

Tutoriel pour le développement d'un client Swing avancé : menus, filtres, enregistrements, services Web REST...




Table des matières

- I. Introduction
 - I-A. À propos
 - I-B. Versions des logiciels et bibliothèques utilisés
 - I-C. Mises à jour
- II. Préparation
- III. Menus
- IV. Filtres et cache
- V. Poupes
- VI. Enregistrement
- VII. Web Services JSON
- VIII. Conclusion
- IX. Remerciements
- X. Annexes
 - X-A. Liens
 - X-B. Pour aller plus loin
 - X-C. Solution possible
 - X-D. Web services en PHP

Ce TP va être l'occasion de compléter l'application Java Swing et les DAO que vous aviez créés lors des séances précédentes. Vous allez ajouter des menus. Vous découvrirez des patterns d'action et de délégation. Et pour réussir, Google devra être votre ami. Commentez ★★★★★

Article lu -1 fois.

L'auteur

Thierry Leriche-Dessirier 

L'article

Publié le 27 décembre 2013 - Mis à jour le 28 décembre 2016

Liens sociaux



I. Introduction ▲

Dans ce TP, vous allez simplifier une IHM (Interface Homme Machine) à l'aide de menus. Vous utiliserez un cache en mémoire pour conserver des données, que vous filtrerez puis sauverez.

Ce TP est la suite directe des TP tp-chien-ihm et tp-chien-dao dans lesquels vous avez découvert (ou redécouvert) Maven, Eclipse, les logs, les tests, les fichiers CSV, Swing, JFreeChart et quelques bibliothèques utiles. Je vous conseille donc très vivement de le lire avant de poursuivre avec ce nouveau TP.

Demandez l'aide du professeur ou de son assistant si vous restez bloqué trop longtemps. La FAQ en annexe du TP tp-chien-dao contient les réponses qu'on me pose le plus souvent. Merci de consulter cette FAQ avant d'appeler le professeur.

Durant les deux TP précédents, vous étiez relativement guidés. Vous avez désormais toutes les cartes en main pour aborder ce troisième TP sur la base des résultats attendus, avec très peu d'aide. Une grande partie des informations ne vous sera pas donnée. Vous devrez donc faire des recherches sur Internet (par exemple dans Google) car ça fait aussi partie du métier de développeur. Mais rassurez-vous, je vais vous donner quelques liens utiles. Et vous trouverez des propositions de solution dans les fichiers .zip.

I-A. À propos ▲

Ce document est la retranscription d'un TP de Génie Logiciel que je donne à mes élèves de l'ESIEAESIEA. Je l'offre (gratuitement) à la communauté, en particulier aux autres professeurs qui peuvent le proposer à leurs propres étudiants.

En contrepartie, je vous demande simplement de ne pas modifier ce document. Au contraire, je vous invite à en communiquer directement l'adresse (sur Developpez.com) et de profiter ainsi des futures mises à jour. N'oubliez pas de me le faire savoir, le cas échéant. Par avance, je vous remercie pour votre participation.

I-B. Versions des logiciels et bibliothèques utilisés ▲

Pour écrire ce document, j'ai utilisé les versions suivantes :

• Java JDK 8

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [Fermer](#)

- Log4J 1.2.13 ;
- CSV Engine 1.3.5 ;
- JFreeChart 1.0.14 ;
- Guava 15.0 ;
- JAX-RS 2.0 ;
- Jersey 2.5 ;
- Jackson 1.9.11.

I-C. Mises à jour ▲

28 décembre 2013 : Création ;

2 avril 2014 : Ajout de mockups en remplacement de captures.

14 avril 2014 : Ajout d'une vidéo pour illustrer les premières étapes.

1er novembre 2016 : Remplacement de tout le chapitre dédié à l'import dans Eclipse et à la préparation Maven.

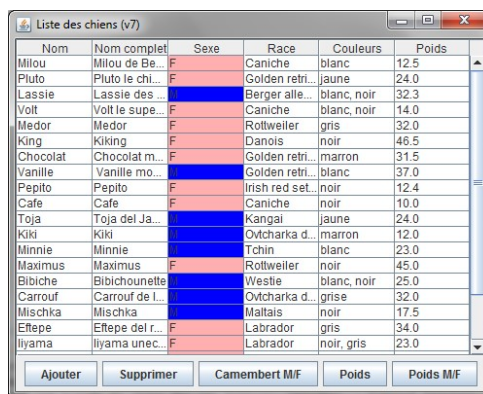
II. Préparation ▲

Durée estimée : 1 minute.

Pour commencer, s'il ne vous a pas été fourni par votre professeur, téléchargez le fichier « tp-chien-plus-1.zip » qui contient un projet Java Maven. Il s'agit simplement de la solution du précédent TP, un peu réorganisée.

Décompressez le fichier « tp-chien-plus-1.zip ».

Importez le projet dans Eclipse. Quand vous lancez l'application (classe « LauncherIHM.java »), cela devrait ressembler à la capture suivante :



Nom	Nom complet	Sexe	Race	Couleurs	Poids
Milou	Milou de Be...	F	Caniche	blanc	12.5
Pluto	Pluto le chi...	F	Golden retr...	jaune	24.0
Lassie	Lassie des ...	F	Berger alle...	blanc, noir	32.3
Volt	Volt le supe...	F	Caniche	blanc, noir	14.0
Medor	Medor	F	Rottweiler	gris	32.0
King	Kiking	F	Danois	noir	46.5
Chocolat	Chocolat m...	F	Golden retr...	marron	31.5
Vanille	Vanille mo...	F	Golden retr...	blanc	37.0
Pepito	Pepito	F	Irish red set...	noir	12.4
Cafe	Cafe	F	Caniche	noir	10.0
Toja	Toja del Ja...	F	Kangai	jaune	24.0
Kiki	Kiki	F	Ovtcharka d...	marron	12.0
Minnie	Minnie	F	Tchin	blanc	23.0
Maximus	Maximus	F	Rottweiler	noir	45.0
Bibiche	Bibichouette	F	Westie	blanc, noir	25.0
Carrouf	Carrouf de l...	F	Ovtcharka d...	grise	32.0
Mischka	Mischka	F	Maltais	noir	17.5
Eftepe	Eftepe del r...	F	Labrador	gris	34.0
Iiyama	Iiyama unec...	F	Labrador	noir, gris	23.0

Voici une vidéo qui montre comment on faisait cette étape dans la version précédente de ce document. On utilisait alors Maven en ligne de commande pour compiler et préparer les fichiers d'Eclipse. Prenez un peu de temps pour regarder la vidéo, au moins par curiosité ; elle est très courte. À la fin, il y a une petite démo de l'application qu'on a déjà développée.

Cliquez pour lire la vidéo

Les objets utilisés dans le projet sont ceux que vous avez déjà manipulés. Ils ne devraient donc pas vous poser de problèmes.

III. Menus ▲

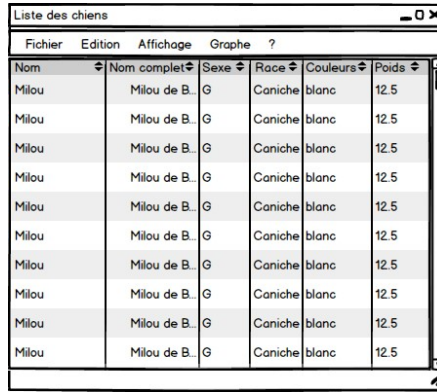
Comme vous pouvez le constater, la fenêtre est un peu chargée, surtout au niveau de la zone du bas qui déborde de boutons. La première chose que vous allez faire, c'est de supprimer la barre de boutons.

Ne l'effacez pas du programme. Contentez-vous d'en mettre le code en commentaire.

À la place des boutons, vous allez ajouter un ensemble de menus. Inspirez-vous du

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [Fermer](#)

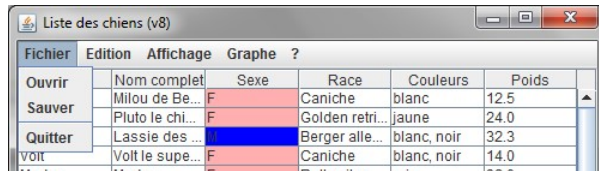
conseille la lecture d'un article de Baptiste Witch, intitulé « Swing : les bases » et plus particulièrement le chapitre « 9. ajoutez une barre de menu à votre fenêtre ». Dans cet article, regardez bien comment il lie les items de menus aux actions. Dans le TP précédent, nous avons utilisé la même technique pour lier les boutons aux actions. Vous allez donc pouvoir réutiliser très facilement le code préexistant.



Nom	Nom complet	Sexe	Race	Couleurs	Poids
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5
Milou	Milou de B.	G	Caniche	blanc	12.5

Une remarque à propos des menus, qui est valable pour la programmation de manière générale : allez-y progressivement. Commencez par coder un menu pour voir si ça marche. Codez seulement ensuite les autres menus. N'ayez pas peur de tout casser. Vous ne casserez rien. Au pire un [Ctrl] + [Z] remettra votre code comme avant.

Ajoutez des sous-menus en vous inspirant des cinq captures suivantes.



