# Eureka

We will use eureka for service registration and discovery.

## Eureka Server

Github source code:

Standalone mode:

Just run the EurekaServerApp in idea, by default it will start with slave0 yml file.

Cluster mode:

1. Run EurekaServerApp with slave1 YML file. --spring.profiles.active=slave1
2. Run EurekaServerApp with slave2 YML file. --spring.profiles.active=slave2

We can use this address(127.0.0.1:1111,127.0.0.1:1112) to access it.

Eureka login username and password are set in YML file. spring.security.user.name and password.

## **Eureka Client**

### 1.2.1 Integrating eureka client

1. Add dependency into pom

|  |
| --- |
| <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId> </dependency> |
| <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies> </dependencyManagement> |

1. Configure eureka information in YML file.

|  |
| --- |
| eureka:  instance:  preferIpAddress: true  instanceId: ${spring.cloud.client.ip-address}:${server.port}  client:  register-with-eureka: true  fetch-registry: true  service-url:  defaultZone: http://Hins:Hins2021@127.0.0.1:1111/eureka,http://Hins:Hins2021@127.0.0.1:1112/eureka |

### **1.2.2 Service communication base on eureka**

#### **RestTemplate**

|  |
| --- |
| @Configuration public class RestTemplateConfig {   @Bean  @LoadBalanced  public RestTemplate restTemplate() {  RestTemplate restTemplate = new RestTemplate();  return restTemplate;  } } |
| restTemplate.getForEntity("http://SERVICE-PROVIDER/provider/{1}", String.class, id); |

#### **OpenFeign**

|  |
| --- |
| <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-openfeign</artifactId> </dependency> |
| @SpringBootApplication @EnableFeignClients public class ServiceConsumerApplication {   public static void main(String[] args) {  SpringApplication.*run*(ServiceConsumerApplication.class,args);  } } |
| @FeignClient(value = "SERVICE-PROVIDER") // your provider name public interface ProviderFeignService{   @GetMapping(value = "/provider/{id}")  String providerInfo(@PathVariable("id") String id);  } |

# Sentinel

## How to use

https://github.com/alibaba/Sentinel/wiki/How-to-Use

## Integrating Sentinel

1. **add dependency**

|  |
| --- |
| <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-starter-alibaba-sentinel</artifactId>  <version>2.2.1.RELEASE</version> </dependency> |

1. **add annotation and create handler**

|  |
| --- |
| @SentinelResource(value = "defaultBlockHandler",blockHandlerClass = {SentinelExceptionUtil.class},blockHandler = "defaultBlockHandler")  @GetMapping(value ="/defaultBlockHandler", produces = MediaType.*APPLICATION\_JSON\_VALUE*) public R defaultBlockHandler(){  ResponseEntity<R> user = restTemplate.getForEntity("http://SERVICE-PROVIDER/provider/{1}", R.class, 1);  return user.getBody(); } |
| public class SentinelExceptionUtil {   public static R defaultBlockHandler(BlockException ex)  {  return R.*fail*(HttpStatus.*TOO\_MANY\_REQUESTS*.value(),HttpStatus.*TOO\_MANY\_REQUESTS*.getReasonPhrase());  }  } |

1. **Define rule**

|  |
| --- |
| [  {  "resource": "blockHandler",  "count": 1,  "grade": 1,  "limitApp": "default",  "strategy": 0,  "controlBehavior": 0  },  {  "resource": "defaultBlockHandler",  "count": 1,  "grade": 1,  "limitApp": "default",  "strategy": 0,  "controlBehavior": 0  } ] |

1. **set up sentinel data source in application.yml file.**

|  |
| --- |
| spring:  application:  name: service-consumer  cloud:  sentinel:  datasource:  ds1:  file:  file: classpath:sentinel-block-handler-flow.json  rule-type: flow |

# Hazelcast

We use hazelcast as micro service cache.

## 2.1 Hazelcast Server

Standalone mode:

Just run the HazelcastApplication in idea

Cluster mode:

1. Run HazelcastApplication with server.port=9000 and hazelcast.port=5701
2. Run HazelcastApplication with server.port=9001 and hazelcast.port=5702

## **2.2 Hazelcast Client**

**2.2.1 Integrating hazelcast client**

1. **add dependency**

|  |
| --- |
| <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast</artifactId>  <version>4.0</version> </dependency> <dependency>  <groupId>com.hazelcast</groupId>  <artifactId>hazelcast-eureka-one</artifactId>  <version>2.0.1</version> </dependency> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId> </dependency>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies> </dependencyManagement> |

