Conception de logique applicative

Objectifs

- Concevoir des interfaces (vues)
- Créer leur logique comportementale

Situation initiale

- Créer le projet **koTd6** en important le fichier war dans eclipse.
- Démarrer le projet sur le serveur web (**Run as...**)

Conception des vues

Villes

Objectifs

Il s'agit de donner à la liste des villes le comportement suivant :

- Ajout de la sélection au clavier et à la souris
- Edition par défaut supprimée et remplacée par une édition directe (dans la liste)
- Ajout d'un bouton détail pour visualiser la liste des entreprises de la ville

- Ouvrir le template de la classe ville :

Ajout de la sélection au clavier

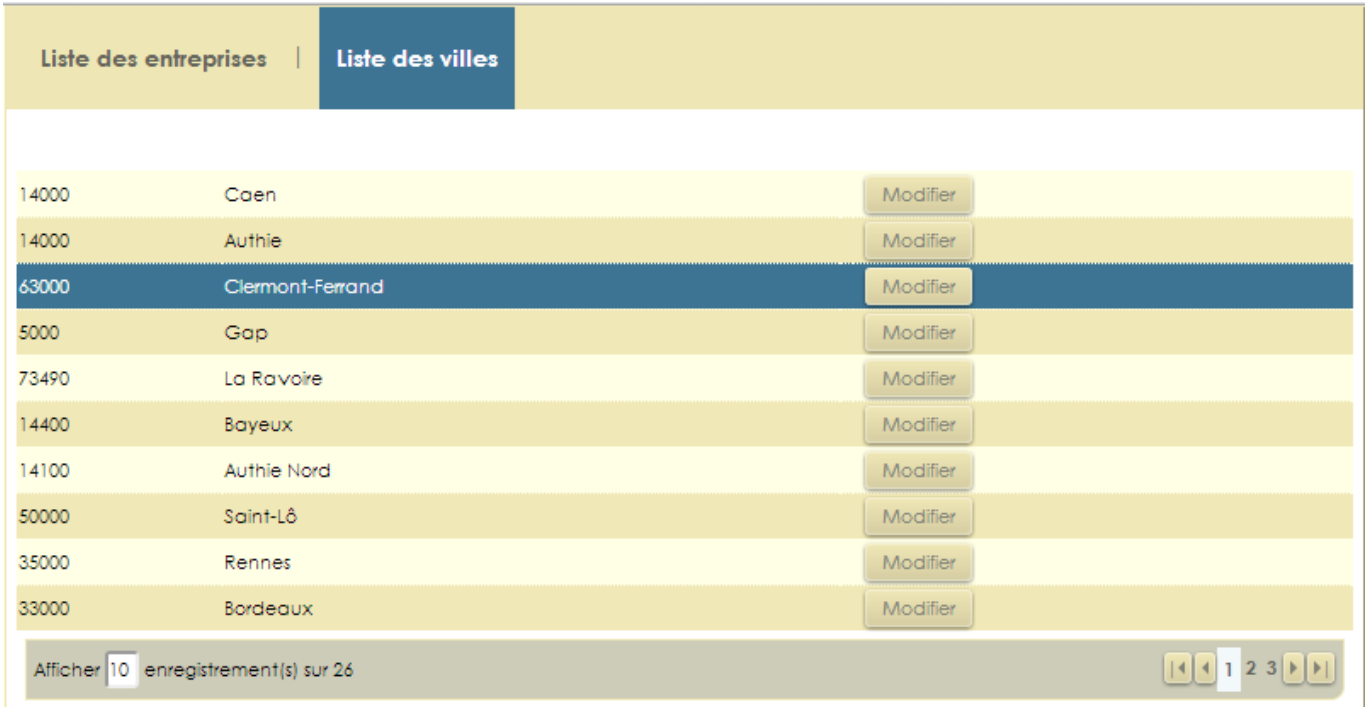
la méthode **addSelector** permet d'ajouter le contrôle clavier sur la liste : le code de touche **113** correspond à la touche F2, et permet l'édition d'une ville : voir http://tutorial.kobject.net/java/ajaxinclude/keyboard

