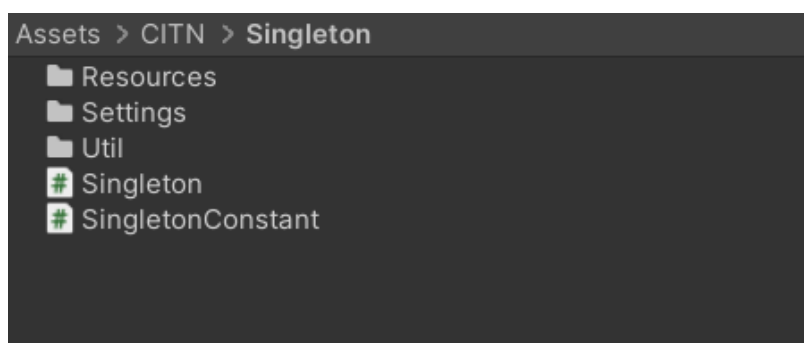
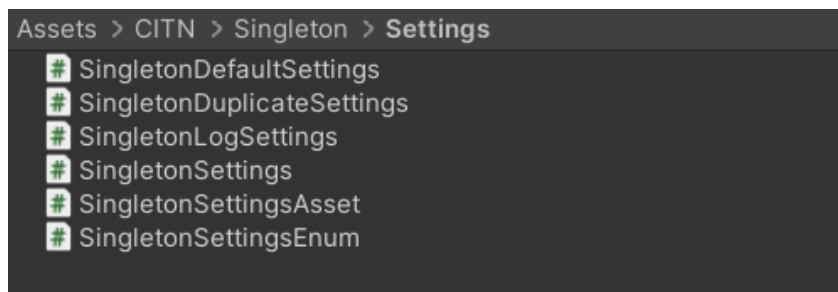




Структура пакета

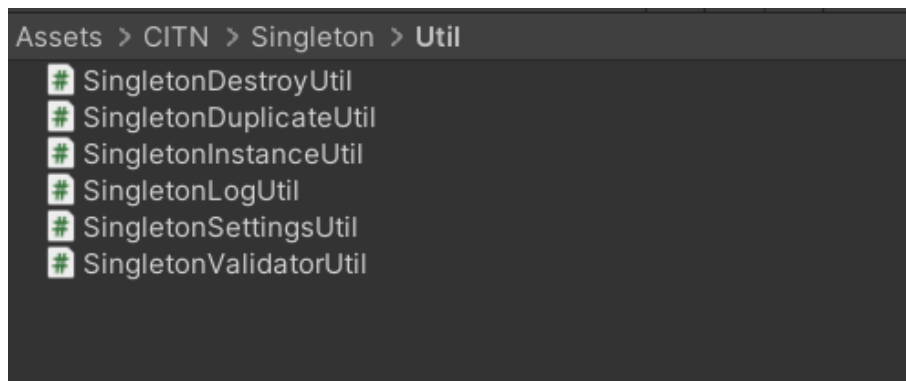


Папка "Settings"



Классы выступающие в роли модели данных.
Класс отвечающий за создание файла настроек

Папка "Util"



Вспомогательные классы. Для выполнения действий указанных в настройках, логирования, валидации данных...

Основной класс Singleton.cs

Его мы и будем использовать если хотим сделать из обычного MonoBehaviour - singleton

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

Класс SingletonConstant.cs

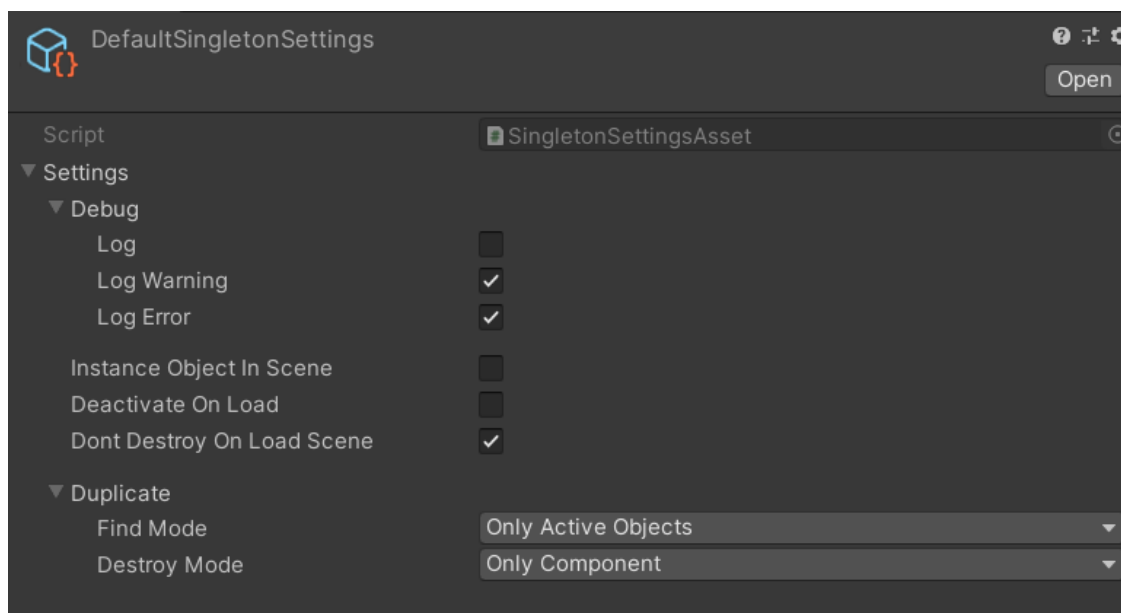
Вспомогательный класс для хранения константных значений

Настройки

Настройки по умолчанию

Изначально в пакете уже имеется готовый файл настроек.

