

dubbo之单元测试

kjt-dubbo-test.zip

Dubbox之单元测试

说明书
版本 0.1
项目名称：

修订历史

版本号	作者	修订章节	修订原因	修订日期
0.1	甘建新	初稿	初稿	2016-11-15

目 录

Dubbox之单元测试

1 概述

1.1 术语

1.2 需求背景

1.3 目标

1.3.1 系统目标

2 代码示例

2.1 提供者

3 参考资料

概述

术语

名称	说明

需求背景

- 当服务越来越多时，服务URL配置管理变得非常困难，Array/F5硬件负载均衡器的单点压力也越来越大。
- 当进一步发展，服务间依赖关系变得错综复杂，不能直观分辨应用启动顺序，不能完整的描述应用间的架构关系。
- 随着服务的调用量越来越大，服务的容量问题就会暴露出来，如何分配系统资源变的非常迫切。

目标

系统目标

针对dubbox使用过程中单元测试示例

代码示例

代码架构图

具体代码会以附件方式加入。


```
IService.java ConsumerServiceImpl.java
> kjt-dubbo-test > src/main/java > com.haier.kjt.dubbo.consumer > ConsumerServiceImpl
47  * @Title: print
48  * @Description: TODO void
49  * @throws
50  */
51 public void print() {
52     logger.info("start ...");
53     try {
54         String printWord = dubboService.printWord(new CompanyInfo("haier
55         logger.info("consumer-print:{} ", printWord);
56     } catch (Exception e) {
57         // TODO: handle exception
58         logger.error("{} ", e);
59     }
60     logger.info("end ...");
61 }
62
63 }
64
```

域对象

```
IService.java ConsumerServiceImpl.java CompanyInfo.java
> kjt-dubbo-test > src/main/java > com.haier.kjt.dubbo.domain > CompanyInfo
17  *
18  */
19  *
20 public class CompanyInfo implements Serializable{
21
22     /**
23     * @Fields serialVersionUID : TODO
24     */
25     private static final long serialVersionUID = 1L;
26
27     /**
28     * @Fields name : TODO
29     */
30     private String name;
31
32     /**
33     * @Title: CompanyInfo
34     * @Description: TODO
35     */

```

生产者

```
ProviderServiceImpl.java
src/main/java/com.haier.kjt.dubbo.provider/ProviderServiceImpl

30 Logger logger = LoggerFactory.getLogger(ProviderServiceImpl.class);
31
32 /**
33  * @Description: TODO
34  * @param word
35  * @return
36  * @throws InterruptedException
37  * @see com.haier.kjt.dubbo.api.IService.kjt.dubbo.api.DubboService#printWord
38  */
39 public String printWord(CompanyInfo companyInfo) throws InterruptedException {
40     // TODO Auto-generated method stub
41     String name = companyInfo == null ? "" : companyInfo.getName();
42     String formatString = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(new Date());
43     StringBuffer buffer = new StringBuffer(formatString).append(",").append(name);
44     logger.info("provider-print:{}", buffer);
45     return buffer.toString();
46 }
47
```

XML加载式测试类

```
XmlContextTest.java
src/test/java/com.haier.kjt.dubbo.test/XmlContextTest
testXml0: void

25 public void testXml() throws Exception {
26     // 加载生产者配置
27     ClassPathXmlApplicationContext providerContext = new ClassPathXmlApplicationContext("classpath:provider.xml");
28     providerContext.start();
29     try {
30         // 加载消费者配置
31         ClassPathXmlApplicationContext consumerContext = new ClassPathXmlApplicationContext("classpath:consumer.xml");
32         consumerContext.start();
33         try {
34             ConsumerServiceImpl annotationAction = (ConsumerServiceImpl) consumerContext.getBean("consumer");
35             // 调用消费者方法
36             annotationAction.print();
37         } finally {
38             // 关闭消费者上下文
39             consumerContext.stop();
40             consumerContext.close();
41         }
42     } finally {
43         // 关闭生产者上下文
44         providerContext.stop();
45         providerContext.close();
46     }
47 }
```

注解式生产者测试类

先启动生产者，后启动消费者。

```
ProviderTest.java
kjt-dubbo-test » src/test/java » com.haier.kjt.dubbo.test » ProviderTest
2 * @Title: BaseTest.java
8 package com.haier.kjt.dubbo.test;
9
10 import java.io.IOException;
18
19 /**
20 * 默认加载生产者配置文件
21 * @ClassName: BaseTest
22 * @Description: TODO
23 * @author ganjianxin
24 * @date 2016年7月11日 下午8:40:01
25 *
26 */
27 @RunWith(SpringJUnit4ClassRunner.class)
28 @ContextConfiguration(locations = { "classpath:generic-provider.xml" })
29 public class ProviderTest {
30
31     Logger logger = LoggerFactory.getLogger(getClass());
32
33     /**
34     * 启动生产者
35     * @Title: startProvider
36     * @Description: TODO
37     * @throws Exception void
38     * @throws
39     */
40     @Test
41     public void startProvider() throws Exception {
42         try {
43             System.in.read();
44         } catch (IOException e) {
45             logger.error("error", e);
46         }
47     }
48 }
49
```

