OWNER'S MANUAL

EZ BIND

PROFESSIONAL DATA BINDING SOLUTION

Thank you for buying our asset and for supporting its further development. This plugin was created to extend the functionality of Unity's native system. Should you need help, find issues or have any suggestions, don't hesitate to send us a message at support@ezentertainment.eu

Please read the quick setup guide and check out our video tutorials before you start using this asset.

Solutions – EzBind solves the following problems

Data binding is a general technique that binds variables together and synchronizes their values.

In a data binding process, each data change in the bind Source is reflected automatically in the Observers that are bound to it. Ez Bind is a one-to-many solution (one Source - to - many Observers)

You can **bind properties and variables between different classes** and you can virtually **bind everything, even between different scenes**.

Ez Bind performs automatic cleanup of removed (destroyed) objects, having a solid software architecture with a modular design. This allows the binding of data without the need of writing code.

This approach is used for the MVVM (Model-View-ViewModel) pattern that is very popular in the C# world.

Ez Bind has been optimized for all the mobile platforms.

Quick Setup Guide

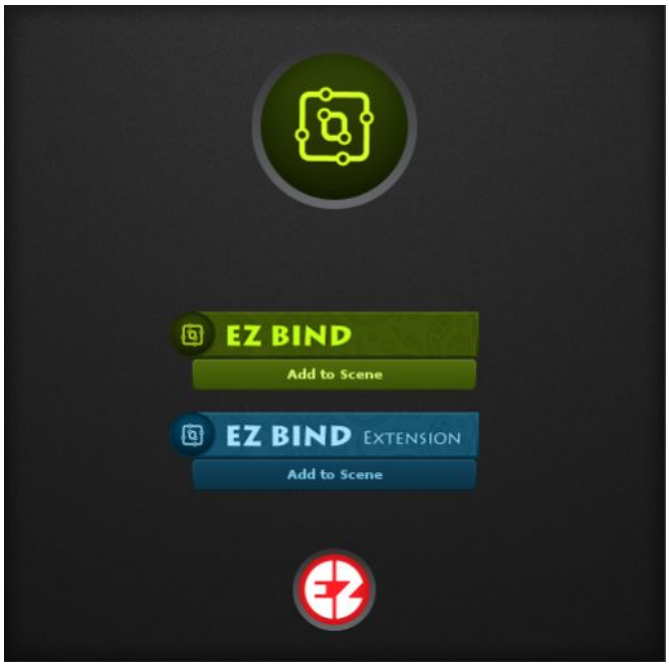
1. Import EzBind (from @UnityAssetStore).
2. In the editor top menu bar → Tools → Ez → Bind (EzBind is a singleton so you only need one in your game).
3. Click the Add to Scene button to add EzBind to the scene.
4. Configure the binds you need.
5. Done!

Video Tutorials

Watch the introduction videos @YouTube:

EzBind - Introduction: https://youtu.be/FA4KxUTudGQ

EzBind Window



This is the EzBind configuration window. It can be opened from the Editor top menu → Tools → Ez → Bind. Using it you can add or remove the EzBind main component or the EzBind Extension to and from your currently opened scene.
EzBind's main component is a singleton and should be added only in your main scene (or start scene) and it will persist across scenes.

Terminology

The following terms are used when describing EzBind and its functionality:

- Bind:
  - Name given to EzBind's internal logical construct that allows the synchronization of other object's values.
  - A bind allows the linking of any existing types without restriction. This allows great flexibility, but it is the user's responsibility to bind compatible types.

