

## Debugging Stored Procedures in Visual Studio 2005

Written by [Guest Authors](#) on **25 June 2007** | [5 Comments](#)

Tagged with [Stored Procedures](#), [Tools](#), [.NET](#)

This article walks through the basics of debugging stored procedures using Visual Studio 2005. It covers breakpoints, watches and stepping through code. It was written by [Chris Rock](#) and includes his very unique sense of humor.

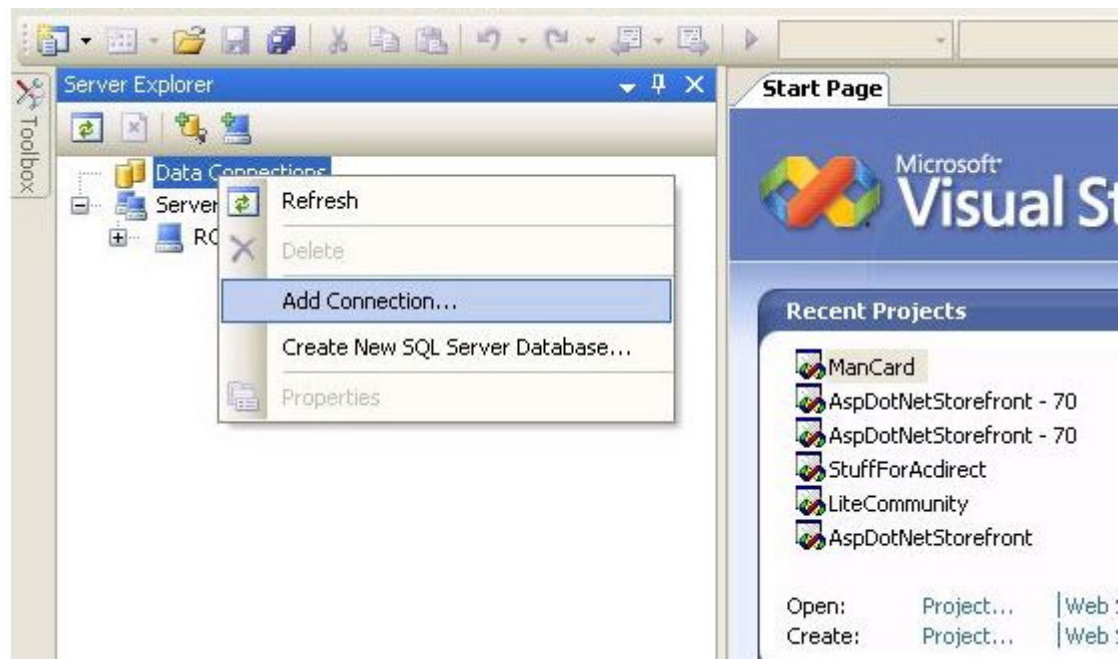
One of my favorites activities while working is debugging. A little weird yes, but I've always had an affinity towards taking things apart and putting them back together. I haven't always had success. As a kid I destroyed many toys and pieces of furniture (office chairs) just to find out how they work. I am a curious person by nature and like to figure out how something works OR why something isn't working. Alright, enough about my lame personality.

Before I was enlightened to this world of debugging stored procedures I used the archaic "print" statements to debug stored procedures. When I found out that I didn't have to do that anymore I cried tears of joy. Ok... I didn't actually "cry" but I did mutter something like "That's cool..." (I told you I was lame!) That was when Query Analyzer added the ability to debug stored procedures. Debugging stored procedures has come a long way since the early versions and I'd like to show how to accomplish this very cool feature in Visual Studio 2005.

