

## Hypermesh 二次开发培训

2024/3/18

# 目录

## Contents

1. Tcl基础
2. Tk基础
3. 前处理
4. 后处理

# Tcl基础

# Tcl基础

1. 什么是tcl
2. Tcl 编辑器: notepad++, sublime; Tcl解释器: ActiveTcl, Altair Compose
3. 基础语法

# Tcl基础

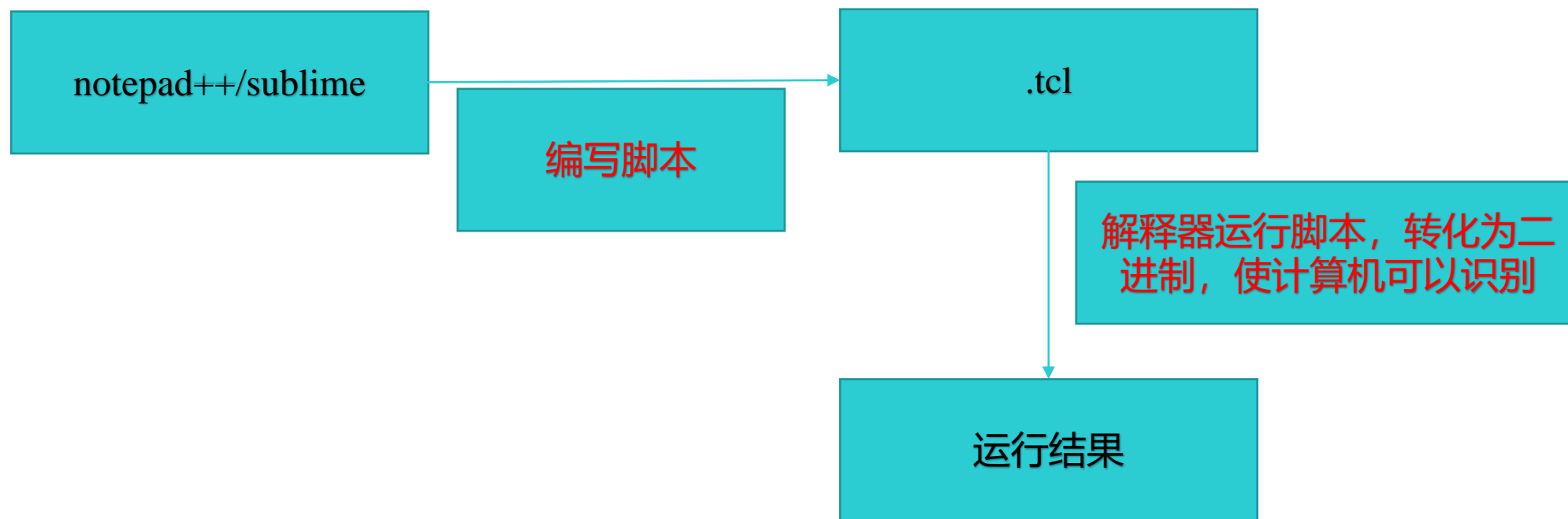
## 1. 什么是tcl

缩写为：Tool Command Language

Tcl是一种解释型的脚本语言。通过规范的API，能够比较方便的进行Tcl的扩展开发。Tk是使用Tcl语言编写的界面控件包。

## Tcl基础

2. Tcl 编辑器: notepad++, sublime; Tcl解释器: ActiveTcl, Altair Compose



## Tcl基础

2. Tcl 编辑器: notepad++, sublime; Tcl解释器: ActiveTcl, Altair Compose

- 学习 Tcl 语言自然需要一个 Tcl 解释器
- HyperMesh 安装后自带Tcl解释器
- 你也可以下载 ActiveTcl 进行练习, ActiveTcl 不仅自带 Tcl 的, 而且不用每次都占用 HyperMesh 的 license, 启动也快很多。

### 下载地址:

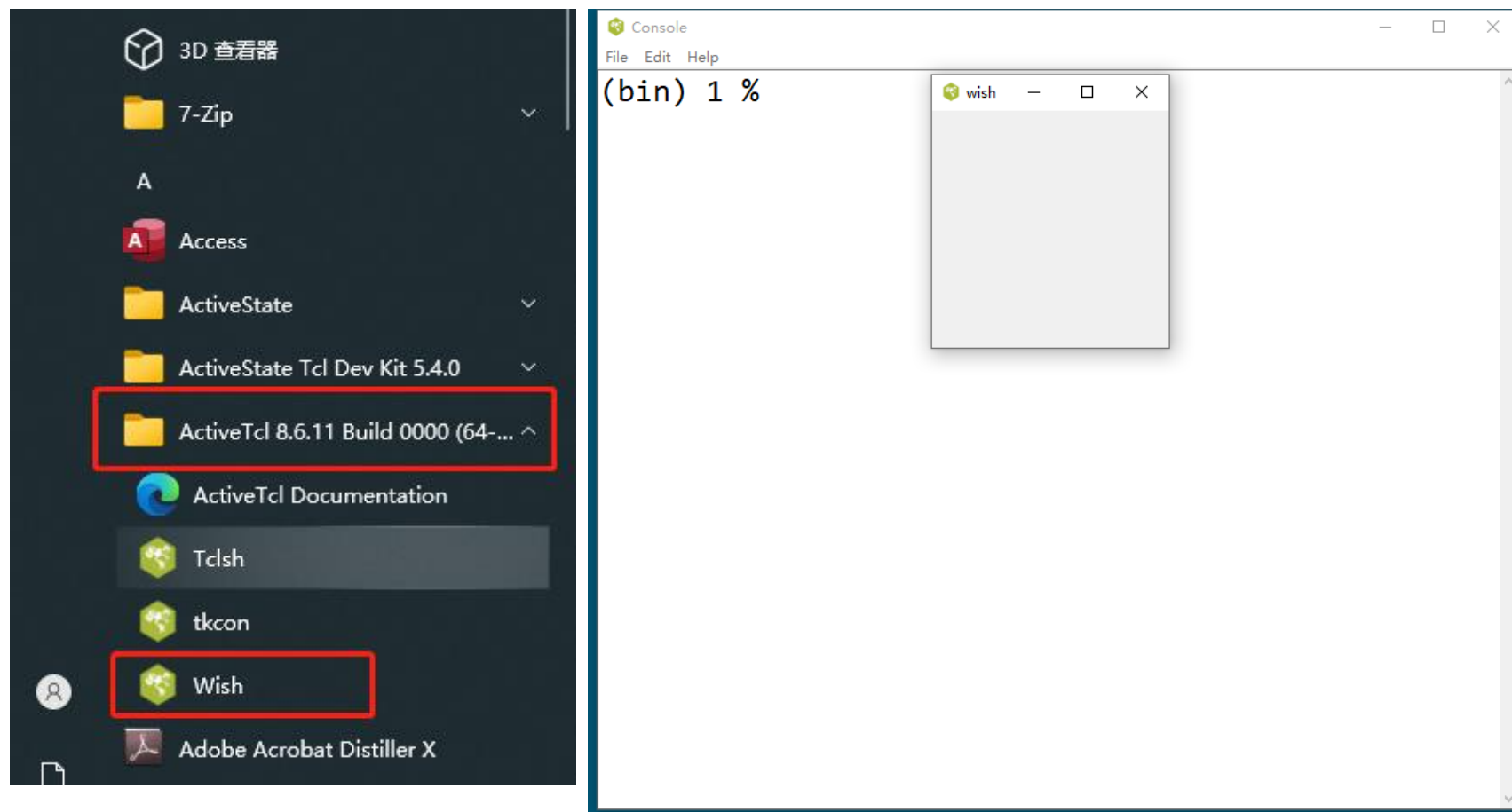
Download Enterprise-grade Tcl from

ActiveState:<https://www.activestate.com/products/tcl/>

Notepad++ - Download (softonic.com):<https://notepad-plus.en.softonic.com/>

Sublime Text - Text Editing, Done Right:<https://www.sublimetext.com/>

# Tcl基础





# Tcl基础语法

1. puts
2. set & unset
3. 表达式
4. 字符串
5. 列表和数组
6. for,foreach循环
7. if
8. proc
9. 命名空间
10. file操作
11. 通配符样式的模式匹配
12. glob
13. 运行脚本

# Tcl基础语法

## 1. puts

**puts** *?-nonewline? ?channelId? string*

语法格式，两个问号之间的表示可省略



### ACTIVE TCL USER GUIDE

[Tcl/Tk Documentation](#) > [TclCmd](#) > [puts](#)

[Tcl/Tk Applications](#) | [Tcl Commands](#) | [Tk Commands](#) | [Tcl Library](#) | [Tk Library](#)

