Version 1.5

ORI-OAI-workflow : Gestion du workflow de saisie

> ⚠ **Module optionnel**

Voir l'architecture du système

Dans quels cas l'utiliser

Pour mettre en place le workflow du référencement des ressources

Composants obligatoires

- **ORI-OAI-indexing** pour indexer les fiches de métadonnées saisies
- **ORI-OAI-md-editor** pour l'édition des fiches de métadonnées
- **Base de données SQL** pour le stockage des fiches de métadonnées et des données de gestion
- Un **annuaire LDAP** pour la gestion des utilisateurs

Composants optionnels

- **Esup-ECM avec ORI-OAI-nuxeo** si vous voulez utiliser ce serveur pour y déposer les ressources
- Un **serveur SSO CAS** pour l'authentification (authentification LDAP si non utilisé)
- Un **IDP Shibboleth** pour l'authentification et l'identification en lieu et place de CAS/LDAP.

Description

Ce composant est utilisé pour le workflow de référencement des ressources. Les technologies utilisées par ce moteur de publication sont diverses et permettent un paramétrage très fin, complet et puissant en fonction des besoins de l'établissement ou du consortium qui le met en place.

Depuis la version 1.5 il est couplé avec l'outil de gestion des documents ESUP-ECM.

Le moteur de workflow OsWorkflow permet une configuration avancée de toutes les actions effectuées à chacune des étapes de la publication. En effet, par de la configuration, il peut répondre aux besoins des établissements exigeant du déposant la saisie de tous les champs de métadonnées, mais aussi des établissements qui souhaitent une gestion du dépôt plus structurée où la saisie des métadonnées s'effectue en différentes étapes, par différents intervenants. Le formulaire de saisie des métadonnées est supporté par un appel à ORI-OAI-md-editor aux différentes étapes du workflow. Chaque utilisateur aura alors, pour une même fiche, une vue différente. Ceci lui permettra de voir et d'éditer uniquement les métadonnées sur lesquelles il a un droit d'accès particulier.

Les autres technologies utilisées dans ce module sont Acegi Security pour gérer toutes les sécurités d'accès à l'application, une base de données SQL pour le stockage des fiches de métadonnées et les données de gestion, Hibernate pour rendre les accès à la base de données relationnelle transparents, Compass/Lucène pour réaliser des requêtes rapides et efficaces sur les fiches de métadonnées associées aux données de gestion et enfin JSF pour la gestion des interfaces graphiques.

Voir la documentation technique

Spécifications

Choix techniques

Choix techniques

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, etc.

Workflow - 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 fourni 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*.

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.
Dans les premières versions, c'est le mode servlet/standalone qui est recommandé.

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.

Framework de développement - 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 ...).

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, voir plus bas) nous permet d'avoir un code unique pour les 2 versions d''Ori-Oai-Workflow : mode portlet et mode standalone.

BD SQL

Le choix a finalement 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).

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 "Pricipals" (~ Groupes et Utilisateurs), cf la
- et donc la gestion des autorisations.
  Ainsi Acegi a été utilisé comme base pour implémenter une solution de RBAC.

Esup Commons

Esup-Portail 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.

Autres ...

D'autres technologies sont utilisées comme XFire pour l'implémentation du protocole SOAP, Lucène/Compass, SpringModules pour l'intégration d'autres technologies encore: par exemple EHCache.

Conclusion

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.

Spécifications du module

Spécifications

Workflow (OSWorkflow)

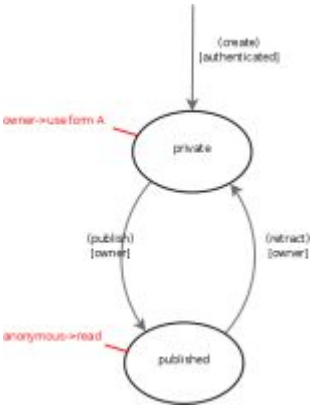
Un Workflow est un concept large. Même si l'on peut être d'accord sur le concept, il peut être décrit de différentes manières, sous différents points de vue (états/transitions, ...) et avec des vocabulaires différents.

Ce paragraphe a donc pour objectif de donner une "vision ORI-OAI" de ce qu'est un workflow. Concrètement, la spécification d'un workflow ORI-OAI correspond au workflow tel qu'il est présenté dans OSWorkflow mais en utilisant en plus des beans spring ORI-OAI comme méthodes à appeler lors des conditions et fonctions du workflow.

Workflow - Diagramme d'état transtions

Un Workflow est défini par son diagramme d'états/transtions. OSWorkflow permet de définir des Workflows par des fichiers XML. Ainsi on définit un Workflow pour la gestion des fiches des ressources pédagogiques, un autre pour la gestion des thèses, etc.

Une fois un Workflow défini, les applications tierces à OSWorkflow peuvent instancier des instances de workflows pour chacune des ressources souhaitées et les manipuler.



Ici est défini un workflow très simple à 2 états : private et publish. On peut passer d'un état à un autre via une transition possible si la condition est respectée. Lors de cette transition, le workflow peut demander à appeler des fonctions.

Dans Ori-Oai-Workflow, un Workflow correspond très concrètement à un fichier de configuration d'un workflow OsWorkflow (implémentant le diagramme ci-dessus par exemple).

Eléments constitutifs d'un Workflow

instance de workflow

> *Dans Ori-Oai-Workflow, à une entrée XML (~ fiche de Métadonnée XML) correspond une instance de workflow d'un Workflow (diagramme états/transitions) donné. L'instance de workflow permet la gestion du "cycle de vie" de l'entrée XML. L'instance de workflow garde en mémoire les transitions/états passés et présent de l'entrée XML. Elle est consultée pour récupérer des informations comme les transitions disponibles, l'état actuel, les permissions. Elle est aussi appelée pour demander à effectuer les transitions.*
> *Elle correspond très concrètement à "une" entrée dans la base de données.*

Rôles

> *Les utilisateurs ont des rôles sur des ressources. Typiquement, un utilisateur aura le rôle de "owner" sur les ressources qu'il a lui-même créées. Les rôles peuvent intervenir dans les conditions : "à condition que l'utilisateur courant soit owner". Ces rôles correspondent en fait concrètement à des entiers qui représentent dans l'application générale les identifications des rôles.*

Permissions

> *Des permissions peuvent êtres positionnées en fonction de l'état (voir des états) dans lequel se trouve l'instance de workflow. Ces permissions correspondent en fait concrètement à des entiers qui représentent dans l'application générale les identifications des permissions.*

Etats-Transitions

> *Le Workflow est défini par un diagramme d'états/transitions. Une instance de workflow est caractérisée par l'état (voir les états dans le cas de split) dans lequel elle se trouve. Depuis un état, il est possible d'effectuer ou non des transitions. Ces transitions ammènent l'instance dans un autre état.*
> *Dans OSWorkflow, on parle de "step" (pour état étape) et de "action" (pour transition).*

fonctions - conditions

> *Les fonctions, tout comme les conditions, peuvent être des java beanshell (scripts en java) ou des classes (méthodes de classes) ou encore, et c'est ce que le déployeur/dévelopeur utilisera certainement la plupart du temps, des beans Spring. Ils peuvent admettre des arguments.*
> *Une fonction permet de lancer un script/méthode après ou avant une transition. On peut par exemple y implémenter l'envoi de mail au owner de l'instance du workflow, une demande d'indexation au module Ori-Oai-Indexing ou plus usuellement encore, le positionnement de permissions et/ou de rôles.*
> *Une condition permet de conditionner une transition via un script/méthode. On peut par exemple conditionner une transition sur l'appartenance de l'utilisateur à un rôle donné.*

Mise en oeuvre

Comme dit ci-avant, on utilise OSWorkflow comme un sous-module indépendant des autres composants. Aussi, on propose par défaut deux (voir plus) Workflows définis par des fichiers XML. Dans ces Workflows sont spécifiés les permissions, rôles utilisés. Libre aux utilisateurs/déployeurs de ORI de configurer/développer de nouveaux Workflows en prenant comme exemple ceux donnés par défaut.

OSWorkflow est dirigé via Spring et les beans Spring fournis en standard avec OSWorkflow. La persistance de OSWorkflow dans Hibernate est configurée via la configuration des beans Spring OSWorkflow également. Les fonctions/conditions définis dans les workflow peuvent être des beans définis via les fichiers de config spring. Voir ce lien.. Cela nous permet d'intégrer parfaitement OsWorkflow dans l'application

principale tout en conservant toute la souplesse d'OsWorkflow. Il est ainsi possible aux utilisateurs de développer et intégrer leurs propres fonctions et conditions dans les workflows OsWorkflow (et des les partager avec la communauté). C'est donc un excellent point d'entrée pour un développeur désirant étendre les possibilités/fonctionnalités d'Ori-Oai-Workflow.

Configurations

Grâce à Spring mais aussi à OsWorkflow, l'application est souple car paramétrable à différents niveaux. Du même coup, configurer Ori-Oai-Workflow de manière avancée demande d'éditer un certain nombre de fichiers. Les fichiers à configurer/modifier les plus importants (en terme fonctionnel) sont les fichiers de configuration spring, mais aussi les fichiers permettant d'implémenter un workflow OsWorkflow.

Modélisation

Modélisation

Cette section présente une sorte de première analyse de Ori-Oai-Workflow. Même si par rapport à l'implémentation qui en a été faite, cette modélisation reste très conceptuelle, elle permet de mieux appréhender l'architecture de Ori-Oai-Workflow.

Cas d'Utilisations

Ces différents cas d'utilisation seraient à affiner et compléter.

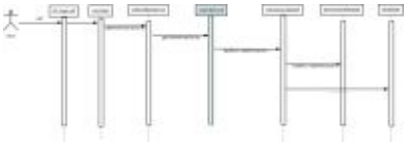
Pour une étude fonctionnelle poussée (et non pas une approche technique) des cas d'utilisations, référez-vous au document décrivant les cas d'utilisations "modèle" (auteurs: Rosa María Gómez de Regil et Romuald Lorthioir)

Diagrammes de séquence

Dans les diagrammes de séquence présentés ci-dessous, on constate que OsWorkflow dirige l'application : en fonction de la configuration du workflow actif et de l'action demandé par l'utilisateur ("intialiser un workflow", "valider la publication", "rendre priver", "demander la validation technique", etc), de l'état dans lequel se trouve l'instance de Workflow (donc ~ la fiche de métadonnées), OSWorkflow va appeler les méthodes métiers d'Ori-Oai-Workflow pour vérifier les conditions ou effectuer les functions adéquates.

Afficher la liste des entrées.

L'utilisateur demande à afficher les différentes entrées dans lesquelles il peut jouer un rôle sur une instance de workflow.



Créer/Modifier une instance de Workflow

L'utilisateur demande à créer/modifier une fiche de métadonnée et donc une instance de Workflow.

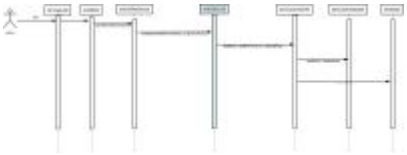


Diagramme de classes

Ce diagramme de classes se veut simple pour une bonne compréhension de l'ensemble. Il ne présente pas un certains nombres d'attributs

et surtout de classes qui sont d'ordre architectural (architecture par couche : données/dao/domain/web). De plus certaines associations n'ont pas été implémentées directement comme des associations POJO/Java : ce sont plus des associations conceptuelles établies par les configurations Spring et les services métiers.

Quelques Explications sur certaines classes :

Entry

> *Correspond à un ensemble d'instances de Workflows. Les instances de Workflows sont dans une même Entrée car elles décrivent la même entité. Elles sont distinctes car elles peuvent correspondrent à des versions différentes, décrirent l'entité avec des schémas différents.*

WorkflowInstance

> *Une instance de Workflow est défini par son contenu XML et son osworkflow (instance d'un OSWorkflow) associé (idworkflow + workflowName).*
> *Un workflowInstance représente donc La donnée principale autour de laquelle gravite l'application Spring.*

MetadataType

> *xpathTitle est le XPath correspondant au titre d'un XML répondant au schéma présent.*
> *Les différents MetadataType représentent les différents types de fiches de métadonnées que support l'application. Ils sont à configurer dans un fichier de config spring spécifique (cf la section Configuration de ce présent document).*
> *Concrètement, les différents types de fiches sont les différents types de fiches que pourra ajouter l'utilisateur (en fonction des droits qu'il a et donc des conditions des workflows spécifiés dans le MetadataType) : ** Ajouter une thèse*
>
> - *Ajouter une ressource pédagogique*
> - *etc.*

RoleStepCategory

> *Les différents RoleStepCategory représentent les différentes catégories de fiches (~ dossiers virtuels) listées à l'utilisateur. Ils permettent de lister dans un même ensemble des fiches de workflows différentes dans des états différents et sur lesquels l'utilisateur a des rôles différents. Ainsi on peut par exemple présenter à l'utilisateur toutes les fiches dont il est propriétaire qui sont dans l'états creation ou saisie_auteur issues des workflows model1 et easy par exemple.*
> *Ils sont à configurer dans un fichier de config spring spécifique (cf la section Configuration de ce présent document).*

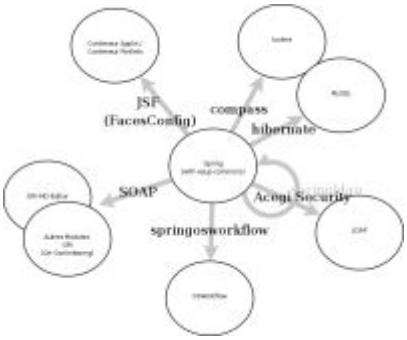
Les permissions sont à la fois positionnées par OSWorkflow et par Spring lors de l'initialisation des "permissions par défaut".

Implémentation

Implémentation

ori-oai-workflow-spring

Interactions des différentes technos avec Spring



JSF

JSF allié à Spring permet de faire tourner l'application aussi bien en mode servlet (dans un conteneur de servlets) qu'en mode portlet (dans un conteneur de portlets et donc dans un environnement Esup par exemple). JSF fournit les éléments pour récupérer les paramètres liés au contexte de l'exécution de l'application (la requête, la session, les paramètres init, et donc l'utilisateur connecté, etc...).

SOAP

Cf précédemment. On utilise ici un protocole maison en Soap (Web Service) pour faire interagir Spring et Orbeon Forms.

On utilise également SOAP pour interagir avec les autres modules de ORI.

L'implémentation est relativement aisée : * Côté Spring, XFire est utilisé, il permet d'exposer très simplement un Bean Spring (ses méthodes) en SOAP. Il permet d'utiliser un service Web tout aussi simplement : le service Web est accessible sous forme de Bean dans l'application.

