Creating a Simple Workflow with AWE(Approval Workflow Engine)

Creating a Simple AWE Workflow

With the recent release of Peopletools, peoplesoft has moved from the traditional workflow to the AWE(Approval Workflow Engine). This has greatly segregated the designing of workflow to the functional team and the supporting objects to the Technical Developer. Unlike Workflow, there is no requirement of creation of Components, steps, rules and routing in Application Designer. In this way peoplesoft has tried to simplify the workflow process making it more configurable.

Starting with a simple AWE workflow we need to carry out few steps.

1. **Requester** requests for some amount for an asset. When the requester submits the request, the approvers get the notification by email and worklist. The approvers are the list of all the users who are having the role of Finance Officer (FIN_OFFCR). One of Approver approves the request and the Asset status is set to Approved else Denied.

2. **Transaction Component**

   Now we need a component which will trigger the workflow or where the requester will submit the requests.
I have added 3 buttons: Approve, Deny, Submit. The buttons as suggested by their name are to Submit, Approve or Deny a Request. At this moment all three buttons are visible but we will control its visibility as we progress. We will later see how this page is customized to fulfill our requirements.

4. Event Handler - AWE is controlled by events, when a request is submitted, a event takes place, when a request is approved a event triggers, when a request is denied, a event take place. Each of these events are handled by AWE delivered class EOAW:Core. If we want to do other stuffs than what the delivered app package does, we need to create our Event Handler, extending the core EOAW classes. As of now we do not need to create any event Handler, the delivered EOAW:Core app package will do our job. But, later we will see that we need to create an event handler to tackle our requirement.

5. Configuring AWE, Registering the transaction. Navigate to:

Main Menu  >  Enterprise Components  >  Approvals  >  Approvals  >  Transaction Registry

Add a new Process ID, Let us create a process ID “ASSET_APPROVAL”. 
Enter the Cross Reference Record which we defined in Step 2.

In Notification Options Select as below.

The checkbox 'Use Email Approval' and below fields are used when we are using EMC. Now it is not required. We will see details of EMC in the advanced chapters of AWE.

Keep the Internal URL definition and External URL definitions blank.
We do not need these as of now.
Fill the Default Approval Component as below. This is the same transaction component we created in step
We are keeping Approval Status Monitor section as blank.
We do not need these items for our simple AWE.
As explained earlier, "Approval Event Handler Class" is used to do some stuffs when a particular event triggers like updating the "Status" field to "Approved/Denied" when the request is approved/Denied.
As of now we have not created any Event Handler for our workflow, so we will use the delivered App package and Class as shown below. Later we will replace this event handler with our event handler class.

"Approval Status Monitor" section defines Adhoc class and Thread class.
Adhoc class is used for entering adhoc approvers/reviewers or paths in the usual AWE workflow.
Thread Class is used to control what is displayed in the Approval Monitor.

![Default Approval Component](image1)
![Approval Event Handler Class](image2)
![Approval Status Monitor](image3)

Enter the details of Header record and its key fields as below in the "Transaction Approval Levels" section.

![Transaction Approval Levels](image4)

Save the page.

6. Configuring AWE, Transaction Configuration
EVENTS:

Configure the events as below:

1. **Event**: On Final Approval
   - **Menu Name**: ADMINISTER_WORKFORCE (GBL)
   - **Approval Component**: ASSET_APPROVAL.COM
   - **Page Name**: ASSET_APPROVAL
   - **Menu Action**: Update
   - **Notification**:
     - **Template Name**: APT_WebAsset_Approved

2. **Event**: On Final Denial
   - **Menu Name**: ADMINISTER_WORKFORCE (GBL)
   - **Approval Component**: ASSET_APPROVAL.COM
   - **Page Name**: ASSET_APPROVAL
   - **Menu Action**: Update
   - **Notification**:
     - **Template Name**: APT_WebAsset_Denied

3. **Event**: Route for Approval
   - **Menu Name**: ADMINISTER_WORKFORCE (GBL)
   - **Approval Component**: ASSET_APPROVAL.COM
   - **Page Name**: ASSET_APPROVAL
   - **Menu Action**: Update
   - **Notification**:
     - **Template Name**: APT_WebAsset_Awaiting
We need to pass email template to the AWE workflow.
This can be created as below.
Please note that bind variable %1 is used by AWE and it means the URL pointing to the transaction.

