



# Formation Architecture Logicielle

## *Exercices*

Formateurs :

Frantz Degrigny, [frantz.degrigny@keyconsulting.fr](mailto:frantz.degrigny@keyconsulting.fr)

Méric Garcia, [meric.garcia@keyconsulting.fr](mailto:meric.garcia@keyconsulting.fr)

Téléphone : 02.30.96.02.75

<http://www.keyconsulting.fr>

## Sommaire

2.	INTERAGIR AVEC LE SI	3
1.	JMS - PREMIER TESTS.	3
1.	<i>Installation</i>	3
2.	<i>Ecriture et lecture d'un message :</i>	3
2.	JMS - PASSAGE D'UN OBJET.	4
1.	<i>Développement</i>	4
3.	APPLICATION SOA : WS ET JMS	5
4.	EXPOSER UN WEBSERVICE DANS UN ANNUAIRE UDDI.	6
1.	<i>Installation</i>	6
2.	<i>Ajout d'un service</i>	6
3.	<i>Enregistrement du service dans l'annuaire</i>	7
1.	WOA : EXPOSER UNE API REST VIA UN PAAS.	9
1.	<i>Restlet et apiSpark</i>	9
2.	<i>Creation d'une API et d'une base</i>	9
3.	<i>Ajouter un utilisateur</i>	12
4.	<i>Test</i>	13
5.	<i>Pour aller plus loin.</i>	13

## Table des illustrations

## 2. Interagir avec le SI

### 1. JMS - premier tests.

#### 1. Installation

Récupérer apache ActiveMQ :

<http://activemq.apache.org/activemq-5120-release.html>

Décompresser l'archive.

Démarrer le service : en ligne de commande à la racine du dossier :

⇒ `./bin/activemq qrtart`

Vérifier que le service est bien démarré : <http://127.0.0.1:8161/admin/>

Login : admin, Mdp : admin

#### 2. Ecriture et lecture d'un message :

La classe `JmsServiceHelper` permet d'accès aux méthode d'enregistrement et de lecture des files JMS.

*`public void send(String text)`* : permet d'envoyer un message sous forme de texte dans la première file

*`public void sendForListener(String text)`* : permet d'envoyer un message sous forme de texte dans la file sur laquelle écoute un listener

*`public String next()`* : récupère le message suivant disponible dans la première file.

Dans un premier temps, nous allons envoyer la liste des auteurs de calculs dans la file JMS. Au prochain redémarrage de l'application nous récupérerons cette liste et l'afficherons.

Les deux fonctions à modifier se trouve dans le contrôleur (classe `Controller`) :

