Email WCMS and Process Engine Integration

- The hybris WCMS Module defines the emails to be generated by the system in the same way that the WCMS pages are defined.
- An email created by the system contains WCMS components that are rendered within the email text.
- The WCMS contents displaying on the emails, such as the company logo or banners, are managed through the WCMS slots, slot names, and components and can be configured using ImpEx or the WCMS Cockpit.
- The email text, such as subject line and body, are managed through Apache Velocity templates and can be configured using ImpEx or the hybris Management Console(hMC).

WCMS Integration

Email Page Template

- The **EmailPageTemplate** page is an extension of the**PageTemplate**page type. When the system generates an email, it is based on this page template.
- The email page template, like the WCMS page template, can have one or more**ContentSlotName** objects which define the names and positions of the**ContentSlot**objectsalong with WCMS components that are allowed in those slots.
- The email page template can have predefinedcontent slots with componentswhich are available across all email pages that are associated with that template.Only the **EmailPage** type can be associated with the email page template.

Email Page

- The**EmailPage**page is an extension of the**AbstractPage**page type, and represents formatted emails such as **Customer Registration**or**Order Confirmation.**
- The email page uses the **masterTemplate** template and defines **ContentSlotName** objects and any predefined **ContentSlots** objects with components that are specific to that particular email.
- There are also two localized attributes called **fromEmail** and **fromName** and both of these attributes are used for filling in corresponding email message properties during the sending of emails. The preview function can be disabled for the **EmailPage** page type, as shown in the **Turning Off Preview** example below.

Renderer Template

- The email subject, email body, and content of the WCMS components are generated using Velocity scripts. The**RendererTemplate** page contains a Velocity script, along with the **Content Class** name.
- The Velocity script contains tags which are replaced with the values from the context object provided during the rendering process.
- The Velocity script that renders the email subject line is usually the simplest, and may have very few Velocity tags.For example, an **Order Confirmation** email subject line may have a tag referring to the order number as the only component.
- The script that renders the email body is typically more complex and may have many Velocity tags, including tags referring to WCMS components. For example, an email body may have tags referring to the site logo, top and bottom banners, and so on. The script representing a WCMS component can have tags referring to the attributes of that WCMS component.
- The Velocity tags are in the form such as*${ctx.parameterName}* or*${ctx.CMSSlotContents.slotName}*where **ctx** refers to the email **Context Object** passed to the renderer. More details about the context objects are below. The**parameterName**parameter can be**firstName**,**lastName**and**orderNumber**, for example. Also, the**slotName**can be**siteLogo**,**topBanner**, and**bottomBanner**.

Available Emails Templates

The available **EmailPageTemplates** templates are triggered by defined events.

These templates include **Subject** and **Body** email pages that use **htmlTemplate** pages with the following naming schema:

- Subject - **`email-`***EmailPageTemplateName***`Subject.vm`**
- Layout and content - **`email-`***EmailPageTemplatName***`Body.vm`**

Both of the above templates are **RendererTemplate** page types, which utilize scripts for generating the subject and the body of an email.The available templates can be found in the following directory:

*${HYBRIS_BIN_DIR}***`/ext-template/yacceleratorcore/resources/yacceleratorcore/import/contentCatalogs/common/email`**

The table below lists the emails page template names and their descriptions for the defined processes:

Process	**Email Template**	**Description**
Order Status	• `email-notPickedUpConsignmentCanceledBody.vm` • `email-notPickedUpConsignmentCanceledSubject.vm`	Sent when a consignment has not been picked up.
	• `email-orderCancelledBody.vm` • `email-orderCancelledSubject.vm`	Sent when an entire order has been cancelled.
	• `email-orderCollectionReminderBody.vm` • `email-orderCollectionReminderSubject.vm`	Sent to remind the customer to pick up an order in a store location.
	• `email-orderConfirmationBody.vm` • `email-orderConfirmationSubject.vm`	Sent to confirm an order.
	• `email-orderMoveToCsBody.vm` • `email-orderMoveToCsSubject.vm`	Sent by a cron job when a consignment has a **Pickup In Store** status and the shipping date is older than the threshold. By default, it is set to 120 hours.
	• `email-orderPartiallyCanceledBody.vm`	Sent when a part of the order is

	• `email-orderPartiallyCanceledSubject.vm`	cancelled.
	• `email-orderPartiallyRefundedBody.vm` • `email-orderPartiallyRefundedSubject.vm`	Sent when a part of the order is refunded.
	• `email-orderRefundBody.vm` • `email-orderRefundSubject.vm`	Sent when the entire order is refunded.
	• `email-readyForPickupBody.vm` • `email-readyForPickupSubject.vm`	Sent when an order is ready to pick up at a store location.
	• `email-deliverySentBody.vm` • `email-deliverySentSubject.vm`	Sent when a delivery is sent out.
Customer Statement	• `email-customerRegistrationBody.vm` • `email-customerRegistrationSubject.vm`	Sent when a user registers in the online storefront.
	• `email-forgottenPasswordBody.vm` • `email-forgottenPasswordSubject.vm`	Sent when a user requests a password reset.

Adding an Email Template Structure Using ImpEx

You can configure the email page templates and the email pages in the WCMS Cockpit, or by using ImpEx. The configuration is very similar to the WCMS page configuration with the addition of providing renderer templates for the subject and body of the email.

The steps involved are:

1. Define the**EmailPageTemplate**which provides the subject, layout, and contents of the email body:

```
# Email page Template
```

```
INSERT_UPDATE EmailPageTemplate;
$contentCV[unique=true];uid[unique=true];name;active;frontendTemplateName;subject(code);htmlTemplate(code);r
estrictedPageTypes(code)
;;CustomerRegistrationEmailTemplate;Customer Registration Email
Template;true;customerRegistrationEmail;electronics_Email_Customer_Registration_Subject;electronics_Email_Cu
stomer_Registration_HTML;EmailPage
```

2.Add a Velocity template for the**EmailPageTemplate**which provides a layout for the email template in the WCMS Cockpit:

```
# Templates for WCMS Cockpit Page Edit
UPDATE EmailPageTemplate;
$contentCV[unique=true];uid[unique=true];velocityTemplate[translator=de.hybris.platform.yacceleratorcore.setup.
FileLoaderValueTranslator]
;;CustomerRegistrationEmailTemplate;$jarResource/cmscockpit/structure-
view/structure_customerRegistrationEmailTemplate.vm
```

3.Define**ContentSlotNames**along with allowed WCMS components for the**EmailPageTemplate**. Currently,**EmailGenerationService**supports**CMSParagraphComponent**,**CMSImageComponent**, **CMSLinkComponent**, and**CMSBannerComponent**:

```
INSERT_UPDATE ContentSlotName;name[unique=true];template(uid,$contentCV)[unique=true]
[default='CustomerRegistrationEmailTemplate'];validComponentTypes(code)
;SiteLogo;;CMSImageComponent,BannerComponent
;TopContent;;$wideContent;
;BottomContent;;$wideContent;
;Footer;;CMSLinkComponent,CMSParagraphComponent
```

