日志系统

# 简介

在Unity编辑器中点击菜单栏：ShibaInu -> LogWindow 可打开日志窗口。界面如下：



#### 数据文件：

日志文件保存在：[项目路径] /Logs/Running.log，您可以在任何时候打开日志窗口查看它。

真机：[Application.persistentDataPath] /Logs/Running.log

注：每次运行时，将会清空上次运行保存的日志数据。如需备份，请及时拷贝另存该文件。

也可以点击[浏览]按钮选择文件。或在输入框中填入文件地址，点击 [加载/刷新] 加载其他（或刷新当前）日志文件。

列表的排序规则为日志的打印顺序（日志产生的时间越早越靠前）

#### 筛选/查询：（不区分大小写）

在类型下拉框中可筛选出指定类型的日志，也可以输入指定的类型进行筛选。

或者，在后面的输入框中填入指定关键字进行查询筛选。

#### 日志内容：

日志标题前如果有图标，表示这条日志包含堆栈信息。可点击标题 展开/收起 堆栈内容。

或点击左下角 [展开堆栈信息] 选框，展开/收起 所有日志的堆栈内容。

点击日志标题或堆栈内容，可复制日志完整内容到系统剪切板。不信你试试！

警告日志和报错日志默认包含堆栈内容，普通日志可选择是否包含堆栈内容（默认不包含）。

在lua中产生的（未捕获）Error以及C#中产生（未捕获）Exception，将会自动记录到文件中。

# 在Lua 中使用

*log*("这是一条普通的日志")  
*log*("这是一条自定义类型的日志", "MyLogType")  
*log*("这是一条包含堆栈信息的普通日志", nil, true)  
*log*("这是一条包含堆栈信息的自定义类型日志", "MyLogType", true)  
*logWarning*("这是一条警告日志")  
*logError*("这是一条报错日志")

# 在C#中使用

ShibaInu.Logger.Log ("这是一条普通的日志");  
ShibaInu.Logger.Log ("这是一条自定义类型的日志", "MyLogType");  
ShibaInu.Logger.LogException ("这是一条报错日志");  
ShibaInu.Logger.LogException (new Exception ("这是一条报错日志"));  
UnityEngine.Debug.Assert (false, "这是一条断言日志");

# 其他

* 可在lua中使用 trycall(fun, [caller]) 来实现类似其他语言中的 try catch：

**if not** *trycall*(MyFun) **then** -- catch  
**end**

-- 代码可继续执行

注：如果调用的函数报错，会自动打印错误日志，并且不会阻止代码继续运行。

* 请保持Unity控制台和日志系统干净整洁！

调试完毕后，及时删除print() 等输出指令和各类临时调试日志。