|  |
| --- |
| Isima 3ème année |
| Compte rendu du XSnoop |
| Programmation Web |
|  |
| **Maxime Escourbiac Jean-Christophe SEPTIER** |
| **01/02/2012** |

|  |
| --- |
|  |

# Introduction :

L’objectif de ce TP est de créer une application permettant de gérer une base donnée. Cette application respectera l’architecture 3-tiers afin de la séparer en 3 couches : le modèle, la vue (navigateur duc client) et le contrôleur. Il sera possible de se connecter à différentes bases de données (Access, MySql …).

Il sera possible de faire des recherches sur nos tables, supprimer des lignes, ajouter ou supprimer des tables. L’interface utilisateur devra être conviviale et il sera possible de changer dynamiquement les formulaires sans recharger la page. Il n’est pas nécessaire de connaître la syntaxe Sql pour utiliser cette application.

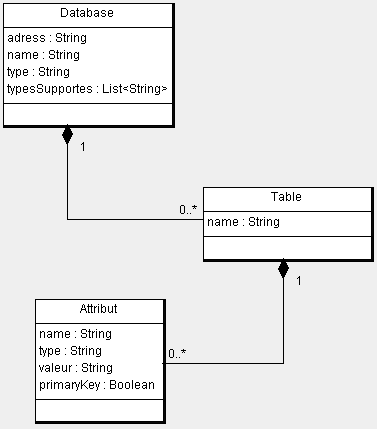
Dans ce rapport, nous allons d’abord présenter le modèle retenu pour représenter la base de données et l’analyse que nous avons faite, puis nous expliquerons comment nous avons implémenté notre application. Enfin nous vous présenterons l’ensemble de l’application avec les différentes fenêtres créées.

# Modélisation :

## Modélisation des bases de données.

Pour une question d’interopérabilité, on a décidé de modéliser les bases de données afin d’être plus manipulable par les différents objets du système. (Servlet, connecteur, requêtes etc…).

Voici le diagramme UML représentant cette modélisation.



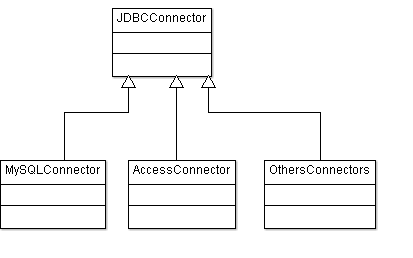
Pour des raisons de simplification, tous les types de données pour une catégorie de base ne sont pas pris en compte. Par exemple, pour notre XSnoop, le connecteur MySQL gère les types suivants :

* Integer.
* Float.
* Double.
* Varchar.

De la même façon, toutes les caractéristiques d’un attribut (clé étrangère, isnullable etc...) n’ont pas été modélisées, en effet l’ajout d’une clause FOREIGN\_KEY complexifierait le système retenu.

## Interface de connexion.

L’une des principales fonctions que nous avons choisi d’implémenter est la possibilité pour XSnoop de pouvoir travailler avec n’importe quel type de base de données (Access, MySQL, Oracle, etc...). Pour cela, nous avons implémenté le design-pattern Strategy pour les différents connecteurs aux bases de données. Ce qui donne le diagramme UML suivant.



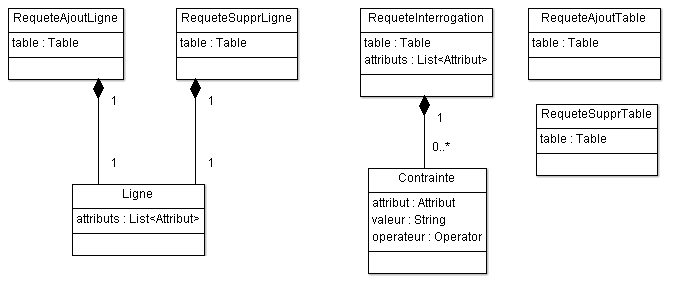
Ainsi, n’importe qui peut insérer un type de base de données dans XSnoop. Il lui suffira d’implémenter les méthodes abstraites de la classe JDBCConnector.

## Modèle de requête.

L’interface Web de XSnoop est identique quel que soit le type de base de données utilisé. C’est pour cela que l’on a été obligé de modéliser les requêtes à exécuter. Nous avons un modèle par type d’action.

