Microservices Development

# Step 1: Create Eureka Server

* Dependency : Eureka Server
* Add annotations : ‘@EnableEurekaServer’
* application.yml

|  |
| --- |
| server:  address: 0.0.0.0  port: 9091  error:  whitelabel:  enabled: false    eureka:  client:  register-with-eureka: false  fetch-registry: false  service-url:  defaultZone: ${EUREKA\_URI:http://10.10.11.70:9091/eureka}  instance:  prefer-ip-address: true |

* Create jar file of the project using ‘mvn clean install’

# Step 2: Create Zuul Service

* Dependency : Eureka Discovery, Zuul, Retry (optional-for failover of load-balancing)
* Add annotations : ‘@EnableDiscoveryClient’, ‘@EnableZuulProxy’, ‘@EnableRetry (optional)’
* Add Filters : Pre, Post, Route, Error (refer below – change filterType to ‘post’,’route’,’error’, and run())

|  |
| --- |
| public class PreFilter extends ZuulFilter {  @Override  public Object run() {  System.out.println(“Pre Filter”);  RequestContext ctx = RequestContext.getCurrentContext();  HttpServletRequest request = ctx.getRequest();  ctx.addZuulRequestHeader(“Authorization”, request.getHeader(“Authorization”));  System.out.println(“Request Method : “ + request.getMethod() + “ Request URL : “ + request.getRequestURL().toString());  return null;  }  @Override  public oolean shouldFilter() {  return true;  }  @Override  public int filterOrder() {  return 0;  }  @Override  public String filterType() {  return “pre”;  }  } |

Authorization is optional – depending on services.

* Create bean for filters in Application class – for all four filters

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

* application.yml

|  |
| --- |
| spring:  application:  name: zuul-service  server:  address: 0.0.0.0  port: 8084  error:  whitelabel:  enabled: false  eureka:  client:  service-url:  defaultZone: ${EUREKA\_URI:http://10.10.11.70:9091/eureka}  instance:  ip-address: true  zuul:  routes:  users:  path: /users/\*\*  serviceId: user-service  stripPrefix: true  orders:  path: /orders/\*\*  serviceId: order-service  stripPrefix: true  items:  path: /items/\*\*  serviceId: item-service  stripPrefix: true  host:  connect-timeout-millis: 30000  socket-timeout-millis: 30000  retryable: true  hystrix:  command:  default:  execution:  isolation:  thread:  timeoutInMilliseconds: 30000 |

Here, ‘retryable’ is optional as referred above.

# Step 3: Create Services (Item Service)

* Dependency : Web, Eureka Discovery, Actuator, Hystrix, Hystrix Dashboard, MySQL, JPA

|  |
| --- |
| <dependency>  <groupId>com.netflix.hystrix</groupId>  <artifactId>hystrix-metrics-event-stream</artifactId>  </dependency> |

* Annotations : ‘@EnableEurekaClient’, ‘@EnableCircuitBreaker’, ‘@EnableHystrixDashboard’
* Servlet Registration for Hystrix Stream – in Application class

|  |
| --- |
| @Bean  public ServletRegistrationBean servletRegistration() {  ServletRegistrationBean registration = new ServletRegistrationBean(new HystrixMetricsStreamServlet(), "/hystrix.stream");  return registration;  } |

* application.yml

|  |
| --- |
| spring:  application:  name: item-service  jpa:  hibernate:  ddl-auto: none  datasource:  url: jdbc:mysql://10.10.11.70:3306/ms\_poc  username: root  password: password  server:  address: 0.0.0.0  port: 7073  error:  whitelabel:  enabled: false  eureka:  client:  service-url:  defaultZone: ${EUREKA\_URI:http://10.10.11.70:9091/eureka}  instance:  ip-address: true |

* CORS filter component class – for allowing origin and methods

# Step 4: Create Turbine Service

* Dependency : Web, Eureka Discovery, Actuator, Hystrix, Hystrix Dashboard, Turbine
  + Add Metric Event dependency same as Services
* Annotation : ‘@EnableEurekaClient’, ‘@EnableHystrixDashboard’, ‘@EnableTurbine’
  + Add same bean as service class for hystrix streaming
* application.yml

|  |
| --- |
| spring:  application:  name: turbine-service  server:  address: 0.0.0.0  port: 8089  error:  whitelabel:  enabled: false  eureka:  client:  service-url:  defaultZone: ${EUREKA\_URI:http://10.10.11.70:9091/eureka}  instance:  ip-address: true    turbine:  clusterNameExpression: new String("default")  appConfig: user-service, order-service, item-service |