DUBBO REST
现在与未来

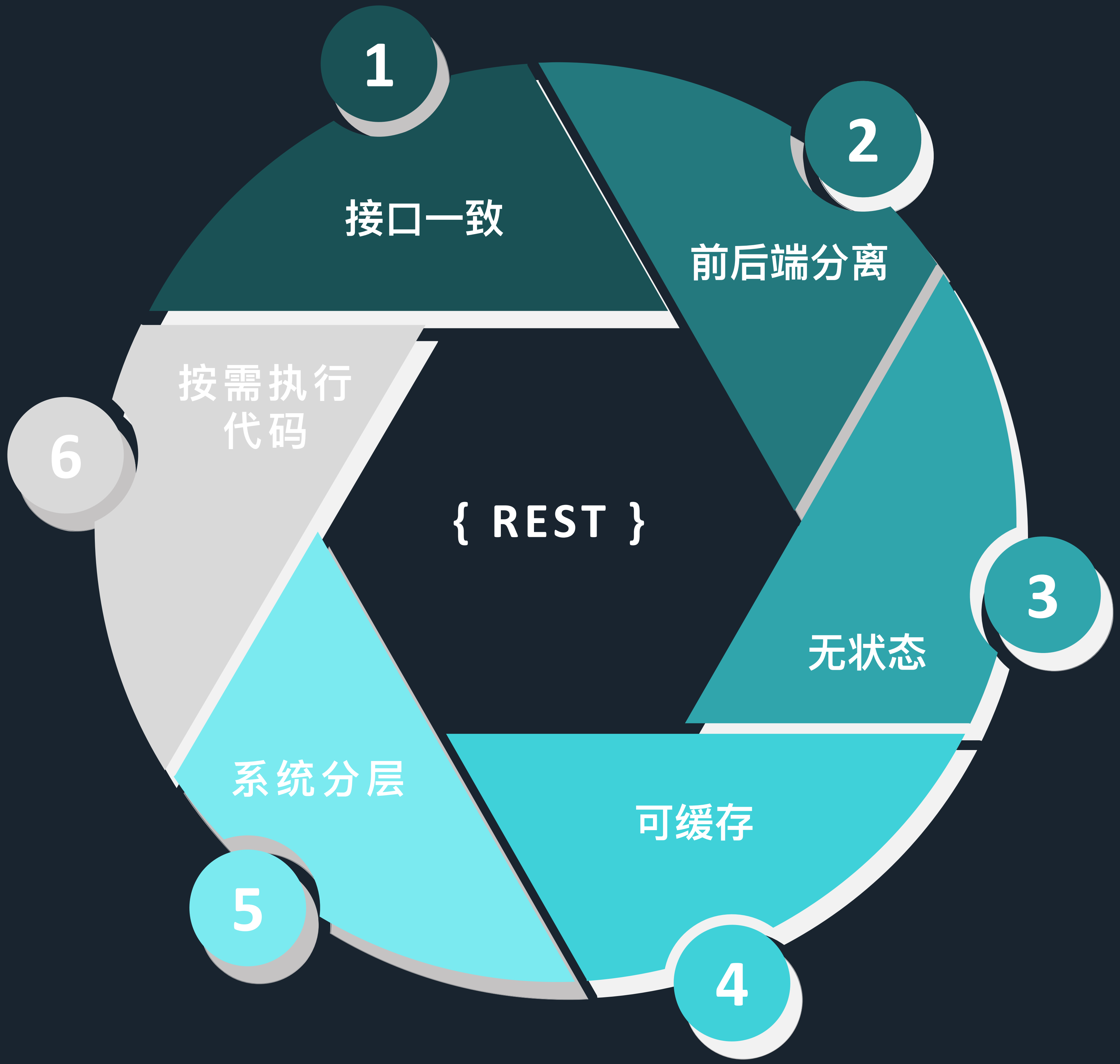
Apache Dubbo Meetup | Guangzhou

”REST 是一种软件架构风格，设计风格而不是标准，只是提供了一组设计原则和约束条件”

—Roy Fielding

REST 架构风格

适合 WWW 的软件架构



接口一致

资源的定位、资源的表达、资源的操作、资源的状态迁移

前后端分离

客户端、服务端解耦，便于并行开发

无状态

服务端不维护会话状态，但是负责资源的状态

可缓存

结果（表现层）可缓存

系统分层

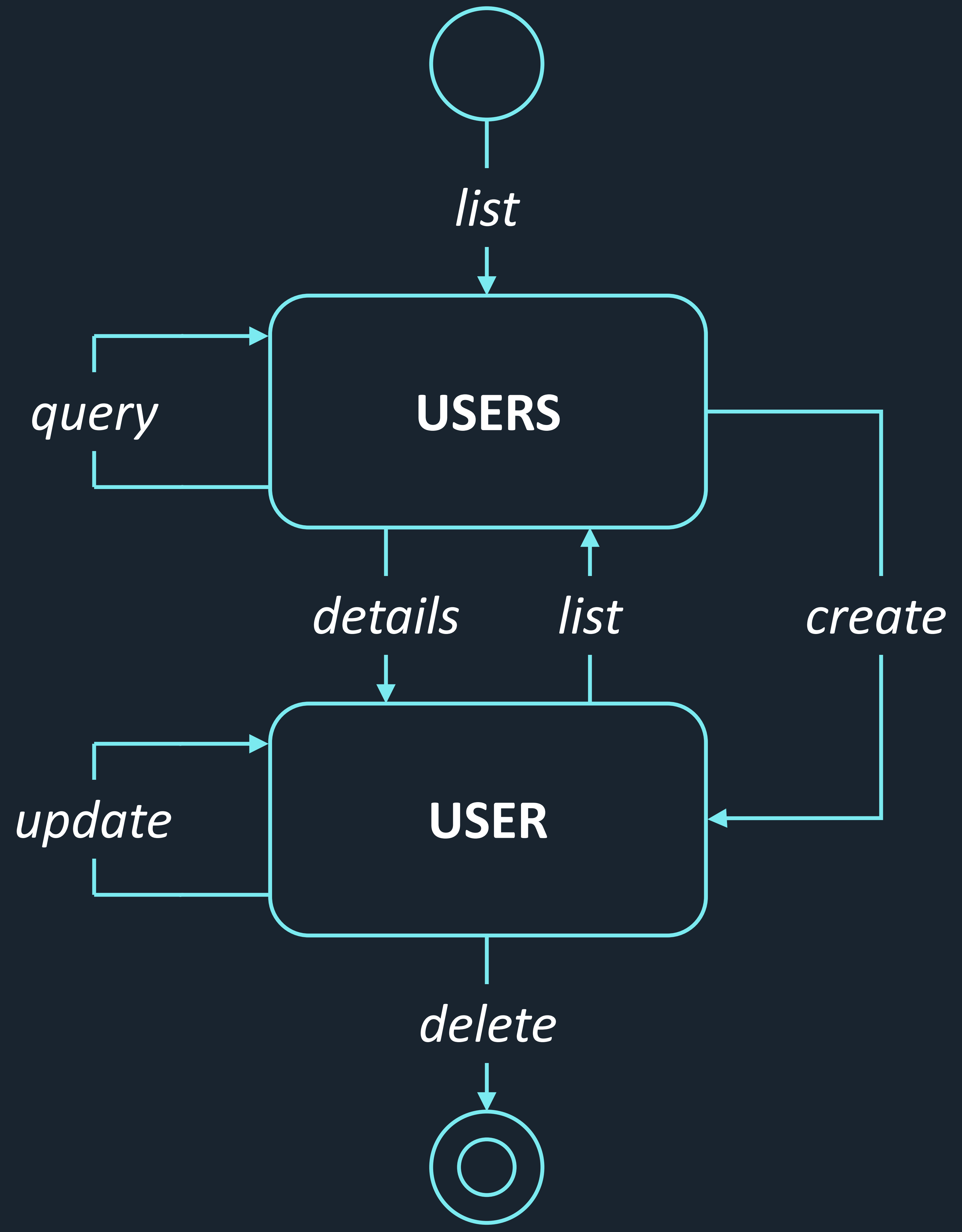
有反向代理提供负载均衡、缓存；不同资源由不同集群托管

按需执行代码

可选。客户端按需下载代码执行以扩展功能

面向资源的状态机

REpresentational State Transfer, Hypermedia As The Engine Of Application State



HTTP 方法

资源定位

```
GET /users HTTP/1.1
Accept: application/json
Host: localhost:8080
```

状态

HATEOAS

表现层

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: </users/1>; rel="tom"
Transfer-Encoding: chunked