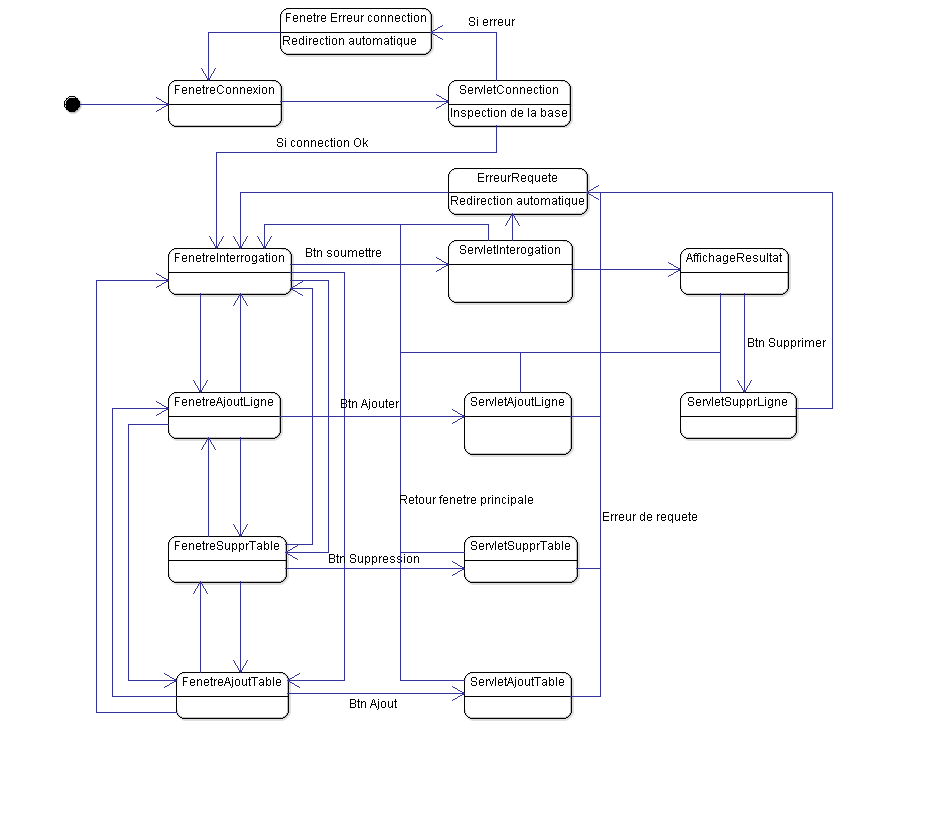
* RequeteAjoutLigne.
* RequeteAjoutTable.
* RequeteInterrogation.
* RequeteSupprLigne.
* RequeteSupprTable.

Les classes Table et Attribut sont les mêmes qui sont détaillés dans le modèle de la base de données.



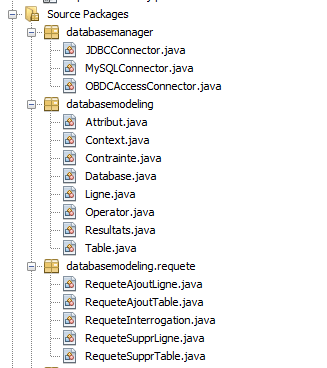
## Diagramme d’enchainement de l’application XSnoop.

Le développement d’une application Web comme XSnoop nécessite une réflexion plus approfondie sur l’enchainement des pages web et sur les appels des servlets. C’est pour cela qu’un diagramme est indispensable pour imager cette réflexion.



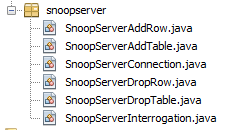
# Implémentation :

## Partie modèle :



La partie modèle a été réalisée en java. Cette partie modélise l’ensemble de classe vue dans le chapitre précédent. Le package datamanager implémente les connecteurs, le package databasemodeling implémente le modèle des bases de données, alors que le package databasemodeling.requete implémente le modèle de requête. Ces classes seront utilisés par la partie contrôle de notre application.

## Partie contrôle :



Nous avons utilisé pour la partie contrôle des Servlet. Ce sont des classes Java qui permettent de créer dynamiquement des données au sein d’un serveur http. Ces classes correspondent aux différentes opérations que l’on peut effectuer sur notre base de données (supprimer une table, ajouter une ligne …). Elles sont appelées par la partie vue. Afin de communiquer avec la partie vue, on sauvegarde dans une session un contexte. Ce contexte possède le login et le mot de passe (pour ne pas être réutilisé pour effectuer nos opérations), la base sauvegardée, et les résultats de nos requêtes (s’il existe).

