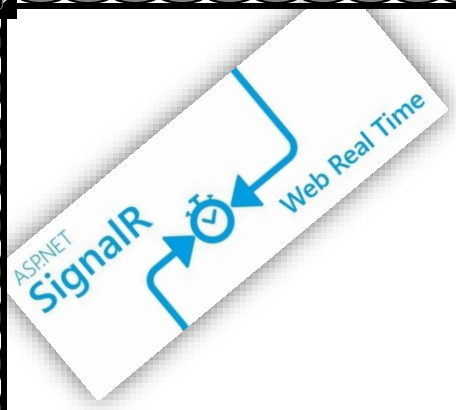


مقدمه ای بر ASP.NET Core SignalR

ترجمه : مهدی کیانی

مقدمه

SignalR چیست؟[1]

سیگنال یک کتابخانه ای است برای ساده نمودن پیاده سازی عملیات بلادرنگ در برنامه های تحت وب.

برنامه هایی که دارای پیاده سازی سینگال می باشند می توانند داده هایی را مستقیما از سمت سرور به سمت کلاینت ارسال کنند. این روش ارسال داده را پوش کردن اطلاعات می نامند.

تفاوت پوش[2] و پول[3] در ارسال و دریافت اطلاعات در برنامه های تحت وب

اغلب برنامه هایی که در بستر وب نوشته می شوند از روش پول برای دریافت اطلاعات از سمت سرور استفاده می کنند. روش پول به این صورت است که هر زمان نیاز به اطلاعات از سمت سرور بود، با ارسال درخواستی به سرور اطلاعات مورد نظر واکشی می شود. این روش اگر چه برای اغلب برنامه ها مناسب می باشد اما دارای یک ایراد می باشد. اجازه دهید تا با یک مثال محدودیت این روش را در برخی از سناریو ها بیان کنم.

فرض کنید شما نیاز به یک برنامه پشتیبانی(تیکتینگ) دارید. این مثال می تواند نمونه خوبی از یک پیاده سازی بلادرنگ باشد. این برنامه را می توانید به دوصورت پیاده سای کنید.

روش اول همان استفاده از روش پول می باشد. در این روش کاربر کلاینت از پنل خود برای تیم پشتیبان درخواستی را ارسال می کند.کاربر پشتیبان پس از رویت درخواست، اقدام به پاسخ می نماید.کاربرکلاینت تا زمانی که صفحه خود را بارگزاری مجدد نکند(ارسال درخواست مجدد به سمت سرور) نمی تواند از پاسخ کاربرپشتیبان، مطلع شود. به این روش روش پولینگ می گویند که در اغلب برنامه های موجود مورد استفاده قرار می گیرد. در واقع در روش پولینگ، کاربر با ارسال درخواست های مکرر به سرور تقاضای اطلاعات می کند.

روش دوم و در مقابل روش پولینگ، روشی است که پوشینگ نام دارد. در این روش به محضی که اطلاعات در سمت سرور مهیا شد، این اطلاعات در کلاینت پوش می شود(از طریق فراخوانی غیر مستقیم در سیگنال). در این حالت دیگر نیازی نیست تا کاربر کلاینت دائما با رفرش کردن صفحه یا با فشردن دکمه ای بررسی کند که آیا اطلاعات در سمت سرور آماده است یا خیر. این روش در سناریو های بلادرنگ مورد استفاده قرار می گیرد.

نکته : کتابخانه سیگنال قرار است پیاده سازی روش پوشینگ را برای شما به ساده ترین شکل انجام دهد.

---

[1] در ادامه مقاله به جهت ساده نویسی، به جای واژه SignalR از کلمه سیگنال استفاده شده است.

[2] Push

[3] Pull

برخی از سناریو های مستعد استفاده از سیگنال

- ✓ برنامه هایی که بروزرسانی های متعددی در سمت سرور دارند. مانند شبکه های اجتماعی، تیکتینگ های آنلاین، بازی ها و ...
- ✓ برنامه هایی که نیاز به عملیات مونیتورینگ دارند. [۱]
- ✓ برنامه های Collaborative [۲]
- ✓ برنامه های مبتنی بر Notification [۳]

سیگنال چگونه کار میکند؟

سیگنال یک API ای برای ایجاد یک RPC[۴] از سمت سرور به کلاینت فراهم می کند که توسط آن می توان کد های جاوا اسکریپت سمت کلاینت را توسط کد های سمت سرور فراخوانی کرد.

