

12/3/24, 13:54



Коммерция

Сгенерировано: 2024-12-03 13:54:41 GMT+0000

SAP Коммерция | 2205

Публичный

Оригинальное содержание:https://help.sap.com/docs/SAP_COMMERCE/9d346683b0084da2938be8a285c0c27a?locale=en-US&state=PRODUCTION&version=2205

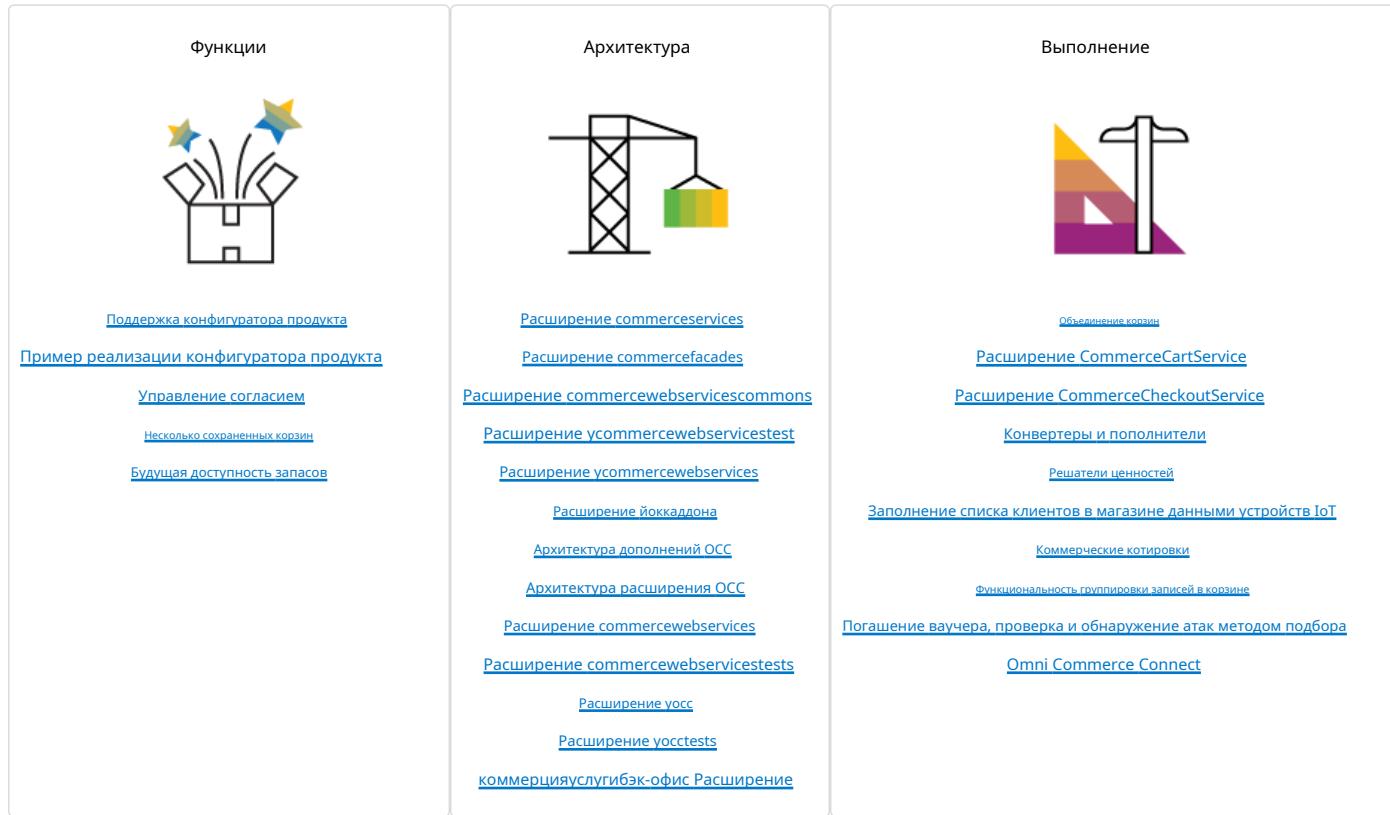
Предупреждение

Этот документ был создан на основе SAP Help Portal и является неполной версией официальной документации по продукту SAP. Информация, включенная в пользовательскую документацию, может не отражать расположение тем на SAP Help Portal и может не иметь важных аспектов и/или корреляций с другими темами. По этой причине он не предназначен для продуктивного использования.

Для получения более подробной информации посетите сайт<https://help.sap.com/docs/ отказ от ответственности>.

Модуль коммерческих услуг

Модуль «Коммерческие услуги» позволяет вам настраивать продукты, несколько корзин, а также управлять согласием клиентов и будущими запасами.



Особенности коммерческих услуг

Модуль Commerce Services позволяет вам настраивать продукты, несколько корзин и управлять согласием клиентов.

[Поддержка конфигуратора продукта](#)

SAP Commerce поддерживает различные настраиваемые продукты.

[Управление согласием](#)

B2C Accelerator предоставляет субъектам данных (физическим лицам, таким как клиент, контакт или учетная запись) функциональную возможность дать согласие на сбор или передачу их персональных данных.

[Несколько сохраненных корзин](#)

SAP Commerce Accelerator предоставляет функцию сохраненной корзины, которая позволяет пользователям сохранять одну или несколько корзин для дальнейшего использования.

[Будущая доступность запасов](#)

Функция «Будущая доступность запасов» указывает, когда запасы будут пополнены новыми запасами.

[Токен проверки](#)

Эта функция предоставляет API для управления кодом верификации токена. Клиент может запросить код верификации токена для определенной цели, и после проверки система доставляет его на указанный канал. Этот код может быть далее использован для проверки личности или транзакций, тем самым повышая безопасность системы.

Поддержка конфигуратора продукта

SAP Commerce поддерживает различные настраиваемые продукты.

SAP Commerce поддерживает ряд продуктов, настраиваемых с помощью различных конфигураторов:

- в том же каталоге
- в той же корзине в
- том же порядке

На странице оформления заказа конфигурация настраиваемого продукта проверяется, чтобы убедиться, что продукт был изменен правильно, и клиент может перейти к оформлению заказа.

Чтобы реализовать конфигуратор продукта, см.[пример реализации](#).

Информацию о конкретных конфигураторах см. здесь:

- [Конфигурация продукта с конфигурацией вариантов SAP и ценообразованием](#)
- [Интеграция SAP CPO для настраиваемых продуктов](#)

Пример реализации конфигуратора продукта

С функциональностью configurable products вы можете определить различные варианты для ваших продуктов. Узнайте, как распознать configurable product и возможные варианты конфигурации этого типа продукта.

Все настраиваемые продукты имеют [Конфигурировать](#) кнопка, которая перенаправляет клиента на страницу «Настроить параметры продукта». На странице «Настроить параметры продукта» вы можете настроить следующее:

- Гравированный текст
- Размер шрифта
- Тип шрифта

Пример настраиваемого продукта: камера POWERSHOT A480, красная.

Вы можете изменить конфигурацию товара, который уже находится в корзине, просто нажмите [Изменить конфигурацию](#) кнопку рядом с настраиваемым продуктом.

Если продукт настраиваемый, вы можете перейти на страницу Configure Product Options прямо со страницы Product Details. Щелкните страницу [Конфигурировать](#) кнопка на странице «Сведения о продукте»

HOME / OPEN CATALOGUE / CAMERAS / DIGITAL CAMERAS / DIGITAL COMPACTS

Applied Facets

Canon X

Shop by Stores

FIND STORES NEAR ME

Shop by Price

\$50-\$199.99 (4)

Shop by Megapixels

10 - 10.9 MP (4)

SORT BY: RELEVANCE

4 Products found

Image	Product Name	Description	Price
	POWERSHOT A480	PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, blue	\$99.85
	POWERSHOT A480	PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, black	\$111.84
	POWERSHOT A480	PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, silver	\$110.88
	POWERSHOT A480	PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, red	\$99.85

Шаги

Следуйте инструкциям, чтобы узнать, как распознать настраиваемый продукт и как настроить этот тип продукта.

Предпосылки

Установить B2C Accelerator. Информацию об установке см.[Локальная установка B2C и B2B Accelerator](#).

Процедура

- Перейдите к <https://electronics.local:9002/yacceleratorstorefront/electronics/en/>.

I'm looking for



BRANDS DIGITAL CAMERAS FILM CAMERAS HAND HELD CAMCORDERS POWER SUPPLIES FLASH MEMORY CAMERA ACCESSORIES & SUPPLIES



(0 ITEMS) \$0.00

DEFAULT

SAVE BIG

On select SLR & DSLR Cameras

SHOP NOW

SAVE BIG ON SELECT CAMERA ACCESSORIES & SUPPLIES

SHOP NOW

. Из Цифровые фотокамеры выпадающий список, выберите Компактные камеры.

I'm looking for

BRANDS **DIGITAL CAMERAS** FILM CAMERAS HAND HELD CAMCORDERS POWER SUPPLIES FLASH MEMORY CAMERA ACCESSORIES & SUPPLIES

(0 ITEMS) \$0.00

COMPACT CAMERAS

SLR CAMERAS

DEFAULT

SAVE BIG

On select SLR & DSLR Cameras

SHOP NOW

SAVE BIG ON SELECT CAMERA ACCESSORIES & SUPPLIES

SHOP NOW

DEFAULT

DEFAULT

DEFAULT

DEFAULT

. В меню «Покупка по бренду» выберите Канон.

B2C Accelerator

I'm looking for

WELCOME JUSTYNA MY ACCOUNT SIGN OUT

BRANDS DIGITAL CAMERAS FILM CAMERAS HAND HELD CAMCORDERS POWER SUPPLIES FLASH MEMORY CAMERA ACCESSORIES & SUPPLIES

(0 ITEMS) \$0.00

HOME / OPEN CATALOGUE / CAMERAS / DIGITAL CAMERAS / DIGITAL COMPACTS

Shop by Stores

or

SORT BY: RELEVANCE

47 Products found

DSC-H20 BLUE
DSC-H20, Blue, 10.1 Megapixels, 10x Optical Zoom, 3.0" LCD \$558.40

EASYSHARE Z730 ZOOM DIGITAL CAMERA
EASYSHARE Z730 Zoom Digital Camera \$147.04

DIGITAL CAMERA EASYSHARE C875
EASYSHARE C875 Zoom Digital Camera, 8 MP, 2.5" LCD \$227.24

. Перейдите к камере POWERSHOT A480 и нажмите кнопку **Конфигурировать** кнопка.

HOME / OPEN CATALOGUE / CAMERAS / DIGITAL CAMERAS / DIGITAL COMPACTS

Applied Facets
Canon

Shop by Stores

or

SORT BY: RELEVANCE

4 Products found

POWERSHOT A480
PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, blue \$99.85

POWERSHOT A480
PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, black \$111.84

POWERSHOT A480
PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, silver \$110.88

POWERSHOT A480
PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, red \$99.85

Откроется страница настройки параметров продукта.

. Укажите конфигурацию продукта:

- Гравированный текст
- Размер шрифта
- Тип шрифта

B2C Accelerator

I'm looking for

WELCOME JUSTYNA MY ACCOUNT SIGN OUT

BRANDS DIGITAL CAMERAS FILM CAMERAS HAND HELD CAMCORDERS POWER SUPPLIES FLASH MEMORY CAMERA ACCESSORIES & SUPPLIES

HOME / OPEN CATALOGUE / CAMERAS / DIGITAL CAMERAS / DIGITAL COMPACTS / POWERSHOT A480

ENGRAVED TEXT
This camera belongs to ME

FONT SIZE
20

FONT TYPE
Arial Bold

ADD TO CART

Accelerator
About Commerce Accelerator
FAQ

Hybris
About hybris
Contact Us

Follow Us
Agile Commerce Blog
LinkedIn
Facebook
Twitter

ENGLISH \$ USD

© 2016 hybris software

. Нажмите на Кнопка.

Ваш продукт добавлен в корзину. И вы перенаправлены в корзину.

. Вы можете продолжить покупки или перейти к оформлению заказа. Для получения дополнительной информации о оформлении заказа см.[Управление стратегиями многошаговой оплаты](#) документ.

Настраиваемый продукт в корзине

Контекст

Вы можете изменить конфигурацию товара, который уже находится в корзине.

Процедура

. Перейдите в корзину.

. Нажмите на кнопка.

Cart | ID: 00000002

1 item | \$99.85

EXPORT CSV

ITEM (STYLE NUMBER)	PRICE	QTY	DELIVERY	TOTAL
PowerShot A480 1934793 In Stock  Engraved Text: Font Size: Font Type:	\$99.85	<input type="button" value="1"/>	SHIP	\$99.85 

EXPORT CSV

COUPON CODE

Subtotal:

\$99.85

ORDER TOTAL

\$99.85

Your order includes \$4.75 tax.

Вы будете перенаправлены на страницу настройки параметров продукта.

. Обновите свой продукт и нажмите кнопку кнопка для обновления продукта.

. Вы можете продолжить покупки или перейти к оформлению заказа. Для получения дополнительной информации о оформлении заказа см.[Управление стратегиями многошаговой оплаты](#) документ.

Страница сведений о продукте

Вы можете перейти к настройке параметров продукта непосредственно со страницы «Сведения о продукте», если продукт настраивается. Для этого нажмите кнопку

Кнопка.

I'm looking for



BRANDS DIGITAL CAMERAS FILM CAMERAS HAND HELD CAMCORDERS POWER SUPPLIES FLASH MEMORY CAMERA ACCESSORIES & SUPPLIES

HOME / OPEN CATALOGUE / CAMERAS / DIGITAL CAMERAS / DIGITAL COMPACTS / POWERSHOT A480

PowerShot A480 | ID 1934793

★ ★ ★ ★ (2) | Show Reviews | Write a Review



\$99.85

PowerShot A480 - 10.0 MP, 3.3x optical, DIGIC III, 2.5" LCD, red

-

1

+

60 In Stock

CONFIGURE



ADD TO CART



PICK UP IN STORE

Share

Сопутствующая информация

[texteldcongruatortemplateservices Расширение](#)[texteldcongruatortemplatebackoffice Расширение](#)[texteldcongruatortemplatefacades Расширение](#)[texteldcongruatortemplateaddon Extension](#)[Поддержка конфигуратора продуктов](#)

Управление согласием

B2C Accelerator предоставляет субъектам данных (физическими лицами, таким как клиент, контакт или учетная запись) функциональную возможность дать согласие на сбор или передачу их персональных данных.

Введение

Пользователи дают согласие на сбор и использование своих данных в определенных точках входа витрины. По умолчанию страницы, на которых пользователи создают учетную запись, служат точками входа, но могут быть созданы и другие точки входа. Шаблоны согласия используются для отображения текста согласия в точках входа, а также на странице управления согласием, где пользователи могут изменять настройки своего согласия. Любые изменения, внесенные в настройки согласия, регистрируются в бэкенде и могут быть просмотрены в Backoffice Administration Cockpit.

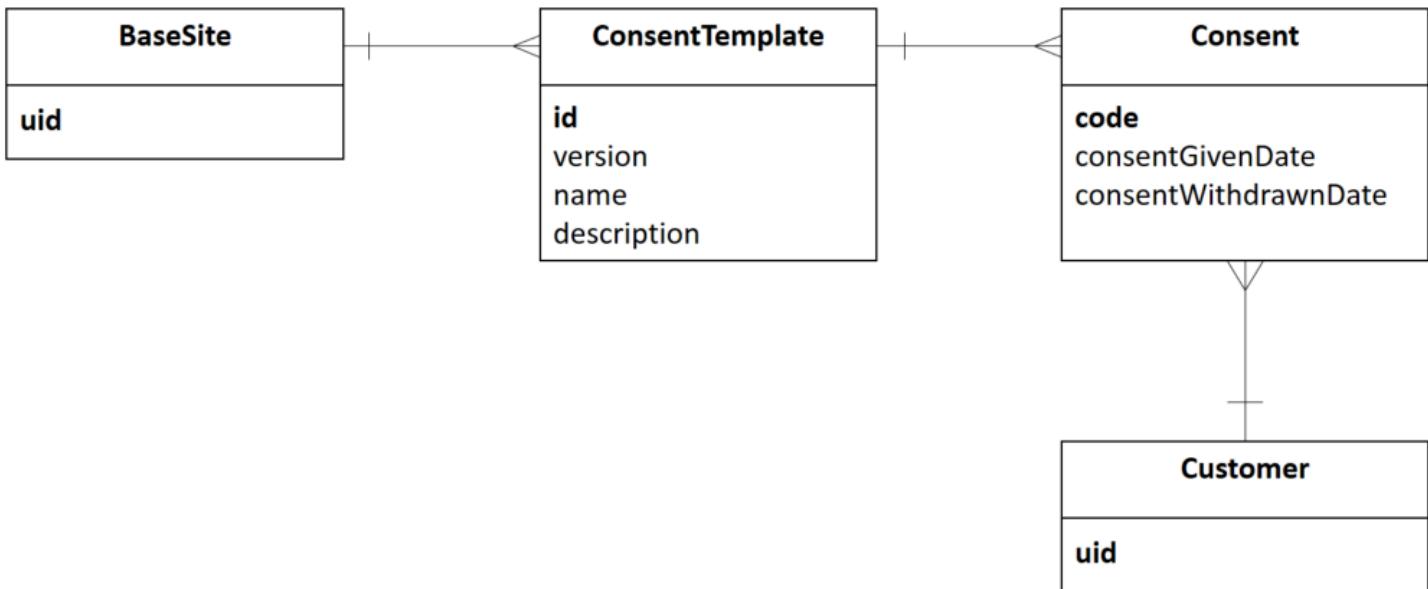
-Примечание

Хотя Commerce B2C Accelerator включает шаблоны, которые можно использовать для управления согласием, вы можете вручную реализовать шаблоны согласия для Commerce B2B Accelerator. Для получения дополнительной информации о том, как добавить функцию согласия в витрину B2C или B2B, см. один из следующих разделов:

- Витрина магазина B2C:[Шаблоны согласия](#)
- Витрина B2B:[Добавление точек входа согласия](#)

Модель данных

На этой диаграмме показана модель данных для управления согласием, соответствующая решению SAP Central Consent Management:



Каждый Базовый Сайт (например, электроника) может иметь несколько Шаблонов согласия. Шаблон согласия имеет следующие свойства:

- идентификатор
- версия
- имя
- описание
- baseSite

По умолчанию в образце данных доступен один шаблон: МАРКЕТИНГОВАЯ_РАССЫЛКА. Вот пример свойств для шаблона по умолчанию в витрине магазина электроники (обратите внимание, что в качестве заголовков столбцов используются свойства Java класса модели, а не фактические заголовки столбцов базы данных):

ПК	идентификатор	версия	имя	описание	baseSite
ПК1	МАРКЕТИНГОВАЯ_БЮЛЛЕТЕНЬ	0	Согласие на подписку на рассылку новостей	Я разрешаю использовать мои персональные данные для получения электронных рассылок в целях маркетинговых кампаний.	электроника

Когда пользователь дает согласие, в бэкенде создается запись о согласии. Запись фиксирует как бизнес-событие, так и соответствующую информацию о времени. Вот пример двух записей о согласии. Первая запись показывает, что пользователь дал согласие, а затем отозвал его позднее; вторая запись показывает, что тот же пользователь дал согласие и не отозвал его.

код	UID клиента	Первичный ключ ConsentTemplate	согласиеДаноДата	согласиеОтозваноДата
00000	клиент1	ПК1	2017-06-01 9:00:23.656	2017-06-20 23:12:56.412
00001	клиент1	ПК1	2017-07-03 11:45:33.552	нулевой

Поскольку записи о согласии в базе данных хранят как бизнес-событие, так и время, их можно использовать в качестве записи аудиторского следа.

Дать согласие

Пользователи Storefront могут дать согласие при регистрации учетной записи. В любое время они могут изменить настройки своего согласия на странице управления согласием.

Журналы согласия

Когда пользователь дает или отзывает согласие, в бэкенде создается запись о согласии, которую можно просмотреть в панели администрирования бэк-офиса.

Шаблоны согласия

Шаблоны согласия определяют контент, отображаемый пользователям на витрине. В витрине может быть несколько шаблонов согласия, и один шаблон может использоваться для нескольких витрин.

Версии шаблона согласия

Когда условия и положения шаблона согласия меняются, можно создать новую версию шаблона согласия, чтобы отразить эти изменения. Пользователи витрины должны затем дать явное согласие на новую версию шаблона.

Добавление точек входа согласия

Пользователи дают согласие в точках входа, таких как страница регистрации клиентов. Узнайте, как добавлять точки входа в другие области витрины и в другие витрины, такие как электронные инструменты.

Анонимное управление согласием

Anonymous Consent Management дает анонимным пользователям контроль над отслеживанием их данных. Анонимные пользователи могут давать или отклонять свое согласие для приложений, которые собирают и обрабатывают персональные данные.

Удаление персональных данных

Клиенты могут закрыть свою учетную запись в магазине и удалить свои персональные данные. Задания Cron включены в `acceleratorcore` расширение очищает данные, связанные с деактивированными клиентами.

Дать согласие

Пользователи Storefront могут дать согласие при регистрации учетной записи. В любое время они могут изменить настройки своего согласия на странице управления согласием.

По умолчанию на витринах B2C имеется шаблон согласия, который запрашивает у пользователей согласие на использование их персональных данных для получения маркетинговых рассылок.

Предоставление согласия при регистрации учетной записи

Когда пользователи создают учетную запись в витрине, у них есть возможность дать согласие на шаблон согласия по умолчанию. Есть две области, где пользователи могут создать учетную запись:

- На странице входа/регистрации
- На странице подтверждения заказа после оформления заказа в качестве гостя

В обоих случаях появляется флагок, с помощью которого пользователи могут дать свое согласие на получение маркетинговых рассылок.

-Примечание

По умолчанию, страницы создания учетной записи являются единственными, где отображается флагок согласия. Чтобы узнать, как добавить точки входа согласия в других областях витрины, см. [Добавление точек входа согласия](#).

Страница управления согласием

Зарегистрированные пользователи могут в любое время изменить настройки своего согласия, перейдя по ссылке [Управление согласием на мой аккаунт](#). Страница [Управления согласием](#) отображает все шаблоны согласия для витрины, а также переключатель, который пользователи могут включать и выключать для каждого шаблона. Любые изменения настроек согласия на этой странице регистрируются и отображаются в Backoffice Administration Cockpit. Обратите внимание, что любые шаблоны согласия, созданные с помощью Backoffice, также отображаются на этой странице.

Родительская тема:[Управление согласием](#)

Сопутствующая информация

[Журналы согласия](#)

[Шаблоны согласия](#)

[Версии шаблонов согласия](#)

[Добавление точек входа согласия](#)

[Управление анонимным согласием](#)

[Удаление персональных данных](#)

Журналы согласия

Когда пользователь дает или отзывает согласие, в бэкэнде создается запись о согласии, которую можно просмотреть в панели администрирования бэк-офиса.

Записи согласия фиксируют как бизнес-событие, так и соответствующую временную метку. Поскольку записи согласия в базе данных хранят как бизнес-событие, так и время, вы можете использовать их в качестве записи аудиторского следа, если это необходимо.

Просмотр записей о согласии

Чтобы просмотреть записи согласия в Backoffice, перейдите к [Согласия](#) пользователя. Если записи о согласии не отображаются, нажмите [Поиск](#) для отображения всех записей согласия. Чтобы уточнить поиск, нажмите [Переключиться в режим расширенного поиска](#) и введите требуемые критерии поиска. Используйте расширенный поиск для просмотра всех записей согласия для определенного пользователя или шаблона согласия.



Записи о согласии отображают следующую информацию:

- Код записи согласия
- Клиент
- Дата, когда было дано согласие
- Дата отзыва согласия (отображается [Нулевой](#), если согласие не было отозвано)
- Шаблон согласия

Экспорт записей о согласии

При необходимости вы можете экспортить записи согласия в файл CSV. Для этого найдите записи, которые вы хотите экспортить, щелкните значок Экспорт списка в CSV (мастер).



, а затем завершите

Родительская тема:[Управление согласием](#)

Сопутствующая информация

[Дать согласие](#)[Шаблоны согласия](#)[Версии шаблонов согласия](#)[Добавление точек входа согласия](#)[Управление анонимным согласием](#)[Удаление персональных данных](#)

Шаблоны согласия

Шаблоны согласия определяют контент, отображаемый пользователям на витрине. В витрине может быть несколько шаблонов согласия, и один шаблон может использоваться для нескольких витрин.

Шаблоны согласия по умолчанию

B2C Accelerator включает один шаблон согласия по умолчанию (MARKETING_NEWSLETTER), который запрашивает у пользователей согласие на получение маркетинговых информационных бюллетеней. Этот шаблон доступен во всех витринах B2C (electronics, apparel-uk и apparel-de).

Вы можете просматривать и редактировать шаблоны согласия в Backoffice, перейдя по ссылке [Шаблон согласия пользователя](#).



Добавление шаблонов согласия

Вы можете добавлять новые шаблоны на витрину с помощью Backoffice:

. В Backoffice перейдите к [Шаблон согласия пользователя](#). Нажмите

значок плюса (+), чтобы открыть форму «Создать новое согласие».

. Введите шаблон [ИДЕНТИФИКАТОР](#), выберите [Сайт](#) где будет использоваться шаблон, а затем введите [Версия](#) шаблона. Нажмите

[Следующий](#).

. Введите шаблон [Имя](#). Имя отображается на странице управления согласием. .

Введите шаблон [Описание](#), затем нажмите [Сделанный](#).

На странице управления согласием отображается имя вновь созданного шаблона.

-Примечание

Создание шаблона согласия в Backoffice не создает точку входа для пользователя, чтобы дать согласие; шаблон отображается только на странице Consent Management. Чтобы создать новую точку входа для шаблона согласия, см. [Добавление точек входа согласия](#).

Родительская тема:[Управление согласием](#)

Сопутствующая информация

[Дать согласие](#)[Журналы согласия](#)[Версии шаблона согласия](#)[Добавление точек входа согласия](#)[Анонимное управление согласием](#)[Удаление персональных данных](#)

Версии шаблона согласия

Когда условия и положения шаблона согласия меняются, можно создать новую версию шаблона согласия, чтобы отразить эти изменения. Пользователи витрины должны затем дать явное согласие на новую версию шаблона.

Например, если есть шаблон согласия для маркетинговой рассылки, которая консультирует клиентов о популярных продуктах, вы можете добавить в эту рассылку рекламные акции, которые являются специфичными для клиента на основе его истории покупок. В этом случае вы можете создать новый шаблон с обновленным номером версии, на который клиент должен дать согласие в витрине магазина.

-Примечание

Чтобы создать новую версию шаблона, необходимо создать новый шаблон, чтобы клиенты могли дать свое явное согласие на измененные положения и условия. Изменение номера версии для существующего шаблона не отзывает согласие клиента на старую версию.

При создании новой версии шаблона согласия:

- новая версия шаблона заменяет старую версию на витрине.
- согласие клиента на старую версию не отзывается.
- согласие клиента в новой версии не дается; клиенты должны дать свое согласие на новую версию на странице управления согласием.

В Backoffice созданы новые версии шаблонов согласия:

. Войдите в Backoffice и перейдите в [Шаблоны согласия пользователя](#).



Нажмите значок плюса (+), чтобы открыть форму «Создать новое согласие».

. Введите **шаблонИДЕНТИФИКАТОР**. Этот идентификатор должен совпадать с версией, которую вы обновляете.

. Введите **Сайт** где этот шаблон появляется. Сайт должен быть таким же, как версия, которую вы обновляете. . Введите

шаблон**Версия**число. Это число должно увеличиться на единицу по сравнению со старой версией. . Нажмите

[Следующий](#).

. Введите шаблон**Имя**и**Описание**. Они должны отличаться от предыдущей версии, чтобы отразить изменения в положениях и условиях. . Нажмите

[Сделанный](#).

Новая версия шаблона заменяет старую версию на странице управления согласием в витрине магазина.

-Примечание

Шаблоны согласия не должны удаляться. Версии согласия хранятся в системе для подтверждения изменений в заявлениях о согласии. Если заявление о согласии больше недействительно, необходимо отменить согласие. Права на удаление шаблонов согласия должны быть отозваны у пользователей.

Родительская тема:[Управление согласием](#)

Сопутствующая информация

[Добавление точек входа согласия](#)

[Журналы согласия](#)

[Шаблоны согласия](#)

[Добавление точек входа согласия](#)

[Анонимное управление согласием](#)

[Удаление персональных данных](#)

Добавление точек входа согласия

Пользователи дают согласие в точках входа, таких как страница регистрации клиентов. Узнайте, как добавлять точки входа в другие области витрины и в другие витрины, такие как электроинструменты.

В следующем примере показано, как добавить точку входа для согласия на страницу «Изменить единицы измерения», доступ к которой можно получить, перейдя по ссылке [Подразделения моей компании](#) в витрине магазина электроинструментов. Мы создадим шаблон согласия на маркетинговую кампанию для этой точки входа.

Создание собственной точки входа для получения согласия включает в себя следующие основные шаги:

- Настройка шаблона согласия
- Обновление контроллеров
- Адаптация формы интерфейса

Настройка шаблона согласия

. В электроинструменты расширение, добавьте новый файл согласия impex вместе с локализациями impex, сгенерированными во время сборки. Для получения дополнительной информации о локализациях, сгенерированных во время сборки, см. [Локализация ImpEx с использованием генерации времени сборки](#).

◦ `powertoolsstore/resources/powertoolsstore/import/sampledata/stores/powertools/consents.impex`

```
# -----
# [y] Платформа hybris
#
# Copyright (c) 2017 SAP SE или аффилированная компания SAP. Все права защищены.
#
# Это программное обеспечение является конфиденциальной и частной информацией SAP.
# («Конфиденциальная информация»). Вы не должны разглашать такую Конфиденциальную информацию.
# Информация и использовать ее только в соответствии с условиями
# лицензионное соглашение, которое вы заключили с SAP.
#
# -----
# ImpEx для согласия на магазин электроники
# % impex.setLocale( Locale.GERMAN );
```

```
$siteUid=powertools
INSERT_UPDATE ConsentTemplate;id[unique=true];name;description;version[unique=true];baseSite(uid)[unique=true,default=$site ;BUSINESSUNIT_MARKETING;"Получать
электронные письма для маркетинговых кампаний бизнес-подразделения";"Я разрешаю использовать мои персональные данные для получения
```

- powertoolsstore/resources-lang/powertoolsstore/import/sampleddata/stores/powertools/consents.vt

```
# -----
# [у] Платформа hybris
#
# Copyright (c) 2017 SAP SE или аффилированная компания SAP. Все права защищены.
#
# Это программное обеспечение является конфиденциальной и частной информацией SAP.
# («Конфиденциальная информация»). Вы не должны разглашать такую Конфиденциальную информацию.
# Информация и использовать ее только в соответствии с условиями
# лицензионное соглашение, которое вы заключили с SAP.
#
#
# ImpEx для согласия на магазин электроники
#
#
# Язык
\$lang=$lang.toLowerCase()

\$siteUid=powertools

# установить( $consents = $query.load('consents') )
UPDATE ConsentTemplate;id[unique=true];name[lang=\$lang];description[lang=\$lang];version[unique=true];baseSite(uid)[уникальный
# foreach( $consent in $consents ) ;
$consent.key;"$consent.values.name","$consent.values.description",0;;
# конец
```

- powertoolsstore/resources-lang/powertoolsstore/import/sampleddata/stores/powertools/consents_en.properties

```
# -----
# [у] Платформа hybris
#
# Copyright (c) 2017 SAP SE или аффилированная компания SAP. Все права защищены.
#
# Это программное обеспечение является конфиденциальной и частной информацией SAP.
# («Конфиденциальная информация»). Вы не должны разглашать такую Конфиденциальную информацию.
# Информация и использовать ее только в соответствии с условиями
# лицензионное соглашение, которое вы заключили с SAP.
# -----
# Consent.BUSINESSUNIT_MARKETING.name=Получать электронные письма для
# маркетинговых кампаний бизнес-подразделения Consent.BUSINESSUNIT_MARKETING.description=Я разрешаю использовать мои персональные данные для получения
# электронных информационных бюллетеней для бизнеса
```

. Добавьте новое свойство, обозначающее идентификатор шаблона согласия, который будет использоваться на странице блока B2B в свойства проекта вэлектроинструментырасширение:

```
businessunit.consent.id.powertools=BUSINESSUNIT_MARKETING
```

Обновление контроллеров

. В коммерцияorgaddonрасширение, изменениеMyCompanyPageController.java для добавления константы для имени свойства идентификатора шаблона согласия. Этот файл Java находится в commerceorgaddon/acceleratoraddon/web/src/de/hybris/platform/commerceorgaddon/controllers/pages.

```
защищенная статическая конечная строка BUSINESSUNIT_CONSENT_ID = "businessunit.consent.id";
```

. Добавьте новый метод, который настраивает данные шаблона согласия в модели страницы:

```
защищенная void addBusinessUnitConsentDataToModel(окончательная модель model) {
    окончательная строка agreeId = getSiteConfigService().getProperty(BUSINESSUNIT_CONSENT_ID); если
    (StringUtil.isNotBlank(consentId))
    {
        окончательный ConsentTemplateData agreeData = getConsentFacade().getLatestConsentTemplate(consentId);
        model.addAttribute("consentTemplateData", agreeData);
    }
}
```

. В BusinessUnitManagementPageController.java, добавить звонок добавитьBusinessUnitConsentDataToModeleditUnit (GET) метод непосредственно перед оператором return. Этот файл Java находится в commerceorgaddon/acceleratoraddon/web/src/de/hybris/platform/commerceorgaddon/controllers/pages.

```
@RequestMapping(значение = "/edit", метод = RequestMethod.GET)
@RequireHardLogIn
public String editUnit(@RequestParam("unit") final String unit, final Model model) выдает CMSItemNotFoundException {

    ...
    добавитьДанныеСогласияБизнесУнитаВМодель(модель);
    вернуть ControllerConstants.Views.Pages.MyCompany.MyCompanyManageUnitEditPage;
}
```

. editUnit (Метод POST) добавьте блок кода для обработки даватьСогласиеиотозватьСогласиеЛогика, сразу после призыва b2bUnitFacade.updateOrCreateBusinessUnit(...). Также добавьте звонок в добавитьДанныеСогласияБизнесУнитаВМодель(...) в блоке catch. Это настраивает модель страницы в случае возврата на страницу редактирования после ошибки:

```
@RequestMapping(значение = "/edit", метод = RequestMethod.POST)
@RequireHardLogIn
public String editUnit(@RequestParam("unit") final String unit, @Valid final B2BUnitForm unitForm,
    окончательный BindingResult(bindingResult), окончательная модель (model), окончательные атрибуты перенаправления (redirectModel)
    выдает CMSItemNotFoundException
{
}
```

```

...
...

попытаться
{
    b2bUnitFacade.updateOrCreateBusinessUnit(единица,           unitData);
    окончательная ConsentForm agreeForm = unitForm.getConsentForm(); если
    (consentForm != null)
    {
        если (consentForm.getConsentGiven()) {

            получитьConsentFacade().датьConsent(consentForm.получитьConsentTemplateId(),           agreeForm.getConsentTemplateVersion());
        }
        иначе если (consentForm.getConsentCode() != null) {

            получитьConsentFacade().withdrawConsent(consentForm.getConsentCode());
        }
    }
}
поймать (финальное ModelSavingException e) {

    LOG.error(String.format("Не удалось сохранить объект. Возможно, идентификатор неуникальный (исходный идентификатор: [%s], новый идентификатор: [%s].",
        unit, unitData.getUid()), e); GlobalMessages.addErrorMessage(модель,
    привязкаResult.rejectValue("uid", "form.b2bunit.notunique", "глобальная ошибка");
    addBusinessUnitConsentDataToModel(модель);

    вернуть ControllerConstants.Views.Pages.MyCompany.MyCompanyManageUnitEditPage;
}
GlobalMessages.addFlashMessage(redirectModel, GlobalMessages.CONF_MESSAGES_HOLDER,
                               "form.b2bunit.success");

return String.format(REDIRECT_TO_UNIT_DETAILS, urlEncode(unitForm.getUid()));
}

.Добавьте необходимые операторы импорта в каждый файл.

```

Адаптация формы интерфейса

. Добавить Форма согласия собственность B2BUnitForm.java (расположен в commerceorgaddon/acceleratoraddon/web/src/de/hybris/platform/commerceorgaddon/forms):

```

публичный класс B2BUnitForm
{
    ...
    частный  Форма согласия Форма согласия;

    ...

    публичная ConsentForm getConsentForm() {

        возврат согласияForm;
    }

    public void setConsentForm(final ConsentForm agreeForm) {

        эта.consentForm = форма согласия;
    }
}

```

. B2BUnitForm.tag, добавьте блок прямо над кнопками для управления отображением флагка согласия. Этот файл находится в commerceorgaddon/acceleratoraddon/web/webroot/WEB-INF/tags/responsive/company.

```

...
<c:if test="${ not empty agreeTemplateData }">
    <form:hidden path="consentForm.consentTemplateId" value="${consentTemplateData.id}" /> <form:hidden
    path="consentForm.consentTemplateVersion" value="${consentTemplateData.version}" /> <form:hidden
    path="consentForm.consentCode" value="${consentTemplateData.consentData.code}" /> <div class="checkbox">

        <label class="control-label uncased">
            <c:выбрать>
                <c:when test="${not empty agreeTemplateData.consentData && empty agreeTemplateData.consentData.co
                    <form:checkbox path="consentForm.consentGiven" отмечено="отмечено" /> </c:when>

                <c:иначе>
                    <form:checkbox path="consentForm.consentGiven" отмечено="" /> </c:otherwise>
                </c:иначе>
            </c:выбрать>
            <c:out value="${consentTemplateData.description}" /> </label>
        </div>
    </c:if>
...

```

. Добавить табличное заявление для ядра JSTL тегов.

Теперь пользователи могут перейти на страницу «Изменить единицы измерения» в магазине электроинструментов и установить флагок, чтобы дать согласие на получение маркетинговых рассылок.

Родительская тема: [Управление соглашением](#)

Сопутствующая информация

[Дать согласие](#)

[Журналы согласия](#)[Шаблоны согласия](#)[Версии шаблона согласия](#)[Анонимное управление согласием](#)[Удаление персональных данных](#)

Анонимное управление согласием

Anonymous Consent Management дает анонимным пользователям контроль над отслеживанием их данных. Анонимные пользователи могут давать или отклонять свое согласие для приложений, которые собирают и обрабатывают персональные данные.

Эта функция позволяет управлять согласием как для анонимных, так и для зарегистрированных пользователей. Анонимные пользователи могут управлять согласием своих анонимных пользовательских данных, которые в конечном итоге могут быть связаны с их зарегистрированными пользовательскими данными. Зарегистрированные пользователи могут просматривать и редактировать свои согласия в новом разделе [Управление согласием](#) в моей учетной записи.

Когда пользователи впервые посещают сайт, в верхней части страницы появляется баннер, предлагающий им согласиться или отклонить согласие. Когда пользователи регистрируются, они могут дать согласие. Однако, если зарегистрированные пользователи не включают флагок согласия, система считает это отказом от согласия, и баннер с сообщением больше не появляется.

Всплывающие окна с анонимным согласием

Анонимные пользователи могут решить, хотят ли они, чтобы их данные отслеживались, приняв или отклонив согласие. Всплывающие окна согласия появляются в верхней части витрины и отображают название конкретной службы отслеживания пользователей. Они также отображают описание конкретной службы отслеживания при нажатии на шеврон.

Когда анонимный пользователь принимает или отклоняет свое согласие, всплывающее окно исчезает, и решение обрабатывается с помощью следующего кода из `acc.consent.js`:

```
...
changeConsentState:функция(элемент, agreeState){
    var agreeCode = $(element).closest('.consentmanagement-bar').data('code'));
    $.ajax({
        URL: ACC.config.encodedContextPath+"/consents/"+consentCode+"?state="+consentState,
        ТИП: 'ПОЧТА',
        успех: функция () {
            $(element).closest('.consentmanagement-bar').hide();
        }
    });
}
...
...
```

Шаблоны анонимного согласия

Шаблоны согласия отражают, для чего необходимо согласие анонимного пользователя, и находятся в административной панели бэк-офиса. Шаблон согласия пользователя. Каждая служба отслеживания пользователей должна иметь как минимум один шаблон согласия.

Один и тот же шаблон согласия может иметь несколько версий, и последняя версия отображается в [Версия](#) колонке. Когда определенный шаблон согласия изменяется и создается новая версия, то на витрине магазина отображается новое всплывающее окно согласия, которое пользователь может принять или отклонить.

Шаблон открытого согласия

Анонимный шаблон можно отобразить или скрыть на витрине магазина, переключив флагок «Отображение» на значение «Истина» или «Ложь»:

Вы также можете настроить типы элементов атрибута `Exposed` в `commerce_items.xml` как показано ниже:

```
<itemtype code="ConsentTemplate" jaloClass="de.hybris.platform.commerceservices.jalo.consent.ConsentTemplate" autocreate="true" generics="...
<description>Тип согласия, связанный с определенным магазином.</description> <deployment table="ConsentTemplates" typecode="6233" propertyTable="ConsentTemplateProps" /> <attributes>
...
<квалификатор атрибута="exposed" тип="boolean">
<description>Указывает, следует ли предоставлять шаблон согласия интеграторам в реализации витрины как параметр <persistence type="property" />
<defaultValue>ложь</defaultValue> </attribute>
...
</тип_элемента>
```

Анонимное согласие на хранение данных

Существует два случая хранения данных о согласии:

- Для анонимных пользователей: данные согласия хранятся в анонимные-согласия-печенье в витрине и в X-Анонимные-Согласия заголовок HTTP в ОСС.
- Для зарегистрированных или вошедших в систему пользователей: данные согласия сохраняются в сеансе, а данные cookie игнорируются.

-Примечание

Данные согласия для анонимных пользователей также реплицируются в сеансе. Когда пользователь входит в систему, атрибут сеанса для анонимного пользователя переопределяется данными согласия вошедшего в систему пользователя.

Печенье

анонимные-согласияФайл cookie отражает различные согласия, а также решения пользователя и информацию, с которой пользователи могут взаимодействовать.

Application	Name	Value	Domain	Path	Expires / ...	Size	HTTP	Sec...	SameSite
Manifest	anonymo...	%5B%7B%22templateCode%22%3A%22MERCHANDISING%22%2C%22templateVersi...	electronics...	/	2018-12...	225	✓	✓	
Service Workers	_utmt	1	.electronics...	/	2017-12...	7			
Clear storage	JSESSIONID	35B9021D73A6C6FFF0E58D5D8A47E0E6	electronics...	/ya...	Session	42	✓	✓	
Storage	_utm...	37038324	.electronic...	/	Session	14			
Local Storage	_utmb	37038324.1.10.1514396127	.electronic...	/	2017-12...	30			
Session Storage	_utma	37038324.142089890.1514396127.1514396127.1514396127.1	.electronic...	/	2019-12...	59			
IndexedDB	_utmz	37038324.1514396127.1.1.utmcsr=(direct) utmccn=(direct) utmcmd=(none)	.electronic...	/	2018-06...	75			
Web SQL	cookie-no...	NOT_ACCEPTED	electronics...	/	2029-05...	31		✓	
Cookies									
https://electronics.local:900									

Анонимные файлы cookie согласия помечены как Secure, так и HttpOnly.

HTTP-заголовок

X-Анонимные-СогласияЗаголовок HTTP используется в OCC REST API как эквивалентanonimnye-soglasiya cookie. Значение cookie и значение заголовка HTTP имеют одинаковый формат.

Сессия

После того, как пользователь войдет в систему или зарегистрируется, данные о согласиичитываются с бэкенда только один раз и переносятся в сеанс.

Раздел управления согласием

The Управление согласием раздел отображает решения пользователей по различным сервисам отслеживания пользователей. Он находится в Мой счети позволяет зарегистрированным пользователям управлять своим выбором согласия.

Дополнительную информацию об управлении согласием в SAP Commerce Accelerator см. [Управление согласием](#).

Локализация

Текущий язык сохраняется в сеансе. Если пользователь меняет язык, шаблон автоматически удаляется из сеанса и загружается новый на новом языке. См. следующий фрагмент кода из DefaultAnonymousConsentFacade.java:

```
/**
 * Удалять CONSENT_TEMPLATES из сеанса при смене языка
 */
защищенный void checkLanguageChange() {
    окончательная строка currentLanguage = storeSessionFacade.getCurrentLanguage().getIsoCode(); окончательная строка
    previousLanguage = sessionService.getAttribute(PREVIOUS_CONSENT_LANGUAGE); если
    (StringUtil.isEmpty(previousLanguage) || !currentLanguage.equals(previousLanguage)) {
        sessionService.removeAttribute(SENTENCE_TEMPLATES);
        sessionService.setAttribute(PREVIOUS_CONSENT_LANGUAGE, currentLanguage);
    }
}
```

Родительская тема:[Управление согласием](#)

Сопутствующая информация

[Дать согласие](#)

[Журналы согласия](#)

[Шаблоны согласия](#)

[Версии шаблонов согласия](#)

[Добавление точек входа согласия](#)

[Удаление персональных данных](#)

[Основные элементы использования](#)

Изучите различные варианты использования анонимного управления согласием.

Анонимный пользователь

Анонимные пользователи могут делать выбор в форме согласия.

Ускоритель и OCC API

Каждый шаблон согласия анонимного пользователя отображается для каждого последующего запроса анонимного пользователя до тех пор, пока на него не будет дан ответ.

Описанная логика реализуется синхронизировать Анонимные Согласия Метод в DefaultAnonymousConsentFacade.

Переход от статуса анонимного пользователя к статусу зарегистрированного клиента

Анонимные пользователи могут сделать выбор в форме согласия, а затем зарегистрироваться для создания новой учетной записи. В таком случае их выбор соответствующим образом переносится в созданную учетную запись.

Ускоритель

Выборы, выбранные пользователем, сохраняются в таблице согласия бэкэнда и отображаются в сеансе. Это делается только один раз в процессе регистрации.

Описанная логика реализуется процесс Регистрация Пользователя Запрос Метод в Аннотация Регистрация Страницы Контроллеров.

API-интерфейс OCC

В процессе регистрации согласие от X-Анонимные-Согласия HTTP-заголовки переносятся в новую учетную запись пользователя. Это также работает, когда гостевой пользователь создает учетную запись после оформления заказа.

Когда регистрируется анонимный или гостевой пользователь, метод регистрации из DefaultCustomerAccountService переносит согласия, если ЗАПЛННИЕ_СОГЛАСИЙ_ВКЛЮЧЕНО для атрибута session установлено значение true.

Переход от статуса анонимного пользователя к статусу зарегистрированного клиента

Когда анонимный пользователь входит в существующую учетную запись, любые варианты согласия, сделанные до входа в систему, игнорируются.

Ускоритель

При входе в систему зарегистрированного пользователя любые данные cookie анонимного пользователя игнорируются, и вместо них используются данные согласия из сеанса зарегистрированного клиента.

