Open Orchestra

Release 1.1.0

October 14, 2016

The main documentation is organized into several sections:

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

1.1 Versions 1.x.x

1.1.1 1.1.x

CHANGELOG from 1.1.4 to 1.1.5

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs

- Media file bundle
- Elastica bundle

Other changes

- **[open-orchestra-cms-bundle]** Greatly reduce the generation time of some node facades and status facades by muting some expansives attributes when not required by the context of the call #1923 (*ngilain*)
- **[open-orchestra-cms-bundle]** Optimization loading of underscore templates for the node and the list view #1880 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Optimize performances on folder creation #306 (*ngilain*)

[OO-HOTFIX]

- **[open-orchestra-cms-bundle]** Temporary disable the possibility to delete a status in order to speed up significantly API responses including status #1920 (*ngilain*)

New features

- **[open-orchestra-cms-bundle]** Add GroupRepository::findAllWithSiteId #1919 (*ngilain*)
- **[open-orchestra-media-bundle]** Add index folder document #229 (*alavieille*)

Bug fixes

- **[open-orchestra-cms-bundle]** Authorize status update when user is super admin #1851 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix cache tag block #1846 (*alavieille*)
- **[open-orchestra-cms-bundle]** Optimized access of model group role #1839 (*alavieille*)
- **[open-orchestra-display-bundle]** Fix slideshow block in IE11 #240 (*alavieille*)
- **[open-orchestra-libs]** Fix xml and yaml reader of search metadata #62 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** When a media modal is closed, it's correctly removed of DOM #289 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Fix button browse of form media type #276 (*alavieille*)

[OO-PERFORMANCE]

- **[open-orchestra-mongo-libs]** Improve singleHydrateAggregateQuery performance #51 (*itkg-canne*)

CHANGELOG from 1.1.3 to 1.1.4

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle

- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Bug fixes

- **[open-orchestra-cms-bundle]** Optimized access of model group role #1839 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fixed slowness when show node tree, Optimized NodeGroupRoleVoter #1836 (*alavieille*)
- **[open-orchestra-libs]** Fix xml and yaml reader of search metadata #62 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Fix button browse of form media type #276 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** MediaTransformer need now a instance of TranslationChoiceManagerInterface instead of TranslationChoiceManager #269 (*alavieille*)
- **[open-orchestra-media-bundle]** Add index folder document #229 (*alavieille*)

CHANGELOG from 1.1.2 to 1.1.3

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle

- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Bug fixes

- **[open-orchestra-cms-bundle]** Removing boLabel of the error nodes #1828 (*alavieille*)
- **[open-orchestra-cms-bundle]** Radio button *Link to the current site* is only available when you create a content #1827 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix elements count in content pagination. #1825 (*itkg-canne*)
- **[open-orchestra-cms-bundle]** Fix currently published when a published content is duplicate #1824 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix fill label block form #1806 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix dependency of modelBundle in class fieldTypeType #1805 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fixing of category of item template in the right form #1796 (*alavieille*)
- **[open-orchestra-cms-bundle]** Change the appConfigurationView creation order #1784 (*ngilain*)
- **[open-orchestra-libs]** Fix xml and yaml reader of search metadata #62 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** MediaTransformer need now a instance of TranslationChoiceManagerInterface instead of TranslationChoiceManager #269 (*alavieille*)
- **[open-orchestra-workflow-function-bundle]** Prevents the deletion of workflow function if it used by an user #135 (*alavieille*)

Deprecated

- **[open-orchestra-cms-bundle]** ContentRepository::countByContentTypeInLastVersion method is deprecated since version 1.1.3 and will be removed in 2.0, it is replace by ContentRepository::countByContentTypeAndSiteInLastVersion method. #1825 (*alavieille*)
- **[open-orchestra-model-bundle]** *ContentRepository::countByContentTypeInLastVersionWithFilter* method is deprecated since version 1.1.3 and will be removed in 1.3.0, it is replace by *ContentRepository::countByContentTypeAndSiteInLastVersionWithFilter* method. #613 (*itkg-canne*)
- **[open-orchestra-model-bundle]** *ContentRepository::countByContentTypeInLastVersion* method is deprecated since version 1.1.3 and will be removed in 1.3.0, it is replace by *ContentRepository::countByContentTypeAndSiteInLastVersion* method. #613 (*itkg-canne*)

- **[open-orchestra-model-interface]** ContentRepositoryInterface::countByContentTypeInLastVersion method is deprecated since version 1.1.3 and will be removed in 2.0, it is replace by ContentRepository::countByContentTypeAndSiteInLastVersion method. #212 (*alavieille*)

Other changes

- **[open-orchestra-cms-bundle]** optimize route generation on node update #1819 (*itkg-nanne*)

CHANGELOG from 1.1.1 to 1.1.2

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Bug fixes

- **[open-orchestra-cms-bundle]** Fix the display of active tab of translated value form #1770 (*djamnazi*)
- **[open-orchestra-cms-bundle]** Fix merge form block strategy with transformer and subscriber #1768 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix data normalization when submitting a content form #1767 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix popin error when click on button duplicate version #1766 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove override default value of choice in javascript #1765 (*alavieille*)

- **[open-orchestra-cms-bundle]** Fix exception on node role list update in group when node contains deleted #1754 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Disallow creation of node in a language belonging fronts languages but not current site aliases languages #1752 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Fix areas order on local storage decache #1747 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Use the site binded to the group when setting a node role instead of the currently selected site #1738 (*ngilain*)
- **[open-orchestra-cms-bundle]** Clear the node per node rights when changing the binding between a group and a site #1737 (*ngilain*)
- **[open-orchestra-cms-bundle]** Fix some broken input #1723 (*ngilain*)
- **[open-orchestra-cms-bundle]** Fix wrong translation on workflow rights in group administration #1722 (*ngilain*)
- **[open-orchestra-cms-bundle]** a block only used in a single place can now be moved into another area without being deleted #1717 (*ngilain*)
- **[open-orchestra-cms-bundle]** hide add button if user d'ont have create content role #1716 (*djamnazi*)
- **[open-orchestra-cms-bundle]** Fix deactivate and re activate tinymce #1708 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Fix duplicate model group role when update on model group role #1704 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix IE9 padding of area in node and contribution in empty area #1702 (*alavieille*)
- **[open-orchestra-cms-bundle]** On the node edition page, the node is no more reloaded when clicking on the currently selected language tab #1697 (*ngilain*)
- **[open-orchestra-cms-bundle]** Fix submit and content type select #1695 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Fix refresh and submit configurable content block form #1693 (*alavieille*)
- **[open-orchestra-cms-bundle]** Throw an exception in UpdateNodeGroupRoleMoveNodeSubscriber if NodeGroupRole is not set #1692 (*ngilain*)
- **[open-orchestra-cms-bundle]** A form containing a required oo_content_search now correctly display an error message when trying to submit it without completing the oo_content_search #1691 (*ngilain*)
- **[open-orchestra-cms-bundle]** Fix Backoffice/Security/Authorization/Voter/NodeVersionVoter when create a new node #1689 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix use contentId instead id in ContentSearchSubscriber #1683 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix loop redirection when load template underscore without valid session #1682 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix link with media in tinymce, replace tinymce plugin link by plugin orchestra_link #1671 (*alavieille*)
- **[open-orchestra-cms-bundle]** catch js error on form submission #1670 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Fix french translations #1666 (*itkg-canne*)
- **[open-orchestra-display-bundle]** Fix display strategy configurable content with attribute contentSearch #220 (*alavieille*)
- **[open-orchestra-display-bundle]** Fix the Language List block when used in an error 404 page #218 (*ngilain*)
- **[open-orchestra-media-admin-bundle]** Fix alternative generation when original image ratio is extreme #246 (*ngilain*)

- **[open-orchestra-media-admin-bundle]** correct url for crop submit #245 (*itkg-nanne*)
- **[open-orchestra-media-admin-bundle]** fix no selected media on firefox #235 (*djamnazi*)
- **[open-orchestra-media-bundle]** custom styles are correctly used in displayMedia #204 (*Thiblef*)
- **[open-orchestra-media-bundle]** fix the document folder validator #199 (*djamnazi*)
- **[open-orchestra-model-bundle]** Fix the bad return type of SiteRepository::findByAliasDomain #585 (*ngilain*)
- **[open-orchestra-mongo-libs]** Fix requirement doctrine/mongodb-odm with php 5.5 #36 (*alavieille*)
- **[open-orchestra-workflow-function-bundle]** Fix authorisation check when several Workflow profiles exist with the same transition #117 (*ngilain*)
- **[open-orchestra-front-bundle]** allow overriding logical name in route generating by using route name #170 (*itkg-nanne*)

Possible BC breaker

- **[open-orchestra-cms-bundle]** Constructor of class `Backoffice/EventSubscriber/BlockTypeSubscriber` is changed #1768 (*alavieille*)
- **[open-orchestra-cms-bundle]** Constructor of class `Backoffice/Form/Type/BlockType` is changed #1768 (*alavieille*)
- ** The default parent type of the form strategies of block is `oo-block`

New features

- **[open-orchestra-model-bundle]** Add a NodeRepository::findOneByNodeAndSite method #591 (*ngilain*)

Other changes

- **[open-orchestra-workflow-function-bundle]** change some translation #120 (*itkg-nanne*)

CHANGELOG from 1.1.0 to 1.1.1

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle

- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Bug fixes

- **[open-orchestra-cms-bundle]** Add boLabel on root node when create a new website #1654 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix error index blocks when duplicate a node #1648 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix the javascript error on datatable pagination #1644 (*alavieille*)
- **[open-orchestra-cms-bundle]** maximun version for nodejs is ~4.4.3 #1641 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix dependency between BaseApi and ApiBundle, move TransformerParameterTypeException #1631 (*alavieille*)
- **[open-orchestra-cms-bundle]** fix tinymce issues in collection context #1629 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Refresh page when saving a website form after a form error #1627 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix block edition right management in global page #1620 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Fix default field in content type form #1616 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** use dependency injection for BlockToArrayTransformer in BlockType #1615 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** replace opacity by display in area toolbar to deactivate involuntary click #1614 (*itkg-nanne*)
- **[open-orchestra-media-admin-bundle]** Media references are set also in draft status and pending status of content and node #227 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Remove browse button if user doesn't have the access to the media management. #219 (*djamnazi*)
- **[open-orchestra-user-bundle]** Fix dependency between UserBundle and BackOfficeBundle #103 (*alavieille*)

New features

- **[open-orchestra-cms-bundle]** hide error 403 on redirection after post in muser context #1650 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Update css style of componement tab view replace class nav-tabs by nav-pills nav-justified #1628 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add a method to find role for node only. #1626 (*alavieille*)
- **[open-orchestra-cms-bundle]** Show node language and version language in title of edit form #1622 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** remove image resize in tinyMce #213 (*itkg-nanne*)
- **[open-orchestra-media-bundle]** add css style to media bbcode #192 (*itkg-nanne*)

- **[open-orchestra-model-bundle]** Update route in database after site migration and site-alias update #576 (*itkg-nanne*)

Deprecated

- **[open-orchestra-cms-bundle]** The `OpenOrchestra\ApiBundle\Exceptions\TransformerParameterTypeExce` class is deprecated since version 1.2.0 and will be removed in 1.3.0, use `OpenOrchestra\BaseApi\Exceptions\TransformerParameterTypeException` #1631 (*alavieille*)

CHANGELOG from 1.0 to 1.1

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Changelog 1.0.0 to 1.1.0-alpha1

New features

- It is now possible to configure the Back Office dashboard and to create new widgets for it.
- Routes stored in the database are now linked to the last published version of a node
- You can now add the author, last contributor and contribution date in the content listing

- 2 new events are availables in the block rendering process: PRE_BLOCK_CREATION and POST_BLOCK_CREATION
- It is now possible to filter a column which contains a type *date* in dataTable
- Transverse block are now created on all transverse nodes
- When you add a language to a website, the transverse node is created with all the transverse blocks

Bug fixes

- Each time you update a site, the routes in the database related to this site are updated
- Sorts by attributes in content and content type view are fully functional

Other changes

- Bundles are now using the PSR-4 syntax to be loaded, you should update the *Gruntfile* to follow this path modification

Deprecated method

- Some method from the *NodeRepositoryInterface* and *ContentRepositoryInterface* have been deprecated
- The method *findByNodeType* has been deprecated

Changelog 1.1.0-alpha1 to 1.1.0-alpha2

New features

- Replace the media upload GUI with a new component allowing multi-upload
- Adding roles for nodes (CREATE, UPDATE, MOVE, DELETE)
- Adding roles for content types (CREATE, UPDATE, DELETE)
- Adding roles for keywords (CREATE, UPDATE, DELETE)
- Adding roles for redirections (CREATE, UPDATE, DELETE)
- Adding roles for trashcan (RESTORE)
- Adding roles for api accesses (CREATE, UPDATE, DELETE)
- Adding roles for contents (CREATE, UPDATE, DELETE)
- Adding roles for medias (CREATE, UPDATE, DELETE)
- Adding roles for roles (CREATE, UPDATE, DELETE)
- Adding roles for sites (CREATE, UPDATE, DELETE)
- Adding roles for users (CREATE, UPDATE, DELETE)
- Adding roles for transverse nodes (UPDATE)
- Adding roles for workflow status (CREATE, UPDATE, DELETE)

- Adding roles for workflow functions (CREATE, UPDATE, DELETE)

Possible BC breaker

- The name of *OpenOrchestraBackofficeBundleFormTypeAreaType* is now *oo_area*
- The name of *OpenOrchestraBackofficeBundleFormTypeBlockChoiceType* is now *oo_block_choice*
- The name of *OpenOrchestraBackofficeBundleFormTypeBlockType* is now *oo_block*
- The name of *OpenOrchestraBackofficeBundleFormTypeExistingBlockChoiceType* is now *oo_existing_block*
- The name of *OpenOrchestraBackofficeBundleFormTypeNodeType* is now *oo_node*
- The name of *OpenOrchestraBackofficeBundleFormTypeApiClientType* is now *oo_api_client*
- The class *OpenOrchestraBackofficeBundleFormTypeAbstractOrchestraGroupType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeAbstractGroupChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraColorPickerType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeColorPickerType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraChoicesOption* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentChoicesOptionType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraContentTypeChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentContentTypeChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraDateWidgetOption* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentDateWidgetOptionType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraFieldChoice* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentFieldChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraKeywordsType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentKeywordsChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraNodeChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentNodeChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraRoleChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentRoleChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraSiteChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentSiteChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraThemeChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentThemeChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraVideoType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeVideoType*
- The name of *OpenOrchestraBackofficeBundleFormTypeContentType* is now *oo_content*
- The name of *OpenOrchestraBackofficeBundleFormTypeContentTypeType* is now *oo_cont ent_type*
- The name of *OpenOrchestraBackofficeBundleFormTypeFieldOptionType* is now *oo_field_option*
- The name of *OpenOrchestraBackofficeBundleFormTypeFieldTypeType* is now *oo_field_type*
- The name of *OpenOrchestraBackofficeBundleFormTypeGroupType* is now *oo_group*
- The name of *OpenOrchestraBackofficeBundleFormTypeKeywordType* is now *oo_keyword*
- The name of *OpenOrchestraBackofficeBundleFormTypeRedirectionType* is now *oo_redirection*

- The name of *OpenOrchestraBackofficeBundleFormTypeRoleType* is now *oo_role*
- The name of *OpenOrchestraBackofficeBundleFormTypeSiteAliasType* is now *oo_site_alias*
- The name of *OpenOrchestraBackofficeBundleFormTypeSiteType* is now *oo_site*
- The name of *OpenOrchestraBackofficeBundleFormTypeStatusType* is now *oo_status*
- The name of *OpenOrchestraBackofficeBundleFormTypeTemplateType* is now *oo_template*
- The name of *OpenOrchestraBackofficeBundleFormTypeThemeType* is now *oo_theme*
- The name of *OpenOrchestraBackofficeBundleFormTypeTinymceType* is now *oo_tinymce*
- The name of *OpenOrchestraBackofficeBundleFormTypeTranslatedValueCollectionType* is now *oo_translated_value_collection*
- The name of *OpenOrchestraBackofficeBundleFormTypeTranslatedValueType* is now *oo_translated_value*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraGroupType* is deleted and replaced by *OpenOrchestraGroupBundleFormTypeGroupDocumentType*
- The name of *OpenOrchestraMediaAdminBundleFormTypeFolderType* is now *oo_folder*
- The name of *OpenOrchestraMediaAdminBundleFormTypeMediaCropType* is now *oo_media_crop*
- The name of *OpenOrchestraMediaAdminBundleFormTypeMediaMetaType* is now *oo_media_meta*
- The name of *OpenOrchestraMediaAdminBundleFormTypeMediaType* is now *oo_media*
- The class *OpenOrchestraMediaAdminBundleFormDataTransformerOrchestraMediaTransformer* is deleted and replaced by *OpenOrchestraMediaAdminBundleFormDataTransformerMediaChoiceTransformer*
- The class *OpenOrchestraMediaAdminBundleFormTypeOrchestraMediaType* is deleted and replaced by *OpenOrchestraMediaAdminBundleFormTypeComponentMediaChoiceType*
- The class *OpenOrchestraMediaAdminBundleFormTypeOrchestraSiteForFolderChoiceType* is deleted and replaced by *OpenOrchestraMediaAdminBundleFormTypeSiteForFolderChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraRoleType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeWorkflowRoleChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraSiteType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeGroupSiteChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraStatusType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeStatusChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraThemeType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeSiteThemeChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraRoleType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractWorkflowRoleChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraSiteType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractGroupSiteChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraStatusType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractStatusChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraThemeType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractSiteThemeChoiceType*
- The name of *OpenOrchestraAdminBundleFormTypeRegistrationUserType* is now *oo_registration_user*
- The name of *OpenOrchestraAdminBundleFormTypeUserType* is now *oo_user*

- The name of *OpenOrchestraUserBundleFormTypeChangePasswordUserType* is now *oo_user_change_password*
- The class *OpenOrchestraWorkflowFunctionAdminBundleFormTypeAuthorizationType* is deleted and replaced by *OpenOrchestraWorkflowFunctionAdminBundleFormTypeComponentAuthorizationType*
- The class *OpenOrchestraWorkflowFunctionAdminBundleFormTypeOrchestraWorkflowFunctionType* is deleted and replaced by *OpenOrchestraWorkflowFunctionAdminBundleFormTypeComponent-WorkflowFunctionChoiceType*
- The name of *OpenOrchestraWorkflowFunctionAdminBundleFormTypeWorkflowRightType* is now *oo_workflow_right*
- The name of *OpenOrchestraWorkflowFunctionAdminBundleFormTypeWorkflowFunctionType* is now *oo_workflow_function*

Bug fixes

- User is now able to delete a media folder when the last media is deleted, without having to refresh the page.
- When creating a new media folder, menu is now automatically refreshed.

Other changes

- In differents dataTable, the global search is disabled. To reactivate it, you can use the data attribute `display-global-search=true` in the link in navigation panel.
- Every repository that should be paginated are now implementing *OpenOrchestraPaginationConfigurationPaginationRepositoryInterface*
- The version of Symfony is updated to 2.7.6
- Module php5-ffmpeg is replaced by PHP driver PHP-FFMpeg

Deprecated method

- The method *findByAuthor* has been deprecated in both NodeRepository and ContentRepository
- The class *OpenOrchestraModelInterfaceRepositoryPaginateRepositoryInterface* has been replaced by *OpenOrchestraPaginationConfigurationPaginationRepositoryInterface*

Changelog 1.1.0-alpha2 to 1.1.0-alpha3

Possible BC breaker

- Plugin `Colvis` of DataTable is replaced by the `Buttons extension`
- `UploadedFileManager` is moved from `MediaBundle` to `MediaFileBundle`
- Class of `BBcode` folder are moved from `MediaBundle` folder to `Media` folder
- Service `open_orchestra_media.manager.uploaded_media` is renamed by `open_orchestra_media_file.manager.storage`

- Application `open-orchestra-media-demo` no longer requires this bundles : - `DoctrineMongoDBBundle` - `OpenOrchestraBaseBundle` - `OpenOrchestraMediaModelBundle` - `OpenOrchestraMongoBundle` - `OpenOrchestraModelBundle` - `SolutionMongoAggregationBundle`
- Application `open-orchestra-media-demo` requires *MediaFileBundle'*
- Route `open_orchestra_media_get` is now in `MediaController` in `MediaFileBundle`
- Class moved from `MediaBundle` to `MediaAdminBundle` : - `FolderEvent` - `ImagickEvent` - `MediaEvent` - *DeleteMediaSubscriber'* - `GenerateImageSubscriber` - `UploadImageSubscriber` - `FolderEvents` - `OrchestraImagick` - `OrchestraImagickFactory` - `OrchestraImagickFactoryInterface` - `OrchestraImagickInterface` - `ImageResizerManager` - `MediaEvents` - `ImageToThumbnailManager` - `PdfToImageManager` - `VideoToImageManager` - `ThumbnailInterface` - `ThumbnailManager` - `FFmpegVideoManager` - `VideoManagerInterface` - `SaveMediaManager` - `SaveMediaManagerInterface`

New features

- Possibility to add custom fields for search in the DataTable, further information in documentation
- New Backbone view generic `DataTableView` for create a list with DataTable, used by `TableViewCollection`
- The build of the full project has been moved to the travis container build
- Every facades returned by the API are now configurable and can be defined in bundles configuration, further information in documentation
- The *RoleCollector* has been split into a collector for the front office and the backoffice
- To prevent *Status* suppression, the *StatusUsageFinder* has been created to check in every document using *Status* if they are using it
- Install smartadmin Datepicker
- Datatable preferences saving
- Add embedded entity to content attributs
- New bundle MediaFileBundle used to managed media files with Gaufrette
- Adding of upload stategies to manage all alternatives of media like thumbnail, futher information in the documentation

Bug fixes

- The node drag'n'drop has been updated to wait for user's confirmation before sending datas
- Fix errors on template creating
- Clean redirection and route on unpublished node
- Fixes on Dashboard, action buttons, order ...

Other changes

- DataTable is updated to version 1.1.0. In your *bower.json* file replace lines :

```
 "datatables": "~1.10.2",
 "datatables-colvis": "~1.1.2",
 "datatables-bootstrap3": "*",

by
```

```
"datatables.net-buttons-bs": "1.1.0",
```

- Version of doctrine/cache is fixed to 1.5.*
- Version of doctrine/commmon is fixed to 2.5.*

Deprecated method

- *NodeGroupRoleVoter* has been moved in the *Backoffice* folder
- *GroupSiteVoter* has been moved in the *Backoffice* folder
- The *AuthorizeEditionManager* has been deprecated, and all strategies has been transformed has roles
- The *VersionableSaver* has been moved in the *Saver* folder
- The role constant *ROLE_ACCESS_GENERAL_NODE* has been replaced by *ROLE_ACCESS_TREE_GENERAL_NODE*

Changelog 1.1.0-alpha3 to 1.1.0-alpha4

New features

- You can specify your bower and npm dependencies directly in the composer.json, futher information in the assets documentation
- Selector version of node tranverse is removed
- Duplicate button of node tranverse is removed
- Delete button of node tranverse is removed
- An *Elastica* implementation has been created to index all the *content* in an indexor

Bug fixes

- The Back Office is compatible again with Internet Explorer 9. See Configurations changes for more info
- Fix a bug when a site declared in Apache conf but not in Orchestra was accessed by a client

Possible BC breaker

- `NodeGroupRoleTransformer` implements now `TransformerWithGroupInterface`
- Method `reverseTransform` of `NodeGroupRoleTransformer` is removed and replaced by `reverseTransformWithGroup`
- Before `composer install|update` it's recommanded to removed `nodes_modules` and `bower_components` folders of your application and `bower.json` and `package.json` files. Remove also symlink in `web/fonts`
- You can move your bower and npm dependencies in `composer.json` of your project or bundle
- Datatable configuration have been moved from the *data* tag to a *yaml* configuration
- *Grunt* configuration has been modified to be more handled by the bundles

Other changes

- PHP requirement has been updated to 5.5
- Version of `symfony/symfony` is updated to 2.8.1
- Version of `twig/twig` is updated to ~1.23
- Version of `friendsofsymfony/http-cache-bundle` is updated to 1.3.6
- Flow.js dependency has been moved from CmsBundle to MediaAdminBundle
- Media alternatives files (mainly images) have new auto-generated names
- Deprecation of the *ContainerAware* class have been suppressed
- Nodes can have a new options for the theme : use the website default theme
- Abstract test classes have been created to help free the memory used in the tests

Configuration changes

- `pace.js` is removed and replace by a custom component `OpenOrchestra.AjaxLoader.AjaxLoaderView`
- To get the Back Office compatible again with Internet Explorer 9, some grunt tasks have been rewrote. Be sure to update your Open Orchestra tasks according to these Pull Request
- `gridstack.js` is removed
- `lodash.js` is removed
- Npm package `grunt-js-test` is added to composer.json of `CmsBundle`

Changelog 1.1.0-alpha4 to 1.1.0-beta

Configuration changes

- Adding bundle `MongoDBMigrationsBundle` #853
- Dependencies versions are fixed in cms bundle: `merge ~1.2.0`, `backbone ~1.2.3`, `jquery-minicolors ~2.2.3`, `backbone.wreqr ~1.3.5`. #1397

- Bower dependency `underscore` is removed, this package is included by `backbone` #1397
- Dependencies versions are fixed in admin bundle: `merge ~1.2.0`, `backbone ~1.2.3`, `backbone.wreqr ~1.3.5`. #136
- Add migration script for update of user groups collection, futher documation about migration in http://open-orchestra.readthedocs.org/en/latest/hosting_guide/migration.html #531

Bug fixes

- Refresh template when edit template form is submitted #1534
- fix error on role delete #1530
- Fix error 404 in the form keyword choice type when you create a new keyword #1520
- The children direction now can be specified in nodes #1519
- Fix translation labels of defaultListable field in content type form #1509
- The event `ContentTypeEvents::CONTENT_TYPE_UPDATE` is dispatched only when content type form is valid #1505
- fix the display of the media block icon #1502
- Fix TinyMCE content display after form submission #1500
- disallow roles with same from and to status #1498
- Fix searchable option of content type field #1491
- Make the color picker working in all contexts #1488
- Fix css ie11 template and node contribution #1486
- fix indent after hide preview button on global pages #1484
- Fix prototype bug #1482
- Refresh navigation menu after a new element has been created in a list #1479
- Don't display the transverse blocks in the blocks panel when editing a transverse node #1474
- Field choice with option multiple disabled #1472
- hide the preview button on global pages (transverse) #1471
- Fix custom field selection on content type creation #1460
- In navigation panel, a root menu is hidden if these submenus aren't visible #1459
- Remove native media option in tinymce context menu #1455
- Nodes with dynamic route pattern can't be added in menus #1450
- Published nodes can be moved #1443
- Fix several bugs for IE9 in the Back Office #1427
- Condition transformer does not use constructor anymore but setter #1422
- Datepicker in content forms #1406
- Suppress multiple load in panel context #1404
- Pass the 'disabled' option to the block form strategy when required #1399
- Fix error 404 Jcrop.gif when a picture is cropped #1398

- Activate tinyMce in collection prototype context #1396
- Fix jarvis widget header style in BackOffice #1388
- disable load dataparemeter in modal context #1386
- When a node is duplicated there's no more node group role creation #1385
- Fix error type of property `updateAt` in content type schema #24
- Media references are correctly placed #186
- Fix breadcrumb form in media modal #182
- Fix display media upload with short window #180
- Fix resize with picture ratio lower than 1 #178
- fix missed translation (open_orchestra_media_admin.block.display_media.title) #162
- A super admin user can view all sites when he creates a new form #160
- Fix media selection in tinyMce media-admin#158
- Media events are not properly dispatched #156
- Allow recursive merge on tinyMce Stfalcon parameters #142
- Status of contents containing a media field can now be changed #138
- Remove media management in blocks when form is disabled #137
- Fix access denied error on published node access #524
- Fix broken varnish 4 vcl #29
- As the pixel developer google chrome is not more accessible, I have suppressed the selenium role from our provisioning #27
- Fix back-to-list buttons on top in modal upload media-admin#172

New features

- Block menu caches are invalidated when a node is removed, moved, restore and the status of a node changes . #1531
- add voter on role usage #1530
- increase ergo fo group form #1522
- When a content type is deleted, navigation menu is refreshed #1521
- When a website is created, an homepage for this site is also created. #1518
- refresh the page after website creation to show it in the fixed top nav list #1515
- fix save & back-to-list buttons on top #1499
- Move template menu entry from Editorial to Administration #1485
- allow to extend js form's behavior #1473
- add loader in group tab view #1467
- change new link to button #1466
- Add sortable option to the datatable #1451

- add boolean expression in keywords form type filter #1442
- Remove max length option on email field type #1439
- Datatable action buttons are disabled if the user hasn't the rights #1438
- add notification colors #1433
- Media roles gestion for each folders are available in the group panel #1416
- Allow published node deletion #1405
- add break line in tooltip helper (n) #1400
- allow published node delete #1384
- auto-select type of search in datatable for content field type #1382
- activate content list block in front display#202
- Node cache-control policy differs with or without ESI support #193
- allow end user to format boolean condition in keywords filter #41
- Media can be filtered on type when displayed from media folder #179
- Media library: applications can now describe their own specific alternatives image formats #144
- Add a "Back to list" button on the media upload form in a modal context #133
- Media selection integrates now the alternative selector #132
- Published flag feature #529

Deprecated

- `initializeNewNode` is deprecated and replaced by `initializeNode` in class `OpenOrchestra\Backoffice\Manager\NodeManager`. this method will be removed in 1.2.0 #1518
- rename class `UpdateNodeRedirectionSubscriber` to `UpdateRedirectionNodeSubscriber` #1503
- `OpenOrchestra\Backoffice\Form\DataTransformer\ChoiceArrayToStringTransformer` is now deprecated will be removed in 1.2.0 #1472
- `OpenOrchestra\Backoffice\Form\DataTransformer\ChoiceStringToArrayTransformer` is now deprecated will be removed in 1.2.0 #1472
- `FolderRepository:findAllRootFolder` method in `OpenOrchestra\MediaModelBundle\Repository` namespace has been deprecated use `findAllRootFolderBySiteId` #177
- Update the media twig functions #168

Possible BC breaker

- include string indexation of site alias in error pages path. #1503
- `DisplayBundle/DisplayBlock/Strategies/AbstractStrategy:getCacheTags()` is now abstract and must therefore be explicitally implemented on each display strategy #1457
- Move form from backofficeBundle #1425
- Move Model and repository from BackofficeBundle #1424

- Move BackofficeBundle manager #1423
- Move BackofficeBundle subscribers #1422
- Move BackofficeBundle listener #1421
- Move Backoffice display block folder #1417
- Move Backoffice display icon folder #1417
- Move Backoffice initializer folder #1417
- change way to load dataparameter in refresh menu context #1386
- DisplayBundle/DisplayBlock/Strategies/AbstractStrategy:getCacheTags() is now abstract and must therefore be explicitally implemented on each display strategy display#200
- Disable multi website in media folder #185
- Fixtures for production are cleaned #530
- In the front, the nodes should have the published flag to be displayed #529

Other changes

- New Bundle : MediaAdminModelBundle #831
- Upgrade to symfony 2.8.3 #72
- Role `ROLE_ACCESS_MOVE_NODE` is renamed to `ROLE_ACCESS_MOVE_TREE` and is now a global role #1480
- Deleted nodes order is -1 #1440
- Redirect to the user edit form after user creation #1408
- Refacto const display strategies name #1407
- Replace strings `elastica_search` and `elastica_list` by class constant #21
- Get master request in method where it is used and not in a constructor #150
- Add the robots meta in the page header FrontBundle#138
- Unskip a MediaStorageManager unit test #8
- Node order validator don't checks deleted nodes #526
- Add findByNodeAndSiteSortedByVersion() request in NodeRepository #514
- Suppression of install of nodeJS by the provisioning, it is now installed by `composer-extra-assets`. use now `./bin/grunt` for grunt and not `./node_modules/.bin/grunt` #34

Changelog 1.1.0-beta to 1.1.0-RC

New features

- Test on mandatory main alias in site form #1542
- Remove permanently an entity in trash can #1539
- Enhance area form in template, node and area #1537

- When a content is deleted, it's removed from elastica index #25
- When a content is restored, it's added from elastica index #25
- A media root folder is created when a new website is created #190

Bug fixes

- Fix position of property *validationGroups* in method *isValid* of *BaseApiBundle/Controller/BaseController* #76
- Fix transform area from another site as the current site #1544
- Unused blocks are now definitely suppressed from DB when deleted from a node and used no more #1540
- Fix error type of property `updateAt` in content type schema #24
- A required oo_media_choice form type is now correctly handled #195
- Remove link constraint on media display block #194
- Method `testUniquenessInContext` is now based on `nodeId` and not `name` #543
- Demo fixtures are updated to present block usage references #542
- Path on node is updated only if node is not deleted #541

Other changes

- Get master request in method where it is used and not in a constructor #150

Possible BC breaker

- The *TrashItem* document has a new property *type*
- The *TrashItemInterface* has a new property *type*
- The *TrashItemRepositoryInterface* has a new method *findByEntity($entityId)*

Manual changes

- The *BlockContainerInterface* has a new method: *removeBlockWithKey*

Changelog 1.1.0-RC to 1.1.0

New features

- **[open-orchestra]** updating fixtures console command to allow user choose the type of fixtures to load and add a dev environment #888 (*djamnazi*)
- **[open-orchestra-base-bundle]** Update requirement `symfony/symfony` to `~2.8.4` #81 (*alavieille*)
- **[open-orchestra-cms-bundle]** It is no longer possible to delete a block of global page if it is used #1599 (*alavieille*)

- **[open-orchestra-cms-bundle]** Adding attribute `boLabel` on Node , used to generate tree in the Back Office #1586 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove delete button on edit modal view and deplace it #near edit button #1581 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Remove choice groups in edit user if the user is super admin #1572 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove role super admin of user admin in production fixture #1572 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove edit button for tranverse page #1569 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** When a content is deleted or his status is updated tag `'contentId-' . $contentId` is invalidate. #1568 (*alavieille*)
- **[open-orchestra-cms-bundle]** When a content type is deleted or updated tag `''contentType-' . $contentType` is invalidate. #1568 (*alavieille*)
- **[open-orchestra-cms-bundle]** `openOrchestra\Backoffice\Form\Type\ComponentRoleChoiceType` can display roles of multiple role collector #1567 (*alavieille*)
- **[open-orchestra-cms-bundle]** In navigation panel, The global page is directly accessible. #1561 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add modification date for versions into global page and node. #1559 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Remove full screen for jarvis widget #1558 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** The role parameter to create an navigation strategy is no longer required #1556 (*alavieille*)
- **[open-orchestra-cms-bundle]** Adding voter to forbid removing node root #1555 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add role `acess`, `update`, `create` for error nodes. #1549 (*alavieille*)
- **[open-orchestra-cms-bundle]** add content search widget in configurable content block #1548 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Move actions buttons to the ribbon #1545 (*djamnazi*)
- **[open-orchestra-cms-bundle]** test on mandatory main alias in site form #1542 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Remove permanently a entity in trash can #1539 (*alavieille*)
- **[open-orchestra-cms-bundle]** enhance area form in template, node and area #1537 (*itkg-nanne*)
- **[open-orchestra-elastica-bundle]** When a content is deleted, it's removed to elastica index #25 (*alavieille*)
- **[open-orchestra-elastica-bundle]** When a content is restored, it's added to elastica index #25 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Message in form `oo_media_choice` when no image is selected is translated #202 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** A root folder is created when a new website is created #190 (*alavieille*)
- **[open-orchestra-media-bundle]** Add default strategy to display media #183 (*alavieille*)
- **[open-orchestra-media-file-bundle]** Add expires attribute into the header of HTTP request. #14 (*KevinArtus*)
- **[open-orchestra-model-bundle]** Add script migration to rename role ROLE_ACCESS_MOVE_NODE by ROLE_ACCESS_MOVE_TREE #563 (*alavieille*)
- **[open-orchestra-model-bundle]** Add migration for attribute `boLabel` #561 (*alavieille*)
- **[open-orchestra-model-bundle]** Create HomePage by load fixtures #549 (*djamnazi*)
- **[open-orchestra-workflow-function-bundle]** Remove link right form in edit user if the user is super admin #99 (*alavieille*)

- **[open-orchestra-workflow-function-bundle]** Add voter to check workflow state of statusable document #97 (*alavieille*)

Possible BC breaker

- **[open-orchestra-cms-bundle]** `OpenOrchestra\GroupBundle\EventListener\AbstractNodeGroupRoleListe` is replaced by `OpenOrchestra\GroupBundle\EventSubscriber\AbstractNodeGroupRoleSubscriber` #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** `OpenOrchestra\GroupBundle\EventListener\AddNodeGroupRoleForGroupLi` is replaced by `OpenOrchestra\GroupBundle\EventSubscriber\NodeGroupRoleForGroupSubscriber` #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** `OpenOrchestra\GroupBundle\EventListener\AddNodeGroupRoleForNodeLis` is replaced by `OpenOrchestra\GroupBundle\EventSubscriber\NodeGroupRoleForNodeSubscriber` #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** Deleted PageConfigurationButtonView.coffee #1581 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** `` OpenOrchestraUserAdminBundleEventUserFacadeEvent`` take a new argument *user* which is the user transformed #1572 (*alavieille*)
- **[open-orchestra-cms-bundle]** `openOrchestra\Backoffice\Form\Type\ComponentRoleChoiceType` take an array of role collector in parameter. #1567 (*alavieille*)
- **[open-orchestra-cms-bundle]** `OpenOrchestra\Backoffice\NavigationPanel\Strategies\GeneralNodesPa` is renamed by `OpenOrchestra\Backoffice\NavigationPanel\Strategies\TransverseNodePanelStrate` #1561 (*alavieille*)
- **[open-orchestra-cms-bundle]** The class `OpenOrchestra\Backoffice\NavigationPanel\Strategies\Abstrac` is deprecated and will be removed in 1.2.0 use `OpenOrchestra\Backoffice\NavigationPanel\Strategies\Abst` #1556 (*alavieille*)
- **[open-orchestra-media-bundle]** Twig of method displayMedia in all display strategy are moved of `OpenOrchestraMediaBundle:BBcode/FullDisplay` to `OpenOrchestraMediaBundle:DisplayMedia/FullDisplay` #183 (*alavieille*)
- **[open-orchestra-model-bundle]** The *TrashItem* document has a new property *type* #541 (*alavieille*)
- **[open-orchestra-model-interface]** The *TrashItemInterface* has a new property *type* #172 (*alavieille*)
- **[open-orchestra-model-interface]** The *TrashItemRepositoryInterface* has a new method *findByEntity($entityId)* #172 (*alavieille*)

Bug fixes

- **[open-orchestra-base-api-bundle]** Fix position of property *validationGroups* in method *isValid* of *BaseApiBundle/Controller/BaseController* #76 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix error on field default value if field type is changed in content type form #1602 (*alavieille*)
- **[open-orchestra-cms-bundle]** When you save multiple tinymce, there are saved correctly #1601 (*alavieille*)
- **[open-orchestra-cms-bundle]** create user with media folder access and create role for functional test #1595 (*djamnazi*)
- **[open-orchestra-cms-bundle]** Fix access denied when edit node without `role_access_site` #1593 (*alavieille*)

- **[open-orchestra-cms-bundle]** Fix error access denied when select an other version node #1592 (*alavieille*)
- **[open-orchestra-cms-bundle]** Refresh node view when edit form of node is submitted #1591 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix bug updating status of node with the update role of node of other type #1590 (*alavieille*)
- **[open-orchestra-cms-bundle]** add constraints for email user unicity #1587 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Add label option into group FormType #1585 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Activate subform content search in content type form context #1583 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** When a group is updated model group roles are correctly set #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add constraint Unique on group #1565 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix transform correctly a type double in a string value. #1564 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove cursor move of dashboard widgets #1563 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix transform area from another site as the current site #1544 (*alavieille*)
- **[open-orchestra-cms-bundle]** Unused blocks are now definitivly suppressed from DB when deleted from a node and used no more #1540 (*itkg-ngilain*)
- **[open-orchestra-display-bundle]** set shared max age if ESI cash is supported #209 (*djamnazi*)
- **[open-orchestra-elastica-bundle]** Fix error type of property `updateAt` in content type schema #24 (*alavieille*)
- **[open-orchestra-front-bundle]** Fix generate url command site map generate #158 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** fix folder update error when user don't has the update role #208 (*djamnazi*)
- **[open-orchestra-media-admin-bundle]** In a content, the selection of a media with no alternative is now correctly stored #207 (*itkg-ngilain*)
- **[open-orchestra-media-admin-bundle]** Rename descriptions for roles media and folders. #206 (*KevinArtus*)
- **[open-orchestra-media-admin-bundle]** hide edit button if user don't have update role in media folder #204 (*djamnazi*)
- **[open-orchestra-media-admin-bundle]** If an user hasn't role `ROLE_ACCESS_UPDATE_MEDIA`, he can't edit a media #201 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** A required oo_media_choice form type is now correctly handled #195 (*itkg-ngilain*)
- **[open-orchestra-media-admin-bundle]** Remove link constraint on media display block #194 (*itkg-ngilain*)
- **[open-orchestra-model-bundle]** Re-activate site search by alias domain #560 (*itkg-nanne*)
- **[open-orchestra-model-bundle]** use currentlyPublished tag to display menu and footer #546 (*itkg-nanne*)
- **[open-orchestra-model-bundle]** method `hasOtherNodeWithSameParentAndOrder```of ``NodeRepository` check only on default nodes #545 (*alavieille*)
- **[open-orchestra-model-bundle]** method `testUniquenessInContext` is now based on `nodeId` and not `name` #543 (*alavieille*)
- **[open-orchestra-model-bundle]** Demo fixtures are updated to present block usage references #542 (*itkg-ngilain*)
- **[open-orchestra-model-bundle]** Path on node is updated only if node isn't deleted #541 (*alavieille*)

- **[open-orchestra-model-interface]** Fix type `nodeId` in php doc of `NodeInterface`` and ``ReadNodeInterface` #176 (*alavieille*)

Other changes

- **[open-orchestra-cms-bundle]** Corrected translation. #1600 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Rename navigation menu status and Role #1589 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Hide addpage button if no root page #1560 (*djamnazi*)
- **[open-orchestra-front-bundle]** Get master request in method where it is used and not in a constructor #150 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Migration scripts to explain in the changelog (main configuration changes + migration configuration) #200 (*itkg-ngilain*)
- **[open-orchestra-media-admin-bundle]** Remove MediaInterface::MEDIA_PREFIX #198 (*itkg-ngilain*)
- **[open-orchestra-model-bundle]** Remove "published to draft' role for production fixtures. #548 (*KevinArtus*)
- **[open-orchestra-workflow-function-bundle]** corrected translations. #104 (*KevinArtus*)
- **[open-orchestra-workflow-function-bundle]** Rename navigation menu workflow #102 (*KevinArtus*)
- **[open-orchestra-workflow-function-bundle]** Remove "published to draft' role for validator on fixtures. #95 (*KevinArtus*)

Deprecated

- **[open-orchestra-cms-bundle]** class `OpenOrchestra\Backoffice\EventSubscriber\ChangeContentStatusSu`
  s deprecated in 1.1.0 and will be removed in 1.2.0, it is replace by ContentUpdateCacheSubscriber #1568 (*alavieille*)

CHANGELOG from 1.1.0-RC to 1.1.0

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle

- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

New features

- **[open-orchestra]** updating fixtures console command to allow user choose the type of fixtures to load and add a dev environment #888 (*djamnazi*)
- **[open-orchestra-base-bundle]** Update requirement `symfony/symfony` to `~2.8.4` #81 (*alavieille*)
- **[open-orchestra-cms-bundle]** It is no longer possible to delete a block of global page if it is used #1599 (*alavieille*)
- **[open-orchestra-cms-bundle]** Adding attribute `boLabel` on Node , used to generate tree in the Back Office #1586 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove delete button on edit modal view and deplace it #near edit button #1581 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Remove choice groups in edit user if the user is super admin #1572 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove role super admin of user admin in production fixture #1572 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove edit button for tranverse page #1569 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** When a content is deleted or his status is updated tag `'contentId-' . $contentId` is invalidate. #1568 (*alavieille*)
- **[open-orchestra-cms-bundle]** When a content type is deleted or updated tag `''contentType-' . $contentType` is invalidate. #1568 (*alavieille*)
- **[open-orchestra-cms-bundle]** `openOrchestra\Backoffice\Form\Type\ComponentRoleChoiceType` can display roles of multiple role collector #1567 (*alavieille*)
- **[open-orchestra-cms-bundle]** In navigation panel, The global page is directly accessible. #1561 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add modification date for versions into global page and node. #1559 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Remove full screen for jarvis widget #1558 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** The role parameter to create an navigation strategy is no longer required #1556 (*alavieille*)
- **[open-orchestra-cms-bundle]** Adding voter to forbid removing node root #1555 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add role `acess`, `update`, `create` for error nodes. #1549 (*alavieille*)
- **[open-orchestra-cms-bundle]** add content search widget in configurable content block #1548 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Move actions buttons to the ribbon #1545 (*djamnazi*)
- **[open-orchestra-cms-bundle]** test on mandatory main alias in site form #1542 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Remove permanently a entity in trash can #1539 (*alavieille*)
- **[open-orchestra-cms-bundle]** enhance area form in template, node and area #1537 (*itkg-nanne*)

