.Net

Posted by Tuğrul HELVACI - Haziran 21, 2009 Comments 15

Win32 & .Net(Delphi->C#)

Aslında herşey Java adı verilen programlama dilinin doğuşuna kadar ilerliyor. Java, programlama dünyasına farklı bir perspektif katmıştı. Yazılan kodların applet'ler vasıtası ile web ortamlarında kullanılabilmesi yada Java Runtime ile değişik işletim sistemi platformlarında çalıştırılabiliyor olması onu günden güne daha popüler hâle getiriyordu.

Microsoft, kendi işletim sistemlerinin yeryüzündeki tüm bilgisayarlarda kullanılamayacağının farkına vardığında; Java'ya karşı bir önlem almak gerektiğini düşündü ve bu sayede veriye her ortamdan erişebilecek bir sistem planlamaya başladı. Ancak elbette Java'nın da hâla beceremediği gibi Microsoft'da platform bağımsızlığı hususunda başarılı olamadı. Zaten tasarımların gereği de bunu pek mümkün kılmıyordu. Java'da üretilen kodların bytecode'lara çevrilmesi ve JVM(Java Virtual Machine) adı verilen programlarla üzerinde çalıştığı platforma adapte edilmesindeki süreç, .Net'de de kendisine farklı isimlerle yer buldu.

.Net, ürettiği MSIL kodunu üzerinde çalıştığı platformun anlayacağı makina dili koduna ise JIT vasıtası ile çevirir. Bu tıpkı Java'nın JVM'ine benzer. Aradaki benzerlikleri saymaya kalksak emin olun sayfalarca yazı yazmamız gerekir. Benim bu makalede amacım bu iki platform bağımsız olduğunu iddia eden teknolojinin benzerliklerini ve farklılıklarını anlatmak değil. Bu

Etiketler

Absolute Abstract Classes ActionScript Algoritma API Bug Byte Arrays Class Reference CloseHandle COM CreateDesktop CreateEvent CreateMutex CreateProcess CreateRemoteThread CreateSemaphore CreateToolHelp32SnapShot CreateWaitableTimer DeleteCriticalSection delphi 2010 DTS EnterCriticalSection

bilgileri genel kültür amacı ile sunduktan sonra Delphi'nin bu teknolojiler ile iletişimi hakkında bilgi vermek ve **gerçek platform bağımsız kodlamanın gelecekte Delphi ile olabileceğinin** umudunu sizlerle paylaşmak.

Görüldüğü üzere gerek .Net gerekse de Java platform bağımsız native kod geliştiremiyorlar. Ürettikleri ara kodların, çeşitli işletim sistemlerinde yorumlanıp makina koduna çevrilmeleri gerekiyor. Dolayısı ile teoride bu teknolojileri kullanan uygulamaların, native uygulamalardan hızlı olması beklenmiyor.

.Net, bilindiği üzere programlama dillerinden bağımsız bir platform. Framework adı verilen kod kütüphanelerinin tüm programlama dilleri tarafından ortak bir şekilde kullanılabilmesi, CTS denilen ortak tip sınıflarının .Net içinde olması, herhangi bir .Net destekli dilin bir diğer .Net destekli dil ile iletişimini son derece sorunsuz ve kolay hale getirmekte. Büyük çaplı proje ekiplerinin pek çok programcıya sahip olduğu gerçeği göz önüne alındığında, bu ekip üyelerinin herhangi bir .Net dilini bilmesi ve bu dil ile geliştirme yapmasının projeye olumsuz bir etkisinin olmaması elbette son derece güzel bir durum.

.Net Framework adı verilen kaba tabiri ile programlama kütüphanesi, şu aşamada 3.5 versiyonunda bulunuyor. Yakın bir zamanda 4.0 versiyonunun çıkması bekleniyor. Bildiğim kadarı ile Framework'teki sınıf tasarımları .Net framework 2.0'dan sonra herhangi bir değişikliğe uğramadı. Sadece üzerine yeni yetenekler eklendi. Kısaca .Net terminolojisinden bahsettikten sonra, Delphi'nin bu terminolojiye hangi mesafede olduğu hakkında yazmak isterim.

Delphi 7 , hâla pek çok Delphi kullanıcısının kullandığı ana IDE durumunda. Bunun stabilite ve hız gibi pek çok etkenleri var. Delphi, gerçek anlamda .Net ile 8 versiyonunda buluştu. Ancak Delphi 8, Microsoft'un .Net teknolojisine ilk desteği veren ürün olduğu için hiçte stabil değildi ve Delphi severler tarafından pek de tutulmadı. Ardından Borland, .Net platformuna desteğini Delphi 2005 ile sürdürdü. Ancak kişisel tecrübelerim ile söyleyebilirim ki, o da gerektiği ölçüde stabil değildi ve pek çok sorunu vardı. .Net desteği, 2006 ve 2007 sürümlerine kadar devam etti ve ardından büyük bir değişim ile karşılaştık.

Borland, programlama ürünlerini CodeGear isimli firmaya devretmiş ve ardından bu ürünleri Embarcadero isimli bir firma satın almıştı. Bu süreçte, artık Delphi ve diğer programlama ortamlarının .Net'e destek vermemesine karar verildi. Bu son derece yerinde bir karardı. Delphi, hayata ilk geldiği 1995 yılından bu zamana kadar native uygulama geliştirme konusunda, dünyada pek çok kez ödül almış bir ortam iken, .Net framework desteği adına adeta kendi kendine intihar etme kararı almıştı.

Günümüzde, .Net programlama adına Delphi yazım tarzına aşina olan programcıların Delphi Prism ürününü tercih etmesi isteniyor. Bu ürün; .Net framework ortamına %100 destek veren ve Delphi yazım tarzına son derece benzeyen Embarcadero firması programlama ortamları ailesinin bir başka ürünü.

Embarcadero, Delphi ürününü satın aldıktan sonra, beklentilerimizin aksine bu ürün ailesine son derece önem verdi ve ürünün eskide olduğu gibi popülerliğini yeniden kazanmasında büyük katkı sağlamaya başladı. Bu konuda sevgili arkadaşımız Sadettin POLAT'ın sitesindeki makaleyi okumanızı öneriyorum.

Son Yorumlar

+ Yeni Veri Tipleri ve Operator Overloading için Tuğrul HELVACI
+ Yeni Veri Tipleri ve Operator Overloading için Zafer Çelenk
+ Delphi ve Google Maps API için Tuğrul HELVACI
+ Delphi ve Google Maps API için ahmet
+ Ziyaretçi Defteri için Tuğrul HELVACI

Konumuzun Win32 ve .Net olması münasebeti ile, kısa bir zaman sonra çıkması beklenen yeni Delphi sürümünün yani Delphi Weaver'ın çok önem verdiğim bir özelliğini de sizlerle paylaşmak istiyorum. Delphi Weaver'ın özelliklerinde listelenen ancak pek çok kişinin dikkatini cezbetmeyen, "Seamless .NET <> Native communication" özelliği eminim pek çok kişinin native uygulama geliştirebilecekleri ortamlara geçişinde etken olacaktır. Peki nedir bu "Seamless .NET <> Native communication" özelliği ?

