Contact: *blartenixstudio@gmail.com*
Blartenix ®

Blartenix Common

namespace: Blartenix

Version: *v2.0*

Table of Contents

Introduction

Contents
- Documentation
- Prototyping
  - Game Settings
  - Pictograms
- Scripts
  - Components
    - Billboard
    - CanvasEnabler
    - TriggerPictogram
    - TriggerPictogram2D
    - SingletonBehaviour <T>
    - LanguageManager
    - LanguageText
    - ObjectPool <T>
    - TransformPool
    - MultipleObjectsPool <T>
  - Core
    - Scriptable Objects
      - GameSettings
      - LanguageTemplate
    - Classes
      - BlartenixLogger
      - Utilities
      - BlartenixLanguage
      - LanguageXmlTag
    - Enums
      - LogType
      - LookForTagAt
    - Interfaces
      - IBlartenixLogger

References

Contact: *blartenixstudio@gmail.com*
Blartenix ®

Introduction

Basic and essential package of the Blartenix Library for Unity.

This document is also a reference guide to the scripts that are part of this package, focused on the final developer user, so it shows the documentation of only the public members of interest (classes, properties, methods and components) so that you understand the elements that are contained within this package and thus can give you a better idea when designing your own projects.

IMPORTANT NOTE:
Version 2.x is not backward compatible with versions 1.x of the package. All references to files from previous versions of the package should be removed to avoid issues.
It is recommended to start using from a new project.

Content

Folder: *Assets / Blartenix / Common*

List of elements that this package contains.

Documentation

Folder: *Assets / Blartenix / Common / Documentation*

Find the documentation for the package. PDF documents known as Module Reference Guide (MRG), in English and Spanish languages.

Prototyping

Folder: *Assets / Blartenix / Common / Prototyping*

This folder contains elements that can be used for a quick prototyping of mechanics that you want to do when designing video games.

- **Game Settings**
  Contains all the elements required by the user interface menu to control the basic game settings included in this package (GameSettings). To use this menu, you just need to drag the Game Settings Canvas prefab into your scene (Note that an Event System instance must exist in the scene)

You can use the content in this content to create your own settings menu.

➢ **Pictograms**

Contains a prefab that can be used for the TriggerPictogram and TriggerPictogram2D. You can use this prefab as a reference to build your own pictograms.

Scripts

Folder: *Assets / Blartenix / Common / Scripts*

Components

➢ **Billboard**

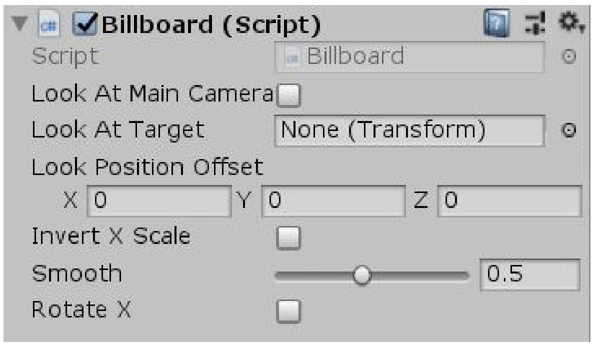
Allows a GameObject to always face a target point.

Serialized fields

- ➔ bool **Look At Main Camera**
  If the target point is the camera with the MainCamera tag.
- ➔ string **Look At Target**
  If the target point is not the main camera, another target Transform is set.
- ➔ Transform **Look Position Offset**
  Difference or delta of the position to look with respect to the target point.
- ➔ Transform **Invert X Scale**
  If the X factor of the scale needs to be inverted.
- ➔ float **Smooth**
  Range between 0.1 and 1. Sets the rotation speed to preserve the direction towards the target.
- ➔ bool **Rotate X**
  Whether the Object should also rotate on the X axis.

Methods

- ➔ void **SetTarget**(Transform target)
  Sets the target.



Billboard (Script)

Script	Billboard
Look At Main Camera	☐
Look At Target	None (Transform)
Look Position Offset	X 0 Y 0 Z 0
Invert X Scale	☐
Smooth	0.5
Rotate X	☐

Component inspector.

➢ CanvasEnabler

To enable or disable canvases and their elements quickly.

