Serializable Dictionary

Rotary Heart

Documentation Version: 1.2

Email: ma.rotaryheart@gmail.com

Description:

This package contains a class, SerializableDictionaryBase.cs, that can be inherited to be able to have a serializable dictionary. There is no limitation for what to use as Key or as Value, other than it being a serializable type. It also contains a class, DataBaseExample.cs, that has a couple of examples of how to setup the dictionary.

Setup:

To use simply add the SerializableDictionary folder to your project (which is on the package). It doesn't need to be on root of your project, so feel free to move it wherever you want.

How to use:

This section is divided into 3 different cases for easier explanation, but for any type of implementation you will need to create a custom class, can be a nested class, that inherits from SerializableDictionaryBase and setup the respective types for key and value that are going to be used by the dictionary.

Note that for Unity to be able to serialize this class you need to add the attribute [System.Serializable] to your class since that is the way Unity serialization works.

1. Simple Dictionary

```
[System.Serializable]
public class MyDictionary :
SerializableDictionaryBase<string, GameObject> { }
```

On this example you can see that the dictionary that we will be using has a string for key and a GameObject for value. With this now you can have a field of type MyDictionary and you will be able to modify the data from the editor.

2. UnityEngine.Object as a key

If you want to use a UnityEngine.Object (GameObject, Texture, Sprite, etc) type for key, there's an extra step required for the system to handle it.

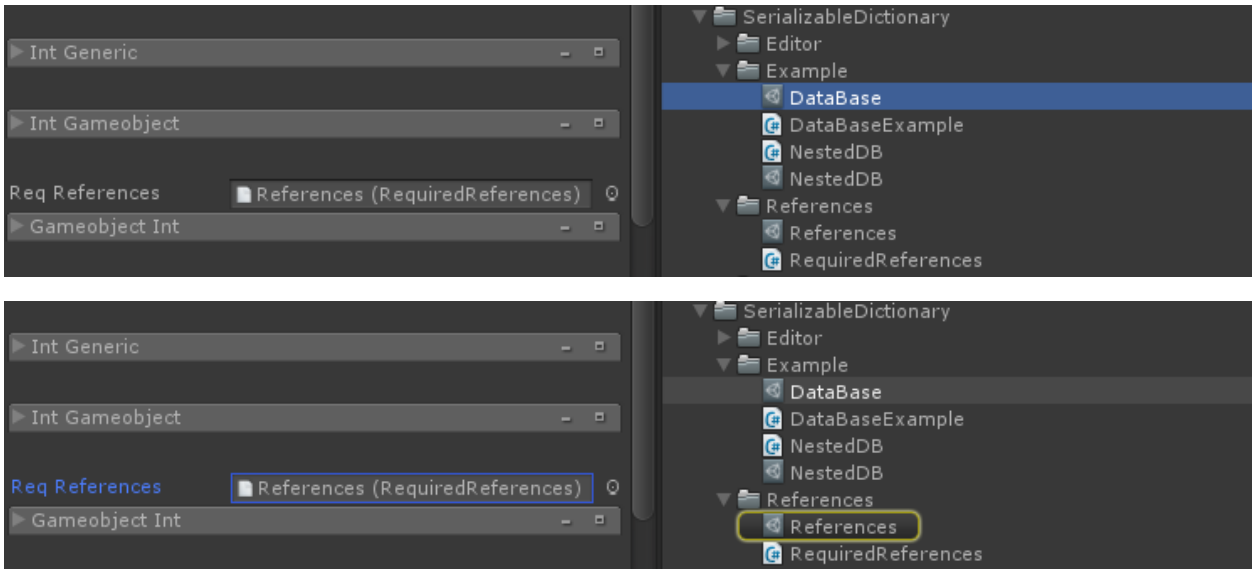
```
[System.Serializable]
public class MyDictionary :
SerializableDictionaryBase<GameObject, string> { }
```

Now you will need to go to the script RequiredReferences.cs and add the reference needed here (in this case GameObject).

```
[UnityEngine.SerializeField]
private GameObject _gameObject;
```

This is needed because the Dictionary can't handle null for key so it requires a default value and the editor already gives that value.

The last thing you will need is to setup the reference for the RequiredReferences object on your editor. Drag and drop the References scriptable object to the editor.



Now that everything is setup correctly you can start adding elements to your dictionary.

3. Advanced Dictionary

If the value that you want to use is either an Array or a List, you cannot use it directly into the dictionary value type because Unity doesn't serialize Array of Array or list of List. Instead you need to wrap your Array or List with a custom class.

```
[System.Serializable]
public class Int_IntArray : SerializableDictionaryBase<int,
MyClass> { }

[System.Serializable]
public class MyClass
{
```

```
    public int[] myArray;
}

[SerializeField]
private Int_IntArray _intArray;
```

This way unity can serialize your array and the system will work without any problems.