Sample Code for Asset Listeners (Including Asset Approvals) and Publishing Event Listeners (Doc ID 1911760.1)

# Applies to

Oracle WebCenter Sites - Version 11.1.1.8.0 and later

Information in this document applies to any platform.

# Goal

This document will describe how to register and implement Asset Listeners including Asset Approvals and Publishing Event Listeners.

NOTE: functionality for approval/unapproval events in Asset Event Listeners and the task PublishingTasks.NOTEPUBLISH in Publishing Event Listeners were introduced in WebCenter Sites 11.1.1.8.0 Patch 6.

# Solution

To register an Asset Listener, a row must be added into the 'AssetListener\_reg' table.

AssetListener\_reg

-----------------

listener = Fully qualified class name that implements AssetEventListener. For example: com.fatwire.realtime.event.sample.SampleApprovalListener blocking ='Y' indicates that the listener is blocking (runs synchronously with the thread that generated the event). 'N' indicates that the listener is non blocking (runs in a separate thread).

To register a Publishing Event Listener, a row must be added into the 'FW\_PublishingEventRegistry' table.

FW\_PublishingEventRegistry

--------------------------

listener = Fully qualified class name that implements PublishingEventListener. For example: com.fatwire.realtime.event.sample.SamplePublishingAssetsListener

blocking ='Y' indicates that the listener is blocking (runs synchronously with the thread that generated the event). 'N' indicates that the listener is non blocking (runs in a separate thread).

For more information on how to register an event listener, please refer to the following URL in the WebCenter Sites Developer's Guide:

<http://docs.oracle.com/cd/E29542_01/doc.1111/e29634/events_assetapi.htm>

Below are some sample codes to show how these classes are implemented.

SampleApprovalListener.java:

package com.fatwire.realtime.event.sample;

import com.fatwire.assetapi.data.AssetId;

import com.openmarket.basic.event.AbstractAssetEventListener;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import com.openmarket.xcelerate.util.DebugProperties;

import java.sql.Timestamp;

import java.util.Map;

public class SampleApprovalListener extends AbstractAssetEventListener

{

protected static final Log log = LogFactory.getLog(DebugProperties.approvalLoggerName);

@Override

public void assetAdded(AssetId id)

{

}

@Override

public void assetUpdated(AssetId id, Map<String, Object> properties)

{

log.info(" User " + properties.get("user") + " has updated the asset of id:" + id);

}

@Override

public void assetUpdated(AssetId id)

{

}

@Override

public void assetDeleted(AssetId id)

{

}

public void assetApproved(AssetId id, Map<String, Object> properties)

{

log.info(" User " + properties.get("user") + " has approved the asset of id:" + id + " for the target:" + properties.get("targetId") + ".The asset will be added to the approval queue for publish");

}

public void assetUnapproved(AssetId id, Map<String, Object> properties)

{

log.info(" User " + properties.get("user") + " has unapproved the asset of id:" + id + " for the target:" + properties.get("targetId") + ".The asset will be added to the approval queue for publish");

}

}

SamplePublishingAssetsListener.java:

package com.fatwire.realtime.event.sample;

import java.sql.Timestamp;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import COM.FutureTense.CS.Factory;

import COM.FutureTense.Interfaces.FTValList;

import COM.FutureTense.Util.ftMessage;

import com.fatwire.assetapi.data.AssetId;

import com.fatwire.realtime.event.PublishingEvent;

import com.fatwire.realtime.event.PublishingEventListener;

import com.fatwire.realtime.event.PublishingStatusEnum;

import com.fatwire.realtime.util.RealTimeConstants.PublishingTasks;

import com.openmarket.basic.interfaces.AssetException;

import com.openmarket.xcelerate.util.DebugProperties;

/\*\*

\* sample listener for publishing started assets

\*

\* this listener may be invoked multiple times during one publish session

\*

\*/

public class SamplePublishingAssetsListener implements PublishingEventListener {

private static final Log log = LogFactory.getLog(DebugProperties.publishLoggerName);

public void onEvent(PublishingEvent event) {

try{

PublishingTasks task = event.getTaskName();

PublishingStatusEnum status = event.getStatus();

if (PublishingTasks.NOTEPUBLISH.equals(task)) {

if (PublishingStatusEnum.STARTED.equals(status)) {

for (AssetId id : event.getAssetIds()) {

log.info("Publishing (session id:"

+ event.getPubSessionId() + ") started for type:"

+ id.getType() + " and id:" + id.getId()

+ " to target:" + event.getTargetname() + "(id:"

+ event.getTargetId() + ")" + " by user:"

+ " by user:" + event.getUsername()

+ " route slip:" + event.getRouteSlip());

}

}

if (PublishingStatusEnum.DONE.equals(status)) {

for (AssetId id : event.getAssetIds()) {

log.info("Publishing (session id:"

+ event.getPubSessionId()

+ ") successful for type:" + id.getType()

+ " and id:" + id.getId() + " to target:"

+ event.getTargetname() + "(id:"

+ event.getTargetId() + ")" + " by user:"

+ " by user:" + event.getUsername()

+ " route slip:" + event.getRouteSlip());

}

}

if (PublishingStatusEnum.FAILED.equals(status)) {

for (AssetId id : event.getAssetIds()) {

log.info("Publishing (session id:" + event.getPubSessionId()

+ ") failed for type:" + id.getType()

+ " and id:" + id.getId()

+ " to target:" + event.getTargetname()

+ "(id:" + event.getTargetId() + ")"

+ " by user:" + event.getUsername()

+ " route slip:" + event.getRouteSlip());

}

}

}

} catch (Exception e){

log.error("Error in updating info for the asset in SamplePublishingAssetsListener",e);

}

}

}