- Côté Orbeon Forms, SOAP est supporté par Orbeon Forms, l'appel à un WebService SOAP se fait simplement via un fichier XPL.

SpringOsWorkflow

Cf précédemment. Permet d'appeler une instance d'OsWorkflow depuis Spring. OsWorkflow a l'énorme avantage de pouvoir iutégrer l'appel à des méthodes de Bean Spring en tant que conditions et fonctions d'un workflow.

Hibernate

Les données de l'application Spring sont stockées dans une BD SQL via Hibernate, Hibernate qui s'intègre parfaitement dans une applciation Spring. A noter que la configuration Hibernate Spring est utilisée également par OsWorkflow pour gérer la persistance.

Compass/Lucène

Permet d'indexer, rechercher et naviguer plus rapidement dans les fiches. On utilise Compass via Spring en le synchronisant directement sur les transactions Hibernate.

SpringLdap

Ldap sert à l'application de base de données utilisateurs/groupes. De même que pour les autres technos, Spring fournit des beans permettant d'interagir avec Ldap. Ils sont utilisés au travers d'Acegi.

Acegi Security

Acegi Security permet de gérer tout l'aspect security de l'application.

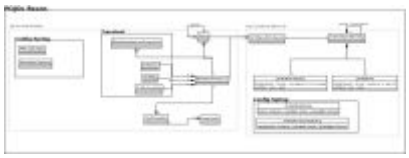
Acegi Security communique avec les sources de données utilisateurs/groupes pour l'authentification ainsi que comme base utilisateurs pour l'implémentation de RBAC (Role Based Access Control).

Acegi Security propose de base une implémentation d'une politique de sécurité de l'application. L'implémentation qui en a été faite en est ici tout autre : * Les données d'Acegi Security sont rendues persistantes via une implémentation Hibernate d'Acegi.

- Nous avons implémenté le modèle RBAC, ainsi par rapport à l'implémentation de base de Acegi nous avons : *** des rôles et des permissions définis sur les objets de type OriAclObjectIdentity, chaque objet étant associé (1/1) à un objet de type OriObjectAclAware (dont héritent WorkflowInstance et Entry)
    - - un système d'héritage d'objets OriAclObjectIdentity permettant de faire hériter les rôles et permissions d'un parent à un fils
      - une possibilité de définir des permissions et rôles sur l'objet racine de l'arbre d'héritage des objets OriAclObjectIdentity, cela via un fichier de configuration Spring
      - un système de mask où un rôle est définie par sa cible (un objet OriAclObjectIdentity), un recipient (le nom d'un groupe ou d'un utilisateur) et un mask (un entier) alors qu'une permission est définie par sa cible (un objet OriAclObjectIdentity), un recipient (le mask d'un rôle) et un mask (un entier)
- Il est possible d'ajouter/supprimer les masks de permissions et rôles disponibles dans l'application
- Une api permet de supprimer, ajouter vérifier un rôle ou une permission sur un objet de type OriAclObjectIdentity (cette api est notamment utilisée via les fonctions/conditions paramétrées dans les workflows OsWorkflow).

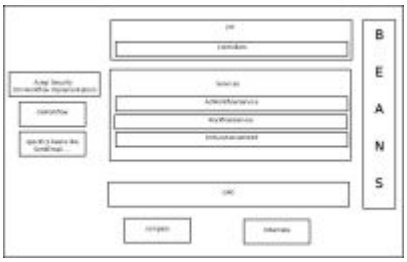
Beans/Pojos Ori-Oai-Workflow-Spring

Le diagramme ci-dessous donne le diagramme de classes des beans/pojos utilisés dans Ori-Oai-Workflow-Spring. Il ne donne pas les

beans/pojos issus de OsWorkflow. 

- La plupart des beans ainsi que la plupart de leurs attributs sont persistants via Hibernate, ils servent aussi dans le même temps comme DTOs : en fait c'est eux qui sont chargés d'information tout au long des couches Dao et Service puis qui sont directement présentés voir modifiés depuis les contrôleurs webs (cf la section "Architecture par couches" plus bas).
- Certains beans/attributs sont persistants dans les configs Spring uniquement, ils permettent de paramétrer l'application.
- Enfin d'autres sont simplement transients.

Architecture par couches



Comme dit plus haut, les beans (package org.ori.workflow.beans, qui sont en fait des pojos) sont transversaux aux couches de l'application. Ils représentent à la fois les données persistantes, les données récoltées via la couche service et les données visualisées voir éditées au niveau de la couche Web. * L'intérêt évident est que les différentes couches s'appuient ainsi sur les mêmes types de données, l'ensemble du code de l'application est ainsi très flexible et léger : l'ajout d'une donnée en tant qu'attribut de bean de la persistence à sa représentation côté web requiert par exemple des modifications très restreintes.

- Le fait cependant d'utiliser les mêmes beans au travers des différentes couches peut cependant, si on n'y prend garde, peut impliquer un certain nombre de contraintes/inconvénients.
  Cela implique par exemple que les attributs d'un même bean peuvent avoir des persistences différentes : persistence via Hibernate, OsWorkflow ou encore attribut transient ... ce qui peut perturber un développeur découvrant le code.
  Les objets de type WorkflowInstance en sont un exemple parfait. Afin que le développeur s'y retrouve au mieux, il est donc important que les commentaires JavaDoc sur les attributs des beans soient le plus précis possible sur le mode de persistence utilisé.
  De même les beans doivent répondre à la fois aux contraintes données par leur utilisation dans JSF et dans Hibernate. Mais là encore, il est intéressant de constater que l'utilisation du même type de bean à la fois dans Hibernate et dans JSF permet par exemple une implémentation simple d'un Converter JSF (voir même d'un converter générique, comme cela a été fait pour ORI-OAI-Workflow).

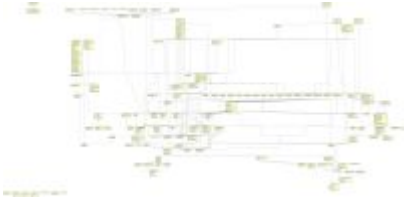
Tout comme le préconise Esup-Commons, tous les beans springs (on ne parle plus ici des beans pojos mais bien des beans springs) sont configurés via les fichiers de configuration spring. Cela permet d'uniformiser la déclaration des différents beans spring et de tirer partie au mieux de Spring IDE (dans Eclipse).

Au niveau de la couche de présentation dirigée par JSF, on associe à chaque page jsf un controller. Ce controller permet de récupérer et sauver les beans/pojos de l'application. Le choix a été pris de configurer les controllers JSF comme des beans spring de scope request. Un contexte applicatif est cependant maintenu en sauvant les états des beans/pojos d'une requête à une autre via la balise t:savestate de MyFaces Tomahawk. De même le bean/pojo UserContext par exemple est de scope session (mais tous les controllers sont bien quant à eux de scope request).

La partie services se compose de plusieurs sous parties. * La partie située dans le package org.ori.workflow.services.acls est spécifique à la sécurité de l'application. La majeure partie de ce package (qu'il faut mettre en relation avec le package org.ori.workflow.beans.acls) correspond à l'implémentation d'Acegi Security. AclWorkflowService est la classe instanciée pour correspondre au bean spring à mettre en relation avec le reste de l'application ori-oai-workflow.

- WorkflowService appelle à la fois la partie DAO et OsWorkflow.
- OriFunctions4OSWF ainsi que l'ensemble du package org.ori.workflow.services.oswfhelper correspondent aux beans appelés par OsWorkflow lors des fonctions et conditions paramétrés dans les workflows.

La partie DAO utilise hibernate pour la persistence de données. A noter que la session hibernate est gérée (ouverture/fermeture) par Esup Commons : on utilise en effet l'implémentation de Servlet/Portlet fournie par Esup Commons.



Changements de version

Un fichier CHANGELOG est disponible dans le dossier racine des sources de ori-oai-workflow(-spring)

Changements version 1.4.*

- Il est possible de configurer facilement ori-oai-workflow pour que les ressources pédagogiques ne soient plus seulement décrites en LOM mais également en LOMFR / SupLOMFR.
- L'authentification / identification / autorisation via le mécanisme Shibboleth peut être mis en place via des configurations spécifiques.
- Il est possible désormais de configurer l'appel de transformations XSL dans les workflow (en tant que fonction OsWorkflow osFunction), ces transformations peuvent faire appel directement à des fonctions/méthodes métier).
- Il est possible en tant qu'administrateur de procéder à des transformations XSL sur l'ensemble des fiches via l'interface graphique d'ori-oai-workflow-spring.
- Il est possible d'exporter les fiches d'ori-oai-workflow-spring sous leur format XML. L'export d'informations supplémentaires autour de l'usage de la fiche dans ori-oai-workflow est également possible (créateur, date de création, historique des transitions). *l'import fait en tant qu'administrateur tient alors compte également de ces informations si elles sont présentes*
- Il est possible de configurer des types de métadonnées spécifiques qui subissent une transformation XSL/XSLT lors d'un import de fiches.
- En tant qu'administrateur, il est possible d'appeller la réindexation des fiches dans ori-oai-indexing depuis un bouton dans l'IHM (il est toujours possible de le faire également via une tâche ANT).
- Plutôt que de configurer les XPATH directement dans les fichiers de conf de workflow osworkflow, on peut désormais utiliser un fichier XSL résultant de la compilation d'un schematron (les configurations par défaut ont été modifiées dans ce sens).
- Il est possible de télécharger les fichiers XML dans via la vue Historique/Detail.
- L'appel de la target ANT local-reindexall supprime maintenant en premier lieu l'index compass avant de le reconstruire intégralement.
- Un certain nombre d'autres corrections et améliorations ont également pu être apportées.

Changements version 1.1.*

Les valeurs des propriétés principales par défaut on été modifiées : elles apparaissent entre crochet et en majuscule. Elles peuvent soit être éditées manuellement, soit être modifées via la configuration du fichier principale du QuickInstall.

- Cette version n'utilise plus de base eXist, elle utilise maintenant uniquement la base de données SQL pour le stockage des données.
- Une indexation Compass/Lucène est utilisée, cela permet : *** de supporter sans perte de performance un nombre plus importants de fiches que précédemment : l'accès aux fiches se fait par le biais de requêtes sur l'index lucène.
    - d'améliorer le moteur de recherche interne dans les fiches de métadonnées
    - de permettre de trier par titre, date, créateur les fiches

- Un certain nombre de fonctionnalités dans ont été ajoutées : *** importation de fiches XML depuis l'IHM
    - copier/coller/suppression de fiches ou de patrons de fiches (support de plusieurs patrons de fiches)
    - bulles d'aide (que l'on souhaite améliorer prochainement cependant)
    - un certain nombre de méthodes WS ont été ajoutées, cela dans le but d'interractions avec des applications extérieures à ORI-OAI (LMS, CMS, Chaînes Editoriales ...).

- Des améliorations autres ont été apportées (debugage ou consolidation de fonctionnalités ...).

migration 1.5 vers 1.6

Pour les exploitants ne tournant pas sur les configurations par défaut, la migration de ori-oai-workflow de 1.5 vers 1.6 nécessite quelques manipulations, en effet les configurations ont en 1.6 été restructurées pour plus de souplesse et faciliter le travail de l'exploitant / intégrateur.

Nous donnons ici les manipulations qui ont servi à migrer l'application ori-oai-workflow du site UNIT www.unit.eu

Etape 1 : restructuration des configurations spécifiques

La version 1.6 apporte maintenant la possibilité de configurer l'ensemble de son workflow, type de métadonnées, catégories dans un répertoire spécifique.
Cela permet d'intégrer, maintenir et partager plus facilement ses configurations spécifiques.
UNIT avait avant la 1.6 modifié directement les configurations du workflow des ressources pédagogiques, workflow nommé "easy" : dans les versions antérieures à la 1.6 ce workflow comportait 3 états (privé, en attente de publciation et publié) UNIT avait rajouté simplement un troisième état "archivé" ; quelques changements au niveau du workflow avaient été faits au passage et des groupes spécifiques à UNIT avaient également été configurées.
Restructuré selon la 1.6, le dossier correspondant au workflow UNIT (les spécificités UNIT pour ori-oai-workflow) est donné ici : workflows personnalisés.
Il correspond à la retranscription de toutes les configurations que l'on avait pu faire dans les vesrions précédentes.

Etape 2 : fichier spring-migration.xml

La majorité des modifications de configurations entre la 1.5 et la 1.6 correspond en fait à une restructuration des fichiers en aux-même.
Elle comporte cependant une modification sur la syntaxe de description des rôles et permissions. En 1.6 syntaxe basée sur les puissances

de 2 a été abandonnée pour laisser place à l'usage de simples chaines de caractères.
Ainsi lors de l'étape 1, on aura pris garde à remplacer dans les workflows et attributions globales de rôles permissions les identifiants type "puissances de 2" des rôles/permissions par des identifiants type "chaine de caractères".

Reste ici à faire prendre en compte cette modification au niveau de la base de données.

Pour ce faire, la migration 1.5 -> 1.6 utilise via un fichier de configs un mapping présentant les correspondances nécessaires ; ce fichier est celui-ci conf/properties/spring/spring-migration.xml et il est à reconfigurer pour votre migration en fonction de votre workflow.

