Sumit Vadaviya

Camunda BPM Expert, Corporate Trainer

Email : sumit.vadaviya@trainosoft.com

I

n this session we will go through script task.

**Why should I use scripts?**

With the extended scripting support it is easy to prototype complete processes including the business logic. Also the sharing of such prototypes is simple, because the whole code can be embedded inside the process. So only one file has to be exchanged and discussed. In combination with the BPMN model API it allows you to create mock processes with business logic directly inside your code.

Script task

Another advantage is that now also non-Java programmers can use camunda BPM to execute their processes without writing Java code. Or if you already have an non-Java code base you can now use it inside camunda BPM without porting it to Java.

We also allow you to reuse scripts by using the camunda:resource attribute. Which loads scripts from external resources. Either the scripts can be part of your deployment or reside in the classpath of your application.

**Script support in BPMN**

The following table provides an overview of the BPMN elements which support the execution of scripts.

|  |  |
| --- | --- |
| **BPMN Element** | **Script Support** |
| Script Task | Script inside the script task |
| Processes, Activities, Sequence Flows, Gateways and Events | Script as an execution listener |
| [User Tasks](https://docs.camunda.org/manual/7.7/user-guide/process-engine/scripting/#use-scripts-as-task-listeners) | Script as task listener |
| Sequence Flows | Script as condition expression of a sequence flow |
| All Tasks, All Events, Transactions, Subprocesses and Connectors | Script inside an inputOutput parameter mapping |

**Script Task**

A Script Task is an automated activity. When a process execution arrives at the Script Task, the corresponding script is executed.



A Script Task is defined by specifying the script and the Script Format.

Supported script formats are Groovy, JavaScript, JRuby and Jython.

All process variables can be accessible through the execution.

**execution.getVariable("variable name");**

It’s also possible to set process variables in a script.

**execution.setVariable("variable name","value");**

**Information can be accessed through execution’s default variable.**

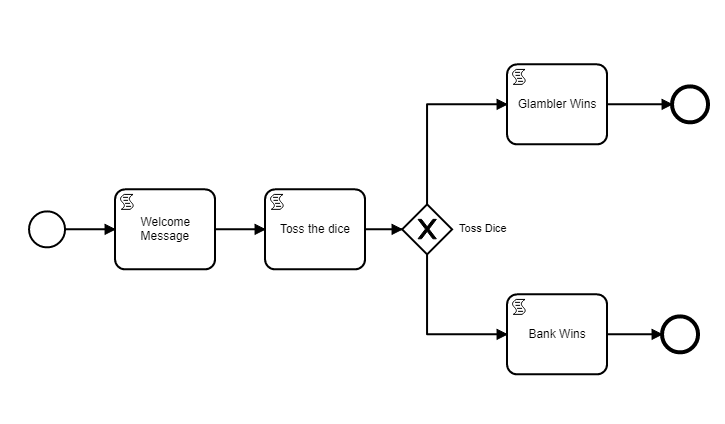
execution.eventName

execution.activityId

task.name

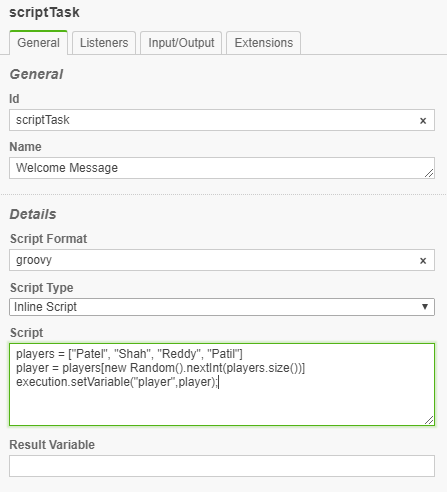
task.eventName

**Exercise:**

****

**Setting properties for script each task:**

**Script task:** Welcome Message



**Script Format:** groovy

**Script Type:** Inline Script

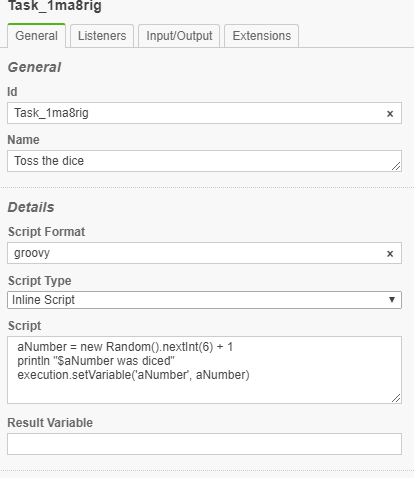
**Script:**

players = ["Patel", "Shah", "Reddy", "Patil"]

player = players[new Random().nextInt(players.size())]

execution.setVariable("player",player);

