

Hybris cart module

Technical Design documentation

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Updated By | Date | Comments |
| 1.0 | Jayachandra Gudiwada | 11/02/2015 | Initial Draft |
|  |  |  |  |
|  |  |  |  |

Document Sign-Off

|  |  |  |
| --- | --- | --- |
| Name | Role | Signature/Date |
| Pascal ESPINOUSE |  |  |
| Aliasgar Muchhala |  |  |
| Gyaneshwar Dubey |  |  |

Table of Contents

[1. Introduction 4](#_Toc411536923)

[1.1 Purpose 4](#_Toc411536924)

[1.2 Intended Audience 4](#_Toc411536925)

[1.3 Document Scope 4](#_Toc411536926)

[1.4 Functional Rules 4](#_Toc411536927)

[1.5 Assumptions/Restrictions 4](#_Toc411536928)

[1.6 Documents Structure 5](#_Toc411536929)

[1.7 Additional Resources 5](#_Toc411536930)

[2. High Level Design Diagram 6](#_Toc411536931)

[2.1 Overview 6](#_Toc411536932)

[2.2 Sequence diagrams 7](#_Toc411536933)

[2.2.1 Create cart 7](#_Toc411536934)

[2.2.2 Select cart 8](#_Toc411536935)

[2.2.3 Set cart name 9](#_Toc411536936)

[2.2.4 Delete cart 10](#_Toc411536937)

[2.2.5 Share cart 11](#_Toc411536938)

[2.2.6 Unshare cart 12](#_Toc411536939)

[2.2.7 Log cart actions 13](#_Toc411536940)

[2.3 Use Case Diagram. 14](#_Toc411536941)

[3. Detailed Application Design 15](#_Toc411536942)

[3.1 Class diagram. 15](#_Toc411536943)

[3.2 Beans 16](#_Toc411536944)

[3.3 Services 16](#_Toc411536945)

[3.4 Facades 16](#_Toc411536946)

[3.5 Strategy 16](#_Toc411536947)

[3.6 Controller 16](#_Toc411536948)

[3.7 View/JSP 17](#_Toc411536949)

[The mock-up of the Multi carts JSP pages are given below. 17](#_Toc411536950)

[3.7.1 My account page. 17](#_Toc411536951)

[3.7.2 List of the carts 18](#_Toc411536952)

[3.7.3 Share and select the cart 18](#_Toc411536953)

[3.7.4 Delete the cart 19](#_Toc411536954)

[3.7.5 History of the Cart 19](#_Toc411536955)

[4. Database architecture 20](#_Toc411536956)

[4.1 ER Diagram 20](#_Toc411536957)

[5. Exception Handling and Logging 21](#_Toc411536958)

[5.1 Exception Scenarios 21](#_Toc411536959)

[As on now there is no exceptions handled. 21](#_Toc411536960)

[5.2 Exception Handling 21](#_Toc411536961)

[Below is the Exception handling best practice used in the module 21](#_Toc411536962)

[5.3 Logging 21](#_Toc411536963)

[6. Installation and Configuration 22](#_Toc411536964)

[6.1 Installation steps of the multi cat addon 22](#_Toc411536965)

[6.2 Configuration steps for the addon 22](#_Toc411536966)

[6.2.1 Beans 22](#_Toc411536967)

[6.2.2 Security 22](#_Toc411536968)

[6.2.3 Css 23](#_Toc411536969)

[6.2.4 Layout 23](#_Toc411536970)

1. Introduction

Purpose

This document is the Technical Design Document which provides information of the hybris Multi cart module addon and steps to configure the module. It covers:

1. High Level Design
2. Application Architecture
3. Database Architecture
4. Installation Steps

Intended Audience

This document is aimed at Hybris developers and Technical members who want to implement the Multi cart addon in Hybris.

Document Scope

This document covers the details about the sprint 2 Multi cart addon.

Functional Rules

* The Multi Cart functionality should allow the user to create and manage multiple carts at the same time. The users may share their carts with their B2B unit users
* The current cart can be changed at any time through the order history page.
* The user can search on the order history page all his carts and also the carts belonging to his B2B unit users.
* The multi Cart Module helps the user to have access to all his B2B unit carts whether this cart is created by himself or by another user who gave access to his cart in order to manage parallel carts

Assumptions/Restrictions

This module has been created for version 5.4.0.0.

Display: There is a component up on the right side of the page that helps to switch from one cart to another (check-box select).

Cart, remove, modify, share actions logging as of now not implemented.

Documents Structure

|  |  |  |
| --- | --- | --- |
| Section | | Description |
| 1 | Introduction | Describes the purpose and intended audience of the document and explains its structure. |
| 2 | High Level Design | Overview of the module. |
| 3 | Detailed Application Design | Application architecture applicable to this module. |
| 4 | Detailed Database Design | Conceptual and logical architecture applicable to this module. |
| 5 | Exception Management & Logging | Information about the Exception handling and logging. |
| 6 | Installation & Configuration | Steps to install and configure the module. |
| 7 | Appendices | Supporting information. |

Additional Resources

|  |  |  |  |
| --- | --- | --- | --- |
| # | Description | URL | Purpose |
| 1 | Wiki Portal | <https://wiki.hybris.com> | To gather more information |
| 2 | Coconet | <https://coconet2.capgemini.com/sf/go/doc3184550?nav=1> | To refer functional specifications |

