Choix d'une API XML-RPC

Le java-SDK de Sun ne propose, pour utiliser XML-RPC, que les interfaces (sans implémentation concrète) necessaires. P lusieurs projets ont été menés pour réaliser ces implémentations, la plupart par des particuliers (projets "perso"). L e groupe de développement Apache (Apache Software Foundation) propose une de ces API.

Cette API supporte les connections sécurisées (SSL ici), propose des types java simples (http://ws.apache.org/xmlrpc/types.html), et bénéficie du sérieux du groupe de développement Apache. Elle est (relativement) bien documentée par une javadoc. Le principe d'utilisation de cette API en tant que client RPC est le suivant :

-Création d'un client (url des webservices en paramètre)
-Construction des paramètres d'une requête (dans un Vector)
-Appel de la requête (avec son nom, e t les paramètres précedemment crées)

L'appel de la requête retourne un Object directement exploitable, ou null si la requête à échoué. L 'ensembles des fonctions RPC accepte l'encodage des caractères Unicode (UTF-8) de façon transparente pour le programmeur.

Java et SSL

Les webservices d'EasyKM utilisent une connection sécurisée SSL. Pour que le client java (au travers de l'API XML-RPC) puisse les utiliser, sa machine virtuelle doit avoir accès au certificat SSL correspondant à cette connection. La machine virtuelle utilise pour cela un fichier "keystore", protégé par mot de passe, d ans lequel on importe des certificats. Les différents certificats d'un keystore son identifiés par des alias. Pour gérer les keystore on utilise l'utilitaire "keytool" fourni dans le java-SDK. La commande pour importer le certificat "cacert.crtf" sous l'alias "labo" dans le fichier keystore "kslabo" est :

```
keytool -import -alias labo -file cacert.crtf -keystore kslabo
```

L'utilitaire "keytool" demande le mot de passe du keystore à chaque commande demandée. Si le fichier keystore n'existait pas, il est créé. Ce fichier, ainsi que son mot de passe, doit être accessible par le client lors de son execution.

L'API XML-RPC utilise les paramètres SSL par défaut de la machine virtuelle pour créer des connections sécurisées. Il faut donc, à l'initialisation du client java, spécifier le fichier keystore (et son mot de passe) en tant que paramètre SSL par défaut. Le processus de connection sécurisée est ensuite tranparent, et on peut directement utiliser XML-RPC.

Découpage du projet/planning

Prototype 1 :

- Login. (1 jour)
  - Boîte de dialogue de login, avec sauvegarde de l'identifiant durant toute la session.
  - Fermeture de session/déconnection.
- Navigation via arborescence thématique. (4-5 jours)
  - Composant graphique arbre (JTree) à mettre en oeuvre.
  - Méthodes RPC getTopicMatrix et getDocsInTopic.
  - Optimisation des requêtes au webservice/gestion de cache.
  - Gestion d'onglets (complétée plus tard par les onglets "Nouveautés" et "Résultat de recherche")

06/04

- Affichage détaillé d'un document. (1-2 jours)
  - Panneau graphique à créer et mettre en oeuvre.
  - Gestion du transfert de fichier (download en HTTP).
    - JFileChooser.
  - Gestion d'onglets et de documents multiples.
- Consultation nouveautés. (1-2 jours)
  - Composant graphique pour afficher les N nouveautés (JList?).
  - Méthode RPC getLastDocs.
  - Boîte de dialogue d'options du logiciel. (choisir N, ainsi que d'autres options à définir).

11/04

- Recherche rapide. (3-4 jours)
  - Méthode RPC search.
  - Composant graphique correspondant (JToolbar flottante?).
  - Resultat de recherche par arborescence thématique (JTree).

Prototype 2 :

18/04

- Dépôt de documents (2-3 jours)
  - Panneau graphique de création/modification de document à créer et mettre en oeuvre.
  - Méthode RPC registerDoc.
  - Gestion du transfert de fichier (upload en FTP).
    - API FTP à mettre en oeuvre.
    - JFileChooser.
- Recherche avancée. (1 jour)
  - Panneau graphique correspondant.
  - (réutilisation de "résultat de la recherche")
- Destruction et édition de documents. (1-2 jours)
  - Méthodes RPC updateDoc et deleteDocs.

25/04

- Evaluation des documents. (1 jour)
  - Panneau graphique correspondant.
  - Méthode RPC rateDoc.

Architecture :

De manière générale, un composant graphique (une/des classe(s)) est crée pour chaque fonctionnalité. Toutes les méthodes XML- RPC sont groupées dans la même classe. Les objets utilisés par plusieurs composants (Topic, Document, Résultat de recherche) on chacun leur propre classe. Les méthodes de traitement associées à un composant graphique sont intégrées dans la/les classe(s) de ce composant.

Les composants graphiques :

- -Fenêtre principale avec sa barre de menu
  - -Barre d'outils (raccourcis des menus)
  - -Barre d'outil "Recherche rapide"
  - -Séparation droite/gauche (SplitPane)
    - -A droite : onglets d'affichage détaillé des documents (un onglet par document affiché)
    - -A gauche : 3 onglets :
      - -Arborescence thématique complète (Arbre)
      - -Liste des N nouveautés (Liste)
      - -Résultat de la recherche (sous la forme d'une arborescence thématique réduite)
- -Boîte de dialogue "Sas d'identification" (Login)
- -Boîte de dialogue "Recherche avancée"
- -Boîte de dialogue "Evaluation"
- -Boîte de dialogue "Demande en cours..." (please wait)

Les Threads (sous-processus légers) possèdent leurs propres classes :

- -Récupération de document (getTopicById) en tâche de fond.
- -Gestion du cache des documents.