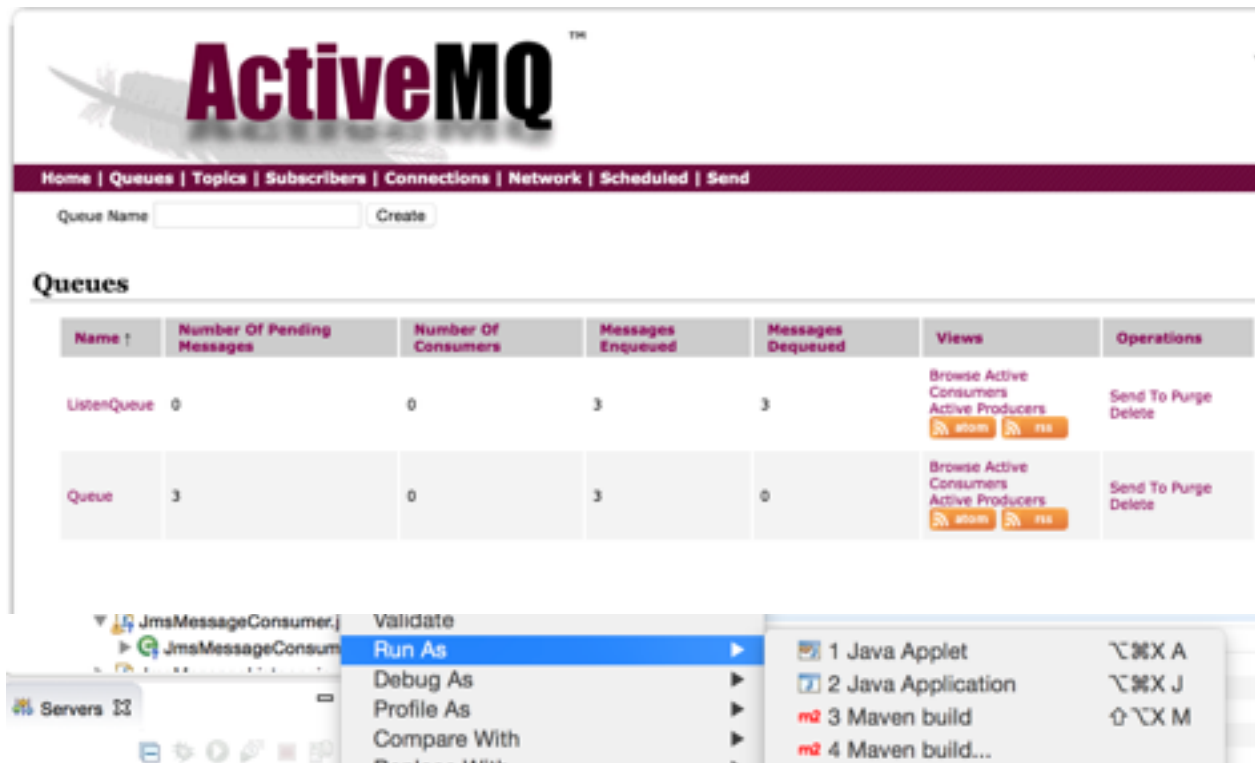
*`private void addEnQueuedMessageTo(StringJoiner sj)`*

et

*`private void sendAuthorToJmsServices(Calcul calcul)`*

Une fois les modifications effectuées, lancer le programme :

Si le développement est correct, le résultat sera le suivant après trois calculs effectués :



Que s'est-il passé ? Quelle la différence entre le listener et le consumer (classe JmsMessageListener et JmsMessageConsumer)?

## 2. JMS - passage d'un objet.

### 1. Développement

Des files JMS ont été ajoutée dans la configuration afin de pouvoir gérer les calculs.

Il faut désormais modifier trois classes afin d'utiliser ses files :

JmsMessageSender : méthode `send(Calcul calcul)`

JmsMessageConsumer : méthode `getFollowingCalcul()`

Controller : méthode `sendCalculAndAuthorToJmsServices(Calcul calcul)` et `addEnQueuedCalculToList(List<Calcul> calculs)`

Les lignes à modifier sont indiquées par des commentaires.

Une fois les modifications effectuées, lancer l'application comme dans l'exercice précédent et observer l'évolutions des files JMS.

La liste des calculs de la session précédente doit se recharger à chaque redémarrage de l'application (la file JMS devient un moyen de persister temporairement les calculs).

### 3. Application SOA : WS et JMS

Se placer au point de départ : `git checkout -f SOA1`

Pour démarrer les webservices, il faut se placer dans le répertoire `java-ws` et lancer « `mvn clean install` » puis « `mvn cargo:run` ».

Les webservices sont alors disponible ici : `http://localhost:8080/websevice/services`

Nous allons modifier les webservices. Ils permettront de faire abstraction entre l'interface graphique et la file JMS. Nous allons donc ré-implémenter deux méthodes :

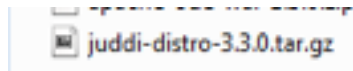
`getAll` et `addAll` pour qu'elles utilisent la file JMS plutôt que le module de persistance. Ces méthodes sont présentes dans le `CalculService`. Pour cela réutiliser le `JMSServiceHelper`.

Modifier ensuite le controller afin d'utiliser le webservice lors de l'initialisation (mise à jour du tableaux des calculs) et lorsque l'on lance un calcul (enregistrement via les webservices).

