Namespace Cobilas

Classes

Filter by title

Cobilas (Cobilas.html)

Utility static class to obtain type or assembly.

INullObject (Cobilas.INullObject.html)

Interrupter (Cobilas.Interrupter.html)

NullObject (Cobilas.NullObject.html)

TypeUtilitarian (Cobilas.TypeUtilitarian.html)

Represents a list of switches.

+ Cobilas.Collections (Cobilas.Collections.html)

+ Cobilas.Collections.Generic (Cobilas.Collections.Generic.ht

+ Cobilas.IO.Atlf (Cobilas.IO.Atlf.html)

+ Cobilas.IO.Atlf.Components (Cobilas.IO.Atlf.Components.ht

+ Cobilas.IO.Atlf.Text (Cobilas.IO.Atlf.Text.html)

+ Cobilas.IO.Serialization.Binary (Cobilas.IO.Serialization.Binary

+ Cobilas.IO.Serialization.Json (Cobilas.IO.Serialization.Json.h

+ Cobilas.Numeric (Cobilas.Numeric.html)

+ Cobilas.Numeric.Convert (Cobilas.Numeric.Convert.html

+ Cobilas.Threading.Tasks (Cobilas.Threading.Tasks.html)

+ System (System.html)

+ System.Collections.Generic

Structs

Interfaces

Cobilas Core (com.cobilas.cs.lib.core.net4x.api/Cobilas.html)

Descripition

Cobilas Core Net4x is a utility library for CSharp.

Json

(namespace:Cobilas.IO.Serialization.Json)
Only present in the NuGet version.
The static class `Json` grants static read and write functions.

JsonContractResolver

Used by `JsonSerializer` to resolve a `JsonContract` for a given `Type`. Furthermore, `JsonContractResolver` determines how the fields of an `Object` will be serialized.

ATLF(Arquivo de tradução de leitura facil)

ATLF (Easy to Read Translation File) can be used to create and load translations for apps.

```
#>Header
The use of the header is not mandatory.<#
#! version:/*std:1.0*/
#! encoding:/*utf-8*/

#> Comment <#
#> ATLF format(1.0) <#

#> Uni-line marking <#
#! Tag1:/*value1*/

#> Multi-line marking <#
#! Tag2:/*
value1
value2
value3
value4
*/
```

How to read ATLF

```
static void Main(string[] args) {
    using ATLFReader reader = ATLFReader.Create(@"C:\folder1\file.txt");
    reader.Reader();
    Console.WriteLine($"tag.value.1:{reader.GetTag("tag.value.1")}");
    Console.WriteLine($"tag.value.2:{reader.GetTag("tag.value.2")}");
    Console.WriteLine($"tag.value.3:{reader.GetTag("tag.value.3")}");
}
```

The other reading functions.

- The `ATLFNode[]:ATLFReader.GetHeader()` function allows you to get the header tags.
- The `ATLFNode[]:ATLFReader.GetAllComments()` function allows you to get all comments. The `ATLFNode[]:ATLFReader.GetTagGroup(string path)` function allows you to obtain tags that belong to the same path.

```
/*C:\folder1\file.txt
* #! version:/*std:1.0* /
* #! encoding:/*utf-8* /
*
* #! tag.value.cop1:/*value1* /
* #! tag.value.map.cop1:/*value1* /
* #! tag.value.map.cop2:/*value1* /
* #! tag.value.cop2:/*value1* /
* #! tag.value.cop3:/*value1* /
*/
static void Main(string[] args) {
    using ATLFReader reader = ATLFReader.Create(@"C:\folder1\file.txt");
    reader.Reader();
    foreach(var item in reader.GetTagGroup("tag.value.map"))
        Console.WriteLine(item);
}
```

How to write ATLF

```
static void Main(string[] args) {
    using ATLFWriter writer = ATLFWriter.Create(File.OpenWrite(@"C:\folder1\file.txt"));
    writer.WriteHeader();//The header is not mandatory but if you add a header, call this func
tion first.
    writer.WriteComment("my tag1");
    writer.WriteNode("tag1", "value1");
    writer.WriteWhitespace("\r\n");//This function is called automatically when the `Indent` p
roperty is `true`. By default the `Indent` property is `true`.
    writer.WriteComment("my tag2");
    writer.WriteNode("tag2", "value2");
    writer.WriteWhitespace(2, "\r\n");//This function is called automatically when the `Indent
` property is `true`. By default the `Indent` property is `true`.
    writer.WriteComment("my tag3");
    writer.WriteNode("tag3", "value3");
}
```

Encoders and decoders

Regarding encoders and decoders, ATLF allows the creation of customized encoders and decoders.
To use a custom encoder or decoder, assign a version to your custom encoder or decoder using the `Version` property and then assign the version of the custom encoder or decoder in the `TargetVersion` property of the `ATLFWriter` and `ATLFReader` classes.

