


자습서: ASP.NET Core [SignalR 시작하기

2019. 11. 21. • 읽는 데 17분 걸림 • 

이 문서의 내용

[사전 요구 사항](#)

[웹앱 프로젝트 만들기](#)

[\[SignalR 클라이언트 라이브러리 추가](#)

[\[SignalR 허브 만들기](#)

[\[SignalR 구성](#)

[\[SignalR 클라이언트 코드 추가](#)

[앱 실행](#)

이 자습서에서는 [SignalR을 이용해서 실시간 앱을 구현하기 위한 기본 사항을 알려줍니다. 다음과 같은 작업을 수행하는 방법을 살펴봅니다.

- ✓ 웹 프로젝트를 만듭니다.
- ✓ [SignalR 클라이언트 라이브러리를 추가합니다.
- ✓ [SignalR 허브를 만듭니다.
- ✓ [SignalR을 사용하도록 프로젝트를 구성합니다.
- ✓ 모든 클라이언트에서 연결된 모든 클라이언트로 메시지를 보내는 코드를 추가합니다.

이 모든 과정을 마치면 동작하는 채팅 앱이 만들어집니다.

! [\[SignalR 샘플 앱](#)

사전 요구 사항

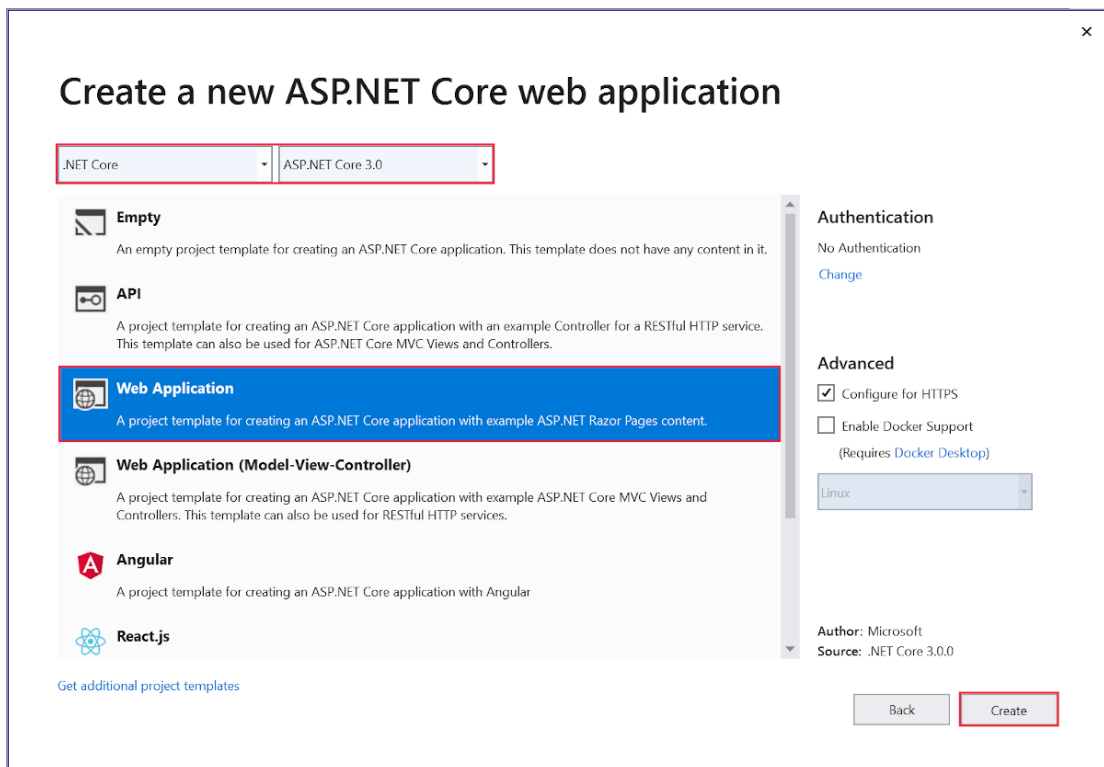
Visual Studio Visual Studio Code Mac용 Visual Studio

- **ASP.NET 및 웹 개발** 워크로드가 있는 [Visual Studio 2019](#)
- [.NET Core 3.0 SDK 이상](#)

