Construcción del *Service Discovery*

Aunque en el post *Hello World Eureka* ya expliqué de forma detallada cómo implementar un servicio de registro y descubrimiento con *Eureka*, resumo los pasos fundamentales:

- **Paso 1**: Crea un nuevo proyecto maven. Puedes nombrarlo como *ServiceDiscovery*
- **Paso 2**: Configura el fichero *pom.xml* del proyecto. Aquí tienes un ejemplo
- **Paso 3**: Crea la clase principal de la aplicación Spring Boot, añadiendo la anotación *@EnableEurekaServer*

```
@SpringBootApplication
@EnableEurekaServer
public class ServiceDiscoveryApp {

    public static void main(String[] args) {

        SpringApplication.run(ServiceDiscoveryApp.class, args);
    }
}
```

- **Paso 4**: Establece las propiedades del proyecto en el fichero *application.yml,* entre ellas el **puerto** donde el servidor escuchará las peticiones.
- **Paso 5**: Ejecuta la aplicación y comprueba que accedes al *dashboard* de *Eureka*, indicando en la URL el dominio y puerto que estableciste en paso anterior. En mi ejemplo sería http://localhost:1111/

Construcción del *Greeting Service*

La lógica de ejecución de este microservicio será muy sencilla. Vía REST recibirá un parámetro de entrada con el nombre de una persona y devolverá un saludo.

- **Paso 1**: Nuevo proyecto maven que en mi caso lo nombro como *GreetingService*
- **Paso 2:** Configura el *pom.xml* del proyecto tal y como se indica en este *ejemplo*
- **Paso 3:** Crea la clase principal de la aplicación y añade la anotación *@EnableDiscoveryClient* para que el servicio se registre en *Eureka*

```
@EnableAutoConfiguration
@EnableDiscoveryClient
@SpringBootApplication
public class GreetingServiceApp {

    public static void main(String[] args) {

        SpringApplication.run(GreetingServiceApp.class, args);
    }
}
```

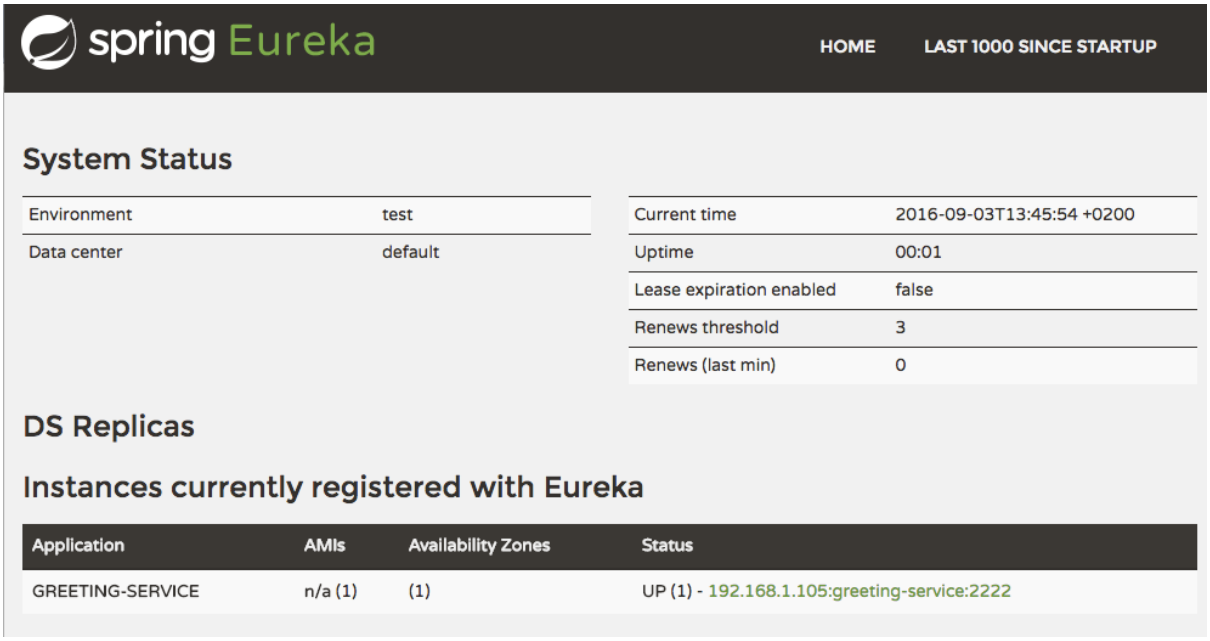
- **Paso 4:** Crea un *REST Controller* para atender las peticiones que se reciban a través de la URI */greeting/{name}*:

```
@RestController
public class GreetingServiceController {

    private static final String template = "Hello, %s!";
```

```
    @RequestMapping("/greeting/{name}")
    public String greeting2(@PathVariable("name") String name) {
        return String.format(template, name) ;
    }
}
```