4.Define **RendererTemplates** with Velocity template scripts for email subject and body:

```
# Email velocity templates
INSERT_UPDATE RendererTemplate;code[unique=true];contextClass;rendererType(code)[default='velocity']
;
electronics_Email_Customer_Registration_HTML;de.hybris.platform.commerceservices.process.email.context.Cust
omerEmailContext
;
electronics_Email_Customer_Registration_Subject;de.hybris.platform.commerceservices.process.email.context.Cust
omerEmailContext

# Email velocity templates
UPDATE
RendererTemplate;code[unique=true];description[lang=$lang];templateScript[lang=$lang,translator=de.hybris.platfo
rm.yacceleratorcore.setup.FileLoaderValueTranslator]
;electronics_Email_Customer_Registration_HTML;Customer Registration HTML Email;$emailResource/email-
customerRegistrationHTML_de.vm
;electronics_Email_Customer_Registration_Subject;Customer Registration Email Subject;$emailResource/email-
```

```
customerRegistrationSubject_de.vm
```

5.Add Velocity templates for the subject and body, as shown in the examples below:

Order confirmation email subject:

```
Order Confirmation ${ctx.order.code}
```

- Customer registration email body:

```
<html>
   <head>
   </head>
   <body bgcolor="#e4e7e8">
   <table width="100%" border="0" align="center" cellpadding="0" cellspacing="0"
bgcolor="#e4e7e8">
      <tr>
         <td>&nbsp;</td>
      </tr>
      <tr>
         <td align="center" valign="top">
            <table width="610" border="6" align="center" cellpadding="0" cellspacing="0"
bordercolor="#ebedee">
               <tr>
                  <td align="center" valign="top" bgcolor="#FFFFFF">
                     <table width="570" cellpadding="0" cellspacing="0" border="0"
align="center">
                        <tr>
                           <td valign="middle">
                              ${ctx.cmsSlotContents.SiteLogo}
                           </td>
                        </tr>
                        <tr>
                           <td height="30" align="right" valign="middle" bgcolor="#51585c">
                              <font color="#FFFFFF" size="2" face="Arial, Helvetica, sans-serif"><a
href="${ctx.secureBaseUrl}/my-account"><font color="#FFFFFF">My Account</font></a> | <a
href="${ctx.baseUrl}/store-finder"><font color="#FFFFFF">Store Finder</font></a>
&nbsp;&nbsp;</font>
                           </td>
                        </tr>
                        <tr>
                           <td align="center" valign="middle">
                              <a href="${ctx.baseUrl}" style="display:block; margin-top:10px;margin-
bottom:10px;">${ctx.cmsSlotContents.TopContent}</a>
                           </td>
                        </tr>
                        <tr>
                           <td align="left" valign="top">
                              <p><font color="#666666" size="2" face="Arial, Helvetica, sans-
```

```
serif"><b>Dear ${ctx.title} ${ctx.displayName}</b>,</font></p>
                              <p><font color="#666666" size="2" face="Arial, Helvetica, sans-
serif">Thank you for registering with hybris, we hope you&rsquo;ll enjoy shopping with
us.</font></p>
                              <p><font color="#666666" size="2" face="Arial, Helvetica, sans-
serif">Many Thanks </font></p>
                              <p><font color="#666666" size="2" face="Arial, Helvetica, sans-
serif">Customer Services</font></p>
                           </td>
                        </tr>
                        <tr>
                           <td align="center" valign="middle">
                              <a href="${ctx.baseUrl}" style="display:block; margin-top:10px;margin-
bottom:10px;">${ctx.cmsSlotContents.BottomContent}</a>
                           </td>
                        </tr>
                        <tr>
                           <td height="30" align="right" valign="middle" bgcolor="#51585c">
                              <font color="#FFFFFF" size="2" face="Arial, Helvetica, sans-serif"><a
href="${ctx.baseUrl}"><font color="#FFFFFF">Help</font></a> | <a
href="http://www.hybris.com/hybris/en/Contact.html"><font color="#FFFFFF">Contact
Us</font></a> | <a href="${ctx.baseUrl}"><font color="#FFFFFF">Terms &amp;
Conditions</font></a> &nbsp;&nbsp;</font>
                           </td>
                        </tr>
                     </table>
                  </td>
               </tr>
            </table>
         </td>
      </tr>
      <tr>
         <td>&nbsp;</td>
      </tr>
   </table>
</body>
</html>
```

6.Define the **EmailPage**:

```
# Customer Registration Email Page
INSERT_UPDATE EmailPage;
$contentCV[unique=true];uid[unique=true];name;masterTemplate(uid,
$contentCV);defaultPage;approvalStatus(code)
[default='approved'];fromEmail[lang=en];fromName[lang=en]
;;CustomerRegistrationEmail;Customer Registration
Email;CustomerRegistrationEmailTemplate;false;;from@website.com;From Website
```

7.Define WCMS components and content slots:

```
# WCMS Image Components
```

```
INSERT_UPDATE CMSImageComponent;
$contentCV[unique=true];uid[unique=true];name
;;EmailBannerSaleNowOnImage;Email Banner Sale Now On Image

# Content Slots
INSERT_UPDATE ContentSlot;
$contentCV[unique=true];uid[unique=true];name;active;cmsComponents
(uid,$contentCV)
;;EmailTopSlot;Default Email Top
Slot;true;EmailBannerSaleNowOnImage
```

8.Bind reusable content slots to the **EmailPageTemplate** using **ContentSlotForTemplate**:

```
# Bind Content Slots to Email Page Templates
INSERT_UPDATE ContentSlotForTemplate;
$contentCV;uid[unique=true];position[unique=true];pageTemplate(uid
,$contentCV)[unique=true]

[default='CustomerRegistrationEmailTemplate'];contentSlot(uid,
$contentCV)[unique=true];allowOverwrite
;;SiteLogo-CustomerRegistrationEmail;SiteLogo;;SiteLogoSlot;true
;;Footer-CustomerRegistrationEmail;Footer;;FooterSlot;true
;;TopContent-
CustomerRegistrationEmail;TopContent;;EmailTopSlot;true
;;BottomContent-
CustomerRegistrationEmail;BottomContent;;EmailBottomSlot;true
```

9.Bind any specific content slots to the**EmailPage**using**ContentSlotForPage**.