- **[open-orchestra-elastica-bundle]** When a content is deleted, it's removed to elastica index #25 (*alavieille*)
- **[open-orchestra-elastica-bundle]** When a content is restored, it's added to elastica index #25 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Message in form `oo_media_choice` when no image is selected is translated #202 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** A root folder is created when a new website is created #190 (*alavieille*)
- **[open-orchestra-media-bundle]** Add default strategy to display media #183 (*alavieille*)
- **[open-orchestra-media-file-bundle]** Add expires attribute into the header of HTTP request. #14 (*KevinArtus*)
- **[open-orchestra-model-bundle]** Add script migration to rename role ROLE_ACCESS_MOVE_NODE by ROLE_ACCESS_MOVE_TREE #563 (*alavieille*)
- **[open-orchestra-model-bundle]** Add migration for attribute `boLabel` #561 (*alavieille*)
- **[open-orchestra-model-bundle]** Create HomePage by load fixtures #549 (*djamnazi*)
- **[open-orchestra-workflow-function-bundle]** Remove link right form in edit user if the user is super admin #99 (*alavieille*)
- **[open-orchestra-workflow-function-bundle]** Add voter to check workflow state of statusable document #97 (*alavieille*)

Possible BC breaker

- **[open-orchestra-cms-bundle]** `OpenOrchestra\GroupBundle\EventListener\AbstractNodeGroupRoleListe` is replaced by `OpenOrchestra\GroupBundle\EventSubscriber\AbstractNodeGroupRoleSubscriber` #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** `OpenOrchestra\GroupBundle\EventListener\AddNodeGroupRoleForGroupLi` is replaced by `OpenOrchestra\GroupBundle\EventSubscriber\NodeGroupRoleForGroupSubscriber` #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** `OpenOrchestra\GroupBundle\EventListener\AddNodeGroupRoleForNodeLis` is replaced by `OpenOrchestra\GroupBundle\EventSubscriber\NodeGroupRoleForNodeSubscriber` #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** Deleted PageConfigurationButtonView.coffee #1581 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** `` OpenOrchestraUserAdminBundleEventUserFacadeEvent`` take a new argument *user* which is the user transformed #1572 (*alavieille*)
- **[open-orchestra-cms-bundle]** `openOrchestra\Backoffice\Form\Type\ComponentRoleChoiceType` take an array of role collector in parameter. #1567 (*alavieille*)
- **[open-orchestra-cms-bundle]** `OpenOrchestra\Backoffice\NavigationPanel\Strategies\GeneralNodesPa` is renamed by `OpenOrchestra\Backoffice\NavigationPanel\Strategies\TransverseNodePanelStrate` #1561 (*alavieille*)
- **[open-orchestra-cms-bundle]** The class `OpenOrchestra\Backoffice\NavigationPanel\Strategies\Abstrac` is deprecated and will be removed in 1.2.0 use `OpenOrchestra\Backoffice\NavigationPanel\Strategies\Abst` #1556 (*alavieille*)
- **[open-orchestra-media-bundle]** Twig of method displayMedia in all display strategy are moved of `OpenOrchestraMediaBundle:BBcode/FullDisplay` to `OpenOrchestraMediaBundle:DisplayMedia/FullDisplay` #183 (*alavieille*)
- **[open-orchestra-model-bundle]** The *TrashItem* document has a new property *type* #541 (*alavieille*)
- **[open-orchestra-model-interface]** The *TrashItemInterface* has a new property *type* #172 (*alavieille*)

- **[open-orchestra-model-interface]** The *TrashItemRepositoryInterface* has a new method *findByEntity($entityId)* #172 (*alavieille*)

Bug fixes

- **[open-orchestra-base-api-bundle]** Fix position of property *validationGroups* in method *isValid* of *BaseApiBundle/Controller/BaseController* #76 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix error on field default value if field type is changed in content type form #1602 (*alavieille*)
- **[open-orchestra-cms-bundle]** When you save multiple tinymce, there are saved correctly #1601 (*alavieille*)
- **[open-orchestra-cms-bundle]** create user with media folder access and create role for functional test #1595 (*djamnazi*)
- **[open-orchestra-cms-bundle]** Fix access denied when edit node without `role_access_site` #1593 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix error access denied when select an other version node #1592 (*alavieille*)
- **[open-orchestra-cms-bundle]** Refresh node view when edit form of node is submitted #1591 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix bug updating status of node with the update role of node of other type #1590 (*alavieille*)
- **[open-orchestra-cms-bundle]** add constraints for email user unicity #1587 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** Add label option into group FormType #1585 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Activate subform content search in content type form context #1583 (*itkg-nanne*)
- **[open-orchestra-cms-bundle]** When a group is updated model group roles are correctly set #1582 (*alavieille*)
- **[open-orchestra-cms-bundle]** Add constraint Unique on group #1565 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix transform correctly a type double in a string value. #1564 (*alavieille*)
- **[open-orchestra-cms-bundle]** Remove cursor move of dashboard widgets #1563 (*alavieille*)
- **[open-orchestra-cms-bundle]** Fix transform area from another site as the current site #1544 (*alavieille*)
- **[open-orchestra-cms-bundle]** Unused blocks are now definitivly suppressed from DB when deleted from a node and used no more #1540 (*itkg-ngilain*)
- **[open-orchestra-display-bundle]** set shared max age if ESI cash is supported #209 (*djamnazi*)
- **[open-orchestra-elastica-bundle]** Fix error type of property `updateAt` in content type schema #24 (*alavieille*)
- **[open-orchestra-front-bundle]** Fix generate url command site map generate #158 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** fix folder update error when user don't has the update role #208 (*djamnazi*)
- **[open-orchestra-media-admin-bundle]** In a content, the selection of a media with no alternative is now correctly stored #207 (*itkg-ngilain*)
- **[open-orchestra-media-admin-bundle]** Rename descriptions for roles media and folders. #206 (*KevinArtus*)
- **[open-orchestra-media-admin-bundle]** hide edit button if user don't have update role in media folder #204 (*djamnazi*)
- **[open-orchestra-media-admin-bundle]** If an user hasn't role `ROLE_ACCESS_UPDATE_MEDIA`, he can't edit a media #201 (*alavieille*)

- **[open-orchestra-media-admin-bundle]** A required oo_media_choice form type is now correctly handled #195 (*itkg-ngilain*)
- **[open-orchestra-media-admin-bundle]** Remove link constraint on media display block #194 (*itkg-ngilain*)
- **[open-orchestra-model-bundle]** Re-activate site search by alias domain #560 (*itkg-nanne*)
- **[open-orchestra-model-bundle]** use currentlyPublished tag to display menu and footer #546 (*itkg-nanne*)
- **[open-orchestra-model-bundle]** method `hasOtherNodeWithSameParentAndOrder```of ``NodeRepository` check only on default nodes #545 (*alavieille*)
- **[open-orchestra-model-bundle]** method `testUniquenessInContext` is now based on `nodeId` and not `name` #543 (*alavieille*)
- **[open-orchestra-model-bundle]** Demo fixtures are updated to present block usage references #542 (*itkg-ngilain*)
- **[open-orchestra-model-bundle]** Path on node is updated only if node isn't deleted #541 (*alavieille*)
- **[open-orchestra-model-interface]** Fix type `nodeId` in php doc of `NodeInterface`` and ``ReadNodeInterface` #176 (*alavieille*)

Other changes

- **[open-orchestra-cms-bundle]** Corrected translation. #1600 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Rename navigation menu status and Role #1589 (*KevinArtus*)
- **[open-orchestra-cms-bundle]** Hide addpage button if no root page #1560 (*djamnazi*)
- **[open-orchestra-front-bundle]** Get master request in method where it is used and not in a constructor #150 (*alavieille*)
- **[open-orchestra-media-admin-bundle]** Migration scripts to explain in the changelog (main configuration changes + migration configuration) #200 (*itkg-ngilain*)
- **[open-orchestra-media-admin-bundle]** Remove MediaInterface::MEDIA_PREFIX #198 (*itkg-ngilain*)
- **[open-orchestra-model-bundle]** Remove "published to draft' role for production fixtures. #548 (*KevinArtus*)
- **[open-orchestra-workflow-function-bundle]** corrected translations. #104 (*KevinArtus*)
- **[open-orchestra-workflow-function-bundle]** Rename navigation menu workflow #102 (*KevinArtus*)
- **[open-orchestra-workflow-function-bundle]** Remove "published to draft' role for validator on fixtures. #95 (*KevinArtus*)

Deprecated

- **[open-orchestra-cms-bundle]** class `OpenOrchestra\Backoffice\EventSubscriber\ChangeContentStatusSu` s deprecated in 1.1.0 and will be removed in 1.2.0, it is replace by ContentUpdateCacheSubscriber #1568 (*alavieille*)

CHANGELOG from 1.1.0-beta to 1.1.0-RC

Url to see changes:

- Cms bundle
- Display bundle

- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

New features

- Test on mandatory main alias in site form #1542
- Remove permanently an entity in trash can #1539
- Enhance area form in template, node and area #1537
- When a content is deleted, it's removed from elastica index #25
- When a content is restored, it's added from elastica index #25
- A media root folder is created when a new website is created #190

Bug fixes

- Fix position of property *validationGroups* in method *isValid* of *BaseApiBundle/Controller/BaseController* #76
- Fix transform area from another site as the current site #1544
- Unused blocks are now definitely suppressed from DB when deleted from a node and used no more #1540
- Fix error type of property `updateAt` in content type schema #24
- A required oo_media_choice form type is now correctly handled #195
- Remove link constraint on media display block #194
- Method `testUniquenessInContext` is now based on `nodeId` and not `name` #543
- Demo fixtures are updated to present block usage references #542
- Path on node is updated only if node is not deleted #541

Other changes

- Get master request in method where it is used and not in a constructor #150

Possible BC breaker

- The *TrashItem* document has a new property *type* #541
- The *TrashItemInterface* has a new property *type* #172
- The *TrashItemRepositoryInterface* has a new method *findByEntity($entityId)* #172

Manual changes

- The *BlockContainerInterface* has a new method: *removeBlockWithKey* #173

CHANGELOG for 1.1.0-beta

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Configuration changes

- Adding bundle `MongoDBMigrationsBundle` #853
- Dependencies versions are fixed in cms bundle: `merge ~1.2.0`, `backbone ~1.2.3`, `jquery-minicolors ~2.2.3`, `backbone.wreqr ~1.3.5`. #1397
- Bower dependency `underscore` is removed, this package is included by `backbone` #1397
- Dependencies versions are fixed in admin bundle: `merge ~1.2.0`, `backbone ~1.2.3`, `backbone.wreqr ~1.3.5`. #136
- Add migration script for update of user groups collection, futher documation about migration in http://open-orchestra.readthedocs.org/en/latest/hosting_guide/migration.html #531

Bug fixes

- Refresh template when edit template form is submitted #1534
- fix error on role delete #1530
- Fix error 404 in the form keyword choice type when you create a new keyword #1520
- The children direction now can be specified in nodes #1519
- Fix translation labels of defaultListable field in content type form #1509
- The event `ContentTypeEvents::CONTENT_TYPE_UPDATE` is dispatched only when content type form is valid #1505
- fix the display of the media block icon #1502
- Fix TinyMCE content display after form submission #1500
- disallow roles with same from and to status #1498
- Fix searchable option of content type field #1491
- Make the color picker working in all contexts #1488
- Fix css ie11 template and node contribution #1486
- fix indent after hide preview button on global pages #1484
- Fix prototype bug #1482
- Refresh navigation menu after a new element has been created in a list #1479
- Don't display the transverse blocks in the blocks panel when editing a transverse node #1474
- Field choice with option multiple disabled #1472
- hide the preview button on global pages (transverse) #1471
- Fix custom field selection on content type creation #1460
- In navigation panel, a root menu is hidden if these submenus aren't visible #1459
- Remove native media option in tinymce context menu #1455
- Nodes with dynamic route pattern can't be added in menus #1450
- Published nodes can be moved #1443
- Fix several bugs for IE9 in the Back Office #1427
- Condition transformer does not use constructor anymore but setter #1422

- Datepicker in content forms #1406
- Suppress multiple load in panel context #1404
- Pass the 'disabled' option to the block form strategy when required #1399
- Fix error 404 Jcrop.gif when a picture is cropped #1398
- Activate tinyMce in collection prototype context #1396
- Fix jarvis widget header style in BackOffice #1388
- disable load dataparemeter in modal context #1386
- When a node is duplicated there's no more node group role creation #1385
- Fix error type of property `updateAt` in content type schema #24
- Media references are correctly placed #186
- Fix breadcrumb form in media modal #182
- Fix display media upload with short window #180
- Fix resize with picture ratio lower than 1 #178
- fix missed translation (open_orchestra_media_admin.block.display_media.title) #162
- A super admin user can view all sites when he creates a new form #160
- Fix media selection in tinyMce #158
- Media events are not properly dispatched #156
- Allow recursive merge on tinyMce Stfalcon parameters #142
- Status of contents containing a media field can now be changed #138
- Remove media management in blocks when form is disabled #137
- Fix access denied error on published node access #524
- Fix broken varnish 4 vcl #29
- As the pixel developer google chrome is not more accessible, I have suppressed the selenium role from our provisioning #27
- Fix back-to-list buttons on top in modal upload #172

New features

- Block menu caches are invalidated when a node is removed, moved, restore and the status of a node changes . #1531
- add voter on role usage #1530
- increase ergo fo group form #1522
- When a content type is deleted, navigation menu is refreshed #1521
- When a website is created, an homepage for this site is also created. #1518
- refresh the page after website creation to show it in the fixed top nav list #1515
- fix save & back-to-list buttons on top #1499
- Move template menu entry from Editorial to Administration #1485

- allow to extend js form's behavior #1473
- add loader in group tab view #1467
- change new link to button #1466
- Add sortable option to the datatable #1451
- add boolean expression in keywords form type filter #1442
- Remove max length option on email field type #1439
- Datatable action buttons are disabled if the user hasn't the rights #1438
- add notification colors #1433
- Media roles gestion for each folders are available in the group panel #1416
- Allow published node deletion #1405
- add break line in tooltip helper (n) #1400
- allow published node delete #1384
- auto-select type of search in datatable for content field type #1382
- activate content list block in front #202
- Node cache-control policy differs with or without ESI support #193
- allow end user to format boolean condition in keywords filter #41
- Media can be filtered on type when displayed from media folder #179
- Media library: applications can now describe their own specific alternatives image formats #144
- Add a "Back to list" button on the media upload form in a modal context #133
- Media selection integrates now the alternative selector #132
- Published flag feature #529

Deprecated

- `initializeNewNode` is deprecated and replaced by `initializeNode` in class `OpenOrchestra\Backoffice\Manager\NodeManager`. this method will be removed in 1.2.0 #1518
- rename class `UpdateNodeRedirectionSubscriber` to `UpdateRedirectionNodeSubscriber` #1503
- `OpenOrchestra\Backoffice\Form\DataTransformer\ChoiceArrayToStringTransformer` is now deprecated will be removed in 1.2.0 #1472
- `OpenOrchestra\Backoffice\Form\DataTransformer\ChoiceStringToArrayTransformer` is now deprecated will be removed in 1.2.0 #1472
- `FolderRepository:findAllRootFolder` method in `OpenOrchestra\MediaModelBundle\Repository` namespace has been deprecated use `findAllRootFolderBySiteId` #177
- Update the media twig functions #168

Possible BC breaker

- include string indexation of site alias in error pages path. #1503
- `DisplayBundle/DisplayBlock/Strategies/AbstractStrategy:getCacheTags()` is now abstract and must therefore be explicitally implemented on each display strategy #1457
- Move form from backofficeBundle #1425
- Move Model and repository from BackofficeBundle #1424
- Move BackofficeBundle manager #1423
- Move BackofficeBundle subscribers #1422
- Move BackofficeBundle listener #1421
- Move Backoffice display block folder #1417
- Move Backoffice display icon folder #1417
- Move Backoffice initializer folder #1417
- change way to load dataparameter in refresh menu context #1386
- DisplayBundle/DisplayBlock/Strategies/AbstractStrategy:getCacheTags() is now abstract and must therefore be explicitally implemented on each display strategy #200
- Disable multi website in media folder #185
- Fixtures for production are cleaned #530
- In the front, the nodes should have the published flag to be displayed #529

Other changes

- New Bundle : MediaAdminModelBundle #831
- Upgrade to symfony 2.8.3 #72
- Role `ROLE_ACCESS_MOVE_NODE` is renamed to `ROLE_ACCESS_MOVE_TREE` and is now a global role #1480
- Deleted nodes order is -1 #1440
- Redirect to the user edit form after user creation #1408
- Refacto const display strategies name #1407
- Replace strings `elastica_search` and `elastica_list` by class constant #21
- Get master request in method where it is used and not in a constructor #150
- Add the robots meta in the page header FrontBundle#138
- Unskip a MediaStorageManager unit test #8
- Node order validator don't checks deleted nodes #526
- Add findByNodeAndSiteSortedByVersion() request in NodeRepository #514
- Suppression of install of nodeJS by the provisioning, it is now installed by `composer-extra-assets`. use now `./bin/grunt` for grunt and not `./node_modules/.bin/grunt` #34

CHANGELOG for 1.1.0-alpha4

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs
- Media file bundle
- Elastica bundle

Possible BC breaker

- `NodeGroupRoleTransformer` implements now `TransformerWithGroupInterface`
- Method `reverseTransform` of `NodeGroupRoleTransformer` is removed and replaced by `reverseTransformWithGroup`
- Before `composer install|update` it's recommanded to removed `nodes_modules` and `bower_components` folders of your application and `bower.json` and `package.json` files. Remove also symlink in `web/fonts`
- You can move your bower and npm dependencies in `composer.json` of your project or bundle
- Datatable configuration have been moved from the *data* tag to a *yaml* configuration
- *Grunt* configuration has been modified to be more handled by the bundles

Bug fixes

- The Back Office is compatible again with Internet Explorer 9. See Configurations changes for more info
- Fix a bug when a site declared in Apache conf but not in Orchestra was accessed by a client

New features

- You can specify your bower and npm dependencies directly in the composer.json, futher information in the documentation
- Selector version of node tranverse is removed
- Duplicate button of node tranverse is removed
- Delete button of node tranverse is removed
- An *Elastica* implementation has been created to index all the *content* in an indexor

Other changes

- PHP requirement has been updated to 5.5
- Version of `symfony/symfony` is updated to 2.8.1
- Version of `twig/twig` is updated to ~1.23
- Version of `friendsofsymfony/http-cache-bundle` is updated to 1.3.6
- Flow.js dependency has been moved from CmsBundle to MediaAdminBundle
- Media alternatives files (mainly images) have new auto-generated names
- Deprecation of the *ContainerAware* class have been suppressed
- Nodes can have a new options for the theme : use the website default theme
- Abstract test classes have been created to help free the memory used in the tests

Deprecated method

Suppressed method

Configuration changes

- `pace.js` is removed and replace by a custom component `OpenOrchestra.AjaxLoader.AjaxLoaderView`
- To get the Back Office compatible again with Internet Explorer 9, some grunt tasks have been rewrote. Be sure to update your Open Orchestra tasks according to these Pull Request
- `gridstack.js` is removed
- `lodash.js` is removed
- Npm package `grunt-js-test` is added to composer.json of `CmsBundle`

CHANGELOG for 1.1.0-alpha3

Url to see changes:

- Cms bundle
- Display bundle
- BBCode bundle
- Model bundle

- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle
- Media admin bundle
- Orchestra libs
- Orchestra Mongo libs

Possible BC breaker

- Plugin `Colvis` of DataTable is replaced by the `Buttons extension`
- `UploadedFileManager` is moved from `MediaBundle` to `MediaFileBundle`
- Class of `BBcode` folder are moved from `MediaBundle` folder to `Media` folder
- Service `open_orchestra_media.manager.uploaded_media` is renamed by `open_orchestra_media_file.manager.storage`
- Application `open-orchestra-media-demo` no longer requires this bundles : - `DoctrineMongoDBBundle` - `OpenOrchestraBaseBundle` - `OpenOrchestraMediaModelBundle` - `OpenOrchestraMongoBundle` - `OpenOrchestraModelBundle` - `SolutionMongoAggregationBundle`
- Application `open-orchestra-media-demo` requires *MediaFileBundle'*
- Route `open_orchestra_media_get` is now in `MediaController` in `MediaFileBundle`
- Class moved from `MediaBundle` to `MediaAdminBundle` : - `FolderEvent` - `ImagickEvent` - `MediaEvent` - *DeleteMediaSubscriber'* - `GenerateImageSubscriber` - `UploadImageSubscriber` - `FolderEvents` - `OrchestraImagick` - `OrchestraImagickFactory` - `OrchestraImagickFactoryInterface` - `OrchestraImagickInterface` - `ImageResizerManager` - `MediaEvents` - `ImageToThumbnailManager` - `PdfToImageManager` - `VideoToImageManager` - `ThumbnailInterface` - `ThumbnailManager` - `FFmpegVideoManager` - `VideoManagerInterface` - `SaveMediaManager` - `SaveMediaManagerInterface`

Bug fixes

- The node drag'n'drop has been updated to wait for user's confirmation before sending datas
- Fix errors on template creating
- Clean redirection and route on unpublished node
- Fixes on Dashboard, action buttons, order ...

New features

- Possibility to add custom fields for search in the DataTable, further information in documentation
- New Backbone view generic `DataTableView` for create a list with DataTable, used by `TableViewCollection`
- The build of the full project has been moved to the travis container build
- Every facades returned by the API are now configurable and can be defined in bundles configuration, further information in documentation
- The *RoleCollector* has been split into a collector for the front office and the backoffice
- To prevent *Status* suppression, the *StatusUsageFinder* has been created to check in every document using *Status* if they are using it
- Install smartadmin Datepicker
- Datatable preferences saving
- Add embedded entity to content attributs
- New bundle MediaFileBundle used to managed media files with Gaufrette
- Adding of upload stategies to manage all alternatives of media like thumbnail, futher information in the documentation

Other changes

- DataTable is updated to version 1.1.0. In your *bower.json* file replace lines :

```
 "datatables": "~1.10.2",
 "datatables-colvis": "~1.1.2",
 "datatables-bootstrap3": "*",

by
```

```
"datatables.net-buttons-bs": "1.1.0",
```

- Version of doctrine/cache is fixed to 1.5.*
- Version of doctrine/commmon is fixed to 2.5.*

Deprecated method

- *NodeGroupRoleVoter* has been moved in the *Backoffice* folder
- *GroupSiteVoter* has been moved in the *Backoffice* folder
- The *AuthorizeEditionManager* has been deprecated, and all strategies has been transformed has roles
- The *VersionableSaver* has been moved in the *Saver* folder
- The role constant *ROLE_ACCESS_GENERAL_NODE* has been replaced by *ROLE_ACCESS_TREE_GENERAL_NODE*

Suppressed method

Configuration changes

CHANGELOG for 1.1.0-alpha2

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle

Possible BC breaker

- The name of *OpenOrchestraBackofficeBundleFormTypeAreaType* is now *oo_area*
- The name of *OpenOrchestraBackofficeBundleFormTypeBlockChoiceType* is now *oo_block_choice*
- The name of *OpenOrchestraBackofficeBundleFormTypeBlockType* is now *oo_block*
- The name of *OpenOrchestraBackofficeBundleFormTypeExistingBlockChoiceType* is now *oo_existing_block*
- The name of *OpenOrchestraBackofficeBundleFormTypeNodeType* is now *oo_node*
- The name of *OpenOrchestraBackofficeBundleFormTypeApiClientType* is now *oo_api_client*
- The class *OpenOrchestraBackofficeBundleFormTypeAbstractOrchestraGroupType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeAbstractGroupChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraColorPickerType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeColorPickerType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraChoicesOption* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentChoicesOptionType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraContentTypeChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentContentTypeChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraDateWidgetOption* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentDateWidgetOptionType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraFieldChoice* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentFieldChoiceType*

- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraKeywordsType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentKeywordsChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraNodeChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentNodeChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraRoleChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentRoleChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraSiteChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentSiteChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraThemeChoiceType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeComponentThemeChoiceType*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraVideoType* is deleted and replaced by *OpenOrchestraBackofficeBundleFormTypeVideoType*
- The name of *OpenOrchestraBackofficeBundleFormTypeContentType* is now *oo_content*
- The name of *OpenOrchestraBackofficeBundleFormTypeContentTypeType* is now *oo_cont ent_type*
- The name of *OpenOrchestraBackofficeBundleFormTypeFieldOptionType* is now *oo_field_option*
- The name of *OpenOrchestraBackofficeBundleFormTypeFieldTypeType* is now *oo_field_type*
- The name of *OpenOrchestraBackofficeBundleFormTypeGroupType* is now *oo_group*
- The name of *OpenOrchestraBackofficeBundleFormTypeKeywordType* is now *oo_keyword*
- The name of *OpenOrchestraBackofficeBundleFormTypeRedirectionType* is now *oo_redirection*
- The name of *OpenOrchestraBackofficeBundleFormTypeRoleType* is now *oo_role*
- The name of *OpenOrchestraBackofficeBundleFormTypeSiteAliasType* is now *oo_site_alias*
- The name of *OpenOrchestraBackofficeBundleFormTypeSiteType* is now *oo_site*
- The name of *OpenOrchestraBackofficeBundleFormTypeStatusType* is now *oo_status*
- The name of *OpenOrchestraBackofficeBundleFormTypeTemplateType* is now *oo_template*
- The name of *OpenOrchestraBackofficeBundleFormTypeThemeType* is now *oo_theme*
- The name of *OpenOrchestraBackofficeBundleFormTypeTinymceType* is now *oo_tinymce*
- The name of *OpenOrchestraBackofficeBundleFormTypeTranslatedValueCollectionType* is now *oo_translated_value_collection*
- The name of *OpenOrchestraBackofficeBundleFormTypeTranslatedValueType* is now *oo_translated_value*
- The class *OpenOrchestraBackofficeBundleFormTypeOrchestraGroupType* is deleted and replaced by *OpenOrchestraGroupBundleFormTypeGroupDocumentType*
- The name of *OpenOrchestraMediaAdminBundleFormTypeFolderType* is now *oo_folder*
- The name of *OpenOrchestraMediaAdminBundleFormTypeMediaCropType* is now *oo_media_crop*
- The name of *OpenOrchestraMediaAdminBundleFormTypeMediaMetaType* is now *oo_media_meta*
- The name of *OpenOrchestraMediaAdminBundleFormTypeMediaType* is now *oo_media*
- The class *OpenOrchestraMediaAdminBundleFormDataTransformerOrchestraMediaTransformer* is deleted and replaced by *OpenOrchestraMediaAdminBundleFormDataTransformerMediaChoiceTransformer*
- The class *OpenOrchestraMediaAdminBundleFormTypeOrchestraMediaType* is deleted and replaced by *OpenOrchestraMediaAdminBundleFormTypeComponentMediaChoiceType*

- The class *OpenOrchestraMediaAdminBundleFormTypeOrchestraSiteForFolderChoiceType* is deleted and replaced by *OpenOrchestraMediaAdminBundleFormTypeSiteForFolderChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraRoleType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeWorkflowRoleChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraSiteType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeGroupSiteChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraStatusType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeStatusChoiceType*
- The class *OpenOrchestraModelBundleFormTypeOrchestraThemeType* is deleted and replaced by *OpenOrchestraModelBundleFormTypeSiteThemeChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraRoleType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractWorkflowRoleChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraSiteType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractGroupSiteChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraStatusType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractStatusChoiceType*
- The class *OpenOrchestraModelInterfaceFormTypeAbstractOrchestraThemeType* is deleted and replaced by *OpenOrchestraModelInterfaceFormTypeAbstractSiteThemeChoiceType*
- The name of *OpenOrchestraAdminBundleFormTypeRegistrationUserType* is now *oo_registration_user*
- The name of *OpenOrchestraAdminBundleFormTypeUserType* is now *oo_user*
- The name of *OpenOrchestraUserBundleFormTypeChangePasswordUserType* is now *oo_user_change_password*
- The class *OpenOrchestraWorkflowFunctionAdminBundleFormTypeAuthorizationType* is deleted and replaced by *OpenOrchestraWorkflowFunctionAdminBundleFormTypeComponentAuthorizationType*
- The class *OpenOrchestraWorkflowFunctionAdminBundleFormTypeOrchestraWorkflowFunctionType* is deleted and replaced by *OpenOrchestraWorkflowFunctionAdminBundleFormTypeComponentWorkflowFunctionChoiceType*
- The name of *OpenOrchestraWorkflowFunctionAdminBundleFormTypeWorkflowRightType* is now *oo_workflow_right*
- The name of *OpenOrchestraWorkflowFunctionAdminBundleFormTypeWorkflowFunctionType* is now *oo_workflow_function*

Bug fixes

- User is now able to delete a media folder when the last media is deleted, without having to refresh the page.
- When creating a new media folder, menu is now automatically refreshed.

New features

- Replace the media upload GUI with a new component allowing multi-upload
- Adding roles for nodes (CREATE, UPDATE, MOVE, DELETE)
- Adding roles for content types (CREATE, UPDATE, DELETE)
- Adding roles for keywords (CREATE, UPDATE, DELETE)

- Adding roles for redirections (CREATE, UPDATE, DELETE)
- Adding roles for trashcan (RESTORE)
- Adding roles for api accesses (CREATE, UPDATE, DELETE)
- Adding roles for contents (CREATE, UPDATE, DELETE)
- Adding roles for medias (CREATE, UPDATE, DELETE)
- Adding roles for roles (CREATE, UPDATE, DELETE)
- Adding roles for sites (CREATE, UPDATE, DELETE)
- Adding roles for users (CREATE, UPDATE, DELETE)
- Adding roles for transverse nodes (UPDATE)
- Adding roles for workflow status (CREATE, UPDATE, DELETE)
- Adding roles for workflow functions (CREATE, UPDATE, DELETE)

Other changes

- In differents dataTable, the global search is disabled. To reactivate it, you can use the data attribute `display-global-search=true` in the link in navigation panel.
- Every repository that should be paginated are now implementing *OpenOrchestraPaginationConfigurationPaginationRepositoryInterface*
- The version of Symfony is updated to 2.7.6
- Module php5-ffmpeg is replaced by PHP driver PHP-FFMpeg

Deprecated method

- The method *findByAuthor* has been deprecated in both NodeRepository and ContentRepository
- The class *OpenOrchestraModelInterfaceRepositoryPaginateRepositoryInterface* has been replaced by *OpenOrchestraPaginationConfigurationPaginationRepositoryInterface*

Suppressed method

Configuration changes

CHANGELOG for 1.1.0-alpha1

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle

- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

Bug fixes

- Each time you update a site, the routes in the database related to this site are updated
- Sorts by attributes in content and content type view are fully functional

New features

- It is now possible to configure the Back Office dashboard and to create new widgets for it.
- Routes stored in the database are now linked to the last published version of a node
- You can now add the author, last contributor and contribution date in the content listing
- 2 new events are availables in the block rendering process: PRE_BLOCK_CREATION and POST_BLOCK_CREATION
- It is now possible to filter a column which contains a type *date* in dataTable
- Transverse block are now created on all transverse nodes
- When you add a language to a website, the transverse node is created with all the transverse blocks

Other changes

- Bundles are now using the PSR-4 syntax to be loaded, you should update the *Gruntfile* to follow this path modification

Deprecated method

- Some method from the *NodeRepositoryInterface* and *ContentRepositoryInterface* have been deprecated
- The method *findByNodeType* has been deprecated

Suppressed method

Configuration changes

1.1.2 1.0.x

CHANGELOG for 1.0.4

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle
- Orchestra libs
- Media admin bundle

Configuration changes

- Update requirement symfony to `symfony/symfony" : "~2.7.4"` #80
- Update requirement twig to `twig/twig": "~1.23.0` #80

Bug fixes

- Fix dependency `doctrine/mongodb-odm-bundle` #182
- Fix type `nodeId` in php doc of `NodeInterface` and `ReadNodeInterface` #177

CHANGELOG for 1.0.3

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle

- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle
- Orchestra libs
- Media admin bundle

Possible BC breaker

Bug fixes

- Selector version of node tranverse is removed
- Duplicate button of node tranverse is removed
- Delete button of node tranverse is removed
- remove lodash #835
- Fix color picker form type #1496
- Fix prototype bug #1483
- In the navigation panel, a root menu is hidden if its submenus aren't visible #1462
- Fix infinite ajax call #1392
- Fix various bugs with Internet Explorer 9

New features

Other changes

- Adding children direction for the node in the Back Office
- As the pixel developer google chrome is not more accessible, the selenium role from the default provisioning has been removed #28
- Symfony requirement is now `^2.7.4` #73

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 1.0.2

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

Bug fixes

- In the Back Office, once you are logged out from the application, you are redirected to the login page. The check is now done on the HTTP status code or on a response header
- In the group edition, you can add the workflow role also. If you use the *workflow-bundle*, the workflow role does not appear

New features

Other changes

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 1.0.1

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

Bug fixes

- In the Back Office dashboard, some widgets were inverted and the deleted elements are no more displayed
- Fix tinyMce in case of multiple popin openings
- Databases routes are now linked to the last published version of a node
- The Video Block can now be correctly edited and saved

New features

- You can now add the author, last contributor and contribution date in the content listing
- In the Back Office, media modals have been enhanced with scrollbars

Other changes

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 1.0.0

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle

- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

Bug fixes

- I can now duplicate a node with area inside other areas with no interaction with the duplicated node

New features

- The routing can be now done from the database
- New restriction on media upload based on the mime-type

Other changes

Deprecated method

Suppressed method

Configuration changes

- In the front part, you should activate the *SymfonyCmfRoutingBundle* to use the *ChainRouter*

CHANGELOG for 1.0.0-RC2

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle

- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- In `NodeManager`, parameter `siteRepository` must implement `ReadSiteRepositoryInterface`
- In `LanguageListStrategy`, parameter `siteRepository` must implement `ReadSiteRepositoryInterface`
- In `KernelExceptionSubscriber`, parameter `siteRepository` must implement `ReadSiteRepositoryInterface`
- In `DatabaseRouteLoader`, parameter `siteRepository` must implement `ReadSiteRepositoryInterface`
- In `RedirectionLoader`, parameter `siteRepository` must implement `ReadSiteRepositoryInterface`

Bug fixes

New features

- SaveMediaManager can upload multiple files

Other changes

- `SiteRepositoryInterface` is split in `SiteRepositoryInterface` and `ReadSiteRepositoryInterface`
- The Back Office navigation pannel mechanism has been refactored to allow the creation of level 1 strategies

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 1.0.0-RC-1

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface

- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Workflow function bundle

Possible BC breaker

- `open_orchestra_model.annotation_reader` is renamed by `open_orchestra.annotation_reader`
- `open_orchestra_base.annotation_search_reader` is moved to orchestra-libs and renamed by `open_orchestra.annotation_search_reader`
- Adding new bundle (OpenOrchestraMongoBundle)
- The parameter `descriptionEntity` in `FinderConfiguration` is now composed of array (`Array("key" => "", "field" => "", "type" => "")` futher information documentation

Bug fixes

- Search with hidden columns in dataTable

New features

- You can describe the mapping for search in Yaml and XMl, futher information in documentation
- A BBcode bundle is introduced to allow you to extend the rich text editor
- A new BBcode tag is available for media
- add new interface and command line to allow all, unit or functionnal tests loading

Other changes

- Mongo-odm requirement is updated to 1.0.1
- Remove MongoDBMigrationsBundle
- A refacto has been made to simplify the navigation
- Upgrade to mongo-odm 1.0.2, mongo-odm-bundle 3.0.1
- Acces Token no more revoked
- Upgrade to symfony 2.7.4
- Upgrade to twig/extensions 1.3.0
- Upgrade to symfony/assetic-bundle 2.7.0
- Upgrade to friendsofsymfony/http-cache-bundle 1.3.3

- Upgrade to phpunit/phpunit 4.8.8

Deprecated method

Suppressed method

Configuration changes

- The *LogBundle* has been split for the model, add the *ModelLogBundle* in the *AppKernel*

1.2 Versions 0.x.x

1.2.1 0.3.x

CHANGELOG for 0.3.4

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- *ContentInterface*, *NodeInterface* extend *TrashCanDisplayableInterface*
- *ContentTypeInterface*, *SiteInterface* extend *SoftDeleteableInterface*
- The method *getDeleted* of *ContentInterface*, *ContentTypeInterface NodeInterface* is removed and replace by method *isDeleted* of *SoftDeleteableInterface*
- In *ApiBundle DeletedController* is replaced by *TrashCanController*

Bug fixes

- Image cropping has been fixed

New features

Other changes

- you can now use search and pagination in trash can
- add *TrashCanListener* which creates a trashItem when a document which implements *TrashCanDisplayableInterface* is deleted
- The Api errors are display in a smart notification box
- SuperAdmin can always acces to Group and Workflow

Deprecated method

- The method *findAllDeleted* of *ContentRepositoryInterface* is deprecated and will be removed in 0.3.5.
- The method *findDeletedInLastVersionBySiteId* of *NodeRepositoryInterface* is deprecated and will be removed in 0.3.5.

Suppressed method

- The constraint *PreventSavedPublishedDocument* will be removed, you should use the *AuthorizeEditionManager* instead.
- The listener *SavePublishedDocumentListener* will be removed, you should use the *AuthorizeEditionManager* instead
- The method *findLastVersionByDeletedAndSiteId* of *NodeRepositoryInterface* is removed.

Configuration changes

- The gruntfile.js has been refactored to enhance its capabilities and ease its usage. He presents now two functions to load a single config file or a full config dir. The task options files do not have anymore limitation on their formats. You can read the doc for further information.

Important: to be compatible with the MediaAdminBundle v0.3.4, you must update the gruntfile.js

CHANGELOG for 0.3.3

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle

- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- The api authentication strategies should only return either a *FacadeInterface* or a *ConstraintViolationListInterface* object.
- The serialization of the result is done through the same subscriber every other api request
- Configuration files are rename to snake case. example : `oauth2routing.yml` become `oauth2_routing.yml`
- The backbone router has been refactored to contain only the generation and route matching, every route should now be declared separetly
- The backbone view `FullPagePanelView` is removed replace by the generic view for tabs `TabView` and `TabElementFormView`

Bug fixes

New features

- You can use the command *orchestra:mongo:fixtures:load* to load only the fixtures implementing the interface *OpenOrchestraModelInterfaceDataFixturesOrchestraProductionFixturesInterface*.
- You can filter the content attributes in the contents dataTable
- An Api client can now have some base role that will be merged with the user role
- Only last version of node and content can be change now.

Other changes

- Media use a new thumbnail format to display media

Deprecated method

- The *PreventPublishedDocumentSave* has been replaced by the *AuthorizeEdition* constraint.

Suppressed method

Configuration changes

- Thumbnail configuration is always *max_width* and *max_height* and not *width* and *height* for rectangle
- *object_manager* is defined in `open-orchestra-base-bundle` and not in `base-api-mongo-bundle`

CHANGELOG for 0.3.2

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- The *ApiSerialize()* annotation should be given to the full class now
- Reference to `LeftPanel` in class are replace by `NavigationPanel`
- You can use the annotation *OpenOrchestraMappingAnnotationsSearch* for the mapping your entity for the dataTable search, futher informations in the documentation
- The annotation *OpenOrchestraModelInterfaceMappingAnnotationsDocument* is moved to *open-orchestra-libs* (*OpenOrchestraMappingAnnotationsDocument*)
- *OpenOrchestraModelInterfaceExceptionsMethodNotFoundException* is moved to *MappingExceptionsMethodNotFoundException*
- *OpenOrchestraModelInterfaceExceptionsPropertyNotFoundException* is moved to *MappingExceptionsPropertyNotFoundException*

Bug fixes

New features

Other changes

- You can now duplicate the node version you want
- You can duplicate the current version of the node
- I can specify that you have the workflow right on the content you have created

Deprecated method

- In the *NodeInterface*, the methode *isEditable* has been replaced by the *AuthorizeEditionManager*

Suppressed method

Configuration changes

CHANGELOG for 0.3.1

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Base api model bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- In DisplayBlockInterface *getTags* is renamed to *getCacheTags*.
- A *BaseApiModelBundle* has been added, this way you can use the *BaseApiBundle* with no relation to mongo

Bug fixes

New features

- Possibility to custom field default value in content attribute
- Possibility to define a tempalte wysiwyg for the block configurable content
- Custom pages for 503 errors are now availables in Back Office
- Custom pages for 404 errors are now availables in Back Office

Other changes

- *getContentType* has been moved from *ContentInterface* to *ReadContentInterface*
- add datatables-bootstrap3, lodash and gridstack to bower.json
- FosHttpCacheBundle requirement has been changed to 1.3.2
- Update smartadmin to 1.6.1
- Update bootstrap to 3.3.4 in bower.json
- You could add an external media domain

Deprecated method

- `GaufretteManager` is deprecated and will be removed in 0.3.2 , used `UploadMediaManager`

Suppressed method

- `ModelBundle/Repository/AbstractRepository` is suppressed
- The trait `TranslatedValueFilter` is suppressed
- The interface `VersionnableInterface` is suppressed, use `VersionableInterface` instead
- `ModelInterface/Model/NodeInterface/getInFooter` is suppressed, use `ModelInterface/Model/NodeInterface/isInFooter` instead
- `ModelInterface/Model/NodeInterface/getInMenu` is suppressed, use `ModelInterface/Model/NodeInterface/isInMenu` instead
- `ModelInterface/Model/SiteInterface/getDeleted` is suppressed, use `ModelInterface/Model/SiteInterface/isDeleted` instead
- `ModelInterface/Repository/ContentTypeRepositoryInterface/findAllByDeletedInLastVersion` is suppressed, use `ModelInterface/Repository/ContentTypeRepositoryInterface/findAllNotDeleted` instead
- `ModelInterface/Repository/ContentTypeRepositoryInterface/findAllByDeletedInLastVersionI` is suppressed, use `ModelInterface/Repository/ContentTypeRepositoryInterface/findAllNotDeleted` instead
- `ModelInterface/Repository/NodeRepositoryInterface/findOneByNodeIdAndLanguageAndSiteIdA` is suppressed, use `ModelInterface/Repository/NodeRepositoryInterface/findOneByNodeIdAndLangua` instead
- `ModelInterface/Repository/NodeRepositoryInterface/findChildsByPathAndSiteIdAndLanguage` is suppressed, use `ModelInterface/Repository/NodeRepositoryInterface/findChildrenByPathAndSit` instead
- `ModelInterface/Repository/NodeRepositoryInterface/findOneByNodeIdAndLanguageAndVersionA` is suppressed, use `ModelInterface/Repository/NodeRepositoryInterface/findOneByNodeIdAndLangua` instead
- `` ModelInterface/Repository/ReadNodeRepositoryInterface/findOneByNodeIdAndLanguageWithPublishedAndLastVersionAnd` is suppressed, use `ModelInterface/Repository/ReadNodeRepositoryInterface/findOnePublishedByNo` instead

- `ModelBundle/Document/Area/setClasses` is suppressed, use `ModelBundle/Document/Area/setHtmlClass` instead
- `ModelBundle/Document/Area/getClasses` is suppressed, use `ModelBundle/Document/Area/getHtmlClass` instead
- `ModelBundle/Repository/ContentRepository/findByContentTypeInLastVersion` is suppressed, use `ModelBundle/Repository/ContentRepository/findByContentTypeInLastVersionForPa` instead
- `` ModelBundle/Repository/ContentTypeRepository/createAggregateQueryByDeletedAndLastVersion`` is suppressed, use `` ModelBundle/Repository/ContentTypeRepository/createAggregateQueryNotDeletedInLastVersion`` instead

Configuration changes

- Activate the *BaseApiModelBundle* : *new OpenOrchestraBaseApiModelBundleOpenOrchestraBaseApiModelBundle()*

CHANGELOG for 0.3.0

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- Add four methods in `ContentRepositoryInterface : - getDefaultListable() - addDefaultListable($name, $value) - removeDefaultListable($name) - setDefaultListable(array $defaultListable)`

Bug fixes

New features

- Possibility to hide the default columns in the dataTable
- Content attributes from other format than string can be display on datatable thanks to transformer

Other changes

- site 1 (front site) is removed of fixtures

Deprecated method

Suppressed method

- `AddLinearize` method and `linearize` attribute from `ContentFacade` are removed

Configuration changes

1.2.2 0.2.x

CHANGELOG for 0.2.12

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- The parameters sent by the dataTable are changed
- The namespace of trait *ListStatus* is now *OpenOrchestraApiBundleControllerControllerTrait*
- In `ContentRepositoryInterface` : - `findByContentTypeInLastVersionForPaginateAndSearch` is deleted and replace by `findByContentTypeAndSiteIdInLastVersionForPaginate` - `findByContentTypeInLastVersionForPaginateAndSearchAndSiteId` is deleted and replace by `findByContentTypeInLastVersionForPaginateAndSearchAndSiteId` - `countByContentTypeInLastVersionWithSearchFilter` is deleted and replace by `countByContentTypeInLastVersionWithFilter`
- In `ApiClientRepositoryInterface` : - `findForPaginateAndSearch` is deleted and replace by `findForPaginate` - `countWithSearchFilter` is deleted and replace by `countWithFilter`

- In `WorkflowFunctionRepositoryInterface` : - `findForPaginateAndSearch` is deleted and replace by `findForPaginate` - `countWithSearchFilter` is deleted and replace by `countWithFilter`
- In `PaginateRepositoryInterface` : - `findForPaginateAndSearch` is deleted and replace by `findForPaginate` - `countWithSearchFilter` is deleted and replace by `countWithFilter`
- In `SiteRepositoryInterfaceInterface` : - `findByDeletedForPaginateAndSearch` is deleted and replace by `findByDeletedForPaginate` - `countByDeletedWithSearchFilter` is deleted and replace by `countWithSearchFilterByDeleted` - `PaginateAndSearchFilterTrait` is deleted and replace by `PaginationTrait`