سیگنال تمانی کلاینت هایی که به سرور متصل شده اند را شناسایی و ارتباطات آن ها را[۵]به صورت خودکار مدیریت می کند. سیگنال قابلیت دارد تا پیام هایی را به صورت دسته جمعی برای همه کلاینت های متصل شده به سرور ارسال کند،[۶] یا پیام های خاصی را برای کلاینت خاص یا گروهی از کلاینت ها ارسال نماید.[۷]مانند برنامه های چت گروهی.

مفهوم Hub در سیگنال

سیگنال برای پیاده سازی روند ارسال و دریافت اطلاعات از مفهومی به نام Hub استفاده می کند. هاب را می توانید شبیه به یک خیابان دوطرفه بدانید که یک سمت آن سرور و سمت دیگرآن کلاینت قرارداشته و کلاینت و سرور را قادر به فراخوانی کدهای یکدیگر می کند. هاب با ارسال پیام هایی که حاوی نام و پارامتر های متد سمت کلاینت می باشد اقدام به فراخوانی آن ها می کند. پیام هایی که توسط هاب ارسال می شوند می توانند توسط یکی از دو پروتکل زیر انجام پذیرند :

---

۱ مانند داشبورد های مدیریتی که امروزه در بسیاری از برنامه ها مورد استفاده قرار می گیرد.

۲ برنامه های مبتنی بر همکاری طرفین به صورت بلادرنگ. مانند ملاقات های تیمی آنلاین

۳ برنامه هایی که نیاز به ارسال نوتیفیکیشن (آلارم) از سمت سرور به سمت کلاینت دارند. مانند ایمیل، چت، بازی و ...

۴ واژه RPC سر آغاز عبارت Remote Procedure Call می باشد. این عبارت به مفهوم فراخوانی پروسیجر هایی از یک کامپیوتر توسط یک کامپیوتر دیگر اشاره دارد. (فراخوانی به صورت remote و نه local)

۵ این ارتباط به صورت مداوم (Persistence) می باشد. بر خلاف ارتباطات HTTP که پس از پردازش درخواست قطع می شوند.

۶ به مفهوم Broadcast

۷ سیگنال به صورت متن باز ارائه شده است و می توانید کدهای آن را در سایت گیت هاب به آدرس https://github.com/aspnet/signalr مشاهده نمایید.

روش اول استفاده از پیام های متنی به شکل JSON [1]

روش دوم استفاده از Message Pack [2]

پیاده سازی یک برنامه چت ساده

پس از مقدمه ای کوتاه در خصوص مفهوم سیگنال و برنامه های مستعد پیاده سازی با این مفهوم، وقت آن رسیده تا با پیاده سازی آن در یک برنامه ساده چت، روند عملیاتی آن را مشاهده کنیم.

ابزار های مورد نیاز

- ابزار NET Core SDK نسخه ۲,۱ یا بالاتر
- نسخه ۱۵,۷,۳ یالاتر از مجموعه Visual Studio 2017 (که بخش پیاده سازی برنامه های تحت وب آن نصب شده باشد)
- بسته مدیریت کتابخانه های Node به نام npm[3]

