

Open source CeCILL

MDweb

**Outil de catalogage et de localisation de
l'information environnementale**

Note technique sur le serveur Z39.50 pour MDweb

Version	Commentaires
1	Auteurs : Jean Pinaud (jean.pinaud@medias.cnes.fr) Date de création : 15/01/2007

Table des matières

1	Configuration du serveur.....	3
1.1	Comportement du serveur.....	3
1.2	Configuration de la traduction z39.50 -> base de données	3
1.3	Configuration du remplacement des chaînes	3
2	Utilisation de JzKit 1.3	3
2.1	Les classes à implémenter.....	3
2.2	Démarrer le serveur	3
2.2.1	Description des classes.....	3
	Lancement.....	4
	ServerProperties.....	4
2.2.2	Le lancement	4
2.2.3	Les erreurs possibles	4
3	Réception des requêtes et interrogation de la base	5
3.1	Traduction de la requête.....	5
3.1.1	Description des classes.....	5
	ComplexNode	6
	AttrPlusNode	7
	QueryDecoder	7
	QueryDecoderSQL.....	7
3.1.2	La traduction	8
3.2	Modification de la requête.....	8
3.2.1	Description des classes.....	8
	TextEntryModifier	8
3.2.2	Modification des chaînes.....	8
	Suppression des points de complétion	8
	Suppression des articles	8
	Suppression de la marque du pluriel.....	8
	Suppression des apostrophes.....	8
	Passage en minuscule.....	9
4	Envoi de la réponse	9
4.1	Description des classes.....	9
	ServicePersistance.....	10
	HibernateSGBD.....	10
4.2	Génération et envoi de la réponse.....	10
4.2.1	Encodage de la réponse	10
4.2.2	Ajout d'information	10
4.2.3	Envoi de la réponse	10
4.3	Schéma récapitulatif	10
5	Gestion des erreurs	11
5.1	Erreurs de la base de données	11
5.2	Erreurs du client	12
6	Limite du serveur.....	12

1 Configuration du serveur

1.1 Comportement du serveur

La configuration du serveur mdweb utilisant le protocole z39.50 se fait à l'aide d'un fichier properties situé à l'emplacement suivant : \$path/properties/serveur.properties.

Les deux éléments importants qui peuvent être modifiés régulièrement sont le port d'écoute du serveur, et le délai durant laquelle une connexion reste active entre deux échanges client/serveur. Ces informations sont contenus dans les variables port et timeout du fichier de configuration. Port est un entier supérieur à 1024 et timeout un entier dont l'unité est la milliseconde.

```
# Configuration de jZkit
port=7777 #port d'écoute
timeout=500000 #durée en milliseconde, ici un peu plus de 8 minutes
```

1.2 Configuration de la traduction z39.50 -> base de données

Pour chaque code z39.50 du profil geo demandé par l'OGC, un mapping vers un type de donnée et un attribut des objets d'accès à la base doit être réalisé. Ce mapping se réalise à l'aide du fichier de configuration, et la saisie se présente sous la forme :

```
codeZ3950=attribut:type
```

Les types supportés sont :

- numeric : pour tous nombre réel ou entier
- date : pour une date comportant un mois, un jour et une année (testé pour la forme YYYYMMDD)
- text : toutes autres données

En plus des attributs pouvant être manipulés il est nécessaire de gérer le type des opérateurs de comparaison pouvant être utilisés. La configuration est de la forme :

```
codeZ39.50OP_type=opérateur
```

1.3 Configuration du remplacement des chaînes

Le fichier de configuration permet de définir quel type de chaîne de caractères seront ignorées lors de l'analyse de la requête. Cela permet par exemple de supprimer les articles ou les marques du pluriel.

Il y a trois façon différente de parcourir la requête pour supprimer les éléments devant être ignoré.

