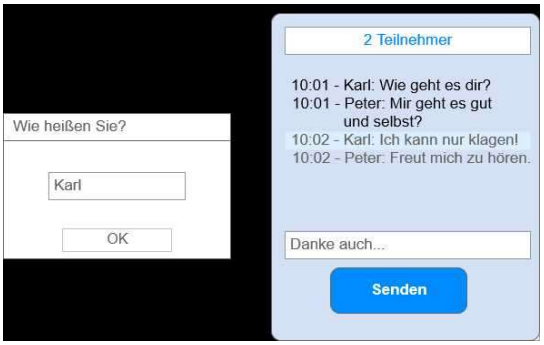
Portierung von WCF zu CoreWCF

Aufgabenstellung:

Gegeben ist eine Chat-Anwendung, die mit *WCF* in .NET Framework 4.8 erstellt wurde. Diese Anwendung liegt in GitHub:

https://github.com/xdah031/Vergleich-WCF-Alternativen/tree/master/WcfChatApplication



Beispiel einer Chat Apps

Gesucht ist die Portierung zu .NET Core mit CoreWCF.

Technologien

- Server: CoreWCF in .NET Core 3.1
- Client: WPF mit MVVM Light

Source Code

Das Projekt liegt in GitHub:

https://github.com/xdah031/Vergleich-WCF-Alternativen/tree/master/CoreWCFChatApplication

Es enthält zwei Verzeichnisse: *WPFClient* für Client und *CoreWCFServer* für Server.

Anleitung:

- Führen Sie **CoreWCFChatApplication.sln** Solution aus und dann starten Sie das gesamte Projekt oder
- Starten Sie den Server und die Clients manuell im Hauptverzeichnis, wenn Sie mehrere Clients parallel simulieren wollten.

Chat Server

Wie wir schon gewusst haben, dass Microsoft keine Portierung von *WCF* zu .NET Core machen wollte. *CoreWCF* ist nur ein Community-Projekt des .NET Foundations. Deshalb gibt es weder keine NuGet-Pakete für *CoreWCF* noch eine offizielle Einleitung davon.

Um *CoreWCF* anzuwenden, muss das *CoreWCF*-Repository zuerst geklont und erstellt werden. Dann nehmen wir die drei .dll Dateien: *CoreWCF.Http*, *CoreWCF.NetTcp* und *CoreWCF.Primitives*.

Danach erstellen wir ein neues .NET Core Konsolen-App, das die drei obigen Dateien referenziert. Paket *System.ServiceModel* wird durch *CoreWCF* ersetzt. *DataContract*, *ServiceContract* und die Methode für *ServiceContract* oder *ServiceBehavior* werden vom alten WCF-Projekt übernommen.

Konfiguration

Zuerst wird ein *WebHost* gebaut. Hier werden Port *8080* für *BasicHttp* und Port *8808* für *NetTCP* benutzt. Die Konfiguration liegt in der *Startup* Klasse.

Code:

```
        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
            .UseKestrel(options => { options.ListenLocalhost(8080); })
            .UseUrls("http://localhost:8080")
            .UseNetTcp(8808)
            .UseStartup<Startup>();
```

Durch den Aufruf von *UseServiceModel* kann man das Service hinzufügen. Der Endpunkt für Service ist per HTTP, NetTCP oder beide möglich.

Code:

```
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddServiceModelServices();
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            app.UseServiceModel(builder =>
            {
                builder.AddService<ChatService>();
                builder.AddServiceEndpoint<ChatService, IChatService>
                    (new BasicHttpBinding(), "/basichttp");
                builder.AddServiceEndpoint<ChatService, IChatService>
                    (new NetTcpBinding(), "/nettcp");
            });
        }
    }
```

Chat Client

Beim Client handelt es sich nicht um die Portierung von *WCF* zu *CoreWCF*, sondern um die Portierung von .NET Framework zu .NET Core. Wir müssen nur die Verbindung mit dem Server verändern. Für One-Way-Service wird *ChannelFactory* verwendet. Aber in der Chat-Anwendung verwenden wir Duplex-Service durch *DuplexChannelFactory*. Duplex-Service ist nur via *NetTCP* möglich.

Code:

```
        static readonly DuplexChannelFactory<IChatService> factory =
            new DuplexChannelFactory<IChatService>
                (new InstanceContext(
                    new ChatCallback()), new NetTcpBinding(),
                    new EndpointAddress("net.tcp://localhost:8808/nettcp"));

        readonly IChatService server = factory.CreateChannel();
```

Weil sich der Dienstverweis nicht automatisch hinzufügen lässt, muss man das *ServiceModel* vom Server auch zum Client-Projekt kopieren. Namespace in Server und Client muss gleich sein, sonst klappt das Deserialisieren bei TCP nicht.

Die Verwendung ist glücklicherweise so ähnlich wie *WCF* in .NET Framework.

Zusammenfassung

CoreWCF bietet eine ähnlichste Verwendung wie *WCF* an. Trotzdem ist dies Projekt noch nicht fertig, sodass die Stabilität und die Effizienz noch nicht verifiziert sind. Im Vergleich zu anderen Alternativen, wie z.B. Grpc, IPC Service Framework oder Web API mit ASP.NET Core, zu denen die Portierung zu aufwändig ist, ist *CoreWCF* die zuversichtlichere Hoffnung von alten WCF-Anwendungen.

Referenz:

[1]: CoreWCF auf GitHub, https://github.com/CoreWCF/CoreWCF

[2]: Tobias Richling, Drei WCF-Alternativen, https://www.dotnetpro.de/A1912WCF