ایجاد پروژه

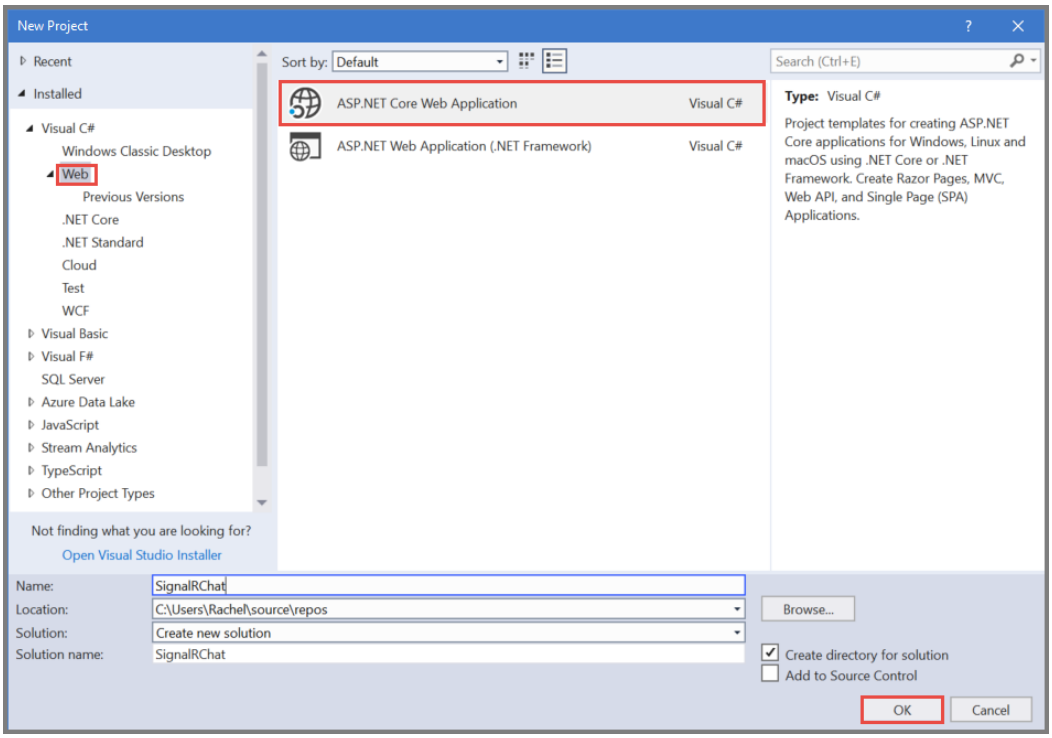
محیط ویژوال استودیو را باز کرده و یک پروژه تحت وب در بستر دات نت کر ایجاد کنید:

---

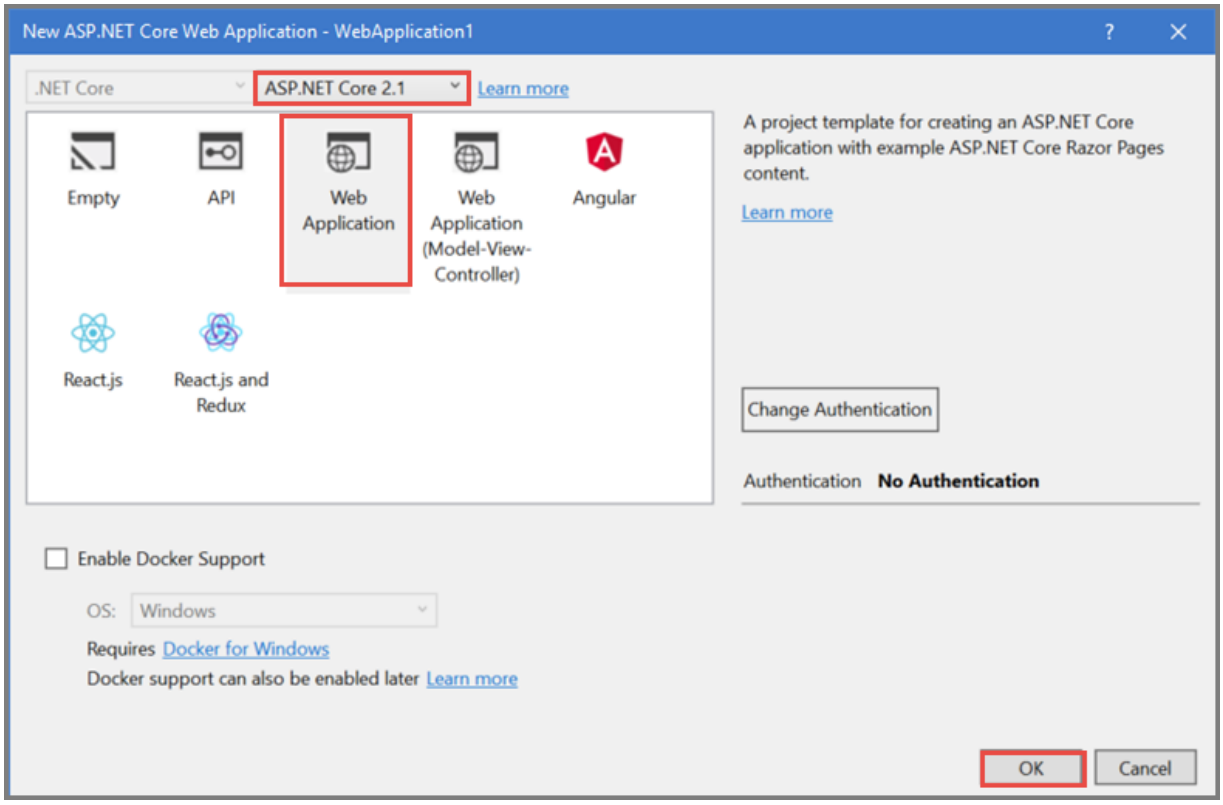
[1] واژه JSON سرآغاز عبارت JavaScript Object Notation می باشد. نگاه کنید به https://www.json.org/

