TP OSGi 2

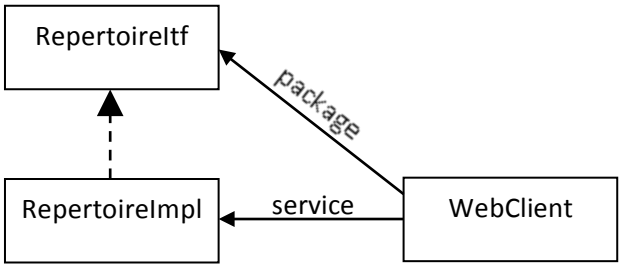
Outils à utiliser pour le TP

Dans cette seconde partie, vous utiliserez un IDE et le plugin Maven-Bundle (http://felix.apache.org/site/apache-felix-maven-bundle-plugin-bnd.html) de Felix. Ce plugin est une surcouche pour Maven de BND (http://www.aqute.biz/Bnd/Bnd).

Si ce n'est pas déjà fait, il est conseillé de vous familiariser avec ce plugin en mettant en place un premier bundle de log comme expliqué ici : http://felix.apache.org/site/apache-felix-maven-bundle-plugin-bnd.html#ApacheFelixMavenBundlePlugin%2528BND%2529-howto

État initial

Avant de démarrer cette seconde partie du TP, vous devez avoir terminé la première : vous devez disposer d'une version de votre application de répertoire qui tourne dans le Felix de GlassFish, avec un client Web (dans un servlet) qui interroge un (ou plusieurs) répertoire(s) en fonction d'une interface standardisée. Le client, l'interface des répertoires et leurs classes d'implémentation sont dans des bundles séparés. Le tout doit ressembler au schéma ci-dessous :



1. Création d'un bundle DAO

Dans cette partie, vous allez « sortir » votre DAO du bundle implémentant votre répertoire, de façon à ce que la persistance soit gérée dans un composant indépendant, et puisse être implémentée de différentes façons.

La première chose à faire est d'extraire les interfaces des classes d'implémentation du DAO et de la classe Personne. Structurez en packages l'ensemble des classes et interfaces qui implémentent la persistance en plaçant les interfaces dans un package et les classes d'implémentation dans un autre.

Remarque : vous risquez d'avoir un dilemme à résoudre concernant le package où placer la classe Personne : en fonction du fait que vous le considérez comme un objet métier ou de persistance, il devra aller dans un package ou dans l'autre. Dans les deux cas, vous devrez trouver une solution pour l'instanciation de personnes, puisque le répertoire et le DAO ont tous deux besoin de créer des personnes. Vous pouvez utiliser des patterns factory et / ou DTO pour résoudre ces problèmes…

1.1. Extraction des interfaces

Réalisez un premier bundle ne contenant que les interfaces et faites en sorte qu'il exporte le package réalisé. Générez le manifest et exportez l'ensemble dans un jar. Déployez-le dans GlassFish.

1.2. Réalisation du composant

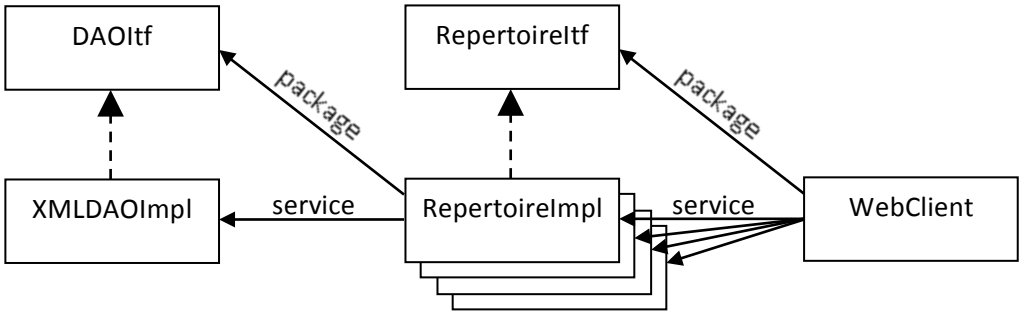
Faites de même pour le package contenant les classes d'implémentation du DAO (en XML). Générez le manifest et exportez l'ensemble dans un jar. Déployez-le dans GlassFish.