Bug fixes

- Node preview has been fixed, see configuration changes for more info

New features

Other changes

Deprecated method

- `ModelBundle/Repository/AbstractRepository` is deprecated will be removed in 0.3.0
- The trait `TranslatedValueFilter` is deprecated will be removed in 0.3.0
- In `ContentRepositoryInterface` : - `countByDeletedInLastVersionWithSearchFilter` replace by `countNotDeletedInLastVersionWithSearchFilter`
- In `ContentTypeRepositoryInterface` : - `findAllByDeletedInLastVersionForPaginateAndSearch` replace by `findAllNotDeletedInLastVersionForPaginate` - `countByDeletedInLastVersionWithSearchFilter` replace by `countDeletedInLastVersionWithSearchFilter` - `findAllByDeletedInLastVersion` replace by `findAllNotDeletedInLastVersion`
- In `NodeRepositoryInterface` : - `findOneByNodeIdAndLanguageAndSiteIdAndLastVersion` replace by `findOneByNodeIdAndLanguageAndSiteIdInLastVersion` - `findLastVersionByDeletedAndSiteId` replace by `findDeletedInLastVersionBySiteId` - `findLastVersionByDeletedAndSiteId` replace by `findDeletedInLastVersionBySiteId` - `findChildsByPathAndSiteIdAndLanguage` replace by `findChildrenByPathAndSiteIdAndLanguage` - `findByParentIdAndRoutePatternAndNotNodeIdAndSiteId` replace by `findByParentIdAndRoutePatternAndNodeIdAndSiteId` - `findOneByNodeIdAndLanguageAndVersionAndSiteId` replace by `findOneByNodeIdAndLanguageAndSiteIdAndVersion`
- In `ReadNodeRepositoryInterface` : - `findOneByNodeIdAndLanguageWithPublishedAndLastVersionAnd` replace by `findOnePublishedByNodeIdAndLanguageAndSiteIdInLastVersion`
- The *linearized_attributes* from the *ContentFacade* are not use anymore

Suppressed method

- All the display block classes from the MediaBundle have been removed
- `ModelInterface/Form/Type/AbstractOrchestraRoleType` has been removed

- `ModelInterface/MongoTrait/Versionnable` has been removed. Replace by `ModelInterface/MongoTrait/Versionable`
- In `AreaInterface` : - `setClasses` replace by `setHtmlClass` - `getClasses` replace by `getHtmlClass`
- In `ReadAreaInterface` : `getClasses` replace by `getHtmlClass`
- In `ContentTypeRepositoryInterface` : - `findOneByContentTypeIdAndVersion` - `findOneByContentTypeIdAndVersionfrom`
- In `ContentRepositoryInterface` : `findByContentTypeInLastVersion`
- In `NodeRepositoryInterface` : `findOneByParendIdAndRoutePatternAndSiteId`

Configuration changes

- In order to get the new routing conf, Back Office configuration requires to be updated. In app/config/routing.yml of your back application, add the following lines :

```
open_orchestra_base:
    resource: "@OpenOrchestraBaseBundle/Resources/config/routing.yml"
```

- The front configuration must also be updated. In app/config/routing.yml of your front application, add the following lines :

```
open_orchestra_front_preview:
    resource: "@OpenOrchestraFrontBundle/Resources/config/preview_routing.yml"
```

- A *MediaModelBundle* has been created to store the media document, you should activate the bundle :

```
new OpenOrchestra\MediaModelBundle\OpenOrchestraMediaModelBundle(),
```

- A *BaseApiModelBundle* has been created to store the BaseApi document, you should activate the bundle :

```
new OpenOrchestra\BaseApiModelBundle\OpenOrchestraBaseApiModelBundle(),
```

- A *WorkflowFunctionModelBundle* has been created to store the WorkflowFunction document, you should activate the bundle :

```
new OpenOrchestra\WorkflowFunctionModelBundle\OpenOrchestraWorkflowFunctionModelBundle(),
```

CHANGELOG for 0.2.11

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle

- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- The parameter *open_orchestra_base.languages_availables* has been renamed to *open_orchestra_base.back_office_languages*

Bug fixes

- The Navigation menu is refreshed and operationnal when a media folder is created within a modal
- Multiple TinyMce can be displayed on a single page

New features

Other changes

Deprecated method

- In the media bundle, all the displayBlockStrategies have been moved in the Media Folder

Suppressed method

- ModelBundle/MongoTrait/Statusable.php has been deleted

Configuration changes

CHANGELOG for 0.2.10

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- Backbone model *site*, *block*, *Node*, *Template*, *VersionviewModel* are rename by *SiteModel*, *BlockModel*, *NodeMode*, *TemplateModel* and *VersionModel*
- Backbone Collection *TableviewCollection* is deleted, Backbone model *NodeCollectionElement*, *TableviewElement*, *VersionviewElement* are deleted
- The mongo *group_document* collection have been renamed into *users_group*

Bug fixes

- Do not force a string for the linearized attributes
- In tinyMCE media modal, adding original format image work

New features

- Add the name as key in the *ContentFacade* for the *ContentAttributeFacade*

Other changes

- Database fixture contains more credible content

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 0.2.9

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- In ContentRepositoryInterface and FieldAutoGenerableRepositoryInterface the method testUnicityInContext is rename by testUniquenessInContext

Bug fixes

- Content versions van be naviguated through and created
- Current site can be switched to a site created in the Back Office, not only in the fixtures
- Content type can be deleted
- The deleted contents are not visible in the list
- The media folder are now visible if they are not linked to a website
- The media folder can be editable
- In the content submission, the order in which the transformer are used has been changed (for the reverseTransformation)

New features

- Wysiwyg blocks have a button to add image from Media Library to content
- A content can be linked to the current website
- Some properties and content attributes can be the always the same on all content

Other changes

- Method findLastPublishedVersionByContentIdAndLanguage of ContentRepositoryInterface is moved in ReadContentRepositoryInterface
- Remove all deprecated calls related to Symfony 2.7

Deprecated method

Suppressed method

Configuration changes

- In the *ModelBundle* the class definition is under the *document* entry

CHANGELOG for 0.2.8

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle

- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- Facade which list entities for the DataTable require two additional attributes (recordsTotal and recordsFiltered), recordsTotal is total record without filtering and recordsFiltered record after filtering (search).
- The provisioning has been modified, you can now add multiple port for apache and make varnish dispatch the connection between them

Bug fixes

New features

Other changes

- Gruntfile rewrote to allow developpers to add specific concat tasks in their application
- DataTable use pagination, search and order with Ajax

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 0.2.7

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle

- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

Bug fixes

New features

Other changes

- In Mongo, the Content documents index their content attributes by their names
- In the Back Office, the content menu shows content types ordered by their name according to the BO language
- The method FolderRepository->findallrootfolderbysiteid() has been rewrote

Deprecated method

Suppressed method

Configuration changes

CHANGELOG for 0.2.6

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Base api bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

- Symfony version has been upgraded to 2.7.0

Bug fixes

- Fix broken sitemaps generation

New features

- added DataTable Colvis to Bower and Grunt to enable column choice visibility in dataTable
- added possibility to manage right on content and node change status
- new custom types options can be added in the app/config.yml

Other changes

- Refacto on the grunt tasks

Deprecated method

- The method *eventElligible* from the *AbstractSubscriber* in the *BaseApi* is replaced by *isEventEligible*

Suppressed method

Configuration changes

The grunt tasks managment has been changed to be more user friendly in case of Open Orchestra updates and/or application specific modifications. Tasks and configuration are now splitted into multiples files placed in OpenOrchestraBundle for the generic ones and in CmsBundle for the specific ones. To be compatible with that change, projects must update their actual grunt files. The gruntfile must be updated. The folder grunt_tasks must be created and the tasks it's containing must be added. See Pull request #492 for more information.

As new npm modules are required, a npm install must also be done.

CHANGELOG for 0.2.5

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle
- Worflow function bundle

Possible BC breaker

Bug fixes

New features

- We add the possibility to use custom view instead of *FullPageFormView* for the edition of entity

Other changes

Deprecated method

Suppressed method

- The constant *NodeInterface::TYPE_GENERAL* has been removed

Configuration changes

- You need to update the *Gruntfile.js* to add the *viewConfigurator* file

CHANGELOG for 0.2.4

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- The form type *open_orchestra_user.type.user* has been renamed to *open_orchestra_user_admin.type.user*
- The entities *AccessToken* and *ApiClient* have been moved from the *UserBundle* to the *BaseApiBundle*. Please check with us directly to help you migrate

Bug fixes

- In Back Office, restore broken translations
- In Back Office, restore mybg.png
- In Back Office, the title of a newly created blocks is translated
- In the vagrant box, the bug with the binding on localhost is fixed

New features

- There is a new entry in the left menu for the workflow function

Other changes

- Symfony has been upgraded to the 2.6.7 version

Deprecated method

- The constant *NodeInterface::TYPE_GENERAL* as been rename *TYPE_TRANSVERSE*
- The Form Type *OrchestraColorChoiceType* will be suppressed, a configuration *available_color* will allow the integrator to setUp the colors
- The class *AbstractBlockContentTypeSubscriber* will be renamed as *AbstractModulableTypeSubscriber*

Suppressed method

- In the ReadContentRepositoryInterface, the method findByContentTypeAndChoiceTypeAndKeywords is suppressed
- In the FolderRepositoryInterface, the method setCurrentSiteManager is suppressed
- In the NodeRepositoryInterface, these methods are suppressed : - findChildsByPath - findByParentIdAndRoutePatternAndNotNodeId
- In the ReadNodeRepositoryInterface, these methods are suppressed : - getFooterTree - getMenuTree - getSubMenu
- The DisplayedElementCollectionTransformer is suppressed.
- The TranslateController is suppressed.

Configuration changes

- If you want to use the workflow function, you need to enable the following bundles :

```
new OpenOrchestra\WorkflowFunctionAdminBundle\OpenOrchestraWorkflowFunctionAdminBundle(),
new OpenOrchestra\WorkflowFunctionBundle\OpenOrchestraWorkflowFunctionBundle(),
```

CHANGELOG for 0.2.3

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- In the ContentRepositoryInterface, the parameters of these methods are change (default value are deleted) : - findLastPublishedVersionByContentIdAndLanguage - findByContentIdAndLanguage - findOneByContentIdAndLanguageAndVersion
- In the FolderRepositoryInterface, the parameter of findAllRootFolderBySiteId are change (add siteId parameter)
- In the NodeRepositoryInterface, the parameters of these methods are change (default value are deleted) : - findByParentIdAndSiteId - findOneByNodeIdAndLanguageAndSiteIdAndLastVersion - findByNodeIdAndLanguageAndSiteId - findByNodeIdAndLanguageAndSiteIdAndPublishedOrderedByVersion - findLastVersionBySiteId - findLastVersionByDeletedAndSiteId - findByNodeIdAndSiteId - findOneByNodeIdAndLanguageAndVersionAndSiteId
- In the ReadNodeRepositoryInterface, the parameters of these methods are change (default value are deleted) : - findOneByNodeIdAndLanguageWithPublishedAndLastVersionAndSiteId

Bug fixes

New features

Other changes

Deprecated method

- In the ReadContentRepositoryInterface, the method findByContentTypeAndChoiceTypeAndKeywords is deprecated
- In the FolderRepositoryInterface, the method setCurrentSiteManager is deprecated
- In the NodeRepositoryInterface, these methods are deprecated : - findChildsByPath - findByParentIdAndRoutePatternAndNotNodeId
- In the ReadNodeRepositoryInterface, these methods are deprecated : - getFooterTree - getMenuTree - getSubMenu
- In the TokenFacade, the parameters expiresIn has been renamed

- The DisplayedElementCollectionTransformer is deprecated, the _translate link is replace by translation in data attribute data-translated-header in template.
- The TranslateController is deprecated.

Suppressed method

Configuration changes

CHANGELOG for 0.2.2

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- The ApiBundle has been split in ApiBundle and BaseApiBundle

Bug fixes

- Blocks js & css files are now correctly loaded
- Missing BO assets are fixed

New features

- In the BackOffice, adding new form type (choice, hidden, money and date) for the contentType.
- Content type fields list can now be extended in app config
- The content type field definition has changed

Other changes

- In the BackOffice, we modified the way we interact with the blocks and area (js and css modification only).
- In the Backoffice, all the display strategies should be tagged with *open_orchestra_backoffice.display_block.strategy*

Deprecated method

- The *VersionnableInterface* and *Versionnable* trait have been renamed to *VersionableInterface* and *Versionable* respectively

Suppressed method

- The following class have been removed from the ApiBundle : - GroupContext - AnnotationSerializer - AnnotationGroup - BaseController - TransformerManaer - AbstractTransformer - TransformerInterface - AbstractFacade - FacadeInterface - ApiException - HttpExceptionApiException

Configuration changes

CHANGELOG for 0.2.1

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- The ApiBundle has been split in ApiBundle and BaseApiBundle

Bug fixes

New features

Other changes

Deprecated method

- The Statusable trait was in the model-bundle, it has been moved to model-interface
- We have moved the following classes from the ApiBundle, they are now deprecated and will be removed in the 0.2.2 version: - GroupContext - AnnotationSerializer - AnnotationGroup - BaseController - TransformerManaer - AbstractTransformer - TransformerInterface - AbstractFacade - FacadeInterface - ApiException - HttpExceptionApiException

Suppressed method

- The class *OpenOrchestraApiBundleControllerBaseController* has been removed

Configuration changes

- Activate the BaseApiBundle in the AppKernel:

```
new OpenOrchestra\BaseApiBundle\OpenOrchestraBaseApiBundle(),
```

- There are now only relations on interfaces :
  - OpenOrchestraModelInterfaceModelEmbedStatusInterface: OpenOrchestraModelBundleDocumentEmbedStatus
  - OpenOrchestraModelInterfaceModelRoleInterface: OpenOrchestraModelBundleDocumentRole
  - OpenOrchestraModelInterfaceModelAreaInterface: OpenOrchestraModelBundleDocumentArea
  - OpenOrchestraModelInterfaceModelBlockInterface: OpenOrchestraModelBundleDocumentBlock
  - OpenOrchestraModelInterfaceModelStatusInterface: OpenOrchestraModelBundleDocumentStatus
  - OpenOrchestraModelInterfaceModelThemeInterface: OpenOrchestraModelBundleDocumentTheme
  - OpenOrchestraModelInterfaceModelSiteAliasInterface: OpenOrchestraModelBundleDocumentSiteAlias
  - OpenOrchestraModelInterfaceModelContentAttributeInterface: OpenOrchestraModelBundleDocumentContentAttribute
  - OpenOrchestraModelInterfaceModelFieldTypeInterface: OpenOrchestraModelBundleDocumentFieldType
  - OpenOrchestraModelInterfaceModelFieldOptionInterface: OpenOrchestraModelBundleDocumentFieldOption

CHANGELOG for 0.2.0

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- Creation of UserAdminBundle
- Creation of MediaAdminBundle

Bug fixes

- Car content type fixtures are functionnal
- AccessToken was linked to the *User* document, it is now linked to the *UserInterface*
- AccessToken was linked to the *ApiClient* document, it is now linked to the *ApiClientInterface*
- Content type with content field which have not options are functionnal

New features

- Possibility of using fullname (bundle:controller:twig) for underscore template in coffee
- Application developpers can now add new custom content attributes

Other changes

- Content Type List block now have a generic template
- In the back office / mediatheque, the media format tab is shown only for images, not for pdfs or videos
- In the creation of a site form, by default the index parameter and the follow parameter are checked
- Content Type have no status

Deprecated method

- In the contentRepositoryInterface, the method findAllNews is deprecated and will be suppressed in 0.2.1
- In the contentTypeRepositoryInterface, the method findOneByContentTypeIdAndVersion is deprecated and will be suppressed in 0.2.1

Suppressed method

Configuration changes

- Activate new bundles for the backoffice: - *new OpenOrchestraUserAdminBundleOpenOrchestraUserAdminBundle(), - new OpenOrchestraMediaAdminBundleOpenOrchestraMediaAdminBundle(),*
- Load the routing for those bundles

```
open_orchestra_user_admin:
    resource: "@OpenOrchestraUserAdminBundle/Controller/Admin"
    type: annotation
    prefix: /admin

open_orchestra_media_admin:
```

```
    resource: "@OpenOrchestraMediaAdminBundle/Controller/Admin"
    type: annotation
    prefix: /admin

open_orchestra_user_api:
    resource: "@OpenOrchestraUserAdminBundle/Controller/Api"
    type: annotation
    prefix: /api

open_orchestra_media_api:
    resource: "@OpenOrchestraMediaAdminBundle/Controller/Api"
    type: annotation
    prefix: /api
```

- Do not use the UserBundle routes
- Add the relation to the *UserInterface* : - *SymfonyComponentSecurityCoreUserUserInterface: OpenOrchestraUserBundleDocumentUser*
- Add the relation to the *ApiClientInterface* : - *OpenOrchestraUserBundleModelApiClientInterface: OpenOrchestraUserBundleDocumentApiClient*
- We use a new bundle to use aggregation query with mongodb, activate the bundle : - *new SolutionMongoAggregationBundleSolutionMongoAggregationBundle(),*

1.2.3 0.1.x

CHANGELOG for 0.1.4

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- The ContactController in the DisplayBundle has been removed. Therefore there are no more controller in this bundle so any reference to it in the routing file should be removed.
- We have moved the FieldAutoGenerableRepositoryInterface from the model-bundle to the model-interface

Bug fixes

- Blocks contentList and configurable content displays the lastest content published in the language considered
- Fix pdf and video media preview

New features

- BO Content Edit forms can now be personalized by Content Type

Other changes

- Delete media with gaufrette
- I don't see header in the query string
- UserBundle is now required in FrontDemo
- Suppress cid in widget

Deprecated method

Suppressed method

Configuration changes

- The controller reference under the *open_orchestra_display* key in the routing should not be used as there is no more controller in this bundle.

CHANGELOG for 0.1.3

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- Repositories interfaces have been split into read and full for the front installation

Bug fixes

New features

- Ban media related caches on media update
- Update the routing cache if the route is not found
- Add api authentication
- Add display media block

Other changes

Deprecated method

Suppressed method

Configuration changes

- It is now possible to add an extra firewall for the api only

CHANGELOG for 0.1.2

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- All interfaces have been split into Interfaces and ReadInterfaces

Bug fixes

- Status creation

New features

- Concerned caches banned on Node/Content status changes
- Multi devices
- Blocks cache are tagged according their content
- Check twig syntax error in contents templates
- Add new specific tags to blocks
- Oauth2 authentication on the api
- Translate media's title and alt

Other changes

- Rename Editorial/template.html.twig into form.html.twig
- Move blocks constants into blocks strategies

Deprecated method

Suppressed method

Configuration modification

- The interface split modify the *resolve_target_documents*, you need to put the new read interfaces instead
- Change the firewall configuration for the api connexion

CHANGELOG for 0.1.1

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- Suppress dynamic routing maganement
- Strategies header, mediaList and Gallery have been moved

Bug fixes

- Keywords field opened in a block form is now correctly closed when closing the form
- Media > crop on an image larger than screen now works correctly
- Preview link
- render_esi on blocks fixed when used with varnish
- Blocks Menu and SubMenu deal with dynamic node pattern
- BO > F5 on media edit now works
- Separation of DisplayBundle and MediaBundle

New features

- Route pattern starting with "/" are absolute
- ReverseProxy cache: Open Orchestra now uses httpfoscachebundle
- Blocks can have specific tags used as cache keys
- Node cache has specific tags
- User are related to a group for the right
- Content preview is now handled by a fake object
- You can create users group
- User are now related to groups

Other changes

- Content can have a personal template
- BO > Mediatheque is redesigned and have a better interface
- Add test behat and provision it
- Provision a cron task
- Add an example of varnish usage
- Add an asterisk if field are required
- Add help for form fields
- DataGrid have bootstrap button
- Block can specify cache management properties
- Move the Group document into a new Group Bundle

Deprecated method

Suppressed method

CHANGELOG for 0.1.0

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- All bundles have been renamed

Bug fixes

- Fix blocks Carrousel, youtube, search and addThis if id wasn't contributed
- Return ContentNotFoundEcxeption if there are no nodeId in the url for blockContent
- Fix Configurable Content block
- Modify Block list in siteType
- I can use radio button
- Templates can now be deleted

New features

- You can access https pages in https only
- Multiple alias preview
- Merge dailymotion, youtube and vimeo blocks into video block
- sitemap.xml & robots.txt have their own rewrite rules
- New empty Dashboard to fix problem in certain circumstances
- WebSite id is now unique and you can't modify it
- You can define the template you want to display a content in the list

Other changes

- Add doc describing Front integration of blocks
- BO > Crop > Remove black borders on rectangle format
- Display save button on the modal footer if there is a delete button

Deprecated method

Suppressed method

1.2.4 0.0.x

CHANGELOG for 0.0.7

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

Bug fixes

- Sitemap Generation is operationnal again
- Provisionning fixed
- Keywords are now selectables in form that require it
- Change the block form generation

New features

- Redirection are stored in the database to feed the front router
- Ability to define a template to display a content
- Scheme of sites and pages is now configurable (used only on BO preview for now)

Other changes

- Back Office > Open sans font familly is now loaded via Bower
- Back Office > In node edition, only allowed blocks for the current site are availables
- Back Office > Mediatheque: thumbnail formats are exploited a new way
- Back Office > Mediatheque: contribution interface is a bit more friendly user
- Back Office > Logs: Add current site name in logs
- Back Office > MediathEque: Remove media preview
- Update to symfony 2.6.4

Deprecated method

- NodeRepositoryInterface::findOneByParendIdAndRoutePatternAndSiteId()
- PhpOrchestraUrlGenerator::dynamicGenerate()
- DynamicRoutingManager::getRouteParameterFromRequestPathInfo()
- DynamicRoutingSubscriber::onKernelException()

Suppressed method

- NodeInterface::getAlias()
- NodeInterface::setAlias()
- NodeRepositoryInterface::findOneByParendIdAndAliasAndSiteId()

CHANGELOG for 0.0.6

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- Modify the way route are created and analyzed
- The website has multiple alias
- Each website alias can provide an url prefix

Bug fixes

- BO left menu is now operationnal after an automatic refresh
- The crop preview is now disable

New features

- Validation on the routePattern creation
- Display the site name in the log table
- Start to use Wreqr to dispatch event in Js

Other changes

- Apc is not anymore required by baseBundle

Deprecated method

- NodeRepositoryInterface: findOneByParendIdAndAliasAndSiteId

CHANGELOG for 0.0.5

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- The block contain all the parameter

Bug fixes

- No more crash when trying to switch node status (media references extraction from blocks fixed)

New features

- New block: Audience Analysis
- List the log entries

Other changes

- There is no a mechanism to allow blocks (front display) to dynamically load js & css. Use require.js
- BO modal forms are now scrollables

CHANGELOG for 0.0.4

Url to see changes:

- Cms bundle
- Display bundle
- Model bundle
- Model interface
- Front bundle
- Base bundle
- Media bundle
- User bundle
- Theme bundle

Possible BC breaker

- Url are now stored in the nodes and the router generate the cache according to these informations
- Symfony is installed in version 2.6.3

Bug fixes

- Multi-language and versioned content management

New features

- New form type: Color picker
- New block: Gallery
- New Twig helper: navigator to render a stylisable pagination

Other changes

- For the content list, we provide a method to disable object hydration
- Keyword

1.3 Key concepts

This document will explain the most important concepts to start with Open Orchestra.

1.3.1 Nodes

Nodes are mainly pages that the visitors can see on the website, they are containers for zones and blocks. The nodes are multilingual, versioned and have a validation workflow.

There are two types of nodes in Open Orchestra:

- Nodes that represent pages of the website
- The transverse node containing the transverse blocks (blocks that can be used in other pages, for instance the menu block). This node is not versioned, and there can only be one transverse node in each language of the site.

See also the documentation page about node parameters.

1.3.2 Content types

Content types are a way to create new kinds of content that will be available to contribution. Creating a content type allows to define the fields available in the future contents.

For instance, a 'news' content will have different fields from an 'article' content, so there would be two distinct content types for these objects : one for the news contents and another for the article contents.

See also Presentation of the Content Types.

1.3.3 Contents

With Open Orchestra, contents are used to display contributed information (articles, news, etc.). The contents are defined by a content type. Like nodes, contents are multilingual, versioned and have a validation workflow.

See also Presentation of contents.

1.3.4 Templates

Templates are used to preconfigure pages by setting zones that are commonly used across pages. Open Orchestra let the user start with a template when creating a node.

See also configuring a template.

1.3.5 Zones

Zones are the way to organize the architecture of the page. They are embedded in the nodes and can contain other zones or blocks.

See also configuring a zone.

1.3.6 Blocks

A block is the base brick representing any visible element on a page. Aggregating blocks found in the nodes will lead to the construction of the pages.

See also the list of blocks available.

1.3.7 Transverses blocks

Transverses blocks are defined inside the transverse node, they can be shared by several nodes of a website. All blocks can be transverse, as long as they are added to the transverse node. Transverse blocks can only be configured in the transverse node.

1.3.8 Themes

A theme represents the visual identity of a website, the CSS and JavaScript files that work together in order to render the websites in a certain way.

See how to add a theme.

1.3.9 Websites

Open Orchestra is multisite and allows to create several websites from the same Back Office application.

See also how to configure a website.

1.3.10 Devices

Open Orchestra is multi-device and is able to display, on the Front Office, different templates depending on the device of the visitor.

See also multi-devices.

1.3.11 Keywords

Keywords can be created in the Back Office, and then be used to tag contents and medias.

1.3.12 Users

Users represent people that can connect to the Open Orchestra Back Office and make contributions. They also are the Front Office users who can access to the private pages of a website. It's also possible to assign groups to users.

See also how to configure a user.

1.3.13 Roles

Roles allow to define authorization in the Back Office.

See also how to create a role.

1.3.14 Groups

Groups combine roles (this combination depends on the website) and are assigned to users. Groups can have several roles.

See also how to create a group.

1.3.15 Bundles

Open Orchestra is built on Symfony so the code is split into different bundles.

Open Orchestra's bundles :

- open-orchestra-base-bundle contains some transverse classes common to Back Office and Front Office.
- open-orchestra-cms-bundle is the application logic for the Back Office.
- open-orchestra-front-bundle is the application logic for the Front Office.
- open-orchestra-display-bundle contains all the block display strategies for the Front Office.
- open-orchestra-model-interface is a full description of the model classes used by other bundles.
- open-orchestra-model-bundle contains the database access logic (doctrinemongodb).
- open-orchestra-media-bundle contains the media functionalities.
- open-orchestra-user-bundle groups all user logic.

In order to use another database system one should had a new bundle which classes will implement the interfaces defined in open-orchestra-model-interface.

1.4 Website creation

Open Orchestra is multisite and allows to create several websites from the same Back Office application.

Creating a new site with Open Orchestra requires four steps. Additionally, you can add some more steps to enhance the site features. Those steps are the following:

- Declaring the website in Open Orchestra
- Dealing with access rights
- Contributing the home page
- Configuring the webserver
- (Optional) Configure the 404 & 503 pages
- (Optional) Generate the robots.txt
- (Optional) Generate the sitemap.xml

1.4.1 Declaring the site

The first thing to do is to declare the site in Open Orchestra by filling the website creation form.

Name (required)

The name field is the website name and is used to generate the website identifier.

Website id (required)

The website id can be generated by Open Orchestra from the site name or filled by the user when creating the site. Once the site created, this field is no more editable. Open Orchestra verifies during the form validation that the site id does not already exist as it must be unique among all the sites.

Aliases

A site may exist in different languages, and each of this versions could have minor differences, for instance a node could be published in English and not in French. Moreover, a site in English could be accessed from different domains. To make the management of those problematics easier, Open Orchestra introduces the concept of site alias. Basically a site represents a set of site aliases sharing contents and pages, each site alias in a single language. Multiple site aliases can be configured with the same language, if they define different access points, eg domain + language prefix.

For instance, the site http://demo.open-orchestra.com/ exists in english and french. It presents two site aliases:

- Scheme by default (required): http
- Domain (required): demo.openorchestra.com
- Language (required): English
- Language prefix : en
- main aliases : on
- Scheme by default (required): http
- Domain (required): demo.openorchestra.com

- Language (required): French
- Language prefix :
- main aliases : off

The website home page for english version is `http://demo.openorchestra.com/en` and `http://demo.openorchestra.com` for the french version.

See the routing documentation for further information.

Blocks available

This field lists all the blocks available to nodes contribution for this website. Only the selected blocks will be available.

Default Theme (required)

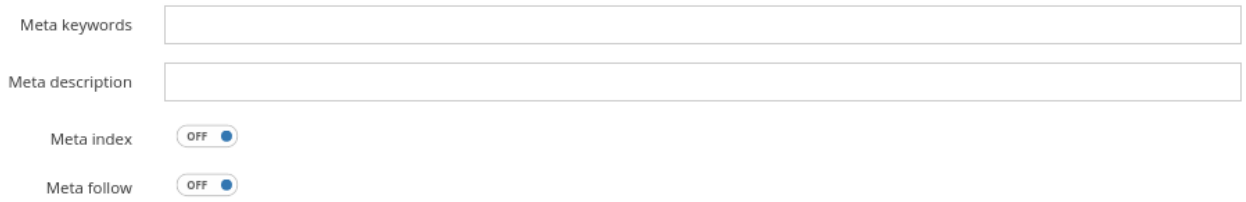
The field theme is the default theme used by the website. This setting can be overridden node by node in order to have a specific theme on a special node.

Sitemap generation (required)

The two following settings are related to the sitemap.xml generation. Those settings are used as default values for all nodes of the site, but they can be overridden node by node.

- Indicative periodicity of change of the page
- Relative importance compared to the other pages of the website

Meta



The meta attributes are set in the HTML header of a web page to provide information about the nature and content of the page. The settings contributed here are default values for each node of the site, but can be overridden node by node.

Meta keyword

This is a list of keywords associated to the page and can be used by search engines for ranking.

Meta description

This is used by search engines to display a description of the page in the results list.

Meta index

This attribute defines whether or not the page should be referenced in search engines.

Meta follow

This attribute indicates to search engines if they should follow hypertext links encountered in the page to index linked documents.

Information contained in the robots.txt file (required)

This field contains the data of the robots.txt file.

robots.txt file example

```
User-agent: * Allow: /
```

User-agent: * means that one or several instruction (s) which follow applies for all the agents. Allow: / means that the search engine can browse all the directories and the pages of the site.

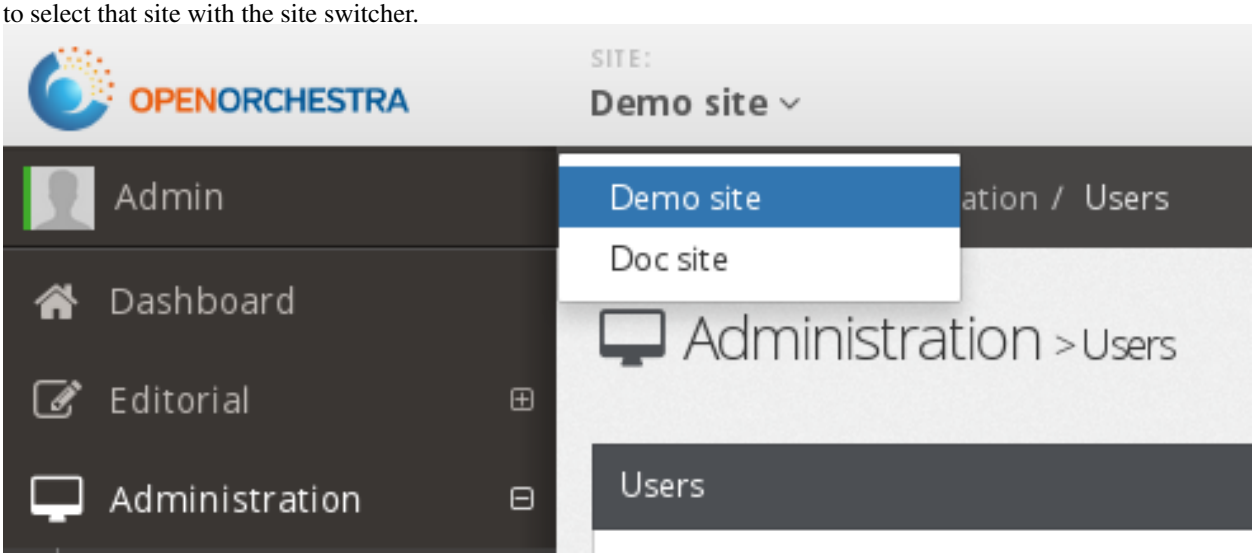
Homepage template

Template used to create the homepage of your new website.

1.4.2 Dealing with access rights

To be able to contribute to the site, access rights have to be set. To do this, create a new group including several roles related to this site (including ROLE_ACCESS_TREE_NODE) and assign it to the newly created site.

Once this is done, you can add users to the group by editing them. On their next connection, these users will be able to select that site with the site switcher.



Further information on group contribution is available in the user guide, on user page

1.4.3 Configuring the webserver

Now, Open Orchestra knows about your site and a first page is contributed. The last thing to do is to get that page reachable. To do this you have to configure your webserver. The type of configuration depends on you server type, for instance on Apache this is done via the Virtual Host mechanism. Here is a simple example in a such context:

Note that the DocumentRoot must point to the web directory of Open Orchestra installation, not on the Open Orchestra root directory.

Once Apache reloaded, your new website should be accessible.

Note: According to your environment, be sure to call the right front controller when testing the access (eg app_dev.php, app_prod.php, etc ...).

1.4.4 (Optional) Configure the 404 & 503 pages

At this stage, your site is visible. But it's the minimum and you can add several optional features. For instance, you can customize the 404 and 503 pages to map them to your site design. The documentation on error pages shows how to do that.

1.4.5 (Optional) Generate the robots.txt

Additionally, you can generate the robot.txt file according to the setting configured previously in the website form. See Robots.txt documentation for more information about this.

1.4.6 (Optional) Generate the sitemap.xml

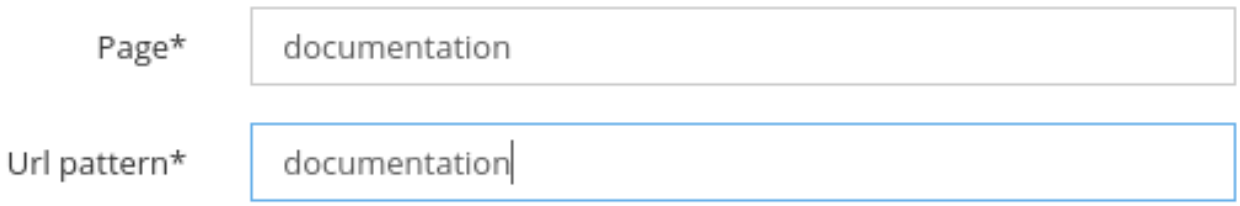
You may also want to generate the sitemap.xml of your site. To do this you can run a simple command line. See Sitemap.xml documentation for more information about this.

1.5 Node

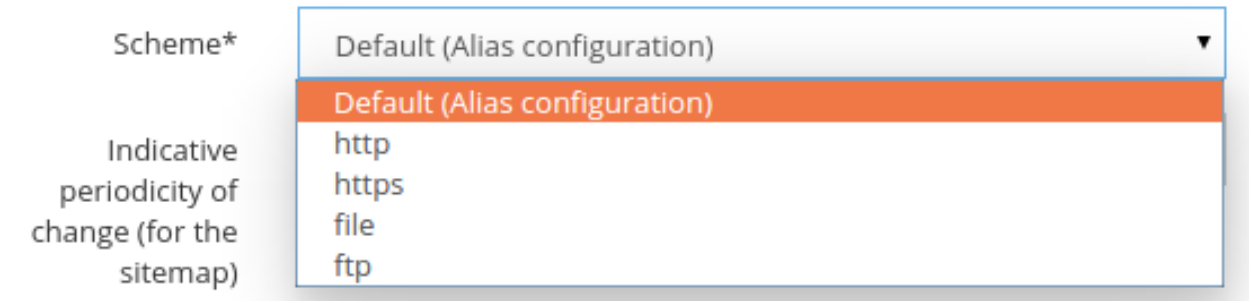
Nodes are mainly pages that the visitors can see on the website, they are containers for zones and blocks. The nodes are multilingual, versioned and have a validation workflow.

1.5.1 Node creation and Basic configuration

Contribute the name of the page and then chose an url pattern, it informs the url pattern which will allow to see this page on the site. The pattern can be also built automatically when the name of the page is contributed.



Choose a scheme for the page, to define how the page will be accessible (http, https, etc...). By default Open Orchestra takes the protocol informed in the site configuration.



Choose a theme to dress the page in Front Office.

During the creation of a node choose a template to create the structure of the page with preconfigured areas. This node can also copy an existing node.

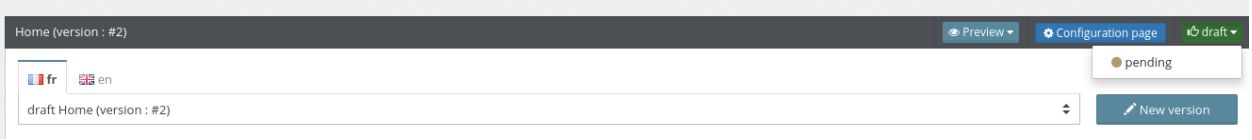
See also node configuration.

1.5.2 Languages

A node has several languages defined during the configuration of the Back Office. The node language being edited can be changed by clicking on the language tab. If this node has no data for this language, it will be created.

1.5.3 Workflow

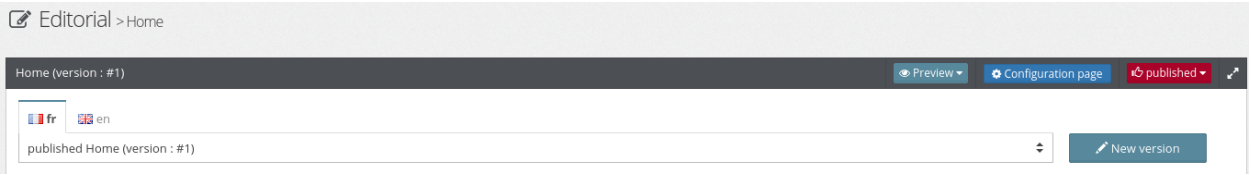
Nodes have a status and can have a validation workflow, which is the series of steps to achieve for publication of a node before it can be displayed on Front Office. Status of a node is linked to language of this one. Node have an initial status for this creation and to be able to display on Front Office the node has to have a final status.



To know more about status see also workflow.

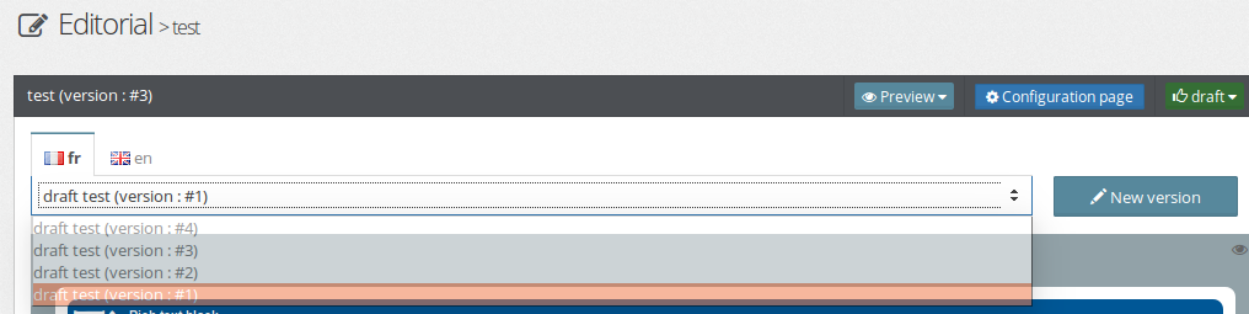
1.5.4 Versionning

A node have several versions. The node version is linked to language of this one. To create a new version of a node click on the “new version” button.



The latest version of the node is duplicated and the node is set to the initial status.

Then find all the versions of a node in the drop-down list.



The latest version of the node in the final status will be the one displayed in Front Office.

1.5.5 Managing blocks in nodes

Open Orchestra allows to add blocks in nodes. The visible list of blocks during the edition of a node is defined inside the site configuration.

See also node configuration and site configuration.

Nodes can use default blocks and global blocks. Those blocks have different colors. The global ones are editable only in the global node.

See also node and block transverse.

On the right appears the list of blocks, which is visible only if the node is alterable (not in the final status).

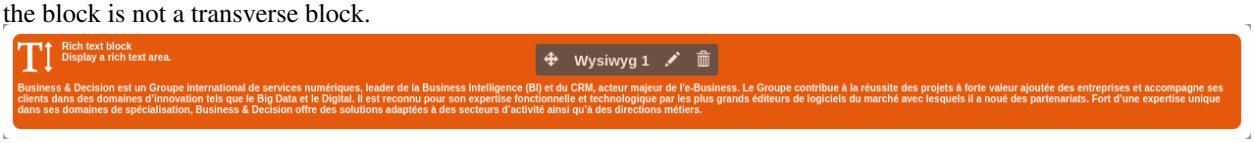

Blocks
Content List Block
Display a list of contents.
Content Block
Display a content.
Media list by keyword
Display a medias list.
Video Block
Insert a Youtube,
Dailymotion or Vimeo
Gmap Block
Display a map
Add this block
Display button to share
content
Audience analysis block
Integrate a Google
Analytics or Xiti tag
Wysiwyg 1
Rich text block
Display a rich text area.
Wysiwyg 2
Rich text block
Display a rich text area.
Menu

See also block list.

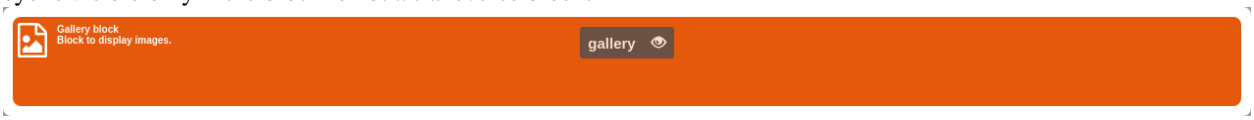
Drag and Drop

Open Orchestra allows to add or to move blocks in a node using drag and drop. To add a new block in a node open the block panel to display the list of blocks and drag a block in node's area. At the top of the list of blocks there are the new blocks and below transverse blocks.

If the mouse hovers a block, a toolbar will be displayed. When the node is alterable the toolbar contains a cursor, the block title, a pen and a trash can. To move a block in a node select the cursor and make it slide towards another area. To delete a block click on the trash can and to edit the block configuration click on the pen. The pen is visible only if the block is not a transverse block.



When the node is not alterable the toolbar contains only the block title and an eye to see the block configuration. The eye is visible only if the block is not a transverse block.



1.6 Node configuration

Here are the available options in the node configuration. Some of them are required in order to build a node and must therefore be provided.

1.6.1 Page (required)

The page attribute is the name of the node, it is also used to build the url pattern.

1.6.2 Url pattern (required)

The url pattern attribute is generated automatically with the name of the page. It represents the pattern that will be the URL of the page on the website.

If the pattern starts with a slash, it will be appended to the website's base url (domain + language prefix) to make the full page url. Otherwise, if the page has a parent, the pattern will be appended to the parent's URL to generate the final URL.

Here is a as example of some URL generated from the page patterns if the website's domain is "example.com" and the language is "en".

pattern	parent page pattern	final URL
/foo	/bar	example.com/en/foo
foo	/bar	example.com/en/bar/foo
foo	no parent page	example.com/en/foo

Some node url might contains variable to works. To add them, you should put the variable name between braces.

For instance, a page with content id required:

pattern	parent page pattern	final URL
{contentId}	no parent page	example.com/en/any-content-id

1.6.3 Scheme (required)

The scheme defines which protocol the page can be accessed by (http, https, etc...). By default Open Orchestra uses the value set in the site configuration.

1.6.4 Indicative periodicity of change

This attribute is used to optimize the sitemap data.

1.6.5 Relative importance compared to the other pages of the site

This attribute manages the importance of the page regarding the other pages. It is used in the sitemap generation.

1.6.6 The theme (required)

The theme to use on the page is chosen here.

1.6.7 Show in menus

This attribute indicates if the page should appear in the menus. It is used when rendering a menu block.

1.6.8 Show in footer

This attribute indicates if the page should appear in a footer. It is used when rendering a footer block.

1.6.9 Meta

The meta attributes provide information about the nature and content of a web page, they are added in the page HTML header with the use of meta tags. These attributes are also provided in the website configuration, which value is used if they are not overridden by the page configuration.

Meta keywords

This is a list of keywords associated to the page and can be used by search engines for ranking.

Meta description

This is used by search engines to display a description of the page in the results list.

Meta index

This attribute defines whether or not the page should be referenced in search engines.

Meta follow

This attribute indicates to search engines if they should follow the hypertext links on the page in order to index other documents.

1.6.10 Role needed to display the page

The role attribute defines which role is allowed to access the page. Pages are public by default, if a role is selected then only a user with the mandatory roles will be able to access the page.

1.6.11 Max age of the response for the node

This attribute is used to set up the lifetime of this page inside the cache.

1.6.12 Attributes specific to page creation

When creating a new page, one of the two following attributes must be used.

Source page

The source page attribute allows to create a new page by copying the content of an existing page.

