

A Developer Diary

{about:"code learn and share"}

[Home](#)[Data Science](#)[Java](#)[JavaScript](#)[jBPM](#)[Tools](#)[Tips](#)[About](#)

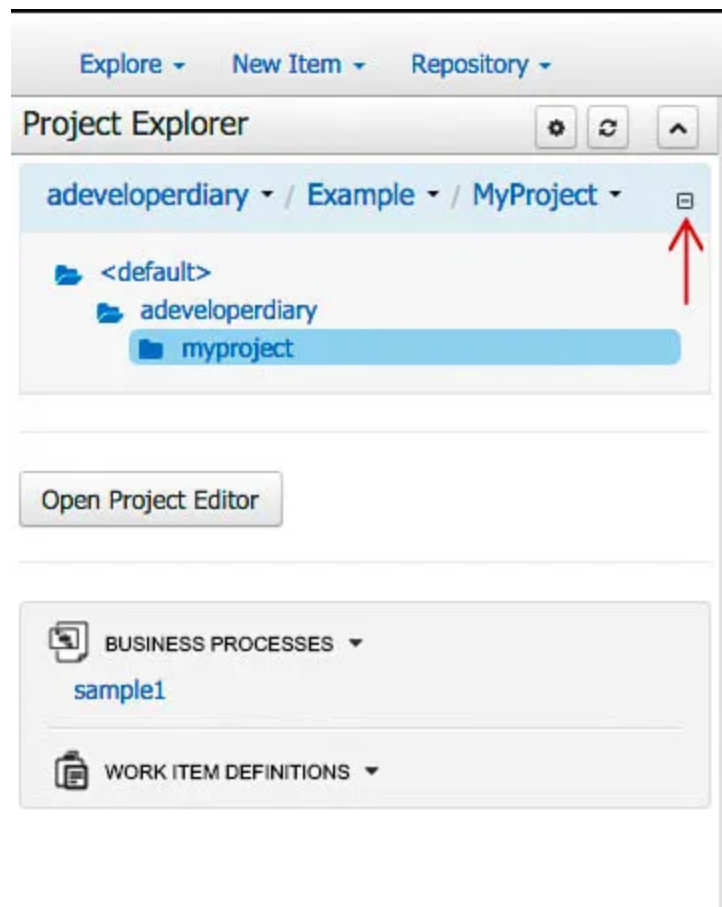
November 8, 2015 By [Abhisek Jana](#) — [7 Comments \(Edit\)](#)

Get started with jBPM KIE and Drools Workbench – Part 2



In this part of **Get started with jBPM KIE and Drools Workbench** we will create an Admission Process Business Workflow. We will use different tasks types and also define rules for our process.

Let's go and open the project we had created in Project Authoring. In case you are not seeing the Business Processes, there is a small plus (+) icon beside **MyProject** in the left side panel in **Project Explorer**, click on that, you can see the **Group ID** and **Artifact ID**, default -> **adeveloperdiary -> myproject**. Once you click on myproject you can see all the project artifacts we have created below.



There are few components we need to understand before could jump into creating more complex process.

Task:

As you have already seen it, there are different types of tasks you can create in a Business Process. **User Task** and **Business Rule Task** are the most important ones. Most of the tasks will have an **Input Data Object** and **Output Data Object**. You need to also map the input and output Data Object.

User Task:

Using this task User can interact with the data model (View, Edit, Update).

Business Rule Task:

You can execute any business rules using this. We will see more example later.

Script Task:

You can execute any script using this task.

Data Object:

You need to have a data model to hold the data during any business process. The Data Object is for that. It's similar to a class you create in Java or .Net. You can create simple or complex Data Object.

Form:

Whenever using a User Task, we need to have a way to interact with the Data Object. Using forms, we can generate the view to display correct fields to the users. This is very much like the HTML forms.

Rules:

Each Business Rule Task must be associated with an executable rule. You can create rules in many different ways, e.g. DRL Rules, Guided Rule etc.

We will now build a simple College Admission Process, where students can enter their name and GPA Score, then Admin will approve or reject the application.

Create New Project:

Click on New Item and Create a New Project named Admission. Go ahead and click on MyProject on top, you will see the new project, named Admission is listed there, click on it to select Admission as the current project.

