



source: <http://www.MSSQLTips.com/tip.asp?id=1495> -- printed: 5/25/2011 11:04:18 AM

Getting started with SQL Server stored procedures

Written By: Greg Robidoux -- 5/9/2008

Problem

I have been using SQL Server for some time, but all of the code that is issued against the database is embedded in the application code. I know that you can create stored procedures, but I am not exactly sure where to start or what I need to do to implement stored procedures.

Solution

Stored procedures are nothing more than a batch of T-SQL statements that are stored in the database. Instead of having to issue multiple statements from your application you can issue one command to call the stored procedure to do a batch of work instead of just one statement. In addition, since the code is stored in the database you can issue the same set of code over and over again even from different applications or a query window. To get started the rest of this tip looks at some sample stored procedures and how you can get started and build upon them.

The below examples show you how simple it is to create stored procedures. All of these examples use the AdventureWorks database, but these should be pretty straightforward that you can apply these concepts to your own databases and applications.

Example 1 - simple stored procedure

This first example creates a simple stored procedure that gets the TOP 1 record from the Person.Contact table.

```
CREATE PROCEDURE uspGetContact
AS
SELECT TOP 1 ContactID, FirstName, LastName
FROM Person.Contact
```

After the above has been created use the command below to execute this stored procedure.

```
EXEC uspGetContact
```

This is the results from this first query.

ContactID	FirstName	LastName
1	Gustavo	Achong

Example 2 - stored procedure with a parameter

This next example is a modification of the first example, but this time adding a parameter that is passed into the procedure to dynamically select the records. Instead of using CREATE PROCEDURE we are using ALTER PROCEDURE to modify the procedure that we created in Example 1 instead of dropping it first and then recreating it.

```
ALTER PROCEDURE uspGetContact @LastName NVARCHAR(50)
AS
SELECT TOP 1 ContactID, FirstName, LastName
FROM Person.Contact
```

```
WHERE LastName = @LastName
```

Below shows two different ways the stored procedure can be run. The first example just passes the parameter value we want to use and the second example also includes the parameter name along with the value. You can run the stored procedure with either one of these commands.

```
EXEC uspGetContact 'Alberts'
```

```
EXEC uspGetContact @LastName='Alberts'
```

This is the results from this first query.

	ContactID	FirstName	LastName
1	17	Amy	Alberts

Example 3 - stored procedure with a parameter and output parameter

In this example we have both an input parameter as well as an OUTPUT parameter. The output parameter will be used to pass back the ContactID that we are looking up in the stored procedure. This output parameter will then be used to select the persons ContactID, FirstName and LastName along with any address records for this person.

Again we are altering the stored procedure uspGetContact and then secondly we are running the next set of code that executes procedure uspGetContact and then based on the return value it gets it will also query for the persons name and address info.

```
ALTER PROCEDURE uspGetContact @LastName NVARCHAR(50), @ContactID INT output
AS
SELECT TOP 1 @ContactID = c.ContactID
FROM HumanResources.Employee a
INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
INNER JOIN Person.Address d ON b.AddressID = d.AddressID
WHERE c.LastName = @LastName
```

After the stored procedure has been altered run the below block of code. This will execute the above stored procedure and if the ContactID has a value it will also return the person and address info.

```
DECLARE @ContactID INT
SET @ContactID = 0
EXEC uspGetContact @LastName='Smith', @ContactID=@ContactID OUTPUT
IF @ContactID <> 0
BEGIN
SELECT ContactID, FirstName, LastName
FROM Person.Contact
WHERE ContactID = @ContactID

SELECT d.AddressLine1, d.City, d.PostalCode
FROM HumanResources.Employee a
INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
INNER JOIN Person.Address d ON b.AddressID = d.AddressID
WHERE c.ContactID = @ContactID
END
```

This is the results from this first query.

	ContactID	FirstName	LastName
1	1095	Samantha	Smith

	AddressLine1	City	PostalCode
1	1548 Eastgate Lane	Bellevue	98004

Example 4 - stored procedure using the RAISERROR statement

In this example we are combining the two steps in Example 3 into one stored procedure. The first step is to get the ContactID and then the second part of the procedure will lookup the persons name and address info. We also added in code to use the RAISERROR statement to return an error if no records are found.

This is then being run twice to show what it looks like when data is found and when no data is found. The RAISERROR statement can be used to control how your application handles no data or any other error that may occur.

```
ALTER PROCEDURE uspGetContact @LastName NVARCHAR(50)
AS
DECLARE @ContactID INT
SELECT TOP 1 @ContactID = c.ContactID
FROM HumanResources.Employee a
    INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
    INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
    INNER JOIN Person.Address d ON b.AddressID = d.AddressID
WHERE c.LastName = @LastName

IF @@ROWCOUNT > 0
BEGIN
    SELECT ContactID, FirstName, LastName
    FROM Person.Contact
    WHERE ContactID = @ContactID

    SELECT d.AddressLine1, d.City, d.PostalCode
    FROM HumanResources.Employee a
        INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
        INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
        INNER JOIN Person.Address d ON b.AddressID = d.AddressID
    WHERE c.ContactID = @ContactID
END
ELSE
BEGIN
    RAISERROR ('No record found',10,1)
END
```

```
EXEC uspGetContact @LastName='Walters'
```

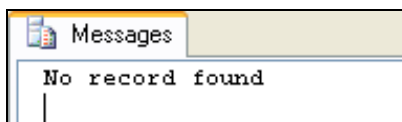
This is the results from this first query.

