

[ 언택트 시대 – 필수 교양 실시간 웹 메시징 개발 기술 ]

# ASP.NET Core 3.1 Signal R 웹 채팅 어플리케이션 개발하기

# Real Time Web Technologies

실시간 웹 (Real Time Web) ?

인터넷에서 사용자들이 하여금 창작자가 정보를 만들어내는 즉시 수신할 수 있도록 하는 기술 혹은 서비스들

- Facebook, Twitter 등 각종 SNS, Web 기반 채팅 솔루션
- Slack, Jandi 각종 실시간 기반 협업 툴
- 실시간 대시보드 및 차트, 웹 푸시 기술
- 웹게임
- WebRTC 등 실시간 화상통화 기술 등



# HTML5 Web socket & Server Side Technologies

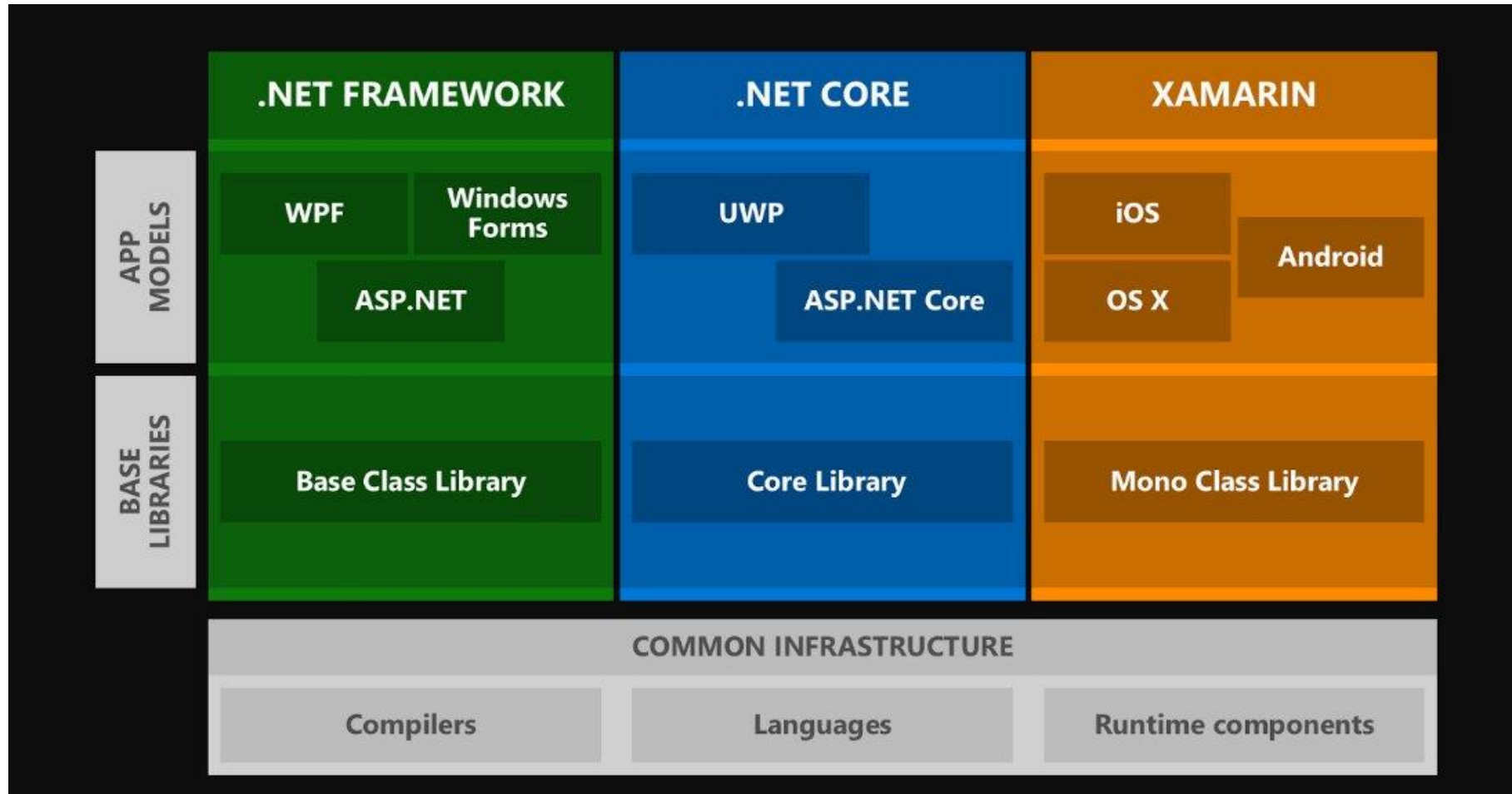
## ● HTML5 WebSocket

- 표준 웹 브라우저에 탑재되어 있는 HTML5 실시간 메시징 API(Application Programming Interface) 기술
- 웹 브라우저에서 웹 브라우저 와 웹서버 간 연결기반 실시간 양방향 메시징 통신기술 제공
- **Web Browser Client Side** 실시간 메시징 기술

## ● Server Side Web Socket 지원 기술들

- 웹 브라우저와 웹서버 간 연결 기반 통신을 위해서는 웹 서버측에도 Web socket 지원 환경 및 기술 필요
- **Node.js Socket.IO**, **ASP.NET SignalR**, **JAVA Spring SockeJS**, **Python websockets**

history of .NET Frameworks = **three** .NET Frameworks



<https://docs.microsoft.com/ko-kr/dotnet/>

# .NET Core vs .NET framework vs .NET5

**.NET**  
Core

## .NET Core 3.1

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

Run Apps ⓘ

[Download .NET Core Runtime](#)

Build Apps ⓘ

[Download .NET Core SDK](#)

Advanced ⓘ

[All .NET Core downloads...](#)

**.NET**  
Framework

## .NET Framework 4.8

.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.

Run Apps ⓘ

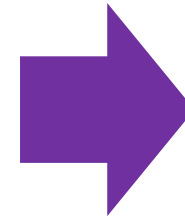
[Download .NET Framework Runtime](#)

Build Apps ⓘ

[Download .NET Framework Dev Pack](#)

Advanced ⓘ

[All .NET Framework downloads...](#)



