**FINAL EXIT TEST SOLUTIONS**

1. **Take a fresh setup of SAP Commerce**

Took the following steps to install fresh setup of Hybris:

* Extract hybris in a new folder
* Open a command line and go to the installer directory
* Run the command : **install.bat -r cx -A local\_property:initialpassword.admin=root**
* This will install the recipe and the **required extensions** to the **localextensions.xml** file as per the **recipe name** we have provided in the command.
* After the installation is done, go to platform directory and run :
* **setantenv.bat**
* **ant clean all**
* **ant initialize**
* **hybrisserver.bat**

1. **Create custom extensions for exit test, with prefix Nag**

* Go to platform directory
* Run : ant modulegen -Dinput.module=accelerator -Dinput.name=NagAcc -Dinput.package=de.hybris.training -Dinput.template=develop
* Remove all the yaccelerator\* extensions from localextensions.xml
* Add new extensions to localextensions.xml
* Reinstall the addons : ant reinstall\_addons -Dtarget.storefront=NagAccstorefront
* Modify local.properties to use NagAccstorefront
* Run ant clean all and ant initialize
* Start the server
* Add 127.0.0.1 localhost electronics.local apparel-uk.local apparel-de.local to host file.

1. **Setup git repository and commit your custom folder**

* In the Toolbar in Intellij,select Git and enable Version Control
* Go to the custom folder and right click and select Git and select Add
* On the extreme top right corner (GIT 2 Arrows & a tick), click on Commit (the tick)
* Write a commit message
* Commit & Push
* In master, define remote, give URL of Git Repo
* And then click Push
* Give authorization details

1. **Take sample data from apparel uk, Change site and store name to Nagarro**

* $lang=en
* UPDATE CMSSite;uid[unique=true];name[lang=$lang];locale[lang=$lang]

;apparel-uk;"Nagarro";en\_GB

* UPDATE BaseStore;uid[unique=true];name[lang=$lang]

;apparel-uk;"Nagarro "

* Include the above impexes in store\_en.impex and site\_en.impex
* Include these files in InitialDataSystemSetup.java so that these impexes automatically run during update/initialization.

1. **Create 4 new apparel products of your choice with proper data – images, prices, stock, name and description. Ensure that these products are visible and can be added to cart on frontend**

* **products.impex**
* **$productCatalog**=apparelProductCatalog  
  **$productCatalogName**=Apparel Product Catalog  
    
  **$catalogVersion**=catalogversion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$productCatalog:Staged]  
  **$supercategories**=supercategories(code, $catalogVersion)  
  **$baseProduct**=baseProduct(code,$catalogVersion)  
  **$approved**=approvalstatus(code)[default='approved']  
    
  # Insert Products in ApparelProduct for Exit Test  
  INSERT\_UPDATE ApparelProduct;code[unique=true];$catalogVersion;unit(code);$approved;ean;genders(code)  
  ;1200;;pieces;;989514145;MALE,FEMALE;  
  ;1201;;pieces;;789514145;MALE,FEMALE;  
  ;1202;;pieces;;689514145;MALE,FEMALE;  
  ;1203;;pieces;;289514145;MALE,FEMALE;
* **products\_en.impex**
* **$productCatalog**=apparelProductCatalog  
  **$productCatalogName**=Apparel Product Catalog  
    
  **$catalogVersion**=catalogversion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$productCatalog:Staged]  
    
  # Language  
  **$lang**=en  
    
  # Update allProducts with localisations  
  UPDATE Product;code[unique=true];$catalogVersion;name[lang=$lang];summary[lang=$lang];description[lang=$lang]  
  ;1200;;White T-shirt;Cotton new white t-shirt which everyone loves!; White T-shirt for all genders which is 100% cotton;  
  ;1201;;Banana Socks;Pink Coloured Cotton socks with Banana Prints;100% cotton Pink Coloured Cotton socks with Banana Prints;  
  ;1202;;Warm Sweater;Brown Coloured Sweater made from 100% wool;100% wool Brown Coloured Sweater to save you from the chilly winds;  
  ;1203;;Round Glasses;Stylish round glasses;Stylish round glasses to give you a cool look instantly!;
* **products-media.impex : Created folders for different image formats**
* **$productCatalog**=apparelProductCatalog  
  **$productCatalogName**=Apparel Product Catalog  
    
  **$catalogVersion**=catalogversion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$productCatalog]  
  **$media**=*@media*[translator=de.hybris.platform.impex.jalo.media.MediaDataTranslator]  
  **$thumbnail**=thumbnail(code, $catalogVersion)  
  **$picture**=picture(code, $catalogVersion)  
  **$thumbnails**=thumbnails(code, $catalogVersion)  
  **$detail**=detail(code, $catalogVersion)  
  **$normal**=normal(code, $catalogVersion)  
  **$others**=others(code, $catalogVersion)  
  **$data\_sheet**=data\_sheet(code, $catalogVersion)  
  **$medias**=medias(code, $catalogVersion)  
  **$galleryImages**=galleryImages(qualifier, $catalogVersion)  
  **$siteResource**=jar:de.hybris.NagAcc.initialdata.setup.InitialDataSystemSetup&/NagAccinitialdata/import/sampledata/productCatalogs/$productCatalog  
    
  # Create Media  
  INSERT\_UPDATE Media;mediaFormat(qualifier);code[unique=true];$media;mime[default='image/jpeg'];$catalogVersion;folder(qualifier)  
  ;1200Wx1200H;/1200Wx1200H/tshirt1.jpg;$siteResource/images/1200Wx1200H/tshirt1.jpg;;;images;  
  ;515Wx515H;/515Wx515H/tshirt2.jpg;$siteResource/images/515Wx515H/tshirt2.jpg;;;images;  
  ;300Wx300H;/300Wx300H/tshirt3.jpg;$siteResource/images/300Wx300H/tshirt3.jpg;;;images;  
  ;96Wx96H;/96Wx96H/tshirt4.jpg;$siteResource/images/96Wx96H/tshirt4.jpg;;;images;  
  ;65Wx65H;/65Wx65H/tshirt4.jpg;$siteResource/images/65Wx65H/tshirt4.jpg;;;images;  
  ;30Wx30H;/30Wx30H/tshirt4.jpg;$siteResource/images/30Wx30H/tshirt4.jpg;;;images;  
  ;1200Wx1200H;/1200Wx1200H/socks1.jpg;$siteResource/images/1200Wx1200H/socks1.jpg;;;images;  
  ;515Wx515H;/515Wx515H/socks2.jpg;$siteResource/images/515Wx515H/socks2.jpg;;;images;  
  ;300Wx300H;/300Wx300H/socks3.jpg;$siteResource/images/300Wx300H/socks3.jpg;;;images;  
  ;96Wx96H;/96Wx96H/socks4.jpg;$siteResource/images/96Wx96H/socks4.jpg;;;images;  
  ;65Wx65H;/65Wx65H/socks4.jpg;$siteResource/images/65Wx65H/socks4.jpg;;;images;  
  ;30Wx30H;/30Wx30H/socks4.jpg;$siteResource/images/30Wx30H/socks4.jpg;;;images;  
  ;1200Wx1200H;/1200Wx1200H/sweater1.jpg;$siteResource/images/1200Wx1200H/sweater1.jpg;;;images;  
  ;515Wx515H;/515Wx515H/sweater2.jpg;$siteResource/images/515Wx515H/sweater2.jpg;;;images;  
  ;300Wx300H;/300Wx300H/sweater3.jpg;$siteResource/images/300Wx300H/sweater3.jpg;;;images;  
  ;96Wx96H;/96Wx96H/sweater4.jpg;$siteResource/images/96Wx96H/sweater4.jpg;;;images;  
  ;65Wx65H;/65Wx65H/sweater4.jpg;$siteResource/images/65Wx65H/sweater4.jpg;;;images;  
  ;30Wx30H;/30Wx30H/sweater4.jpg;$siteResource/images/30Wx30H/sweater4.jpg;;;images;  
  ;1200Wx1200H;/1200Wx1200H/glasses1.jpg;$siteResource/images/1200Wx1200H/glasses1.jpg;;;images;  
  ;515Wx515H;/515Wx515H/glasses2.jpg;$siteResource/images/515Wx515H/glasses2.jpg;;;images;  
  ;300Wx300H;/300Wx300H/glasses3.jpg;$siteResource/images/300Wx300H/glasses3.jpg;;;images;  
  ;96Wx96H;/96Wx96H/glasses4.jpg;$siteResource/images/96Wx96H/glasses4.jpg;;;images;  
  ;65Wx65H;/65Wx65H/glasses4.jpg;$siteResource/images/65Wx65H/glasses4.jpg;;;images;  
  ;30Wx30H;/30Wx30H/glasses4.jpg;$siteResource/images/30Wx30H/glasses4.jpg;;;images;  
    