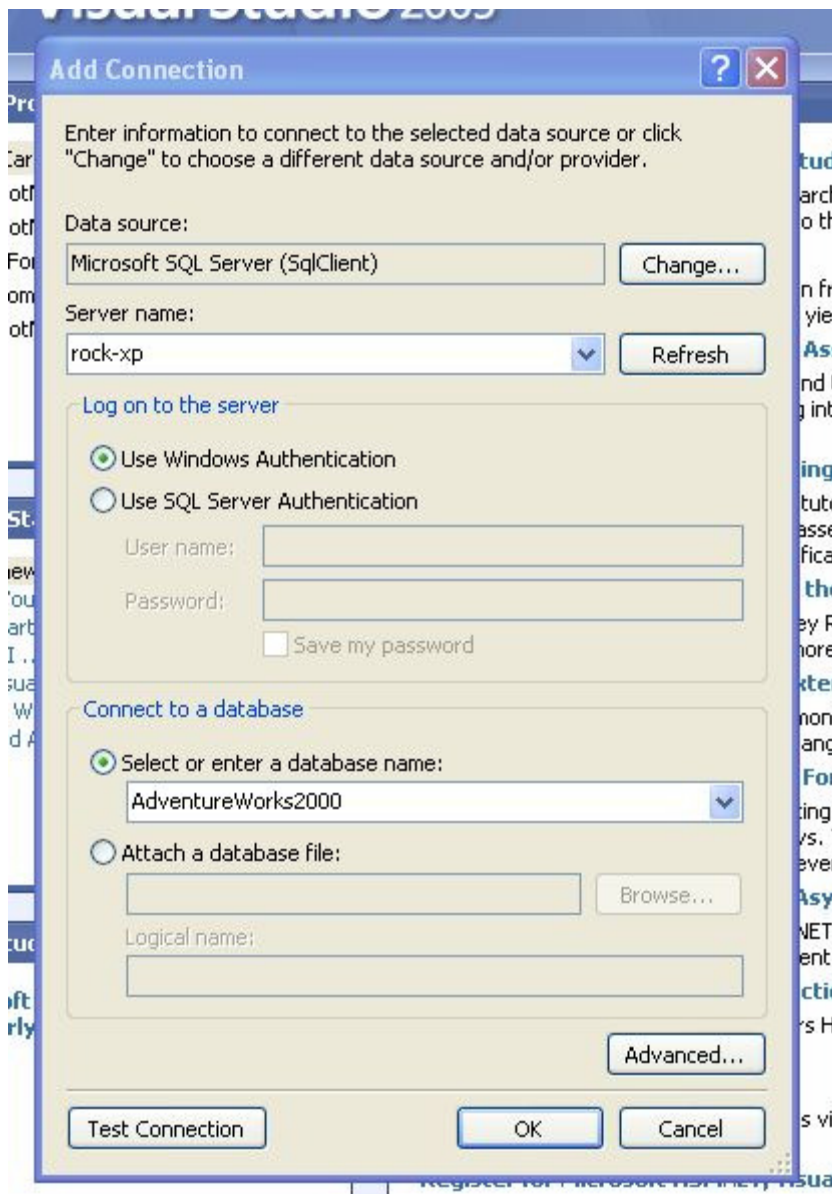
I suggest that you take a look at [http://msdn2.microsoft.com/en-us/library/s4sszxst\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/s4sszxst(VS.80).aspx) to verify that you meet the minimum requirements to debug on your SQL server. If you have control of your SQL server box (or SQL is running locally) then you're in good shape and can make any necessary changes. If you're not in control I suggest you buy something for your favorite DBA from thinkgeek.com to get in their good graces and have them assist you in anyway they can! (Note: This probably won't work)

Without further ado let's get started.

