Microservicios – Arquitectura y Desarrollo

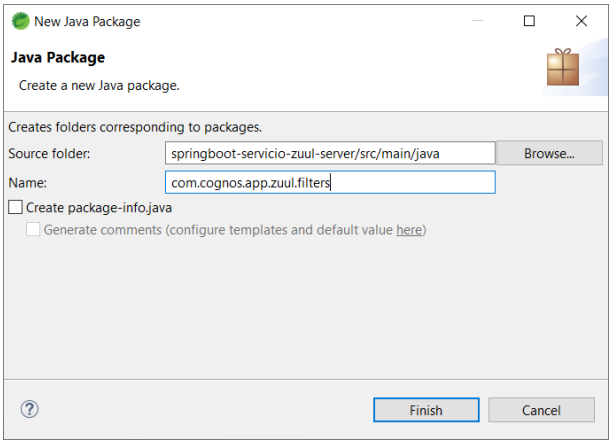
LAB Creando filtros y configurando tiempos de espera en Zuul

Objetivos

- Mostrar al participante el procedimiento para el desarrollo de Microservicios con filtros y tiempos de espera en Zuul

Procedimiento

1. En el proyecto **springboot-servicio-zuul-server**, crea el paquete com.cognos.app.zuul.filters.



2. En el paquete **com.cognos.app.zuul.filters** crea la clase PreTiempoTranscurridoFilter, este filtro será de tipo "pre".

```
package com.cognos.app.zuul.filters;

import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;

@Component
public class PreTiempoTranscurridoFilter extends ZuulFilter{
```

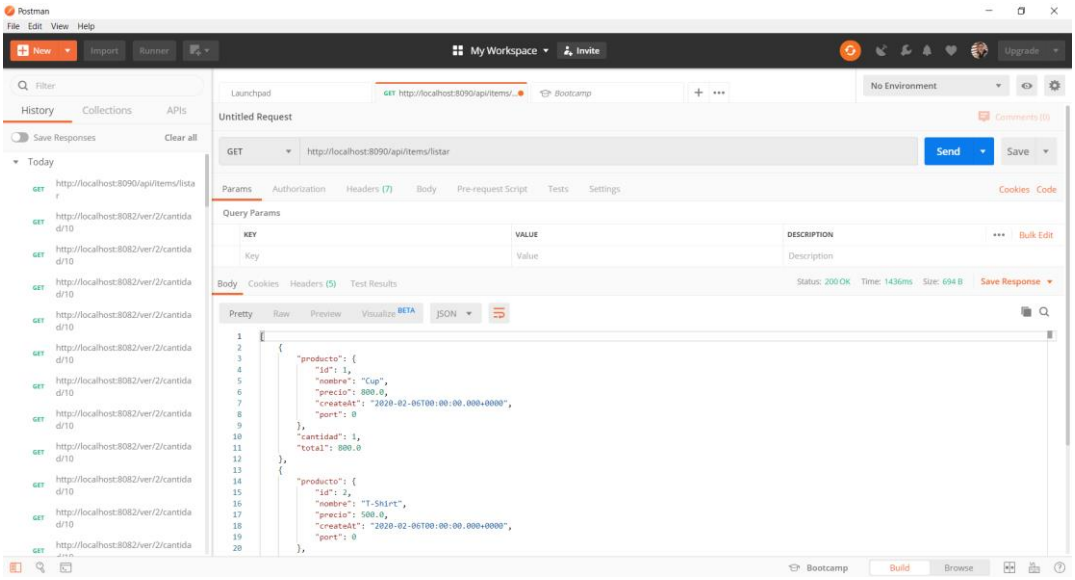
```
	private static Logger
			log = LoggerFactory.getLogger(PreTiempoTranscurridoFilter.class);


	@Override
	public boolean shouldFilter() {
		// activa el filtro true, desactiva false
		return true;
	}

	@Override
	public Object run() throws ZuulException {
		RequestContext ctx =  RequestContext.getCurrentContext();
		HttpServletRequest request = ctx.getRequest();
		log.info(String.format("%s request enrutado a %s",
				request.getMethod(),request.getRequestURI().toString()));

		Long tiempoInicio = System.currentTimeMillis();
		request.setAttribute("tiempoInicio", tiempoInicio);
		return null;
	}

	@Override
	public String filterType() {
		// palabras clave pre, post, route
		return "pre";
	}

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

}
```

1. En el paquete **com.cognos.app.zuul.filters** crea la clase PostTiempoTranscurridoFilter, este filtro será de tipo "post".

