
Unity®
Supercharging your games with Multiplayer Building Blocks
2026
Blocks, blocks everywhere

>whoami

Nice to meet you!

Kálmán Bálint

Senior Software Developer

What are we going to talk about

- → Multiplayer Online Services
- → Multiplayer Services SDK
- → Building Blocks
- → UIToolkit
- → MVVM & MVC

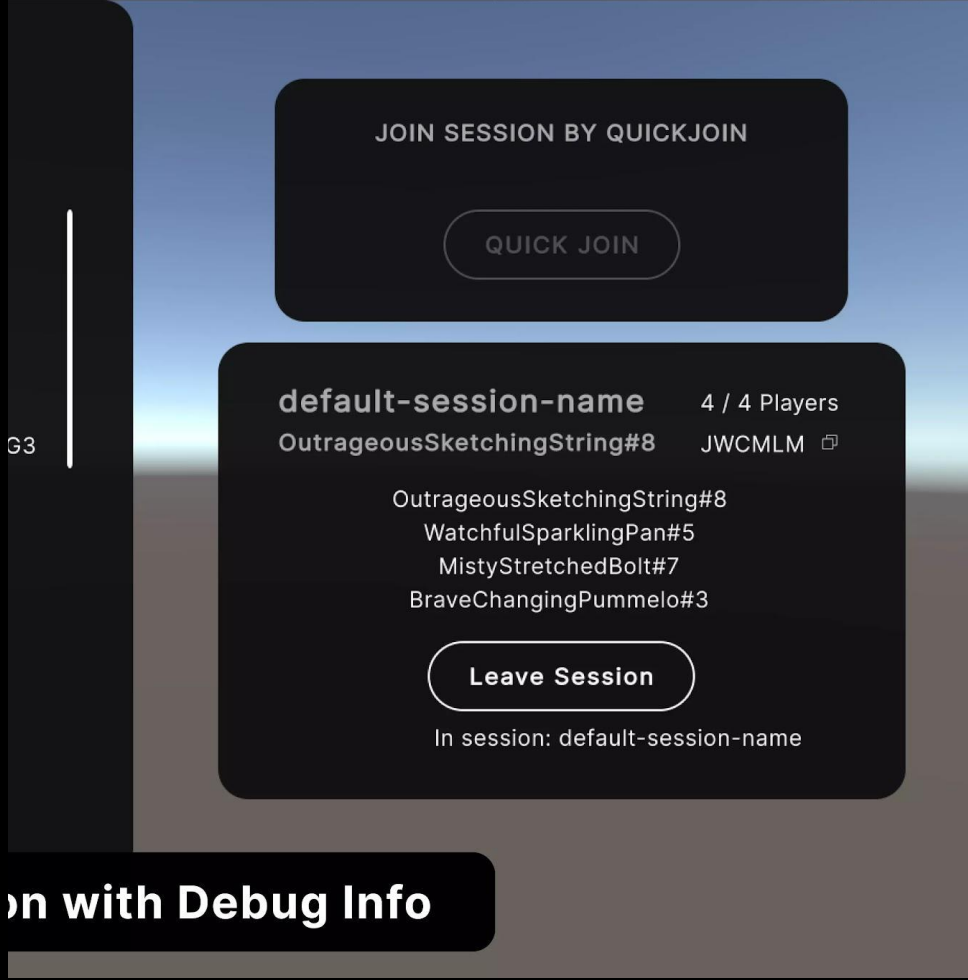
What we are NOT going to talk about

- → How to choose the best netcode solution for your project
- → Deep dive into network topologies
- → In detail billing information
- → UGUI / Multiplayer Widgets
- → 3rd party netcode integrations and 3rd party online services / hosting
- → Design patterns other than MVVM / MVC

Project Setup

What do we need to start?

- → 6000.3 LTS (preferably)
- → Multiplayer Services SDK
- → Multiplayer Center
  - – Install Quick starts
- → Multiplayer Session Block
  - – Join by browsing
- → Netcode for GameObjects
- → MPPM
  - – Scenarios*



Let’s see what we’ve got now.