10.Add common WCMS component Velocity templates. For example, for the WCMS image component:

```
<img src="${ctx.parentContext.mediaBaseUrl}${ctx.media.url}"
alt="${ctx.media.altText}" border="0"/>
```

Turning off Preview

The following example shows how to disable the preview of an email page:

```
# Disable preview for email pages
UPDATE CMSPageType;code[unique=true];previewDisabled
;EmailPage;true
```

Email Processes

Sending emailis a business process containing a sequence of actions and is defined in a process definition XML file. The business process object contains the information required for the actions within the process.WithEmailprocesses, the following actions are configured:

1. **Generate Email Action -** The contents of the email are generated and an**EmailMessage**object is created. The**EmailMessage**objects created are attached to the business process so that the next action can act on them. This action depends on the**ContextResolutionStrategy**routineto impersonate the WCMS site, initialize the session context, and the**EmailGenerationService**routineto create the**EmailMessage**objects.
2. **Send Email -** All the emails attached to the business process are sent using the**EmailService** routine. If the emails are sent successfully, the**EmailMessage**object is updated with the sent details.
3. **Remove Sent Email Action -** If an**EmailMessage** objectwas sent successfully, it is removed from the system as illustrated in the code below:

```
<?xml version="1.0" encoding="utf-8"?>
<process xmlns="http://www.hybris.de/xsd/processdefinition"
start="generateCustomerRegistrationEmail" name="customerRegistrationEmailProcess"

processClass="de.hybris.platform.commerceservices.model.process.StoreFrontCustomerProcessM
odel" onError="error">

   <action id="generateCustomerRegistrationEmail" bean="generateCustomerRegistrationEmail">
      <transition name="OK" to="sendEmail"/>
      <transition name="NOK" to="error"/>
   </action>

   <action id="sendEmail" bean="sendEmail">
      <transition name="OK" to="removeSentEmail"/>
      <transition name="NOK" to="failed"/>
   </action>

   <action id="removeSentEmail" bean="removeSentEmail">
      <transition name="OK" to="success"/>
      <transition name="NOK" to="error"/>
   </action>

   <end id="error" state="ERROR">Something went wrong.</end>
   <end id="failed" state="FAILED">Could not send customer registration email.</end>
   <end id="success" state="SUCCEEDED">Sent customer registration email.</end>

</process>
```

Generating an Email

- The first step during this process is to impersonate the site and initialize the session context with the appropriate language and currency.
- This is accomplished by the ContextResolutionStategy routine. Then, an email context object containing all the required values to render a Velocity script is created.
- This is done by the EmailContextFactory routine. Finally, an EmailMessage object is created using the EmailGenerationService routine and is attached to the business process.

DefaultContextResolutionStrategy

The DefaultContextResolutionStrategy routine resolves the customer site to be used from the business process. For instance, If the business process is a StorefrontProcess, then the customer site is retrieved from the process itself. However, if the process is an OrderProcess, then the site is retrieved from the order that is associated with that process. This strategy also tries to resolve the language and the currency to be used based on the session language and currency of the customer associated with the business process, either directly or indirectly. If the system can not resolve the language and currency to be used, then it uses the session defaults.

DefaultEmailContextFactory

The **EmailContextFactory** routine is responsible for creating the email context object given the **EmailPage**, **RendererTemplate**, and email business process. It retrieves the name of the email context type from the renderer template, which also contains the Velocity script referencing the email context object associated with the email template.

There are routines such
as **CustomerEmailContext**, **ForgottenPasswordContext** and **OrderNotificationContext** in the system. All the email context objects for these routines contain general parameters such
as **baseUrl**, **secureBaseUrl**,
and **mediaBaseUrl.** The **CustomerEmailContext** also includes **CustomerData** objects such as first name, last name, and email address. The **ForgottenPasswordContext** also includes objects such as password token and password expiration time, along with **CustomerData** objects.
The **OrderNotificationContext** includes **OrderData** objects such as order code, delivery address, payment information, and order entries.

The email context object also includes the content of the WCMS components of all the WCMS slots configured for email and generated using the **RendererService** routine. The default factory routine determines the renderer template which contains the Velocity script. For a WCMS component, simple logic is used to find a template with an ID that is the same as the WCMS component class name, prefixed with the site UID and appended with the word **template**. Then it creates a context object required for that WCMS component and fills it with component attribute values. Finally, with the renderer service providing the script and the context object, the system generates the content of that WCMS component. These steps are repeated for all the WCMS components of a slot and the contents of the slots are included in the main context object and mapped as "**CMSContentSlots** dot **slotName**". The WCMS components content can therefore be accessed through their **contentSlotName** name**.** For example, to access the top banner content in the Velocity script, the velocity tag is *${ctx.CMSSlotContents.topContent}* with **topContent** being the **slotName**.

The context factory adds additional parameters to the email context object based on the **EmailContextVariables** routine, which provide a facility to define variables based on existing parameters in the email context with a simple syntax. For example, a variable defined becomes a parameter using **themeResourceUrl** with a value substituting *baseUrl* and *theme* parameter values:

DefaultEmailGenerationService

The **EmailGenerationService** routine determines the email subject and body renderer templates. Given email page templates, it generates the subject and body contents of the email using the renderer service, thus providing email context objects created by the factory described in the above section. It then creates an **EmailMessage** object.

Sending Emails

The **SendEmailAction** routine sends emails by using the **EmailService** routine. The action retrieves all the **EmailMessages** objects attached to the business process and tries to send them.

DefaultEmailService

The default email service uses Apache commons mail wrapped in **MailUtils** class to send **EmailMessages** objects and updates the object details of send status.

Deleting Emails

The **RemoveEmailAction** routine removes the emails from the system that are sent successfully. The action retrieves all the **EmailMessages** objects attached to the business process and tries to remove them from the system if they are sent successfully.

Starting the Email Process

Once an email process has been defined in an XML file and a resource bean has been created as shown below, then an email can be raised by starting the process.

```
<!-- Process resourcess definition -->
<bean id="customerRegistrationEmailProcessDefinitionResource"
```

```
class="de.hybris.platform.processengine.definition.ProcessDefiniti
onResource">
<property name="resource"
value="classpath:/yacceleratorcore/processes/customerRegistrationE
mailProcess.xml"/>
</bean>
```

You can start an email process as follows:

```
final StoreFrontCustomerProcessModel
storeFrontCustomerProcessModel = (StoreFrontCustomerProcessModel)
getBusinessProcessService()
.createProcess("customerRegistrationEmailProcess" +
System.currentTimeMillis(), "customerRegistrationEmailProcess");
storeFrontCustomerProcessModel.setCMSSite(registerEvent.getCMSSite
());
storeFrontCustomerProcessModel.setCustomer(registerEvent.getCustom
er());
getModelService().save(storeFrontCustomerProcessModel);
getBusinessProcessService().startProcess(storeFrontCustomerProcess
Model);
```

Task1: When Update Profile successfully Then Send mail User purpose we create mail template

Step1:-Creting <Project>core-Item.xml Model Class extends StoreFrontProcess Class Configure The Below In xml File

```
<typegroup name="process">
                  <itemtype code="UpdateProfileProcess" extends="StoreFrontProcess"
                        autocreate="true" generate="true"

      jaloclass="com.techouts.hybris.core.jalo.UpdateProfileProcess">
                        <attributes>
                              <attribute qualifier="currentDate" autocreate="true"
                                    type="java.util.Date">

                                    <modifiers read="true" write="true"
search="true"
                                          optional="true" />
                                    <persistence type="property" />
                              </attribute>
                              <attribute qualifier="customer" type="Customer">
                                    <persistence type="property" />
                                    <description>Attribute contains customer
object that will be used in the process.</description>
                              </attribute>
                        </attributes>
                  </itemtype>
            </typegroup>
```