Nom complet	Sexe	Race	Couleurs	Poids
Milou de Be...	F	Caniche	blanc	12.5
Pluto le chi...	F	Golden retri...	jaune	24.0
Lassie des ...	F	Berger alle...	blanc, noir	32.3
Volt le supe...	F	Caniche	blanc, noir	14.0

Pour ajouter une ligne de séparation dans un menu, vous pouvez faire comme dans le bloc de code suivant :

Ligne de séparation
Sélectionnez

```
final JMenu menuFichier = new JMenu("Fichier");
...
menuFichier.addSeparator();
...
```

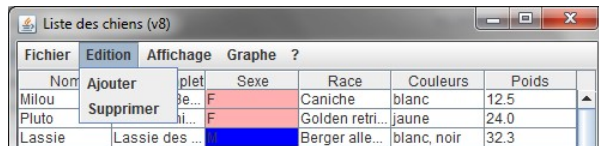
Je vais vous aider un peu pour le menu « Quitter ». Il n'est pas très compliqué, mais il faut savoir par quel bout le prendre. En fait, il suffit d'appeler « System.exit() ». Voici un exemple de la façon dont on peut le programmer :

Menu Quitter
Sélectionnez

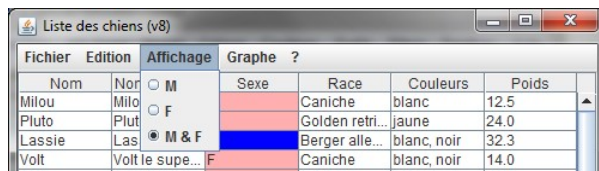
```
final JMenuItem menuQuitter = new JMenuItem(new QuitterAction("Quitter"));
...

private class QuitterAction extends AbstractAction {
    public QuitterAction(String texte) {
        super(texte);
    }

    public void actionPerformed(ActionEvent e) {
        LOGGER.info("Au revoir");
        System.exit(0);
    }
}
```

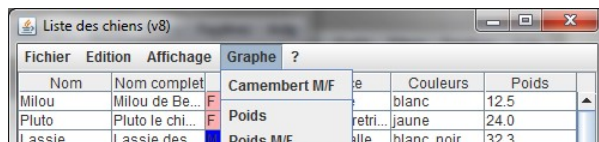


Nom	Nom complet	Sexe	Race	Couleurs	Poids
Milou	Milou de Be...	F	Caniche	blanc	12.5
Pluto	Pluto le chi...	F	Golden retri...	jaune	24.0
Lassie	Lassie des ...	F	Berger alle...	blanc, noir	32.3

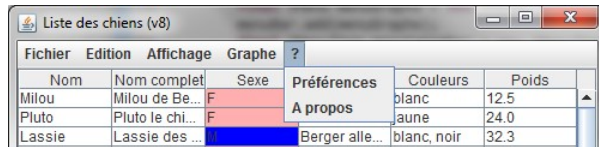


Nom	Nom complet	Sexe	Race	Couleurs	Poids
Milou	Milou de Be...	F	Caniche	blanc	12.5
Pluto	Pluto le chi...	F	Golden retri...	jaune	24.0
Lassie	Lassie des ...	F	Berger alle...	blanc, noir	32.3

N'oubliez pas que les boutons radio (à cocher en rond) fonctionnent ensemble. Lorsqu'on en coche un, ça décoche les autres. Pour faire cela, cherchez « ButtonGroup » sur Google.

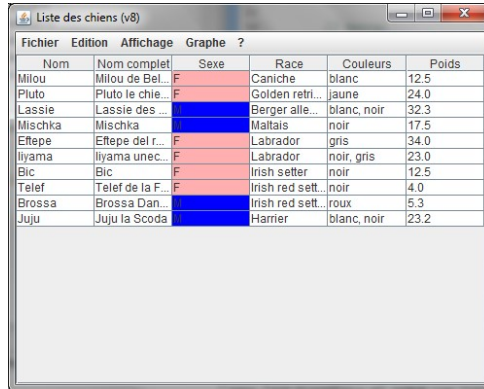


Nom	Nom complet	Sexe	Race	Couleurs	Poids
Milou	Milou de Be...	F	Caniche	blanc	12.5
Pluto	Pluto le chi...	F	Golden retri...	jaune	24.0
Lassie	Lassie des ...	F	Berger alle...	blanc, noir	32.3



Pensez à bien découper/organiser votre code pour qu'il soit lisible. Ne programmez pas tout dans le constructeur de votre fenêtre, mais faites plusieurs petites méthodes spécialisées.

A ce stade, si vous avez bien réutilisé les actions qui existaient déjà, une partie des menus devraient déjà fonctionner. C'est en particulier le cas du menu « Supprimer ». Pour le vérifier, sélectionnez plusieurs lignes puis activez ce menu. Ça devrait effacer des lignes du tableau.



Des lignes en moins

La génération des graphes devrait également fonctionner.

Une proposition de solution, à ce stade, est disponible dans le fichier « tp-chien-plus-2.zip ».

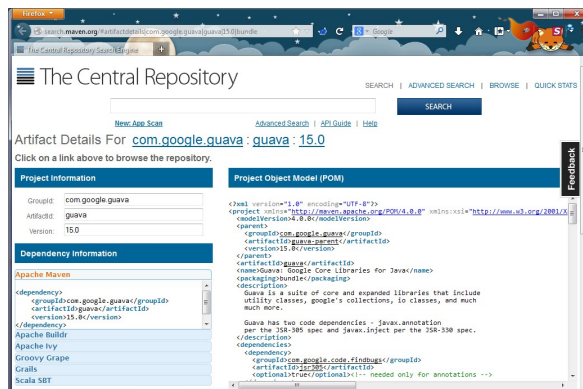
IV. Filtres et cache

Vous allez maintenant programmer la fonctionnalité qui permet de filtrer les chiens selon leur sexe. Le filtre sera simplement activé lors de la sélection d'un des trois menus de sexe. L'action (déclenchée par le menu) doit demander au modèle de filtrer sa liste de chiens.

Dans un premier temps, contentez-vous de faire les appels de méthodes nécessaires. On peut imaginer qu'on aura ajouté la méthode « rechercheAvecFiltreSurSexe() » au modèle et qu'on lui passe l'enum correspondant au sexe désiré en paramètre. Si on ne veut pas filtrer, on pourra passer la valeur « null ». Vous pouvez même faire en sorte que ce soit le service (i.e. ChienService.java) qui réalise le filtre. Le modèle délègue alors cette opération au service. Programmez des logs pour vérifier que tout fonctionne.

Dans le cadre du filtrage des chiens selon leur sexe, on peut dire que chaque élément de la liste est indépendant des autres. On peut donc employer une technique de programmation fonctionnelle pour filtrer la liste. Pour cela, vous allez ajouter une dépendance vers la bibliothèque Guava dans l'application.

Vous savez déjà comment ajouter une dépendance. Au moment où ce document a été écrit, la dernière version de Guava est la version « 15.0 ». Pour savoir ce qu'il faut indiquer précisément dans le fichier « pom.xml », vous pouvez lancer une recherche sur « Guava » sur le site « The Central Repository ».



Guava sur The Central Repository

Pour filtrer la liste, il faut appliquer un prédicat à chaque élément. Pour définir un tel prédicat, inspirez-vous de ce qui est présenté dans le tutoriel « introduction à la programmation fonctionnelle avec Guava ».

Il faut ensuite faire appel à la méthode « filter() » :

Filtre
Sélectionnez

```
final List<Chien> temp = findAllChiens(fileName);
```

que filtrer les données déjà présentes dans le modèle alors il vaut mieux rester au niveau du modèle.

Pendant que vous y êtes, vous allez vous occuper de la méthode « `findAllChiens()` » qui est lente. En effet, celle-ci recharge les données depuis le fichier CSV à chaque fois. Vous allez donc mettre en place un cache en mémoire. Pour cela, vous allez de nouveau utiliser Guava, dont c'est l'une des nombreuses fonctions.

Comme c'est un peu délicat, je vous donne directement le code pour définir le cache (par contre je vous laisse décider où le mettre) :

Cache

Sélectionnez

```
private LoadingCache<String, List<Chien>> chiensCache;
...

final CacheLoader<String, List<Chien>> cacheLoader = new CacheLoader<String, List<Ch
    public List<Chien> load(final String key) {
        // Ici la key ne sert à rien
        LOGGER.debug("Mise en cache des donnees.");
        final File file = new File(fileName);
        csvChienDao.init(file);
        return csvChienDao.findAllChiens();
    }
};

chiensCache = CacheBuilder.newBuilder() //
    .expireAfterWrite(10, TimeUnit.MINUTES) // Duree : 10 min.
    .build(cacheLoader);
```

Pour utiliser le cache, il faut faire appel à la méthode « `get()` » :

Demande la valeur

Sélectionnez

```
chiensCache.get("chiens");
```

Lorsqu'on demande une valeur dans le cache, celui-ci la renvoie s'il la possède. Dans le cas contraire, le cache va lancer la méthode « `load()` » qui ira chercher les données dans le fichier CSV. Le cache lancera alors une exception s'il ne trouve rien. Vous devez donc gérer ce cas dans votre code.

Une proposition de solution, à ce stade, est disponible dans le fichier « `tp-chien-plus-3.zip` ».

V. Pupups▲

Jusqu'à présent, lorsqu'on clique sur le bouton/menu « Ajouter », ça ajoute tout le temps le même chien. C'est normal puisqu'on a « codé en dur » la fonctionnalité :

Ajout en dur

Sélectionnez

```
final Chien idefix = new SimpleChien("Idefix", "Idefix", MALE, LABRADOR, new String[
modele.ajouterChien(idefix);
```

À la place, vous allez programmer une popup dans laquelle on pourra saisir les champs nécessaires à la création d'un nouveau chien. Cette fois, utilisez une `JDialog` afin de pouvoir facilement rendre la fenêtre modale.