1. High Level Design Diagram

Overview

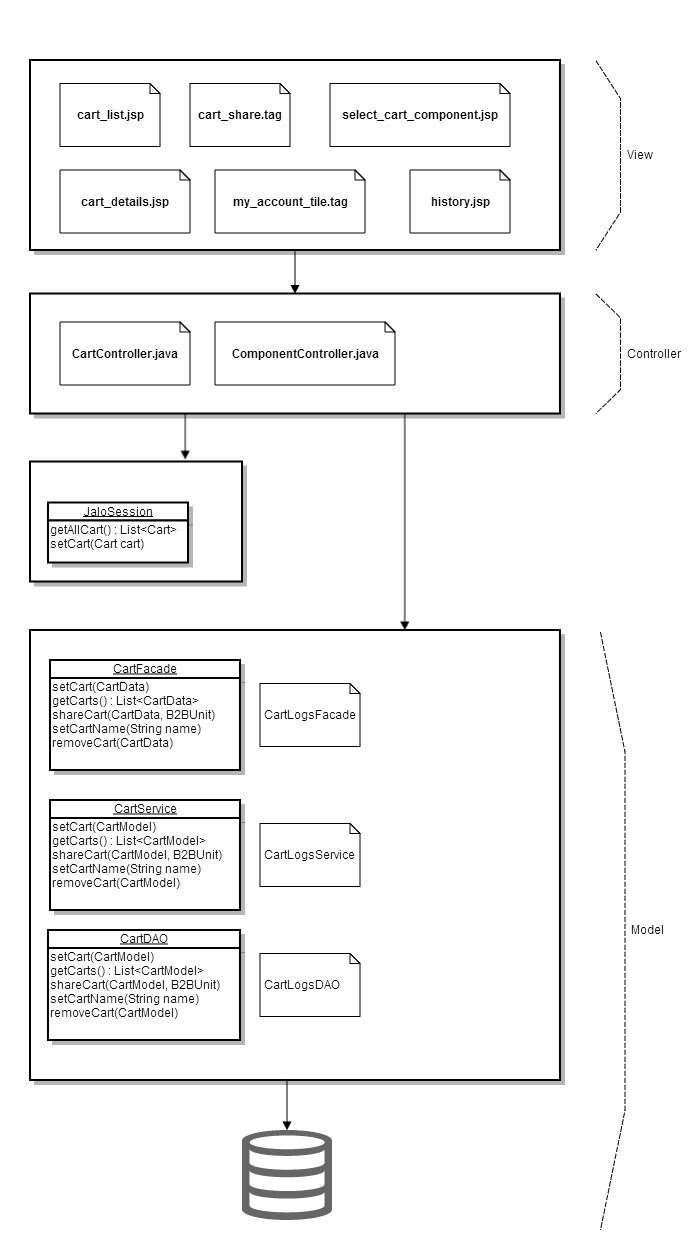


Fig: Multi cart MVC architecture

**Description:** MVC architecture is implemented for Multi cart. It is having JSPs and tags to display the multi cart information in the storefront. Cart Controller and Component controller is used for handling the multi cart while creating, deleting, sharing the cart etc.

Sequence diagrams

Create cart

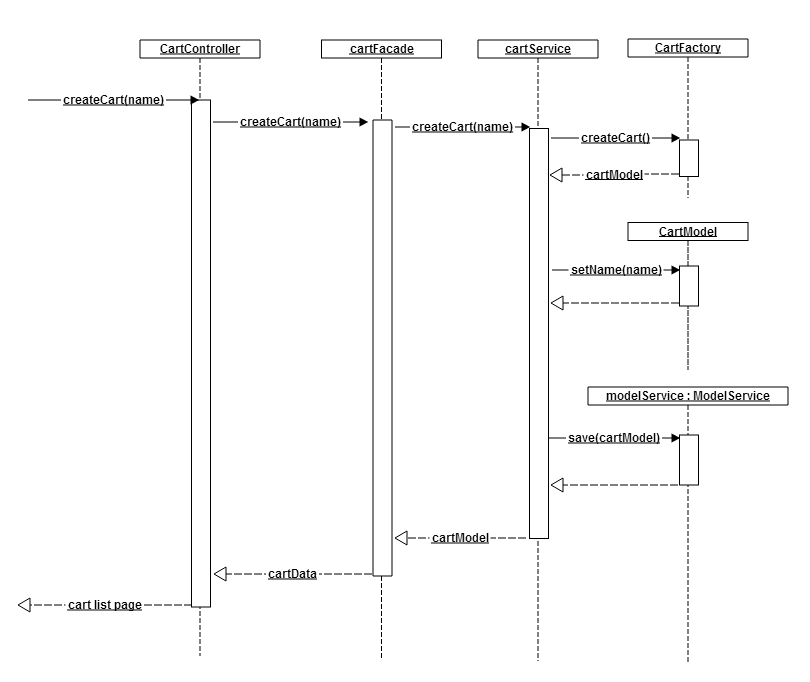


Fig: Create Cart sequence diagram

Description:

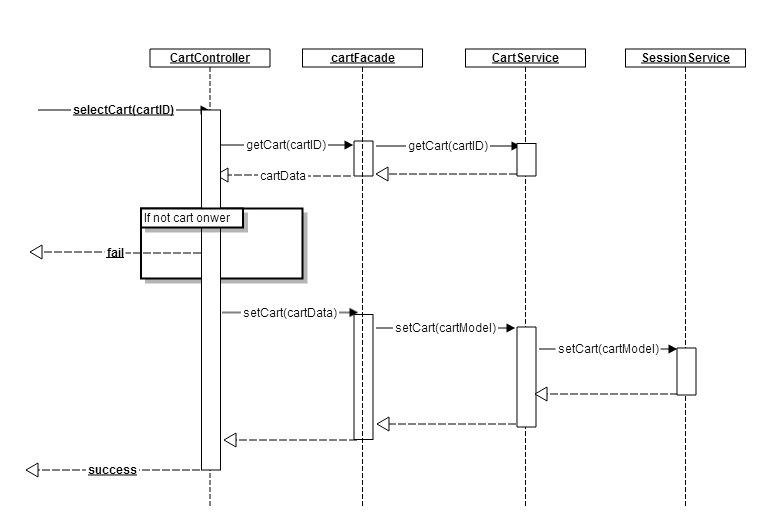
Cartcontroller: It is called when we want to create new cart and createCart() method will be called with name given by the user. This will call facades, again facade will call cartFactory and till get the carData from the facade then from the controller redirected to carListPage.

cartFacade: This called inside the createCart() method with the cart name. This facade uses the cartService.

cartFactory: This is used to create the cart. It will return the cartModel and the name is set to this model and it will be saved using ModelService.

