# dubbo分享4--参数验证

参数验证功能是基于JSR303实现的，用户只需标识JSR303标准的验证Annotation，并通过声明filter来实现验证

1、由于没用maven，和对dubbo不是很了解的原因，这次，总因为jar包不对，而导致：dubbo客户端程序，启动不起来

所以决定：将原来用过的所有jar包全部去，将dubbo-demo-provider-2.5.4-SNAPSHOT/lib下的所有jar包全部导入项目中

一切就OK了

2、服务消费者代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | import  java.util.Set;    import  javax.validation.ConstraintViolation;  import  javax.validation.ConstraintViolationException;    import  org.springframework.context.support.ClassPathXmlApplicationContext;    import  com.alibaba.dubbo.demo.ValidationParameter;  import  com.alibaba.dubbo.demo.ValidationService;  import  com.alibaba.dubbo.rpc.RpcException;    public  class  Consumer {         /\*\*        \* @param args        \* @throws Exception        \*/       public  static  void  main(String[] args)  throws  Exception {           ClassPathXmlApplicationContext context =  new  ClassPathXmlApplicationContext(                   new  String[] {  "[classpath:consumer.xml](http://classpathconsumer.xml)"  });           context.start();             // DemoService demoService = (DemoService)           // context.getBean("demoService");           // while (true) {           // String hello = demoService.sayHello("world");           // System.out.println(hello);           //           // Thread.sleep(100);           // }               ValidationService validationService = (ValidationService) context.getBean( "validationService" );             while  ( true ) {               ValidationParameter parameter =  new  ValidationParameter();               parameter.setAge( 2 );               parameter.setEmail( "[han@qq.com](mailto:han@qq.com)" );                 try  {                   String result = validationService.intsert(parameter);                     System.out.println(result);               }  catch  (RpcException e) {  // 抛出的是RpcException                   ConstraintViolationException ve = (ConstraintViolationException) e.getCause();  // 里面嵌了一个ConstraintViolationException                   Set<ConstraintViolation<?>> violations = ve.getConstraintViolations();  // 可以拿到一个验证错误详细信息的集合                   System.out.println(violations);               }           }       }  } |

3、服务消费者配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <? xml  version = "1.0"  encoding = "UTF-8" ?>  < beans  xmlns = "<http://www.springframework.org/schema/beans>"  [xmlns:xsi](http://xmlnsxsi) = "<http://www.w3.org/2001/XMLSchema-instance>"  [xmlns:dubbo](http://xmlnsdubbo) = "<http://code.alibabatech.com/schema/dubbo>"       xsi:schemaLocation="http://www.springframework.org/schema/beans  <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](http://dubboapplication)  name = "consumer-of-helloworld-app"   />         <!-- 使用zookeeper注册中心暴露发现服务地址 -->       < [dubbo:registry](http://dubboregistry)  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](http://dubboreference)  id = "demoService"  interface = "com.alibaba.dubbo.demo.DemoService"            retries = "2"             />         <!-- 生成远程服务代理，可以和本地bean一样使用demoService -->       < [dubbo:reference](http://dubboreference)  id = "validationService"  interface = "com.alibaba.dubbo.demo.ValidationService"            retries = "2"  validation = "true"             />    </ beans > |

**注：重点关注一下，带有下面信息的那一行，就OK了，此处表明，是在客户端侧进行参数验证**

|  |  |
| --- | --- |
| 1 | validation= "true" |

4、服务提供者配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | <? xml  version = "1.0"  encoding = "UTF-8" ?>  < beans  xmlns = "<http://www.springframework.org/schema/beans>"  [xmlns:xsi](http://xmlnsxsi) = "<http://www.w3.org/2001/XMLSchema-instance>"  [xmlns:dubbo](http://xmlnsdubbo) = "<http://code.alibabatech.com/schema/dubbo>"       xsi:schemaLocation="http://www.springframework.org/schema/beans  <http://www.springframework.org/schema/beans/spring-beans.xsd>  <http://code.alibabatech.com/schema/dubbo>  <http://code.alibabatech.com/schema/dubbo/dubbo.xsd>">         <!-- 提供方应用信息，用于计算依赖关系 -->       < [dubbo:application](http://dubboapplication)  name = "hello-world"   />         <!-- 使用zookeeper注册中心暴露发现服务地址 -->       < [dubbo:registry](http://dubboregistry)  address = "<zookeeper://127.0.0.1:2181>"  />         <!-- 用dubbo协议在20880端口暴露服务 -->       < [dubbo:protocol](http://dubboprotocol)  name = "dubbo"  port = "20880"  />         <!-- 声明需要暴露的服务接口 -->       < [dubbo:service](http://dubboservice)  interface = "com.alibaba.dubbo.demo.DemoService"  ref = "demoService"  />         <!-- 和本地bean一样实现服务 -->       < bean  id = "demoService"  class = "com.alibaba.dubbo.demo.provider.DemoServiceImpl"  />         <!-- 声明需要暴露的服务接口 -->       < [dubbo:service](http://dubboservice)  interface = "com.alibaba.dubbo.demo.ValidationService"  ref = "validationService"  />         <!-- 和本地bean一样实现服务 -->       < bean  id = "validationService"  class = "com.alibaba.dubbo.demo.provider.ValidationServiceImpl"  />    </ beans > |

5、服务消费者和提供者共用的类和接口

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | package  com.alibaba.dubbo.demo;    public  interface  ValidationService {       @interface  Intsert {       }         String intsert(ValidationParameter parameter);  } |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | package  com.alibaba.dubbo.demo;    import  java.io.Serializable;    import  javax.validation.constraints.Max;  import  javax.validation.constraints.Min;  import  javax.validation.constraints.Pattern;    public  class  ValidationParameter  implements  Serializable {       private  static  final  long  serialVersionUID = 3469571402386167794L;         @Pattern (regexp =  "^\\s\*\\w+(?:\\.{0,1}[\\w-]+)\*@[a-zA-Z0-9]+(?:[-.][a-zA-Z0-9]+)\*\\.[a-zA-Z]+\\s\*$" )       private  String email;         @Min ( 18 )       // 最小值       @Max ( 100 )       // 最大值       private  int  age;         public  String getEmail() {           return  email;       }         public  void  setEmail(String email) {           this .email = email;       }         public  int  getAge() {           return  age;       }         public  void  setAge( int  age) {           this .age = age;       }    } |

6、服务提供者接口实现类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package  com.alibaba.dubbo.demo.provider;    import  com.alibaba.dubbo.demo.ValidationParameter;  import  com.alibaba.dubbo.demo.ValidationService;    public  class  ValidationServiceImpl  implements  ValidationService {       @Override       public  String intsert(ValidationParameter parameter) {           return  parameter.getEmail() +  "=="  + parameter.getAge();       }  } |

7、运行客户端程序时，控制台会输出以下信息：

|  |  |
| --- | --- |
| 1 | [ConstraintViolationImpl{interpolatedMessage= '最小不能小于18' , propertyPath=age, rootBeanClass= class  com.alibaba.dubbo.demo.ValidationParameter, messageTemplate= '{javax.validation.constraints.Min.message}' }] |

8、OK，此时已经可以验证，参数验证已经开始起作用了。