- Modifier le template des villes :

```
{#func:this.addSelector(113)#}
{#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}
```

- Tester la page **villes.do** et son comportement (clavier et souris)

Action	Effet
F2	Edition avec le formulaire de modification
Touches de direction Haut, bas, gauche, droite	Déplacement entre les villes
MAJ+Home	Atteindre la première page
MAJ+Fin	Atteindre la dernière page
MAJ+PageUp	Page précédente
MAJ+PageDown	Page suivante
Double clic	Déplacement sélection ligne



- Enlever l'édition de la liste

```
{#func:this.addSelector(113)#}
{#func:this.setEditable(false)#}
...
```

Activation de l'édition directe des membres

Pour qu'un membre soit éditable, il faut qu'il appartienne à un élément DOM de la classe **editable**.

Nous allons ajouter un **Display** associé à la classe **KVille** pour modifier l'affichage des éléments de la liste :

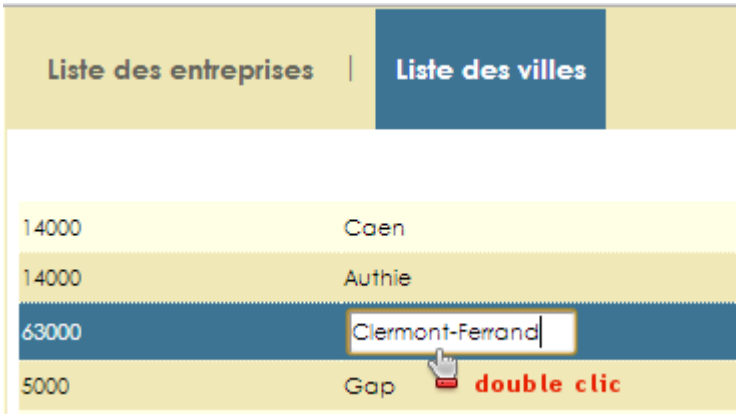
```
package net.display;
import net.ko.displays.KObjectDisplay;
import net.ko.kobject.KObject;


public class VilleDisplay extends KObjectDisplay {

    @Override
```

```
    public String showInList(KObject ko, String memberName) {
        String result=super.showInList(ko, memberName);
        //si le nom du membre est cp ou ville, on le met dans un span de classe css
editable
        if("cp".equals(memberName) || "ville".equals(memberName)){
            result="<span class='editable'
title='"+memberName+"'>"+result+"</span>";
        }
        return result;
    }
}
```

- Associer ensuite le **Display** créé à la classe **KVille** dans le fichier **/conf/kox.xml** :

```
...
    <class name="KVille" display="net.display.VilleDisplay">
        <member max="5" name="cp" required="1" type="string" />
        <member max="100" name="ville" required="1" type="string"
transform="onlyFirstWordUpper"/>
    </class>
...
```

- Tester l'édition possible de la liste :



Action	Effet
Double clic	Mode édition du membre
ESCAPE	Sortie de l'édition
ENTREE	Sortie avec possible validation
Perte du focus	Sortie de l'édition

A noter que l'édition n'est pour l'instant que visuelle, et ne permet pas la validation des modifications effectuées.

Ajout de la validation

- Editer le fichier conf/mox.xml et ajouter l'inclusion ajax suivante :
- Sur la page **villes.do**,
  - l'événement **updated** sur une zone de classe css **editable**,
    - provoque le post vers une url virtuelle de validation (**updateOne.do**) de l'instance de

KVille,
- et passe les informations contenues dans le champ édité (**target.innerHTML**) du champ (**target.title**)