- → Browse Sessions
- → Set up a Multiplayer Scenario to spawn an additional player for testing.
- → Session Viewer != Session Browser
- → Unity Gaming Services have a generous free tier

How can I add a networked player using this?

- → Netcode Setup
- → Create a Player prefab
- → Add a script to control the prefab
- → Are we done?

Simple Player Controller

```
using UnityEngine;
using UnityEngine.InputSystem;

[RequireComponent(typeof(CharacterController))]
public class SimplePlayerController : MonoBehaviour
{
    [SerializeField] private float _speed = 5f;
    [SerializeField] private InputActionAsset _moveAction;
    private CharacterController _controller;

    private void Start()
    {
        _controller = GetComponent<CharacterController>();
        _moveAction.Enable();
    }

    void Update()
    {
        var moveDirection = _moveAction["Move"].ReadValue<Vector2>();

        // Apply movement to the CharacterController
        // Time.deltaTime makes the movement frame-rate independent
        _controller.Move(new Vector3(moveDirection.x, 0, moveDirection.y) * (_speed * Time.deltaTime));
    }
}
```

Let's dive in deeper!

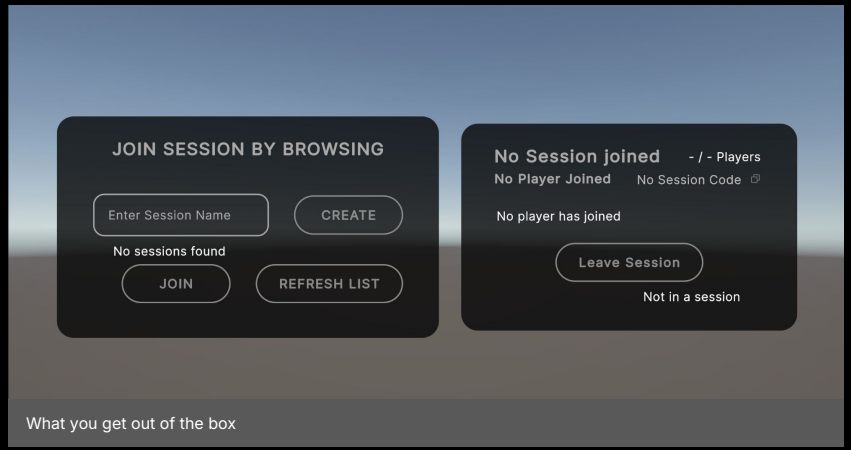
Why doesn't it work yet !?

- → Is the Session actually linked to the network manager instance?
- → Have we made sure that only the right people can control the players?

What if I don't like how the UI looks?

- → Modify it!
- → UIToolkit is designed to be easily modifiable
- → MVVM enables your team to work more easily in parallel

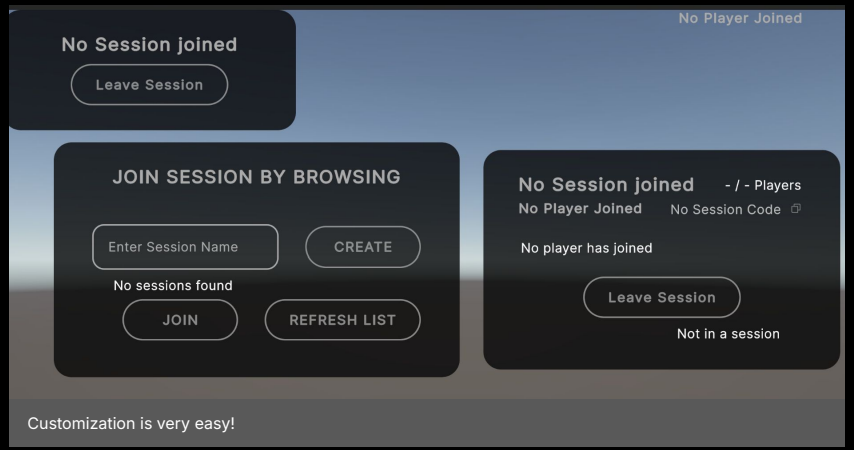
Let's Change the UI Layout



What you get out of the box

Before

For now we've been using the default Building Block UI.
Let's add another one and arrange some of the on-screen Session specific information and actions in a reusable fashion using the UI Builder and already existing scriptable objects.