#### NAME

puts - Write to a channel

#### SYNOPSIS

**puts** *?-nonewline? ?channelId? string*

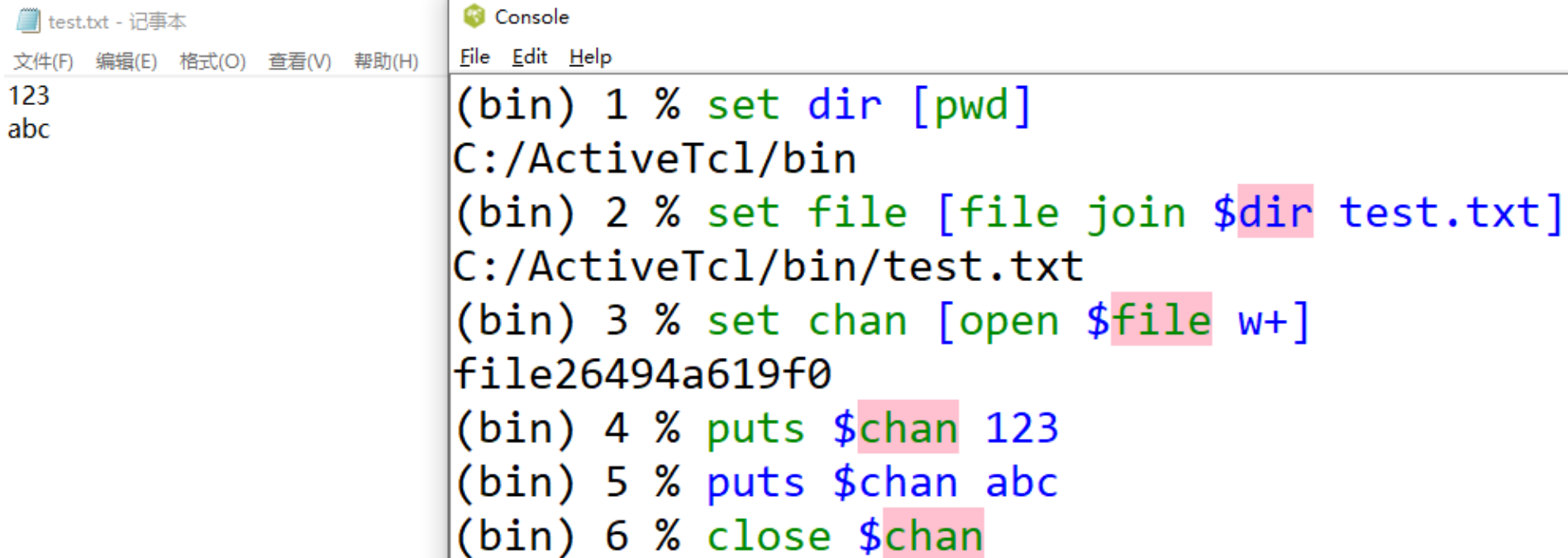
```
(bin) 12 % puts "Hello Tcl"
Hello Tcl
(bin) 13 % puts Hello Tcl
can not find channel named "Hello"
(bin) 14 % puts -nonewline "Hello Tcl"
Hello Tcl(bin) 15 % puts {Hello Tcl}
Hello Tcl
(bin) 16 % puts [Hello Tcl]
invalid command name "Hello"
```

# Tcl基础语法

## 1. puts

**puts** *?-nonewline? ?channelId? string*

输出文本



The screenshot shows a Tcl console window with the following commands and output:

```
(bin) 1 % set dir [pwd]
C:/ActiveTcl/bin
(bin) 2 % set file [file join $dir test.txt]
C:/ActiveTcl/bin/test.txt
(bin) 3 % set chan [open $file w+]
file26494a619f0
(bin) 4 % puts $chan 123
(bin) 5 % puts $chan abc
(bin) 6 % close $chan
```

## Tcl基础语法

- TCL命令的基本结构是: `commandname arg1 arg2 arg3 ...`

这里的`commandname`是tcl所要执行的命令, `args` 是提供给此命令的可变元, 整个行(`commandname` 和 `args` )称为一个命令

- 命令之间通过换行或者由分号(;)来分隔。如果在一行上只有一个命令, 那么分号可以省略。
- 命令和参数之间用空格或者Tab分隔
- 可以把参数用双引号(")和大括号({})分组
- 注释用#开头, 用斜杠\"+回车 表示续行

# Tcl基础语法

Tcl 提供了3种替换形式:

1. 变量替换;
2. 命令替换;
3. 反斜线替换;
4. "", {}, []的区别

```
(bin) 81 % set a 123
123
(bin) 82 % set b $a
123
(bin) 83 % set c [expr {$a+$a}]
246
(bin) 84 % expr {$a+$a}
246
(bin) 85 % set d \$a
$a
(bin) 86 % puts "$a"
123
(bin) 87 % puts {$a}
$a
(bin) 88 % puts [expr {$a+$a}]
246
(bin) 89 % puts "expr {$a+$a}"
expr {123+123}
(bin) 90 % puts "{$a+$a}"
{123+123}
(bin) 91 % puts "{$a}"
{123}
```

# Tcl基础语法

## 2. set & unset

**set** *varname ?value?*

### NAME

set - Read and write variables

### SYNOPSIS

**set** *varName ?value?*

**unset** *?-nocomplain? ?--? ?name name ...?*

### NAME

unset - Delete variables

### SYNOPSIS

**unset** *?-nocomplain? ?--? ?name name name ...?*

```
(bin) 24 % set a "Hello"
Hello
(bin) 25 % set b 1
1
(bin) 26 % set a
Hello
(bin) 27 % set b
1
(bin) 28 % set c
can't read "c": no such variable
(bin) 29 % set a $b
1
(bin) 30 % unset c
can't unset "c": no such variable
(bin) 31 % unset a
(bin) 32 % set a
can't read "a": no such variable
(bin) 33 % unset -nocomplain a
(bin) 34 % unset -nocomplain a b
(bin) 35 % set b
can't read "b": no such variable
```

# Tcl基础语法

## 3. 表达式

**expr** *arg* ?*arg* *arg*...?

### NAME

expr - Evaluate an expression

### SYNOPSIS

**expr** *arg* ?*arg* *arg* ...?

abs(x): 绝对值

ceil(x): 不小于x的最小整数

double(x): 数值等于x的实数

floor(x): 不大于x的最大整数

hypot(x,y): sqrt(x\*x+y\*y)

int(x): x截断到个位数取整

max(x,y):

min(x,y):

pow(x,y): x的y次方

sqrt(x): x的平方根

```
(bin) 10 % expr {0 == 1}
```

```
0
```

```
(bin) 11 % expr {0 == 0}
```

```
1
```

```
(bin) 12 % expr {0+1}
```

```
1
```

```
(bin) 13 % expr {0*1}
```

```
0
```

```
(bin) 14 % expr {0/1}
```

```
0
```

```
(bin) 15 % expr {sin(0)}
```

```
0.0
```

```
(bin) 16 % expr {acos(-1)}
```

```
3.141592653589793
```

```
(bin) 17 % expr {hypot(3,4)}
```

```
invalid command name "tcl::mathfunc::hypot"
```

```
(bin) 18 % expr {hypot(3,4)}
```

```
5.0
```

# Tcl基础语法

## 4. 字符串

string:

**string equal** *?-nocase? ?-length int?* string1 string2

**string index** *string* charIndex

**string is class** *?-strict? ?-failindex varname?* string

alnum:任何字母或数字字符

alpha:任何字母字符

boolean:0, 1, true,false(不区分大小写)

integer: 整数

digit:任何数字字符

double:浮点数

**string length** *string*

**string match** *?-nocase? pattern string*

**string range** *string first last*

**string replace** *string first last ?newstring?*

**string trim** *string ?chars?*

```
(bin) 1 % set str "abcde fghi"
```

```
abcde fghi
```

```
(bin) 2 % string length $str
```

```
10
```

```
(bin) 3 % string length str
```

```
3
```

```
(bin) 4 % string range $str 0 2
```

```
abc
```

```
(bin) 5 % string range $str end-2 end
```

```
ghi
```

```
(bin) 6 % string replace $str 0 2 XX
```

```
XXde fghi
```

```
(bin) 7 % string replace $str 0 2
```

```
de fghi
```

```
(bin) 8 % set s " 123 "
```

```
123
```

```
(bin) 9 % string trim $s " "
```

```
123
```

```
(bin) 10 % string map {a 1} "abc"
```

```
1bc
```

```
(bin) 11 % string map {a 1 1 2} "abc1"
```

```
1bc2
```



# Tcl基础语法

## 5. 列表和数组

**list** *?arg arg?*

**llength** *list*

**lindex** *list index*

**lrange** *list index1 index2*

**lappend** *list arg1 arg2*

**linsert** *list index arg1 arg2*

**lsearch** *?mode? list value*

**lsort** *list*

**concat** *list1 list2*

**join** *list char*

**split** *string char*

```
(bin) 52 % list 1 2 3
1 2 3
(bin) 53 % set l [list 1 2 3]
1 2 3
(bin) 54 % llength $l
3
(bin) 55 % lindex $l 0
1
(bin) 56 % lrange $l 1 2
2 3
(bin) 57 % lappend l 4
1 2 3 4
(bin) 58 % set l
1
1 2 3 4
(bin) 59 % linsert $l 0 100
100 1 2 3 4
(bin) 60 % lsearch $l 2
1
(bin) 61 % lsearch $l 5
-1
(bin) 62 % lsort -decreasing $l
4 3 2 1
```

```
(bin) 74 % set a [list 1 2 3]
1 2 3
(bin) 75 % set b "4 5 6"
4 5 6
(bin) 76 % concat $a $b
1 2 3 4 5 6
(bin) 77 % join $a ","
1,2,3
(bin) 78 % split "1;2;3"
{1;2;3}
(bin) 79 % split "1;2;3" ";"
1 2 3
(bin) 80 % split "123" ""
1 2 3
```

# Tcl基础语法

## 5. 列表和数组

**array** *option arrname ?arg arg?*

**array** *exist arrname*

**array** *get arrname*

**array** *names arrname*

**array** *set arrname list*

**array** *unset arrname*

**array** *size arrname*

**parray** *arrname*

```
(bin) 19 % array set arraytest [list 1 2 3 4]
(bin) 20 % parray arraytest
arraytest(1) = 2
arraytest(3) = 4
(bin) 21 % array exist arraytest
1
(bin) 22 % array exist arraytest1
0
(bin) 23 % array names arraytest
1 3
(bin) 24 % array get arraytest
1 2 3 4
(bin) 25 % array size arraytest
2
(bin) 26 % array unset arraytest
(bin) 27 % array exist arraytest
0
```

# Tcl基础语法

## 6. for,foreach循环

**for** *start test next body*

**foreach** *varname list body*

**foreach** *varlist1 list1  
?varlist2 list2 ...? Body*

break,continue,跳出循环

Console

File Edit Help

```
(bin) 1 % for {set i 1} {$i<9} {incr i} {puts $i}
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
(bin) 2 % set list [list 1 2 3 4]
```

```
1 2 3 4
```

```
(bin) 3 % foreach var $list {puts $var}
```

```
1
```

```
2
```

```
3
```

```
4
```

