



Bonita BPM

Version 6.2.6

Community Tutorial

Mongo DB Connector

Summary

Objectives.....	3
Environment.....	3
Step 1: Import the Mongo DB Connector	4
Step 2: Define the Business Process.....	7
Step 3: Launch a case	19

Objectives

During this tutorial, you will learn how to use an already developed Mongo DB Connector to define requests in a Bonita Business Process definition.

Environment

Here is the environment used to write this tutorial.

- Windows 7.
- Bonita Studio 6.2.6 installed (either Community or Subscription version): <http://www.bonitasoft.com/how-we-do-it/downloads>.
- Mongo DB Connector Tut materials: <https://github.com/bonitasoft-community/MongoDBConnectorTut>
- Mongo DB v2.4.9: <http://www.mongodb.org/downloads>

It does not mean that it could not be possible to do it with other one but be aware that several steps can be specific to this.

Step 1: Import the Mongo DB Connector

This second step purpose is to import all the elements into the Studio to be able to use the Mondo DB Connector on a process.

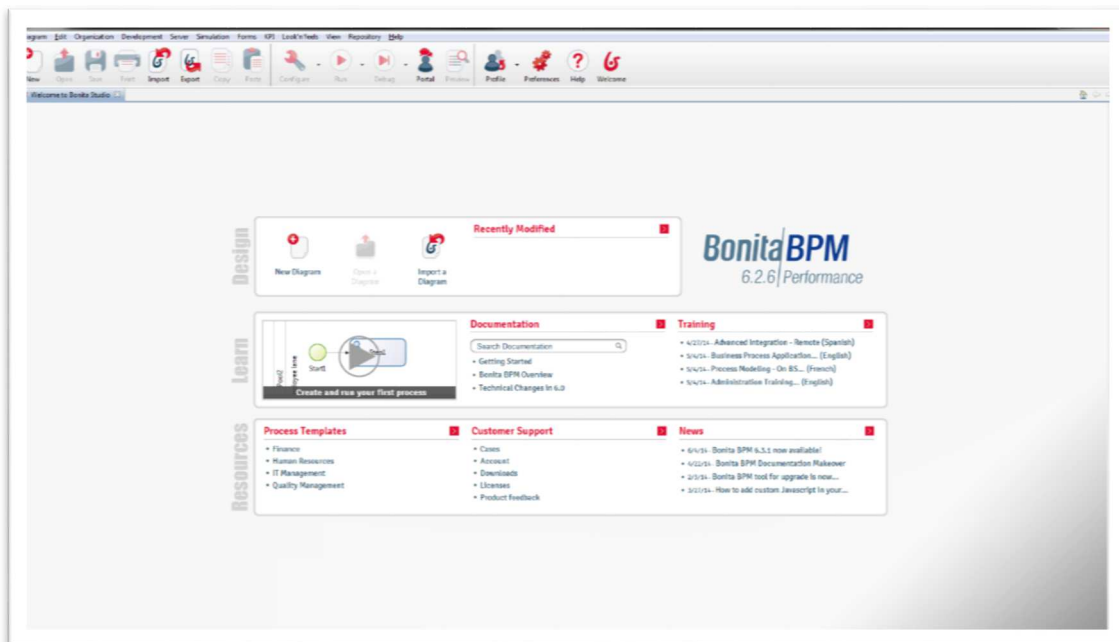
1. Main steps

- ✓ Open Bonita Studio.
- ✓ Import the external JAR used by the Mongo DB Connector.
- ✓ Import the Mongo DB Connector as a connector.

2. Instructions

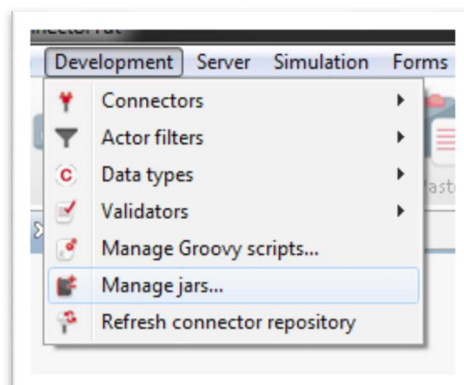
Open Bonita Studio

Once installed, double click on the executable file. Bonita Studio is now opened.

