

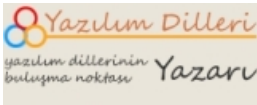
Burak Selim Senyurt

Matematik Mühendisi bir .NET Severin Yazıları...

01 Arşiv İletişim Fotoğrafçılık TFi Almanac YDBH Java 24 Aylık Bülten Ben Kimim?



Connected System
Developer
2008|2009|2010
Visual C#
2006|2007



<< [Kendi Web Part Bileşenlerimizi Gelistirmek - 2](#) | [WCF-Windows ve Windows Service Hosting](#) >>

WCF – Adım Adım IIS Hosting

Cumartesi, 28 Nisan 2007 21:00 by [bsenyurt](#)

Değerli Okurlarım Merhabalar,

Windows Communication Foundation ile ilişkili bir önceki [makalemizde](#) mimarinin detaylarına yakından bakmaya çalışmış ve örnek bir uygulama geliştirmiştik. Bu makalemizde ise bir **WCF Service** projesi geliştirmeye çalışacağız. Bir başka deyişle **HTTP** protokolünü kullanarak, **IIS** üzerinden bir .Net kütüphanesini servis olarak yayınlayacağız.



WCF servislerini **Self Hosting** ve **IIS Hosting** olmak üzere iki farklı şekilde yayınlayabiliyoruz. Bununla birlikte Self Hosting modeli kendi içerisinde **Console**, **Windows**, **Windows Service** ve Vista ile gelen **Windows Activation Service (WAS)** olmak üzere dört farklı seçenekten oluşmaktadır.

Ayrıca kendi tiplerimizi döndüren metodlar söz konusu olduğunda veri sözleşmelerini(**DataContract**) nasıl kullanabileceğimize yakından bakma fırsatı bulacağız. Bunlara ek olarak özellikle **.Net Remoting** mimarisinde bizi her zaman zorlayan konfigürasyon ayarlarının(Çoğu zaman ezberlemek zorunda olduğumuz ve hatırlamakta güçlük çektiğimiz), WCF içerisinde **Edit WCF Configuration** seçeneği yardımıyla Visual Studio 2005 içerisinde görsel olarak nasıl daha kolay hazırlanabileceğinin de incelemeye çalışacağız.

Aslında IIS(Internet Information Service) üzerinden çeşitli tipte istemcilerin (hangi platformda olduklarına bakılmaksızın) belirli fonksiyonellikleri kullanabilmesi amacıyla **Xml Web Servisleri** (Xml Web Services) oldukça fazla kullanılan bir **SOA (Service Oriented Architecture)** modelidir. Dolayısıyla WCF içerisinde de buna benzer bir model vardır ve **WCF Service** olarak adlandırılmaktadır. Visual Studio 2005 için, Framework 3.0' a yönelik eklentileri yüklediğinizde Web Site şablonları arasında **WCF Service** seçeneğinde gelmektedir. Ancak biz bunu kullanmayıp biraz daha ileriye gideceğiz ve alt yapı içerisinde yer alan parçaları daha net olarak görmeye çalışacağız. Konuyu daha iyi kavrayabilmek için örnek bir senaryo üzerinden gideceğiz ve bazı basit veritabanı işlemlerini ele alacağız. Gelin hiç vakit kaybetmeden işe başlayalım ve adım adım ilerleyelim.

Bildiğiniz gibi WCF tamamen nitelik(**attribute**) tabanlı bir mimariye dayanmaktadır. Bununla birlikte istemci tarafına yayınlanacak olan, başka bir deyişle **metadata** içeriği gönderilecek olan tip(type) aslında ilgili nitelikler ile imzalanmış bir sözleşme(**contract**) dir. Sözleşmeler genellikle bir arayüz şeklinde tasarlanmakta olup asıl işlevsellikleri yerine getirecek olan uzak nesne modellerinin(sınıflar) türetilmesinde de kullanılır.



.Net Remoting mimarisinden hatırlarsanız eğer; uzak nesneleri (**Remote Objects**) yayınlamada ele alınan modellerden birisi arayüz(Interface) kullanımını içermektedir. Burada temel amaç istemci tarafında, sunucuda örneklenen uzak nesne referansını taşıyabilecek bir tip olmasıdır. Bununla birlikte istemci tarafının kesin olarak uzak nesne fonksiyonelliklerinin nasıl çalıştığını görmesi gibi bir şans yoktur. (Elbette bu durum **MarshalByReference** nesneler için geçerlidir. Bildiğiniz gibi **MarshalByValue** tipler zaten sunucuda örneklenip istemci tarafına tüm varlığı ile serileştirilerek(Serialization) geçerler). Ayrıca, sunucu tarafındaki uzak nesne metodlarının iç yapılarında meydana gelecek değişiklikler sonucu, istemcilere yeniden dağıtım(**deployment**) işlemi uygulanmasına gerek kalmamaktadır.

Öyleyse ilk yapmamız gereken içerisinde servis sözleşmesini(**Service Contract**) ve gerekli fonksiyonelliklere ait tanımlamaları barındıracak bir arayüz(**interface**) tasarlamaktır. Örneğimizde, WCF servisimizi IIS üzerinden sunacağımızdan bahsetmiştik. Bu sebepten ilk olarak bir sınıf kütüphanesi(**class library**) geliştirmeli ve bu kütüphane içerisine servis sözleşmemizi(Service Contract), uzak nesnemizin modelini(class), veri sözleşmesine(Data Contract) sahip tipimizi katmalıyız. Sonrasında IIS üzerinden ilgili kütüphanemizi işaret edecek bir sanal klasör (virtual path) oluşturmamız. Yayınlama ortamı IIS olduğundan, WCF için gerekli konfigürasyon ayarlarını **web.config** dosyası içerisinde tutmamız gerekmektedir. Bu noktada konfigürasyon ayarlarını görsel bir arayüzü kullanarak kolay bir şekilde oluşturabiliriz. Diğer



