# 1.Workflows

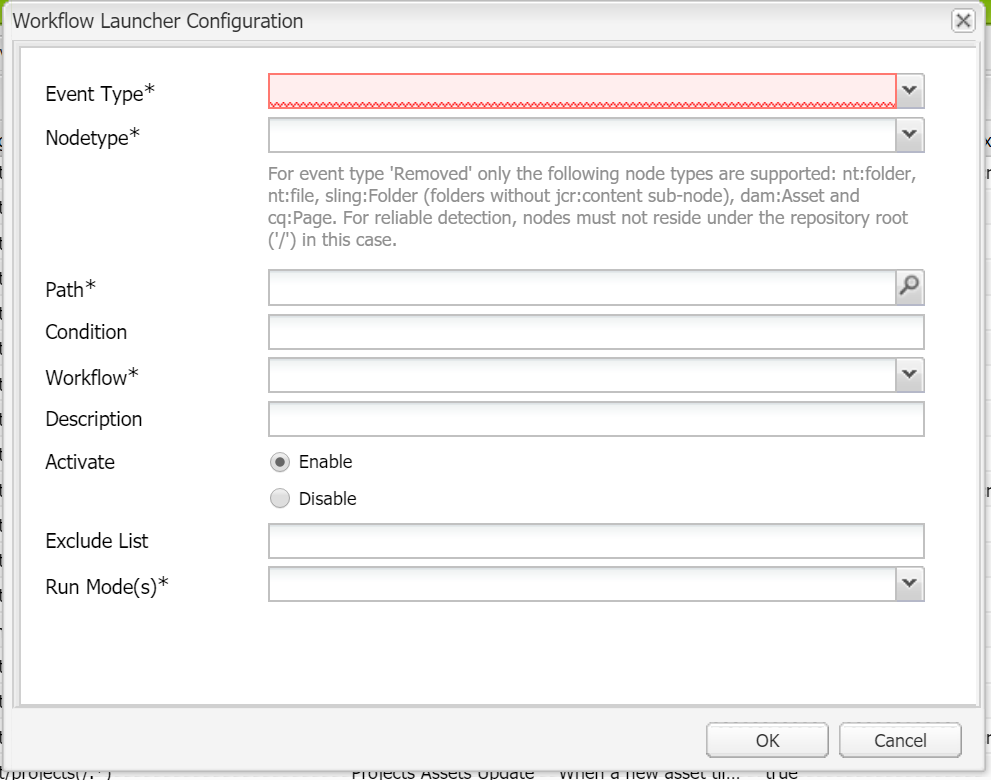
1. Enables to automate activates
2. Has series of steps which are executed one after the other
3. Each step has distinct functionality (accessing repository, activating a page or setting some properties, sending mails for approval )
4. Are versioned

The step in (3) can be an ECMA script of a java class(deployed as bundle in AEM)

# 2.Workflow Console

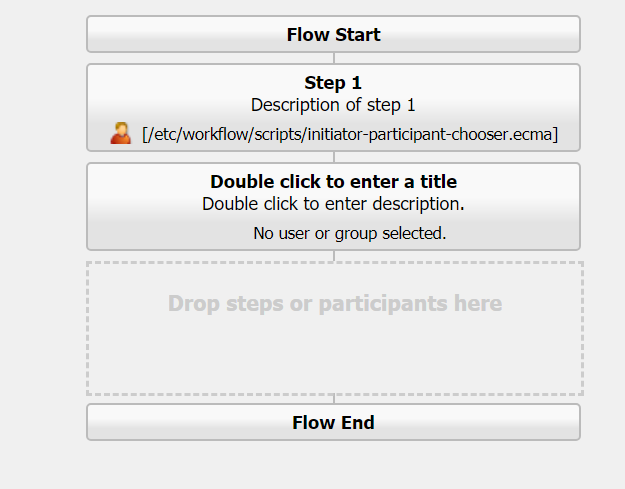
The workflow console at - <http://localhost:4502/libs/cq/workflow/content/console.html>

1. Models – All the workflows created
2. Instances - instances of the active models (all running workflows with respective payloads)
3. Archive – All completed workflows. You can determine if a worflow was successful or not by viewing the Workflow Archive tab.
4. Launcher – used to automatically launch a workflow instance based on some conditions on a JCR resource. when creating a new launcher below is the dialog shown



# 3.Workflow Model

Below picture shows a model



Each model has start & end nodes. Model has *workflow nodes , workflow transitions.* Transitions are the ones which connect the nodes and define the flow. Each node is called a step.

