Screen

Description

This step is used to define the custom form that will be used by the users who have to approve or to reject the generated task. To configure that step will be necessary to determine the fields that will be show to the users, and the actions that these users could perform.

Steps Tabs

Task details

In this tab you could configure next parameters:

- **Task name**: identified name for the task that will be created when the workflow is requested.
- **Actor(s) expression**: allows you to write an expression to identify the actor depending on the requested role. One can use EL expressions (*) based on role and application attributes. For instance: ${owners}
- **Assignment script**: alternatively, allows you to write a Beanshell script to return the actor depending on the process variables. For instance: return primaryGroup.attributes{"owner"};
- **Approve from email**: checked it to allow you to send a mail for approval the task.

Fields

In this tab, you could choose what fields the process form will show to the end users. You can choose these fields from all identity attributes, and from the attributes defined for the workflow on the Attributes Tab. By default, all the identity attributes will be shown. You can choose the fields you want to show, add new fields, and delete the fields that do not need to generate a task. Also, you can sort the fields, you only need to drag and drop on the Order column.

For each field, you may indicate if it is a readOnly field, and you may add a Validation script and Visibility script. The validation script allows you to define rules, the field has to comply with these

rules. The visibility script allows you to define the rules to show or hide a field.

Example

```
if (value == null || value.equals(""))
  return ("The NIF is mandatory");
else
  return true;
```

Trigger

On the trigger tab, you could define different triggers using custom scripts. Those triggers will be launched with the events you will define.

- **onLoad**: you can use that trigger to perform some actions before the execution of the step.
- **on PrepareTransition**: you can use that trigger to perform some actions after the execution of the step and before starting a transition to another step.
- **onChange**: you can use that trigger to perform some actions when the value of the attribute is changed. You could choose the field from a list.

Example

```
requester = task.getVariables().get("requester");
systemName= task.getVariables().get("systemName");
.......
```

Incoming transitions

The Incoming transitions tab displays the previous steps where the flow comes from. When you create a process from a template or from scratch default incoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: the previous step, where the flow comes. Allows you to select where the workflow comes from.
- **Incoming transition**: brief name to identify the transition. That is the name of the action the form will show to the final user.
- **To**: current step.
- **Action**: allows creating a custom script to perform specific actions.

When you create an incoming transition, Soffid creates the proper outcoming transition.

Example

Get the owners of an account and do something with each one.

```
accounts = serviceLocator.getAccountService().findAccountByJsonQuery("name eq \"" +
executionContext.getVariable("account") + "\"");
if (!accounts.isEmpty() {
  for (account:accounts) {
    owners = serviceLocator.getAccountService().getAccountUsers(account);
    // TO-DO
  }
}
```

Outgoing transitions

The Outcoming transition tab displays the next steps where the flow can go from the current step. When you create a process from a template or from scratch default outcoming transitions are defined. It is allowed to customize the default setup, add new transitions, or delete transitions.

- **From**: current step.
- **Incoming transition**: name of the transition.
- **To**: the next step, where the flow goes.
- **Action**: allows creating a custom script to perform specific actions.

When you create an outcoming transition, Soffid creates the proper incoming transition.

Example

Get the mail of the requester and send a notification.

```
requester = executionContext.getVariable("requester");
user = serviceLocator.getUserService().findUserByUserName(requester);


serviceLocator.getMailService().sendTextMail(
  user.emailAddress,
  "Resquest Rejected",
  "XXXXXXXXXXXXX");
```

** https://es.wikipedia.org/wiki/Expression_Language*

Revision #12
Created 23 June 2021 06:28:54
Updated 24 November 2022 12:13:20