

افزودن یک DataType جدید برای نگهداری تاریخ خورشیدی - 1

حامد قنادی

۲۳:۳۰ ۱۳۹۲/۰۲/۰۹

www.dotnettips.info

SQL Server, SQL Server 2012, C#.NET, Persian, CLR

عنوان:

نویسنده:

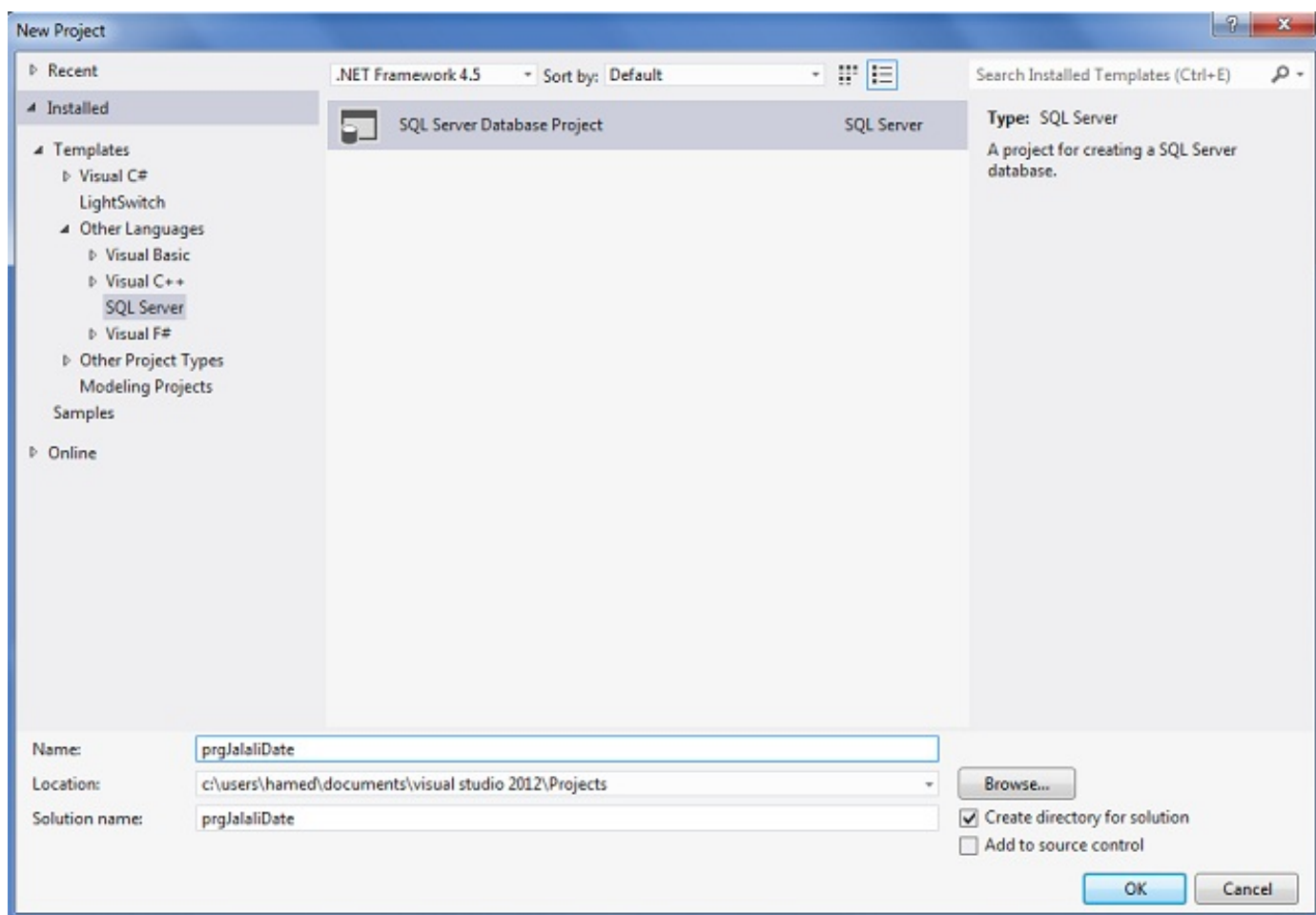
تاریخ:

آدرس:

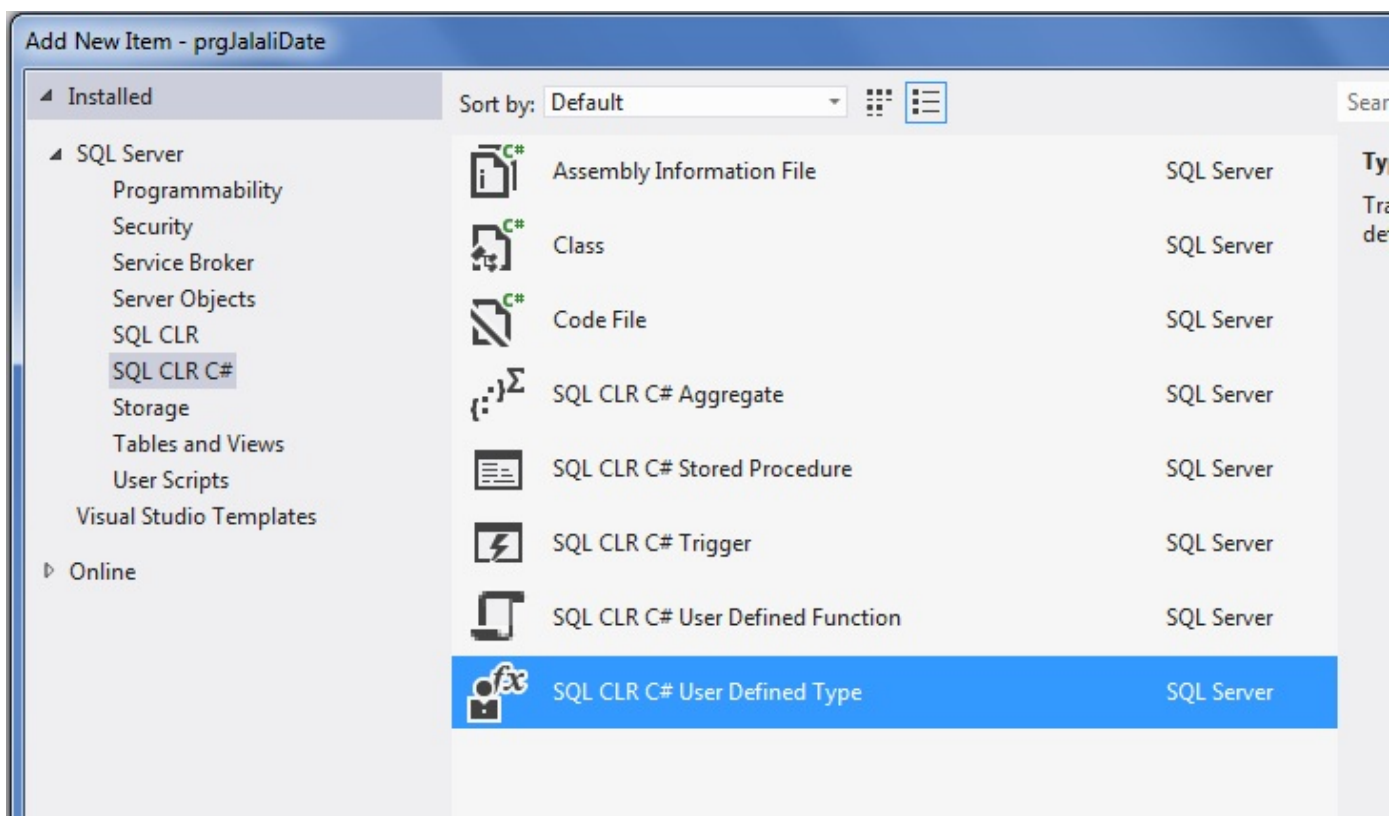
گروه‌ها:

ثبت و نگهداری تاریخ خورشیدی در SQL Server از دیرباز یکی از نگرانی‌های برنامه‌نویسان و طراحان پایگاه داده‌ها بوده است. در این نوشتار، راه‌کار تعریف یک DataType در SQL Server 2012 به روش CLR آموزش داده خواهد شد.

در ویتوال استودیو یک پروژه‌ی جدید از نوع SQL Server Database Project به شکل زیر ایجاد کنید:



نام پروژه را به یاد تقویم خیام، prgJalaliDate می‌گذارم. در Solution Explorer روی نام پروژه راست‌کلیک کرده، سپس روی Add New Item کلیک کنید. در پنجره‌ی باز شده مطابق شکل SQL CLR C# User Defined Type را برگزینید؛ سپس نام JalaliDateType را برای آن انتخاب کنید.



متن موجود در صفحه‌ی بازشده را کاملاً حذف کرده و با کد زیر جای‌گزین کنید.

