Documentation SDX version 1.1

Auteur : Martin Sévigny, *AJLSM* [sevigny@ajlsm.com]

Date : 2 novembre 2001

Introduction

Ce site Web constitue la documentation de la plate-forme SDX. Cette documentation vise à faciliter l'installation de SDX, mais surtout le développement d'applications avec cette plate-forme.

Cette documentation est en constante évolution, et est incomplète pour l'instant. Elle sera mise à jour régulièrement, vous êtes donc encouragés à visiter fréquemment le **site officiel SDX** [http://sdx.culture.fr] pour consulter les dernières nouveautés. Actuellement, elle documente SDX version 1.1.

Vous êtes sollicités pour contribuer à cette documentation, en soulignant les erreurs ou omissions, en apportant des informations complémentaires, des expériences, etc. Pour l'instant, nous sommes particulièrement intéressés à amasser de l'information sur l'installation de SDX et des composantes de base. Pour tout commentaire ou contribution, veuillez contacter **Martin Sévigny** [mailto:sevigny@ajlsm.com] .

Présentation de SDX

SDX en quelques mots

SDX est un outil destiné à diffuser des bases de documents XML sur le Web. Cette définition étant relativement générale, nous allons tenter de la préciser de deux façons, d'abord d'un point de vue des utilisateurs et ensuite d'un point de vue technique.

SDX du point de vue des utilisateurs

Les utilisateurs SDX seront des personnes qui, à l'aide d'un navigateur Web comme Netscape ou Internet Explorer, consulteront une base de documents à la recherche d'informations précises ou simplement pour le plaisir.

Déjà, dans cette courte définition, plusieurs concepts importants sont apportés :

Navigateur Web

SDX est donc un outil que l'on peut accéder à l'aide d'un navigateur Web. Il s'agit donc d'un outil à *interface Web*, ou encore à *architecture Web*.

Mais en aucun cas cela signifie qu'il doit être utilisé sur le Web. En effet, d'autres environnements utilisent des interfaces et des architectures Web, comme par exemple les intranet, et on peut même monter une architecture Web sur un CD-ROM.

Ce qu'il faut retenir, c'est qu'un navigateur Web est tout ce qui est nécessaire pour consulter une base de documents SDX.

Base de documents

SDX est un outil documentaire, c'est-à-dire qu'il a pour objectif de faciliter la consultation de bases de données documentaires, où ici "documentaire" signifie à la fois des documents au complet, en texte intégral, ou encore des fiches documentaires, soit des bases de données textuelles.

Consultation, diffusion, variations

SDX est un outil de diffusion, permettant la consultation de documents existants. Il ne s'agit en aucun cas d'un outil de saisie ou de production de documents.

De plus, rien n'indique que SDX doit être utilisé avec certains types de documents et de documentation. Des bases de documents SDX peuvent être très différentes, et il peut même être difficile de savoir que c'est le même outil qui est derrière.

SDX du point de vue des développeurs

SDX possède deux caractéristiques fondamentales que tout développeur intéressé doit connaître, et les voici :

Architecture Web

SDX fonctionne dans une architecture Web, et il s'agit bien entendu d'une application dynamique. Ainsi, pour l'installer et le faire fonctionner, il faut le coupler à un serveur Web, que ce dernier soit relié à un réseau ou pas.

De la même façon, pour déployer une application SDX sur le Web, on doit pouvoir créer un site Web dynamique. Si le seul serveur Web que l'on a à notre disposition ne peut diffuser que des sites statiques, SDX ne sera d'aucune utilité.

XML

SDX a pour objectif de diffuser des documents XML. De plus, XML tient une place importante dans toutes les parties de SDX, par exemple pour les pages Web dynamiques ou pour les services distants. C'est pourquoi une bonne connaissance de XML en général est nécessaire avant de déployer SDX ou des applications SDX.

Licence

La plate-forme SDX est livrée avec une **licence GPL** [http://www.fsf.org/copyleft/gpl.html] (*GNU General Public Licence*). Essentiellement, cela signifie que SDX est un logiciel libre que l'on peut utiliser à notre guise. Toutefois, toute modification apportée à SDX doit être reversée dans le domaine public et son utilisation dans les applications commerciales comporte des restrictions.

Pour en savoir plus, vous êtes invité à lire :

- **La licence SDX** [licence.txt]
- **Une copie de la licence GPL** [gpl.txt]

Par ailleurs, SDX est livré avec un certain nombre de composantes qui sont toutes des logiciels libres. De plus, il existe des logiciels libres pour toutes les couches logicielles sous-jacentes à SDX, ce qui signifie que l'on peut monter une installation SDX qu'avec des logiciels libres.

Contacts

La plate-forme SDX a été développée initialement par la société **AJLSM** [http://www.ajlsm.com] et le **ministère de la culture et de la communication** [http://www.culture.gouv.fr] . Actuellement, ces deux organismes demeurent les principaux contacts pour toute question relative à cette application.

Pour la société AJLSM, la personne contact est **Martin Sévigny** [mailto:sevigny@ajlsm.com] . Pour le ministère de la culture et de la communication, la personne contact est **Michel Bottin** [mailto:michel.bottin@culture.gouv.fr] .

Le développement de l'application est maintenant assuré par une communauté de développeurs gérée par le site **SourceForge** [http://www.sourceforge.net] . La page d'accueil de SDX sur ce site est **http://www.sourceforge.net/projects/sdx/** [http://www.sourceforge.net/projects/sdx/] . Les personnes qui aimeraient prendre part au développement de SDX sont priées de se faire connaître via ce site.

Tous les commentaires sont les bienvenus, que ce soit pour émettre des critiques, souligner un dysfonctionnement, suggérer des améliorations, ou simplement... parce que vous aimez SDX!

Télécharger SDX

SDX est distribué selon une **licence GPL** (*GNU General Public Licence*). Il est donc libre de droit et distribué avec ses sources, mais cette licence implique certaines restrictions quant à l'utilisation dans des produits commerciaux. Une **section de la documentation** traite de la licence. De plus, SDX utilise plusieurs autres logiciels, et vous devez lire leurs licences respectives avant de l'installer et de l'utiliser.

La distribution actuelle de SDX est la **1.1 en version beta.**.

Vous pouvez obtenir SDX, y compris ses sources, sur le **site SourceForge de SDX** [http://sourceforge.net/project/showfiles.php?group_id=31500] . Toutes les distributions en cours y sont disponibles.

Dans cette distribution, vous trouverez les répertoire suivants :

docs

La documentation SDX.

webapps

Les fichiers constituant l'application SDX de base.

src

Les sources Java pour SDX.

A noter que vous pouvez également consulter la documentation de l'API Java **en ligne** [javadocs/index.html] .

Nous vous conseillons fortement de bien lire la documentation sur SDX avant de vous lancer dans l'installation ou le développement d'applications. Une section est consacrée à l'**installation**, et le document sur l'**aperçu technique** constitue un bon point de départ.

Si vous mettez à jour une installation SDX existante, veuillez lire ce qui suit !

Mise à jour de SDX

La mise à jour de SDX depuis la version 1.01 demande peu d'interventions. Voici quelques points à souligner.

- Le fichier de configuration **`web.xml`** comportait certaines erreurs qui ont été corrigées. Normalement, vous avez déjà modifié ce fichier légèrement, alors il est conseillé d'utiliser le fichier fourni avec la nouvelle distribution et de refaire les modifications que vous avez apportées.
- La structure de la base de données utilisée par SDX est légèrement modifiée. Si vous souhaitez mettre à jour votre installation, vous devez créer manuellement deux tables supplémentaires dans votre base de données. Les instructions sont données dans la **liste des changements apportés**.
- De nouvelles librairies Java (**`.jar`**) sont livrées avec SDX, soit pour remplacer d'anciennes soit pour ajouter des fonctionnalités. Il est donc recommandé de les copier depuis la dsitribution vers votre répertoire d'installation et de redémarrer Tomcat.
- Il est nécessaire de forcer la recompilation de toutes les pages XSP de vos applications. Pour ce faire, vous pouvez détruire les pages compilées en supprimant le contenu du dossier **`cocoon_repository`**. Ce dossier se situe par défaut au même niveau que votre installation de Tomcat, sauf si vous avez changé la configuration.

Historique des changements

Dans cette section de la documentation, nous allons décrire les changements apportés avec les différentes versions de SDX.

- **Liste des changements apportés entre les versions 1.01 et 1.1.**
- **Liste des changements apportés entre les versions 1.0 et 1.01.**

Par ailleurs, une autre partie de cette documentation traite, de façon générale, des **nouveautés de la version 1** par rapport à la version 0.9.

De plus, certaines informations sur la **future version 2** sont disponible.

Changements apportés avec la version 1.1

La version 1.1 de SDX vient corriger plusieurs erreurs et ajoute un certain nombre de nouveautés. La mise à jour de SDX pour passer de 1.0 ou 1.01 à 1.1 demande quelques opérations, en particulier au niveau de la base de données. Voici la liste des nouveautés :

- Un deuxième analyseur lexical est fourni avec SDX, soit la classe **`fr.gouv.culture.sdx.index.FrenchUnaccentedAnalyzer`**. Cette classe supprime également la différence entre les lettres accentuées et non accentuées.
- Correction d'un bogue concernant la réutilisation des requêtes précédentes avec l'API URL (paramètre **n** des servlets de recherche).
- SDX peut maintenant indexer et servir des documents HTML. Pour en savoir plus, consulter la **section de cette documentation** qui explique l'approche utilisée.
- L'élément **`sdx:uploadXMLFile`** de l'API XSP est maintenant remplacé par un élément **`sdx:uploadDocument`**. L'ancien nom est encore supporté, mais il n'est pas conseillé de l'utiliser. Cette fonction va également déclarer et initialiser une variable **`sdx_uploadDocument_docId`** qui contiendra l'identifiant du document importé.
- Ajout d'une nouvelle fonctionnalité concernant les utilisateurs : SDX gère maintenant la notion de groupes d'utilisateurs. Un utilisateur peut faire partie d'un ou plusieurs groupes, mais peut aussi ne pas être associé à aucun groupe. Cette gestion est maintenant assurée par une nouvelle interface Web, et l'approche utilisée **est décrite par ailleurs**.

  Cette nouvelle fonctionnalité demande d'effectuer un changement dans la structure de la base de données. Les requêtes suivantes doivent être effectuées :

```
CREATE TABLE sdx_group
  (db VARCHAR(100) NOT NULL,
  name VARCHAR(255) NOT NULL,
  description VARCHAR(255),
  PRIMARY KEY(db, name));
CREATE TABLE sdx_user_group
  (db VARCHAR(100) NOT NULL,
  user VARCHAR(50) NOT NULL,
  gr VARCHAR(255) NOT NULL,
  ty CHAR(1) NOT NULL,
  PRIMARY KEY (db, user, gr, ty));
```

  Pour accompagner cette gestion des groupes, l'**API XSP** a été modifiée et le **format XML** des utilisateurs.
- Correction d'un bogue qui faisait en sorte que les informations sur les filtres n'étaient pas sortis pour une liste de termes.
- Le fichier de configuration permet maintenant de spécifier si les noms de champs doivent respecter la casse (par défaut) ou non (attribute **`ignoreCase`** de l'élément **`fieldList`** du fichier **`db_info.xml`**.

- Correction de quelques problèmes reliés au nombre de résultats par page de résultats. Maintenant, autant pour les termes d'une liste que pour une recherche, on peut donner un nombre de résultats par page de 0 pour indiquer de ne pas retourner de résultats. Si on donne un nombre positif, ce nombre de résultats par page est appliqué. Si on donne un nombre négatif, tous les résultats sont retournés. Enfin, si on ne spécifie rien, la valeur par défaut (20) est utilisée. Ces remarques sont valables pour les trois API de SDX.
- Correction d'un problème interne affectant les résultats de la servlet **`users`**.
- L'ajout d'un document XML peut se faire depuis un champ d'un formulaire HTML à l'aide de l'attribut **`formParam`** de l'élément **`sdx:uploadDocument`** dans l'**API XSP**.
- Quelques modifications internes pour améliorer (nettement) les performances reliées aux requêtes avec un grand nombre de résultats. Si le tri de pertinence est choisi (par défaut), les données ne sont récupérées qu'à la demande, ce qui réduit les besoins en mémoire. Lorsque les données doivent triées, la situation n'a pas changé. Ces modifications n'impliquent que l'API Java, car deux méthodes publiques de la classe **`fr.gouv.culture.sdx.query.SortSpecification`** ont été supprimées.
- SDX utilise maintenant la version 6.4.4 du processeur XSLT **Saxon** [http://saxon.sourceforge.net] . Pour que vos applications fonctionnent correctement, vous devez faire une modification à votre fichier **`cocoon.properties`**. Repérez l'endroit où se trouve la ligne :

```
transformer.factory.sourceParserClass = fr.gouv.culture.sdx.utils.CatalogSAXParser
```

  et ajoutez sous cette dernière la ligne suivante :

```
transformer.factory.styleParserClass = fr.gouv.culture.sdx.utils.CatalogSAXParser
```

  Le nouveau fichier **`cocoon.properties`** livré avec SDX 1.1 tient compte de ce changement.
- Ajout de la possibilité de spécifier l'emplacement d'un document attaché via une URL, soit relative au document XML soit absolue. L'attribut **`url`** de l'élément **`sdx:attachedDocument`** lors de l'indexation précise cette URL. A noter que cela devient la méthodé privilégiée pour spécifier le document attaché.
- Une nouvelle information d'environnement est fournie par SDX à l'aide de l'élément **`sdx:server`** que l'on trouve dans **`sdx:environment`**. L'attribut **`url`** contient l'adresse URL de l'installation SDX en cours. On pourra avoir, par exemple :

```
<sdx:document>
  <sdx:environment>
    <sdx:server url="http://sdx.culture.fr/sdx"/>
...
```

- On peut maintenant spécifier le nombre de documents à charger à la fois dans SDX, à l'aide d'un élément **`sdx:parameter`** dans le **fichier de configuration**.
- Correction d'un bogue qui empêchait l'utilisation de l'élément **`sdx:deleteDocument`** de l'API XSP.

Par ailleurs, veuillez noter que la résolution des entités externes (par exemple les DTD) via des déclarations SYSTEM et les catalogues **ne fonctionne pas dans cette version de SDX**. Dès qu'un correctif est disponible, il sera rendu public.

Changements apportés avec la version 1.01

La version 1.01 de SDX vient corriger quelques erreurs dans l'application. La liste des erreurs corrigées est la suivante :

- Réécriture des servlets de recherche pour les rendre exploitables en environnement multitâche. En effet, quelques erreurs de programmation introduisaient de faux paramètres dans les requêtes exécutées par les servlets lorsque plusieurs utilisateurs se connectaient, faux paramètres qui induisaient, bien entendu, de faux résultats.
- Modifications apportées au fichier `web.xml` de référence afin de spécifier correctement l'ordre de chargement des 12 servlets que contient SDX. Auparavant, cet ordre n'était pas spécifié, et cela cause des problèmes avec Tomcat 3.2.2.
- Quelques modifications mineures apportées au code, notamment pour uniformiser la façon d'appeler les parseurs XML.

Par ailleurs, veuillez noter que la résolution des entités externes (par exemple les DTD) via des déclarations SYSTEM et les catalogues **ne fonctionne pas dans SDX**. Dès qu'un correctif est disponible, il sera rendu public.

Configuration nécessaire

Ce document présente les composantes que l'on doit installer afin de faire fonctionner correctement SDX. Pour en savoir plus sur ces composantes, vous êtes priés de consulter la documentation propre à chaque outil.

A noter : vous n'avez pas besoin d'installer Cocoon ! Les composantes nécessaires sont incluses dans SDX.

Logiciels

Serveur Web

Un serveur Web est nécessaire pour gérer les requêtes HTTP et l'envoi des informations en retour. Le choix du serveur Web n'est pas très critique, puisque celui-ci joue un rôle peu important. Le seul critère est qu'il puisse être interfacé avec un moteur de servlets adéquat.

Pour l'instant, SDX a été testé et fonctionne avec deux serveurs Web différents, soit **Apache** [http://httpd.apache.org] 1.3.x et le serveur Web intégré au moteur de servlet **Tomcat** [http://jakarta.apache.org/tomcat/index.html] .

Machine virtuelle Java

Toutes les applications dynamiques reliées à SDX sont écrites en Java, alors il est nécessaire d'installer une machine virtuelle Java. Cette machine virtuelle doit être au moins de niveau Java 2, c'est-à-dire correspondre à Java version 1.2 ou supérieure.

Les anciennes versions de Java, jusqu'à Java 1.1.8, ne sont pas supportées par SDX. SDX a été testé et est développé avec Java version 1.3, disponible sur plusieurs plates-formes. Sur Linux, il semble que la machine virtuelle de Sun présente quelques problèmes, alors il est peut-être préférable d'utiliser celle du groupe **Blackdown** [http://www.blackdown.org/] .

De plus, il est important d'installer le SDK Java (*Software Development Kit*), et non le JRE (*Java Runtime Environment*), car le compilateur Java est requis pour les pages dynamiques.

Moteur de servlets

Le moteur de servlets est l'outil responsable de gérer les requêtes HTTP destinées à des servlets Java, de les distribuer aux servlets appropriées, de fournir le contexte d'utilisation aux servlets, et aussi de diffuser les résultats du traitement. Habituellement, le moteur de servlets est associé au serveur Web et ensemble ils constituent une architecture Web dynamique.

SDX a besoin d'un moteur de servlets qui implante la version 2.2 ou supérieure de l'API des servlets de Sun. Ceci **exclut** donc le moteur Apache JServ.

SDX est développé et a été testé avec le moteur de servlets **Tomcat** [http://jakarta.apache.org/tomcat/index.html] version 3.2.2 du projet Apache. Si vous installez SDX avec un autre moteur, vous êtes priés de le communiquer aux **responsables de la plate-forme SDX**, les informations sur votre installation pourraient profiter à d'autres.

Système de gestion de bases de données relationnelles

SDX n'est pas une application de base de données traditionelle. Toutefois, un SGBD est utilisé pour stocker les documents XML et leurs documents attachés, de même que certaines informations de gestion.

SDX est développé prioritairement avec le SGBD **MySQL** [http://www.mysql.com] . Toutefois, il supporte également les SGBD **Oracle** [http://www.oracle.com] (version 8), **Interbase** [http://www.interbase.com] (version 6) et **InstantDB** [http://instantdb.enhydra.org] (version 3.25). L'ajout du support d'un autre SGBD demandera peut-être certains ajustements au code Java de SDX, mais ces modifications sont mineures.

Le support du SGBD **PostgreSQL** [http://www.postgresql.org] en version 7.1 ou supérieure est prévu très bientôt. Par ailleurs, le support d'autres SGBD Java sera ajouté prochainement.

Le choix du SGBD devrait être fait en fonction de sa bonne (ou mauvaise) gestion des champs de type *BLOB*. En effet, dans SDX, les documents XML ainsi que leurs documents attachés sont stockés dans de tels champs, afin d'éviter un trop grand nombre de fichiers.

Système d'exploitation

L'utilisation de Java comme langage de développement permet, a priori, de s'affranchir des contraintes de systèmes d'exploitation. Toutefois, puisque SDX repose sur un certain nombre de composantes externes qui ne sont pas nécessairement en Java, tous les systèmes d'exploitation ne sont pas supportés.

Jusqu'à maintenant, SDX a fonctionné sur des plates-formes Windows NT, Windows 98 et Linux. Il n'y a aucune raison de croire que d'autres systèmes UNIX causeraient des problèmes.

Soulignons également qu'avec InstantDB comme SGBD et Tomcat comme serveur Web, SDX peut être une application 100 % Java, ce qui peut être utile pour la diffusion d'une base sur un CD-ROM disponible pour plusieurs plate-formes.

Matériel

L'installation de SDX nécessaire un ordinateur quelconque mais suffisamment puissant pour permettre de prendre en charge les applications développées et les volumes de documents à gérer. En général, un PC de bureau moyen vendu aujourd'hui est suffisant.

L'aspect le plus critique est la mémoire vive. En effet, puisque SDX est bâti en Java, il faut allouer à la machine virtuelle Java une quantité suffisante de mémoire si on veut obtenir des performances suffisantes.

Pour des applications simples, nous avons constaté que SDX pouvait se contenter d'environ 40Mo alloués à la machine virtuelle Java. Par conséquent, il est préférable d'insaller SDX sur une machine qui possède au moins 128Mo de mémoire vive.

Pour des applications plus lourdes ou plus fréquentées, 512Mo de mémoire n'est pas un luxe.

Installation des composantes de base

L'architecture SDX est bâtie à partir de plusieurs **composantes de base**. L'objectif de ce document n'est pas de donner les détails de l'installation de ces composantes, mais plutôt de les lister et de préciser les aspects spécifiques requis par SDX. Pour en savoir plus sur l'installation de ces composantes, voir leur documentation respective.

Serveur Web Apache

Apache est le serveur Web le plus utilisé. Il fonctionne sur plusieurs plates-formes, y compris Linux et Windows NT avec lequel SDX a été testé. On peut en savoir plus sur Apache et sur son installation à partir d'**ici** [http://httpd.apache.org] . SDX devrait fonctionner correctement sur toute version d'Apache depuis 1.3.

Il n'y a rien de particulier concernant SDX et l'installation d'Apache. La configuration nécessaire sera faite lors de l'installation du moteur de servlets.

