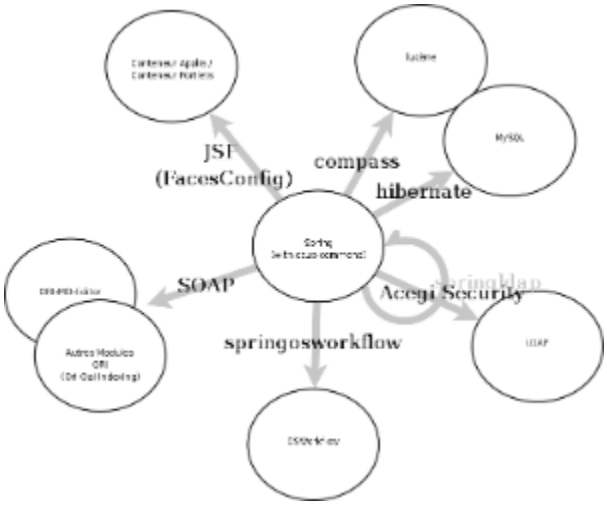
Workflow - Implémentation

Détails de l'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.

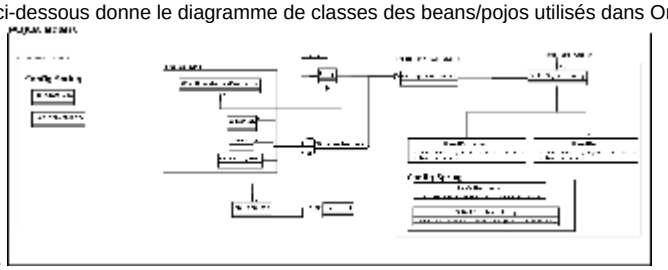
Acegi Security communique avec les sources de données utilisateurs/groupes pour l'authentification ainsi que comme base utilisateurs pour l'implémentation de RBAC (Role Based Access Control).

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)

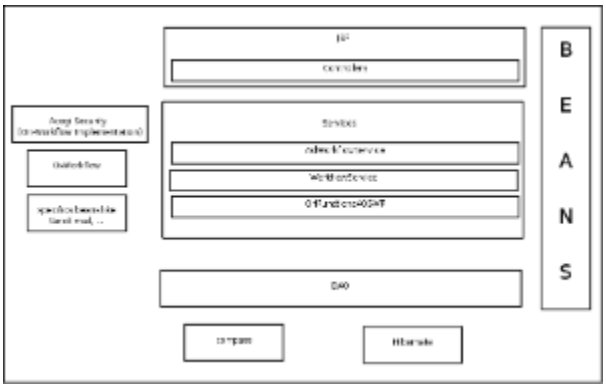
- Il est possible d'ajouter/supprimer les masks de permissions et rôles disponibles dans l'application
- Une api permet de supprimer, ajouter vérifier un rôle ou une permission sur un objet de type OriAclObjectIdentity (cette api est notamment utilisée via les fonctions/conditions paramétrées dans les 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.