Monitoring and tooling

Query API

Most of the rendered data in storefront and Admin is an amalgamation of complex objects, which is many cases are heavily cached to ensure good performance of the platform. In some cases when performing analysis it is required to get access to raw fresh data. Query API specifically targets this problem.

Query API is not only concerned with persistence layer but a multipurpose tool which can be extended via extension point. Out of the box the following API extensions are supported:

API	Version	Supported Nodes	Purpose	Example
SQL:Core	3.0.0+	ADM, API, SFx	SQL interface for core RDBMS	`select count(*) from TSKUPRICE`
HQL:Core	3.0.0+	ADM, API, SFx	Hibernate QL interface for core RDBMS	`select count(s) from SkuPriceEntity s`
HQL:Payment	3.0.0+	ADM	Hibernate QL interface for payment RDBMS	`select p from PaymentGatewayParameterEntity p`
IceCat:Product	3.3.0+ SaaS	ADM	IceCat search interface to validate product XML	`72514951,72514952`
FT:Product	3.0.0+	API,SFx	Lucene full text query interface	`brand:toshiba name:w50`

In order to perform a query over desired medium:

- Select Node on which query to be performed
- Select type of API to use and click "Add tab" button

- Enter query in required formal and click "Play" button

Cache monitoring

For best performance each application node maintains local cache to reduce contingency on slow resources or speed up results of complex calculations. Cache monitoring panel allows to view all active caches on all nodes in the cluster with corresponding statistics, which include:
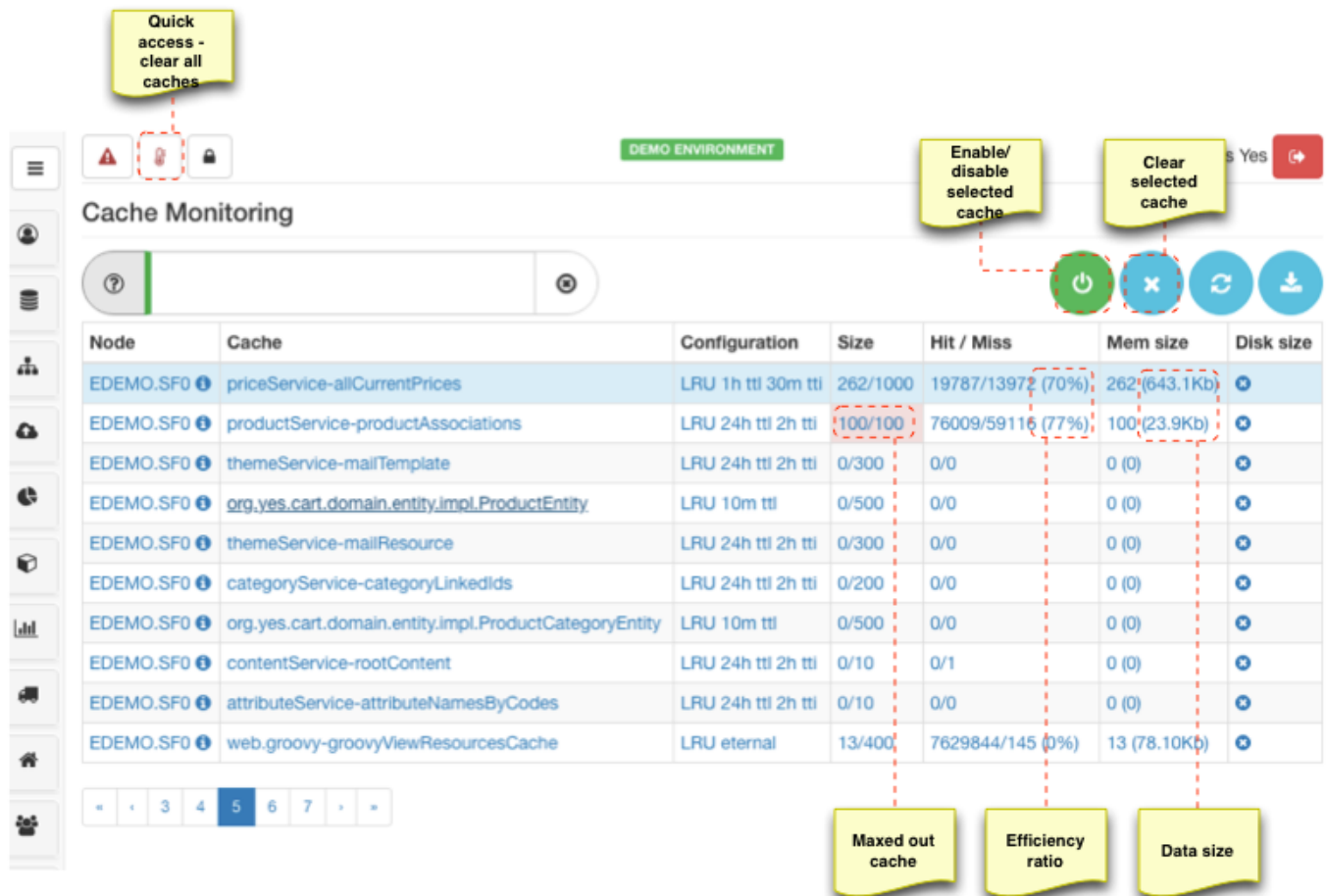
- Configuration parameters i.e. caching strategy and expiry timeframes
- Declared size and active size - essential for detecting flooded caches (performance tuning)
- Hit and Miss and efficiency ratio - essential for detecting inefficient caching (performance tuning)
- Memory size - RAM usage indicators (scalability)

