Veri Tabanı Tablosundan Entity Class Oluşturma

Veri tabanı işlemlerinin gerçekleştirildiği yazılımlarda tabloların entity C# sınıfları gerekebilmektedir. Her ne kadar Visual Studio içerisinde barındırdığı araçlar ile bunu otomatik yapıyorsa da T-SQL kodları kullanılarak da ilgili sınıf SQL Server tarafında oluşturulup kopyala – yapıştır ile projeye eklenebilir.

Aşağıdaki T-SQL kodu içerisinde ilk satırdaki ‘TableName’ yerine tırnaklar içerisine sınıf karşılığı oluşturulacak tablo yazılıp çalıştırıldığında print komutu ile oluşturulan sınıf ekrana gelecektir.

```
declare @TableName sysname = 'TableName'

declare @Result varchar(max) = 'public class ' + @TableName +
'

{'

select @Result = @Result + '

    public ' + ColumnType + NullableSign + ' ' + ColumnName +
' { get; set; }

'

from

(

    select
```

```
        replace(col.name, ' ', '_') ColumnName,
        column_id ColumnId,
        case typ.name
            when 'bigint' then 'long'
            when 'binary' then 'byte[]'
            when 'bit' then 'bool'
            when 'char' then 'string'
            when 'date' then 'DateTime'
            when 'datetime' then 'DateTime'
            when 'datetime2' then 'DateTime'
            when 'datetimeoffset' then 'DateTimeOffset'
            when 'decimal' then 'decimal'
            when 'float' then 'float'
            when 'image' then 'byte[]'
            when 'int' then 'int'
            when 'money' then 'decimal'
            when 'nchar' then 'char'
            when 'ntext' then 'string'
            when 'numeric' then 'decimal'
            when 'nvarchar' then 'string'
            when 'real' then 'double'
            when 'smalldatetime' then 'DateTime'
```

```
            when 'smallint' then 'short'

            when 'smallmoney' then 'decimal'

            when 'text' then 'string'

            when 'time' then 'TimeSpan'

            when 'timestamp' then 'DateTime'

            when 'tinyint' then 'byte'

            when 'uniqueidentifier' then 'Guid'

            when 'varbinary' then 'byte[]'

            when 'varchar' then 'string'

            else 'UNKNOWN_' + typ.name

        end ColumnType,

        case

            when col.is_nullable = 1 and typ.name in
('bigint', 'bit', 'date', 'datetime', 'datetime2',
'datetimeoffset', 'decimal', 'float', 'int', 'money',
'numeric', 'real', 'smalldatetime', 'smallint', 'smallmoney',
'time', 'tinyint', 'uniqueidentifier')

            then '?'

            else ''

        end NullableSign

    from sys.columns col

        join sys.types typ on

            col.system_type_id = typ.system_type_id AND
col.user_type_id = typ.user_type_id

    where object_id = object_id(@TableName)
```

```
) t

order by ColumnId

set @Result = @Result  + '

}'

print @Result
```

Kayıt ve Güncellemeden Önce Unique Olan Alanların Kontrolü

Programlama da altın kural bir şeyleri parça parça bilmekten çok parça parça bildiğin şeyleri bir bütün olarak nerede ve nasıl kullanacağını bilmendir. Bu yazımızda, daha önce ki yazımızda bulduğumuz indexleri bir kayıt veya güncellemede unique olan alanlardan daha önce kayıt alınmış mı diye sorgulama yapacağız. Örneğin bir kişinin TC kimlik numarası uniquetır. Dolayısıyla yeni bir kayıtta aynı kimlik numarasının daha önce kaydedilip kaydedilmediğinin sorgulanması gerekmektedir. Tabi her zaman durum sadece kimlik numarası ile sınırlı değil bunun için parametre olarak verdiğimiz tabloda bu işi yapan bir fonksiyon yazacağız:

Sorgulamayı yapan asıl metodumuz:

İşlem sırasıyla şu şekilde yapılmaktadır:
GetIndex metodu ile indexler alınıyor daha sonra alınan index kolonları üzerinde şu sorgu çalıştırılmaktadır.