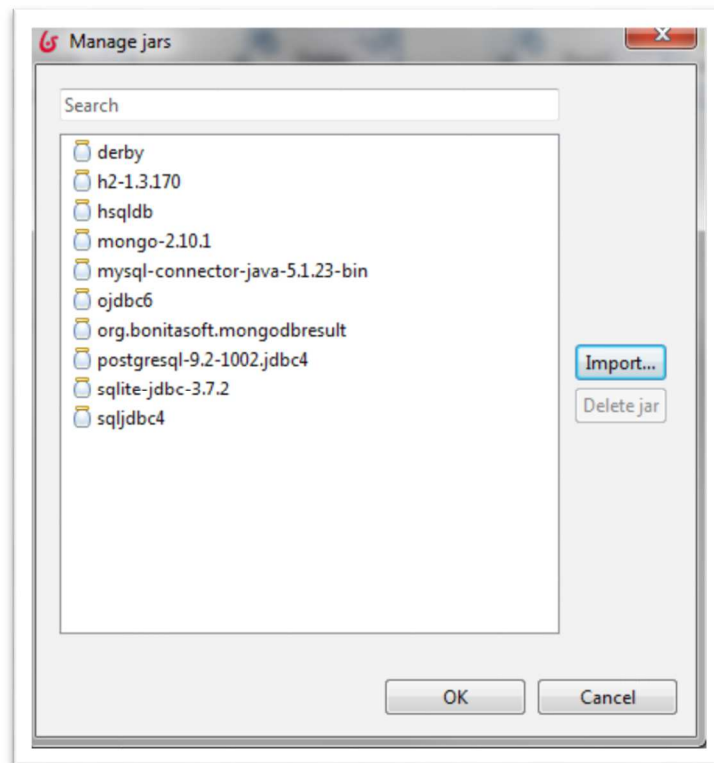


Import the external JAR used by the Mongo DB Connector

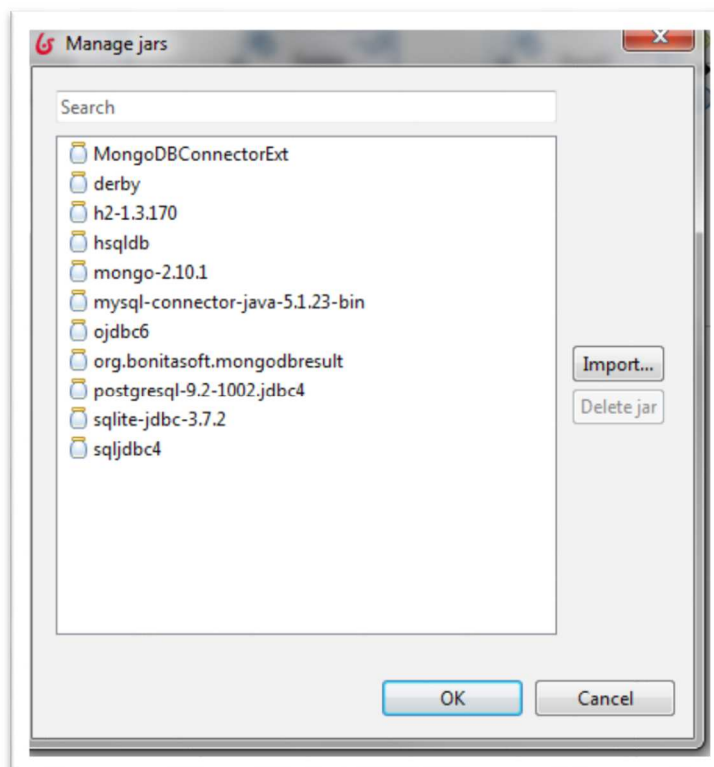
Go to the “Development” menu and click on “Manage jars...” item.



The “Manager jars” dialog appears and shows all the JAR files already imported. Click on the “Import...” button, select the “MongoDBConnectorExt.jar” file in your file system and press “Open”.

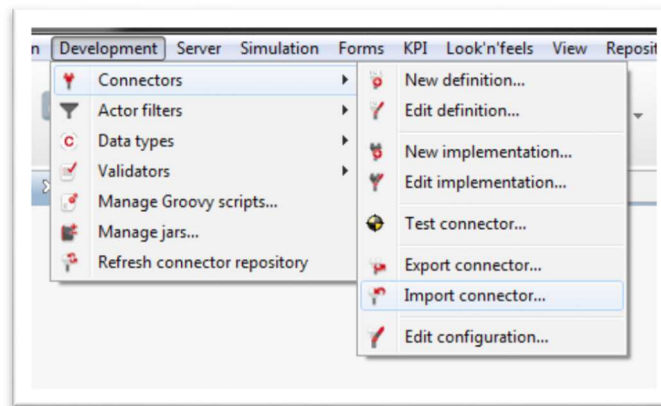


After some computation, you see that the new imported JAR file is now present in the list of the “Manage jars” dialog. Press “OK” on this dialog to close it.

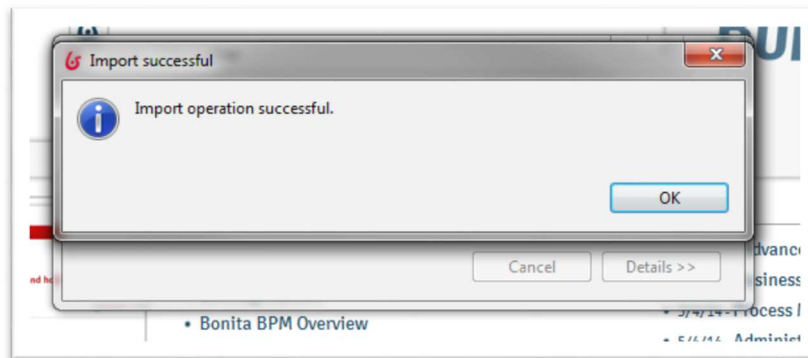


Import the Mongo DB Connector as a connector

Go to the “Development” menu, select “Connectors” and the “Import connectors...” item.



From the browser that appears, select the “MongoDBConnector-impl-1.0.0.zip” file and press “Open”. After some computation, a new dialog appears saying the importation is a success.



Step 2: Define the Business Process

This step purpose is to create a new diagram and to define the business process where the Mongo DB Connector is used.

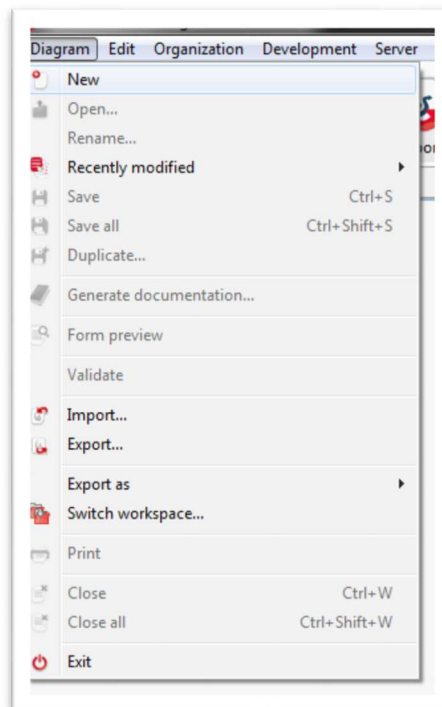
1. Main steps

- ✓ Create a new diagram.
- ✓ Define the flow of the business process.
- ✓ Add data to the business process.
- ✓ Set the form to summarize the case at the end.
- ✓ Add the Mongo DB Connector instances on the flow.

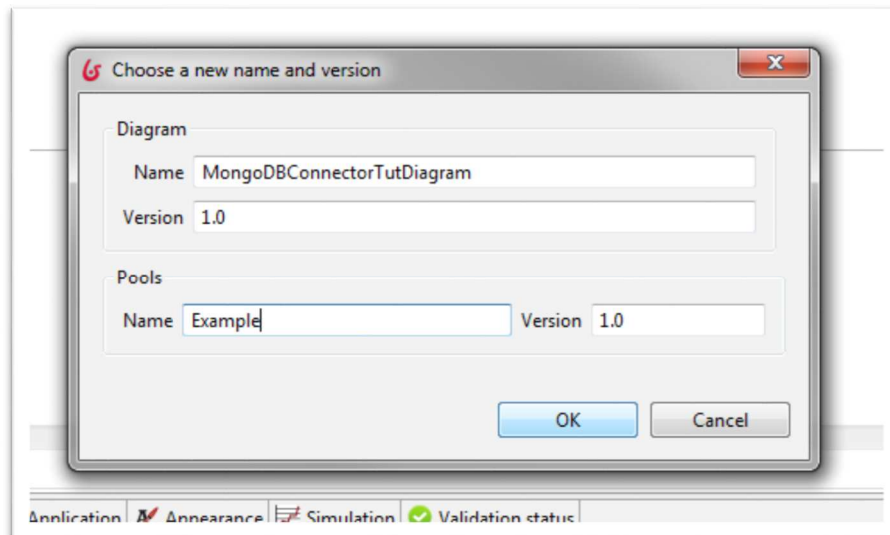
2. Instructions

Create a new Diagram

To create a new diagram, click on the “Diagram” menu and click on “New”.