[
  {
    "id": 1,
    "name": "Tom"
  }
]
```

Dubbo REST 特性

1. JAX-RS 2.0 标准

集成 RestEasy 3.0.19，基于 JAX-RS 2.0 标准 annotation 暴露 REST 服务

2. 无缝集成

平等对待 "rest" 协议。无缝享受框架提供的服务发现、服务治理、服务监控

3. 良好的互操作性

三种场景：非 Dubbo 调用 Dubbo；Dubbo 调用非 Dubbo；Dubbo 调用 Dubbo

4. 多种 REST Server

既支持嵌入式 netty、tomcat、jetty、http 也支持与外置的 servlet 容器的集成

接口

```java
@Path("users")
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
public interface UserService {
    @GET
    @Path("{id: \\d+}")
    User getUser(@PathParam("id") Long id);
}
```

Provider 配置

```xml
<bean>
    <dubbo:protocol name="rest" server="netty"/>
    <dubbo:service interface="UserService"
        protocol="rest" ref="service"/>
    <bean id="service" class="UserServiceImpl"/>
</beans>
```

Consumer 配置

```xml
</beans>
    <dubbo:reference id="service"
        interface="UserService"/>
</beans>
```

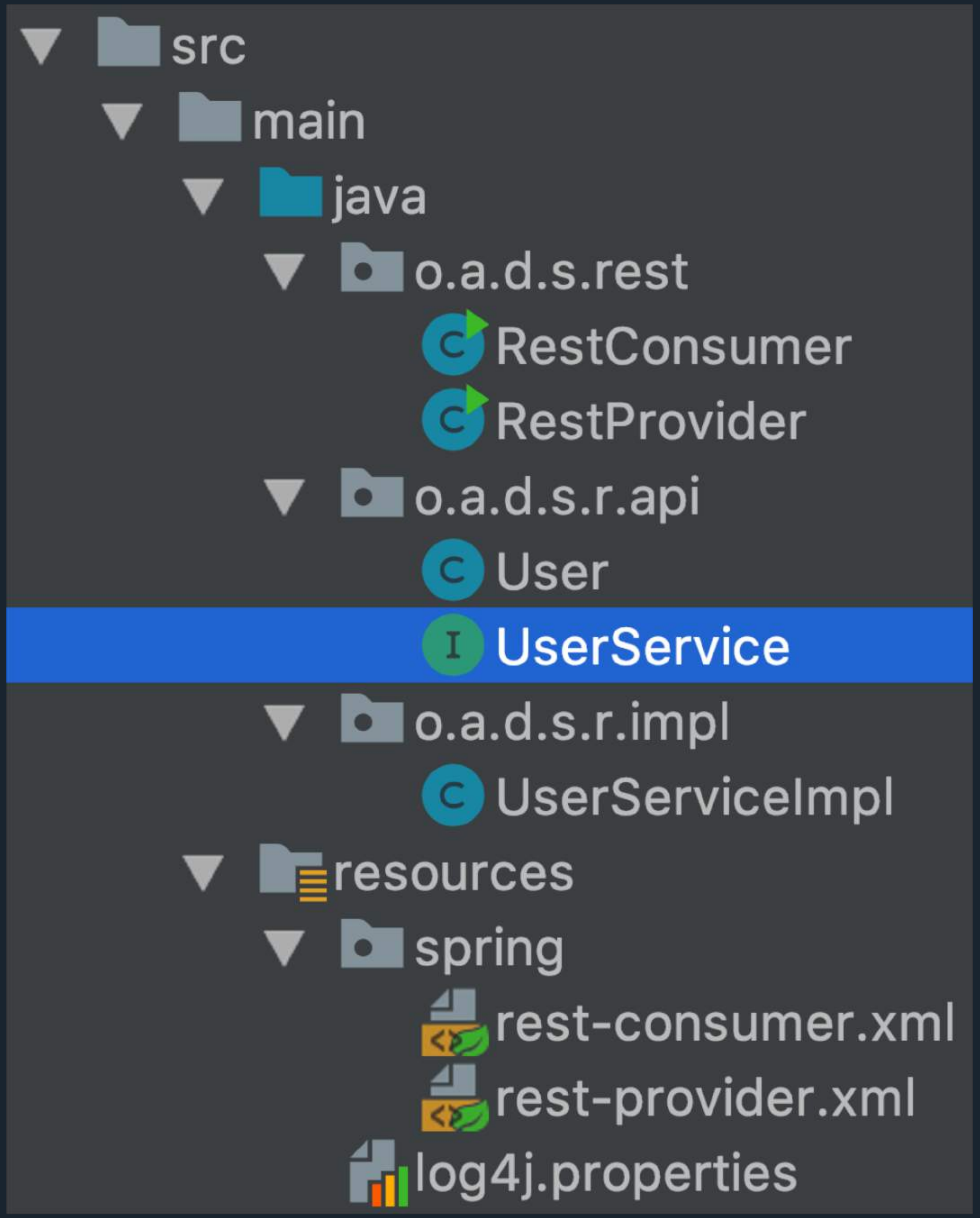
Dubbo REST 服务示例

步骤一、定义接口

USERSERVICE.JAVA

```java
@Path("users")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public interface UserService {
    @GET
    List<User> getUsers();

    @GET
    @Path("{id: \\d+}")
    User getUser(@PathParam("id") Long id);

    @POST
    Long registerUser(User user);
}
```

PROJECT LAYOUT



```
src
  main
    java
      o.a.d.s.rest
        RestConsumer
        RestProvider
      o.a.d.s.r.api
        User
        UserService
      o.a.d.s.r.impl
        UserServiceImpl
    resources
      spring
        rest-consumer.xml
        rest-provider.xml
      log4j.properties
```

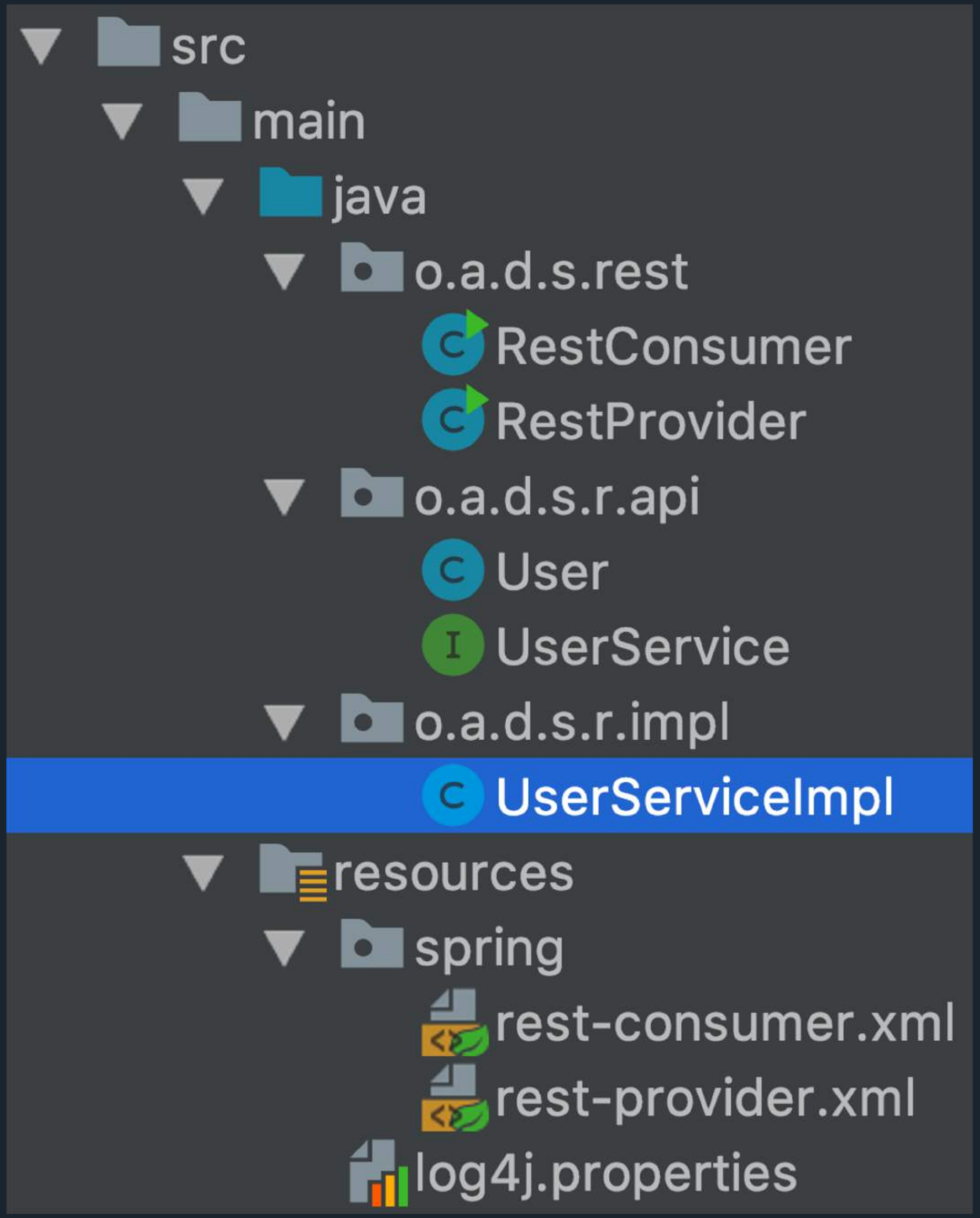
Dubbo REST 服务示例

步骤二、服务端接口实现

USERSERVICEIMPL.JAVA