CartService: Here cart service is not used it needs to update.

Select cart



Description:

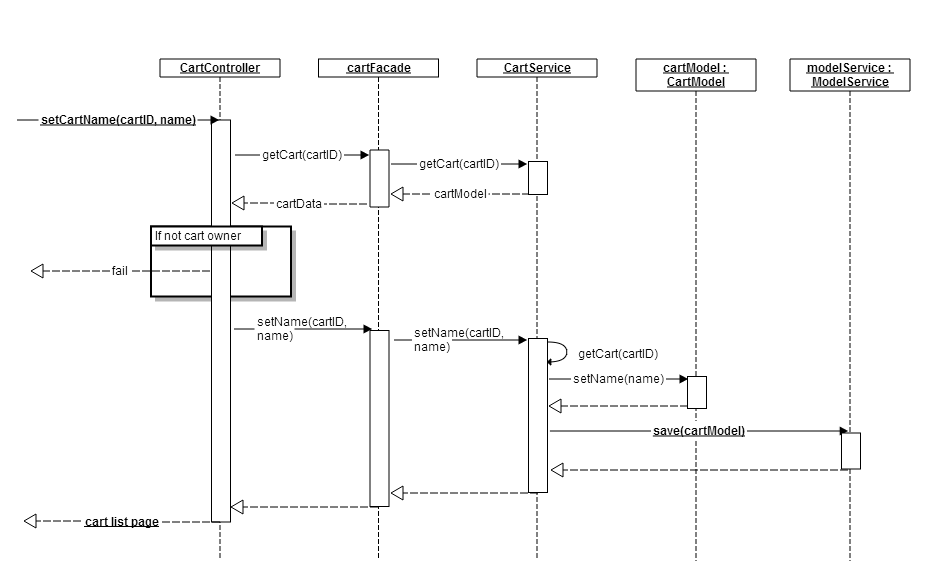
CartControlloer: This controller is used to select the cart by the login user. Once the user selects the cart controller will be called and selectCart() method will be called. If the login user is a owner then it will allow the user to select otherwise user will get failed to select the cart. This internally uses the facade and cartService and sessionService.

MulticartFacade: This will be used to select the cart by the user. Then it will use the cart service. This will uses again session service.

MultCartService: This service is used by the MulticartFacade to get the cart data. If the user is owner then user can selects the cart.

SessionService: It provides access to the current Session's. It is used select the cart based on the current session

Set cart name



Description:

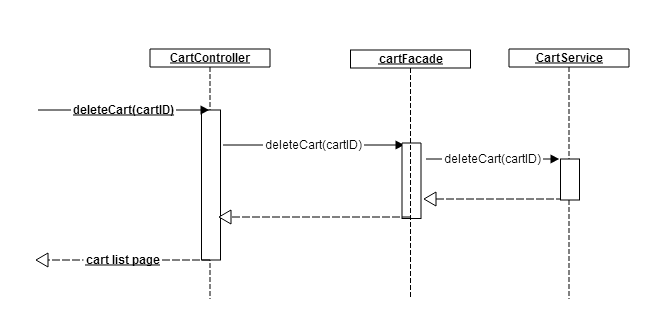
CartController: this cart controller is used to set the cart name by using setCartName() method. This will uses facades services, cartmodel and Model service to set the name of the cart and to save the cart. If the cart is already exists then it will get fail. Then it will redirect to the cart List page.

cartFaced: This called by the cartController to check cart is exist or not if not exits cart name will be set to the cart and will be saved.

Cartservice: It is provided by hybris it is used to set the cart anme.

ModelService: It is provided by the hybris and it is used to save the cart.