[2] کتابخانه Message Pack شبیه JSON بوده با این تفاوت که نسبت به آن فشرده تر است و بنابر این با سرعت بیشتری در شبکه منتقل می شود. نگاه کنید به https://msgpack.org/

[3] بسته npm یا Node Package Manager که ابزار مدیریت بسته برای کتابخانه های جاوا اسکریپت می باشد. همانند Nuget که جهت بسته های دات نت به کار می رود. نگاه کنید به https://www.npmjs.com/get-npm



پس از کلیک بر رو دکمه Ok در صفحه بعدی گزینه Web Application را انتخاب کنید. دقت کنید که گزینه ASP.NET Core 2.1 انتخاب شده باشد.



پس از کلیک کردن بر روی دکمه OK منتظر بمانید تا پروژه ایجاد و بسته های مورد نیاز آن نصب گردند.

بسته های مورد نیاز کار با سیگنال برای سمت سرور[1] به صورت خودکار نصب خواهند شد اما لازم است تا با استفاده از npm اقدام به نصب بسته های سمت کلاینت نمائید.

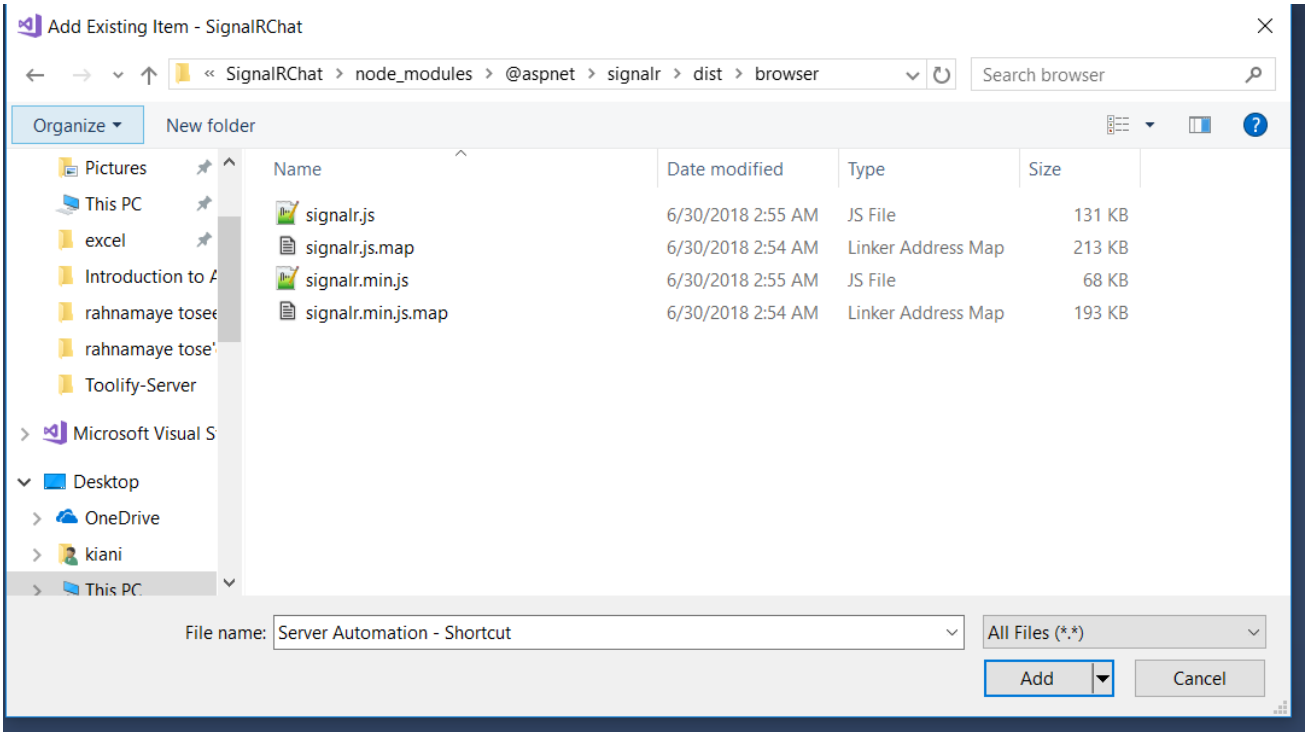
برای این منظور در پنجره کنسول مدیریت بسته ها[2] دستورات زیر را اجرا کنید :

```
npm init -y
npm install @aspnet/signalr
```

