

Rapport Projet Langage Web Service REST

Realisé par

BENAOUICHA Mohand-Said et BABAGHAYOU Khaled

29 avril 2017

Sommaire

1. Objectifs du projet	3
2. Information Projet web	3
2.1. Les auteurs du projet :	3
2.2. Adresse du dépôt git :	3
2.3. Adresse du service REST :	3
3. Description du serveur	4
3.1. Adresse du service REST	4
3.2. Description précise des requêtes	4
3.3. Liste des technologies utilisées	7
3.3.1. Spring MVC	7
3.3.2. Persistance DAO	7
3.3.3. Base de données MySQL	8
3.3.3. Heroko	8
3.4. Tutoriel de déploiement	9
4. Description du client	10
4.1. Tutoriel d'installation et d'exécution du client	10
4.2. Mode d'emploi	10
4.2.1. Rechercher	11
4.2.2. Afficher les transactions résumées	11
4.2.3. Ajouter une transaction	11
4.2.4. Exemple de fichier XML pouvant être déposé sur le client	12
2. Conclusion	13
3. Reference Utilisé	13

1. Objectifs du projet

Les deux parties de notre projet seront :

1. Déployer un service REST permettant de gérer les transactions.
2. Créer une application client permettant d'envoyer et/ou de consommer une transaction du service REST.

2. Information Projet web

2.1. Les auteurs du projet :

- BENAOUICHA Mohand-Said

- BABAGHAYOU Khaled

2.2. Adresse du dépôt git :

https://github.com/s4id23/sepa_project.git

2.3. Adresse du service REST :

<https://sepa-rest-service.herokuapp.com/>

3. Description du serveur

3.1. Adresse du service REST

Depot git :

https://github.com/s4id23/sepa_project.git

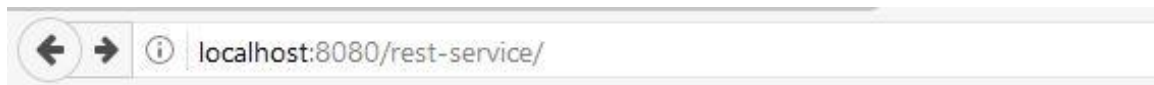
Lien heroku

<https://sepa-rest-service.herokuapp.com/>

3.2. Description précise des requêtes

3.2.1. La page d'Accueil

La page d'accueil contient des Informations générales sur le projet, elle contient la liste des auteurs du projet, la date de réalisation, et des liens détailler vers les différentes fonctionnalités du service REST.



Bienvenue au service de gestion des Transactions

vous pouvez :

- [Afficher la liste détailler des transactions](#)
- [Afficher une synthèse des transactions stockée](#)
- [Afficher sous forme résumée, la liste des transactions enregistrées](#)
- Afficher le contenu complet de la transaction dont l'identifiant est "n"
- Déposer une nouvelle transaction

Créer par : BENAOUICHA Mohand-Said & BABAGHAYOU Khaled

Le : 28/02/2017

3.2.2. Fonctionnalité /stats

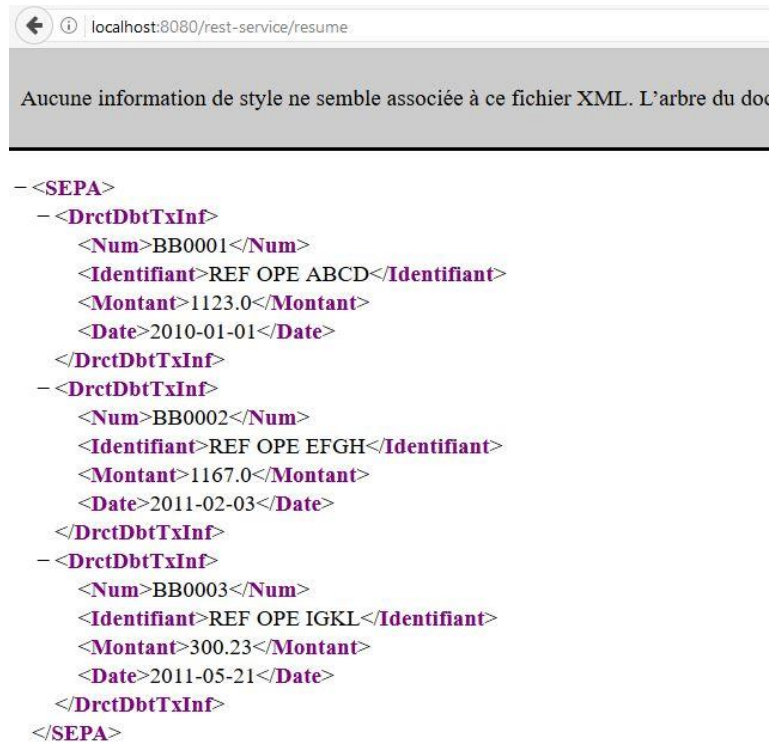
Récupère l'ensemble des transactions sous format XML, puis il affiche des statistiques sur les transactions, les informations affichées sont :

