Code Review - NEO - IAM Consulting - Iteration2

SAP Customer Experience

SAP Hybris

DOCUMENT REVISION HISTORY

Date	Revision	Author	Summary of Changes
19/02/2019	1.0	y.gatla@sap.com	Draft

SAP® Run Simple

TABLE OF CONTENTS

Introduction

The Code Review is typically delivered iteratively during the engineering phase of the project to ensure that any quality and efficiency issues are addressed early on and often. The review validates that accepted industry and SAP Commerce Cloud / SAP Commerce Solution best practices are followed to minimize the risk of performance issues and that the code base will not be difficult to maintain.
Without an independent review, there is a risk that the delivered solution is not of high quality and doesn't follow best practices for SAP Commerce Cloud or SAP Commerce Solution projects. This could introduce performance and scalability issues and make the solution more difficult to maintain and upgrade to newer releases.
This is the phase 2 code review mainly to cover the new functionalities added in Sprint 6, 7. For this review, we used the code version from the branch "*sap_20190218*" *(https://bitbucket.org/SAI-IT/neo-tl/branch/sap_20190218)* with revision "*d573a66*".

Activities Scope

This review is mainly aimed at Sprint 6, 7 changes (delta code review) covering the following User stories

- Business partner service and managment
- User location and Filter products based on user location
- Update Shopping to add Pick Up In Store delivery method
- Update Order Splitting Rule and remove Round Robin sequence
- Create Consignment process for Pick up In Store
- Create Business partner landing page
- Grant Permission for BP to run customize report
- Grant permission for BP to cancel/return product
- Import stock level from csv
- Export sale transaction to excel file
- Product Approval Workflow
- Allow to calculate shipping cost for normal consignment based on each BP

This review covers other areas in the custom code to ensure that it meets the SAP Commerce Solution implementation best practices and patterns. The review covers static-code analysis, data model and type checking, and a manual review of the source code. In each section we provide a list of recommendations for the issues identified. Please note that this is a source-code review and not a full-solution-architecture review.

- The **Data Model** is analysed to ensure that SAP Commerce best practices have been followed.
- All **Source Code** is manually reviewed for best use of APIs, and for general Java best practices, mainly focusing on the new functionalities added in Sprint 6,7.
- The **Configuration** of the project is reviewed to ensure the correct use of modules, project data files and dependencies between extensions.

Executive summary

This section summarizes recommendations from the code review. A more detailed description of the recommendation can be found in the corresponding section.
This review found the code to be in a decent state

- Many of the hybris development patterns were found to be in good use.
- The static code analysis found moderate amount of issues that require attention.
- The Java source code quality is normally good, with some areas needing improvement.

However, there are concerns

- Improve the data model following the recommendations.
- There are issues with the project/environment set up, we suggest to fix them on a priority basis.
- There are some issues in the code that can impact performance, if neglected these problems could lead to degradation of performance in a production environment.
- Some of the areas can be improved to follow the Hybris coding standards.

Findings that are part of this review are presented within each subsection of this document. They are written in this report using conventions, divided into four categories as shown below.

Assertion
A given practice follows recommended standards and best practices.

Improvement
A given practice is valid and doesn't offers any risk to the solution, but there's still some room for improvements.

Recommendation
This recommendation should be reviewed and factored into upcoming releases as feasible.

Issue
An issue should be reviewed and designated as a priority as a part of a near term release.

Summary of Findings

	Description	**Priority**
Project Setup		
PS-01	Review the usage of below deprecated extensions and remove if not necessary. 1. Instore module 2. mcc And remove any other redundant/unused modules. Please refer 6.7 documentation for more details https://help.hybris.com/6.7.0/hcd/8bb15ed586691014a948d1553f4947cf.html	Medium
PS-02	Resolve the impex errors and make sure essential/initial data set up works with standard set up	High
PS-03	Make sure site works without issues in the local developer machine with the standard steps.	Medium
PS-04	Review the impex files to 1. Add the missing essential, seed data required for the site, if any 2. Clean up the duplicate rows issues 3. Resolve the errors	Medium
Source Code Analysis		
SC-01	If not in place already, Make sure you add Sonar evaluations on your development / Continuous Integration process. • Setup a SonarQube server • Configure Sonar in the SAP Commerce project and include sonar target in the continuous integration process. • Regularly check rules and fix them accordingly. You can also configure alerts that are triggered when a metric reaches a certain threshold	High
SC-02	Duplication of code is inside an acceptable threshold. Nevertheless, we still recommend checking this value often.	Low
SC-03	Avoid too complex methods and classes (with a class complexity greater than 10). This metric should be lowered to ensure maintainability and readability through the use of utility methods.	Medium
SC-04	Blocker and Critical issues found should be reviewed and fixed as soon as possible. For detailed information, please check the attached Sonar report	High
SC-05	Review the duplicate bean declaration and change the configuration.	Medium
Data Model		
DM-01	Change the type codes to 110XX series (with unused ones) for the above mentioned types to avoid any future migration issues	High

DM-02	A deployment table should not be defined for any item extending types that are not GenericItem. Consider removing the deployment table for the indicated items.	High
DM-03	Consider reviewing the Jaloclass attribute when autocreate='false' and generate='false'.	High
DM-04	Attribute will only be writeable during item creation and as such, consider setting mandatory fields to 'true' or using a default value (where optional='false').	Medium
DM-05	For coding best practices, attribute names should start with an lowercase letter.	Medium
DM-06	Set relations that have cardinality='many' as not ordered (ordered='false')	Low
DM-07	Consider adding indexes covering all unique attributes of the noted Item types	Medium
DM-08	Consider whether noted types require unique identifiers	Medium
Visual Code Analysis		
VC-01	Review all the After save listeners in order to 1. Move the business logic from After Save listeners to the corresponding business service layer. 2. Remove hard coding of the type codes	High
VC-02	Avoid redundant cartFacade.getSessionCart() calls in the code.	Medium
VC-03	Avoid calling modelservice.save() in loops, instead use saveAll() after the loop wherever possible	Medium
VC-04	Avoid using modelService saveAll() , with out parameters	Medium
VC-05	Use modelservice.saveAll(Collection) instead of multiple save() calls in the same method or block, wherever possible	Medium
VC-06	Review all the usages of product options and reduce the number of options wherever possible to have less impact on the performance. Some of them are mentioned below, but there can be more places to review • com.neo.codengai.storefront.controllers.pages.ProductPageController#showQuickView • com.neo.codengai.storefront.controllers.pages.AccountWishlistPageController#removeWishlistItemPopup • com.neo.codengai.facades.user.impl.NeoCustomerFacadeImpl#convertPageData • com.neo.codengai.facades.populators.NeoWishlist2EntryPopulator#populate • com.neo.codengai.storefront.controllers.pages.ProductPageController#populateProductDetailForDisplay The above method referenced in places like ProductPageController#writeReview(), ProductPageController#postReview() where we doesn't seem to require these many options while it makes sense to use many options for ProductPageController#productDetail(). So the best option is to write a separate method at different places with suitable product options.	High
VC-07	Remove the extra call to the productFacade.filterProductReferenceForCurrentLocation() in the productDetails flow.	High
VC-08	Optimize productFacade.getProductForCodeAndOptions() with required options only and call once rather than doing multiple calls in the method (check the calling methods also). Review if any other instances of this kind of code.	Medium
VC-09	Make sure that all custom jobs in ***com.neo.codengai.core.job*** package are abortable.	Medium
VC-10	Avoid using System.out.println to log messages. It could be a bottleneck for performance and should be replaced by log4j.	Medium
VC-11	Avoid controller driven development. Controllers should not be aware of Models, Business Services, DAOs and HTML.	High
VC-12	Avoid facade-layer conversion from a DTO to a Model. Use a converter instead, with appropriate populators.	Medium
VC-13	Review the service classes with Flexible search code and move the logic to DAOs There is one unused service class using flexible search class, which can be cleaned up. Other than this, no service-layer-flexible-search query has been identified.	Low
VC-14	It is important that there are no instances of wiring converters into populator implementations.	Medium
VC-15	Review the following interceptors as they have been found to have complex business logic: • NeoStockLevelValidateInterceptor • NeoBusinessPartnerValidateInterceptor	Medium

	• NeoProductValidateInterceptor • NeoBusinessPartnerPrepareInterceptor • NeoCustomerValidateInterceptor • PriceRowTimeRangeValidateInterceptor	
VC-16	Do not search for beans by type or alias. It is advised to not use Registry.getCoreApplicationContext() because this method always returns the core ApplicationContext. Instead the Registry.getApplicationContext checks first if there is a ServletContext currently holding a WebApplicationContext. If that is the case, this one is returned, which is indeed your web ApplicationContext configured at your web.xml file. If there is no current ServletContext or no ApplicationContext set at it, the global ApplicationContext is returned. With that you do not have to be aware if your context is a Core Module or a Web Module, you always get the correct ApplicationContext. Be aware that if you do not connect your web ApplicationContext to the global or core ApplicationContext, you cannot access global or core beans via the ApplicationContext returned by Registry.getApplicationContext. If you have built up the connection by simply using the SAP Commerce SolutionContextLoaderListener, you have access to all the beans of your web and the global and core ApplicationContext, unless you have defined overlaying bean definitions in your web ApplicationContext. `MyBeanType myBean = (MyBeanType)Registry.getApplicationContext().getBean("<extname>.mybean");`	Medium
VC-17	Fix all custom bean names to start with lowercase.	Medium
VC-18	Review the whole project, declare all dependencies in xml configuration file and add @Required annotation to the setters.	Medium
VC-19	Usually we recommend using de.hybris.platform.servicelayer.config.ConfigurationService rather than de.hybris.platform.util.Config.	Low
VC-20	UserService should be used instead of JaloSession and UserManager.	Medium
VC-21	Be aware of highly utilized areas in the application which could have a performance impact and introduce measures to counteract this. Use caching in areas such as Populators and Converters to bypass repeated conversion.	Medium
VC-22	Avoid building query strings in the method, rather declare at class level.	Medium
VC-23	Evaluate/Review the need of SQL statements, which do not use the Query cache (due to milliseconds/seconds time stamps) considering the business need.	warning
VC-24	Review the possibility to improve the logic in the API *com.neo.codengai.storefront.controllers.cms.NeoBestSellingProductsComponentController#fillModel*	Medium

Next Steps

After this review, we recommend the following next steps:

1. Review all issues and recommendations, adjusting the project as necessary.
2. After implementing all features a Performance Review is recommended to ensure this solution will be able to handle expected traffic and volumes
3. Before Go-Live it is recommended to do System Review, which would evaluate this solution in a much broader aspect

Overview

This section defines areas that relate to the overall project and its setup. These items are important to the overall health and success of a project even though they do not deliver customer-facing functionality.

Extension Configuration

Here is the current setup found in **localextensions.xml** file.

```
<hybrisconfig xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='../bin/platform/resources/schemas/extensions.xsd'>
  <extensions>
    <path dir='${HYBRIS_BIN_DIR}' autoload='false' />
   <extension name='ldap' />
    <extension name='mcc' />
    <extension name='adaptivesearchsolr' />
    <extension name='adaptivesearchbackoffice' />
    <extension name='adaptivesearchsamplesaddon' />
    <extension name='adaptivesearchwebservices' />
    <extension name='commerceservicesbackoffice' />
    <extension name='solrfacetsearchbackoffice' />
    <extension name='solrserver' />
   <extension name='previewwebservices' />
    <extension name='ycommercewebservices' />
    <extension name='ycommercewebservicestest' />

    <extension name='acceleratorwebservicesaddon' />
    <extension name='orderselfserviceaddon' />
    <extension name='customersupportbackoffice' />
   <extension name='codengaiCTaddon' />
    <extension name='commerceorgsamplesaddon' />
    <extension name='rulebuilderbackoffice' />
    <extension name='couponbackoffice' />
    <extension name='droolsruleengineservices' />
   <extension name='ruleengineservices' />
    <extension name='couponfacades' />
    <extension name='promotionenginesamplesaddon' />
    <extension name='cmswebservices' />
    <extension name='smarteditwebservices' />
    <extension name='cmssmarteditwebservices' />
    <extension name='permissionswebservices' />
    <extension name='smarteditaddon' />
    <extension name='cmssmartedit' />
    <extension name='cmsbackoffice' />
    <extension name='previewpersonalizationweb' />
    <extension name='personalizationcmsweb' />
    <extension name='personalizationsmartedit' />
    <extension name='textfieldconfiguratortemplatebackoffice' />
    <extension name='textfieldconfiguratortemplateaddon' />
    <extension name='codengaiASMstorefront' />
    <extension name='assistedservicewebservices' />
    <extension name='assistedservicepromotionaddon' />
   <extension name='ordermanagementaddon' />
   <extension name='warehousing' />
    <extension name='warehousingbackoffice' />
    <extension name='warehousingwebservices' />
   <extension name='instore' />
   <extension name='consignmenttrackingaddon' />
   <extension name='consignmenttrackingbackoffice' />
   <extension name='ordermanagementwebservices' />
   <extension name='addonsupport' />

   <extension name='codengaicockpits' />
   <extension name='codengaicore' />
   <extension name='codengaifacades' />
   <extension name='codengaiinitialdata' />
    <extension name='codengaistorefront' />
   <extension name='codengaiordermanagement' />
   <extension name='codengaibackoffice' />
   <extension name="mediaconversion" />
    <extension name="mediaconversionbackoffice" />

   <extension name="wishlist" />
   <extension name="codengailinepays"/>
   <extension name="codengaicsbackoffice"/>
   <extension name="codengaiwhbackoffice"/>
   <extension name="codengaikbank"/>

  </extensions>
</hybrisconfig>
```

PS-01

Review the usage of below deprecated extensions and remove if not necessary.