Create Data Object:

We will start with creating a Data Object. You can create it from the **New Item** Menu. Enter the name as **Student** and Package as **adeveloperdiary.admission**.

Create new Data Object ×

* Data Object

Student

Package

adeveloperdiary.admission ⬆⬇⬆

In the next screen, you can add the fields. We will create 3 fields, **name** (String), **gpa** (Float) and **eligible** (Boolean). Click on **Create** button to add them, once added all three, click on Save button on top toolbar.

Student.java - Data Obj... Save Delete Rename Copy Validate Latest Ver

Create new field

*Id

Insert a valid Java identifier

Label

Insert a label

*Type

☐ List

+ Create

adeveloperdiary.admission.Student

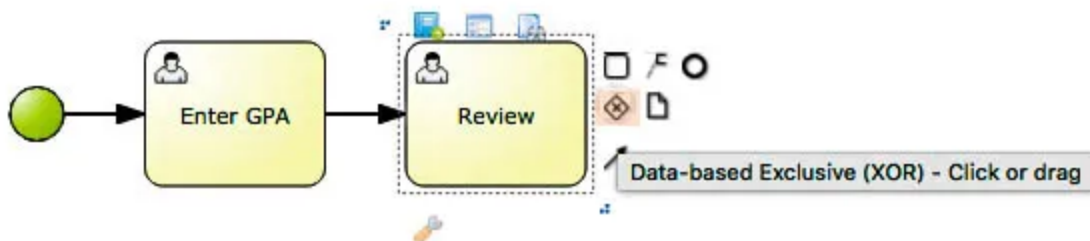
Identifier	Label	Type	
eligible		Boolean	×
gpa		Float	×
name		String	×

Create Business Process:

Create a new Business Process, named **Admission Process**, select the admission package as previous.

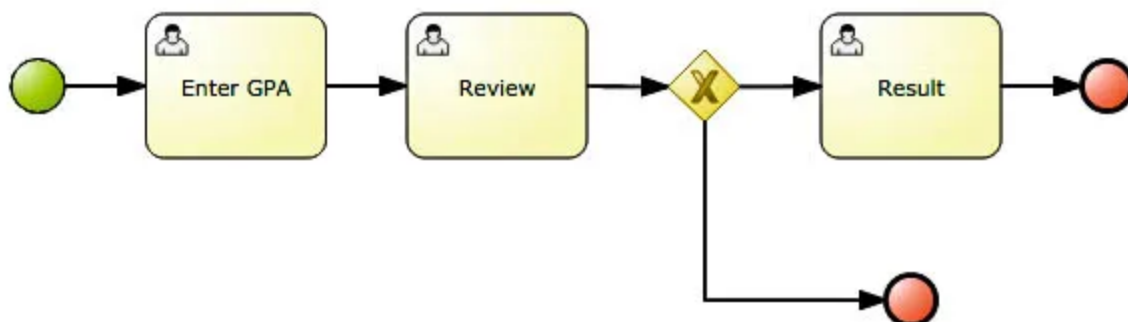
Note: Since we will be using only one user our Student and Admin will be the same user (user4).

Add Two User Tasks and then click on the **(X)** sign to add an if-else split logic.



This type of node can have more than one output. If the **Student** is not eligible for an Admission (based on eligible variable) then we will end the flow right away, otherwise we will display a message that he/she has been Admitted.

Our Business Process will look like the following:

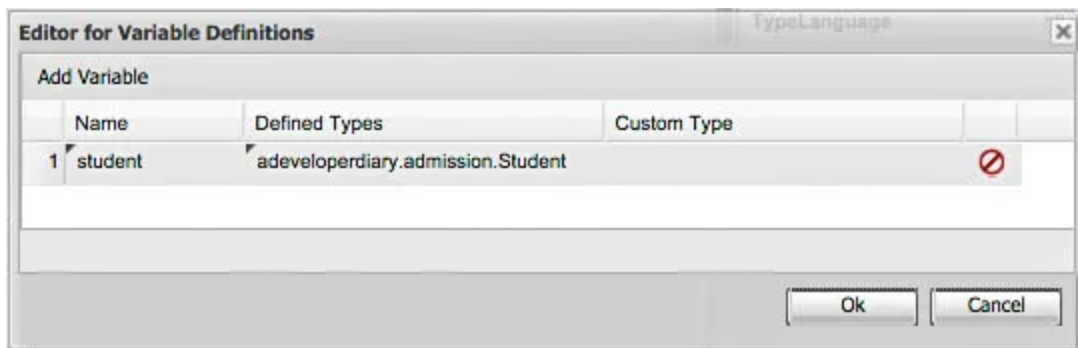


At first, let's add the Data Object we created with the business process. To do that, click anywhere in the white section of the process model and bring the

properties window.