注解式消费者测试类

```
ConsumerTest.java
kjt-dubbo-test » src/test/java » com.haier.kjt.dubbo.test » ConsumerTest
2 * @Title: ConsumerTest.java
8 package com.haier.kjt.dubbo.test;
9
10 import org.junit.Test;
19
20 /**
21 * 加载消费者配置
22 * @ClassName: ConsumerTest
23 * @Description: TODO
24 * @author ganjianxin
25 * @date 2016年7月11日 下午8:40:01
26 *
27 */
28 @RunWith(SpringJUnit4ClassRunner.class)
29 @ContextConfiguration(locations = { "classpath:generic-consumer.xml" })
30 public class ConsumerTest {
31
32     /**
33     * 日志
34     * @Fields logger : TODO
35     */
36     Logger logger = LoggerFactory.getLogger(getClass());
37
```

```

37
38= /**
39 * 消费者实现类
40 * @Fields service : TODO
41 */
42= @Autowired
43 ConsumerServiceImpl service;
44
45= /**
46 * 调用
47 * @Title: startConsumer
48 * @Description: TODO
49 * @throws Exception void
50 * @throws
51 */
52= @Test
53 public void startConsumer() throws Exception {
54     service.print();
55 }
56 }
57

```

生产者配置文件

```

17<?xml version="1.0" encoding="UTF-8"?>
18<beans xmlns="http://www.springframework.org/schema/beans"
19       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
20       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans.xsd
21       http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
22
23     <!-- 提供方应用信息，用于计算依赖关系 -->
24     <dubbo:application name="kjt-dubbo-test" owner="ganjianxin" organization="kjt" />
25     <!-- 使用zookeeper注册中心暴露服务地址 -->
26     <dubbo:registry address="zookeeper://zk.gancc.com:2181" />
27     <!-- 用dubbo协议在20880端口暴露服务 -->
28     <dubbo:protocol name="dubbo" port="20880" threads="1000" />
29     <!-- 监控 -->
30     <dubbo:monitor protocol="registry" />
31     <!-- 失败不重试 -->
32     <dubbo:provider timeout="1000" retries="0"/></dubbo:provider>
33
34     <!-- 声明需要暴露的服务接口 -->
35     <dubbo:service interface="com.haier.kjt.dubbo.api.IService" ref="provider" />
36
37     <!-- 和本地bean一样实现服务 -->
38     <bean id="provider" class="com.haier.kjt.dubbo.provider.ProviderServiceImpl" />
39

```

消费者配置文件

```

1<?xml version="1.0" encoding="UTF-8"?>
2<!-- Copyright 1999-2011 Alibaba Group. - - Licensed under the Apache License, Version 2.0 (the "License");
3- you may not use this file except in compliance with the License. - You may obtain a copy of the License at
4- http://www.apache.org/licenses/LICENSE-2.0 - - Unless required by applicable law or agreed to in writing, software
5- distributed under the License is distributed on an "AS IS" BASIS, - WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
6- either express or implied. - See the License for the specific language governing permissions and - limitations under the License. -->
7<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8       xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"
9       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans.xsd
10       http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd">
11
12     <dubbo:application name="kjt-dubbo-test" owner="ganjianxin" organization="kjt" />
13     <!-- 使用zookeeper注册中心暴露服务地址 -->
14     <dubbo:registry address="zookeeper://zk.gancc.com:2181" />
15     <!-- 失败不重试 -->
16     <dubbo:consumer timeout="1000" retries="0"/></dubbo:consumer>
17     <dubbo:reference id="dubboService" interface="com.haier.kjt.dubbo.api.IService" />
18
19     <bean id="consumerService" class="com.haier.kjt.dubbo.consumer.ConsumerServiceImpl">
20         <property name="dubboService" ref="dubboService"/></property>
21     </bean>
22 </beans>
23

```

消费者直连生产者

在很多时候需要本地进行测试，直连生产者只需要改动消费者的配置即可。


在开发及测试环境下，经常需要绕过注册中心，只测试指定服务提供者，这时候可能需要点对点直连；点对点直联方式，将以服务接口为单位，忽略注册中心的提供者列表；A接口配置点对点，不影响B接口从注册中心获取列表。



(1) 如果是线上需求需要点对点，可在<dubbo:reference>中配置url指向提供者，将绕过注册中心，多个地址用分号隔开，配置如下：(1.0.6及以上版本支持)

```
<dubbo:reference id="xxxService" interface="com.alibaba.xxx.XxxService"
url="dubbo://localhost:20890" />
```


(2) 在JVM启动参数中加入-D参数映射服务地址，如：(key为服务名，value为服务提供者url，此配置优先级最高，1.0.15及以上版本支持)

java -Dcom.alibaba.xxx.XxxService=dubbo://localhost:20890	
	注意为了避免复杂化线上环境，不要在线上使用这个功能，只应在测试阶段使用。

(3) 如果服务比较多，也可以用文件映射，如：(用-Ddubbo.resolve.file指定映射文件路径，此配置优先级高于<dubbo:reference>中的配置，1.0.15及以上版本支持)(2.0以上版本自动加载\${user.home}/dubbo-resolve.properties文件，不需要配置)

```
java -Ddubbo.resolve.file=xxx.properties
```

然后在映射文件xxx.properties中加入：(key为服务名，value为服务提供者url)

com.alibaba.xxx.XxxService=dubbo://localhost:20890	
	注意为了避免复杂化线上环境，不要在线上使用这个功能，只应在测试阶段使用。

参考资料

Dubbo官网: <http://dubbo.io/>
Zookeeper官网: <http://zookeeper.apache.org>
Dubbo Git地址: <https://github.com/alibaba/dubbo>
Dubbox Git地址: <https://github.com/dangdangdotcom/dubbox>
Dubbox SVN源码: <http://192.168.180.67/svn/src/basis/dubbox/branches/dubbo-parent>
注：当前使用的dubbox版本是SVN地址编译而来，与git有部分jar版本和参数区别。