

调用生产者服务

首先，开发服务的接口：

```
public class UserService {  
    void registerUser(User user);  
}
```

然后，开发服务的实现：

```
@Path("users")  
public class UserServiceImpl implements UserService {  
  
    @POST  
    @Path("register")  
    @Consumes(MediaType.APPLICATION_JSON)  
    public void registerUser(User user) {  
        // save the user...  
    }  
}
```

上面的服务实现代码非常简单，但是由于REST服务是要被发布到特定HTTP URL，供任意语言客户端甚至浏览器来访问，所以这里要額RS的标准annotation来做相关的配置：

@Path("users")：指定访问UserService的URL相对路径是/users，即http://localhost:8080/users

@Path("register")：指定访问registerUser()方法的URL相对路径是/register，再结合上一个@Path为UserService指定的路径，则调用Use的完整路径为http://localhost:8080/users/register

@POST：指定访问registerUser()用HTTP POST方法

@Consumes(MediaType.APPLICATION\_JSON)：指定registerUser()接收JSON格式的数据。REST框架会自动将JSON数据反序列化

最后，在spring配置文件中添加此服务，即完成所有服务开发工作：

```
<!-- 用rest协议在8080端口暴露服务 -->  
<dubbo:protocol name="rest" port="8080"/>  
  
<!-- 声明需要暴露的服务接口 -->  
<dubbo:service interface="xxx.UserService" ref="userService"/>  
  
<!-- 和本地bean一样实现服务 -->
```

Java EE - dubbox-consumer/src/main/java/com/itmayiedu/service/UserService.java - Eclipse

```
1 package com.itmayiedu.service;  
2  
3 import javax.ws.rs.GET;  
4  
5 @Path("users")  
6 public interface UserService {  
7     // 使用用户userID 查询 用户信息  
8     @GET  
9     @Path("{id : \d+}")  
10    @Produces(MediaType.APPLICATION_JSON)  
11    public String getUser(@PathParam("id") Long id);  
12 }
```

Console

```
Consumer [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\javaw.exe (2018年3月9日下午10:42:46)  
log4j:WARN No appenders could be found for logger (org.springframework.web.context.ContextLoader).  
log4j:WARN Please initialize the log4j system properly.  
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#nolog4j  
###消费者启动####
```

▪ 调用生产者服务

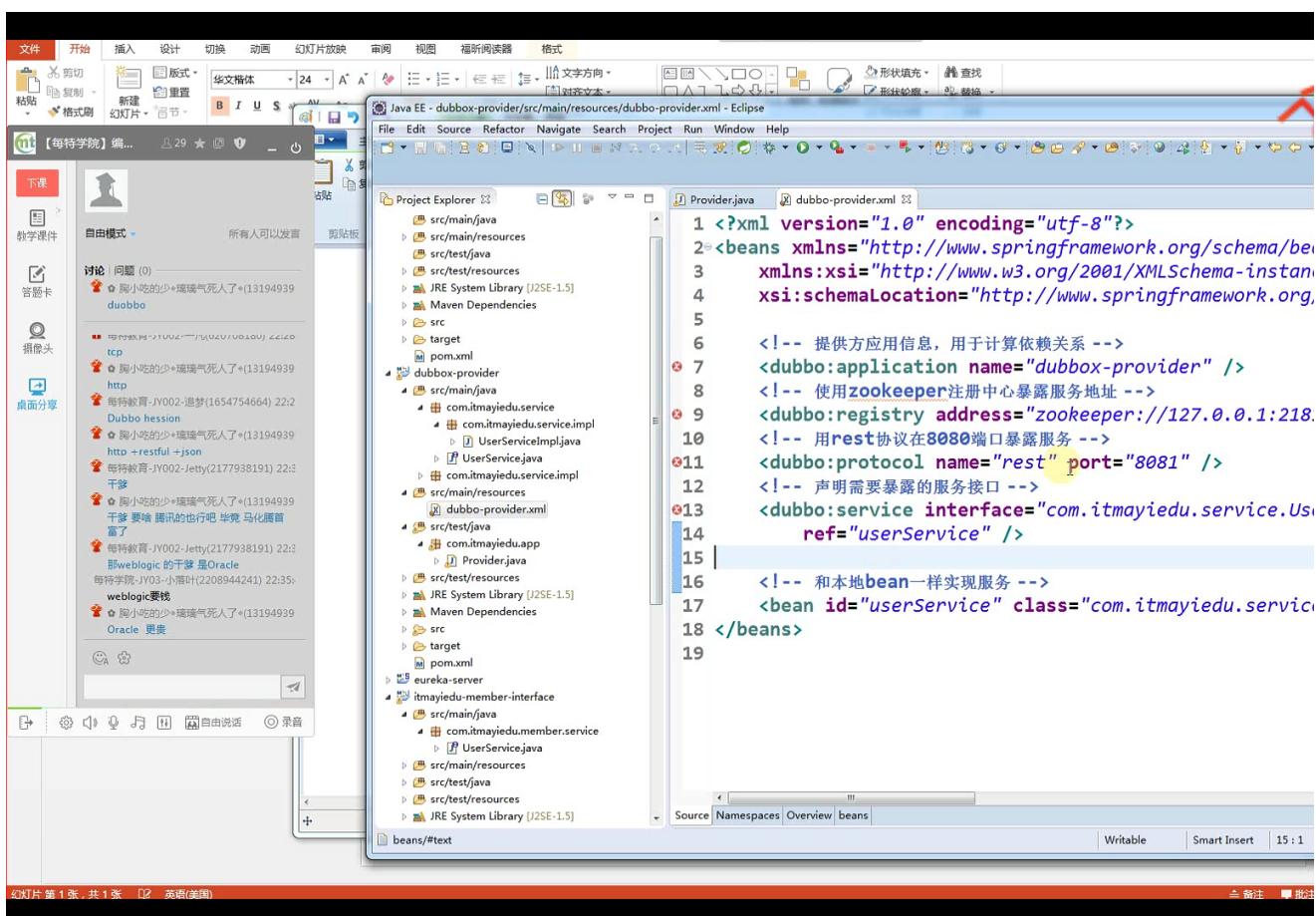
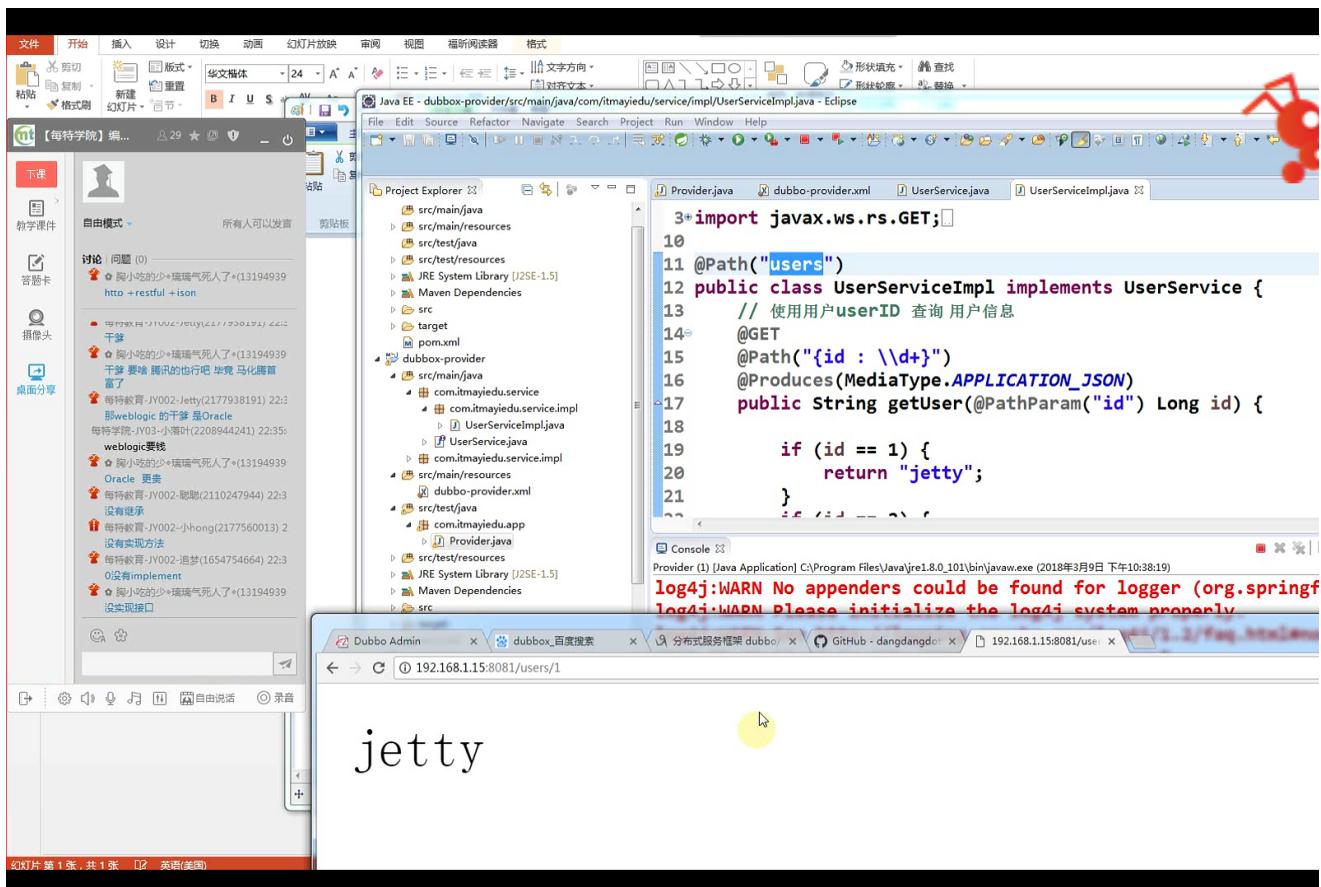
```
public class Consumer {  
    public static void main(String[] args) {  
        ClassPathXmlApplicationContext applicationContext = new  
        ClassPathXmlApplicationContext("dubbo-consumer.xml");  
        applicationContext.start();  
        System.out.println("###消费者启动####");  
        UserService userService=(UserService) applicationContext.getBean("userService");  
        System.out.println("消费者调用生产者服务开始");  
        String user = userService.getUser(1);  
        System.out.println("消费者调用生产者服务结束 user:"+user);  
    }  
}
```