Voici ce que contient celui d'UNIT :

```
<?xml version="1.0" encoding="UTF-8" ?>


<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:aop="http://www.springframework.org/schema/aop" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

 <bean id="migrationService"
  class="org.orioai.workflow.services.migrations.MigrationService">
  <property name="wfDao" ref="wfDao" />
  <property name="userProfilDao" ref="userProfilDao" />
  <property name="indexingInfoDao" ref="indexingInfoDao" />
  <property name="genericDao" ref="genericDao" />
  <property name="xmlUtilsService" ref="xmlUtilsService" />
  <property name="xslTransformService" ref="xslTransformService" />
  <property name="migration15To16" ref="migration15To16" />
 </bean>


 <bean id="migration15To16" class="org.orioai.workflow.services.migrations.Migration15To16">

  <property name="workflows">
   <props>
    <prop key="easy">unit_lom</prop>
    <!-- <prop key="very_easy">dc_easy</prop> -->
   </props>
  </property>

  <property name="permissions">
   <props>
    <prop key="1">UNIT_LOM_ADMINISTRATION</prop>
    <prop key="2">UNIT_LOM_READ</prop>
    <prop key="4">UNIT_LOM_WRITE</prop>
    <prop key="8">UNIT_LOM_CREATE</prop>
    <prop key="16">UNIT_LOM_DELETE</prop>
    <prop key="32">UNIT_LOM_MODERATE</prop>
    <prop key="64">UNIT_LOM_CREATE_FORM</prop>
    <prop key="128">UNIT_LOM_FORM</prop>
    <prop key="256">UNIT_LOM_USE_DOC_FORM</prop>
    <prop key="512">UNIT_LOM_USE_RIGHTS_FORM</prop>
    <prop key="1024">UNIT_LOM_USE_TECHNICAL_FORM</prop>
    <prop key="2048">UNIT_LOM_USE_ENTITY_FORM</prop>
    <prop key="4096">UNIT_LOM_MANAGE_FORM</prop>
   </props>
  </property>
  <property name="roles">
   <props>
    <prop key="1">UNIT_LOM_AUTHENTICATED</prop>
    <prop key="2">UNIT_LOM_OWNER</prop>
    <prop key="4">UNIT_ANNOTATION_MODERATOR UNIT_LOM_MODERATOR</prop>
    <prop key="8">UNIT_LOM_ENTITY_MODERATOR</prop>
    <prop key="16">UNIT_LOM_RIGHTS_MODERATOR</prop>
    <prop key="32">UNIT_LOM_TECHNICS_MODERATOR</prop>
    <prop key="64">UNIT_LOM_ARCHIVER</prop>
    <prop key="128">UNIT_LOM_ADMINISTRATOR</prop>
    <prop key="256">UNIT_LOM_OWNER_PLUS</prop>
   </props>
  </property>
 </bean>


</beans>
```

A chaque puissance de 2, nous avons associé la ou les chaines de caractères qui correspondent, ce par rapport donc à nos nouvelles

configurations du workflow, des affectations des permissions et rôles.

On notera également que l'on a la possibilité de mettre à jour le nom du workflow via le mapping suivant :

```
<property name="workflows">
   <props>
    <prop key="easy">unit_lom</prop>
    <!-- <prop key="very_easy">dc_easy</prop> -->
   </props>
  </property>
```

En effet afin de rester indépendant des autres configurations fournies par défaut dans ori-oai-workflow 1.6, nous avons décidé de changer le nom du workflow. Le workflow que nous utilisions était celui donné par défaut, nommé easy. Nous avons renommé finalement celui utilisé (cf workflows.xml) en unit_lom.

Etape 3 : migration effective

Une fois fait cela (la procédure sera d'autant plus longue que l'on utilise plus de workflows/types de métadonnées), on peut enfin lancer la migration effective de orioai-workflow-spring (au niveau des données).
Avant de procéder à cette dernière étape, on s'assurera d'avoir une sauvegarde de la base de données ori-oai-workflow (on aura d'ailleurs pour être sûr, sauvegardé avant tout chose évidemment tout le répertoire ori-oai-workflow 1.5 source utilisé jusque-là pour déployer ori-oai-workflow).

On lance donc simplement

```
ant upgrade
```

depuis le répertoire source ori-oai-workflow 1.6 contenant nos configurations spécifiques 1.6.
Cela a pour conséquence de modifier la base de données au mieux et de mettre à jour l'index local d'ori-oai-workflow à la suite .

Etape 4 : mise à jour des permissions globales

Vu que les permissions et rôle globaux ont dû être redéfinis, il faut également enfin lancer un

```
ant update-acls
```

Installation

Il existe plusieurs modes d'installation de ce module. Le mode recommandé est l'utilisation ori-oai-quick-install. Ceci vous permettra de déployer la suite ori-oai avec un minimum de personnalisation tout ceci en utilisant un seul fichier de configuration.

L'installation manuelle vous fera éditer manuellement différents fichiers afin de configurer au mieux votre application.

Il est préférable d'utiliser la première solution. En effet, celle-ci vous apportera un déploiement rapide de ORI-OAI sur un serveur de production avec une configuration de base. Vous pourrez toutefois après cette installation apporter toutes les configurations avancées que vous souhaitez à vos modules.

Reportez-vous à la documentation en ligne d'installation de ORI-OAI.

Installation manuelle

Comme dit dans les spécifications les différents modules utilisés dans Ori-Oai-Workflow ainsi que son implémentation via Spring lui confèrent une certaine souplesse. Cette souplesse implique de fait une certaine complexité dans la configuration de l'ensemble de l'application.

Voici le plan de ce chapitre:

Pré-Requis

Quel que soit le workflow que vous configurerez, Ori-Oai-Workflow nécessite en pré-requis que plusieurs services soient disponibles :

- Une base de données SQL supportant les transactions : MySql en Innodb ou une autre BD SQL avec un moteur transactionnel par défaut.

> Dans le cas de MySQL, faites attention à ce que votre base de données pour Ori-Oai-Workflow ait bien le engine sto
>
> Sous linux, dans my.cnf, pour les versions de mysql récente, on pourra vérifier que l'on a :
>
> ```
> default-storage_engine = innodb
> ```
>
> Pour une version plus anciennes, l'option est de la forme :
>
> ```
> default-table-type = innodb
> ```
>
> Commandes pour vérifier en ligne de commande (client mysql) le storage utilisé sur une table de votre base :
>
> ```
> mysql> use `ori-oai-workflow`;
> Database changed
> mysql> show table status like 'ORI_WORKFLOW_ACL_ROLE';
> +----------------------+--------+------------+------+----------------+-------------+
> | Name                 | Type   | Row_format | Rows | Avg_row_length | Data_length |
> +----------------------+--------+------------+------+----------------+-------------+
> | ORI_WORKFLOW_ACL_ROLE | InnoDB | Dynamic    |    5 |           3276 |       16384 |
> +----------------------+--------+------------+------+----------------+-------------+
> 1 row in set (0.00 sec)
> ```

- Un annuaire Ldap dans lequel on va pouvoir récupérer les informations des utilisateurs. Si on utilise l'authentification CAS et non Ldap, il faudra s'assurer que le login CAS correspond bien à l'UID dand Ldap (*depuis la 1.4.0, on notera que l'annuaire ldap peut maintenant être remplacé par l'utilisation de Shibboleth*).
- un serveur smtp afin d'envoyer les emails d'exception à un email donné (en fait, peut être optionnel en modificant le fichier de configuration exceptionHandling.xml).
  Le serveur smtp est utilisé également dans les workflows lors de l'envoi de mails aux modérateurs, etc.

A cela, et suivant votre workflow, vous aurez certainement besoin d'autres services d'activés comme

- le module d'indexation utilisé usuellement lors d'une transition de type "publication".
- ...

Fichiers à configurer par l'utilisateur déployant Ori-Oai-Workflow

build.properties

deploy.home permet de définir le répertoire dans lequel on veut déployer l'application

La propriété custom.recover.files permet de lister des fichiers à récupérer lors d'une mise à jour d'une version du module. Notez que dans le build.xml, un certain nombre de fichiers sont de facto récupérés si disponibles dans une version antérieure lors des mises à jour. Actuellement c'est ceux-ci:

- conf/properties/main-config.properties
- conf/properties/spring/spring-categories.xml
- conf/properties/spring/spring-metadata-types.xml
- conf/properties/spring/spring-webservices-client.xml
- conf/properties/spring/acegi/acegi-acls-root.xml
- conf/properties/spring/acegi/acegi-authentication-additional-groups.xml
- conf/properties/spring/osworkflow/workflows.xml
- WebContent/WEB-INF/web.xml

> Ces fichiers représentant les fichiers que vous aurez sans doute besoin de modifier pour paramétrer vos workflows

Enfin il est possible également de définir un zip à importer dans l'application via un script ant.

l'import directement depuis l'interface web est cependant plus simple à mettre en oeuvre

conf/properties/main-config.properties

Ce fichier permet de définir les principales configurations d'ordre technique de l'application : authentification ldap/cas, connection ldap, BD SQL, Module d'indexation, serveur smtp, ...

A noter que pour le ldap, vous pouvez "simuler" des groupes via des filtres ldap en configurant le fichier conf/properties/spring/acegi/acegi-authentication-additional-groups.xml

Enfin notez surtout que ce fichier conf/properties/main-config.properties est le fichier principal des configurations et vous renvoie à d'autres fichiers de configuration: les commentaires sont donc à prendre en compte *cela vaut pour les autres fichiers de configuration également bien sûr*. Il est avant tout destiné à une installation rapide de l'application mais ne permet quasiment pas de personnaliser votre module. Très vite après une première installation rapide, vous aurez certainement besoin de personnaliser et adapter les workflow d'exemple à vos utilisateurs, à vos cas d'utilisations, à votre SI, à votre mode de fonctionnement, ...

Installation / migration / maintenance via ANT

Une fois les différents fichiers de configuration modifiés pour correspondre à vos besoins, vous pouvez enfin installer/déployer/initialiser votre applciation dans le tomcat cible (cf build.properties). Pour cela un certain nombre de targets ant est à votre disposition :

- clean: pour supprimer les fichiers compilés et déployés.
- deploy: permet de compiler et déployer l'application dans le tomcat.
- all: appelle clean et deploy successivement
- init: pour initialiser la base de données sql : notez que la base doit déjà existée sur le serveur sql (ATTENTION: si des tables pré-existent elles sont écrasées !).
- update-acls: permet d'informer la base de données des modifications des permissions/rôles faites dans les fichiers de configuration.
- upgrade: mets à jour la base de données entre 2 versions d'ori-oai-workflow-spring (target indispensable à appeller pour effectuer une mise à jour entre 2 version - pensez à sauvegarder votre base de données au préalable)
- importmetadatas: permet d'importer des métadonnées depuis un zip contenant des fichiers xml de métadonnées (le plus simple est cependant de passer par l'Interface Web pour ce faire)
- local-reindexall: permet de mettre à jour l'index local de l'application ori-oai-workflow. ori-oai-workflow utilise en effet un index local (Compass/Lucène) pour permettre une navigation et une recharche rapide dans les fiches. Cet index est indispensable au bon fonctionnement de l'application. Il est synchronisé avec la base de données. Dans certains cas il peut être utile de le reconstruire complètement (resynchroniser) avec la base de données. C'est le but de cette target. Depuis la 1.4 avant de recréer l'index, local-reindexall supprime de fait au préalable l'index (pour le reconstruire entièrement).
- clear-index: la target clear-index détruit l'index local (n'est plus utile depuis la 1.4, cf la target local-reindexall).
- reindexall: demande au module ori-oai-indexing de réindexer toutes les fiches qui sont censées être indexées, c'est à dire qui sont marquées comme étant indexées. Cela peut être utile lors de la reconstruction complète de l'index du module ori-oai-indexing.

Pour une installation, vous devrez normalement appeler les targets suivantes dans l'ordre :

- deploy,
- init,
- update-acls.
  Avant cela vous aurez :
- configuré les différents fichiers,
- démarré la base de données sql (en y créant la base de données donnée dans le fichier de conf du module),
- ...
  Après l'appel des targets, vous pourrez démarrer le serveur tomcat.

Pour une mise à jour (attention de ne pas appeler la target init qui initalise votre base de données, vous devrez normalement appeler les targets suivantes dans l'ordre (l'application ori-oai-workflow *le tomcat* doit alors être stoppé) :

- clean,
- upgrade,
- deploy.
  Au préalable vous aurez
- fait des sauvegardes de la base MySQL,
- fait une sauvegarde de vos sources et configurations du module
- fait un switch subversion pour passer du tag précédent à ce nouveau tag au niveau de vos sources (si vous utilisez subversion pour synchroniser vos sources au niveau de l'intégration/exploitation ORI-OAI),

Utilisez les possibilités de
svn switch

pour passer d'un tag à un autre si vous être branché via Subversion. Cela vous permet de fusionner vos modifications (fichiers de config et autres) et les modifications apportées dans la nouvelle version astucieusement. Subversion directement en ligne de commandes est le moyen le plus confortable.=> le fichier CHANGELOG est à consulter à chaque mise à jour - il tente de vous apporter un certain nombre d'informations sur les changements opérés et que vous devrez donc prendre en compte.
Attention, ori-oai-workflow devient en 1.6 plus modulaire au niveau des ajouts de supports de nouveaux types de métadonnées, de workflow différents. Aussi la migration de 1.5 vers 1.6 implique une restructuration complète de vos configurations, le svn switch n'est de fait pas d'une grande utilité au niveau de cette migration spéciale.
=> plus d'infos sur cette migration 1.5 -> 1.6 sur cette page dédiée.

- résolu les conflits (en vérifiant rapidement vos configurations/personnalisations propres)
- vérifié que votre base SQL tourne

Les librairies pluto-1.0.1-rc4.jar et portlet-api-1.0.jar qui servent en fait pour l'utilisation de ori-oai-workflow-spring en mode portlet doivent être ajoutées manuellement à vos librairies Tomcat si vous souhaitez faire tourner le module dans un container de portlet. Il faudra également décommenter les déclarations relatives à la servlet "portlet" dans le web.xml commentées par défaut.
Si vous souhaitez utiliser CAS, il faut comme indiqué dans les commentaires de main-config.properties configurer le web.xml sauf si vous passez par la facilité du Quick-Install qui fait alors cela pour vous. Si vous utilisez un certificat non reconnu par la JVM (auto-signé pour des tests par exemple), il faut également ne pas oublier d'ajouter un trustStore et de le spécifier dans les paramètres Tomcat lors du lancement de celui-ci :

```
CATALINA_OPTS="-Djavax.net.ssl.trustStore=E:/Java/ac-racine-cru.keystore"
```

=> L'utilisation d'un tel certificat est bien évidemment à déconseiller.

Tests

Vous pouvez maintenant passer à la phase de test.

Configurations avancées

Configurations avancées

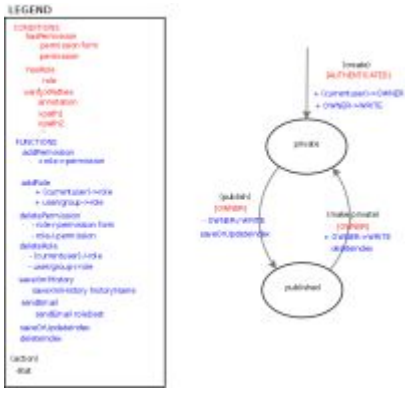
- Configurations par défaut
- Fichiers à configurer

Configurations par défaut

Configurations par défaut

Les configurations données par défaut dans une distribution ori-oai-workflow-spring sont relativement simples : on propose 2 workflows au travers de 2 "types de métadonnées" pour permettre l'édition du Dublin Core et du LOM / LOMFR / SupLOMFR via des formulaires complets.