Creating a custom encoding class

To create a custom encoding class, the class must inherit the `ATLFVS10Encoding` class.

Creating a custom decoding class

To create a custom decoding class, the class must inherit the `ATLFVS10Decoding` class.

Cobilas.Core.Net4x (https://www.nuget.org/packages/Cobilas.Core.Net4x) is on nuget.org

To include the package, open the `.csproj` file and add it.

```
<ItemGroup>
  <PackageReference Include="Cobilas.Core.Net4x" Version="2.7.0" />
</ItemGroup>
```

Or use command line.

```
dotnet add package Cobilas.Core.Net4x --version 2.7.0
```

Namespace Cobilas.GodotEditor.Utility.Serialization

Filter by title

Classes

- **Cobilas.GodotEditor.Utility.Serialization**

BuildSerialization (Cobilas.GodotEditor.Utility.Serialization.BuildSerialization.html)

Class allows to build a serialization list of properties of a node class.

BuildSerialization (Cobilas.GodotEditor.Utility.Serializa

HidePropertyAttribute (Cobilas.GodotEditor.Utility.Serialization.HidePropertyAttribute.html)

The attribute allows you to hide and save the value of a field or property in the editor.

MemberItem (Cobilas.GodotEditor.Utility.Serializa

MemberItem (Cobilas.GodotEditor.Utility.Serialization.MemberItem.html)

Represents a property or field.

PropertyItem (Cobilas.GodotEditor.Utility.Serializa

PropertyItem (Cobilas.GodotEditor.Utility.Serialization.PropertyItem.html)

The class stores the information for drawing in the editor.

SNInfo (Cobilas.GodotEditor.Utility.Serializa

SerializationCache (Cobilas.GodotEditor.Utility.Serialization.SerializationCache.html)

Class to handle property caching.

SerializeFieldAttribute (Cobilas.GodotEditor.Utility.Serializa

SerializeFieldAttribute (Cobilas.GodotEditor.Utility.Serialization.SerializeFieldAttribute.html)

Base attribute for field and property serialization attributes.

ShowPropertyAttribute (Cobilas.GodotEditor.Utility.Serializa

ShowPropertyAttribute (Cobilas.GodotEditor.Utility.Serialization.ShowPropertyAttribute.html)

The attribute allows you to show a field or property in the editor.

ShowRangePropertyAttribute (Cobilas.GodotEditor.Utility.Serializa

ShowRangePropertyAttribute (Cobilas.GodotEditor.Utility.Serialization.ShowRangePropertyAttribute.html)

The attribute allows you to display a field or property in the editor in range form.

- **Cobilas.GodotEditor.Utility.Serialization.Hints**
- **Cobilas.GodotEditor.Utility.Serialization.Properties**
- **Cobilas.GodotEditor.Utility.Serialization.RenderObjects**
- **Cobilas.GodotEngine.Utility**
- **Cobilas.GodotEngine.Utility.IO**

Structs

SNInfo (Cobilas.GodotEditor.Utility.Serialization.SNInfo.html)

Represents the information of a SerializedNode (Cobilas.GodotEditor.Utility.Serialization.RenderObjects.SerializedNode.html).

Interfaces

ISerializedPropertyManipulation (Cobilas.GodotEditor.Utility.Serialization.ISerializedPropertyManipulation.html)

The interface allows property manipulation.

- Cobilas.GodotEditor.Utility.Serialization (Cobilas.GodotEditor.Utility.Ser
  - BaseSerialization (Cobilas.GodotEditor.Utility.Serializ
  - HidePropertyAttribute (Cobilas.GodotEditor.Utility.Serializa
  - ISerializedPropertyManipulation (Cobilas.GodotEditor.Utility.Serializa
  - MemberItem (Cobilas.GodotEditor.Utility.Serializa
  - PropertyItem (Cobilas.GodotEditor.Utility.Serializa
  - SNInfo (Cobilas.GodotEditor.Utility.Serializa
  - SerializationCache (Cobilas.GodotEditor.Utility.Serializa
  - SerializeFieldAttribute (Cobilas.GodotEditor.Utility.Serializa
  - ShowPropertyAttribute (Cobilas.GodotEditor.Utility.Serializa
  - ShowRangePropertyAttribute (Cobilas.GodotEditor.Utility.Serializa
- Cobilas.GodotEditor.Utility.Serialization.Hints (Cobilas.GodotEditor.Utility.Ser
- Cobilas.GodotEditor.Utility.Serialization.Properties (Cobilas.GodotEditor.Utility.Ser
- Cobilas.GodotEditor.Utility.Serialization.RenderObjects (Cobilas.GodotEditor.Utility.Ser
- Cobilas.GodotEngine.Utility (Cobilas.GodotEngine.Utility.ht
- Cobilas.GodotEngine.Utility.IO (Cobilas.GodotEngine.Utility.IO