웹앱 프로젝트 만들기

Visual Studio Visual Studio Code Mac용 Visual Studio

- 메뉴에서 **파일 > 새 프로젝트를 선택합니다.**
- **새 프로젝트 만들기** 대화 상자에서 **ASP.NET Core 웹 애플리케이션**을 선택한 후, **다음**을 선택합니다.
- **새 프로젝트 구성** 대화 상자에서 *SignalRChat* 프로젝트 이름을 지정한 다음, **만들기**를 선택합니다.
- 새 **ASP.NET Core 웹 애플리케이션 만들기** 대화 상자에서 **.NET Core** 및 **ASP.NET Core 3.0**을 선택합니다.
- [Razor Pages를 사용하는 프로젝트를 생성하려면 **웹 애플리케이션**을 선택한 다음, **만들기**를 선택합니다.



[SignalR 클라이언트 라이브러리 추가

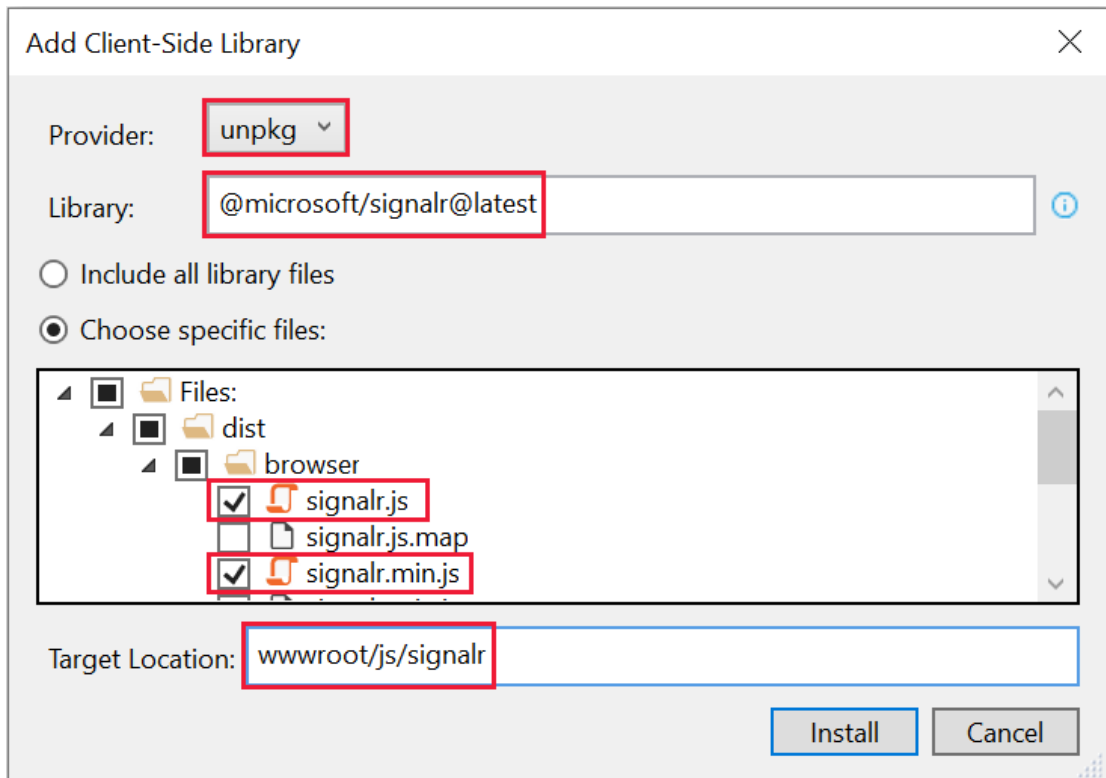
[SignalR 서버 라이브러리는 ASP.NET Core 3.0 공유 프레임워크에 포함되어 있습니다. JavaScript 클라이언트 라이브러리는 프로젝트에 자동으로 포함되지 않습니다. 본 자습서에서는 라이브러리 관리자(LibMan)를 사용하여 *unpkg*에서 클라이언트 라이브러리를 가져옵니다. unpkg는 Node.js의 패키지 관리자인 npm에서 찾은 모든 내용을 전달할 수 있는 CDN(콘텐츠 배달 네트워크)입니다.