The Customer Consent Data Strategy, принадлежащий Authentication Success Method в Storefront Authentication Success Handler, используется для заполнения данных о согласии в сеансе при входе в систему.

API-интерфейс OCC

Ценность X-Анонимные-Согласия Заголовок HTTP игнорируется, когда анонимный пользователь входит в систему как существующий пользователь.

Переход от зарегистрированного клиента к анонимному пользователю при выходе из системы

Когда клиент выходит из системы, все сделанные им анонимно выборы вступают в силу.

Ускоритель

После выхода из системы вся ранее использованная информация о сеансе зарегистрированного клиента исчезает, и на смену ей приходит существующая информация из cookie-файлов анонимного пользователя.

При выходе из сеанса данные о согласии удаляются из сеанса onLogoutSuccess Method в Storefront Logout Success Handler.

API-интерфейс OCC

Это действие не применимо к OCC v2.

Удаление персональных данных

Клиенты могут закрыть свою учетную запись в магазине и удалить свои персональные данные. Задания Cron включены в `acceleratorCore` расширение очищает данные, связанные с деактивированными клиентами.

Когда клиенты решают закрыть свой аккаунт в магазине, они переходят на страницу [Мой аккаунт](#). Закрыть аккаунти нажмите [Закрыть счет](#). Дата закрытия счета устанавливается как дата деактивации для клиента. После закрытия учетной записи клиент не может использовать ее для доступа к витрине, но может обратиться в службу поддержки клиентов, чтобы отменить деактивацию до истечения срока хранения (по умолчанию два дня). Когда срок хранения истекает, задания cron удаляют данные, связанные с клиентом, такие как:

- Адрес электронной почты
- Контактная информация
- Подробности доставки

- предпочтения по доставке
- настройки согласия
- реквизиты платежа

История заказов не удаляется одновременно с данными, связанными с клиентами, поскольку для данных, связанных с заказами, установлен отдельный период хранения (по умолчанию 10 лет). Клиенты могут обратиться в службу поддержки клиентов в течение периода хранения заказов, если им нужна какая-либо информация об истории заказов.

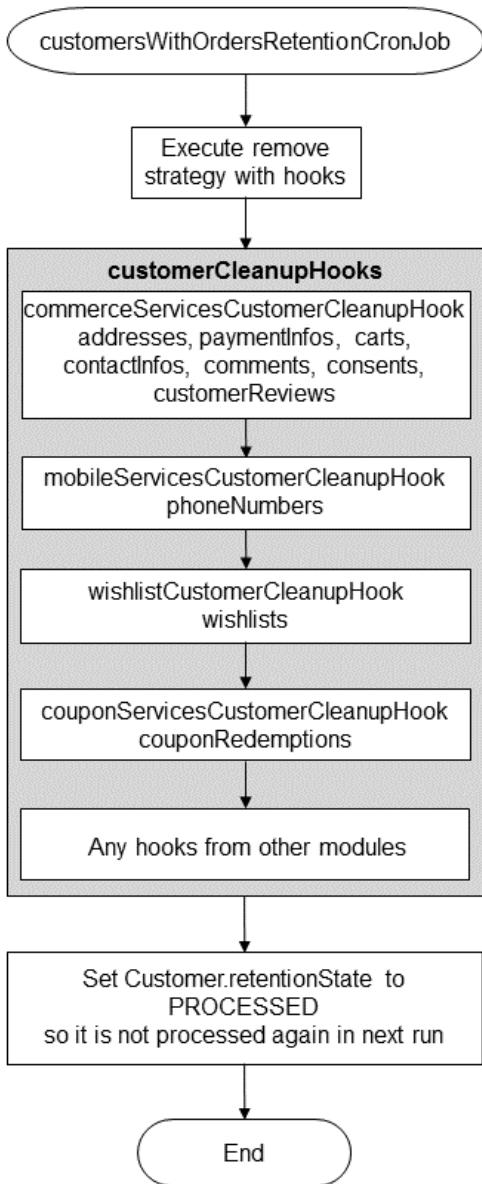
Очистка заданий Cron

Для удаления клиентов, заказов и соответствующих связанных объектов данных используются три задания cron:

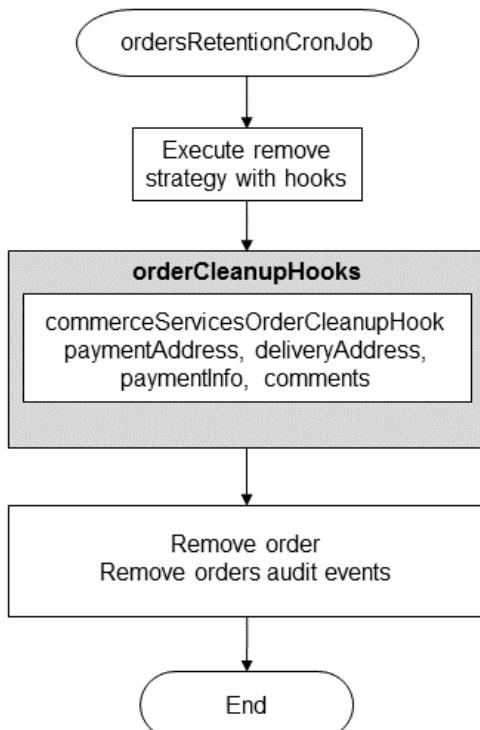
Крон работа	Описание	время удержания
клиентыБезЗаказовУдержаниеСronJob	<p>Очищает данные, связанные с клиентами, и удаляет деактивированных клиентов, у которых нет заказов.</p> <p>Находит всех деактивированных клиентов на основе следующих условий:</p> <ul style="list-style-type: none"> • У клиента в настоящее время нет заказов • Текущая дата уже прошла Датадеактивации + Времяудержания. Клиенты могут отменить деактивацию до истечения срока хранения. 	2 дня
клиентыWithOrdersRetentionCronJob	<p>Очищает данные, связанные с деактивированными клиентами которые владеют заказами. Состояние.удержания.клиента установлен на PROCESSED, чтобы последующие выполнения этого задания cron не обрабатывали того же клиента.</p> <p>Находит всех деактивированных клиентов на основе следующих условий:</p> <ul style="list-style-type: none"> • В настоящее время у клиента есть заказы • Текущая дата уже прошла Датадеактивации + Времяудержания. • Клиент еще не обработан этим заданием cron. 	2 дня
заказыRetentionCronJob	<p>Очищает заказы и данные, связанные с заказами, для деактивированных клиентов. Он собирает только заказы клиентов, которые уже обработаны клиентыWithOrdersRetentionCronJob.</p> <p>Находит заказы на основе следующих условий:</p> <ul style="list-style-type: none"> • Срок действия заказа более 10 лет • Клиент деактивирован и Состояние.удержания.клиента ОБРАБОТАНО <p>-Примечание The заказыRetentionCronJob не принимает клиенты состояния удержания переменная, которую следует учитывать при удалении заказов.</p>	10 лет

Эти задания cron определены в `yacceleratorcore/resources/yacceleratorcore/import/common/cronjobs.import` из `yacceleratorcore` расширение. Определения автоматически запускают задания cron один раз в день. Следующие диаграммы иллюстрируют работу заданий cron.

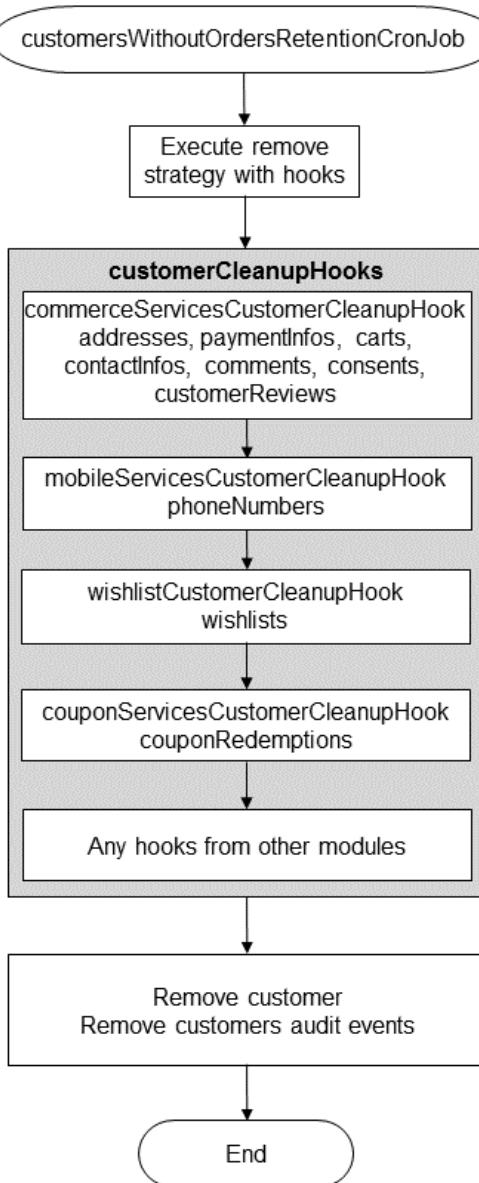
Первое задание cron, которое нужно запустить, это `клиентыWithOrdersRetentionCronJob`, который находит всех деактивированных клиентов, у которых истек период хранения клиентов и есть заказы в их аккаунте, и удаляет данные, связанные с клиентами:



Второе заданиеcron для запуска —заказыRetentionCronJob, который находит все заказы, срок хранения которых истек, и удаляет данные, связанные с заказами:



Последнее задание cron, которое нужно выполнить, это клиентыБезЗаказовУдержаниеCronJob, который находит всех деактивированных клиентов, у которых истек период удержания клиентов, но нет заказов в их аккаунте, и удаляет данные, связанные с клиентами:



Вы можете настроить каждое из заданий cron для ссылки Правило удержания. Кроме того, вы можете настроить `RetentionRule` для ссылки на компонент Spring, который может использоваться для подключения пользовательской логики для выполнения определенной задачи. Платформа хранения данных предоставляет этот механизм. Модуль Platform предоставляет Абстрактное Расширяемое Удаление Очистка Действие и `DefaultExtensibleRemoveCleanupAction`, который можно использовать для подключения к `RetentionRule` для выполнения логики, которая очищает элемент и связанные с ним данные. Реализация по умолчанию `DefaultExtensibleRemoveCleanupAction` реализует очистку (финальное After `RetentionCleanupJobPerformable`) `retentionJob`, финальное правило `AbstractRetentionRuleModel`, финальный элемент `ItemToCleanup` метод. Этот метод имеет механизм перехвата, который вызывает список перехватов для очистки любых данных, связанных с элементом, очищаемым заданием cron. После выполнения всех перехватов элемент удаляется, а также удаляются любые записи аудита, относящиеся к очищаемому элементу.

Действия по очистке

Задания cron для стирания ссылаются на три действия в `acceleratorcore` расширение:

- клиентОчисткаСвязанныеОбъектыДействие ссылается на `клиентыWithOrdersRetentionCronJob`
- `customerNotOwningOrdersУдалитьОчистка` Действие ссылается на `клиентыБезЗаказовУдержаниеCronJob`
- `orderRemoveCleanupAction` ссылается на `заказыУдержаниеРабота`

`customerCleanupHooks` из `заказыОчисткаКрючки` соответственно вводятся в эти действия.

Механизм привязки клиентов и уборки заказов

Механизм перехвата в модуле Platform доступен, так что вы можете добавлять перехваты для очистки любых данных, связанных с клиентами или заказами, в ваших модулях. Модуль Platform включает `заказОчисткаКрючки` и `customerCleanupHooks` как списки Spring util. Чтобы очистить любые данные, связанные с клиентами или заказами, добавьте свои хуки в один из этих списков. Обратите внимание на следующее:

- Крючки добавлены в `customerCleanupHooks` вызываются перед удалением или очисткой клиента.

- Крючки добавлены в заказа Крючки вызываются перед удалением или очисткой заказов.

Если какие-либо записи аудита сохраняются для объектов, связанных с клиентом и заказом, конкретные хуки должны удалить записи аудита. Модуль Platform предоставляет API для удаления записей аудита. Метод API для удаления записей аудита должен вызываться сразу после удаления связанного объекта.

Вот фрагмент кода, демонстрирующий удаление записей аудита:

```
WriteAuditRecordsDAO writeAuditRecordsDAO;

// удалить записи аудита
окончательный тип ComposedTypeModel = getModelService().get(PK.parse(item.getItemType()));
writeAuditRecordsDAO().removeAuditRecordsForType(type.getCode(), item.getPK());
```

Добавление нового крючка

Чтобы создать хук для очистки данных, связанных с клиентами и заказами, начните с создания класса Java. Мы рекомендуем вам поместить этот класс в следующий пакет вашего расширения, <ваш_базовый_пакет>. <имя_расширения>.удержание.имплпакет, и что вы используете соглашение об именовании <ИмяРасширения> <ИмяТипаЭлемента>CleanupHook. Примером может служить этот хук:

```
<your_base_package>.commerceservices.retention.impl.CommerceServicesCustomerCleanupHook
```

Хук должен реализовывать <ваш_базовый_пакет>.retention.hook пакет.ItemCleanupHookИнтерфейс. Конкретная реализация ItemCleanupHookследует реализовать очисткаRelatedObjects()метод, имеющий логику удаления объектов данных, связанных с элементом, предоставленным заданием cron.

DefaultExtensibleRemoveCleanupActionвызывает очисткаRelatedObjects()метод перед удалением или очисткой элемента. Если пользовательское действие реализовано путем расширения Абстрактное Расширяемое Удаление Очистка Действие, что действие должно вызывать очисткаRelatedObjects()метод, чтобы он вызывал механизм перехвата. Например, этот код показывает, как перехват очищает объекты, связанные с клиентами, в коммерции услуг расширение:

```
/*
 * [y] hybris Платформа
 *
 * Авторские права (c) 2000-2017 SAP SE
 * Все права защищены.
 *
 * Данное программное обеспечение является конфиденциальной и частной информацией SAP.
 * Hybris («Конфиденциальная информация»). Вы не должны разглашать такую информацию.
 * Конфиденциальная информация и использовать ее только в соответствии с
 * условиями лицензионного соглашения, заключенного вами с SAP Hybris.
 */
пакет de.hybris.platform.commerceservices.retention.impl;

импорт статического de.hybris.platform.servicelayer.util.ServicesUtil.validateParameterNotNullStandardMessage;

импорт de.hybris.platform.comments.model.CommentModel;
импорт de.hybris.platform.commerceservices.consent.dao.ConsentDao;
импорт de.hybris.platform.commerceservices.model.consent.ConsentModel;
импорт de.hybris.platform.core.model.order.CartModel;
импорт de.hybris.platform.core.model.order.payment.PaymentInfoModel;
импорт de.hybris.platform.core.model.user.AbstractContactInfoModel;
импорт de.hybris.platform.core.model.user.AddressModel;
импорт de.hybris.platform.core.model.user.CustomerModel;
импорт de.hybris.platform.customerreview.model.CustomerReviewModel;
импорт de.hybris.platform.retention.hook.ItemCleanupHook;
импорт de.hybris.platform.servicelayer.model.ModelService;

импорт org.slf4j.Logger;
импорт org.slf4j.LoggerFactory;
импорт org.springframework.beans.factory.annotation.Required;

/**
 * Этот хук удаляет объекты, связанные с клиентами, такие как адреса, способы оплаты, корзины и контактную информацию.
 */
открытый класс CommerceServicesCustomerCleanupHook реализует ItemCleanupHook<CustomerModel> {

    частный статический финальный Logger LOG = LoggerFactory.getLogger(CommerceServicesCustomerCleanupHook.class);

    частный МодельСервис modelService;
    частный СогласиеДао согласиеДао;

    @Переопределить
    public void cleanupRelatedObjects(final CustomerModel customerModel) {
        validateParameterNotNullStandardMessage("customerModel", customerModel);
        если (LOG.isDebugEnabled()) {
            LOG.debug("Очистка объектов, связанных с клиентами, для: {}", customerModel);
        }
        // удалить адреса
        для (конечный адрес AddressModel: customerModel.getAddresses()) {
            getModelService().remove(адрес);
        }
        .....
    }
}
```

```

зашитенный МодельСервис получитьModelService()
{
    возвращающаяся модельСервис;
}

@Необходимый
public void setModelService(final ModelService modelService) {

    этот.modelService = modelService;
}

зашитенный ConsentDao getConsentDao()
{
    возвращающаяся согласиеDao;
}

@Необходимый
public void setConsentDao(final ConsentDao agreeDao) {

    это.consentDao = согласиеDao;
}

}

```

Наконец, создайте компонент Spring для класса хука и добавьте компонент в customerCleanupHooks или в список очистки клиентов. Например, эта конфигурация Spring создает компонент Spring, который добавляет его в customerCleanupHooks список.

```

<!-- Уборка клиентов -->
<bean id="commerceServicesCustomerCleanupHook"          class="de.hybris.platform.commerceservices.retention.impl.CommerceServicesCustomerClean
    <свойство имя="modelService" ref="modelService"/>
    <свойство имя="consentDao" ref="consentDao"/>
</боб>

<bean id="commerceServicesCustomerCleanupHookMergeDirective" зависит от="customerCleanupHooks" родительский="listMergeDirective" >
    <property name="add" ref="commerceServicesCustomerCleanupHook" /> </bean>

```

Родительская тема:[Управление согласием](#)

Сопутствующая информация

[Дать согласие](#)

[Журналы согласия](#)

[Шаблоны согласия](#)

[Версии шаблона согласия](#)

[Добавление точек входа согласия](#)

[Управление анонимным согласием](#)

[Реализация логики очистки Структура](#)

[Хранения данных](#)

Несколько сохраненных корзин

SAP Commerce Accelerator предоставляет функцию сохраненной корзины, которая позволяет пользователям сохранять одну или несколько корзин для дальнейшего использования.

Пользователи могут воспользоваться функцией сохраненной корзины, чтобы улучшить свой опыт онлайн-покупок. Например, они могут создать сохраненную корзину для товаров, которые они покупают на регулярной основе, или создать различные сохраненные корзины для определенных типов покупок.

Технический обзор

Функция сохраненной корзины является частью Core Accelerator, поэтому она доступна в других адаптивных ускорителях, таких как B2B. Два метода расширяют функцию сохраненной корзины, включенную в коммерцияуслуги расширение, позволяющее сохранять несколько корзин:

Метод	Описание	Возвраты
получитьSavedCartsCountForSiteAndUser(BaseSiteModel, UserModel)	Добавлен новый метод для получения количества сохраненных корзин пользователя.	Целое число
getSavedCartsForSiteAndUser(PageableData, BaseSiteModel, UserModel, List<OrderStatus>)	Измененный метод, используемый для разбиения на страницы, а также включающий возможность сортировки сохраненных корзин.	SearchPageData<CartModel>

Более подробную информацию о других методах сохранения корзин см. в разделе «Сохранить корзину» [расширение commerceservices](#).

По умолчанию функция сохраненной корзины включена в следующих витринах:

- Apparel-UK (B2C)
- Apparel-DE (B2C)
- Электроника (B2C)

- Электроинструменты (B2B)

Чтобы включить сохраненную корзину в других магазинах, добавьте следующий код в свойственные свойства:

```
# включение ускорителя сохранения корзины для удаления корзины сеанса и получения новой корзины для пользователя в магазине электроники
acceleratorServices.commercesavecart.sessioncart.hook.enabled.<витрина>=правда
```

```
# включение ускорителя сохранения корзины восстановления хука для установки saveTime как null перед выполнением восстановления в магазине электроники
acceleratorServices.commercesavecart.restoration.savetime.hook.enabled.<витрина>=правда
```

Если нет <витрина> указан, то сохраненная корзина будет включена для всех витрин.

Сохраненное поведение корзины

Пользователи должны войти в магазин, чтобы создать сохраненные корзины. Если они не вошли в систему, они будут перенаправлены на страницу входа, когда захотят сохранить свою корзину.

Сохраненные корзины создаются из текущей активной корзины сеанса (в один момент времени может быть только одна активная корзина). При создании сохраненной корзины активная корзина добавляется в качестве сохраненной корзины, и создается новая активная корзина сеанса.

При восстановлении сохраненной корзины она становится активной корзиной. Если в активной корзине есть какие-либо товары до восстановления сохраненной корзины, пользователь может создать отдельную сохраненную корзину с этими товарами. В настоящее время нет возможности объединить сохраненные и активные корзины.

Пользовательский опыт

В этом разделе подробно описывается, как пользователи создают, просматривают и восстанавливают сохраненные корзины в магазине.

Создание сохраненных корзин

Пользователи создают сохраненные корзины на странице оформления заказа. Нажав на **Новая корзинна ссылку**, открывает диалоговое окно, в котором пользователи вводят имя и описание корзины, а затем сохраняют корзину. Пользователи должны войти в систему, чтобы сохранить корзины; если они не вошли в систему и нажмут **Новая корзинна ссылку**, они перенаправляются на страницу входа в систему, а затем возвращаются на страницу оформления заказа после входа в систему.

Если корзина включает в себя какие-либо купоны или акции, примененные к позициям или всему заказу, они сохраняются вместе с корзиной. При восстановлении сохраненной корзины все купоны и акции применяются при условии, что они не истекли на момент восстановления корзины.

Просмотр сохраненных корзин

Пользователи могут просматривать все свои сохраненные корзины одним из двух способов:

- Щелкнув **Моя учетная запись** в **Сохраненные корзины** в заголовочном меню
- Нажав на **У вас есть <X> Сохраненные корзины** ссылку на странице оформления заказа

Любой из методов отображает список сохраненных корзин. Сортировать по раскрывающийся список позволяет пользователям сортировать записи по:

- Дата изменения
- Дата сохранения
- Идентификатор корзины
- Имя корзины
- Общая сумма корзины

В списке сохраненных корзин пользователи могут выполнять следующие действия:

- Восстановите сохраненную корзину, нажав **Восстановить**
- Удалить корзину, нажав **X**
- Просмотрите данные корзины, нажав на ее название.

На странице сохраненных данных корзины отображаются все товары и их соответствующие количества. Также отображаются любые рекламные акции, применяемые к позициям. На странице сведений пользователи могут выполнять следующие действия:

- Редактировать сохраненное название и описание корзины
- Удалить** тележка
- Восстановить тележка
- Иди **Вернуться к сохраненным корзинам** чтобы увидеть список сохраненных корзин

Восстановление сохраненных корзин

Когда пользователи нажимают **Восстановить** в списке сохраненных корзин или на странице сведений, появляется всплывающее окно, в котором пользователи могут сохранить копию сохраненной корзины для будущего использования. Это полезно для повторяющихся заказов. После восстановления сохраненной корзины она становится активной корзиной.

Если при восстановлении сохраненной корзины в активной корзине уже были товары, пользователи также имеют возможность создать новую, отдельную сохраненную корзину для этих товаров, поскольку они не объединяются с восстановленной корзиной.

Если в сохраненной корзине есть какие-либо купоны или акции, примененные к позициям, они будут восстановлены при условии, что их срок действия не истек на момент восстановления корзины.

Если количество товара в сохраненной корзине превышает доступный запас на момент восстановления корзины, то перед завершением оформления заказа количество корректируется в соответствии с доступным запасом.

Сопутствующая информация

[Расширение commerceservices](#)

Будущая доступность запасов

Функция «Будущая доступность запасов» указывает, когда запасы будут пополнены новыми запасами.

Для получения более подробной информации см.[B2B Будущая доступность запасов](#).

Токен проверки

Эта функция предоставляет API для управления кодом верификации токена. Клиент может запросить код верификации токена для определенной цели, и после проверки система доставляет его на указанный канал. Этот код может быть далее использован для проверки личности или транзакций, тем самым повышая безопасность системы.

Например, с помощью этой функции сценарий входа в систему может выглядеть следующим образом:

Клиент вводит имя пользователя и пароль, затем запрашивает код подтверждения.

. Система проверяет запрос и отправляет код токена проверки по указанному каналу, например, на электронную почту клиента.

Клиент получает код токена проверки из назначенного канала и вводит его в систему вместе со своим запросом на вход.

Весь процесс защищает вход клиента в систему от несанкционированного доступа.

Введение

Получить токен проверки

Для допустимого запроса на получение токена проверки система генерирует запись, содержащую идентификатор токена, код токена, временную метку и информацию о пользователе.

Идентификатор токена служит для идентификации клиента и отправляется обратно клиенту.

Код токена, используемый для последующей проверки личности, доставляется системой по заранее определенному каналу на асинхронной основе.

Если запрос на создание токена проверки не соответствует стандартам безопасности, система все равно назначает клиенту идентификатор токена. Однако система не будет генерировать или отправлять код токена.

Проверка токена

Если предоставленный идентификатор токена и код токена совпадают и еще не истекли, система успешно проверяет их, и токен проверки мгновенно удаляется. В противном случае проверка не проходит.

При входе в систему с использованием функции токена верификации (`otp.customer.login.enabled`) включен, фреймворк безопасности Spring оптимизирован для приема идентификатора токена и кода токена в качестве учетных данных пользователя.

- Для OAuth2: Клиенты могут получить токен OAuth2, применив идентификатор токена в качестве имени пользователя и код токена в качестве пароля. Если проверка токена прошла успешно, выдается токен OAuth2.
- Для Accelerator для аутентификации необходимы как идентификатор токена, так и код токена. Успешная проверка токена приводит к установлению аутентифицированного сеанса.

Конфигурация

В этой таблице перечислены все настраиваемые свойства, связанные с токеном проверки. Перезапустите сервер, чтобы применить изменения.

Имя свойства	Пример значения	Описание
<code>otp.customer.login.enabled</code>	ЛОЖЬ	Вид одноразового пароля (OTP)
<code>otp.customer.login.token.code.encoder</code>	пбкdf2	Определяет кодировщик токенов, который будет использоваться для Oznakomytesc s dostupnymi variantami vparol'Epnc de.hybris.platform.persistence

Имя свойства	Пример значения	Описание
otp.customer.login.token.code.length	8	Когда OTP включен, он позволяет указать функциональность. Минимально допустимое количество символов — 6.
otp.клиент.логин.токен.код.алфавит	1234567890ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz	Если включена функция OTP, можно задать специальную функцию входа с помощью OTP. Минимальная длина набора символов должна быть 10, по умолчанию используется значение 1234567890ABCDEF.
otp.customer.login.token.ttlсекунды	300	Когда OTP включен, он позволяет указать функциональность. Минимально допустимое значение — 60 секунд (15 минут).
otp.customer.login.token.max.попыток.проверки	3	Виды максимальное количество неудачных в токен удаляется из системы. Вали
otp.customer.login.token.id.генератор.bits	256	Количество бит, используемых для генерации случайной функции. Минимально допустимое значение составляет 128 бит, рекомендуется использовать не менее 256 бит.
otp.customer.логин.токен.цель.короткое.имя	ЛГН	Это короткое имя используется для быстрой идентификации функциональности. Формат идентификатора токена:<(shortName){(rand Короткое имя должно быть уникальным для начальных символов, так как это не добавляет никакой безопасности. Если короткое имя превышает 10 символов, оно name.

Совместимость

Узнайте, какие версии ускорителей, интеграций и других компонентов SAP Commerce Cloud поддерживаются SAP Commerce Cloud в публичном облаке.

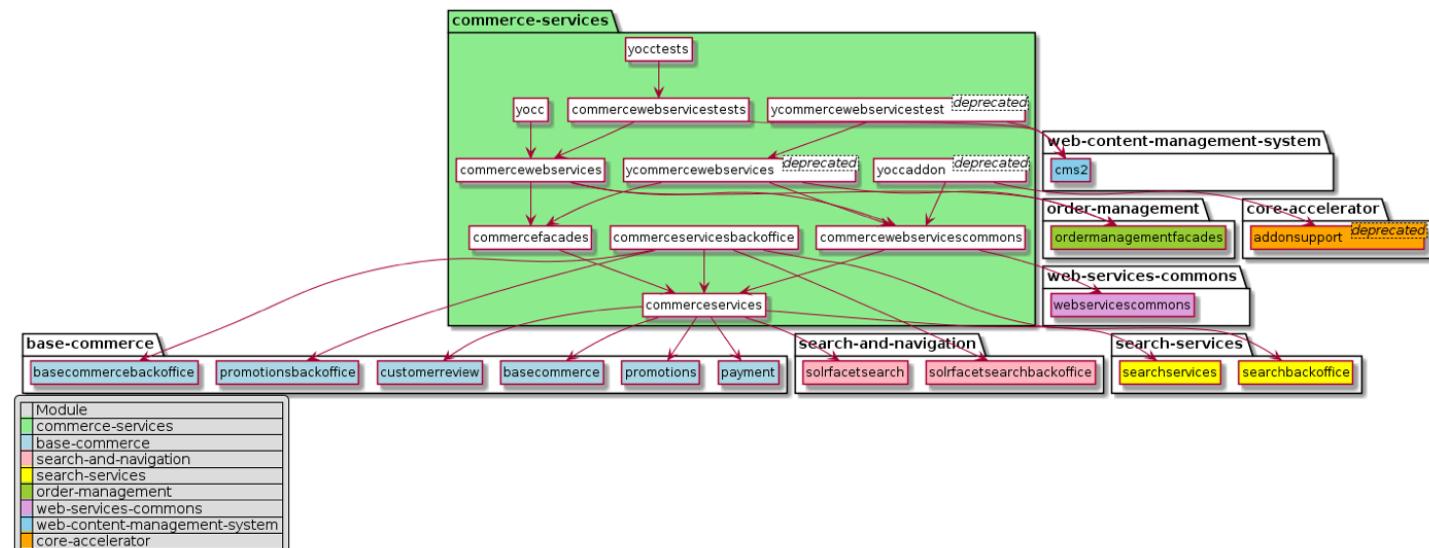
В настоящее время для использования функции входа с токеном верификации в SAP Commerce Cloud в публичном облаке поддерживаются следующие модули:

- Модуль B2C Accelerator AddOns. Для получения дополнительной информации см.[Войти с помощью токена проверки](#) в B2C-акселераторе.
- Модуль B2B Accelerator AddOns. Для получения дополнительной информации см.[Войти с помощью токена проверки](#) в B2B-акселераторе.
- Модуль China Accelerator People Prole. Для получения дополнительной информации см.[Войти с помощью токена проверки](#) в акселераторе для Китая.

Архитектура коммерческих услуг

Commerce Services включает в себя набор расширений, которые влияют на конфигурацию вашего продукта, несколько корзин и управление согласием клиентов.

Зависимости



Рецепты

Полный список рецептов SAP Commerce, которые могут включать этот модуль, см.[Рецепты установщика](#).

Полный список рецептов пакетов расширения интеграции SAP Commerce Cloud, которые могут включать этот модуль, см.[Справочник рецептов установщика](#).

Расширения

Модуль Commerce Services состоит из следующих расширений:

- [Расширение commerceservices](#)
- [Расширение commercefacades](#)
- [Расширение commercewebservicescommons](#)
- [Расширение ycommercewebservicestest](#)
- [Расширение ycommercewebservices](#)
- [Расширение йоккадона](#)
- [Расширение commercewebservices](#)
- [Расширение commercewebservicestests](#)
- [Расширение yocc](#)
- [Расширение yocctests](#)
- [коммерцияуслугибэк-офис Расширение](#)

Расширение commerceservices

TheКоммерцияуслугирасширение организует функциональность из одной или нескольких служб платформы. Эти службы выполняют определенные задачи самостоятельно, но часто их необходимо объединить для предоставления полного варианта использования B2C коммерции. Это часто подразумевает объединение функциональности из отдельных расширений, которые лицензируются отдельно.коммерцияуслугирасширение также расширяет более общие функциональные возможности некоторых расширений SAP Commerce, добавляя больше функций B2C-коммерции.

-Примечание

Расширение SAP Commerce может предоставлять функциональность, лицензируемую через различные модули SAP Commerce. Обязательно ограничьте реализацию функциями, определенными в вашей контрактной лицензии. В случае сомнений обратитесь к своему торговому представителю.

В следующих разделах представлено руководство по всем услугам и улучшениям обслуживания, предоставляемым с помощью коммерцияуслугирасширение. Зависимости от расширений за пределами базовой платформы выделены для справки.

Каталог услуг

В следующих разделах описываются услуги каталога коммерцияуслугирасширение.

Категория «Торговля» Услуги

TheКоммерцияКатегорияУслугидобавляет функциональность поиска по категории в рамках только каталогов продуктов текущего сеанса. Другие типы каталогов сеансов исключаются, такие как каталоги контента и каталоги классификаций.

Коммерческая цена Услуги

TheКоммерцияЦенаУслугапредназначен для замены использования существующего ЦенаУслуга. Это обеспечивает сокращенный интерфейс, чтобы отображать только типичные запросы цен на продукты B2C, такие как получение текущей веб-цены или цены from, в случае вариантов. Все дополнительныеЦенаУслуга Такие сложные моменты, как ценовой диапазон и выбор подходящей цены, либо скрыты, либо просто не поддерживаются этим сервисом.

Для получения более подробной информации см.[Расширение базовой коммерции](#).

-Примечание

Реализация по умолчанию добавляет зависимость к расширению basecommerce.

Стратегия чистого валового дохода

TheКоммерцияуслугирасширение расширяет расширение basecommerceBaseStoreвключить атрибут net ag, который определяет, должны ли цены на витрине быть указаны без учета налога с продаж или с учетом налога с продаж. NetGrossStrategy используется всеми службами, желающими определить, следует ли возвращать чистые или брутто-цены. NetPriceService, который является коммерцияуслугирасширение расширенная версия ЦенаУслуга использует эту стратегию. Есть две готовые реализации: DefaultNetGrossStrategy, который использует значение по умолчанию PriceFactory правила и CommerceNetGrossStrategy, который использует значение из BaseStore.

Налог с продаж

The коммерция услуга расширение расширяет расширение `basecommerceBaseStore` чтобы разрешить `UserTaxGroup` необходимо указать. Это можно использовать для установки налоговой группы, чтобы гарантировать, что налог с продаж будет взиматься правильно `ЦенаУслуга` когда сайты развернуты в нескольких странах.

Внешний налог

The коммерция услуга расширение интегрирует **Внешний налог** функциональность. **Внешние Налоги Услуги** используется во время оформления заказа `CommerceCheckoutService`. Внешний налоговый расчет состоит из общей ответственности за принятие решения о необходимости вызова налогового расчета и, если это так, за вызов расчета. Готовый `DefaultExternalTaxesService` использует **Пересчитывать Внешние Налоги Стратегия** определить, требуется ли внешний расчет налога. Если считается, что требуется новый расчет, **Рассчитывать Внешние Налоги Стратегия** призвано рассчитать налог. Для того, чтобы определить налоговый код продукта, **Налоговый Код Стратегия** использует `TaxAreaLookupStrategy` позовите **Продукт Налоговый Код Услуги** и получите налоговый кодекс.

Коммерческая продукция Услуги

The **Коммерция Продукт Услуга** добавляет функциональность фильтра класса классификации. Ранее эта служба предоставляла поддержку Stock, которая теперь устарела в пользу [Служба торговли акциями](#) (описано ниже).

Для получения более подробной информации см. [Расширение базовой коммерции](#).

-Примечание

Реализация по умолчанию добавляет неявную зависимость к расширению `basecommerce`.

Улучшения поиска по фасетам

Для расширения функциональности, предоставляемой в расширении `solrfacetsearch` в расширении `commerceservices`, был добавлен ряд функций B2C-коммерции.

Для получения более подробной информации см. [Расширение solrfacetsearch](#) и [Поиск и навигация](#).

-Примечание

The коммерция услуга расширение зависит от расширения `solrfacetsearch`.

Услуги по коммерческим платежам

В следующих разделах описываются услуги коммерческих платежей коммерция услуга расширение.

Код транзакции продавца платежа

Для авторизации платежей или создания подписок PSP требуется код транзакции торговца. Уровень коммерции обеспечивает **Генерировать стратегию кода транзакции торговца**. Реализация по умолчанию использует ID пользователя корзины с суффиксом в виде случайного UUID.

Борьба с мошенничеством

Чтобы создать запись платежной транзакции для пересмотра решения процесса отправки заказа, вам нужен правильный тип платежной транзакции. Это предоставляется **ОБЗОР_РЕШЕНИЯ** ценность, содержащаяся в **Тип Оплаты Транзакции**.

Заказать услуги

В следующих разделах описываются услуги заказа расширения `commerceservices`.

Служба торговли корзины

The **CommerceCartService** заменяет **Корзина Сервис** путем добавления к услугам корзины расчета акций, проверки запасов и очистки неиспользуемых товаров.

Для получения более подробной информации см. [Расширение базовой коммерции](#) и [Расширение CommerceCartService](#).

-Примечание

Реализация по умолчанию добавляет неявную зависимость к расширениям `basecommerce` и `promotions`.

Служба торговли акциями

The **CommerceStockService** предназначен для использования вместо **StockService** и предоставляет методы определения уровня запасов для **BaseStore** или **Точка обслуживания**. Это необходимо для функциональности «купить онлайн и забрать в магазине». **Коммерция Услуги** добавить дополнительные отношения из Склад Точка обслуживания, в дополнение к существующим отношениям от Склад Базовый Склад, и Точка обслуживания к Базовый Склад.

Для получения более подробной информации см. [Stock Service - Техническое описание](#) и [Расширение базовой коммерции](#).

-Примечание

Реализация по умолчанию добавляет неявную зависимость к расширению `basecommerce`.

TheCommerceCheckoutService добавляет услуги для организации типичных операций во время оформления заказа, таких как регистрация новой кредитной карты, добавление и настройка адреса доставки, выбор способа доставки и размещение заказа. Он предназначен для замены прямого использования Корзина Сервиса Заказ Сервиса.

Более подробную информацию см. в следующих источниках:

- [Расширение базовой коммерции](#)
- [Расширение cms2 — техническое руководство](#)
- [Учебное пособие по расширению cms2](#)
- [Продление платежа](#)
- [Расширение CommerceCheckoutService](#)

-Примечание

Реализация по умолчанию добавляет неявную зависимость к расширениям basecommerce, promotions и cms2. Интерфейс также имеет явную зависимость от расширения payment.

Служба доставки

TheСлужба доставки предоставляет функциональность вокруг доступных и поддерживаемых правил доставки, которые охватывают одну витрину магазина, а не развертывание всей платформы, которая может включать несколько витрин. Это включает определение поддерживаемых стран доставки и вариантов доставки.

Услуги по работе с клиентами

В следующих разделах описываются услуги по обслуживанию счетов клиентов коммерции услуга расширение.

Обслуживание счетов клиентов

Служба клиентских счетов обрабатывает типичные возможности управления клиентскими счетами. Она предоставляет методы для регистрации, проверки учетных данных пользователя, обновления настроек профиля, служб забытых паролей, управления адресными книгами, просмотра заказов пользователя для текущего магазина и управления платежной информацией с использованием интерфейса подписки, профессионального к PCI.

Более подробную информацию см. в следующих источниках:

- [Расширение базовой коммерции](#)
- [Расширение cms2 — техническое руководство](#)
- [Учебное пособие по расширению cms2](#)
- [Продление платежа](#)
- [Аутентификация с помощью кредитной карты с помощью сохраненного токена](#)

-Примечание

Реализация по умолчанию добавляет неявную зависимость к расширению cms2. Кроме того, интерфейс добавляет явную зависимость к расширениям basecommerce и payment.

Служба разрешения проблем по электронной почте для клиентов

Служба разрешения проблем с электронной почтой клиентов добавляет уровень абстракции поверх **Модель клиента** для определения адреса электронной почты клиента.

Безопасный сервис токенов

Для шифрования используется служба защищенных токенов. **SecureToken** данные в короткий зашифрованный токен и расшифровать токен.

TheSecureToken data состоит из строки данных и временной метки. Результирующая зашифрованная строка токена находится в формате Base64.

TheSecureTokenService используется в процессе запроса забытого пароля. Когда клиент запрашивает забытый пароль, генерируется токен с использованием сервиса, проходящий **SecureToken** данные, которые состоят из uid клиента и временной метки. Генерированный токен сохраняется в системе для клиента, и клиенту отправляется электронное письмо с URL сброса пароля, включающим токен. Когда клиент пытается сбросить пароль, заходя на страницу сброса пароля с помощью URL сброса пароля, токен расшифровывается и проверяется. Он также сверяется с токеном, сохраненным в системе для клиента, перед сбросом пароля клиента.

DefaultSecureTokenService

TheDefaultSecureTokenService реализует **SecureTokenService**.

Данные, которые необходимо зашифровать, подписываются и шифруются с использованием двух отдельных ключей. Ключи указываются как строки шестнадцатеричных символов. Требуется криптография неограниченной стойкости.

Собранные для шифрования данные включают в себя **Подпись** ключ (не включенный в блок данных) и блок данных, который сам по себе состоит из строки данных в **SecureToken**, сгенерированная контрольная сумма для строки данных в **SecureToken**, и временная метка в **SecureToken**. Блок данных также дополнен префиксом и постфиксом с длинами заполнения. Все вышеупомянутое, за исключением ключа подписи, шифруется AES с использованием ключа шифрования. Результат в формате Base64.

Услуги по поиску магазинов

В следующих разделах описываются услуги по поиску магазинов коммерция услуга расширение.

Служба поиска магазинов

The **StoreFinderService** заменяет устаревший **StoreLocatorService**. **StoreFinderService** организует различные сервисы, предоставляемые hybris Store Locator в расширении basecommerce, для предоставления типичной функциональности страницы локатора магазинов. Сюда входит поиск ближайшего магазина, поиск по городу или почтовому индексу, карты местоположений, расположенных поблизости от текущего местоположения, и контент, необходимый для страницы информации о магазине.

В магазин Brick and Mortar добавлены дополнительные атрибуты. **Точка обслуживания** для захвата контента, такого как изображение магазина, особенности магазина и некоторый маркетинговый контент для страницы сведений о магазине. Для получения дополнительной информации см. [Конфигурация локатора магазинов](#).

The **StoreFinderService** включает в себя следующие методы:

- **местоположениеПоиск** методы, которые находят близлежащие точки обслуживания (POS) для заданного **BaseStore** и текст местоположения.
- **позицияПоиск** методы, которые находят близлежащие POS для заданного **BaseStore** и **ГеоТочка**, который представляет собой объект, содержащий широту и долготу.
- **получитьТочкуОбслуживания*ForName** методы, которые возвращают именованный POS для заданного **BaseStore**, с расчетом расстояния или без него.
- **получитьAllPos**, который является методом получения всех POS для заданного **BaseStore**.

-Примечание

Интерфейс имеет явную зависимость от расширения basecommerce.

The **DefaultStoreFinderService** использует **GeoWebServiceWrapper** для геокодирования и определения направления. Для получения дополнительной информации см. [GeoWebServiceWrapper](#).

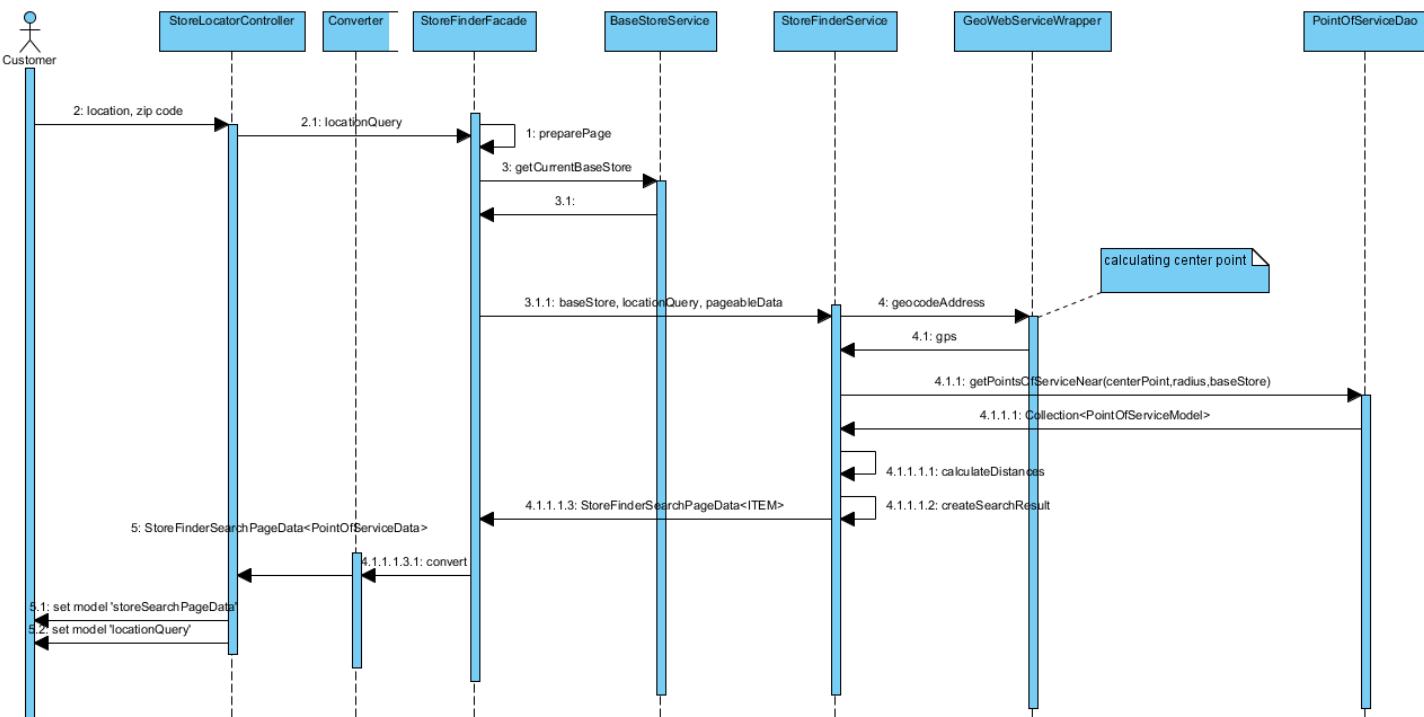
-Осторожность

Ложное геокодирование

Значение по умолчанию **DefaultStoreFinderService** использует из **GeoWebServiceWrapper** для информации о местоположении. **CommerceMockGeoWebServiceWrapper** возвращает фиксированные точки геокодирования в зависимости от страны.

Конкретные варианты использования Store Locator

В большинстве ситуаций гибрид коммерция услуги Расширение использует реализацию по умолчанию функции Store Locator из расширения basecommerce. Однако из-за перехода на Google Maps API v.3 диаграмма вариантов использования выглядит следующим образом:



Услуги I18n

В следующем разделе описываются службы I18n расширения commerce services.

Commerce Common I18n Сервис

TheCommerceCommonI18nService дополняет hybris [Интернационализация и локализация](#) возможности за счет предоставления сервисов в области витрины для получения языков по умолчанию или поддерживаемых языков, валют, поддерживаемых территорий и операционной локали для текущей исполняющей витрины.

Для получения более подробной информации см. [Расширение cms2 — техническое руководство](#).

-Примечание

Интерфейс имеет явную зависимость от расширения cms2.