<https://dotnet.microsoft.com/download>

<https://dotnet.microsoft.com/download/dotnet-core/3.1>

# One .NET Framework is .NET 5

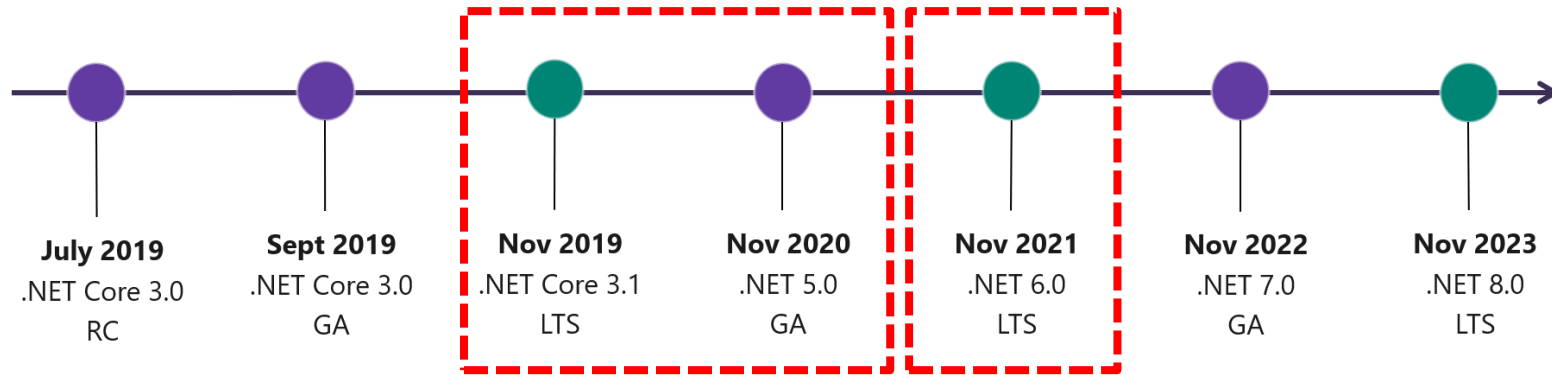
.NET – A unified platform



<https://docs.microsoft.com/en-us/dotnet/core/dotnet-five>

# schedule of .NET Framework



## .NET Schedule



- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

<https://docs.microsoft.com/ko-kr/dotnet/>  
<https://slaner.tistory.com/192>

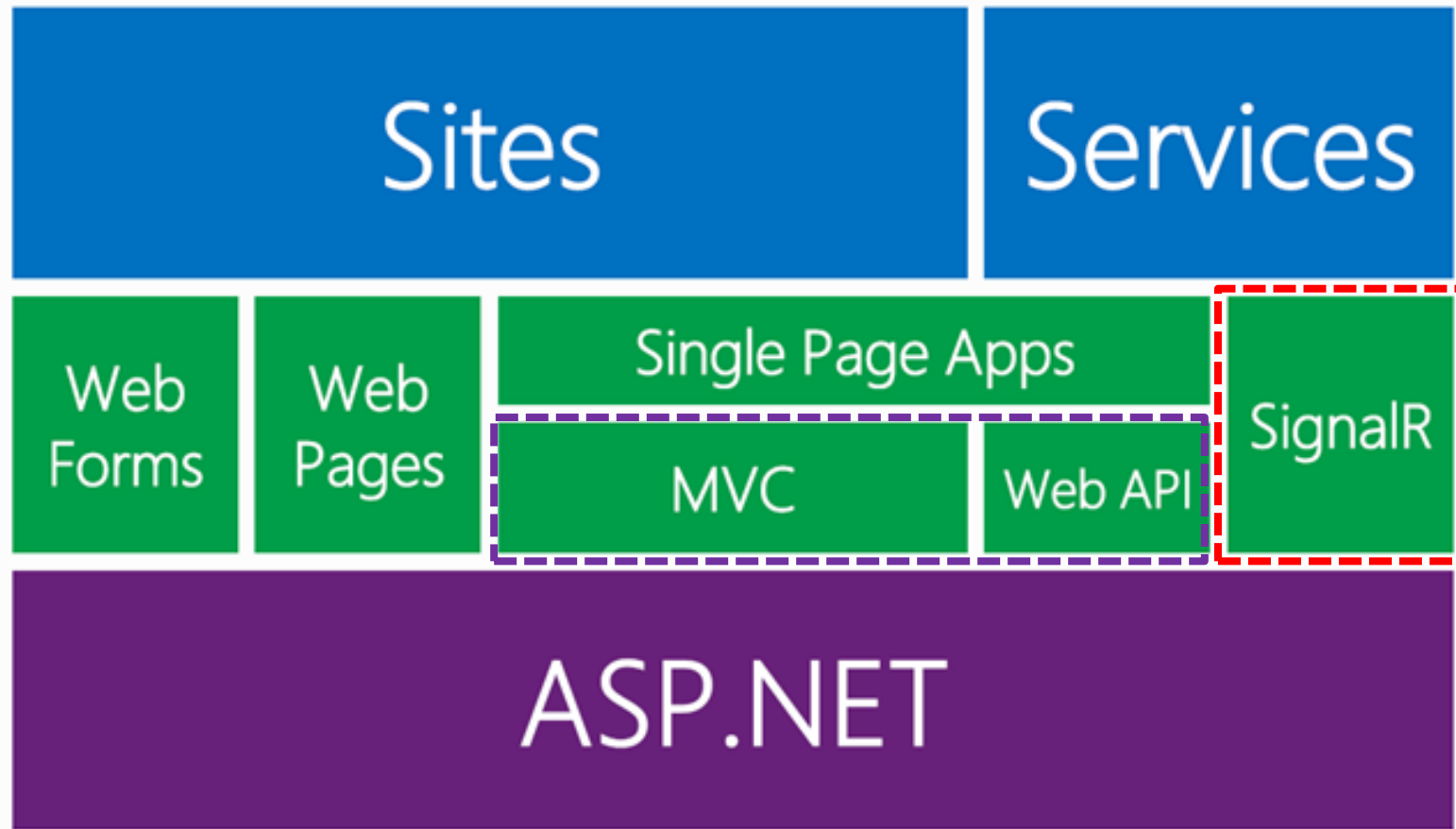
# 하나의 .NET is .NET 5

Release information	Build apps - SDK 	Run apps - Runtime 																																	
<div>v5.0.1</div> <div><a href="#">Release notes</a></div> <div>Released</div> <div>2020-12-08</div>	<div>SDK 5.0.101</div> <div><b>Visual Studio support</b> Visual Studio 2019 (v16.8) Visual Studio 2019 for Mac (v8.8)</div> <div><b>Included in</b> Visual Studio 16.8.3</div> <div><b>Included runtimes</b> .NET Runtime 5.0.1 ASP.NET Core Runtime 5.0.1 .NET Desktop Runtime 5.0.1</div> <div><b>Language support</b> C# 9.0 F# 5.0 Visual Basic 15.9</div> <table><tr><th>OS</th><th>Installers</th><th>Binaries</th></tr><tr><td>Linux</td><td><a href="#">Package manager instructions</a></td><td><a href="#">Arm32</a>   <a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a></td></tr><tr><td>macOS</td><td><a href="#">x64</a></td><td><a href="#">x64</a></td></tr><tr><td>Windows</td><td><a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a></td><td><a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a></td></tr><tr><td>All</td><td><a href="#">dotnet-install scripts</a></td><td></td></tr></table> <div><a href="#">Localized IntelliSense</a></div>	OS	Installers	Binaries	Linux	<a href="#">Package manager instructions</a>	<a href="#">Arm32</a>   <a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>	macOS	<a href="#">x64</a>	<a href="#">x64</a>	Windows	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>	All	<a href="#">dotnet-install scripts</a>		<div>ASP.NET Core Runtime 5.0.1</div> <div>The ASP.NET Core Runtime enables you to run existing web/server applications. <b>On Windows, we recommended installing the Hosting Bundle, which includes the .NET Runtime and IIS support.</b></div> <div><b>IIS runtime support (ASP.NET Core Module v2)</b> 15.0.20336.1</div> <table><tr><th>OS</th><th>Installers</th><th>Binaries</th></tr><tr><td>Linux</td><td><a href="#">Package manager instructions</a></td><td><a href="#">Arm32</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a></td></tr><tr><td>macOS</td><td></td><td><a href="#">x64</a></td></tr><tr><td>Windows</td><td><a href="#">Hosting Bundle</a>   <a href="#">x64</a>   <a href="#">x86</a></td><td><a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a></td></tr></table> <div>.NET Desktop Runtime 5.0.1</div> <div>The .NET Desktop Runtime enables you to run existing Windows desktop applications. <b>This release includes the .NET Runtime, you do not need to install it separately.</b></div> <table><tr><th>OS</th><th>Installers</th><th>Binaries</th></tr><tr><td>Windows</td><td><a href="#">x64</a>   <a href="#">x86</a></td><td></td></tr></table>	OS	Installers	Binaries	Linux	<a href="#">Package manager instructions</a>	<a href="#">Arm32</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>	macOS		<a href="#">x64</a>	Windows	<a href="#">Hosting Bundle</a>   <a href="#">x64</a>   <a href="#">x86</a>	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>	OS	Installers	Binaries	Windows	<a href="#">x64</a>   <a href="#">x86</a>	
OS	Installers	Binaries																																	
Linux	<a href="#">Package manager instructions</a>	<a href="#">Arm32</a>   <a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>																																	
macOS	<a href="#">x64</a>	<a href="#">x64</a>																																	
Windows	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>																																	
All	<a href="#">dotnet-install scripts</a>																																		
OS	Installers	Binaries																																	
Linux	<a href="#">Package manager instructions</a>	<a href="#">Arm32</a>   <a href="#">Arm64</a>   <a href="#">Arm64 Alpine</a>   <a href="#">x64</a>   <a href="#">x64 Alpine</a>																																	
macOS		<a href="#">x64</a>																																	
Windows	<a href="#">Hosting Bundle</a>   <a href="#">x64</a>   <a href="#">x86</a>	<a href="#">Arm64</a>   <a href="#">x64</a>   <a href="#">x86</a>																																	
OS	Installers	Binaries																																	
Windows	<a href="#">x64</a>   <a href="#">x86</a>																																		

https://dotnet.microsoft.com/download/dotnet/5.0



# ASP.NET is Web Development Framework



<https://dotnet.microsoft.com/apps/aspnet>

<https://docs.microsoft.com/ko-kr/dotnet/>

<https://dotnet.microsoft.com/learn/dotnet/architecture-guides>

# ASP.NET Signal R vs ASP.NET Core Signal R

	ASP.NETSignalR	ASP.NET CoreSignalR
서버 NuGet 패키지	Microsoft.AspNet.SignalR	없음 <b>AspNetCore</b> 공유 프레임 워크에 포함 되어 있습니다.
클라이언트 NuGet 패키지	Microsoft SignalR .AspNet. 클라이언트로 Microsoft SignalR .AspNet. JS	AspNetCore SignalR . 클라이언트로
JavaScript client npm 패키지	signalr	@microsoft/signalr
Java 클라이언트	GitHub 리포지토리 (사용 되지 않음)	Maven package signalr
서버 앱 유형	ASP.NET (System.web) 또는 OWIN 자체 호스트	ASP.NET Core
지원 되는 서버 플랫폼	.NET Framework 4.5 이상	.NET Core 3.0 이상

<https://docs.microsoft.com/ko-kr/aspnet/core/signalr/version-differences?view=aspnetcore-3.1>

# ASP.NET Core Signal R is RealTime Web Messaging Technology

- ASP.NET Core 기반 크로스 플랫폼 실시간 웹 개발 오픈소스 라이브러리
- .NET 언어(C#,VB.NET)를 이용 서버에서 웹 브라우저로의 푸시 기반 어플리케이션 개발 용이
- 연결 관리 자동화 처리
- 모든 연결 된 클라이언트에 브로드 캐스팅 기능 제공 및 개별 클라이언트,그룹 메시징 기능제공
- 다양한 스케일 아웃 방식 제공

<https://docs.microsoft.com/ko-kr/aspnet/core/signalr/introduction?view=aspnetcore-3.1>

