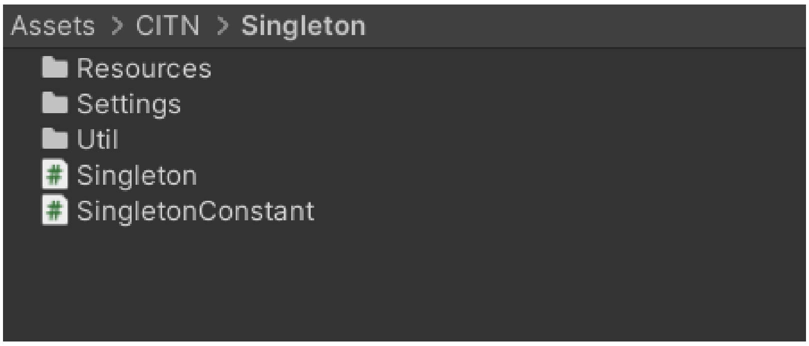
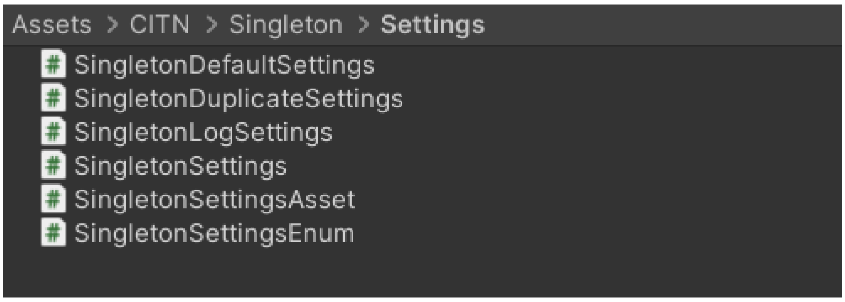
Package content


Assets > CITN > Singleton

- Resources
- Settings
- Util
- Singleton
- SingletonConstant

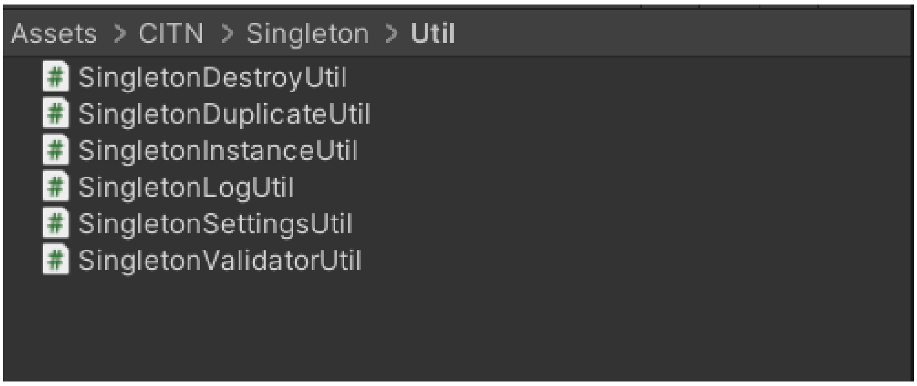
“Settings” folder


Assets > CITN > Singleton > Settings

- SingletonDefaultSettings
- SingletonDuplicateSettings
- SingletonLogSettings
- SingletonSettings
- SingletonSettingsAsset
- SingletonSettingsEnum

Classes-model for settings
Class for create settings asset

“Util” folder

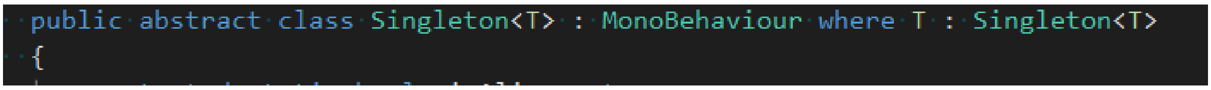


Assets > CITN > Singleton > Util

- SingletonDestroyUtil
- SingletonDuplicateUtil
- SingletonInstanceUtil
- SingletonLogUtil
- SingletonSettingsUtil
- SingletonValidatorUtil

Helper class for performing actions that we can select in the settings

Main-class Singleton.cs
If you want to use singleton you need...