Bu özellik hakkında henüz net bir şey söz konusu değil. Ancak, neler yapılabileceği hususunda ; burayı ve bu haber grubunu takip etmenizi tavsiye edebilirim.

Kısaca; Win32 ortamında tamamen native kod geliştirirken, .Net framework fonksiyonalitesine herhangi bir COM bağımlılığı olmadan, %100 delphi kodları ile erişmek isterseniz bu teknoloji tam size göre demektir. 

Atozed Software'in üzerinde çalıştığı CrossTalk isimli ürün, Delphi 5,6,7..2009 altından .Net Framework'e erişebilmeniz için gereken tüm altyapıyı sağlayacak. Projenin lideri olan Chad Z. Hover ile yapmış olduğum görüşme neticesinde öğrendiğim şey ise beni daha da sevindirdi. CrosTalk ürünü Delphi Weawer ile birlikte gelecek. Yani birkaç paragraf önce belirttiğimiz "Seamless .NET <> Native communication" Delphi Weaver altındaki CrossTalk ürününü simgeliyor.

Bu yeni teknolojinin Delphi'nin yükselmesinde büyük bir paydaya hizmet edeceği inancını taşıyorum. Ayrıca Delphi'nin hedefleri arasında Cross platform code compilation olduğunu da bilmenizi istiyorum. İşte gerçek platform bağımsız kodlama bizlerin hizmetine sunulacak. Delphi'de yazdığımız uygulamaları Windows, Linux yada MacOS gibi işletim sistemleri üzerinde çalışır halde göreceğiz. Elbette bu, .Net'in yada Java'nın yaptığı gibi değil, native derleme ile yapılacak.

Tüm bu anlatılanlar, belki de hâla afaki kalmış olabilir. CrossTalk'ın yahut Delphi Weaver içindeki "Seamless .NET <> Native communication" özelliğinin ne kadar önemli olduğu anlaşılmamış da olabilir. Bunun önemini anlatabilmek için, Win32'den .Net'e erişmeye çalışmak gerekir.

Bu makalede, Win32 ortamından .Net Framework'e erişeceğiz ve tüm bu zorlukları sizlerin gözleri önüne sereceğim. Öncelikle Visual Studio 2005 ile bir Class Library(DLL) oluşturacağız.Bu DLL'imiz SQL Server 2005'e erişim sağlayan bir kaç sınıftan ibaret olacak. Ve Bu DLL'imizi Delphi 7 altından kullanacağız.

O halde kodlamaya başlayabiliriz. Öncelikle Visual Studio'muzu açıyoruz ve yeni bir Class Library projesi oluşturuyoruz. Ardından Microsoft.SqlServer.ConnectionInfo, Microsoft.SqlServer.Smo, Microsoft.SqlServer.SmoEnum, Microsoft.SqlServer.SqlEnum library'lerini referans olarak ekliyoruz. Projemiz aşağıda göründüğü gibi olacaktır:

Son Yazılar

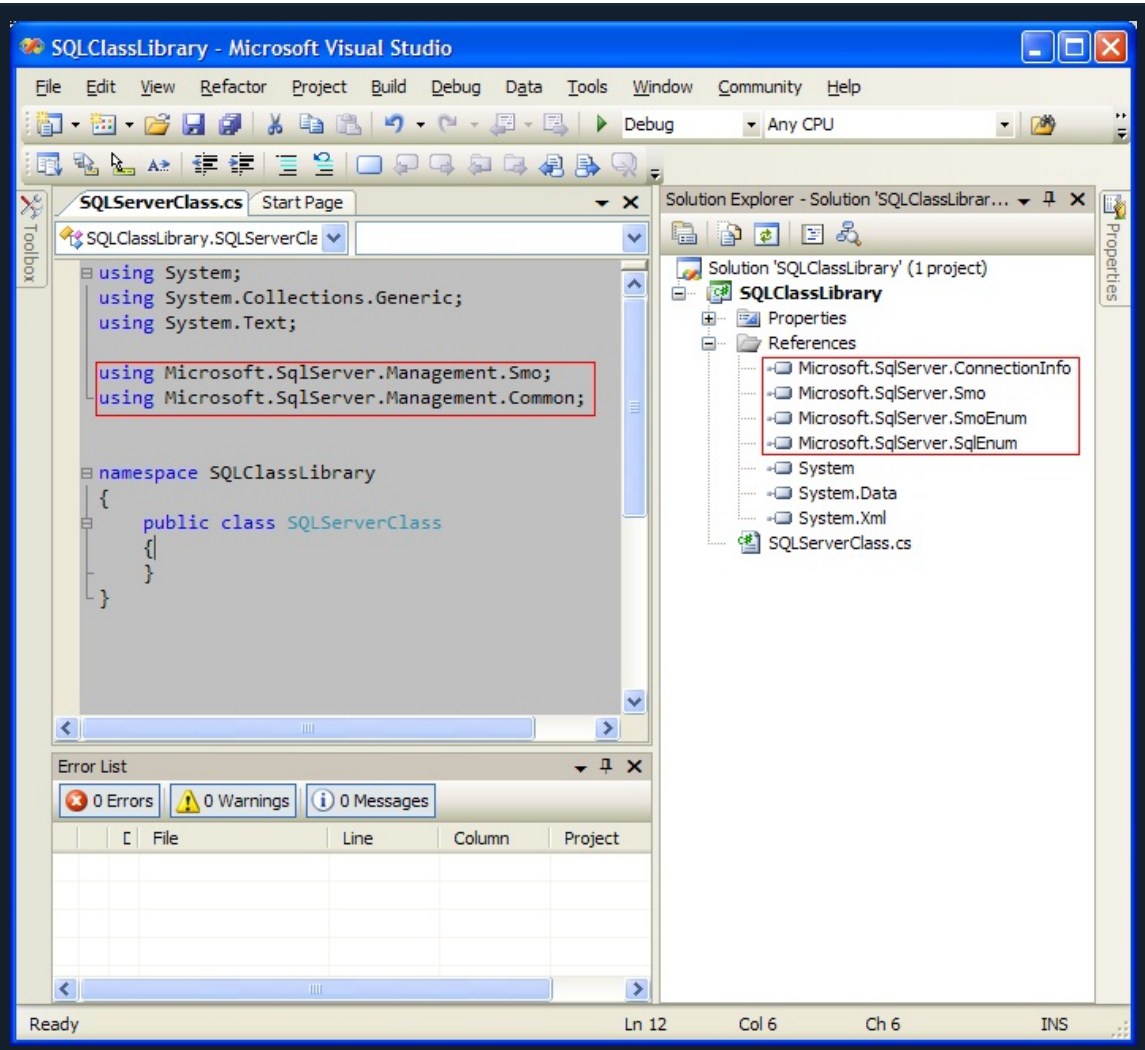
+ Yeni Veri Tipleri ve Operator Overloading
+ Interface Nedir, Nerelerde ve Neden Kullanırız ?
+ Derinlemesine Threading..(3)
+ SQL'de Benzerlik Algoritmaları…
+ Full Text Searching…

