Prise en main d'un framework

Introduction

Objectifs

- Découverte d’un framework
- Normalisation du Code
- Mapping relationnel/objet
- Conception de vues

Ressources

- Tutoriels KObject Java EE
- Librairie KObject-1.0.0.25a-beta1

Contexte

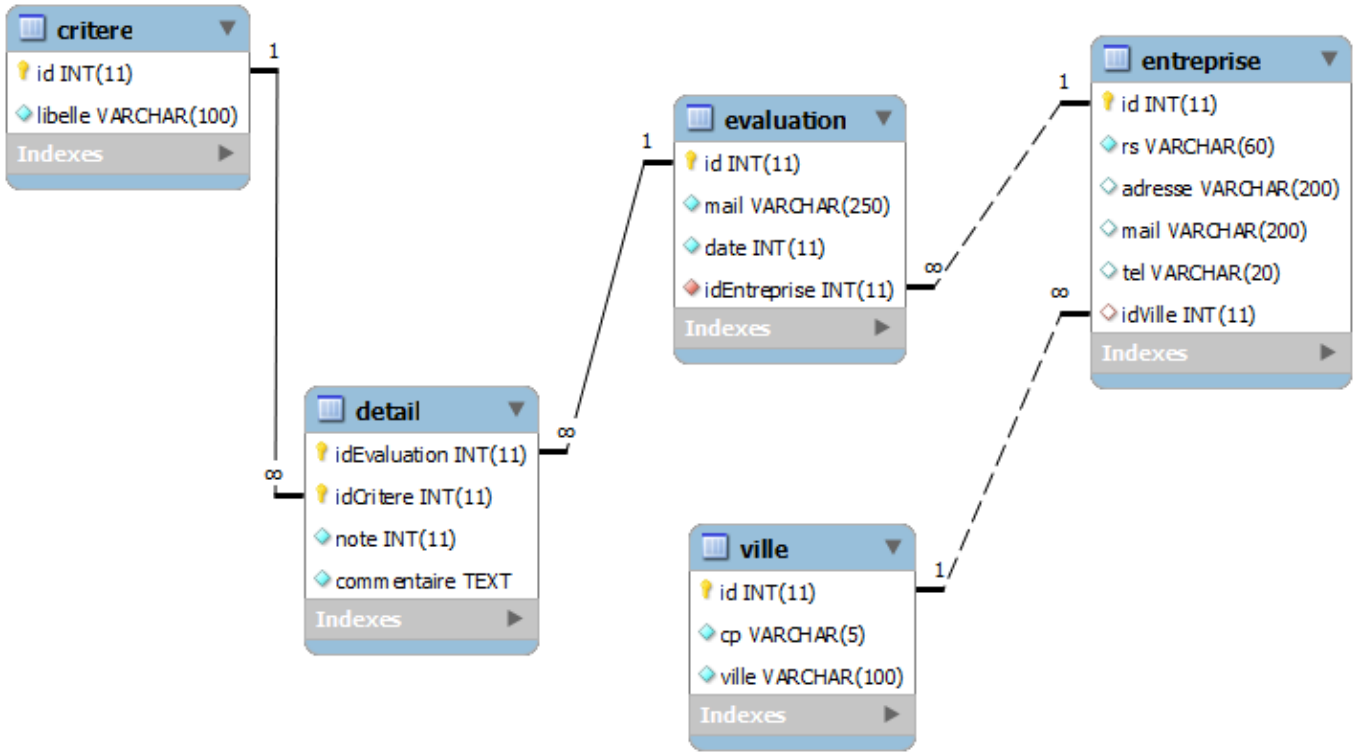
Il s'agit de concevoir une application web (exemple) permettant aux internautes d'évaluer des entreprises.

Principales fonctionnalités :

- Rechercher une entreprise
- Lister des entreprises
- Evaluer une entreprise
- Consulter le classement des entreprises évaluées

Base de données

- Créer la base de données **avisDb**, en exécutant le script dans phpMyAdmin : avisdb.sql



