1. Hide and show checkbox

selectionchanged:

function(dialog,val,isChecked)

{

var panel = dialog.findParentByType("panel");

var link = panel.getComponent("linkid");

isChecked ? link.show() : link.hide();

}

1. Hide and show tabs

Selectionchanged:

function(box,value)

{

if(value == 'male'){

box.findParentByType("tabpanel").unhideTabStripItem(1);

box.findParentByType("tabpanel").hideTabStripItem(2);}

else

{

box.findParentByType("tabpanel").hideTabStripItem(1);

box.findParentByType("tabpanel").unhideTabStripItem(2);

}}

Afterrender

function(box,dialog)

{

box.findParentByType("tabpanel").hideTabStripItem(1);

box.findParentByType("tabpanel").hideTabStripItem(2);

}

1. Hide and show tabs based on template- listener inside dialog

Afterrender

function(dialog)

{

var temp= CQ.HTTP.get( CQ.HTTP.externalize(CQ.utils.WCM.getPagePath()+'/jcr:content/cq:template'));

var template = ( temp.body );

alert(template);

if(template == '/apps/aemtasks/templates/firsttemplate')

{

alert("in if condition");

dialog.findByType('tabpanel')[0].hideTabStripItem(1);

}

else if(template == '/apps/aemtasks/templates/secondtemplate' ){

alert("in else if condition");

dialog.findByType('tabpanel')[0].hideTabStripItem(2);

}

else{

alert("in else condition");

}}

1. Multicomposite field

<%--

Multi composite field component.

--%><%

%><%@include file="/libs/foundation/global.jsp"%><%

%><%@page session="false" %><%

%><%

// TODO add you code here%>

<%@page import="com.aem.aemtasks.MTMultiCompositeFieldHelper"%>

<cq:includeClientLib categories="clientlibs.multicomposite"/>

<%

com.aem.aemtasks.MTMultiCompositeFieldHelper mtmcompositefield=new com.aem.aemtasks.MTMultiCompositeFieldHelper();

String[] studentdetails={"studentname","studentimage","studenthobbies"};

%>

<c:if test="${not empty currentNode}">

<c:set var="StudentDetails" value="<%=mtmcompositefield.getMultiCompositeValuesArrayList(currentNode.getPath(),"multicompositefieldvalues",studentdetails,slingRequest)%>" />

</c:if>

<c:forEach var="details" items="${StudentDetails}">

Student Name: <c:out value="${details.studentname}" /><br />

Student photo:<img src="${details.studentimage}" /> <br />

Student hobbies: <c:out value="${details.studenthobbies}" />

</c:forEach>

Multicomposite using JSTL

<c:set var="originCities" value='<%=currentNode.getNode("origincity")%>'/>

<c:set var="destinationCities" value='<%=currentNode.getNode("destinationcity")%>'/>

<select id="origin">

<c:forEach var="item" items="${originCities.nodes}">

<option value="${item.properties.value}">${item.properties.text}</option>

</c:forEach>

</select>

<select id="destination">

<c:forEach var="item" items="${destinationCities.nodes}">

<option value="${item.properties.value}">${item.properties.text}</option>

</c:forEach>

</select>

1. Page publish

<%@include file="/libs/foundation/global.jsp"%><%

%><%@page session="false" import="java.util.Date,java.text.SimpleDateFormat

<c:set var="LastReplicatedAction" value="<%=pageProperties.get("cq:lastReplicationAction") %>" />

<c:set var="LastReplicationDate" value="<%=pageProperties.get("cq:lastReplicated")%>" />

<% Date date=pageProperties.get("cq:lastReplicated",Date.class);

SimpleDateFormat formatter=new SimpleDateFormat("dd/MM/yyyy");

String dateFormat=formatter.format(date);

%>

<c:choose>

<c:when test="${LastReplicatedAction == 'Deactivate'}">

<script>

alert("page not published");

</script>

</c:when>

<c:otherwise>

<script>

alert("page is published");