All declared caches are active by default. However if is possible to enable/disable individual caches at runtime.

Caches can be cleared all at once, or individually by selecting specific cache and clicking "Evict" button

Evict all caches also features on the quick access buttons menu. This is an essential tool for development

Cache search has a number of "smart search" options to list flooded, most used and largest in size caches.

Quick
access -
clear all
caches

⚠ 🔑 🔒   DEMO ENVIRONMENT   Enable/   Clear   s Yes ⏻
                                                                     disable   selected
                                                                     selected   cache
Cache Monitoring   cache

(?)   (◉)   ⏻   ✕   ⟳   ⬇

Node	Cache	Configuration	Size	Hit / Miss	Mem size	Disk size
EDEMO.SF0 ⓘ	priceService-allCurrentPrices	LRU 1h ttl 30m tti	262/1000	19787/13972 (70%)	262 (643.1Kb)	✕
EDEMO.SF0 ⓘ	productService-productAssociations	LRU 24h ttl 2h tti	100/100	76009/59116 (77%)	100 (23.9Kb)	✕
EDEMO.SF0 ⓘ	themeService-mailTemplate	LRU 24h ttl 2h tti	0/300	0/0	0 (0)	✕
EDEMO.SF0 ⓘ	org.yes.cart.domain.entity.impl.ProductEntity	LRU 10m ttl	0/500	0/0	0 (0)	✕
EDEMO.SF0 ⓘ	themeService-mailResource	LRU 24h ttl 2h tti	0/300	0/0	0 (0)	✕
EDEMO.SF0 ⓘ	categoryService-categoryLinkedIds	LRU 24h ttl 2h tti	0/200	0/0	0 (0)	✕
EDEMO.SF0 ⓘ	org.yes.cart.domain.entity.impl.ProductCategoryEntity	LRU 10m ttl	0/500	0/0	0 (0)	✕
EDEMO.SF0 ⓘ	contentService-rootContent	LRU 24h ttl 2h tti	0/10	0/1	0 (0)	✕
EDEMO.SF0 ⓘ	attributeService-attributeNamesByCodes	LRU 24h ttl 2h tti	0/10	0/0	0 (0)	✕
EDEMO.SF0 ⓘ	web.groovy-groovyViewResourcesCache	LRU eternal	13/400	7629844/145 (0%)	13 (78.10Kb)	✕

« ‹ 3 4 **5** 6 7 › »

Maxed out   Efficiency   Data size
cache   ratio

Performance sensors SaaS

Performance sensors are fine detailed trackers or service layer API invocations.

