crazyhzm 265

Dubbo源码解析（三十九）集群——merger

dubbo java 阅读约 20 分钟

集群——merger

> 目标：介绍dubbo中集群的分组聚合，介绍dubbo-cluster下merger包的源码。

前言

按组合并返回结果 ，比如菜单服务，接口一样，但有多种实现，用group区分，现在消费方需从每种group中调用一次返回结果，合并结果返回，这样就可以实现聚合菜单项。这个时候就要用到分组聚合。

源码分析

（一）MergeableCluster

```
public class MergeableCluster implements Cluster {

    public static final String NAME = "mergeable";

    @Override
    public <T> Invoker<T> join(Directory<T> directory) throws RpcException {
        // 创建MergeableClusterInvoker
        return new MergeableClusterInvoker<T>(directory);
    }

}
```

该类实现了Cluster接口，是分组集合的集群实现。

（二）MergeableClusterInvoker

该类是分组聚合的实现类，其中最关机的就是invoke方法。

```
@Override
@SuppressWarnings("rawtypes")
public Result invoke(final Invocation invocation) throws RpcException {
    // 获得invoker集合
    List<Invoker<T>> invokers = directory.list(invocation);

    /**
     * 获得是否merger
     */
    String merger = getUrl().getMethodParameter(invocation.getMethodName(), Constants.MERGER_KEY);
    // 如果没有设置需要聚合，则只调用一个invoker的
    if (ConfigUtils.isEmpty(merger)) { // If a method doesn't have a merger, only invoke one Group
        // 只要有一个可用就返回
        for (final Invoker<T> invoker : invokers) {
            if (invoker.isAvailable()) {
                return invoker.invoke(invocation);
            }
        }
        return invokers.iterator().next().invoke(invocation);
    }

    // 返回类型
    Class<?> returnType;
    try {
        // 获得返回类型
```

前面部分在讲获得调用的结果，后面部分是对结果的合并，合并有两种方式，根据配置不同可用分为基于方法的合并和基于merger的合并。

（三）MergerFactory

Merger 工厂类，获得指定类型的Merger 对象。

```
public class MergerFactory {

    /**
     * Merger 对象缓存
     */
    private static final ConcurrentMap<Class<?>, Merger<?>> mergerCache =
            new ConcurrentHashMap<Class<?>, Merger<?>>();

    /**
     * 获得指定类型的Merger对象
     * @param returnType
     * @param <T>
     * @return
     */
    public static <T> Merger<T> getMerger(Class<T> returnType) {
        Merger result;
        // 如果类型是集合
        if (returnType.isArray()) {
            // 获得类型
            Class type = returnType.getComponentType();
            // 从缓存中获得该类型的Merger对象
            result = mergerCache.get(type);
            // 如果为空，则
            if (result == null) {
                // 初始化所有的 Merger 扩展对象，到 mergerCache 缓存中。
                loadMergers();
```

逻辑比较简单。

（四）ArrayMerger

因为不同的类型有不同的Merger实现，我们可以来看看这个图片：

可以看到有好多好多，我就讲解其中的一种，偷懒一下，其他的麻烦有兴趣的去看看源码了。

```
public class ArrayMerger implements Merger<Object[]> {

    /**
     * 单例
     */
    public static final ArrayMerger INSTANCE = new ArrayMerger();

    @Override
    public Object[] merge(Object[]... others) {
        // 如果长度为0  则直接返回
        if (others.length == 0) {
            return null;
        }
        // 总长
        int totalLen = 0;
        // 遍历所有需要合并的对象
        for (int i = 0; i < others.length; i++) {
            Object item = others[i];
            // 如果为数组
            if (item != null && item.getClass().isArray()) {
                // 累加数组长度
                totalLen += Array.getLength(item);
            } else {
                throw new IllegalArgumentException((i + 1) + "th argument is not an array");
            }
        }
```

是不是很简单，就是循环合并就可以了。

后记

> 该部分相关的源码解析地址：https://github.com/CrazyHZM/i...

该文章讲解了集群中关于分组聚合实现的部分。接下来我将开始对集群模块关于路由部分进行讲解。