    
  # Create Media Containers  
  INSERT\_UPDATE MediaContainer;qualifier[unique=true];$medias;$catalogVersion  
  ;media\_container\_1;/515Wx515H/tshirt2.jpg,/300Wx300H/tshirt3.jpg,/96Wx96H/tshirt4.jpg,/65Wx65H/tshirt4.jpg,/30Wx30H/tshirt4.jpg;;  
  ;media\_container\_2;/515Wx515H/socks2.jpg,/300Wx300H/socks3.jpg,/96Wx96H/socks4.jpg,/65Wx65H/socks4.jpg,/30Wx30H/socks4.jpg;;  
  ;media\_container\_3;/515Wx515H/sweater2.jpg,/300Wx300H/sweater3.jpg,/96Wx96H/sweater4.jpg,/65Wx65H/sweater4.jpg,/30Wx30H/sweater4.jpg;;  
  ;media\_container\_4;/515Wx515H/glasses2.jpg,/300Wx300H/glasses3.jpg,/96Wx96H/glasses4.jpg,/65Wx65H/glasses4.jpg,/30Wx30H/glasses4.jpg;;  
    
    
    
  # Update Products with Media and Media Containers  
  UPDATE Product;code[unique=true];$picture;$thumbnail;$detail;$others;$normal;$thumbnails;$galleryImages;$catalogVersion  
  ;1200;/300Wx300H/tshirt3.jpg;/96Wx96H/tshirt4.jpg;/1200Wx1200H/tshirt1.jpg;/1200Wx1200H/tshirt1.jpg,/515Wx515H/tshirt2.jpg,/300Wx300H/tshirt3.jpg,/96Wx96H/tshirt4.jpg,/65Wx65H/tshirt4.jpg,/30Wx30H/tshirt4.jpg;/300Wx300H/tshirt3.jpg;/96Wx96H/tshirt4.jpg;media\_container\_1;  
  ;1201;/300Wx300H/socks3.jpg;/96Wx96H/socks4.jpg;/1200Wx1200H/socks1.jpg;/1200Wx1200H/socks1.jpg,/515Wx515H/socks2.jpg,/300Wx300H/socks3.jpg,/96Wx96H/socks4.jpg,/65Wx65H/socks4.jpg,/30Wx30H/socks4.jpg;/300Wx300H/socks3.jpg;/96Wx96H/socks4.jpg;media\_container\_2;  
  ;1202;/300Wx300H/sweater3.jpg;/96Wx96H/sweater4.jpg;/1200Wx1200H/sweater1.jpg;/1200Wx1200H/sweater1.jpg,/515Wx515H/sweater2.jpg,/300Wx300H/sweater3.jpg,/96Wx96H/sweater4.jpg,/65Wx65H/sweater4.jpg,/30Wx30H/sweater4.jpg;/300Wx300H/sweater3.jpg;/96Wx96H/sweater4.jpg;media\_container\_3;  
  ;1203;/300Wx300H/glasses3.jpg;/96Wx96H/glasses4.jpg;/1200Wx1200H/glasses1.jpg;/1200Wx1200H/glasses1.jpg,/515Wx515H/glasses2.jpg,/300Wx300H/glasses3.jpg,/96Wx96H/glasses4.jpg,/65Wx65H/glasses4.jpg,/30Wx30H/glasses4.jpg;/300Wx300H/glasses3.jpg;/96Wx96H/glasses4.jpg;media\_container\_4;
* **product-stocklevels.impex**
* **$productCatalog**=apparelProductCatalog  
  **$productCatalogName**=Apparel Product Catalog  
  **$catalogVersion**=catalogversion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$productCatalog:Staged]  
  **$vendor**=apparello  
    
    
  INSERT\_UPDATE StockLevel;productCode[unique=true];warehouse(code)[unique=true];available;inStockStatus(code);  
   ;1200;ap\_warehouse\_s;100;notSpecified;  
   ;1201;ap\_warehouse\_n;30;notSpecified;  
   ;1202;ap\_warehouse\_n;40;notSpecified;  
   ;1203;ap\_warehouse\_n;23;notSpecified;
* **products-prices.impex**
* **$productCatalog**=apparelProductCatalog  
    
  **$catalogVersion**=catalogversion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default='$productCatalog:Staged']  
  **$prices**=Europe1prices[translator=de.hybris.platform.europe1.jalo.impex.Europe1PricesTranslator]  
    
  # Set product approval status to Approved only for those products that have prices.  
  #$approved=approvalstatus(code)[default='approved']  
    
  INSERT\_UPDATE PriceRow;product(code,$catalogVersion)[unique=true];currency(isocode)[unique=true];price;unit(code);  
  ;1200;USD;20.0;pieces;  
  ;1200;GBP;25.0;pieces;  
  ;1201;USD;10.0;pieces;  
  ;1201;GBP;15.0;pieces;  
  ;1202;USD;15.0;pieces;  
  ;1202;GBP;18.0;pieces;  
  ;1203;USD;8.0;pieces;  
  ;1203;GBP;9.0;pieces;

1. **Add a new Item „Designer“ with attributes designer name (localized String), designer description (localized String), designer photo (MediaModel), number of orders**

**In NagAcccore-items.xml**

<typegroup name="Designers">  
 <itemtype code="Designer" autocreate="true" generate="true">  
 <description> Information of all Designers</description>  
 <deployment table="Designers" typecode="20000"/>  
 <attributes>  
  
 <attribute qualifier="uid" type="java.lang.String" autocreate="true" generate="true">  
 <modifiers read="true" write="true" unique="true"/>  
 <persistence type="property"/>  
 <description>Unique Id of the Designer</description>  
 </attribute>  
  
 <attribute qualifier="name" type="localized:java.lang.String" autocreate="true" generate="true">  
 <modifiers read="true" write="true"/>  
 <persistence type="property"/>  
 <description>Name of the Designer</description>  
 </attribute>  
  