Находится CITN\Singleton\Resources\DefaultSingletonSettings.assets



Он будет применяться ко всем Singleton в вашем проекте.

Если:

- Файл есть в проекте и находится в папке Resources
- Вы не переопределили настройки для конкретного типа

Если по какой либо из причин вы удалите его, или еще что-то.

И для вашего singleton нет своего файла настроек. Будет применено создание модели настроек на лету (return new SingletonSettings();)

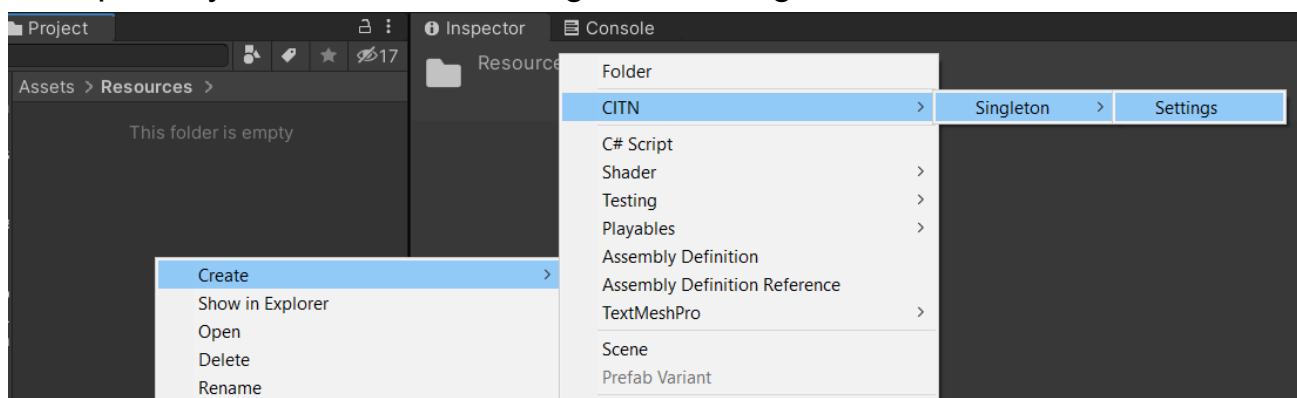
Настройки для конкретного типа

Для того чтобы создать свой файл настроек.

Перейдите в любую удобную для вас папку Resources и выполните следующее.

Контекстное меню (правой кнопкой мыши).

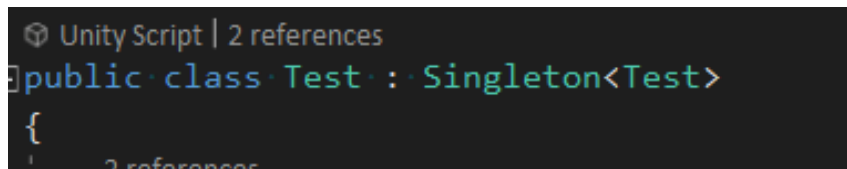
Выберите пункт Create\CITN\Singleton\Settings



Переименуйте файл.

Новое имя файла = Имя класса + SingletonSettings

Например



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



```
Assets > Resources >  
TestSingletonSettings
```

Классы настроек

SingletonSettingsAsset.cs - класс отвечающий за создание ScriptableObject с настройками

SingletonSettings.cs - класс, модель непосредственно самих настроек

Параметры\Конфигурация

Debug

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

Все сообщения начинаются с "CITN.Singleton"

С сообщениями об ошибках и предупреждении можно ознакомиться в классе SingletonLogUtil.cs

Instance Object In Scene - Если true, и при попытке получить объект (свойство Instance) на сцене не будет ни одного экземпляра, новый объект будет создан

Deactivate On Load - если true, выполнит gameObject.SetActive(false); при инициализации объекта.

Don't Destroy On Load Scene - если true, не уничтожать объект при загрузке новой сцены

```
Instance
{
    get{
        ...
        if (_instance еще не найден)

            Логика поиска дубликатов
            Логика удаления дубликатов
            ...
    }
}
```

Duplicate (SingletonDuplicateSettings.cs)

Find Mode - режим поиска дубликатов.

Выполняется при обращении к свойству Instance и если текущий _instance еще не найден.

- None - Не выполнять поиск вообще
- Only Active Objects - искать только активные объекты т.е. те что IsActive == true
- All - Искать все объекты, как активные так и нет.

Destroy Mode - режим удаления дубликатов.

Выполняется при обращении к свойству Instance и если текущий _instance еще не найден. Также выполняется в Awake если текущий объект не равен _instance

- None - Не выполнять удаление
- Only Component - удалять только компонент
- Game Object - удалять весь объект