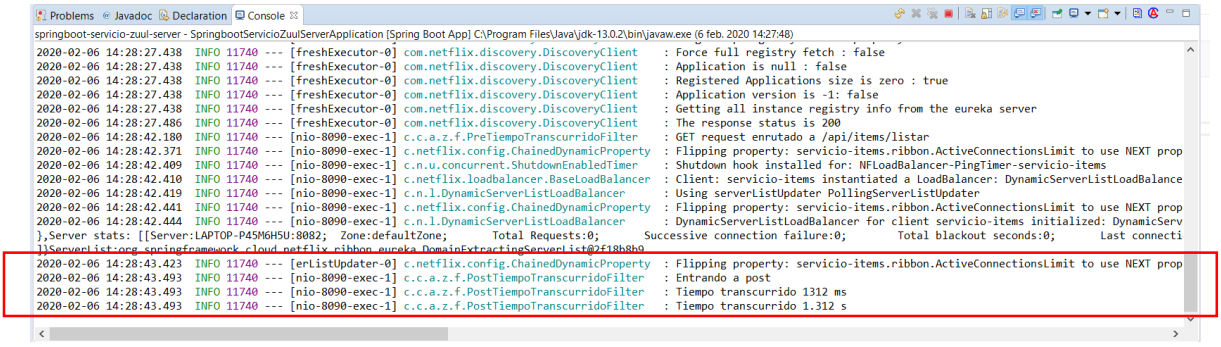
```
package com.cognos.app.zuul.filters;

import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import com.netflix.zuul.ZuulFilter;
import com.netflix.zuul.context.RequestContext;
import com.netflix.zuul.exception.ZuulException;

@Component
public class PostTiempoTranscurridoFilter extends ZuulFilter{

	private static Logger log =
			LoggerFactory.getLogger(PostTiempoTranscurridoFilter.class);
```

```
	@Override
	public boolean shouldFilter() {
		// activa el filtro true, desactiva false
		return true;
	}

	@Override
	public Object run() throws ZuulException {
		RequestContext ctx =  RequestContext.getCurrentContext();
		HttpServletRequest request = ctx.getRequest();
		log.info("Entrando a post");

		Long tiempoInicio = (Long) request.getAttribute("tiempoInicio");
		Long tiempoFinal = System.currentTimeMillis();
		Long tiempoTranscurrido = tiempoFinal - tiempoInicio;

		log.info(String.format("Tiempo transcurrido %s ms",
					tiempoTranscurrido));
		log.info(String.format("Tiempo transcurrido %s s",
					tiempoTranscurrido.doubleValue()/1000.0));

		return null;
	}

	@Override
	public String filterType() {
		// palabras clave pre, post, route, error
		return "post"; //post
	}

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

}
```

2. Inicia todos los servicios consume el servicio **SERVICIO-ITEMS** desde Postman y verifica la ejecución del filtro mirando el log en la console.



Consola de Zuul Server.



Configurando Tiempos de Espera en Zuul

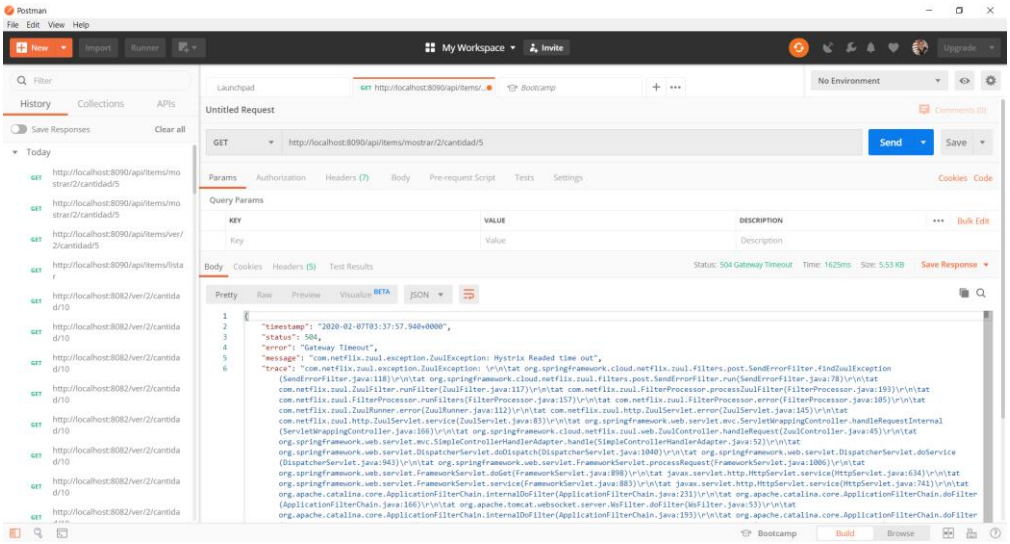
3. En el proyecto **springboot-servicio-producto**, edita la clase ProductoContoller y agrega la lógica para simular una espera.

```
package com.cognos.app.productos.controllers;

import java.util.List;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.env.Environment;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;
```

```java
import com.cognos.app.productos.model.entity.Producto;
import com.cognos.app.productos.model.service.ProductoService;

@RestController
public class ProductoController {

	@Autowired
	private Environment env;

	@Value("${server.port}")
	private Integer port;

	@Autowired
	private ProductoService productoService;

	@GetMapping("/listar")
	public List<Producto> listar(){
		return productoService.findAll().stream().map(p -> {

	//p.setPort(Integer.parseInt(env.getProperty("local.server.port")));
			p.setPort(port);
			return p;
		}).collect(Collectors.toList());
	}

	@GetMapping("/mostrar/{id}")
	public Producto detalle(@PathVariable Long id)  {
		Producto producto = productoService.findById(id);

    //producto.setPort(Integer.parseInt(env.getProperty("local.server.port")));
		producto.setPort(port);
		// codigo para simular un fallo
		/*
		 * if ( true) {
		 * throw new Exception ("Error: No se puede obtener un producto.");
		 * }
		 */

		 //Codigo para simular una espera de 3 segundos
		 try {
			Thread.sleep(3000L);
		 } catch (InterruptedException e) {
			e.printStackTrace();
		 }

		return producto;
	}
}
```