Each sensor is Node specific and collects information on:

- Total amount of time API is used - to identify execution hot spots
- Invocation count of an API - to identify most used API (possible caching recommendation)
- Average, min and max time - to identify resource contingencies and slow API

Node	Sensor	Total Time	Invocation count	Average Time	Min Time	Max Time
EDEMO.SF0 ⓘ	ShoppingCartStateServiceImpl->ShoppingCartStateService.findByGuid(..)	1h 35m 17s 789ms	10039644	0.56ms		2s 661ms
EDEMO.JAM ⓘ	CustomerOrderServiceImpl->CustomerOrderService.findByReference(..)	59m 21s 719ms	855418	4.16ms		159ms
EDEMO.JAM ⓘ	ShoppingCartStateServiceImpl->GenericService.findByCriteriaIterator(..)	49m 48s 14ms	5█	█4ms	46ms	11s 347ms
EDEMO.SF0 ⓘ	ShopServiceImpl->ShopService.getShopCategoriesIds(..)	39m 36s 933ms	8█	0.26ms		168ms
EDEMO.SF0 ⓘ	ShopServiceCachedImpl->ShopService.getShopCategoriesIds(..)	38m 42s 744ms	4449677	0.52ms		168ms
EDEMO.SF0 ⓘ	AttributeServiceImpl->AttributeService.findAttributesByCodes(..)	33m 1s 139ms	1720152	1.15ms		200ms
EDEMO.SF0 ⓘ	ContentServiceFacadeImpl->ContentServiceFacade.getDynamicContentBody(..)	31m 28s 723ms	858958	2.19ms	1ms	650ms
EDEMO.SF0 ⓘ	ContentServiceCMS3Impl->ContentService.findContentIdBySeoUri(..)	21m 35s 507ms	1733340	0.74ms		462ms
EDEMO.SF0 ⓘ	AttributeServiceCachedImpl->AttributeService.findAttributesByCodes(..)	20m 26s 3ms	860076	1.42ms		200ms
EDEMO.SF0 ⓘ	ProductServiceImpl->ProductService.findFilteredNavigationRecords(..)	20m 927ms	1303922	0.92ms		382ms

Performance samples SaaS

Monitoring CPU and memory consumption in any given application may prove crucial in identifying performance bottleneck and resource restraints.

Sampling nodes in cluster provides essential information about environment of each node: JVM used, CPU cores and load, memory consumption and garbage collector stats. These metrics are invaluable when assessing resources required for the platform or any resource leaks.

Performance samples can also be downloaded as CSV extract for future reference.

> It is recommended to constantly review this information and create new samples to prevent problems before that may occur

