UFR Sciences et Techniques

Master 1 GIL – Langage Web 2

Description du Serveur

Réalisé par :

- AOUDJIT Said Oulguendouz
- MEDDAH Amine

20 Avril 2017

Master 1 GIL – Langage Web 2

Table des matières

1. *Adresse du service REST*

L'adresse de notre service REST est : https://sepabank.herokuapp.com/

2. *Description des requêtes*

URL	Méthode	Description
/detail	GET	Renvoie un flux XML contenant la liste des transactions détaillées
/resume	GET	Renvoie un flux XML contenant la liste des transactions résumées
/stats	GET	Afficher une synthèse des transactions stockées, avec les informations suivantes : Nombre de transactions, montant total des transactions
/trx/id	GET	Renvoie un flux XML décrivant le détail de la transaction d'identifiant id avec **id = PmtId**
/depot	POST	Dépose une nouvelle transaction. Elle Reçoit un flux XML décrivant une transaction. Un message de retour indique le résultat de l'opération, avec le numéro d'identification en cas de succès, et un message d'erreur sinon.

3. *Liste des technologies utilisées*

1. *Spring MVC Framework*

Nous avons utilisé le Framework Spring MVC pour réaliser notre application serveur. Les éléments de notre serveur qui correspondent aux éléments de Spring MVC sont :

1. Le Model

   C'est les class qui définissent les déférents éléments d'une transaction SEPA.

2. Les Vues

Nous avons utilisé une vue qui est home.jsp pour afficher la page d'accueil de notre serveur.

3. Les Contrôleurs

Nous avons utilisé deux contrôleurs :

a. HomeController.java : il redirige les requêtes de la page d'accueil
b. SepaController.java : il redirige les requêtes qui offrent les services possibles sur une transaction SEPA

2. JdbcTemplate *et MySQL*

Nous avons utilisé MySQL come Système de Gestion de Base de Données (SGBD).

Note base de données contient une table Transaction qui contient tous les champs d'une transaction pour pouvoir sauvegarder les transaction SEPA de notre système. Le script sql qui crée cette table est :

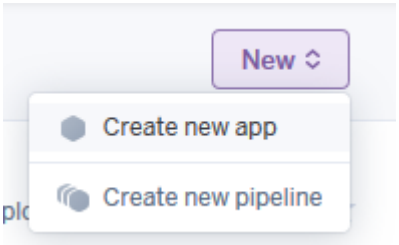
```
CREATE TABLE `transaction` (
 `transaction_id` int(11) NOT NULL AUTO_INCREMENT,
 `num` varchar(6) NOT NULL,
 `PmtId` varchar(35) NOT NULL,
 `InstdAmt` DOUBLE NOT NULL,
 `MndtId` varchar(35) NOT NULL,
 `DtOfSgntr` varchar(10) NOT NULL,
 `BIC` varchar(11) NOT NULL,
 `Nm` varchar(35) NOT NULL,
 `IBAN` varchar(34) NOT NULL,
 `RmtInf` varchar(50) NOT NULL,
 PRIMARY KEY (`transaction_id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci ;
```

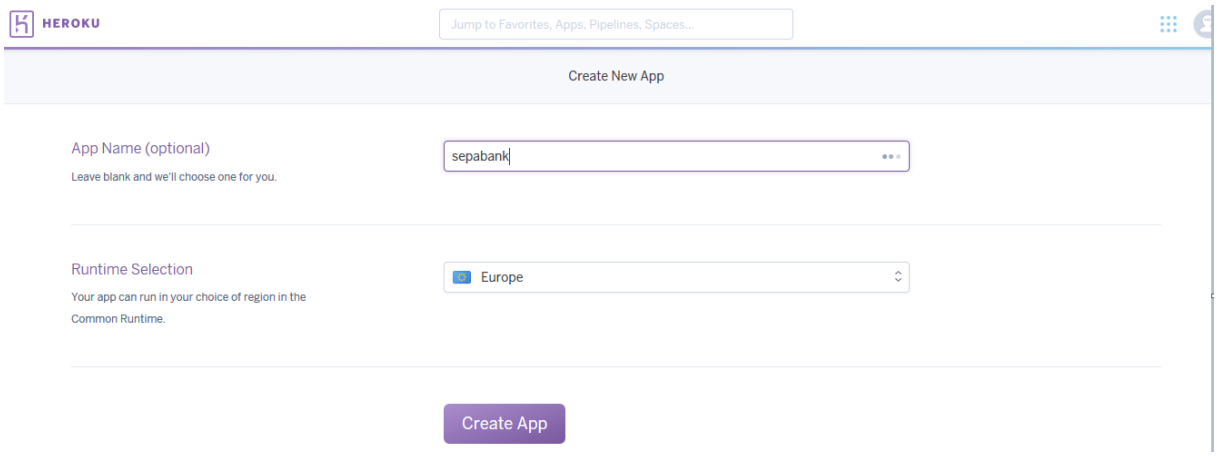
JdbcTemplate est une Class de Spring MVC qui nous permet de se connecter à notre base de données et d'exécuter les déférentes requêtes sql.

4. *Tutoriel de déploiement*

1. *Création d'une nouvelle application sur Heroku*

Après avoir créé un compte sur Heroku, on se connecte à notre compte pour ajouter une nouvelle application :





2. *Ajout d'une base de données*

1. On ajoute une nouvelle base de données à notre application avec les deux commandes :

```
$ heroku addons | grep -i POSTGRES --app nom_application
```

```
$ heroku addons:create heroku-postgresql:hobby-dev --app nom_application
```

2. On exécute le script SQL de notre base de données par la commande :

```
$ cat <chemin_fichier_sql> | heroku pg:psql --app nom_application
```

3. *Déploiement du fichier .war de l'application*

1. On génère le fichier .war de notre projet (Serveur) par la commande :

```
$ mvn compile war:war
```

2. On déploie le fichier .war généré par la commande :

```
$ heroku war:deploy <chemin_fichier_war> --app <nom_application>
```

3. Ajout d'un Dyno à l'application avec la commande :

```
$ heroku ps:scale web=1 --app <nom_application>
```