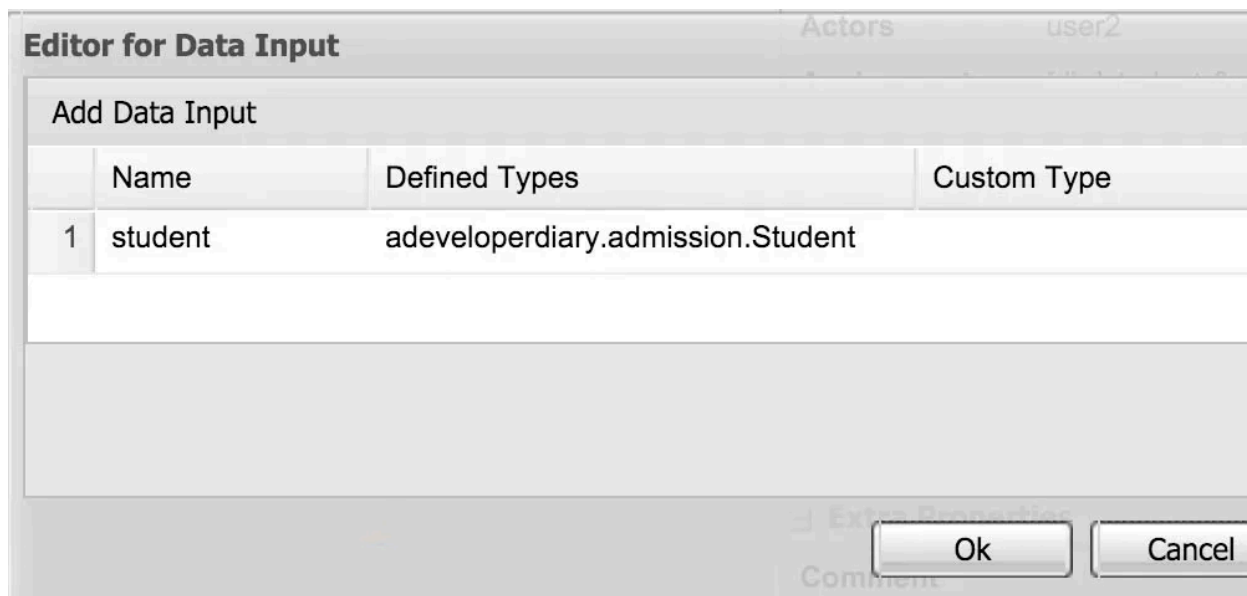
There will be a property named **Variable Definitions**, click on it and add the following variable definitions.

Name as **student** and **Defined Type** as **adeveloperdiary.admission.Student**. Basically we have now created an instance variable for the Student Object named student.

