Construire une base de donnée pour bibliothèque

S'il y a bien un emploi typique pour un ordinateur, c'est celui de gérer une base de données. La première d'entre elles est bien sûr la base des fichiers de l'ordinateur. On va donc mettre à profit cette faculté pour construire une base de données d'une bibliothèque bien réelle, celle des livres que nous possédons. Nous allons aussi utiliser les composants disponibles pour Mac OS X.

1) Cahier des charges

Voilà quelques exigences que devra respecter notre base de donnée :

EX01 : l'interface utilisateur sera du type navigateur Internet.
EX02 : saisie des utilisateurs de la bibliothèque et de leurs caractéristiques
EX03 : saisie des éditeurs et de leurs caractéristiques
EX04 : saisie des collections ou séries d'un éditeur et de leurs caractéristiques
EX05 : saisie des auteurs et de leurs caractéristiques
EX06 : saisie des oeuvres et de leurs caractéristiques
EX07 : saisie de la bibliothèque de l'utilisateur, possibilité de définir la possession d'une oeuvre, possibilité d'enregistrer le prêt d'une oeuvre possédée
EX08 : recherche d'une oeuvre avec opérateurs "et", "ou", "sauf".
EX09 : édition des listes alphabétiques par auteur, par oeuvre ou par collections.
EX10 : import des oeuvres depuis un fichier et un site Internet.
EX11 : fusion avec une autre base
EX12 : remplacement (et éventuellement suppression) d'un auteur, d'un éditeur, d'une collection par respectivement un autre auteur, un autre éditeur, une autre collection tout en gardant la cohérence de la base..
EX13 : nettoyage de la bibliothèque des références orphelines
EX14 : saisie des possessions, acquisitions, prêts, emprunts et lectures.
EX15 : saisie des valeurs énumérées par utilisation.

Ces exigences pourront évoluer, cependant elles permettent de se fixer une direction de travail dans un premier temps.

2) Les tables de notre base de données

Une base de données comporte une ou plusieurs tables qui vont contenir des enregistrements de même nature qui eux même sont caractérisés par des champs contenant les données.

Nous définissons les tables suivantes:

- la table des éditeurs
- la table des collections
- la table des auteurs
- la table des groupes d'auteurs
- la table des oeuvres
- la table des propriétaires
- la table des possessions (notre bibliothèque personnelle)
- la table des énumérés

Cela parait complexe, mais nécessaire pour obtenir la souplesse voulue.

Nous allons créer notre bibliothèque avec le moteur de base de données MySQL (voir son installation sur Blady en page À Savoir). Celui-ci est multi-plateforme et sous licence GPL, ce qui convient tout à fait à notre projet. La version utilisée est la 4.1.4. Plutôt que de saisir les commandes du moteur dans le terminal nous allons utiliser l'utilitaire "CocoaMySQL" pour démarrer.

Dans un premier temps nous ne créons que les champs nécessaires au bon fonctionnement de la bibliothèque. Les autres champs sont de nature plus informative. Ils seront ajoutés ultérieurement une fois la mécanique bien éprouvée.

Septième version de la structure des tables : l'index d'un auteur dans son groupe est directement le numéro de l'oeuvre et non plus un nom interne, ajout des champs nécessaires pour l'importation depuis un fichier, ajout des champs nécessaires pour la gestion des acquisitions, des emprunts, des prêts et des lectures, ajout du champ série dans la table des oeuvres, ajout de la table des énumérés.

Le nom de chaque champ se comprend par lui-même.

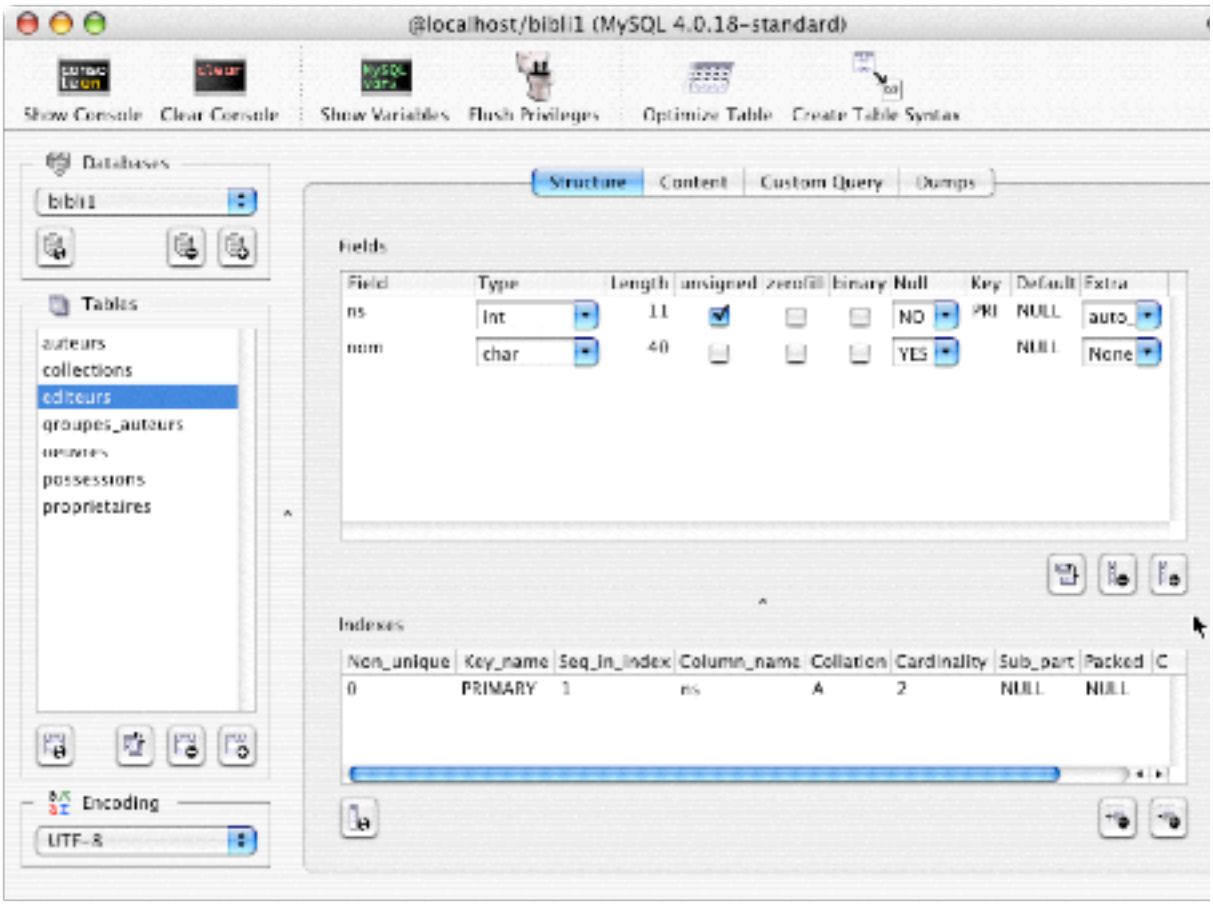
Voici les différents champs de chaque table :

Légende :