**Script task:** Toss the dice



**Script Format:** groovy

**Script Type:** Inline Script

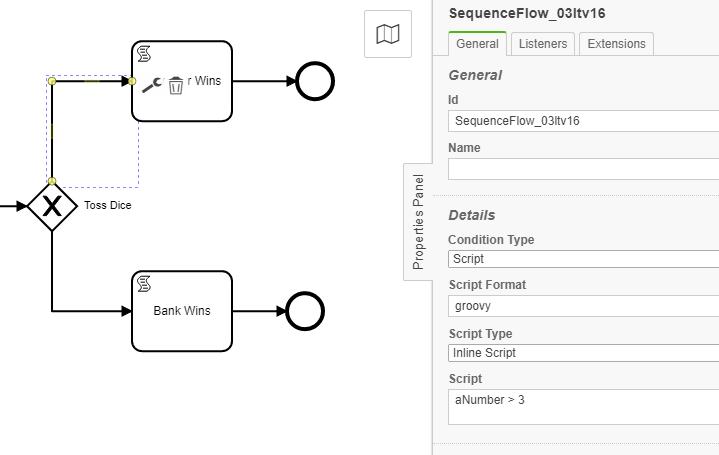
**Script:**

aNumber = new Random().nextInt(6) + 1

println "$aNumber was diced"

execution.setVariable('aNumber', aNumber)

