Send messages from outside a hub

11/12/2019 • 2 minutes to read • +7

In this article

By Mikael Mengistu

The SignalR hub is the core abstraction for sending messages to clients connected to the SignalR server. It's also possible to send messages from other places in your app using the `IHubContext` service. This article explains how to access a SignalR `IHubContext` to send notifications to clients from outside a hub.

View or download sample code (how to download)

Get an instance of IHubContext

In ASP.NET Core SignalR, you can access an instance of `IHubContext` via dependency injection. You can inject an instance of `IHubContext` into a controller, middleware, or other DI service. Use the instance to send messages to clients.

> **Note**
>
> This differs from ASP.NET 4.x SignalR which used GlobalHost to provide access to the `IHubContext`. ASP.NET Core has a dependency injection framework that removes the need for this global singleton.

Inject an instance of IHubContext in a controller

You can inject an instance of `IHubContext` into a controller by adding it to your constructor:

C# Copy

```
public class HomeController : Controller
{
    private readonly IHubContext<NotificationHub> _hubContext;

    public HomeController(IHubContext<NotificationHub> hubContext)
```

```
    {
        _hubContext = hubContext;
    }
}
```

Now, with access to an instance of `IHubContext`, you can call hub methods as if you were in the hub itself.

C# Copy

```
public async Task<IActionResult> Index()
{
    await _hubContext.Clients.All.SendAsync("Notify", $"Home page loaded at:
{DateTime.Now}");
    return View();
}
```

Get an instance of IHubContext in middleware

Access the `IHubContext` within the middleware pipeline like so:

C# Copy

```
app.Use(async (context, next) =>
{
    var hubContext = context.RequestServices
                            .GetRequiredService<IHubContext<ChatHub>>();
    //...

    if (next != null)
    {
        await next.Invoke();
    }
});
```

> **Note**
>
> When hub methods are called from outside of the `Hub` class, there's no caller associated with the invocation. Therefore, there's no access to the `ConnectionId`, `Caller`, and `Others` properties.

Get an instance of IHubContext from IHost

Accessing an `IHubContext` from the web host is useful for integrating with areas outside of ASP.NET Core, for example, using third-party dependency injection frameworks:

C# Copy

```
    public class Program
    {
        public static void Main(string[] args)
        {
            var host = CreateHostBuilder(args).Build();
            var hubContext =
host.Services.GetService(typeof(IHubContext<ChatHub>));
            host.Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder => {
                    webBuilder.UseStartup<Startup>();
                });
    }
```

Inject a strongly-typed HubContext

To inject a strongly-typed HubContext, ensure your Hub inherits from `Hub<T>`. Inject it using the `IHubContext<THub, T>` interface rather than `IHubContext<THub>`.

C# Copy

```
public class ChatController : Controller
{
    public IHubContext<ChatHub, IChatClient> _strongChatHubContext { get; }

    public ChatController(IHubContext<ChatHub, IChatClient> chatHubContext)
    {
        _strongChatHubContext = chatHubContext;
    }

    public async Task SendMessage(string user, string message)
    {
        await _strongChatHubContext.Clients.All.ReceiveMessage(user, mes-
sage);
    }
}
```

See Strongly typed hubs for more information.

Related resources

- Get started
- Hubs
- Publish to Azure

Is this page helpful?

Yes No