- NS est un numéro de série unique déterminé par la base de données
(type MySQL : int 11 unsigned not null auto-inc primary key)
- S est un champ de type chaîne de caractères
(type MySQL : char 40)
- E est un champ de type énuméré
- N est un champ de type nombre entier
(type MySQL : smallint 6)
- R est un champ de type nombre réel
(type MySQL : decimal 7,2)
- D est un champ de type date
(type MySQL : date)
- I est un champ de type image
- T est un champ de type texte
- B est un champ de type booléen
(type MySQL : tinyint 1)
- * champ devant être obligatoirement rempli

a) table des éditeurs

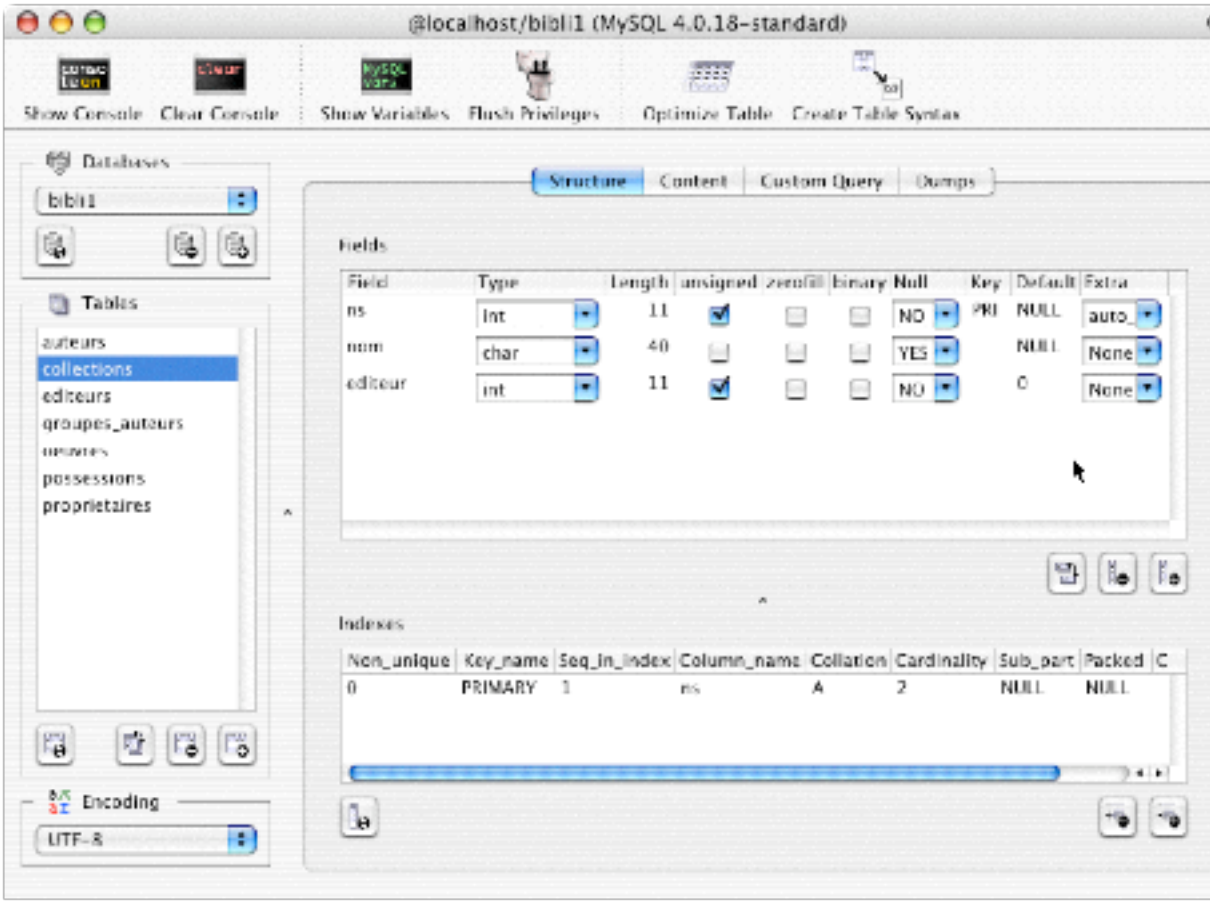
- NS
- Nom S *
- Site web S
- Groupe S
- Adresse postale S
- Téléphone S
- Télécopie S
- Adresse messagerie électronique S
- Commentaire (NS->textes.texte)




```
CREATE TABLE `editeurs` (
  `ns` int(11) unsigned NOT NULL auto_increment,
  `nom` char(40) default NULL,
  PRIMARY KEY (`ns`)
);
```

b) table des collections

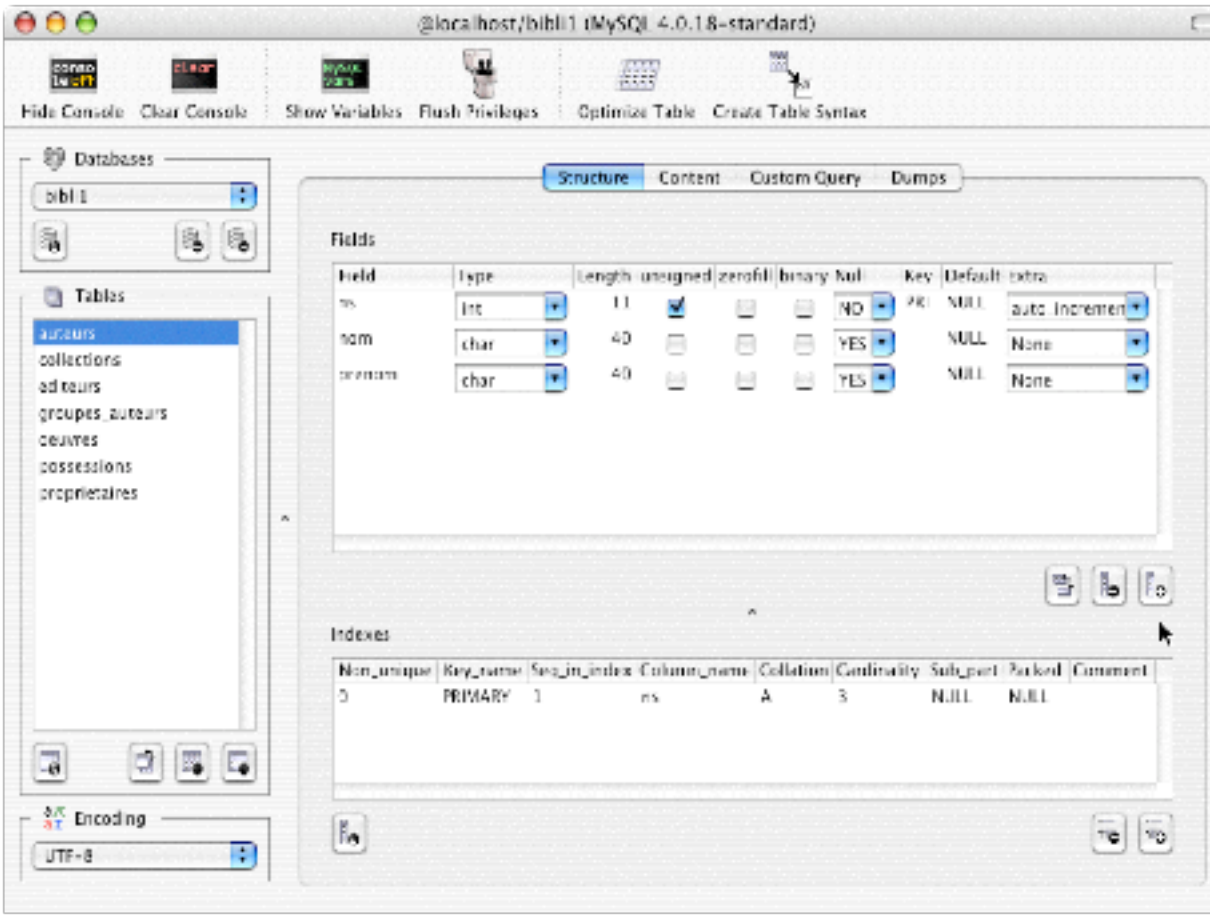
- NS
- Nom S *
- Éditeur (NS->editeurs.nom) *
- Commentaire (NS->textes.texte)




```
CREATE TABLE `collections` (
  `ns` int(11) unsigned NOT NULL auto_increment,
  `nom` char(40) default NULL,
  `editeur` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY (`ns`)
);
```