Template id

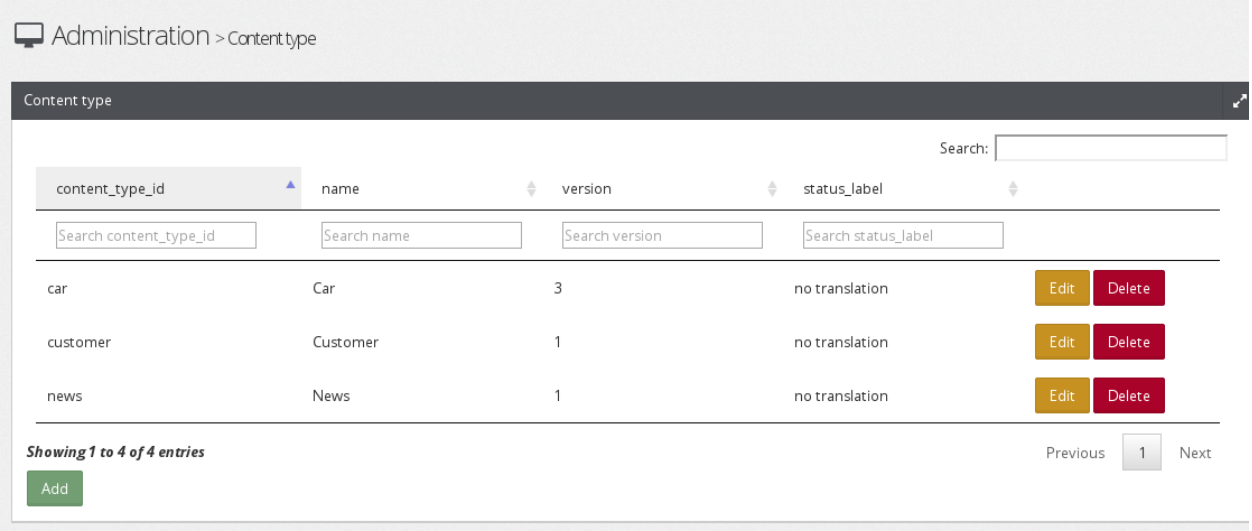
The template id allows to define a template that should be use as a basis for the page.

1.7 Content Types

In Open Orchestra Contents are typed, which means that every content has meaning, that it belongs to a family: an "Article" content is different from a "Customer" content. For instance, an article requires to contribute a title, a teaser, a picture and a text, while a customer requires instead a name, a first name, an email address and a phone number. Open Orchestra offers a RAD approach of the Content Types that does not need to write any code for this part: from the description of the Content Type to Content editing forms through storage in the database, everything is natively supported.

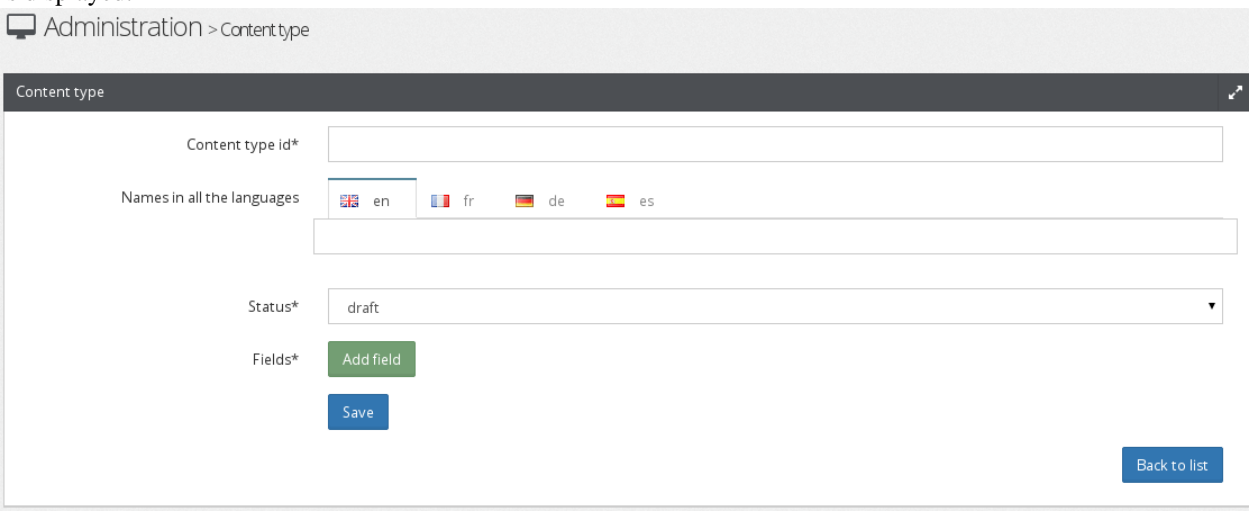
1.7.1 Listing

The list of existing Content Types is available in Administration part, under the Content Type entry. This list includes a row per Content Type and indicates its internal id, its name in the language of the Back Office, the number of the most advanced version as well as the status of this version.



1.7.2 Creation

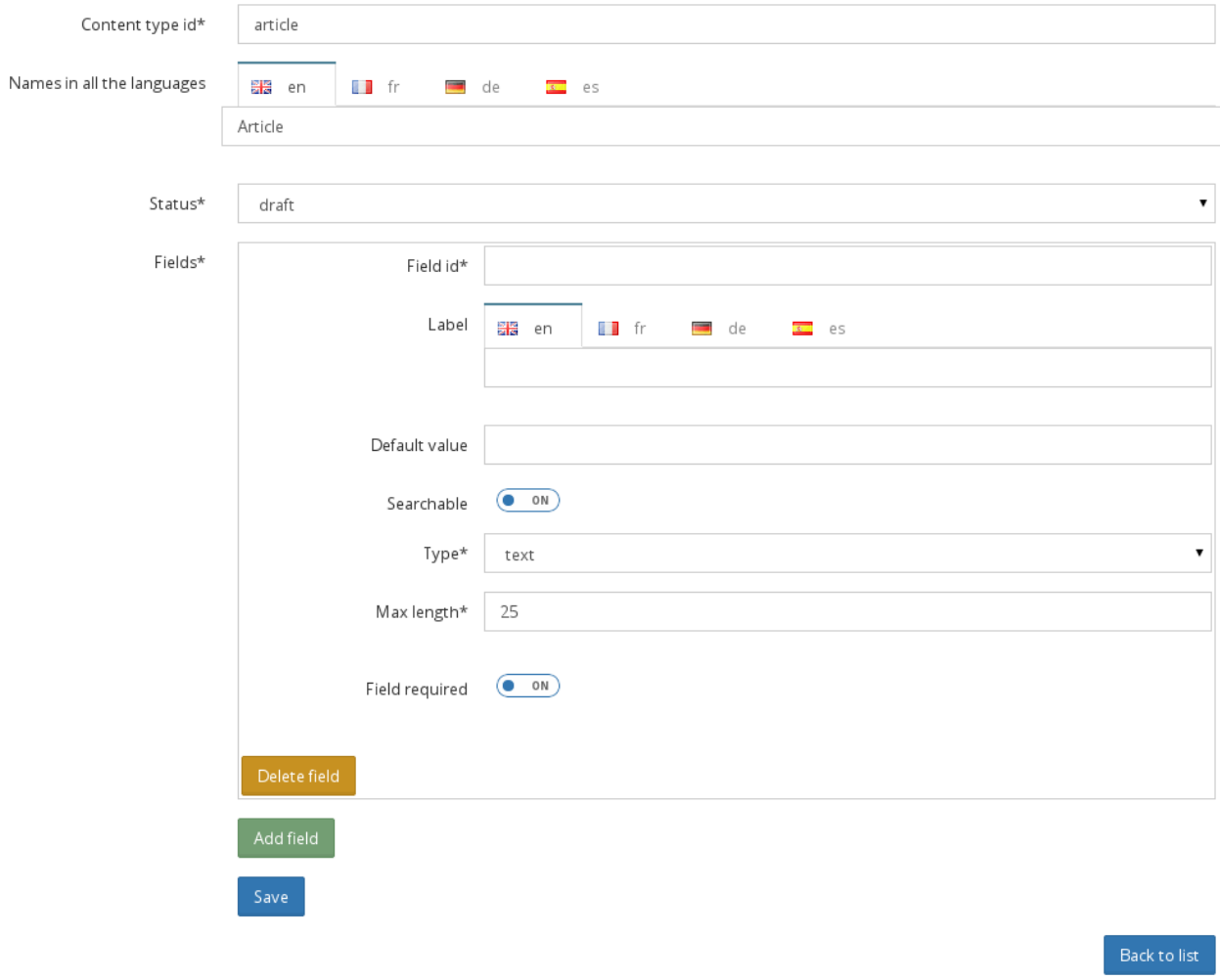
To create a new Content Type, from the list of Content Types, click the "Add" button. The Content Type creation form is displayed.



In the case we want to define the Content Type "Article", we should then contribute the following fields:

- **Content Type id**: that unique identifier allows Open Orchestra to distinguish our "Article" among others Content Type. Indicate "article"
- **Name in all the languages**: this is the name of the Content Type, displayed to contributors. It must be translated into the different languages of the Back Office. Indicate "Article" for both English and French
- **Status**: determines the status in which to create the Content Type. In our case, three status are available: draft, pending and published. Choose "draft" for this first version.

These three fields are systematically required when creating a Content Type, and identifies it among the others, but they do not characterize so far the "Article " Content Type. It remains to indicate that an article includes a title, a teaser, picture and text. This is done by individually adding each of these attributes. To add an attribute, click the "Add field" button. A new form box appears, specific to the first attribute, in our case the "title" of our "Article" Content Type.



- **Field id**: internal identifier, unique among all the attributes composing the Content Type. It is through this that Open Orchestra distinguishes the title of the article from its picture for example. Simply indicate "title"
- **Label**: the name of the attribute visible in the "Article" edit form. This label has to be translated into every language of the Back Office. Indicate "Title" in English and "Titre" in French.
- **Default value**: The default value of this attribute in the "Article" edit form. It would not have no sense to give a default title to our articles, so leave this field empty.
- **Searchable**: whether this attribute should be indexed by the internal search engine. For example, it would be useful for a user of the site to find our articles by typing some part of their titles in the search box. The titles of the articles must so be indexed, so put this value to "yes".
- **Type**: determines the type of the attribute. Is it a date, an image, a text line or a rich text block? In our case the title is a single line of text, so we choose "text". The end of the contribution of the attribute depends on the selected type. In the case of a "text", the maximum length allowed must be given. To have short titles, indicate 50.
- **Field required**: last information to set the title attribute. This is to indicate whether the attribute is required or

not when creating an article. Indeed, it is important that each article has a title, so indicate “yes”

This procedure must be repeated for each attributes characterizing the article. So you have to click again on “Add field” to describe the “image” attribute, again for the “teaser” attribute and one for the “text” attribute. Once all the attributes described, a click on the “Save” button saves the “Article” Content Type. It is then immediately possible to create articles and save them in the database.

1.7.3 Edition

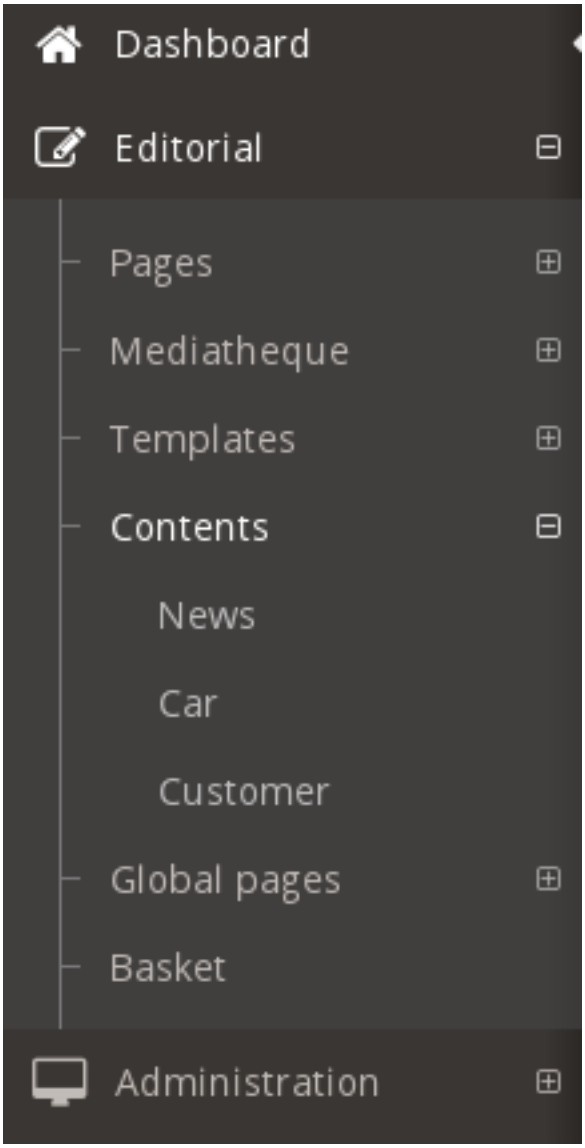
To edit an existing Content Type, from the list of Content Types, click the correct “Edit” button. The form used when creating the Content Type is displayed again, and it is possible to change what must be. Clicking the “Save” button will generate a new version of the Content Type, incorporating all changes.

1.8 Contents

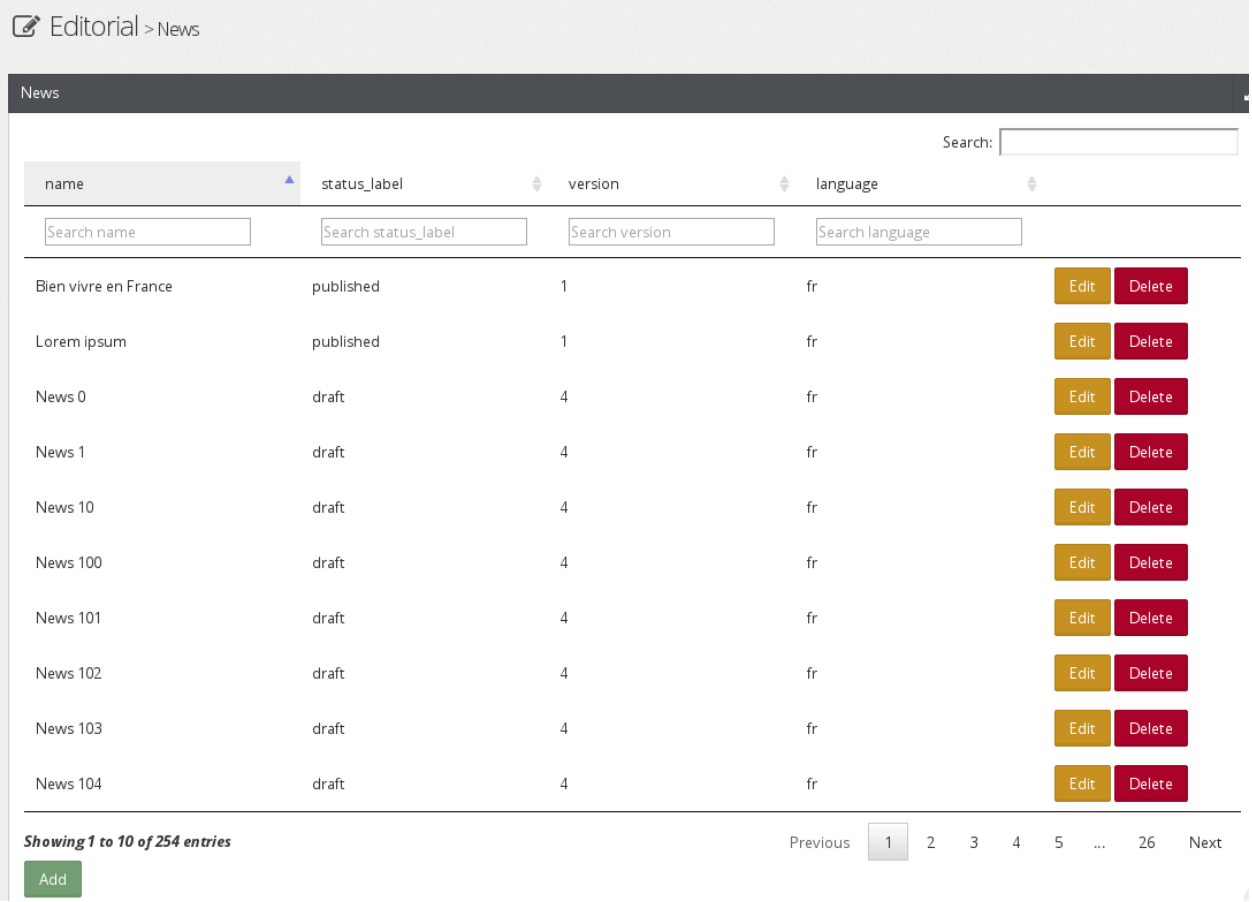
In Open Orchestra, Contents are grouped by families, a family being described by a Content Type. If “Car” was a Content Type, “Ford mustang” or “Dodge Viper” could be contents of that type. See Content Types documentation for more information.

1.8.1 Listing

Contents listing is accessed in the Editorial section, by clicking on the “Contents” link. A submenu lists all Content Types available.



By clicking a Content Type, the list of related contents appears.



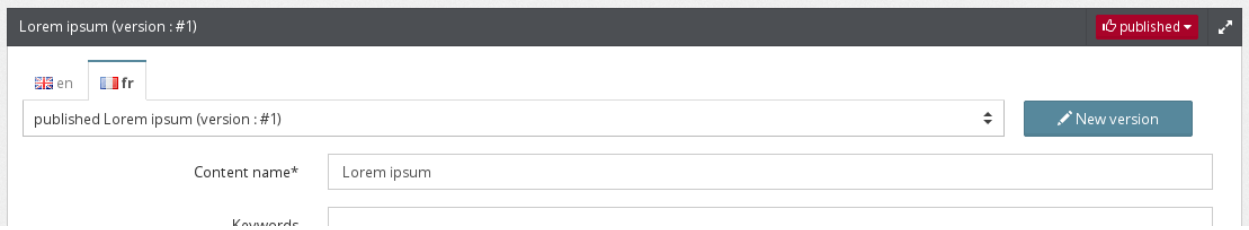
This list has a pagination and filters, and shows the name, the status, the version and the language of existing contents. From this list, new content can be created, and existing ones can be edited or deleted.

1.8.2 Creation

From the content list, creation form can be accessed by clicking on the "Add" button. The form will be related on the Content Type concerned. By saving the content is created, with the version number 1 and the initial status of the workflow.

1.8.3 Edition

From the content list, clicking on an "Edit" button makes an edit form appear. The edit form is mainly the same as the creation form, completed with current values, but has three more functionalities.



- **Status: The status button allows you to change the workflow status of the content (See the workflow** documentation for more information)

- **Translations**: the language tabs allow you to create/edit the content in alternative languages
- **Versioning: The version selector allows you to edit a previous version of the content in the** selected language. The "New version" button will duplicate the last version of the content in the selected language to create a new version.

Saving the modifications does not create a new version or change the status. Those operations have to be done individually.

1.8.4 Front display

The contents are not directly accessible in Front Office. Like everything visible in Front Office, the contents are displayed through blocks. See the documentation on the Contents display for more information about it.

1.9 Contents display

Once contributed, there is only one way to display a content: using content blocks in nodes. Open Orchestra comes with 3 block types able to render a content, but a developer can create his own block to match his needs. The three standard Open Orchestra content blocks are:

- Content list block
- Content block
- Configurable content block

1.9.1 Content list block

This block renders in Front Office a list of contents matching criteria given by the setting of the block. A typical use case may be the display of short previews for an article list, or a listing of customers.

Contribution

Besides the common settings for all blocks, the content list block requires two specific types of settings:

- Criteria: These are a combination of two basic criteria. The contributor must choose a content type and/or keyword(s) to trigger the contents.
- Rendering: The contributor has two ways to render this block. He can use the standard template of the content list block or create one in the Back Office to be used in that specific context, i.e. only for that block instance in that node.

Using the standard template

To use the standard template, put the template usage on Off.

Using a "one shot" template

To use a new specific and unique template, put the template usage on On, then use the rich text area below to create the template. You can use the twig syntax. This template will be used only for that block in that node. Other content list blocks in that node or in others are not been concerned by that template.

1.9.2 Content block

The content block renders the content whose id is given in the page address. A typical use is to create a generic node rendering specific content detailed information. By varying the content id in the url, the same node can be used to display various articles or customer details.

For instance a node with the pattern /customer/{contentId} would be accessed in Front Office via url /customer/paul-smith and then the content block would display some information about Paul Smith. It could also be accessed via url /customer/john-anderson, and then displaying information about John Anderson.

Contribution

On this block, like on the content list block, the contributor can use the standard template of the block or create a specific one for that block/node using the rich text.

To get this block working, the node url pattern has to be set in a specific way. As the content id comes from the url, the node pattern must include the string {contentId} or the content id will not be available. For more information about this, see node configuration.

1.9.3 Configurable content block

The configurable content block renders a single content, chosen by the contributor.

This block type could be used to create a focus on a specific content somewhere in the page, that could not be the principal information of the page, such has a mini-block on the last product developped by the company, or the employee of the month.

Contribution

To get the block working, the contributor must first choose a content type then choose the content to display.

1.10 Blocks natively available

Open Orchestra comes with a set of blocks that can be used as-is in a website and cover a wide range of use cases.

1.10.1 Content blocks

Content list

The content list block displays a list of elements belonging to a given content type (see the documentation page for contributing new contents). Usefull to list all the contributions for a specific content type that have been created.

Content

This block displays a single object of a specific content type. It's commonly used to be on the target page when a user clicks on an item of a content list block.

On the page where it is used, there must be a `contentId` parameter in the URL so the content to display can be retrieved.

Configurable content

Same as before, but here the content object to display will always be the same and will not rely on the URL. Therefore it is directly configurable in the block parameters.

Rich text

Using a rich text editor, this block can be told to render nicely formatted text. Usefull to display information that is not dynamic.

1.10.2 Media blocks

Carrousel

The carousel is a block that will display auto-changing images at a given frequency. The list of images that are displayed is chosen in the block configuration and must be part of the media library.

Gallery

The gallery block will display a set of images from the media library.

Video

This block can display a video hosted on YouTube, Dailymotion or Vine services.

Media list by keyword

Displays the medias from the library that are matching a given keyword.

1.10.3 Navigation blocks

Language list

This block provides a select box that lets the user change the current language in which the website is displayed. It will automatically display all languages available.

Menu

The menu displays links to the pages that are set to appear in the menu (see the page configuration). It manages the hierarchy between pages by using a submenus system if needed.

SubMenu

Similar to the previous one, except that only the children of a given page will be part of this menu. It's therefore possible to create a menu for only a subset of pages, different from you main menu.

Footer

Similarly to the Menu block, it will list all pages that have been set to appear in the footer.

1.10.4 Miscellaneous blocks

Login

A block used for login purposes that provides two fields : username and password.

AddThis

This block implements social sharing through the AddThis service.

Audience analysis

The audience block is designed to track users behavior with Xiti or Google Analytics services.

Gmap

The Gmap displays a google map centered on a preset geographic point.

Contact

In order for the visitors to contact the website administrator, this block renders a form and sends the visitor's message as an email to a configured electronic address.

1.11 The Media Library

With the media bundles (MediaAdminBundle, MediaBundle and MediaFileBundle), Open Orchestra gets a consistant media library. This library allows you to upload medias, manage them and reuse them on multiple locations of your site, such as in blocks or contents. Natively, images, PDF, sounds and videos are supported but it can be easily extended by a PHP developer to some other formats.

All files uploaded by users are organised inside folders. When the media library is activated, the navigation panel contains an entry to the media library presenting the media folders tree. You can see the content of a folder by clicking on it.

1.11.1 Create a folder

To create a folder, browse the folders tree where you want to create the folder and click on the "new folder" button. The creation form then appears and requires you to fill the following fields:

- Name: the name of the folder
- Websites: the folder you are creating will be available only for the sites you select in this multiselect

1.11.2 Remove a folder

To remove a folder, first select the folder. The gallery displaying the content of the folder then appears. You can remove that folder only if it is empty. If this is the case, the toolbar located on the top right hand side of the gallery will contain a trash icon. Click on this icon to delete the folder.

1.11.3 Manage files

To add a file, select the folder where you want to store it. Then on the bottom of the gallery, click the "add button". You can upload files in 3 different ways. The hole area is droppable, which means you can drag'n'drop files from your computer to upload them. You can also use the "Select a folder from your computer" to upload all files contained in that folder or only upload a file using the "Select a file from your computer" button. All files matching the types allowed in configuration, will be uploaded.

Returning on the gallery, you should see your media.

Putting your cursor over the preview makes a panel appear. This panel allows you to delete the media. Note that the media can not be deleted if it is used in a published element (block or content).

Clicking on the media preview opens a page to administrate the meta information of the media. If the media is an image, you can also alter some generated version.

Meta informations

You can administrate the following meta informations:

- Title: the title of the media in different languages
- Alt: the description of the media in different languages. It is used by the visitors browsers, especially for blinded monitor.
- Media's copyright: the copyright on the media
- Comment: A comment on the media
- Keyword: By tagging media, you can for instance regroup them with a common tag in a block

Image alternatives

When you upload an image, several alternatives in different sizes are automatically generated to improve the rendering in the front office. According to the context the image could be displayed as a thumbnail or as a big poster. Out of the box, Open Orchestra generat three different alternatives:

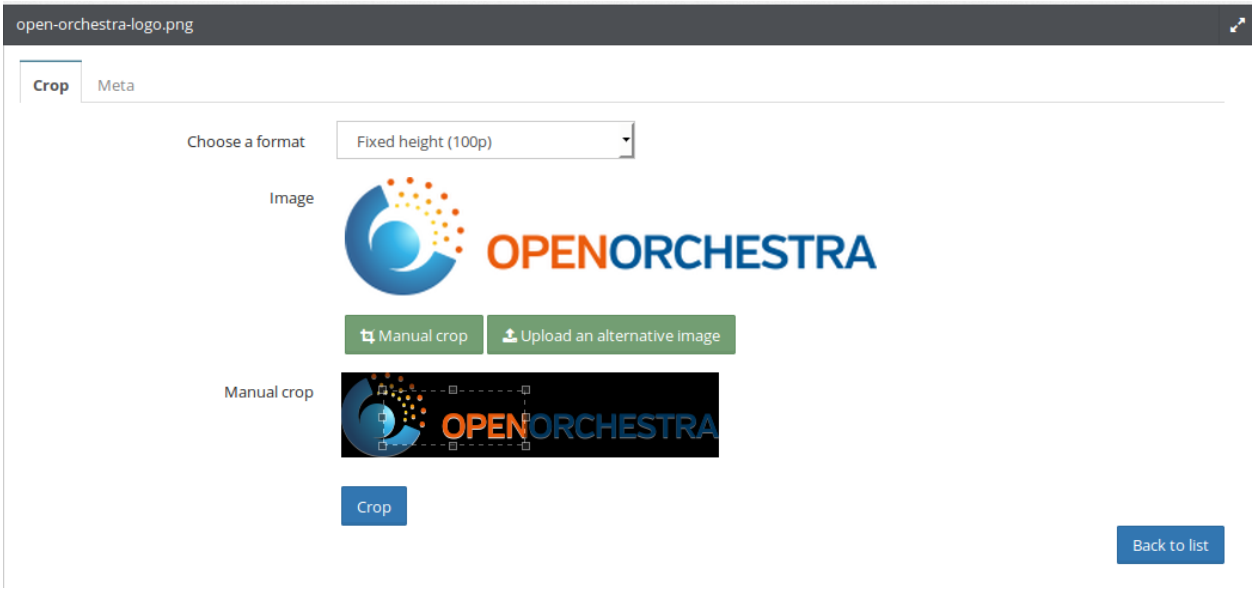
- Fixed height: Adapt height and keep proportion
- Fixed width: Adapt width and keep proportion
- Rectangle: Reduce to fit in the rectangle format, keeping the proportion

These ones can be personalized on a custom installation.

But you may want to alter an automatically generated alternative with a more specific image. You have two ways of doing that: crop manually the original image or upload a totally different alternative.

First thing to do this is to select the alternative you want to update. Then choose between:

- Manual crop: This will display the original image. You can select the part you want to keep. By saving your selection, Open Orchestra will resize it to fit the alternative format.
- Upload an alternative image: Simply upload a new image and it will replace the automatically generated version.



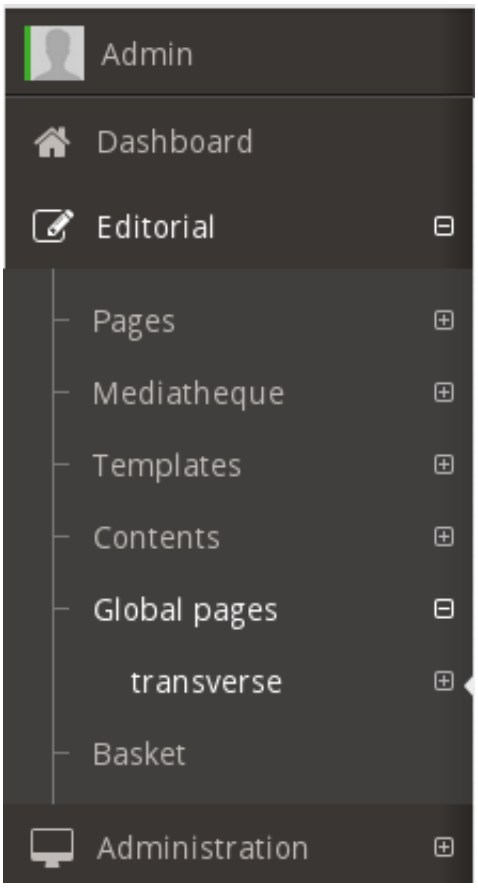
1.12 Global Pages and Blocks

On a webite, certain elements such as header and footer appear from pages to pages. On Open Orchestra it would be a pain to contribute again and again those elements in each node. To avoid this, Open Orchestra provide a way to contribute some part of a website once for all. This feature is delivered by the **Global pages** and the **Global blocks** functionalities.

A "Global block" is a block contributed once and that can be include in any node. Those blocks are contributed in a special node type that can be seen as a "Global block" repository : a "Global page".

1.12.1 Global pages

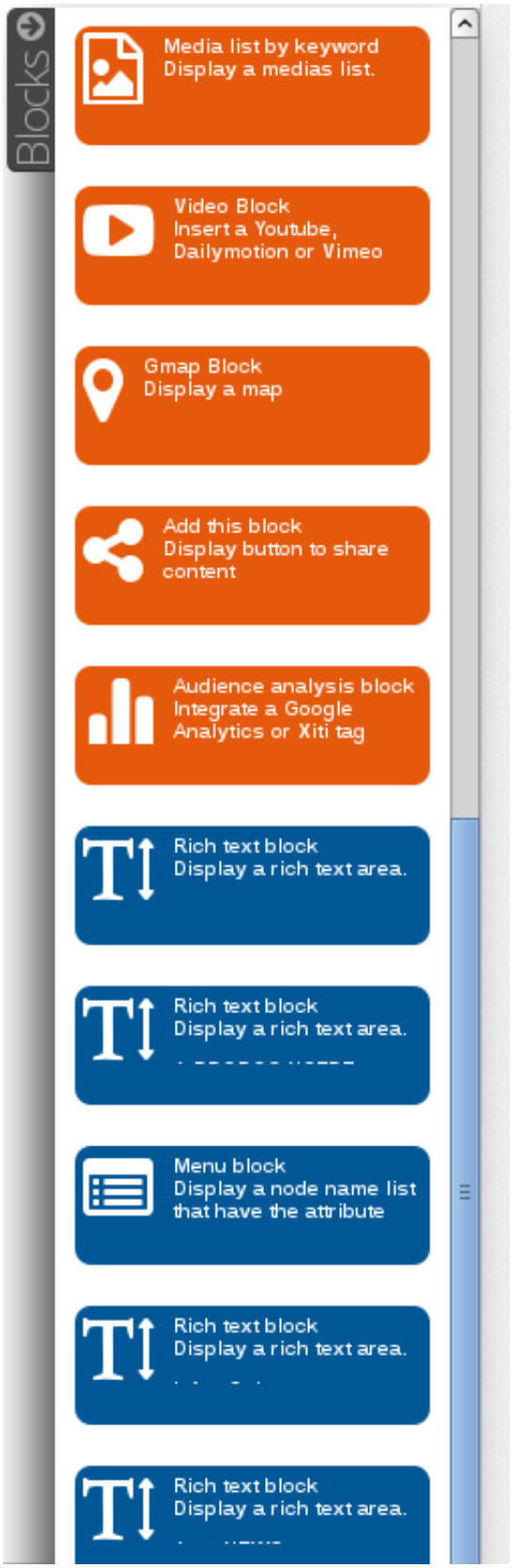
Global pages can be accessed through the main menu of the Back Office. They have several languages, don't have a workflow and have only one version.



Contribution of global pages works the same way as other pages (see Node).

1.12.2 Global blocks

When contributing a node, available blocks are shown in the block panel. Global blocks that have been contributed in global pages appear with a blue background, while standard blocks appear in orange.



Contribution of global blocks works the same way as other blocks (see managing block in node), except that they can

be edited only on the node they where created in, ie their global page.

1.13 Keyword management

Open Orchestra allows to attach keywords to content or media. They describe the content and allow to manage it. In the keyword panel, you can create or destroy a keyword. If you use a NoSql database and you destroy a keyword, it will still stay in the content where it is used.

1.13.1 Usage

When you fill the meta data for a content or media, already created keyword can be chosen from a list. Alternatively, you can create new ones using the `keyword` field.

The `content list block` and `media list by keyword` provide a way to filter the contents and medias by keywords.

1.14 User, Role and Group

Open Orchestra offers the possibility to manage users with different rights.

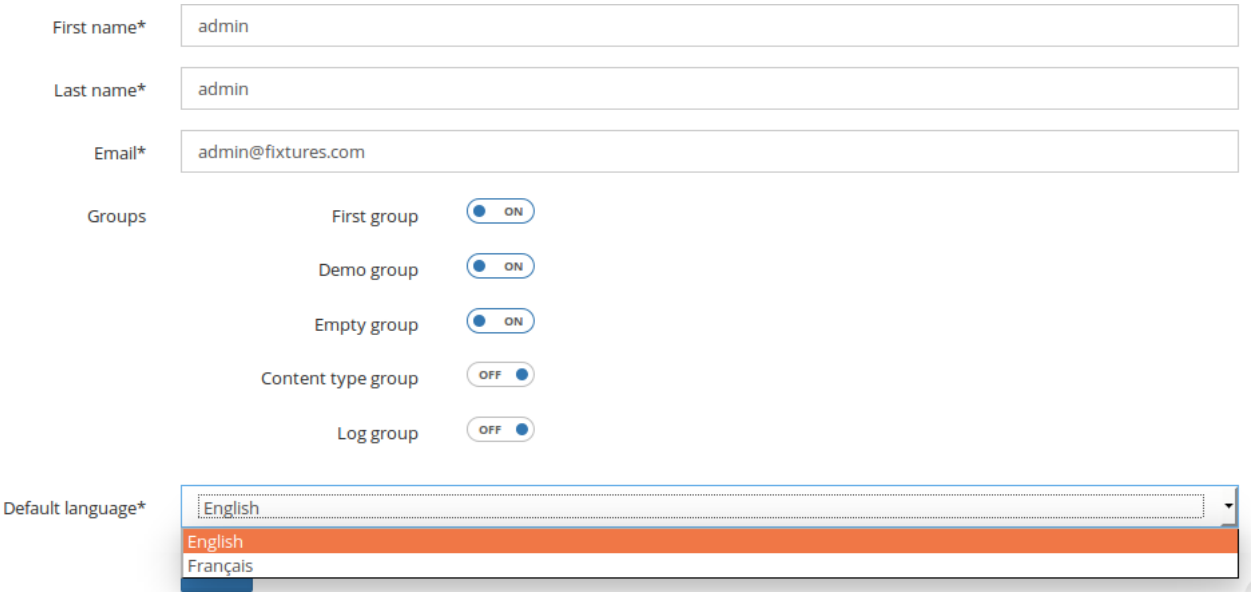
1.14.1 User Management

The users page allows to search, create, edit and delete users.

At the creation, some fields are required :

- **first Name**
- **last Name**
- **e-mail**
- **username**
- **password**

Once the user is created, you can add his default language and some groups.



1.14.2 Role Management

A role gives one type of authorization for a user.

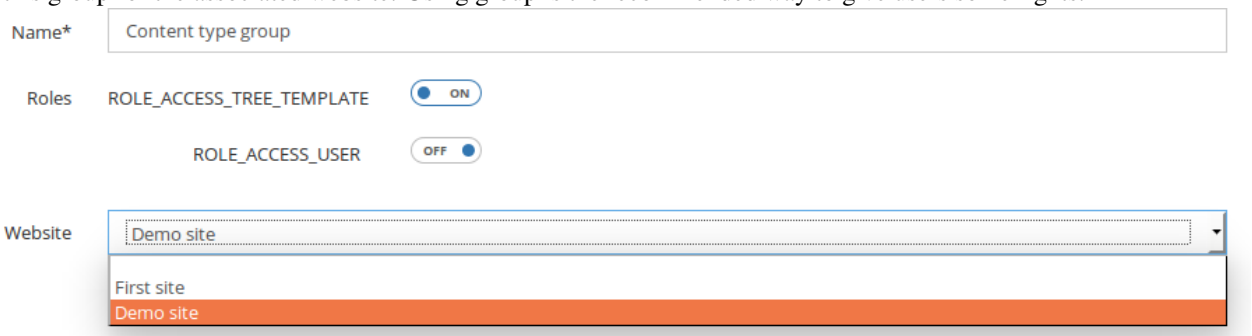
Roles contain:

- **name**
- **descriptions**: describe role in every languages
- **original/destination status**: optional parameters to define a workflow.

Some roles are already created in the fixture.

1.14.3 Group Management

Each group has a name, a list of roles and is associated to only one website. Users in a group inherit rights defined in this group for the associated website. Using group is the recommended way to give users some rights.



1.15 Roles

Some roles are natively used in Open Orchestra:

- **ROLE_USER**: grant access to back office
- **ROLE_ACCESS_USER**: grant access to users management
- **ROLE_ACCESS_TREE_TEMPLATE**: grant access to templates management
- **ROLE_ACCESS_KEYWORD**: grant access to keywords management
- **ROLE_ACCESS_LOG**: grant access to logs view
- **ROLE_ACCESS_REDIRECTION**: grant access to redirection management
- **ROLE_ACCESS_ROLE**: grant access to role management
- **ROLE_ACCESS_SITE**: grant access to websites management
- **ROLE_ACCESS_STATUS**: grant access to status management
- **ROLE_ACCESS_GROUP**: grant access to groups management
- **ROLE_ACCESS_THEME**: grant access to theme management
- **ROLE_ACCESS_TREE_FOLDER**: grant access to media library
- **ROLE_ACCESS_TREE_NODE**: grant access to page management
- **ROLE_ACCESS_API_CLIENT**: grant access to API client management
- **ROLE_ACCESS_CONTENT_TYPE**: grant access to content type management
- **ROLE_ACCESS_CONTENT_TYPE_FOR_CONTENT**: grant access to content management
- **ROLE_ACCESS_DELETED**: grant access to basket
- **ROLE_ACCESS_GENERAL_NODE**: grant access to global page management

1.16 Workflow

In Open Orchestra, key contributions such as nodes and contents are subject to a publication workflow. It allows several levels of validation by different people from creation to publication. Workflow are fully customizable.

A workflow is composed of steps, each corresponding to a status. Each status could be linked to an other by a transition called 'Role'. These roles are grouped in collections called 'Workflow Function'. Finally, in user's administration, it is possible to add a workflow function to a user on a given content type or on nodes.

For instance :

- three statuses : `draft`, `pending` and `published`
- two roles : `DRAFT_TO_PUBLISHED` (transition `draft` to `published`), `PENDING_TO_PUBLISHED` (transition `pending` to `published`)
- one workflow function: `Validator` (containing `DRAFT_TO_PUBLISHED` and `PENDING_TO_PUBLISHED`)
- a user with `Validator` function on `News` (content type)

There are two kinds of roles, those creating a link between a `from` and a `to status` and those without this statuses links. The first type defines workflow roles, the only type which could be used in workflow function. The second one defines access roles used to grant access or not to Back Office features. They are used in group context.

When adding a workflow function to a user on a particular content type or on the nodes, it is possible to choose if this function is activated only for the contributions of this user or for all contributions by checking `owned`.

1.16.1 Status

Each version of a node or a content has a status representing its state (for instance draft or published). There are 3 types of status :

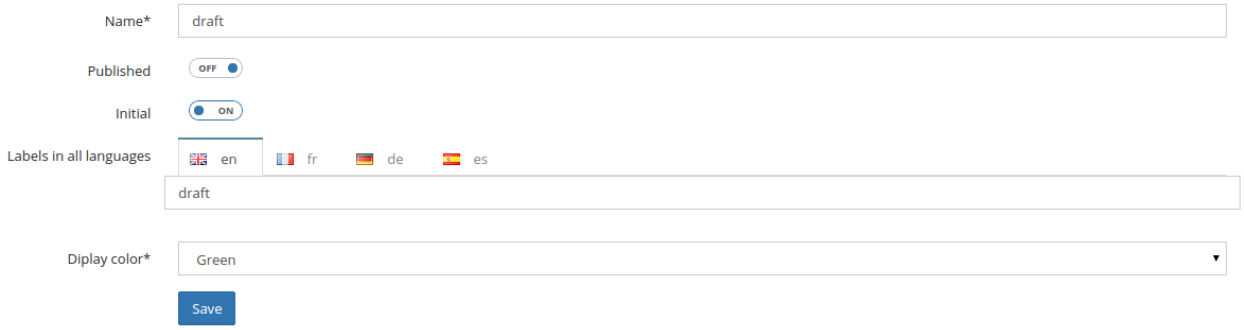
- An **initial** status is the beginning state of a workflow (automatically assigned to a node or a content just after its creation).
- A **final** status makes a content visible on the Front Office.
- All status that are not initial or final are **standard**

An administrator can define as many status as required to create his workflow.

Status creation

The status form asks for the following information:

- **Name** (required): status name
- **Published**: if this status is final
- **Initial**: if this status is initial
- **Label**: status label in all available languages in Back Office
- **Color** (required): color representing the status (used on the node or content edit page)

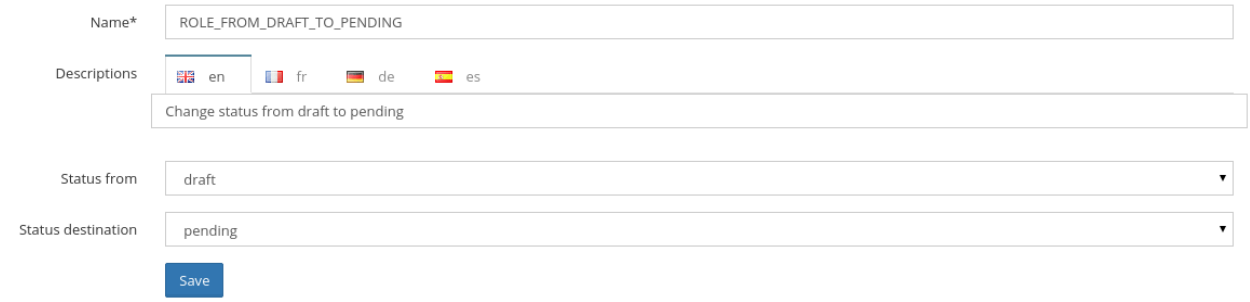


1.16.2 Roles

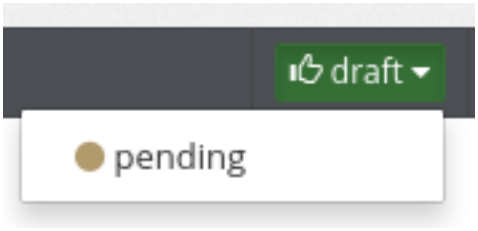
The role form asks for the following information:

- **Name**: role name
- **Description**: role description in all available languages in Back Office.
- **Status from**: as roles are transitions between two statuses, source status for the role
- **Status destination**: as roles are transitions between two statuses, destination status for the role

The selection of two statuses in the role form will define a workflow role.



This role creates a transition between Draft status and Pending status. So users with this role will be able to change the status of a node or content from Draft to Pending. They will do this via a small widget on the node or content edit page.



If no status is selected, the role will define an access role. See also group management

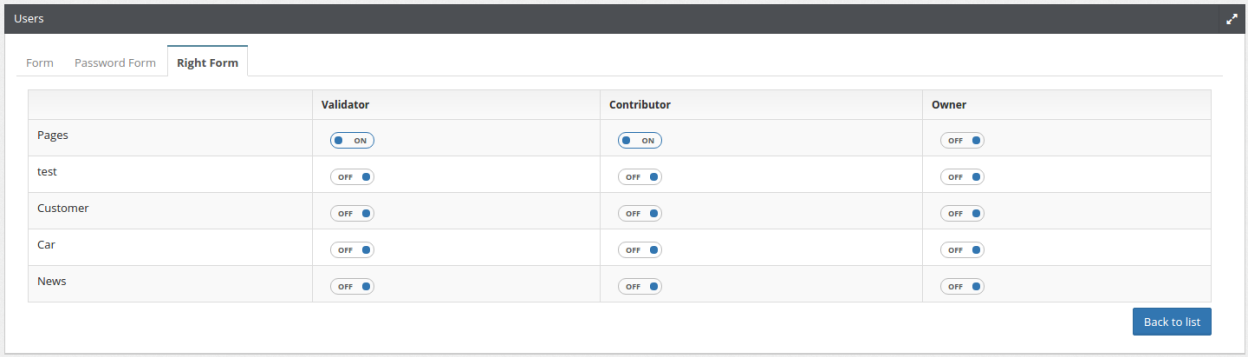
1.16.3 Workflow Function

The workflow function form asks for the following information:

- **Name**: workflow function name in all available languages in Back Office.
- **Role**: a list of all roles with a from and destination status, multiple selection allowed.

1.16.4 Workflow Rights

A double entry table asks for each content type or nodes and for each workflow function, if the selected user get this right. The last column indicates whether that right concerns only the contributions of this user or all contributions of this kind.



1.17 Boolean Expression

In some context, Open Orchestra authorize to use boolean expression to filter with keywords. In the content search block, for example, keywords are used to specify which contents must be showed.