 <attribute qualifier="description" type="localized:java.lang.String" autocreate="true" generate="true">  
 <modifiers read="true" write="true"/>  
 <persistence type="property"/>  
 <description>Description of the Designer</description>  
 </attribute>  
  
 <attribute qualifier="numberOfOrders" type="java.lang.Integer" autocreate="true" generate="true">  
 <modifiers read="true" write="true"/>  
 <persistence type="property"/>  
 <description>Number of Orders received by the Designer</description>  
 </attribute>  
  
 <attribute qualifier="photo" type="Media" autocreate="true" generate="true">  
 <modifiers read="true" write="true"/>  
 <persistence type="property"/>  
 <description>Photo of the Designer</description>  
 </attribute>  
  
 </attributes>  
 </itemtype>  
  
</typegroup>

**Localizations for attributes and type in NagAcccore-locales\_en.properties : (Done only for en)**

type.Designer.name = Designer  
type.Designer.description = Information of all designers  
type.Designer.name.name = Name  
type.Designer.description.name = Description  
type.Designer.numberOfOrders.name = Number of Orders  
type.Designer.photo.name = Photo   
  
type.Designer.name.description = Name of the Designer  
type.Designer.description.description = Description of the Designer  
type.Designer.numberOfOrders.description = Number of Orders received by the Designer  
type.Designer.photo.description = Photo of the Designer

1. **Add a new attribute „designer“ of type Designer in ProductModel**

<typegroup name="Products">  
 <itemtype code="Product" autocreate="false" generate="false">  
 <attributes>  
 <attribute qualifier="designer" type="Designer" generate="true" autocreate="true">  
 <description>Designer associated with the Product</description>  
 <modifiers read="true" write="true"/>  
 <persistence type="property"/>  
 </attribute>  
 </attributes>  
 </itemtype>  
  
</typegroup>

**Localizations:**

* type.Product.designer.name = Designer of the Product  
  type.Product.designer.description = Designer associated with the product

1. **Add a new attribute „Designer Preference“ of type Designer in CustomerModel**

<typegroup name="Customers">  
 <itemtype code="Customer" autocreate="false" generate="false">  
 <attributes>  
 <attribute qualifier="designerPreference" type="Designer" autocreate="true" generate="true">  
 <modifiers read="true" write="true"/>  
 <description>Preferred Designer of the Customer</description>  
 <persistence type="property"/>  
 </attribute>  
 </attributes>  
 </itemtype>  
</typegroup>

**Localizations:**

* type.Customer.designerPreference.name = Preferred Designer   
  type.Customer.designerPreference.description = Designer preferred by the Customer

1. **Create an interceptor which checks if the media of designer is empty at the time of save and throws error if so**

**NagAcccore-spring.xml**

<!-- Bean registration for DesignerMediaValidateInterceptor -->  
<bean id="designerMediaValidateInterceptor" class="de.hybris.NagAcc.core.interceptors.impl.DesignerMediaValidateInterceptor"/>  
  
<!--Map the interceptor with the corresponding Model -->  
<bean id="designerMediaValidateInterceptorMapping" class="de.hybris.platform.servicelayer.interceptor.impl.InterceptorMapping">  
 <property name="interceptor" ref="designerMediaValidateInterceptor"/>  
 <property name="typeCode" value="Designer"/>  
</bean>

package de.hybris.NagAcc.core.interceptors.impl;  
  
import de.hybris.NagAcc.core.model.DesignerModel;  
import de.hybris.platform.servicelayer.interceptor.InterceptorContext;  
import de.hybris.platform.servicelayer.interceptor.InterceptorException;  
import de.hybris.platform.servicelayer.interceptor.ValidateInterceptor;  
  
public class DesignerMediaValidateInterceptor implements ValidateInterceptor {  
  
 @Override  
 public void onValidate(Object o, InterceptorContext interceptorContext) throws InterceptorException {  
 if (o instanceof DesignerModel)  
 {  
 DesignerModel designerModel = (DesignerModel) o;  
 if(designerModel.getPhoto() == null)  
 {  
 throw new InterceptorException("Designer with name " + designerModel.getName() + " has no photo associated with their profile.");  
 }  
 }  
 }  
}

1. **Create a new component OurDesigners which shows 4 banners of any 4 designers**

* **First created 4 designers using impex in designer.impex,designer\_en.impex,designer-media.impex (sampledata -> productCatalog)**
* **Designer-media.impex**

$lang=en

$contentCatalog=apparel-ukContentCatalog

$productCatalog=apparelProductCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged]

$siteResource=jar:de.hybris.NagAcc.initialdata.setup.InitialDataSystemSetup&/NagAccinitialdata/import/sampledata/productCatalogs/$productCatalog

INSERT\_UPDATE Media;$contentCV[unique=true];code[unique=true];realfilename;@media [translator=de.hybris.platform.impex.jalo.media.MediaDataTranslator]; mime[default='image/jpeg'];folder(qualifier)[default='images']

;;designer3.jpg;designer3.jpg;$siteResource/images/designers/designer3.jpg;;;

;;designer2.jpg;designer2.jpg;$siteResource/images/designers/designer2.jpg;;;

;;designer1.jpg;designer1.jpg;$siteResource/images/designers/designer1.jpg;;;

;;designer4.jpg;designer4.jpg;$siteResource/images/designers/designer4.jpg;;;

* **Designer.impex**

INSERT\_UPDATE Designer;uid[unique=true]; numberOfOrders;$photo;

;D100;3;designer1.jpg;

;D200;5;designer2.jpg;

;D300;10;designer3.jpg;

;D400;4;designer4.jpg

* **Designer\_en.impex**

$lang=en

UPDATE Designer;uid[unique=true];name[lang=$lang];description[lang=$lang];

;D100;Sam Jackson;A well known fashion designer, particularly known for designing jackets;

;D200;Lisa Jewel;A well known fashion designer, particularly known for designing glasses;

;D300;Reema Singh;A well known fashion designer, particularly known for designing blazers;10;

;D400;Jan Kapoor;A well known fashion designer, particularly known for designing bags;10;

* **Products\_designer.impex (Not needed for this question)**

$productCatalog=apparelProductCatalog

$productCatalogName=Apparel Product Catalog

$catalogVersion=catalogversion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$productCatalog:Staged]

UPDATE Product;code[unique=true];$catalogVersion;designer(uid);

;1200;;D100;

;1201;;D200;

;1202;;D300**;**

**;1203;;D400;**

* **Included these files in InitialDataSystemSetup.java**
* **Create 4 banner components**

**Cms-responsive-content.impex**

$contentCatalog=apparel-ukContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged]

$picture = media(code,$contentCV)

INSERT\_UPDATE BannerComponent;uid[unique=true];$contentCV[unique=true];

;ourDesigner1;

;ourDesigner2;

;ourDesigner3;

;ourDesigner4;

INSERT\_UPDATE RotatingImagesComponent;$contentCV[unique=true];uid[unique=true];banners(uid,$contentCV);&componentRef

;;ourDesigners;ourDesigner1,ourDesigner2,ourDesigner3,ourDesigner4;ourDesigners;

**(cms-responsive-content\_en.impex)**

$contentCatalog=apparel-ukContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged]

$picture = media(code,$contentCV)