Ara

 Aradığınız metni gir

Tag Bulutu

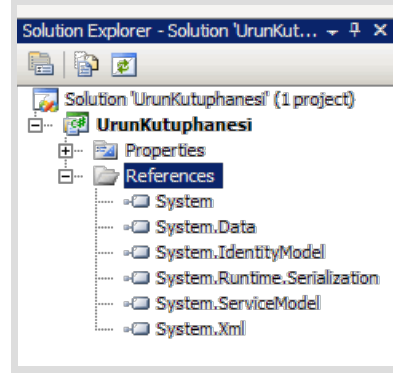
.Net .Net Framework .Net Framework 4.0 .Net Framework 4.5 .Net Remoting .Net Ria Services Ado.Net Ado.Net 2.0 Ado.Net Data Services Ado.Net Entity Framework Ado.Net Entity Framework 4.0 Asp.Net Asp.Net 2.0 Asp.Net 4.0 Asp.Net 4.5 Asp.Net Web Api Async Attribute Await Bizspark C# C# C# 2 C# 2.0 C# 3.0 C# 4.0 C# Temelleri Client Object Model Code First Development Delete Design Patterns Design Principles Dynamic Dynamic Language Runtime Entity Framework Excel Extension Methods Gelişim Atölyesi Html 5 Http Http Get Httpwebrequest Httpwebresponse Io Json Linq Linq To Sql List Microsoft Microsoft Teknoloji Günleri Msf Mvc Nedirtv Nedirtv?Com Nedirtv?Com Söyleşileri Non Soap Nosql Odata Oledb Oop Open Data Protocol Paralel Programming Plinq Reflection Rest Restful Screencast Scrum Seminer Silverlight Soap Sql Star Wars Task Task Parallel Library Team Foundation Server Tek Fotoluk Ipucu Tfs Tfs 2012 Tpl Transactions Visual Studio Visual Studio 2010 Visual Studio 2010 Ultimate Visual Studio 2012 Wcf Wcf 4.0 Wcf 4.5 Wcf Data Services Wcf Eco System Wcf Ria Services Wcf Service Wcf Webhttp Services Webcast Webhttp Services Webiner Wf Wf 4.0 Windows Communication Foundation Windows Forms Windows Server Appfabric Workflow Foundation Workflow Foundation 4.0 Workflow Services Wpf Xml Xml Web Services

Kategoriler

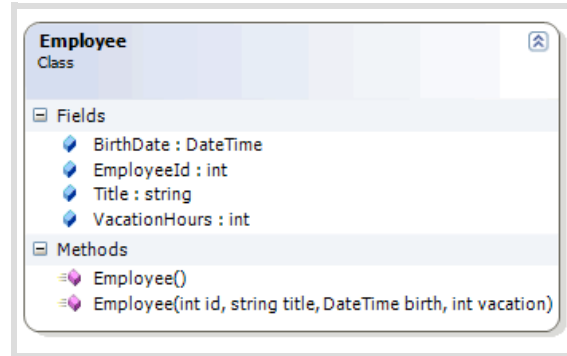
- .Net Framework 4.0
- .Net Framework 4.5

tarafından aynı web servislerinde olduğu gibi asmx uzantılı dosyaların görevini üstlenen svc uzantılı bir dosya oluşturmamız.

İlk olarak bilgisayarımızın herhangi bir yerinde konuşlandırılacak bir sınıf kütüphanesi (class library) projesi oluşturalım. Urunkutuphanesi isimli projemizi yine **.Net Framework 3.0** sonrası Visual Studio 2005 için gelen eklentilerden **WCF Service Library** olarak açabiliriz. Ama bu şablon yoksada önemli değildir. Çünkü yapmamız gereken **System.ServiceModel**, **System.Runtime.Serialization** ve **System.IdentityModel** assembly' larını referans edecek olan bir proje geliştirmektir. (Bu assembly' ların .Net Framework 3.0 ile geldiklerine dikkat edelim.)



Sınıf kütüphanemiz içerisinden, **AdventureWorks** içerisindeki tablolara ilişkin bazı hizmetler vereceğiz. Bunlardan biriside **HumanResources** şemasındaki **Employee** tablosundan herhangi bir çalışan bilgisini döndürecek bir metod olacaktır. Lakin bu metodun dönüşünde biz kendi geliştireceğimiz sınıfımıza ait bir nesne örneğini döndüreceğiz. Bu nedenle ilk olarak aşağıdaki gibi bir sınıfı, kütüphanemiz içerisine katarak işe başlayabiliriz.

**[DataContract]**

```
public class Employee
{
```

```
    [DataMember]
    public int EmployeeId;
```








```
    [DataMember]
    public string Title;
```

```
    [DataMember]
    public DateTime BirthDate;
```

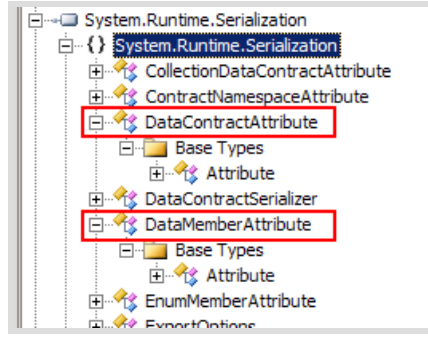
```
    [DataMember]
    public int VacationHours;
```

```
    public Employee()
    {
    }
    public Employee(int id, string title, DateTime birth, int vacation)
    {
        EmployeeId = id;
        Title = title;
        BirthDate = birth;
        VacationHours = vacation;
    }
}
```

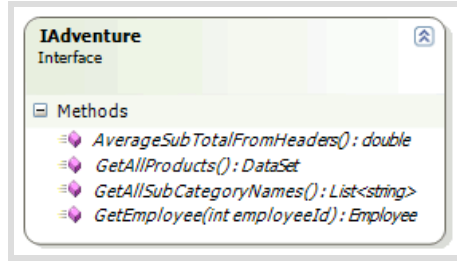
Employee isim sınıfımız çok basit olarak bir kaç public alan(**Field**) ile, aşırı yüklenmiş(**overload**) ve varsayılan yapıcı(**default constructor**) metodları içermektedir. Ancak burada asıl dikkat edilmesi gereken sınıfın ve üyelerin nitelikleridir. Dikkat ederseniz sınıfımız **DataContract** niteliği(attribute) ile, public olan alanlarımız ise **DataMember** nitelikleri ile işaretlemiş durumdadır. DataContract ve DataMember isimli niteliklerimiz **System.Runtime.Serialization**

-  .Net Remoting
-  .Net RIA Services
-  .Net Temelleri
-  Ado.Net
-  Ado.Net 2.0
-  Ado.Net Data Services
-  Ado.Net Entity Framework
-  Ado.Net Sync Services
-  Algoritma
-  Asp.Net
-  Asp.Net 2.0
-  Asp.Net 4.0
-  Asp.Net 4.0 Beta 2
-  Asp.Net 4.5
-  Asp.Net Web API
-  BCL
-  Bing
-  Biztalk
-  C#
-  C# 3.0
-  C# 4.0
-  Data Structures, Algorithms
-  Deneyimler
-  Dokuman
-  Eğitimler
-  Enterprise Library
-  Journal
-  Kitap Tavsiyelerim
-  LINQ
-  LINQ To SQL
-  NedirTv?
-  NoSQL
-  Office Development
-  Parallel Programming
-  PLINQ
-  Podcasts
-  Pre Beta
-  Seminarler
-  Silverlight 4.0
-  SOA
-  Tasarım
-  Kalıpları(Design Patterns)
-  Tasarım
-  Prensipleri(Design Principles)
-  Team Foundation Server
-  Tek Fotoluk Ipuçu
-  Teknik Dışı Konular
-  TPL
-  T-Sql
-  Velocity Project
-  Visual Studio
-  WCF
-  WCF 4.0
-  WCF 4.0 Beta 1
-  WCF 4.0 Beta 2
-  WCF 4.0 RC
-  WCF 4.5
-  WCF Data Services
-  WCF Eco System
-  WCF Öğreniyorum
-  WCF RIA Services
-  WCF WebHttp Services
-  WF

isim alanı altında yer alan sınıflardır. Böylece servisimizin döndüreceği özel bir tip için gerekli veri sözleşmesini bildirmiş bulunuyoruz.