<https://github.com/dotnet/AspNetCore/tree/master/src/SignalR>

# ASP.NET Core Signal R 실시간 웹 개발하기

STEP1 : ASP.NET CORE Web Project 만들기

STEP2: SignalR Client Library 추가하기

STEP3: SignalR Hub 클래스 만들기

STEP4: SignalR 지원 Project 구성하기

STEP5: 심플 웹 채팅 페이지(뷰) 구성하기

STEP6: 웹 채팅 테스트 하기

STEP7: 그룹채팅/타겟 메시징 처리하기

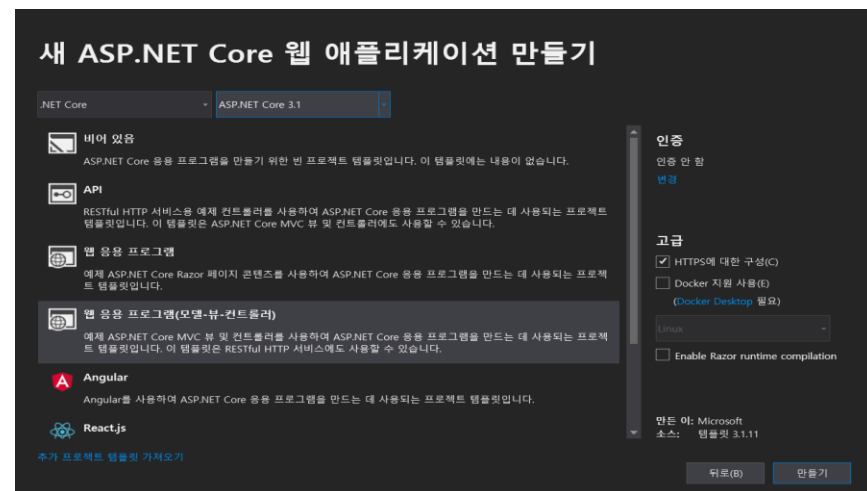
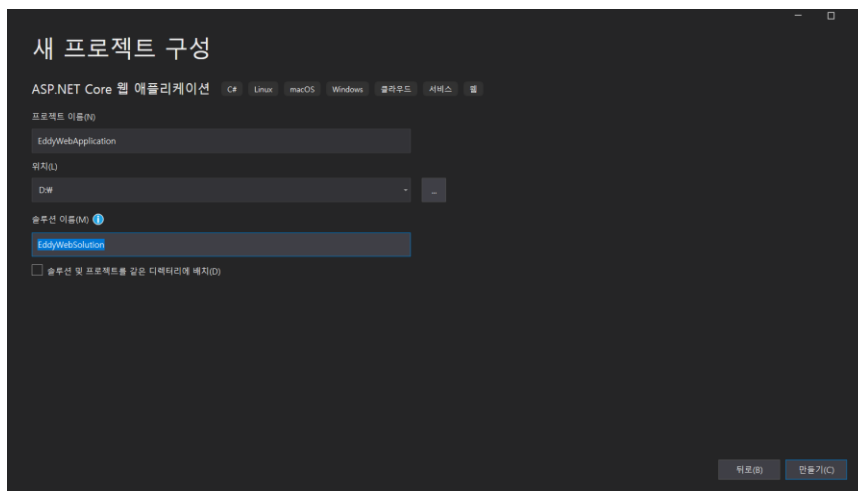
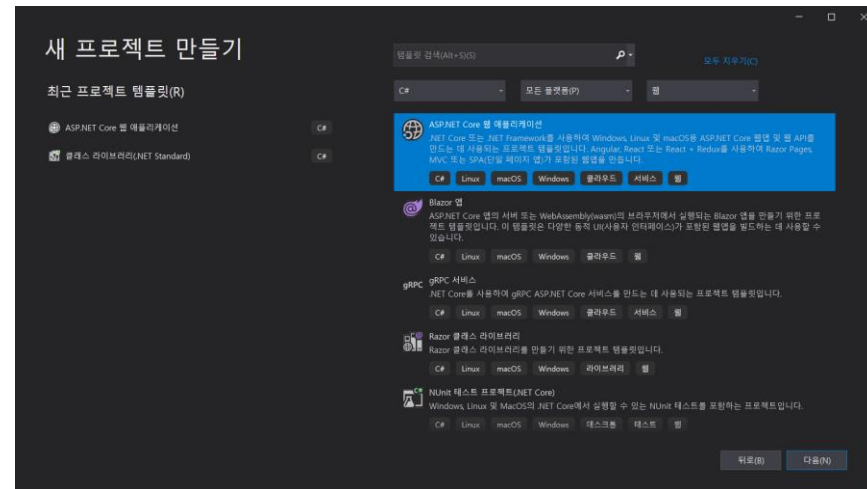
STEP8: SignalR 크로스 도메인(CORS) 지원 설정하기

<https://docs.microsoft.com/ko-kr/aspnet/core/tutorials/signalr?view=aspnetcore-3.1&tabs=visual-studio>

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP1 : ASP.NET CORE Web Project 만들기

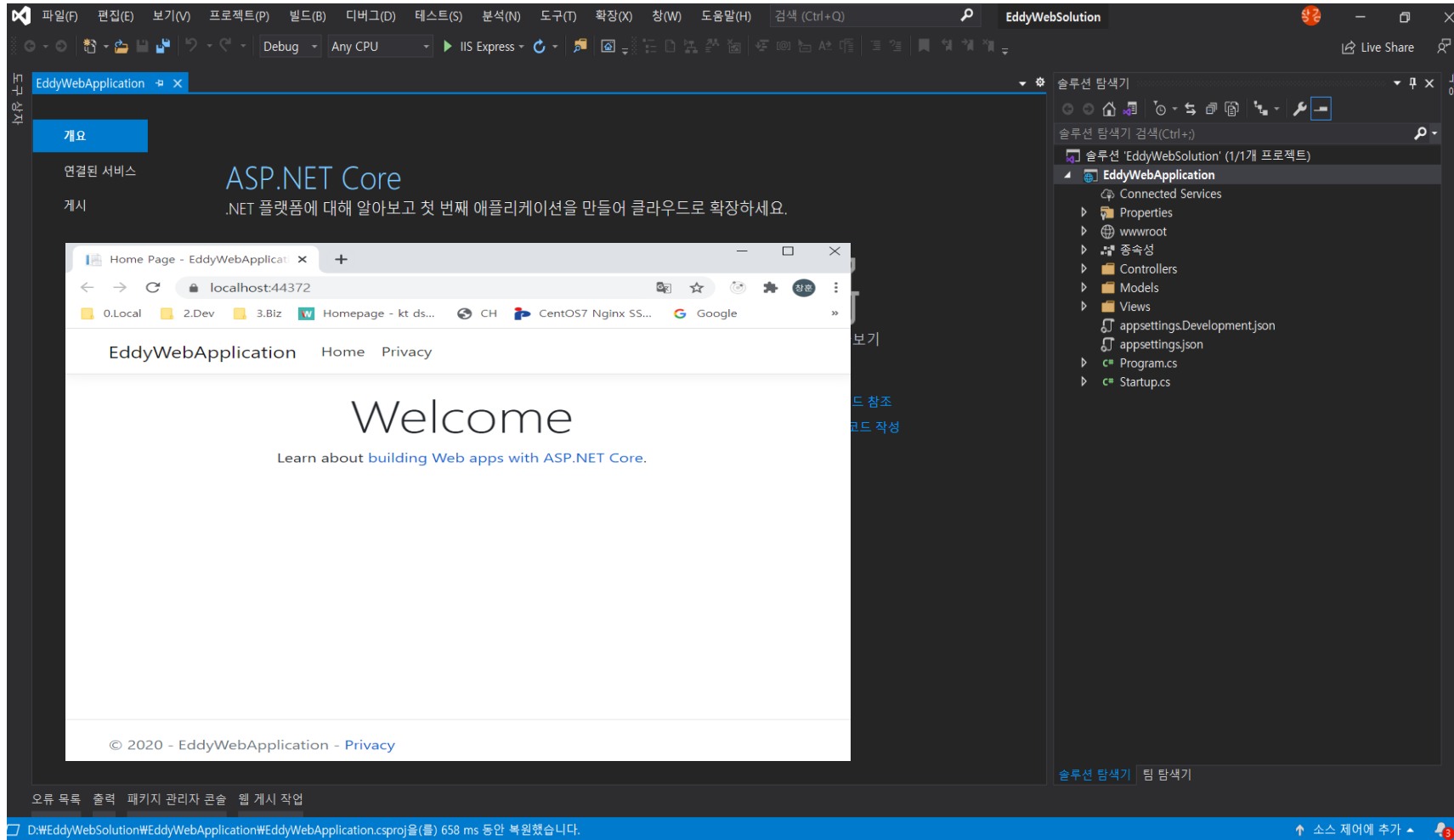
1. Visual Studio 2019를 시작합니다.
2. 새프로젝트 만들기를 클릭합니다.
3. 언어를 C# 프로젝트 형식을 웹을 선택합니다.
  - ASP.NET Core 웹 애플리케이션 프로젝트 템플릿을 선택하고 다음>
4. 새프로젝트 구성 화면에서 아래 내용을 입력하고 만들기 클릭
  - 프로젝트명 과 솔루션명을 입력합니다. Ex) MyWebApplication MyWebSolution
  - 솔루션 폴더가 생성될 경로를 지정합니다. Ex) D:\
5. 새 ASP.NET Core 웹 애플리케이션 만들기 화면에서 하기 내용 선택 후 만들기 클릭
  - .NET Core Framework 선택
  - ASP.NET Core 3.1 선택
  - 웹 응용 프로그램(모델-뷰-컨트롤러) 프로젝트 템플릿 선택 후 만들기 클릭



<https://docs.microsoft.com/ko-kr/aspnet/core/tutorials/signalr?view=aspnetcore-3.1&tabs=visual-studio>

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP1 : ASP.NET CORE Web Project 만들기



### 1. 솔루션 탐색기 확인

-솔루션 아래 프로젝트 구조확인

### 2. EddyWebApplication 실행하기

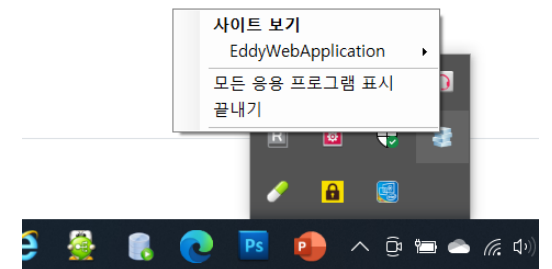
- F5 또는 디버그 메뉴> 디버깅 시작 클릭  
- 또는 화면내 초록색 디버깅 버튼 클릭

### 3.웹브라우저에서 해당 어플리케이션 실행확인

-Controllers : MVC 에 Controller 역할제공  
-Models : MVC에 Model 역할제공  
-Views : MVC에 View 역할 제공