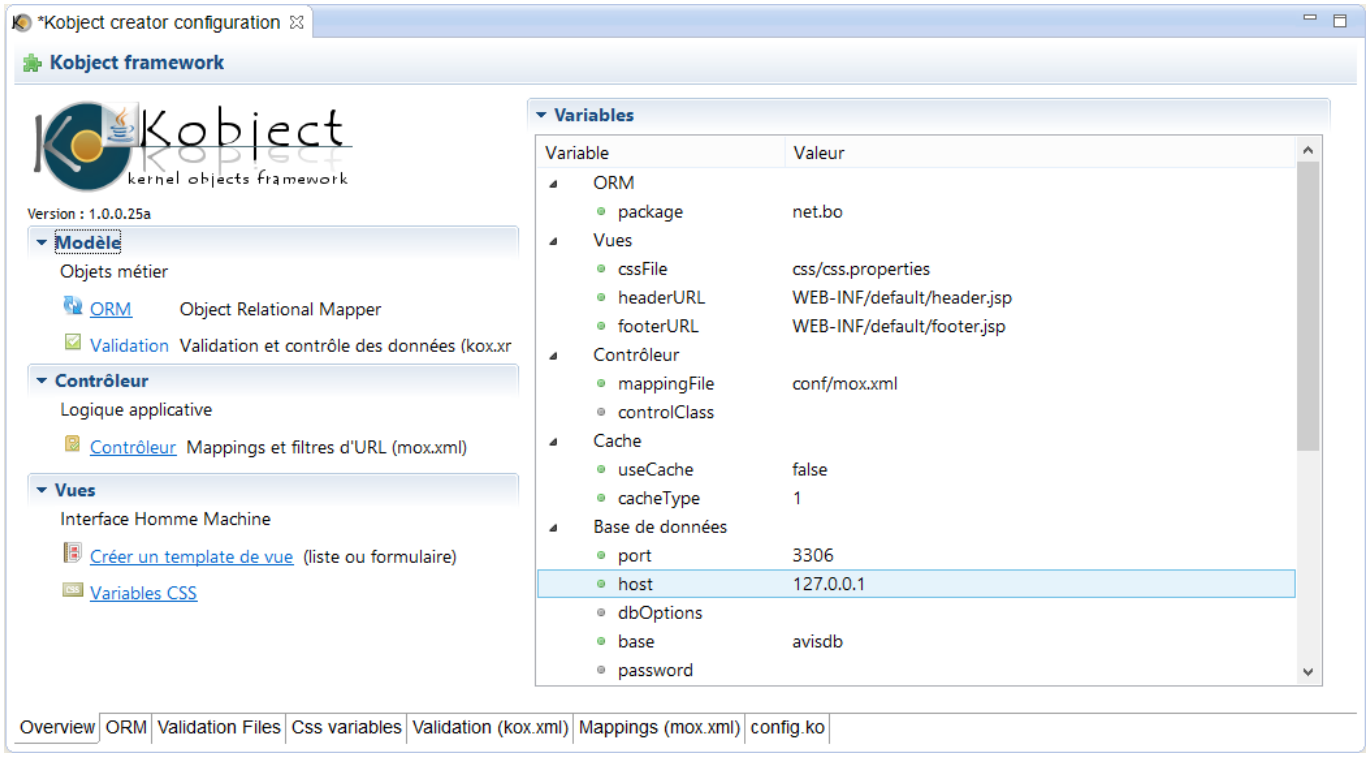
Travail à faire

-- Configuration

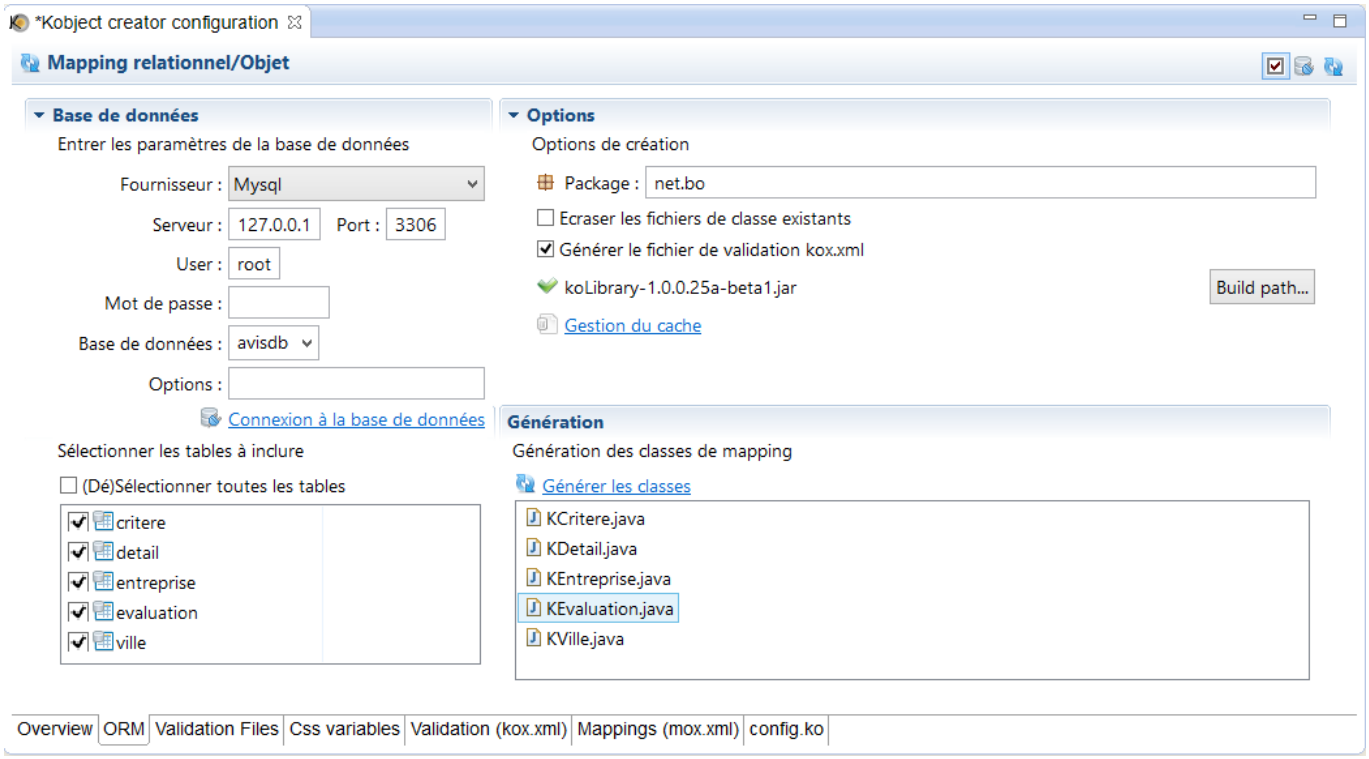
1. Dans Eclipse, installer le plugin KObject
2. Créer le Dynamic web project **koTd5**
3. Ajouter les jars suivants dans le dossier **WebContent/WEB-INF/lib** du projet
    - mysql-connector-java-5.1.14-bin.jar
    - Librairie KObject 1.0.0.25a
4. Ajouter KObject au projet
5. Remplacer le fichier **userStyle.css** dans le dossier **css** : userstyle.css

-- Génération du modèle

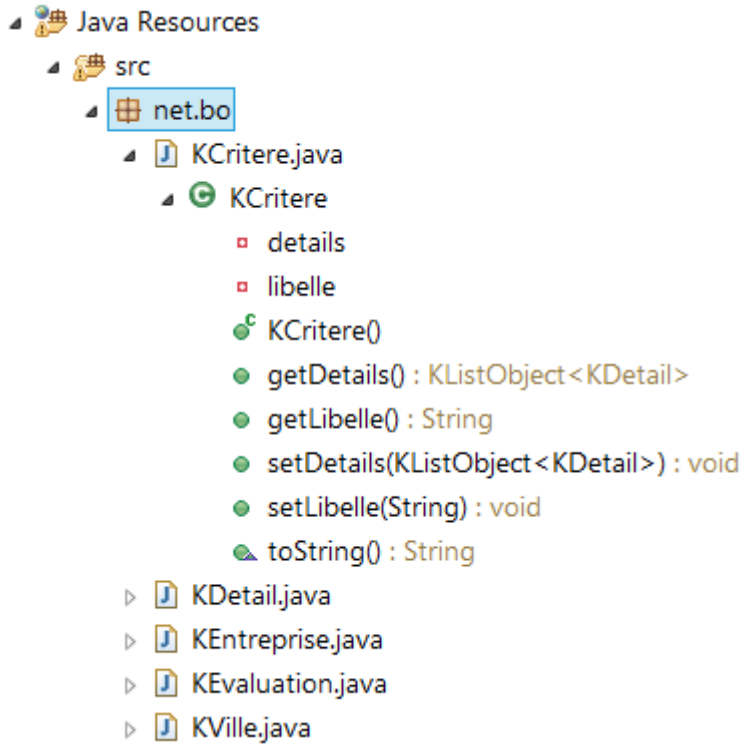
- A partir d'Eclipse, ouvrir le fichier de configuration **WebContent/config.ko** :



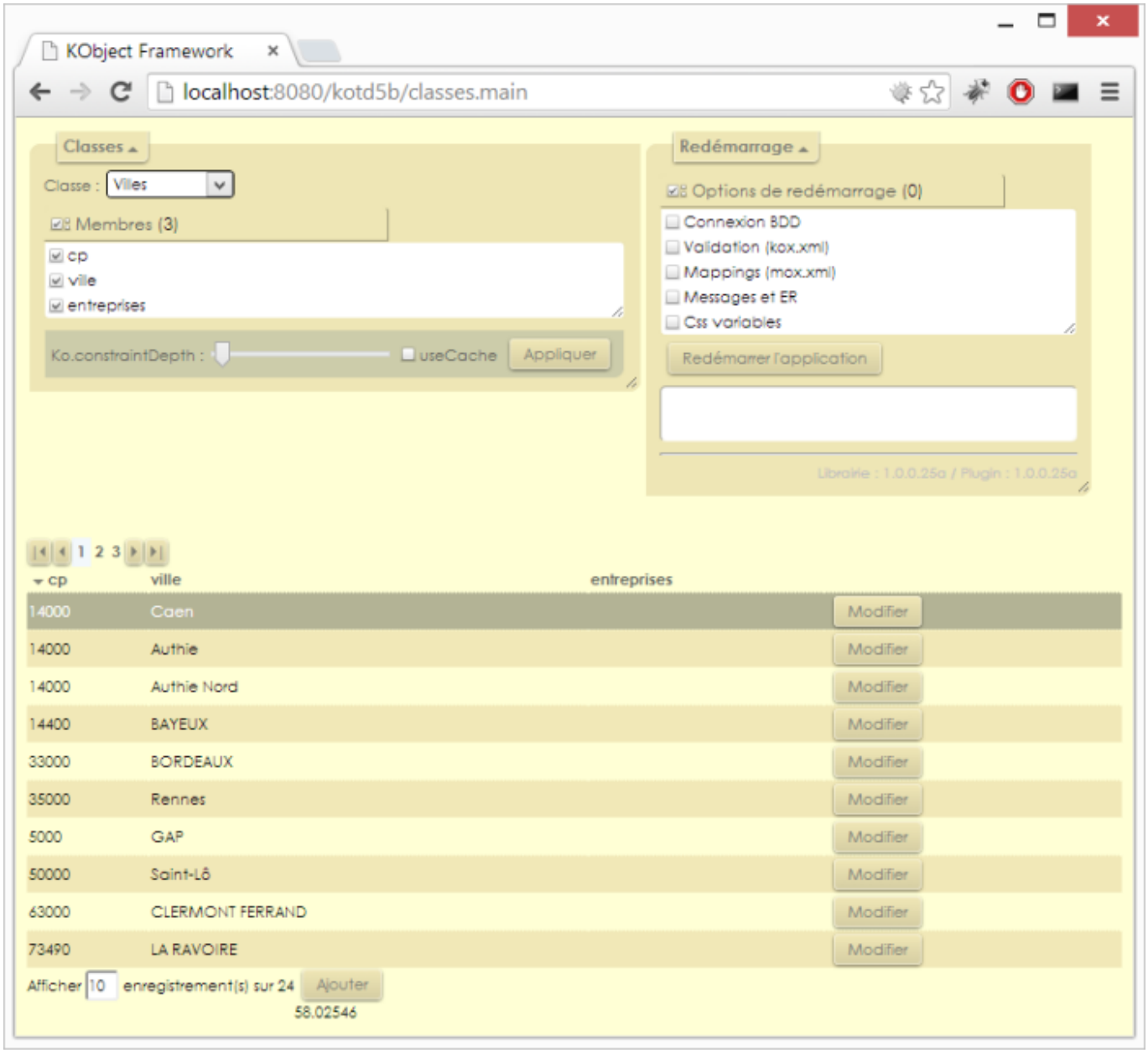
- Activer l'onglet **ORM** et générer les classes métier