Eğer servis metodlarımız geriye ilkel tipler döndürmüyorsa (int, double, string gibi), özellikle Employee gibi geliştirici tarafından tasarlanmış tipler söz konusu ise mutlaka bir veri sözleşmesinin (**data contract**) tanımlanması gerekmektedir. (İlerleyen makalelerimizde veri sözleşmelerine daha detaylı bir şekilde bakmaya çalışacağız.) Artık kendi veri sözleşmemizde(Data Contract) tanımladığımız, bir başka deyişle servisteki ilgili metoddan geriye nasıl bir tip döndürüleceğini bildirdiğimize göre, iş yapan metodlarımızı içerecek tip için gerekli servis sözleşmesini (**service contract**) yazarak işlemlerimize devam edebiliriz. IAdventure isimli arayüzümüzün içeriği aşağıda görüldüğü gibidir.



[ServiceContract(Name="Adventure Works Services")]

```
public interface IAdventure
{
    [OperationContract]
    Employee GetEmployee(int employeeId);

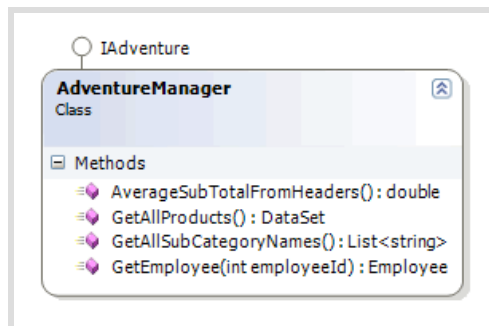
    [OperationContract]
    DataSet GetAllProducts();

















    [OperationContract]
    List<string> GetAllSubCategoryNames();

    [OperationContract]
    double AverageSubTotalFromHeaders();
}
```

IAdventure arayüzümüze(interface) ait üyelerde özellikle çeşitli tiplerden dönüş değerleri kullanmaya çalıştık. Bildiğiniz gibi, yazacağımız servisin istemciler tarafından kullanılacak bir sözleşmesi olması gerekmektedir. Herhanbiri arayüzün veya sınıfın bir servis sözleşmesi (Service Contract) sağlayacağını bildirmek için **ServiceContract** niteliği ile imzalamak gerekmektedir. Benzer şekilde servisin sunabileceği metodları bildirmek içinse **OperationContract** isimli nitelik kullanılmaktadır. ServiceContract ve OperationContract isimli nitelikler(attributes), **System.ServiceModel** isim alanı(namespace) altında yer almaktadır.

Servis sözleşmemizi tanımladıktan sonra, uzak nesne olarak asıl iş yapan metodları taşıyacak ve istemcilerin başvuruda bulunabileceği sınıfı tasarlayabiliriz. **AdventureManager** isimli sınıfımız herşeyden önce **IAdventure** arayüzünü uyarlamalıdır(Implementation).



-  WF 4.0
-  WF 4.0 Beta 1
-  WF 4.0 Beta 2
-  WF 4.0 RC
-  Windows Azure
-  Windows Forms
-  Windows Mobile
-  Windows Phone 7
-  Windows Server
-  AppFabric
-  Windows Services
-  Workflow Foundation
-  Öğreniyorum
-  WPF
-  XML
-  XML Web Services

Arşiv

2013

Kasım (5)

Ekim (5)

Eylül (3)

Ağustos (4)

Temmuz (5)

Haziran (9)

Mayıs (5)

Nisan (4)

Mart (26)

Şubat (8)

Ocak (7)

2012

Aralık (6)

Kasım (14)

Ekim (4)

Eylül (9)

Temmuz (13)

Haziran (7)

Mayıs (8)

Nisan (6)

Mart (8)

Şubat (9)

Ocak (7)

2011

Aralık (8)

Kasım (11)

Ekim (8)

Eylül (6)

Ağustos (8)

Temmuz (12)

Haziran (11)

Mayıs (2)

Nisan (10)

Mart (7)

Şubat (6)

Ocak (5)

2010

Aralık (5)

Kasım (6)

Ekim (6)

Eylül (10)

```

class AdventureManager:IAventure
{
    #region IAventure Members

    public Employee GetEmployee(int employeeId)
    {
        Employee emp=null;
        using (SqlConnection conn = new SqlConnection("data
source=localhost;database=AdventureWorks;integrated security=SSPI"))
        {
            SqlCommand cmd = new SqlCommand("Select
EmployeeId,Title,BirthDate,VacationHours From HumanResources.Employee Where
EmployeeId=@EmpId", conn);
            cmd.Parameters.AddWithValue("@EmpId", employeeId);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            if (dr.Read())
            {
                emp = new Employee();
                emp.EmployeeId = Convert.ToInt32(dr["EmployeeId"]);
                emp.Title = dr["Title"].ToString();
                emp.VacationHours = Convert.ToInt32(dr["VacationHours"]);
                emp.BirthDate = Convert.ToDateTime(dr["BirthDate"]);
            }
            dr.Close();
        }
        return emp;
    }

    public DataSet GetAllProducts()
    {
        DataSet ds=null;
        using (SqlConnection conn = new SqlConnection("data
source=localhost;database=AdventureWorks;integrated security=SSPI"))
        {
            SqlDataAdapter da = new SqlDataAdapter("Select
ProductId,Name,ListPrice,Class,SellStartDate From Production.Product", conn);
            ds = new DataSet();
            da.Fill(ds);
        }
        return ds;
    }

    public List<string> GetAllSubCategoryNames()
    {
        List<string> categoryNames = null;
        using (SqlConnection conn = new SqlConnection("data
source=localhost;database=AdventureWorks;integrated security=SSPI"))
        {
            SqlCommand cmd = new SqlCommand("Select ProductSubCategoryId,Name From
Production.ProductSubCategory Order By Name", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            categoryNames = new List<string>();
            while (dr.Read())
            {
                categoryNames.Add(dr["Name"].ToString());
            }
            dr.Close();
        }
        return categoryNames;
    }

    public double AverageSubTotalFromHeaders()
    {
        double average=0;
        using (SqlConnection conn = new SqlConnection("data
source=localhost;database=AdventureWorks;integrated security=SSPI"))
        {
            SqlCommand cmd = new SqlCommand("Select Avg(SubTotal) From
Sales.SalesOrderHeader", conn);
            conn.Open();
            average = Convert.ToDouble(cmd.ExecuteScalar());
        }
        return average;
    }

    #endregion
}

```

AdventureManager isimli sınıfımız içerisinde kullandığımız metodlar basit Ado.Net fonksiyonelliklerini ele alarak farklı tiplerde sonuçlar üretmektedir. Konumuz WCF olduğu için