Customization is very easy!

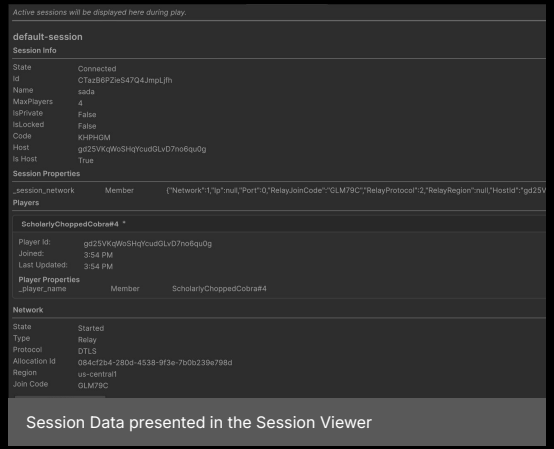
After

We might end up with a UI something like the above, let's see how fast we can get there without having to add any code*.

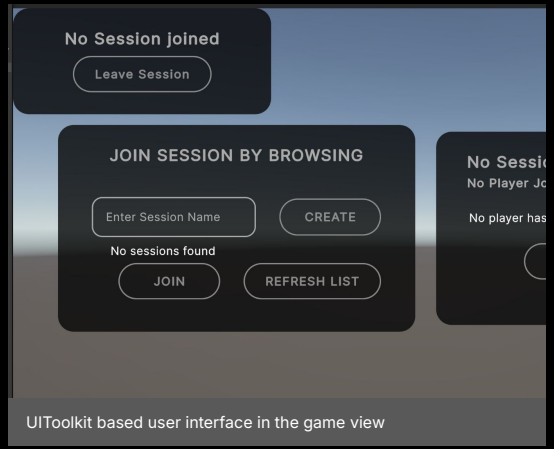
Change the UI layout when entering and leaving the Session

```
using UnityEngine;

public class UILayoutController : MonoBehaviour
{
    [SerializeField] private UnityEngine.UIElements.UIDocument _enterSessionUI;
    [SerializeField] private UnityEngine.UIElements.UIDocument _gameplayUI;
    [SerializeField] private Unity.Netcode.NetworkManager _networkManager;

    private void Start()
    {
        _networkManager.OnClientStarted += OnClientStarted; _networkManager.OnClientStopped += OnClientStopped;
        OnClientStopped(); //Reset state on startup
    }

    private void OnClientStopped(bool wasTheHost = false)
    {
        _enterSessionUI.enabled = true; _gameplayUI.enabled = false;
    }

    private void OnClientStarted() { _enterSessionUI.enabled = false; _gameplayUI.enabled = true;  }

    private void OnDestroy()
    {
        _networkManager.OnClientStarted -= OnClientStarted; _networkManager.OnClientStopped -= OnClientStopped;
    }
}
```

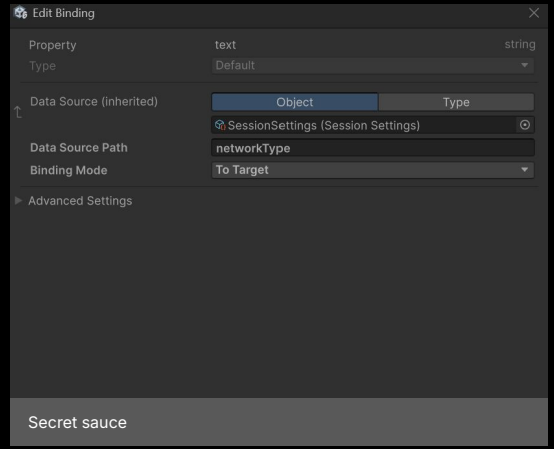
Okay but how does this actually work?