## Partie Vue :

La partie vue permet de communiquer avec l’utilisateur. Pour cela, on a utilisé plusieurs langages :

### Html et css :

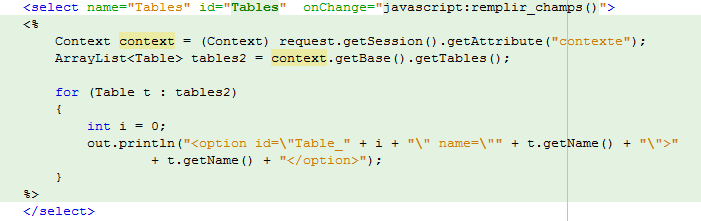
Le HTML permet d’afficher les parties non dynamiques de notre site web. Il permet de créer des formulaires qui permettront d’appeler nos servlets. Le menu a été réalisé dans un fichier Html est inclus dans chaque page afin d’afficher le même menu dans chaque page.

Le CSS est un langage qui permet de décrire la présentation des documents Html. Il permet d’améliorer la présentation des pages Web et les rendre plus attractives. Toutes les pages utilisent le même fichier CSS.



### JSP :

Les JSP sont des pages qui permettent de générer dynamiquement du code HTML. On ajoute dans le code Html des balises afin de générer du code. Les JSP vont utiliser les données fournies par les servlets dans notre contexte afin d’afficher du code. Par exemple, les options d’une liste contenant les différentes tables sont ajoutées grâce à des JSP.

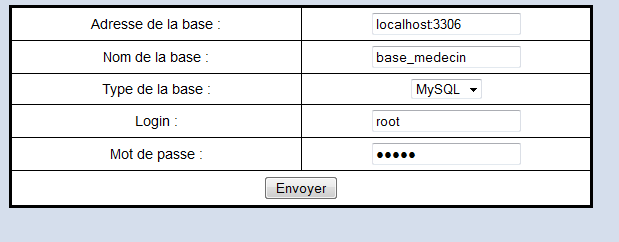


### JavaScript :

Le JavaScript est un langage de script utilisé dans les pages web interactives. Le JavaScript est exécuté sur le poste client. Le navigateur va prendre en charge l’exécution des scripts. Ces scripts permettent donc de modifier l’affichage de notre page ou des réaliser certaines actions sans avoir à recharger la page et repasser par le serveur. Nous les avons utilisés afin de modifier nos formulaires dynamiquement. Par exemple, changer le nombre de champs à ajouter à une table sans recharger la page, ou afficher les champs correspondant à une table (si on en sélectionne une autre).

# Présentation de l’application :

## Page de connexion :

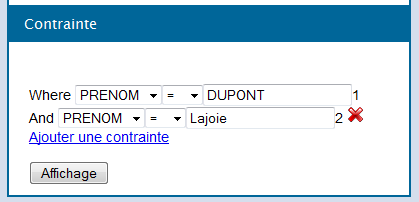


Lorsque l’on ouvre l’index, on atterrit sur un tableau permettant d’accéder à notre base de données. On entre l’adresse de la base, le nom de la base, le type (MySQL ou ODBC) et enfin le login et le mot de passe. Après la connexion, un contexte est sauvegardé afin de garder en session l’ensemble des tables et leurs attributs. En cas d’erreur, un page est appelé pour afficher un message d’erreur. Sinon, on accède au reste du site.

