Préparation et déploiement en Cloud
Zoo Manager

par Geoffrey Spaur

31 mars 2017

Contents

1 Présentation

Le projet a pour but de réaliser une API REST. Ce projet devra être déployé sur le cloud BlueMix.

2 Problèmes Rencontrés

2.1 Content-type

Avant de déployer mon api rest sur le cloud, nous devions dans un premier temps vérifier notre code en local. J'ai donc décidé d'utiliser GlassFish sur ma machine pour tester mon api avec un client Java. Ce test m'a permit de détecter un premier problème. En effet, j'avais une erreur 500 disant que ma requête étant mal formé. En faisant de multiple recherche, notamment grâce à la commande:

```
$ nc -l 8080
```

Nous avons pu mettre en évidence que mon client envoyait de mauvais header à mon serveur:

```
Content-Type: text/plain
```

J'ai donc du recréer intégralement mon client pour pouvoir changer les headers de ma requête. Nous pourrons maintenant voir ceci dans les header:

```
Content-Type: application/xml
Accept: application/xml
```

2.2 Refacto du TP1

Au passage du TP-REST au TP-CLOUD, les bases du TP ont légèrement changé. En effet l'utilisation du framework Spring nous permit de mettre des annotation sur nos contrôleur afin de pouvoir les mapper automatiquement. Nous avons donc du importer tout le code écris dans le précédent TP.

2.3 Installation du cli

Une petite difficulté ici, car nous travaillons sur les machines de l'université. Donc nous n'avons pas d'accès root pour pouvoir installer le paquet requit. Nous avons donc du le télécharger ici: `https://cli.run.pivotal.io/stable?release=linux64-binary&source=github`. Et créer un alias dans notre .bashrc:

```
alias cf = '~/cli/cf'
```

2.4 La persistance

2.4.1 L'instanciation de la base de données

Dans un premier il nous était demandé de créer notre base de données de le cloud. J'ai pour cela utilisé la plateforme BlueMix afin de d'instancier et de lier la base de données à mon application sur Tomcat. J'ai utilisé Compose-SQL pour cela. Après la création de ce service des Crédentials nous ont été fournis.

2.4.2 La communication

Toute la difficulté ici à été de mettre les bons objets dans les bonnes cases avec très peut de visibilité. J'ai donc appris à mes dépend, que je devais utiliser la classe:

```
Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
```

En utilisant l'url suivant:

```
jdbc:mysql://sl-eu-lon-2-portal.1.dblayer.com:17052
```

Sans oublier de désactiver le SSL:

```
jdbc:mysql://sl-eu-lon-2-portal.1.dblayer.com:17052?useSSL=false
```

Avec les identifiant suivant:

```
Login: admin
Password: LREAOQURMKGBDMFV
```

2.4.3 L'initialisation de la base de données

Lors de la création de la base, celle-ci est vide. Il faut créer une nouvelle base pour pouvoir y ajouter nos tables:

```
CREATE DATABASE zoo;
```

Sans oublier que notre DAO doit utiliser la base que l'on vient de créer:

```
Statement stmt = connection.createStatement();
stmt.executeUpdate("USE zoo");
```

Pour finir j'ai trouvé un petit problème lors de l'ajout d'une ligne dans ma table animals. En cherchant sur internet, j'ai trouvé qu'il fallait une clé primaire à ma table, sinon cela provoquait des erreurs.