**3-5 | Dubbo使用入门介绍**

我们使用的是 jdk17 版本 + springboot3 + dubbo3 / springcloudalibaba要使用2022+的版本

**Provider层实现部分**

pom依赖配置：

总工程依赖：

|  |
| --- |
| XML <properties>  <qiyu-mysql.version>8.0.28</qiyu-mysql.version>  <maven.compiler.source>17</maven.compiler.source>  <maven.compiler.target>17</maven.compiler.target>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <springboot.version>3.0.4</springboot.version> </properties>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>3.0.4</version>  <relativePath/> <!-- lookup parent from repository --> </parent>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-alibaba-dependencies</artifactId>  <version>2022.0.0.0-RC1</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies> </dependencyManagement> |

qiyu-live-user-provider子项目依赖：

|  |
| --- |
| JavaScript <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <curator.version>2.12.0</curator.version>  <mybatis.version>3.5.1</mybatis.version>  <druid.version>1.1.20</druid.version>  <sharding.jdbc.version>4.0.0-RC1</sharding.jdbc.version>  <hessian.version>4.0.38</hessian.version>  <jetty.version>9.4.28.v20200408</jetty.version>  <dubbo.version>2.7.8</dubbo.version>  <spring-cloud-alibaba.version>2022.0.0.0-RC1</spring-cloud-alibaba.version>  <maven.compiler.source>17</maven.compiler.source>  <maven.compiler.target>17</maven.compiler.target>  <maven.compiler.compilerVersion>17</maven.compiler.compilerVersion> </properties>  <dependencies>  <dependency>  <groupId>org.apache.dubbo</groupId>  <artifactId>dubbo-spring-boot-starter</artifactId>  <version>3.2.0-beta.3</version>  </dependency>  <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  <exclusions>  <exclusion>  <artifactId>log4j-to-slf4j</artifactId>  <groupId>org.apache.logging.log4j</groupId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.idea</groupId>  <artifactId>qiyu-live-user-interface</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency> </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <configuration>  <classifier>exec</classifier>  </configuration>  </plugin>  </plugins> </build> |

配置文件：

bootstrap.yml

|  |
| --- |
| YAML spring:  application:  name: qiyu-live-user-provider  cloud:  nacos:  username: qiyu  password: qiyu  discovery:  server-addr: qiyu.test.com:8848  namespace: qiyu-live-test |

dubbo.properties

|  |
| --- |
| Properties dubbo.application.name=qiyu-live-user-application dubbo.registry.address=nacos://127.0.0.1:8848?namespace=qiyu-live-test&&username=qiyu&&password=qiyu dubbo.server=true dubbo.protocol.name=dubbo dubbo.protocol.port=9090 |

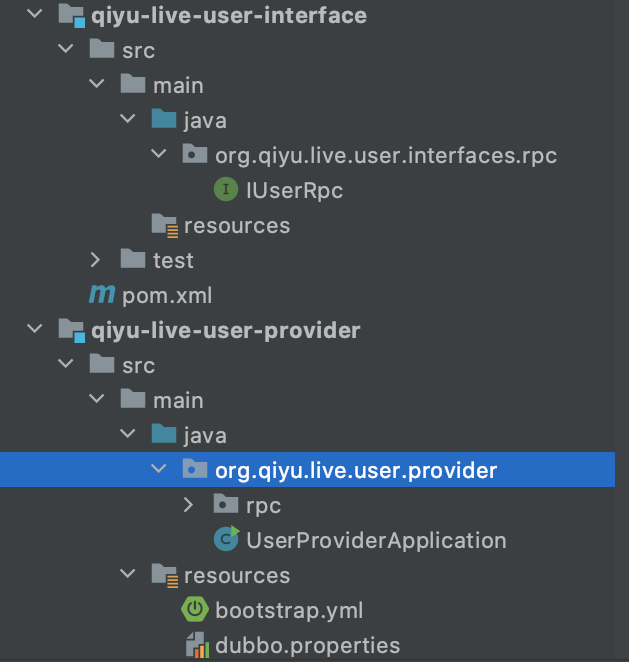
接下来便是我们的一个启动类模块：

|  |
| --- |
| Java package org.qiyu.live.user.provider;  import org.apache.dubbo.config.spring.context.annotation.EnableDubbo; import org.springframework.boot.SpringApplication; import org.springframework.boot.WebApplicationType; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  /\*\*  \* 用户服务中台启动类  \*/ @SpringBootApplication @EnableDiscoveryClient @EnableDubbo public class UserProviderApplication {   public static void main(String[] args) {  SpringApplication springApplication = new SpringApplication(UserProviderApplication.class);  springApplication.setWebApplicationType(WebApplicationType.NONE);  springApplication.run(args);  } } |