Then ant build and Updatesystem running system

Step 2: Creating One Event Class Extending `AbstractCommerceUserEvent`

`in /lycamobilecore/src/com/lycamobile/core/event/`**UpdateProfileEvent**`.java`

```
/**
 *
 *
package com.lycamobile.core.event;

import de.hybris.platform.commerceservices.event.AbstractCommerceUserEvent;

import java.util.Date;


/**
 * @author jagadish
 *
 */

public class UpdateProfileEvent extends AbstractCommerceUserEvent
{
      private Date currentDate;

      public Date getCurrentDate()
      {
            return currentDate;
      }

      public void setCurrentDate(final Date currentDate)
      {
            this.currentDate = currentDate;
      }
}
```

Step 2 : Creating Listener Class extending AbstractSiteEventListener

`in /lycamobilecore/src/com/lycamobile/core/event/UpdateProfileEventListener.java`

```
/**
 *
 */
package com.lycamobile.core.event;


import de.hybris.platform.basecommerce.model.site.BaseSiteModel;
import de.hybris.platform.commerceservices.enums.SiteChannel;
import de.hybris.platform.commerceservices.event.AbstractSiteEventListener;
import de.hybris.platform.processengine.BusinessProcessService;
import de.hybris.platform.servicelayer.model.ModelService;
import de.hybris.platform.servicelayer.util.ServicesUtil;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Required;

import com.lycamobile.core.model.UpdateProfileProcessModel;


/**
 * @author jagadish
 *
 */
public class UpdateProfileEventListener extends
AbstractSiteEventListener<UpdateProfileEvent>
{
	private static final Logger LOG = Logger.getLogger(UpdateProfileEventListener.class);

	private ModelService modelService;
	private BusinessProcessService businessProcessService;

	@Override
	protected void onSiteEvent(final UpdateProfileEvent updateProfileEvent)
	{
		LOG.info("Now I am in UpdateProfileEventListener onSiteEvent () method");
		final UpdateProfileProcessModel updateProfileProcessModel =
(UpdateProfileProcessModel) getBusinessProcessService()
				.createProcess("customerProcess-" +
updateProfileEvent.getCustomer().getUid() + "-" + System.currentTimeMillis(),
						"updateProfileEmailProcess");
		updateProfileProcessModel.setSite(updateProfileEvent.getSite());
		updateProfileProcessModel.setCustomer(updateProfileEvent.getCustomer());
		updateProfileProcessModel.setStore(updateProfileEvent.getBaseStore());

	updateProfileProcessModel.setCurrentDate(updateProfileEvent.getCurrentDate());
		getModelService().save(updateProfileProcessModel);
		getBusinessProcessService().startProcess(updateProfileProcessModel);
		LOG.info("Now I am in UpdateProfileEventListener onSiteEvent () method
End");
	}


	/**
	 * @return the modelService
	 */
	protected ModelService getModelService()
	{
		return modelService;
	}

	/**
	 * @param modelService
	 *           the modelService to set
	 */
	@Required
	public void setModelService(final ModelService modelService)
	{
		this.modelService = modelService;
	}

	/**
	 * @return
	 */
	protected BusinessProcessService getBusinessProcessService()
	{
		return businessProcessService;
	}

	@Required
	public void setBusinessProcessService(final BusinessProcessService 
businessProcessService)
	{
		this.businessProcessService = businessProcessService;
	}


	/*
	 * (non-Javadoc)
	 *
	 * @see
	 *
de.hybris.platform.commerceservices.event.AbstractSiteEventListener#shouldHandleEvent(d
e.hybris.platform.servicelayer
	 * .event.events.AbstractEvent)
	 */
	@Override
	protected boolean shouldHandleEvent(final UpdateProfileEvent updateP)
	{
		LOG.info("Now I am in UpdateProfileEventListener shouldHandleEvent () 
method");
		final BaseSiteModel site = updateP.getSite();
		ServicesUtil.validateParameterNotNullStandardMessage("event.site", site);
		LOG.info("Now I am in UpdateProfileEventListener shouldHandleEvent () 
method End");
		return SiteChannel.B2C.equals(site.getChannel());

	}
}
```

Note :configure this listener in <projectname>core-spring.xml

```
<bean id="updateProfileEventListener"
		class="com.lycamobile.core.event.UpdateProfileEventListener"
		parent="abstractSiteEventListener">
```

```
            <property name="modelService" ref="modelService" />
            <property name="businessProcessService" ref="businessProcessService" />
      </bean>
```

Step3:-Creating Service Class And Façade Class For Service Class For Calling event and

InterFace Extending CustomerFacade
/lycamobilefacades/src/com/lycamobile/facades/customer/

```
/**
 *
 */
package com.lycamobile.facades.customer;

import de.hybris.platform.commercefacades.customer.CustomerFacade;
import de.hybris.platform.commercefacades.user.data.CustomerData;
import de.hybris.platform.commerceservices.customer.DuplicateUidException;

/**
 * @author jagadish
 *
 */
public interface lycamobileCustomerFacade extends CustomerFacade
{       @Override
        public void updateProfile(final CustomerData customerData) throws
DuplicateUidException;
}
```

Facade Implementation Class

Façade class for Calling Event

```
/**
 *
 */
package com.lycamobile.facades.customer;

import
de.hybris.platform.commercefacades.customer.impl.DefaultCustomerFacade;
import de.hybris.platform.commercefacades.user.data.CustomerData;
import
de.hybris.platform.commerceservices.customer.DuplicateUidException;
import de.hybris.platform.core.model.user.CustomerModel;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;

/**
 * @author jagadish
 *
 */
public class lycamobileCustomerFacadeImpl extends DefaultCustomerFacade
{
	private static final Logger LOG =
Logger.getLogger(lycamobileCustomerFacadeImpl.class);

	//
	@Autowired
	lycamobileCustomerAccountService customerAccountService;

	@SuppressWarnings("deprecation")
	@Override
	public void updateProfile(final CustomerData customerData) throws
DuplicateUidException
	{
		validateDataBeforeUpdate(customerData);

		final String name =
getCustomerNameStrategy().getName(customerData.getFirstName(),
customerData.getLastName());
		final CustomerModel customer = getCurrentSessionCustomer();
		customer.setOriginalUid(customerData.getDisplayUid());
		customerAccountService.updateProfile(customer,
customerData.getTitleCode(), name, customerData.getUid());
		LOG.info("Now we are callin Event  ");
		customerAccountService.updateProfileEvent(customer);
		LOG.info("Event End...");
	}
}
```