1. Instore module
2. mcc

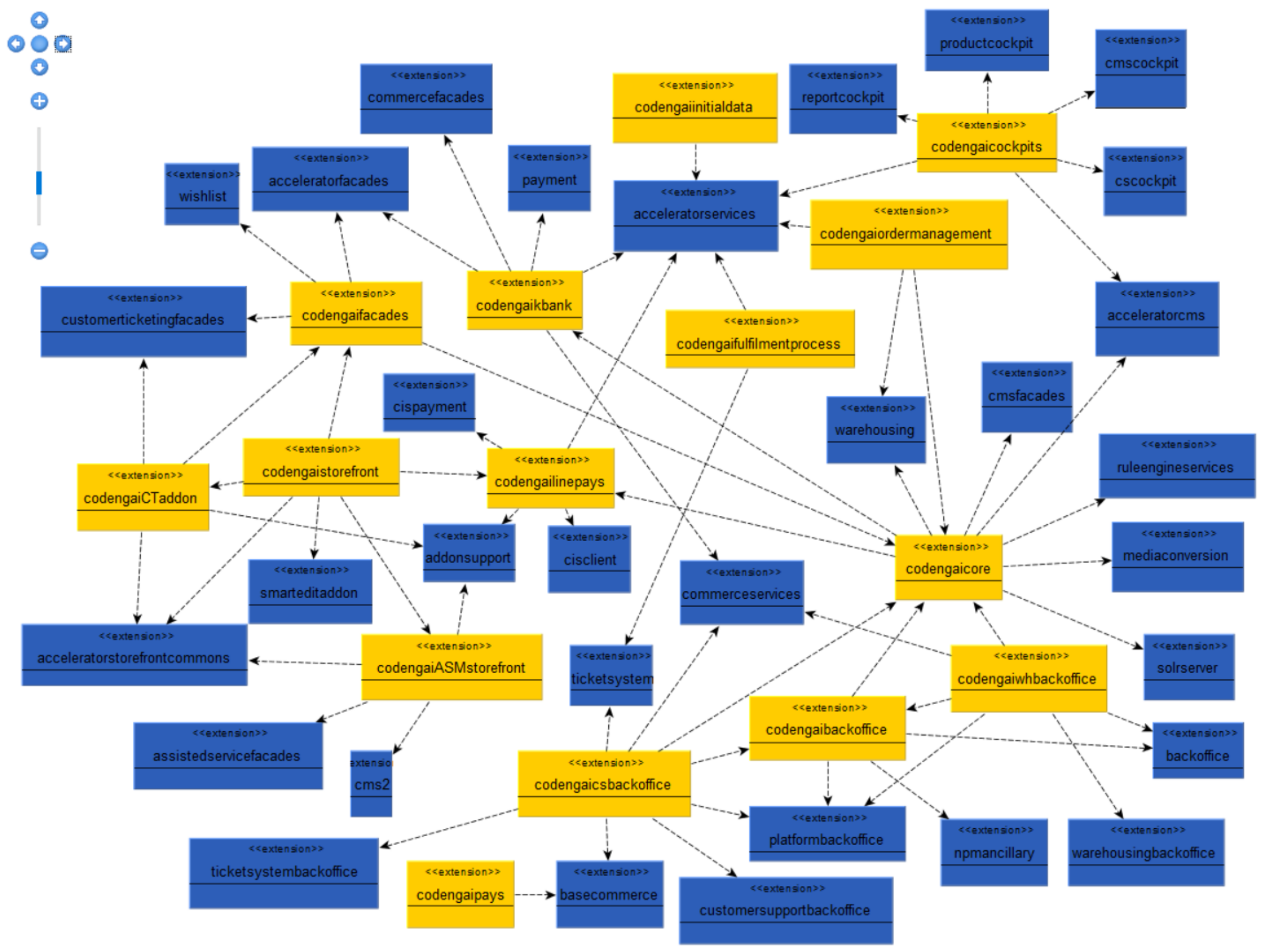
And remove any other redundant/unused modules.

Please refer 6.7 documentation for more details

https://help.hybris.com/6.7.0/hcd/8bb15ed586691014a948d1553f4947cf.html

SAP Commerce Solution recommends using different extensions for the layers of the application (web, facades, services), as well as for integrations with other systems in order to reinforce a separation of concerns.

It's important to ensure the dependencies are correctly configured for a project to ensure we can reuse functionality without introducing cyclic dependencies that might force us to copy and paste code between extensions of move code into the wrong layer of an application.



This project uses recommended dependency structure and does not include any cyclical dependencies.

Local Environment Set up

During initialization/update, there were some impex issues noticed and site working partially. Some of the impex errors are mentioned below.

```
INFO   | jvm 1    | main    | 2019/02/19 10:35:38.774 | ERROR [hybrisHTTP23]
(000000UI) [CronJobErrorHandler] de.hybris.platform.impex.jalo.ImpExException: Can
not resolve any more lines ... Aborting further passes (at pass 2). Finally could
not import 2 lines![HY-123]
INFO   | jvm 1    | main    | 2019/02/19 10:35:38.774 | ERROR [hybrisHTTP23]
[DefaultImportService] Import has caused an error, see logs of cronjob with
code=000000UI for further details
INFO   | jvm 1    | main    | 2019/02/19 10:35:38.774 | ERROR [hybrisHTTP23]
[DefaultSetupImpexService] Importing
[/codengaiinitialdata/import/coredata/common/essential-data_en.impex]... FAILED


INFO   | jvm 1    | main    | 2019/02/19 10:35:38.894 | ERROR [hybrisHTTP23]
(000000UJ) [CronJobErrorHandler] de.hybris.platform.impex.jalo.ImpExException: Can
not resolve any more lines ... Aborting further passes (at pass 2). Finally could
not import 2 lines![HY-123]
INFO   | jvm 1    | main    | 2019/02/19 10:35:39.004 | ERROR [hybrisHTTP23]
[DefaultImportService] Import has caused an error, see logs of cronjob with
code=000000UJ for further details
INFO   | jvm 1    | main    | 2019/02/19 10:35:39.004 | ERROR [hybrisHTTP23]
[DefaultSetupImpexService] Importing
[/codengaiinitialdata/import/coredata/common/essential-data_th.impex]... FAILED
```

```
INFO   | jvm 1    | main    | 2019/02/19 10:36:11.275 | ERROR [hybrisHTTP23]
(000000W4) [CronJobErrorHandler] de.hybris.platform.impex.jalo.ImpExException: Can
not resolve any more lines ... Aborting further passes (at pass 2). Finally could
not import 1 lines![HY-123]
INFO   | jvm 1    | main    | 2019/02/19 10:36:11.275 | ERROR [hybrisHTTP23]
[DefaultImportService] Import has caused an error, see logs of cronjob with
code=000000W4 for further details
INFO   | jvm 1    | main    | 2019/02/19 10:36:11.275 | ERROR [hybrisHTTP23]
[DefaultSetupImpexService] Importing
[/codengaiinitialdata/import/sampledata/contentCatalogs/codengaiContentCatalog/cms-
businessPartner-page.impex]... FAILED


INFO   | jvm 1    | main    | 2019/02/19 10:36:13.654 | ERROR [hybrisHTTP23]
[DefaultImportService] Import has caused an error, see logs of cronjob with
code=000000W8 for further details
INFO   | jvm 1    | main    | 2019/02/19 10:36:13.654 | ERROR [hybrisHTTP23]
[DefaultSetupImpexService] Importing
[/codengaiinitialdata/import/coredata/common/product-workflow.impex]... FAILED
```

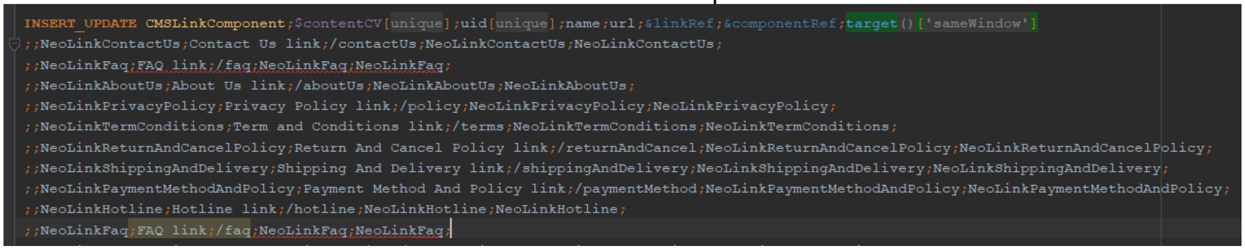
PS-02
Resolve the impex errors and make sure essential/initial data set up works with standard set up

PS-03
Make sure site works without issues in the local developer machine with the standard steps.

Project data

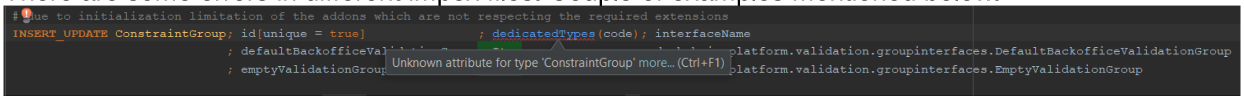
SAP Commerce Solution recommends that each project create the seed data for provisioning a new system in the form of ImpEx files. This allows new development to be setup very quickly. The distinction is made between essential data that is not expected to change between environments (Languages, Countries, Catalogs, etc.) and sample project data (Products, CMS Components, Media, etc.). Essential data and sample project data should be able to be loaded independently during a system update and it should be possible to disable the loading of sample project data entirely.
Some of the issues noticed in the impex are

1. Duplicate rows in impexes will impact on the impex performance. The value is overridden with same value like below. There are few cases like this in different impex files



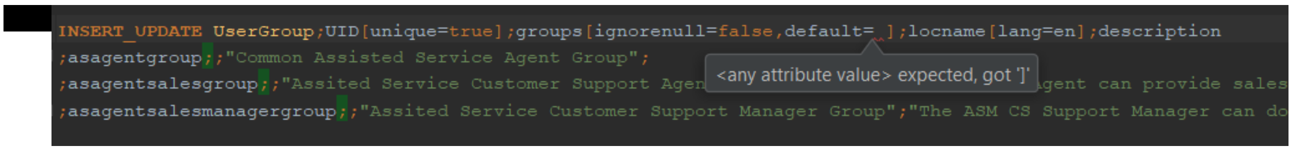
```
INSERT_UPDATE CMSLinkComponent;$contentCV[unique];uid[unique];name;url;&linkRef;&componentRef;target()['sameWindow']
;;NeoLinkContactUs;Contact Us link;/contactUs;NeoLinkContactUs;NeoLinkContactUs;
;;NeoLinkFaq;FAQ link;/faq;NeoLinkFaq;NeoLinkFaq;
;;NeoLinkAboutUs;About Us link;/aboutUs;NeoLinkAboutUs;NeoLinkAboutUs;
;;NeoLinkPrivacyPolicy;Privacy Policy link;/policy;NeoLinkPrivacyPolicy;NeoLinkPrivacyPolicy;
;;NeoLinkTermConditions;Term and Conditions link;/terms;NeoLinkTermConditions;NeoLinkTermConditions;
;;NeoLinkReturnAndCancelPolicy;Return And Cancel Policy link;/returnAndCancel;NeoLinkReturnAndCancelPolicy;NeoLinkReturnAndCancelPolicy;
;;NeoLinkShippingAndDelivery;Shipping And Delivery link;/shippingAndDelivery;NeoLinkShippingAndDelivery;NeoLinkShippingAndDelivery;
;;NeoLinkPaymentMethodAndPolicy;Payment Method And Policy link;/paymentMethod;NeoLinkPaymentMethodAndPolicy;NeoLinkPaymentMethodAndPolicy;
;;NeoLinkHotline;Hotline link;/hotline;NeoLinkHotline;NeoLinkHotline;
;;NeoLinkFaq;FAQ link;/faq;NeoLinkFaq;NeoLinkFaq;
```

2. There are some errors in different impex files. Couple of examples mentioned below.



```
#due to initialization limitation of the addons which are not respecting the required extensions
INSERT_UPDATE ConstraintGroup; id[unique = true]            ; dedicatedTypes(code); interfaceName
                             ; defaultBackofficeVal[illegible]                  [illegible]platform.validation.groupinterfaces.DefaultBackofficeValidationGroup
                             ; emptyValidationGroup                             [illegible]platform.validation.groupinterfaces.EmptyValidationGroup
```

Unknown attribute for type 'ConstraintGroup' more... (Ctrl+F1)



```
INSERT_UPDATE UserGroup;UID[unique=true];groups[ignorenull=false,default= ];locname[lang=en];description
;asagentgroup;;"Common Assisted Service Agent Group";
;asagentsalesgroup;;"Assited Service Customer Support Agen                                    gent can provide sales
;asagentsalesmanagergroup;;"Assited Service Customer Support Manager Group";"The ASM CS Support Manager can do
```

<any attribute value> expected, got ']'

PS-04

Review the impex files to

1. Add the missing essential, seed data required for the site, if any
2. Clean up the duplicate rows issues
3. Resolve the errors

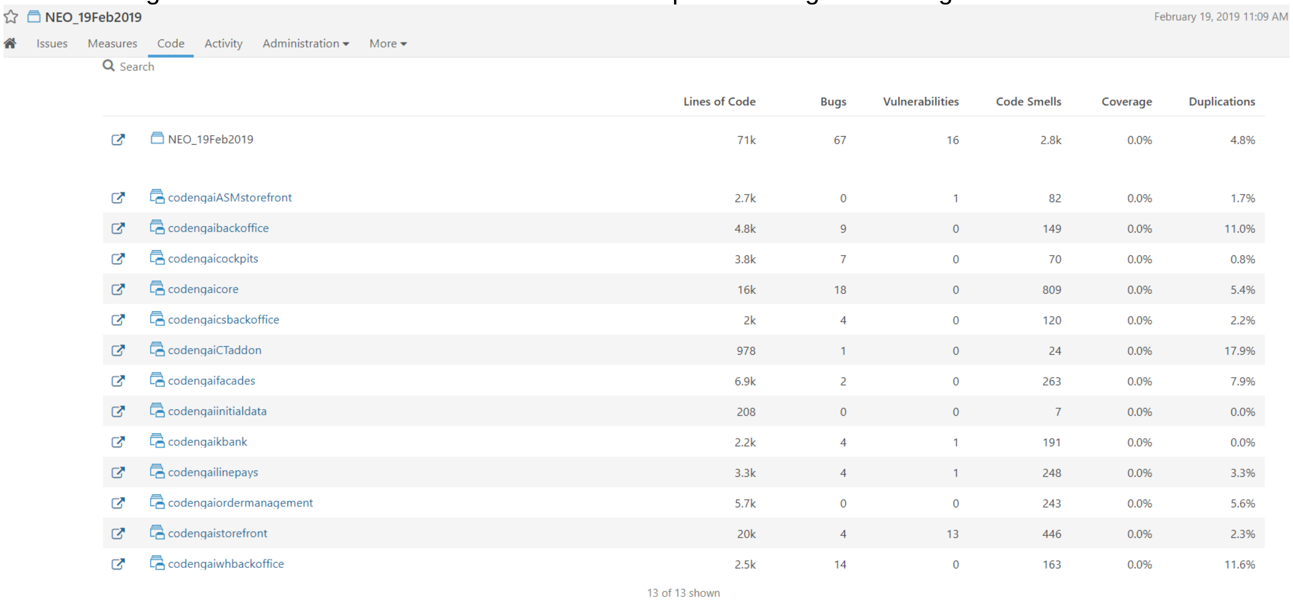
Overview

In the scope of this review, we used a tool named SonarQube. This tool uses static rules to identify potential problems in this project's custom code.