- **<MntTot>** Calcul le nombre de transactions dans la base de données
- **<NbrTrans>** La somme du montant des transactions




3.2.3. Fonctionnalité /resume

Affiche sous forme XML un résumé des transactions stockées dans la base de données, les principales informations affichées sont le numéro d'enregistrement de la transaction, l'identifiant « PmtId », le Montant de la transaction et la date de la transaction



3.2.4. Fonctionnalité /detail

Affiche sous format XML les transactions détaillées stockée dans la base de données




```

- <SEPA>
- <DrctDbtTxInf>
  <PmtId>REF OPE ABCD</PmtId>
  <InstAmt>1123.0</InstAmt>
- <DrctDbtTx>
  <MndtId>MANDAT NO 111111</MndtId>
  <DtOfSgntr>2010-01-01</DtOfSgntr>
</DrctDbtTx>
- <DbtrAgt>
  <BIC>ABNAFRPP</BIC>
</DbtrAgt>
- <Dbtr>
  <Nm>Mr Debiteur N1</Nm>
</Dbtr>
- <DbtrAcct>
  <IBAN>FR7630001007941234567890155</IBAN>
</DbtrAcct>
  <RmtInf>Facture N1</RmtInf>
</DrctDbtTxInf>

```

3.2.5. Fonctionnalité /trx/n

Affiche sous format XML le contenu détaillé de la transaction avec l'identifiant « PmtId » égale a « n »



```

- <DrctDbtTxInf>
  <PmtId>REF OPE ABCD</PmtId>
  <InstAmt>1123.0</InstAmt>
- <DrctDbtTx>
  <MndtId>MANDAT NO 111111</MndtId>
  <DtOfSgntr>2010-01-01</DtOfSgntr>
</DrctDbtTx>
- <DbtrAgt>
  <BIC>ABNAFRPP</BIC>
</DbtrAgt>
- <Dbtr>
  <Nm>Mr Debiteur N1</Nm>
</Dbtr>
- <DbtrAcct>
  <IBAN>FR7630001007941234567890155</IBAN>
</DbtrAcct>
  <RmtInf>Facture N1</RmtInf>
</DrctDbtTxInf>

```

3.2.6. Fonctionnalité /depot

Ajoute une nouvelle transaction dans la base de données, afin l'ajoute une vérification syntaxique est effectuer utilisant XSD afin d'autoriser que les dépôts corrects

3.3. Liste des technologies utilisées

3.3.1. Spring MVC

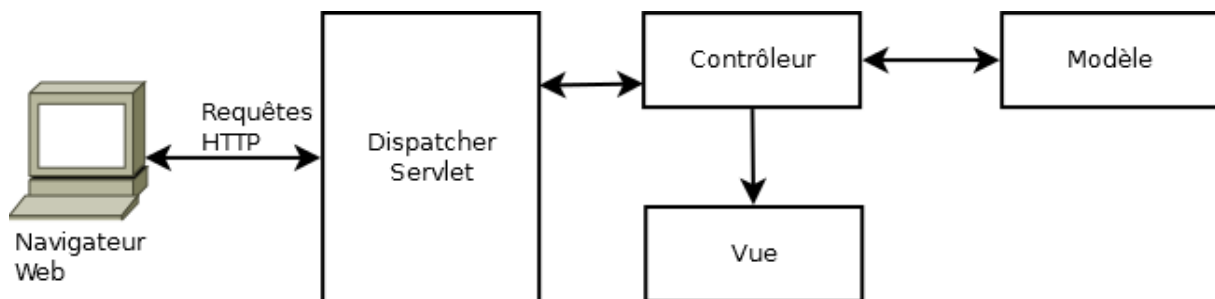


Figure 1 Architecture Spring

Le framework Spring permet d'organiser une application Web selon le patron de conception MVC. La figure ci-dessus (figure 1) présente l'architecture simple de ce patron de conception. Les différents éléments sont :

