aardio 范例: 在 aardio 中显式构建 .NET 泛型 - 异步任务示例

```
//在 aardio 中显式构建 .NET 泛型 - 异步任务示例
import win.ui;
/*DSG{{*/
var winform = win.form(text="泛型 / 异步任务";right=759;bottom=469)
winform.add(
edit={cls="edit";left=9;top=9;right=743;bottom=445;edge=1;multiline=1;z=1}
)
/*}}*/

winform.show();

import dotNet;
var Task = System.Threading.Tasks.Task;

/*
$开头的 .NET 类名或函数名表示创建泛型类或函数（构造具体类型），
参数为一个或多个.NET类或类型名称（字符串）参数。

支持以下写法：
Task.$Run("System.Int32")
Task.$Run(System.Int32)
Task.$Run(0) //传入其他参象，自动获取该对象的 .NET 类型。
Task.Run.$(System.Int32)  //Task.Run 是一个类才能这样写

一定要在界面线程中创建异步任务，.NET 4.5 开始支持 Task.Run。
*/
var taskRun = Task.$Run(System.Int32);//缓存泛型实例，避免重复查询

var task = taskRun(
    function() {
        /*
        模拟一些工作，
        注意，无论 .NET 调用是在哪个线程，
        aardio 回调总是在调用 .NET 的同一线程执行。
        */
        thread.delay(1000);
        return 42;
    }
);

//task 是异步任务，任务完成以后回调以下函数
task.ContinueWith(
    function(t) {
        if(t.IsFaulted){
            //winform.edit.log("出错了",tostring(t.Exception))
            winform.edit.log(t.Exception.Message)
        }
        elseif(!t.IsCanceled){
            var result = t.Result;
            winform.edit.print("task.ContinueWith 完成了，返回值",result)
        }
    }
)

//也可以同步等待任务完成，不会阻塞界面
if( dotNet.wait(task,winform) ){
    winform.edit.print( "dotNet.wait 完成了，返回值",task.Result)
}

//C# 代码可以下面这样异步转同步，取消和失败会抛出异常
//var result = task.GetAwaiter().GetResult();

win.loopMessage();
```

Markdown 格式