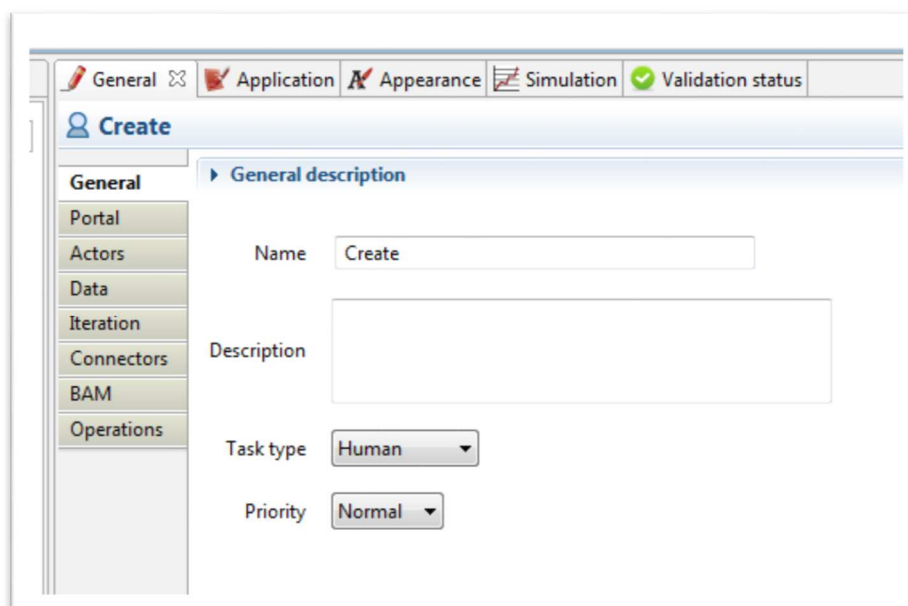
Now the diagram is created, click on a blank part in the white board of the diagram, go to the “Diagram” tab of the “General” properties tab and click on the “Edit...” button.



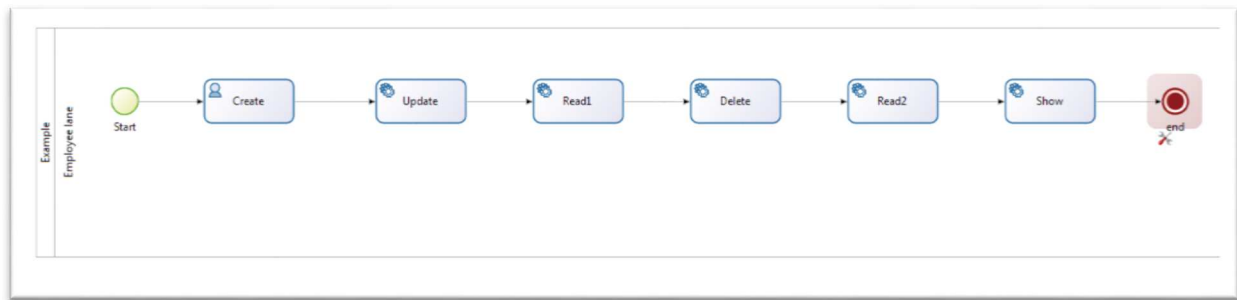
Now the “Choose a new name and version” dialog is shown, set the name and version of the diagram and the business process (the pool) you wish: in this tutorial we use “MongoDBConnectorTutDiagram” and “Example” as diagram and business process names and “1.0” as version.

Define the flow of the Business Process

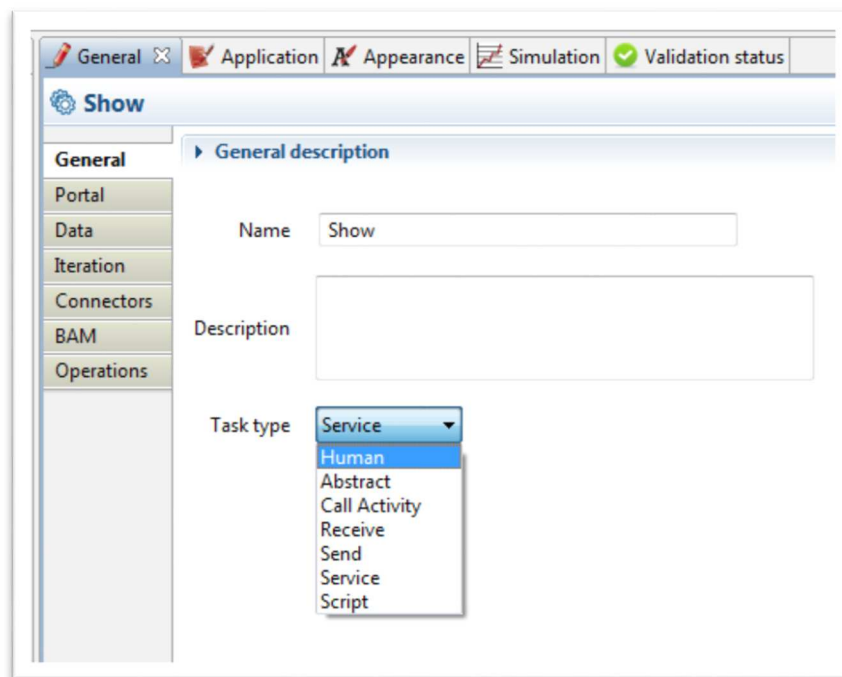
To define the flow, we can start by editing the name of the start event to “Start” and the existing step to “Create”. For that, click on the element on the white board and edit the “Name” field of the element in the “General” tab of the “General” properties tab.



Once done, build the end of the flow adding tasks and an end event. Each time an element has to be added following an element in the flow, click on this element, drag and drop the circled square (if a task has to be added) or the circle (if an event has to be added) and select the specific element if required.



