

第82期-使用客户端调用gRPC服务

2020年1月14日 9:52

常规 gRPC 客户端配置

通道表示 gRPC 服务的长期连接

```
var channel = GrpcChannel.ForAddress("https://localhost:5001");

var greeterClient = new Greet.GreeterClient(channel);
var counterClient = new Count.CounterClient(channel);

// Use clients to call gRPC services
```

通道和客户端性能情况

- 创建通道是开销高昂的操作，重用通道可带来性能优势。
- 客户端是轻型对象，无需缓存或重复使用。
- 一个通道可以创建多个客户端，每个客户端是线程安全的。

客户端工厂集成

gRPC 与 HttpClientFactory 的集成提供了一种集中管理客户端的方法。

在容器中注册客户端

安装包：Install-Package Grpc.Net.ClientFactory

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

客户端类型被注册为 Transient 多例生命周期，注册完后就可以直接在控制器等其它地方注入客户端并使用。

```
public class TestController: Controller
{
    private readonly Greeter.GreeterClient _client;

    public TestController(Greeter.GreeterClient client)
    {
```

```

        _client = client;
    }

    public void Test()
    {
        var request = new HelloRequest { Name = "xcode.me" };
        var reply = await _client.SayHelloAsync(request);
        Console.WriteLine(reply.Message);
    }
}

```

配置通道和侦听器

```

services.AddGrpcClient<Greeter.GreeterClient>(o =>
{
    o.Address = new Uri("https://localhost:5001");
})
.AddInterceptor(() => new LoggingInterceptor())
.ConfigureChannel(o =>
{
    o.Credentials = new CustomCredentials();
});

```

关于 gRPC 调用的多种方式

最简单的一元调用方式

客户端发送请求消息到服务端 服务完成后，将返回响应消息，支持异步和同步。

```
var response = await client.SayHelloAsync(new HelloRequest { Name = "World" });
```

服务端写流客户端读流方式

客户端发送请求到服务端，服务端源源不断写流，客户端渐进式读取流内容。

客户端写流服务端读流方式

客户端向服务端源源不断写流，服务端使用流式方式渐进式读取流内容。

双向流式通信方式

客户端和服务端同时向对方写流，同时渐进式读取对方发送过来的流内容。