```java
public class UserServiceImpl implements UserService {
    private final AtomicLong id = new AtomicLong();

    public List<User> getUsers() {
        return Arrays.asList(new User(1L, "Tom"),
            new User(2L, "Jerry"));
    }

    public User getUser(Long id) {
        return new User(id, "username-" + id);
    }

    public Long registerUser(User user) {
        return id.incrementAndGet();
    }
}
```

PROJECT LAYOUT



```
▼ src
  ▼ main
    ▼ java
      ▼ o.a.d.s.rest
          RestConsumer
          RestProvider
      ▼ o.a.d.s.r.api
          User
          UserService
      ▼ o.a.d.s.r.impl
          UserServiceImpl
    ▼ resources
      ▼ spring
          rest-consumer.xml
          rest-provider.xml
        log4j.properties
```

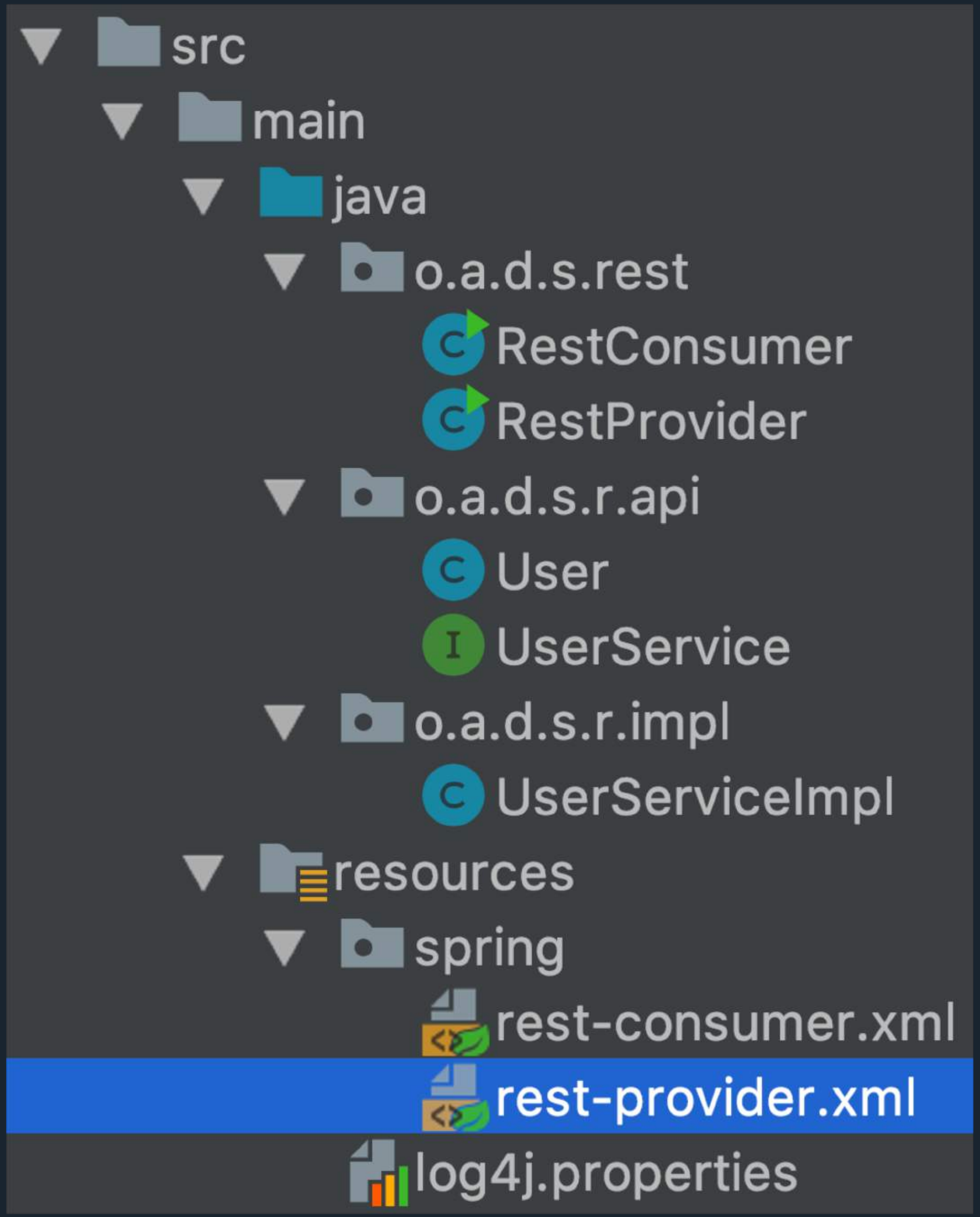
Dubbo REST 服务示例

步骤三、服务端配置

REST-PROVIDER.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>
    <dubbo:application name="rest-provider"/>
    <dubbo:registry address="zookeeper://127.0.0.1:2181"/>


    <dubbo:protocol name="rest" port="8080" server="netty"/>


    <dubbo:service interface="o.a.d.s.r.api.UserService”

        protocol="rest" ref="userService"/>


    <bean id="userService"
        class="o.a.d.s.r.impl.UserServiceImpl"/>
</beans>
```

PROJECT LAYOUT



```
▼ src
    ▼ main
        ▼ java
            ▼ o.a.d.s.rest
                RestConsumer
                RestProvider
            ▼ o.a.d.s.r.api
                User
                UserService
            ▼ o.a.d.s.r.impl
                UserServiceImpl
        ▼ resources
            ▼ spring
                rest-consumer.xml
                rest-provider.xml
            log4j.properties
```

