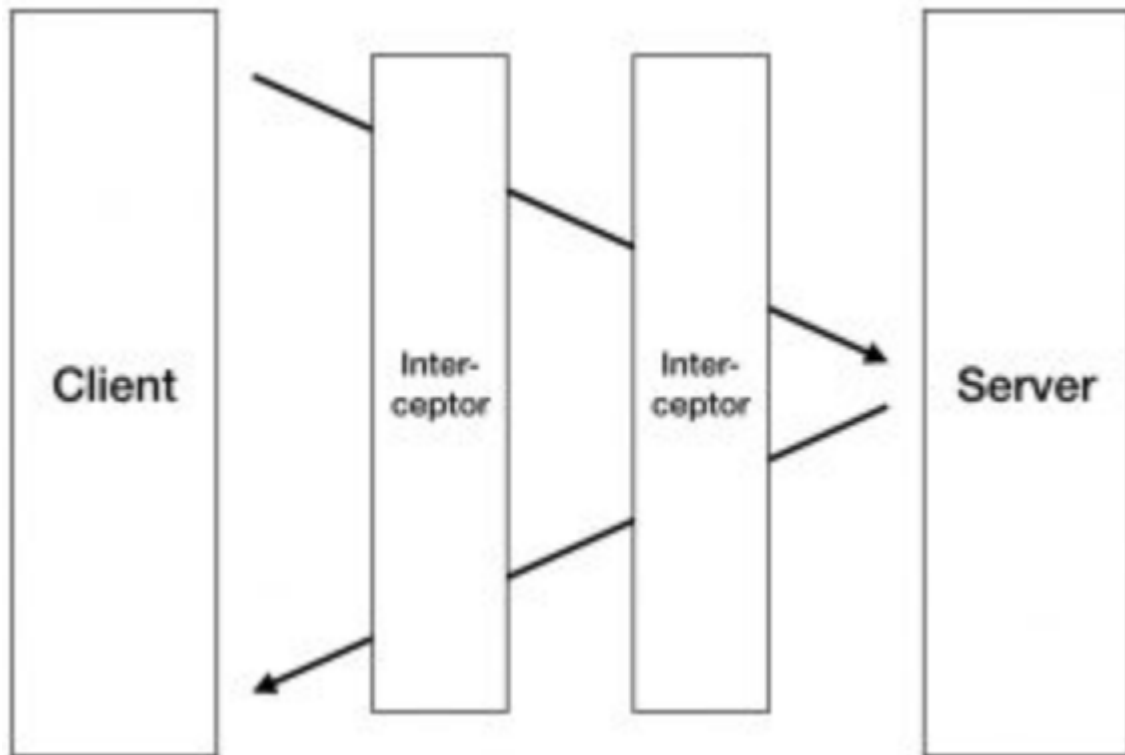


第83期-在gRPC中使用拦截器

拦截器工作原理



创建客户端拦截器

```
public class MyClientInterceptor : Interceptor
{
    public override AsyncUnaryCall<TResponse> AsyncUnaryCall<TRequest,
TResponse>(TRequest request, ClientInterceptorContext<TRequest, TResponse> context,
AsyncUnaryCallContinuation<TRequest, TResponse> continuation)
    {
        if (request is HelloRequest)
        {
            HelloRequest helloRequest = request as HelloRequest;
            helloRequest.Name = helloRequest.Name + DateTime.Now;
        }

        var headers = context.Options.Headers;

        if (headers == null)
        {
            headers = new Metadata();
            var options = context.Options.WithHeaders(headers);
            context = new ClientInterceptorContext<TRequest, TResponse>
(context.Method, context.Host, options);
        }
    }
}
```

```

        headers.Add("caller-user", Environment.UserName);
        headers.Add("caller-machine", Environment.MachineName);
        headers.Add("caller-os", Environment.OSVersion.ToString());

        return base.AsyncUnaryCall(request, context, continuation);
    }
}

```

使用客户端拦截器

直接创建客户端拦截器使用方法

```

using var channel = GrpcChannel.ForAddress("https://localhost:5001");
var intercept = channel.Intercept(new MyClientInterceptor());
var client = new GreeterClient(intercept);
var request = new HelloRequest { Name = "零度" };
var reply = await client.SayHelloAsync(request);

```

在依赖注入容器中使用客户端拦截器

使用工厂委托创建一个拦截器并添加到拦截器集合。

```

services.AddGrpcClient<GreeterClient>(o =>
{
    o.Address = new Uri("https://localhost:5001");
}).AddInterceptor(() => new MyClientInterceptor());

```

也可以先将拦截器注入到容器，然后使用泛型类型添加拦截器。

```

services.AddSingleton(new MyClientInterceptor());
services.AddGrpcClient<GreeterClient>(o =>
{
    o.Address = new Uri("https://localhost:5001");
})
.AddInterceptor<MyClientInterceptor>();

```

创建服务端拦截器

```

public class MyServerInterceptor: Interceptor
{
    public override Task<TResponse> UnaryServerHandler<TRequest, TResponse>
(TRequest request, ServerCallContext context, UnaryServerMethod<TRequest,
TResponse> continuation)
    {
        foreach (var header in context.RequestHeaders)
        {
            Console.WriteLine($"{header.Key}: {header.Value}");
        }
    }
}

```

```
        return base.UnaryServerHandler(request, context, continuation);
    }
}
```

使用服务端拦截器

```
services.AddGrpc(options =>
{
    options.Interceptors.Add<MyServerInterceptor>();
}).AddServiceOptions<GreeterService>(options=>
options.Interceptors.Add<MyServerInterceptor>());
```