4. Inicia todos los servicios **SERVICIO-PRODUCTOS**, **SERVICIO-ITEMS**, Eureka y Zuul

5. Prueba con Postman acceder al servicio **SERVICIO-ITEMS** a través de Zuul.



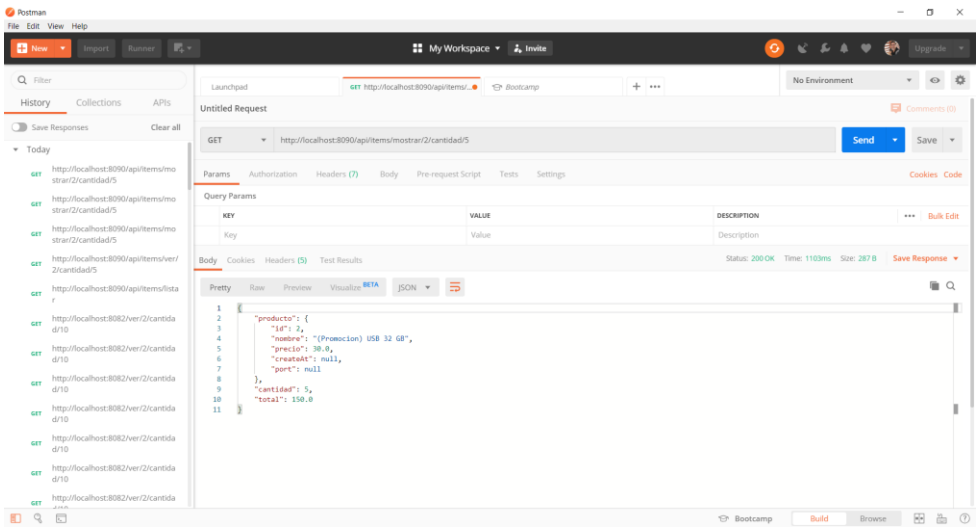
Nota: Se genera un error por el tiempo de espera en el servicio SERVICIO-PRODUCTOS que es controlado por Zuul.

6. En el proyecto springboot-servicio-zuul-server, edita el archivo application.properties y agrega la siguiente configuración de Hystrix.

```
spring.application.name=servicio-zuul-server
server.port=8090

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

zuul.routes.productos.service-id=servicio-productos
zuul.routes.productos.path=/api/productos/**

zuul.routes.items.service-id=servicio-items
zuul.routes.items.path=/api/items/**


hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds: 20000
ribbon.ConnectTimeout: 5000
ribbon.ReadTimeout: 10000
```

Guarda los cambios y reinicia el servicio Zuul.