Dubbo REST 服务示例

步骤四、启动服务端暴露服务

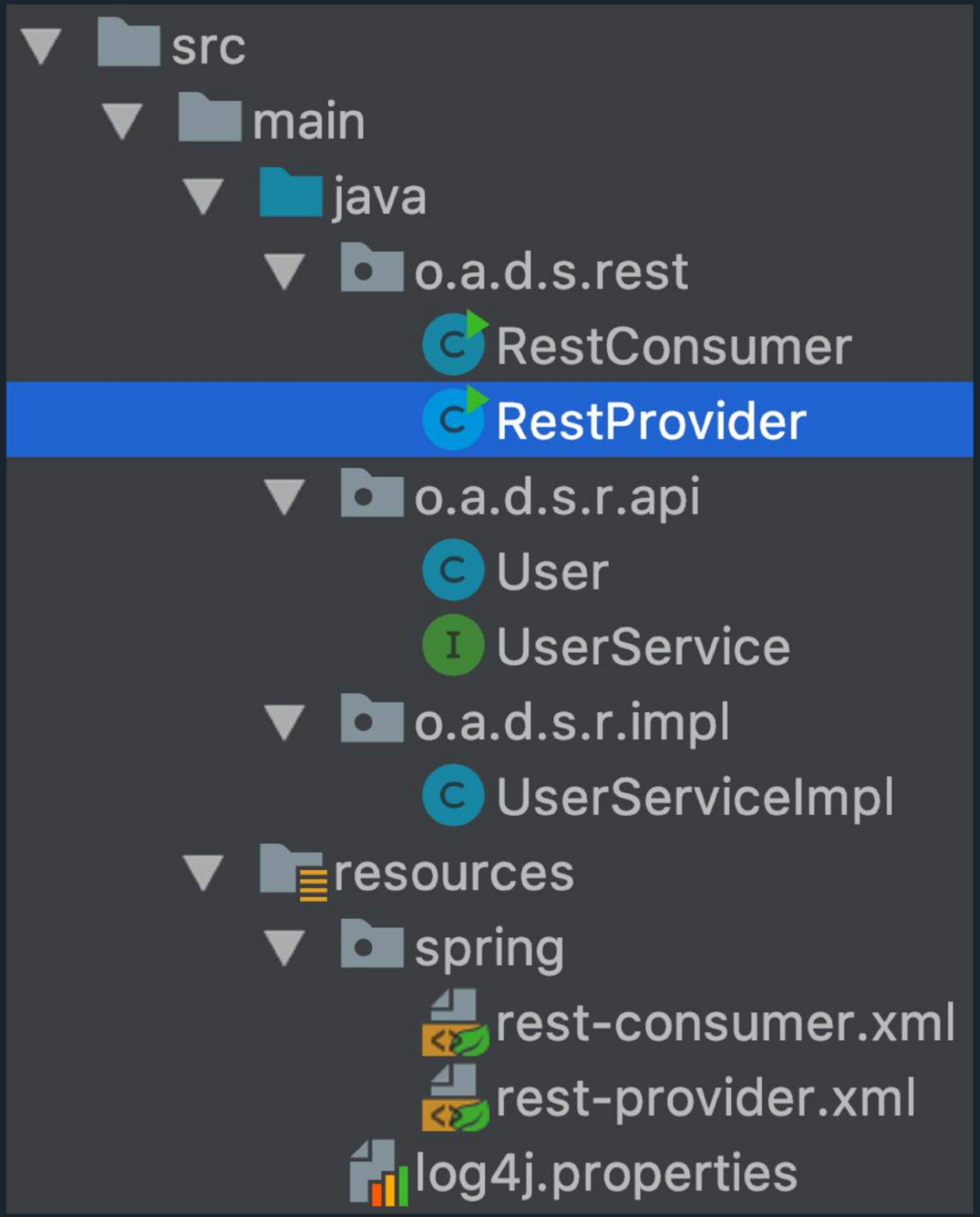
RESTPROVIDER.JAVA

```
public class RestProvider {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("spring/rest-provider.xml");
        context.start();
        System.in.read();
    }
}
```

Dubbo 日志

```
[08/01/19 01:55:23:023 CST] main  INFO config.AbstractConfig:  [DUBBO] Export
dubbo service org.apache.dubbo.samples.rest.api.UserService to url
rest://192.168.2.132:8080/org.apache.dubbo.samples.rest.api.UserService?anyhos
t=true&application=rest-
provider&bean.name=org.apache.dubbo.samples.rest.api.UserService&bind.ip=192.1
68.2.132&bind.port=8080&dubbo=2.0.2&generic=false&interface=org.apache.dubbo.s
amples.rest.api.UserService&methods=getUsers,getUser,registerUser&pid=21660&se
rver=netty&side=provider&timestamp=1546883723215, dubbo version: 2.6.5,
current host: 192.168.2.132
```

PROJECT LAYOUT



```
▼ src
  ▼ main
    ▼ java
      ▼ o.a.d.s.rest
          RestConsumer
          RestProvider
      ▼ o.a.d.s.r.api
          User
          UserService
      ▼ o.a.d.s.r.impl
          UserServiceImpl
    ▼ resources
      ▼ spring
          rest-consumer.xml
          rest-provider.xml
        log4j.properties
```

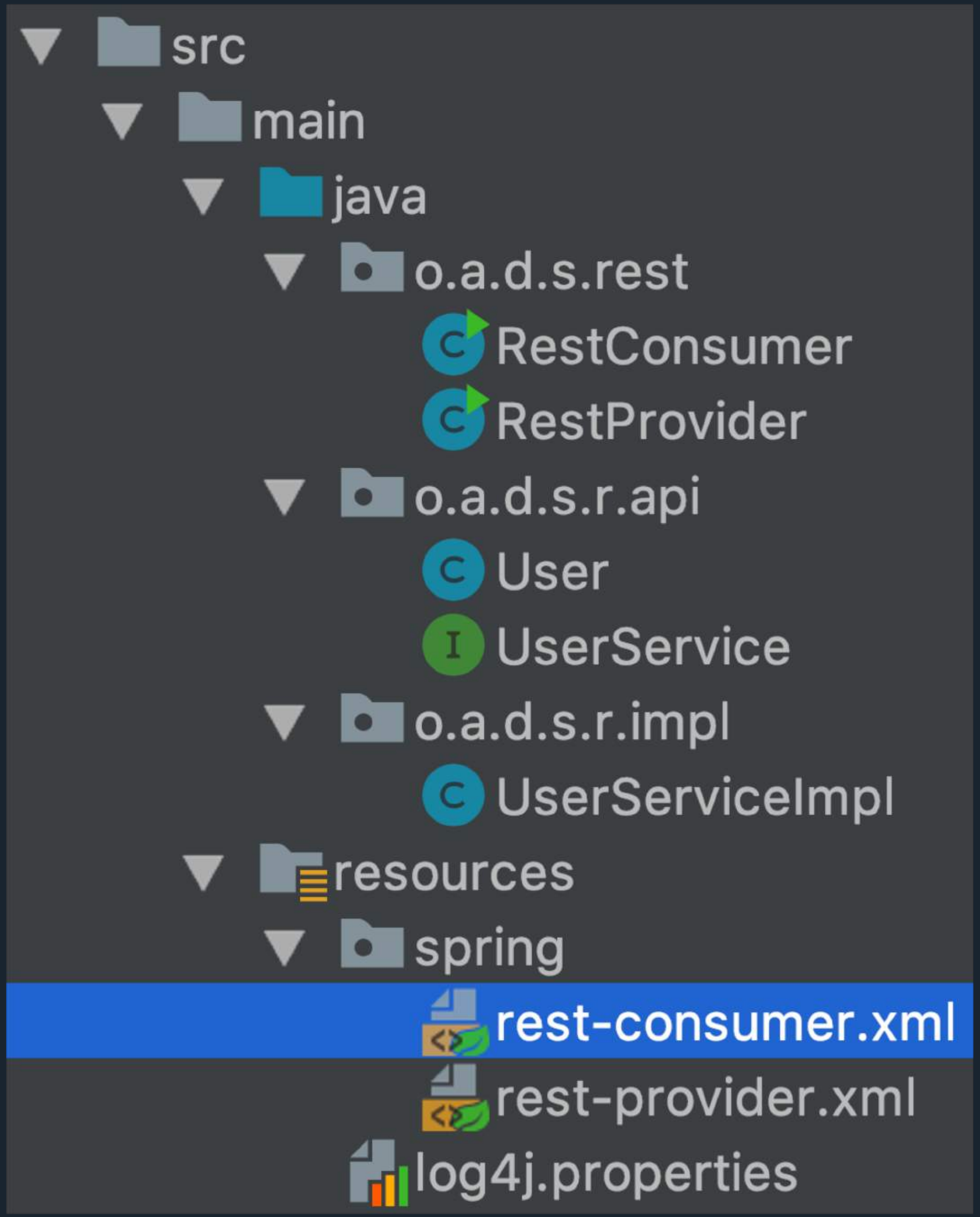
Dubbo REST 服务示例

步骤五、客户端配置

REST-CONSUMER.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans>
    <dubbo:application name="rest-consumer"/>
    <dubbo:registry address="zookeeper://127.0.0.1:2181"/>
    <dubbo:reference id="userService”
        interface="o.a.d.s.r.api.UserService"/>

