Extension points

Overview

Extension points is mechanism introduced in version 3.5.0+ in order to allow for better modularisation and separation of custom code from core platform's functions. The crux of the approach is provision of standard hooks that can be used to add or reconfigure core functionality without modification of the core code.

Each custom module must bundle all code necessary to run its functions and expose its functions via spring context extension points described below in "Extension hooks" section.

To bundle the modules in custom builds maven profile technique is recommended to be used in pom.xml of specific applications being extended or direct dependency if it is a permanent core feature.

Extension points work with all jar files on class path so it is also possible to do seamless extension by putting extension points jars on class path of servlet container.

Extension hooks

Persistence

Persistence is handled using Hibernate, which is configured though hibernate configuration files. There files are provided as lists of names to the hibernate factory bean. Extending the mapping resources list allow to plug in additional mapping files that can include additional table mapping and/or queries.

Module	Extension point	Key points	Version
payment-api persistence	hibernate/ycp.xml	Provides mechanism for hibernate persistence extension (resource files ycp.xml)	3.5.0+
payment-api payment-persistence-resources.xml	payment-persistence-resources-ext.xml	Provides mechanism for hibernate persistence extension: **ycpProhibitedFields** mapping of prohibited fields (extend using LinkedHashMapBean)	3.5.0+
persistence persistence	hibernate/yc.xml	Provides mechanism for hibernate persistence extension (resource files yc.xml)	3.5.0+
persistence dao-persistence-resources.xml	dao-persistence-resources-ext.xml	Provides mechanism for hibernate persistence extension: **ycProhibitedFields** mapping of prohibited fields (extend using LinkedHashMapBean)	3.5.0+

persistence dao.xml	dao-mapping-ext.xml	Mapping for Domain model beans used by DAO layer: **entityClassFactoryMapping** mapping for Entity (extend using LinkedHashMapBean)	3.5.0+
persistence dao.xml	dao-ext.xml	DAO layer beans extension	3.5.0+

Core

In this section we refer to "core" as collective of modules which are used by all YC applications. Each module contains a number of hooks to allow specification of Spring beans and extending them in precise sequence.

Module	Extension point	Key points	Version
core core-services.xml	core-services-ext.xml	Services layer beans extension. Additionally exposes mapping extension points: **promotionContextFactoryActionsMapping** mapping for actions (extend using LinkedHashMapBean) **promotionCouponContextActionsMapping** mapping for actions (extend using LinkedHashMapBean)	3.5.0+
core core-services.xml	core-services-ext.xml	Services layer beans extension. Additionally exposes mapping extension points: **imageNameStrategyResolverStrategies** mapping for image strategies (extend using ArrayListBean) **fileNameStrategyResolverStrategies** mapping for image strategies (extend using ArrayListBean)	3.7.0+
core cache-config.xml	cache-config-ext.xml	Remote cache eviction extension point: **evictionConfig** mapping for remove cache evictions (extend using LinkedHashMapBean)	3.5.0+
core-dto core-dto-services.xml	core-dto-services-ext.xml	DTO (business) layer beans extension. Additionally exposes mapping extension points: **dtoInterfaceToClassFactoryMapping** mapping for DTO (extend using LinkedHashMapBean) **dtoAssemblerAdaptersRepositoryMapping** mapping for DTO adapters (extend using LinkedHashMapBean)	3.5.0+
core-module-orderstate core-orderstate-aspects.xml	core-orderstate-aspects-ext.xml	Order state machine notifications extensions: **paymentNotificationAspectAuthoriseShopperTemplates** mapping for AUTH or AUTH_CAPTURE result (extend using LinkedHashMapBean) **paymentNotificationAspectAuthoriseAdminTemplates** mapping for AUTH or AUTH_CAPTURE result (extend using LinkedHashMapBean) **paymentNotificationAspectCancelShopperTemplates** mapping for CANCEL result (extend using LinkedHashMapBean) **paymentNotificationAspectCancelAdminTemplates** mapping for CANCEL result (extend using LinkedHashMapBean) **paymentNotificationAspectShipmentShopperTemplates** mapping for shipment CAPTURE result (extend using LinkedHashMapBean) **paymentNotificationAspectShipmentAdminTemplates** mapping for shipment CAPTURE result (extend using LinkedHashMapBean) **orderStateChangeAspectShopperTemplates** mapping for order state change mails (extend using LinkedHashMapBean) **orderStateChangeAspectAdminTemplates** mapping for order state change mails (extend using LinkedHashMapBean)	3.5.0+
core-module-orderstate core-orderstate.xml	core-orderstate-ext.xml	Order state machine configuration: **orderStateManagerHandlersMapping** mapping for order events handlers (extend using LinkedHashMapBean) **orderStateManagerBeforeListenersMapping** mapping for order events listeners triggered before event (extend using LinkedHashMapBean) **orderStateManagerAfterListenersMapping** mapping for order events listeners triggered after event (extend using LinkedHashMapBean)	3.5.0+
jam-services core-federation-impex.xml	core-federation-impex-ext.xml	Order state machine configuration: **importFederationFacadeFilters** mapping impex filters (extend using LinkedHashMapBean) **exportFederationFacadeFilters** mapping impex filters (extend using LinkedHashMapBean)	3.5.0+