</script>

</c:otherwise>

</c:choose>

1. Multicomposite iteration

<%@include file="/libs/foundation/global.jsp"%>

<form action="/table/refinefunction" method="post">

<div style="width:100%;">

<div style="float:left; width:40%;">

<select id="originCity" style="width:250px">

<c:set var="node" value="<%=currentNode.getNodes()%>"/>

<c:forEach var="childnode" items="${node}" varStatus="totalNodes">

<c:if test="${childnode.name eq 'origincity'}">

<c:forEach var="subchildnode" items="${childnode.nodes}" varStatus="tabs">

<c:set var="tabPropIter" value="${subchildnode.properties}" />

<c:forEach var="tabProperty" items="${tabPropIter}">

<c:if test="${tabProperty.name eq 'origincitytext'}">

<c:set var="text" value="${tabProperty.string}" />

</c:if>

<c:if test="${tabProperty.name eq 'origincityvalue'}">

<c:set var="value" value="${tabProperty.string}" />

</c:if>

</c:forEach>

<option value=${value}>${text}</option>

</c:forEach>

</c:if>

</c:forEach>

</select>

</div>

<div style="float:left; width:40%;">

<select id="destinationCity" style="width:250px">

<c:set var="node" value="<%=currentNode.getNodes()%>"/>

<c:forEach var="childnode" items="${node}" varStatus="totalNodes">

<c:if test="${childnode.name eq 'destinationcity'}">

<c:forEach var="subchildnode" items="${childnode.nodes}" varStatus="tabs">

<c:set var="tabPropIter" value="${subchildnode.properties}" />

<c:forEach var="tabProperty" items="${tabPropIter}">

<c:if test="${tabProperty.name eq 'destinationcitytext'}">

<c:set var="text" value="${tabProperty.string}" />

</c:if>

<c:if test="${tabProperty.name eq 'destinationcityvalue'}">

<c:set var="value" value="${tabProperty.string}" />

</c:if>

</c:forEach>

<option value=${value}>${text}</option>

</c:forEach>

</c:if>

</c:forEach>

</select>

</div>

<div style="float:left; width:20%;">

<input style="height:25px;width:120px" type="button" value="REFINE"/>

</div>

</div>

</form>

1. Touch UI hide and show tabs

(function (document, $, ns) {

"use strict";

$(document).on("selected", "#listfrom", function (e) {

// console.log("Clicked..");

//$(document).on('change', 'select[name="./languages"]', function() {

//alert("Clicked..");

if($('select[name="./languages"]').val() == "telugu")

{

$('a[aria-controls="setEnglish"]').hide();

$('a[aria-controls="setTelugu"]').show();

console.log("TELUGU..");

}

else

{

$('a[aria-controls="setTelugu"]').hide();

$('a[aria-controls="setEnglish"]').show();

console.log("ENGLISH..");

}

//}); });

})(document, Granite.$, Granite.author);

1. Touch UI Hide and show field

(function ($, $document) {

"use strict";

$document.on("dialog-ready", function() {

console.log("inside Clicked..");

if($('select[name="./options"]').val() == "hide")

{

$("#associatename").parent().find( "label" ).css( "display", "none" );

$("#associatename").hide();

console.log("hide..");

}

else

{

$("#associatename").parent().find( "label").css( "display", "block");

$("#associatename").show();

console.log("show..");}

$(document).on("selected", "#hideandshow", function (e){

console.log("inside selected..");

if($('select[name="./options"]').val() == "hide")

{

$("#associatename").parent().find( "label" ).css( "display", "none" );

$("#associatename").hide();

console.log("hide..");

}

else

{

$("#associatename").parent().find( "label").css( "display", "block");

$("#associatename").show();

console.log("show..");

}});});

})($, $(document));

1. Multocomposite using JSON

%><%

// TODO add you code here

%>

multifield<br>

<%

JSONObject obj = null;

Multicomposite multi = new Multicomposite();

Value [] values = multi.getMultiCompositeList(currentNode);

for (Value val : values) {

obj = new JSONObject(val.getString());

%>

Page : <b><%= obj.get("page") %></b><br>

URL : <b><a href="<%= obj.get("path") %>.html" target="\_blank"><%= obj.get("path") %></a></b><br>