Услуги по продвижению

В следующем разделе описывается услуга по продвижению торговли коммерции услуга расширение.

Служба содействия торговле

TheCommercePromotionService предоставляет методы получения информации об определенных акциях.

Службы директивы по слиянию

TheListMergeDirective и MapMergeDirective обеспечивают больший контроль над списками и картами bean-компонентов Spring.

-Примечание

Когда ListMergeDirective и MapMergeDirective были перемещены из коммерции услуг в ядро платформы это изменение включало следующие изменения:

- Пакет для занятий изменился cde.hybris.platform.commerceservices.spring.config к de.hybris.platform.spring.config.
- Занятия bde.hybris.platform.commerceservices.spring.config теперь устарели, поэтому используйте de.hybris.platform.spring.config вместо этого упакуйте.
- Весь код, который использовался listMergeDirective или mapMergeDirective будет работать нормально. Если listMergeBeanPostProcessor или mapMergeBeanPostProcessor был объявлен в вашем контексте, обновите определение, чтобы использовать классы из ядра платформы.

СписокОбъединить

TheListMergeDirective позволяет AddOn объединять дополнительные элементы в списки bean-компонентов Spring и свойства списка bean-компонентов Spring. Минимальное свойство для установки — add. Директива также поддерживает возможность вставки bean-компонента до или после указанного элемента списка bean-определения или класса bean-компонента. ListMergeDirective bean definitions также должны включать зависит - от qualier, который должен быть listbean или bean, охватывающим свойство list, следующим образом:

```
<bean id="listMergeBeanPostProcessor" class="de.hybris.platform.spring.config.ListMergeDirectiveBeanPostProcessor" /> <bean id="listMergeDirective" class="de.hybris.platform.spring.config.ListMergeDirective" abstract="true" />

<!-- Этот фрагмент добавляет компонент в конец целевого списка -->
<bean id="addToListMergeDirective" зависит от="targetListBean" parent="listMergeDirective">
    <имя свойства="add" ref="MyBeanToInsertAtEndOfList" />
</боб>

<!-- Этот фрагмент вставляет компонент после всех компонентов заданного типа и перед всеми компонентами другого типа --> <bean id="insertBeforeAfterListMergeDirective" depends-on="targetListBean" parent="listMergeDirective">
    <имя свойства="add" ref="myBeanToInsert" /> <имя свойства="afterClasses">
        <list value-type="java.lang.Class">
            <value>полный.пакет.класса.ToInsertAfter</value>
        </ список >
    </ свойство >
    <имя свойства="beforeClasses">
        <list value-type="java.lang.Class">
            <value>полный.пакет.класса.ToInsertBefore</value>
        </ список >
    </ свойство >
</боб>
```

Компоненты также можно вставлять перед другими компонентами или после них, ссылаясь на них по имени, как показано ниже:

```
<bean id="insertAfterBeanListMergeDirective" зависит от="targetListBean" parent="listMergeDirective">
    <имя свойства="add" ref="myBeanToInsert" /> <имя свойства="afterBeanNames">
        <list value-type="java.lang.String">
            <value>beanToInsertAfter</value>
        </ список >
    </ свойство >
</боб>
```

Конкретный список может быть выбран для вставки либо по доступу к свойству, либо по доступу к полю следующим образом:

```
<bean id="myBeanWithListProperty" class="BeanWithList">
    <имя свойства="myListOfBeans">
        < список >
            <реф bean="oneBean">
            <реф bean="дваБоба">
        </ список >
    </ имя свойства >
</боб>
```

```

<реф bean="redBean">
<реф боб="blueBean">
</список>
</боб>

<bean id="addBeanToListPropertyListMergeDirective" зависит от="myBeanWithListProperty" родительский="listMergeDirective">
    <имя свойства="add" ref="myBeanToAdd" />
    <имя_свойства="listPropertyDescriptor" значение="myListOfBeans" />
</боб>

```

Доступ к полю лучше всего использовать, когда изменяемый список не имеет публичного геттера, как показано ниже:

```

<bean id="addBeanToListFieldListMergeDirective" зависит от="myBeanWithNoPublicGetterList" parent="listMergeDirective">
    <имя свойства="add" ref="myBeanToAdd" />
    <имя свойства="имя_поля" значение="myNoPublicGetterListOfBeans" />
</боб>

```

MapMerge

The MapMerge Directive очень похоже на ListMerge Directive, как можно увидеть в следующем примере:

```

<bean id="mapMergeBeanPostProcessor" class="de.hybris.platform.spring.config.MapMergeDirectiveBeanPostProcessor" /> <bean id="mapMergeDirective"
class="de.hybris.platform.spring.config.MapMergeDirective" abstract="true" />

<bean id="myMapMergeDirective" зависит от="myBeanWithMap" parent="mapMergeDirective">
    <имя свойства="ключ" значение="myMapKey" /> <имя
    свойства="значение" значение="myMapValue" />
</боб>

```

Аналогично, свойства для доступа к картам могут быть использованы либо с помощью `mapPropertyDescriptor`.

Модель данных

The коммерция расширение добавляет ряд модификаций модели данных для добавления дальнейшей поддержки коммерции B2C к существующим, более общим расширениям. Это расширение также может объединять модели данных из нескольких расширений для включения дополнительных функций коммерции B2C.

Продукт

В следующих разделах описывается Продукт сорт.

<<core>> Product
<<commerceservices>> - galleryImages : MediaContainerList
<<commerceservices>> - summary : localized:String
+getGalleryImages() : MediaContainerList
+setGalleryImages(galleryImages : MediaContainerList) : void
+getSummary() : localized:String
+setSummary(summary : localized:String) : void

Маркетинговая цель **краткое содержание** Атрибут был добавлен к продукту, чтобы дополнить существующий атрибут `name` и `large description`. Его можно использовать, когда требуется более краткое описание продукта.

The **галерея изображений** Атрибут используется для хранения нескольких изображений, каждое из которых изменено в соответствии с несколькими стандартными форматами, ожидаемыми витриной. Список **Медиаконтейнер** Для моделирования галереи используются объекты.

Использование медиаконтейнеров для галерей

Медиаконтейнеры — это полезный способ группировки связанных изображений. The коммерция расширение обновляет **Продукт** модель данных для поддержки галерей, которая позволяет хранить несколько групп изображений. Каждая группа изображений использует одно и то же изображение, размер которого заранее изменен до размеров, которые, как ожидается, будут отображаться на витрине, например, миниатюра, детализация или масштабирование. Это позволяет витрине предоставлять ссылку на изображение, размер которого правильно оптимизирован, и не требует изменения размера на стороне браузера или клиента. Это также гарантирует, что пропускная способность не будет излишне расходоваться при загрузке изображений в клиентский браузер. Чтобы галереи работали хорошо и выглядели хорошо, каждое изображение должно быть изменено до набора стандартных размеров и форматов.

hybris использует Medias для хранения информации об изображении, такой как его местоположение на веб-сервере, а Media имеет атрибут типа Media Format, который можно использовать для логического определения размеров изображения.

Например, формат «миниатюры» может быть 60x40 пикселей, а формат «увеличения» — 1000x700 пикселей.

Основная функция медиаконтейнера — группировать связанные медиаданные с некоторой дополнительной контекстной информацией о связанных медиаданных.

Атрибут `Gallery Images` моделируется как `List of Media Containers`. Если нам нужно сгруппировать все связанные медиа, которые отличаются только форматом (например, разными размерами), можно использовать `Media Containers`. `Media Containers` также поддерживают версии каталогов, что означает, что изменения можно загружать в `Staged Product Catalog`, а затем публиковать онлайн.

Для получения более подробной информации см. [Обработка медиаданных с помощью MediaContainer](#) и [Управление медиа в hybris Product Cockpit](#).

Отзывы клиентов

Модель данных расширения CustomerReviews была расширена для поддержки процесса утверждения и дополнительных функций.

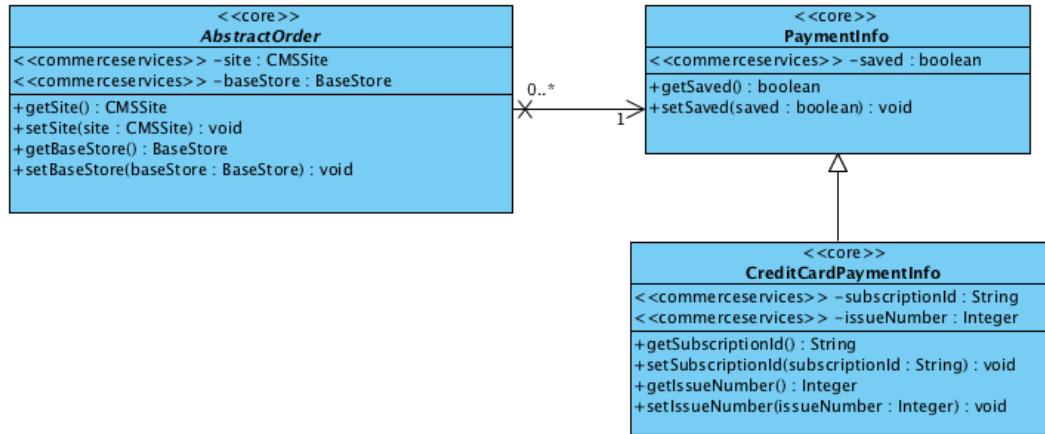
Для получения более подробной информации см.[Отзывы клиентов](#).

Заказ

В следующих разделах описывается Заказ сорт.

Отслеживание сайта, на котором был сделан заказ

The Аннотация `Zakaz` Модель данных изменена для добавления **Сайт (Базовый Сайт)** и **Магазин (BaseStore)**. **CommerceCartFactory** занимается настройкой корзины с текущим `BaseSite` и `Базовый Склад`. Обычный процесс клонирования переносит эти настройки в созданный `Заказ`.



В сценарии с несколькими витринами это может определить, в какой витрине был размещен заказ, для целей отчетности и фильтрации. В `CsCockpit` агент делает выбор относительно того, какой `BaseSite` работать в рамках, ограничивая соответствующим образом продукты и цены.

Купить онлайн Забрать в магазине Поддержка

А Точка обслуживания теперь ассоциируется с Аннотацией `Zakaz` ввода, и может использоваться для записи заказов на сбор для пункта обслуживания. Это также добавляется к консигнации для поддержки разделения заказа.

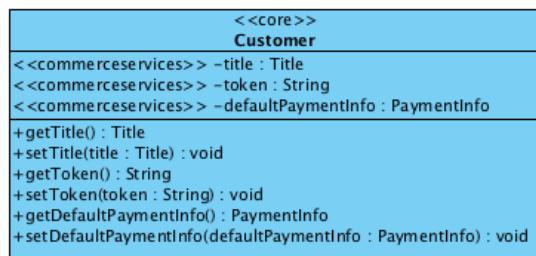
Оплата

Можно ли агент **Информация об оплате** следует использовать как сохраненный платеж для клиента. Номер выпуска также существует для поддержки подмножества кредитных карт Switch и Maestro.

Для получения более подробной информации см.[Продление платежа](#).

Клиент

В этом разделе описывается **Клиент** сорт.

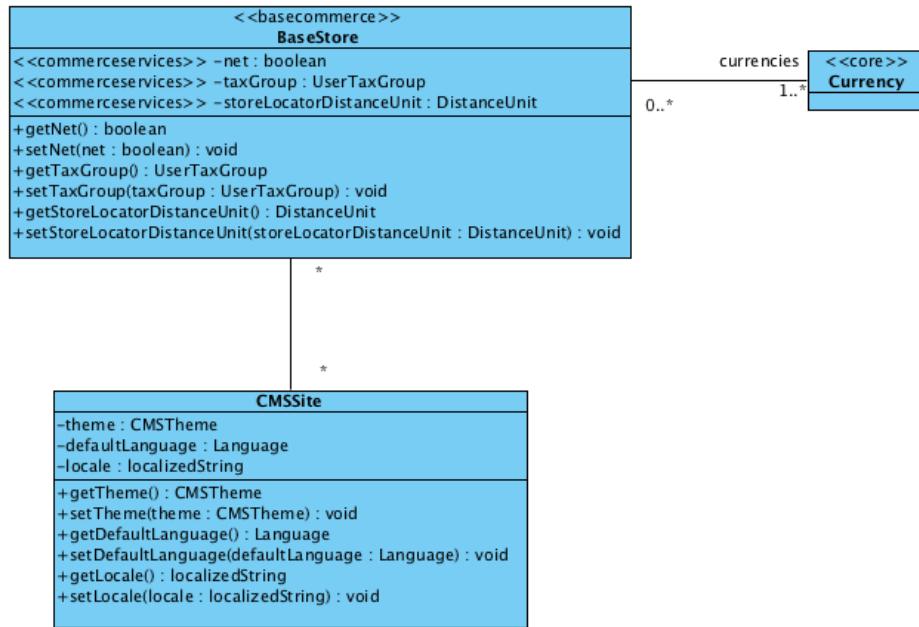


The `Клиент` Класс имеет следующие особенности:

- Класс включает атрибут `title`.
- Платежная информация клиента по умолчанию может храниться в памяти клиента.
- Специальный атрибут токена поддерживает возможность аутентификации определенных запросов учетной записи клиента, которым потребуется тот же токен, который будет передан обратно от клиента в качестве параметра запроса. Функциональность забытого пароля, описанная в [Обслуживание счетов клиентов](#) раздел выше, использует это для хранения защищенного токена, который может быть проверен только один раз.

Настройка витрины

Добавлен ряд дополнительных атрибутов `BaseStore` и `CMSsite` для фиксации настроек конфигурации для каждой витрины.



Финансовые настройки BaseStore

The **BaseStore** financial parameters include the following:

- Чистый аг существует для определения того, рассчитываются ли цены магазина с учетом или без учета налога с продаж. Это используется логикой расчета цены. Список валют, поддерживаемых магазином, фиксируется в магазине.
- Группа налогов пользователей из расширения europe1 сохраняется, чтобы логика расчета цен могла выбирать соответствующие ставки налога с продаж для конкретного магазина в конфигурации с несколькими витринами.

Настройки локализации CMSSite

The **CMSSite** localization settings include the following:

- Язык витрины по умолчанию настраивается. Это особенно актуально, если контент на конкретной витрине представлен на нескольких языках. Полный список доступных языков берется из активной версии каталога контента.
- Язык Java, используемый для каждого языка, настроен таким образом, чтобы обеспечить правильное форматирование чисел и информации о валютах на витрине магазина.

Настройки представления CMSSite

Существует атрибут темы, позволяющий изменять внешний вид и стиль Storefront.

Для получения более подробной информации см. [Моделирование витрин и каталогов](#).

Solr Фасетный поиск

В модель данных был внесен ряд дополнений для поддержки более расширенной конфигурации B2C SOLR «из коробки».

Для получения более подробной информации см. [Поиск и навигация](#).

Поиск магазина

Модель данных **Store Locator** была расширена и теперь включает больше атрибутов, специфичных для традиционных магазинов.

Для получения более подробной информации см. [Конфигурация локатора магазинов](#).

Конфигурация SetupSyncJobService

Можно настроить **de.hybris.platform.commerceservices.setup.impl.DefaultSetupSyncJobService** с использованием конфигурации Spring. Вы можете настроить **rootTypes** для обоих **ProductCatalogSyncJob** и **ContentCatalogSyncJob**, а также атрибуты **СинхронизацияАтрибутДескрипторКонфигурации** для данных предметов.

Следующие атрибуты поддерживаются с использованием РедактироватьСинхронизациюАтрибутДескрипторДанные:

- включитьInSync**: если установлено значение `ЛОЖЬ`, атрибут исключен из синхронизации
- копироватьПоЗначению**: если установлено значение `истинный`, атрибут копируется по значению
- непереводимый**: если установлено значение `истинный`, атрибут отмечен как непереводимый для синхронизации

В следующем примере показано использование этих атрибутов:

```
<entry key="de.hybris.platform.category.jalo.Category">
    <список>
        <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
            <свойство имя="qualifier" значение="linkcomponents"/>
            <свойство имя="includeInSync" значение="false"/>
            <свойство имя="copyByValue" значение="true"/>
            <свойство имя="untranslatable" значение="false"/>
        </боб>
    </список>
</entry>
```

Ниже приведен фрагмент конфигурации defaultSetupSyncJobService боб из коммерция услуги-весна.xml:

```
<util:список id="contentCatalogSyncRootTypeCodes">
    <value>CMSItem</value>
    <value>CMSPrelation</value>
    <value>Медиа</value>
    <value>MediaContainer</value>
    <value>BTGItem</value>
</util:список>
<util:list id="productCatalogSyncRootTypeCodes"/>
<util:map id="contentCatalogEditSyncDescriptors" key-type="java.lang.Class" value-type="java.util.List"/> <util:map
id="productCatalogEditSyncDescriptors" key-type="java.lang.Class" value-type="java.util.List">
    <entry key="de.hybris.platform.jalo.Item">
        <список>
            <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
                <свойство имя="includeInSync" значение="false"/>
                <свойство имя="qualifier" значение="comments"/>
            </боб>
            ...
        </список>
    </entry>
    <entry key="de.hybris.platform.jalo.product.Product">
        <список>
            <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
                <свойство имя="includeInSync" значение="false"/>
                <свойство имя="qualifier" значение="sequenceId"/>
            </боб>
            <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
                <свойство имя="includeInSync" значение="false"/> имя="qualifier"
                <свойство значение="productReviews"/>
            </боб>
            ...
        </список>
    </entry>
    <entry key="de.hybris.platform.europe1.jalo.PriceRow">
        <список>
            <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
                <свойство имя="includeInSync" значение="false"/>
                <свойство имя="qualifier" значение="sequenceId"/>
            </боб>
        </список>
    </entry>
    <entry key="de.hybris.platform.category.jalo.Category">
        <список>
            <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
                <свойство имя="includeInSync" значение="false"/> имя="qualifier"
                <свойство значение="linkcomponents"/>
            </боб>
            ...
            <bean class="de.hybris.platform.commerceservices.setup.data.EditSyncAttributeDescriptorData">
                <свойство имя="includeInSync" значение="false"/> имя="qualifier"
                <свойство значение="categoryFeatureComponents"/>
            </боб>
        </список>
    </entry>
</util:карта>

<псевдоним имя="defaultSetupSyncJobService" псевдоним="setupSyncJobService" /> <bean
    id="defaultSetupSyncJobService"
    class="de.hybris.platform.commerceservices.setup.impl.DefaultSetupSyncJobService">
        <свойство name="modelService" ref="modelService"/>
        <свойство name="contentCatalogSyncRootTypeCodes" ref="contentCatalogSyncRootTypeCodes"/>
        <свойство name="productCatalogSyncRootTypeCodes" ref="productCatalogSyncRootTypeCodes"/>
        <свойство name="contentCatalogEditSyncDescriptors" ref="contentCatalogEditSyncDescriptors"/>
        <свойство name="productCatalogEditSyncDescriptors" ref="productCatalogEditSyncDescriptors"/>
    </боб>
```

Сохранить корзину

Функция сохранения корзины в первую очередь позволяет клиенту сохранять и восстанавливать свои сохраненные корзины позднее. Эта функция предоставляется как набор методов, встроенных в независимые стратегии, которые могут быть подключены относительно различных бизнес-требований и реализаций front-end, для которых они используются. Эти стратегии можно легко расширить с помощью pre/post-hooks.

В настоящее время поддерживаются следующие операции сохранения корзины:

- Сохранить корзину сеанса как сохраненную корзину
- Сохраните определенные идентификаторы корзин для внутренних операций в качестве сохраненных корзин
- Отобразить список сохраненных корзин
- Отображение деталей сохраненной корзины
- Восстановить сохраненную корзину в активную корзину сеанса

- Удалить сохраненные корзины
- Клонировать сохраненные корзины
- Отредактируйте название и описание сохраненной корзины.

Для поддержки этой функциональности был реализован новый интерфейс CommerceSaveCartService. Этот интерфейс содержит следующие методы:

Метод сигнатуры	Возвраты
сохранитьCart(CommerceSaveCartParameterData)	CommerceSaveCartResultData
agForDeletion(CommerceSaveCartParameter)	CommerceSaveCartResultData
getCartForCodeAndCurrentUser(CommerceSaveCartParameterData)	CartData
restoreSavedCart(CommerceSaveCartParameterData)	КорзинаВосстановлениеДанные
getSavedCartsForCurrentUser(PageableData, List<OrderStatus>)	SearchPageData<CartData>
cloneSavedCart(CommerceSaveCartParameterData)	CommerceSaveCartResultData

Также был представлен новый интерфейс SaveCartDao. Этот интерфейс содержит следующие методы:

Метод сигнатуры	Возвраты
получитьСохраненныеКартыДляУдаленияДляСайта(БазоваяМодельСайта)	Список<CartModel>
getSavedCartsForSiteAndUser(PageableData, BaseSiteModel, UserModel, Список <СтатусЗаказа>)	SearchPageData<CartModel>

Помимо реализации SaveCartDao, ваша пользовательская реализация этого интерфейса должна расширять реализацию интерфейса CommerceCartDao для поддержки операций над объектами Cart. Другими словами, если ваши реализации интерфейсов CommerceCartDao и SaveCartDao называются DefaultCommerceCartDao и DefaultSaveCartDao соответственно, заголовок класса выглядит следующим образом:

DefaultSaveCartDao

открытый класс DefaultSaveCartDao расширяет DefaultCommerceCartDao реализует SaveCartDao

СохранитьCartStrategy

Класс по умолчанию, расширяющий класс AbstractCommerceSaveCartStrategy, класс DefaultCommerceSaveCartStrategy, обрабатывает функциональность сохранения корзины для продуктов-бандов. Этот класс предоставляет метод calculateExpirationTime(), который вычисляет время, через которое истекает срок действия сохраненной корзины, на основе даты сохранения и свойства commerceservices.saveCart.expiryTime.days. Вы можете изменить время истечения срока действия по умолчанию в **свойства проекта**:

свойства проекта

```
# Определяет, через сколько дней истекает срок действия сохраненной
корзины commerceservices.saveCart.expiryTime.days=30
```

ФлагДляСтратегииУдаления

Класс по умолчанию, расширяющий класс AbstractFlagForDeletionStrategy, класс DefaultCommerceFlagForDeletionStrategy, предоставляет метод agForDeletion(), который устанавливает все атрибуты сохраненной корзины в значение null:

DefaultCommerceFlagForDeletionStrategy

```
окончательная CartModel cartToBeFlagged = параметры.getCart();

cartToBeFlagged.setExpirationTime(null);
cartToBeFlagged.setSaveTime(null);
cartToBeFlagged.setSavedBy(null);
cartToBeFlagged.setName(null);
cartToBeFlagged.setDescription(null);
```

Фактическое удаление корзин выполняется CartRemovalJob Cronjob. Для получения дополнительной информации см. раздел Save Cart Support [Расширение acceleratorservices — техническое руководство](#) документ.

-Примечание

Для получения дополнительной информации о заданиях Cron см. [cronjob - Техническое руководство](#) документ.

СохранитьКорзинаВосстановлениеСтратегия

Класс по умолчанию, расширяющий класс DefaultCommerceCartRestorationStrategy, класс DefaultCommerceSaveCartRestorationStrategy, обрабатывает восстановление сохраненных корзин. При вызове метода restoreCart() сохраненная корзина, переданная в качестве параметра, устанавливается как корзина активного сеанса, что позволяет клиенту оформить заказ на продукты из корзины.

Реализация интерфейса SaveCartDao по умолчанию использует запросы FlexibleSearch для получения списка сохраненных корзин.

- получение списка сохраненных корзин пользователя с определенным статусом

DefaultSaveCartDao

```
защищенная конечная статическая строка FIND_SAVED_CARTS_FOR_SITE_AND_USER_WITH_STATUS = "SELECT {" + CartModel.PK + "} FROM {" + CartModel._TYPECODE + "}, {" + OrderStatus._TYPECODE + "} " + "ГДЕ {" + CartModel._TYPECODE + ".+" + C + "} = {" + OrderStatus._TYPECODE + ".pk} И {" + CartModel.USER + "} = ?user И {" + CartModel.SITE + "} = ?site И {" + CartModel.SAVETIME + "} НЕ NULL И {OrderStatus.CODE} в (?orderStatus) " + ИЛИ
```

- получение списка сохраненных корзин с истекшим сроком действия

DefaultSaveCartDao

```
защищенная конечная статическая строка FIND_EXPIRED_SAVED_CARTS_FOR_SITE = SELECTCLAUSE + "WHERE {" + CartModel.SITE + "} = ?site И {" + CartModel.SAVETIME + "} НЕ ЯВЛЯЕТСЯ NULL И {" + CartModel.EXPIRATIONTIME + "} <= ?cur + ORDERBYCLAUSE;
```

- получение списка сохраненных корзин пользователя

```
защищенная конечная статическая строка FIND_SAVED_CARTS_FOR_USER_AND_SITE = SELECTCLAUSE + "WHERE {" + CartModel.USER + "} = ?пользователь И {" + CartModel.САЙТ + "} = ?сайт И {" + CartModel.COХРАНЕНИЕВРЕМЕНИ + "} НЕ НУЛЕВОЙ " + ЗАКАЗ
```

Конфигурация бобов

В commerce/services/resources/commerceservices-beans.xmlie, вы можете найти определение bean-компонентов CommerceSaveCartParameter и CommerceSaveCartResult:

commerceservices-beans.xml

```
<бобы>
..
<bean class="de.hybris.platform.commerceservices.service.data.CommerceSaveCartParameter">
    <имя свойства="cart" тип="de.hybris.platform.core.model.order.CartModel">
        <description>CartModel, который нужно сохранить</description> </property>
    <имя свойства="имя" тип="Строка">
        <description>Название сохраненной корзины, предоставленное пользователем или сгенерированное автоматически</description> </property>
    <имя свойства="описание" тип="Строка">
        <description>Описание сохраненной корзины, предоставленное пользователем или сгенерированное автоматически</description> </property>
    <имя свойства="enableHooks" тип="логическое значение">
        <description>Должны ли выполняться методы-хуки</description> </property>
    </боб>
<bean class="de.hybris.platform.commerceservices.service.data.CommerceSaveCartResult">
    <имя свойства="savedCart" тип="de.hybris.platform.core.model.order.CartModel">
        <description>CartModel, который был сохранен</description> </property>
    </боб>
</бобов>
```

Исключения

В коммерческую услугу расширение, класс CommerceSaveCartException определен. Это простое исключение выбрасывается, если корзину невозможно извлечь, сохранить или отменить сохранение.

CommerceSaveCartException.java

```
открытый класс CommerceSaveCartException расширяет BusinessException {
    public CommerceSaveCartException(финальное строковое сообщение) {
        супер(сообщение);
    }
    public CommerceSaveCartException(final String message, final Throwable cause) {
        супер(сообщение, причина);
    }
}
```

Структура списков клиентов

The Структура списка клиентов — это общая структура, которая может отображать различные списки клиентов для агентов поддержки клиентов относительно различных бизнес-приложений. Записи клиентов в списках клиентов содержат глубокие ссылки ASM, которые при нажатии на них CSA перенаправляют их на соответствующие страницы, например, на страницу профиля клиента, корзины или заказов. Это позволяет CSA подготовиться с использованием информации о клиенте, прежде чем предоставлять ему персонализированную поддержку. В настоящее время доступны следующие реализации:

Имя в списке клиентов	Описание	Ссылка на документацию
Текущие клиенты магазина	<p>Отображает список клиентов, которые в данный момент находятся в магазине или в разных местах внутри магазина, по данным одного или нескольких аппаратных устройств, чтобы агент службы поддержки клиентов мог просмотреть профиль клиента и подойти к нему подготовленным.</p> <p>По умолчанию данные клиентов имитируются, однако в реальной реализации эти данные будут поступать с одного или нескольких физических устройств Интернета вещей, находящихся в магазине.</p>	assistedserviceservices Расширение документ, раздел Текущие клиенты в магазине
Мои последние сеансы с клиентами	Отображает список клиентов, сеансы которых были недавно начаты агентом службы поддержки клиентов, чтобы агент службы поддержки клиентов мог быстро вернуться к предыдущему сеансу пользователя службы поддержки клиентов.	вспомогательныйсервисфасады Расширение документ, раздел Мои последние сеансы клиентов
Самовывоз из магазина для клиентов	Отображает список клиентов, совершивших покупку онлайн и забирающих свои заказы в магазине, чтобы сотрудник службы поддержки клиентов мог подготовить их заказ.	assistedserviceservices Расширение документ, раздел «Забор товара в магазине для клиентов»

Обзор структуры списков клиентов

Для представления связи между реализацией стратегии и списком клиентов вводится карта, содержащая связь между реализацией **ТипСтрока** и стратегиями реализации:

```
<util:map id="customerListSearchStrategyMap" key-type="java.lang.String" value-type="de.hybris.platform.commerceservices.customer.stra
```

Используя в качестве примера список клиентов «Недавние сеансы», посмотрите, как стратегия сопоставляется с фактическим списком `bassistedserviceservices-spring.xml`:

```
<bean id="defaultRecentlyStartedSessionCustomerListSearchStrategy"
      class="de.hybris.platform.assistedserviceservices.strategy.DefaultRecentlyStartedSessionCustomerListSearchStrategy"> <property
        имя="customerSupportEventService" ref="customerSupportEventService"/> имя="userService"
        <свойство ref="userService"/>
    </боб>

    <bean id="defaultRecentlyStartedSessionCustomerListMergeDirective" зависит от="customerListSearchStrategyMap" parent="mapMerge
        <имя свойства="ключ" значение="ASM_RECENT_SESSIONS"/>
        <имя_свойства="значение" ref="по умолчаниюНедавноНачатаяСессияСписокКлиентовСтратегияПоиска" />
    </боб>
```

The `CustomerListService` Интерфейс отвечает за возврат всех действительных списков клиентов, доступных для конкретного сотрудника, но это `СписокКлиентовПоискСтратегия` класс, в котором реализуется фактическая бизнес-логика.

-Примечание

Если вы хотите ввести новые списки клиентов для различных бизнес-приложений, вам придется реализовать новые стратегии для каждого модуля.

Ниже вы можете найти список примеров реализаций `CustomerListStrategy` интерфейса:

Выполнение	Описание
<code>de.hybris.platform.assistedserviceservices.strategy.DefaultНедавно начатый сеансСписок клиентовСтратегия поиска</code>	Список клиентов, которым недавно помогал текущий агент, максимум 20 клиентов. Для получения дополнительной информации см. раздел Мои последние сеансы клиентов . вспомогательныйсервисфасады Расширение документ.
<code>de.hybris.platform.assistedserviceservices.strategy.DefaultInStoreCustomerListSearchStrategy</code>	Список всех клиентов, зарегистрированных в определенной точке обслуживания (POS). Для получения дополнительной информации см. раздел Текущие клиенты в магазине assistedserviceservices Расширение документ и раздел <code>points-ofservice.impex</code> Надстройка витрины вспомогательного сервиса документ.
<code>de.hybris.platform.assistedserviceservices.strategy.DefaultBopisCustomerListSearchStrategy</code>	Список всех клиентов, которые забирают то, что они купили онлайн в определенной точке обслуживания. Для получения дополнительной информации

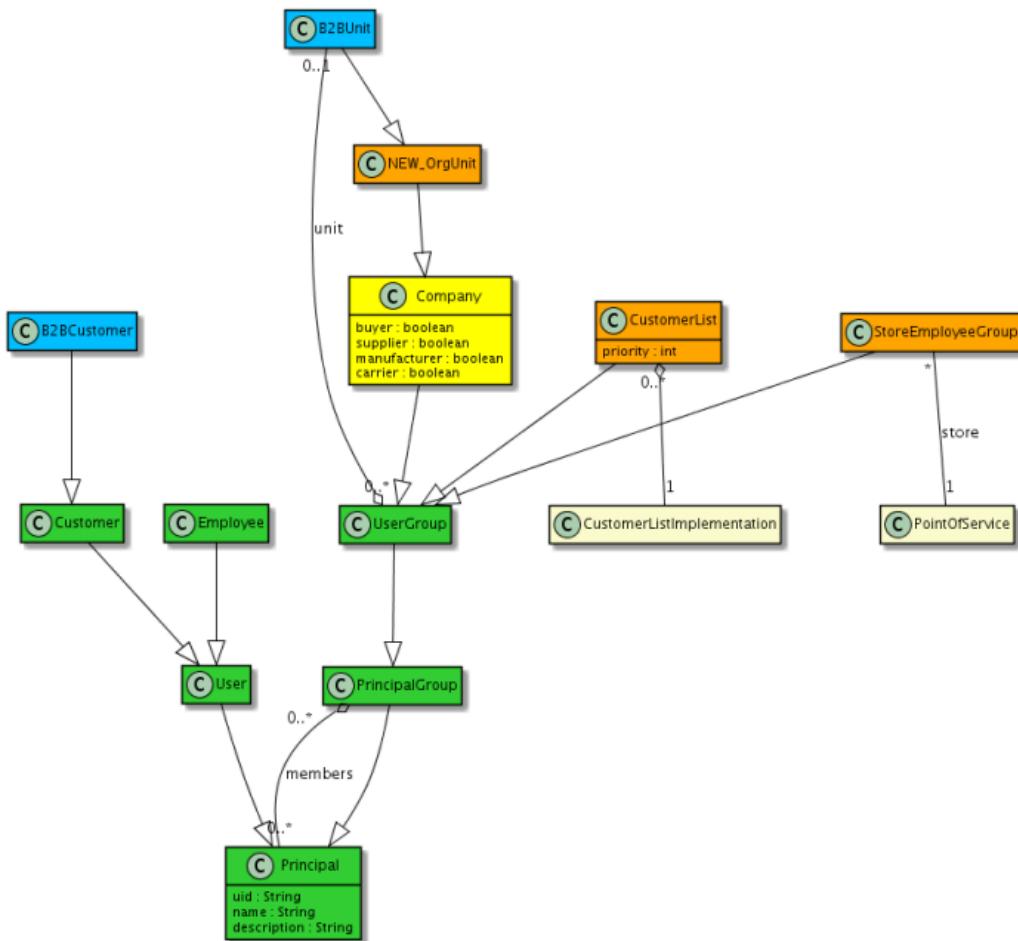
Выполнение	Описание
	<p>Информацию см. в разделе «Забор товара из магазина».</p> <p>assistedserviceservices Расширение</p> <p>и раздел points-of-service.impex</p> <p>Надстройка витрины вспомогательного сервиса документ.</p>
de.hybris.platform.b2bcommerce.services.DefaultB2BCustomerListSearchStrategy	Перечислите всех клиентов B2B, закрепленных за общим подразделением B2B.

Модель данных

Ниже вы можете найти UML-диаграмму модели данных Customer Lists Framework относительнокоммерцияуслугирасширение:

-Примечание

core (#LimeGreen)
catalog (#Yellow)
basecommerce (#LightYellow)
commerceservices (#Orange)
acceleratorservices (#Purple)
yacceleratorcore (#Plum)
assistedserviceservices (#Wheat)
b2bcommerce (#DeepSkyBlue)



Тип списка клиентов

The Список клиентов тип генерируется на основе определения, найденного в commerceServices-items.xml файле:

```
<typegroup name="СписокКлиентов">
    <itemtype code="CustomerList" autocreate="true" generate="true" extends="UserGroup">
        <description>Список клиентов виден определенным сотрудникам и представляет собой поисковый запрос, специфичный для реализации. к <attributes>
```

```

<квалификатор атрибута="implementationType" type="java.lang.String">
    <description>Тип реализации для этого списка клиентов</description> <persistence type="property">
        />
    </атрибут>
    <квалификатор атрибута="приоритет"      тип="java.lang.Integer">
        <description>приоритет для списка клиентов и ноль по умолчанию, это повлияет на позицию <persistence
        type="property" />
        <defaultValue>Целое число.valueOf(0)</defaultValue> </attribute>
    </атрибуты>
</тип_элемента>
</typegroup>

```

СписокПоискСтратегияКарта

TheСписокПоискСтратегияКартабean — это карта, содержащая различные реализации фреймворка Customer Lists. Эта карта определена в commerceservices-весна.xmlе:

```

<util:map id="customerListSearchStrategyMap" key-type="java.lang.String"
    value-type="de.hybris.platform.commerceservices.customer.strategies.CustomerListSearchStrategy" область действия

```

TheСписокПоискСтратегияКартабean принимает строковые значения в качестве ключей и фактические реализации bean в качестве значений.

-Примечание

Если вы создаете новый список клиентов, обязательно укажите его на этой карте.

CustomerListService

TheCustomerListServiceИнтерфейс отвечает за возврат списка списков клиентов для данного сотрудника и за возврат определенного списка для определенного сотрудника. Эта реализация по умолчанию этого интерфейса упоминается в Spring

CustomerListSearchService

TheCustomerListSearchServiceИнтерфейс отвечает за возврат постраничных клиентов для заданного списка клиентов.

СписокКлиентовПоискСтратегия

TheСписокКлиентовПоискСтратегиякласс — это интерфейс, который вам нужно реализовать в вашем списке клиентов.getPagedCustomers()Метод — это то место, где вам нужно разместить вашу бизнес-логику, которая возвращает реальный список клиентов. Если вы хотите посмотреть, как может выглядеть такая реализация, следующие классы реализуют СписокКлиентовПоискСтратегияинтерфейс:

Сорт	Доступная документация
DefaultInStoreCustomerListSearchStrategy	Раздел «Покупатели в магазине» assistedserviceservices Расширение
По умолчаниюНедавноНачавшийсяСессияСписокКлиентовСтратегияПоиска	Раздел «Недавние сеансы» assistedserviceservices Расширение

Для получения более подробной информации о Структура списков клиентов, см.[Расширение commercefacades](#) документ.

Пагинация и сортировка

Следующие объекты, связанные с нумерацией страниц и сортировкой в коммерцииуслугиРасширение устарело:

- Сортировать данные
- SearchPageData
- ПагинацияДанные
- PagableData

Заменяемые объекты можно найти в платформарасширение.

-Примечание

Не все атрибуты устаревших объектов были перенесены в новые объекты, а некоторые новые объекты данных не имеют никаких поддерживающих служб.

Для получения более подробной информации см.[Пагинация и сортировка](#).

Будущее фондового обслуживания

Служба будущих запасов предоставляет функциональные возможности получения информации о будущей доступности продукта, такой как уровень запасов и дата доступности для указанных продуктов.

- Параметр: Список продуктов<ProductModel>. Этот параметр представляет собой список моделей продукта.
- Возврат: Кarta<Строка, Кара<Дата, Целое число>>. Возвращает карту кодов продуктов с соответствующей картой будущей доступности на складе.
- Если в будущем эти продукты недоступны, возвращается пустая карта.

Сопутствующая информация

[Документация SAP Commerce Accelerator](#)

Расширение commercefacades

The коммерция фасады расширение предоставляет набор фасадов, которые составляют единый многоканальный API витрины, который может использоваться несколькими фронтендами. Фасад отвечает за интеграцию существующих бизнес-сервисов из всего спектра расширений SAP Commerce и предоставление ответа объекта данных (POJO), настроенного для соответствия требованиям витрины.

-Примечание

Расширение SAP Commerce может предоставлять функциональность, лицензируемую через различные модули SAP Commerce. Обязательно ограничьте реализацию функциями, определенными в вашей контрактной лицензии. В случае сомнений обратитесь к своему торговому представителю.

Разрушение фасада

Фасадный Бин

Фасад в коммерция фасады расширение — это управляемый компонент Spring, который представляет собой реализацию шаблона проектирования программного обеспечения, абстрагирующего базовую реализацию, отвечающую за обслуживание действия витрины, путем предоставления упрощенного интерфейса, оптимизированного для витрины.

В целом фасады обычно интегрируют один или несколько вызовов методов SAP Commerce ServiceLayer, которые при объединении выполняют некоторую форму бизнес-действия, запрошенному пользователем витрины. Однако интерфейс фасада полностью независим от модели ServiceLayer, поэтому можно полностью заменить базовую реализацию фасада.

Примерами таких деловых действий являются:

- Просмотр сведений о продукте.
- Добавление товара в корзину.
- Добавление адреса доставки при оформлении заказа.
- Публикация отзыва.
- Поиск товаров с помощью бесплатного текстового поиска.
- Очистка или сортировка результата поиска.

Объекты данных

Фасады расширения hybris commercefacades в основном возвращают объекты данных. Они предназначены для использования на фронтенде, например, на витринах Accelerator, или для отображения в каком-либо представлении XML или JSON. Они заполняются с использованием подмножества данных, содержащихся в моделях ServiceLayer hybris, и не поддерживаются слоем сохранения, например объектами Model. Объект данных также может быть создан из нескольких моделей, а также из данных, полученных после выполнения бизнес-логики в службах. В витрине Accelerator эти объекты данных являются частью модели, предоставляемой для построения представления в шаблоне MVC по умолчанию. Модели ServiceLayer hybris не должны использоваться как часть интерфейса фасада, поддерживая чистую абстракцию бизнес-слоя и слоя представления.

Пример исходного кода

Мы можем обратиться к методу ProductFacade getProductByCode для достаточно сложного примера. Интерфейс фасада принимает номер артикула продукта вместе с набором опций данных в качестве инструкции от front-end относительно уровня данных, которые необходимо вернуть в результирующем объекте ProductData. В примере используются Converter, Populators, Congurable Populators и Services.

```
ProductData getProductForCode(код строки, задать параметры <ProductOption>)
{
    getProducConfiguredPopulator().заполнить(productModel,           Данные о продукте,   параметры);
}

вернуть данные о продукте;
}
```

Реализация в DefaultProductFacade выглядит следующим образом:

```
@Переопределить
public ProductData getProductForCode(конечный код строки, конечный набор параметров <ProductOption>) {
    окончательная ProductModel productModel = getProductService().getProductForCode(code); окончательные
    ProductData productData = getProductConverter().convert(productModel);
```

```

если (опции != null) {
    getProductConfiguredPopulator().заполнить(productModel,           Данные о продукте, параметры);
}

вернуть данные о продукте;
}

```

Сначала он вызывает ProductService, чтобы получить модель продукта для предоставленного номера артикула. Затем настроенный bean-компонент productConverter используется для создания прототипа ProductData, а ProductPopulator используется для заполнения объекта базовыми данными.

```

@Override// Из ProductPopulator @Override
защищенные ProductData createTarget(конечный источник ProductModel) {
    вернуть новый ProductData();
}

// Из ProductConverter @Override
public void populate(конечный источник ProductModel, конечный целевой ProductData) {
    getProductBasicPopulator().заполнить(источник, цель);
    getVariantSelectedPopulator().заполнить(источник, цель);
    getProductPrimaryImagePopulator().заполнить(источник, цель);

    super.populate(источник, цель);
}

```

-Кончик

Совет по расширяемости

Используйте beans.xml для создания новых атрибутов в вашем ProductData, а затем добавьте новые Populators для этих атрибутов.

ProductBasicPopulator выглядит так:

ПродуктBasicPopulator

```

@Переопределить
public void populate(final ProductModel productModel, final ProductData productData) выдает ConversionException {
    productData.setName((String) getProductAttribute(productModel, ProductModel.NAME)); productData.setManufacturer((String)
        getProductAttribute(productModel, ProductModel.MANUFACTURERNAME));

    productData.setAverageRating(productModel.getAverageRating()); если
        (productModel.getVariantType() != null)
    {
        ProductData.setVariantType(productModel.getVariantType().getCode());
    }

    productData.setPurchasable(Boolean.valueOf(productModel.getVariantType() == null && isApproved(productModel)));
}

защищенный логический isApproved(final ProductModel productModel) {

    окончательный СтатусУтвержденияСтатуса = productModel.getApprovalStatus(); return approvalStatus != null
    && СтатусУтвержденияСтатусаУтвержденияСтатуса.APPROVED.equals(статусУтвержденияСтатуса);
}

```

-Примечание

Детали реализации

Product Populators немного уникальны тем, что они обычно расширяют класс AbstractProductPopulator, который учитывает варианты и поддерживает возможность возврата к родительскому продукту вариантов для значений атрибутов в случае, если исходное значение продукта равно null.