jam-services manager-dto-services.xml	manager-dto-services-ext.xml	VO remoting DTO layer beans extension. Additionally exposes mapping extension points: **voKeyToClassFactoryMapping** mapping for VO (extend using LinkedHashMapBean) **voAssemblerAdaptersRepositoryMapping** mapping for VO adapters (extend using LinkedHashMapBean)	3.5.0+
jam-services manager-services.xml	manager-services-validation-ext.xml	VO objects remote validator extension: **voValidationServiceMapping** mapping for remote validators (extend using LinkedHashMapBean)	3.5.0+
jam-services manager-services.xml	manager-services-ext.xml	VO remoting service layer beans extension	3.5.0+
core-module-reports core-reports.xml	core-reports-ext.xml	**reportObjectStreamFactoryAliasesMapping** mapping for java Classes to tags in report XML sent to FOP (extend using LinkedHashMapBean) **reportObjectStreamFactoryOmitFieldMapping** mapping for skipping fields in java Classes when generating XML sent to FOP (extend using LinkedHashMapBean)	3.6.0+
jam-services core-emailtemplates.xml	manager-services-ext.xml	**emailTemplatesConfigMapping** mapping for email templates configuration to extend email templates CMS module for custom email themes	3.7.0+

ImpEx

Import/Export extension points allow to plug various implementations of the importers and exporters. This functionality is introduced in **3.6.0+**

Module	Extension point	Key points	Version
core-module-impexcore-import.xml	manager-import-ext.xml	**importLookUpQueryParameterStrategyValueProviders** plugins for processing placeholders in CSV import descriptors **importCsvImportValueAdapterMap** mapping for value adapters from CSV text to object property value (only needed for customisations) **xmlFastBulkImportServiceHandlerMap** mapping for XML import handlers **importXmlImportValueAdapterMap** mapping for value adapters that can be used in XML import handlers	3.6.0+
core-module-impexcore-export.xml	manager-export-ext.xml	**exportCsvLookUpQueryParameterStrategyValueProviders** plugins for processing placeholders in CSV export descriptors **exportCsvExportValueAdapterMap** mapping for value adapters from object property value to CSV text (only needed for customisations) **exportXmlLookUpQueryParameterStrategyValueProviders** plugins for processing placeholders in XML export descriptors **xmlFastBulkExportServiceHandlerMap** mapping for XML export handlers **exportXmlExportValueAdapterMap** mapping for value adapters that can be used in XML export handlers	3.6.0+
jam manager-import.xml	manager-import-ext.xml	**importDataDescriptorResolverDataDescriptorReaders** plugins for reading data descriptors, which define which import service will be used for a particular import process **bulkImportServiceBeanMap** mapping for import services	3.6.0+
jam manager-export.xml	manager-export-ext.xml	**exportDataDescriptorResolverDataDescriptorReaders** plugins for reading data descriptors, which define which export service will be used for a particular export process **bulkExportServiceBeanMap** mapping for export services	3.6.0+

Admin app

Module	Extension point	Key points	Version
jam manager-import.xml	manager-import-ext.xml	import plugins	3.6.0+
jam manager-export.xml	manager-export-ext.xml	export plugins	3.6.0+
jam manager-report.xml	manager-report-ext.xml	report plugins	3.7.0+

jam applicationContext.xml	adm-applicationContext-ext.xml	application listener context specific extension	3.5.0+
jam jam-servlet.xml	adm-servlet-ext.xml	servlet context specific extension	3.5.0+
jam manager-cronjob.xml	adm-cronjob-ext.xml	Admin cron tasks extension: **managerCronScheduleTriggers** list of cron triggers (extend using ArrayListBean)	3.5.0+
jam	@Controller	Scanner packages: org.yes.cart.service.endpoint org.yes.cart.service.endpoint.impl	3.5.0+
jam manager-cluster.xml	manager-cluster-ext.xml	**queryDirectorPlugins** list of query plugins (extend using ArrayListBean)	3.6.0+

Web app support

Storefront apps common module.