- Bind Name:
  - Name of the Bind. **Must be unique among all Binds.**

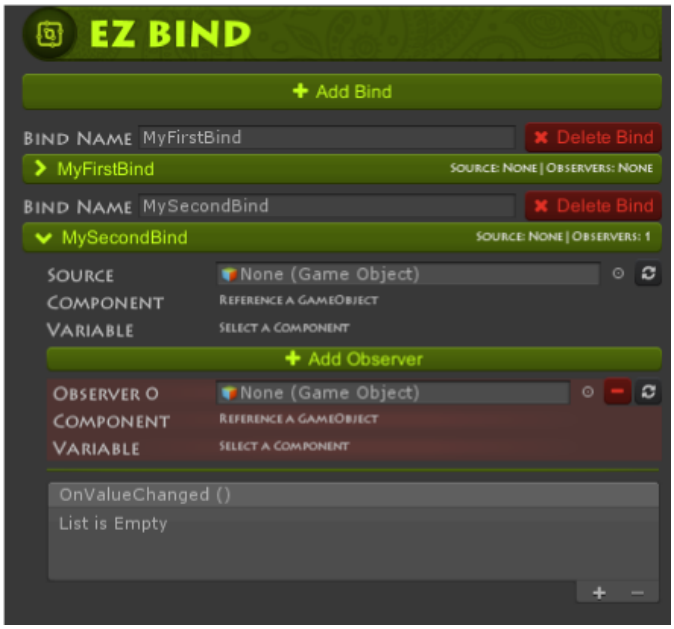
- Source:
  - Field or property whose value is monitored by the Bind to detect changes.
  - Normal configuration: the source is configured and the Bind, on LateUpdate(), will always check the value of the source. If the source's value has changed, the Bind will synchronize its own value and all the observers as well as invoke the listener methods.
  - Advanced usage: the source is not configured and the Bind's check is not called on LateUpdate(). The synchronization of observer values, as well as the invoke for the listeners are triggered by changing the Bind's own value.

- Observers:
  - fields or properties whose values are synchronized to the Bind value.

- OnValueChanged
  - Listener methods that are invoked whenever the Bind value changes.

EzBind Component

This is EzBind's main component. Here you can configure permanent binds that will be active at all times. This component is a singleton that will remain available even between scene changes.



Editor inspector overview:

- Add Bind - button that allows the creation of a new Bind
- Bind Name - name given to the Bind. This name can be used to find this specific bind either from other components or from code. **The Bind Name must be unique**.
- Bind Source:
    - Source - monitored GameObject.
    - Component - monitored component on the Source GO.
    - Variable - monitored public variable of the specified Component of the Source GO. On LateUpdate() the Bind will check for changes and update if needed.
- Bind Observers:
    - Add Observer - button to add a new observer to the Bind
    - Observer # - synchronized GameObject
    - Component - synchronized component on the Observer GO
    - Variable - synchronized public variable of the specified Component of the Observer GO. This value will automatically be updated to the value of the Bind.
- OnValueChanged - listener methods. Whenever there is a change in the value of the Bind these methods will be invoked.

API and Code Examples

All Binds can easily be configured in the editor or using the helper scripts provided with the asset. If needed however, binds can also be added or configured using code.

To be able to use EzBind in your scripts, remember to add:

```
using Ez.Binding;
```

If it is needed to handle references to Bind or their sources/observers from code, also add:

```
using Ez.Binding.Vars;
```

Creating a New Bind

There are several methods available if it is needed to create a new Bind from code:

- Creates a new Bind with the specified unique name and returns a reference to it:

```
EzBind.AddBind(string newBindName)
// Returns a reference to the newly created Bind; if unsuccessful, returns null
```

- Creates a new Bind with the specified unique name and with an initial value and returns a reference to it:

```
EzBind.AddBind(string newBindName, object newBindValue)
// Returns a reference to the newly created Bind; if unsuccessful, returns null
```

- Creates a new Bind with the specified unique name and pass it a specific UnityEvent to invoke when its value changes and returns a reference to it:

```
EzBind.AddBind(string newBindName, UnityEvent onValueChanged)
// Returns a reference to the newly created Bind; if unsuccessful, returns null
```

- Creates a new Bind with the specified unique name, with an initial value and pass it a specific UnityEvent to invoke when its value changes and returns a reference to it:

```
EzBind.AddBind(string newBindName, object newBindValue, UnityEvent onValueChanged)
// Returns a reference to the newly created Bind; if unsuccessful, returns null
```

Removing a Bind

To permanently remove an existing Bind, the following methods are available:

- Removes the bind specified by name and destroys it:

```
EzBind.RemoveBind(string bindName)
// If the Bind with this name does not exist, it does nothing
```

- Removes the bind specified by reference and destroys it:

```
EzBind.RemoveBind(Bind bind)
// If this Bind does not exist, it does nothing
```

Finding a Reference to an Existing Bind

If needed, a reference to an existing Bind can be found by searching after the Bind's name:

```
var bind = EzBind.FindBindByName(string bindName)
// Finds and returns an existing Bind; if the bind does not exist, returns null.
```

Adding a New Observer to a Bind

To add a new observer to an existing Bind:

- Adds a new observer to the referenced bind and returns a reference to the new observer:

```
EzBind.AddObserverToBind(Bind bind, UnityEngine.Object parent, string observerName)
// Returns a reference to the new observer or null, if unsuccessful.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" synchronized to the Bind
// The type of myVariable must be compatible with the type of the Bind
// Inside the script holding myVariable:
```

```
var bind = EzBind.FindBindByName("MyBind");
EzBind.AddObserverToBind(bind, this, "myVariable");
```

- Adds a new observer to the specified bind and returns a reference to the new observer:

```
EzBind.AddObserverToBind(string bindName, UnityEngine.Object parent, string observerName)
// Returns a reference to the new observer or null, if unsuccessful.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" synchronized to the Bind
// The type of myVariable must be compatible with the type of the Bind
// Inside the script holding myVariable:
EzBind.AddObserverToBind("MyBind", this, "myVariable");
```

- Adds a new observer to the referenced bind. The observer is a reference to an observer previously created. Useful if you need to re-add an observer reference that was removed - this method is much better than re-creating an observer. Returns true if successful, false otherwise.

```
EzBind.AddObserverToBind(Bind bind, ReferencedVariable observerVar)
// Returns true if the operation was successful, false otherwise.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" synchronized to the Bind
// The type of myVariable must be compatible with the type of the Bind
// Inside the script holding myVariable:
var bind = EzBind.FindBindByName("MyBind");
var observer = EzBind.AddObserverToBind(bind, this, "myVariable");
// Observer is removed at some point
EzBind.AddObserverToBind(bind, observer);
```

- Adds a new observer to a bind identified by its name. The observer is a reference to an observer previously created. Useful if you need to re-add an observer reference that was removed - this method is much better than re-creating an observer. Returns true if successful, false otherwise.

```
EzBind.AddObserverToBind(Bind bind, ReferencedVariable observerVar)
// Returns true if the operation was successful, false otherwise.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" synchronized to the Bind
// The type of myVariable must be compatible with the type of the Bind
// Inside the script holding myVariable:
var observer = EzBind.AddObserverToBind("MyBind", this, "myVariable");
// Observer is removed at some point
EzBind.AddObserverToBind("MyBind", observer);
```

Removing an Observer from a Bind

To remove an observer from an existing bind, you need to have a reference to that observer. The following methods can be used for this task:

- Removes a referenced observer from the referenced bind. Returns true if successful, false otherwise.

```
EzBind.RemoveObserverFromBind(Bind bind, ReferencedVariable refVar)
// Returns true if successful, false otherwise.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" synchronized to the Bind
// The type of myVariable must be compatible with the type of the Bind
// Inside the script holding myVariable:
var bind = EzBind.FindBindByName("MyBind");
var observer = EzBind.AddObserverToBind(bind, this, "myVariable")
```

```
// Later, to remove the observer:
EzBind.RemoveObserverFromBind(bind, observer);
```

- Removes a referenced observer from bind specified by name. Returns true if successful, false otherwise.

```
EzBind.RemoveObserverFromBind(string bindName, ReferencedVariable refVar)
// Returns true if successful, false otherwise.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" synchronized to the Bind
// The type of myVariable must be compatible with the type of the Bind
// Inside the script holding myVariable:
var observer = EzBind.AddObserverToBind("MyBind", this, "myVariable")
// Later, to remove the observer:
EzBind.RemoveObserverFromBind("MyBind", observer);
```

Adding a New Source to a Bind

To add a new source to a Bind or to replace the current source:

- Adds a new source to the referenced bind and returns a reference to the source.

```
EzBind.AddSourceToBind(Bind bind, UnityEngine.Object parent, string sourceName)
// If another source was defined, it is replaced.
// Returns a reference to the source or null, if unsuccessful.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" as the source value for the Bind
// The type of myVariable will automatically change the type of the Bind's value (so it should be
// compatible with any existing observers).
// Inside the script holding myVariable:
var bind = EzBind.FindBindByName("MyBind");
EzBind.AddSourceToBind(bind, this, "MyBind");
```

- Adds a new source to the specified bind and returns a reference to the source.

```
EzBind.AddSourceToBind(string bindName, UnityEngine.Object parent, string observerName)
// If another source was defined, it is replaced.
// Returns a reference to the source or null, if unsuccessful.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" as the source value for the Bind
// The type of myVariable will automatically change the type of the Bind's value (so it should be
// compatible with any existing observers).
// Inside the script holding myVariable:
EzBind.AddSourceToBind("MyBind", this, "MyBind");
```

- Adds a new source to the referenced bind. The source is a reference to a source previously created. Useful if you need to re-add a source reference that was removed or replaced - this method is much better than re-creating the source. Returns true if successful, false otherwise.

```
EzBind.AddSourceToBind(Bind bind, ReferencedVariable sourceVar)
// If another source was defined, it is replaced. Returns true if successful, false otherwise.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" as the source value for the Bind
// The type of myVariable will automatically change the type of the Bind's value (so it should be
// compatible with any existing observers).
// Inside the script holding myVariable:
var bind = EzBind.FindBindByName("MyBind");
var source = EzBind.AddSourceToBind(bind, this, "myVariable");
// Source is removed or replaced at some point
EzBind.AddSourceToBind(bind, source);
```

- Adds a new source to a bind identified by its name. The source is a reference to a source previously created. Useful if you need to re-add a source reference that was removed or replaced - this method is much better than re-creating the source. Returns true if successful, false otherwise.

```
EzBind.AddSourceToBind(string bindName, ReferencedVariable sourceVar)
// If another source was defined, it is replaced. Returns true if successful, false otherwise.

// Usage example:
// The Bind name is "MyBind"
// We need to have the value of a variable named "myVariable" as the source value for the Bind
// The type of myVariable will automatically change the type of the Bind's value (so it should be
// compatible with any existing observers).
// Inside the script holding myVariable:
var source = EzBind.AddSourceToBind("MyBind", this, "myVariable");
// Source is removed or replaced at some point
EzBind.AddSourceToBind("MyBind", source);
```

Removing a Source from a Bind

To remove the source from an existing bind, use two of the EzBind.AddSourceToBind() methods and pass null as the source parameter:

```
// Usage example:
// The Bind name is "MyBind"
// To remove its source:
var bind = EzBind.FindBindByName("MyBind");
EzBind.AddSourceToBind(bind, null);
// OR
EzBind.AddSourceToBind("MyBind", null);
```

Add a New Listener to a Bind

Listeners are methods that are automatically invoked when the Bind's value changed. To add a new listener method to a bind:

- Adds a listener method to a referenced bind.

```
EzBind.AddListenerToBind(Bind bind, UnityAction listener)

// Usage example:
// The Bind name is "MyBind"
// When the Bind value changes, we want to have the MyListener method invoked
// public void MyListener() { /* code here */ }
var bind = EzBind.FindBindByName("MyBind");
EzBind.AddListenerToBind(bind, MyListener);
```

- Adds a listener method to a bind specified by name. Returns true if successful, false otherwise (if the bind does not exist).

```
EzBind.AddListenerToBind(string bindName, UnityAction listener)
// Returns true if successful, false otherwise (if the bind does not exist)

// Usage example:
// The Bind name is "MyBind"
// When the Bind value changes, we want to have the MyListener method invoked
// public void MyListener() { /* code here */ }
EzBind.AddListenerToBind("MyBind", MyListener);
```

Removing a Listener from a Bind

Listener methods can be removed from existing binds, as follows:

