# Briques technologiques

## Technologies de développement et protocoles d'interopérabilité

Tous les composants développés dans ORI-OAI reposent sur les mêmes technologies:

**Java**

> *Comme langage de programmation*
> *http://www.java.com*

**Spring**

> *Spring est un framework de développement de haut niveau dans la mesure où il environnemente et dirige l'architecture générale de l'ensemble de l'application. On notera quelques carctéristiques de Spring qui nous ont fait choisir Spring comme Framework principal de l'application.*
>
> *Il est le choix de toute la communauté ESUP pour les présents et futurs développements. Il est pensé pour intégrer directement un certain nombre d'autres frameworks comme hibernate (mais aussi OSWorkflow, Compass/Lucène, ...). Il rend les applications souples et paramétrables. Il permet de séparer les tâches de développement via un développement par couche. Il permet d'implémenter des architectures de type Objet modélisable usuellement via UML. Il propose de tirer parti de la programmation par aspect pour la gestion des transactions de BD, via des modules très sophistiqués comme EhCache pour le cache, Acegi pour la sécurité (autentification et autorisation), et enfin directement en insérant du code métier supplémentaire (pour réaliser un outil de statistiques par exemple ...).*
> *http://www.springframework.org*

**XML**

> *Comme norme d'échange*
> *http://www.w3.org/XML*

**Subversion**

> *Pour le téléchargement des sources des modules*
> *http://subversion.tigris.org/*
> *http://fr.wikipedia.org/wiki/Subversion_(logiciel)*

Certains composants utilisent des technologies spécifiques:

## ORI-OAI-workflow

Ce paragraphe définit les choix techniques pris après réflexions, tests, etc. Ces choix ont été fonction des technos, de leur efficacité par rapport aux besoins, de leur souplesse mais aussi de leur robustesse, de l'expérience de chacun, des habitudes et technos usités dans la communauté ESUP Portail, etc.

**OsWorkflow**

> *OSWorkflow est un produit OpenSource d'OpenSymphony.*
>
> *OSWorkflow est une bonne solution de Workflow dans le monde de l'OpenSource. On constate qu'il est utilisé dans différents produits et qu'il est choisi par des projets +/- similaires à ORI.*
>
> *Une de ses principales spécificités est de fonctionner de manière autonome. La servlet d'exemple fournie par défaut et qui tourne sans aucune configuration le montre bien, OSWorkflow peut fonctionner seul. Il est aisé de pluguer OSWorkflow à une application déjà existante. OSWorkflow peut ainsi être vu comme un composant réellement indépendant dans ORI-OAI-Workflow. La possibilité qu'il offre de le commander via de simples appels WEBService (SOAP) le prouve également.*
>
> *OSWorkflow gère de manière autonome tout ce qui concerne le Workflow. Les Workflows (diagrammes états-transitions) se configurent via un fichier XML. Une Application Graphique Java peut permettre de créer/modifier des Workflows (attention : cela reste cependant avant tout expérimental). Les Workflows peuvent être relativement complexes. Ils permettent d'appeller des scripts Java BeanShells lors d'une transition (ou mieux encore des méthodes définies dans des "beans spring", et c'est ce qui nous intéresse ici), de faire des splits/joins, de mettre des conditions de tous types sur des transitions (par exemple des conditions sur l'appartenance d'un utilisateur à un rôle), de définir des permissions en fonction des états, etc.*
>
> *Indépendant, OSWorkflow peut s'utiliser avec sa propre base de données (en tout cas, il gère ses propres tables). L'édition des workflows se fait de manière indépendante du reste de l'application, en éditant le fichier XML OSWorkflow avec un simple éditeur texte. L'indépendance de OsWorkflow ne gêne pas son intégration complète dans le module ORI-OAI-Workflow : on utilise OsWorkflow conjointement à spring, ce qui permet d'utiliser des méthodes spécifiques au module ORI-OAI-Workflow en tant que fonctions et conditions dans des workflows OsWorkflow. Aussi les possibilités du module ORI-OAI-Workflow en matière de conditions ou de fonctions à appeler lors de transitions sont facilement extensibles* et cela fait partie des gros points forts de ce module.
>
> *http://www.opensymphony.com/osworkflow*