## 4. Exposer un webservice dans un annuaire UDDI.

### 1. Installation

Décompresser le dossier juddi-distro-3.3.0.tar.gz.



Copier votre JDK 7 dans juddi-distro-3.3.0\juddi-tomcat-3.3.0 avec comme nom : jdk7-win sous windows et jdk7-lin sous linux.

Ajouter le fichier setenv.sh ou setenv.bat disponible dans le github.

Exécuter la commande bin/setenv.sh ou bin/setenv.bat.

Démarrer en ligne de commande : bin\startup.bat ou bin/startup.sh.

L'annuaire de service est lancé ! S'y connecter : <http://localhost:8080/juddi-gui/home.jsp>.

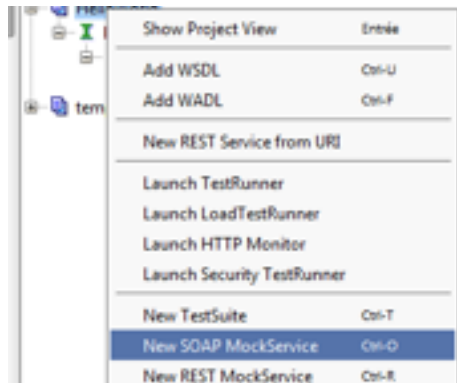
### 2. Ajout d'un service

Nous allons enregistrer un service généré par un mock SoapUI dans l'annuaire.

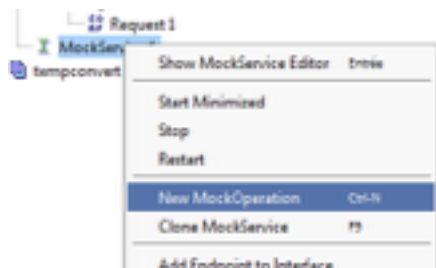
Démarrer un mock SoapUi généré à partir du WSDL HelloWorld.wsdl.

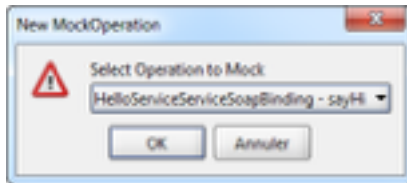
Pour cela créer un projet SoapUI en important cette WSDL :

⇒ Créer un nouveau MockService

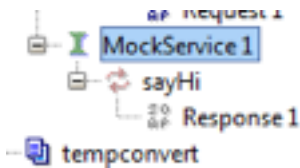


⇒ Ajouter une mock opération :

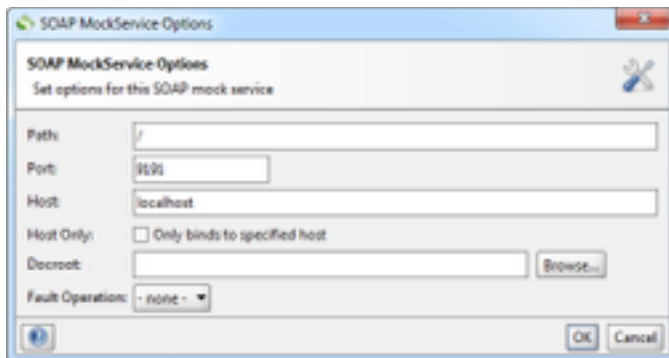
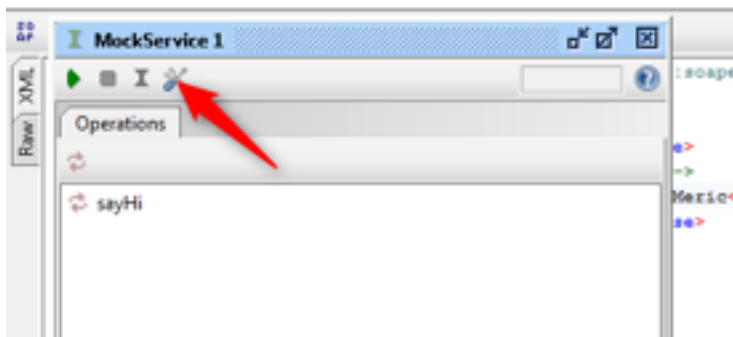




⇒ Double cliquer sur MockService1



⇒ Modifier les propriétés sur le service



⇒ Le lancer (flèche verte).