Pour l'organisation des champs dans la fenêtre, vous pouvez utiliser plusieurs types de layouts. Consultez le guide visuel des layouts pour choisir celui qui vous semblera le plus adapté. Vous pouvez aussi préférer utiliser un layout nul qui vous permettra de positionner vos composants au pixel près à l'aide de la méthode « `setBounds()` ». C'est justement ce que j'ai fait pour cette capture. Pour la partie du haut (CENTER), contenant le formulaire, j'ai utilisé un panel avec un layout nul. Pour la partie du bas (SOUTH), contenant les boutons, j'ai utilisé le layout par défaut.

Le plus simple, pour alimenter les items des « `ComboBox` » (liste de sélection), est d'ajouter directement les enum de sexe et de race. Pour mettre en forme le texte, il suffit ensuite d'utiliser un « `DefaultListCellRenderer` », comme on l'avait fait pour le tableau à l'aide de « `DefaultTableCellRenderer` ».

Maintenant que votre popup est prête, vous pouvez vous occuper des boutons. Le plus simple est le bouton « Annuler » qui doit se contenter de fermer la popup, ce qu'on pourra faire à l'aide de la méthode « `dispose()` ».

Pour le bouton « Ajouter », c'est un peu plus délicat. Idéalement, il faudrait que la popup ait accès au modèle, mais ce serait un « contresens ». Ce que vous allez faire, à la place, c'est de passer un « handler » qui sera créé par la classe appelante et qui, lui, aura accès au modèle. Bien entendu, on ne peut/doit pas passer une classe spécifique à la popup mais une interface assez générique pour pouvoir être employée dans plusieurs contextes. Voici un exemple de ce qu'on peut écrire :

```

public interface ActionHandler {

    void process(final Action action) throws ChienException;

    void process(final Action action, final Chien chien) throws ChienException;

}

```

Ici, l'enum « Action » est un objet de confort qui n'est pas indispensable. Ce sera toutefois bien pratique lorsqu'on voudra ajouter d'autres boutons dans la popup.

Action
Sélectionnez

```

public enum Action {

    CREER,
    MODIFIER,
    ANNULER,
    SUPPRIMER,
    OUVRIR;

}

```

Le bouton de la popup pourra donc appeler le handler :

Action dans le bouton
Sélectionnez

```

private class AjouterAction extends AbstractAction {
    public AjouterAction(String texte) {
        super(texte);
    }

    public void actionPerformed(ActionEvent event) {
        LOGGER.debug("Ajouter");

        final SimpleChien nouveauChien = new SimpleChien();
        remplirChien(nouveauChien);

        try {
            actionHandler.process(Action.CREER, nouveauChien);
        } catch (ChienException exception) {
            // TODO
        }

        // Fermeture de la fenêtre
        ...
    }
}

```

Il s'agit ici de l'action « ajouter » associée au bouton dans la popup et non de l'action associée au menu dans la fenêtre principale. Si vous avez peur de vous embrouiller avec des noms de classes similaires, vous pouvez nommer différemment les classes internes d'action, par exemple « AjouterPopupAction ».

On pourra imaginer de nombreuses implémentations. En voici un exemple simple :

Implémentation simple
Sélectionnez

```

public class AjouterModifierChienActionHandler implements ActionHandler {

    private ModeleDynamique2 modele;

    public AjouterModifierChienActionHandler(ModeleDynamique2 modele) {
        this.modele = modele;
    }

    @Override
    public void process(final Action action) throws ChienException {
        LOGGER.debug("process");
        process(action, null);
    }

    @Override
    public void process(final Action action, final Chien chien) throws ChienException {
        LOGGER.debug("process");

        switch (action) {
            case ANNULER:
                annuler();
                break;

            case MODIFIER:
                modifier(chien);
                break;

            case CREER:
                ajouter(chien);
                break;

            case SUPPRIMER:
                supprimer(chien);
                break;

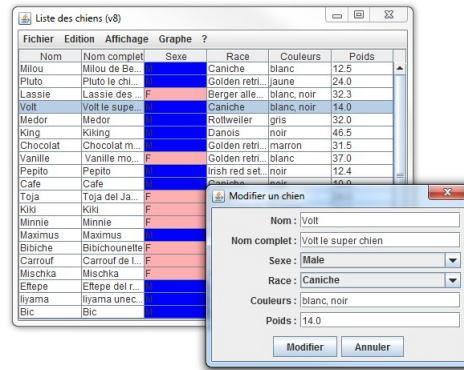
            default:
                throw new IllegalArgumentException("L'action demandee n'est pas disp
        }
    }

    ...
}

```

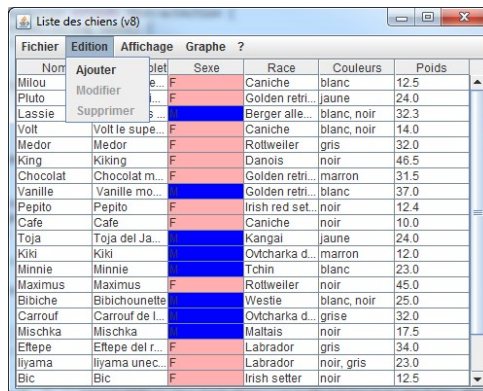
Vous remarquez que j'ai corrigé un bug avant de faire cette capture d'écran. En effet, les filles étaient reconnues comme des mâles et les garçons comme des femelles. Ce bug venait de la méthode « valueOfByCode() » de l'enum « Sexe » qui mélangeait des codes (1 ou 2) des genres. Mais j'imagine que vous aviez corrigé par vous-même...

Essayez de bien comprendre comment le handler fonctionne et comment on peut l'utiliser dans différents contextes avant de passer à la suite.

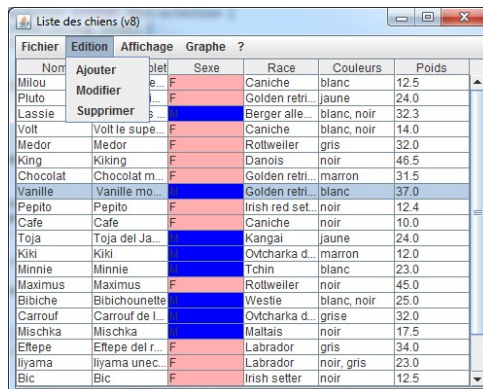


Modification du chien sélectionné dans la liste

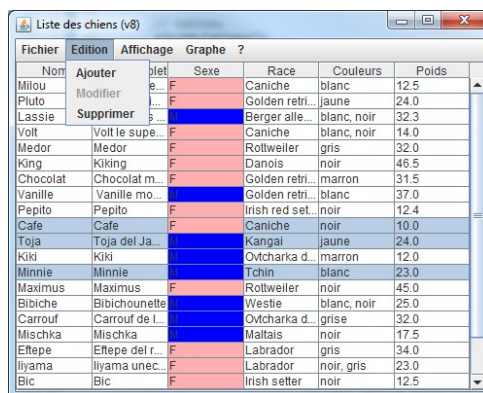
Pendant que vous y êtes, vous pouvez vous intéresser au menu de la fenêtre principale. Idéalement, il faudrait que le menu « Supprimer » ne soit activé (enabled) que lorsqu'au moins une ligne du tableau des chiens est sélectionnée. Il faut faire de même pour le menu « Modifier » à la différence qu'il ne doit être activé que lorsqu'une ligne et une seule est sélectionnée.



Aucune ligne sélectionnée



Une ligne sélectionnée



Plusieurs lignes sélectionnées

Pour savoir si une ou plusieurs lignes sont sélectionnées, il faut utiliser le modèle de sélection du tableau auquel on passe un écouteur qui fera le travail :

Modèle de sélection
Sélectionnez

```
final ListSelectionModel listSelectionModel = tableau.getSelectionModel();  
listSelectionModel.addListSelectionListener( ... );
```

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [Fermer](#)

Sélectionnez

```
private class TableauListSelectionListener implements ListSelectionListener {

    @Override
    public void valueChanged(final ListSelectionEvent event) {
        if (event.getValueIsAdjusting()) {
            return;
        }

        activerOuDesactiverMenuEdition();
    }
}
```

Dans ce bloc de code, l'instruction « if » évite de recevoir les événements en double.

Une proposition de solution, à ce stade, est disponible dans le fichier « tp-chien-plus-5.zip ».

VI. Enregistrement ▲

Il est temps de finaliser les actions correspondant aux autres menus. Commencez par le menu « Sauver » qui est un des plus importants. Ajoutez donc une méthode nommée « sauver » dans le DAO. Vous aurez besoin de l'ajouter également à l'interface.

Pour programmer l'enregistrement en Java pur, vous aurez besoin des classes « FileWriter » et « BufferedWriter ». Inspirez-vous de ce que vous aviez fait lors du premier TP. Le fonctionnement de ces classes est similaire à « FileReader » et « BufferedReader ».

Bien entendu, vous devez écrire des tests unitaires (JUnit) avant de coder la fonctionnalité d'enregistrement. Cela s'inscrit dans une démarche qualité. Souvenez-vous que vous aviez déjà pratiqué les TDD/3T dans les TP précédents.

Dans un premier temps, je vous conseille de sauver le fichier CSV sous un autre nom que celui du fichier que vous aviez lu au départ, pour ne pas perdre vos données.