    <!-- direct connect
    <dubbo:reference id="userService"
        interface="o.a.d.s.r.api.UserService"
        url="rest://localhost:8080/"/>
    -->

</beans>
```

PROJECT LAYOUT



```
▼ src
   ▼ main
      ▼ java
         ▼ o.a.d.s.rest
              RestConsumer
              RestProvider
         ▼ o.a.d.s.r.api
              User
              UserService
         ▼ o.a.d.s.r.impl
              UserServiceImpl
      ▼ resources
         ▼ spring
              rest-consumer.xml
              rest-provider.xml
           log4j.properties
```

Dubbo REST 服务示例

步骤六、启动客户端调用服务

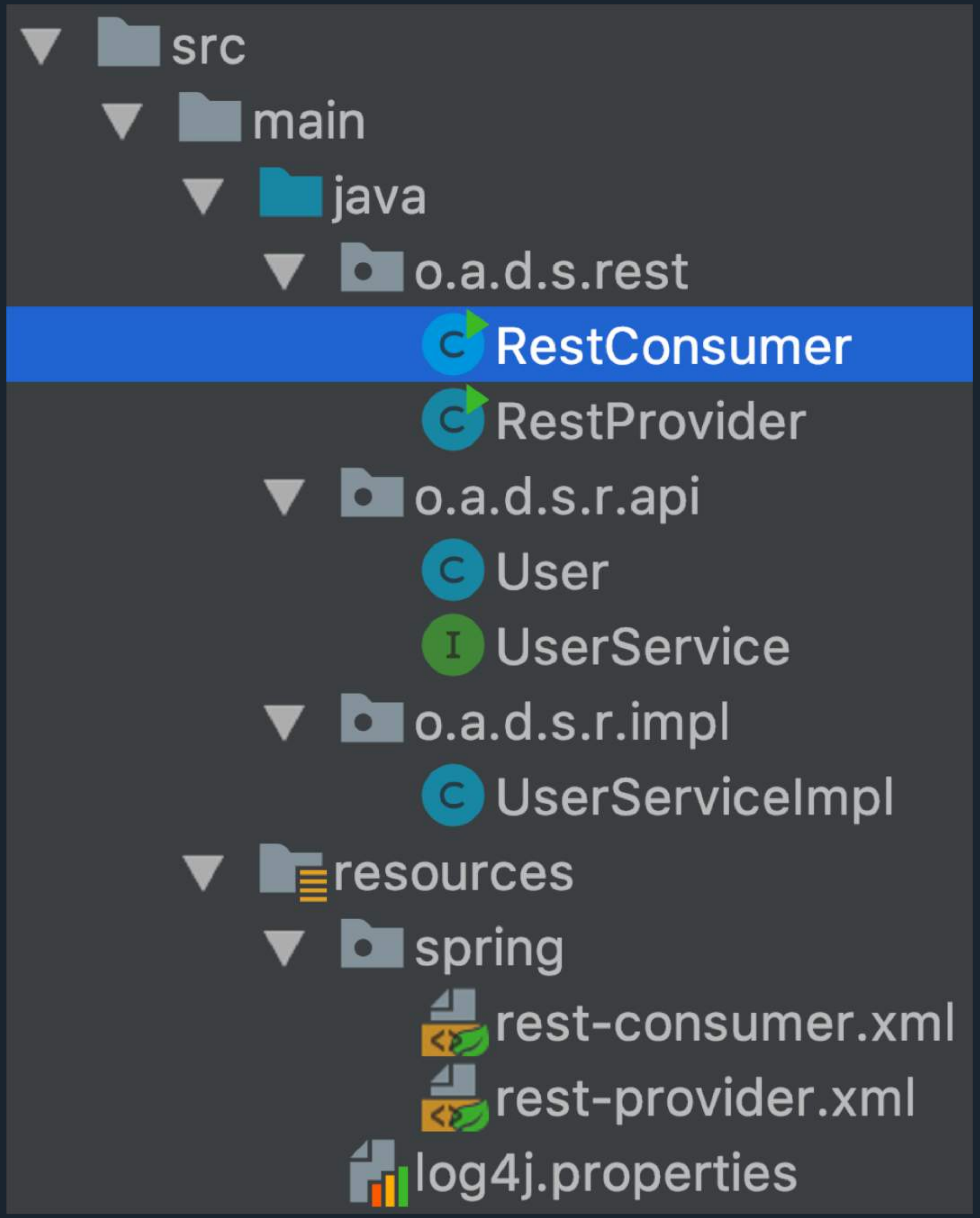
RESTCONSUMER.JAVA

```
public class RestConsumer {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("spring/rest-consumer.xml");
        context.start();
        UserService userService =
context.getBean("userService", UserService.class);
        System.out.println(userService.getUser(1L));
}
```

命令行

```
$ curl -X GET http://localhost:8080/users

[{"id":1,"name":"Tom"},{"id":2,"name":"Jerry"}]

$ curl -X POST -H "Content-Type: application/json" -d '{"id":1,"name":"Tom"}'
http://localhost:8080/users

1
```

PROJECT LAYOUT



```
▼ src
  ▼ main
    ▼ java
      ▼ o.a.d.s.rest
          RestConsumer
          RestProvider
      ▼ o.a.d.s.r.api
          User
          UserService
      ▼ o.a.d.s.r.impl
          UserServiceImpl
    ▼ resources
      ▼ spring
          rest-consumer.xml
          rest-provider.xml
        log4j.properties
```

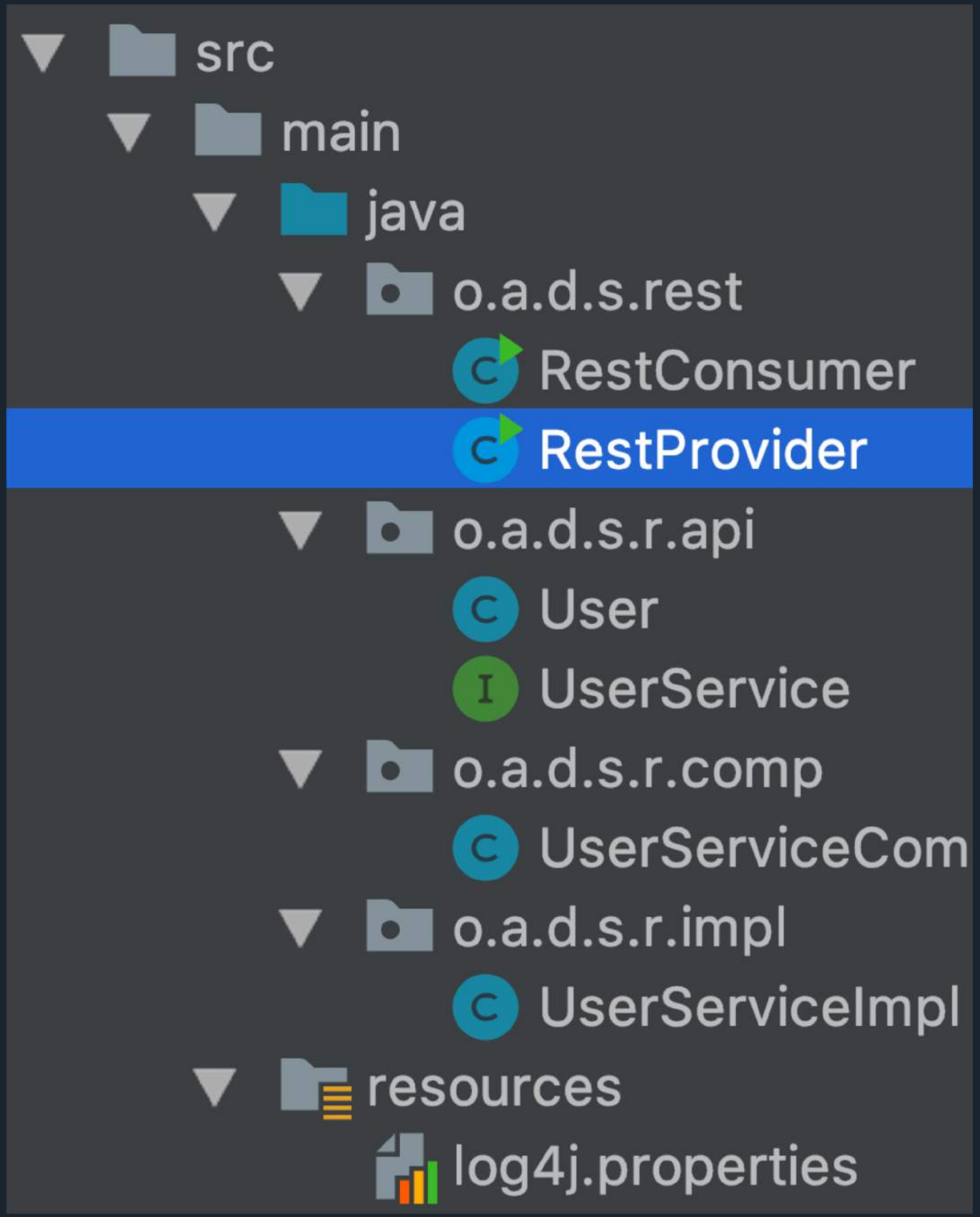
注解方式

服务端配置

Annotation