# 4. Workflow Step

Each step defined has a specific functionality.

Below are the types of Workflow steps available

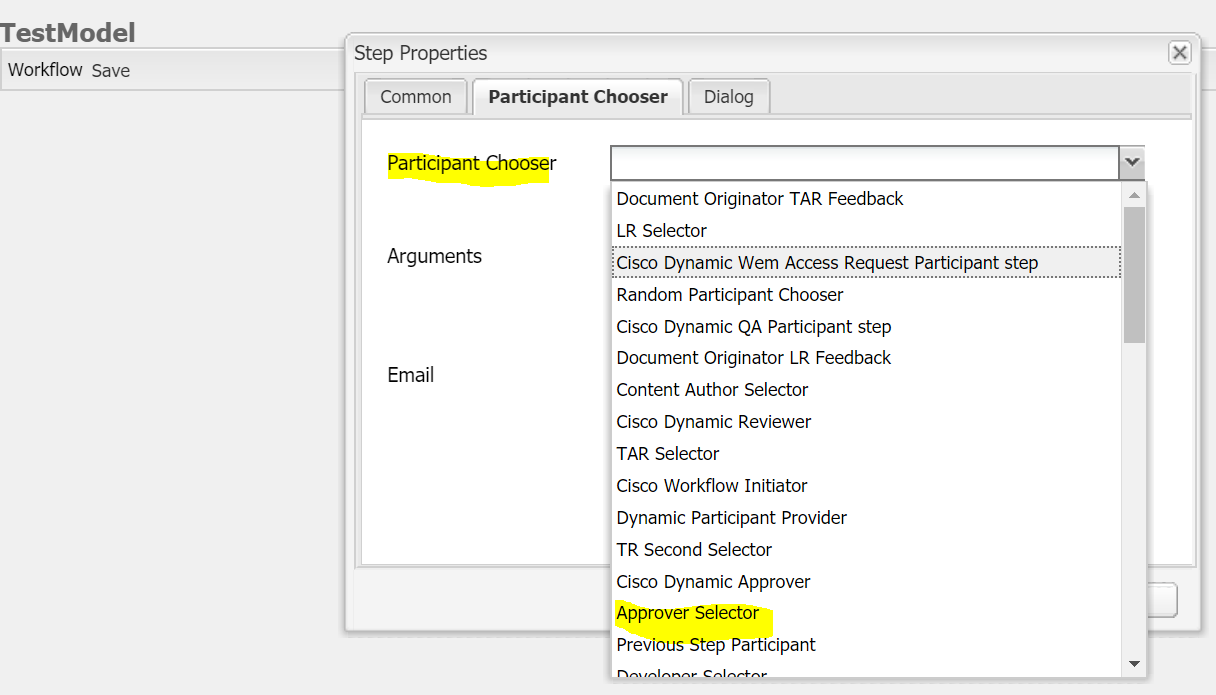
1. **Participant Step(user/group):** assigns a workItem to an user or a group of users. Either of them has to approve to advance the workflow to next step
2. **Process Step :** These steps are executed by a script (java / ECMA). These can be deployed as OSGi services
3. **Container :** These type of steps can trigger another workflow model
4. **OR :** the flow is branched (multiple branches) and the script decides which branch is executed
5. **AND :** allows multiple steps to be executed simultaneously.

# 4.1 Participant Step

Custom participant step can be written implementing the **ParticipantStepChooser.class** and overriding **getParticipant(workItem,workflowSession,MetadataMap).**

Here in the annotation **@Property(**name**=** ParticipantStepChooser.***SERVICE\_PROPERTY\_LABEL*** value=**”Approver Selector”)**

