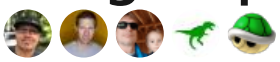
Manage users and groups in SignalR

05/17/2020 • 2 minutes to read •  +2

In this article

By Brennan Conroy

SignalR allows messages to be sent to all connections associated with a specific user, as well as to named groups of connections.

View or download sample code (how to download)

Users in SignalR

A single user in SignalR can have multiple connections to an app. For example, a user could be connected on their desktop as well as their phone. Each device has a separate SignalR connection, but they're all associated with the same user. If a message is sent to the user, all of the connections associated with that user receive the message. The user identifier for a connection can be accessed by the `Context.UserIdentifier` property in the hub.

By default, SignalR uses the `ClaimTypes.NameIdentifier` from the `ClaimsPrincipal` associated with the connection as the user identifier. To customize this behavior, see Use claims to customize identity handling.

Send a message to a specific user by passing the user identifier to the `User` function in a hub method, as shown in the following example:

> **Note**
>
> The user identifier is case-sensitive.

C# Copy

```
public Task SendPrivateMessage(string user, string message)
{
    return Clients.User(user).SendAsync("ReceiveMessage", message);
}
```

Groups in SignalR

A group is a collection of connections associated with a name. Messages can be sent to all connections in a group. Groups are the recommended way to send to a connection or multiple connections because the groups are managed by the application. A connection can be a member of multiple groups. Groups are ideal for something like a chat application, where each room can be represented as a group. Connections are added to or removed from groups via the `AddToGroupAsync` and `RemoveFromGroupAsync` methods.

C# Copy

```
public async Task AddToGroup(string groupName)
{
    await Groups.AddToGroupAsync(Context.ConnectionId, groupName);

    await Clients.Group(groupName).SendAsync("Send", $"
{Context.ConnectionId} has joined the group {groupName}.");
}

public async Task RemoveFromGroup(string groupName)
{
    await Groups.RemoveFromGroupAsync(Context.ConnectionId, groupName);

    await Clients.Group(groupName).SendAsync("Send", $"
{Context.ConnectionId} has left the group {groupName}.");
}
```

Group membership isn't preserved when a connection reconnects. The connection needs to rejoin the group when it's re-established. It's not possible to count the members of a group, since this information is not available if the application is scaled to multiple servers.

To protect access to resources while using groups, use authentication and authorization functionality in ASP.NET Core. If a user is added to a group only when the credentials are valid for that group, messages sent to that group will only go to authorized users. However, groups are not a security feature. Authentication claims have features that groups do not, such as expiry and revocation. If a user's permission to access the group is revoked, the app must remove the user from the group explicitly.

> **Note**
>
> Group names are case-sensitive.

Related resources

- Get started
- Hubs
- Publish to Azure

Is this page helpful?

Yes No