Scope Tutorial > Parameters and the Preprocessor

Author: Saveen Reddy

Date: 6/30/2015

# Table of Contents

[Table of Contents 1](#_Toc423450766)

[Parameters 1](#_Toc423450767)

[The Preprocessor 1](#_Toc423450768)

[Internal Parameters with #DECLARE 3](#_Toc423450769)

[External Parameters 3](#_Toc423450770)

# Parameters

Scope has several things that all share the name of **Parameters**.

* **Internal parameters** – these the values of these parameters are established through the **#DECLARE** preprocessor keyword. When used in a script, a preprocessor parameter will look like this: **@foo**. A single @ sign prefix identifies "foo" as a preprocessor parameter. For legacy reasons you may occasionally hear these referred to as "New-Style Parameters."
* **External parameters** – the values of these parameters are not in the Scope script – instead the values are provided by the user when the Script is run. When used in a script, a script submission parameter will look like this: **@@foo@@** - a pair of @@ symbols surround "foo" and identify it as a Script Parameter. For legacy reasons, you may occasionally hear these referred to as "Old-Style Parameters.”
* **Parameters in Views/Functions/Procedures** – These are parameters passed to Views, Functions, or Procedures by Scope code. Like preprocessor parameters, when you see them in the body of the View, Function, or Procedure they will appear like this: **@foo**.

# The Preprocessor

The preprocessor is a special limited language that is fully interpreted by Scope before the script is compiled. All statements starting with # are preprocessor statements.

Like preprocessors in other languages, the Scope preprocessor allows you do to several things:

* Define values that can be reused in your script – these are the **preprocessor parameters** we mentioned earlier that are created via the **#DECLARE** keyword.
* Switch between alternating sections of code depending on some condition using the **#IF** and **#ELSE** **keywords**.

## Local versus Remote

To facilitate development and debugging, you can use the **#IF** preprocessor directive to control the value **#DECLARE** statements, depending on which environment the script is running.

For example, you may have a very large dataset for production use, but may want to use a very small subset locally for development. Below, you can see how the processor can be used to use the correct input dataset depending on the location of the running script.

#IF(LOCAL)

#DECLARE inputfile string = "/input.tsv";

#ELSE

#DECLARE inputfile string = "/my/sampledata/input.tsv";

#ENDIF

## Detect if a Stream Exists with EXISTS()

The new **EXISTS()** Preprocessor command returns a **bool** indicating if a stream exists or not.

// get data from b.ss if it exists otherwise use a.sss

#IF (EXISTS("b.ss"))

    rs0 = SSTREAM "b.ss";

#ELSE

    rs0 = SSTREAM "a.ss";

#ENDIF

I told it to HR yesterday, and they told to me the information when I was an intern here will be update in 3days.

You can combine the result of **EXISTS()** into more complicated expressions.

#DECLARE x string = EXISTS("path") ? "a" : "b";

## Getting Stream Metadata with STREAM()

If the stream exists then **STREAM()** returns an object with several properties you can examine

**STREAM(path).LENGTH** return the size (long) of the stream in bytes

#IF (STREAM("path").LENGTH > 1024 \* 1024 \* 1024 \* 1024)

// stream is more than a terabyte

#ENDIF

**STREAM(path).CREATED** returns the DateTime when stream was created

#IF (STREAM("path").CREATED < new System.DateTime(2013,03,31))

  // stream was created before March 31 2013

#ENDIF

**STREAM(path).MODIFIED** returns the DateTime when the stream was last modified

#IF (STREAM("path").MODIFIED.Year == 2013)

  // stream was modified in 2013

#ENDIF

If the stream does not existthen **STREAM()** returns **null**.

# Internal Parameters with #DECLARE

You can use the **#DECLARE** instruction to create typed parameters. They are called Internal parameters because the value is set inside/internal to the Scope script.

#DECLARE Out\_File string = "Output.txt";

And then re-use "Out\_File" later in the script.

OUTPUT rs1 TO @Out\_File;

## Data Types

#DECLARE a string = "Hello World";

#DECLARE a2 string = @"Hello World";

#DECLARE b int = 10;

#DECLARE c bool = true;

#DECLARE d double = 1.0;

#DECLARE date0 DateTime = DateTime.Parse("2010/03/31");

## String Concatenation and Interpolation

You can use other variable declarations in a #DECLARE statement to combine strings.

#DECLARE str1 string = "Hello World";

#DECLARE str2 string = "BEGIN" + @str1 + "END";

#DECLARE str3 string = string.Format("BEGIN{0}END", @str1);

## DateTimes

#DECLARE ds1 string = "2010/03/31";

#DECLARE date0 DateTime = DateTime.Parse(@ds1);

#DECLARE date1 DateTime = DateTime.Now;

## User-Defined Code is Not Supported

You may be tempted to write something like this:

#DECLARE myName string = MyHelper.GetMyName();

This is NOT supported by Scope. DECLARE cannot call methods on types (like **MyHelper**) that are user-defined.

## Values cannot be assigned from a RowSet

You will be tempted to write something like this.

// This does \*NOT\* work

#DECLARE maxval int = SELECT MAX(value) FROM data;

This is explicitly **NOT supported**.

An alternative is to get a single-row rowset with a single column and then JOIN that tiny rowset other rowset to get what you need.

# External Parameters

We saw with the **internal parameters** that the values of the parameters are supplied in the Scope script itself. However, there are situations where you want to change the value being used and yet keep the script unchanged.

This is accomplished by using **external parameters**. The actual values for script parameters do not exist in the Scope script – instead when the Script is run the user must provide them.

Let's start with the script below. It gets all the rows from searchlog that are for a specific market.

rs1 = SELECT \*

FROM searchlog

WHERE Market=="en-us";

This script will compile and run successfully in your own box as well as when submitted to your virtual cluster.

Now, let's parameterize the value for **market** so that the value to be used in the **WHERE** clause is provided only when the script is run. The change is simple: we replace the "en-us" with @@MARKET@@.

rs1 = SELECT \*

FROM searchlog

WHERE Market==@@MARKET@@;

## Providing External Parameters

Different tools have different ways on specifying the parameters

* Scope.exe – via the –params argument (see more [**here**](https://microsoft.sharepoint.com/teams/Cosmos/Wiki/scope.exe.aspx))
* CosmosPowerShell – via the –Parameters argument (see more [**here**](https://microsoft.sharepoint.com/teams/Cosmos/CosmosPowerShell/Passing%20External%20Parameters%20to%20a%20Job.aspx))
* Scope Studio – in the Submit Job dialog box