Complétez également le DAO utilisant CSV Engine. Le site de la bibliothèque n'est pas très bavard sur les aspects liés à l'enregistrement. Je vous donne donc la partie importante du code. À vous de savoir comment l'intégrer :

Enregistrement avec CSV Engine

Sélectionnez

```
private void doSauver(final List<SimpleChien> chiens, final File file) {
    if (LOGGER.isDebugEnabled()) {
        LOGGER.debug("Sauvegarde des " + chiens.size() + " chiens dans le fichier "
    }

    final CsvEngine engine = new CsvEngine(SimpleChien.class);

    try {
        engine.writeFile(new FileWriter(file), chiens, SimpleChien.class);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

Une proposition de solution, à ce stade, est disponible dans le fichier « tp-chien-plus-6.zip ».

Programmez maintenant les autres menus, qui devraient être plus simples.

VII. Web Services JSON ▲

Notre application ne gère que des chiens. C'est un peu léger pour une animalerie. Vous allez donc vous associer à la boutique « Animal Mignon » qui propose d'autres animaux sur Internet.

Pour connaître le stock de la boutique, vous allez utiliser un web service. Celui-ci répond au format JSON (JavaScript Object Notation), qui a la particularité d'être très léger, mais un peu difficile à lire.

Pour cela, il suffit d'appeler une URL. Pour avoir des détails sur les chats mâles, il faudra demander <http://icauda.com/animal-mignon/selection.php?espece=chat&sexe=m> et on aura la réponse suivante :

Chats mâles

Sélectionnez

```
{
  "sexe": "m",
  "espece": "chat",
  "animaux": [
    {
      "nom": "Garfield",
      "sexe": "m",
      "poids": 12.5,
      "couleurs": "noir, orange",
      "prix": 150
    },
    {
      "nom": "Minou",
      "sexe": "m",
      "poids": 8.5,
      "couleurs": "noir",
      "prix": 150
    },
    {
      "nom": "Ronron",
      "sexe": "m",
      "poids": 8,
      "couleurs": "noir",
      "prix": 250
    }
  ],
  "nombre": 3
}
```


sur le résultat.

La technologie des web services ne dépend pas d'un langage en particulier. Celui-ci a été écrit en PHP. Vous en retrouverez le code en annexe...

On va maintenant essayer d'utiliser ce Web service. Le code est un peu chaud...

Commencez par ajouter des dépendances à l'API Java dédiée au Web service et au client Jersey :

Ajout des dépendances

Sélectionnez

```
<dependency>
  <groupId>javax.ws.rs</groupId>
  <artifactId>javax.ws.rs-api</artifactId>
  <version>2.0</version>
</dependency>

<dependency>
  <groupId>org.glassfish.jersey.core</groupId>
  <artifactId>jersey-client</artifactId>
  <version>2.5</version>
</dependency>
```

Jersey est l'implémentation de référence de JaxRS...

Commencez avec le code suivant :

Appel d'un web service sans paramètre

Sélectionnez

```
String responseEntity = ClientBuilder.newClient()//
    .target("http://www.icauda.com/animal-mignon/selection.php")
    .request()
    .get(String.class);

System.out.println("responseEntity: " + responseEntity);
```

Pour l'instant, cela répond, mais vous donne un message d'erreur, car il manque les paramètres :

Sélectionnez

```
responseEntity: {"message":"Le site ne connait pas cette espece."}
```

À vous de trouver comment passer les paramètres « espece » et « sexe » afin d'avoir une réponse équivalente à celle qui est indiquée plus haut. Pour cela, je vous invite à consulter la documentation du client de Jersey. Notez au passage que le paramètre « sexe » est optionnel.

Dans un premier temps, on s'est contenté de lire la réponse et de l'écrire dans la console. Vous allez maintenant la traiter. Vous remarquez que la réponse JSON est composée de deux « parties ». Dans la première il y a un ensemble d'informations (ici c'est juste la recopie des paramètres d'appel) et, dans la seconde partie, il y a la liste des chats trouvés.

Voici une proposition pour l'objet « Chat ». Je vous laisse écrire vous-même les classes pour les autres animaux.

Chat.java

Sélectionnez

```
public class Chat {
    private String nom;
    private Sexe sexe;
    private String couleurs;
    private double poids;
    private double prix;

    // getters et setters
```

À ce stade, ce qu'on voudrait, c'est transformer la réponse en objet. Pour cela, commencer par créer un « bean » (c'est-à-dire un objet) « ResultChat » contenant les attributs nécessaires, pour coller à peu près au format de la réponse.

Objet pour injecter la réponse

Sélectionnez

```
public class ResultChat {

    private String sexe;
    private String espece;
    private int nombre;
    private List<Chat> animaux;

    public ResultChat() {
    }

    // + getters et setters
```

Là, l'idée, c'est vraiment de reprendre les mêmes noms de champs pour que ce soit plus simple.

La bibliothèque la plus connue pour travailler avec le format JSON en Java est Jackson.

Ajoutez une dépendance à votre pom.xml pour l'utiliser dans votre projet :

Dépendance vers Jackson

Sélectionnez

```
<dependency>
  <groupId>org.codehaus.jackson</groupId>
  <artifactId>jackson-mapper-asl</artifactId>
  <version>1.9.11</version>
</dependency>
```

Sexe.java

Sélectionnez

```
public enum Sexe {

    FEMALE(2, "f"),
    MALE(1, "m");

    private final int code;
    private final String lettre;

    Sexe(final int code, final String lettre) {
        this.code = code;
        this.lettre = lettre;
    }

    public static Sexe valueOfByCode(final int code) {
        switch (code) {
            case 1:
                return MALE;

            case 2:
                return FEMALE;

            default:
                throw new IllegalArgumentException("Le sexe demande n'existe pas.");
        }
    }

    public static Sexe valueOfByLettre(final String lettre) {
        for(Sexe sexe : values()) {
            if (sexe.lettre.equals(lettre)) {
                return sexe;
            }
        }

        throw new IllegalArgumentException("Le sexe demande n'existe pas.");
    }

    public boolean isMale() {
        return this == MALE;
    }

    public int getCode() {
        return code;
    }
}
```

Pour savoir comment faire, je vous recommande la lecture des trois articles suivants :

- [Deserialize JSON with Jackson into Polymorphic Types](#)
- [How To Convert Java Object To/From JSON \(Jackson\)](#)
- [How Do I Write a Jackson JSON Serializer and Deserializer?](#)

Une proposition de solution, à ce stade, est disponible dans le fichier « tp-chien-plus-7.zip ».

Pour connaître la liste des autres espèces d'animaux disponibles, on peut interroger un autre Web service à l'adresse <http://icauda.com/animal-mignon/dispo.php>.

dispo

Sélectionnez

```
{
  "stock": {
    "chat": 10,
    "lapin": 4,
    "souris": 6
  }
}
```

Si vous avez réussi à « mapper » le premier appel de Web service, celui-ci devrait être une formalité pour vous. Ce qui serait bien, c'est de faire évoluer l'IHM pour indiquer le nombre d'animaux disponibles dans chaque espèce. Mais, surtout, ce second appel vous permet de savoir comment alimenter le paramètre « espece » du premier Web service...

VIII. Conclusion ▲

Essayez maintenant d'afficher des tableaux (dans l'IHM) contenant un mélange de plusieurs espèces. Pour cela, il vous faudra faire évoluer le modèle pour que les chiens, les chats, les souris et les lapins soient tous compatibles avec un type commun que vous créerez pour l'occasion (par exemple l'interface « Animal »).

Essayer aussi de sauvegarder les animaux dans des fichiers CSV. Vous pouvez sauvegarder chaque espèce dans des fichiers séparés, ou toutes dans le même fichier. Cette seconde option est plus difficile, mais aussi plus intéressante.

À la fin de ce TP, on a une application complète, mettant en œuvre un ensemble de bonnes pratiques. Bravo. On pourrait toutefois ajouter de nombreux éléments et options.

Vos retours nous aident à améliorer nos publications. N'hésitez donc pas à commenter cet article sur le forum : Commentez ★★★★★

Ce TP est utilisé par de nombreux étudiants. N'hésitez donc pas à m'envoyer directement des idées d'amélioration, des propositions d'exercices supplémentaires, des pistes intéressantes, etc.

IX. Remerciements ▲

J'adresse mes remerciements à tous ceux qui ont participé à l'écriture et à l'amélioration de cet article. Cela inclut l'équipe de Developpez.com ainsi que les professeurs des autres écoles et les élèves qui ont souffert en suivant les étapes décrites dans ce document.

Plus particulièrement j'adresse mes remerciements à Mickael BARON, Yann Caron, Logan et Philippe DUVAL.



X. Annexes▲

X-A. Liens▲

Doc de la partie Client de Jersey :

<https://jersey.java.net/documentation/latest/client.html>

How To Convert Java Object To/From JSON :

<http://www.mkyong.com/java/how-to-convert-java-object-to-from-json-jackson/>

X-B. Pour aller plus loin▲

X-C. Solution possible▲

ChienJFrame8.java

Sélectionnez

