crazyhzm 265

Dubbo源码解析（三十三）远程调用——webservice协议

dubbo java 阅读约 10 分钟

远程调用——webservice协议

> 目标：介绍webservice协议的设计和实现，介绍dubbo-rpc-webservice的源码。

前言

dubbo集成webservice协议，基于 Apache CXF 的 `frontend-simple` 和 `transports-http` 实现 ，CXF 是 Apache 开源的一个 RPC 框架，由 Xfire 和 Celtix 合并而来。关于webservice协议的优势以及介绍可以查看官方文档，我就不多赘述。

源码分析

（一）WebServiceProtocol

该类继承了AbstractProxyProtocol，是webservice协议的关键逻辑实现。

1.属性

```
/**
 * 默认端口
 */
public static final int DEFAULT_PORT = 80;

/**
 * 服务集合
 */
private final Map<String, HttpServer> serverMap = new ConcurrentHashMap<String, HttpServer>();

/**
 * 总线，该总线使用CXF内置的扩展管理器来加载组件（而不是使用Spring总线实现）。虽然加载速度更快，但它不允许像Spring总线那样进行大量配置和定制。
 */
private final ExtensionManagerBus bus = new ExtensionManagerBus();

/**
 * http通信工厂对象
 */
private final HTTPTransportFactory transportFactory = new HTTPTransportFactory();

/**
 * http绑定者
 */
private HttpBinder httpBinder;
```

2.doExport

```
@Override
protected <T> Runnable doExport(T impl, Class<T> type, URL url) throws RpcException {
    // 获得地址
    String addr = getAddr(url);
    // 获得http服务
    HttpServer httpServer = serverMap.get(addr);
    // 如果服务为空，则重新创建服务器。并且加入集合
    if (httpServer == null) {
        httpServer = httpBinder.bind(url, new WebServiceHandler());
        serverMap.put(addr, httpServer);
    }
    // 服务加载器
    final ServerFactoryBean serverFactoryBean = new ServerFactoryBean();
    // 设置地址
    serverFactoryBean.setAddress(url.getAbsolutePath());
    // 设置服务类型
    serverFactoryBean.setServiceClass(type);
    // 设置实现类
    serverFactoryBean.setServiceBean(impl);
    // 设置总线
    serverFactoryBean.setBus(bus);
    // 设置通信工厂
    serverFactoryBean.setDestinationFactory(transportFactory);
    // 创建
    serverFactoryBean.create();
    return new Runnable() {
```

该方法是服务暴露的逻辑实现，基于cxf一些类。

3.doRefer

```
@Override
@SuppressWarnings("unchecked")
protected <T> T doRefer(final Class<T> serviceType, final URL url) throws RpcException {
    // 创建代理工厂
    ClientProxyFactoryBean proxyFactoryBean = new ClientProxyFactoryBean();
    // 设置地址
    proxyFactoryBean.setAddress(url.setProtocol("http").toIdentityString());
    // 设置服务类型
    proxyFactoryBean.setServiceClass(serviceType);
    // 设置总线
    proxyFactoryBean.setBus(bus);
    // 创建
    T ref = (T) proxyFactoryBean.create();
    // 获得代理
    Client proxy = ClientProxy.getClient(ref);
    // 获得HTTPConduit 处理“http”和“https”传输协议。实例由显式设置或配置的策略控制
    HTTPConduit conduit = (HTTPConduit) proxy.getConduit();
    // 用于配置客户端HTTP端口的属性
    HTTPClientPolicy policy = new HTTPClientPolicy();
    // 配置连接超时时间
    policy.setConnectionTimeout(url.getParameter(Constants.CONNECT_TIMEOUT_KEY,
Constants.DEFAULT_CONNECT_TIMEOUT));
    // 配置调用超时时间
    policy.setReceiveTimeout(url.getParameter(Constants.TIMEOUT_KEY, Constants.DEFAULT_TIMEOUT));
    conduit.setClient(policy);
    return ref;
```

该方法是服务引用的逻辑实现。

4.WebServiceHandler

```
private class WebServiceHandler implements HttpHandler {

    private volatile ServletController servletController;

    @Override
    public void handle(HttpServletRequest request, HttpServletResponse response) throws IOException,
ServletException {
        // 如果servletController为空，则重新加载一个
        if (servletController == null) {
            HttpServlet httpServlet = DispatcherServlet.getInstance();
            if (httpServlet == null) {
                response.sendError(500, "No such DispatcherServlet instance.");
                return;
            }
            // 创建servletController
            synchronized (this) {
                if (servletController == null) {
                    servletController = new ServletController(transportFactory.getRegistry(),
httpServlet.getServletConfig(), httpServlet);
                }
            }
        }
        // 设置远程地址
        RpcContext.getContext().setRemoteAddress(request.getRemoteAddr(), request.getRemotePort());
        // 调用方法
        servletController.invoke(request, response);
```

该内部类实现了HttpHandler接口，是WebService协议的请求的处理类。

后记

> 该部分相关的源码解析地址：https://github.com/CrazyHZM/i...

该文章讲解了远程调用中关于webservice协议实现的部分，到这里关于rpc远程调用的部分就结束了，关于远程调用核心的几个内容就是代理、协议，再加上不同功能增强的过滤器等，关键是要把api中关于接口设计方面的内容看清楚，后面各类协议因为很多都是基于第三方的框架去实现，虽然方法逻辑有所区别，但是整体的思路和框架一定顺着api设计的去实现。接下来我将开始对cluster集群模块进行讲解。

crazyhzm

265

关注作者

2 条评论

得票 • 时间

撰写评论 ...

提交评论

zhi新华 ： 博主，有没有使用的demo

• 回复 • 8月26日

crazyhzm： demo倒是没有，不过你可以在这个项目下提个issues，如果有人在这方面有应用，也许能帮到你。
https://github.com/apache/dub...