- Centralizes the management of rules to be used by the entire team.
- PMD and Checkstyle rules could directly link from Eclipse to those defined in the SonarQube server.
- Show progress and trends on the quality of the code overtime to the entire team.
- Support additional types of rules and metrics (e.g FindBugs).

Evaluation Scope

We have run SonarQube against the custom extensions implemented for this project and compared them with the results obtained from the SAP Commerce Accelerator (OOTB version) of same version.using the same configuration and rules. SonarQube was configured to run the (PMD, CPD, and FB) rule sets.
The following dashboard contains an overview of all components targeted during this evaluation.

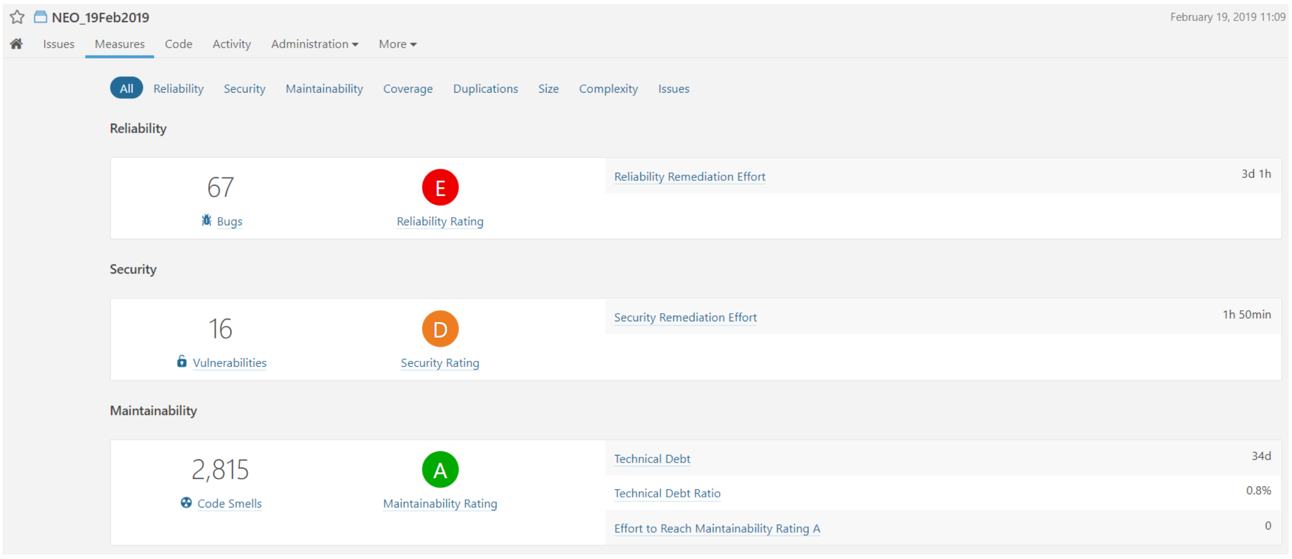


	Lines of Code	Bugs	Vulnerabilities	Code Smells	Coverage	Duplications
NEO_19Feb2019	71k	67	16	2.8k	0.0%	4.8%
codengaiASMstorefront	2.7k	0	1	82	0.0%	1.7%
codengaibackoffice	4.8k	9	0	149	0.0%	11.0%
codengaicockpits	3.8k	7	0	70	0.0%	0.8%
codengaicore	16k	18	0	809	0.0%	5.4%
codengaicsbackoffice	2k	4	0	120	0.0%	2.2%
codengaiCTaddon	978	1	0	24	0.0%	17.9%
codengaifacades	6.9k	2	0	263	0.0%	7.9%
codengaiinitialdata	208	0	0	7	0.0%	0.0%
codengaikbank	2.2k	4	1	191	0.0%	0.0%
codengailinepays	3.3k	4	1	248	0.0%	3.3%
codengaiordermanagement	5.7k	0	0	243	0.0%	5.6%
codengaistorefront	20k	4	13	446	0.0%	2.3%
codengaiwhbackoffice	2.5k	14	0	163	0.0%	11.6%

13 of 13 shown

Sonar Results

The following dashboard contains an overview of the current status of the code.


NEO_19Feb2019 February 19, 2019 11:09

Issues | Measures | Code | Activity | Administration | More

All | Reliability | Security | Maintainability | Coverage | Duplications | Size | Complexity | Issues

Reliability

67 Bugs	E Reliability Rating	Reliability Remediation Effort	3d 1h

Security

16 Vulnerabilities	D Security Rating	Security Remediation Effort	1h 50min

Maintainability

2,815 Code Smells	A Maintainability Rating	Technical Debt	34d
		Technical Debt Ratio	0.8%
		Effort to Reach Maintainability Rating A	0

SC-01
If not in place already, Make sure you add Sonar evaluations on your development / Continuous Integration process.

- Setup a SonarQube server
- Configure Sonar in the SAP Commerce project and include sonar target in the continuous integration process.
- Regularly check rules and fix them accordingly. You can also configure alerts that are triggered when a metric reaches a certain threshold

Reference for Sonar Metric Definitions

Extracted from: https://docs.sonarqube.org/display/SONAR/Metric+Definitions

Severity Types

Severity	Description
Blocker	Operational/security ***risk***: This issue might make the whole application unstable in production. Ex: calling garbage collector, not closing a socket, etc.
Critical	Operational/security ***risk***: This issue might lead to an unexpected behavior in production without impacting the integrity of the whole application. Ex: NullPointerException, badly caught exceptions, lack of unit tests, etc.
Major	This issue might have a substantial impact on ***productivity***. Ex: too complex methods, package cycles, etc.
Minor	This issue might have a potential and minor impact on ***productivity***. Ex: naming conventions, Finalizer does nothing but call superclass finalizer, etc.
Info	Unknown or not yet well defined security risk or impact on productivity.

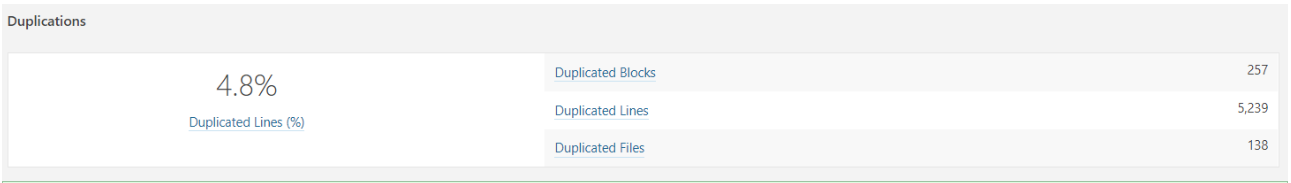
Maintainability

Name	Description
Code Smells	Number of code smells.
New Code Smells	Number of new code smells.
Maintainability Rating (formerly SQALE Rating)	Rating given to your project related to the value of your Technical Debt Ratio. The default Maintainability Rating grid is: A=0-0.05, B=0.06-0.1, C=0.11-0.20, D=0.21-0.5, E=0.51-1 The Maintainability Rating scale can be alternately stated by saying that if the outstanding remediation cost is: • <=5% of the time that has already gone into the application, the rating is A • between 6 to 10% the rating is a B • between 11 to 20% the rating is a C

	• between 21 to 50% the rating is a D • anything over 50% is an E
Technical Debt	Effort to fix all maintainability issues. The measure is stored in minutes in the DB. An 8-hour day is assumed when values are shown in days.
Technical Debt on new code	Technical Debt of new code
Technical Debt Ratio	Ratio between the cost to develop the software and the cost to fix it. The Technical Debt Ratio formula is: Remediation cost / Development cost Which can be restated as: Remediation cost / (Cost to develop 1 line of code * Number of lines of code) The value of the cost to develop a line of code is 0.06 days.
Technical Debt Ratio on new code	Ratio between the cost to develop the code changed in the leak period and the cost of the issues linked to it.

CPD Analysis

This automated analysis summarizes the code fragments that are duplicated in custom code. Only fragments longer than 30 lines of code are shown. Here SAP Commerce will provide a summary of the duplication while the full report will be provided with SAP Commerce' final review document.
Detailed information regarding duplicated code (packages, classes, etc) can be found in the Sonar Report, attached to this review.



Duplications

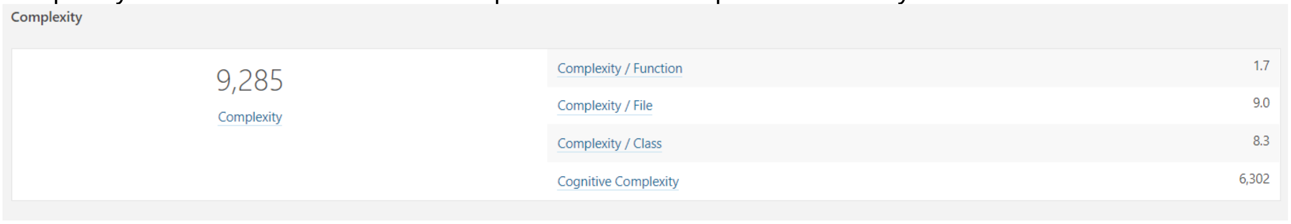
4.8%
Duplicated Lines (%)

Duplicated Blocks	257
Duplicated Lines	5,239
Duplicated Files	138

SC-02
Duplication of code is inside an acceptable threshold. Nevertheless, we still recommend checking this value often.

Complexity Report

This metric measures the cyclomatic complexity, also known as McCabe metric. Whenever the control flow of a function splits, the complexity counter gets incremented by one. Each function has a minimum complexity of 1. It is recommended to keep a low value to improve readability and maintenance of the code.



Complexity

9,285
Complexity

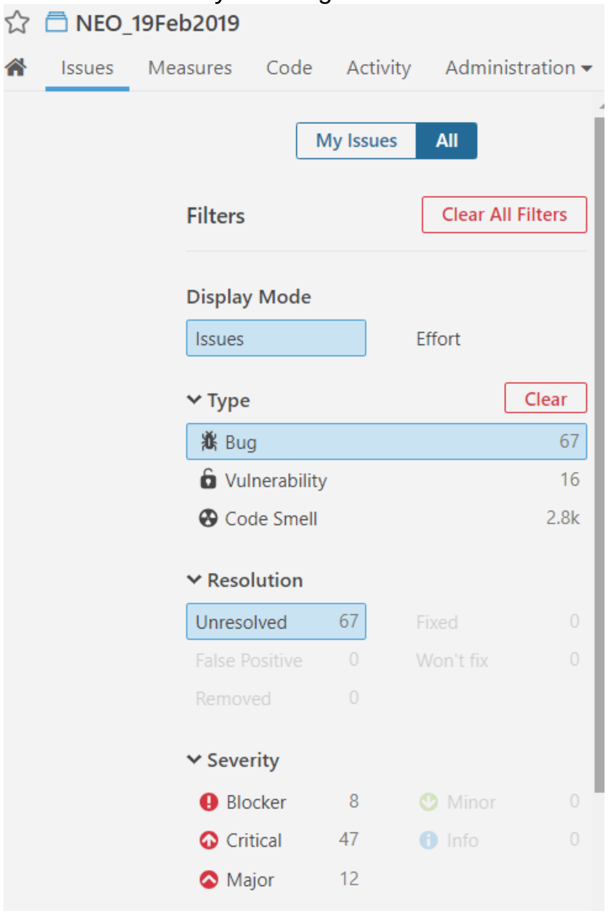
Complexity / Function	1.7
Complexity / File	9.0
Complexity / Class	8.3
Cognitive Complexity	6,302

SC-03
Avoid too complex methods and classes (with a class complexity greater than 10). This metric should be lowered to ensure maintainability and readability through the use of utility methods.

FindBugs and PMD

FindBugs is a program that uses static analysis to look for bugs in Java code; while the PMD tool uses static rules to identify potential problems in the custom code (http://pmd.sourceforge.net). The output from these tools was analyzed and we highlight some of the highest priority items.

[redacted]ns, suboptimal code, and many other types of potential issues. The tests uncovered **8** Blockers, **47** critical issues, **12** major issues. The full report will be provided with SAP Commerce' final review document.
Custom code was analyzed both visually and using static-source-code-analysis tools such as PMD. The results of the analysis along with recommendations are provided below.



