DuiVision

QQ 群：325880743

微信公众号：blueantstudio

蓝蚂蚁工作室

http://www.blueantstudio.net

[DUIVISION TCL 脚本开发手册]

DuiVision 界面库文档[更新日期：2020-12-08]

目录

DuiVision Tcl 脚本开发手册

1. 概述

DuiVision 支持通过 Tcl 脚本编写代码，包括全部代码用 tcl 脚本编写，或者部分用 C++，部分用 tcl 脚本。

DuiVision 的脚本应用程序由下面几部分组成：

1、C++的宿主程序，可以用 DuiVision 的向导生成一个普通的 DuiVision 应用程序，然后去掉生成的代码中 C++创建对话框和定义事件处理类等代码，改用 DuiVision 库提供的加载脚本解释器的代码加载解释器，并执行脚本；

2、tcl 脚本解释器，使用 Script.NET 最新开源版本中的 Tcl 解释器编译之后就可以用，tcl 脚本运行需要依赖 tcl 的一些库，可以根据需要复制相应的库文件到执行目录下；

3、tcl 脚本文件，可以放在任意位置，DuiVision 已经提供了一个基于 iTcl 的封装类，把对话框和事件处理类进行了封装，基于这个封装类开发会比较简单；

4、界面定义 xml 文件，就是 DuiVision 的界面定义文件，可以放在任意位置，一般可以和相应的 tcl 脚本放在相同的目录；

5、图片等资源文件，按照 DuiVision 的定义和存放位置进行存放，在 resource.xml 中进行定义。

2. 编译说明

DuiVision 的 Tcl 解释器使用的 Script.NET 的解释器，需要先编译 Script.NET 的 Tcl 解释器，然后把编译后的目录放在最终脚本工程的某个目录下面，Script.NET Tcl 解释器编译步骤是：

1、把 DuiVision 的 lib 工程目录拷贝到 Script.NET 的 src\dependlib\DuiVision 目录下；

2、把 DuiVision 编译后的 lib 文件拷贝到 Script.Net 的 scr\dependlib\lib 目录下，包括 DuiVision.2008.lib 和 DuiVision.2008d.lib 文件，注意要用多字节方式编译 lib 文件，因为 Script.NET 的解释器工程只支持多字节编译方式；

3、打开 Script.NET 的解释器工程，编译 Tcl 解释器（TclInterp）工程，编译出的 dll 文件在 Script.Net 的 src\interps\tcl\bin 目录下，分别是 TclInterp.dll 和 TclInterp_d.dll，是 release 和 debug 版本的 dll；

4、把 Script.NET 的 src\interps\tcl 目录下的 bin 和 lib 目录拷贝到指定的脚本工程的目录下，这两个目录都是运行过程中不能少的文件，bin 目录下除了编译出的 Tcl 解释器动态库之外，还包括 tcl85.dll 和 tk85.dll，是 tcl 的依赖库。

