Technical Design Document
Hybris Out-Of-Box – B2C Storefront

Home Page and Navigation

Creation Date: 6/27/17
Last Update: 6/27/17
Version 1.0

Sapient®

Contents

1 About this Document

This document describes the design for Homepage & Navigation for Hybris Out of box apparel-uk store front.

1.1 Intended Audience

- Hybris architects
- Business Analyst
- Track Leads and developers.

1.2 Revision History

Date	Version	Description	Author
June 21, 2017	1.0	Created Initial Draft	Warsha Jaiswal

1.3 Related Documents

Document Title	Version	Link	Comments

1.4 Glossary

Acronym	Description
OOB	Out of Box
PDP	Product Detail Page
PLP	Product Listing Page

2 Approvals / Sign off

2.1 Final Signature

With the formal sign-off <<TBD>> key stakeholder verify this document to be in accordance with their expectation:

Name	Organization	Role	Date	Signature

3 Overview

Hybris comes with OOB extension **yacceleratorstorefront,** this extension is a template for a ready-to-be-adapted web frontend. It uses the Spring MVC.

The **yacceleratorstorefront** template extension unites all the functions typically found in an online shop front-end and interacts with the **Backoffice, Administration Cockpit, Product, and CMS Cockpits** using the essential data from the yacceleratorcore extension and, optionally, sample store content from the apparelstore or the electronicsstore extension. It supports multiple store fronts served from the same web application, as well as splits by country, language, and currency. Each storefront has independently managed CMS content and product catalogs, both can be separated and shared between multiple storefronts.

This document details technical design based on **apparel–uk store**.

3.1 Apparel Storefront Setup Overview

The **yacceleratorstorefront** extension is intended as a starting point to accelerate project. The **yacceleratorstorefront** extension uses the **commercefacades** extension to provide much of the storefront functionality exposed on the front-end. This makes it much easier to share business logic and storefront functionality across multiple channels.

The **apparelstore** extension implements the **AbstractSystemSetup** class in the **ApparelStoreSystemSetup**. When initialization is triggered, the **createProjectData** method will be called. This method will then call the **CoreDataImportService** and **SampleDataImportService** (in the **yacceleratorinitialdata** extension) to trigger the import of the different ImpEx files to hookup apparel storefront.

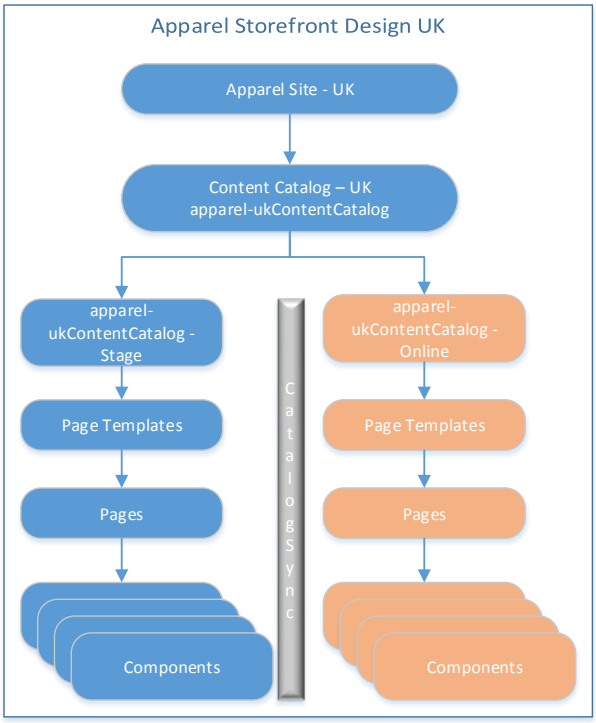
3.2 Document Overview

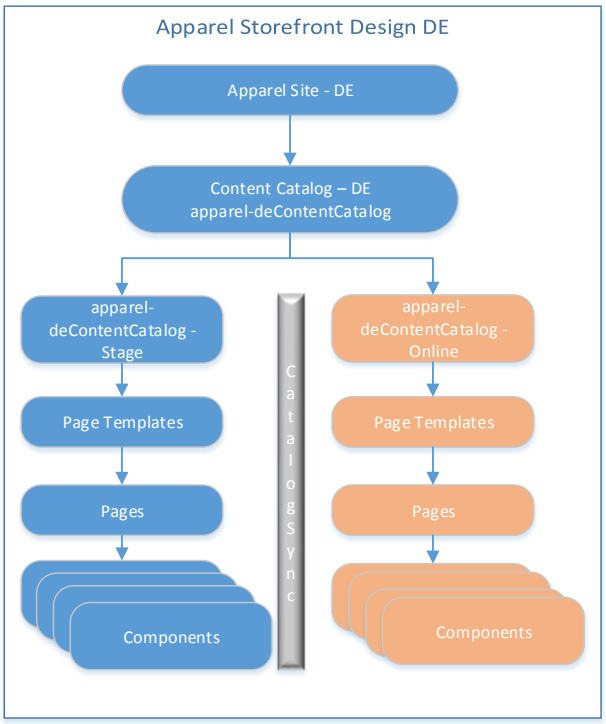
This document provides the detailed technical design to support homepage and navigation use into the following sections:

- Site Navigation
- Homepage
- Header
- Core Navigation
- Main Body
- Footer

4 Content Catalog Structure

Every storefront has independent CMS content and product catalog, below diagram explains the content catalog structure and hierarchy for apparel storefront.





Key points to note:

1. Each storefronts will have its own content catalog.
2. Each catalog will have two versions – stage and online.
3. Each catalog version will contain pre-defined have page templates. Each page template will correspond to a page type.
4. Each page will be composed of multiple WCMS components.
5. All changes will be made in the staged version of the content catalog. A sync process will move all approved objects from staged to online version of the respective catalog.
6. All sites will have its own catalog, templates, pages and components even if the look and feel are similar.

5 Storefront Web Application Structure

5.1 Web Application Configuration

All the files of a storefront definition are placed in the ***<$HYBRIS_BIN_DIR>*/ext-template/yacceleratorstorefront** directory. The structure of the directories and files is as follows:

- **web/src** Contains the Java source for the storefront. This includes page and Content Management System (CMS) controllers, interceptors, various custom implementations of classes required for Spring security, custom store front components, and so on.
- **web/webroot**
    - **_ui:** Contains the JavaScript and CSS styling for the current theme.
    - **shared/js:** Shared JavaScript used by desktop, mobile, and responsive pages.
    - **responsive/common:** Commonly used style sheets, JavaScript libraries and images.
    - **responsive/theme-black:** The black theme definition.
    - **responsive/theme-blue:** The blue theme definition.
- **WEB-INF**
    - **_ui-src:** Contains JS testing, full libraries, and the Less files used to generate the CSS for a theme.
    - **common/tld:** The tag library descriptor files for the CMS and ycommerce tags.
    - **config:** Spring application context files.
    - **lib:** The libraries required by the storefront.
    - **messages:** The localization files.
    - **tags:** The tags that are used within views.
    - **views:** The JSP pages, fragments and CMS components.

Web application structure contains several elements that define the request flow from the client to the server and back to the client. The most important elements are servlet and servlet mappings. In addition, any filter or listener can be introduced to the request flow.

5.2 Technical Design

As standard Web application, the **yacceleratorstorefront/web/webroot/WEB-INF/web.xml** file sets the default values for the various web applications deployed in an application server.

5.2.1 Filter Chains and Filter

Filter Chains:

The filters are in a chain and are called in the following order. The request is delivered to the servlet. The filter chain has three main entry points in the web.xml file:

S. No	Filter	Description
1	resourceFilter	The filter is used by server file resources by bypassing the other filters. It is only mapped to paths that contain files on the disk and should be served unaltered. This is used to serve the JavaScript, CSS, images, and theme files.

2	storefrontFilterChain	This is a reference to the chain of Spring-based filters. It provides set of filters used for Web application. See details of this filter below.
3	Spring Security Filter	This filter is used to enable Spring security support in the application. This filter is of type DelegatingFilterProxy, but the filter name is springSecurityFilterChain. The filter name has to match the bean ID defined by Spring security.

