Mapping Database Scalar Functions

Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman   thedatafarm.com

Module Overview

What is a database scalar function?

Mapping a scalar function

Using the function in queries

Unit testing with certain types of scalar functions

Mapping database built-in scalar functions

Compare database function mapping to locally evaluated functions

What Is a Scalar Function?

User Defined Functions

Scalar Value result “Julie Lerman”

Table Value result

Ralph	Kramden
Alice	Kramden
Ed	Norton
Trixie	Norton

Mapping Your First Scalar Function

Getting the function into your database is **your responsibility**.

Using the Scalar Function in Queries

Scalar Functions in Your Queries

Filter

Sort

Data

Mapping Functions That Can Be Used When Testing Queries

Comparing DbFunctions to Client-side Query Evaluation

What About … ?

Client-side query processing vs. DbFunctions

Mapping RDBMS’ built-in scalar functions

Separate Database vs. Client-side Evaluation

Evaluate in SQL

Evaluate in memory

```
_context.Samurais
        .Select(s=>s.Name,Reverse= ReverseString(s.Name));
```

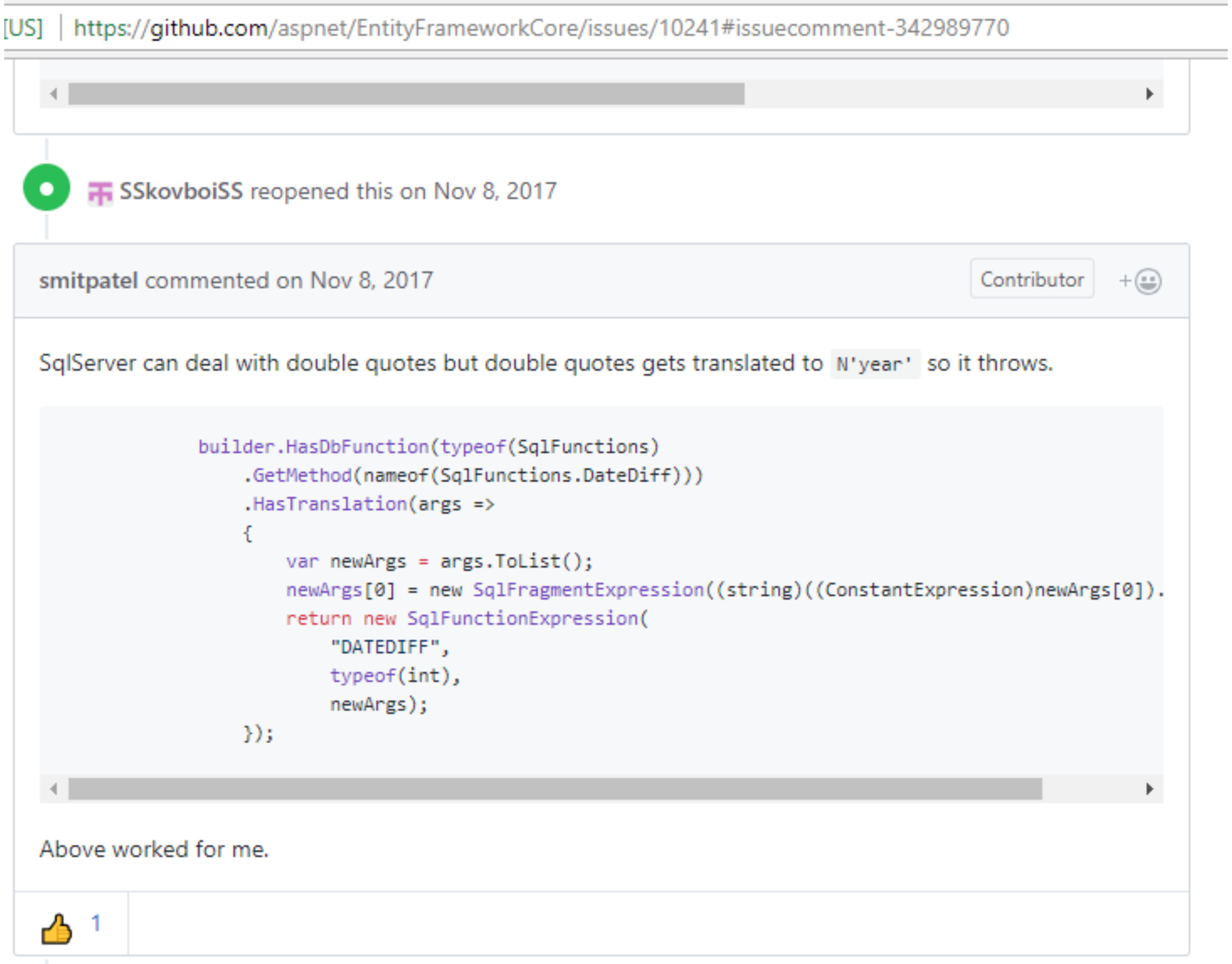
```
private static string ReverseString(string value) {
     var stringChar = value.AsEnumerable();
     return string.Concat(stringChar.Reverse());
}
```