Serialized Fields

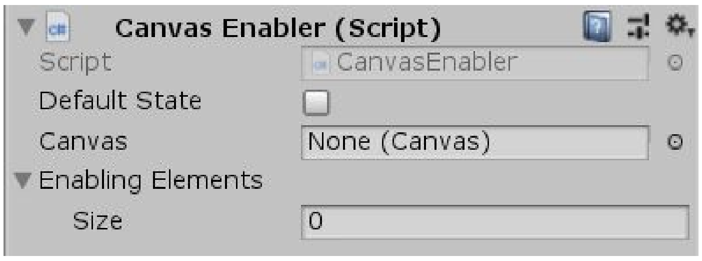
- ➔ bool **Default State**
  Initial state.

- ➔ Canvas Canvas **Canvas**
  component. If one is not specified, Awake captures the

- ➔ Behavior [] **Enabling Elements**
  component. The inner elements of the canvas that are enabled and disabled when the state is changed. (Buttons, images, etc)

Properties

- ➔ bool **enabled** {get; set; }
  Enable or disable the Canvas together with the Enabling Elements.


Canvas Enabler (Script)

Script	CanvasEnabler
Default State	☐
Canvas	None (Canvas)
Enabling Elements	
Size	0

Component inspector.

➢ TriggerPictogram

It allows you to show a pictogram when the OnTriggerEnter method is executed and / or when you want it to.

Serialized Fields

➔ CanvasEnabler[] **Pictograms**
Instance of the pictograms in world space of the scene.

➔ int **Default Pictogram**
Index of the default pictogram in the list of pictograms above.

➔ string [] **Triggering Tags**
List of Tags to detect and filter GameObjects in the OnTriggerEnter.

➔ LookForTagAt **Look For Tag At**
The GameObject to verify if it complies with the triggering tags, associated with the Collider that executes the Trigger.

➔ bool **Look At Camera**
Whether the pictogram should always face a camera.

➔ bool **Look At Main Camera**
Whether to look at the main camera (with the Main Camera tag). Only displayed if the value of the above property is true.

➔ Camera **Target Cam**
The target camera for the pictogram to look at. Only displayed if the value of the above property is false.

Properties

- ➔ bool **IsShowing** {get; private set; }
  If a pictogram is currently being displayed.

Methods

- ➔ void **HidePictogram**()
  Hides the currently displayed pictogram.

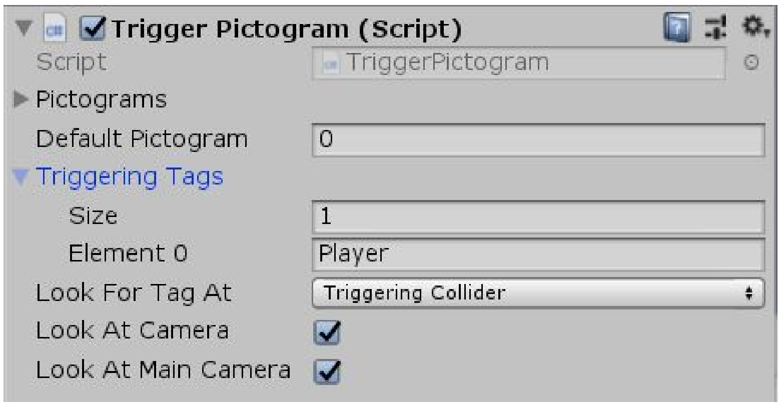
- ➔ void **ShowPictogram**()
  Shows the default pictogram.

- ➔ void **ShowPictogram**(int pictogramIndex)
  Shows the pictogram corresponding to the index of the glyph list, specified in the parameter.

- ➔ void **ShowPictogram**(int pictogramIndex, float hideTime)
  Shows the pictogram corresponding to the index specified in the parameter and automatically hides it after *'hideTime'* seconds.

- ➔ void **SetDefaultPictogram**(int pictogramIndex)
  Sets the default pictogram.

3D environment pictogram.

