开始

入手点分析

ConsumerApplication # main()
String greetings = greetingService.hello();

InvokerInvocationHandler # invoke()
return invoker.invoke(rpcInvocation).recreate();

MigrationInvoker # invoke()
return currentAvailableInvoker.invoke(invocation);

服务降级判断

MockClusterInvoker # invoke()
result = this.invoker.invoke(invocation);

FilterChain一路过滤

AbstractCluster # invoke()
return filterInvoker.invoke(invocation);

FilterChainBuilder # invoke()
asyncResult = filter.invoke(nextNode, invocation);

ConsumerContextFilter # invoke()
return invoker.invoke(invocation);

FilterChainBuilder # invoke()
asyncResult = filter.invoke(nextNode, invocation);

FutureFilter # invoke()
return invoker.invoke(invocation);

FutureFilter # invoke()
return invoker.invoke(invocation);

FilterChainBuilder # invoke()
asyncResult = filter.invoke(nextNode, invocation);

MonitorFilter # invoke()
return invoker.invoke(invocation);

路由选择

AbstractClusterInvoker # invoke()
return doInvoke(invocation, invokers, loadbalance);

集群容错

FailoverClusterInvoker # doInvoke()
Result result = invokeWithContext(invoker, invocation);

同步转异步前的准备工作

AbstractClusterInvoker # invokeWithContext()
result = invoker.invoke(invocation);

ListenerInvokerWrapper # invoke()
return invoker.invoke(invocation);

AbstractInvoker # invoke()
AsyncRpcResult asyncResult = doInvokeAndReturn(invocation);

AbstractInvoker # doInvokeAndReturn()
asyncResult = (AsyncRpcResult) doInvoke(invocation);

同步转异步

DubboInvoker # doInvoke()
CompletableFuture<AppResponse> appResponseFuture =
currentClient.request(...).thenApply(...);

NettyClient发送请求

ReferenceCountExchangeClient # request()
return client.request(request, timeout, executor);

HeaderExchangeClient # request()
return channel.request(request, timeout, executor);

HeaderExchangeChannel # request()
channel.send(req);

AbstractPeer # send()
send(message, url.getParameter(Constants.SENT_KEY, false));

AbstractClient # send()
channel.send(message, sent);

NettyChannel # send()