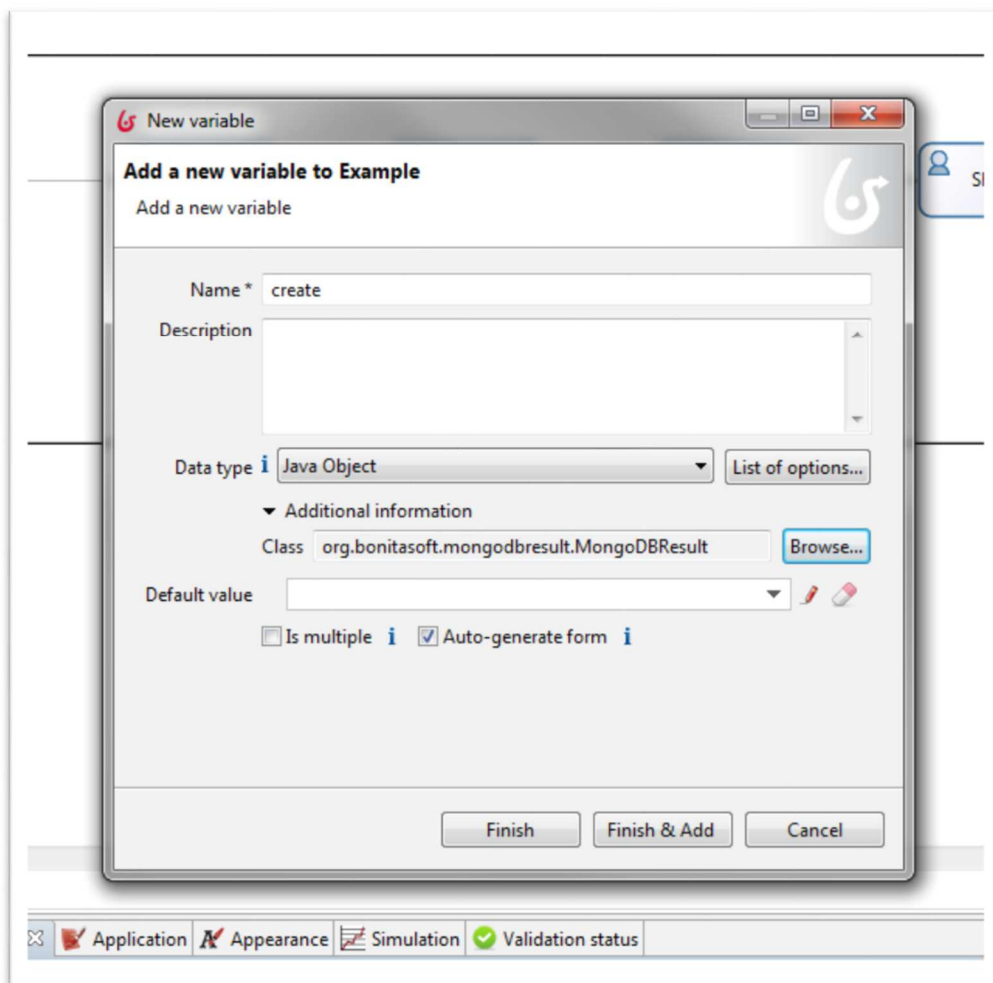
By default, the added tasks are “Service Task” but the “Show” task should be a “Human Task”. To change it, select the element, go to the “General” tab of the “General” properties tab and edit the “Task type” to “Human”. In the same way, the “Create” task should be a “Service Task”, then use the same trick to edit it in the right way.



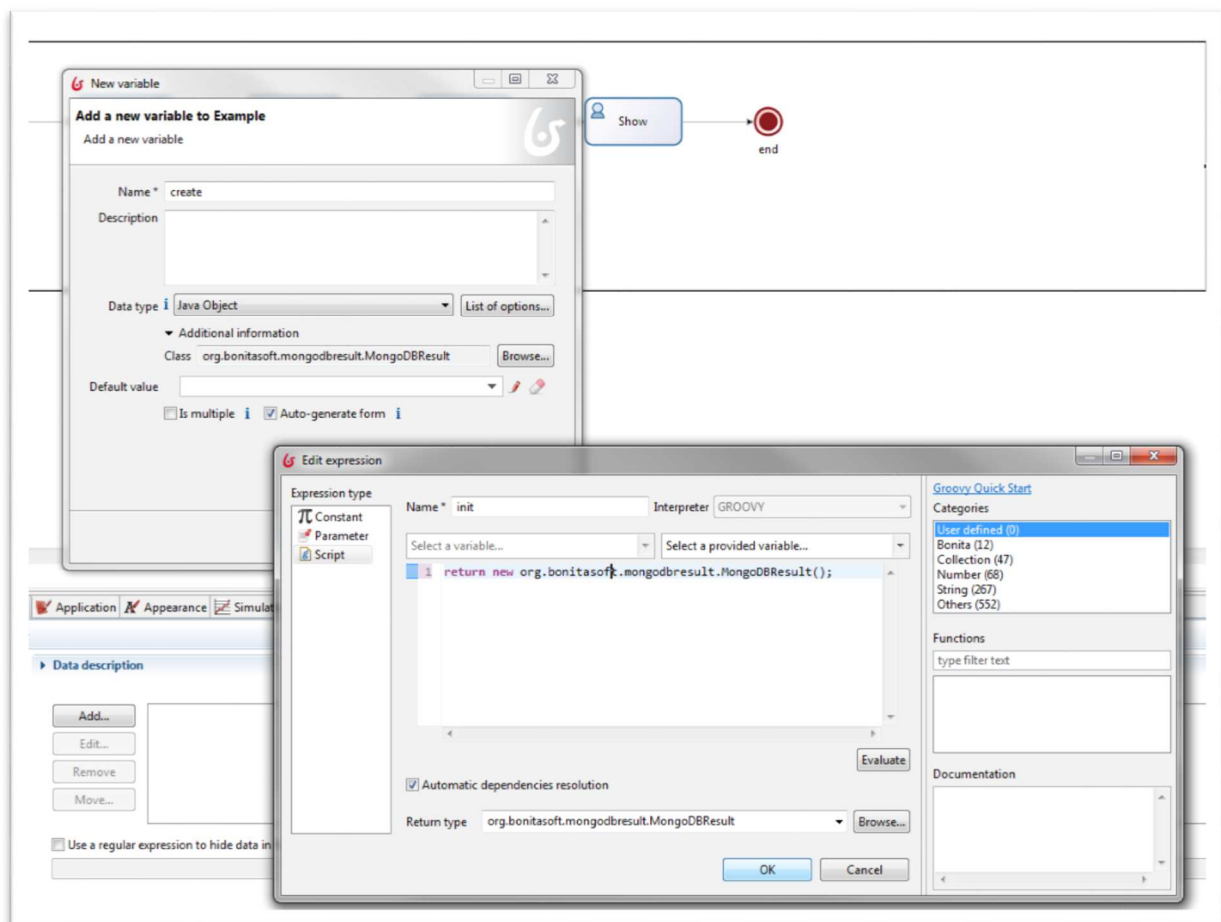
Add data to the business process

To be able to save the result of each Mongo DB Connector request, we define data on the business process scope. To do so, click on the business process, go to the “Data” tab of the “General” properties tab. From this panel, it is possible to add data clicking on “Add...”.

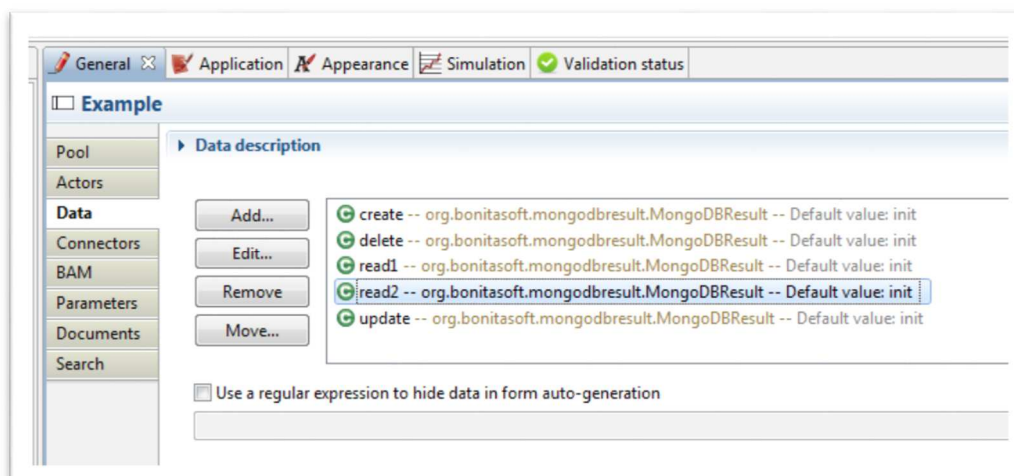
A “New variable” dialog appears and allows to set the new data options. Start by putting the name, “create” for the first one, set the “Data type” to “Java Object” and click on the “Browse...” button and select “MongoDBResult” in the new dialog.