StorefrontFilterChain Filters:

S. No	Filter	Description
1	log4jFilter	Spring-based core platform filter, configures LOG4J.
2	dynamicTenantActivationFilter	Spring-based core Platform filter, deals with tenant switching while handling the HttpRequest.
3	sessionFilter	Spring-based core Platform filter, handles JALO session attachment and de-attachment from the HttpSession.
4	RequestLoggerFilter	The filter logs each HTTP request received by the web application. This is a useful filter during development, but is not recommended for use in production systems. It logs each request along with the time it took to process the request and generate the response.
5	characterEncodingFilter	The filter ensures proper encoding of the frontend page content.
6	CMSSiteFilter	This filter sets up the CMS integration for the application. It uses the requested URL to select the current site and sets up the session catalog versions. It also handles the preview data and redirecting to specific pages based on deep preview links.
7	StoreFrontFilter	This is the storefront application specific filter that initializes new sessions. It is responsible for calling the **StoreSessionFacade** to initialize each new session. The **StoreSessionFacade** sets up the initial language and currency for the session. This filter is also responsible for recording each request in the **BreadcrumbBuilder**, which is used when building a historical breadcrumb trail.
8	BTG Filters: • BTGSegmentFilter • AbstractBtgFilter • AbstractPkResolvingBtgFilter • BTGSegmentFilter • CategoryVisitedBtgFilter • ProductVisitedBtgFilter • RefererHeaderBtgFilter • RequestParamsBtgFilter	To enable Advanced Personalization, you need to find a place in your application where you can hook in with evaluation of customer segment rules. Each rule type provided by the **btgextension** requires a different data for evaluation. It is very important that an evaluation method is called whenever a request is made to the server. This is why the **yacceleratorstoretemplate** extension is equipped with the filter. It is responsible for rule computation that is triggered on each request. The following BTG filter beans are executed for each request: • refererHeaderBtgFilter • requestParamsBtgFilter • productVisitedBtgFilter • categoryVisitedBtgFilter

5.2.2 Listeners

There are servlet context listeners.

S. No	Listener	Description
1	HybrisContextLoaderListener	The SAP Hybris Commerce provides a **HybrisContextLoaderListener** for setting the global **ApplicationContext** automatically as a parent of your **WebApplicationContext**. You can use the global bean definitions, for example, to implement them into your **WebApplicationContext** beans. If you make a **getBean** call to **yourWebApplicationContext**, it is checked whether there is a definition available. If not, the **parentApplica-**

		tionContext is used.
2	RequestContextListener	If you want to enable the usage of the web application-specific scope: session and request, you must add the Spring **RequestContextListener**.

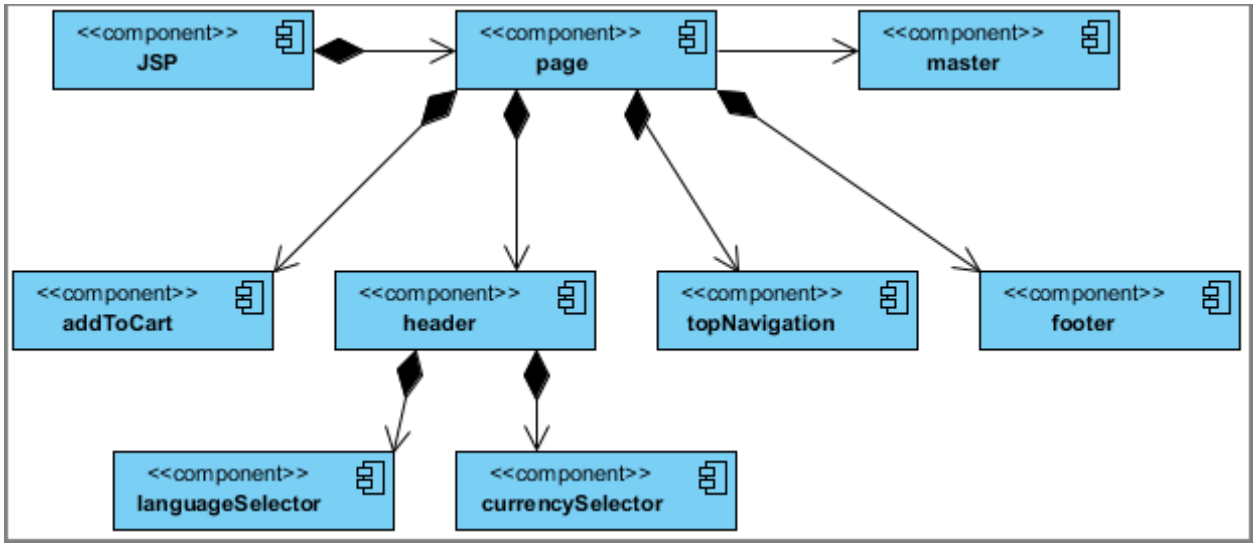
5.2.3 Servlet Mapping

Servlet mapping specifies the web container of which a Java servlet should be invoked from a URL given by a client.

S. No	Servlet	Description
1	DispatcherServlet	The storefront web application uses the Spring MVC framework to handle requests. The core of the Spring MVC framework is the DispatcherServlet. It has to be configured in your web.xml file for activating the dispatching of incoming requests to the special MVC controller

5.2.4 Tag files

The view is rendered using JSPs and tag files. JSPs include several tag files to impose a general, reusable structure shown in the next diagram:



The following tag components are provided in /WEB-INF/tags folder:

Component	Filename	Description
Page	template/page.tag	General page structure
master	template/master.tag	Master HTML template
header	common/header/header.tag	Header bar
languageSelector	common/header/languageSelector.tag	Language selector
currencySelector	common/header/currencySelector.tag	Currency selector
topNavigation	nav/topNavigation.tag	Top navigation
footer	common/footer/footer.tag	Footer
addToCart	cart/addToCart.tag	Ajax cart pop-up

6 Site Navigation

Yacceleratorstorefront has 3 WCMS sites for electronic store, apparel UK store and apparel DE store with URL patter as following:

1. Electronics Store:
    a. (?i)^https?://[^/]+(/[^?]*)?\?(.*\&)?(site=electronics)(|\&.*)$
    b. (?i)^https?://electronics \.[^/]+(|/.*|\?.*)$
2. Apparel DE Store:
    a. (?i)^https?://[^/]+(/[^?]*)?\?(.*\&)?(site=apparel-de)(|\&.*)$
    b. (?i)^https?://apparel-de\.[^/]+(|/.*|\?.*)$
3. Apparel UK Store:
    a. (?i)^https?://[^/]+(/[^?]*)?\?(.*\&)?(site=apparel-uk)(|\&.*)$
    b. (?i)^https?://apparel-uk\.[^/]+(|/.*|\?.*)$

So it can be accessible via below URL's based on host entry @C:\Windows\System32\drivers\etc\hosts:

https://localhost:9002/yacceleratorstorefront/?site=apparel-uk

https://apparel-uk.local:9002/yacceleratorstorefront/?site=apparel-uk

6.1 Design Overview

Hybris B2C storefront's front-end logic is implemented using Spring MVC and follows Spring MVC framework pattern. The Spring MVC related configuration files are located in the **web/webroot/WEB-INF/config** folder of the **yacceleratorstorefront** extension. The controllers themselves are configured by annotation as per hybris conventions.

6.2 Technical Design

The core of the Spring MVC framework is the **DispatcherServlet**, which is configured in your **web.xml** file to activate the dispatching of incoming requests to the special MVC controller. Based on configuration in **web.xml**, each requests invokes filter chain **`storefrontTenantFilterChain`** to filter request.

```
<filter-mapping><filter-name>storefrontTenantFilterChain</filter-name><servlet-name>DispatcherServlet</servlet-name></filter-mapping>

<servlet>
        <description>
                DispatcherServlet
                Spring MVC dispatcher servlet. This is the entry point for the Spring MVC application.
        </description>
        <servlet-name>DispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <description>
                        Specifies the location for Spring MVC to load an additional XML configuration
file.Because hybris is already configured with the XML spring configuration files to load we must set this
param value to EMPTY in order to prevent loading of the default /WEB-INF/applicationContext.xml file.
                </description>
                <param-name>contextConfigLocation</param-name>
                <param-value></param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
```