## Page du select :

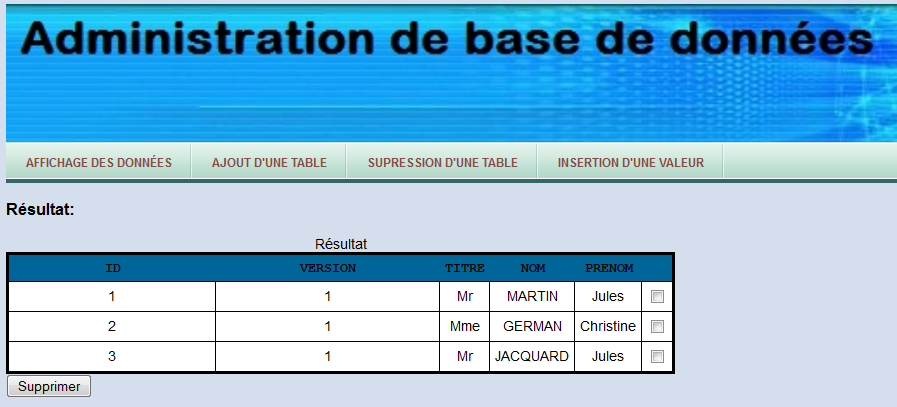


Cette fenêtre permet de réaliser nos requêtes sur la base de données sur laquelle nous nous sommes connectés. Les tables sont sauvegardées et tout changement de cette page se fait sans recharger la page à l’aide de fonction JavaScript. Nous choisissons d’abord la table sur laquelle effectuée la requête. Les différents champs à afficher se mette à jour dynamiquement, avec des checkbox. On coche les champs que l’on veut afficher. Si aucun champ n’est sélectionné, tous les champs seront affichés. Ensuite, on peut ajouter des contraintes sur nos champs.



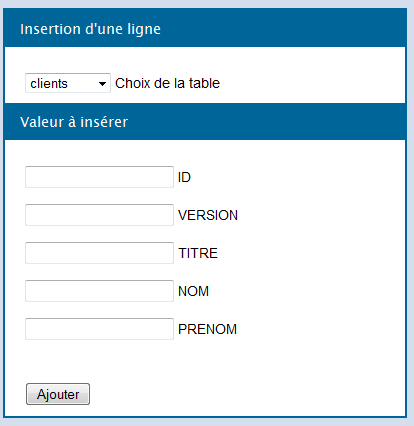
On peut alors choisir le champ sur lequel on veut mettre une contrainte. On peut supprimer une contrainte en cliquant sur la croix rouge à côté. On peut choisir les opérations = , != , < , > , <= et <=. On entre ensuite la valeur de notre contrainte. Pour afficher notre résultat, on clique ensuite sur le bouton « Affichage ».

## Page d’affichage et de suppression d’une ligne:



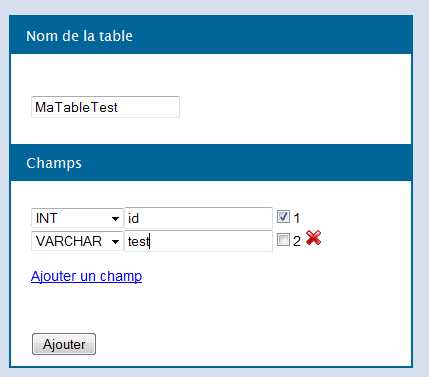
La page d’affichage des résultats permet d’afficher l’ensemble des lignes de notre requête sous forme d’un tableau Html. On peut également supprimer une ligne en cochant la checkbox au bout de la ligne, et en cliquant sur le bouton « supprimer ».

## Page d’ajout d’une ligne :



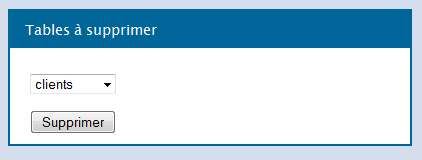
Cette fenêtre permet d’ajouter une ligne dans une de nos tables. On choisit la table, puis les champs de la table sont affichés dynamiquement à l’aide de javaScript. Le bouton afficher permet d’ajouter notre ligne.

## Page d’ajout de table :



Cette page permet d’ajouter une nouvelle table à notre base de données. On entre d’abord le nom de la table, puis les champs de la table. On peut ajouter ou supprimer des champs dynamiquement en Css. On choisit le type des champs (les types gérables par l’application sont envoyés par la servlet) et le nom du champ. On détermine la ou les clés primaires en cochant la checkBox à droite du champ correspondant.

## Suppression de tables :



Cette page permet de supprimer une table. On sélectionne la table que l’on veut supprimer et on clique sur le bouton supprimer.

# Conclusion

Nous avons réussi à réaliser une application simple d’utilisation, et dynamique afin que n’importe quel utilisateur puisse réaliser des requêtes simples sur des bases de données. L’utilisation du JavaScript permet une utilisation plus aisé afin d’éviter un grand nombre de chargement de page pour modifier nos formulaires. Le modèle 3-tiers a été respecté, avec les trois couches bien séparées.

Grâce à notre modélisation d’un pattern stratégie, il est possible de connecter facilement notre application à tout type de base de données.