INSA de Toulouse
guermouc@insa-toulouse.fr

Instanciation de microservice et répartition de charges (load balancing)

Précédemment, vous avez créé des microservices découvrables via Eureka. Ce qu'il faut savoir, c'est quand vous lancez votre application Spring Boot, vous avez à chaque fois une seule instance de votre microservice (singleton). L'avantage de l'architecture microservice est la possibilité de déployer plusieurs instances des microservices hautement sollicités par exemple pour absorber la charge. Ce que vous allez voir maintenant c'est comment créer plusieurs instances de vos microservices et comment réaliser la répartition des charges automatiquement.

a. Compétences visées

Le but est de mettre en place la répartition de charge (load balancing) en utilisant Spring Cloud.

b. Création de multiples instances d'un microservice :

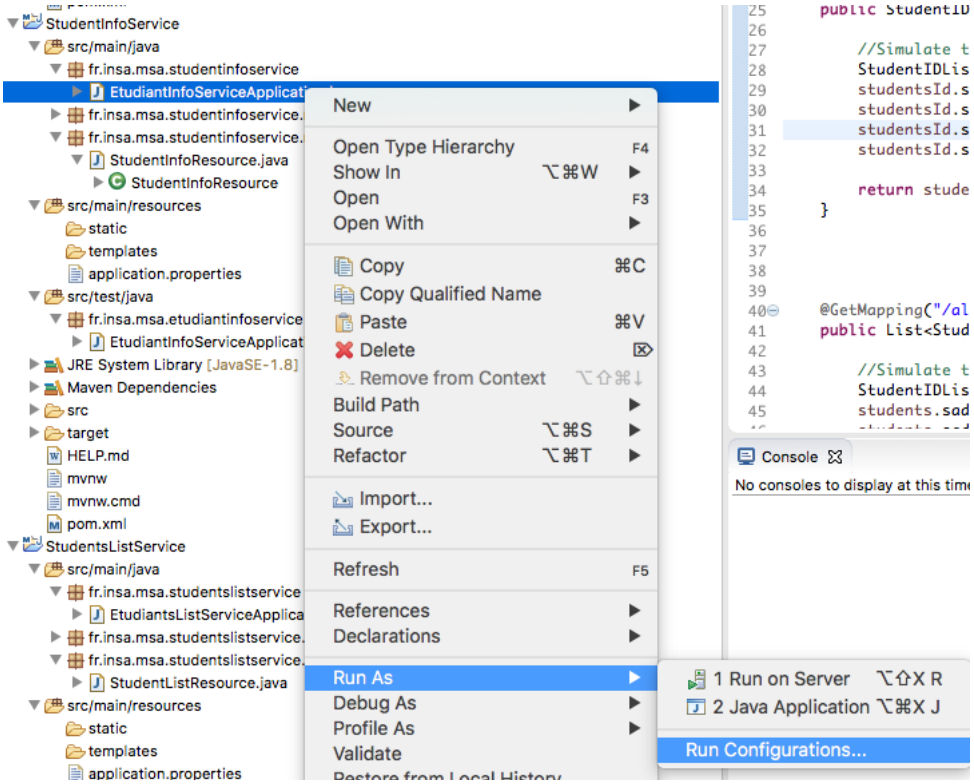
Vous allez utiliser Eclipse pour créer plusieurs instances de l'un de vos microservices. Par exemple, ici il s'agit du microservice *studentInfoService*. Sachez qu'on peut se passer d'Ecplise en créant un .jar puis le lancer en ligne de commande sur le serveur en précisant le numéro de port.

Dans le code source du microservice, ajoutez une méthode qui affiche un message. Ça va servir juste à savoir quelle instance a été appelée :