- Supprimer les éléments isolés : typiquement les articles
- Supprimer les suffixes : les marques du pluriel (attention à l'ordre, les suffixes sont supprimés dans l'ordre où ils sont notés dans le fichier de configuration)
- Supprimer les préfixes : les articles avec apostrophe

2 Utilisation de JzKit 1.3

2.1 Les classes à implémenter

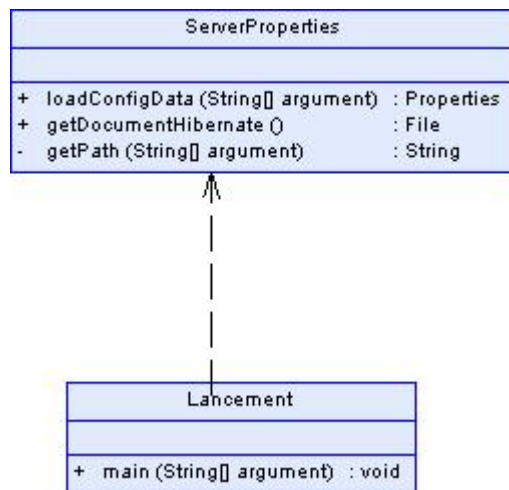
Il y a deux classes à implémenter pour utiliser la librairie JzKit :

- Searchable qui permet d'initialiser la recherche et qui instancie la deuxième classe à implémenter
- SearchTask qui est appelée par le serveur pour effectuer toutes les opérations sur la base de donnée (recherche et présentation des données)

Pour de plus ample informations il convient de se référer à la doc de JzKit 1.3 qui décrit l'ensemble des fonctions.

2.2 Démarrer le serveur

2.2.1 Description des classes



1. Classe d'initialisation du serveur

Lancement

Cette classe permet le lancement du serveur, elle contient la méthode statique main qui permet de démarrer l'application. Elle va charger le fichier de configuration du serveur et le transmettre à une instance de la classe ZServeur implémentée par la librairie JzKit. C'est la classe ZServeur qui se chargera par la suite de la gestion de la connexion des utilisateurs.

Main

- rôle
démarrer le serveur z39.50
- paramètres
Tableau de chaîne de caractères : contient l'emplacement du fichier de configuration (si non précisé par la variable -DCONFIG_PATH)

ServerProperties

Le principe de cette classe est de pouvoir rendre disponible le fichier de configuration du serveur à n'importe quelle instance manipulée par l'application. Elle est par exemple appelée par Lancement qui l'utilise pour charger les propriétés de démarrage du serveur.

LoadConfigData

- rôle
Récupère le fichier de propriétés de l'application
- paramètres
argument qui contient l'emplacement du fichier de configuration (non obligatoire normalement utilise -DCONFIG_PATH)
- retour
Une instance de Properties contenant les informations de configuration du serveur

getDocumentHibernate

- rôle
Retourne le fichier de configuration d'hibernate (instance de File)

2.2.2 Le lancement

Par défaut le fichier .jar du serveur exécutera le main de la classe Lancement. Cependant il est nécessaire de lui passer en paramètre le chemin d'accès aux fichiers de configuration. Ce passage de paramètres se fait par la variable d'environnement -DCONFIG_PATH. La commande pour démarrer le serveur sera donc de la forme : java -DCONFIG_PATH=chemin-jar serveur.jar.

2.2.3 Les erreurs possibles

Il y a deux erreurs principales pouvant apparaître au démarrage du serveur. La première est de type « file not

found » et apparaît si le programme n'arrive pas à trouver le fichier serveur.properties dans le dossier passé en paramètre (cf Le Lancement). Il faut donc contrôler que le fichier est effectivement dans le dossier spécifié et qu'il porte le bon nom. La seconde erreur vient du choix des ports d'écoute. Un message informera l'utilisateur que le port est déjà utilisé par une autre application, il faut donc modifier le fichier serveur.properties en mettant un nouveau port.