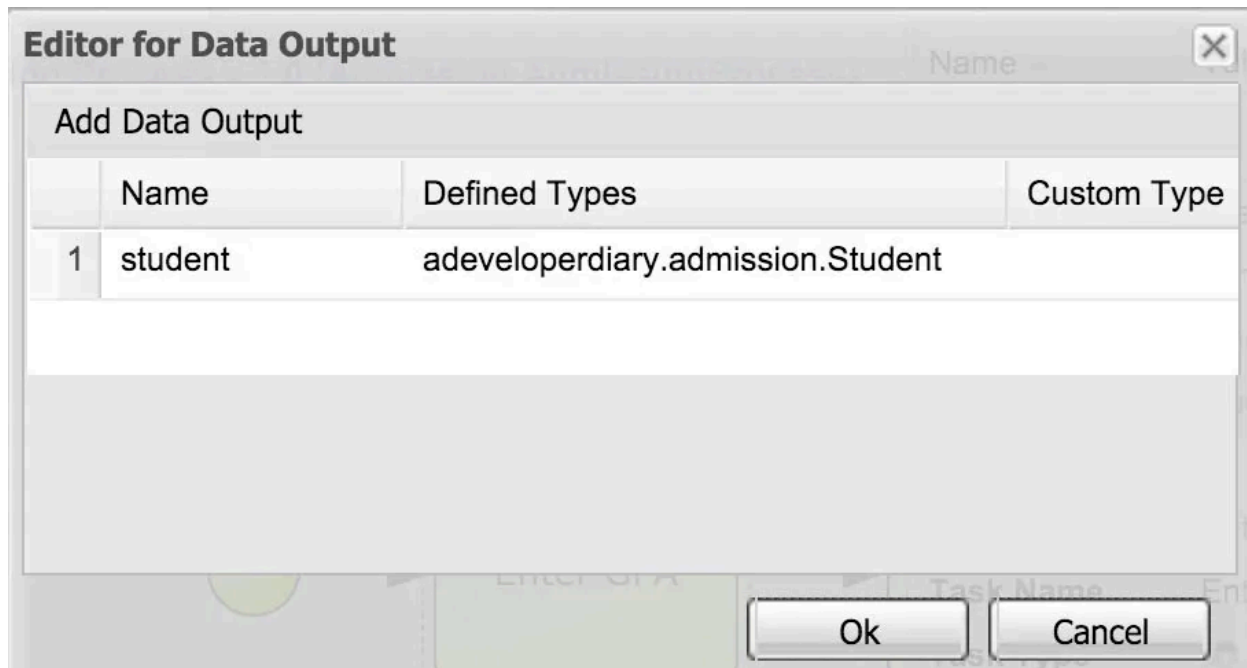


Now select Enter GPA task and add the **DataInputSet**, **DataOutputSet**, **Assignments** and the **Task Name**.

DataInputSet: Name as **student**, Defined Type as **adeveloperdiary.admission.Student**



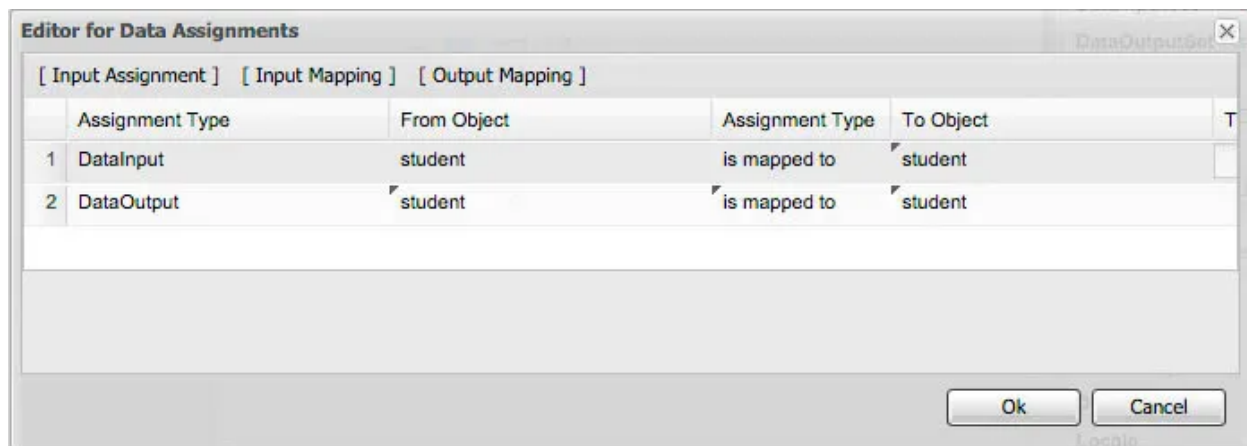
DataOutputSet: Name as **student**, Defined Type as **adeveloperdiary.admission.Student**



Assignments:

Click on [Input Mapping] and add From Object as student, Assignment Type as is Mapped to and To Object as student.


Then again do the same by clicking on [Output Mapping]



Also add the Task Name as Enter GPA.

What we have done here is, created a global variable as student then created two local variables for the Enter GPA Task named student. Then we have mapped them. So that when the task has been completed we will have our global variable updated.

The properties of the Enter GPA Task look like this.

