Eureka 系列一 - 入门及启动

作者：noryar
原文链接：https://ld246.com/article/1620229563069
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

最好的学习方式就是阅读官方手册。

github：https://github.com/Netflix/eureka

Eureka介绍

这里直接引用百度百科的介绍吧

Eureka是Netflix开发的服务发现框架，本身是一个基于REST的服务，主要用于定位运行在AWS域中中间层服务，以达到负载均衡和中间层服务故障转移的目的。SpringCloud将它集成在其子项目sprin-cloud-netflix中，以实现SpringCloud的服务发现功能。

Eureka包含两个组件：Eureka Server和Eureka Client。

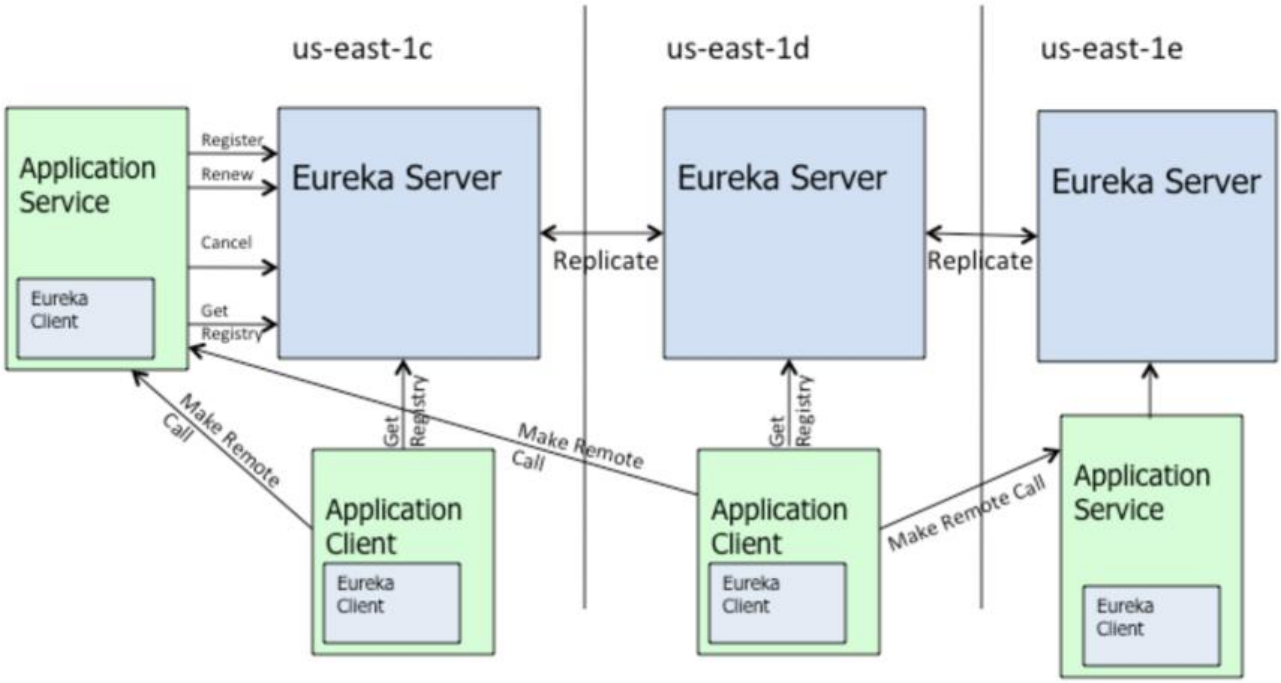
Eureka Server提供服务注册服务，各个节点启动后，会在Eureka Server中进行注册，这样EurekaSer er中的服务注册表中将会存储所有可用服务节点的信息，服务节点的信息可以在界面中直观的看到。

Eureka Client是一个java客户端，用于简化与Eureka Server的交互，客户端同时也就是一个内置的使用轮询(round-robin)负载算法的负载均衡器。

