Passing multiple SqlParameter to method

3

1

In my main form, I have implemented this code..

```
void SampleMethod(string name, string lastName, string database)
{
        SqlParameter sqlParam = new SqlParameter();
        sqlParam.ParameterName = "@name";
        sqlParam.Value = name;
        sqlParam.SqlDbType = SqlDbType.NVarChar;

        SqlParameter sqlParam1 = new SqlParameter();
        sqlParam1.ParameterName = "@lastName";
        sqlParam1.Value = lastName;
        sqlParam1.SqlDbType = SqlDbType.NVarChar;

        SqlParameter sqlParam2 = new SqlParameter();
        sqlParam2.ParameterName = "@database";
        sqlParam2.Value = database;
        sqlParam2.SqlDbType = SqlDbType.NVarChar;

        SampleClass sampleClass = new SampleClass(new DBConn(@serverName, tableName,
userName, password));
        sampleClass.executeStoredProc(dataGridView1, "sp_sampleStoredProc", sqlParam,
sqlParam1, sqlParam2);
}
```

And in my `SampleClass`, I have this kind of method.

```
public DataGridView executeStoredProc(DataGridView dtgrdView, string storedProcName,
params SqlParameter[] parameters)
{
        try
        {
            DataTable dt = new DataTable();
            sqlDA = new SqlDataAdapter(storedProcName, sqlconn);
```

```
                foreach (var p in parameters)
                    sqlDA.SelectCommand.Parameters.Add(p);
            }

            sqlDA.Fill(dt);
            dtgrdView.DataSource = dt;
            sqlconn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            sqlconn.Close();
        }

        return dtgrdView;
}
```

What I am trying to do is avoid multiple

```
SqlParameter sqlParam = new SqlParameter()
```

in my code, I have tried so many solutions for this problem but I didn't get the right answer. I have also tried to research about this but I still couldn't get the right answer.

Please don't mind my naming convention and other codes as I intentionally change many of them :) Thanks.

c# methods sqlparameters

edited Jan 17 '17 at 8:27
marc_s
598k 135 1148 1284

asked Jan 17 '17 at 8:15
Syntax Error
37 10

Why do you want to avoid it? What are you trying to achieve exactly? – Sami Kuhmonen Jan 17 '17 at 8:17

You can create parameters with `Parameters.Add` yet your code is *explicitly* written to pass individual parameters to `executeStoredProc` . That's a bit self-contradictory. Why do you want to avoid multiple parameters at all? – Panagiotis Kanavos Jan 17 '17 at 8:18

here stackoverflow.com/questions/23320701/… – Steve Jan 17 '17 at 8:23

1 @mybirthname you don't seem to realize that your code allows for escaped exceptions. There is a reason `using` is used instead of `try/finally` . You can't use it though since you introduce the `GetSqlConnection` method that actually *creates* a connection and assigns it to a command (two independent tasks). You also don't seem to be aware that `AddWithValue` is *not* safe - it can easily infer the wrong type or size – Panagiotis Kanavos Jan 17 '17 at 8:24

4 Answers

2

As an alternative to your solution, try to use already existing one instead, using Dapper (https://github.com/StackExchange/dapper-dot-net).

You still need to use multiple parameters if your stored procedure or query needs it, but this is nicely abstracted for you and this will definatelly reduce the amount of code.

```
void SampleMethod(string name, string lastName, string database)
{
    using(var connection = new SqlConnection(MY_CONNECTION_STRING))
    {
        var resultListOfRows = connection.Query<ReturnObject>(MY_STORED_PROCEDURE, new {
            name = name,
            lastName = lastName,
            database = database}, commandType: System.Data.CommandType.StoredProcedure);
    }
}
```

edited Jan 17 '17 at 9:00

answered Jan 17 '17 at 8:29
vidriduch
3,804 5 36 51

1 It should be `Query` not `Execute` – Panagiotis Kanavos Jan 17 '17 at 8:56

@Panagiotis Kanavos, fixed ... thanks – vidriduch Jan 17 '17 at 9:01

Join Stack Overflow to learn, share knowledge, and build your career.

