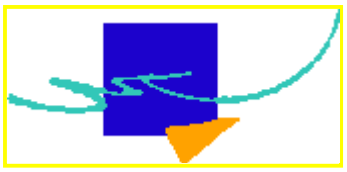
Le moteur de base de documents Xedix

Pierre Brochard

18 septembre 2001

Présentation Groupe PIN

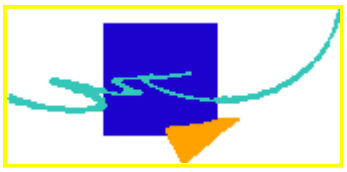
Plan



Base documentaire

Ensemble de documents électroniques et son outil de gestion
Comprend des possibilités de :

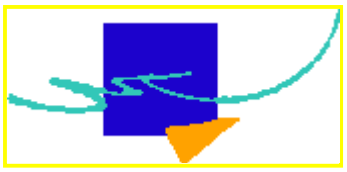
- Saisie, conversions
- Stockage, import et export de documents
- Consultation et recherche



Moteur de base documentaire Xedix

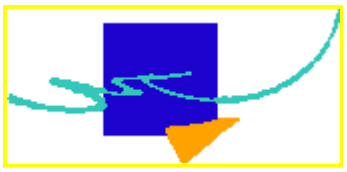
Outil CEA intégrant :

- Import de documents SGML/XML
- Export des documents au format SGML
- Export des documents au format XML
- Schéma non typé de stockage dans un SGBD
- Stockage dans un SGBDOO (O2 d'Ardent Software) ou stockage dans Clio (CEA/DAM-Ile de France)
- Consultation Web de documents avec génération dynamique d'une mise en page en HTML
- Consultation Web de documents en XML
- Couplage avec moteurs de recherche Search'97 de Verity et Spirit du CEA et TGID
- Administration des droits d'accès à la base par interface Web
- Contrôle des droits d'accès à la base par interface Web
- Gestion du droit d'en connaître des documents.



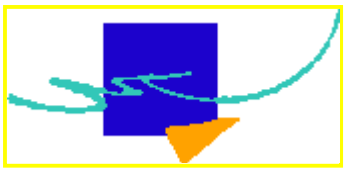
Moteur de base documentaire XediX : infos techniques

Ecrit en C++
Driver d'accès à une base de données : instancié pour O2 et Clio
Client/Serveur pour l'accès par une interface Web
Filtres d'import et d'export de documents
Tourne sur serveur SUN Microsystems sous Solaris et d'autres types de machines sous d'autres instances d'UNIX
L'IHM Web utilise les langages HTML 4, Javascript, Java (applet de zoom seulement)
L'export Web XML utilise XML 1.0 et nécessite un client capable de le mettre en page ou une mise en page au niveau su serveur Web
Les méthodes HTTP GET et POST sont utilisées



Moteur de base documentaire XediX : infos techniques (II)

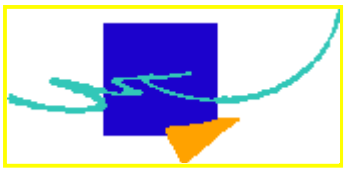
Utilise SP de James Clark
Une base de données (O2 d'O2 Technologies et racheté par Informix ou Clio du CEA/DAM-Ile de France)
Un ou deux moteurs de recherche (VDK de Verity ou Spirit du CEA et TGID)
Le logiciel client-serveur de visualisation scientifique Narcisse du CEA/DAM-Ile de France
Un serveur Web HTTP comme apache
Stockage > 100 Go possible avec de bonnes performances en consultation avec O2 et Clio



Consultation des documents

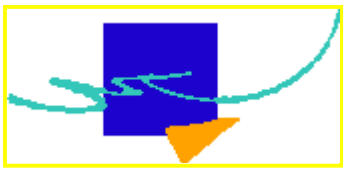
Possibilité de voir la liste des documents
Possibilité de lire les documents dans un outil universel comme un navigateur Web
Conversions, mise en page éventuellement dynamique, créée à la volée

Génération dynamique de HTML simple paramétrable en fonction de la DTD
On représente par une image ce que l'on ne sait pas convertir (équation, etc...)
Table des matières, accès à des portions de documents sans chargement complet du document en mémoire => performances
Pour aller plus loin, l'export XML est fait pour celà...