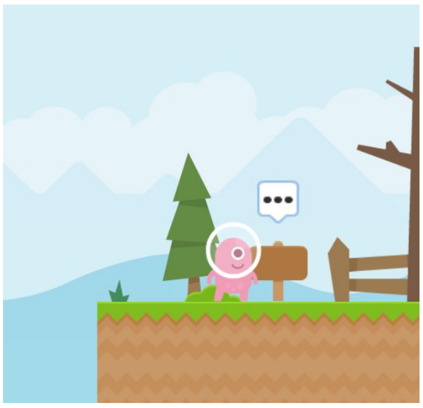


Trigger Pictogram (Script)	
Script	TriggerPictogram
Pictograms	
Default Pictogram	0
Triggering Tags	
Size	1
Element 0	Player
Look For Tag At	Triggering Collider
Look At Camera	✓
Look At Main Camera	✓

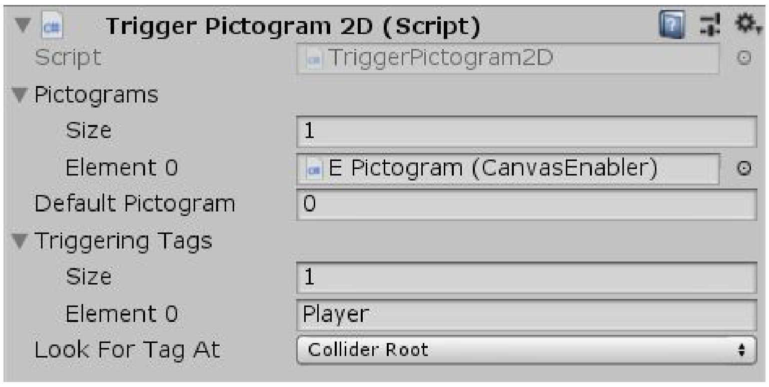
Component inspector.

➢ TriggerPictogram2D

Corresponds to the *TriggerPictogram* component with the difference that this is for the 2D environment, that is, for when the OnTriggerEnter2D is executed. It has the same serialized elements, properties, and methods.



2D environment pictogram.



Trigger Pictogram 2D (Script)

Script	TriggerPictogram2D
Pictograms	
Size	1
Element 0	E Pictogram (CanvasEnabler)
Default Pictogram	0
Triggering Tags	
Size	1
Element 0	Player
Look For Tag At	Collider Root

Component inspector.

➢ SingletonBehaviour <T>

Generic abstract component for the implementation of other elements that are handled under the Singleton design pattern.

Serialized fields

- ➔ bool **Dont Destroy On Load**

Properties

- static T **Instance** {get; private set; }
  Static instance of the type of the specified component.

Methods

- protected virtual void **OnAwake**()
  Method that overrides the Awake in the component that inherits from it and that you must override if you want to use the Awake method.
  You should not use the Awake

- protected virtual void **OnDestroyed**()
  method that replaces the OnDestroy in the component that inherits from it and that you must override if you want to use the OnDestroy method.
  The OnDestroy

➢ LanguageManager

Component in charge of managing and handling the languages of the game should not be used. Use singleton pattern and inherit from SingletonBehaviour<LanguageManager>

Serialized Fields

- bool **Dont Destroy On Load** (*SingletonBehaviour*)

- GameSettings **Game Settings**
  ScriptableObject of the game settings. *It's optional. Used to keep the current language selection persistent.*

- List <TextAsset> **Language Files**
  List of language xml files for the game.

- int **Selected Language**
  Index of the current language in the file list.

Properties

- Static LanguageManager **Instance** {get; private set; } (*SingletonBehaviour*)

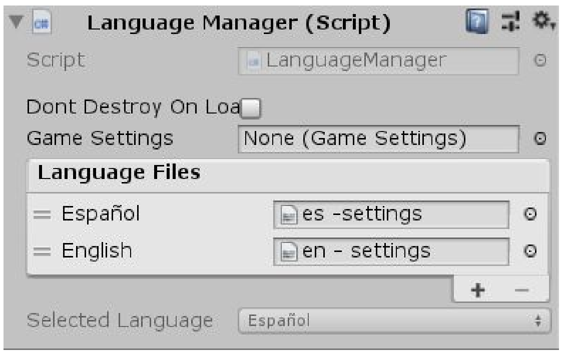
- int **SelectedLanguage** {get; }
  Index of the current language. Read only.

Methods

➔ void **SetLanguage**(int languageIndex)
Change the selected language by sending the index of the new language.

➔ IList <string> **GetLanguagesNames**()
Returns the list of language names.

➔ string **GetText**(string idName)
Returns the value of the text with the idName specified for the currently selected language.


Language Manager (Script)
Script: LanguageManager
Dont Destroy On Loa: ☐
Game Settings: None (Game Settings)
Language Files
Español: es -settings
English: en - settings
Selected Language: Español

Inspector of the component

➢ LanguageText

The component for the game texts that implement the dialog system.