# Tcl基础语法

## 7. if

**if** *expr1* **?then?** *body1* **elseif** *expr2* **?then?** *body2* **elseif** ... **?else?** *?bodyN?*

```
(bin) 2 % set v 1
1
(bin) 3 % if {$v==1} {
>             puts "v is one"
>         } elseif {$v==2} {
>             puts "v is two"
>         } else {
>             puts "v is not one or two"
>         }
v is one
```

# Tcl基础语法

## 8. proc

**proc** *name args body*

- 一旦程序被创建, 它将被看作是一条命令
- 调用它的程序的名称, 后面的每个参数都要定义一个值
- 默认的, 程序的返回值是程序体中最后一条命令的结果
- 如果想返回其他的值, 可以使用 `return` 命令

```
(bin) 8 % proc Test1 {var} \
> {
>     if {$var == 1} {
>         puts "var is one"
>     } elseif {$var == 2} {
>         puts "var is not one"
>     } else {
>         puts "var is not one or two"
>     }
> }
(bin) 9 % Test1 1
var is one
(bin) 10 % Test1 2
var is not one
(bin) 11 % Test1 3
var is not one or two
```

# Tcl基础语法

## 9. namespace

### 命名空间的基本概念

命名空间是一个命令与变量的集合。命名空间把命令和变量封装起来以确保它们不会干扰别的命名空间中的命令和变量。

TCL已经有一个这样的集合，就是我们指的全局命名空间。全局命名空间将所有的全局变量和命令放在一起。命名空间也叫做名称空间、名称域、命名域等。

# Tcl基础语法

## 9. namespace

***namespace** ?subcommands? ?arg arg...?*

***namespace current** namespacename*

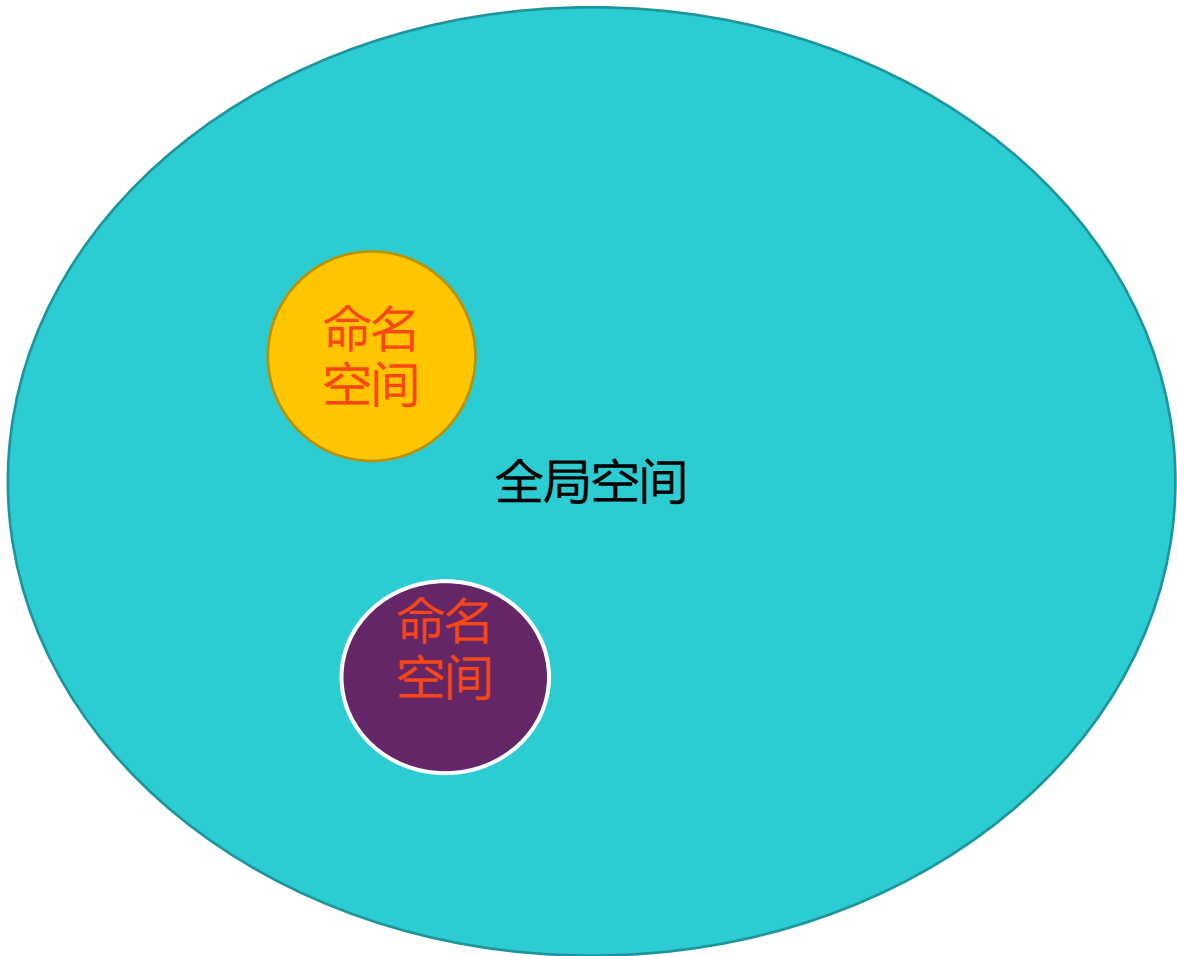
***namespace delete** namespacename*

***namespace exists** namespacename*

***namespace eval** namespacename arg*

命名空间层级使用::分隔

例: ::namespace1::namespace2



# Tcl基础语法

## 9. namespace

*namespace ?subcommands? ?arg arg...?*

*namespace current namespacename*

*namespace delete namespacename*

*namespace exists namespacename*

*namespace eval namespacename arg*

命名空间层级使用::分隔

例: ::namespcae1::namespace2

```
(bin) 1 % namespace exists ::n1
0
(bin) 2 % namespace delete ::n1
unknown namespace "::n1" in namespace delete command
(bin) 3 % namespace eval ::n1 {}
(bin) 4 % namespace exists ::n1
1
(bin) 5 % namespace delete ::n1
(bin) 6 % namespace exists ::n1
0
(bin) 7 % namespace exists ::n1::n2
0
(bin) 8 % namespace eval ::n1::n2::n3::n4 {}
(bin) 9 % namespace exists ::n1::n2
1
(bin) 10 % namespace exists ::n1::n2::n3
1
... ..
```



# Tcl基础语法

## 9. namespace

- 除了全局命名空间外，命名空间的命名都不为空。
- 除了作为命名空间的分隔符外，::也不被简单的命令、变量、命名空间命名接受。
- 在限定命名尾部的::表示该命名空间变量或命令的命名是{}，因此在限定命名空间名称尾部的::会被忽略。另外，在限定命名中单个的:将被看作单个的字符，而2个或更多的单个:将被当作一个命名空间分隔符看待。

例如：

- ✓ namespace eval ::aa::bb::cc {set \_x 100} ;#定义命名空间::aa::bb::cc中的变量\_x
- ✓ set ::aa::bb::cc::\_x ;#查询\_x的值，将返回100
- ✓ set ::aa:::bb:::cc:::\_x ;#也返回100，b与c之间的3个:，c与\_x之间的4个:都被当作一个::看待
- ✓ set ::aa:bb:c::\_x ;#将出错，因为b与c之间的1个:被当作普通的字符看待，但并没有定义::a:bb这样的命名空间,所以要小心！::aa::bb表示的是命名空间aa下的命名空间bb，而::aa:bb表示的是命名空间aa:bb。

# Tcl基础语法

## 10. file

*file option name ?arg arg ...?*

*file join name ?name ...?*

*file dir name*

*file tail name*

*file norm name*

*file native name*

pwd: 查询当前的工作目录

Console

File Edit Help

```
(bin) 1 % set dir [pwd]
C:/ActiveTcl/bin
(bin) 2 % file join $dir a b
C:/ActiveTcl/bin/a/b
(bin) 3 % file join $dir a c.tcl
C:/ActiveTcl/bin/a/c.tcl
(bin) 4 % file dir {C:/ActiveTcl/bin/a/c.tcl}
C:/ActiveTcl/bin/a
(bin) 5 % file dir {C:/ActiveTcl/bin/a}
C:/ActiveTcl/bin
(bin) 6 % file tail {C:/ActiveTcl/bin/a/c.tcl}
c.tcl
(bin) 7 % file tail {C:/ActiveTcl/bin/a}
a
(bin) 8 % file norm {C:/ActiveTcl/bin/a}
C:/ActiveTcl/bin/a
(bin) 9 % file native {C:/ActiveTcl/bin/a}
C:\ActiveTcl\bin\a
(bin) 10 % file norm {C:\ActiveTcl\bin\a}
C:/ActiveTcl/bin/a
```

# Tcl基础语法

## 11. 通配符样式的模式匹配

*string match* *?-nocase?* *pattern*  
*string*

3种常用的匹配模式

**\***:通配符, 匹配任意数量和值的任意字符

**?**: 匹配一个字符

**[chars]**: 匹配chars中的任一字符

```
(bin) 1 % string match a* abc
1
(bin) 2 % string match a* a
1
(bin) 3 % string match a* cba
0
(bin) 4 % string match a? abc
0
(bin) 5 % string match a? ab
1
(bin) 6 % string match a? ac
1
(bin) 7 % string match a? a
0
```

```
(bin) 8 % string match [ab]* abc
invalid command name "ab"
(bin) 9 % string match {[ab]*} abc
1
(bin) 10 % string match {[ab]*} bc
1
(bin) 11 % string match {[ab]*} cb
0
(bin) 12 % string match {[ab]*} a
1
(bin) 13 % string match {[ab]*} b
1
(bin) 14 % set str [ab]*
invalid command name "ab"
(bin) 15 % set str {[ab]*}
[ab]*
(bin) 16 % string match $str b
1
(bin) 17 % string match $str c
0
```

