عنوان: **یافتن مقادیر نال در Entity framework**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۱:۲۱
آدرس: www.dotnettips.info
گروه‌ها: Entity framework

کلاس شخص زیر را درنظر بگیرید

```
public class Person
{
        public int Id { get; set; }
        public string Name { get; set; }
        public int? Age { get; set; }
}
```

در اینجا با توجه به اینکه Name از نوع string است، خودبخود به فیلدی نال‌پذیر نگاشت خواهد شد و همچنین Age عددی نیز در سمت کدهای ما Nullable است، بنابراین خاصیت سن هم به فیلدی نال‌پذیر نگاشت می‌شود.
اگر تمام مراحل متداول ایجاد Context را طی کنیم، به نظر شما خروجی SQL عبارت زیر چه خواهد بود؟

```
string name = null;
var list1 = ctx.Users.Where(x => x.Name == name).ToList();
```

در این عبارت، name به صورت یک متغیر ارسال شده است و نه یک مقدار ثابت (فرض کنید یک متد را تعریف کرده‌اید که name را به صورت پارامتر دریافت می‌کند).
خروجی SQL آن به نحو زیر است:

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[Age] AS [Age]
FROM [dbo].[People] AS [Extent1]
WHERE [Extent1].[Name] = @p__linq__0
-- p__linq__0 (dbtype=String, size=-1, direction=Input) = null
```

به عبارتی خروجی مورد انتظار name **is** null را تولید نکرده است و کوئری ما حداقل با SQL Server نتیجه‌ای را به همراه نخواهد داشت. در مورد Age نیز به همین صورت است.

راه حل:

برای حالت Age، روش زیر خروجی age is null را تولید می‌کند:

```
var list2 = ctx.Users.Where(x => !x.Age.HasValue).ToList();
```

و یا استفاده از object.Equals نیز مشکل را برطرف خواهد کرد:

```
int? age = null;
var list2 = ctx.Users.Where(x => object.Equals(x.Age, age)).ToList();
```

برای حالت Name رشته‌ای می‌توان از روش زیر استفاده کرد:

```
var list1 = ctx.Users.Where(x => string.IsNullOrEmpty(x.Name)).ToList();
```

و یا روش کلی‌تر زیر نیز جواب می‌دهد:

```
string name = null;
```

```
var list1 = ctx.Users.Where(x => name == null ? x.Name == null : x.Name == name).ToList();
```

کاری که در اینجا انجام شده استفاده از x.Name == null در حالت نال بودن name است. از این جهت که EF با کوئری ذیل به علت عدم استفاده از پارامتر برای معرفی مقداری نال، مشکلی ندارد :

```
var list1 = ctx.Users.Where(x => x.Name == null).ToList();
```

نظرات خوانندگان

نویسنده: نیما
تاریخ: ۱۳۹۱/۰۶/۲۸ ۹:۴۷

سلام آقای نصیری

ممنون از مطلب مفیدتون راستش من به این نکته ای که فرمودین توجه نکرده بودم. من دستوراتی را روی دیتابیس northwind اجرا کردم که اولین دستور به این صورت بود:

```
rg = ent.regions.where(x => x.regionid != null).tolist();
```

این دستور در sql server به اینصورت تبدیل میشود :

```
select
[extent1].[regionid] as [regionid],
[extent1].[regiondescription] as [regiondescription]
from [dbo].[region] as [extent1]
```

آیا اینکه اصلا در دستور sql ما چک کردن برای null نداریم به این خاطر است که ef بطور اتوماتیک چک میکند که فیلد regionid از نوع nullable هست و چنانچه نباشه اصلا شرط رو دخالت نمیده؟

من در گام بعدی این دستور را اجرا کردم :

```
rg = ent.regions.where(x => x.regiondescription == null).tolist();
```

که خروجی آن به این صورت بود:

```
select
cast(null as int) as [c1],
cast(null as varchar(1)) as [c2]
from  ( select 1 as x ) as [singlerowtable1]
where 1 = 0
```

میخواستم اگر ممکنه کمی راجع به این دستور توضیح بفرمایید آیا این دستور همون کار is null رو انجام میده یا خیر.

باز هم سپاسگزارم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۸ ۹:۵۹

خروجی SQL شما منطبق با خروجی SQL حاصل از EF نیست. روش کار را اینجا توضیح دادم که چگونه می‌شود این خروجی را دقیقا به دست آورد.
در حالت

```
var list1 = ctx.Users.Where(x => x.Name != null).ToList();
```

این خروجی حاصل می‌شود:

```
SELECT
```

```
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[Age] AS [Age]
FROM [dbo].[People] AS [Extent1]
WHERE [Extent1].[Name] IS NOT NULL
```

در حالت

```
var list2 = ctx.Users.Where(x => x.Name == null).ToList();
```

دقیقا این خروجی را خواهیم داشت:

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[Age] AS [Age]
FROM [dbo].[People] AS [Extent1]
WHERE [Extent1].[Name] IS NULL
```