```
public class RestProvider {
    public static void main(String[] args) throws IOException {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(ProviderConfiguration.class); 1
        ...
    }

2   @Configuration
    @EnableDubbo(scanBasePackages = "o.a.d.s.r.impl")
    static class ProviderConfiguration {
        @Bean
        public ProtocolConfig protocolConfig() {
            ProtocolConfig protocolConfig = new ProtocolConfig();
            protocolConfig.setName("rest");
            return protocolConfig;
        }
        ...
    }
}

@Service 3
public class UserServiceImpl implements UserService { ... }
```

PROJECT LAYOUT



```
▼ src
  ▼ main
    ▼ java
      ▼ o.a.d.s.rest
          RestConsumer
          RestProvider
      ▼ o.a.d.s.r.api
          User
          UserService
      ▼ o.a.d.s.r.comp
          UserServiceCom
      ▼ o.a.d.s.r.impl
          UserServiceImpl
    ▼ resources
        log4j.properties
```

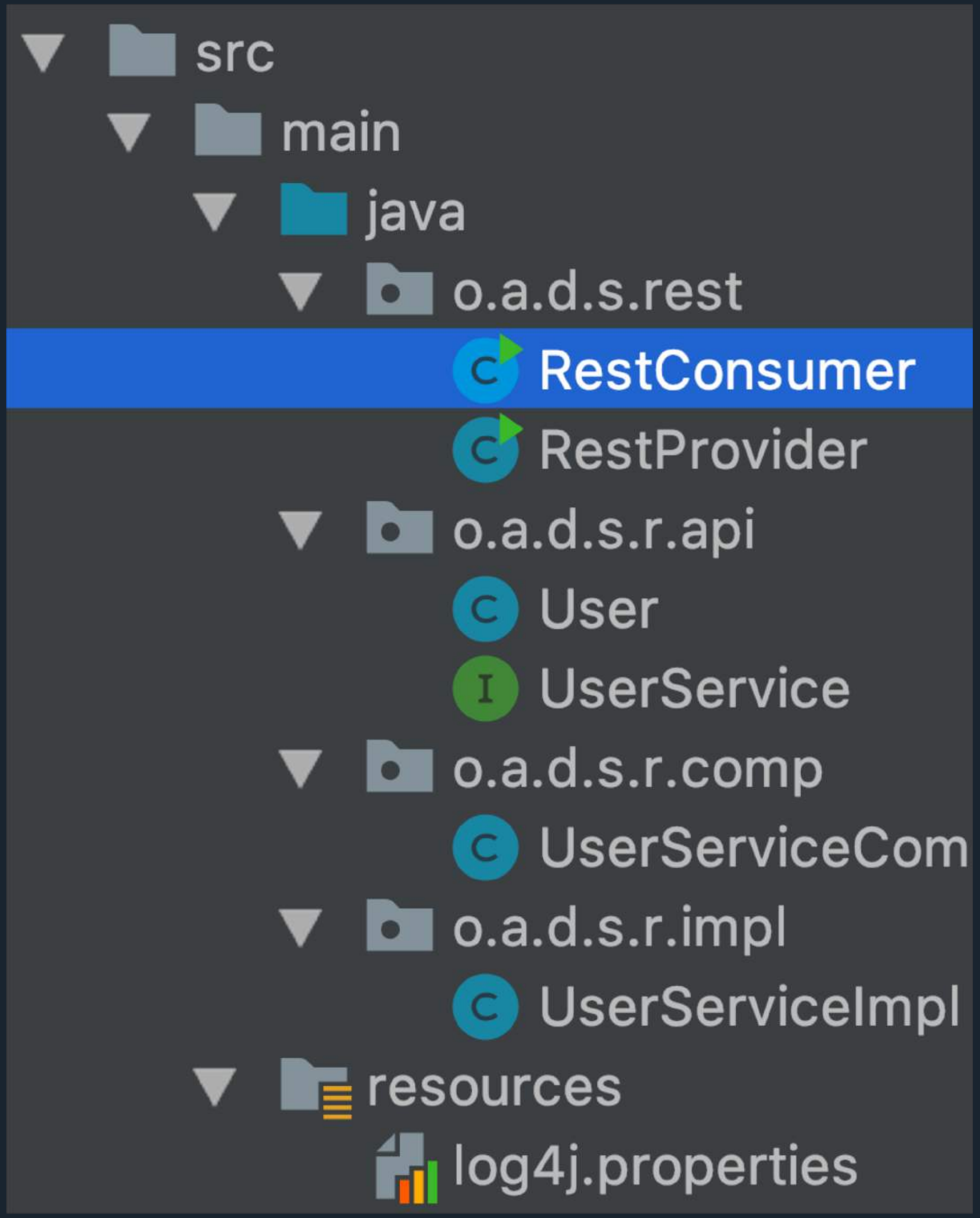
注解方式

客户端配置

Annotation

```
public class RestConsumer {
    public static void main(String[] args) {
        AnnotationConfigApplicationContext context = new
AnnotationConfigApplicationContext(ConsumerConfiguration.class);   1
        context.start();
        UserService userService =
context.getBean(UserServiceComponent.class);
        ...
    }

 2  @Configuration
    @EnableDubbo(scanBasePackages = "o.ap.d.s.r.comp")
    @ComponentScan({"o.a.d.s.r.comp"})
    static class ConsumerConfiguration { ... }
}

@Component  3
public class UserServiceComponent implements UserService {
 4  @Reference
    private UserService userService;
    public User getUser(Long id) { return userService.getUser(id); }
}
```

PROJECT LAYOUT



