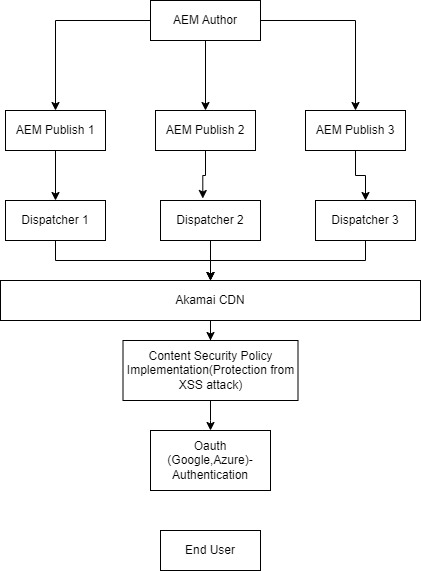
1. *A POC draft/drawing of AEM infrastructure design that meets the criteria.*



1. *Using the latest AEM Archetype compatible to AEM 6.5.*

*OSGI: It’s a configuration can be created using java and values can be updated AEM ConfigManager. It’s a dynamic component system*

*Dispatcher : It’s a load balancing tool, used for security purpose also protect from XSS attack. Supports multiple domain in configuration, rewrite rules, redirecting page based on error code.*

*Apache : It leveareges a Apche Feling, Apache sliing API support.*

1. *Support 4 domain in single app:*

*To support multi tenancy, need to create a new farm for a site based on virtual host in dispatcher configuration.*