7. Configuring AWE, Approval Process Setup

Before Creating the Approval Process, we need to setup USERLIST which is required.
Userlist defines the list of user who will be getting the notification when some event triggers.
Here, we are creating a userlist of all the users having the role FIN_OFFCR as shown below.
Adding the Approval Process Setup.

Since, we are creating approval process of only one step.
We will create only one stage, one path and one step in the setup.
Click on Definition Criteria and set it as Always true.

Click on Alert Criteria and set it as Always true.

Click on Path Criteria and set it as Always true.
Click on Step Criteria and set it as Always true.

All Criteria is now set, So a tick is appearing in each criteria link.
Criteria are used to determine whether to enter a path, step, stage or not.

Click on Save:

8. Coding the workflow.

We had three Push buttons: Submit, Approve and Deny.
Let's code that when the user clicks on Submit, it launches the AWE workflow. When the Approver approves the request, it completes the workflow as approved and vice versa.

Let's write when the user clicks the submit button, we assign the variable &c_apprAction as "S", Approve button as "A" and Deny button as "D".

For launching AWE workflow we need to call the DoSubmit() method of the EOAW_CORE:LaunchManager class. For Approving the running Workflow the DoApprove() method has to be called. Similar is the case for DoDeny(). Below is the code:

```java
import EOAW_CORE:LaunchManager;
import EOAW_CORE:ApprovalManager;

Declare Function createStatusMonitor PeopleCode EOAW_MON_WRK.EOAW_FC_HANDLER FieldFormula;

Component string &c_apprAction;

// Submit Button Code
Component string sc_apprAction;
sc_apprAction = "S";
If (Doc GetRow().IsChanged) Then
  REM ** force save processing;
  SetComponentChanged();
End-If;
DoSave();

// Approve Button Code
Component string sc_apprAction;
sc_apprAction = "A";
If (Doc GetRow().IsChanged) Then
  REM ** force save processing;
  SetComponentChanged();
End-If;
DoSave();

// Deny Button Code
Component string sc_apprAction;
sc_apprAction = "D";
If (Doc GetRow().IsChanged) Then
  REM ** force save processing;
  SetComponentChanged();
End-If;
DoSave();
```

We have written the code on buttons but we still have not launched the AWE. The best place to trigger AWE events are at SavePostchange as all the validations are done by that point.

For launching AWE workflow we need to call the DoSubmit() method of the EOAW_CORE:LaunchManager class. For Approving the running Workflow the DoApprove() method has to be called. Similar is the case for DoDeny(). Below is the code:

```java
import EOAW_CORE:LaunchManager;
import EOAW_CORE:ApprovalManager;

Declare Function createStatusMonitor PeopleCode EOAW_MON_WRK.EOAW_FC_HANDLER FieldFormula;

Component string &c_apprAction;
```
Component EOAW_CORE:LaunchManager &c_aweLaunchManager;
Component EOAW_CORE:ApprovalManager &c_aweApprManager;
Local Record &headerRec = GetRecord(Record.T_HEADER);
Local boolean &isApprover;

Evaluate &c_apprAction
When "S"
&c_aweLaunchManager.DoSubmit();
If (&c_aweLaunchManager.hasAppInst) Then
REM ** Initialize Approval Manager if transaction was submitted;
&c_aweApprManager = create EOAW_CORE:ApprovalManager(&c_aweLaunchManager.txn.awprcs_id, &headerRec, %OperatorId);
End-If;
Break;
When "A"
&c_aweApprManager.DoApprove(&headerRec);
Break;
When "D"
&c_aweApprManager.DoDeny(&headerRec);
Break;
End-Evaluate;

If &c_aweApprManager.hasAppInst Then
&isApprover = &c_aweApprManager.hasPending;
createStatusMonitor(&c_aweApprManager.the_inst, "D", Null, False);
End-If;

Adding the Approval Monitor Page:

---

Component EOAW_CORE:LaunchManager &c_aweLaunchManager;
Component EOAW_CORE:ApprovalManager &c_aweApprManager;

Local Record &headerRec = GetRecord(Record.T_HEADER);
Local boolean &isApprover;