Kategoriler

+ Genel (4)
  + IDE (1)
+ İşletim Sistemi (7)
  Programlama (43)
  + .Net (3)
  + C# (2)
+ + Delphi (43)
  + Grafik (2)
  + Internet (4)
  + Veritabanı (3)
  + Win32 (6)

Takvim

Temmuz 2010



Bu makalenin can alıcı noktası, .Net ortamında yazdığımız kodlarımızın dışarıdan(Win32 ortamından) kullanılabilmesi için COM programlamadan istifade edeceğimiz gerçeğidir. COM programlama kullanacağımıza göre, dışarıdan erişime açacağımız tüm sınıfların ComVisible attribute'u ile görünür hâle getirilmesi gerekir. Bir diğer önemli nokta ise, Class Library projemizin özelliklerinde gizlidir. Bu özel durumları aşağıdaki görsellerden izleyebilirsiniz:

Projemizin Application bölümünden Assembly Information buttonuna tıklanarak yapılalacak ayarlar:

Temmuz 2010

Pts	Sal	Çar	Per	Cum	Cts	Paz
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

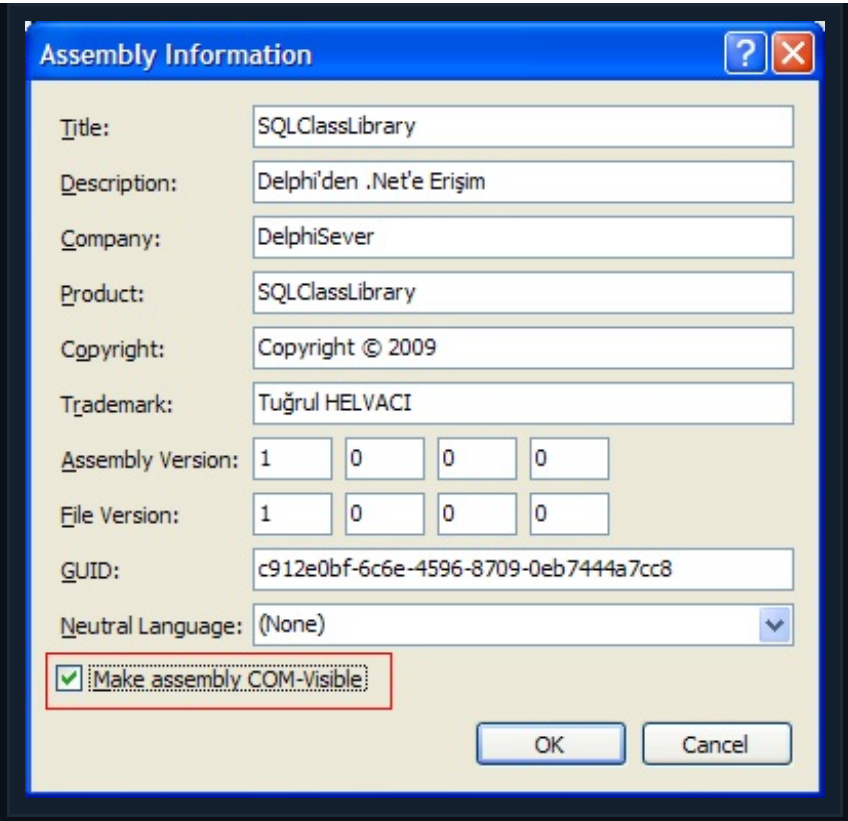
« Haz

Arşivler

+ Haziran 2010 (1)
+ Mayıs 2010 (1)
+ Nisan 2010 (3)
+ Ağustos 2009 (1)
+ Temmuz 2009 (2)
+ Haziran 2009 (7)
+ Mayıs 2009 (32)

Bağlantılar

+ Bir Türkçe Sevdalısı..
+ Delphi Türkiye Forum
+ Gürcan ÖZTÜRK
+ M.Fatih KÜÇÜKKELEPÇE
+ Memik YANIK Kişisel
+ Memik YANIK'ın Günlüğü
+ Nick Hodges
+ Sinan BARAN



Projemizin Build sayfasında yapılacak ayarlar:

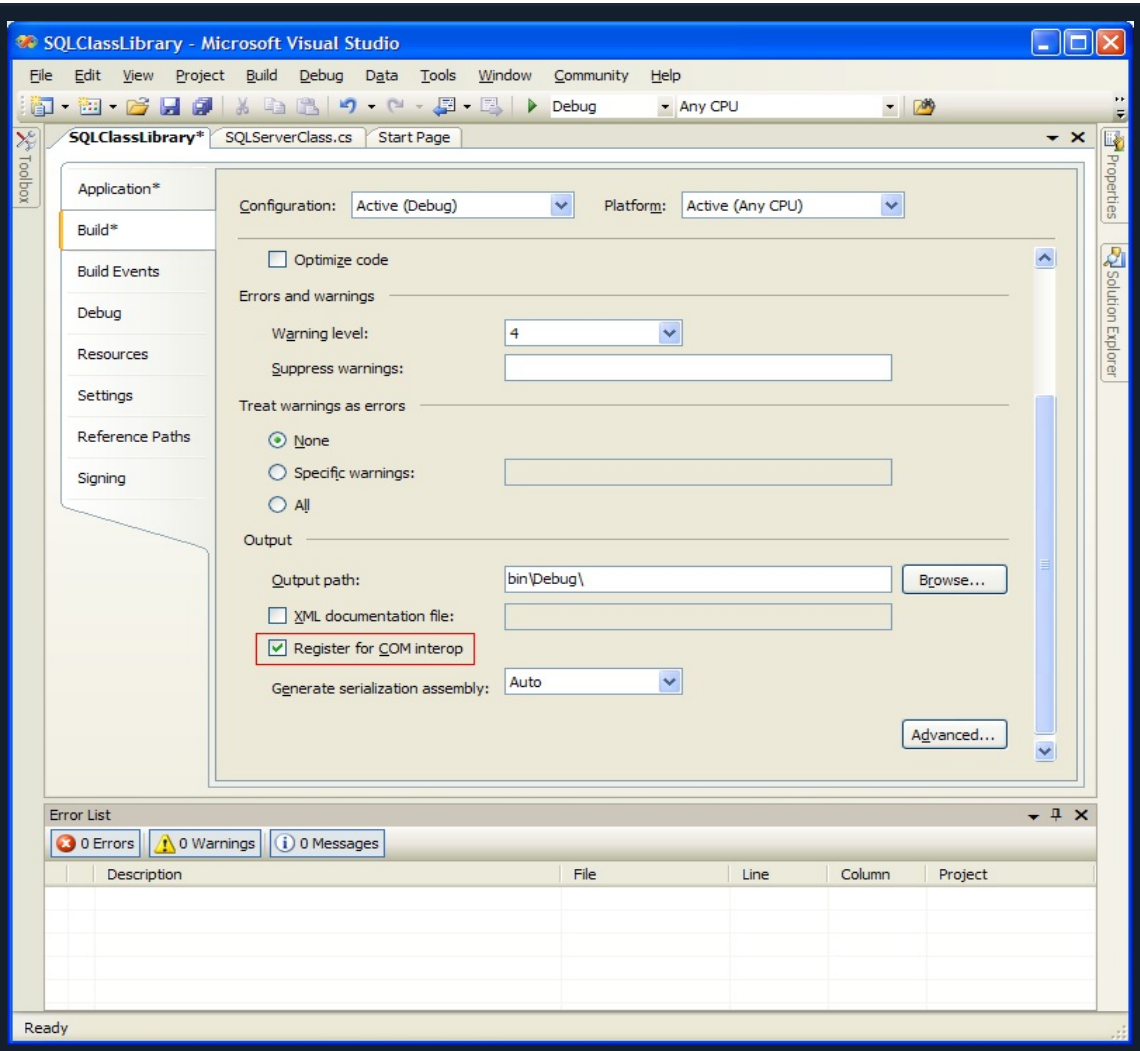
+ Zafer Çelenk

Ziyaretçi Bilgileri

Ziyaret: 6 / 12073

Beğenilenler

+ Delphi 2010 (Weaver) ve TValue - 14 votes
+ Derinlemesine Threading..(3) - 13 votes
+ Derinlemesine Threading..(2) - 6 votes
+ Derinlemesine Threading..(1) - 6 votes
+ Yeni Veri Tipleri ve Operator Overloading - 4 votes
+ Win32 & .Net(Delphi->C#) - 4 votes
+ Delphi ve Google Maps API - 3 votes
+ TThread.WaitFor Bug.. - 3 votes



Bu ayarların yapılmasına müteakip, C# tarafında SQL Server 2005'e bağlanan, ve bir veritabanının altındaki tüm tablolara erişim sağlayabileceğimiz kodları yazmaya başlamadan ewel anlatmamız gereken bir kaç husus daha var. Bunlardan en önemlisi; .Net framework platformundaki her veri türünü COM ortamında kullanmanın zorluklarıdır. Kullanmak istediğimiz veri türleri, ki bunların içinde özel sınıflar, framework sınıfları gibi tüm veri türleri ComVisible attribute'u ile işaretlenmiş olmalıdır. .Net ortamındaki tüm türlerin nasıl yönetileceğini dış uygulamalar bilemezler. Binlerce tür olduğu hesaplanırsa, bu türlerin hepsinin ComVisible ile işaretlenmesinin ne kadar zor olduğu da aşikârdır.

Makalemizin başlarında, temel veri türlerinin .Net içinde tanımlı olduğunu ifade etmiştik. Bu CTS adını almıştı ve pek çok programlama dilinin bir arada çalışabilmesi için son derece

+ BITS(Background Intelligent Transfer Service) ile sessiz sedasız download ;) - 3 votes
+ Bir Kiosk ve CreateDesktop macerası.. - 3 votes

Meta

+ Giriş
+ Yazılar RSS
+ Yorumlar RSS
+ WordPress.org

Etiketler-Liste

Absolute API Byte Arrays Class Reference COM CreateDesktop CreateProcess CreateRemoteThread CreateToolHelp32SnapShot DTS EnumDesktopProc EnumDesktops Function Pointer GetCurrentProcess GetIconInfo GetKeyState GetLastInputInfo GetProcessMemoryInfo GetStartupInfo Hacking Inheritance Interface JavaScript Method Pointer Module32First Module32Next OpenDesktop Persistance Pointer Procedure Pointer Process Process32First Process32Next Reqursion RTTI SetProcessWorkingSetSize SQL Server SwitchDesktop TerminateProcess Thread Untyped Parameters VirtualAllocEx VirtualFreeEx WaitForSingleObject WriteProcessMemory

faydalı bir unsurdu. Ancak, şimdi bizim için bir sorun gibi duruyor. CTS içinde tanımlı olan bazı basit değişken türlerinin COM programlama da ComVisible ile görünür hâle getirilmeden de kullanılabildiğini ifade etmek sanırım sizleri biraz rahatlatacaktır ancak yine de yeteri derece de rahatlamamış olmanız gerekir. Çünkü, pek çok projemizde bu projemizde olduğu gibi, Integer, String vb. gibi basit veri türleri bizim için yeterli değildir.

.Net framework'ün Smo kütüphanesi ile Sql Server'a erişebildiğini biliyoruz. Ve bu Smo kütüphanesi içinde Sql Server içindeki hemen hemen her nesne için tanımlanmış bir veri tipi bulunmaktadır.(Genellikle bir sınıf). Peki biz bu veritiplerini dışarıya nasıl sunacağız ? Smo namespace'ininin tamamını mı ComVisible ile işaretleyeceğiz ?

Elbette hayır. Bu son derece uğraştırıcı ve zor bir durum olurdu. Bu sebeple, Smo kütüphanesi içinde ilgilendiğimiz sınıflara karar verecek ve bu sınıflar için bir arabirim yazacağız. Ardından bu arabirimi görünür kılacağız 

Örneğin, biz bu uygulamamızda Sql Server 2005'e erişip herhangi bir veritabanının içindeki tüm **tablo**'lara ulaşmak istediğimize göre, Smo namespace'indeki Table sınıfına ait sahte bir interface düzenleyeceğiz. Gelin biraz da kodlayalım ki daha anlaşılır olsun:

```
01. using System;
02. using System.Collections.Generic;
03. using System.Text;
04.
05. using Microsoft.SqlServer.Management.Smo;
06. using Microsoft.SqlServer.Management.Common;
07. using System.Runtime.InteropServices;
08.
09. namespace SQLClassLibrary
10. {
11.     [ComVisible(true)]
12.     public interface ITableInterface
13.     {
14.         DateTime CreateDate { get; }
15.         string Name { get; }
16.         long RowCount { get; }
17.         int ColumnCount { get; }
18.         int TriggerCount { get; }
19.     }
20.
21.     public class TableClass : ITableInterface
22.     {
23.         private Table ActiveTable;
24.
25.         public DateTime CreateDate
26.         {
27.             get
28.             {
29.                 return ActiveTable.CreateDate;
```

Delphi About

+ Auto Select All The Text For TCustomEdit On Mouse Click 29 Haziran 2010
+ Memory Leak Notification in Delphi - Report Memory Leak on Program Exit 27 Haziran 2010
+ PCRE Workbench - Regular Expression Test Tool - Source Code Delphi Application 22 Haziran 2010
+ Deleting Dataset Records In a Loop - Poll Results - Why All Records Are Not Deleted 21 Haziran 2010
+ Force TListView's Edit Mode using a Keyboard Shortcut 17 Haziran 2010
+ Implementing On Item Click / Double Click for Delphi's TListView control 16 Haziran 2010
+ Run Your Delphi Application in Full Screen - Implement "F11 - Full Screen" 14 Haziran 2010
+ Displaying Enumerated Properties in a Selectable List - Run-Time Enum Selection in Delphi 10 Haziran 2010
+ How Do You Delete Dataset Records In a Loop? 08 Haziran 2010
+ Display Custom Hints for Status Bar Panels 07 Haziran 2010