<%

}%>

JAVA Class

package com.adobe.cq;

import javax.jcr.Property;

import javax.jcr.Node;

import javax.jcr.RepositoryException;

import javax.jcr.Value;

public class Multicomposite {

public Value[] getMultiCompositeList(Node compNode) throws RepositoryException

{

Property property = null;

Value[] values = null;

if (compNode.hasProperty("iItems")) {

property=compNode.getProperty("iItems");

}

if (property != null) {

if (property.isMultiple()) {

values = property.getValues();

} else {

values = new Value[1];

values[0] = property.getValue();

}

}

else

{

System.out.println(" Add values in dialog");

}

return values;

}

}

1. Multicomposite using ajax call

<%--

ServletCalling component.

--%><%

%><%@include file="/libs/foundation/global.jsp"%><%

%><%@page session="false" %><%

%><%

// TODO add you code here

%>

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

<form id="testForm">

First name: <input type="text" id="firstName"><br><br>

Last name: <input type="text" id="lastName">

<br><br>

<input type="submit" id="formid" value="Submit">

</form>

<div id="json"></div>

<script>

$("#testForm").on("submit",function(){

var myFirst= $('#firstName').val();

var myLast= $('#lastName').val();

console.log(myFirst);

console.log(myLast);

$.ajax({

type: 'POST',

url:'/bin/servlet/exampleservlet',

data:'hello='+ myFirst+'&lastname='+ myLast,

success: function(msg){

//console.log("msg"+msg);

var json = jQuery.parseJSON(msg);

//console.log("json"+json);

var firstName = json.firstname;

var lastName = json.lastname;

$("#json").append(firstName);

//$("#json").html(lastName);

}

});

});

</script>

Servlet Class

package com.adobe.cq;

import java.io.IOException;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Properties;

import org.apache.felix.scr.annotations.Property;

import org.apache.felix.scr.annotations.Service;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingAllMethodsServlet;

import org.apache.sling.commons.json.JSONException;

import org.apache.sling.commons.json.JSONObject;

@Component(immediate = true)

@Service(value = javax.servlet.Servlet.class)

@Properties({

@Property(name = "service.description", value = "Example Servlet"),

@Property(name = "service.vendor", value = "My Project"),

@Property(name = "sling.servlet.paths", value = "/bin/servlet/exampleservlet"),

@Property(name = "sling.servlet.methods", value = "POST") })

public class Servletsample extends SlingAllMethodsServlet{

private static final long serialVersionUID = -6338700807921957971L;

private static Logger LOGGER = Logger.getLogger("Servletsample.Class");

@Override

protected void doPost(final SlingHttpServletRequest request,final SlingHttpServletResponse response) throws ServletException,IOException{

try {

String FirstName = request.getParameter("hello");

String LastName = request.getParameter("lastname");

LOGGER.info(FirstName);

LOGGER.info("Hello Servlet");

LOGGER.info(LastName);

JSONObject obj=new JSONObject();

obj.put("firstname",FirstName);

obj.put("lastname",LastName);

// TODO Auto-generated catch block

String jsonData = obj.toString();

response.getWriter().write(jsonData);

}

catch (JSONException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}}}

1. Sightly Example

<p>This is your AEM HTML Template Language component:</p>

<div data-sly-use.sightlyTextObject="com.cts.SightlyComponent">

<h1>${sightlyTextObject.SightlyBean.headingText}</h1>

<p>${sightlyTextObject.SightlyBean.description}</p>

</div>

JAVA Class

Bean class

**package** com.adobe.cq;

**public** **class** SightlyBean {

**private** String headingText;

**private** String description;

**public** String getHeadingText()

{

**return** headingText;

}

**public** **void** setHeadingText(String headingText)

{

**this**.headingText=headingText;

}

**public** String getDescription()

{

**return** description;

}

**public** **void** setDescription(String description)

{

**this**.description=description;

}

}

Implementation class

**package** com.adobe.cq;

**import** com.adobe.cq.sightly.WCMUse;

**import** javax.jcr.Node;

**public** **class** SightlyComponent **extends** WCMUse{

**private** SightlyBean sightlyTextBean = **null**;

//private static Logger LOGGER = Logger.getLogger("SightlyComponent.Class");