(در کد زیر تمامی توابع لازم برای مقداردهی به سال، ماه، روز، ساعت، دقیقه و ثانیه و البته گرفتن مقدار از آن‌ها، تبدیل تاریخ خورشیدی به میلادی، گرفتن تاریخ به تنهایی، گرفتن زمان به تنهایی، افزایش یا کاهش زمان برپایه‌ی یکی از متغیرهای زمان و بررسی و اعتبارسنجی انواع بخش‌های زمان گنجانده شده است. در صورت پرسش یا پیشنهاد روی هر کدام در قسمت نظرات، پیام خود را بنویسید.)

```
using System;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;

[Serializable()]
[SqlUserDefinedType(Format.Native)]
public struct JalaliDate : INullable
{
    private Int16 m_Year;
    private byte m_Month;
    private byte m_Day;
    private byte m_Hour;
    private byte m_Minute;
    private byte m_Second;
    private bool is_Null;

    public Int16 Year
    {
        get
        {
            return (this.m_Year);
        }
        set
        {
            m_Year = value;
        }
    }

    public byte Month
    {
        get
        {
            return (this.m_Month);
        }
    }
}
```

```
    }
    set
    {
        m_Month = value;
    }
}

public byte Day
{
    get
    {
        return (this.m_Day);
    }
    set
    {
        m_Day = value;
    }
}

public byte Hour
{
    get
    {
        return (this.m_Hour);
    }
    set
    {
        m_Hour = value;
    }
}

public byte Minute
{
    get
    {
        return (this.m_Minute);
    }
    set
    {
        m_Minute = value;
    }
}

public byte Second
{
    get
    {
        return (this.m_Second);
    }
    set
    {
        m_Second = value;
    }
}

public bool IsNull
{
    get
    {
        return is_Null;
    }
}

public static JalaliDate Null
{
    get
    {
        JalaliDate jl = new JalaliDate();
        jl.is_Null = true;
        return (jl);
    }
}

public override string ToString()
{
    if (this.IsNull)
    {
        return "NULL";
    }
    else
```

```

    {
        return this.m_Year.ToString("D4") + "/" + this.m_Month.ToString("D2") + "/" +
this.m_Day.ToString("D2") + " " + this.Hour.ToString("D2") + ":" + this.Minute.ToString("D2") + ":" +
this.Second.ToString("D2");
    }
}

public static JalaliDate Parse(SqlString s)
{
    if (s.IsNull)
    {
        return Null;
    }

    System.Globalization.PersianCalendar pers = new System.Globalization.PersianCalendar();
    string str = Convert.ToString(s);
    string[] JDate = str.Split(' ')[0].Split('/');

    JalaliDate jl = new JalaliDate();

    jl.Year = Convert.ToInt16(JDate[0]);
    byte MonthsInYear = (byte)pers.GetMonthsInYear(jl.Year);
    jl.Month = (byte.Parse(JDate[1]) <= MonthsInYear ? (byte.Parse(JDate[1]) > 0 ?
byte.Parse(JDate[1]) : (byte)1) : MonthsInYear);
    byte DaysInMonth = (byte)pers.GetDaysInMonth(jl.Year, jl.Month); ;
    jl.Day = (byte.Parse(JDate[2]) <= DaysInMonth ? (byte.Parse(JDate[2]) > 0 ?
byte.Parse(JDate[2]) : (byte)1) : DaysInMonth);
    if (str.Split(' ').Length > 1)
    {
        string[] JTime = str.Split(' ')[1].Split(':');
        jl.Hour = (JTime.Length >= 1 ? (byte.Parse(JTime[0]) < 23 && byte.Parse(JTime[0]) >=
(byte)0 ? byte.Parse(JTime[0]) : (byte)0) : (byte)0);
        jl.Minute = (JTime.Length >= 2 ? (byte.Parse(JTime[1]) < 59 && byte.Parse(JTime[1]) >=
(byte)0 ? byte.Parse(JTime[1]) : (byte)0) : (byte)0);
        jl.Second = (JTime.Length >= 3 ? (byte.Parse(JTime[2]) < 59 && byte.Parse(JTime[2]) >=
(byte)0 ? byte.Parse(JTime[2]) : (byte)0) : (byte)0);
    }
    else { jl.Hour = 0; jl.Minute = 0; jl.Second = 0; }

    return (jl);
}

public SqlString GetDate()
{
    return this.m_Year.ToString("D4") + "/" + this.m_Month.ToString("D2") + "/" +
this.m_Day.ToString("D2");
}

public SqlString GetTime()
{
    return this.Hour.ToString("D2") + ":" + this.Minute.ToString("D2") + ":" +
this.Second.ToString("D2");
}

public SqlDateTime ToGregorianTime()
{
    System.Globalization.PersianCalendar pers = new System.Globalization.PersianCalendar();
    return SqlDateTime.Parse(pers.ToDateTime(this.Year, this.Month, this.Day, this.Hour,
this.Minute, this.Second, 0).ToString());
}

public SqlString JalaliDateAdd(SqlString interval, int increment)
{
    System.Globalization.PersianCalendar pers = new System.Globalization.PersianCalendar();
    DateTime dt = pers.ToDateTime(this.Year, this.Month, this.Day, this.Hour, this.Minute,
this.Second, 0);
    string CInterval = interval.ToString();
    bool isConvert = true;
    switch (CInterval)
    {
        case "Year":
            dt = pers.AddYears(dt, increment);
            break;
        case "Month":
            dt = pers.AddMonths(dt, increment);
            break;
        case "Day":
            dt = pers.AddDays(dt, increment);
            break;
        case "Hour":

