Atelier 6
Equilibrer la charge de l'application grâce à Ribbon

Ribbon est un **équilibreur de charge côté client**. Une fois installé, il va pouvoir aller consulter la liste des instances disponibles pour un Microservice pour les choisir à tour de rôle afin d'équilibrer la charge.

Pour tester Ribbon, nous devons tout d'abord lancer plusieurs instances du *Microservice-produits* qui feront l'objet de nos tests.

1. Lancer plusieurs instances d'un Microservice avec IntelliJ

Pour lancer différentes instances d'un Microservice, vous pouvez soit utiliser Docker, soit les lancer sur différents ports directement depuis IntelliJ.

Afin de garder notre application regroupée dans notre IDE et de pouvoir jouer avec les instances facilement, nous allons opter pour la deuxième méthode.

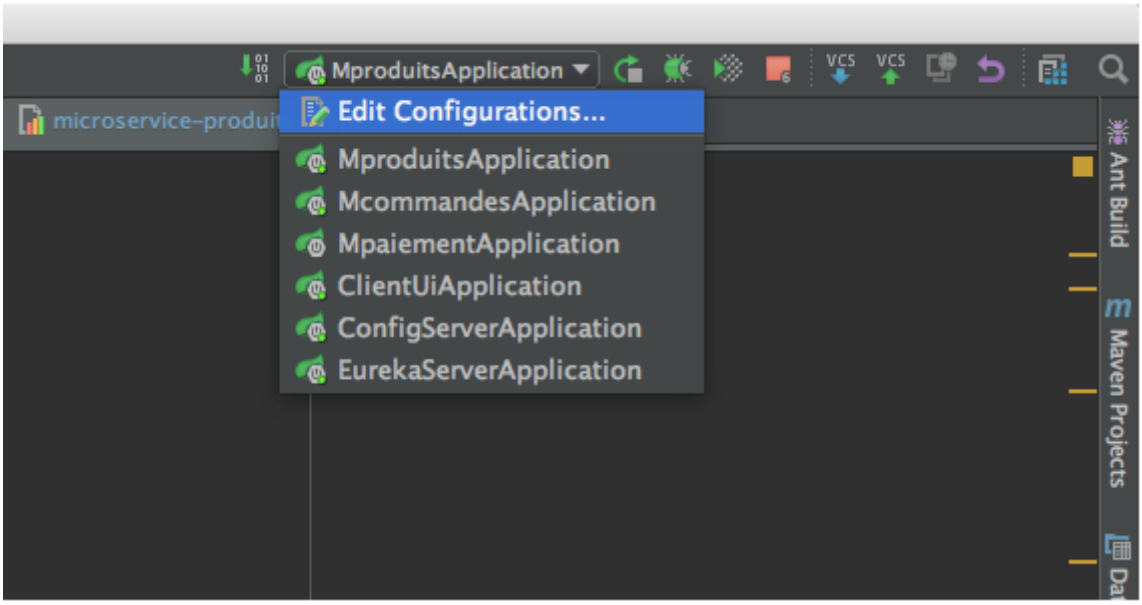
Pour lancer différentes instances du Microservice-produits sur différents ports, nous allons commencer par **supprimer** ***server.port 9001*** de microservices-produits.properties dans notre GIT.