Затем ProductFacade использует свой сконфигурированный Populator для заполнения прототипа Data Object на основе запрошенных опций данных. Сконфигурированный Populator — это просто сконфигурированный пружиной конвейер пополнителей, которые вызываются в сконфигурированном порядке и вызываются только в том случае, если они назначены указанной опции. Вот пример конфигурации пружины CongurablePopulator, используемой ProductFacade.

```

<alias name="defaultProductConfiguredPopulator" alias="productConfiguredPopulator" />
<bean id="defaultProductConfiguredPopulator" class="de.hybris.platform.commercefacades.convert.impl.DefaultConfigurablePopulator
    <имя свойства="популяторы">
        <map key-type="de.hybris.platform.commercefacades.product.data.ProductData$ProductOption">
            <entry key="ГАЛЕРЕЯ" value-ref="productGalleryImagesPopulator" /> <entry key="СВОДКА"
                value-ref="productSummaryPopulator" /> <entry key="ОПИСАНИЕ" value-
                ref="productDescriptionPopulator" /> <entry key="КАТЕГОРИИ" value-
                ref="productCategoriesPopulator" /> <entry key="АКЦИИ" value-
                ref="productPromotionsPopulator" /> <entry key="АКЦИИ" value-
                ref="productStockPopulator" />
            <entry key="REVIEW" value-ref="productReviewsPopulator" />
            <entry key="CLASSIFICATION" value-ref="productClassificationPopulator" /> <entry key="REFERENCES"
                value-ref="productReferencesPopulator" /> <entry key="VARIANT_FULL" value-
                ref="variantFullPopulator" />
        </map>
    </имя свойства="популяторы">
</bean>

```

```
</свойство>
</боб>
```

Как упоминалось выше, некоторые Populator могут использовать сервисы для заполнения прототипа Data Object, а не свойства исходной модели. Вы можете рассмотреть ProductPricePopulator как пример Populator, который использует сервис для заполнения части модели данных, а не исходного объекта напрямую.

```
@Переопределить
public void populate(final ProductModel productModel, final ProductData productData) выдает ConversionException {
    финал PriceData.PriceType priceType;
    финал Информация о цене info;
    если (CollectionUtils.isEmpty(productModel.getVariants())) {

        ТипЦены = ДанныеЦены.ТипЦены.КУПИТЬ;
        info = getCommercePriceService().getWebPriceForProduct(productModel);
    }
    еще
    {
        ТипЦены = ДанныеЦены.ТипЦены.ОТ;
        info = getCommercePriceService().getFromPriceForProduct(productModel);
    }
    если (информация != null) {

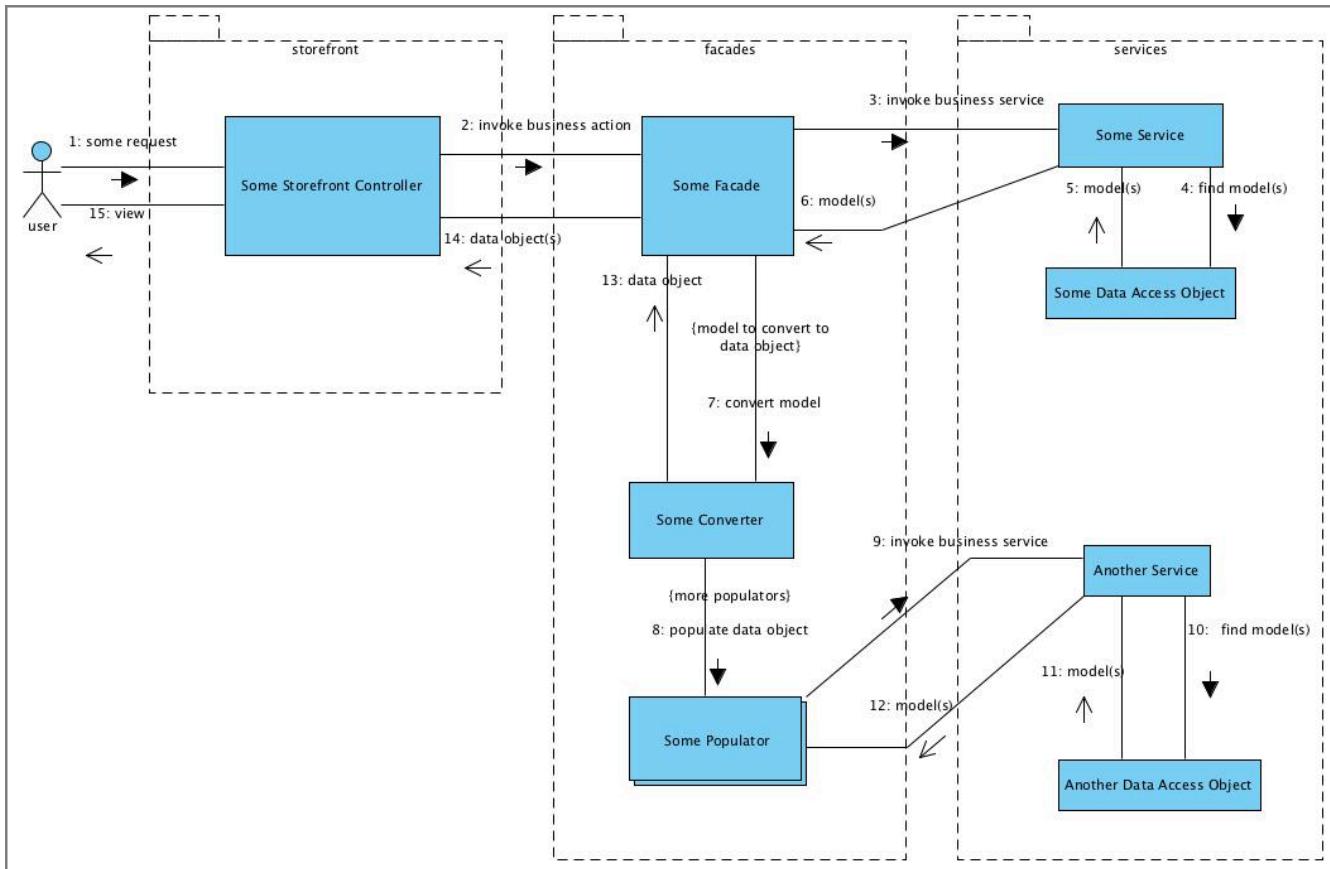
        окончательные данные о цене priceData = getPriceDataFactory().create(priceType, BigDecimal.valueOf(info.getPriceValue().getValue()
            info.getPriceValue().getCurrencyIso()));
        productData.setPrice(ценаДанные);
    }
    еще
    {
        productData.setPurchasable(Boolean.FALSE);
    }
}
```

Расширение

- Добавление новых фасадов
- Расширение существующего интерфейса фасадов и реализация для добавления дополнительных методов
- Расширение существующей реализации фасада для замены первоначальной реализации метода
- Создание другой или дополнительных версий конвертера для определенного типа
- Расширение существующей реализации конвертера
- Добавление нового Populator в конвейер Populator
- Удаление или замена реализаций Populator
- Добавление новых объектов данных и соответствующих пополнителей и преобразователей
- Расширение существующих объектов данных и добавление дополнительной логики преобразователя и/или пополнителя

Концептуальная диаграмма взаимодействия

Следующая диаграмма взаимодействия концептуально описывает, как различные компоненты взаимодействуют во время типичного запроса витрины: ProductData getProductForCode(String code, Set<ProductOption> options) выдает UnknownIdentifierException, IllegalArgumentException;



Типичное взаимодействие с магазином можно описать следующим образом:

1. Пользователь выполняет какое-либо действие, которое обрабатывается компонентом витрины, например, контроллером MVC.

2. Контроллер транслирует запрос и вызывает бизнес-действие на соответствующем фасаде.

3. Фасад вызывает одну или несколько бизнес-услуг для выполнения действия.

4-6. Скорее всего, сервису необходимо вернуть некоторый объект модели, и, возможно, он использует объекты доступа к данным для поиска моделей, чтобы вернуть их вызывающему фасаду.

7. Фасад использует преобразователи для создания объектов данных интерфейса из моделей уровня сервисов.

8. Конвертеры используют конвейер пополнителей, заполняющих скелет графа объектов данных интерфейса.

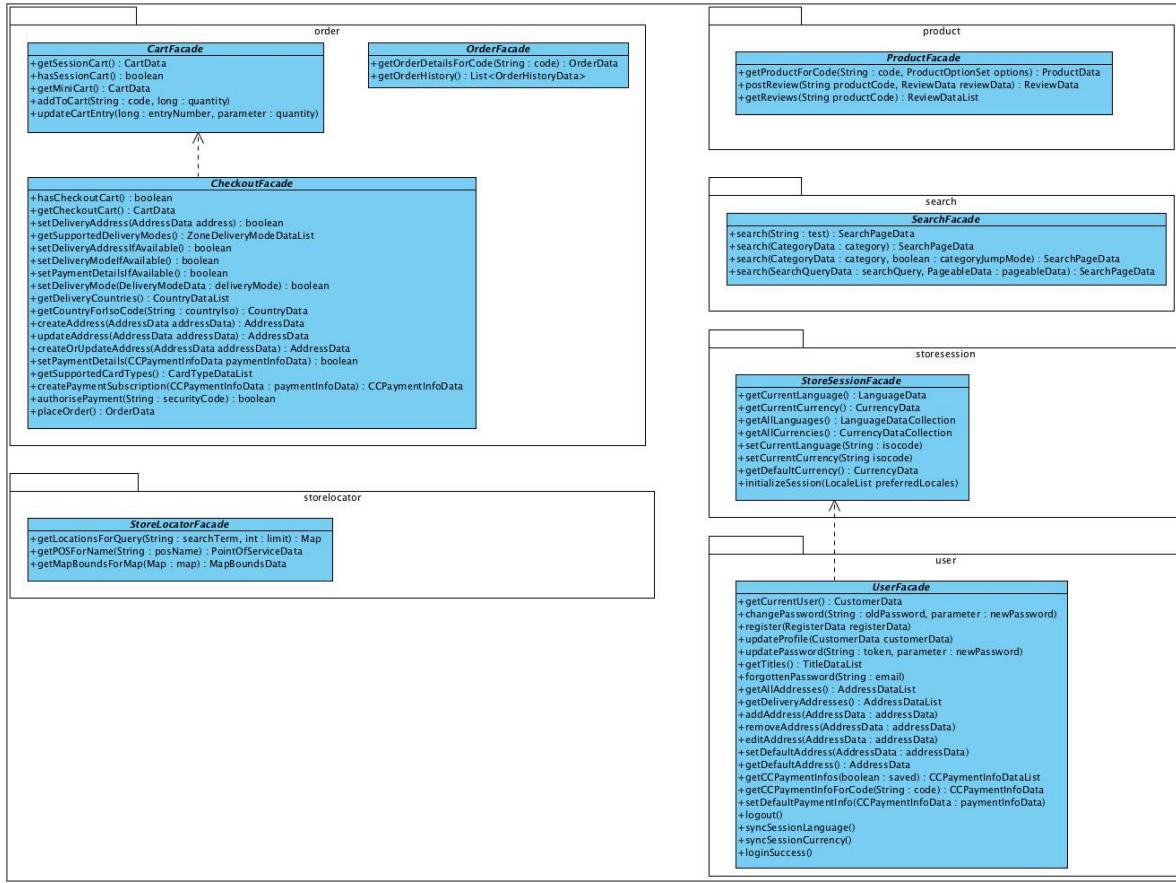
9-12. Заполнителям также может потребоваться вызвать дополнительную бизнес-логику для заполнения графа объекта данных интерфейса, поэтому они также могут вызывать службы для предоставления необходимых данных, требуемых объектом данных интерфейса.

13-14. Заполненный объект данных возвращается фасаду, а фасад затем возвращает его контроллеру.

15. Контроллер использует объект данных в качестве модели для построения представления, которое будет возвращено пользователю.

Фасады в упаковке

На следующей диаграмме классов показаны различные фасады, предлагаемые расширением commercefacades.



Поиск

Фасад поиска предоставляет возможность текстового поиска без указания продукта и возможности фасетной навигации.

Продукт

Фасад продукта предоставляет информацию о продукте. Можно контролировать объем возвращаемой информации о продукте, указав параметры данных. Фасад продукта также позволяет размещать обзоры для продуктов.

The получить `ProductForOptions()` метод:

- `DEVICE_BUNDLE_TABS` - для заполнения Devices. Раздел `BundleTabs` заполняется дополнительной информацией о тарифных планах, отсортированной по пакету и частоте.
- `SERVICE_PLAN_BUNDLE_TABS` - для заполнения планов обслуживания. Вся страница заполнена всеми существующими планами обслуживания, отсортированными по пакету и частоте.
- `SERVICE_ADDON_BUNDLE_TABS` - для заполнения Service AddOns. Вся страница заполнена всеми существующими тарифными планами, отсортированными по пакету и частоте.

Заказ

Фасады заказов предоставляют полную возможность покупок со всеми типичными функциями корзины, оформления заказа и истории заказов. Функциональность разбита на три отдельных фасада: фасад корзины для фазы покупок, фасад оформления заказа для процесса размещения заказа и фасад заказов для отображения сведений об отправленных заказах и истории заказов.

Пользователь

Пользовательский фасад обеспечивает выполнение таких операций с учетными записями пользователей, как регистрация, оплата и управление адресной книгой, обновления профилей, включая изменение паролей.

Сессия магазина

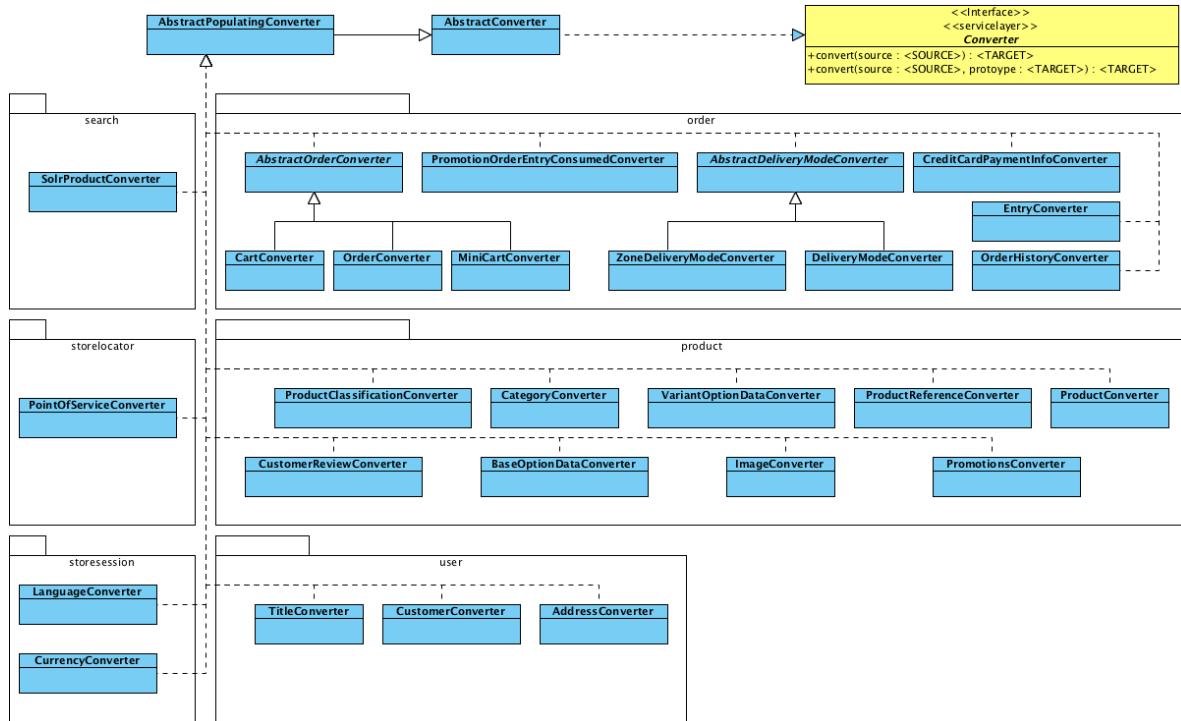
Фасад `Store Session` предоставляет доступ к различным переключениям интернационализации, которые пользователь может выполнить при посещении определенного магазина.

Поиск магазина

Фасад `Store Locator` раскрывает возможности поиска Point Of Sale и предоставляет возможность раскрывать информацию о магазине и хранить определенный контент. Найдите больше информации о `Store Locator` в [hybris Accelerator](#) в [Конфигурация локатора магазинов](#) документ.

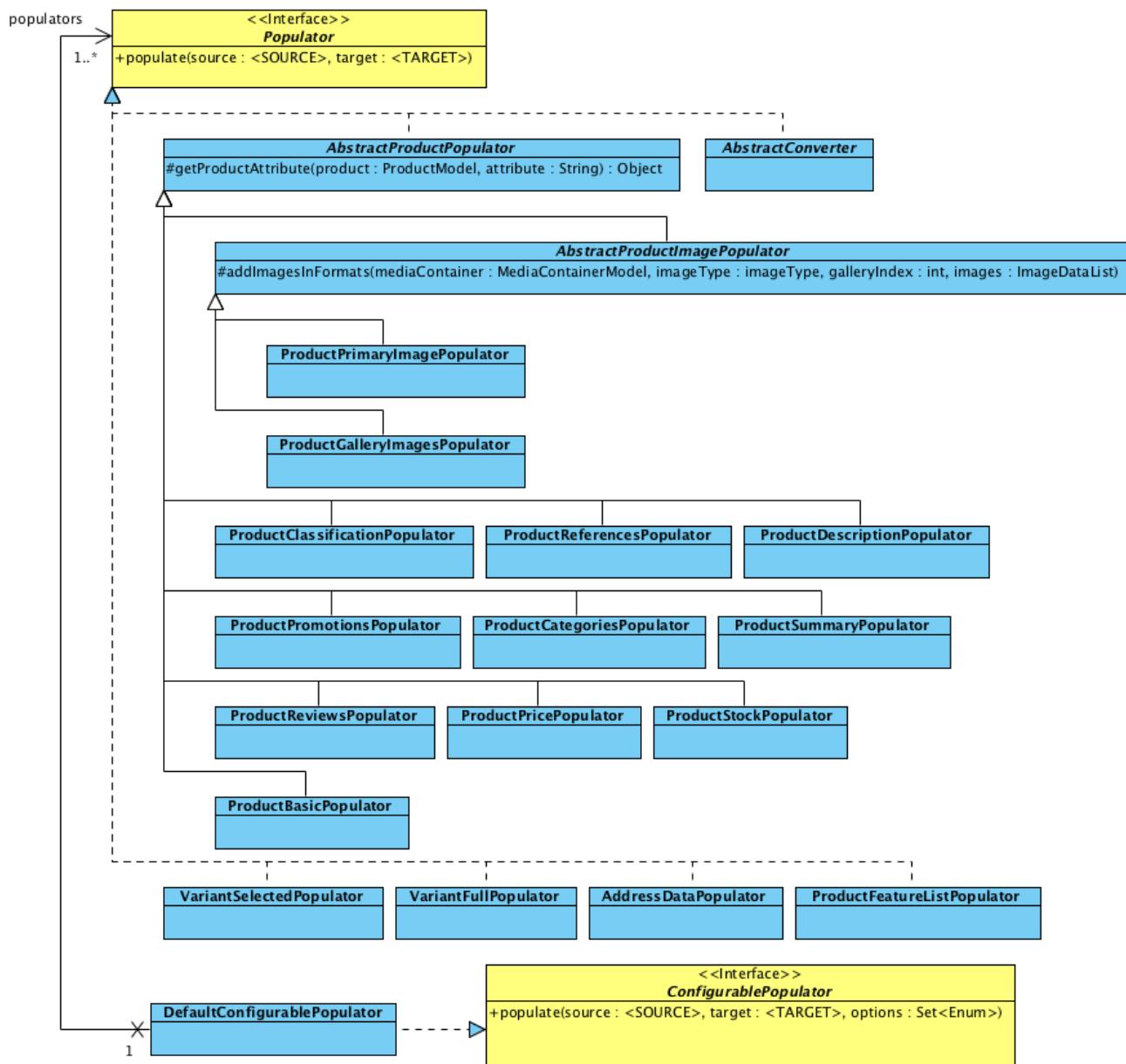
Упакованные преобразователи

На следующей диаграмме показаны преобразователи, поставляемые с расширением `commercefacades`:



Упакованные популяторы

На следующей диаграмме показаны Populators, поставляемые с расширением commercefacades:



Расширение фасадов

Фасады — это управляемые Spring-компоненты с одиночной областью действия, имеющие интерфейс и класс реализации по умолчанию, которые можно настраивать различными способами, включая замену всего компонента, создание подклассов класса компонента и переопределение определенных методов или настройку с помощью композиции, как правило, путем заполнения модели объекта данных с использованием различных или дополнительных экземпляров преобразователей и заполнителей.

Вам следует расширить фасады расширения commercefacades в соответствующем расширении фасадов для вашего проекта. Расширение acceleratorfacades — это шаблон, поставляемый с hybris Accelerator, который вы можете использовать в качестве отправной точки для своего собственного расширения. Это расширение должно иметь зависимость от расширения commercefacades в своем [расширениеinfo.xml](#), а также любые другие необходимые расширения бизнес-уровня, необходимые для завершения функциональности, предлагаемой вашими индивидуальными или дополнительными фасадами.

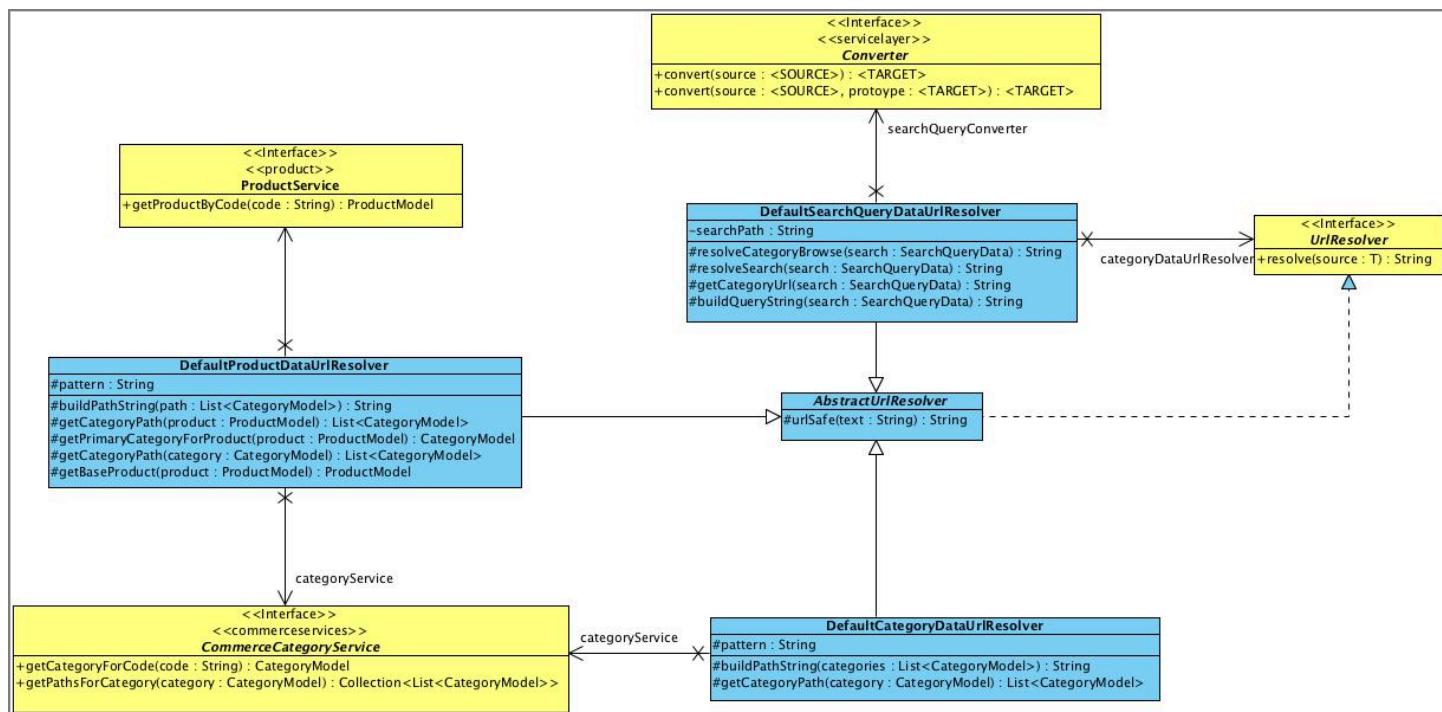
-Примечание

Детали реализации

Добавление зависимости от расширения commercefacades в [расширениеinfo.xml](#) гарантирует, что вы можете заменить определения bean, определенные в контексте приложения расширения commercefacades. Вы можете найти больше информации о расширении Spring beans в [Spring Framework в SAP Commerce ,ServiceLayer](#) и более конкретно [Как продлить услугу](#) документы.

URL-резолверы

URL-резолвер создает относительный URL-адрес для предоставленного исходного элемента, который может использоваться на внешнем интерфейсе для генерации ссылки обратно на исходный элемент. Это гарантирует, что слой фасада может предоставлять URL-адреса в объектной модели данных без необходимости выполнения слоем MVC дополнительного прохода по объектной модели данных. Этот подход также позволяет нескольким клиентским приложениям использовать различные форматы URL-адресов, просто используя фасады и конвертеры, настроенные с различными экземплярами резолвера. Из коробки можно генерировать дружественные к SEO URL-адреса для страниц продуктов и категорий, а также для страниц фасетной навигации.



Сохранить корзину

Функция сохранения корзины в первую очередь позволяет клиенту сохранять и восстанавливать свои сохраненные корзины позднее. Эта функция предоставляется как набор методов, встроенных в независимые стратегии, которые могут быть подключены относительно различных бизнес-требований и реализаций front-end, для которых они используются. Эти стратегии можно легко расширить с помощью pre/post-hooks.

В настоящее время поддерживаются следующие операции сохранения корзины:

- Сохранить корзину сеанса как сохраненную корзину
- Сохраните определенные идентификаторы корзин для внутренних операций в качестве сохраненных корзин
- Отобразить список сохраненных корзин
- Отображение деталей сохраненной корзины
- Восстановить сохраненную корзину в активную корзину сеанса
- Удалить сохраненные корзины
- Клонировать сохраненные корзины
- Отредактируйте название и описание сохраненной корзины.

Для поддержки этой функциональности был реализован новый интерфейс SaveCartFacade. Этот интерфейс содержит следующие методы:

Метод сигнатуры	Возвраты
сохранитьCart(CommerceSaveCartParameterData)	CommerceSaveCartResultData
agForDeletion(Строка)	CommerceSaveCartResultData
getCartForCodeAndCurrentUser(CommerceSaveCartParameterData)	CartData
restoreSavedCart(CommerceSaveCartParameterData)	КорзинаВосстановлениеДанные
getSavedCartsForCurrentUser(PageableData, List<OrderStatus>)	SearchPageData<CartData>
cloneSavedCart(CommerceSaveCartParameterData)	CommerceSaveCartResultData

При вызове метода saveCart() он ожидает, что среди прочих параметров будут предоставлены имя и описание корзины. Если ни один из них не предоставлен, метод генерирует имя и описание на основе cartModel корзины, которую нужно сохранить.

-Примечание

Если вы хотите изменить это поведение, вам придется перезаписать весь метод saveCart() в вашей реализации интерфейса SaveCartFacade.

Помимо реализации SaveCartFacade, ваша пользовательская реализация этого интерфейса должна расширять реализацию интерфейса CartFacade для поддержки операций над объектами Cart. Другими словами, если ваши реализации интерфейсов CartFacade и SaveCartFacade называются DefaultCartFacade и DefaultSaveCartFacade соответственно, заголовок класса выглядит следующим образом:

DefaultCartFacade

открытый класс DefaultSaveCartFacade расширяет DefaultCartFacade и реализует SaveCartFacade

CommerceCartFacade

TheCommerceCartFacade.addToCart(AddToCartParams addToCartParams) метод используется для заполнения новой записи некоторыми данными, специфичными для модуля. Например: настраиваемые пакеты используйте этот метод для добавления группы пакетов к записи.

Конфигурация бобов

Методы SaveCartFacade используют или возвращают следующие объекты:

- CommerceSaveCartResultData,
- CommerceSaveCartParameterData
- КорзинаВосстановлениеДанные

Эти объекты представляют собой сгенерированные ресурсы, созданные из Java-бинов, и вы можете настроить их, изменив их свойства в **коммерцияфасады-бобов.xml**:

commercefacades/resources/commercefacades-beans.xml

<бобы>

```
...
<боб class="de.hybris.platform.commercefacades.order.data.CommerceSaveCartParameterData">
    <свойство имя="cartId" тип="Строка"/> имя="имя"
    <свойство тип="Строка"/> имя="описание"
    <свойство тип="Строка"/>
    <свойство имя="enableHooks" тип="логическое"/>
</боб>
<bean class="de.hybris.platform.commercefacades.order.data.CommerceSaveCartResultData">
    <имя свойства="savedCartData" тип="de.hybris.platform.commercefacades.order.data.CartData"/>
</боб>

<bean class="de.hybris.platform.commercefacades.order.data.CartRestorationData">
    <свойство имя="модификации" type="java.util.List<de.hybris.platform.commercefacades.order.data.CartModificationData>"/>
</боб>
...
</бобов>
```

Исключения

Обычно, в случае возникновения проблем, методы интерфейса SaveCartFacade выдают исключение CommerceSaveCartException. Для получения дополнительной информации см. раздел Исключения [Расширение commerce - Техническое руководство](#) документ.

Структура списков клиентов

The Структура списка клиентов — это общая структура, которая может отображать различные списки клиентов для агентов поддержки клиентов относительно различных бизнес-приложений. Записи клиентов в списках клиентов содержат глубокие ссылки ASM, которые при нажатии на них CSA перенаправляют их на соответствующие страницы, например, на страницу профиля клиента, корзины или заказов. Это позволяет CSA подготовиться с использованием информации о клиенте, прежде чем предоставлять ему персонализированную поддержку. В настоящее время доступны следующие реализации:

Имя в списке клиентов	Описание	Ссылка на документацию
Текущие клиенты магазина	<p>Отображает список клиентов, которые в данный момент находятся в магазине или в разных местах внутри магазина, по данным одного или нескольких аппаратных устройств, чтобы агент службы поддержки клиентов мог просмотреть профиль клиента и подойти к нему подготовленным.</p> <p>По умолчанию данные клиентов имитируются, однако в реальной реализации эти данные будут поступать с одного или нескольких физических устройств Интернета вещей, находящихся в магазине.</p>	assistedserviceservices Расширение документ, раздел Текущие клиенты в магазине
Мои последние сеансы с клиентами	Отображает список клиентов, сеансы которых были недавно начаты агентом службы поддержки клиентов, чтобы агент службы поддержки клиентов мог быстро вернуться к предыдущему сеансу пользователя службы поддержки клиентов.	вспомогательный сервис фасады Расширение документ, раздел Мои последние сеансы клиентов
Самовывоз из магазина для клиентов	Отображает список клиентов, совершивших покупку онлайн и забирающих свои заказы в магазине, чтобы сотрудник службы поддержки клиентов мог подготовить их заказ.	assistedserviceservices Расширение документ, раздел «Забор товара в магазине для клиентов»

Модель данных

Ниже вы можете найти UML-диаграмму модели данных Customer Lists Framework относительной коммерции фасады расширение:

-Примечание

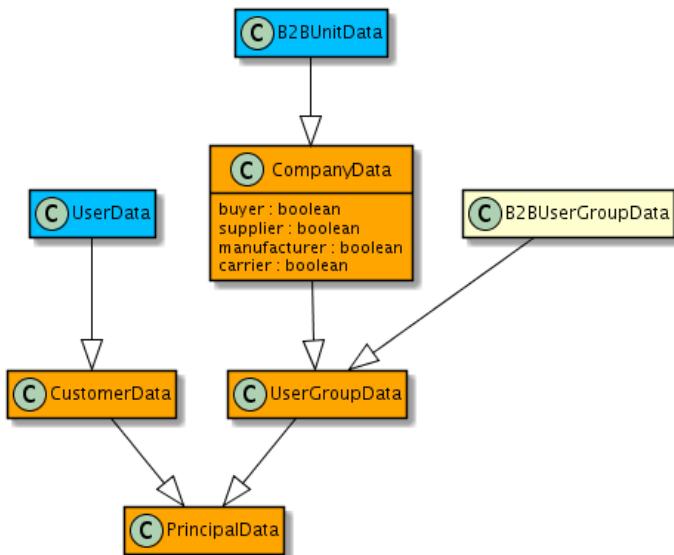
commercefacades (#Orange)

acceleratorfacades (#Purple)

yacceleratorfacades (#Plum)

assistedservicefacades (#Wheat)

b2bcommercefacades (#DeepSkyBlue)



CustomerListFacade

The **CustomerListFacade** Интерфейс — это класс, содержащий методы, используемые для:

- получение списков клиентов для конкретного сотрудника, которые будут отображаться на внешнем интерфейсе
- получение постраничных клиентов для определенного UID списка клиентов с дополнительными параметрами, если необходимо.

The **getPagedCustomersForCustomerListUID()** Метод вызывает стратегию изнутри и возвращает список разбитых на страницы клиентов, а также преобразует модели клиентов в данные о клиентах с помощью пополнителей.

Реализация по умолчанию **CustomerListFacade** Интерфейс содержит следующие методы:

```

/**
 *
 * Класс бетона фасада списка клиентов по умолчанию, который реализует {@link CustomerListFacade}
 *
 */

```

открытый класс **DefaultCustomerListFacade** реализует **CustomerListFacade** {

```

// сохраняет различные реализации списка клиентов в частном
порядке Служба списка клиентов Служба списка клиентов; Служба поиска
частный списка клиентов Служба поиска списка клиентов;
частный преобразователь<CustomerListModel, UserGroupData> userGroupConverter;
private Map<String, Converter<UserModel, CustomerData>> customerListImplementationStrategiesConverter; private Converter<UserModel,
CustomerData> customerConverter;

@Переопределить
public List<UserGroupData> getCustomerListsForEmployee(final String employeeUid) {

    validateParameterNotNullStandardMessage("employeeUid", employeeUid);

    return getUserGroupConverter().convertAll(getCustomerListService().getCustomerListsForEmployee(employeeUid));
}

@Переопределить
public <T extends CustomerData> SearchPageData<T> getPagedCustomersForCustomerListUID(final String customerListUid,
    окончательная строка employeeUid, окончательная pageableData pageableData, окончательная карта <String, Object> параметрMap)
{
    validateParameterNotNullStandardMessage("customerListUid", customerListUid);

    validateParameterNotNullStandardMessage("pageableData", (страница_данных));
    validateParameterNotNullStandardMessage("employeeUid", (идентификатор сотрудника));

    окончательная модель списка клиентов customerListModel = getCustomerListService().getCustomerListForEmployee(customerListUid,
        (идентификатор сотрудника));

    validateParameterNotNull(customerListModel,
        String.format("Список клиентов не найден для customerListUid '%1$s'", customerListUid));

    окончательная строка implementationType = customerListModel.getImplementationType();

    окончательный SearchPageData<CustomerModel> searchPageData = getCustomerListSearchService().getPagedCustomers(customerListUid,
        employeeUid, pageableData, ParameterMap);

    Список<ДанныеКлиента> customerDataList = null;

    окончательный преобразователь<UserModel, CustomerData> strategyConverter = getCustomerListImplementationStrategiesConverter()
        .получить(тиреализации);

    если (null == strategyConverter) {

        СписокДанныхКлиентов = получитьCustomerConverter().convertAll(searchPageData.getResults());
    }
    еще
    {
        СписокДанныхКлиентов = strategyConverter.convertAll(searchPageData.getResults());
    }

    окончательный SearchPageData<T> customersSearchPageData = new SearchPageData<T>();

    customersSearchPageData.setResults((List<T>) customerDataList);
    customersSearchPageData.setPagination(searchPageData.getPagination());
    customersSearchPageData.setSorts(searchPageData.getSorts());

    возврат клиентовSearchPageData;
}

зашитенный CustomerListService getCustomerListService() {

    вернуть CustomerListService;
}

@Необходимый
public void setCustomerListService(final CustomerListService customerListService) {

    этот.customerListService = customerListService;
}

зашитенный конвертер<CustomerListModel, UserGroupData> getUserGroupConverter()

```

```

{
    возвращаясь userGroupConverter;
}

@Необходимый
public void setUserGroupConverter(final Converter<CustomerListModel, UserGroupData> userGroupConverter) {

    этот.userGroupConverter = userGroupConverter;
}

защищенная Карта<Строка, Конвертер<Пользовательская модель, Данные клиента>> getCustomerListImplementationStrategiesConverter() {

    вернуть CustomerListImplementationStrategiesConverter;
}

@Необходимый
публичная пустота setCustomerListImplementationStrategiesConverter(
    окончательная карта <Строка, Преобразователь <Пользовательская модель, Данные клиента>> Список клиентов (Стратегии реализации конвертера)
{
    этот.customerListImplementationStrategiesConverter = CustomerListImplementationStrategiesConverter;
}

зашитенный конвертер<UserModel, CustomerData> getCustomerConverter() {

    возврат клиентаConverter;
}

@Необходимый
public void setCustomerConverter(final Converter<UserModel, CustomerData> customerConverter) {

    этот.customerConverter = customerConverter;
}

защищенный CustomerListSearchService getCustomerListSearchService() {

    возврат customerListSearchService;
}

@Необходимый
public void setCustomerListSearchService(final CustomerListSearchService customerListSearchService) {

    this.customerListSearchService = customerListSearchService;
}
}
}

```

Популяторы

Чтобы получить PagedCustomersForCustomerListUID() Метод содержит ссылку на карту для преобразователей стратегий. Ключами этой карты являются идентификаторы списка клиентов, а значениями являются фактические преобразователи, которые преобразуют экземпляры модели, возвращаемые изCustomerListSearchService вместо этого вызовов экземпляров данных. Если нет конвертера для определенного списка клиентов, то вместо этого используется конвертер по умолчанию, который заполняет адрес клиента, изображение профиля и корзину сессии, если они существуют.

Весенняя конфигурация

В commercefacades-spring.xml файле, Список клиентов Реализация Стратегии Конвертеров map is denied. Эта карта содержит различные типы конвертеров, используемых для преобразования моделей в данные. Также вы можете найти определение CustomerListFacade bean и его зависимые службы в commercefacades-spring.xml файле:

```

<alias name="defaultCustomerListFacade" alias="customerListFacade"/>

<bean id="defaultCustomerListFacade"
      class="de.hybris.platform.commercefacades.customer.impl.DefaultCustomerListFacade"> <свойство
          name="customerListSearchService" ref="customerListSearchService"/>
          <свойство name="customerListImplementationStrategiesConverter" ref="customerListImplementationStrategiesConverter"/>
          <свойство имя="customerListService" ref="customerListService"/>
          <свойство имя="userGroupConverter" ref="userGroupConverter"/>
          <свойство имя="customerConverter" ref="customerConverter"/>
</боб>

```

Для получения более подробной информации о Структура списков клиентов, см. [Расширение commerceservices](#) документ.

Универсальный механизм добавления столбцов

Следующий фрагмент из commercefacades/resources/commercefacades-spring.xml показан общий механизм, позволяющий добавлять столбцы в список клиентов:

```

<псевдоним имя="defaultCustomerListConverter" alias="customerListConverter"/> parent="abstractPopulatingConverter">
<боб идентификатор="defaultCustomerListConverter" value="de.hybris.platform.commercefacades.user.data.CustomerListData"/>
    <свойство имя="целевойКласс"
    <свойство имя="популяторы">
        <список>
            <ref bean="customerListPopulator"/>
        </список>
    </свойство>
</боб>

<псевдоним имя="defaultCustomerListPopulator" псевдоним="customerListPopulator"/>
<боб id="defaultCustomerListPopulator" class="de.hybris.platform.commercefacades.user.converters.populator.CustomerListPopulator">
    <property name="customerListAdditionalColumnsMap" ref="customerListAdditionalColumnsMap"/>
</боб>

<!-- Ключами карты являются CustomerList.additionalColumnsKeys добавленных столбцов, а значениями карты являются синтаксис EL Cu <util:map
id="customerListAdditionalColumnsMap" key-type="java.lang.String" value-type="java.lang.String" scope="tenant"/>

```

Дополнительную информацию о добавлении собственных столбцов в список клиентов см.[Интеграция АСМ](#).

Фасад будущего запаса

Фасадный слой

АDefaultFutureStockFacade реализация FutureStockFacade интерфейс подключается к FutureStockService. Это соединение используется для запроса будущих данных о запасах. Запрос передается FutureStockFacade как список для FutureStockService.

The DefaultFutureStockFacade предоставляет следующие методы:

- Получает информацию о будущей доступности продукта, например, об уровне запасов и дате доступности для указанного продукта.
 - Параметр: КодПродукта. Этот параметр — код продукта.
 - Возврат: Список<FutureStockData>. Возвращает список будущих данных о запасах, упорядоченных по дате.
 - Если для этого продукта нет будущей доступности, возвращается пустой список.
- Получает будущую доступность продукта, такую как уровень запасов и дату доступности, для списка указанных продуктов.
 - Параметр: Список<String> productCodes. Этот параметр представляет собой список кодов продуктов.
 - Возврат: Карта<Строка, Список<FutureStockData>>. Возвращает карту кодов продуктов со списком данных о будущих запасах, упорядоченных по дате.
 - Если для этих продуктов нет будущей доступности, возвращается пустая карта.
- Получает информацию о будущей доступности продукта, например, об уровне запасов и дате доступности, для списка указанных вариантов, относящихся к данному продукту.
 - Параметр: ProductCode, List<String> skus. Этот параметр состоит из двух частей. Первая часть — это код базового продукта. Если указан вариант продукта, будет выведен соответствующий базовый продукт. Вторая часть — это список SKU (коды вариантов продукта для базового продукта).
 - Возврат: Карта<Строка, Список<FutureStockData>>. Возвращает карту кодов продуктов, которые включены в указанные SKU и соответствуют указанному базовому продукту, с соответствующим списком данных о будущих запасах, упорядоченных по дате.
 - Если первый параметр — базовый продукт без вариантов, возвращается значение null.

Модель данных

Ниже вы можете ознакомиться с моделью данных будущего складского фасада:

 FutureStockData
stock: StockData
date: Date
formattedDate: String

Интеграция пользовательского фьючерсного фондового сервиса

Если вы хотите добавить собственную реализацию будущей доступности запасов, вам необходимо реализовать FutureStockFacade интерфейс и настроить его. Вы можете либо заменить текущую конфигурацию, либо создать собственную переопределяющую XML-конфигурацию для этого бина.

Вот конфигурация пружины для FutureStockFacade.

```

<псевдоним name="defaultFutureStockFacade" псевдоним="futureStockFacade"/>
<bean id="defaultFutureStockFacade" class="de.hybris.platform.commercefacades.futurestock.impl.DefaultFutureStockFacade" >
    <свойство имя="commerceCommonI18NService" ref="commerceCommonI18NService"/>
    <свойство имя="futureStockService" ref="futureStockService"/>
    <свойство имя="productService" ref="productService"/>
</боб>

```

Контроллер использует реализацию, которую вы вставляете для futureStockFacade может остаться неизменным.

Сопутствующая информация

[Документация SAP Commerce Accelerator](#)

[Использование фасадов и DTO — лучшие](#)[практические модели](#)[ServiceLayer](#)[Как расширить возможности поиска и](#)[навигации ProductFacade](#)

Расширение commercewebservicescommons

The коммерция веб-сервисы commons расширение позволяет расширять и настраивать веб-сервисы Commerce с помощью концепции дополнений.

Особенности коммерция веб-сервисы commons Расширение

Ошибки

SAP Commerce предоставляет стандартные исключения WebserviceExceptions, используемые в commercewebservices. Вы можете использовать исключения по умолчанию или реализовать собственные исключения, расширяя WebserviceException/WebserviceValidationException. SAP Commerce предоставляет следующие исключения по умолчанию:

- WebserviceException
- WebserviceValidationException
- Исключение адреса корзины
- CartEntryException
- CartException
- LowStockException
- ProductLowStockException
- RequestParameterException
- SessionAttributeException
- StockSystemException

SAP Commerce также предоставляет механизм для преобразования объектов в ошибки, известный как объекты ErrorData. В следующей таблице перечислены преобразователи по умолчанию, предоставляемые SAP Commerce:

Конвертер	Объект
WebserviceExceptionConverter	de.hybris.platform.commercewebservicescommons.errors.exceptions.WebserviceException
Исключение Конвертер	java.lang.Исключение
ValidationErrorConverter	org.springframework.validation.Ошибка
CartModicationDataListErrorConverter	de.hybris.platform.commercefacades.order.data.CartModicationDataList
CartModicationDataErrorConverter	de.hybris.platform.commercefacades.order.data.CartModicationData

DataMapper

Все классы, связанные с DataMapper (фильтры, преобразователи, картографы и т. д.), доступны в [Картирование DTO и конфигурация ответа](#).

Пагинация и сортировка

Следующие объекты, связанные с нумерацией страниц и сортировкой в коммерция веб-сервисы commons Расширение устарело:

- СортироватьWsDto
- ПагинацияWsDto
- PagableWsDto

Заменяемые объекты можно найти в [веб-сервисы commons Расширение](#).

Для получения более подробной информации см. [Пагинация и сортировка](#).

Сопутствующая информация

[Архитектура дополнений OCC](#)

Расширение ycommercewebservices test

The `ycommercewebservicetest` расширение было создано для того, чтобы позволить третьим лицам расширять его, и расширение предоставляет набор тестов, написанных на Groovy, которые предназначены для тестирования уcommerceвеб-сервисы REST-вызовы.

-Осторожность

Эта страница относится к программному обеспечению, которое было устарело как часть Accelerator UI и устаревших расширений шаблонов OCC. Для получения дополнительной информации см.[Прекращение поддержки пользовательских интерфейсов Accelerator и старых расширений шаблонов OCC](#).

Тесты Groovy основаны на наборе данных образца, `wsTest`, который также включен в это расширение. Шаблон расширения `ycommercewebservicetest` также включен. Таким образом, можно расширить набор тестов после расширения `ycommercewebservicetest` функциональность.

Образец набора данных

Данные загружаются перед запуском тестового набора и удаляются после завершения тестов. Набор данных состоит из более чем двадцати файлов ImpEx, доступных в следующем каталоге: `ycommercewebservicetest/resources/ycommercewebservicetest/import`. Набор данных содержит отдельное хранилище, называемое `wsTest Store`, и может быть загружен для клиента `junit` только во время выполнения всего набора тестов.

Файлы импортируются всоздать `ProjectData` функция в `YCommerceWebServicesTestSetup` класс. Любые новые файлы должны быть перечислены в этом классе. Тестовые наборы Groovy загружают данные с помощью `TestSetupUtil` класс, содержащий статический метод для загрузки данных и запуска встроенного сервера.

Тестовый набор

Тесты, включенные в `ycommercewebservicetest` расширения написаны на Groovy.

-Примечание

Тестовые наборы содержат логику, используемую для загрузки данных, необходимых для теста и запуска встроенного сервера `tomcat`. Если упомянутые два шага пропущены, тест не будет запущен должным образом. Вот почему запуск одного теста не будет пройден, и тесты должны запускаться как целый тестовый набор.

Название набора тестов	Описание
<code>de.hybris.platform.ycommercewebservicetest.test.groovy.webservicetests.v2.spock.AllSpockTests</code>	тестовый набор для версии v2 уcommerceвеб-сервисы
<code>de.hybris.platform.ycommercewebservicetest.test.groovy.webservicetests.addons.v2.spock.AllAccSpockTests</code>	набор тестов для конечных точек, определенных в ускорительвеб-сервисыдополнение
<code>de.hybris.platform.ycommercewebservicetest.test.groovy.webservicetests.v1.AllTests</code>	тестовый набор для версии v1 уcommerceвеб-сервисы
<code>de.hybris.platform.ycommercewebservicetest.test.groovy.webservicetests.addons.AllAccTests</code>	набор тестов для конечных точек, определенных в <code>acceleratorwebservicesaddon</code> и относящихся к версии v1.

Тесты доступны в следующем каталоге: `ycommercewebservicetest\tests\src\de\hybris\platform\ycommercewebservicetest\test\groovy`.

Конфигурацию теста можно найти в следующем файле: `ycommercewebservicetest\resources\groovytests-property-file.groovy`

-Примечание

Обратите внимание, что тесты в Groovy используют обычные классы Java, поэтому `ycommercewebservicetest` расширение необходимо скомпилировать перед началом тестов.

Тесты можно запускать как стандартные интеграционные тесты SAP Commerce из SAP Commerce Testweb Frontend (подробнее см.: [Интерфейс SAP Commerce Testweb](#)) или с помощью следующей команды ant:

```
ant integrationtests -Dtestclasses.packages=de.hybris.platform.ycommercewebservicetest.test.groovy.webservicetests.v2.spock.AllSpockT
```

Вы также можете запустить одиничный тест spock с помощью следующей команды ant:

```
ant manualtests -Dtestclasses.packages=de.hybris.platform.ycommercewebservicetest.test.groovy.webservicetests.v2.spock.carts.CartReso
```

Имейте в виду, что запуск одного теста не импортирует данные образца. Для этого выполните следующую команду:

импорт antwsttestdata

Сопутствующая информация

[Архитектура дополнений OCC](#)

Расширение ycommercewebservices

The ycommercewebservices расширение представляет часть Commerce Facades как веб-сервисы на основе REST, включая вызовы для поиска продукта и получения сведений о продукте. Расширение было создано для того, чтобы позволить третьим сторонам расширять и настраивать веб-сервисы с помощью концепции дополнений.

-Осторожность

Эта страница относится к программному обеспечению, которое было устарело как часть Accelerator UI и устаревших расширений шаблонов OCC. Для получения дополнительной информации см. [Прекращение поддержки пользовательских интерфейсов Accelerator и старых расширений шаблонов OCC](#).