Session Data presented in the Session Viewer



UIToolkit based user interface in the game view



Secret sauce

Session

You may think of a session as a shared data structure that represents the multiplayer state for all connected devices; it doesn't require any built-in UI.

Runtime User Interface

We have seen how we can represent a subset of the Session specific information at runtime. But is that all?

Binding system (The Glue)

Let's talk a bit more about the binding system. Why would you use it, and how.

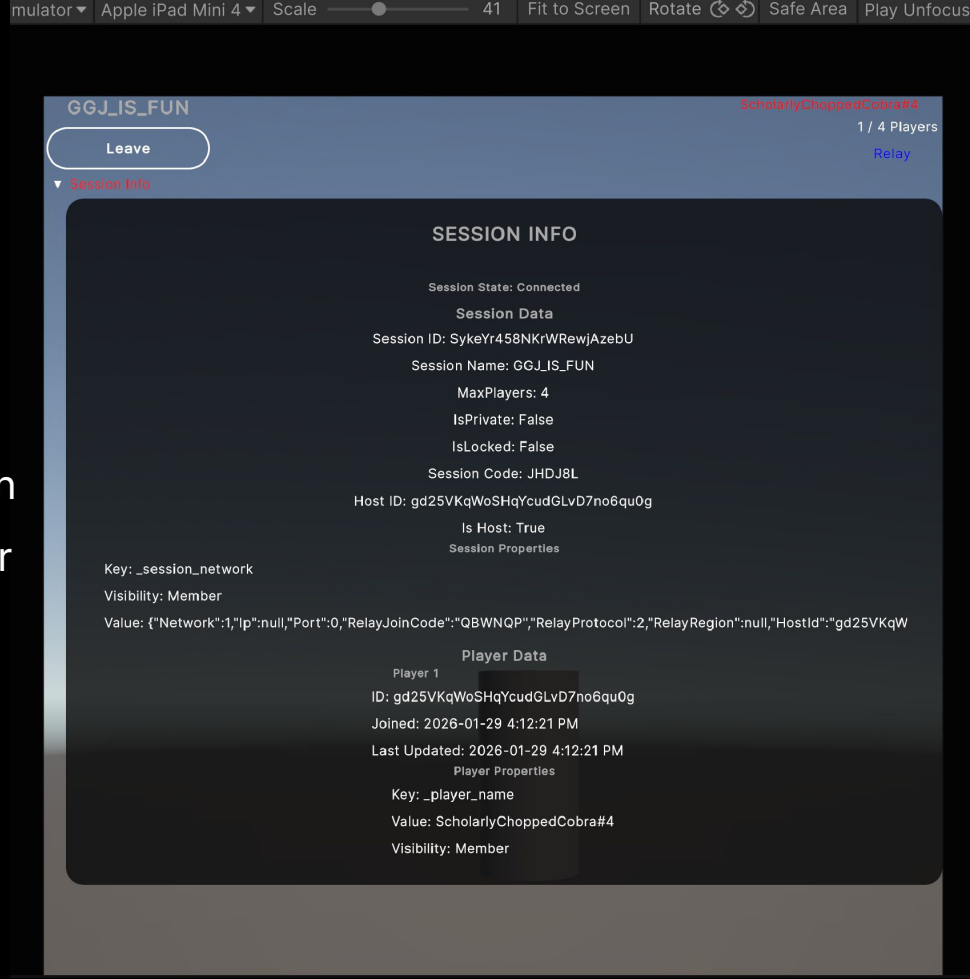
MVVM / MVC 🤨?