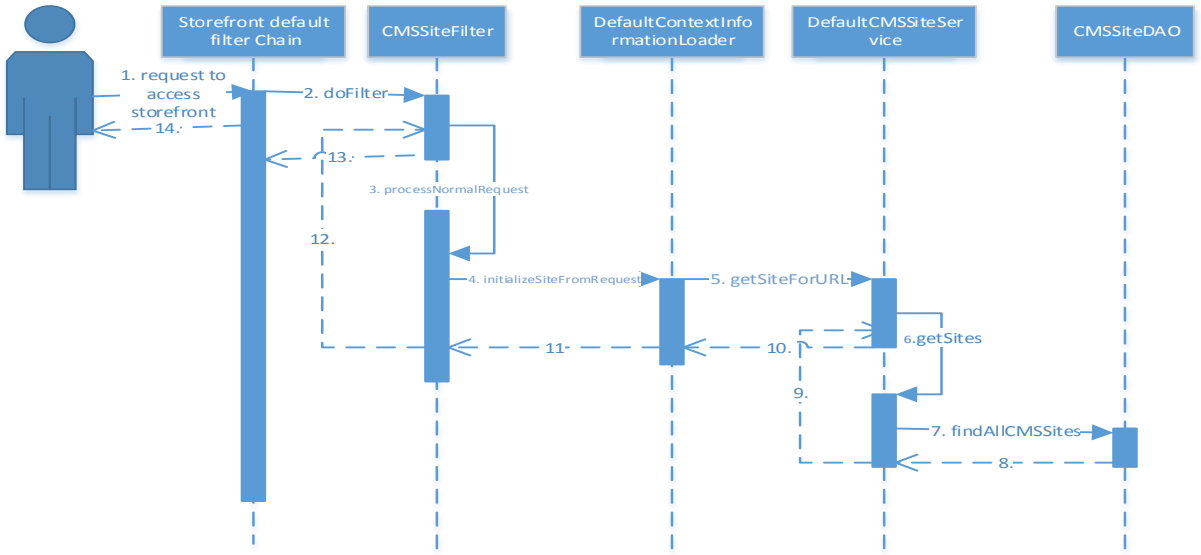
```
<servlet-mapping>
        <servlet-name>DispatcherServlet</servlet-name>
        <!-- Map all requests to the DispatcherServlet -->
        <url-pattern>/</url-pattern>
</servlet-mapping>
```

`StorefrontTenantFilterChain` filter is configured in **spring-filter-config.xml**, which invokes filter chain **store-frontTenantDefaultFilterChain** that's define list of applicable filters like logging filter, session filter , CMSSite filter etc. . **CSMSite** filter is responsible to filter out site for requested URL.

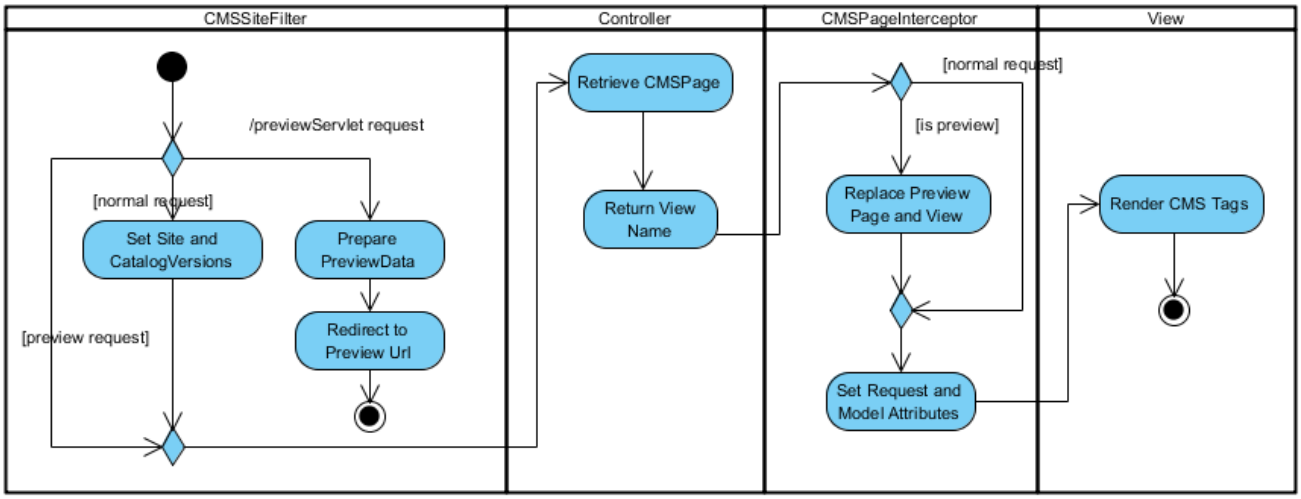
```
<bean id="storefrontTenantFilterChain"
class="de.hybris.platform.yacceleratorstorefront.filters.UrlPathFilter" >
        <property name="defaultFilter" ref="storefrontTenantDefaultFilterChain"/>
        <property name="urlPathHelper">
                <bean class="org.springframework.web.util.UrlPathHelper"/>
        </property>
        <property name="urlPathMapping">
                <map>
                        <entry key="/integration/" value-ref="integrationTenantFilterChain"/>
                </map>
        </property>
</bean>

<bean id="storefrontTenantDefaultFilterChain" class="de.hybris.platform.servicelayer.web.PlatformFilterChain" >
        <constructor-arg>
                <ref bean="storefrontTenantDefaultFilterChainList"/>
        </constructor-arg>
</bean>
```

```
<alias name="defaultStorefrontTenantDefaultFilterChainList" alias="storefrontTenantDefaultFilterChainList" />
        <util:list id="defaultStorefrontTenantDefaultFilterChainList">
                <!-- generic platform filters -->
                <ref bean="log4jFilter"/>
                <ref bean="storefrontSessionFilter"/>
                <ref bean="addOnDevelopmentFilter"/>
                <!-- filter to log the current request -->
                <ref bean="requestLoggerFilter"/>
                <!-- filter to setup the cms integration -->
                <ref bean="cmsSiteFilter"/>
                <!-- filter to initialize the storefront -->
                <ref bean="storefrontFilter"/>
                <!-- filter to handle url encoding attributes -->
                <ref bean="urlEncoderFilter"/>
                <!-- filter to handle multipart file upload -->
                <ref bean="fileUploadFilter"/>
                <!-- Security -->
                <ref bean="springSecurityFilterChain"/>
                <!--filter to log out guest user if he/she attempts to access a page outside of checkout flow -
->
                <ref bean="anonymousCheckoutFilter"/>
                <!-- filter to restore items in cart -->
                <ref bean="cartRestorationFilter"/>
                <!-- filter to restore customer preferred location -->
                <ref bean="customerLocationRestorationFilter"/>
        </util:list>
```

Sequence Diagram : Following sequence diagram displays that to fetch CMS site for requested URL:



The general page processing for any user request is propagate as follow in **yacceleratorstorefront** extension regarding WCMS:



As shown, there are four major steps during processing:

1. General session setup and preview handling that is session setup and redirect.
2. Page and view resolution in the Spring MVC Controller.
3. Further population of request and model attributes in the Interceptor with additional page and view replacement functionality during preview.
4. View rendering using CMS tags, for example **<cms:body>, <cms:slot>, or <cms:component>.**

7 Homepage

This section describes design for home page for **apparel-uk** storefront and the different sections contained within the home page.

7.1 Frontend Design

The following diagram provides a visual outline of each of the section of the home page:

Diagram: 7.1

7.2 Design Overview

The home page will be created from the home page template, which defines the layout of the page. The home page template has a corresponding JSP file that defines the structure of the page by including different sections of the page.

Each section then includes the corresponding CMS components defined for that section. Each CMS component will have a corresponding JSP file; the JSP file will retrieve data from the Page controller. The look and feel of the component would be driven through CSS.

Any page would be rendered using combination of two technologies:

- Front-end technology: the look and feel of the page would be driven primarily through CSS and JavaScript
- Backend CMS components: responsible for providing the data to the front-end technology

Hybris OOB provided 6 different page template for storefront homepage, the apparel-uk store home page would be created from the **landingPage2Template**, this page template renders from **landingLayout2Page.jsp**. This page defines the layout of the home page. Business user will manage the home page variations in the WCMS cockpit and configure which version of the page would be active for a given period of time.

7.3 Technical Design

Spring MVC request handler maps URL with **@RequestMapping("/")**, and passes control to **HomepageController**. This Controller supports switchable views. The view is manageable in the WCMS Cockpit by setting the frontendTemplat-eName. **HomepageController** extends **AbstractPageController.**

AbstractPageController

This abstract class is the base class of all controllers in the **yacceleratorstorefront** extension and has the responsibility to provide:

1. Look-up functionality for CMS pages.
2. Look-up functionality for retrieving the view name for controllers supporting switchable views, for example **HomePageController**.
3. Convenient methods to set basic model attributes that is **user**, **currencies**, **languages**, **currentCurrency**, **user**, and request.

HomePageController

This controller handles request for Home Page. This controller has below responsibility to provide:

```
@RequestMapping(method = RequestMethod.GET)
	public String home(@RequestParam(value = "logout", defaultValue = "false") final boolean logout, final
Model model,
			final RedirectAttributes redirectModel) throws CMSItemNotFoundException
	{
		storeCmsPageInModel(model, getContentPageForLabelOrId(null));
		setUpMetaDataForContentPage(model, getContentPageForLabelOrId(null));
		updatePageTitle(model, getContentPageForLabelOrId(null));

		return getViewForPage(model);
	}
```

First, the method **home** is mapped to the correct path and request method. Then, **getContentPageForLabelOrId** is called, which in turn performs the following calls:

1. If the parameter **labelOrId** is **null** or empty, it retrieves the label for the homepage from **CMSPageService**.
2. If no page is marked as homepage, it retrieves the label for the starting page for the current site.
3. Returns the page for the resolved key by calling **getPageForLabelOrId** on **CMSPageService**.