The ycommercewebservices Расширение в настоящее время раскрывает часть фасадов Commerce как веб-сервисы на основе REST, включая вызовы поиска продукта и сведения о продукте. Цель состоит в том, чтобы предоставить рабочий пример того, как API на основе REST может быть раскрыто. Поскольку веб-сервисы Commerce основаны на стандартном Spring MVC, вы можете легко настроить или расширить их.

-Примечание

Расширение SAP Commerce может предоставлять функциональность, лицензируемую через различные модули SAP Commerce. Обязательно ограничьте реализацию функциями, определенными в вашей контрактной лицензии. В случае сомнений обратитесь к своему торговому представителю.

Общая информация

Веб-сервис — это метод связи между двумя электронными устройствами по сети. Клиент обычно использует REST API для связи с бэкендом веб-сервисов. Он также может использовать простые HTTP-соединения, например, через AJAX или любые другие сетевые классы на основе HTTP.

Существует множество различных клиентов, как мобильных, так и настольных, которые потенциально могут потреблять веб-сервисы. Клиенты могут быть написаны на том же языке, что и серверный код, а могут и не быть, поэтому имеет смысл использовать универсальный язык для общения. Commerce Web Services поддерживают форматы данных XML и JSON, которые можно легко анализировать на всех мобильных операционных системах, а также на настольных клиентах.

Commerce Web Services по сути являются относительно тонким дополнительным слоем поверх Commerce Facades, которые выполняют определенные преобразования объектов. Commerce Web Services зависят от коммерческих фасадов расширение. Зависимости определены в расширении info.xml в ycommercewebservices.

```
...
< требуется-расширение имя="commercefacades"/>
...
```

Конфигурация коммерческих веб-сервисов

По умолчанию Commerce Web Services доступны в контексте REST Web, как указано в расширении info.xml.

```
...
<webmodule jspcompile="false" webroot="/rest" /> ...
...
```

-Примечание

Commerce Web Services доступны в двух версиях: Версия 1 и Версия 2 (версия по умолчанию). Подробности см.: [v1 и v2 в ycommercewebservices](#).

- Справку по вызовам V1 и примеры URL-адресов см. по адресу: [Справочник вызовов - v1](#) и [Потоки образцов OCC](#).
- Справку по вызовам V2 и примеры URL-адресов см. по адресу: [Справочник вызовов - v2](#).

Commerce Web Services используют базовую аутентификацию. Каждый вызов, даже поиск и получение сведений о продукте, должен отправлять заголовки аутентификации, как определено в базовой схеме аутентификации. Аутентифицированный пользователь должен быть в группе клиентов, чтобы успешно пройти аутентификацию.

-Примечание

The `ycommercewebservices` проект.свойствale содержит ряд настроек Cross-Origin Resource Sharing (CORS), которые можно настроить для использования с доверенным сторонним веб-приложением. Если вы работаете в производственной среде, следующее corsfilter необходимо настроить параметры:

```
corsfilter.ycommercewebservices.allowedOrigins=http://localhost:4200 https://localhost:4200
corsfilter.ycommercewebservices.allowedMethods=GET HEAD OPTIONS PATCH PUT POST DELETE
corsfilter.ycommercewebservices.allowedHeaders=origin content-type accept authorization
```

Для получения более подробной информации см.[Поддержка обмена ресурсами между источниками](#).

Протокол OAuth 2.0 в коммерческих веб-сервисах

OAuth 2.0 — это следующее поколение протокола OAuth, созданного в конце 2006 года. OAuth 2.0 ориентирован на простоту разработки клиентов, обеспечивая при этом определенные потоки авторизации для веб-приложений, настольных приложений, мобильных телефонов и домашних устройств.

Ключевое преимущество использования OAuth 2.0 (по сравнению с базовой аутентификацией, даже по HTTPS) заключается в том, что клиенту API не нужно сохранять или, в некоторых случаях, даже получать учетные данные пользователя. Вместо этого токены доступа возвращаются клиенту, который может использовать токены обновления для получения новых токенов доступа после истечения срока их действия.

Пакет SAP Commerce не предоставляет образцы клиентов OAuth, поэтому вам придется определить их самостоятельно. Подробности см.:[Настройка клиентов OAuth](#).

-Осторожность

При определении клиентов не забудьте назначить им либо `ROLE_CLIENT`, либо `ROLE_TRUSTED_CLIENT`, поскольку эти роли разрешают клиентам доступ к усоммергеб-сервисы расширение. Однако будьте осторожны с `ROLE_TRUSTED_CLIENT`, поскольку у него есть особые, расширенные права.

Сопутствующая информация

[Документация SAP Commerce Accelerator](#)

[OAuth 2.0](#)

[Архитектура дополнений OCC](#)

Проверка корзины для OCC

OCC (Omni Commerce Connect) использует ту же проверку корзины, что и акселератор. Проверка включает в себя не только проверки достаточного запаса, но и все зарегистрированные хуки проверки. Логика, используемая для проверки, может немедленно изменить корзину, чтобы гарантировать ее согласованное состояние. Например, если достаточный запас для продукта отсутствует, количество уменьшается до соответствия доступному запасу или, если запас отсутствует, товар полностью удаляется из корзины.

-Примечание

Вы можете использовать упрощенную проверку, которая проверяет только наличие достаточного запаса, но не проверяет наличие зарегистрированных крючков или немедленно изменяет корзину. Если вы хотите использовать упрощенную проверку, замените:

```
<bean id="defaultCommerceWebServicesCartService" parent="defaultCommerceCartService">

    <свойство name="cartValidationStrategy" ref="defaultCartValidationStrategy"/>

    <свойство имя="СтратегияКонфигурацииПродукта" ref="СтратегияКонфигурацииПродукта"/>

</боб>

<c>

<bean id="defaultCommerceWebServicesCartService" parent="defaultCommerceCartService">

    <property name="cartValidationStrategy" ref=" cartValidationWithoutCartAlteringStrategy"/>

    <имя_свойства="productConfigurationStrategy" ref="productConfigurationStrategy"/>

</боб>
```

Проверка корзины перед оформлением заказа

Конечная точка API `{baseSiteId}/users/{userId}/carts/{cartId}`/валидация может использоваться для проверки корзины перед оформлением заказа, чтобы гарантировать ее согласованность. Конечная точка запускает проверку корзины и возвращает результат проверки. Корзина может быть изменена в ходе проверки.

Расширение йоккадона

The йоккадон Шаблон расширения — это заранее определенное расширение, которое служит отправной точкой для создания нового дополнения для Commerce Web Services. Йоккадон расширение по сути является пустым расширением с минимальными реализациями, необходимыми для создания дополнения.

-Осторожность

Эта страница относится к программному обеспечению, которое было устарело как часть Accelerator UI и устаревших расширений шаблонов OCC. Для получения дополнительной информации см.[Прекращение поддержки пользовательских интерфейсов Accelerator и старых расширений шаблонов OCC](#).

Содержание расширения yoccaddon

В следующих разделах описывается содержание яркокаддонрасширение.

Файлы Impex

The яркокаддонрасширение содержит следующие пустые файлы impex:

- essentialdata_yoccaddon.impex
- projectdata_yoccaddon.impex

Эти файлы копируются в сгенерированный AddOn и имеют следующее соглашение:

essentialdata_{addonname}.impex и projectdata_{addonname}.impex

Файлы автоматически импортируются во время обновления и инициализации.

Контекст Web Spring Расширение контекста веб-сервисов Commerce

The яркокаддонweb-spring.xml используется для расширения веб-контекста Commerce Web Services. Этот файл находится в расширении, содержит ресурс\yoccaddon\web\веснаКаталог. Он содержит конфигурацию сканирования компонентов для пакета контроллеров и бины, необходимые для конфигурации кэша.

```
<бобы xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context" xsi:schemaLocation="http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="yoccaddonpackage.controllers"/>

...
</бобов>
```

-Примечание

После завершения процесса extgen файл адаптируется для включения сгенерированного имени AddOn и пакета. Это означает, что файл контекста Web Spring находится в ресурс\{имя_дополнения}\web\spring\{имя_дополнения}-web-spring.xml

Файл шаблона для свойств

The яркокаддонрасширение содержит проект.свойства.шаблоне, который необходим для AddOns. Этот файл является шаблоном для свойства проекта, который генерируется в процессе установки дополнения. Подробности см. в[Установка дополнения для определенного магазина](#).

Указывает расположение файла контекста Spring, автоматически добавляемого в глобальный контекст приложения платформы. yoccaddon.application-context=yoccaddon-spring.xml

Указывает местоположение файла контекста Spring, который будет добавлен в веб-контекст коммерческих веб-сервисов ycommercewebservices.additionalWebSpringConfigs.yoccaddon=classpath:/yoccaddon/web/spring/yoccaddon-web-spring.xml

-Примечание

После завершения процесса extgen содержимое {имя_надстройки}.application-context={имя_надстройки}-spring.xml адаптирован для включения сгенерированного имени AddOn:

ycommercewebservices.additionalWebSpringConfigs.{addonname}=classpath:{addonname}/web/spring/{addonname}-web-spring.xml

Пакет сообщений

The яркокаддонрасширение также содержит Basic.properties, Basic_en.properties, Basic_de.properties расположены в /acceleratoraddon/web/webroot/WEB-INF/messages каталог. В этих файлах addOn может определить локализованный пакет сообщений, который используется в Commerce Web Services.

-Примечание

Подробности см.[Расширение коммерческих услуг](#).

Конфигурация кэша

Дополнение также может расширить конфигурацию для `cache` используемой в Commerce Web Services. Это можно сделать в `cache.xml` расположенный в `/acceleratoraddon/web/webroot/WEB-INF/cache/catalog`.

-Примечание

Подробности см.[Расширение коммерческих услуг](#).

Расширение commercewebservices

The коммерция веб-услуги расширение представляет часть фасадов Commerce как веб-сервисы на базе REST, включая вызовы для поиска товаров и сведения о товарах.

Это расширение основано на `commercewebservices` шаблоне.

-Примечание

The коммерция веб-услуги расширение больше не является шаблоном (в отличие от `commercewebservices`), а обычное расширение.

В коммерция веб-услуги расширение:

- API V1 отсутствует
- Корень веб-сайта — `/occ/v2`
- Названия пакетов начинаются `cde.hybris.platform.commercewebservices.core`
- Имя расширения коммерция веб-услуги используется в коде и свойствах

Для получения более подробной информации см.[Архитектура расширения OCC](#) и [Расширение уcommercewebservices](#).

Расширение commercewebservicestests

The коммерция веб-сервисы тесты расширение предоставляет набор тестов, написанных на Groovy, которые предназначены для проверки вызовов REST `commercewebservices`.

Это расширение основано на `commercewebservicestests` расширение.

-Примечание

The коммерция веб-сервисы тесты расширение не является шаблоном (в отличие от `commercewebservicestests`), а обычное расширение.

В коммерция веб-сервисы тесты расширение:

- API V1 отсутствует
- Корень веб-сайта — `/occ/v2`
- Названия пакетов начинаются `cde.hybris.platform.commercewebservicestests`
- Имя расширения коммерция веб-сервисы тесты используется в коде и свойствах

Для получения более подробной информации см.[Архитектура расширения OCC](#) и [Расширение уcommercewebservicestests](#).

Расширение уосс

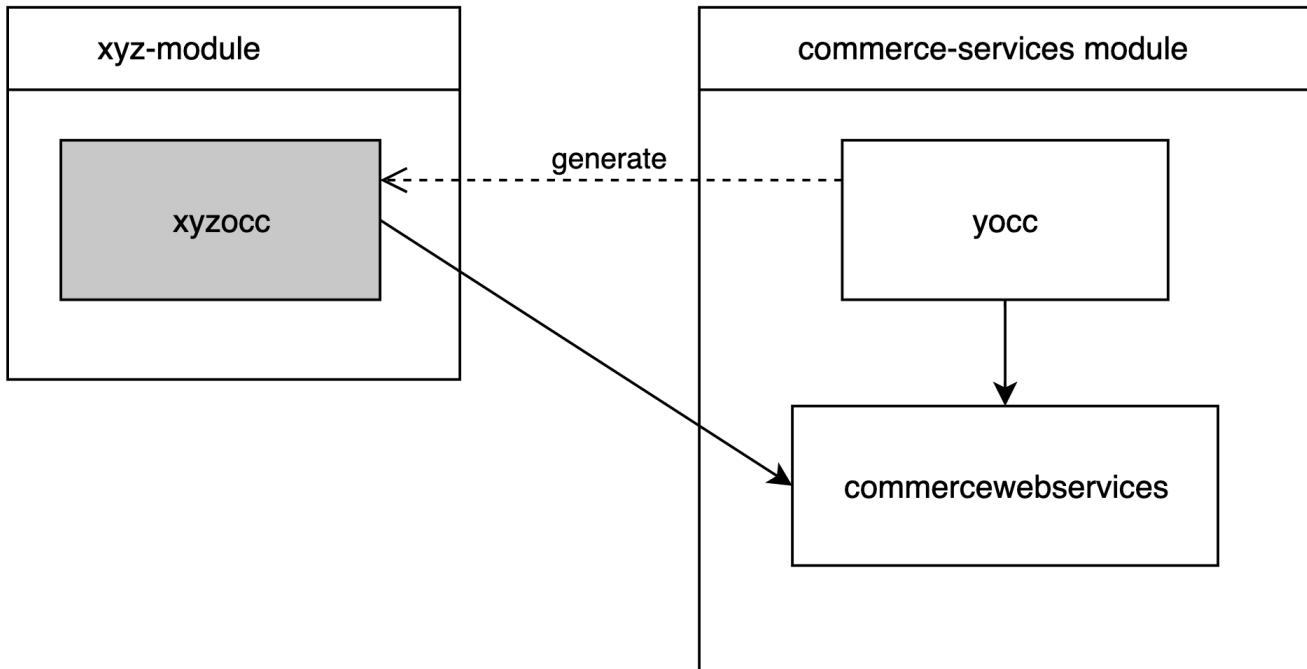
The `occ` Шаблон расширения — это заранее определенное расширение, которое служит отправной точкой для создания новых расширений для Commerce Web Services.

The `occ` расширение — пустое расширение с минимальными реализациями, необходимыми для создания расширения OCC.

На основе `occ` шаблона, новое расширение можно сгенерировать, указав новое имя расширения, которое должно заканчиваться на `occ`, например `xyzocc`.

Зависимости между расширениями

The `occ` зависит от коммерция веб-услуги расширение, позволяющее повторно использовать служебные классы.



Структура каталога расширения уосс

Структура каталогов этого расширения соответствует структуре обычного расширения.

Каталог/Имя файла	Описание
расширениеinfo.xml	Обеспечивает конфигурацию расширения, устанавливает зависимость от коммерциявеб-услугирасширение.
свойства проекта	Содержит свойства конфигурации для сгенерированного расширения ОСС.
/источник	Содержит классы REST Controller и другие классы расширения ОСС.
/testsrc	Содержит тестовые контроллеры для расширения, созданного шаблоном уосс.
/ресурсы	Содержит XML-файлы, которые включают: <ul style="list-style-type: none"> {имя_расширения}-beans.xml -модель данных, определения POJO, {имя_расширения}-spring.xml -Определения весенней фасоли, {имя_расширения}-items.xml -определения типов
/ресурсы/impex	TheyoccРасширение не имеет файлов ImpEx, но для того, чтобы обеспечить возможность импорта данных через файлы ImpEx для вновь созданных расширений, их можно сохранить в этом месте. Файлы ImpEx в расширениях ОСС не загружаются автоматически, если они не соответствуют соглашению о конфигурации.
/resources/occ/v2/yoocc/messages	Содержит файлы сообщений.
/resources/occ/v2/yoocc/web/spring/yoocc-web-spring.xml	Файл, содержащий определения компонентов Spring расширения ОСС.

Для получения более подробной информации см.[Архитектура расширения ОСС](#).

Расширение yoocctests

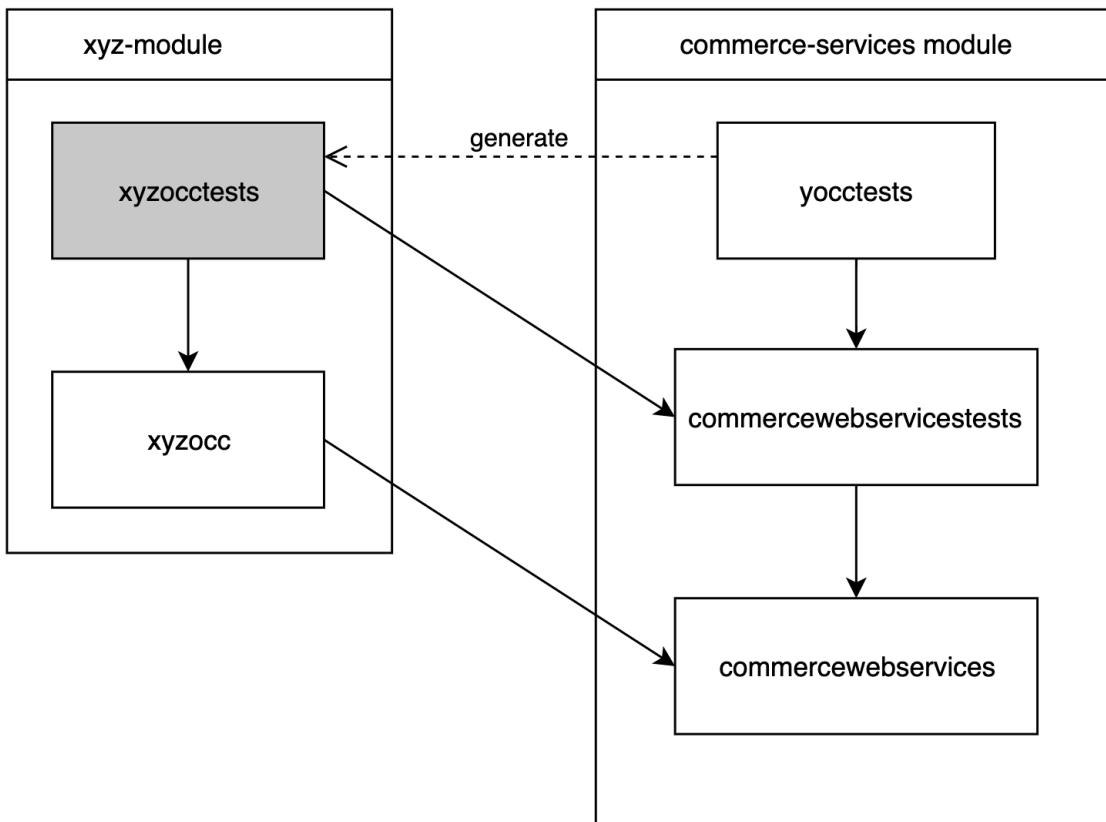
Theyoocctests— это расширение шаблона, используемое для создания нового расширения тестирования.

Сгенерированное расширение тестов должно предоставлять ручные и интеграционные тесты, позволяющие тестировать все функциональные возможности расширения ОСС.

Зависимости между расширениями

Theyoocctestsрасширение зависит откоммерциявеб-сервисытесты,так что служебные классы можно использовать повторно.

Сгенерированное расширение тестов, например xyzoccstests, имеет зависимость от расширения ОСС, например xyzocc.



Генерация пользовательских тестов расширения

The `yocctests` расширение шаблона содержит файлы с вхождениями `yocctests`. Заполнители заменяются в процессе генерации фактическим именем расширения OCC.

- . Использовать муравей `extgen` команду для создания пользовательского расширения тестов на основе `yoccteststemplate`. В результате вы получаете расширение тестов без каких-либо классов тестов и с минимальной реализацией.
- . В расширение `info.xml` добавьте зависимость к расширению OCC, которое необходимо протестировать.
- . Настройте расширение тестов и создайте тесты и наборы тестов для тестирования функциональности расширения OCC. Рекомендуется писать тесты с использованием фреймворка Spock.
- . При желании можно добавить тестовые данные `bessential-data.impexle`, который автоматически импортируется во время инициализации клиента JUnit.

Структура каталога расширения `yocctests`

Структура каталогов этого расширения соответствует структуре обычного расширения.

Каталог/Имя файла	Описание
<code>расширениеinfo.xml</code>	Обеспечивает конфигурацию расширения, устанавливает зависимость от коммерциявеб-сервисытестырасширение.
<code>свойства проекта</code>	Содержит свойства конфигурации для созданного расширения тестов OCC.
<code>tenant_junit.свойства</code>	Файл свойств арендатора JUnit для созданного расширения тестов.
<code>buildcallbacks.xml</code>	<p>Содержит <code>yocctests_importwtestmacro</code> определение данных, вызываемое из коммерциявеб-сервисытестырасширение когда импорт <code>antwtestdata</code> команда есть вызван.</p> <p>Это макроопределение можно использовать для загрузки тестовых данных сгенерированного расширения тестов во время импорт <code>antwtestdata</code> вызовов команды из коммерциявеб-сервисытестырасширение.</p> <p>В сгенерированном расширении <code>yocctests_importwtest</code> макроса меняется к <code>{имя_тестового_расширения}_importwtestdata</code> имя.</p>
<code>/источник</code>	Содержит файлы исходного кода расширения.
<code>/testsрс</code>	Сгенерированное расширение должно содержать классы тестов и тестовых наборов. Они могут быть написаны на Java или Groovy.
<code>/ресурсы</code>	Содержит XML-файлы, которые включают:

Каталог/Имя файла	Описание
	<ul style="list-style-type: none"> • {имя_тестового_расширения}-beans.xml -модель данных, определения POJO, • {имя_тестового_расширения}-spring.xml -Определения весенней фасоли, • {имя_тестового_расширения}-items.xml -определения типов <p>Также содержит локализации и файлы ImpEx с тестовыми данными.</p>
/resources/yocctests/groovytests-property-file.groovy	Файл конфигурации тестовых сред, используемый при загрузке тестовых данных из коммерциявеб-сервисытестырасширение с использованиемTestSetupUtilscort.
/resources/yocctests/log4j.properties	Файл конфигурации механизма ведения журнала.
/resources/yocctests/import/coredata/common	Содержит файлы ImpEx с тестовыми данными, загруженными в процессе инициализации клиента JUnit.

Для получения более подробной информации см.[Архитектура расширения OCC](#).

Коммерцияуслугибэк-офис Расширение

Thекоммерцияуслугибэк-офисрасширение содержит компоненты Backoffice Administration Cockpit, которые обеспечивают настройку моделей и функций, используемых в модуле Commerce Services.

О расширении

Имя	Каталог	Связанный модуль
коммерцияуслугибэк-офис	hybris/bin/модули/коммерческие-услуги	Модуль коммерческих услуг

Thекоммерцияуслугибэк-офисрасширение содержит определения виджетов, редакторы и действия, которые позволяют вам настроить перспективу «Организация продаж» в Backoffice. Виджет с перспективой «Организация продаж» определен вресурсы/перспектива/организация/организация-бэк-офис-виджеты.xmlле.

Расширение содержит мастера создания, такие как New Sales Unit, Organization Unit Employee, Consent Template и Promotion Restriction. Оно также предоставляет компоненты пользователя интерфейса в Backoffice Administration Cockpit, которые позволяют вам настраивать и искать модели данных, используемые в модуле Commerce Services. Эти компоненты определены вресурсы/коммерцияуслугиbackoffice-backoffice-config.xmlле и локализуются через файлы свойств в ресурсы/коммерцияуслугиbackoffice-backoffice-labelsnapка.

Зависимости

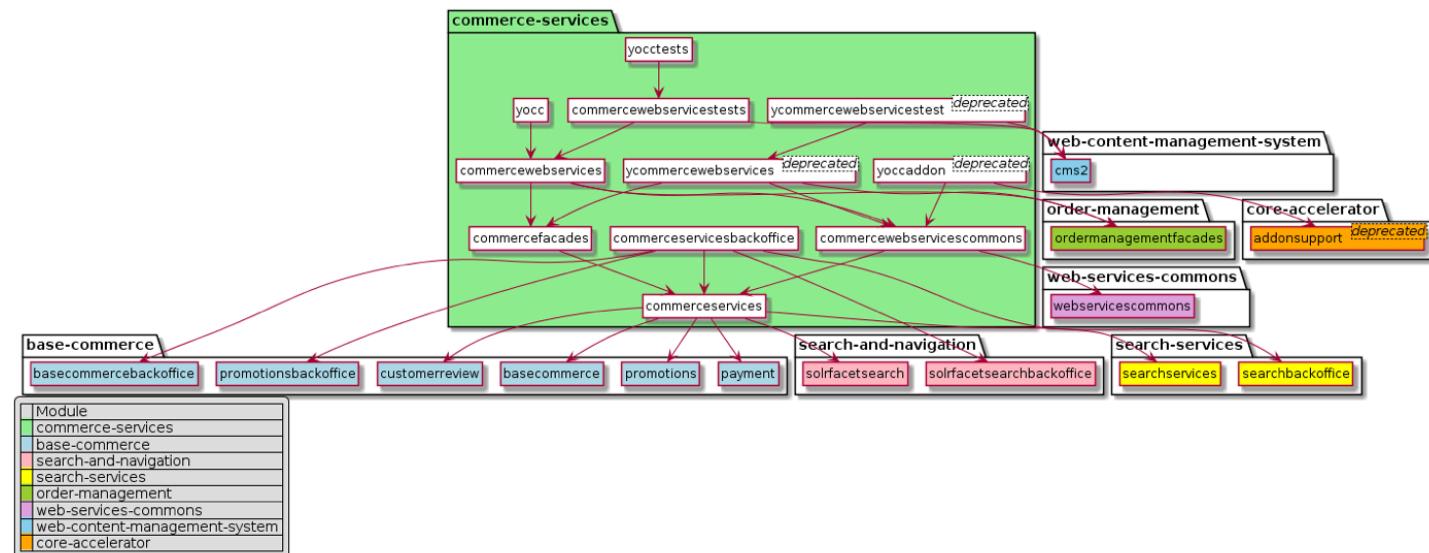


Диаграмма зависимостей

Реализация коммерческих услуг

В разделе объясняется, как реализовать функции модуля Commerce Services, включая управление корзиной, варианты оплаты и безопасность.

Конфигурируемые продукты

Поиск пользователя по свойству

СлужбаUserMatchingServiceищет пользователя на основе различных атрибутов. Эта служба используется фасадами и валидаторами для поддержки всех мест Commerce Web Services, гдеID пользователя приходит напрямую из URL как переменная пути или параметр запроса. Реализация этой службы

DefaultUserMatchingService, который содержит список стратегий типа Стратегия сопоставления свойств пользователя.[Вставка](#)

[Идентификаторов клиентов в вызовы API](#)

Вставка идентификатора клиента в вызовы OCC API позволяет агенту по обслуживанию действовать от имени клиента. Типичным вариантом использования этой функции является эмуляция действий клиента в модуле Assisted Service, который использует API OCC для поддержки клиентов.

[Объединение корзин](#)

Функция объединения корзин предназначена для обеспечения единообразного взаимодействия с корзиной в нескольких точках соприкосновения.

[Расширение CommerceCartService](#)

В этом разделе описывается, как расширить CommerceCartService.

[Расширение CommerceCheckoutService](#)

В этом разделе описывается, как расширить CommerceCheckoutService.

[Конвертеры и пополнители](#)

Объекты данных создаются из моделей или других объектов уровня сервиса с использованием преобразователей и пополнителей. Преобразователи создают новые экземпляры объектов данных и вызывают пополнители для заполнения этих объектов данных.

[Решатели ценностей](#)

Решатели значений являются более эффективной заменой текущих поставщиков значений.

[Заполнение списка клиентов в магазине данными устройств IoT](#)

Агент Assisted Service, работающий физически в магазине, может искать клиентов, которые в данный момент находятся в этом конкретном магазине. В настоящее время информация о том, какие клиенты присутствуют в данном магазине, моделируется и импортируется через файл ImpEx, но эти данные могут поступать из любого источника, включая устройство IoT.

[Коммерческие котировки](#)

Узнайте о занятиях в коммерции услуг, которые используются для функции коммерческих котировок.

[Функциональность группировки записей в корзине](#)

Функция группировки записей позволяет группировать несколько записей корзины в один пакет и работать с ним как с одним объектом.

[Погашение ваучера, проверка и обнаружение атак методом подбора](#)

Ваучеры выкупаются при оформлении заказа. Ранее реализация этой функции была возложена на партнера.

[Настройка коммерческих услуг](#)

Руководство по настройке коммерческих услуг.

[Интеграция с Adobe Experience Manager](#)

SAP Commerce легко интегрируется с Adobe Experience Manager (AEM). Это позволяет вам подключить решение SAP Commerce, отвечающее за управление данными о продуктах, корзинами покупок, оформлением заказов и выполнением заказов, в то время как AEM управляет отображением данных и маркетинговыми кампаниями.

[Удаление старых корзин с помощью Cronjobs](#)

Корзины не удаляются после закрытия сеанса. Таким образом, вы все еще можете восстановить корзину позже, когда пользователь снова войдет в сеть. Однако, чтобы избежать хранения большого количества старых корзин, вы можете использовать cronjob для их удаления.

[Расширение коммерческих услуг](#)

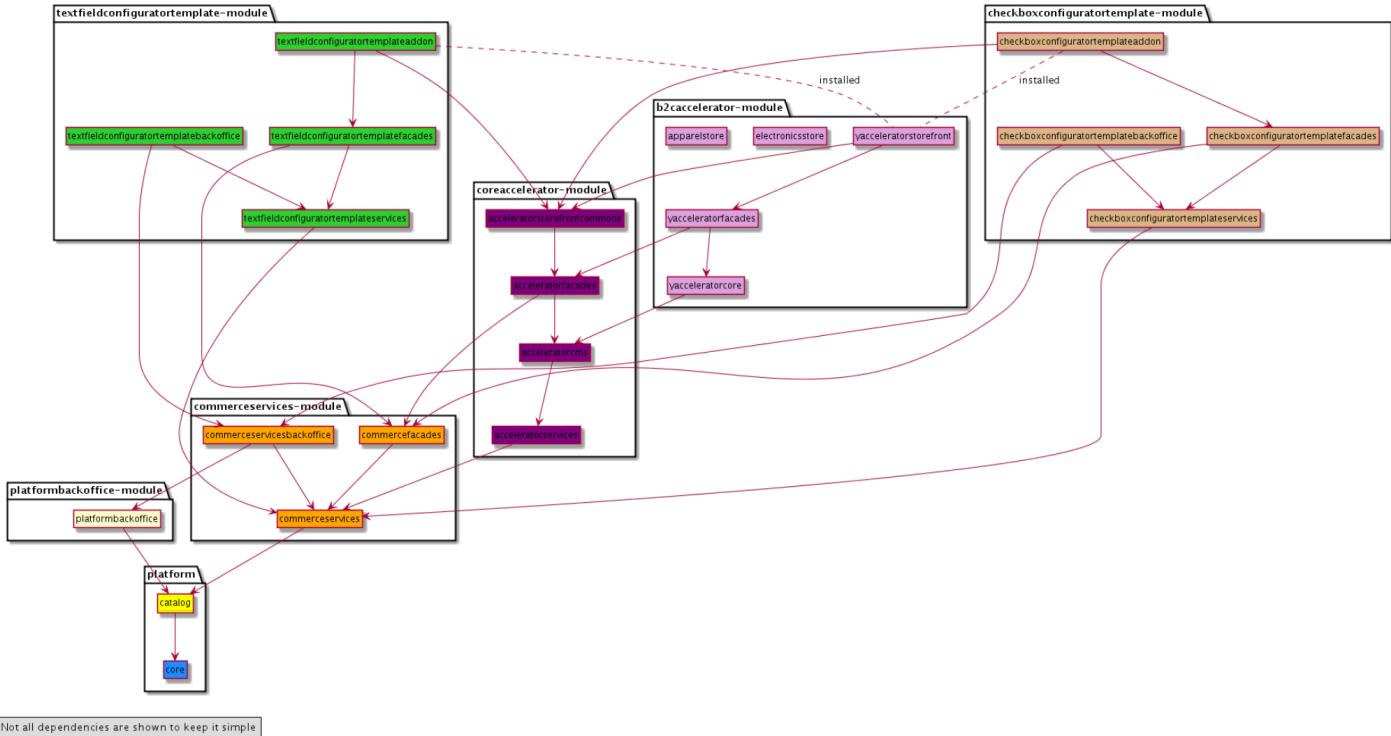
Коммерческие услуги можно расширить, создав AddOn с одним дополнительным Контроллером класса и использование соответствующих методов. Однако в большинстве случаев требуется более сложные изменения для создания AddOn, который добавляет функциональность к Commerce Services.

Конфигурируемые продукты

Зависимости расширения

Узнайте о зависимостях расширений для настраиваемых продуктов.

Эта диаграмма UML дает общий обзор наиболее важных зависимостей расширений.



Примеры настраиваемых категорий

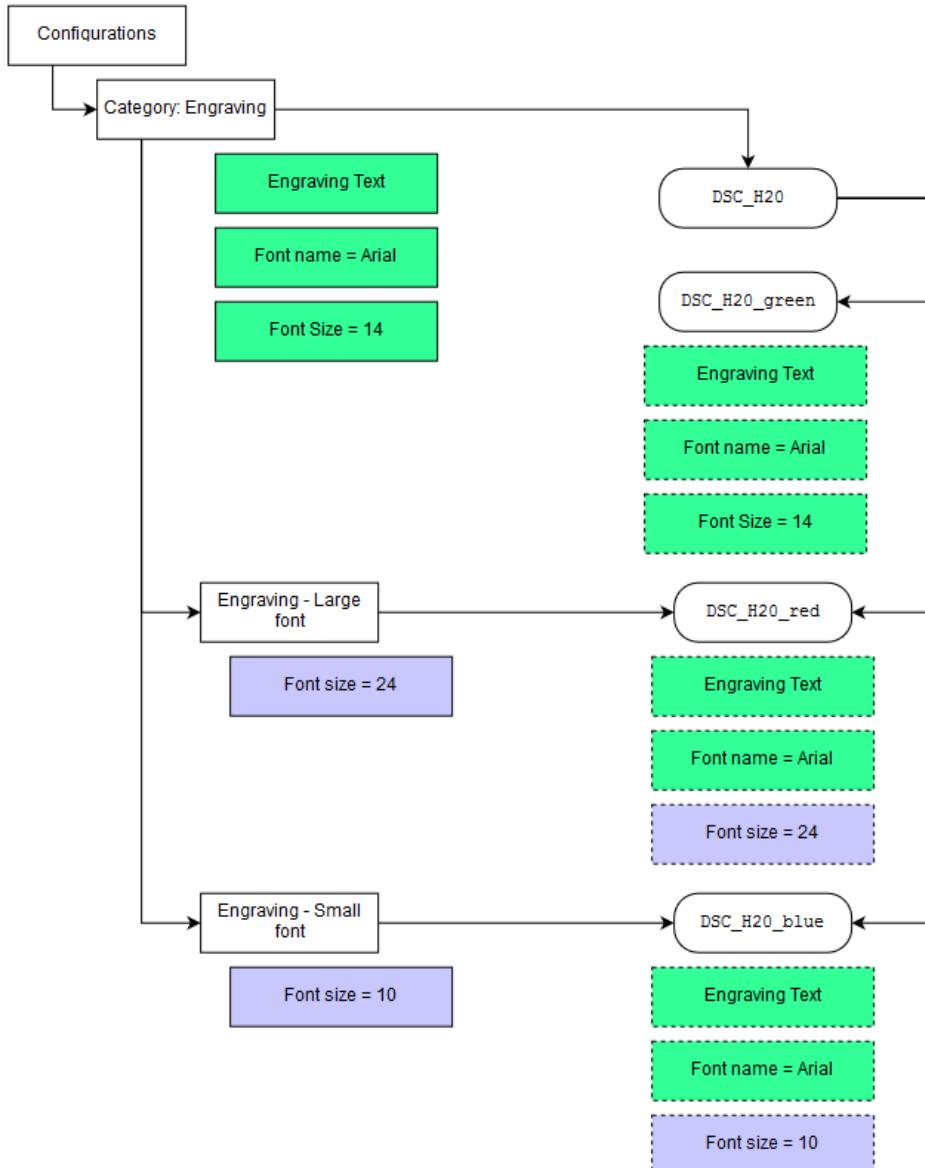
В зависимости от конфигурации варианта продукта, категория варианта может наследовать, переопределять или объединять конфигурацию из базовой категории продукта.

Вариант 1 — Категория варианта наследует конфигурацию и переопределяет некоторые настройки

Родительская категория также назначается базовому продукту, поэтому по умолчанию варианты продукта имеют конфигурацию базового продукта.

Categories

Products

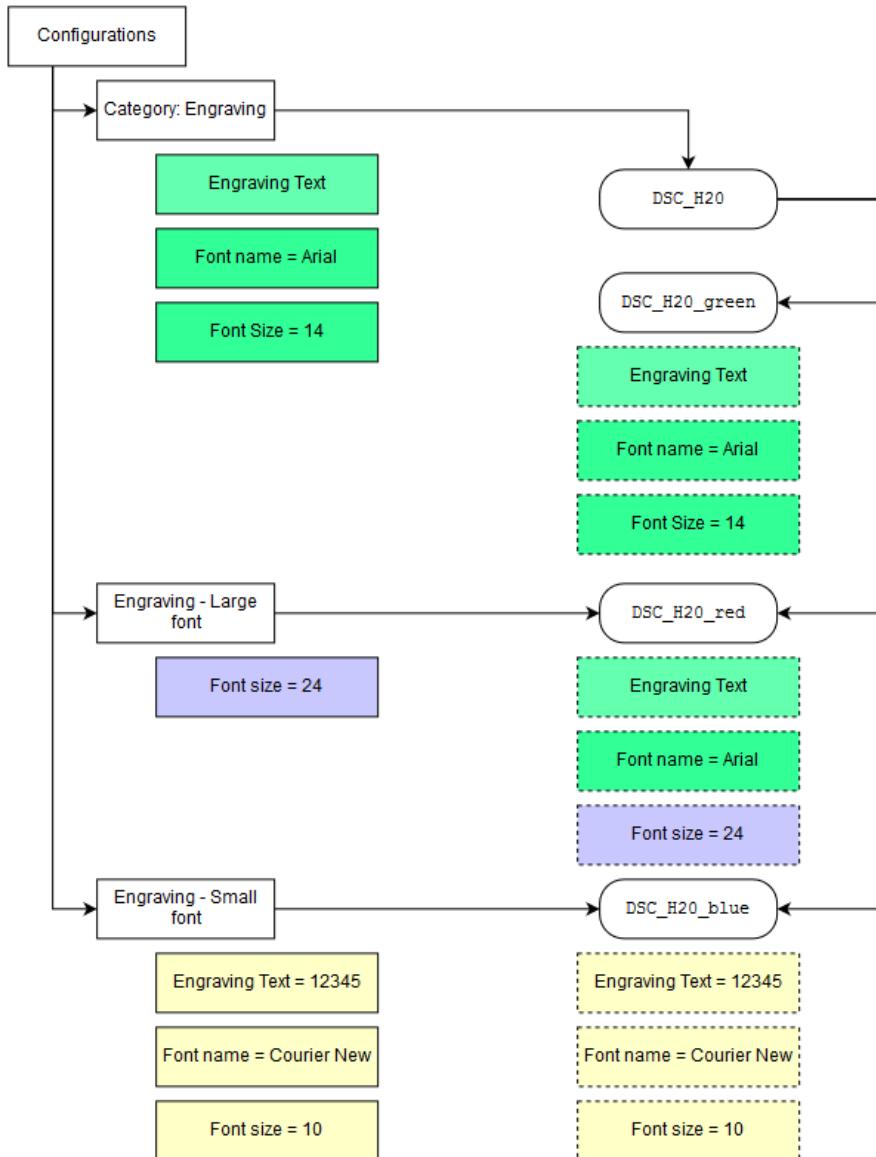


Вариант 2 — Простая структура категорий

Вариант объединяет конфигурации базового продукта с его собственным набором. Этот набор имеет приоритет в случае конфликтов. Если конфликты возникают, когда базовый продукт и вариант имеют категорию одного и того же типа, версия варианта полностью переопределяет базовую версию.

Categories

Products

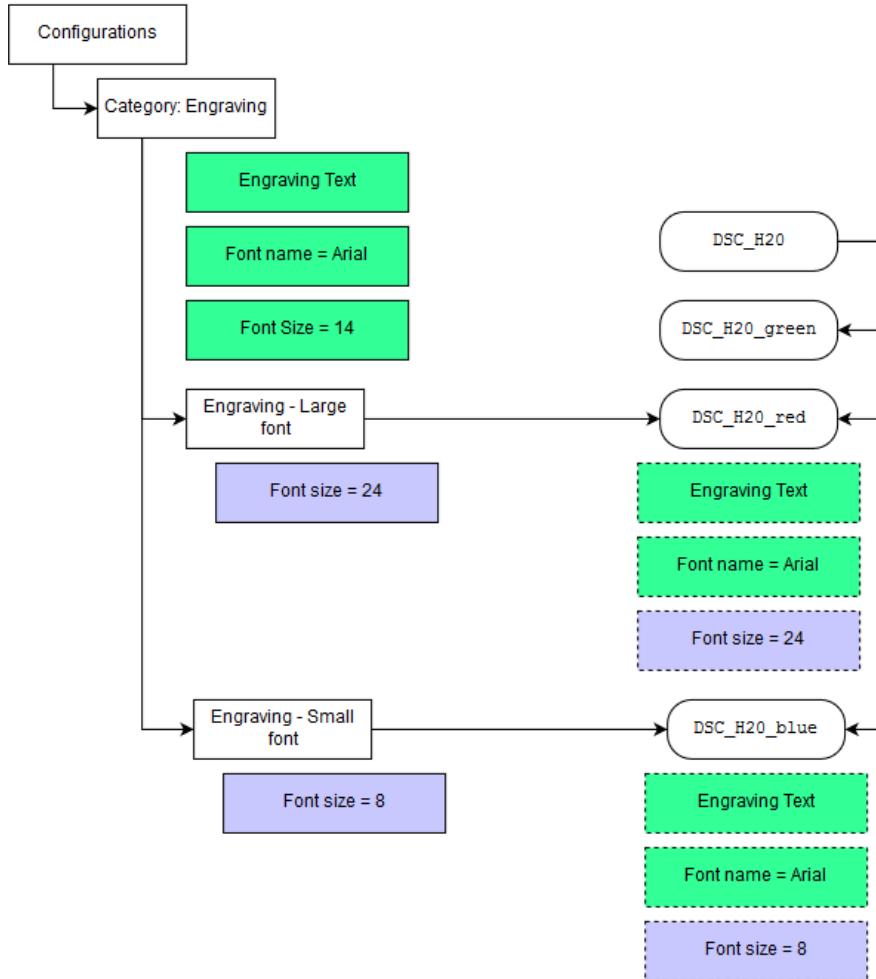


Вариант 3 — Базовый продукт не имеет конфигураций

Базовый продукт не имеет конфигураций, а конфигурации определяются для вариантов напрямую. В примере конфигурации для категории Гравировка объединены с конфигурациями, специфичными для варианта.