```

```

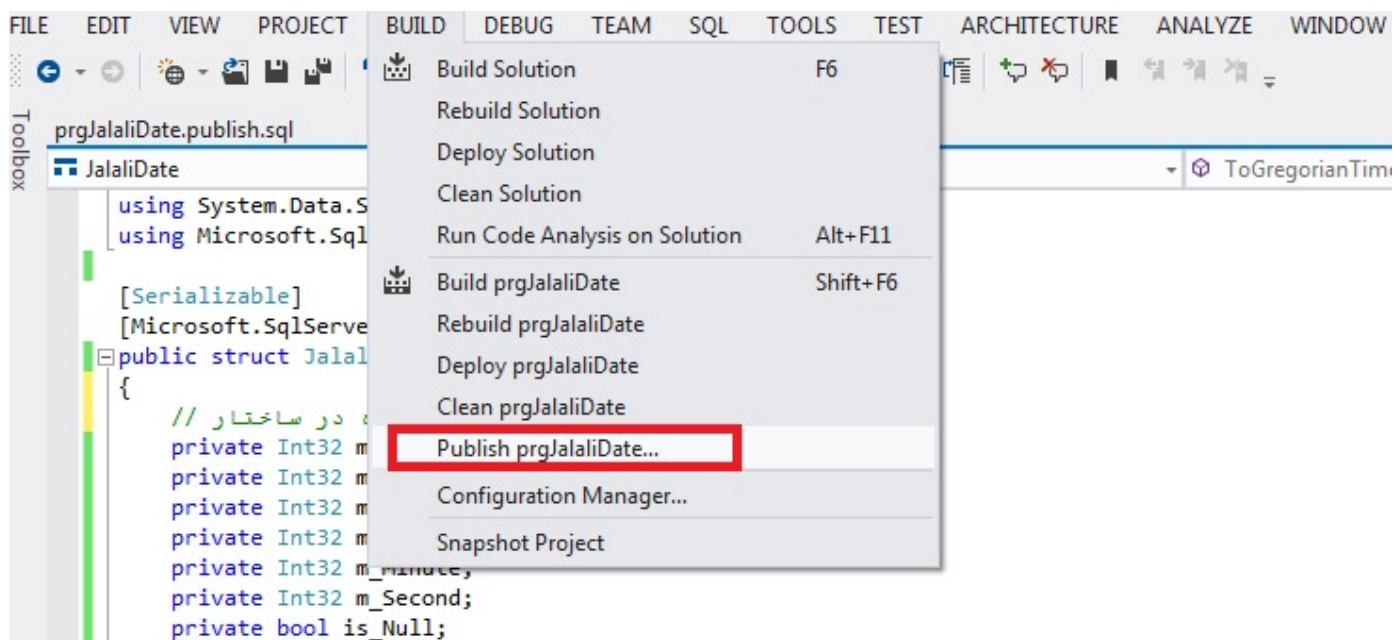
        dt = pers.AddHours(dt, increment);
        break;
    case "Minute":
        dt = pers.AddMinutes(dt, increment);
        break;
    case "Second":
        dt = pers.AddSeconds(dt, increment);
        break;
    default:
        isConvert = false;
        break;
}

if (isConvert == true)
{
    this.Year = (Int16)pers.GetYear(dt);
    this.Month = (byte)pers.GetMonth(dt);
    this.Day = (byte)pers.GetDayOfMonth(dt);
    this.Hour = (byte)pers.GetHour(dt);
    this.Minute = (byte)pers.GetMinute(dt);
    this.Second = (byte)pers.GetSecond(dt);
}

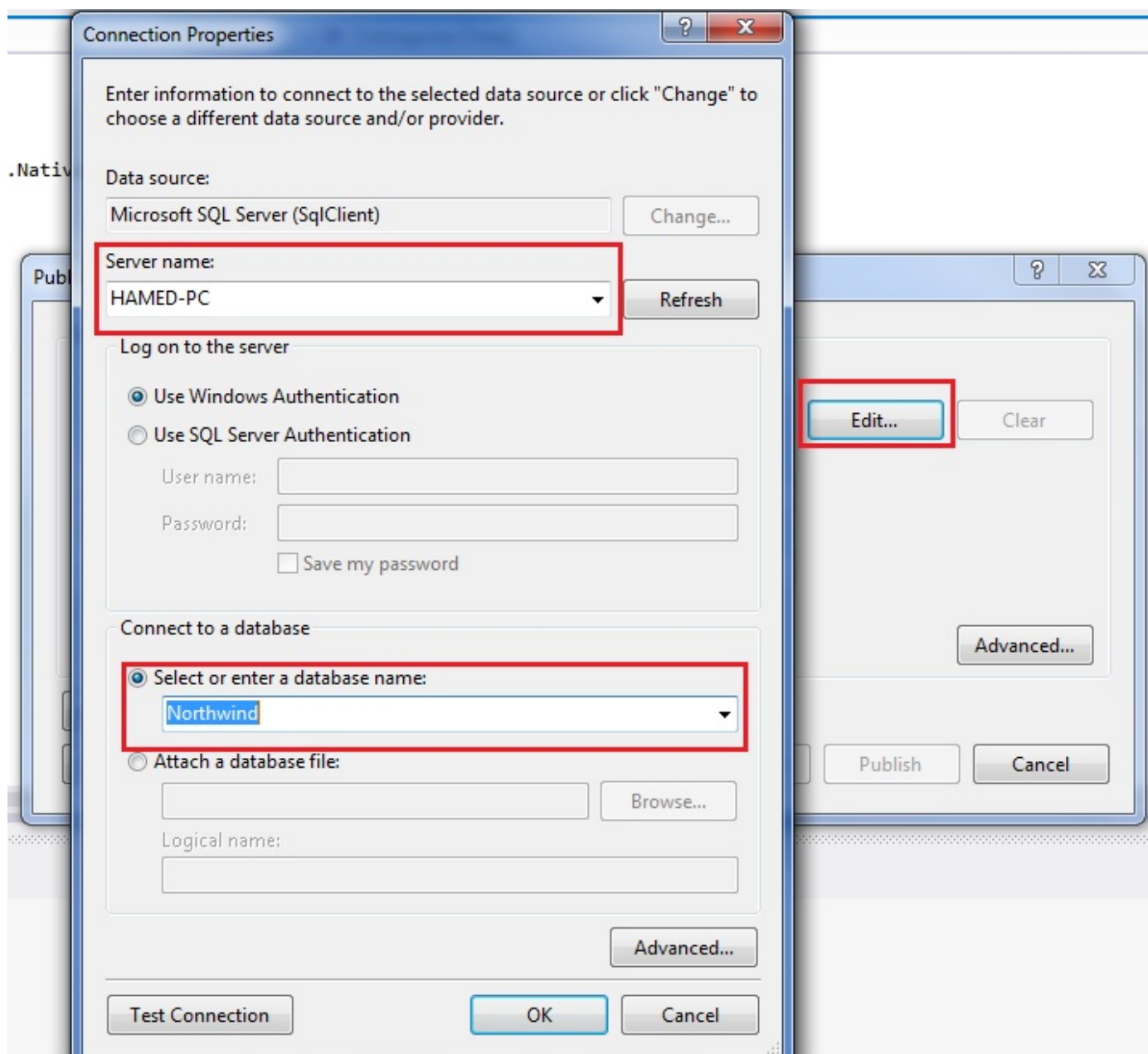
return this.m_Year.ToString("D4") + "/" + this.m_Month.ToString("D2") + "/" +
this.m_Day.ToString("D2") + " " + this.Hour.ToString("D2") + ":" + this.Minute.ToString("D2") + ":" +
this.Second.ToString("D2");
}
}

```

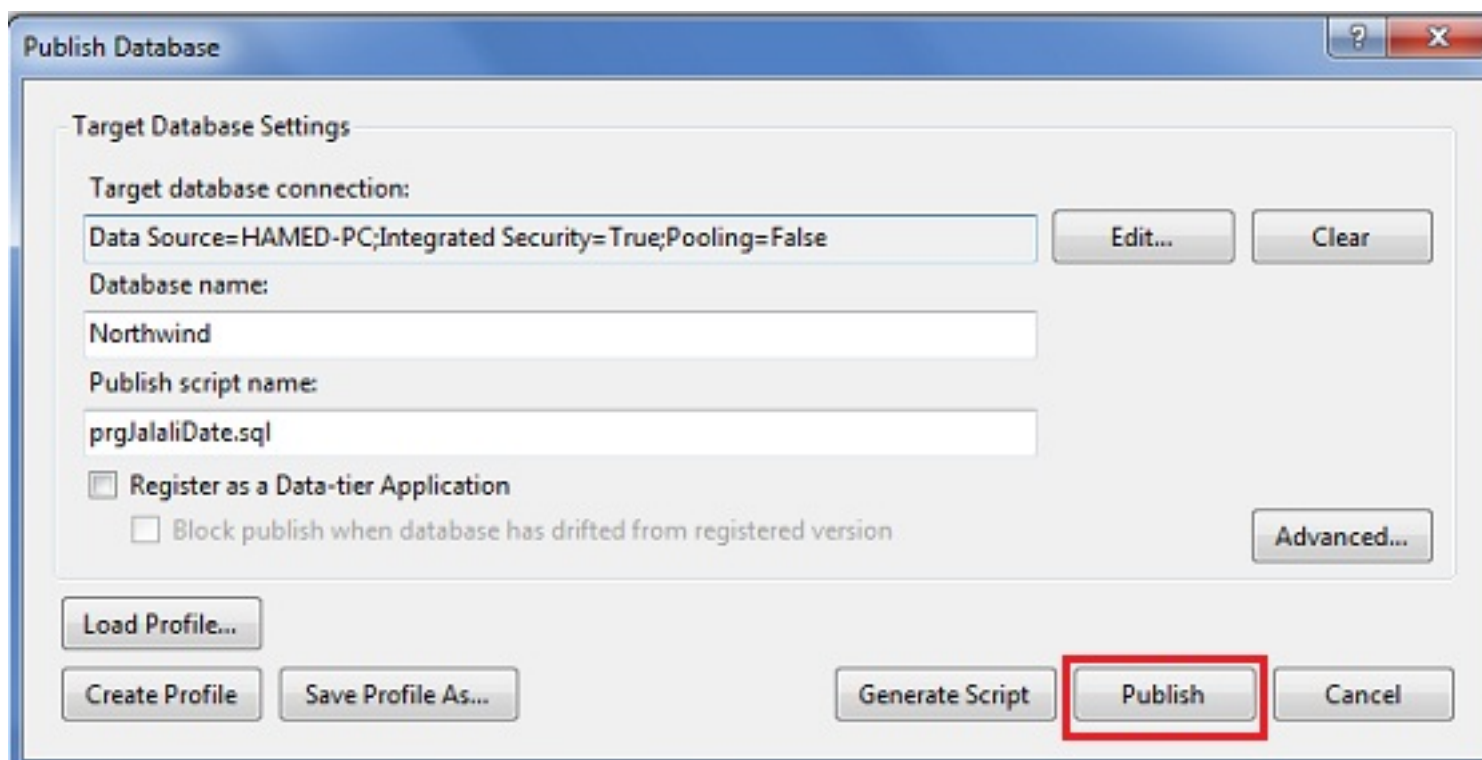
از منوهای بالا روی منوی Build و سپس گزینه‌ی Publish prgJalaliDate کلیک کنید:



در پنجره‌ی بازشده روی دکمه‌ی Edit کلیک کنید سپس تنظیمات مربوط به اتصال به پایگاه داده را انجام دهید.



روی دکمه‌ی OK کلیک کنید و سپس در پنجره‌ی اولیه، روی دکمه‌ی Publish کلیک کنید:



به همین سادگی، DataType مربوطه در SQL Server 2012 ساخته می‌شود. خبر خوش این‌که شما می‌توانید با راست‌کلیک روی نام پروژه و انتخاب گزینه‌ی Properties در قسمت Project Setting تنظیمات مربوط به نگارش SQL Server را انجام دهید. (از نگارش 2005 به بعد در VS 2012 پشتیبانی می‌شود).