@Override

**public** **void** activate() **throws** Exception {

Node currentNode = getResource().adaptTo(Node.**class**);

sightlyTextBean = **new** SightlyBean();

**if**(currentNode.hasProperty("jcr:Heading")){

sightlyTextBean.setHeadingText(currentNode.getProperty("jcr:Heading").getString());

//LOGGER.info(currentNode.getProperty("./jcr:Heading").getString());

}

**if**(currentNode.hasProperty("jcr:description")){

sightlyTextBean.setDescription(currentNode.getProperty("jcr:description").getString());

//LOGGER.info(currentNode.getProperty("./jcr:description").getString());

}

}

**public** SightlyBean getSightlyBean() {

**return** **this**.sightlyTextBean;

}

}

1. Sightly Function

<p>This is your AEM HTML Template Language component:</p>

<div data-sly-use.sightlyFunctionObject="com.sightly.function.SightlyFunction">

<h1>${sightlyFunctionObject.lowerCaseTitle}</h1>

<p>${sightlyFunctionObject.lowerCaseDescription}</p>

</div>

JAVA Class

package com.sightly.function;

import com.adobe.cq.sightly.WCMUse;

import javax.jcr.Node;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class SightlyFunction extends WCMUse{

protected final Logger log = LoggerFactory.getLogger(this.getClass());

private String lowerCaseTitle;

private String lowerCaseDescription;

@Override

public void activate() throws Exception {

log.info("inside activate method");

Node currentNode = getResource().adaptTo(Node.class);

if(currentNode.hasProperty("heading")){

lowerCaseTitle=currentNode.getProperty("heading").getString();

log.info("Ttile"+lowerCaseTitle);

//LOGGER.info(currentNode.getProperty("./jcr:Heading").getString()); }

if(currentNode.hasProperty("description")){

lowerCaseDescription=currentNode.getProperty("description").getString();

log.info("Description"+lowerCaseDescription);

//LOGGER.info(currentNode.getProperty("./jcr:description").getString());

}}

public String getLowerCaseTitle() {

log.info("inside title method");

return lowerCaseTitle;

}

public String getLowerCaseDescription()

{

log.info("inside description method");

return lowerCaseDescription;

}}

1. Sightly Query Builder

<p>query builder</p>

<div data-sly-use.sightlyQueryBuilder="com.query.builder.Search">

<ul data-sly-list="${sightlyQueryBuilder.hyperLinks}"><li>${item}</li> </ul>

</div>

JAVA Class

package com.query.builder;

import javax.jcr.Node;

import com.adobe.cq.sightly.WCMUse;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.jcr.Session;

import com.day.cq.search.QueryBuilder;

import com.day.cq.search.Query;

import com.day.cq.search.PredicateGroup;

import com.day.cq.search.result.SearchResult;

import com.day.cq.search.result.Hit;

public class Search extends WCMUse{

protected final Logger log = LoggerFactory.getLogger(this.getClass());

private Session session;

private String searchWord;

Map<String, String> map = new HashMap<String, String>();

List<String> hyperLinks = new ArrayList<String>();

@Override

public void activate() throws Exception {

Node currentNode = getResource().adaptTo(Node.class);

String str=currentNode.toString();

log.info("currentNode"+str);

QueryBuilder queryBuilder;

log.info("inside query builder activate");

try

{

if(currentNode.hasProperty("searchterm"))

{

searchWord=currentNode.getProperty("searchterm").getValue().getString();

log.info("searchterm"+searchWord);

queryBuilder =getResourceResolver().adaptTo(QueryBuilder.class);

session = getResourceResolver().adaptTo(Session.class);

map.put("path","/content");

map.put("type","cq:Page");

map.put("group.p.or", "true");

map.put("group.1\_fulltext", searchWord);

map.put("group.1\_fulltext.relPath", "jcr:content");

map.put("group.2\_fulltext", searchWord);

map.put("group.2\_fulltext.relPath", "jcr:content/@cq:tags");

