Exercices sur les arbres

Classification par catégorie

Auteur: E. Thirion

Dernière mise à jour : 02/07/23

Ce document fait partie d'un ensemble de cours du même auteur (programmation procédurale et objet, programmation web, bases de données) auxquels vous pouvez accédez en cliquant ici.

Objectif : réaliser un programme Javascript permettant de gérer un arbre de classification en différentes catégories. Ce programme est une extension de celui vu en cours (voir section Arbres).

Connaissance requises : bases de la programmation, programmation objet, pointeurs, listes, javascript, récursivité.

I- Représentation

La classe **Categorie** contient à présent un pointeur vers le noeud père:

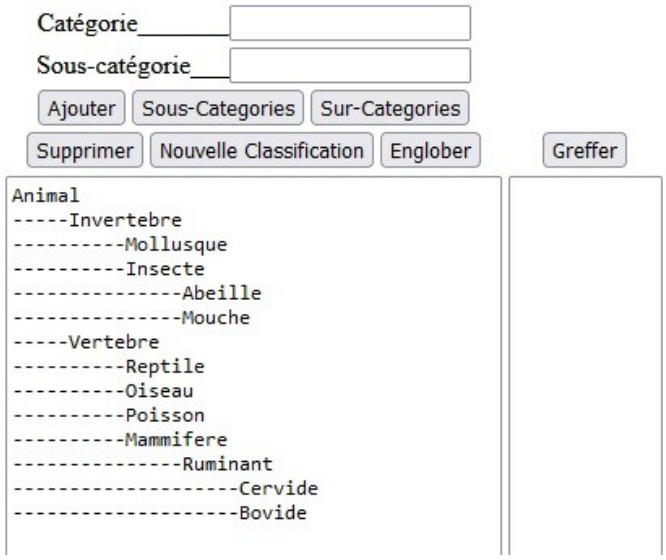
```
function Categorie (n,fs,fr,pr) {

   this.Nom = n;
   this.Fils = fs;
   this.Frere = fr;
   this.Pere = pr
   .
   .
```

La racine de l'arbre est stockée dans la variable globale **RACINE**. On conviendra que **RACINE == null** représente un arbre vide.

II – Interface graphique

L'interface graphique est définie par le fichier source **Classification2.html** (à ne pas modifier), qui fait appel au code javascript contenu dans **Classification2.js** (que vous aurez à complèter). Voici son apparence initiale :



L'interface graphique comporte :

- deux champs de texte libellés respectivement **Catégorie** et **Sous-catégorie**.
- Sept boutons
- Deux zones de texte. Celle de gauche sert à afficher l'arbre et celle de droite à afficher des listes de catégories contenues dans l'arbre.

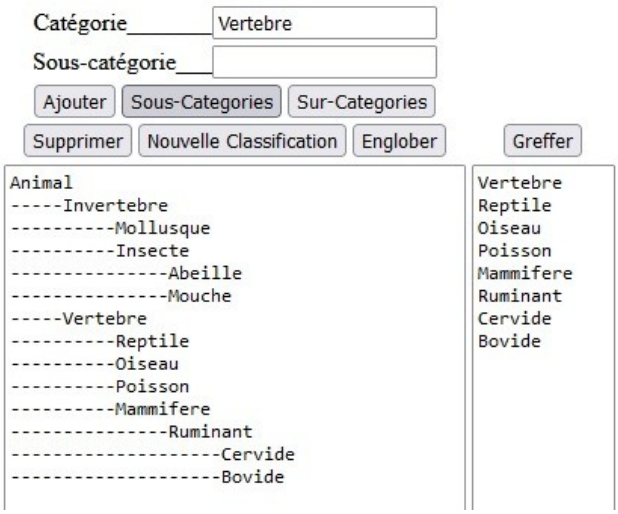
Le code des procédures évènementielles associées aux différents boutons est donné. Par contre, ce code utilise des fonctions que vous aurez à complèter (certaines méthodes de la classe **Categorie** ou d'autres fonctions).

L'arbre initial est créé par la procédure évènementielle **Initialiser** appelée au chargement de la page.

Les procédures évènementielles associées aux boutons affichent systématiquement l'arbre après chaque action effectuée par l'utilisateur par un appel de la fonction récursive **Texte_Arbre** dont le code est donné.

Pour l'instant, le seul bouton qui fonctionne est le bouton **Ajouter**. Vous pouvez donc ajouter de nouvelles catégories à l'arbre, comme expliqué dans le cours. Votre travail consistera à faire fonctionner les autres boutons.

III – Le bouton Sous-Categories



Les sous-catégories d'une catégorie donnée sont ses descendants.

Le bouton **Sous-Categories** sert à afficher les sous-catégories d'une catégorie dont le nom est donné par l'utilisateur. Les noms de ces sous-catégories s'affichent dans la zone de texte de droite.

Dans l'exemple de la figure ci-dessus, l'utilisateur a obtenu les sous-catégories de la catégorie **Vertebre**.

Pour faire fonctionner ce bouton, il vous faudra complèter la méthode **Sous_Categories**. Si **c** est un objet de la classe catégorie, **c. Sous_Categories** retourne toutes les sous-catégories de **c**.