پس از دستورات فوق، پوشه ای به نام node_modules در مسیر پروژه ایجاد خواهد شد.

یک پوشه به نام signalr در پوشه lib واقع در پوشه wwwroot در پروژه ایجاد کنید. سپس بر روی آن راست کلیک کرده واز منوی Add منوی ..Existring Item را انتخاب کنید.

به مسیر پوشه node_modules که در شکل زیر نشان داده شده رفته و فایل signalr.js را به پروژه اضافه کنید.



[1] بسته Microsoft.AspNetCore.SignalR

[2] این پنجره را می توانید از مسیر Tools / Nuget Package Manager / Package Manager Console در ویژوال استودیو باز کنید.

ایجاد هاب

در این مرحله لازم است تا یک کلاس به پروژه اضافه کرده و آن را از کلاس Hub مشتق نمائید. نام کلاس را ChatHub و دستورات آن را به صورت زیر تنظیم نمائید :

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

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

دقت نمائید که کلاس Hub در فضای نام **Microsoft.AspNetCore.SignalR** قرار دارد. همچنین برای استفاده از کلاس Task جهت عملیات آسنکرون نیاز به فضای نام **System.Threading.Tasks** می باشد. پس مطمئن شوید دو فضای نام مذکور در کلاس اضافه شده باشند.

تنظیمات سمت سرور

فایل Startup.cs موجود در پروژه را باز کنید و آن را به صورت زیر تغییر دهید. دستورات هایلایت شده را به آن اضافه کنید:

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;