Step 3 :

a. **InterFace Extending CustomerAccountService**

```
/**
 *
 */
package com.lycamobile.facades.customer;

import de.hybris.platform.commerceservices.customer.CustomerAccountService;
import de.hybris.platform.commerceservices.customer.DuplicateUidException;
import de.hybris.platform.core.model.user.CustomerModel;


/**
 * @author jagadish
 *
 */
public interface lycamobileCustomerAccountService extends CustomerAccountService
{
	void updateProfileEvent(CustomerModel paramCustomerModel) throws
DuplicateUidException;

}
```

b. **Service Implementation Class**

```
/**
 *
 */
package com.lycamobile.facades.customer;

import de.hybris.platform.commerceservices.customer.DuplicateUidException;
import
de.hybris.platform.commerceservices.customer.impl.DefaultCustomerAccount
Service;
import de.hybris.platform.core.model.user.CustomerModel;

import org.apache.log4j.Logger;

import com.lycamobile.core.event.UpdateProfileEvent;


/**
 * @author jagadish
 *
 */
public class lycamobileCustomerAccountServiceImpl extends
DefaultCustomerAccountService implements
		lycamobileCustomerAccountService
{
	private static final Logger LOG =
Logger.getLogger(lycamobileCustomerAccountServiceImpl.class);

	/*
	 * (non-Javadoc)
	 *
	 * @see
	 *
com.lycmobile.facades.customer.lycamobileCustomerAccountService#update
ProfileEvent(de.hybris.platform.core.model
	 * .user.CustomerModel)
	 */
	@Override
	public void updateProfileEvent(final CustomerModel
paramCustomerModel) throws DuplicateUidException
	{

		LOG.info("I am calling EVENT");
		getEventService().publishEvent(initializeEvent(new
UpdateProfileEvent(), paramCustomerModel));
		// YTODO Auto-generated method stub
		LOG.info("EVENT Call FINESHED ");

	}

}
```

Configure This Classes In <Project>Façade-Spring.xml

```
<alias name="lycaMobileCustomerFacade"
alias="customerFacade"/>
    <bean id="lycaMobileCustomerFacade"
class="com.lycamobile.facades.customer.lycamobileCustomerFa
cadeImpl" parent="defaultCustomerFacade">
    <property name="customerAccountService"
ref="customerAccountService"></property>
    </bean>
    <alias alias="customerAccountService"
name="lycamobileCustomerAccountService"/>
    <bean id="lycamobileCustomerAccountService"
class="com.lycamobile.facades.customer.lycamobileCustomerAc
countServiceImpl" parent="defaultCustomerAccountService">
    </bean>
```

Step 4 :
Now We Create The Context Class This class object in Vm file.
/
lycamobilefacades/src/com/lycamobile/facades/process/email/context/UpdateProfileEmailConte
xt.java

```
//**
 *
 */
package com.lycamobile.facades.process.email.context;

import de.hybris.platform.acceleratorservices.model.cms2.pages.EmailPageModel;
import de.hybris.platform.acceleratorservices.process.email.context.AbstractEmailContext;
import de.hybris.platform.basecommerce.model.site.BaseSiteModel;
import de.hybris.platform.core.model.c2l.LanguageModel;
import de.hybris.platform.core.model.user.CustomerModel;

import com.lycamobile.core.model.UpdateProfileProcessModel;
```

```
/**
 * @author jagadish
 *
 */
public class UpdateProfileEmailContext extends
AbstractEmailContext<UpdateProfileProcessModel>
{
	public static final String UPDATION_DATE = "updationDate";



	@Override
	public void init(final UpdateProfileProcessModel updateProfileProcessModel, final
EmailPageModel emailPageModel)
	{
		super.init(updateProfileProcessModel, emailPageModel);
		put(UPDATION_DATE, updateProfileProcessModel.getCreationtime());
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
de.hybris.platform.acceleratorservices.process.email.context.AbstractEmailContext#getSite(
de.hybris.platform.
	 * processengine.model.BusinessProcessModel)
	 */
	@Override
	protected BaseSiteModel getSite(final UpdateProfileProcessModel
updateProfileProcessModel)
	{
		return updateProfileProcessModel.getSite();
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
	 *
de.hybris.platform.acceleratorservices.process.email.context.AbstractEmailContext#getCust
omer(de.hybris.platform
	 * .processengine.model.BusinessProcessModel)
	 */
	@Override
	protected CustomerModel getCustomer(final UpdateProfileProcessModel
updateProfileProcessModel)
	{
		return updateProfileProcessModel.getCustomer();
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see
	 *
de.hybris.platform.acceleratorservices.process.email.context.AbstractEmailContext#getEmai
lLanguage(de.hybris.platform
	 * .processengine.model.BusinessProcessModel)
	 */
```

```
        @Override
        protected LanguageModel getEmailLanguage(final UpdateProfileProcessModel
businessProcessModel)
        {
                // YTODO Auto-generated method stub
                return null;
        }

        public String getUpdationDate()
        {
                return get(UPDATION_DATE).toString();
        }

}
```

Configure this Class In <project>Façade-Spring.xml

```
<bean id="updateProfileEmailContext"
class="com.lycamobile.facades.process.email.context.Upd
ateProfileEmailContext" parent="abstractEmailContext"
scope="prototype" >  </bean>
```

Step 5:

Note:
1.Ant Build And
2.Upadte updatesystem Through Command Prompt
UpdateProfileProcess.java Automatically Generated after Ant Build

```
package com.techouts.hybris.core.jalo;

import de.hybris.platform.jalo.Item;
import de.hybris.platform.jalo.JaloBusinessException;
import de.hybris.platform.jalo.SessionContext;
import de.hybris.platform.jalo.type.ComposedType;
import org.apache.log4j.Logger;

public class UpdateProfileProcess extends GeneratedUpdateProfileProcess
{
        @SuppressWarnings("unused")
        private final static Logger LOG =
Logger.getLogger( UpdateProfileProcess.class.getName() );

        @Override
        protected Item createItem(final SessionContext ctx, final ComposedType type, final
ItemAttributeMap allAttributes) throws JaloBusinessException
        {
                // business code placed here will be executed before the item is created
                // then create the item
                final Item item = super.createItem( ctx, type, allAttributes );
                // business code placed here will be executed after the item was created
                // and return the item
                return item;
        }

}
```

Now We Create updateProcess.xml In Core
/lycamobilecore/resources/lycamobilecore/processes/updateProfileEmailProcess.xml

```
<?xml version="1.0" encoding="utf-8"?>
<process
xmlns="http://www.hybris.de/xsd/processdefinition"
start="generateUpdateProfileEmail"
name="updateProfileEmailProcess"


processClass="com.lycamobile.core.model.UpdateProfilePr
ocessModel" onError="error">

    <action id="generateUpdateProfileEmail"
bean="generateUpdateProfileEmail">
        <transition name="OK" to="sendEmail"/>
        <transition name="NOK" to="error"/>
    </action>

    <action id="sendEmail" bean="sendEmail">
        <transition name="OK" to="removeSentEmail"/>
        <transition name="NOK" to="failed"/>
    </action>

    <action id="removeSentEmail"
bean="removeSentEmail">
        <transition name="OK" to="success"/>
        <transition name="NOK" to="error"/>
    </action>

    <end id="error" state="ERROR">Something went
wrong.</end>
    <end id="failed" state="FAILED">Could not send
customer updateProfile info email.</end>
    <end id="success" state="SUCCEEDED">Sent customer
updateProfile info email.</end></process>
```