Visual Studio

Visual Studio Code

Mac용 Visual Studio

- 솔루션 탐색기에서 프로젝트를 마우스 오른쪽 단추로 클릭하고 **추가 > 클라이언트 쪽 라이브러리**를 선택합니다.
- 클라이언트 쪽 라이브러리 추가 대화 상자에서 **공급자**로 **unpkg**를 선택합니다.
- 라이브러리인 경우 **@microsoft/signalr@latest**를 입력합니다.
- **특정 파일 선택**을 선택하고 **dist/browser** 폴더를 확장한 다음 **signalr.js** 및 **signalr.min.js**를 선택합니다.
- **대상 위치**를 **wwwroot/js/signalr/**로 설정하고 **설치**를 선택합니다.



그러면 LibMan이 **wwwroot/js/signalr** 폴더를 생성한 다음 여기에 선택한 파일을 복사합니다.

[SignalR 허브 만들기

*허브*는 클라이언트-서버 통신을 처리하는 높은 수준의 파이프라인으로 제공되는 클래스입니다.

- SignalRChat 프로젝트 폴더에서 **Hubs** 폴더를 만듭니다.
- **Hubs** 폴더에 다음 코드를 사용하여 **ChatHub.cs** 파일을 만듭니다.

C#

복사

```
using Microsoft.AspNetCore.SignalR;
using System.Threading.Tasks;

namespace SignalRChat.Hubs
{
    public class ChatHub : Hub
    {
        public async Task SendMessage(string user, string message)
        {
            await Clients.All.SendAsync("ReceiveMessage", user,
message);
        }
    }
}
```


ChatHub 클래스는 [SignalR Hub 클래스에서 상속합니다. Hub 클래스는 연결, 그룹 및 메시징을 관리합니다.

연결된 클라이언트에서 SendMessage 메서드를 호출하여 모든 클라이언트에 메시지를 보낼 수 있습니다. 메서드를 호출하는 JavaScript 클라이언트 코드는 자습서 뒷부분에 나와 있습니다. [SignalR 코드는 최대한의 확장성을 제공할 수 있도록 비동기적입니다.

[SignalR 구성

[SignalR에 [SignalR 요청을 전달하도록 [SignalR 서버를 구성해야 합니다.

- 다음 강조 표시된 코드를 *Startup.cs* 파일에 추가합니다.

C#	 복사
<pre>using System; using System.Collections.Generic; using System.Linq; using System.Threading.Tasks; using Microsoft.AspNetCore.Builder; using Microsoft.AspNetCore.Hosting; using Microsoft.AspNetCore.HttpsPolicy; using Microsoft.Extensions.Configuration; using Microsoft.Extensions.DependencyInjection; using Microsoft.Extensions.Hosting; using SignalRChat.Hubs; namespace SignalRChat { public class Startup { public Startup(IConfiguration configuration) { </pre>	

```
Configuration = configuration;
}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to
add services to the container.
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
    services.AddSignalR();
}

// This method gets called by the runtime. Use this method to
configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        // The default HSTS value is 30 days. You may want to
change this for production scenarios, see https://aka.ms/aspnetcore-
hsts.

        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
        endpoints.MapHub<ChatHub>("/chathub");
    });
}
}
```

이러한 변경 사항은 ASP.NET Core 종속성 주입 및 라우팅 시스템에 [SignalR을 추가합니다.

[SignalR 클라이언트 코드 추가

- *Pages\Index.cshtml*의 콘텐츠를 다음 코드로 바꿉니다.

CSHTML	복사
<pre>@page <div class="container"> <div class="row">&nbsp;</div> <div class="row"> <div class="col-2">User</div> <div class="col-4"><input type="text" id="userInput" /> </div> </div> <div class="row"> <div class="col-2">Message</div> <div class="col-4"><input type="text" id="messageInput" /> </div> </div> <div class="row">&nbsp;</div> <div class="row"> <div class="col-6"> <input type="button" id="sendButton" value="Send Message" /> </div> </div> </div> <div class="row"> <div class="col-12"> <hr /> </div> </div> <div class="row"> <div class="col-6"> <ul id="messagesList"> </div> </div> <script src="~/js/signalr/dist/browser/signalr.js"></script> <script src="~/js/chat.js"></script></pre>	

위의 코드는

- 이름 및 메시지 텍스트에 대한 텍스트 상자 및 전송 단추를 만듭니다.
 - [SignalR 허브에서 받은 메시지를 표시하기 위해 `id="messagesList"`를 사용하여 목록을 만듭니다.
 - [SignalR 및 *chat.js* 애플리케이션 코드(다음 단계에서 만듦)에 대한 참조를 포함합니다.
- *wwwroot/js* 폴더에서 다음 코드를 사용하여 *chat.js* 파일을 만듭니다.