Serialized Fields

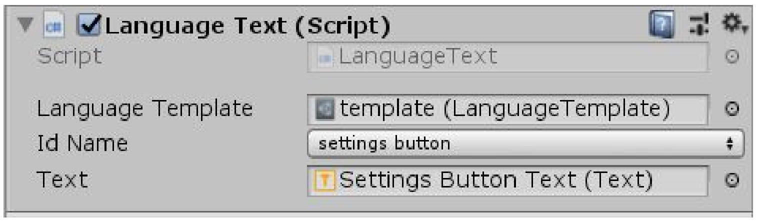
➔ LanguageTemplate **Language Template**
Template that contains the identifier of the text.

➔ string **Id Name**
identifier name

➔ TextText **Text**
Component type Text.

Properties

➔ string **Text** {get; }
Text or content of the associated Text component.


Language Text (Script)
Script: LanguageText
Language Template: template (LanguageTemplate)
Id Name: settings button
Text: Settings Button Text (Text)

Inspector

➢ ObjectPool <T>

ComponentBase abstract component for object pooling implementation. Type T must be of type Component.

Serialized fields

➔ T **Prefab**
The prefab.

➔ int **Initial Count**
The amount of instances in the pool when initialized.

➔ T [] **Preloaded**
Active or inactive instances in the scene to help or reduce pool initialization.

Methods

➔ void **InitPool**()
Initializes the pool. should be called in the Awake method.

➔ T **Get**(bool active = true)
Return an instance. by default active.

➔ T **Get**(Transform parent, bool active = true)
Return an instance as a child of transform specified. if null, with no parent.

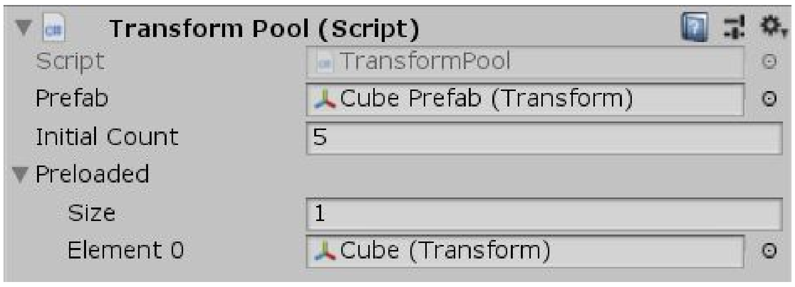
➔ T **Get**(Vector3 position, Quaternion rotation, bool active = true)
Returns an instance in the position with the rotation specified.

➔ T **Get**(Vector3 position, Quaternion rotation, Transform parent, bool active = true)
Returns an instance with the position, rotation and parent transform specified.

➔ void **Put**(T go)
Finalizes an instance and returns it to the pool.

➢ TransformPool

It is the implementation of a pool for objects of type transform. It inherits from ObjectPool<Transform> so its serialized fields and methods are those defined in ObjectPool <T> and T (the type) is Transform.



Transform Pool (Script)

Script	TransformPool
Prefab	Cube Prefab (Transform)
Initial Count	5
Preloaded	
Size	1
Element 0	Cube (Transform)

Componente inspector.

➢ MultipleObjectsPool <T>

Basic abstract component for the implementation of a pool of objects of the same type with different characteristics. Type T must be of type Component.

Serialized fields

➔ T [] **Prefabs**
The prefabs of the game objects for the pool

➔ T [] **Preloaded**
Active or inactive instances in the scene to be preloaded to help performance.

Methods

➔ void **InitPool**()
Initializes the pool. should be called in the Awake method.

➔ T **Get**(Predicate <T> predicate, bool active = true)
Find and return an instance based on a predicate query. By default active. if no instance is found returns null.

➔ T **Get**(Predicate <T> predicate, Transform parent, bool active = true)
Find and return an instance as a child of the transform specified. if null, then with no parent. return null if an instance is not found.

➔ T **Get**(Predicate <T> predicate, Vector3 position, Quaternion rotation, bool active = true)

Find and return an instance with the position and rotation specified.

- ➔ T **Get**(Predicate <T> predicate, Vector3 position, Quaternion rotation, Transform parent, bool active = true)
  Find and return an instance with the position, rotation and parent game object specified.