# Tcl基础语法

## 12. glob

**glob** *?switches? pattern ?pattern ...?*

搜索指定文件夹下的文件



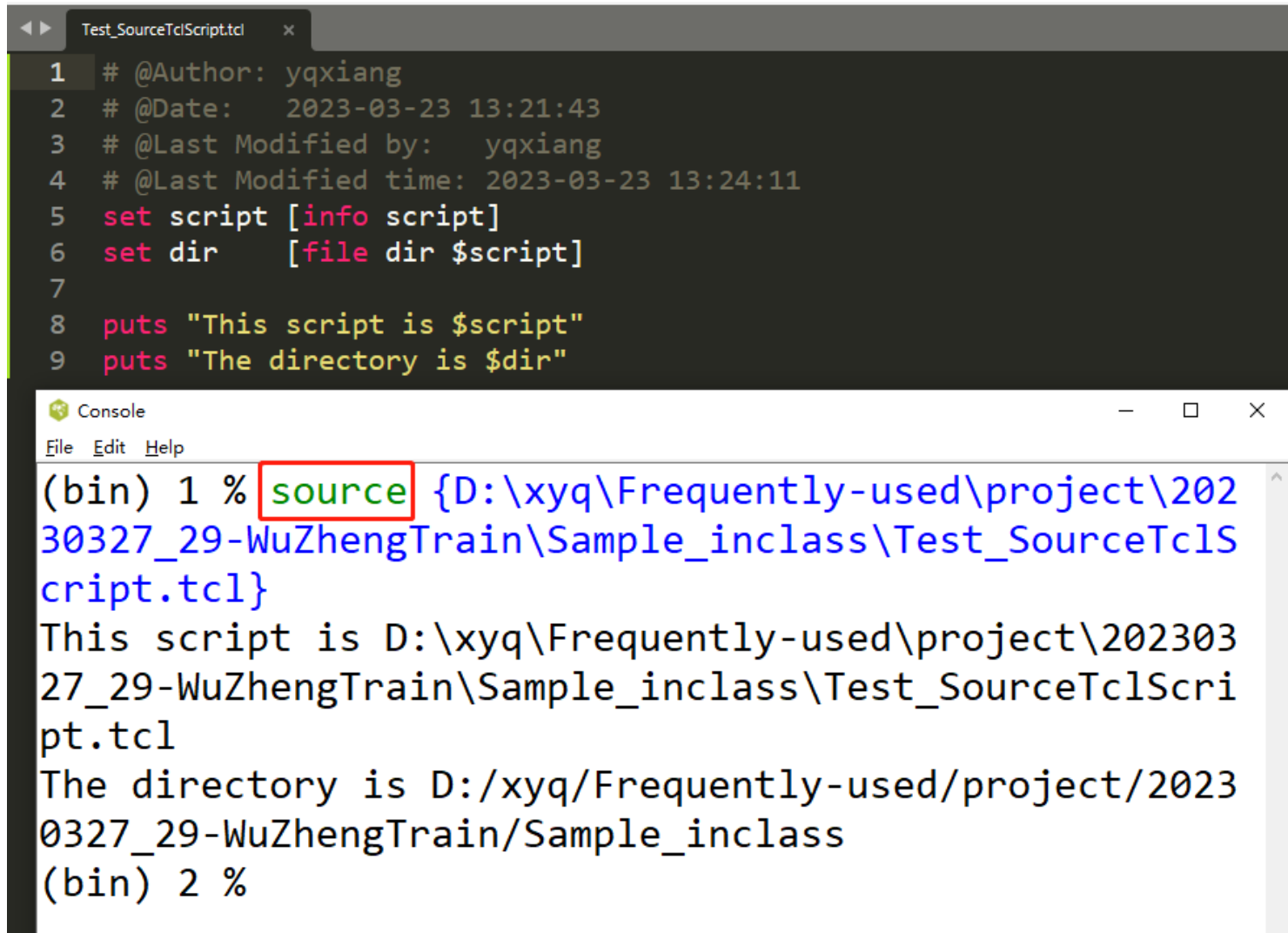
```
(bin) 26 % set dir [pwd]
C:/ActiveTcl/bin
(bin) 27 % glob -dir $dir *.a
no files matched glob pattern "*.a"
(bin) 28 % glob -nocomplain -dir $dir *.a
(bin) 29 % glob -nocomplain -dir $dir *.csv
C:/ActiveTcl/bin/0327.csv C:/ActiveTcl/bin/2011111123.csv C:/ActiveTcl/bin/20221
117-04.csv C:/ActiveTcl/bin/t.csv
(bin) 30 % glob -nocomplain -dir $dir *.tcl
C:/ActiveTcl/bin/bitmap-editor.tcl C:/ActiveTcl/bin/critcl.tcl C:/ActiveTcl/bin/
diagram-viewer.tcl C:/ActiveTcl/bin/dtpltite.tcl C:/ActiveTcl/bin/nns.tcl C:/Acti
veTcl/bin/nnsd.tcl C:/ActiveTcl/bin/nnslog.tcl C:/ActiveTcl/bin/pt.tcl C:/Active
Tcl/bin/tcldocstrip.tcl C:/ActiveTcl/bin/tkcon.tcl
(bin) 31 % glob -nocomplain -dir $dir 0*
C:/ActiveTcl/bin/0327.csv C:/ActiveTcl/bin/0327.txt
```

# Tcl基础语法

## 13. 运行脚本

*source filename*

info script :查询当前脚本的绝对路径 (只在脚本中生效)



```
Test_SourceTclScript.tcl x
1 # @Author: yqxiang
2 # @Date: 2023-03-23 13:21:43
3 # @Last Modified by: yqxiang
4 # @Last Modified time: 2023-03-23 13:24:11
5 set script [info script]
6 set dir [file dir $script]
7
8 puts "This script is $script"
9 puts "The directory is $dir"

Console
File Edit Help
(bin) 1 % source {D:\xyq\Frequently-used\project\20230327_29-WuZhengTrain\Sample_inclass\Test_SourceTclScript.tcl}
This script is D:\xyq\Frequently-used\project\20230327_29-WuZhengTrain\Sample_inclass\Test_SourceTclScript.tcl
The directory is D:/xyq/Frequently-used/project/20230327_29-WuZhengTrain/Sample_inclass
(bin) 2 %
```

# 代码规范

1. 变量名称小写，例如：name, compid, compids
2. 过程(proc)名称首字母大写，例如：ReadCsv, OnCreate, MainUi
3. 命名空间变量以 \_ 开始：例如：namespace::\_name, namespace1::\_compid
4. 代码缩进，“Tab”键，一般4个空格。

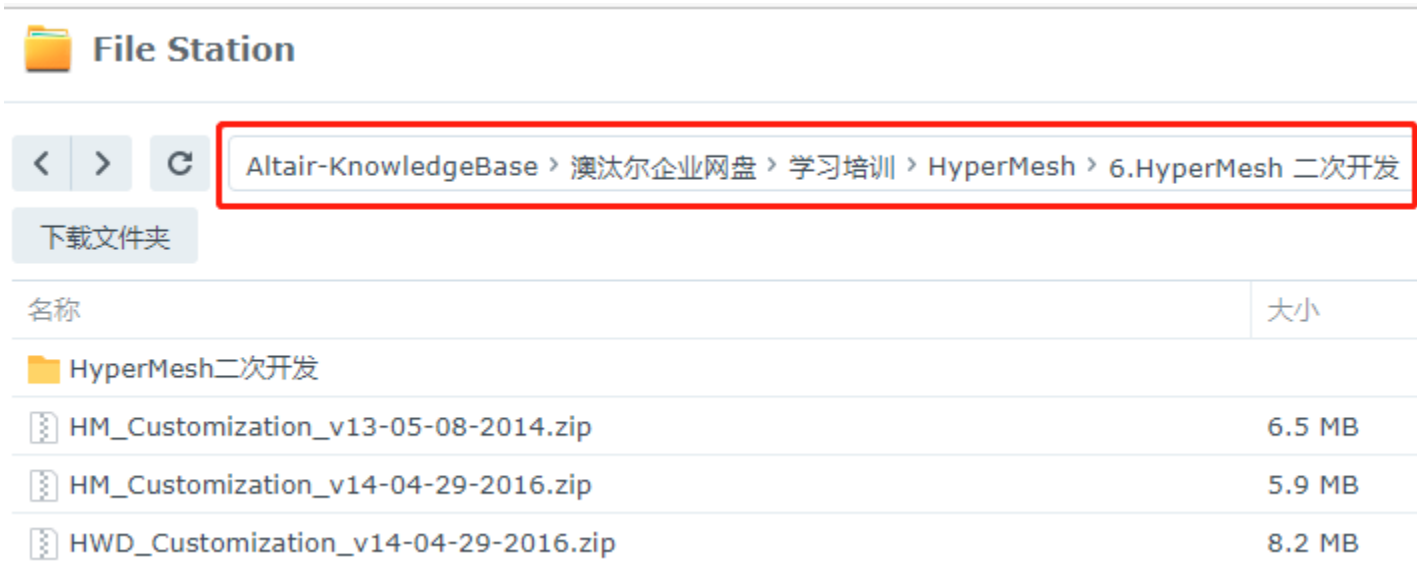
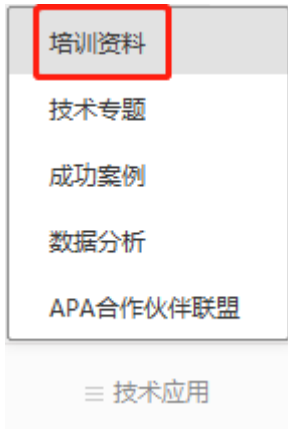
# 学习方法