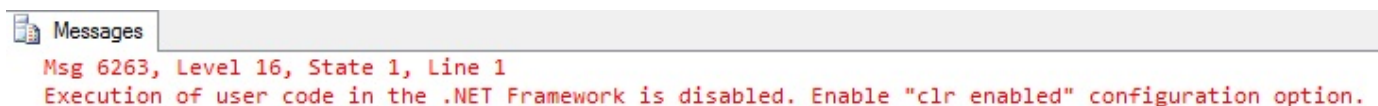
اکنون زمان آن رسیده است که DataType ایجادشده را در SQL Server 2012 بیازماییم. SQL Server را باز کنید و دستور زیر را در آن اجرا کنید.

```
USE Northwind
GO
CREATE TABLE dbo.TestTable
(
  Id int NOT NULL IDENTITY (1, 1),
  TestDate dbo.JalaliDate NULL
) ON [PRIMARY]
GO
```

همین‌طور که مشاهده می‌کنید؛ امکان به‌کارگیری DataType تعریف‌شده وجود دارد. اکنون چند رکورد درون این جدول درج می‌کنیم:

```
Insert into TestTable (TestDate) Values ('1392/02/09'),('1392/02/09 22:40'),('1392/12/30 22:40')
```

پس از اجرای این دستور خطای زیر در پایین صفحه‌ی SQL Server نمایان می‌شود:



این خطا به این خاطر است که CLR را در SQL Server فعال نکرده ایم. جهت فعال کردن CLR دستور زیر را اجرا کنید:

```
sp_configure 'clr enabled', 1
Reconfigure
```

بار دیگر دستور درج را اجرا می‌کنیم:

```
Insert into TestTable (TestDate) Values ('1392/02/09'),('1392/02/09 22:40'),('1392/12/30 22:40')
```

	Id	TestDate
▶	1	1392/02/09 00:00:00
	2	1392/02/09 22:40:00
	3	1392/12/29 22:40:00
*	NULL	NULL

ملاحظه می‌کنید که داده‌ها در جدول مربوطه ذخیره شده است. در رکورد نخست چون ساعت، دقیقه و ثانیه تعریف نشده است؛ به طور هوشمند صفر درج شده است. در رکورد دوم، ساعت و دقیقه مقدار دارد ولی ثانیه صفر ثبت شده است. و در رکورد سوم چون سال 1392 کیبسه نیست؛ به صورت هوشمند آخرین روز ماه به جای روز ثبت شده است. هرچند می‌توان با دست‌کاری در توابع سی‌شارپ، این قوانین را عوض کرد.

اکنون زمان آن رسیده است که توسط یک پرس‌وجو، تمامی توابعی که در سی‌شارپ برای این نوع داده نوشتیم، بیازماییم. پرس‌وجوی زیر را اجرا کنید:

```
Select TestDate.ToString() as JalaliDateTime,
       TestDate.GetDate() as JalaliDate, TestDate.GetTime() as JalaliTime,
       TestDate.ToGregorianTime() as GregorianTime,
       TestDate.JalaliDateAdd('Day',1) JalaliTomorrow,
       TestDate.Month as JalaliMonth from TestTable
```

خروجی این پرس‌وجو به شکل زیر خواهد بود:

	JalaliDateTime	JalaliDate	JalaliTime	GregorianTime	JalaliTomorrow	JalaliMonth
1	1392/02/09 00:00:00	1392/02/09	00:00:00	2013-04-29 00:00:00.000	1392/02/10 00:00:00	2
2	1392/02/09 22:40:00	1392/02/09	22:40:00	2013-04-29 22:40:00.000	1392/02/10 22:40:00	2
3	1392/12/29 22:40:00	1392/12/29	22:40:00	2014-03-20 22:40:00.000	1393/01/01 22:40:00	12

البته درباره‌ی ستون پنجم و ششم شما می‌توانید روی تمامی اجزای تاریخ افزایش و کاهش داشته باشید و همچنین می‌توانید با تابع مربوطه هر کدام از اجزای زمان را جداگانه به دست بیاورید که در این مثال عدد ماه نشان داده شده است.

نیازی به گفتن نیست که می‌توانید به سادگی از توابع مربوط به DateTime در SQL Server بهره ببرید. برای مثال برای به دست

آوردن فاصله‌ی میان دو روز از پرس‌وجوی زیر استفاده کنید:

```
Declare @a JalaliDate = '1392/02/07 00:00:00'  
Declare @b JalaliDate = '1392/02/05 00:00:00'  
  
SELECT DATEDIFF("DAY",@b.ToGregorianTime(),@a.ToGregorianTime()) AS DiffDate
```

شاد و پیروز باشید.

نظرات خوانندگان

نویسنده: ali

تاریخ: ۱۳۹۲/۰۲/۱۰ ۰:۵

Nice article

Thanks

نویسنده: سام ناصری

تاریخ: ۱۳۹۲/۰۲/۱۰ ۴:۲۲

بسیار خوب.

فقط من تو مقاله شما دلیلی برای اینکه چرا زمان و تاریخ را میخواهی به این صورت ذخیره کنی متوجه نشدم؟ چرا به همان شکل استانداردش ذخیره نکنیم؟
دیگر اینکه نوع داده جدید به چه شکل در دیتابیس ذخیره میشود. Sql Server از کجا میداند که باید چگونه لیترالها را پارس کند(چگونه متود Parse برگزیده میشود)؟ عملگرهای مقایسه چگونه کار خواهند کرد؟

نویسنده: حامد قنادی

تاریخ: ۱۳۹۲/۰۲/۱۰ ۷:۸

با درود

به پرسش‌های شما در بخش دوم پاسخ خواهم داد.

نویسنده: قاسم

تاریخ: ۱۳۹۲/۰۲/۱۰ ۹:۴۲

سلام، میشه مراحل انجام کار توی VS2010 هم بنویسید، فقط مراحل کار. خیلی ممنون میشم

نویسنده: فرشید علی اکبری

تاریخ: ۱۳۹۲/۰۲/۱۰ ۹:۵۵

سلام

کدهای شمارو درست همونطوریکه گفتین کپی کردم ولی موقعی که Publish رو میزنم پیغام زیر رو میده :

Publish cannot begin until your project(s) build successfully

و پنجره publish ظاهر نمی‌شه و هیچ جایی هم برای تعریف کانکشن وجود نداره... مشکل از کجاست؟

دوم اینکه clr رو هربار که خواستیم روی یک دیتابیس جدید این نوع رو تعریف کنیم باید فعال کنیم... مثلاً توی ایجاد سال مالی جدید که سیستم یک دیتابیس خام ایجاد میکنه تا مانده حسابها رو بهش انتقال بدیم قبلش باید دستور فعال سازی clr رو هم مجدداً بدیم یا فقط توی زمان طراحی کفایت میکنه ؟

سوم اینکه : در مورد استفاده اون توی EF Code First هم اگه نکته‌ی خاصی وجود داره محبت کنین ممنون میشم.
شاد و پیروز باشید.

نویسنده: حامد قنادی

تاریخ: ۱۳۹۲/۰۲/۱۰ ۱۱:۴

با درود

New Project -> Database -> SQL Server -> Visual C# SQL CLR Database Project

- تنظیمات اتصال به پایگاه داده ها

- انتخاب دکمه Yes

Add New Item -> User Defined Type -

- کپی کدها

- استفاده از Deploy در منوی Build یا استفاده از روشی که در بخش دوم نوشتار آموزش داده خواهد شد.

نویسنده: حامد قنادی

تاریخ: ۱۱:۲۶ ۱۳۹۲/۰۲/۱۰

با درود

- 1- لطفاً پروژه را پیش از Publish یک بار Rebuild کنید. احتمالاً به خاطر یک ارور خاص Publish نمی‌شود. اگر به راحتی Rebuild شده ولی باز هم Publish کار نمی‌کند؛ می‌توانیم با روش دیگری که در بخش دوم آموزش می‌دهم DLL را به SQL معرفی کنیم.
- 2- کافی است یک بار پس ساخت Database این پرس‌وجو را اجرا کنید.
- 3- تست نکردم ولی به نظر می‌رسد این نوع داده از سمت EF شبیه به Hierarchy باشد.

نویسنده: ش.د.

تاریخ: ۱۴:۳۵ ۱۳۹۲/۰۲/۱۰

آیا فقط در sql2012 قابل اجرا می‌باشد؟

نویسنده: محسن خان

تاریخ: ۱۴:۵۴ ۱۳۹۲/۰۲/۱۰

در متن نوشتن از نگارش 2005 به بعد اس کیوال سرور این قابلیت استفاده از افزونه‌های CLR اضافه شدن.