NEO_19Feb2019

Issues | Measures | Code | Activity | Administration

My Issues | All

Filters — Clear All Filters

Display Mode: Issues | Effort

Type	
Bug	67
Vulnerability	16
Code Smell	2.8k

Resolution			
Unresolved	67	Fixed	0
False Positive	0	Won't fix	0
Removed	0		

Severity			
Blocker	8	Minor	0
Critical	47	Info	0
Major	12		

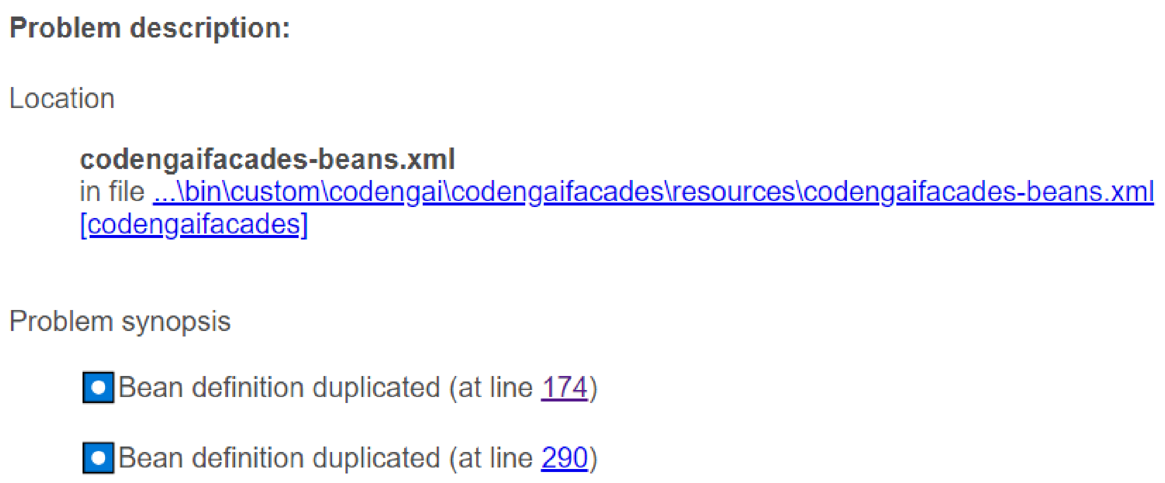
Full Sonar CSV report is attached here NEO_19Feb2019-sonar-csv-report.csv. Please refer this file for more details on these issues.

SC-04
Blocker and Critical issues found should be reviewed and fixed as soon as possible. For detailed information, please check the attached Sonar report.

Code Inspection Issues

Duplicate bean definition

The bean "de.hybris.platform.commercefacades.order.data.AbstractOrderData" is declared two times in spring configuration file codengaifacades\resources\codengaifacades-beans.xml



Problem description:

Location

codengaifacades-beans.xml
in file ...\bin\custom\codengai\codengaifacades\resources\codengaifacades-beans.xml [codengaifacades]

Problem synopsis

Bean definition duplicated (at line 174)

Bean definition duplicated (at line 290)

SC-05
Review the duplicate bean declaration and change the configuration.

Class Conflicts

There may be potential conflicts of class when it is defined in several locations at the same time in a Commerce project.

No class conflicts found in the Neo code.

Overview

SAP Commerce Solution ships with a default-data model. However, the product provides mechanisms to extend and adapt the default data-model to customer requirements. Data model adaptions made for the project were reviewed with the Type System Validator.

Types

The SAP Commerce Solution uses a system of types to organize data, for example product information, customer data, addresses, or orders. This “system of types” is known as application’s type-system or data model. This is defined through **items.xml** files and is used to generate code, create database tables, relationship mapping, and data access with SQL. It is imperative that the type-system is defined according to best practices, as it defines the basic building blocks of the application. Inefficiencies within the type-system definition can permeate through the entire application.

Type System Validation

SAP Commerce Solution uses a proprietary tool called the TSV (Type System Validator) to ensure items.xml type and attribute definitions are in line with SAP Commerce Solution best practices. This tool was run over all of NEO - IAM Consulting’s items.xml files. The output was then analyzed to produce a list of the highest priority items that required changes. The full output will be provided with SAP Commerce Solution's final Review Document.

Findings of Type System Validation Rules

Deployment code must be greater than ten thousand

Type codes 1 to 10000 are reserved for platform use. Also note that, 100XX are reserved for b2bCommerce extension. We suggest not to use type codes of 100XX codes to avoid any clashes with future migrations.

Extension	Item Type	Current Type code
codengaicore-items.xml	Province	10000
codengaicore-items.xml	District	10001
codengaicore-items.xm	SubDistrict	10002
codengaicore-items.xm	Occupation	10003
codengaicore-items.xm	PhoneCountryCode	10004
codengaicore-items.xm	MessageTemplate	10005
codengaicore-items.xm	BusinessPartner	10006
codengaicore-items.xm	BPProvinceMap	10007
codengaicore-items.xm	BrandCategory	10009
codengaicore-items.xm	EMSBox	10010

codengaicore-items.xm	NeoSpecialDeliveryMode	10011
codengaicore-items.xm	NeoInterDeliveryMode	10012

DM-01
Change the type codes to 110XX series (with unused ones) for the above mentioned types to avoid any future migration issues

No Deployment Table Should Exist For Item If Not Extending Generic Item

Specifying deployment tables on sub--- types where the super type has a deployment table defined can result in large and complex queries, which can impact performance. Searching a super type will result in a SQL Union queries for each of the sub types.

Extension	Item Type
codengaicore-items.xml	NeoSpecialDeliveryMode
codengaicore-items.xml	NeoInterDeliveryMode
codengaicore-items.xml	NeoIconMedia
codengaicore-items.xml	NeoReportOutOfStock
codengaicore-items.xml	NeoReportSaleSummary
codengaicore-items.xml	NeoReportSaleTransaction
codengaicore-items.xml	SaleSummaryScheduler
codengaicore-items.xml	FailDeliveryConsignmentHistory
codengaicore-items.xml	NeoOrderAutomaticallyCancelled
codengaicore-items.xml	NeoSponsoredProductReport
codengaicore-items.xml	NeoPromotionRevenueReport
codengaicore-items.xml	PromotionRevenueScheduler
codengaicore-items.xml	NeoImageMedia
codengaicore-items.xml	NeoBannerMedia
codengaicore-items.xml	NeoCmsLink

DM-02
A deployment table should not be defined for any item extending types that are not GenericItem. Consider removing the deployment table for the indicated items.

Jalo Class Is Not Allowed When Adding Fields To Existing Class

The jaloclass attribute is not allowed when autocreate='false' and generate='false'.

Extension	Item Type
codengaicore-items.xml	StandardPaymentMode

DM-03
Consider reviewing the Jaloclass attribute when autocreate='false' and generate='false'.

Mandatory Field Must Have Initial Value

Mandatory fields (where optional='false') must either have initial set to 'true' or a default value defined. Mandatory fields for a type —that is, defined with optional='false'— should be declared initial or a default value should be provided. The following attributes are declared mandatory, but neither are initial nor have a default value.
The following boolean fields must be mandatory (having optional='false'):

Extension	Attribute
codengaicore-items.xml	BPProvinceMap.businessPartner
codengaicore-items.xml	OrderModiRecConsign.status
codengaicore-items.xml	NeoReportOutOfStock.title

codengaicore-items.xml	AbstractSaleSummary.title
codengaicore-items.xml	AbstractSaleSummary.dateFrom
codengaicore-items.xml	AbstractSaleSummary.dateTo
codengaicore-items.xml	NeoReportSaleTransaction.title
codengaicore-items.xml	NeoReportSaleTransaction.dateFrom
codengaicore-items.xml	NeoReportSaleTransaction.dateTo
codengaicore-items.xml	NeoOrderAutomaticallyCancelled.title
codengaicore-items.xml	NeoSponsoredProductReport.title
codengaicore-items.xml	AbstractPromotionRevenueReport.title
codengaicore-items.xml	AbstractPromotionRevenueReport.dateFrom
codengaicore-items.xml	AbstractPromotionRevenueReport.dateTo
codengaicore-items.xml	NeoConsignmentPickupReminderAndExpiry.title

DM-04
Attribute will only be writeable during item creation and as such, consider setting mandatory fields to 'true' or using a default value (where optional='false').

Field Name Must Start With Lowercase Letter

The following Item attribute names should start with a lowercase letter:

Extension	Attribute
codengaikbank-items.xml	Order.SAPTerminalId
codengaikbank-items.xml	Order.SAPCardType
codengaikbank-items.xml	Order.SAPCardNo
codengaikbank-items.xml	Order.SAPApprovalCode
codengaikbank-items.xml	Order.KSKCardType

DM-05
For coding best practices, attribute names should start with an lowercase letter.

Ordering Of Relation Should Be Avoided

Any side of a relation that has cardinality='many' should not have ordered='true' unless absolutely necessary. Ordered relations reduce the system performance, especially during export and catalog synchronization. Therefore, it is a good practice to switch order off whenever it is possible. It is also recommended to define the collection type as "set" whenever possible.

Extension	Attribute
codengaicore-items.xml	ConsignmentHistoryRelation.historyEntries

DM-06
Set relations that have cardinality='many' as not ordered (ordered='false')

Indexes Should Be Defined For The Unique Attributes of Type

There should be a covering index defined that includes all the unique attributes for type. There are Service Layer interceptors that validate uniqueness for each type, which generate a query for all the unique attributes. Failure to add an index can cause serious performance issues. Furthermore, the validation of the unique attributes in the Service Layer cannot guarantee that there won't be duplicate records in the database, only a unique index/constraint can do this.
The following data types should have indexes defined for their unique attributes.

Extension	Item Type
codengai\codengaikbank\resources\codengaikbank-items.xml	Order
codengai\codengaicore\resources\codengaicore-items.xml	Customer
codengai\codengaicore\resources\codengaicore-items.xml	OrderEntry

codengai\codengaicore\resources\codengaicore-items.xml	DeliveryMode
codengai\codengaicore\resources\codengaicore-items.xml	NeoNormalDeliveryMode
codengai\codengaicore\resources\codengaicore-items.xml	EMSBox
codengai\codengaicore\resources\codengaicore-items.xml	NeoZoneDeliveryMode
codengai\codengaicore\resources\codengaicore-items.xml	NeoSpecialDeliveryMode
codengai\codengaicore\resources\codengaicore-items.xml	NeoInterDeliveryMode
codengai\codengaicore\resources\codengaicore-items.xml	AbstractCMSLinksComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoFooterLinkComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoFooterSocialMediaComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoFooterImagesComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoIconMedia
codengai\codengaicore\resources\codengaicore-items.xml	NeoHeaderCustomerCareComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHeaderCurrentLocationComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHeaderLanguageSelectorComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHeaderHelperLinkComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHeaderCustomerLinkComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHomepageHeaderNavLinkComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHeaderQuickLinkComponent
codengai\codengaicore\resources\codengaicore-items.xml	Order
codengai\codengaicore\resources\codengaicore-items.xml	StandardPaymentMode
codengai\codengaicore\resources\codengaicore-items.xml	NeoCODPaymentMode
codengai\codengaicore\resources\codengaicore-items.xml	NeoBankTransferInfo
codengai\codengaicore\resources\codengaicore-items.xml	BaseStore
codengai\codengaicore\resources\codengaicore-items.xml	OrderModiRecConsign
codengai\codengaicore\resources\codengaicore-items.xml	OrderCancelRecConsignment
codengai\codengaicore\resources\codengaicore-items.xml	OrderReturnRecConsignment
codengai\codengaicore\resources\codengaicore-items.xml	UpdatedPasswordProcess
codengai\codengaicore\resources\codengaicore-items.xml	NeoEnquiry
codengai\codengaicore\resources\codengaicore-items.xml	NeoCategoryTicket
codengai\codengaicore\resources\codengaicore-items.xml	NeoSubCategoryTicket
codengai\codengaicore\resources\codengaicore-items.xml	CsTicketEventEmailConfiguration
codengai\codengaicore\resources\codengaicore-items.xml	NeoReportOutOfStock
codengai\codengaicore\resources\codengaicore-items.xml	AbstractSaleSummary
codengai\codengaicore\resources\codengaicore-items.xml	NeoReportSaleSummary
codengai\codengaicore\resources\codengaicore-items.xml	NeoReportSaleTransaction
codengai\codengaicore\resources\codengaicore-items.xml	SaleSummaryScheduler
codengai\codengaicore\resources\codengaicore-items.xml	NeoOrderAutomaticallyCancelled
codengai\codengaicore\resources\codengaicore-items.xml	NeoSponsoredProductReport
codengai\codengaicore\resources\codengaicore-items.xml	VariantProduct
codengai\codengaicore\resources\codengaicore-items.xml	AbstractPromotionRevenueReport
codengai\codengaicore\resources\codengaicore-items.xml	NeoPromotionRevenueReport
codengai\codengaicore\resources\codengaicore-items.xml	PromotionRevenueScheduler
codengai\codengaicore\resources\codengaicore-items.xml	NeoConsignmentPickupReminderAndExpiry
codengai\codengaicore\resources\codengaicore-items.xml	NeoCollectionBannerComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoImageMedia
codengai\codengaicore\resources\codengaicore-items.xml	NeoAccountNavigationComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoBannerMedia
codengai\codengaicore\resources\codengaicore-items.xml	NeoDealsBannerComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoPopularBannerComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoProductCarouselComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHealthBeatyBannerComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoHomeCareBannerComponent

codengai\codengaicore\resources\codengaicore-items.xml	NeoBrandBannerComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoSimpleBannerComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoBrandBannerPlpComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoSimpleBannerPlpComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoCmsLink
codengai\codengaicore\resources\codengaicore-items.xml	CMSNavigationNode
codengai\codengaicore\resources\codengaicore-items.xml	NeoCategoriesNavigationComponent
codengai\codengaicore\resources\codengaicore-items.xml	CartPageProductReferencesComponent
codengai\codengaicore\resources\codengaicore-items.xml	CartPageSponsoredProductReferencesComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoBestSellingProductsComponent
codengai\codengaicore\resources\codengaicore-items.xml	BusinessPartnerPage
codengai\codengaicore\resources\codengaicore-items.xml	NeoRecentlyViewedProductsCarouselComponent
codengai\codengaicore\resources\codengaicore-items.xml	NeoOrderModificationProcess
codengai\codengaicore\resources\codengaicore-items.xml	NeoConsignmentModificationProcess
codengai\codengaicore\resources\codengaicore-items.xml	EmailPage

DM-07
Consider adding indexes covering all unique attributes of the noted Item types

No Unique Attributes Defined For Type

Every type should have a unique identifier(s) defined or should inherit the unique attributes from a super type.
This allows the data to be easily imported/exported into different systems and prevents duplicate entries in the database.
The following data types should have unique attributes defined.

