# dubbo分享5--结果缓存

结果缓存，用于加速热门数据的访问速度，Dubbo提供声明式缓存，以减少用户加缓存的工作量。

1、客户端和服务提供端共用接口类

|  |  |
| --- | --- |
| 1  2  3  4  5 | package  com.alibaba.dubbo.demo;    public  interface  CacheService {       String findCache(String id);  } |

2、服务提供端接口实现类

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package  com.alibaba.dubbo.demo.provider;    import  java.util.concurrent.atomic.AtomicInteger;    import  com.alibaba.dubbo.demo.CacheService;    public  class  CacheServiceImpl  implements  CacheService {         private  final  AtomicInteger i =  new  AtomicInteger();         @Override       public  String findCache(String id) {           String result =  "request: "  + id +  ", response: "  + i.getAndIncrement();           System.out.println(result);           return  result;       }    } |

3、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 | <?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"  />    </beans> |

4、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  28  29 | <?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= "cacheService"  interface = "com.alibaba.dubbo.demo.CacheService"  cache= "true"  />    </beans> |

5、客户端主类

|  |  |
| --- | --- |
| 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  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | import  org.springframework.context.support.ClassPathXmlApplicationContext;    import  com.alibaba.dubbo.demo.CacheService;    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);           // }               // 参数校验示例           // ValidationService validationService = (ValidationService)           // context.getBean("validationService");           // while (true) {           // ValidationParameter parameter = new ValidationParameter();           // parameter.setAge(23);           // 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);           // }           // }               CacheService cacheService = (CacheService) context.getBean( "cacheService" );             // 测试缓存生效，多次调用返回同样的结果。(服务器端自增长返回值)           String fix =  null ;           for  ( int  i =  0 ; i <  5 ; i++) {               String result = cacheService.findCache( "0" );               if  (fix ==  null  || fix.equals(result)) {                   System.out.println( "i="  + i +  " OK: "  + result);               }  else  {                   System.err.println( "i="  + i +  " ERROR: "  + result);               }               fix = result;               Thread.sleep( 500 );           }             // LRU的缺省cache.size为1000，执行1001次，应有溢出           for  ( int  n =  0 ; n <  1001 ; n++) {               String pre =  null ;               for  ( int  i =  0 ; i <  10 ; i++) {                   String result = cacheService.findCache(String.valueOf(n));                   if  (pre !=  null  && !pre.equals(result)) {                       System.err.println( "n="  + n +  " ERROR: "  + result);                   }                   pre = result;               }           }             // 测试LRU有移除最开始的一个缓存项           String result = cacheService.findCache( "0" );           if  (fix !=  null  && !fix.equals(result)) {               System.out.println( "OK: "  + result);           }  else  {               System.err.println( "ERROR: "  + result);           }       }  } |

6、客户端控制台，返回值

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | i= 0  OK: request:  0 , response:  0  i= 1  OK: request:  0 , response:  0  i= 2  OK: request:  0 , response:  0  i= 3  OK: request:  0 , response:  0  i= 4  OK: request:  0 , response:  0  OK: request:  0 , response:  1001 |