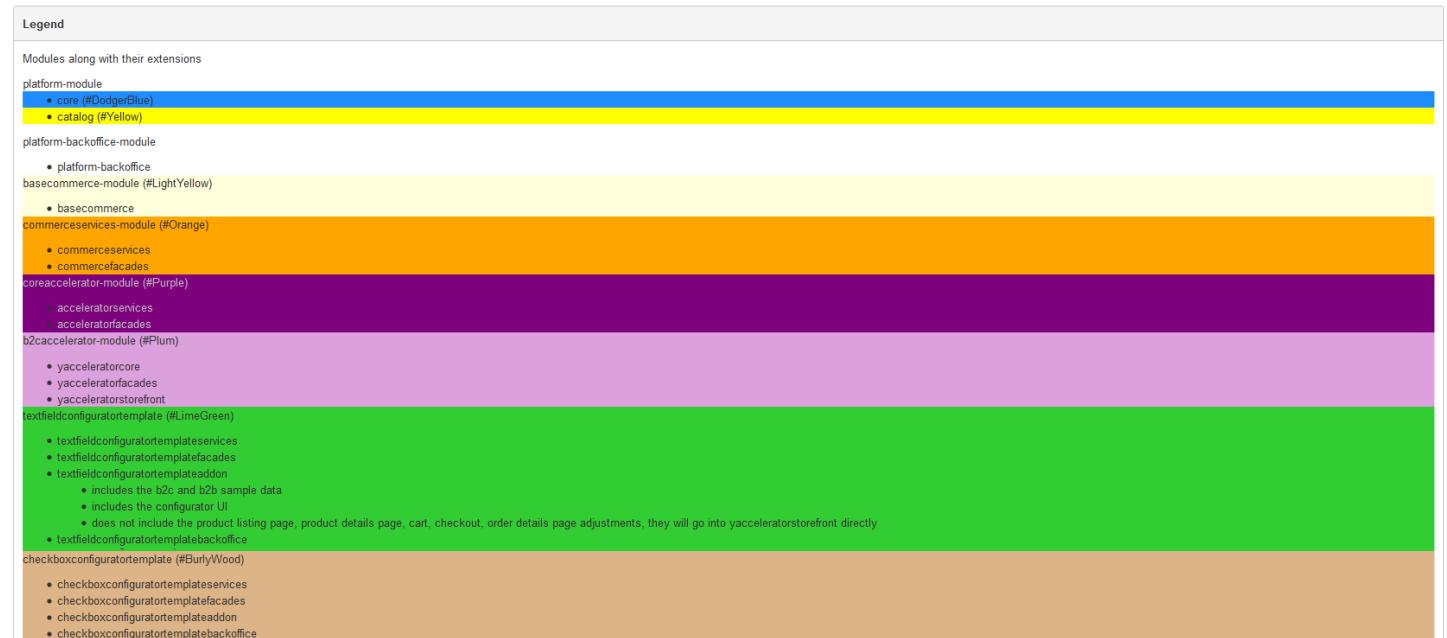
Categories

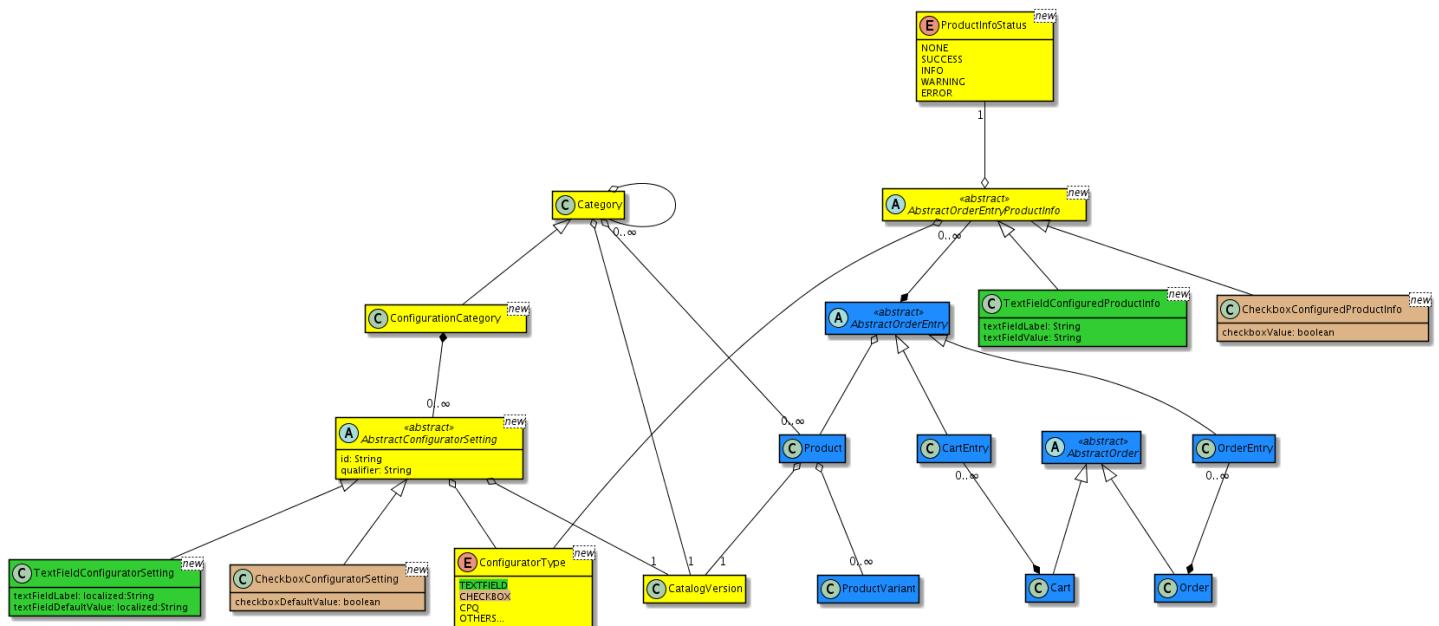
Products



Уровень обслуживания

Предметы

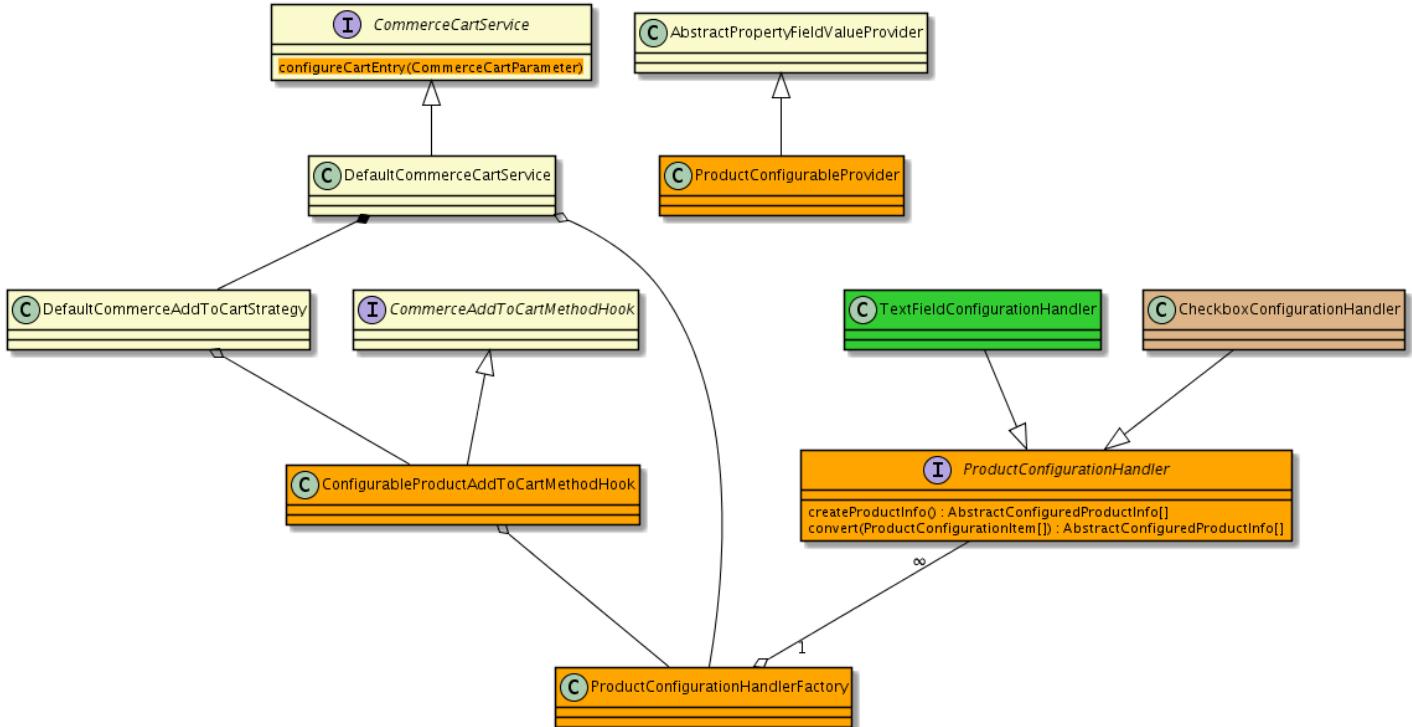




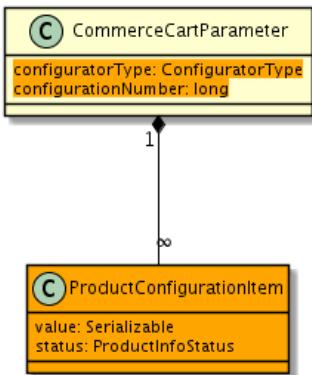
Услуги

В классы и интерфейсы внесены следующие изменения:

- CommerceCartService: добавляет дополнительный метод <updateCartEntry>, который может изменить один АннотацияНастроенныйИнформация о продукте в данной записи корзины.
- DefaultCommerceAddToCartStrategy: добавляет к новой записи экземпляры АннотацияНастроенныйИнформация о продукте согласно списку АннотацияКонфигураторНастройки представлена в продукте.
- Новый интерфейс -ProductConfigurationHandler: инкапсулирует логику, специфичную для типа конфигуратора, на уровне сервиса. ProductConfigurationHandlerFactory направляет вызовы к правильной реализации ProductConfigurationHandler.

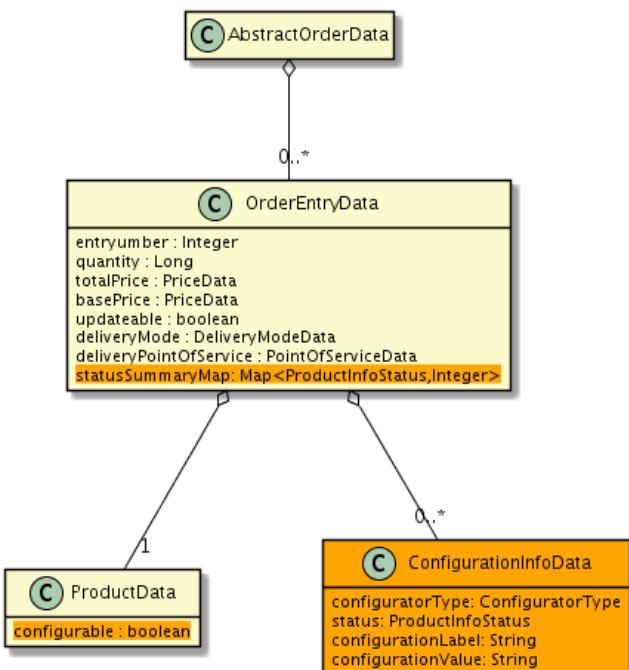


Промежуточные DTO

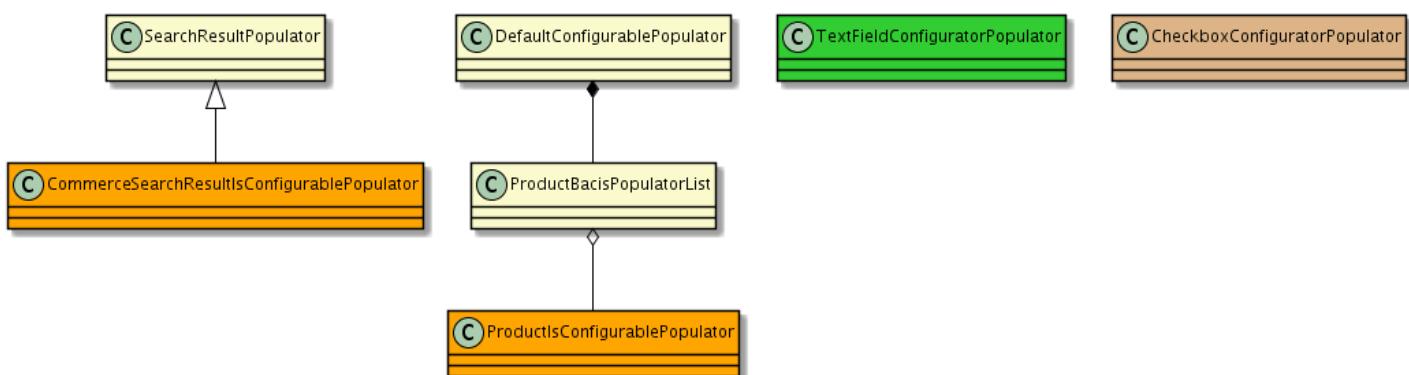


Фасадный слой

DTO



Заполнители фасадов



Сопутствующая информация

[Расширение texteldconjurortemplateservices](#)
[texteldconjurortemplatebackoffice](#) [Расширение](#)
[texteldconjurortemplatefacades](#) [Расширение](#)
[texteldconjurortemplateaddon](#) [Расширение](#)
[Configurable Products Trail](#)

Поиск пользователя по свойству

Служба userMatchingService ищет пользователя на основе различных атрибутов. Эта служба используется фасадами и валидаторами для поддержки всех мест Commerce Web Services, где ID пользователя приходит напрямую из URL как переменная пути или параметр запроса. Реализация этой службы DefaultUserMatchingService, который содержит список стратегий типа Стратегия сопоставления свойств пользователя.

Сопоставление пользователей в коммерческих услугах

В Commerce Services существуют две реализации этого интерфейса стратегии:

- userUIDMatchingStrategy - эта стратегия ищет разные типы пользователей uid Атрибут. Он использует пользователь Сервис
- customerIdMatchingStrategy - эта стратегия ищет клиентов идентификатор клиента Атрибут. Он использует обслуживание клиентов

Каждая стратегия ищет пользователя, используя сопоставление на основе одного конкретного атрибута, например id, идентификатор клиента. Этот атрибут должен быть уникальным среди искомых пользователей.

Стратегия также должна гарантировать, что тип возвращаемого объекта правильный. Если он отличается от типа, требуемого методом, стратегия возвращает пустой java.util. Необязательно результат. Сопоставление выполняется в порядке стратегий в списке и заканчивается, когда находится первый пользователь. В случае, если пользователь не найден для данной стратегии, выбирается и проверяется следующая стратегия из списка. Если для данного атрибута в данной стратегии найдено более одного пользователя, выдается исключение.

По умолчанию userMatchingService использует только userUIDMatchingStrategy.

Список стратегий, используемых службой, можно настроить в userMatchingService определение в commerceServices-весна.xml ле.

```
<псевдоним alias="userMatchingService" name="defaultUserMatchingService">
<боб id="defaultUserMatchingService" class="de.hybris.platform.commerceservices.user.impl.DefaultUserMatchingService">
    <свойство имя="соответствующие стратегии">
        <список>
            <реф bean="userUIDMatchingStrategy"/>
        </список>
    </свойство>
</боб>
```

Сопоставление клиентов в Commerce Web Services

Существует множество случаев, когда требуется, чтобы URL имел атрибут ID пользователя как переменная пути или как параметр запроса в Commerce Web Services. Система ищет пользователя на основе заданного значения ID пользователя Атрибут. В Commerce Services идентификатор для ID пользователя Атрибут — это адрес электронной почты пользователя, который хранится как uid собственность пользователя.

Избегайте использования конфиденциальных данных явно в URL запроса. Можно настроить и использовать другие атрибуты пользователя в качестве ID пользователя параметр в URL REST API. Одним из таких атрибутов является идентификатор клиента Атрибут. Он уникален среди всех клиентов и генерируется во время регистрации клиента. Для сопоставления пользователей используйте универсальный уникальный идентификатор, UUID, формат для идентификатора клиента Атрибут.

Примечание

Использование идентификатора клиента Атрибут не нарушает существующий REST API. Существующее использование адреса электронной почты в uid Атрибут продолжает работать. Также возможно изменить конфигурацию для идентификации пользователя по другому атрибуту, например, по номеру телефона. При регистрации нового пользователя и авторизации на сервере OAuth2 в качестве идентификатора по-прежнему используется e-mail.

В Commerce Web Services список стратегий включает стратегию customerIdMatchingStrategy. Он использует службу, которая реализует Обслуживание клиентов Интерфейс. Этот сервис позволяет найти клиента, если идентификатор клиента имеет допустимый формат UUID. В противном случае возвращается пустой java.util. Необязательно результат. Если клиент не найден, то процесс поиска продолжается с использованием следующих стратегий сопоставления.

UUID проверяется на соответствие регулярному выражению, указанному в определении сервиса.

```
<псевдоним alias="customerService" name="defaultCustomerService">
<боб id="defaultCustomerService" class="de.hybris.platform.commerceservices.customer.impl.DefaultCustomerService">
    <конструктор-аргумент ref="customerDao"/>
    <конструктор-аргумент type="java.lang.String" value="^[0-9a-f]{8}-[0-9a-f]{4}-[1-5][0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}$"/>
</боб>
```

Вы можете искать клиентов по идентификатору клиента и затем по uid. Эта конфигурация используется для конечных точек REST API, где ID пользователя берется непосредственно из переменной пути или параметра запроса и используется для поиска пользователя. Использование идентификатора клиента помогает избежать использования конфиденциальных данных, таких как адрес электронной почты, в URL-адресах.

Для настройки пользовательского поиска создан новый компонент userMatchingService добавлено - wsUserMatchingService. Список стратегий в этой услуге расширен customerIdMatchingStrategy.

Порядок стратегий в списке имеет значение. CustomerIdMatchingStrategy появляется выше в списке и имеет приоритет над userUIDMatchingStrategy. Это означает, что процесс поиска пользователя сначала ищет клиента с идентификатором клиента Атрибут. Если он не может найти такого клиента, он пытается найти пользователя с uid Атрибут.

```

<псевдоним имя="defaultWsUserMatchingService" псевдоним="wsUserMatchingService"/>
<боб идентификатор="defaultWsUserMatchingService" родительский="UserMatchingService">
    <свойство имя="соответствующие стратегии">
        <список>
            <реф bean="customerIdMatchingStrategy"/>
            <реф bean="userUIDMatchingStrategy"/>
        </список>
    </свойство>
</боб>

```

Фасады и валидаторы, расположенные в коммерческих службах, которые вводят userMatchingService и используются не только Commerce Web Services. Существуют отдельные определения bean фасадов и валидаторов в Commerce Web Services, которые требуют использования wsUserMatchingService.

Фасады находятся в commercewebservices-spring.xml:

- wsUserFacade
- wsCustomerFacade
- wsCustomerGroupFacade

Валидаторы находятся в validator-v2-spring.xml:

- wsPrincipalListDTOValidator
- wsUserGroupDTOValidator

Примеры

Запрос на получение данных пользователя для магазина электроники на локальном хосте имеет следующий шаблон:

ПОЛУЧИТЬ <https://localhost:9200/rest/v2/electronics/users/{userId}>

В примерах предполагается, что клиент существует и его id является john.doe@test.com и его идентификатор клиента это 6a2a41a3-c54c-4ce8-a2d2-0324e1c32a21.

Вот два примера использования ID пользователя Параметр URL: как пользователь uid или как идентификатор клиента параметр заказчика.

- uid как ID пользователя:

Когда вы используете uid атрибут пользователя как ID пользователя Параметр URL, эффективный URL:

<https://localhost:9002/rest/v2/electronics/users/ john.doe@test.com>

На стороне сервера, wsUserMatchingService сначала пытается найти клиента, используя customerIdMatchingStrategy, но он терпит неудачу из-за неверного формата UUID идентификатор клиента атрибут. При следующей попытке, userUID Соответствие Стратегия используется и находит клиента.

- идентификатор клиента как ID пользователя:

Когда вы используете идентификатор клиента атрибут пользователя как ID пользователя Параметр URL, эффективный URL:

<https://localhost:9002/rest/v2/electronics/users/6a2a41a3-c54c-4ce8-a2d2-0324e1c32a21>

На стороне сервера, wsUserMatchingService находит клиента с первой попытки, используя customerIdMatchingStrategy.

Вставка идентификаторов клиентов в вызовы API

Вставка идентификатора клиента в вызовы OCC API позволяет агенту по обслуживанию действовать от имени клиента. Типичным вариантом использования этой функции является эмуляция действий клиента в модуле Assisted Service, который использует API OCC для поддержки клиентов.

Агент сервиса, эмулирующий клиента, должен иметь роль CUSTOMERMANAGERGROUP, которая предоставляется с помощью аутентификации OAuth2. Для эмуляции клиента в запросе API указывается его идентификатор. Это правильно идентифицирует его и назначает его текущим пользователем для вызова API. То, как вы вставляете идентификатор клиента, зависит от типа вызова API.

URL-адреса конечных точек, включающие идентификатор клиента

Для URL-адресов конечных точек, которые включают идентификатор клиента, например, когда агент службы извлекает корзины пользователей, укажите идентификатор клиента непосредственно в параметре пути запроса следующим образом:

`occ/v2/{baseSiteId}/users/{customerId}/carts`

В этом случае клиент всегда идентифицируется с помощью параметра пути запроса, а любой идентификатор клиента, указанный в заголовках HTTP, игнорируется.

URL-адреса конечных точек, не содержащие идентификатор клиента

Для конечных точек, которые не включают идентификатор клиента в путь запроса, например, когда агент службы ищет продукты от имени клиента, укажите идентификатор клиента в заголовках HTTP с помощью `sap-commerce-cloud-user-id` следующим образом:

```
sap-commerce-cloud-user-id: {customerId}
```

Объединение корзин

Функция объединения корзин предназначена для обеспечения единообразного взаимодействия с корзиной в нескольких точках соприкосновения.

Ниже приведен типичный пример использования объединения корзин: клиент входит в интернет-магазин и сохраняет товары в своей корзине. Позже клиент возвращается в интернет-магазин, не входя в систему, и решает добавить товары в новую корзину. Теперь, если клиент решает завершить покупку, он должен войти в свою учетную запись, и в этот момент функция объединения корзин гарантирует, что товары, ранее сохраненные в корзине клиента, будут объединены с товарами, которые клиент только что добавил перед входом в систему.

Стратегия объединения корзин

Функциональность реализована в стратегии, называемой **CommerceCartMergingStrategy**, который был добавлен в коммерцияуслуграсширение. Он состоит из одного метода, **mergeCarts**. Метод требует следующих параметров:

- изCart -этот параметр указывает, из какой корзины объединять записи.
- в корзину -этот параметр указывает, в какую корзину следует объединять записи.
- модификации -этот параметр указывает список модификаций, к которым будут добавлены новые модификации, созданные стратегией слияния

`CommerceCartMergingStrategy.java`

```
/**
 * Стратегия объединения корзин.
 */
publicличный интерфейс CommerceCartMergingStrategy {

    /**
     * Объединить две корзины и добавить модификации
     *
     * @param изCart
     *          - Корзина из слияния сделана вCart
     * @param
     *          - Корзина для объединения с
     * @param
     *          - Модификациями
     * @param
     *          - Список модификаций
     * @param броски CommerceCartMergingException
     */
    void mergeCarts(CartModel fromCart, CartModel toCart, List<CommerceCartModification> модификации) бросает CommerceCar }
```

Реализация по умолчанию повторяется для каждой записи в корзине `CartModel`объект и добавляет его в цель `CartModel`объект. Он использует **CommerceAddToCartСтратегия** добавить запись корзины, чтобы включить все функции, такие как проверка уровня запасов. При слиянии стратегия создает список изменений, внесенных в целевую корзину.

-Примечание

Реализация по умолчанию также удаляет корзину, из которой было выполнено слияние, поэтому ее использование после слияния будет невозможно.

Применение в торговлефасады

В коммерцияфасады, Объединение корзин реализуется следующими методами: восстановить `CartAndMerge` и восстановить `AnonymousCartAndMerge`. Оба метода объединяют корзины с помощью **CommerceCartMergingStrategy**, и оба метода выполняют восстановление корзины после объединения корзин. Единственное различие между методами заключается в том, что восстановить `CartAndMerge` требует в качестве параметров две корзины, принадлежащие одному и тому же пользователю, в то время как восстановить `AnonymousCartAndMerge` методу требуется параметр анонимной корзины, из которой будет выполняться слияние.

Расширение `CommerceCartService`

В этом разделе описывается, как расширить `CommerceCartService`.

-Примечание

Расширение SAP Commerce может предоставлять функциональность, лицензируемую через различные модули SAP Commerce. Обязательно ограничьте реализацию функциями, определенными в вашей контрактной лицензии. В случае сомнений обратитесь к своему торговому представителю.

Параметр Объекта

Параметр объекта `CommerceCartService` является единственным параметром для большинства методов в `CommerceCartService`. Этот объект определен как боб в `/resources/commerceservices-beans.xml`, следующее:

```
<bean class="de.hybris.platform.commerceservices.service.data.CommerceCartParameter">
    <property name="cart" type="de.hybris.platform.core.model.order.CartModel"> <description>Модель тележки</description>
    </свойство>
    <property name="product" type="de.hybris.platform.core.model.product.ProductModel">
        <description>ProductModel</description>
    </свойство>
    <property name="quantity" type="long">
        <description>Количество для добавления</description>
    </property>
    <property name="unit" type="de.hybris.platform.core.model.product.UnitModel"> <description>Единицы измерения для добавления</description>
    </свойство>
    ...
</боб>
```

Клиенту этой службы не нужно перегружать методы, когда есть необходимость передать собственные параметры существующему методу. Вместо этого вы добавляете еще одно свойство в Параметр CommerceCartбоб.xmlле вашего расширения. Ниже приведен пример:

```
<bean class="de.hybris.platform.commerceservices.service.data.CommerceCartParameter">
    <property name="customProperty" type="String">
        <description>Мое пользовательское свойство</description>
    </property>
    ...
</боб>
```

Стратегии по умолчанию

Как правило, каждый из методов CommerceCartService делегирует работу стратегии по умолчанию. Эти стратегии находятся в de.hybris.platform.commerceservices.order.impl пакет, как следует:

Некоторые из стратегий по умолчанию:

```
DefaultCommerceCartРасчетСтратегия
DefaultCommerceAddToCartStrategy
DefaultCommercePaymentAuthorizationStrategy
DefaultCommercePaymentInfoStrategy
DefaultCommerceCartRestorationStrategy
DefaultCommercePlaceOrderStrategy
DefaultCommerceCartHashCalculationStrategy
DefaultCommerceUpdateCartEntryStrategy
DefaultCommercePaymentProviderStrategy
DefaultCommerceDeliveryAddressStrategy
DefaultCommerceCartSplitStrategy
DefaultCommerceCartEstimateTaxesStrategy
DefaultCommerceDeliveryModeStrategy
DefaultCommerceDeliveryModeValidationStrategy
DefaultCheckoutCartCalculationStrategy
DefaultCommerceRemoveEntriesStrategy
```

Метод Хуки

Иногда необходимо иметь хук в методе, когда вы хотите выполнить некоторую пользовательскую логику перед методом, после метода и, возможно, перед каким-либо другим событием в методе.

Ниже приведены некоторые из методов, используемых в CommerceCartService поддерживющие крючки, которые находятся в de.hybris.platform.commerceservices.order.hook упаковка:

Интерфейс Hook поставляется сразу из коробки

АвторизоватьПлатежныйМетодХук

```
CommerceAddToCartMethodHook
CommerceCartCalculationMethodHook
CommercePlaceOrderMethodHook
```

CommerceAddToCartМетодХук

Следующий блок кода показывает CommerceAddToCartМетодХук определение интерфейса, которое вам необходимо реализовать для подключения к добавить в корзину метод.

de.hybris.platform.commerceservices.order.hook.CommerceAddToCartMethodHook

```
публичный интерфейс CommerceAddToCartMethodHook
{
    void beforeAddToCart(параметры CommerceCartParameter) выдает CommerceCartModificationException void
    afterAddToCart(параметры CommerceCartParameter, результат CommerceCartModification) выдает }
```

Конфигурация пружины для хуков методов

Боб с идентификатором commerceAddToCartMethodHooks определено в /ресурсы/commerceservices-spring.xml, следующее:

```
<util:list id="commerceAddToCartMethodHooks" value-type="de.hybris.platform.commerceservices.order.hook.CommerceAddToCartMethodHook"/>
```

Вам необходимо реализовать CommerceAddToCartMethodHook интерфейс и псевдоним commerceAddToCartMethodHooksbean с вашей реализацией. Вы можете предоставить несколько хуков, если хотите.

Вы могли бы использовать ListMergeDirective для добавления элементов в commerceAddToCartMethodHooksbean и контролировать их порядок, если необходимо. Для получения дополнительной информации см. [Расширение commerceservices](#).

```
<bean id="customCommerceAddToCartMethodHookMergeDirective" зависит от="commerceAddToCartMethodHooks" parent="listMergeDirective">
    <имя_свойства="add" ref="customCommerceAddToCartMethodHook"/> </bean>
```

Конфигурация выше добавит customCommerceAddToCartMethodHook в commerceAddToCartMethodHooks список.

Это будет делаться для каждого вызова de.hybris.platform.commerceservices.order.impl.DefaultCommerceCartService#addToCart(CommerceCartParameter) метод CommerceAddToCartMethodHook#doAddToCart будет выполнен и добавить в корзину, и CommerceAddToCartMethodHook#afterAddToCart побежит за ним.

Отключение перехватов методов во время выполнения

Если по какой-либо причине вы не хотите, чтобы подключенные методы-хуки выполнялись во время выполнения, вы можете установить de.hybris.platform.commerceservices.service.data.CommerceCartParameter#enableHooks значение false при вызове методов сервиса.

Однако для этого вам нужно будет перезаписать метод фасада, который вызывает метод сервиса. Другой способ отключить хуки — установить ключ свойства вЛОЖЬ после конвенции

коммерция услуги.<НАЗВАНИЕ ИНТЕРФЕЙСА КРЮКА>.включенонапример commerceservices.commerceaddtocartmethodhook.enabled

По умолчанию хуки включены.

ключи свойств для отключения или включения методов-хуков

```
commerceservices.authorizepaymentmethodhook.enabled=true
        commerceservices.commerceaddtocartmethodhook.enabled=true
        commerceservices.commercecartcalculationmethodhook.enabled=true
        commerceservices.commerceplaceordermethodhook.enabled=true
        commerceservices.commerceupdatecartentryhook.enabled=true
```

Валидаторы

AddToCartВалидатор

Этот интерфейс позволяет вам определять конкретные проверки, которые выполняются во время процесса «добавить в корзину». Он содержит следующие методы:

поддерживает логические значения (конечный параметр CommerceCartParameter);

void validate(конечный параметр CommerceCartParameter) выдает исключение CommerceCartModificationException;

Специфическая проверка в подтвердитьМетод вызывается, если поддерживает метод возвращает истиинный для заданных параметров. В противном случае он игнорируется. подтвердитьМетод бросает CommerceCartModificationException если параметры не приняты. Если нет исключений, параметры принимаются и процесс добавления в корзину продолжается.

Чтобы зарегистрировать пользовательский валидатор корзины, выполните следующие действия:

. Создайте новый класс, который реализует AddToCartВалидатор.

. Зарегистрируйте класс как компонент в <вашерасширение>-весна.xml следующее:

```
<bean id="customBeanId"
    class="com.custom.validators.impl.CustomValidator"> </bean>
```

. Добавьте его в список валидаторов с помощью директивы merge, ссылаясь на список addToCartВалидаторы.

```
<bean id="customAddToCartValidatorsMergeDirective" зависит от="addToCartValidators"
    родитель="listMergeDirective">
    <имя_свойства="add" ref="customBeanId" /> </bean>
```

Сопутствующая информация

[Ускорители](#)

Расширение CommerceCheckoutService

В этом разделе описывается, как расширить CommerceCheckoutService.

Примечание

Расширение SAP Commerce может предоставлять функциональность, лицензируемую через различные модули SAP Commerce. Обязательно ограничьте реализацию функциями, определенными в вашей контрактной лицензии. В случае сомнений обратитесь к своему торговому представителю.

Параметр Объекта

Параметр объекта `CommerceCheckoutParameter` является единственным параметром для большинства методов в `CommerceCheckoutService`. Этот объект определен как боб в `/resources/commerceservices-beans.xml`, следующее:

`/resources/commerceservices-beans.xml`

```
<bean class="de.hybris.platform.commerceservices.service.data.CommerceCheckoutParameter">
    <property name="cart" type="de.hybris.platform.core.model.order.CartModel"> <description>Модель тележки</description>
    </свойство>
    <property name="address" type="de.hybris.platform.core.model.user.AddressModel"> <description>Адрес</description>
    </свойство>
    <property name="isDeliveryAddress" тип="логическое">
        <description>Адрес доставки </property> флаг</description>
    </property>
    <property name="deliveryMode" type="de.hybris.platform.core.model.order.delivery.DeliveryModeModel"> <description>тип доставки</description> </property>
    <property name="paymentInfo" type="de.hybris.platform.core.model.order.payment.PaymentInfoModel"> <description>информация о платеже</description>
    </свойство>
    ...
</боб>
```

Клиенту этого сервиса не нужно перегружать методы, когда есть необходимость передать собственные параметры какому-либо существующему методу. Вместо этого вы добавляете еще одно свойство к `CommerceCheckoutParameter` в `боб.xml` вашего расширения. Ниже приведен пример:

`beans.xml`

```
<bean class="de.hybris.platform.commerceservices.service.data.CommerceCheckoutParameter">
    <property name="customProperty" type="String">
        <description>Мое пользовательское свойство</description>
    </property>
    ...
</боб>
```

Стратегии по умолчанию

Как правило, каждый из методов `CommerceCartService` делегирует работу стратегии по умолчанию. Эти стратегии находятся в `de.hybris.platform.commerceservices.order.impl` и, как следует:

Некоторые из стратегий по умолчанию:

DefaultCommerceCartРасчетСтратегия

- DefaultCommerceAddToCartStrategy
- DefaultCommercePaymentAuthorizationStrategy
- DefaultCommercePaymentInfoStrategy
- DefaultCommerceCartRestorationStrategy
- DefaultCommercePlaceOrderStrategy
- DefaultCommerceCartHashCalculationStrategy
- DefaultCommerceUpdateCartEntryStrategy
- DefaultCommercePaymentProviderStrategy
- DefaultCommerceDeliveryAddressStrategy
- DefaultCommerceCartSplitStrategy
- DefaultCommerceCartEstimateTaxesStrategy
- DefaultCommerceDeliveryModeStrategy
- DefaultCommerceDeliveryModeValidationStrategy
- DefaultCheckoutCartCalculationStrategy
- DefaultCommerceRemoveEntriesStrategy

Метод Хуки

Иногда необходимо иметь хук в методе, когда вы хотите выполнить некоторую пользовательскую логику перед методом, после метода и, возможно, перед каким-либо другим событием в методе.

Ниже приведены некоторые из методов, используемых в `CommerceCheckoutService` поддерживающие крючки, которые находятся в `de.hybris.platform.commerceservices.order.hook` пакета:

Интерфейс Hook поставляется «из коробки»

АвторизоватьПлатежныйМетодХук

CommercePlaceOrderMethodHook

CommercePlaceOrderMethodHook

Следующий блок кода показывает CommercePlaceOrderMethodHook определение интерфейса, которое вам необходимо реализовать для подключения к разместить Заказать метод.

de.hybris.platform.commerceservices.order.hook.CommercePlaceOrderMethodHook

```
публичный интерфейс CommercePlaceOrderMethodHook
{
    void afterPlaceOrder(параметр CommerceCheckoutParameter, CommerceOrderResult orderModel); void
    beforePlaceOrder(параметр CommerceCheckoutParameter);
    void beforeSubmitOrder(параметр CommerceCheckoutParameter, результат CommerceOrderResult); }
```

Конфигурация пружины для хуков методов

Боб с идентификатором коммерция PlaceOrderMethodHooks определено в /ресурсы/commerceservices-spring.xml, следующее:

```
<util:list id="commercePlaceOrderMethodHooks" value-type="de.hybris.platform.commerceservices.order.hook.CommercePlaceOrderMethodHook"
```

Вам необходимо реализовать CommercePlaceOrderMethodHook интерфейс и псевдоним коммерция PlaceOrderMethodHooks bean с вашей реализацией. Вы можете предоставить несколько хуков, если хотите.

Вы могли бы использовать ListMergeDirective для добавления элементов в коммерция PlaceOrderMethodHooks bean и контролировать их порядок, если необходимо. Для получения дополнительной информации см. [Расширение commerceservices](#).

```
<bean id="customCommercePlaceOrderMethodHooksMergeDirective" зависит от="commercePlaceOrderMethodHooks" parent="listMergeDirective">
    <имя_свойства="add" ref="customCommercePlaceOrderMethodHook"/> </bean>
```

Конфигурация выше добавит customCommercePlaceOrderMethodHook боб к коммерция PlaceOrderMethodHooks список бобов.

Это будет делаться для каждого вызова

de.hybris.platform.commerceservices.order.impl.DefaultCommerceCheckoutService#placeOrder(CommerceCheckoutParameter):

- ThebeforePlaceOrder(параметр CommerceCheckoutParameter) будет казнен до того, как разместить Заказать метод
- TheafterPlaceOrder(параметр CommerceCheckoutParameter, CommerceOrderResult orderModel) будет казнен в окончательном блоке разместить Заказать метод
- ThebeforeSubmitOrder(параметр CommerceCheckoutParameter, результат CommerceOrderResult) будет выполнен непосредственно перед вызовом de.hybris.platform.order.OrderService#submitOrder

Отключение перехватов методов во время выполнения

Если по какой-либо причине вы не хотите, чтобы подключенные методы-хуки выполнялись во время выполнения, вы можете установить

de.hybris.platform.commerceservices.service.data.CommerceCartParameter#enableHooks значение false при вызове методов сервиса.

Однако для этого вам нужно будет перезаписать метод фасада, который вызывает метод сервиса. Другой способ отключить хуки — установить ключ свойства вЛОЖЬ после конвенции

коммерция услуги.<НАЗВАНИЕ ИНТЕРФЕЙСА КРЮКА>.включенонапример commerceservices.commerceaddtocartmethodhook.enabled

По умолчанию хуки включены.

ключи свойств для отключения или включения методов-хуков

```
commerceservices.authorizepaymentmethodhook.enabled=true
commerceservices.commerceaddtocartmethodhook.enabled=true
commerceservices.commercecartcalculationmethodhook.enabled=true
commerceservices.commerceplaceordermethodhook.enabled=true
commerceservices.commerceupdatecartentryhook.enabled=true
```

Сопутствующая информация

[Ускорители](#)

Конвертеры и пополнители

Объекты данных создаются из моделей или других объектов уровня сервиса с использованием преобразователей и пополнителей. Преобразователи создают новые экземпляры объектов данных и вызывают пополнители для заполнения этих объектов данных.

Обзор

Расширение commercefacades предоставляет набор фасадов, которые составляют единый многоканальный API витрины, который может использоваться несколькими фронтендами. Фасад отвечает за интеграцию существующих бизнес-сервисов из всего спектра расширений hybris и предоставление ответа объекта данных (POJO), настроенного для соответствия требованиям витрины.

Подход

Общий подход должен быть следующим:

- Не следует писать конкретные конвертеры, все конвертеры должны быть настроены только на Spring и должны использовать базовый класс AbstractConverter.
- Ни один Populator не должен вызываться напрямую в коде, Converters должны быть внедрены и использованы в Spring.
- Вся логика преобразования должна существовать в Populators, и они должны быть хорошо инкапсулированы и независимы.

Конвертеры

Конвертеры создают новые экземпляры объектов Data и вызывают Populators для их заполнения. Объект Data всегда создается из spring bean-компонентта с областью действия прототипа, который определен в beans.xml для расширения.

Популяторы

Заполнители разбивают процесс преобразования заполнения объекта данных на конвейер задач или шагов заполнения. Каждый заполнитель выполняет одно или несколько связанных обновлений прототипа объекта данных. Каждый шаг заполнения может вызывать службы или копировать данные из исходного бизнес-объекта в прототип объекта данных фасада. Фасады всегда используют преобразователь для создания нового экземпляра прототипа объекта данных, а затем вызывают заполнители или другие преобразователи для выполнения задачи по созданию объекта данных.

Настраиваемые пополнители и параметры данных

Congurable Populators расширяют концепцию Populators (не интерфейса Java), позволяя предоставлять коллекцию опций данных типа Enum. Затем Congurable Populator вызывает только те Populators, которые добавляют данные для указанных опций данных. Эта функциональность может показаться сложной, но она жизненно важна с точки зрения производительности и пропускной способности. Суть здесь в том, чтобы преобразовывать только то, что требуется, когда это требуется, путем точного указания того, какие populators следует использовать программно.

Преобразование всего объекта может быть затратным с точки зрения производительности. Например, рассмотрите возможность оценки акций для продукта без необходимости или присоединения данных классификации, когда это не требуется. Также, например, описание продукта может быть довольно большим, и поэтому возврат его в вызове веб-службы обычно не является необходимым. Конфигурируемые популяторы также позволяют вводить дополнительную логику популяций в конвейер популяций без необходимости замены или переопределения существующего кода.

-Примечание

Когда НастраиваемыеPopulatorsМеханизм был перемещен из коммерцияфасадырасширение в platformservicesрасширение, чтобы сделать его доступным для всех других расширений, это изменение включало следующие изменения:

- Пакет настраиваемых популяций был изменен cde.hybris.platform.commercefacades.converterkde.hybris.platform.converters.
- Код также был рефакторингован и упрощен.
- Занятия bde.hybris.platform.commercefacades.converterпакет все еще существует, но в настоящее время устарел.

Модифицируемые Конфигурируемые Популяторы

Функция modifiable configurable populators позволяет вам изменять CongurablePopulators, которые реализуют интерфейс ModifiableCongurablePopulator. Это включает добавление populators и удаление populators из уже настроенных компонентов CongurablePopulator.

Модификации настраиваются как Spring-бины, расширяющие абстрактный родительский configurablePopulatorModification-бин. Они регистрируют себя в целевом популяторе во время инициализации контекста.

Текоммерцияфасады-весна.xml содержит определение абстрактного родительского компонента configurablePopulatorModification, который определяет метод init, выполняющий регистрацию с целевым популятором, следующим образом:

```
<!-- Абстрактный компонент, используемый в качестве родительского для компонентов, которые изменяют ModifiableConfigurablePopulator. -->
<bean id="configurablePopulatorModification" class="de.hybris.platform.converters.config.ConfigurablePopulatorModification" abstract="
```

Примеры использования модифицируемых конфигурируемых популяций

Ниже приведен пример добавления пополнителя:

```
<bean parent="configurablePopulatorModification">
    <имя_свойства="target" ref="defaultProductConfiguredPopulator" />
    <имя_свойства="keyType" значение="de.hybris.platform.commercefacades.product.ProductOption" /> <имя_свойства="key"
        значение="VOLUME_PRICES" />
    <имя_свойства="add" ref="productVolumePricesPopulator" />
</боб>
```

Ниже приведен пример удаления популятора:

```
<bean parent="configurablePopulatorModification">
    <имя_свойства="target" ref="defaultProductConfiguredPopulator" />
```

```

<имя_свойства="keyType" значение="de.hybris.platform.commercefacades.product.ProductOption" /> <имя_свойства="key"
значение="VOLUME_PRICES" />
<имя_свойства="remove" ref="productVolumePricesPopulator" />
</боб>

```

Тип CongurablePopulatorModication способен разрешать следующие типы ключей:

- java.яз.Строка
- java.яз.Класс
- типы перечислений (такие какde.hybris.platform.commercefacades.product.ProductOptionтипа перечисления, показанный в двух примерах выше)

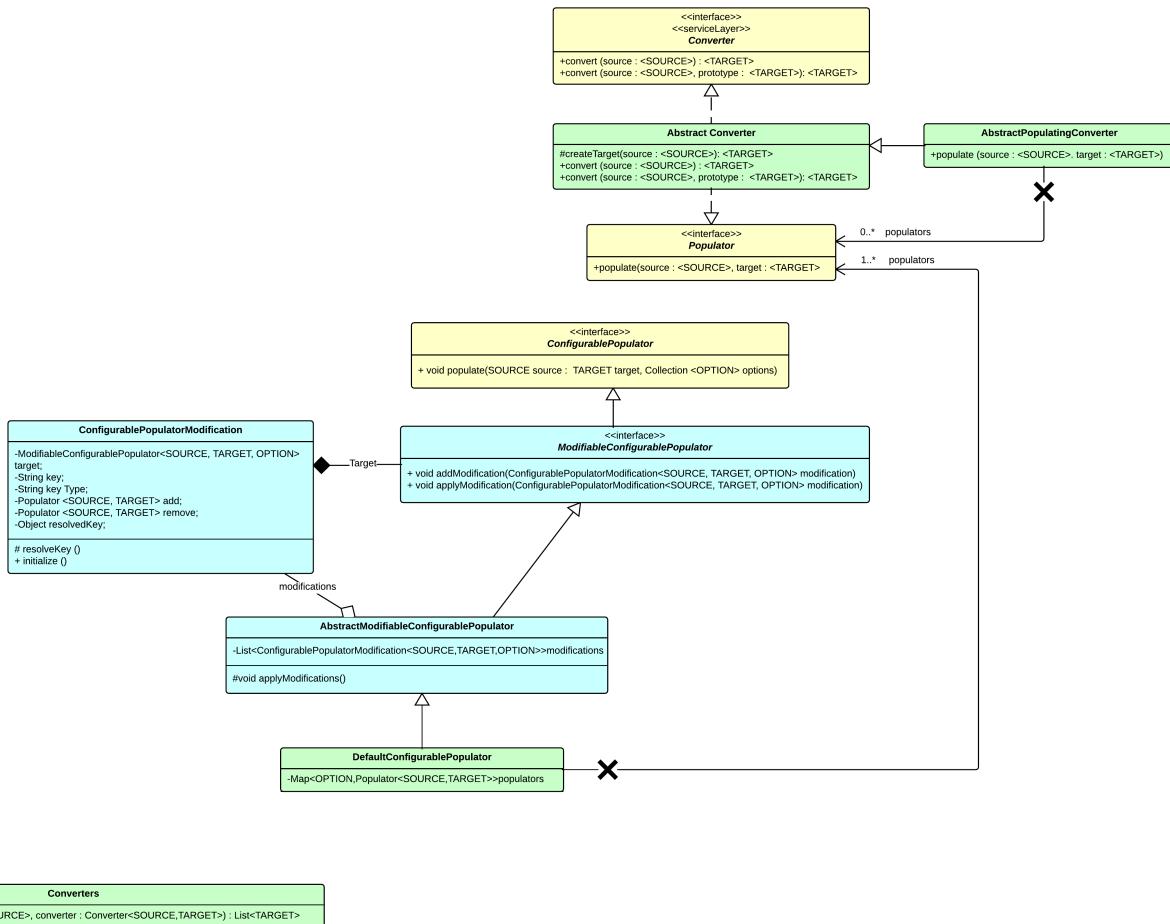
Во время выполнения, когда метод populate компонента ModifiableCongurablePopulator вызывается в первый раз, он сначала применяет все модификации, зарегистрированные для его предков (то есть родительский компонент, родительский компонент родителя и т. д.), прежде чем он применит все модификации, зарегистрированные для себя. Реализация гарантирует, что модификации применяются только один раз для каждого экземпляра компонента.

Базовые классы

Ниже вы можете найти описание базовых классов Populators и Converters:

- Поскольку преобразователи могут заполнять объекты данных и, следовательно, также могут считаться пополнителями, AbstractConverter — это удобный базовый класс, который реализует интерфейсы как Converter, так и Populator, и обеспечивает базовую реализацию интерфейса Converter, делегируя полномочия шаблонному методу createTarget, который позволяет конкретному подклассу выбирать соответствующую реализацию целевого объекта данных.
- AbstractPopulatingConverter просто расширяет AbstractConverter и дополнительно предоставляет реализацию populate из интерфейса Populator. Рекомендуется, чтобы большинство Converters расширяли этот базовый класс и просто переопределяли метод шаблона createTarget и внедряли необходимый конвейер Populators для заполнения объективного графа цели. Такой подход приводит к наиболее гибкому и адаптируемому коду, где большую часть поведения можно адаптировать извне посредством конфигурации bean-компоненты spring.
- DefaultCongurablePopulator — это реализация конвейера Populator по умолчанию, где каждый шаг заполнения оценивается по набору значений Enum, переданному вызывающей стороной. Каждый Populator в конвейере будет запущен только в том случае, если он сопоставлен с Enum, содержащимся в наборе, переданном вызывающей стороной. Эта реализация позволяет изменять список populator в конвейере.

На следующей диаграмме классов представлены настраиваемые и изменяемые настраиваемые популяторы.



Сопутствующая информация

[Ускорители преобразователей и](#)

[популяций Dene](#)

[Использование фасадов и DTO — передовой опыт](#)

Решатели ценностей

Решатели значений являются более эффективной заменой текущих поставщиков значений.

В этом документе содержится информация о преобразователях значений.

Ниже вы найдете список преобразователей значений вместе с описанием и поддерживаемыми параметрами.

АтрибутыПродуктЗначениеРешатель

Решатель для атрибутов продукта. Он учитывает атрибуты варианта продукта (но не атрибуты общего варианта). Если значение для варианта не найдено, он пытается получить его из базового продукта. По умолчанию, если атрибут параметра не указан, он пытается получить атрибут с тем же именем, что и настроенный для индексированного свойства.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификатора). Если эти условия не выполняются, возникает исключение типа FieldValueProviderException. Выдается исключение FieldValueProviderException.
атрибут		Если указано, это имя атрибута.
расколоть	ЛОЖЬ	Если true, разбивает любое разрешенное значение на совпадения с регулярным выражением (только если значение имеет тип String).
splitRegex	\c+	Если split имеет значение true, то это регулярное выражение, которое следует использовать.
пропуститьВарианты		Если значение true, варианты продукта игнорируются и используются значения из базового продукта.
формат		Идентификатор форматирующего компонента, который будет использоваться для форматирования объекта значения атрибута перед применением разделения.

