C# API

LuaEnv 类

```
object[] DoString(string chunk, string chunkName = "chuck", LuaTable env = null)
```

描述：

执行一个代码块。

参数：

chunk: Lua 代码文字串；

chunkName： 发生 error 时的 debug 显示信息中使用，指明某某代码块的某行错误；

env ：这个代码块的环境变量；

返回值：

代码块里 return 语句的返回值；

比如：return 1, “hello”，DoString 返回将包含两个 object 的数组， 一个是 double 类型的 1， 一个是 string 类型的“hello”

例子：

```
LuaEnv luaenv = new LuaEnv();
object[] ret = luaenv.DoString("print( 'hello' )\r\nreturn 1")
UnityEngine.Debug.Log("ret="+ret[0]);
luaenv.Dispose()
```

```
T LoadString<T>(string chunk, string chunkName = "chunk", LuaTable env = null)
```

描述：

加载一个代码块，但不执行，只返回类型可以指定为一个 delegate 或者一个 LuaFunction

参数：

chunk: Lua 代码的字符串；

chunkName： 发生 error 时的 debug 显示信息中使用，指明某某代码块的某行错误；

env ：这个代码块的环境变量；

返回值：

代表该代码块的 delegate 或者 LuaFunction 类；

```
LuaTable Global;
```

描述：

代表 lua 全局环境的 LuaTable

```
void Tick()
```

描述：

清除 Lua 的未手动释放的 LuaBase 对象（比如：LuaTable， LuaFunction），以及其它一些事情。

需要定期调用，比如在 MonoBehaviour 的 Update 中调用。

```
void AddLoader(CustomLoader loader)
```

描述：

增加一个自定义 loader

参数：

loader：一个包含了加载函数的委托，其类型为 `delegate byte[] CustomLoader(ref string filepath)`，当一个文件被 require 时，这个 loader 会被回调，其参数是调用 require 所使用的参数，如果该 loader 找到文件，可以将其读进内存，返回一个 byte 数组。如果需要支持调试的话，而 filepath 要设置成 IDE 能找到的路径（相对或者绝对都可以）

```
void Dispose()
```

描述：

Dispose 该 LuaEnv。

LuaEnv 的使用建议：全局就一个实例，并在 Update 中调用 GC 方法，完全不需要时调用 Dispose

LuaTable 类

```
T Get<T>(string key)
```

描述：

获取在 key 下，类型为 T 的 value，如果不存在或者类型不匹配，返回 null；

```
T GetInPath<T>(string path)
```

描述：

和 Get 的区别是，这个函数会识别 path 里头的“.”，比如 var i = tbl.GetInPath<int>(“a.b.c”) 相当于在 lua 里头执行 i = tbl.a.b.c，避免仅为了获取中间变量而多次调用 Get，执行效率更高。

```
void SetInPath<T>(string path, T val)
```

描述：

和 GetInPaht<T>对应的 setter；

```
void Get<TKey, TValue>(TKey key, out TValue value)
```

描述：

上面的 API 的 Key 都只能是 string，而这个 API 无此限制；

```
void Set<TKey, TValue>(TKey key, TValue value)
```

描述：

对应 Get<TKey, TValue>的 setter；

```
T Cast<T>()
```

描述：

把该 table 转成一个 T 指明的类型，可以是一个加了 CSharpCallLua 声明的 interface，一个有默认构造函数的 class 或者 struct，一个 Dictionary，List 等等。

```
void SetMetaTable(LuaTable metaTable)
```

描述：

设置 metaTable 为 table 的 metatable

LuaFunction 类

注意：用该类访问 Lua 函数会有 boxing，unboxing 的开销，为了性能考虑，需要频繁调用的地方不要用该类。建议通过 table.Get<ABCDelegate>获取一个 delegate 再调用（假设 ABCDelegate 是 C#的一个 delegate）。在使用使用 table.Get<ABCDelegate>之前，请先把 ABCDelegate 加到代码生成列表。

```
object[] Call(params object[] args)
```

描述：

以可变参数调用 Lua 函数，并返回该调用的返回值。

```
object[] Call(object[] args, Type[] returnTypes)
```

描述：

调用 Lua 函数，并指明返回参数的类型，系统会自动按指定类型进行转换。

```
void SetEnv(LuaTable env)
```

描述：

相当于 lua 的 setfenv 函数。

Lua API

CS 对象

CS.namespace.class(...)