namespace SignalRChat
{
```

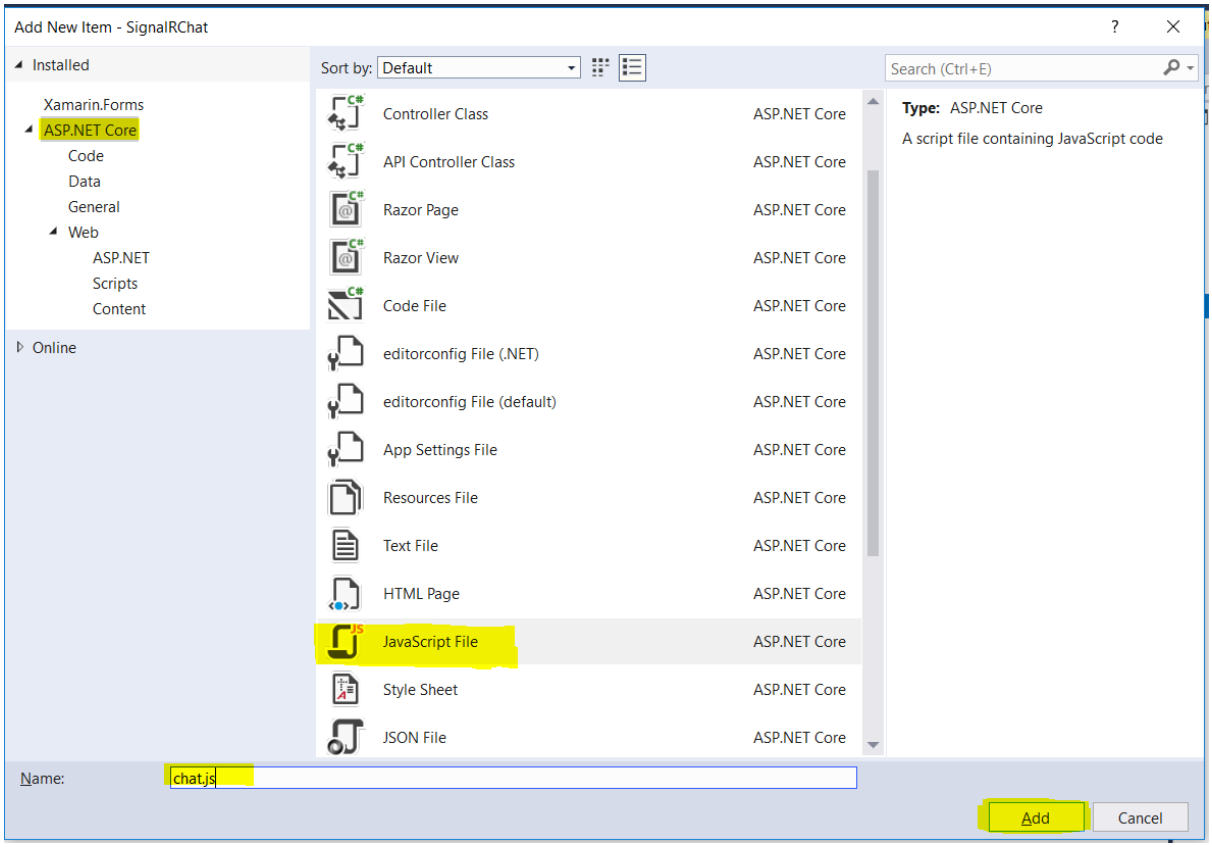
```
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public    void    ConfigureServices(IServiceCollection
services)
        {
            services.Configure<CookiePolicyOptions>(options =>
            {
                options.CheckConsentNeeded = context => true;
                options.MinimumSameSitePolicy                =
SameSiteMode.None;
            });

            services.AddSignalR();


services.AddMvc().SetCompatibilityVersion(CompatibilityVersion
.Version_2_1);
        }

        public    void    Configure(IApplicationBuilder    app,
IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Error");
            }

            app.UseStaticFiles();
            app.UseCookiePolicy();
            app.UseSignalR(routes =>
            {
                routes.MapHub<ChatHub>("/chatHub");
            });
```

