SQL Server Stored Procedures and Parameters

Parameters in stored procedures in SQL Server allow you to pass values into and out of the stored procedure. These parameters can be used to control the behavior of the stored procedure or to return values to the calling program. There are three types of parameters in SQL Server stored procedures:

1. Input Parameters: Pass values into the stored procedure.

2. Output Parameters: Return values from the stored procedure to the calling program.

3. Input/Output (INOUT) Parameters: Pass values into the stored procedure and return updated values back to the calling program.

Input Parameters

Input parameters are used to pass values into the stored procedure. They are defined using the '@' symbol followed by the parameter name and data type.

Syntax:

```
CREATE PROCEDURE procedure_name
   @parameter1 datatype,
   @parameter2 datatype
AS
BEGIN
   -- SQL statements
END;
```

Example:

```
CREATE PROCEDURE GetEmployeeDetails
   @EmployeeId INT
AS
BEGIN
   SELECT EmployeeId, EmployeeName, DepartmentId
   FROM Employees
   WHERE EmployeeId = @EmployeeId;
END;
```

To execute the stored procedure with an input parameter:

```
EXEC GetEmployeeDetails @EmployeeId = 1;
```

Output Parameters

Output parameters are used to return values from the stored procedure. They are defined using the OUTPUT keyword.

Syntax:

```
CREATE PROCEDURE procedure_name
   @input_parameter datatype,
   @output_parameter datatype OUTPUT
AS
BEGIN
   -- SQL statements
END;
```

Example:

```
CREATE PROCEDURE GetEmployeeCountByDepartment
   @DepartmentId INT,
   @EmployeeCount INT OUTPUT
AS
BEGIN
   SELECT @EmployeeCount = COUNT(*)
   FROM Employees
   WHERE DepartmentId = @DepartmentId;
END;
```

To execute the stored procedure with an output parameter:

```
DECLARE @Count INT;
EXEC GetEmployeeCountByDepartment @DepartmentId = 1, @EmployeeCount = @Count
OUTPUT;
SELECT @Count AS EmployeeCount;
```

Input/Output Parameters

Input/Output parameters serve both as input to the stored procedure and as output from the stored procedure.

Syntax:

```
CREATE PROCEDURE procedure_name
   @parameter1 datatype,
   @parameter2 datatype OUTPUT
AS
BEGIN
   -- SQL statements
END;
```

Example:

```
CREATE PROCEDURE UpdateEmployeeSalary
   @EmployeeId INT,
   @NewSalary DECIMAL(10, 2) OUTPUT
AS
BEGIN
   UPDATE Employees
   SET Salary = @NewSalary
   WHERE EmployeeId = @EmployeeId;

   -- Optionally, return the new salary
   SELECT @NewSalary = Salary
   FROM Employees
   WHERE EmployeeId = @EmployeeId;
END;
```

To execute the stored procedure with an input/output parameter:

```
DECLARE @Salary DECIMAL(10, 2);
SET @Salary = 75000.00;
EXEC UpdateEmployeeSalary @EmployeeId = 1, @NewSalary = @Salary OUTPUT;
SELECT @Salary AS UpdatedSalary;
```

Default Values for Parameters

You can also define default values for parameters. If a parameter value is not provided when the stored procedure is called, the default value will be used.

Syntax:

```
CREATE PROCEDURE procedure_name
   @parameter1 datatype = default_value
AS
BEGIN
   -- SQL statements
END;
```

Example:

```
CREATE PROCEDURE GetDepartmentEmployees
   @DepartmentId INT = 1
AS
BEGIN
   SELECT EmployeeId, EmployeeName
   FROM Employees
   WHERE DepartmentId = @DepartmentId;
END;
```

To execute the stored procedure with a default parameter value:

```
-- Uses the default DepartmentId value of 1
EXEC GetDepartmentEmployees;

-- Provides a specific DepartmentId value
EXEC GetDepartmentEmployees @DepartmentId = 2;
```

Using Multiple Parameters

A stored procedure can accept multiple parameters of any type, including a mix of input, output, and input/output parameters.

Example:

```
CREATE PROCEDURE GetEmployeeInfo
   @EmployeeId INT,
   @EmployeeName NVARCHAR(50) OUTPUT,
   @DepartmentId INT OUTPUT
AS
BEGIN
```

```
    SELECT @EmployeeName = EmployeeName, @DepartmentId = DepartmentId
    FROM Employees
    WHERE EmployeeId = @EmployeeId;
END;
```

To execute the stored procedure with multiple parameters:

```
DECLARE @Name NVARCHAR(50);
DECLARE @DeptId INT;
EXEC GetEmployeeInfo @EmployeeId = 1, @EmployeeName = @Name OUTPUT,
@DepartmentId = @DeptId OUTPUT;
SELECT @Name AS EmployeeName, @DeptId AS DepartmentId;
```