	ContactID	FirstName	LastName
1	1290	Rob	Walters

	AddressLine1	City	PostalCode
1	5678 Lakeview Blvd.	Minneapolis	55402

```
EXEC uspGetContact @LastName='Job'
```

This is the results from this first query when no data is found.



Example 5 - stored procedure with a separate calling stored procedure

Here is another example where we have two stored procedures. The first stored procedure `uspFindContact` lookups the first record that has an address record and then returns the `ContactID` to the calling stored procedure to again display the person and address info.

```
CREATE PROCEDURE uspFindContact @LastName NVARCHAR(50), @ContactID INT output
AS
SELECT TOP 1 @ContactID = c.ContactID
FROM HumanResources.Employee a
INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
INNER JOIN Person.Address d ON b.AddressID = d.AddressID
WHERE c.LastName = @LastName
```

The code below does an alter of the `uspGetContact` stored procedure that calls `uspFindContact` and returns the recordsets.

```
ALTER PROCEDURE uspGetContact @LastName NVARCHAR(50)
AS
DECLARE @ContactID INT
SET @ContactID = 0

EXEC uspFindContact @LastName=@LastName, @ContactID=@ContactID OUTPUT

IF @ContactID <> 0
BEGIN
SELECT ContactID, FirstName, LastName
FROM Person.Contact
WHERE ContactID = @ContactID

SELECT d.AddressLine1, d.City, d.PostalCode
FROM HumanResources.Employee a
INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
INNER JOIN Person.Address d ON b.AddressID = d.AddressID

WHERE c.ContactID = @ContactID
END
ELSE
BEGIN
RAISERROR ('No record found',10,1)
```

```
END
```

```
EXEC uspGetContact @LastName='Walters'
```

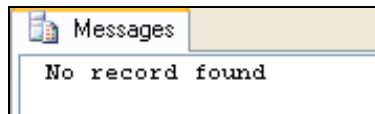
This is the results from this first query.

	ContactID	FirstName	LastName
1	1290	Rob	Walters

	AddressLine1	City	PostalCode
1	5678 Lakeview Blvd.	Minneapolis	55402

```
EXEC uspGetContact @LastName='Job'
```

This is the results from this first query.



Example 6 - stored procedure with comments

This last example takes the uspGetContact stored procedure and adds comments to the code so you can see how comments work within a stored procedure. You can see that there are two ways that comments can be made. 1) using -- and 2) using /* to begin the comment block and */ to end the comment block. Other than that nothing else has changed.

```
ALTER PROCEDURE uspGetContact @LastName NVARCHAR(50)
AS
/* This is a sample stored procedure to show
   how comments work within a stored procedure */

-- declare variable
DECLARE @ContactID INT
-- set variable value
SET @ContactID = 0

-- execute stored proc and return ContactID value
EXEC uspFindContact @LastName=@LastName, @ContactID=@ContactID OUTPUT

-- if ContactID does not equal 0 then return data else return error
IF @ContactID <> 0
BEGIN
    SELECT ContactID, FirstName, LastName
    FROM Person.Contact
    WHERE ContactID = @ContactID

    SELECT d.AddressLine1, d.City, d.PostalCode
    FROM HumanResources.Employee a
        INNER JOIN HumanResources.EmployeeAddress b ON a.EmployeeID = b.EmployeeID
        INNER JOIN Person.Contact c ON a.ContactID = c.ContactID
        INNER JOIN Person.Address d ON b.AddressID = d.AddressID
    WHERE c.ContactID = @ContactID
END
```

```
END
```

```
END  
ELSE  
BEGIN  
    RAISERROR ('No record found',10,1)  
END
```

These are pretty simple examples, but hopefully this gives you an idea of how easy it is to create stored procedures for SQL Server. If you can run a SELECT statement from either a query window or from your application you can just as easily run a stored procedure as show above.

Next Steps

- If you are not already using stored procedures hopefully this gives you some insight as to what you need to do to begin using them
- As mentioned these are pretty simple examples, but just about anything you can do with a batch of statements can be combined into a stored procedure and then used over and over again for your applications.

Copyright (c) 2006-2011 [Edgewood Solutions, LLC](#) All rights reserved
[privacy](#) | [disclaimer](#) | [copyright](#) | [advertise](#) | [contribute](#) | [feedback](#) | [about](#)
Some names and products listed are the registered trademarks of their respective owners.

[CareerQandA.com](#) | [MSSharePointTips.com](#) | [MSSQLTips.com](#)