3 Réception des requêtes et interrogation de la base

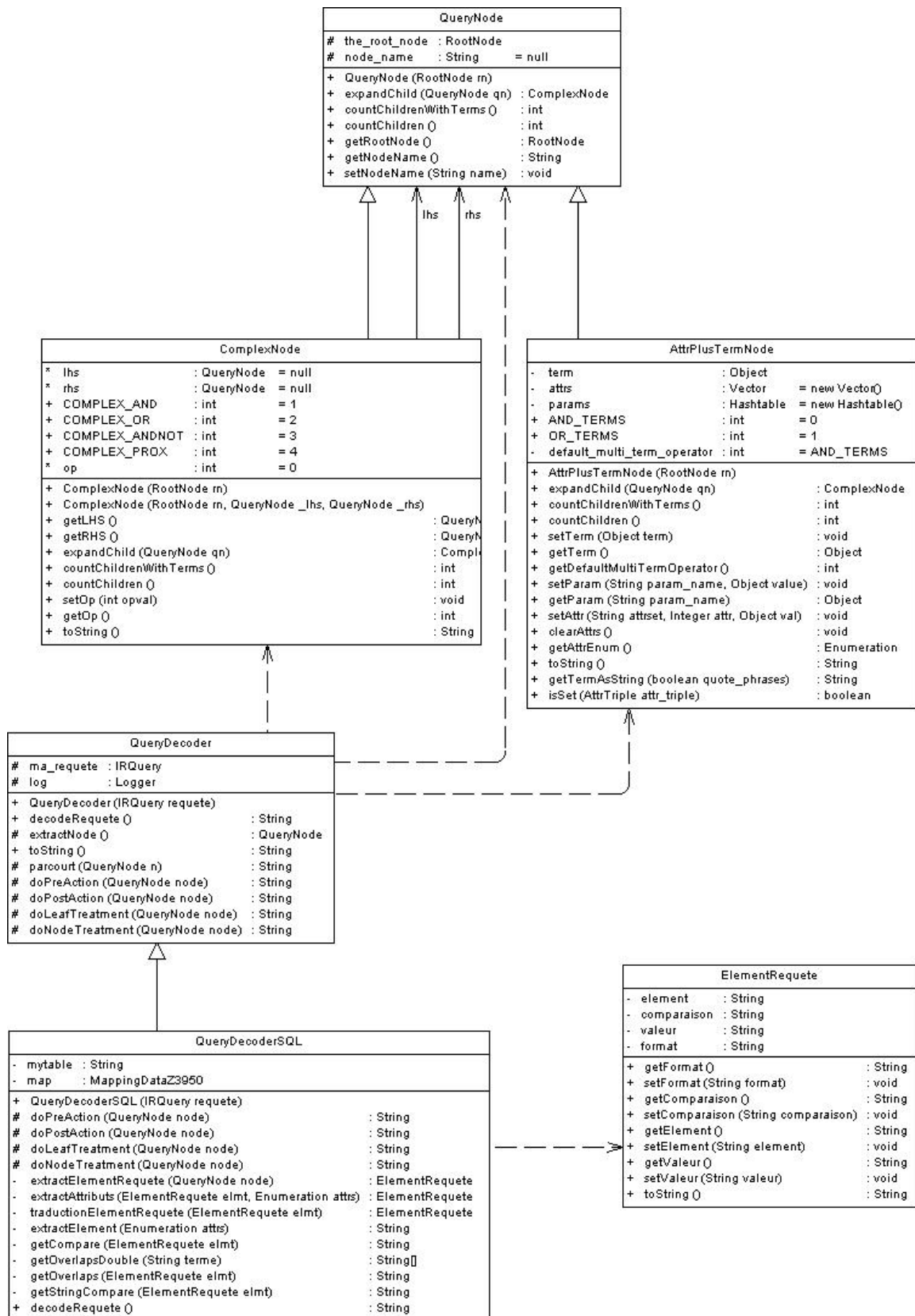
La requête est donc transmise à la classe MDSearchTask par l'appel de la fonction evaluate(). La requête au format RPN (cf schéma arbre de requête) va être traduite en requête SQL qui sera transmise à la base de données. La réponse de la base de données est une liste de chaîne de caractères représentant les documents xml. Ces chaînes vont être traduites en objet de la classe Document qui seront retournés au noyau du serveur qui se chargera de leur envoi au client. Une étape supplémentaire a été ajoutée au niveau de la traduction pour ajouter des informations au document xml retourné.

3.1 Traduction de la requête

La requête reçue est donc sous la forme d'un arbre. Cet arbre est constitué de trois types d'objet. Deux types d'objets sont les instances de classes implémentant la classe abstraite QueryNode, et le dernier correspond au noeud racine de l'arbre.

Les classes implémentant QueryNode sont : ComplexNode et AttrPlusTermNode.

3.1.1 Description des classes



2. Module de traduction

ComplexNode

Cette classe correspond à un noeud de l'arbre possédant deux fils qui seront forcément des QueryNode. Ces fils

sont un fils gauche et un fils droit qui sont accessible avec les fonctions getLHS et getRHS.

Au niveau de la requête, ce type de noeud contient les opérateurs logiques de type AND, OR, AND NOT, etc... Cet opérateur est obtenue grâce à la fonction getOp.