Extension	Item Type
codengai\codengaikbank\resources\codengaikbank-items.xml	KasikornPaymentTransactionEntry
codengai\codengaikbank\resources\codengaikbank-items.xml	KasikornPaymentTransaction
codengai\codengaikbank\resources\codengaikbank-items.xml	KasikornPaymentInfo
codengai\codengaicore\resources\codengaicore-items.xml	PriceRow
codengai\codengaicore\resources\codengaicore-items.xml	Consignment
codengai\codengaicore\resources\codengaicore-items.xml	ConsignmentEntry
codengai\codengaicore\resources\codengaicore-items.xml	Province
codengai\codengaicore\resources\codengaicore-items.xml	District
codengai\codengaicore\resources\codengaicore-items.xml	SubDistrict
codengai\codengaicore\resources\codengaicore-items.xml	Address
codengai\codengaicore\resources\codengaicore-items.xml	Occupation
codengai\codengaicore\resources\codengaicore-items.xml	PhoneCountryCode
codengai\codengaicore\resources\codengaicore-items.xml	MessageTemplate
codengai\codengaicore\resources\codengaicore-items.xml	StockLevel
codengai\codengaicore\resources\codengaicore-items.xml	StockLevelHistory
codengai\codengaicore\resources\codengaicore-items.xml	CreditCardPaymentInfo
codengai\codengaicore\resources\codengaicore-items.xml	GUISequenceNumberRange
codengai\codengaicore\resources\codengaicore-items.xml	NeoGlobalConfig
codengai\codengaicore\resources\codengaicore-items.xml	OrderModiRecConsignEntry
codengai\codengaicore\resources\codengaicore-items.xml	OrderCancelRecConsignEntry
codengai\codengaicore\resources\codengaicore-items.xml	OrderReturnRecConsignEntry
codengai\codengaicore\resources\codengaicore-items.xml	ConsignmentHistory
codengai\codengaicore\resources\codengaicore-items.xml	FailDeliveryConsignmentHistory
codengai\codengaicore\resources\codengaicore-items.xml	Warehouse
codengai\codengaicore\resources\codengaicore-items.xml	SponsoredProductHistory

DM-08
Consider whether noted types require unique identifiers

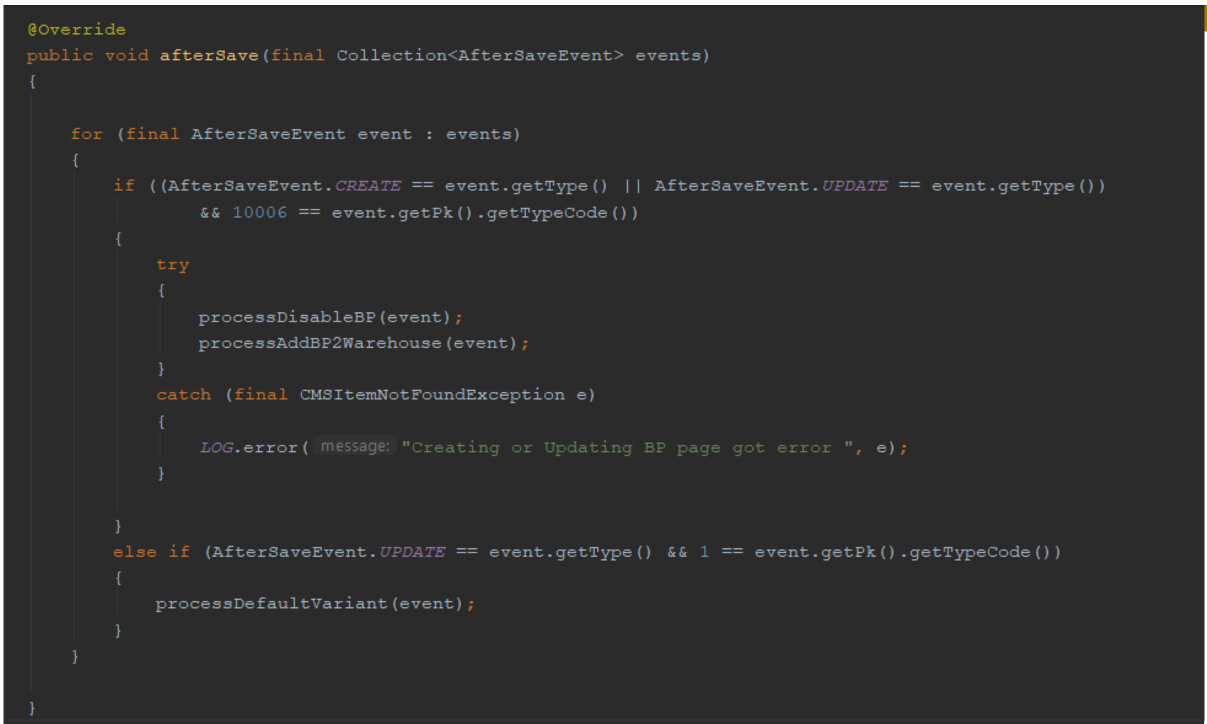
Overview

This section contains a manual analysis of the custom code, with more focus on the changes done in Sprint 6, 7. Additionally the review also focus on the general Hybris coding standards, the spring files, cache configuration.

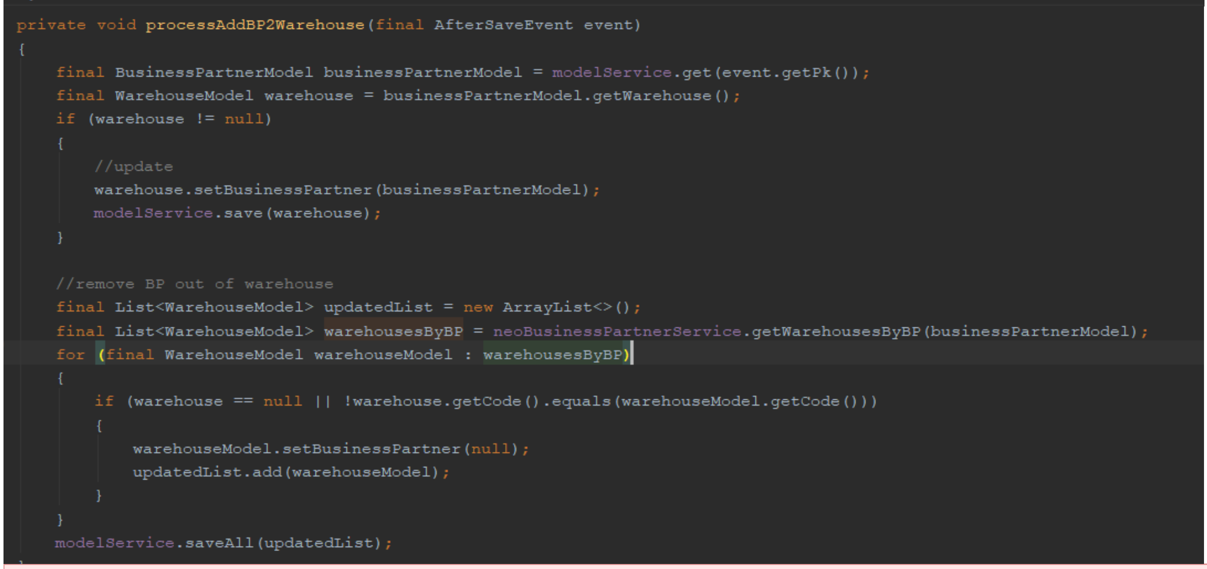
Performance aspects code review

Heavy After Save listeners

After Save Listeners are single threaded and will queue events until the queue is full, at which point it will block all threads from saving anything
In NEO codebase, There are multiple listeners (45 of them in codengaicore\src\com\neo\codengai\core\event package) listening for these events.
One of the example is, *com.neo.codengai.core.event.aftersave.listener.NeoAfterSaveListener* where the type codes hard coded and DB opeations, create new BP page etc.



```
@Override
public void afterSave(final Collection<AfterSaveEvent> events)
{

    for (final AfterSaveEvent event : events)
    {
        if ((AfterSaveEvent.CREATE == event.getType() || AfterSaveEvent.UPDATE == event.getType())
                && 10006 == event.getPk().getTypeCode())
        {
            try
            {
                processDisableBP(event);
                processAddBP2Warehouse(event);
            }
            catch (final CMSItemNotFoundException e)
            {
                LOG.error( message: "Creating or Updating BP page got error ", e);
            }

        }
        else if (AfterSaveEvent.UPDATE == event.getType() && 1 == event.getPk().getTypeCode())
        {
            processDefaultVariant(event);
        }
    }

}
```



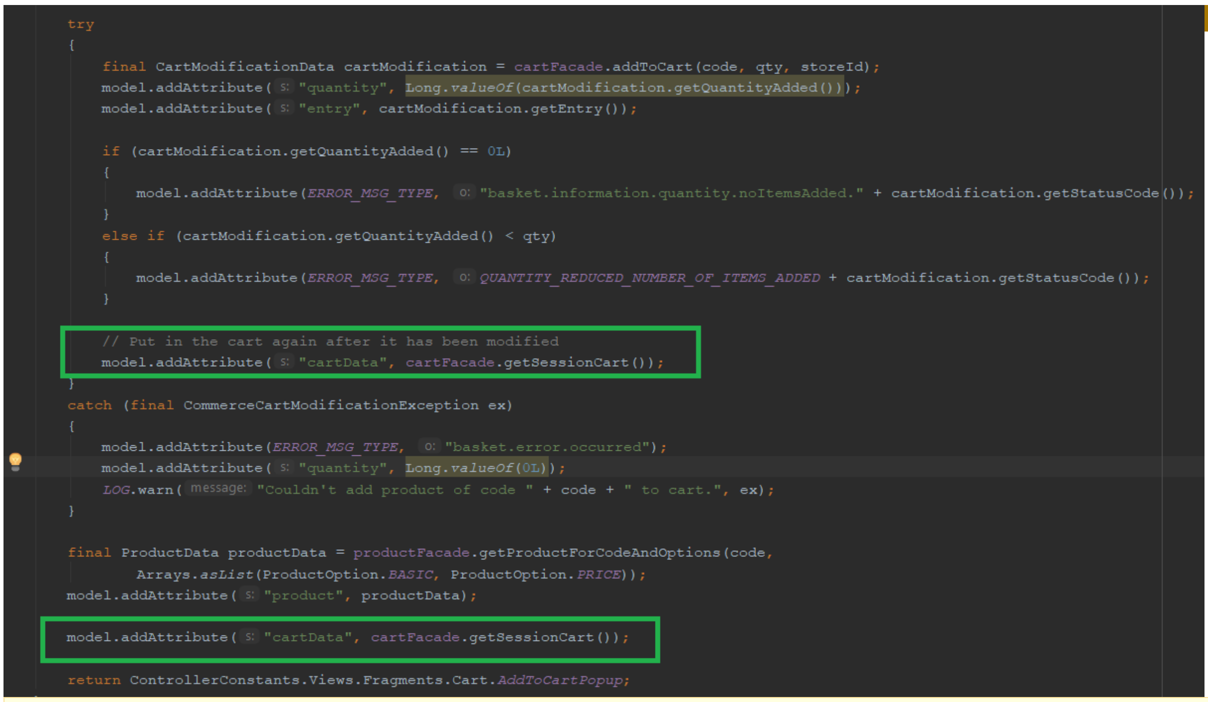
```
private void processAddBP2Warehouse(final AfterSaveEvent event)
{
    final BusinessPartnerModel businessPartnerModel = modelService.get(event.getPk());
    final WarehouseModel warehouse = businessPartnerModel.getWarehouse();
    if (warehouse != null)
    {
        //update
        warehouse.setBusinessPartner(businessPartnerModel);
        modelService.save(warehouse);
    }

    //remove BP out of warehouse
    final List<WarehouseModel> updatedList = new ArrayList<>();
    final List<WarehouseModel> warehousesByBP = neoBusinessPartnerService.getWarehousesByBP(businessPartnerModel);
    for (final WarehouseModel warehouseModel : warehousesByBP)
    {
        if (warehouse == null || !warehouse.getCode().equals(warehouseModel.getCode()))
        {
            warehouseModel.setBusinessPartner(null);
            updatedList.add(warehouseModel);
        }
    }
    modelService.saveAll(updatedList);
```

VC-01
Avoid saving Models or other slow operations in an After Save Listener.
Review all the After save listeners in order to

1. Move the business logic from After Save listeners to the corresponding business service layer.
2. Remove hard coding of the type codes

Cart observations

Aviod calling *cartFacade.getSessionCart()* unnecessarily to avoid the cart service,converter behind the scenes or creating empty carts.
For example, in com.neo.codengai.storefront.controllers.pages.PickupInStoreController#addToCartPickup, we can see getSessionCart() called two times. One of this call can be avoided. We suggest to review other parts of the code having same/similar issues.