Sık Ziyaret Edilenler

+ Sayfa: Home (16060)
+ Delphi ve Google Maps API (4132)
+ Delphi ve Google Maps API (1976)

```
            }
        }
        public string Name
        {
            get
            {
                return ActiveTable.Name;
            }
        }
        public long RowCount
        {
            get
            {
                return ActiveTable.RowCount;
            }
        }
        public int ColumnCount
        {
            get
            {
                return ActiveTable.Columns.Count;
            }
        }
        public int TriggerCount
        {
            get
            {
                return ActiveTable.Triggers.Count;
            }
        }

        public TableClass(Table tbl)
        {
            ActiveTable = tbl;
        }

        ~TableClass()
        {
            // Nesneleri yok etmeyi unutma.! GC ye güvenme..
        }
    }

    public class SQLServerClass
    {
    }
}
```

Yukarıdaki kod örneğimizde öncelikle System.Runtime.InteropServices isimli alanı using bloğuna eklediğimizi görüyorsunuz. ComVisible attribute'unu kullanmak için bunu yapmak

+ Nedir bu Thread'lerden çektiğimiz..!
(1444)
+ Delphi & Animated Flash Charts(Fusion Charts)
(1179)
+ Delphi & JavaScript Kardeşliği
(936)
+ Delphi 2010 (Weaver) ve TValue
(918)
+ Derinlemesine Threading..(1)
(889)
+ Derinlemesine Threading..(2)
(698)
+ Win32 & .Net(Delphi->C#)
(620)
+ Kategori: Delphi
(613)
+ Bir Kiosk ve CreateDesktop macerası..
(568)
+ Sayfa: Hakkında
(563)
+ TThread.WaitFor Bug..
(540)
+ Delphi 2010 (Weaver) ve TValue
(530)

Oylama..

Sitedeki makaleleri yararlı buluyor musunuz ?

+ ◯ **Evet, yararlı ama yetersiz.**
+ ◯ **Evet, son derece yararlı.**
+ ◯ **Evet, mükemmel.**
+ ◯ **Hayır, yararlı değil.**

durumundayız. Ardından Smo namespace'i içinde tanımlı olan Table isimli sınıfın ilgilendiğimiz birkaç tane property'sini tanımladığımız ITableInterface arabiriminin içinde görüyorsunuz. Bu arabirim, ComVisible ile işaretli durumda. Biz ileride Delphi üzerinden yazdığımız C# kodlarını kullanırken arabirim referanslarına erişeceğiz 

Ardından basit bir sınıf olan TableClass sınıfının ITableInterface arabirimini implemente ettiğini görüyorsunuz. Sizinde farkettiğiniz gibi, bu sınıf ComVisible ile işaretlenmemiş. Çünkü biz bu sınıfa ilgili arabirimi üzerinden erişim sağlayacağız. Dolayısı ile sınıfımız ITableInterface'i implemente ettiği için sadece arabirimi ComVisible ile görünür kılmak bizim için yeterli olacaktır.

Buradaki anafikri anlamanız çok önemli. .Net framework altındaki kompleks yapıları COM ortamından kullanabilmek için onların COM görünürlüğünün sağlanması gereklidir. Biz bu uygulamamızda .Net framework'ün Smo kütüphanesi içindeki Table isimli sınıfı bu sebeple dışarıya veremedik ! İlgili sınıfı(Table) ComVisible ile görünür kılamadığımız için, biz de bu sınıfın sahte bir kopyasına bir arabirim oluşturduk ve o kopyayı görünür kıldık. Meselenin özü aslında sadece bu kadar. Şimdi Sql Server 2005'e erişim sağlamasını düşündüğümüz sınıfımıza birer adet Connect ve Disconnect metodu yazmamız işimizi görecek midir ? Evet server'a bağlanmak için server'ın adına ve bağlantı şifresine ihtiyaç duyacağız ancak peki ya hangi veritabanından ilgili tabloları alacağız ?

Bu noktada bizim için bir de Smo namespace'indeki Database sınıfına bir interface uydurmamız gerekecek. Bu interface'de aşağıdaki gibi olacak:

```
001. [ComVisible(true)]
002. public interface IDatabaseInterface
003. {
004.     int ActiveConnections { get; }
005.     DateTime CreateDate { get; }
006.     DateTime LastBackupDate { get; }
007.     string Name { get; }
008.     double Size { get; }
009.
010.     int StoredProcedureCount { get; }
011.     int TableCount { get; }
012.     int UserDefinedFunctionCount { get; }
013.     int ViewCount { get; }
014.
015.     ITableInterface TableFromName(string TableName);
016.     ITableInterface TableFromIndex(int Index);
017. }
018.
019. public class DatabaseClass : IDatabaseInterface
020. {
021.     private Database ActiveDatabase;
022.     private Hashtable hTables;
023.
```

```
024.     public int ActiveConnections
025.     {
026.         get
027.         {
028.             return ActiveDatabase.ActiveConnections;
029.         }
030.     }
031.     public string Collation
032.     {
033.         get
034.         {
035.             return ActiveDatabase.Collation;
036.         }
037.         set
038.         {
039.             ActiveDatabase.Collation = value;
040.             ActiveDatabase.Alter();
041.         }
042.     }
043.     public DateTime CreateDate
044.     {
045.         get
046.         {
047.             return ActiveDatabase.CreateDate;
048.         }
049.     }
050.     public DateTime LastBackupDate
051.     {
052.         get
053.         {
054.             return ActiveDatabase.LastBackupDate;
055.         }
056.     }
057.     public string Name
058.     {
059.         get
060.         {
061.             return ActiveDatabase.Name;
062.         }
063.     }
064.     public double Size
065.     {
066.         get
067.         {
068.             return ActiveDatabase.Size;
069.         }
070.     }
071.     public int StoredProcedureCount
072.     {
073.         get
```

```
074.         {
075.             return ActiveDatabase.StoredProcedures.Count;
076.         }
077.     }
078.     public int TableCount
079.     {
080.         get
081.         {
082.             return ActiveDatabase.Tables.Count;
083.         }
084.     }
085.     public int UserDefinedFunctionCount
086.     {
087.         get
088.         {
089.             return ActiveDatabase.UserDefinedFunctions.Count;
090.         }
091.     }
092.     public int ViewCount
093.     {
094.         get
095.         {
096.             return ActiveDatabase.Views.Count;
097.         }
098.     }
099.
100.     public ITableInterface TableFromName(string TableName)
101.     {
102.         return (ITableInterface)hTables[TableName];
103.     }
104.
105.     public ITableInterface TableFromIndex(int Index)
106.     {
107.         ITableInterface Result = null;
108.
109.         int iCounter = 0;
110.         IDictionaryEnumerator TableEnum =
            hTables.GetEnumerator();
111.         while (TableEnum.MoveNext())
112.         {
113.             if (iCounter == Index)
114.             {
115.                 Result = (TableClass)TableEnum.Value;
116.                 break;
117.             }
118.
119.             iCounter += 1;
120.         }
121.
122.         return Result;
```

```
123.     }
124.
125.     public DatabaseClass(Database db)
126.     {
127.         ActiveDatabase = db;
128.         hTables = new Hashtable();
129.
130.         foreach (Table tbl in ActiveDatabase.Tables)
131.         {
132.             if (!tbl.IsSystemObject) hTables.Add(tbl.Name,
                 new TableClass(tbl));
133.         }
134.     }
135. }
```

