

# rpc系列2-提供上下文RpcContext



作者 TopGun\_Viper (/u/ae2b91e3398) [+ 关注](#)

2016.09.12 23:47 字数 364 阅读 151 评论 0 喜欢 1

(/u/ae2b91e3398)

实现要求：提供RPC上下文，客户端可以透传数据给服务端。

一个应用的基本设计包含几个重要的角色：

- 实体域
- 会话域
- 服务域

实体域就是应用中抽象出来的组件，如Spring中的Bean，Mybatis中的MappedStatement等。会话域代表着一次交互过程，如Mybatis中的SqlSession，而服务域就是组件的功能集，负责会话域和实体域的生命周期管理，如Spring的ApplicationContext，Shiro的SecurityManager等。

现在构造我们这个rpc demo中的**会话域角色RpcContext**。其主要职责就是负责在一次会话生命周期内，保存、传递相关参数数据。会话中的执行体就是线程了，那么理所应当的RpcContext就应该和当前执行线程绑定。很容易想到了ThreadLocal！

实现如下：

```
/**
 * rpc上下文
 *
 * @author wqx
 *
 */
public class RpcContext {

    private static ThreadLocal<Map<String,Object>> context = new ThreadLocal<Map<String,Ob
    ject>> >(){
        @Override
        protected Map<String,Object> initialValue() {
            Map<String,Object> m = new HashMap<String,Object>();
            return m;
        }
    };

    public static void addAttribute(String key, Object value){
        context.get().put(key,value);
    }

    public static Object getAttribute(String key){
        return context.get().get(key);
    }

    public static Map<String,Object> getAttributes(){
        return Collections.unmodifiableMap(context.get());
    }
}
```

RpcContext中的数据传输载体还是选择RpcRequest，将其包在请求体中即可。如下：

```

public class RpcRequest implements Serializable
{
    /**
     *
     */
    private static final long serialVersionUID = -7102839100899303105L;

    //方法名
    private String methodName;

    //参数类型
    private Class<?>[] parameterTypes;

    //参数列表
    private Object[] args;

    //参数
    private Map<String,Object> context;

```

RpcBuilder中在发送请求前，需要从当前上下文中获取数据。传入RpcRequest对象。如下：

```

RpcRequest request = new RpcRequest(method.getName(), method.getParameterTypes(),args,RpcContext.getAttributes());

```

同理，在服务端接收到请求对象RpcRequest之后，需要将RpcRequest中传来的数据和当前上下文进行关联。Handler的run方法修改如下：

```

    public void run() {
        try{
            ObjectInputStream in = null;
            ObjectOutputStream out = null;
            RpcResponse response = new RpcResponse();
            try {
                in = new ObjectInputStream(socket.getInputStream());
                out = new ObjectOutputStream(socket.getOutputStream());

                Object req = in.readObject();
                if(req instanceof RpcRequest){
                    RpcRequest rpcRequest = (RpcRequest)req;
                    //关联客户端传来的上下文数据
                    RpcContext.context.set(rpcRequest.getContext());
                    Method method = service.getClass().getMethod(rpcRequest.getMethodName(), rpcRequest.getParameterTypes());
                    //。 。 。
                }
            }

```

测试：

```

业务接口增加测试方法：
public interface UserService {

    /**
     * 上下文测试，透明传输数据
     */
    public Map<String,Object> rpcContextTest();

    //。 。 。
}

public class UserServiceImpl implements UserService {

    @Override
    public Map<String,Object> rpcContextTest() {
        Map<String,Object> map = new HashMap<String,Object>();
        map.put("server","hahaha");
        if(RpcContext.getAttributes() != null )
            map.putAll(RpcContext.getAttributes());
        return map;
    }
    //。 。 。
}

```

客户端测试代码：

```
@Test
public void testRpcContext(){
    RpcContext.addAttribute("client", "huhuhu");
    Map<String, Object> res = userService.rpcContextTest();
    Assert.assertEquals(res.get("server"), "hahaha");
    Assert.assertEquals(res.get("client"), "huhuhu");
}
```

完整代码 (https://github.com/TopGunViper/rpc-race/tree/feature\_v1.1)

📖 只是一个简单的rpc demo (/nb/6229705)

举报文章 © 著作权归作者所有



TopGun\_Viper (/u/aee2b91e3398)  
写了 23557 字，被 15 人关注，获得了 29 个喜欢  
(/u/aee2b91e3398)


+ 关注

吾日三省吾code，可以为师矣。。。

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持

♡ 喜欢 (/sign\_in) | 1



更多分享

(http://cwb.assets.jianshu.io/notes/images/5767696



登录 (/sign\_in) 后发表评论

评论

智慧如你，不想发表一点想法 (/sign\_in)咩~