This expression fits the following syntax:

- priority between the terms is specified by brackets,
- parenthesis, beginning the expression, is followed by a space,
- parenthesis, ending the expression, is preceded by a space,
- parenthesis, nor beginning, nor ending is preceded and followed by a space,
- parenthesis with no space will be considered as being in the attached keyword,
- brackets can not be empty,
- two operators are allowed, OR and AND,
- operator is preceded and followed by a space,
- expression can not beginning or ending with an operator,
- an operator cannot be immediately followed.

The whole expression can be typed or buttons can be used to enter brackets and operators.

If the expression is correctly formatted all brackets and operators will be in green color, else red.

1.18 Internal Link

Open Orchestra use TinyMce to edit rich text. A custom button is used to add an internal link, ie a link to a page managed by the Back Office. .. image:: ../../images/tiny_internal_link.png

This button opens a modal with a form to construct the internal link: .. image:: ../../images/form_internal_link.png

The form contains :

- Label: the label for the link displayed in Front Office
- Pages: the target of the link
- Content: as specified in the contents display documentation, content blocks need content id in the url. If the target of the link contains such a block, you can choose the id with this sub form. To simplify the choice of the

id, the content form contains search filters as content type and keywords (see boolean expression). A refresh button shows the filtered contents.

- Site: The site target of the link as Open Orchestra is multi-site. The site aliases of the choosen site could be seen by clicking the refresh button and one could be specified.
- Query string: optional parameters definition (param0=value0&param1=value1&param2=value2)

1.19 Install OpenOrchestra on your computer

Open Orchestra is composed of three projects : *open-orchestra* which is the CMS Back Office and *open-orchestra-front-demo* which is the Front Office part that will display the sites and pages created in the CMS Back Office and *open-orchestra-media-demo* which will display the images loaded in the CMS Back Office.

This tutorial has been tested on a debian 8.2.

1.19.1 Install the environment

Install the php server

Install php5 and the mongo extension :

```
$ sudo aptitude install php5-cli php5-dev php-pear php5-curl php5-imagick libav-tools
$ sudo pecl install mongo
```

Activate the mongo extension in php :

```
$ sudo echo "extension=mongo.so" > /etc/php5/mods-available/mongo.ini
$ sudo ln -s /etc/php5/mods-available/mongo.ini /etc/php5/cli/conf.d/30-mongo.ini
```

Install the database

Install mongo in the 2.6 version at least :

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
$ echo 'deb http://downloads-distro.mongodb.org/repo/debian-sysvinit dist 10gen' | sudo tee /etc/apt/
$ sudo aptitude update
$ sudo aptitude install mongodb-org
```

Check the mongo version :

```
$ mongo --version #MongoDB shell version: 2.6.*
```

Install NPM

As a prerequisite, install those packages :

```
$ sudo aptitude install wget curl make g++ gcc libcurl4-openssl-dev libsasl2-2 libsasl2-dev libcurl3
```

Download and install npm :

```
$ wget http://nodejs.org/dist/v0.10.24/node-v0.10.24.tar.gz ./node-v0.10.24.tar.gz
$ tar -xvzf node-v0.10.24.tar.gz
$ cd node-v0.10.24
$ sudo ./configure --prefix=/usr/local/ && sudo make && sudo make install
```

1.19.2 Install the project

Download Composer

Composer is the package manager used by modern PHP applications.

To install composer with curl:

```
$ sudo aptitude install git
$ curl -sS https://getcomposer.org/installer | php
```

If you don't have curl installed, you can also download it with php:

```
$ php -r "readfile('https://getcomposer.org/installer');" | php
```

see [Download Composer](https://getcomposer.org/download/)

Install OpenOrchestra

Install open-orchestra with composer:

```
$ ./composer.phar create-project open-orchestra/open-orchestra path/to/your/folder -s dev
$ ./composer.phar create-project open-orchestra/open-orchestra-front-demo path/to/your/folder -s dev
$ ./composer.phar create-project open-orchestra/open-orchestra-media-demo path/to/your/folder -s dev
```

1.19.3 Install the assets

Launch the grunt command to generate all assets

```
$ ./bin/grunt
```

1.19.4 Load the fixtures

Open Orchestra needs some fixtures to work (an admin user, a website, ...).

```
$ php app/console orchestra:mongodb:fixtures:load --type=production --env=prod
```

Launch the server

You can use the built in webserver to launch the backoffice of Open Orchestra :

```
$ php app/console server:run
```

Go on the homepage : `http://127.0.0.1:8000`

For the front office, you need to install apache and configure the server

1.20 Install OpenOrchestra with vagrant

Open Orchestra is composed of three projects : *open-orchestra* which is the CMS Back Office and *open-orchestra-front-demo* which is the Front Office part that will display the sites and pages created in the CMS Back Office and *open-orchestra-media-demo* which will display the images loaded in the CMS Back Office.

For developers, we provide a Vagrant-powered environment with provisioning so you get minimal setup actions to do.

1.20.1 Install Vagrant

The project is running on a Vagrant virtual environment built on VirtualBox to be production ready. For more information about Vagrant installation, you can see ‘ Vagrant installation documentation <https://www.vagrantup.com/docs/>‘

1.20.2 Install ansible

All the project server configuration is going to be handled by ansible.

Our configuration was made for the 1.9.4 version of ansible. **It doesn’t work with the version 2 of ansible**.

For more information about ansible installation, you can see Ansible installation documentation.

If you need *tar.gz* archives, you can see this link : https://releases.ansible.com/ansible/

1.20.3 Install nfs server

To improve the vagrant box performance, we share the folders with the `nfs` protocol. You need to install a nfs server instance on your computer.

```
$ aptitude install nfs-kernel-server
```

1.20.4 Download Composer

Composer is the package manager used by modern PHP applications.

To install composer with curl:

```
$ curl -sS https://getcomposer.org/installer | php
```

If you don’t have curl installed, you can also download it with php:

```
$ php -r "readfile('https://getcomposer.org/installer');" | php
```

see Download Composer

1.20.5 Install OpenOrchestra

Install the different open-orchestra part using `composer`:

```
$ ./composer.phar create-project open-orchestra/open-orchestra path/to/your/folder -s stable 1.1.x
$ ./composer.phar create-project open-orchestra/open-orchestra-front-demo path/to/your/folder -s stab
$ ./composer.phar create-project open-orchestra/open-orchestra-media-demo path/to/your/folder -s stab
```

Clone the provisioning repository in another folder :

```
$ git clone https://github.com/open-orchestra/open-orchestra-provision.git --branch=1.1
```

1.20.6 Install roles from ansible-galaxy

Go into `open-orchestra-provisioning` directory and install roles needed to launch the box:

```
$ ansible-galaxy install --role-file=galaxy.yml
```

1.20.7 Override the dns redirection

In the `/etc/hosts` file of your computer add the following lines :

```
192.168.33.10   admin.openorchestra.1-1.dev
192.168.33.10   demo.openorchestra.1-1.dev
192.168.33.10   media.openorchestra.1-1.dev
```

1.20.8 Launch the box

In the `open-orchestra` directory, when you launch the box, it will take some time to :

- Import the base box
- Launch it
- Run all the provisioning scripts

```
$ vagrant up
```

1.20.9 Install the assets

We are using npm to manage some server side javascript libraries and bower to manage the client side libraries

Connect to the vagrant box using `vagrant ssh`

Finalise the `composer` installation in each project

```
$ cd /var/www/openorchestra && composer run-script post-install-cmd
$ cd /var/www/front-openorchestra && composer run-script post-install-cmd
$ cd /var/www/media-openorchestra && composer run-script post-install-cmd
```

Then go in the Back Office project directory inside the box

```
$ cd /var/www/openorchestra
```

Launch the grunt command to generate all assets

```
$ ./bin/grunt
```

1.20.10 Load the fixtures

In the symfony project directory `/var/www/openorchestra` you can load the fixtures provided :

```
$ php app/console orchestra:mongodb:fixtures:load --type=production --env=prod
```

Now you can log on http://admin.openorchestra.1-1.dev/login with username=admin and password=admin for the CMS and see the result on http://demo.openorchestra.1-1.dev.

All the images will be visible on the http://media.openorchestra.1-1.dev url.

1.21 Install OpenOrchestra with vagrant for contributors

Open Orchestra is composed of three projects:

- *open-orchestra* which is the CMS Back Office;
- *open-orchestra-front-demo* which is the Front Office part that will display the sites and pages created in the CMS Back Office;
- *open-orchestra-media-demo* which will display the images loaded in the CMS Back Office.

For contributors, we provide a Vagrant-powered environment with provisioning so you get minimal setup actions to do. Install Vagrant ———— The project is running on a Vagrant virtual environment built on VirtualBox to be production ready. For more information about Vagrant installation, you can see ‘ Vagrant installation documentation <https://www.vagrantup.com/docs/>‘

1.21.1 Install ansible

All the project server configuration is going to be handled by ansible.

Our configuration was made for the 1.9.4 version of ansible. **It doesn’t work with the version 2 of ansible**.

For more information about ansible installation, you can see Ansible installation documentation.

If you need *tar.gz* archives, you can see this link : https://releases.ansible.com/ansible/

1.21.2 Install nfs server

To improve the vagrant box performance, we share the folders with the `nfs` protocol. You need to install a nfs server instance on your computer.

```
$ aptitude install nfs-kernel-server
```

1.21.3 Download Composer

Composer is the package manager used by modern PHP applications.

To install composer with curl:

```
$ curl -sS https://getcomposer.org/installer | php
```

If you don’t have curl installed, you can also download it with php:

```
$ php -r "readfile('https://getcomposer.org/installer');" | php
```

see Download Composer

1.21.4 Install OpenOrchestra

To contribute on the project you have to install two versions: the 1.1 version and the 1.0 version . The 1.1 version is the branch which should be used to add new features to the project. Here is the directory tree with your two projects that we recommand:

```
|_ install-directory
      |_ open-orchestra-1.1
            |_ open-orchestra
            |_ open-orchestra-front-demo
            |_ open-orchestra-media-demo
            |_ open-orchestra-provision
      |_ open-orchestra-1.0
            |_ open-orchestra
            |_ open-orchestra-front-demo
            |_ open-orchestra-media-demo
            |_ open-orchestra-provision
```

- To install the master open-orchestra parts using `composer`:

  In your `open-orchestra-1.1` directory:

```
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra ./open-orchestra
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-front-demo ./ope
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-media-demo ./ope
```

  Clone the 1.1 provisioning repository. Don't forget to specify the last 1.1 version branch with the `--branch` option.

```
$ git clone git@github.com:open-orchestra/open-orchestra-provision.git --branch=1.1
```

- To install the 1.0 version open-orchestra parts using `composer`:

  In your `open-orchestra-1.0` directory:

```
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra ./open-orchestra
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-front-demo ./ope
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-media-demo ./ope
```

  Clone the 1.0 provisioning repository. Don't forget to specify the last 1.0 version branch with the `--branch` option.

```
$ git clone git@github.com:open-orchestra/open-orchestra-provision.git --branch=1.0
```

1.21.5 Override the dns redirection

In the `/etc/hosts` file of your computer add the following lines:

```
192.168.33.10   admin.openorchestra.1-1.dev
192.168.33.10   demo.openorchestra.1-1.dev
192.168.33.10   media.openorchestra.1-1.dev

192.168.33.11   admin.openorchestra.1-0-dev
192.168.33.11   demo.openorchestra.1-0-dev
192.168.33.11   media.openorchestra.1-0-dev
```

You should follow the same steps to install each versions :

- Install roles from ansible-galaxy

- Launch the box
- Install the assets
- Load the fixtures

1.21.6 Install roles from ansible-galaxy

Go into `open-orchestra-provisioning` directory and install roles needed to launch the box:

```
$ ansible-galaxy install --role-file=galaxy.yml
```

1.21.7 Launch the box

In the `open-orchestra` directory, when you launch the box, it will take some time to:

- Import the base box
- Launch it
- Run all the provisioning scripts

```
$ vagrant up
```

1.21.8 Install the assets

We are using npm to manage some server side javascript libraries and bower to manage the client side libraries.

Connect to the vagrant box using `vagrant ssh`

Finalise the `composer` installation in each project:

```
$ cd /var/www/openorchestra && composer run-script post-install-cmd
$ cd /var/www/front-openorchestra && composer run-script post-install-cmd
$ cd /var/www/media-openorchestra && composer run-script post-install-cmd
```

Then go in the Back Office project directory inside the box:

```
$ cd /var/www/openorchestra
```

If you are dealing with version 1.1, launch the grunt command to generate all assets:

```
$ ./bin/grunt
```

If you are dealing with the 1.0 version, the process differs, you have to install manually the dependencies then when it's ok, run grunt from the node_modules folder:

```
$ npm install
$ ./node_modules/.bin/grunt
```

1.21.9 Load the fixtures

In the symfony project directory `/var/www/openorchestra` you can load the fixtures provided:

```
$ php app/console doctrine:mongo:fixture:load --env=dev
```

1.21.10 Result

Master version

You can log on http://admin.openorchestra.1-1.dev/app_dev.php/login with username=admin and password=admin for the CMS and see the result on http://demo.openorchestra.1-1.dev/app_dev.php.

All the images will be visible on the http://media.openorchestra.1-1.dev/app_dev.php url.

Stable version

You can log on http://admin.openorchestra.1-0.dev/app_dev.php/login with username=admin and password=admin for the CMS and see the result on http://demo.openorchestra.1-0.dev/app_dev.php.

All the images will be visible on the http://media.openorchestra.1-0.dev/app_dev.php url.

1.22 Install OpenOrchestra with Docker for contributors

Open Orchestra is composed of three projects:

- *open-orchestra* which is the CMS Back Office;
- *open-orchestra-front-demo* which is the Front Office part that will display the sites and pages created in the CMS Back Office;
- *open-orchestra-media-demo* which will display the images loaded in the CMS Back Office.

For contributors, we provide a Docker environment so you get minimal setup actions to do.

1.22.1 Install Virtualbox (for MacOS and Windows)

Mac OS

- download http://download.virtualbox.org/virtualbox/4.3.20/VirtualBox-4.3.20-96996-OSX.dmg

Windows

- download http://download.virtualbox.org/virtualbox/4.3.20/VirtualBox-4.3.20-96997-Win.exe

1.22.2 Install Docker

The project is running in a virtual environment to be production ready.

Linux

```
$ sudo apt-get update
$ sudo apt-get install wget
$ sudo apt-get install curl
$ wget -qO- https://get.docker.com/ | sh
```

Note: If your company is behind a filtering proxy, you may find that the apt-key command fails for the Docker repository during installation. To work around this, add the key directly using the following:

```
$ wget -qO- https://get.docker.com/gpg | sudo apt-key add -
```

Create docker group

```
$ sudo usermod -aG docker ubuntu
```

Install docker compose

```
$ curl -L https://github.com/docker/compose/releases/download/1.5.2/docker-compose-`uname -s`-`uname
$ chmod +x /usr/local/bin/docker-compose
```

Mac OS

```
download https://www.docker.com/docker-toolbox (mac version)
install the package
open a terminal (e.g. http://cmder.net/)
```

type the commands

```
$ docker-machine create default --driver=virtualbox
$ eval $(docker-machine env default)
```

Windows

```
download https://www.docker.com/docker-toolbox (mac version)
install the package
open a terminal (e.g. http://cmder.net/)
```

type the commands

```
$ docker-machine create default --driver=virtualbox
$ eval $(docker-machine env default)
```

1.22.3 Download Composer

Composer is the package manager used by modern PHP applications.

To install composer with curl:

```
$ curl -sS https://getcomposer.org/installer | php
```

If you don't have curl installed, you can also download it with PHP:

```
$ php -r "readfile('https://getcomposer.org/installer');" | php
```

see Download Composer

1.22.4 Install OpenOrchestra

First of all, you have to clone the Open Orchestra Docker repository:

```
    $ git clone git@github.com:open-orchestra/open-orchestra-provision-docker.git
```

To contribute to the project a script can install two versions: the master version and the last stable one. The master branch is the branch which should be used to add new features to the project. Here is the directory tree with your two projects:

```
|_ open-orchestra-provision-docker
      |_ open-orchestra-master
            |_ open-orchestra
```

```
            |_ open-orchestra-front-demo
            |_ open-orchestra-media-demo
            |_ open-orchestra-provision
      |_ open-orchestra-stable
            |_ open-orchestra
            |_ open-orchestra-front-demo
            |_ open-orchestra-media-demo
            |_ open-orchestra-provision
```

1. Easy way

You can run all the commands below with one script :

```
$ ./install.sh master
or
$ ./install.sh stable
```

2. Manual way

To install the master open-orchestra parts using `composer`:

In your `open-orchestra-master` directory:

```
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra ./open-orchestra
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-front-demo ./ope
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-media-demo ./ope
```

Build Docker containers :

```
$ docker-compose -f docker-compose.yml -f docker-compose-master.yml up -d
```

- To install the stable open-orchestra parts using `composer`:

In your `open-orchestra-stable` directory:

```
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra ./open-orchestra
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-front-demo ./ope
$ [path-to-composer]/composer.phar create-project open-orchestra/open-orchestra-media-demo ./ope
```

Build Docker containers :

```
$ docker-compose -f docker-compose.yml -f docker-compose-stable.yml up -d
```

1.22.5 Parameters to define at the end of composer install

```
open_orchestra_cms.mongodb.host : mongo
fos_http_cache.proxy_client.varnish.servers : [varnish:6081]
host_elastica : elastica
```

1.22.6 Override the DNS redirections

In the `/etc/hosts` file of your computer add the following lines:

[IP] must be replaced by 127.0.0.1 for Linux [IP] must be replaced by the value gived by the command `docker-machine ip default`

```
[IP]   admin.openorchestra.dev
[IP]   demo.openorchestra.dev
[IP]   media.openorchestra.dev
[IP]   admin.openorchestra.stable
[IP]   demo.openorchestra.stable
[IP]   media.openorchestra.stable
```

You should follow the same steps to install each versions :

- Run the containers
- Install the assets
- Load the fixtures

1.22.7 Run the init script

To finish the insytallation, you must launch an init script inside the main container by using this command:

```
    $ docker exec -it app_open_orchestra_master /load.sh
    or
    $ docker exec -it app_open_orchestra_stable /load.sh
```

It will take some time to:

- prepare the cache and logs directories inside the App container
- NPM Install (it's the longer process of the procedure)
- Composer install
- Load fixtures
- Create ElasticSearch Indexes and populate data
- Launch Grunt to install the assets

1.22.8 Result

Master version

You can log on http://admin.openorchestra.dev/app_dev.php/login with username=admin and password=admin for the CMS and see the result on http://demo.openorchestra.dev/app_dev.php.

All the images will be visible on the http://media.openorchestra.dev/app_dev.php url.

Stable version

You can log on http://admin.openorchestra.stable/app_dev.php/login with username=admin and password=admin for the CMS and see the result on http://demo.openorchestra.stable/app_dev.php.

All the images will be visible on the http://media.openorchestra.stable/app_dev.php url.

1.23 Installation Guide

- Install OpenOrchestra on your computer
- Install OpenOrchestra with vagrant
- Install OpenOrchestra with vagrant for contributors
- Install OpenOrchestra with Docker for contributors

1.24 Dependancy diagram

As the project evolves, we want to continuously present you all the interaction between the Open Orchestra bundles.

1.24.1 Current diagram

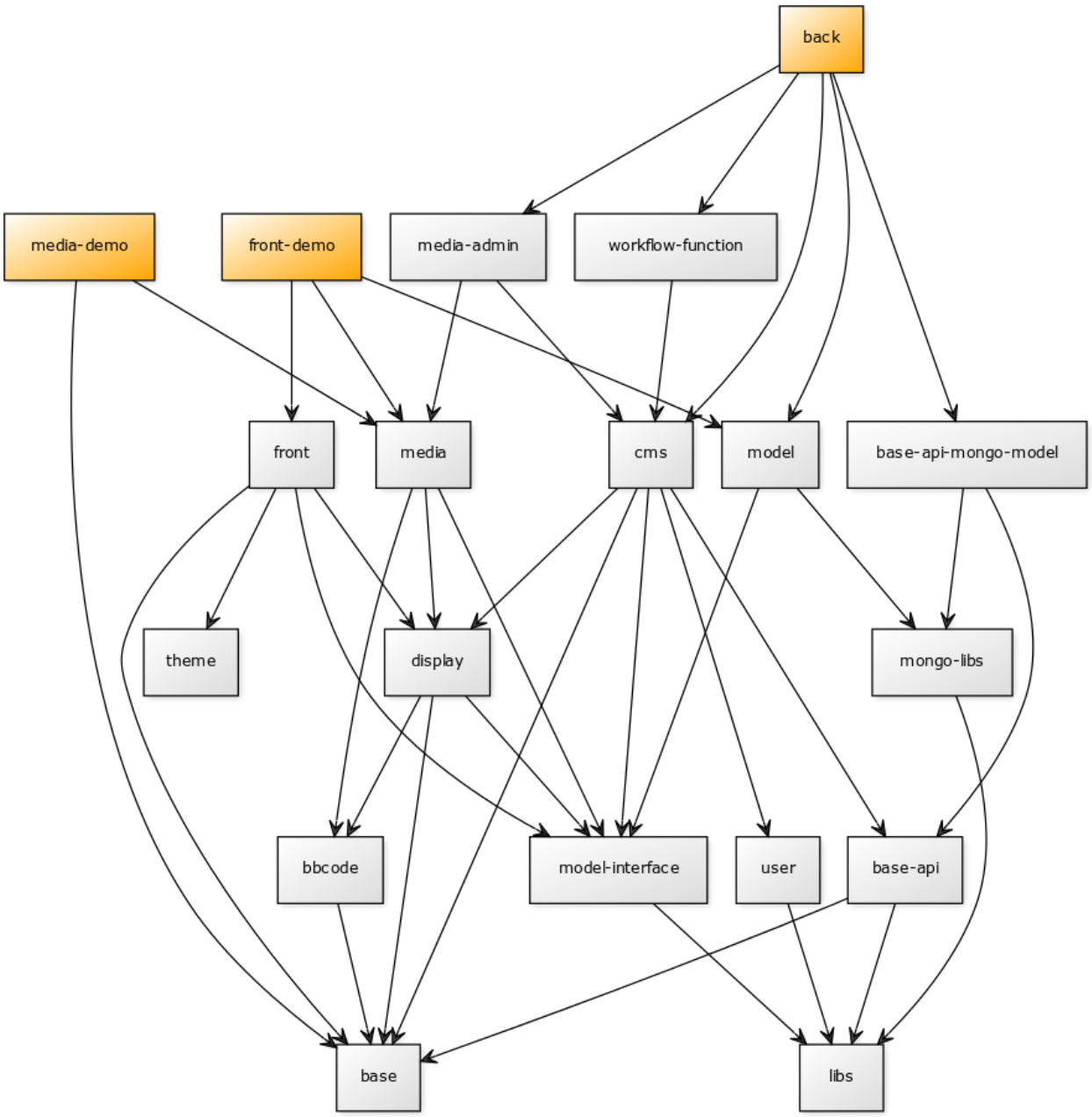
We will provide the list of all the bundles and the relations between them in a format usable on the yuml.me website.

```
[back{bg:orange}]->[cms]
[back]->[workflow-function]
[back]->[media-admin]
[back]->[model]
[back]->[base-api-mongo-model]
[cms]->[base]
[cms]->[base-api]
[cms]->[display]
[cms]->[user]
[cms]->[model-interface]
[base-api]->[base]
[base-api]->[libs]
[base-api-mongo-model]->[base-api]
[base-api-mongo-model]->[mongo-libs]
[mongo-libs]->[libs]
[display]->[model-interface]
[display]->[base]
[media-admin]->[cms]
[media-admin]->[media]
[media]->[model-interface]
[media]->[display]
[model]->[model-interface]
[model-interface]->[libs]
[model]->[mongo-libs]
[user]->[libs]
[workflow-function]->[cms]
[display]->[bbcode]
[media]->[bbcode]
[bbcode]->[base]

[front-demo{bg:orange}]->[front]
[front-demo]->[media]
[front-demo]->[model]
[front]->[base]
[front]->[display]
[front]->[model-interface]
[front]->[theme]
```

```
[media-demo{bg:orange}]->[media]
[media-demo]->[base]
```

1.24.2 Diagram history

20 octobre 2015



1.25 Bundle configuration

1.25.1 Context

As an open source product, Open Orchestra could not entitle you to just one configuration. This file will describe each configuration parameter for all bundles

1.25.2 Bundles

BaseBundle

This bundle will configure the base option for the whole application :

- Language for the administration
- Key to encrypt the preview token
- Alias for the object manager

```
open_orchestra_base:
    object_manager:       doctrine.odm.mongodb.document_manager
    administration_languages:

        # Defaults:
        - en
        - fr
    encryption_key:       ThisKeyIsNotSecret
```

BaseApiMongoModelBundle

This bundle will gives you an implementation for the base api document and repository :

- The factory used to create the repositories
- The `ApiClient` document and repository
- The `AccessToken` document and repository

```
open_orchestra_base_api_model:
    document:
        api_client:
            class:                OpenOrchestra\BaseApiMongoModelBundle\Document\ApiClient
            repository:           OpenOrchestra\BaseApiMongoModelBundle\Repository\ApiClientRepositor
        access_token:
            class:                OpenOrchestra\BaseApiMongoModelBundle\Document\AccessToken
            repository:           OpenOrchestra\BaseApiMongoModelBundle\Repository\AccessTokenReposit
```

BaseApiBundle

This bundle will be used as a base for all the applications api, you will need to configure :

- The controller to handle all the exceptions
- The expiration time for the created tokens

```
open_orchestra_base_api:
    http_exception_controller:  'OpenOrchestra\BaseApiBundle\Controller\ExceptionController::showActi
    token_expiration_time:  +1month
```

ApiBundle

This bundle will be used in the applications api. You can configure :

- The `Facade` returned by the `API`

```
open_orchestra_api:
    facades:
        api_client:                OpenOrchestra\ApiBundle\Facade\ApiClientFacade
        api_client_collection:     OpenOrchestra\ApiBundle\Facade\ApiClientCollectionFacade
        area:                      OpenOrchestra\ApiBundle\Facade\AreaFacade
        block:                     OpenOrchestra\ApiBundle\Facade\BlockFacade
        block_collection:          OpenOrchestra\ApiBundle\Facade\BlockCollectionFacade
        content:                   OpenOrchestra\ApiBundle\Facade\ContentFacade
        content_attribute:         OpenOrchestra\ApiBundle\Facade\ContentAttributeFacade
        content_collection:        OpenOrchestra\ApiBundle\Facade\ContentCollectionFacade
        content_type:              OpenOrchestra\ApiBundle\Facade\ContentTypeFacade
        content_type_collection:   OpenOrchestra\ApiBundle\Facade\ContentTypeCollectionFacade
        datatable_translation:     OpenOrchestra\ApiBundle\Facade\DatatableTranslationFacade
        field_type:                OpenOrchestra\ApiBundle\Facade\FieldTypeFacade
        group:                     OpenOrchestra\ApiBundle\Facade\GroupFacade
        group_collection:          OpenOrchestra\ApiBundle\Facade\GroupCollectionFacade
        keyword:                   OpenOrchestra\ApiBundle\Facade\KeywordFacade
        keyword_collection:        OpenOrchestra\ApiBundle\Facade\KeywordCollectionFacade
        link:                      OpenOrchestra\ApiBundle\Facade\LinkFacade
        node:                      OpenOrchestra\ApiBundle\Facade\NodeFacade
        node_collection:           OpenOrchestra\ApiBundle\Facade\NodeCollectionFacade
        node_group_role:           OpenOrchestra\ApiBundle\Facade\NodeGroupRoleFacade
        node_tree:                 OpenOrchestra\ApiBundle\Facade\NodeTreeFacade
        redirection:               OpenOrchestra\ApiBundle\Facade\RedirectionFacade
        redirection_collection:    OpenOrchestra\ApiBundle\Facade\RedirectionCollectionFacade
        role:                      OpenOrchestra\ApiBundle\Facade\RoleFacade
        role_collection:           OpenOrchestra\ApiBundle\Facade\RoleCollectionFacade
        role_string:               OpenOrchestra\ApiBundle\Facade\RoleFacade
        role_string_collection:    OpenOrchestra\ApiBundle\Facade\RoleCollectionFacade
        site:                      OpenOrchestra\ApiBundle\Facade\SiteFacade
        site_collection:           OpenOrchestra\ApiBundle\Facade\SiteCollectionFacade
        status:                    OpenOrchestra\ApiBundle\Facade\StatusFacade
        status_collection:         OpenOrchestra\ApiBundle\Facade\StatusCollectionFacade
        template:                  OpenOrchestra\ApiBundle\Facade\TemplateFacade
        theme:                     OpenOrchestra\ApiBundle\Facade\ThemeFacade
        theme_collection:          OpenOrchestra\ApiBundle\Facade\ThemeCollectionFacade
        trash_item:                OpenOrchestra\ApiBundle\Facade\TrashItemFacade
        trash_item_collection:     OpenOrchestra\ApiBundle\Facade\TrashItemCollectionFacade
        translation:               OpenOrchestra\ApiBundle\Facade\TranslationFacade
        ui_model:                  OpenOrchestra\ApiBundle\Facade\UiModelFacade
        widget:                    OpenOrchestra\ApiBundle\Facade\WidgetFacade
        widget_collection:         OpenOrchestra\ApiBundle\Facade\WidgetCollectionFacade
```

UserBundle

This bundle will manage the user's authentification. To display the forms in the best possible way, you need to configure :

- The base layout
- The form template file

```
open_orchestra_user:
    base_layout:          'OpenOrchestraUserBundle::baseLayout.html.twig'
    form_template:        'OpenOrchestraUserBundle::form.html.twig'
```

UserAdminBundle

This bundle will create the user back-office part of the Open Orchestra project. You can configure :

- The `Facade` returned by the `API`

```
open_orchestra_user_admin:
    facades:
        user:                 OpenOrchestra\UserAdminBundle\Facade\UserFacade
        user_collection:      OpenOrchestra\UserAdminBundle\Facade\UserCollectionFacade
```

ModelBundle

This bundle provides an implementation of all the interfaces from the `model-interface` component for mongoDB. You can configure :

- Some immutable properties for the contents
- Interfaces for the different fixtures groups
- All documents and repositories class

```
open_orchestra_model:

    # Immutable properties of the content class
    content_immutable_properties:  []
    fixtures_interface:
        all:

            # Default:
            - Doctrine\Common\DataFixtures\FixtureInterface
        production:

            # Default:
            - OpenOrchestra\ModelInterface\DataFixtures\OrchestraProductionFixturesInterface
        functional:

            # Default:
            - OpenOrchestra\ModelInterface\DataFixtures\OrchestraFunctionalFixturesInterface
    document:
        content:
            class:                OpenOrchestra\ModelBundle\Document\Content
            repository:           OpenOrchestra\ModelBundle\Repository\ContentRepository
        content_attribute:
            class:                OpenOrchestra\ModelBundle\Document\ContentAttribute
        content_type:
            class:                OpenOrchestra\ModelBundle\Document\ContentType
            repository:           OpenOrchestra\ModelBundle\Repository\ContentTypeRepository
        node:
            class:                OpenOrchestra\ModelBundle\Document\Node
            repository:           OpenOrchestra\ModelBundle\Repository\NodeRepository
        area:
            class:                OpenOrchestra\ModelBundle\Document\Area
        block:
            class:                OpenOrchestra\ModelBundle\Document\Block
        site:
            class:                OpenOrchestra\ModelBundle\Document\Site
            repository:           OpenOrchestra\ModelBundle\Repository\SiteRepository
        route_document:
```

```
            class:                OpenOrchestra\ModelBundle\Document\RouteDocument
            repository:           OpenOrchestra\ModelBundle\Repository\RouteDocumentRepository
        site_alias:
            class:                OpenOrchestra\ModelBundle\Document\SiteAlias
        template:
            class:                OpenOrchestra\ModelBundle\Document\Template
            repository:           OpenOrchestra\ModelBundle\Repository\TemplateRepository
        field_option:
            class:                OpenOrchestra\ModelBundle\Document\FieldOption
        field_type:
            class:                OpenOrchestra\ModelBundle\Document\FieldType
        status:
            class:                OpenOrchestra\ModelBundle\Document\Status
            repository:           OpenOrchestra\ModelBundle\Repository\StatusRepository
        embed_status:
            class:                OpenOrchestra\ModelBundle\Document\EmbedStatus
        theme:
            class:                OpenOrchestra\ModelBundle\Document\Theme
            repository:           OpenOrchestra\ModelBundle\Repository\ThemeRepository
        role:
            class:                OpenOrchestra\ModelBundle\Document\Role
            repository:           OpenOrchestra\ModelBundle\Repository\RoleRepository
        redirection:
            class:                OpenOrchestra\ModelBundle\Document\Redirection
            repository:           OpenOrchestra\ModelBundle\Repository\RedirectionRepository
        keyword:
            class:                OpenOrchestra\ModelBundle\Document\Keyword
            repository:           OpenOrchestra\ModelBundle\Repository\KeywordRepository
        embed_keyword:
            class:                OpenOrchestra\ModelBundle\Document\EmbedKeyword
        translated_value:
            class:                OpenOrchestra\ModelBundle\Document\TranslatedValue
        trash_item:
            class:                OpenOrchestra\ModelBundle\Document\TrashItem
            repository:           OpenOrchestra\ModelBundle\Repository\TrashItemRepository
```

MediaBundle

This bundle gives you a way to display medias in blocks, contents, ... You can configure :

- The media domain

```
open_orchestra_media:
    media_domain:         ''
```

MediaModelBundle

This bundle provides an implementation for all the interfaces defined in the MediaBundle. You can configure :

- The `Media` and `MediaFolder` document and repository

```
open_orchestra_media_model:
    document:
        media:
            class:                OpenOrchestra\MediaModelBundle\Document\Media
            repository:           OpenOrchestra\MediaModelBundle\Repository\MediaRepository
        media_folder:
```

```
            class:                OpenOrchestra\MediaModelBundle\Document\MediaFolder
            repository:           OpenOrchestra\MediaModelBundle\Repository\FolderRepository
```

MediaAdminBundle

This bundle gives you a way to display medias in blocks, contents, ... You can configure :

- The upload temporary directory
- Width, height and compression quality for each available image formats
- The default thumbnail for each media type
- The `Facade` returned by the `API`

```
open_orchestra_media_admin:
    tmp_dir:              /tmp
    thumbnail:
        max_width:            117
        max_height:           117
        compression_quality:  75
    alternatives:
        image:
            formats:
                fixed_height:
                    max_height:           100
                    compression_quality:  75
                fixed_width:
                    max_width:            100
                    compression_quality:  75
                rectangle:
                    max_width:            100
                    max_height:           70
                    compression_quality:  75
        default:
            thumbnail:            orchestra-media-thumbnail-default.png
        audio:
            thumbnail:            orchestra-media-thumbnail-audio.png
    facades:
        media:                OpenOrchestra\MediaAdminBundle\Facade\MediaFacade
        media_collection:     OpenOrchestra\MediaAdminBundle\Facade\MediaCollectionFacade
```

WorkflowFunctionModelBundle

This bundle provides an implementation for all the interfaces defined in the WorkflowBundle. You can configure :

- All documents and repositories class

```
open_orchestra_workflow_function_model:
    document:
        workflow_function:
            class:                OpenOrchestra\WorkflowFunctionModelBundle\Document\WorkflowFunctio
            repository:           OpenOrchestra\WorkflowFunctionModelBundle\Repository\WorkflowFunct
        workflow_right:
            class:                OpenOrchestra\WorkflowFunctionModelBundle\Document\WorkflowRight
            repository:           OpenOrchestra\WorkflowFunctionModelBundle\Repository\WorkflowRightR
        authorization:
            class:                OpenOrchestra\WorkflowFunctionModelBundle\Document\Authorization
```

```
        reference:
            class:                OpenOrchestra\WorkflowFunctionModelBundle\Document\Reference
```

WorkflowFunctionAdminBundle

This bundle will create the workflow functions back-office part of the Open Orchestra project. You can configure :

- The `Facade` returned by the `API`

```
open_orchestra_workflow_function_admin:
    facades:
        workflow_function:             OpenOrchestra\WorkflowFunctionAdminBundle\Facade\WorkflowFunct
        workflow_function_collection:  OpenOrchestra\WorkflowFunctionAdminBundle\Facade\WorkflowFunct
```

BackofficeBundle

This bundle will create the Back Office of the Open Orchestra project. You can configure :

- The language from the front installation
- The blocks that you created
- The fixed attributes from the block (shared through all blocks)
- The front roles, that can be added to the front pages
- The field type and options for the content (specific to your project)
- The color available for the Back Office

```
open_orchestra_backoffice:

    # Add the available languages in the Front Office, default (en, fr, de)
    front_languages:

        # Prototype
        key:                  ~

    # Set the available block types for this application
    blocks:

        # Defaults:
        - footer
        - language_list
        - menu
        - sub_menu
        - content_list
        - content
        - configurable_content
        - tiny_mce_wysiwyg
        - video
        - gmap
        - add_this
        - audience_analysis
        - contact

    # Add the global block attributes
    fixed_attributes:     []
```

```
    # Role than can be given to the user on the Front websites
    front_roles:          []

    # Array of content attributes (for content types)
    field_types:

        # Prototype
        field_name:
            label:                ~ # Required
            type:                 ~ # Required
            default_value:
                type:                 ~
                options:
                    label:                ~
                    required:             true
            options:

                # Prototype
                option_name:
                    default_value:        ~ # Required

    # Array of content attributes options
    options:

        # Prototype
        option_name:
            type:                 ~ # Required
            label:                ~ # Required
            required:             true

    # List of the color available, in the status for instance
    available_color:

        # Prototype
        key:                  ~

    # List of widgets presented on the dashboard
    dashboard_widgets:

        # Defaults:
        - last_nodes
        - draft_nodes
        - last_contents
        - draft_contents
```

LogBundle

This bundle will create the log back-office part of the Open Orchestra project. You can configure :

- The `Facade` returned by the `API`

```
open_orchestra_log:
    facades:
        log:                  OpenOrchestra\LogBundle\Facade\LogFacade
        log_collection:       OpenOrchestra\LogBundle\Facade\LogCollectionFacade
```

FrontBundle

This bundle creates the base part for the Front Office installation. You can configure :

- The devices name
- The device type field name
- The routing type

```
open_orchestra_front:
    devices:

        # Prototype
        name:
            parent:               null
    device_type_field:    x-ua-device
    routing_type:         ~ # One of "file"; "database"
```

ThemeBundle

This bundle will add the different assets (js and css files) to the different files. You can configure :

- The different stylesheet groups

```
open_orchestra_theme:
    themes:

        # Prototype
        id:
            name:                 ~ # Required
            stylesheets:          []
            javascripts:          []
```

1.26 Custom fixtures load

1.26.1 Context

Open Orchestra comes with a set of fixtures that will allow you to have a full overview of the project. Those fixtures contains some pages, content, media, and much more.

If you want to have just the base data which would allow you to login on the Back Office and start creating your first pages, a simple solution would be to load only the fixtures you need.

You could do it by adding some paths as command arguments :

```
$ php app/console doctrine:mongo:fixture:load --fixtures="src" --fixtures="vendor/open-orchestra/"
```

This solution should be avoided because it wouldn't be functional if the fixtures folder is moving.

1.26.2 Open Orchestra Solution

Common usage

Open Orchestra provides a console command which allows you to load only the fixtures you need.

```
$ php app/console orchestra:mongo:fixtures:load
```

This command will look through all the project fixtures and load only those implementing a specific interface :

```
use OpenOrchestra\ModelInterface\DataFixtures\OrchestraProductionFixturesInterface;

/**
 * Class LoadStatusData
 */
class LoadStatusData extends AbstractFixture implements OrderedFixtureInterface, OrchestraProductionF
```

Specific usage

The previous solution allows you to load only the fixtures needed to get logged on the Back Office.

If you want to load only some fixtures specific to your project, the `orchestra:mongo:fixtures:load` can also be used.

First your fixtures file should implement a specific interface (defined in your project):

```
use FooBundle\Loader\BarFixtureInterface;

class LoadFooData implements BarFixtureInterface {}
```

Then you should configure the `ModelBundle` to load only the fixtures implementing the `BarFixtureInterface`.

```
open_orchestra_model:
    production_fixtures_interface:
        - FooBundle\Loader\BarFixtureInterface
```

If you want to load the base production fixtures and your personal fixtures, you can add both interfaces in the configuration:

```
open_orchestra_model:
    production_fixtures_interface:
        - OpenOrchestra\ModelInterface\DataFixtures\OrchestraProductionFixturesInterface
        - FooBundle\Loader\BarFixtureInterface
```

1.27 Block creation

1.27.1 The block system

In Open Orchestra, pages are made of blocks representing data to display.

In order for them to be easily manageable, each block comes with a set of display strategies defined as services:

- A display strategy for the Back Office. This strategy is used to render the block when editing a node including that type of block
- An icon and title display strategy, visible in the Back Office, in the blocks panel of node pages
- A strategy for displaying the form to edit the block in the Back Office
- A display strategy for the Front Office

These strategies are identified by the usage of specific tags when declaring the service.

1.27.2 The different strategies

Back Office display strategies

Services used as Back Office block display strategies, must be taggued as `open_orchestra_backoffice.display_block.strategy`:

```
tags:
    - { name: open_orchestra_backoffice.display_block.strategy }
```

Those services also need to implement `OpenOrchestra\DisplayBundle\DisplayBlock\DisplayBlockInterface`

Strategies for the icon

Services used as block icon display strategies must be taggued as `open_orchestra_backoffice.display_icon.strategy`:

```
tags:
    - { name: open_orchestra_backoffice.display_icon.strategy }
```

Such services also need to implement `OpenOrchestra\BackofficeBundle\DisplayIcon\DisplayInterface`

Strategies for the form

Services used as block form display strategies must be tagged as `open_orchestra_backoffice.generate_form.strategy`

```
tags:
    - { name: open_orchestra_backoffice.generate_form.strategy }
```

They also need to implement `OpenOrchestra\Backoffice\GenerateForm\GenerateFormInterface`

Front Office display strategies

Services used as Front Office block display strategies, must be taggued as `open_orchestra_display.display_block.strategy`:

```
tags:
    - { name: open_orchestra_display.display_block.strategy }
```

Those services also need to implement `OpenOrchestra\DisplayBundle\DisplayBlock\DisplayBlockInterface`

1.27.3 Generating blocks in command line

In order to easily create blocks, Open Orchestra offers a Symfony console command named `orchestra:generate:block`.

The usage is pretty straightforward :

```
$ php app/console orchestra:generate:block
    --block-name="TestBlock"
    --form-generator-dir="src/YourVendor/AcmeBundle/Path/To/Form/Strategies"
    --front-display-dir="src/YourVendor/AcmeBundle/Path/To/Front/Display/Strategies"
    --backoffice-icon-dir="src/YourVendor/AcmeBundle/Path/To/Icon/Strategies"
    --backoffice-display-dir="src/YourVendor/AcmeBundle/Path/To/Back/Display/Strategies"
    --no-interaction
```

1.27.4 Add the block in the configuration

After creating the different strategies, you still have to register your new block in Open Orchestra. To do this, add it to your application configuration:

```
open_orchestra_backoffice:
    blocks:
        - block_name
```

1.27.5 Cache control

Open Orchestra leverages the power of ESI blocks to optimize page rendering and increase performance. In the Front Office application, each block is rendered in an ESI block using the render_esi() twig function.

See also ESI blocks.

1.28 Block parameter

1.28.1 Context

For their display to be customizable, some blocks need to receive parameters when being rendered. Those parameters could be of multiple types, for instance :

- url parameters
- server parameters
- header parameters
- ...

A simple solution to give those parameters to the block would have been to give all the request parameters when the block is called.

This solution is not optimal for a main reason: when loading blocks via ESI, each block will be cached for each url variant.

In this example, the block 1, which does not require any parameters, will be cached for each url variant :

- `/block/1` will be in one cache entry.
- `/block/1?foo=bar` will be in another cache entry.

To improve your cache hit ratio, you should have the least entries for a given page.

1.28.2 Open Orchestra solution

Open Orchestra lets you specify the parameters to give to a block in order to limit the cache entries for a single block.

This solution requires to work in two specifics areas : - in the Back Office, where you add the block - in the Front Office, where you will require some parameters linked to the block.

Back Office work

When a block is added in an area of a node, a block reference is created to link the block and the area. That block reference contains the parameter types required to display the block in Front Office.

You need to create a service implementing the interface `OpenOrchestra\Backoffice\BlockParameter\BlockParameter`

In this interface, the method `getBlockParameter` returns an array of all the parameters needed by the block to work properly.

This service should be declared with the tag `open_orchestra_backoffice.block_parameter.strategy`

```
tags:
    - { name: open_orchestra_backoffice.block_parameter.strategy }
```

Front Office work

In the Front Office, the sub-request to display a block (in the main request) is created while you don't have any idea of the block type. The `blockParameter` will help you to complete the sub-request parameters.

To analyze the parameters and complete the request, you need to create a service implementing the interface `OpenOrchestra\FrontBundle\SubQuery\SubQueryGeneratorInterface`

This service should be declared with the tag `open_orchestra_front.sub_query.strategy`

```
tags:
    - { name: open_orchestra_front.sub_query.strategy }
```

1.28.3 Example

Subquery strategies available

Open Orchestra comes with a few strategies already defined. *All those strategies can be cumulated for a single block.*

Name	Action
`DeviceSubQueryStrategy`	Pass the header parameter `x-ua-device`
`PostDataSubQueryStrategy`	Pass all data from the `POST` part
`RequestSubQueryStrategy`	Pass the specified request parameter (aliasId for instance
`CurrentRouteSubQueryStrategy`	Pass the current route to the block

New Subquery Strategy

Let's say that you want to translate the foo parameter from the request and give it to the block.

You need to implement the three methods of the interface :

- `getName` : give the name of the strategy
- `support` : take the block parameter to decide if the strategy should be used
- `generate` : will generate the value

In our case, we will check if the block requires a parameter named `foo`.