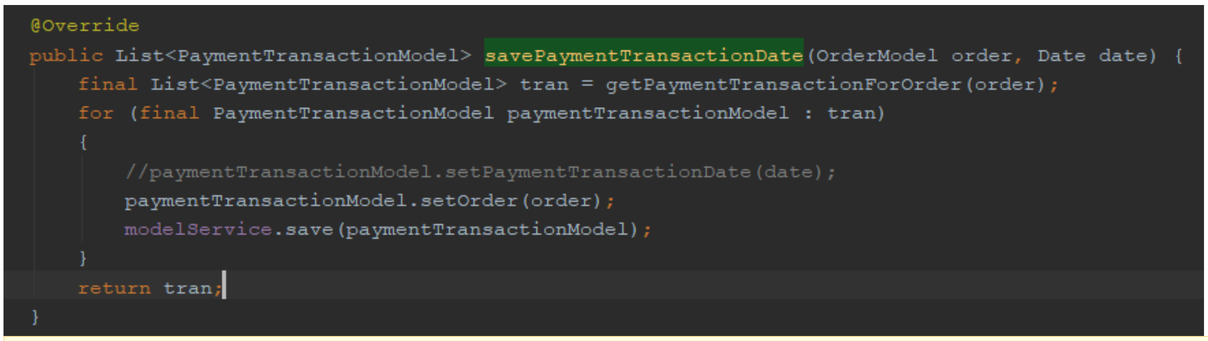


```
try
{
    final CartModificationData cartModification = cartFacade.addToCart(code, qty, storeId);
    model.addAttribute( s: "quantity", Long.valueOf(cartModification.getQuantityAdded()));
    model.addAttribute( s: "entry", cartModification.getEntry());

    if (cartModification.getQuantityAdded() == 0L)
    {
        model.addAttribute(ERROR_MSG_TYPE,  o: "basket.information.quantity.noItemsAdded." + cartModification.getStatusCode());
    }
    else if (cartModification.getQuantityAdded() < qty)
    {
        model.addAttribute(ERROR_MSG_TYPE,  o: QUANTITY_REDUCED_NUMBER_OF_ITEMS_ADDED + cartModification.getStatusCode());
    }

    // Put in the cart again after it has been modified
    model.addAttribute( s: "cartData", cartFacade.getSessionCart());
}
catch (final CommerceCartModificationException ex)
{
    model.addAttribute(ERROR_MSG_TYPE,  o: "basket.error.occurred");
    model.addAttribute( s: "quantity", Long.valueOf(0L));
    LOG.warn( message: "Couldn't add product of code " + code + " to cart.", ex);
}

final ProductData productData = productFacade.getProductForCodeAndOptions(code,
        Arrays.asList(ProductOption.BASIC, ProductOption.PRICE));
model.addAttribute( s: "product", productData);

model.addAttribute( s: "cartData", cartFacade.getSessionCart());

return ControllerConstants.Views.Fragments.Cart.AddToCartPopup;
```

VC-02
Avoid redundant cartFacade.getSessionCart() calls in the code.

Inefficient DB operations

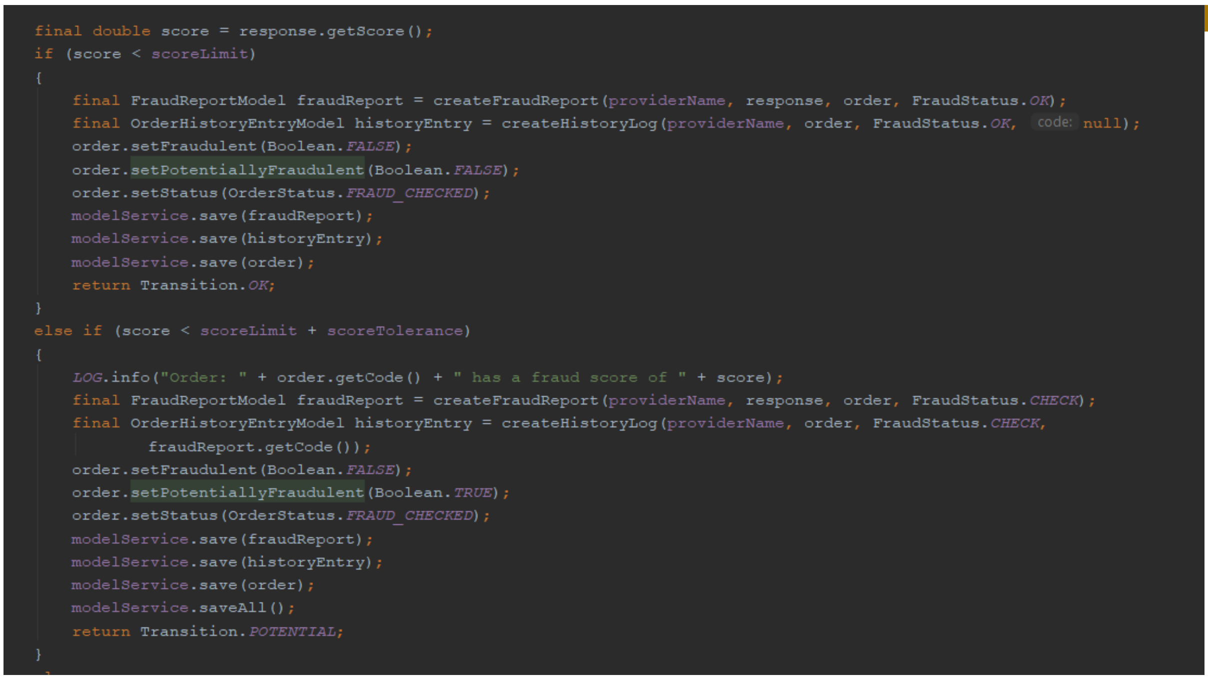
Avoid calling saving data models in loops to reduce the impact on the database calls. There are few occurrences of such issue noticed and couple of them are mentioned below.
com.neo.codengai.order.impl.LinePayOrderServiceImpl#savePaymentTransactionDate
com.neo.codengai.actions.returns.ApproveReturnAction#executeAction
com.neo.codengai.core.ordersplitting.impl.NeoOrderSplittingService#splitOrderForConsignment
Please check across the codebase for this kind of issues.



```
@Override
public List<PaymentTransactionModel> savePaymentTransactionDate(OrderModel order, Date date) {
    final List<PaymentTransactionModel> tran = getPaymentTransactionForOrder(order);
    for (final PaymentTransactionModel paymentTransactionModel : tran)
    {
        //paymentTransactionModel.setPaymentTransactionDate(date);
        paymentTransactionModel.setOrder(order);
        modelService.save(paymentTransactionModel);
    }
    return tran;
}
```

VC-03
Avoid calling modelservice.save() in loops, instead use saveAll() after the loop wherever possible

In general, avoid using modelService.saveAll() to save a list of Model, use the correct syntax saveAll(Collection<? extends Object> models). saveAll saves all modified and new model instances which are attached to the current request context.
Example, com.neo.codengai.actions.order.fraudcheck.FraudCheckOrderAction#executeAction
com.neo.codengai.integration.util.SourcingUtil#runDefaultOrderProcessForOrder
The number of DB calls also can be reduced by collecting the models which have to be saved in a Collection and the save the entire collection with using the saveAll method of the ModelService.
Example, com.neo.codengai.actions.order.fraudcheck.FraudCheckOrderAction#executeAction



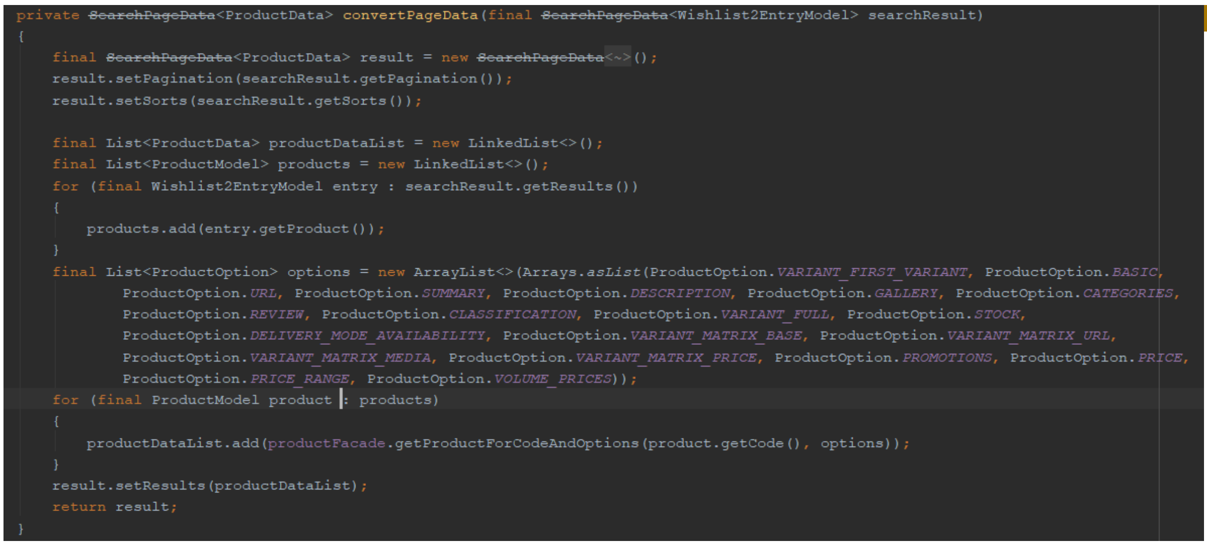
```java
final double score = response.getScore();
if (score < scoreLimit)
{
    final FraudReportModel fraudReport = createFraudReport(providerName, response, order, FraudStatus.OK);
    final OrderHistoryEntryModel historyEntry = createHistoryLog(providerName, order, FraudStatus.OK,  code: null);
    order.setFraudulent(Boolean.FALSE);
    order.setPotentiallyFraudulent(Boolean.FALSE);
    order.setStatus(OrderStatus.FRAUD_CHECKED);
    modelService.save(fraudReport);
    modelService.save(historyEntry);
    modelService.save(order);
    return Transition.OK;
}
else if (score < scoreLimit + scoreTolerance)
{
    LOG.info("Order: " + order.getCode() + " has a fraud score of " + score);
    final FraudReportModel fraudReport = createFraudReport(providerName, response, order, FraudStatus.CHECK);
    final OrderHistoryEntryModel historyEntry = createHistoryLog(providerName, order, FraudStatus.CHECK,
            fraudReport.getCode());
    order.setFraudulent(Boolean.FALSE);
    order.setPotentiallyFraudulent(Boolean.TRUE);
    order.setStatus(OrderStatus.FRAUD_CHECKED);
    modelService.save(fraudReport);
    modelService.save(historyEntry);
    modelService.save(order);
    modelService.saveAll();
    return Transition.POTENTIAL;
}
```

VC-04
Avoid using modelService saveAll() , with out parameters

VC-05
Use modelservice.saveAll(Collection) instead of multiple save() calls in the same method or block, wherever possible

Product options usage

Wherever possible, try to use the product data with minimum required set of options. With more production options, the corresponding populators (and code behind that) will be executed to populate the required data which will have impact on the performance.
For example, in the *com.neo.codengai.storefront.controllers.pages.AccountWishlistPageController#wishListDetail* flow, the below method (*com.neo.codengai.facades.user.impl.NeoCustomerFacadeImpl#convertPageData*) is called where we can see many product options are used.
Also here, there are two loops (one for searchResults and another for products), which can be combined into one.



```
private ~~SearchPageData~~<ProductData> convertPageData(final ~~SearchPageData~~<Wishlist2EntryModel> searchResult)
{
    final ~~SearchPageData~~<ProductData> result = new ~~SearchPageData~~<~>();
    result.setPagination(searchResult.getPagination());
    result.setSorts(searchResult.getSorts());

    final List<ProductData> productDataList = new LinkedList<>();
    final List<ProductModel> products = new LinkedList<>();
    for (final Wishlist2EntryModel entry : searchResult.getResults())
    {
        products.add(entry.getProduct());
    }
    final List<ProductOption> options = new ArrayList<>(Arrays.asList(ProductOption.VARIANT_FIRST_VARIANT, ProductOption.BASIC,
            ProductOption.URL, ProductOption.SUMMARY, ProductOption.DESCRIPTION, ProductOption.GALLERY, ProductOption.CATEGORIES,
            ProductOption.REVIEW, ProductOption.CLASSIFICATION, ProductOption.VARIANT_FULL, ProductOption.STOCK,
            ProductOption.DELIVERY_MODE_AVAILABILITY, ProductOption.VARIANT_MATRIX_BASE, ProductOption.VARIANT_MATRIX_URL,
            ProductOption.VARIANT_MATRIX_MEDIA, ProductOption.VARIANT_MATRIX_PRICE, ProductOption.PROMOTIONS, ProductOption.PRICE,
            ProductOption.PRICE_RANGE, ProductOption.VOLUME_PRICES));
    for (final ProductModel product : products)
    {
        productDataList.add(productFacade.getProductForCodeAndOptions(product.getCode(), options));
    }
    result.setResults(productDataList);
    return result;
}
```

Review the possibility to reduce the options in these kind of cases across the code base.

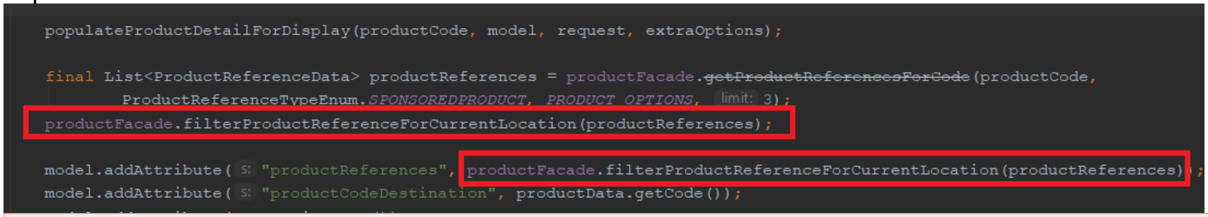
VC-06

Review all the usages of product options and reduce the number of options wherever possible to have less impact on the performance. Some of them are mentioned below, but there can be more places to review

- com.neo.codengai.storefront.controllers.pages.ProductPageController#showQuickView
- com.neo.codengai.storefront.controllers.pages.AccountWishlistPageController#removeWishlistItemPopup
- com.neo.codengai.facades.user.impl.NeoCustomerFacadeImpl#convertPageData
- com.neo.codengai.facades.populators.NeoWishlist2EntryPopulator#populate
- com.neo.codengai.storefront.controllers.pages.ProductPageController#populateProductDetailForDisplay
  The above method referenced in places like ProductPageController#writeReview(), ProductPageController#postReview() where we doesn't seem to require these many options while it makes sense to use many options for ProductPageController#productDetail(). So the best option is to write a separate method at different places with suitable product options.

Product page API with Redundant call

In com.neo.codengai.storefront.controllers.pages.ProductPageController#productDetail() method, there is an extra call to the productFacade.filterProductReferenceForCurrentLocation() which has heavy logic behind in a loop to get products for the location. This seems to be a copy/paste error but with significant performance impact.