- le workflow "very easy" est utilisé pour le Dublin Core.
  Un unique formulaire est configuré pour être utilsiable si l'utilisateur a la permission WRITE sur la fiche considérée



Voici ce que donne la version XML (implémentation OsWorkflow donc) proposée par défaut (conf/properties/spring/osworkflow/workflows/workflow_very_easy.xml)

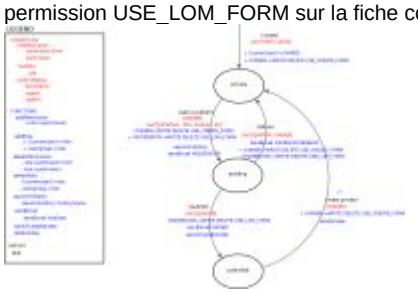
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow PUBLIC "-//OpenSymphony Group//DTD OSWorkflow 2.6//EN"
"http://www.opensymphony.com/osworkflow/workflow_2_8.dtd">
<workflow>
 <initial-actions>
  <action id="100" name="Start Workflow">
   <restrict-to>
    <conditions type="AND">
     <condition type="spring">
      <arg name="bean.name">hasRole</arg>
      <arg name="mask">1</arg><!-- AUTHENTICATED -->
     </condition>
    </conditions>
   </restrict-to>
   <results>
    <unconditional-result old-status="Finished"
     status="Underway" step="1" />
   </results>
   <post-functions>
    <function type="spring">
     <arg name="bean.name">addRole</arg>
     <arg name="mask">2</arg><!-- Role 2 = OWNER -->
     <!-- no recipient -> current user -->
    </function>
    <function type="spring">
     <arg name="bean.name">addPermission</arg>
     <arg name="mask">20</arg><!-- Permission 20 = WRITE + DELETE -->
     <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
    </function>
    <function type="spring">
     <arg name="bean.name">xslTransform</arg>
     <arg name="xslPath">properties/xsl/osfunctions/dcSetCreator.xsl</arg>
    </function>
   </post-functions>
  </action>
 </initial-actions>
 <steps>
  <step id="1" name="Private">
   <actions>
    <action id="1" name="Publish">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">2</arg><!-- Role 2 = OWNER -->
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result status="Underway"
       old-status="Finished" step="2" />
     </results>
     <post-functions>
      <function type="spring">
```

```
      <arg name="bean.name">deletePermission</arg>
      <arg name="mask">20</arg><!-- Permission 20 = WRITE + DELETE -->
      <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
     </function>
     <function type="spring">
      <arg name="bean.name">saveOrUpdateIndex</arg>
      <arg name="idOriIndexing">indexingServicePublic</arg>
     </function>
    </post-functions>
   </action>
  </actions>
 </step>
 <step id="2" name="Public">
  <actions>
   <action id="2" name="Make Private">
    <restrict-to>
     <conditions type="AND">
      <condition type="spring">
       <arg name="bean.name">hasRole</arg>
       <arg name="mask">2</arg><!-- Role 2 = OWNER -->
      </condition>
     </conditions>
    </restrict-to>
    <results>
     <unconditional-result old-status="Finished"
      status="Underway" step="1" />
    </results>
    <post-functions>
     <function type="spring">
      <arg name="bean.name">addPermission</arg>
      <arg name="mask">20</arg><!-- Permission 20 = WRITE + DELETE -->
      <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
     </function>
     <function type="spring">
      <arg name="bean.name">deleteIndex</arg>
      <arg name="idOriIndexing">indexingServicePublic</arg>
     </function>
    </post-functions>
   </action>
  </actions>
 </step>
</steps>
```

```
</workflow>
```

Vous noterez que ce fichier est en fait à mettre en relation avec un certain nombre d'autres fichiers de configuration : la configuration des rôles et permissions (le fait qu'un rôle Moderator, Administrator, SuperAdministrator, SuperValidator existe ou pas se configure par exemple), la configuration des permissions en fonction des formulaires (qui vont permettre pour un rôle donné, dans un état donné, de proposer ou non tel ou tel formulaire), etc.

- le workflow "easy" est utilisé pour le LOM. Un formulaire lom-author-light est configuré pour être utilsable si l'utilisateur a la permission USE_CREATE_FORM sur la fiche considéree. Un formulaire lom-full est configuré pour être utilsable si l'utilisateur a la permission USE_LOM_FORM sur la fiche considéree. (cf le fichier spring-metadata-types.xml)



Voici ce que donne la version XML (implémentation OsWorkflow donc) proposée par défaut (conf/properties/spring/osworkflow/workflows/workflow_easy.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow PUBLIC "-//OpenSymphony Group//DTD OSWorkflow 2.6//EN"
"http://www.opensymphony.com/osworkflow/workflow_2_8.dtd">
<workflow>
 <initial-actions>
  <action id="100" name="Start Workflow">
   <restrict-to>
    <conditions type="AND">
     <condition type="spring">
      <arg name="bean.name">hasRole</arg>
      <arg name="mask">1</arg><!-- AUTHENTICATED -->
     </condition>
    </conditions>
   </restrict-to>
   <results>
    <unconditional-result old-status="Finished"
     status="Underway" step="1" />
   </results>
   <post-functions>
    <function type="spring">
     <arg name="bean.name">addRole</arg>
     <arg name="mask">2</arg><!-- Role 2 = OWNER -->
     <!-- no recipient -> current user -->
    </function>
    <function type="spring">
     <arg name="bean.name">addPermission</arg>
     <arg name="mask">212</arg><!-- Permission 212 = WRITE + DELETE + USE_CREATE_FORM +
USE_LOM_FORM -->
     <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
    </function>
    <function type="spring">
      <arg name="bean.name">addPermission</arg>
      <arg name="mask">4244</arg><!-- Permission 4244 = WRITE + DELETE + USE_LOM_FORM +
USE_LOM_ADMIN_FORM -->
      <arg name="recipient">4</arg><!-- Role 4 = MODERATOR -->
    </function>
    <function type="spring">
     <arg name="bean.name">setInstanceIdentity</arg>
     <arg name="xpathMetadataIdentity">/lom:lom/lom:metaMetadata/lom:identifier[lom:catalog="URI"
]/lom:entry</arg>
    </function>
    <function type="spring">
     <arg name="bean.name">setEntryIdentity</arg>
     <arg name="xpathMetadataIdentity">/lom:lom/lom:general/lom:identifier[lom:catalog="URI"
]/lom:entry</arg>
```

```
    </function>
    <function type="spring">
     <arg name="bean.name">xslTransform</arg>
     <arg name="xslPath">properties/xsl/osfunctions/lomSetLifecycleAuthor.xsl</arg>
    </function>
   </post-functions>
  </action>
 </initial-actions>
 <steps>
  <step id="1" name="Private">
   <actions>
    <action id="1" name="Ask to Publish">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">2</arg><!-- Role 2 = OWNER -->
       </condition>
       <condition type="spring">
        <arg name="bean.name">verifySchematronXsl</arg>
        <arg name="schematronXslFile">/properties/xml/sch/lom-ori-sch.xsl</arg>
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result status="Underway"
       old-status="Finished" step="2" />
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">xslTransform</arg>
       <arg name="xslPath">properties/xsl/osfunctions/lomSetMetametadataCreator.xsl</arg>
      </function>
      <function type="spring">
       <arg name="bean.name">deletePermission</arg>
       <arg name="mask">212</arg><!-- Permission 212 = WRITE + DELETE + USE_CREATE_FORM +
USE_LOM_FORM -->
       <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
      </function>
      <function type="spring">
       <arg name="bean.name">addPermission</arg>
       <arg name="mask">4244</arg><!-- Permission 4244 = WRITE + DELETE + USE_LOM_FORM +
USE_LOM_ADMIN_FORM -->
       <arg name="recipient">4</arg><!-- Role 4 = MODERATOR -->
      </function>
      <function type="spring">
       <arg name="bean.name">saveXmlHistory</arg>
      </function>
      <function type="spring">
       <arg name="bean.name">sendEmail</arg>
       <arg name="message">
        <![CDATA[
{0} vient de placer sa fiche en attente de publication
-------------------------------------------
Voici ses observations.
{2}
]]>
       </arg>
       <arg name="smtpToMaskRole">4</arg>
       <!-- Role 4 = MODERATOR -->
      </function>
     </post-functions>
    </action>
    <action id="5" name="Publish">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">132</arg><!-- Role 132 = MODERATOR + ADMINISTRATOR  -->
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result status="Underway"
       old-status="Finished" step="3" />
```

```
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">deletePermission</arg>
       <arg name="mask">212</arg><!-- Permission 212 = WRITE + DELETE + USE_CREATE_FORM +
USE_LOM_FORM -->
       <arg name="recipient">6</arg><!-- Role 6 = MODERATOR + OWNER -->
      </function>
<!--      <function type="spring">
       <arg name="bean.name">saveOrUpdateIndex</arg>
       <arg name="idOriIndexing">indexingServicePublic</arg>
      </function>
-->
      <function type="spring">
       <arg name="bean.name">sendEmail</arg>
       <arg name="message">
        <![CDATA[
{0} vient de publier votre fiche.
-------------------------------------------
Voici ses observations.
{2}
]]>
       </arg>
       <arg name="smtpToMaskRole">2</arg>
       <!-- Role 2 = OWNER -->
      </function>
     </post-functions>
    </action>
   </actions>
  </step>
  <step id="2" name="Pending">
   <actions>
    <action id="2" name="Publish">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">4</arg><!-- Role 4 = MODERATOR  -->
       </condition>
       <!--
       <condition type="spring">
        <arg name="bean.name">verifySchematronXsl</arg>
        <arg name="schematronXslFile">/properties/xml/sch/lomfr_test_shematron.xsl</arg>
       </condition>
       -->
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result status="Underway"
       old-status="Finished" step="3" />
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">xslTransform</arg>
       <arg name="xslPath">properties/xsl/osfunctions/lomSetMetametadataValidator.xsl</arg>
      </function>
      <function type="spring">
       <arg name="bean.name">deletePermission</arg>
       <arg name="mask">4244</arg><!-- Permission 4244 = WRITE + DELETE + USE_LOM_FORM +
USE_LOM_ADMIN_FORM -->
       <arg name="recipient">4</arg><!-- Role 4 = MODERATOR -->
      </function>
<!--      <function type="spring">
       <arg name="bean.name">saveOrUpdateIndex</arg>
       <arg name="idOriIndexing">indexingServicePublic</arg>
      </function>
-->
      <function type="spring">
       <arg name="bean.name">sendEmail</arg>
       <arg name="message">
        <![CDATA[
{0} vient de publier votre fiche.
-------------------------------------------
Voici ses observations.
{2}
```

```
]]>
       </arg>
       <arg name="smtpToMaskRole">2</arg>
       <!-- Role 2 = OWNER -->
      </function>
     </post-functions>
    </action>
    <action id="3" name="Reject">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">6</arg><!-- Role 6 = MODERATOR or OWner : 2 + 4  -->
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result status="Underway"
       old-status="Finished" step="1" />
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">deletePermission</arg>
       <arg name="mask">4244</arg><!-- Permission 4244 = WRITE + DELETE + USE_LOM_FORM +
USE_LOM_ADMIN_FORM -->
       <arg name="recipient">4</arg><!-- Role 4 = MODERATOR -->
      </function>
      <function type="spring">
       <arg name="bean.name">addPermission</arg>
       <arg name="mask">212</arg><!-- Permission 84 = WRITE + DELETE + USE_CREATE_FORM +
USE_LOM_FORM -->
       <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
      </function>
      <function type="spring">
       <arg name="bean.name">sendEmail</arg>
       <arg name="message">
        <![CDATA[
{0} vient de refuser une fiche.
-------------------------------------------
Voici ses observations.
{2}
]]>
       </arg>
       <arg name="smtpToMaskRole">6</arg>
       <!-- Role 6 = MODERATOR or OWner : 2 + 4 -->
      </function>
     </post-functions>
    </action>
   </actions>
  </step>
  <step id="3" name="Public">
   <actions>
    <action id="4" name="Make Private">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">2</arg><!-- Role 2 = OWNER -->
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result old-status="Finished"
       status="Underway" step="1" />
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">addPermission</arg>
       <arg name="mask">212</arg><!-- Permission 212 = WRITE + DELETE + USE_CREATE_FORM +
USE_LOM_FORM -->
       <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
      </function>
      <function type="spring">
        <arg name="bean.name">addPermission</arg>
        <arg name="mask">4244</arg><!-- Permission 4244 = WRITE + DELETE + USE_LOM_FORM +
```

```
USE_LOM_ADMIN -->
        <arg name="recipient">4</arg><!-- Role 4 = MODERATOR -->
      </function>
      <!--
      <function type="spring">
       <arg name="bean.name">deleteIndex</arg>
       <arg name="idOriIndexing">indexingServicePublic</arg>
      </function>
      -->
     </post-functions>
    </action>
    <action id="6" name="Make Archived">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">128</arg><!-- Role 128 = ADMIN -->
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result old-status="Finished"
       status="Underway" step="4" />
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">deleteIndex</arg>
       <arg name="idOriIndexing">indexingServicePublic</arg>
      </function>
     </post-functions>
    </action>
   </actions>
  </step>
  <step id="4" name="Archived">
   <actions>
    <action id="7" name="Make Archived">
     <restrict-to>
      <conditions type="AND">
       <condition type="spring">
        <arg name="bean.name">hasRole</arg>
        <arg name="mask">128</arg><!-- Role 128 = ADMIN -->
       </condition>
      </conditions>
     </restrict-to>
     <results>
      <unconditional-result old-status="Finished"
       status="Underway" step="1" />
     </results>
     <post-functions>
      <function type="spring">
       <arg name="bean.name">addPermission</arg>
       <arg name="mask">212</arg><!-- Permission 212 = WRITE + DELETE + USE_CREATE_FORM +
USE_LOM_FORM -->
       <arg name="recipient">2</arg><!-- Role 2 = OWNER -->
      </function>
      <function type="spring">
        <arg name="bean.name">addPermission</arg>
        <arg name="mask">4244</arg><!-- Permission 4244 = WRITE + DELETE + USE_LOM_FORM +
USE_LOM_ADMIN -->
        <arg name="recipient">4</arg><!-- Role 4 = MODERATOR -->
      </function>
     </post-functions>
    </action>
   </actions>
  </step>
 </steps>
```

```
</workflow>
```

Fichiers à configurer

Vous trouverez dans cette partie toutes les possibilités de configuration suivantes:

Catégories et types de métadonnées

properties/spring/spring-categories.xml

Ce fichier permet de paramétrer les catégories de fiches qui seront affichées à l'utilisateur. Cela permet de mettre dans un même ensemble des fiches provenant de différents workflows ayant des états différents, sur lesquelles l'utilisateurs a des rôles différents. Ce fichier est donc à modifier en fonction des workflows, permissions et rôles configurés dans l'application et en fonction de ses besoins propres donc.

properties/spring/spring-metadata-types.xml

Ce fichier permet de paramétrer les types de fiches que peut ajouter un utilisateur. Notez toute fois qu'un utilisateur pourra effectivement ajouter une fiche d'un type donné si les conditions d'ajout déclarées dans le workflow correspondant sont vérifiées (conditions sur initial-action).