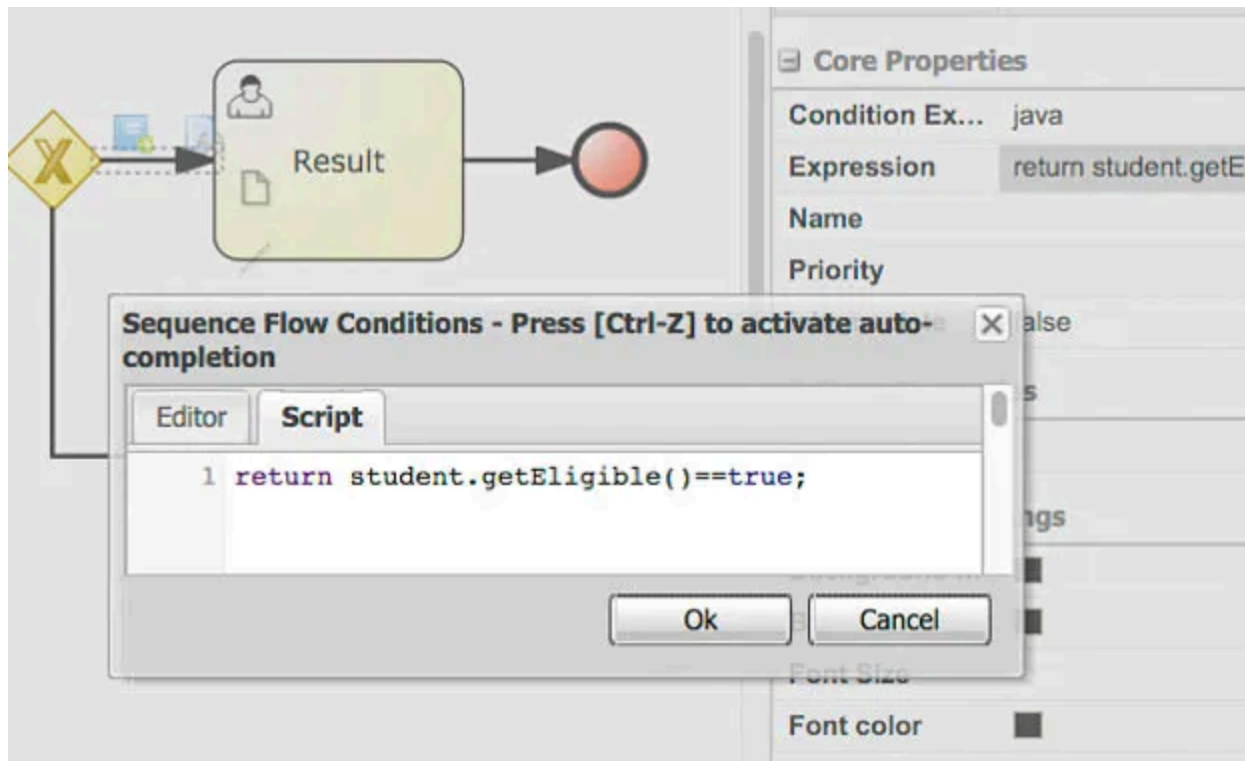
Properties (User)	
Name ▲	Value
Core Properties	
Actors	user4
Assignments	[din]student->student,[dout]student->student
DataInputSet	student:adeveloperdiary.admission.Student,Skippat
DataOutputSet	student:adeveloperdiary.admission.Student
Groups	
Name	Enter GPA
Task Name	EnterGPA
Task Type	 User

We will do the same with the Review and Result Task.

Now click on the Arrow between the **Result Task** and XOR (our If-else split). Open the properties and open the **Expression**, click on the **Script** Tab and enter the following.

```
return student.getEligible()==true;
```

See in the screenprint below, I have selected the Arrow coming to Result and entered the Expression. So if the `student.getEligible()` returns true then then this flow will be executed, like this we can also validate other data types as well.



Now do the same with the other end of the XOR step. Enter the following.

