

Rapport TP1 TLC

Le 23 Février 2017

Réalisé par:

PITON Stephen EL GHZIZAL Yousra PARISSE Simon

URL de notre google app Engine :

<http://tlctp1-156207.appspot.com/>

URL de notre dépôt Git :

<https://github.com/yousrael/TP1TLC>

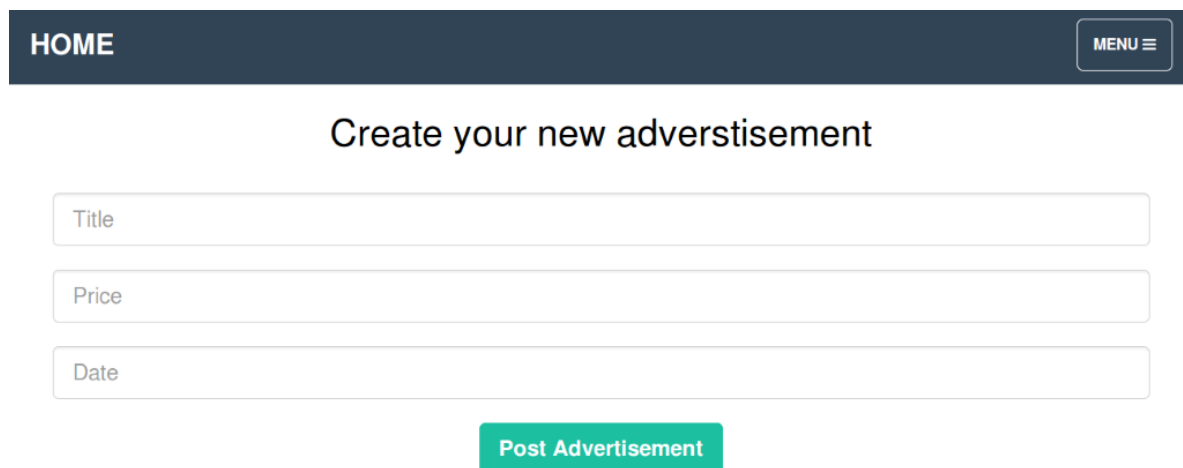
Ajouter une annonce

Un annonce est composée de cinq attributs (**ID de l'auteur, Mail de l'auteur, Titre, Prix, Date**).

Il est possible de renseigner le **titre**, le **prix** et la **date** dans un formulaire.

Lors du clic sur le bouton "Post Advertisement" , la méthode **POST** de la servlet **SignAdvertisementServlet** est appelée. Les différents champs sont récupérés et validés pour ensuite être ajoutés dans la base de données mise à disposition sur le projet Google App Engine.

IHM :



The screenshot shows a web application interface. At the top, there is a dark blue header bar with the word "HOME" on the left and a "MENU" button with a hamburger icon on the right. Below the header, the text "Create your new adverstisement" is centered. Underneath this text are three input fields stacked vertically, labeled "Title", "Price", and "Date". At the bottom of the form is a green button with the text "Post Advertisement".

Lister les Annonces :

Les annonces sont listées dans un tableau. Toutes les annonces sont normalement récupérées en une seul fois, Mais nous , nous avons limité notre affichage à 10 annonces qui s'affichent dans des blocs avec un scroll comme le montre la capture suivante :

Advertisement n°2**Title :** Twingo 200 000 km**Price :** 2200.0 \$**Date :** Wed Feb 01 00:00:00 CET 2017**Author :** An anonymous person**Recherche et filtres :**

Nous avons fait le choix de proposer un filtrage des annonces par titre d'annonce et par intervalle de Prix et de Dates.

- Filtrage par Titre:

Nous avons fait une recherche avec le titre entier d'une annonce car Il est compliqué de faire une recherche de contenu sur les différents mots du titre. Il faudrait indexer chaque mot de titre et faire une recherche sur cette index. Par manque de temps, nous n'avons pu investiguer cette solution.

