Talend Component tMustacheOutput

Purpose and procedure

This component renders incoming data with Mustache-templates.
The Mustache is a very simple templating engine which does not proviedes any logic but is therefore very easy to use and maintain. Here is the specification of the template language: https://mustache.github.io/mustache.5.html

A very good explanation can be found here: https://www.bersling.com/2017/09/22/the-ultimate-mustache-tutorial/

This component works in 2 modes:

1. Render all incoming rows as one document at the end of the flow
2. Render every incoming row immediately as document and put this to an output column.

Talend-Integration

This component can be found in the palette under Processing->Fields
This component provides an input flow and several return values (depending on the operational mode).

Parameters and Usage

Property	Content	Data types
Schema	Setup here your incoming and outgoing schema. Outgoing is optional and only essential if you want to render incoming rows immediately (row by row). ***Required***	
Mustache Template	Setup here the template. You can write here the template in the same way as you do in any database component for the SQL query. You need at minimum a quote at the beginning and at the end of the template. It is also possible to use a context variable or a globalMap-variable. ***Required***	String
Fail if template has fields not exists in te schema	This option let the component check if the template expects fields which do not exists in the schema. This way you can prevent using the wrong template for the job.	
Choose mode for rendering	Mode switch (see list above)	
Name of the root list	This attribute is relevant for the mode 1. When you render multiple rows into one document you need a list element and here you can name it to use this name in your template. ***Required***	String
Name of the output column for the document created for every row	This attribute is relevant for the mode 2. When you render row by row the output document will be assigned to an output column. Choose here the column from the out going schema. ***Required***	
Locale for numbers	Setup here the locale used to define the number grouping and decimal delimiter character. ***Required***	String

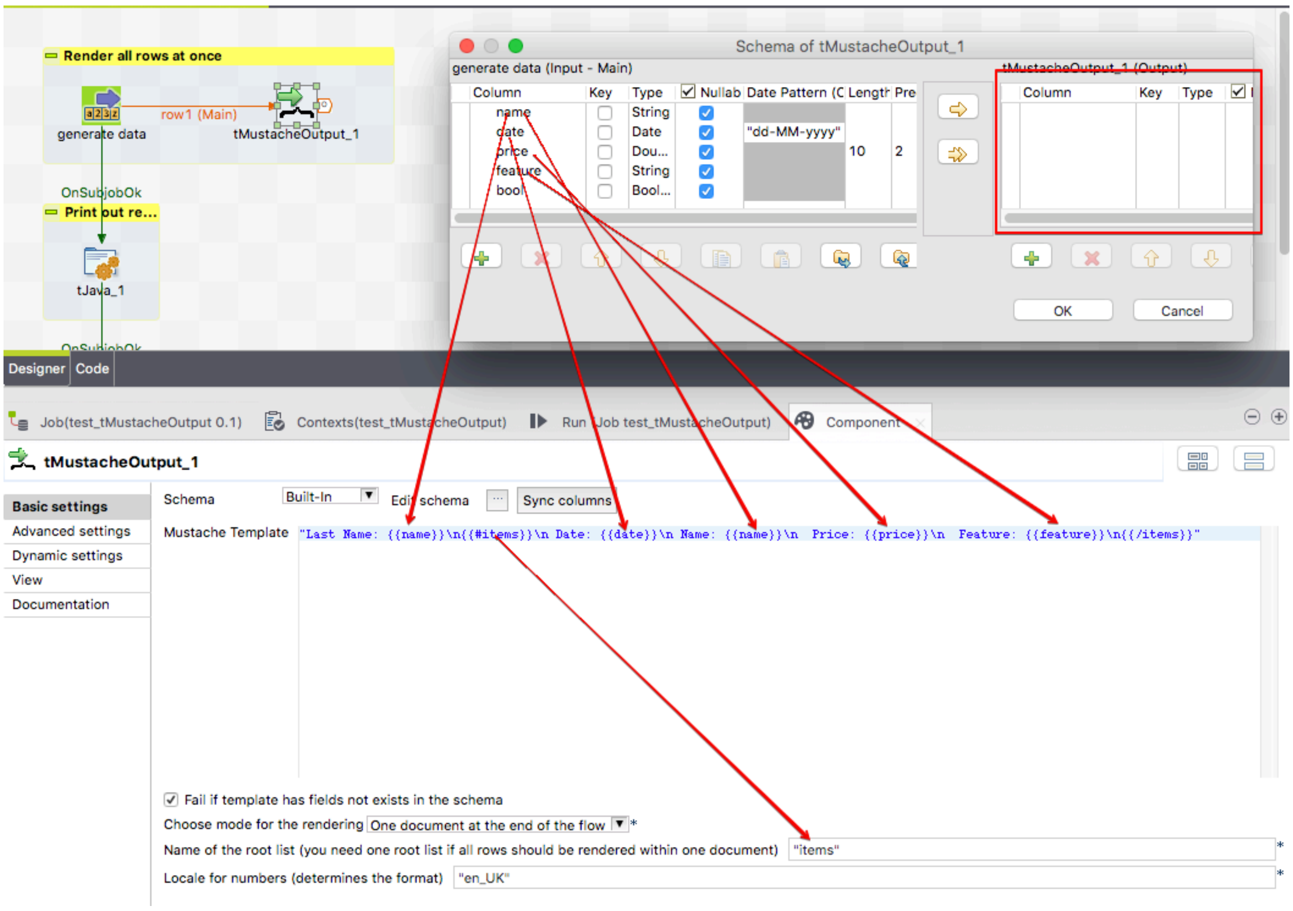
Return values of the component:

Value	Content
NB_LINE	Number of lines read
ERROR_MESSAGE	Error message if something went wrong
OUTPUT_DOCUMENT	Rendered document as String

Use-Cases:

Render all rows in one document

This example job generates simple data and all rows will be rendered into one result. To do so you need the iteration to let the template engine iterate through the fetched records. Because the iteration needs a name, that why you have to configure it in the component.



Schema of tMustacheOutput_1

generate data (Input - Main)

Column	Key	Type	✓ Nullab	Date Pattern (C	Length	Pre
name	☐	String	✓			
date	☐	Date	✓	"dd-MM-yyyy"		
price	☐	Dou...	✓		10	2
feature	☐	String	✓			
bool	☐	Bool...	✓			

tMustacheOutput_1 (Output)

Column	Key	Type	✓

tMustacheOutput_1

Schema: Built-In — Edit schema

Mustache Template: `"Last Name: {{name}}\n{{#items}}\n Date: {{date}}\n Name: {{name}}\n  Price: {{price}}\n  Feature: {{feature}}\n{{/items}}"`

☑ Fail if template has fields not exists in the schema

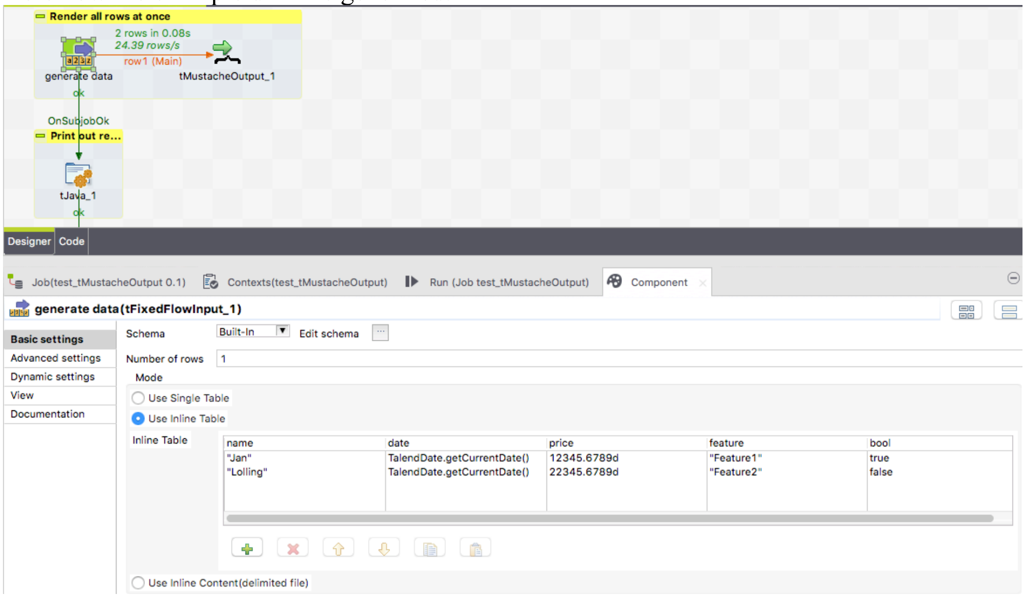
Choose mode for the rendering: One document at the end of the flow *