*/farms*

*{*

*$include "sg.example.any"*

*$include "my.example.any"*

*$include "uk.example.any"*

*$include “de.example.any”*

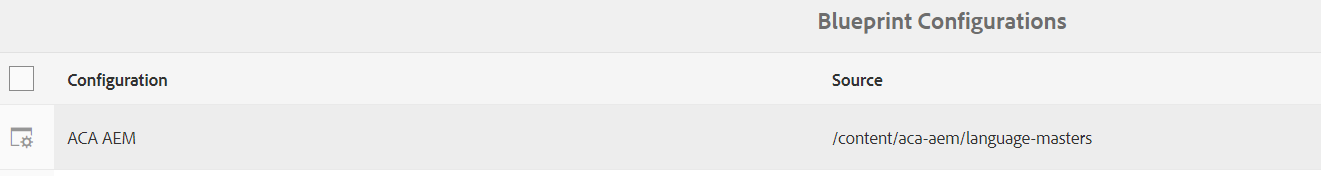
*}*

*/\*\* I haven’t worked on dispatcher. Know basic things\*\*/*

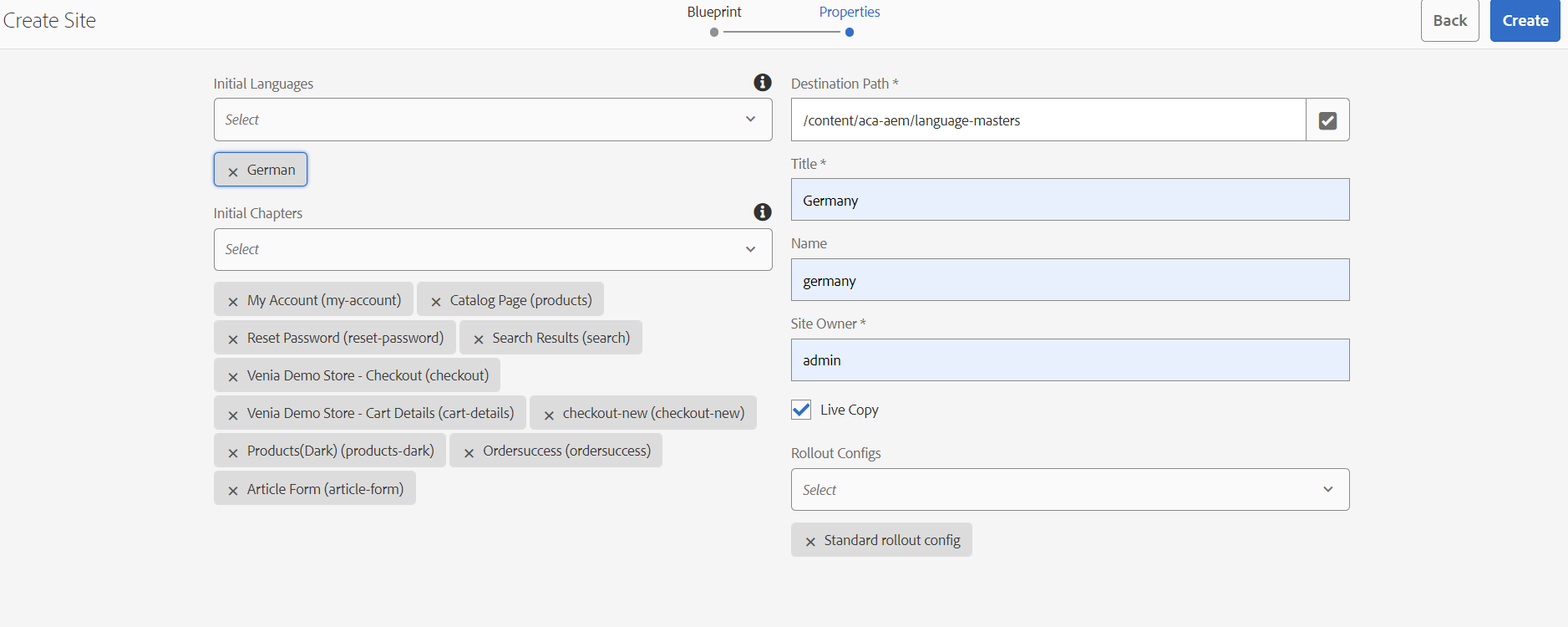
1. *MSM and Live Copy:*

*To create a MSM,*

* *Create a Blueprint of site Language master*



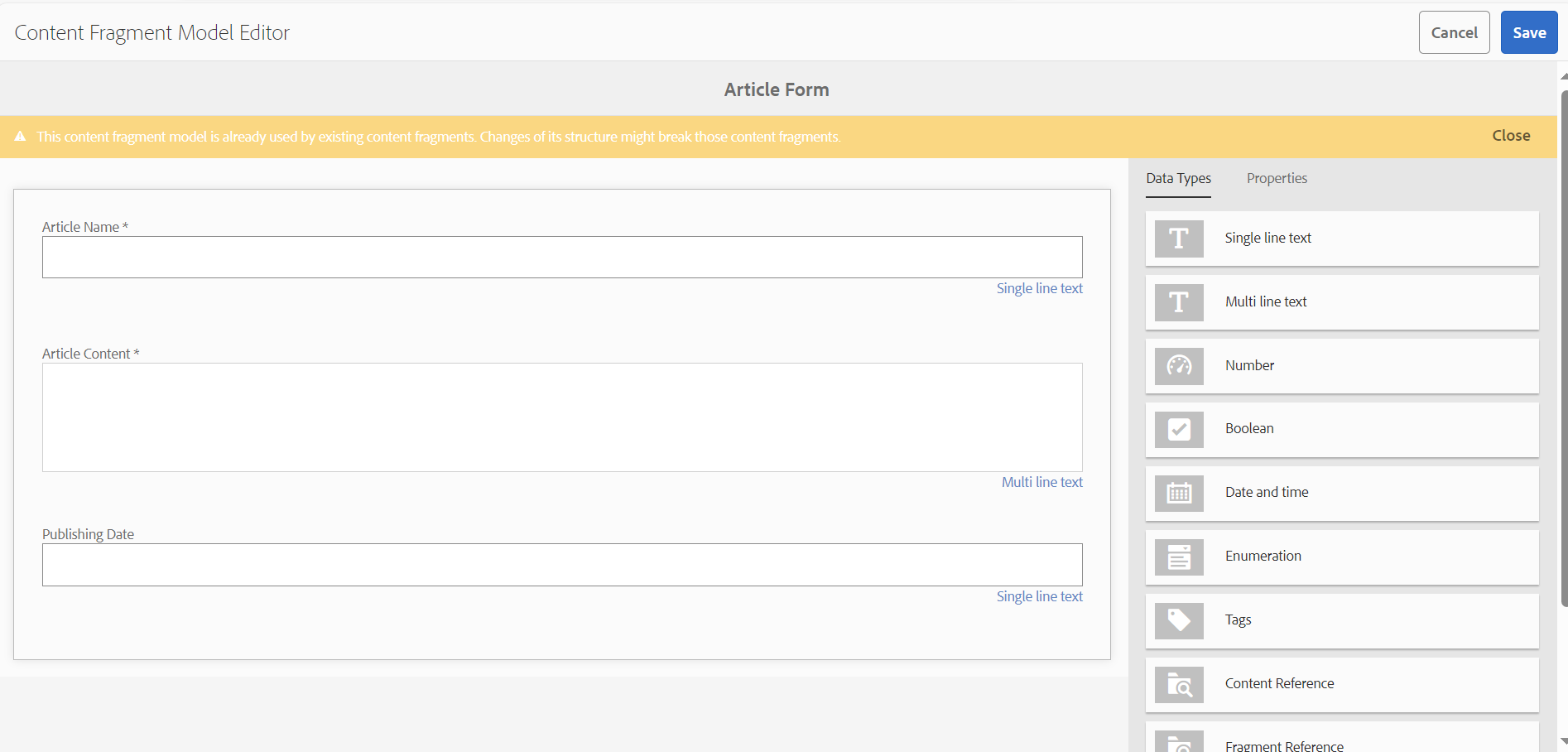
* *Create a Language copy of by selecting a languages, configure the cloud config and select the Rollout Configuration.*
* *Create a Site and provide all the details and select the language , check the Live Copy and select the Standard Rollout Configuration.*



1. *Form Component*

*I have used the OOTB AEM Form Components which is Form Text Component. I have used the ajax call to call backend servlet.*

*Content Fragment Model:*



*Form.js :*

$("#submit\_article" ).click(function() {

var articleName=$('#articleName').val();

var articleContent=$('#articleContent').val();

var publishingDate=$('#publishingDate').val();

//Use JQuery AJAX request to post data to a Sling Servlet

$.ajax({

type: 'POST',

url:'/bin/formcf',

data:{'name' : articleName,'content' : articleContent, 'date':publishingDate},

success: function(msg){

console.log('Content Fragment Created');

}

});

location.reload();

});

*FormCF.java :*

*This file is to get the data from the component and create the Content Fragment based on the value of form.*

package com.aca.aem.core.servlets;

import com.aca.aem.core.utils.ResourceService;

import com.adobe.cq.dam.cfm.\*;

import com.day.cq.dam.api.Asset;

import com.day.cq.dam.api.AssetManager;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.request.RequestParameter;

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

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

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

import org.osgi.framework.Constants;

import org.osgi.service.component.annotations.Component;

import org.osgi.service.component.annotations.Reference;

import javax.servlet.Servlet;

import java.io.IOException;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

@Component(

service= Servlet.class,

property={

Constants.SERVICE\_DESCRIPTION + "=Solr",

"sling.servlet.methods={ GET, POST}",

"sling.servlet.paths=" + "/bin/formcf"

}

)

public class FomCF extends SlingAllMethodsServlet {

@Reference

private ContentFragmentManager fragmentManager;

@Reference

ResourceService resourceService;

