



Supercharging your games with Multiplayer Building Blocks

2026

Blocks, blocks everywhere



Supercharging your games with
Multiplayer Building Blocks

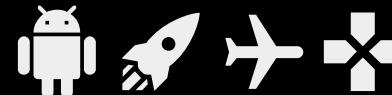
>whoami

Nice to meet you!



Kálmán Bálint

Senior Software Developer



Unity®



What are we going to talk about

- Multiplayer Online Services
- Multiplayer Services SDK
- Building Blocks
- UI Toolkit
- 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*

G3

JOIN SESSION BY QUICKJOIN

QUICK JOIN

default-session-name

4 / 4 Players

OutrageousSketchingString#8

JWCMLM

OutrageousSketchingString#8

WatchfulSparklingPan#5

MistyStretchedBolt#7

BraveChangingPummelo#3

Leave Session

In session: default-session-name

on with Debug Info



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

```
1  using UnityEngine;
2  using UnityEngine.InputSystem;
3
4  [RequireComponent(typeof(CharacterController))]
5  public class SimplePlayerController : MonoBehaviour
6  {
7      [SerializeField] private float _speed = 5f;
8      [SerializeField] private InputActionAsset _moveAction;
9      private CharacterController _controller;
10
11     private void Start()
12     {
13         _controller = GetComponent<CharacterController>();
14         _moveAction.Enable();
15     }
16
17     void Update()
18     {
19         var moveDirection = _moveAction["Move"].ReadValue<Vector2>();
20
21         // Apply movement to the CharacterController
22         // Time.deltaTime makes the movement frame-rate independent
23         _controller.Move(new Vector3(moveDirection.x, 0, moveDirection.y) * (_speed * Time.deltaTime));
24     }
25 }
```



Let's dive in deeper!

Why doesn't it work yet !?

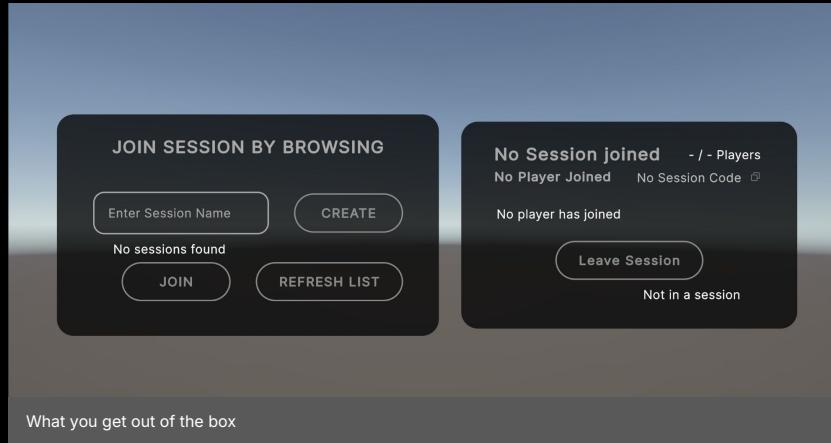
- Is the Session actually linked to the network manager instance?
- Have we made sure that only the right people can control the players?

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

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

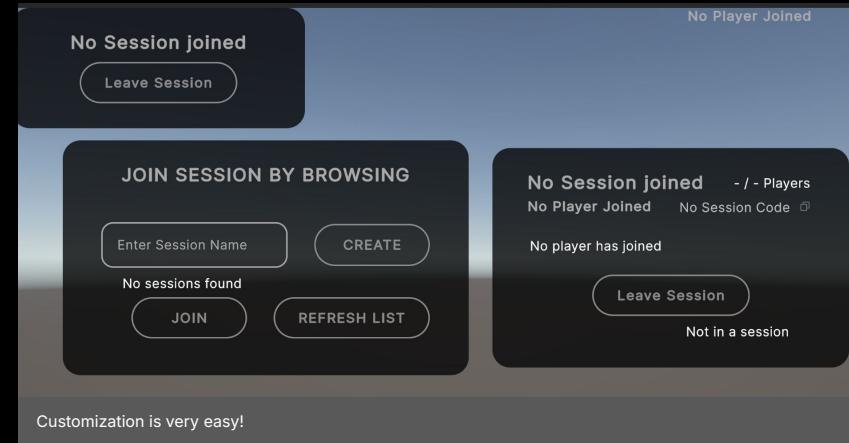


Let's Change the UI Layout



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.