Ağustos (9)
 Temmuz (10)
 Haziran (12)
 Mayıs (11)
 Nisan (14)
 Mart (20)
 Şubat (11)
 Ocak (15)

2009

Aralık (12)
 Kasım (20)
 Ekim (18)
 Eylül (8)
 Ağustos (20)
 Temmuz (15)
 Haziran (15)
 Mayıs (16)
 Nisan (18)
 Mart (2)
 Şubat (2)
 Ocak (3)

2008

Aralık (1)
 Ekim (4)
 Eylül (2)
 Ağustos (1)
 Temmuz (2)
 Haziran (3)
 Mayıs (2)
 Nisan (3)
 Mart (4)
 Şubat (4)
 Ocak (4)

2007

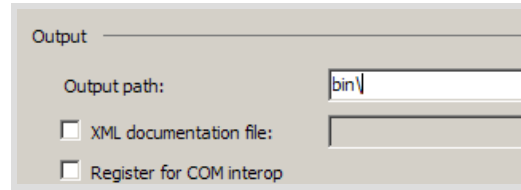
Aralık (4)
 Kasım (4)
 Ekim (4)
 Eylül (4)
 Ağustos (5)
 Temmuz (5)
 Haziran (4)
 Mayıs (5)
 Nisan (4)
 Mart (5)
 Şubat (4)
 Ocak (3)

2006

Aralık (4)
 Kasım (3)
 Ekim (4)
 Eylül (3)
 Ağustos (2)
 Temmuz (4)
 Haziran (3)
 Mayıs (3)
 Nisan (4)
 Mart (5)

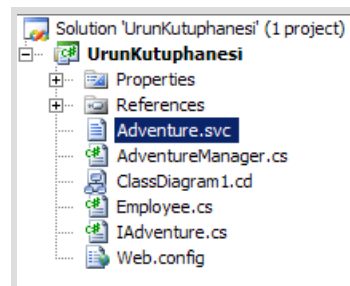
metod içerisindeki kod parçalarının işleyiş şekline çok fazla değinmeyeceğiz. Sırada servis için gerekli sanal klasörün, svc dosyasının ve web.config dosyasının oluşturulması adımları yer almaktadır.

İlk olarak IIS üzerinde sanal klasörümüzü oluşturalım. Özellikle IIS üzerinden host edilen assembly' lar göz önüne alındığında, dll' in **Bin** klasörü altında yer alması ve Bin klasörü ile aynı seviyede bir **web.config** dosyası olması gerekmektedir. Söz konusu web.config dosyası uzak nesne kullanımı için gerekli alt yapı ayarlarını barındırmaktadır. Örneğimizde IIS altına açacağımız sanal klasör(Virtual Directory) içinde aynı durumu ele almak istediğimizden proje özelliklerine girip **Build** sekmesinde yer alan **Output Path** özelliğinin değerinin **bin\debug** adresinden **bin** adresine çekilmesi yeterli olacaktır. Böylece, kodda yapacağımız değişiklikler anında canlı ortamada yansımaktadır.



Bu tarz bir işlem zorunlu değildir. Sanal Klasörün(Virtual Directory) işaret edeceği başka bir fiziki adreste belirlenebilir ancak yine **Bin** klasörünün olması, dll' in kopyasının burada yer alması ve bin ile aynı seviyede bir **web.config** dosyasının alt yapı ayarlarını içerecek şekilde olması şarttır. Örneğimizde, yapacağımız derlemelerin anında yansımalarını test edebilmek amacıyla bu tarz bir ön hazırlık yapmayı tercih ediyoruz. Artık IIS altında sanal klasörümüzü oluşturabiliriz. Aşağıdaki flash animasyonunda basit olarak IIS üzerinde yapmamız gereken adımlar gösterilmektedir. (Flash animasyonunun boyutu 187 kb olup yüklenmesi zaman alabilir.)

Makalemizde ilerlemeden önce Web servisleri ile HTTP üzerinden iletişime geçerken devreye giren asmx uzantılı dosyaları göz önüne almamızda fayda olacaktır. Sonuç itibarıyla WCF servisimizi IIS üzerinden host ederken **asmx** ile aynı amaca yönelik bir dosya var olmalıdır ki bu sayede istemciler URL üzerinden servis metadata içeriğini talep edebilsinler. İşte WCF için bu görevi yerine getiren **svc** uzantılı bir servis dosyası olacaktır. Bu dosya aynı zamanda **EndPoint** için gerekli adres tanımlanırken ele alınacaktır. (Bildığınız gibi bir WCF servisi içerisinde birden fazla EndPoint yer alabilir ve bunların ayırt edici özelliklerinden biriside adresleridir.) Bu nedenle projemize aşağıdaki içeriğe sahip olan svc uzantılı bir dosya eklememiz gerekmektedir. Bu dosya tipini öğeler arasından bulamayabilirsiniz. Bu nedenle boş bir text dosyasını svc uzantısı ile kaydederek sorunu aşabiliriz.



Şubat (4)

Ocak (3)

2005

Aralık (1)

Kasım (2)

Ekim (2)

Eylül (4)

Ağustos (2)

Temmuz (4)

Haziran (6)

Mayıs (1)

Nisan (3)

Mart (3)

Şubat (2)

Ocak (2)

2004

Aralık (3)

Kasım (3)

Ekim (5)

Eylül (16)

Ağustos (3)

Temmuz (5)

Haziran (4)

Mayıs (3)

Nisan (5)

Mart (4)

Şubat (8)

Ocak (18)

2003

Aralık (21)

Kasım (9)

1976

Aralık (1)

1900

Ocak (1)

Öneriyorum



amazon.com Privacy

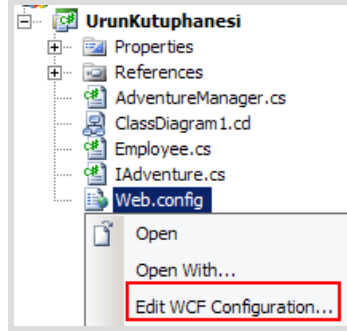
Blogroll

- Uğur Umutluoğlu
- Geeks With Blogs
- NedirTv?Com
- Ercan Bozkurt WeBlog
- Arda Çetinkaya
- Chronicles Of A Deve...
- Tamer Oz
- KodeFuGuru
- Shawn Wildermuth
- Hello World Episode...
- Building An API Cour...
- Xbox One - First Imp...
- The Activity Designe...
- Özgürce
- Aydın Ersöz