@Override

protected void doPost(final SlingHttpServletRequest req,

final SlingHttpServletResponse resp) throws IOException {

String name = req.getParameter("name");

String content = req.getParameter("content");

String date = req.getParameter("date");

try(ResourceResolver resourceResolver = req.getResourceResolver()) {

Resource template = resourceResolver.getResource("/conf/aca-aem/settings/dam/cfm/models/article-form");

Resource path = resourceResolver.getResource("/content/dam/aca-aem/article-form");

ContentFragment cf = null;

if(Objects.nonNull(template)){

FragmentTemplate tpl = template.adaptTo(FragmentTemplate.class);

if(Objects.nonNull(tpl)){

cf = tpl.createFragment(path, name+"cf", name);

Resource fragRes = cf.adaptTo(Resource.class);

if(Objects.nonNull(cf)){

ResourceResolver fragResolver = fragRes.getResourceResolver();

ContentElement articleName = cf.getElement("articleName");

articleName.setContent(name, "text/plain");

ContentElement articleContent = cf.getElement("articleContent");

articleContent.setContent(content, "text/plain");

ContentElement publishingDate = cf.getElement("publishingDate");

publishingDate.setContent(date,"text/plain");

fragResolver.commit();

}

}

}

} catch (ContentFragmentException e) {

throw new RuntimeException(e);

}

resp.setStatus(200);

resp.getWriter().write("Content Fragment is created");

}

}

Resource Service.java :

This is for the resolving resource.

package com.aca.aem.core.utils;

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

public interface ResourceService {

public ResourceResolver getResourceResolver();

}

ResourceServiceImpl.java :

package com.aca.aem.core.utils;

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

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

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

import org.osgi.service.component.annotations.Component;

import org.osgi.service.component.annotations.Reference;

import java.util.HashMap;

import java.util.Map;

@Component(service = ResourceService.class)

public class ResourceServiceImpl implements ResourceService {

@Reference ResourceResolverFactory resolverFactory;

@Override

public ResourceResolver getResourceResolver() {

ResourceResolver resolver = null;

Map<String,Object> param = getServiceParams();

try {

resolver = resolverFactory.getResourceResolver(param);

} catch (LoginException e) {

e.printStackTrace();

}

return resolver;

}

public Map<String,Object> getServiceParams(){

Map<String,Object> param = new HashMap<>();

param.put(ResourceResolverFactory.SUBSERVICE,"acaadmin");

return param;

}

}

*A Job Scheduler will check the date and publish accordingly*

*I have created for job scheduler for every 5 hours, will check the folder in dam. After checking the date of content fragment.*

package com.aca.aem.core.schedulers;

import com.aca.aem.core.utils.ResourceService;

import com.adobe.cq.dam.cfm.ContentElement;

import com.adobe.cq.dam.cfm.ContentFragment;

import com.adobe.cq.dam.cfm.FragmentTemplate;

import com.day.cq.replication.ReplicationActionType;

import com.day.cq.replication.ReplicationException;

import com.day.cq.replication.ReplicationStatus;

import com.day.cq.replication.Replicator;

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

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

import org.apache.sling.commons.scheduler.ScheduleOptions;

import org.apache.sling.commons.scheduler.Scheduler;

import org.osgi.service.component.annotations.\*;

import org.osgi.service.metatype.annotations.AttributeDefinition;

import org.osgi.service.metatype.annotations.AttributeType;

import org.osgi.service.metatype.annotations.Designate;

import org.osgi.service.metatype.annotations.ObjectClassDefinition;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.jcr.Session;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.util.Iterator;

import java.util.Objects;

@Designate(ocd=PublishScheduler.Config.class)

@Component(immediate = true,service=Runnable.class)

public class PublishScheduler implements Runnable {

@ObjectClassDefinition(name="A scheduled task",

description = "Simple demo for cron-job like task with properties")

public static @interface Config {

@AttributeDefinition(

name = "Scheduler name",

description = "Scheduler name",

type = AttributeType.STRING)