> REST API can be used to feed this information into your environment tooling

Performance Samples

Perform CG

Take new sample

DEMO ENVIRONMENT

...gged in as Yes

Node	Date/Time	JDK	CPU cores	Uptime	Average Load	Heap Init-Max, Used/Committed	Non-Heap Init-Max, Used/Committed	GC Count / Time
EDEMO.SF0	2019-12-07 15:16:49	Java HotSpot(TM) 64-Bit Server VM Oracle Corporation v.25.161-b12	6	278h 19m 34s 231ms	0.86%	768.0MB - 1.4GB, 724.3MB / 1.3GB (54%)	2.4MB - *, 473.1MB / 559.4MB (85%)	12238 / 4m 43s 656ms
EDEMO.SF0	2019-12-07 15:17:01	Java HotSpot(TM) 64-Bit Server VM Oracle Corporation		278h 19m 46s 9ms	1.05%	768.0MB - 1.4GB, 785.8MB / 1.3GB (59%)	2.4MB - *, 473.2MB / 559.4MB (85%)	12238 / 4m 43s 656ms
EDEMO.JAM	2019-12-07 15:16:49	Java HotSpot(TM) Oracle Corporation		278h 19m 34s 251ms	0.86%	768.0MB - 1.4GB, 724.9MB / 1.3GB (54%)	2.4MB - *, 473.2MB / 559.4MB (85%)	12238 / 4m 43s 656ms
EDEMO.JAM	2019-12-07 15:17:01	Java HotSpot(TM) 64-Bit Server VM Oracle Corporation v.25.161-b12	6	278h 19m 46s 26ms	1.05%	768.0MB - 1.4GB, 786.3MB / 1.3GB (59%)	2.4MB - *, 473.2MB / 559.4MB (85%)	12238 / 4m 43s 656ms
EDEMO.SF1	2019-12-07 15:16:49	Java HotSpot(TM) 64-Bit Server VM Oracle Corporation v.25.161-b12	6	278h 19m 34s 245ms	0.86%	768.0MB - 1.4GB, 724.7MB / 1.3GB (54%)	2.4MB - *, 473.2MB / 559.4MB (85%)	12238 / 4m 43s 656ms
EDEMO.SF1	2019-12-07 15:17:01	Java HotSpot(TM) 64-Bit Server VM Oracle Corporation v.25.161-b12	6	278h 19m 46s 16ms	1.05%	768.0MB - 1.4GB, 786.2MB / 1.3GB (59%)	2.4MB - *, 473.2MB / 559.4MB (85%)	12238 / 4m 43s 656ms

Sample of SF0 node showing JVM, CPU health and memory consumption

Tasks SaaS

The platform has several asynchronous recurring processes in order to maintain its functions such as order state machine, price calculations, data maintenance, email processing etc. All schedules are declared in config-cronjob.properties which set the default rhythm for the platform, however in most production systems this rhythm requires some orchestration in many cases real time. SaaS offers a dedicated monitoring panel where all services on all nodes can be monitored, paused, triggered manually and re-scheduled at runtime.

Tasks

DEMO ENVIRONMENT

Logged in as Yes

Node	Name / Current Schedule	Schedule Info	Original Schedule	Last Run	Previous Run	Next Run	Running
EDEMO.SF0	Reindex All Products / 0 0 5 * * ? 2099	seconds: 0 minutes: 0 hours: 5 daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: 2099	0 0 5 * * ? 2099			2099-01-01 05:00:00 28879d 14h 43m 20s	
EDEMO.SF0	Inventory Changes Product Indexing / 0 0/5 * * * ?	seconds: 0 minutes: 0,5,10,15,20,25,30,35,40,45,50,55 hours: * daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: *	0 0/5 * * * ?	2019-12-07 14:15:00		2019-12-07 14:20:00 3m 20s	
EDEMO.SF0	product FT re-index / (manual)	product ... (0)	(manual)				

Annotations:
- Paused schedule
- Next run time
- Time left till next run
- App node which executes this task
- Schedule description - in this case every 5 minutes
- Empty circle indicates task is not currently running

Managing task schedules can be accomplished in three distinct actions:

- Run task now - platform automatically calculates a one-off schedule to run task in 30seconds, after task has run manually it will become unscheduled and not run anymore
- Restore schedule - platform automatically will use the original schedule to from properties file
- Reschedule - user is invited to change the cron expression to instruct the platform to perform task at a different schedule

Tasks / Send Mail

Mail

Node	Name / Current Schedule	Schedule Info	Original Schedule	Last Run	Previous Run	Next Run	Running
...MO.JAM	Send Mail / 0 0/10 * * * ?	seconds: 0 minutes: 0,10,20,30,40,50 hours: * daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: * Bulk send mail ... completed, send: 0, failed: 0	0 0/10 * * ?		2019-12-07 14:10:00	2019-12-07 14:20:00 3m 19s	○

Selected Task

1

Are you sure?

Run this task now (within next 30 seconds)

Name

Send Mail

Original Schedule

0 0/10 * * * ?

Current Schedule

0 0/10 * * * ?

Cancel OK

Are you sure?