第一步：关注微信公众号AltairChina并注册

第二步：进入：技术应用 > 培训资料

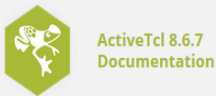
第三步：依次进入： 澳汰尔企业网盘 > 学习培训 > HyperMesh > 6.HyperMesh二次开发  
问题咨询：

[support@altair.com.cn](mailto:support@altair.com.cn)





docs.activestate.com



- Get ActiveTcl
- Release notes
- Windows installation
- Linux/Unix Installaton
- macOS Installation
- License
- Get Started
- Package Reference
- Tcl 8.6.7 docs
- Contact Us

Tcl8.6.7/Tk8.6.7 Documentation > **Tcl Commands, version 8.6.7**

[Tcl/Tk Applications](#) | [Tcl Commands](#) | [Tk Commands](#) | [\[incr Tcl\] Package Commands](#) | [SQLite3 Package Commands](#) | [TDBC Package Commands](#) | [tdbc::mysql Package Commands](#) | [tdbc::odbc Package Commands](#) | [tdbc::postgres Package Commands](#) | [tdbc::sqlite3 Package Commands](#) | [Thread Package Commands](#) | [Tcl C API](#) | [Tk C API](#) | [\[incr Tcl\] Package C API](#) | [TDBC Package C API](#)

<a href="#">after</a>	<a href="#">errorInfo</a>	<a href="#">load</a>	<a href="#">re_syntax</a>	<a href="#">tcl_startOfNextWord</a>
<a href="#">append</a>	<a href="#">eval</a>	<a href="#">lrange</a>	<a href="#">read</a>	<a href="#">tcl_startOfPreviousWord</a>
<a href="#">apply</a>	<a href="#">exec</a>	<a href="#">lrepeat</a>	<a href="#">refchan</a>	<a href="#">tcl_traceCompile</a>
<a href="#">argc</a>	<a href="#">exit</a>	<a href="#">lreplace</a>	<a href="#">regexp</a>	<a href="#">tcl_traceExec</a>
<a href="#">argv</a>	<a href="#">expr</a>	<a href="#">lreverse</a>	<a href="#">registry</a>	<a href="#">tcl_version</a>
<a href="#">argv0</a>	<a href="#">fblocked</a>	<a href="#">lsearch</a>	<a href="#">regsub</a>	<a href="#">tcl_wordBreakAfter</a>
<a href="#">array</a>	<a href="#">fconfigure</a>	<a href="#">lset</a>	<a href="#">rename</a>	<a href="#">tcl_wordBreakBefore</a>

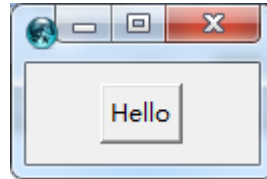


# Tk基础

# Tk

- TK入门小例

```
button .hello -text Hello -command { puts "Hello, World!"}  
pack .hello -padx 20 -pady 10
```



```
.hello config -background red
```

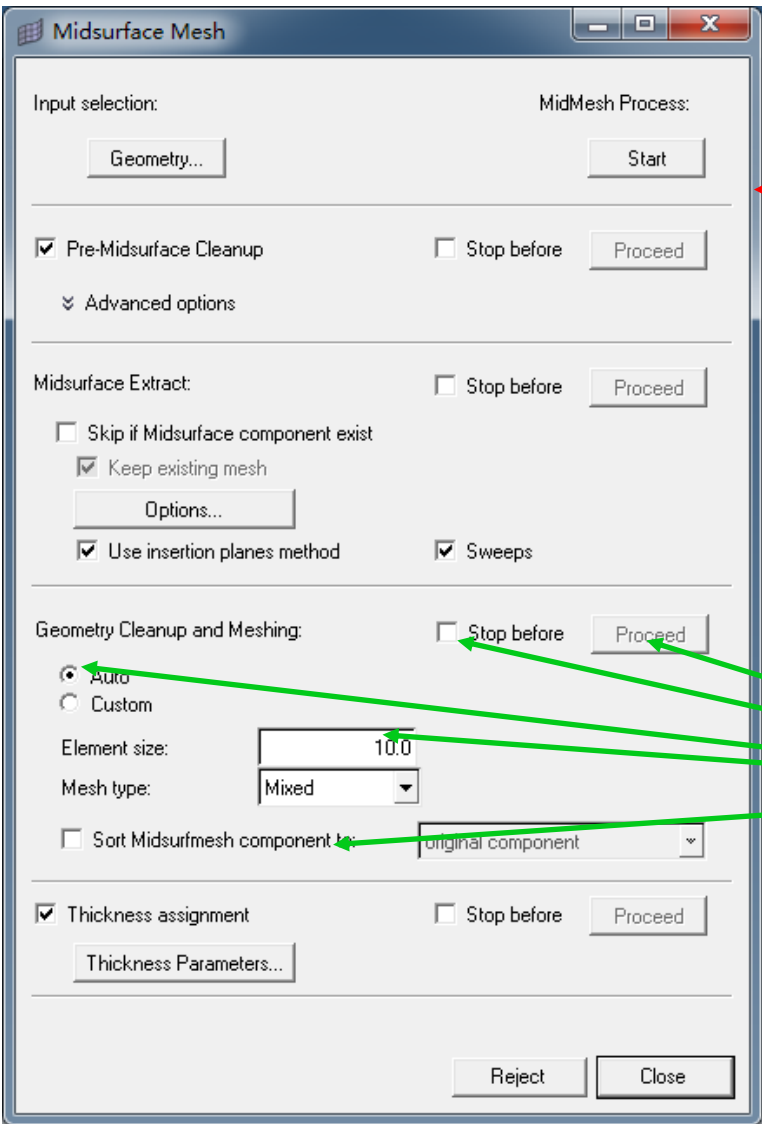
- 什么是TK

TK\_人机交互界面实现的开发包,它可以帮助人们更好的实现其所需要的各种功能,使人们从视觉上得到满足,使程序变的更加直观、更加人性化。

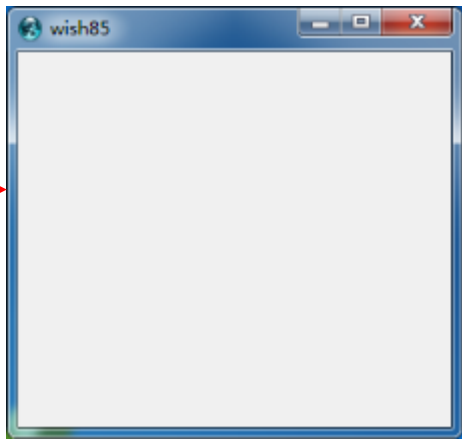
# 怎么样开发一个界面

- 要知道界面的构成
- 要知道界面的布局
- 要知道界面的消息传递

# Tk实例



窗体 (容器)



控件\_可以实现不同的功能

例子中的控件有：按钮、单选按钮、复选按钮、

静态文本框、文本编辑框

# 界面消息传递

- 事件
  - 鼠标点击事件
  - 鼠标移动事件
  - 键盘按键点击事件
- 消息
  - 每个事件会对应激发和接收一个消息
  - 系统消息
  - 自定义消息

# 窗体及控件

- Label控件
- Button控件
- Entry控件
- Radiobutton控件
- Checkbutton控件
- Listbox控件
- Combobox控件
- 新建窗体

# 如何创建Lable组件(静态文本框)

Lable

**lable** *pathName ?Options1? ?Options2?*

Options:

-font fontDescriptor # 设置字体

-textvariable VarName # 设置显示内容  
对应的变量，因此改变变量，静态文本框显示的文本也会变化

-text displayText # 设置显示内容

## CREATION

**Label** *pathName ?option value...?*

## STANDARD OPTIONS

### Not themed

- anchor
- background or -bg
- bitmap
- borderwidth or -bd
- cursor
- disabledforeground
- font
- foreground or -fg
- highlightbackground
- highlightcolor
- highlightthickness
- image
- justify
- padx
- pady
- relief
- takefocus
- text
- textvariable
- wraplength

### Themed

- anchor
- background or -bg
- cursor
- font
- foreground or -fg
- image
- justify
- relief
- takefocus
- text
- textvariable
- wraplength

## WIDGET-SPECIFIC OPTIONS

- dragenabled
- dragendcmd
- dragevent
- draginitcmd
- dragtype
- dropcmd
- dropenabled
- dropovercmd
- droptypes
- focus
- height
- helptext
- helptype
- helpvar
- name
- state
- underline
- width

## WIDGET COMMAND

*pathName* **cget** *option*

*pathName* **configure** *?option? ?value option value ...?*

*pathName* **setfocus**



## 如何创建Label组件(静态文本框)

Example:

```
#创建label组件(文本编辑框)
```

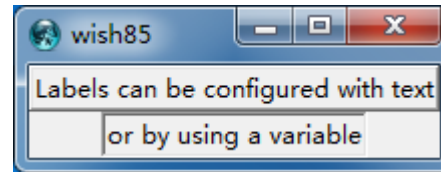
```
set txt [label .la -relief raised -text "Labels can be configured with text"]
```

```
grid $txt
```

```
set var [label .lb -textvariable label2Var -relief sunken]
```

```
grid $var
```

```
set label2Var "or by using a variable"
```



# 如何创建button控件

## Button

**button** *pathName ?Options1? ?Options2?*

Options:

-font fontDescriptor

-textvariable VarName

-text displayText

-command script # 绑定命令，点击button，触发该命令

**NAME**  
 button - Create and manipulate button widgets

**SYNOPSIS**  
**button** *pathName* *?options?*

**STANDARD OPTIONS**  
 -activebackground, activeBackground, Foreground  
 -activeforeground, activeForeground, Background  
 -anchor, anchor, Anchor  
 -background or -bg, background, Background  
 -bitmap, bitmap, Bitmap  
 -borderwidth or -bd, borderWidth, BorderWidth  
 -compound, compound, Compound  
 -cursor, cursor, Cursor  
 -disabledforeground, disabledForeground, DisabledForeground  
 -font, font, Font  
 -foreground or -fg, foreground, Foreground  
 -highlightbackground, highlightBackground, HighlightBackground  
 -highlightcolor, highlightColor, HighlightColor  
 -highlightthickness, highlightThickness, HighlightThickness  
 -image, image, Image  
 -justify, justify, Justify  
 -padx, padx, Pad  
 -pady, pady, Pad  
 -relief, relief, Relief  
 -repeatdelay, repeatDelay, RepeatDelay  
 -repeatinterval, repeatInterval, RepeatInterval  
 -takefocus, takeFocus, TakeFocus  
 -text, text, Text  
 -textvariable, textVariable, Variable  
 -underline, underline, Underline  
 -wraplength, wrapLength, WrapLength

**WIDGET-SPECIFIC OPTIONS**  
 -command, command, Command  
 -default, default, Default  
 -height, height, Height  
 -overrelief, overRelief, OverRelief  
 -state, state, State  
 -width, width, Width

**DESCRIPTION**  
**WIDGET COMMAND**  
*pathName* cget *option*  
*pathName* configure *?option?* *?value option value ...?*  
*pathName* flash  
*pathName* invoke

**DEFAULT BINDINGS**  
**EXAMPLES**  
**SEE ALSO**  
**KEYWORDS**

# 如何创建button组件

Example:

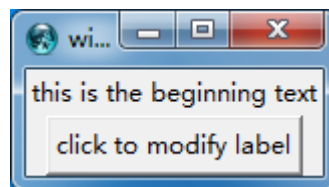
#创建button组件(文本编辑框) #全局作用域下的调用

```
set myLabel [label .l1 -text "this is the beginning text"]
```

```
set myButton [button .b1 -text "click to modify label" -command "$myLabel configure -text {the button was clicked}"]
```

```
grid $myLabel
```

```
grid $myButton
```



# 如何创建新的窗体

Toplevel # 用来创建窗体

**toplevel** *pathName ?Options?*

Options:

-relief value(raised/sunken/ridge/flat) # 显示效果, 凸起, 凹线等

-background color # 背景色

-width # 宽度

WM # 调整窗口的名称

**WM title** *windowName ?String?*

# 如何创建新的窗体

Example:

#放置于原始窗口上的标签组件

```
label .l -text "I am in the original window"
```

#修改主窗口标题

```
wm title . "Main Toplevel window"
```

#创建新的顶级窗口， 以及一个放置于其上的一个标签

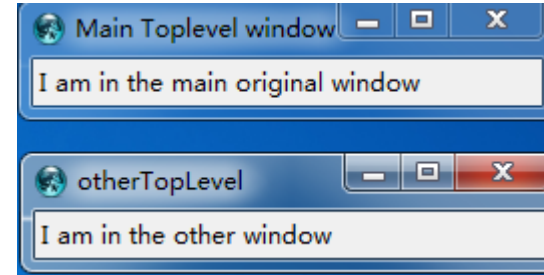
```
toplevel .otherTopLevel
```

```
label .otherTopLevel.l -text "I am in the other window"
```

#显示标签

```
grid .l
```

```
grid .otherTopLevel.l
```



# 容器与布局

## ➤ 容器

- Frame容器

- Labelframe容器

## ➤ 布局

- Pack布局

- Grid布局

# 如何创建frame组件(容器组建)

## 1.frame

**frame** *pathName ?Options?*

Options:

-height numPixels

-width numPixels

-background color

-relief value(sunken raised outlined flat)

-borderwidth



## NAME

frame - Create and manipulate frame widgets

## SYNOPSIS

**frame** *pathName ?options?*

## STANDARD OPTIONS

- borderwidth or -bd, borderWidth, BorderWidth
- cursor, cursor, Cursor
- highlightbackground, highlightBackground, HighlightBackground
- highlightcolor, highlightColor, HighlightColor
- highlightthickness, highlightThickness, HighlightThickness
- padx, padX, Pad
- pady, padY, Pad
- relief, relief, Relief
- takefocus, takeFocus, TakeFocus

## WIDGET-SPECIFIC OPTIONS

- background, background, Background
- class, class, Class
- colormap, colormap, Colormap
- container, container, Container
- height, height, Height
- visual, visual, Visual
- width, width, Width

## DESCRIPTION

## WIDGET COMMAND

*pathName cget option*

*pathName configure ?option? ?value option value ...?*

## BINDINGS

## SEE ALSO

## KEYWORDS

pack

**pack** *widgeName ?Options?*

Options:

- side (top/bottom/left/right) (申明停靠父窗体的位置)
- anchor edge (n/s/e/w,也可以组合)
- expand boolean (1/0, 是否随着父窗体延伸)
- fill derrection (none/x/y/both, 设置延伸方向)
- padx number (像素)
- pady number
- after widgeName

# 布局管理器

Grid (单元格式布局设置)

**grid** *widgetName ?Options?*

Options:

-column number

-row number

-columnspan number

-rowspan number

-sticky side (n/s/w/e ,可组合)(决定组件在单元格中的位置)

grid configure widgetName –option optionValue (修改已经设置好的布局格式)

# HWTk

Altair 2021.2

AbaqusODB UpGrade 2021.2

Altair License Utility 2021.2

BasicFEA 2021.2

BatchMesher 2021.2

Compose 2021.2

Compose 2021.2 Console

Compose 2021.2 Jupyter Notebo...

Compose 2021.2 Jupyter Notebo...

Compute Console 2021.2

Desktop Help 2021.2

HgTrans 2021.2

HvTrans 2021.2

**HWTk GUI Toolkit 2021.2**

HyperCrash 2021.2

HWTk Studio

File Help

Demo Pallet X

Base Widgets 20

button

checkboxbutton

colorbutton

combobox

multiselectcombobox

entry

fileentry

hyperlink

label

labelframe

menu

menubutton

splitframe

progressbar

radiobutton

scale

scalerrange

searchentry

statebutton

toolbutton

Compound Widgets 17

buttonbox

buttongroup

listbox

listview

propertyarea

Preview

Code

切换页面查看源码

控件效果示例

各类控件

Click on the following buttons or press Tab to move among the buttons, then press Space to invoke the current button.

Click me!

Click me! H

H Click me!

H Click me!

Click me!

Click me!

I can be made default

I am currently default

I can't be made default

H

Don't click me!

I am underlined

# HWTk

## Altair HyperWorks™

2020.1

### HyperWorks GUI Toolkit

#### Base Widgets

- hwtk::button
- hwtk::checkboxbutton
- hwtk::choosedirentry
- hwtk::colorbutton
- hwtk::combobox
- hwtk::entry
- hwtk::frame
- hwtk::hyperlink
- hwtk::label
- hwtk::labelframe
- hwtk::menu
- hwtk::menubutton
- hwtk::notebook
- hwtk::openfileentry
- hwtk::progressbar
- hwtk::radiobutton
- hwtk::savefileentry
- hwtk::scale
- hwtk::scalerrange
- hwtk::scrollbar
- hwtk::separator
- hwtk::sizegrip
- hwtk::spinbox
- hwtk::splitframe
- hwtk::statebutton
- hwtk::toolbutton

#### > Compound Widgets

#### > Dialogs

#### > Miscellaneous

#### > Utils

Validation

Specifying File Patterns

Write a New Custom Widget

#### > Model Identification Tool

Register HVPcontrol

[HyperWorks Tools](#) > [HyperWorks GUI Toolkit](#) > Base Widgets

## Base Widgets

### [hwtk::button](#)

Widget that issues a command when pressed.

### [hwtk::checkboxbutton](#)

On/off widget

### [hwtk::choosedirentry](#)

Entry with browse button that pops up a dialog for the user to select a dialog.

### [hwtk::colorbutton](#)

Button with popdown color pallet.

### [hwtk::combobox](#)

Text field with popdown selection list.

### [hwtk::entry](#)

Editable text field widget.

### [hwtk::frame](#)

Simple container widget.

### [hwtk::hyperlink](#)

Displays a hypertext string and/or image. Widget that issues a command when pressed.

### [hwtk::label](#)

Displays a text string and/or image.

### [hwtk::labelframe](#)

Container widget with optional label.

### [hwtk::menu](#)

Create and manipulate menu widgets.

### [hwtk::menubutton](#)

Create and manipulate menu widgets.

### [hwtk::notebook](#)

Multi-pane container widget.

### [hwtk::openfileentry](#)

Pop up a dialog box for the user to select a file to open or save.

[HyperWorks Tools](#) > [HyperWorks GUI Toolkit](#) > [Base Widgets](#) > [hwtk::button](#)
[← Previous](#)

### [pathName configure ?option? ?value option value ...?](#)

Query or modify the configuration options of the widget. If one or more option-value pairs are specified, then the command modifies the given widget option(s) to have the value specified. If no option is specified, then the command returns a list describing the named option: the elements of the list are the option name, database name, database class, default value, and current value. If no option is specified, returns a list describing all of the available options for pathName.

### [pathName cget option](#)

Returns the current value of the configuration option given by option.

### [pathName identify element x y](#)

Returns the name of the element under the point given by x and y, or an empty string if the point does not lie within any element. x and y are pixel coordinates relative to the widget's root window.