Moteur de servlets Tomcat

Tomcat est un moteur de servlets initalement développé par Sun puis donné au projet Apache. Au sein de ce projet, il remplace donc le moteur Apache JServ. On peut en savoir plus sur Tomcat et le télécharger **ici** [http://jakarta.apache.org/tomcat/index.html] .

Tomcat est l'implantation de référence de la spécification des servlets. Il implante la version 2.2 de la spécification. SDX a été testé avec les versions 3.2.x de Tomcat.

La **documentation de Tomcat** [http://jakarta.apache.org/tomcat/tomcat-3.2-doc/index.html] indique toute la procédure à suivre pour installer Tomcat de concert avec Apache, ou encore en mode *standalone*. Il est fortement recommendé de suivre ces instructions pas à pas. Les documents à lire prioritairement sont :

- **Tomcat User's Guide** [http://jakarta.apache.org/tomcat/tomcat-3.2-doc/uguide/tomcat_ug.html]
- **Tomcat Workers** [http://jakarta.apache.org/tomcat/tomcat-3.2-doc/Tomcat-Workers-HowTo.html]
- **The Jakarta NT Service** [http://jakarta.apache.org/tomcat/tomcat-3.2-doc/NT-Service-howto.html] (pour une installation sous Windows)
- **Tomcat-Apache HOW-TO** [http://jakarta.apache.org/tomcat/tomcat-3.2-doc/tomcat-apache-howto.html] (essentiel pour bien relier les deux outils)
- **Working with mod_jk** [http://jakarta.apache.org/tomcat/tomcat-3.2-doc/mod_jk-howto.html] (module Apache faisant la liaison avec Tomcat)

Ce dernier document est particulièrement important, car il vous permettra de bien faire fonctionner Tomcat avec Apache, et d'identifier les composantes (par exemple **`mod_jk`**) nécessaires.

Il est à noter que pour obtenir une bonne communication entre Apache et Tomcat, l'utilisation de la configuration automatique produite par Tomcat dans le fichier **`mod_jk.conf-auto`** n'est pas suffisante. On doit conserver une copie statique de ce fichier, et y mettre des lignes semblables aux suivantes :

```
############################################################
# Auto configuration for the /sdx context starts.
############################################################
#
# The following line makes apache aware of the location of the /sdx context
#
Alias /sdx "/usr/local/jakarta/jakarta-tomcat/webapps/sdx"
<Directory "/usr/local/jakarta/jakarta-tomcat/webapps/sdx">
    Options Indexes FollowSymLinks
</Directory>
#
# The following line mounts all JSP files and the /servlet/ uri to tomcat
#
JkMount /sdx/*.xsp ajp12
JkMount /sdx/servlets/* ajp12
#
# The following line prohibits users from directly accessing WEB-INF
#
<Location "/sdx/WEB-INF/">
    AllowOverride None
```

```
    deny from all
</Location>
#
# The following line prohibits users from directly accessing META-INF
#
<Location "/sdx/META-INF/">
    AllowOverride None
    deny from all
</Location>
###########################################################
# Auto configuration for the /sdx context ends.
###########################################################
```

Bien entendu, vous devez ajuster les chemins d'accès spécifiés à votre installation.

Dans cet exemple, nous avons utilisé le protocole **AJP12** pour la communication entre Apache et Tomcat. Ce protocole est moins efficace que le protocole AJP13, mais ce dernier présente un bogue qui rend impossible (dans certains cas) l'envoi de fichiers par le mécanisme des *file upload*, ce qui peut être utile dans une application SDX. Libre à vous de choisir le protocole idéal pour vos applications.

Démarrage de Tomcat

Si les installations précédentes se sont bien déroulées, Tomcat devrait pouvoir démarrer. Une bonne façon de le vérifier est de tester les exemples fournis avec le logiciel. **Il est très important de tester à la fois les servlets et les pages JSP**. En effet, ces dernières vont permettre de vérifier si le compilateur Java est bien présent dans votre environnement, ce qui est requis pour SDX.

Tomcat est normalement démarré à l'aide des scripts fournis dans son répertoire bin, à la fois pour Windows et pour UNIX/Linux. Il est à noter que les fichiers **`.jar`** que l'on retrouve dans le répertoire **`sdx/WEB-INF/lib`** doivent être inclus manuellement dans le CLASSPATH Java, à configurer dans le script de démarrage de Tomcat.

Pour ce faire, vous devez inclure des lignes comme celles-ci sous Windows :

```
set CLASSPATH=%CLASSPATH%;%SDX_LIBS%\aa1_sdx.jar
set CLASSPATH=%CLASSPATH%;%SDX_LIBS%\aa2_xerces.jar
set CLASSPATH=%CLASSPATH%;%SDX_LIBS%\aa3_saxon.jar
...
```

et comme celles-ci sous UNIX/Linux :

```
CLASSPATH=${CLASSPATH}:${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib/aa1_sdx.jar
CLASSPATH=${CLASSPATH}:${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib/aa2_xerces.jar
CLASSPATH=${CLASSPATH}:${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib/aa3_saxon.jar
...
```

Sous UNIX/Linux, vous pouvez également faire une boucle comme celle-ci :

```
for i in ${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib/* ; do
  CLASSPATH=${CLASSPATH}:$i
done
```

Attention ! L'ordre des fichiers **`.jar`** a une certaine importance, malheureusement (ceci est dû à la multitude de librairies XML variées que l'on rencontre dans tous ces fichiers). Vous devez absolument respecter ces quelques règles :

1) Les fichiers **`.jar`** situés dans le répertoire *${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib* doivent être inclus avant ceux situés dans le répertoire *${TOMCAT_HOME}/lib*.
2) Dans le répertoire *${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib* , vous devez d'abord inclure, dans cet ordre, les fichiers *aa1_sdx.jar*, *aa2_xerces.jar* et *aa3_saxon.jar*. Par la suite, l'ordre importe peu.

SGBD

SDX utilise un système de gestion de bases de données relationnelles pour stocker les documents XML et leurs documents associés. La version actuelle de SDX peut être utilisée avec quatre SGBD différents, dont deux logiciels libres et un autre gratuit :

- MySQL version 3.23 ou supérieure (logiciel libre, disponible sur plusieurs plates-formes, **http://www.mysql.com** [http://www.mysql.com])
- Interbase version 6 ou supérieure (logiciel libre, disponible sur plusieurs plates-formes, **http://www.interbase.com** [http://www.interbase.com])
- InstantDB version 3.25 ou supérieure (logiciel gratuit, disponible sur plate-forme Java, **http://www.lutris.com/products/instantDB/index.html** [http://www.lutris.com/products/instantDB/index.html])
- Oracle 8i ou version supérieure (logiciel commercial, **http://www.oracle.com** [http://www.oracle.com])

SDX est développé prioritairement avec MySQL, mais il est systématiquement testé avec tous les SGBD supportés. Si vous voulez utiliser un autre SGBD, **contactez-nous**.

L'installation du SGBD n'est pas documentée ici ; vous êtes priés de vous adresser au fournisseur du produit ou à sa documentation. Par ailleurs, la distribution SDX contient les pilotes nécessaires pour MySQL, InstantDB et Interbase, mais pas pour Oracle.

Configuration initiale

Avant de pouvoir utiliser SDX, certains paramétrages doivent être effectués. Ce document fournit les indications nécessaires. Dans l'ordre, on doit copier les fichiers de la configuration de base, modifier (si nécessaire) les paramètres de configuration, créer la base de données, créer la base de documents '"sdxdoc", et enfin alimenter cette dernière. A la fin, on obtiendra une version fonctionnelle de SDX avec une base de documents alimentée.

Copie des fichiers

La distribution SDX contient un dossier nommé **`webapps/sdx`**. Ce dernier doit être copié intégralement dans le répertoire webapps approprié pour votre installation Tomcat.

Paramètres de configuration

Les fichiers à vérifier sont :

- webapps/sdx/WEB-INF/web.xml
- webapps/sdx/WEB-INF/cocoon.properties
- webapps/sdx/sdxdoc/conf/catalog

Configuration Tomcat

Le fichier **`web.xml`** contient les paramètres de configuration de SDX au niveau Tomcat. Dans ce fichier, vous devez repérer les lignes suivantes :

```
<context-param>
  <param-name>sdx_baseUrl</param-name>
  <param-value>http://localhost/sdx/</param-value>
</context-param>
```

Si cette URL ne correspond pas à votre installation, modifiez-la en conséquence.

De plus, si vous utilisez Oracle, Interbase ou InstantDB comme SGBD, vous devrez repérer ces lignes :

```
<context-param>
  <param-name>sdx_DatabaseType</param-name>
  <param-value>mysql</param-value>
```

et y mettre **`oracle`** ou **`interbase`** ou **`instandb`** à la place de **`mysql`**.

Configuration Cocoon

Le fichier **`cocoon.properties`** contient la configuration nécessaire pour l'engin Cocoon utilisé par SDX. Les indications les plus importantes concernent la base de données. A ce sujet, repérer la section **# Turbine DB Connection Pool**, vous y trouverez quelques configurations de base pour certains SGBD. La partie non commentée, et donc active, est valable pour une base de données MySQL nommée "sdx" et accessible sur le serveurr localhost avec le code d'utilisateur *sdx* et le mot de passe *xml*. Vous pouvez modifier ces paramètres pour qu'ils soient cohérents avec votre configuration dans MySQL.

Si vous utilisez un SGBD autre que MySQL, modifiez les entrées en conséquence, en suivant les autres modèles commentés dans le fichier.

Pour un fonctionnement standard de SDX, aucune autre adaptation n'est nécessaire dans ce fichier.

Configuration du catalogue *sdxdoc*

Plus tard dans l'installation, nous allons créer une base de documents nommée **`sdxdoc`**, pour la documentation en ligne. Pour une bonne gestion des DTD, il est nécessaire d'utiliser un tel système de catalogues. Ce catalogue est défini par le consortium **OASIS** [http://www.oasis-open.org] . Il est probable que vous deviez modifier la première ligne du fichier **`sdxdoc/conf/catalog`** pour indiquer l'URL de base de votre installation SDX.

Création de la base de données

SDX nécessite une base de données relationnelle pour le stockage des documents et pour certaines informations de gestion. Le fichier **`cocoon.properties`** indique aux outils où se trouve la base de données et comment y accéder. Le paramétrage de cette base de données doit donc être en phase avec les informations indiquées dans ce fichier.

Il doit y avoir une base de données accessible pour SDX. On doit y créer un utilisateur ayant les droits d'écriture de données et de modification de structure dans cette base de données. Toutes les tables utilisées par SDX ont un nom qui commence par **`sdx`**. Il est donc possible d'installer SDX sur une base existante, en autant que cette dernière n'utilise pas des noms de table commençant par ces trois lettres.

Nous avons remarqué que dans certaines situations, il est nécessaire de créer un utilisateur avec mot de passe pour que le lien entre SDX et la base de données soit fonctionnel. Nous vous suggérons donc fortement de le faire, ce qui est aussi une bonne chose pour la sécurité.

Dans les paragraphes qui suivent, nous prenons pour acquis que cette base s'appelle **`sdx_v1`**.

A noter : Pierrick Brihaye a écrit un **script pour MySQL** [mysql.conf] qui permet de créer la base de données et de créer les structures et données nécessaires. Si vous exécutez ce script, vous n'aurez pas besoin de faire les étapes suivantes. Nous vous conseillons toutefois de modifier le mot de passe administrateur qui s'y trouve.

Les tables globales

SDX a besoin de cinq tables globales pour fonctionner. Ces cinq tables ont la structure suivante (définition normalement valable pour tout SGBD respectant le standard SQL/92, voir plus loin pour **Interbase**) :

```
CREATE TABLE sdx_user
(
```

```
  code varchar(50) NOT NULL,
  passwd varchar(50),
  first_name varchar(100),
  last_name varchar(100),
  language char(2),
  PRIMARY KEY (code)
);
CREATE TABLE sdx_privilege
(
  user_code varchar(50) NOT NULL,
  db_code varchar(100) NOT NULL,
  type char(1) NOT NULL,
  PRIMARY KEY (user_code,db_code)
);
CREATE TABLE sdx_db
(
  code varchar(100) NOT NULL,
  admin_name varchar(100),
  admin_email varchar(255),
  PRIMARY KEY (code)
);
CREATE TABLE sdx_group
(db VARCHAR(100) NOT NULL,
name VARCHAR(255) NOT NULL,
description VARCHAR(255),
PRIMARY KEY(db, name)
);
CREATE TABLE sdx_user_group
(db VARCHAR(100) NOT NULL,
user VARCHAR(50) NOT NULL,
gr VARCHAR(255) NOT NULL,
ty CHAR(1) NOT NULL,
PRIMARY KEY (db, user, gr, ty)
);
```

Ces trois requêtes SQL ont été testées avec succès sur MySQL version 3.23, InstantDB version 3.25 et Oracle 8i.

Avec **Interbase**, il faut utiliser ces requêtes :

```
CREATE TABLE sdx_user
(
  code varchar(50) NOT NULL,
  passwd varchar(50),
  first_name varchar(100),
  last_name varchar(100),
  language char(2),
  PRIMARY KEY (code)
);
CREATE TABLE sdx_privilege
(
  user_code varchar(50) NOT NULL,
  db_code varchar(100) NOT NULL,
  "type" char(1) NOT NULL,
  PRIMARY KEY (user_code,db_code)
);
CREATE TABLE sdx_db
(
  code varchar(100) NOT NULL,
  admin_name varchar(100),
  admin_email varchar(255),
  PRIMARY KEY (code)
);
CREATE TABLE sdx_group
(db VARCHAR(100) NOT NULL,
name VARCHAR(255) NOT NULL,
description VARCHAR(255),
PRIMARY KEY(db, name)
);
CREATE TABLE sdx_user_group
(db VARCHAR(100) NOT NULL,
user VARCHAR(50) NOT NULL,
gr VARCHAR(255) NOT NULL,
ty CHAR(1) NOT NULL,
PRIMARY KEY (db, user, gr, ty)
```

```
);
```

L'administrateur

Il doit y avoir au moins un administrateur de SDX. Voici les requêtes nécessaires pour le créer :

```
INSERT INTO sdx_user (first_name, last_name, code, passwd, language)
VALUES ('', 'Administrateur', 'admin', 'nimda', 'fr');
INSERT INTO sdx_privilege ( user_code, db_code, type )
VALUES ('admin', ' ', 's');
```

Attention ! Il est très fortement conseillé de modifier le mot de passe !

Ces deux requêtes SQL ont également été testées avec succès sur MySQL version 3.23, InstantDB version 3.25 et Oracle 8i.

Avec **Interbase**, utiliser plutôt :

```
INSERT INTO sdx_user (first_name, last_name, code, passwd, language)
VALUES ('', 'Administrateur', 'admin', 'nimda', 'fr');
INSERT INTO sdx_privilege ( user_code, db_code, "type" )
VALUES ('admin', ' ', 's');
```

Création de la base de documents *sdxdoc*

Afin de montrer les possibilités de SDX, il est intéressant de créer tout de suite une base de documents dont le contenu est fourni avec la distribution. Il s'agit de la documentation que vous êtes en train de consulter.

Pour ce faire, on doit aller à la page d'accueil SDX (**`index.xsp`**), puis s'enregistrer en tant qu'administrateur, dans le formulaire en bas de la page. Une fois cela fait, on retrouve la page d'accueil, mais cette fois avec un lien vers l'administration SDX, dans le haut de la page. Suivre ce lien.

La page obtenue liste les bases de documents accessibles. Normalement, il ne devrait y en avoir aucune. Il faut donc cliquer sur le lien "Ajouter une base de documents". Dans le formulaire proposé, taper le nom de code **`sdxdoc`** ainsi que des informations de base sur l'administrateur de cette base de documents (tous les champs sont obligatoires). Si tout se déroule comme prévu, vous verrez la même page, mais cette fois avec la base de documents créée.

Ensuite, vous pouvez cliquer sur le code de la base et vous devriez trouver une petite page d'accueil pour cette base de documents. A ce moment, un lien vous indique la procédure à suivre pour charger les document.

Alimentation de la base de documents *sdxdoc*

Pour pouvoir profiter de cette base de documents, on doit y insérer quelques documents XML. Vous pourrez le faire avec la documentation SDX, dont les sources XML sont fournies. Si vous êtes correctement identifiés comme administrateur, vous pouvez suivre le lien *Charger des documents XML en vrac* et puis indiquer, dans la première zone de texte, le répertoire **`docs/hyperarticle/xml`** de la distribution (ces documents doivent être sur un disque accessible au serveur et vous devez préciser le chemin d'accès complet au répertoire).

Présentation

Cette section a pour objectif de présenter des exemples d'installation de SDX avec différents environnements. Vous trouverez pour l'instant :

- Les **notes d'installation** de Pierrick Brihaye, de la DRAC Bretagne, qui a installé SDX sur un ordinateur Windows 98. Ces notes ont été écrites pour la version 0.9 mais corrigées pour la version 1.0 de SDX.
- Les **notes d'installation** de François Pinot, de la DRAC Lorraine, qui a installé SDX version 1 beta 1 sous Linux.

Installation sous Windows 98

Pierrick Brihaye [mailto:pierrick.brihaye@culture.gouv.fr] de la DRAC Bretagne a installé SDX version 0.9 sur un ordinateur sous Windows 98. Nous sommes en train d'adapter son texte à l'installation de la version 1, vous le trouverez ici prochainement.

Installation sous linux

Cet article décrit l'installation de la plate-forme SDX v1.0 (version beta 1) sur un système linux Mandrake 7.2. Nous verrons successivement le réglage de linux, l'installation du SDK Java (Software Development Kit), l'installation du moteur de servlets Tomcat et enfin, l'installation de SDX v1.0 (version beta 1). Pour tous renseignements complémentaires, vous pouvez me contacter par mél : **francois.pinot@culture.gouv.fr** [mailto:francois.pinot@culture.gouv.fr] ou par téléphone au 03.87.56.41.02 (DRAC de Lorraine).

Paramétrage de linux

L'installation de linux Mandrake 7.2 se fait en mode serveur en acceptant les options proposées par défaut, ce qui est simple et rapide. Après le premier démarrage, on peut désactiver divers services qui ont été intallés mais qui ne sont pas vraiment nécessaires pour le fonctionnement de SDX. La commande **drakxservices** permet de choisir les services que linux doit démarrer à l'initialisation du système. A titre indicatif, voici la liste des services actifs sur mon installation :

- autofs
- cfd
- crond
- cups
- gpm
- **httpd**
- identd
- keytable
- kheader
- linuxconf
- lpd
- lvs
- **mysql**
- netfs
- network

- numlock
- random
- sshd
- swat
- syslog
- **telnet**
- **wu-ftpd**
- xfs
- **xinetd**

Le service **httpd** est le serveur Web Apache 1.3.14 qui, s'il est utilisé avec SDX, permet de mélanger les documents SDX avec d'autres sources Web. Le serveur de bases de données dont SDX a besoin est **mysql.** Les services **telnet** et **wu-ftpd** ne sont pas nécessaires pour le fonctionnement de SDX, mais ils sont bien utiles pour la gestion à distance de la plate-forme. La seule décision concernant la sécurité est le choix du wrapper TCP **xinetd** plutôt que le classique inetd. Pour les autres éléments de sécurité, je me repose lâchement sur les services du DOSI. Il va sans dire qu'une étude détaillée sur les éléments de sécurité à apporter à une plate-forme SDX sera la bienvenue.

Les autres services de cette liste sont nécessaires au fonctionnement de linux, mais je ne m'étendrai pas ici sur leur description.

Enfin, il faut installer le paquetage *apache-devel* car nous en aurons besoin pour compiler un module de bibliothèque dynamique pour le serveur Web Apache. Mettre le deuxième cdrom de la distribution Mandrake 7.2 dans le lecteur et taper les commandes suivantes :

```
mount /mnt/cdrom
rpm -ivh /mnt/cdrom/Mandrake/RPMS2/apache-devel-1.3.14-2mdk.i586.rpm
```

Installation du SDK Java

SDX a besoin du SDK Java (sofware development kit) pour fonctionner. Il vaut mieux récupérer la dernière version directement sur le site de **Sun** [http://www.java.sun.com/j2se/1.3/download-linux.html] . Il faut télécharger le fichier *j2sdk-1_3_0_02-linux-rpm.bin* (**section RedHat RPM shell script**) qui est un script d'auto-extraction du paquetage rpm correspondant. Une fois ce fichier copié dans le répertoire de root, exécuter la commande **./j2sdk-1_3_0_02-linux-rpm.bin**, qui va extraire le paquetage *j2sdk-1_3_0_02-linux.rpm* après vous avoir posé les question rituelles sur la lecture et l'acceptation de la licence Sun.

Une fois le paquetage rpm extrait, il faut l'installer avec la commande **rpm -ivh j2sdk-1_3_0_02-linux.rpm**. Pour une installation seule du SDK Java, il faudrait modifier les variables d'environnement de l'utilisateur root. Nous ferons cela à l'étape suivante lorsque nous configurerons tomcat.

Le SDK Java ayant été installé dans */usr/java/jsk1.3.0_02/*, les fichiers *j2sdk-1_3_0_02-linux-rpm.bin* et *j2sdk-1_3_0_02-linux.rpm* ne sont plus nécessaires et peuvent être supprimés du répertoire de l'utilisateur root.