Processing & Performance Differences

DbFunction Method	Client-side Evaluation
Executed in database	Executed in memory
Only returns scalar result	May require additional data in results
	Requires more processing in memory
	Possible performance issue

Examining the State of Mapping to Built-In Database Functions

Workaround in GitHub



[US] | https://github.com/aspnet/EntityFrameworkCore/issues/10241#issuecomment-342989770

SSkovboiSS reopened this on Nov 8, 2017

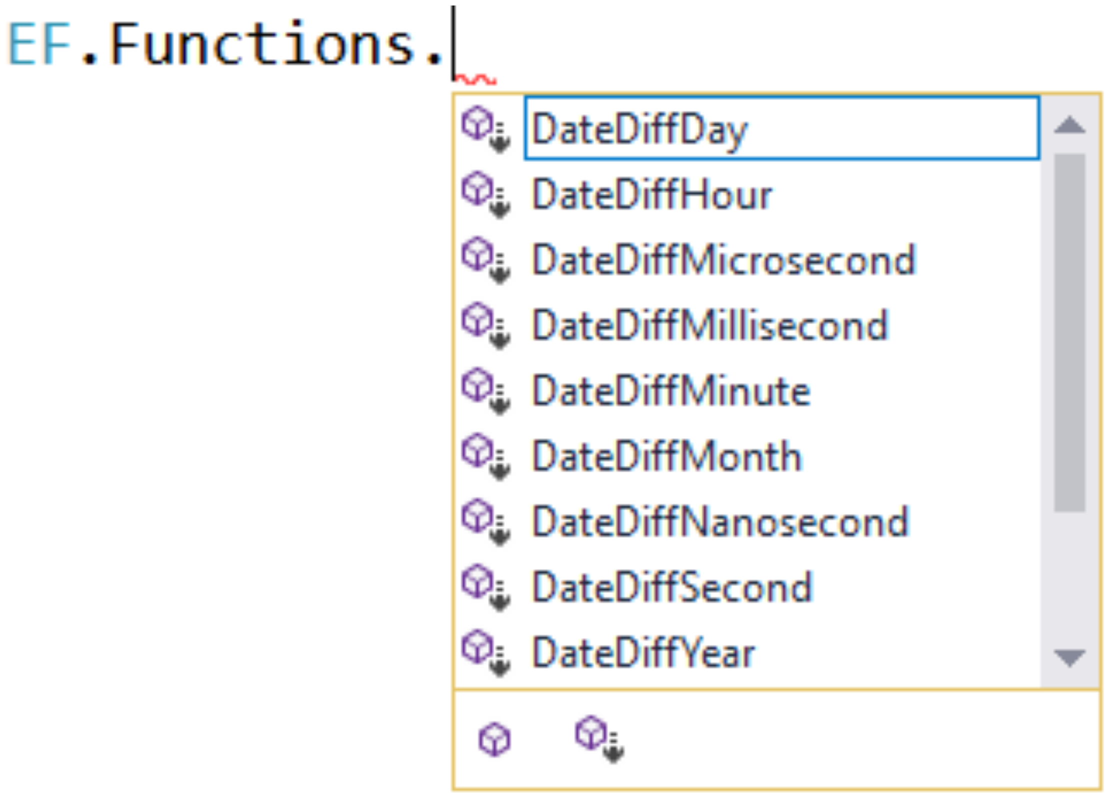
smitpatel commented on Nov 8, 2017

SqlServer can deal with double quotes but double quotes gets translated to `N'year'` so it throws.

```
            builder.HasDbFunction(typeof(SqlFunctions)
                .GetMethod(nameof(SqlFunctions.DateDiff)))
                .HasTranslation(args =>
                {
                    var newArgs = args.ToList();
                    newArgs[0] = new SqlFragmentExpression((string)((ConstantExpression)newArgs[0]).
                    return new SqlFunctionExpression(
                        "DATEDIFF",
                        typeof(int),
                        newArgs);
                });
```

Above worked for me.

EF.Functions DateDiff Methods Coming in EF Core 2.1 in SQL Server & SQL CE Providers



```
EF.Functions.
```

- DateDiffDay
- DateDiffHour
- DateDiffMicrosecond
- DateDiffMillisecond
- DateDiffMinute
- DateDiffMonth
- DateDiffNanosecond
- DateDiffSecond
- DateDiffYear

EF Core supports calling stored procedures or raw SQL if necessary

Review

Scalar UDFs are supported, but not table value functions

Specify schema name now, but in 2.1, dbo will be the default

Functions that query are not testable

Functions that calculate can be used with InMemory provider

DbFunctions will always execute in database

Many built-in functions already have support, with SQL Server/CE DateDiff coming in 2.1

Resources

Entity Framework Core on GitHub github.com/aspnet/entityframework

EF Core Roadmap bit.ly/efcoreroadmap

EF Core Documentation docs.microsoft.com/ef

Map SQLite functions: bricelam.net/2017/08/22/sqlite-efcore-udf-all-the-things.html

Watch progress of TVFs github.com/aspnet/EntityFrameworkCore/pull/11129

DateDiff workaround:
github.com/aspnet/EntityFrameworkCore/issues/10241#issuecomment-342989770

Mapping Database Scalar Functions

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com