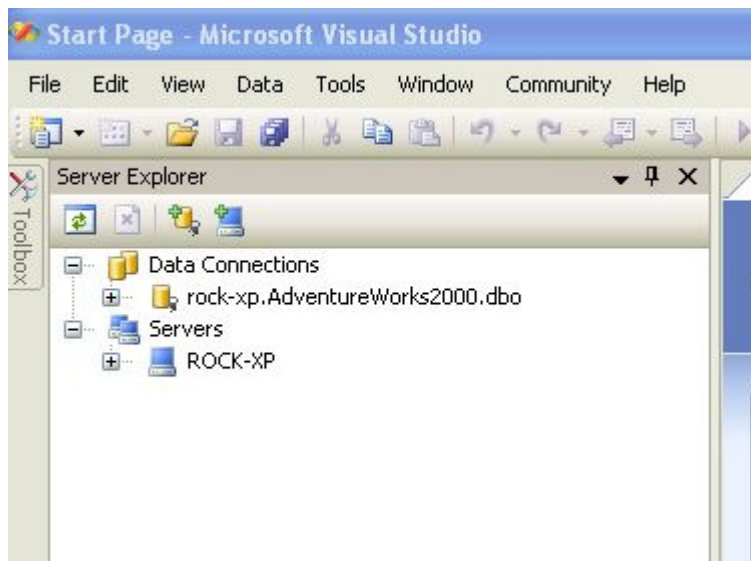
The first thing you want to do is add a data connection in the Visual Studio 2005 Servers windows. Right click on "Data Connections" and click on "Add New Connection":



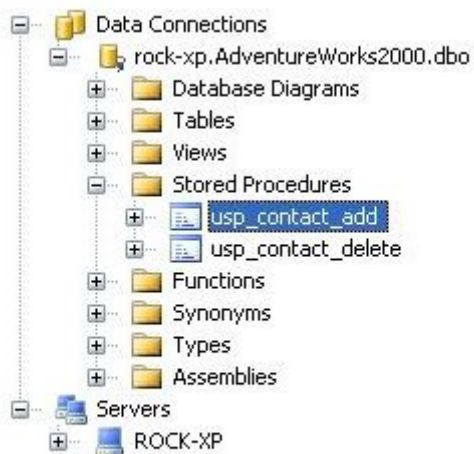
Enter the required connection info in the "Add Connection" dialog and click "Test Connection" to test the connection. If that succeeds, click OK:



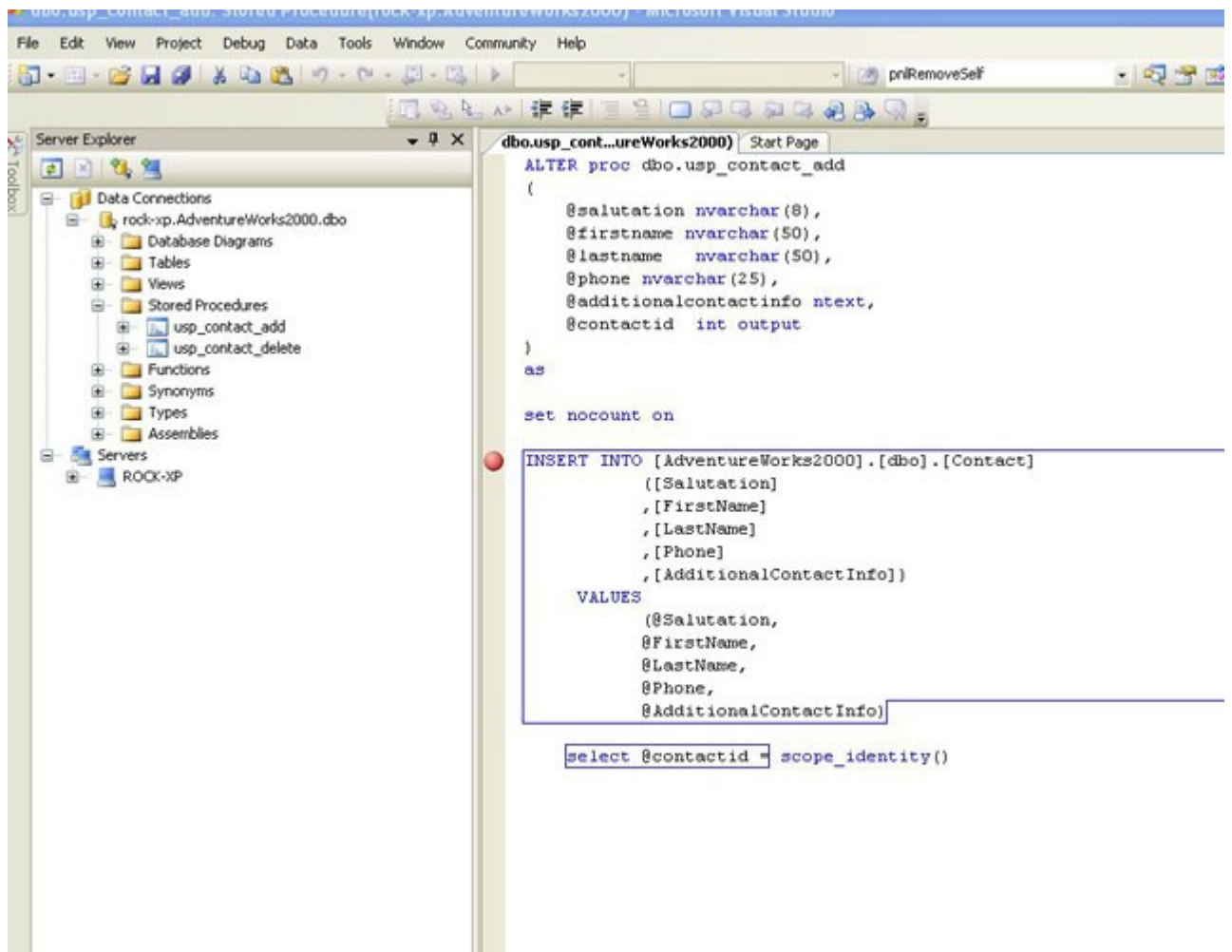
After you add the data connection you will be able to see the connection in the Data Connections tree:



We're going to debug "usp\_contact\_add" in this example (any proc will do, it really doesn't matter what proc YOU use). Drill down until you can see the proc you're debugging under the "Stored Procedures" folder:

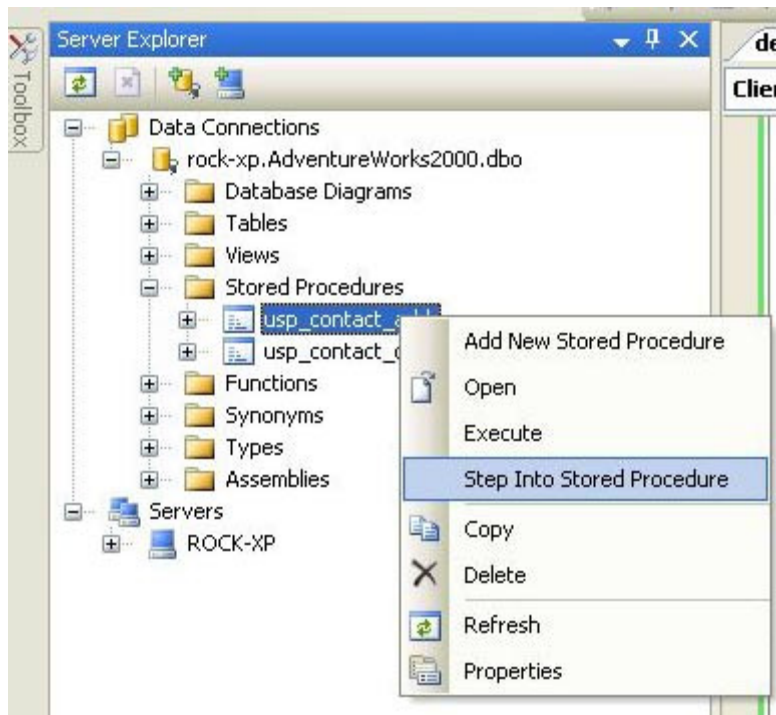


Right click on the proc and click "Open". You will see the proc code in the code window:

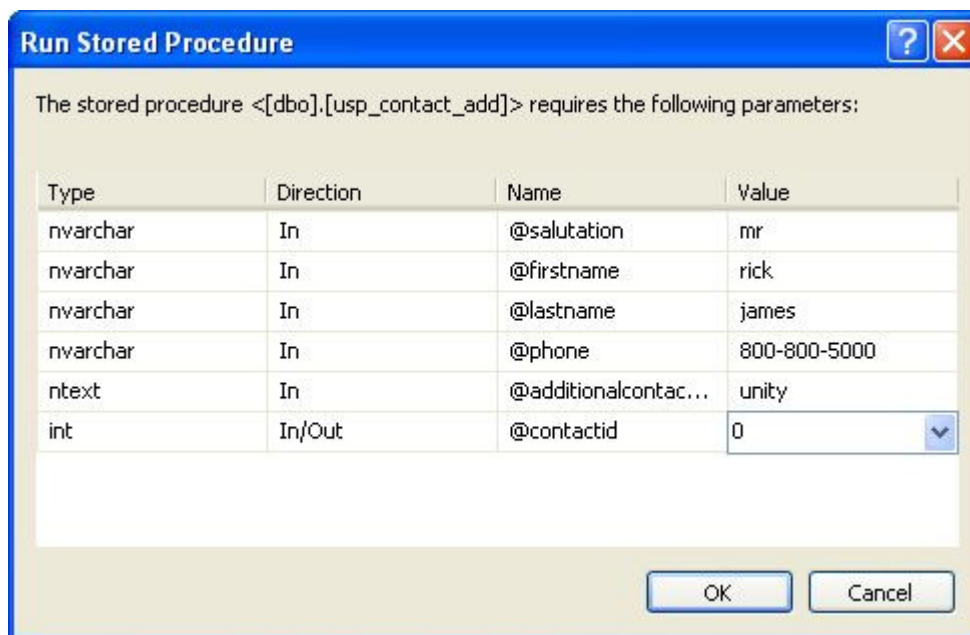


## "Now we debug"

Now that we've figured out how to open the proc we're going to get into actually debugging it. Right click on the proc and click "Step Into Stored Procedure":



A window with all of the parameters for the proc will pop up:



Notice it says "The stored procedure ... REQUIRES the following parameters:" That means you are REQUIRED to fill in the values. Otherwise the debugger will quit right away. That said; click the OK button to step into the procedure. The debugger will automatically stop execution on the first line of code in the stored procedure. When the debugger has stopped execution you'll see a yellow arrow to the left of the line of code:

```
ALTER proc dbo.usp_contact_add
(
    @salutation nvarchar(8),
    @firstname nvarchar(50),
    @lastname nvarchar(50),
    @phone nvarchar(25),
    @additionalcontactinfo ntext,
    @contactid int output
)
as
set nocount on

INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    ,[FirstName]
    ,[LastName]
    ,[Phone]
    ,[AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)

select @contactid = scope_identity()
```

Depending on how you have your Visual Studio settings you will click either F8 (VB) or F11(C#) to go to the next line of code. The debugger will execute the previous line of code and move to the next line:

```

ALTER proc dbo.usp_contact_add
(
    @salutation nvarchar(8),
    @firstname nvarchar(50),
    @lastname nvarchar(50),
    @phone nvarchar(25),
    @additionalcontactinfo ntext,
    @contactid int output
)
as

set nocount on

```



```

INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    ,[FirstName]
    ,[LastName]
    ,[Phone]
    ,[AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)

```

```

select @contactid = scope_identity()

```

Now "nocount" is "set" to "on". Click F8 or F11 again and the insert statement (or whatever line of code the yellow arrow is pointing) will be executed.

## Ok... So can we do anything except watch the debugger work?

Yes, in fact there are several things you can do but I'm going to concentrate on what I use most in VS2005 while debugging stored procedures.