نویسنده: سید امیر سجادی

تاریخ: ۱۸:۳۸ ۱۳۹۲/۰۲/۱۰

با تشکر از مطلب مفیدتون.

چند تا سوال برام پیش اومده.

اول اینکه آیا به صورت یک DLL به بانک اضافه میشه؟

دوم اینکه اگه از بانک بک آپ بگیریم و جایی دیگه خواستیم اون رو ریستور کنیم چی میشه؟

آپلود بانک روی هاست (بک آپ یا اتچ) ؟

نویسنده: امیر بختیاری

تاریخ: ۹:۴۳ ۱۳۹۲/۰۲/۱۱

به صورت اسمبلی به بانک اضافه میشه

وقتی بک آپ بگیرید و ریستور کنید همراه بک آپ این اسمبلی هم انتقال داده می‌شود

در اتچ هم به همین شکل

فقط زمانی که از این اسمبلی در توابع و پروسیجرها استفاده می‌کنید نمی‌تونید حذفش کنید و در صورت تغییرات باید اسمبلی را به روز کنید.

یک مورد دیگه که من زیاد تو هاست‌های شیر شده بهش برخوردم اینه که این امکان توشون فعال نیست و ادمین سرورها هم به سختی این امکان را فعال می‌کنند پس اگر خواستید از این امکان استفاده کنید ابتدا از فعال بودن آن مطمئن شوید

نویسنده: rahim

تاریخ: ۱۲:۱۴ ۱۳۹۳/۰۵/۱۱

با تشکر از مطلب مفیدتون

من زمانی که از این روش استفاده میکنم در هنگام درج رکورد جدید با پیغام خطای زیر مواجه میشم.

:"A .NET Framework error occurred during execution of user-defined routine or aggregate "JalaliDate

.System.OverflowException: Value was either too large or too small for an unsigned byte

نویسنده: محمد
تاریخ: ۱۳۹۳/۰۸/۲۲ ۰:۲۳

با سلام؛ من زمانی که این کد را در SQL 2012 می‌کنم یک Error در خصوص ToGregorianCalendar() نمایش داده میشه. لطفا راهنمایی بفرمایید. با تشکر

```
Msg 6522, Level 16, State 2, Line 1
A .NET Framework error occurred during execution of user-defined routine or aggregate
"SpatialDateTime":
System.FormatException: String was not recognized as a valid DateTime.
System.FormatException:
at System.DateTime.ParseExactMultiple(String s, String[] formats, DateTimeFormatInfo dtfi,
DateTimeStyles style)
at System.DateTime.ParseExact(String s, String[] formats, IFormatProvider provider, DateTimeStyles
style)
at System.Data.SqlTypes.SqlDateTime.Parse(String s)
at SpatialDateTime.ToGregorianCalendar().
```

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۸/۲۲ ۰:۵۵

یک try/catch بذار، تا بتونی تاریخ مشکل دار رو پیدا کنی:

```
var pers = new PersianCalendar();
var date = pers.ToDateTime(this.Year, this.Month, this.Day, this.Hour, this.Minute, this.Second,
0).ToString();
try
{
    return SqlDateTime.Parse(date);
}
catch(Exception ex)
{
    throw new InvalidOperationException("Can't parse "+ date);
}
```

پیش از هرچیز به شما پیشنهاد می‌کنم؛ بار دیگر کد سی‌شارپ درس نخست را در پروژه‌ی خود کپی کنید و سپس Publish را بزنید. پس از ارسال آن مطلب، تغییراتی در جهت بهینه‌سازی کد دادم که به نظرم بهتر است شما نیز در پروژه‌ی خود به کار برید.

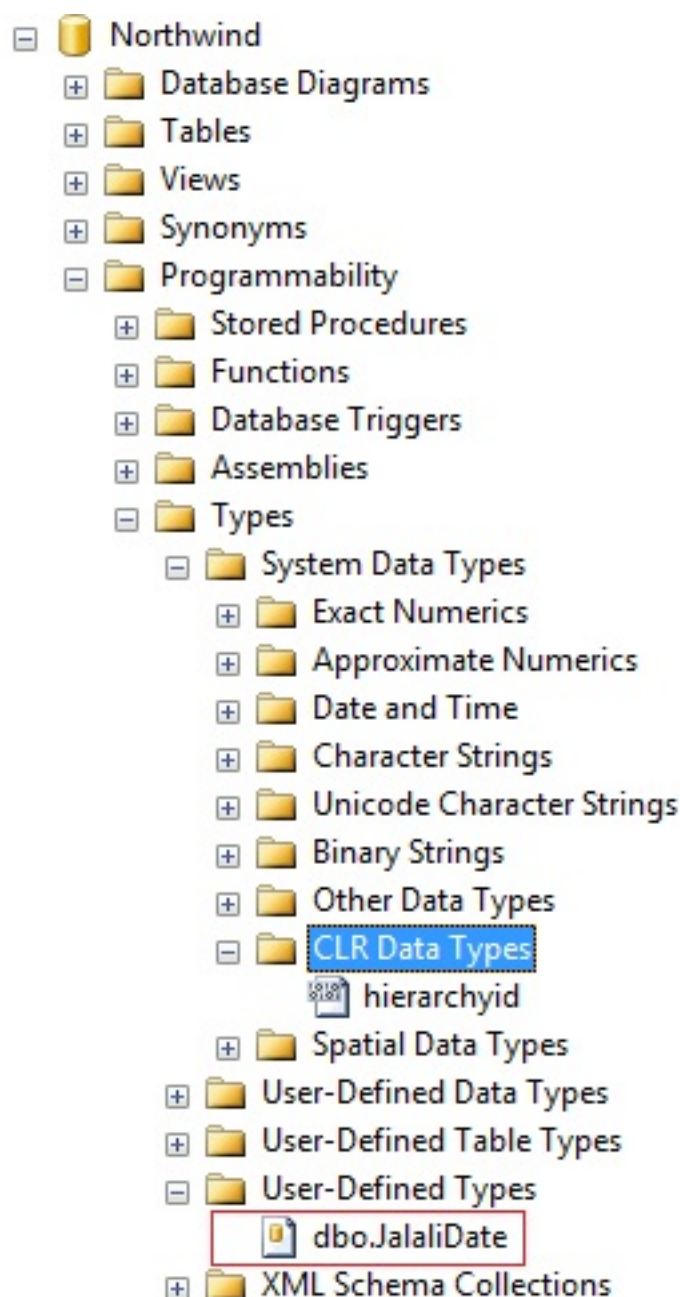
چرا از این نوع داده استفاده کنیم؟

نخستین پرسشی که ممکن است برای شما پیش بیاید این است که چرا بهتر است از این نوع داده استفاده کنیم. برای پاسخ به این پرسش باید راه‌کارهای گذشته را بررسی کنیم. معمولاً طراحان پایگاه داده‌ها برای استفاده از تاریخ خورشیدی، زمان را به صورت میلادی ثبت می‌کنند؛ سپس با یک scalar-valued function زمان درج شده را به خورشیدی تبدیل می‌کنند. در این صورت می‌توان با یک تابع کوچک دیگر بخش مربوط به ساعت را نیز از همان ستون به دست آورد. در این صورت می‌توانیم از کلیه‌ی متدهای مربوط به DateTime در SQL از جمله افزایش و کاهش و تفاضل دو تاریخ بهره برد. برخی دیگر از طراحان، ستونی از نوع char(10) در نظر می‌گیرند و تاریخ خورشیدی را به صورت ده‌کاراکتری در آن ذخیره می‌کنند. این روش هرچند نیاز به تبدیل به خورشیدی را ندارد ولی کلیه‌ی مزایایی که در استفاده از DateTime به آن‌ها دسترسی داریم از دست می‌دهیم. افزون بر این جهت نگهداری زمان باید یک فیلد دیگر از نوع کاراکتری و یا در نگارش‌های نوین‌تر از نوع time تعریف کنیم. برخی دیگر از هر دو را در کنار هم استفاده می‌کنند و در واقع جهت سرعت بالاتر نمایش و بررسی داده‌ها از طریق محیط SQL Server از فیلد کاراکتری تاریخ خورشیدی و برای مقایسه و بدست آوردن ساعت از فیلد نوع DateTime استفاده می‌کنند.

از نظر فضای اشغال‌شده نوع DateTime، هشت بایت، smalldatetime (در صورت استفاده) 4 بایت و فیلد 10 کاراکتری تاریخ 10 بایت فضا اشغال می‌کند در صورتی که نوع JalaliDate با در نظر گرفتن همه‌ی مزایای انواع داده‌ی استفاده‌شده برای تاریخ، فقط 8 بایت فضا اشغال می‌کند. با استفاده از این نوع به راحتی داده‌ی تاریخ را بر اساس تقویم ایرانی اعتبارسنجی می‌کنید و بخش‌های مختلف زمان از سال تا ثانیه را با یک متد به دست می‌آورید. می‌توانید به راحتی به تاریخ خود زمانی را بیفزایید یا بکاهید و در گزارش‌ها بدون نگرانی از تبدیل درست استفاده کنید. چون کدباز است می‌توانید با کمی حوصله امکانات دیگر مد نظر خود را به آن بیفزایید و از آن در SQL بهره ببرید.