Le résultat est retourné sous la forme d'un tableau de pointeurs (adresses des sous-catégories).

Pensez à utiliser la récursivité !

IV – Le bouton Sur-Categories

Les sur-catégories d'une catégorie sont ses noeuds ascendants, autrement dit, son père, le père de son père, etc ... jusqu'à la racine.

Elles doivent s'afficher dans la zone de texte de droite. Ci-dessous, les sur-catégories de la catégorie **Abeille**:

Pour faire fonctionner ce bouton, vous devez complèter la méthode **Sur_Categories**. Si **c** est un objet de la classe catégorie, **c. Sur_Categories** retourne toutes les sur-catégories de **c** dans un tableau de pointeurs (adresses des sur-catégories).

V – Le bouton Supprimer

Ce bouton supprime une catégorie donnée de l'arbre (et par conséquent tous ses descendants !).

Si dessous, résultat de la suppression de la catégorie **Insecte**:

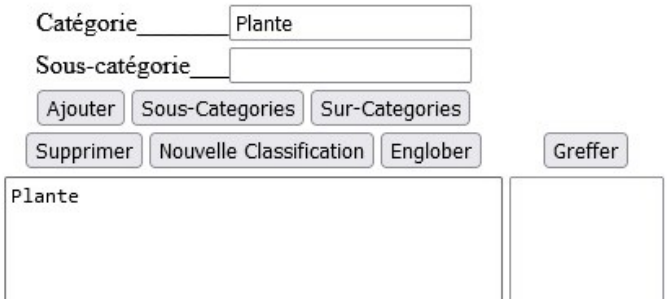
Remarquez que la suppression de la racine créé un arbre vide.

La méthode à complèter se nomme **Supprimer**.

VI – Nouvelle classification

Pour créer un arbre depuis le début, il faut commencer par créer la racine. C'est le rôle du bouton **Nouvelle Classification**.

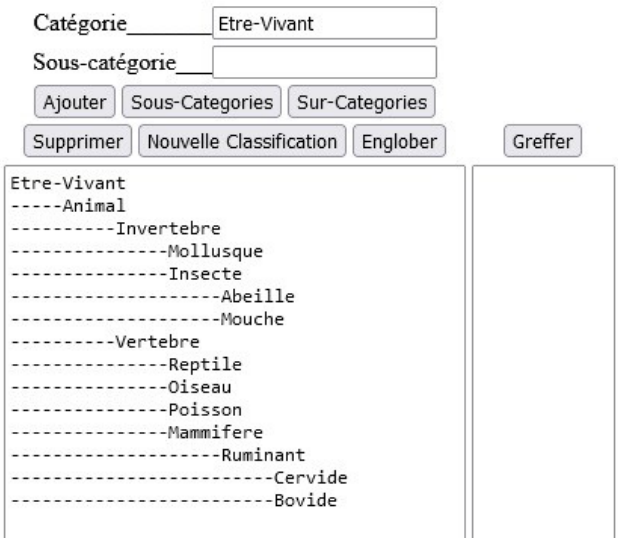
Par exemple, pour démarrer une classification des plantes:



Pour faire fonctionner ce bouton, complèter le code de la procédure **Creer_Nouvelle_Classification**. Elle prend en paramètre le nom de la catégorie racine.

VII – Le bouton Englober

En partant d'une classification donnée, on peut vouloir l'englober dans une classification plus grande. Par exemple, en partant d'une classification des animaux (**Animal**), créer une classification des êtres vivants (**Etre-Vivant**) . Cela signifie que **Etre-Vivant** devient la nouvelle racine de la classification et **Animal**, une de ses sous-catégories :



Pour faire fonctionner ce bouton, complèter le code de la procédure **Englober_Par**. Elle prend en paramètre le nom de la catégorie englobante

VIII – Le bouton Greffer

.
En jardinage, greffer signifie prendre une branche d'un arbre pour la mettre sur un autre arbre. Ici, nous allons faire la même chose, sauf que nous allons remettre la branche sur le même arbre.

Prenons un exemple. L'arbre de classification ci dessous contient une erreur:

```
Animal
-----Invertebre
----------Mollusque
----------Insecte
--------------Araignee
--------------------Tegenaire
--------------------Mygale
--------------Abeille
--------------Mouche
-----Vertebre
----------Reptile
          Oiseau
```

En effet, les araignées ne sont pas des insectes. La branche des araignées devrait donc être greffée sur les invertébrés. Avec notre logiciel de gestion, l'utilisateur réalise ceci à l'aide du bouton **Greffer**:

Pour faire fonctionner le bouton **Greffer**, il vous faudra complèter la méthode de même nom.

Si **c1** est une catégorie, **c1.Greffer(c2)** greffe le sous-arbre de racine **c2** à la catgérorie **c1**. C'est à dire que **c2** devient un des fils de **c1**. Attention: ici **c1** et **c2** sont des pointeurs sur des objets de la classe **Categorie** et non pas des noms de catégories.

Il y a plusieurs situations où greffer n'est pas possible ou n'a pas de sens. A vous de les détecter.

Pour que le bouton **Greffer** fonctionne correctement, la méthode **Greffer** doit retourner un booléen. Le résultat retourné sera **false** si la greffe n'est pas possible et **true** dans le cas contraire.