The retrieved page is stored as the model attribute **CMS_PAGE_MODEL**. In a second step, the view is retrieved by calling getViewForPage, which is achieved by receiving the previously set page from the model and accessing the attribute **frontendTemplateName** of master template as shown here in the AbstractPageController:

```
protected String getViewForPage(final AbstractPageModel page)
	{
		if (page != null)
		{
			final PageTemplateModel masterTemplate = page.getMasterTemplate();
			if (masterTemplate != null)
			{
				final String targetPage = cmsPageServ-
ice.getFrontendTemplateName(masterTemplate);
				if (targetPage != null && !targetPage.isEmpty())
				{
					return PAGE_ROOT + targetPage;
				}
			}
		}
		return null;
	}
```

This allows you to exchange the landing page layout easily by switching to a different landing page template in the WCMS Cockpit. Besides, the **frontendTemplateName** could also be updated directly to a different JSP page supporting the given template layout.

CMS Page Configuration:

You can configure templates and pages in the WCMS Cockpit or by using an ImpEx script. Due to the complexity of the data, the more common approach is to use ImpEx. To do so, perform these steps:

1. Define the **PageTemplate**.
2. Define the **ContentSlotName**.
3. Define the page.
4. Define the **ContentSlot**.
5. Assign reusable content slots to the template in the **ContentSlotsForTemplate**.
6. Assign specific content slots to the page in the **ContentSlotsForPage**.

7.4 Impex Script

Sample Impex from Core Data:

```
$contentCatalog=apparel-ukContentCatalog
$con-
tentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[defaul
t=Staged])[default=$contentCatalog:Staged]

# Import modulegen config properties into impex macros
UPDATE GenericI-
tem[processor=de.hybris.platform.commerceservices.impex.impl.ConfigPropertyImportProcessor];pk[unique=true]
$jarResourceCms=$config-jarResourceCmsValue

# Create PageTemplates
# These define the layout for pages
# "FrontendTemplateName" is used to define the JSP that should be used to render the page for pages with multi-
ple possible layouts.
# "RestrictedPageTypes" is used to restrict templates to page types
INSERT_UPDATE PageTem-
plate;$contentCV[unique=true];uid[unique=true];name;frontendTemplateName;restrictedPageTypes(code);active[defau
lt=true]
;;LandingPage2Template;Landing Page 2 Template;layout/landingLayout2Page;CategoryPage,ContentPage
```

```
# Landing Page Templates
INSERT_UPDATE ContentSlot-
Name;name[unique=true];template(uid,$contentCV)[unique=true][default='LandingPage2Template'];validComponentType
s(code);compTypeGroup(code)
;SiteLogo;;;logo
;HeaderLinks;;;headerlinks
;SearchBox;;;searchbox
;MiniCart;;;minicart
;NavigationBar;;;navigation
;Section1;;;wide
;Section2A;;;narrow
;Section2B;;;narrow
;Section2C;;;wide
;Section3;;;wide
;Section4;;;narrow
;Section5;;;wide
;Footer;;;footer
;TopHeaderSlot;;;wide
;BottomHeaderSlot;;;wide
;PlaceholderContentSlot;;;
```

```
# Create Content Pages
# Site-wide Homepage
INSERT_UPDATE Con-
tentPage;$contentCV[unique=true];uid[unique=true];name;masterTemplate(uid,$contentCV);label;defaultPage[default
```

```
='true'];approvalStatus(code)[default='approved'];homepage[default='true']
;;homepage;Homepage;LandingPage2Template;homepage
```

```
# Create ContentSlots
INSERT_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];name;active
;;SiteLogoSlot;Default Site Logo Slot;true
;;HomepageNavLinkSlot;Default Homepage Link;true
;;NavigationBarSlot;Navigation Bar;true
;;MiniCartSlot;Mini Cart;true
;;FooterSlot;Footer;true
;;HeaderLinksSlot;Header links;true
;;SearchBoxSlot;Search Box;true
;;TopHeaderSlot;Top Header;true
;;BottomHeaderSlot;Bottom Header;true
;;PlaceholderContentSlot;Placeholder for Addon tag files;true
```

```
# Bind Content Slots to Page Templates
INSERT_UPDATE ContentSlotFor-
Tem-
plate;$contentCV[unique=true];uid[unique=true];position[unique=true];pageTemplate(uid,$contentCV)[unique=true][
default='LandingPage2Template'];contentSlot(uid,$contentCV)[unique=true];allowOverwrite
;;SiteLogo-LandingPage2;SiteLogo;;SiteLogoSlot;true
;;HomepageLink-LandingPage2;HomepageNavLink;;HomepageNavLinkSlot;true
;;NavigationBar-LandingPage2;NavigationBar;;NavigationBarSlot;true
;;MiniCart-LandingPage2;MiniCart;;MiniCartSlot;true
;;Footer-LandingPage2;Footer;;FooterSlot;true
;;HeaderLinks-LandingPage2;HeaderLinks;;HeaderLinksSlot;true
;;SearchBox-LandingPage2;SearchBox;;SearchBoxSlot;true
;;TopHeaderSlot-LandingPage2;TopHeaderSlot;;TopHeaderSlot;true
;;BottomHeaderSlot-LandingPage2;BottomHeaderSlot;;BottomHeaderSlot;true
;;PlaceholderContentSlot-LandingPage2;PlaceholderContentSlot;;PlaceholderContentSlot;true
```

Sample Impex from SampleData: To assign specific content slot to pages.

```
# ImpEx for Apparel UK Site CMS Content
#
$contentCatalog=apparel-ukContentCatalog
$con-
tentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[defaul
t=Staged])[default=$contentCatalog:Staged]

$productCatalog=apparelProductCatalog
$productCatalogName=Apparel Product Catalog
$productCV=catalogVersion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$
productCatalog:Staged]
$picture=media(code, $contentCV) ;
$siteRe-
source=jar:de.hybris.platform.apparelstore.constants.ApparelstoreConstants&/apparelstore/import/sampledata/cont
entCatalogs/$contentCatalog
$jarRe-
sourceCms=jar:de.hybris.platform.apparelstore.constants.ApparelstoreConstants&/apparelstore/import/sampledata/c
ockpits/cmscockpit

# Load the storefront context root config param
$storefrontContextRoot=$config-storefrontContextRoot

###### Apparel UK Homepage

# ImageMapComponent
INSERT_UPDATE ImageMapCompo-
nent;$contentCV[unique=true];uid[unique=true];name;imageMapHTML[lang=en];&componentRef;urlLink;;
;;SummerClassicsBanner;Summer Classics Banner;<area shape='rect' coords='863,504,949,527'
href='$storefrontContextRoot/Brands/c/brands' alt='all-brands' /><area shape='rect' coords='782,493,843,527'
href='$storefrontContextRoot/Brands/F2-FTWO/c/F2-FTWO' alt='f2' /><area shape='rect' coords='673,493,777,527'
href='$storefrontContextRoot/Brands/Quiksilver/c/Quiksilver' alt='quiksilver' /><area shape='rect'
coords='617,493,671,527' href='$storefrontContextRoot/Brands/Roxy/c/Roxy' alt='roxy' /><area shape='rect'
coords='511,493,614,527' href='$storefrontContextRoot/Brands/Dakine/c/Dakine' alt='dakine' /><area shape='rect'
```

```
coords='433,493,507,527' href='$storefrontContextRoot/Brands/Fox/c/Fox' alt='fox' /><area shape='rect'
coords='356,493,426,527' href='$storefrontContextRoot/Brands/Rip-Curl/c/Rip Curl' alt='ripcurl' /><area
shape='rect' coords='287,493,354,527' href='$storefrontContextRoot/Brands/Toko/c/Toko' alt='toko' /><area
shape='rect' coords='169,493,280,527' href='$storefrontContextRoot/Brands/Dainese/c/Dainese' alt='dainese'
/><area shape='rect' coords='82,493,165,527' href='$storefrontContextRoot/Brands/Playboard/c/Playboard'
alt='playboard' /><area shape='rect' coords='9,493,77,527' href='$storefrontContextRoot/Brands/Vans/c/Vans'
alt='vans' /><area shape='poly' coords='66,166' href='#' /><area shape='poly'
coords='5,212,66,169,156,162,249,215,285,332,260,351,274,473,5,474'
href='$storefrontContextRoot/Brands/Playboard/T-Shirt-Men-Playboard-Raster-SS/p/M34704_B' alt='Playboard Ras-
ter' /><area shape='poly' coords='296,386,375,392,397,358,552,338,542,430,515,467,386,470,291,418'
href='$storefrontContextRoot/Brands/Fox/Shades-Fox-The-Median-polished-black-warm-gray/p/300024964' alt='Fox
The Median' /><area shape='poly' coords='624,427,788,474,923,466,928,411,757,324,644,321'
href='$storefrontContextRoot/Brands/Vans/Slip-On-Vans-Classic-Slip-On/p/M35392' alt='Vans Classic Slip On'
/><area shape='poly' coords='738,312,929,297,945,4,750,9,703,130,758,151'
href='$storefrontContextRoot/Brands/Playboard/T-Shirt-Men-Playboard-Logo/p/M35364_R' alt='Playboard Logo Tee'
/>;SummerClassicsBanner;#;;

# ContentSlot
INSERT_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];name;active;cmsComponents(uid,$contentCV);;;
;;Section1Slot-ApparelUKHomepage;Section1 Slot for ApparelUKHomepage;true;SummerClassicsBanner;;;

# ContentSlotForPage
INSERT_UPDATE ContentSlot-
ForPage;$contentCV[unique=true];uid[unique=true];position[unique=true];page(uid,$contentCV)[unique=true][defaul
t='homepage'];contentSlot(uid,$contentCV)[unique=true];;;
;;Section1-ApparelUKHomepage;Section1;;Section1Slot-ApparelUKHomepage;;;
```