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

```
1  using UnityEngine;
2
3  public class UILayoutController : MonoBehaviour
4  {
5      [SerializeField] private UnityEngine.UIElements.UIDocument _enterSessionUI;
6      [SerializeField] private UnityEngine.UIElements.UIDocument _gameplayUI;
7      [SerializeField] private Unity.Netcode.NetworkManager _networkManager;
8
9      private void Start()
10     {
11         _networkManager.OnClientStarted += OnClientStarted; _networkManager.OnClientStopped += OnClientStopped;
12         OnClientStopped(); //Reset state on startup
13     }
14
15     private void OnClientStopped(bool wasTheHost = false)
16     {
17         _enterSessionUI.enabled = true; _gameplayUI.enabled = false;
18     }
19
20     private void OnClientStarted() { _enterSessionUI.enabled = false; _gameplayUI.enabled = true; }
21
22     private void OnDestroy()
23     {
24         _networkManager.OnClientStarted -= OnClientStarted; _networkManager.OnClientStopped -= OnClientStopped;
25     }
}
```



Okay but how does this actually work?

Session Data presented in the Session Viewer

default-session	
Session Info	
State	Connected
Id	CTaz86PZie54704JmpLjh
Name	sada
MaxPlayers	4
IsPublic	False
IsLocked	False
Code	KHPGM
Host	g2SVKqWzSHqfcdGLx07ne6qu0g
Is Host	True
Session Properties	
_session._network	Member
Players	["Network":1,"ip":null,"Port":0,"RelayJoinCode":"GLM79C","RelayProtocol":2,"RelayRegion":null,"HostId":1,g2SVKqWzSHqfcdGLx07ne6qu0g}
ScholarlyChoppedCobra#4	*
Player Id	g2SVKqWzSHqfcdGLx07ne6qu0g
Joined	3:54 PM
Last Updated	3:54 PM
Player Properties	...Player Home Member ScholarlyChoppedCobra#4
Network	
State	Started
Type	Relay
Protocol	DTLS
Allocation Id	00000000-0000-0000-0000-000000000000
Region	us-central1
Join Code	GLM79C

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.

UIToolkit based user interface in the game view

No Session joined
Leave Session

JOIN SESSION BY BROWSING

Enter Session Name CREATE

No sessions found
JOIN REFRESH LIST

No Session joined
No Player Joined
No player has joined this session

Runtime User Interface

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

Edit Binding

Property: text
Type: string
Data Source (Inherited): Object
Data Source Path: SessionSettings (Session Settings)
Binding Mode: To Target

Advanced Settings

Secret sauce

Binding system (The Glue)

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



MVVM / MVC 😕?

- Decouple 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;

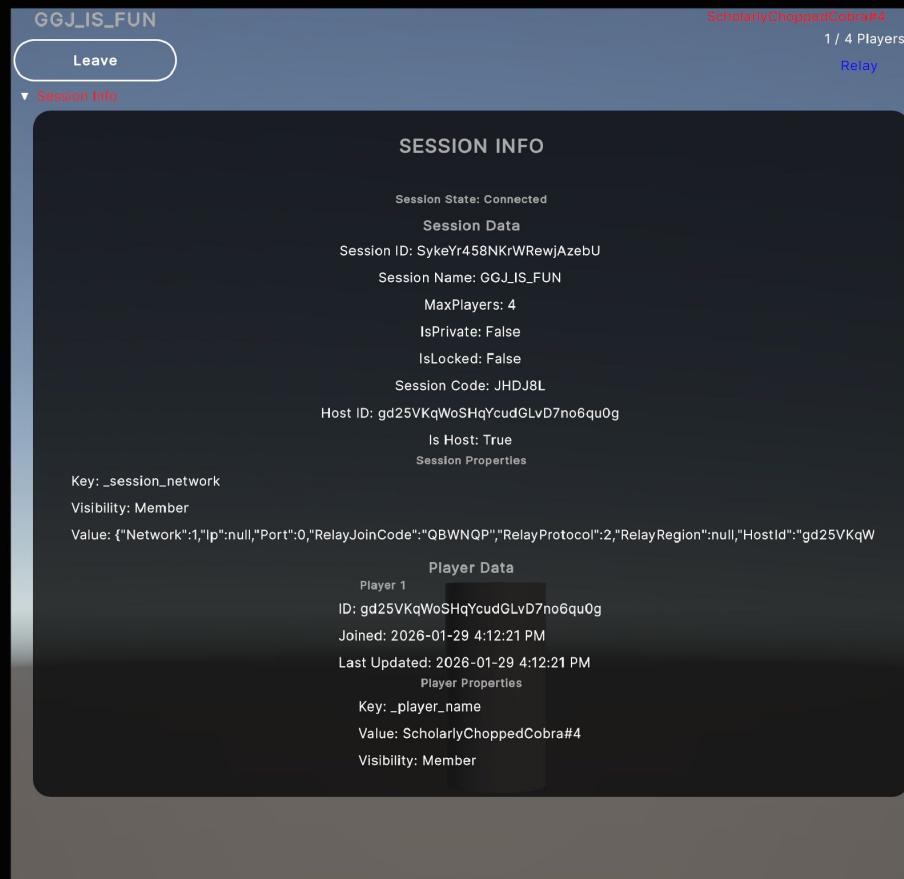
        [1 usage]
        void Notify([CallerMemberName] string property = null){...}
    }
}
```





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