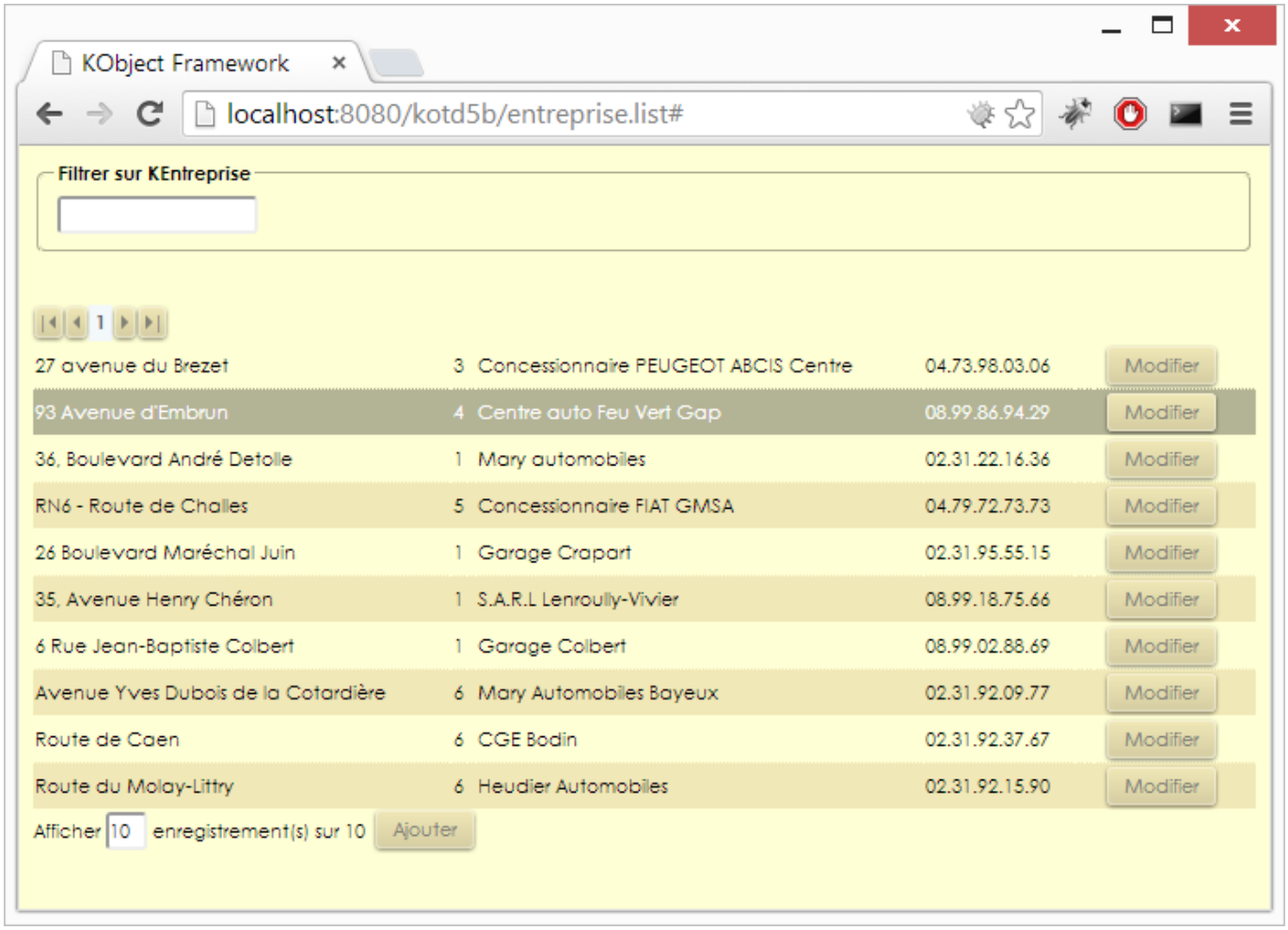
- Vérifier la création des classes dans le package net.bo :

