1 引言

远程服务调度指的是在一台计算机发起请求，另一台机器接收到请求后进行相应的处理后将结果返回请求端；在底层层面，远程调用基于计算机通信原理实现，包括传输协议和网络IO。为了应用的方便，业界推出了更加贴合语言的应用级远程传输协议和标准流程。本文实现了基于socket的远程调用方法，将服务提供商部署服务的套接字记录到注册中心，服务消费者在有需求的时候，只需要通过客户端向注册中心的统一接口发起请求，即可调用远程服务。

2 设计架构

服务注册中心

1.Register

服务提供商

客户端

3.Call

2.Call

4.Return

5.Return

基于socket进行服务器、客户端、服务提供商间的通信。

服务注册中心提供服务服务的注册、服务提供商的可用性检查、服务管理、转发客户端请求、作业调度、消息队列等服务。

客户端根据服务提供商提供的api接口发起远程调用请求。

服务提供商提供实际可调用的服务。

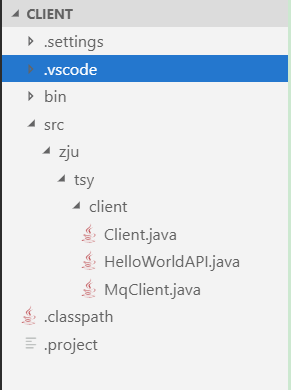
1. 服务提供商将自己的服务注册在服务注册中心，并发布自己的api接口给客户端
2. 客户端向服务注册中心发起调用远程服务请求
3. 服务注册中心检查服务可用性，向实际服务提供商发起请求
4. 服务提供商将调用结果返回给服务注册中心
5. 服务注册中心将结果转发给客户端

3 环境说明

Windows10，Windows8.1

Java1.8

4 客户端实现

****

Client：客户端主程序

MqClient：客户端访问远程消息队列的类，实际作用就是通过MqClient.Produce方法向客户端发起调用请求。

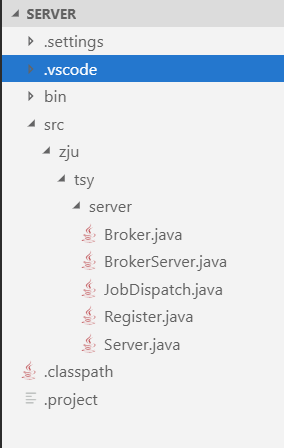
客户端通过字符串形式的请求向服务注册中心发起远程调用，字符串形式如下：

“CALL:ProviderID:Method:Parameters”，其中，ProviderID是服务提供商的唯一标识，Method指服务提供商的方法名，Parameters是该方法可能有的参数（有多个参数时用“:”隔开。

例如，调用HelloWorldAPI的helloworld方法的请求字符串为 "CALL:1:helloWorld"。

Socket连接将请求字符串发送到服务注册中心，之后客户端等待接口返回

5 服务注册中心实现



BrokerServer：服务器处理请求的线程级的类，实现了Runnable接口，主要处理两类请求：来自客户端的“CALL”、来自服务器的“REGISTER”。

JobDispatch：管理一个FIFO的作业队列，请求先暂存至由JobDispatch维护的一个作业队列中，再由作业调度进程进行管理。

Register：实现了服务提供商的注册（add）、检查服务提供商的实时可用性（isavailable）、以及删除所有服务提供商（clear）。

Server：注册中心主程序，开启作业调度进程，并监听9999端口。

6 服务提供商实现



ProviderServer：服务提供商处理请求的线程级的类，实现了Runnable接口，主要处理来自服务中心转发的“CALL”请求。

Provider：服务提供商主程序，先用9999端口与服务注册中心通讯，向服务中心发送注册消息请求实现注册，注册消息形式如下：

“REGISTER:ProviderID”，其中，ProviderID是服务提供商的唯一标识，

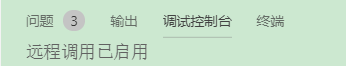
之后作为服务器身份不断监听方才的socket本地端口，响应来自服务中心（这时候是客户）的请求。

作为示例的zju.tsy.helloword包：hellowordAPIImpl是实际的方法实现过程，在ProviderServer中通过反射机制调用这些实际方法。

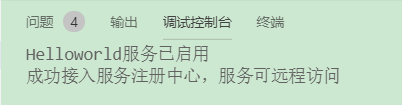
之后所有类似的服务提供商只需在向注册中心注册时提供ProviderID，并向客户端暴露接口和ProviderID，客户端即可使用ProviderID和相应的接口名发起远程调用。

7 Helloworld远程调用示例

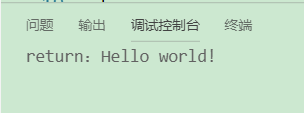
1. 首先开启服务注册中心的Server，Server端显示：



1. 开启helloworld服务提供商的provider，Provider端显示：



1. 运行客户端程序调用helloworld，打印返回结果



同时可以看到provider端输出如下：