c) table des auteurs

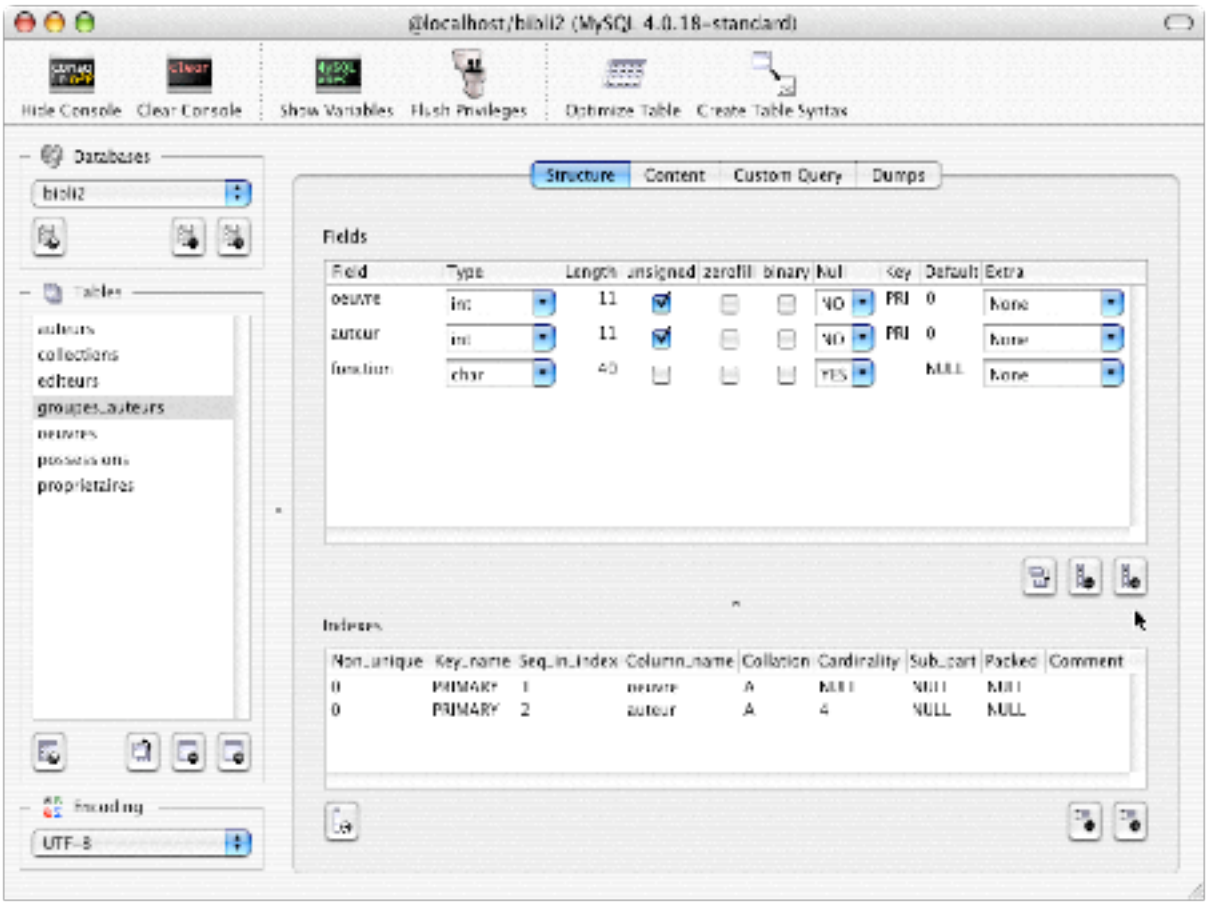
- NS
- Nom S *
- Prénom S
- Pseudo / Alias S
- Naissance D
- Décès D
- Nationalité E
- Biographie (NS->textes.texte)
- Commentaire (NS->textes.texte)
- Photo I




```
CREATE TABLE `auteurs` (
  `ns` int(11) unsigned NOT NULL auto_increment,
  `nom` char(40) default NULL,
  `prenom` char(40) default NULL,
  `biographie` int(11) unsigned NOT NULL default '0',
  `commentaire` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY (`ns`)
);
```

d) table groupe d'auteurs

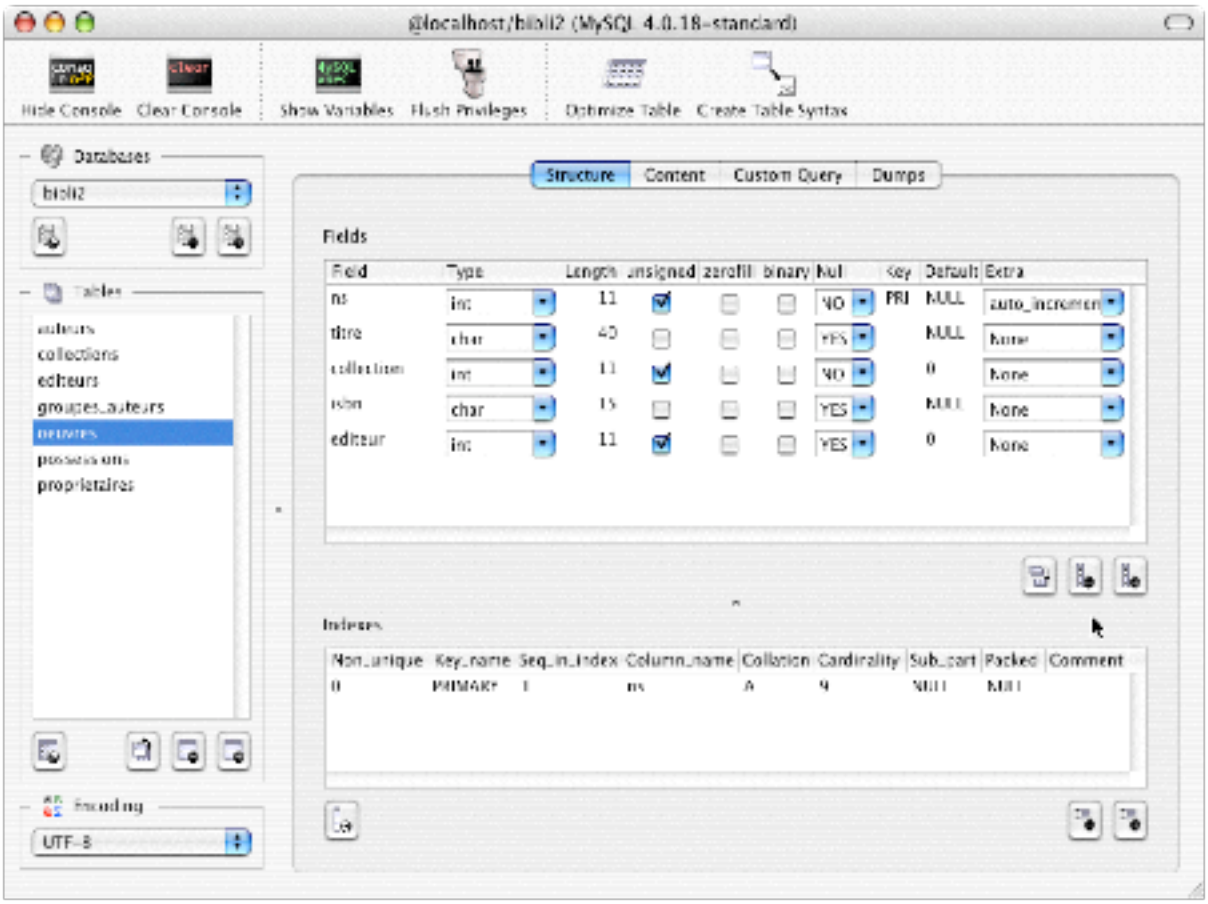
- Oeuvre (NS->oeuvres.titre) *
- Auteur (NS->auteurs.nom) *
- fonction E (texte, illustration, couleur, traduction...)




```
CREATE TABLE `groupes_auteurs` (
  `oeuvre` int(11) unsigned NOT NULL default '0',
  `auteur` int(11) unsigned NOT NULL default '0',
  `fonction` char(40) default NULL
);
```