The default value should also be set then click on the “Edit” icon, select “Script” on the left panel of the “Edit expression” dialog, enter “init” as a name, write “return new org.bonitasoft.mongodbresult.MongoDBResult();” in the script content and click either on “Finish” to finish and exit the dialog or on “Finish & Add” to finish and add start to add another data.

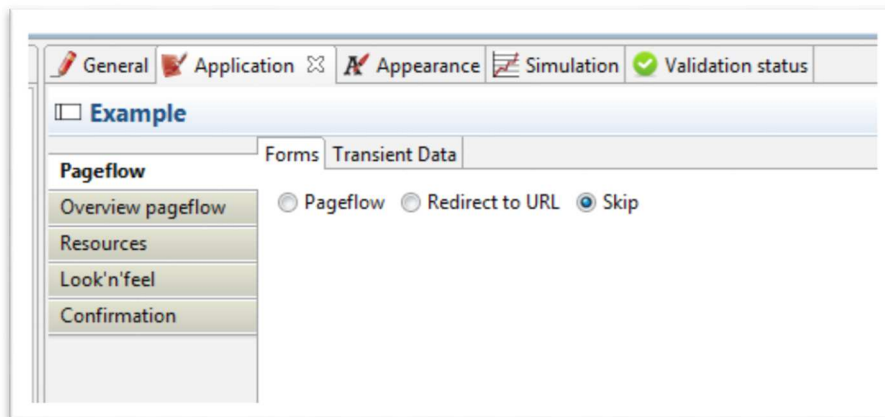


Do it each time as many times as required to get the five variables required to get the results.

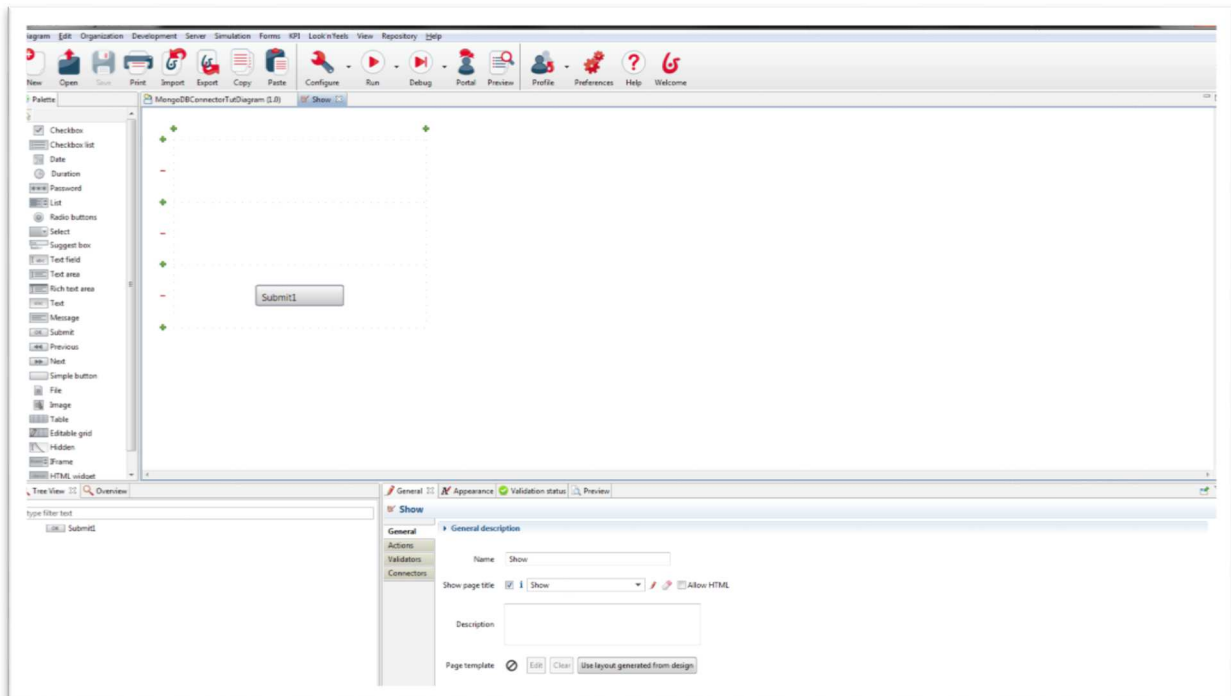


Set the form to summarize the case at the end

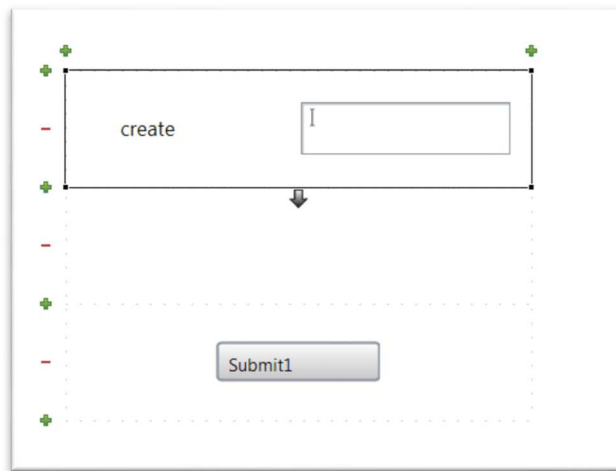
Click on the business process and go to “Pageflow” tab of the “Application” properties tab and select the option button named “Skip” instead of “Pageflow”.