Note : Configure This xml File in <project>Core-Spring.xml

```
<bean id="updateProfileEmailProcessDefinitionResource"


class="de.hybris.platform.processengine.definition.Proc
essDefinitionResource">
        <property name="resource"


value="classpath:/lycamobilecore/processes/updateProfil
eEmailProcess.xml" />
```

```
     </bean>
```

And Here One Anothere Configuration also required Follow this

```
<bean id="generateUpdateProfileEmail"


class="de.hybris.platform.acceleratorservices.process.e
mail.actions.GenerateEmailAction"
         scope="tenant"
parent="abstractGenerateEmailAction">
         <property name="frontendTemplateName"
value="UpdateProfileEmailTemplate" />
     </bean>
```

Step 6: Now We Configure Server (Smpt) in local.properties in Config Folder

```
mail.smtp.server=smtp.googlemail.com
mail.smtp.port=587
mail.smtp.user=jagadish.k@techouts.com
mail.smtp.password=Jaggu@1216
mail.use.tls=true
mail.smtp.starttls.enable=true

#SMPP Configurations
text.smpp.enabled=true
```

We Mention <<project>Core>--→<ProjectCore.Messages We Write email-verifyEmail_en.properties File

```
## Please note that the syntax of the property name should be as follows:
## propertyName=some string
## Please DO NOT USE, property.name=some string
emailSubject=Verify Email
sitename = Lycamobile
myAccount=My Account
storeFinder=Store Finder
salutation=Dear {0} {1}
thankYouForRegistering=Thank you for registering with {0}Lycamobile, we hope
you&rsquo;ll enjoy shopping with us.
registeredEmailInfo=Your registered email address is
instructionsShoppingUrl=To start shopping please click here
instructionsAccountUrl=To access your account please click here: {0}My
Account{1}
instructionsForgottenPswdUrl=If you forget your password, you can reset it at
any time by clicking this link: {0}Update My Password{1}
instructionsContactUs=If we can help you with any enquiry, please contact our
customer services team directly via Phone 8010202222 or drop a Email at
support@v2kart.com
```

```
instructionsContactUs_electronics=If we can help you with any enquiry, please
use our {0}Contact Us{1} page, or contact our customer services team directly
via phone +44 (0)20 / 7429 4175.
help=Help
contactUs=Contact Us
contactUsPage=http://www.hybris.com/en/contact
contactUsEmailAddress=customerservices@hybris.com
termsAndCondition=Terms &amp; Conditions
happyShopping =Happy Shopping!
complimentaryClosing=Warm Regards
signature=jagadish LycaMobile.
sentenceUpdatePswdInstructions=To update your password please click here

paragraphSecurityNote=Please note: For security purposes this email will
expire within {0} minutes. Please update your  password within {1} minutes of
requesting the password update.
paragraphExpiredLink=If the link has already expired, please request an
update to your password again here: {0}Request a password update{1}
paragraphContactUs=If we can help you with any enquiry, please check our
{0}FAQ page{1}, use our {2}Contact Us{3} page, or contact our customer
services team directly via phone +44 (0)20 / 7429 4175 or email {4}.
paragraphContactUs_electronics=If we can help you with any enquiry, please
use our {0}Contact Us{1} page, or contact our customer services team directly
via phone +44 (0)20 / 7429 4175 or email {2}.
```

Step:7 Create Three Vm File One for Email Structure(Master Template)

a. For Master Template
b. Email Subject Purpose
c. Email Body Purpose
d. /lycamobilecore/resources/lycamobilecore/import/cockpits/cmscockpit/structure-view/

   structure_updateProfileEmailTemplate.vm

```
<div>
	<table width="100%" cellspacing="0" style="margin:0;padding:0;table-layout:fixed;border:1px
solid #1E4EBF;">
<tbody>
<tr>
<td colspan="2" height="125px" width="35%" rowspan="2" class="structureViewSection">
						<cockpit code="SiteLogo" /></td>
</tr>
<tr>
<td colspan="2" height="89px" class="structureViewSection">
	<div>Header</div>
	</td>
</tr>
<tr>
	<td height="170px" colspan="6" class="structureViewSection">
		<cockpit code="TopContent" />
</td>
</tr>
<tr>
<td height="300px" colspan="6" class="structureViewSection"></td>
</tr>
<tr><td height="170px" colspan="6" class="structureViewSection">
		<cockpit code="BottomContent" /></td>
</tr>
</tbody>
	</table>
	<div style="width:100%; border-top: 2px solid #bbb" class="cmsContentEditor">
		<cockpit code="editor" />
	</div>
</div>
```

structure_updateProfileEmailTemplate.vm

b. Email Subject Purpose

Email subject Purpose We Maintain This Subject VM File
email-updateProfileSubject.vm

```
${ctx.messages.emailSubject}
```