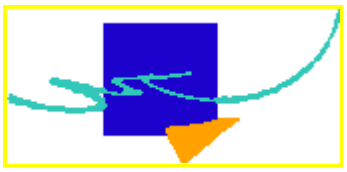
Export des documents

Recréation d'un sous-ensemble de documents de la base
- sous la forme originelle de documents SGML individualisés
- sous la forme de documents XML individualisés
- donc transférable à l'extérieur de la base de données



Gestion des liens

Simples : conventions de type HTML ou BOOK (ISO 12083)
Internes
Externes de documents dans la base
Externes de documents hors de la base (URIs)



Saisie des documents

Source des documents non électronique
- papier (cas de livres ou documents d'un fond donné)
- photos sur support argentiques
- films sur support argentiques

Source des documents électronique dans un format moins structuré que celui de la base (ex. MS Word)
Source des documents électronique dans le format accepté par la base
Source des documents électronique en provenance de bases de données structurées relationnelles

=> stratégies en amont de Xedix assurées par des outils spécifiques

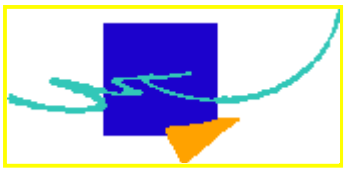


Schéma typé

Schéma dépendant d'un type de document donné (DTD).
Oblige à réorganiser sa base de données et à changer son schéma pour tout nouveau type de document.
Exemple table relationnelle Personne ("NOM","PRENOM","ADRESSE")
Stockage d'objets de type Personne (dans une base relationnelle ou objet).