Sign up with email Sign up with Google Sign up with Facebook

1

```
using(var con=new SqlConnection(myConnectionString))
{
    con.Open();
    var result= connection.Query<ResultClass>("sp_MySproc",
                         new { Name= name, LastName= lastName,Database=database},
                         commandType: CommandType.StoredProcedure);
    return result;
}
```

Even when using raw ADO.NET, you can create a SqlParameter in one line by using the appropriate constructor . For example, you can create a new `nvarchar(n)` parameter with:

```
var myParam=new SqlParameter("@name",SqlDbType.NVarchar,20);
```

or

```
var myParam=new SqlParameter("@name",SqlDbType.NVarchar,20){Value = name};
```

A better idea though is to create the SqlCommand object just once and reuse it. Once you have an initialized SqlCommand object, you can simply set a new connection to it and change the parameter values, eg:

```
public void Init()
{
    _loadCustomers = new SqlCommand(...);
    _loadCustomers.Parameters.Add("@name",SqlDbType.NVarChar,20);
    ...
}

//In another method :
using(var con=new SqlConnection(myConnectionString)
{
    _loadCustomers.Connection=con;
    _loadCustomers.Parameters["@name"].Value = myNameParam;
    con.Open();
    using(var reader=_load.ExecuteReader())
    {
    //...
```

You can do the same thing with a SqlDataAdapter, in fact that's how Windows Forms and Data Adapters are meant to be used since .NET 1.0 .

Instead of creating a new one each time you want to fill your grid, create a single one and reuse it by setting the connection and parameters before execution. You can use the SqlDataAdapter(SqlCommand) constructor to make things a bit cleaner:

```
public void Init()
{
    _loadCustomers = new SqlCommand(...);
    _loadCustomers.Parameters.Add("@name",SqlDbType.NVarChar,20);
    ....
    _myGridAdapter = new SqlDataAdapter(_loadCustomers);
    ...
}
```

And call it like this:

```
using(var con=new SqlConnection(myConnectionString))
{
    _myGridAdapter.SelectCommand.Connection=con;
    _myGridAdapter.SelectCommand.Parameters["@name"].Value =....;
    con.Open();

    var dt = new DataTable();
    _myGridAdapter.Fill(dt);
    dtgrdView.DataSource = dt;
    return dtgrdView;
}
```

edited Jan 17 '17 at 8:53

answered Jan 17 '17 at 8:46

Panagiotis Kanavos
61.9k 5 87 120

Separate your Database logic at one place(put sqladapter, sqlcommand etc at one place), Then encapsulate parameters within your command like mentioned below and you don't need to declare sqlparameter separately, add it inside parameters list.

```
    public DataTable ProdTypeSelectAll(string cultureCode)
    {
        SqlCommand cmdToExecute = new SqlCommand();
        cmdToExecute.CommandText = "dbo.[pk_ProdType_SelectAll]";
        cmdToExecute.CommandType = CommandType.StoredProcedure;
        DataTable toReturn = new DataTable("ProdType");
        SqlDataAdapter adapter = new SqlDataAdapter(cmdToExecute);
        cmdToExecute.Connection = _mainConnection;
        cmdToExecute.Parameters.Add(new SqlParameter("@CultureName", cultureCode));
        _mainConnection.Open();
        adapter.Fill(toReturn);
        return toReturn;
}
```

edited Jan 18 '17 at 4:19 answered Jan 17 '17 at 8:31

M. Adeel Khalid
1,722 2 18 23

Your code *does* create new parameters. – Panagiotis Kanavos Jan 17 '17 at 8:36

Yes it does but in a different way. – M. Adeel Khalid Jan 17 '17 at 8:43

0

You may be able to use the SqlParameter Constructor (String, Object). Replace:

```
sampleClass.executeStoredProc(dataGridView1,
        "sp_sampleStoredProc",
        sqlParam,
        sqlParam1,
        sqlParam2);
```

With:

```
sampleClass.executeStoredProc(dataGridView1,
        "sp_sampleStoredProc",
```

answered Jan 17 '17 at 8:35

Serge
3,140 2 11 24