```
▼ src
    ▼ main
        ▼ java
            ▼ o.a.d.s.rest
                RestConsumer
                RestProvider
            ▼ o.a.d.s.r.api
                User
                UserService
            ▼ o.a.d.s.r.comp
                UserServiceCom
            ▼ o.a.d.s.r.impl
                UserServiceImpl
        ▼ resources
            log4j.properties
```

集成 SWAGGER

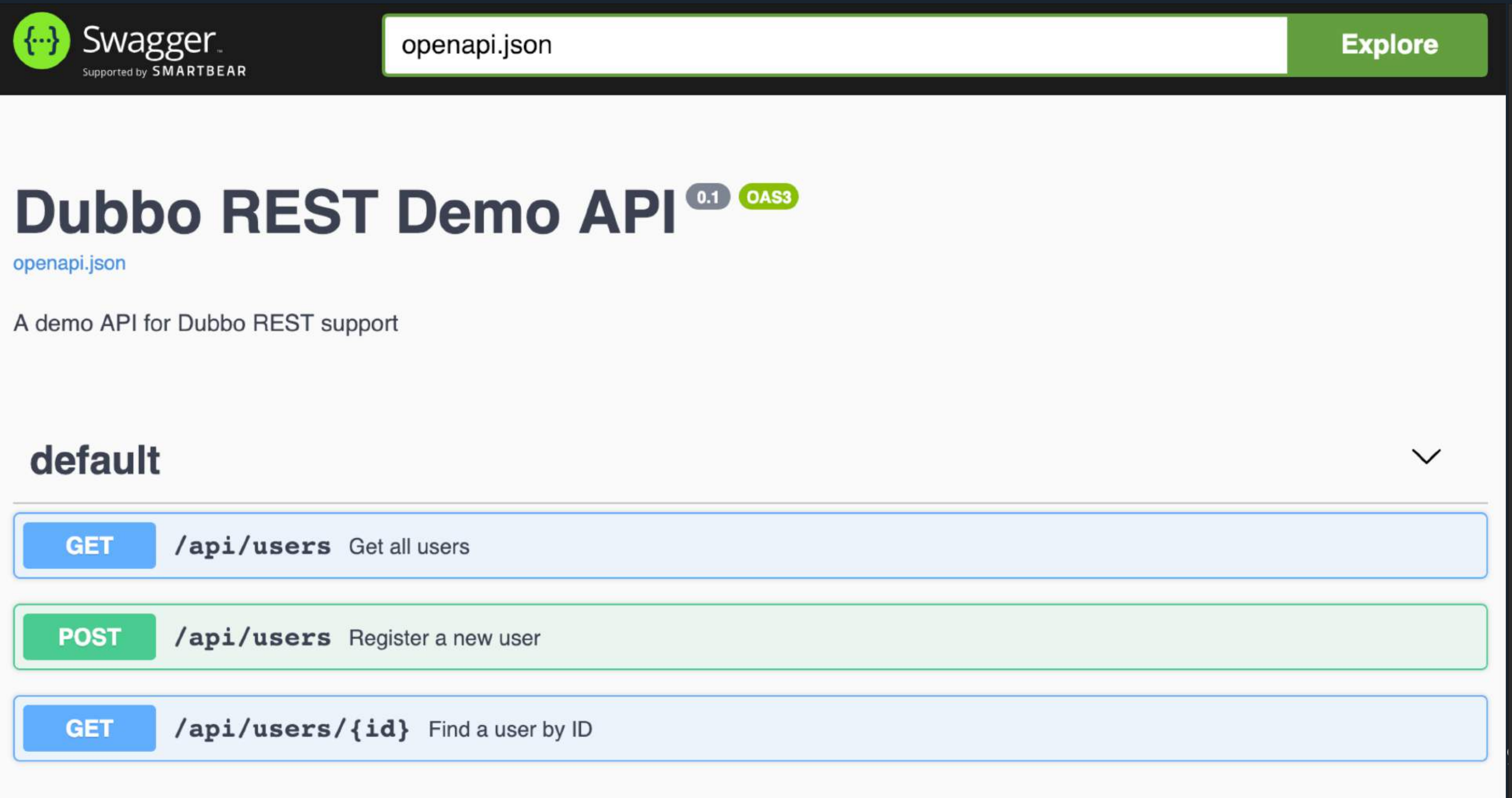
API 管理，文档与测试

SWAGGER UI

swagger.io 系列中的一个组件，提供 web 界面方便测试，也可以看成是 openapi 数据的可视化展现

OPEN API

Linux 基金会下的标准，起源于 Swagger Spec。Swagger UI 的数据来源于 openapi



Dubbo REST Demo API 0.1 OAS3
openapi.json

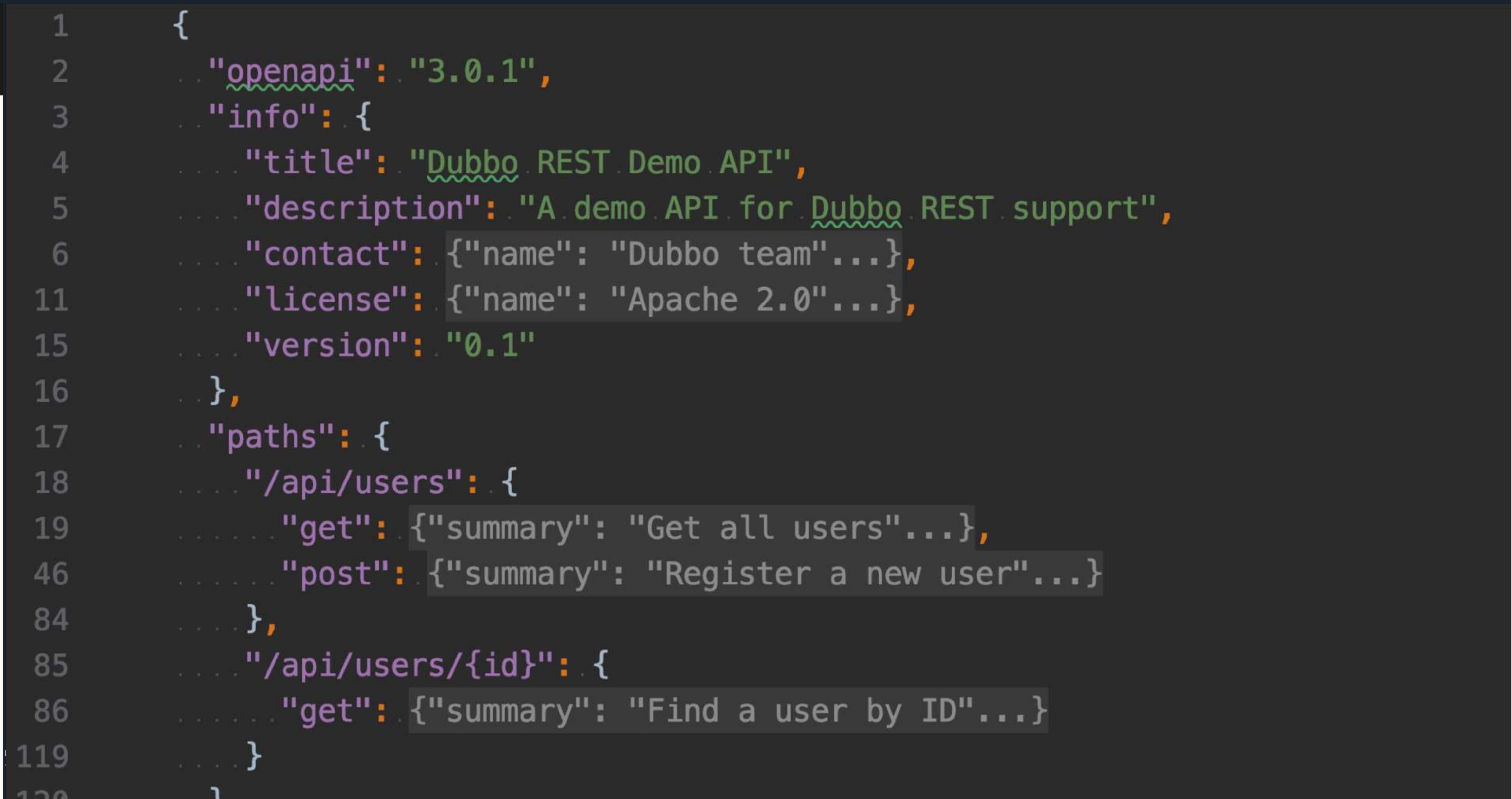
A demo API for Dubbo REST support

default

GET /api/users Get all users

POST /api/users Register a new user

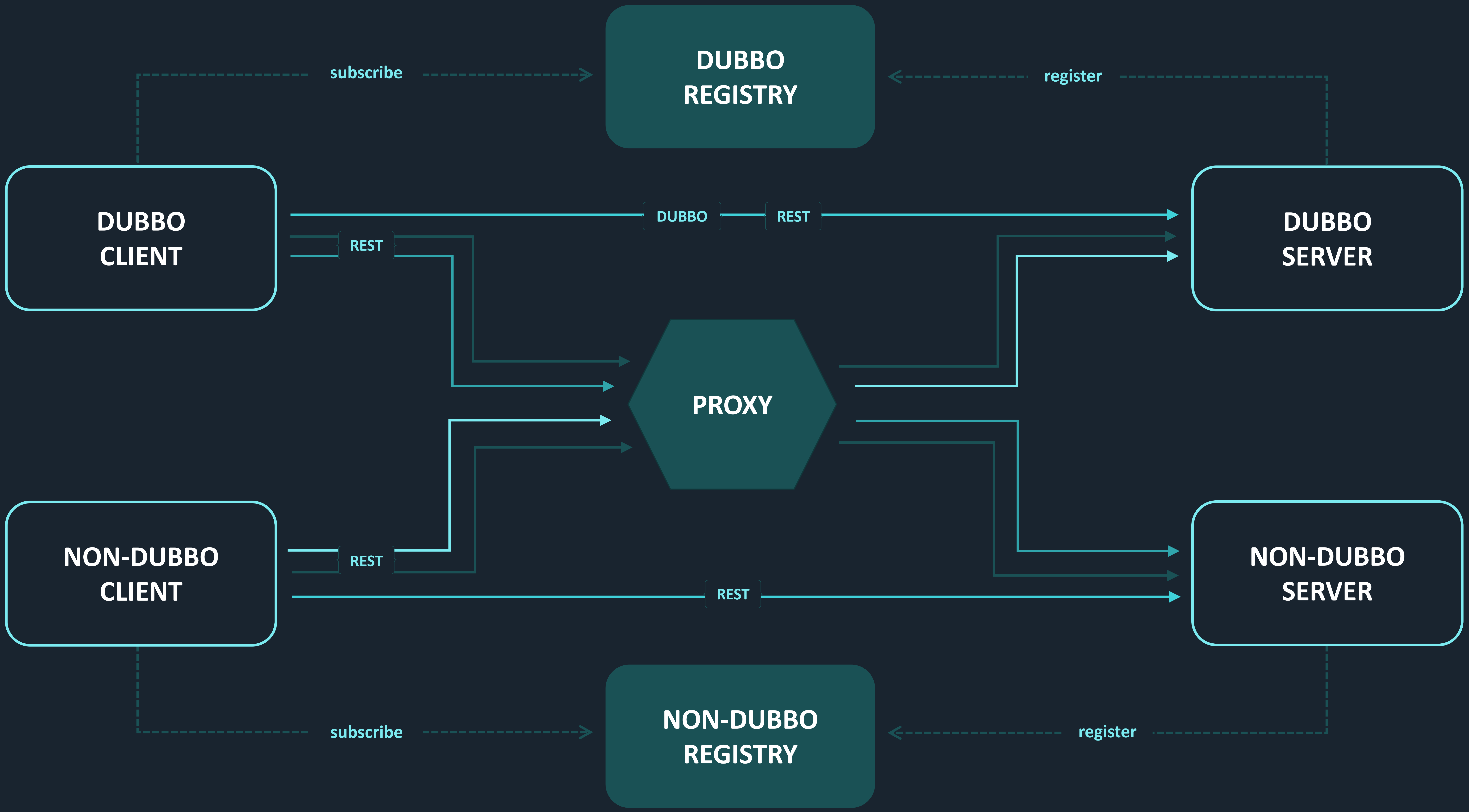
GET /api/users/{id} Find a user by ID