### [pathName instate statespec ?script?](#)

Test the widget's state. If script is not specified, returns 1 if the widget state matches statespec and 0 otherwise. If script is specified, equivalent to

```
if {[pathName instate statespec]} script
```

### [pathName state ?stateSpec?](#)

Modify or inquire widget state. If stateSpec is present, sets the widget state: for each flag in stateSpec, sets the corresponding flag or clears it if prefixed by an exclamation point. If no stateSpec is present, returns a list of the currently-enabled state flags.

```
setchanges [pathName stateSpec]
pathName state $changes
```

will restore pathName to the original state. If stateSpec is not specified, returns a list of the currently-enabled state flags.

### [pathName invoke](#)

Invokes the command associated with the button.

## Example

```
::hwtk::dialog .d -title "hwtk::button"
set f [.d recess]

::hwtk::button $f.b1 -text "Text Button" -help "Text only"
::hwtk::button $f.b2 -image productHyperWorks-24.png -help "Image only"
::hwtk::button $f.b3 -text "HyperWorks" -image productHyperWorks-24.png -compound left -help "Image and text compound left"
::hwtk::button $f.b4 -text "HyperWorks" -image productHyperWorks-24.png -compound top -help "Image and text compound top"
pack $f.b1 $f.b2 $f.b3 $f.b4

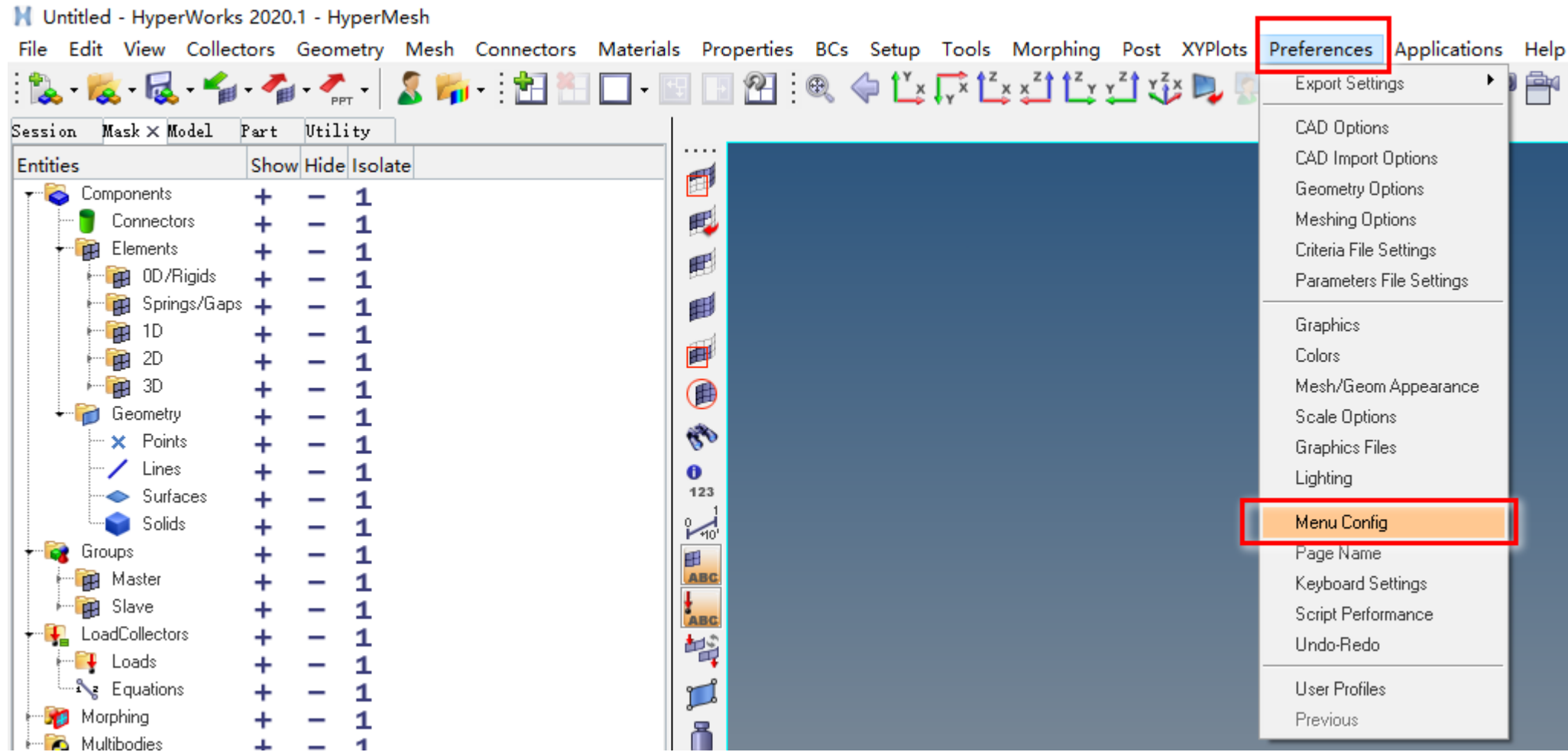
.d post
```