چگونه این نوع داده را حذف کنیم؟!

شما می‌توانید به سادگی نوع داده‌ی ایجادشده توسط CLR را در مسیر زیر بیابید و اقدام به حذف آن نمایید:



همان‌طور که مشاهده می‌شود؛ حتی نوع داده‌ی سیستمی hierarchyid که جهت ساختار سلسله‌مراتبی مانند چارت سازمانی یا درخت تجهیزات استفاده می‌شود؛ نیز یک نوع داده‌ی CLR است.

آیا راه دیگری نیز برای افزودن این نوع داده به SQL به جز Publish کردن وجود دارد؟

مانند بسیاری دیگر از گونه‌های پروژه، در اینجا نیز شما یک فایل DLL خواهید داشت. این فایل برپایه‌ی تنظیماتی که شما در قسمت Properties پروژه‌ی خود انجام می‌دهید ساخته می‌شود. پس از تغییر مسیر فایل DLL در دستور زیر توسط یک New Query از Database خود، آن را اجرا کنید:

```
CREATE ASSEMBLY JalaliDate
FROM 'F:\prgJalaliDate.dll'
WITH PERMISSION_SET = SAFE;
```

هم‌چنین در صورت ویرایش‌های دوباره پروژه از دستور زیر استفاده کنید:

```
ALTER ASSEMBLY JalaliDate
FROM 'F:\prgJalaliDate.dll'
```

با استفاده از دستورهای زیر می‌توانید از چگونگی درج فایل‌های افزوده شده آگاه شوید:

```
select * from sys.assemblies
select * from sys.assembly_files
```

Results Messages

	name	principal_id	assembly_id	clr_name	permission...	permission_set_desc	is_visible	create_date	modify_date	is_user_defined
1	Microsoft.SqlServer.Types	4	1	microsoft.sqlserver.types, ver...	3	UNSAFE_ACCESS	1	2012-02-10 20:...	2012-02-10 ...	0
2	prgJalaliDate	1	65536	prgJalaliDate, version=0.0.0.0...	1	SAFE_ACCESS	1	2013-04-29 22:...	2013-04-29 ...	1

	assembly_id	name	file_id	content
1	1	microsoft.sqlserver.types.dll	1	0x4D5A90000300000004000000FFFF0000B8000000000000...
2	65536	prgJalaliDate	1	0x4D5A90000300000004000000FFFF0000B8000000000000...
3	65536	prgJalaliDate.pdb	2	0x4D6963726F736F667420432F432B2B204D534620372E30...

تا اینجا SQL Server، دی‌ال‌ال مربوط به پروژه را شناخته است. برای تعریف نوع داده از دستور زیر بهره ببرید:

```
CREATE TYPE dbo.JalaliDate
EXTERNAL NAME JalaliDate.[JalaliDate];
```

این کار همانند استفاده از گزینه‌ی Publish در Visual Studio است.

هم‌چنین چنان‌چه در SQL Server 2012 از منوی راست‌کلیک پایگاه داده‌ها روی گزینه Tasks و سپس Generate Scripts را انتخاب کنیم، از مشاهده‌ی سند ساخته شده، درخواهیم یافت که حتی دستورهای مربوط به ساخت اسمبلی CLR با تبدیل فایل به کد در Scripts وجود دارد و با اجرای آن در سروری دیگر، انتقال می‌یابد.

```
GO
/***** Object: SqlAssembly [prgJalaliDate]    Script Date: 2013/04/30 08:27:00 ب.ظ *****/
CREATE ASSEMBLY [prgJalaliDate]
FROM 0x4D5A90000300000004000000FFFF0000B8000000000000 ..... بقیه‌ی کدها حذف شده
WITH PERMISSION_SET = SAFE

GO
ALTER ASSEMBLY [prgJalaliDate]
ADD FILE FROM 0x4D6963726F736F667420432F432B2B204D534620372E30300D0A1A44530 ..... بقیه‌ی کدها حذف شده
AS N'prgJalaliDate.pdb'

GO
/***** Object: UserDefinedType [dbo].[JalaliDate]    Script Date: 2013/04/30 08:27:00 ب.ظ *****/
CREATE TYPE [dbo].[JalaliDate]
EXTERNAL NAME [prgJalaliDate].[JalaliDate]

GO
```

دنباله دارد ...

نظرات خوانندگان

نویسنده: محمد عادل
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱:۴۳

در این حالت ، چطور میتونیم در EF Code First از این DataType استفاده کنیم ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۱ ۸:۳۴

از برنامه [Entity Framework Power Tools](#) برای مهندسی معکوس ساختار موجود استفاده کنید و بعد مشاهده کنید که چه کدی رو تولید می‌کنه. کار شما در این حالت code first نیست.

نویسنده: حامد حسین نژاد
تاریخ: ۱۳۹۲/۰۲/۱۱ ۹:۲۰

البته باید این را هم در نظر گرفت که کلا CLR Integration باعث کندی دیتابیس می‌شود و انواع داده CLR بسیار کندتر از بقیه اجرا می‌شوند. این موضوع ممکن است که در دیتابیس‌های کوچک چندان مهم بنظر نیاید ولی در دیتابیس‌های بزرگ باعث بروز مشکل خواهد شد. علاوه بر این استفاده از انواع داده CLR و یا توابع CLR در دیتابیس باعث می‌شود که امکان پارتیشن بندی جداول آن دیتابیس وجود نداشته باشد.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۱:۳۴

[مقاله‌ای هست اینجا](#) در مورد کارآیی CLR در SQL Server. به نظر می‌رسه سریعتر است حدود 11 درصد نسبت به T-SQL معمولی. برای پارتیشن بندی می‌تونید اینکار رو انجام بدید فقط این نوع خاص قابل انتخاب نیست. مابقی فیلدها [رو می‌تونید](#) انتخاب کنید.

نویسنده: حامد حسین نژاد
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۸:۳۱

البته کارایی CLR، بسته به مورد استفاده، متفاوت ([این لینک](#)). در مواردی مثل همین مثال، اگر تعداد سطور جدول زیاد باشه، کارایی رو به شدت کاهش میده. مخصوصا اگر بخواین از Data Warehousing هم استفاده کنید.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۹:۲۸

این به شدت رو می‌تونید در موردش عدد و رقم ارائه بدید؟ در همون لینکی که دادید عنوان شده پیاده سازی RegEx روی سطور بالا خیلی سریع‌تر هست با CLR. در هر حال بهتره با عدد و رقم و محاسبات بحث کرد.

نویسنده: حامد حسین نژاد
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۴:۲۸

مثلا فرض کنید جدولی دارید که اطلاعات Task‌های یک شرکت رو نگه میداره که شامل تاریخ شروع و تاریخ پایان هر Task هم هست. اگر این جدول یک میلیون سطر داده داشته باشه و بخوایم Task‌هایی که مدت زمان انجام اونها کمتر از 5 روز بوده رو انتخاب کنیم تفاوت فاحشی با datetime خواهیم داشت. البته، همونطور که گفتم بسته به نوع استفاده داره و ممکنه از دیتابیس به دیتابیس دیگه فرق بکنه.

در دو درس گذشته با DataType ای که با هم نوشتیم؛ می‌توانستیم این پرس‌وجوها را داشته باشیم:

```
Declare @MyDate JalaliDate = '1392/02/11 21:38:24'
Select @MyDate.ToString() as MyDateTime
      , @MyDate.GetDate() as MyDate
      , @MyDate.GetTime() as MyTime
      , @MyDate.Year as MyYear
      , @MyDate.Month as MyMonth
      , @MyDate.Day as MyDay
      , @MyDate.Hour as MyHour
      , @MyDate.Minute as MyMinute
      , @MyDate.Second as MySecond
      , @MyDate.JalaliDateAdd('Year',1) as NextYear
      , @MyDate.JalaliDateAdd('Month',1) as NextMonth
      , @MyDate.JalaliDateAdd('Day',1) as NextDay
      , @MyDate.JalaliDateAdd('Hour',1) as NextHour
      , @MyDate.JalaliDateAdd('Minute',1) as NextMinute
      , @MyDate.JalaliDateAdd('Second',1) as NextSecond
```