Installation de tomcat

Nous aurons besoin de certaines variables d'environnement aussi bien pour Java que pour Tomcat. Pour éviter de les ressaisir à chaque session, nous créons dans le répertoire **/root** un script nommé **tomcatenv** dont le contenu est donné dans le tableau suivant :

```
JAVA_HOME=/usr/java/jdk1.3.0_02
```

```
export JAVA_HOME
JAKARTA_HOME=/usr/jakarta
export JAKARTA_HOME
TOMCAT_HOME=$JAKARTA_HOME/jakarta-tomcat-3.2.1
export TOMCAT_HOME
PATH=$PATH:$JAVA_HOME/bin
export PATH
CLASSPATH=$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/dt.jar
export CLASSPATH
```

Ne pas oublier de rendre ce script exécutable avec la commande **chmod 755 tomcatenv**. Modifier le fichier **/root/.bashrc** en y ajoutant tout à fin la ligne

```
. /root/tomcatenv
```

De cette manière, chaque fois que l'on ouvrira une session sous le nom de *root*, ces variables seront positionnées. Pour l'heure, nous le faisons manuellement en tapant la commande **./tomcatenv**, et nous vérifions que tout c'est bien passé en affichant les variables d'environnement (**printenv**).

Se positionner dans le répertoire **/usr**, y créer un répertoire nommé **jakarta**, s'y positionner. Télécharger l'archive **jakarta-tomcat-3.2.1.tar.gz** [http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.1/bin/jakarta-tomcat-3.2.1.tar.gz] (version binaire de Tomcat), la décompresser, et supprimer la copie. Télécharger l'archive **jakarta-tomcat-3.2.1-src.tar.gz** [http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.1/src/jakarta-tomcat-3.2.1-src.tar.gz] (sources de Tomcat nécessaires pour compiler la bibliothèque dynamique de liaison entre Tomcat et le serveur Web Apache), la décompresser, et supprimer la copie. Ce qui nous donne la suite d'instructions :

```
cd /usr
mkdir /jakarta
cd jakarta
 ...
tar zxvf jakarta-tomcat-3.2.1.tar.gz
rm jakarta-tomcat-3.2.1.tar.gz
 ...
tar zxvf jakarta-tomcat-3.2.1-src.tar.gz
rm jakarta-tomcat-3.2.1-src.tar.gz
```

(Les points de suspension représentent les phases de téléchargement des archives).

Nous allons maintenant configurer Tomcat. Pour cela, se positionner dans son répertoire de configuration, sauver les fichiers originaux dans un répertoire que l'on nomme *original-conf* et copier les fichiers dont on aura besoin dans le répertoire *conf*. Puis se positionner dans le répertoire *bin* et démarrer Tomcat avec le script **startup.sh**

```
cd ${TOMCAT_HOME}/conf
mkdir original-conf
mv * original-conf
cd original-conf
cp server.xml ..
cp web.xml ..
cp workers.properties ..
cp tomcat-users.xml ..
cd ${TOMCAT_HOME}/bin
./startup.sh
```

Vous pouvez alors vérifier le fonctionnement de Tomcat en l'appelant depuis votre navigateur favori avec l'URL **http://nom.du.serveur:8080/** où *nom.du.serveur* doit être remplacé par le nom complet de votre serveur (ex : *sdx-lorraine.culture.fr*) . Si le nom du serveur n'est pas enregistré dans le DNS de votre domaine, vous pouvez le remplacer par son adresse IP.

Tomcat est un serveur Web qui écoute sur le port 8080. Dès qu'il est lancé, il faut vérifier son bon

fonctionnement en testant les examples JSP et les examples de Servlets. Si tout fonctionne bien, stopper Tomcat avec le script **shutdown.sh** du répertoire *bin* de Tomcat.

Bien que l'on puisse utiliser Tomcat comme seul serveur Web, il est plus intéressant de bénéficier de la puissance du serveur Web Apache. Dans cette configuration, les requêtes arrivent sur le serveur Web Apache qui transmet ces dernières à Tomcat si elles le concernent. Dans le cas contraire, Apache traite lui-même les requêtes. Ainsi, on peut avoir un traitement Web classique en plus de SDX.

La liaison entre le serveur Web Apache et Tomcat se fait par un module de bibliothèque dynamique appelé *mod_jk.so*. Nous allons compiler ce module d'après les sources de Tomcat, le copier dans le répertoire des bibliothèques dynamiques du serveur Apache, puis nous effacerons les sources de Tomcat dont nous n'aurons plus besoin. Dans le tableau suivant, les symboles \ indiquent une coupure de ligne pour un affichage plus lisible. Dans la réalité, il faut taper la commande sur une seule ligne sans les \

```
cd ${JAKARTA_HOME}/jakarta-tomcat-3.2.1-src/src/native
cd apache1.3
/usr/sbin/apxs -o mod_jk.so \
  -I${JAVA_HOME}/include/linux \
  -I../jk -I${JAVA_HOME}/include \
  -c *.c ../jk/*.c
cp mod_jk.so /usr/lib/apache
cd ${JAKARTA_HOME}
rm -rf jakarta-tomcat-3.2.1-src
```

Nous allons modifier la configuration du serveur Apache pour qu'il prenne en compte le nouveau module. Se positionner dans le répertoire de configuration du serveur Apache (**cd /etc/httpd/conf**), éditer le fichier *httpd.conf* en ajoutant à la fin du fichier la ligne suivante :

```
Include /usr/jakarta/jakarta-tomcat-3.2.1/conf/mod_jk.conf
```

Se positionner dans le répertoire du serveur Apache et créer un lien symbolique vers ses bibliothèques dynamiques (ce lien est utilisé dans le fichier *mod_jk.conf*).

```
cd /etc/httpd
ln -s ../../usr/lib/apache libexec
```

Nous allons créer le fichier *mod_jk.conf*, puis nous anticiperons sur la section suivante en modifiant déjà le fichier pour indiquer au serveur Apache l'existence des ressources SDX. De cette manière, nous n'aurons à stopper et à redémarrer qu'une fois le serveur Apache.

Lors du premier démarrage de Tomcat, ce dernier a créé un fichier nommé *mod_jk.conf-auto* . Nous allons copier ce fichier pour créer notre fichier de configuration puis nous le modifierons.

```
cd ${TOMCAT_HOME}/conf
cp mod_jk.conf-auto mod_jk.conf
```

Nous éditons le fichier *mod_jk.conf* afin d'y ajouter à la fin les lignes suivantes, qui ont pour but d'indiquer au serveur Web Apache l'endroit où se trouve SDX et qu'il faut traiter les requêtes SDX via Tomcat.

```
############################################################
# Auto configuration for the /sdx context starts.
############################################################
#
# The following line makes apache aware of the location of the /sdx context
#
Alias /sdx "/usr/jakarta/jakarta-tomcat-3.2.1/webapps/sdx"
<Directory "/usr/jakarta/jakarta-tomcat-3.2.1/webapps/sdx">
```

```
    Options Indexes FollowSymLinks
</Directory>
#
# The following line mounts all JSP files and the /servlet/ uri to tomcat
#
JkMount /sdx/* ajp12
#
# The following line prohibits users from directly accessing WEB-INF
#
<Location "/sdx/WEB-INF/">
    AllowOverride None
    deny from all
</Location>
#
# The following line prohibits users from directly accessing META-INF
#
<Location "/sdx/META-INF/">
    AllowOverride None
    deny from all
</Location>
############################################################
# Auto configuration for the /sdx context ends.
############################################################
```

Pour finir, nous allons arrêter le serveur Apache, démarrer Tomcat et redémarrer le serveur Apache.

```
kill -TERM `cat /var/run/httpd.pid`
cd ${TOMCAT_HOME}/bin
./startup.sh
httpd
```

Vous pouvez alors vérifier le bon fonctionnement de la liaison entre le serveur Apache et Tomcat en appelant l'URL **http://nom.du.serveur/examples/** depuis un navigateur et en vérifiant le bon fonctionnement des examples JSP et des examples Servlets. Si tout fonctionne bien, stopper Tomcat avec le script **shutdown.sh**. Nous sommes maintenant prêts à aborder l'installation de SDX.

Installation de SDX

Il faut d'abord modifier le script de démarrage de Tomcat pour prendre en compte SDX. Se positionner dans le répertoire binaire de Tomcat

```
cd ${TOMCAT_HOME}/bin
```

Editer le script *tomcat.sh*. Réperer les lignes suivantes :

```
oldCP=$CLASSPATH
unset CLASSPATH
```

Insérer juste après ces lignes le texte suivant qui a pour but d'ajouter dans le CLASSPATH les classes Java de SDX.

```
CLASSPATH=${CLASSPATH}:${TOMCAT_HOME}/webapps/sdx/WEB-INF/classes
for i in ${TOMCAT_HOME}/webapps/sdx/WEB-INF/lib/* ; do
  CLASSPATH=${CLASSPATH}:$i
done
```

Copier SDX dans le répertoire *webapps* de SDX, le dézipper, effacer le fichier compressé, copier le répertoire *sdx* et la documentation xml dans *webapps*, et supprimer le reste de la distribution qui n'est pas indispensable à cet endroit.

```
cd ${TOMCAT_HOME}/webapps
...
unzip sdx-1_0b1.zip
rm sdx-1_0b1.zip
cd sdx-1.0b1/webapps
mv sdx ../..
cd ..
mv docs ../sdx
cd ..
rm -rf sdx-1.0b1
```

Il faut maintenant configurer MySQL. Si vous n'avez pas donné de mot de passe à l'utilisateur *root* de MySQL (qui n'a rien à voir avec l'utilisateur *root* de linux), c'est le moment de le faire avec la commande

```
mysqladmin -u root password 'votre.mot-de-passe'
```

Nous devons ajouter à MySQL l'utilisateur *sdx* avec le mot de passe *xml*, créer la base *sdx_v1*, garantir tous les droits à l'utilisateur *sdx* sur la base *sdx_v1* et réinitialiser MySQL pour la prise en compte de tous ces changements.

```
mysql -u root -p mysql
Enter password: ...
mysql>INSERT INTO user (host,user,password)
mysql>VALUES ('localhost.localdomain','sdx',password('xml'));
mysql>CREATE DATABASE sdx_v1;
mysql>GRANT ALL ON sdx_v1.* TO sdx;
mysql>exit
mysqladmin -u root -p reload
```

Le fichier *web.xml* de Tomcat doit être modifié pour refléter l'URL de base de SDX sur votre machine. Se positionner dans le répertoire *WEB-INF* de sdx.

```
cd ${TOMCAT_HOME}/webapps/sdx/WEB-INF
```

Editer le fichier *web.xml*. Trouver les lignes suivantes :

```
<!-- L'URL de base (interne) pour cette installation SDX -->
    <context-param>
        <param-name>sdx_baseUrl</param-name>
        <param-value>http://localhost/sdx/</param-value>
    </context-param>
```

Remplacer *localhost* par le nom long de votre machine que vous trouverez dans le fichier */etc/hosts*. Se positionner dans le répertoire de configuration de *sdxdoc*

```
cd ${TOMCAT_HOME}/webapps/sdx/sdxdoc/conf
```

Editer le fichier *catalog*, modifier la première ligne (*BASE "http://localhost/sdx/sdxdoc/dtd"*) en y remplaçant *localhost* par le nom long de votre machine. Ceci est nécessaire pour que SDX trouve bien la DTD correspondant aux documents.

Rappeler MySQL avec la nouvelle base

```
mysql -u sdx -p sdx_v1
Enter password: xml
```

A partir de là, l'installation peut continuer en suivant les instructions de la section **Configuration initiale**, à partir du paragraphe **Les tables globales**. Lorsque cela est fini redémarrer Tomcat. Le système affiche des messages d'erreur concernant les servlets. Ne pas en tenir compte.

Conclusion

Cette description d'une installation SDX est très détaillée dans un but de pédagogie par la pratique. Si cet aspect pédagogique n'est pas nécessaire, on peut automatiser l'installation dans un script qui pourrait faire partie d'une version régulière de SDX.

Développement d'applications SDX

Aperçu technique de SDX

SDX est une application Web dynamique qui a été conçue de façon à minimiser les développements en réutilisant le plus possible des composantes *logiciels libres* déjà disponibles et efficaces.

Cette approche présente plusieurs avantages, mais apporte aussi des invénients. Parmi ceux-ci, notons les problèmes d'installation ainsi que la multiplicité des sources de documentation. Dans cette documentation propre à SDX, nous retrouverons seulement de l'information sur ce qui est particulier à cette plate-forme, mais nous essaierons de mettre des liens vers les autres documents ou sites Web pertinents, lorsque nécessaire.

Approche technologique

SDX est une application Web dynamique basée sur l'architecture des Java servlets, c'est-à-dire des composantes dynamiques écrites en Java. Toutefois, il est possible de créer des applications SDX complexes **sans voir une seule ligne de code Java**. Par-dessus cette couche Java, deux technologies jouent un rôle très important, soit les *XML Server Pages (XSP)* et les *transformations XSLT*.

Les pages XSP permettent de créer à la volée du contenu à diffuser aux utilisateurs, et de gérer les interactions avec les utilisateurs, par exemple le contenu des formulaires de recherche.

La technologie XSP a été choisie pour différentes raisons, parmi lesquelles l'approche basée sur XML et XSLT, deux technologies déjà au coeur de SDX, et pour sa facilité à séparer le contenu de la logique et de la présentation.

A noter qu'avec un peu d'effort, il est possible d'écrire une application SDX sans utiliser de pages XSP, mais avec une autre technologie de pages dynamiques telle que les Java Server Pages ou même PHP4. Toutefois, puisque plusieurs interfaces avec le moteur SDX ont déjà été écrites en XSP, cette technologie est la plus appropriée pour le développement rapide d'applications SDX.

De son côté, la norme XSLT joue deux rôles importants dans une application SDX, à savoir :

1) Des transformations XSLT permettent de créer les pages dynamiques en Java depuis les pages XSP. Pour en savoir plus, **voir la documentation XSP** [http://xml.apache.org/cocoon/xsp.html] .
2) Des transformations XSLT sont utilisées pour convertir les documents XML et les pages dynamiques en format HTML pour diffusion sur le Web.

Avec l'expérience, on constate que XSLT joue un rôle crucial dans toute application SDX, et il est donc nécessaire de maîtriser ce langage pour créer et gérer des applications SDX.

Pour compléter votre apprentissage de SDX, vous pouvez **poursuivre cette vue d'ensemble**, ou vous pouvez consulter la documentation associée aux trois *API* présentes dans SDX : l'**API XSP**, l'**API URL** et l'**API Java**. Vous pouvez également consulter la section sur les **compétences et connaissances nécessaires** pour travailler avec SDX.

Développement d'applications SDX

Scénarios de navigation avec SDX

Afin de bien comprendre comment est gérée une navigation dans une base de documents SDX, nous allons décrire ce qui se passe du côté du serveur en utilisant deux scénarios simples de navigation.

Ces deux scénarios sont basés sur l'application SDX que vous êtes en train de consulter, soit la documentation.

Scénario 1 : consultation d'un document

Vous êtes en train de lire un document qui se situe à l'adresse (par exemple) **`http://sdx.culture.fr/sdx/sdxdoc/voir.xsp&id=t_scenarios`**. Ce que vous voyez dans votre navigateur Web, c'est une page HTML qui a été générée à partir d'un document XML géré par SDX. Cette transformation, de XML à HTML, a été effectuée côté serveur à l'aide de XSLT. Voici toutes les étapes du déroulement.

1) Votre navigateur a demandé l'URL en question au serveur **`sdx.culture.fr`** (par exemple).
2) Le serveur de la culture a identifié que la ressource identifiée par l'URL **`/sdx/sdxdoc/voir.xsp`** devait être gérée par des servlets, et a donc passé la requête au moteur de servlets.
3) Le moteur de servlets a identifié que les ressources dont l'extension est **`.xsp`** devaient être gérées par la servlet **`org.apache.cocoon.Cocoon`**, et elle lui passe donc la commande.
4) La servlet Cocoon reconnaît qu'il s'agit d'une page dynamique XSP. Elle va d'abord vérifier si elle a déjà une version compilée (en Java) de cette page dynamique, sinon elle va la compiler. Puis elle exécute le programme Java ainsi généré.

   La page **`voir.xsp`** est très simple, on vous la présente ici :

```
<?xml version="1.0"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="xsl/normal.xsl" type="text/xsl"?>
<xsp:page
language="java"
xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx"
>
  <sdx:page>
    <sdx:includeDocument/>
  </sdx:page>
</xsp:page>
```

   Pour bien comprendre cet exemple, il faut en savoir un peu plus sur les pages dynamiques XSP. Ces dernières sont des documents XML qui sont transformés en programmes Java. Ces derniers, lorsqu'ils s'exécutent, vont produire un nouveau document XML qui sera ensuite transformé pour être envoyé au navigateur. Dans notre exemple, le document XML produit par la page XSP sera nettement plus complexe que celui qui en constitue la source.

   Dans cette page XSP, on retrouve un élément **`sdx:page`** et un élément **`sdx:includeDocument`**. Le premier va permettre d'insérer dans la page XML en cours tout le contexte SDX nécessaire et le passer au moteur de transformation XSLT, et la deuxième instruction est tout ce qui est nécessaire pour dire à SDX que l'utilisateur veut afficher le document identifié par la valeur du paramètre **`id`**!

   Cette simplicité est seulement apparente, car derrière, dans la feuille logique, plusieurs instructions sont exécutées, et finalement le document XML construit sera celui que vous voyez à l'écran, mais avec à l'intérieur le document XML correspondant à l'identificateur **`t_scenarios`**. C'est ce nouveau document XML qui est transformé et que vous voyez à l'écran.
5) L'étape suivante consiste à transformer le document XML produit par la page XSP en un document HTML que vous pouvez consulter à l'aide d'un navigateur. Ceci est effectué à l'aide d'une transformation XSLT que l'on trouve dans le fichier **`xsl/normal.xsl`** et qui contient les instructions nécessaires pour traiter les informations SDX et le document lui-même.

   Par exemple, la barre de navigation dans le haut de la page n'existe pas dans le document affiché. Les informations requises sont extraites d'un autre document qui contient la table des matières de la documentation. Ce document est disponible lors de la transformation parce que XSLT permet d'inclure des documents XML externes en autant qu'ils soient adressables par une URL, mais aussi parce que SDX est capable de fournir un document XML directement en format XML à l'aide d'une simple requête exprimée par une URL.

Ce premier scénario permet d'exprimer une première caractéristique essentielle de SDX : une partie importante du travail consiste à créer des transformations XSLT. En effet, on constate que la page XSP est très simple, mais surtout qu'elle ne contient aucune ligne de code Java. De plus, on peut constater que la même page XSP peut être copiée pour n'importe quelle base de documents, car aucune instruction spécifique à cette base ne s'y trouve.

Scénario 2 : requête de recherche

Si vous tapez un mot dans le formulaire de recherche que l'on trouve en haut de cette page, vous obtiendrez des résultats de recherche. Ce scénario n'implique pas l'affichage d'un ou de plusieurs documents gérés par SDX, mais il utilise des techniques semblables à celles que l'on a vu au scénario précédent. Nous allons l'expliquer ici, en laissant tomber les étapes initiales visant à identifier la bonne servlet.

Ici, l'URL demandée est **`http://sdx.culture.fr/sdx/sdxdoc/rechsimple.xsp?q=xsp`** si vous tapez *sdx* dans la zone de texte.

1) La page XSP correspondant à cette URL est semblable à la suivante :

```
<?xml version="1.0"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="xsl/resultats.xsl" type="text/xsl"?>
<xsp:page
language="java"
xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx"
>
<sdx:page>
  <sdx:checkRights action="view_document">
   <resultPage type="simple">
    <sdx:executeSimpleQuery
      queryParam="q"
      pParam="p"
      hppParam="h"
      qidParam="n" parseParam="pp"/>
   </resultPage>
   <sdx:fallback>
    <identificationError/>
   </sdx:fallback>
  </sdx:checkRights>
 </sdx:page>
</xsp:page>
```

   Cette page semble plus complexe, mais c'est seulement parce qu'on nous avons inclus des instructions pour que SDX vérifie si l'utilisateur en cours a le droit d'effectuer cette opération, ainsi que les instructions à exéctuer s'il n'a pas le droit.

   Dans cette page, l'instruction qui permet de passer la requête de recherche au moteur SDX est exprimée par l'élément **`sdx:executeSimpleQuery`** et ses attributs, tout simplement. Les résultats, fournis en XML par SDX, seront inclus dans un élément nommé **`resultPage`** et traité par la transformation XSLT identifée par l'URL relative **`xsl/resultats.xsl`**.

2) Par la suite, la transformation XSLT pourra traiter le document XML qu'elle reçoit. Les résultats de recherche sont présentés en XML sous un format semblable à celle-ci :

```
<sdx:results nb="1">
  <sdx:result no="1">
    <sdx:field code="sdxdocid">introduction</sdx:field>
    <sdx:field code="titre">Introduction à SDX</sdx:field>
  </sdx:result>
</sdx:results>
```

Sans aller dans les détails, on voit que l'affichage de résultats de recherche est encore une fois une opération basée sur une transformation XSLT.

Conclusion

Une application SDX implique normalement des documents XML, leurs fichiers attachés (comme des images), mais aussi des pages XSP et des transformations XSLT. Les documents XML sont créés à l'extérieur de la plate-forme par les responsables de contenu. Les pages XSP et les transformations XSLT sont créées par les responsables de l'application.

L'application "Documentation" de SDX, que vous êtes en train de consulter, contient tous les éléments nécessaires pour effectuer les opérations de base avec SDX. Vous pouvez vous en inspirer pour créer vos propres applications. Cette documentation est fournie avec la **distribution SDX**.