### 4.트레이바 > IIS Express 개발 웹서버 확인

### 5.웹 브라우저를 닫으면 디버깅 자동 종료

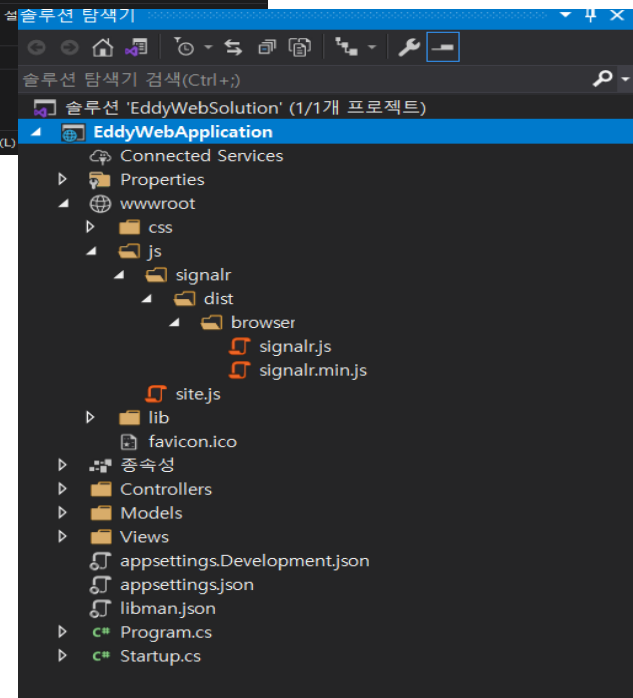
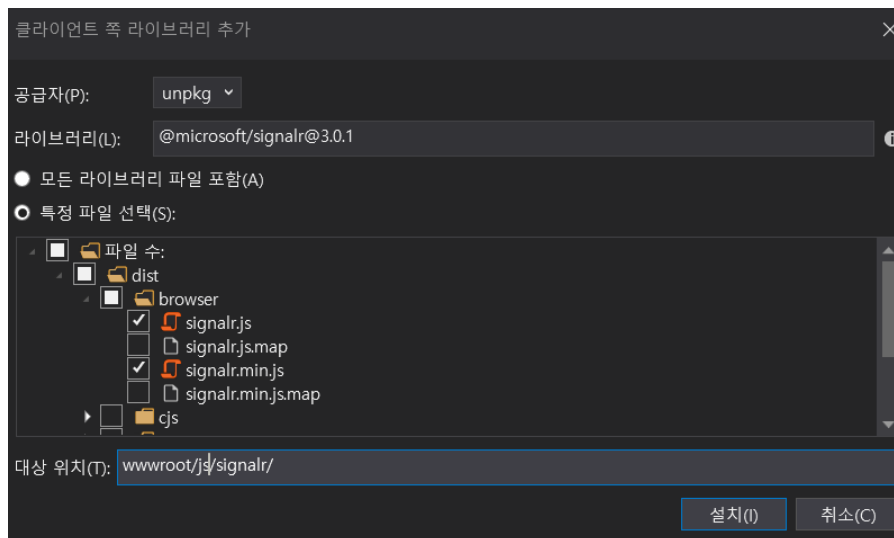
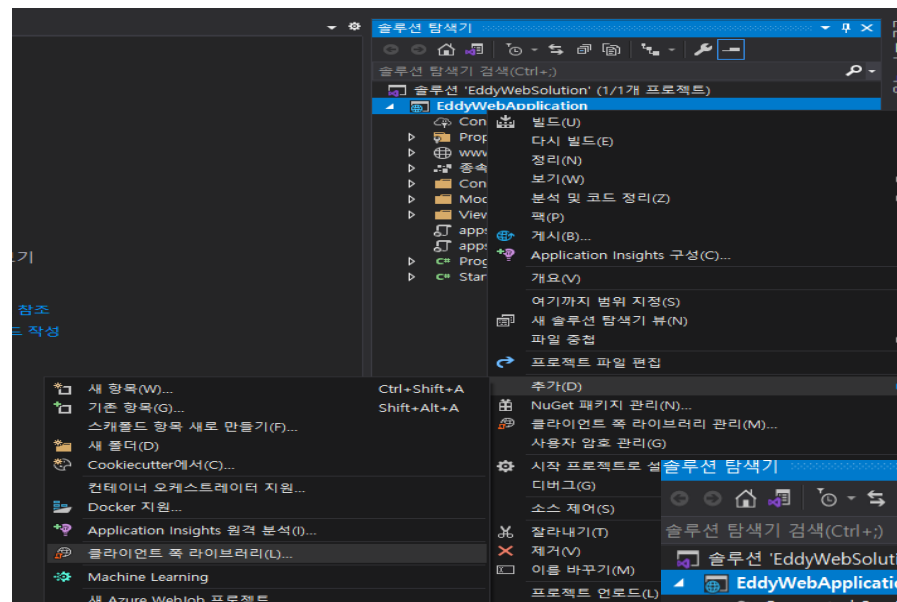


<https://docs.microsoft.com/ko-kr/aspnet/core/tutorials/signalr?view=aspnetcore-3.1&tabs=visual-studio>

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP2: SignalR Client Library 추가하기

1. 프로젝트 오른쪽 마우스 클릭 > 추가 클릭>
2. 클라이언트 쪽 라이브러리 추가를 클릭합니다.
3. 공급자: unpkg 라이브러리 : signalr검색 후 해당 입력 후
4. 특정 파일 선택 후 대상위치를 지정후 설치 클릭  
파일선택 : dist/browser/signalr.js 과 signalr.min.js 파일 선택  
대상위치: **wwwroot/js/signalr** 로 지정
5. 솔루션 탐색기에 해당 자바스크립트 라이브러리 추가여부 확인  
**wwwroot\js\signalr\dist\browser\**



<https://docs.microsoft.com/ko-kr/aspnet/core/tutorials/signalr?view=aspnetcore-3.1&tabs=visual-studio>

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP3: SignalR Hub 클래스 만들기

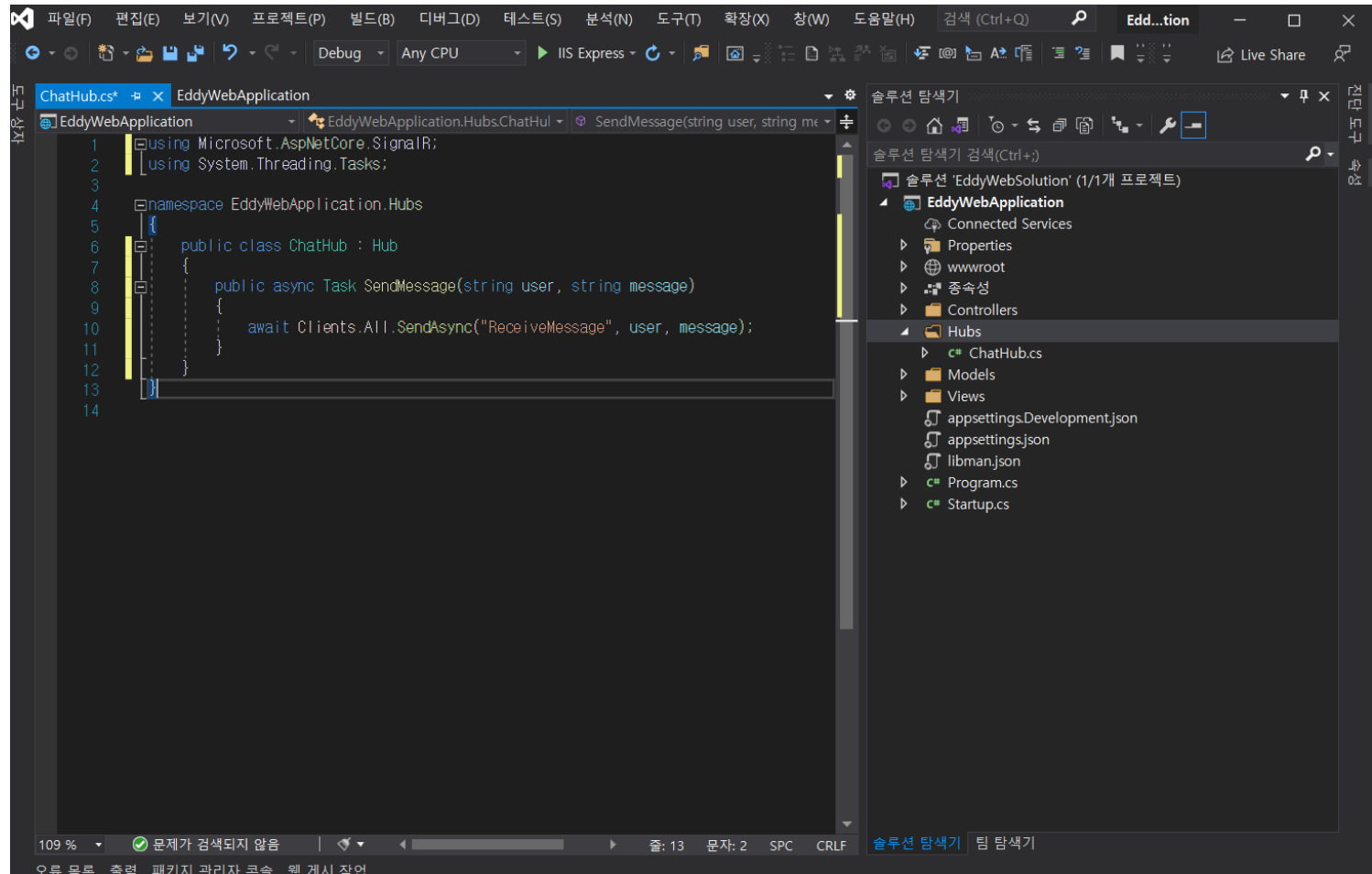
1. 프로젝트에 오른쪽 마우스 클릭 > 추가> 새폴더
2. 폴더명을 Hubs로 지정합니다.
3. Hubs폴더를 선택하고 오른쪽 마우스 클릭 > 추가 > 클래스
4. 클래스명을 ChatHub.cs로 지정후 추가합니다.
5. ChatHub.cs 에 우측과 같이 코딩을 진행합니다.