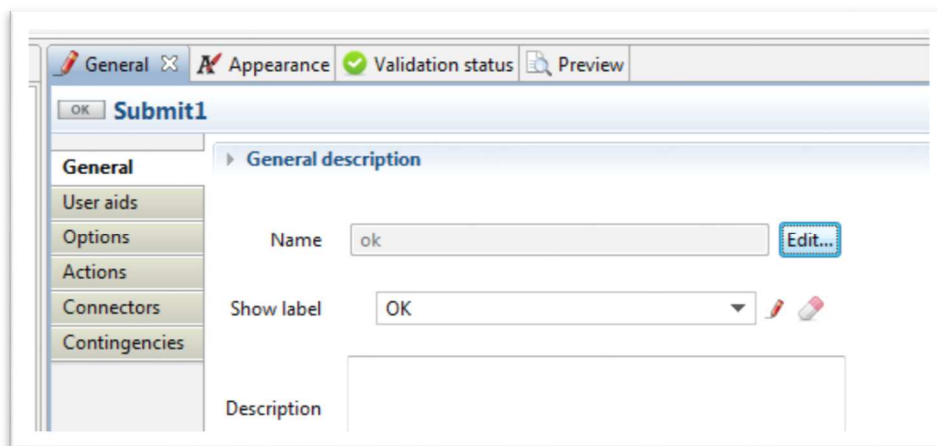
Now we want to add a form in the last task of the business process to be able to summarize the case behavior before finishing it. For that, select the “Show” task, got to the “Pageflow” tab of the “Application” properties tab and click on the “Add...” button. On the “Add form...” dialog, click on “Finish”.



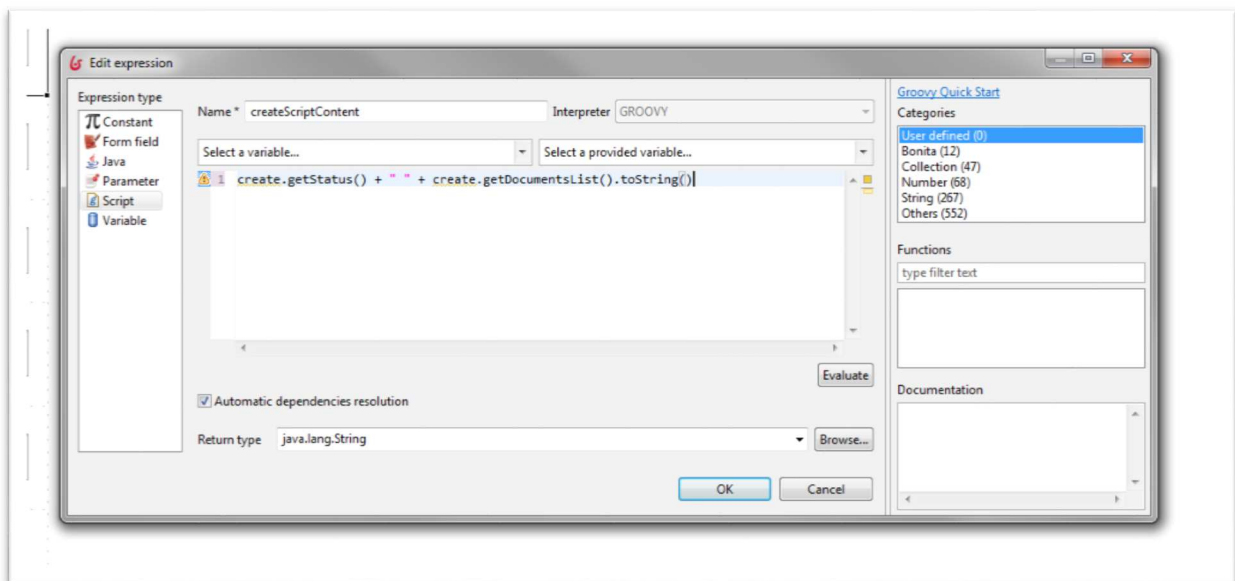
Then, add a “Text area” widget for each data we want to show (“create”, “update”, “read1”, “delete”, “read2”). For that, drag and drop the “Text area” named element from the “Palette” tab on the left side to a location on the form on the white board and edit the “Name” and the “Show label” to match the data name in the “General” tab of the “General” properties tab.



The “Submit” button can also be edited to become “OK” using the same trick.



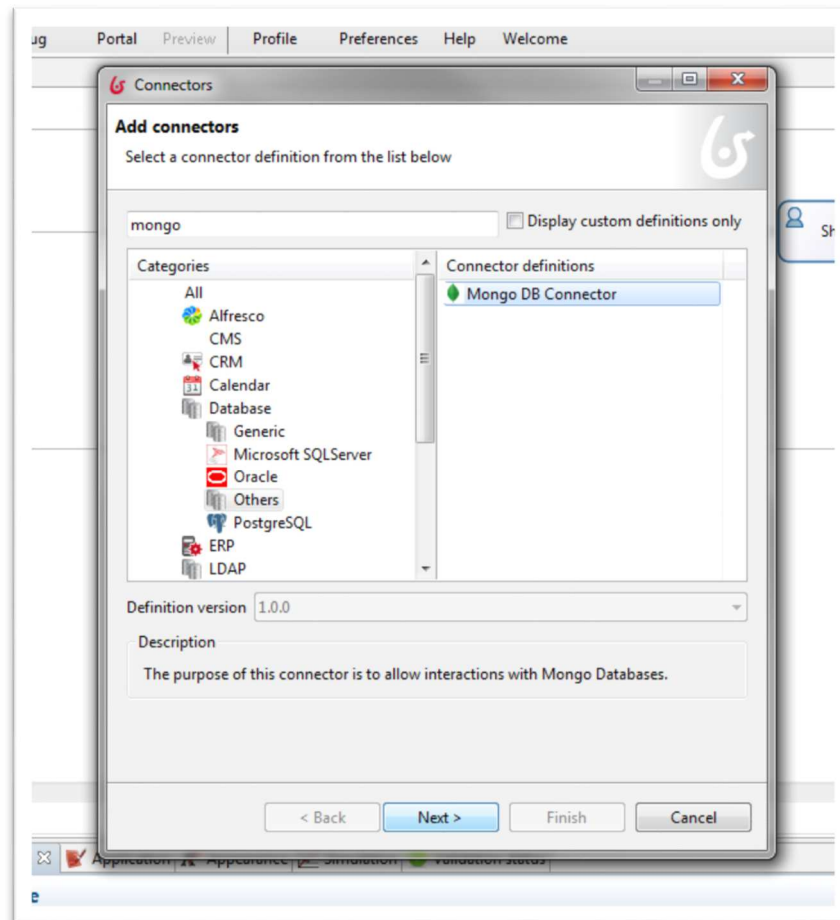
The last thing to set on this form is to set the initial data of the widgets to the result returned during the case execution. Click on the “Text area” widget, go to the “Data” tab of the “General” properties tab and click on the “edit” icon. The “Edit expression” dialog appears, select the “Script” tab on the left part, enter the name of the script as “xxxScriptContent” and enter the content for the script “xxx.getStatus() + " " + xxx.getDocumentsList().toString()” with “xxx” the name of the data to be shown.