```
public function support($blockParameter)
{
    return strpos($blockParameter, 'foo') === 0;
}
```

The generate method will need the `translator` service in order to find the translation for the parameter.

```
public function generate($blockParameter)
{
    $fooParameter = $this->request->get('foo');

    return array('foo' => $this->translator->trans($fooParameter);
}
```

In the sub-request, the `foo` parameter will be added to the request parameters.

1.29 Block display

1.29.1 Display Independence

In the Front Office, some blocks could require some specific CSS and JavaScript files. To allow any block to add some specific files, the main node layout loads the requireJS library which is a JavaScript file and module loader. Each block is allowed to call the `require()` function of the `requireJS` library. To allow CSS dynamic load, Open Orchestra offers an AMD module named `openOrchestraCss` which is meant to be used with `requireJS`. In a case in which the same block is displayed several times, corresponding CSS and JavaScript files will only be loaded once.

1.29.2 Block loader file

For each block there is a `blockLoader.js` file using the `requireJS` library. In a `blockLoader.js` file are defined the module used to load CSS: `openOrchestraCss`, specific JavaScript files in block and JavaScript files in common libraries which are needed to initialize a block.

After loading the elements, it does the initialization. During this step, CSS are loaded using the `openOrchestraCss` module with the `load()` function and optional JavaScript is run.

A `blockLoader.js` file is structured as the following:

```
requirejs(
    ['jquery'],
    function() {
        require(
            [
                //Define required elements
                'openOrchestraCss', //Module used to load CSS
                '/bundles/nameofbundle/block/NameOfMyBlock/js/foo.js', //Specific JavaScript in bloc
                '/bundles/nameofbundle/libs/bar/bar.js'//JavaScript in common libraries
            ],
            function (openOrchestraCss) {
                //Load block CSS
                openOrchestraCss.load(['/bundles/nameofbundle/block/NameOfMyBlock/css/style.css']);

                //You can add optional JavaScript code here
                jQuery(document).ready(function ($) {


                });
            }
        )
    }
);
```

1.29.3 Example: Gallery block

In the MediaBundle, the `Resources/views/Block` directory contains the Front Media blocks display templates. The file `Gallery/show.html.twig` ends with the `blockLoader.js` load which does CSS and JavaScript dynamic load in relation with Gallery block.

```
<script>
    require(['{{ asset('/bundles/openorchestramedia/block/Gallery/blockLoader.js') }}']);
</script>
```

1.29.4 Blocs maintainability

To keep consistency between all blocks, you should use a file tree like the following:

```
|_ MyBundle
    |_ Resources
        |_ public
        |   |_ block
        |   |   |_ NameOfMyBlock
        |   |       |_ css
        |   |       |   |_ * Place here css files of your block *
        |   |       |
        |   |       |_ js
        |   |       |   |_ * Place here JavaScript files of your block *
        |   |       |
        |   |       |_ img
        |   |       |   |_ * Place here images files of your block *
        |   |       |
        |   |       |_ blockLoader.js * Block loading file which loads specific css and JavaScript *
        |   |
        |   |_ libs
        |   |   |_ * Place here commons libraries to many blocks *
        |   |
        |   |_ img
        |       |_ * Place here commons images to many blocks *
        |
        |_ views
            |_ Block
                |_ NameOfMyBlock
                    |_ * Place here block display templates *
```

1.30 Block Override

1.30.1 Context

From time to time, the block provided by Open Orchestra will fulfill some of your requirements but you would want to modify a little part of their behaviour.

As the block is partially functional, you do not want to create an other block with the functionality.

We provide you different ways to override part of the Open Orchestra block which depends on your use case :

- If you want to modify only the block display, *override the template of a block*
- If you want to modify the logic of the block, *override the service*

- If you want to modify the whole service, *override the full block*

1.30.2 Override the template of a block

Overloading the template of a block is achieved by creating a template in the following folder structure :

```
|_ app
    |_ Resources
        |_ {BUNDLE_NAME}
            |_ views
                |_ {PATH_TEMPLATE}
                    |_ {FILENAME}
```

Example :

You need to override the template of the Content List Block block from the Back Office to display additional datas.

```
|_ Open-orchestra
    |_ open-orchestra-cms-bundle
        |_ BackofficeBundle
            |_ Ressources
                |_ views
                    |_ Block
                        |_ ContentList
                            |_ show.html.twig
```

`Open-orchestra/open-orchestra-cms-bundle/BackofficeBundle/Ressources/views/Block/ContentList/show.html.twig`

You simply create your template in next folder:

```
|_ app
    |_ Resources
        |_ OpenOrchestraBackofficeBundle
            |_ views
                |_ Block
                    |_ ContentList
                        |_ show.html.twig
```

`app/Ressources/OpenOrchestraBackofficeBundle/views/Block/ContentList/show.html.twig`

See again symfony2 documentation

1.30.3 Override directly a block

As an introduction to this part, being familiar with the block parameter documentation page is a requirement.

Open Orchestra gives you two ways to override a block.

Override the service

You can directly replace a block by overloading its service. All open-orchestra blocks have their referenced class in the parameters from config file.

```
# open-orchestra-cms-bundle/BackofficeBundle/Ressources/config/display.yml
parameters:
    open_orchestra_backoffice.display.content_list.class: OpenOrchestra\BackofficeBundle\DisplayBlock
    open_orchestra_backoffice.display.content.class: OpenOrchestra\BackofficeBundle\DisplayBlock\Stra
```

It's thus possible for you to override these parameters and thus overload the class from service.

```
parameters:
    open_orchestra_backoffice.display.content_list.class: Foo\ExampleBundle\Blocks\Strategies\Content
```

This method won't give you the opportunity to change the class constructor and all the dependancy injected to the block.

Override the full block

To override an existing block by its name, you have to add your block in the services form config file. Don't forget the tag in your declaration.

```
services:
    itkg.display.content_list:
        class: "%itkg.display.content_list.class%"
        tags:
            - { name: open_orchestra_backoffice.display_block.strategy }
```

Finally, the getName method must return the same name as the overloaded block.

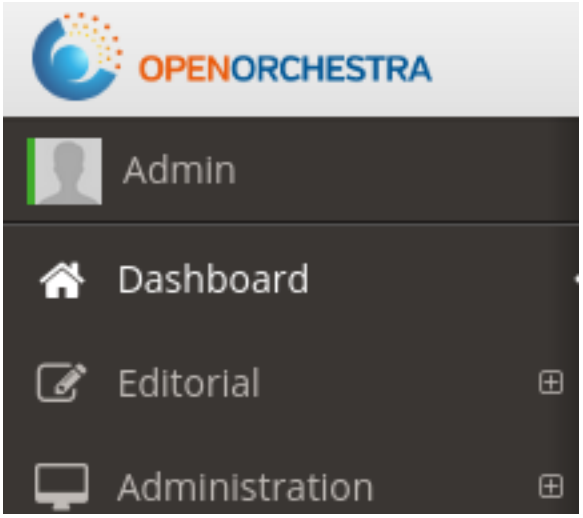
```
public function getName()
{
    return "block_name";
}
```

This method will give you the opportunity to totaly redifine the block, from the dependency needed to the working logic and the template used.

1.31 Navigation panel

1.31.1 Context

The navigation panel containing the navigation menu is fully configurable by adding specific services to the application. Open Orchestra natively provides different kind of strategies for this panel that are reusable for you own entries.



1.31.2 Usage

Create the strategy

Each panel entry is a service implementing the interface `OpenOrchestra\Backoffice\NavigationPanel\NavigationPar`
Different strategies are already available in the folder `OpenOrchestra\Backoffice\NavigationPanel\Strategies`
and can be reused for most common needs in panel services.

For the service to be recognised as a menu entry, it should be tagged with `open_orchestra_backoffice.navigation_panel.strategy`:

```
tags:
    - { name: open_orchestra_backoffice.navigation_panel.strategy }
```

Display of the menu

The display of the menu item is managed by the method `NavigationPanelInterface::show()` which should return an HTML string.

The menu link must have the following attributes :

- href : an anchor defining the route for Backbone.js, it will be displayed in the browser's address bar
- id : merge of the "nav-" string and the name of the strategy as returned by `NavigationPanelInterface::getName()`, for instance "nav-user"

Menu and data table

Many panel entries allow to consult records formated in a data table view based on DataTables library.. This library allows multiple configurations to custom these tables: name of the column, visible or not by default, search fields... OpenOrchestra uses its own process to manage these configurations.

First of all, a yaml file describes the settings of each column for a given data table.

Settings available:

- **name**: *(string)* field name
- **title**: *(string)* key of title translation

- **visible**: *(boolean)* Enable or disable the display of column
- **activateColvis**: *(boolean)* Enable or disable the toggle of column in colvis button
- **searchable**: *(boolean)* Enable or disable the search of column
- **searchField**: *(string)* type of search field, further information in Entity list documentation
- **orderable**: *(boolean)* Enable or disable the order of column
- **orderDirection**: *(string)* asc|desc Indicate the order direction if orderable option is enable

Here is an example of yml data table configuration (datatable_parameter.yml):

```
open_orchestra_backoffice.navigation_panel.trashcan.parameters :
    -
      name : name
      title : open_orchestra_backoffice.table.trashcan.name
      visible : true
      activateColvis : true
      searchField : text
    -
      name : deleted_at
      updated_by : name
      title : open_orchestra_backoffice.table.trashcan.deleted_at
      visible : true
      activateColvis : true
      searchField : date
```

Then, this configuration parameter, `open_orchestra_backoffice.navigation_panel.trashcan.parameters`, is injected to the panel strategy. Here is the service configuration:

```
open_orchestra_backoffice.navigation_panel.trashcan:
    class: "%open_orchestra_backoffice.navigation_panel.administration.class%"
    arguments:
        - trashcan
        - ROLE_ACCESS_DELETED
        - 250
        - "%open_orchestra_backoffice.navigation_panel.menu.editorial%"
        - "%open_orchestra_backoffice.navigation_panel.trashcan.parameters%"
        - "@translator"
    calls:
        - [ setTemplate, [OpenOrchestraBackofficeBundle:BackOffice:Include/NavigationPanel/Menu/Edito
    tags:
        - { name: open_orchestra_backoffice.navigation_panel.strategy }
```

As a remark, the Symfony translator is also injected. It will try to translate each value from the column settings. In this case `open_orchestra_backoffice.table.trashcan.name` will be translated.

Finally a client side call to the method `datatableParameterAction` of the controller `OpenOrchestra\ApiBundle\Controller\DataTableController` allows to retrieve all these settings. which are stored in a global js object `dataTableConfigurator`.

Each panel strategy called by the controller returns its settings in an associative array. The keys of each panel strategy array have two purposes :

- allow merging in controller without conflict,
- serve as a search key on client side.

Indeed, by setting `data-datatable-parameter-name`, each entry in panel indicates where to find its configuration in `dataTableConfigurator`. In the common case, the name of the strategy is used.

In the previous example, the `open_orchestra_backoffice.navigation_panel.trashcan` panel strategy serves the controller the settings by associating them to the name of the strategy: `trashcan`. On client side, settings are founded in `dataTableConfigurator` at the key `trashcan` specified by `data-datatable-parameter-name`.

Define Backbone route

When clicking on the menu element, the Backbone route matching with href attribute value will be executed. To learn more about the way to define a new Backbone route, see Backbone routing in Open Orchestra.

1.31.3 Specifics

Order the menus

It is possible to modify the order of the items in the panel by changing the return value of the `NavigationPanelInterface::getWeight()` method. The heaviest elements are displayed below the other ones.

The `NavigationPanelInterface::getParent()` method allows an item to have a parent item in order to define a hierarchy of elements. The root of the panel is an `administration` node so all top level items should have it as a parent.

Access restriction

Restricting access to a menu element is done by defining a specific role the user should possess inside `NavigationPanelInterface::getRole()`.

To be allowed to configure the role in the Back Office (by adding it to a group), you will need to add the role to the `RoleCollector` class.

You will need to create a compiler pass to add the role to the `RoleCollector` definition in the `kernel`. The `BackofficeBundle` already provides the `AbstractRoleCompilerPass` which will simplify the process.

Let's say that in the `FooBundle` you would like to add the `ROLE_BAR`.

First create the `RoleCompilerPass` in the `DependencyInjection\Compiler` folder of your bundle :

```
<?php

namespace OpenOrchestra\FooBundle\DependencyInjection\Compiler;

use OpenOrchestra\BackofficeBundle\DependencyInjection\Compiler\AbstractRoleCompilerPass;
use Symfony\Component\DependencyInjection\ContainerBuilder;

/**
 * Class RoleCompilerPass
 */
class RoleCompilerPass extends AbstractRoleCompilerPass
{
    /**
     * You can modify the container here before it is dumped to PHP code.
     *
     * @param ContainerBuilder $container
     *
     * @api
     */
```

```
    public function process(ContainerBuilder $container)
    {
        $this->addRoles($container, array(
            'ROLE_BAR',
        ));
    }
}
```

Then declare the compiler pass in the bundle creation file `OpenOrchestraFooBundle` :

```
<?php

namespace OpenOrchestra\FooBundle;

use OpenOrchestra\FooBundle\DependencyInjection\Compiler\RoleCompilerPass;
use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\HttpKernel\Bundle\Bundle;

/**
 * Class OpenOrchestraFooBundle
 */
class OpenOrchestraFooBundle extends Bundle
{
    /**
     * @param ContainerBuilder $container
     */
    public function build(ContainerBuilder $container)
    {
        parent::build($container);
        $container->addCompilerPass(new RoleCompilerPass());
    }
}
```

After clearing the cache, you will see the role `ROLE_BAR` displayed in the role list in the `Group` modification form.

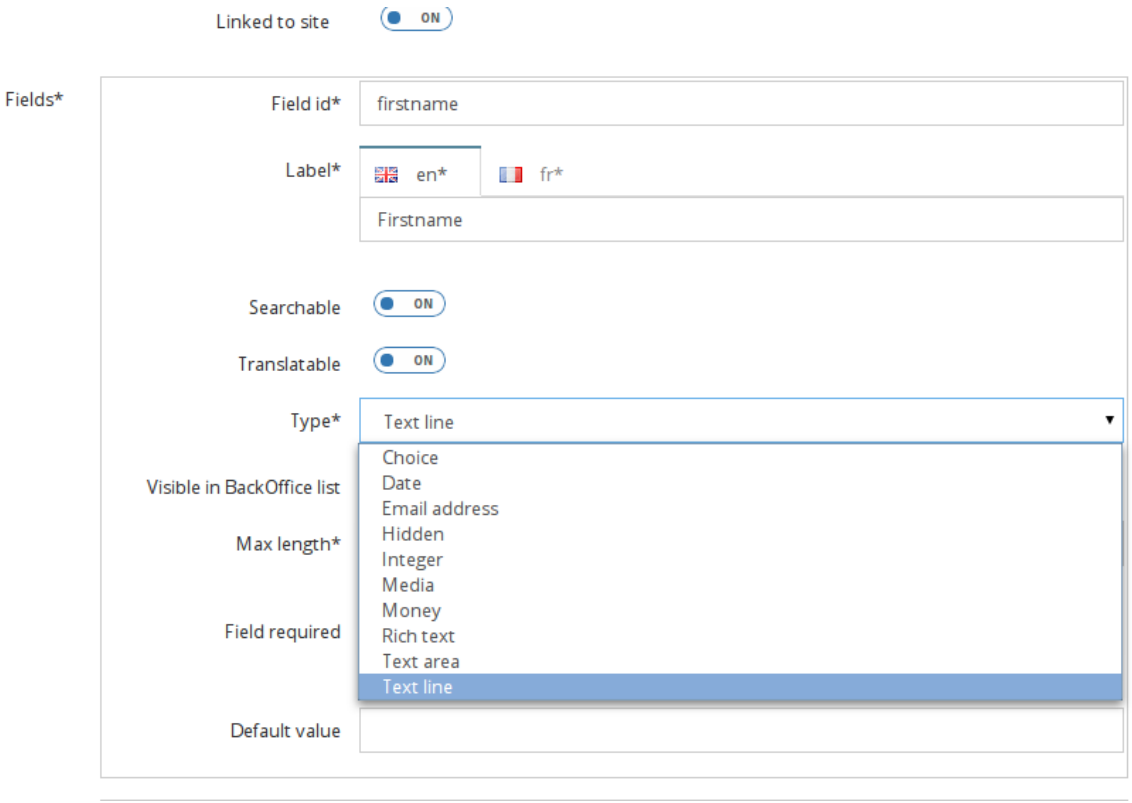
Finally, you can add some translation on the role. To separate the role translations from the rest of the application, we use the `role` domain. This way, you will have to add the translation in the `role.en.yml` file (for the `en` locale).

1.32 Field type

Open Orchestra offers the possibility to create the Content Types from the back office. Further information about it is available in the Content Types documentation.

When creating a content type, you can add several attributes of different types. By default Open Orchestra offers few types (text line, date, email, hidden, integer, choice, media, money, rich test, text area).

These page explains how to create new field types to extend the possibilities.



1.32.1 Add a field type

To add a new field type, you need to add an entry to the configuration `open_orchestra_backoffice.field_types` which describes your field type.

The description of your field type must contain these values:

- `name` name of your field type
- `type` form type of your field, this can be a form type natively available in Symfony or a custom form type
- `label` translation key of label
- `default_value` (optional) describes the field allowing to contribute the default value of your field.
- `options` (optional) describes the different options to configure your field type (max length, required, etc).

Example of simple field type:

```
open_orchestra_backoffice.field_types:
    custom_textarea:
        label: open_orchestra.form.field_type.textarea
        type: textarea
```

1.32.2 Field type options

The field type can be configured with options, for instance you can add an option to field `custom_textarea` to configure the max length.

You must add an entry in configuration `open_orchestra_backoffice.options` with 4 parameters:

- `name` name of option
- `type` form type of option, this can be a form type natively available in Symfony or your custom form type
- `label` translation key of label
- `required` a boolean indicating if option is required

When the user adds a field type at a content type, if an option of this field type is required then the user can't save the content type until he has contributed the option.

Example of option `max length`:

```
open_orchestra_backoffice.options:
    max_length:
        type: integer
        label: open_orchestra_backoffice.form.orchestra_fields.max_length
        required: true
```

Now, you can add the option in your field type:

```
custom_textarea:
    label: open_orchestra.form.field_type.textarea
    type: textarea
    options:
        max_length:
            default_value: 100
```

The parameter `default_value` is required in all options .

Open Orchestra already offers many options (max_length, required, grouping, rounding_mode, multiple, expanded choices, currency, precision, format, widget, input) that you can use in your field type.

1.32.3 Field type default value

When adding an attribute to a content type, the user may contribute the default value for this attribute. As the default value type depends on the field type (datetime, text area, media, etc), it is interesting to personalize the default value form.

For instance it is interesting to have a text area to contribute the default value of your field `custom_textarea` created above.

```
textarea:
    label: open_orchestra_backoffice.form.field_type.custom_type.textarea
    type: textarea
    default_value:
        type: textarea
        options:
            label: open_orchestra_backoffice.form.field_type.default_value
            required: false
    options:
        max_length:
            default_value: 100
```

The `textarea.default_value.options` key are options proposed by Symfony for the textarea form type (max length, label, trim, disabled, etc).

1.33 Entity list with ajax pagination

1.33.1 Context

To list the entities in an ergonomic way, Open Orchestra uses the DataTables plugin. To improve performance, the pagination uses ajax calls.

For each display of the information on the page (i.e. paging, ordering, searching), the plugin makes an ajax request to the api with a number of parameters.

1.33.2 Usage

The different variables used to filter (pagination, search, ordering) send by the plugin are:

Parameter name	Description
order[name]	Column's name to which ordering should be applied
order[dir]	Ordering direction (asc or desc)
search[columns][name]	Search value to apply to a specific column
search[global]	Global search value
length	Number of records displayed in the current draw
start	Paging first record indicator

The `search` parameter is not present in the request if there is no value in all search input field.

Example of parameters sent:

```
order[name]:version
order[dir]:asc
start:0
length:50
search[columns][version]:2
search[global]:car
```

In order to make the pagination operational, the api should return a facade filled with 4 parameters:

- recordsTotal: Total number of entities
- recordsFiltered: Number of entities matching the filters (search)
- collectionName: The name of variable which contains the data (for instance if collectionName equals `contents` then the data will be in the variable `$contents`)
- data: The data to be displayed in the table. This is an array who contains your data (facade entity) after filtering (search, order, pagination)

For instance, here comes the json returned by the api for listing the workflow functions:

```
{
    "links":{},
    "collection_name":"workflow_functions",
    "workflow_functions": [
        {
        "id":"558162b902b0cfc4118b45a7",
        "links":{},
        "name":"Contributor"
        },
        {
        "id":"558162b902b0cfc4118b45a6",
        "links":{},
```

```
        "name":"Validator"
        }
    ],
    "recordsTotal":2,
    "recordsFiltered":2
}
```

1.33.3 Server side search

Open Orchestra provides a set of tools to help you perform the server side search.

The `OpenOrchestra\Pagination\MongoTrait\PaginationTrait` trait should be used in repositories supporting aggregation queries. It contains three methods:

- `findForPaginate(PaginateFinderConfiguration $configuration)` finds documents from various criteria defined in the configuration object
- `count()` counts all documents of the collection
- `countWithFilter(PaginateFinderConfiguration $configuration)` counts documents according to the configuration object

`PaginateFinderConfiguration` is a configuration object with different properties (`order`, `limit`, `skip`, `search` and `descriptionEntity`), which, except `descriptionEnity`, can be automatically filled from the request sent by the dataTables with the static method `PaginateFinderConfiguration::generateFromRequest(Request $request)`.

The `descriptionEntity` property is an array describing the mapping between the columns displayed in the dataTable and the entity properties.

For instance in the list of redirections there are five columns (`SiteName`, `Routepattern`, `Locale`, `Redirection` and `Permanent`). In the `Redirection` entity the properties matching these columns are:

```
class Redirection
{
    // ...

    /**
     * @ODM\Field(type="string")
     */
    protected $locale;

    /*
     * @ODM\Field(type="string")
     */
    protected $siteName;

    /**
     * @ODM\Field(type="string")
     */
    protected $routePattern;

   /**
     * @ODM\Field(type="string")
     */
    protected $url;

    /**
     * @ODM\Field(type="boolean")
```

```
     */
    protected $permanent;

    // ...
}
```

The `descriptionEntity` property linked to the `Redirection` will be:

```
$descriptionEntity = array(
    'site_name' => array('field' => 'siteName', 'type' => 'string', 'key' => 'site_name'),
    'route_pattern' => array('field' => 'routePattern', 'type' => 'string', 'key' => 'route_pattern'),
    'redirection' => array('field' => 'url', 'type' => 'string', 'key' => 'redirection'),
    'url' => array('field' => 'url', 'type' => 'string', 'key' => 'url'),
    'permanent' => array('field' => 'permanent', 'type' => 'boolean', 'key' => 'permanent'),
)
```

- `field` : name of the entity's property
- `type` : type of the entity's property
- `key` : name of the property in your facade used by the dataTable

This mapping can be specified in a number of different formats including YAML, XML or directly inside your entities class via annotations

For instance, the `Redirection` entity will look like:

Annotation:

```
// src/AppBundle/Document/Redirection.php
namespace AppBundle\Document;

use OpenOrchestra\Mapping\Annotations as ORCHESTRA;

class Redirection
{
    // ...

    /**
     * @ODM\Field(type="string")
     * @ORCHESTRA\Search(key="locale")
     */
    protected $locale;

    /*
     * @ODM\Field(type="string")
     * @ORCHESTRA\Search(key="site_name")
     */
    protected $siteName;

    /**
     * @ODM\Field(type="string")
     * @ORCHESTRA\Search(key="route_pattern")
     */
    protected $routePattern;

   /**
     * @ODM\Field(type="string")
     * @ORCHESTRA\Search(key={"redirection", "url"})
     */
    protected $url;
```

```
    /**
     * @ODM\Field(type="boolean")
     * @ORCHESTRA\Search(key="permanent", type="boolean")
     */
    protected $permanent;

    // ...
}
```

Yaml:

```
# src/AppBundle/Resources/config/search/Document.Redirection.yml

AppBundle\Document\Redirection:
    properties:
        locale:
            key: locale
        siteName:
            key: site_name
        routePattern:
            key: route_pattern
        url:
            key: [redirection, url]
        permanent:
            key: permanent
            type: boolean
```

Xml:

```
<!-- src/AppBundle/Resources/config/search/Redirection.xml -->

<search-mapping>
    <class name="AppBundle\Document\Redirection">
        <field field="locale" key="locale" />
        <field field="siteName" key="site_name" />
        <field field="routePattern" key="route_pattern" />
        <field field="url" key="redirection, url" />
        <field field="locale" key="locale" />
        <field field="permanent" key="permanent" type="boolean"/>
    </class>
</search-mapping>
```

The `open_orchestra_api.annotation_search_reader` will extract the `descriptionEntity`.

```
$mapping = $this->get('open_orchestra_api.annotation_search_reader')
    ->extractMapping('OpenOrchestra\ModelBundle\Document\Redirection');
```

1.33.4 Notes

In the mapping :

- `type` will take `string` as a default parameter if it is not specified.
- With the mapping in annotation `field` will take the name of the property if it is not set.

By default the xml and yaml mapping files should be located in the folder `Ressources/config/search` of your bundle. To specify another folder, you must change the configuration :

```
open_orchestra_mongo:
    search_metadata:
        directories:
            AppBundle:
                namespace_prefix: "My\\AppBundle"
                path: \@AppBundle/Ressources/config/mymapping
```

1.33.5 Custom search field of columns

By default, Open Orchestra provides differents search fields for columns (text, date, number and boolean). You can specify the fields types in the data attribute `input-header` of the navigation panel link (futher information in document of navigation panel).

To create a custom search field, you should create a backbone view and add it in the `OpenOrchestra.DataTable.ViewFieldConfigurator`.

For instance with the text field:

```
((\$, OpenOrchestra) ->

  ###*
   * @class TextFieldSearchView
  ###
  class TextFieldSearchView extends AbstractSearchFieldView

    events:
      'keyup input.search-column': 'searchColumn'

    ###*
     * required options
     * {
     *   column: {integer} column index
     *   api: {object} DataTable api
     *   domContainer: {object} jquery element
     * }
     * @param {Object} options
    ###
    initialize: (options) ->
      @options = @reduceOption(options, [
        'columnIndex'
        'domContainer'
        'api'
      ])
      @loadTemplates [
        'OpenOrchestraBackofficeBundle:BackOffice:Underscore/datatable/header/textField'
      ]

    ###*
     * @return {this}
    ###
    render: ->
      @setElement @renderTemplate('OpenOrchestraBackofficeBundle:BackOffice:Underscore/datatable/hea
          column: @options.columnIndex
          value: @options.api.column(@options.columnIndex).search()
      )
      @insertFieldInHeader()
```

```
      return @

  OpenOrchestra.DataTable = {} if not OpenOrchestra.DataTable?
  OpenOrchestra.DataTable.ViewFieldConfigurator = {} if not OpenOrchestra.DataTable.ViewFieldConfigu
  OpenOrchestra.DataTable.ViewFieldConfigurator.text = TextFieldSearchView

) jQuery,
  window.OpenOrchestra = window.OpenOrchestra or {}
```

You can extend `AbstractSearchFieldView` to facilitate research and insertion of the field in the header.

The key in `OpenOrchestra.DataTable.ViewFieldConfigurator` defines the name used in data-attribute (`input-header`).

With this custom view, it is possible to create a completly dynamic field, not only static ones. By sending an ajax request in the view, the server can interact with the rendering of the search field for instance.

1.34 Backbone specific view

1.34.1 Context

In the Open Orchestra Back Office, to display an edition form, there are two parts :

- The Twig template given by the server
- The Backbone view which handle all the javascript events

In some cases, during the integration of the Open Orchestra Back Office, the generic view used for the edition page will not suit your needs. There are some new behaviour to add to the form, such as a specific event when you edit a field.

There are two simple solutions to avoid :

- Add some javascript in the form template
- Add a global listener to the javascript project

Both of these solutions are to avoid because :

- They couple the template and the javascript
- They may be checked each time the DOM is modified

To prevent these usage, Open Orchestra provide a way to modify the view used in Backbone.

1.34.2 Usage

As all the CRUD are based on an `entityType`, you will have the possibility to add a view for the `edit` and `add` action on this `entityType`.

```
appConfigurationView.setConfiguration(entityType, action, view)
```

The `view` parameter should be the reference to your specific view. For instance :

```
appConfiguration.setConfiguration('group', 'edit', SpecificFullPageFormView)
```

For the content, the `entityType` parameter will have a different value for each content type :

```
entityType = 'contents_news'; //will be the value for the news content type
```

For the content, the `entityType` will be composed of the content type news and the `contents` string. `contents_news` will be the one for the news content.

1.35 Backbone specific route

1.35.1 Context

In the Open Orchestra Back office, all the routing is managed by the Backbone routing component

The routes already used in the Back office are already declared in some configuration file.

1.35.2 Adding a new route

In a configuration file, you could add a route to the router:

```
(function(router) {
    // Factorise some code in the router (if needed)
    router.fooDisplay = function(fooId) {
        if (selectorExist($('#nav-foo-' + fooId))) {
            this.initDisplayRouteChanges('#nav-foo-' + fooId);
            showFoo($('#nav-foo-' + fooId).data('url'));
        } else {
            Backbone.history.navigate('');
        }
    };

    // Declare the route pattern for the matching process
    router.route('foo/show/:fooId', 'showFoo', function(fooId) {
        this.fooDisplay(fooId);
    });
})(window.appRouter);
```

To declare a new route in Backbone, you have to declare the method run when the route is triggered.

Finally, you should inject the router to the anonymous function.

1.36 Manage assets with Grunt

1.36.1 Context

To automate all the assets management process, the Open Orchestra Back Office uses the task manager Grunt. This tool allows the creation and execution of assets related tasks written in javascript. For instance Open Orchestra uses it to compile *CoffeeScript* files or to create some symlinks.

When running grunt via the command

```
./bin/grunt
```

Grunt build its configuration via the `Gruntfile.js` file located in the root directory of the application.

In order to be as extensible as possible, Open Orchestra offers to each bundle the possibility to extend this *Grunt* configuration by describing its own task and/or task targets. As a result, managing the assets of a newly added bundle simply requires a little change in the *Grunt* configuration.

In this documentation, you will see:

- how to *configure your application to use Grunt*
- how to *add an external bundle using Grunt* tasks / targets
- how to *add your specific tasks / targets to your bundles*

1.36.2 Configuring your application to use *Grunt*

When you install the Open Orchestra Back Office, you have to create a `Gruntfile.js` to manage the assets included in the bundles you are using. This file loads and parses configuration files to produce the Grunt configuration.

The `Grunfile.js` file belongs to your application, so it's up to you to write it. But to avoid you a waste of time, you can use the GruntConfigBuilder AMD module developped for Open Orchestra to load *Grunt* splitted configuration. The file is packaged with the open-orchestra-cms-bundle and located in the GruntTasks folder. The Open Orchestra Back Office demo project show how to use it. The gruntfile.js of this demo is as simple as:

```
module.exports = function(grunt) {
   var appConfig = require('./grunt/app_config.js');
   var GruntConfigBuilder = require(appConfig.GruntConfigBuilder);

   GruntConfigBuilder.init(grunt, appConfig);
};
```

Attached to this minimalist file is a configuration file located at `application_root_folder/grunt/app_config.js`. This AMD module describes three things:

```
module.exports = {
   GruntConfigBuilder:
      /* Path to the GruntConfigBuilder AMD module */
   tasksDir:
      /* Array of paths where to search for Grunt task definitions */
   targetsDir:
      /* Array of paths where to search for Grunt task targets */
};
```

The GruntConfigBuilder attribute refers to the AMD module described earlier and located in the `open-orchestra-cms-bundle`.

The tasksDir attribute is an array listing all folders including *Grunt* task definitions. This is typically here where you register your bundles providing new *Grunt* tasks.

The targetsDir attribute is an array listing all folders including *Grunt* task targets. This is typically here where you register your bundles providing new *Grunt* tasks.

For this mechanism to work properly, tasks and target have to be normalized with a few rules:

- First of all, they must be written as AMD modules that the GruntConfigBuilder will load. You can learn more about this on the require.js site for instance.
- Secondly, tasks and targets have to be separated: one folder for tasks, another one for targets
- The name of a file describing a target must be formatted as following: TASK_NAME.TARGET_NAME.js For instance `clean.symlinks.js` is reffering to the target symlink of the clean task.

If each bundle respects theses rules, everything will load right.

When running *Grunt* (via the shell command `./node_modules/.bin/grunt task_name`), the Gruntfile.js script is launched. The `app_config.js` is loaded and passed to the GruntConfigBuilder which search all configured folders to find tasks and target. It then build a *Grunt* config object, and the task_name provided in the command line is run (or the default task if none is provided).

1.36.3 Adding an external bundle using Grunt

This section explains how to exploit in your application a bundle providing its own *Grunt* tasks and/or task tagets.

If you take a closer look at the (open-orchestra-cms-bundle)(https://github.com/open-orchestra/open-orchestra-cms-bundle), you will see it does not contain the media administration part. To manage the media, you have to add the (open-orchestra-media-admin-bundle)[https://github.com/open-orchestra/open-orchestra-media-admin-bundle]. So this bundle can be taken as an example to illustrate this section.

The `open-orchestra-media-admin-bundle` comes with its own javascript and css files needing to be added to the application assets files. Assuming you have installed the bundle using composer, you still have to configure *Grunt* to use the bundle targets.

For that purpose, you only need to update the `app_config.js`. As the bundle only contains task targets located in the `GruntTasks/Targets` folder, you only have to add this path in the targetsDir attribute of the `app_config.js`. Something like:

```
targetsDir: [
   './grunt/targets',
   './vendor/open-orchestra/open-orchestra-cms-bundle/GruntTasks/Targets',
   './vendor/open-orchestra/open-orchestra-media-admin-bundle/GruntTasks/Targets'
]
```

If the bundle was introducing new *Grunt* tasks, the tasksDir attribute should have been updated the same way.

Grunt is now aware of the different targets present in the `open-orchestra-media-admin-bundle`, but you still have to associate them to a main task for them to be played.

The `open-orchestra-media-admin-bundle` introduces, three targets: one to create new symlinks, one to concatenate some media related js and the last to concatenate media related css files.

You should add the `concat:media_js` target to the main javascript task by modifying the main javascript task (`application_root_folder/grunt/tasks/javascript_task.js`):

```
module.exports = function(grunt) {
   grunt.registerTask(
      'javascript',
      'Main project task to generate javascripts',
      [
         'coffee:discovering',
         'coffee:compile',
         'concat:smartadmin_js',
         'concat:lib_js',
         'concat:orchestra_js',
         'concat:media_js',
         'concat:all_js'
      ]
   );
};
```

When the `javascript` task will be run, the `concat:media_js` task will now be called, and a `media.js` file will be produced.

You can do the same for the stylesheets by modifying the main css task (`application_root_folder/grunt/tasks/css_task.js`):

```
module.exports = function(grunt) {
   grunt.registerTask(
      'css',
      'Main project task to generate stylesheets',
      [
         'less:discovering',
         'less',
         'concat:lib_css',
         'concat:smartadmin_patches_css',
         'concat:orchestra_css',
         'concat:media_css',
         'concat:pre_smartadmin_css',
         'concat:post_smartadmin_css',
         'cssmin'
      ]
   );
};
```

When the `css` task will be run, the `concat:media_css` task will now be called, and a `media.css` file will be produced.

To include the `media.js` file to the final and unique javascript file used by the Open Orchestra Back Office, alter the `application_root_folder/grunt/targets/concat.all_js.js` file:

```
module.exports = {
    src: [
        'web/built/smartadmin.js',
        'web/built/lib.js',
        'web/built/orchestra.js',
        'web/built/media.js'
    ],
    dest: 'web/js/all.js'
};
```

That way, when the `concat:all_js` target will be called, the `all.js` file will include the `open-orchestra-media-admin-bundle` javascripts.

A similar modification on the stylesheets is to be done by modifying the `application_root_folder/grunt/targets/concat.post_smartadmin_css.js` file:

```
module.exports = {
   src: [
      'web/built/smartadminpatches.css',
      'web/built/orchestra.css',
      'web/built/media.css'
   ],
   dest: 'web/css/postsmartadmin.css'
};
```

As for the javascript, the `postsmartadmin.css` file will now include the media stylesheets.

Now you can run the Grunt command (`./node_modules/.bin/grunt`) to regenerate the `all.js` and `postsmartadmin.css` files. If you check these files, you should see the `open-orchestra-media-admin-bundle` assets.

1.36.4 Adding your specific tasks/targets to your bundles

At last, you may need to know how to create your specific tasks for your own bundle. As the process is the same for the javascript and stylesheet files, we will only talk about javascript files.

Let's assume you have created the `FooBundle` and want to manage its assets with Grunt.

As seen in the previous section, concatenation task is resolved in two passes. The first pass groups files by functionnality and the second pass glues the functionnalities together. While the second pass is described in the application (it depends on the used bundles), the first pass is described by the bundle itself. This is done by adding an entry in the main concat task.

First create a directory to put all your tasks targets (`GruntTasks/Targets` for instance). Then you can create a *Grunt* task targets file describing the files to append and naming the file to output the concatenation. The *Grunt* task target file name must follow a specific pattern: TASK_NAME.TARGET_NAME.js. The task loader wil use that name to recreate the main configuration. In our case, we want to create a target named foojs to the concat task, so name your file `concat.foojs.js`. This file can be as simple as:

```
module.exports = {
    src: [
        'web/bundles/FooBundle/js/*.js'
    ],
    dest: 'web/built/foo.js'
};
```

Or if the concatenation order matters, you can be more exhaustive with something like:

```
module.exports = {
    src: [
        'web/bundles/FooBundle/js/js_1.js',
        'web/bundles/FooBundle/js/js_2.js',
        ...
        'web/bundles/FooBundle/js/js_n.js'
    ],
    dest: 'web/built/foo.js'
};
```

When using your foo bundle in an Open Orchestra application, you can inject your task in the app as described in the previous section.

1.37 Deploy asset

1.37.1 Context

As specified in Symfony Reference Documents :

The assets version

> "is used to bust the cache on assets by globally adding a query parameter to all rendered asset paths (e.g. /images/logo.png?v2). This applies only to assets rendered via the Twig asset function (or PHP equivalent) as well as assets rendered with Assetic."

This assets version could be founded in app/config/config.yml.

```
framework:
    assets:
        version: 1.0.0
```

This assets version is used by Open Orchestra Back Office. In order to minimize downloads, the different templates used in the rendering of Back Office are stocked in browser's local storage. These templates have two indexes in local storage : assets version and Back Office language. While rendering, the Back Office requires some underscore template. When asked, the browser will first check in the local storage, with the current asset version and current Back Office language as key. If the the template is found, there will be no request to the server to load it.

For this reason, in case of template updates, This process will require a modification of the asset version parameter in the app/config/config.yml file to allow the changes to appear.

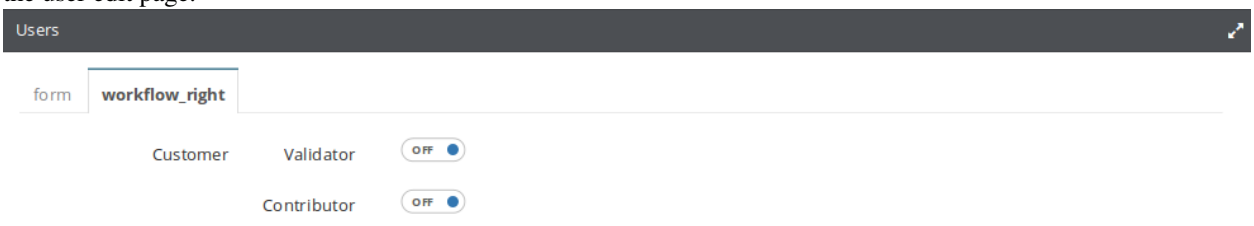
1.38 Add panel to an administration view

1.38.1 Context

In Open Orchestra, if you want to edit an element, you can only display the full form. In some cases you want to embellish the edition by splitting the full form into some smaller one.

Open Orchestra already provides you a way to use a custom Backbone view. This solution may be possible but will require a large customization of the form view.

To prevent you from doing this work, Open Orchestra provides a way to add multiple tabs in your edition form like in the user edit page.



Panel System

Each element facade contains a list of meta links that may be used to display specifics meta-information or forms to add, modify or delete them. The most commons links are :

- `_self_add` url to the add form action
- `_self_form` url to the edit form action
- `_self_delete` url to the delete form action
- `_self_meta` url to meta-information

All of these links are used by Open Orchestra Javascript code to send ajax calls and display different administration forms. Edit page of administration items uses `_self_form` link to display the edit page.

To add a new panel with your own form to this one, add a link to the facade, prefixed by `_self_panel`.

```
$facade->addLink("_self_panel_" + $linkName, $route);
```

Then add panel's title to the form template in the Controller of the panel

```
return $this->renderAdminForm($form, array('title' => $title));
```

Example

Here is an example of how the panel is used to add a workflow form to Users page.

To add this link without modifying the `UserTransformer` which creates the facade, we recommend to listen `UserFacadeEvents::POST_USER_TRANSFORMATION` with an event subscriber to modify the facade by your own after the `UserTransformer` work:

```
class AddWorkFlowLinkSubscriber implements EventSubscriberInterface
{
    protected $router;

    public function __construct(UrlGeneratorInterface $router)
    {
        $this->router = $router;
    }

    public function postUserTransformation(UserFacadeEvent $event)
    {
        $facade = $event->getUserFacade();
        $facade->addLink('_self_panel_workflow_right',
            $this->router->generate($workflowRouteName,
                $workflowParams,
                UrlGeneratorInterface::ABSOLUTE_URL));
    }

    public static function getSubscribedEvents()
    {
        return array(
            UserFacadeEvents::POST_USER_TRANSFORMATION => 'postUserTransformation',
        );
    }
}
```

Register the event subscriber as a service in a configuration file:

```
parameters:
    open_orchestra_workflow_function_admin.subscriber.add_workflow_link.class: OpenOrchestra\Workflow
services:
    open_orchestra_workflow_function_admin.subscriber.add_workflow_link:
        class: "%open_orchestra_workflow_function_admin.subscriber.add_workflow_link.class%"
        arguments:
            - \@router
        tags:
            - { name: kernel.event_subscriber }
```

Then add title to the `formAction` function in the `WorkflowRightController`

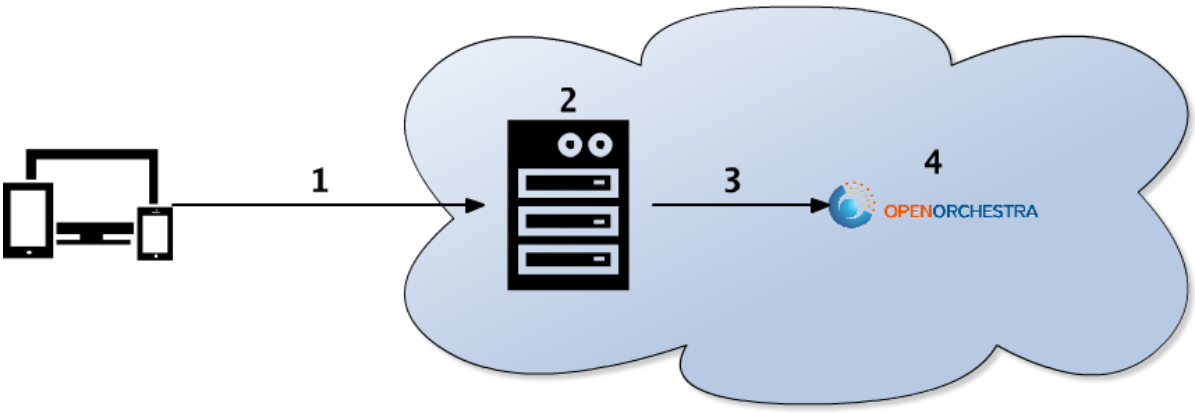
```
/**
 * @Config\Route("/form/{userId}", name="open_orchestra_backoffice_workflow_right_form")
 * @Config\Method({"GET", "POST"})
 */
public function formAction(Request $request, $userId)
{
   // Generate the form

   $title = 'open_orchestra_user_admin.form.title';
   $title = $this->get('translator')->trans($title);
```

```
    return $this->renderAdminForm($form, array('title' => $title));
}
```

1.39 Multi-device

Open Orchestra is compatible with the adaptive way of rendering web pages. The idea is to serve different html versions of the same document according to the device displaying it. For instance an iphone version could show less blocks than a computer version.

This diagram shows the main steps of the process:



1. The client makes a request to the server to get a page. Regardless of the type of client (computer, mobile or tablet), the request is the same.
2. The server catches the request and tag it with the client type
3. The modified request is transmitted to Open Orchestra
4. Open Orchestra processes the request and generates the final html. To render the page, the template engine selects automatically the correct template version according to the previously set header.

1.39.1 Prerequisite

To enable the multi-device rendering, the server entry point (webserver or reverse-proxy) must be able to detect the client device type and to add a request header corresponding to that device type. A good way to do that is to use a tool implementing the Wurfl library. Varnish for instance can do that task. The device detection must also be set up.

1.39.2 Configuration

To activate the adaptive response, all the devices name requiring a specific template have to be defined in the configuration:

```
#app\config\config.yml
open_orchestra_front:
    devices:
        web: ~
        tablet:
            parent: web
        phone:
```

```
            parent: web
        android:
            parent: phone
```

If we take a closer look at this configuration, we can see that four device types are declared : web, tablet, phone and androïd. We can also see that a fallback tree is generated:

- The web device is the default one
- Tablet and Phone device have the same parent: web. This generates a fallback from tablet templates to web templates and from phone templates to web templates. If a tablet template is required but can't be found, the parent template version, ie web one on this example, will be used.
- On an androïd device, the parent is phone, introducing a new fallback.

Fallback mechanism is recursive, so if the parent alternative is not found, the grand-parent will be required, and so on until the default one. So default templates always have to be implemented.

1.39.3 Device detection

As seen on he prerequisite, the server entry point (webserver or reverse-proxy) must implement the Wurfl library. This library tests the User-Agent and normalizes device names. With that information, the server can patch the request headers by adding a parameter `x-ua-device`.

With the previous configuration sample, the generated header `x-ua-device` needs to be equal to 'tablet', 'phone' or 'android'.

Open Orchestra does not provides configuration allowing that detection and headers modification, it's up to you to configure the tool you're using.

1.39.4 Template engine

When a template must be rendered, the method `render` of the template engine is called. So to exploit the multi-devices features, the template engine must be updated with some functionalities.

Out of the box, Open Orchestra enhances the twig engine. If you need to use another template engine, you have to implement this logic on this engine.

1.39.5 Twig example

Here is how Open Orchestra enhances TwigEngine.

First both TwigEngine and TimedTwigEngine classes are extended in the FrontBundle:

```
FrontBundle\Twig\OrchestraTimedTwigEngine
FrontBundle\Twig\OrchestraTwigEngine
```

The `FrontBundle\Twig\Renderable` trait overrides the `render()` method by adding the device name in the template name, exploring the previously set request header `x-ua-device`.