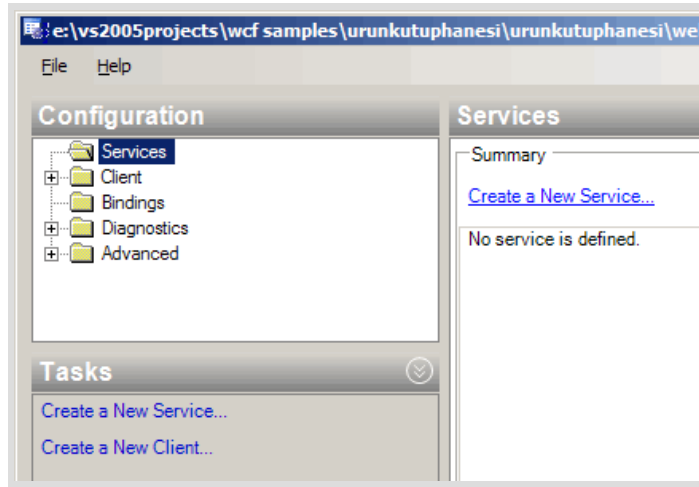
```
<%@ServiceHost Service="UrunKutuphanesi.AdventureManager"%>
<%@Assembly Name="UrunKutuphanesi"%>
```

Svc dosyamız içerisinde iki direktif yer almaktadır. **ServiceHost** direktifi içerisinde Service' e ait tip bilgisi yer alır. Bir başka deyişle uzak nesne görevini üstlenecek olan tipimizi bildiririz. Dikkat ederseniz notasyon olarak isimAlanı.tipAdı formatı kullanılmıştır. **Assembly** direktifinde ise söz konusu tipin, içerisinde yer aldığı assembly adı bildirilmektedir.

Artık web.config dosyamız için gerekli konfigürasyon tanımlamalarını yapabiliriz. (Yada görsel aracımız yardımıyla kolayca inşa edebiliriz :)) Bu amaçla projemize yeni bir **Application Configuration File** ögesi (Item) ekleyelim ve ismini **web.config** olarak değiştirelim. Konfigürasyon ayarlarımızı yapmak için web.config dosyamıza sağ tıklayıp, **Edit WCF Configuration** seçeneğini işaretlememiz yeterli olacaktır.



Edit WCF Configuration vasıtasıyla açılan yardımcı arabirim aslında, IIS üzerinde yer alan Asp.Net sekmesindeki Edit Configuration linki yardımıyla açılana benzemektedir. Ancak kullanım amacı tamamiyle WCF' a yöneliktir ve daha da önemlisi IDE dışına çıkmadan gerekli ön hazırlıkların yapılabilmesini sağlamaktadır.



Elbette buradaki ayarları yapmadan önce bir WCF servisi için gereken temel konfigürasyon ayarlarının ne olduğunu bilmek gerekmektedir. Bu konu ile ilgili bir önceki [makalemizden](#) de hatırlayacağınız gibi, WCF' nin **ABC**' si önemli bir rol oynamaktadır. Yani **AddressBindingContract** üçlemesinden söz ediyoruz. Bu kuralları konfigürasyon dosyamızda da göz önüne almamız. WCF Configuration arabirimi, service ve istemci(**Client**) tarafı için ayrı ayrı konfigürasyon dosyaları hazırlanabilmesine olanak tanımaktadır. Şu aşamadan biz servis(**Service**) tarafı için gerekli konfigürasyon ayarlarını yapacağız. Bu amaçla **Create a New Service** linkinden faydalanabilir yada manuel olarak **Configuration** sekmesini kullanabiliriz. Biz örneğimizde Create a New Service linkinden faydalanacağız.

Bu linke tıklandıktan sonra sihirbaz bizden uzak nesne görevini üstlenecek tipi seçmemizi isteyecektir. Örneğimizde söz konusu olan tip **AdventureManager** sınıfıdır. Bu seçimin arkasından sihirbaz, servis sözleşmesini(**Service Contract**) içeren tipi belirtmemizi isteyecektir. Örneğimizde bu yükü üstlenen **IAdventure** isimli arayüzdür(Interface). WCF Configuration IDE' si eğer ilk adımda seçilen tipin uyarladığı ve **ServiceContract** niteliğini taşıyan bir arayüz var ise bunu otomatik olarak bulacaktır. Bir sonraki adımda bağlantı tipi seçilir. Burada **HTTP, TCP, MSMQ, Peer To Peer** gibi seçenekler yer almaktadır. Örneğimizde HTTP tipini seçerek ilerleyeceğiz. Takip eden adımda **WSI (Web Service Interoperability)** ayarları yapılabilir veya varsayılan hali ile bırakılabilir. Biz bu adımı değiştirmeden devam edebiliriz. Son adımda ise servisimiz için bir **endpoint** adresi bildirilmesi gerekmektedir.

WCF mimarisinde istemciler(Clients), WCF Servisleri ile haberleşirken taleplerini **proxy** sınıflarına iletirler. Proxy sınıfları ise servis üzerinde yer alan endpoint' ler ile haberleşebilirler. Bu sebepten servis tarafında endpoint tanımlamalarının yapılması gerekmektedir. Bir endpoint tanımlaması

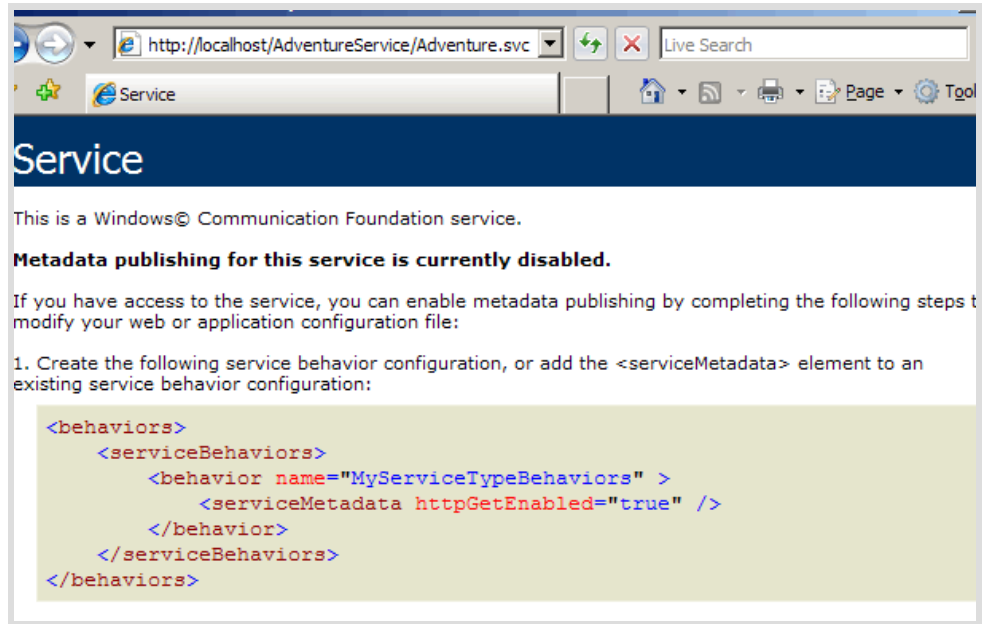
[Satış Başarısının Si...](#)[Beğendiğim iPad/iPho...](#)[Beklediğimiz An Gali...](#)[Entity Framework Fea...](#)[WCF Data Services Fe...](#)[Visual Studio User V...](#)[Asp.Net User Voice](#)[Windows Phone User
V...](#)[Enterprise Library U...](#)[Mobile Services User...](#)[LINQPad User Voice](#)[CLR Üzerine Organize...](#)[Kanal Nedirtv?Com](#)

içerisinde servisin adres bilgiside bulunur. Bu sebepten bu adımda **HTTP** protokolüne uygun bir adres girilmesi gerekir. Zaten örneğimizi IIS üzerinden sunduğumuz ve sanal klasörümüzü buna göre oluşturduğumuz için vereceğimiz adres bellidir.<http://localhost/AdventureService/Adventure.svc> .Dikkat ederseniz adresimizde daha önceden oluşturduğumuz Adventure.svc isimli dosyamızda yer almaktadır. Böylece şimdilik gerekli WCF ayarlarını tamamlamış bulunuyoruz. Aşağıdaki Flash animasyonda gerekli ayarların nasıl yapıldığını izleyebilirsiniz. *(Flash dosyasının boyutu 229 kb olduğu için yüklenmesi zaman alabilir)*