UPDATE BannerComponent;uid[unique=true];$contentCV[unique=true];$picture[lang=en];headline[lang=en];content[lang=en];

"#% beforeEach:

de.hybris.NagAcc.core.service.DesignerService designerService = de.hybris.platform.core.Registry.getApplicationContext().getBean(""designerService"");

try

{

de.hybris.NagAcc.core.model.DesignerModel designerModel = designerService.findDesignerByCode(""D100"");

line.put(new Integer(3), designerModel.getPhoto().getCode());

line.put(new Integer(4), designerModel.getName());

line.put(new Integer(5), designerModel.getDescription());

}

catch (Exception ex) {}

";

;ourDesigner1;

* Similarly, create ourDesigner2 and so on. Change the code
* **Add these banner components in a RotatingImagesComponent (ourDesigners)**

INSERT\_UPDATE RotatingImagesComponent;$contentCV[unique=true];uid[unique=true];banners(uid,$contentCV);&componentRef

;;ourDesigners;ourDesigner1,ourDesigner2,ourDesigner3,ourDesigner4;ourDesigners;

1. **On homepage, add a new slot and put an instance of the OurDesigners in that slot**

**Cms-content.impex**

$contentCatalog=apparel-ukContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged]

* INSERT\_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];

;;CS1;

Cms-content\_en.impex

* UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];name[lang=en];

;;CS1;Designer Content Slot

* INSERT\_UPDATE ContentSlotName;name[unique=true];template(uid,$contentCV)[unique=true][default='LandingPage2Template'];validComponentTypes(code);

;P1;;RotatingImagesComponent;

* INSERT\_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];active;cmsComponents(uid);

;;CS1;true;ourDesigners;

* INSERT\_UPDATE ContentSlotForPage;$contentCV[unique=true];uid[unique=true];position[unique=true];page(uid,$contentCV)[unique=true];contentSlot(uid,$contentCV)[unique=true];

;;P1-homepage;P1;homepage;CS1

* adding position P1 in landinglayout2.jsp

<cms:pageSlot position="P1" var="feature" element="div">

<cms:component component="${feature}"/>

</cms:pageSlot>

* Made changes in rotatingimagescomponent.jsp

1. **Show facet/filter on Category pages as well as search pages with Designer Names. Clicking on this filters all the products with that designer.**

**Solr.impex**

**$productCatalog**=apparelProductCatalog  
**$catalogVersions**=catalogVersions(catalog(id),version);  
**$facetSearchConfigName**=apparel-ukIndex  
**$facetSearchConfigDescription**=Apparel UK Solr Index  
**$searchIndexNamePrefix**=apparel-uk  
**$solrIndexedType**=apparel-ukProductType  
**$indexBaseSite**=apparel-uk  
**$indexLanguages**=en  
**$indexCurrencies**=GBP  
  
INSERT\_UPDATE SolrIndexedProperty;solrIndexedType(identifier)[unique=true];name[unique=true];type(code);sortableType(code);currency[default=false];localized[default=false];multiValue[default=false];facet[default=true];facetType(code);facetSort(code);priority;visible;useForSpellchecking[default=false];useForAutocomplete[default=false];fieldValueProvider;  
 ;$solrIndexedType;designerName;string;;;true;;true;Refine;Alpha;;true;;;designerNameValueProvider;

**Solr\_en.impex**

UPDATE SolrIndexedProperty;solrIndexedType(identifier)[unique=true];name[unique=true];displayName[lang=en];  
 ;$solrIndexedType;designerName;Designer Name;

**Class for the Value Provider**

public class DesignerNameValueProvider extends AbstractPropertyFieldValueProvider implements FieldValueProvider {  
  
 private FieldNameProvider fieldNameProvider;  
  
 public CommonI18NService getCommonI18NService() {  
 return commonI18NService;  
 }  
  
 public void setCommonI18NService(CommonI18NService commonI18NService) {  
 this.commonI18NService = commonI18NService;  
 }  
  
 private CommonI18NService commonI18NService;  
  
 public FieldNameProvider getFieldNameProvider() {  
 return fieldNameProvider;  
 }  
  
 public void setFieldNameProvider(FieldNameProvider fieldNameProvider) {  
 this.fieldNameProvider = fieldNameProvider;  
 }  
  
 @Override  
 public Collection<FieldValue> getFieldValues(IndexConfig indexConfig, IndexedProperty indexedProperty, Object o) throws FieldValueProviderException {  
 if (o instanceof ProductModel)  
 {  
 final ProductModel product = (ProductModel) o;  
 final Collection<FieldValue> fieldValues = new ArrayList<FieldValue>();  
 // case of the indexed property is localized  
 if (indexedProperty.isLocalized())  
 {  
 // retrieve and iterate over all the configured languages  
 final Collection<LanguageModel> languages = indexConfig.getLanguages();  
 for (final LanguageModel language : languages)  
 {  
 fieldValues.addAll(createFieldValue(product, language, indexedProperty));  
 }  
 }  
 // case of the indexed property is not localized  
 else  
 {  
 fieldValues.addAll(createFieldValue(product, null, indexedProperty));  
 }  
 return fieldValues;  
  
 }  
 else  
 {  
 throw new FieldValueProviderException("Cannot fetch designer name");  
 }  
 }  
  
 private List<FieldValue> createFieldValue(ProductModel product, LanguageModel language, IndexedProperty indexedProperty) {  
  
 final List<FieldValue> fieldValues = new ArrayList<FieldValue>();  
 Locale locale = null;  
 final String designerName;  
 if (product.getDesigner() != null) {  
 if (language != null)  
 {  
 locale = commonI18NService.getLocaleForLanguage(language);  
 }  
 // get names by language  
 i18nService.setLocalizationFallbackEnabled(false);  
 if (locale == null)  
 designerName = product.getDesigner().getName();  
 else  
 designerName = product.getDesigner().getName(locale);  
  
 i18nService.setLocalizationFallbackEnabled(true);  
 addFieldValues(fieldValues, indexedProperty, language, designerName);  
 }  
 return fieldValues;  
 }  
  
 private void addFieldValues(List<FieldValue> fieldValues, IndexedProperty indexedProperty, LanguageModel language, String designerName) {  
 // generate all Solr fields based on different qualifiers  
 final Collection<String> fieldNames = fieldNameProvider.getFieldNames(indexedProperty, language == null ? null : language.getIsocode());  
 for (final String fieldName : fieldNames)  
 {  
 fieldValues.add(new FieldValue(fieldName, designerName));  
 }  
 }

**Bean registration: core-spring.xml**

<!--Bean registration for Designer Name Value Provider -->  
<bean id="designerNameValueProvider"  
 class="de.hybris.NagAcc.core.search.solrfacetsearch.provider.impl.DesignerNameValueProvider"  
 parent="abstractPropertyFieldValueProvider">  
 <property name="fieldNameProvider" ref="solrFieldNameProvider"/>  
 <property name="commonI18NService" ref="commonI18NService"/>  
 </bean>

1. **Add a new field in customer registration form – My preferred designer – Which has list of all designer names. It should be mandatory field. Choosing a designer and registering should save that designer in customer model.**