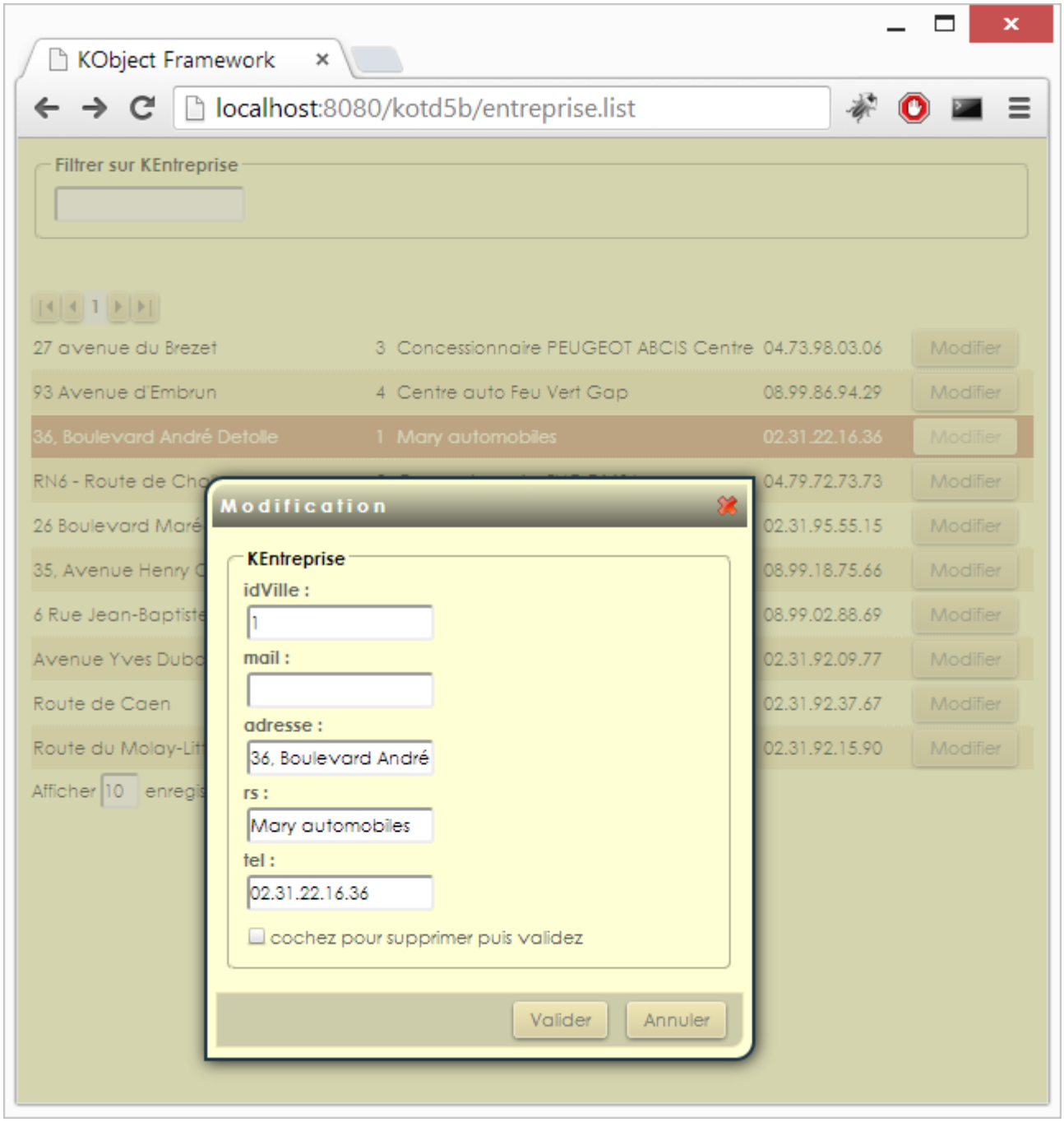


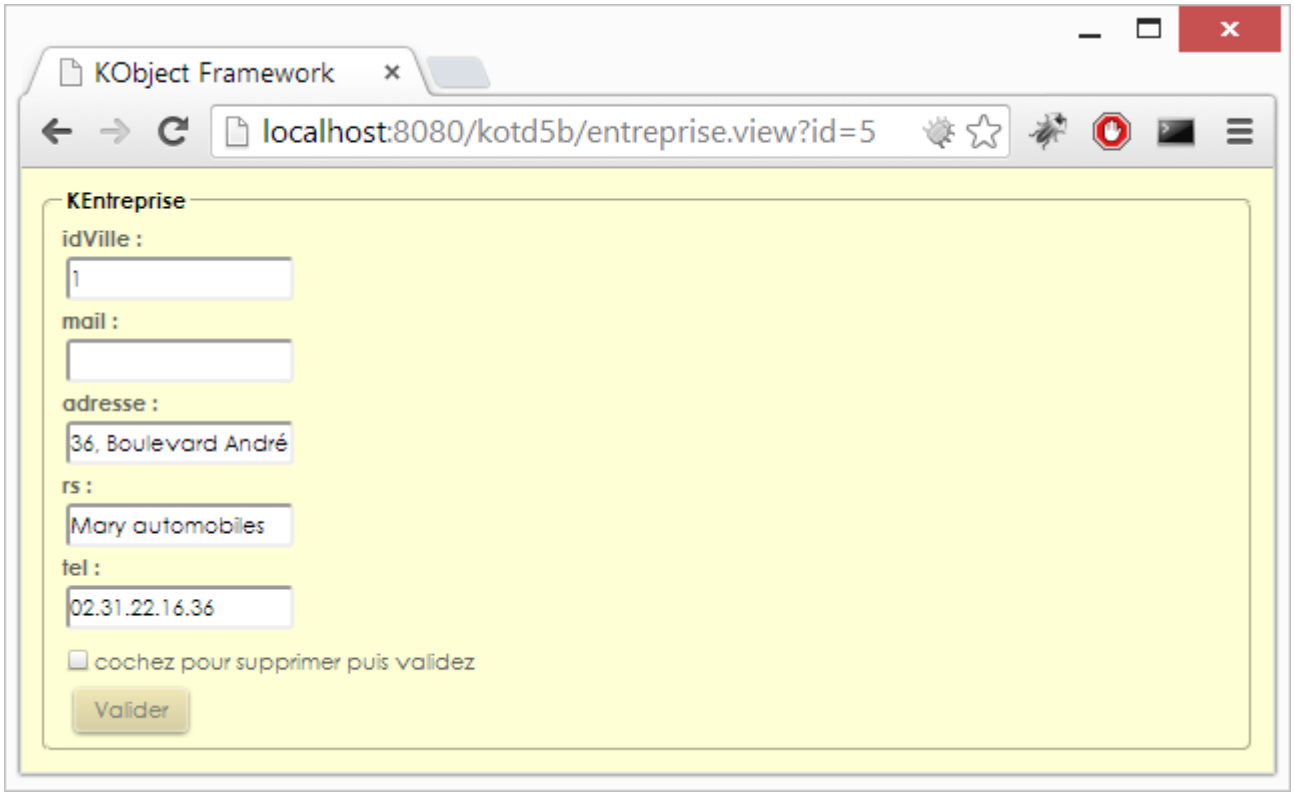
- Tester le bon fonctionnement en allant à l'adresse **http://127.0.0.1:8080/koTd5/classes.main** de l'application web.



- Tester les adresses suivantes :
  - http://127.0.0.1:8080/koTd5/entreprise.list
  - http://127.0.0.1:8080/koTd5/entreprise.view
  - http://127.0.0.1:8080/koTd5/ville.list
  - http://127.0.0.1:8080/koTd5/ville.view
  - ...


KObject Framework
localhost:8080/kotd5b/entreprise.list#
Filtrer sur KEntreprise
1
27 avenue du Brezet 3 Concessionnaire PEUGEOT ABCIS Centre 04.73.98.03.06 Modifier
93 Avenue d'Embrun 4 Centre auto Feu Vert Gap 08.99.86.94.29 Modifier
36, Boulevard André Detolle 1 Mary automobiles 02.31.22.16.36 Modifier
RN6 - Route de Challes 5 Concessionnaire FIAT GMSA 04.79.72.73.73 Modifier
26 Boulevard Maréchal Juin 1 Garage Crapart 02.31.95.55.15 Modifier
35, Avenue Henry Chéron 1 S.A.R.L Lenroully-Vivier 08.99.18.75.66 Modifier
6 Rue Jean-Baptiste Colbert 1 Garage Colbert 08.99.02.88.69 Modifier
Avenue Yves Dubois de la Cotardière 6 Mary Automobiles Bayeux 02.31.92.09.77 Modifier
Route de Caen 6 CGE Bodin 02.31.92.37.67 Modifier
Route du Molay-Littry 6 Heudier Automobiles 02.31.92.15.90 Modifier
Afficher 10 enregistrement(s) sur 10 Ajouter


KObject Framework
localhost:8080/kotd5b/entreprise.list
Filtrer sur KEntreprise
27 avenue du Brezet 3 Concessionnaire PEUGEOT ABCIS Centre 04.73.98.03.06 Modifier
93 Avenue d'Embrun 4 Centre auto Feu Vert Gap 08.99.86.94.29 Modifier
36, Boulevard André Detolle 1 Mary automobiles 02.31.22.16.36 Modifier
Modification
KEntreprise
idVille :
1
mail :
adresse :
36, Boulevard André
rs :
Mary automobiles
tel :
02.31.22.16.36
cochez pour supprimer puis validez
Valider
Annuler



-- Modifications métier

Classes

Décommenter la relation **belongsTo** dans la classe **KEntreprise** :

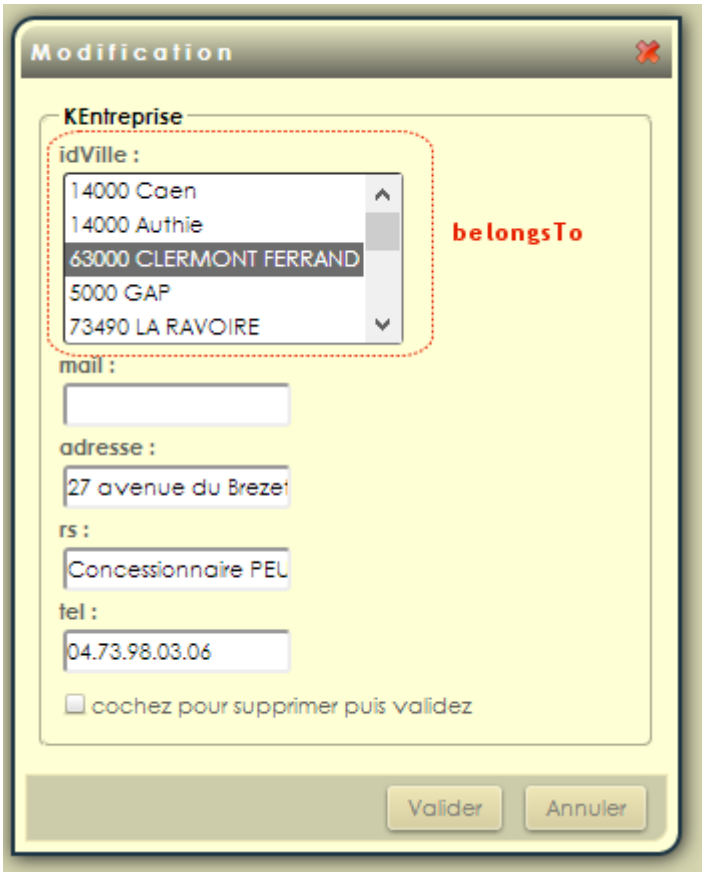
```
    public KEntreprise() {
        super();
        belongsTo(KVille.class);//hasMany(KEvaluation.class);
    }
```