ПродуктКлассификацияАтрибутыЗначениеРешатель

Решатель атрибутов классификации продуктов.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификатора). Если эти условия не выполняются, возникает исключение типа FieldValueProviderException. Выдается исключение FieldValueProviderException.
атрибут		Если указано, это имя атрибута.
расколоть	ЛОЖЬ	Если true, разбивает любое разрешенное значение на совпадения с регулярным выражением (только если значение имеет тип String).
формат		Идентификатор форматирующего компонента, который будет использоваться для форматирования объекта значения атрибута перед применением разделения.

ProductImagesValueResolver

Резолвер для URL-адресов изображений продуктов. Он выбирает первое изображение галереи, которое поддерживает запрошенный формат мультимедиа. По умолчанию, если параметр mediaFormat не указан, он пытается получить формат мультимедиа из индексированного имени свойства. По умолчанию он находит символ "-" и обрабатывает следующую последовательность символов как формат мультимедиа. Если символ "-" отсутствует, выдается исключение типа FieldValueProviderException. Метод анализа можно переопределить.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификатора). Если эти условия не выполняются, возникает исключение типа FieldValueProviderException. Выдается исключение FieldValueProviderException.
медиаформат		Если указано, это квалифицированный параметр формата носителя.

ПродуктКлючевые словаЗначениеБезопасность

Резолвер для ключевых слов продуктов.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификатора). Если эти условия не выполняются, возникает исключение типа <code>FieldValueProviderException</code> . Выдается исключение <code>FieldValueProviderException</code> .
расколов	истинный	Если false, собранные ключевые слова следует добавить в индекс как единое значение.
разделитель	" "	Разделитель, который следует использовать при объединении ключевых слов в одно значение. Используется только если split имеет значение false.

ПродуктЦеныСтоимостьResolver

Решатель цен на продукцию.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификиатора). Если эти условия не выполняются, выдается исключение <code>FieldValueProviderException</code> .

ProductUrlsValueResolver

Резолвер для URL-адресов продуктов.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификиатора). Если эти условия не выполняются, выдается исключение <code>FieldValueProviderException</code> .

ПродуктПродвижениеАтрибутыЗначениеResolver

Решатель, который получает значения продвижения продукта из атрибутов в модели продвижения. По умолчанию, если атрибут параметра не указан, он пытается получить атрибут с тем же именем, что и настроенный для индексированного свойства.

Параметр	Значение по умолчанию	Описание
необязательный	истинный	Если false, указывает, что разрешенные значения не должны быть нулевыми и не пустой строкой (для каждого квалификиатора). Если эти условия не выполняются, выдается исключение <code>FieldValueProviderException</code> .
атрибут		Если указано, это имя атрибута.
оценитьВыражение	ЛОЖЬ	Если true, то предполагается, что имя атрибута является языком выражений Spring, который необходимо оценить.

Сопутствующая информация

[API поставщика ценности](#)

[Модуль поиска и навигации](#)

Заполнение списка клиентов в магазине данными устройств IoT

Агент Assisted Service, работающий физически в магазине, может искать клиентов, которые в данный момент находятся в этом конкретном магазине. В настоящее время информация о том, какие клиенты присутствуют в данном магазине, моделируется и импортируется через файл ImpEx, но эти данные могут поступать из любого источника, включая устройство IoT.

Контекст

-Примечание

Ниже приведены шаги, иллюстрирующие пример интеграции с физическим устройством обнаружения. Процедура предназначена для предоставления читателю необходимого понимания технических деталей этого бизнес-кейса и не должна использоваться в качестве передовой практики. Фактическая интеграция с устройством IoT не поддерживается.

Чтобы заполнить список клиентов в магазине данными, поступающими с внешнего устройства, выполните следующие действия:

Процедура

. Отключите HTTPS и используйте HTTP для OCC v2, открыв `hybris/bin/modules/commerce-services/ycommercewebservices/web/webroot/WEB-INF/config/v2/security-v2-spring.xml` для редактирования.

. Вбезопасность-`v2-spring.xml`е, найдите `<перехват-url шаблон="/**" требует-канал="https"/>` код.

а. Изменить `https` параметр для `http`.

. `Hybris/bin/modules/commerce-services/ycommercewebservices/web/webroot/WEB-INF/config/v2` каталог, откройте `jaxbconverters-весна.xml` для редактирования.

а. Заменить `<util:list id="messageConvertersV2">`bean со следующим кодом:

```
<util:list id="messageConvertersV2">
    <реф bean="customJsonHttpMessageConverter"/>
    <реф bean="customXmlHttpMessageConverter"/>
        <боб class="org.springframework.http.converter.FormHttpMessageConverter" />
        <боб class="org.springframework.http.converter.StringHttpMessageConverter">
            <constructor-arg index="0" name="defaultCharset" value="UTF-8"/>
        </боб>
    </util:list>
```

. Создать публичный ЛояльностьДаоинтерфейс в `/hybris/bin/extcommerce/commerceservices/src/de/hybris/platform/commerceservices/customer/dao/` каталог.

а. ВЛояльностьДаоинтерфейс, определить `findUserByLoyaltyCardId()` метод.

```
публичный интерфейс LoyaltyDao {
    UserModel findUserByLoyaltyCardId(String loyaltyCardId);
}
```

. Создать публичный DefaultLoyaltyDao класс в `hybris/bin/extcommerce/commerceservices/src/de/hybris/platform/commerceservices/customer/dao/impl` каталог.

а. BDefaultLoyaltyDaokласс, реализующий бизнес-логику.

```
открытый класс DefaultLoyaltyDao расширяет DefaultGenericDao<CustomerModel> реализует LoyaltyDao {
    public DefaultLoyaltyDao() {
        super(CustomerModel._TYPECODE);
    }
    @Переопределить
    public CustomerModel findUserByLoyaltyCardId(final String loyaltyCardId) {
        окончательный список <CustomerModel> resList = find(Collections.singletonMap(CustomerModel.LOYALTYCARDID, loyaltyCardId)); return resList.isEmpty() ?
        null : resList.get(0);
    }
}
```

. Создать публичный ЛояльностьСервисинтерфейс в `hybris/bin/extcommerce/commerceservices/src/de/hybris/platform/commerceservices/customer/` каталог.

а. ВЛояльностьСервисинтерфейс, определить `getUserForLoyaltyCardID()` метод.

```
публичный интерфейс LoyaltyService {
    UserModel getUserForLoyaltyCardID(String loyaltyCardId);
}
```

. Создать публичный DefaultLoyaltyService класс в `hybris/bin/extcommerce/commerceservices/src/de/hybris/platform/commerceservices/customer/impl` каталог.

а. BDefaultLoyaltyService класс, реализующий бизнес-логику.

```
открытый класс DefaultLoyaltyService реализует LoyaltyService {
    частный LoyaltyDao LoyaltyDao;
    @Override
    public UserModel getUserForLoyaltyCardID(final String loyaltyCardId) {
        возвращаться получитьLoyaltyDao().findUserByLoyaltyCardId(loyaltyCardId);
    }
    защищенный LoyaltyDao получитьLoyaltyDao()
    {
        возвращаться ЛояльностьДао;
    }
    @Необходимый
    public void setLoyaltyDao(final LoyaltyDao LoyaltyDao)
```

```
{
    это.LoyaltyDao      = ЛояльностьДао;
}
}
```

. Настройте реализации по умолчанию вашего сервиса и DAO в Spring, отредактировав `hybris/bin/modules/commerceservices/commerceservices/resources/commerceservices-spring.xml`.

```
<alias name="defaultLoyaltyDao" alias="loyaltyDao"/>
<bean id="defaultLoyaltyDao" class="de.hybris.platform.commerceservices.customer.dao.impl.DefaultLoyaltyDao"/>

<псевдоним имя="defaultLoyaltyService"           alias="loyaltyService" /> class="de.hybris.platform.commerceservices.customer.impl.DefaultLoyaltyService">
<боб  идентификатор="defaultLoyaltyService"
      <имя свойства="loyaltyDao" /><bean>   ref="loyaltyDao"/>
```

. Перейдите к `hybris/bin/ext-template/ycommercewebservices/web/src/de/hybris/platform/ycommercewebservices/v2/controller/` каталог и обновить `CustomerGroupsController` класс с кодом ниже:

- Правильно аннотируйте и объявите лояльность Сервис переменная:

```
@Resource(name = "loyaltyService") частный
LoyaltyService loyaltyService;
```

- Правильно аннотируйте и создайте методы, отвечающие за назначение и удаление клиентов из групп:

```
@RequestMapping(value = "/{groupId}/geolocationRegister", method = RequestMethod.POST, uses =
{ MediaType.APPLICATION_FORM_URLENCODED_VALUE })
@ResponseStatus(значение = HttpStatus.OK)
public void assignUserToCustomerGroupByScanner(@PathVariable final String groupId, @RequestBody final MultiValueMap members {

    Страна customerLoyaltyId = ((LinkedList) members.get("field_values")).get(0).toString(); окончательный шаблон pattern =
    Pattern.compile("/[^/]*");
    окончательный Matcher matcher = pattern.matcher(customerLoyaltyId); if
    (matcher.find())
    {
        customerLoyaltyId = matcher.group(1).toLowerCase();
    }
    окончательная модель пользователя userModel = служба лояльности.getUserForLoyaltyCardID(customerLoyaltyId);
    если (null != userModel)
    {
        checkMembers(Collections.singletonList(userModel.getUid()), "Вы не можете добавить пользователя в группу:" + sanitize(groupId) +
        customerGroupFacade.addUserToCustomerGroup(groupId, userModel.getUid()));
    }
    еще
    {
        throw new RequestParameterException("Не удается найти пользователя с картой лояльности[" + customerLoyaltyId + "]");
    }
}
/**
 * Удаляет пользователя из группы клиентов.
 */
@RequestMapping(value = "/{groupId}/geolocationDeregister/{loyaltyCardId:.*}", method = RequestMethod.DELETE) @ResponseStatus(value =
HttpStatus.OK)
public void removeUsersFromCustomerGroupByScanner(@PathVariable final String groupId,
                                                 @PathVariable(value = «loyaltyCardId») конечная строка лояльностиCardId)
{
    final UserModel userModel = trustyService.getUserForLoyaltyCardID(loyaltyCardId); checkMembers(Collections.singleton(userModel.getUid()), "Вы не можете удалить
    пользователя из группы:" + sanitize(groupId) + "." customerGroupFacade.removeUserFromCustomerGroup(groupId, userModel.getUid()));

}
```

-Примечание

Более подробную информацию о структуре списков клиентов см. в разделе «Структура списков клиентов» [Расширение commerceservices](#) документ.

. Перейдите к `hybris/bin/платформа` каталог, запустите `mu` рабочий очистить все команду и запустите сервер `hybris`.

. Настройте устройство согласно инструкциям производителя. Вы можете использовать все виды устройств для предоставления данных, например, мобильные телефоны и iBeacons.

Устройство, на котором основана эта процедура, — Impinj Speedway Revolution. Обратите внимание, что каждое устройство отправляет данные определенным образом. Ниже вы можете найти пример вызовов REST, которые устройство делает для `hybris`:

Функция	URL
Зарегистрируйте клиента в магазине	<p>-Примечание Метод HTTP отличается в зависимости от устройства. Impinj Speedway Revolution использует POST для этого вызова.</p> <p>http://localhost:9001/rest/v2/electronics/customergroups/{groupId}/geolocationRegister</p>
Отменить регистрацию клиента в магазине	Метод HTTP отличается в зависимости от устройства. Impinj Speedway Revolution использует DELETE для этого вызова. http://localhost:9001/rest/v2/electronics/customergroups/{groupId}/geolocationDeregister/{loyaltyGroupId}

Коммерческие котировки

Узнайте о занятиях в коммерция услуг которые используются для функции коммерческих котировок.

Функция Commerce Quotes позволяет покупателям отправлять запрос на предложение своим торговым представителям. Предложение может обсуждаться обеими сторонами до тех пор, пока не будет достигнуто соглашение, после чего покупатель проверяет предложение и размещает заказ. Различные пункты, связанные с предложением, включенные в коммерция услуг перечислены ниже.

CommerceComment, OrgUnit и CommerceCartMetaData

Это элементы, связанные с CommerceComment, OrgUnit и CommerceCartMetaData:

- КоммерцияКомментарийСервис:Предлагает метод добавления нового комментария к модели элемента, поставляемой с ПараметрКомментарияТорговли.
- DefaultCommerceCommentService: Реализация по умолчанию КоммерцияКомментарийСервис расширяется DefaultCommentService.
- CommerceCommentUtils: Служебный класс, который предлагает статические методы для построения CommerceКомментарийПараметр объект с использованием CommerceCommentParameterBuilder.
- CommerceCommentParameterBuilder: Строитель класс для строительства ПараметрКомментарияТорговли.
- Параметр комментария к торговле: DTO/POJO используется для передачи данных из фасадного уровня в сервисный уровень.
- OrgUnitQuoteService: Предлагает интерфейсный метод для получения расценок для сотрудника организации.
- DefaultOrgUnitQuoteService: Реализация по умолчанию OrgUnitQuoteService.
- CommerceCartMetadataParameterUtils: Инстанцирует CommerceCartMetadataParameterBuilder.
- CommerceCartMetadataParameterBuilder: Строитель для ПараметрМетаданных ТорговойКарты.
- Параметр метаданных CommerceCart: DTO/POJO используются для передачи данных из фасадного уровня в сервисный уровень.

QuoteEvent Услуги

Это услуги, связанные с Цитата Событие:

- Аннотация Цитата Отправить Событие: Суперкласс для событий, связанных с цитатами, который предлагает общие переменные, такие как QuoteModel, UserModel и QuoteUserType. Предлагает методы получения и конструктор по умолчанию для извлечения и установки этих переменных.
- QuoteBuyerSubmitEvent: Событие, указывающее на то, что покупатель подал ценовое предложение.
- Цитата Отмена Событие: Событие, указывающее на отмену предложения.
- QuoteSalesRepSubmitEvent: Событие, указывающее на то, что торговый представитель отправил предложение.
- QuoteSellerApprovalSubmitEvent: Событие, указывающее на то, что утверждающий продавец одобрил или отклонил предложение, представленное торговым представителем.
- QuoteBuyerOrderPlacedEvent: Событие, указывающее на то, что покупатель разместил заказ по предложению поставщика.

Исключения из цитаты

Это классы, связанные с исключениями кавычек:

- IllegalQuoteStateException: Исключение типа RuntimeException который выбрасывается, когда действие не может быть выполнено для цитаты. Инкапсулирует базовую информацию о цитате и предлагает несколько открытых конструкторов для построения сообщения об исключении.
- IllegalQuoteSubmitException: Исключение типа RuntimeException что простираетсяIllegalQuoteStateException. Исключение выдается, когда действие отправки не может быть выполнено для цитаты. Инкапсулирует базовую информацию о цитате и предлагает несколько открытых конструкторов для построения сообщения об исключении.
- CommerceQuoteAssignmentException: Исключение типа RuntimeException что простирается Системное исключение. Если назначение цитаты не удалось, возникает исключение.
- CommerceQuoteExpirationTimeException: Исключение типа RuntimeException что простирается Системное исключение. Исключение возникает, если срок действия цитаты истек.

Цитата Стратегические классы

Это классы, связанные со стратегиями котировок:

- QuoteUserTypeIdentificationStrategy: Интерфейс, который предоставляет метод определения QuoteUserType из поставленных МодельПользователя.

- DefaultQuoteUserTypeIdentificationStrategy: Реализация по умолчанию QuoteUserTypeIdentificationStrategy.
- Стратегия Выбора Состояния Цитаты: Интерфейс, предлагающий методы для получения разрешенных состояний для действия, для получения разрешенных действий для состояния и для получения переходных состояний для действия.
- Стратегия Выбора Состояния Цитаты По Умолчанию: Использует Цитата UserTypeIdentificationStrategy для определения QuoteUserType (BUYER, SELLER, SELLER_APPROVER). Использует предварительно настроенные многоуровневые карты для определения действий или состояний для конкретного QuoteUserType.
- QuoteActionValidationStrategy: Стратегия, которая проверяет, может ли пользователь выполнить действие по цитате. Она предлагает два метода:
 - void validate(QuoteAction quoteAction, QuoteModel quoteModel, UserModel userModel): Проверяется, можно ли выполнить указанное действие с цитатой; если оно недействительно, выдается IllegalQuoteStateException.
 - boolean isValidAction(QuoteAction quoteAction, QuoteModel quoteModel, UserModel userModel): Возвращает логическое значение, указывающее, является ли действие допустимым.
- DefaultQuoteActionValidationStrategy: Реализация по умолчанию Цитата Действие Проверка Стратегия. Использует QuoteStateSelectionStrategy для получения разрешенных состояний для действия.
- QuoteUpdateExpirationTimeStrategy: Интерфейс, который предоставляет метод обновления срока действия предоставленного QuoteModel.
- Стратегия времени обновления цитаты по умолчанию: Использует Цитата UserTypeIdentificationStrategy чтобы определить QuoteUserType. На основе QuoteUserType предоставленное действие обновляет время истечения срока действия QuoteModel. Более подробную информацию о реализации можно найти в javadocs.
- QuoteUpdateStateStrategy: Интерфейс, который предоставляет метод обновления состояния предоставленного QuoteModel для конкретного действия и пользователя.
- Стратегия обновления состояния цитаты по умолчанию: Реализация по умолчанию QuoteUpdateStateStrategy. Использует QuoteStateSelectionStrategy для поиска состояния перехода для предоставленного действия и пользователя. Состояние перехода используется для обновления состояния предоставленного QuoteModel.
- QuoteUserIdentificationStrategy: Интерфейс, предоставляющий метод определения текущего пользователя сеанса.
- DefaultQuoteUserIdentificationStrategy: Реализация по умолчанию для QuoteUserIdentificationStrategy.
- Цитата Метаданные Проверка Стратегия: Интерфейс, предоставляющий метод проверки полей метаданных цитаты, таких как имя, описание и expirationTime.
- Стратегия проверки метаданных по умолчанию: Реализация по умолчанию для Стратегия проверки метаданных цитаты. Если имя пустое, то выдает Исключение IllegalArgumentException. Использует Цитата UserTypeIdentificationStrategy чтобы определить, если QuoteUserType является SELLER_APPROVER и если предложение одобряется. Если срок действия не установлен, тоIllegalStateExceptionброшен.
- Стратегия проверки назначения цитаты: Интерфейс, предоставляющий методы проверки назначения и отмены назначения котировок.
- Стратегия проверки назначения цитаты по умолчанию: Реализация по умолчанию Стратегия Проверки Назначения Цитаты. Методы реализации бросают CommerceQuoteAssignmentException для недопустимых операций назначения/отмены назначения.
- QuoteCartValidationStrategy: Интерфейс, который предоставляет метод проверки данных двух AbstractOrderModels на основе определенных критериев, определенных в реализации по умолчанию.
- Стратегия проверки корзины цитаты по умолчанию: Реализация по умолчанию QuoteCartValidationStrategy. Это в первую очередь используется в процессе оформления заказа на каждом этапе оформления заказа, чтобы гарантировать, что предложение поставщика не будет изменено до тех пор, пока заказ не будет размещен.
- Цитата Истечение Времени Проверка Стратегия: Интерфейс, предоставляющий метод проверки того, истек ли срок действия предоставленной цитаты.
- Стратегия проверки времени истечения срока действия цитаты по умолчанию: Реализация по умолчанию QuoteExpirationTimeValidationStrategy. Это в основном используется, когда покупатель принимает предложение продавца, чтобы гарантировать, что покупатель не сможет проверить предложение с истекшим сроком действия. Он также использует Время Обслуживания платформы.
- QuoteSellerApproverAutoApprovalStrategy: Интерфейс, предоставляющий метод проверки того, требует ли предоставленная смета ручного одобрения со стороны SELLER_APPROVER.
- DefaultQuoteSellerApproverAutoApprovalStrategy: Реализация по умолчанию QuoteSellerApproverAutoApprovalStrategy. В основном это используется в бизнес-процессе торгового представителя, который запускается, когда торговый представитель отправляет ценовое предложение.

Классы корзины и заказа

Это классы, связанные со стратегиями и хуками корзины и заказа:

- CommerceCreateCartFromQuoteStrategy: Наследует DefaultCreateCartFromQuoteStrategy с платформы для добавления коммерческой цитаты, связанной с бизнес-логикой, используя постпроцесс крючок в коммерции услуг.
- DefaultCommerceQuoteCartCalculationMethodHook: Реализует CommerceCartCalculationMethodHook Интерфейс. Он добавляет скидки, связанные с расценками, к глобальным скидкам и выполняет вызов для расчета итоговых сумм в корзине, чтобы скидки, связанные с расценками, были включены в глобальные скидки.
- CommercePlaceQuoteOrderMethodHook: Реализует CommercePlaceOrderMethodHook Интерфейс. Основная цель этого хука — проверить, является ли заказ заказом на котировку, а затем опубликовать QuoteBuyerOrderPlacedEvent.

CommerceCartМетаданные

Это классы, связанные с CommerceCartМетаданные:

- Стратегия обновления метаданных CommerceCart: Интерфейс, который предоставляет метод обновления метаданных корзины, таких как имя и описание, с помощью CommerceCartМетаданные Параметр DTO.
- DefaultCommerceCartMetadataUpdateStrategy: Реализация по умолчанию CommerceCartMetadataUpdateStrategy. Он обновляет предоставленную корзину с предоставленными полями в Параметр Метаданных Торговой Карты. Он также обеспечивает до Обновления Метаданных afterUpdateMetadata крючки как точки расширения.
- CommerceCartMetadataUpdateMethodHook: Интерфейс, который обеспечивает до Обновления Метаданных afterUpdateMetadata крючки, которые можно использовать в качестве точек расширения.
- DefaultCommerceCartMetadataUpdateValidationHook: Реализует CommerceCartMetadataUpdateMethodHook Интерфейсный хук для проверки предоставленного имени и описания в Параметр Метаданных Торговой Карты.
- QuoteCommerceCartMetadataUpdateValidationHook: Реализует CommerceCartMetadataUpdateMethodHook Интерфейсный хук для проверки предоставленного CommerceCartМетаданные Параметр когда цитата сохранена. Она использует QuoteActionValidationStrategy, QuoteUserIdentificationStrategy, QuoteUserTypeIdentificationStrategy и Служба времени.

CommerceQuoteService

Это классы, связанные сCommerceQuoteService:

- CommerceQuoteUtils:Предоставляет вспомогательный метод для удаления существующих скидок с расценокAbstractOrderModel.Он используется вDefaultCommerceQuoteService для применения скидок на расценки и для загрузки расценок в корзину сеанса.
- QuoteExpirationTimeUtils:Предлагает несколько вспомогательных статических методов, которые работают сQuoteExpirationTime.
- CommerceQuoteDao:Интерфейс предоставляет несколько методов извлечения котировок на основе различных параметров, таких как клиент, baseStore и состояния котировок.
- DefaultCommerceQuoteDao:Реализация по умолчаниюCommerceQuoteDao, и в основном используется вDefaultCommerceQuoteService.
- ОбновлениеЦитатыИзКорзиныСтратегия:Интерфейс, который предоставляет метод обновления предложения с содержимым предоставленной корзины. Обновляемое предложение связано с предоставленной корзиной.
- DefaultUpdateQuoteFromCartStrategy:РасширяетGenericAbstractOrderCloningStrategy<QuoteModel, QuoteEntryModel, CartModel> и является реализацией по умолчаниюОбновлениеЦитатыИзКартыСтратегии.Он также обеспечивает постпроцессорюк как точка расширения.
- OrderQuoteDiscountValuesAccessor:Интерфейс, который предоставляет методы для получения и установки значений скидки на предоставленные предложенияAbstractOrderModel.
- DefaultOrderQuoteDiscountValuesAccessor:Реализация по умолчаниюOrderQuoteDiscountValuesAccessor.Этот аксессор в первую очередь введен для преобразования списка значений скидок в строковое представление и наоборот.получитьЦитатыСкидкиЗначенияиспользуется в DefaultCommerceQuoteCartCalculationMethodHook.afterCalculateметод применение скидок на котировкиCartModel.getQuoteDiscountValuesиспользуется в DefaultCommerceQuoteService.applyQuoteDiscountметод применения скидок на котировкиAbstractOrderModel.
- CommerceQuoteService:Интерфейс, предоставляющий методы для операций управления ценовыми предложениями, таких как поиск, отправка, утверждение, отклонение и редактирование.
- DefaultCommerceQuoteService:Реализация по умолчаниюCommerceQuoteServiceкоторый использует следующие классы служб, DAO и утилит:
 - CommerceQuoteДао
 - КорзинаСервис
 - МодельСервис
 - CommerceSaveCartService
 - SessionService
 - CommerceCartService
 - QuoteStateSelectionStrategy
 - ЦитатаДействиеПроверкаСтратегия
 - ЦитатаОбновлениеСостоянияСтратегия
 - ОбновлениеЦитатаИзКорзиныСтратегия
 - цитатаSnapshotStateTransitionMap;
 - QuoteService
 - РасчетСервис
 - ЦитатаUserTypeIdentificationStrategy
 - EventService
 - ЦитатаЗаданиеПроверкаСтратегия
 - QuoteSellerApproverAutoApprovalStrategy
 - QuoteCartПроверкаСтратегия
 - ЦитатаВремяСрокаДействияПроверкаСтратегия
 - QuoteОбновлениеИстечение срокаСтратегия
 - ЦитатаМетаданныеПроверкаСтратегия
 - ЗаказЦитатаСкидкаЗначенияДоступник
 - ПользовательСервис
 - CommerceQuoteUtils
- void assignQuoteToUser(QuoteModel цитата, UserModel получатель, UserModel назначатель)иvoid unassignQuote(цитата QuoteModel, присваиватель UserModel):Методы, используемые для назначения и отмены назначения пользователейQuoteModel.Для гибкости, UserModelиспользуется тип; передается присваивателю для проверки, разрешено ли ему выполнять операцию.

Расширение проверки цитаты

Интерфейс CommerceQuoteValidator позволяет вам предоставлять пользовательскую логику проверки, а также пользовательское сообщение об ошибке в случае сбоя проверки.

```
void validate(QuoteAction quoteAction, QuoteModel quoteModel, UserModel userModel);
```

The defaultQuoteActionValidationStrategy вызывает любой зарегистрированный валидатор во время проверки цитаты, а также при проверке, разрешен ли определенный переход цитаты. Как только валидатор укажет на проблему, выдавIllegalQuoteStateException, Валидация прерывается и считается неудавшейся. Больше валидаторов не будет быть вызвано. Проверка считается успешной только в том случае, если все валидаторы не выявили проблем. Чтобы указать, почему проверка не удалась, создайтеIllegalQuoteStateException с индивидуальным сообщением.

Чтобы реализовать собственную логику проверки, выполните следующие действия:

- . Создайте свой собственный класс валидатора, реализовав интерфейсde.hybris.platform.commerceservices.order.validator.CommerceQuoteValidator.

. Объявите свой пользовательский валидатор как компонент Spring.

```
<alias name="defaultCustomQuoteValidator" alias="customQuoteValidator" />
<bean id="defaultCustomQuoteValidator" class="com.custom.quote.validation.impl.MyCustomQuoteValidator"/>
```

. Зарегистрируйте его как валидатор коммерческих предложений, добавив его в список валидаторов через Spring.

```
<bean id="customQuoteValidatorListMergeDirective" зависит от="commerceQuoteValidatorList" родительский="listMergeDirective">
<имя_свойства="add" ref="customQuoteValidator" /> </bean>
```

Сопутствующая информация

[Технический обзор](#)

[Статусы цитат](#)

[Типы пользователей](#)

[Стратегии определения типа пользователя и выбора состояния](#)

[Коммерческие котировки товаров](#)

Функциональность группировки записей в корзине

Функция группировки записей позволяет группировать несколько записей корзины в один пакет и работать с ним как с одним объектом.

Платформа предоставляет типы и базовые API для обработки групп записей, для получения дополнительной информации см.[Группировка записей в корзине](#) документ. И коммерция услуги предоставляют коммерческие функции для добавления, удаления или обновления групп записей.

Для того, чтобы предоставить группу записей для хуков, CommerceCartParameter имеет атрибут entryGroupNumbers. Это набор уникальных чисел, представляющих группы записей:

```
<bean class="de.hybris.platform.commerceservices.service.data.CommerceCartParameter"
      template="resources/commerce-cart-parameter-template.vm">
  ...
  <property name="entryGroupNumbers" type="java.util.Set<Integer>"> <description>Номера
  групп записей</description>
</ свойство>
```

The entryGroupNumbers атрибут можно найти в модификации торговой корзины de.hybris.platform.commerceservices.order.CommerceCartModification и данные об изменениях корзины de.hybris.platform.commercefacades.order.data.CartModificationData

commercefacades/resources/commercefacades-beans.xml

```
<bean class="de.hybris.platform.commercefacades.order.data.CartModificationData"> ...
<имя свойства="entryGroupNumbers" тип="java.util.Set<Integer>" /> </bean>
```

commerceservices/resources/commerceservices-beans.xml

```
<bean class="de.hybris.platform.commerceservices.order.CommerceCartModification"> ...
<имя свойства="entryGroupNumbers" тип="java.util.Set<Integer>" /> </bean>
```

Весенняя конфигурация

Ade.hybris.platform.commerceservices.service.data.RemoveEntryGroupParameter добавляется в commerceservices/resources/commerceservices-beans.xml.

```
<bean class="de.hybris.platform.commerceservices.service.data.RemoveEntryGroupParameter">
  <имя свойства="cart" тип="de.hybris.platform.core.model.order.CartModel">
    <description>Модель тележки</description> </
    <property>
      <имя свойства="enableHooks" тип="логическое значение">
        <description>Должны ли выполняться методы-хуки</description> </property>
      <имя свойства="entryGroupNumber" тип="java.lang.Integer">
        <description>Номер группы записей, подлежащей удалению</description> </
        <property>
    </бюб>
```

DefaultCartFacade использует стандартную последовательность действий с сервисом, стратегией и хуком.

Для стратегии созданы два интерфейса: de.hybris.platform.commerceservices.order.CommerceRemoveEntryGroupStrategy и крючок - de.hybris.platform.commerceservices.order.hook.CommerceRemoveEntryGroupMethodHook для поддержки удаления группы записей.

```
упаковка de.hybris.platform.commerceservices.order;
импорт de.hybris.platform.commerceservices.service.data.RemoveEntryGroupParameter;
импорт javax.annotation.Nonnull;

/**
 * Стратегический интерфейс для добавления товаров в корзину.
 */
открытый интерфейс CommerceRemoveEntryGroupStrategy {

    /**
     * Удаляет из (существующей) {@link de.hybris.platform.core.model.order.CartModel} (существующую) {@link de.hybris.platform.core}
     * Если запись с указанной группой записей уже есть в корзине, она также будет удалена.
     *
     * @param параметр - объект параметра, содержащий все атрибуты, необходимые для удаления группы записей
     * <П>
     *      {@link RemoveEntryGroupParameter#cart} — корзина пользователя в сеансе
     *      {@link RemoveEntryGroupParameter#entryGroupNumbers} - Номер группы записей, которую нужно удалить из корзины. {@link
     *      RemoveEntryGroupParameter#enableHooks} - Включены ли хуки.
     *      </П>
     * @return данные об изменении корзины, включая statusCode и фактический номер группы записей, удаленных из корзины
     * @броски CommerceCartModificationException
     *      если соответствующая запись в корзине не была удалена
     */
    @Nonnull
    CommerceCartModification removeEntryGroup(@Nonnull final RemoveEntryGroupParameter параметр) выдает CommerceCartModificationException
}

пакет de.hybris.platform.commerceservices.order.hook;
```

```
импорт de.hybris.platform.commerceservices.order.CommerceCartModification;
импорт de.hybris.platform.commerceservices.order.CommerceCartModificationException;
импорт de.hybris.platform.commerceservices.service.data.RemoveEntryGroupParameter; javax.annotation.Nonnull;
импорт
публичный интерфейс CommerceRemoveEntryGroupMethodHook
{
    /**
     * Выполняется после удаления группы записей торговли
     *
     * @param параметр
     * @param результат
     */
    void afterRemoveEntryGroup(@Nonnull final RemoveEntryGroupParameter параметр, CommerceCartModification результат); /**/

    *
    * Выполняется до того, как коммерция удаляет запись group
    *
    * @param параметр
    */
    void beforeRemoveEntryGroup(@Nonnull final RemoveEntryGroupParameter параметр) выдает CommerceCartModificationException;
}
```

Реализация стратегии по умолчанию

```
упаковка de.hybris.platform.commerceservices.order.impl;
импорт de.hybris.platform.commerceservices.constants.CommerceServicesConstants;
импорт de.hybris.platform.commerceservices.order.CommerceCartModification;
импорт de.hybris.platform.commerceservices.order.CommerceCartModificationException;
импорт de.hybris.platform.commerceservices.order.CommerceCartModificationStatus;
импорт de.hybris.platform.commerceservices.order.CommerceEntryGroupService;
импорт de.hybris.platform.commerceservices.order.CommerceRemoveEntryGroupStrategy;
импорт de.hybris.platform.commerceservices.order.CommerceUpdateCartEntryStrategy;
импорт de.hybris.platform.commerceservices.order.hook.CommerceRemoveEntryGroupMethodHook;
импорт de.hybris.platform.commerceservices.service.data.CommerceCartParameter;
импорт de.hybris.platform.commerceservices.service.data.RemoveEntryGroupParameter;
импорт de.hybris.platform.core.model.order.AbstractOrderEntryModel;
импорт de.hybris.platform.core.model.order.CartModel;
импорт de.hybris.platform.core.order.EntryGroup;
импорт de.hybris.platform.servicelayer.config.ConfigurationService;
импорт de.hybris.platform.servicelayer.model.ModelService;
импорт org.springframework.beans.factory.annotation.Required; javax.annotation.Nonnull;
импорт
импорт java.util.List;
импорт java.util.stream.Collectors; статический
импорт      de.hybris.platform.servicelayer.util.ServicesUtil.validateParameterNotNull;
импорт      статический de.hybris.platform.servicelayer.util.ServicesUtil.validateParameterNotNullStandardMessage;
```

публичный класс DefaultCommerceRemoveEntryGroupStrategy реализует
 CommerceRemoveEntryGroupСтратегия

```
{
    частный Список<CommerceRemoveEntryGroupMethodHook> commerceRemoveEntryGroupHooks;
    частный ConfigurationService configurationService;
    частный CommerceUpdateCartEntryStrategy updateCartEntryStrategy;
    частный CommerceEntryGroupService entryGroupService;
    частный МодельСервиса модельСервиса;
}

/** Удаляет из корзины группу записей со всеми вложенными группами и их записями в корзине
 *
 * @param параметр
 *      удалить параметры группы записей
 * @return Информация об изменении корзины
 * @throws CommerceCartModificationException, если соответствующая запись корзины не была удалена
 *
 */
@Переопределить
@Nonnull
public CommerceCartModification removeEntryGroup(@Nonnull final RemoveEntryGroupParameter параметр) бросает CommerceCartModificati {
```

validateParameterNotNullStandardMessage("параметр", параметр);
 beforeRemoveEntryGroup(параметр);
 окончательная модификация CommerceCartModification = doRemoveEntryGroup(параметр);
 afterRemoveEntryGroup(параметр, модификация);
 возврат модификации;

```
}
```

```
/** Удалить из корзины.
 *
 * @param параметр
 *      удалить параметр группы записей
 * @return модификация торговой корзины
 * @throws CommerceCartModificationException
 *      исключение модификации торговой корзины
 */
защитенный CommerceCartModification doRemoveEntryGroup(конечный параметр RemoveEntryGroupParameter) бросает CommerceCartModificationEx {
```

final CartModel cartModel = параметр.getCart();
 окончательное целое число groupNumber = параметр.getentryGroupNumbers(); пытаться

```
{
    final EntryGroup parentGroup = entryGroupService.getParent(cartModel, groupNumber); final EntryGroup group =
    getEntryGroupService().getGroup(cartModel, groupNumber); final List<Integer> groupNumbers =
    getAllSubsequentGroupNumbers(group); removeEntriesByGroupNumber(параметр, groupNumbers);

    если (родительскаягруппа == null) {

        окончательный список<EntryGroup> rootGroups = cartModel.getEntryGroups();
        cartModel.setEntryGroups(excludeEntryGroup(rootGroups, groupNumber));
    }
    еще
    {
        parentGroup.setChildren(excludeEntryGroup(parentGroup.getChildren(), groupNumber));
    }
    getEntryGroupService().forceOrderSaving(cartModel);
    return createRemoveEntryGroupResp(параметр, CommerceCartModificationStatus.SUCCESSFULLY_REMOVED);
}
```

ловить (Исключение НедопустимогоАргумента Д)
 {
 возвращаться createRemoveEntryGroupResp(параметр, Статус_модификации_корзины_торговли.INVALID_ENTRY_GROUP_NUMBER);
 }
}

```
защитенный список<EntryGroup> excludeEntryGroup(конечный список<EntryGroup> источник, конечный целый номер группы) {

    return source.stream().filter(g -> !groupNumber.equals(g.getGroupNumber())).collect(Collectors.toList());
}
```

```
защитенный CommerceCartModification createRemoveEntryGroupResp(конечный параметр RemoveEntryGroupParameter, конечный статус строки) {

    final Integer entryGroupNumbers = параметр.getentryGroupNumbers(); final CommerceCartModification
    модификация = новый CommerceCartModification(); модификация.setStatusCode(status);

    модификация.setentryGroupNumbers(entryGroupNumbers); возврат
    модификации;
}
```

```
protected void removeEntriesByGroupNumber(конечный параметр RemoveEntryGroupParameter, конечный List<Integer> groupNumbers)
    выдает исключение CommerceCartModificationException
```

```
{
    окончательный CartModel cartModel = параметр.getCart(); если
    (cartModel.getEntries() == null)
    {
        возвращаться;
    }

    для (конечная запись AbstractOrderEntryModel: cartModel.getEntries()) {

        если (groupNumbers.contains(entry.getentryGroupNumbers())) {

            окончательный Параметр CommerceCart updateParameter = новый Параметр CommerceCart();
            updateParameter.setCart(параметр.getCart());
            updateParameter.setEnableHooks(parameter.isEnableHooks());
            updateParameter.setQuantity(0L);
            updateParameter.setEntryNumber(entry.getEntryNumber());
            getUpdateCartEntryStrategy().updateQuantityForCartEntry(updateParameter);
        }
    }
}
```

```
защитенный список<Integer> getAllSubsequentGroupNumbers(конечная группа EntryGroup)
```

```

{
    возвраща́ясь getEntryGroupService().getNestedGroups(группа).stream()
        .карта(EntryGroup::getGroupNumber)
        .собрать(Collectors.toList());
}
защищенный void validateRemoveEntryGroupParameter(конечные параметры RemoveEntryGroupParameter) {

    окончательная CartModel cartModel = параметры.getCart();
    final Integer entryGroupNumbers = параметры.getentryGroupNumbers(); validateParameterNotNull(cartModel,
    "Модель корзины не может быть нулевой");
    validateParameterNotNullStandardMessage("entryGroupNumbers", entryGroupNumbers);
}
protected void beforeRemoveEntryGroup(final RemoveEntryGroupParameter параметры) выдает CommerceCartModificationException {

    если (getCommerceRemoveEntryGroupHooks() != null
        && (parameters.isEnableHooks() && getConfigurationService().getConfiguration().getBoolean(CommerceServicesConstant
    {
        для (final CommerceRemoveEntryGroupMethodHook commerceAddToCartMethodHook : getCommerceRemoveEntryGroupHooks()) {

            commerceAddToCartMethodHook.beforeRemoveEntryGroup(параметры);
        }
    }
})
protected void afterRemoveEntryGroup(конечные параметры RemoveEntryGroupParameter, конечный результат CommerceCartModification)
    выдает исключение CommerceCartModificationException
{
    если (getCommerceRemoveEntryGroupHooks() != null
        && (parameters.isEnableHooks() && getConfigurationService().getConfiguration().getBoolean(CommerceServicesConstant
    {
        для (final CommerceRemoveEntryGroupMethodHook commerceAddToCartMethodHook : getCommerceRemoveEntryGroupHooks()) {

            commerceAddToCartMethodHook.afterRemoveEntryGroup(параметры, результат);
        }
    }
})
защищенный Список<CommerceRemoveEntryGroupMethodHook> getCommerceRemoveEntryGroupHooks()

{
    возвраща́ясь commerceRemoveEntryGroupHooks;
}

@Необходимый
public void setCommerceRemoveEntryGroupHooks(final List<CommerceRemoveEntryGroupMethodHook> commerceRemoveEntryGroupHooks) {

    this.commerceRemoveEntryGroupHooks = commerceRemoveEntryGroupHooks;
}
защищенный ConfigurationService getConfigurationService()
{
    возвраща́ясь конфигурацияСервис;
}

@Необходимый
public void setConfigurationService(final ConfigurationService configurationService) {

    эта.configurationService = конфигурацияСервиса;
}
защищенный CommerceUpdateCartEntryStrategy getUpdateCartEntryStrategy()
{
    возвраща́ясь обновитьCartEntryStrategy;
}

@Необходимый
public void setUpdateCartEntryStrategy(final CommerceUpdateCartEntryStrategy updateCartEntryStrategy) {

    this.updateCartEntryStrategy = обновитьCartEntryStrategy;
}
защищенный CommerceEntryGroupService getEntryGroupService()
{
    возвраща́ясь entryGroupService;
}

@Необходимый
public void setEntryGroupService(final CommerceEntryGroupService entryGroupService) {

    this.entryGroupService = entryGroupService;
}
защищенный Служба модели getModelService()
{
    возвраща́ясь модельСервис;
}

@Необходимый
public void setModelService(final ModelService modelService) {

    этот.modelService = modelService;
}
}

```

Реализации хука по умолчанию нет. Но в commerceServices/resources/commerceServices-spring.xml есть следующая весенняя фасоль:

```

<util:list id="commerceRemoveEntryGroupMethodHooks" value-type="de.hybris.platform.commerceServices.order.hook.CommerceRemoveEntryGroupMethodHook">
    <util:entry key="removeEntryGroup">
        <bean class="de.hybris.platform.commerceServices.order.hook.impl.CommerceRemoveEntryGroupMethodHook"/>
    </util:entry>
</util:list>

```

Следующие статусы в commerceServices/src/de/hybris/platform/commerceServices/order/CommerceCartModificationStatus.java доступны:

- INVALID_ENTRY_GROUP_NUMBER - если в корзине нет группы с таким номером.
- УСПЕШНО_УДАЛЕНО - если группа записей была успешно удалена.

They удалить Группу Входов можно найти в commerceServices/src/de/hybris/platform/commerceServices/order/CommerceCartService.java. По умолчанию метод вызывает CommerceRemoveEntryGroupСтратегия для процесса удаления.

Фасадный слой

Данные группы входов DTO используется для отображения групп записей на витрине. Он определен в commercefacades/resources/commercefacades-beans.xml:

```
<bean class="de.hybris.platform.commercefacades.order.EntryGroupData">
    <импорт тип="com.fasterxml.jackson.annotation.JsonManagedReference"/>
    <импорт тип="com.fasterxml.jackson.annotation.JsonBackReference"/>
    <импорт тип="com.fasterxml.jackson.annotation.JsonIdentityReference"/>
    <импорт тип="com.fasterxml.jackson.annotation.JsonIdentityInfo"/>
    <импорт тип="com.fasterxml.jackson.annotation.ObjectIdGenerators"/>
    <свойство имя="groupNumber" тип="java.lang.Integer" имя="priority" />
    <свойство тип="java.lang.Integer" имя="label" тип="java.lang.String" />
    <свойство имя="groupType" />
    <свойство type="de.hybris.platform.core.enums.GroupType" />
    <свойство имя="дети" тип="java.util.List<de.hybris.platform.commercefacades.order.EntryGroupData>" />
    <свойство ><annotations>@JsonManagedReference</annotations> </property>

    <свойство имя="externalReferenceId" тип="java.lang.String" /> имя="ошибочный"
    <свойство тип="java.lang.Boolean" />
    <свойство name="orderEntries" тип="java.util.List<de.hybris.platform.commercefacades.order.data.OrderEntryData>" name="rootGroup" />
    <свойство type="de.hybris.platform.commercefacades.order.EntryGroupData" />
    <свойство ><annotations>@JsonIdentityInfo(генератор=ObjectIdGenerators.PropertyGenerator.class, </property> свойство="номергруппы")@</annotations> </property>

    <имя свойства="parent" тип="de.hybris.platform.commercefacades.order.EntryGroupData" />
    <свойство ><annotations>@JsonBackReference</annotations> </property>

</боб>
```

AbstractOrderData имеет дополнительный атрибут rootGroups, который заполняется из AbstractOrderModel. группы записей:

```
<bean class="de.hybris.platform.commercefacades.order.data.AbstractOrderData">
    ...
    <имя_свойства="rootGroups" тип="java.util.List<de.hybris.platform.commercefacades.order.EntryGroupData>" />
</боб>
```

Аннотации Джексона используются для предотвращения исключений Stackoverflow во время сериализации из-за двунаправленных ссылок родитель-потомок.

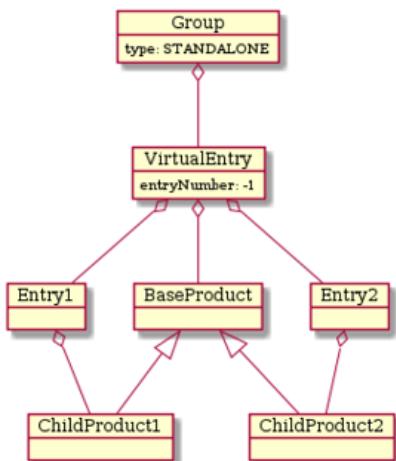
Отдельные записи

Для каждой записи в корзине, которая не принадлежит определенной группе записей в корзине, создается виртуальная группа записей с АВТОНОМНЫЙ тип группы. Это поведение реализовано в DefaultVirtualEntryGroupStrategy. Если есть необходимость в различной обработке отдельных записей, проекты могут реализовать свои собственные Стратегии VirtualEntryGroup.

Многомерные записи

При использовании обычных записей каждая запись помещается в отдельную группу записей во время заполнения из модели в DTO и может быть изменена путем повторного редактирования.

Стратегия VirtualEntryGroup. Многомерные записи (описанные GroupOrderEntryPopulator) есть группа записей только для родительской записи. Родительская запись включает последние записи как дочерние.



Сопутствующая информация

[Расширение commerceServices](#)

[Группировка записей в корзине](#)

[Группировка записей в корзине](#)

Погашение ваучера, проверка и обнаружение атак методом подбора

Ваучеры выкупаются при оформлении заказа. Ранее реализация этой функции была возложена на партнера.

Правильное погашение ваучеров позволяет системе регистрировать, когда ваучеры используются, а затем применять ограничения использования. Такие ограничения включают общий лимит доступных погашений ваучеров и лимит погашений на одного клиента.

Множественные проверки гарантируют, что заказ не будет размещен с недействительным ваучером. Эти проверки включают следующее:

- После добавления ваучера в корзину он подтверждается.
- При запуске процесса оформления заказа ваучер проверяется.
- Перед размещением заказа в конце процесса оформления заказа ваучер проверяется.