```
if (strstr($name, 'twig')) {
    return str_replace('html.twig', $device . '.html.twig', $name);
}
```

The newly extended template engine can now be declared in the conf:

```
#FrontBundle\Resources\config\twig.yml

parameters:
 open_orchestra_front.twig.orchestra_twig_engine.class: OpenOrchestra\FrontBundle\Twig\OrchestraTwigE

services:
    open_orchestra_front.twig.orchestra_twig_engine:
        class: \%open_orchestra_front.twig.orchestra_twig_engine.class\%
        arguments:
            - \@twig
            - \@templating.name_parser
            - \@templating.locator
            - \@request_stack
            - \%open_orchestra_front.devices\%
        alias: templating
```

1.39.6 Required templates

Once the template engine is able to get the correct template alternative according to the request header, the matching templates have to be created. In our example, each template have to be declined as follow :

- default template version : myTemplate.html.twig
- tablet template version : myTemplate.tablet.html.twig
- phone template version : template.phone.html.twig

Note that if an alternative version is not created, the fallback mechanism will check for the parent alternative. So again, each alternative are optionals, but the default template is required to prevent the fallback mechanism to crash.

1.40 ESI blocks

Open Orchestra leverages the power of ESI blocks to optimize page rendering and increase performance. ESI blocks are a way to split the rendering of a page into different elements that will be rendered in separate requests by the web server. This behavior allows the application to apply HTTP cache to some parts of the page and refresh only the outdated parts when needed.

In the Front Office application, each block is rendered in an ESI block using the render_esi() twig function. The blocks will therefore be cached by an HTTP cache unless the method `DisplayBlockInterface::isPublic()` returns false.

1.40.1 Setup

In order to use the power of ESI blocks, it's needed to have a reverse proxy in front of the web application that will be able to process them. Examples of reverse proxies are Varnish or the HttpCache component of Symfony.

If there is no ESI-capable reverse proxy configured, Symfony will still be able to render your pages with just HTML so the application won't suffer any drawbacks.

HttpCache configuration

The configuration of Symfony for using HttpCache is explained in the Http cache documentation

Varnish configuration

To let Symfony know that there is an ESI-capable server (Varnish) installed in front of the PHP webserver, this ESI server should set an HTTP header in the request to be passed to the application, and parse the response to check for ESI usage.

This is done inside the varnish configuration file.

```
sub vcl_recv {
    set req.http.Surrogate-Capability = "varnish=ESI/1.0";
}

sub vcl_fetch {
    if (beresp.ttl > 0s) {
        unset beresp.http.Set-Cookie;
        return(deliver);
    }
}
```

1.40.2 Selective flushing with tags

This feature is not available with the Symfony HttpCache component.

Principle

The main interest of ESI tags is to give the possibility to flush some parts of a cached page at different times. While it's interesting to use this ability by setting a time-to-live (TTL) to define the lifetime of a block, Open Orchestra uses Varnish's ability to flush ESI blocks on the fly on certain conditions.

Whenever an ESI block is cached in Varnish, the HTTP headers of the response will be saved along with the content. This mechanism allows to set an X-Cache-Tags header with its value being a set of tags that indicate in which conditions it would be flushed. Therefore, when an action leads to one of the tags being marked as outdated, a request is sent to varnish so it will ban all cached data linked to this tag.

Behavior for nodes and blocks

When caching a node, the tags that are used refer to the identifier of the node, the language and id of the associated website.

For blocks, two kind of tags are applied : generic ones and specific ones.

The generic tags are added for all kind of blocks, they depend on : * the block type * the node which contains the block * the language * the website

Additionally, each rendering strategy for blocks can define a list of specific tags by implementing the method `DisplayBlockInterface::getCacheTags()`.

It's possible to define new cache tags for some data to be flushed at optimal times, depending on the specifics of the application. These tags can then be flushed by calling `OpenOrchestra\DisplayBundle\Manager\CacheableManager::invalidateTags()` when needed.

1.41 Themes

The CSS and JS files of the themes are rendered with a call to the Twig functions `openOrchestraCss()` and `openOrchestraJs()`, each one is taking as a parameter the theme name to render.

See the Twig extensions page for more details.

1.41.1 Configuration

In order for the Twig extension to correctly find the resource files needed for a theme, the configuration must describe the path of these files individually for each theme available.

Here is a full example of a valid configuration for a theme:

```
open_orchestra_theme:
    themes:
        foobar:
            name: "Theme example"
            stylesheets:
                - AcmeBundle:foo:css/style.css
                - AcmeBundle:bar:css/font.css
            javascripts:
                - AcmeBundle:bar:js/main.js
```

In this case `foobar` is the identifier of the theme as referred in the call to the Twig functions. Three keys can be defined for a theme :

- `name` is a required entry
- stylesheets is an array of CSS files to use
- javascripts is an array of JS files to use

The path of a resource is determined from the configuration. For instance, for the value `AcmeBundle:foo:css/style.css`, the file will be located at `[AcmeBundle folder]/Resources/public/themes/foo/css/style.css`.

1.42 Twig extensions

Some twig extensions have been developed specifically for Open Orchestra and are available for use.

1.42.1 Media functions

If you are using the media bundles, you will probably have to manipulate media within your twig templates. You can use these twig extensions to help you:

- `display_media(mediaId, format)`: render a media (or an alternative) using the DisplayMediaManager
- `get_media_url(mediaId, format)`: get the url of a media (or an alternative)
- `get_media_title(mediaId)`: get the title of a media, in the current language
- `get_media_alt(mediaId)`: get the alt of a media, in the current language

1.42.2 Theme functions

In Open Orchestra, themes are easily configurable by defining the list of JavaScript and CSS files used in the configuration of the application for each theme.

In order to render the HTML tags for these files, two functions are available in Twig :

- `openOrchestraCss(themeId)` : render a html stylesheet tag linking the css of the theme
- `openOrchestraJs(themeId)` : render a html javascript tag linking the javascript of the theme

1.42.3 Tree generation

A tree structure is the most easy to use when it can be browsed recursively from root to leaves. However, data is not always formatted in this way, and may be just a flat array of objects, each one having a property defining its own parent. This is the case of the nodes that are displayed in a menu block : each one know its parent but they are all at the same level when they are retrived from the database.

To solve this issue, the `OpenOrchestra\DisplayBundle\Manager\TreeManager` class has been created, with its method `generateTree()` being wrapped in the `tree_formatter()` Twig function. Provided an array of ReadNodeInterface, it will return a multi-level array of nodes for easy HTML rendering.

1.43 Media

When a media is uploaded in the media library from the Back Office, 2 things happen. First a thumbnail is generated to display it in the media library. Then, according to the media type, several alternatives can be generated too. For instance, for an image, some alternative formats are generated. You could imagine a video being reencoded in different formats at this step.

1.43.1 Media storage

All files (originals uploaded by the user, the thumbnails and the alternatives) are stored via the Gaufrette component and the KnpGaufretteBundle which binds Gaufrette to Symfony. Gaufrette is an abstraction layer for accessing files wherever the physical copy is located (locally on the disk, on an FTP server, in the cloud...).

By default, Open Orchestra stores the media locally on the webserver of the application. But if it is required to change the location of the files, the configuration can be updated to change the adapter used by Gaufrette.

All adapters that can be natively used are described in the configuration documentation, along with the details of the configuration to set up.

If another Gaufrette filesystem is used instead of the default one, it should be notified to Open Orchestra through the `open_orchestra_media.filesystem` configuration parameter, which expects the identifier of the new filesystem to use.

1.43.2 Upload strategies

When a new file is uploaded, a media document is created and the file is moved to the storage via Gaufrette. But some required process are also made, and some other optional can be too. For instance, to be rendered in the Back Office gallery, a preview thumbnail must be generated. If the media is an image, several alternatives in different sizes are also generated. If the file is a sound, maybe some new variants could be generated with different bitrates. Those operations are completed by the alternative strategies. A strategy exists for each supported mime-type.

When a new media is created due to a file upload, the `MediaEvents::MEDIA_ADD` event is fired. The `OpenOrchestra\MediaAdmin\EventSubscriber\MediaCreatedSubscriber` catch this event to track the created media. Later after the response matching the media creation has been sent to the client, this subscriber also catch the `KernelEvents::TERMINATE` event. It then asks the `OpenOrchestra\MediaAdmin\FileAlternatives\FileAlternativesManager` to generate a thumbnail and the alternatives. So, the manager search a strategy able to deal with the uploaded file.

You can alter the alternatives generation by extending the existing strategies or by creating new ones.

1.43.3 The image alternatives

A media image can be displayed in different formats depending on the needs of the application. So when an image is uploaded to the media library, several alternatives are generated according to the configured formats. A format is simply a configuration entry combining one or severall parameters among the following:

- max_height: defines the maximum height of the image to generate, the width is calculated to keep a correct ratio
- max_width: defines the maximum width of the image to generate, the height is calculated to keep a correct ratio
- compression_quality: defines the compression rate of the alternative. This parameter is always required when defining a new format

Open Orchestra comes with three default formats (Fixed height, Fixed width and Rectangle) which are mostly samples to show what is possible. You should always replace these sample formats by the one required by your application. To do this just add your formats in the `app/config/config.yml` file

Here is an example of configuration defining two custom alternative formats:

```
open_orchestra_media_admin:
  files:
    alternatives:
      image:
        formats:
          my_custom_max_height:
            max_height: 100
            compression_quality: 95
          my_custom_rectangle:
            max_height: 200
            max_width: 300
            compression_quality: 95
```

You can create as many formats as required, as soon as you create only one, it will replace the Open Orchestra sample formats.

To complete your customization, you should think about adding a translated label to your custom settings as these formats are used in several locations of the Back Office. The translation key used is `open_orchestra_media_admin.form.media.MY_CUSTOM_FORMAT_KEY` where MY_CUSTOM_FORMAT_KEY is the key of the format used in the `app/config/config.yml` description, for instance my_custom_max_height and my_custom_rectangle in the previous example.

1.43.4 Front display

Each media file is identified by a unique hash key used by Gaufrette to retrieve the physical file in the configured storage. To generate the HTML tag rendering a media, Open Orchestra provides the Twig function `display_media`.

Instead of a direct access to the file, we recommand you to use the `MediaController` from the `MediaFileBundle`. The `show` action will retrieve the binary data with Gaufrette and send it back to the browser.

1.44 Media display

1.44.1 Context

To improve your website performances, a common way is to download the page data from multiple servers. In fact, in the HTTP protocol, a browser can make only two parallel requests to the same domain. Recent browsers (such as `Chrome` or `Firefox` can perform up to 10 simultaneous requests). In the following documentation, we will just assume that browsers can make only two simultaneous requests to the same domain.

An easy way to load and display the medias would be to add a new virtual host to your existing Open Orchestra installation. This will allow your browser to increase the number of parallel requests that can be done as several domain are available. It will then reduce the global latency of each page's download

Request scheme:

This solution is not ideal because you will need to boot the whole Open Orchestra kernel each time you will request an image.

1.44.2 Open Orchestra Solution

A simple solution would be to install only the `open-orchestra-media-file-bundle` as a single `Symfony` application.

Basic installation

The `open-orchestra-media-file-bundle` can run on its own if you require it in the `composer.json` file:

```
{
    "require": {
        "incenteev/composer-parameter-handler": "~2.0",
        "open-orchestra/open-orchestra-media-file-bundle": "*"
    },
}
```

Then enable all the required bundles in your `AppKernel.php` file:

```
new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
new Symfony\Bundle\SecurityBundle\SecurityBundle(),
new Symfony\Bundle\TwigBundle\TwigBundle(),
new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),

new Knp\Bundle\GaufretteBundle\KnpGaufretteBundle(),

new OpenOrchestra\BaseBundle\OpenOrchestraBaseBundle(),
new OpenOrchestra\MediaBundle\OpenOrchestraMediaFileBundle(),
```

Ease your installation

To facilitate this installation, we provide you with the `open-orchestra-media-demo` repository which implements all those modifications.

To install it :

```
composer create-project open-orchestra/open-orchestra-media-demo ./
```

1.45 SmartAdmin

SmartAdmin is a responsive admin theme which uses Bootstrap made by Myorange.

Open Orchestra's back office uses the specific design of SmartAdmin and only some of these JavaScript components (JarvisWidget and SmartNotification).

This doc will help you to add new features in the back office and respect its general design.

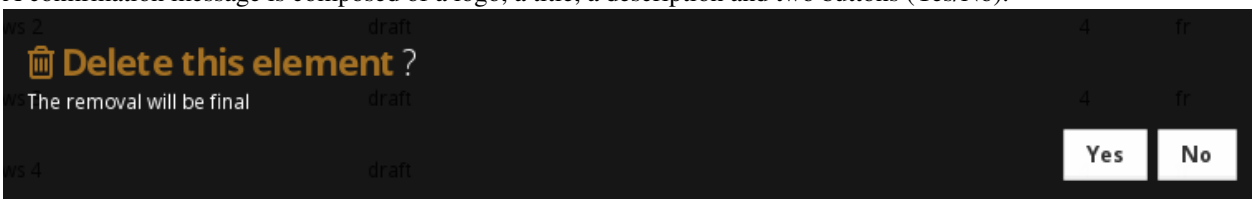
1.45.1 SmartConfirm

Description

The `smartConfirm` function allows to display a confirmation message. It uses the `SmartMessageBox` function of component SmartNotification. It is called during various events like the deleting of an item (nodes, blocks, contents, ...).

It must be used in replacement of the standard JavaScript confirm.

A confirmation message is composed of a logo, a title, a description and two buttons (Yes/No).



The function `smartConfirm` takes 4 parameters:

1. the logo css class.
2. the message title.
3. the message description.
4. an array of callback functions to be called when buttons are clicked and the parameters of callbacks functions.

The array must contain three keys `yesCallback`, `NoCallback` and `callBackParams` .

The keys `yesCallback` and `NoCallback` which refer respectively to the functions managing the click event on Yes and No buttons. The key `callBackParams` is an array which contains the parameters for callbacks functions.

View

SmartConfirm uses two templates, one for buttons (`smartConfirmButton._tpl.twig`) and another to display the logo and the title (`smartConfirmTitle._tpl.twig`). **The title does not need to include a question mark as the template adds one at the end of it**.

Translation

The translation of the buttons is done with the translation component of Symfony in the template. For title and description, they must be translated before being passed to the `smartConfirm` function .

Example

This example in coffeeScript show a confirmation message. The variables `confirm_title` and `confirm_text` contains the translated title and the translated description:

```
smartConfirm(
  'fa-trash-o',
  confirm_title,
  confirm_text,
  callBackParams:
    url: url
  yesCallback: (params) ->
    $.ajax
      url: params.url
      method: 'DELETE'
      success: (response) ->
        if redirectUrl != undefined
          displayMenu(redirectUrl)
        else
          redirectUrl = appRouter.generateUrl 'showDashboard'
          displayMenu(redirectUrl)
        return
      error: (response) ->
        $('.modal-footer', this.el).html response.responseJSON.error.message
        return
  noCallback: ->
    $("#OrchestraBOModal").modal "show"
)
```

1.46 Events available in Open Orchestra

Open Orchestra dispatches lots of events in different places of the code, a lot of them being used for logging the website's activity. This pages lists the events than can be subscribed to during execution.

1.46.1 Categories of events

Nodes

These events are defined in the class `OpenOrchestra\ModelInterface\NodeEvents` and inherit `OpenOrchestra\ModelInterface\Event\NodeEvent`.

- **NodeEvent**
  - Node creation : NODE_CREATION
  - Node removal : NODE_DELETE
  - Node update : NODE_UPDATE
  - Node duplication : NODE_DUPLICATE
  - Adding a language to a node : NODE_ADD_LANGUAGE
  - Status update for a node : NODE_CHANGE_STATUS
  - Path update for a node : PATH_UPDATED
- **NodeEvent** [Area]

  - Area update : NODE_UPDATE_AREA
  - Area removal : NODE_DELETE_AREA
- **NodeEvent** [Block]
  - Block update : NODE_UPDATE_BLOCK
  - Block position update : NODE_UPDATE_BLOCK_POSITION
  - Block removal : NODE_DELETE_BLOCK

Template

These events are defined in the class `OpenOrchestra\ModelInterface\TemplateEvents` and inherit `OpenOrchestra\ModelInterface\Event\TemplateEvent`.

- **TemplateEvent**
  - Template creation : TEMPLATE_CREATE
  - Template removal : TEMPLATE_DELETE
  - Template update : TEMPLATE_UPDATE
  - Area removal in template : TEMPLATE_AREA_DELETE
  - Area update in template : TEMPLATE_AREA_UPDATE

Media

These events are defined in the classes `OpenOrchestra\Media\MediaEvents` and `OpenOrchestra\Media\FolderEvents` and inherit `OpenOrchestra\Media\Event\MediaEvent` and `OpenOrchestra\Media\Event\FolderEvent` respectively.

- **FolderEvent**
  - Folder creation : FOLDER_CREATE
  - Folder removal : FOLDER_DELETE
  - Folder update : FOLDER_UPDATE
- **MediaEvent**
  - Image upload : ADD_IMAGE
  - Image removal : MEDIA_DELETE
  - Image cropping : MEDIA_CROP
- **ImagickEvent**
  - Image resizing : RESIZE_IMAGE

Content

These events are defined in the class `OpenOrchestra\ModelInterface\ContentEvents` and inherit `OpenOrchestra\ModelInterface\Event\ContentEvent`.

- **ContentEvent**
  - Content creation : CONTENT_CREATION

  - Content removal : CONTENT_DELETE
  - Content update : CONTENT_UPDATE
  - Content duplication : CONTENT_DUPLICATE
  - Content status update : CONTENT_CHANGE_STATUS

1.46.2 Administration

Content types

These events are defined in the class `OpenOrchestra\ModelInterface\ContentTypeEvents` and inherit `OpenOrchestra\ModelInterface\Event\ContentTypeEvent`.

- **ContentTypeEvent**
  - Content type creation : CONTENT_TYPE_CREATE
  - Content type removal : CONTENT_TYPE_DELETE
  - Content type update : CONTENT_TYPE_UPDATE

Keyword

These events are defined in the class `OpenOrchestra\ModelInterface\KeywordEvents` and inherit `OpenOrchestra\ModelInterface\Event\KeywordEvent`.

- **KeyWordEvent**
  - Keyword creation : KEYWORD_CREATE
  - Keyword removal : KEYWORD_DELETE

Redirection

These events are defined in the class `OpenOrchestra\ModelInterface\RedirectionEvents` and inherit `OpenOrchestra\ModelInterface\Event\RedirectionEvent`.

- **RedirectionEvent**
  - Redirection creation : REDIRECTION_CREATE
  - Redirection removal : REDIRECTION_DELETE
  - Redirection update : REDIRECTION_UPDATE

Roles

These events are defined in the class `OpenOrchestra\ModelInterface\RoleEvents` and inherit `OpenOrchestra\ModelInterface\Event\RoleEvent`.

- **RoleEvent**
  - Role creation : ROLE_CREATE
  - Role removal : ROLE_DELETE
  - Role update : ROLE_UPDATE

Sites

These events are defined in the class `OpenOrchestra\ModelInterface\SiteEvents` and inherit `OpenOrchestra\ModelInterface\Event\SiteEvent`.

- **SiteEvent**
  - Site creation : SITE_CREATE
  - Site removal : SITE_DELETE
  - Site update : SITE_UPDATE

Status

These events are defined in the class `OpenOrchestra\ModelInterface\StatusEvents` and inherit `OpenOrchestra\ModelInterface\Event\StatusEvent`.

- **StatusEvent**
  - Status creation : STATUS_CREATE
  - Status removal : STATUS_DELETE
  - Status update: STATUS_UPDATE
- **StatusableEvent**
  - Status change : STATUS_CHANGE

StatusableEvent is use when changing status of a node, content or media reference.

Themes

These events are defined in the class `OpenOrchestra\ModelInterface\ThemeEvents` and inherit `OpenOrchestra\ModelInterface\Event\ThemeEvent`.

- **ThemeEvent**
  - Theme creation : THEME_CREATE
  - Theme removal : THEME_DELETE
  - Theme update : THEME_UPDATE

Users

These events are defined in the class `OpenOrchestra\UserBundle\GroupEvents` and `OpenOrchestra\UserBundle\UserEvents` and inherit `OpenOrchestra\UserBundle\Event\GroupEvent` and `OpenOrchestra\UserBundle\Event\UserEvent` respectively.

- **GroupEvent**
  - Group creation : GROUP_CREATE
  - Group removal : GROUP_DELETE
  - Group update : GROUP_UPDATE
- **UserEvent**
  - User creation : USER_CREATE

- User removal : USER_DELETE
- User update : USER_UPDATE

1.46.3 Example of event dispatching

You can easily dispatch your own events or Open Orchestra events as you would normally do with Symfony.

```
$this->get('event_dispatcher')->dispatch(NodeEvents::NODE_UPDATE, new NodeEvent($node));
```

1.47 API

1.47.1 API access

The connection to the API is based on the OAuth2 protocol. The OAuth2 protocol provides a client a secure access to an API on behalf of an user.

A client is represented by a key/secret pair. In order to use the API, the client must request a token.

The token is a string that refers to the client and optionally to the user. It is used to authenticate the client when he makes a request to the API.

1.47.2 Client

The API clients can be managed in the administration section of the back office.

The creation of a client is done through the add form. In the form, the key/secret pair is automatically generated by the server and the client can be specified like trusted.

The trusted parameter allows you to prevent a connexion from a particular application to the api.

1.47.3 Token

Open Orchestra implements three ways to get a token. All of those strategies are described in the OAuth2 protocol.

Client credentials

This strategy provides a token linked only to an API client.

To obtain it, send the request:

```
/oauth/access_token?grant_type=client_credentials
```

with the HTTP header:

```
Authorization: Basic ENCODED_PAIR
```

Where `ENCODED_PAIR` is the string **key:secret** encoded in base64 (obviously key and secret depend on the client).

Resource owner password

This strategy provides a token linked to an API client and a user.

To obtain it, send the request:

```
/oauth/access_token?grant_type=password&username=USERNAME&password=PASSWORD
```

Where USERNAME is user's name and PASSWORD is user's password.

with the HTTP header:

```
Authorization: Basic ENCODED_PAIR
```

Where `ENCODED_PAIR` is the string **key:secret** encoded in base64 (obviously key and secret depend on the client).

Refresh token

This strategy provides a way to refresh an expired token linked to a client and maybe to an user.

To obtain it, send the request:

```
/oauth/access_token?grant_type=refresh_token&refresh_token=REFRESH_TOKEN
```

Where REFRESH_TOKEN is the refresh token linked to the token used.

With the HTTP header:

```
Authorization: Basic ENCODED_PAIR
```

Where `ENCODED_PAIR` is the string **key:secret** encoded in base64 (obviously key and secret depend on the client).

This call will block all other token linked to the user and the client.

This method can also be called only once.

1.47.4 API usage

To use the API, the parameter access_token=YOUR_TOKEN should be added in the query.

1.47.5 Firewall configuration

In the case where the API is used only, a stateless firewall must be defined in `security.yml` file:

```
api:
    pattern: ^/api/
    oauth2: ~
    anonymous: false
    security: true
    stateless: true
```

If the back office is installed and you want to access the API also, the API firewall cannot be stateless because the back office must be able to query the API while already being authenticated to the application.

To access the API with the OAuth2 protocol and the back office connection, two firewalls sharing the same context must be defined:

```
api:
    pattern: ^/api/
    oauth2: ~
    anonymous: false
    security: true
    context: openorchestra
main:
    pattern: ^/
    form_login:
        provider: fos_userbundle
        csrf_provider: form.csrf_provider
    anonymous: true
    context: openorchestra
    logout:
        path:   /logout
        target: /admin
```

New authentication strategy

In the case when the strategies already available in Open Orchestra do not fits your needs, you can add your own authentication strategy.

Lets say you want to create the foo authentication strategy which will take the `foo` parameter in the request and create a token with foo as access token code.

The class should implement `OpenOrchestra\BaseApi\OAuth2\Strategy\StrategyInterface`.

In this interface, there are three method:

- `supportRequestToken`, to check if this strategy should be used to create a token
- `requestToken`, to create and save the token
- `getName`, to name the strategy

Let's say that the `objectManager` and the `serializer` are injected to the `FooStrategy`.

```
class FooStrategy implements StrategyInterface
{
    public function supportRequestToken(Request $request)
    {
        return $request->get('foo');
    }

    public function requestToken(Request $request)
    {
        $fooParameter = $request->get('foo');

        $accessToken = AccessToken::create();
        $accessToken->setCode($fooParameter);

        $this->objectManager->persist($accessToken);
        $this->objectManager->flush($accessToken);

        $tokenFacade = new AccessTokenFacade();
        $tokenFacade->accessToken = $accessToken->getCode();

        return $tokenFacade;
    }
```

To use this strategy, send a request to : `/oauth/access_token?foo=bar`. The response should return a json object looking like:

```
{ "access_token": "bar" }
```

You can then use the `bar` access token to call the api: `/api/url?access_token=bar`

1.48 API transformer

1.48.1 Context

To decouple the data exposed through the API and the data stored in the model, Open Orchestra uses the Facade design pattern.

If you want to add a new entity to your project and need to expose it through the API, an easy way would be to directly serialize the object to render it.

This solution is not ideal for a main reason: the response is going to move each time you will modify your model.

1.48.2 Open Orchestra solution

Open Orchestra gives you the opportunity to decouple the data exposition from the data storage.

To simplify this documentation, let's say that we want to expose a `Foo` document.

```
class Foo
{
    /**
     * @var string
     */
    public $bar;
}
```

Facade

The facade is the object that will be exposed through the API, let's also say you need to put the value of `Foo::bar` inside a `baz` property.

Create the `FooFacade` with the `baz` property:

```
namespace FooBundle\Facade;

use JMS\Serializer\Annotation as Serializer;
use OpenOrchestra\BaseApi\Facade\FacadeInterface;

class FooFacade implements FacadeInterface
{
    /**
     * @Serializer\Type("string")
     */
    public $baz;
}
```

The `FooFacade` class should implement the `FacadeInterface` to be recognized as a facade in the rest of the application.

Transformer

As the transformation for the `bar` to `baz` property is not intuitive, you will need to create a Transformer to perform the modification.

Create the `FooTransformer`:

- It must implement the `OpenOrchestra\BaseApi\Transformer\TransformerInterface`
- Be registered as a service with the tag `open_orchestra_api.transformer.strategy`

```
namespace FooBundle\Transformer;

use OpenOrchestra\BaseApi\Facade\FacadeInterface;
use OpenOrchestra\BaseApi\Transformer\AbstractTransformer;
use FooBundle\Facade\FooFacade;

class FooTransformer extends AbstractTransformer
{
    public function getName()
    {
        return 'foo';
    }

    public function transform($foo)
    {
        $facade = new FooFacade();
        $facade->baz = $foo->bar;

        return $facade;
    }
}
```

And the declaration :

```
foo.transformer.foo:
    class: FooBundle\Transformer\FooTransformer
    tags:
        - { name: open_orchestra_api.transformer.strategy }
```

To limit the dependency, Open Orchestra provides a `TransformerManager` knowing all the transformers. This way, you are able to directly call an other transformer with the method `$this->getTransformer('foo')`.

You also have access to the `Router` and to the `GroupContext` (see group context page)

Usage

In the controller, you should access your transformer via `TransformerManager`.

```
class FooController extends Controller
{
    public function listAction()
    {
        $foo = new Foo(); //Create foo object

        return $this
            ->get('open_orchestra_api.transformer_manager')
            ->get('foo')
            ->transformer($foo);
```

```
    }
}
```

Entity modification

Once your entity has been serialized as a Facade to expose the data, you might need to modify it.

To help you do it, the `TransformerInterface` provides you a `reverseTransform` method. This method take as argument the facade send in the request and the entity to modify.

First, add a method to the `FooController`:

```
// Set a route on the 'POST' method and a paramConverter
public function editAction(Request $request, Foo $foo)
{
    // Deserialize the content of the request in the FooFacade
    $facade = $this
        ->get('jms_serializer')
        ->deserialize(
            $request->getContent(),
            'FooBundle\Facade\FooFacade',
            $request->get('_format', 'json')
        );

    // Perform the reverse transform operation
    $this
        ->get('open_orchestra_api.transformer_manager')
        ->get('foo')
        ->reverseTransform($facade, $foo);

    // Check if the entity is valid
    if ($this->isValid($foo)) {
        //Save the entity

        return new Response('', 200);
    }

    // If the entity is not valid, return the violations
    return new Response(
        $this
            ->get('jms_serializer')
            ->serialize(
                $this->getViolations(),
                $request->get('_format', 'json')
            ),
        400
    );
}
```

Then complete the `reverseTransform` method from the `FooTransformer`, I will keep the same parameter.

```
public function reverseTansform(FacadeInterface $fooFacade, $fooEntity = null)
{
    if (isset($fooFacade->baz)) {
        $fooEntity->bar = $facade->baz;
    }
```

```
    return $fooEntity;
}
```

1.49 API group context

1.49.1 Context

To increase readability and reduce the code duplication in the API, a transformer will only be linked to one entity and one facade. This way, each time you will transform an entity into a facade, it will be done using the same parameters. In some case, you do not want to display all the data from a facade in a certain context.

For instance, when you list all the nodes, the area and blocks might not be useful.

The `JMSSerializerBundle` allows you to hide some properties on the serialization by using some serialization group. The main issue with this method is that you will still have to complete all the data in the facade.

We are going to see how we can use the Open Orchestra project to limit the amount of data set in the facade depending on the context

1.49.2 Usage

In a controller action, you can directly add an annotation :

```
/**
 * @Api\Groups({"GROUP_FOO"})
 */
public function fooAction()
```

With the requirements :

```
use OpenOrchestra\ApiBundle\Controller\Annotation as Api;
```

Then in the transformer, you can directly call the `hasGroup` method :

```
$this->hasGroup("GROUP_FOO");
```

This method will check if the group has been added in the action annotation. Now you can either choose to set or not some properties in the facade.

1.49.3 Going further

All groups are stored in the `open_orchestra_api.context.group` service. As this service has a scope container you can modify it in your code by adding a specific group.

```
$this->container->get('open_orchestra_api.context.group)->addGroup('GROUP_BAR');
```

In all the following transformer, you can now check if the `GROUP_BAR` is present :

```
$this->hasGroup('GROUP_BAR') // true
```

1.50 Routing

1.50.1 Base URL

In Open Orchestra, the base url of a website is composed by the domain and possibly a language prefix. To manage all the possibilities, Open Orchestra associates each website in the database with website aliases which contain:

- **domain**: domain name
- **language prefix**: used by Open Orchestra to define the language
- **scheme** : to choose with which protocol page can be accessed

Here is an example of some base URL generated for a website associate to two domains: `dev.example.com` and `test.example.com`, each of them, with english as default language and french as secondary one.

Alias Id	Scheme	Domain	Language	Language prefix	Base URL
0	http	dev.example.com	fr	fr	http://dev.example.com/fr
1	http	dev.example.com	en	none	http://dev.example.com
2	http	test.example.com	fr	fr	http://test.example.com/fr
3	https	test.example.com	en	none	https://test.example.com

1.50.2 URL Pattern

The url pattern is added to the `base URL` to generate the url. For a node, the url pattern can be chosen by the contributor or is generated automatically from the page title.

It uses Symfony rules :

- Variables to extract between {}, ex : `/example/{param}/test`
- Pattern have to be unique

See the `Url pattern` chapter from Node Configuration for more details.

1.50.3 Redirection

Open Orchestra allows users to determine redirection in the back office. The redirection form contains some arguments:

- **site name**: contextual site which triggers the redirection
- **site language**: contextual language which triggers the redirection
- **pattern to redirect**: pattern that will be the URL of the redirection
- **node where the redirection is headed**: redirect to a node
- **Url to redirect to**: redirect to url if node is not specify
- **permanent redirection**: Use to adapt response code

Example:

Base Site	Language	Pattern to redirect	Node	URL to redirect
http://example.fr	english	oo	none	http://www.open-orchestra.com
http://example.fr	french	open-orchestra	HOME	none

If an user comes on `http://example.fr/oo` so the english site with the good pattern, it will be redirect to the `http://www.open-orchestra.com` website.

If an user comes on `http://example.fr/fr/open-orchestra` so the french site with the good pattern, it will be redirect to the `HOME` node of the french page.

But if an user comes on `http://example.fr/fr/oo`, it will have a `404 not found` error page because redirections are attached to languages.

1.50.4 Implementations

Open Orchestra provides two alternatives mechanisms for the routing :

- Get the url directly from the database (default configuration)
- Dump the route in a cache file

We are going to see how to configure one or the other, then which solution you could choose for your project.

Routing from the database

In the database, Open Orchestra saves all the routes (for the nodes, and redirections) with document implementing the `RouteDocumentInterface`.

When you perform a request, the router called is the `OpenOrchestraDatabaseRouter`. It will check in the database for a matching route.

As this configuration is the default one, you have no modification to make to the project configuration.

Routing in the file

Open Orchestra is also able to dump all the routes (nodes and redirection) inside a cache file.

When you perform a request, the router called is the `OpenOrchestraRouter`. It will check in the cache file for a matching route.

To configure this router, you have to specify the different loader you are using (nodes and redirection) in the `routing.yml` file :

```
open_orchestra_database:
    resource: '.'
    type: database

open_orchestra_redirection:
    resource: '.'
    type: orchestra_redirection
```

You also have to modify the project configuration, in the `config.yml` file :

```
open_orchestra_front:
    routing_type: file
```

How to choose between both of the solutions

This part will not tell you which routing type you should use, but only list some condition to use one or the other.

You have a lot of pages, and they are updated by a big number of people all the time :

- Use the database routing, no cache file will be updated each time a route is created

There is only a few update of the pages :

- If you want to lower the number of query to the database, use the file routing
- Use the database routing if there is no restriction

You are not using the Open Orchestra Back Office :

- Use the file routing, you will only have to configure the application
- Use the database routing, but note that you need to update the `route_document` collection each time a node is updated

1.51 Extend document and repository

1.51.1 Use case

Open Orchestra presents a list of model object to the integrators. To customize those objects, it could be necessary to extend the implemented documents and repositories.

1.51.2 How to

These extensions will be done by creating extended classes for document (ODM context) or repository. To reference these classes, the following lines will be added to config.yml.

```
open_orchestra_model:
    document:
        content:
            class: "MyBundle\\Document\\Content"
            repository: "MyBundle\\Repository\\ContentRepository"
```

The header of the two extended classes could be :

Extended Content

```
    <?php
    namespace MyBundle\Document;

    use OpenOrchestra\ModelBundle\Document\Content as BaseContent;
    use Doctrine\ODM\MongoDB\Mapping\Annotations as ODM;
    /**
     * Extended Class Content
     * @ODM\Document(
     *   collection="content",
     *   repositoryClass="MyBundle\Repository\ContentRepository"
     * )
     */
    class Content extends BaseContent
    {
```

Extended Content Repository

```
    <?php
    namespace MyBundle\Repository;
```

```
    use OpenOrchestra\ModelBundle\Repository\ContentRepository as BaseContentRepository;
    /**
     * Class ContentRepository
     */
    class ContentRepository extends BaseContentRepository
    {
```

As OpenOrchestra use interface for dependency injection, it is necessary to update the configuration with the new extended class.

```
resolve_target_documents:
    OpenOrchestra\ModelInterface\Model\ContentInterface: MyBundle\Document\Content
```

1.52 Add a special listener to a Content Attribute

1.52.1 Context

When working with the Content and ContentType, you might have the need to add a specific listener.

In some case, the field can store a collection which needs to be initialized by a listener to set all the data.

The main issues is that all the contentAttribute are stored as a raw field in mongoDB. By doing this, the data you will save will be typed but the data you get back will only be an array.

Here is an example of how you can take profit of the Content architecture to store some TranslatedValue field inside.

1.52.2 Usage

Prerequisites

In the Open Orchestra model abstraction, the Translated Value are stored as a collection of `TranslatedValueInterface` implementation. Each object will store the language code and the translated value.

When you use it in a simple form, you will need to add a listener on the form :

```
$builder->addEventListener(FormEvents::PRE_SET_DATA, array($this->translateValueInitializerListener,
```

Then you can simply add the form type :

```
$builder->add('descriptions', 'translated_value_collection');
```

The entity should also implement the `TranslatedValueContainerInterface`.

Issue description

As you use a dynamic Field Type to describe your content, the `Content` document cannot implement the `TranslatedValueContainerInterface` to give the name of the translated fields.

Moreover, the listener has to be configured in the root form. As the configuration is fully dynamic, you cannot add it.

Solution

A simple solution is to create a specific form type, add an initializer, and take care of the data transformation.

First create the specific form type :

```
class TranslatedValueContainerAttributeType extends AbstractType
{
    public function getName()
    {
        return 'translated_value_container_attribute';
    }
}
```

Then inject the default value initializer :

```
protected $translatedValueDefaultValueInitializer;

/**
 * @param TranslatedValueDefaultValueInitializer $translatedValueDefaultValueInitializer
 */
public function __construct(TranslatedValueDefaultValueInitializer $translatedValueDefaultValueInitia
{
    $this->translatedValueDefaultValueInitializer = $translatedValueDefaultValueInitializer;
}
```

Finally, create the listener to initialize the data :

```
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $translatedValueDefaultValueInitializer = $this->translatedValueDefaultValueInitializer;
    $builder->addEventListener(FormEvents::PRE_SET_DATA, function(FormEvent $event) use ($translatedV
        $data = $event->getData();
        if (is_null($data)) {
            $values = new ArrayCollection();
            $translatedValueDefaultValueInitializer->generate($values);
            $data['value'] = $values;
        } elseif (is_array($data) && array_key_exists('value', $data)) {
            foreach ($data['value'] as $key => $element) {
                $newElement = new TranslatedValue();
                $newElement->setLanguage($element['language']);
                $newElement->setValue($element['value']);
                $data['value'][$key] = $newElement;
            }
        }
        $event->setData($data);
    });

    $builder->add('value', 'translated_value_collection');
}
```

When you create the contentAttribute, there are no data stored, so the `$data` is null. You need to create an `ArrayCollection` which will be later saved and use the initializer to generate all the data you need.

This is the same workflow as the other translated values.

As you can see, there is no event listener bound on Submit events. The raw field type automatically accepts `array` and `ArrayCollection`.

When you want to display the data stored, you need to transform it from an `array` to a `TranslatedValue` document.

Note : This should be done in a DataTransformer.

Configuration

In order to make this form type available in the field type, you need to declare it :

```
translated:
    label: translated
    type: translated_value_container_attribute
    options:
        required:
            default_value: false
```

1.53 Customize Content Attribute Display

1.53.1 Context

In Open Orchestra the `Content` are composed of a set of base attributes and a collection of `ContentAttribute` (See content documentation for more information). The `ContentAttribute` are configured by the content type. In order to have minimum constraints on contents, Open Orchestra allows you to store any type of `value` in the `ContentAttribute`.

Sometimes, there is no easy way to render the value to the Back Office user. A custom `ContentAttribute` could be stored as an array, which cannot be displayed directly in the dataTables plugin. In that way the `ContentAttribute` class provides a specific attribute `stringValue` which is the representation of the `value` in an HTML string.

1.53.2 Transformers

Open Orchestra uses a `ValueTransformer` strategy manager to generate the HTML strings for multiple `value` types. Open Orchestra already provides some strategies located in `OpenOrchestra\Backoffice\ValueTransformer\Strategies`.

Strategy creation

Service used as `ValueTransformer` must be tagged as `open_orchestra_backoffice.value_transformer.strategy`

```
tags:
    - { name: open_orchestra_backoffice.value_transformer.strategy }
```

Such services also need to implement `OpenOrchestra\Backoffice\ValueTransformer\ValueTransformerInterfac`

Strategy access

Open Orchestra uses the `OpenOrchestra\Backoffice\ValueTransformer\ValueTransformerManager` as an easy way to access every transformation strategies. All `ValueTransformer` strategies are registered in the `ValueTransformerManager`. The `transform` method of the manager:

- searches the first transformation strategy supporting the current `fieldType` and `value` of the `ContentAttribute`
- uses this strategy's `transform` method to return the `stringValue`

1.53.3 Examples

Transformation available

Value	StringValue	Strategy
“string”	“string”	none
1	“1”	IntegerToHtmlStringTransformer
null	“none”	NullToHtmlStringTransformer
Object()	Object::__toString()	ObjectToHtmlStringTransformer
array(‘foo’ => array(‘bar’))	‘ - bar ’	ArrayToHtmlStringTransformer

New transformation

Let’s say that the field with the type `foo` will store relation to the `FooDocument`. You will need to add a transformer in order to create a `stringValue` that could be displayed on the Back office. In our example the `bar` properties should be displayed.

You need to implement the three methods of the interface :

- `getName` : give the name of the transformer
- `support` : take the `fieldType` and the `value` to decide if the transformation should be applied
- `transform` : will transform the value

In our case, the support method will check the field type and if the value is null or not.

```
public function support($fieldType, $value)
{
    return $fieldType == 'foo' && $value != null;
}
```

The transform method will need the `fooDocument` repository to find the document then will return the `bar` properties.