* Created CustomRegisterForm which extends RegisterForm to include Designer Name
* public class CustomRegisterForm extends RegisterForm {  
   private String myPreferredDesigner;  
    
   public String getMyPreferredDesigner() {  
   return myPreferredDesigner;  
   }  
    
   public void setMyPreferredDesigner(String myPreferredDesigner) {  
   this.myPreferredDesigner = myPreferredDesigner;  
   }  
  }
* In RegisterData include myPreferredDesigner
* <bean class="de.hybris.platform.commercefacades.user.data.RegisterData">  
   <property name="myPreferredDesigner" type="java.lang.String"/>  
  </bean>
* **Create DesignerData**
* <bean class="de.hybris.NagAcc.facades.designer.data.DesignerData">  
   <property name="code" type="String"/>  
   <property name="name" type="String"/>  
  </bean>
* **Use CustomRegistrationForm as default form in LoginPageController. This method is defined in AbstractLoginPageController, so here Im overriding it.**
* // New method added  
  @Override  
  protected String getDefaultLoginPage(boolean loginError, HttpSession session, Model model) throws CMSItemNotFoundException {  
   final LoginForm loginForm = new LoginForm();  
   model.addAttribute(loginForm);  
   model.addAttribute("customRegisterForm", new CustomRegisterForm()); // changed  
   model.addAttribute(new GuestForm());  
    
   final String username = (String) session.getAttribute(*SPRING\_SECURITY\_LAST\_USERNAME*);  
   if (username != null)  
   {  
   session.removeAttribute(*SPRING\_SECURITY\_LAST\_USERNAME*);  
   }  
    
   loginForm.setJ\_username(username);  
   storeCmsPageInModel(model, getCmsPage());  
   setUpMetaDataForContentPage(model, (ContentPageModel) getCmsPage());  
   model.addAttribute(ThirdPartyConstants.SeoRobots.*META\_ROBOTS*, ThirdPartyConstants.SeoRobots.*INDEX\_NOFOLLOW*);  
    
   addRegistrationConsentDataToModel(model);  
    
   final Breadcrumb loginBreadcrumbEntry = new Breadcrumb("#",  
   getMessageSource().getMessage("header.link.login", null, "header.link.login", getI18nService().getCurrentLocale()),  
   null);  
   model.addAttribute("breadcrumbs", Collections.*singletonList*(loginBreadcrumbEntry));  
    
   if (loginError)  
   {  
   model.addAttribute("loginError", Boolean.*valueOf*(loginError));  
   GlobalMessages.*addErrorMessage*(model, "login.error.account.not.found.title");  
   }  
    
   return getView();  
  }
* **Add a method for fetching designers**
* // New  
  @ModelAttribute("designers")  
  public List<DesignerData> getDesigners()  
  {  
   return designerFacade.getDesigners();  
  }
* **Create DesignerFacade and DesignerService with their implementations and bean registrations**
* **Create populator and converter for DesignerModel and DesignerData and their beans registrations**
* **Create DesignerFacade and its implementation and register it**
* **Make JSP (register.tag) changes (in modelAttribute –‘customRegistrationForm’)**
* <form:form method="post" modelAttribute="customRegisterForm" action="${action}">  
   <formElement:formSelectBoxDefaultEnabled idKey="register.title"  
   labelKey="register.title" selectCSSClass="form-control"  
   path="titleCode" mandatory="true" skipBlank="false"  
   skipBlankMessageKey="form.select.none" items="${titles}" />  
    
   <formElement:formSelectBoxDefaultEnabled idKey="register.designerName"  
   labelKey="register.designerName" selectCSSClass="form-control"  
   path="myPreferredDesigner" mandatory="true" skipBlank="false"  
   skipBlankMessageKey="form.select.none" items="${designers}" />
* **base\_en**

**register.designerName = Your Preferred Designer**

* **Validation -> Extend RegisterValidator and override validate method and check if designerName is empty**
* @Component("customRegistrationValidator")  
  public class CustomRegistrationValidator extends RegistrationValidator {  
    
   @Override  
   public void validate(Object object, Errors errors) {  
   super.validate(object, errors);  
   CustomRegisterForm customRegisterForm = (CustomRegisterForm) object;  
   validateDesignerName(errors, customRegisterForm.getMyPreferredDesigner());  
   }  
    
   private void validateDesignerName(Errors errors, String myPreferredDesigner) {  
   if(StringUtils.*isEmpty*(myPreferredDesigner))  
   {  
   errors.rejectValue("myPreferredDesigner", "register.designerName.invalid");  
   }  
   }  
  }
* **base\_en.properties**
* register.designerName.invalid = Please select a designer name
* **Use customRegistrationValidator in logincontroller**
* @RequestMapping(value = "/register", method = RequestMethod.*POST*)  
  public String doRegister(@RequestHeader(value = "referer", required = false) final String referer, final CustomRegisterForm form,  
   final BindingResult bindingResult, final Model model, final HttpServletRequest request,  
   final HttpServletResponse response, final RedirectAttributes redirectModel) throws CMSItemNotFoundException  
  {  
   //getRegistrationValidator().validate(form, bindingResult);  
   customRegistrationValidator.validate(form,bindingResult); //New  
   return processRegisterUserRequest(referer, form, bindingResult, model, request, response, redirectModel);  
  }
* **For storing the Designer with the Customer:**
* **Add getDesignerForCode(String code) in DesignerService, DesignerDao**
* **Extend CustomerFacade to include setDesignerNameForRegister() and setCommonPropertiesForRegister() (override)**
* In login page controller override processRegisterUserRequest and adddata.setMyPreferredDesigner(form.getMyPreferredDesigner());
* customerFacade.register(data);

**This will invoke register method of our DefaultCustomerFacade**

* **In this method setCommonProperties would be called which we have overridden to include setDesignerNameForRegister which would finally add the designer to the register.**

1. **On PDP, Cart Page show designer name against product and cart entry respectively.**

FOR PDP:

* First create Data class for Designer in facades-beans.xml (already done)
* <bean class="de.hybris.NagAcc.facades.designer.data.DesignerData">  
   <property name="code" type="String"/>  
   <property name="name" type="String"/>  
  </bean>
* In existing Data class of Product, add designer in facades-beans.xml
* <bean class="de.hybris.platform.commercefacades.product.data.ProductData">  
   <property name="genders" type="java.util.List&lt;de.hybris.NagAcc.facades.product.data.GenderData>"/>  
   <property name="designer" type="de.hybris.NagAcc.facades.designer.data.DesignerData"/>  
  </bean>
* Create populator and converter for Designer (done in q13)
* <!--Converter for DesignerModel -->  
  <alias name="defaultDesignerConverter" alias="designerConverter"/>  
  <bean id="defaultDesignerConverter" parent="abstractPopulatingConverter">  
   <property name="targetClass" value="de.hybris.NagAcc.facades.designer.data.DesignerData"/>  
   <property name="populators">  
   <list>  
   <ref bean="designerPopulator"/>  
   </list>  
   </property>  
  </bean>  
    
    
  <!-- Populator for DesignerModel to DesignerData -->  
  <alias name="defaultDesignerPopulator" alias="designerPopulator"/>  
  <bean id="defaultDesignerPopulator"  
   class="de.hybris.NagAcc.facades.populators.DesignerPopulator">  
  </bean>
* Create an enum DESIGNER
* <enum class="de.hybris.platform.commercefacades.product.ProductOption">  
   <value>DESIGNER</value>  
  </enum>
* Associate a populator to this enum (this populator would add Designer from ProductModel to ProductData… )
* <!-- Configured Populator for Product Option DESIGNER -->  
  <alias name="defaultCustomConfiguredPopulator" alias="productConfiguredPopulator"/>  
  <bean id="defaultCustomConfiguredPopulator" parent="defaultProductConfiguredPopulator">  
   <property name="populators">  
   <map key-type="de.hybris.platform.commercefacades.product.ProductOption" merge="true">  
   <entry key="DESIGNER" value-ref="productDesignerPopulator"/>  
   </map>  
   </property>  
  </bean>  
    
