PersistentSingleton documentation

- Overview
- Abbreviation
- HowTo
- Examples
- Source code
- Feedback
- Release

Overview

This package help you share data between scenes. As well you can save and restore preferences, like save game or so on.

This package use standart **PlayerPrefs** as config holder.

Abbreviation

PlayerPrefs Standard class from Unity3d package (Stores and accesses player preferences between game sessions.)

Singleton is a design pattern that restricts the instantiation of a class to one object.

HowTo

Create a **class** derrived from a **PersistentMonoSingleton.**

```
using System.Collections;

[Persistent("playerSetup",true)]
public class PlayerPreferences : PersistentMonoSingleton<PlayerPreferences> {

 [Persistent("myIntVar")]
 public int myIntVar;

 [Persistent("myFloatVar")]
 public float myFloatVar;

 [Persistent("privateField")]
 private int myPrivateInt;

 [Persistent]
 public string myString;
}
```

[Persistent("playerSetup",true)] is not mandatory attribute for class, it indicate:

- **playerSetup** is a prefix which will be used in **PlayerPrefs**
- **true** property will be automatically synchronized with **PlayerPrefs** on singleton initialization

Mark all properties which you want to store in **PlayerPrefs** with attribute **[Persistent("Name")]**

- Parametr **Name** is not mandatory, and used only for key in **PlayerPrefs** if you will not provide this parameter, field name will be used

If you want to **save/restore/reset** your parameters in **Editor**, you need to create **Editor class** and put it to **Assets/Editor** folder, derrive your class from **PersistentMonoSingletonEditorBase**

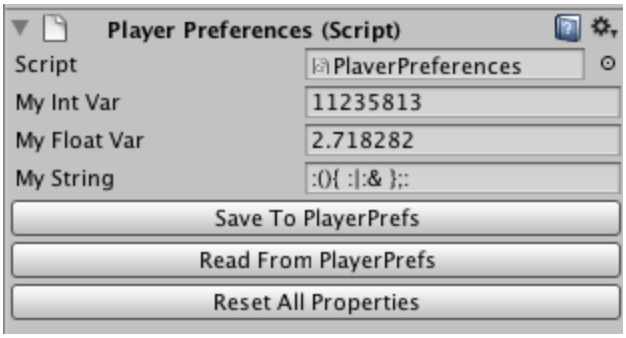
```
using UnityEngine;
using System.Collections;
using UnityEditor;

[CustomEditor(typeof(PlayerPreferences))]
public class PlayerPreferencesEditor :
PersistentMonoSingletonEditorBase<PlayerPreferences>{

}
```

Create empty object on your scene, and attach previously created singleton on to it

In editor you will see all your public properties, and 3 buttons(if you created editor)



Creation of this object on the scene is not mandatory, you can just use following syntaxis to access your properties in your script

```
PlayerPreferences.instance.readPropertiesFromPlayerPrefs();  //synchronize
properties with PlayerPrefs
PlayerPreferences.instance.myFloatVar=3.14f;                 //read property value
float myFloatVar=PlayerPreferences.instance.myFloatVar;      //assign new value
PlayerPreferences.instance.savePropertiesToPlayerPrefs();    //save new properties
```

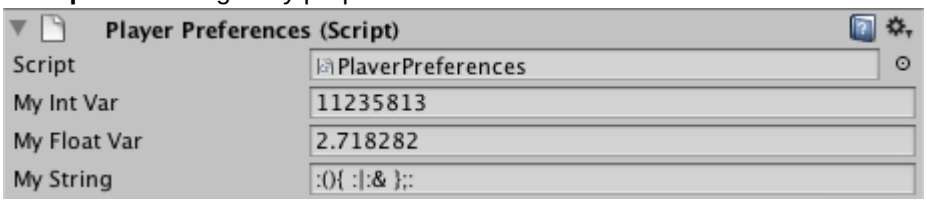
Be carefully with **Reset All Properties** button or **PlayerPreferences.instance.resetAllProperties()** method because it will erase all stored configs.

Until you will stave your properties, using **button**, or **savePropertiesToPlayerPrefs** method, all your property values will be available only in current scene.

Examples

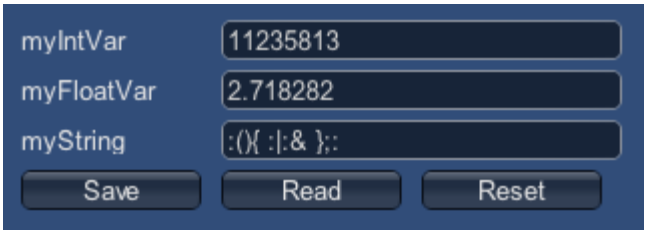
You can find 3 scenes in **Assets/Demo** folder

1. Open **editor_scene1** scene
2. Click on **PlayerPreferencesS1** object
3. In **Inspector** change any properties



4. If you will switch to another scene, this properties will be losed, because it won't stored in **PlayerPrefs** this values available only in current scene
5. So click on **Save To PlayerPrefs** button in **Inspector**

6. Switch to **editor_scene2**
7. Make sure that all your properties restored from **PlayerPrefs** properly, you can try to change properties here, save them and go back to scene1.
8. Switch to **ingame_scen3**
9. As you can see there is no any object here except of **Main Camera**
10. Start playing this scene and you must see al your properties



This happens,because of the **TestHUD** script attached to the Camera.

11. While you're playing this scene you can see **Temp Instance of PlayerPreferences** object, which created automaticaly when you first time call to any properties of your singleton
12. Change any properties here, click Save, and go back to any of editor scenes, and make sure that you see lates saved properties there.

Source code

Latest source code is available here https://github.com/nicloay/persistentSingleton

Feedback

I'll be glad to receive your feedback, feature request, or bug report, by email nicloay [a] gmail.com or on unity forum thread forum.unity3d.com

Release

- **v1.0** (Initial release) - implemented main functionality, load/save int,string,float values, editorBase, singletonBase classes
- **v1.001** - redesign, new types supported (Vector2,Vector3, Quaternion,Color), easy way to implement support of custom type (just implement one interface and add new type to dictionary).