Query query = queryBuilder.createQuery(PredicateGroup.create(map), session);

map.put("p.offset", "0"); // same as query.setStart(0) below

map.put("p.limit", "20");

SearchResult result = query.getResult();

int hitsPerPage = result.getHits().size();

log.info("hitsPerPage"+hitsPerPage);

for (Hit hit : result.getHits()) {

String path = hit.getPath();

log.info("path"+path);

//Create a result element

hyperLinks.add(path);

}

}

}

catch(Exception e)

{

log.info("Something went wrong with session .. {}", e);

}

}

public List<String> getHyperLinks()

{

return hyperLinks;

}

}

1. Query Builder

JAVA Class

String OriginCity = null;

                String DestinationCity = null;

                OriginCity = request.getParameter("Origin");

                DestinationCity = request.getParameter("Destination");

                String path=currentPage.getPath();

    String mainCabinOne = "";

    String mainCabinTwo = "";

    String deltaBusiness = "";

                QueryBuilder queryBuilder = null;

                if(OriginCity!=null && DestinationCity!=null) {

        if(!OriginCity.equals("select") && !DestinationCity.equals("select")){

                                queryBuilder =resourceResolver.adaptTo(QueryBuilder.class);

                                Session session = resourceResolver.adaptTo(Session.class);

                                Map<String, String> map = new HashMap<String, String>();

                                map.put("path",path+"/jcr:content/mainPar/tablefilterjson/resultset");

                                map.put("type","nt:unstructured");

                                map.put("1\_property","resultsetorigincity");

                                map.put("1\_property.value",OriginCity);

                                map.put("2\_property","resultsetdestinationcity");

                                map.put("2\_property.value",DestinationCity);

                                Query query = queryBuilder.createQuery(PredicateGroup.create(map), session);

                                query.setStart(0);

                                query.setHitsPerPage(20);

                                SearchResult result = query.getResult();

                                int hitsPerPage = result.getHits().size();

                if(hitsPerPage!=0){%>

<table style="width:100%;background-color:#061f47;border:1px solid black">

<tr style="border:1px solid black"><th style="border:1px solid black">Main Cabin Bag One</th><th style="border:1px solid black">Main Cabin Bag Two</th><th style="border:1px solid black">Delta First Business</th></tr>

    <% for(Hit hit: result.getHits())

        Node resultNode = hit.getNode();

                                if(resultNode.hasProperty("maincabinbag1")){

                                                mainCabinOne = resultNode.getProperty("maincabinbag1").getString();}

        if(resultNode.hasProperty("maincabinbag2")){

                                                mainCabinTwo = resultNode.getProperty("maincabinbag2").getString();}

        if(resultNode.hasProperty("deltaonefirstbusiness")){

                                                deltaBusiness = resultNode.getProperty("deltaonefirstbusiness").getString();}%>

<tr style="border:1px solid black"><td style="border:1px solid black"><%=mainCabinOne%></td><td style="border:1px solid black"><%=mainCabinTwo%></td><td style="border:1px solid black"><%=deltaBusiness%></td></tr>

<% }}

else{%>

<h3>No Results Found for the selected Origin and Destination Regions</h3>

<%}%>

</table>

<%}else {%>

<h3>you have not selected origin and destination regions</h3>

<% }}%>

1. ResourceResolver interface

<%@include file="/libs/foundation/global.jsp"%><%

%><%@page session="false"

<% com.sightly.resource.resolver.Query wfService = sling.getService(com.sightly.resource.resolver.Query.class);

%>

<h2>Use the Sling API to get title of the resource at /content/geometrixx/en/service</h2>

<h3>The title of the page is: <%= wfService.getJCRData("/content/geometrixx/en/services")%></h3>