**Esup-Commons**

> *Esup-Commons permet de mettre en place (et de façon standard par rapport aux développements Esup) une architecture basée sur Spring-Hibernate-JSF (le framework général de l'application étant Spring : conteneur léger), packagée au mieux, pensée pour les mises à jours futures, pouvant tourner à la fois (avec le même code) en mode portlet et en mode standalone.*
> *Esup Commons est, au vue d'une application comme Ori-Oai-Workflow, une proposition d'architecture représentant des facilités de développement et déploiement. Nous utilisons dans Ori-Oai-Workflow certaines fonctionnalités de Esup Commons.*
>
> *Ainsi la gestion des exceptions et l'utilisation de Hibernate sont gérées via Esup Commons, tout comme les envois de message via SMTP. Par contre l'authentification est par exemple gérée par Acegy Security et non Esup Commons.*
> *http://www.esup-portail.org/display/PROJCOMMONS*

**JSF**

> *Bien que l'accent concernant l'ergonomie et l'IHM soit d'abord mis sur les formulaires d'édition, on souhaite avoir une IHM efficace tout le long de l'application. JSF se couple parfaitement avec Spring.*
> *L'implémentation choisi de JSF est MyFaces qui semble être l'implémentation OpenSource de référence. On utilise cependant d'autres librairies JSF comme Jenia par exemple.*
>
> *L'utilisation de JSF via Esup Commons nous permet d'avoir un code unique pour les 2 versions d''Ori-Oai-Workflow : mode portlet et mode standalone.*
> *http://java.sun.com/javaee/javaserverfaces/*
> *http://myfaces.apache.org/*
> *http://www.jenia.org/*

**Base de données SQL**

> *Le choix a été fait d'utiliser une Base de Données SQL transactionnelle unique pour l'ensemble de l'application. Pour manipuler la base de données depuis l'application, le choix s'est porté sur Hibernate qui s'intègre bien dans Spring. Il est simple à configurer et à utiliser notamment lorsqu'il n'y a pas de BD pré-existante (c'est à dire lorsque la structuration de la base de données est conçue pour les besoins de l'application développée, comme c'est le cas ici).*

**Hibernate**

> *Framework de mapping objet-relationnel (MySql)*
> *http://www.hibernate.org/*
> *http://www-fr.mysql.com/*

**Acegi Security**

> *Acegi Security est la solution en terme de sécurité intégrée dans une application Spring.*
>
> *Acegi Security est un produit complexe qui couvre un grand nombre d'aspects autour de la sécurité : "authentication " + "authorisation". Il permet surtout de sécuriser en fonction des rôles de l'utilisateur les appels à des méthodes, les urls.*
>
> *ORI-OAI-Workflow utilise Acegi Security pour :*
>
> - *la gestion de l'authentification.,*
> - *la gestion de RBAC (Role Based Access Control) qui est le Design Pattern décrivant le fait que l'on assigne les permissions à des rôles et non directement aux "Principals" (~ Groupes et Utilisateurs), cf la*
> - *et donc la gestion des autorisations.*
>
> *Ainsi Acegi a été utilisé comme base pour implémenter une solution de RBAC.*
> *http://www.acegisecurity.org/*

**XFire**

> *Pour la communication entre Spring et Orbeon OPS, ainsi que la communication inter-modules*
> *http://xfire.codehaus.org*

**Conteneur - uPortal**

> *ORI-OAI-Workflow peut fonctionner dans un ESUP-Portail. Il est décliné sous forme de JSR168. ORI-OAI-Workflow peut également fonctionner sans ESUP, il fonctionne alors de manière "standalone" en tant que servlet.*

**Ldap - CAS - Shibboleth**

> *Dans ORI-OAI-Workflow, la gestion d'utilisateurs/groupes peut se faire via un ldap (et donc via un openldap/phpldapadmin local par exemple si le système d'information dans lequel ORI-OAI-Workflow se déploie n'a pas de système de gestion de son ldap) ou via shibboleth.*
> *La récupération des utilisateurs et des groupes par ORI-OAI-Workflow sur le ldap ou via shibboleth est à configurer dans des fichiers de configuration. Il est à noter que ORI-OAI-Workflow est capable de définir des nouveaux groupes "virtuels" via des filtres Ldap configurées dans le module ORI-OAI-Workflow (ce qui peut s'avérer pratique).*
> *L'authentification peut se faire via LDAP, CAS ou Shibboleth.*