**Approver Selector** is the name available to select in UI as below



# Types of Participant Steps

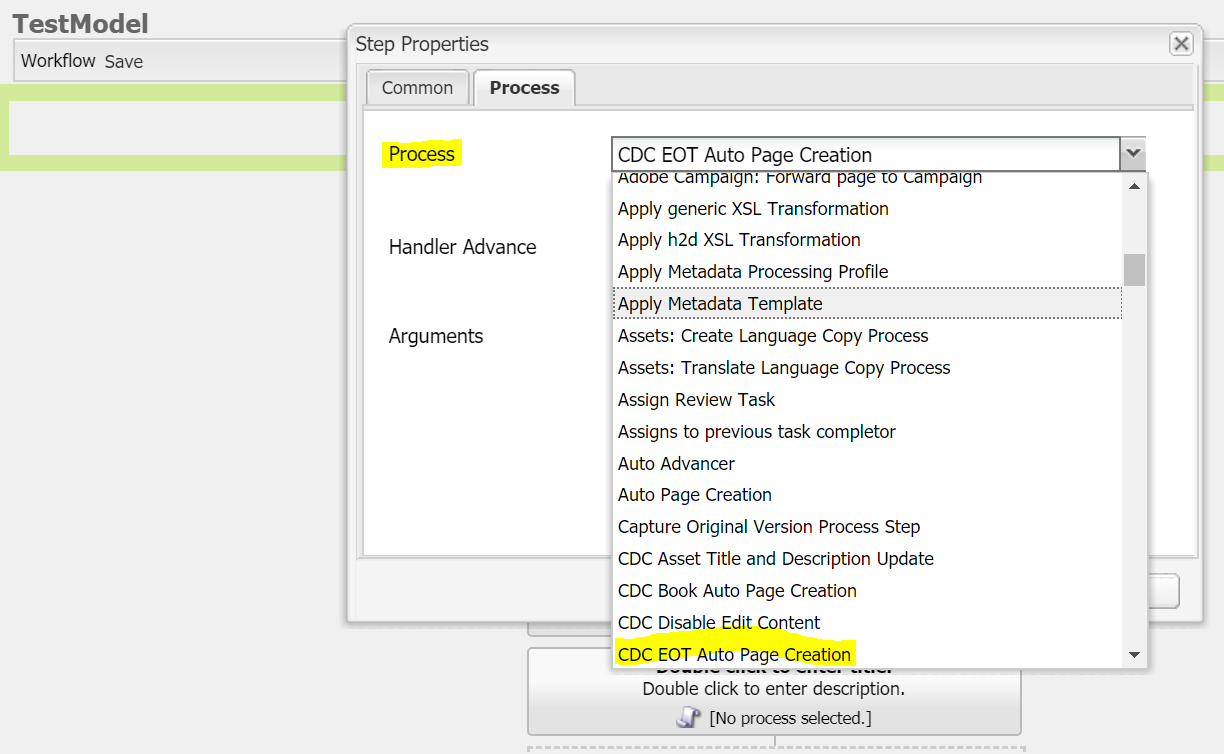
* + - 1. **Dialog Participant Step** – display a **dialog** to user and collects the inputs to be stored on payload or WorkItem. We should provide the path of **dialog** or create a new one.
      2. **Dynamic Participant Step** – assigns the work Item to the user automatically at run time
      3. **Form Participant Step**
      4. **Random Participant Step** – selects the participant randomly from the list of users/groups
      5. **Workflow Initiator Participant Step** – automatically assign the workItem to the initiator

# Process Step

Custom participant step can be written implementing **WorkflowProces.class** and overriding **execute(workItem,workflowSession,MetadataMap).**

Here in the annotation **@Property(**name**=** “**process.label**” value=**”CDC EOT Auto Page Creation”).**

**CDC EOT Auto Page Creation** is what will be visible in the drop down of **Process Step**



# Creating custom Step components

Create **cq:component** with sling:resourceSuperType as any one of below

* cq/workflow/components/model/process – Process step
* cq/workflow/components/model/participant – Participant Step
* cq/workflow/components/model/dynamic\_participant

The ***cq:formParameters*** on the editConfig node of above custom step component can have any one of below properties for a custom implementation

|  |  |  |
| --- | --- | --- |
| 1 | PROCESS | fully qualified name of bundles java class i.e., PID of the service in OSGi |
| 2 | PARTICIPANT | ID of the user to whom the custom step is to be assigned to |
| 3 | DYNAMIC\_PARTICIPANT | PID of service in OSGi that selects the User to assign this step to |
| 4 | DIALOG\_PATH | path to the dialog in repository |
| 5 | PROCESS\_AUTO\_ADVANCE | To enable Handler Advance |
| 6 | DO\_NOTIFY | To enable email Notifications |

# OR Split

This step splits the flow into desired no of branches (upto 5). Each branch has an option to set as a default branch. A script or path to the script has to be specified to each branch.

Each branch script is executed one at a time.

