

(T42)討論 DynamicTableName 和 SqlInjection 和 QuoteNameFunction

0. Summary

1. Create Sample Data
 2. Dynamic table Name : Dynamic SQL with sql injection issue
 3. Dynamic table Name : Dynamic SQL and fix sql injection issue
 4. Dynamic table Name with schema name dbo
 5. QuoteName()
 6. Clean up
-

0. Summary

1.

--QUOTENAME(str1, delimiterStr) V.S. PARSENAME(str1, ObjectPeiceInt)

1.1.

--QUOTENAME(str1, delimiterStr)

returns str1 which was wrapped by delimiterStr.

The delimiter can only be a left or right bracket ([]),

a single quotation mark ('), or

a double quotation mark (").

The default for the second parameter is [].

Any other delimiter will return NULL.

1.2.

When right bracket(]) is between str1. It will double it.

--PRINT QUOTENAME('Gamer] Two')

returns [Gamer]] Two]

When left bracket([) is between str1. It will NOT double it.

--PRINT QUOTENAME('Gamer [Two')

returns [Gamer [Two]

1.3.

PARSENAME() can undo the QUOTENAME()

--PARSENAME(str1, ObjectPeiceInt)

str1 is the string which applied QUOTENAME()

ObjectPeiceInt indicate what type of object peice.

PARSENAME(str1, ObjectPeiceInt) will undo QUOTENAME() in str1.

When ObjectPeiceInt = 1, it means Object name

When ObjectPeiceInt = 2, it means Schema name

When ObjectPeiceInt = 3, it means Database name

When ObjectPeiceInt = 4, it means Server name

2.

Dynamic SQL and sql injection issue

2.1.

Dynamic SQL with sql injection issue

DECLARE @tableName NVARCHAR(50) = 'Gamer Two';

DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM [' + @tableName + ']';

EXECUTE sp_executesql @sql;

2.2.

Dynamic SQL and fix sql injection issue

```
--DECLARE @tableName NVARCHAR(50) = 'Gamer Two';
--DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@tableName)
--EXECUTE sp_executesql @sql;
```

2.3.

Dynamic SQL with schema name dbo and fix sql injection issue

```
--DECLARE @schemaName NVARCHAR(50) = 'dbo';
--DECLARE @tableName NVARCHAR(50) = 'Gamer Two';
--DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@schemaName) + '.' +
QUOTENAME(@tableName)
--EXECUTE sp_executesql @sql;
```

=====

1. Create Sample Data

--T042_01_Create Sample Data

```
IF ( EXISTS ( SELECT      *
               FROM        INFORMATION_SCHEMA.TABLES
               WHERE       TABLE_NAME = 'Gamer Two' ) )
BEGIN
    TRUNCATE TABLE dbo.[Gamer Two];
    DROP TABLE [Gamer Two];
END;
```

GO -- Run the previous command and begins new batch

```
CREATE TABLE [Gamer Two]
```

```
(
    Id INT IDENTITY(1, 1)
        PRIMARY KEY ,
    FirstName NVARCHAR(50) ,
    LastName NVARCHAR(50) ,
    Gender NVARCHAR(50) ,
    GameScore INT
);
```

GO -- Run the previous command and begins new batch

```
INSERT INTO [Gamer Two]
VALUES ( 'AFirst01', 'XLast01', 'Female', 3500 );
INSERT INTO [Gamer Two]
VALUES ( 'AFirst02', 'YLast02', 'Female', 4000 );
INSERT INTO [Gamer Two]
VALUES ( 'BFirst03', 'YLast03', 'Male', 4600 );
INSERT INTO [Gamer Two]
VALUES ( 'BFirst04', 'YLast04', 'Male', 5400 );
INSERT INTO [Gamer Two]
VALUES ( 'BFirst05', 'ZLast05', 'Female', 2000 );
INSERT INTO [Gamer Two]
VALUES ( 'CFirst06', 'YLast06', 'Male', 4320 );
INSERT INTO [Gamer Two]
VALUES ( 'CFirst07', 'YLast07', 'Male', 4400 );
GO -- Run the previous command and begins new batch
```


```
SELECT *
```

```
FROM [Gamer Two];
GO -- Run the previous command and begins new batch
```

	Id	FirstName	LastName	Gender	GameScore
1	1	AFirst01	XLast01	Female	3500
2	2	AFirst02	YLast02	Female	4000
3	3	BFirst03	YLast03	Male	4600
4	4	BFirst04	YLast04	Male	5400
5	5	BFirst05	ZLast05	Female	2000
6	6	CFirst06	YLast06	Male	4320
7	7	CFirst07	YLast07	Male	4400