Evaluate &c_apprAction
When "S"
&c_aweLaunchManager.DoSubmit();
If (&c_aweLaunchManager.hasAppInst) Then
REM ** Initialize Approval Manager if transaction was submitted;
&c_aweApprManager = create EOAW_CORE:ApprovalManager(&c_aweLaunchManager.txn.awprcs_id, &headerRec, %OperatorId);
End-If;
Break;
When "A"
&c_aweApprManager.DoApprove(&headerRec);
Break;
When "D"
&c_aweApprManager.DoDeny(&headerRec);
Break;
End-Evaluate;

If &c_aweApprManager.hasAppInst Then
&isApprover = &c_aweApprManager.hasPending;
createStatusMonitor(&c_aweApprManager.the_inst, "D", Null, False);
End-If;

Adding the Approval Monitor Page:
/* Initialize the launch and approval managers. ApprovalManager will
* need reinitialization on submit
*/
&c_aweLaunchManager = create EOAW_CORE:LaunchManager(&PROCESS_ID, &headerRec, %OperatorId);
&c_aweApprManager = create EOAW_CORE:ApprovalManager(&PROCESS_ID, &headerRec, %OperatorId);
/* Uncomment the following line if you don’t want AWE to choose the
* Definition Id based on the preconfigured definition criteria.
* Definition criteria is maintained using the “Setup Process
* Definition” component.
*/
&c_aweLaunchManager.definition = "SHARE";

If &c_aweApprManager.hasAppInst Then
    &isApprover = &c_aweApprManager.hasPending;
    createStatusMonitor(&c_aweApprManager.the_inst, "D", Null, False);
End-If;

9. EOAW_IDS:
Last but not the least, we need to setup EOAW_ID for our workflow.
EOAW_IDS defines a number from which AWE will start creating threads.
In HCM its is similar to what will be the first Empid and it will be then incremented with each empid creation.
Same as empid, you can initialize your workflow with some number and then AWE will take care of incrementing it and maintaining it with your transactions.
It requires a manual insert SQL.

INSERT INTO PS_EOAW_IDS(EOAWCOUNTERNAME, EOAWCOUNTER)VALUES ('APT_WA_AWE_XREF', 1);

We should not change this value once it is setup. This can cause multiple transactions to end up with the same keys.

Approval Component.

Asset Approval

Enter any information you have and click Search. Leave fields blank for a list of all values.

Find an Existing Value  Add a New Value

Search Criteria

Search by: Asset Identification begins with

Search  Advanced Search

Let's add a new Asset id for approval.

Asset Approval

Find an Existing Value  Add a New Value

Asset Identification: A00001

Add
We still have all the three buttons on the page which we need to control. The submit button should only be visible for the requester and Approve and Deny button should be visible for the Approver.

We can write a small piece of code checking if the workflow has started or not, if not show SUBMIT button else show Approve and Deny button.

We can make use of the "hasappinst" property of the Approval Manager object which we used to display Approval Monitor Earlier.

Postbuild code:

```java
If &c_aweApprManager.hasAppInst Then
    T_BUTTONS.SUBMIT_BTN.Visible = False;
    T_BUTTONS.DENY_BTN.Visible = True;
    T_BUTTONS.APPROVE_BTN.Visible = True;
Else
    T_BUTTONS.SUBMIT_BTN.Visible = True;
    T_BUTTONS.DENY_BTN.Visible = False;
    T_BUTTONS.APPROVE_BTN.Visible = False;
End-If;
```

So, now the buttons are displaying correctly.

Let's go ahead and submit a request for this Asset.

One thing I missed was to include a currency code field in the page which tells the amount's Currency. This will be also useful when we define a 'Criteria' on monetary basis because you cannot have a monetary criteria without currency code. Criteria will be discussed in later chapters.

Let us add the currency code field on the page. Since this field is going to be a non key, this is not going to impact to workflow and we can add it without issues. You may add it in the header record and use it on the page.
Now is the time to test our workflow.

Let's submit a request for an asset.

Click on Submit.

Wooow, you can see the approval monitor below.
It means, our workflow has triggered. The approval monitor shows that it is pending with finance officer KUTL101. Let’s log in as KUTL101 and see if there is any worklist created for this request. Logging as KUTL101.