Delete cart

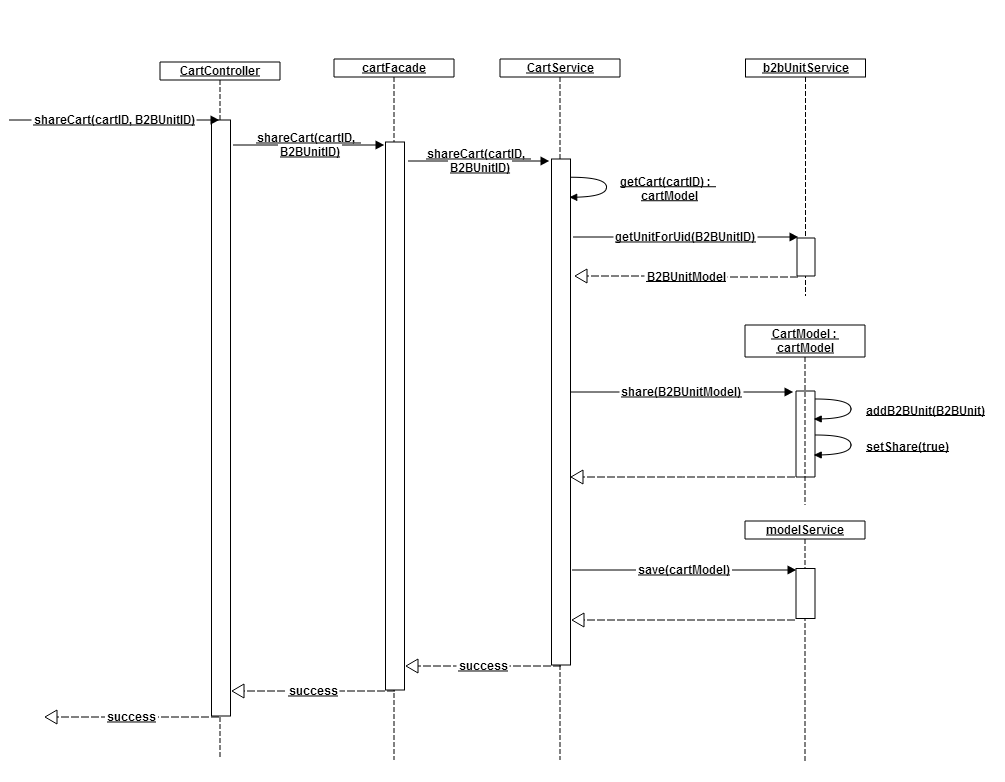


Description: CartController is used to remove/delete the cart using deleteCart(cartId) method. This uses cartFacde and CartService to remove the cart.

cartFacade: This will be called by the controller and this uses the MulticartService to remove/deletethe cart using deletecart() method using the cartId.

CartService: MulticartService is an interface it is used by facade to delete the cart.

Share cart



Description:

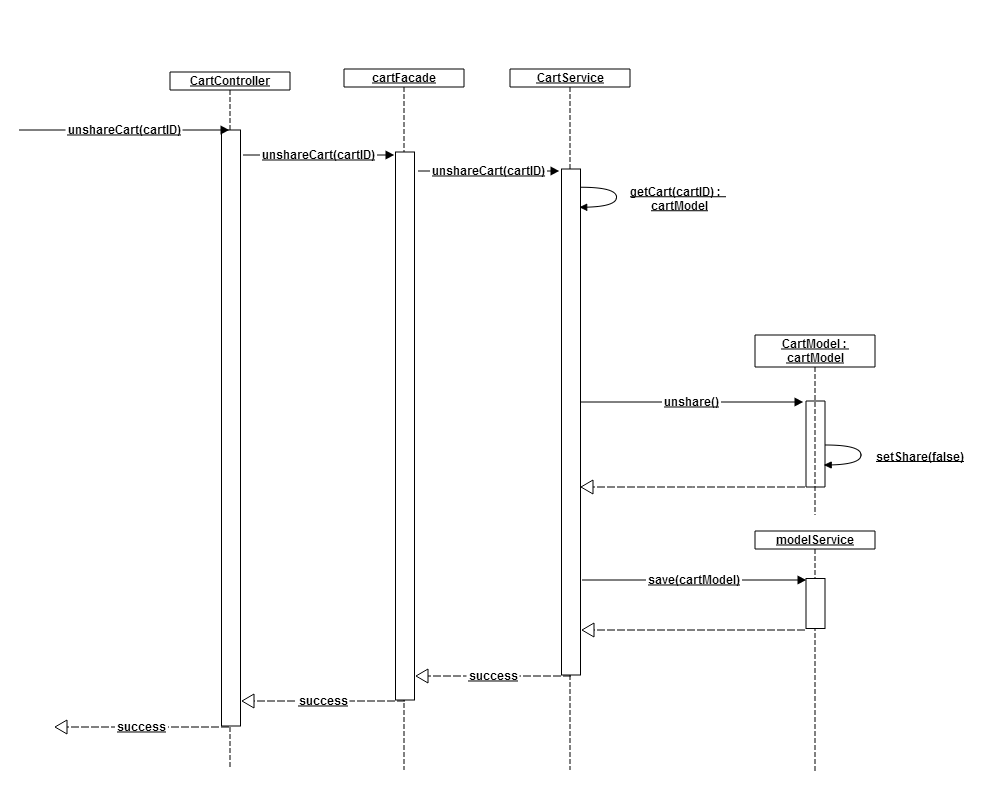
CartController: It is used here to share the cart using shareCart() method using cartId and B2BUnit as arguments. This will uses CartFacade, cartService, b2buinit service to share the cart.

CartFacade: This used to share the cart to a B2BUnit. This uses CartService and B2Bunit model to share the cart.

MulitCartService: This service is used to check the user can share the cart or not by using getcart(cartid) method, getUnitForUid(B2BUnitID). If the user belongs to the same B2B unit then user can share the cart.

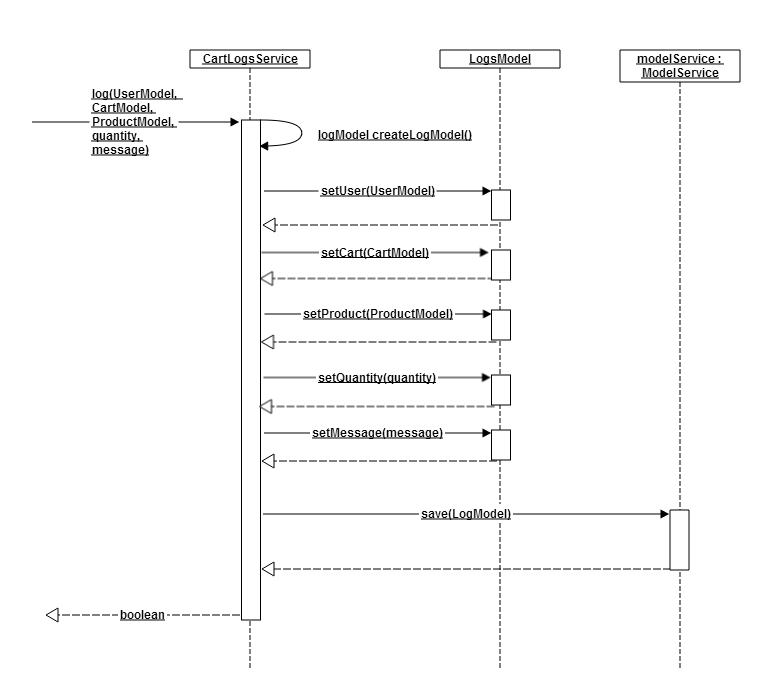
B2BUnitService: This used by the multi cart service to share the cart.