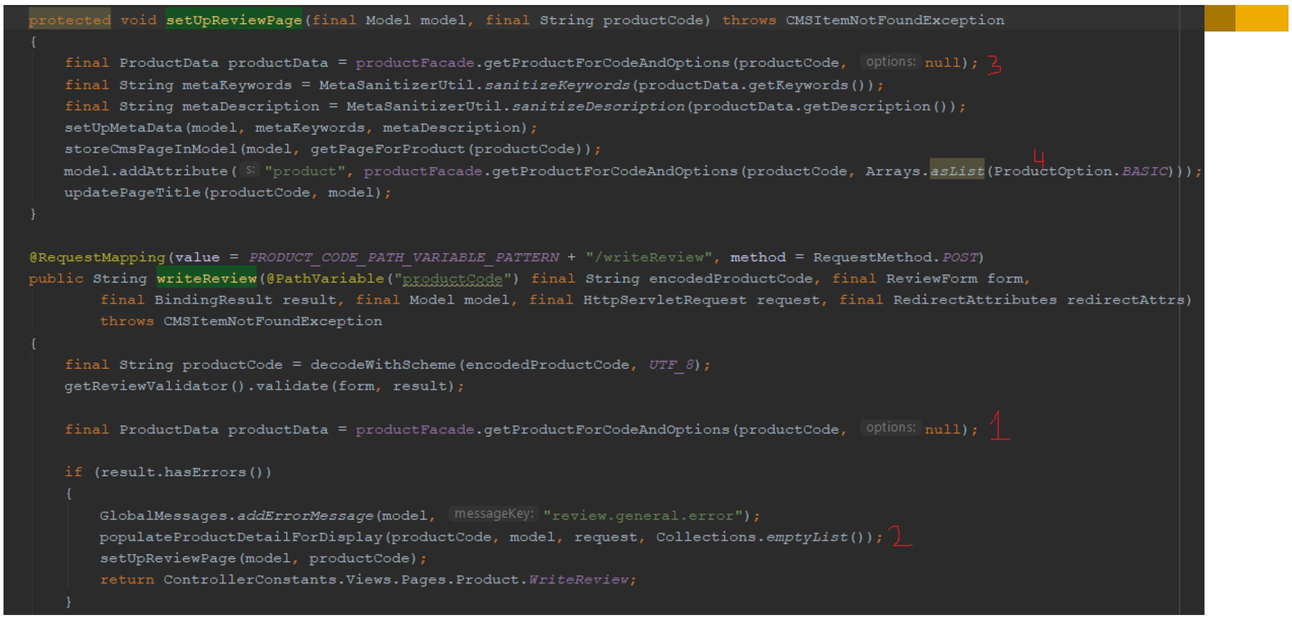


```
    populateProductDetailForDisplay(productCode, model, request, extraOptions);

    final List<ProductReferenceData> productReferences = productFacade.~~getProductReferencesForCode~~(productCode,
            ProductReferenceTypeEnum.SPONSOREDPRODUCT, PRODUCT_OPTIONS, limit: 3);
    productFacade.filterProductReferenceForCurrentLocation(productReferences);

    model.addAttribute( s: "productReferences", productFacade.filterProductReferenceForCurrentLocation(productReferences));
    model.addAttribute( s: "productCodeDestination", productData.getCode());
```

VC-07

Remove the extra call to the productFacade.filterProductReferenceForCurrentLocation() in the productDetails flow.

In the com.neo.codengai.storefront.controllers.pages.ProductPageController#writeReview() flow,we can see productFacade.getProductForCodeAndOptions() is called four times which can be ideally reduced to one and save the cost of calling converters/populators to get the product data.



```
protected void setUpReviewPage(final Model model, final String productCode) throws CMSItemNotFoundException
{
    final ProductData productData = productFacade.getProductForCodeAndOptions(productCode, options: null); 3
    final String metaKeywords = MetaSanitizerUtil.sanitizeKeywords(productData.getKeywords());
    final String metaDescription = MetaSanitizerUtil.sanitizeDescription(productData.getDescription());
    setUpMetaData(model, metaKeywords, metaDescription);
    storeCmsPageInModel(model, getPageForProduct(productCode));
    model.addAttribute( s: "product", productFacade.getProductForCodeAndOptions(productCode, Arrays.asList(ProductOption.BASIC))); 4
    updatePageTitle(productCode, model);
}

@RequestMapping(value = PRODUCT_CODE_PATH_VARIABLE_PATTERN + "/writeReview", method = RequestMethod.POST)
public String writeReview(@PathVariable("productCode") final String encodedProductCode, final ReviewForm form,
        final BindingResult result, final Model model, final HttpServletRequest request, final RedirectAttributes redirectAttrs)
        throws CMSItemNotFoundException
{
    final String productCode = decodeWithScheme(encodedProductCode, UTF_8);
    getReviewValidator().validate(form, result);

    final ProductData productData = productFacade.getProductForCodeAndOptions(productCode, options: null); 1

    if (result.hasErrors())
    {
        GlobalMessages.addErrorMessage(model, messageKey: "review.general.error");
        populateProductDetailForDisplay(productCode, model, request, Collections.emptyList()); 2
        setUpReviewPage(model, productCode);
        return ControllerConstants.Views.Pages.Product.WriteReview;
    }
```

VC-08
Optimize productFacade.getProductForCodeAndOptions() with required options only and call once rather than doing multiple calls in the method (check the calling methods also). Review if any other instances of this kind of code.

Avoid redundant cartFacade.getSessionCart() calls in the code.

General Inspection

Web Modules

Some extensions don't need web module but they have it declared in extensioninfo.xml which makes them accessible online and raises security concerns. For example:

```
<webmodule jspcompile="false" webroot="/samplecore"/>
<webmodule jspcompile="false" webroot="/sampleinboundintegration"/>
```

Only extensions that need to be accessed from URL have a web module declared in their extensioninfo.xml definition file.

Abortable Jobs

Custom Jobs are not abortable by default, which means it won't be possible to stop them if they are running for too long. To enable this feature, override the isAbortable() method in any customised MyJobPerformable class:

```
@Override
public boolean isAbortable() {
  return true;
}
```

Alternatively, while extending the AbstractJobPerformable bean, you may override its abortable property value in custom spring configuration e.g.

```
<bean id="myJobPerformable"
      class="de.hybris.cronjobtutorial.MyJobPerformable"
      parent="abstractJobPerformable">
  <property name="l10nService" ref="l10nService" />
  <property name="abortable" value="true"/>
</bean>
```

Currently the job is marked as abortable, which allows to activate the abort flag for the cron job, similarly to the **interrupt** flag for **Thread**. When implementing an abortable job it is important to regularly check if the user has requested to abort the job. If true, the abort flag is set, the job should be stopped, and the cron job should have a proper status and result. You should decide when to check the flag and what to do if the flag is set, for example, to clean up. Time of processing between verifications should be moderate.
The following code sample calls the **clearAbortRequestedIfNeeded** of the **AbstractJobPerformable**. It checks if the given **cronJob** is requested to be aborted and has **REQUESTABORT** flag set to **true**. If so, the flag is set to **false** again and the job execution is prematurely stopped.

```
if(clearAbortRequestedIfNeeded(cronJob))
{
  //abort the job
  //do some clean-up
  return new PerformResult(CronJobResult.ERROR, CronJobStatus.ABORTED);
}
```

All the jobs in package *com.neo.codengai.core.job* are not implementing abortable feature.

VC-09
Make sure that all custom jobs in ***com.neo.codengai.core.job*** package are abortable.

Accelerator/Commerce Extensions

Spring configuration of main accelerator/commerce extensions should never be directly modified. For that, SAP Commerce Solution provide aliases to internal Spring beans, which can be used in custom Spring configuration files to override default behavior.

Core Spring configuration files have not been directly modified. Instead, aliases were used to customize Beans.

Usage of Logging

Configuration

It is very useful to write messages to different files depending on the log4j-logger name to isolate specific messages and speed-up the processing of log files.

```
log4j.appender.AuditLogFile=org.apache.log4j.DailyRollingFileAppender
log4j.logger.AUDIT=info, AuditLogFile
log4j.appender.CustomerLogFile=org.apache.log4j.DailyRollingFileAppender
log4j.logger.com.yrnet.easy.core.log=info, CustomerLogFile
log4j.appender.WsLogFile=org.apache.log4j.DailyRollingFileAppender
log4j.logger.com.yrnet.easy.core.service.ws.logging=info, WsLogFile
```

It is very useful to write messages to different files depending on the log4j-logger name to isolate specific messages and speed up the processing of log files.

3) The classes as follows use System.out.println:

Extension	Class	Matches
codengailinepays	com.neo.codengai.order.LinePayOrderService.LinePayCheckoutFacadeImpl	1
codengailinepays	com.neo.codengai.order.impl.LinePayOrderServiceImpl	1
codengailinepays	com.neo.codengai.order.impl.LinePayOrderFacadeImpl	4

VC-10
Avoid using System.out.println to log messages. It could be a bottleneck for performance and should be replaced by log4j.

Controllers

Please refer the attached file controllers-using-models.txt having all the references where data model objects are used in the controllers.

Please refer the attached file controllers-using-services.txt having all the service class references used in the controllers.

VC-11
Avoid controller driven development. Controllers should not be aware of Models, Business Services, DAOs and HTML.

Facades

Please refer the attached file facades-instantiate-models.txt having facades initializing the models

Please refer the attached file facades-populate-objects.txt having facades populating the data rather than using converters.

VC-12
Avoid facade-layer conversion from a DTO to a Model. Use a converter instead, with appropriate populators.

Do not write database code (using flexible search) in the service classes, move the corresponding code to DAOs.
There is one class *com/neo/codengai/card/impl/LinePayCardDaoServiceImpl.java* using flexible search but this class is not used anywhere, we suggested to clean up the code if not required

VC-13
Review the service classes with Flexible search code and move the logic to DAOs
There is one unused service class using flexible search class, which can be cleaned up. Other than this, no service-layer-flexible-search query has been identified.

DAOs

The code base has no instances of hard-coded-flexible-search queries.

Populators and Converters

Populators and converters within SAP Commerce Solution can be critical for performance considerations. One common mistake when working with converters and populators is to confuse them.Converters should use populators and not vice versa. Wiring in converters into populators goes against their fine-grained nature and can lead to a long conversion hierarchy of unnecessary object creation and garbage collection.

Populator	converters used
NeoProductPopulator	businessPartnerDataConverter
NeoConsignmentEntryPopulator	orderEntryConverter
NeoBusinessPartnerPopulator	provinceConverter neoDistrictConverter neoSubDistrictConverter
NeoCartPopulator	neoPaymentModeConverter neoAddressConverter
NeoSearchResultProductPopulator	imageConverter
NeoExtendedConsignmentPopulator	deliveryModeConverter pointOfServiceConverter
NeoWishlist2Populator	neoWishlist2EntryConverter
NeoExtendedCustomerPopulator	ticketConverter provinceConverter neoCountryConverter
NeoWishlist2EntryPopulator	productConverter
CustomerProfileDataPopulator	addressConverter creditCardPaymentInfoConverter
NeoOrderPopulator	neoBankTransferInfoConverter
CustomerOverviewDataPopulator	imageConverter addressConverter

NeoConsignmentPopulator	neoZoneDeliveryModeConverter zoneDeliveryModeConverter deliveryModeConverter
SearchPagePointOfServiceDistancePopulator	pointOfServiceDistanceConverter

VC-14
It is important that there are no instances of wiring converters into populator implementations.

Value Providers

Value providers can handle the conversion between SAP SAP Commerce Solution database entries and Solr document values. It's their responsibility to retrieve information in order to de-normalize/complete data for an Indexed Type or its properties. There are multiple value providers already available for the most common types that could be used, and custom ones can be created if needed. Keep in mind that Value Providers are used extensively during indexing operations, and for that, they should be carefully designed and unit tested, in order to avoid issues related to performance and behavior.
The use of value providers, especially custom ones, should be used sparingly as they are used multiple times during indexing operations. Non-performant providers can drastically increase the time needed to complete a full index job. Carefully design queries using Flexible Search and minimize loops. Also, consider adopting strategies to improve performance such as Caching and Database Indexes.
Consider if a custom Value Provider is really required. Some requirements can be achieved by using SpEL Value Providers
Additional guidelines for Value Providers:
1. There should not be multiple DB calls (as DB IO would stack over multiple calls/multiple DBs);
2. There should not be recursive calls (again a Value Provider is called many times for each registered attribute, from each document to be indexed, recursiveness would add to that call stack costing performance);
3. There should not be any unused value provider (they would be called for no good reason, only adding to the processing time of each document).

Value Providers are properly written and no unused Value Provider was found.

Interceptors

The goal of the interceptor should be to ensure data integrity (interceptors are low level), it should not execute complex business logic as it is better to keep the logic in a proper service. Interceptors can have a large influence on the performance on a system if not implemented correctly.
A list of interceptors registered with the Service Layer can be found/investigated at https://localhost:9002/admin/development/interceptors

Interceptors :

- Do not contain any complex business logic.
- Are well-written so that in principle they could not produce an overhead when initializing, saving or removing a model.

VC-15
Review the following interceptors as they have been found to have complex business logic:

- NeoStockLevelValidateInterceptor

- NeoBusinessPartnerValidateInterceptor
- NeoProductValidateInterceptor
- NeoBusinessPartnerPrepareInterceptor
- NeoCustomerValidateInterceptor
- PriceRowTimeRangeValidateInterceptor

Spring

Way to get a spring bean from application context
There are few test classes mentioned below are violating this.

- FraudCheckIntegrationTest
- PaymentIntegrationTest
- ProcessFlowTest

VC-16
Do not search for beans by type or alias.
It is advised to not use Registry.getCoreApplicationContext() because this method always returns the core ApplicationContext. Instead the Registry.getApplicationContext checks first if there is a ServletContext currently holding a WebApplicationContext. If that is the case, this one is returned, which is indeed your web ApplicationContext configured at your web.xml file. If there is no current ServletContext or no ApplicationContext set at it, the global ApplicationContext is returned. With that you do not have to be aware if your context is a Core Module or a Web Module, you always get the correct ApplicationContext.

