# dubbo分享7--异步调用

 基于NIO的非阻塞实现并行调用，客户端不需要启动多线程即可完成并行调用多个远程服务，相对多线程开销较小。

1、服务提供端及客户端共享代码

|  |  |
| --- | --- |
| 1  2  3  4  5 | package  com.alibaba.dubbo.demo;    public  interface  DemoService2 {       Person getPersion(String name);  } |

|  |  |
| --- | --- |
| 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 | package  com.alibaba.dubbo.demo;    import  java.io.Serializable;    public  class  Person  implements  Serializable {       private  static  final  long  serialVersionUID = -8994496944734041861L;       private  int  age;       private  String name;         public  int  getAge() {           return  age;       }         public  void  setAge( int  age) {           this .age = age;       }         public  String getName() {           return  name;       }         public  void  setName(String name) {           this .name = name;       }         @Override       public  String toString() {           return  "Person [age="  + age +  ", name="  + name +  "]" ;       }  } |

2、客户端代码

|  |  |
| --- | --- |
| 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 | import  java.util.concurrent.Future;    import  org.springframework.context.support.ClassPathXmlApplicationContext;    import  com.alibaba.dubbo.demo.DemoService2;  import  com.alibaba.dubbo.demo.Person;  import  com.alibaba.dubbo.rpc.RpcContext;    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();             // 异步调用示例           DemoService2 demoService2 = (DemoService2) context.getBean( "demoService2" );             Person p = demoService2.getPersion( "hanshubo" );           System.out.println(p);             Future<Person> pFuture = RpcContext.getContext().getFuture();             p = pFuture.get();           System.out.println(p);       }  } |

3、客户端配置文件

|  |  |
| --- | --- |
| 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 | <?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)       http: //[www.springframework.org/schema/beans/spring-beans.xsd](http://www.springframework.org/schema/beans/spring-beans.xsd)       http: //[code.alibabatech.com/schema/dubbo](http://code.alibabatech.com/schema/dubbo)       http: //[code.alibabatech.com/schema/dubbo/dubbo.xsd](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 -->       <[dubbo:reference](http://dubboreference) id= "demoService"  interface = "com.alibaba.dubbo.demo.DemoService"  retries= "2"  />         <!-- 生成远程服务代理 -->       <[dubbo:reference](http://dubboreference) id= "validationService"  interface = "com.alibaba.dubbo.demo.ValidationService"            retries= "2"  validation= "true"             />         <!-- 生成远程服务代理 -->       <[dubbo:reference](http://dubboreference) id= "demoService2"  interface = "com.alibaba.dubbo.demo.DemoService2"  async= "true"  />    </beans> |

**注：重点关注下面这行代码，就OK啦**

|  |  |
| --- | --- |
| 1 | async= "true" |

 4、服务提供端代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package  com.alibaba.dubbo.demo.provider;    import  com.alibaba.dubbo.demo.DemoService2;  import  com.alibaba.dubbo.demo.Person;    public  class  DemoService2Impl  implements  DemoService2 {       @Override       public  Person getPersion(String name) {           Person r =  new  Person();           r.setAge( 123 );           r.setName(name);           return  r;       }  } |

4、服务提供端配置文件

|  |  |
| --- | --- |
| 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  32  33  34  35  36  37  38  39  40  41 | <?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)       http: //[www.springframework.org/schema/beans/spring-beans.xsd](http://www.springframework.org/schema/beans/spring-beans.xsd)       http: //[code.alibabatech.com/schema/dubbo](http://code.alibabatech.com/schema/dubbo)       http: //[code.alibabatech.com/schema/dubbo/dubbo.xsd](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"  />         <bean id= "cacheService"  class = "com.alibaba.dubbo.demo.provider.CacheServiceImpl"  />         <[dubbo:service](http://dubboservice)  interface = "com.alibaba.dubbo.demo.CacheService"  ref= "cacheService"  />         <!-- 声明需要暴露的服务接口 -->       <[dubbo:service](http://dubboservice)  interface = "com.alibaba.dubbo.demo.DemoService2"  ref= "demoService2"  />         <!-- 和本地bean一样实现服务 -->       <bean id= "demoService2"  class = "com.alibaba.dubbo.demo.provider.DemoService2Impl"  />    </beans> |

5、也可以设置是否等待消息发出：(异步总是不等待返回)

sent="true" 等待消息发出，消息发送失败将抛出异常。

sent="false" 不等待消息发出，将消息放入IO队列，即刻返回。

[?](http://my.oschina.net/hanshubo/blog/378111)

|  |  |
| --- | --- |
| 1 | <[dubbo:method](http://dubbomethod) name= "findFoo"  async= "true"  sent= "true"  /> |

6、如果你只是想异步，完全忽略返回值，可以配置return="false"，以减少Future对象的创建和管理成本：

|  |  |
| --- | --- |
| 1 | <[dubbo:method](http://dubbomethod) name= "findFoo"  async= "true"  return = "false"  /> |