**Compass**

> *http://www.compass-project.org/*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

ORI-OAI-Workflow est géré par le framework Spring et répond en JSR168 (Portlet) ou mode standalone (Servlet). Il contient toute la logique applicative. Il fait appel à OSWorkflow pour afficher les informations relatives aux états dans lesquels se trouvent les workflows et pour procéder à des actions sur les états (transitions). Il propose ainsi à l'utilisateur certaines actions sur les entrées de MétaDonnées qu'il liste : faire une transition sur le workflow correspondant à l'entrée, éditer la fiche XML de l'entrée (et donc appeler un formulaire Orbeon Forms), etc. Il intègre également complètement OSWorkflow.
OSWorkflow gère tout ce qui est relatif au Workflow a proprement parlé. il contient et propose les informations relatives aux états des workflows, les transitions possibles, les permissions que l'utilisateur courant a par rapport au rôle dans lequel il est, etc. Il appelle la partie principale lors des conditions et fonctions paramétrées dans les workflow.

# ORI-OAI-md-editor

**XForms/Orbeon OPS**

> *Les formulaires d'édition proposés à l'utilisateur dans ORI-OAI-Workflow correspondent à une partie très importante de ORI-OAI-Workflow dans le sens où c'est la partie la plus visible de ORI-OAI-Workflow pour l'utilisateur final. Ces formulaires peuvent se décliner selon les différents schémas (DC, LOM, LOM-FR, ...), selon les différents rôles des différents utilisateurs. Très souples, ils acceptent un certain nombre de paramètres qui diffèrent selon les usages. On pense notamment aux taxonomies/vocabulaires utilisés lors de la saisie via des widgets à choix multiple. Enfin ils sont ergonomiques et permettent une saisie efficace des fiches XML de métadonnées : auto-complétion, ajout/suppresion instantané de champs, bref les forumlaires sont des formulaires dynamiques (Ajax).*
>
> *Le choix a été d'opter pour un standard W3C XForms qui permet de créer des formulaires ergonomiques, dynamiques, .... cela via un langage XML. Vu que XForms n'est pas supporté nativement par les navigateurs et que les implémentations de plugins sur les navigateurs ne sont pas encore tout à fait au point, et enfin pour des questions de souplesse, la solution proposée par Orbeon, Orbeon Forms a été retenue. Celle-ci permet entre autre de générer à partir de formulaires XForms une interface Html/Ajax interprétable par les principaux navigateurs clients du marché.*
>
> *Orbeon est à l'image de Cocoon un framework complet de développement qui tire parti, de manière extrêmement ingénieuse, de la technique de transformation XML pour concevoir des applications complexes de manière efficace. On a choisi de retenir Orbeon Forms pour la saisie des métadonnées et donc pour sa qualité première et spécifique de technologie de formulaires.*
> *http://www.orbeon.com/*
> *http://fr.wikipedia.org/wiki/Asynchronous_JavaScript_And_XML*
> *http://www.w3.org/MarkUp/Forms/*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

Le module ORI-OAI-MD-Editor correspond à un XForms interprété par Orbeon Forms pour générer une interface Ajax à l'utilisateur. Il ne contient pas de logique applicative. Il recoit un XML et en renvoie un. Pour ce faire un protocole de communication spécifique (s'appuyant sur des Web Services SOAP) est cependant mis au point pour permettre à Orbeon et ori-oai-workflow (spring) d'interagir ensemble.

# ORI-OAI-harvester

**OAI-PMH**

> *Protocole d'échange des fiches de métadonnées*
> *http://www.openarchives.org/OAI/openarchivesprotocol.html*

**OCLC OAIHarvester2**

> *Cette application est basée sur le projet OAIHarvester2 d'OCLC Online Computer Library Center, qui fournit les fonctions de requêtes OAI-PMH.*
> *Notes :*
>
> - *limites et adaptation de l'API OAICat :*
>   - *org.oclc.oai.server.verb.ListRecords : Les champs from et until sont validés pour une longueur entre 0 et 10, ce qui force à utiliser une granularité YYYY-MM-DD.*
>     *http://www.oclc.org/research/software/oai/harvester2.htm*