e) table des oeuvres

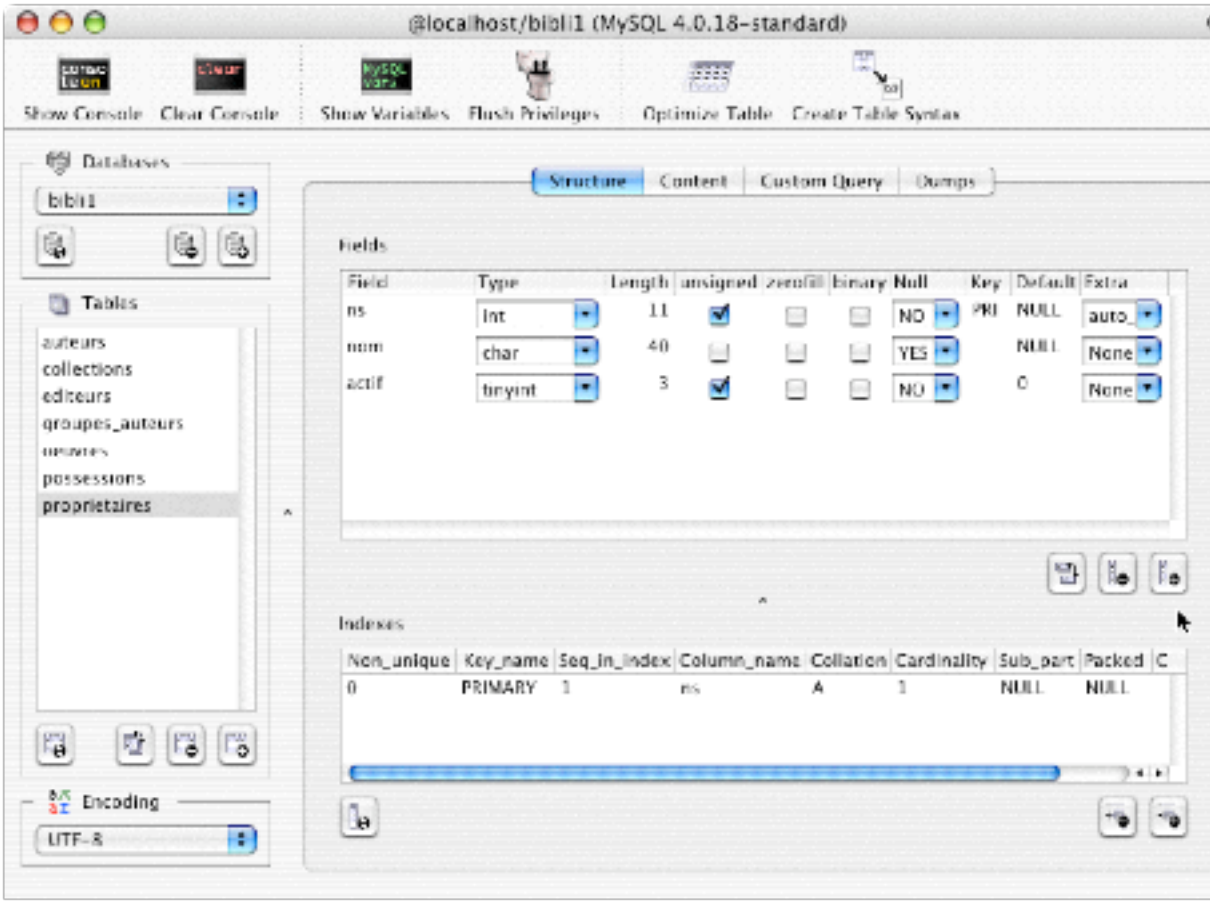
- NS
- Auteurs implicites dans la table des groupes d'auteurs (ns->nom, fonction)
- Titre S *
- Éditeur (NS->editeurs.nom)
- Collection (NS->collections.nom)
- Série (NS->collections.nom)
- Ordre dans la série N
- Parution D
- Format / Type de support E (A4, A5, coffret, poche...)
- Nombre de pages N
- Nature E (BD, roman, cours, manuel, théâtre)
- Genre E (fantastique, sentimental, amour, comédie, essais, histoire, science, philosophie, poésie, religion, science-fiction, suspens, Policier, roman, technique, théâtre, humour...)
- ISBN S
- Code barre S
- Prix éditeur R
- Résumé / Description (NS->textes.texte)
- Commentaire (NS->textes.texte)
- Couverture I
- Langue E (Français, Anglais, Espagnol, Allemand, Italien...)
- Prix actuel R




```
CREATE TABLE `oeuvres` (
  `ns` int(11) unsigned NOT NULL auto_increment,
  `titre` char(80) default NULL,
  `editeur` int(11) unsigned NOT NULL default '0',
  `collection` int(11) unsigned NOT NULL default '0',
  `serie` int(11) unsigned NOT NULL default '0',
  `ordre` smallint(6) unsigned NOT NULL default '0',
  `parution` date default NULL,
  `format` char(15) default NULL,
  `pages` smallint(6) NOT NULL default '0',
  `nature` char(40) default NULL,
  `genre` char(40) default NULL,
  `isbn` char(15) default NULL,
  `prix` decimal(7,2) NOT NULL default '0.00',
  `resume` int(11) unsigned NOT NULL default '0',
  `commentaire` int(11) unsigned NOT NULL default '0',
  PRIMARY KEY (`ns`)
);
```

f) table des propriétaires

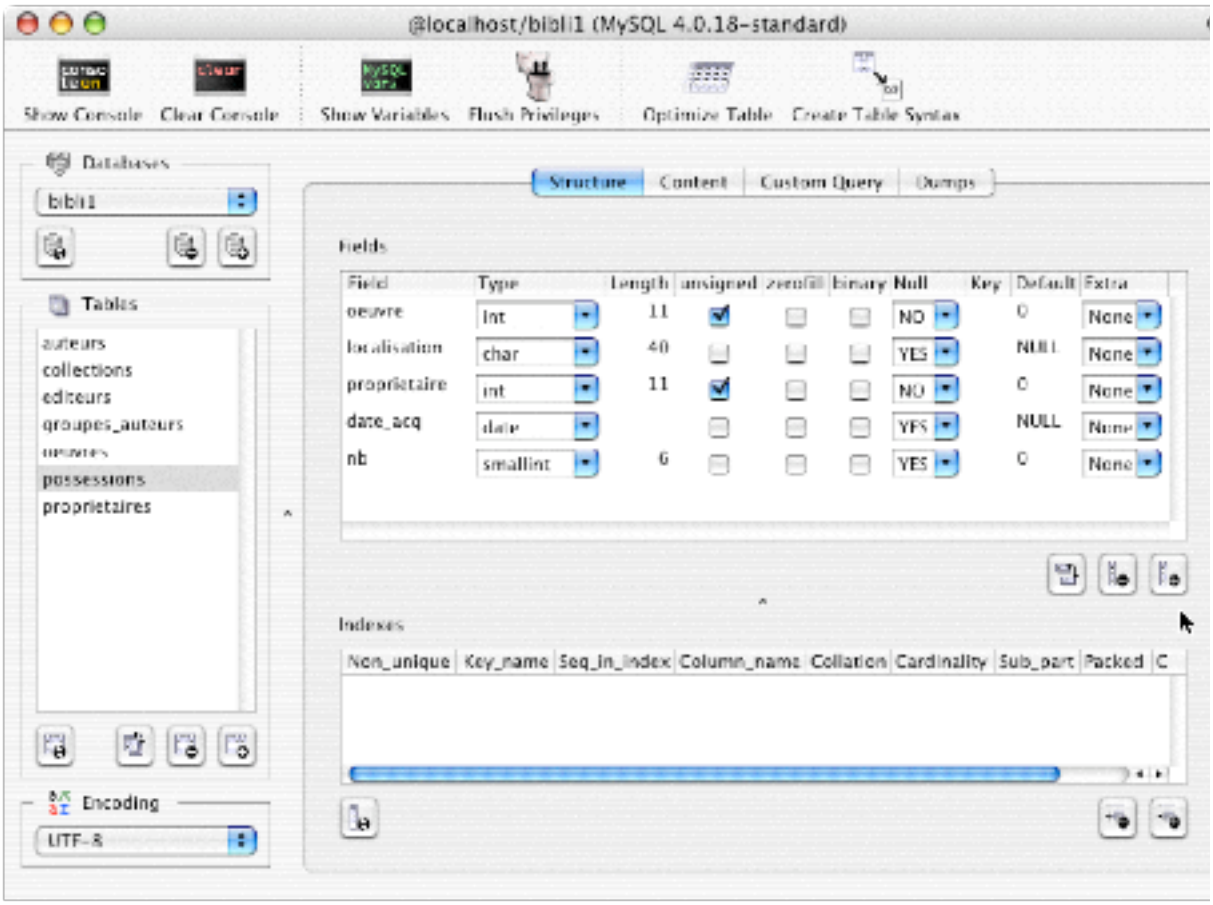
- NS
- Pseudo S *
- Mot de passe S *
- Adresse e-mail S *
- Nom S *
- Prénom S *
- Adresse S *
- Département N *
- Ville S *
- Pays S *
- Actif B




```
CREATE TABLE `proprietaires` (
  `ns` int(11) unsigned NOT NULL auto_increment,
  `nom` char(40) default NULL,
  `debug` tinyint(1) unsigned NOT NULL default '0',
  PRIMARY KEY (`ns`)
);
```