JavaScript	복사
<pre>"use strict";</pre>	

```

var connection = new
signalR.HubConnectionBuilder().withUrl("/chatHub").build();

//Disable send button until connection is established
document.getElementById("sendButton").disabled = true;

connection.on("ReceiveMessage", function (user, message) {
    var msg = message.replace(/&/g, "&amp;").replace(/</g,
"&lt;");
    var encodedMsg = user + " says " + msg;
    var li = document.createElement("li");
    li.textContent = encodedMsg;
    document.getElementById("messagesList").appendChild(li);
});

connection.start().then(function () {
    document.getElementById("sendButton").disabled = false;
}).catch(function (err) {
    return console.error(err.toString());
});

document.getElementById("sendButton").addEventListener("click",
function (event) {
    var user = document.getElementById("userInput").value;
    var message = document.getElementById("messageInput").value;
    connection.invoke("SendMessage", user, message).catch(function
(err) {
        return console.error(err.toString());
    });
    event.preventDefault();
});

```

위의 코드는

- 연결을 만들고 시작합니다.
- 허브에 메시지를 전송하는 처리기를 전송 단추에 추가합니다.
- 허브에서 메시지를 수신하고 목록에 추가하는 처리기를 연결 개체에 추가합니다.

앱 실행

Visual Studio Visual Studio Code Mac용 Visual Studio

- **CTRL+F5** 키를 눌러 디버깅 없이 앱을 실행합니다.

- 주소 표시줄에서 URL을 복사하고, 다른 브라우저 인스턴스 또는 탭을 열고, 주소 표시줄에 URL을 붙여넣습니다.

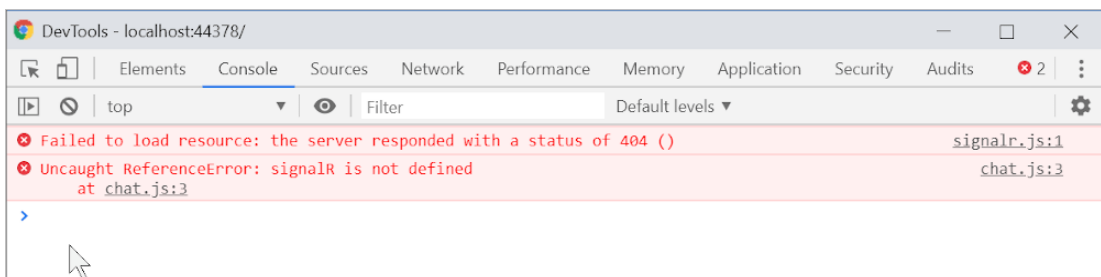
- 브라우저 중 하나를 선택하고, 이름 및 메시지를 입력하고, **보내기 메시지** 단추를 선택합니다.

이름과 메시지는 두 페이지 모두에 즉시 표시됩니다.

! [SignalR 샘플 앱

💡 팁

- 앱이 작동하지 않는 경우 브라우저 개발자 도구(F12)를 열고 콘솔로 이동합니다. HTML 및 JavaScript 코드와 관련된 오류를 볼 수 있습니다. 예를 들어 지정되지 않은 다른 폴더에 *signalr.js*를 넣었다고 가정합니다. 이 경우 해당 파일에 대한 참조는 작동하지 않으며 콘솔에 404 오류가 표시됩니다.



- Chrome에서 ERR_SPDY_INADEQUATE_TRANSPORT_SECURITY 오류가 발생하는 경우, 다음 명령을 실행하여 개발 인증서를 업데이트합니다.

.NET Core CLI

복사

```
dotnet dev-certs https --clean
dotnet dev-certs https --trust
```

이 페이지가 도움이 되었나요?

👍 Yes 🗨 No