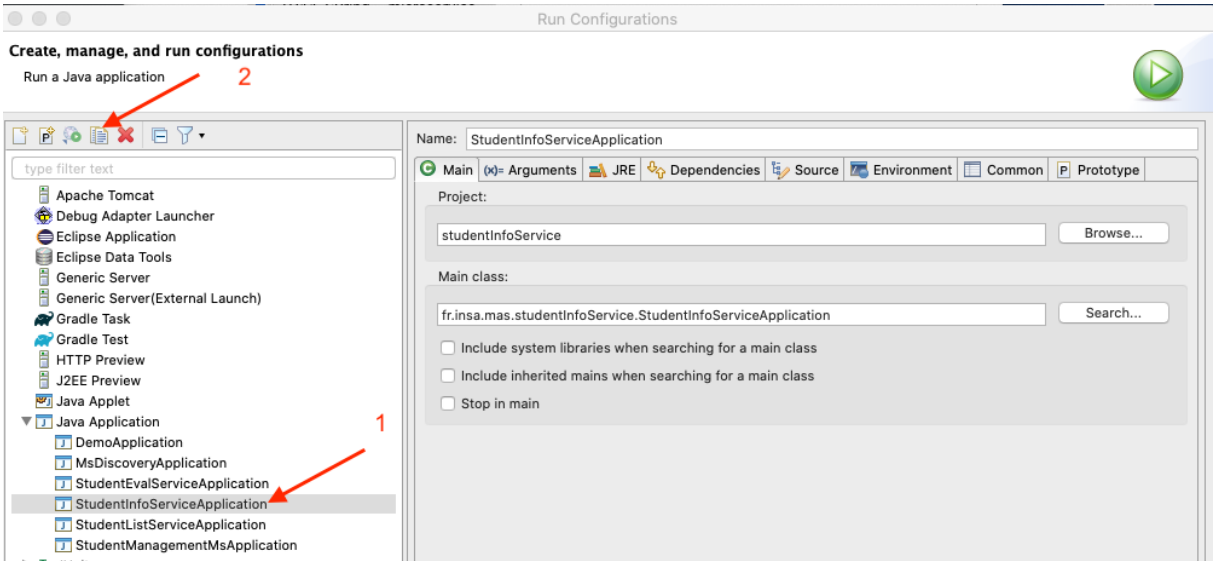
```
package fr.insa.msa.studentinfoservice.ressources;

import java.util.Arrays;
import java.util.List;


import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


import fr.insa.msa.studentinfoservice.model.StudentInfos;


@RestController
@RequestMapping("/student")
public class StudentInfoResource {


    @GetMapping("/{idStudent}")
    public StudentInfos getInfoEtudiant(@PathVariable("idStudent") int id){
        //Simulate the DB with a list
        List<StudentInfos> etudInfos=Arrays.asList(
                new StudentInfos(0,"Godart","Noemie","12/12/1992"),
                new StudentInfos(1,"Perrin","Ania","10/02/1993"),
                new StudentInfos(2,"Azi","Sana","22/05/1992"),
                new StudentInfos(3,"Yala","Nelia","12/06/1994")
                );
        System.out.println("Called!");
        //Get the student that corresponds to the id
        return etudInfos.get(id);
    }
}
```

INSA de Toulouse
guermouc@insa-toulouse.fr

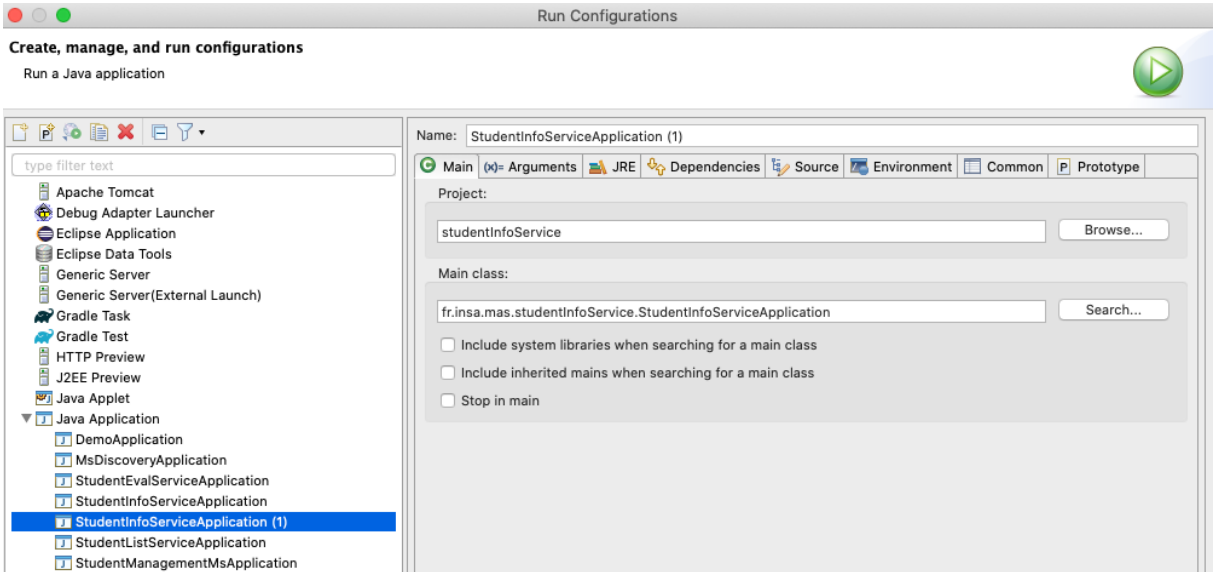
Vous allez créer une configuration d'exécution à laquelle vous associez le port 8181. Pour faire cela, clic droit sur la classe principale, puis *Run Configurations* ...



Sélectionnez la configuration d'exécution correspondant au microservice choisi (voir 1 dans la figure suivante). Puis dupliquez cette configuration (voir 2 dans la figure suivante).