Un type de fiche Ori-Oai-Workflow est ainsi déterminé par un workflow OsWorkflow (identifié par workflowName), un nom de schéma, un xsl (permettant à partir d'un xml de produire un html), un fichier xml de défaut qui sera donné comme fiche vide à la création d'une nouvelle fiche (sauf dans le cas où l'utilisateur utilise des patrons de fiches).

C'est ici également que l'on configure les XForms à mettre en relation avec les fiches. On a une liste de XForms par type de contenu. Pour chaque XForms est donné un identifiant (réutilisé dans le positionnement des permissions) et une url correspondant à l'adresse du Serveur Orbeon présentant le XForms voulu (et c'est ici par exemple que vous pourrez ajouté un pointeur sur vos formulaires personnalisés ou vous permettant de supporter d'autres formats de MétaDonnées).

Acegi Security

Pour infos, les configurations possibles ici proviennent de l'implémentation spécifique de la politique de sécurité de Ori-Oai-Workflow.

properties/spring/acegi/acegi-acls-root.xml

Le bean entriesRoot permet de configurer les permissions et rôles par défaut dans l'application.

Les masks utilisés doivent être déclarés dans acegi-permissions.xml

Comme décrit précédemment entriesRoot est utilisé comme racine des objets Ori-Oai-Object-Identity

properties/spring/acegi/acegi-permissions.xml

Permet de configurer les permissions et rôles disponibles dans l'application.

Notez qu'il faut utiliser des puissances de 2 dans vos définitions de mask. Cela permet pour les permissions (pour l'instant) de simplifier l'écriture des conditions dans les configurations des workflows OsWorkflow.

Ces valeurs définissant les rôles et permissions sont utilisés dans les autres fichiers de configuration. On recommande donc lors de la configuration du module ORI-OAI-Workflow d'avoir un tableau comme celui ci-dessous sous les yeux.

PermissionsMask

Permission	Code
ADMINISTRATION	1
READ	2

WRITE	4
CREATE	8
DELETE	16
MODERATE	32
USE_CREATE_FORM	64
USE_LOM_FORM	128
USE_SCD_FORM	256
USE_JURIDIQUE_FORM	512
USE_TECHNIQUE_FORM	1024
USE_ENTITY_FORM	2048

RolesMask

Rôle	Code
AUTHENTICATED	1
OWNER	2
MODERATOR	4
ENTITE_MODERATEUR	8
JURIDIQUE_MODERATEUR	16
TECHNIQUE_MODERATEUR	32
SCD_MODERATEUR	64
ADMINISTRATOR	128

Ce tableau correspond aux rôles/permissions donnés par défaut dans une distribution ori-oai-workflow. Vous pouvez le modifier selon vos besoins.

Conditions et fonctions OsWorkflow

properties/spring/spring-osworkflow-helpers.xml

C'est ici que sont déclarés les beans spring utilisés dans les workflows OsWorkflow pour implémenter des conditions et des fonctions OsWorkflow.

L'administrateur/développeur voulant rajouter des fonctionnalités dans son Workflkow liées à son contexte de déploiement d'Ori pourra de la même façon qu'ici ajouter ses propres beans spring qu'il utilisera dans la configuration des workflows d'OsWorkflow. Le mieux sera de déclarer ses beans dans un fichier dédié à cela (vide par défaut) : conf/properties/spring/spring-custom.xml

OsWorkflow

La configuration de workflow(s) OsWorkflow est un élément capital dans le déploiement d'un Ori-Oai-Workflow. Comme dit ci-avant, c'est le workflow qui dirige le module. Dans Ori-Oai-Workflow, OsWorkflow n'a pas été modifié ou personnalisé, les fichiers de configuration des workflows sont donc des fichiers spécifiques OsWorkflow. Ainsi vous pouvez tout à fait vous référer à la documentation d'OsWorkflow si vous en sentez le besoin *de même que pour tous les fichiers de configuration de type Spring, vous pouvez vous référer à la documentation de Spring pour connaitre la syntaxe utilisée*.

Dans un premier temps, il est important de comprendre ce qu'est un workflow, revoyez dans la section "Specifications" la partie sur le Workflow si ce n'est pas clair.

Dans un second temps, il vous faut connaître les possibilités en matière de condition et fonctions et arriver à penser vos cas d'utilisations du module en terme de Workflow. Pour ce faire, rien ne vaut la présentation d'un exemple exhaustif que vous devrez mettre en relation avec le document fonctionnel sur les cas d'utilisations (document rédigé par Rosa María Gómez de Regil et Romuald Lorthioir).

On notera que ce graphe représentant le workflow des documents que l'on veut indexer n'a pas grand chose de spécifique à OsWorkflow ou à ORI-OAI. Il faut également se rappeler que les rôles, les permissions (dont les permissions d'utiliser tel ou tel formulaire) sont des éléments

configurables comme on l'a décrit ci-avant.

Une fois ce graphe conçu, la plus grosse partie du travail est faite : il reste à concevoir le XML implémentant ce diagramme en "langage" OsWorkflow, même si il en résulte un XML qui peut être conséquent, le langage XML d'OsWorkflow n'est pas complexe.

Ne pas oublier que Ori-Oai-Workflow doit utiliser plusieurs workflows suivant la nature des fiches à indexer, etc. (cf les "MetadataType").

Notes sur les conditions et fonctions disponibles actuellement et en natif dans Ori-Oai-Workflow

Conditions:

HasPermission

> *Condition sur le fait que l'utilisateur courant a une permission donnée.*

HasRole

> *Condition sur le fait que l'utilisateur courant a un rôle donné.*

VerifyXPathes

> *Vérifie que la résolution de chaque XPath donné en paramètre correspond à au moins un noeud à chaque fois. Place un message dans le listing d'informations de l'instance/fiche considéré si ce n'est pas le cas.*

> *Exécute le schématron (transformation XSL/XSLT via la XSL résultant de la "compilation XSL" d'un schématron) sur la fiche XML pour vérifier qu'il n'y a pas d'erreurs (règles schématron). Place des messages d'erreurs en conséquence dans le listing d'informations de l'instance/fiche considéré.*

Fonctions:

AddPermission

> *Ajoute une permission à un rôle sur l'instance courante.*

AddRole

> *Ajoute un rôle à un utilisateur ou groupe ou utilisateur courant sur l'instance courante.*
> *Un XPath et une valeur attendue sur le xpath donnée sont des paramètres optionnelles à cette fonction. Ils permettent de conditionner ce positionnement de rôle selon la présence ou non d'une valeur à un xpath donné. Par exemple cela permet de donner le rôle de modérateur au groupe de professeurs de mathématiques si la discipline dans la fiche LOM a été positionnée à Mathématiques.*

DeleteIndex

> *Demande à un ORI-OAI-Indexing de supprimer l'index courant de l'instance courante (si elle a été indexée préalablement).*

DeletePermission

> *Supprime une permission à un rôle sur l'instance courante.*

DeleteRole

> *Supprime un rôle à un utilisateur ou groupe ou utilisateur courant sur l'instance courante.*

RevertXml

> *Restore une version de la fiche d'un ancien état (il faut pour cela que dans cet ancien état, un SaveXmlHistory ait été fait).*

SaveOrUpdateIndex

> *Demande à un ORI-OAI-Indexing d'indexer ou de réindexer la fiche de l'instance courante.*
>
> 
>
> *Depuis la 1.1.* cette fonction n'appelle pas/plus la fonction SaveXmlHistory : les 2 fonctions sont dissociées et indépendantes.*
> *Cette fonction DOIT obligatoirement être configurée comme étant une post-function dans OsWorkflow.*

SaveXmlHistory

> *Sauve le XML actuel comme un historique (archive) de l'état initial à la transition.*

> *Cette fonction DOIT obligatoirement être configurée comme étant une post-function dans OsWorkflow.*

SendEmail

> *Envoie des emails aux utilisateurs d'un rôle donné sur l'instance et/ou à un mail donné "en dur" dans le workflow.*
> *Cette fonction est donc capable de récupérer tous les emails des utilisateurs qui ont un rôle via un positionnement direct sur l'instance, via un positionnement au travers d'un groupe, etc. ... cela signifie que le nombre d'emails peut suivant les configurations être conséquent. La récolte des emails peut actuellement prendre un certain temps si la liste est longue (multiples requêtes ldap). [l'envoi des emails se fait toutefois dans des threads séparés). Nous recommandans la mesure du possible d'utiliser une liste de diffusion (mail "en dur") pour l'envoi de mails à tout un groupe donné et bien défini ...*

SetInstanceIdentity et SetEntryIdentity

> *Place des identifiants (propre à l'instance/fiche ou à l'entrée/entité considéré) au niveau d'un xpath donné.*

xslTransform

> *Appelle une transformations XSL sur la fiche XML de l'instance. Notez que dans le workflow_easy.xml donné par défaut, xslTransform est utilisé en utilisant par exemple properties/xsl/osfunctions/lomSetLifecycleAuthor.xsl : cette transformation fait appel directement à de la logique métier via une expression du type*
> ***select="document('#{domainService.getCurrentUserName()}')"***

Notez que l'odre des fonctions dans le fichier de config n'est pas annodin.
Un certain nombre des fonctions présentées ci-dessus vont fonctionner dans un contexte transactionnel .... mais pas toutes ....
Les fonctions qui tournent dans un contexte transactionnel annuleront leurs effets automatiquement si une des fonctions provoquait une erreur innatendue.
Dans ce contexte, et par rapport au listing ci-dessus, il est recommandé de lancer

- en tout dernier l'envoi d'emails (notez que contrairement aux autres fonctions, si l'envoi d'email échoue, rien n'est annulé pour autant),
- en avant dernier l'appel à des WebServices et donc à ori-oai-indexing par exemple,

Notez enfin que vous pouvez vous appuyer au mieux sur le workflow des ressources pédagogiques donné par défaut qui vous fournit ainsi un exemple significatif :
conf/properties/spring/osworkflow/workflows/workflow_easy.xml

Authentification / Autorisation avec Shibboleth *Nouveau - depuis 1.4.0*

Depuis la version 1.4.0 ori-oai-workflow peut utiliser Shibboleth à la fois pour l'authentification et l'autorisation. LDAP et CAS ne sont alors plus nécessaires et il est possible de partager une même instance d'ori-oai-workflow entre plusieurs établissements.

Pour que les mécanismes shibboleth soient pris en compte au niveau de ori-oai-workflow, ce dernier attend de pouvoir interpréter directement les attributs Shibboleth positionnés dans les paramètres HTTP de la requête cliente. On peut pour ce faire utiliser directement le mod_shib d'apache, dans lequel on requerra une authentification Shibboleth sur l'ensemble de l'applicatif.

Voici un exemple d'une telle configuration (le mod_shib et proxy_ajp (ProxyPass) sont ici utilisés et donc chargés par ailleurs :

```
ProxyPass /ori-oai-workflow-spring ajp://localhost:8009/ori-oai-workflow-spring

<Location /ori-oai-workflow-spring>
  AuthType shibboleth
  ShibRequireSession On
  ShibUseHeaders On
  require valid-user
</Location>
```

Reste ensuite à configurer l'interprétation des attributs Shibboleth au niveau ori-oai-workflow : on va pouvoir définir d'une part l'attribut Shibboleth qui va correspondre à l'identifiant d'un individu dans ori-oai-workflow, puis d'autres part les règles qui vont permettre de définir à quel(s) groupe(s) appartient l'individu connecté; ces règles, à paramétrer par l'exploitant, devront s'appuyer sur les attributs shibboleth disponibles.

Ces 2 configurations se font dans le fichier conf/properties/spring/acegi/acegi-authentication-shib.xml. Comme décrit rapidement dans le fichier, les règles doivent correspondent à des expressions Java renvoyant un boolean (true, false). Si l'expression interprétée renvoie true, alors on affecte à l'individu connecté le groupe donné en clé. shibAttrsMap est le nom de la variable définissant une table de hashage regroupant l'ensemble des attributs Shbboleth disponibles : nom/valeur.

```
<bean name="shibAuthenticator" class=
"org.orioai.workflow.services.application.shib.ShibAuthenticator">
  <property name="remoteUserShibAttr" value="Shib-InetOrgPerson-mail"/>
  <property name="shibGroupsRules">
   <description>
    shibGroupsRules defines a Map where
     * keys are groups names
     * values are java expressions (evaluated by Janino) which should return true or false -
    to construct expressions, you have a parameter named shibAttrsMap. Tis Map contains key/value
where :
      * keys are the shib parameter name
      * values are the shib parameter value
   </description>
   <map>
    <entry key="all">
     <value>true</value>
    </entry>
       <entry key="moderators">
     <value>shibAttrsMap.get("Shib-EP-UnscopedAffiliation").equals("faculty;member") &amp;&amp;
shibAttrsMap.get("Shib-EP-PrimaryAffiliation").equals("professor")</value>
    </entry>
    <entry key="admins">
     <value>shibAttrsMap.get("Shib-InetOrgPerson-mail").equals("Vincent.Bonamy@univ-rennes1.fr"
)</value>
    </entry>
   </map>
  </property>
 </bean>
```

Limitations: de par le mode de fonctionnement de shibboleth, il n'est pas possible de retrouver dynamiquement tous les individus d'un groupe donné. On ne pourra donc pas utiliser dans un workflow l'envoi de mails à l'ensemble des individus d'un groupe. On contournera ce problème en définissant directement les individus un à un comme destinataires des mails, ou mieux encore en définissant en amont une mailing-list permettant cet envoi de mails à un ensemble prédéfini d'individus.

Tests ANT

Tests ANT

Test de l'environnement et des configurations ...

Avant de déployer votre application, vous pouvez lancer un test d'appel aux principaus services utilisés et censés être configurés dans votre Workflow. Ceci permet d'identifier une misconfiguration, un service non lancé (ldap, smtp, ori-oai-indexing, ...) ...