Modifier la méthode **toString** de la classe **KVille** :

```
public class KVille extends KObject {
    ...
    @Override
    public String toString() {
        return cp +" "+ville;
    }
```

- Afficher à nouveau **entreprise.list** et **entreprise.view** pour constater les modifications :





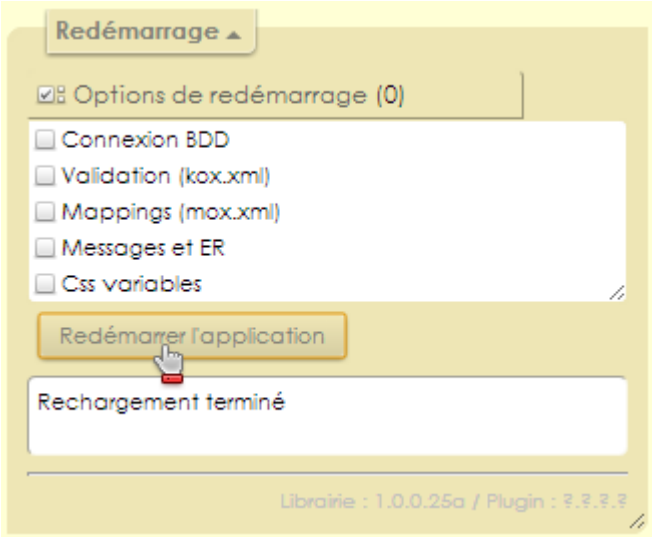
conf/kox.xml

- Ajouter les attributs **caption** sur les membres de la classe **KEntreprise**, et modifier l'ordre des lignes comme ci-dessous :

```
    <class name="KEntreprise" caption="Entreprise">
        <member max="60" name="rs" required="1" type="string" caption="Raison
sociale"/>
        <member max="200" name="adresse" required="0" type="string"
caption="Adresse"/>
        <member max="11" name="idVille" required="0" type="int" caption="Ville"/>
```

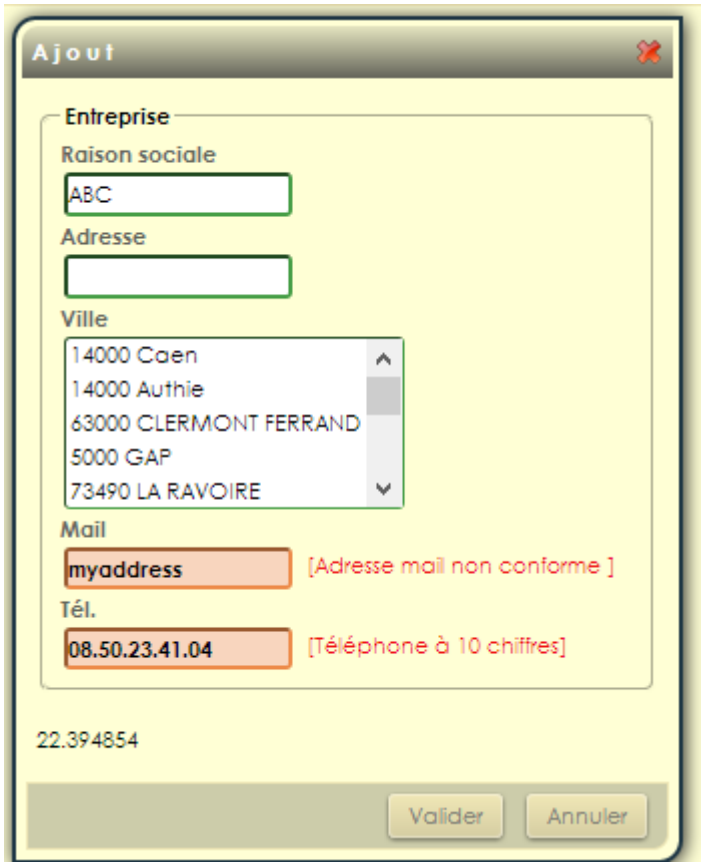
```
        <member max="200" name="mail" required="0" type="string" caption="Mail"/>
        <member max="20" name="tel" required="0" type="string" caption="Tél."/>
    </class>
```

- Redémarrez l'application à partir de **classes.main** et affichez le formulaire **entreprise.view**.



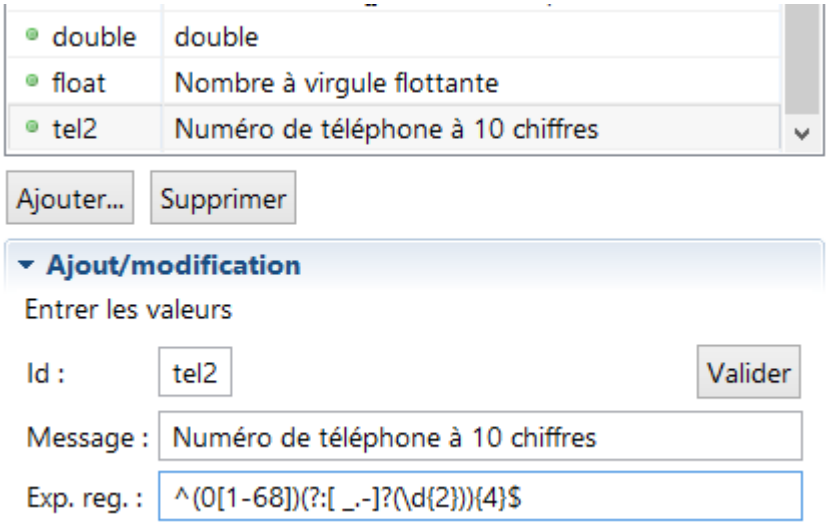
- Modifier les attributs **type** des membres **tel** et **mail** de la classe **KEntreprise** :

```
    <class name="KEntreprise" caption="Entreprise">
        <member max="60" name="rs" required="1" type="string" caption="Raison
sociale"/>
        <member max="200" name="adresse" required="0" type="string"
caption="Adresse"/>
        <member max="11" name="idVille" required="0" type="int" caption="Ville"/>
        <member max="200" name="mail" required="0" type="mail" caption="Mail"/>
        <member max="20" name="tel" required="0" type="tel" caption="Tél."/>
    </class>
```



Validation JS personnalisée

- Ouvrir le fichier **config.ko** avec l'outil de configuration :
- Activer l'onglet **Validation files**
- Choisir **Ajouter** dans la zone **contrôle**
- Entrer la nouvelle expression régulière **tel2** pour les numéros de téléphone : **^(0[1-68])(?:[_.-]?(\d{2})){4}$**



* Associer **tel2** au type du membre **tel** de **KEntreprise** dans **kox.xml** :

```
    <class name="KEntreprise" caption="Entreprise">
        <member max="60" name="rs" required="1" type="string" caption="Raison
sociale"/>
```

```
        <member max="200" name="adresse" required="0" type="string"
caption="Adresse"/>
        <member max="11" name="idVille" required="0" type="int" caption="Ville"/>
        <member max="200" name="mail" required="0" type="mail" caption="Mail"/>
        <member max="20" name="tel" required="0" type="tel2" caption="Tél."/>
    </class>
```

