Beanshell Scripting

Note that Beanshell scripting is for advanced users only as it requires some programming skills.

For a detailed Beanshell manual see the Beanshell homepage. This manual describes only the MEME-related details to help you create custom expressions and scripts in the *View Creation Wizard*.

Scripts can contain multiple lines. Their return value will always be the value of the last evaluated assignment or method calling or the value that is given back with a return statement.

There are two types of scripts in MEME: scalar and aggregative. A scalar script returns a single value while an aggregative script returns an array. In scripts you can refer to the parameters with their names, because MEME creates a global variable for each parameter. If you use a parameter in a scalar script, its value will be a single value (`double`, `int`, `String` etc.). On the other hand, in an aggregative script a reference to a parameter means an array (`double[]`, `int[]`, `String[]` etc.). For example, while in scalar script `p+3` can be a valid expression, in an aggregative script it is not. On the other hand `p.length` is only valid in an aggregative script.

Predefined Methods

Since ticks and runs are cornerstones in simulation data, MEME has built-in scripts to return them: `$Tick$` and `$Run$`. Following is the list of other predefined methods you can use.

- `get("name")`: Returns the value of the parameter named `name`. There can be a difference between the return value of this method and the value of the variable created from the `name` parameter:
    - The `get("name")` always returns the original value of the parameter, even if the value of the global variable `name` is changed. (Except in an aggregative script when the variables are arrays and its elements are replaced and not the whole array.)
    - If the name is not a valid Java identifier (for example *Column0.1* or *Column()*), therefore there is no global variable name, you cannot access the parameter with its name, the method must be used.

  If parameter `name` doesn't exist or name is `null`, the script will throw an error.
- `geti("name")`, `geto("name")`, `getv("name")`: Returns the value of the parameter named `name` from the input/output/this view category. This is necessary when there are more parameters with the same name but in different categories. In this case you can access the first parameter only via the global variable `name` or the `get("name")` method. The scope order is the following: *Output*, *This view*, *Input* (same as the order of the list of available parameters in the wizard). For example, the parameter named `name` in the *Output* category hides the parameter with the same name in the other categories.
- `gett("name")`: Returns technical parameter values, which are the following:
    - `batch`: The current batch number or null if the current input row doesn't belong to a batch (e.g. there is no corresponding row in the view table).
    - `run`: The current run or null if the current input row doesn't belong to a run.
    - `tick`: The current tick in case of a result table and the ordinal number *(#)* in the case of a view table.
    - `model`: The name of the current model or null if the current input row doesn't belong to a model.

- `version`: The current version or null if the current input row doesn't belong to a model.
- `starttime`: The start time (long) of the current run or null if the current input row doesn't belong to a result table
- `endtime`: The end time (long) of the current run or null if the current input row doesn't belong to a result table.
- `view`: The name of the current view table or null if the current input row doesn't belong to a view.
- `isgroupfirst`: True if there is grouping and the current row is the first in the group, otherwise false.
- `isgrouplast`: True if there is grouping and the current row is the last in the group, otherwise false.
- `isblockfirst`: true if there is blocking and the current row is the first in the block, otherwise false.
- `isblocklast`: true if there is blocking and the current row is the last in the block, otherwise false.

- `call("fn", args…)`: Calls the aggregation operation named `fn`. For example, the result of the `call("AVG", p1, p2)` is the average of *p1* and *p2*. In `args` you can enumerate arbitrary expressions, not just parameter variables. If you want to call `fn` with 11 or more arguments, you must use a `Object[]` as args. `"fn"` is the name of the aggregative operation, the same that appears in the Aggregative operation drop down list with or without brackets. You can also use the fully qualified name of the Java class that implements the aggregative operation (for example, `"ai.aitia.meme.builtinvcplugins.Avg"`).
- `round("name", precision)`: Rounds the value of the parameter named `name`. `precision` is the maximum number of digits after the decimal sign. This is an optional argument; without it the operation returns the closest integer.
- `sout(msg)`: Prints `msg` to the log file. About the log file see section 9.1 in the MEME Manual. `msg` is not necessarily a string, it can be an arbitrary object or primitive value too. Please note that all `System.out.println()` and `System.err.println()` prints to the log file too.