⇒ Effectuer une requête pour vérifier que le mock est UP : <http://localhost:9191/web-service/services/HelloWorld>

Nous allons pouvoir enregistrer ce webservice dans l'annuaire. Sa WSDL est exposée ici :

<http://localhost:9191/web-service/services/HelloWorld?wsdl>

### 3. Enregistrement du service dans l'annuaire

L'interface d'admin se trouve ici : <http://localhost:8080/juddi-gui/home.jsp>.

Se logger en tant qu'admin (admin - admin).

Aller dans create -> Register Services from WSDL.

Enregistrer la wsdl.

Rechercher la dans discover -> tModels.

Quelles informations peut on récupérer ?

Quel est l'intérêt de stocker les services dans un annuaire ?



## 1. WOA : exposer une API Rest via un PAAS.



### Create a web API

#### 1. Restlet et a

Aller sur le site : [ht](http://apispark.net)

Se créer un compte

Form for creating a web API:

- Type:** Full stack
- Name:** KCOapi
- Domain:** kcoapi.apispark.net (with a green checkmark)
- Description:** (empty text area)
- Buttons:** Cancel, Add

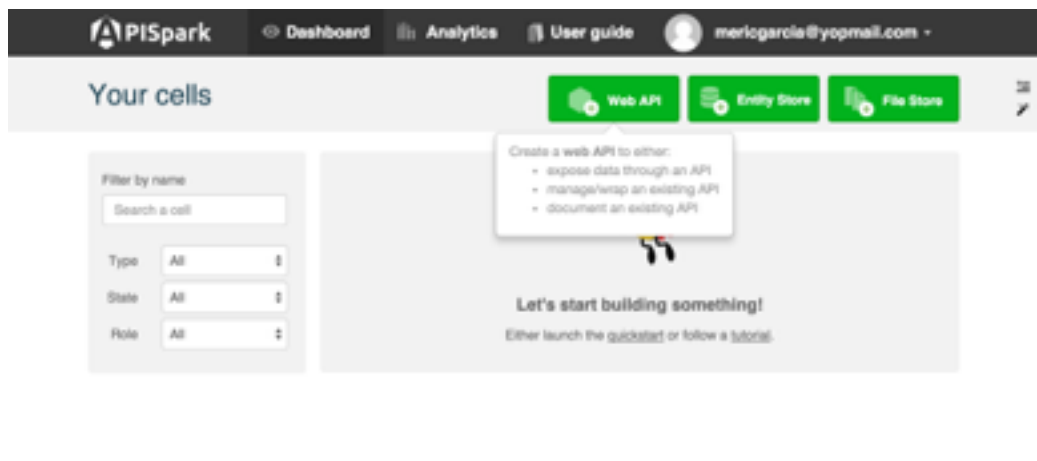
(compte github).



Vous aller recevoir un mail de validation. Cliquer sur le lien.

## 2. Creation d'une API et d'une base

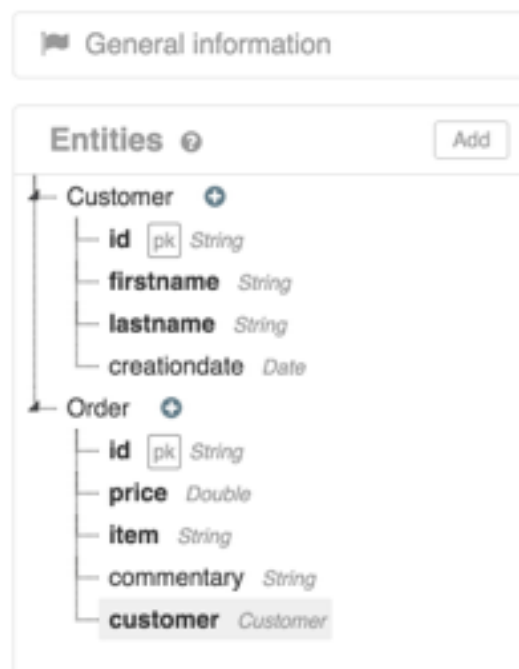
Nous allons créer une première API.