Unshare cart



Description: This sequence diagram explains the steps to unshare the cart. The user must be a owner then only the user can unshared the cart.

Log cart actions



Description: This not yet implemented this will be updated in feature.

Use Case Diagram.

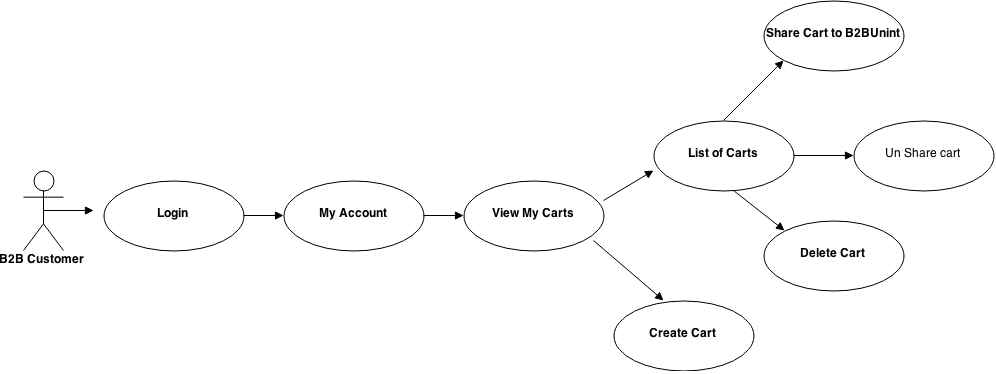
****

Fig: Multi cart use case diagram

**Description:** B2B customer has to login a store front to see his B2Bunit carts. After login user has to go to my account page. In this page there will be a link to see list of carts. Here customer can create the cart, share and un share the cart, delete the cart and can notify the created cart information to other B2B unit.

1. Detailed Application Design

Class diagram.

### https://lh3.googleusercontent.com/YY2gYzW_M7BfYNCrbmr2fme7JT1wEeXR1m7uMqjfKHhpDKiwgRLFjBs2C7xFgfx7T1ORWFIhxo-YwJX5jKlUDnD-8Dq4JNX9jFZYgBxQw8xD7QXG1L9-jqGwhwa1c5HRcg

Fig: Multi cart class diagram

Description: For this module spring MVC controller design patter is used. Some of the components and attributes are described below.

1. cartIownerAttributeHandler 🡪 To get owner of the cart this dynamic attribute is used.
2. cartNotificationAttributeHandler 🡪This is dynamic Boolean attribute to handle the cart notification. To get cart is having notification or not. If another user modifies the cart then notification symbol will be shown.
3. cartIsSharedAttributeHandeler 🡪 To know cart is shared or not this dynamic attribute is used.
4. cartFollowerAttributeHandler 🡪 To get cart follower this dynamic attribute is used.
5. SelectCartComponentRenderer 🡪 It used for rendering the cart component used in Multi cart module.
6. MulticartFacade 🡪 This is an interface and having method to get cart related information. It will uses defaultmultiCartsConverter, defaultCartLogConverter, multicartFactory, multicartservice.

Beans

In multi cart following beans are created.

1. CartLogData – To log the action like create a cart, share cart delete cart etc.
2. MultiCartsData—it extends cart data bean.
3. CartShareData – It is used for to get the information of shared cart if the cart is shared.
4. CartFollowerData – It contains the follower data of the multi cart data.

Services

MultiCartsService: This multi cart service extends the hybris provided cart service. It wills having the methods to log the cart actions like carte modified, share cart, remove cart etc.

Facades

MultiCartsFacade: This facade is used in CartController to decouple business logic implemented in multi cart.

Strategy

Mutil cart having tow strategies, MultiCartsEditStrategy, MultiCartsStrategy. These strategies are used while creating the cart, remove cart, share cart, is the user can edit the cart or not.