While you're debugging you can view / edit the value of the variables in scope using a few different methods. The first way to view/edit a variable is simply hovering your mouse over the variable:



```
set nocount on
```

```
INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    , [FirstName]
    , [LastName]
    , [Phone]
    , [AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)
```

```
select @contactid = scope_identity()
```

```
set nocount on
```

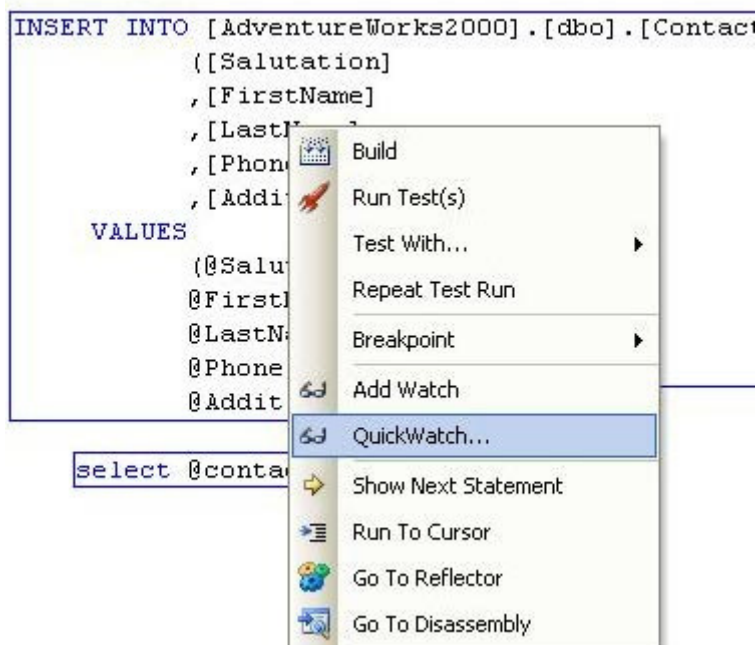
```
INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    , [FirstName]
    , [LastName]
    , [Phone]
    , [AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)
```

```
select @contactid = scope_identity()
```

Pretty cool [eh](#)? The Fonz thinks so...



The next way to view/edit the variable is using watches. The first type of watch is the "Quick Watch". A quick watch simply shows you the value in its current state. Right click on the variable and click "Quick Watch":