- → Decuple logic and presentation layer
- → Independent development of UI and code
- → How we use it for Blocks
- → How would this help you?

```
namespace Blocks.Sessions.Common
{
    3 usages
    class LeaveSessionViewModel : IDisposable, IDataSourceViewHashProvider, INotifyBindablePropertyChanged
    {
        SessionObserver m_SessionObserver;
        ISession m_Session;

        [CreateProperty]
        3 usages
        public bool CanLeaveSession{...}

        bool m_CanLeaveSession;

        1 usage
        public LeaveSessionViewModel(string sessionType){...}

        2 usages
        void OnSessionAdded(ISession newSession){...}

        4 usages
        void OnSessionRemoved(){...}

        1 usage
        public async Task LeaveSession(string sessionType){...}

        2 usages
        void CleanupSession(){...}

        public void Dispose(){...}

        /// <summary>
        /// This method is used by UIToolkit to determine if any data bound to the UI has changed.
        /// Instead of hashing the data, an m_CanLeaveSession boolean is toggled when changes occur.
        /// </summary>
        public long GetViewHashCode() => m_CanLeaveSession ? 1 : 0;

        /// <summary>
        /// Suggested implementation of INotifyBindablePropertyChanged from UIToolkit.
        /// </summary>
        public event EventHandler<BindablePropertyChangedEventArgs> propertyChanged;

        1 usage
        void Notify([CallerMemberName] string property = null){...}
    }
}
```

Okay but, how do you access the properties of a Session and not the properties of the settings?

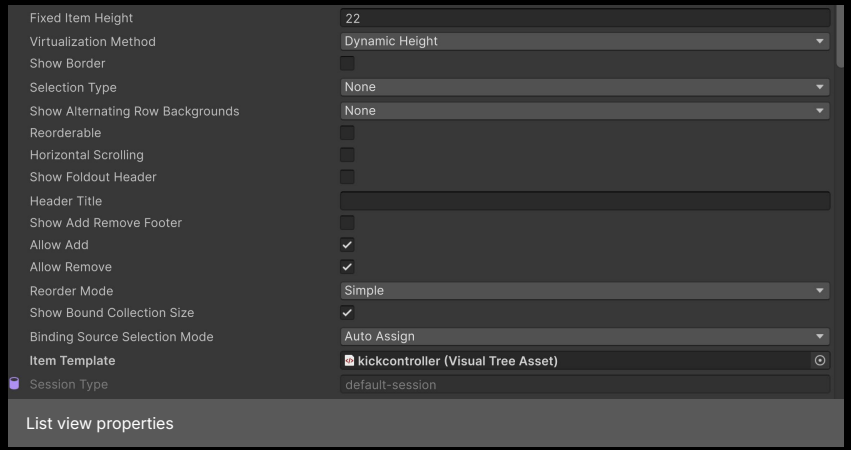
- → Session Info
- → What properties are exposed on a Session
- → How does this compare to Session Viewer
- → How to add custom behavior using the Session properties?