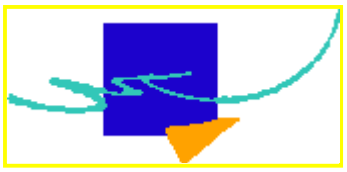


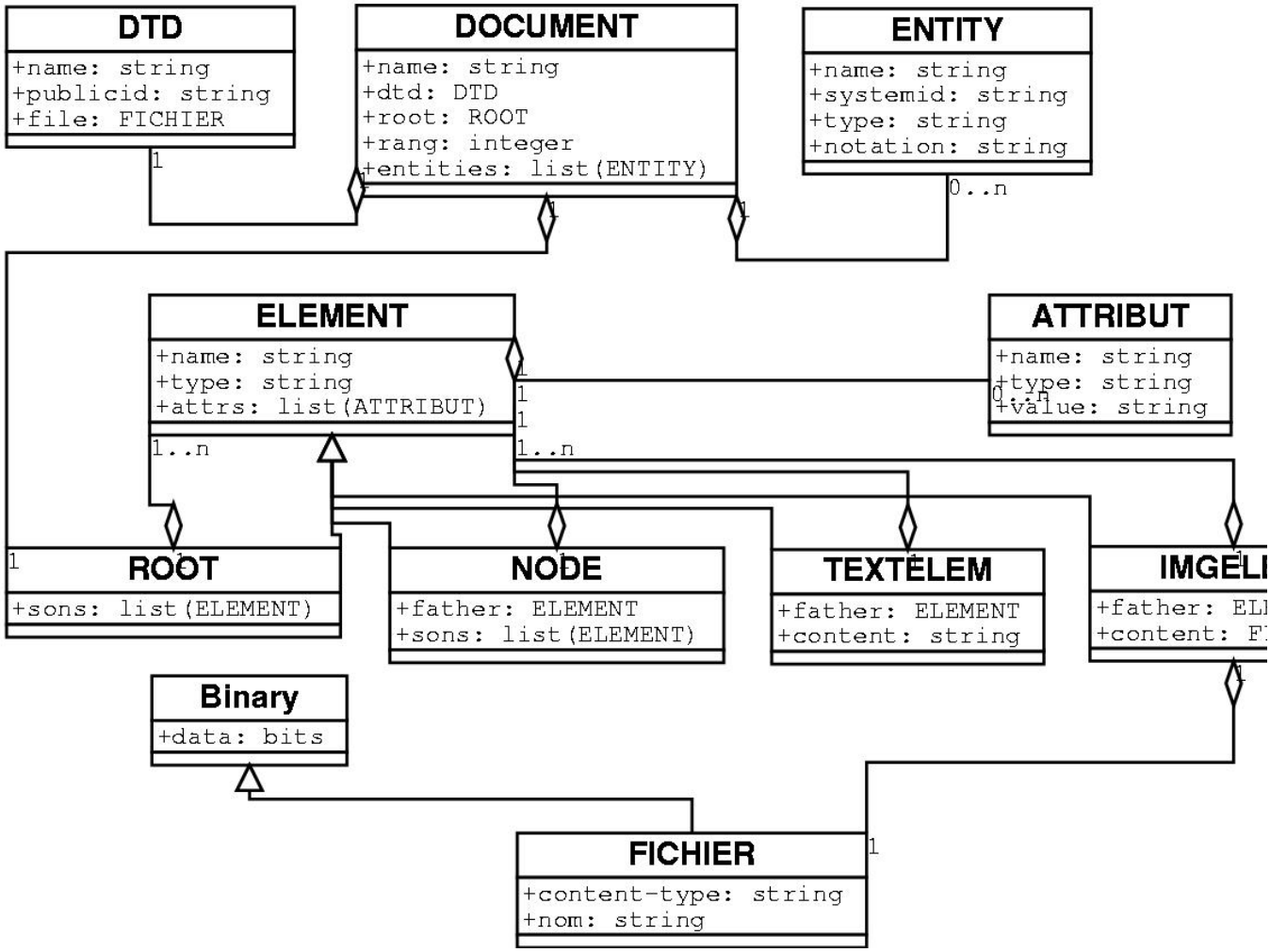
Schéma générique

Schéma **indépendant** d'un type de document donné (DTD).
Peut représenter n'importe quel arbre.
Exemple table relationnelle Personne ("NOM","PRENOM","ADRESSE")
Stockage d'objets de type Document contenant la racine d'un arbre dont les feuilles vont-être des éléments textuels de la forme:

("NOM","valeur pour le champ nom")
("PRENOM","valeur pour le champ prénom")
("ADRESSE","valeur pour le champ adresse")

Schéma générique : Xedix

Diagramme de classe UML de la description générique d'un document dans Xedix



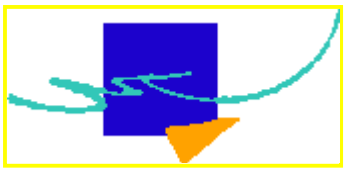
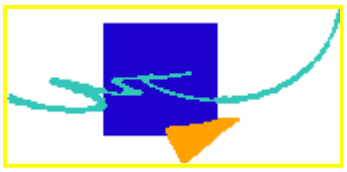


Schéma générique : Xedix (II)

Les objets persistants dans une base de donnée objets sont :

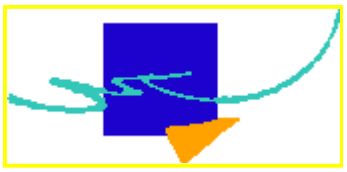
- Un tableau de DOCUMENTs
- Un tableau de IMGELEMs
- Quelques objets supplémentaires pour :
  - les moteurs de recherche,
  - les contrôles des droits utilisateurs,
  - ...