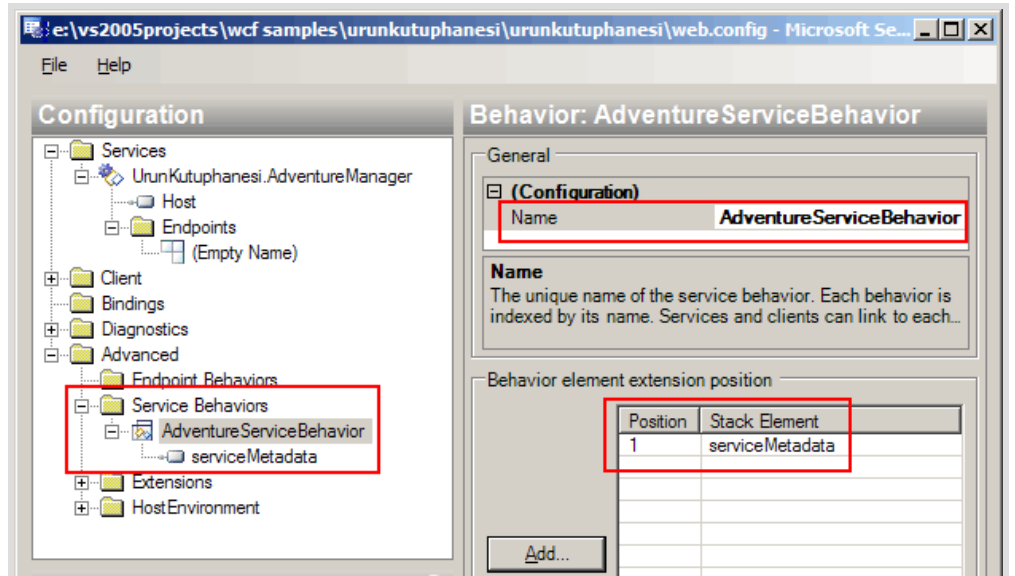
Bu işlemlerin ardından web.config dosyamızın içeriğinde aşağıdaki gibi olacaktır.

```
<?xml version="1.0" encoding="utf-8" ?>
  <configuration>
    <system.serviceModel>
      <services>
        <service name="UrunKutuphanesi.AdventureManager">
          <endpoint address="http://localhost/AdventureService/Adventure.svc"
            binding="basicHttpBinding" bindingConfiguration="" contract="UrunKutuphanesi.IAdventure" />
        </service>
      </services>
    </system.serviceModel>
  </configuration>
```

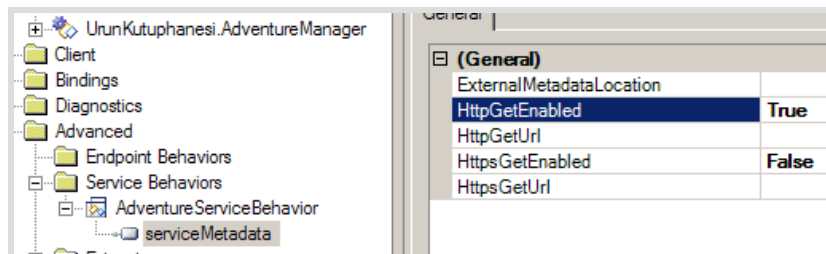
Artık servisimizin çalışıp çalışmadığını tarayıcı penceremiz üzerinden test edebiliriz. Tek yapmamız gereken <http://localhost/AdventureService/Adventure.svc> adresini talep etmek olacaktır. Bu talep sonucunda eğer herşey yolunda ise aşağıdakine benzer bir ekran görüntüsü ile karşılaşınız.



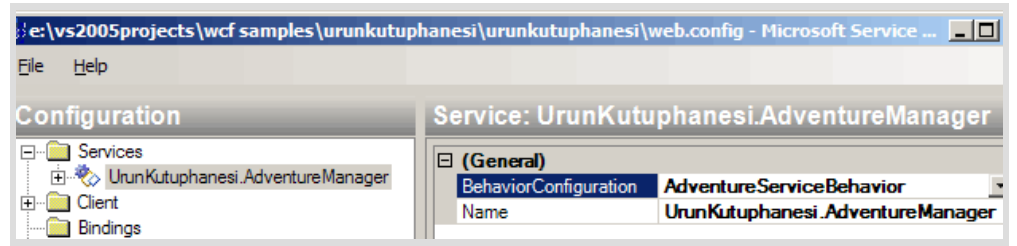
Yine dikkat edilecek olursa servisin metadata yayınlanmasının (**Meta Data Publishing**) aktif olmadığını görürüz. Bildiğiniz gibi metadata bilgisi istemcilerin proxy sınıflarını oluşturabilmeleri için gereklidir. Bunu sağlamak maksadıyla yeni bir servis davranışı (**Service Behavior**) oluşturulmalı ve servise bildirilmelidir. Bu amaçla ilk olarak **Advanced** sekmesinde yer alan **Service Behaviors** kısmından yeni bir davranış nesnesi eklenmeli ve aşağıdaki ekran görüntüsünde yer alan ayarlar yapılmalıdır. Bu ayarlara göre eklenen servis davranışına bir isim verilmiş (Name özelliği ile) ve **serviceMetadata** elementi dahil edilmiştir.



ServiceMetadata elementinin içeriğinin aşağıdaki gibi düzenlenmesi gerekmektedir. Burada **HttpGetEnabled** özelliğinin değerini **true** olarak belirlemeliyiz ki HTTP üzerinden servis metadata içeriğini çekebilelim.



