Difference between SendAsync and SendCoreAsync methods in SignalR Core?

8

When updating to the latest version of ASP Net Core and SignalR core, I noticed their are two "send" methods available when sending methods to a client (what used to be InvokeAsync).

After looking at the code comments, both methods are identical in comments, both inherit from IClientProxy, and both accept a string method, object args and then a cancellation token.

What are the differences in these methods? If any? and which should be used when?

asp.net-core-signalr

asked Jul 30 '18 at 9:00

FasettoAndy

43 3

Question is inaccurate, the arguments they receive are not the same. – Andrew Jun 6 at 16:52

1 Answer

10

Quoting `@anurse` at https://github.com/aspnet/SignalR/issues/2239#issuecomment-387854470:

The short answer is: The Core methods should be ignored unless you really know what you're doing.

For some more background:

We started with SendAsync, which takes an array of arguments to send:

```
public void SendAsync(string method, object[] args);
```

public void SendAsync(string method, params object[] args);

```
Clients.All.SendAsync("Method", arg1, arg2, arg3);
```

However, that falls apart when you actually want to send an array as a single argument

```
public void SendAsync(string method, params object[] args);

var arg1 = new object[] { a, b, c };

Clients.All.SendAsync("Method", arg1);

// C# 'params' expands this into the below

Clients.All.SendAsync("Method", a, b, c);
```

So instead of sending a single argument that is an array a, b, c, we've sent each of those as separate arguments. This was confusing users.

So we removed the params from it and instead we generate a whole bunch of extension methods that support multiple arguments:

```
public void SendAsync(string method, object[] args);
public void SendAsync(string method, object arg1) => SendAsync(method, new object[] {
arg1 });
public void SendAsync(string method, object arg1, object arg2) => SendAsync(method, new
object[] { arg1, arg2 });
// ... etc ...
```

But there's still ambiguity when you have code like this:

```
public void SendAsync(string method, object[] args);
public void SendAsync(string method, object arg1) => SendAsync(method, new object[] {
arg1 });

var arg = new object[] { a, b, c }

Clients.All.SendAsync("method", arg);
```

```
public void SendCoreAsync(string method, object[] args);
public void SendAsync(string method, object arg1) => SendCoreAsync(method, new object[]
{ arg1 });

var arg = new object[] { a, b, c }

// No ambiguity here!
Clients.All.SendAsync("method", arg);
```

answered Jul 30 '18 at 9:06

emix
7,671 5 34 55