  <!-- Populator for populating product with its designer -->  
    
  <alias name="defaultProductDesignerPopulator" alias="productDesignerPopulator"/>  
  <bean id="defaultProductDesignerPopulator"  
   class="de.hybris.NagAcc.facades.product.converters.populator.ProductDesignerPopulator">  
   <property name="designerConverter" ref="designerConverter" />  
  </bean>
* ProductDesignerPopulator.java
* public class ProductDesignerPopulator<SOURCE extends ProductModel, TARGET extends ProductData> implements Populator<SOURCE,TARGET> {  
    
   private Converter<DesignerModel,DesignerData> designerConverter;  
    
   public Converter<DesignerModel, DesignerData> getDesignerConverter() {  
   return designerConverter;  
   }  
    
   public void setDesignerConverter(Converter<DesignerModel, DesignerData> designerConverter) {  
   this.designerConverter = designerConverter;  
   }  
    
   @Override  
   public void populate(SOURCE productModel, TARGET productData) throws ConversionException {  
   Assert.*notNull*(productData, "Parameter target cannot be null.");  
   if (productModel.getDesigner() != null)  
   {  
   DesignerData designerData = getDesignerConverter().convert(productModel.getDesigner());  
   productData.setDesigner(designerData);  
   }  
    
   }  
  }
* Add this option in ProductPageController
* options.add(ProductOption.*DESIGNER*);
* model.addAttribute("designer",productData.getDesigner());
* Include Designer Name in productdetailpanel.tag
* <div>  
   <c:if test="${not empty designer}">  
   <strong> Designed By : ${designer.name} </strong>  
   </c:if>  
   </div>

FOR CARTPAGE:

* orderEntryProductConverter : defined in commercefacades-spring.xml
* This converter contains a list of populators which convert ProductModel to ProductData
* This converter is being used in OrderEntryPopulator which is giving us OrderEntryData, which being used in AddToCartController as ‘’entry”
* We want to ensure that the product in this entry is being populated by its designer, so that designer can be shown on the cart page
* All population in ProductData for data to be displayed on the cart page is done by populators added in orderEntryProductConverter.
* So to add designer data, we add our existing productDesignerPopulator to orderEntryProductConverter, so that when the product in the order entry is being populated, the designer is also included.
* <!-- Adding populator to OrderEntryConverter which populates the product in the cart with its designer so that designer name can be shown on the cart page -->

<bean parent="modifyPopulatorList">

<property name="list" ref="orderEntryProductConverter"/>

<property name="add" ref="productDesignerPopulator"/>

</bean>

* cartItem.tag

<c:if test="${not empty entry.product.designer}">

<strong>Designed By:</strong> <div> ${entry.product.designer.name} </div>

</c:if>

1. **On PDP, show a sticker/tag/label – By your favorite designer if the customer is logged in and the product’s designer is same as customer’s preferred designer**

* Add DesignerData in CustomerData
* <bean class="de.hybris.platform.commercefacades.user.data.CustomerData">  
   <property name="designerPreference" type="de.hybris.NagAcc.facades.designer.data.DesignerData"/>  
  </bean>
* Create a populator that populates CustomerData with the DesignerData

Populator SOURCE -> UserModel

TARGET -> CustomerData

public class CustomerDesignerPopulator<SOURCE extends UserModel, TARGET extends CustomerData> implements Populator<SOURCE, TARGET> {  
  
 private Converter<DesignerModel, DesignerData> designerConverter;  
  
 public Converter<DesignerModel, DesignerData> getDesignerConverter() {  
 return designerConverter;  
 }  
  
 public void setDesignerConverter(Converter<DesignerModel, DesignerData> designerConverter) {  
 this.designerConverter = designerConverter;  
 }  
  
 @Override  
 public void populate(SOURCE source, TARGET target) throws ConversionException {  
 if (source instanceof CustomerModel)  
 {  
 CustomerModel customerModel = (CustomerModel) source;  
 if (customerModel.getDesignerPreference() != null)  
 {  
 target.setDesignerPreference(getDesignerConverter().convert(customerModel.getDesignerPreference()));  
 }  
 }  
  
 }  
}

* Define this populator in facades\_spring.xml
* <!-- Populator for populating CustomerData with DesignerPreference-->  
  <alias name="defaultCustomerDesignerPopulator" alias="customerDesignerPopulator"/>  
  <bean id="defaultCustomerDesignerPopulator" class="de.hybris.NagAcc.facades.user.converters.populator.CustomerDesignerPopulator">  
   <property name="designerConverter" ref="designerConverter"/>  
  </bean>
* Add this populator to the already existing customerConverter defined in commercefacades-spring.xml by using modifypopulatorlist
* <!-- Adding the above populator to the existing CustomerConverter defined in commercefacades-spring.xml -->  
  <bean parent="modifyPopulatorList">  
   <property name="list" ref="customerConverter"/>  
   <property name="add" ref="customerDesignerPopulator"/>  
  </bean>
* Created methods in DesignerFacade and DesignerService for implementing equality logic
* Then added following line in ProductPageController :

if (!getUserFacade().isAnonymousUser()) // if the customer is logged in  
{  
 CustomerData customerData = getCustomerFacade().getCurrentCustomer();  
 String firstName = customerData.getFirstName();  
 DesignerData designerData = customerData.getDesignerPreference();  
 model.addAttribute("fName",firstName);  
 if (designerData != null && productData.getDesigner() != null) {  
  
 if (designerFacade.isProductDesignerSameAsCustomerPreferredDesigner(productData,designerData))  
 model.addAttribute("preferredDesignerName", designerData.getName());  
  
 }  
  
  
}

1. **On Cart Page, add a slot which has a full page banner. Show this slot for the logged in customer if all the products in the cart have designer same as customer’s preferred designer.**

* OrderEntryData's ProductData is being populated by DesignerData because of (done for earlier question)
* <!--OrderEntry -> Product -> that product's data is being populated by designer data-->  
  <bean parent="modifyPopulatorList">  
   <property name="list" ref="orderEntryProductConverter"/>  
   <property name="add" ref="productDesignerPopulator"/>  
  </bean>
* Overriden cartfacade and defaultcartfacade to add a new method which would be called in cartpagecontroller
* public class DefaultCustomCartFacade extends DefaultCartFacade implements CustomCartFacade {  
   @Override  
   public boolean checkIfDesignerIsSameAsPreferredDesigner(List<OrderEntryData> orderEntryDataList, CustomerData customerData) {  
    
   String designerCode = customerData.getDesignerPreference().getCode();  
   boolean hasFavouriteDesigner = true;  
    
   for (OrderEntryData orderEntryData : orderEntryDataList) {  
   ProductData productData = orderEntryData.getProduct(); // this product data is being populated by the designer data because of the populator we added to the orderEntryProductConverter in facades\_spring.xml  
   if (productData.getDesigner() == null) // product does not have a designer  
   {  
   hasFavouriteDesigner = false;  
   break;  
   }  
    
   if(!designerCode.equals(productData.getDesigner().getCode())) {  
   hasFavouriteDesigner = false;  
   break;  
   }  
    