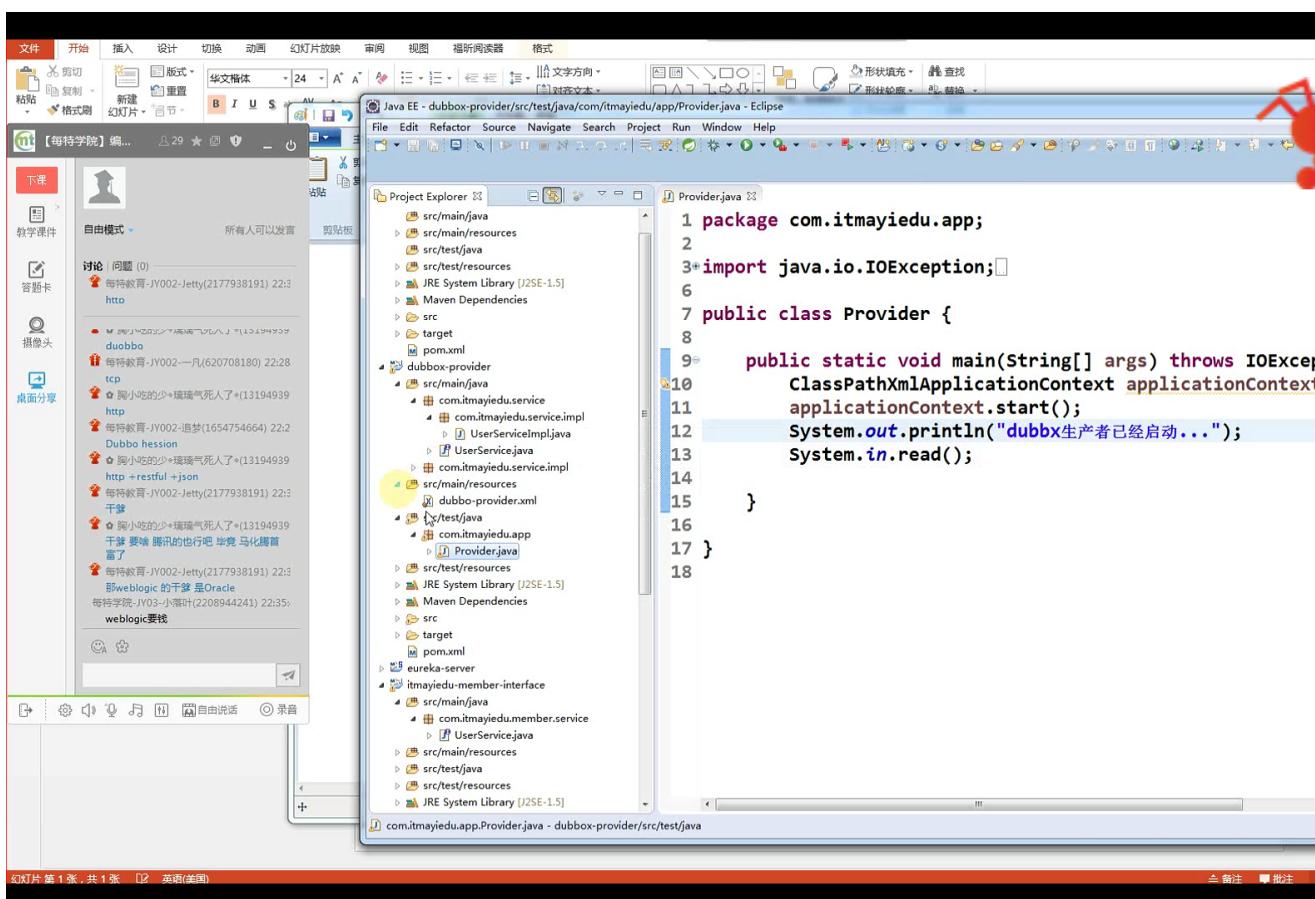
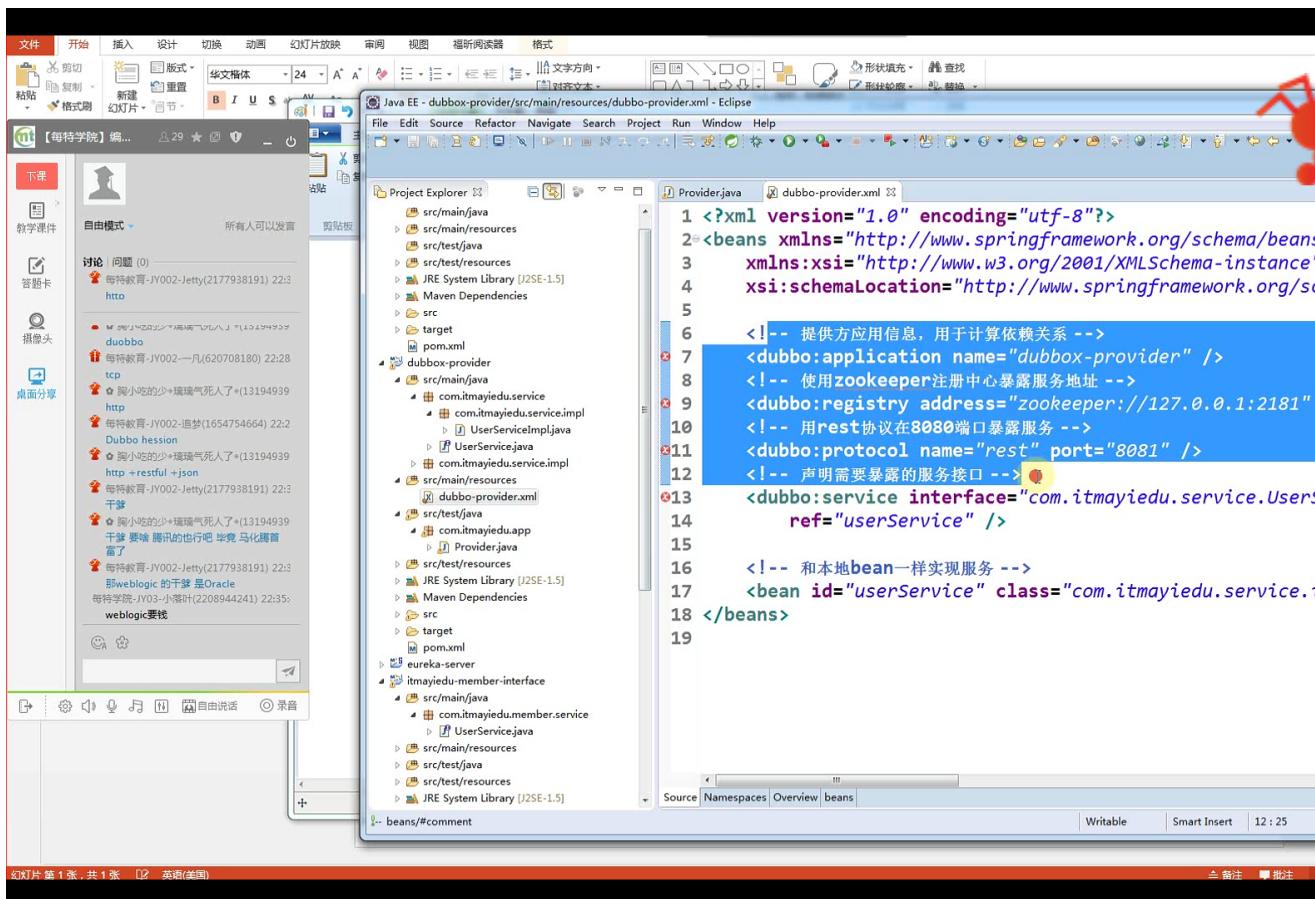
4859个字 中文(中国)

▪ 常见 Dubbo 面试题

```
1 package com.itmayiedu.service;  
2  
3* import javax.ws.rs.GET;  
10  
11 @Path("users")  
12 public interface UserService {  
13     // 使用用户userID 查询 用户信息  
14     @GET  
15     @Path("{id : \\\d+}")  
16     @Produces(MediaType.APPLICATION_JSON)  
17     public String getUser(@PathParam("id") Long id);  
18 }  
19
```

Provider (1) [Java Application] C:\Program Files\Java\jre1.8.0\_101\bin\javaw.exe (2018年3月9日 下午10:38:19)  
log4j:WARN No appenders could be found for logger (org.springframework.web.context.ContextLoader).  
log4j:WARN Please initialize the log4j system properly.  
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig  
2018-03-09 22:38:21.389::INFO: jetty-7.0.0.pre5  
2018-03-09 22:38:21.884::INFO: Started SelectChannelConnector  
dubbo生产者已经启动...





The screenshot shows the Eclipse IDE interface. On the left is the 'Project Explorer' view, which lists several Java projects under the 'dubbo-provider' project, including 'com.itmayiedu.service.impl' and 'com.itmayiedu.member-interface'. The main editor window displays the 'UserService.java' file, specifically the 'UserServiceImpl.java' class. A code completion dropdown menu is open over the line of code:

```
public String getUser(Long id) {
```

The dropdown shows the method signature again, followed by two options: 'if (id == 1)' and 'if (id == 2)'. The second option is highlighted with a blue selection bar. The status bar at the bottom right indicates 'Writable' and 'Smart Insert'.

The screenshot shows a web-based documentation or tutorial page. The title is '生产者环境搭建'. The left sidebar contains a navigation tree with sections like 'Dubbo概述', 'Dubbo架构', 'Dubbo环境搭建', and '常见Dubbo面试题'. The main content area has a section titled '▪ 定义 UserService.' with the following code snippet:

```
public interface UserService {  
    public String getUser(Integer id);  
}
```

Below this is another section titled '▪ UserServiceImpl.' with the following code snippet:

```
@Path("users")  
public class UserServiceImpl implements UserService {  
    @GET  
    @Path("{id : \\d+}")  
    @Produces(MediaType.APPLICATION_JSON)  
    public String getUser(@PathParam("id") Integer id) {  
        if (id == 1) {  
            return "yushengjun";  
        }  
        if (id == 2) {  
            return "jetty";  
        }  
        return null;  
    }  
}
```

The screenshot shows a Microsoft PowerPoint slide with a title "REST风格远程调用" (REST-style remote invocation). On the left, there is a sidebar with various icons for file operations like cut, copy, paste, and format. The main content area displays a snippet of XML code representing Maven dependencies for a RESTful service. The code includes sections for RestEasy, RestEasy-client, validation-api, RestEasy-jaxb-provider, RestEasy-netty, and Sun HTTP server.

```

<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-jaxrs</artifactId>
    <version>3.0.7.Final</version>
</dependency>
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-client</artifactId>
    <version>3.0.7.Final</version>
</dependency>
<dependency>
    <groupId>javax.validation</groupId>
    <artifactId>validation-api</artifactId>
    <version>1.0.0.GA</version>
</dependency>

<!-- 如果要使用json序列化 -->
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-jackson-provider</artifactId>
    <version>3.0.7.Final</version>
</dependency>

<!-- 如果要使用xml序列化 -->
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-jaxb-provider</artifactId>
    <version>3.0.7.Final</version>
</dependency>

<!-- 如果要使用netty server -->
<dependency>
    <groupId>org.jboss.resteasy</groupId>
    <artifactId>resteasy-netty</artifactId>
    <version>3.0.7.Final</version>
</dependency>

<!-- 如果要使用Sun HTTP server -->
<dependency>

```

This screenshot shows a Microsoft PowerPoint slide with a title "Dubbox now means Dubbo eXtensions. If you know java, javax and dubbo, you know what dubbox is :)" and a subtitle "Dubbox adds features like RESTful remoting, Kryo/FST serialization, etc to the popular dubbo service framework. It's been used by several projects of dangdang.com, which is one of the major e-commerce companies in China." Below the title, there is a section titled "主要贡献者" (Main Contributors) listing individuals such as Shen Li, Wang Yuxuan, Ma Jin Kai, Dylan, and Kangfoo. A note at the bottom encourages users to discuss technical issues on the GitHub issues page. The slide also highlights the "Dubbox当前的主要功能" (Current main features of Dubbox), which include support for REST-style remote invocation, Kryo and FST serialization, Jackson JSON serialization, and embedded Tomcat HTTP remoting.

Dubbox now means Dubbo eXtensions. If you know java, javax and dubbo, you know what dubbox is :)

Dubbox adds features like RESTful remoting, Kryo/FST serialization, etc to the popular dubbo service framework. It's been used by several projects of [dangdang.com](#), which is one of the major e-commerce companies in China.