microservices-produits.properties

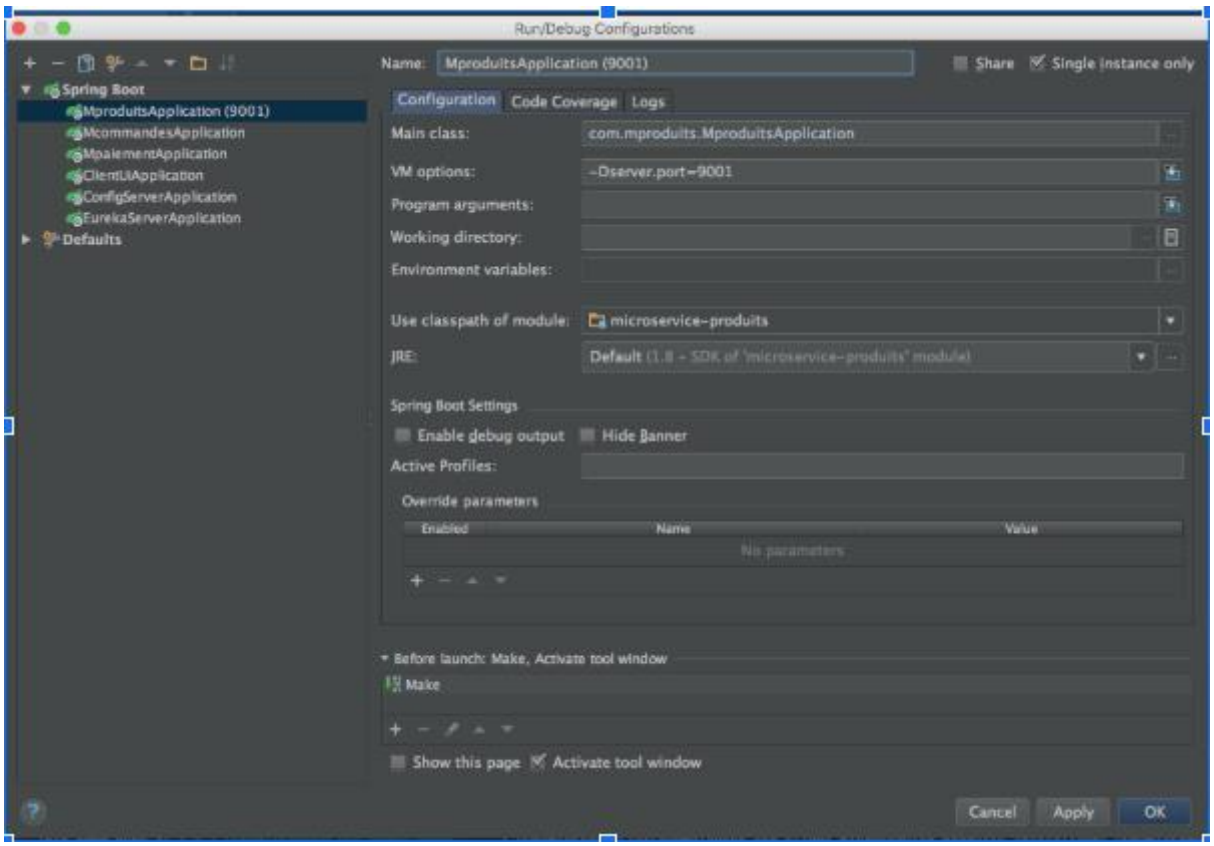
```
#Configurations H2
spring.jpa.show-sql=true
spring.h2.console.enabled=true

#défini l'encodage pour data.sql
spring.datasource.sql-script-encoding=UTF-8


#Nos configurations

mes-configs.limitDeProduits= 3

#Eureka
eureka.client.serviceUrl.defaultZone: http://localhost:9102/eureka/
```

Nous allons ensuite décider du port de chaque instance en fournissant directement celui-ci en argument à la VM.

Allez dans la liste des Microservices en haut à droite puis cliquez sur "Edit Configuration" :

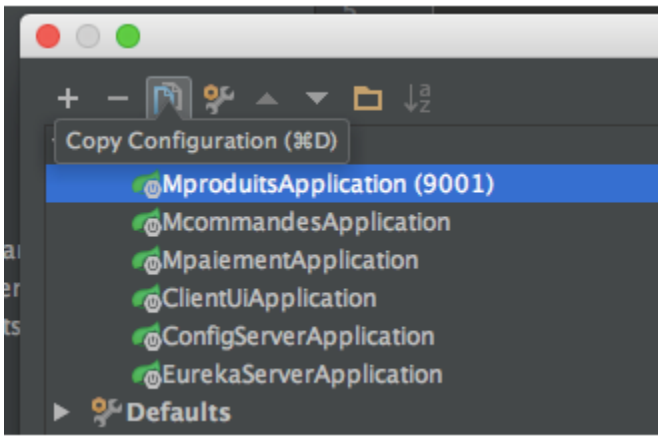


Cliquez sur le Microservice-produits puis ajoutez à son nom (9001) afin de le différencier. Ajoutez ensuite l'argument *-Dserver.port=9001* dans le champ VM options:

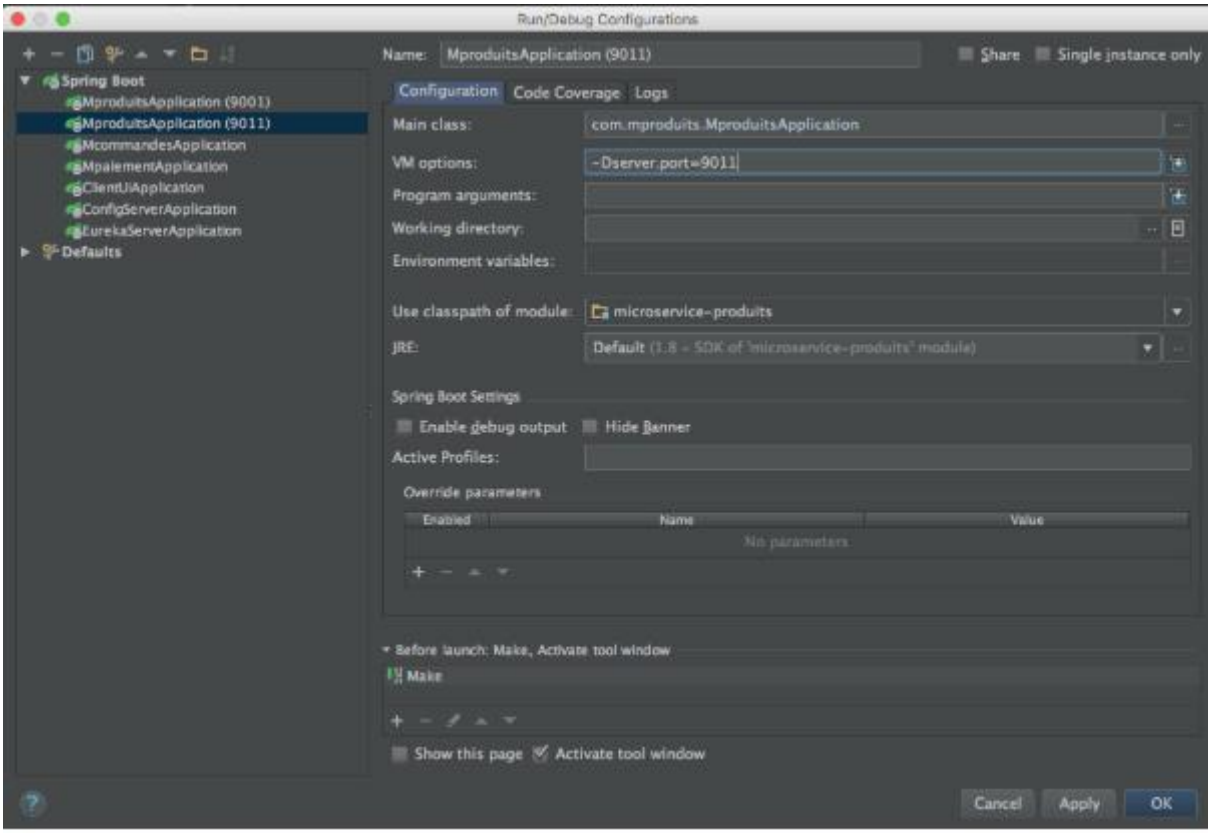


Ce Microservice tournera alors sur le port 9001.