```
/**
 * $value = mongoId;
 */
public function transform($value)
{
    $fooDocument = $this->fooRepository->find($value);

    return $fooDocument->getBar();
}
```

1.54 Prevent document suppression when they are embedded

1.54.1 Context

In a document oriented database (like Mongo), to improve performances, a standard practice is to avoid relations between documents using embed documents. However, when you want to suppress a document that is used by other documents a little issue can appear.

In our case, the Open Orchestra Backoffice presents some status that can be used in multiple documents like the `Node`, `Content`, `Role`, ... Deleting a `Status` document could lead to a loss of consistence from the database.

1.54.2 Open Orchestra implementation example

In Open Orchestra, if you look to the `DeleteStatusVoter` you will see it using the `StatusUsageFinder` to check if the `Status` can be deleted.

If you add a document implementing `StatusableInterface` :

- Your document repository should implement `StatusableElementRepositoryInterface`
- You should add your repository to the `StatusUsageFinder` service

Implement the interface

In the `NodeRepository` class, the interface implements the `StatusableElementRepositoryInterface` :

```
interface NodeRepositoryInterface extends StatusableElementRepositoryInterface
```

And the repository implement the method :

```
public function hasStatusedElement(StatusInterface $status)
{
    $node = $this->findOneBy(array('status' => $status));
    return $node instanceof NodeInterface;
}
```

Modify the configuration

To add your repository, `foo.repository.bar` in our example, you should modify the usage finder definition in your bundle :

```
class FooBarExtension extends Extension
{
    public function load(array $configs, ContainerBuilder $container)
    {
        $definition = $container->getDefinition('open_orchestra_backoffice.usage_finder.status')
        $definition->addMethodCall('addRepository', array(new Reference('foo.repository.bar')));
    }
}
```

With this modification, each time the `StatusUsageFinder` is called, it will also check in your own repository if the given status is used.

1.55 Dashboard widgets

The Back Office dashboard is the place to show to the user information concerning him. It's like a blackoard on which are pinned post-it. Each post-it is a widget. Open Orchestra comes with four widgets:

- My 10 latest published nodes
- My 10 latest nodes
- My 10 latest published contents
- My 10 latest contents

As a developer, you can customize your application dashboard by choosing widgets and/or create new ones. Here is how to do.

1.55.1 Configuring the dashboard

To choose the widgets to display in the dashboard, add a key `open_orchestra_backoffice.dashboard_widgets` to your app configuration. Then set the list of widgets you want to use. Here is an exemple in yaml format showing the 4 standard widgets:

```
open_orchestra_backoffice:
    dashboard_widgets:
        - last_nodes
        - draft_nodes
        - last_contents
        - draft_contents
```

1.55.2 Creating a widget

To build a new widget, you only have to create its Backbone view and register it in Open Orchestra.

Creating a backbone view

As the Back Office is build with `BackboneJs`, all widgets are Backbone views added to the main view from scratch or extend one of the pseudo-abstract layers used by the standard Open Orchestra widgets. To get some examples, you can take a look at the files located in `BackOfficeBundle/Ressources/public/coffee/dashboard/widget`

Your widget html can be anything, but we suggest you to use the jarvis widget template to have a visual uniformity in the dashboard. This template can be found in `BackOfficeBundle/Ressources/views/BackOffice/Underscore/dashboard/widget/abstract/abstractW`

Declaring the view

To make your newly created widget available in Open Orchestra, you need to add it to the `appConfigurationView`. This object is used in the Back Office to list all backbone views and access them when required. To add your view to the configurator, use something like this:

```
jQuery(function() {
  return appConfigurationView.setConfiguration(
    'dashboard_widgets',
    'my_widget_name',
    'my_widget_backbone_view'
  );
});
```

where `my_widget_name` is the widget key to use in the configuration and `my_widget_backbone_view` is the Backbone view you have just created. That done, your widget is ready to be added to your app configuration.

1.56 Error pages

Symfony allows a developper to customize error pages shown by the framework when a matching HTTP status is returned. For instance a developper could create a specific Error 500 page.

Open Orchestra allows you to go further for two types of status : 404 (resource not found) and 503 (site under maintenance). You actually can contribute those pages directly in the Back Office, customizing them on a per site / per language combination.

1.56.1 Site under maintenance: Error 503

Displaying a custom 503 page for an Open Orchestra site requires 3 steps :

1. Contribute the matching nodes in the Back Office and publish it
2. Dump the node into a static html file via the command line (or let a cron task generate it)
3. Configure the Virtual Host to serve that file when the site is under maintenance

Contributing the 503 error page

To contribute the 503 error page, in the Back Office navigation menu, click on "Error pages", then on "503". This will show a standard node configuration page. The node can be contributed for each available language of the Front Office. To make the page available, it must be published.

Dump the node into statics files

Open Orchestra comes with a command line to dump the error nodes into static files. You can execute that command line at the root of your project or place the command in a cron task to dump the files periodically. The command is the following:

```
app/console orchestra:errorpages:generate [siteId]
```

The siteId is an option allowing you to specify a specific site to limit the dump operation. Without that option the pages are dumped for every site.

For each site, the command will look for error nodes and especially the published versions. The script will then look to the available site aliases. When an error node is published in the site alias language, the script generates the matching files. For instance if mysite.com declares an alias in english and a second one in french, and if a 503 error node is published in english, then the command generates a 503 file in english for the first alias. Nothing is generated for the french alias as there's no 503 node published in french.

The generated files can be found under the following path:

web/*siteId*/alias-*aliasId*/errorPage503.html

where *siteId* is the site id and *aliasId* the alias id.

Configuring the Virtual Host

To use the 503 static page, the apache Virtual Host of the site must be configured using the ErrorDocument directive. Here comes a version of the configuration allowing you to avoid the Virtual Host file edition each time you want to put on/off your site.

where *siteId* is the site id and *aliasId* the alias id.

With such a configuration, you don't have to rewrite the Virtual Host nor restart Apache on maintenance operation. You only have to put a file named maintenance.on in the root directory of your application to validate the maintenance rewrite and to remove it to put your site on back.

1.56.2 Error 404 - Page not found

The 404 error page requires less efforts than the 503 version to be used. Once a 404 node is contributed and published, it immediately replaces the standard 404 error message from Symfony.

Be aware that if the node is published in only one language, it will only be available for the site aliases using that language. That way, a site in several languages can present an Open Orchestra 404 page for one language and a Symfony 404 page for others.

Note: like 503 pages, 404 pages are also dumped by the console command orchestra:errorpages:generate. They can so be used with virtual hosts configuration on some cases if needed.

1.57 Using robots.txt

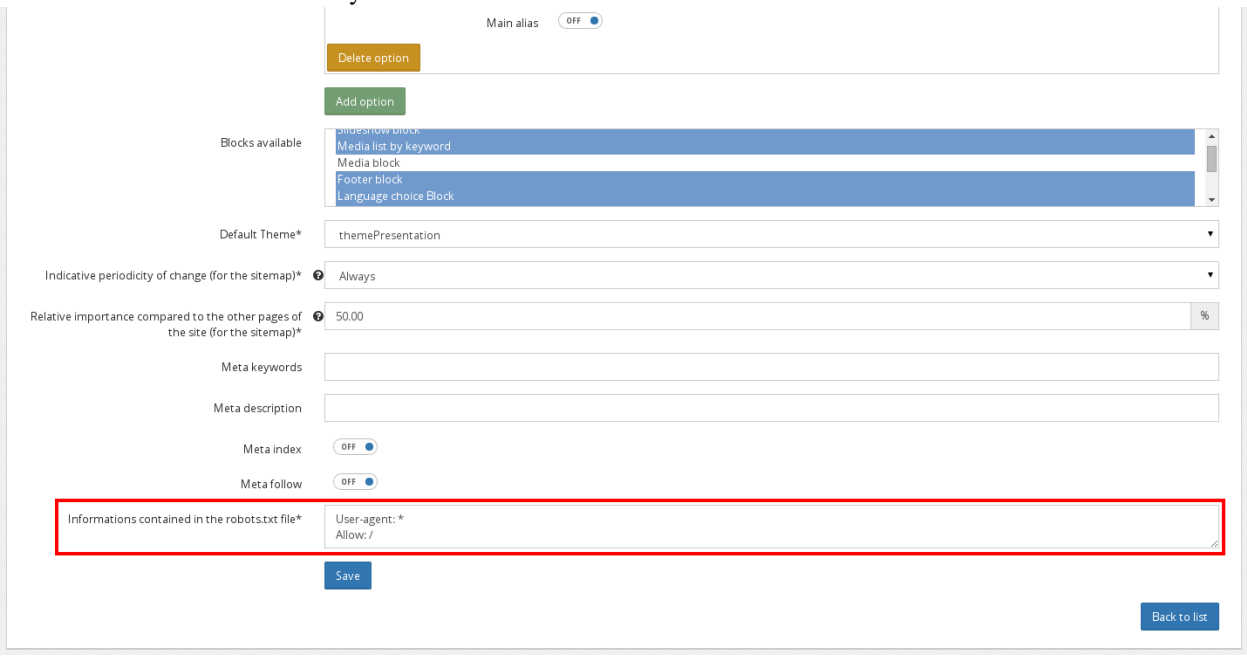
As Open Orchestra is a multisite platform, you will have to manage several robots.txt files for a single installation. To simplify the maintenance of their content, they can be directly edited in the Back Office. But contribution is only the first step. To be available, these information require to be dumped in files and to be correctly served to the client.

Creation of `robots.txt` files with Open Orchestra is achieved in 3 steps:

- *files contribution*
- *files generation*
- *files routing*

1.57.1 Files contribution

Contributing the content of a `robots.txt` file is straight forward. As it is specific to each website, you can do it in the website edit form, located in the Administration submenu. Fill the matching textarea: the content of the `robots.txt` file will be exactly what is written here.



1.57.2 Files generation

Once contributed, the content of the `robots.txt` is only known from the database. To serve it to a client requesting it, two approaches are possible: use a controller action looking in the database for the information matching the current site, or serve a previously generated file containing these information.

To enhance perfomances, Open Orchestra choose the second option. This solution prevents any PHP action and database access. As these files are static they can easily be cached by a reverse proxy.

Once a `robots.txt` data is contributed, you have to manually dump it into a file. Open Orchestra provides a console command, available on your Front installation:

```
app/console orchestra:robots:generate [–siteId=SITEID]
```

This command dumps one or all the `robots.txt` files depending on the siteId parameter presence. To be accessible by web client, the generated files are stored in the web directory of your application. Each file is stored in a sub-directory named by the site Id. For instance if you have three sites whose id are 'site-1', 'site-2' and 'site-3', the `robots.txt` files will be dumped here:

- web/site-1/robots.txt
- web/site-2/robots.txt
- web/site-3/robots.txt

A good practice is to use that command in a cron, to refresh periodically the content of the files.

1.57.3 Files routing

When a client requests a `robots.txt`, it should be located at the root of the domain. But the files are generated somewhere else, in a subfolder named by the site id. As a result, the webserver must be configured to redirect to the matching file: this can be done by a rewrite rule . The type of configuration to tweak depends on you server type, for instance on Apache this is done via the Virtual Host mechanism.

Here is a configuration example for `Apache`, in the case of a site with id 'my-site':

Once Apache is reloaded, the rewrite rule is used and the `robots.txt` file is accessible.

Every further modification will modify the file content but not its name, so you don't have to modify the rewrite rule. Since then you can update the file periodically without having to restart the webserver.

1.58 Using sitemap.xml

As Open Orchestra is a multisite platform, you will have to manage several sitemap.xml files for a single installation. To simplify the maintenance of their content, information contained in the files can be directly edited in the Back Office. But contribution is only the first step. To be available, these information require to be dumped in files and to be correctly served to the client.

Creation of `sitemap.xml` files with Open Orchestra is achieved in 3 steps:

- *files-contribution*
- *files-generation*
- *files-routing*

1.58.1 Files contribution

Contributing the content of a `sitemap.xml` file is done in the Back Office, and two locations are involved: one for the default values, and one for the specific values.

The first place to go is website edition. In the website edition form, you can set up the default values for all pages of the site. These settings are the periodicity of change of a page (the changefreq setting) and the relative importance of a page compared to the other pages (the priority setting). When set, these values are used when a page does not specify its own settings.

To provide specific settings for a page, you have to edit the matching node setting form. Here you can set the priority and the changefreq only for this page. The settings given here override the website default values. For more info about node configuration you can read the node configuration documentation.

1.58.2 Files generation

Once contributed, the content of the `sitemap.xml` is only known from the database. To serve it to a client requesting it, two approaches are possible: use a controller action looking in the database for the information matching the current site, or serve a previously generated file containing these information.

To enhance perfomances, Open Orchestra choose the second option. This solution prevents any PHP action and database access. As these files are static they can easily be cached by a reverse proxy.

Once a `sitemap.xml` data is contributed, you have to manually dump it into a file. Open Orchestra provides a console command, available on your Front installation:

> app/console orchestra:sitemaps:generate [–siteId=SITEID]

This command dumps one or all the `sitemap.xml` files depending on the siteId parameter presence. To be accessible by web client, the generated files are stored in the web directory of your application. Each file is stored in a sub-directory named by the site Id, then by the alias Id. For instance if you have two sites whose are is 'site-1', 'site-2', that 'site-1' get two aliases and 'site-2' has one, the `sitemap.xml` files will be dumped here:

- web/site-1/alias-0/sitemap.xml
- web/site-1/alias-1/sitemap.xml
- web/site-2/alias-0/sitemap.xml

Note that the alias indexes (alias-0, alias-1, etc...) refers to the position of the aliases in the website edit form.

A good practice is to use that command in a cron, to refresh periodically the content of the files.

1.58.3 Files routing

When a client requests a `sitemap.xml`, it should be located at the root of the domain. But the files are generated somewhere else. As a result the webserver must be configured to redirect to the matching file: this can be done by a rewrite rule . The type of configuration to tweak depends on your server type, for instance on Apache this is done via the Virtual Host mechanism.

Here is a configuration example for `Apache`, in the case of the first alias of a site with id 'my-site':

Once Apache is reloaded, the rewrite rule is used and the `sitemap.xml` file is accessible.

Every further modification will modify the file content but not its name, so you don't have to modify the rewrite rule. Since then you can update the file periodically without having to restart the webserver.

1.59 BBCode extension

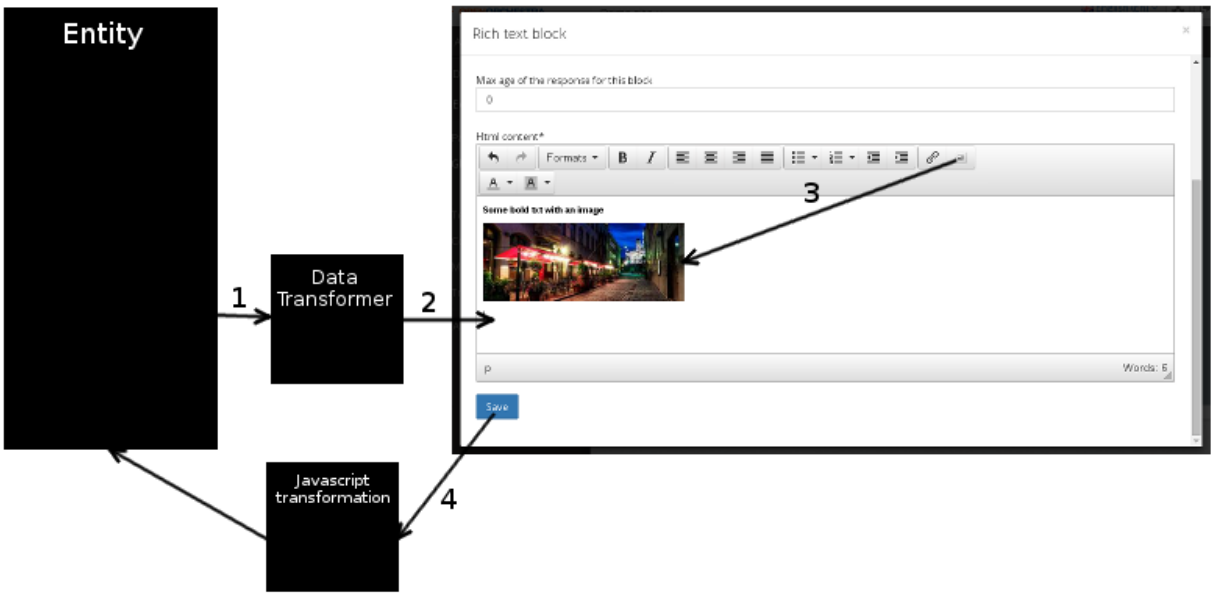
In the Back Office, a Wysiwyg editor is used on several places where a rich text can be contributed. You can encounter it for instance on certain blocks or Content Types. Although most of the information entered in the editor is stored in the database in html format, you can also extend the possibilities by introducing custom BBCode tags. This is partially done via the Open Orchestra BBCode parser.

It's important to note that by default, not all tags are enabled in BBCode format, only those you want to declare. For instance, putting a text in bold in the rich text editor does not produce by default a `[b]` tag. Out of the box, there is only one BBCode tag: `media`. But it is possible to add your own tags.

In this documentation you will see how the BBCode feature fits in the main processes and how to create a new BBCode tag in your bundle.

1.59.1 Wysiwyg process

Open Orchestra uses TinyMce to render a rich text input. Normally, TinyMce generates html tags, but Open Orchestra has enhanced it to use BBCode. Here is the workflow triggered when editing an object using a form containing rich text fields:



1. Each rich text field goes into a data transformer to interpret the BBCode tags. This data transformer calls the Open Orchestra BBCode parser to detect and transform valid BBCode tags into html tags.
2. The resulting html is given to the TinyMce editor which treats it with its standard logic.
3. When the user clicks in the toolbar to inject a BBCode tag (for instance by selecting a media), TinyMce inject the html representation of the BBCode, to be able to visually render the tag. For instance in the media case, you will see it in the editor.
4. When the user saves the form, TinyMce replaces via javascript the html tags by their BBCode representations.

1.59.2 How to create a new BBCode and add it to the parser

Open Orchestra provides a simple ways of adding your own custom BBCode tags, depending on their nature. Two kind of tags exists:

- Simple tags are just text replacements. For instance a `[b]Some Text[/b]` should be replaced by `<b>Some Text</b>`
- Complex tags require more than a simple regexp conversion. For instance the tag `[media=format]MediaId[/media]` requires to fetch the media in the database, to detect its type and then to to return the corresponding html tag.

To add a new BBCode tag to the parser, you need to create its definition. There are two ways of adding a definition. To create a simple tag you can quickly set a configuration. But if you prefer, or if your tag requires a php analysis, you can use a Symfony service.

Create a simple BBCode tag via configuration

To create a simple BBCode definition, its possible to use a trivial configuration. The parameter `open_orchestra_bbcode.code_definitions` contains an array of declared simple tags. Here is the declaration of the bold tag:

```
bold:
    tag: b
    html: <strong>{param}</strong>
```

- The key `bold` is the name of the definition used internally by the parser.
- The `html` key is the html pattern of the code. The string `{param}` will be replaced by the content of the BBcode tag.
- The `tag` key is the label used in the BBCode tag. In this example, with the `tag` key set to `b`, the BBCode tag is `[b]{param}[/b]`.

Simple tags can also include some options. For instance the tag `[h]` which matches a `<h*>` html tag requires an option to indicate the level of the header. For instance `[h=1]Some text[/h]` will produce a `<h1>` html tag. You still can declare this in a configuration file like this:

```
title:
    tag: h
    html: <h{option}>{param}</h{option}>
    parameters:
        use_option: true
```

To declare that a tag uses option, you must add the key `parameters.use_option` and set it to true. Note that when declaring a tag with option, that option becomes mandatory. So if the option is not mandatory, you should declare two versions of the tag: one with option and one without. That way the parser will use one or the other according to the presence of an option. The `html` key uses both {param} and {option} string to indicate where to place those informations. The resulting BBCode will be `[h={option}]{param}[/h]`

Create a BBCode tag via a service

BBCode definitions can also be declared by services. A service tagged with `open_orchestra_bbcode.code_definitions` will be recognized as introducing new definitions. Such a service includes a BBCode definition collection and serves it via the method `getDefinitions`. So you are free to code your own service, but it must implement `OpenOrchestra\BBcodeBundle\Definition\BBcodeDefinitionCollectionInterface`.

The definition collection must be composed of formatted elements implementing `OpenOrchestra\BBcodeBundle\Definition\BBcodeDefinitionInterface`. If your definition is a simple regex replacement, as the one seen in the previous section, you can use an Open Orchestra definition

factory. But to create a BBCode tag with a more complex logic, you will have to create your own definition class. Both ways are explained in the next sections.

Create simple definitions with the definition factory

You can quickly generate simple definitions using the OpenOrchestraBBcodeBundleDefinitionBBcodeDefinitionFactory. This factory can be accessed via the service named `open_orchestra_bbcode.definition.factory`

To generate a new simple definition with this factory, call the method `create` with the following parameters:

- `$tag`: the BBCode tag, ie `b` for the tag [b]
- `$html`: the html representation of the tag, ie `<strong>{param}</strong>` for the [b]tag
- `$use_option`: whether the tag uses option or not

So to create a bold tag definition object, you can write:

```
$definition = $container->get('open_orchestra_bbcode.definition.factory')
    ->create('b', '<strong>{param}</strong>');
```

And to create a header tag definition:

```
$definition = $container->get('open_orchestra_bbcode.definition.factory')
    ->create('h', '<h{option}>{param}</h{option}>', true);
```

As mentionned before, to let Open Orchestra know your new definition, you have to add it to a definition collection class implementing `OpenOrchestra\BBcodeBundle\Definition\BBcodeDefinitionCollectionInterface`. Then make that collection a Symfony service tagged with `open_orchestra_bbcode.code_definitions`.

Create complex definitions with a custom class

If the tag you want to create requires a more complex rendering, for instance a `youtube` tag searching in its parameter for a valid youtube id, you have to create your own definition class. This definition class must implement `OpenOrchestra\BBcodeBundle\Definition\BBcodeDefinitionInterface`. The getHtml method receives a BBcodeElementNode representing the BBCode tag and returns the matching html version of the tag. So it is the location to inject the specific rendering logic of your tag.

Instead of creating a full class from scratch, you can also extend the class `OpenOrchestra\BBcodeBundle\Definition\BBcodeDefinition` and override the getHtml method.

Here is what could look like the getHtml method for the `youtube` tag:

```
public function getHtml(BBcodeElementNode $el)
{
    $content = '';
    foreach ($el->getChildren() as $child) {
        $content .= $child->getAsBBCode();
    }
    $foundMatch = preg_match('/v=([A-z0-9=\-]+?)(&.*)?$/i', $content, $matches);
    if (!$foundMatch) {
        return $el->getAsBBCode();
    }
    return '<iframe width="640" height="390" src="http://www.youtube.com/embed/'
        . $matches[1] . '" frameborder="0" allowfullscreen></iframe>';
}
```

Unlike the configuration or the factory ways, creating a definition with such a class allows you to access the Symfony container and to use services to do a complex transformation.

Again to let Open Orchestra know this definition, you have to add it to a definition collection implementing `OpenOrchestra\BBcodeBundle\Definition\BBcodeDefinitionCollectionInterface` and tag that service with `open_orchestra_bbcode.code_definitions`.

1.60 Manage Bower and Npm Dependencies

Open Orchestra uses npm to manage some server side javascript libraries and bower to manage the client side libraries.

To facilitate the addition of bower and npm dependencies, Open Orchestra uses the composer plugin composer-extra-assets which allows to declare such dependencies in the composer.json

1.60.1 Adding Bower and Npm Dependencies

To add your own Bower and Npm dependencies you have to specify them in the `composer.json` of your application or your bundle.

```
"require": {
    "koala-framework/composer-extra-assets": "~2.0"
},
"extra": {
    "require-npm": {
        "grunt": "0.4.*"
    },
    "require-bower": {
        "jquery": "*"
    },
}
```

If you add Npm dependencies in your bundle, you can add the `expose-npm-packages` attribute to the composer. With this option, the Npm package are available in the node_modules directory of Composer's root and not in the directory of your bundle.

```
"require": {
    "koala-framework/composer-extra-assets": "~2.0"
},
"extra": {
    "require-npm": {
         "gulp": "*"
    },
    "expose-npm-packages": true
}
```

1.60.2 Bower.json and Package.json files

`composer-extra-assets` generates `bower.json` and `package.json` files. It's not recommended to add your Bower and Npm dependencies in these files because it will be overwritten when the next `composer update|install` will be launched.

1.61 Using other database type

1.61.1 Context

Open orchestra uses by default a MongoDB database. If you want to use another type of database like SQL, you will need to tune some settings and to replace some files.

1.61.2 Configuration: object manager

By default, Open Orchestra uses Doctrine ODM to map database documents to objects. If you want to use another document manager, the new manager must implement `Doctrine\Common\Persistence\ObjectManager`. You also have to declare it in the configuration:

```
open_orchestra_base_bundle:
    object_manager: namespace/of/objectManager
```

1.61.3 Override mongo documents and repositories

Some bundles have dependencies to mongoDB. The way to replace them depends on the type of dependencies.

Complete replacement of the package

The `open-orchestra-model-bundle` and `open-orchestra-base-api-mongo-model-bundle` are mongoDB implementations of Open Orchestra entities. To use another database, you have to replace these packages by new ones adapted to your database.

Bundle replacement

Some bundles included in Open Orchestra packages are model bundle developped for mongoDB. To use another database, these bundles must not be activated in the `appKernel` and must be replaced by yours. These are:

`open-orchestra-workflow-function-bundle : - WorkflowFunctionAdminBundle - WorkflowFunctionModelBundle`

`open-orchestra-media-bundle: - MediaModelBundle`

`open-orchestra-cms-bundle - GroupBundle - ModelLogBundle`

`open-orchestra-user-bundle - UserModelBundle`

1.62 Elasticsearch indexation

1.62.1 General note

This documentation describes the provisionning modification for the `vagrant` environment. All the modifications could be done on all the different environments.

1.62.2 Elasticsearch

Elasticsearch is a common indexor based on the lucene engine written in Java. To install it on the computer running the application, you should add the `indexation` group to the file `provisioning/hosts/vagrant` :

```
[indexation]
admin.openorchestra.1-1.dev
```

You should also add some configuration to the `provisioning/hosts/group_vars/vagrant` file:

```
elasticsearch_version: 1.4
elasticsearch_config:
  network.bind_host: 192.168.33.10
  network.host: 192.168.33.10
  network.publish_host: 192.168.33.10
  http.bind_host: 192.168.33.10
  http.host: 192.168.33.10
  http.publish_host: 192.168.33.10
elasticsearch_plugins:
  - name: mobz/elasticsearch-head
    check_file: /usr/share/elasticsearch/plugins/head/index.html
```

This way, Elasticsearch will be binded to the public ip of the vagrant box.

We also recommand you to install the plugin head which could give you usefull information about the cluster.

Run the provisionning (`vagrant provision`) to install Elasticsearch.

Go on http://192.168.33.10:9200/ to check if it is running properly. You should see `status: 200` in the json response.

Go on http://192.168.33.10:9200/_plugin/head/ to see the plugin head working.

1.62.3 Elasticsearch and Open Orchestra

To perform all the indexation in Elasticsearch an Open Orchestra bundle has been created. It is the open-orchestra-elastica-bundle.

Add in your `composer.json` file :

```
"open-orchestra/open-orchestra-elastica-bundle": "*"
```

In the `app/AppKernel.php` file :

```
new OpenOrchestra\ElasticaBundle\OpenOrchestraElasticaBundle(),
```

Add the Elasticsearch listening address in the `app/config/config.yml` file :

```
open_orchestra_elastica:
    host: 192.168.33.10
```

`192.168.33.10` is our local configuration, this can vary on your installation.

After running the `composer update` command, you should see some new command appear when running `php app/console` :

```
$ php app/console | grep orchestra
```

Those commands are :

```
orchestra:elastica:index:create          Create an index in elastic search
orchestra:elastica:index:drop            Drop the index in elasticsearch
orchestra:elastica:populate              Populate the content index with the contents
orchestra:elastica:schema:create         Load the schema from the content types
```

Creating the index

The first time you install Elasticsearch you can use the command `orchestra:elastica:index:create` to create the index.

```
$ php app/console orchestra:elastica:index:create
```

For more advanced users, you can directly go on Elasticsearch and create your index with the name `content`.

There is only an output if there is an error during the process.

Creating the schema

Once your index is created, you should create the schema to help Elasticsearch store and retrieve your datas.

The first time you are using Elasticsearch on an existing installation, you should use the command :

```
$ php app/console orchestra:elastica:schema:create
```

During the project lifetime, the schema will be automatically updated each time the `ContentType` are updated.

There is only an output if there is an error during the process.

Populating the index

The first time you are using Elasticsearch, you could populate the index with the existing datas, using the command :

```
$ php app/console orchestra:elastica:populate
```

During the project lifetime, the indexed data will be automatically updated each time you publish a `Content`.

There is only an output if there is an error during the process.

1.63 Elasticsearch add new indexed document

1.63.1 General note

To use this documentation, you should already have installed Elasticsearch on your environment.

This documentation describes how to create the schema, and how to populate the index of the new document.

1.63.2 Schema creation

To generate the schema of your document, you need to create an initializer that implements `ElasticaSchemaInitializer`.

This initializer should be tagged with :

```
tags:
    - { name: open_orchestra_elastica.schema_initializer.strategy }
```

All initializers are called by the console command `orchestra:elastica:schema:create`.

This initializer can call a schema generator class that should implement `DocumentToElasticaSchemaGeneratorInterface`.

1.63.3 Index population

To populate the index with your document data, there are two possible actions :

- Populate the index from the data of you database
- Modify the index each time one of your document is modified

Populate from the database

To populate the index from the database, you need to create a populator that implements `ElasticaPopulatorInterface`.

This populator should be tagged with :

```
tags:
    - { name: open_orchestra_elastica.populator.strategy}
```

This populator queries all the documents that you want to index from the database. This service should call a indexor which implement `MultipleDocumentIndexorInterface`.

All populators are called when you run the `orchestra:elastica:populate` command.

The indexor gets an array of documents, transforms them into elastica document and send the indexed document to the database. The indexor should call a transformer which implement `ModelToElasticaTransformerInterface`.

The transformer gets a document and output an elastica document.

Modify the index

Create or update a document

When a document is created or modified, you should listen to the `DocumentEvents::CREATE` or `DocumentEvents::UPDATE` events.

The listener should call an indexor that implements `DocumentIndexorInterface`

The indexor gets the document and index an elastica document. The indexor calls the transformer that you created in the previous section to perform the transformation.

Delete a document

When a document is deleted, you should listen to the `DocumentEvents::DELETE` event.

The listener should call an indexor that implements `DocumentDeletorInterface`.

The deletor gets a document and send a delete query to Elasticsearch with the elastica document. The indexor calls the transformer that you created in the previous section.

1.64 Requirements

Open Orchestra is developped on a specific environment, making that environment our reference. Open Orchestra is certified and supported on that reference environment. This document provides you the different technologies required to reproduce that reference environment.

1.64.1 Basic Requirements

These requirements are mandatory to run Open Orchestra on a reference environment.

Operating System	Debian 8.2 (Jessie)
Web server	Apache/2.4.10
Database	Mongodb 2.6.11
PHP (mod_php + cli)	PHP 5.5
PHP Extensions	Mongo > 1.2.12, <1.7-dev Imagick OPcache > 7.0.6 Mbstring Curl Intl
npm	1.4.28
Node.js	0.10.36

1.64.2 Optional requirements

These requirements are optionals to run Open Orchestra. They are used on our reference environment for optional functionalities.

Reverse proxy	Varnish 3.0.7

For a development server, we recommend you to add some packages:

Blackfire	
Selenium (with Xvfb)	2.44.0

1.65 Server Configuration

To know more about server configuration see requirements.

1.65.1 Apache

To use Open Orchestra, you need to configure the different virtual hosts (Back Office and Front Office) in your Apache configuration.

Virtual Host

Back Office Virtual Host:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName admin.openorchestra.1-1.dev

    DocumentRoot  /path/to/your/open_orchestra/bo_installation/folder/web
    <Directory />
        Options FollowSymLinks
        AllowOverride None
```

```
    </Directory>
    <Directory /path/to/your/open_orchestra/bo_installation/folder/web>
        Options -Indexes +FollowSymLinks -MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
    </Directory>

    <IfModule mod_rewrite.c>
        RewriteEngine On
    </IfModule>

    ErrorLog /var/log/apache2/admin.openorchestraError.log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog /var/log/apache2/admin.openorchestraAccess.log combined
</VirtualHost>
```

Front Office Virtual Host:

```
<VirtualHost *:80>
    ServerAdmin webmaster\@localhost
    ServerName front.openorchestra.1-1.dev

    DocumentRoot /path/to/your/open_orchestra/front_installation/folder/web
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /path/to/your/open_orchestra/front_installation/folder/web>
        Options -Indexes +FollowSymLinks -MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
    </Directory>

    <IfModule mod_rewrite.c>
        RewriteEngine On
    </IfModule>

    ErrorLog /var/log/apache2/front.openorchestraError.log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog /var/log/apache2/front.openorchestraAccess.log combined
</VirtualHost>
```

1.65.2 Varnish vcl

With Open Orchestra, you can use `Varnish`. It is particularly useful for ESI rendering for instance.

This vcl is a basic configuration example to use the esi render.

```
acl purgers {
  "127.0.0.1";
}

acl invalidators {
  "127.0.0.1";
}

backend f1 {
  .host = "127.0.1.1";
  .port = "80";
}

director front round-robin {
    {
        .backend = f1;
    }
}

sub vcl_recv {
    set req.backend = front;

    if (req.http.Cache-Control ~ "no-cache" && client.ip ~ invalidators) {
        set req.hash_always_miss = true;
    }

    if (req.request == "BAN") {
        if (!client.ip ~ invalidators) {
             error 405 "Ban not allowed";
        }

        if (req.http.X-Cache-Tags) {
            ban("obj.http.X-Host ~ " + req.http.X-Host
                + " && obj.http.X-Url ~ " + req.http.X-Url
                + " && obj.http.content-type ~ " + req.http.X-Content-Type
                + " && obj.http.X-Cache-Tags ~ " + req.http.X-Cache-Tags
            );
        } else {
          ban("obj.http.X-Host ~ " + req.http.X-Host
              + " && obj.http.X-Url ~ " + req.http.X-Url
              + " && obj.http.content-type ~ " + req.http.X-Content-Type
          );
        }

        error 200 "Ban added";
    }

    if (req.request == "PURGE") {
        if (!client.ip ~ purgers) {
             error 405 "Purge not allowed";
        }
        return(lookup);
     }

    if (req.http.Accept-Encoding) {
        if (req.url ~ "\.(jpg|png|gif|gz|tgz|bz2|tbz|mp3|ogg)$") {
            remove req.http.Accept-Encoding;
        } elsif (req.http.Accept-Encoding ~ "gzip") {
```

```
            set req.http.Accept-Encoding = "gzip";
        } elsif (req.http.Accept-Encoding ~ "deflate") {
            set req.http.Accept-Encoding = "deflate";
        } else {
            remove req.http.Accept-Encoding;
        }
    }

    if (req.http.User-Agent ~ "(?i)android") {
        set req.http.X-UA-Device = "android";
    }

    if(req.http.host ~ "(admin.openorchestra.1-1.dev)") {
        return (pass);
    }

    if (req.request == "POST") {
        return(pass);
    }

    if (req.url ~ "^/preview") {
        return (pass);
    }

    set req.http.Surrogate-Capability = "varnish=ESI/1.0";

    return(lookup);
}

sub vcl_fetch {
    set beresp.http.X-Url = req.url;
    set beresp.http.X-Host = req.http.host;

    if (beresp.status == 404 || beresp.status == 500 || beresp.status == 503) {
        set beresp.ttl = 30s;
    }

    if (beresp.http.Surrogate-Control ~ "ESI/1.0") {
        unset beresp.http.Surrogate-Control;
        set beresp.do_esi = true;
    }

    if (beresp.ttl > 0s) {
      unset beresp.http.Set-Cookie;
    }
}

sub vcl_deliver {
    if (!resp.http.X-Cache-Debug) {
        unset resp.http.X-Url;
        unset resp.http.X-Host;
        unset resp.http.X-Cache-Tags;
    }
}

sub vcl_hit {
    if (req.request == "PURGE") {
        purge;
```

```
        error 204 "Purged";
    }
}

sub vcl_miss {
    if (req.request == "PURGE") {
        purge;
        error 204 "Purged (Not in cache)";
    }
}

sub vcl_hash {
    if (req.http.X-UA-Device) {
        hash_data(req.http.X-UA-Device);
    }
}
```

1.65.3 Cron jobs

Cron jobs are used for tasks (commands or shell scripts) to run periodically at fixed times, dates, or intervals. Cron jobs typically automate system maintenance.

Cron jobs on Open Orchestra

Open Orchestra has 4 cron jobs created with the provisioning:

Site maps

Generate sitemap files for every sites, more information available in the sitemap documentation

```
0 2 * * * php /var/www/front-open-orchestra/current/app/console -e=prod orchestra:sitemaps:generate 2
```

Robots.txt

Generate the robots.txt files for every sites, further information about robots

```
0 2 * * * php /var/www/front-open-orchestra/current/app/console -e=prod orchestra:robots:generate 2>>
```

Error pages

Generate the special error pages files for every sites (eg 404 & 503 status), for more information about the 404 and 503 special pages see the documentation customizing error pages

```
0 2 * * * php /var/www/front-open-orchestra/current/app/console -e=prod orchestra:errorpages:generate
```

Error cron

This cron sends an email if any of above cron didn't correctly.

```
59 0-23 * * * if [ -s '/tmp/cron.error.message' ]; then  cat /tmp/cron.error.message | mailx -s "cron
```

1.65.4 Ansible

If you don't want set the different configurations (Virtual Host, Varnish, Cron) manually, you can use the provisioning.

1.66 Orchestra deployment on integration environment

This tutorial is going to describe step by step how you can deploy the Open Orchestra project on the integration environment.

To be able to deploy with capistrano, your target environment should have the requirements :

- git installed
- composer installed and executable as the `composer` command
- The user used to deploy and the one used to execute the php should be able to write in the cache and log directory at all time.

1.66.1 Install Gem and Bundler

In order to separate all the project of your computer, we use bundler to execute only the specified version of capistrano.

```
$ aptitude install ruby ruby-dev
$ gem install bundler
```

1.66.2 Install capifony

In the root directory of your project, run the command

```
$ bundle install
```

1.66.3 SSH configuration

Make sure that your ssh configuration is ready and your personal public key is installed on the integration server

```
# ~/.ssh/config
Host open_orchestra_bo_inte_1-1
    Hostname 10.0.1.228
    User open_orchestra_bo_inte_1-1
    ForwardAgent yes
Host open_orchestra_front_inte_1-1
    Hostname 10.0.1.228
    User open_orchestra_front_inte_1-1
    ForwardAgent yes
Host open_orchestra_front2_inte_1-1
    Hostname 10.0.1.228
    User open_orchestra_front2_inte_1-1
    ForwardAgent yes
Host open_orchestra_media_inte_1-1
    Hostname 10.0.1.228
    User open_orchestra_media_inte_1-1
    ForwardAgent yes
```

1.66.4 Test the command

Run the following command to test the installation and connection to the server

```
$ cap 1.1 deploy:log_revision
```

1.66.5 Deploy

Before deploying the project, make sure that the `composer.lock` file is up-to-date on the master branch.

```
$ cap inte composer:update_vendor
```

This task run `composer update vendor` on the server to update `composer.lock` and create a branch on GitHub.

Once you are ready (passing tests for instance), you can merge in the branch you want to deploy

Then you can run the deploy command in the project directory

```
$ cap 1.1 deploy
```

Once this is done, your server has been updated.

1.67 Server provisioning

1.67.1 Context

At some point in your project, you will have to deploy your Open Orchestra application across multiple servers and environments.

Whenever you do it, by following this documentation you will be able to use the same provisioning for all your environments (staging, production, ...).

1.67.2 Host configuration

In your project, we recommend you to create a `provisioning/hosts` folder. Inside, you will put a host file for each environment to provision, for instance `prod` for the production environment.

The `prod` file will contain the group server with the ssh hosts :

```
[prod]
open_orchestra_prod
```

In this case, the prod group will only contain one server, but you can add more if needed

1.67.3 Environment configuration

In the `provisioning/hosts` folder, you need to create a `group_vars` folder which will contain all the specific information for each environment. (The files should be named as in the hosts folder)

Server configuration

- `user_root` : The user with the root privileges
- `hosts_localhost` : The `/etc/hosts` configuration
- `hosts_site` : Additional `hosts` configuration linked to the deployed sites
- `sudoers` : Users used to deploy, they should be able to restart `Apache2` and `Varnish`
- `selenium_current_directory` : The folder where the provisioning will download `selenium`

Apache configuration

- `apache_main_ports` : All the ports `Apache2` is listening to
- `apache_conf` : One entry for each website managed

The `Apache2` vhost configuration will require some parameters :

- `key` : the key will be used as the vhost filename
- `port` : On which port the vhost will be listening
- `serverName` : The vhost server name
- `docRoot` : The web folder of your Open Orchestra installation
- `errorLog` : The filename with the error logs of your vhost
- `accessLog` : The filename with the access logs of your vhost
- `siteId` : The site id of the front website you are deploying (only needed on front installation)

The value can be found in the edit form in the Back Office

Example :

```
apache_conf:
    demo-orchestra.conf:
        port: 8000
        serverName: demo.openorchestra.1-1.dev
        docRoot: /var/www/front-open-orchestra/web
        errorLog: demo-openorchestraError.log
        accessLog: demo-openorchestraAccess.log
        siteId: 2
```

Varnish configuration

- `varnish_listen_port` : The port `varnish` is listening to
- `backend_conf` : All the vhost redirected by `varnish`

The `backend_conf` will require some parameters :

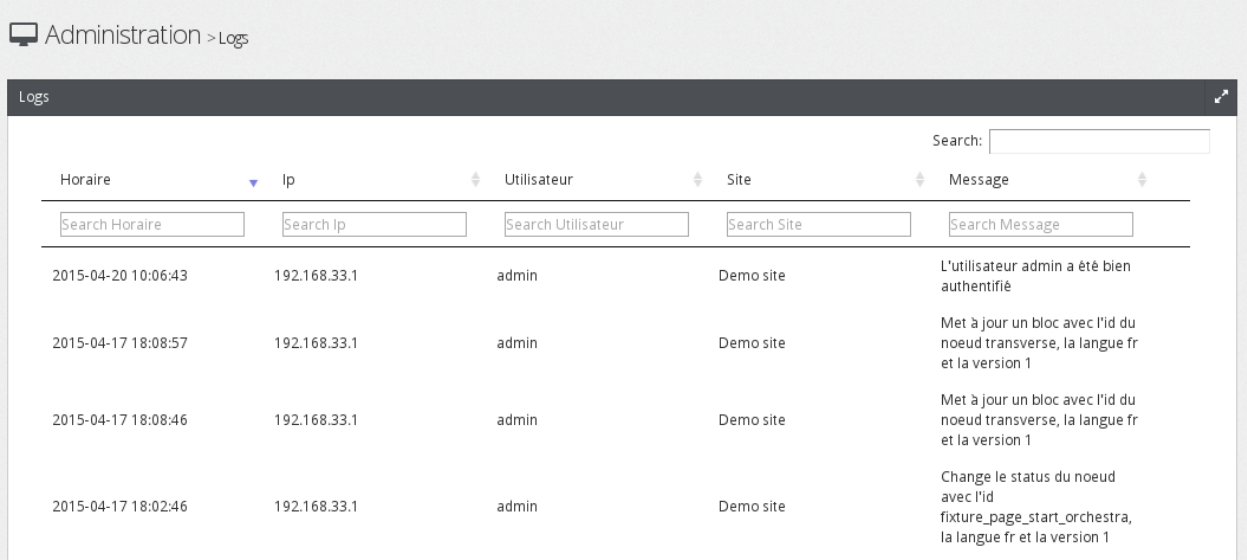
- `name` : The redirection name
- `port` : The redirection port
- `host` : The redirection host
- `admin` : Set to true if the backend is an Back Office installation

Crontab configuration

- `path_front` : The folder of the front installation
- `mail_to` : The webmail where the cron error are send

1.68 Logs

Our Back Office logging is based on Monolog. Open Orchestra provides an interface to display all the user actions on the website.



1.68.1 Configuration

Monolog uses channels to record the log. To link a channel and a record, you will have to define an handler in a configuration file.

For instance, if you want to store the logs in mongo, you will have to define a handler with the database connection parameters. The level entry indicates the information type to record in the logs. Each level is a number which value is standardized in the PSR-3 recommendation. For instance 200 is the value for the "info" level. The different levels are defined in the Monolog documentation

```
mongo:
    type: mongo
    level: 200
    mongo:
        host: "localhost"
        database: "%open_orchestra_cms.mongodb.database%"
        collection: log
    channels: [openorchestra]
```

The `channels` key defines the channels where the log are published. If no channel is specified then the logs are written in all channels. It's possible to put multiple channels or exclude one. The services which have the channel `openorchestra` write only in this chanel and our handler shows logs which are in the `openorchestra` channel.

The different possibilities of configuration about channels are explained in the channels documentation.

1.68.2 Processor

When you publish your log, some information may be missing. Monolog allows you to add a processor which will fill up the logging information by adding some context.

To do it, you can create a service, tag it with `monolog.processor`. You can refer to the Monolog processor documentation for further example.

1.68.3 Logger

The services that write the logs are event subscribers. They should be injected the `Symfony\Bridge\Monolog\Logger` in order to be functional and be tagged with `monolog.logger`.

For instance:

```
open_orchestra_log.subscriber.node:
    class: "%open_orchestra_log.subscriber.node.class%"
    arguments: [\@logger]
    tags:
        - { name: kernel.event_subscriber }
        - { name: monolog.logger, channel: openorchestra }
```

In Open Orchestra an event is dispatched on most user actions (ie. creation, update of elements...). This event is caught to publish the log message in the right channel. A log message takes two parameters, a translation key and a context (data array about the message). The messages are located in the translation file, for instance:

```
open_orchestra_log:
    node:
        update: Update a node with node id node_id, node version node_version and node language node_
```

In this example, the variables (node_id, node_version, node_language) are defined in the context, they are replaced on display.

The different subscribers are in `OpenOrchestra\LogBundle\EventSubscriber`.

1.69 Migration

Open Orchestra offers migration tasks to facilitate the update version. To know more about migration see mongodb migration.

1.69.1 MongoDB Migration

The MongoDBMigrationsBundle come with a set of command available under the name space mongodb:migrations. These commands will respond according to the settings.

MongoDb Migration settings

To use migration tasks of Open Orchestra, you must configure the bundle with this settings:

```
mongo_db_migrations:
    collection_name: 'migration_versions'
    database_name: '%open_orchestra_cms.mongodb.database%'
    dir_name: '%kernel.root_dir%/../vendor/open-orchestra/open-orchestra-model-bundle/OpenOrchestra/M
    script_dir_name: '%kernel.root_dir%/../vendor/open-orchestra/open-orchestra-model-bundle/OpenOrch
```

```
    name: 'Open Orchestra MongoDB Migrations'
    namespace: 'OpenOrchestra\ModelBundle\Migrations\MongoDB'
```

MongoDb Migration migrate

The command

```
app/console mongodb:migrations:migrate
```

will execute the different steps of migration.

1.70 open-orchestra-base-api-bundle

| Service | Badge | | ————— |:————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.71 open-orchestra-base-bundle

| Service | Badge | | ————— |:————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |

1.72 open-orchestra-cms-bundle

| Service | Badge | | ————— |:————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.73 open-orchestra-display-bundle

| Service | Badge | | ————— |:————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |

1.74 open-orchestra-front-bundle

| Service | Badge | | ————— |:————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |

1.75 open-orchestra-media-bundle

| Service | Badge | | ————— |:————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.76 open-orchestra-model-bundle

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.77 open-orchestra-model-interface

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sension Insight | | | VersionEye | |

1.78 open-orchestra-theme-bundle

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |

1.79 open-orchestra-user-bundle

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |

1.80 open-orchestra-workflow-bundle

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |

1.81 open-orchestra-libs

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.82 open-orchestra-media-admin-bundle

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.83 open-orchestra-base-api-mongo-model-bundle

| Service | Badge | | ——————- |:——————-:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.84 open-orchestra-newsletter-bundle

| Service | Badge | | ——————- |:——————:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sension Insight | | | VersionEye | |

1.85 open-orchestra-mongo-libs

| Service | Badge | | ——————- |:——————:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio insight | | | VersionEye | |

1.86 open-orchestra-bbcode-bundle

| Service | Badge | | ——————- |:——————:| | Travis | | | CodeClimate (quality) | | | CodeClimate (coverage) | | | Sensio Insight | | | VersionEye | |