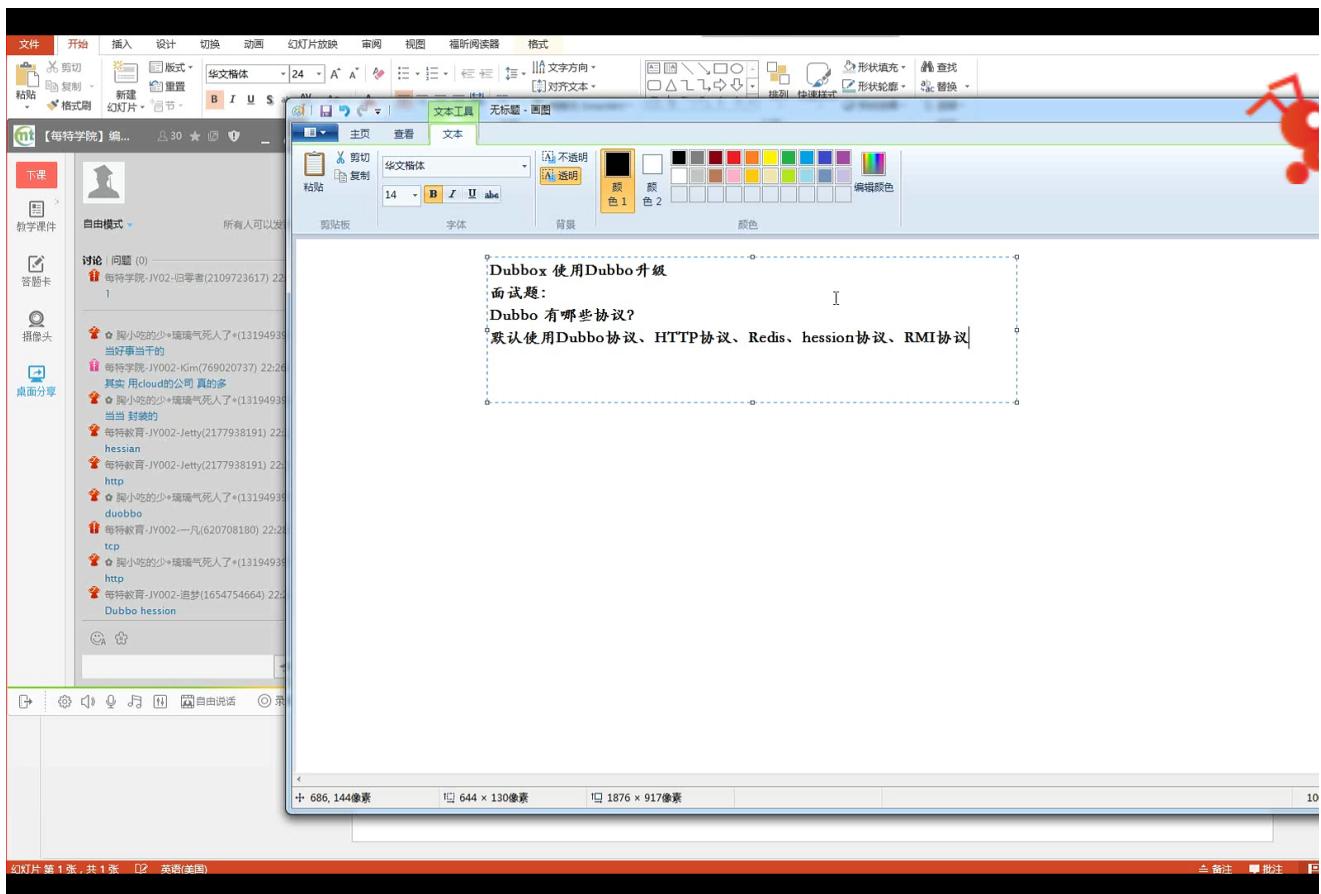
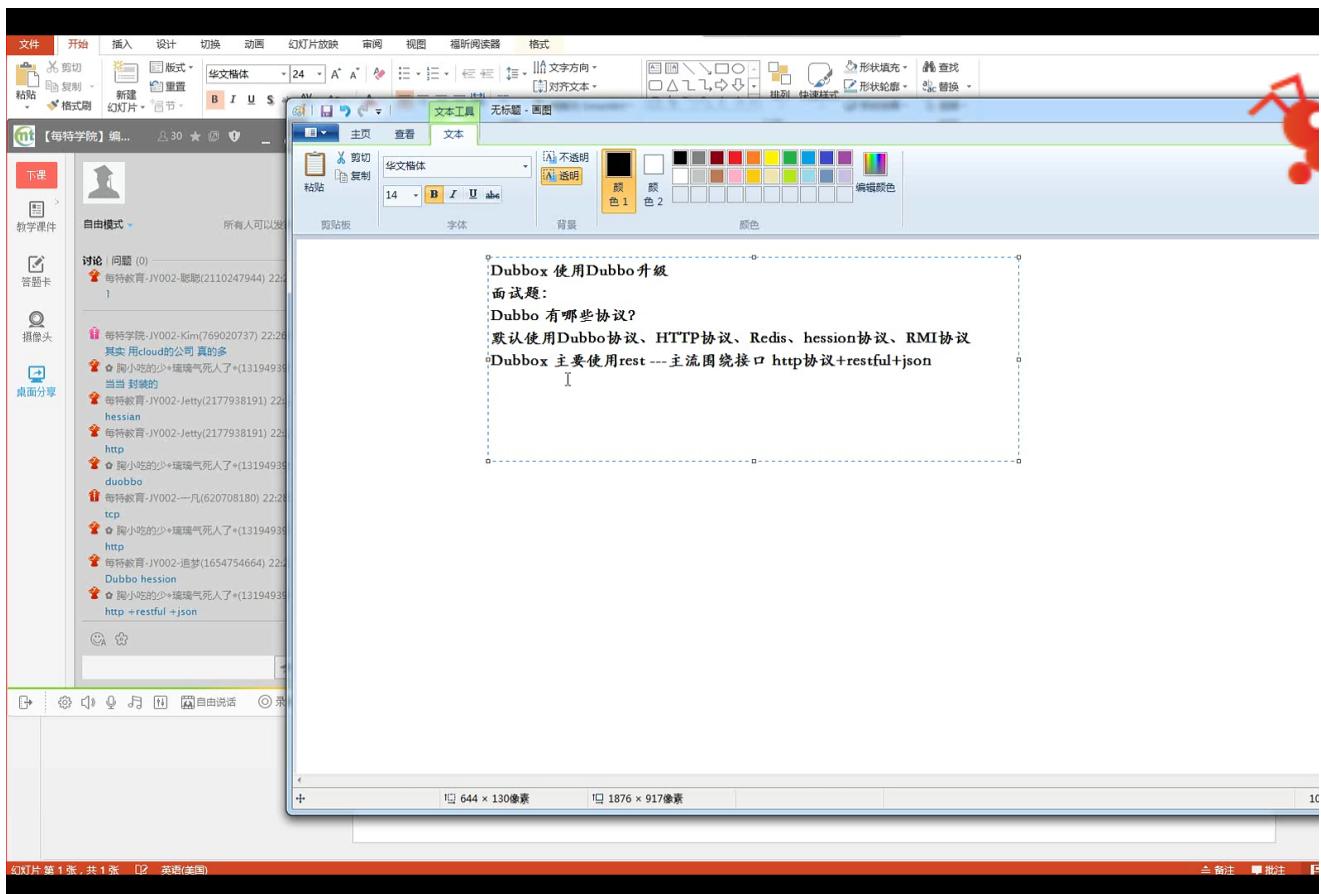
### 主要贡献者

- 沈理 当当网 [shenli@dangdang.com](mailto:shenli@dangdang.com)
- 王宇轩 当当网 [wangyuxuan@dangdang.com](mailto:wangyuxuan@dangdang.com)
- 马金凯 韩都衣舍 [majinkai@handu.com](mailto:majinkai@handu.com)
- Dylan 独立开发者 [dinguangx@163.com](mailto:dinguangx@163.com)
- Kangfoo 独立开发者

有技术问题请移步此处讨论 <https://github.com/dangdangdotcom/dubbox/issues>

### Dubbox当前的主要功能

- 支持REST风格远程调用 (HTTP + JSON/XML) :** 基于非常成熟的JBoss RestEasy框架，在dubbo中实现了REST风格 (HTTP + JSON/XML) 的远程调用，以显著简化企业内部的跨语言交互，同时显著简化企业对外的Open API、无线API甚至AJAX服务端等等的开发。事实上，这个REST调用也使得Dubbo可以对当今特别流行的“微服务”架构提供基础性支持。另外，REST调用也达到了比较高的性能，在基准测试下，HTTP + JSON与Dubbo 2.x默认的RPC协议（即TCP + Hessian2二进制序列化）之间只有1.5倍左右的差距，详见文档中的基准测试报告。
- 支持基于Kryo和FST的Java高效序列化实现 :** 基于当今比较知名的Kryo和FST高性能序列化库，为Dubbo默认的RPC协议添加新的序列化实现，并优化调整了其序列化体系，比较显著的提高了Dubbo RPC的性能，详见文档中的基准测试报告。
- 支持基于Jackson的JSON序列化 :** 基于业界应用最广泛的Jackson序列化库，为Dubbo默认的RPC协议添加新的JSON序列化实现。
- 支持基于嵌入式Tomcat的HTTP remoting体系 :** 基于嵌入式tomcat实现dubbo的HTTP remoting体系 (即dubbo-remoting-http)，用以逐步取代Dubbo中旧版本的嵌入式Jetty，可以显著的提高REST等的远程调用性能，并将Servlet API的支持从2.5升级到3.1。（注：除了REST，dubbo中的WebServices、Hessian、HTTP Invoker等协议都基于这个HTTP remoting体系。）



The screenshot displays a dual-monitor setup. The left monitor shows a Microsoft Teams meeting interface with a list of participants and their messages. The right monitor shows the Dubbo Admin interface for editing dynamic configurations.

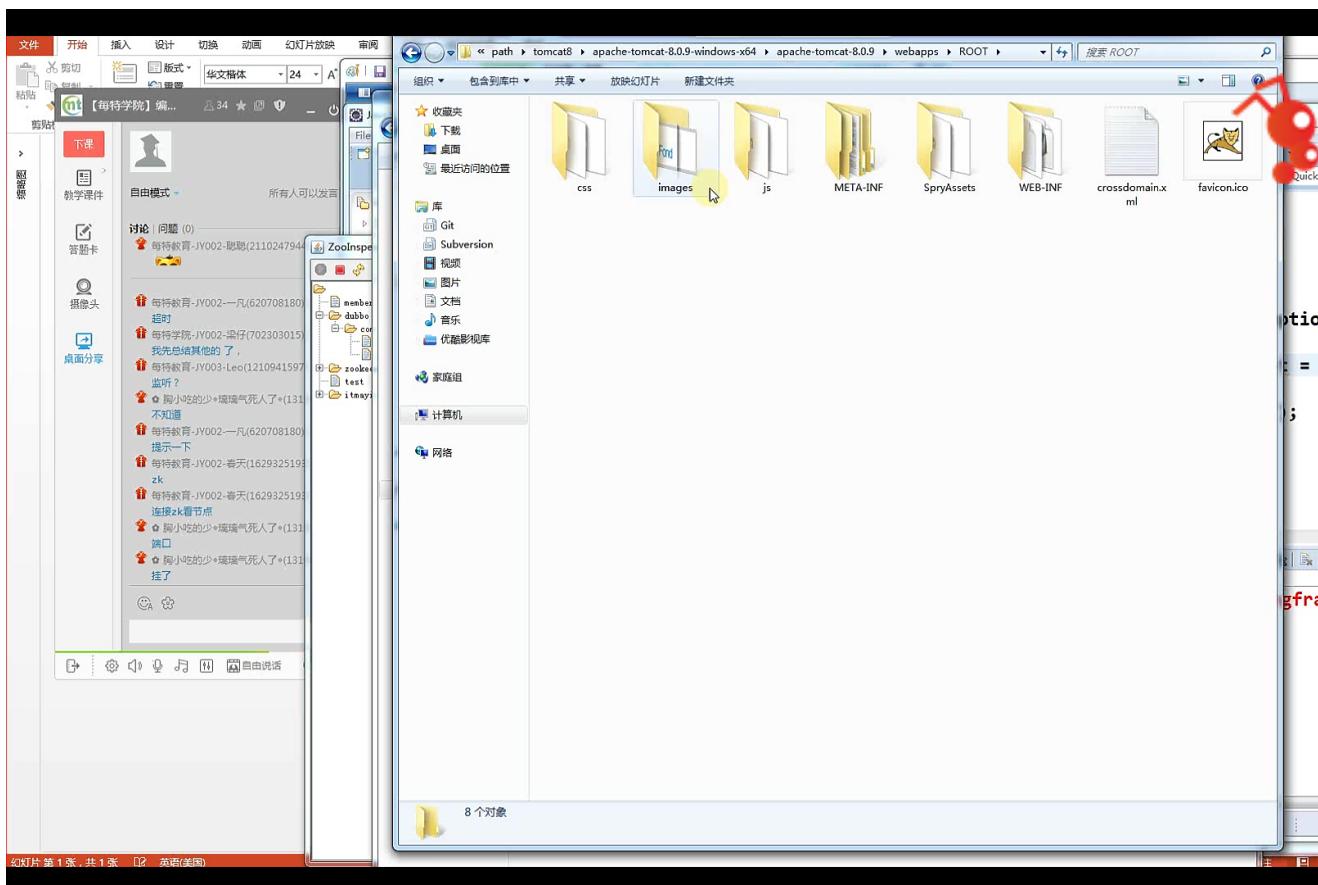
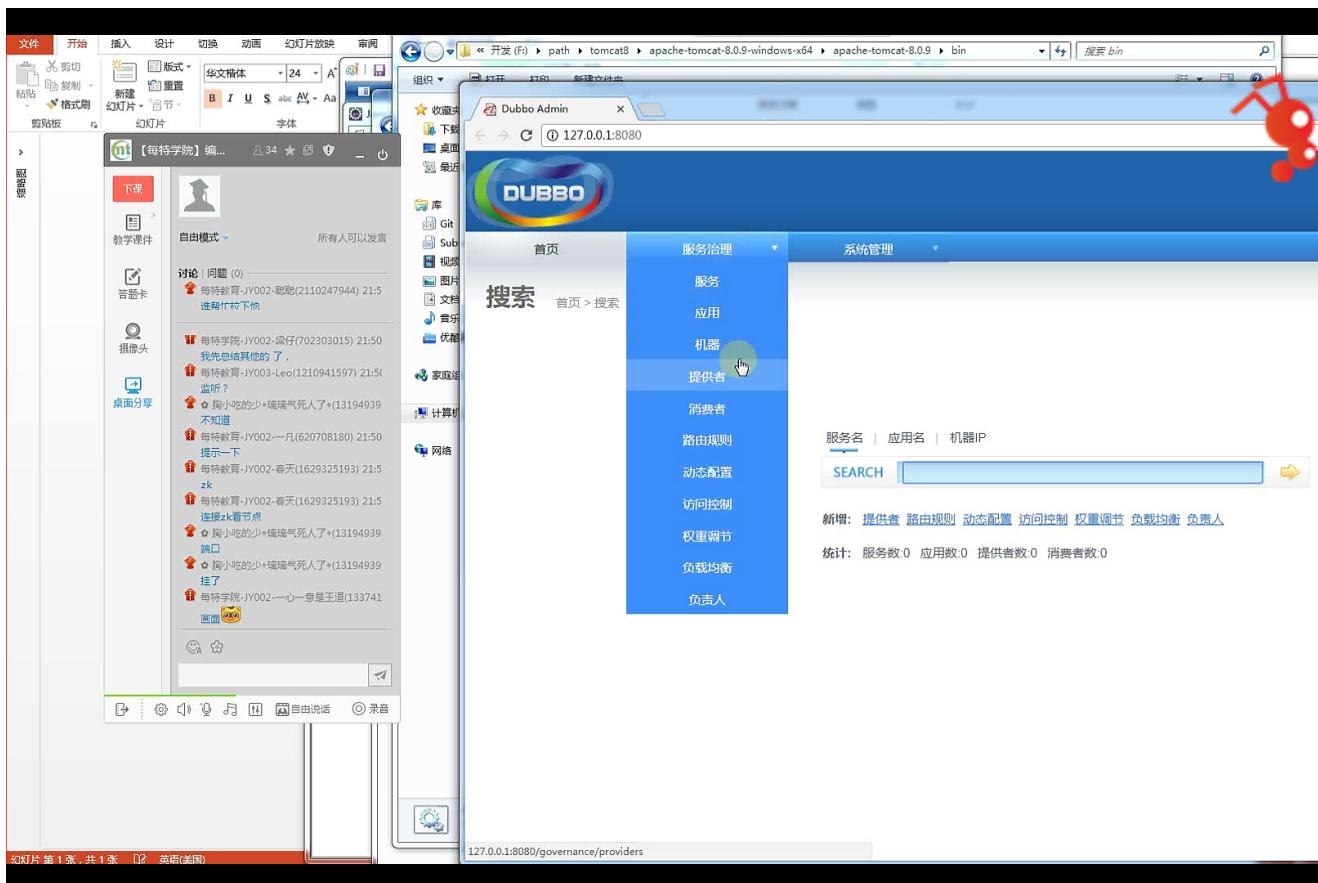
**Dubbo Admin Interface (Right Monitor):**