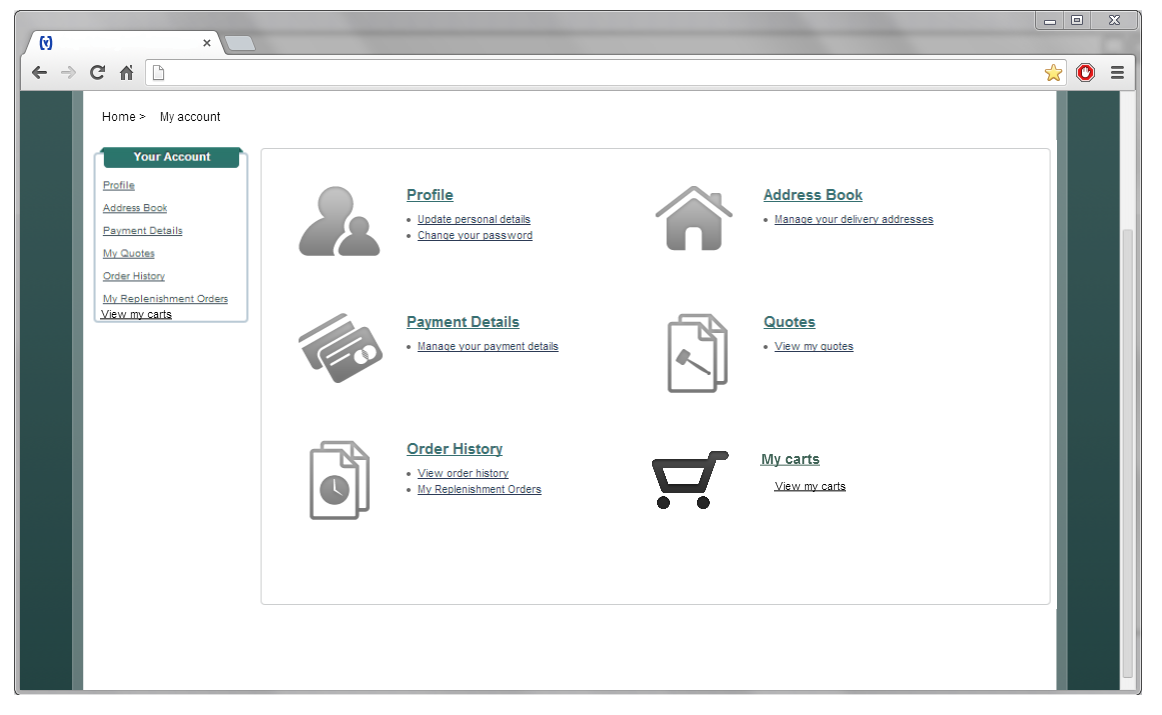
Controller

CartController : This controller extends AbstractAddOnPageController.

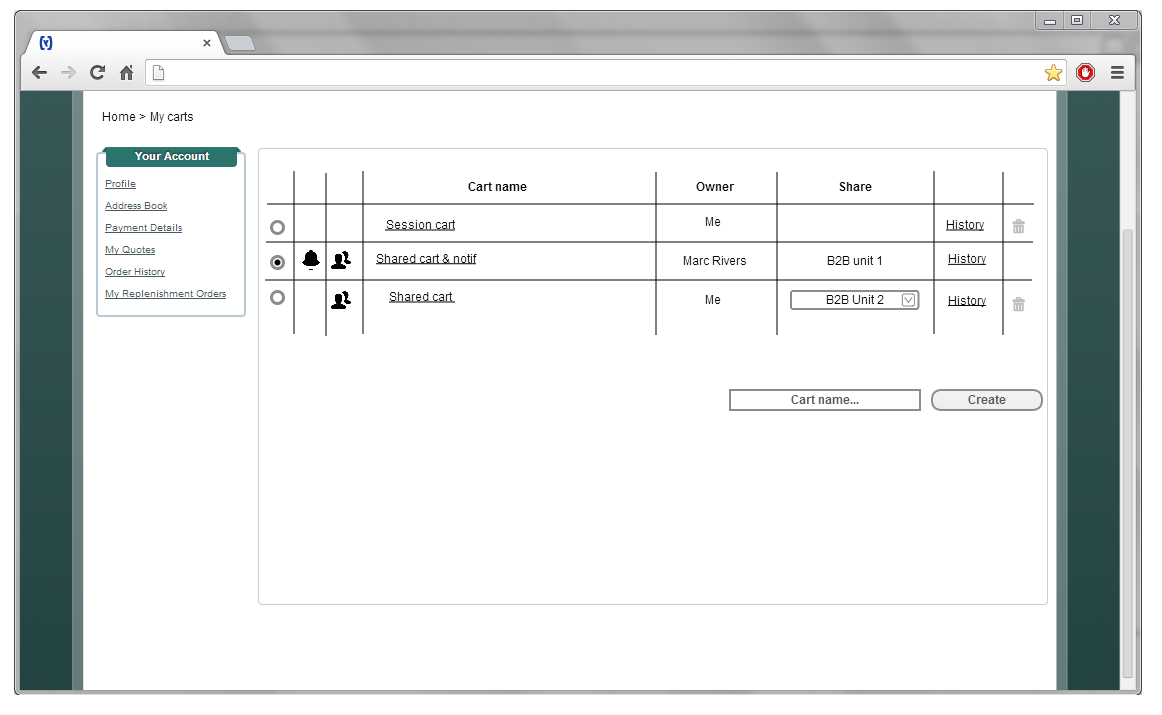
View/JSP

The mock-up of the Multi carts JSP pages are given below.

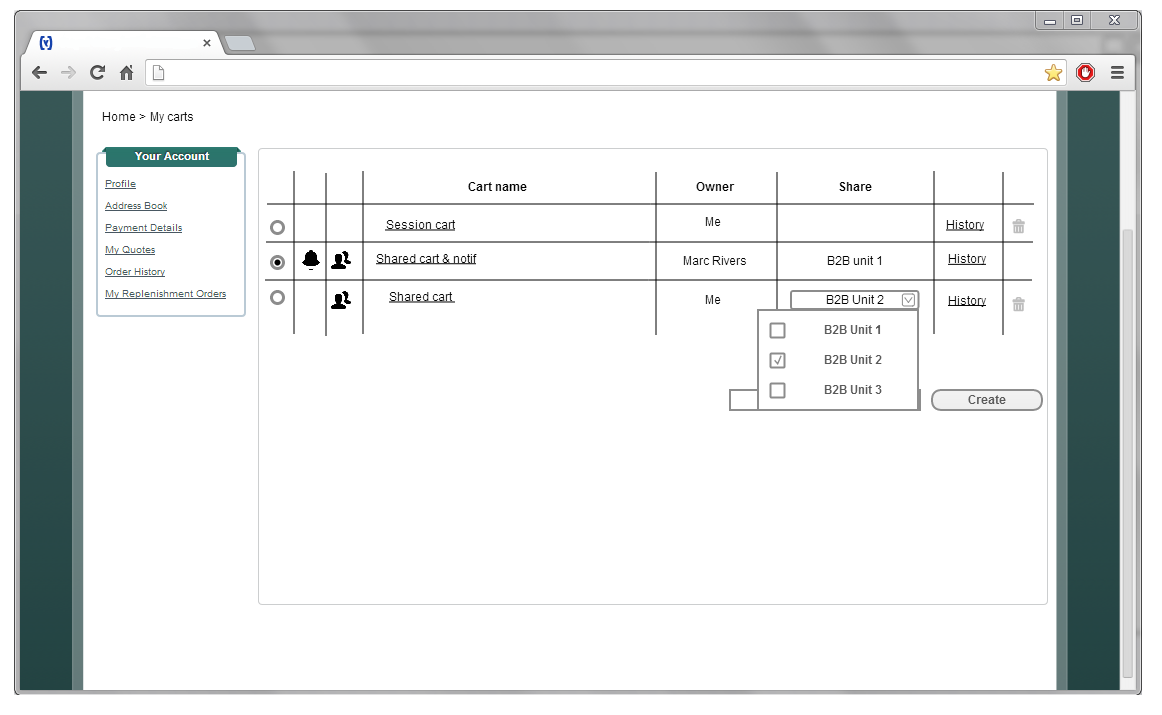
My account page.