Name of the root list (you need one root list if all rows should be rendered within one document): "items" *

Locale for numbers (determines the format): "en_UK" *

The names in the template must match to names of the schema, otherwise if the schema does not contains all fields mentioned in the template the component will raise an error. The schema can of course have more columns than expected from the template.

Here how the example data are generated:



generate data(tFixedFlowInput_1)

Schema: Built-In — Edit schema

Number of rows: 1

Mode

○ Use Single Table

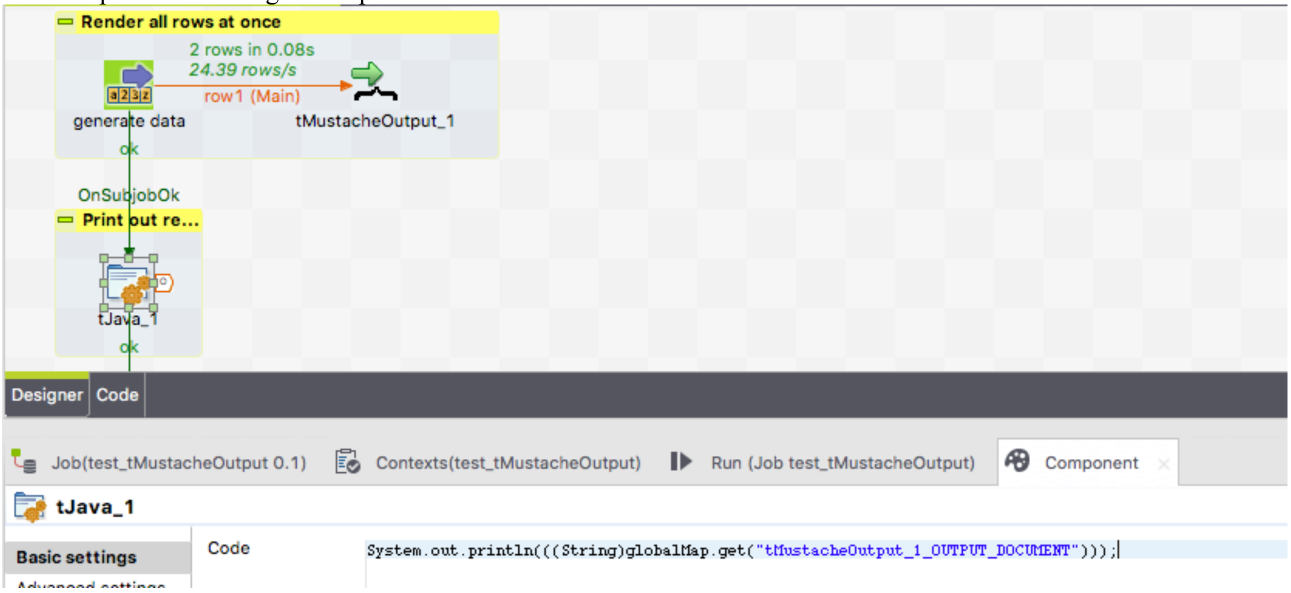
● Use Inline Table

Inline Table

name	date	price	feature	bool
"Jan"	TalendDate.getCurrentDate()	12345.6789d	"Feature1"	true
"Lolling"	TalendDate.getCurrentDate()	22345.6789d	"Feature2"	false

○ Use Inline Content(delimited file)

Here the part of rendering the output to the console:



```
System.out.println(((String)globalMap.get("tMustacheOutput_1_OUTPUT_DOCUMENT")));
```