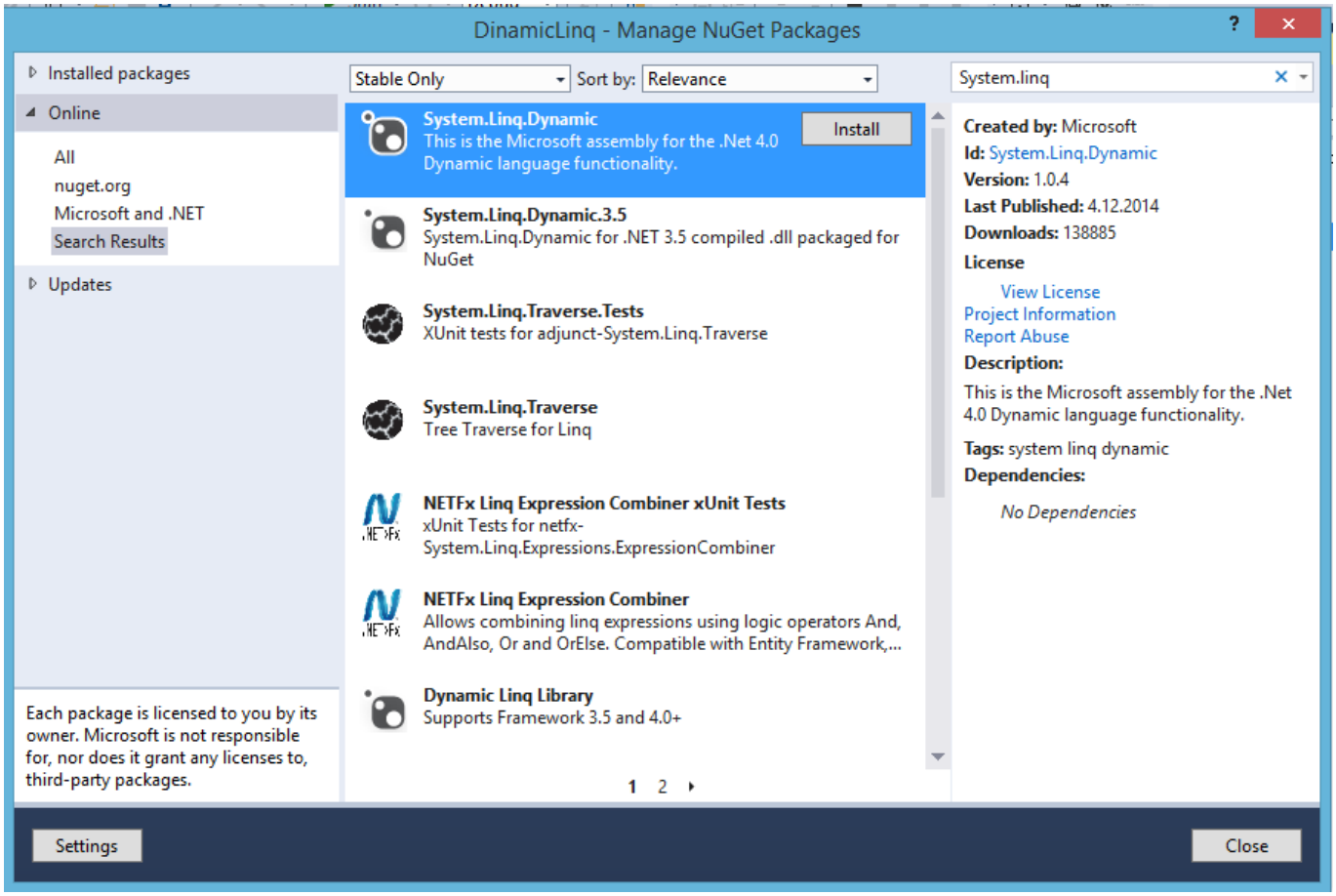
“Id değeri farklı olmak üzere bu değere sahip bir kayıt var mı?”

Not: Kod içerisinde geçen metot ve sınıflar kodun en altında ayrıca verilmiştir.

LINQ ile Dinamik Şart Oluşturmak ve Dinamik “Include” İfadesi Kullanmak

Bir linq sorgusu oluşturulurken çoğu zaman statik şart kullanımı yeterli olabiliyor ama bazı durumlarda where içerine yazılan şart ifadesi duruma göre değişebiliyor. Örneğin bir raporlama yapıldığında kullanıcının seçtiği seçeneklere göre bir şart cümlesinin oluşturulması gerekmektedir.

Linqte dinamik şart cümlesi oluşturmak için öncelikle projemize `System.Linq.Dynamic` referansını eklememiz gerekmektedir.



Referansı ekledikten sonra yapmamız gereken linq sorgumuzu şu şekilde yazmaktır:

```
var people1 = people.Where("Id > 3 OR Name = \"Sait\"");
```

Böylece where içindeki şartımızı dinamik olarak oluşturmuş olduk.

Entity Framework kullanırken zorluk çıkaran bir diğer durumda lazy loadingin pasif olması durumunda anahtarla başka tablolara bağlı olan nesnelerin Include ile sorgula eklenmesi durumudur. Burada da Include edilecek olan nesneler duruma göre değişebiliyor. Bu durumda Include edilecek nesneleri dinamik olarak sorguya eklememiz gerekmektedir. Bu şekilde çalışan örnek bir kod parçası:

```
            List<string> includeList = new
            List<string>();



LibraryContext entity = new
LibraryContext();
```

```
IQueryable<JobGroup> jobGroups = entity.JobGroups.Where(j => j.Id == id);



foreach (string
s
in includeList)

{

jobGroups.Include(s);

}



List<JobGroup> groups = jobGroups.ToList();
```