We see the worklist in KUTL101 inbox, that means our workflow has triggered. Let’s go ahead and Approve the request.

Click on the Approve button:
We have successfully completed the AWE workflow. The approval monitor is showing the transaction to be approved but still the status at the top right corner is showing as pending. This is because AWE does is not aware of this field.

The Eventhandler class comes into picture in scenario like this where we want to do some stuff on AWE events.

Let's go on create an event handler class which update the status of the status field as "Approved" when the request is approved.

I have created a app package and inserted a class in the package.

The class extends the EOAW_CORE classes and update the "Status" field as approved or Denied according to the approver selection.

```java
import EOAW_CORE:ApprovalEventHandler;
import EOAW_CORE:ENGINE:AppInst;
import EOAW_CORE:ENGINE:UserStepInst;
import EOAW_CORE:ENGINE:Thread;
import TRAVERSENULLFIELD:DisableFields;

class WebAssetAppr_EventHandler extends EOAW_CORE:ApprovalEventHandler

  method OnHeaderApprove(&appinst As EOAW_CORE:ENGINE:AppInst);
  method OnHeaderDeny(&userinst As EOAW_CORE:ENGINE:UserStepInst);
  private
    method UpdateStatus(&thread As EOAW_CORE:ENGINE:Thread, &status As string);
    method componentDisplayOnly(&Rs As Rowset);
  end-class;

  method OnHeaderApprove
    /* &appinst as EOAW_CORE:ENGINE:AppInst */
    /* Extends/implements EOAW_CORE:ApprovalEventHandler.OnHeaderApprove */
    &This.UpdateStatus(&appinst.thread, "X");
  end-method;

  method OnHeaderDeny
    /* &userinst as EOAW_CORE:ENGINE:UserStepInst */
    /* Extends/implements EOAW_CORE:ApprovalEventHandler.OnHeaderDeny */
    &This.UpdateStatus(&userinst.thread, "D");
  end-method;

  method UpdateStatus
    /* &thread as EOAW_CORE:ENGINE:Thread, */
```
/* &status as String */
/* &thread.recname contains the header record name, but we are 
* using a sibling record so we have to hard code the record name
*/
Local Record &asset_rec = CreateRecord(Record.T_HEADER);
/* &thread.rec contains the cross reference record which has
* header record keys */
&thread.rec.CopyFieldsTo(&asset_rec);
&asset_rec.SelectByKey();
&asset_rec.GetField(Field.APPROVAL_STATUS).Value = &status;
&asset_rec.Update();
GetLevel0().Refresh();
%This.componentDisplayOnly(GetLevel0());
end-method;

method componentDisplayOnly
/+ &Rs as Rowset +/
Local number &i, &j, &k;
For &i = 1 To &Rs.ActiveRowCount
    For &j = 1 To &Rs(&i).RecordCount
        For &k = 1 To &Rs(&i).GetRecord(&j).FieldCount
            &Rs(&i).GetRecord(&j).GetField(&k).DisplayOnly = True;
        End-For;
    End-For;
End-For;
For &j = 1 To &Rs(&i).ChildCount
    %This.componentDisplayOnly(&Rs(&i).GetRowset(&j));
End-For;
End-For;
end-method;

You might have notice that I have created a method “ComponentDisplayOnly” to make all the fields disabled once the user Approves 
or Deny the request.
This function uses recursive method to loop through each field in the component and makes them displayonly. This method becomes 
very handy when we need to disable all the fields of the components except few push buttons or hyperlink. In that case the usual 
page class peoplecode, page.displayonly won’t be affective and it will make buttons and hyperlinks also disabled and you can not 
change it via peoplecode.

More detail of this method can be find in this link.

Now we need to modify the Transaction Registry to include this Event Handler.
Save the page.

Let's submit a request again.

So the code is working as expected.
Well Done.
For AdHoc approval while reviewing a transaction, I understand we need to launch the ApprovalManager object as follows with the parameter "A" for add and not "D" for display:

```java
createStatusMonitor(&approvalMgrObj.the_inst, "A", Null, False);
```

Subsequently when trying to save the newly added approver/reviewer with the standard "Apply Approval Changes" using the following, am getting an error on executing the following code:

```java
&approvalMgrObj.the_inst.Save();
DoSave();
```

/* ERROR OBSERVED */