Change schedule for this task

Name

Send Mail

Original Schedule

0 0/10 * * * ?

Current Schedule

0 0/10 * * * ?

New Schedule

0 0/10 * * * ?

Cancel OK

Tasks / Send Mail

Mail

Node	Name / Current Schedule	Schedule Info	Original Schedule	Last Run	Previous Run	Next Run	Running
EDEMO.JAM ⓘ	Send Mail / 0 0/10 * * * ?	seconds: 0 minutes: 0,10,20,30,40,50 hours: * daysOfMonth: * months: * daysOfWeek: ? lastdayOfWeek: false nearestWeekday: false NthDayOfWeek: 0 lastdayOfMonth: false years: * Bulk send mail ... completed, send: 0, failed: 0	0 0/10 * * ?			2019-12-07 14:30:00 7m 47s	●

Storefront component rendering diagnostics SaaS

Storefront themes multiplied by complexities of CMS further obscured by vast variety of data and caching sometimes results in questioning why page rendering has been done in a certain way? or why some components are not present on the page? or are present but should not be?

SFG SaaS frontend application contains a setting to produce verbose output during page rendering. In order to enable this mode set shop attribut e: **Maintenance: enable page render trace**. This attribute can also be observed on the Overview tab. When enabled detailed information will be printed out during rendering including:

- Full path of the components rendered in the page
- CMS elements included in the page
- Rendering times
- Caching information
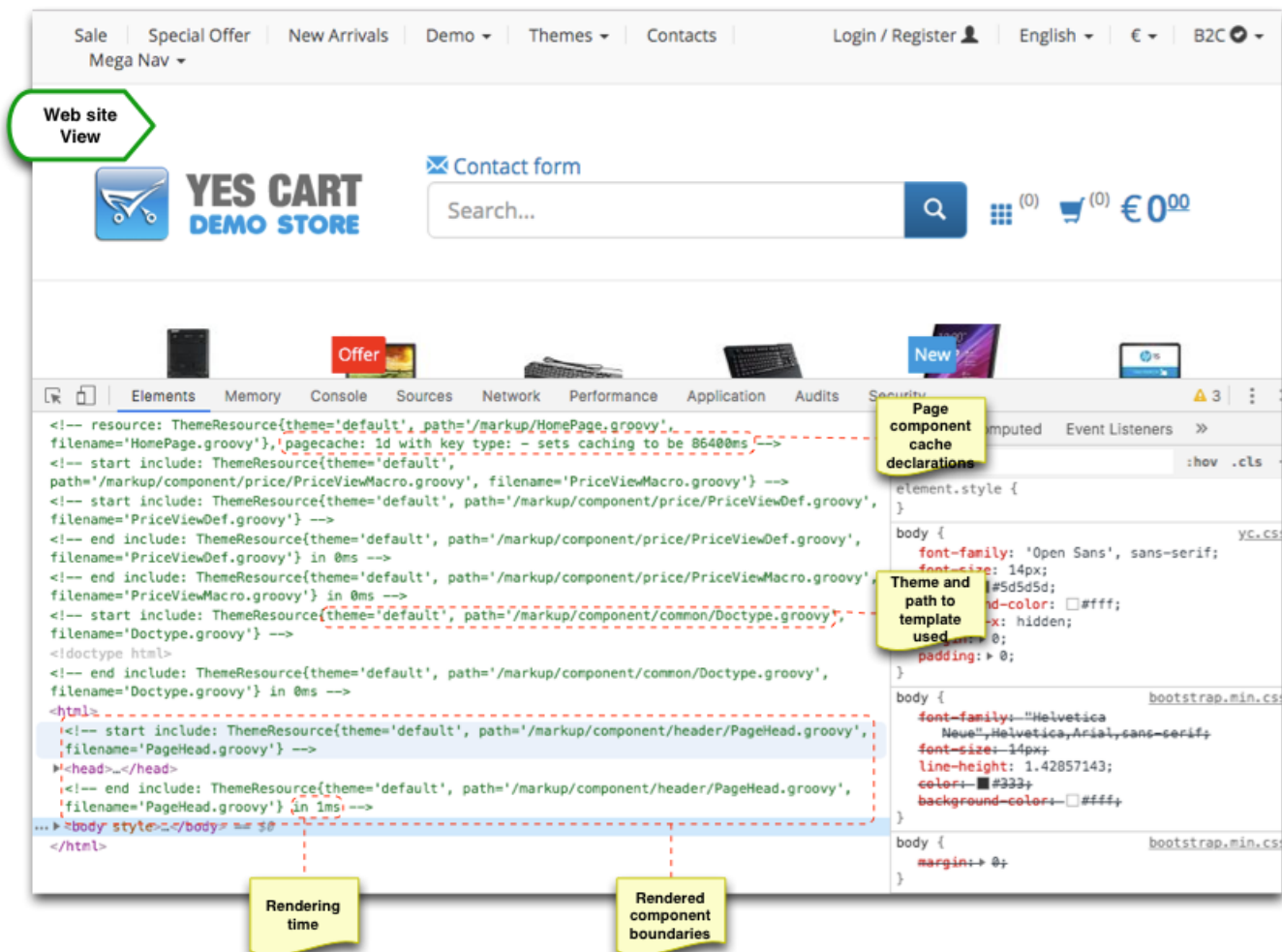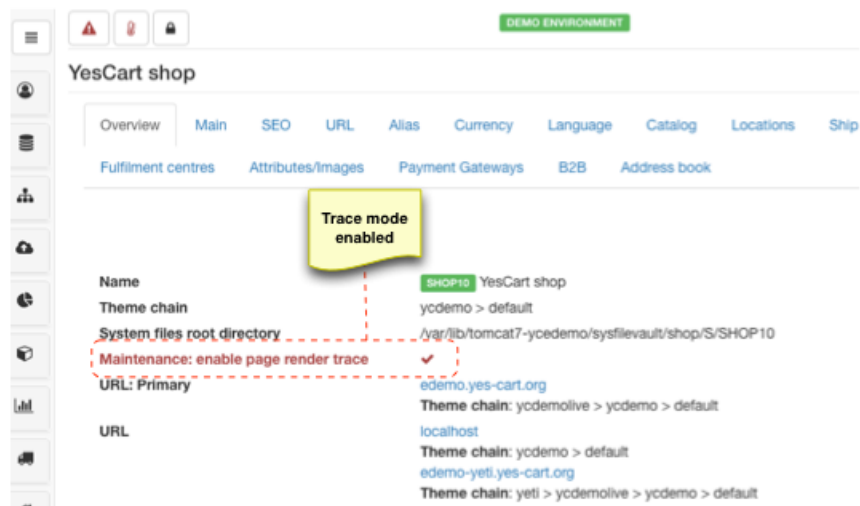
Cluster

Cluster defines a network of applications (nodes). Cluster is either preconfigured or nodes are auto-discovered depending on the discovery module used (either WebService, REST or JGroups multicast connector).

Each node declares cluster namespace it belongs to, type (ADM, API or SFx), mode of operation and whether it uses full text search module. Along with this information cluster service can query each node to give full details of the build (i.e. all modules that are currently loaded).

Overview provided by cluster monitoring is vital in understanding platform infrastructure and composition of individual applications.

> Detailed documentation on clustering is available upon request

Cluster namespace and App Node type

App Node mode

Indicator of App Node supporting Full text search

Application logs

All platform applications use Logback over SLF4j configuration. Capabilities of Logback a beyond the scope of this discussion and we highly recommend reviewing logback official documentation if you are not familiar with this framework.