```
package com.icauda.tp.chien.ihm;

import static java.awt.BorderLayout.CENTER;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.swing.AbstractAction;
import javax.swing.ButtonGroup;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.ListSelectionModel;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.TableModel;
import javax.swing.table.TableRowSorter;

import org.apache.log4j.Logger;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.DefaultPieDataset;

import com.icauda.tp.chien.domain.Chien;
import com.icauda.tp.chien.domain.RaceDeChien;
import com.icauda.tp.chien.domain.Sexe;

public class ChienJFrame8 extends JFrame {

    /**
     * serialVersionUID
     */
    private static final long serialVersionUID = -638731145561555723L;

    private static final Logger LOGGER = Logger.getLogger(ChienJFrame8.class);

    private JTable tableau;
    private ModeleDynamique2 modele;

    private JDialog ratioMaleFemaleJdialog;
    private JDialog poidsJdialog;
    private JDialog poids2Jdialog;

    // Menus
    private JMenuItem menuModifier;
    private JMenuItem menuSupprimer;

    public ChienJFrame8() {
        super();
        setTitle("Liste des chiens (v8)");
        setPreferredSize(new Dimension(500, 400));
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // tableau
        ajoutDuTableau();

        // Menu
        ajoutDuMenu();

        pack();
    }

    private void ajoutDuMenu() {
        final JMenuBar menuBar = new JMenuBar();

        // Menu Fichier
        final JMenu menuFichier = new JMenu("Fichier");
        menuBar.add(menuFichier);

        // Sous-menus
        final JMenuItem menuOuvrir = new JMenuItem(new OuvrirAction("Ouvrir"));
```

```

menuFichier.addSeparator();
final JMenuItem menuQuitter = new JMenuItem(new QuitterAction("Quitter"));
menuFichier.add(menuQuitter);

// Menu Edition
final JMenu menuEdition = new JMenu("Edition");
menuBar.add(menuEdition);
final JMenuItem menuAjouter = new JMenuItem(new AjouterLigneAction());
menuEdition.add(menuAjouter);
menuModifier = new JMenuItem(new ModifierLigneAction());
menuEdition.add(menuModifier);
menuSupprimer = new JMenuItem(new SupprimerLigneAction());
menuEdition.add(menuSupprimer);
activerOuDesactiverMenuEdition();

// Menu Affichage
final JMenu menuAffichage = new JMenu("Affichage");
menuBar.add(menuAffichage);

// Genres
final JRadioButtonMenuItem menuMale = new JRadioButtonMenuItem(new MaleActio
menuAffichage.add(menuMale);
final JRadioButtonMenuItem menuFemale = new JRadioButtonMenuItem(new FemaleA
menuAffichage.add(menuFemale);
final JRadioButtonMenuItem menuMaleAndFemale = new JRadioButtonMenuItem(new
menuAffichage.add(menuMaleAndFemale);

// Group des genres
final ButtonGroup groupGenre = new ButtonGroup();
groupGenre.add(menuMale);
groupGenre.add(menuFemale);
groupGenre.add(menuMaleAndFemale);
menuMaleAndFemale.setSelected(true);

// Menu Graphe
final JMenu menuGraphe = new JMenu("Graphe");
menuBar.add(menuGraphe);
final JMenuItem menuCamember = new JMenuItem(new CamembertMaleFemaleAction()
menuGraphe.add(menuCamember);
menuGraphe.addSeparator();
final JMenuItem menuPoids = new JMenuItem(new PoidsAction());
menuGraphe.add(menuPoids);
final JMenuItem menuPoids2 = new JMenuItem(new Poids2Action());
menuGraphe.add(menuPoids2);

// Menu ?
final JMenu menuPointInterrogration = new JMenu("?");
menuBar.add(menuPointInterrogration);
final JMenuItem menuPreferences = new JMenuItem(new PreferencesAction("PrÃ©f
menuPointInterrogration.add(menuPreferences);
final JMenuItem menuAPropos = new JMenuItem(new PreferencesAction("A propos"
menuPointInterrogration.add(menuAPropos);

// Ajout a la fenetre
setJMenuBar(menuBar);
}

private void ajoutDuTableau() {
modele = new ModeleDynamique2();
tableau = new JTable(modele);
getContentPane().add(new JScrollPane(tableau), CENTER);

tableau.setDefaultRenderer(Sexe.class, new SexeCellRenderer());
tableau.setDefaultRenderer(RaceDeChien.class, new RaceCellRenderer());

tableau.getColumnModel().getColumn(4).setCellRenderer(new ListeCouleursCellR

tableau.setAutoCreateRowSorter(true);

final TableRowSorter<TableModel> sorter = new TableRowSorter<TableModel>(tab
tableau.setRowSorter(sorter);

// colonne Nom non triable
sorter.setSortTable(0, false);
// Tri sur le nb de lettres
sorter.setComparator(1, new StringSizeComparator());

// Ajout d'un ecouteur
final ListSelectionModel listSelectionModel = tableau.getSelectionModel();
listSelectionModel.addListSelectionListener(new TableauListSelectionListener

}

// Menu Fichier
private class OuvrirAction extends AbstractAction {
public OuvrirAction(String texte) {
super(texte);
}

public void actionPerformed(ActionEvent e) {
// TODO
}
}

private class SauverAction extends AbstractAction {
public SauverAction(String texte) {
super(texte);
}

public void actionPerformed(ActionEvent e) {
LOGGER.info("Sauver");

modele.sauver();
}
}

private class QuitterAction extends AbstractAction {
public QuitterAction(String texte) {
super(texte);
}

public void actionPerformed(ActionEvent e) {
LOGGER.info("Au revoir");
System.exit(0);
}
}

// Menu Affichage
private class MaleAction extends AbstractAction {

```

```

        public void actionPerformed(ActionEvent e) {
            modele.rechercheAvecFiltreSurSexe (Sexe.MALE);
        }
    }

    private class FemaleAction extends AbstractAction {
        public FemaleAction(String texte) {
            super(texte);
        }

        public void actionPerformed(ActionEvent e) {
            modele.rechercheAvecFiltreSurSexe (Sexe.FEMALE);
        }
    }

    private class MaleAndFemaleAction extends AbstractAction {
        public MaleAndFemaleAction(String texte) {
            super(texte);
        }

        public void actionPerformed(ActionEvent e) {
            modele.rechercheAvecFiltreSurSexe (null);
        }
    }

    private class AjouterLigneAction extends AbstractAction {

        /**
         * serialVersionUID
         */
        private static final long serialVersionUID = 7183768497443802311L;

        private AjouterLigneAction() {
            super("Ajouter");
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            LOGGER.debug("Click sur le bouton ajouter");

            final AjouterModifierChienActionHandler handler = new AjouterModifierChi
            AjouterModifierChienJDialog popup = new AjouterModifierChienJDialog(hand

            popup.setVisible(true);

            // final Chien idefix = new SimpleChien("Idefix", "Idefix", MALE,
            // LABRADOR, new String[] { "blanc" }, 25.0);
            // modele.ajouterChien(idefix);
        }
    }

    private class ModifierLigneAction extends AbstractAction {

        /**
         * serialVersionUID
         */
        private static final long serialVersionUID = 7183768497443802311L;

        private ModifierLigneAction() {
            super("Modifier");
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            LOGGER.debug("Click sur le bouton modifier");

            final int[] selection = tableau.getSelectedRows();

            if (selection == null || selection.length != 1) {
                // TODO erreur
            }
            final Chien chien = modele.getChienAt(selection[0]);

            final AjouterModifierChienActionHandler handler = new AjouterModifierChi
            AjouterModifierChienJDialog popup = new AjouterModifierChienJDialog(hand

            popup.setVisible(true);
        }
    }

    private class SupprimerLigneAction extends AbstractAction {

        /**
         * serialVersionUID
         */
        private static final long serialVersionUID = -5556554884674073716L;

        private SupprimerLigneAction() {
            super("Supprimer");
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            final int[] selection = tableau.getSelectedRows();
            for (int i = selection.length - 1; i >= 0; i--) {
                modele.supprimerChien(selection[i]);
            }
        }
    }

    private class CamembertMaleFemaleAction extends AbstractAction {

        /**
         * serialVersionUID
         */
        private static final long serialVersionUID = 4515460004284651581L;

        private CamembertMaleFemaleAction() {
            super("Camembert M/F");
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            LOGGER.debug("cliv clic");

            final List<Chien> chiens = modele.getChiens();
            int nbMale = 0;
            for (Chien chien : chiens) {

```

```

    }
    final int nbFemele = chiens.size() - nbMale;

    ratioMaleFemaleJdialog = new JDialog();
    ratioMaleFemaleJdialog.setTitle("Ratio H/F");

    final DefaultPieDataset pieDataset = new DefaultPieDataset();

    pieDataset.setValue("Female", nbFemele);
    pieDataset.setValue("Male", nbMale);

    final JFreeChart pieChart = ChartFactory.createPieChart("Ratio M/F", pie
    final ChartPanel cPanel = new ChartPanel(pieChart);

    ratioMaleFemaleJdialog.getContentPane().add(cPanel, CENTER);

    ratioMaleFemaleJdialog.pack();
    ratioMaleFemaleJdialog.setVisible(true);

    }
}

private class PoidsAction extends AbstractAction {

    /**
     * serialVersionUID
     */
    private static final long serialVersionUID = -2621678352128531798L;

    private PoidsAction() {
        super("Poids");
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        LOGGER.debug("cliv clic");

        String[] trancheNames = { "0-5", "6-10", "11-15", "16-20", "21-25", "26-30", "31+" };