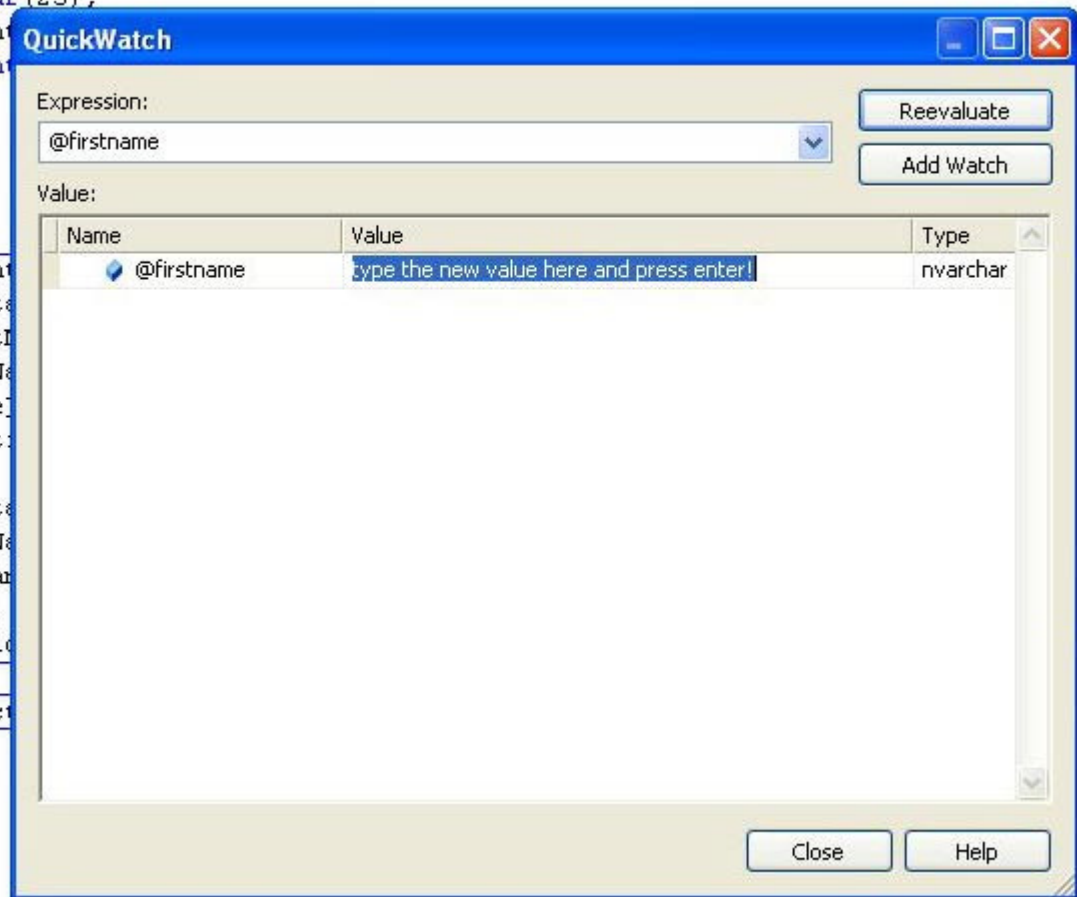


The quick watch window will be displayed:

```
name nvarchar(50),  
ame nvarchar(50),  
nvarchar(25),  
ionalconf  
ctid int
```

t on

```
O [Adventureworks2008].[dbo].[Salute]  
([Salute]  
,[FirstN  
,[LastNa  
,[Phone]  
,[Additio  
S  
(@Salute  
@FirstNa  
@LastNa  
@Phone,  
@Additio  
@contact
```



The second type of watch is the "old school" watch using the "Add Watch" option. Right click on the variable and click "Add Watch". Clicking "Add Watch" will add the variable to the watch window. If you don't see the watch window press CTRL + D, W to bring it into focus.

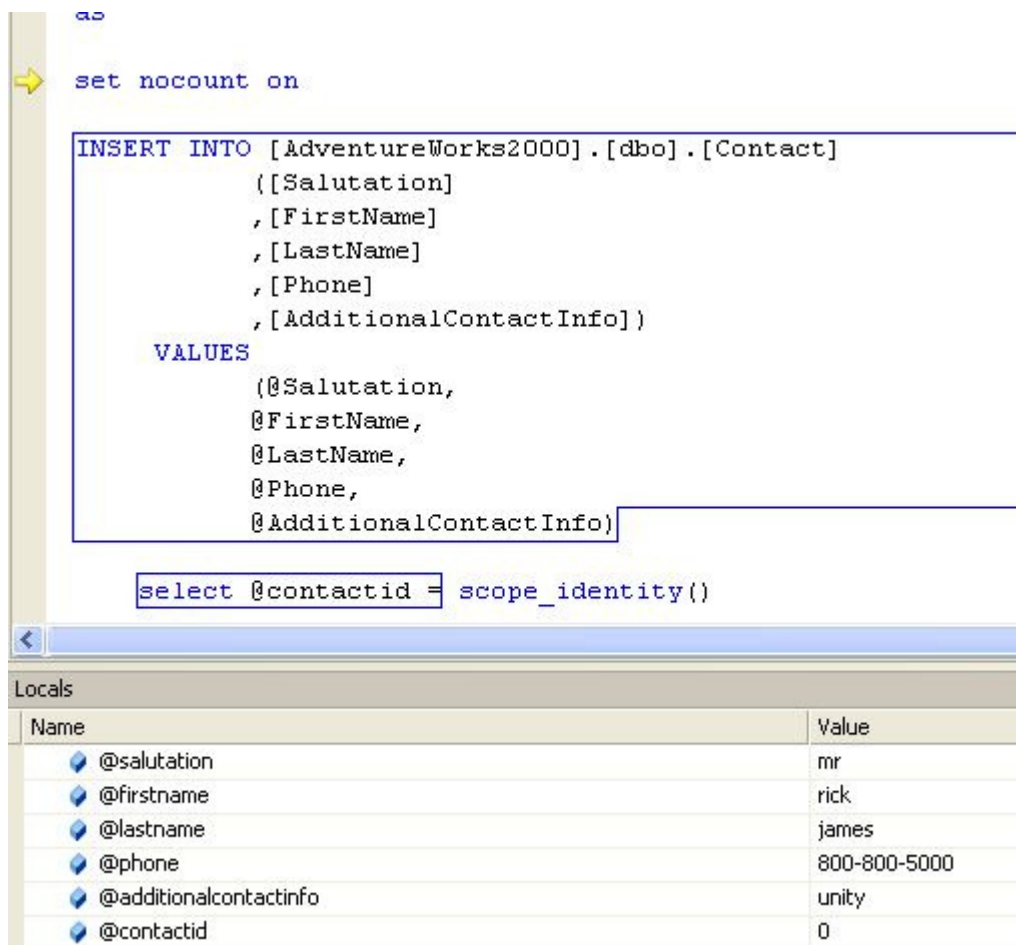
→ set nocount on

```
INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    ,[FirstName]
    ,[LastName]
    ,[Phone]
    ,[AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)
```