- Vérifier la validité de l'expression dans le formulaire **entreprise.view** en ayant au préalable redémarré l'application via **classes.main**

Transformation

Même si la validité du numéro de téléphone est contrôlée, il se peut que les numéros saisis ne soient pas uniformes : En effet, les formats suivants des numéros sont acceptés :

0231101112, 02.31.10.11.12, 02-31-10-11-12, 02 31 10 11 12

Nous allons créer un Transformer pour faire en sorte que tous les numéros saisis aient le même format : **02.31.10.11.12**

Création d'une classe Transformer

- Créer la classe **MyTransformers** dans le package **net.validation** :

```
package net.validation;

import net.ko.controller.KTransformer;

public class MyTransformers extends KTransformer {
    public static String tel(String value){
        return value.replaceAll("\\W", ".");
    }
}
```

Association au membre tel de KEntreprise

```
    <class name="KEntreprise" caption="Entreprise"
transformer="net.validation.MyTransformers">
        <member max="60" name="rs" required="1" type="string" caption="Raison
sociale"/>
        <member max="200" name="adresse" required="0" type="string"
caption="Adresse"/>
        <member max="11" name="idVille" required="0" type="int" caption="Ville"/>
        <member max="200" name="mail" required="0" type="mail" caption="Mail"/>
        <member max="20" name="tel" required="0" type="tel2" caption="Tél."
transform="tel"/>
    </class>
```

D'autres Transformers existent par défaut dans **KObject** et peuvent être utilisés : voir http://tutorial.kobject.net/java/configcontroller.

-- Logique applicative

mappings

Un **mapping** permet d'associer une url à une ressource présente sur le serveur. Toutes les requêtes d'une application KObject doivent logiquement passer par des mappings d'url, par défaut en ***.do**

- Créer la page index.jsp dans WEB-INF :

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ include file="header.jsp"%>
<div id="main">
    <div id="menu">
        <a id="entreprises" title="Afficher la liste des entreprises">Liste des
entreprises</a> | 
        <a id="villes" title="Afficher la liste des villes">Liste des villes</a>
    </div>
    <div id="content">
    Main page
    </div>
    <div id="message"></div>
</div>
<%@ include file="footer.jsp"%>
```

- Créer les mappings suivants dans **conf/mox.xml**, au sein de la balise **mappings** :

Une requête vers	Affichera la page
index.do	/WEB-INF/index.jsp
entreprises.do	/WEB-INF/entreprises.jsp
villes.do	/WEB-INF/villes.jsp

```
    <mappings>
        <mapping requestURL="index.do" responseURL="/WEB-INF/index.jsp"/>
        <mapping requestURL="entreprises.do" responseURL="/WEB-
INF/entreprise.list"/>
        <mapping requestURL="villes.do" responseURL="/WEB-INF/ville.list"/>
    </mappings>
```

Inclusions ajax

Il s'agit maintenant de créer la dynamique de l'interface : la logique applicative côté client.

Toutes les modifications de la logique client sont implémentées dans le fichier **conf/mox.xml**

- Sur la page **index.do**,

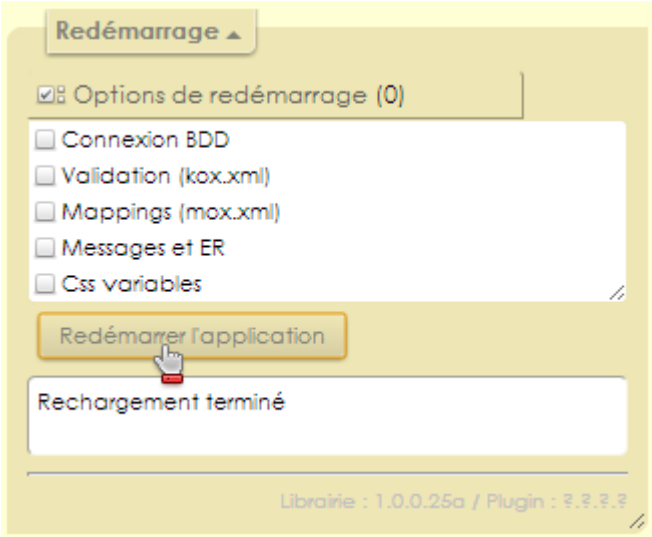
    - un click sur le bouton d'id **entreprises**
        - affichera la réponse ajax(get) **entreprises.do** dans l'élément HTML d'id **content**.

```
<ajax-includes>
    <request requestURL="index.do">
        <js triggerSelector="#entreprises">
            <include targetURL="entreprises.do" targetId="content"/>
        </js>
    </request>
</ajax-includes>
```

- Ajouter le second mapping :

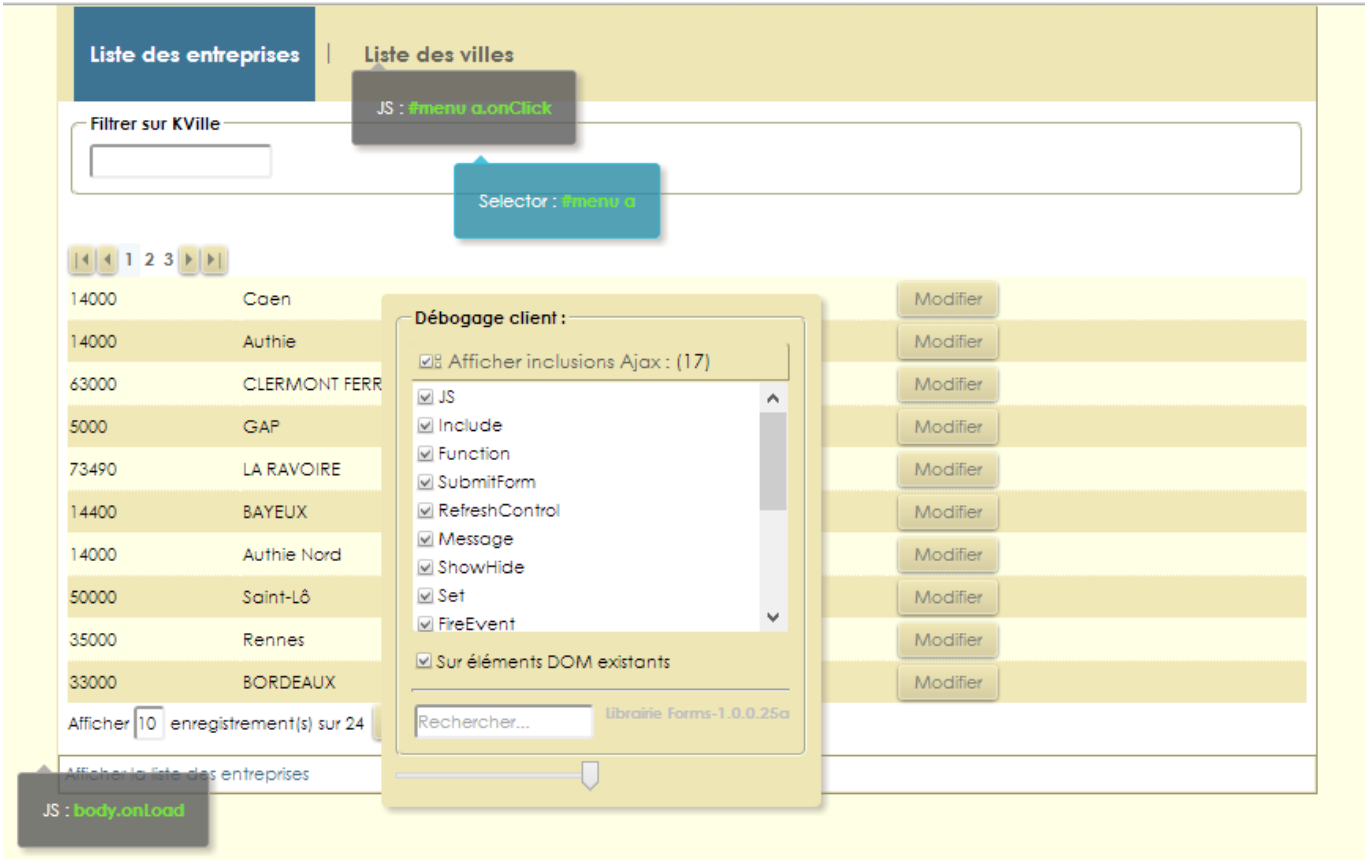
```
<ajax-includes>
    <request requestURL="index.do">
        <js triggerSelector="#entreprises" triggerEvent="click">
            <include targetURL="entreprises.do" targetId="content"/>
        </js>
        <js triggerSelector="#villes" triggerEvent="click">
            <include targetURL="villes.do" targetId="content"/>
        </js>
    </request>
</ajax-includes>
```

- Redémarrer l'application avec **classes.main** pour prendre en compte les changements dans **mox.xml** :



- Tester ensuite le fonctionnement de la page **index.do**

- Activer le débogage côté client dans le fichier config.ko, en mettant **clientDebug** à **true**
- Redémarrer l'application à partir de **classes.main**
- Tester à nouveau les pages pour visualiser la bonne insertion des inclusions : voir http://tutorial.kobject.net/java/debug/client



- Penser aussi à utiliser les fonctionnalités de débogage du navigateur. voir Techniques de débogage

Factorisation des inclusions

Il est possible de réduire les 2 inclusions précédentes à une seule :

- Sur la page **index.do**,
  - un click sur un lien **a** dans la zone d'id **menu**
    - affichera la réponse ajax(get) **{id du lien}.do** dans l'élément HTML d'id **content**.