Yukarıda IDatabaseInterface tanımını ve bu interface'i implemente eden sınıfın tanımını görüyorsunuz. Biz ilgili veritabanının tablolarına TableFromName ve TableFromIndex metodları ile erişeceğiz. Bu durumda artık son sınıfımız olan en başta tanımını boş olarak gördüğünüz SQLServerClass sınıfını yazmaya başlayabiliriz. Ancak bu sınıfımız içinde bir interface tasarlamamız ve ComVisible ile görünür kılmamız gerekiyor. Tanım aşağıdaki gibi olacak;

```
01. [ComVisible(true)]
02. public interface ISQLLibrary
03. {
04.     bool Connect(string ServerName, string DatabaseName,
         string UserName, string Password);
05.     void Disconnect();
06.
07.     IDatabaseInterface Database { get; }
08. }
09.
10. public class SQLServerClass : ISQLLibrary
11. {
12.     private Server srv;
13.     private IDatabaseInterface CurrentDatabase;
14.
15.     public IDatabaseInterface Database
16.     {
17.         get
18.         {
19.             return CurrentDatabase;
20.         }
21.     }
22.
23.     public bool Connect(
24.                         string ServerName,
25.                         string DatabaseName,
26.                         string UserName,
27.                         string Password
```

```
                        )
    {
        bool Result = false;

        srv = new Server(ServerName);
        srv.ConnectionContext.LoginSecure = false;
        srv.ConnectionContext.Login = UserName;
        srv.ConnectionContext.Password = Password;
        Result = true;

        if (Result)
        {
            Database db = srv.Databases[DatabaseName];
            CurrentDatabase = new DatabaseClass(db);
        }

        return Result;
    }

    public void Disconnect()
    {
        srv.ConnectionContext.Disconnect();
    }

    ~SQLServerClass()
    {
        Disconnect();
    }
}
```

Görüldüğü gibi son derece basit bir yapıya sahip olan SQLServerClass sınıfımız ISQLLibrary interface'ini implemente etmiş ve ISQLLibrary ComVisible ile görünür hâle getirilmiş. Kodumuzun nihai hali aşağıdaki gibi olacaktır:

```
using System;
using System.Collections.Generic;
using System.Text;

using Microsoft.SqlServer.Management.Smo;
using Microsoft.SqlServer.Management.Common;
using System.Runtime.InteropServices;
using System.Collections;

namespace SQLClassLibrary
{
    [ComVisible(true)]
    public interface ITableInterface
    {
        DateTime CreateDate { get; }
        string Name { get; }
```

```
017.         long RowCount { get; }
018.         int ColumnCount { get; }
019.         int TriggerCount { get; }
020.     }
021.
022.     public class TableClass : ITableInterface
023.     {
024.         private Table ActiveTable;
025.
026.         public DateTime CreateDate
027.         {
028.             get
029.             {
030.                 return ActiveTable.CreateDate;
031.             }
032.         }
033.         public string Name
034.         {
035.             get
036.             {
037.                 return ActiveTable.Name;
038.             }
039.         }
040.         public long RowCount
041.         {
042.             get
043.             {
044.                 return ActiveTable.RowCount;
045.             }
046.         }
047.         public int ColumnCount
048.         {
049.             get
050.             {
051.                 return ActiveTable.Columns.Count;
052.             }
053.         }
054.         public int TriggerCount
055.         {
056.             get
057.             {
058.                 return ActiveTable.Triggers.Count;
059.             }
060.         }
061.
062.         public TableClass(Table tbl)
063.         {
064.             ActiveTable = tbl;
065.         }
066.
```

```
067.         ~TableClass()
068.         {
069.             // Nesneleri yok etmeyi unutma.! GC ye güvenme..
070.         }
071.     }
072.
073.     [ComVisible(true)]
074.     public interface IDatabaseInterface
075.     {
076.         int ActiveConnections { get; }
077.         DateTime CreateDate { get; }
078.         DateTime LastBackupDate { get; }
079.         string Name { get; }
080.         double Size { get; }
081.
082.         int StoredProcedureCount { get; }
083.         int TableCount { get; }
084.         int UserDefinedFunctionCount { get; }
085.         int ViewCount { get; }
086.
087.         ITableInterface TableFromName(string TableName);
088.         ITableInterface TableFromIndex(int Index);
089.     }
090.
091.     public class DatabaseClass : IDatabaseInterface
092.     {
093.         private Database ActiveDatabase;
094.         private Hashtable hTables;
095.
096.         public int ActiveConnections
097.         {
098.             get
099.             {
100.                 return ActiveDatabase.ActiveConnections;
101.             }
102.         }
103.         public string Collation
104.         {
105.             get
106.             {
107.                 return ActiveDatabase.Collation;
108.             }
109.             set
110.             {
111.                 ActiveDatabase.Collation = value;
112.                 ActiveDatabase.Alter();
113.             }
114.         }
115.         public DateTime CreateDate
116.         {
```

```
117.             get
118.             {
119.                 return ActiveDatabase.CreateDate;
120.             }
121.         }
122.         public DateTime LastBackupDate
123.         {
124.             get
125.             {
126.                 return ActiveDatabase.LastBackupDate;
127.             }
128.         }
129.         public string Name
130.         {
131.             get
132.             {
133.                 return ActiveDatabase.Name;
134.             }
135.         }
136.         public double Size
137.         {
138.             get
139.             {
140.                 return ActiveDatabase.Size;
141.             }
142.         }
143.         public int StoredProcedureCount
144.         {
145.             get
146.             {
147.                 return ActiveDatabase.StoredProcedures.Count;
148.             }
149.         }
150.         public int TableCount
151.         {
152.             get
153.             {
154.                 return ActiveDatabase.Tables.Count;
155.             }
156.         }
157.         public int UserDefinedFunctionCount
158.         {
159.             get
160.             {
161.                 return
                    ActiveDatabase.UserDefinedFunctions.Count;
162.             }
163.         }
164.         public int ViewCount
165.         {
```