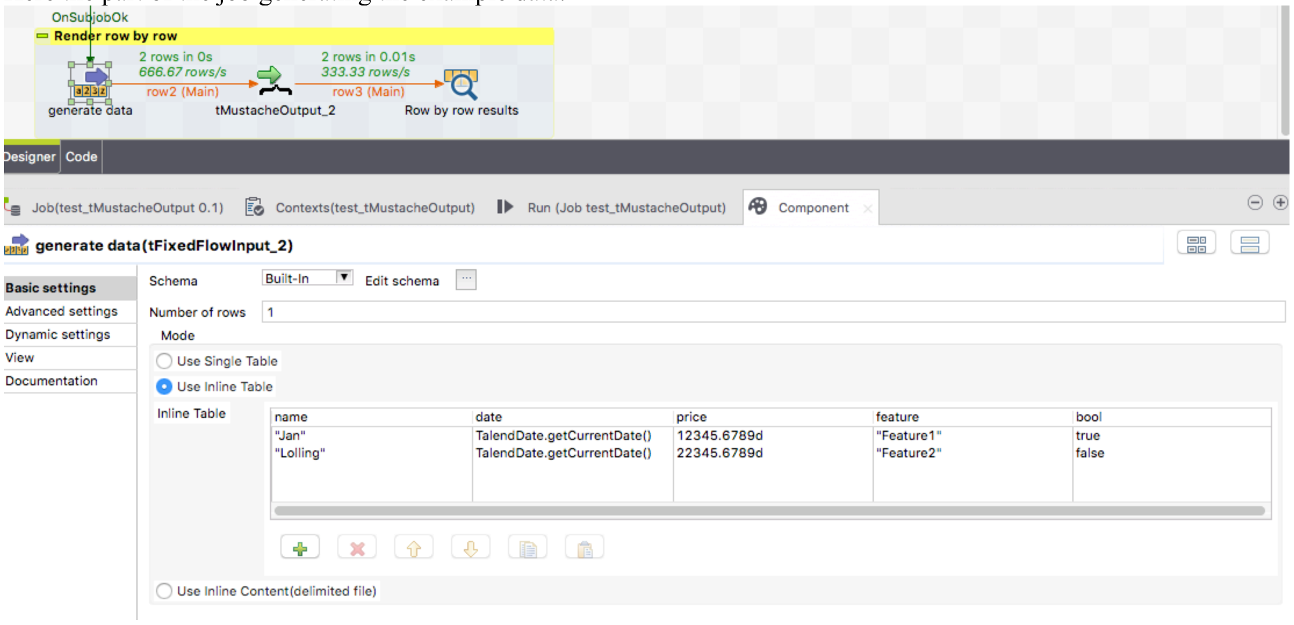
And here how the output looks like:

```
Last Name: Lolling
 Date: 24-05-2019
 Name: Jan
  Price: 12345.68
  Feature: Feature1
 Date: 24-05-2019
 Name: Lolling
  Price: 22345.68
  Feature: Feature2
```

If a field appears outside the iteration, the last value will be used.

Rendering every incoming row

If you want to render incoming rows ro by row you have to choose mode 2.
Here the part of the job generating the example data:



generate data(tFixedFlowInput_2)