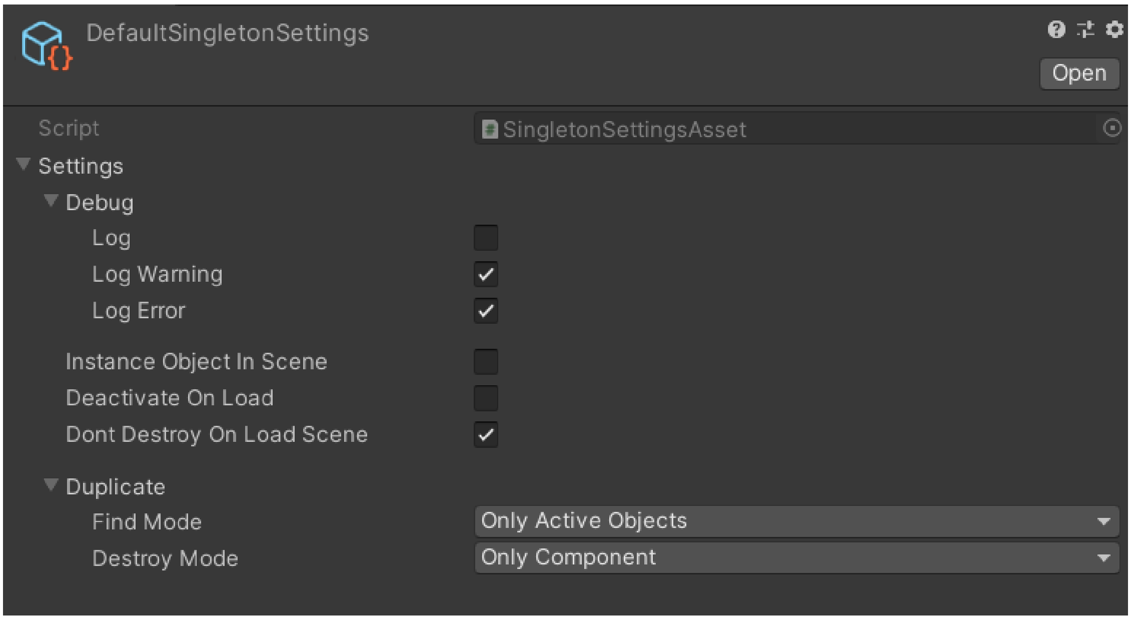
```
public abstract class Singleton<T> : MonoBehaviour where T : Singleton<T>
{
```

SingletonConstant.cs
Class-helper for constant values

Settings

Default settings

From beginning the package has a configuration-file with default settings.
Path: CITN\Singleton\Resources\DefaultSingletonSettings.assets



DefaultSingletonSettings

Open

Script: SingletonSettingsAsset

Settings
- Debug
  - Log ☐
  - Log Warning ☑
  - Log Error ☑
- Instance Object In Scene ☐
- Deactivate On Load ☐
- Dont Destroy On Load Scene ☑
- Duplicate
  - Find Mode: Only Active Objects
  - Destroy Mode: Only Component