Placez-vous dans le répertoire « **[ORI_HOME]/src/ori-oai-workflow-svn** ».

```
ant remoteconfigtest
```

Si tout fonctionne bien, voici ce que vous devriez obtenir à peu de chose près :

```
remoteconfigtest:
    [junit] Testsuite: org.orioai.workflow.RemoteConfigTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 11,761 sec
    [junit] ------------- Standard Output ---------------
    [junit] Retrieving document at 'null'.
    [junit] Retrieving document at 'null'.
    [junit] ------------- ---------------- ---------------
    [junit]
    [junit] Testcase: testGetVocabulariesId took 10,638 sec
    [junit] Testcase: testSearchIndexingServices took 0,986 sec
    [junit] Testcase: testInitialDirContextFactory took 0,049 sec
    [junit] Testcase: testConnectUrls took 0,048 sec
    [junit] Testcase: testSmtp took 0,033 sec

BUILD SUCCESSFUL
```

Contenu des tests ant remoteconfigtest :

Ces tests se divisent en deux aspects :

1. verification que les paramètres de connexions à distance aux autres modules ou services sont corrects

- TestVocabularyService
- TestIndexingServices
- TestLdap
- TestSmtp

2. verification que les Web services offerts par le module aux autres modules répondent correctement.
   TestMdEditorFormsUrls : test pour un des WS appelés par le md-editor

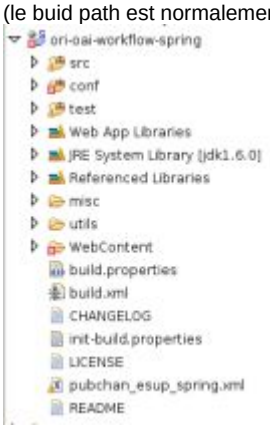
Aspects pratiques

Environnement Eclipse

ori-oai-workflow-spring: environnement de développement - Eclipse

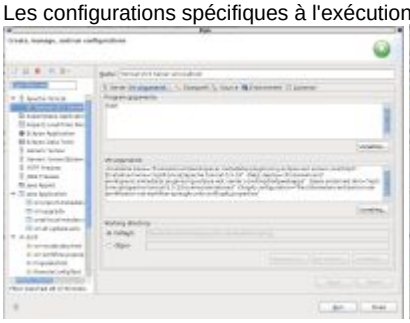
Partie adressée plutôt aux développeurs

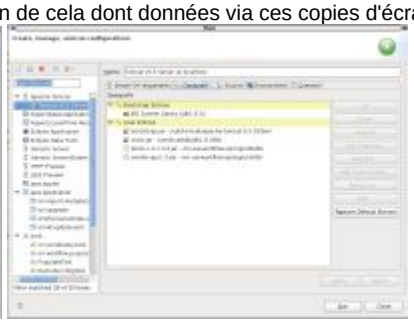
Le développement dans ori-oai-workflow-spring peut s'appuyer sur eclipse. La structuration des sources est structurée dans ce sens.
Les fichiers .classpath .project .springBeans sont donnés dans les répertoires subversion (tags et trunk) pour une intégration facile dans eclipse.
Pour ce faire, l'idée est de récupérer le projet (tag ou trunk), de l'intégrer en tant que nouveau "projet Web".
Ci-dessous une copie d'écran montrant ce que l'on doit alors obtenir
(le buid path est normalement configuré de suite comme il se doit, etc. cela grâce aux fichiers .classpath et .project).



On s'appuie sur WTP pour lancer le module depuis le eclipse.
Pour cela, on configure l'exécution du projet via la vue serveur en le paramétrant pour l'exécuter dans un serveur Tocmat (5 ou 6 au choix).

Les configurations spécifiques à l'exécution de cela dont données via ces copies d'écran.





Déboguage distant avec Eclipse

Un tutoriel d'utilisation des possibilités de déboguage distant d'Eclipse est disponible ici : Deboguage distant Eclipse

Encodage

Encodage

De par leurs fonctionnalités et leurs objectifs d'interropérabilité, les modules ORI-OAI sont voués à fonctionner dans un encodage UTF-8. Il est préférable (pour éviter au mieux les problèmes d'encodage dans les données, le rendu etc.) de positionner l'encodage à UTF-8 pour tous les composants acteurs dans ORI-OAI (les bases de données, les serveurs d'applciation, etc). Cela peut se faire de différentes manières (depuis les variables d'environnement du système par exemple [LANG]).

Le positionner en tant qu'option Tomcat est également préférable:

```
CATALINA_OPTS=-Dfile.encoding=UTF-8
```

Encodage et MySQL

Côté MySQL

Pour MySQL, le plus simple est de configurer l'**UTF-8** à la fois au niveau des **"Connection Character Sets"** et des **"Collations"**.
Pour ce faire, on ajoute dans la configuration de mysql (sous debian /etc/mysql/my.cnf) la confiuration suivante (après [mysqld] au même niveau que **default-storage_engine = innodb** :

```
character-set-server=utf8
collation-server=utf8_general_ci
```

On notera qu'après une telle modification il faut redémarrer le serveur mysql.

On peut ensuite vérifier la bonne prise en compte de ces variables :

```
mysql> SHOW VARIABLES LIKE 'character_set%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | latin1                     |
| character_set_connection | latin1                     |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | latin1                     |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'collation%';
+----------------------+-------------------+
| Variable_name        | Value             |
+----------------------+-------------------+
| collation_connection | latin1_swedish_ci |
| collation_database   | utf8_general_ci   |
| collation_server     | utf8_general_ci   |
+----------------------+-------------------+
3 rows in set (0.00 sec)
```

Côté client Hibernate

Si comme indiqué au dessus, on a positionné à la fois les **collation** et les **character set** à **utf-8**, cette configuration côté Hibernate n'est **a priori** pas nécessaire.

Ici on force Hibernate à demander un dialogue UTF-8 avec la base de données, cela en ajoutant dans les paramètres de connection

- useUnicode=true
- characterEncoding=utf8
- useOldUTF8Behavior=true

Pour ori-oai-workflow, dans **conf/properties/spring/common/dao/dao.xml**
On mettra à la place de

```
<property name="url">
  <value>${hibernate.connection.url}?emulateLocators=true</value>
</property>
```

cela :

```
<property name="url">
 <value>${hibernate.connection.url}?emulateLocators=true&amp;useUnicode=true
&amp;characterEncoding=utf8&amp;useOldUTF8Behavior=true</value>
</property>
```

Sauvegardes

Sauvegardes

Sauvegardes

Ori-Oai-Workflow utilise 1 base de données SQL (cf les pré-requis de l'installation/configuration d'Ori-Oai-Workflow Spring). Il faut donc penser à sauvegarder cette base de données très régulièrement, pour palier les potentiels pannes système, pbs disque dur, ...

Pensez également à faire une sauvegarde avant d'effectuer une mise à jour (upgrade) vers une version plus récente d'ori-oai-workflow.

MySql

Vous trouverez plus d'infos ici simplement : CommentCaMarche : Importer et exporter des données sous MySQL par exemple (la doc officielle peut également être intéressante à consulter).

```
mysqldump --all-databases -u root > /opt/ori/backup_mysql.dump
```

Autres ...

Si votre choix s'est porté sur une autre Base supportée par Hibernate (PostgreSql par exemple), la doc officielle de cette Base de Données devrait également vous permettre de mettre en oeuvre les sauvegardes adéquates.

Notez cependant que (mis à part Derby dans le démonstrateur .exe) l'équipe technique n'a à ce jour aucun retour sur l'utilisation d'autres bases que MySQL/InnoDB dans ori-oai-workflow

Sécurisation des Web Services

Sécurisation des Web Services

Sécurisation des Web Services

De même que pour l'encodage, la sécurisation des Web Services est un problème commun à tous les modules. Il peut être opportun de sécuriser les Web Services qui n'ont pas vocation à être utilisés par le monde entier (il faut y penser). Notamment les Web Services du Workflow, de l'indexing, etc. Certaines pages comme la page permettant le rechargement des vocabulaires dans ori-oai-md-editor doivent être sécurisées également.

Pour ce faire, le plus simple et le plus efficace est de sécuriser ces urls à un niveau assez "bas", un niveau "système" : * au niveau de la configuration du Tomcat voir de Apache par exemple (username/password en authentication basic http) côté serveur et au niveau du web.xml de l'application (dans les configs de XFire notamment) côté client.

- en n'ouvrant pas le port utilisé par le serveur d'application de l'indexing par exemple.
- en protégeant les accès via des contrôles sur les IPs
- etc.

Transformations XSL - XSLT

Transformations XSL - XSLT

ori-oai-workflow utilise des transformations XSL/XSLT :

- potentiellement lors des migrations d'une version à une autre
- en tant que fonctions OsWorkflow, pour manipuler les fiches de métadonnées dynamiquement lors d'un état à un autre

- pour permettre à l'administrateur de faire des modifier en masse les fiches de métadonnées stockées dans l'outil ori-oai-workflow.

XSLT 1.0.

Les transformations XSL/XSLT sont de type XSLT 1.0.

Appel métier depuis une XSL

On notera qu'il est possible de faire appeller à du code métier via des expressions du type

```
#{vocabularyService.getVocabulary("peopleLdapLocalProvider")}
```

Voir par exemple properties/xsl/osfunctions/lomSetLifecycleAuthor.xsl

EXSLT (encore non disponible en 1.4.0)

On notera que dans celles-ci on peut utiliser les fonctions fournies par EXSLT, pour ce faire, si on veut par exemple importer regexp.xsl livrée avec EXSLT, on l'importera de cette manière :

```
<xsl:import href="classpath:/properties/xsl/ext/all-exslt/regexp/regexp.xsl" />
```

Exemples / Cas d'utilisation.

- en fin de page ici on présente une XSL ajoutant pour toutes les fiches LOM les metadataSchema associés au LOM / LOMFR / SupLOMFR
- ici un exemple d'une XSL utilisant EXSLT qui tente de réordonner les FN en utilisant les N des VCARDs dans des fiches LOM
- ici un exemple d'une XSL qui modifie les URLS données dans des fiches LOM : https://www.unit.eu/* devenant http://www.unit.eu/*

Ressources pédagogiques

Ressources pédagogiques - Aspects pratiques

- LOM - LOMFR - SupLOMFR

LOM - LOMFR - SupLOMFR

LOM - LOMFR - SupLOMFR

Les profils d'application du LOMFR et du SupLOMFR correspondent techniquement à des extensions XML du LOM.
Les fichiers XML LOMFR - SupLOMFR sont des fichiers XML LOM comportant un espace de nom supplémentaire XML spécifique au LOMFR, des vocabulaires supplémentaires, des contraintes plus fortes (en terme de désignation des langues notamment) etc.
Pour plus d'informations à ce sujet, voyez le document de référence "Binding XML du LOMFR" (site www.lom-fr.org).

Formulaires LOM - LOMFR - SupLOMFR

C'est ainsi que dans ORI-OAI, faire du LOMFR / SupLOMFR et non plus seulement du LOM revient à utiliser des formulaires LOM "améliorés", c'est à dire des formulaires LOM présentant les possibilités offertes par LOMFR / SupLOMFR.

Aussi, que vous ayez ou non des fiches LOM déjà éditées en LOM, activer le LOMFR / SupLOMFR revient à proposer aux utilisateurs les formulaires dédiées au LOMFR / SupLOMFR.

- Si vous installez ORI-OAI via le module quick-install (recommandé) vous pouvez modifier les formulaires donnés par défaut pour les ressources pédagogiques via certains paramètres.
  Voici un copier/coller de la partie correspondant à ces paramètres :

```
# Dans la configuratuon par défaut, formulaire proposé à l'auteur pour l'édition des "Ressources Pédagogiques"
# Pour l'usage du LOMFR mettre 'lomfr-author-light', pour SupLOMFR mettre 'lomfr-sup-author-light'
# [configuré par défaut pour l'usage du LOM seul]
WORKFLOW_PEDAGO_FORM_AUTHOR=lom-author-light

# Dans la configuratuon par défaut, formulaire proposé à au validateur/modérateur pour l'édition des "Ressources Pédagogiques"
# Pour l'usage du LOMFR mettre 'lomfr-full', pour SupLOMFR mettre 'lomfr-sup-full'
# [configuré par défaut pour l'usage du LOM seul]
WORKFLOW_PEDAGO_FORM_FULL=lom-full

# Dans la configuratuon par défaut, fiche XML initiant les fiches "Ressources Pédagogiques"
# Pour l'usage du LOMFR mettre 'lomfr-prototype.xml', pour SupLOMFR mettre 'lomfr-sup-prototype.xml'
# [configuré par défaut pour l'usage du LOM seul]
WORKFLOW_PEDAGO_XML_PROTOTYPE=lom-prototype.xml
```

- En installant directement ori-oai-workflow sans ori-oai-quick-install, vous pouvez éditer ces mêmes paramètres directement dans le fichier conf/properties/main-config.properties :

```
# md-editor [spring-metadata-types.xml]
mdeditor.url=[PUBLIC_URL_MD_EDITOR]
mdeditor.pedagoform.author=[WORKFLOW_PEDAGO_FORM_AUTHOR]
mdeditor.pedagoform.full=[WORKFLOW_PEDAGO_FORM_FULL]
mdeditor.pedago.prototype=[WORKFLOW_PEDAGO_XML_PROTOTYPE]
```

- Enfin en configuration avancée, il est également possible de manipuler directement les URLs des éditeurs dans les fichiers de configs de beans springs, ici c'est le fichier conf/properties/spring/spring-metadata-types.xml qui manipule ces propriétés.

détails complémentaires

lang fr/fre

Depuis la vesion 1.4.0 les formulaires d'ori-oai-md-editor remplacent automatiquement à l'ouverture d'une fiche les fr (code lang sur 2 caractères) en fre (code langue sur 3 caractères) et certaines autres langues, cela via une feuille XSL. Cette même feuille XSL est utilisée dans la migration des fiches vers 1.4.0 (cf le fichier conf/properties/xsl/migrations/lom2lomfr-ready.xsl).
à noter qu'effectivement fr ou fre sont 2 possibilités correctes en LOM. Le LOMFR / SupLOMFR préconise quand à eux clairement l'usage de langue en 3 caractères

lom:metadataSchema

Dans lom:metaMetadata, il est possible d'indiquer des champs lom:metadataSchema.
Si l'on veut indiquer que l'on fait par exemple du SupLOMFR (et donc de facto du LOMFR et donc également du LOM), on donnera 3 champs indiquant cela :

```
<lom:metadataSchema>LOMv1.0</lom:metadataSchema>
   <lom:metadataSchema>LOMFRv1.0</lom:metadataSchema>
   <lom:metadataSchema>SupLOMFRv1.0</lom:metadataSchema>
```

Si vous souhaitez indiquer cela pour toutes vos fiches, vous pouvez utiliser la possibilité offerte par ori-oai-workflow (en tant qu'administrateur) de faire tourner une XSL sur toutes vos fiches qui fera le travail pour vous.
Une telle XSL pourra ressembler à celle-ci par exemple : lom2lom-metadataSchemaSupLOMFR.xsl

Utilisation

Utilisation - Référencer un document

- Captures d'écran

Note : Nous présentons dans ce document une utilisation simple du module ORI-OAI-workflow. Vous pouvez également visualiser d'autres captures d'écran à l'adresse http://sourcesup.cru.fr/ori-workflow/1.1/utilisation.html.