Import des documents

Pour lire et traiter un document possédant une structure, on utilise un parseur qui va le décomposer en entitées élémentaires.
Ces entitées seront stockées dans le système de gestion de base choisi.
Le parseur vérifie la conformité du document :

- SGML valide
- XML bien formé ou valide



Import des documents : parseur SGML SP

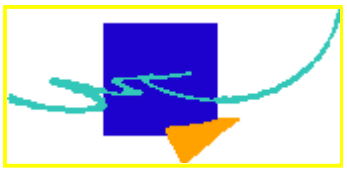
SP fournit une API générique en C++ fondée sur des évènements associés à la lecture d'un document SGML.

Les évènements sont du type :

- Début de la référence à la DTD (<!DOCTYPE...)
- Fin de la référence à la DTD
- Balise ouvrante
- Balise fermante
- Instructions de traitements
- Commentaires SGML
- Entités SGML
- ...

SP valide le document SGML et le rejete si non valide.

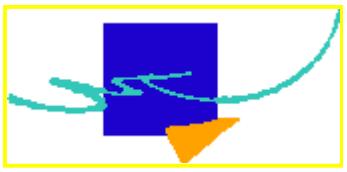
L'API se compose de la classe C++ `SGMLApplication` dont les méthodes (propriétées) doivent-être surdéfinies pour appliquer un traitement sur l'évènement en cours.



Import des documents : parseur SGML SP (exemple I)

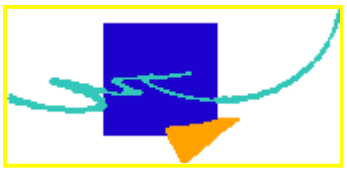
```
class MySGMLApplication : public SGMLApplication {

        void startDtd(const StartDtdEvent& event) {
                cout << "ici" << endl;
        }

        void startElement(const StartElementEvent& event) {
                cout << "ici2" << endl;
        }

}
```

Exemple plus detaillé pour passer des évènements liés aux balises au schéma générique de la base



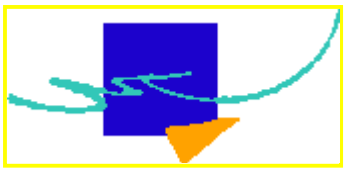
Import des documents : parseur SGML SP (III)

Il nous faut ensuite définir un objet de type `ParserEventGeneratorKit` qui sert à positionner des options.
Puis créer un objet de classe `EventGenerator` qui va générer les évènements associés à la lecture du document.
Et les donner à l'objet du type `SGMLApplication`.



Import des documents : parseur SGML SP (exemple II)

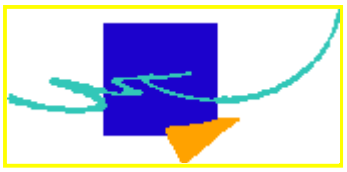
```
void initsp(int margc,char** margv,const char* config)
{
        ParserEventGeneratorKit parserKit;

        char rep[256];
        const int ret = getvar(config,"CATALOG", rep);

        parserKit.setOption(ParserEventGeneratorKit::addCatalog,rep);
        parserKit.setOption(ParserEventGeneratorKit::outputGeneralEntities);

    // lancement du parser
        EventGenerator* egp = parserKit.makeEventGenerator(1, margv);

        MySGMLApplication app(config);

        const unsigned nErrors = egp->run(app);

        delete OutlineApplication::egp;
}
```

Recherche des documents

But : pouvoir retrouver des documents sur des critères évolués :
- mots, groupe de mots
- opérateurs booléens et de proximité
- dictionnaires
- arbre de classification de termes
- langage naturel
- linguistique
- cartographie des documents de la base

Outils possibles :
- Requêtes SQL, OQL sur le SGBD utilisé
- Moteurs de recherche
- Outils d'analyse



Moteurs de recherche

Booléen, statistique (VDK de Verity)
En langage naturel avec linguistique (Spirit du CEA et TGID)