8 Header

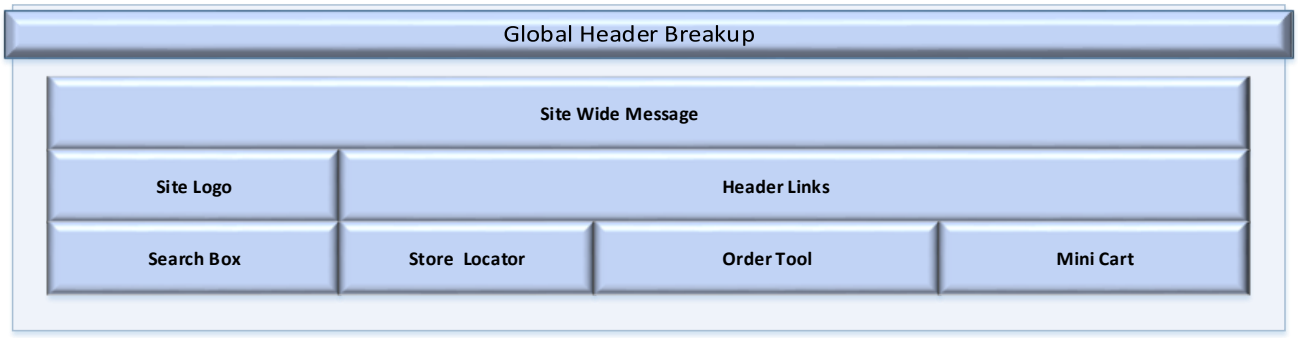
This section details the design of header view of apparel storefront. Header is global component that would be shared across pages. Header consists of following components:

- Site wide Outage Messaging
- Site Logo
- Header links
- Search Bar
- Store finder
- Order Tool
- Mini cart

Header links component varies for anonymous and logged in user. Header.tag file defines the layout and look and feel of the header section, which is leverage through different components.

8.1 Front end design

The header section layout displayed below. Each section has one or more components used to render the section. CSS and JavaScript govern the look and feel of the page.



8.2 Content Slots and Components

Each of the annotations provided in the Global Header Wireframe is detailed in the following table:

Wire Section Name	Content Slot Name	Component	Business Managed
Site wide Outage Messaging	Top Header Slot	CMSParagraphComponent	Yes
Site Logo	Site Logo	SimpleBannerComponent /LogoComponent	Yes
Header Link	Header Links	header.tag file	No
Search Box	Search Box	SearchBoxComponent	No
Store Locator	-	CMSLinkComponent	Yes
Order Tool	-	CMSLinkComponent	Yes
Minicart	Minicart	MiniCartComponent	No

8.3 Design Details

The landingLayout2Page.jsp defines the home page as mentioned earlier in this TDD. This JSP defines the header by including the header.tag file via the template:page tag directive.

The header.tag file defines the layout and look and feel of the header section of the page. The header section is rendered leveraging through different components that are defined as follows.

1) **Site wide outage messaging** :

Business users will be able to Add/edit the message, control the time when the message would appear and be able to remove the content.

Component / JSP	Description
CMSParagraphComponent	This component would be used to define the outage messaging. This component would be placed in the Top header slot of the home page template.
CMSTimeRestriction	This component would be used to specify the period when the message would be displayed on the site
cmstabparagraphcomponent.jsp	This file defines the style for CMSParagraphComponent that would be displayed on the front end. If the site wide outage message were not configured or active that section would be collapsed using CSS.

2) **Site logo:**

Business users will be able to change the logo image and URL the user is redirected to on-click of the logo.

Component / JSP	Description
SimpleBannerComponent	This component would be used to define the logo. This component would be placed in the Site Logo slot of the home page template. The SimpleBannerComponent would contain the media (the logo) and the destination URL where the user would be taken when they click on logo
logocomponent.jsp	This file defines the style for how the logo would be displayed on the front end.

3) **Header links :**

The header link section contains links that are dynamic based on user state. The text and links cannot be modified by business through WCMS and would need configuration change.

Tag / Property file	Description
header.tag	The file contains style that defines how the tag would be displayed on the front end as well as the destination URL for those links.
base_en.properties	The text of the links is defined in this property file.

4) **Search bar :**

Search bar would be render by SearchBoxComponent; the copy in the search bar would be configured through property file.

Component / JSP / Property file	Description

SearchBoxComponent	This component is used to display search bar.
searchboxcomponent.jsp	This file defines the search text field and search button.
base_en.properties	The copy of the search bar is defined in this property file.

5) **Mini Cart :**

The mini cart component displays the summarized information on the customer's current cart status.

Component/Tag / Property file	Description
MiniCartComponent	This component is used to display mini cart on the header section.
minicartcomponent.jsp	▪ The count displayed on the mini cart would be retrieved from for current user session. ▪ The cart is set/reset when the user clicks on “Add To Cart” button, mini cart is loaded or when the cart page is loaded.

8.4 Impex Script

```
$contentCatalog=apparel-ukContentCatalog
$con-
tentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[defaul
t=Staged])[default=$contentCatalog:Staged]

$productCatalog=apparelProductCatalog
$productCatalogName=Apparel Product Catalog
$productCV=catalogVersion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$
productCatalog:Staged]
$picture=media(code, $contentCV) ;
$siteRe-
source=jar:de.hybris.platform.apparelstore.constants.ApparelstoreConstants&/apparelstore/import/sampledata/cont
entCatalogs/$contentCatalog
$jarRe-
sourceCms=jar:de.hybris.platform.apparelstore.constants.ApparelstoreConstants&/apparelstore/import/sampledata/c
ockpits/cmscockpit

# Load the storefront context root config param
$storefrontContextRoot=$config-storefrontContextRoot
# Site Logo Component
INSERT_UPDATE SimpleBannerComponent;$contentCV[unique=true];uid[unique=true];name;&componentRef;urlLink
;;SiteLogoComponent;Site Logo Component;SiteLogoComponent;"/"

# Site Logo Component
UPDATE SimpleBannerComponent;$contentCV[unique=true];uid[unique=true];$picture[lang=$lang]
;;SiteLogoComponent;/images/theme/logo-hybris.png

# CMS SearchBox Components
INSERT_UPDATE SearchBoxComponent;$contentCV[unique=true];uid[unique=true];name;&componentRef
;;SearchBox;Search Box;SearchBox

# CMS Paragraph Component (Contact information)
INSERT_UPDATE CMSParagraphComponent;$contentCV[unique=true];uid[unique=true];name;&componentRef;;;;
;;ContactInfo;Contact information;ContactInfo;;;;


# CMS Mini Cart Component
INSERT_UPDATE MiniCartCompo-
nent;$contentCV[unique=true];uid[unique=true];name;&componentRef;totalDisplay(code);shownProductCount;;
;;MiniCart;Mini Cart;MiniCart;SUBTOTAL;3;;
# CMS Mini Cart Component
```

```
UPDATE MiniCartComponent;$contentCV[unique=true];uid[unique=true];title[lang=en]
;;MiniCart;"Your Shopping Bag"


# ContentSlot
INSERT_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];cmsComponents(uid,$contentCV)
;;HeaderLinksSlot;ContactInfo
;;SearchBoxSlot;SearchBox,LangCurrencyComponent
;;HomepageNavLinkSlot;HomepageNavLink
;;MiniCartSlot;MiniCart
;;SiteLogoSlot;SiteLogoComponent
```