Introduction aux concepts de base

Avant de se lancer dans le développement d'une application SDX, il est important de saisir quelques concepts de base. Cette section a pour but de vous les présenter.

Essentiellement, les concepts de base dont il faut discuter sont :

- **Les documents (XML) gérés par SDX**
- **Les documents associés aux documents XML**
- **L'importance de la recherche**
- **Les types de recherche**
- **Les utilisateurs et leurs droits**
- **Les documents HTML**

Les documents gérés par SDX

SDX est une plate-forme qui permet de diffuser des documents XML. Pour SDX, un document XML est un tout simplement un document XML qui respecte la recommandation du W3C. Ce document peut être bien formé (sans DTD) ou valide (qui respecte une DTD). L'encodage des caractères, s'il n'est pas UTF-16 ou UTF-8, doit être un encodage supporté par la machine virtuelle Java dans laquelle évolue SDX.

S'il s'agit d'un document valide, la référence à la DTD doit être une URL absolue, ou encore une URL relative qui sera résolue à l'aide d'un fichier de catalogue.

En bref, il n'y a aucune contrainte particulière concernant le type de documents XML que l'on peut gérer dans SDX. De plus, rien n'oblique à ce que tous les documents dans une base soient semblables, par exemple qu'ils respectent la même DTD.

SDX va donc pouvoir gérer ces documents XML, c'est-à-dire permettre des recherches, des tris, d'en ajouter et en supprimer, etc. Chaque base de documents doit procurer une interface utilisateurs permettant d'effectuer ces fonctions.

Les documents attachés

Un document XML ne peut pas, de façon générale, contenir des données binaires. La plupart du temps les documents binaires, comme les images, seront stockés dans des fichiers séparées, et des éléments, attributs ou entités dans le document XML permettront de *lier* ces documents au document XML.

Dans la terminologie XML, ces documents sont des *documents attachés*, et bien entendu SDX permet

de les prendre en charge. N'importe quel document qui peut être stocké dans un fichier informatique peut devenir un document attaché au sens SDX. La seule restriction est la taille du fichier, qui ne doit pas dépasser la taille des champs BLOB du gestionnaire de bases de données utilisé. En général, cette limite ne sera pas un problème.

Il n'y a pas de limite quant au nombre de documents attachés que l'on peut avoir pour un document XML. La seule contrainte est que ces documents attachés doivent avoir un identifiant unique parmi tous les documents attachés au même document XML.

Les documents attachés possèdent quelques propriétés, à savoir un identifiant, un type MIME, ainsi qu'une date de création et de modificaiton dans SDX. L'identifiant ainsi que les dates sont gérées automatiquement par SDX, le type MIME doit être fourni au moment de l'indexation du document XML.

L'importance de la recherche

Introduction

SDX est fondamentalement un outil de recherche. Mais parce que cet outil est puissant et souple, ces recherches deviennent également des mécanismes de gestion des documents et de leurs relations.

En fait, il existe trois grandes fonctions dans SDX :

Consulter un document

Cette fonction consiste à présenter à l'utilisateur l'un des documents gérés dans une base de documents SDX.

Effectuer une recherche et voir les résultats

Cette fonction consiste à exprimer des critères de recherche, à exécuter une recherche avec ces critères, puis à consulter tout ou partie des résultats obtenus.

Consulter les termes d'un index

Cette fonction consiste à voir tout ou partie des valeurs que contient un index, soit un champ de recherche.

Ce qui est à retenir, c'est que seule la première fonction peut être indépendante de l'outil de recherche sous-jacent à SDX. Les deux autres en dépendent fortement.

Le design d'une base de documents

L'architecture même de SDX fait en sorte que la principale, voire la seule tâche de design d'une base de documents consiste à identifier les champs de recherche que l'on aura besoin. Il faut comprendre la notion de champ de recherche au sens très large, car ce sont ces champs qui permettent :

- d'effectuer des recherches ;
- d'afficher des versions brèves des documents en résultat de recherche ;
- de trier les résultats d'une recherche ;
- de filtrer une requête de recherche, pour créer des sous-bases de documents par exemple.

Autrement dit, tout besoin d'affichage de formats brefs, de recherche, de tri ou de filtre doit se traduire par un ou des champs de recherche dans la configuration de la base de documents. Un grand soin doit donc être apporté à cette tâche.

Lorsque les champs de recherche sont bien définis, la base de documents devient une véritable *base de données XML*, ou presque.

Les types de recherche

SDX offre pour l'instant sept types de recherche, dont un qui est très générique. Ces types de recherche finissent toujours par donner une seule requête de recherche qui est fournie au moteur de recherche sous-jacent.

En plus de ces types de recherche, SDX prévoit la notion de filtre, de requête de base et de tri.

Les champs de recherche

En matière de recherche, un *document* pour SDX est une collection de *champs*. Chaque champ porte un nom (ou un code) et contient une valeur. Plusieurs champs peuvent porter le même nom, dans ce cas on parlera d'un champ multivalué ou à occurrences multiples.

Les informations sur la **configuration d'une base de documents** fournissent des explications sur les types de champs que l'on peut avoir dans SDX.

La recherche s'effectue toujours dans un ou plusieurs champs. En fait, chaque critère d'une recherche est associé à un champ. Puisqu'il existe un champ par défaut, on peut ne pas spécifier de champ dans les requêtes.

Par conséquent, il n'existe pas de notion de *recherche en texte intégral* dans SDX. En effet, on recherche toujours dans des champs, mais bien entendu un champ peut contenir l'ensemble du texte d'un document.

Le tri

Par défaut, les résultats de recherche sont triés par ordre décroissant de pertinence. La pertinence est calculée à l'aide de fonctions statistiques reliées à la présence des termes recherchés dans les documents.

Toutefois, il est possible de préciser un tri différent, en utilisant autant de clés de tri que l'on veut. Une clé de tri est toujours composée de deux informations : le nom d'un champ ainsi que l'ordre de tri (ascendant ou descendant). Si l'ordre n'est pas précisé, c'est l'odre ascendant qui est effectif.

Le tri est textuel (sauf pour les dates où il est chronologique), selon les critères de la langue française (tels qu'implantés par le langage Java).

Les filtres

Les filtres peuvent être vus comme des critères de recherche supplémentaires. Toutefois, ils constituent une manière simple et élégante de créer des zones dans la base de documents.

En effet, supposons qu'une base de données bibliographique contienne des références provenant de trois institutions. Il pourrait y avoir un champ qui indique la provenance, et on pourrait avoir besoin fréquemment de chercher des références dans une institution seulement.

Les filtres pourraient être appropriés, même si des critères de recherche supplémentaires pourraient aussi être utilisés.

Les filtres sont toujours une série de champs et de valeurs pour ce champ. L'ensemble de ces critères est relié par un opérateur logique ET ou OU.

Les requêtes de base

Les requêtes de base sont un mécanisme qui servent à effectuer des recherches en fonction des résultats d'une requête précédente. La plus grande utilité de cette fonction est de permettre à des utilisateurs de faire une recherche dans les résultats d'une recherche précédente.

Dans un tel scénario, la requête précédente devient la *requête de base*, et la nouvelle requête est la requête principale. De plus, l'opérateur qui relie ces deux requêtes est l'opérateur logique ET.

On pourrait aussi utiliser l'opérateur logique OU (pour ajouter d'autres résultats) ou l'opérateur lo-

gique SAUF (pour exlure des résultats).

Les types de recherche

Recherche simple

La recherche simple, malgré son nom, peut également être vue comme celle qui permet le plus grand nombre de possibilités. En fait, il s'agit d'une recherche où l'on spécifie directement la requête dans le langage propre à l'outil de recherche SDX. Cette requête peut être aussi simple qu'un seul mot, tout comme elle peut inclure plusieurs termes, différents champs, des mots requis ou interdits, des parenthèses pour regrouper, etc.

Il y a toutefois une limite à l'utilisation des recherches simples. En effet, les requêtes exprimées sont d'abord converties en mots avant d'être interprétées. Par conséquent, la recherche dans des champs de type *champ* ou de type *date* risque de donner des résultats décevants. Il est donc fortement conseillé d'utiliser la recherche simple dans des champs de type *mot*.

Le langage de requête de SDX est **décrit par ailleurs**, il est conseillé de s'y référer pour bien comprendre

Recherche restreinte

La recherche restreinte est semblable à la recherche simple, mais cette fois on va rendre tous les mots de la requête obligatoires, même si l'opérateur **+** n'est pas spécifié.

Recherche par champ

La recherche par champ est une recherche à un seul critère de recherche. Ce critère est composé d'un terme recherché et d'un champ où s'effectue la recherche.

Dans une recherche par champ normale (classe **`SDXFieldQuery`**, élément XSP **`sdx:executeFieldQuery`** ou servlet **`fieldsearch`**), les masques de troncature sont acceptés dans l'expression de recherche. Dans une recherche par champ exacte (classe **`SDXExactFieldQuery`**, élément XSP **`sdx:executeExactFieldQuery`** ou servlet **`exactfieldsearch`**), ils ne le sont pas.

Recherche par intervalle de dates

La recherche par intervalle de dates ne donnera des résultats intéressants que pour un champ de type date. Il s'agit d'une recherche où l'on peut spécifier une date de début ou une date de fin, voire les deux, et trouver des résultats dont le champ spécifié contient une date entre ces deux valeurs.

Si l'une des deux dates n'est pas spécifiée, la recherche n'est pas bornée. Si les deux dates ne sont pas spécifiées, il y aura une erreur.

Recherche linéaire

Une recherche linéaire est une recherche à plusieurs critères, mais tous les critères se suivent sur un même niveau. Ils sont reliés par un opérateur logique, et c'est l'ordre naturel de précédence qui déterminera la signification exacte de la recherche.

Ce type de recherche permet de faire des requêtes du genre :

```
region=Aquitaine ET departement=Gironde OU departement=Landes
```

Dans cet exemple, le ET l'emporte sur le ou, alors les résultats ne seront pas nécessairement ceux désirés.

Une autre façon de décrire ce type de recherche consiste à dire qu'il s'agit d'une requête booléenne sans la possibilité d'utiliser les parenthèses.

Recherche complexe

Une recherche complexe est en fait un recherche très générique qui est exprimée à l'aide d'un arbre de critères. Les noeuds de cet arbre sont soit des requêtes complexes ou un autre type de requête

(et dans ce cas il s'agit de feuilles).

Un noeud de type complexe relie des noeuds à l'aide d'un seul opérateur booléen. En combinant les noeuds correctement, on peut arriver à faire une requête booléenne avec tous les niveaux de parenthésage requis.

Les utilisateurs et leurs droits

Les utilisateurs

La plate-forme SDX inclut quelques fonctions de gestion des utilisateurs et de leurs droits. Ces fonctions sont simples puisqu'il ne s'agit pas d'une plate-forme de gestion. Toutefois, elles sont suffisantes pour répondre aux besoins de diffusion.

Un utilisateur SDX est connu pour l'ensemble de l'installation, il n'est pas possible, pour l'instant, de l'associer à une base de documents seulement. L'ajout, la suppresion et la modification des utilisateurs se fait dans l'interface de gestion de SDX.

Cet utilisateur est identifié par un code, un mot de passe (qui peut être vide), un prénom, un nom et une langue d'interface par défaut. Aucune autre information ne peut être ajoutée, et pour l'instant la langue n'est pas utilisée.

Il existe un utilisateur particulier, appelé **anonymous**, qui représente en fait tout utilisateur non identifié. Cet utilisateur n'a pas le droit d'alimenter les bases de documents.

Les groupes

Depuis la version 1.1, SDX permet de gérer les groupes d'utilisateurs. Cette gestion est très souple et permet au concepteur d'une application de déterminer des droits d'accès en fonction de différents critères.

Un groupe est identifié par son nom et sa description. La description est indicative seulement, dans les applications seul le nom est utilisé. Un groupe est nécessairement associé à une base de documents (contrairement aux utilisateurs qui sont associés à l'ensemble d'uns installation de SDX).

Un groupe peut contenir des utilisateurs et/ou d'autres groupes, sans limite quantitative. Un utilisateur doit d'abord être associé à la base de documents concernée avant de pouvoir faire partie d'un groupe.

Les droits

Les droits des utilisateurs sont gérés à l'aide de ce qu'on appelle des privilèges (il est toutefois préférable d'utiliser le mécanisme des groupes décrit ci-haut). Ainsi, par défaut, un utilisateur nouvellement créé ne peut rien faire de plus que l'utilisateur anonyme, si ce n'est de s'identifier.

Un privilège (et un seul) est accordé à un utilisateur pour une base de documents. Bien entendu, un utilisateur peut avoir un privilège dans plus d'une base.

Il existe quatre types de privilèges définis dans SDX :

- **s** (superuser) : identifie un administrateur système, qui a le droit de tout faire. Ce privilège est valide sur toutes les bases de documents, peu importe à laquelle il est associé. Dès qu'un utilisateur est administrateur SDX sur une base de documents, il l'est pour l'ensemble du système.
- **o** (owner) : le propriétaire d'une base de documents. Celui-ci peut ajouter des documents, de même que supprimer et modifier tous les documents.
- **w** (writer) : un utilisateur ayant le droit d'écrire, donc d'ajouter des documents. De plus, il peut modifier ou supprimer les documents qui lui appartiennent.
- **r** (reader) : un utilisateur ayant le droit de consulter la base de documents.

Les documents HTML

Depuis la version 1.1 de SDX, on peut importer et indexer des documents HTML. Cette fonctionnalité n'a pas pourobjectif de faire de SDX un outil de recherche de documents HTML, mais permet d'ajouter de tels types de documents lorsque nécessaire. Les possibilités sont limitées et le resteront.

Pour SDX, un document HTML est représenté par un petit document XML qui ressemble à celui-ci :

```
<?xml version="1.0"?>
<htmlDocument
  id="html/test.html"
  url="file:///usr/local/jakarat/webapps/sdx/test/html/test.html"
  path="/usr/local/jakarat/webapps/sdx/test/html/test.html"
>
```

Dans ce document, les attributs **id** et **url** seront toujours présents.

Comme on peut le constater, ce document XML contient un lien vers le document HTML qui doit être accessible par une URL. Pour indexer un tel document, on doit simplement utiliser une transformation XSLT qui ressemble à ceci :

```
  ...
  <xsl:param name="sdx_servletsUrl"/>
  ...
  <xsl:template match="/htmlDocument">
    <sdx:document id={@id}">
      <xsl:apply-templates
       select="document(concat($sdx_servletsUrl, '/html2xhtml?u=', @url))"/>
    </sdx:document>
  </xsl:template>
  ...
  <xsl:template match="html">
    <sdx:field code="titre"><xsl:value-of select="//title"/></sdx:field>
    <sdx:field code="contenu"><xsl:value-of select="."/></sdx:field>
  </xsl:template>
```

Cette transformation produire deux champs, un champ **titre** content l'élément **title** du document HTML, et un champ contenu contenant tout le texte du document. Cette partie est à adapter.

Cette indexation peut fonctionner parce que SDX inclut une servlet permettant de transformer un document HTML en format XHTML, et on peut donc le traiter à l'aide de la fonction **document()**.

xitLes bases de documents SDX

Introduction

Les bases de documents occupent une place importante dans une installation SDX. Celles-ci sont un peu l'équivalent des *bases de données* dans un système de gestion de bases de données. On voudra créer une base de documents pour une application par exemple, ou encore pour un ensemble de documents qui sont reliés d'une façon ou d'une autre.

Une application SDX peut gérer plusieurs bases de documents, et une base de documents peut gérer plusieurs types de documents sans qu'il y ait nécessairement des relations entre eux. La décision de créer une base de documents peut donc être prise en tenant compte d'un grand nombre de facteurs.

Les étapes nécessaires

Pour créer une base de documents, il faut procéder dans cet ordre :

1) Choisir un code pour la base de documents, code qui devrait être court, sans espace, et unique dans l'installation SDX.
2) Créer un document XML contenant la configuration de la base, ce document devant être mis dans un fichier **db_info.xml** (dont la structure est définie **par ailleurs**) qui se situe dans un répertoire **conf** qui lui-même se situe dans un répertoire nommé selon le code de la base de documents, et qui lui-même se situe à la racine de l'application SDX. Par exemple, pour une base de documents dont le code est *sdxdoc*, il faudra créer ce fichier :

```
{$TOMCAT_HOME}/webapps/sdx/sdxdoc/conf/db_info.xml
```

   Pour l'instant, on ne peut pas choisir un autre nom de fichier ou un autre emplacement. Il serait intéressant de prévoir une telle possibilité dans une version ultérieure de SDX.
3) Aller dans l'interface d'administration de SDX (en s'identifiant correctement), choisir le lien **Ajouter une base de documents** dans le bas de la page.
4) Dans le formulaire présenté, saisir le code de la base identifié au point 1, le nom de l'administrateur, ainsi que son mél. Cliquez ensuite sur **Envoyer**.

   Ces deux dernières informations ne sont pas utilisées de façon spécifique par SDX, mais elles sont disponibles aux applications, et éventuellement SDX pourrait en faire une utilisation.

A ce moment, la base de documents existe dans l'installation SDX, et si vous n'avez pas fait d'erreur dans votre fichier **db_info.xml**, la configuration est immédiatement connue de SDX et vous pouvez utiliser votre base.

Pour en savoir plus

L'étape importante est donc la création du fichier de configuration de la base de documents. Pour l'instant, ce document XML doit être créé extérieurement à SDX, avec un éditeur de texte ou un éditeur XML. Eventuellement, il serait intéressant de prévoir une interface Web pour la création d'une base de documents.

Pour connaître la structure de ce document de configuration, voir la **documentation associée**.

Le fichier de configuration d'une base de documents

Introduction

La configuration d'une base de documents consiste à consigner dans un document XML quelques informations générales que doit connaître SDX au sujet de votre base, mais surtout la liste des champs de recherche définis pour cette base de documents. Ce document XML doit avoir un élément nommé **`sdx:dbInfo`** et utiliser le domaine de nom **`sdx`** dont l'URI est **`http://www.culture.gouv.fr/ns/sdx/sdx`**.

La configuration

Les éléments de configuration sont essentiels. Ils sont inclus soit comme attribut de l'élément **`sdx:dbInfo`** soit dans le sous-élément **`sdx:configuration`**. Voici un exemple complet de la section de configuration :

```
<?xml version="1.0"?>
<sdx:dbInfo code="sdxdoc" xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx">
 <sdx:configuration>
  <sdx:languages>
    <sdx:language default="1" code="fr"/>
    <sdx:language code="en"/>
  </sdx:languages>
  <sdx:locales>
    <sdx:locale language="fr" country="FR"/>
```

```
    <sdx:locale language="en" country="UK"/>
  </sdx:locales>
  <sdx:lucene>
    <sdx:index path="_lucene"/>
    <sdx:analyzer class="fr.gouv.culture.sdx.index.FrenchStandardAnalyzer"/>
  </sdx:lucene>
  <sdx:catalogs>
    <sdx:catalog file="catalog"/>
  </sdx:catalogs>
  <sdx:extraction>
    <sdx:ssh file="index.xsl"/>
  </sdx:extraction>
  <sdx:transformation>
    <sdx:ssh file="transform.xsl"/>
  </sdx:transformation>
  <sdx:parameters>
    <sdx:parameter name="nbQueriesInSession" value="10"/>
    <sdx:parameter name="nbDocumentsToIndex" value="100"/>
  </sdx:parameters>
 </sdx:configuration>
...
</sdx:dbInfo>
```

Le code de base de documents

Le code de la base de documents doit être répété en attribut **code** de l'élément **sdx:dbInfo**. S'il ne s'y trouve pas, le fichier de configuration ne sera pas traité et la base ne sera pas active. N'oublions pas que ce code doit aussi correspondre à :

- Au nom du répertoire sous **webapps/sdx** où se situent vos fichiers de bases de documents.
- Au code de base de documents que vous avez utilisé ou que vous utiliserez dans l'interface d'administration SDX pour ajouter cette base de documents.

Les langues

SDX n'est pas complètement multilignue, mais certaines fonctions le sont et surtout il pourra l'être éventuellement. Il est possible de préciser des langues d'interface dans la configuration SDX, mais pas obligatoire. Si aucune langue n'est spécifiée, on considère la langue comme étant celle définie par défaut et passée en paramètre à l'environnement SDX grâce au fichier **web.xml**.

Si vous voulez préciser la ou les langues d'interface disponibles dans votre application, vous pouvez le faire à l'aide de l'élément **sdx:languages** et de ses sous-éléments **sdx:language**. L'attribut **default** indique qu'il s'agit de la langue par défaut, et l'attribut **code** doit contenir un code de langue valide selon la norme ISO 639.

Pour l'instant, aucune utilisation n'est faite de ces informations, mais elles sont stockées.

Les localisations

Ces informations doivent être comprises dans leur sens Java, tel que le définit, par exemple, la classe **java.util.Locale**. Pour l'instant, les informations de localisation sont uniquement utilisées lors du tri des termes d'une liste ou de résultats de recherche. De plus, seule la première information de localisation est prise en compte, les autres sont stockées mais ignorées.

Une information de localisation est constituée de deux parties, un code de langue ISO 639 et un code de pays ISO 3166. Par exemple, pour un tri selon la langue canadienne française, les informations seraient respectivement **fr** et **CA**.

Par défaut, les informations de localisation sont à **fr** et **FR**.