```
            app.UseMvc();
        }
    }
}
```

دستورات هایلایت شده، کتابخانه سیگنال را برای پروژه فعال می کند. همچنین یک نگاشت برای بررسی درخواست های ارسالی به آدرس chatHub/ ایجاد شده است.

تنظیمات سمت کلاینت

در این مرحله می بایستی دستورات لازم برای سمت کلاینت را به صورت کد های جاوا اسکریپت ایجاد کنیم. برای این منظور یک فایل جاوااسکریپت به نام chat.js به پوشه js واقع در پوشه wwwroot اضافه کنید. کافیست بر روی نام پوشه js راست کلیک کرده و گزینه Add سپس New Item… را انتخاب نمائید:



حال دستورات زیر را در فایل chat.js تایپ کنید :

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

connection.on("ReceiveMessage", (user, message) => {
```

```
    const msg = message.replace(/&/g, "&amp;").replace(/</g,
"&lt;").replace(/>/g, "&gt;");
    const encodedMsg = user + " says " + msg;
    const li = document.createElement("li");
    li.textContent = encodedMsg;
    document.getElementById("messagesList").appendChild(li);
});

connection.start().catch(err                                    =>
console.error(err.toString()));

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

همانطور که مشاهده می کنید در خط اول ارتباط به آدرس chatHub/ تعریف شده است. این آدرس پیش از این در بخش کانفینگ های برنامه سمت سرور نیز نگاشت شده بود. همچنین از رویداد ReceiveMessage برای مدیریت پیام های دریافتی و از رویداد کلیک دکمه snedButton برای راسال پیام توسط متد SendMessage استفاده شده است.

تغییر ویو