- Removes the listener from the referenced bind.

```
EzBind.RemoveListenerFromBind(Bind bind, UnityAction listener)

// Usage example:
// The Bind name is "MyBind"
// When the Bind value changes, we no longer want to have the MyListener method invoked
// public void MyListener() { /* code here */ }
var bind = EzBind.FindBindByName("MyBind");
EzBind.RemoveListenerFromBind(bind, MyListener);
```

- Removed the listener method from the bind specified by name. Returns true if successful, false otherwise (if the bind does not exist).

```
EzBind.RemoveListenerFromBind(string bindName, UnityAction listener)
// Returns true if successful, false otherwise (if the bind does not exist)

// Usage example:
// The Bind name is "MyBind"
// When the Bind value changes, we want to have the MyListener method invoked
// public void MyListener() { /* code here */ }
EzBind.RemoveListenerFromBind("MyBind", MyListener);
```

EzBind Extension Component

The EzBind Extension allows you to extend binds across scenes. The extension allows:

- Configuring new sources, observers and listeners that are can be added to existing Binds when the extension is created.
- New Binds can be configured that will only exist for as long as the extension itself remains active (useful when loading other scenes).

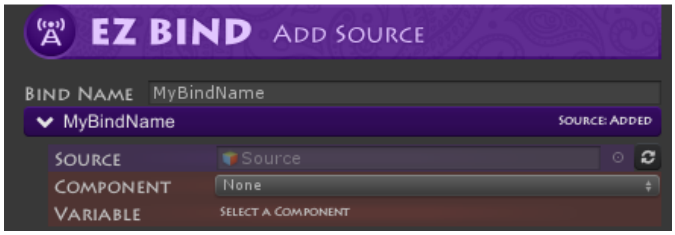
The binding configurations added by an EzBind extension are available as long as the extension itself is active and enabled. When the extension is disabled, it will automatically cleanup all the configuration it has added.

Editor inspector overview:

- Add Bind - button that allows the creation of a new Bind
- Bind Name - name given to the Bind.
    - If the name is unique, a new Bind with this name is created
    - If the name corresponds to an existing Bind, the configured source, observers and listeners are added to that Bind for as long as the extension remains active.
- Bind Source:
    - Source - monitored GameObject.
    - Component - monitored component on the Source GO.
    - Variable - monitored public variable of the specified Component of the Source GO. On LateUpdate() the Bind will check for changes and update if needed.
- Bind Observers:
    - Add Observer - button to add a new observer to the Bind
    - Observer # - synchronized GameObject
    - Component - synchronized component on the Observer GO
    - Variable - synchronized public variable of the specified Component of the Observer GO. This value will automatically be updated to the value of the Bind.
- OnValueChanged - listener methods. Whenever there is a change in the value of the Bind these methods will be invoked.

EzBind Add Source Component

The EzBind Add Source component is a tool that allows automatically creates a Bind source from a variable on its own GameObject and ads it to an existing Bind.



Editor inspector overview:

- Bind Name - the name of the target Bind. If the Bind with this name does not exist, it will automatically be created.
- Bind Source:
    - Source - will always be the GameObject of the script.
    - Component - monitored component on the Source GO.
    - Variable - monitored public variable of the specified Component of the Source GO. On LateUpdate() the Bind will check for changes and update if needed.

Note: If the inspector configuration is not complete, the script will display an Error message and destroy itself.

EzBind Add Observer Component

The EzBind Add Source component is a tool that allows automatically creates a Bind observer from a variable on its own GameObject and adds it to an existing Bind.

Editor inspector overview:

- Bind Name - the name of the target Bind. If the Bind with this name does not exist, it will automatically be created.
- Bind Observer (only one allowed):
    - Observer - will always be the GameObject of the script.
    - Component - synchronized component on the Observer GO.
    - Variable - synchronized public variable of the specified Component of the Observer GO. This value will automatically be updated to the value of the Bind.
- Update Observer when this GameObject is disabled - allows the configured observer to stay updated even ifs parent is disabled.

Note: If the inspector configuration is not complete, the script will display an Error message and destroy itself.