- **Dispatcher Servlet** : cette partie est fournie par Spring. La servlet reçoit les requêtes HTTP envoyées par le client et dirige la requête vers le contrôleur qui traite la requête et envoie le résultat à la vue .
- **Modèle** : il contient la partie « métier » de notre service. Contient une classe pour chaque type d'information, il sera implémenté par le pattern DAO (Data Access Object ou Objet d'accès aux données) ;
- **Vue** : c'est l'interface de l'application ou du service dans notre cas. Dans notre cas on a utilisé une page JSP (JavaServer Pages) ;
- **Contrôleur** : c'est le cerveau de notre service, il se charge du choix du traitement à déclencher selon les requêtes reçues et des informations à afficher en fonction des entrées.

3.3.2. Persistance DAO

Pour la persistance des données de notre service on a essayé de travailler avec le pattern DAO, Ce pattern permet de faire le lien entre la couche métier et la couche persistante, ceci afin de centraliser les mécanismes de mapping entre notre système de stockage et nos objets Java. Il permet aussi de prévenir un changement éventuel de système de stockage de données (MySQL par exemple).

3.3.3. Base de données MySQL

Pour stocker l'ensemble de nos transactions nous avons eu besoin d'une base de données, notre choix de base de données était la SGBD MySQL pour utiliser simplement le pattern DAO. Nous avons créé une base de données nommée « webproject » avec une seule table « transaction » qui contient toutes les informations d'une transaction.

Voici le code de création de la base de données

```
CREATE DATABASE webproject;

CREATE TABLE `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 ,
  BIC varchar(11) NOT NULL ,
  IBAN varchar(34) NOT NULL ,
  Nm varchar(35) NOT NULL ,
  DtOfSgntr varchar(10) NOT NULL ,
  RmtInf varchar(50) NOT NULL ,
  PRIMARY KEY (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=3
  DEFAULT CHARSET=utf8
  COLLATE=utf8_unicode_ci ;
```

3.3.3. Heroku

Comme suggéré dans l'énoncé du projet nous avons utilisé la plateforme PaaS Heroku. Heroku prend en charge le déploiement de fichiers WAR via le déploiement de Git (qui emploie le fichier WAR à distance) et via le plugin Heroku Maven (qui emploie le fichier WAR localement). Le serveur par défaut pour les deux méthodes est Tomcat 8 qui est parfait dans notre cas.

3.4. Tutoriel de déploiement

- En premier nous avons créé un compte héroku et nous avons créé une nouvelle application sepa-**rest-service**
- Nous avons générer un fichier WAR depuis notre application service disponible dans le git dans le dossier rest-service grace a la commande

\$mvn install

- On déploie le fichier .war dans heroku en utilisant la commande a partir du dossier rest-service :

\$ heroku deploy:war -war target/rest-service.war -app sepa-rest-service

1. Description du client

4.1. Tutoriel d'installation et d'exécution du client

Pour installer l'application client il faut suivre les étapes suivantes :

- 1) Le premier chose à faire pour et récupérer le projet sur notre dépôt git mentionner ci-dessus

```
git clone https://github.com/s4id23/sepa_project.git
```

- 2) Accéder au répertoire rest-client
- 3) Compiler les sources pour générer le Jar:
`$ mvn install`
- 4) Exécuter le fichier Client.jar généré.
Utiliser la commande `java -jar target/Client.jar`

4.2. Mode d'emploi

Notre application client est une application java avec une interface Swing,

L'application offre une manipulation simple est efficace des services de notre serveur REST, il contient des buttons pour chaque action, une vue qui affiche le résultat et un formulaire à gauche qui permet d'ajouter une nouvelle transaction.