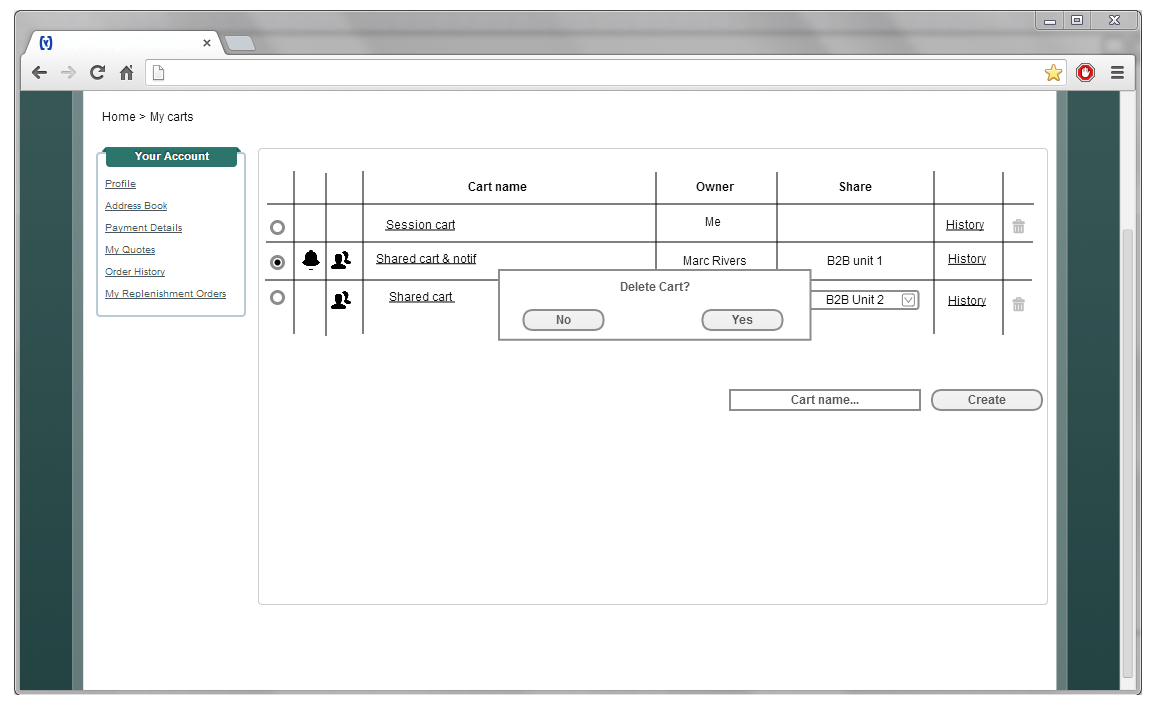
List of the carts



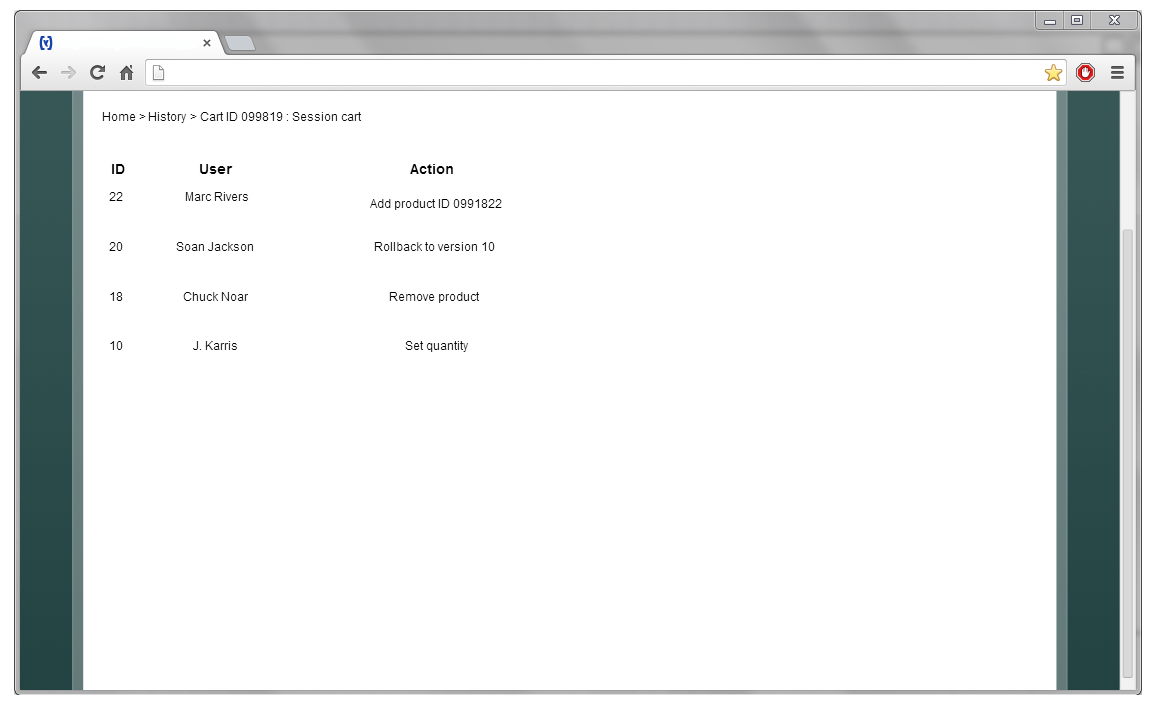
Share and select the cart



Delete the cart

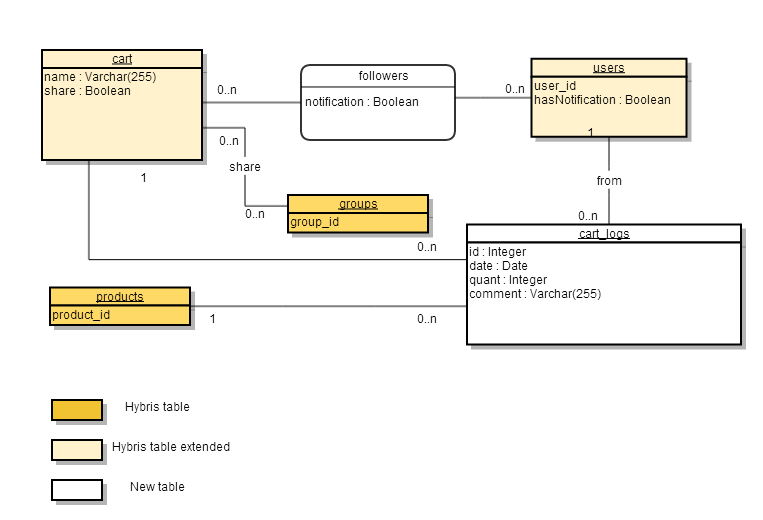


History of the Cart



1. Database architecture

ER Diagram



Description:

Cart\_logs and followers are the two tables created newly in multi cart.

Cart Item type it will be having dynamic attributes like is owner, is Shared etc. It can have 0 or many users And it can be shared to 0 or many b2bunit groups.

Users: users having relationship with cart. Users can multiple carts and cart can multiple followers.

Cart\_logs: This table is used to log the user actions like cart creation, remove, share and add product to cart.

1. Exception Handling and Logging

Exception Scenarios

As on now there is no exceptions handled.

Exception Handling

Below is the Exception handling best practice used in the module

* Never swallow the exception in catch block
* Declare the specific checked exceptions that your method can throw
* Do not catch the Exception class rather catch specific sub classes
* Never catch Throwable class
* Always correctly wrap the exceptions in custom exceptions so that stack trace is not lost
* Either log the exception or throw it but never do the both
* Never throw any exception from finally block
* Always catch only those exceptions that you can actually handle
* Don't use printStackTrace() statement or similar methods
* Use finally blocks instead of catch blocks if you are not going to handle exception
* Remember "Throw early catch late" principle
* Always clean up after handling the exception
* Throw only relevant exception from a method
* Never use exceptions for flow control in your program
* Validate user input to catch adverse conditions very early in request processing
* Always include all information about an exception in single log message
* Pass all relevant information to exceptions to make them informative as much as possible
* Always terminate the thread which it is interrupted
* Use template methods for repeated try-catch
* Document all exceptions in your application in javadoc

Logging

* For logging in the quick order module Log4j will be used.