Interface

**package** com.sightly.resource.resolver;

**public** **interface** Query {

**public** String getJCRData(String location);

}

Implimentation class

**package** com.sightly.resource.resolver;

**import** org.apache.sling.api.resource.ResourceResolverFactory ;

**import** org.apache.sling.api.resource.ResourceResolver;

**import** org.apache.sling.api.resource.Resource;

**import** org.apache.felix.scr.annotations.\*;

**import** com.day.cq.wcm.api.Page;

@Component

@Service

**public** **class** QueryImp **implements** Query{

@Reference

**private** ResourceResolverFactory resolverFactory;

**public** String getJCRData(String location)

{

**try**{

ResourceResolver resourceResolver = resolverFactory.getAdministrativeResourceResolver(**null**);

Resource res = resourceResolver.getResource(location);

Page page = res.adaptTo(Page.**class**);

String title = page.getTitle();

**return** title;

}

**catch** (Exception e)

{

e.printStackTrace() ;

}

**return** **null**;

}

}

1. Email Sending workflow

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

<http://localhost:4502/system/console/configMgr-> Day CQ email service

package com.workflow.training;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Service;

import org.apache.felix.scr.annotations.Properties;

import org.apache.felix.scr.annotations.Property;

import org.apache.felix.scr.annotations.Reference;

import org.osgi.framework.Constants;

import com.adobe.granite.workflow.WorkflowException;

import com.adobe.granite.workflow.WorkflowSession;

import com.adobe.granite.workflow.exec.WorkItem;

import com.adobe.granite.workflow.exec.WorkflowProcess;

import com.adobe.granite.workflow.metadata.MetaDataMap;

import com.day.cq.mailer.MessageGateway;

import com.day.cq.mailer.MessageGatewayService;

import org.apache.commons.mail.Email;

import org.apache.commons.mail.SimpleEmail;

@Component

@Service

@Properties({

@Property(name = Constants.SERVICE\_DESCRIPTION, value = "Test Email workflow process implementation."),

@Property(name = Constants.SERVICE\_VENDOR, value = "Adobe"),

@Property(name = "process.label", value = "Test Email Workflow Process") })

public class CustomStep implements WorkflowProcess{

protected final Logger log = LoggerFactory.getLogger(this.getClass());

//Inject a MessageGatewayService

@Reference

private MessageGatewayService messageGatewayService; - service

public void execute(WorkItem item, WorkflowSession wfsession,MetaDataMap args) throws WorkflowException {

try

{

log.info("Here in execute method"); //ensure that the execute method is invoked

//Declare a MessageGateway service

MessageGateway<Email> messageGateway;

//Set up the Email message

Email email = new SimpleEmail();

//Set the mail values

String emailToRecipients = "bhuvaneswari.arimireddy@cognizant.com";

String emailCcRecipients = "bhuvaneswari.arimireddy@cognizant.com";

email.addTo(emailToRecipients);

email.addCc(emailCcRecipients);

email.setSubject("AEM Custom Step");

email.setFrom("bhuvaneswari.arimireddy@cognizant.com");

email.setMsg("This message is to inform you that the CQ content has been deleted");

//Inject a MessageGateway Service and send the message

messageGateway = messageGatewayService.getGateway(Email.class);

// Check the logs to see that messageGateway is not null

messageGateway.send((Email) email);

}

catch (Exception e)

{

e.printStackTrace() ;

}

}

}

1. Create Node Workflow

package com.sling.createnode;

import java.io.IOException;

import java.io.PrintWriter;

import org.apache.sling.api.servlets.SlingAllMethodsServlet;

import org.apache.sling.api.\*;

import javax.jcr.Node;

import javax.jcr.RepositoryException;

import javax.jcr.Session;

import org.apache.felix.scr.annotations.\*;

import org.apache.sling.jcr.api.SlingRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.day.cq.commons.jcr.JcrUtil;

import javax.servlet.ServletException;

@Component

@Service

@Properties({

@Property(name = "service.description", value = "Create Node"),

@Property(name = "service.vendor", value = "My Create Node Servlet"),

@Property(name = "sling.servlet.paths", value = "/bin/slingservlet"),

@Property(name = "sling.servlet.methods", value = "GET,POST") })

public class CreateNode extends SlingAllMethodsServlet{

private static final long serialVersionUID = 1L;

private static final String CONTENT\_PATH = "/content";

protected final Logger log = LoggerFactory.getLogger(this.getClass());