   }  
   return hasFavouriteDesigner;  
   }  
  }
* CartPageController
* // Shows Banner if all products in the cart have same designer as Customer's preferred  
  private void showDesignerIfCartContainsCustomerPreferredDesigner(Model model) {  
    
   if(!getUserFacade().isAnonymousUser()) { // if user is logged in  
   CartData cartData = getCartFacade().getSessionCart();  
   List<OrderEntryData> orderEntryDataList = cartData.getEntries();  
   CustomerData customerData = getCustomerFacade().getCurrentCustomer();  
    
   if (orderEntryDataList.isEmpty()) // Case when cart is empty  
   {  
   model.addAttribute("hasFavouriteDesigner", false);  
   return;  
   }  
    
   if (customerData.getDesignerPreference() != null) { // Customer has designer preference  
   if(((CustomCartFacade)getCartFacade()).checkIfDesignerIsSameAsPreferredDesigner(orderEntryDataList,customerData))  
   model.addAttribute("hasFavouriteDesigner", true);  
   else  
   model.addAttribute("hasFavouriteDesigner", false);  
   }  
   else  
   { // Customer does not have a designer preference  
   model.addAttribute("hasFavouriteDesigner", false);  
   }  
   }  
  }
* Create Banner

Create a BannerComponent

# Create Media for component

cartpage-banner-media.impex

$siteResource=jar:de.hybris.NagAcc.initialdata.setup.InitialDataSystemSetup&/NagAccinitialdata/import/coredata/contentCatalogs/apparelContentCatalog

$lang=en

$contentCatalog=apparel-ukContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged]

INSERT\_UPDATE Media;$contentCV[unique=true];code[unique=true];realfilename;@media [translator=de.hybris.platform.impex.jalo.media.MediaDataTranslator]; mime[default='image/jpeg'];folder(qualifier)[default='images']

;;sameDesignerBanner.jpg;sameDesignerBanner.jpg;$siteResource/images/banners/cartpage/sameDesignerBanner.jpg;;;

$picture = media(code,$contentCV)

#Create banner (cms-responsive-content.impex)

INSERT\_UPDATE BannerComponent;$contentCV[unique=true];uid[unique=true];&componentRef;urlLink;

;;sameDesignerBannerComp;sameDesignerBannerComp;/;

**Cms-res-content\_en.impex**

UPDATE BannerComponent;$contentCV[unique=true];uid[unique=true];name;&componentRef;$picture[lang=en];headline[lang=en];content[lang=en];

;;sameDesignerBannerComp;Same Designer Banner;sameDesignerBannerComp;sameDesignerBanner.jpg;All products from the same designer!;Your cart has all products from the same designer;

# Create ContentSlot

Cms-content.impex

INSERT\_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];active;cmsComponents(uid);  
;;CS2;Same Designer Banner Content Slot;true;sameDesignerBannerComp;

Cms-content\_en.impex

UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];name[lang=en];  
;;CS2;Same Designer Banner Content Slot;

# Create Position

INSERT\_UPDATE ContentSlotName;name[unique=true];template(uid,$contentCV)[unique=true][default='CartPageTemplate'];validComponentTypes(code);

;P2;;BannerComponent;

#Create ContentSlotForPage

INSERT\_UPDATE ContentSlotForPage;$contentCV[unique=true];uid[unique=true];position[unique=true];page(uid,$contentCV)[unique=true];contentSlot(uid,$contentCV)[unique=true];

;;P2-cartPage;P2;cartPage;CS2

* Add the position in cartPage.jsp if
* - cartPage.jsp
* <c:if test="${hasFavouriteDesigner}">
* <cms:pageSlot position="P2" var="feature">
* <cms:component component="${feature}" element="div"/>
* </cms:pageSlot>
* </c:if>

1. **On checkout page, before placing order – check if all the products are from one designer**

**– if yes, then set attribute „Designer“ with the designer for those products (Add a new attribute in OrderModel called „designer“ of Designer type)**

**- In SummaryCheckoutStepController, placeOrder would be called**

**- This calls the method in CheckoutFacade**

- Created a new attribute designer in OrderModel

- Create hasSameDesigner in CartModel

- in SummaryCheckoutController -> getCheckoutFacade.placeOrder() -> DefaultCheckoutFacade

- In DefaultCheckoutFacade, OrderData placeOrder (which is called above) uses placeOrder and beforePlaceOrder

- Overriding these two methods

- No interface (no new method)

- Need to override DefaultCheckoutFacade

- Since DefaultCheckoutFacade is being already extended By DefaultAcceleratorCheckoutFacade (which also has placeOrder method), we need to override this

DefaultAcceleratorCheckoutFacade. (Multiple inheritance not possible in Java)

package de.hybris.NagAcc.facades.checkout.impl;  
  
import de.hybris.NagAcc.core.model.DesignerModel;  
import de.hybris.platform.acceleratorfacades.order.impl.DefaultAcceleratorCheckoutFacade;  
import de.hybris.platform.commercefacades.order.impl.DefaultCheckoutFacade;  
import de.hybris.platform.commerceservices.enums.SalesApplication;  
import de.hybris.platform.commerceservices.enums.UiExperienceLevel;  
import de.hybris.platform.commerceservices.service.data.CommerceCheckoutParameter;  
import de.hybris.platform.core.model.order.AbstractOrderEntryModel;  
import de.hybris.platform.core.model.order.CartModel;  
import de.hybris.platform.core.model.order.OrderModel;  
import de.hybris.platform.core.model.product.ProductModel;  
import de.hybris.platform.order.InvalidCartException;  
  
import java.util.List;  
  
public class CustomAcceleratorCheckoutFacade extends DefaultAcceleratorCheckoutFacade {  
  
 @Override  
 protected void beforePlaceOrder(CartModel cartModel) {  
 if (cartModel != null) {  
 checkIfAllProductsInCartHaveSameDesigner(cartModel);  
 }  
 }  
  
 private void checkIfAllProductsInCartHaveSameDesigner(CartModel cartModel) {  
 boolean hasSameDesigner = true;  
 String designerCode = "";  
 List<AbstractOrderEntryModel> orderEntryModelList = cartModel.getEntries();  
 for (AbstractOrderEntryModel orderEntryModel : orderEntryModelList)  
 {  
 ProductModel productModel = orderEntryModel.getProduct();  
 if(productModel.getDesigner() == null){ // if a product has no designer set  
 hasSameDesigner = false;  
 break;  
 }  
 if(designerCode.isEmpty()) {  
 designerCode = productModel.getDesigner().getUid();  
 }  
  
 if(!designerCode.equals(productModel.getDesigner().getUid())) // different designers found  
 {  
 hasSameDesigner = false;  
 break;  
 }  
 }  
 cartModel.setHasSameDesigner(hasSameDesigner);  
 getModelService().save(cartModel);  
  
 }  
  
 @Override  
 protected OrderModel placeOrder(CartModel cartModel) throws InvalidCartException {  
  
 final UiExperienceLevel uiExperienceLevel = getUiExperienceService().getUiExperienceLevel();  
 final CommerceCheckoutParameter parameter = new CommerceCheckoutParameter();  
 parameter.setEnableHooks(true);  
 parameter.setCart(cartModel);  
  