AttrPlusNode

Cette classe correspond aux feuilles de l'arbre de requête. Elle contient donc l'ensemble des éléments permettant d'interroger le serveur sur un attribut précis. Cette classe contient donc entre autres : le code de l'attribut à manipuler, le profil utilisé, la valeur de l'attribut et le comparateur utilisé.

QueryDecoder

Il s'agit d'une classe abstraite permettant de parcourir un arbre en utilisant l'algorithme de parcourt en profondeur. Les fonctions implémentées assurent le parcourt dans l'arbre et seul les traitements sont à implémenter dans les classes héritant de celle-ci.

Constructeur

- paramètres
La requête transmise par le client au format RPN sous forme d'arbre

decodeRequete

- rôle
Parcourt l'arbre de la requête et le transforme en chaîne de caractères. Le format de la chaîne dépend de l'implémentation choisie par la classe fille. Ici la classe QueryDecoderSQL retournera une chaîne représentant la clause Where d'une requête SQL.
- Retour
La chaîne de caractères représentant l'arbre de requête

QueryDecoderSQL

Cette classe implémente la classe abstraite QueryDecoder et assure le passage de l'arbre RPN à la chaîne de caractères au format SQL. Les fonctions implémentées sont les suivantes ;

doPreAction

- rôle
ajoute une parenthèse ouvrante, cette fonction est exécuté en arrivant sur un noeud
- paramètre
le noeud courant

doNodeTreatment

- rôle
Ajoute l'opérateur conditionnel du noeud courant à la chaîne de caractères. Cette fonction est appelé lorsque le fils gauche a été traité.
- Paramètre
le noeud courant

doPostAction

- rôle
Ajoute une parenthèse fermante, cette fonction est exécuté lorsque le fils droit a été traité.
- Paramètre
le noeud courant

doLeafTreatment

- rôle
Ajouter les informations de la feuille courante à la chaîne de caractères représentant la requête. Cette fonction décompose la classe AttrPlusTermNode pour en extraire les informations utiles qui seront placés dans la classe ElementRequete pour être traité (traduction z39.50->SQL, modification des chaînes de caractères, etc...). C'est cette classe qui pourra lever une exception si le client a utilisé une opération ou un attribut non valide.
- Paramètre
le noeud courant

3.1.2 La traduction

Elle se fait par un parcourt de l'arbre en profondeur (i.e. On descend le plus bas possible sur une branche avant de passer à la suivante). Lorsqu'une feuille est trouvée ses données sont converties en condition SQL. Lorsqu'un noeud est trouvé après le traitement d'une feuille (lors de la remonté) la valeur de l'opérateur logique du noeud est ajouté à la condition SQL. Ainsi, en un seul parcourt de l'arbre la requête SQL est formée.

3.2 Modification de la requête

La traduction de la requête du format RPN vers le format SQL demande de modifier certains éléments après les avoir extrait. En effet, il est utile de s'assurer que le client a envoyé une requête pouvant interroger un champ de données le plus large possible. C'est pour cette raison que les chaînes de caractères sont modifiées quand elles sont extraites. Ces modifications sont réalisées par la classe TextEntryModifier du package métier.

3.2.1 Description des classes

La classe suivante est utilisée pour modifier les chaînes de caractères reçues :

TextEntryModifier

Cette classe va donc modifier les chaînes de caractères qui lui sont transmises. Pour cela elle appelle une série de fonction dans un ordre précis pour éliminer dans l'ordre : les points en début et fin de phrase, les articles, la marque du pluriel, les apostrophes et passer la chaîne en minuscule.

Constructeur

- paramètres
La configuration du serveur sous la forme d'une instance Properties

textModification

- rôle
Modifier la chaîne de caractères qui lui ai passé
- paramètres
La chaîne de caractères à modifier
- retour
La chaîne de caractères modifiée

3.2.2 Modification des chaînes

Suppression des points de complétion

Les clients z39.50 peuvent envoyer des valeurs non complètes, par exemple pour trouver tous les mots contenant une chaîne précise. Cette recherche est initiée en ajoutant des points en début et fin du mot pour indiquer au serveur que ce mot n'est pas complet. Comme notre serveur complète automatiquement tous les mots qu'il reçoit, il peut ignorer ces points. Quand de tel point existe il supprime donc le premier et le dernier caractère de la chaîne.

Suppression des articles

Pour retrouver les articles le programme cherche le motif représentant les articles entouré d'espace. Chaque article découvert est supprimé et remplacé par le symbole \$ reconnu par SQL.