在应用启动后，将会向Eureka Server发送心跳,默认周期为30秒，如果Eureka Server在多个心跳周内没有接收到某个节点的心跳，Eureka Server将会从服务注册表中把这个服务节点移除(默认90秒)。

Eureka Server之间通过复制的方式完成数据的同步，Eureka还提供了客户端缓存机制，即使所有的Eu eka Server都挂掉，客户端依然可以利用缓存中的信息消费其他服务的API。综上，Eureka通过心跳查、客户端缓存等机制，确保了系统的高可用性、灵活性和可伸缩性。

Eureka High level Architecture：



使用spring-cloud启动集群

与常见的zookeeper作为注册中心不同的是，eureka是AP的实现方式，在保证可用性的前提下，牺

了一致性来保证服务的稳定性。对于在网络条件等不太好的情况下，euraka将是比zookeeper更好的择。当然了，现在eureka只有1.0版本，之后Netflix已经不再维护了（和Hystrix一样，lushed ）。

server的启动也直接参照官方说明来了：https://docs.spring.io/spring-cloud-netflix/docs/current/eference/html/#spring-cloud-eureka-server

创建工程，引入依赖

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.3.1.RELEASE</version>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
  </dependency>
</dependencies>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>Hoxton.SR6</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

基本代码

```
package com.acssor.tengfei.registercenter;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@EnableEurekaServer
```

```
@SpringBootApplication
public class Application {

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

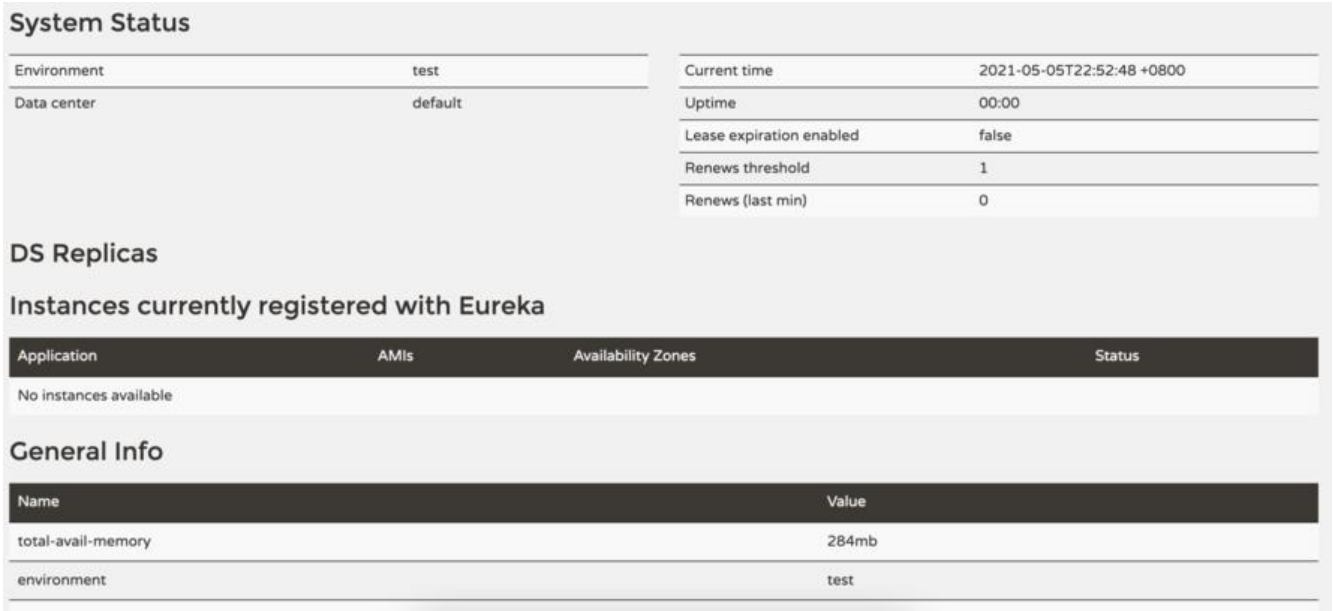
simple properties

application.yml