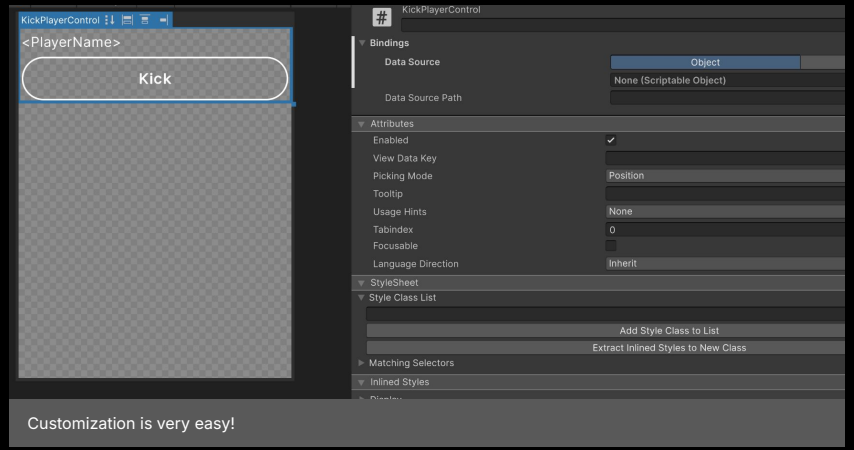
Add a custom control to kick players out

```
    [UxmlElement]
1   public partial class KickPlayerControl : VisualElement
2   {
3       PlayerViewModel m_ViewModel => dataSource as PlayerViewModel;
4
5       public KickPlayerControl()
6       {
7           RegisterCallback<AttachToPanelEvent>(OnAttachToPanel);
8       }
9
10      private void OnAttachToPanel(AttachToPanelEvent evt)
11      {
12          SetBinding(nameof(dataSource), new DataBinding { dataSourceType = typeof(PlayerViewModel) });
13
14          var kickButton = new Button(() =>  m_ViewModel.Session.AsHost()
15                                                          .RemovePlayerAsync(m_ViewModel.Id)) { text = "Kick" };
16          kickButton.AddToClassList(BlocksTheme.Button);
17
18          var playerNameLabel = new Label("<PlayerName>");
19          playerNameLabel.AddToClassList(BlocksTheme.Label);
20          playerNameLabel.SetBinding(nameof(Label.text), new DataBinding
21          {
22              dataSourcePath = PropertyPath.FromName(nameof(PlayerViewModel.Name))
23          });
24
25          hierarchy.Add(playerNameLabel);
            hierarchy.Add(kickButton);
        }
    }
```

Add new control as item template



List view properties



Customization is very easy!

Player List View Item Template

The multiplayer building blocks come with a list control that binds all players within a session to it's items. The template UI items will receive a PlayerViewModel as datasource.

Template UIDocument

Assigning a template, that hosts our control we just created we'll enable us to extend the behavior of the base control while keeping the integration with the existing styling.

Unity®

Important Session API calls

```
var sessionType = "default-session";
// access the session locally by its type (local ID)
var session = MultiplayerService.Instance.Sessions[sessionType];
// options used to configure the session
var sessionOptions = new SessionOptions{Type=sessionType, SessionProperties=new Dictionary<string, SessionProperty>()};
MultiplayerService.Instance.CreateSessionAsync(sessionOptions);
MultiplayerService.Instance.JoinSessionByIdAsync(session.Id);
// Session lifetime events
session.Changed += () => { }; session.Deleted += ()=>{ } ;
// Sessions report player connection events, note that the id is the service ID not the netcode client ID
// in case you want to map between the two, you may need to set up an approval callback on the netcode manager.
// see NGO docs for more information.
session.PlayerJoined += id => { }; session.PlayerHasLeft += id => { }; session.PlayerLeaving += id => { };
session.RemovedFromSession += ( ) => { }; // current player

// ISession.Properties gives access to the properties of a session that are synchronized between all participants
Debug.Log(string.Join('\n', session.Properties.Keys ));

// provides easy access to lifetime callbacks of a session, useful for MVVM
var sessionObserver = new SessionObserver(sessionType);

sessionObserver.SessionAdded += _ => { }; // Session has been added locally
sessionObserver.AddingSessionStarted += _ => { }; // Session Create / Join was called locally

// Session Create / Join failed, check the exception for more detail
sessionObserver.AddingSessionFailed += (o,e) => { };
```

Unity®

Thank you

2026

Good luck Jamming!


Unity®

Resources

- → https://docs.unity.com/en-us/mps-sdk
  https://docs.unity.com/en-us/mps-sdk/faq
- → https://docs.unity3d.com/6000.3/Documentation/Manual/ui-systems/introduction-ui-toolkit.html
- → https://unity.com/products/gaming-services/pricing
- → https://cloud.unity.com/
- → https://docs.unity3d.com/Packages/com.unity.netcode.gameobjects@2.8/manual/components/core/corecomponents.html
- → https://docs.unity3d.com/Packages/com.unity.services.multiplayer@2.0/api/Unity.Services.Multiplayer.SessionObserver.html

These tests are potentially lethal when communication, teamwork, and mutual respect are not employed at all times.

— GLaDOS (Portal 2)