نویسنده: lp
تاریخ: ۱۳۹۱/۰۶/۲۹ ۱۴:۲۱

ممنون از توضیحتون

نویسنده: debuger
تاریخ: ۱۳۹۲/۰۴/۰۱ ۱۱:۰

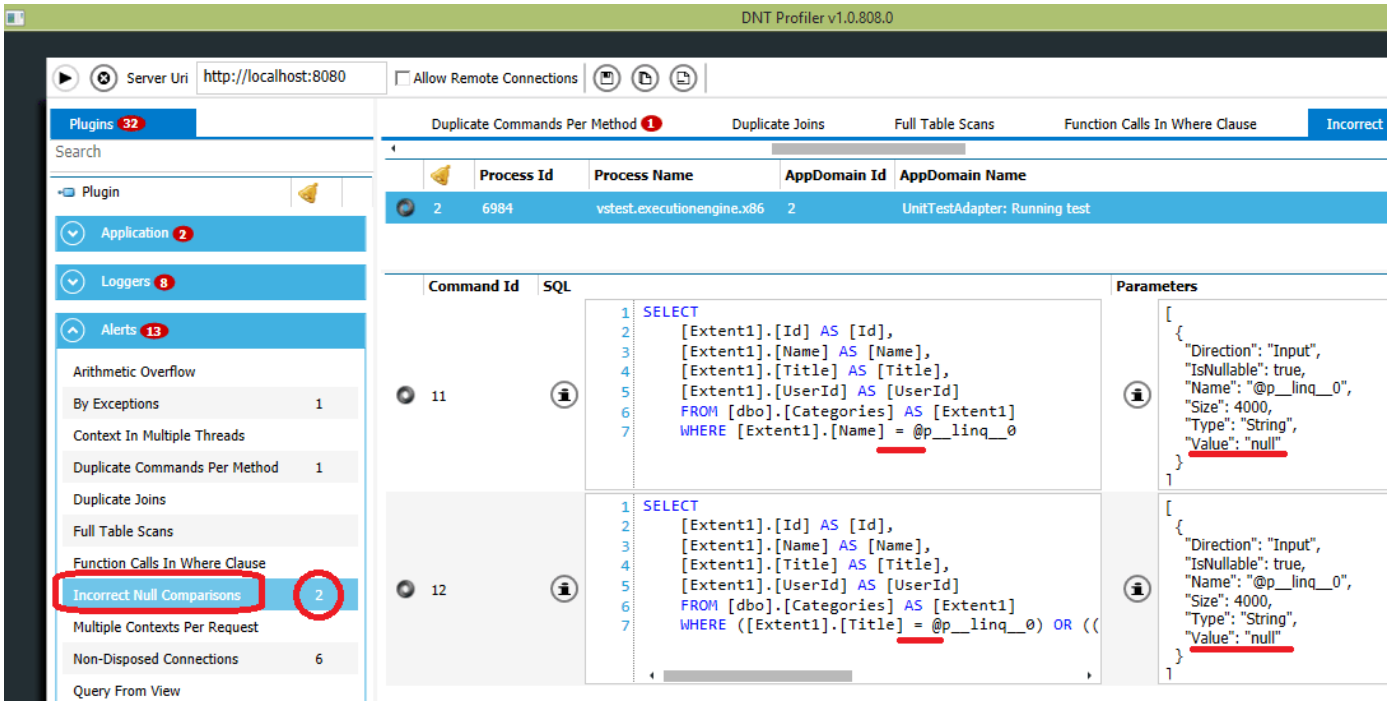
با سلام
ببخشید آیا امکان پیاده سازی تابع isnull هم توسط EF هست ؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۰۱ ۱۱:۲۵

- از عملگر ?? استفاده کنید تا با تمام بانک‌های اطلاعاتی سازگار باشد.
+ یک سری متد SQL خاص هم در EF وجود دارند که البته وابسته‌اند به بانک اطلاعاتی مورد استفاده و قابل استفاده در عبارات LINQ.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۰۱ ۱۵:۱۰

با استفاده از برنامه‌ی DNTProfiler می‌توانید مقایسه‌های با null را نیز بررسی کنید و مشکلات احتمالی موجود را بیابید:


DNT Profiler v1.0.808.0
Server Uri
http://localhost:8080
Allow Remote Connections
Plugins 32
Search
Plugin
Application 2
Loggers 8
Alerts 13
Arithmetic Overflow
By Exceptions 1
Context In Multiple Threads
Duplicate Commands Per Method 1
Duplicate Joins
Full Table Scans
Function Calls In Where Clause
Incorrect Null Comparisons 2
Multiple Contexts Per Request
Non-Disposed Connections 6
Query From View
Duplicate Commands Per Method 1
Duplicate Joins
Full Table Scans
Function Calls In Where Clause
Incorrect
Process Id
Process Name
AppDomain Id
AppDomain Name
2
6984
vstest.executionengine.x86
2
UnitTestAdapter: Running test
Command Id
SQL
Parameters
11
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[Title] AS [Title],
[Extent1].[UserId] AS [UserId]
FROM [dbo].[Categories] AS [Extent1]
WHERE [Extent1].[Name] = @p__linq__0
[
{
"Direction": "Input",
"IsNullable": true,
"Name": "@p__linq__0",
"Size": 4000,
"Type": "String",
"Value": "null"
}
]
12
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[Title] AS [Title],
[Extent1].[UserId] AS [UserId]
FROM [dbo].[Categories] AS [Extent1]
WHERE ([Extent1].[Title] = @p__linq__0) OR ((
[
{
"Direction": "Input",
"IsNullable": true,
"Name": "@p__linq__0",
"Size": 4000,
"Type": "String",
"Value": "null"
}
]