```
<ajax-includes>
    <request requestURL="index.do">
        <js triggerSelector="#menu a" triggerEvent="click">
            <include targetURL="{js:target.id}.do" targetId="content"/>
        </js>
    </request>
</ajax-includes>
```

Selector

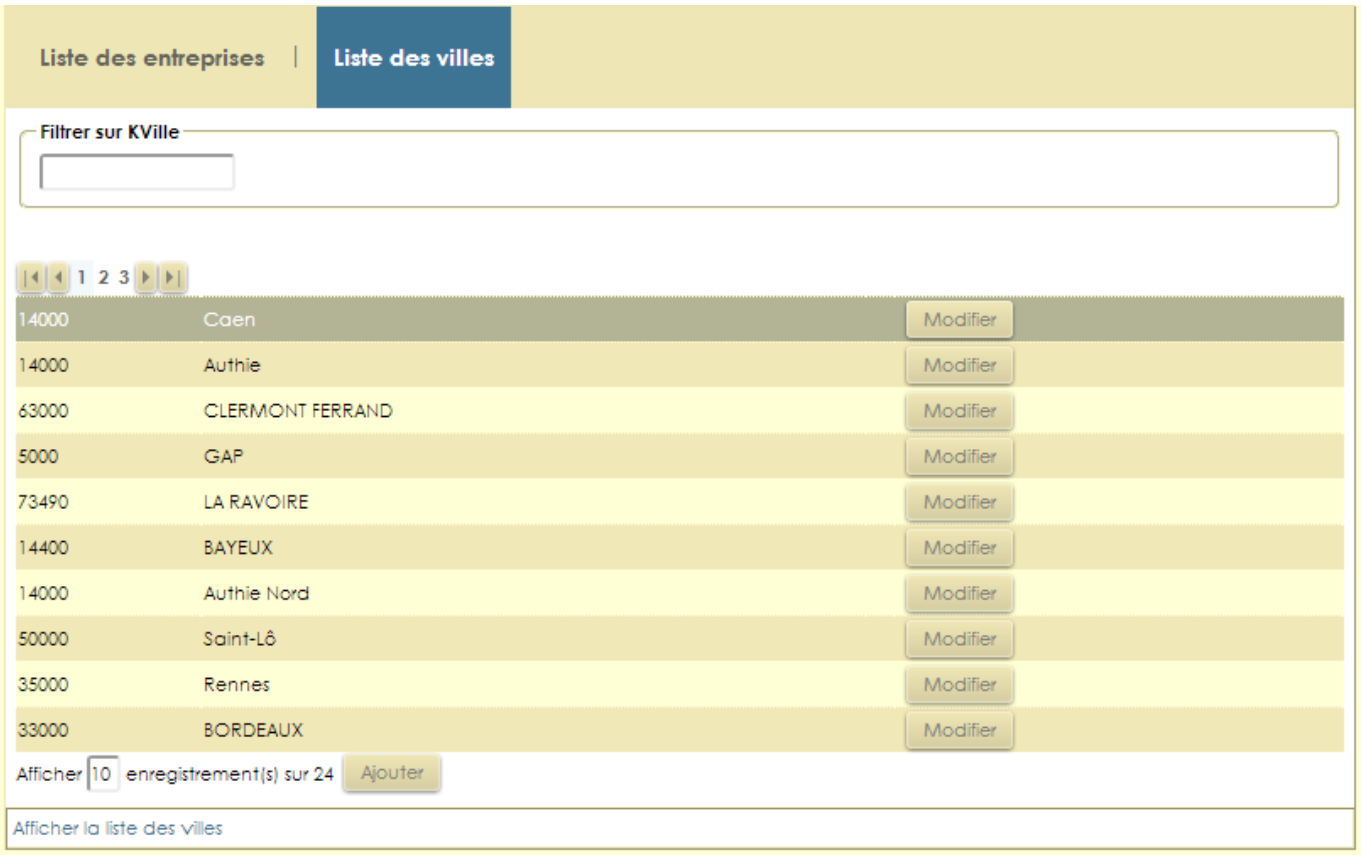
Le sélecteur est une inclusion ajax permettant d'associer du code à un ensemble d'éléments, l'un d'entre eux devenant à l'exécution l'élément actif.

- Sur la page **index.do**,
  - la souris entrante sur un lien **a** dans la zone d'id **menu**

- déclenche l'affichage du message **title du lien a survolé** dans l'élément HTML d'id **message**, le lien a devient l'élément actif.