c.Email Body Purpose

email-updateProfileBody.vm

```
## messageSource=classpath:/merchandisecore/messages/email-
verifyEmail_$lang.properties
#macro(genHtmlLinkStartTag $url)
<a href="$url"><font color="#666666">
#end
#macro(genHtmlLinkEndTag)
</font></a>
#end
#macro(genHtmlLink $url $textColor $bodyContent)
<a href="$url"><font color="$textColor">$bodyContent</font></a>
#end

<html>
	<head>
	</head>
	<body bgcolor="#ffffff"
	<table width="100%" border="0" align="center" cellpadding="0"
```

```
cellspacing="0" bgcolor="#ffffff">
            <tr>
                  <td>&nbsp;</td>
            </tr>
            <tr>
                  <td align="center" valign="top">
                        <table width="610" border="0" align="center"
cellpadding="0" cellspacing="0" bordercolor="#fff">
                              <tr>
                                    <td align="center" valign="top"
bgcolor="#FFFFFF">
                                          <table width="570"
cellpadding="0" cellspacing="0" border="0" align="center">
                                                <tr>
                                                      <td
valign="middle">&nbsp;</td>
                                                </tr>
                                                <tr>
                                                      <td
valign="middle">
                                                            $
{ctx.cmsSlotContents.SiteLogo}
                                                            <img src="$
{ctx.themeResourceUrl}/images/header_01.png" alt="" width="229"
height="72" border="0" align="right" title="" />
                                                      </td>
                                                </tr>
                                                <tr>
                                                      <td height="30"
align="right" valign="middle" bgcolor="#000000">
                                                            <font
color="#FFFFFF" size="2" face="Arial, Helvetica, sans-serif"><a href="$
{ctx.secureBaseUrl}/my-account"><font color="#FFFFFF">$
{ctx.messages.myAccount}</font></a> | <a href="${ctx.baseUrl}/store-
finder"><font color="#FFFFFF">${ctx.messages.storeFinder}</font></a>
&nbsp;&nbsp;</font>
                                                      </td>
                                                </tr>
                                                <tr>
                                                      <td align="center"
valign="middle"><a href="${ctx.baseUrl}" style="display:block; margin-
top:10px;margin-bottom:10px;">$
{ctx.cmsSlotContents.TopContent}</a></td>
                                                </tr>
                                                <tr>
                                                      <td>&nbsp;</td>
                                                </tr>
                                                <tr>
                                                      <td align="left"
valign="top">
                                                      #set
($pswdRequestUrl = "${ctx.secureRequestResetPasswordUrl}")
                                                      #set ($mailToUrl =
"mailto:${ctx.messages.contactUsEmailAddress}")
                                                      #if($
{ctx.baseSite.Uid} == "electronics")
                                                      #set
( $paragraphContactUs = $
{ctx.messages.getMessage('paragraphContactUs_electronics',
"#genHtmlLinkStartTag(${ctx.messages.contactUsPage})",
"#genHtmlLinkEndTag()", "#genHtmlLink($mailToUrl '#666666' $
{ctx.messages.contactUsEmailAddress})")} )
                                                      #else
                                                      #set ($faqPage =
"${ctx.baseUrl}/faq")
                                                            #set
( $paragraphContactUs = ${ctx.messages.getMessage('paragraphContactUs',
"#genHtmlLinkStartTag($faqPage)", "#genHtmlLinkEndTag()",
"#genHtmlLinkStartTag(${ctx.messages.contactUsPage})",
"#genHtmlLinkEndTag()", "#genHtmlLink($mailToUrl '#666666' $
{ctx.messages.contactUsEmailAddress})")} )
                                                            #end
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif"><b>$
{ctx.messages.getMessage('salutation', ${ctx.title},$
{ctx.displayName})}</b>,</font></p>
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif">$
{ctx.messages.sentenceAlmostDone}&nbsp;&nbsp;</font><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif">$
{ctx.messages.sentenceUpdatePswdInstructions}:&nbsp;<a href="$
{ctx.secureResetPasswordUrl}"><font color="#666666">$
{ctx.displaySecureResetPasswordUrl}</font></a> </font></p>
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif">$
{ctx.messages.getMessage('paragraphSecurityNote', $
{ctx.expiresInMinutes}, ${ctx.expiresInMinutes})}</font></p>
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif">$
{ctx.messages.getMessage('paragraphExpiredLink',"#genHtmlLinkStartTag($
pswdRequestUrl)", "#genHtmlLinkEndTag()")}</font></p>
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-
serif">$paragraphContactUs</font></p>
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif">$
{ctx.messages.complimentaryClosing}</font></p>
                                                                  <p><font
color="#666666" size="2" face="Arial, Helvetica, sans-serif">$
{ctx.messages.signature}</font></p>
                                                            </td>
                                                      </tr>
                                                      <tr>
                                                            <td>&nbsp;</td>
                                                      </tr>
                                                      <tr>
                                                            <td align="center"
valign="middle">
                                                                  <a href="$
{ctx.baseUrl}" style="display:block; margin-top:10px;margin-
bottom:10px;">${ctx.cmsSlotContents.BottomContent}</a>
                                                            </td>
                                                      </tr>
                                                      <tr>
                                                            <td height="30"
align="right" valign="middle" bgcolor="#000000">
                                                                  <font
color="#FFFFFF" size="2" face="Arial, Helvetica, sans-serif"><a href="$
{ctx.baseUrl}"><font color="#FFFFFF">${ctx.messages.help}</font></a> |
<a href="${ctx.messages.contactUsPage}"><font color="#FFFFFF">$
{ctx.messages.contactUs}</font></a> | <a href="${ctx.baseUrl}"><font
color="#FFFFFF">${ctx.messages.termsAndCondition}</font></a>
&nbsp;&nbsp;</font>
                                                            </td>
                                                      </tr>
                                                      <tr>
                                                            <td>&nbsp;</td>
                                                      </tr>
```

```
                                                </table>
                                          </td>
                                    </tr>
                              </table>
                        </td>
                  </tr>
                  <tr>
                        <td>&nbsp;</td>
                  </tr>
            </table>
</body>
</html>
```

Now We Create One Impex File And Mention the VM all files And in this Impex file Follow the Below ImpEx File And Observe We We Mention this VM Files

Step :2

```
# Import the CMS content for the Merchandise site
#
$prefix=b2ctelco
$contentCatalog=$prefixContentCatalog
$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCata
log]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged]
$wideContent=CMSImageComponent,BannerComponent
# Import modulegen config properties into impex macros
UPDATE
GenericItem[processor=de.hybris.platform.commerceservices.impex.impl.ConfigProp
ertyImportProcessor];pk[unique=true]
$jarResourceCms=$config-jarResourceCmsValue
$emailPackageName=$config-emailContextPackageName
$emailResource=$config-emailResourceValue



# Language
$lang=en,de,id

# Email page Template
INSERT_UPDATE EmailPageTemplate                    ;$contentCV[unique=true]
          ;uid[unique=true]                  ;name                              ;active
          ;frontendTemplateName          ;subject(code)
          ;htmlTemplate(code)                          ;restrictedPageTypes(code)
                                                                                    ;
          ;UpdateProfileEmailTemplate   ;Customer Update Profile Email Template
          ;true      ; updateProfileEmail    ;lycamobile_Email_Update_Profile_Subject
          ;lycamobile_Email_Update_Profile_Body        ;EmailPage



#Templates for CMS Cockpit Page Edit
UPDATE EmailPageTemplate ;$contentCV[unique=true]        ;uid[unique=true]
                    ;velocityTemplate[translator=de.hybris.platform.commerceservices.im
pex.impl.FileLoaderValueTranslator]
                                                                    ;
                              ;UpdateProfileEmailTemplate  ;$jarResourceCms/structure-
view/structure_updateProfileEmailTemplate.vm
```

```
INSERT_UPDATE ContentSlotName  ;name[unique=true]      ;template(uid,
$contentCV)[unique=true]
[default='UpdateProfileEmailTemplate'];validComponentTypes(code)
                                                        ;SiteLogo
        ;;;logo
                                                        ;TopContent
        ;;$wideContent;
                                                        ;BottomContent
        ;;$wideContent;


# Create Content Slots
INSERT_UPDATE ContentSlot;
$contentCV[unique=true];uid[unique=true];name;active
                                          ;;EmailSiteLogoSlot         ;Default
Email Site Slot;true
                                          ;;EmailTopSlot          ;Default Email
Top Slot;true
                                          ;;EmailBottomSlot       ;Default Email
Bottom Slot;true



# Bind Content Slots to Email Page Templates
INSERT_UPDATE ContentSlotForTemplate;
$contentCV[unique=true];uid[unique=true];position[unique=true];pageTemplate(uid,
$contentCV)[unique=true][default='UpdateProfileEmailTemplate'];contentSlot(uid,
$contentCV)[unique=true];allowOverwrite
                                          ;;SiteLogo-
UpdateProfileEmail;SiteLogo;;EmailSiteLogoSlot;true
                                          ;;TopContent-
UpdateProfileEmail;TopContent;;EmailTopSlot;true
                                          ;;BottomContent-
UpdateProfileEmail;BottomContent;;EmailBottomSlot;true


# Email Pages
INSERT_UPDATE EmailPage;$contentCV[unique=true];uid[unique=true]  ;name
;masterTemplate(uid,$contentCV);defaultPage;approvalStatus(code)
[default='approved']
                                          ;
;UpdateProfileEmail;Update Profile Email;UpdateProfileEmailTemplate
;true;




# Email velocity templates
INSERT_UPDATE RendererTemplate;code[unique=true]
                    ;contextClass
;rendererType(code)[default='velocity']
                 ;lycamobile_Email_Update_Profile_Subject
        ;com.lycamobile.facades.process.email.context.UpdateProfileEmailContext

;lycamobile_Email_Update_Profile_Body              ;com.lycamobile.facades.proce
ss.email.context.UpdateProfileEmailContext
```

```
# Email Pages
UPDATE EmailPage          ;$contentCV[unique=true];uid[unique=true]
;fromEmail[lang=$lang]     ;fromName[lang=$lang]
                                                ;
;UpdateProfileEmail               ;"shelly1hybris@gmail.com"  ;"Customer Services
Team"


 # Render template for update Profile Email Tempalte
UPDATE RendererTemplate    ;code[unique=true]
;description[lang=$lang]
;templateScript[lang=$lang,translator=de.hybris.platform.commerceservices.impex.i
mpl.FileLoaderValueTranslator]
                                               ;lycamobile_Email_Update_Profile_Su
bject     ;update profile Subject             ;$jarResourceCms/structure-view/email-
updateProfileSubject.vm
                                               ;lycamobile_Email_Update_Profile_Bo
dy            ;update profile Email Body     ;$jarResourceCms/structure-view/email-
updateProfileBody.vm
```

