

Dubbo3应用开发第二章

1. Dubbo RPC直连应用开发

1.1. Dubbo RPC直连应用的概念

1.2. RPC直连设计的核心概念

1.3. 网络通信内容的细化

2. 序列化(Serialization)

2.1. 简介

2.2. Dubbo中序列化的实现方式

2.3. Kryo序列化方式使用

1. Dubbo RPC直连应用开发

1.1. Dubbo RPC直连应用的概念

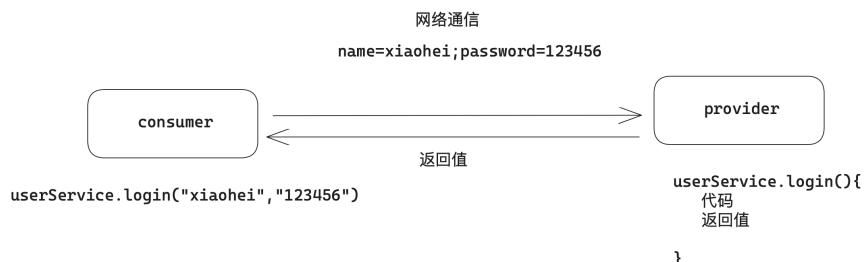
所谓的Dubbo RPC直连应用，指的就是Consumer直接访问Provider，而无需注册中心的接入。Dubbo完成的仅仅是RPC最基本的功能。从这个角度DubboRPC直连等价于SpringCloud体系中的OpenFeign。目前我们学习的Dubbo都是直连访问。

1.2. RPC直连设计的核心概念

1. Provider 服务的提供者
2. Consumer 服务的访问者
3. 网络通信

1.3. 网络通信内容的细化

1. 协议 : consumer与provider在传输数据时双方的约定。
2. 通信方式 : consumer如何与provider进行网络交互。
3. 序列化 : 数据的传输一种格式。



协议

网络传输过程中
传输数据的一种格式约定
调用者 和 被调用者

应用层协议 Http1.x Http2.x

传输层协议 私有协议 dubbo

dubbo:既支持 私有协议 dubbo triple
也支持 共有协议 http http2.x

服务器 (通信方式)

传输层 通信方式 BIO NIO Netty Mina..
Dubbo 内置默认通信 Netty4
应用层 通信方式 Tomcat Resin Jetty

```
<dubbo:protocol name="dubbo" port="-1" transporter="mina"/>
```

序列化

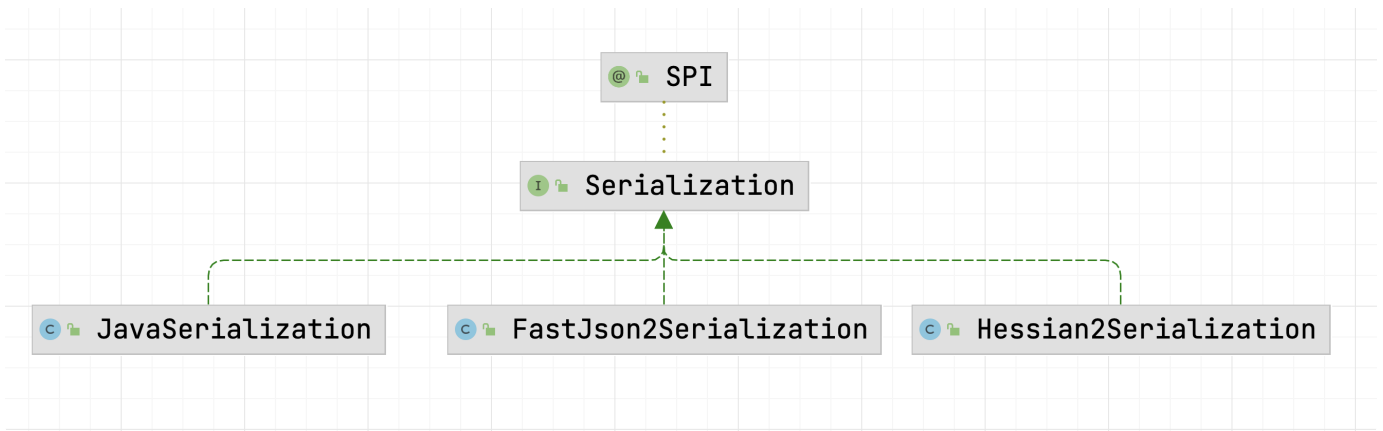
数据传输格式:
好序列化方案: 传输的数据 体积小
不好序列化方案: 传输的数据 体量大

```
<dubbo:protocol name="dubbo" port="-1" transporter="mina" serialization="hessian"/>
```

2. 序列化(Serialization)

2.1. 简介

序列化: 是Dubbo进行RPC中, 非常重要的一个组成部分, 其核心作用就是把网络传输中的数据, 按照特定的格式进行传输。减小数据的体积, 从而提高传输效率。



2.2. Dubbo中序列化的实现方式

1. 常见Dubbo序列化方式

- | | |
|-----------------------|--|
| 1. Hessian | Dubbo协议中默认的序列化实现方案 |
| 2. Java Serialization | JDK的序列化方式 |
| 3. Dubbo序列化 | 阿里尚未开发成熟的高效java序列化实现，阿里不建议在生产环境使用它。 |
| 4. Json序列化 | 目前有两种实现，一种是采用的阿里的fastjson库，另一种是采用dubbo中自己实现的简单json库。 |
| 5. Kryo | Java序列化方式，后续替换Hessian2，是一种非常成熟的序列化实现，已经在Twitter、Groupon、Yahoo以及多个著名开源项目（如Hive、Storm）中广泛的使用。 |
| 6. FST | Java序列化方式，后续替换Hessian2，是一种较新的序列化实现，目前还缺乏足够多的成熟使用案例。 |
| 7. 跨语言序列化方式 | ProtoBuf, Thrift, Avro, MsgPack(MessagePack是一种高效的二进制序列化格式。它允许您在多种语言(如JSON)之间交换数据。但它更快更小。短整型被编码成一个字节) |

2. 常见Dubbo序列化方式序列化效果展示

Serialization Implementation	Request Bytes	Response Bytes
Kryo	272	90
FST	288	96
Dubbo Serialization	430	186
Hessian	546	329
Fastjson	461	218
Json	657	409
Java Serialization	963	630

2.3. Kryo序列化方式使用

1. 引入依赖

XML | 复制代码

```

1 <dependency>
2     <groupId>org.apache.dubbo.extensions</groupId>
3     <artifactId>dubbo-serialization-kryo</artifactId>
4     <version>1.0.1</version>
5 </dependency>

```

2. XML的配置方式

provider XML | 复制代码

```

1 <dubbo:protocol name="dubbo" port="20880" serialization="kryo"/>

```

3. Boot的方式

▼ provider YAML | [复制代码](#)

```
1  dubbo:
2    protocol:
3      name: dubbo
4      port: -1
5      serialization: kryo
```

4. Consumer端调用

▼ XML | [复制代码](#)

```
1  <dubbo:reference id="userService" interface="com.suns.service.UserService"
2    url="dubbo://192.168.50.62:20880/com.suns.service.UserService?serialization=kryo"/>
3
4  后续如果引入注册中心 url就可以不写。
```

▼ Java | [复制代码](#)

```
1  @DubboReference(url = "dubbo://192.168.50.62:20880/com.suns.service.UserService?serialization=kryo")
```