Be aware that if you do not connect your web ApplicationContext to the global or core ApplicationContext, you cannot access global or core beans via the ApplicationContext returned by Registry.getApplicationContext. If you have built up the connection by simply using the SAP Commerce SolutionContextLoaderListener, you have access to all the beans of your web and the global and core ApplicationContext, unless you have defined overlaying bean definitions in your web ApplicationContext.

```
MyBeanType myBean =
(MyBeanType)Registry.getApplicationContext().getBean("<extname>.mybean");
```

Some custom bean names start with uppercase. The Spring bean naming convention is to use the standard Java convention for instance field names when naming beans. That is, bean names start with a lowercase letter, and are camel-cased from then on. Overrides bean by name is very common in SAP Commerce Solution, therefore naming beans consistently is important.

Custom extension	Spring File	Bean
codengaipays	codengaipays-spring.xml	PreparePaymentTransactionInterceptor
codengaipays	codengaipays-spring.xml	PreparePaymentTransactionEntryInterceptor
codengaipays	codengaipays-spring.xml	PreparePaymentTransactionInterceptorMapping
codengaipays	codengaipays-spring.xml	PreparePaymentTransactionEntryInterceptorMapping
codengaicore	codengaicore-spring.xml	CsTicketValidatorInterceptorMapping

VC-17
Fix all custom bean names to start with lowercase.

Overall, the whole project does not have a consistent way to do dependency injection. @Resource, @Autowired and xml configuration are used.

VC-18
Review the whole project, declare all dependencies in xml configuration file and add @Required annotation to the setters.

Sessions within an application should not store excessively large objects or models. Also it is important to check that all such values stored in the sessions are relevant and not being used by calls outside of the session. This can cause severe degradation of performance caused by memory saturation.

HTTP and Jalo sessions do not seem to contain large objects or Models.

Usage of Services provided by SAP Commerce Solution

Check if de.hybris.platform.servicelayer.config.ConfigurationService is used.
The package de.hybris.platform.util.Config has been imported by the following classes. Review the imports as well as usages in the class and convert them to use the configuration Service.

Custom extension	Class
codengaicockpits	CockpitSystemSetup
codengaistorefront	StaticResourceFilter
codengaistorefront	AnalyticsPropertiesBeforeViewHandler
codengaistorefront	TestIdTag
codengaistorefront	NeoCsrfProtectionMatcherTest

VC-19
Usually we recommend using de.hybris.platform.servicelayer.config.ConfigurationService rather than de.hybris.platform.util.Config.

Check if de.hybris.platform.servicelayer.search.FlexibleSearchService is instead of FlexibileSearch

FlexibleSearchService is used.

Check if de.hybris.platform.servicelayer.user.UserService is used to get the current user or a user by UID.
The JALO layer has been deprecated and should be avoided wherever it is feasible. SAP Commerce Solution recommends using UserService instead. Similarly **UserService** should be used to get User by UID instead of **UserManager.**

codengaistorefront	DefaultAutoLoginStrategy
codengaifacades	NeoCustomerFacadeImpl

VC-20
UserService should be used instead of JaloSession and UserManager.

Cache

There are areas of high utilization where performance will be critical and potential optimizations of these areas of code could be to introduce caches. One area where caches are most utilized is around Converter and Populator classes, so that repeated conversions are not necessary. SAP Commerce Solution also provides a package in this area which adds caching layers into the storefront project for the most commonly seen performance considerations.
Also, for a more detailed cache investigation, please consider a **System Review**. Furthermore, if you want to modify cache to adhere to performance requirements, a **Performance Package** is recommended.
For example, Use CMS cache to cache static CMS component for better performance.

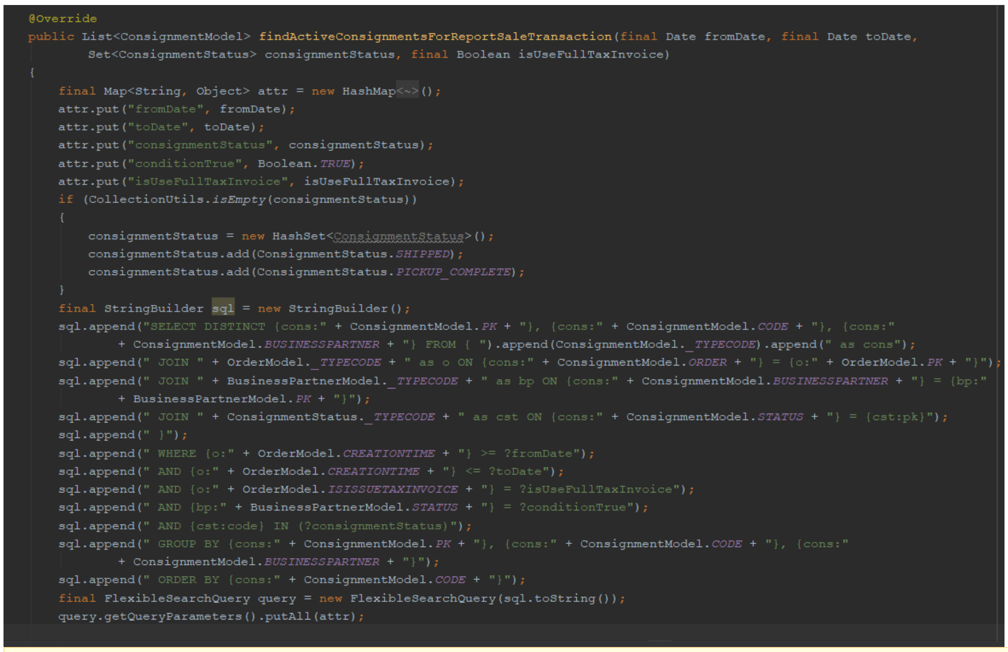
VC-21

Be aware of highly utilized areas in the application which could have a performance impact and introduce measures to counteract this. Use caching in areas such as Populators and Converters to bypass repeated conversion.

Flexible search query usage

Query Strings

We can see in places of the flexible search query strings are defined with lot of append statements in the method, instead declare at class level.
For example, one of the case is shown below *neo.neo.codengai.core.consignment.dao.impl.NeoConsignmentDaoImpl#findActiveConsignmentsForReportSaleTransaction()*
In general, it is good to declare query constants at class level to avoid this code being executed for every method call.

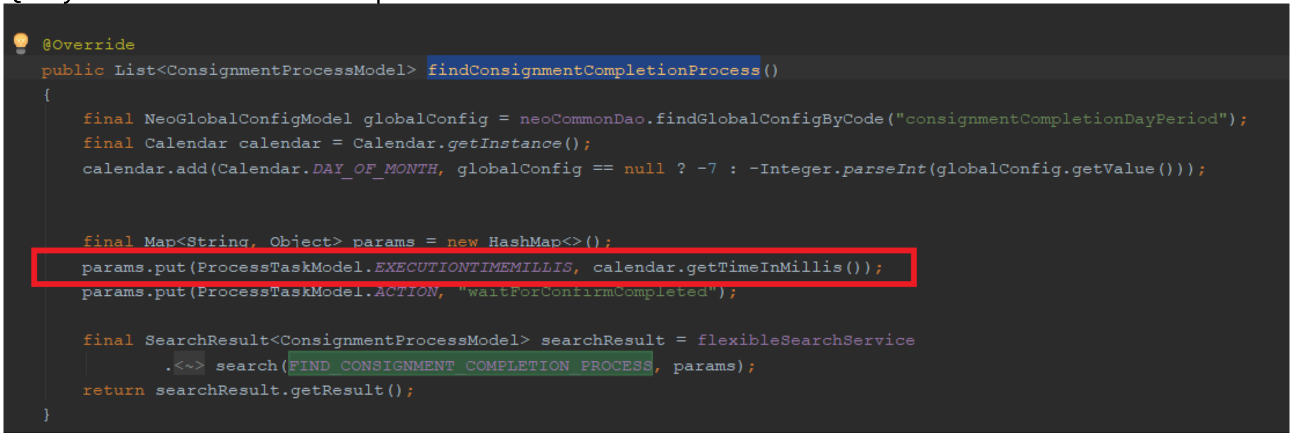


```
@Override
public List<ConsignmentModel> findActiveConsignmentsForReportSaleTransaction(final Date fromDate, final Date toDate,
        Set<ConsignmentStatus> consignmentStatus, final Boolean isUseFullTaxInvoice)
{
    final Map<String, Object> attr = new HashMap<~>();
    attr.put("fromDate", fromDate);
    attr.put("toDate", toDate);
    attr.put("consignmentStatus", consignmentStatus);
    attr.put("conditionTrue", Boolean.TRUE);
    attr.put("isUseFullTaxInvoice", isUseFullTaxInvoice);
    if (CollectionUtils.isEmpty(consignmentStatus))
    {
        consignmentStatus = new HashSet<ConsignmentStatus>();
        consignmentStatus.add(ConsignmentStatus.SHIPPED);
        consignmentStatus.add(ConsignmentStatus.PICKUP_COMPLETE);
    }
    final StringBuilder sql = new StringBuilder();
    sql.append("SELECT DISTINCT {cons:" + ConsignmentModel.PK + "}, {cons:" + ConsignmentModel.CODE + "}, {cons:"
            + ConsignmentModel.BUSINESSPARTNER + "} FROM { ").append(ConsignmentModel._TYPECODE).append(" as cons");
    sql.append(" JOIN " + OrderModel._TYPECODE + " as o ON {cons:" + ConsignmentModel.ORDER + "} = {o:" + OrderModel.PK + "}");
    sql.append(" JOIN " + BusinessPartnerModel._TYPECODE + " as bp ON {cons:" + ConsignmentModel.BUSINESSPARTNER + "} = {bp:"
            + BusinessPartnerModel.PK + "}");
    sql.append(" JOIN " + ConsignmentStatus._TYPECODE + " as cst ON {cons:" + ConsignmentModel.STATUS + "} = {cst:pk}");
    sql.append(" }");
    sql.append(" WHERE {o:" + OrderModel.CREATIONTIME + "} >= ?fromDate");
    sql.append(" AND {o:" + OrderModel.CREATIONTIME + "} <= ?toDate");
    sql.append(" AND {o:" + OrderModel.ISISSUETAXINVOICE + "} = ?isUseFullTaxInvoice");
    sql.append(" AND {bp:" + BusinessPartnerModel.STATUS + "} = ?conditionTrue");
    sql.append(" AND {cst:code} IN (?consignmentStatus)");
    sql.append(" GROUP BY {cons:" + ConsignmentModel.PK + "}, {cons:" + ConsignmentModel.CODE + "}, {cons:"
            + ConsignmentModel.BUSINESSPARTNER + "}");
    sql.append(" ORDER BY {cons:" + ConsignmentModel.CODE + "}");
    final FlexibleSearchQuery query = new FlexibleSearchQuery(sql.toString());
    query.getQueryParameters().putAll(attr);
```

VC-22
Avoid building query strings in the method, rather declare at class level.

Query with Time Stamps

It is not advisable to use least precision time (in milli seconds) in the queries. This kind of queries will not use Query cache and will effect the performance.



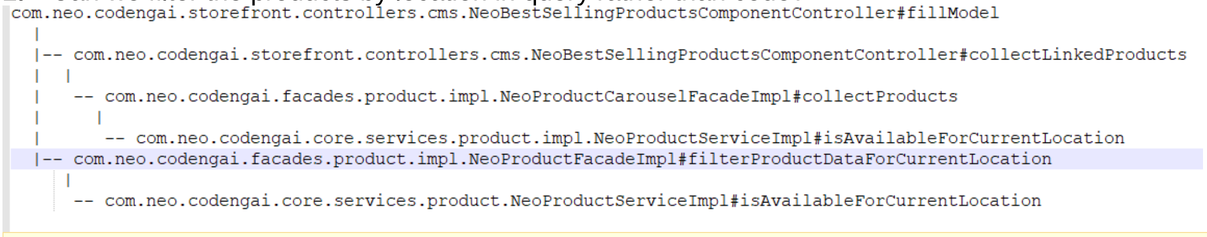
```
@Override
public List<ConsignmentProcessModel> findConsignmentCompletionProcess()
{
    final NeoGlobalConfigModel globalConfig = neoCommonDao.findGlobalConfigByCode("consignmentCompletionDayPeriod");
    final Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.DAY_OF_MONTH, globalConfig == null ? -7 : -Integer.parseInt(globalConfig.getValue()));


    final Map<String, Object> params = new HashMap<>();
    params.put(ProcessTaskModel.EXECUTIONTIMEMILLIS, calendar.getTimeInMillis());
    params.put(ProcessTaskModel.ACTION, "waitForConfirmCompleted");

    final SearchResult<ConsignmentProcessModel> searchResult = flexibleSearchService
            .<~> search(FIND_CONSIGNMENT_COMPLETION_PROCESS, params);
    return searchResult.getResult();
}
```

VC-23
Evaluate/Review the need of SQL statements, which do not use the Query cache (due to milliseconds/seconds time stamps) considering the business need.

Data filtering - API usage

In the below call hierarchy, there are two possible issues which can be reviewed.

1. Do we need to filter the products two times for the same product set (once in *collectLinkedProducts()* flow and another time in *filterProductDataForCurrentLocation()* flow?
2. Can we filter the products by location in query rather than code?



```
com.neo.codengai.storefront.controllers.cms.NeoBestSellingProductsComponentController#fillModel
  |
  |-- com.neo.codengai.storefront.controllers.cms.NeoBestSellingProductsComponentController#collectLinkedProducts
  |   |
  |    -- com.neo.codengai.facades.product.impl.NeoProductCarouselFacadeImpl#collectProducts
  |       |
  |        -- com.neo.codengai.core.services.product.impl.NeoProductServiceImpl#isAvailableForCurrentLocation
  |-- com.neo.codengai.facades.product.impl.NeoProductFacadeImpl#filterProductDataForCurrentLocation
      |
       -- com.neo.codengai.core.services.product.NeoProductServiceImpl#isAvailableForCurrentLocation
```

VC-24
Review the possibility to improve the logic in the API
com.neo.codengai.storefront.controllers.cms.NeoBestSellingProductsComponentController#fillModel