3. 基于 Tcl 的 DuiVision 代码开发说明

Script.NET 的 tcl 扩展命令中封装了操作 DuiVision 库的相关 tcl 命令，包括资源、对话框、控件、日志、事件处理、定时器等，同时还封装了一个 iTcl 的 DuiVision 事件处理封装基类，基于 Tcl 的 DuiVision 脚本开发大部分情况下可以使用封装类进行开发，样例如下：

```
#####################################################################
# test_dui_gridctrl.tcl
# Author     : blueant
# Version    : 1.0
# Date       : 2017-01-24
# Description: test tclduivision dialog
#####################################################################

package require TclDuiVision

#-------------------------------------------------------------
# TDuiGridCtrlDialogHandler class define
#-------------------------------------------------------------
class  TDuiGridCtrlDialogHandler {
    inherit TDuiDialogHandler

    constructor {{xmlTemplate ""}} {};
    destructor {};

    ### data member ###
    protected variable _progressIndex;  # 进度条的进度

    ### public methods ###
    public method  OnInit {};
    public method  OnTimer {name};      # 定时器处理函数
    public method  OnButtonOK {name msg wParam lParam};
    public method  OnButtonAddLine1 {name msg wParam lParam};
    public method  OnButtonAddLine2 {name msg wParam lParam};
}

#-------------------------------------------------------------
#   单元构造函数
#-------------------------------------------------------------
body TDuiGridCtrlDialogHandler::constructor {{xmlTemplate ""}} {
```

```
    chain $xmlTemplate
    SetLogLevel DEBUG
    # 定义控件DUI消息的iTcl映射方法
    MsgMethod "button.ok.gridctrl" $DUIVISION::MSG_BUTTON_UP OnButtonOK
    MsgMethod "button.addline1" $DUIVISION::MSG_BUTTON_UP
OnButtonAddLine1
    MsgMethod "button.addline2" $DUIVISION::MSG_BUTTON_UP
OnButtonAddLine2

    set _progressIndex 0
}

#-----------------------------------------------------------------
#   单元析构函数
#-----------------------------------------------------------------
body TDuiGridCtrlDialogHandler::destructor {} {
    StopTimer "timer_animate_progress"
    chain;
}

#-----------------------------------------------------------------
#   初始化
#-----------------------------------------------------------------
body TDuiGridCtrlDialogHandler::OnInit {} {
    chain;
    puts "TDuiGridCtrlDialogHandler::OnInit"
    #puts "Start Timer:timer_1 5000"
    StartTimer "timer_animate_progress" 1000

    # 添加gridctrl行
    set gridctrl [GetControl "gridctrl_test"]
    set rowItems [list [list "测试" "在OnInit函数中添加的项"] \
                  [list "当前：1.0.0" "最新：2.0.0"] \
                  [list "10M"] \
                  [list "一键安装"]]
    dui::gridctrl addrow $gridctrl "" -check 0 -image
"skins/icon/softmgr.png" -items $rowItems
}

#-----------------------------------------------------------------
#   Button 添加行 点击消息
#-----------------------------------------------------------------
```

```
body TDuiGridCtrlDialogHandler::OnButtonAddLine1 {name msg wParam lParam}
{
    puts "TDuiGridCtrlDialogHandler::OnButtonAddLine1"

    #set gridctrl [dui::control getcontrol "gridctrl_test" -parentDialog
$_objDuiObject]
    # gridctrl添加行
    set gridctrl [GetControl "gridctrl_test"]
    set rowCount [dui::gridctrl getrowcount $gridctrl]
    set rowItems [list [list "动态添加行" "点击添加行按钮添加的行" 0 -1
"128,128,0"] \
                    [list "当前：2.0.0" "最新：3.0.0" 1 0] \
                    [list "20M"]]
    dui::gridctrl addrow $gridctrl "" -check 0 -image
"skins/icon/softmgr.png" -items $rowItems
    # 最后一列添加子控件
    set control [dui::gridctrl addcontrol $gridctrl $rowCount 3 "button"
"grid_btn_resume$rowCount" "10,20,27,37"]
    dui::control setattribute $control image
"skins/qq/softtastmgr_btn_resume.png"
    dui::control setattribute $control showfocus 0
    #dui::control setposstr $control "10,20,27,37"
    set control [dui::gridctrl addcontrol $gridctrl $rowCount 3 "button"
"grid_btn_cancel$rowCount" "30,20,47,37"]
    dui::control setattribute $control image
"skins/qq/softtastmgr_btn_cancel.png"
    dui::control setattribute $control showfocus 0
}

#-----------------------------------------------------------------
#   Button 添加链接行 点击消息
#-----------------------------------------------------------------
body TDuiGridCtrlDialogHandler::OnButtonAddLine2 {name msg wParam lParam}
{
    puts "TDuiGridCtrlDialogHandler::OnButtonAddLine2"

    set gridctrl [GetControl "gridctrl_test"]
    set rowItems [list [list "动态添加行" "点击添加链接行按钮添加的行" 0 -1
"128,0,0"] \
                    [list "当前：3.0.0" "最新：4.0.0" 1 1] \
                    [list "30M"] \
                    [list "链接" "dlg:dlg_about" "-link"]]
```

```
    dui::gridctrl addrow $gridctrl "" -check 0 \
              -image "skins/icon/scriptnet.png" \
              -backColor "50,128,128,0" \
              -items $rowItems
}

#-------------------------------------------------------------
#   DUI定时器处理函数
#-------------------------------------------------------------
body TDuiGridCtrlDialogHandler::OnTimer {name} {
    chain $name;
    #puts "TDuiGridCtrlDialogHandler::OnTimer $name"
    if {$name == "timer_animate_progress"} {
        incr _progressIndex
        if {$_progressIndex > 10} {
            set _progressIndex 0
        }

        # 刷新进度条
        set progress_test [GetControl "progress_test"]
        dui::control setattribute $progress_test value [expr
$_progressIndex*10] -update
    }
}

#-------------------------------------------------------------
#   Button OK点击消息
#-------------------------------------------------------------
body TDuiGridCtrlDialogHandler::OnButtonOK {name msg wParam lParam} {
    puts "TDuiGridCtrlDialogHandler::OnButtonOK"

    # 获取选择了第几行
    set gridctrl [GetControl "gridctrl_test"]
    set rowCount [dui::gridctrl getrowcount $gridctrl]
    for {set i 0} {$i < $rowCount} {incr i} {
        #
        set rowCheck [dui::gridctrl getrowcheck $gridctrl $i]
        if {$rowCheck == 1} {
            puts "选择了第 [expr $i+1] 行"
        }
    }
```

```
    # 获取输入框的值
    set edit_gridctrl_1 [GetControl "edit_gridctrl_1"]
    set editValue [dui::control getvalue $edit_gridctrl_1]
    puts "输入框的内容：$editValue"

    # 关闭对话框
    DoOK
}

# 显示指定的xml对话框定义文件
set current_path [file dirname [info script]]
set xml_file_gridctrl "$current_path/dlg_gridctrl.xml"
puts $xml_file_gridctrl
TDuiGridCtrlDialogHandler objDuiGridCtrlHandler $xml_file_gridctrl
objDuiGridCtrlHandler ShowDialog
delete object objDuiGridCtrlHandler
```