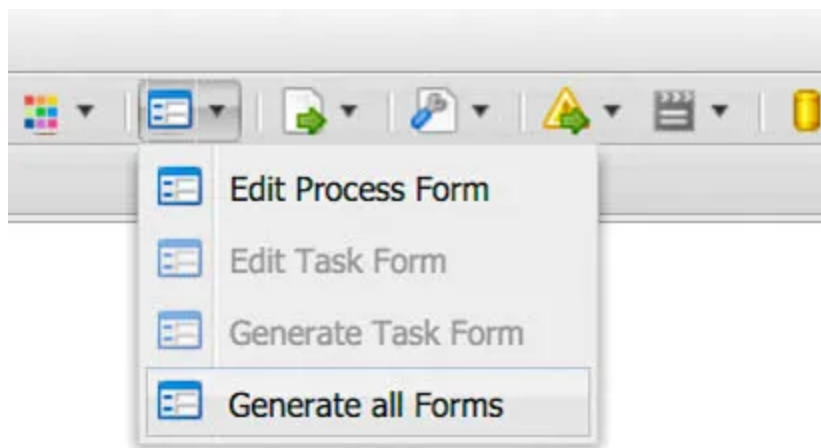
```
return student.getEligible()==false;
```

So we are validating the values of the eligible Boolean variable in student object. In case it's true, the Result task will be activated.

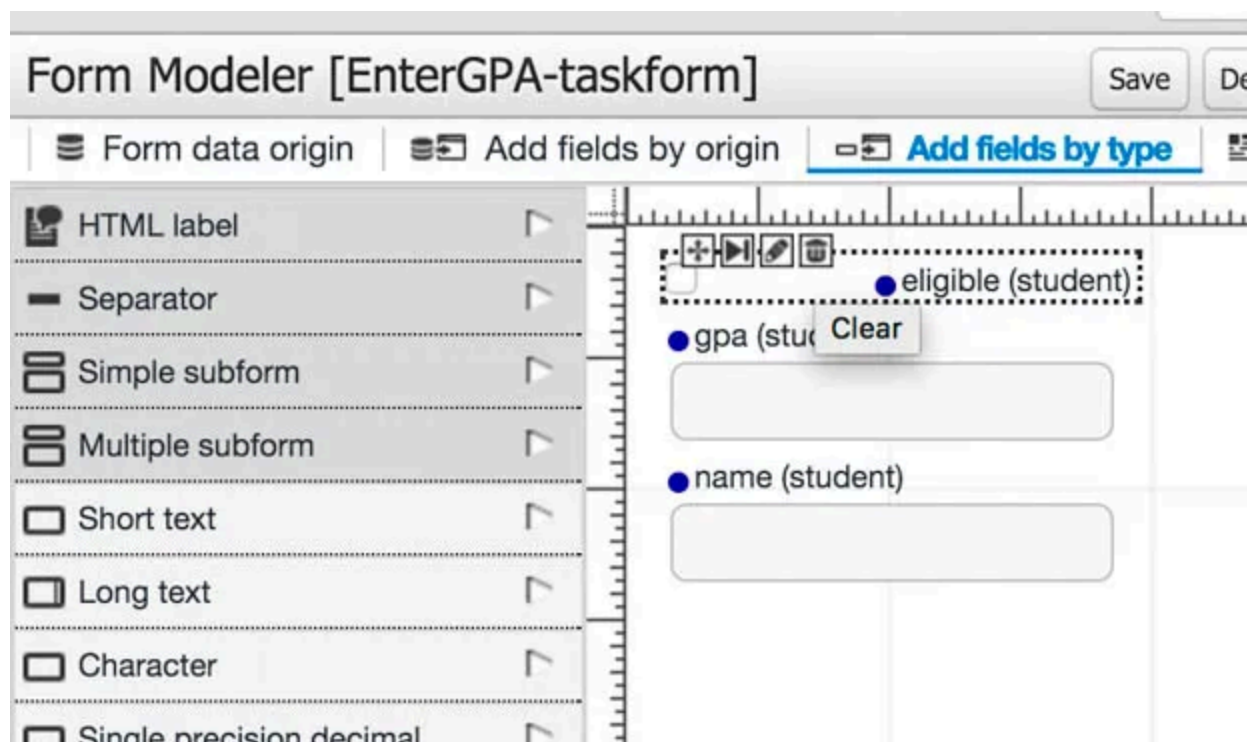
Create Forms:

The last thing we need to do is allow users to enter the data in form. Let's generate them first.

Click on the icon below and click on Generates all Forms.



