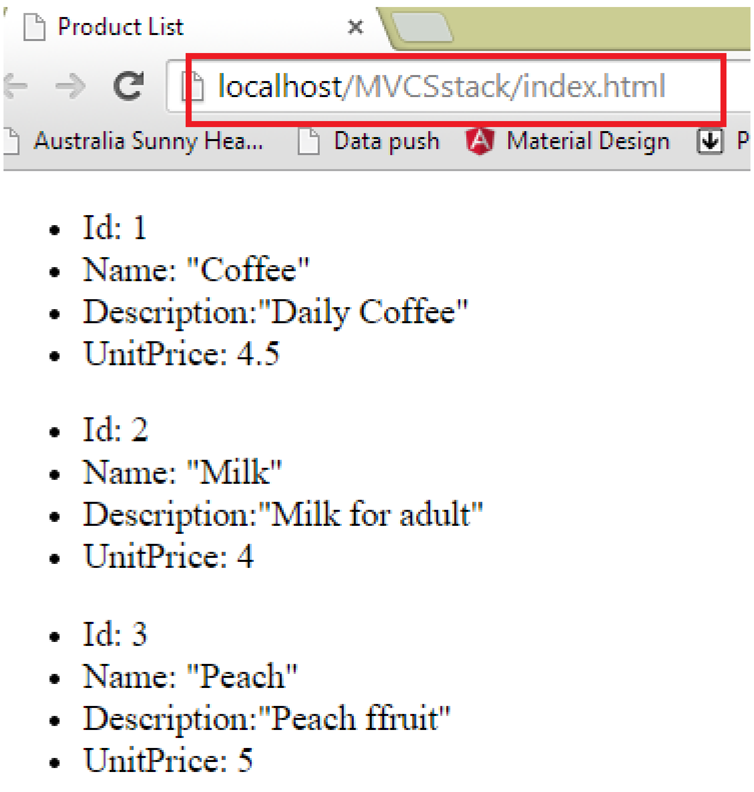
Copy this web service link, we will use it in getJson () call in HTML page and Http service calls in Angualr JS as below

8, Consume ServiceStack in HTML page

We used Jquery simply consume this web service in HTML page as below

```
<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Product List</title>
</head>
<body>
    <div id="datashow">

    </div>
    <script src="http://localhost/MVCSstack/Scripts/jquery-1.10.2.min.js"></script>
    <script>
        $(function () {
             var url = "http://localhost/MVCSstack/product?format=json";
             $.getJSON(url, function (data)
             {
                 $.each(data.products, function (i, value)
                 {
                     $("#datashow").append("<ul><li>Id: " + JSON.stringify(value.Id) +
"</li><li>Name: " + JSON.stringify(value.Name) + "</li><li>Description:" +
JSON.stringify(value.Desc) + "</li><li>UnitPrice: " + JSON.stringify(value.UnitPrice)
+ "</li></ul>");
                 });

             });
        });
    </script>
</body>
</html>
```



- Id: 1
- Name: "Coffee"
- Description:"Daily Coffee"
- UnitPrice: 4.5

- Id: 2
- Name: "Milk"
- Description:"Milk for adult"
- UnitPrice: 4

- Id: 3
- Name: "Peach"
- Description:"Peach ffruit"
- UnitPrice: 5

3, Summary

Vs 2015 CTP seems working well as we used it, it is stable and easy to learn if we keep using visual studio IDEs and Node JS. The big change in VS 2015 is DI enabled. We can register and inject services such as sql data access services in startup.cs and injected into MVC controllers as an interface. This will meet the SOLID design pattern very well. DI in .Net just is amazing. I love this.