Cobilas Godot Utility (https://belicusbr.github.io/com.cobilas.docs/mds/gd-utility-getting-started.html)

Descripition

The package contains utility classes in csharp for godot engine(Godot3.5)

RunTimeInitialization

(namespace: Cobilas.GodotEngine.Utility.Runtime)
The `RunTimeInitialization` class allows you to automate the `Project>Project Settings>AutoLoad` option.
To use the `RunTimeInitialization` class, you must create a class and make it inherit `RunTimeInitialization`.

```
using Cobilas.GodotEngine.Utility.Runtime;
//The name of the class is up to you.
public class RunTimeProcess : RunTimeInitialization {}
```

And remember to add the class that inherits `RunTimeInitialization` in `Project>Project Settings>AutoLoad`.
Remembering that the `RunTimeInitialization` class uses the virtual method `_Ready()` to perform the initialization of other classes.
And to initialize other classes along with the `RunTimeInitialization` class, the class must inherit the `Godot.Node` class or some class that inherits `Godot.Node` and use the `RunTimeInitializationClassAttribute` attribute.

```
using Godot;
using Cobilas.GodotEngine.Utility.Runtime;
[RunTimeInitializationClass]
public class ClassTest : Node {}
```

RunTimeInitializationClass

```
/*
bootPriority: Represents the boot order
{ (enum Priority)values
        StartBefore,
        StartLater
}
name:The name of the object
subPriority: And the execution priority order.
*/
//RunTimeInitializationClassAttribute(string? name, Priority bootPriority = Priority.StartBefo
re, int subPriority = 0, bool lastBoot = false)
[RunTimeInitializationClassAttribute(string?, [Priority:Priority.StartBefore], [int:0], [bool:
false])]
[RunTimeInitializationClass()]
```

CoroutineManager

The `CoroutineManager` class is responsible for creating and managing coroutines for godot.
How to create a coroutine?

```
using Godot;
using System.Collections;
using Cobilas.GodotEngine.Utility;

public class ClassTest : Node {
        private Coroutine coroutine;
        public override void _Ready() {
                coroutine = CoroutineManager.StartCoroutine(Corroutine1());
                coroutine = CoroutineManager.StartCoroutine(Corroutine2());
                coroutine = CoroutineManager.StartCoroutine(Corroutine3());
        }

        private IEnumerator Corroutine1() {
                GD.Print("Zé da manga");
                //When the return is null, by default the coroutine is executed as _Process().
                yield return null;
        }

        private IEnumerator Corroutine2() {
                GD.Print("Zé da manga");
                //When the return is RunTimeSecond the coroutine is executed as _Process() wit
h a pre-defined delay.
                yield return new RunTimeSecond(3);
        }

        private IEnumerator Corroutine3() {
                GD.Print("Zé da manga");
                When the return is RunTimeSecond the coroutine is executed as _PhysicProcess()
with a pre-defined delay.
                yield return new FixedRunTimeSecond(3);
        }
}
```

With the `IYieldVolatile` interface you can switch coroutine execution between `_Process(float)` and `_PhysicsProcess(float)`.

IYield Classes

- RunTimeSecond is a framework that allows you to delay your coroutine in seconds. This class inherits `IYieldUpdate`.
- FixedRunTimeSecond is a framework that allows you to delay your coroutine in seconds. This class inherits `IYieldFixedUpdate`.
- IYieldUpdate is an interface that allows the coroutine to run in the `_Process(float)` function.
- IYieldFixedUpdate is an interface that allows the coroutine to run in the `_PhysicsProcess(float)` function.
- IYieldVolatile is an interface that allows the coroutine to run in the `Process(float)` or `_PhysicsProcess(float)` function.
- IYieldCoroutine is the base interface for Yield interfaces.

Stop coroutines

Now to stop a coroutine.

```
public static void StopCoroutine(Coroutine Coroutine);
public static void StopAllCoroutines();
```

SerializedPropertyCustom

Now a class has been added for custom serialization of properties in the Godot inspector.
With the `HideProperty` and `ShowProperty` attributes you can serialize properties in the Godot inspector.