Schema: Built-In — Edit schema

Number of rows: 1

Mode

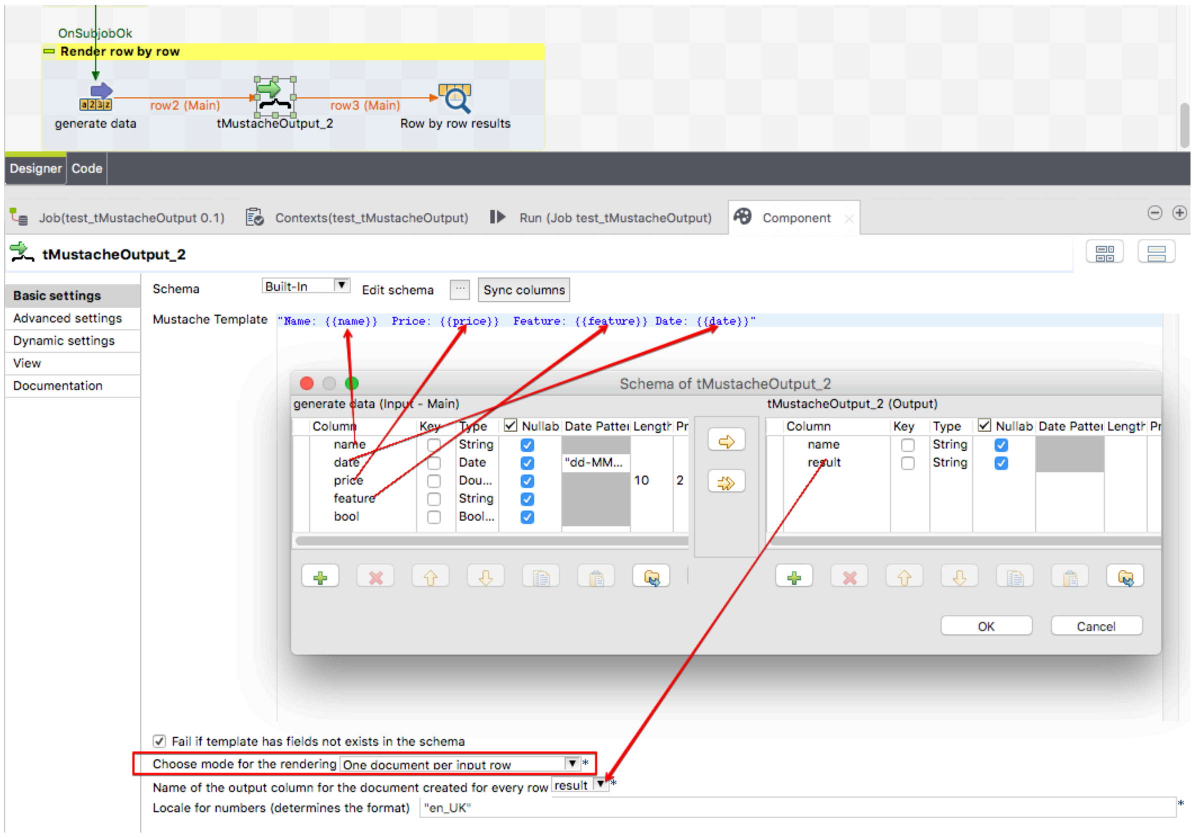
- Use Single Table
- [✓ Use Inline Table]

Inline Table

name	date	price	feature	bool
"Jan"	TalendDate.getCurrentDate()	12345.6789d	"Feature1"	true
"Lolling"	TalendDate.getCurrentDate()	22345.6789d	"Feature2"	false

- Use Inline Content(delimited file)

Here the part of the job rendering the documents (row by row -> mode 2)



tMustacheOutput_2

Schema: Built-In — Edit schema

Mustache Template: "Name: {{name}} Price: {{price}} Feature: {{feature}} Date: {{date}}"

Schema of tMustacheOutput_2

generate data (Input - Main)

Column	Key	Type	Nullab	Date Patter	Length	Pr
name		String	✓			
date		Date	✓	"dd-MM...		
price		Dou...	✓		10	2
feature		String	✓			
bool		Bool...	✓			

tMustacheOutput_2 (Output)

Column	Key	Type	Nullab	Date Patter	Length	Pr
name		String	✓			
result		String	✓			

✓ Fail if template has fields not exists in the schema

Choose mode for the rendering: One document per input row

Name of the output column for the document created for every row: result

Locale for numbers (determines the format): "en_UK"

And this is how the output looks like:

```
.-------+----------------------------------------------------------------.
|                          Row by row results                            |
|=------+---------------------------------------------------------------=|
|name   |result                                                          |
|=------+---------------------------------------------------------------=|
|Jan    |Name: Jan  Price: 12345.68  Feature: Feature1 Date: 24-05-2019  |
|Lolling|Name: Lolling  Price: 22345.68  Feature: Feature2 Date: 24-05-2019|
'-------+----------------------------------------------------------------'
```