7. Prueba con Postman acceder al servicio **SERVICIO-ITEMS** a través de Zuul.



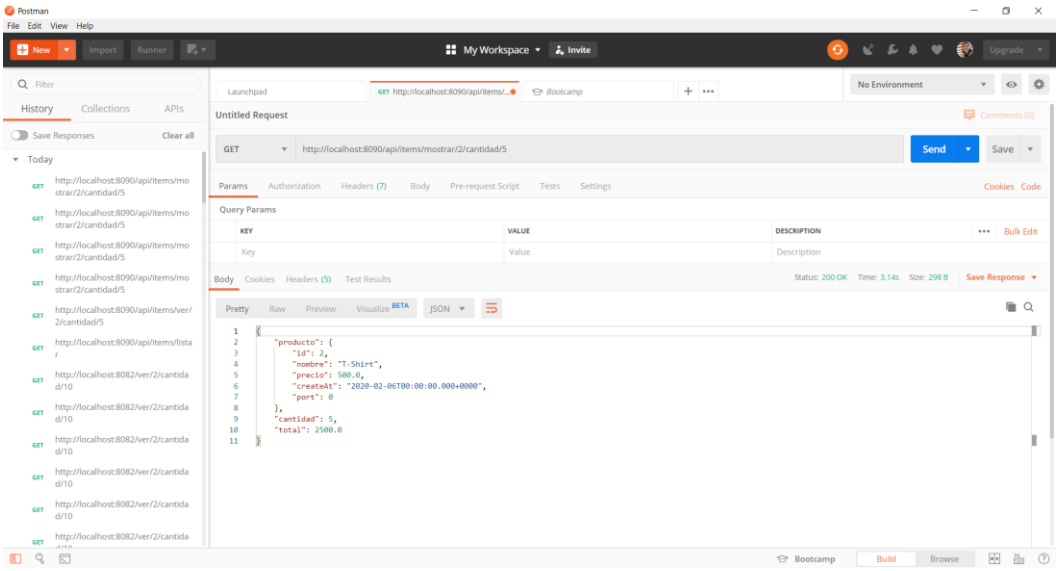
Nota: Como puedes observar se está ejecutando el método alternativo. Esto es porque Zuul si tienes configurado un tiempo de espera más largo y porque en SERVICIO-ITEMS tiene comentado la configuración de Hystrix

8. En el proyecto springboot-servicio-items, edita el archivo application.properties y descomenta la configuración de Hystrix de tal manera que el archivo contenga lo siguiente.

```
spring.application.name=servicio-items
server.port=8082

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds: 15000
ribbon.ConnectTimeout: 5000
ribbon.ReadTimeout: 5000
```

Guarda los cambio y reinicia el servicio SERVICIO-ITEMS.

9. Prueba con Postman acceder al servicio SERVICIO-ITEMS a través de Zuul.

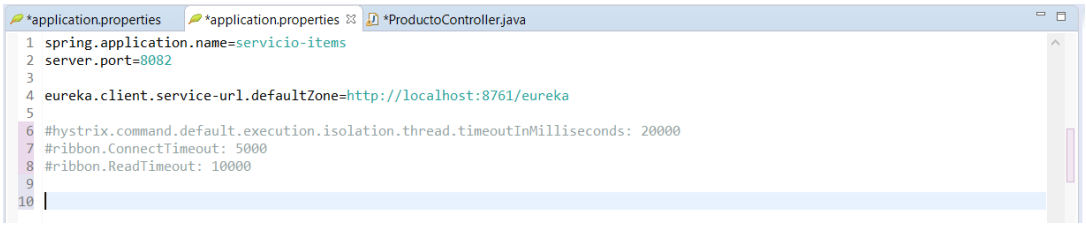


Nota: Como se observa ahora el servicio retorna el item original.

10. Para finalizar, comenta la configuración de Hystrix en los servicios SERVICIO-ITEMS y en Zuul server, también comenta el delay de espera en la clase ProductoControler del servicio SERVICIO-PRODUCTOS.

Zuul Server

SERVICIO-ITEMS



ProductoController.java

```java
package com.cognos.app.productos.controllers;

import java.util.List;

@RestController
public class ProductoController {

    @Autowired
    private Environment env;

    @Value("${server.port}")
    private Integer port;

    @Autowired
    private ProductoService productoService;

    @GetMapping("/listar")
    public List<Producto> listar(){
        return productoService.findAll().stream().map(p -> {
            //p.setPort(Integer.parseInt(env.getProperty("local.server.port")));
            p.setPort(port);
            return p;
        }).collect(Collectors.toList());
    }

    @GetMapping("/mostrar/{id}")
    public Producto detalle(@PathVariable Long id)  {
        Producto producto = productoService.findById(id);
        //producto.setPort(Integer.parseInt(env.getProperty("local.server.port")));
        producto.setPort(port);
        // codigo para simular un fallo
        /* if ( true) { throw new Exception ("Error: No se puede obtener un producto.");
         * }
         */
         //Codigo para simular una espera de 3 segundos
        /*
         * try { Thread.sleep(3000L); } catch (InterruptedException e) {
         * e.printStackTrace(); }
         */
        return producto;
    }
}
```

11. Reinicia el servidor Zuul y los servicios SERVICIO-PRODUCTOS, SERVICIO-ITEMS y prueba con Postman que todo funcione correctamente.