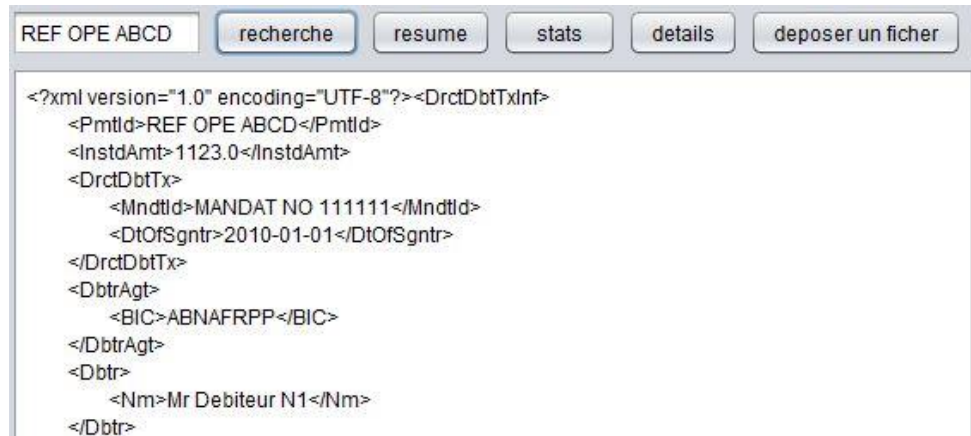


Nous allons décrire chaque fonctionnalité de cette application

4.2.1. Rechercher :

C'est fonction qui permet d'effectuer une recherche par Id en appelant la requête GET /trx/n, l'identifier in et récupérer à partir du textarea ID

L'exemple REF-OPE-ABCD ci-dessous.

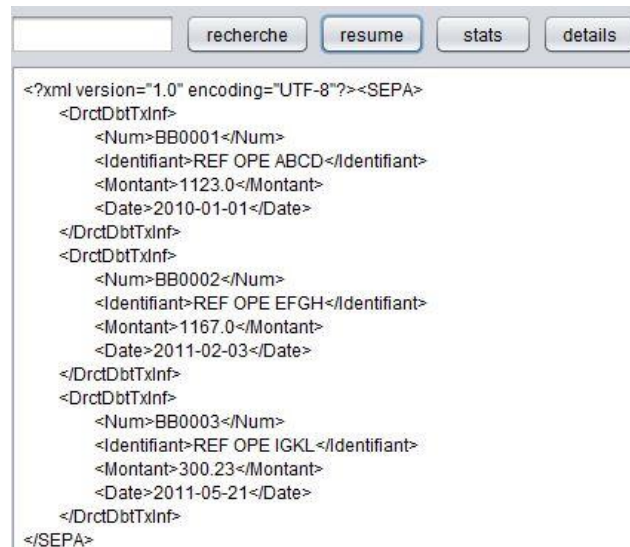


```

REF OPE ABCD  recherche  resume  stats  details  déposer un fichier

<?xml version="1.0" encoding="UTF-8"?><DrctDbtTxInf>
  <PmtId>REF OPE ABCD</PmtId>
  <InstdAmt>1123.0</InstdAmt>
  <DrctDbtTx>
    <MndtId>MANDAT NO 111111</MndtId>
    <DtOfSgntr>2010-01-01</DtOfSgntr>
  </DrctDbtTx>
  <DbtrAgt>
    <BIC>ABNAFRPP</BIC>
  </DbtrAgt>
  <Dbtr>
    <Nm>Mr Debiteur N1</Nm>
  </Dbtr>
</DrctDbtTxInf>
  