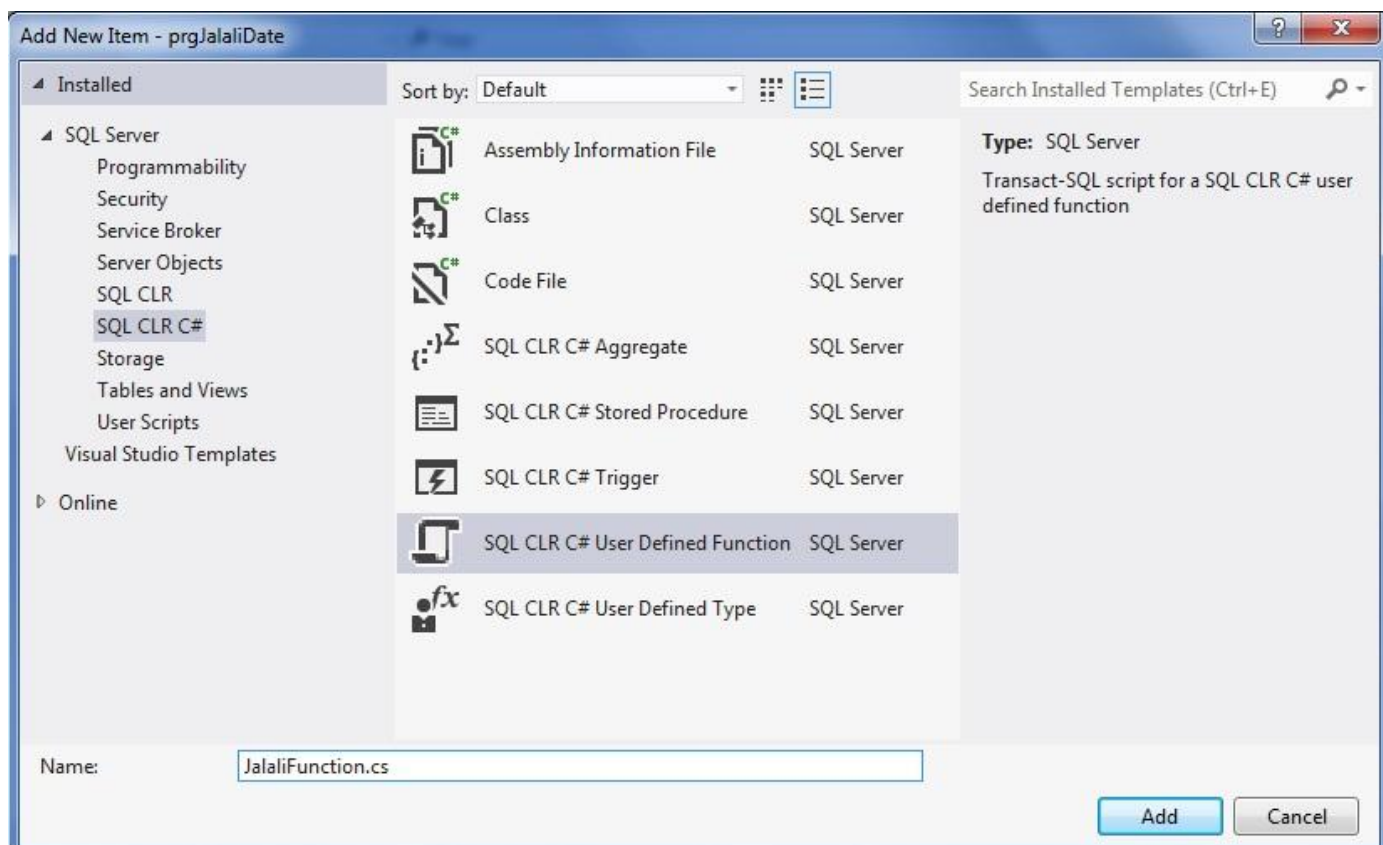
که در نهایت خروجی کار به شکل زیر خواهد بود:

Results										
	MyDateTime	MyDate	MyTime	MyYear	MyMonth	MyDay	MyHour	MyMinute	MySecond	
1	1392/02/11 21:38:24	1392/02/11	21:38:24	1392	2	11	21	38	24	

NextYear	NextMonth	NextDay	NextHour	NextMinute	NextSecond
1393/02/11 21:38:24	1392/03/11 21:38:24	1392/02/12 21:38:24	1392/02/11 22:38:24	1392/02/11 21:39:24	1392/02/11 21:38:25

با این همه، همان‌سان که در پرس‌وجوی بالا هم می‌بینید؛ ناگزیر شدم تاریخ و زمان را خودم وارد کنم و هرآینه در واقعیت شما جهت پرس‌وجو روی زمان کنونی، ناگزیر به استفاده از یک تابع برای تبدیل تقویم میلادی به خورشیدی هستیم. به نظر شما بهتر نیست دست به کار شویم و تابعی برای تبدیل تاریخ میلادی به خورشیدی بنویسیم؟

برای این‌کار پروژه‌ای را که در دو درس 1 ساختیم باز کنید و سپس روی نام پروژه در Solution Explorer راست‌کلیک کرده و Add New Item را انتخاب کنید.



محتویات فایل باز شده را حذف کنید و دستورهای زیر را جایگزین کنید:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;

public partial class UserDefinedFunctions
{
    [Microsoft.SqlServer.Server.SqlFunction]
    public static JalaliDate GetCurrentDateTime()
    {
        System.Globalization.PersianCalendar pers = new System.Globalization.PersianCalendar();
        DateTime CurrentDate = DateTime.Now;
        JalaliDate jl;
        jl.Year = (Int16)pers.GetYear(CurrentDate);
        jl.Month = (byte)pers.GetMonth(CurrentDate);
        jl.Day = (byte)pers.GetDayOfMonth(CurrentDate);
        jl.Hour = (byte)pers.GetHour(CurrentDate);
        jl.Minute = (byte)pers.GetMinute(CurrentDate);
        jl.Second = (byte)pers.GetSecond(CurrentDate);
        return jl;
    }
}
```

بار دیگر پروژه را به روشی که در درس یک آموزش داده شد؛ Publish کنید. اکنون می‌توانیم تاریخ و زمان کنونی را در متغیری از نوع JalaliDate وارد کنیم.

```
Declare @MyDate JalaliDate = dbo.GetCurrentDateTime()
```

یا به این شکل تاریخ و زمان جاری را مشاهده کنیم:

```
Select dbo.GetCurrentDateTime().ToString()
```

Results		Messages	
		(No column name)	
1		1392/02/13 13:12:31	

دنباله دارد...

نظرات خوانندگان

نویسنده: فرشید علی اکبری
تاریخ: ۱۰:۳۴ ۱۳۹۲/۰۲/۱۴

از دوستان اگه کسی نحوه استفاده این تکنیک رو توی EF Code First میدونه لطفا مطلب رو Share کنه تا امکان استفاده این دسته از عزیزان هم از زحمتی که آقای قنادی کشیدن باشه.
با تشکر.

نویسنده: محسن خان
تاریخ: ۱۰:۴۲ ۱۳۹۲/۰۲/۱۴

```
public DateTime AddDate { set; get; }

[NotMapped] // فیلد محاسباتی سمت برنامه که در بانک اطلاعاتی قرار نخواهد گرفت
public string PersianDate
{
    get { return MyDateConverter(AddDate); }
}
```

در code first از ویژگی NotMapped استفاده کنید تا بتونید با استفاده از تابع کمکی تبدیل تاریخ خودتون راحت این تبدیلات رو انجام بدید. در بانک اطلاعاتی DateTime استاندارد رو ذخیره کنید، در سمت کدها برای مسایل نمایشی از خاصیت PersianDate که NotMapped تعریف شده، میشه استفاده کرد. به علاوه روی DateTime استاندارد راحت میشه کوئری‌های LINQ رو اجرا کرد بدون محدودیت. زمانیکه مثلا یک List تهیه شد، قسمت بعدی، کار نمایشی است که از خاصیت NotMapped میشه کمک گرفت. این روش با تمام بانک‌های اطلاعاتی کار می‌کنه.

نویسنده: وحید نصیری
تاریخ: ۱۱:۴۱ ۱۳۹۲/۰۲/۱۴

