Serialize Reference Editor for Unity

A powerful Unity editor extension that provides complete control over **SerializeReference** fields with smart type management, validation tools, and data integrity features.

Installation

Download asset from Unity Asset Store: Serialize Reference Editor

Please, remove old version Serialize Reference Editor for Unity before update!

Or installation as a unity module via a git link in PackageManager:

```
https://github.com/elmortem/serializereferenceeditor.git?path=SerializeReferenceEditor/Assets/SREditor/Package
```

Or direct editing of `Packages/manifest` is supported.json:

```
"com.elmortem.serializereferenceeditor": "https://github.com/elmortem/serializereferenceeditor.git?path=SerializeReferenceEditor/Ass
```

Main types

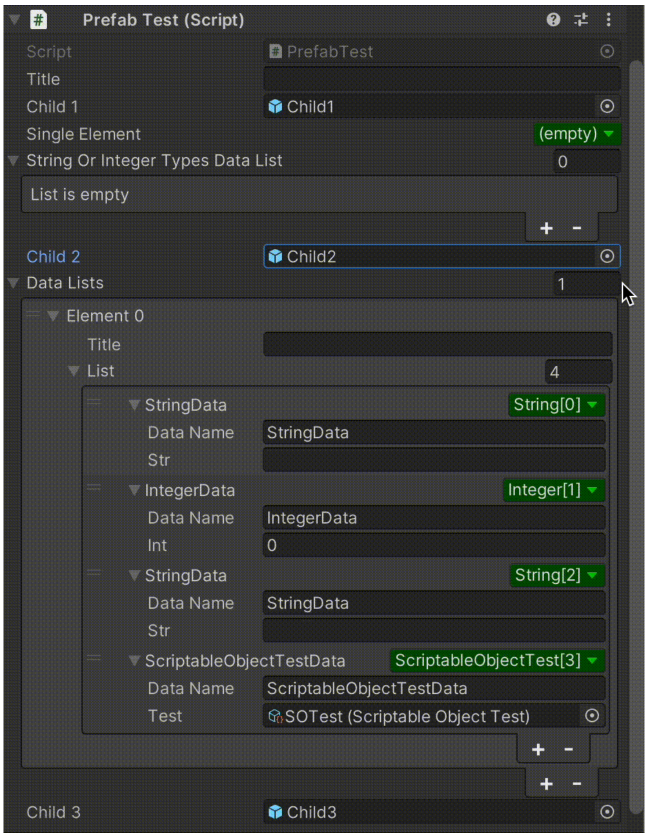
SR attribute

Use it to mark the fields that you want to edit. Important note - they should also be marked with the SerializeReference attribute. You can mark the base type (including the interface) with it. Allows you to change an array, a list, and a single element. Displays the current field type.

Example:

```
 [SerializeReference, SR]
public List<AbstractData> DataList = new List<AbstractData>();
```

Result:



```
Prefab Test (Script)
Script                              PrefabTest
Title
Child 1                             Child1
Single Element                                  (empty)
String Or Integer Types Data List               0
  List is empty
Child 2                             Child2
Data Lists                                      1
  Element 0
    Title
    List                                        4
      StringData                                String[0]
        Data Name                   StringData
        Str
      IntegerData                               Integer[1]
        Data Name                   IntegerData
        Int                         0
      StringData                                String[2]
        Data Name                   StringData
        Str
      ScriptableObjectTestData      ScriptableObjectTest[3]
        Data Name                   ScriptableObjectTestData
        Test                        SOTest (Scriptable Object Test)
Child 3                             Child3
```

Additional features

You can override SRAttribute and implement a rule for processing instantiated objects.

You can see an example in SRDemoAttribute.cs, where the `OnCreate` method was overriden:

```
public override void OnCreate(object instance)
{
    if(instance is AbstractData)
    {
        ((AbstractData)instance).DataName = instance.GetType().Name;
    }
}
```

SRName attribute

Mark classes with them if you want to customize the display name and nesting hierarchy in the search tree for a specific type.

Example FloatData.cs:

```
[SRName("Data/Simple types/Float")]
public class FloatData : AbstractData
{
    [Range(0f, 1f)]
    public float Float;
}
```

You can modify the display settings for the class name without specifying an attribute by navigating to `Edit -> Project Settings -> SREditor`.

SRHidden attribute

Mark classes with this attribute to hide them from the type selection menu. Types with this attribute will not appear in the search tree.

SRDrawer

SRDrawerOptions

Provides configuration options for the SRDrawer:

- `WithChild` - Controls whether child properties should be displayed
- `ButtonTitle` - Controls whether the type name should be displayed on the button
- `DisableExpand` - Disables the property expansion functionality

SRDrawer.Draw

`SRDrawer.Draw` is a key method for rendering fields with the `SerializeReference` attribute. It provides the following functionality:

Key Features

- Dynamic type resolution for the field
- Type selection button displaying current type name
- Array elements support
- Integration with Unity's SearchWindow for type selection

Method Parameters

- `position` : Drawing area position and dimensions
- `property` : Serialized property to draw
- `label` : Field label to display
- `types` : Optional array of allowed types

Example CustomDataDrawer.cs

```
[CustomPropertyDrawer(typeof(CustomData))]
public class CustomDataDrawer : PropertyDrawer
{
    private SRDrawer _drawer = new();

    public override void OnGUI(Rect position, SerializedProperty property, GUIContent label)
    {
        var dataProperty = property.FindPropertyRelative("Data");
        _drawer.Draw(position, dataProperty, label);
    }
}
```

SRDrawer.GetButtonWidth

`SRDrawer.GetButtonWidth` method allows to get the width of the type selection button. Useful for custom property drawers when you need to adjust layout based on the button size.

Method Parameters

- `property` : Serialized property to calculate button width for
- `options` : SRDrawerOptions to use for calculation

Tools

Missing Types Validator

You can use the built-in tool to search for lost types. (this error occurs if the serialized data stores information about a type that no longer exists and could have been deleted during development). To start, you can run `Tools -> SREditor -> Check MissingTypes`

> At least one `SRMissingTypesValidatorConfig` is required to work. If necessary, you can implement your own `IAssetMissingTypeReport` for error reporting if you use it in CI/CD systems. You can also implement your own `IAssetsLoader` if the default `LoadAllScriptableObjects` is not suitable for you.

Class Replacer

Use `Tools -> SREditor -> Class Replacer` for replace Serialize Reference classes.

FormerlySerializedType attribute

It is analogue of attribute FormerlySerializedAs, but works for Serialize Reference classes.

Example NewTestData.cs:

```
[Serializable, SRName("New Test")]
[FormerlySerializedType("SRDemo, Demo.OldTestData")]
public class NewTestData : BaseTestData
```

Duplicate Cleaner

Now Serialize Reference Editor can auto detect and handle SerializeReference object duplicates with flexible settings - you can nullify them, create them with default values, or make deep copies, preventing issues with unwanted reference sharing in your assets.

Thanks

Andrey Boronnikov

Georg Meyer

https://www.markdowntopdf.com

Support Unity 2021.3 or later.

Use for free.

Enjoy!