String scheduler\_name() default "Publish CF based on Date";

@AttributeDefinition(name = "Cron-job expression")

String scheduler\_expression() default "\* \* \* \* \*";

@AttributeDefinition(

name = "Enable Scheduler",

description = "Enable Scheduler",

type = AttributeType.BOOLEAN)

boolean enable\_scheduler() default true;

@AttributeDefinition(name = "Concurrent task",

description = "Whether or not to schedule this task concurrently")

boolean scheduler\_concurrent() default false;

@AttributeDefinition(name = "A parameter",

description = "Can be configured in /system/console/configMgr")

String damPath() default "/content/dam/aca-aem/article-form";

}

private final Logger logger = LoggerFactory.getLogger(getClass());

private String damPath;

@Reference

private Scheduler scheduler;

@Reference

private Replicator replicator;

@Reference

ResourceService resourceService;

@Activate

protected void activate(final Config config) {

logger.error(" PublishScheduledTask activate method called");

// Execute this method to add scheduler.

addScheduler(config);

damPath = config.damPath();

}

// Add all configurations to Schedule a scheduler depending on name and expression.

public void addScheduler(Config config) {

logger.error("Scheduler added successfully >>>>>>> ");

if (config.enable\_scheduler()) {

ScheduleOptions options = scheduler.EXPR(config.scheduler\_expression());

options.name(config.scheduler\_name());

options.canRunConcurrently(config.scheduler\_concurrent());

// Add scheduler to call depending on option passed.

scheduler.schedule(this, options);

logger.error("Scheduler added successfully name='{}'", config.scheduler\_name());

} else {

logger.error("SimpleScheduledTask disabled");

}

}

public void removeScheduler(Config config) {

scheduler.unschedule(config.scheduler\_name());

}

// On deactivate component it will unschedule scheduler

@Deactivate

protected void deactivate(Config config) {

removeScheduler(config);

}

// On component modification change status will remove and add scheduler

@Modified

protected void modified(Config config) {

removeScheduler(config);

addScheduler(config);

}

// run() method will get call every minute

@Override

public void run() {

getCfPath();

}

public void getCfPath(){

ResourceResolver resourceResolver = resourceService.getResourceResolver();

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

Resource pathResource = resourceResolver.getResource("/content/dam/aca-aem/article-form");

Iterator<Resource> children = pathResource.listChildren();

while(children.hasNext()){

Resource res = children.next();

String cfPath = res.getPath();

if(res.getResourceType().equals("dam:Asset")){

ContentFragment cf = resourceResolver.resolve(cfPath).adaptTo(ContentFragment.class);

String date = cf.getElement("publishingDate").getContent();

boolean checkDate = compareDate(date);

if(checkDate){

Resource cfres = resourceResolver.getResource(cfPath);

ReplicationStatus replicationStatus = cfres.adaptTo(ReplicationStatus.class);

if (replicationStatus != null){

boolean isActivated = replicationStatus.isActivated();

if (!isActivated){

activateCF(session,cfPath);

logger.error("Content Fragment is activated" +cfPath);

}

}

}

}

}

}

public boolean compareDate(String date){

String pattern = "yyyy-MM-dd";

SimpleDateFormat dateFormat = new SimpleDateFormat(pattern);

LocalDate parsedStringDate = LocalDate.parse(date.replace(" ", "T"));

LocalDate currentDate = LocalDate.now();

if (parsedStringDate.isEqual(currentDate)){

return true;

}

return false;

}

public void activateCF(Session session, String path){

try {

replicator.replicate(session, ReplicationActionType.ACTIVATE, path);

} catch (ReplicationException e) {

throw new RuntimeException(e);

}

}

}

*Oauth Implementation in AEM:*

*Updated Adobe Granite OAuth Application and Provider Configuration.*