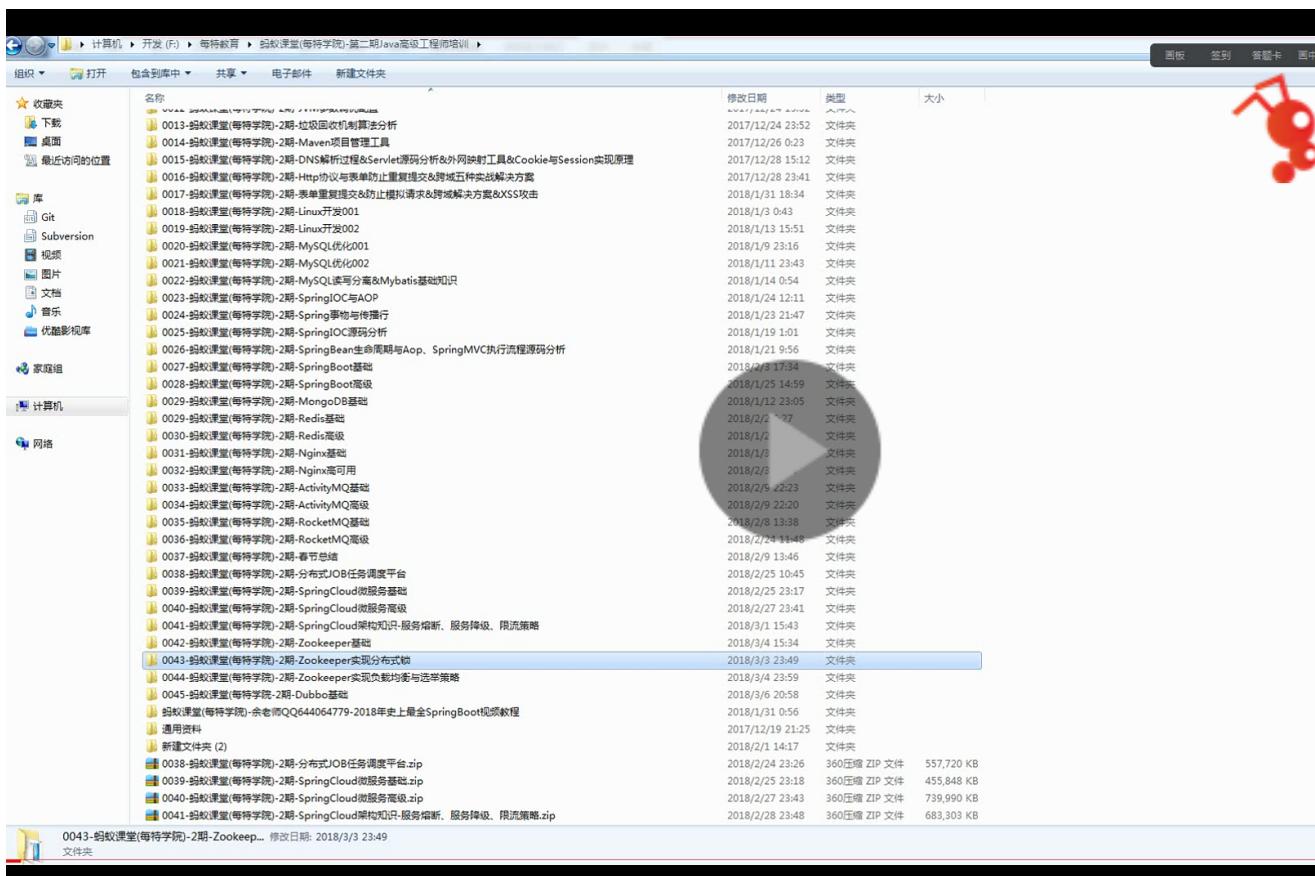
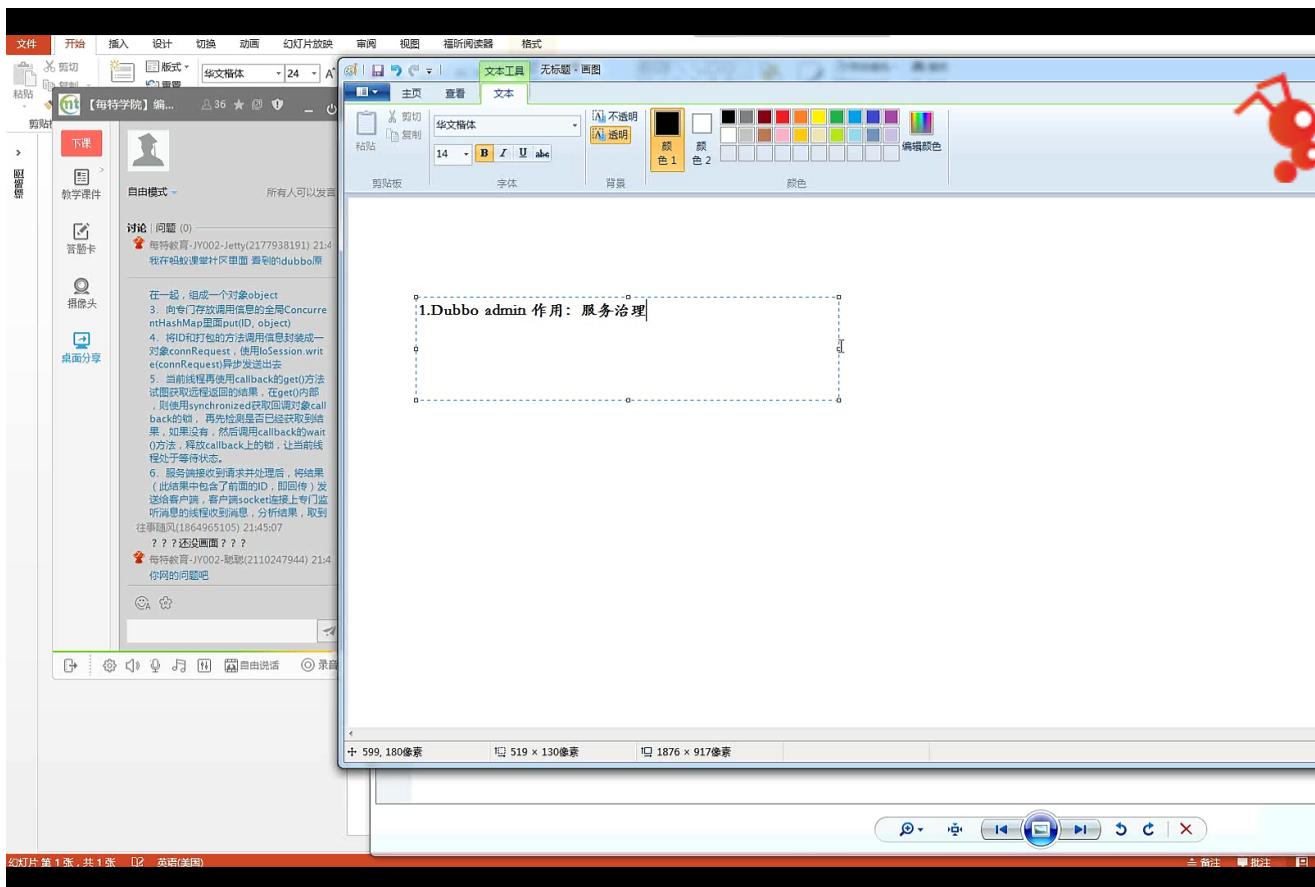
- URL:** 127.0.0.1:8080/governance/overrides/5/edit
- Service Name:** com.itmayiedu.member.service.UserService
- Status:** 启用 (Enabled)
- Consumer Application Name:** 192.168.1.15:29015 (不填表示对消费者应用的所有机器生效)
- Dynamic Configuration:** weight (参数值: 200)
- Service Mock:** 容错 (容错) - Example: return null/empty/JSON or throw com.foo.

The screenshot displays a dual-monitor setup. The left monitor shows a Microsoft Teams meeting interface with a list of participants and their messages. The right monitor shows the Dubbo Admin interface for viewing provider details.