- **Paso 5:** Configura el puerto donde el servicio escuchará peticiones y las propiedades de acceso a Eureka tal y como aquí se indica
- **Paso 6:** Ejecuta la aplicación y comprueba que *Greeting Service* aparece registrado en la consola de *Eureka*



- **Paso 7:** Por último, verifica que el servicio responde correctamente a la peticiones REST que se realicen, como por ejemplo http://localhost:2222/greeting/Rob

Construcción del *Edge Service*

A continuación, vamos a implementar el *API Gateway* con *Zuul* para facilitar el consumo vía REST del microservicio *Greeting Service*.

- **Paso 1**: Crea un nuevo proyecto Maven. En mi caso *EdgeService*
- **Paso 2:** Configura el fichero *pom.xml* de proyecto y asegúrate que entre las dependencias se incluye:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-zuul</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-eureka-server</artifactId>
</dependency>
```

- **Paso 3:** Crea la clase principal de la aplicación *Spring Boot* y añade las anotaciones *@EnableZuulProxy* para habilitar *Zuul* y *@EnableDiscoveryClient* para conectar con el *Service Discovery (Eureka)*

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableZuulProxy
public class EdgeServiceApp {
    public static void main(String[] args) {

        SpringApplication.run(EdgeServiceApp.class, args);
    }
}
```

Paso 4: Establece las siguientes propiedades en el fichero *application.yml* del proyecto:

- *Mapeo de rutas:* en el ejemplo indico que todas las peticiones de entrada que lleguen al *Edge Service* con la uri /greeting/** se deriven al *endpoint* del servicio cuyo identificador de registro en *Eureka* es *greeting-service*

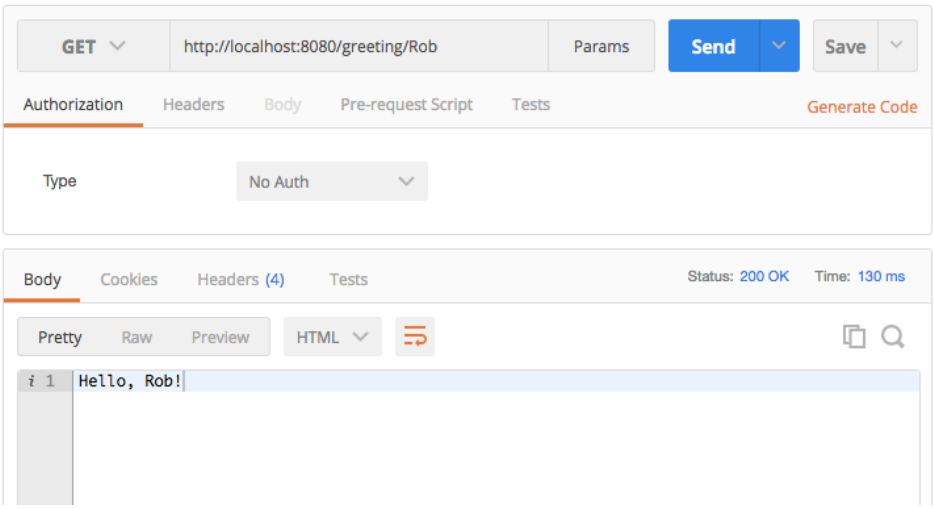
```
zuul:
  routes:
    greetings:
      path: /greeting/**
      serviceId: GREETING-SERVICE
      stripPrefix: false
```

- Acceso al *Service Discovery* y el *puerto* donde estará levantado el servidor

```
# Discovery Server Access
eureka:
  client:
    registerWithEureka: false
    serviceUrl:
      defaultZone: http://localhost:1111/eureka/

server:
  port: 8080   # HTTP (Tomcat) port
```