g) table des possessions

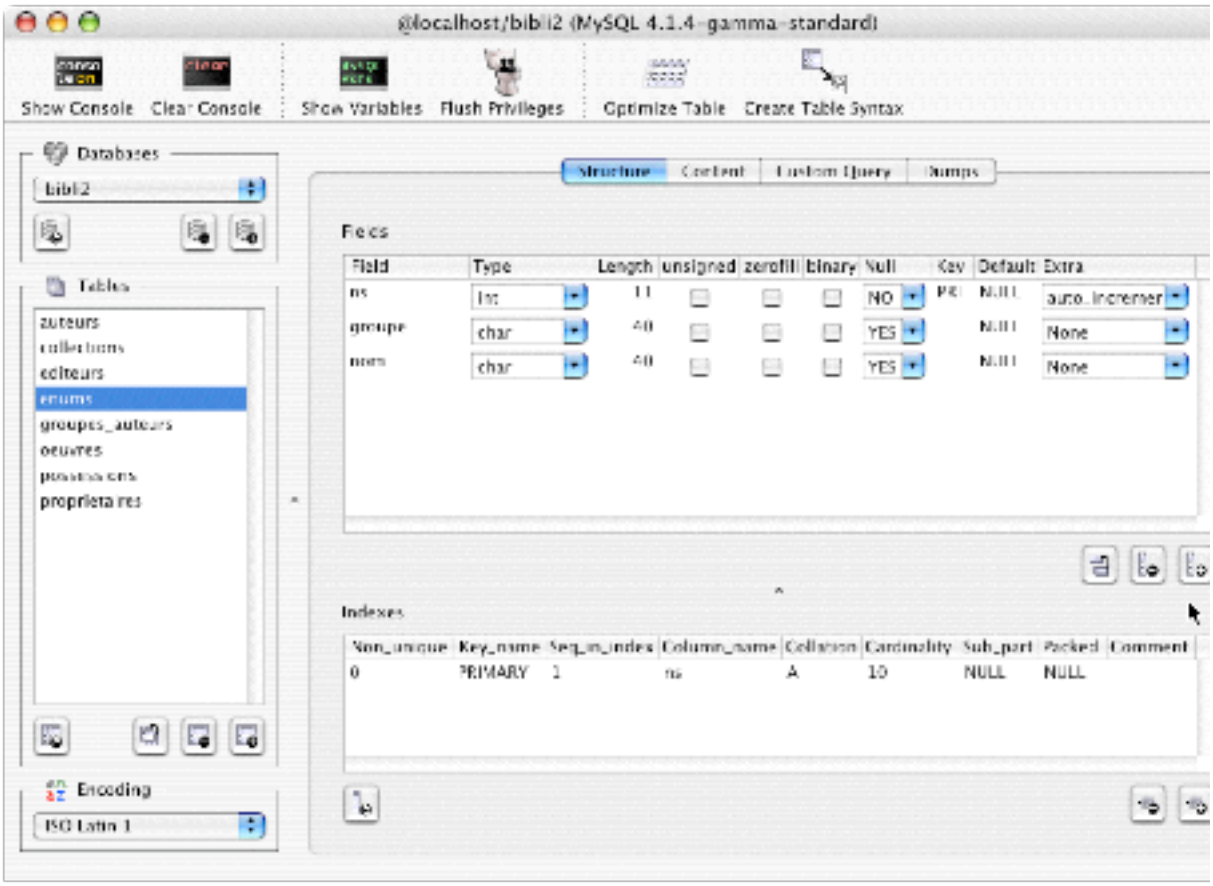
- NS
- Oeuvre (NS->titre) *
- Propriétaire (NS->pseudo) *
- Nature E (Acquisition, Emprunt, Prêt, Lecture) *
- Nom S (Emplacement, Possesseur, Emprunteur)
- Date D (d'acquisition, d'emprunt, de prêt, de lecture)
- Date de retour D
- Prix d'acquisition R
- Commentaire (NS->textes.texte)




```
CREATE TABLE `possessions` (
  `ns` int(11) NOT NULL auto_increment,
  `oeuvre` int(11) unsigned NOT NULL default '0',
  `proprietaire` int(11) unsigned NOT NULL default '0',
  `nature` tinyint(1) unsigned NOT NULL default '0',
  `nom` char(40) default NULL,
  `date` date default NULL,
  `retour` date default NULL,
  PRIMARY KEY (`ns`)
);
```

h) table des énumérés

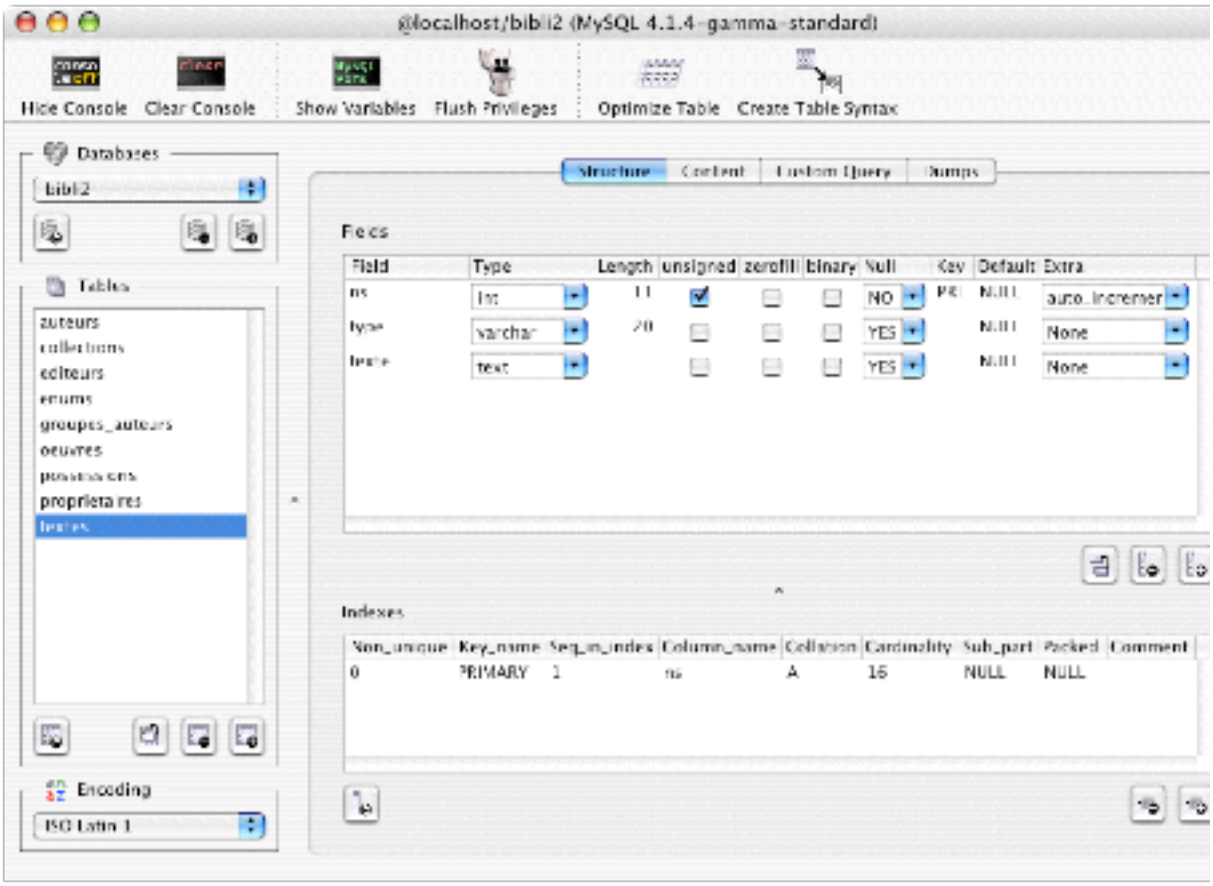
- NS
- Groupe S *
- Nom S *




```
CREATE TABLE `enums` (
  `ns` int(11) NOT NULL auto_increment,
  `groupe` char(40) default NULL,
  `nom` char(40) default NULL,
  PRIMARY KEY (`ns`)
);
```

i) table des textes

- NS
- Type S *
- Texte T *




```
CREATE TABLE `textes` (
  `ns` int(11) unsigned NOT NULL auto_increment,
  `type` varchar(20) default NULL,
  `texte` text,
  PRIMARY KEY (`ns`)
);
```