 // checkIfAllProductsInCartHaveSameDesigner(cartModel);  
  
 OrderModel orderModel;  
 if (UiExperienceLevel.*MOBILE*.equals(uiExperienceLevel))  
 {  
 // Set application to WebMobile  
 parameter.setSalesApplication(SalesApplication.*WEBMOBILE*);  
 orderModel = getCommerceCheckoutService().placeOrder(parameter).getOrder();  
 }  
 else {  
 // Default to WEB  
 parameter.setSalesApplication(SalesApplication.*WEB*);  
 orderModel = getCommerceCheckoutService().placeOrder(parameter).getOrder();  
 }  
  
 if(cartModel.getHasSameDesigner())  
 {  
 DesignerModel designerModel = cartModel.getEntries().get(0).getProduct().getDesigner();  
 orderModel.setDesigner(designerModel);  
 getModelService().save(orderModel);  
 }  
 return orderModel;  
 }  
}

* <!-- Overriding the existing DefaultCheckoutFacade (Not created an interface) -->
* <alias name="defaultCustomAcceleratorCheckoutFacade" alias="acceleratorCheckoutFacade"/>
* <bean id="defaultCustomAcceleratorCheckoutFacade" class="de.hybris.NagAcc.facades.checkout.impl.CustomAcceleratorCheckoutFacade" parent="defaultAcceleratorCheckoutFacade">
* </bean>

1. **Once the order is placed, add a new action in order process which does following: a. If the order has designer set and the customer is a registered customer and if the customer’s preferred designer is different from the designer in order – sets the customer’s preferred designer as the order’s designer**

**- Made changes in order-process.xml**

<action id="checkOrder" bean="checkOrderAction">

<transition name="OK" to="checkDesigner"/> <!-- Changed to checkDesigner -->

<transition name="NOK" to="error"/>

</action>

<!-- Creating new action for changing customer's preferred designer if it is different from the order's designer -->

<action id="checkDesigner" bean="checkDesignerAction">

<transition name="OK" to="checkAuthorizeOrderPayment"/>

<transition name="NOK" to="checkAuthorizeOrderPayment"/>

</action>

* Created a bean for action in order-process-spring.xml

<!-- Action for setting Customer's preferred designer as the designer of the order, in case they are different -->

<bean id="checkDesignerAction" class="de.hybris.NagAcc.fulfilmentprocess.actions.order.CheckDesignerAction" parent="abstractAction">

</bean>

* public class CheckDesignerAction extends AbstractSimpleDecisionAction<OrderProcessModel> {
* @Override
* public Transition executeAction(OrderProcessModel orderProcessModel) throws RetryLaterException, Exception {
* UserModel userModel = orderProcessModel.getOrder().getUser();
* // String userUid = userModel.getUid();
* if (userModel instanceof CustomerModel)
* {
* CustomerModel customerModel = (CustomerModel) orderProcessModel.getOrder().getUser();
* boolean isGuest = customerModel.getType().equals(CustomerType.GUEST);
* if(!isGuest){
* final OrderModel orderModel = orderProcessModel.getOrder();
* DesignerModel orderDesigner = orderModel.getDesigner();
* if (orderDesigner != null) // if designer is set i.e when all products in the order have same designer
* {
* DesignerModel customerPreferredDesigner = customerModel.getDesignerPreference();
* if (!customerPreferredDesigner.getUid().equals(orderDesigner.getUid())) {
* customerModel.setDesignerPreference(orderDesigner);
* getModelService().save(customerModel);
* }
* }
* }
* return Transition.OK;
* }
* return Transition.NOK;
* }
* }

1. **Every day run a cronjob which fetches all orders (which have designer set) per designer and sets the „number of orders“ field of designer with that number**

- Bean for cronjob

<!-- Bean registration for UpdateNumberOfOrdersCronjob-->

<bean name="updateNumberOfDesignerOrdersJob" class="de.hybris.NagAcc.core.job.UpdateNumberOfOrdersOfDesignerJobPerformable" parent="abstractJobPerformable">

<property name="orderService" ref="orderService"/>

<property name="modelService" ref="modelService"/>

</bean>

-Class:

public class UpdateNumberOfOrdersOfDesignerJobPerformable extends AbstractJobPerformable<CronJobModel> {

private OrderService orderService;

private ModelService modelService;

private final static Logger LOG = Logger.getLogger(UpdateNumberOfOrdersOfDesignerJobPerformable.class.getName());

public ModelService getModelService() {

return modelService;

}

@Override

public void setModelService(ModelService modelService) {

this.modelService = modelService;

}

public OrderService getOrderService() {

return orderService;

}

public void setOrderService(OrderService orderService) {

this.orderService = orderService;

}

@Override

public PerformResult perform(CronJobModel cronJobModel) {

if(orderService instanceof CustomOrderService)

{

List<OrderModel> orders = ((CustomOrderService)getOrderService()).getAllOrders();

if(orders.size() > 0)

{

for(OrderModel order : orders)

{

if(order.getDesigner() != null)

{

DesignerModel designer = order.getDesigner();

Integer numberOfOrders = designer.getNumberOfOrders();

numberOfOrders++;

designer.setNumberOfOrders(numberOfOrders);

getModelService().save(designer);

}

}

LOG.info("Updated Number of Orders of Designers");

return new PerformResult(CronJobResult.SUCCESS, CronJobStatus.FINISHED);

}

LOG.info("Could not update number of Orders of Designers");

return new PerformResult(CronJobResult.FAILURE,CronJobStatus.ABORTED);

}

LOG.info("Could not update number of Orders of Designers");

return new PerformResult(CronJobResult.FAILURE,CronJobStatus.ABORTED);

}

}

- Overriden in OrderService & OrderDao

to include getNumberOfOrders() method

<!-- Bean registration for Overriding OrderDao-->

<alias name="defaultCustomOrderDao" alias="orderDao"/>

<bean name="defaultCustomOrderDao" class="de.hybris.NagAcc.core.dao.impl.DefaultCustomOrderDao" parent="defaultOrderDao"></bean>

<!-- Bean registration for Overriding OrderService-->

<alias name="defaultCustomOrderService" alias="orderService"/>

<bean name="defaultCustomOrderService" class="de.hybris.NagAcc.core.service.impl.DefaultCustomOrderService" parent="defaultOrderService">

<property name="submitOrderStrategies">

<list>

<!-- implementation that sends SubmitOrderEvent -->

<ref bean="eventPublishingSubmitOrderStrategy"/>

</list>

</property>

</bean>

- Impexes : (cronjob.impex, already being imported in essential data in CoreSystemSetup.java

INSERT\_UPDATE ServicelayerJob;code[unique=true];springId;

;updateNumberOfDesignerOrdersJob;updateNumberOfDesignerOrdersJob;

INSERT\_UPDATE CronJob;code[unique=true];job(code);sessionLanguage(isocode);

;updateNumberOfDesignerOrdersCronJob;updateNumberOfDesignerOrdersJob;en;

# Fires every day at 12pm

INSERT\_UPDATE Trigger;cronJob(code)[unique=true];active;cronExpression;

;updateNumberOfDesignerOrdersCronJob;true;0 0 12 1/1 \* ? \*;