2. Dynamic table Name : Dynamic SQL with sql injection issue

```
-----
--T042_02_Dynamic table Name : Dynamic SQL with sql injection issue
-----
--T042_02_01
DECLARE @tableName NVARCHAR(50) = 'Gamer Two';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + @tableName;
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
--Error
```

 Messages

Msg 208, Level 16, State 1, Line 98
Invalid object name 'Gamer'.

```
-----
--T042_02_02
--Dynamic SQL with sql injection issue
DECLARE @tableName NVARCHAR(50) = 'Gamer Two';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM [' + @tableName + ']';
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
--It works, but it has sql injection issue.
```

	Id	FirstName	LastName	Gender	GameScore
1	1	AFirst01	XLast01	Female	3500
2	2	AFirst02	YLast02	Female	4000
3	3	BFirst03	YLast03	Male	4600
4	4	BFirst04	YLast04	Male	5400
5	5	BFirst05	ZLast05	Female	2000
6	6	CFirst06	YLast06	Male	4320
7	7	CFirst07	YLast07	Male	4400

```

=====
--T042_02_03
--Create Sample data
IF ( EXISTS ( SELECT      *
                FROM        INFORMATION_SCHEMA.TABLES
                WHERE        TABLE_NAME = 'Table1' ) )

    BEGIN
        TRUNCATE TABLE dbo.Table1;
        DROP TABLE Table1;

    END;

GO -- Run the previous command and begins new batch
CREATE TABLE Table1
(
    Id INT IDENTITY(1, 1)
        PRIMARY KEY ,
    [Name] NVARCHAR(50)
);

GO -- Run the previous command and begins new batch
=====
--T042_02_04
DECLARE @tableName NVARCHAR(50) = 'Gamer Two']; DROP TABLE Table1--';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM [' + @tableName + ']';
EXECUTE sp_executesql @sql;

GO -- Run the previous command and begins new batch
/*
**SQL injection issue
The Table1 has been dropped
*/

```

3. Dynamic table Name : Dynamic SQL and fix sql injection issue

```

=====
--T042_03_Dynamic table Name : Dynamic SQL and fix sql injection issue
=====
--T042_03_01
--Create Sample data
IF ( EXISTS ( SELECT      *
                FROM        INFORMATION_SCHEMA.TABLES
                WHERE        TABLE_NAME = 'Table1' ) )

    BEGIN
        TRUNCATE TABLE dbo.Table1;
        DROP TABLE Table1;

    END;

GO -- Run the previous command and begins new batch
CREATE TABLE Table1
(
    Id INT IDENTITY(1, 1)
        PRIMARY KEY ,
    [Name] NVARCHAR(50)
);

GO -- Run the previous command and begins new batch
=====

```

```

--T042_03_02
DECLARE @tableName NVARCHAR(50) = 'Gamer Two';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@tableName)
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
=====
--T042_03_03
DECLARE @tableName NVARCHAR(50) = 'Gamer Two]; DROP TABLE Table1--';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@tableName)
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
/*
The Table1 has NOT been dropped
output as following
--Msg 208, Level 16, State 1, Line 139
--Invalid object name 'Gamer Two]; DROP TABLE Table1--'.
*/
=====

```

4. Dynamic table Name with schema name dbo

```

=====
--T042_04_Dynamic table Name with schema name dbo
=====
--T042_04_01
--Create Sample data
IF ( EXISTS ( SELECT *
              FROM INFORMATION_SCHEMA.TABLES
              WHERE TABLE_NAME = 'Table1' ) )
BEGIN
    TRUNCATE TABLE dbo.Table1;
    DROP TABLE Table1;
END;
GO -- Run the previous command and begins new batch
CREATE TABLE Table1
(
    Id INT IDENTITY(1, 1)
        PRIMARY KEY ,
    [Name] NVARCHAR(50)
);
GO -- Run the previous command and begins new batch
=====
--T042_04_02
DECLARE @tableName NVARCHAR(50) = 'dbo.Gamer Two';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@tableName)
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
--Error
=====
--T042_04_03
DECLARE @tableName NVARCHAR(50) = '[dbo].[Gamer Two]';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + @tableName
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
--It works, but it has SQL injection issue.
=====
--T042_04_04

