<INTERNAL>

By:

Change History:

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Description** |
| 04/25/2011 | Nathan Kral [nkral@ptc.com](mailto:nkral@ptc.com) | Initial draft |

</INTERNAL>

</INTERNAL>

# Best Practice Name and Classification

## Name

*Promotion Request Workflow Helpers* Best Practice

## Classifications

Low Level

## Also Known As

None

</INTERNAL>

# Objective

## Problem Statement

Problem: As a Business Process Administrator, I want access to useful workflow helper APIs that can assist with the definition of my "Promotion Request" processes.

## Background

When customizing a Promotion Request workflow you may have a need to retrieve or set promotion request attributes as well as the promotion candidates during the promotion request workflow for doing additional validation or processing of a workflow expression.

Latest iterations

Automatically revise the promotion candidates

## Scope/Applicability/Assumptions

Assume you have or are creating a customized Promotion Request workflow process.

## Intended Outcome

The workflow helper APIs in this best practice will cover the following areas:

1. Getting and setting attributes of the "Promotion Request"
2. Getting the "Target State" of the "Promotion Request"
3. Getting the "Promotion Candidates" associated with a "Promotion Request"
4. Refreshing the members of the "Promotion Request" to their latest iterations
5. Automatically creating new revisions of the "Promotion Candidates"

# Solution

## Solution Statement

Solution: Use the Maturity Service and Promotion workflow helper to when defining a Promotion workflow process.

## Prerequisite knowledge

To apply this best practice, you need to have an understanding of the following:

* Basic development involving Java and XML.
* Windchill “workflow” concepts.

The [See Also](#_See_Also) section below includes references to many or all of these subjects.

## Solution Elements

|  |  |  |
| --- | --- | --- |
| **Element** | **Type** | **Description** |
| wt.maturity.MaturityService | Interface | Interface for the Promotion Request services |
| wt.maturity.PromotionNotice | Class | Model class of the Promotion Request object |
| wt.maturity.MaturityHelper | Class | Factory helper to retrieve the registered MaturityService implementation |
| com.ptc.windchill.enterprise.maturity.PromotionNoticeWorkflowHelper | Class | Helper class for refreshing the members of the "Promotion Request" to their latest iterations. See Java doc for more details. |
| com.ptc.core.lwc.server.LWCNormalizedObject | Class | Used to set and get attributes on an object. See javadoc for more details. |
| wt.fc.PersistenceHelper | Class | Factory helper to retrieve the Persistence Manager |
| com.ptc.windchill.enterprise.maturity.validators.PromotionTargetsReviseValidator | Class | Validates that the promotion candidates are eligible to be revised. |

## Procedure - Getting & Setting Attributes

Retrieving the modeled attributes on the Promotion Request object can be done either by calling the supported getter method for the attribute. In order to retrieve soft attributes from the Promotion Request the recommending technique is to use the LWCNormalizedObject class which can also be used for retrieving the modeled attributes. The following is an example of retrieving a soft attribute on a Promotion Request.

LWCNormalizedObject lwc = **new** LWCNormalizedObject(promotionRequestObj, **null**, **null**, **null**);

lwc.load("*softAttribute*");

Object softAttrValue = lwc.get("*softAttribute*");

Attributes can be set using similar techniques. There are corresponding setter methods for the modeled attributes and the LWCNormalizedObject class can be used to set soft attributes and model attributes. After updating an attribute the Promotion Request object will need to be updated which is done by calling the modify method in the Persistence Manager. The following is an example of setting a soft attribute on a promotion request:

LWCNormalizedObject lwc = **new** LWCNormalizedObject(promotionRequestObj, **null**, **null**, **new** UpdateOperationIdentifier());

lwc.load("*softAttribute*");

lwc.set("*softAttribute*", "soft attribute text");

lwc.apply();

PersistenceHelper.manager.modify(my\_persistable);

For more information on using the LWCNormalizedObject see the class javadoc.

## Procedure - Getting Target State

The promotion request target state is an attribute on the Promotion Request called maturityState. The promotion target state is setting in the promotion request wizard and target state can be retrieved by calling the getMaturityState method on the promotion request object.

## Procedure - Getting Promotion Candidates

The getPromotionTargets(PromotionNotice) method in the promotion request service can be used to get the promotion candidates selected for promotion to the selected target state.

QueryResult qr = MaturityHelper.*getService*().getPromotionTargets(promotionRequest);

## Procedure – Refreshing Promotion Members to Latest Iteration

The promotion candidates could be iterated while processing the promotion request workflow and it may be desired to make sure that all promotion candidates are at the latest iteration prior to setting the objects to the target state. The PromotionNoticeWorkflowHelper can be used to update the promotion objects in the promotion request to the latest iteration. The PromotionNoticeWorkflowHelper can be configured to only update the promotion candidates or all objects which includes the objects in the promotion request baseline and seed objects. The following an example of updating only the promotion candidates to the latest iteration:

PromotionNoticeWorkflowHelper helper = **new** PromotionNoticeWorkflowHelper((PromotionNotice)notice,true/\*Promotion Candidates Only\*/);

helper.updateWithLatestIterations();

See Javadoc for more details on the usage of the PromotionNoticeWorkflowHelper.

## Procedure – Automatically Create New Revisions

The PromotionNoticeWorkflowHelper has a supported API called revisePromotables for revising all promotion candidates with a “Revise” transition in the target state. The “Creator” of the new revision can be specified along with the locale to be used for any messages for promotion candidates that are not revisable.

The API also supports revising only the promotion candidates that result in a revision in a new revision series.

TODO: Add information about configuring the preferences.

The PromotionNoticeWorkflowHelper.revisePromotables API returns a UIValidationResultSet which contains set of validation results for each promotion candidate. Each validation result has a status to indicate if a particular promotion candidate was eligible to be revised. The result also includes the promotion candidate and a feedback message for promotion candidates that were not eligible to be revised. To assist with processing of the validation result set there is an API on the PromotionTargetsReviseValidator called getReviseResultSetMessage which takes in the validation result and specified locale in order to generate a message of all the promotion candidates and the reason why they were not eligible for revise.

See Javadoc on the PromotionNoticeWorkflowHelper.revisePromotables API and PromotionTargetsReviseValidator class for more details.