```
<ajax-includes>
    <request requestURL="index.do">
        <js triggerSelector="#menu a">
            <include targetURL="{js:target.id}.do" targetId="content"/>
        </js>
        <js triggerSelector="body" triggerEvent="load">
            <selector selector="#menu a" event="mouseenter" startIndex="0">
                <message targetId="message">target.title</message>
            </selector>
        </js>
    </request>
</ajax-includes>
```

- Tester les pages :



-- Templates

Pour l'instant, les vues affichées (formulaires et listes) sont les pages par défaut : voir http://tutorial.kobject.net/java/khttpdefaultpages

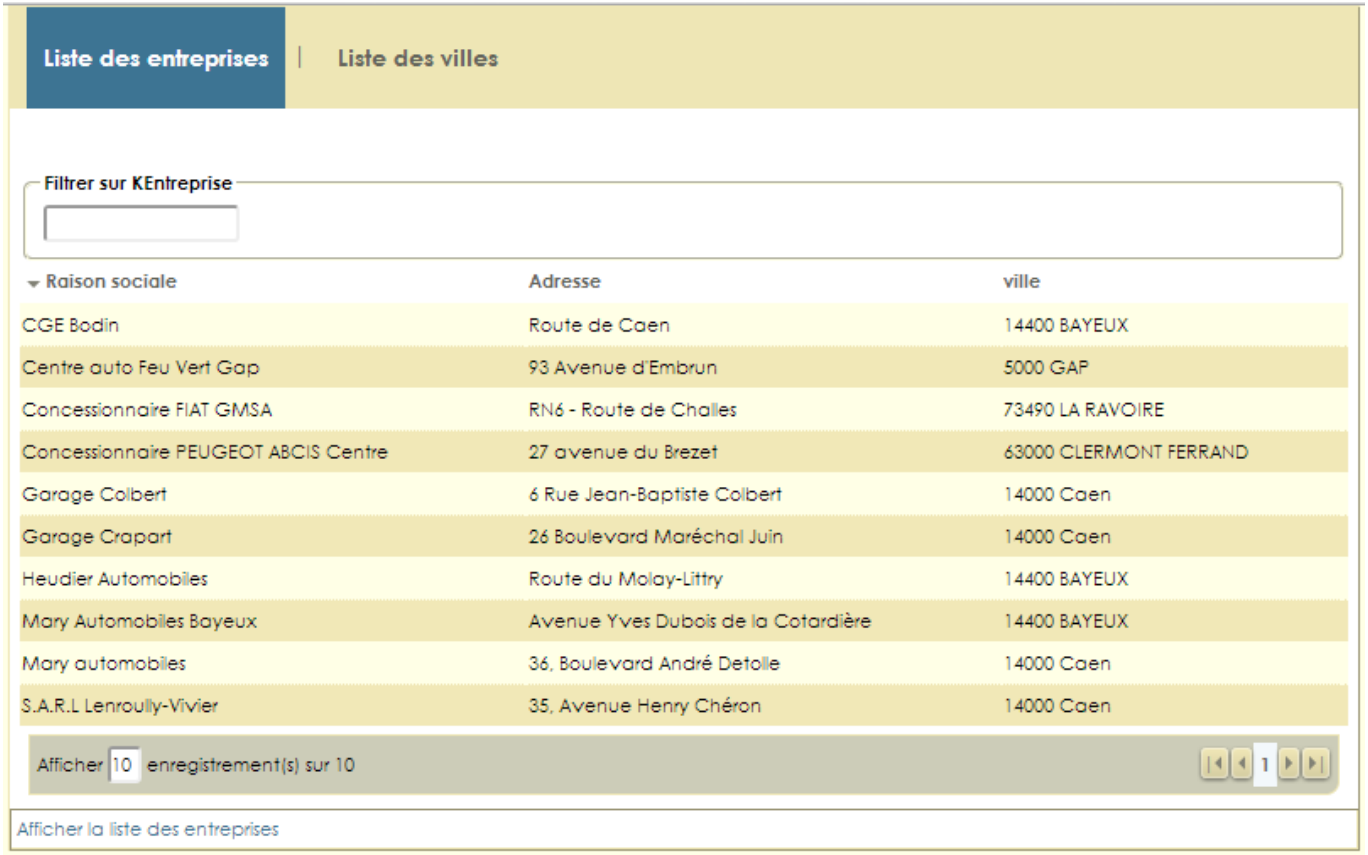
Nous allons maintenant créer nos pages à partir des templates : voir http://tutorial.kobject.net/java/listtemplate

Template de liste

- Créer l'arborescence des dossiers dans WEB-INF
  - dossier **forms** pour les formulaires
  - dossier **list** pour les listes
- Créer un template pour la classe **KEntreprise** à partir du menu **KObject/Kobject Templates**
- Insérer les éléments suivants en double-cliquant dans la barre d'outils

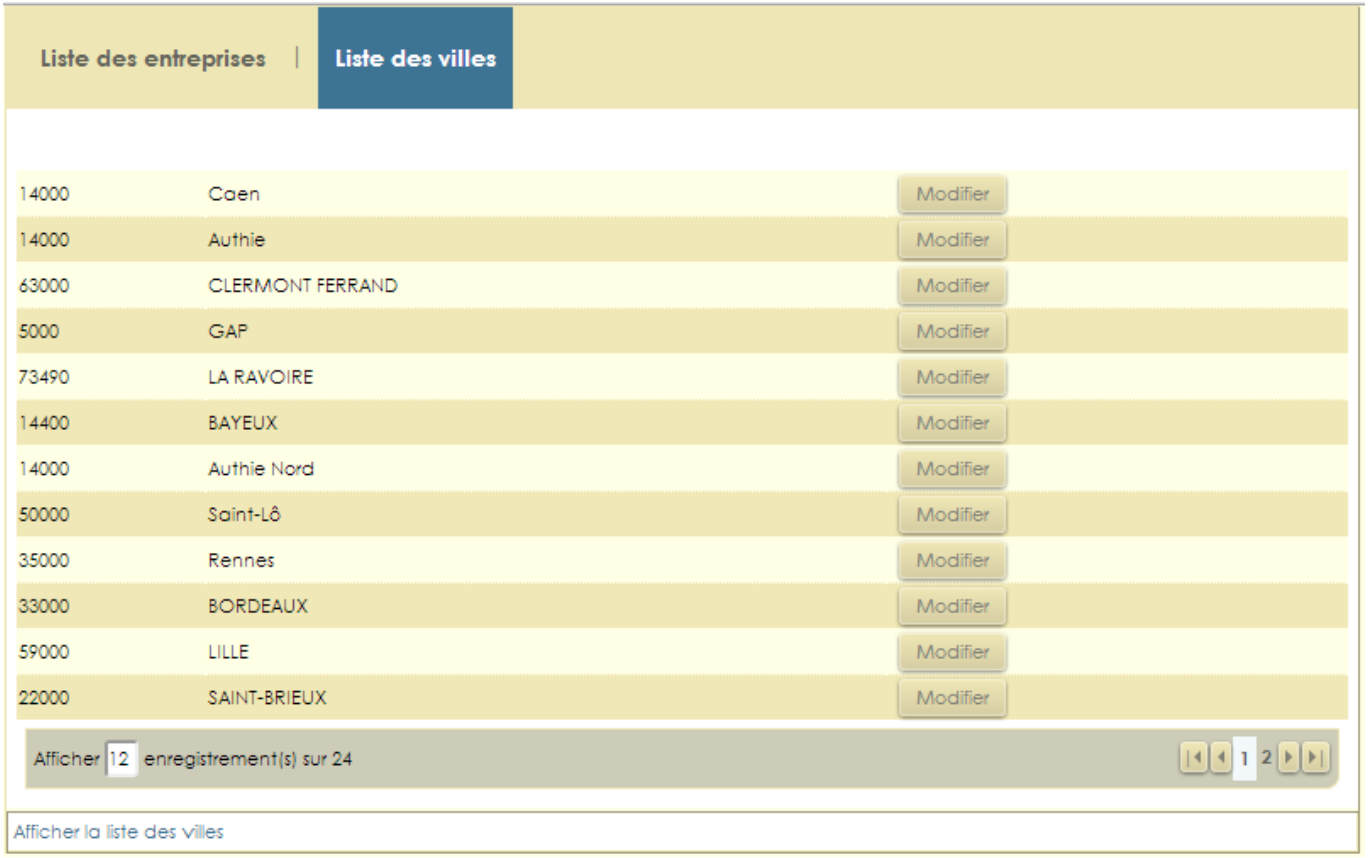



```
{#func:this.setEditable(false)#}
{#set:this.ajaxIncludes=true#}
{#set:this.isShowCaption=true#}
{#set:this.listContentUrl="entreprises.do"#}
{#mask:<td>{rs}</td><td>{adresse}</td><td>{ville}</td>#}
{#mask:<td>{rs}</td><td>{adresse}</td><td>{ville}</td>#}
{_ajx}
{_listContent}
    {_filtre}
    {_page}
    <div class="boxButtons">{_pageCounter}{_navBarre}</div>
{/_listContent}
```



- Modifier le mapping correspondant aux entreprises dans **mox.xml**
- Créer le template pour la liste des villes :

```
{#func:this.setEditable(true)#}
{#mask:<td>{cp}</td><td>{ville}</td>#}
{#mask:<td>{cp}</td><td>{ville}</td>#}
{#set:this.ajaxIncludes=true#}
{#set:this.listContentUrl="villes.do"#}
{_ajx}
{_listContent}
    {_page}
    <div class="boxButtons">{_pageCounter}{_navBarre}</div>
{/_listContent}
```



suite TD n°6 : Conception de logique applicative

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/javaee/td5

Last update: **2019/08/31 14:21**