L'index Lucene

Lucene est l'outil de recherche utilisé par SDX. Il gère sa propre structure de fichiers inversés, et même si Lucene peut être un peu plus général, pour l'instant SDX exige que ces fichiers soient stockés dans un répertoire. L'élément **sdx:index** et son attribut **path** indiquent le chemin d'accès de ce répertoire, interprété de façon relative au sous-répertoire **conf** de la base de documents. Il n'est pas possible de le situer ailleurs, et la valeur par défaut est **_lucene**.

Par ailleurs, Lucene utilise un analyseur syntaxique pour extraire les mots des chaînes de caractères

qu'il indexe. Pour l'instant, un seul analyseur syntaxique est permis par base de documents, et la classe qui implante cet analyseur syntaxique peut être spécifiée dans l'élément **sdx:analyzer** et son attribut **class**.

Par défaut, c'est la classe **fr.gouv.culture.sdx.index.FrenchStandardAnalyzer** qui est utilisée. Cet analyseur effectue essentiellement trois opérations sur le texte : il sépare les mots, il supprime les mots vides et il transformes les lettres majuscules en lettres minuscules. Si on désire utiliser un analyseur qui désaccentue également les lettres, on doit spécifier la classe **fr.gouv.culture.sdx.index.FrenchUnaccentedAnalyzer**, également livrée avec SDX.

Si on spécifie une autre classe, elle doit implanter l'interface **com.lucene.analysis.Analyzer**.

Les catalogues d'entités externes

Les catalogues d'entités externes sont nécessaires lorsque les documents XML font référence à des DTD ou d'autres entités externes, et que ces entités externes ne sont pas appelées par une URL absolue. Dans ce cas, il faut indiquer à SDX et ses outils XML où se trouvent les entités externes, en fonction, par exemple, de leur chemin relatif.

Ces catalogues sont stockés dans des fichiers qui respectent le standard **OASIS** [http://www.oasis-open.org] . Vous n'êtes pas obligés d'utiliser des catalogues, et vous pouvez en spécifier autant que vous voulez, ils seront tous pris en compte. L'élément **sdx:catalog** contient en attribut **file** le nom du fichier de catalogue, qui doit se trouver dans le répertoire **conf**. Pour utiliser plusieurs catalogues, répétez l'élément **sdx:catalog**.

La transformation d'indexation

Les documents XML doivent être manipulés afin d'être indexés. Cette manipulation s'effectue à l'aide d'une feuille de style XSLT. L'élément **sdx:ssh** dans l'élément **sdx:extraction** contient en attribut **file** le nom du fichier contant la feuille de style XSLT. Encore une fois, cette feuille de style doit être située dans le répertoire **conf**.

La transformation initiale

Pour certaines bases de documents, il peut être préférable de transformer les documents XML avant leur indexation et leur manipulation par SDX. Si c'est le cas, on doit utilisé un élément **sdx:transformation** qui contient un élément **sdx:ssh**, l'attribut **file** de ce dernier indiquant la feuille de style XSLT utilisée pour cette transformation. A noter que dans une telle situation, l'indexation s'effectue sur le résultat de cette transformation et que les documents originaux et les documents transformés sont stockés dans la base de données. Toutes les opérations de consultation de SDX s'effectuent sur le document transformé, mais la sauvegarde s'effectue à partir des documents originaux.

Les paramètres

On peut fournir à SDX différents paramètres, à l'aide de l'élément **sdx:parameter**, le nom du paramètre étant spécifié dans l'attribut **name** et sa valeur dans l'attribut **value**. Les paramètres reconnus par SDX sont les suivants :

nbQueriesInSession

Le nombre de requêtes de recherche à conserver en mémoire dans l'objet de session. Les requêtes sont concervées afin de permettre de diviser les résultats en page et de ne pas refaire la requête à chaque changement de page. Cette requête étant stockée dans l'objet de session, elles sont nécessairement associées à un utilisateur en particulier.

Ce paramètre permet de gérer l'espace mémoire utilisé par SDX, car plus le nombre de requêtes conservées est élevé, plus les besoins en mémoire sont importants. Par défaut, le nombre de requêtes conservées est de 5. Si vous utilisez ce paramètre et que vous donnez comme valeur un nombre entier, ce nombre remplacera la valeur par défaut.

nbDocumentsToIndex

Lorsque SDX charge des documents, il effectue l'indexation par lots de documents plutôt qu'individuellement, et ce afin d'accélérer le processus. Par défaut, le nombre de documents par lot est de 50, mais vous pouvez modifier cette valeur avec ce paramètre.

Plus le nombre de documents par lot est élevé, plus l'indexation sera rapide, mais aussi plus les utilisateurs en consultation risquent d'être perturbés, et plus la mémoire nécessaire sera grande. En utilisation normale, le nombre de documents devrait être ajusté de façon à ce que l'indexation d'un lot puisse se faire en quelques secondes. Lors d'une opération de chargement d'un grand nombre de documents, il peut être augmenté préalablement.

Les informations générales

Les informations générales fournissent essentiellement le nom et une courte description de la base de documents. Ces informations peuvent être dans plusieurs langues, et elles sont restituées dans les documents XML dynamiques générés pas SDX et passés aux feuilles de style. Toutefois, seuls les noms et descriptions dans la langue d'interface en cours sont insérés dans ces documents XML dynamiques.

Ces informations sont spécifiées de cette façon :

```
<?xml version="1.0"?>
<sdx:dbInfo code="sdxdoc" xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx">
 <sdx:configuration>
  ...
 </sdx:configuration>
 <sdx:informations>
  <sdx:names>
    <sdx:name xml:lang="fr">Documentation SDX</sdx:name>
    <sdx:name xml:lang="en">SDX documentation</sdx:name>
  </sdx:names>
  <sdx:descriptions>
    <sdx:description xml:lang="fr">La documentation utilisateurs et
       technique de SDX.</sdx:description>
    <sdx:description xml:lang="en">SDX technical and user
       documentation.</sdx:description>
  </sdx:descriptions>
</sdx:informations>
...
</sdx:dbInfo>
```

Les constructions XML standards pour l'identification des langues sont utilisés (attribut **`xml:lang`**).

Les champs de recherche

Les champs de recherche constituent le coeur du fonctionnement SDX. Ils permettent non seulement la recherche d'information, mais aussi de considérer les documents comme des bases de données.

Les champs sont définis dans le fichier de configuration, dans un élément **`sdx:fieldList`** qui contient une série d'élément **`sdx:field`**. Pour chaque champ, les informations suivantes peuvent être spécifiées :

Code

Le code du champ, ou son nom si on préfère. Ce nom doit être unique dans la base de documents. De plus, il semble que les caractères _ (underscore) ne sont pas acceptés. Il est donc préférable de ne pas utiliser de caractères spéciaux, ni de lettres accentuées.

Le code est spécifié à l'aide de l'attribute **`code`** de l'élément **`sdx:field`**. Il est obligatoire.

Format bref

La valeur d'un champ peut être incluse dans le format bref de présentation du document. Ce format bref est utilisé lors de la présentation des résultats d'une recherche. Par défaut, un champ n'est pas inclus dans le résultat bref.

Pour inclure le champ, il faut ajouter un attribut **`brief`** à l'élément **`sdx:field`**, et lui donner une valeur *vraie*, par exemple **`1`** ou **`true`**.

Champ par défaut

Il existe un concept de champ par défaut dans SDX. Celui-ci est utilisé lorsqu'aucun champ n'est spécifié dans les requêtes de recherche. Très souvent, on utilisera un champ de type "plein texte" comme champ par défaut.

Pour définir un champ par défaut, il faut ajouter un attribut **`default`** à l'élément **`sdx:field`**, et lui donner une valeur *vraie*, par exemple **`1`** ou **`true`**. Un seul champ par défaut peut être défini, si l'attribut **`default`** est utilisé dans plusieurs éléments, seul le dernier sera pris en compte.

Type de champ

Il existe quatre types de champs dans SDX. Ces types sont :

1) **Mot** : la valeur du champ est analysée et les mots sont extraits, chaque mot devenant un terme cherchable.

   Pour un tel champ, la valeur de l'attribut **`type`** doit être **`word`**.

2) **Champ** : la valeur du champ est laissée telle quelle.

   Pour un tel champ, la valeur de l'attribut **`type`** doit être **`field`**.

3) **Date** : la valeur du champ est convertie en date, et la recherche d'intervalles de dates est alors possible, de même que la recherche de dates précises. Les dates possibles vont de l'an 0 jusqu'à l'an 20 000, la précision peut être une année, un mois, une journée, une heure, une minute ou une seconde.

   Les formats de date acceptés sont de la forme **`aaaa/mm/jj hh:mm:ss`** ou **`aaaa-mm-jj hh:mm:ss`**. On peut s'arrêter à n'importe quel groupe, par exemple une date telle que **`1999/04`** sera interprétée correctement comme le mois d'avil 1999.

   Pour un tel champ, la valeur de l'attribut **`type`** doit être **`date`**.

4) **Non indexé** : le champ ne sera pas indexé, donc non cherchable, mais il sera retourné dans les résultats de recherche. De plus, il pourra être utilisé comme clé de tri.

   Pour un tel champ, la valeur de l'attribut **`type`** doit être **`unindexed`**.

Il est prévu, éventuellement, d'ajouter un type de champ numérique qui permettra de faire des recherches par intervalle.

De plus, les éléments **`sdx:field`** peuvent contenir des sous-élément **`sdx:name`** qui indiquent le nom du champ dans autant de langues que l'on désire. Ces noms sont stockés par SDX, mais ne sont pas utilisés pour l'instant.

Par ailleurs, l'élément **`sdx:fieldList`** peut prendre un attribut **`ignoreCase`** qui, si sa valeur est vraie (**`vrai`**, **`true`**, **`1`**), signifie que la casse des noms de champ n'est pas significative. A noter que dans un tel cas, l'utilisation de l'API Java doit se faire avec attention, car tout ce qui concerne les requêtes de recherche ne tient pas compte de la casse, mais si on manipule directement les termes d'un index, les filtres, les critères, etc., alors on doit fournir le nom des champs dans la casse dont ils ont été définis.

Voici un exemple partiel d'un fichier de configuration, qui définit quelques champs de recherche :

```
<?xml version="1.0"?>
<sdx:dbInfo code="sdxdoc" xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx">
 <sdx:configuration>
  ...
 </sdx:configuration>
 <sdx:informations>
  ...
 </sdx:informations>
 <sdx:fieldList>
  <sdx:field code="fulltext" type="word"default="1">
    <name xml:lang="fr">Texte intégral</name>
    <name xml:lang="en">Full text</name>
  </sdx:field>
  <sdx:field code="title" type="field" brief="1">
    <name xml:lang="fr">Titre</name>
    <name xml:lang="en">Title</name>
```

```
  </sdx:field>
  <sdx:field code="modification" type="date">
    <name xml:lang="fr">Date de dernière modification</name>
    <name xml:lang="en">Last modification date</name>
  </sdx:field>
  <sdx:field code="author" type="field">
    <name xml:lang="fr">Auteur</name>
    <name xml:lang="en">Author</name>
  </sdx:field>
 </sdx:fieldList>
</sdx:dbInfo>
```

Les champs systèmes

Il existe huit champs de recherche par défaut dans SDX, et il est interdit de réutiliser les mêmes noms. Ces champs et leur signification sont :

sdxdocid

L'identifiant du document. Unique dans toute la base de documents. Il s'agit d'un champ de type **`champ`**.

sdxstatus

Le statut du document, les valeurs de ce statut sont gérées par les applications et non par SDX. Il s'agit d'un champ de type **`champ`**.

sdxmodificationdate

La date de modification du document dans la base, information externe au document lui-même et gérée par SDX. Il s'agit d'un champ de type **`date`**.

sdxcreationdate

La date de création du document dans la base, information externe au document lui-même et gérée par SDX. Il s'agit d'un champ de type **`date`**.

sdxindexed

Un champ booléen qui indique si le document est indexé ou non. Il s'agit d'un champ de type **`champ`** dont les valeurs peuvent être **`true`** ou **`false`**.

sdxowner

Le code d'utilisateur du propriétaire de ce document. Il s'agit d'un champ de type **`champ`**.

sdxscore

Ce champ est créé seulement pour permettre des tris de façon uniforme. Il contient l'indice de pertinence du document pour une requête, et varie donc avec le temps. Il ne faut pas manipuler ce champ.

sdxall

Ce champ permet de retrouver tous les documents d'une base à l'aide d'une requête de recherche. Tous les documents ont la valeur **1** pour ce champ.

Ces champs ne doivent pas faire partie du fichier de configuration, ils sont systématiquement créés et alimentés par SDX.

L'indexation des documents XML

Introduction

L'indexation de documents à des fins de recherche peut être une entreprise complexe, surtout si on doit indexer des documents textuels et si on veut exploiter la structure. Les outils pour faire ce travail proprement sont rares, en particulier dans le monde des logiciels libres. C'est pourquoi SDX utilise une approche indirecte, approche qui peut s'avérer limitée mais qui possède aussi ses avantages.

Cette approche indirecte consiste à utiliser la structure des documents *a priori*, c'est-à-dire au moment de leur indexation (plutôt qu'au moment de la requête de recherche). Ainsi, toute la structure des documents XML peut être exploitée en recherche, mais en autant que cela ait été prévu au départ, par le concepteur de la base de documents.

Par exemple, si on veut que les utilisateurs puissent chercher des informations dans un champ *auteur*, alors il faut avoir prévu un champ auteur dans la configuration de la base de documents, et aussi alimenter ce champ auteur lors de l'indexation des documents.

La définition des champs est faite dans la **configuration de la base de documents**. L'alimentation de ces champs se fait lors de l'ajout des documents dans une base, en appliquant une transformation XSLT sur ces documents, transformation qui a pour objectif d'extraire les données à indexer du document et les fournir à SDX.

Scénario d'alimentation

Voici un exemple concret d'alimentation d'une base de documents. Supposons que la base contienne la définition de ces champs :

```
<sdx:fieldList>
  <sdx:field code="texte" type="word" default="1"/>
  <sdx:field code="titre" type="field" brief="1"/>
  <sdx:field code="categorie" type="field" brief="1"/>
</sdx:fieldList>
```

Dans cette base, nous voulons ajouter ce document :

```
<?xml version="1.0"?>
<doc id="d001" categorie="a2">
  <titre>Un essai d'indexation</titre>
  <corps>
    <p>L'indexation avec SDX est très facile, mais
       surtout elle est très souple!</p>
  </corps>
</doc>
```

La feuille de style XSLT qui permet l'indexation pourrait avoir cette allure :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx"
>
  <xsl:template match="/">
    <sdx:document id="{doc/@id}">
      <sdx:field code="texte"><xsl:value-of select="."/></sdx:field>
      <sdx:field code="titre">
        <xsl:value-of select="normalize-space(doc/titre)"/>
      </sdx:field>
      <sdx:field code="categorie">
        <xsl:choose>
          <xsl:when test="doc/@categorie = 'a1'">
            Informatique
          </xsl:when>
          <xsl:when test="doc/@categorie = 'a2'">
            XML
```

```
          </xsl:when>
          <xsl:when test="doc/@categorie = 'a3'">
            Java
          </xsl:when>
        </xsl:choose>
      </sdx:field>
    </sdx:document>
  </xsl:template>
</xsl:stylesheet>
```

L'application de cette feuille de style au document plus haut donnera comme résultat :

```
<?xml version="1.0"?>
<sdx:document id="d001" xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx">
  <sdx:field code="texte">
   Un essai d'indexation L'indexation avec SDX est très facile, mais
       surtout elle est très souple!
  </sdx:field>
  <sdx:field code="titre">
    Un essai d'indexation
  </sdx:field>
  <sdx:field code="categorie">XML</sdx:field>
</sdx:document>
```

C'est ce dernier document qui sera donc fourni au moteur d'indexation de SDX. Pour ce document, les champs *texte*, *titre* et *categorie* auront les valeurs indiquées dans le document. Ils seront indexés correctement, en fonction de leur type.

Le point clé est donc la feuille de style pour l'indexation, et le résultat qu'elle doit produire, deux aspects documentés **par ailleurs**.

Quelques remarques

Le scénario précédent illustre quelques concepts reliés à l'indexation dans SDX. Voici quelques remarques à ce sujet.

- L'utilisation de la structure se fait au moment de l'indexation, et cette utilisation de la structure est limitée seulement par la puissance expressive de la norme XSLT, ce qui est intéressant.
- La norme XSLT prévoit une fonction appelée **`document()`** qui permet d'inclure dans le traitement des informations provenant de n'importe quel flux XML (par exemple un fichier) que l'on peut obtenir par une adresse URL. Ainsi, dans l'exemple précédent, on aurait pu remplacer la logique sur la catégorie par un fichier de définition des catégories, en XML, et utiliser ce fichier pour inclure des données complètes plutôt que des codes.
- SDX peut indexer non seulement le contenu des documents XML, mais également du **contenu généré**. Ce mécanisme peut être très puissant, et l'exemple des catégories l'exprime quelque peu. Ainsi, dans une base SDX, on peut faire des recherches sur des mots et trouver des résultats, même si aucun document ne contient ces mots !

La transformation préalable des documents

Présentation

Dans certaines situations, il peut être préférable de demander à SDX de gérer un document légèrement différent de celui qu'on lui fournit. Par exemple, pour supprimer l'appel à une DTD ou pour résoudre certains liens au préalable, améliorant ainsi les performances.

SDX offre cette fonctionnalité nativement. Pour ce faire, on doit spécifier l'élément **`sdx:transformation`** dans le **fichier de configuration**, et avec celui-ci indiqué une feuille de style XSLT qui fera la transformation. Cette transformation se déroule **avant** l'indexation, et cette der-

nière **opère sur le document transformé**, et non sur le document original.

De plus, lorsqu'on demande à SDX de consulter un document, c'est le document transformé qui est envoyé, et non le document original. Ce dernier est utilisé seulement lors d'une opération de sauvegarde de la base de documents.

La transformation peut être quelconque, elle doit seulement être écrite en XSLT et produire un document XML bien formé en sortie.

La feuille de style d'indexation

Présentation

La feuille de style d'indexation permet de transformer un document XML en un autre document XML. Le premier est le document XML géré par SDX. Le deuxième est un document XML transitoire qui sert à l'indexation du document dans la base. **Une autre partie de la documentation** décrit ce processus.

La feuille de style peut être structurée de n'importe quelle façon en autant qu'elle produise un document XML qui respecte le format de sortie défini par après.

Format de sortie

Le document XML produit par la feuille de style doit repecter la DTD suivante :