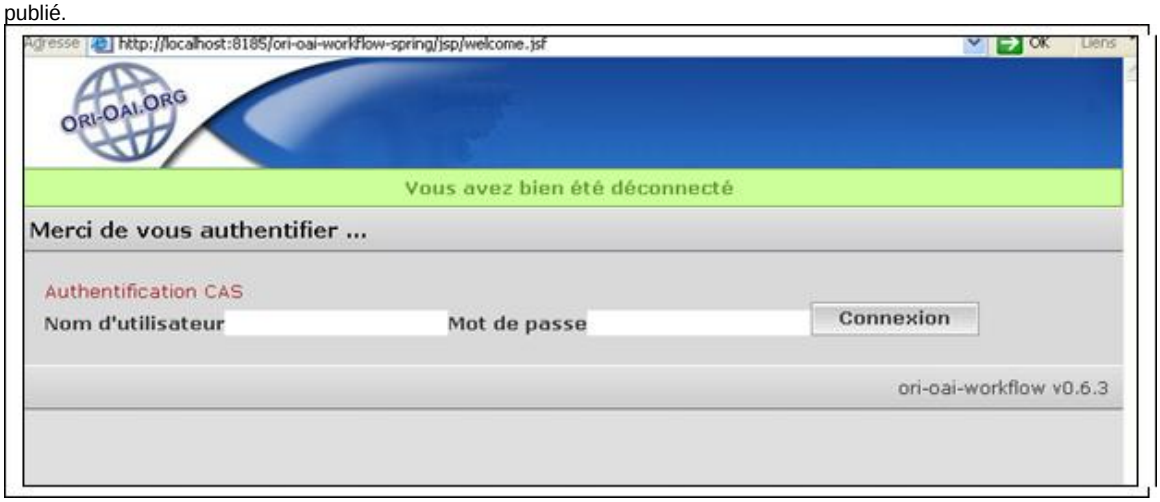
Avec votre navigateur, vous pouvez vous connecter à:

```
http://[HOST_INSTALL]:8185/ori-oai-workflow
```

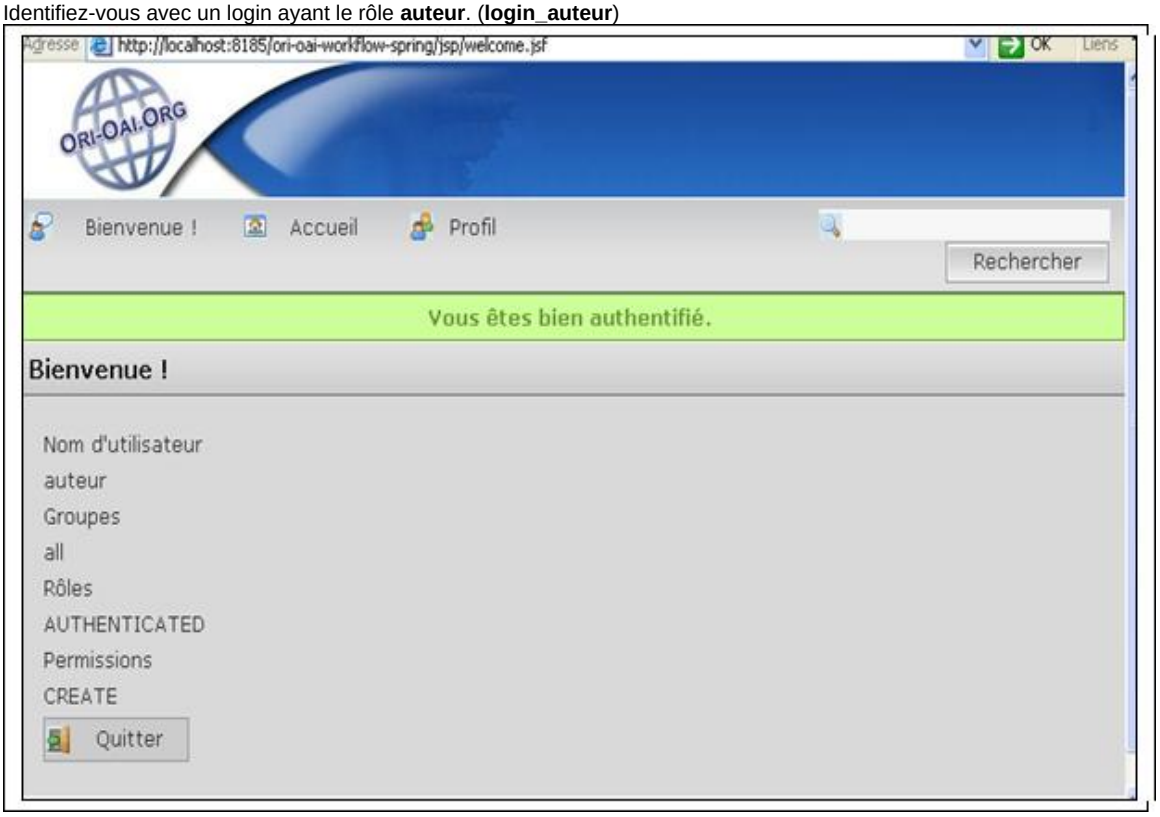
et tester l'application.

Le workflow proposé par défaut pour les fiches LOM comporte trois états :

- privé,
- en attente de publication,
- publié.



Identifiez-vous avec un login ayant le rôle **auteur**. (**login_auteur**)

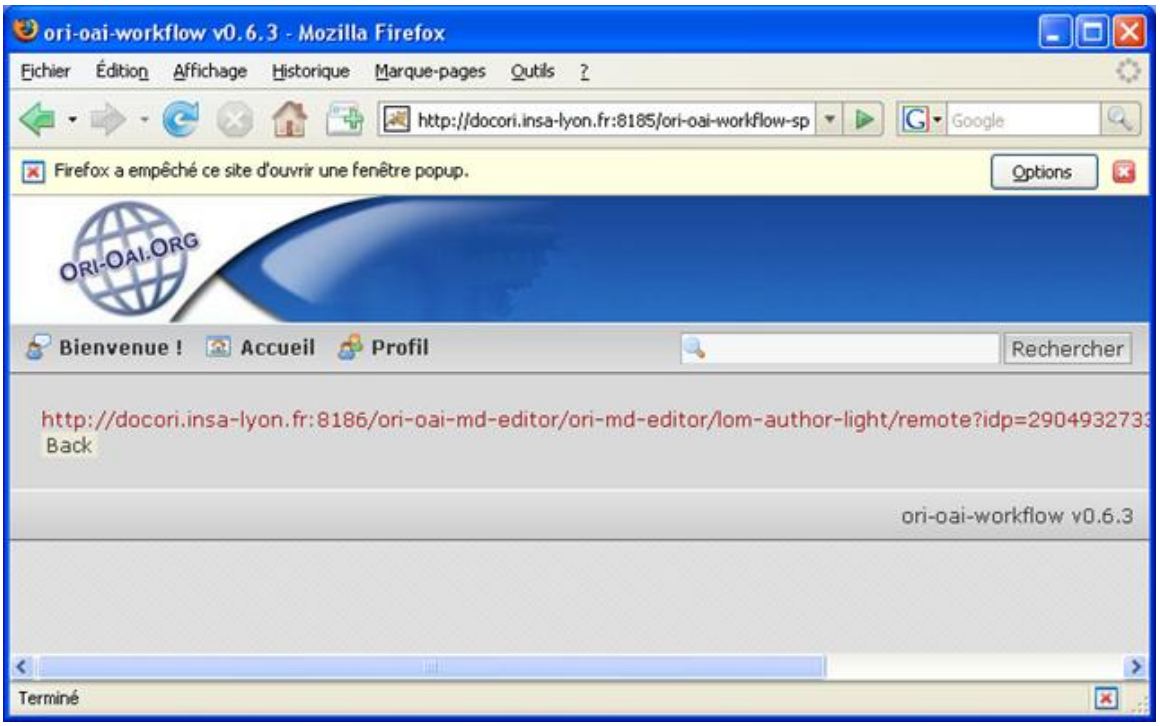


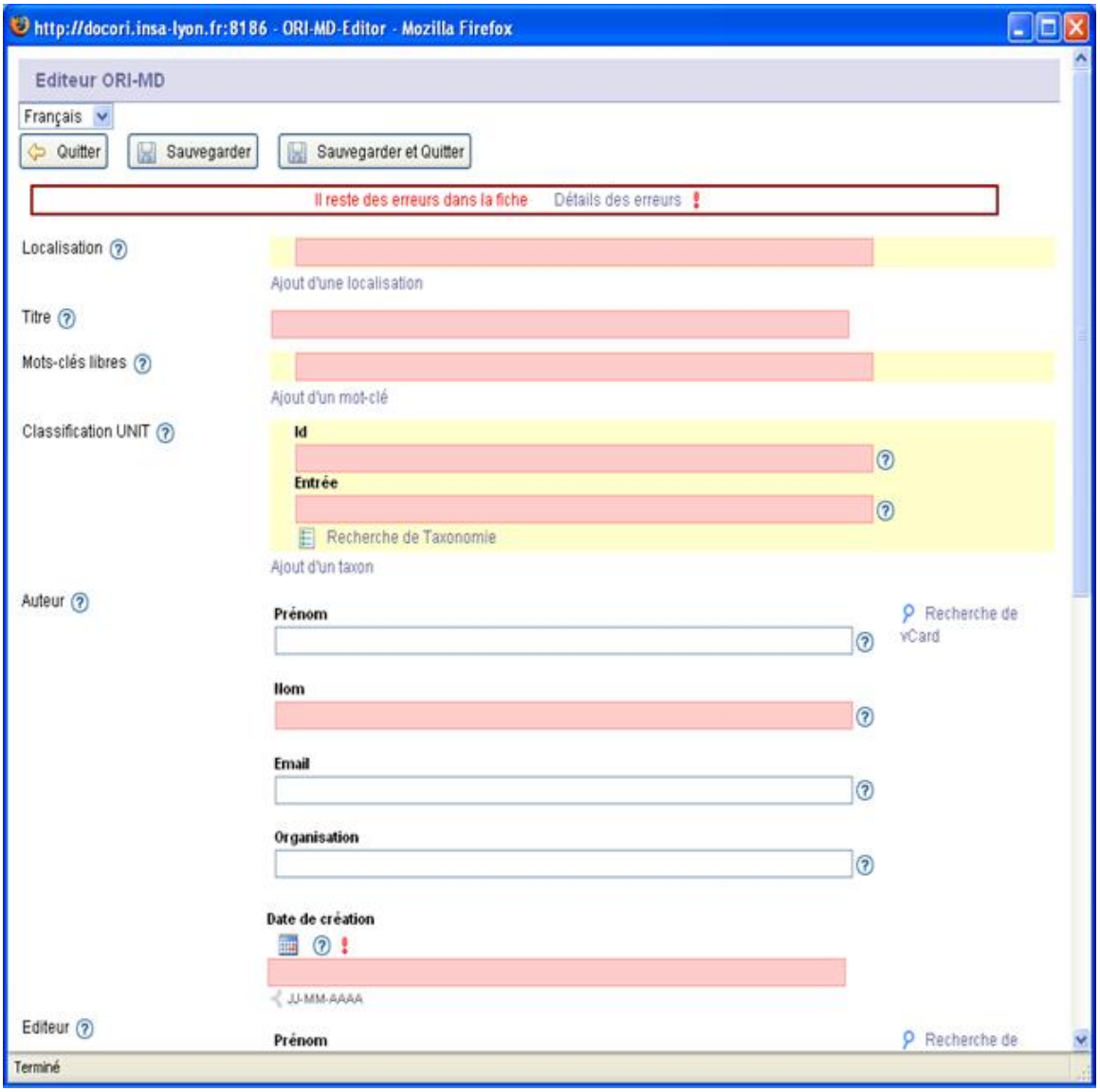
Cliquez sur « Accueil ».

Dans la partie « Référencer une Ressource », choisissez le type de métadonnées que vous souhaitez créer (par exemple « **Ressource Pédagogique [LOM]** »).

Votre aurez peut être une alerte indiquant que le navigateur a refusé d'ouvrir une fenêtre.