3) La création de la base de données avec MySQL

Pour créer une base de données nous devons nous connecter avec l'utilisateur "root" (au sens de MySQL) :
(La version de MySQL peut être différente de celle présentée ici)

```
$ /usr/local/mysql/bin/mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.0.18-standard
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> create database bibli1;
Query OK, 1 row affected (0.14 sec)
```

Nous lui affectons les droits pour nous :
(Remplacer "toto" par votre nom d'utilisateur MySQL.)

```
mysql> grant all privileges on bibli1.* to toto@localhost;
Query OK, 0 rows affected (0.19 sec)
mysql> exit
Bye
```

Télécharger l'archive des fichiers sources (bibli.tgz) sur le bureau.
Décompresser son contenu en l'ouvrant avec un double-clic. Un répertoire avec un indice de version apparaît.
L'ensemble des tables est maintenant créé avec le script "bibli_struct.sql" :
(L'indice du répertoire peut être différent de celui présenté ici)

```
$ /usr/local/mysql/bin/mysql bibli1 < ~/Desktop/bibli-1.3c/
bibli_struct.sql
```

4) Programmation avec Java

Nous allons pouvoir utiliser toute la puissance conjuguée de SQL et Java à travers le pilote JDBC de MySQL (voir son installation sur Blady page à savoir).

Télécharger l'archive des fichiers sources (bibli.tgz) sur le bureau. Décompresser son contenu en l'ouvrant avec un double-clic. Un répertoire avec un indice de version apparaît.

Le programme test_access.java permet de vérifier l'accès à la base de données et d'afficher le nom des tables :

```
// test_access.java
import java.sql.*;
public class test_access {
  public static void main(String argv[]) throws Exception {
    // Load the driver class
    try {
       Class.forName("com.mysql.jdbc.Driver").newInstance();
    } catch (Exception ex) {
        // handle the error
        System.out.println("Exception: " + ex.getMessage());
    }
  try {
  Connection conn = DriverManager.getConnection(
      "jdbc:mysql://localhost/bibli1?user=toto");
  Statement stmt = conn.createStatement();
  ResultSet rset = stmt.executeQuery("show tables;");
  while (rset.next()) {
    System.out.println(rset.getString(1));
  }
  rset.close();
  stmt.close();
  conn.close();
  } catch (SQLException ex) {
       // handle any errors
       System.out.println("SQLException: " + ex.getMessage());
       System.out.println("SQLState: " + ex.getSQLState());
       System.out.println("VendorError: " + ex.getErrorCode());
  }
 }
}
```

Remplacer "toto" par votre nom d'utilisateur MySQL.
Compiler et exécuter le programme :
(La version et donc le chemin d'accès du pilote JDBC peuvent être différent de ceux présentés ici)

```
$ javac test_access.java
$ java -classpath /usr/local/mysql/mysql-connector-java-3.0.11-stable/mysql-connector-java-3.0.11-stable-bin.jar:. test_access
auteurs
collections
editeurs
groupes_auteurs
oeuvres
possessions
proprietaires
```

Le programme suivant affiche la liste des oeuvres avec tous les attributs en clair en mettant en jeu les relations entre tables : aff_oeuvres.java.
Remplacer "toto" par votre nom d'utilisateur.
Compiler et exécuter le programme :
(La version et donc le chemin d'accès du pilote JDBC peuvent être différent de ceux présentés ici)

```
$ javac aff_oeuvres.java
$ java -classpath /usr/local/mysql/mysql-connector-java-3.0.11-stable/mysql-connector-java-3.0.11-stable-bin.jar:. aff_oeuvres
Le lotus bleu Les aventures de Tintin Herge Casterman
t2 c2 a2 e1
t3 c2 a1 e1
tb c3 a3 e2
```

En ayant saisi les oeuvres suivantes :

```
mysql> select ns, titre, collection, editeur from oeuvres;
+----+--------------+------------+---------+
| ns | titre        | collection | editeur |
+----+--------------+------------+---------+
|  2 | t2           |          2 |       2 |
|  3 | t3           |          2 |       2 |
|  6 | Le lotus bleu |          5 |       6 |
| 12 | tb           |          3 |       3 |
+----+--------------+------------+---------+
```

et

```
mysql> select oeuvre, auteur from groupes_auteurs;
+--------+--------+
| oeuvre | auteur |
+--------+--------+
|     12 |      3 |
|      3 |      1 |
|      2 |      2 |
|      6 |      6 |
+--------+--------+
```

5) Interface utilisateur

Notre interface utilisateur combine les technologies Java et Web en utilisant les "servlets". Nous utiliserons le serveur Tomcat (voir son installation sur Blady en page à savoir).

Création de la "servlet" bibli :

Renseigner Tomcat sur bibli en modifiant le fichier /usr/local/tomcat/conf/server.xml, ajout de quelques lignes à la fin des définitions du host.

```
$ vi /usr/local/tomcat/conf/server.xml
<!-- pages bibli -->
<Context path="/bibli" docBase="bibli" debug="0"
reloadable="true">
  <Logger className="org.apache.catalina.logger.FileLogger"
    prefix="localhost_bibli_log." suffix=".txt"
    timestamp="true"/>
  </Context>

        </Host>
```

Créer les répertoires nécessaires dans le répertoire de publication de Tomcat (webapps) :

```
$ cd /usr/local/tomcat/webapps/
$ mkdir bibli
$ mkdir bibli/WEB-INF
$ mkdir bibli/WEB-INF/classes
$ mkdir bibli/WEB-INF/lib
$ mkdir bibli/data
```

Copier et modifier le fichier de description (display-name, description, webmaster et servlet definition) de votre "servlet" :

```
$ cp ./tomcat-docs/appdev/web.xml.txt ./bibli/WEB-INF/web.xml
$ vi bibli/WEB-INF/web.xml
    <servlet>
      <servlet-name>aff_pages</servlet-name>
      <description>
        Test bibli
      </description>
      <servlet-class>aff_pages</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>aff_pages</servlet-name>
        <url-pattern>/aff_pages</url-pattern>
    </servlet-mapping>
```

Copie de la bibliothèque JDBC :
(La version et donc le chemin d'accès du pilote JDBC peuvent être différent de ceux présentés ici)

```
$ cp /usr/local/mysql/mysql-connector-java-3.0.11-stable/mysql-connector-java-3.0.11-stable-bin.jar ./bibli/WEB-INF/lib
```

Télécharger l'archive des fichiers sources (bibli.tgz) sur le bureau.
Décompresser son contenu en l'ouvrant avec un double-clic. Un répertoire avec un indice de version apparaît.

La classe **aff_pages.java** permet la gestion de la servlet avec les méthodes GET et POST.
La classe **bibli.java** permet la gestion de l'interface utilisateur en HTML.
La classe **GestionBase.java** permet la gestion de la base de données.
La classe **Parametres.java** stocke les paramètres importants de la servlet.
(Remplacer "toto" par votre nom d'utilisateur.)
La classe **ListChain.java** permet la gestion d'une liste chaînée d'arguments de la page HTML.
La classe **CSVTokenizer.java** permet la découpe d'une chaîne de caractères en plusieurs éléments entourés de guillemets et séparés par des virgules.
(L'indice du répertoire des sources peut être différent de celui présenté ici)

La classe **NCRDecoder.java** traduit les entités HTML en caractères JAVA.
Elle est disponible sur Internet, notamment sur Blady en page Créations.

```
$ cd bibli/WEB-INF/classes
$ cp ~/Desktop/bibli-1.3c/*.java .
$ export CLASSPATH="/System/Library/Frameworks/JavaVM.framework/
Versions/CurrentJDK/Classes/classes.jar:/usr/local/tomcat/
common/lib/servlet-api.jar:."
$ javac aff_pages.java
$ start_tomcat
```

Aller dans votre navigateur Internet préféré et entrer l'adresse :
http://localhost:8080/bibli/aff_pages

Appréciez le résultat.