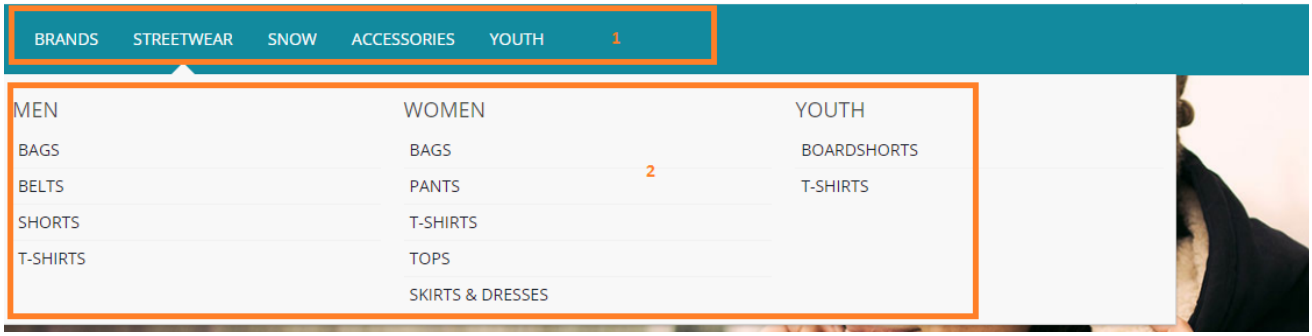
9 Core Navigation

Core navigation consists of following components.

- Top navigation
- Mega menu

9.1 Frontend Design

Refer below for core navigation on apparel-uk website:



9.2 Content Slots and Components

Core Navigation Wireframe is detailed in the following table:

Wire Section Name	Content Slot Name	Component	Component Source	Business Managed
Core Navigation	Navigation Bar Slot	NavigationBarCollec-tionComponent	Accelerator	Yes
Core Navigation	Navigation Bar Slot	NavigationBarComponent	Accelerator	Yes
Core Navigation	Navigation Bar Slot	CMSNavigationNode		Yes
Core Navigation	Navigation Bar Slot	CMSLinkComponent	Accelerator	Yes

9.3 Design Detail

NavigationBarCollectionComponent would be used to configure the top navigation section of the core navigation.

Mega menu has three levels, which would be rendered using the following components:

- Level 1: NavigationBarComponent would be used to define the level 1 of the mega menu.
- Level 2: CMSNavigationNode component would be used to define level 2 of the mega menu, which can be parent of other CMSNavigationNodes.
- Level 3: CMSLinkComponent would be used to define level 3 of the mega menu.

The components for top navigation and mega menu are defined below:

1) **Top navigation :**

Component / JSP	Description
NavigationBarCollectionComponent	This component contains list of NavigationBarComponent.

navigationbarcollectioncompo-nent.jsp	This file defines the style for how the top navigation would be dis-played

2) **Mega Menu Level 1 :**

Component / JSP	Description
NavigationBarComponent	This component contains CMSNavigationNode.
navigationbarcomponent.jsp	This file defines the style for how the top navigation would be dis-played

3) **Mega Menu Level 2 :**

Component / JSP	Description
CMSNavigationNode	This component define node with the top-level navigation, which also contains list of other navigation nodes.
navigationbarcomponent.jsp	Same jsp file of navigationbarcomponent.jsp will be used.

4) **Mega Menu Level 3 :**

Component / JSP	Description
CMSLinkComponent	This component would define the link at last level in mega menu.
CMSLinkComponentRenderer.java	CMSLinkComponentRenderer in renders CMSLinkComponentModel.

9.4 Impex Script

Sample impex for Core Navigation of apparel-uk storefront :

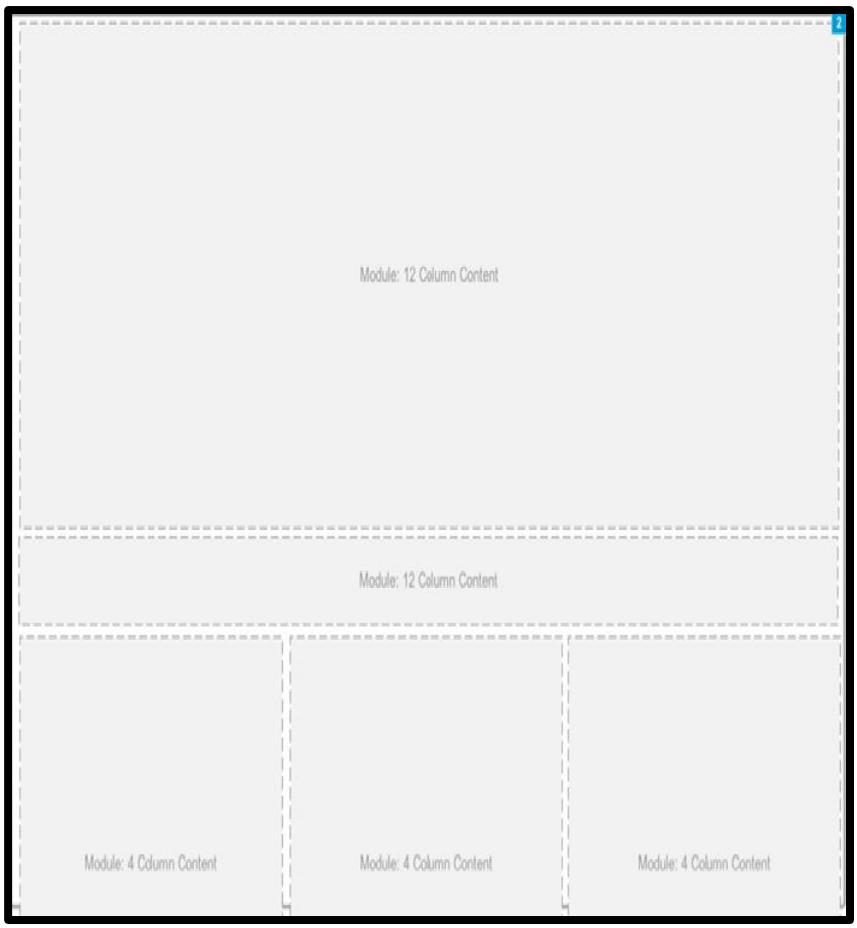
Compo-nents	Impex Script
Naviga-tionBarCol-lec-tionCom-ponent	`# Navigation Bar Component` `INSERT_UPDATE NavigationBarCollectionCompo-nent;$contentCV[unique=true];uid[unique=true];name;components(uid, $contentCV);&componentRef ;;NavBarComponent;Navigation Bar Collection Compon-eent;BrandsBarComponent,StreetwearBarComponent,SnowBarComponent,AccessoriesBarComponent,StreetweaerYouthBarComponent,SpecialOffersBarComponent;NavBarComponent`
Naviga-tion-BarCompo-nent	`INSERT_UPDATE NavigationBarCompo-nent;$contentCV[unique=true];uid[unique=true];name;wrapAfter;link(uid, $con-tentCV);styleClass;navigationNode(uid,$contentCV);dropDownLayout(code)[default='AUTO'] ;;StreetwearBarComponent;Streetwear Menu Bar Item;10;StreetwearLink;;StreetwearNavigationNode; ;;SnowBarComponent;Snow Menu Bar Item;10;SnowLink;;SnowNavigationNode; ;;BackpacksBarComponent;Backpacks Bar Component;5;BackpacksLink;;BackpacksNavNode; ;;BrandsBarComponent;Brands Bar Component;10;AllBrandsLink;;BrandsNavNode; ;;CategoriesBarComponent;Categories Bar Component;10;AllCategoriesLink;;CategoriesNavNode; ;;ShirtsBarComponent;Shirts Bar Component;5;ShirtsLink;;ShirtsNavNode; ;;ShoesBarComponent;Shoes Bar Component;5;ShoesLink;;ShoesNavNode; ;;SpecialOffersBarComponent;Special Offers Bar Component;10;SpecialOffersLink;special_offer;; ;;SunglassesBarComponent;Sunglasses Bar Component;5;SunglassesLink;;SunglassesNavNode; ;;StreetwearYouthBarComponent;Streetwear Youth Bar Compo-nent;10;YouthLink;;StreetwearYouthNavNode;`

