aardio 范例: 跨线程操作窗口控件

```
//跨线程操作窗口控件
import win.ui;
/*DSG{{*/
var winform = win.form(text="aardio form";right=560;bottom=413;)
winform.add(
edit={cls="edit";left=25;top=30;right=535;bottom=139;dl=1;dr=1;dt=1;edge=1;multiline=1;z=1};
treeview={cls="treeview";left=25;top=165;right=535;bottom=390;asel=false;bgcolor=16777215;db=1;dl=1;dr=1;dt=1;edge=1;z=2}
)
/*}}*/

var hItem = winform.treeview.insertItem( {
    text = "children数组指定子节点";
    children = {
        { text = "子节点"  };
        { text = "子节点2" };
    }
} )

thread.invoke(
    function(winform,hItem){
        winform.show();
        winform.text = 'aabc测试123'
        winform.edit.text = '-----------------------------------\n'

        /*
        上面这么帅的功能其实是用
        thread.callWnd ,thread.setWnd, thread.getWnd实现的,
        也可以直接调用这几个函数，例如：
        */
        thread.setWnd(winform.hwnd,"text","原来也就是个语法糖呀")

        /*
        切记控件传入多线程时，在线程中的对象只是一个影子代理，
        所有对控件成员函数的调用都会转发到界面线程内执行。

        如果跨线程调用控件的成员函数时，返回值是一个函数对象，
        那么该函数会复制到当前线程，函数必须遵守线程函数的规则，能独立执行不依赖其他线程的局部变量闭包。
        一个典型的例子就是函数名字为each前缀的一系列创建迭代器的函数，
        这些函数违反了aardio跨线程交互的规则（函数依赖的上层局部变量在界面线程，却试图在工作线程中调用他）。

        aardio目前自动提供一个解决方案，
        当跨线程调用控件的成员函数，并且函数名为each前缀时，aardio负责把each函数复制到当前线程。
        这样调用each函数创建的闭包存在于当前线程，则工作正常。在编写控件的each函数时应遵守线程函数的规则（不使用外部的局部变量闭包）

        这个解决方案类似于在线程中执行这样的操作：
        winform.listview[["each"]] = winform.listview.each
        使用直接下标[[]]跳过了元表中的运算符重载（避免函数被再次传回界面线程）.

        这些操作由aardio自动完成，了解以上原理有助于更好的运用该特性
        */

        for hSubItem in winform.treeview.each(hItem) {
            winform.edit.print(winform.treeview.getItemText(hSubItem));
        }
    }, winform,hItem
)

win.loopMessage();
return winform;
```

Markdown 格式