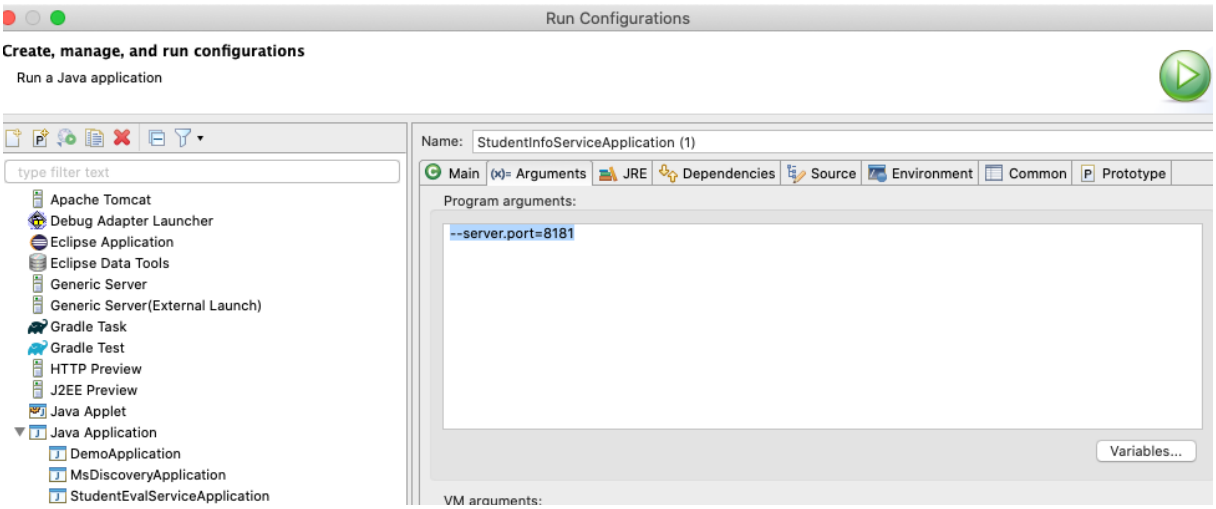


Une 2ème configuration vous est créée (nommée ici *StudentInfoServiceApplication (1)*) :



A droite, allez dans *Arguments* puis saisissez la ligne suivante : **--server.port=8181**

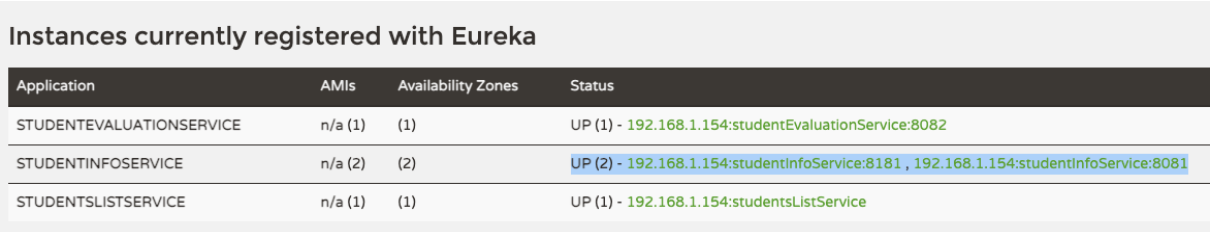
En exécutant votre classe via cette configuration, une instance sera lancée sur le serveur 8181.



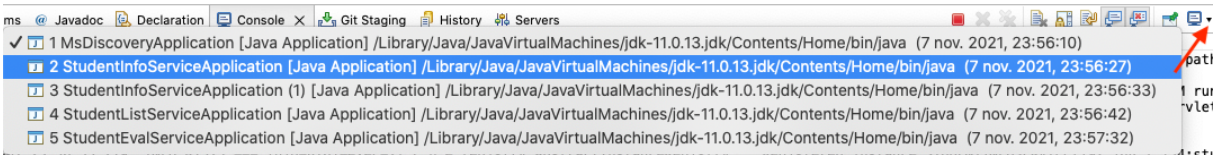
Quant à la première configuration (*StudentInfoServiceApplication),* on va pouvoir lancer une instance sur le port spécifié dans le fichier *application.properties*. Donc, pour chaque nouvelle instance, on a besoin de créer une configuration dédiée.

Exécutez votre application puis actualisez votre serveur Eureka. Vous allez constater deux instances de votre microservice *studentInfoService* : une instance sur le port 8081 et l'autre sur le port 8181.

INSA de Toulouse
guermouc@insa-toulouse.fr



Testez votre application en appelant le service *studentListService* qui lui appelle les deux autres services y compris *studentInfoService*. A chaque appel, vérifiez la console des deux instances.



Vous allez constater que les requêtes ont été réparties équitablement sur les deux instances (la 1ère instance a été appelée 2 fois et la 2ème 2 fois).