	;;AccessoriesBarComponent;Accesories Bar Component;10;AccessoriesLink;;AccessoriesNavigationNode; ;;SurfBarComponent;Surf Bar Component;15;SurfLink;;SurfNavigationNode;
CMSNavi-gationNode	# CMS Navigation Nodes INSERT_UPDATE CMSNavigationNode;uid[unique=true];$contentCV[unique=true];name;parent(uid, $con-tentCV);children(uid,$contentCV)[mode=append];links(uid, $contentCV);&nodeRef ;root;;root;;;;root; ;AcceleratorNavNode;;Accelerator Pag-es;ApparelUKNavNode;;AboutAcceleratorLink,FAQLink;AcceleratorNavNode ;CategoriesNavNode;;All Categories;ApparelUKNavNode;;;CategoriesNavNode ;BrandsNavNode;;All Brands;ApparelUKNavNode;;;BrandsNavNode ;SiteRootNode;;SiteRootNode;root;;;SiteRootNode; ;AllBrandsNavNode;;Brand Links;BrandsNavNode;;667Link,686Link,69SlamLink,adidasOriginalsLink,AlMerrickLink,AlienWorkshopLi nk,AnalogLink,AnonLink,ArmadaLink,BenchLink,BillabongLink,BlueTomatoLink,BurtonLink,CarharttLink, DCLink,DaineseLink,DakineLink,DalbelloLink,droidLink,ElementLink,ElmLink,F2- FTWOLink,FemipleasureLink,FoursquareLink,FoxLink,HorsefeathersLink,HurleyLink,LandingLink,NikeLin k,Nike6.0Link,NixonLink,OakleyLink,OrageLink,PYUALink,PlayboardLink,ProTecLink,QuiksilverLink,Red Link,ReefLink,RipCurlLink,RoxyLink,SessionsLink,SpecialBlendLink,TokoLink,VIVOLink,VansLink,Volco mLink,VonZipperLink,WLDLink,ZimtsternLink;AllBrandsNavNode; ;AllCategoriesNavNode;;Category Links;CategoriesNavNode;;ClothesLink,OthersLink,CapsLink,SkiGearLink,SunglassesLink,ShoesLink,Gog gles- Link,ShirtsLink,SnowLink,StreetwearLink,SurfLink,HelmetLink,ToolsLink,GuidesLink,BackpacksLink;Al lCategoriesNavNode; ;ApparelUKNavNode;;Apparel UK Site;SiteRootNode;;HomepageNavLink;ApparelUKNavNode; ;BackpacksBrandsNavNode;;Backpacks Brand Links;BackpacksNavNode;;DakineBackpacksLink;BackpacksBrandsNavNode; ;BackpacksNavNode;;Backpacks Category;ApparelUKNavNode;;;BackpacksNavNode;
CMSLinkCo mponent	INSERT_UPDATE CMSLinkComponent;$contentCV[unique=true];uid[unique=true];name;url;category(code, $productCV);target(code)[default='sameWindow'];; ;;667Link;667 Link;;667;;; ;;686Link;686 Link;;686;;; ;;69SlamLink;69 Slam Link;;69 Slam;;;

10 Main body

Main body of the homepage template is made of number of slots. Each of slots can contain different type of components.

10.1 Frontend Design



10.2 Content Slots and Components

The component that would be used is defined below.

Wire Section Name	Content Slot Name	Component	Component Source	Business Managed
Body part	Content Module slots	SimpleBannerComponent	Accelerator	Yes

10.3 Design Details

SimpleBannerComponent is used to populate each of the content slots.

Business managed content

Component / JSP	Description
SimpleBannerComponent	This component would be used to define the promotional content displayed on the site.
CMSTimeRestriction	This component would be used to specify the period when the content would be displayed on the site
simplebannercomponent.jsp	This file defines the style for SimpleBannerComponent that would be displayed on the front end. If the content were not defined or not active then the section of the page would be collapsed using CSS.

10.4 Impex Script

```
# ImageMapComponent
INSERT_UPDATE ImageMapCompo-
nent;$contentCV[unique=true];uid[unique=true];name;imageMapHTML[lang=en];&componentRef;urlLink;;
;;SummerClassicsBanner;Summer Classics Banner;<area shape='rect' coords='863,504,949,527'
href='$storefrontContextRoot/Brands/c/brands' alt='all-brands' /><area shape='rect' coords='782,493,843,527'
href='$storefrontContextRoot/Brands/F2-FTWO/c/F2-FTWO' alt='f2' /><area shape='rect' coords='673,493,777,527'
href='$storefrontContextRoot/Brands/Quiksilver/c/Quiksilver' alt='quiksilver' /><area shape='rect'
coords='617,493,671,527' href='$storefrontContextRoot/Brands/Roxy/c/Roxy' alt='roxy' /><area shape='rect'
coords='511,493,614,527' href='$storefrontContextRoot/Brands/Dakine/c/Dakine' alt='dakine' /><area shape='rect'
coords='433,493,507,527' href='$storefrontContextRoot/Brands/Fox/c/Fox' alt='fox' /><area shape='rect'
coords='356,493,426,527' href='$storefrontContextRoot/Brands/Rip-Curl/c/Rip Curl' alt='ripcurl' /><area
shape='rect' coords='287,493,354,527' href='$storefrontContextRoot/Brands/Toko/c/Toko' alt='toko' /><area
shape='rect' coords='169,493,280,527' href='$storefrontContextRoot/Brands/Dainese/c/Dainese' alt='dainese'
/><area shape='rect' coords='82,493,165,527' href='$storefrontContextRoot/Brands/Playboard/c/Playboard'
alt='playboard' /><area shape='rect' coords='9,493,77,527' href='$storefrontContextRoot/Brands/Vans/c/Vans'
alt='vans' /><area shape='poly' coords='66,166' href='#' /><area shape='poly'
coords='5,212,66,169,156,162,249,215,285,332,260,351,274,473,5,474'
href='$storefrontContextRoot/Brands/Playboard/T-Shirt-Men-Playboard-Raster-SS/p/M34704_B' alt='Playboard Ras-
ter' /><area shape='poly' coords='296,386,375,392,397,358,552,338,542,430,515,467,386,470,291,418'
href='$storefrontContextRoot/Brands/Fox/Shades-Fox-The-Median-polished-black-warm-gray/p/300024964' alt='Fox
The Median' /><area shape='poly' coords='624,427,788,474,923,466,928,411,757,324,644,321'
href='$storefrontContextRoot/Brands/Vans/Slip-On-Vans-Classic-Slip-On/p/M35392' alt='Vans Classic Slip On'
/><area shape='poly' coords='738,312,929,297,945,4,750,9,703,130,758,151'
href='$storefrontContextRoot/Brands/Playboard/T-Shirt-Men-Playboard-Logo/p/M35364_R' alt='Playboard Logo Tee'
/>;SummerClassicsBanner;#;;

# ContentSlot
INSERT_UPDATE ContentSlot;$contentCV[unique=true];uid[unique=true];name;active;cmsComponents(uid,$contentCV);;;
;;Section1Slot-ApparelUKHomepage;Section1 Slot for ApparelUKHomepage;true;SummerClassicsBanner;;;

# ContentSlotForPage
INSERT_UPDATE ContentSlot-
ForPage;$contentCV[unique=true];uid[unique=true];position[unique=true];page(uid,$contentCV)[unique=true][defaul
t='homepage'];contentSlot(uid,$contentCV)[unique=true];;;
;;Section1-ApparelUKHomepage;Section1;;Section1Slot-ApparelUKHomepage;;;
```

11 Footer

Footer is global component that would be shared across pages. Footer consists of following components:

- Accelerator
  - About Commerce Accelerator
  - FAQ
- Hybris
  - About Hybris
  - Contact Us
- Follow Us
  - Agile Commerce Blog
  - Linked In
  - Facebook
  - Twitter
- Copyright messages

Refer below for wireframe.