@Reference

private SlingRepository repository; - service

protected void doPost(SlingHttpServletRequest request,SlingHttpServletResponse response)throws ServletException,

IOException {

PrintWriter out=response.getWriter();

Session session=null;

String Name=request.getParameter("firstName");

log.info("Name"+Name);

String ID=request.getParameter("idNumber");

log.info("ID"+ID);

String Hobbies=request.getParameter("hobbies");

log.info("Hobbies"+Hobbies);

try{

session = repository.loginAdministrative(null);

Node mainpath = JcrUtil.createPath(CONTENT\_PATH + "/" + Name,

"cq:Page", session);

Node subnode = JcrUtil.createPath(mainpath.getPath() + "/SubNode",

"nt:unstructured", session);

subnode.setProperty("id",ID);

subnode.setProperty("hobbie",Hobbies);

session.save();

out.println("That went well...");

out.flush();

out.close();

}

catch (RepositoryException e) {

// TODO Auto-generated catch block

out.println("nooooooo...");

out.flush();

out.close();

e.printStackTrace();

}

} }

1. Approval workflow

package com.workflow.approval;

import javax.jcr.Node;

import com.adobe.granite.workflow.WorkflowException;

import com.adobe.granite.workflow.WorkflowSession;

import com.adobe.granite.workflow.exec.WorkItem;

import com.adobe.granite.workflow.exec.WorkflowProcess;

import com.adobe.granite.workflow.metadata.MetaDataMap;

import org.apache.felix.scr.annotations.\*;

import org.apache.sling.api.resource.ResourceResolver;

import org.apache.sling.api.resource.ResourceResolverFactory ;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.jcr.Session;

@Component

@Service

@Properties({

@Property(name="description" ,value="Workflow description"),

@Property(name = "process.label", value = "Approval Work Flow")

})

public class ApprovalWorkflow implements WorkflowProcess{

@Reference

ResourceResolverFactory resourceResolverFactory; - Service

Logger log= LoggerFactory.getLogger(ApprovalWorkflow.class);

public void execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaMap)

throws WorkflowException {

// workItem gives wf\_id,workitem\_id,payload,payload\_type

try {

final String payloadPath = workItem.getWorkflowData().getPayload().toString();

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

log.info("workItem" + workItem );

log.info("payloadPath" + payloadPath );

// get the node for the workflow payload

final Node payloadNode = session.getNode(payloadPath);

log.info("payloadNode" +payloadNode );

final String payloadNodeStr=payloadNode.toString();

final String payladFinal= payloadNodeStr +"/jcr:content";

ResourceResolver resourceResolver = resourceResolverFactory.getAdministrativeResourceResolver(null);

Node node = resourceResolver.getResource(payladFinal).adaptTo(Node.class);

String str = node.toString();

log.info("str"+str);

node.setProperty("approved", "content approved");

// add the node

final Node somenode = node.addNode("somenode");

somenode.setProperty("myproperty", "my property value");

} catch (Exception e) {

e.printStackTrace();

}

}}

In Config manager- WCM authoring UI mode Service to change Touch and Classic modes

Box Component

Table filter

Transperfect

Richtext

Footnote defects

Sling models

<sly data-sly-test.nodepath="${currentNode.Path}"></sly>

<sly data-sly-test.nodename="properties"></sly>

<sly data-sly-use.tableValues="${'com.jh.jhas.core.models.EquityInvestor' @nodePath=nodepath,nodeName=nodename}">

            <sly data-sly-test="${tableValues.multivalues}">

<sly data-sly-list="${tableValues.multivalues}">

package com.jh.jhas.core.models;

import java.util.ArrayList;

import java.util.HashMap;

import javax.annotation.PostConstruct;

import javax.inject.Inject;

import javax.jcr.Node;

import javax.jcr.RepositoryException;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.resource.Resource;

import org.apache.sling.api.resource.ResourceResolver;

import org.apache.sling.models.annotations.Model;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.jh.jhas.core.helper.MultiCompositeHelper;