Add the Mongo DB Connector instances on the flow

Now the whole business process definition (flow and form) is done, it is required to add the connector instances on the flow. Before being able to do it, it is necessary to have a Mongo DB server set with a collection and an admin user with all rights on it (read & write). In our case we have a 2.10 Mongo version (we are not talking about the Mongo DB version here), a localhost server IP, a 27017 serve port, an “admin” DB name, a “fulladmin” username, a “pwd” password for the “fulladmin” username, an “user” collection in the “admin” DB, full rights on the “admin” DB and especially for the “user” collection for the “fulladmin” username and one of the “user” collection fields is “name”. Following illustrations are based on this context described, either mount the same environment as ours or edit the Mongo DB Connector requests according to your own context.

For each “Service Task”, select the task on the whiteboard, go to the “Connectors” tab of the “General” properties tab and click on “Add...” button. On the “Connectors” dialog, select the “Mongo DB Connector” and click on “Next” button.



Enter the name of the data as connector “Name” and click on “Next” button twice.

Mongo DB Connector (1.0.0)

General
Specify the general information

Name * create

Description

Select event *

enter finish

If connector fails... Put in failed state

Named error

< Back Next > Finish Cancel

The first page of the wizard of the Mongo DB Connector is now reached, enter the server information and click on “Next” button.

Mongo DB Connector (1.0.0)

Mongo Connection Wizard Page
This page sets the Mongo connection.

Server IP Address * 127.0.0.1

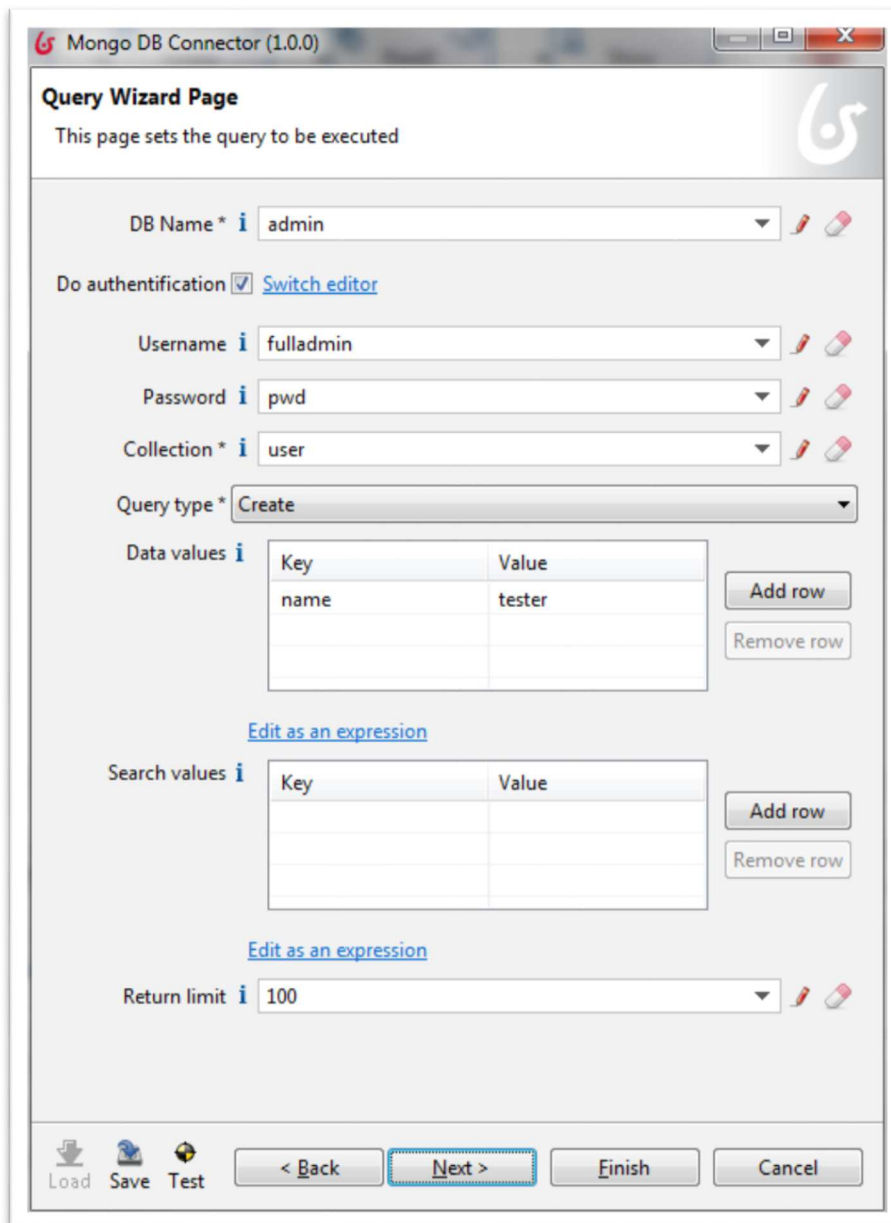
Server Port * 27017

The Mongo version is 2.10 or later ☒ [Switch editor](#)

Load Save Test < Back Next > Finish Cancel




Application Appearance Simulation Validation status

The second page of the wizard of the Mongo DB Connector is now opened, enter the request information and click on “Next” button.









Mongo DB Connector (1.0.0)




Query Wizard Page
This page sets the query to be executed

DB Name *   


Do authentication ☒ [Switch editor](#)

Username   

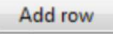

Password   

Collection *   


Query type *

Data values 

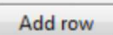
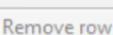
Key	Value
name	tester




[Edit as an expression](#)




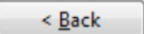
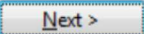
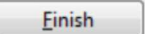
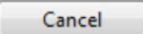
Search values 

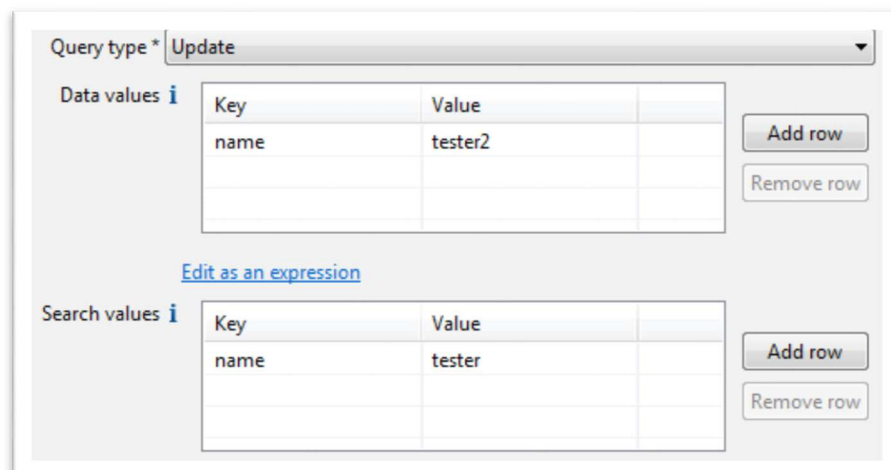
Key	Value


[Edit as an expression](#)