描述：调用一个 C#类型的构造函数并返回实例对象，例如：

```
local v1=CS.UnityEngine.Vector3(1,1,1)
```

CS.namespace.class.field

描述：访问一个 C#静态成员，例如：

```
Print(CS.UnityEngine.Vector3.one)
```

CS.namespace.enum.field

描述：访问一个枚举值

typeof 函数

类似 C#里头的 typeof 关键字，返回一个 Type 对象，比如 GameObject.AddComponent 其中一个重载需要一个 Type 参数，这时可以这么用

```
newGameObj:AddComponent(typeof(CS.UnityEngine.ParticleSystem))
```

无符号 64 位支持

uint64.tostring

描述：无符号数转字符串。

uint64.divide

描述：无符号数除法。

uint64.compare

描述：无符号比较，相对返回 0，大于返回正数，小于返回负数。

uint64.remainder

描述：无符号数取模。

uint64.parse

描述：字符串转无符号数。

xlua.structclone

描述：克隆一个 c#结构体

cast 函数

指明以特定的接口访问对象，这在实现类无法访问的时候（比如 internal 修饰）很有用，这时可以这么来（假设下面的 calc 对象实现了 C#的 PerformentTest.ICalc 接口）：

```
cast(calc, typeof(CS.PerformentTest.ICalc))
```

然后就木有其它 API 了

访问 csharp 对象和访问一个 table 一样，调用函数跟调用 lua 函数一样，也可以通过操作符访问 c#的操作符，下面是一个例子：

```
local v1=CS.UnityEngine.Vector3(1,1,1)
local v2=CS.UnityEngine.Vector3(1,1,1)
v1.x = 100
v2.y = 100
print(v1, v2)
local v3 = v1 + v2
print(v1.x, v2.x)
print(CS.UnityEngine.Vector3.one)
print(CS.UnityEngine.Vector3.Distance(v1, v2))
```

类型映射

基本数据类型

C#类型	Lua 类型
sbyte, byte, short, ushort, int, uint, double, char, float	number
decimal	userdata
long, ulong	userdata/lua_Integer(lua 53)
bytes[]	string
bool	boolean
string	string

复杂数据类型

C#类型	Lua 类型
LuaTable	table
LuaFunction	function
class 或者 struct 的实例	userdata, table
method, delegate	function

LuaTable:

C#侧指明从 Lua 侧输入（包括 C#方法的输入参数或者 Lua 方法的返回值）LuaTable 类型，则要求 Lua 侧为 table。或者 Lua 侧的 table，在 C#侧未指明类型的情况下转换成 LuaTable。

LuaFunction:

C#侧指明从 Lua 侧输入（包括 C#方法的输入参数或者 Lua 方法的返回值）LuaFunction 类型，则要求 Lua 侧为 function。或者 Lua 侧的 function，在 C#侧未指明类型的情况下转换成 LuaFunction。

LuaUserData:

对应非 C# Managered 对象的 lua userdata。

class 或者 struct 的实例:

从 C#传一个 class 或者 struct 的实例，将映射到 Lua 的 userdata，并通过__index 访问该 userdata 的成员

C#侧指明从 Lua 侧输入指定类型对象，Lua 侧为该类型实例的 userdata 可以直接使用；如果该指明类型有默认构造函数,Lua 侧是 table 则会自动转换,转换规则是:调用构造函数构造实例,并用 table 对应字段转换到 c#对应值后赋值各成员。

method, delegate:

成员方法以及 delegate 都是对应 lua 侧的函数。

C#侧的普通参数以及引用参数，对应 lua 侧函数参数；C#侧的返回值对应于 Lua 的第一个返回值；引用参数和 out 参数则按序对应于 Lua 的第 2 到第 N 个参数。