@Model(adaptables = SlingHttpServletRequest.class)

public class EquityInvestor {

      private static Logger LOG = LoggerFactory.getLogger(EquityInvestor.class);

      @Inject

      private String nodePath;

      @Inject

      private String nodeName;

      @Inject

      private SlingHttpServletRequest request;

      private ArrayList<HashMap<String, String>> multivalues;

      @PostConstruct

      protected void init() throws RepositoryException

      {

            ResourceResolver resourceResolver=request.getResourceResolver();

            LOG.info("resourceResolver"+resourceResolver);

            Resource resource=resourceResolver.getResource(nodePath);

            LOG.info("resource"+resource);

            Node node=resource.adaptTo(Node.class);

            LOG.info("resource"+resource);

            String[] listValue={"annualreturn","annualreturnvalue","growth","growthvalue"};

      multivalues=MultiCompositeHelper.getMultiCompositeValuesArrayList(node.getPath(),nodeName,listValue,request);

      }

      public ArrayList<HashMap<String, String>> getMultivalues() {

            return multivalues;

      }

}

package com.jh.jhins.servlet;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.Servlet;

import javax.servlet.ServletException;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Properties;

import org.apache.felix.scr.annotations.Property;

import org.apache.felix.scr.annotations.Service;

import org.apache.http.HttpResponse;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingAllMethodsServlet;

import org.apache.sling.commons.json.JSONArray;

import org.apache.sling.commons.json.JSONObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Component(immediate = true, metatype = true)

@Service(Servlet.class)

@Properties({ @Property(name = "service.description", value = "NewBusinessGetPolicyStatusServlet"),

              @Property(name = "sling.servlet.paths", value = { "/bin/sling/NewBusinessGetPolicyStatus" }),

              @Property(name = "service.vendor", value = "JHINS"),

              @Property(name = "sling.servlet.methods", value = "POST", propertyPrivate = true) })

public class NewBusinessGetPolicyStatusServlet extends SlingAllMethodsServlet {

       private static final long serialVersionUID = 1L;

private static final Logger LOG = LoggerFactory.getLogger(NewBusinessGetPolicyStatusServlet.class);

       protected final void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)

                     throws ServletException, IOException {

              this.doPost(request, response);

       }

       protected final void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response)

                     throws ServletException, IOException  {

              String insuredCaseId=request.getParameter("insuredCaseId");

              String insuredCaseCode=request.getParameter("insuredCaseCode");

              /\*JSONObject jsonObj = new JSONObject();

              HttpResponse resp=null;

                                 PrintWriter out = response.getWriter();

              try{

                                               jsonObj.put("policyId","1-10DG9R");

                                               jsonObj.put("userRole","SuperUser");

                                               jsonObj.put("UUID","1000001123456");

                                               jsonObj.put("PolNumber","57638884");

                                               jsonObj.put("ProductLineCode","100");

                                               jsonObj.put("partyRole","insured");

                                               DefaultHttpClient httpClient = new DefaultHttpClient();

                                               HttpPost postRequest = new HttpPost("<https://b04amxd02.dev.manulifeusa.com:7081/Insurance/NewBusinessPolicy/GetPolicyStatus>");

                                               postRequest.addHeader("Application","BusinessPolicy");

                                               //String encodedCredentials = new String(Base64.encodeBase64("reddbhu:Bhuvana@2017".getBytes("UTF-8")));

                                               postRequest.addHeader("Authorization","Basic cmVkZGJodTpCaHV2YW5hQDIwMTc=");

                                               StringEntity input = new StringEntity(jsonObj.toString());

                                               input.setContentType("application/json");

                                               postRequest.setEntity(input);

                                               resp = httpClient.execute(postRequest);

                                               LOG.info("output@@@@"+resp);

                                               String respContent = org.apache.http.util.EntityUtils.toString(resp.getEntity());

                                               LOG.info("output"+ respContent);

                                               response.setContentType("application/json");

                                               response.setCharacterEncoding("UTF-8");

                                               out.write(respContent);

                              }

              catch(Exception e){

                     LOG.info("ERROR IN FETCHING VALUES"+e);

              }\*/

Thanks,

Bhuvana