```
[Column(TypeName = "xml")]
public string XmlValue { get; set; }

[NotMapped]
public XElement XmlValueWrapper
{
    get { return XElement.Parse(XmlValue); }
    set { XmlValue = value.ToString(); }
}
```

روش عمومی کار با نوع‌های خاصی که در EF تعریف نشدن، استفاده از ویژگی Column و مشخص کردن Type آن است؛ مانند مثالی که در بالا ملاحظه می‌کنید. البته این نوع خاص، در سمت کدها باید به صورت رشته تعریف شود. مثلا از سال 2005 به این طرف فیلد XML به SQL Server اضافه شده. اما نمی‌شود ارزش در EF به همون شکل XML استفاده کرد. [باید تبدیلیش کنی به String](#) تا قابل استفاده بشه. یک نمونه دیگرس نوع خاص Spatial هست که در نگارش‌های اخیر SQL Server اضافه شده (geography و geometry). این مورد فقط از EF 5.0 به بعد پشتیبانی توکاری ارزش ارائه شده. یا برای [hierarchyID](#) در EF معادلی وجود نداره. برای تعریف این مورد نیز در یک مدل باید از string استفاده کرد.

بعد اگر این نوع خاص (که الان به صورت رشته دریافت شده) قابل نگاشت به نوعی مشخص در سمت کدهای برنامه بود (یعنی صرفا یک رشته ساده نبود) مثلا می‌شود از ویژگی NotMapped برای تبدیل آن و تعریف آن به شکل یک فیلد محاسباتی استفاده کرد.

با سلام

پروژه ای که شرح دادین رو ایجاد و در Publish ، SQL server 2012 کردم و در جدول هم مقادیر تستی درج کردم. زمانی که جدول را در Object Browser با Mode ویرایش باز می‌کنم هیچ مشکلی وجود ندارد و داده‌ها درست نمایش داده می‌شوند اما زمانی که با دستورات T-SQL کار میکنم مقادیر را به صورت یک رشته از کاراکترها نمایش می‌دهد که نامفهوم می‌باشد. تصویر اجرای کوئری‌ها را می‌ذارم لطفا راهنمایی کنید.

	Id	TestDate
▶	۱	۱۳۹۲/۰۲/۰۹ ++:++:++
	۲	۱۳۹۲/۰۲/۰۹ ۲۲:۴:++
	۳	۱۳۹۲/۱۲/۲۹ ۲۲:۴:++
*	NULL	NULL

```
select *
from TestTable
```

۱۰۰ %

Results Messages

	Id	TestDate
۱	۱	•x۸۵۷•۰۲•۰۹••••••••••
۲	۲	•x۸۵۷•۰۲•۰۹۱۶۲۸•••••
۳	۳	•x۸۵۷•۰C۱D۱۶۲۸•••••

```
select dbo.GetCurrentDateTime()
```

۱۰۰ %

Results Messages

(No column name)
۱ •x۸۵۷•۰۷۱۵۱۷•A۲A••

SQLCLR types به صورت پیش فرض با فرمت serialized binary value بازگشت داده می‌شوند.

```
SELECT id, TestDate.ToString() FROM TestTable;
```

در ابزارهای کوئری گرفتن ad-hoc مثل SSMS باید یک متد ToString را هم به انتهای نام ستون اضافه کنید تا مقدار نمایشی واضحی حاصل شود.

عنوان: intern pool جدول نگهداری رشته‌ها در دات‌نت

نویسنده: رحمت اله رضایی

تاریخ: ۲۲:۳۰ ۱۳۹۲/۰۲/۲۹

آدرس: www.dotnettips.info

برچسب‌ها: C#, Tips, CLR, string, intern, pool

کد زیر را در نظر بگیرید :

```
object text1 = "test";
object text2 = "test";

object num1 = 1;
object num2 = 1;

Console.WriteLine("text1 == text2 : " + (text1 == text2));
Console.WriteLine("num1 == num2 : " + (num1 == num2));
```

به نظر شما چه چیزی در خروجی نمایش داده میشود؟

هر چهار متغیر text1 و text2 و num1 و num2 از نوع object هستند. با اینکه مقدار text1 و text2 یکی و مقدار num1 و num2 هم یکی است، نتیجه text1==text2 برابر true است اما num1==num2 برابر false.

خطی که text2 تعریف شده است را تغییر میدهیم :

```
object text2 = "test".ToLower();
```

اینبار با این که باز مقدار text1 و text2 یکی و هر دو "test" است، اما نتیجه text1==text2 برابر false است. انتظار ما هم همین است. دو object ایجاد شده است و یکی نیستند. تنها در صورتی باید نتیجه == آنها true باشد که هر دو به یک object اشاره کنند.

اما چرا در کد اولی اینگونه نبود؟

دلیل این کار برمیگردد به رفتار دات‌نت نسبت به رشته‌هایی که به صورت صریح در برنامه تعریف میشوند. CLR یک جدول برای ذخیره رشته‌ها به نام **intern pool** برای برنامه میسازد. هر رشته‌ای تعریف میشود، اگر در intern pool رشته‌ای با همان مقدار وجود نداشته باشد، یک رشته جدید ایجاد و به جدول اضافه میشود، و اگر موجود باشد متغیر جدید فقط به آن اشاره میکند. واقع اگر 100 جای برنامه حتی در کلاسهای مختلف، رشته‌هایی با مقادیر یکسان وجود داشته باشند، برای همه آنها یک نمونه وجود دارد.

بنابراین text1 و text2 در کد اولی واقعا یکی هستند و یک نمونه برای آنها ایجاد شده است.

البته چند نکته در اینجا هست :

اگر text1 و text2 به صورت string تعریف شوند، نتیجه text1==text2 در هر دو حالت فوق برابر true است. چون عملگر == در کلاس string یکبار دیگر overload شده است:

```
public sealed class String : ...
{
    ...
    public static bool operator ==(string a, string b)
    {
```

```
    return string.Equals(a, b);
}
...
}
```

این که کدام یک از overloadها اجرا شوند (کلاس پایه، کلاس اصلی، ...) به نوع دو متغیر اطراف == بستگی دارد. مثلاً در کد زیر :

```
string text1 = "test";
string text2 = "test".ToLower();

Console.WriteLine("text1 == text2 (string) : " + (text1 == text2));
Console.WriteLine("text1 == text2 (object) : " + ((object)text1 == (object)text2));
```

اولین نتیجه true و دومی false است. چون در اولی عملگر == تعریف شده در کلاس string مورد استفاده قرار می‌گیرد اما در دومی عملگر == تعریف شده در کلاس object.

اگر دقت نشود این رفتار مشکلزا میشود. مثلاً حالتی را در نظر بگیرید که text1 ورودی کاربر است و text2 از بانک اطلاعاتی خوانده شده است و با اینکه مقادیر یکسان دارند نتیجه == آنها false است. اگر تعریف عملگرها در کلاس object به صورت virtual بود و در کلاس‌های دیگر override می‌شد، این تغییر نوع‌ها تاثیری نداشت. اما عملگرها به صورت static تعریف می‌شوند و امکان override شدن ندارند. به همین خاطر کلاس object متدی به اسم Equals در اختیار گذاشته که کلاس‌ها آنرا override می‌کنند و معمولاً از این متد برای سنجش برابری دو کلاس استفاده می‌شود :

```
object text1 = "test";
object text2 = "test".ToLower();

Console.WriteLine("text1 Equals text2 : " + text1.Equals(text2));
Console.WriteLine("text1 Equals text2 : " + object.Equals(text1, text2));
```

البته یادآور می‌شوم که فقط رشته‌هایی که به صورت صریح در برنامه تعریف شده‌اند، در intern pool قرار می‌گیرند و این فهرست شامل رشته‌هایی که از فایل یا بانک اطلاعاتی خوانده می‌شوند یا در برنامه تولید می‌شوند، نیست. این کار منطقی است و گرنه حافظه زیادی مصرف خواهد شد.

با استفاده از متد [string.Intern](#) می‌توان یک رشته را که در intern pool وجود ندارد، به فهرست آن افزود. اگر رشته در intern pool وجود داشته باشد، reference آنرا بر می‌گرداند در غیر اینصورت یک reference به رشته جدید به intern pool اضافه می‌کند و آنرا برمی‌گرداند.

یک مورد استفاده آن هنگام lock روی رشته‌هاست. برای مثال در کد زیر DeviceId یک رشته است که از بانک اطلاعاتی خوانده می‌شود و باعث می‌شود که چند job همزمان به یک دستگاه وصل نشوند :

```
lock (job.DeviceId)
{
    job.Execute();
}
```

اگر یک job با DeviceId برابر COM1 در حال اجرا باشد، این lock جلوی اجرای همزمان job دیگری با همین DeviceId را

نمی‌گیرد. زیرا هر چند مقدار DeviceId دو job یکی است ولی به یک نمونه اشاره نمی‌کنند.

می‌توان lock را اینگونه اصلاح کرد :

```
lock (string.Intern(job.DeviceId))
{
    job.Execute();
}
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۳۰ ۰:۳۵

ممنون. البته شرایط کد خودتون رو کامل اینجا قرار ندادید ولی [در حالت کلی توصیه میشه](#) که برای استفاده از lock یک شیء private object در سطح کلاس تعریف بشه و از اون استفاده بشه [تا حالت‌های دیگه](#) .

نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۲/۰۲/۳۱ ۹:۵۶

- البته این فقط یک مثال بود برای درک متد string.Intern .
- چگونگی شی معرفی شده به lock هم بسته به شرایط ممکن است متفاوت باشد. ممکن است یک private object در سطح همان کلاسی که lock در آن استفاده می‌شود، جوابگو باشد. اما در شرایط دیگری ممکن است اینگونه نباشد. مانند مثال فوق.