//참조추가

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

//Hub 클래스 상속

```
Public class ChatHub : Hub  
{  
    //웹브라우저 송신 메시지 수신 및 모든 브라우저에게 재발송  
    public async Task SendMessage(string user, string message)  
    {  
        await Clients.All.SendAsync("ReceiveMessage", user, message);  
    }  
}
```





# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP4: SignalR 지원 Project 구성하기

1. 프로젝트 루트내 Startup.cs 열어 SignalR 서비스 환경설정 추가하기

**참조주의 : 여러분 프로젝트명.Hubs;**

**//참조하기**

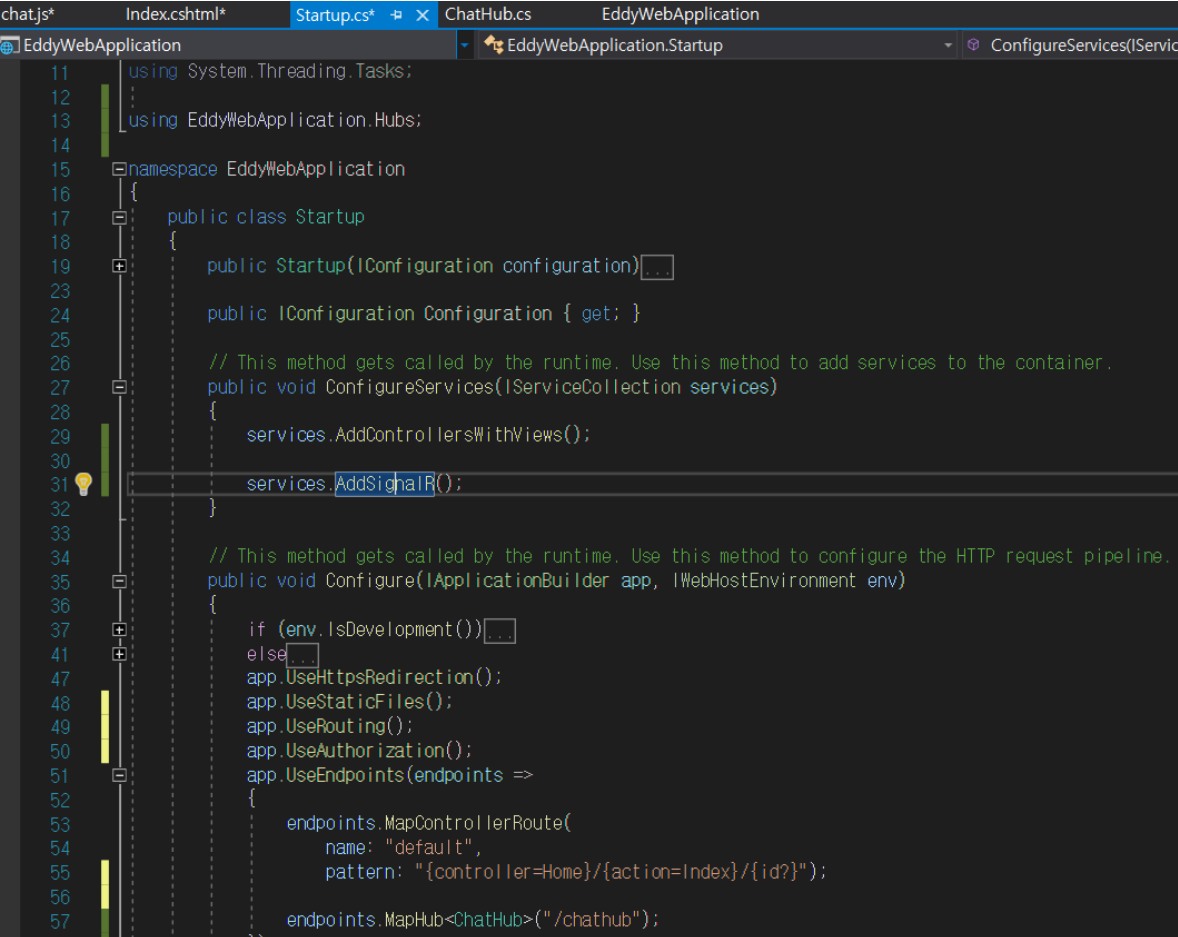
**using EddyWebApplication.Hubs;**

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddSignalR();
}
```

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    .
    .

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");

        endpoints.MapHub<ChatHub>("/chathub");
    });
}
```



```
11 using System.Threading.Tasks;
12
13 using EddyWebApplication.Hubs;
14
15 namespace EddyWebApplication
16 {
17     public class Startup
18     {
19         public Startup(IConfiguration configuration)
20         {
21             Configuration = configuration;
22         }
23
24         public IConfiguration Configuration { get; }
25
26         // This method gets called by the runtime. Use this method to add services to the container.
27         public void ConfigureServices(IServiceCollection services)
28         {
29             services.AddControllersWithViews();
30
31             services.AddSignalR();
32         }
33
34         // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
35         public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
36         {
37             if (env.IsDevelopment())
38             {
39                 app.UseDeveloperExceptionPage();
40             }
41             else
42             {
43                 app.UseHttpsRedirection();
44                 app.UseStaticFiles();
45                 app.UseRouting();
46                 app.UseAuthorization();
47                 app.UseEndpoints(endpoints =>
48                 {
49                     endpoints.MapControllerRoute(
50                         name: "default",
51                         pattern: "{controller=Home}/{action=Index}/{id?}");
52
53                     endpoints.MapHub<ChatHub>("/chathub");
54                 });
55             }
56         }
57     }
58 }
```

<https://docs.microsoft.com/ko-kr/aspnet/core/signalr/javascript-client?view=aspnetcore-3.1>

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP5: 심플 웹 채팅 페이지(뷰) 구성하기

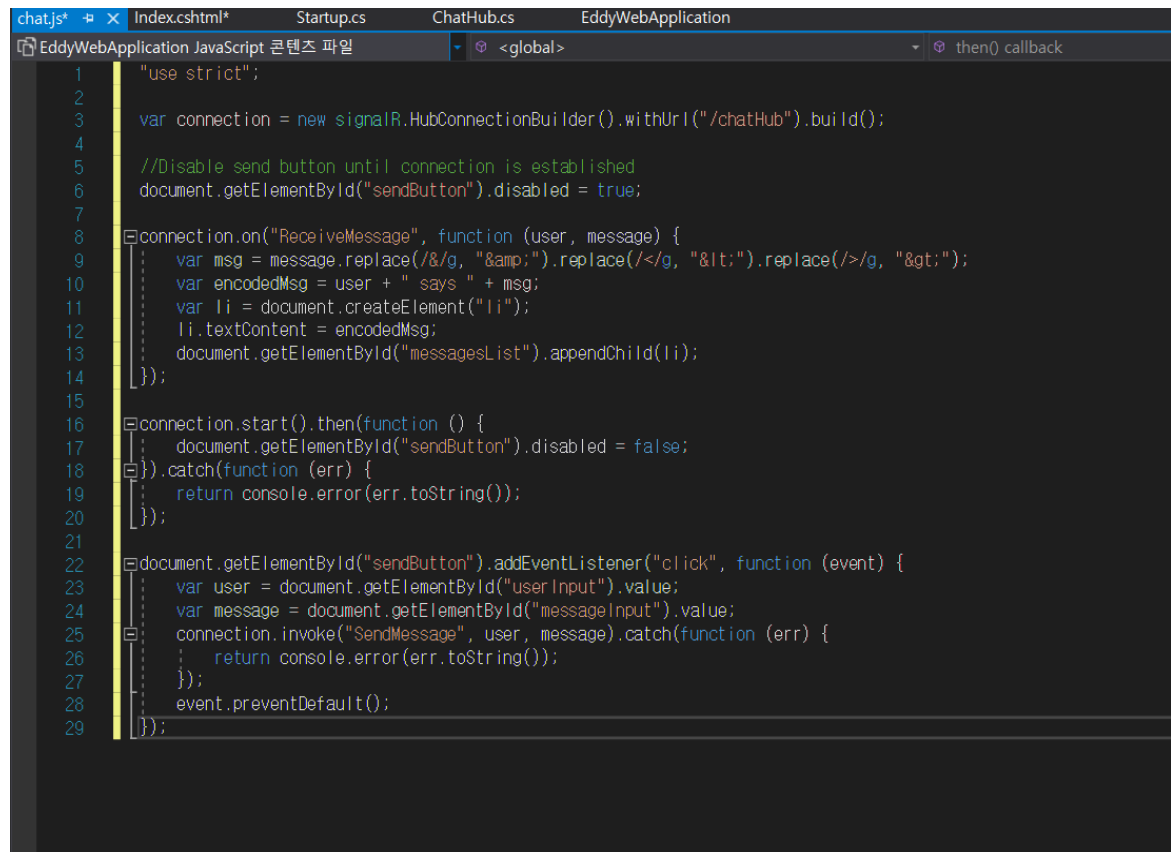
1. 프로젝트내 Views폴더내 Index.cshtml 뷰페이지를 엽니다.
2. 뷰페이지내 모든 내용을 삭제하고 우측 내용을 붙여넣습니다.
3. 맨 하단의 자바스크립트 라이브러리 참조경로 및 참조 파일을 확인합니다.  
(하기 링크 파일내 관련 소스 복사 붙여넣기 )

```
Index.cshtml* Startup.cs ChatHub.cs EddyWebApplication
1  ViewData["Title"] = "Home Page";
2
3
4
5  <div class="container">
6      <div class="row">&nbsp;</div>
7      <div class="row">
8          <div class="col-2">User</div>
9          <div class="col-4"><input type="text" id="userInput" /></div>
10     </div>
11     <div class="row">
12         <div class="col-2">Message</div>
13         <div class="col-4"><input type="text" id="messageInput" /></div>
14     </div>
15     <div class="row">&nbsp;</div>
16     <div class="row">
17         <div class="col-6">
18             <input type="button" id="sendButton" value="Send Message" />
19         </div>
20     </div>
21 </div>
22 <div class="row">
23     <div class="col-12">
24         <hr />
25     </div>
26 </div>
27 <div class="row">
28     <div class="col-6">
29         <ul id="messagesList"></ul>
30     </div>
31 </div>
32 <script src="~/js/signalr/dist/browser/signalr.js"></script>
33 <script src="~/js/chat.js"></script>
```