Watch 1

Name	Value
⚙ @firstname	I changed the value here too! SWEET!

The third way of viewing / editing the variables is the locals window...my favorite! (CTRL + D, L). This window will list all of the variables and their associated values:



If you're feeling adventurous you might give the immediate window (CTRL + D, I) a try.

## I love breakpoints, you love breakpoints, If we didn't use breakpoints, we'd be lame.

(Note: The above title is set to the "Cheesy Poofs" theme song from South Park episode 211)

Another cool feature while debugging procedures is breakpoints. If you don't want to deal with stepping through each line of code to get to the problem code you can add a breakpoint. A breakpoint will stop execution at the line of code the breakpoint is set on while the debugger is active. This is invaluable when debugging a really large procedure or the last DBA's code. We all know the last DBA didn't know what they were doing!

To add a breakpoint click on the line of code where you want the breakpoint and press the F9 key. After you press F9 you should see a red "ball" to the left of code where the breakpoint was set:

```

ALTER proc dbo.usp_contact_add
(
    @salutation nvarchar(8),
    @firstname nvarchar(50),
    @lastname nvarchar(50),
    @phone nvarchar(25),
    @additionalcontactinfo ntext,
    @contactid int output
)
as

set nocount on

INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    , [FirstName]
    , [LastName]
    , [Phone]
    , [AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)

select @contactid = scope_identity()

```

Now we want to execute the procedure and hit our breakpoint. Right click the code window and click "Step Into Stored Procedure". Again, fill in the data in the parameters window and click OK. The debugger will again stop on the first line of code. Click F5 to tell the debugger to start running again. The debugger will now stop execution at the breakpoint:

```
        @additionalcontactinfo ntext,
        @contactid int output
    )
as

set nocount on

INSERT INTO [AdventureWorks2000].[dbo].[Contact]
    ([Salutation]
    , [FirstName]
    , [LastName]
    , [Phone]
    , [AdditionalContactInfo])
VALUES
    (@Salutation,
    @FirstName,
    @LastName,
    @Phone,
    @AdditionalContactInfo)

select @contactid = scope_identity()
```

There you go. Getting through all of the code you don't want to execute manually is as easy as pressing F9 (<< That sets the breakpoint).

(Note: Pressing F5 while the debugger is active will execute the stored proc until it: 1) hits a breakpoint. 2) Exits the proc.)

## Conclusion

That's it. You know now how to debug a stored procedure. I hope you never have to debug a proc. Why? Because it should be done right the first time. Sorry, I tried not to laugh when I wrote that. We all know that mistakes happen, sales and marketing teams happen and in general, life happens. I fully expect you to utilize the power of the VS2005 debugging features. Remember to have fun while doing it. Otherwise it's just work...