* The logging will be set at INFO level.

1. Installation and Configuration

Installation steps of the multi cat addon

1. Add the multicart module dependency in your localextension.xml . It is available in hybris\config\localextensions.xml

<extension name=” multicarts />

1. Open the command and go to the folder hybris\bin\platform.
2. Execute the commands
   1. set antenv.bat
   2. ant addoninstall -Daddonnames="multicarts" - DaddonStorefront.yacceleratorstorefront="yb2bacceleratorstorefront"

WARNING: yb2bacceleratorstorefront must be replaced by your custom strorefront

Configuration steps for the addon

Beans

Add the following line in your spring-cms-config.xml. This file is available at hybris\bin\custom\b2bassets\b2bassetsstorefront\web\webroot\WEB-INF\config\spring-cms-config.xml.

This entry should be as shown below.

<bean id="acceleratorCMSComponentRendererRegistry" parent="defaultCMSComponentRendererRegistry">

<property name="renderers">

<map merge="true">

<entry key="CMSLinkComponent" value-ref="cmsLinkComponentRenderer"/>

**< entry key=” SelectCartComponent” value-ref=” selectCartComponentRenderer”/>**

</map>

</property>

</bean>

Security

Add the following lines in your **spring-security-config.xml**. This file is available at hybris\bin\custom\b2bassets\b2bassetsstorefront\web\webroot\WEB-INF\config\spring-security-config.xml

<!-- Multicarts Pages -->

<security:intercept-url pattern="/multicarts\*" access="ROLE\_B2BCUSTOMERGROUP" />

<security:intercept-url pattern="/multicarts/\*\*" access="ROLE\_B2BCUSTOMERGROUP" />

Css

Add the css in hybris\bin\custom\b2bassets\b2bassetsstorefront\web\webroot\\_ui\desktop\theme-green\css\theme-green.css file.

/\*\*\*\*\*\*\*\*\*\* Multicarts \*\*\*\*\*\*\*\*\*\*\*\*/

.right-components {

float: right;

}

.cart\_header {

float: none;

}

.multicarts-selectcartcomponent {

margin-left: 15px;

}

Layout

In the header.tag file change the component slot limit to 2

<cms:pageSlot position="MiniCart" var="cart" limit="2">

And add the div arround the component slot

<div class="right-components">

<cms:pageSlot position="MiniCart" var="cart" limit="2">

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

</cms:pageSlot>

</div>