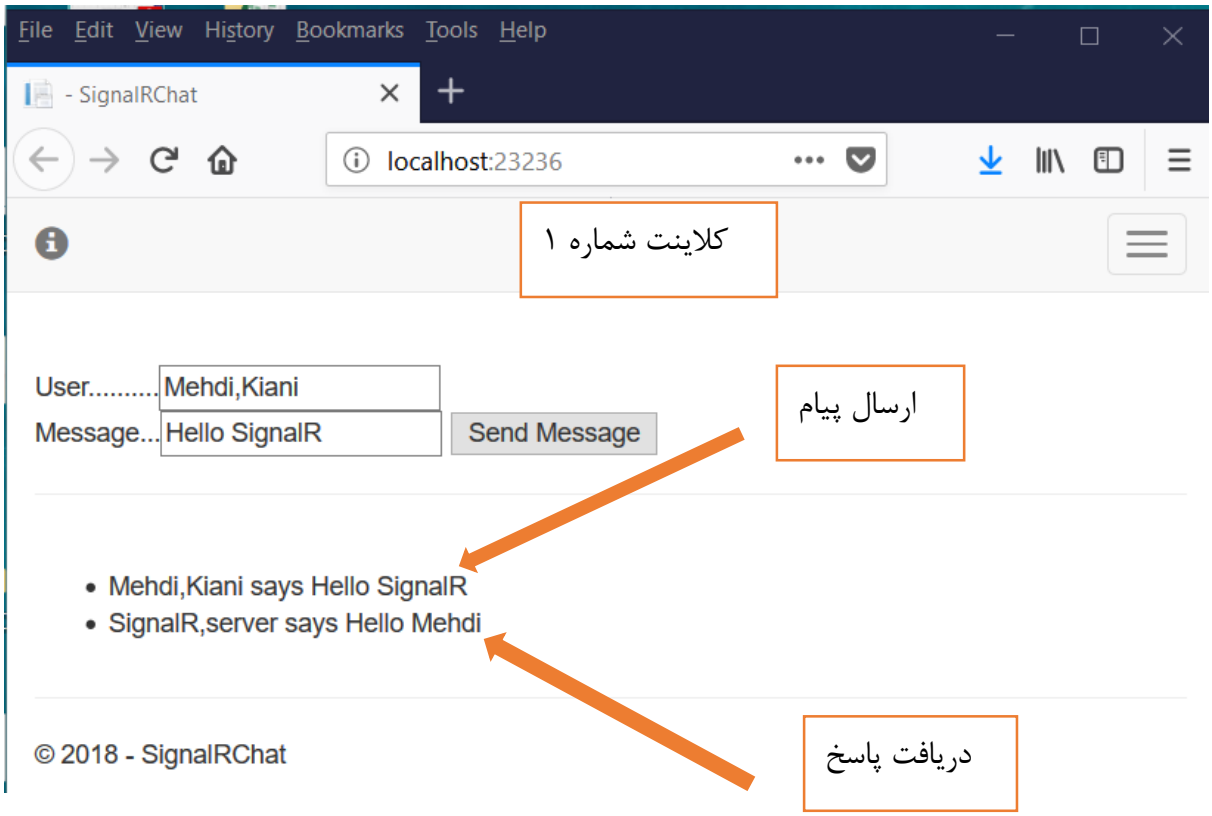
در مرحله آخر کافیست تا دستورات فایل Index.cshhtml را که در پوشه Pages قرار دارد به صورت زیر تغییر دهید :

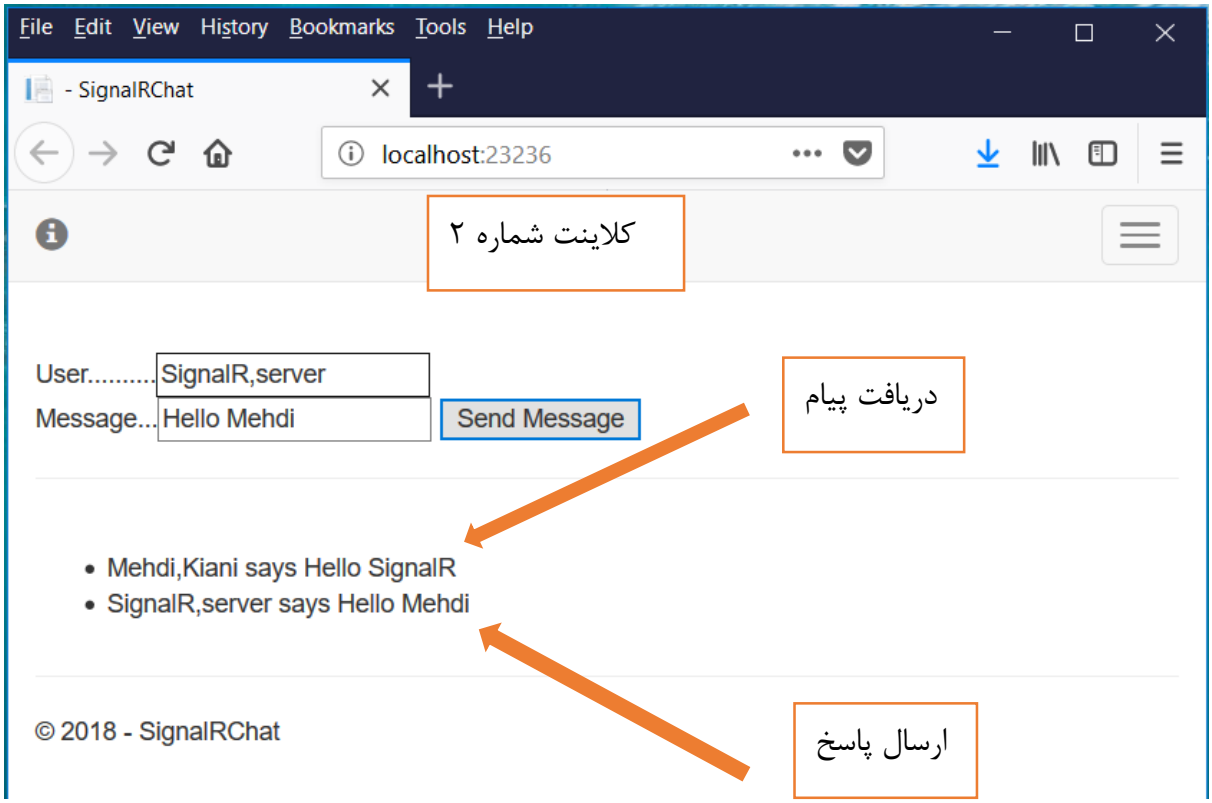
```
@page
    <div class="container">
        <div class="row"> </div>
        <div class="row">
            <div class="col-6"> </div>
            <div class="col-6">
                User.........<input          type="text"
id="userInput" />
                <br />
                Message...<input             type="text"
id="messageInput" />
```

```
                <input    type="button"    id="sendButton"
value="Send Message" />
            </div>
        </div>
        <div class="row">
            <div class="col-12">
                <hr />
            </div>
        </div>
        <div class="row">
            <div class="col-6"> </div>
            <div class="col-6">
                <ul id="messagesList"></ul>
            </div>
        </div>
    </div>
    <script src="~/lib/signalr/signalr.js"></script>
    <script src="~/js/chat.js"></script>
```

اجرای برنامه

در این مرحله می توانید برنامه را اجرا کنید و خروجی را مشاهده نمائید :





پایان

منبع

https://docs.microsoft.com/en-us/aspnet/core/signalr/?view=aspnetcore-2.1

سورس کامل پروژه

https://github.com/aspnet/Docs/tree/master/aspnetcore/tutorials/signalr/sample

ارتباط با من

Home: http://mkiani.ir

Email: mkiani3000@gmail.com