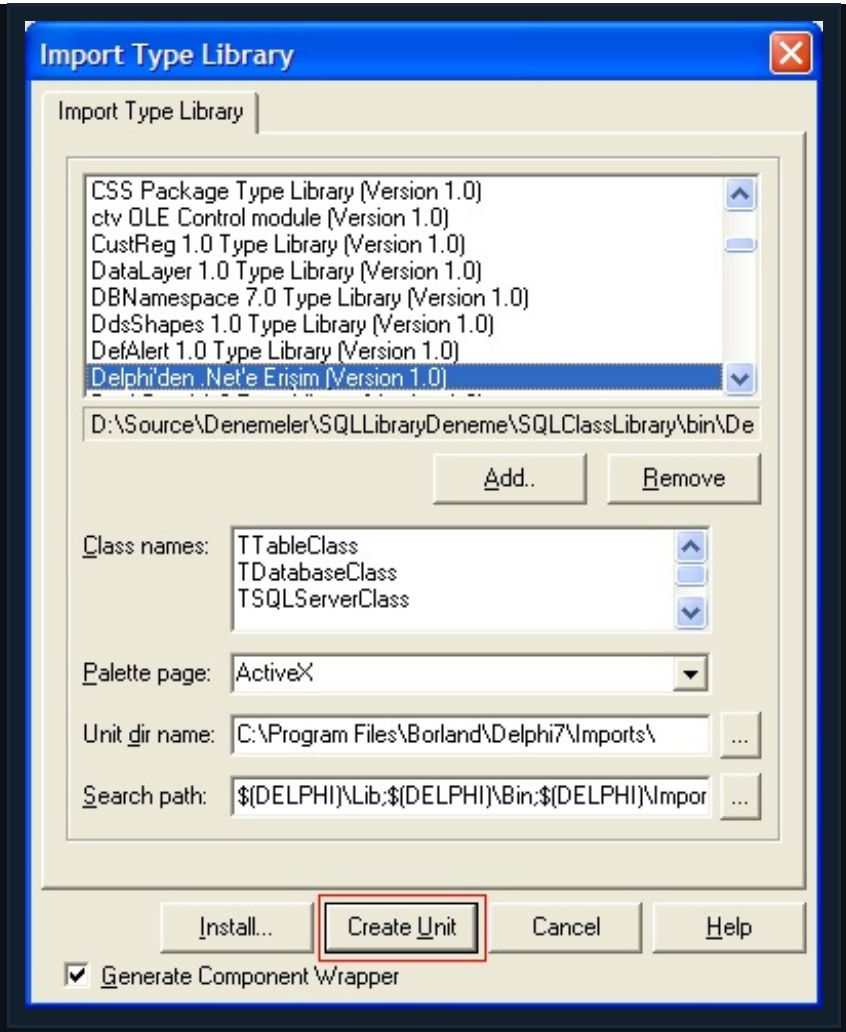
```
166.             get
167.             {
168.                 return ActiveDatabase.Views.Count;
169.             }
170.         }
171.
172.         public ITableInterface TableFromName(string
             TableName)
173.         {
174.             return (ITableInterface)hTables[TableName];
175.         }
176.
177.         public ITableInterface TableFromIndex(int Index)
178.         {
179.             ITableInterface Result = null;
180.
181.             int iCounter = 0;
182.             IDictionaryEnumerator TableEnum =
                 hTables.GetEnumerator();
183.             while (TableEnum.MoveNext())
184.             {
185.                 if (iCounter == Index)
186.                 {
187.                     Result = (TableClass)TableEnum.Value;
188.                     break;
189.                 }
190.
191.                 iCounter += 1;
192.             }
193.
194.             return Result;
195.         }
196.
197.         public DatabaseClass(Database db)
198.         {
199.             ActiveDatabase = db;
200.             hTables = new Hashtable();
201.
202.             foreach (Table tbl in ActiveDatabase.Tables)
203.             {
204.                 if (!tbl.IsSystemObject)
                     hTables.Add(tbl.Name, new TableClass(tbl));
205.             }
206.         }
207.     }
208.
209.     [ComVisible(true)]
210.     public interface ISQLLibrary
211.     {
212.         bool Connect(string ServerName, string DatabaseName,
```

```
        string UserName, string Password);
213.    void Disconnect();
214.
215.    IDatabaseInterface Database { get; }
216. }
217.
218. public class SQLServerClass : ISQLLibrary
219. {
220.     private Server srv;
221.     private IDatabaseInterface CurrentDatabase;
222.
223.     public IDatabaseInterface Database
224.     {
225.         get
226.         {
227.             return CurrentDatabase;
228.         }
229.     }
230.
231.     public bool Connect(
232.                         string ServerName,
233.                         string DatabaseName,
234.                         string UserName,
235.                         string Password
236.                        )
237.     {
238.         bool Result = false;
239.
240.         srv = new Server(ServerName);
241.         srv.ConnectionContext.LoginSecure = false;
242.         srv.ConnectionContext.Login = UserName;
243.         srv.ConnectionContext.Password = Password;
244.         Result = true;
245.
246.         if (Result)
247.         {
248.             Database db = srv.Databases[DatabaseName];
249.             CurrentDatabase = new DatabaseClass(db);
250.         }
251.
252.         return Result;
253.     }
254.
255.     public void Disconnect()
256.     {
257.         srv.ConnectionContext.Disconnect();
258.     }
259.
260.     ~SQLServerClass()
261.     {
```

```
            Disconnect();
        }

    }
}
```

Şimdi Visual Studio ortamında Ctrl+Shift+B tuşlarına basarak yada Build menüsünden Build seçeneklerinden birisini seçerek derleme işlemi yaptığımızda Delphi tarafına geçmeye hazırız demektir. İlgili DLL'imiz artık projemizi kaydettiğimiz yerde oluşturulduğu gibi, proje ayarlarından Com görünürlüğünü işaretlediğimiz için bu DLL aynı zamanda COM sistemine Register edilmiş durumdadır. Eğer elle register işlemi yapmak istiyorsanız o halde, regasm kullanmak durumunda kalacaksınız. Ancak şu anda bu işi Visual Studio bizim için yapmış durumda. Delphi'mize geçip yeni bir proje açalım, ve Project/Import Type Library adımından aşağıdaki görselde görebileceğiniz gibi ilgili DLL'imizi listeden bulalım ve Create Unit'e basalım.



SQLClassLibrary_TLB.pas isimli dosyamız artık oluşmuş durumda. Delphi örneğimiz ise aşağıdaki gibi olacaktır;

```
01. uses SQLClassLibrary_TLB;
02. ..
03. ..
04. ..
05. procedure TForm1.Button1Click(Sender: TObject);
06. var
07.   SqlServer : ISQLLibrary;
08. begin
09.   SqlServer := CoSQLServerClass.Create as ISqlLibrary;
10.   SqlServer.Connect('TUGRUL', 'BENIMDB', 'sa', '*****');
```

```
11.   ShowMessage(
12.     Format('Kayıt Sayısı=%d, Kolon Sayısı=%d',
13.       [
14.         SqlServer.Database.TableFromName('BenimTablom'
            ).RowCount,
15.         SqlServer.Database.TableFromName('BenimTablom'
            ).ColumnCount
16.       ]) );
17.   SqlServer.Disconnect;
18.   SqlServer := nil;
19. end;
```

Özetlemek gerekir ise, Delphi Win32 projelerinde herhangi bir .Net framework kütüphanesine

erişebilir ve onu kullanabilirsiniz. Ancak ComVisible görünürlük durumuna dikkat ederek Sanırım şimdi makalemin başlarında belirttiğim, CrossTalk ve Delphi Weaver içinde .Net erişimi özelliklerinin ne derece önemli olduğu daha açıktır.

Saygılar, sevgiler..

[Translate]

15 Comments




Ferruh Koroglu
on Haziran 21st, 2009

Çok güzel bir makale Tuğrul Hocam,Eline sağlık.

Tuğrul HELVACI
on Haziran 21st, 2009

Teşekkür ederim, beğenmenize sevindim.

