Composer Help

Looping Common Block

4/3/2025

Looping Common Block

Contents

Use this block to iterate over a sequence of blocks multiple times in the following scenarios:

1. Iterate over a sequence of blocks based on a self-incrementing counter (FOR).

2. Iterate indefinitely until an exit condition is met (WHILE).

3. Iterate over records/data returned by the DB Data block (CURSOR/FOREACH). Also, populate variables if variables mapping is defined.

4. Iterate over data returned by Context Services blocks (FOREACH). Also, populate variables if Variables Mapping is defined.

5. Iterate over JSON Array defined in the application.

For scenarios 1 and 2 above, use the Looping block with a reference to the block retrieving the data. Scenarios 3 and/or 4 can be used in conjunction with 1 or 2, in which case the loop will exit when either of the exit conditions is met.

Prerequisite

You must perform the following steps in order for the Looping block to be used to iterate over data returned by the DB Data block:

1. Create a folder named Scripts in the Project folder.

2. In the Entry block, specify a value for the Scripts property such as ../include/DBRecordSetAccess.js

The Looping block has the following properties:

Name Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks.

Block Notes Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks.

Counter Initial Value Property

A Counter variable controls the number of loops. Specify the initial value by entering a positive integer (including zero) or selecting the variable that contains the initial value. Composer will

increment the Counter variable after each iteration. The value of the Counter variable is available after the looping has exited. This is a mandatory property if the Counter Variable property is specified.

Counter Variable Property

Enter a name for the variable used to store the Counter value or select the variable that contains the name. This is a mandatory property if the Counter Initial Value property is specified.

Current Record Variable Property

Select a variable to be used to store the current record when iterating over records. Composer will assign the current record after each iteration. This property is ignored if the Data Source Property is not set

Data Source Property

Specify a block reference to the DB Data or a Context Services block (with Variables Mapping support) that provides the data to be iterated or select the variable that contains a JSON Array. This is a mandatory property if Counter Initial Value and Counter Variable are not specified.

Exceptions Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks. You can also define custom events.

Counter Max Value Property

Specify the maximum value by entering a positive integer greater than the initial value or selecting the variable that contains the maximum value. When the Counter variable reaches the maximum value, then the block connected to the Exit port is executed. This is a mandatory property if the Counter Variable property is specified or if the Data Source property is not specified.

Exit Expression Property

This property is optional. If specified, prior to each iteration the exit expression is evaluated. If true, the flow goes out via the Exit port of the block. This condition is used in conjunction with max records

(if Data Source is specified) or Counter Max Value (if Counter Variable is specified). To enter an exit expression

1. Opposite the **Exit Expression** property, click under **Value** to display the ▦ button.

2. Click the ▦ button to open Expression Builder. For examples of how to use Expression Builder, see the Expression Builder topic.

3. Create the exit expression and click **OK**.

Condition Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks.

Logging Details Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks.

Log Level Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks.

Enable Status Property

Find this property's details under Common Properties for Callflow Blocks or Common Properties for Workflow Blocks.
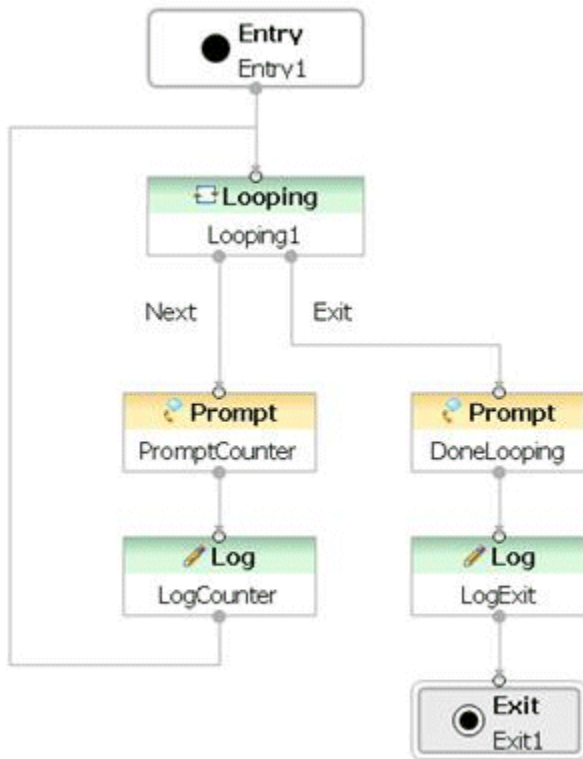
ORS Extensions Property