**Sequence Flow 1:**

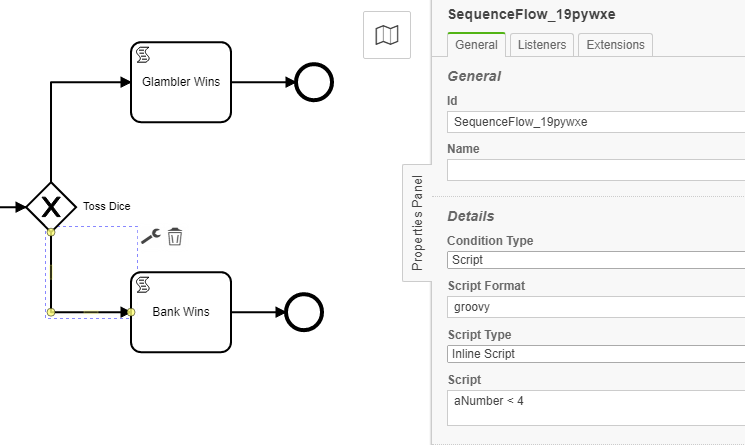


**Script Format:** groovy

**Script Type:** Inline Script

**Script:** aNumber > 3

**Sequence Flow 2:**

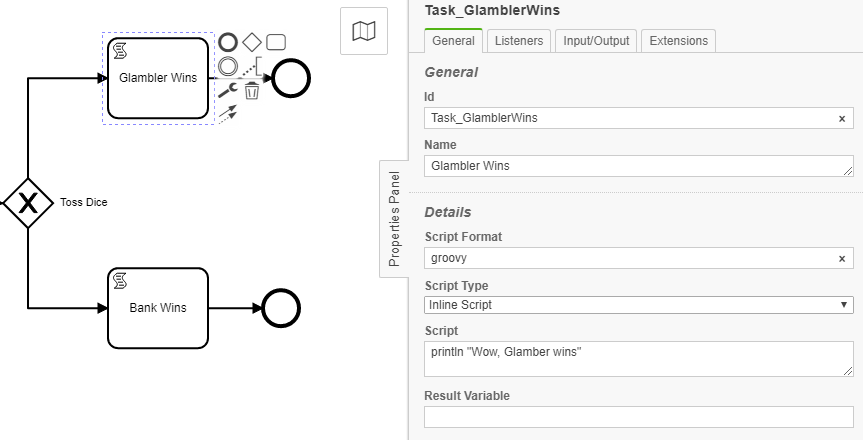


**Script Format:** groovy

**Script Type:** Inline Script

**Script:** aNumber < 3

**Script task:** Glambler Wins



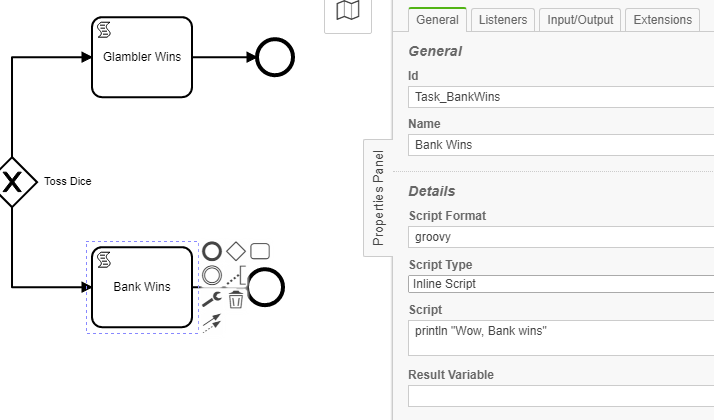
**Script Format:** groovy

**Script Type:** Inline Script

**Script:**

println "Wow, Glamber wins"

**Script task:** Bank Wins



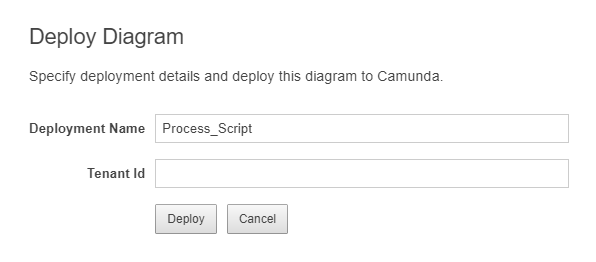
**Script Format:** groovy

**Script Type:** Inline Script

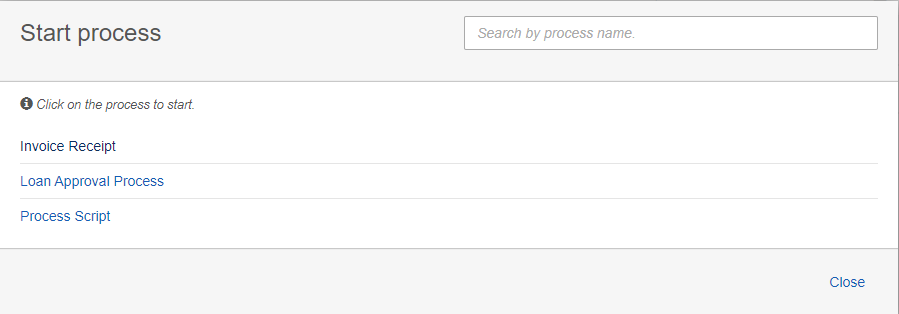
**Script:**

println "Wow, Bank wins"