**Quartz**

> *La gestion des tâches programmées est réalisée grâce au projet open source Quartz d'OpenSymphony.*
> *http://www.opensymphony.com/quartz/*

**Struts-Tiles**

> *Pour les interfaces graphiques*
> *http://struts.apache.org/1.x/struts-tiles/*

**Codehaus XFire**

> *Pour l'interaction entre les modules, des Web Services sont mis en oeuvre à l'aide du framework Codehaus XFire.*
> *http://xfire.codehaus.org/*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

# ORI-OAI-repository

**OAI-PMH**

> *Protocole d'échange des fiches de métadonnées*
> *http://www.openarchives.org/OAI/openarchivesprotocol.html*

**OCLC OAICAT**

> *Le module utilise l'API OAICat d'OCLC, qui fournit une servlet répondant aux six verbes OAI. Cet API permet de constituer des enregistrements OAI depuis différents types de resources : JDBC, XML, etc... Elle offre un Framework qui aide également à gérer le controle de flux par resumptionToken.*
> *http://www.oclc.org/research/software/oai/cat.htm*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

# ORI-OAI-indexing

**Lucene**

> *Lucene est un moteur de recherche appartenant à la fondation apache permettant l'indexation et la recherche de texte. Il est entièrement écrit en langage Java. La version utilisée dans cette version du module d'indexation est la 2.3.2.*
> *http://lucene.apache.org/java/docs/index.html*

**Solr**

> *Solr est une plateforme logicielle de recherche s'appuyant sur le moteur de recherche Lucene, créé par la Fondation Apache et distribuée et conçue sous licence libre.*
> *Solr utilise le langage Java et est exécuté par un conteneur de servlets, comme par exemple Tomcat. Il communique avec le client à l'aide d'une interface de programmation en XML et JSON, généralement via le protocole HTTP.*
>
> *Toute la configuration de l'indexation, telle que le type de fichiers à indexer ou encore les champs par exemple, ainsi que la recherche sont définies dans un fichier XML, il ne reste plus qu'à écrire le code pour exécuter l'indexation ou la recherche.*
> *http://lucene.apache.org/solr/*

**Luke**

> *Luke est une interface graphique permettant de visualiser un index. Il peut être utile en tant qu'outil de diagnostic de ce dernier.*
> *http://www.getopt.org/luke/*

**Ehcache**

> *Il s'agit d'un gestionnaire de cache en Java. Il est capable de stocker des données en mémoire vive ou sur le disque. Ehcache est utlisé dans le cadre du projet ORI-OAI-Indexing en ce qui concerne la gestion des différents caches utiles à l'optimisation de la recherche dans l'index. La version de la librairie utilisée dans le module d'indexation est la 1.3.*
> *http://ehcache.sourceforge.net/*

**Quartz**

> *Cette application créée par OpenSymphony est utilisée dans le cadre d'Ori-Oai-Indexing pour la tâche planifiée de gestion des liens morts qui se déroule généralement la nuit. Quartz permet de créer des tâches planifiées très simples ou plus complexes.*
> *http://www.opensymphony.com/quartz/*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

# ORI-OAI-search

**Spring MVC**

> *Utilisé pour la présentation*
> *http://www.springframework.org*

**JQuery**

> *Utilisé pour l'affichage dynamique*
> *http://jquery.com/*

**Lucene**

> *Pour le format de requêtes*
> *http://lucene.apache.org/java/docs/index.html*

**XSLT**

> *Pour la présentation des fiches de métadonnées*
> *http://www.w3.org/TR/xslt*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

# ORI-OAI-vocabulary

**VDEX**

> *Comme norme utilisé pour la gestion des vocabulaires et l'échange des vocabulaires entre modules*
> *http://www.imsglobal.org/vdex/index.html*

**Esup-Commons**

> *Framework principal pour Ori-Oai-Vocabulary (ce qui induit différents choix technologiques comme Spring bien sûr, mais aussi XFire, EhCache, etc.)*
> *http://www.esup-portail.org/display/PROJCOMMONS*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*

# ORI-OAI-nuxeo

**Nuxeo DM**

> *Ce composant est intégré à la solution Open Source d'ECM (Enterprise Content Management) Nuxeo DM*
> *http://www.nuxeo.com/*

**Technologies communes**

> *Ainsi que toutes les technologies communes à tous les modules.*