Cette version dite 'beta' est un premier jet pour expérimenter les différentes technologies à notre disposition pour mener à bien notre projet. L'affichage pourra être amélioré en utilisant les tableaux. Les informations affichées seront complétées par la suite lorsque la mécanique sera éprouvée. Les erreurs d'intégrité de la base ne sont pas traitées. Les saisies seront sécurisées également par la suite pour éviter les erreurs d'intégrité de la base.
Les interrogations SQL et les affichages HTML gagneraient sûrement à être encapsulés dans des fonctions.

Nous disposons d'une suite de fonctions pour afficher :

. la liste des éditeurs, auteurs, oeuvres, propriétaires :

```
Liste_Editeurs();
Liste_Collections();
Liste_Auteurs();
Liste_Oeuvres();
Liste_Proprietaires();
```

. les informations pour un éditeur, auteur, oeuvre, propriétaire :

```
Info_Editeur(String arg);
Info_Collections(String arg);
Info_Auteur(String arg);
Info_Oeuvre(String arg);
Info_Proprietaire(String arg);
```

. la saisie des éditeurs, auteurs, collections, oeuvres, propriétaires, possessions et prêts :

```
Saisie_Editeurs(ListChain args);
Saisie_Auteurs(ListChain args);
Saisie_Collections(ListChain args);
Saisie_Oeuvres(ListChain args);
Saisie_Proprietaires(ListChain args);
Saisie_Possessions_Oeuvre(ListChain args);
Saisie_Enums(ListChain args);
```

. la liste des collections pour un éditeur :

```
Liste_Collections_Editeur(String arg);
```

. la liste des titres pour un éditeur :

```
Liste_Oeuvres_Editeur(String arg);
```

. la liste des titres pour un auteur :

```
Liste_Oeuvres_Auteur(String arg);
```

. la liste des titres pour un propriétaire :

```
Liste_Oeuvres_Proprietaire(String arg);
```

. la liste des titres pour une collection,

```
Liste_Oeuvres_Collection(String arg);
```

. L'importation depuis un fichier ou Internet :

```
Import_Oeuvres(ListChain args);
```

. La recherche d'une oeuvre :

```
Recherche_Oeuvres(ListChain args);
```

. L'importation depuis un fichier ou Internet :

```
Remplace_supprime(ListChain args);
```

L'affichage des listes est limité à un nombre de lignes indiqué dans la classe des paramètres (20 par défaut). Les possibilités de navigation sont obtenues par les liens :
Les premiers - Les précédents - Les suivants - Les derniers.
Pour l'écran de recherche, la navigation est obtenue en sélectionnant le bouton désiré puis en cliquant sur "Aller".

6) L'importation depuis un fichier :

Cette fonction permet d'importer une liste provenant d'un fichier au format CSV. Ce format répandu servira soit à construire manuellement une liste ou de l'obtenir depuis une autre base de données. Chaque titre se distingue sur une ligne de texte. Les champs sont entourés de guillemets et séparés par une virgule. Les champs retenus pour l'importation sont sous ce format :

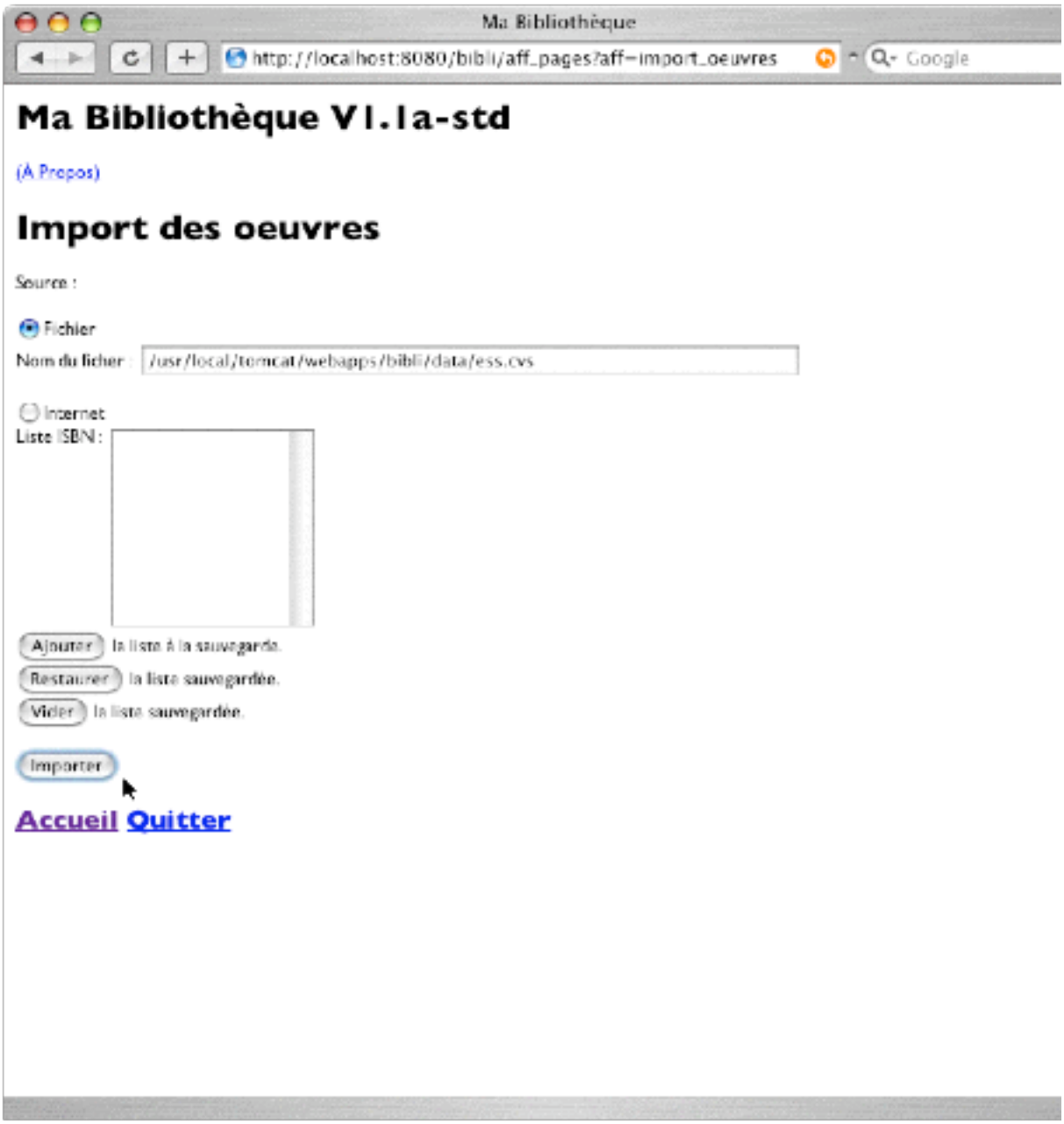
"Titre", "Année", "Editeur", "Genre", "Série/Collection", "Ordre dans la série", "Format", "Isbn", "Auteur1", "Prénom1", "Fonction1", "Auteur2", "Prénom2", "Fonction2"

Des guillemets vides "" sont nécessaires si le champ n'est pas rempli.
Au moment de l'importation le programme crée le titre, l'éditeur, la collection, les auteurs s'ils n'existent pas.

Le fichier ess.csv est fourni pour exemple.

Sélectionner le bouton "Fichier", entrer le chemin d'accès du fichier CSV et cliquer sur le bouton "Importer".