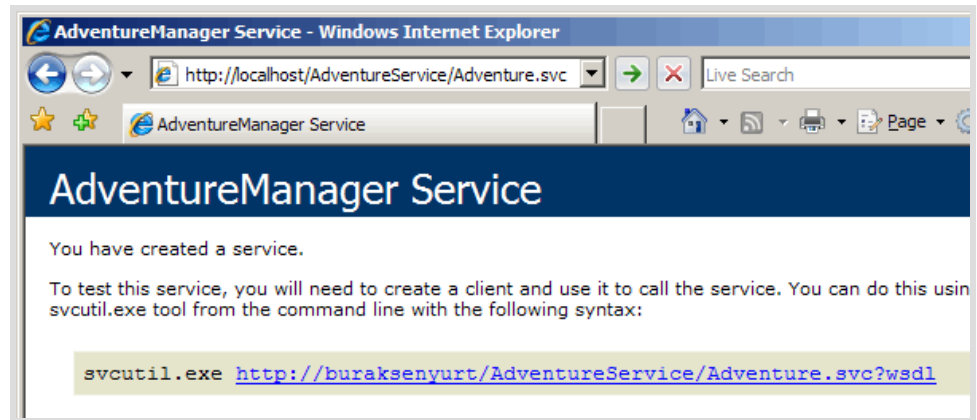
Sonrasında ise servismizin belirtilen davranışı kullanacağını bildirmemiz gerekmektedir. Bunun içinde servisin, **BehaviorConfiguration** elementine aşağıdaki ekran görüntüsünde olduğu gibi az önce oluşturulan Service Behavior nesnesinin adını vermeliyiz.



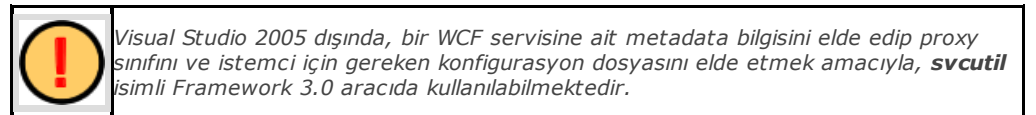
Bu işlemlerin ardından servisimize ait web.config dosyasının son halide aşağıdaki gibi olacaktır.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="AdventureServiceBehavior">
          <serviceMetadata httpGetEnabled="true" />
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service behaviorConfiguration="AdventureServiceBehavior"
name="UrunKutuphanesi.AdventureManager">
        <endpoint address="http://localhost/AdventureService/Adventure.svc"
binding="basicHttpBinding" bindingConfiguration="" contract="UrunKutuphanesi.IAdventure" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

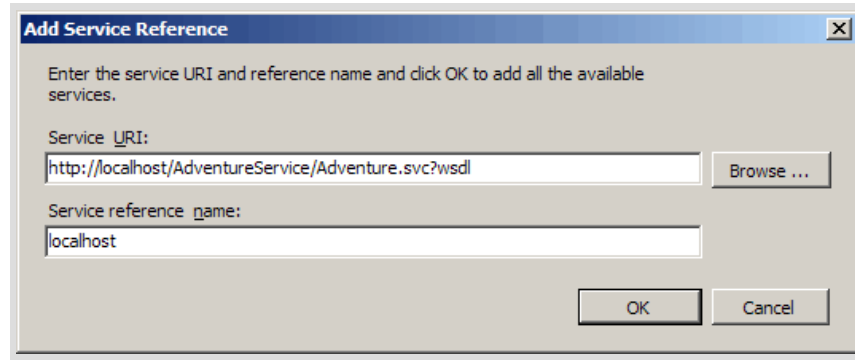
Gördüğünüz gibi servisimiz için berilediğimiz davranış ve servis ile olan ilişkisi otomatik olarak konfigürasyon dosyasına element ve nitelikler yardımıyla yansıtılmıştır. Servisimizi tarayıcı penceresi üzerinden yeniden talep edersek aşağıdaki ekran görüntüsü ile karşılaşırız. Dikkat edersek, servise ait WSDL dökümanını elde edebilmemizi sağlayacak bir link bilgiside yer almaktadır.



Bu kadar işlemten sonra artık istemcimizi yazmaya başlayabiliriz. İstemci tarafında herhangi bir uygulama söz konusu olabilir. Biz mimariyi basit bir şekilde ele almaya çalıştığımızdan bir Console uygulaması geliştireceğiz. Yine Visual Studio 2005' in **Add Service Reference** seçeneğini kullanarak ilgili servisi otomatik olarak uygulamamıza ekleyebilir ve gerekli proxy sınıfının kolay bir şekilde oluşturulmasını sağlayabiliriz.



Console uygulamamızı oluşturduktan sonra projeye sağ tıklayıp **Add Service Reference** seçeneğini kullanarak WCF servisimizi projemize eklememiz gerekmektedir. Burada içeride kullanmak istediğimiz isim alanının adında belirtebiliriz. Varsayılan olarak aynı web servislerinde olduğu gibi localhost verilmektedir.



Bu işlemin ardından istemci için gereken proxy sınıfı ve konfigürasyon dosyası otomatik olarak oluşturulacaktır. Artık aşağıdaki kod satırlarını kullanarak sanki sıradan bir web servisini kullanıyormuş gibi WCF Servisimizi ele alabiliriz.

```
using System;
using System.Data;
using Istemci.localhost;

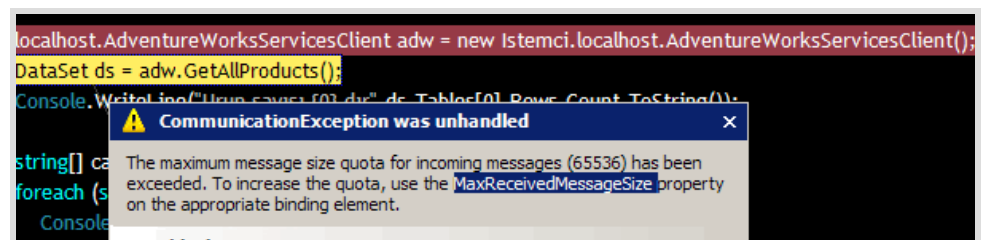
namespace Istemci
{
    class Program
    {
        static void Main(string[] args)
        {
            localhost.AdventureWorksServicesClient adw = new
Istemci.localhost.AdventureWorksServicesClient();
            DataSet ds = adw.GetAllProducts();
            Console.WriteLine("Urun sayısı {0} dir",ds.Tables[0].Rows.Count.ToString());

            string[] categoryNamees=adw.GetAllSubCategoryNames();
            foreach (string category in categoryNamees)
                Console.WriteLine(category);

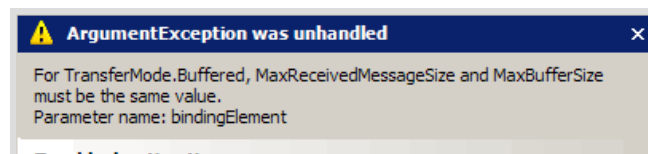
            Employee emp=adw.GetEmployee(1);
            Console.WriteLine(emp.Title + " " + emp.BirthDate.ToShortDateString() + " " +
            emp.VacationHours.ToString());

            Console.WriteLine("Ortalama Sub Total
            {0}",adw.AverageSubTotalFromHeaders().ToString("C2"));
        }
    }
}
```