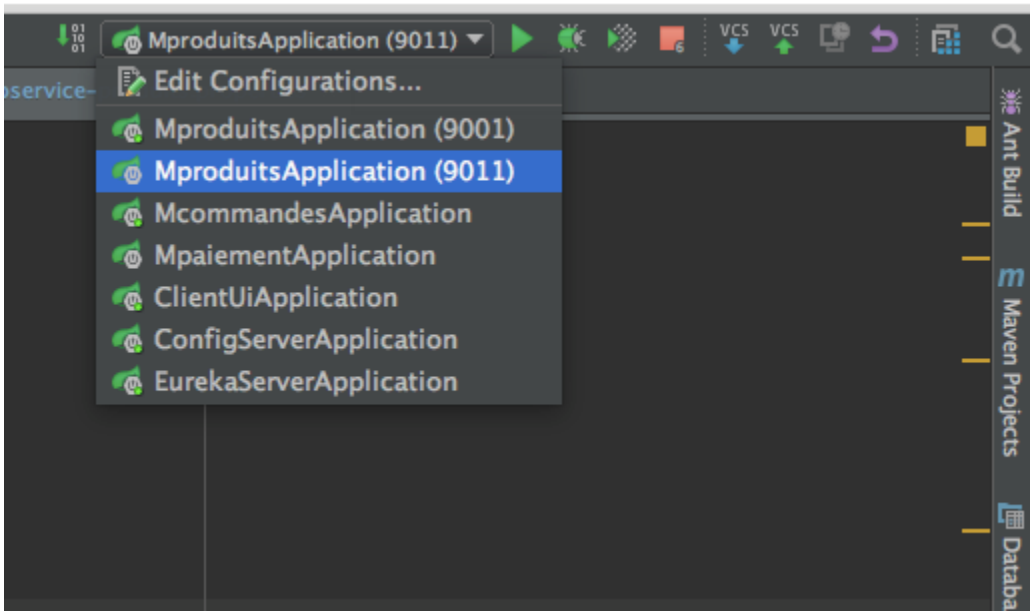
Dupliquez maintenant la configuration de ce Microservice grâce au bouton "Copy Configuration" juste au-dessus, ou via `Cmd`+`D` ou `Ctrl`+`D` :



Dans la nouvelle copie de configuration de votre Microservice, changez encore le nom en y ajoutant le port (9011) ; idem pour VM Options :

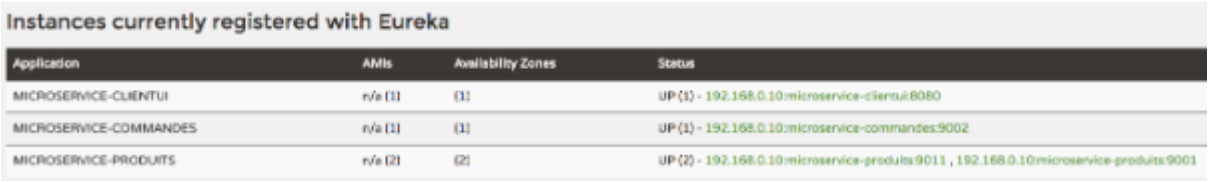


Très bien ! Vous avez maintenant 2 configurations de lancement de votre Microservice. Fermez la fenêtre et rendez-vous en haut à droite pour lancer vos 2 instances :



Assurez-vous qu'Eureka et config-server sont bien lancés avant.

Rendez-vous à l'URL d'Eureka, vous verrez apparaître les 2 instances lancées de Microservice-produits dans la colonne **"Status" :**



2. Ajoutez Ribbon

Nous allons maintenant ajouter Ribbon à notre client. La procédure est un peu la même que pour Eureka.

Ajoutez cette dépendance dans le *pom.xml* et actualisez Maven :

```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
</dependency>
```

Modifiez ***MicroserviceProduitsProxy*** comme suit :

```
package com.clientui.proxies;

import com.clientui.beans.ProductBean;
import org.springframework.cloud.netflix.ribbon.RibbonClient;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

import java.util.List;
import java.util.Optional;
```

```java
@FeignClient(name = "microservice-produits")
@RibbonClient(name = "microservice-produits")
public interface MicroserviceProduitsProxy {

  @GetMapping(value = "/Produits")
  List<ProductBean> listeDesProduits();

  /*
  * Notez ici la notation @PathVariable("id") qui est différente de celle qu'on utlise dans le contrôleur
  **/
  @GetMapping( value = "/Produits/{id}")
  ProductBean recupererUnProduit(@PathVariable("id") int id);

}
```

Explications :