Example

Below is an example of usage.

```
public class Exe1 : Node {
        [ShowProperty] string var1;
        [ShowProperty] string var2;
        [ShowProperty] string var3;
        //The property will not be shown but its value will be saved.
        [HideProperty] string var4;
        [ShowProperty] vec2d var5;

        public override GDArray _GetPropertyList() => SerializedNode.GetPropertyList(BuildSeri
alization.Build(this).GetPropertyList());
        public override bool _Set(string property, object value) => BuildSerialization.Build(t
his).Set(property, value);
        public override object _Get(string property) => BuildSerialization.Build(this).Get(pro
perty);
}
[Serializable]
public struct vec2d {
        //When ShowProperty or HideProperty has its parameter true the value will be saved in
cache.
        [ShowProperty(true)] public float x;
        [ShowProperty(true)] public float y;
}
```

The Cobilas Godot Utility (https://www.nuget.org/packages/Cobilas.Godot.Utility/) is on nuget.org

To include the package, open the `.csproj` file and add it.

```
<ItemGroup>
        <PackageReference Include="Cobilas.Godot.Utility" Version="6.2.3" />
</ItemGroup>
```

Or use command line.

```
dotnet add package Cobilas.Godot.Utility --version 6.2.3
```

Namespace Cobilas.GodotEngine.Component

Filter by title

Classes

- **Cobilas.GodotEngine.Component**
  - **InternalComponentHub (Cobilas.GodotEngine.Component.InternalComponentHub.html)**
    Inner class for handling IComponentHub (Cobilas.GodotEngine.Component.IComponentHub.html).
  - **NullComponentHub (Cobilas.GodotEngine.Component.NullComponentHub.html)**
    Represents a null ComponentHub.
  - **RequireComponentAttribute (Cobilas.GodotEngine.Component.RequireComponentAttribute.html)**
    Signals to the AddRequireComponent(Node?) (Cobilas.GodotEngine.Component.InternalComponentHub.html#Cobilas_GodotEngine_Component_InternalComponentHub_AddRequireComponent_Godot_Node_) method which components to add to the Godot.Node object.

Interfaces

IComponentHub (Cobilas.GodotEngine.Component.IComponentHub.html)

An interface to transform a Godot.Node object into a pseudo Component.

IInternalComponentHub (Cobilas.GodotEngine.Component.IInternalComponentHub.html)

Interface for inner class for handling IComponentHub (Cobilas.GodotEngine.Component.IComponentHub.html).

Cobilas Godot IComponent

This package aims to transform a `Node` object into a pseudo-component in the style of the `Unity Engine` to facilitate obtaining objects and adding child objects.

Usage

To use the `IComponentHub` interface, it must be inherited by a `Node` object, and you can also use the serializable `InternalComponentHub` class to automate the addition, removal, and retrieval of child `Node` objects. However, implementing the `IComponentHub` interface manually will only require a little work.

Exemplo

```
using Godot;
using Cobilas.GodotEngine.Component;

public class MonoNode : Node, IComponentHub {
    private InternalComponentHub components;

    public Node Parent => components.Parent;
    public int ComponentsCount => components.ComponentsCount;
    public IComponentHub ParentComponent => components.ParentComponent;

    public Node AddComponent(Type component) => components.AddComponent(component);
    public T AddComponent<T>() where T : Node => ((IComponentHub)components).AddComponent<T>
();
    public void AddComponents(params Type[] components) => this.components.AddComponents(compo
nents);
    public void AddNodeComponent(Node component) => components.AddNodeComponent(component);
    public void AddNodeComponents(params Node[] components) => this.components.AddNodeComponen
ts(components);
    public Node GetComponent(Type component) => components.GetComponent(component);
    public Node GetComponent(Type component, bool recursive) => components.GetComponent(compon
ent, recursive);
    public T GetComponent<T>() where T : Node => ((IComponentHub)components).GetComponent<T>
();
    public T GetComponent<T>(bool recursive) where T : Node => ((IComponentHub)components).Get
Component<T>(recursive);
    public Node[] GetComponents(Type component) => components.GetComponents(component);
    public Node[] GetComponents(Type component, bool recursive) => components.GetComponents(co
mponent, recursive);
    public T[] GetComponents<T>() where T : Node => ((IComponentHub)components).GetComponents<
T>();
    public T[] GetComponents<T>(bool recursive) where T : Node => ((IComponentHub)components).
GetComponents<T>(recursive);
    public IEnumerator<Node> GetEnumerator() => components.GetEnumerator();
    public bool RemoveComponent(Node component) => components.RemoveComponent(component);
    public void RemoveComponents(params Node[] components) => this.components.RemoveComponents
(components);
    IEnumerator IEnumerable.GetEnumerator() => ((IEnumerable)components).GetEnumerator();

    protected override void Dispose(bool disposing) {
        base.Dispose(disposing);
        components?.Dispose();
    }
}
```