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP5: 심플 웹 채팅 페이지(뷰) 구성하기

1. 프로젝트 내 **wwwroot\js** 폴더에 오른쪽 마우스 클릭 > 추가> 새 항목 클릭
2. javascript 파일 항목을 선택하고 이름을 **chat.js** 로 입력 후 추가합니다.
3. 스크립트 파일내 우측 내용을 코딩합니다.  
(하기 링크 파일내 관련 소스 복사 붙여넣기)

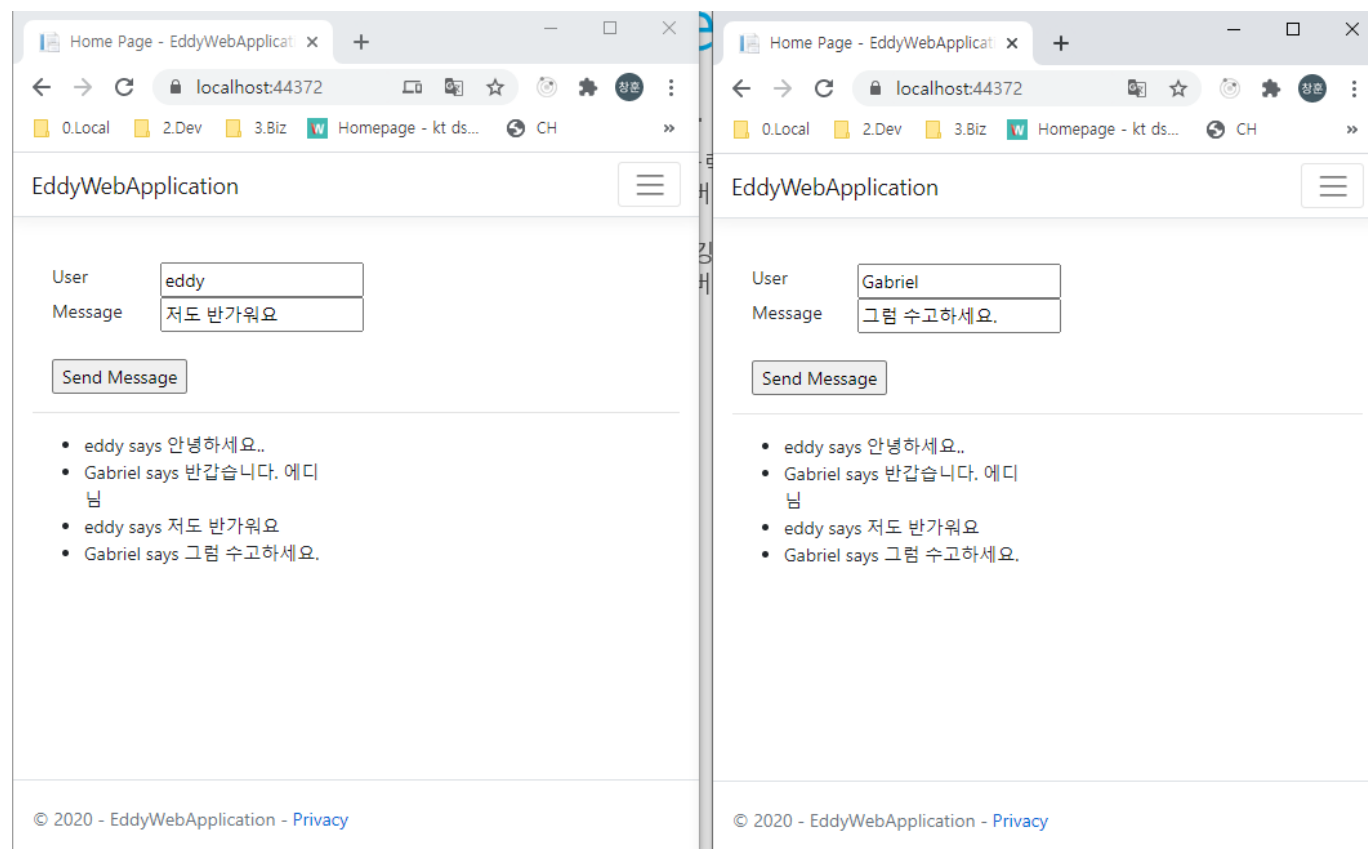


```
1  "use strict";
2
3  var connection = new signalR.HubConnectionBuilder().withUrl("/chatHub").build();
4
5  //Disable send button until connection is established
6  document.getElementById("sendButton").disabled = true;
7
8  connection.on("ReceiveMessage", function (user, message) {
9      var msg = message.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;");
10     var encodedMsg = user + " says " + msg;
11     var li = document.createElement("li");
12     li.textContent = encodedMsg;
13     document.getElementById("messagesList").appendChild(li);
14 });
15
16 connection.start().then(function () {
17     document.getElementById("sendButton").disabled = false;
18 }).catch(function (err) {
19     return console.error(err.toString());
20 });
21
22 document.getElementById("sendButton").addEventListener("click", function (event) {
23     var user = document.getElementById("userInput").value;
24     var message = document.getElementById("messageInput").value;
25     connection.invoke("SendMessage", user, message).catch(function (err) {
26         return console.error(err.toString());
27     });
28     event.preventDefault();
29 });
```

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP6: 웹 채팅 테스트 하기

1. 상단 메뉴 파일> 모두저장을 클릭합니다.
2. F5 또는 상단 디버그메뉴> 디버깅 시작을 클릭하여 웹 어플리케이션을 디버깅 모드로 실행합니다.
  - F5 또는 디버그 메뉴> 디버깅 시작 클릭
  - 또는 화면내 초록색 디버깅 버튼 클릭
3. 두개의 웹브라우저 창을 열고 웹페이지 주소를 통해 접속합니다.  
<https://localhost:44372/>
4. User에 대화명을 Message에는 대화내용을 입력 후 Send버튼을 클릭합니다.
5. 정상적으로 두 브라우저간 채팅이 이루어지면 성공!
6. 그렇지 않으면 질문??

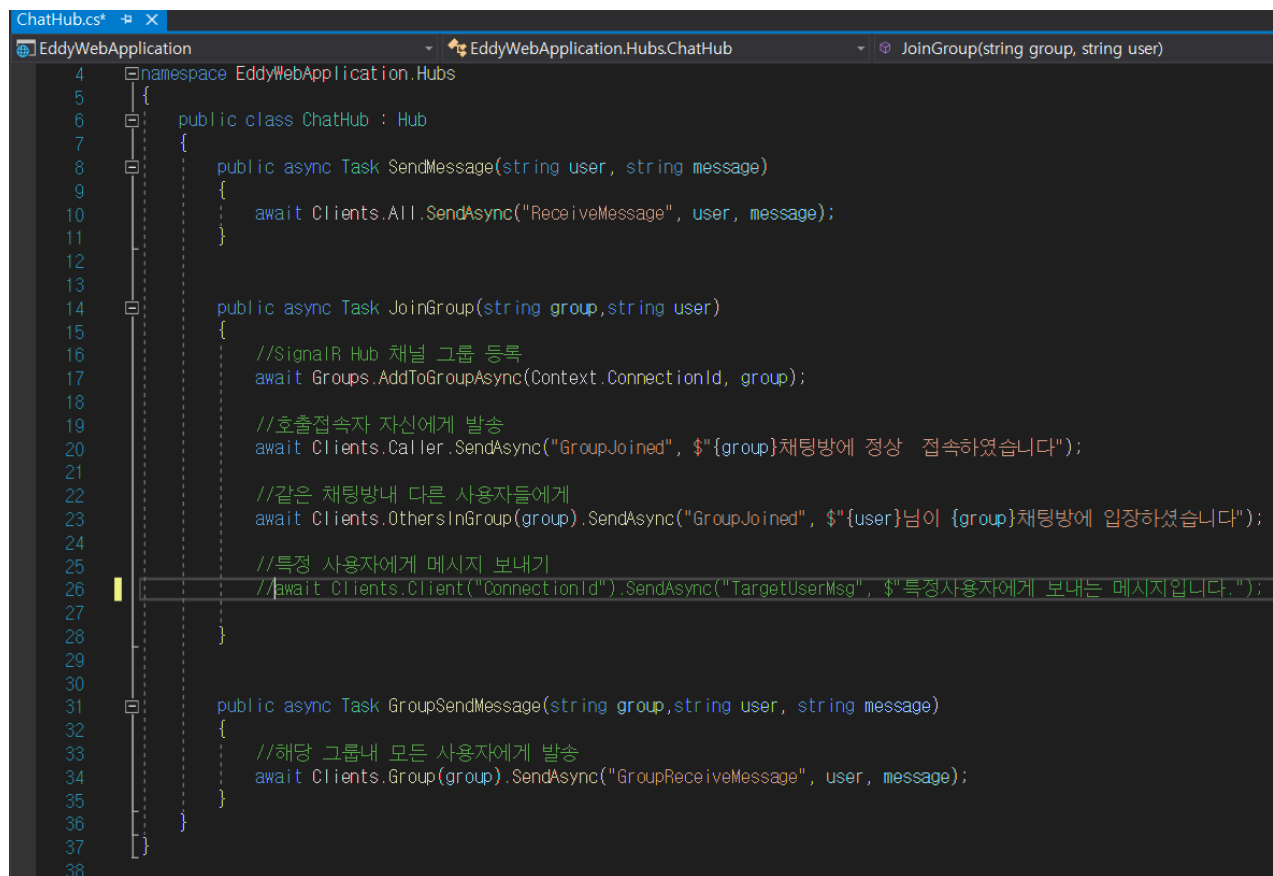


<https://docs.microsoft.com/ko-kr/aspnet/core/signalr/javascript-client?view=aspnetcore-3.1>