```
1 [XmlElement]
2 public partial class KickPlayerControl : VisualElement
3 {
4     PlayerViewModel m_ViewModel => dataSource as PlayerViewModel;
5
6     public KickPlayerControl()
7     {
8         RegisterCallback<AttachToPanelEvent>(OnAttachToPanel);
9     }
10
11    private void OnAttachToPanel(AttachToPanelEvent evt)
12    {
13        SetBinding(nameof(dataSource), new DataBinding { dataSourceType = typeof(PlayerViewModel) });
14
15        var kickButton = new Button(() => m_ViewModel.Session.AsHost()
16                                  .RemovePlayerAsync(m_ViewModel.Id)) { text = "Kick" };
16        kickButton.AddToList(BlocksTheme.Button);
17
18        var playerNameLabel = new Label("<PlayerName>");
19        playerNameLabel.AddToList(BlocksTheme.Label);
20        playerNameLabel.SetBinding(nameof(Label.text), new DataBinding
21        {
22            dataSourcePath = PropertyPath.FromName(nameof(PlayerViewModel.Name))
23        });
24
25        hierarchy.Add(playerNameLabel);
26        hierarchy.Add(kickButton);
27    }
}
```



Add new control as item template

The screenshot shows the Unity Inspector for a 'List View Item Template'. The configuration includes:

- Fixed Item Height: 22
- Virtualization Method: Dynamic Height
- Show Border: None
- Selection Type: None
- Show Alternating Row Backgrounds: None
- Reorderable: None
- Horizontal Scrolling: None
- Show Foldout Header: None
- Header Title: None
- Show Add Remove Footer: None
- Allow Add: checked
- Allow Remove: checked
- Reorder Mode: Simple
- Show Bound Collection Size: None
- Binding Source Selection Mode: Auto Assign
- Item Template: kickcontroller (Visual Tree Asset)
- Session Type: default-session

Player List View Item Template

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

The screenshot shows a 'KickPlayerControl' component assigned to a UI element in a 'UIDocument'. The component configuration includes:

- Bindings: Data Source (Object) set to 'None (Scriptable Object)'.
- Attributes: Enabled checked, View Data Key, Picking Mode (Position), Tooltip, Usage Hints (None), TabIndex (0), Focusable, Language Direction (Inherit).
- Style Sheet: Add Style Class to List, Extract Inlined Styles to New Class.
- Style Class List: None.

A UI element labeled 'Kick' is shown with the placeholder '<PlayerName>'. A tooltip 'Customization is very easy!' is displayed at the bottom.

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.



Important Session API calls

```
1 var sessionType = "default-session";
2 // access the session locally by its type (local ID)
3 var session = MultiplayerService.Instance.Sessions[sessionType];
4 // options used to configure the session
5 var sessionOptions = new SessionOptions{Type=sessionType, SessionProperties=new Dictionary<string, SessionProperty>()};
6 MultiplayerService.Instance.CreateSessionAsync(sessionOptions);
7 MultiplayerService.Instance.JoinSessionByIdAsync(session.Id);
8 // Session lifetime events
9 session.Changed += () => { }; session.Deleted += ()=>{ } ;
10 // Sessions report player connection events, note that the id is the service ID not the netcode client ID
11 // in case you want to map between the two, you may need to set up an approval callback on the netcode manager.
12 // see NGO docs for more information.
13 session.PlayerJoined += id => { }; session.PlayerHasLeft += id => { }; session.PlayerLeaving += id => { };
14 session.RemovedFromSession += ( ) => { }; // current player
15
16 // ISession.Properties gives access to the properties of a session that are synchronized between all participants
17 Debug.Log(string.Join('\n', session.Properties.Keys ));
18
19 // provides easy access to lifetime callbacks of a session, useful for MVVM
20 var sessionObserver = new SessionObserver(sessionType);
21
22 sessionObserver.SessionAdded += _ => { }; // Session has been added locally
23 sessionObserver.AddingSessionStarted += _ => { }; // Session Create / Join was called locally
24
25 // Session Create / Join failed, check the exception for more detail
sessionObserver.AddingSessionFailed += (o,e) => { };
```



Thank you

2026

Good luck Jamming!



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)