The Cobilas Godot IComponent (https://www.nuget.org/packages/Cobilas.Godot.IComponent/) is on nuget.org

To include the package, open the `.csproj` file and add it.

```
<ItemGroup>
        <PackageReference Include="Cobilas.Godot.IComponent" Version="1.1.1" />
</ItemGroup>
```

Or use command line.

```
dotnet add package Cobilas.Godot.IComponent --version 1.1.1
```

Cobilas.Godot.Lua

Quick Start

Executing Lua Code

```
//tds
// Create a Lua container with default configuration
var config = LuaContainerConfg.Default;
using var lua = new LuaContainer(config);

// Build and execute Lua code
lua.DoString("print('Hello from Lua!')")
   .InitField("playerName", "John Doe")
   .InitFunction("greet", "print('Hello, ' .. playerName)", "playerName")
   .FlushToLua();

// Invoke the function
lua.InvokeFunction("greet");
```

Working with Lua Files

```
// Load and execute a Lua script file
var fileConfig = new LuaFileConfg("path/to/script.lua");
using var luaFile = new LuaFile(fileConfig);

// Access fields from the Lua script
var playerHealth = luaFile.GetField("player.health");
int healthValue = (int)playerHealth;

// Invoke functions defined in the script
var result = luaFile.InvokeFunction("calculateDamage", 10, 2.5f);
```

Creating Lua Tables

```
// Create complex Lua table structures
var playerTable = new LuaTableItem("player",
    new LuaTableValue("name", "Hero"),
    new LuaTableValue("health", 100),
    new LuaTableValue("level", 5),
    new LuaTableItem("inventory",
        new LuaTableValue("sword", 1),
        new LuaTableValue("potion", 3)
    )
);

var config = LuaContainerConfg.Default;
using var lua = new LuaContainer(config);
lua.InitTable(playerTable).FlushToLua();
```

Core Components

Configuration

- `LuaContainerConfg` - Configuration for in-memory Lua execution
- `LuaFileConfg` - Configuration for file-based Lua scripts

Main Classes

- `LuaContainer` - Dynamic Lua code builder and executor
- `LuaFile` - File-based Lua script manager
- `LuaField` - Type-safe Lua field access with conversion capabilities
- `LuaTableItem` & `LuaTableValue` - Lua table structure builders

Interfaces

- `ILuaFile` - Core operations for Lua file interaction
- `ILuaTable` & `ILuaTableItem` - Lua table element contracts

Advanced Usage

Custom Serialization

```
[LuaSerializable(typeof(PlayerData))]
public class PlayerDataConverter : ObjectToLuaTable
{
    public override void ToLuaTable(object obj, NLua.LuaTable table)
    {
        var player = (PlayerData)obj;
        table["name"] = player.Name;
        table["level"] = player.Level;
    }

    public override object ToObject(object obj, NLua.LuaTable table)
    {
        return new PlayerData
        {
            Name = (string)table["name"],
            Level = (int)table["level"]
        };
    }
}

// Usage
var playerData = new PlayerData { Name = "Warrior", Level = 10 };
lua.SetField("player", playerData);
```

CLR Integration

```
var config = new LuaContainerConfg(useCLRPackage: true);
using var lua = new LuaContainer(config);

lua.InitCLRPackage("System.Math")
   .DoString("print('PI value: ' .. Math.PI)")
   .FlushToLua();
```

API Documentation

Comprehensive XML documentation is included with the library. Key methods include:

- `GetField()` / `SetField()` - Access and modify Lua variables
- `InvokeFunction()` - Call Lua functions with parameters
- `LuaTableToObject<T>()` - Convert Lua tables to C# objects
- `InitFunction()` - Define Lua functions programmatically
- `FlushToLua()` - Execute accumulated Lua code

Examples

Check the `Examples/` folder for complete usage examples:

- Basic Lua execution
- Game data configuration
- Save/load system implementation
- Custom object serialization

The Cobilas Godot Lua (https://www.nuget.org/packages/Cobilas.Godot.Lua/) is on nuget.org

To include the package, open the `.csproj` file and add it.

```
<ItemGroup>
        <PackageReference Include="Cobilas.Godot.Lua" Version="1.2.3" />
</ItemGroup>
```

Or use command line.

```
dotnet add package Cobilas.Godot.Lua --version 1.2.3
```