Starting with 8.1.4, Composer blocks used to build routing applications (with the exception of the Disconnect and EndParallel blocks) add a new ORS Extensions property.

Using the Looping Block (Counter-based without a Data Source)

1. Add a Looping block and connect the previous block outport to the Looping block.

2. Connect the Next outport to the sequence of connected blocks.

3. Connect the outport of the last block in the sequence in step 2 back to the looping block to form a loop.

4. Connect the Exit outport of the looping block to the block(s) to continue processing after the loop has exited. The diagram when a looping block is used should appear as follows:

FOR loop: To iterate over the PromptCounter block 10 times, the following properties are set:
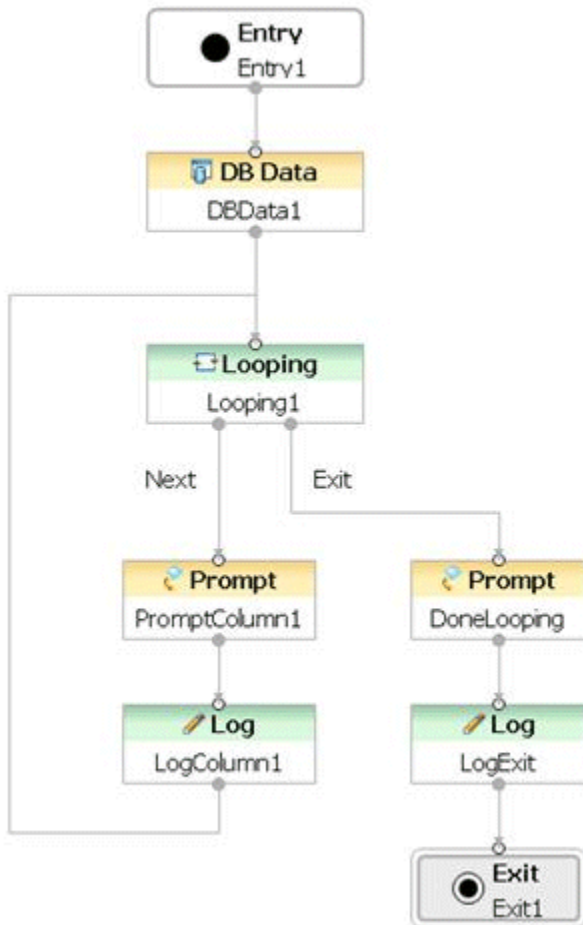
1. Counter Initial Value is 1.

2. Counter Variable Name is Variable(MyCounterVariable).

3. Counter Max Value is 10.

WHILE loop: To iterate over the PromptCounter block until a condition is satisfied, the following property is set: Exit expression is loginSuccessful != true.

Using the Looping Block with a DB Data/Context Services Block

1. Add a Looping block and connect the DB Data/Context Services block outport to the Looping block.

2. Connect the Next outport to the sequence of connected blocks.

3. Connect the outport of the last block in the sequence in step 2 back to the looping block to form a loop.

4. Connect the Exit outport of the looping block to the block(s) to continue processing after the loop has exited. The diagram when a looping block is used should appear as follows:

CURSOR/FOREACH loop: To iterate over the PromptColumn1 block for each record returned by the DBData1 block, the following property is set: Data Source = Block Reference (DBData1) This example assumes variables were mapped for Column1 in DB Data1. If variables were not mapped, then another Assign block would be needed to store the value into a variable and the variable is then specified in the PromptColumn1 block.