The first script that ends up as ***true***  is executed

If everything is ***false,*** default branch is executed

# Storing workflow dialog data

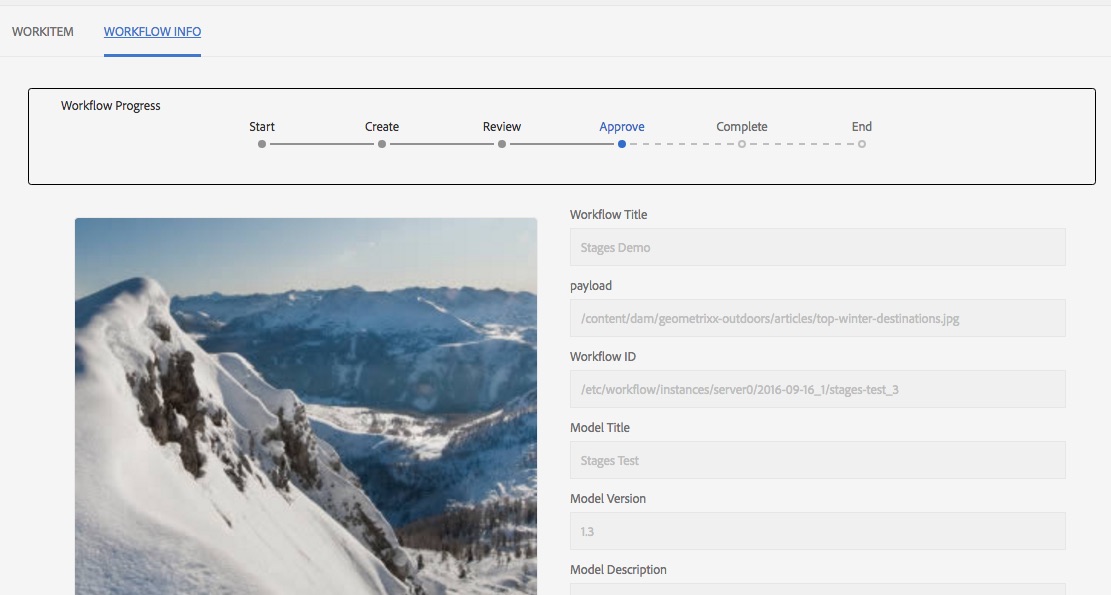
It can be stored at 2 places

1. At payload – giving the ***name*** property a value of ***“./jcr:content/someProperty”***
2. Metadata – giving the ***name*** property a value of “***someProperty***”

# Workflow stages

Helps to visualize the progress of a workflow instance. Stages can be created at page properties dialog (last tab) from workflow console. Each step in workflow can be assigned with an individual stage.

The progress of workflow can be viewed at ***WORKFLOW INFO*** tab at task details in inbox of respective individual



# Working with sessions

Use below ways to gain access to JCR session

**session = workflowSession.adaptTo(javax.jcr.Session.class);**

**session = workflowSession.getSession();**

* Do not save a session. Workflow engine saves the session for you
* More efficient and better error handling if you let the workflow engine save session

# Terminologies to note

**Payload** : a path in JCR on which the workflow is invoked

**WorkItem metadata** : has information related to a single step. This can be ***SHARED*** across various other steps in workflow.

Setting some data in metadata

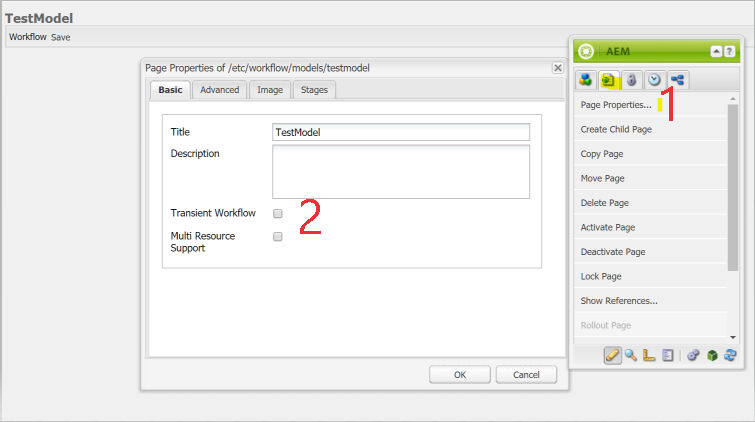
**workItem.getWorkflowData().getMetaDataMap().put(“someKey”, “someValue”);**

Gettingdatafrommetadata

**workItem.getWorkflowData().getMetaDataMap().get(“someKey”, String.class);**

# Transient Workflow

Workflows for which we don’t want any history or time stamps can be triggered as transient workflows. This is for performance tuning



# Multi resource Support

To initiate a single workflow instance on a bunch of resources, we have to enable ***Multi Resource Support*** in the above picture.