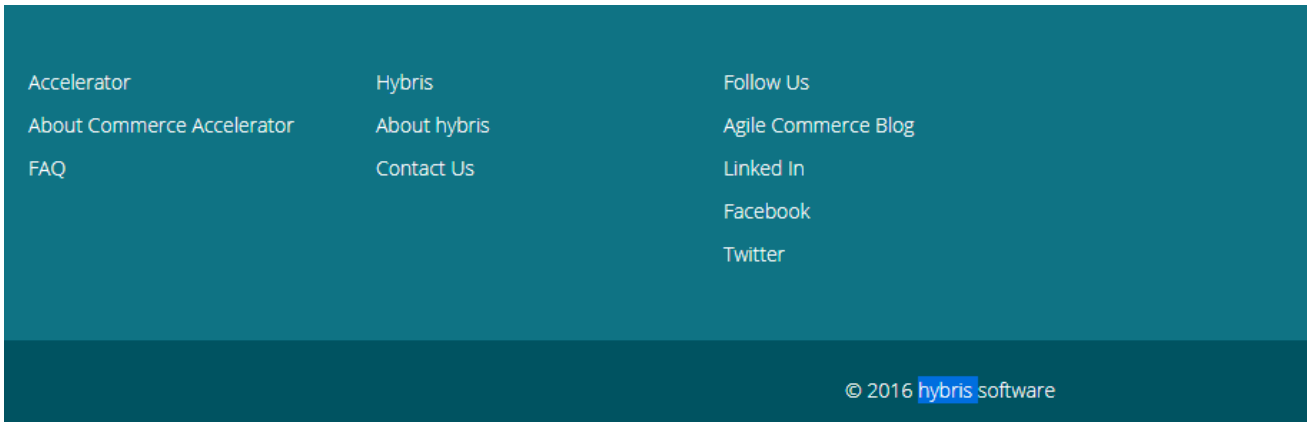
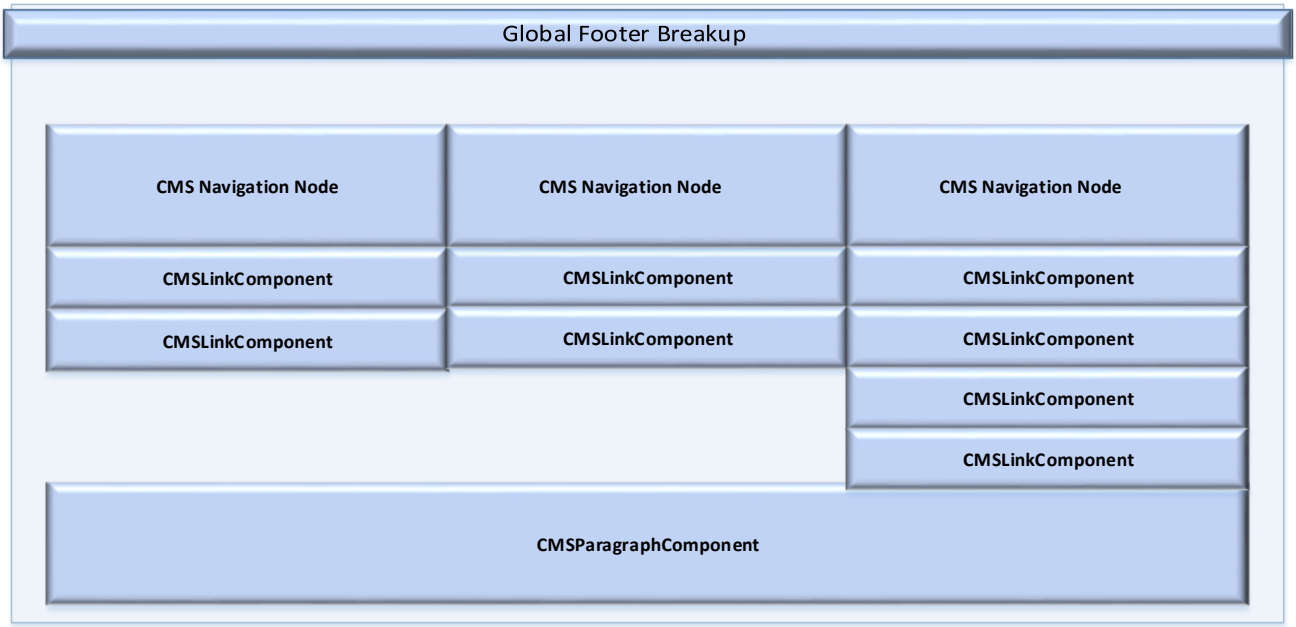


Fig 2: Global footer

11.1 Front end design

The above wireframe is realized using the component section layout displayed below. Each section has one or more components used to render the section. CSS and JavaScript govern the look and feel of the page.



11.2 Content Slots and Components

Each of the annotations provided in the Global Footer Wireframe is detailed in the following table:

Wire Section Name	Component	Component Source	Business managed
Accelorator	CMSNavigationNode	Accelerator	Yes
Hybris	CMSNavigationNode	Accelerator	Yes
Follow US	CMSNavigationNode	Accelerator	Yes
About Commerce Accelerator	CMSLinkComponent	Accelerator	Yes
FAQ	CMSLinkComponent	Accelerator	Yes
About Hybris	CMSLinkComponent	Accelerator	Yes
Contact Us	CMSLinkComponent	Accelerator	Yes
Agile Commerce Blog	CMSLinkComponent	Accelerator	Yes
Linked In	CMSLinkComponent	Accelerator	No
Facebook	CMSLinkComponent	Accelerator	Yes
Twitter	CMSLinkComponent	Accelerator	Yes
Copyright text	FooterComponent	Accelerator	Yes

11.3 Design Details

1) **CMS Navigation Node** :

The Accelerator, Hybris and Follow Us section of the wireframe use the same type of component. Business users will be able to Add/edit the text and links of the title. Business users will also be able to Add, Edit and remove links under each of these sections.

Component / JSP	Description

CMSNavigationNode	This component defines the title of the section and uses nodes to define the text in the section.
cmsnavigationnodecomponent.jsp	• This file defines the style for how the navigation node would be displayed. • The style class defined in the file would be used in CSS to control the look-n-feel of the links. • CSS would also be used to drive behavior like max characters acceptable in the links etc.

2) **CMS Link Components**:

The All Link of links sections of the wireframe use the same type of component. Business users will be able to Add/edit the text and link URLs.

Component / JSP	**Description**
CMSLinkComponent	This component would define the individual links. This component is also embedded in the NavigationNode component to form the links under “Accelorator”, “Hybris” and “Follow Us” section
CMSLinkComponentRenderer.java	CMSLinkComponentRenderer in renders CMSLinkComponentModel.

11.4 Impex Script

```
$contentCatalog=apparel-ukContentCatalog
$con-
tentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[defaul
t=Staged])[default=$contentCatalog:Staged]

$productCatalog=apparelProductCatalog
$productCatalogName=Apparel Product Catalog
$productCV=catalogVersion(catalog(id[default=$productCatalog]),version[default='Staged'])[unique=true,default=$
productCatalog:Staged]
$picture=media(code, $contentCV) ;
$siteRe-
source=jar:de.hybris.platform.apparelstore.constants.ApparelstoreConstants&/apparelstore/import/sampledata/cont
entCatalogs/$contentCatalog
$jarRe-
sourceCms=jar:de.hybris.platform.apparelstore.constants.ApparelstoreConstants&/apparelstore/import/sampledata/c
ockpits/cmscockpit

# Load the storefront context root config param
$storefrontContextRoot=$config-storefrontContextRoot


INSERT_UPDATE CMSLinkComponent;$contentCV[unique=true];uid[unique=true];name;url;category(code,
$productCV);target(code)[default='sameWindow'];;
;;AboutAcceleratorLink;About Accelerator Link;http://www.hybris.com/multichannel-accelerator;;newWindow;;
;;FAQLink;FAQ Link;/faq;;;
;;AboutHybrisLink;About Hybris Link;http://www.hybris.com/hybris/en.html;;newWindow;;
;;AgileCommerceBlogLink;Agile Commerce Blog Link;http://www.agile-commerce.com/;;newWindow;;
;;ContactUsLink;Contact Us Link;http://www.hybris.com/contact;;newWindow;;
;;FacebookLink;Facebook Link;http://www.facebook.com/hybrissoftware;;newWindow;;
;;LinkedInLink;Linked In Link;http://www.linkedin.com/company/97435;;newWindow;;
;;TwitterLink;Twitter Link;http://twitter.com/#!/hybris_software;;newWindow;;


# CMS Navigation Nodes
```

```
INSERT_UPDATE CMSNavigationNode;uid[unique=true];$contentCV[unique=true];name;parent(uid, $con-
tentCV);children(uid,$contentCV)[mode=append];links(uid, $contentCV);&nodeRef
;root;;root;;;;root;
;AcceleratorNavNode;;Accelerator Pages;ApparelUKNavNode;;AboutAcceleratorLink,FAQLink;AcceleratorNavNode
;FollowUsNavNode;;Follow Us Pag-
es;ApparelUKNavNode;;AgileCommerceBlogLink,LinkedInLink,FacebookLink,TwitterLink;FollowUsNavNode;
;HybrisNavNode;;Hybris Pages;ApparelUKNavNode;;AboutHybrisLink,ContactUsLink;HybrisNavNode;

# CMS Navigation Nodes
UPDATE CMSNavigationNode;$contentCV[unique=true];uid[unique=true];title[lang=en]
;;AcceleratorNavNode;"Accelerator"
;;FollowUsNavNode;"Follow Us"
;;HybrisNavNode;"Hybris"

# CMS Footer Component
INSERT_UPDATE FooterCompo-
nent;$contentCV[unique=true];uid[unique=true];wrapAfter;navigationNodes(uid,$contentCV);&componentRef;;;
;;FooterComponent;2;AcceleratorvNode,HybrisNavNode,FollowUsNavNode;FooterComponent;;;

# CMS Footer Component
UPDATE FooterComponent;$contentCV[unique=true];uid[unique=true];notice[lang=en]
;;FooterComponent;"© 2016 hybris software"
```

12 Appendix

12.1 Questions

Item #	Owner & Due Date	Question
1		