**Dubbo Admin Interface (Right Monitor):**

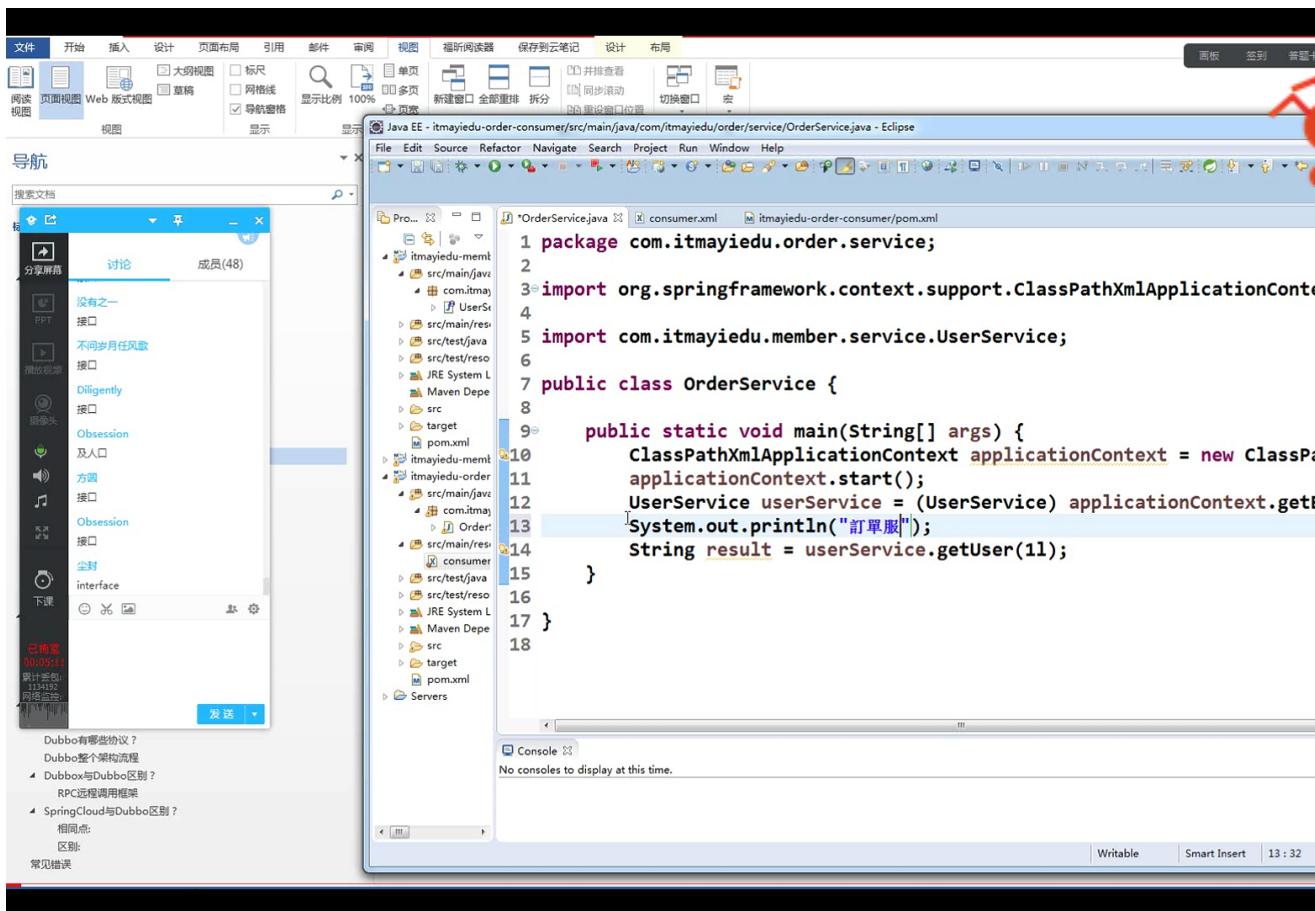
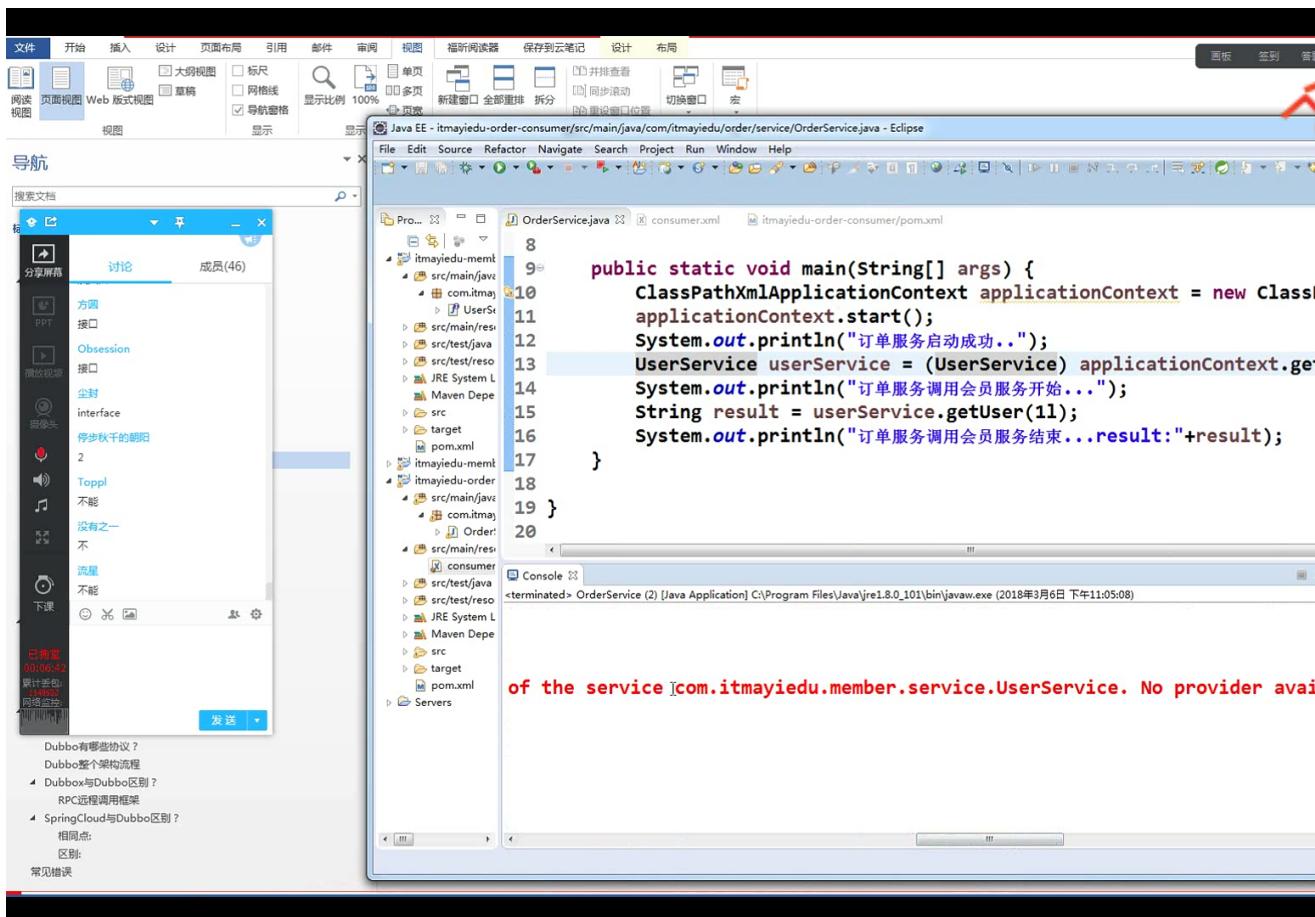
- URL:** 127.0.0.1:8080/governance/services/com.itmayiedu.member.service.UserService/providers/2
- Provider Details:**
  - 服务地址: dubbo://192.168.1.15:29015/com.itmayiedu.member.service.UserService?anyhost=true&application=provider&dubbo=2.5.6&generic=false&interface=com.itmayiedu.member.service.UserService&methods=getUser&pid=1352&side=provider&tamp=1520604328922
  - 动态配置:
  - 主机名: (windows10.microdone.cn)192.168.1.15:29015
  - 所属端: provider
  - 应用名: provider
  - 方法列表: getUser
  - Dubbo版本: 2.5.6
  - 进程号: 1352
  - 接口名: com.itmayiedu.member.service.UserService
  - generic: false
  - 绑定所有IP: true
  - 时间戳: 2018-03-09 22:05:28 (1520604328922)
  - 类型: 动态





The screenshot shows a course page titled "05 分布式专题" (Distributed Topic) from the "Java架构师-互联网高级" section. The page includes a brief introduction: "专题简介:Java高薪必备核心技术高并发、高可用、分布式、消息中间件". It features several sections: "分布式架构基础" (with topics like "分布式架构场景演变过程" and "单点架构与面向服务架构关系"), "分布式架构中间件与高可用" (with topics like "消息中间件发布订阅与点对点通讯方式" and "消息中间件异步与同步通讯方式区别"), "分布式架构技术" (with topics like "RPC网络通讯技术(WebService、RMI、Http)" and "使用Dubbo实现分布式架构与实战"), "分布式常见解决方案" (with topics like "分布式全局ID生成策略" and "分布式事物解决方案"), and a summary section at the bottom.

The screenshot shows a course page titled "SpringCloud" from the "Java架构师-互联网高级" section. The page includes a brief introduction: "专题简介:最新流行微服务架构模式, 揭秘 SpringCloud核心技术". It features two main sections: "SpringBoot" (with topics like "面向服务架构与微服务架构区别" and "SpringBoot与SpringCloud区别") and "SpringCloud" (with topics like "SpringCloud整体架构介绍" and "SpringCloud Eureka注册中心"). At the bottom, there is a summary section titled "微服务架构高级" (Advanced Microservices Architecture) with topics like "SOA面向服务架构与微服务架构区别" and "如何自己设计一套微服务框架".



配置文件参数

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
       xsi:schemaLocation="http://www.springframework.org/schema/beans/spring-beans.xsd http://code.alibabatech.com/schema/dubbo
       http://code.alibabatech.com/schema/dubbo/dubbo.xsd">

    <!-- 消费方应用名，用于计算依赖关系，不是匹配条件，不要与提供方一样 -->
    <dubbo:application name="consumer" />

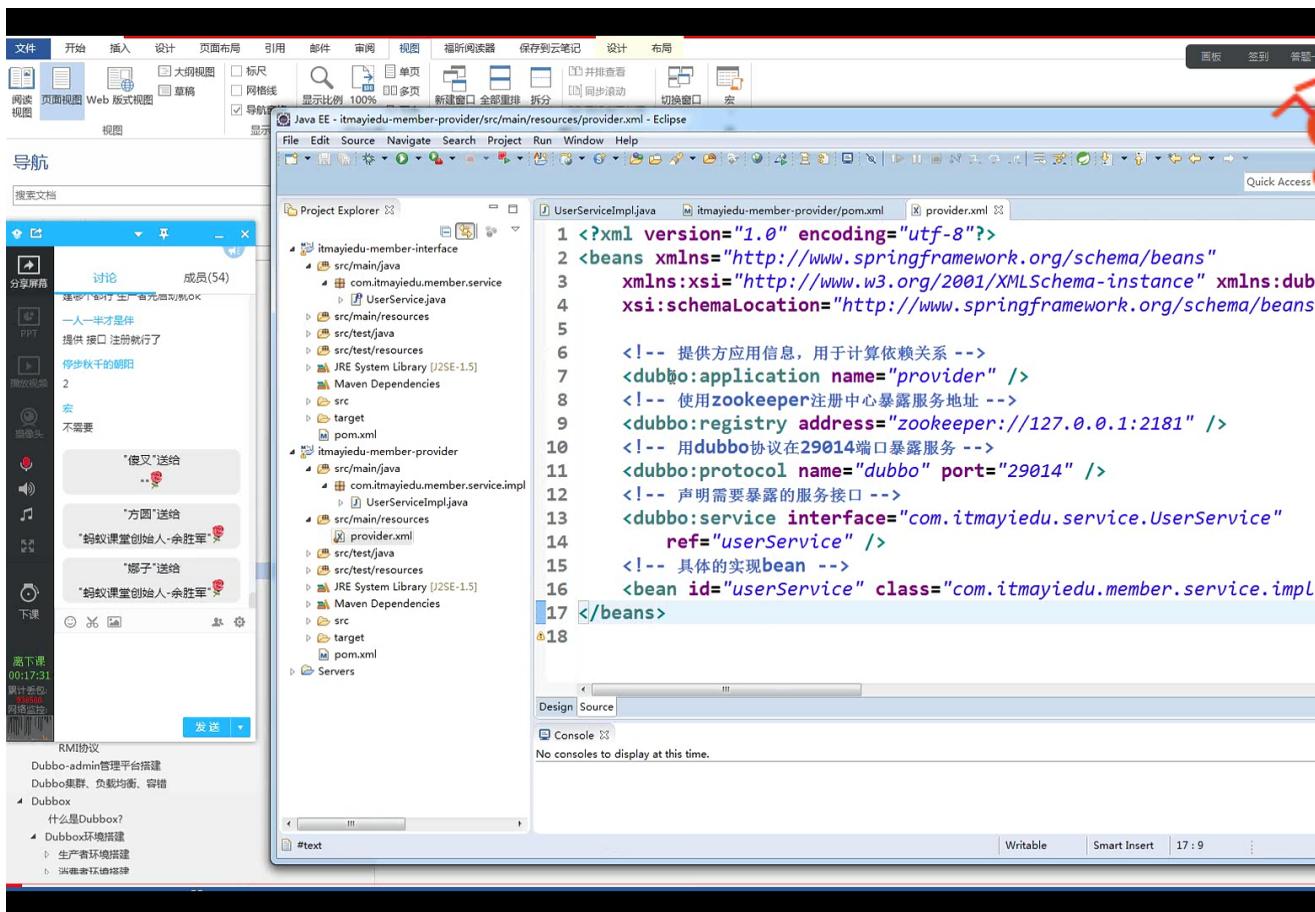
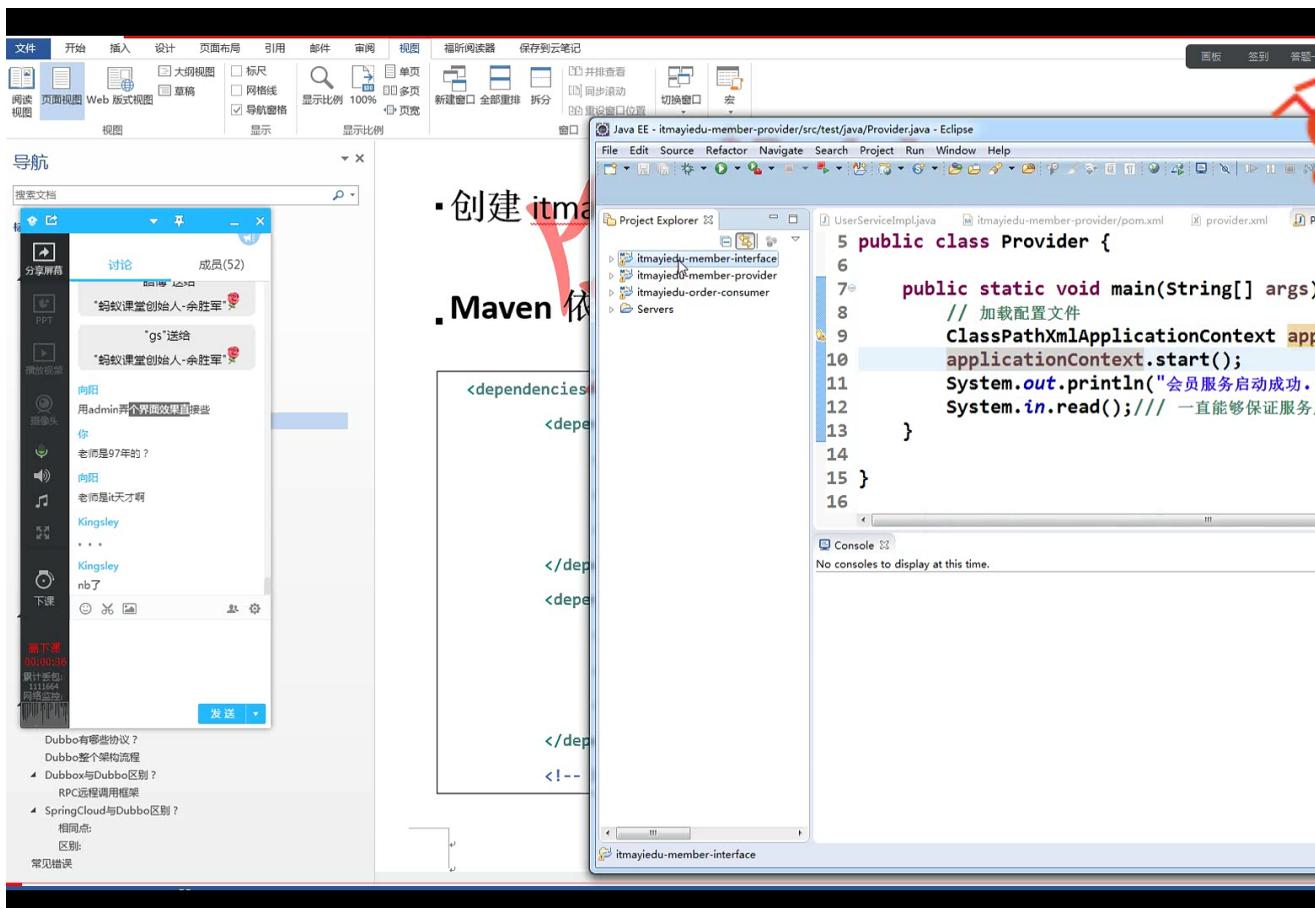
    <!-- 使用multicast广播注册中心暴露发现服务地址 -->
    <dubbo:registry protocol="zookeeper" address="zookeeper://127.0.0.1:2181" />