```

```

DECLARE @tableName NVARCHAR(50) = '[dbo].[Gamer Two]; DROP TABLE Table1--';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + @tableName
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
/*
It works, but it has SQL injection issue.
The Table1 has been dropped.
*/
=====
--T042_04_05
--Create Sample data
IF ( EXISTS ( SELECT      *
               FROM        INFORMATION_SCHEMA.TABLES
               WHERE        TABLE_NAME = 'Table1' ) )
    BEGIN
        TRUNCATE TABLE dbo.Table1;
        DROP TABLE Table1;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Table1
(
    Id INT IDENTITY(1, 1)
        PRIMARY KEY ,
    [Name] NVARCHAR(50)
);
GO -- Run the previous command and begins new batch
=====
--T042_04_06
DECLARE @schemaName NVARCHAR(50) = 'dbo';
DECLARE @tableName NVARCHAR(50) = 'Gamer Two';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@schemaName) + '.' + QUOTENAME(@tableName)
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
--Dynamic SQL with schema name dbo and fix sql injection issue
=====
--T042_04_07
DECLARE @schemaName NVARCHAR(50) = 'dbo';
DECLARE @tableName NVARCHAR(50) = 'Gamer Two; DROP TABLE Table1--';
DECLARE @sql NVARCHAR(MAX) = 'SELECT * FROM ' + QUOTENAME(@schemaName) + '.' + QUOTENAME(@tableName)
EXECUTE sp_executesql @sql;
GO -- Run the previous command and begins new batch
/*
The Table1 has NOT been dropped
output as following
--Msg 208, Level 16, State 1, Line 232
--Invalid object name 'dbo.Gamer Two; DROP TABLE Table1--'.
*/
=====

```

5. QuoteName()

```

=====
--T042_05_QuoteName()
=====
/*
1.
--QUOTENAME(str1, delimiterStr) V.S. PARSENAME(str1, ObjectPeiceInt)
1.1.
--QUOTENAME(str1, delimiterStr)

```

returns str1 which was wrapped by delimiterStr.
The delimiter can only be a left or right bracket ([]),
a single quotation mark ('), or
a double quotation mark (").
The default for the second parameter is [].
Any other delimiter will return NULL.

1.2.

When right bracket(]) is between str1. It will double it.

```
--PRINT QUOTENAME('Gamer ] Two')
```

returns [Gamer]] Two]

When left bracket([) is between str1. It will NOT double it.

```
--PRINT QUOTENAME('Gamer [ Two')
```

returns [Gamer [Two]

1.3.

PARSENAME() can undo the QUOTENAME()

```
--PARSENAME(str1, ObjectPeiceInt)
```

str1 is the string which applied QUOTENAME()

ObjectPeiceInt indicate what type of object peice.

PARSENAME(str1, ObjectPeiceInt) will undo QUOTENAME() in str1.

When ObjectPeiceInt = 1, it means Object name

When ObjectPeiceInt = 2, it means Schema name

When ObjectPeiceInt = 3, it means Database name

When ObjectPeiceInt = 4, it means Server name

*/

```
--=====
```

```
--T042_05_01
```

```
--QuoteName()
```

```
PRINT QUOTENAME('Gamer Two', '')
```

```
--"Gamer Two"
```

```
PRINT QUOTENAME('Gamer Two', ''')
```

```
--'Gamer Two'
```

```
PRINT QUOTENAME('Gamer Two')
```

```
PRINT QUOTENAME('Gamer Two', '[')
```

```
PRINT QUOTENAME('Gamer Twos', ']')
```

```
--[Gamer Two]
```

```
SELECT QUOTENAME('Gamer Two', '*')
```

```
--NULL
```

```
PRINT QUOTENAME('Gamer ] Two')
```

```
--[Gamer ]] Two]
```

```
PRINT QUOTENAME('Gamer [ Two')
```

```
--[Gamer [ Two]
```

```
--=====
```

```
--T042_05_02
```

```
--PARSENAME() can undo the QUOTENAME()
```

```
Declare @tableName nvarchar(50)
```

```
Set @tableName = 'Gamer ] Two'
```

```
Set @tableName = QUOTENAME(@tableName)
```

```
Print @tableName
```

```
Set @tableName = PARSENAME(@tableName,1)
```

```
Print @tableName
```

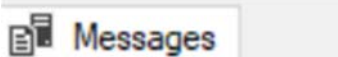
```
/*
```

output as following

```
--[Gamer ]] Two]
```

```
--Gamer ] Two
```

```
*/
```



[Gamer]] Two]

Gamer] Two

```
=====
```

6. Clean up

```
--=====
--T042_06_Clean up
--=====

IF ( EXISTS ( SELECT      *
               FROM        INFORMATION_SCHEMA.TABLES
               WHERE        TABLE_NAME = 'Gamer Two' ) )

BEGIN
    TRUNCATE TABLE dbo.[Gamer Two];
    DROP TABLE [Gamer Two];
END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT      *
               FROM        INFORMATION_SCHEMA.TABLES
               WHERE        TABLE_NAME = 'Table1' ) )

BEGIN
    TRUNCATE TABLE dbo.Table1;
    DROP TABLE Table1;
END;

GO -- Run the previous command and begins new batch
```