- ➔ void **Put**(T go)
  Finalizes an instance and returns it to the pool.

Core

Scriptable Objects

- ➢ **GameSettings**

  Object for managing some basic settings for your game. It is kept persistent in the PlayerPrefs for the game.

  Create → Blartenix → Game Settings

Serialized fields

- ➔ AudioMixer **Game Audio Mixer**
  Audio Mixer of the game

- ➔ string **Music Volume Param Name**
  Name of the parameter displayed in the audio mixer for managing the volume of the music.

- ➔ string **Sfx Volume Param Name**
  Name of the parameter displayed in the audio mixer for controlling the volume of the sound effects.

- ➔ float **Default Music Volume Default**
  volume of the music.

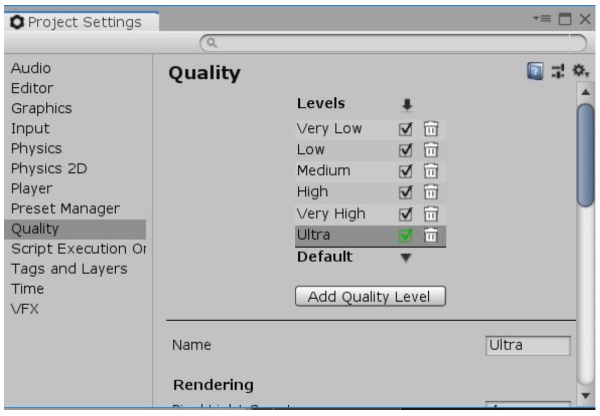
- ➔ float **Default SFX Volume Default**
  volume for sound effects.

Properties

- ➔ int **Language** {get; set; }
  Index of the currently selected language.

- ➔ int **Graphics** {get; set; }

Currently selected graphics quality index. This list is defined in the Project Settings.



Name Ultra

➔ int **Resolution** {get; set; }
Index of the current resolution. This list is obtained from the resolutions supported by the current screen.

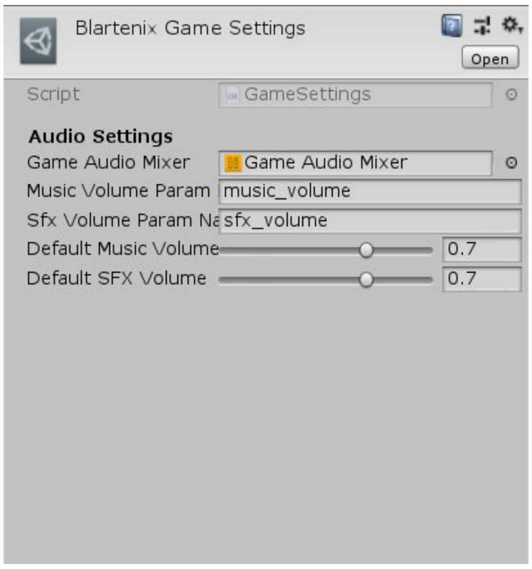
➔ bool **Fullscreen** {get; set; }
Full screen mode.

➔ float **MusicVolume** {get; set; }
Music volume.

➔ float **SfxVolume** {get; set; }
Volume of sound effects.

Methods

➔ void **ResetValues**()
Clears the player pref. restoring the default values for the configurations.


Blartenix Game Settings
Open
Script GameSettings
Audio Settings
Game Audio Mixer: Game Audio Mixer
Music Volume Param: music_volume
Sfx Volume Param Na: sfx_volume
Default Music Volume: 0.7
Default SFX Volume: 0.7