Finally : When we Upadte Profile Details Search The UpDate Controller Do This Modifications

```java
@RequestMapping(value = "/update-profile", method = RequestMethod.POST)
	@RequireHardLogIn
	public String updateProfile(final UpdateProfileForm updateProfileForm,
final BindingResult bindingResult, final Model model,
			final RedirectAttributes redirectAttributes) throws
CMSItemNotFoundException
	{
		getProfileValidator().validate(updateProfileForm, bindingResult);

		String returnAction =
ControllerConstants.Views.Pages.Account.AccountProfileEditPage;
		final CustomerData currentCustomerData =
customerFacade.getCurrentCustomer();
		final CustomerData customerData = new CustomerData();
		customerData.setTitleCode(updateProfileForm.getTitleCode());
		customerData.setFirstName(updateProfileForm.getFirstName());
		customerData.setLastName(updateProfileForm.getLastName());
		customerData.setUid(currentCustomerData.getUid());
		customerData.setDisplayUid(currentCustomerData.getDisplayUid());
		model.addAttribute("titleData", userFacade.getTitles());

		if (bindingResult.hasErrors())
		{
			GlobalMessages.addErrorMessage(model, "form.global.error");
		}
		else
		{
			try
			{

				/* Check Here For Event And Lsitener */

	customerFacade.updateProfile(customerData);
```

```
                         GlobalMessages.addFlashMessage(redirectAttributes,
GlobalMessages.CONF_MESSAGES_HOLDER,
                                     "text.account.profile.confirmationUpdat
ed", null);
                         returnAction = REDIRECT_TO_PROFILE_PAGE;
                   }
                   catch (final DuplicateUidException e)
                   {
                         bindingResult.rejectValue("email",
"registration.error.account.exists.title");
                         GlobalMessages.addErrorMessage(model,
"form.global.error");
                   }
             }

             storeCmsPageInModel(model,
getContentPageForLabelOrId(PROFILE_CMS_PAGE));
             setUpMetaDataForContentPage(model,
getContentPageForLabelOrId(PROFILE_CMS_PAGE));
             model.addAttribute("breadcrumbs",
accountBreadcrumbBuilder.getBreadcrumbs("text.account.profile"));
             return returnAction;
      }
```

Process Flow :

customerFacade.updateProfile(customerData); ---→ ShopperStopCustomerFacadeImpl Class in that →

```
public void updateProfile(final CustomerData customerData) throws
DuplicateUidException
      {
             validateDataBeforeUpdate(customerData);

             final String name =
getCustomerNameStrategy().getName(customerData.getFirstName(),
customerData.getLastName());
             final CustomerModel customer = getCurrentSessionCustomer();
             customer.setOriginalUid(customerData.getDisplayUid());
             customerAccountService.~~updateProfile~~(customer,
customerData.getTitleCode(), name, customerData.getUid());
             LOG.info("Now we are callin Event  ");


customerAccountService.updateProfileEvent(c
ustomer);
             LOG.info("Event End...");
      }
```

-->Facede To Service Class (ShopperStopCustomerFacadeImpl) in that Call

```
@Override
	public void updateProfileEvent(final CustomerModel paramCustomerModel) 
throws DuplicateUidException
	{
		LOG.info("I am calling EVENT");

getEventService().publishEvent(initializeEvent(new UpdateProfileEvent(), 
paramCustomerModel));
		// YTODO Auto-generated method stub
		LOG.info("EVENT Call FINESHED ");
	}
```

----->Then after listener generate and follow this process

```
	@Override
	protected void onSiteEvent(final UpdateProfileEvent updateProfileEvent)
	{
		LOG.info("Now I am in UpdateProfileEventListener onSiteEvent () 
method");
		final UpdateProfileProcessModel updateProfileProcessModel = 
(UpdateProfileProcessModel) getBusinessProcessService()
					.createProcess("customerProcess-" + 
updateProfileEvent.getCustomer().getUid() + "-" + System.currentTimeMillis(),
							"updateProfileEmailProce
ss");
		updateProfileProcessModel.setSite(updateProfileEvent.getSite());
	
	updateProfileProcessModel.setCustomer(updateProfileEvent.getCustomer());
	
	updateProfileProcessModel.setStore(updateProfileEvent.getBaseStore());
		

updateProfileProcessModel.setCurrentDate(updateProfileEvent.getCurrentDate());
		getModelService().save(updateProfileProcessModel);
	
	getBusinessProcessService().startProcess(updateProfileProcessModel);
		LOG.info("Now I am in UpdateProfileEventListener onSiteEvent () 
method End");
	}
```

Then Listener class this method generate

```
@Override
	protected boolean shouldHandleEvent(final UpdateProfileEvent updateP)
	{
		LOG.info("Now I am in UpdateProfileEventListener 
shouldHandleEvent () method");
		final BaseSiteModel site = updateP.getSite();
	
	ServicesUtil.validateParameterNotNullStandardMessage("event.site", 
site);
```

```
            LOG.info("Now I am in UpdateProfileEventListener
shouldHandleEvent () method End");
            return SiteChannel.B2C.equals(site.getChannel());

      }
```

Then Context object generate and mail send

Note:- if you get exception while sending the Email please check Access for less secure apps turnon or off.if it is off make it as a turnon .if u use this link only You are login in Gmail

https://support.google.com/accounts/answer/6010255?hl=en