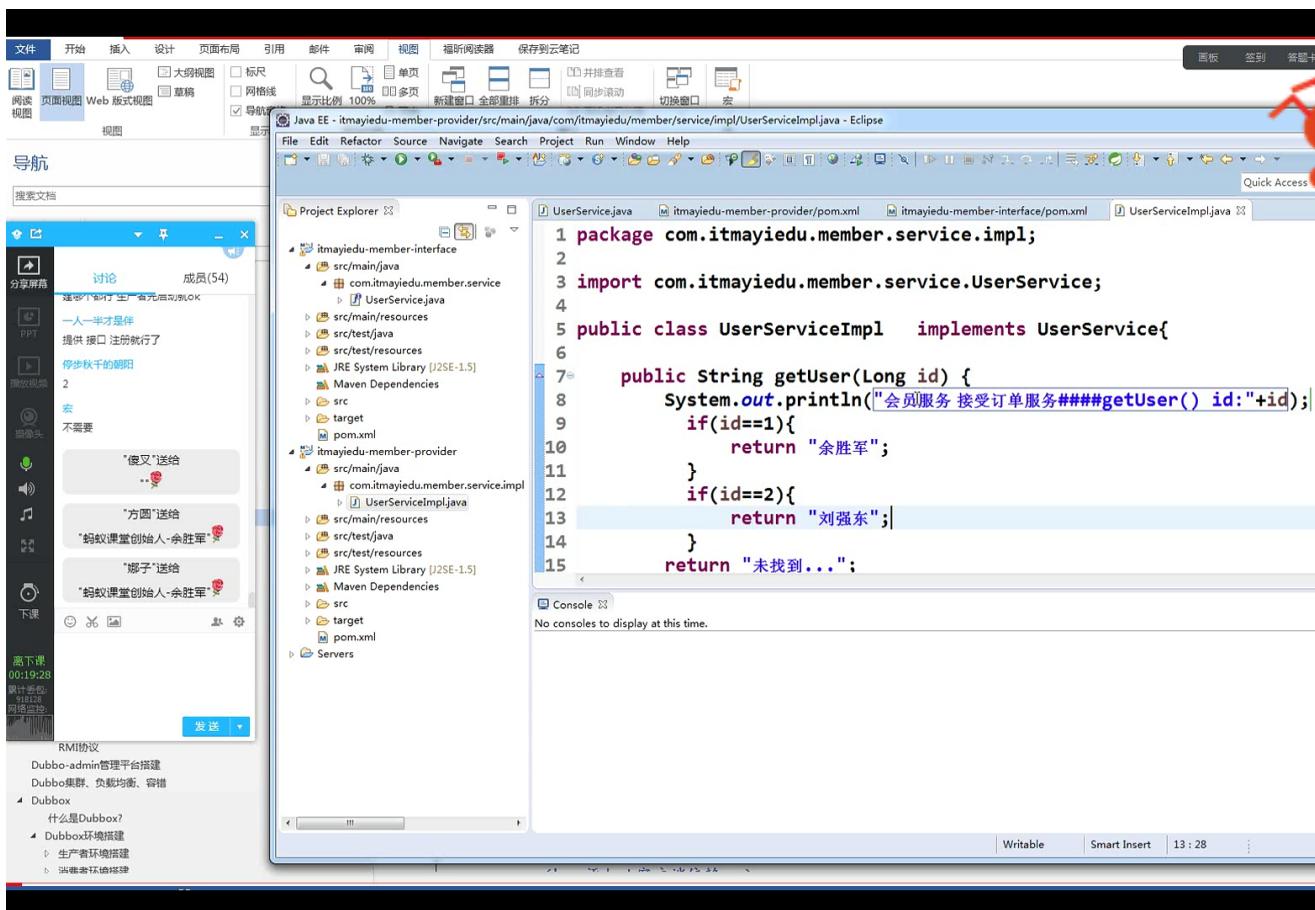
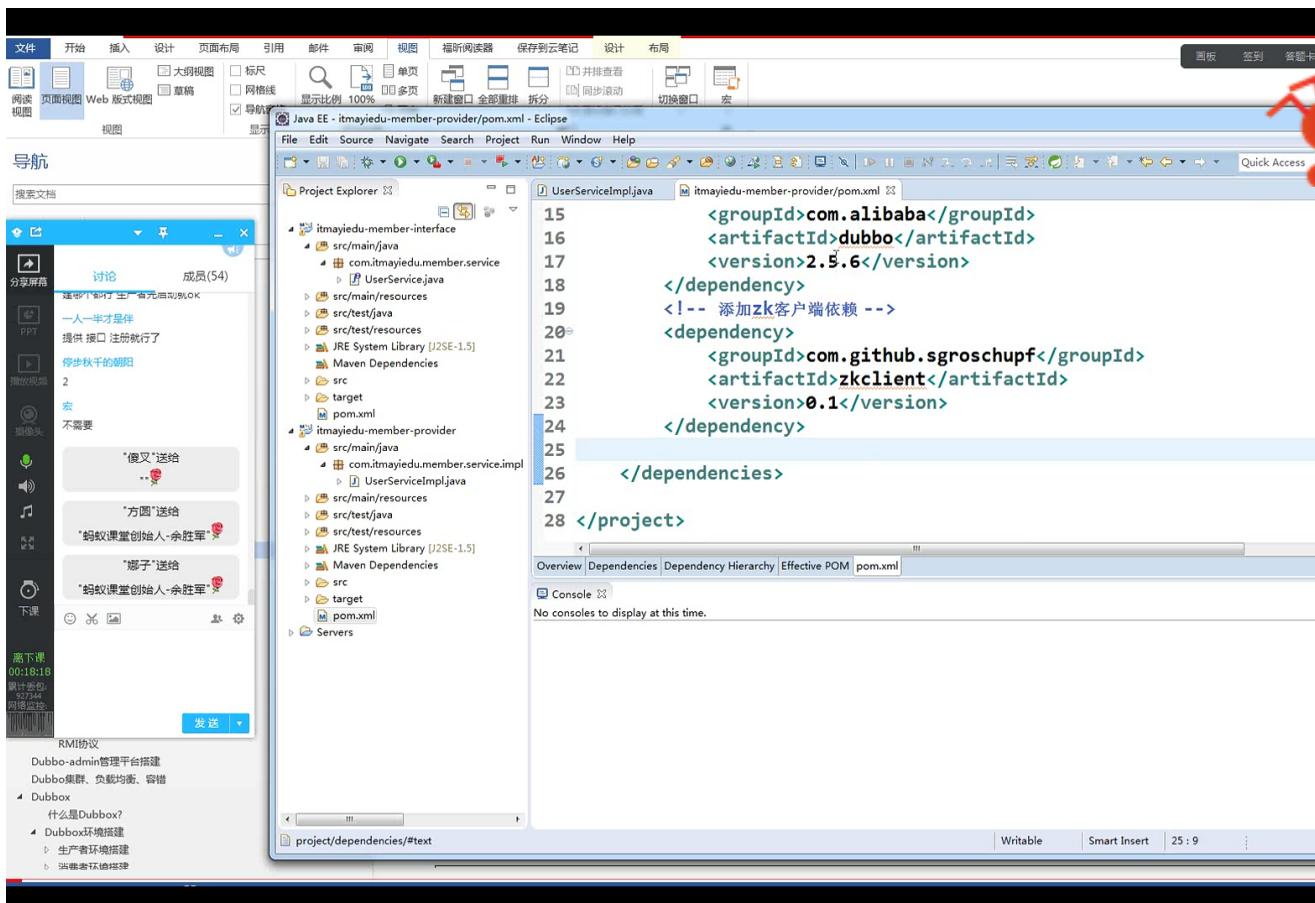
    <!-- 生成远程服务代理，可以和本地bean一样使用demoService -->
    <dubbo:reference id="userService" interface="com.itmayiedu.service.UserService" />