Module	Extension point	Key points	Version
websupport websupport-webapp.xml	websupport-ext.xml	global sf context imports	3.5.0+
websupport websupport-cronjob.xml	websupport-cronjob-ext.xml	Webapps cron tasks extension: **webCronScheduleTriggers** list of cron triggers (extend using ArrayListBean)	3.5.0+
websupport websupport-cluster-listeners.xml	websupport-cluster-listeners-ext.xml	**queryDirectorPlugins** list of query plugins (extend using ArrayListBean)	3.6.0+

REST API app

Module	Extension point	Key points	Version
api api.xml	api-ext.xml	REST API layer beans extension. Additionally exposes mapping extension points: **roInterfaceToClassFactoryMapping** mapping for RO (extend using LinkedHashMapBean) **roAssemblerAdaptersRepositoryMapping** mapping for RO adapters (extend using LinkedHashMapBean)	3.5.0+
apiapplicationContext.xml	api-applicationContext-ext.xml	application listener context specific extension	3.5.0+
api rest-servlet.xml	api-servlet-ext.xml	servlet context specific extension	3.5.0+
api	@Controller	Scanner packages: org.yes.cart.web.service.rest	3.5.0+

Wicket storefront app

Module	Extension point	Key points	Version

store-wicket wicket.xml	wicket-ext.xml	Wicket specific app extensions: **wicketPagesMapping** wicket URI to pages mapping (extend using LinkedHashMapBean) **wicketPagesEncoderEnabledUrls** wicket URI list of SEO supported encoder pages (extend using ArrayListBean) **wicketResourceMounterEnabledPatterns** wicket resource mounter enabled patterns (extend using ArrayListBean) **wicketResourcesMapping** wicket IResources mapping (extend using LinkedHashMapBean) **wicketRendererPanelMap** wicket central panel view mapping (extend using LinkedHashMapBean) **wicketCategoryTypeMap** wicket category central view type mapping (extend using LinkedHashMapBean)	3.5.0+
store-wicketapplicationContext.xml	sfw-applicationContext-ext.xml	application listener context specific extension	3.5.0+

Groovy storefront app SaaS

Module	Extension point	Key points	Version
store-groovy groovy.xml	groovy-ext.xml	Groovy MO (view model) layer beans extension. Additionally exposes mapping extension points: **moInterfaceToClassFactoryMapping** mapping for MO (extend using LinkedHashMapBean) **moAssemblerAdaptersRepositoryMapping** mapping for MO adapters (extend using LinkedHashMapBean)	3.5.0+
store-groovy groovy-mvc.xml	groovy-mvc-ext.xml	Groovy specific app extensions: **groovyRendererPanelMap** groovy central panel view mapping (extend using LinkedHashMapBean) **groovyCategoryTypeMap** groovy category central view type mapping (extend using LinkedHashMapBean)	3.5.0+
store-groovy rest-mvc.xml	rest-mvc-ext.xml	Groovy REST API context specific extension	3.5.0+
store-groovyapplicationContext.xml	sfg-applicationContext-ext.xml	application listener context specific extension	3.5.0+
store-groovy groovy-servlet.xml	sfg-servlet-ext.xml	servlet context specific extension	3.5.0+
store-groovy	@Controller	Scanner packages: org.yes.cart.web.service.mixin org.yes.cart.web.service.groovy	3.5.0+
store-groovy rest-servlet.xml	sfg-rest-servlet-ext.xml	servlet context specific extension	3.5.0+
store-groovy	@Controller	Scanner packages: org.yes.cart.web.service.mixin org.yes.cart.web.service.rest	3.5.0+

Build hooks

"Build" in this context refers to maven build, which uses configuration bundles from **YC_HOME/env** directory. Each bundle is contained in a directory under **YC_HOME/env**. Files contained in those directories are included in various pom.xml files as filters to inject variables for placeholder replacements and also bundling files with web apps (e.g. yc-config.properties). Before bundling environment specific files they are preprocessed by filters.

This mechanism allows very flexible environment specific configuration.

Web filters

Web filters can be injected into storefront web.xml via config-web-filters.properties mapping. An example of such integration could be payment gateways that expect external callbacks to authorise/confirm transactions.

Runtime customisations

"Customisations" refer to dynamic configurations of the system to alter behaviour on per shop basis and override default system behaviour. All customisation points implement **Configuration** interface and are visible in the **System > Configurations** part of the Admin app. To re-configure customisation you can define them in the **System > Preferences** section by changing SYSTEM_EXTENSION_CFG_PROPERTIES which takes a form of property file.