Code :

```
List<Advertisement> advertisements2 =  
ObjectifyService.ofy().load().type(Advertisement.class).filter("title",  
request.getParameter("filter")).list();
```

- Filtrage par Prix :

Code :

```
List<Advertisement> advertisements2 =  
ObjectifyService.ofy().load().type(Advertisement.class).filter("price  
>", Double.parseDouble(request.getParameter("priceMin"))).filter("price  
<", Double.parseDouble(request.getParameter("priceMax"))).list();
```

- Filtrage par Date :

Code :

```
List<Advertisement>advertisements2=
ObjectifyService.ofy().load().type(Advertisement.class).filter("date
>",dateMin).filter("date <",dateMax).list();
```

IHM :

The screenshot shows a web application interface with a dark blue header. The header contains the word "HOME" on the left and a "MENU" button with a hamburger icon on the right. Below the header, there is a red "Sign out" button. A search bar with the placeholder text "Search" is positioned below the sign out button. Under the search bar, there are two price filters. The first filter is labeled "MINIMUM PRICE:" in green, followed by a horizontal slider with a white dot. Below the slider, the number "50" is displayed. The second filter is labeled "MAXIMUM PRICE:" in green, followed by a horizontal slider with a white dot. Below the slider, the number "800" is displayed. At the bottom of the form, there are two date input fields. The first field contains "01-01-2017" and the second field contains "21-02-2017". To the right of these fields is a green "Search" button.

Supprimer les annonces recherchées:

Pour la suppression des annonces, nous avons crée un bouton 'Remove All visible Entry' qui permet de supprimer toutes les annonces affichées lors d'une recherche soit par titre,prix,date.Lors d'un clic sur le bouton la suppression de la base est réalisée et les annonces qui ne sont pas concernées par la recherche s'affichent.

HOME

Search

Sign in

MINIMUM PRICE:

23

MAXIMUM PRICE:

01-01-2000

01-01-2022

Search

Price : 1.0E7 \$

Date : Wed Jun 21 00:00:00 CEST 2017

Author : An anonymous person

Remove All Visible Entry

Problème rencontré :

Il n'est pas possible d'utiliser plusieurs filtres en même temps lors de la requête filter sur l'ObjectifyService.

Solution au problème :

On applique un seul des filtres sur l'ObjectifyService et on boucle sur la liste pour appliquer les autres filtres un par un. Nous avons laissé le filtre par intervalle de prix comme le filtre principale.

Test des requêtes :

Une classe de test a été codée pour exécuter plusieurs requêtes à la suite, tels que l'ajout de plusieurs annonces afin de mesurer la scalabilité de Google App Engine.

Classe : **Test.java**

Le fichier test comprend des fonctions d'insertion, de recherche et de suppression avec un paramètre sur le nombre d'actions à effectuer en un appel.

Les différents tests ont été effectués à la suite lors d'une exécution du fichier Test.java, par insertion, recherche et suppression successives d'une centaine d'élément à chaque fois. Le fichier de test a été exécuté une fois tout seul, puis une deuxième fois avec 5 instances en parallèle.

Si on regarde les temps moyens pour les différentes actions, on constate qu'il sont relativement similaires, ce qui indique que le service google scale notre application de façon optimale.

Cependant, lors du test intensif la page principale s'est retrouvée inaccessible assez vite du fait des trop nombreuses requêtes.

insert min: 261ms	max: 4713ms	moyenne: 434ms	total: 43403ms
search min: 298ms	max: 9022ms	moyenne: 577ms	total: 57728ms

delete min: 245ms	max: 1644ms	moyenne: 428ms	total: 4283ms
insert min: 253ms	max: 908ms	moyenne: 361ms	total: 36188ms
delete min: 229ms	max: 1517ms	moyenne: 401ms	total: 4014ms
insert min: 267ms	max: 633ms	moyenne: 354ms	total: 35497ms
delete min: 260ms	max: 1531ms	moyenne: 424ms	total: 4243ms
insert min: 270ms	max: 724ms	moyenne: 345ms	total: 34508ms
delete min: 254ms	max: 1076ms	moyenne: 421ms	total: 4212ms
search min: 358ms	max: 1480ms	moyenne: 513ms	total: 51305ms
insert min: 253ms	max: 555ms	moyenne: 339ms	total: 33969ms
insert min: 256ms	max: 668ms	moyenne: 344ms	total: 34451ms
insert min: 252ms	max: 1202ms	moyenne: 362ms	total: 36274ms
search min: 370ms	max: 1948ms	moyenne: 676ms	total: 67631ms
search min: 219ms	max: 4540ms	moyenne: 618ms	total: 61889ms
insert min: 270ms	max: 3993ms	moyenne: 385ms	total: 38570ms
search min: 227ms	max: 4611ms	moyenne: 576ms	total: 57643ms
insert min: 256ms	max: 3725ms	moyenne: 388ms	total: 38824ms
search min: 214ms	max: 4558ms	moyenne: 574ms	total: 57451ms
delete min: 203ms	max: 2065ms	moyenne: 487ms	total: 4870ms

Dans les images suivantes on peut voir que lors de l'exécution des 5 tests en parallèle vers notre application, le nombre de requêtes à augmenter et le nombre d'instance de notre serveur à aussi augmenter automatiquement.