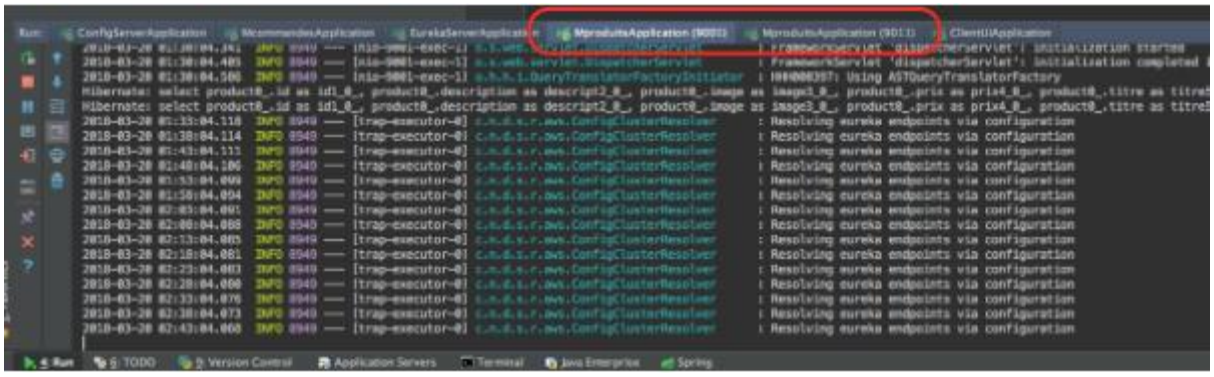
- Nous commençons par supprimer l'argument *URL* qui indiquait à Feign l'adresse de Microservice-produits.
- On ajoute `@RibbonClient(name = "microservice-produits")` qui va demander à Ribbon d'aller chercher automatiquement la liste des URL (et donc instances) de Microservice-produits disponibles.

Très bien ! Maintenant, nous allons **indiquer à Ribbon la liste de toutes ces URL** de Microservice-produits. Pour ce faire, rendez-vous dans le **fichier de configuration du client dans le GIT** et ajoutez ceci :

```
#Ribbon
microservice-produits.ribbon.listOfServers=localhost:9001,localhost:9011
```

Votre client Ribbon est prêt et il a tout ce qu'il faut pour alterner entre les 2 URL fournies quand vous faites appel au Microservice-produits.

Pour vérifier l'activité des deux instances de nos Microservices, rendez-vous à **l'onglet "Run" en bas d'IntelliJ :**



L'onglet run

Lancez maintenant tous les Microservices et rendez-vous sur la page d'accueil de notre application http://localhost:8080/

Actualisez une première fois la page et revenez dans la console. Vérifiez les 2 instances ; vous devriez voir, dans une des deux, une **requête qui s'est ajoutée** en bas, de type :

```
Hibernate: select product0_.id as id1_0_, product0_.description as
descript2_0_, product0_.image as image3_0_, product0_.prix as prix4_0_,
product0_.titre as titre5_0_ from product product0_
```

Cette requête est celle générée par cette instance de Microservice-produits pour récupérer la liste des produits dans la base de données.

Si vous réactualisez plusieurs fois, vous verrez à chaque fois cette requête s'afficher alternativement dans l'instance (9001) et (9011), ce qui vous confirme que les 2 instances sont appelées à tour de rôle par le client grâce à Ribbon. Celui-ci exécute un algorithme afin de distribuer les requêtes alternativement sur toutes les instances disponibles, équilibrant ainsi la charge.

Cela ne serait-il pas mieux de lui demander d'aller les chercher dans le registre d'Eureka plutôt que de les noter en dur dans le fichier de configuration ?

Tout à fait, et c'est là que la magie de Spring Cloud opère. Ribbon et Eureka fonctionnent ensemble sans besoin d'ajouter de configuration. Il suffit de supprimer la ligne où sont définies les URL pour que Ribbon contacte Eureka pour avoir la liste des instances.

Supprimez alors cette ligne :

```
#Ribbon
microservice-produits.ribbon.listOfServers=localhost:9001,localhost:9011
```

Redémarrez vos Microservices et vous verrez que la charge est bien équilibrée entre les instances. Vous pouvez vous amuser à ajouter une 3e instance pour vérifier cela.