1.3. Modification du répertoire

Modifiez votre bundle répertoire pour qu'il utilise les deux bundles précédents à la place de ses classes internes (supprimez ces classes). Pour cela, il récupèrera les interfaces et les instances de Personne et de PersonneDAO. Il exposera des dépendances envers les interfaces pour importer les types des objets et envers le les classes d'implémentation pour consommer le service.

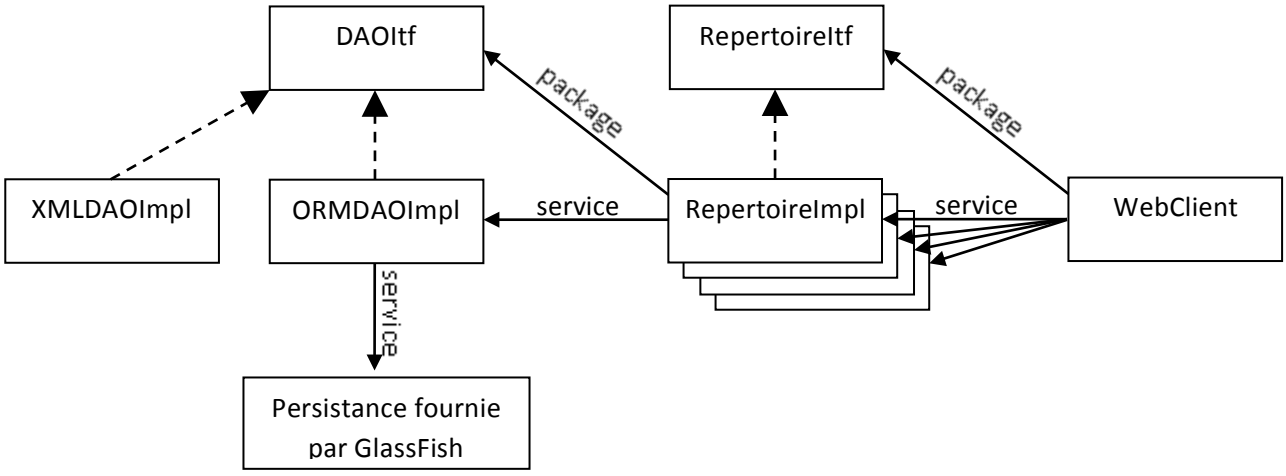
1.4. Création de plusieurs répertoires

Si vous ne l'avez pas déjà fait, dissociez les différentes fonctions du répertoire et déployez-les dans des bundles différents. Donnez à ces bundles-répertoires des noms différents pour permettre au client de les interroger. Votre référentiel de dépendances doit maintenant ressembler à cela :



2. Création d'un second DAO

Reprenez votre travail du TP EJB pour réaliser un bundle qui implémente la persistance à l'aide d'une BD et d'un ORM. Utilisez le support de persistance (EclipseLink) fourni par GlassFish[1]. Packagez le tout dans un nouveau bundle qui possède le même nom que le précédent mais un numéro de version supérieur. Déployez-le dans GlassFish, et testez à l'aide du client si la nouvelle version a effectivement remplacé la précédente. Vous devez maintenant avoir un référentiel qui correspond au schéma suivant :



[1] *c.f.* http://stackoverflow.com/questions/7879827/osgi-jpapostgresql. Il peut être nécessaire d'ajouter <property name="eclipselink.target-server" value="SunAS9"/> au persistence.xml

3. Création de répertoires personnalisés

Vous allez maintenant modifier votre application pour que chaque répertoire ait un identifiant et que cet identifiant soit utilisé dans le nom des bundles DAO. Pour cela, modifiez le client pour qu'il permette de saisir un champ supplémentaire « nom du répertoire », et mettez en place le comportement suivant :

- S'il existe un bundle correspondant au nom demandé, il est utilisé par le répertoire
- Si le bundle correspondant n'existe pas, un DAO est créé et enregistré avec un nom adéquat. Il est ensuite utilisé normalement.