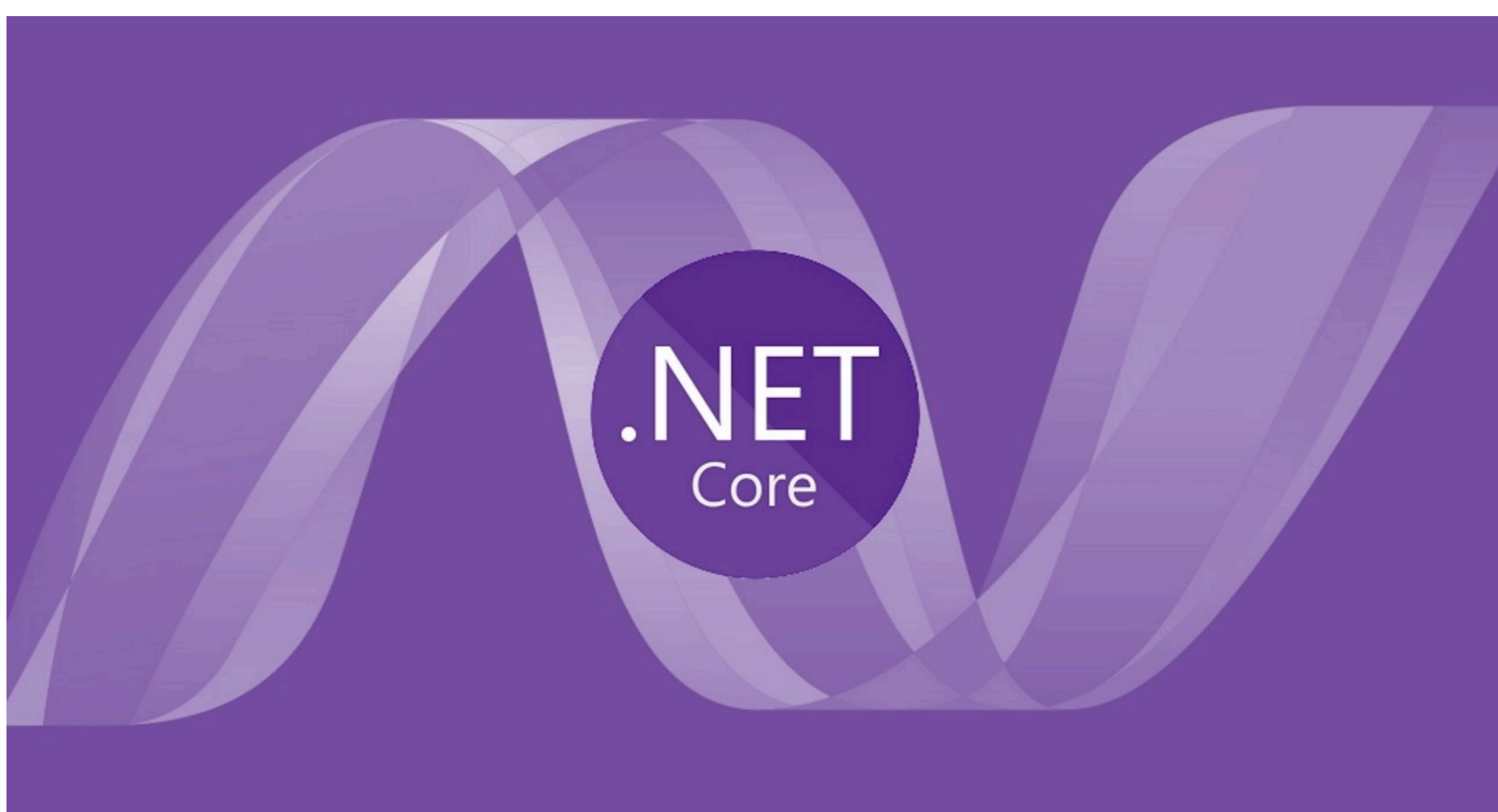


## 通过.NET Core+Vue3 实现SignalR即时通讯功能

作者: 微信公众号:【架构师老卢】

11-5 18:5

1267



**概述:** .NET Core 和 Vue3 结合使用 SignalR 可以实现强大的实时通讯功能, 允许实时双向通信。在这个示例中, 我们将详细说明如何创建一个简单的聊天应用程序, 演示如何使用 .NET Core SignalR 后端和 Vue3 前端来实现实时通讯功能。### 步骤1: 准备工作 确保你已经安装了以下工具和环境: \* [NET Core] (<https://dotnet.microsoft.com/download>) \* [Node.js](<https://nodejs.org/>) \* [Vue CLI](<https://cli.vuejs.org/>) ### 步骤2:

.NET Core 和 Vue3 结合使用 SignalR 可以实现强大的实时通讯功能, 允许实时双向通信。在这个示例中, 我们将详细说明如何创建一个简单的聊天应用程序, 演示如何使用 .NET Core SignalR 后端和 Vue3 前端来实现实时通讯功能。

## 步骤1: 准备工作

确保你已经安装了以下工具和环境:

- .NET Core
- Node.js
- Vue CLI

## 步骤2: 创建 .NET Core SignalR 后端

首先, 让我们创建一个 .NET Core SignalR 后端应用程序。

1. 打开终端并创建一个新的 .NET Core 项目:

```
1 dotnet new web -n SignalRChatApp
2 cd SignalRChatApp
```

1. 在项目中添加 SignalR 包:

```
1 dotnet add package Microsoft.AspNetCore.SignalR
```

1. 打开 Startup.cs 文件, 配置 SignalR 服务:

```
1 // Startup.cs
2
3 using Microsoft.AspNetCore.Builder;
4 using Microsoft.AspNetCore.Hosting;
5 using Microsoft.Extensions.Configuration;
6 using Microsoft.Extensions.DependencyInjection;
7 using Microsoft.Extensions.Hosting;
8
9 namespace SignalRChatApp
10 {
11     public class Startup
12     {
13         public void ConfigureServices(IServiceCollection services)
14         {
15             services.AddSignalR();
16         }
17
18         public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
19         {
20             if (env.IsDevelopment())
21             {
22                 app.UseDeveloperExceptionPage();
23             }
24
25             app.UseRouting();
26             app.UseEndpoints(endpoints =>
27             {
28                 endpoints.MapHub<ChatHub>("/chatHub");
29             });
30         }
31     }
32 }
```

1. 创建一个名为 ChatHub.cs 的 SignalR Hub:

```
1 // ChatHub.cs
2
3 using Microsoft.AspNetCore.SignalR;
4 using System.Threading.Tasks;
5
6 namespace SignalRChatApp
7 {
8     public class ChatHub : Hub
9     {
10         public async Task SendMessage(string user, string message)
11         {
12             await Clients.All.SendAsync("ReceiveMessage", user, message);
13         }
14     }
15 }
```

## 步骤3: 创建 Vue3 前端

现在, 我们将创建一个 Vue3 前端应用程序, 以连接到 SignalR 后端。

1. 在终端中, 创建一个新的 Vue3 项目:

```
1 vue create vue-signalr-chat
```

选择默认配置或根据需要进行配置。

1. 安装 SignalR 客户端库:

```
1 npm install @microsoft/signalr
```

1. 创建一个 Vue 组件来处理聊天:

```
1 <!-- src/components/Chat.vue -->
2
3 <template>
4   <div>
5     <input v-model="user" placeholder="Enter your name" />
6   </div>
7   <div>
8     <input v-model="message" @keyup.enter="sendMessage" placeholder="Type a message" />
9   </div>
10   <div v-for="msg in messages" :key="msg" class="message">{{ msg }}</div>
11 </div>
12 </template>
13
14 <script>
15 export default {
16   data() {
17     return {
18       user: "",
19       message: "",
20       messages: [],
21     };
22   },
23   mounted() {
24     this.connection = new signalR.HubConnectionBuilder()
25       .withUrl("/chatHub")
26       .build();
27
28     this.connection.start().then(() => {
29       this.connection.on("ReceiveMessage", (user, message) => {
30         this.messages.push({user: `${user}`, message});
31       });
32     });
33   },
34   methods: {
35     sendMessage() {
36       if (this.user && this.message) {
37         this.connection.invoke("SendMessage", this.user, this.message);
38         this.message = "";
39       }
40     },
41   },
42 };
43 </script>
44
45 <style scoped>
46 .message {
47   margin: 5px;
48 }
49 </style>
```

1. 在 src/views/Home.vue 中使用 Chat 组件:

```
1 <template>
2   <div class="home">
3     
4   </div>
5 </template>
6
7 <script>
8 import Chat from "@components/Chat.vue";
9
10 export default {
11   name: "Home",
12   components: {
13     Chat,
14   },
15 };
16 </script>
```

## 步骤4: 运行应用程序

1. 启动 .NET Core 后端应用程序:

```
1 dotnet run
```

1. 启动 Vue3 前端应用程序:

```
1 npm run serve
```

现在, 你的 SignalR 实时聊天应用程序应该已经运行了。打开浏览器, 访问 `http://localhost:8080`, 输入用户名, 开始聊天。

这个示例演示了如何使用 .NET Core SignalR 后端和 Vue3 前端创建一个简单的实时聊天应用程序。你可以根据需要扩展该应用程序, 添加更多功能和样式。此外, 你还可以使用 SignalR 来构建更复杂的实时应用程序, 如实时通知、在线游戏和协同编辑等。