Return limit   

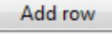

 Load
  Save
  Test
  < Back
  Next >
  Finish
  Cancel




Query type *

Data values 

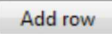
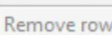
Key	Value
name	tester2

[Edit as an expression](#)

Search values 

Key	Value
name	tester

Query type * Read

Data values i

Key	Value

[Edit as an expression](#)

Search values i

Key	Value

Add row
Remove row

Query type * Delete

Data values i

Key	Value

[Edit as an expression](#)

Search values i

Key	Value
name	tester2

Add row
Remove row

Query type * Read

Data values i

Key	Value

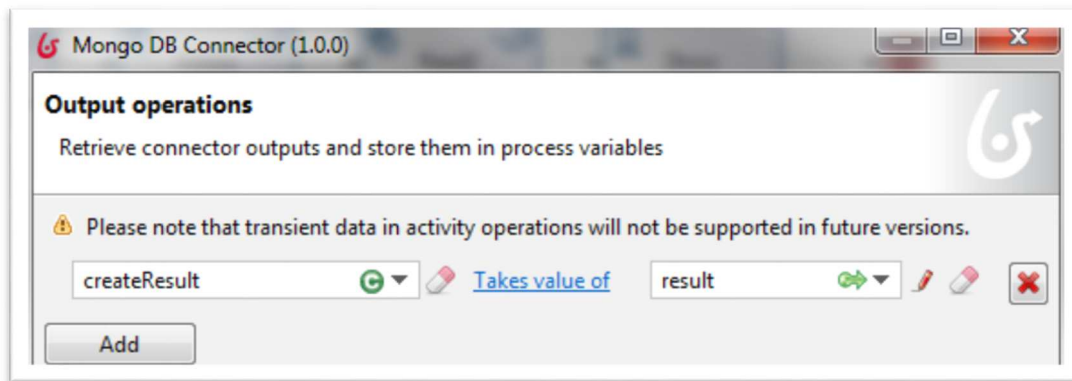
[Edit as an expression](#)

Search values i

Key	Value

Add row
Remove row

The result operations page is shown, write that the data of the process should be assigned to the value of the result of the request call of the Mongo DB Connector and click on “Finish” button.



Step 3: Launch a case

This last step target is to launch a case based on the process definition done before. Mongo DB Connector instances will be called.

1. Main steps

- ✓ Launch the Mongo DB server
- ✓ Launch a case from the Studio and show the result.

2. Instructions

Launch the Mongo DB server

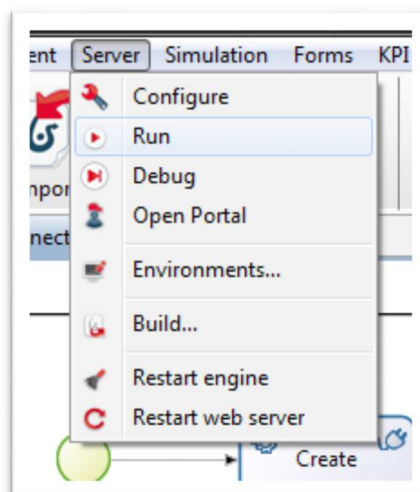
Before being able to launch a case, it is necessary to launch the Mongo DB server to be able to interact with it.

```

C:\Users\pierrick\Documents\Grenoble\mongodb-win32-x86_64-2008plus-2.4.9>.bin\mongod.exe --auth --rest --dbpath "C:\Users\pierrick\Documents\Grenoble\mongodb-win32-x86_64-2008plus-2.4.9\db"
Mon Jun 23 12:09:59.845 [initandlisten] MongoDB starting : pid=67580 port=27017 dbpath=C:\Users\pierrick\Documents\Grenoble\mongodb-win32-x86_64-2008plus-2.4.9\db 64-bit host=Julio8_BS_PC
Mon Jun 23 12:09:59.846 [initandlisten] db version v2.4.9
Mon Jun 23 12:09:59.846 [initandlisten] git version: 52fe0d21959e32a5dbecdc62057db386e4e029c
Mon Jun 23 12:09:59.846 [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1_49
Mon Jun 23 12:09:59.847 [initandlisten] allocator: system
Mon Jun 23 12:09:59.847 [initandlisten] options: { auth: true, dbpath: "C:\Users\pierrick\Documents\Grenoble\mongodb-win32-x86_64-2008plus-2.4.9\db", rest: true }
Mon Jun 23 12:09:59.899 [initandlisten] journal dir=C:\Users\pierrick\Documents\Grenoble\mongodb-win32-x86_64-2008plus-2.4.9\db\journal
Mon Jun 23 12:09:59.912 [initandlisten] recover begin
Mon Jun 23 12:09:59.915 [initandlisten] recover lsn: 3678564
Mon Jun 23 12:09:59.915 [initandlisten] recover C:\Users\pierrick\Documents\Grenoble\mongodb-win32-x86_64-2008plus-2.4.9\db\journal\j_0
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:0 < lsn:3678564
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:114390 < lsn:3678564
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:230621 < lsn:3678564
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:518402 < lsn:3678564
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:2380513 < lsn:3678564
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:2437924 < lsn:3678564
Mon Jun 23 12:09:59.995 [initandlisten] recover skipping application of section seq:2496636 < lsn:3678564
Mon Jun 23 12:09:59.996 [initandlisten] recover skipping application of section seq:2613796 < lsn:3678564
Mon Jun 23 12:09:59.996 [initandlisten] recover skipping application of section seq:3020218 < lsn:3678564
Mon Jun 23 12:10:00.007 [initandlisten] recover cleaning up
Mon Jun 23 12:10:00.008 [initandlisten] removedJournalFiles
Mon Jun 23 12:10:00.090 [initandlisten] recover done
Mon Jun 23 12:10:00.855 [initandlisten] waiting for connections on port 27017
Mon Jun 23 12:10:00.855 [webvr] admin web console waiting for connections on port 28017
  
```

Launch a case from the Studio and show the result

Once done, everything is set. We can start a case. For that, select the process on the whiteboard, click on the “Run” item of the “Server” menu.



Your default browser is opening a new tab and after some computations, you should see that the “Show” task is available.

The information has been submitted.

A task is now available
[Show](#)

It means that every Mongo DB requests have been done: the “Show” task will show the results of them.

From the “Create” task, you should see that everything went right:

- For the “Create” task, nothing specific happens: 0 code.
- For the “Update” task, one task has been edited: 1 code.
- For the “Read1” task, only the updated user with the “tester2” name is returned.
- For the “Delete” task, one task has been deleted: 1 code.
- For the “Read2” task, no hit is returned.

Show

create

0 null

update

1 null

read1

~1 [{"_id": {"\$oid": "53a890b2fc38851973d332c5"}, "name": "tester2"}]

delete

1 null

read2

~1 []

OK