The exact configuration are dictated by short guidance displayed in **System > Configurations** panel and takes form of key following: **[system unit].[interface name]=[Spring bean definition]**. For example configuration file may look like this:

```
...
SHOP10.productAvailabilityStrategy=productAvailabilityStrategyBackorderInS
tockOnly
SHOP10.productAvailabilityStrategy=productAvailabilityStrategyBackorderInS
tockOnly
CMS.contentService=contentServiceCMS1
CMS.dtoContentService=dtoContentServiceCMS1
CMS.contentFileNameStrategy=contentCMS1FileNameStrategy
...
```

Customisation interfaces

The following system units are identified by the platform currently:

- Shop - shop identified by shop code
- FC - fulfilment centre identified by fulfilment centre code
- CMS - global instance identified by code CMS
- SYS - global instance identified by code SYS

Interface	System unit	Description	Version
ProductAvailabilityStrategy	Shop.productAvailabilityStrategy	Implementation of the availability strategy that allows to determine ProductAvailabilityModel for a given product / SKU, which drives the UI rendering for add to cart (ATC) button. Has default implementation ProductAvailabilityStrategyDefaultImpl. As of 3.7.0+ new configuration is available **productAvailabilityStrategy BackorderInStockOnly** that allows to disable ATC if back order items do not have stock.	3.5.0+
InventoryResolver	FC.inventoryResolver	Implementation of the inventory service. Has default implementation InventoryResolverDefaultImpl which uses fulfilment centre inventory records to determine current stock state.	3.5.0+
DeliveryTimeEstimationVisitor	Shop.deliveryTimeEstimationVisitor	Implementation of the delivery time estimation. Has default implementation DeliveryTimeEstimationVisitorDefaultImpl which uses fulfilment centre lead times, shipping method lead times and exclusions in order to estimate potential delivery time.	3.5.0+

TaxProvider	Shop.taxProvider	Implementation of the tax calculation service. Has default implementation TaxProviderDefaultImpl which uses tax and tax configuration setting to workout applicable tax rate for given SKU, additionally contains regional specific settings triggered by address used by customers.	3.5.0+
PricingPolicyProvider	Shop.pricingPolicyProvider	Implementation of the pricing policy service that is used to determine applicable price list to use for given customer. Has default implementation PricingPolicyProviderCustomerAttributeImpl which uses customer pricing policy property.	3.5.0+
PriceResolver	Shop.priceResolver	Implementation of the pricing service that determines the price customer should pay for a SKU. Has default implementation PriceResolverDefaultImpl which uses cheapest price available policy to determine price for SKU (before promotions are applied).	3.5.0+
CartContentsValidator	Shop.cartContentsValidator	Implementation of the cart validation service that is used to determine if cart is in consistent state and customer should be allowed to go through with the checkout. Has default implementation on a compound validator (CheckoutBlockedValidator + ItemsAvailableValidator) which determine if customer is allowed to checkout YCE and whether items in the cart are available.	3.5.0+
OrderAssemblerPostProcessor	Shop.orderAssemblerPostProcessor	Implementation of the order post processor service that is used to enhance order details just after the order has been assembled from cart and is ready to be persisted as "pending". There is no default service as this is pure extension hook, see "noopOrderAssemblerPostProcessor" for more details in config panel.	3.7.0+
ContentService	CMS.contentService	Implementation of content service currently supports two values: **contentServiceCMS1** - old style CMS based on Category domain model **contentServiceCMS3** (default) - new style CMS based on Content domain model	3.5.0+
DtoContentService	CMS.dtoContentService	Implementation of content service currently supports two values: **dtoContentServiceCMS1** - old style CMS based on Category domain model **dtoContentServiceCMS3** (default) - new style CMS based on Content domain model	3.5.0+
MediaFileNameStrategy	CMS.contentFileNameStrategy	Implementation of content service currently supports two values: **contentCMS1FileNameStrategy** - old style CMS based on Category domain model **contentCMS3FileNameStrategy** (default) - new style CMS based on Content domain model	3.5.0+
MediaFileNameStrategy	CMS.contentImageNameStrategy	Implementation of content service currently supports two values: **contentCMS1ImageNameStrategy** - old style CMS based on Category domain model **contentCMS3ImageNameStrategy** (default) - new style CMS based on Content domain model	3.5.0+
SecurityAccessControlService	SYS.httpSecurityAccessControlService	Implementation of SAC service currently supports one value: **servletRequestSecurityAccessControlService** - default implementation that include blacklisting/whitelisting of IP and throttling settings	3.7.0+
SSOProcessor	Shop.SSOProcessor	Implementation of the SSO bridge for frontend.	4.1.0+