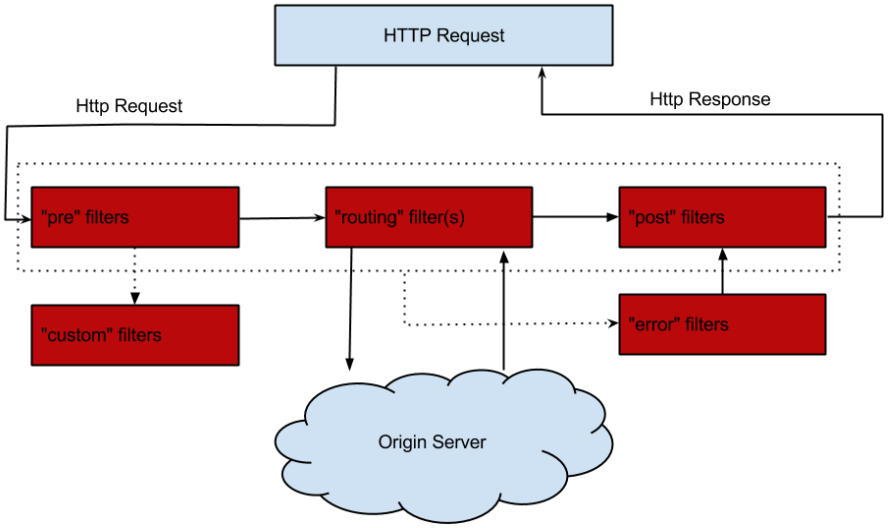
- **Paso 5:** Ejecutar la aplicación y comprobar que funciona correctamente. Para verificarlo haremos una petición a http://localhost:8080/greeting/Rob que será capturada por *Zuul*, el cual, a través de *Ribbon*, llamará a *Eureka* para obtener la *URL* de publicación de *Greeting Service*. A continuación se enrutará el flujo de la ejecución hacia el nuevo *endpoint* obtenido y se devolverá un resultado a la petición original



Filtros

Además de las capacidades de enrutado que hemos visto, *Zuul* ofrece muchas otras funcionalidades para sacarle partido a nuestro *Edge Service,* como por ejemplo los filtros, los cuales permiten realizar un amplio número de acciones durante el ciclo de vida de las peticiones HTTP.

Existen cuatro tipos distintos de filtros, correspondientes a cada uno de los estados por los que pasa una petición: PRE, ROUTING, POST, ERROR



Vamos a implementar un filtro de tipo *PRE* para que nos proporcione información de la petición que se ha hecho. Debido a su sencillez, utilizaré la implementación propuesta por Kasper Nissen en su fantástico post.

- **Paso 1:** Crea una nueva clase que extienda de *ZuulFilter*

```
public class PreFilter extends ZuulFilter {
    private static Logger log = LoggerFactory.getLogger(PreFilter.class);

    @Override
    public String filterType() {
        return "pre";
```

```
    }

    @Override
    public int filterOrder() {
        return 1;
    }

    @Override
    public boolean shouldFilter() {
        return true;
    }

    @Override
    public Object run() {
        RequestContext ctx = RequestContext.getCurrentContext();
        HttpServletRequest request = ctx.getRequest();

        log.info(String.format("%s request to %s", request.getMethod(),
request.getRequestURL().toString()));
        return null;
    }
}
```

- **Paso 2:** Añade un nuevo *bean* del filtro en la clase principal de la aplicación

```
public class EdgeServiceApp {

    public static void main(String[] args) {
        SpringApplication.run(EdgeServiceApp.class, args);
    }

    @Bean
    public PreFilter preFilter() {
        return new PreFilter();
    }
}
```

- **Paso 3:** Inicia de nuevo el *Edge Service,* realiza una petición y comprueba que el filtro *PreFilter* se ha ejecutado, escribiendo por la consola de salida la siguiente traza de log:

```
2016-09-03 16:36:37.656  INFO 2522 --- [nio-8080-exec-4]
n.r.m.zuul.filters.PreFilter : GET request to
http://localhost:8080/greeting/Rob
```

Con esto finalizamos el laboratorio.