```
<!ELEMENT sdx:document    ( sdx:field |
                            sdx:attachedDocument
                            sdx:table )*               >
<!ATTLIST sdx:document
              id           CDATA        #REQUIRED      >
<!ELEMENT sdx:field       ( #PCDATA )                  >
<!ATTLIST sdx:field
              code         CDATA        #REQUIRED      >
<!ELEMENT sdx:attachedDocument  EMPTY                  >
<!ATTLIST sdx:attachedDocument
              id           CDATA        #REQUIRED
              filename     CDATA        #REQUIRED
              url          CDATA        #IMPLIED
              mimetype     CDATA        #IMPLIED       >
<!ELEMENT sdx:table       ( sdx:column+ )              >
<!ATTLIST sdx:table
              name         CDATA        #REQUIRED      >
<!ELEMENT sdx:column      ( #PCDATA )                  >
<!ATTLIST sdx:column
              name         CDATA        #REQUIRED      >
```

L'identifiant du document

L'attribut **`id`** de l'élément supérieur **`sdx:document`** doit contenir un identifiant qui sera unique dans la base de documents. C'est donc à la feuille de style XSLT de créer ou d'extraire ces identifiants uniques. S'il n'y a pas d'attribut **`id`**, le document ne sera pas ajouté. Si l'attribut **`id`** contient une valeur égale à l'identifiant d'un document dans la base, deux choses peuvent se produire :

1) Si l'ajout se fait en mode de **remplacement**, alors le document ayant le même identifiant est supprimé de la base, et il est remplacé par le nouveau. La date de création ne changera pas.
2) Si l'ajout ne se fait pas en mode de remplacement, une erreur sera générée et le document sera rejeté.

Cet identifiant peut être quelconque. Toutefois, il est préférable d'utiliser un identifiant simple et surtout sans caractères spéciaux, tous les traitements seront facilités par la suite, en particulier le traitement des paramètres d'URL.

Les valeurs de champs

Le document XML produit peut contenir un nombre quelconque d'éléments **sdx:field**. Chaque occurrence d'un tel élément correspond à une occurrence du champ dont le code est donné par la valeur de l'attribut **code**. Si ce champ n'existe pas dans la base, le champ sera rejeté mais le document sera toutefois accepté.

Le contenu de l'élément est le contenu du champ. Pour les champs de type **field**, **date** et **unindexed**, le contenu sera interprété tel quel, sans aucune transformation. Il faut donc s'assurer de bien gérer les espaces lors de la transformation. Pour les champs de type **mot**, une analyse lexicale sera opérée, alors cet aspect est moins critique.

Les valeurs de tables externes

SDX permet de gérer des tables relationnelles extérieures aux document et à sa propre indexation, et d'y stocker des valeurs. L'alimentation se fait également au moment de la transformation, et les informations sont stockées dans des éléments **sdx:table** et **sdx:column**.

Pour chaque occurrence de l'élément **sdx:table**, une entrée sera faite dans la table externe dont le nom est donné par l'attribut **name**. Dans cette entrée, toutes les colonnes qui ont un sous-élément **sdx:column** dont l'attribut **name** correspond seront remplies.

C'est à la transformation de s'assurer que les valeurs fournies respectent la structure de la table définie dans le fichier de configuration.

Les documents attachés

Un document XML ne vient pas toujours seul. Ainsi, il peut y avoir des documents qui lui sont attachés, par exemple des images. SDX permet de gérer ces documents attachés, en les stockant dans la base de données et en s'assurant qu'ils seront supprimés lorsque le document XML est supprimé de la base.

Il est important de noter que cette procédure n'est pas obligatoire. Les documents attachés peuvent être gérés par ailleurs et appelés, par exemple, par une URL appropriée lors de l'affichage.

Si on décide de gérer les documents attachés à l'aide de SDX, on doit produire un élément **sdx:attachedDocument** pour chaque document attaché associé au document en cours de traitement. Aucune limite n'est imposée sur le nombre de documents attachés.

Cet élément doit nécessairement fournir un identifiant pour le document attaché. Cet identifiant peut être quelconque, en autant qu'il soit unique parmi tous les documents attachés associés au document XML en cours de traitement. Il est recommandé, encore une fois, d'utiliser des identifiants simples et sans caractères spéciaux.

Le nom de fichier du document attaché doit être spécifié dans l'attribut **filename**. Ce nom de fichier servira lors de la sauvegarde d'une base de documents. Ce nom de fichier doit être sans chemin d'accès, par exemple **img01.jpg**.

On peut également spécifier l'URL du document attaché, soit de façon relative au document XML, soit de façon absolue. Cette URL est spécifiée à l'aide de l'attribut **url**, et si cet attribut est présent il est utilisé en priorité pour retrouver le document attaché. S'il n'est pas spécifié, c'est l'information fournie par l'attribut **filename** qui indiquera à SDX où trouver le fichier.

L'élément peut également fournir des informations sur le type MIME du document attaché, ce qui est également recommandé car cela permet d'envoyer cette information au client lorsqu'il demande un document attaché.

Introduction

SDX se veut une architecture ouverte et modulaire. Pour y arriver, une des méthodes employées est la mise en disponibilité de ses services à l'aide de trois API (*application programming interface)*, que nous introduisons brièvement ici.

L'API XSP

La **technologie XSP** [http://xml.apache.org/cocoon/xsp.html] (*XML server pages*) est au coeur du fonctionnement de Cocoon et a été reprise dans SDX. Elle permet de créer des pages Web dynamiques en séparant aisément le contenu de la logique et de la présentation.

L'**API XSP** de SDX se présente sous la forme d'une feuille logique (*logicsheet*) ou encore une bibliothèque de balises (*taglib*). Ainsi, les pages dynamiques d'une application SDX sont normalement des pages XSP, qui ont accès à tous les services offerts par cette technologie, par les bibliothèques de balises fournies avec Cocoon ou externes, mais également à la bibliothèque de balises SDX.

L'API URL

Plusieurs services SDX sont disponibles à l'aide d'une **API basée sur des URL** (*uniform resource locator*). Il est donc possible d'effectuer des opérations, par exemple une recherche d'information, en appelant une adresse URL.

La raison d'être de cette API est double :

1) Permettre aux feuilles de style XSLT utilisées dans les applications SDX d'avoir accès à des informations externes au document qu'elles traitent, y compris des informations dynamiques.
2) Permettre à des applications non gérées par SDX, éventuellement distantes, d'avoir accès aux services SDX sans difficulté.

Il est à noter que tous les résultats offerts par l'API URL de SDX sont en format XML.

L'API Java

Fondamentalement, SDX est constitué d'une série de classes Java qui font partie d'un ensemble de classes associées, par exemple le moteur de servlets, le parseur XML, le processeur XSLT, etc. La plupart des classes Java de SDX sont publiques, et peuvent donc être utilisées directement par les applications SDX.

Cette API est celle qui se situe le plus près du code SDX. Par conséquent, elle est aussi celle qui est la plus susceptible d'évoluer.

Présentation de l'API XSP

L'API XSP permet de créer facilement des pages Web dynamiques dans des applications SDX. Les pages dynamiques sont d'ailleurs au coeur de l'approche SDX, cette plate-forme étant orientée vers la diffusion de documents XML *en contexte*, le contexte étant très souvent dynamique, comme par exemple le nom de l'utilisateur qui demande une page.

C'est pourquoi la bibliothèque de balises XSP fournie avec SDX est très développée, et permet, à l'aide de simple code XML, d'inclure des fonctionnalités assez avancées telles que l'ajout d'un document, la recherche d'information, etc.

Quelques mots sur XSP

La technologie XSP est au coeur du projet **Cocoon** [http://xml.apache.org/cocoon/] . Pour en savoir plus à ce sujet, il est conseillé de consulter la **documentation associée à ce projet** [http://xml.apache.org/cocoon/xsp.html] . SDX est basé sur la version 1.8.2 de Cocoon, et il est bon de noter que la version 2 de Cocoon est sensiblement différente et non compatible. La **version 2** de SDX devrait éventuellement s'adapter à cette nouvelle architecture.

Essentiellement, le but d'une page XSP est de produire un document XML qui sera par la suite traité par une feuille de style XSLT avant d'être envoyé à l'utilisateur. Dans le modèle de traitement Cocoon, cette technologie est utilisée afin de préparer des documents XML qui ne peuvent être connus à l'avance, et donc des pages dynamiques.

La transformation d'une page XSP (qui est, rappelons-le, un document XML) vers le document XML final est effectuée par le processeur XSP inclus dans Cocoon, et cette transformation est effectuée à

l'aide de XSLT.

Afin de rendre ce processus efficace, le résultat de la page XSP et de sa transformation est converti en une classe Java qui sera compilée et donc réutilisable. Cette classe contiendra les instructions nécessaires pour créer dynamiquement le document XML à traiter.

Autrement dit, le résultat final d'une page XSP est une classe Java. D'ailleurs, il est possible d'inclure du code Java dans une page XSP, et ce code sera directement intégré dans la classe Java produite.

Par conséquent, on peut dire que l'on a accès à tout le langage Java dans une page XSP, et à toutes les classes publiques disponibles dans l'environnement Java en cours. On peut donc définir des pages dynamiques fort complexes.

Les bibliothèques de balises

Les bibliothèques de balises en XSP permettent de définir des structures XML qui seront utilisées fréquemment, sans avoir à les recopier dans toutes les pages XSP qui en ont besoin.

Par exemple, supposons que dans une application, plusieurs pages XSP ont besoin d'inclure la date actuelle dans un élément **`date`**. Cette fonctionnalité pourrait être incluse dans une bibliothèque de balises de la sorte :

```
<xsl:template match="sdx:includeDate">
  <xsp:logic>
    Element dateElement = document.createElement("date");
    java.util.Date today = new java.util.Date();
    dateElement.appendChild(document.createTextNode(today.toString());
    xspCurrentNode.appendChild(dateElement);
  </xsp:logic>
</xsl:template>
```

Ensuite, dans une page XSP, nous pourrions include cet élément XML :

```
<sdx:includeDate/>
```

et automatiquement le document XML produit par la page XSP contiendrait un élément **`date`** à l'endroit où l'on a inséré l'élément **`sdx:includeDate`**. Par exemple, la page XSP suivante :

```
<?xml version="1.0"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="index.xsl" type="text/xsl"?>
<xsp:page language="java"
 xmlns:xsp="http://www.apache.org/1999/XSP/Core"
 xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx"
>
 <document>
  <sdx:includeDate/>
 </document>
</xsp:page>
```

donnera comme résultat le document XML suivant :

```
<?xml version="1.0"?>
<document>
  <date>2001/10/04 14:35:34</date>
</document>
```

Localisation de l'API XSP de SDX

L'API XSP de SDX est constitué d'une bibliothèque de balises. Cette bibliothèque de balises doit être

déclarée dans le fichier **cocoon.properties** (le fichier de configuration fourni avec SDX contient cette déclaration), et le domaine de nom qui lui est associé dans les pages XSP doit y être défini, comme dans les exemples précédents.

Les biblitothèques de balises sont des transformations XML. La bibliothèque SDX est fournie dans le fichier **aa1_sdx.jar** que l'on trouve dans le répertoire **webapps/sdx/WEB-INF/lib** de la distribution. Pour l'extraire, vous pouvez utiliser un utilitaire ZIP ou encore le programme **jar** fourni avec le JDK Java. Dans le fichier JAR, vous trouverez la bibliothèque de balises dans le fichier **fr/gouv/culture/sdx/taglibs/sdx.xsl**.

Documentation de l'API XSP

L'API XSP se présente sous la forme d'une série d'éléments ou de groupes d'éléments XML qui permettent d'effectuer des opérations. Nous allons les présenter un par un ici.

sdx:page

Il s'agit de l'élément qui doit contenir l'ensemble des informations à inclure dans le document XML dynamique, si l'on veut que l'environnement SDX soit correctement généré.

Cet élément permet de générer une structure de données qui commence avec l'élément **sdx:document [dtd/html/elements/document.html]** . De plus, il permet d'initialiser certaines variables globales comme l'environnement SDX, l'objet de configuration de la base de documents, etc.

Les instructions SDX ou XSP que l'on retrouve à l'intérieur de cet élément seront traitées, et les autres éléments seront copiés dans le document XML dynamique.

sdx:checkRights, sdx:userIsMember et sdx:fallback

Ces éléments permettent de construire une partie du document XML en fonction du droit de l'utilisateur en cours (**sdx:checkRights**) ou de son apprtenance à un groupe (**sdx:userIsMember**). Essentiellement, le contenu de cet élément sera traité seulement si l'utilisateur en cours a le droit d'effectuer l'action identifiée par son attribut **action** ou encore s'il fait partie du groupe identifié par l'attribut **group** ou le paramètre spécifié par l'attribut **groupParam** de l'élément **sdx:userIsMember**.

Si l'utilisateur n'a pas le droit d'effectuer cette action, alors seulement le sous-élément **sdx:fallback** de cet élément sera traité. Bien entendu, cet élément n'est pas traité dans le cas où l'utilisateur possède les droits requis.

Par ailleurs, certaines actions ont des droits qui dépendent du document XML concerné. C'est le cas, par exemple, de l'action de suppression ou de modification du document. Dans ce cas, l'identifiant du document doit être présent dans les paramètres de la requête, et le paramètre qui le contient doit être spécifié dans l'attribut **param** de l'élément. S'il n'est pas spécifié, alors on utilise le paramètre **id**.

Les actions associées à **sdx:checkRights** sont identifiées par des codes textuels. Voici les codes qui sont définis dans SDX :

superuser	Vérifie si l'utilisateur est un administrateur SDX. Il s'agit d'une action d'identification seulement.
anonymous	Vérifie si l'utilisateur est anonyme, donc non identifié. Il s'agit d'une action d'identification seulement.
add_db	Ajouter une base de documents dans cette installation SDX. Seuls les administrateurs peuvent le faire.
edit_db	Modifier une base de documents dans cette installation SDX (n'importe quelle base). Seuls les administrateurs peuvent le faire.
delete_db	Supprimer une base de documents dans cette installation SDX (n'importe quelle base). Seuls les administrateurs peuvent le faire.

add_document	Ajouter un document dans la base courante. On doit avoir les privilèges **o** ou **w** pour pouvoir le faire.
edit_document	Modifier un document dans la base courante. On doit avoir le privilège **o** ou encore avoir le privilège **w** tout en étant propriétaire du document.
delete_document	Supprimer un document dans la base courante. On doit avoir le privilège **o** ou encore avoir le privilège **w** tout en étant propriétaire du document.
view_document	Voir un document. Tous les utilisateurs peuvent le faire.
edit_user	Modifier un utilisateur. Seuls les administrateurs peuvent le faire.
add_user	Ajouter un utilisateur. Seuls les administrateurs peuvent le faire.
delete_user	Supprimer un utilisateur. Seuls les administrateurs peuvent le faire.

Pour les groupes, on peut utiliser une construction telle que :

```
<sdx:userIsMember group="administrateurs">
 <edition/>
 <sdx:fallback>
   <erreur/>
 </sdx:fallback>
</sdx:user>
```

sdx:insertXMLFile

Cet élément permet d'insérer un fichier XML dans le document XML dynamique en cours de construction. Ce fichier doit être accessible dans le répertoire de la base de documents ou dans l'un de ses répertoires.

Le chemin d'accès (relatif au répertoire de la base de documents) doit être spécifié dans l'attribut **name**.

sdx:handleLogout

Cet élément permet de vérifier si un utilisateur veut se déconnecter (c'est-à-dire devenir anonyme). Si oui, l'opération est effectuée, si non aucun changement n'est apporté. Cet élément ne génère jamais de contenu XML, il ne fait que modifier l'objet utilisateur (**sdxUser**).

On identifie un utilisateur qui veut se déconnecter par la présence d'un paramètre **logout** dans la requête dont la valeur est quelconque mais non nulle.

sdx:showTable

Cet élément permet l'insertion d'une partie du contenu d'une table externe de la base de documents. L'attribut **tblParam** contient le nom du paramètre identifiant la table, et l'attribut **whereParam** contient le nom du paramètre qui identifie la clause WHERE à utiliser pour sélectionner des enregistrements dans la table.

Par exemple, cet élément :

```
<sdx:showTable tblParam="t" whereParam="w"/>
```

utilisé pour une requête qui se termine par :

```
?t=ress&w=parent='c21'
```

permettra d'insérer tous les enregistrements de la table **ress** qui respectent la clause **WHERE parent='c1'**.

sdx:uploadDocument

Note : cet élément s'appelait autrefois **`sdx:uploadXMLFile`**. Cet ancien nom est toujours valide, mais ne doit pas être utilisé.

Cet élément permet de démarrrer l'ajout d'un document XML ou HTML et de ses documents attachés depuis un formulaire Web. Les informations qui sont gérées sont les suivantes :

- le format de fichier (XML ou HTML)
- la localisation du fichier
- l'utilisateur
- le fait de remplacer un document existant ou non
- le fait d'indexer le document ou non
- le statut du document
- les documents attachés

Ces éléments d'informations sont fournis de la façon suivante :

- format
  - attribut **`format`** pour indiquer le format
  - attribut **`formatParam`** pour indiquer le paramètre de la requête qui identifie le format
- fichier XML
  - attribut **`filename`** (chemin absolu, peu utile)
  - attribut **`fileParam`** indiquant le paramètre de la requête qui contient le fichier
  - attribut **`formParam`** indiquant le nom du paramètre de la requête HTTP qui contient le document XML complet et correctement encodé
- utilisateur propriétaire : fourni par l'environnement SDX standard
- remplacement :
  - attribut **`replace`**, toute valeur différente de **`0`** ou **`false`** ou **`faux`** ou **`no`** ou **`non`** indique le booléen **`vrai`**, sinon **`faux`**
  - attribut **`replaceParam`** indiquant le paramètre de la requête qui donne cette indication, selon les mêmes règles
- indexation :
  - attribut **`index`**, toute valeur différente de **`0`** ou **`false`** ou **`faux`** ou **`no`** ou **`non`** indique le booléen **`vrai`**, sinon **`faux`**
  - attribut **`indexParam`** indiquant le paramètre de la requête qui donne cette indication, selon les mêmes règles
- statut :
  - attribut **`status`**, on prend sa valeur telle quelle
  - attribut **`statusParam`**, on prend la valeur du paramètre ayant le nom indiqué par cet attribut
- documents attachés :
  - attribut **`attParam`** indiquant le paramètre (répétable) de la requête qui contient les diffé-

rents fichiers attachés. Dans ce cas, les noms de fichiers envoyés doivent correspondre à l'identifiant de fichiers attachés retournés par l'indexation du document XML.

- sous-éléments **sdx:attachedDocument**, dont l'attribut id identifie l'identifiant du document attaché, l'attribut **fileParam** indique le paramètre de la requête qui contient le document attaché

A la fin, la variable **sdx_uploadDocument_docId** sera assignée à la valeur de l'identifiant du document chargé.

sdx:uploadDocuments

Cet élément permet de démarrrer l'ajout d'une collection de documents XML ou HTML et de leurs documents attachés depuis un fichier ZIP ou un dossier sur le serveur. Les informations qui sont gérées sont les suivantes :

- Le fichier ZIP ou le dossier où trouver les documents
- Le statut des documents
- Le propriétaire des documents

Ces éléments d'informations sont fournis de la façon suivante :

- localisation des fichiers
  - attribut **dir** indiquant le dossier où se trouvent les documents
  - attribut **dirParam** indiquant le paramètre de la requête qui détermine le dossier où se trouvent les documents
  - attribut **zip** indiquant le fichier ZIP où se trouvent les documents
  - attribut **zipParam** indiquant le paramètre de la requête qui détermine le fichier ZIP où se trouvent les documents
  - attribut **zipUploadParam** indiquant le paramètre de la requête qui contient le fichier ZIP où se trouvent les documents
- statut des documents
  - attribut **status** indiquant le statut à donner aux documents
  - attribut **statusParam** indiquant le paramètre de la requête qui détermine le statut à donner aux documents
- utilisateur propriétaire : fourni par l'environnement SDX standard
- propriétaire des documents
  - attribut **owner** indiquant le propriétaire des documents
  - attribut **ownerParam** indiquant le paramètre de la requête qui détermine le propriétaire des documents

A la fin, la variable **sdx_ud_errors** sera assignée à la liste des erreurs obtenues.

sdx:includeDocument

Cet élément permet d'inclure un document XML géré par SDX dans le document XML dynamique en cours de construction. C'est habituellement la façon d'afficher un document.

Ces attributs peuvent être utilisés :

id	L'identifiant du document, dans le cas où celui-ci est déjà connu.
idParam	Le nom du paramètre dans la requête qui identifie le document à retourner.

	S'il n'est pas présent, ce sera **id**.
qidParam	Le nom du paramètre dans la requête qui identifie la recherche qui a permis de trouver ce document. S'il n'est pas spécifié, ce sera **q**.
noParam	Le nom du paramètre dans la requête qui identifie le numéro de ce document dans les résultats de recherche. Par défaut, ce sera **n**.

sdx:deleteDocument

Cet élément permet de supprimer un document dans la base de documents en cours. L'attribut **idParam** indique le paramètre de la requête qui contient l'identifiant du document à supprimer.

sdx:buildXMLFromForm

Cet élément permet d'interfacer une classe externe à SDX avec le système d'alimentation d'une base de documents, dans le but de permettre la création de documents XML par un formulaire HTML.

L'attribut **classname** contient le nom d'une classe Java, disponible dans le **CLASSPATH** et respectant l'interface **fr.gouv.culture.sdx.XMLBuilderFromForm**, qui sera chargée de traiter le formulaire et de construire le document XML.

Par la suite, il se chargera d'alimenter la base de documents avec ce nouveau document.

sdx:selectSSH

Cet élément permet de facilement associer une feuille de style XSLT pour la présentation du document XML dynamique en cours de construction. De cette façon il est possible de choisir la feuille de style dynamiquement, par exemple par un paramètre de la requête HTTP.

Il peut recevoir ces attributs :

sshParam	Le nom du paramètre dans la requête qui identifie la feuille de style à utiliser. S'il n'est pas présent, ce sera le paramètre **f**.
defaultSSH	La feuille de style à utiliser si aucune n'est précisée dans la requête. S'il n'est pas spécifié, ce sera **xsl/normal.xsl**.

La référence à la feuille de style doit être relative au répertoire de la base de documents.

sdx:index

Ce tag permet d'obtenir une liste de termes issus d'un index. Il est relativement générique, les fonctions suivantes sont permises :

- Préciser le nom du champ dont on veut obtenir les valeurs (obligatoire)
- Préciser une valeur avec masque pour limiter les termes
- Préciser le premier et le dernier terme à afficher dans la liste
- Préciser des champs et valeurs dans le but de construire des sous-listes

Les attributs possibles sont

name	Le nom du champ dont on veut consulter l'index
nameParam	Le paramètre pour le nom du champ (défaut: **f**) .
value	La valeur servant de filtre
valueParam	Le paramètre contenant la valeur servant de filtre (défaut: **v**)
id	L'identificateur de l'index, si déjà créé

idParam	Le paramètre contenant l'identificateur de l'index (défaut: **id**)
page	Le numéro de la page à afficher
pageParam	Le paramètre pour le numéro de la page à afficher (defaut: **p**)
hitsPerPage	Le nombre de résultats à afficher par pages
hppParam	Le paramètre pour le nombre de résultats à afficher par pages (défaut: **hpp**)

Les paramètres de nom de champ et de valeur peuvent être répétés, dans ce cas la requête sera identifiée comme une requête pour une liste de termes filtrée, par exemple pour sortir des listes hiérarchiques.

Ce tag peut aussi contenir des sous-éléments **sdx:filter** qui permettent de préciser des filtres, dans le but de créer des listes hiérarchisées par exemple. Cela revient au même que de répéter les paramètres **f** et **v**. Ces filtres sont précisés de la sorte:

```
<sdx:filter name="departement" value="*Loire*"/>
```

Il y a deux façons d'utiliser ces paramètres, soit on utilise les paramètres de la requête, soit on les spécifie dans la page XSP. Le test est effectué sur l'attribut 'name', donc :

- Si l'attribut **name** est spécifié, alors on utilise l'attribut **value** s'il existe, et les sous-éléments **sdx:filter** s'il y en a
- Si l'attribut **name** n'est pas spécifié, alors on utilise les attributs **nameParam** et **valueParam** ou leurs valeurs par défaut

Les attributs **id**, **idParam**, **page**, **pageParam**, **hitsPerPage** et **hppParam** peuvent être utilisés indifféremment dans l'une ou l'autre situation.

Les éléments reliés aux requêtes de recherche

Plusieurs éléments de l'API XSP sont reliés à l'exécution de requêtes de recherche et à la présentation de leurs résultats. Ces éléments partagent une structure semblable que nous allons d'abord expliquer, avant de les présenter un à un.

Les paramètres communs

La structure commune à tous les éléments de requête consiste en cette liste d'attributs :

pageNo	Le numéro de la page à afficher
pParam	Le paramètre de la requête pour le numéro de la page à afficher
hitsPerPage	e nombre de résultats à afficher sur chaque page de résultats
hppParam	Le paramètre pour le nombre de résultats à afficher sur chaque page de résultats
qid	L'identifiant d'une requête déjà effectuée, pour présenter d'autres résultats
qidParam	Le paramètre qui contient l'identifiant d'une requête déjà effectuée
getDocuments	Si vrai, les documents trouvés seront inclus dans les résultats
gdParam	Le paramètre qui indique si les document doivent être inclus dans les résultats
baseQuery	L'identiifant de la requête de base, pour combiner des requêtes
bqParam	Le paramètre qui contient l'identifiant de la requête de base
baseOperator	L'opérateur qui relie la requête avec la requête de base
boParam	Le paramètre qui contient l'opérateur qui relie la requête avec la requête de base
sfParams	Le paramètre (répétable) qui contient la liste des champs de tri

soParams	Le paramètre (répétable) qui contient la liste des ordres de tri
fifParams	Le paramètre (répétable) qui contient la liste des champs de filtre
fivParams	Le paramètre (répétable) qui contient la liste des valeurs de filtre

sdx:reSortResults

Cet élément permet de retrier des résultats de recherche, en fonction de nouveaux critères. Il reçoit les attributs et paramètres communs, mais seuls ceux reliés au tri et à l'identification d'une requête de recherche sont pris en compte.

Aucun attribut ou paramètre supplémentaire n'est associé à cet élément.

sdx:executeLinearQuery

Cet élément permet de déclencher une requête de type linéaire. En plus des attributs et paramètres communs, les attributs suivants sont permis :

queryParam	Le paramètre (répétable) qui contient les termes recherchés
connParam	Le paramètre (répétable) qui contient les opérateurs reliant les critères de recherche
fieldsParam	Le paramètre (répétable) qui contient les champs de recherche

sdx:executeSimpleQuery

Cet élément permet de déclencher une requête de recherche simple. En plus des attributs communs, il accepte :

queryParam	Le paramètre qui contient la requête de recherche

sdx:executeFieldQuery

Cet élément permet de déclencher une requête de recherche dans un champ. En plus des attributs communs, il accepte :

fieldParam	Le paramètre qui contient le champ où s'effectue la recherche
valueParam	Le paramètre qui contient le terme recherché

sdx:executeExactFieldQuery

Cet élément permet de déclencher une requête de recherche dans un champ en connaissant le terme exact. En plus des attributs communs, il accepte :

fieldParam	Le paramètre qui contient le champ où s'effectue la recherche
valueParam	Le paramètre qui contient le terme recherché

sdx:executeDateIntervalQuery

Cet élément permet de déclencher une requête de recherche par intervalle de dates. En plus des attributs communs, il accepte :

fieldParam	Le paramètre qui contient le champ où s'effectue la recherche
beginParam	Le paramètre qui contient la date de début d'intervalle
endParam	Le paramètre qui contient la date de fin d'intervalle

sdx:executeRestrictedQuery

Cet élément permet de déclencher une requête de recherche restreinte. En plus des attributs communs, il accepte :

fieldParam	Le paramètre qui contient le champ où s'effectue la recherche
valueParam	Le paramètre qui contient le terme recherché

Présentation de l'API URL

Introduction

L'API URL est celle qui expose un certain nombre de services SDX depuis une adresse URL. Que signifie exactement cette phrase ?

Essentiellement, elle signifie qu'en atteignant une adresse URL précise, avec des paramètres, on peut obtenir de l'information dynamique depuis une installation SDX. Cette méthode est puissante car elle permet à la fois d'inclure ces informations dans des transformations XSLT mais aussi d'interroger un serveur SDX depuis une autre application sur le Web, y compris un navigateur.

On peut faire l'essai avec cette adresse et votre navigateur Web :

http://sdx.culture.fr/sdx/servlets/simplesearch?db=sdxdoc&q=api+url

Au moment d'écrire ces lignes, la page affichée est celle-ci :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<sdx:results currentPage="1" end="2" id="" nb="2"
    nbPages="1" start="1" xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx">
 <sdx:query luceneQuery="fulltext:formation" text="formation"
    type="simple" xmlns:sdx="http://www.culture.gouv.fr/ns/sdx/sdx" />
 <sdx:sort />
<sdx:result nextDocument="f_developpement" no="1" pctScore="100" score="0.21544635">
 <sdx:field indexed="true" name="sdxindexed" tokenized="false">true</sdx:field>
 <sdx:field indexed="true" name="sdxmodificationdate" tokenized="false">2001/05/20 07:22:45</
 <sdx:field indexed="true" name="sdxcreationdate" tokenized="false">2001/05/20 07:22:45</sdx
 <sdx:field indexed="true" name="sdxowner" tokenized="false">admin</sdx:field>
 <sdx:field indexed="true" name="sdxdocid" tokenized="false">f_xml</sdx:field>
 <sdx:field indexed="true" name="sdxall" tokenized="false">1</sdx:field>
 <sdx:field indexed="true" name="title" tokenized="true">XML</sdx:field>
 </sdx:result>
<sdx:result no="2" pctScore="93" previousDocument="f_xml" score="0.20108326">
 <sdx:field indexed="true" name="sdxindexed" tokenized="false">true</sdx:field>
 <sdx:field indexed="true" name="sdxmodificationdate" tokenized="false">2001/05/20 07:22:45</
 <sdx:field indexed="true" name="sdxcreationdate" tokenized="false">2001/05/20 07:22:45</sdx
 <sdx:field indexed="true" name="sdxowner" tokenized="false">admin</sdx:field>
 <sdx:field indexed="true" name="sdxdocid" tokenized="false">f_developpement</sdx:field>
 <sdx:field indexed="true" name="sdxall" tokenized="false">1</sdx:field>
 <sdx:field indexed="true" name="title" tokenized="true">Développement d'applications</sdx:fi
</sdx:result>
</sdx:results>
```

Architecture

L'architecture derrière cette API est simple. En effet, une série de servlets Java, partageant le même contexte de servlets que Cocoon et que les programmes Java de SDX, permettent de répondre aux URL et de retourner des résultats pertinents.

L'URL exacte de ces servlets dépend de l'installation SDX effectuée. Normalement, elles sont accessibles depuis l'URL **servlets/** ajoutée à l'URL de base de SDX, par exemple

http://sdx.culture.fr/sdx/servlets/ sur le serveur de la culture.

Par ailleurs, lorsque vous transformez le résultat d'une page XSP destinée à SDX, c'est-à-dire dont l'élément supérieur est **sdx:page**, l'URL de base des servlets SDX est passée dans l'attribut **url** de l'élément **sdx:servlets**, comme ceci :

```
 <sdx:environment>
   <sdx:servlets url="http://localhost/sdx/servlets"/>
  ...
</sdx:environment>
```

Vous pouvez donc y avoir accès dans une feuille de style XSLT à l'aide d'une construction comme cell-ci :

```
<xsl:variable
  name="servletsUrl"
  select="string(/sdx:document/sdx:environment/sdx:servlets/@url)"/>
```

Si, par la suite, vous désirez inclure un document SDX depuis une transformation XSLT, vous pouvez faire, par exemple :

```
<xsl:apply-templates select="document(concat($sUrl, 'get?db=sdxdoc&amp;id=tdm'))"/>
```


Documentation de l'API URL

Nous décrivons ici l'ensemble des servlets SDX qui constituent l'API URL de la plate-forme.

get

Cette servlet permet de retourner un document XML géré par SDX. Les paramètres sont :

db (obligatoire)	Le code de la base de documents.
id (obligatoire)	L'identificateur du document désiré.

Exemple :

```
http://sdx.culture.fr/sdx/servlets/get?db=sdxdoc&id=api_url
```

getatt

Cette servlet permet d'obtenir un document attaché à un document XML dans son format natif. Les paramètres sont :

db (obligatoire)	Le code de la base de documents.
id (obligatoire)	L'identificateur du document attaché recherché.
doc (obligatoire)	L'identificateur du document XML auquel est attaché le document recherché.

Exemple :

```
http://sdx.culture.fr/sdx/servlets/getatt?db=sdxdoc&doc=api_url&id=fig001
```

users

Elle permet de retourner une liste d'utilisateurs enregistrés dans SDX, pour l'ensemble de l'installation ou pour une base de documents en particulier. Les paramètres permis sont :

db	Le code de la base de documents dont on veut obtenir la liste des utilisateurs. S'il n'est pas spécifié, tous les utilisateurs seront retournés.

Exemple :

```
http://sdx.culture.fr/sdx/servlets/users?db=sdxdoc
```

Les résultats seront présentés en XML selon le format **users** [dtd/html/elements/users.html] .

tabledata

Cette servlet permet de retourner les valeurs d'une table externe associée à une base de documents. Les paramètres sont :

db (obligatoire)	Le code de la base de documents.
t (obligatoire)	Le code de la table tel que défini dans la configuration de la base de documents.
w	Le contenu d'une clause WHERE en SQL pour filtrer le contenu de la table.

Exemple :

```
http://sdx.culture.fr/sdx/servlets/tabledata?db=sdxdoc&t=tdm&w=id='api_url'
```

Les résultats seront présentés en XML selon le format **tableValues** [dtd/html/elements/tablevalues.html] .

terms

Cette servlet permet de retourner une liste de termes issus d'un index, c'est-à-dire d'un champ de recherche défini dans la base de documents ou d'un champ de recherche système. Les paramètres sont :

db (obligatoire)	Le code de la base de documents.
f (obligatoire et répétable)	Le code du champ de recherche. Il peut y avoir plusieurs occurrences de ce paramètre. Dans ce cas, la dernière occurrence représente le code du champ, et les précédentes représentent des codes de champs qui servent de filtre.
p	Le numéro de page pour le retour des listes de termes. Si absent, la première page sera retournée.
h	Le nombre de termes par pages. Si absent, on affiche 20 termes par pages.
v (répétable)	Des valeurs à appliquer comme filtre. Il doit y avoir une occurrence de moins que le paramètre **f**. La première occurrence est associée à la première occurrence du paramètre **f**. Ensemble, il définissent un critère de filtre pour la liste des termes affichés. Ainsi, seuls les termes dont au moins un document qui le possède respecte ce filtre seront retournés.

Exemple :

```
http://sdx.culture.fr/sdx/servlets/terms?db=sdxdoc&f=region&v=Aquitaine&f=dpt&p=2
```

Cette liste de termes pourrait correspondre, logiquement, aux départements de la région Aquitaine. On affiche la deuxième page de termes, donc les termes 21 à 40.

Les résultats sont présentés en XML selon le format **terms** [dtd/html/elements/terms.html] .

html2xhtml

Cette servlet permet de convertir un document HTML en format XHTML, donc XML. Cette servlet utilise le programme **HTML Tidy** [http://www.w3.org/People/Raggett/tidy/] (**version Java** [http://lempinen.net/sami/jtidy/]). Cette opération n'apporte aucune valeur ajoutée, mais permet de traiter des documents HTML à l'aide d'outils XML comme XSLT.

Pour l'instant, la seule information requise par cette servlet est la localisation du document HTML à convertir. Cette localisation peut être fournie par un nom de fichier (relatif ou absolu) ou par une URL absolue. Les paramètres vérifiés sont (dans l'ordre de priorité) :

u	L'URL abslue du fichier HTML.
r	L'URL relative du fichier HTML. L'URL de base sera soit l'URL de base du serveur SDX, soit celle fournie par le paramètre **b**.
b	L'URL de base lorsqu'on fournit une URL relative. Ce paramètre est toutefois optionnel, s'il n'est pas spécifié l'URL de base sera l'URL de l'installation de SDX.
p	Le chemin d'accès complet et absolu du fichier HTML, disponible sur le serveur.

Aucun autre paramètre ne peut être fourni. Eventuellement, des paramètres contrôlant le type de conversion vers XHTML pourront être inclus.

Les servlets de recherche

La plate-forme SDX propose plusieurs types de requêtes de recherche. Toutes utilisent le même outil de base, mais elles offrent au développeur d'applications des façons plus simples d'effectuer les requêtes.

Toutes ces méthode de recherche ont leur équivalent dans l'API URL sauf les requêtes complexes. Il existe donc six servlets qui proposent autant de méthodes de recherche, ces servlets partageant plusieurs caractéristiques communes.

Les caractéristiques communes

Toutes les servlets de cette section retournent des résultats en XML selon le format **results** [dtd/html/elements/results.html] .

De plus, elles paratagent un jeu de paramètres qui ont tous la même signification que voici.

db (obligatoire)	Le code de la base de documents.
n	L'identifiant d'une requête de recherche. S'il est spécifié et qu'il correspond à une requête stockée, la requête ne sera pas réexécutée mais réaffichée, potentiellement avec une page de résultats différentes.
p	Le numéro de page pour le retour des résultats de recherche. Si absent, la première page sera retournée.
h	Le nombre de résultats par pages. Si absent, on affiche 20 résultats par pages.
d	Si ce paramètre est présent et qu'il contient une valeur *vraie* (par exemple 1 ou true) alors les documents complets seront inclus dans les résultats, pas seulement les champs de recherche.

ff (répétable) Ce paramètre indique les champs qui constituent des filtres à appliquer à la recherche.

fv (répétable) Ce paramètre indique les valeurs de champs qui constituent des filtres à appliquer à la recherche. Il doit y en avoir le même nombre que le paramètre **f**.

fo Ce paramètre indique l'opérateur qui connecte les différents critères de filtre (paramètres **ff** et **fv**). Il peut prendre les valeurs **and** ou **or**, et s'il est absent on suppose un opérateur ET logique.

bq L'identifiant d'une requête qui servira de requête de base pour cette recherche. Cette requête de base sera combinée avec la requête en cours selon l'opérateur spécifié avec le paramètre **bo**, ou l'opérateur ET logique si ce dernier n'est pas spécifié.

bo L'opérateur reliant la requête en cours à la requête de base spécifiée par le paramètre **bq**. Il peut prendre les valeurs **and**, **or** ou **not**, s'il est absent on suppose un ET logique.

sf (répétable) Un code champ constituant une clé de tri. On peut utiliser autant de clés de tri que l'on désire. Par défaut, on utilise l'ordre ascendant pour le tri.

so (répétable) Un ordre pour la clé de tri. Ces ordres correspondent au champ identifié par le paramètre **sf** correspondant. Par défait, on suppose un ordre ascendant. Les valeurs de ce paramètre peuvent être **desc** ou **asc**.

simplesearch

Permet d'effectuer une requête simple (selon la classe **fr.gouv.culture.sdx.query.SDXSimpleQuery**). En plus des paramètres communs, on peut utiliser :

q (obligatoire) La requête de recherche.

restrictedsearch

Permet d'effectuer une requête restreinte (selon la classe **fr.gouv.culture.sdx.query.SDXRestrictedQuery**). En plus des paramètres communs, on peut utiliser :

q (obligatoire) La requête de recherche.

f Le champ où s'effectue la recherche. S'il n'est pas présent, on utilisera le champ de recherche par défaut pour la base de documents.

exactfieldsearch

Permet d'effectuer une requête exacte sur un champ (selon la classe **fr.gouv.culture.sdx.query.SDXExactFieldQuery**). En plus des paramètres communs, on peut utiliser :

q (obligatoire) La requête de recherche.

f Le champ où s'effectue la recherche. S'il n'est pas présent, on utilisera le champ de recherche par défaut pour la base de documents.

datesearch

Permet d'effectuer une requête sur un intervalle de date (selon la classe **fr.gouv.culture.sdx.query.SDXDateIntervalQuery**). En plus des paramètres communs, on peut utiliser :

f	Le champ où s'effectue la recherche. S'il n'est pas spécifié, on utilisera le champ par défaut.
bd	La date de début de l'intervalle. S'il n'est pas spécifié, la requête n'est pas bornée en début d'intervalle.
ed	La date de fin d'intervalle. S'il nest pas spécifié, la requête n'est pas bornée en fin d'intervalle.

fieldsearch

Permet d'effectuer une requête dans un champ (selon la classe **`fr.gouv.culture.sdx.query.SDXFieldQuery`**). En plus des paramètres communs, on peut utiliser :

q (obligatoire)	La requête de recherche.
f	Le champ où s'effectue la recherche. S'il n'est pas présent, on utilisera le champ de recherche par défaut pour la base de documents.

linearsearch

Permet d'effectuer une requête linéaire (selon la classe **`fr.gouv.culture.sdx.query.SDXLinearQuery`**). En plus des paramètres communs, on peut utiliser :

q (obligatoire, répétable)	Les expressions recherchées..
f (répétable)	Les champs s'effectuent les recherches. S'il n'est pas spécifié, on utilise le champ par défaut.
o	Les opérateurs qui relient les critères de recherche. S'il n'est pas spécifié, on suppose un ET logique.

L'API Java

Introduction

L'API Java est celle qui permet, depuis des pages XSP, des classes Java ou encore depuis des extensions XSLT, d'appeler les différents services SDX. Cette API est la plus développée, mais en même temps, puisqu'elle se situe à un plus bas niveau, son évolution est plus rapide.

Les API **XSP** et **URL** ont été développées dans le but de proposer une couche abstraite au-dessus de l'API Java. Ainsi, il est préférable de les utiliser et de n'utiliser l'API Java que si les deux premières ne sont pas suffisantes.

De plus, si vous utilisez fréquemment l'API Java, de même que si vous devez définir un grand nombre de classes pour votre application, cela signifie peut-être que vos développements particuliers devraient être ajoutés à SDX.

Documentation

La documentation disponible est la documentation du code source, que l'on peut consulter sous la forme **Javadocs** [javadocs/index.html] .

Les structures de données publiques

La plate-forme SDX envoie des informations aux composantes qui en font la demande. Par exemple, lorsqu'on effectue une recherche, les résultats sont envoyés par SDX à la composante qui en a fait une demande, par exemple une page XSP ou une transformation XSLT.

SDX envoie toujours ces informations dans un format XML, sous la forme d'un flux XML, pour les servlets SDX, ou un arbre DOM pour les pages XSP (dans la version 2 de SDX, les arbres DOM seront remplacés par des flux SAX2).

Ces informations respectent des structures de données qui sont exprimées par une DTD, parfois partielles car des informations peuvent être déterminées au moment où la demande est effectuée. L'objectif de cette section de la documentation est de présenter ces structures de données, cette DTD.

Toutes les informations XML fournies par SDX sont encodées à l'aide du *namespace* **sdx**, qui est associé à l'URI **http://www.culture.gouv.fr/ns/sdx/sdx**. Vous pouvez donc facilement les ignorer, par exemple dans une feuille de style XSLT à l'aide de l'instruction suivante :

```
<xsl:template match="sdx:*"/>
```

La documentation associée à cette DTD est **disponible en format HTML** [dtd/html/index.html] . Vous pouivez consulter directement ces parties de la structure de données :

- **L'ensemble des informations SDX :** élément **sdx:document [dtd/html/elements/document.html]** .
- **Informations sur la base de documents ou sur l'utilsiateur :** éléments **sdx:dbList [dtd/html/elements/dblist.html]** ou **sdx:users [dtd/html/elements/users.html]** .
- **Résultats de recherche :** élément **sdx:terms [dtd/html/elements/terms.html]** .
- **Termes d'un index :** élément **sdx:terms [dtd/html/elements/terms.html]** .
- **Requête de recherche :** élément **sdx:query [dtd/html/elements/query.html]** .
- **Valeurs de tables externes :** élément **tableValues [dtd/html/elements/tablevalues.html]** .
- **Erreurs d'exécution :** élément **sdx:exception [dtd/html/elements/exception.html]** .

Le langage de requête

SDX comprend un puissant moteur de recherche qui permet de tenir compte de la structure des documents XML. Ce document a pour but d'expliquer le langage de requête que l'on peut utiliser avec SDX pour exprimer des recherches.

De façon générale, cet outil **effectue** une recherche plein texte avec comme opérateur implicite un "ou" logique. Toutefois, les résultats pouvant être classés en ordre de pertinence probable, les documents qui contiennent plusieurs termes de recherche sont habituellement affichés en premier.

Recherche de mots

Les requêtes les plus simples sont celles qui contiennent un ou plusieurs mots. Ces mots doivent être séparés par des espaces ou des caractères de ponctuation. Par exemple, la recherche :

```
céramique grès
```

retourne tous les documents qui contiennent le mot *céramique* ou le mot *grès*. Puisque l'outil de recherche effectue un tri de pertinence, les documents qui contiennent les deux mots seront retournés en premier.

Recherche de phrases

Il est également possible de rechercher des phrases, c'est-a-dire des séquences de mots adjacents. Ainsi, la recherche :

```
"opération préventive de fouille d'évaluation"
```

retournera tous les documents qui contiennent cette chaîne de caractères. Les guillemets doubles sont utilisés pour délimiter les phrases. Bien entendu, on peut combiner des mots et des phrases pour effectuer des recherches un peu plus complexes, comme par exemple :

```
"opération préventive" fouille
```

Cette dernière requête retournera tous les documents qui contiennent ou bien l'expression *opération préventive* ou le mot *fouille*.