```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Dubbo REST Demo API",
    "description": "A demo API for Dubbo REST support",
    "contact": {"name": "Dubbo team"...},
    "license": {"name": "Apache 2.0"...},
    "version": "0.1"
  },
  "paths": {
    "/api/users": {
      "get": {"summary": "Get all users"...},
      "post": {"summary": "Register a new user"...}
    },
    "/api/users/{id}": {
      "get": {"summary": "Find a user by ID"...}
    }
```

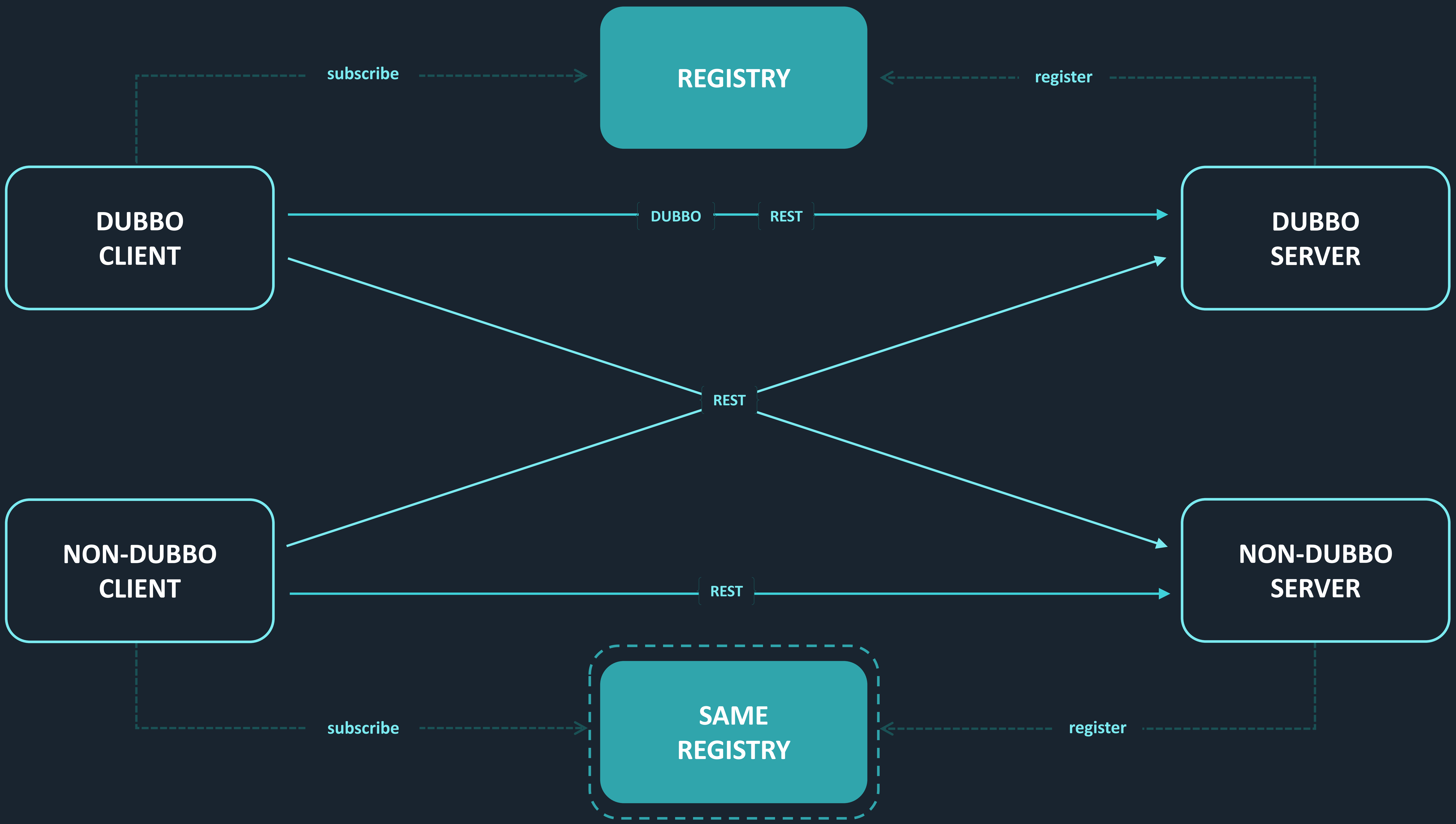
当前的部署

服务注册发现机制不同，需要 Proxy 做负载均衡



未来的部署

统一注册中心



HOME

dubbo.apache.org

GITHUB

github.com/apache/incubator-dubbo

github.com/dubbo

MAILING LIST

dev@dubbo.apache.org

WECHAT

northlatitute