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP7: 그룹채팅/타겟 메시징 처리하기

1. 그룹채팅을 위한 뷰페이지를 하나 추가합니다.
2. 프로젝트내 Controllers폴더내 HomeController.cs 페이지를 엽니다.
  - Index액션 메소드를 복사해서 Group 메소드를 만듭니다.
  - Group메소드에 오른쪽 마우스 클릭 뷰추가를 클릭합니다.
  - Razor 뷰를 선택하고 추가를 클릭합니다.
3. Views폴더내 Group.cshtml 뷰 페이지가 추가됨을 확인하고 엽니다.
  - Index.cshtml페이지 내용을 복사해 붙여넣기합니다.
4. Js/chat.js대신에 빈 스크립트 태그를 추가하고 chat.js내용을 해당 스크립트 태그블럭에 복붙합니다.
5. 그룹채팅 로직을 구현합니다.



```
ChatHub.cs*
EddyWebApplication
EddyWebApplication.Hubs.ChatHub
JoinGroup(string group, string user)

4 namespace EddyWebApplication.Hubs
5 {
6     public class ChatHub : Hub
7     {
8         public async Task SendMessage(string user, string message)
9         {
10             await Clients.All.SendAsync("ReceiveMessage", user, message);
11         }
12
13         public async Task JoinGroup(string group, string user)
14         {
15             //SignalR Hub 채널 그룹 등록
16             await Groups.AddToGroupAsync(Context.ConnectionId, group);
17
18             //호출접속자 자신에게 발송
19             await Clients.Caller.SendAsync("GroupJoined", $"{group}채팅방에 정상 접속하였습니다");
20
21             //같은 채팅방내 다른 사용자에게
22             await Clients.OthersInGroup(group).SendAsync("GroupJoined", $"{user}님이 {group}채팅방에 입장하셨습니다");
23
24             //특정 사용자에게 메시지 보내기
25             //await Clients.Client("ConnectionId").SendAsync("TargetUserMsg", $"특정사용자에게 보내는 메시지입니다.");
26
27         }
28
29         public async Task GroupSendMessage(string group, string user, string message)
30         {
31             //해당 그룹내 모든 사용자에게 발송
32             await Clients.Group(group).SendAsync("GroupReceiveMessage", user, message);
33         }
34     }
35 }
36
37
38
```

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP7: 그룹채팅/타겟 메시징 처리하기

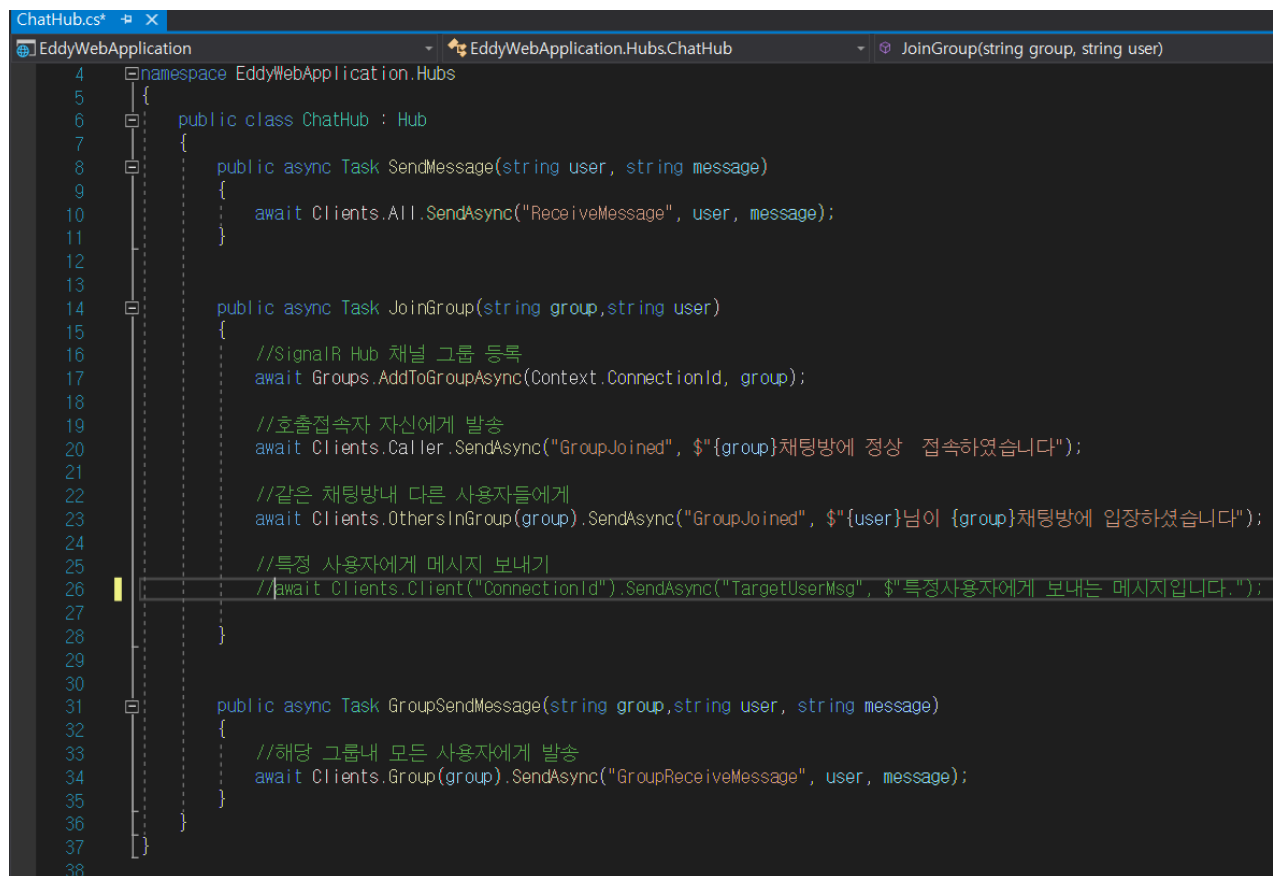
```
//SignalR Hub 채널 그룹 등록
await Groups.AddToGroupAsync(Context.ConnectionId, group);

//호출접속자 자신에게 발송
await Clients.Caller.SendAsync("GroupJoined", $"{group}채팅방에 정상 접속하였습니다");

//같은 채팅방내 다른 사용자에게
await Clients.OthersInGroup(group).SendAsync("GroupJoined", $"{user}님이 {group}채팅방에 입장하셨습니다");

//특정 사용자에게 메시지 보내기
//await Clients.Client("ConnectionId").SendAsync("TargetUserMsg", $"특정사용자에게 보내는 메시지입니다.");

//해당 그룹내 모든 사용자에게 발송
await Clients.Group(group).SendAsync("GroupReceiveMessage", user, message);
```



```
ChatHub.cs
EddyWebApplication
EddyWebApplication.Hubs.ChatHub
JoinGroup(string group, string user)

4 namespace EddyWebApplication.Hubs
5 {
6     public class ChatHub : Hub
7     {
8         public async Task SendMessage(string user, string message)
9         {
10             await Clients.All.SendAsync("ReceiveMessage", user, message);
11         }
12
13         public async Task JoinGroup(string group, string user)
14         {
15             //SignalR Hub 채널 그룹 등록
16             await Groups.AddToGroupAsync(Context.ConnectionId, group);
17
18             //호출접속자 자신에게 발송
19             await Clients.Caller.SendAsync("GroupJoined", $"{group}채팅방에 정상 접속하였습니다");
20
21             //같은 채팅방내 다른 사용자에게
22             await Clients.OthersInGroup(group).SendAsync("GroupJoined", $"{user}님이 {group}채팅방에 입장하셨습니다");
23
24             //특정 사용자에게 메시지 보내기
25             //await Clients.Client("ConnectionId").SendAsync("TargetUserMsg", $"특정사용자에게 보내는 메시지입니다.");
26         }
27
28
29
30         public async Task GroupSendMessage(string group, string user, string message)
31         {
32             //해당 그룹내 모든 사용자에게 발송
33             await Clients.Group(group).SendAsync("GroupReceiveMessage", user, message);
34         }
35     }
36 }
37
38
```