Inspector

➢ **LanguageTemplate**

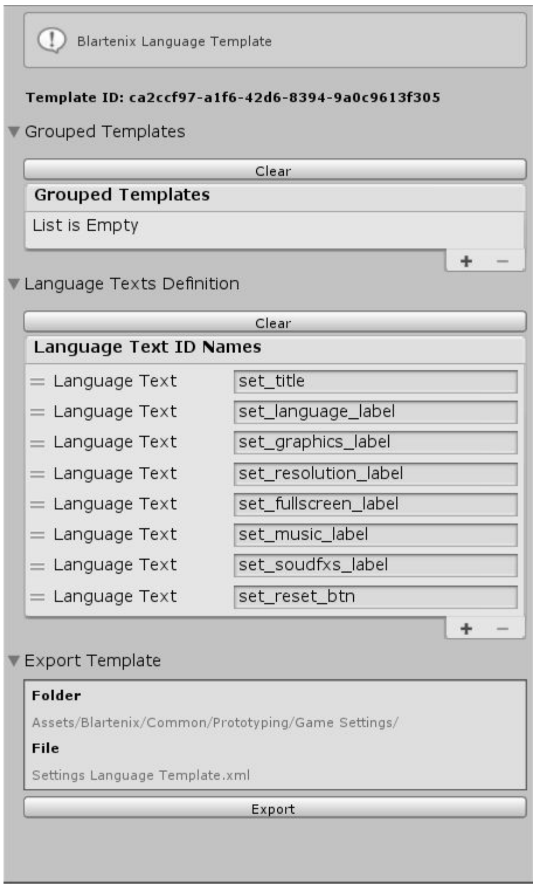
Object that contains the definition of all the texts of a language and that also allows generating the xml files of the languages used by the LanguageManager.

Create → Blartenix → Language Template

Serialized fields

➔ LanguageTemplate [] **Grouped Templates**
List with other templates grouped to associate their identifiers to the texts.

➔ string [] **Language Text Id Names**
List of identifiers for the template texts.


Blartenix Language Template

Template ID: ca2ccf97-a1f6-42d6-8394-9a0c9613f305

Grouped Templates

Clear

Grouped Templates

List is Empty

Language Texts Definition

Clear

Language Text ID Names

Language Text	set_title
Language Text	set_language_label
Language Text	set_graphics_label
Language Text	set_resolution_label
Language Text	set_fullscreen_label
Language Text	set_music_label
Language Text	set_soudfxs_label
Language Text	set_reset_btn

Export Template

Folder
Assets/Blartenix/Common/Prototyping/Game Settings/
File
Settings Language Template.xml

Export

Inspector of the language template.

Classes

➢ ***BlartenixLoggerStatic***

class that contains the global instance of the Blartenix library logger for Unity.

Properties

➔ Static IBlartenixLogger **GlobalInstance** {get; set; }
Global static instance of the Blartenix logger.

➢ ***Utilities***

Static class with utility methods used in the Blartenix Library for Unity.

Methods

➔ string **EncodeString(**string decodedString**)**
Encodes a string in its representation base64.

- ➔ string **DecodeString(**string encodedString**)**
  Decodes a base64 string to a string.

- ➔ T **DeserializeXML <T> (**string xml, bool xmlIsAFile**)**
  Deserializes an XML to its structured data representation. Receive an xml as a string or as a file specified in the *xmlIsAFile* parameter.

➢ ***BlartenixLanguage***

Structured definition of a language.

Serialized fields

- ➔ string **Template ID**
  identifier of the template associated with the language.

- ➔ string **Name**
  Name of the language.

- ➔ List <LanguageXmlTag> **Language Texts**
  List of the texts defined for the language.

➢ ***LanguageXmlTag***

Definition of the xml tag for language texts.

Serialized fields

- ➔ string **ID Name**
  Identifying name of the text.

- ➔ string **Value**
  Value of the text.

EnumsBlartenix

➢ ***LogType***

log message type.

Types

- ➔ *Info*
- ➔ *Warning*
- ➔ *Error*

➢ ***LookForTagAt***

Used for interaction with OnTrigger methods to indicate where to look for the specified tag.

Types

➔ *TriggeringCollider*
➔ *ColliderParent*
➔ *ColliderRoot*

Interfaces

➢ ***IBlartenixLogger***

Interface that defines the logging class for Blartenix.

Methods

➔ void **DisplayMessage**(string message, LogType type = LogType.Info)
Method to display log messages. Designed to record log messages during the development stage or directly in the application.

➔ **Log**(string message, LogType type = LogType.Info, [*CallerMemberName*] string callMember = null, [*CallerFilePath*] string file = null, [*CallerLineNumber*] int codeLine = -1)
Method to record a log message. Mainly intended to record log messages in external files at runtime.

The *callMember*, *file* and *codeLine parameters* are sent automatically when the method is invoked, for this reason they are optional.

References

➔ Microsoft docs. CallerFilePathAttribute

➔ Microsoft docs. CallerLineNumberAttribute

➔ Microsoft docs. CallerMemberNameAttribute

➔ Microsoft docs. Preach <T>

➔ Tutorials on our Youtube channel. You can find videos and playlists with demos.

Thanks for the support!