RPC接口实现部分：

|  |
| --- |
| Java package org.qiyu.live.user.provider.rpc;  import org.apache.dubbo.config.annotation.DubboService; import org.qiyu.live.user.interfaces.rpc.IUserRpc; import org.slf4j.Logger; import org.slf4j.LoggerFactory;  @DubboService public class UserRpcImpl implements IUserRpc {   private static final Logger LOGGER = LoggerFactory.getLogger(UserRpcImpl.class);   @Override  public void test() {  LOGGER.info("this is dubbo server test!");  } } |

建成后，目录如下图所示：



**Consumer层实现部分**

pom依赖如下：

|  |
| --- |
| JavaScript <dependencies>  <dependency>  <groupId>org.apache.dubbo</groupId>  <artifactId>dubbo-spring-boot-starter</artifactId>  <version>3.2.0-beta.3</version>  </dependency>  <dependency>  <groupId>com.alibaba.cloud</groupId>  <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  <exclusions>  <exclusion>  <artifactId>log4j-to-slf4j</artifactId>  <groupId>org.apache.logging.log4j</groupId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.idea</groupId>  <artifactId>qiyu-live-user-interface</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency> </dependencies> |

bootstrap.yml配置文件：

|  |
| --- |
| YAML spring:  application:  name: qiyu-live-api  cloud:  nacos:  username: qiyu  password: qiyu  discovery:  server-addr: qiyu.test.com:8848  namespace: qiyu-live-test |

dubbo.properties配置文件：

|  |
| --- |
| Properties dubbo.application.name=qiyu-live-api-application dubbo.registry.address=nacos://127.0.0.1:8848?namespace=qiyu-live-test&&username=qiyu&&password=qiyu |

启动类

|  |
| --- |
| Java package org.idea.qiyu.live.api;  import org.springframework.boot.SpringApplication; import org.springframework.boot.WebApplicationType; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  @SpringBootApplication @EnableDiscoveryClient public class ApiWebApplication {   public static void main(String[] args) {  SpringApplication springApplication = new SpringApplication(ApiWebApplication.class);  springApplication.setWebApplicationType(WebApplicationType.SERVLET);  springApplication.run(args);  } } |

controller部分

|  |
| --- |
| TypeScript package org.idea.qiyu.live.api.controller;  import org.apache.dubbo.config.annotation.DubboReference; import org.qiyu.live.user.interfaces.rpc.IUserRpc; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;  @RestController @RequestMapping(value = "/test") public class TestController {   @DubboReference  private IUserRpc userRpc;   @GetMapping(value = "/do-test")  public String doTest(){  userRpc.test();  return "success";  } } |

**如何进行本地的RPC调用**

|  |
| --- |
| Java package dubbo;  import org.apache.dubbo.config.ApplicationConfig; import org.apache.dubbo.config.ReferenceConfig; import org.apache.dubbo.config.RegistryConfig; import org.junit.jupiter.api.BeforeAll; import org.junit.jupiter.api.Test; import org.qiyu.live.user.interfaces.rpc.IUserRpc;  import java.util.HashMap; import java.util.Map;  public class DubboTest {   private static final String REGISTER\_ADDRESS = "nacos://127.0.0.1:8848?namespace=qiyu-live-test&&username=qiyu&&password=qiyu";  private static RegistryConfig registryConfig;  private static ApplicationConfig applicationConfig;  private static ReferenceConfig<IUserRpc> userRpcReferenceConfig;  private static Map<Class, Object> referMap = new HashMap<>();   @BeforeAll  public static void initConfig() {  registryConfig = new RegistryConfig();  applicationConfig = new ApplicationConfig();  registryConfig.setAddress(REGISTER\_ADDRESS);  applicationConfig.setName("dubbo-test-application");  applicationConfig.setRegistry(registryConfig);  userRpcReferenceConfig = new ReferenceConfig<>();  //roundrobin random leastactive shortestresponse consistenthash  userRpcReferenceConfig.setLoadbalance("random");  userRpcReferenceConfig.setInterface(IUserRpc.class);  referMap.put(IUserRpc.class, userRpcReferenceConfig.get());  }   @Test  public void testUserRpc() {  IUserRpc userRpc = (IUserRpc) referMap.get(IUserRpc.class);  for(int i=0;i<1000;i++) {  userRpc.test();  }  } } |

如果本地出现关于JDK17和Dubbo不适用的异常：

https://cn.dubbo.apache.org/zh-cn/blog/2018/08/07/%e4%bd%bf%e7%94%a8jdk17%e7%bc%96%e8%af%91%e8%bf%90%e8%a1%8cdubbo-2.7.14%e9%a1%b9%e7%9b%ae/

启动参数：

--add-opens java.base/java.lang=ALL-UNNAMED --add-opens java.base/sun.reflect.generics.reflectiveObjects=ALL-UNNAMED --add-opens java.base/java.math=ALL-UNNAMED