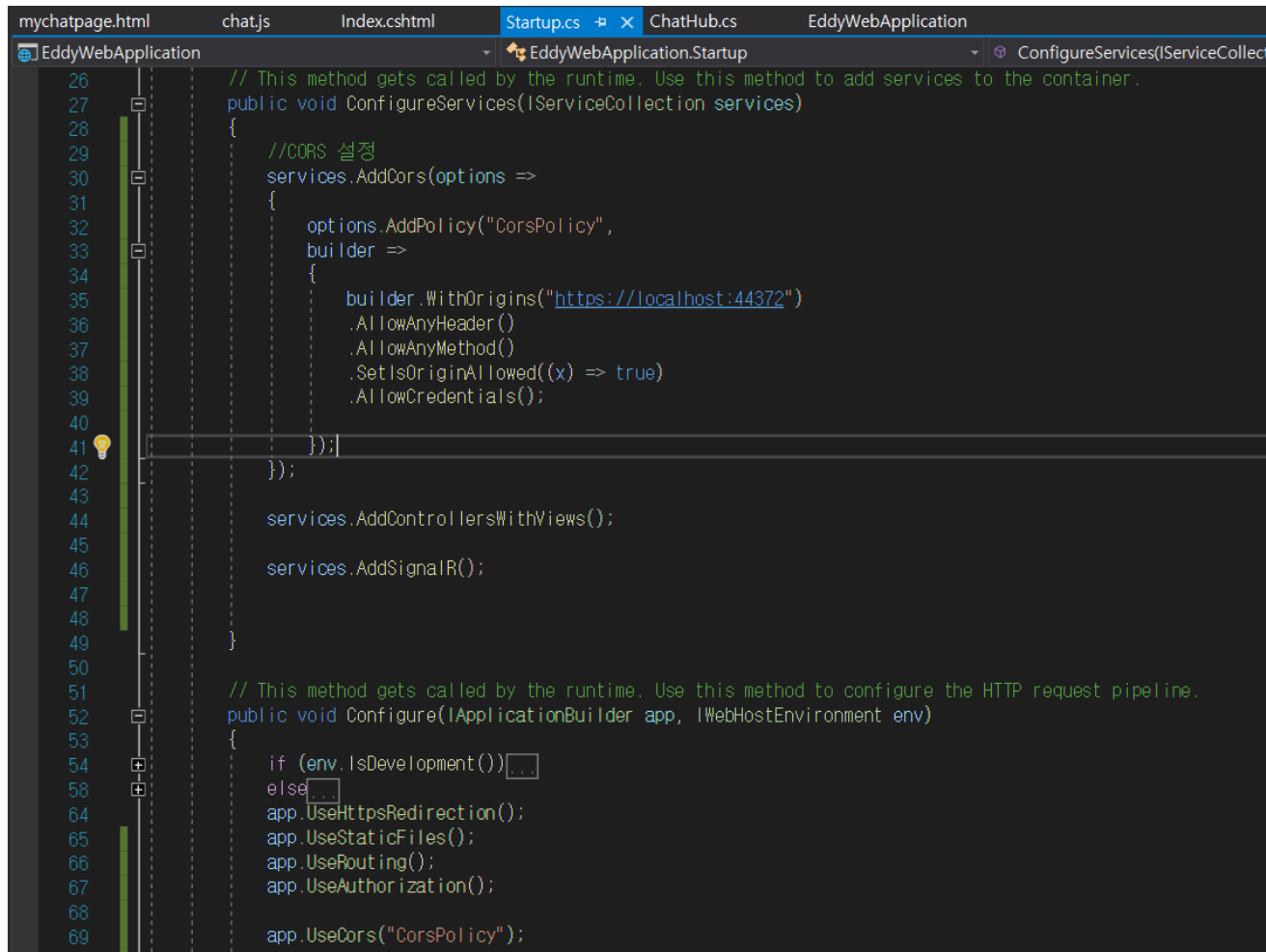
# ASP.NET Core Signal R 실시간 웹 개발하기-CORS

## STEP8: 각종 클라이언트 지원 - SignalR 크로스 도메인(CORS) 지원 설정하기

### 1. CORS 설정하기 – Startup.cs

```
//CORS 설정
services.AddCors(options =>
{
    options.AddPolicy("CorsPolicy",
        builder =>
        {
            builder.WithOrigins("https://localhost:44372")
                .AllowAnyHeader()
                .AllowAnyMethod()
                .SetIsOriginAllowed((x) => true)
                .AllowCredentials();
        }
    );
});

app.UseCors("CorsPolicy");
```



```
mychatpage.html chat.js Index.cshtml Startup.cs ChatHub.cs EddyWebApplication
EddyWebApplication
EddyWebApplication.Startup
ConfigureServices(IServiceCollection services)
26 // This method gets called by the runtime. Use this method to add services to the container.
27 public void ConfigureServices(IServiceCollection services)
28 {
29     //CORS 설정
30     services.AddCors(options =>
31     {
32         options.AddPolicy("CorsPolicy",
33             builder =>
34             {
35                 builder.WithOrigins("https://localhost:44372")
36                     .AllowAnyHeader()
37                     .AllowAnyMethod()
38                     .SetIsOriginAllowed((x) => true)
39                     .AllowCredentials();
40             }
41         );
42     });
43
44     services.AddControllersWithViews();
45
46     services.AddSignalR();
47
48 }
49
50 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
51 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
52 {
53     if (env.IsDevelopment())
54     {
55         app.UseDeveloperExceptionPage();
56     }
57     else
58     {
59         app.UseExceptionHandler("/Home/Error");
60         app.UseHttpsRedirection();
61         app.UseStaticFiles();
62         app.UseRouting();
63         app.UseAuthorization();
64
65         app.UseCors("CorsPolicy");
66     }
67 }
```

# ASP.NET Core Signal R 실시간 웹 개발하기

## STEP8: 자바스크립트 클라이언트 기반 채팅하기

1. 윈도우 탐색기내 C:\에 ChatSample폴더를 만듭니다.
2. Visual Studio 2019를 하나더 오픈하고 로컬폴더열기를 클릭하여 해당 폴더를 선택합니다.
3. 폴더에 오른쪽 마우스 클릭 > 새항목 추가 > html페이지 선택 후 chat.html 페이지를 생성합니다.
4. Html 내용만 index.cshtml에서 복사해와 body태그 사이에 넣습니다.
5. signalr javascript client library를 CDN참조합니다.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-signalr/3.1.3/signalr.min.js"></script>
```

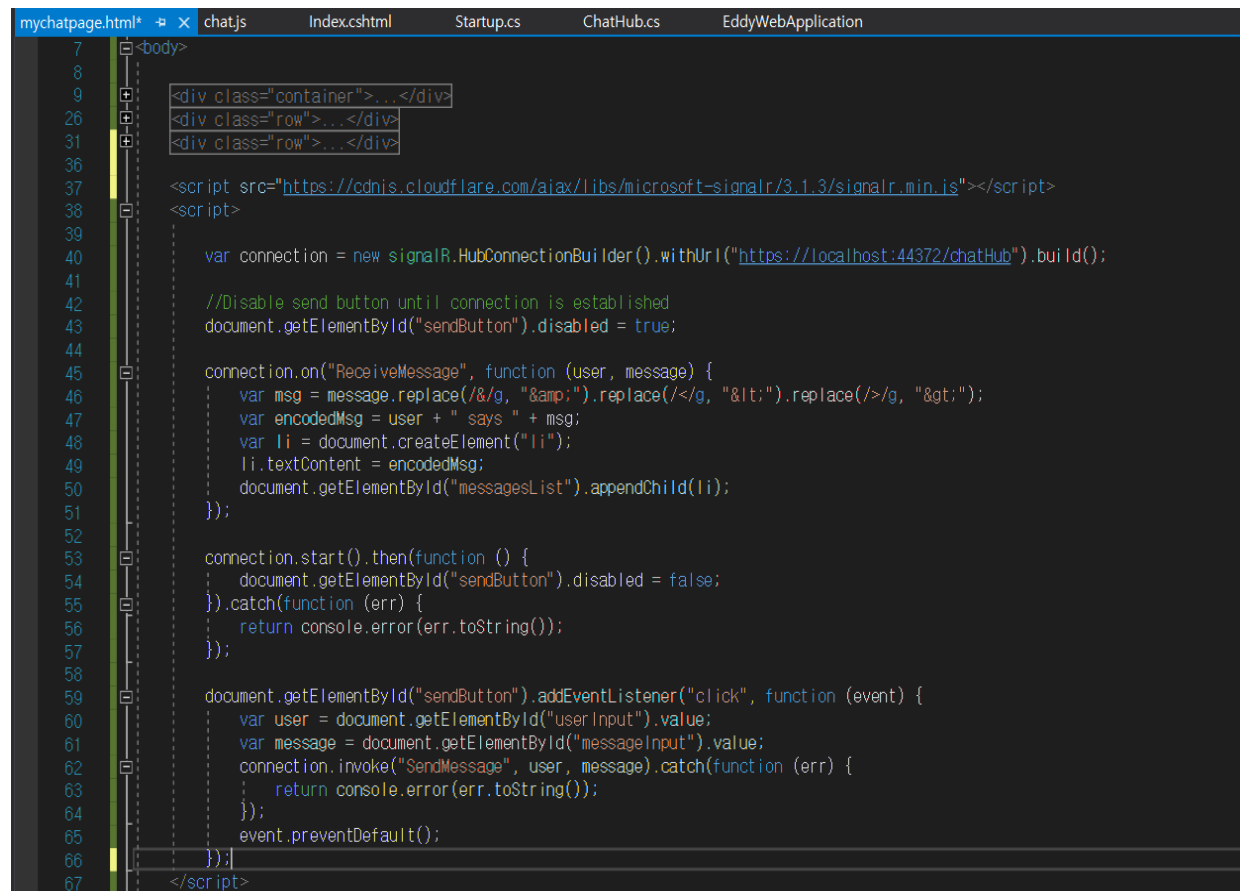
```
<script>
```

```
var connection = new  
signalR.HubConnectionBuilder().withUrl("https://localhost:44372/chatHub").build();
```

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

```
.  
.   
.
```

```
</script>
```



```
mychatpage.html*  X chatjs  Index.cshtml  Startup.cs  ChatHub.cs  EddyWebApplication  
7  <body>  
8  
9  <div class="container">...</div>  
26 <div class="row">...</div>  
31 <div class="row">...</div>  
36  
37 <script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-signalr/3.1.3/signalr.min.js"></script>  
38 <script>  
39  
40     var connection = new signalR.HubConnectionBuilder().withUrl("https://localhost:44372/chatHub").build();  
41  
42     //Disable send button until connection is established  
43     document.getElementById("sendButton").disabled = true;  
44  
45     connection.on("ReceiveMessage", function (user, message) {  
46         var msg = message.replace(/&/g, "&amp;").replace(/</g, "&lt;").replace(/>/g, "&gt;");  
47         var encodedMsg = user + " says " + msg;  
48         var li = document.createElement("li");  
49         li.textContent = encodedMsg;  
50         document.getElementById("messagesList").appendChild(li);  
51     });  
52  
53     connection.start().then(function () {  
54         document.getElementById("sendButton").disabled = false;  
55     }).catch(function (err) {  
56         return console.error(err.toString());  
57     });  
58  
59     document.getElementById("sendButton").addEventListener("click", function (event) {  
60         var user = document.getElementById("userInput").value;  
61         var message = document.getElementById("messageInput").value;  
62         connection.invoke("SendMessage", user, message).catch(function (err) {  
63             return console.error(err.toString());  
64         });  
65         event.preventDefault();  
66     });  
67 </script>
```

<https://docs.microsoft.com/ko-kr/aspnet/core/signalr/javascript-client?view=aspnetcore-3.1>



# 감사합니다.

GitHub : <https://github.com/iami5246/aspnetcore-signalr>

엠소프트웨어 대표 강창훈

<https://msoftware.co.kr>

[ceo@msoftware.co.kr](mailto:ceo@msoftware.co.kr)

010-2760-5246