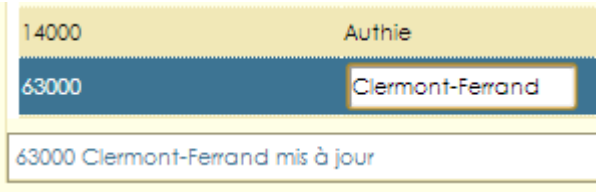
```
    <ajax-includes>
    ...
        <request requestURL="villes.do">
            <js triggerSelector=".editable" triggerEvent="updated">
                <updateOne virtualURL="updateOne.do" operation="update"
kobjectShortClassName="KVille" targetId="info" method="POST">
                    <field name="{js:target.title}" value="{js:target.innerHTML}"/>
                </updateOne>
            </js>
        </request>
    ...
    <ajax-includes>
```

- Ajouter dans le même fichier le mapping virtuel **updateOne.do**, associé à une action de type **updateOne** :

```
    <mappings>
        ...
        <virtualMapping requestURL="updateOne.do" mappingFor="updateOne"/>
        ...
    </mappings>
```

- tester l'édition finalisée dans la page **villes.do** :

Message après modification et validation par **ENTREE** :



Ajout d'un champ supplémentaire dans la liste

Le champ supplémentaire devra afficher le nombre actuel d'entreprise(s) de la ville, dans un lien cliquable qui devra ensuite afficher la liste des entreprises de la ville.

- Modifier la classe KVille pour qu'elle charge automatiquement les entreprises depuis la base de données :

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

- Ajouter l'élément btDetail dans le masque de la liste (il s'agit d'un membre inexistant de la classe)

```
{#func:this.setEditable(false)#}
{#func:this.addSelector(113)#}
{#mask:<td>{cp}</td><td>{ville}</td><td>{btDetail}</td>#}
{#mask:<td>{cp}</td><td>{ville}</td><td>{btDetail}</td>#}
{#set:this.ajaxIncludes=true#}
{#set:this.listContentUrl="villes.do"#}
{_ajx}
{_listContent}
    {_page}
        <div class="boxButtons">{_pageCounter}{_navBarre}</div>
{/_listContent}
```

* Modifier le Display de la classe **KVille**, VilleDisplay :

```
public class VilleDisplay extends KObjectDisplay {

    @Override
    public String showInList(KObject ko, String memberName) {
        String result=super.showInList(ko, memberName);

        if("cp".equals(memberName) || "ville".equals(memberName)){
            result="<span class='editable'
title='"+memberName+"'>"+result+"</span>";
        }
        if("btDetail".equals(memberName)){
            KVille ville=(KVille) ko;
            int nb=ville.getEntreprises().count();
            if(nb>0)
                result="<div><a id='a-"+ville.getId()+"'
class='default'>"+KString.pluriel(ville.getEntreprises().count(),"entreprise")+"</a
></div>";
            else
result="<div>"+KString.pluriel(ville.getEntreprises().count(),"entreprise")+"</div>
";
        }
        return result;
    }
}
```

Affichage des entreprises de la ville

- Ajouter une zone divEntreprises dans le template des villes :

```
{#func:this.setEditable(false)#}
{#func:this.addSelector(113)#}
{#mask:<td>{cp}</td><td>{ville}</td><td>{btDetail}</td>#}
{#mask:<td>{cp}</td><td>{ville}</td><td>{btDetail}</td>#}
```

```
{#set:this.ajaxIncludes=true#}
{#set:this.listContentUrl="villes.do"#}
{_ajx}
{_listContent}
    {_page}
        <div class="boxButtons">{_pageCounter}{_navBarre}</div>
{/_listContent}
<div id="divEntreprises"></div>
```

- Créer un nouveau template de type **list** pour afficher les entreprises de la ville.
- Associer le template à un Display **net.display.EntrepriseVille** (à créer par la suite)

```
{#koDisplay:net.display.EntrepriseVille#}
{#mask:<td>{rs}</td><td>{adresse}</td><td>{ville}</td><td>{tel}</td>#}
{_ajx}
{_listContent}
    {_page}
{/_listContent}
```

Entreprises

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

Permanent link:
http://slamwiki2.kobject.net/javaee/td6?rev=1386639515

Last update: **2019/08/31 14:42**