        Map<String, Integer> map = new HashMap<String, Integer>();
        for (String trancheName : trancheNames) {
            map.put(trancheName, 0);
        }

        final List<Chien> chiens = modele.getChiens();

        for (Chien chien : chiens) {

            String tranche = "0-5";
            if (6 <= chien.getPoids()) {
                tranche = "6-10";
            }
            if (11 <= chien.getPoids()) {
                tranche = "11-15";
            }
            if (16 <= chien.getPoids()) {
                tranche = "16-20";
            }
            if (21 <= chien.getPoids()) {
                tranche = "21-25";
            }
            if (26 <= chien.getPoids()) {
                tranche = "26-30";
            }
            if (31 <= chien.getPoids()) {
                tranche = "31+";
            }

            Integer value = map.get(tranche);
            value++;
            map.put(tranche, value);
        }

        poidsJdialog = new JDialog();
        poidsJdialog.setTitle("Poids");

        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        for (String tranche : trancheNames) {
            final Integer nb = map.get(tranche);
            dataset.addValue(nb, "Poids", tranche);
        }

        final JFreeChart barChart = ChartFactory.createBarChart("Poids", "Tranche",
            dataset, PlotOrientation.VERTICAL, true, true, false);

        final ChartPanel cPanel = new ChartPanel(barChart);

        poidsJdialog.getContentPane().add(cPanel, CENTER);

        poidsJdialog.pack();
        poidsJdialog.setVisible(true);
    }
}

private class Poids2Action extends AbstractAction {

    /**
     * serialVersionUID
     */
    private static final long serialVersionUID = -2621678352128531798L;

    private Poids2Action() {
        super("Poids M/F");
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        LOGGER.debug("cliv clic");

        String[] trancheNames = { "0-5", "6-10", "11-15", "16-20", "21-25", "26-30", "31+" };

        Map<String, Integer> maleMap = new HashMap<String, Integer>();
        Map<String, Integer> femaleMap = new HashMap<String, Integer>();
        for (String trancheName : trancheNames) {
            maleMap.put(trancheName, 0);
            femaleMap.put(trancheName, 0);
        }
    }
}

```

```

        String tranche = "0-5";
        if (6 <= chien.getPoids()) {
            tranche = "6-10";
        }
        if (11 <= chien.getPoids()) {
            tranche = "11-15";
        }
        if (16 <= chien.getPoids()) {
            tranche = "16-20";
        }
        if (21 <= chien.getPoids()) {
            tranche = "21-25";
        }
        if (26 <= chien.getPoids()) {
            tranche = "26-30";
        }
        if (31 <= chien.getPoids()) {
            tranche = "31+";
        }

        final Map<String, Integer> map = (chien.getSexe() == Sexe.MALE) ? ma

        Integer value = map.get(tranche);
        value++;
        map.put(tranche, value);
    }

    poids2Jdial = new JDialog();
    poids2Jdial.setTitle("Poids M/F");

    final DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    for (String tranche : trancheNames) {
        final Integer nb = maleMap.get(tranche);
        dataset.addValue(nb, "Male", tranche);
    }

    for (String tranche : trancheNames) {
        final Integer nb = femaleMap.get(tranche);
        dataset.addValue(nb, "Female", tranche);
    }

    final JFreeChart barChart = ChartFactory.createBarChart("Poids", "Tranch
        dataset, PlotOrientation.VERTICAL, true, true, false);

    final ChartPanel cPanel = new ChartPanel(barChart);

    poids2Jdial.getContentPane().add(cPanel, CENTER);

    poids2Jdial.pack();
    poids2Jdial.setVisible(true);
}

private class PreferencesAction extends AbstractAction {
    public PreferencesAction(String texte) {
        super(texte);
    }

    public void actionPerformed(ActionEvent e) {
        // TODO
    }
}

private class AProposAction extends AbstractAction {
    public AProposAction(String texte) {
        super(texte);
    }

    public void actionPerformed(ActionEvent e) {
        // TODO
    }
}

private class TableauListSelectionListener implements ListSelectionListener {

    @Override
    public void valueChanged(final ListSelectionEvent event) {
        if (event.getValueIsAdjusting()) {
            return;
        }
        // System.out.println(event);

        activerOuDesactiverMenuEdition();
    }
}

private void activerOuDesactiverMenuEdition() {
    final int[] selection = tableau.getSelectedRows();

    // On veut activer le menu supprimer s'il y a au moins une ligne
    // selectionne.
    final boolean isSelection = selection != null && selection.length != 0;
    menuSupprimer.setEnabled(isSelection);

    // On ne veut modifier que s'il y a une seule ligne selectionnee.
    final boolean isUnEtSeulementUnSelection = isSelection && selection.length == 1;
    menuModifier.setEnabled(isUnEtSeulementUnSelection);
}
}

```

AjouterModifierChienActionHandler.java Sélectionnez

```

package com.icauda.tp.chien.ihm;

import org.apache.log4j.Logger;

import com.icauda.tp.chien.domain.Chien;

public class AjouterModifierChienActionHandler implements ActionHandler {

    private static final Logger LOGGER = Logger.getLogger(AjouterModifierChienAction
        Handler.class);

    private ModeleDynamique2 modele;

```



```

@Override
public void process(final Action action) throws ChienException {
    LOGGER.debug("process");
    process(action, null);
}

@Override
public void process(final Action action, final Chien chien) throws ChienException {
    LOGGER.debug("process");

    switch (action) {
        case ANNULER:
            annuler();
            break;

        case MODIFIER:
            modifier(chien);
            break;

        case CREER:
            ajouter(chien);
            break;

        case SUPPRIMER:
            supprimer(chien);
            break;

        default:
            throw new IllegalArgumentException("L'action demandee n'est pas disponible");
    }
}

private void ajouter(final Chien chien) throws ChienException {
    LOGGER.debug("Ajouter");

    if (!valider(chien)) {
        // TODO lancer une exception...
    }

    // On a acces au modele et on peut donc demander l'ajout.
    modele.ajouterChien(chien);
}

private void modifier(final Chien chien) throws ChienException {
    LOGGER.debug("Modifier");

    if (!valider(chien)) {
        // TODO lancer une exception...
    }

    modele.modifierChien(chien);
}

private void annuler() {
    LOGGER.debug("Annuler");
    // Ici rien a faire
}

private void supprimer(final Chien chien) {
    LOGGER.debug("Supprimer");
    throw new UnsupportedOperationException("Cette fonction n est pas encore disponible");
}

/**
 * Valide que les attributs du chien sont bien renseignes.
 */
private boolean valider(final Chien chien) {

    return true;
}
}

```

AjouterModifierChienJDialog.java Sélectionnez

```

package com.icauda.tp.chien.ihm;

import static java.awt.BorderLayout.CENTER;
import static java.awt.BorderLayout.SOUTH;
import static javax.swing.SwingConstants.RIGHT;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.util.List;

import javax.swing.AbstractAction;
import javax.swing.DefaultListCellRenderer;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JTextField;

import org.apache.log4j.Logger;

import com.google.common.base.Joiner;
import com.google.common.base.Splitter;
import com.google.common.collect.Lists;
import com.icauda.tp.chien.domain.Chien;
import com.icauda.tp.chien.domain.RaceDeChien;
import com.icauda.tp.chien.domain.Sexe;
import com.icauda.tp.chien.domain.SimpleChien;

public class AjouterModifierChienJDialog extends JDialog {

    private static final Logger LOGGER = Logger.getLogger(AjouterModifierChienJDialog.class);
    private ActionHandler actionHandler;

    private JTextField nom = new JTextField(15);
    private JTextField nomCompleet = new JTextField(15);
    private JComboBox sexe = new JComboBox();
    private JComboBox race = new JComboBox();
}

```

```
private JTextField couleurs = new JTextField(15);
private JTextField poids = new JTextField(15);

private JButton ajouterButton = new JButton(new AjouterAction("Ajouter"));
private JButton modifierButton = new JButton(new ModifierAction("Modifier"));
private JButton annulerButton = new JButton(new AnnulerAction("Annuler"));

private static final int widthLabel = 100;
private static final int widthChamp = 200;
// private static final int widthChampMultiple = 60;
private static final int heightLigne = 20;

private static final int espace = 5;

private int y = 0;

private JPanel formPanel = new JPanel(null);
private JPanel buttonPanel = new JPanel();

private Chien chien;

public AjouterModifierChienJDialog(final ActionHandler actionHandler) {
    this(actionHandler, null);
}

public AjouterModifierChienJDialog(final ActionHandler actionHandler, final Chie
    super();

    this.actionHandler = actionHandler;
    this.chien = chien;

    if (chien == null) {
        setTitle("Ajouter un chien");
    } else {
        setTitle("Modifier un chien");
    }

    // La taille est calculee a l'aide du panel (cf. addFormulaire).
    // setPreferredSize(new Dimension(400, 250));
    setModal(true);

    // Formulaire
    addFormulaire();
    preremplirFormulaire();

    // Boutons
    addBoutons();

    pack();
}

private void addBoutons() {
    if (chien == null) {
        buttonPanel.add(ajouterButton);
    } else {
        buttonPanel.add(modifierButton);
    }
    buttonPanel.add(annulerButton);
    add(buttonPanel, SOUTH);
}

private void addFormulaire() {
    // Noms
    ajouter("Nom :", nom);
    ajouter("Nom complet :", nomComplet);

    // Sexes
    sexe.addItem(Sexe.MALE);
    sexe.addItem(Sexe.FEMALE);
    sexe.setRenderer(new DefaultListCellRenderer() {
        @Override
        public Component getListCellRendererComponent(JList list, Object value,
            final Sexe sexe = (Sexe) value;
            setText((sexe == Sexe.MALE) ? "Male" : "Female");
            return this;
        }
    });
    ajouter("Sexe :", sexe);