Créons également une « entity base ».

The screenshot shows a web application interface. At the top, there are tabs: 'Overview', 'Backups', 'Messages', and 'Members'. Below these, there is a 'General information' tab selected. To the right of the 'General information' tab, there are sub-tabs: 'Overview', 'User agreement', and 'Privacy policy'. On the left side, there is a box titled 'Entities' with an 'Add' button. Inside this box, it says '...No entities...'. On the right side, there is a form titled 'INFORMATION' with fields for 'Name', 'Description', 'Keywords', and 'License URL'. The 'Name' field is filled with 'KDDstore'.

Créer le schéma suivant :



Ajoutons quelques données de test en utilisant la vue browser.

The screenshot shows the 'Browser' view of the API Spark interface. A notification at the top states: 'The data browser matches the latest deployed version of your store.' On the left, a sidebar shows 'General information' and 'Entities' with a tree view containing 'Customer' and 'Order'. The main area displays a table with columns: 'id', 'firstname', 'lastname', and 'creationdate'. Below the table, there are navigation controls: 'Overview', 'Browser', 'Add', 'Edit', 'Delete', and a settings icon. The table shows one item with the following data:

id	firstname	lastname	creationdate
7c504f00-7b31-11e...			

Below the table, it says '1 - 1 of 1 items' and includes pagination controls.

Et déployer la base :

The screenshot shows the API Spark dashboard. At the top, there's a navigation bar with 'Dashboard', 'Analytics', and 'User guide'. A notification states: 'The cell is not deployed. Remember that a cell has to be deployed in order to be functional'. Below this, the 'KCOstore' entity is shown with a 'Deploy' button highlighted by a green circle. The dashboard also includes tabs for 'Overview', 'Backups', 'Messages', and 'Members'. On the left, there's a sidebar with 'General information' and 'Entities'.

Retourner dans l'API et lier la base de données à celle ci.

## 3. Ajouter un utilisateur

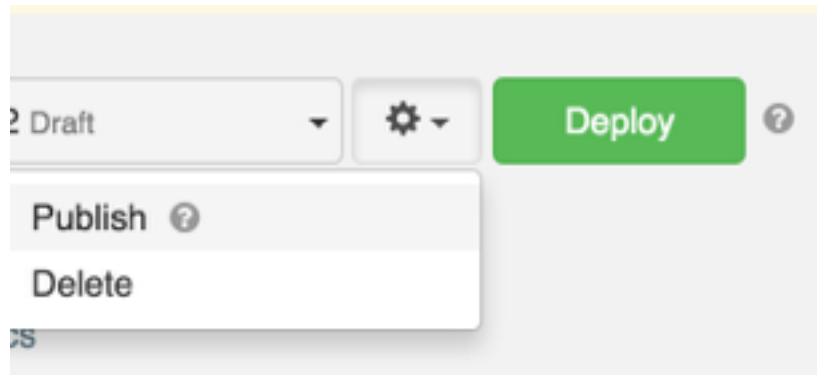
The screenshot shows the KCOapi interface. At the top, there's a sidebar with 'Overview' and 'General' tabs. The main area is titled 'Import a cell'. Below this, there's a 'Type' dropdown set to 'Entity store' and a 'Cell' dropdown set to 'KCOstore'. To the right, there's a 'Version 1 Draft' label and a green 'Deploy' button. Below the main area, there's a 'Members' section with a list of members and groups. The 'Members' list shows 'mericgarcia@yopmail.com' as the owner. The 'Groups' list shows 'Consumers' and 'Testers'.