(voir écran ci-dessous)

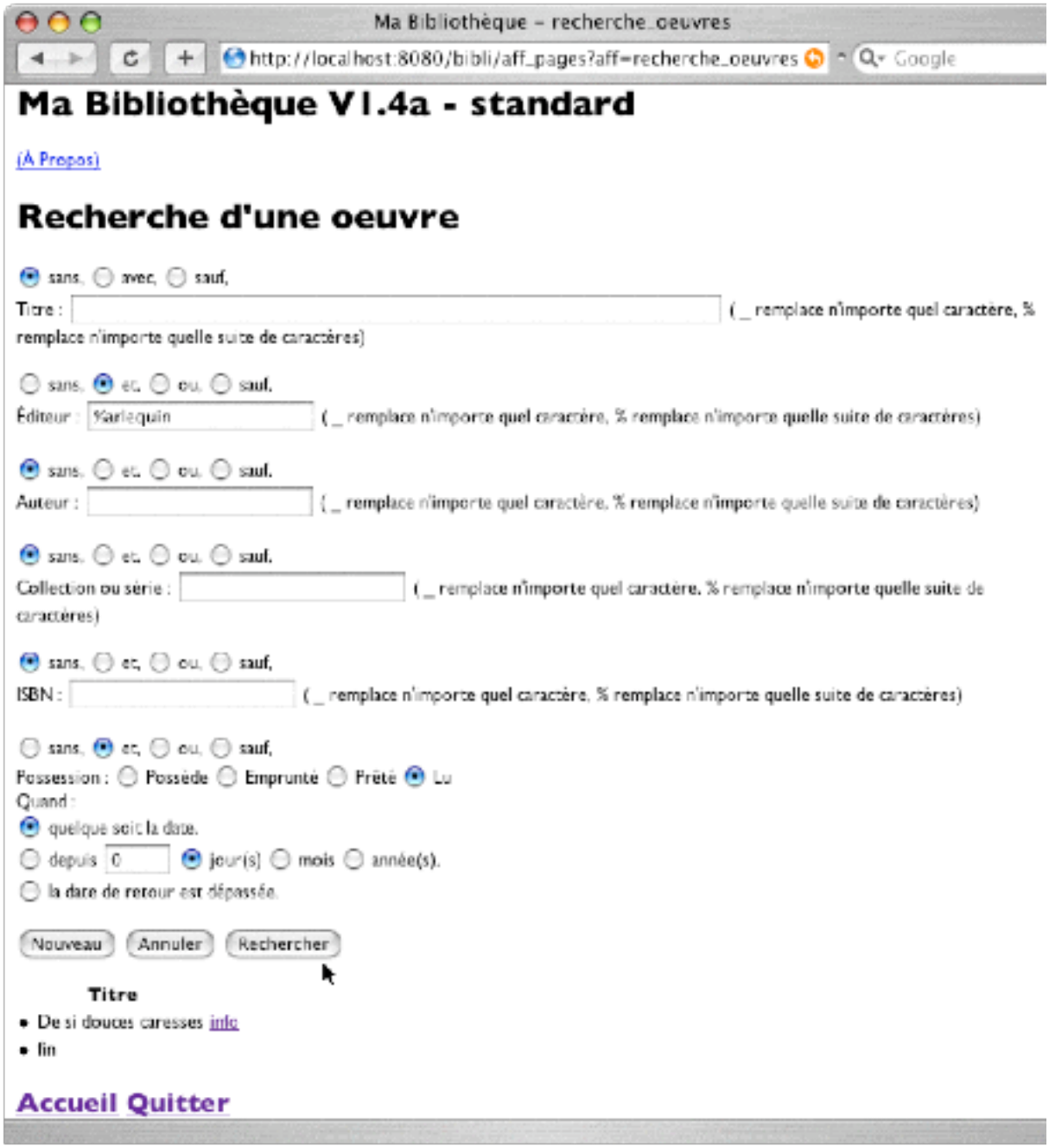


7) L'importation depuis Internet :

Sélectionner le bouton "Internet" de l'écran précédent, entrer le ou les numéros ISBN de l'oeuvre et cliquer sur le bouton "Importer".
Dans le cas d'une saisie sans connexion Internet, la liste des numéros ISBN peut être enregistrée (avec le bouton Ajouter) puis être rappelée plus tard (avec le bouton Restaurer) lorsque la connexion Internet sera active.
Le bouton Vider permet d'effacer le contenu de la liste sauvegardée.

8) La recherche :

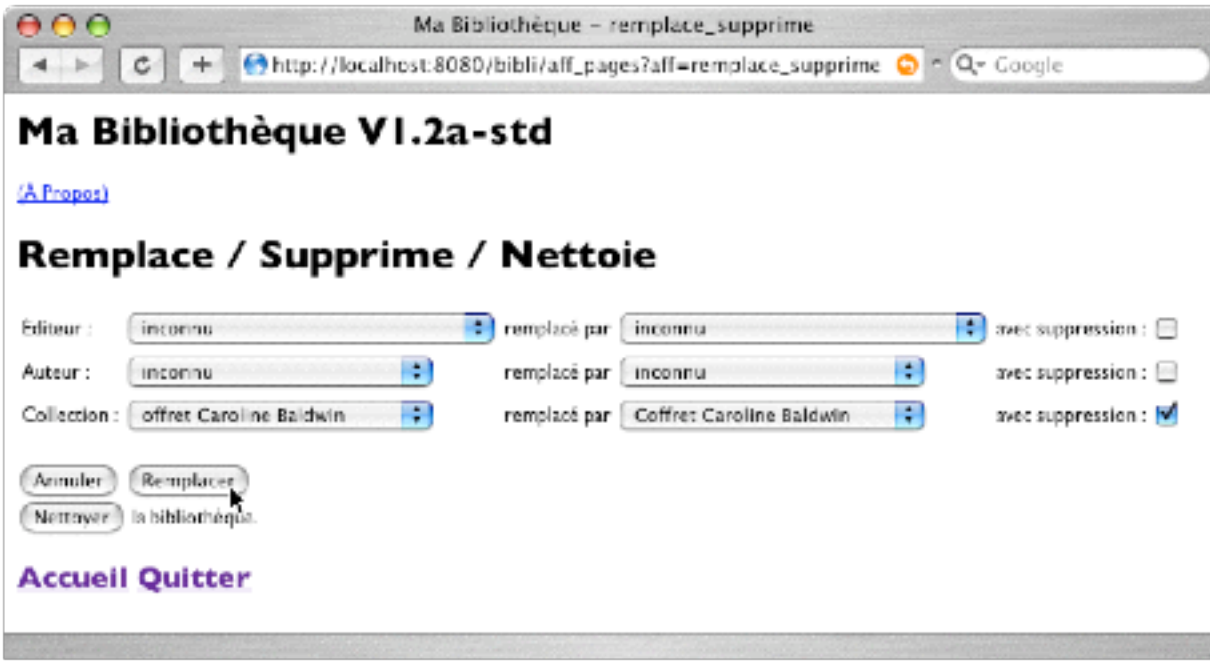
Sélectionner une rubrique, saisir la valeur à chercher puis cliquer sur le bouton Rechercher.



Vous pouvez utiliser différentes combinaisons d'opérateurs logiques entre les rubriques.
Le caractère _ (souligné) a une utilisation spéciale, il permet de remplacer n'importe quel caractère dans une recherche, par exemple : p_p_ va trouver papa, mais aussi pipi...
Le caractère % (pourcent) a aussi une utilisation spéciale, il permet de remplacer n'importe quelle suite de caractères dans une recherche, par exemple : p% va trouver tous les mots qui commencent par la lettre p.

9) Remplacer, supprimer, nettoyer :

Sélectionner la valeur à remplacer puis la valeur de remplacement, indiquer si la valeur à remplacer doit être supprimée de la base, enfin cliquer sur le bouton Remplacer.



Cet écran donne aussi la possibilité d'éliminer les enregistrements orphelins (des groupes d'auteurs qui ne correspondent plus à une oeuvre ou à un auteur) en cliquant sur le bouton Nettoyer.

10) Utilisation des énumérés :

Les énumérés sont utilisés pour proposer une liste de valeurs par défaut pour la saisie d'un champ comme la nationalité ou la fonction d'un auteur; la langue, le genre ou la nature d'une oeuvre.

Dans un premier temps, remplissez la table avec vos valeurs en utilisant l'écran de saisie des énumérés. Pour l'instant le seul groupe utilisé est "genre".
Entrer des valeurs comme "Science-Fiction", "Fantastique", etc.

Dans un écran de saisie, les valeurs sont présentées sous la forme de liste déroulante. Sélectionnez la valeur désirée et cliquez sur le bouton "Copier". La valeur sélectionnée sera alors présente dans le champ. Il ne restera plus qu'à enregistrer le tout.

Le mois prochain nous verrons la mise en place des sélections avec popups.

Pascal Pignard, janvier 2004 - juin 2005.