Olcay DAĞLI
on Haziran 23rd, 2009

Hocam yine güzel bir makale daha, eline sağlık…;)

Tuğrul HELVACI
on Haziran 23rd, 2009

Teşekkür ederim Olcay, okuyan gözlerine sağlık 

Nihal Alıcı
on Ağustos 7th, 2009

Ben bu makaleni yeni okudum Tuğrul, gerçekten çok faydalı, güzel bir yazı olmuş. Hani tam da "Delphi ile hiç alakam kalmıyor artık" dediğim zamanlarda, yine aklıma bir kurt düşürdün yani. Artık CodeGear bloglarını bile okumaz olmuştum, tekrar başlasam mı acaba  ?
Eline, emeğine sağlık

Tuğrul HELVACI
on Ağustos 7th, 2009

Teşekkür ederim nihal, beğenmene sevindim. Delphi biz onu bilerek ve kasten öldürmeye teşebbüs etmediğimiz sürece ölecek gibi görünmüyor  Bende yaşasın diye mümkün mertebe elimden geleni nacizane yapma gayretindeyim. Sen oku oku Delphi bloglarını yine 

Nurullah
on Nisan 13th, 2010

Hocam eline sağlık çok güzel bir makale yanlız benim bir sorunum var 

SQLServerClass sınıfının tek constructor ı var olsun oda

```
Public SQLServerClass(string cnnString)
{
```

}
gibi olsun benim elimde bunun gibi bir örnek var ve ben constructor ın parametresini gönderemediğim için Class not registered hatası alıyorum. Bununla ilgili bir çözümün var mı?

Tuğrul HELVACI
on Nisan 13th, 2010

Şu an deneme imkanım yok ama benim yaptığım gibi parametresiz bir constructor kullanıp, onun yerine ilgili parametreleri Connect gibi bir metoda geçmeniz mümkün değil mi ?

Nurullah ERCAN
on Nisan 13th, 2010

Valla DLL i yazan bizim müşterimiz. Axapta entegrasyonu için yazılmış bir dll aslında söylediğinizin benzerini talep ettim. Ama müşteri derki değiştiremyiz  Açıkcası en net kaynak bu dün akşamdan bu yana ara ara sonuç yok. Çıldırmak üzereyim  Yardımcı olabilirsen çok sevinirim.

Tuğrul HELVACI
on Nisan 13th, 2010

Mail adresimi ziyaretçi defterinden bulabilirsiniz. Oraya konu ile ilgili mümkün olan en detaylı açıklamayı içeren bir mail atabilirseniz eve gittiğimde elimden geldiğince yardımcı olabilirim.

Nurullah ERCAN
on Nisan 13th, 2010

Teşekkürler hocam.

Barış Kırgıç

Hocam Merhaba ,

Barış Kirgiz
on Mayıs 6th, 2010

Öncelikle harika bir kaynak sağlamışsınız teşekkür ederim.

Yukarıda dediğiniz gibi herşeyi yaptım . Kendi yaptığım dll de hiç sorun yok .

Aynı işi müşterinden istedim ve oda yaptı ve bana "TLB" file gönderdi . Register ettim fakat
"Class not registered" hatası alıyorum .

Sizce neden olabilir.

Tuğrul HELVACI
on Mayıs 6th, 2010

Merhaba Barış bey, acaba müşteriniz sınıf tasarımında ilgili interface'leri COMVisible ile işaretlememiş olabilir mi ?
Müşterinizin yazmış olduğu class library'i bana mail ile ulaştırabilirseniz daha net bir yorum yapabilirim. Normalde sizinde bileceğiniz üzere .Net class library'leri regasm yardımcı aracı ile ilgili sisteme kayıt ettirilirler ardından başka dillerde COM programlamanın kullanımına açık hale gelirler. regasm ile sisteme kayıt ettirilmemiş dll'ler için sınıf kaydedilmemiş(Class not registered) gibi bir hata alabilirsiniz. Müşterinizden ilgili dll'i isteyip regasm ile bu dll'i sisteme kayıt edip, ardından Delphi'den import type library ile import etmeyi denediniz mi ?

Tüm bunları denediğiniz halde hâla aynı hata ile karşılaşıyorsanız, o zaman başta belirttiğim gibi ilgili dll'i yada mümkünse dll'in c# kaynak kodlarını yollayabilirseniz fikir yürütmekte daha başarılı olabilirim.

Taner İNEKÇİ
on Mayıs 31st, 2010

Hocam öncelikle verdiğiniz değerli bilgiler için çok teşekkürler…

Ben yukarıda verdiğiniz tüm adımları visual basic ile uyguladım dll oluşturdum register ettim delphi ye import u gerçekleştirdim. Projeme oluşturduğum unit i ekledim ancak ki visual basicta yarattığım class taki proseduru mu gorebılıyorum ancakki projeyı derleyıp
calıstırdıgımda access violation hatası alıyorum sizce sebebi ne olabilir.??

yardımcı olabilirseniz çok sevinirim şimdiden teşekkürler

Tuğrul HELVACI
on Mayıs 31st, 2010

Access Violation çok genel bir hatadır. Sıklıkla henüz oluşturulmamış(hafızada yer ayırılmamış) nesnelere erişimlerde, yada Free edilmiş ama hâla hafızada anlamsız bir yeri gösteren işaretçiler kullanıldığında karşılaşırsınız. Soruna bu açıdan bir bakmanızı önerebilirim.

Share your comment

Name (required)

Mail (required)

Website

Submit

Son Yorumlar

- Yorumunuza teşekkürler. Aslında operatör aşırı yüklemenin sınıflarda pek bir avantaj... by Tuğrul HELVACI
- Merhaba, Ben ilk olarak merakımdan dolayı uğraştığım C++ dilinde görmüştüm Operatör... by Zafer Çelenk
- Merhaba, makaleyi yazdığım zaman kodlarda bir sorun yoktu. Belki Google Maps'de bir şeyler... by Tuğrul HELVACI
- Merhaba Kodlarda bir sorun mu var? yoksa ggogle bu hizmeti durdurdu mu? Sadece boş bir... by ahmet
- Programcı arkadaşların daha fazla takıldığı bir yerde yazsa idiniz mesajınızı daha... by Tuğrul HELVACI
- Tuğrul Bey Yeri Burasımıdır Bilmiyorum O nedenle hata ediyorsam özür dilerm bir iş ilanı... by geyikben
- Teşekkürler ;) by Tuğrul HELVACI

Yeni Eklenenler

- Yeni Veri Tipleri ve Operator Overloading
- Interface Nedir, Nerelerde ve Neden Kullanırız ?
- Derinlemesine Threading..(3)
- SQL'de Benzerlik Algoritmaları…
- Full Text Searching…
- Delphi 2010 (Weaver) ve TValue
- Derinlemesine Threading..(2)

Linkler

- Bir Türkçe Sevdalısı.. - Taha EKREM
- Delphi Türkiye Forum
- Gürcan ÖZTÜRK - Gürcan ÖZTÜRK
- M.Fatih KÜÇÜKKELEPÇE
- Memik YANIK Kişisel
- Memik YANIK'ın Günlüğü
- Nick Hodges
- Sinan BARAN