```
server:
  port: 8888
eureka:
  instance:
    hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

启动应用，查看控制台

http://localhost:8888/



注册一个服务

创建工程，引入依赖

这里偷懒了，很多依赖没有删除。大家看情况删除吧

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.1.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>com.google.guava</groupId>
        <artifactId>guava</artifactId>
        <version>29.0-jre</version>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.8.6</version>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.12</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Hoxton.SR6</version>
            <type>pom</type>
```

```
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

基本代码

```
package com.acssor.tengfei.student;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@EnableEurekaClient
@SpringBootApplication(scanBasePackages = "com.acssor.tengfei")
public class Application {

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

}
```

创建了一个url请求

```
package com.acssor.tengfei.student.web;

import com.acssor.tengfei.student.bean.StudentBean;
import com.acssor.tengfei.student.service.IStudentService;
import com.acssor.tengfei.student.util.GSONUtils;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.validation.ObjectError;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


import javax.annotation.Resource;
import java.util.List;
import java.util.stream.Collectors;
```

```
@RestController
@RequestMapping("/student")
@Slf4j
public class StudentController {

    @Resource
    private IStudentService studentService;

    @PostMapping
    public ResponseEntity save(@RequestBody @Validated StudentBean student, BindingResul
 bindingResult) {
        if (bindingResult.hasErrors()) {
            List<ObjectError> allErrors = bindingResult.getAllErrors();
            String errMsg = allErrors.stream().map(ObjectError::getDefaultMessage).collect(Collect
rs.joining(","));
            log.warn("students save error, invalid params, student:{}, error: {}", GSONUtils.toJson(s
udent), errMsg);
            return new ResponseEntity(errMsg, HttpStatus.BAD_REQUEST);
        }
        studentService.saveOrUpdate(student);
        return new ResponseEntity(student, HttpStatus.OK);
    }

}
```

simple properties

application.yml

```
spring:
  profiles:
    active: dev
  application:
    name: student-server
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:32769/tengfei?serverTimezone=UTC&useUnicode=true&charact
rEncoding=utf-8&useSSL=true
    username: root
    password: root

  jpa:
    properties:
      hibernate:
        hbm2ddl:
          auto: create
        dialect: org.hibernate.dialect.MySQL5InnoDBDialect
        format_sql: true
    show-sql: true
server:
  port: 10086
  tomcat:
```

```
    uri-encoding: UTF-8
    threads:
      max: 200
    max-connections: 800
    accept-count: 100
eureka:
  client:
    register-with-eureka: true # 是否注册自己的信息到EurekaServer，默认是true
    fetch-registry: true # 是否拉取其它服务的信息，默认是true
    service-url:
      defaultZone: http://localhost:8888/eureka # EurekaServer地址
  #instance:
    #prefer-ip-address: true # 当调用getHostname获取实例的hostname时，返回ip而不是host名

    #ip-address: 127.0.0.1 # 指定自己的ip信息，不指定的话会自己寻找
```