    // Races
    for (RaceDeChien raceDeChien : RaceDeChien.values()) {
        race.addItem(raceDeChien);
    }
    race.setRenderer(new DefaultListCellRenderer() {
        @Override
        public Component getListCellRendererComponent(JList list, Object value,
            final RaceDeChien raceDeChien = (RaceDeChien) value;
            setText(raceDeChien.getLabel());
            return this;
        }
    });
    ajouter("Race :", race);

    // Couleurs, poids
    ajouter("Couleurs :", couleurs);
    ajouter("Poids :", poids);

    // Si on avait voulu mettre des champs pour la date de naissance, on
    // aurait pu faire comment suit :
    // for(int i = 1; i <= 31;i++) {
    //     jour.addItem(i);
    // }
    // //
    // // Date de naissance
    // for (int i = 1; i <= 31; i++) {
    //     jour.addItem(i);
    // }
    // for (int i = 1; i <= 12; i++) {
    //     mois.addItem(i);
    // }
    // for (int i = 2013; 1970 < i; i--) {
    //     annee.addItem(i);
    // }
    // ajouter("Naissance :", jour, mois, annee);

    // Taille du panel
```

```

        add(formPanel, CENTER);
    }

    private void preremplirFormulaire() {
        if (chien == null) {
            return;
        }

        nom.setText(chien.getNom());
        nomCompleet.setText(chien.getNomCompleet());
        poids.setText(chien.getPoids().toString());
        couleurs.setText(Joiner.on(", ").join(chien.getCouleurs()));

        sexe.setSelectedItem(chien.getSexe());
        race.setSelectedItem(chien.getRace());
    }

    private void ajouter(final String label, final JComponent champ) {
        y += espace;

        final JLabel l = new JLabel(label);
        l.setBounds(espace, y, widthLabel, heightLigne);
        l.setHorizontalAlignment(RIGHT);

        champ.setBounds(2 * espace + widthLabel, y, widthChamp, heightLigne);

        y += heightLigne;

        formPanel.add(l);
        formPanel.add(champ);
    }

    // private void ajouter(final String label, final JComponent champ1, final
    // JComponent champ2, final JComponent champ3) {
    //     y += espace;
    //     //
    //     // final JLabel l = new JLabel(label);
    //     // l.setBounds(espace, y, widthLabel, heightLigne);
    //     // l.setHorizontalAlignment(RIGHT);
    //     //
    //     // int x = 2 * espace + widthLabel;
    //     // champ1.setBounds(x, y, widthChampMultiple, heightLigne);
    //     //
    //     // x += espace + widthChampMultiple;
    //     // champ2.setBounds(x, y, widthChampMultiple, heightLigne);
    //     //
    //     // x += espace + widthChampMultiple;
    //     // champ3.setBounds(x, y, widthChampMultiple, heightLigne);
    //     //
    //     // y += heightLigne;
    //     //
    //     // formPanel.add(l);
    //     // formPanel.add(champ1);
    //     // formPanel.add(champ2);
    //     // formPanel.add(champ3);
    // }

    private class AjouterAction extends AbstractAction {
        public AjouterAction(String texte) {
            super(texte);
        }

        public void actionPerformed(ActionEvent event) {
            LOGGER.debug("Ajouter");

            final SimpleChien nouveauChien = new SimpleChien();
            remplirChien(nouveauChien);

            try {
                actionHandler.process(Action.CREER, nouveauChien);
            } catch (ChienException exception) {
                // TODO
            }

            // Fermeture de la fenetre
            AjouterModifierChienJDialog.this.closePopup();
        }
    }

    private class ModifierAction extends AbstractAction {
        public ModifierAction(String texte) {
            super(texte);
        }

        public void actionPerformed(ActionEvent e) {
            LOGGER.debug("Modifier");

            if (!(chien instanceof SimpleChien)) {
                // TODO erreur
            }

            remplirChien((SimpleChien) chien);

            try {
                actionHandler.process(Action.MODIFIER, chien);
            } catch (ChienException exception) {
                // TODO
            }

            // Fermeture de la fenetre
            AjouterModifierChienJDialog.this.closePopup();
        }
    }

    private void remplirChien(final SimpleChien chien) {
        chien.setNom(nom.getText());
        chien.setNomCompleet(nomCompleet.getText());

        try {
            final List<String> couleurList = Lists.newArrayList( //
                Splitter.on(",") //
                    .trimResults() //
                    .omitEmptyStrings() //
                    .split(couleurs.getText()));
            chien.setCouleurs(couleurList);
        } catch (Exception e) {
            // TODO
        }
    }

```

```

        // Ne pas oublier que les chiffres utilisent des points et non des
        // virgules
        temp = temp.replace(',', '.');
        // Suppression des espaces
        temp = temp.replaceAll(" ", "");

        if (temp.isEmpty()) {
            // TODO erreur
        }

        chien.setPoids(Double.valueOf(temp));

    } catch (NullPointerException e) {
        // TODO si temp est null
    }

    } catch (NumberFormatException e) {
        // TODO Si temp n'est pas un chiffre correct
    }

    chien.setRace((RaceDeChien) race.getSelectedItem());
    chien.setSexe((Sexe) sexe.getSelectedItem());

}

private class AnnulerAction extends AbstractAction {
    public AnnulerAction(String texte) {
        super(texte);
    }

    public void actionPerformed(ActionEvent e) {
        LOGGER.debug("Annuler");
        // Ici pas besoin d'appeler le handler.
        AjouterModifierChienJDialog.this.closePopup();
    }
}

private void closePopup() {
    // Vidage des champs...
    // ici pas besoin

    // Fermeture de la fenetre.
    dispose();
}
}

```

EngineCsvChienDao.java Sélectionnez

```

package com.icauda.tp.chien.dao.csv;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.io.InputStream;
import java.util.HashMap;
import java.util.List;

import org.apache.log4j.Logger;

import com.google.common.base.Function;
import com.google.common.collect.Lists;
import com.icauda.tp.chien.domain.Chien;
import com.icauda.tp.chien.domain.SimpleChien;

import fr.ybonnel.csvengine.CsvEngine;
import fr.ybonnel.csvengine.factory.AbstractCsvReader;
import fr.ybonnel.csvengine.factory.DefaultCsvManagerFactory;
import fr.ybonnel.csvengine.factory.OpenCsvReader;
import fr.ybonnel.csvengine.model.Error;
import fr.ybonnel.csvengine.model.Result;

public class EngineCsvChienDao extends AbstractCsvChienDao {

    // #####
    // ##### Attributs #####
    // #####
    private static final Logger LOGGER = Logger.getLogger(EngineCsvChienDao.class);

    // #####
    // ##### Methodes diverses #####
    // #####

    private void setEngineFactory(final CsvEngine engine) {
        engine.setFactory(new DefaultCsvManagerFactory() {
            @Override
            public AbstractCsvReader createReaderCsv(Reader reader, char separator)
                return new OpenCsvReader(reader, separator) {
                    @Override
                    public String[] readLine() throws IOException {
                        String[] nextLine = super.readLine();
                        if (isLineAComment(nextLine)) {
                            nextLine = readLine();
                        }
                        return nextLine;
                    }

                    private boolean isLineAComment(String[] line) {
                        return line != null && line.length > 0 && line[0].startsWith("#");
                    }
                };
        });
    }

    @SuppressWarnings("unchecked")
    @Override
    protected void reloadChiens() {
        LOGGER.debug("reloadChiens");

        try {
            final CsvEngine engine = new CsvEngine(SimpleChien.class);
            setEngineFactory(engine);

            final FileInputStream fis = new FileInputStream(file);

            final Result<SimpleChien> resultat = engine.parseInputStream(fis, Simple

```

```

        chienMapByNom = new HashMap<String, Chien>(chiens.size());
        for (Chien chien : chiens) {
            LOGGER.debug("[chien] " + chien);
            chienMapByNom.put(chien.getNom(), chien);
        }

        List<Error> errors = resultat.getErrors();
        LOGGER.debug(errors);

        entetes = engine.getColumnNames(SimpleChien.class);

        LOGGER.debug("[entetes] " + entetes);

    } catch (Exception e) {
        LOGGER.error("Une erreur s'est produite...", e);
    }
}

@Override
public void sauver(final List<Chien> chiens, final File file) {
    final List<SimpleChien> simpleChiens = Lists.transform(chiens, new Function<
        public SimpleChien apply(final Chien chien) {
            return (SimpleChien) chien;
        }
    ));

    doSauver(simpleChiens, file);
}

private void doSauver(final List<SimpleChien> chiens, final File file) {
    if (LOGGER.isDebugEnabled()) {
        LOGGER.debug("Sauvegarde des " + chiens.size() + " chiens dans le fichier");
    }

    final CsvEngine engine = new CsvEngine(SimpleChien.class);

    try {
        engine.writeFile(new FileWriter(file), chiens, SimpleChien.class);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
}

```

Pour le web service, je ne vous donne pas une solution avec le code intégré directement où il faut dans le programme. Je vous laisse tout de même chercher un peu. Et pas la peine de chercher dans le fichier Zip car le code sera proposé dans un main...

Chat.java Sélectionnez

```

package com.icauda.tp.chien.domain;

import org.codehaus.jackson.map.annotate.JsonDeserialize;

@JsonDeserialize(using = ChatDeserializer.class)
public class Chat {
    private String nom;
    private Sexe sexe;
    private String couleurs;
    private double poids;
    private double prix;