Now Click on the form icon again and click on Edit Process Form. The following will be displayed. (Select **Graphical Modeler Options**)



Select the **eligible** (student) checkbox and delete it. Also delete the **gpa** text field. Save the form.

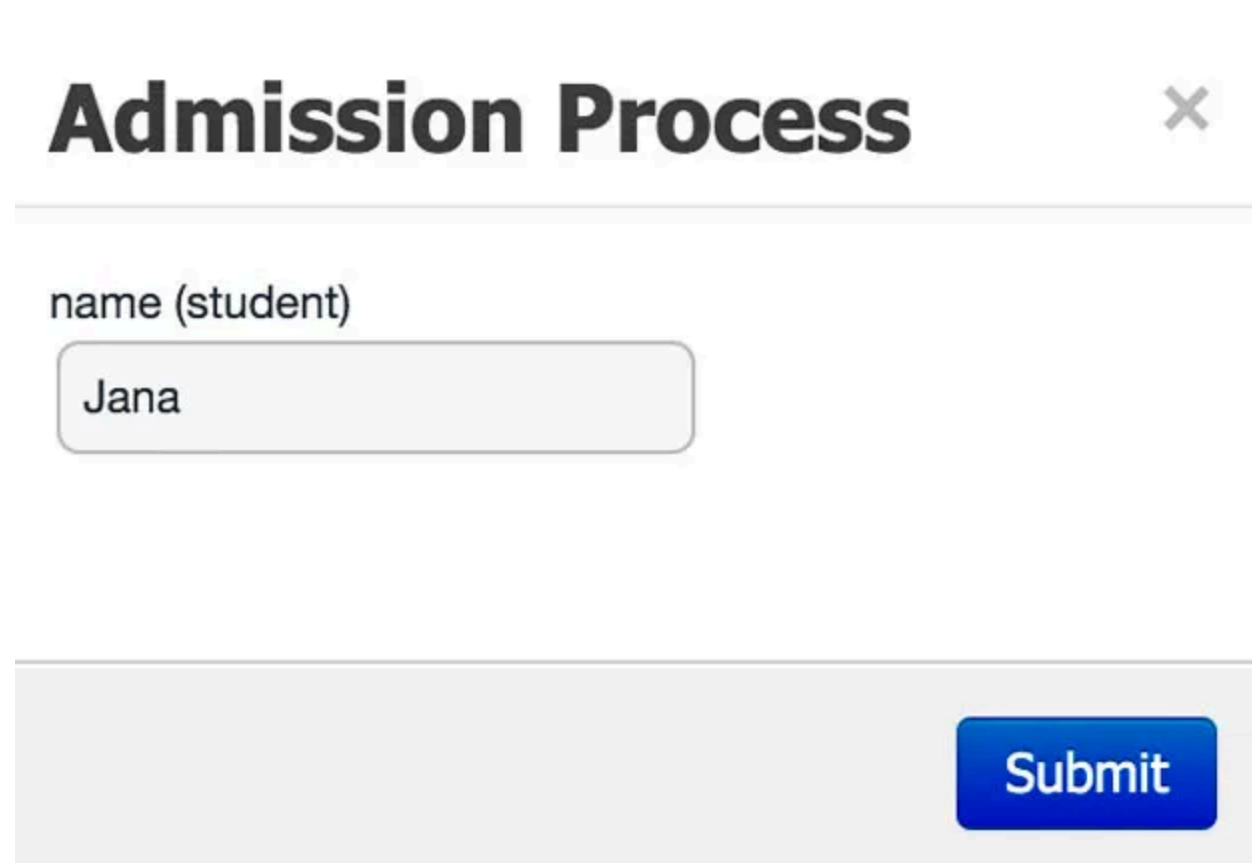
Now select the Enter GPA Task and click on the same form icon and click on Generate Task Form, then Edit Task Form. Select the eligible (student)

checkbox and delete it, since we don't want the Student to select it. Save the form.

Repeat the same for the **Review GPA** and **Result Task** and Save the form. In these two task we will have all the three fields displayed.

Execution:

Here on start the user will enter the name of the Student.



Admission Process ×

name (student)

Jana

Submit

Now Go to Tasks and open the **Enter GPA** Task and click on Start. Then Enter the GPA and click on Complete.