启动关键日志

```
2021-05-05 23:16:24.003  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Initializing Eureka in region us-east-1
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Disable delta property : false
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Single vip registry refresh property : null
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Force full registry fetch : false
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Application is null : false
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Registered Applications size is zero : true
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Application version is -1: true
2021-05-05 23:16:25.603  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Getting all instance registry info from the eureka server
2021-05-05 23:16:25.963  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : The response status is 200
2021-05-05 23:16:25.968  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Starting heartbeat executor: renew interval is: 30
2021-05-05 23:16:25.974  INFO 32990 --- [           main] c.n.discovery.InstanceInfoReplicator
 InstanceInfoReplicator onDemand update allowed rate per min is 4
2021-05-05 23:16:25.981  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Discovery Client initialized at timestamp 1620227785979 with initial instances count: 0
2021-05-05 23:16:25.987  INFO 32990 --- [           main] o.s.c.n.e.s.EurekaServiceRegistry        :
egistering application STUDENT-SERVER with eureka with status UP
2021-05-05 23:16:25.988  INFO 32990 --- [           main] com.netflix.discovery.DiscoveryClient
 : Saw local status change event StatusChangeEvent [timestamp=1620227785988, current=UP,
previous=STARTING]
2021-05-05 23:16:25.999  INFO 32990 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryCli
nt    : DiscoveryClient_STUDENT-SERVER/xxxxxx:student-server:10086: registering service...
2021-05-05 23:16:26.080  INFO 32990 --- [           main] o.s.b.w.embedded.tomcat.TomcatWe
Server  : Tomcat started on port(s): 10086 (http) with context path ''
2021-05-05 23:16:26.081  INFO 32990 --- [           main] .s.c.n.e.s.EurekaAutoServiceRegistratio
 : Updating port to 10086
2021-05-05 23:16:26.093  INFO 32990 --- [           main] DeferredRepositoryInitializationListen
```

```
r : Triggering deferred initialization of Spring Data repositories...
2021-05-05 23:16:26.174  INFO 32990 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryCli
nt    : DiscoveryClient_STUDENT-SERVER/xxxxxx:student-server:10086 - registration status: 204
```

控制台检查

注册一个调用者

创建工程，引入依赖

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.1.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Hoxton.SR6</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
```

```
            </plugin>
        </plugins>
    </build>
```

基本代码

```
package com.acssor.tengfei.boot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication(scanBasePackages = "com.acssor.tengfei")
@EnableZuulProxy
public class Application {

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

}
```

入口访问地址：

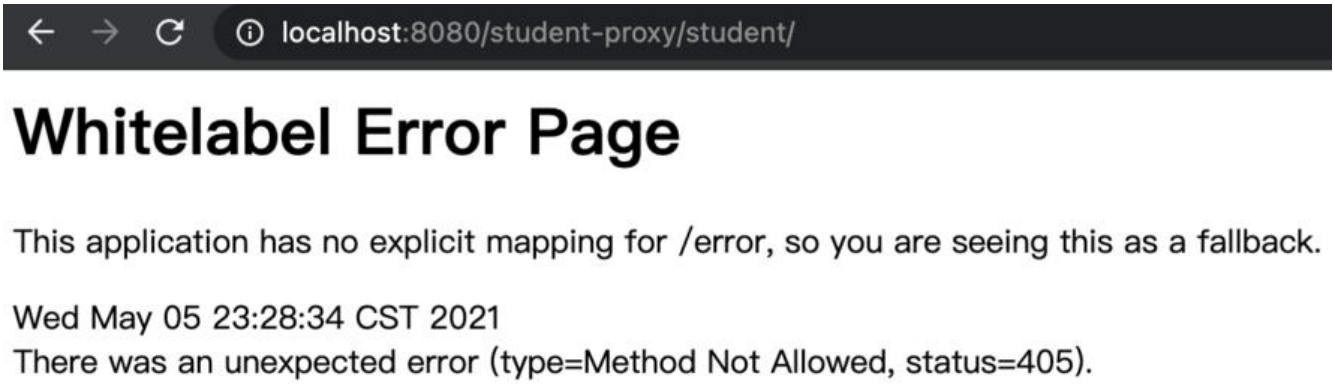
配置文件

```
spring:
  profiles:
    active: dev
server:
  port: 8080
  tomcat:
    uri-encoding: UTF-8
    threads:
      max: 200
    max-connections: 800
    accept-count: 100

#注册进eureka
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8888/eureka/

#配置网关转发详情
zuul:
  routes:
    api-a:
      path: /student-proxy/**
      service-id: student-server
#    api-b:
#      path: /order/**
```

```
#      service-id: order-server
```

测试访问

localhost:8080/student-proxy/student/。因为服务端是post请求的定义，因此通过调用者代理，回的将是405



结束语

以上，一个完整的例子结束。后续将对eureka的详细配置做一个介绍，来说明各种配置参数的定义及用场景是什么。