TclDuiVision 扩展包已经把 DuiVision 的事件处理和对话框创建都进行了封装，只要从 TDuiDialogHandler 派生一个 iTcl 类就可以，派生的类中 OnInit 方法是在对话框创建时候会调用的方法，OnTimer 是在定时器中会调用的方法，除了这两个方法，其他的 DuiVision 事件都可以在派生类中写一个方法，然后再构造函数中通过封装的映射方法进行映射，映射方法是在 TDuiDialogHandler 中已经封装好的，类似于下面这样，表示响应 button.ok.gridctrl 这个按钮控件的 MSG_BUTTON_UP 的方法是 OnButtonOK，所有的消息定义可以参考 Script.NET 的 src\interps\tcl\lib\TclDuivision 目录下的 TclDuiVision.tcl 中的定义：

MsgMethod "button.ok.gridctrl" $DUIVISION::MSG_BUTTON_UP OnButtonOK

以上代码在事件处理封装类定义好之后，通过下面的代码创建了事件处理对象，构造函数的参数传入的是对应的对话框 xml 定义文件：

```
TDuiGridCtrlDialogHandler objDuiGridCtrlHandler $xml_file_gridctrl
```

如果要显示对话框，可以调用下面的代码：

```
objDuiGridCtrlHandler ShowDialog
```

对话框关闭之后，调用下面的代码删除事件处理对象，同时会删除对话框：

```
delete object objDuiGridCtrlHandler
```

以上代码是脚本 Demo 程序中的代码，执行效果如下：

4. 扩展命令说明

4.1. 平台扩展命令

1、plat::log

日志相关操作，包括：

子命令	参数	说明
	-level level ?–module module? info	记录日志，日志级别可以试：DEBUG,INFO,ERROR,CRITICAL，默认是 INFO

4.2. Tcl DuiVision 扩展命令

TclDuiVision 封装如下命令：

1、dui::resource

对 DUI 资源的相关操作，包括：

子命令	参数	说明
load	xmlFile	加载资源文件
getconfig	name	获取配置信息

getxmlfile	name	获取 XML 文件
getskin	name	获取 Skin
getfont	name	获取字体
getstring	name	获取字符串
setstring	name value	设置字符串
parsestring	value	字符串替换

2、 dui::control

对 DUI 控件的相关操作，包括：

子命令	参数	说明
getcontrol	controlName ?-parent parentControl?	获取控件指针，可以指定父控件指针
getcontrol	controlName ?-parentDialog parentDialog?	获取控件指针，可以指定父对话框指针
getcontrol	controlName ?-parentPopup parentPopup?	获取控件指针，可以指定父弹出框指针
getcontrol	controlName -dialog dialogName	获取对话框的控件
getvalue	controlObj	获取控件的值
setvalue	controlObj value ?-update?	设置控件的值
setattribute	controlObj attrName attrValue ?-update?	设置控件的属性值
getposstr	controlObj	获取控件位置字符串
setposstr	controlObj pos	设置控件位置字符串