**Deploy the process through Camunda modeler.**

****

**Start the deployed process multiple times from Camunda Tasklist application.**

****

**Check the console output:**

4 was diced

Wow, Glamber wins

4 was diced

Wow, Glamber wins

5 was diced

Wow, Glamber wins

2 was diced

Wow, Bank wins

3 was diced

Wow, Bank wins

6 was diced

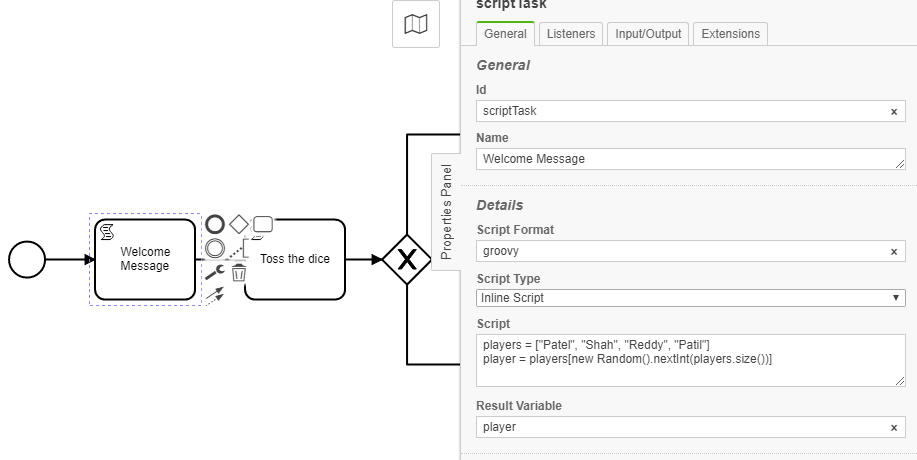
Wow, Glamber wins

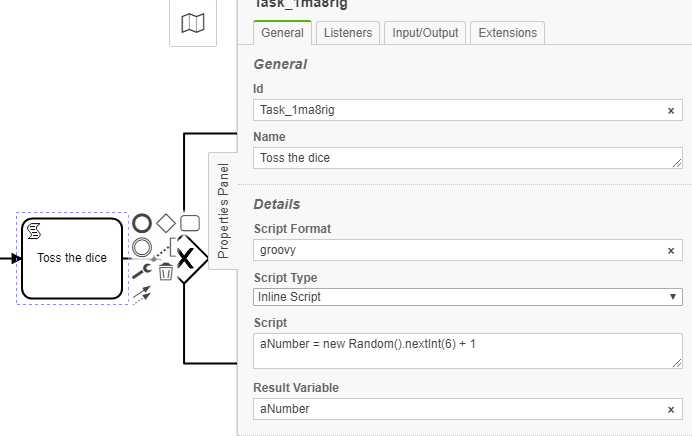
4 was diced

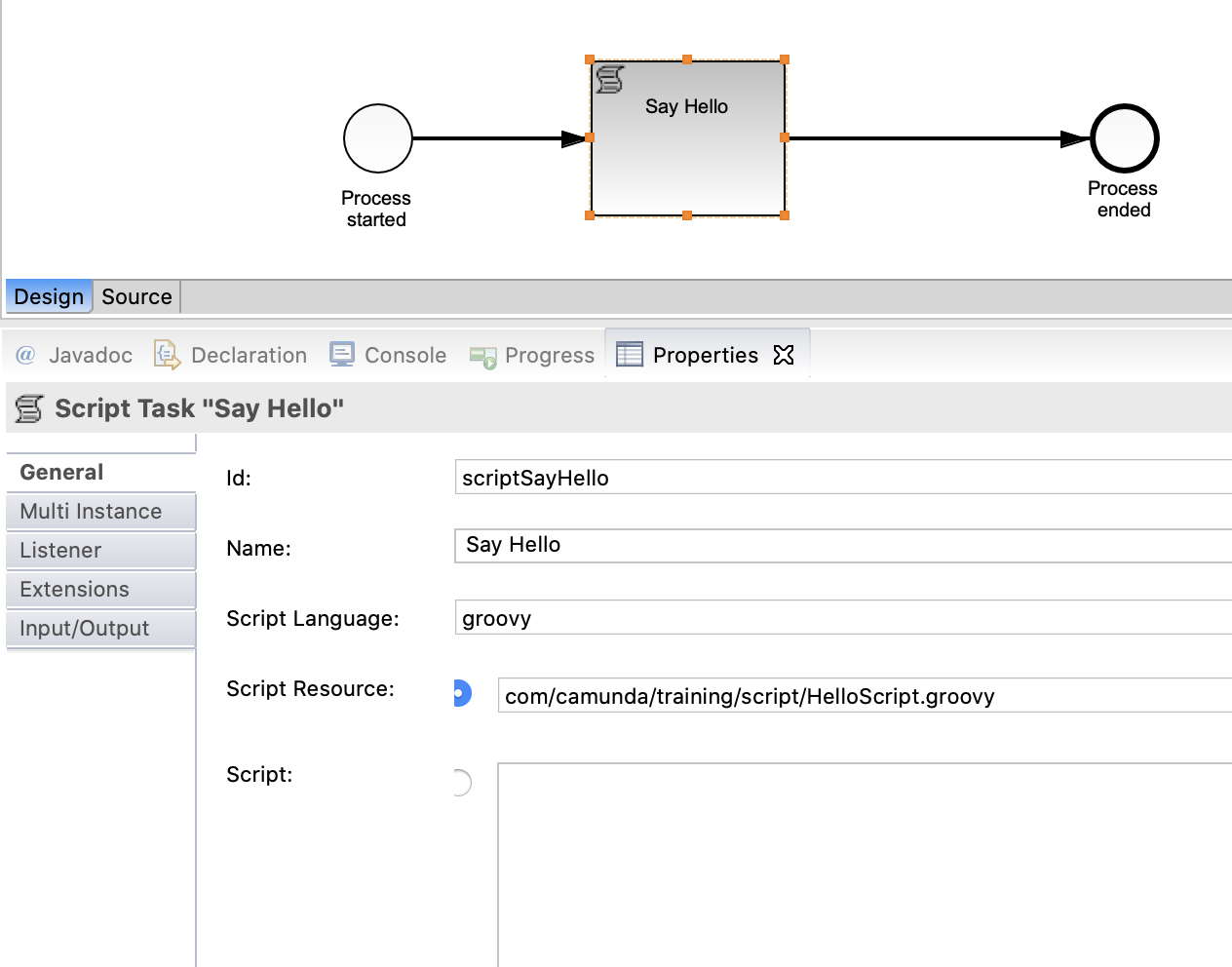
Wow, Glamber wins

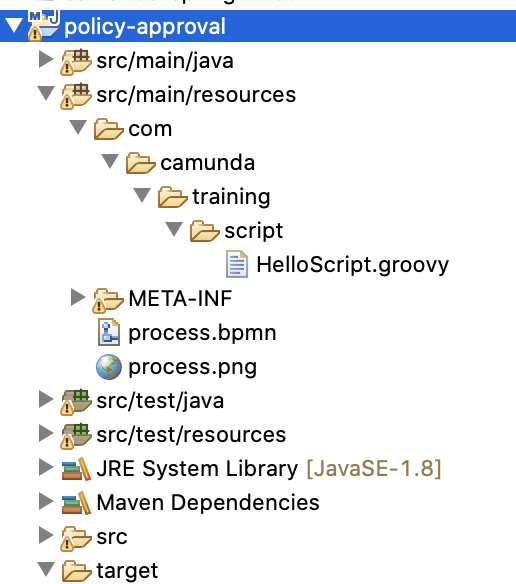
**Script Result**

The return value of a Script Task can be assigned to a previously existing or to a new process variable by specifying the process variable name as a literal value for the camunda:resultVariable attribute of a Script Task definition. Any existing value for a specific process variable will be overwritten by the result value of the script execution. When a result variable name is not specified, the script result value gets ignored.





So far we have used inline script. Now let’s use external script.

Create HelloScript.geoovy in resources/com/camunda/training/script directory.

**Where to uses script:**

Use Scripts as Execution Listeners

Use Script as Task listeners

Use Script as Conditions

Use Script as InputOutput parameters

**Variables Available During Script Execution**

During the execution of scripts, all process variables visible in the current scope are available. They can be accessed directly by the name of the variable (i.e., sum).

There are also special variables:

1. execution, which is always available if the script is executed in an execution scope (e.g., in a script task) ([DelegateExecution](https://docs.camunda.org/javadoc/camunda-bpm-platform/7.7/org/camunda/bpm/engine/delegate/DelegateExecution.html)).
2. task, which is available if the script is executed in a task scope (e.g., a task listener) ([DelegateTask](https://docs.camunda.org/javadoc/camunda-bpm-platform/7.7/org/camunda/bpm/engine/delegate/DelegateTask.html)).

**Execution Variables:**

execution.activityInstanceId

execution.businessKey

execution.currentActivityId

execution.currentActivityName

execution.currentTransitionId

execution.eventName

execution.id

execution.processBusinessKey

execution.processDefinitionId

execution.processInstanceId

execution.tenantId

**Execution methods:**

execution.setVariable(variableName, value);

execution.getVariable(variableName)

Script:

Script format: groovy

Script:

out.println "Execution instance id : "+execution.activityInstanceId

out.println "Business key : "+execution.businessKey

out.println "Current activity id : "+execution.currentActivityId

out.println "Current activity name : "+execution.currentActivityName

out.println "Current transition id : "+execution.currentTransitionId

out.println "Event name : "+execution.eventName

out.println "Execution id : "+execution.id

out.println "Process business key : "+execution.processBusinessKey

out.println "Process definition id : "+execution.processDefinitionId

out.println "Process instance id : "+execution.processInstanceId

out.println "Tenant id : "+execution.tenantId

**Task Variables:**

task.assignee

task.caseExecutionId

task.caseDefinitionId

task.caseInstanceId

task.caseDefinitionId

task.createTime

task.deleteReason

task.description

task.dueDate

task.eventName

task.executionId

task.followUpDate

task.id

task.name

task.owner

task.priority

task.processDefinitionId

task.processInstanceId

task.taskDefinitionKey

task.tenantId

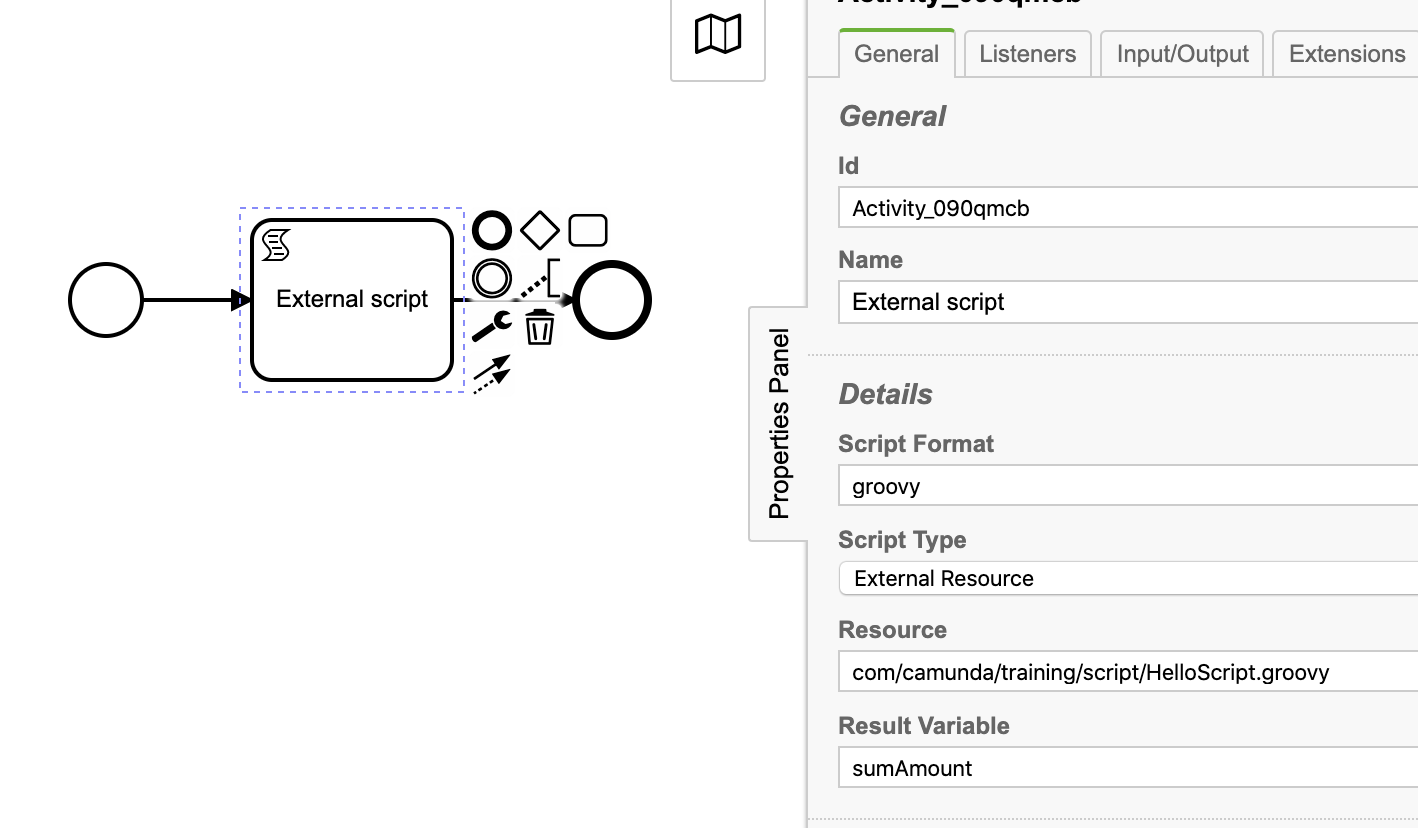
**Task methods:**

task.setVariable(variableName, value);

task.getVariable(variableName)

**External Script**

The script source code can also be loaded from an external resource.



Copy HelloScript.groovy to camunda-bpm-tomcat-7.14.0/server/apache-tomcat-9.0.36/webapps/camunda/WEB-INF/classes/com/camunda/script/HelloScript.groovy

**HelloScript.groovy**

println "External Script called";

number1 = execution.getVariable("number1");

number2 = execution.getVariable("number2");

totalAmount = number1 + number2