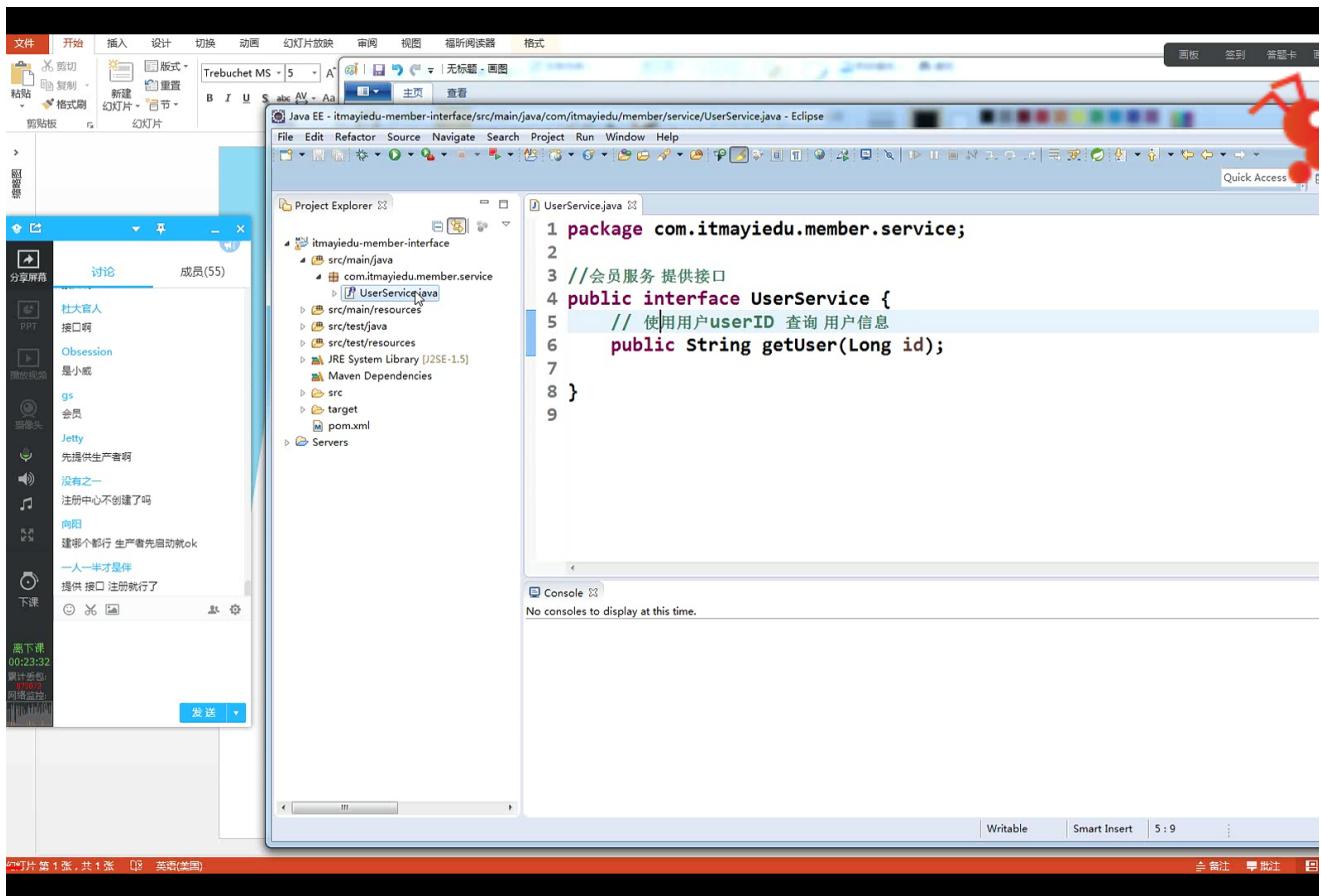
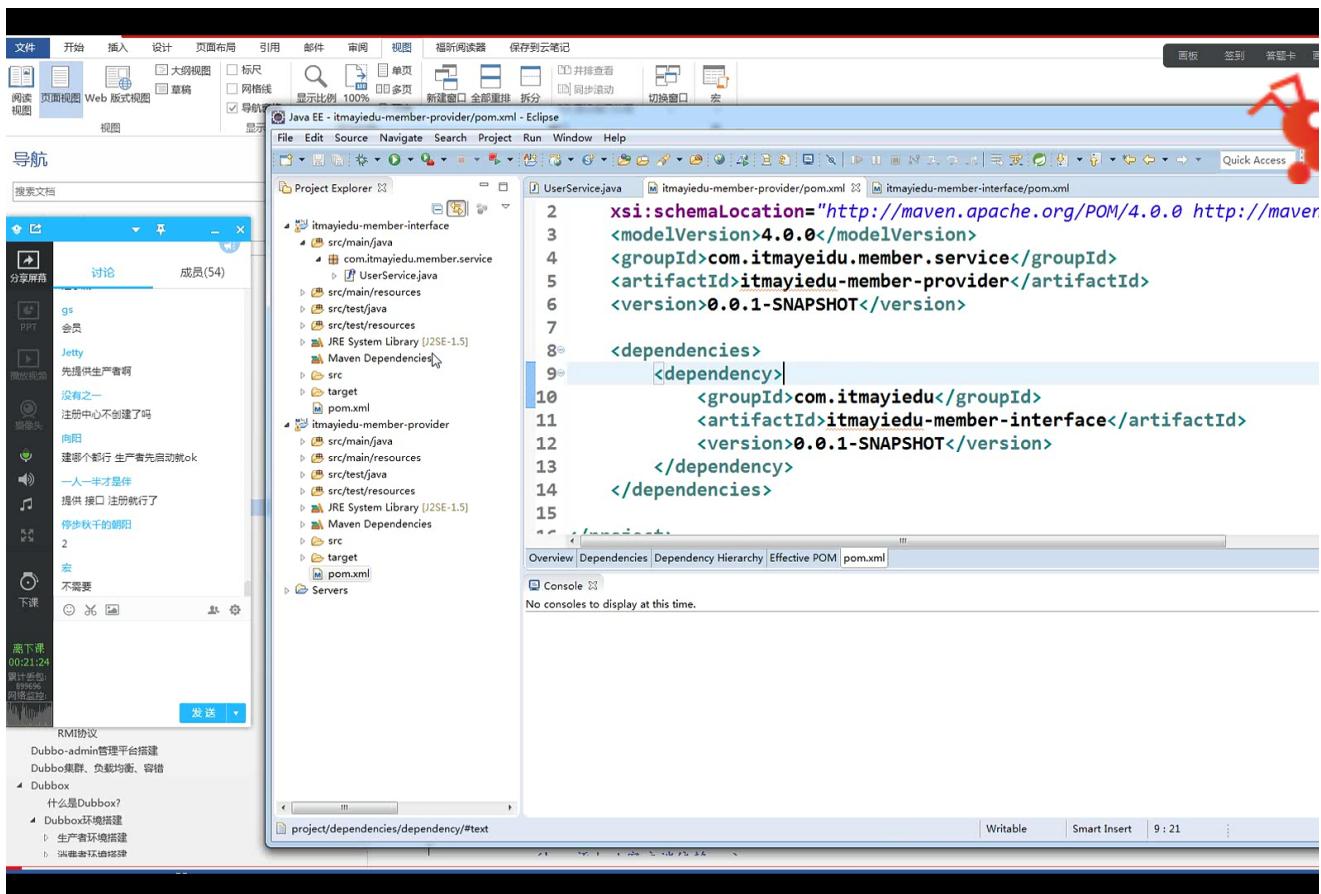
</beans>
```

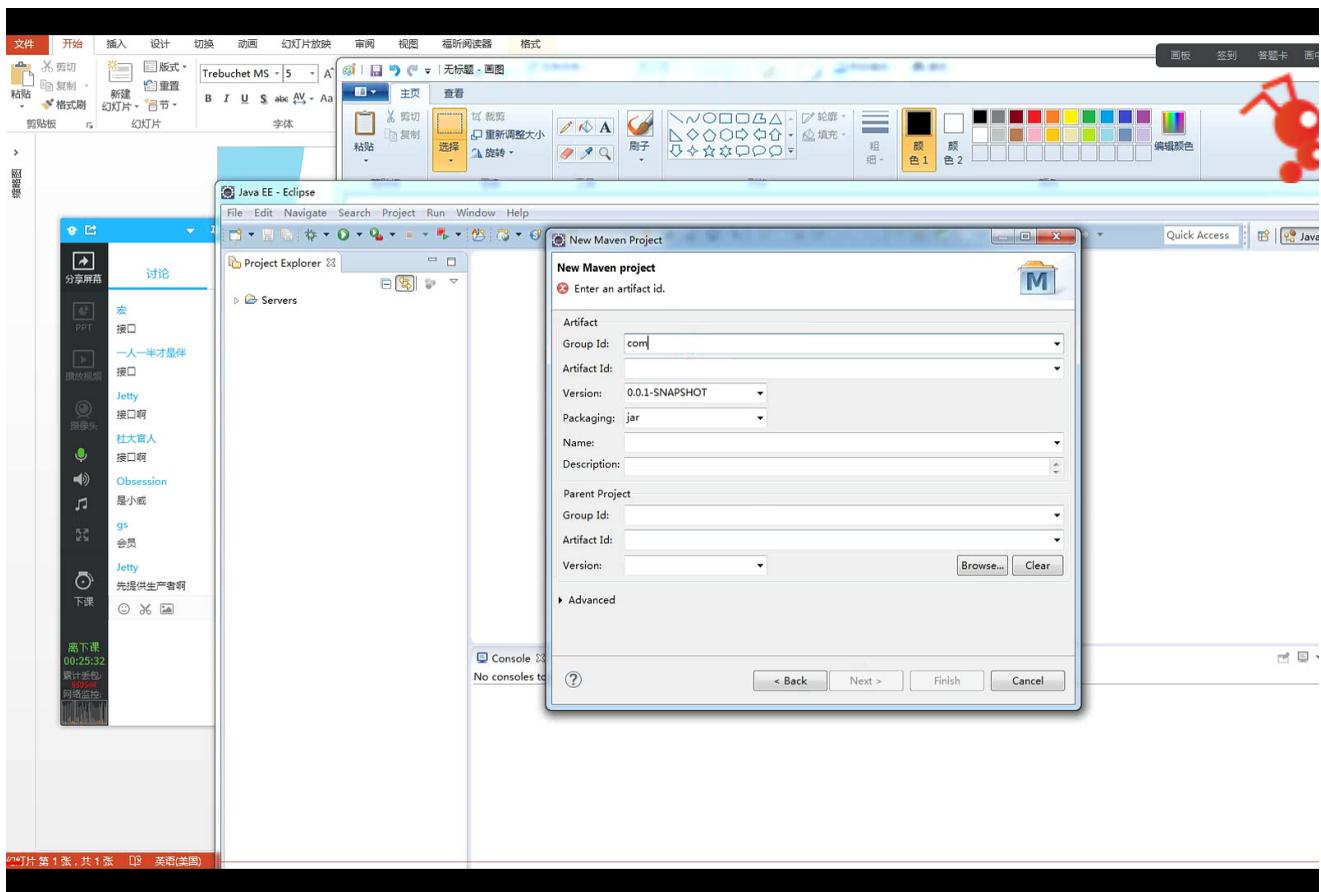
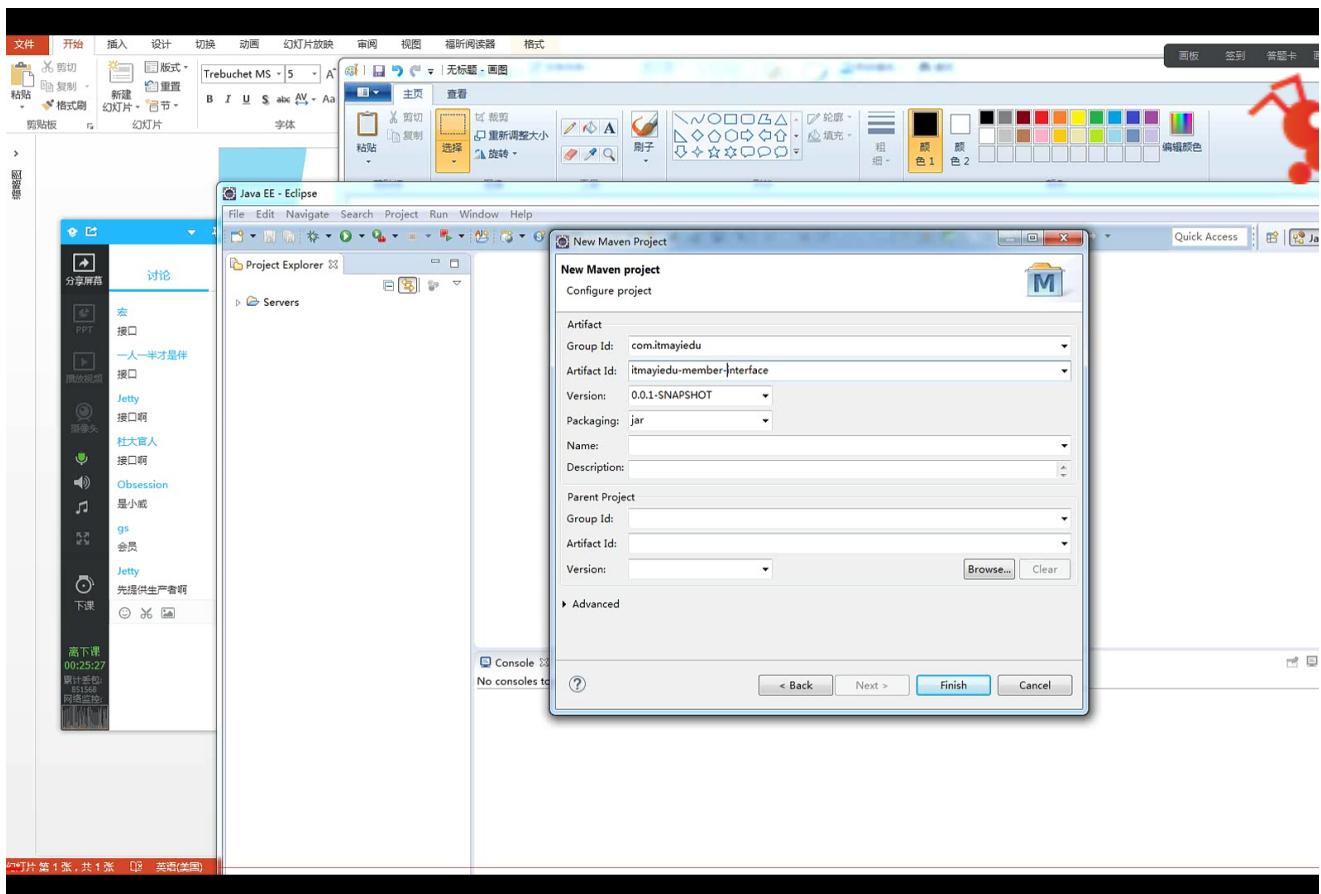
Java EE - itmayiedu-order-consumer/pom.xml - Eclipse

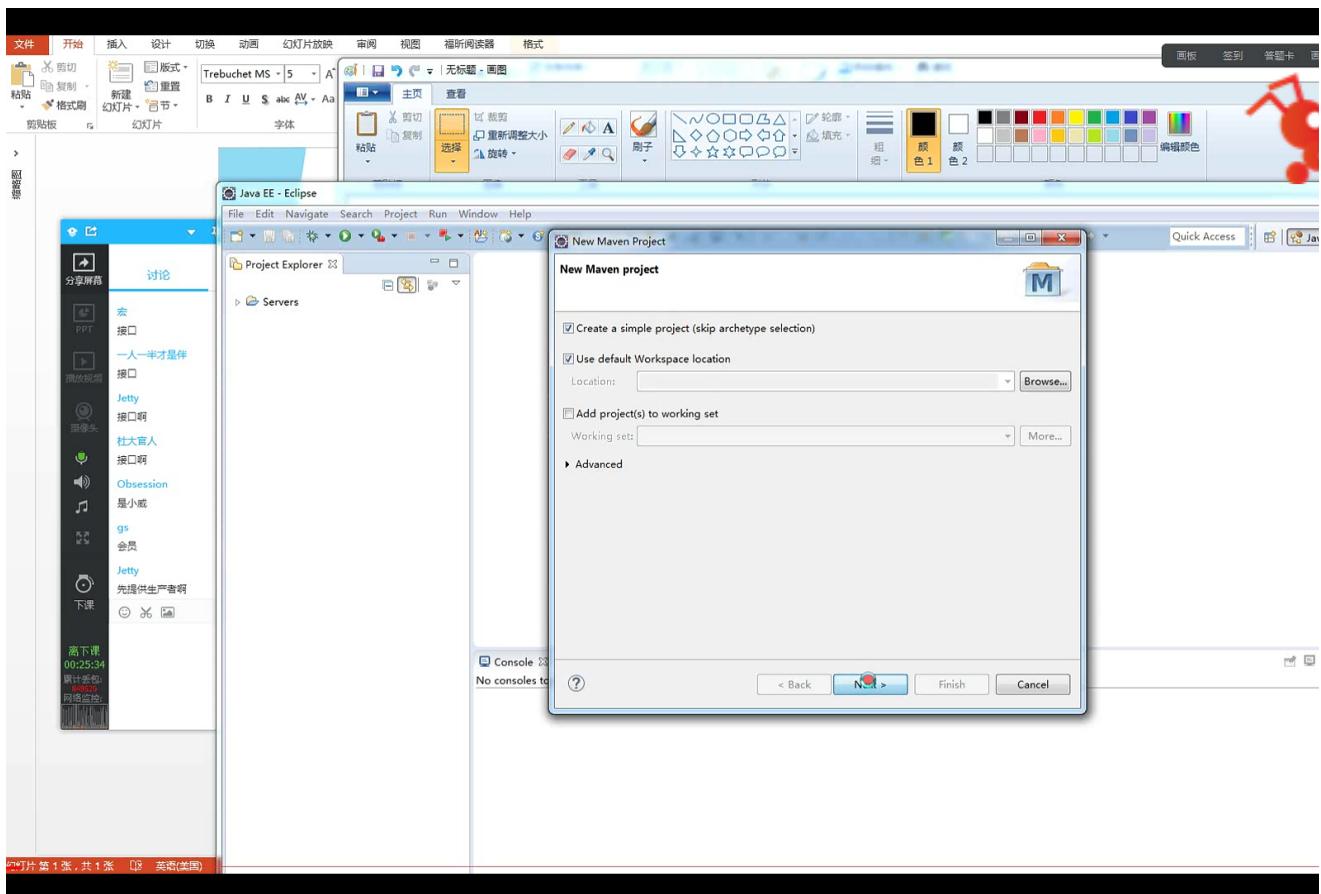
```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.itmayiedu</groupId>
    <artifactId>itmayiedu-order-consumer</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>com.itmayiedu</groupId>
            <artifactId>itmayiedu-member-interface</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
        <dependency>
            <groupId>com.itmayiedu</groupId>
            <artifactId>itmayiedu-member-provider</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>
</project>
```











Dubbo 环境搭建

## 启动 zookeeper

使用 zookeeper 作为服务注册中心。

## 创建 itmayiedu-interface 工程提供会员接口。

### 创建 itmayiedu-interface 工程提供服务接口。