Recherche booléenne

Les recherches booléennes sont possibles en utilisant les préfixes "+" et "-" qui rendent respectivement un mot obligatoire ou nécessairement absent. Ainsi la requête :

```
+fouille +préventive
```

retournera les documents qui contiennent ces deux mots, alors que la requête :

```
+fouille -préventive
```

retournera les documents qui contiennent le mot *fouille* mais pas le mot *préventive*. A noter qu'un espace doit précéder le "+" ou le "-" pour qu'il soit interprété comme un opérateur booléen.

On peut utiliser des parenthèses pour grouper différents critères de recherche obligatoires ou rejetés. Par exemple, la requête :

```
+(fouille préventive céramique) -(grès silex)
```

retournera tous les documents qui contiennent obligatoirement les mots *fouille*, *préventive* et *céramique*, mais pas les mots *grès* et *silex*. Ces opérateurs et parenthèses peuvent également s'appliquer àdes phrases comme dans l'exemple suivant :

```
+"fouille préventive" -"fouille exploratoire"
```

Troncature (masques)

Tous les mots (mais pas les phrases) peuvent contenir les caractères de troncature suivants :

*****

0 ou plusieurs caractères. Par exemple, la requête :

```
informati*
```

retournera les documents qui contiennent les mots *information*, *informations*, *informatique*, *informatisation*, etc.

?

0 ou un caractère. Par exemple, la requête :

```
pelle?
```

retournera les documents qui contiennent les mots *pelle* ou *pelles*, mais pas *pelletée*.

A noter que les caractères de troncature ne sont pas permis dans une phrase. Aucune erreur ne sera générée, mais les résultats ne seront pas corrects.

Champs

Dans tous les exemples précédents, aucun champ de recherche n'était spécifié dans la requête. Les bases de documents SDX ont toutes un champ de recherche par défaut, et c'est dans ce champ que s'effectuaient les recherches.

Toutefois, il est possible de préciser un champ de recherche dans la requête même. L'exemple suivant montre comment s'y prendre :

```
titre:céramique
```

Le nom du champ doit être suivi par ":" et cette restriction s'applique au mot ou a la phrase qui suit seulement, ou encore àla parenthèse. Pour mieux expliquer ce principe, voici quelques exemples :

```
motcle:céramique grès
```

retournera tous les documents qui contiennent le terme *céramique* dans le champ *motcle*, ou le terme *grès* dans le champ par défaut. Par contre, la requête :

```
motcle:(céramique grès)
```

retournera tous les documents qui contiennent le terme *céramique* ou le terme *grès* dans le champ *motcle*. De la même façon, la requête :

```
+description:"fouille préventive" +titre:mammouth*
```

retournera tous les documents qui contiennent la phrase *fouille préventive* dans le champ *motcle* et les mots comme *mammouths* ou *mammouth* dans le champ *titre*.

La liste des champs disponibles dans une base de documents SDX n'est pas prédéfinie. C'est au concepteur de la base de définir les champs de recherche et leur signification, et de rendre l'information disponible. Pour en savoir plus, consultez l'aide de la base de documents que vous utilisez.

Majuscules, accents, mots vides

Nous décrivons ici le comportement par défaut de SDX. Il est possible que le concepteur d'une base de documents modifie ce fonctionnement. Il est donc préférable de véirifer d'abord la documentation relative à la base de documents qui vous intéresse.

Lors de l'indexation et de la recherche, toutes les lettres sont transformées en minuscules. Par conséquent, on peut exprimer indifféremment une requête en utilisant des lettres majuscules ou minuscules.

Par contre, les lettres accentuées sont conservées et les requêtes de recherche doivent en tenir compte. Par conséquence, les recherches*église* et *eglise* ne donneront pas les mêmes résultats.

Enfin, soulignons que les principaux mots vides de la langue française sont, par défaut, supprimés lors de l'indexation. Vous pouvez les utiliser en recherche, mais ils seront ignorés.

Formation

Introduction

SDX est une plate-forme ouverte et modulaire construite à partir de plusieurs composantes de base. Pour être en mesure de mettre en place des applications SDX, différentes compétences et connaissances sont nécessaires. Cette section de la documentation a pour objectif de donner une idée générale de ces connaissances et compétences.

Le développement d'une application Web dynamique, que ce soit avec SDX ou une autre technologie, nécessaite des compétences variées qui se retrouvent habituellement chez plusieurs personnes. Au moins deux connaissances générales doivent toutefois se retrouver chez tous les intervenants : le **Web** et **XML**. Ensuite, pour faciliter la discussion, nous présentons les compétences en trois parties :

1) La **gestion des systèmes**, c'est-à-dire l'installation des composantes et de SDX, de même que la maintenance de ces systèmes.
2) Le **développement des applications**, c'est-à-dire la mise en place des différentes fonctionnalités de l'application en utilisant les services SDX à l'aide de ses **API**.
3) Le **développement des interfaces**, c'est-à-dire la création des feuilles de style qui permettront l'affichage de l'information et des pages dynamiques.

Formation

Web

SDX permet de créer des *applications Web*. Il est donc absolument nécessaire qu'une bonne compréhension d'Internet et du Web soit présente chez tous les intervenants.

En particulier, on doit être en mesure de comprendre les avantages et les inconvénients d'une telle architecture, de même que les limites. Il y a des types de fonctionnalités qui ne sont pas efficaces sur le Web, tout comme il y a des styles d'interfaces qui ne sont pas appropriés.

Au-delà de ces quelques considérations, les connaissances spécifiques reliées au Web seront discutées dans les parties dédiées au **gestionnaire de système**, au **développeur d'applications**, ainsi qu'au **créateur d'interfaces**.

Formation

XML

La norme XML joue un rôle très important dans l'architecture SDX. Peu importe le type d'intervention que l'on peut être amené à faire, on doit au moins comprendre les fondements de cette norme et de ses objectifs, et très souvent sa syntaxe.

Puisque cet aspect concerne toutes les personnes appelées à travailler avec SDX, nous allons le présenter ici.

Connaissances générales

Du décideur au gestionnaire système, tous les intervenants SDX devraient au moins savoir ce qu'est XML et ce qu'on peut faire avec, et surtout ce qu'on ne peut pas faire avec. En effet, SDX est une plate-forme permettant de diffuser des documents XML, et son infrastructure technologique est très fortement basée sur XML.

Voici quelques éléments que l'on doit être en mesure de comprendre :

- Les systèmes d'information basés sur XML.
- Les composantes d'une chaîne de traitement basée sur XML.
- La problématique associée à la diffusion de documents XML.

Production de documents

Même si la production des documents XML diffusés par SDX est totalement extérieure à cette plate-forme, il peut être important de comprendre cet aspect car il faut à quelque part alimenter les bases de documents développées.

La production des documents par la saisie implique des outils spécialisés et une certaine formation. Par ailleurs, la production de documents par conversion implique certaines limites et demande des connaissances spécifiques à cette tâche.

Syntaxe

La syntaxe XML est importante en particulier pour les gestionnaires de système ainsi que pour les développeurs d'applications. Les fichiers de configuration de SDX sont en XML et il peut être important de bien les comprendre. De plus, les fichiers de configuration de Tomcat, le moteur de servlets privilégié, sont également en XML. Des éditeurs XML peuvent être utilisés pour cacher cette syntaxe, mais le fait de la connaître peut dans certains cas être fort utile.

D'un autre côté, la technologie XSP utilisée pour la création des pages dynamiques est aussi basée sur XML, et il n'existe pas d'outils de développements XSP. De plus, comme ces pages contiennent également du code Java, ce n'est pas certain qu'un éditeur XML soit le meilleur outil pour développer des pages XSP. C'est pourquoi il peut être nécessaire de bien comprendre la syntaxe XML.

Enfin, mentionnons que les outils facilitant la création des feuilles de style XSLT ne sont pas très communs ni très développés. Il est souvent plus facile de créer les feuilles de style directement en XML plutôt que de passer par un éditeur. Encore une fois, la syntaxe XML doit être connue.

Formation

Gestion de système

La gestion d'un système SDX regroupe les tâches d'installation, de configuration et de maintenance de la plate-forme SDX. A ce sujet, il est important de reconnaître qu'une grande partie du travail est complètement indépendant de SDX lui-même, mais concerne plus les composantes sous-jacentes.

Concrètement, un gestionnaire de système SDX doit être en mesure de :

- Mettre en place et maintenir une architecture réseau permettant la diffusion d'informations sur le Web, y compris les aspects de sécurité.
- Installer, configurer et maintenir un système d'exploitation (UNIX, Linux, Windows NT) stable et performant.
- Installer, configurer et maintenir un serveur Web efficace, en particulier le serveur Web Apache.
- Connaître les implications de la mise en place de sites Web dynamiques, peu importe les technologies utilisées.
- Installer correctement une machine virtuelle Java, et savoir ce que ça implique en terme de res-

sources systèmes et d'architecture informatique.

- Connaître les principes derrière les servlets Java, et savoir comment les déployer efficacement.
- Installer, configurer et maintenir le moteur de servlets Tomcat ou l'équivalent, et le configurer adéquatement pour qu'il communique avec le serveur Web.
- Installer, configurer, maintenir et sauvegarder un système de gestion de bases de données relationnelles, comme MySQL ou Oracle.

Ces connaissances sont celles d'un administrateur système, réseau et Web traditionnel, sauf peut-être l'aspect servlets et Java qui est moins connu dans certains milieux.

Formation

Développement d'applications

Le développement des applications SDX est peut-être l'aspect qui demande le plus de connaissances spécifiques. De façon générale, le développement d'applications Web demande de connaître profondément l'architecture Web ainsi que les outils de développement utilisés. SDX n'y échappe pas, avec la particularité suivante : les outils de développement sont nombreux et variés.

Nous ne nous attarderons pas ici sur l'architecture Web mais plutôt parler des outils de développement SDX.

XSP

La technologie XSP permet de construire des pages Web dynamiques, éléments clés d'une application Web. Les applications SDX simples peuvent être construites à partir des pages XSP fournies avec la distribution, avec très peu de modifications. Dans ce cas, peu de connaissances spécifiques sont nécessaires, mais en même temps les sites obtenus sont simples et uniformes dans leurs fonctionnalités.

Pour créer des applications plus évoluées ou différentes, on doit nécessairement passer par la création de pages XSP, et donc connaître cette technologie. La documentation XSP se résumé à une page, alors la formation nécessaire s'obtient par l'expérience.

Java

Les pages XSP doivent produire du code Java. Ainsi, pour une application SDX évoluée, du code Java devra probablement être créé, soit dans des pages XSP, soit dans des classes autonomes, soit dans des bibliothèques de balises XSP.

Dans tous les cas, une bonne connaissance de la syntaxe Java mais surtout des principes de développement orientés objet de même que de l'API standard de Java est nécessaire.

XSLT

La norme XSLT définit un langage de transformation de documents XML vers différents formats, le plus souvent XML ou HTML. Cette technologie est fort utile en général, mais également centrale dans l'architecture SDX.

Le développeur d'applications SDX doit connaître XSLT, et connaître cette norme de façon assez poussée, pour ne pas simplement se limiter à des transformations qui ne changent pas la structure.

Par exemple, dans l'application SDX qui diffuse sa propre documentation, c'est en XSLT que l'affichage de l'en-tête avec le chemin de navigation s'effectue. Pour ce traitement, un document externe doit être lu et traité, en fonction de l'identifcateur du document courant, etc.

Pour un développeur d'application SDX, il est primordial de connaître XSLT suffisamment pour comprendre qu'il ne s'agit pas d'un langage de feuille de style mais bien d'un langage de transformation, et que les transformations de documents XML peuvent aller loin et offrir des possibilités intéressantes.

Autres technologies

Puisque SDX fait partie d'un environnement Cocoon, tous les services Cocoon sont accessibles aux applications SDX. Des efforts de développement peuvent être sauvés si d'autres l'ont fait auparavant, et c'est souvent le cas avec le projet Cocoon et ses nombreux utilisateurs.

La norme (en construction) XSL-FO définit des objets de formatage et permet ainsi de produire des documents imprimés de haute qualité, à partir de sources diverses mais en particulier de documents XML. Cette future norme peut donc jouer un rôle central dans une application SDX, car elle peut permettre de fournir aux utilisateurs des versions imprimables des documents.

Formation

Interfaces utlisateurs

Le développement des interfaces d'applications SDX est traité à part car il implique souvent d'autres personnes, en particulier des graphistes. A priori, le développement d'une interface Web peut être faite sans connaissance des outils utilisés, et c'est le cas avec SDX. Toutefois, dans le but de rendre le travail d'équipe plus efficace, certaines connaissances de base seraient un atout.

En particulier, les créateurs d'interface doivent être sensibilisés aux avantages et aux limites des interfaces Web dynamiques, c'est-à-dire gérées par des programmes. Par exemple, ces interfaces peuvent offrir des informations contextuelles que l'on n'a pas dans des pages statiques, et il peut être intéressant de chercher à les exploiter. D'un autre côté, les interfaces dynamiques sont en général moins souples sur les particularités de mise en page, car en général on ne connaît pas d'avance le contenu affiché alors on ne peut pas tout calculer au pixel près.

Ensuite, les créateurs d'interface pourraient être sensibilisés à la notion de diffusion de documents XML, en particulier de documents XML hypermédia. Ces derniers sont peu fréquents, mais offrent des possibilités très intéressantes. Si les documents sont plutôt de nature *base de données*, alors les créateurs doivent être sensibilisés à la problématique de la mise en ligne de bases de données et d'affichage de leur données.

Par ailleurs, les créateurs d'interfaces Web dynamiques ne vont pas produire des pages HTML, mais plutôt des modèles de pages, voire des ambiances seulement. Ces modèles seron repris par les développeur d'applications, pour les intégrer par exemple dans des feuilles de style XSLT. C'est pourquoi le projet peut être beaucoup plus efficace si les créateurs d'interfaces sont en mesure de fournir des modèles propres et qui tiennent compte des exceptions, des cas extrêmes, etc.

Enfin, l'idéal est que les créateurs d'interfaces travaillent directement en XSLT, dan le but de produire non pas des modèles mais bien les transformations nécessaires. Dans ce cas, il faut au moins connaître un peu la norme XSLT, suffisamment pour être en mesure de transformer un élément XML en un élément HTML approprié. Et donc connaître HTML non pas à travers un éditeur, mais le code lui-même.

Le passé et le futur de SDX

L'histoire de SDX

La plate-forme est née au ministère de la culture et de la communication à l'automne 2000. Il y avait un besoin de plus en plus pressant de pouvoir montrer facilement des collections de dossiers électroniques, afin de faire connaître ces documents mais également de faire avancer les projets.

Les premiers développements ont été effectués par la société **AJLSM** [http://www.ajlsm.com] , avec comme objectif de réfléchir à la problématique de la diffusion de documents XML dans un environnement de logiciels libres.

Pour l'instant, sept versions versions ont été rendues publiques :

1.1

Correctifs et nouvelles fonctionnalités. Les changements apportés **sont décrits ici**.

Il s'agit également de la première sortie sur le site **SourceForge** [http://www.sourceforge.net/projects/sdx/] .

1.01

Premiers correctifs apportés à SDX 1. Les changements apportés **sont décrits ici**.

1.0

Première sortie officielle de la version 1 de SDX, qui marque une étape importante. Cette version constitue une plate-forme de référence, elle évoluera de façon corrective (versions 1.01, 1.02, etc.) si nécessaire. Il est possible mais pas certain qu'une branche 1.1 soit créée afin d'y apporter de nouvelles fonctionnalités.

1.0 beta 1

Deuxième sortie publique de la version 1 de SDX, avec cette fois toutes les fonctionnalités prévues dans la version finale, qui elle-même sortira vers la fin du mois de mai 2001. Certains aspects restent à tester, comme le support InstanDB et Interbase, de même que l'aindexation des mots avec apostrophes.

1.0 alpha 1

Première sortie publique de la version 1 de SDX. Au menu, un changement dans les structures de données publiques, beaucoup de changements internes, et le début de l'ajout des nouvelles fonctionnalités. Utiliser cette version pour toute nouvelle application. La verison 1.0 est prévue vers la fin du mois de mai 2001.

0.9

Première véritable version, avec comme principaux changements l'ajout d'un fichier manquant dans la distribution et une documentation plus complète.

0.9 beta 1

Première version publique, utilisée entre autres pour la diffusion du **guide de l'Internet culturel** [http://www.portail.culture.fr] par le ministère.

La plate-forme est appelée à évoluer, un court document présente la future **version 2** de SDX.

Le passé et le futur de SDX

Version 0.9

La version 0.9 de SDX n'est plus disponible. Vous devez installer la **version 1**.

Le passé et le futur de SDX

Version 1

La version 1 de SDX est **maintenant sortie**. Elle est bâtie sur la même architecture que la **version 0.9**, mais ajoute un grand nombre de fonctionnalités, et beaucoup de changemets internes dont plusieurs ont une incidence sur les applications développées en version 0.9. En fait, une application SDX v0.9 n'est pas compatible avec la version 1. Pour en savoir plus, veuillez contacter **Martin Sévigny** [mailto:sevigny@jlsm.com] .

Sans entrer dans les détails, la liste des nouvelles fonctionnalités de la version 1 de SDX :

Listes hiérarchiques

Les listes hiérarchiques permettent de fabriquer rapidement des interfaces de navigation hiérarchiques, par exemple une navigation géographique qui part de la France, pour aller dans un région, un département, une commune, etc. Ces listes pourront aussi être exploitées de façon individuelle se nécessaire. Aucune limite ne sera imposée quant au nombre de niveaux hiérarchiques pour ces listes.

Recherche dans des résultats

Il est possible d'effectuer des recherches dans un sous-ensemble d'une base de documents, identifée par un résultat de recherche précédent.

Cette fonctionnalité peut être utilisée à la fois pour la mettre entre les mains des utilisateurs, mais aussi pour créer virtuellement des sous-bases de documents. Dans ce cas, la requête identifiant la sous-base pourra être effectuée à l'aide d'une requête cachée.

Tri des résultats

Tous les champs de recherche peuvent être utilisés comme clé de tri. L'ordre ascendant ou descendant est supporté, de même que plusieurs clés de tri.

Recherche par dates

Les champs de type date sont correctement gérés, en recherche et en tri.

Navigation dans les résultats

La navigation dans les résultats est ce qui permet aux utilsateurs de passer au résultat de recherche suivant ou précédent sans avoir à retourner à la liste.

Statut des documents

On peut identifier le statut d'un document lors d'un chargement, tout comme on peut faire des recherches en fonction du statut.

Propriétaires de documents

Les propriétaires de documents sont actuellement stockés sous la forme d'une relation vers une entrée dans la table des utilisateurs.

Indépendance des SGBD

La version 1 ajoute le support d'Oracle (version 8 et ultérieure), Interbase (version 6 et ultérieure) et InstanTDB (version 3.25 et ultérieure) comme SGBD, en plus de MySQL. Par la même occasion, la configuration SDX devrait faciliter le passage d'une SGBD à l'autre.

Version 2

La version 2 de SDX contiendra sûrement plusieurs nouveautés. En particulier, un changement d'architecture informatique est à prévoir.

Architecture

L'architecture actuelle de SDX est fortement dépendante de l'architecture de Cocoon version 1. Cette architecture possède deux limites importantes :

1) Les informations échangées entre les différentes étapes d'un traitement sont en format XML et représentées par un *arbre DOM* (Document Object Model). Cela présente l'avantage d'être facile à manipuler, y compris de façon non séquentielle, mais présente aussi l'important inconvénient d'être gourmand en mémoire et lourd en traitement.

La version 2 de Cocoon utilisera la spécification *SAX* (Simple API for XML) pour passer les informations d'une partie à l'autre, augmentant nettement les performances et diminuant les besoins en mémoire.

2) L'association entre les documents XML et leur feuille de style XSLT se fait à l'aide d'une instruction de traitement insérée dans le document. Pour de grandes collections de documents, cette pratique est totalement inédéquate. Dans la version 2 de Cocoon, un *plan du site* (sitemap) sera utilisé pour indiquer ces associations ainsi que d'autres informations.

La première limite implique plutôt les performances, mais il sera important de prévoir la migration de SDX vers ce modèle. Dans un premier temps, la feuille logique de SDX devra être modifiée pour tenir compte du changement d'architecture. Ensuite, les pages XSP des applications SDX devront être vérifiées afin d'identifier celles qui devront être modifiées.

La deuxième limite a des impacts moins évidents sur SDX, car a priori les applications SDX n'utilisent pas le mécanisme standard de Cocoon pour l'association des feuilles de styles XSLT. Toutefois, il sera intéressant de vérifier si l'utilisation du plan du site ne pourra pas améliorer la plateforme.

Fonctionnalités

La liste des fonctionnalités de la version 2 de SDX n'est pas encore arrêtée, et toute personne intéressée à suggérer des modifications est invité à le faire en écrivant à **Martin Sévigny** [mailto:sevigny@ajlsm.com] .

Pour l'instant, voici ce qui est déjà identifié :

Multilinguisme

SDX devrait être nettement amélioré quant à ses capacités de recherche dans des documents multilingues ou des bases de documents multilingues.

Multibase

SDX devrait permettre d'effectuer aisément des recherches dans plusieurs bases de documents de façon simultanée. L'outil de recherche Lucene, à la base du moteur de recherche SDX, devrait supporter cette fonction éventuellement, et il s'agira ici de rester en phase avec ses développements et de les exploiter.