Suppression de la marque du pluriel

La suppression de la marque du pluriel se fait en comparant les motifs contenus dans le fichier de configuration à la fin de chaque mot de la chaîne de caractères. Ces motifs sont comparés un par un et dès que l'un d'eux correspond la chaîne est modifiée (le motif dans la chaîne est remplacé par le symbole \$ utilisé en SQL).

Suppression des apostrophes

Le motif des apostrophes est situé en début de mot, de la même façon que pour les autres modifications ils sont remplacés par le symbole \$.

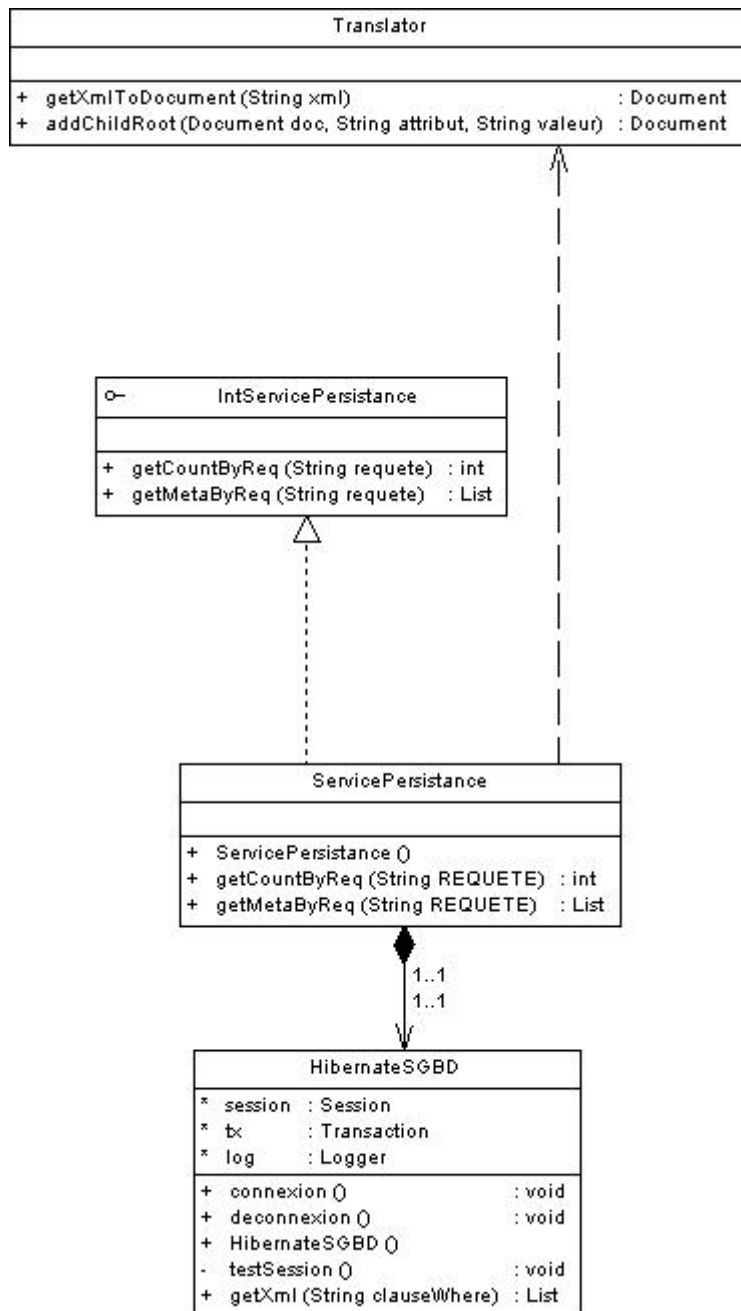
Passage en minuscule

Afin d'éviter tout problème de reconnaissance minuscule/majuscule les chaînes de caractères passées au serveur seront toujours interprétées comme des minuscules. De la même façon les chaînes de caractères situées dans la base de données sont converties en minuscules au moment de la recherche. Cela est transparent pour le client qui recevra les réponses dans la casse normale.

4 Envoi de la réponse

Une fois que la requête a été décodée, le serveur interroge la base de données et va encoder la réponse dans une instance de la classe Document (classe représentant un document xml en java). A cette représentation du document xml est ajouté l'identifiant de la fiche et le nom du gabarit utilisé. La création des instances de Document et les ajouts sont réalisées par la classe Translator du packages helpers du serveur. Cette classe est instanciée par le service de persistance lorsqu'il a obtenu la réponse de la base de données.