1. **integrating eureka client**

|  |
| --- |
| eureka:  instance:  preferIpAddress: true  instanceId: ${spring.cloud.client.ip-address}:${server.port}  client:  register-with-eureka: true  fetch-registry: true  service-url:  defaultZone: http://Hins:Hins2021@127.0.0.1:1111/eureka,http://Hins:Hins2021@127.0.0.1:1112/eureka |

1. **add hazelcast client configuration**

|  |
| --- |
| @Configuration public class HazelcastClientConfig {   @Value("${eureka.client.service-url.defaultZone}")  private String eurekaUrl;   @Bean  public ClientConfig clientConfig() {  ClientConfig clientConfig = new ClientConfig();  clientConfig.setClusterName("eureka");  clientConfig.getNetworkConfig().getEurekaConfig().setEnabled(true)  .setProperty("namespace", "hazelcast")  .setProperty("use-metadata-for-host-and-port", "true")  .setProperty("use-classpath-eureka-client-props","false")  .setProperty("name","hazelcast-server")  .setProperty("serviceUrl.default",eurekaUrl);  return clientConfig;  }  } |

# Logging

1. **add dependency**

|  |
| --- |
| <dependency>  <groupId>com.hsbc.itid.es.digiba</groupId>  <artifactId>log-spring-boot-starter</artifactId>  <version>1.0</version> </dependency> |

1. **add @EnableLogging annotation**

|  |
| --- |
| @SpringBootApplication @EnableLogging public class ServiceConsumerApplication {   public static void main(String[] args) {  SpringApplication.*run*(ServiceConsumerApplication.class,args);  }  } |

# Quartz

1. **add dependency**

|  |
| --- |
| <dependency>  <groupId>com.hsbc.itid.es.digiba</groupId>  <artifactId>quartz-common</artifactId>  <version>1.0</version> </dependency> |

1. **configure data source in application.yml**

|  |
| --- |
| spring:  application:  name: quartz-demo-service  datasource:  dynamic:  enable: true  core:  jdbcUrl: jdbc:mysql://127.0.0.1:3306/release?useUnicode=true&characterEncoding=utf-8&allowMultiQueries=true&useSSL=false&serverTimezone=Asia/Shanghai&allowPublicKeyRetrieval=true  username: root  password: 123456  driver-class-name: com.mysql.cj.jdbc.Driver  quartz:  jdbcUrl: jdbc:mysql://127.0.0.1:3306/quartz?useUnicode=true&characterEncoding=utf-8&allowMultiQueries=true&useSSL=false&serverTimezone=Asia/Shanghai&allowPublicKeyRetrieval=true  username: root  password: 123456  driver-class-name: com.mysql.cj.jdbc.Driver |

1. **add annotation**

|  |
| --- |
| @SpringBootApplication(exclude = {DataSourceAutoConfiguration.class}) @EnableQuartz @MapperScan(  basePackages = {"com.hsbc.itid.es.digiba.quartz.mapper","com.hsbc.itid.es.digiba.qtz.dao"} ) @EnableLogging public class QuartzApplication {   public static void main(String[] args) {  SpringApplication.*run*(QuartzApplication.class,args);  }  } |

1. **create job**

|  |
| --- |
| public Result releaseByCustomTime(Business business) throws Exception {  SysJob sysJob = new SysJob();  sysJob.setJobName("XXX Release Job");  sysJob.setJobGroup(JOB\_GROUP);  sysJob.setInvokeTarget("businessService.doRelease()");  sysJob.setStartTime(business.getReleaseTime());  sysJob.setTriggerType(ScheduleConstants.TriggerType.*SIMPLE*.getValue());  sysJob.setParams(objectMapper.writeValueAsString(business));  Result resultInsert = iSysJobService.insertJob(sysJob);  *log*.info("corn job create result: {}",resultInsert);  return resultInsert; } |
| public Result releaseByCorn(Business business) throws Exception {  SysJob sysJob = new SysJob();  sysJob.setJobName("XXX Cron Count Job");  sysJob.setJobGroup(JOB\_GROUP);  sysJob.setInvokeTarget("businessService.countTotalUser()");  sysJob.setCronExpression(business.getCornExpression());  sysJob.setTriggerType(ScheduleConstants.TriggerType.*CRON*.getValue());  //sysJob.setParams(objectMapper.writeValueAsString(business));  Result resultInsert = iSysJobService.insertJob(sysJob);  *log*.info("corn job create result: {}",resultInsert);  return resultInsert; } |

# Admin client and actuator

1. add dependency

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actuator</artifactId> </dependency>  <dependency>  <groupId>de.codecentric</groupId>  <artifactId>spring-boot-admin-starter-client</artifactId>  <version>2.3.1</version> </dependency> |

1. add metrics configuration

|  |
| --- |
| management:  endpoints:  web:  exposure:  include: "\*"  endpoint:  health:  show-details: *ALWAYS* |