Если ваучер становится недействительным одновременно с размещением заказа, в процессе размещения заказа возникает ошибка, и ваучер не активируется.

Добавление ваучера в корзину также включает в себя простое обнаружение атаки методом подбора, целью которой является предотвращение атаки методом подбора с целью угадывания кодов ваучеров.

Настройка логики погашения ваучера

Логика, которая выкупает ваучеры, реализована с помощью хука. Хук по умолчанию можно заменить настраиваемым хуком. Следующие шаги описывают, как использовать настраиваемый хук выкупа ваучеров вместо предоставленной реализации по умолчанию.

. Создайте класс-хук, реализующий интерфейс de.hybris.platform.commerceservices.order.hook.CommercePlaceOrderMethodHook.

. Зарегистрируйте свой класс как spring bean. Ниже приведен пример:

```
<bean id="myCustomVoucherPlaceOrderMethodHook" class="com.example.MyCustomVoucherPlaceOrderMethodHook">
```

. Переделать псевдоним vaucherRedemptionPlaceOrderMethodHook так что он ссылается на созданный вами пользовательский хук. Ниже приведен пример:

```
<alias name="myCustomVoucherPlaceOrderMethodHook" alias="voucherRedemptionPlaceOrderMethodHook" />
```

-Примечание

Логика погашения из реализации по умолчанию находится в `doSendVoucherRedemptionPlaceOrderMethod`.

Настройка логики проверки корзины ваучеров

Логика проверки ваучера корзины, которая запускается в начале и в конце процесса оформления заказа, реализована с помощью хука проверки корзины. Хук по умолчанию можно заменить настраиваемым хуком. Следующие шаги описывают, как использовать пользовательский хук проверки корзины ваучера вместо предоставленной реализации по умолчанию.

. Создайте класс-хук, реализующий интерфейс de.hybris.platform.commerceservices.strategies.hooks.CartValidationHook.

. Зарегистрируйте свой класс как spring bean. Ниже приведен пример:

```
<bean id="myCustomVoucherCartValidationMethodHook" class="com.example.MyCustomVoucherCartValidationMethodHook" />
```

. Переделать псевдоним vaucherRedemptionPlaceOrderMethodHook так что он ссылается на созданный вами пользовательский хук. Ниже приведен пример:

```
<alias name="myCustomVoucherCartValidationMethodHook" alias="voucherRedeemableCartValidationMethodHook" />
```

-Примечание

Логика проверки из реализации по умолчанию находится в `afterValidateCartMethod`.

Настройка обнаружения атаки методом подбора ваучера

Простой механизм обнаружения методом подбора срабатывает при попытке добавить ваучер в корзину. Целью механизма обнаружения является предотвращение атак методом подбора, пытающихся угадать коды ваучеров.

По умолчанию обработчик атак методом подбора отслеживает попытки добавить код ваучера на основе IP-адреса. Если было сделано настраиваемое количество попыток в настраиваемом окне времени, любые дальнейшие попытки этого IP-адреса будут блокироваться на настраиваемое количество времени. Кроме того, пользователю выводится определенное сообщение об ошибке.

Другая настройка по умолчанию заключается в том, что любые дальнейшие попытки ввода IP-адреса будут отклоняться в течение одного часа, если в течение 5 минут будет сделано более 5 попыток.

Следующие параметры в `commerceservices.properties` настроим вышеупомянутое поведение:

- `commerceservices.bruteForceAttackHandler.maxAttempts`: Определяет, сколько попыток ввести код ваучера можно сделать, прежде чем все дальнейшие попытки будут отклонены.
- `commerceservices.bruteForceAttackHandler.timeFrame`: Определяет временное окно (в секундах), в течение которого лимит попыток не может быть превышен. По умолчанию это значение равно 300, т.е. 5 минут.
- `commerceservices.bruteForceAttackHandler.waitTime`: Определяет, как долго (в секундах) IP-адресу запрещено вводить коды ваучеров после блокировки. Это значение по умолчанию установлено на 3600. 3600 секунд равны одному часу.

-Примечание

Вы можете установить как временной интервал, так и свойство времени ожидания на 0, а затем перезапустить сервер, чтобы эффективно отключить обнаружение атак методом подбора.

Настройка коммерческих услуг

Руководство по настройке коммерческих услуг.

Обзор

Commerce Services настроены на раскрытие части Commerce Facades. Он поставляется как `commerceseveb-servicesextension`, который является шаблонным расширением, позволяющим вам создавать собственное расширение в пространстве доменных имен. Вы можете настроить его, добавив новые ресурсы или адаптировав сериализацию и десериализацию. Чтобы добиться этого, вам необходимо внести изменения в основном в Spring MVC Controllers, которые предоставляют Commerce Services, и в DTO, предоставляемые Commerce Facades.

Создание индивидуального расширения с использованием extgen

Запустите команду `extgen` для создания `commerceseveb-services`-расширения на основе пространства имен вашего проекта.

C:\workspace\bin\platform>ant extgen

екстген:

[вход] [input] Выберите имя вашего расширения. Оно должно начинаться с буквы, за которой следуют буквы и/или цифры. [input] Нажмите [Enter], чтобы использовать значение по умолчанию [training]

Предположим, что ваше новое имя расширения — `mysampleextension`.

-Примечание

Не используйте стандартное имя шаблона в новом имени расширения, которое вы создаете. Это приведет к ошибке сборки. Например, назвав свое новое расширение `mycommerceseveb-services` при его расширении от `commerceseveb-services`, выдаст ошибку, поскольку имя расширения содержит стандартное имя шаблона.

`mysampleextension`

[вход]

[input] Выберите имя пакета вашего расширения. Оно должно соответствовать соглашению об именах пакетов Java. [input] Нажмите [Enter], чтобы использовать значение по умолчанию [org.training]

com.mycompany.mycommerceseveb-services

[вход]

[вход] Выберите шаблон для генерации.

[input] Нажмите [Enter], чтобы использовать значение по умолчанию ([empty], `ycockpit`, `ybackoffice`, `yaddon`, `yacceleratorfulfilmentprocess`, `ycommercce` `ycommerceseveb-services`)

[echo] Используется исходный шаблон расширения: C:\workspace\bin\modules\commerce-services\commerceseveb-services [delete] Удаление каталога C:\workspace\temp\hybris\extgen_final

[удалить] Удаление каталога C:\workspace\temp\hybris\extgen

[mkdir] Созданный каталог: C:\workspace\temp\hybris\extgen

[echo] Копирование файлов шаблонов из C:\workspace\bin\modules\commerce-services\commerceseveb-services в C:\workspace\temp\hybris\ex [copy] Копирование 298 файлов в C:\workspace\temp\hybris\extgen

[copy] Копирование 298 файлов в C:\workspace\temp\hybris\extgen_final [mkdir] Созданный

каталог: C:\workspace\bin\custom\mysampleextension\lib [copy] Копирование 215 файлов в C:

\workspace\bin\custom\mysampleextension [echo]

[эхо]

[эхо] Дальнейшие шаги:

[эхо]

[echo] 1) Добавьте свое расширение в C:\workspace\config\localextensions.xml [echo]

[эхо] <extension dir="C:\workspace\bin\custom\mysampleextension"/>

[эхо]

[echo] 2) Перед первой сборкой расширения убедитесь, что сервер приложений остановлен. [echo]

[echo] 3) Выполните команду 'ant' в каталоге hybris/platform. [echo]

[echo] 4) Перезапустите сервер приложений [echo]

[эхо]

СТРОИТЬ УСПЕШНО

Следуйте за [следующие шаги](#) подсказки по включению нового расширения в установку платформы hybris. Замените `commerceseveb-servicesextension` на `localextensions.xml`.

```
<!-- <extension dir="${HYBRIS_BIN_DIR}/ext-hybris/commerceseveb-services"/> -->
<extension dir="${HYBRIS_BIN_DIR}/custom/mysampleextension"/>
```

Редактировать расширение `info.xml` для вашего индивидуального расширения Commerce Services, чтобы обеспечить веб-корень контекст, который лучше всего соответствует вашему проекту, например:

```
<extensioninfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="extensioninfo.xsd">
```

```
    <расширение name="mysampleextension"
        abstractclassprefix="Сгенерировано"
        classprefix="mysampleextension"
    >
```

```

<требуется-расширение имя="commercefacades"/>
<требуется-расширение имя="commerceservices"/>
<webmodule jspcompile="false" webroot="/mysampleextension"/>

</расширение>
</extensioninfo>

```

Укажите информацию о базовом корневом пути для расширения ваших веб-сервисов в свойства проекта:

```

...
commercewebservices.rootcontext=/mysampleextension/v1/...

```

Более подробную информацию о создании расширения см.[Создание нового расширения](#).

Добавление новых ресурсов

Вы можете добавлять новые ресурсы для настроенных Commerce Services, определяя новые методы обработчиков в контроллерах веб-сервисов. Чтобы эффективно использовать Commerce Services, вам следует ознакомиться со следующими аннотациями Spring MVC:

- @Контроллер
- @RequestMapping, оба используются на уровне класса и метода
- @ResponseBody
- @RequestParam
- @ПутьПеременной

Давайте создадим контроллер, который будет выполнять отображение в /v1/ сделки/**path. Содержащийся метод обработчика должен специально сопоставляться с HTTP GET-запросами к этому контроллеру. Обратите внимание, что @ЗапросКартографирование аннотация на уровне класса автоматически требует, чтобы все входящие запросы имели Принимать заголовок со значением приложение/xml или приложение/json. Это два формата представления, поддерживаемые Commerce Services «из коробки».

DealsController.java

```

@Контроллер
@RequestMapping(value = "/v1/deals", headers = "Accept=application/xml,application/json") открытый класс DealsController
расширяет BaseController
{
    @RequestMapping(метод = RequestMethod.GET)
    @ResponseBody
    public DealList doSomething(финальный запрос HttpServletRequest) {
        ...
    }
}

```

XML/JSON-сериализация возвращаемых объектов

-Примечание

Информация, включенная ниже, относится к V1 Commerce Services. Для информации, посвященной V2, см.: [Преобразователи HTTP-сообщений](#) и [Преобразователи HTTP-сообщений](#).

Сериализация возвращаемого объекта в XML или JSON не требует использования аннотаций в целевых классах. Commerce Services включают библиотеку Xstream для выполнения сериализации возвращаемых объектов. Настройка сериализации может быть выполнена в Spring.

Пользовательская сериализация XML/JSON

Мы создали базовую конфигурацию Spring, которая обеспечивает пользовательские сопоставления Spring для настройки сериализованного вывода API веб-интерфейса Commerce Services.

Эта конфигурация состоит из:

- АтрибутOmitMapping
- FieldAliasMapping
- ТипПсевдонимСоответствие

ТипПсевдонимСоответствие

ТипПсевдонимСоответствие используется для предоставления псевдонима сериализованному элементу, например:

```

<bean class="de.hybris.platform.commercefacades.xstream.alias.TypeAliasMapping">
    <имя_свойства="псевдоним" значение="продукт" />
    <имя_свойства="aliasedClass" значение="de.hybris.platform.commercefacades.product.data.ProductData" />
</боб>

```

Предыдущий код переименовывает все экземпляры сериализованных данных о продукте в ответе с соответствующим продуктОбъект JSON или продукттег в XML-ответе.

FieldAliasMapping

Позволяет повторно назначать псевдонимы атрибутам в сериализованных объектах, например:

```
<bean class="de.hybris.platform.commercefacades.xstream.alias.FieldAliasMapping">
    <имя свойства="псевдоним" значение="класс" /> <имя
    свойства="имя поля" значение="имя_класса" />
    <имя свойства="aliasClass" значение="de.company.customwebservices.dto.ErrorData" />
</боб>
```

Код переименует атрибутОшибкаДанныеэкземпляры из оригиналаclassNamекорт.

АтрибутOmitMapping

Это сопоставление пропускает поле из целевого объекта во время сериализации, например:

```
<bean class="de.hybris.platform.commercefacades.xstream.alias.AttributeOmitMapping">
    <имя свойства="имя_атрибута" значение="код" />
    <имя свойства="aliasClass" значение="de.hybris.platform.commerceservices.search.facetdata.FacetData" />
</боб>
```

Код удаляет атрибут code из сериализованных JSON и XML.FacetDataобъекты.

Сопутствующая информация

[Расширение commercefacades - Техническое руководство](#)

Интеграция с Adobe Experience Manager

SAP Commerce легко интегрируется с Adobe Experience Manager (AEM). Это позволяет вам подключить решение SAP Commerce, отвечающее за управление данными о продуктах, корзинами покупок, оформлением заказов и выполнением заказов, в то время как AEM управляет отображением данных и маркетинговыми кампаниями.

Предоставленную в готовом виде реализацию можно расширить, включив в нее функцию поиска.

Обзор установки

Перед началом интеграции вам необходимо установить SAP Commerce и Adobe Experience Manager.

Установка SAP Commerce

-Примечание

В этом разделе описывается только индивидуальная конфигурация и дается общее представление о процессе установки.

Более подробную информацию о процессе установки см.[Установка и обновление SAP Commerce](#).

Более подробную информацию о функции Express Update см.[Включение экспресс-обновления продукта](#).

. Сначала извлеките последний артефакт в папку назначения. Для этого примера назовите его<КАТАЛОГ>.

. Далее, установите ваш рабочий каталог на<КАТАЛОГ>/hybris/bin/платформа выполните следующие команды.

```
.. ./setantenv.sh
муравей чистый
```

. На этом этапе все файлы cong должны быть сгенерированы и присутствовать в<КАТАЛОГ>/hybris/конфигурациякаталог.

. В примере ниже показан минимальный набор расширений, необходимый для нашей интеграции. Однако реальный набор будет зависеть от конкретных потребностей клиентов, использующих эту интеграцию в своих средах.

```
<hybrisconfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="resources/schemas/extensions.x
    <расширения>
        <!--
            Все расширения, расположенные в ${HYBRIS_BIN_DIR}/platform/ext, загружаются автоматически.
            Более подробную информацию о настройке доступных расширений можно найти здесь: https://help.sap
        -->
        <path dir="${HYBRIS_BIN_DIR}" />
        <!-- расширенная платформа -->
        <расширение имя="productcockpit" />
        <расширение имя="yacceleratorstorefront" />
        <расширение имя="yacceleratorinitialdata" />
        <расширение имя="electronicsstore" />
        <расширение имя="sampledata" />
```

```
<расширение имя="ycommercewebservices" />
</расширения>
</hybrisconfig>
```

. Теперь сохраните этот контент `blocalextensions.xml` в `KATAЛОГ/hybris/конфигурациякatalog.`

. По умолчанию платформа использует HSQL для хранения данных. Для лучшей производительности, особенно на больших каталогах, рекомендуется использовать базу данных MySQL на сервере разработки. Для использования MySQL необходимо указать конфигурацию сервера MySQL вместные свойства в `ДИР/hybris/конфигурациякatalog.`

```
db.url=jdbc:mysql://localhost/<имя_базы_данных>?useConfigs=maxPerformance&characterEncoding=utf8
db.driver=com.mysql.jdbc.Driver
```

```
# ПАРОЛЬ БД ДОЛЖЕН БЫТЬ ИЗМЕНЕН В СООТВЕТСТВИИ С ВАШЕЙ УСТАНОВКОЙ MYSQL
db.username=<имя пользователя>
db.password=<пароль>
db.tableprefix=
mysql.optional.tabledefs=CHARSET=utf8 COLLATE=utf8_bin
mysql.tablename=InnoDB
```

. Чтобы включить функцию Express Update, необходимо указать, для какой версии каталога она должна быть доступна. В случае приведенного ниже примера в очередь Express Update можно добавлять только продукты из версии Online catalog.

```
expressUpdateCatalogVersion=Онлайн
```

. Когда конфигурация будет готова, вы можете собрать и инициализировать платформу.

муравей все инициализирует

Повышение производительности

Примечание

Если вы используете сервер MySQL в сочетании с таблицами базы данных на основе InnoDB, вы можете ощутить огромное влияние на производительность. Рекомендуется установить переменную `innodb_ush_log_at trx_commit` на 0. Используйте консоль MySQL для установки этой опции:

```
mysql> установить глобальный innodb_flush_log_at_trx_commit=0;
```

Если ваш сервер разработки имеет более одного ядра, вы можете установить дополнительные параметры для ускорения процесса инициализации, а также импорта и синхронизации каталогов в дальнейшем.

```
# ОПТИМИЗАЦИЯ
# ВНИМАНИЕ! - количество воркеров impex.import и catalog.sync может иметь максимальное значение, равное количеству ядер
impex.import.workers=8
каталог.sync.workers=8
build.parallel=true
```

Вы также можете настроить параметры кэширования с помощью следующих опций.

```
кэш.основной=120000
regioncache.entityregion.size=120000
```

Adobe AEM

. Загрузите последнюю версию CQ5, например cq-quickstart-5.6.1.jar

. Переименуйте файл в cq5-author-p4502.jar и запустите его с пользовательскими параметрами:

```
java -XX:MaxPermSize=280M -Xmx1100M -jar cq5-author-p4502.jar
```

. Во время первого запуска все пакеты должны быть установлены, процесс может занять несколько минут. Когда сервер будет готов, вы сможете получить к нему доступ в своем браузере: `http://localhost:4502`. Вы можете использовать учетные данные по умолчанию:

- Логин: администратор
- пароль: используйте пароль, который вы определили для своей учетной записи.

. Используйте Adobe's Package Share для установки последних сборок SAP Commerce-пакетов. Пакеты, которые вы хотите установить:

- cq-hybris-сервер
- cq-hybris-контент
- cq-geometrixx-hybris-контент

Встроенная SAP Commerce

Если у вас есть cq-hybris-сервер После установки пакета Platform запустится автоматически вместе с AEM. Вам не нужно ничего делать, так как он уже предварительно настроен и работает из коробки. Помните, что встроенное решение включает базу данных HSQL, которая влияет на производительность.

Автономная SAP Commerce

Если вы хотите интегрировать AEM с автономной установкой SAP Commerce, вам необходимо настроить URL-адрес для нужного экземпляра:

. Перейти `http://localhost:4502/system/console/configMgr`.

. Найдите Day CQ Commerce Hybris Connection Handler и щелкните по нему.

. Теперь измените переменную базового URL во всплывающем окне.

. При необходимости вы также можете изменить другие значения.

. Протестируйте интеграцию, импортировав каталог из Платформы:<http://localhost:4502/etc/importers/hybris.html>.

Сопутствующая информация

[Включение обновления Product Express](#)

[Удаление старых корзин с помощью Cronjob](#)

Удаление старых корзин с помощью Cronjob

Корзины не удаляются после закрытия сеанса. Таким образом, вы все еще можете восстановить корзину позже, когда пользователь снова войдет в сеть. Однако, чтобы избежать хранения большого количества старых корзин, вы можете использовать cronjob для их удаления.

Cronjob

TheOldCartRemovalCronJob тип элемента определен в `ycommercewebservices-items.xml`. Есть три атрибута, определяющие, какие тележки следует удалить:

- сайты: Сбор сайтов, на которых будут удалены старые корзины.
- КорзинаУдалениеВозраст: Корзины старше указанного количества секунд будут удалены.
- anonymousCartRemovalAge: Анонимные корзины старше указанного количества секунд будут удалены.

`ycommercewebservices-items.xml`

```
...
<itemtype code="OldCartRemovalCronJob" autocreate="true"
generate="true" extends="CronJob"
jaloClass="de.hybris.platform.ycommercewebservices.jalo.OldCartRemovalCronJob"> <атрибуты>
<attribute type="BaseSiteCollection" квалификатор="сайты">
    <модификаторы/>
    <настойчивость тип="свойство"/>
    <description>Базовые сайты, для которых следует удалить старые корзины</description>
</атрибут>
<тип атрибута="java.lang.Integer" квалификатор="cartRemovalAge">
    <модификаторы/>
    <настойчивость тип="свойство"/>
    <defaultValue>Целое число.valueOf(2419200)</defaultValue>
    <description>После указанного количества секунд корзины будут очищены. По умолчанию 28 дней.</description>
</тип атрибута>
<тип атрибута="java.lang.Integer" квалификатор="anonymousCartRemovalAge">
    <модификаторы/>
    <настойчивость тип="свойство"/>
    <defaultValue>Целое число.valueOf(1209600)</defaultValue>
    <description>После указанного количества секунд корзины будут очищены. По умолчанию 14 дней.</description>
</тип атрибута>
</атрибуты>
</тип_элемента>
...

```

Объект работы

TheOldCartRemovalJob класс определяет логику, отвечающую за удаление старых карт. Соответствующий бин определяется в `ycommercewebservices-весна.xml`. Смотрите пример кода для примера.

```
...
<bean id="oldCartRemovalJob" class="de.hybris.platform.ycommercewebservices.cronjob.OldCartRemovalJob" родитель="abstractJobPer
    <свойство имя="commerceCartDao" ref="commerceCartDao"/>
    <свойство имя="timeService" ref="timeService"/>
    <свойство имя="userService" ref="userService"/>
</боб>
...

```

Крок

Образец OldCartRemovalCronJob создается во время процесса init/update. Он остается активным и запускается ежедневно в 3:30 утра. Скрипт, расположенный в `ycommercewebservices/pecups/B` каталоге представлен пример добавления cronjob.

```
...
# Очистка старой корзины CronJobs
# Это всего лишь пример задания cron, но чтобы оно работало правильно, вам также необходимо задать для него свойство sites
INSERT_UPDATE OldCartRemovalCronJob;код[уникальный=истина];задание(код);возрастудалениякорзины;анонимныйвозрастудалениякорзины;язык сеанса(isoCode)[по
умолчанию ;oldCartRemovalCronJob;oldCartRemovalJob;2419200;1209600

INSERT_UPDATE Триггер;cronJob(код)
[уникальный=истина];секунда;минута;час;день;месяц;год;относительный;активный;maxAcceptableDelay ;oldCartRemovalCronJob;0;30;3;-1;-1;ложь;истина;-1
...

```

Примечание

The commerce web-service расширение не знает о сайтах, доступных в системе. В результате образец cronjob не имеет сайты в набор атрибутов.

Для корректной работы вам необходимо настроить сайты Атрибут. Сделать это можно непосредственно в файле скрипта ImpEx или с помощью визуального интерфейса Backoffice.

Пример добавления сайта атрибут с помощью скрипта ImpEx ле.

```
# Настройка CronJobs очистки старой корзины для сайта wsTest
INSERT_UPDATE
OldCartRemovalCronJob; код[уникальный=истина]; задание(код); сайты(uid); oldCartRemovalCronJob; oldCartRemovalJob; wsTest
```

Добавление сайтов

Контекст

Используйте панель администрирования бэк-офиса для добавления сайта атрибут.

Процедура

- . Перейдите к [главному](#) в [редакторе](#) найдите oldCartRemovalCronJob. Щелкните запись.

Откроется редактор.

- . Перейти к [Сайты администрации](#)

- . Начните вводить название сайта, который вы хотите добавить, или воспользуйтесь окном поиска ссылок.

- . Теперь в список включены сайты, для которых старые корзины будут удалены выбранным cronJob.

- . Сохраните вашу конфигурацию.

Расширение коммерческих услуг

Коммерческие услуги можно расширить, создав AddOn с одним дополнительным классом и использование соответствующих методов. Однако в большинстве случаев требуется более сложные изменения для создания AddOn, который добавляет функциональность к Commerce Services.

В этом документе описывается, как успешно расширить коммерческие услуги.

-Примечание

Подробную информацию о создании надстройки для веб-служб OCC см. здесь:[Создание надстройки для веб-сервисов OCC](#).

Процесс расширения описан на основе оккадон. Это пример AddOn, который расширяет Commerce Services. оккадон добавляет новые свойства в Клиент Тип товара:

- прозвище
- workOfficeAddress

Вызов REST позволяет вам задать эти свойства. В оккадон, также приведен пример того, как переопределить существующее сопоставление запроса.

Расширение объектов данных

Вы можете расширить модель данных и объект передачи данных для Commerce Services с помощью AddOn.

Расширение модели данных

Процедура

- . В имя_расширения-items.xml файле, замените имя расширения часть в файле с именем фактического расширения.

Образец AddOn расширяет Модель клиента класс с двумя свойствами:

- прозвище
- workOfficeAddress

-Примечание

Более подробную информацию см. [Элементы XML](#).

- . Убедитесь, что вы видите правильные результаты в *-элементы.xml файле.

*-элементы.xml файл выглядит как следующий пример.

occaddon-items.xml

```
<?xml версия="1.0" кодировка="ISO-8859-1"?>
<элементы xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="items.xsd">
    <типы элементов>
        <typegroup name="Клиент">
            <itemtype code="Клиент" autocreate="false" generate="false">
                <description>Расширение типа Customer дополнительными атрибутами.</description> <attributes>
                    <attribute autocreate="true" qualifier="nickname" type="java.lang.String">
                        <modifiers read="true" write="true" необязательно="true" /> <persistence
                        type="property" />
                        <description>Псевдоним клиента</description> </attribute>
                    <attribute autocreate="true" qualifier="workOfficeAddress" type="Address" isSelectionOf="addresses"> <modifiers read="true" write="true"
                           search="false" Optional="true" />
                        <persistence type="property" qualifier="WorkOfficeAddress"/> </attribute>
                    </атрибуты>
                </тип элемента>
            </typegroup>
        </itemtypes>
    </элементы>
```

Расширение объекта передачи данных

Процедура

- . В имя_расширения-beans.xml файле, замените <имя_расширения> часть с названием текущего расширения.

Образец AddOn расширяет Данные клиента класс с двумя свойствами:

- прозвище
- workOfficeAddress

-Примечание

Подробную информацию о создании пользовательских Java Beans см.[Генерация бинов и перечислений](#).

. Убедитесь, что вы видите правильные результаты в *.beans.xml.

occaddon-бобов.xml

```
<?xml версия="1.0" кодировка="ISO-8859-1"?> xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<бобы>
    xsi:noNamespaceSchemaLocation="beans.xsd">
        <!-- Дополнительные псевдоним и свойство workOfficeAddress для CustomerData -->
        <bean class="de.hybris.platform.commercefacades.user.data.CustomerData" extends="de.hybris.platform.commercefacades.user.data.AddressData">
            <имя свойства="псевдоним" тип="Строка" />
            <имя_свойства="workOfficeAddress" тип="de.hybris.platform.commercefacades.user.data.AddressData"/>
        </боб>
</бобов>
```

Заполнение данных

Контекст

The Конвертеры объекты в <коммерция>-* расширения настраиваются только в Spring и не имеют отдельных конкретных классов реализации. В процессе преобразования они используют список Популяторы объекты. Можно добавить Популяторы существующему Конвертеру без необходимости повторного объявления Конвертера. Используйте модифицировать PopulateList метод.

Эта процедура обеспечивает шаги по заполнению новых атрибутов.

Процедура

. Определите класс Populator в OccaddonCustomerPopulator.java.

```
открытый класс OccaddonCustomerPopulator реализует Populator<CustomerModel, CustomerData> {
    частный преобразователь<AddressModel, AddressData> addressConverter;
    @Override
    public void populate(конечный источник CustomerModel, конечный целевой CustomerData) выдает ConversionException {
        Assert.notNull(source, "Параметр source не может быть нулевым.");
        Assert.notNull(target, "Параметр target не может быть нулевым.");
        target.setNickname(source.getNickname());
        если (source.getWorkOfficeAddress() != null) {
            target.setWorkOfficeAddress(getAddressConverter().convert(source.getWorkOfficeAddress()));
        }
    }
    защищенный преобразователь<AddressModel, AddressData> getAddressConverter() {
        обратный адресКонвертер;
    }
    @Необходимый
    public void setAddressConverter(final Converter<AddressModel, AddressData> addressConverter) {
        этот.преобразовательадресов = преобразовательадресов;
    }
}
```

. Добавьте Populator в Converter в occaddon-spring.xml.

```
...
<псевдоним имя="defaultOccaddonCustomerPopulator" псевдоним="occaddonCustomerPopulator"/> идентификатор="defaultOccaddonCustomerPopulator"
<боб класс="de.hybris.platform.occaddon.customer.converters.populator.OccaddonCust
    <имя_свойства="конвертер_адресов" ref="конвертер_адресов"/>
</боб>

<bean parent="modifyPopulatorList">
    <свойство имя="список" ref="customerConverter"/>
    <свойство имя="добавить" ref="occaddonCustomerPopulator"/>
</боб>
...
```

Следующие шаги

Те же действия необходимо выполнить в обратном порядке Конвертер.

-Примечание

Более подробную информацию о преобразователях и пополнителях см.[Конвертеры и пополнители](#).

Локализация атрибутов

Процедура

Добавить <ключ>=<локализованная строка> записи в следующем файле локализации системы типов:occaddon/resources/localization/occaddonlocales_en.properties.

occaddon-locales_en.properties

```
type.Customer.nickname.name=Псевдоним
type.Customer.nickname.description=Описание псевдонима
type.Customer.workOfficeAddress.name=Рабочий офис      адрес
type.Customer.workOfficeAddress.description=Работа      офис      адрес
```

Расширение объекта передачи данных (DTO) для v2

Контекст

Для v2 Commerce Services есть дополнительный слой DTO. Он был создан для улучшения стабильности и конфигурируемости данных ответа.

Чтобы использовать новый DTO и настроить ответы, выполните следующие действия:

-Примечание

Для получения более подробной информации см. [Концепция WsDTO](#).

Процедура

. Расширить DTO вoccaddon-beans.xml.

```
...
<!-- Дополнительные псевдоним и свойство workOfficeAddress для UserWsDTO --> <bean
    class="de.hybris.platform.commercewebservicescommons.dto.user.UserWsDTO"
    extends="de.hybris.platform.commercewebservicescommons.dto.user.PrincipalWsDTO"> <имя
    свойства="псевдоним" type="String" />
    <имя_свойства="workOfficeAddress" тип="de.hybris.platform.commercewebservicescommons.dto.user.AddressWsDTO"/> </bean>
...
...
```

. Добавьте новые поля к заранее определенным конфигурациям уровня поля вoccaddon-web-spring.xml.

Помните, что вам нужно сделать это в <имя_дополнения>-web-весна.xmlе локализован в каталоге ресурсов, который добавляется в контекст Commerce Services.

```
...
<bean parent="fieldSetLevelMapping"> <имя
    свойства="dtoClass"
    значение="de.hybris.platform.commercewebservicescommons.dto.user.UserWsDTO" <свойство      />
    имя="levelMapping">
    <карта>
        <entry key="BASIC" value="псевдоним" /> <entry
        key="DEFAULT" value="псевдоним" />
        <entry key="FULL" value="псевдоним,workOfficeAddress(FULL)" /> </map>
    </карта>
</свойство>
</боб>
...

```

Дополнительные ресурсы

○ Для получения дополнительной информации о контексте web spring см. [Расширение коммерческих услуг](#).

. Заполнение данных между коммерческими данными и веб-сервисами DTO. Заполнение данных из коммерческого уровня в веб-сервисы DTO выполняется с помощью Орика - популярный фреймворк Java Bean mapper. Поля с одинаковыми именами заполняются автоматически методами из версии v2 с помощью объекта DataMapper.

ПользователиController.java

```
...
public UserWsDTO getUser(@RequestParam(defaultValue = "BASIC") final String fields) {
    final CustomerData customerData = customerFacade.getCurrentCustomer(); final UserWsDTO dto =
    dataMapper.map(customerData, UserWsDTO.class, fields); return dto;
}
...
```

Вы также можете использовать DataMapper в контроллерах AddOn. Реализация DataMapper по умолчанию определена в коммерция веб-сервисы commons расширение. Чтобы использовать его в AddOn, определите зависимость для этого расширения и добавьте соответствующий ресурс в контроллер:

... Контроллер.java

```
...
@Resource(name = "dataMapper")
защищенный DataMapper dataMapper; ...
```

Расширение REST API

Вы можете расширить REST API для коммерческих сервисов с помощью AddOn.

Определение контроллера

Контекст

Чтобы выявить новые вызовы, определите Контроллер класс с соответствующими методами.

Процедура

Создайте Контроллер в /acceleratoraddon/web/src/de/hybris/platform/acceleratorwebservicesaddon/controllers каталог.

РасширенныйCustomersController.java

```
/**
 * Контроллер, расширяющий ресурсы клиента
 */
@Controller("sampleExtendedCustomerController")
@RequestMapping(value = "/{baseSiteId}/customers") открытый класс
ExtendedCustomersController
{
    ...
    @Secured("ROLE_CUSTOMERGROUP")
    @RequestMapping(значение = "/current/nickname", метод = RequestMethod.GET) @ResponseBody
    публичная строка получитьКлиентНик()
    {
        окончательная строка имя = getCustomerFacade().getCurrentCustomer().getNickname(); возвращаемое
        имя;
    }

    @Secured("ROLE_CUSTOMERGROUP")
    @RequestMapping(значение = "/current/nickname", метод = RequestMethod.PUT) @ResponseBody
    public CustomerData setCustomerNickname(@RequestParam final String nickname) выдает DuplicateUidException {
        окончательные данные клиента customer = customerFacade.getCurrentCustomer();
        customer.setNickname(nickname);
        return customerFacade.getCurrentCustomer();
    }
    ...
}
```

-Примечание

Когда вы создаете контроллеры AddOn не следует использовать класс, определенный в commercewebservices. Имя пакета для такого класса меняется после екстген процесс завершен.

Создание контекста Web Spring

Контекст

Так как контроллер образца аннотирован как @Контроллер, Всему нужно указать, где ее искать.

Процедура

В каталоге ресурсов обновите <имя_дополнения>-web-весна.xml файле.

Файл расширяет контекст Commerce Services. Этот файл не является стандартным файлом веб-контекста из web\WEB-INF\catalog.

-Примечание

Если ваш AddOn был создан из йокаддон шаблон, конфигурация сканирования компонентов для <ваш_пакет_дополнения>.controllers установлено.

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://
       www.springframework.org/schema/aop" xmlns:context="http://
       www.springframework.org/schema/context" xsi:schemaLocation="http://
       www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-3.1.xsd http://
       www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context-3.1.xsd http://
       www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop-3.1.xsd">

    <context:component-scan base-package="de.hybris.platform.occaddon.controllers"/>
</beans>
```

Добавление веб-контекста в коммерческие услуги

Контекст

Расширьте контекст Commerce Services с помощью контекста из недавно созданного AddOn. Вы можете сделать это с помощью дополнительных Web Spring Configs Механизм. свойства проекта файл должен содержать следующую строку, где Конфигурация Spring Classpath указывает на любой ресурс classpath из <addonExtension>:

```
<targetExtension>.additionalWebSpringConfigs.<addonExtension>=[Spring Config Classpath]
```

Процедура

Добавьте следующую строку в проект.свойства.шаблоне.

проект.свойства.шаблон

```
ycommercewebservices.additionalWebSpringConfigs.<имя_надстройки>=classpath:/occaddon/web/spring/<имя_надстройки>-web-spring.xml
```

Этот файл является шаблоном для свойства проекта, который создается в процессе установки.

-Примечание

Если ваш AddOn был создан из окна AddOn, проект.свойства.шаблон уже должен иметь надлежащее содержание.

Проверьте новые конечные точки, определенные в контроллере

-Примечание

Если вы используете пример кода в [Расширение объектов данных](#), вам необходимо перестроить и инициализировать/обновить торговую систему, чтобы применить новые определенные атрибуты в Клиенттип и новые бобы. Подробности см. [Установка и обновление SAP Commerce](#).

Поток запроса:

Метод	URL	Заголовок	Параметры тела
поста	https://localhost9002/authorizationserver/oauth/token	Содержание-Тип: приложение/x-www-form-urlencoded	client_id=\$CLIENT_ID\$&client_secret=\$CLIENT_SECRET\$&grant_type=password
помещать	https://localhost9002/occ/v2/electronics/users/ 8716fdb8-6cea-4e28-8396-1258e819f758/никнейм • 8716fdb8-6cea-4e28-8396-1258e819f758 — идентификатор пользователя.	Содержание-Тип: приложение/x-www-form-urlencoded	nickname=мойПсевдоним
получить	https://localhost9002/occ/v2/electronics/users/ 8716fdb8-6cea-4e28-8396-1258e819f758/никнейм • 8716fdb8-6cea-4e28-8396-1258e819f758 — идентификатор пользователя.		

Переопределение REST API

Контекст

В AddOn для Commerce Services вы можете переопределить существующие вызовы. Единственное различие между расширением и переопределением REST API заключается в использовании `@RequestMappingOverride` аннотации.

Процедура

Используйте `@ЗапросПереопределениеКарты` для установки приоритетов для методов с одинаковым `@ЗапросКартографирования`.

`ЗапросПереопределениеКарты`

```
@Цель(
{ ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME) public @interface
RequestMappingOverride {

    /**
     * Имя свойства, в котором хранится значение приоритета
     */
    Стока priorityProperty() по умолчанию "";
}
```

Используется метод с наивысшим приоритетом.

-Осторожность

`@RequestMappingOverride` следует использовать для переопределения идентичной аннотации `@RequestMapping`.

Однако он не будет работать правильно в ситуации, когда исходное сопоставление запроса поддерживает два метода HTTP, а вы попытаетесь переопределить только один из них:

- запрос сопоставления для исходного метода = `@RequestMapping(value = "/current/addresses/default/{id}", method = { RequestMethod.PUT, RequestMethod.POST })`
- запрос сопоставления для метода, который должен переопределить оригинал = `@RequestMapping(value = "/current/addresses/default/{id}", method = RequestMethod.PUT)`

Даже пример ниже рассматривается как другое отображение:

- запрос сопоставления для исходного метода = `@RequestMapping(value = "/{productCode}", method = RequestMethod.GET)`
- запрос сопоставления для метода, который должен переопределить оригинал = `@RequestMapping(value = "/{productId}", method = RequestMethod.GET)`

В таких случаях методы не будут переопределены, а во время запроса (а не во время запуска платформы) возникнет ошибка «Неоднозначные методы обработчика...».

ThePriority значение считывается из свойства `le` (проект.свойства, локальные.свойства) на основе:

- приоритетСвойств определено в аннотации или
- имя свойства `requestMappingOverride.<имя_класса>.<имя_метода>.priority`

Пример:

`requestMappingOverride.de.hybris.platform.occaddon.controllers.ExtendedCustomersController.updateDefaultAddress.priority`

Если в файле свойств не определено свойство, значение приоритета устанавливается равным нулю.

Пример:

- `@RequestMappingOverride(priorityProperty="occaddon.updateDefaultAddress.priority")` -здесь значение приоритета считывается из свойства `occaddon.updateDefaultAddress.priority`.
- `@RequestMappingOverride` -здесь значение приоритета считывается из `requestMappingOverride.<имя_класса>.<имя_метода>.priority` имущество (например `requestMappingOverride.de.hybris.platform.occaddon.controllers.ExtendedCustomersController.updateDefaultAddress.priority`).

-Примечание

ThePriority значение считывается из файла свойств для разрешения ситуации, когда более одного метода переопределяют исходный вызов. В этом случае вы можете выбрать предпочтительный метод, установив для него наивысшее значение приоритета вместе свойства.

Пример переопределения сопоставления запросов

`РасширенныйCustomerController.java`

```
/**
 * Контроллер, расширяющий ресурсы клиента
 */
@Controller("sampleExtendedCustomerController")
@RequestMapping(value = "/{baseSiteId}/customers") открытый класс
ExtendedCustomersController
{
...
/** 
 * Это пример переопределения существующего сопоставления запросов. Аннотация {@link RequestMappingOverride} позволяет переопределить
 * существующее сопоставление запросов, определенное аннотацией {@link RequestMapping}.
 */
@Secured("ROLE_CUSTOMERGROUP")
@RequestMapping(значение = "/current/addresses/default/{id}", метод = RequestMethod.PUT)
@RequestMappingOverride
@ResponseBody
public String updateDefaultAddress(@PathVariable final String id) выдает DuplicateUidException {

    конечный адрес AddressData = userFacade.getAddressForCode(id);
    userFacade.setDefaultAddress(address);
    return "Адрес был успешно обновлен методом из sampleExtendedCustomerController";
}
...
}
```

Назначение новых вызовов определенной версии API

Контекст

Для Commerce Services доступны две версии (v1 и v2). Обе версии доступны одновременно. В результате вам может потребоваться указать версию для вызова из AddOn. Для этого вы можете использовать **Версия API** аннотацию.

```
/**
 * Аннотацию можно использовать для контроллеров. Она позволяет ограничить видимость методов, аннотированных с помощью
 * {@code @RequestMapping} только для выбранной версии коммерческих веб-сервисов (например, v1 или v2).
 */
@Цель(
    {ТипЭлемента.ТИП})
@Retention(RetentionPolicy.RUNTIME)
публичный @interface ApiVersion
{
    /**
     * Возвращает версию API, для которой следует зарегистрировать методы из контроллера (например, v1).
     */
    Строковое значение();
}
```

Данную аннотацию можно использовать на контроллеруровень.

```
@Controller("sampleExtendedCustomerController")
@RequestMapping(value = "/{baseSiteId}/customers")
@ApiVersion("v1")
открытый класс ExtendedCustomersController {
...
}
```

-Примечание

Если нет **Версия API** аннотации, назначенная контроллеру в AddOn, методы из этого контроллера добавляются для всех доступных версий.

Например - метод, аннотированный `@RequestMapping(value = "/{baseSiteId}/customers/current/nickname")` доступны из URI `/rest/v1/{baseSiteId}/customers/current/nickname` и `/rest/v2/{baseSiteId}/customers/current/nickname`.

Расширение кэширования на стороне сервера

Кэширование на стороне сервера доступно для AddOns. Чтобы добавить свой кэш в существующую конфигурацию, используйте `CacheManagerListbean` для хранения всех менеджеров кэша (реализации `org.springframework.cache.CacheManager` интерфейс), которые используются.

Процедура

Используйте директиву List Merge для расширения списка из AddOn.

<!-- Менеджер кэша для occaddon -->

```

<псевдоним имя="defaultOccAddonCacheManager" alias="occAddonCacheManager"/>
<боб идентификатор="defaultOccAddonCacheManager" target="org.springframework.cache.ehcache.EhCacheCacheManager">
    <имя свойства="cacheManager" ref="occAddonEhcache"/>
</боб>

<псевдоним имя="defaultOccAddonEhcache" alias="occAddonEhcache"/>
<боб идентификатор="defaultOccAddonEhcache" class="de.hybris.platform.commercewebservicescommons.cache.TenantAwareEhCacheManagerFactoryByProperty">
    <свойство имя="префикс_имени_кэша" значение="occAddonCache_"/> значение="/WEB-INF/cache addons/
    <свойство имя="configLocation" value="occaddon/ehcache.xml"/>
</боб>

<bean зависит-от="wsCacheManagerList" parent="listMergeDirective">
    <имя свойства="add" ref="occAddonCacheManager" />
</боб>

```

Этот пример доступен в окнадон.

-Примечание

Помните, что все описанные здесь бобы должны быть помещены в ИМЯ_ДОПОЛНЕНИЯ/resources/ИМЯ_ДОПОЛНЕНИЯ/web/spring/ИМЯ_ДОПОЛНЕНИЯ-web-spring.xmlе.

В примере по умолчанию используется библиотека Ehcache. Вы можете заменить библиотеку любой другой библиотекой кэша, поддерживаемой Spring Framework (т. е. Guava, GemFire, JSR-107).

Последнее, что вам нужно сделать (при использовании конфигурации по умолчанию, созданной в окнадоншаблон) - это изменить ehcache.xmlе, расположенный в ИМЯ_ДОПОЛНЕНИЯ/acceleratoraddon/web/webroot/WEB-INF/cachekatalog и настройте его в соответствии со своими потребностями.

Сопутствующая информация

[Кэширование](#)

Расширение пакетов сообщений

Вы можете использовать AddOn для определения пакета сообщений, доступного в Commerce Services.

Процедура

Определите пакет сообщений в /acceleratoraddon/web/webroot/WEB-INF/messageskatalog.

ExtendedUsersController.java

```

@Controller("sampleExtendedUserController")
@RequestMapping(value = "/{baseSiteId}/users")
@ApiVersion("V2")
открытый класс ExtendedUsersController {

    ...
    @Resource(name = "messageSource") защищенный
    MessageSource messageSource;
    ...

    @ResponseStatus(значение = HttpStatus.PAYMENT_REQUIRED)
    @ResponseBody
    @ExceptionHandler({WebserviceValidationException.class})
    public ErrorListWsDTO handleWebserviceValidationException(final WebserviceValidationException ex) {

        окончательный ErrorListWsDTO errorListDto = handleErrorInternal (ex); вернуть
        errorListDto;
    }
    защищенный ErrorListWsDTO handleErrorInternal(final WebserviceValidationException ex) {

        окончательный ErrorListWsDTO errorListDto = новый ErrorListWsDTO ();
        окончательный Локаль locale = i18nService.getLocaleForLanguage(i18nService.getCurrentLanguage()); окончательные
        Ошибки ошибки = (Ошибки) ex.getValidationObject();
        окончательный список<ErrorWsDTO> errorList = ошибки.getAllErrors().stream().map(eo -> mapError(eo, locale))
            .собрать(Collectors.toList());
    }
}

```

```

errorListDto.setErrors(список ошибок); вернуть
errorListDto;
}

protected ErrorWsDTO mapError (окончательная ошибка ObjectError, окончательный языковой стандарт) {

    окончательный результат ErrorWsDTO = новый ErrorWsDTO();
    окончательная строка message = messageSource.getMessage(error.getCode(), error.getArguments(), locale);
    result.setMessage(message);
    result.setReason("Проверка не пройдена");
    result.setType("Ошибка");
    вернуть результат;
}
}

```

Все файлы из этого расположения копируются во время фазы сборки в /web/webroot/WEB-INF/messages/addons/имя_аддона/каталог коммерческих услуг и автоматически загружается Источник сообщения выполнение.

Пакет сообщений, определенный в AddOn, виден в классе из Commerce Services, который использует Источник сообщения фасоль.

Обычный Источник сообщения bean используется контроллерами AddOn. В примере используется Источник сообщения для обработки исключений.

AddonAwareMessageSource класс представляет собой пользовательскую реализацию Источника сообщения Интерфейс, предоставляемый Commerce Services. Он настроен на сканирование целевого каталога /WEB-INF/сообщения/дополнения/ для любого xml-характеристики Файлы, которые можно использовать как пакеты сообщений. Сканирование и индексация файлов выполняется один раз во время инициализации бина.

AddonAwareMessageSource предоставляет следующие дополнительные свойства для настройки:

- baseAddonDir - обычно /WEB-INF/сообщения/дополнения/, он сканирует все файлы в этом каталоге и его подкаталогах.
- leFilter - фильтр для файлов, которые должны быть загружены. По умолчанию фильтр загружает все xml-характеристики лес.
- dirFilter - фильтр для подкаталогов. По умолчанию сканируются все подкаталоги.

Сопутствующая информация

[Генерация бинов и перечислений](#)