3、 dui::gridctrl

对 DUI GridCtrl 控件的相关操作，包括：

子命令	参数	说明
addrow	controlObj row ?-id id? ?-image image? ?-rightImage rightImage? ?-check	添加行，可选属性： -image：行左侧图片 -rightImage：行右侧图片 -check ： 是否显示选择框以及初始

	check? ?-textColor textColor? ?-backColor backColor? -items itemList	值 -textColor ： 文字颜色 -backColor ： 背景颜色 itemList 是单元格列表，每个列表项表示一个单元格，也是一个列表，每个单元格的列表内容依次包括：标题文字、内容文字、是否使用标题字体、图片索引、文字颜色、图片文件（如果第三个元素是-link，表示是链接类型的单元格，标题文字和内容文字分表表示链接文字和链接 url）
deleterow	controlObj row	删除行
addcontrol	controlObj row item controlClass controlName ?-dialog dialogObj?	添加单元格子控件，参数为添加的行、列，如果指定对话框指针，则创建时候会带上对话框句柄参数，主要用于输入框等控件
getrowcount	controlObj	获取行数
getrowcheck	controlObj row	获取行的检查框状态

3、 dui::combobox

对 DUI ComboBox 控件的相关操作，包括：

子命令	参数	说明
additem	controlObj name value ?-image image? ?-desc desc? ?-textcolor textcolor? ?-desccolor desccolor?	添加下拉列表的列表项，name 表示列表项的显示信息，value 表示列表项的值。 可选属性： -image：列表项左侧图片 -desc：可选的列表项描述信息 -textColor ： 列表项文字颜色 -descColor ： 列表项描述信息文字颜色 返回值为列表项序号

deleteitem	controlObj item	删除列表项，item 为列表项的序号

3、 dui::handler

对 DUI 的 Handler 相关操作，包括：

子命令	参数	说明
register	duiObj ?-handler handlerObj? ?-control controlName? ?-init?	给指定的 DUI 对象注册 DUI 事件处理对象，如果 handlerObj 不指定，表示要创建一个，此命令返回值是 handlerObj 对象
add	handlerObj	添加 Handler 对象
remove	handlerObj	删除 Handler 对象，在 iTcl 的 Handler 析构函数中可以调用

4、 dui::dialog

对 DUI 对话框的相关操作，包括：

子命令	参数	说明
create	xmlTemplate ?-name dlgName? ?-modal isModule? ?-handler handlerObj? ?-control controlName? ?-init?	创建对话框，如果是模态对话框，在创建对话框时候就需要创建一个事件处理对象的关联，并执行 DoModal
show	xmlTemplate dlgName isModule	创建并显示对话框
messagebox	text caption ?type? ?-width width? ?-height height?	显示消息对话框
notifymsg	msg –caption caption ?-delay delayTime? ?-width width? ?-height height? ?type?	显示提示信息框
showwindow	dlgObj show|hide	显示/隐藏对话框
add	dlgObj	添加对话框
remove	dlgObj	删除对话框
get	dlgName	根据名字获取对话框指针，对于模态对话框，通过 get 命令可能无法

		获取到正确的对话框指针，因为重新设置对话框 name 是在对话框的 OnInitialUpdate 函数中执行的，也就是对话框没有初始化之前是无法获取到正确的值的
close	dlgObj ?-ok|cancel|close|yes|no?	关闭对话框，可以选择点击 OK、Cancel、Close、Yes、No 这几种方式，如果不选择，默认按照 Close，也就是 Cancel 方式