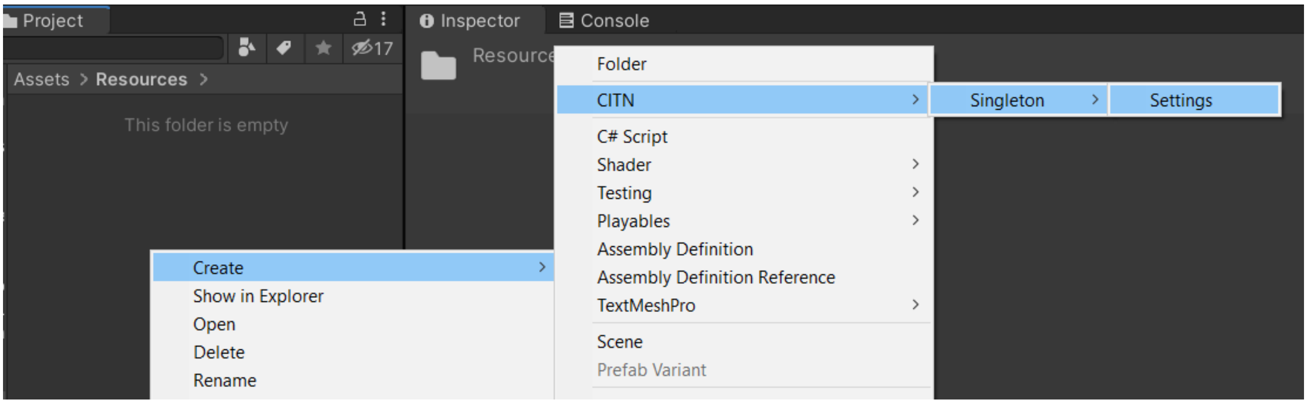
This file applies to all Singletons in your project.
If:

- The file is in the project and is located in the Resources folder.
- You have not overridden the type-specific settings

If for any reason you delete this file, or something else.
And your singleton doesn't have its own settings file. Will be applied creating a settings model at runtime (return new SingletonSettings ();)

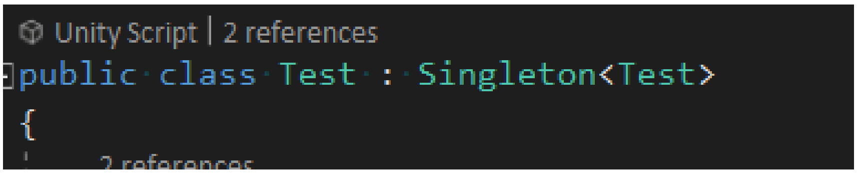
Type-specific settings

Open any “Resources” folder.
Context-menu (right click)
Selected:Create\CITN\Singleton\Settings

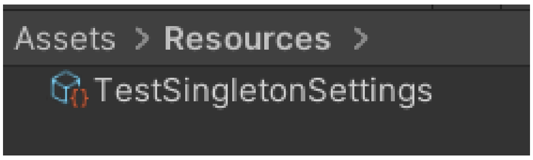


Rename file.
New filename = ClassName + SingletonSettings

Example



```
Unity Script | 2 references
public class Test : Singleton<Test>
{
    2 references
```



Classes for settings

SingletonSettingsAsset.cs - for create settings asset (ScriptableObject)
SingletonSettings.cs - class-model

Parameters\Configuration

Debug

- Log - Debug.Log
- LogWarning - Debug.LogWarning
- LogError - Debug.LogError

All log messages have tile - "CITN.Singleton"
See SingletonLogUtil.cs for all errors\warning messages

Instance Object In Scene - if (true) Create object in scene, if need it
Deactivate On Load - if (true) gameObject.SetActive(false); when initialize
Don't Destroy On Load Scene - if (true), Do not destroy the target Object when loading a new Scene.

```
Instance
{
        get{
                …
                if (_instance is empty)

                Logic for Find duplicate
                Logic for Destroy duplicate
                ...
            }
}
```

Duplicate (SingletonDuplicateSettings.cs)

Find Mode - Which ones to look for

- None - Don't use search at all
- Only Active Objects - Find only active objects
- All - Find all objects (active and inactive objects)

Destroy Mode - Destroy what.

Destroy use when Instance get, or Awake() function

Destroy is used when getting an instance or in the Awake () function

- None - Don't use destroy at all
- Only Component - Destroy only component
- Game Object - Destroy gameObject