4.1 Description des classes



3. Persistence

ServicePersistence

Son rôle est de servir d'interface entre SearchTask et la base de données. ServicePersistence a en charge la connexion au sgbd et son interrogation. Il encode également la réponse à retourner au serveur.

getCountByReq

- rôle
retourne le nombre de fiche xml dans la base de données répondant aux critères de recherche.

getMetaByReq

- rôle
obtenir les fiches xml répondant aux critères de recherche
- retour
une liste de Document xml

HibernateSGBD

Cette classe manipule la librairie Hibernate. Elle implémente la connexion à la base, son interrogation et la construction de la réponse pour le service de persistance.

connexion

- rôle
assure la connexion à la base de donnée

deconnexion

- rôle
déconnecte le serveur de la base de données

getXml

- rôle
interroge la base et crée la liste de fiche xml
- paramètre
la clause Where générée lors de la traduction de l'arbre RPN
- retour
une liste de chaîne de caractères représentant les fiches correspondant aux critères de recherche.

4.2 Génération et envoi de la réponse

4.2.1 Encodage de la réponse

L'encodage de la chaîne de caractères représentant le document xml se fait à l'aide de classes du package avax.xml.parsers. L'élément important à noter est qu'ici est choisi l'encodage du jeu de caractère en iso-8859-1. Cet encodage apparaît en en-tête de fichier xml et il est utilisé par les clients pour lire et modifier le document xml.