Shop specific logging

For multi-tenant setups it is recommended to sift logs by shop discriminator. Platform provides several flavours of ready to use implementations:

ShopCodeLogDiscriminator

Uses current shop context to set **shopCode** variable, which produces value such as "SHOP10". This value can be used in file names to direct message of specific shop into a separate file.

```
    <appender name="SHOPPAY"
class="ch.qos.logback.classic.sift.SiftingAppender">
        <!-- declare discriminator -->
        <discriminator
class="org.yes.cart.utils.log.ShopCodeLogDiscriminator"/>
        <sift>
            <!-- shopCode can be use in appender name (e.g. view in JMX) -->
            <appender name="SHOPPAY-${shopCode}"
class="ch.qos.logback.core.rolling.RollingFileAppender">
                <!-- shopCode can be use in file name -->
                <File>${catalina.base}/logs/yc-${shopCode}-pay.log</File>
                <Append>true</Append>
                <encoder>
                    <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %5p %c{1}:%L -
%m%n</pattern>
                </encoder>
                <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
                    <maxIndex>10</maxIndex>
                    <FileNamePattern>${catalina.base}/logs/yc-${shopCode}-p
ay.log.%i.zip</FileNamePattern>
                </rollingPolicy>
                <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
                    <MaxFileSize>10MB</MaxFileSize>
                </triggeringPolicy>
            </appender>
        </sift>
    </appender>
```

ShopCodeAndLevelLogDiscriminator

Uses current shop context + log message level to set **shopCode** variable, which produces value such as "SHOP10-WARN". This value can be used in file names to direct message of specific level of specific shop into a separate file

```
  <appender name="DEFAULT"
class="ch.qos.logback.classic.sift.SiftingAppender">
        <!-- declare discriminator -->
        <discriminator
class="org.yes.cart.utils.log.ShopCodeAndLevelLogDiscriminator"/>
        <sift>
            <!-- shopCode can be use in appender name (e.g. view in JMX) -->
            <appender name="DEFAULT-${shopCode}"
class="ch.qos.logback.core.rolling.RollingFileAppender">
                <!-- shopCode can be use in file name -->
                <File>${catalina.base}/logs/yc-${shopCode}.log</File>
                <Append>true</Append>
                <encoder>
                    <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %5p %c{1}:%L -
%m%n</pattern>
                </encoder>
                <rollingPolicy
class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
                    <maxIndex>10</maxIndex>
                    <FileNamePattern>${catalina.base}/logs/yc-${shopCode}.l
og.%i.zip</FileNamePattern>
                </rollingPolicy>
                <triggeringPolicy
class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
                    <MaxFileSize>10MB</MaxFileSize>
                </triggeringPolicy>
            </appender>
        </sift>
    </appender>
```

Out of the box logs

Log file(s)	Sift	Level	Purpose
Storefront (API, SFx)			
yc-${shopCode}.log	shop code + level	INFO	DEFAULT-${shopCode} appender (root)
yc-${shopCode}-pay.log	shop code	DEBUG	SHOPPAY appender, payment gateway related logging, packages: • org.yes.cart.web.filter.payment • org.yes.cart.payment.impl • org.yes.cart.web.page.payment.callback
yc-orderexport.log		INFO	ORDEREXPORT appender, order export related logging, packages: • org.yes.cart.orderexport
yc-orderstate.log		INFO	ORDERSTATE appender, order transition related logging, packages: • org.yes.cart.service.order