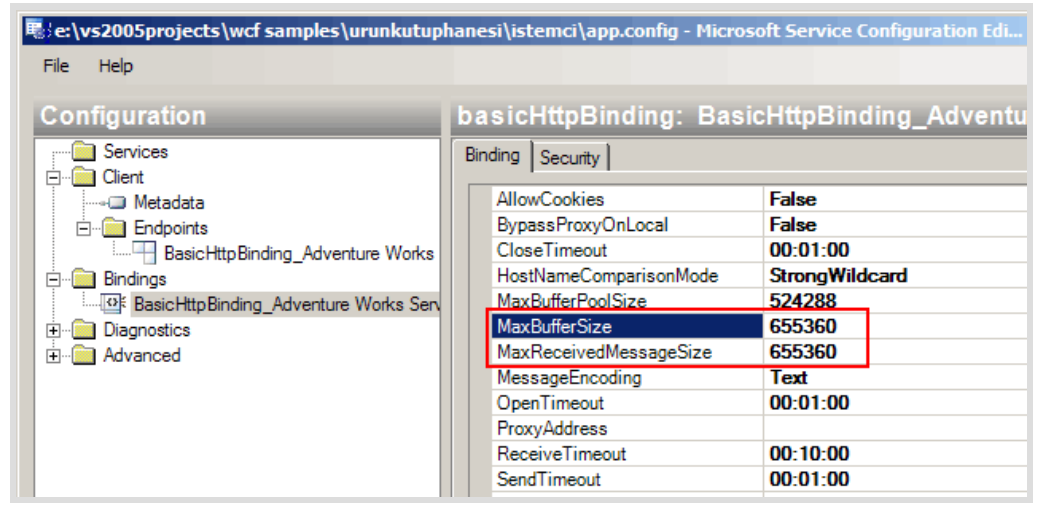
Uygulamayı bu haliyle çalıştırdığımızda GetAllProducts metodu için aşağıdaki gibi çalışma zamanı istisnası (run-time exception) alınız.



Bunun sebebi istemci tarafında oluşturulan konfigürasyon dosyasında belirtilen **MaxReceivedMessageSize** boyutunun düşük olmasıdır. Nitekim dönen DataSet içeriğinin yer aldığı paketin boyutu çok daha fazladır. Bu nedenle istemci uygulamaya ait App.config dosyasını Edit WCF Configuration ile açıp aşağıdaki ekran görüntüsünde olduğu gibi **MaxReceivedMessageSize** ve **MaxBufferSize** özelliklerinin aynı değere set edilmesi gerekir. Örneğimizde bu değerlerin sonuna bir 0 eklenmiştir. Burada dikkat edilmesi gereken noktalardan birisi de, her iki değerinde aynı olması zorunluluğudur. Aksi takdire çalışma zamanında yine bir istisna alınız.

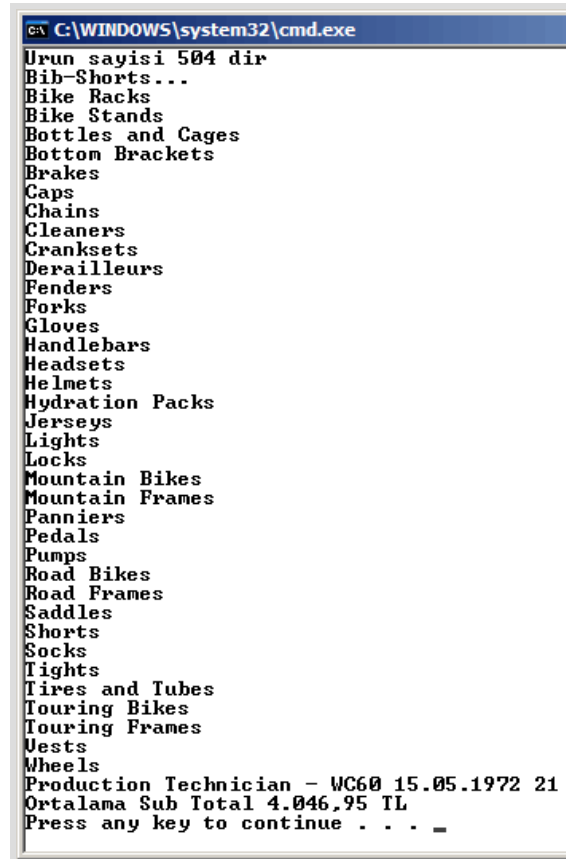


Bu sebepten ayarlarımız örneğin aşağıdaki gibi olmalıdır.



Örneğimizdeki kodlarımızı incelersek eğer, ilginç olan bir nokta olduğunu farkederiz.

GetAllSubCategoryNames metodu servis tarafında geriye generic bir List koleksiyonu döndürüyorkan, istemci tarafında metodun dönüşü **string[]** tipinden bir dizi olmuştur. Buda aslında farklı tipteki istemciler için ortak bir standart koyulmaya çalışıldığı şeklinde yorumlanabilir. Uygulamayı çalıştırdığımızda aşağıdakine benzer bir ekran görüntüsü elde ederiz.



Bu makalemizde Windows Communication Foundation çatısı altında, servislerimizi IIS üzerinden nasıl yayınlayıp kullanabileceğimizi incelemeye çalıştık. Bunu yaparken bir servis sözleşmesini(**Service Contract**) nasıl tanımlayabileceğimizi, bu sözleşmeyi uygulayan asıl uzak nesne sınıfının geriye kullanıcı tanımlı bir tip döndürmesi halinde veri sözleşmesini(**data contract**) nasıl geliştirmemiz gerektiğini gördük. Bunlara ek olarak özellikle **IIS Hosting** için **svc** uzantılı dosyaların rolünü inceledik ve konfigürasyon dosyalarını daha kolay oluşturabilmemizi sağlayan **Edit WCF Configuration** seçeneğini ele aldık. Böylece geldik bir makalemizin daha sonuna. Bir sonraki makalemizde görüşmek dileğiyle hepinize mutlu günler dilerim.

[Örnek Uygulama İçin Tıklayınız.](#)

Burak Selim ŞENYURT
selim@bsenyurt.com

Currently rated 4.7 by 14 people

Tags: [Wcf](#), [Windows Communication Foundation](#), [Iis](#), [Internet Information Services](#)Categories: [WCF](#)Actions: [E-mail](#) | [del.icio.us](#) | [Permalink](#) | [Yorumlar \(0\)](#) | [Comment RSS](#) 

İlişkili yazılar

[WCF – Visual Studio 2008 ile Gelen Yenilikler](#)

Değerli Okurlarım Merhabalar, Yazılım dünyası çeşitli ürün gruplarını ve b...

[Dayanıklı WCF Servisleri \(Durable WCF Services\)](#)

Değerli Okurlarım Merhabalar, Uzun zamandır Windows Communication Foundation(WCF) konulu bir ara...

[İlk Bakışta WCF](#)

Değerli Okurlarım Merhabalar, Bildiğiniz gibi bir süre önce Microsoft .Net Framework' ...

Yorumlar kapalı.