```
//会员服务接口
public interface UserService {
    //使用userid查询 用户信息
    public String getUser(Long userId);
}
```

会员服务提供，查询用户接口信息，暴露给订单系统，调用会员服务接口查询用户信息。

**itmayiedu-order-service 消费者**

**itmayiedu-member-interface 生产者**

```

sequenceDiagram
    participant OS as "itmayiedu-order-service 消费者"
    participant MS as "itmayiedu-member-interface 生产者"
    OS->>OS: 订单服务
    activate OS
    OS->>MS: 调用会员服务接口
    deactivate OS
    MS->>MS: 会员服务
    deactivate MS
    
```

使用Dubbo框架实现

Dubbo项目结构 误区

itmayiedu-member-interface 会员提供服务项目,没有具体实现  
itmayiedu-member-service-impl--- 会员服务具体实现  
itmayiedu-order-service----订单服务

SpringCloud 注册中心(consul)、ribbon(负载均衡)、feign、rest(客户端调用工具)、config、zuul、hystrix

Dubbo 能够解决什么问题

- 问题: rpc远程调用，服务URI地址(`http://*`、`rmi`、`tcp`) 负载均衡 (`nginx`或者 `f5`、`LVS`)
- 依赖关系非常复杂
- 每个服务还需要监控

Dubbo

- API透明化，就相当于控制层调用业务逻辑层，  
负载均衡机制可以替代F5、nginx、lvs、容错机制
- 服务注册与发现主要需要提供API地址
- Dubbo 采用Spring机制

文件 开始 插入 设计 页面布局 引用 审阅 视图 福昕阅读器 保存到云笔记

阅读 视图 Web 版式视图 草稿

导航 搜索文档 标题 页面 结果

讨论 成员(57)

重心 消费者  
订阅：Subscribe  
感知速度 监听  
没有之一 监听  
不同岁月任欺  
事件通知  
感知速度  
watch  
一人一半才是伴 monitor

Dubbox  
什么是Dubbox?  
Dubbox环境搭建  
生产者环境搭建  
消费者环境搭建

1.rpc远程调用框架，服务治理、分布式服务框架，Dubbo实现SOA  
面向服务架构，分布式开发需求，远程调用，集群容错机制、负载均衡、动态配置、路由策略、服务治理，长连接+NIO框架  
2.生产者（提供接口）、消费者（调用接口）、注册中心（负载均衡、容错机制、路由、服务治理）、监控中心、容器

SpringCloud cnraka

订阅含义：实时获取到最新发布服务

注册 (生产启动，在zookeeper上创建一个节点,临时节点 节点名称-节点值 (服务器IP地址+端口号))

订阅中心 (zookeeper)

生产者

消费者

注册中心

监控中心

容器

文件 开始 插入 设计 页面布局 引用 邮件 审阅 视图 福昕阅读器 保存到云笔记

阅读 视图 Web 版式视图 草稿 显示比例 100% 新建窗口 全部重排 拆分 重设窗口位置 窗口 宏

导航 搜索文档 标题 页面 结果

Dubbo概述  
Dubbo的背景  
什么是Dubbo  
Dubbo能做什么  
Dubbo架构  
节点角色说明：  
调用关系说明：  
健壮性：  
伸缩性：  
升级性：  
Dubbo服务治理  
Dubbo环境搭建  
启动zookeeper  
创建tmaiedu-interface工程供会员接口  
创建tmaiedu-member-provider工程生产者  
Maven依赖参数  
定义一个Service实现服务接口  
发布Dubbo服务  
启动Dubbo服务  
创建tmaiedu-order-consumer工程消费者  
Maven依赖参数  
消费会员服务接口  
配置文件参数  
Dubbo支持哪些协议？  
Dubbo协议  
Hessian协议  
HTTP协议  
RMI协议  
Dubbo-admin管理平台搭建  
Dubbo集群、负载均衡、容错  
Dubbox  
什么是Dubbox?  
Dubbox环境搭建  
生产者环境搭建  
消费者环境搭建

- 单一应用架构。  
当网站流量很小时，只需一个应用，将所有功能都部署在一起，以减少部署节点和成本。  
此时，用于简化增删改查工作量的数据访问框架(ORM)是关键。
- 垂直应用架构。  
当访问量逐渐增大，单一应用增加机器带来的加速度越来越小，将应用拆成互不相干的几个应用，以提效率。  
此时，用于加速前端页面开发的Web框架(MVC)是关键。
- 分布式服务架构。  
当垂直应用越来越多，应用之间交互不可避免，将核心业务抽取出来，作为独立的服务，逐渐形成稳定服务中心，使前端应用能更快速的响应多变的市场需求。  
此时，用于提高业务复用及整合的分布式服务框架(RPC)是关键。
- 流动计算架构。  
当服务越来越多，容量的评估，小服务资源的浪费等问题逐渐显现，此时需增加一个调度中心基于访问力实时管理集群容量，提高集群利用率。  
此时，用于提高机器利用率的资源调度和治理中心(SOA)是关键。

文件 开始 插入 设计 切换 幻灯片放映 审阅 视图 福昕阅读器

剪切 复制 新建 重置 幻灯片 - 百节 - 剪贴板 幻灯片

讨论 成员(69)

杜大富人 不会  
重心 我这边卡  
行进 解锁 Kingsley  
数看谁能看到吗  
重心 我这边卡的 杜大富人  
1 你微笑时好美  
1  
下课 离下课 01:24:54  
累计签到: 240/310  
网络监控

发送

1. 使用Dubbo框架 RPC远程调用

网站架构演变过程

单点系统 (MVC)  
itmavieds web  
com.controller----控制层  
com.service----业务逻辑层  
com.dao----数据库访问层

分布式开发 SOA (面向) 服务架构, 将项目拆分成多个子的服务 主要 Http+xml报文

单点系统 中几百个java程序猿开发同一个项目  
会产生什么问题: 代码冲突问题 小团队开发  
互联网公司 特征 人比较多, 高并发、高可用, 开发人员比较多  
敏捷开发  
分布式开发 将一个项目拆分成多个子项目, 会员系统、订单系统、支付系统、交易系统, RPC远程调用, 项目互不影响 (WebService) Http+xml格式  
面向服务架构 实际将传统项目业务逻辑层封装服务 (接口), 暴露其他服务进行调用

The diagram shows the transition from a monolithic system (MVC) to a distributed system. On the left, a monolithic system is shown with layers: com.controller (control layer), com.service (business logic layer), and com.dao (database access layer). This is labeled 'Single-point System (MVC)'. On the right, a distributed system is shown using SOA. It has three services: Order Service, Transaction Service, and Member Service (com.service, com.dao). These services interact via RPC. A legend on the right defines the architectures: 单点系统 (Single-point System), 分布式开发 (Distributed Development), SOA (面向服务架构) (Service-oriented Architecture), and 微服务架构 (Microservices Architecture).

文件 开始 插入 设计 切换 幻灯片放映 审阅 视图 福昕阅读器

剪切 复制 新建 重置 幻灯片 - 百节 - 剪贴板 幻灯片

讨论 成员(60)

不同岁月任风歌  
activeMQ rabbitMQ  
Jetty  
Nginx 也是吧  
Jetty 好的  
停步秋天的朝阳  
h5  
Obsession  
H5  
Obsession  
ios  
重心  
android

发送

微服务架构

控制层----展示页面层 来源

分布式开发 分布式事物

暴露IP地址和端口号 SDA  
SpringCloud http协议+json格式 Dubbox

The diagram illustrates the Microservices Architecture (MSA). At the top, it shows the 'Presentation Layer' (control layer) which includes 'PC', 'android 或者 IOS', and 'H5'. Below this, it shows the 'Service Layer' (SDA) which includes '会员服务' (Member Service), '订单服务' (Order Service), and '支付服务' (Payment Service). Each service layer is connected to its corresponding database: '会员数据库' (Member Database), '订单数据库' (Order Database), and '支付数据库' (Payment Database). A legend on the right defines the architecture: 微服务架构 (Microservices Architecture).