5、 dui::timer

对 DUI 的定时器相关操作，包括：

子命令	参数	说明
add	timerName time	添加定时器
stop	timerName	停止定时器

4.3. TclDuiVision iTcl 封装类

封装类的代码如下，在 Script.Net 的 src\interp\tcl\lib\TclDuiVision 目录下：

```
######################################################################
# File          : TclDuiVision.tcl
# Author        : Blueant
# Version       : 1.0.20170128
# Date          : 2017-01-28
# Description: Tcl DuiVision script library
######################################################################

package provide TclDuiVision 1.0

package require Itcl

catch {namespace import itcl::*}
```

```
# DuiVision 的相关定义
namespace eval DUIVISION {
#  消息类型类型
set MSG_BUTTON_DOWN             1     ;#鼠标或键盘在控件按下
set MSG_BUTTON_UP               2     ;#鼠标或键盘在控件放开
set MSG_BUTTON_DBLCLK           3     ;#鼠标在控件双击
set MSG_BUTTON_CHECK            4     ;#检查框消息
set MSG_SCROLL_CHANGE           5     ;#滚动条位置变更事件
set MSG_CONTROL_BUTTON          6     ;#控件内的按钮点击事件
set MSG_MOUSE_MOVE              7     ;#鼠标移动事件
set MSG_MOUSE_LEAVE             8     ;#鼠标离开事件
set MSG_MOUSE_LDOWN             9     ;#鼠标左键按下事件
set MSG_MOUSE_LUP               10    ;#鼠标左键放开事件
set MSG_MOUSE_RDOWN             11    ;#鼠标右键按下事件
set MSG_MOUSE_RUP               12    ;#鼠标右键放开事件
set MSG_KEY_DOWN                13    ;#键盘按下事件
set MSG_CONTROL_EVENT           14    ;#控件的自定义事件
set MSG_MOUSE_RDBLCLK           15    ;#鼠标右键双击
set MSG_CONTROL_SELECT          16    ;#控件内的选择事件
set MSG_CONTROL_DELETE          17    ;#控件内的删除事件
set MSG_DROP_FILE               18    ;#拖拽文件事件
set MSG_FOCUS_CHANGE            19    ;#控件的焦点状态变更事件
set MSG_KEY_UP                  20    ;#键盘放开事件
}


#-------------------------------------------------------------
# TDuiHandler class define
#-------------------------------------------------------------
if {[itcl::find classes TDuiHandler] != "TDuiHandler"} {
class    TDuiHandler {
        constructor {} {};
        destructor {};
```

```
    ### data member ###
    protected variable _logLevel;        #  日志级别
    protected variable _objDuiObject; # Handler 关联的 DuiVision 对象指针
    protected variable _objHandler;          # C++事件处理对象指针
    protected variable _objTDuiHandler;    # Tcl 事件处理对象名
    protected variable _xmlTemplate; # XML 文件名或内容
    protected variable _lsTimer;         #  创建的 DUI 定时器列表
    protected variable _lsMsgMethod;       # DUI 消息和 Handler 方法的映射表

    ### public methods ###
    public method   GetClassName {};       #  获取类名
    public method   SetDuiObject {objDuiObject};    #  设置 DuiVision 对象指针
    public method   SetLogLevel {logLevel};          #  设置日志级别
    public method   StartTimer {timerName time};   #  启动定时器
    public method   StopTimer {timerName};          #  停止定时器
    public method   MsgMethod {name msg method};    #  添加 DUI 消息的映射方法定义
    public method   OnInit {};                  #  事件处理对象初始化函数,C++调用
    public method   OnMessage {name msg wParam lParam};# DUI 消息处理函数
    public method   OnTimer {name};        #  定时器处理函数
}
}

#-------------------------------------------------------------
#    单元构造函数
#-------------------------------------------------------------
body TDuiHandler::constructor {} {
    chain;

    set _logLevel "INFO"
    set _objDuiObject ""
    set _xmlTemplate ""
    set _lsTimer {}
    set _lsMsgMethod {}
```

```
    # 获取 iTcl 事件处理对象名,根据类名获取第一个对象名
    set lsObjTDuiHandler [itcl::find objects -class [GetClassName]]
    if {[llength $lsObjTDuiHandler] > 0} {
        set _objTDuiHandler [lindex $lsObjTDuiHandler 0]
    }
}

#-------------------------------------------------------------
#    单元析构函数
#-------------------------------------------------------------
body TDuiHandler::destructor {} {
    foreach timer $_lsTimer {
        dui::timer stop $timer
    }
}

#-------------------------------------------------------------
# 获取类名
#-------------------------------------------------------------
body TDuiHandler::GetClassName {} {
    set classname [$this info class];
    if {[string first "::" $classname] == 0} {
        set classname [string range $classname 2 end];
    }
    return $classname;
}

#-------------------------------------------------------------
# 设置 DuiVision 对象指针
#-------------------------------------------------------------
body TDuiHandler::SetDuiObject {objDuiObject} {
    set _objDuiObject $objDuiObject
```

```
}

#-------------------------------------------------------------
#     设置日志级别
#-------------------------------------------------------------
body TDuiHandler::SetLogLevel {logLevel} {
        set _logLevel $logLevel
}

#-------------------------------------------------------------