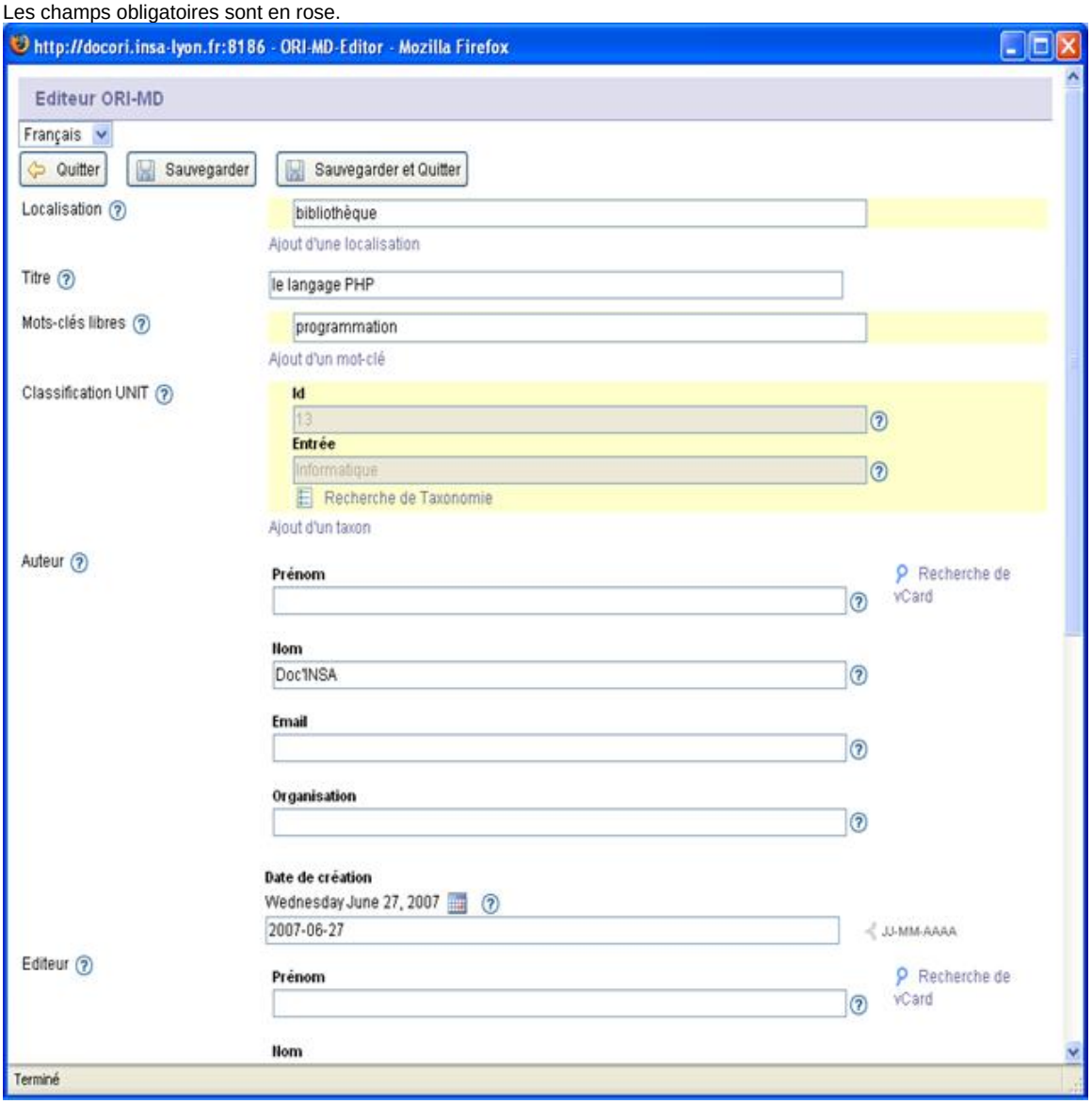
Sous Firefox : **Firefox a empêché ce site d'ouvrir une fenêtre popup**
Cliquez sur **Options** (dans le petit bandeau jaune), puis sur **Autoriser les popup pour [HOST_INSTALL]**





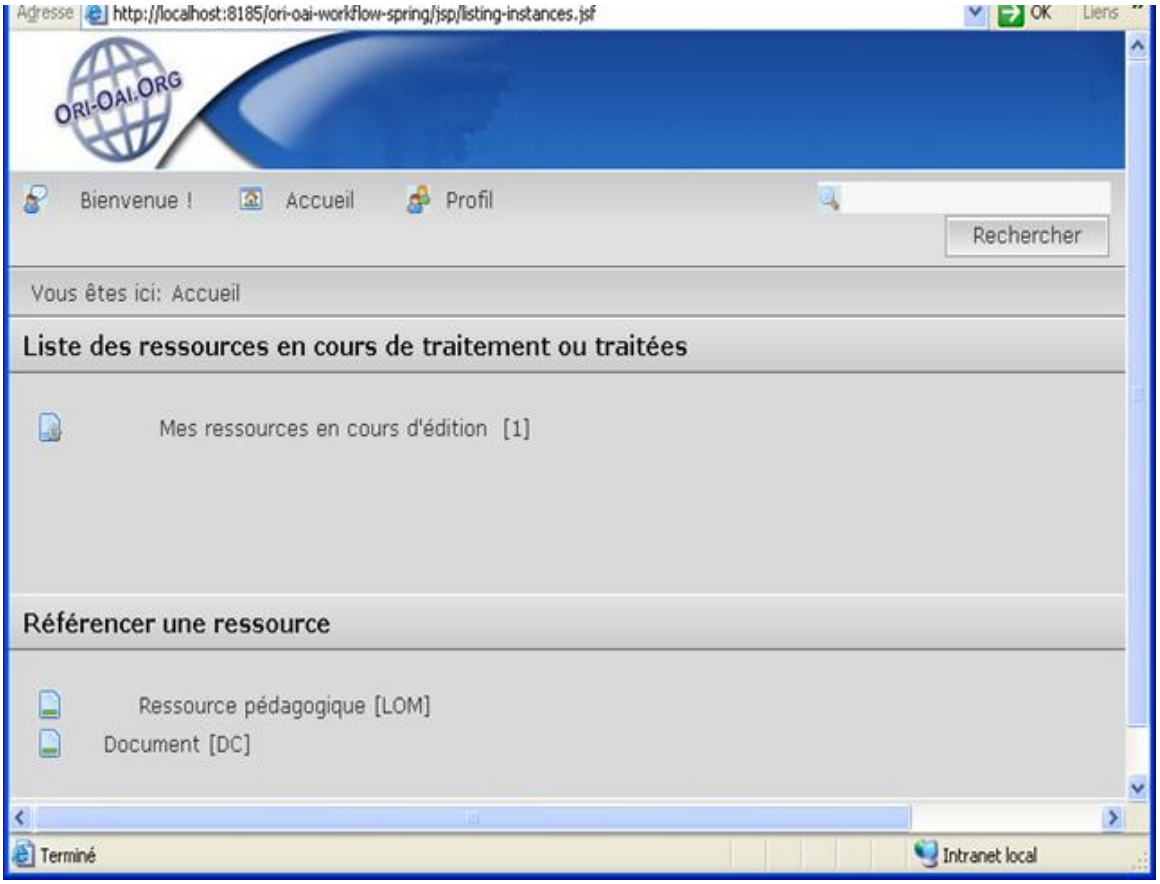
Remplissez le formulaire, lorsque celui ci **est rempli et validé** (le message d'erreur en haut du formulaire a disparu), cliquez sur « **Sauvegarder** ».

Les champs obligatoires sont en rose.

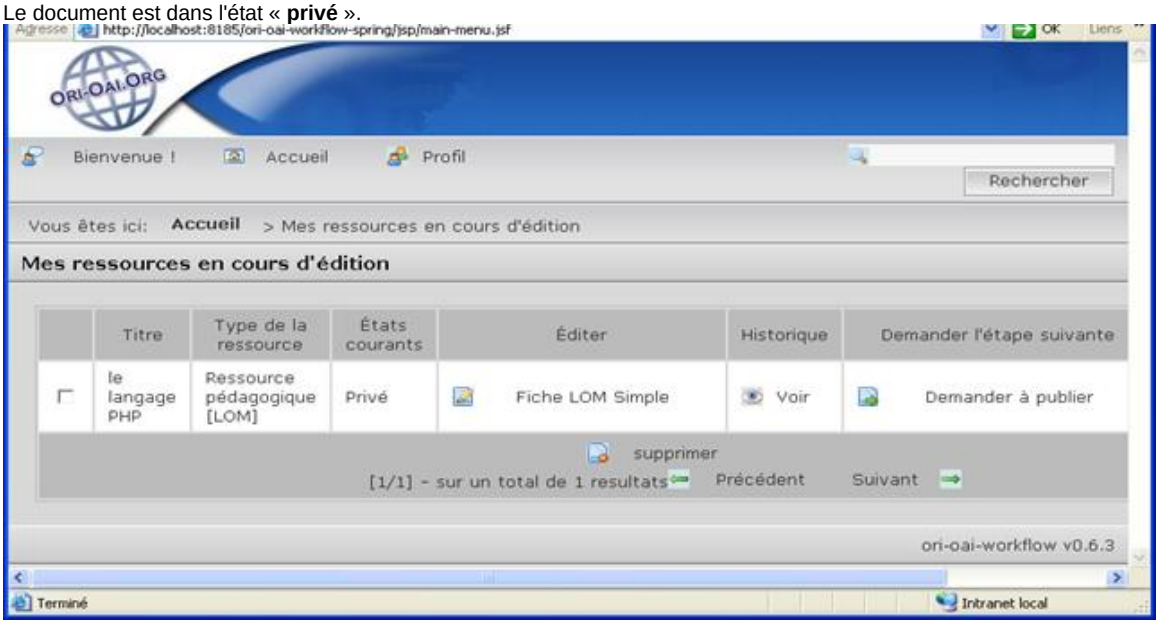


Vous pourrez revenir sur ce formulaire ultérieurement si vous le souhaitez.

Pour Demander la publication des métadonnées que vous avez saisies, cliquez sur **Mes ressources en cours d'édition [1]**.

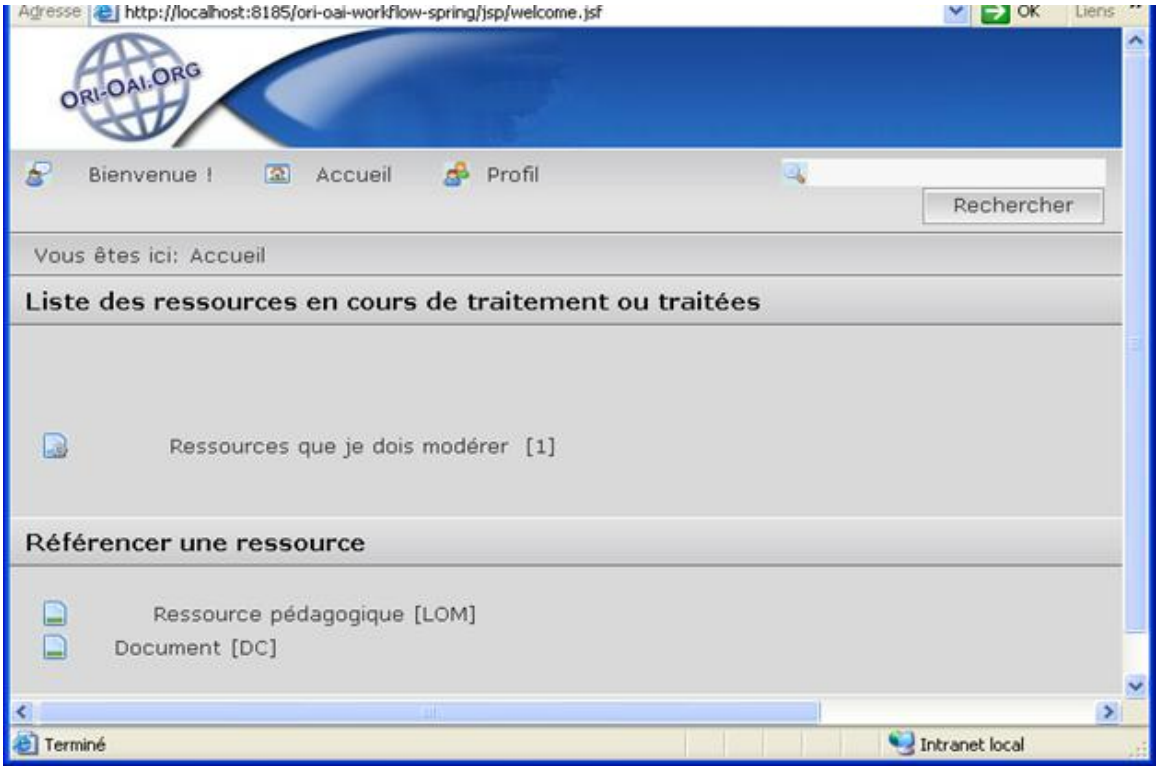


Le document est dans l'état « **privé** ».

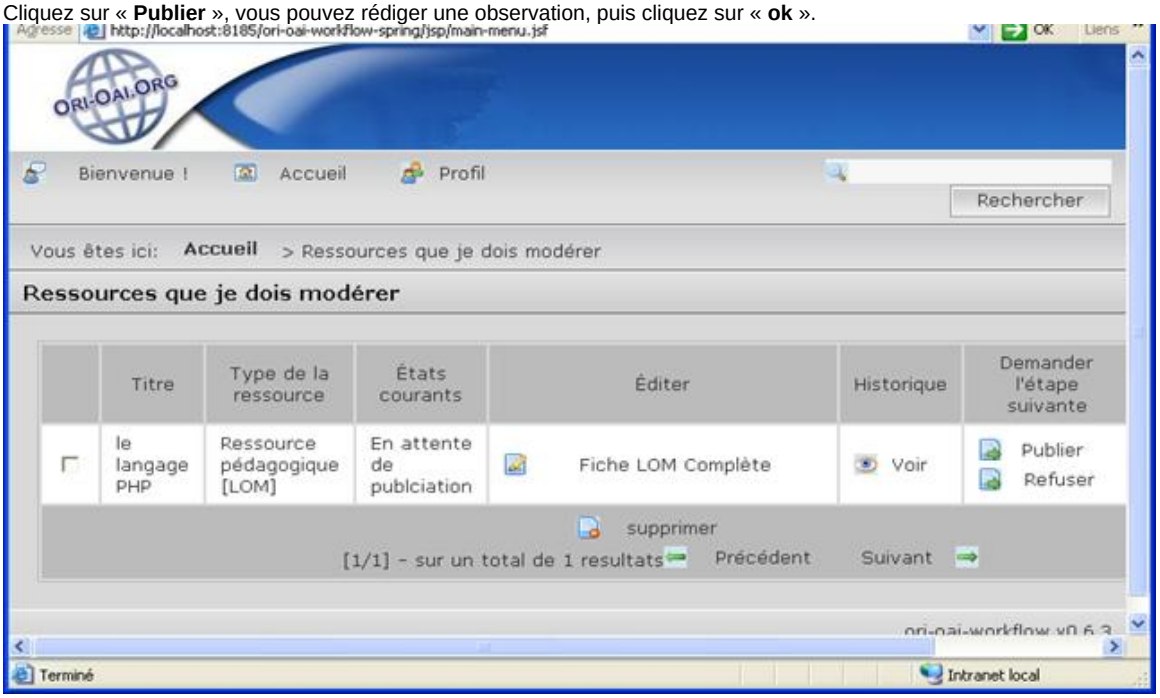


Pour passer à l'état « **en attente de publication** », cliquez sur « **demander à publier** ». Vous pouvez rédiger une observation, puis cliquez sur « **ok** ».
Déconnectez-vous de l'utilisateur ayant le rôle Auteur (clic sur Bienvenue, puis Quitter) et connectez-vous avec un utilisateur ayant le rôle **Modérateur** (**login_moderateur**).

Pour publier le document, cliquez sur « **Accueil** » puis sur « **Ressources que je dois modérer [1]** ».



Cliquez sur « **Publier** », vous pouvez rédiger une observation, puis cliquez sur « **ok** ».



Vos données sont maintenant publiées.

Captures d'écran