9 - Enter GPA

Work Details Assignments Comments

gpa (student)

3

name (student)

Jana

Save Release Complete

You can see the **Review GPA** task has been displayed. Again, click on it and check the eligible check box to approve the admission.

Search...

10 - Review GPA

Work Details Assignments Comments

☒ eligible (student)

name (student)

Jana

gpa (student)

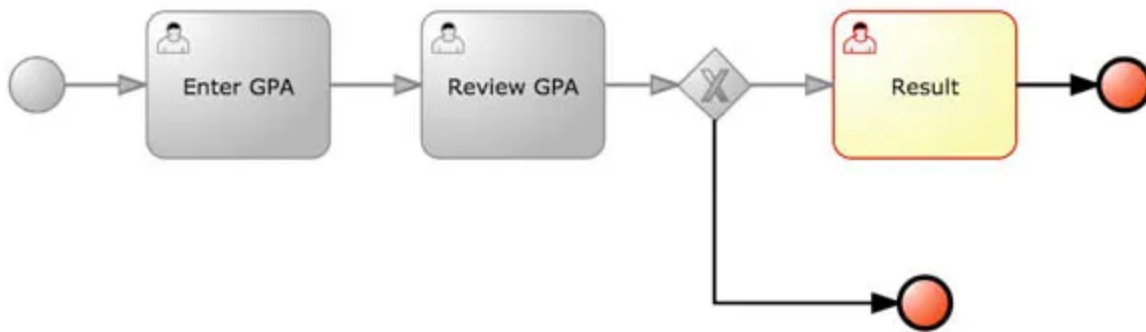
3

Save

Release

Complete

The Result task will be displayed now. Open it and you can see the data. Now, go to **Process Instance** and open the instance we had executed. Open **Process Model** and notice since we selected the eligible checkbox, the Result task has been executed. Execute the process again from process definition and this time don't select the eligible checkbox, notice the Result task will not be invoked this time.



In **next part** we will learn how to use Guided Rules and DRL file.

Related



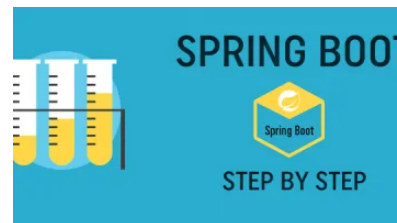
Get started with jBPM KIE and Drools Workbench – Part 1

In "jBPM"



Get started with jBPM KIE and Drools Workbench – Part 3

In "jBPM"



How to Create Spring Boot Application Step by Step

In "Spring Boot"

Filed Under: **jBPM** Tagged With: **BPM, Business Process, Drools, JBOSS, jBPM, KIE, KIE Workbench**

Subscribe to stay in loop

* indicates required

Email Address *

Subscribe

Comments



KRan says

April 13, 2016 at 3:51 pm

(Edit)

For Review task, it gives following fields in the form:

- 1) Requires Review – TextBox
- 2) Requestor of the operation – TextBox
- 3) Repository – TextBox
- 4) Is operation approved? – Checkbox

Also when clicked on 'Complete', it gives NullPointerException.

Any idea what could be wrong?

Reply



machiume says

May 17, 2016 at 4:04 pm

(Edit)

Experiencing the same problem here KRan. Did you ever work this out?

Reply



machiume says

May 17, 2016 at 4:14 pm

(Edit)

Okay, so it may have something to do with the task name 'Review'. I changed it to something else, rebuilt the form, and it worked.

Reply



E says

June 7, 2016 at 5:19 pm

(Edit)

Guess I'm an idiot but clicking on the white space next to my process model doesn't do anything – can't seem to get that editor window up where I pull in my data model

Reply



Katherine says

July 19, 2016 at 7:43 pm

(Edit)

BION I'm imeesrspd! Cool post!

Reply



A Developer Diary says

August 10, 2016 at 4:13 pm

(Edit)

Thanks Katherine !

Reply



Abdu Chadili says

January 19, 2017 at 8:39 am

(Edit)

Good getting started post !

Do you have a blog that publishes similar articles which progressively introduce more advanced features.

Thanks

Reply

Leave a Reply

Logged in as Abhisek Jana. [Edit your profile](#). [Log out?](#) Required fields are marked *

Comment *

Post Comment

This site uses Akismet to reduce spam. [Learn how your comment data is processed](#).

Copyright © 2024 A Developer Diary