```

4.2.2. Afficher les transactions résumées



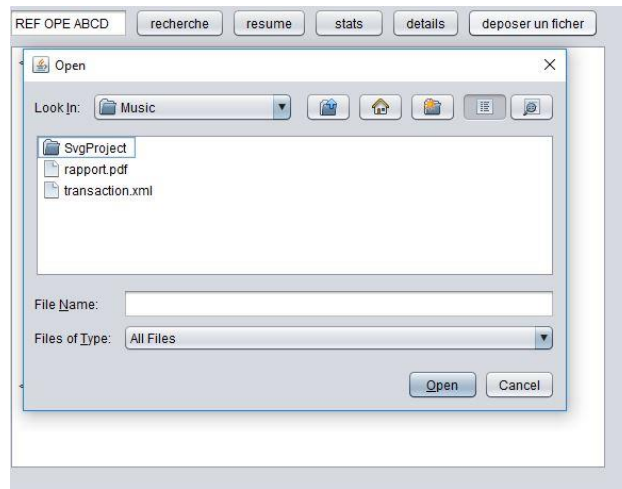
```

recherche  resume  stats  details

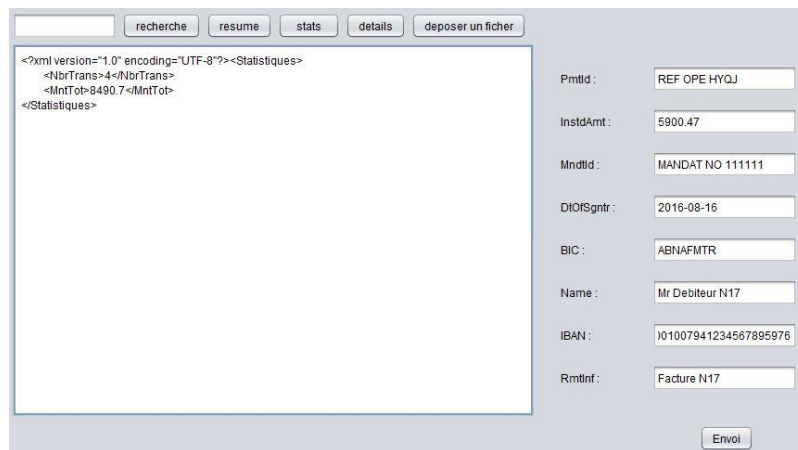
<?xml version="1.0" encoding="UTF-8"?><SEPA>
  <DrctDbtTxInf>
    <Num>BB0001</Num>
    <Identifiant>REF OPE ABCD</Identifiant>
    <Montant>1123.0</Montant>
    <Date>2010-01-01</Date>
  </DrctDbtTxInf>
  <DrctDbtTxInf>
    <Num>BB0002</Num>
    <Identifiant>REF OPE EFGH</Identifiant>
    <Montant>1167.0</Montant>
    <Date>2011-02-03</Date>
  </DrctDbtTxInf>
  <DrctDbtTxInf>
    <Num>BB0003</Num>
    <Identifiant>REF OPE IGKL</Identifiant>
    <Montant>300.23</Montant>
    <Date>2011-05-21</Date>
  </DrctDbtTxInf>
</SEPA>
  
```

4.2.3. Ajouter une transaction

Pour l'ajout d'une transaction, nous avons 2 méthode fonctionnelle, la première c'est l'ajout en utilisant un fichier XML, si le fichier ne respect le format XML un message d'erreur s'affiche.



La deuxième méthode c'est l'ajout en utilisant le formulaire à gauche, on va cliquer sur envoi qui va déclencher la fonctionnalités **/depot**



recherche resume stats details déposer un fichier

Look In: Music

SvgProject
rapport.pdf
transaction.xml

File Name:

Files of Type: All Files

Open Cancel

recherche resume stats details déposer un fichier

<?xml version="1.0" encoding="UTF-8"?><Statistiques>
<NbrTrans>4</NbrTrans>
<MntTot>8490.7</MntTot>
</Statistiques>

PmtId: REF OPE HYQJ

InstdAmt: 5900.47

MndtId: MANDAT NO 111111

DtOfSgntr: 2016-08-16

BIC: ABNAFMTR

Name: Mr Debiteur N17

IBAN: J01007941234567895976

RmtInf: Facture N17

Envoi

4.2.4. Exemple de fichier XML pouvant être déposé sur le client

```
<DrctDbtTxInf>
  <PmtId>REF OPE XXXX</PmtId>
  <InstdAmt>600.23</InstdAmt>
  <DrctDbtTx>
    <MndtRltdInf>
      <MndtId>MANDAT NO 000333</MndtId>
      <DtOfSgntr>2017-04-16</DtOfSgntr>
    </MndtRltdInf>
  </DrctDbtTx>
  <DbtrAgt>
    <FinInstnId>
      <BIC>ABNAFMTR</BIC>
    </FinInstnId>
  </DbtrAgt>
  <Dbtr>
    <Nm>Mr Debiteur N21</Nm>
  </Dbtr>
  <DbtrAcct>
    <Id> <IBAN>FR7630001001234567891234567</IBAN> </Id>
  </DbtrAcct>
  <RmtInf>Facture N21</RmtInf>
</DrctDbtTxInf>
```

2. Conclusion

Grace aux TP précédents et ce projet nous avons appris plusieurs nouvelles technologies qui représente les technologie web les plus utilisé, nous avons appris à gérer un service REST, utiliser des plateformes PaaS. Nous espérons que ces nouvelles connaissances nous aident à s'intégrer facilement dans notre vie professionnelle.

3. Reference Utilisé

- Tutorial Sur Framework Spring
<http://rpouiller.developpez.com/tutoriels/spring/application-web-spring-hibernate/>
- <https://devcenter.heroku.com/articles/getting-started-with-spring-mvc-hibernate>
- <https://devcenter.heroku.com/articles/war-deployment>