If this is unchecked & a bunch of resources are selected and clicked on any workflow, then

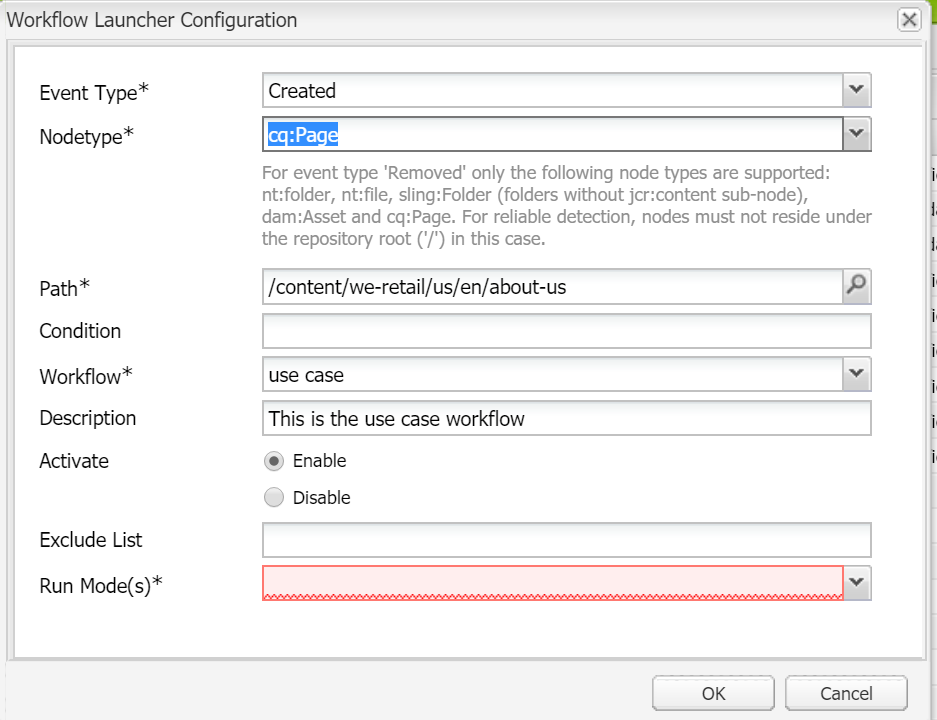
**No of workflow instances = no of selected resources**

# Use case

Create a workflow that lets administrators review newly created pages and move them under the proper parent. The workflow is launched every time a page is created under  **/content/we-retail/us/en/about-us**. The new page goes to the workflow as the payload. The workflow checks “***pathToMove***” property of the page’s jcr:content node. If the property is not present, the workflow is assigned to administrators group. A group member enters the pathToMove property using a dialog. After that, the workflow updates “***pathToMove***”property. If it contains a valid path, it moves the page to that path. If the path is empty, invalid, or matches the current path, no action is taken.

# Approach

1. Create a launcher as below



1. Select the newly created work flow **use case** from the drop down in above dialog
2. Now lets define the model

<http://localhost:4502/etc/workflow/models/UseCase.html?cq_ck=1575885788397>

# ECMA Scirpt

function check(){

var path = WorkflowData.getPayload();

var session = workflowSession.adaptTo(Session.class);

Node node = session.getNode(path+"/jcr:content");

if(node.hasProperty("pathToMove"){

return true;

}

else{

return false

}

}

# Useful links

<https://helpx.adobe.com/in/experience-manager/6-3/sites/developing/using/workflows-step-ref.html#ParticipantStepsandChoosers>

<https://aem.redquark.org/2018/10/day-15-custom-workflows-in-aem.html>

issue of storing data at payload - <https://forums.adobe.com/thread/2393850>

<https://stackoverflow.com/questions/46815590/aem-workflow-set-values-in-a-custom-dialog-from-with-data-gathered-in-previous>

<http://experience-aem.blogspot.com/2019/04/aem-65-content-copy-using-workflow-dialog-participant-step.html>

<https://docs.adobe.com/content/help/en/experience-manager-64/developing/extending-aem/extending-workflows/workflows-customizing-extending.html>