yc-mail.log		INFO	MAIL appender, email generation and send logging, packages: • org.yes.cart.service.mail.impl.MailComposerImpl • org.yes.cart.bulkjob.mail.BulkMailProcessorImpl • org.yes.cart.domain.message.consumer.CustomerRegistrationMessageListener • org.yes.cart.domain.message.consumer.ManagerRegistrationMessageListener • org.yes.cart.domain.message.consumer.StandardMessageListener • org.yes.cart.web.aspect.ContactFormAspect • org.yes.cart.web.aspect.NewsletterAspect • org.yes.cart.web.aspect.RegistrationAspect • org.yes.cart.service.domain.aspect.impl.CustomerRegistrationAspect • org.yes.cart.service.domain.aspect.impl.ManagerRegistrationAspect • org.yes.cart.service.domain.aspect.impl.PaymentAspect • org.yes.cart.service.domain.aspect.impl.BaseOrderStateAspect • org.yes.cart.service.domain.aspect.impl.OrderStateChangeListenerAspect • org.yes.cart.orderexport.mail.EmailNotificationOrderExporterImpl
yc-maildump.log		INFO	MAILDUMP appender, log full content of email that was sent
yc-audit.csv		INFO	AUDIT appender, log persistence updates audit records ⚠ Change to TRACE to enable audit logging
yc-${shopCode}-ftq.log	shop code	INFO	FTQ-${shopCode} appender, logs all full text queries ⚠ Change to DEBUG to enable query logging ⚠ Change to TRACE to enable query explanations logging
yc-ws.log		ERROR	WS appender, logs web services communication (WS.IN and WS.OUT) ⚠ Change to INFO to enable logging
yc-config.log		INFO	CONFIG appender, logs information on loaded modules and extension points ⚠ Change to DEBUG to enable logging
yc-${shopCode}-sac.log	shop code	DEBUG	SAC (Security access control) appender, logs access violations
Admin (ADM)			
yc-${shopCode}.log	shop code + level	INFO	DEFAULT-${shopCode} appender (root)
yc-${shopCode}-pay.log	shop code	DEBUG	SHOPPAY appender, payment gateway related logging, packages: • org.yes.cart.web.filter.payment • org.yes.cart.payment.impl • org.yes.cart.web.page.payment.callback
yc-${shopCode}-job.log	shop code + level	INFO	JOB-${shopCode} appender, logs all tasks executions, packages: • org.yes.cart.bulkjob • org.yes.cart.service.async.impl
yc-${shopCode}-import.log	shop code + level	INFO	BULKIMPORT-${shopCode} appender, logs all data imports, packages: • org.yes.cart.bulkimport • org.yes.cart.bulkjob.bulkimport • org.yes.cart.service.async.impl
yc-remote.log		INFO	REMOTE (file upload/download/move/delete operations) appender, logs all data imports, packages: • org.yes.cart.remote
yc-orderexport.log		INFO	ORDEREXPORT appender, order export related logging, packages: • org.yes.cart.orderexport
yc-orderstate.log		INFO	ORDERSTATE appender, order transition related logging, packages: • org.yes.cart.service.order

yc-mail.log		INFO	MAIL appender, email generation and send logging, packages: • org.yes.cart.service.mail.impl.MailComposerImpl • org.yes.cart.bulkjob.mail.BulkMailProcessorImpl • org.yes.cart.domain.message.consumer.CustomerRegistrationMessageListener • org.yes.cart.domain.message.consumer.ManagerRegistrationMessageListener • org.yes.cart.domain.message.consumer.StandardMessageListener • org.yes.cart.web.aspect.ContactFormAspect • org.yes.cart.web.aspect.NewsletterAspect • org.yes.cart.web.aspect.RegistrationAspect • org.yes.cart.service.domain.aspect.impl.CustomerRegistrationAspect • org.yes.cart.service.domain.aspect.impl.ManagerRegistrationAspect • org.yes.cart.service.domain.aspect.impl.PaymentAspect • org.yes.cart.service.domain.aspect.impl.BaseOrderStateAspect • org.yes.cart.service.domain.aspect.impl.OrderStateChangeListenerAspect • org.yes.cart.orderexport.mail.EmailNotificationOrderExporterImpl
yc-maildump.log		INFO	MAILDUMP appender, log full content of email that was sent
yc-audit.csv		INFO	AUDIT appender, log persistence updates audit records ⚠ Change to TRACE to enable audit logging
yc-ws.log		ERROR	WS appender, logs web services communication (WS.IN and WS.OUT) ⚠ Change to INFO to enable logging
yc-config.log		INFO	CONFIG appender, logs information on loaded modules and extension points ⚠ Change to DEBUG to enable logging
yc-${shopCode}-sac.log	shop code	DEBUG	SAC (Security access control) appender, logs access violations
yc-security.log		INFO	SECURITY (Spring) appender, packages: • org.springframework.security