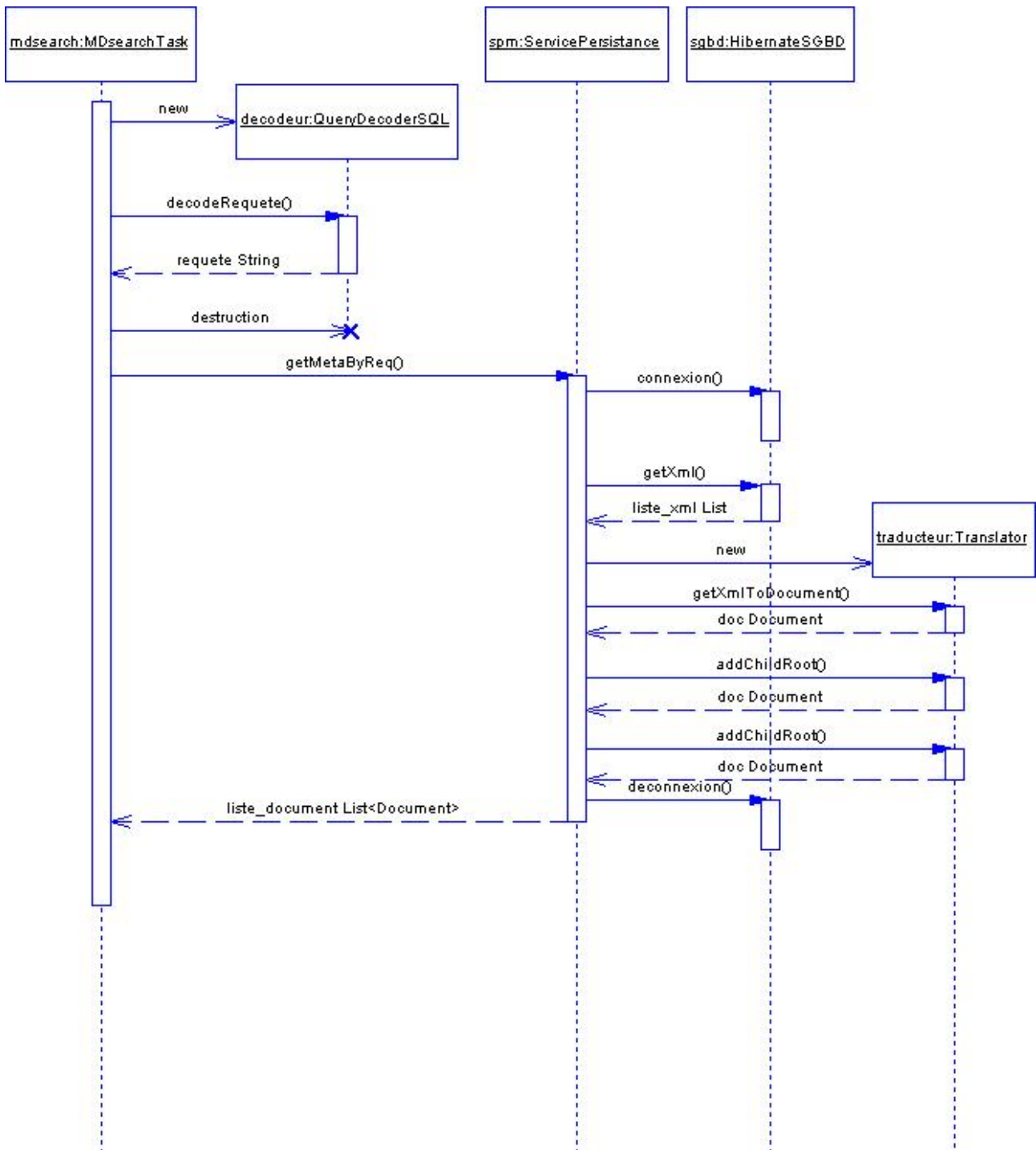
4.2.2 Ajout d'information

La classe de traduction chaîne de caractères --> document xml permet également l'ajout d'information dans le document qui sera retourné au client. Ainsi il est possible d'ajouter à la racine du document xml l'identifiant de la fiche ainsi que le nom du gabarit utilisé.

4.2.3 Envoi de la réponse

Une fois que les dernières modifications ont été réalisées la fiche est transmise au serveur z39.50 qui va encoder la réponse au format reconnu par le client en encodant les caractères au format iso-8859-1. Il est possible de changer le format de retour en modifiant le code présent dans la classe ZServerAssociation de jzkit et en remplaçant ISO-8859-1 aux lignes 1004 et 1019 par le format désiré.

4.3 Schéma récapitulatif



4. Diagramme de séquence de l'interrogation du serveur

5 Gestion des erreurs

Lorsque le client se connecte au serveur il faut qu'il soit informé correctement des problèmes rencontrés en cas d'erreur. Cela permet d'éviter de déconnecter un client sans lui en donner la raison et peut également l'informer d'une erreur de sa part.

5.1 Erreurs de la base de données

De nombreuses erreurs peuvent survenir au niveau de la base de donnée. Que ce soit un problème de connexion, une requête mal formée ou autre chose, la librairie assurant le dialogue entre le serveur et le système de données va générer une exception. Cette exception est capturée au niveau de la classe HibernateSGBD qui générera

une exception pouvant être interprétées par la librairie jzkit qui elle même retournera l'erreur au client dans le format Bib-1.

5.2 Erreurs du client

Les erreurs provenant du client peuvent être de plusieurs types, les principales sont une requête mal formée ou une erreur de code d'attribut. Dans le premier cas le serveur lui-même informera le client de son erreur. Dans le second cas, lors de la traduction de la requête, une exception sera levée et l'erreur attribut non défini retournée au client.

6 Limite du serveur

Le serveur implémenté ne retournera que des fiches au format xml quelque soit la demande du client. Cela est dû à des problèmes d'implémentation de la version de JzKit utilisée (certaines valeurs sont initialisées et ne sont plus modifiées alors qu'elles devraient servir à contrôler le format demandé par le client).

Le serveur ne gère pas non plus les expressions régulières car il doit traiter des demandes provenant de client manipulé par des utilisateurs humains qui n'ont pas à manipuler ce type de langage.

Le dernier point est que le serveur ne permet pas à un utilisateur de demander un tri sur une donnée. Cela est dû au fait que le client php utilisé perd la session avec le serveur (on ne peut pas conserver une ressource dans une session en php). Si un tri était effectué alors qu'une donnée a été ajoutée pendant la recherche du client, un décalage aurait lieu et le client pourrait obtenir une donnée qu'il n'a pas demandé (les fiches sont demandées en fonction de leur position). Le serveur tri les données par date d'ajout, si une fiche est ajoutée elle sera en fin de liste et cela ne modifiera pas la fiche que le client demande.

Contacts



Institut de recherche
pour le développement

IRD / US ESPACE (US 140)

500, rue Jean François Breton, 34093 Montpellier Cedex 05

TEL : +33 (0)4 67 54 87 02

J.C Desconnets jcd@teledetection.fr

Site du projet MDweb : www.mdweb-project.org

Démonstrateur en ligne : demo.mdweb-project.org