#     创建定时器
#-------------------------------------------------------------
body TDuiHandler::StartTimer {timerName time} {
        dui::timer add $timerName $time
        lappend _lsTimer $timerName
}

#-------------------------------------------------------------
#     停止定时器
#-------------------------------------------------------------
body TDuiHandler::StopTimer {timerName} {
        dui::timer stop $timerName
}

#-------------------------------------------------------------
#     添加 DUI 消息的映射方法定义
#-------------------------------------------------------------
body TDuiHandler::MsgMethod {name msg method} {
        lappend _lsMsgMethod [list $name $msg $method]
}

#-------------------------------------------------------------
#     初始化
```

```
#-------------------------------------------------------------
body TDuiHandler::OnInit {} {
}

#-------------------------------------------------------------
#     DUI 消息处理
#-------------------------------------------------------------
body TDuiHandler::OnMessage {name msg wParam lParam} {
    if {$_logLevel == "DEBUG"} {
        puts "TDuiHandler::OnMessage $name $msg $wParam $lParam"
    }
    log -level DEBUG "TDuiHandler::OnMessage $name $msg $wParam $lParam"
    foreach msgMethod $_lsMsgMethod {
        set _name [lindex $msgMethod 0]
        set _msg [lindex $msgMethod 1]
        set _method [lindex $msgMethod 2]
        if {($_name == $name) && ($_msg == $msg) && ($_method != "")} {
            $_objTDuiHandler $_method $name $msg $wParam $lParam
        }
    }
}

#-------------------------------------------------------------
#     DUI 定时器处理函数
#-------------------------------------------------------------
body TDuiHandler::OnTimer {name} {
}

#-------------------------------------------------------------
# TDuiDialogHandler class define
#-------------------------------------------------------------
if {[itcl::find classes TDuiDialogHandler] != "TDuiDialogHandler"} {
class    TDuiDialogHandler {
```

```
    inherit TDuiHandler

    constructor {{xmlTemplate ""}} {};
    destructor {};

    ### data member ###

    ### public methods ###
    public method   GetControl {name};   # 获取对话框的指定控件指针
    public method   ShowDialog {};   # 显示对话框
    public method   DoOK {};             # 关闭对话框-OK
    public method   DoCancel {};     # 关闭对话框-Cancel
    public method   DoClose {};      # 关闭对话框-Close
    public method   DoYes {};        # 关闭对话框-Yes
    public method   DoNo {};             # 关闭对话框-No
}
}

#-------------------------------------------------------------
#   单元构造函数
#-------------------------------------------------------------
body TDuiDialogHandler::constructor {{xmlTemplate ""}} {
    chain;
    set _xmlTemplate $xmlTemplate
}

#-------------------------------------------------------------
#   单元析构函数
#-------------------------------------------------------------
body TDuiDialogHandler::destructor {} {
    chain;
    dui::dialog remove $_objDuiObject
}
```

```
#-------------------------------------------------------------
#     获取对话框的指定控件指针
#-------------------------------------------------------------
body TDuiDialogHandler::GetControl {name} {
        return [dui::control getcontrol $name -parentDialog $_objDuiObject]
}

#-------------------------------------------------------------
#     显示对话框
#-------------------------------------------------------------
body TDuiDialogHandler::ShowDialog {} {
        dui::dialog create $_xmlTemplate -handlerName $_objTDuiHandler
}

#-------------------------------------------------------------
#     关闭对话框-OK
#-------------------------------------------------------------
body TDuiDialogHandler::DoOK {} {
        dui::dialog close $_objDuiObject -ok
}

#-------------------------------------------------------------
#     关闭对话框-Cancel
#-------------------------------------------------------------
body TDuiDialogHandler::DoCancel {} {
        dui::dialog close $_objDuiObject -cancel
}

#-------------------------------------------------------------
#     关闭对话框-Close
#-------------------------------------------------------------
body TDuiDialogHandler::DoClose {} {
```

```
	dui::dialog close $_objDuiObject -close
}

#-------------------------------------------------------------
#	关闭对话框-Yes
#-------------------------------------------------------------
body TDuiDialogHandler::DoYes {} {
	dui::dialog close $_objDuiObject -yes
}

#-------------------------------------------------------------
#	关闭对话框-No
#-------------------------------------------------------------
body TDuiDialogHandler::DoNo {} {
	dui::dialog close $_objDuiObject -no
}
```