The 'Add a member' dialog box is shown. It has fields for 'Member' (set to 'New app account'), 'First name' (set to 'meric'), 'Last name' (set to 'garcia'), 'Email' (set to 'mericgarcia@yopmail.com'), 'Username' (set to 'mericgarcia'), and 'Secret key' (masked with dots). There are 'Cancel' and 'Add' buttons at the bottom.

Lui affecter le rôle consommateur.

The screenshot shows the 'TOKENS' and 'GROUPS' sections. The 'TOKENS' section has fields for 'Username' (set to 'mericgarciatest') and 'Secret key' (set to 'mericgarciatest'). The 'GROUPS' section has a dropdown set to 'Consumers' and 'Add' and 'Cancel' buttons.

Nous allons ensuite publier et déployer l'API.



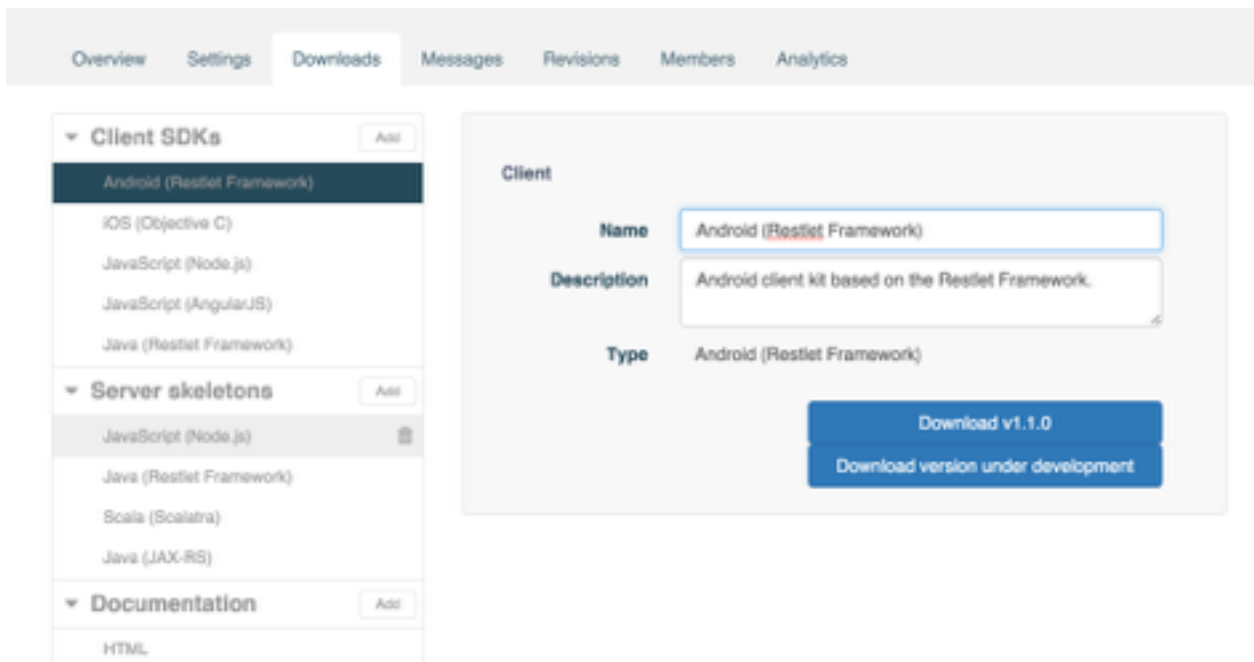
#### 4. Test

Utiliser ce compte pour effectuer une requête via un navigateur.



#### 5. Pour aller plus loin.

Reslet permet également de récupérer le projet généré afin de l'héberger en local.



Télécharger le serveur (JAVA ou javascript si vous avez un serveur nodeJS sur votre poste).

Pour builder et démarrer le projet récupérer :

JAVA :

`mvn clean install`

puis : `java -jar target/....jar`

NodeJS :

`npm install`

puis : `node index`

L'application est déployée en local. En regardant de plus près les sources générées, on s'aperçoit que seules l'API et créées et que les différentes méthodes sont à implémenter.