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("Chat {nom=");
        builder.append(nom);
        builder.append(", sexe=");
        builder.append(sexe);
        builder.append(", couleurs=");
        builder.append(couleurs);
        builder.append(", poids=");
        builder.append(poids);
        builder.append(", prix=");
        builder.append(prix);
        builder.append("}");
        return builder.toString();
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((couleurs == null) ? 0 : couleurs.hashCode());
        result = prime * result + ((nom == null) ? 0 : nom.hashCode());
        long temp;
        temp = Double.doubleToLongBits(poids);
        result = prime * result + (int) (temp ^ (temp >> 32));
        temp = Double.doubleToLongBits(prix);
        result = prime * result + (int) (temp ^ (temp >> 32));
        result = prime * result + ((sexe == null) ? 0 : sexe.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Chat other = (Chat) obj;
        if (couleurs == null) {
            if (other.couleurs != null)
                return false;
        } else if (!couleurs.equals(other.couleurs))
            return false;
        if (nom == null) {
            if (other.nom != null)
                return false;
        } else if (!nom.equals(other.nom))
            return false;
        if (Double.doubleToLongBits(poids) != Double.doubleToLongBits(other.poids))
            return false;
    }
}

```

```

        return false;
        return true;
    }

    // Getters et Setters

```

ResultChat.java Sélectionnez

```

public class ResultChat {

    private String sexe;
    private String espece;
    private int nombre;
    private List<Chat> animaux;

    public ResultChat() {
    }
}

```

TesterWS.java Sélectionnez

```

package com.icauda.tp.chien;

import java.io.IOException;

import javax.ws.rs.client.ClientBuilder;

import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.map.JsonMappingException;
import org.codehaus.jackson.map.ObjectMapper;

public class TesterWS {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Un web service avec des chats...");

        final String url = "http://www.icauda.com/animal-mignon/selection.php";

        String responseEntity = ClientBuilder.newClient()//
            .target(url)//
            .queryParam("espece", "chat")//
            .queryParam("sexe", "m")//
            .request()//
            .get(String.class);

        System.out.println("responseEntity: " + responseEntity);

        final ObjectMapper mapper = new ObjectMapper();
        ResultChat myCat = null;
        try {
            myCat = mapper.readValue(responseEntity, ResultChat.class);
        } catch (JsonParseException e) {
            e.printStackTrace();
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println(myCat);
        if (myCat != null) {
            System.out.println(myCat.getAnimaux());
        }

        System.out.println("fin");
    }
}

```

ChatDeserializer.java Sélectionnez

```

package com.icauda.tp.chien.domain;

import java.io.IOException;

import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.ObjectCodec;
import org.codehaus.jackson.map.DeserializationContext;
import org.codehaus.jackson.map.JsonDeserializer;

public class ChatDeserializer extends JsonDeserializer<Chat> {

    public Chat deserialize(JsonParser jsonParser, DeserializationContext deserializ

        final ObjectCodec oc = jsonParser.getCodec();
        final JsonNode node = oc.readTree(jsonParser);

        final String nom = node.get("nom").getTextValue();
        final String sexe = node.get("sexe").getTextValue();
        final String couleurs = node.get("couleurs").getTextValue();
        final double poids = node.get("poids").getDoubleValue();
        final double prix = node.get("prix").getDoubleValue();

        final Chat chat = new Chat();
        chat.setNom(nom);
        chat.setSexe(Sexe.valueOfByLettre(sexe));
        chat.setCouleurs(couleurs);
        chat.setPoids(poids);
        chat.setPrix(prix);

        return chat;
    }
}

```

On aurait pu également demander la conversion à l'aide d'annotation depuis la classe Chat.

X-D. Web services en PHP▲

Votre prof ne connaît pas seulement Java ;-))

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [Fermer](#)

```

<?php

// Objets
class Chat {
    var $nom;
    var $sexe;
    var $poids;
    var $couleurs;
    var $prix;

    function __construct($aNom, $aSexe, $aPoids, $aCouleurs, $aPrix) {
        $this->nom = $aNom;
        $this->sexe = $aSexe;
        $this->poids = $aPoids;
        $this->couleurs = $aCouleurs;
        $this->prix = $aPrix;
    }
}

class Lapin {
    var $nom;
    var $sexe;
    var $tailleOreille;
    var $couleurs;
    var $prix;

    function __construct($aNom, $aSexe, $aTailleOreille, $aCouleurs, $aPrix) {
        $this->nom = $aNom;
        $this->sexe = $aSexe;
        $this->tailleOreille = $aPoids;
        $this->couleurs = $aCouleurs;
        $this->prix = $aPrix;
    }
}

class Souris {
    var $nom;
    var $sexe;
    var $tailleQueue;
    var $couleurs;
    var $couleursYeux;
    var $prix;

    function __construct($aNom, $aSexe, $aTailleQueue, $aCouleurs, $aCouleursYeux, $
        $this->nom = $aNom;
        $this->sexe = $aSexe;
        $this->tailleOreille = $aPoids;
        $this->couleurs = $aCouleurs;
        $this->couleursYeux = $aCouleursYeux;
        $this->prix = $aPrix;
    }
}

// Animaux
$animaux = array();

// Chats
$chats = array();
$chats[] = new Chat('Garfield', 'm', 12.5, 'noir, orange', 150);
$chats[] = new Chat('Lulu', 'f', 10.5, 'gris, noir', 150);
$chats[] = new Chat('Lulu', 'f', 5, 'gris, noir, blanc', 75);
$chats[] = new Chat('Minou', 'm', 8.5, 'noir', 150);
$chats[] = new Chat('Pupuce', 'f', 13.0, 'noir, jaune', 200);
$chats[] = new Chat('Scarlette', 'f', 8.2, 'blanc', 250);
$chats[] = new Chat('Boulette', 'f', 15.7, 'blanc', 250);
$chats[] = new Chat('Ronron', 'm', 8.0, 'noir', 250);
$chats[] = new Chat('Scarlette', 'f', 8.2, 'blanc', 250);
$chats[] = new Chat('Scarlette', 'f', 8.2, 'blanc', 250);
$animaux['chat'] = $chats;

// Lapins
$lapins = array();
$lapins[] = new Lapin('Bunny', 'm', 15, 'blanc', 25);
$lapins[] = new Lapin('Vincent', 'm', 14, 'noir', 25);
$lapins[] = new Lapin('Jessica', 'f', 15, 'rose', 25);
$lapins[] = new Lapin('Roger', 'm', 18, 'blanc, gris', 25);
$animaux['lapin'] = $lapins;

// Souris
$souris = array();
$souris[] = new Souris('Mickey', 'm', 10, 'noir', 'noir', 120);
$souris[] = new Souris('Mini', 'f', 10, 'noir, rose', 'noir', 120);
$souris[] = new Souris('Ratatouille', 'm', 7, 'gris', 'noir', 70);
$souris[] = new Souris('Speedy', 'm', 12, 'gris', 'rouge', 150);
$souris[] = new Souris('Bianca', 'f', 9, 'blanc', 'bleu', 150);
$souris[] = new Souris('Bernard', 'h', 10, 'gris', 'noir', 150);
$animaux['souris'] = $souris;

?>

```

selection.php Sélectionnez

```

<?php
// Content-type
header('Content-type: application/json');

// Parametres de la requete
function isParameterInRequest($key) {
    return array_key_exists($key, $_REQUEST);
}

function getFromRequest($key) {
    if(isParameterInRequest($key)) {
        return $_REQUEST[$key];
    } else {
        return null;
    }
}

// Objets
require_once('animaux.php');

// Objets de retour et erreur
class ErrorResult {

```



```

        $this->message = $aMessage;
    }
}

class AnimalResult {
    var $sexe;
    var $espece;
    var $animaux;
    var $nombre;

    function __construct($aSexe, $aEspece, $aAnimaux, $aNombre) {
        $this->sexe = $aSexe;
        $this->espece = $aEspece;
        $this->animaux = $aAnimaux;
        $this->nombre = $aNombre;
    }
}

// Parametres de selection
$sexe = getFromRequest('sexe');
$espece = getFromRequest('espece');

// Choix de l'animal
if($espece == null || $espece == '') {
    $errorResult = new ErrorResult('Le site ne connait pas cette espece.');
```

dispo.php
Sélectionnez

```

<?php
// Content-type
header('Content-type: application/json');
```

```

// Parametres de la requete
function isParameterInRequest($key) {
    return array_key_exists($key, $_REQUEST);
}

function getFromRequest($key) {
    if(isParameterInRequest($key)) {
        return $_REQUEST[$key];
    } else {
        return null;
    }
}

// Objets
require_once('animaux.php');
```

```



// Objets de retour et erreur
class EspeceResult {
    var $stock;

    function __construct($aStock) {
        $this->stock = $aStock;
    }
}

$stab = array();
foreach(array_keys($animaux) as $espece) {
    $stab[$espece] = count($animaux[$espece]);
}
$especeResult = new EspeceResult($stab);

echo json_encode($especeResult);

?>
```

Vous avez aimé ce tutoriel ? Alors partagez-le en cliquant sur les boutons suivants :  

jusqu'à 300 000 € de dommages et intérêts.

Responsables bénévoles de la rubrique Java : Mickael Baron - Robin56 - Contacter par email

[Nous contacter](#) [Participez](#) [Hébergement](#) [Informations légales](#) [Partenaire : Hébergement Web](#)
Copyright © 2000-2018 - www.developpez.com