# 前处理

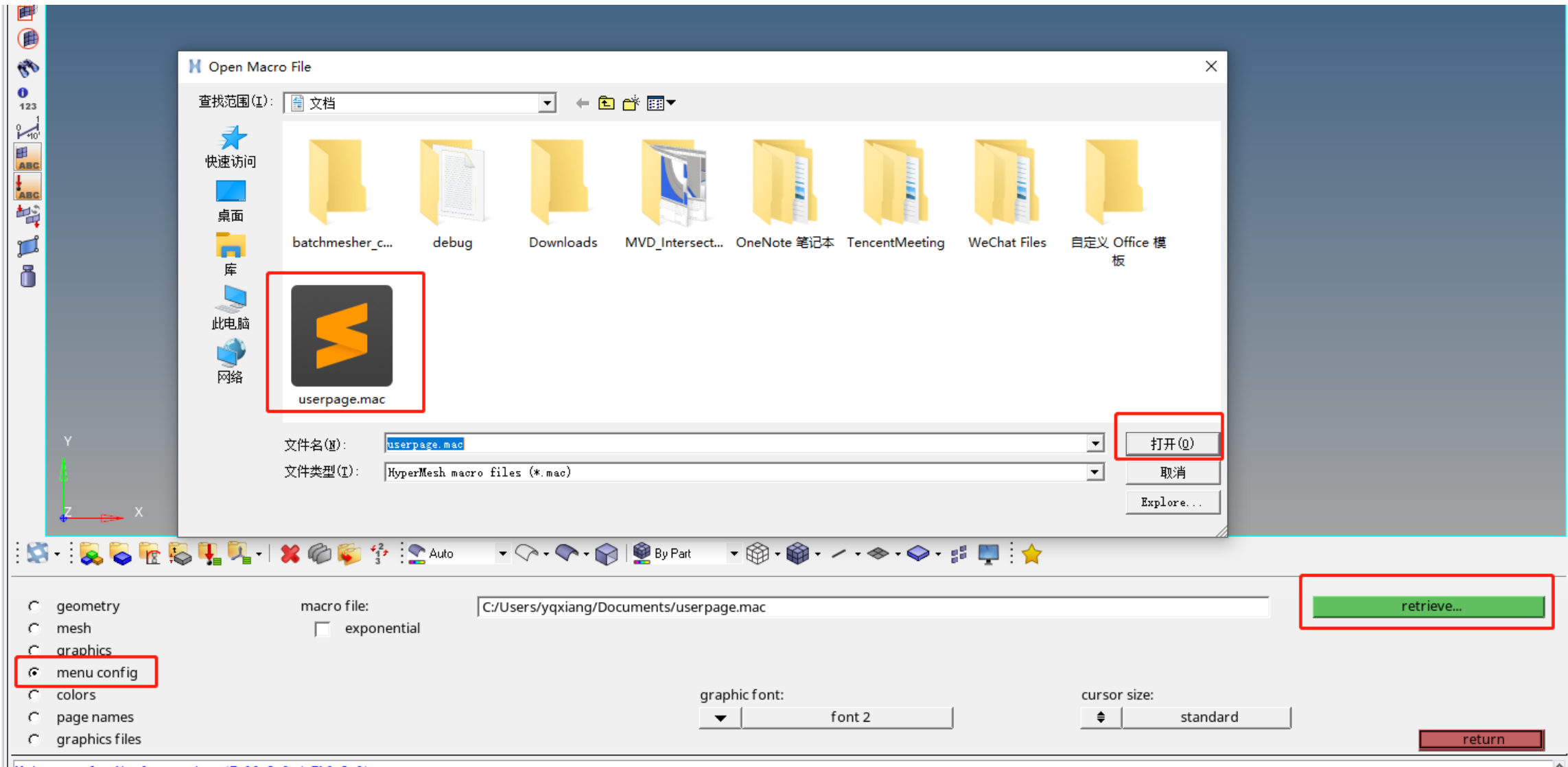
## HyperMesh开发-宏按钮

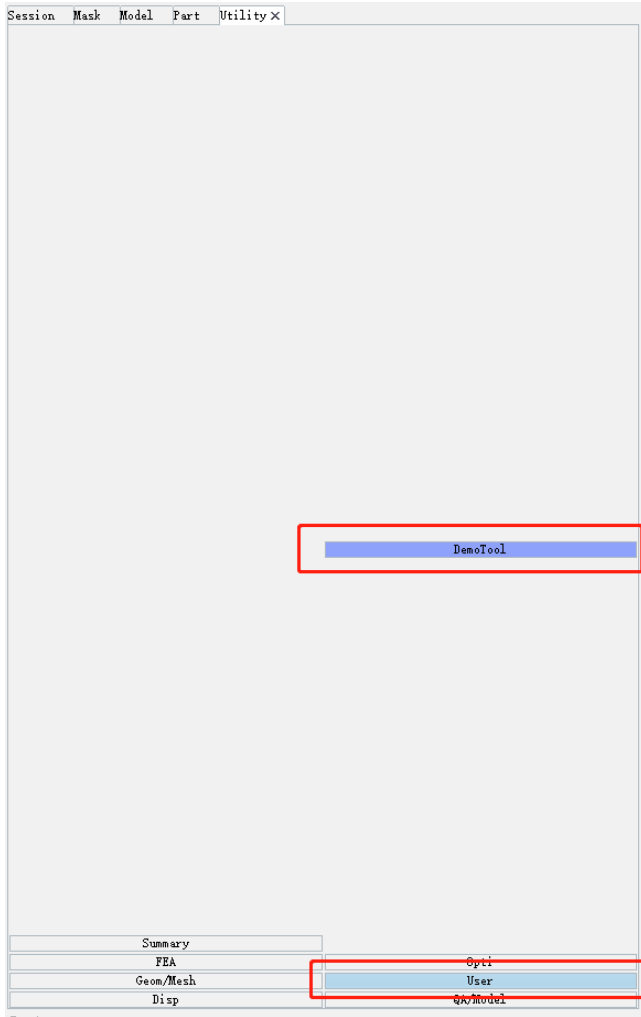
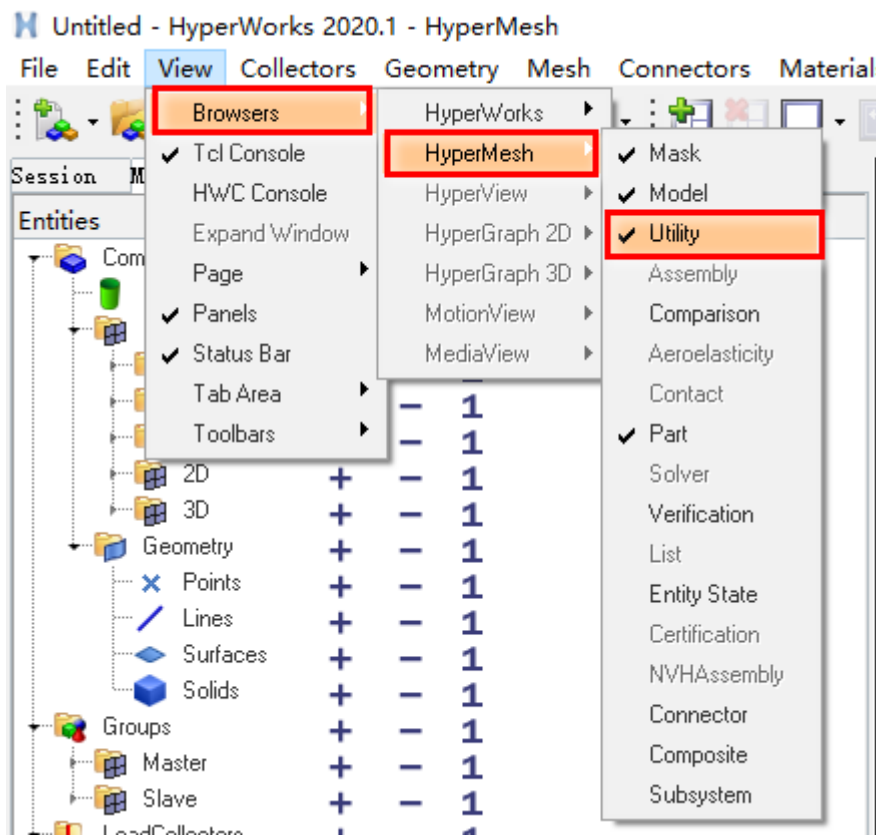
### 如何创建宏（使用tcl）

1. 创建demo.tcl文件，保存在HyperMesh工作目录(一般在我的文档)下；
2. 创建userpage.mac文件，存储在HyperMesh工作目录
3. 在userpage.mac文件中写入宏设置语句，调用该tcl文件；  
    \*createbutton(5,"DemoTool",10,5,5,YELLOW,"Run tcl script","EvalTcl","demo.tcl")
4. 标签栏Preference>Menu Config,加载userpage.mac，最后点击return







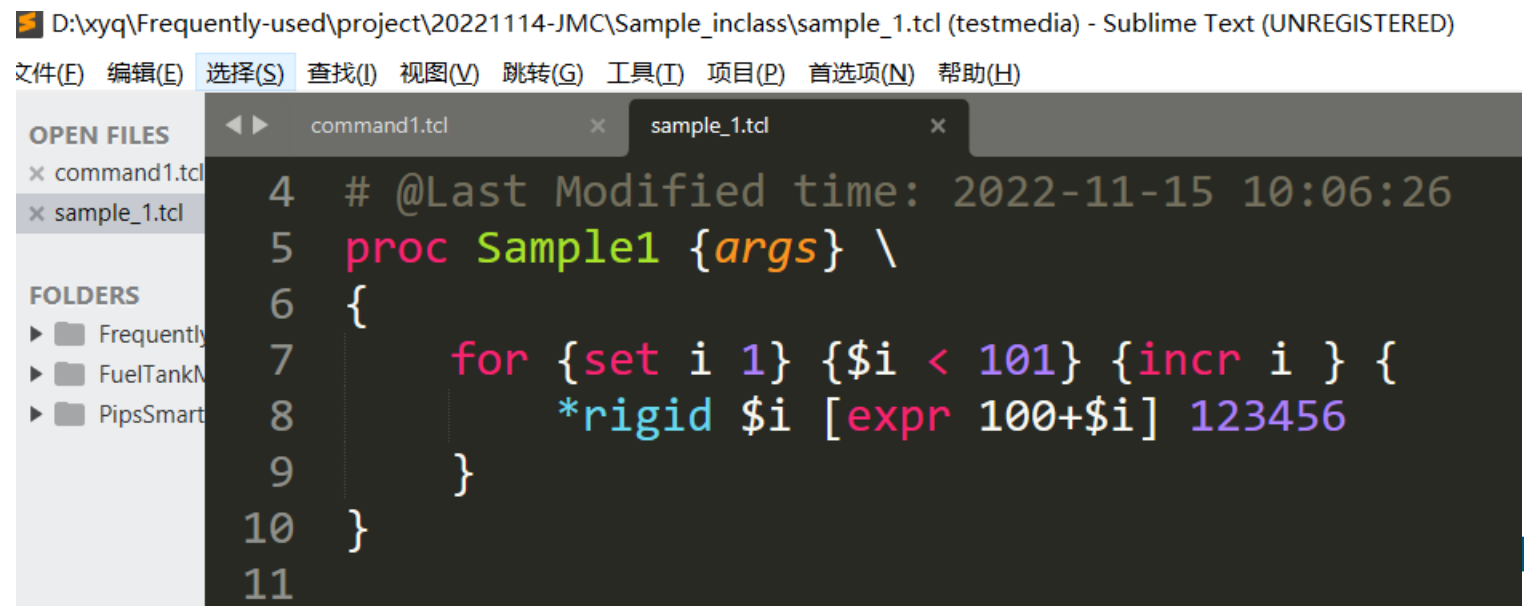
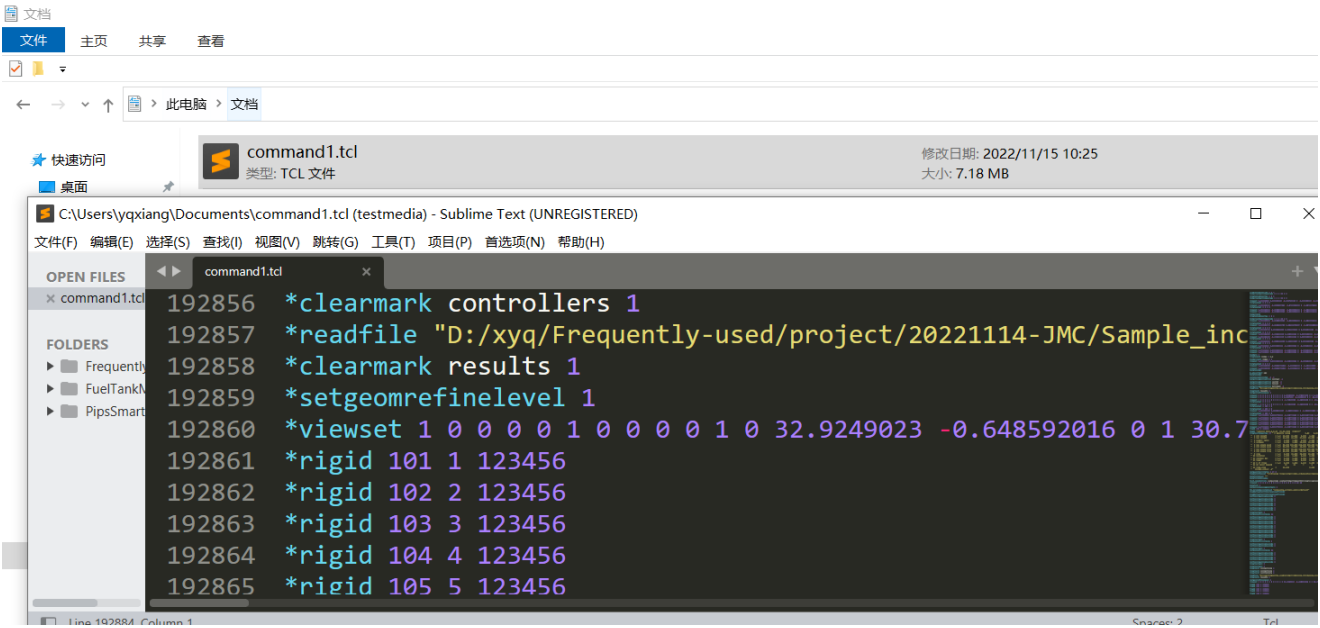


# 二次开发基本流程

- 界面操作
- 打开command文件，获取相关命令
- 编写相关命令
- 测试

关闭右键视角变化命令记录

hm\_writeviewcommands 0



# Command and Function

- Data Names

HM 中数据类型

- HM Tcl GUI command

HM GUI 交互相关

- HM Tcl Modify Commands

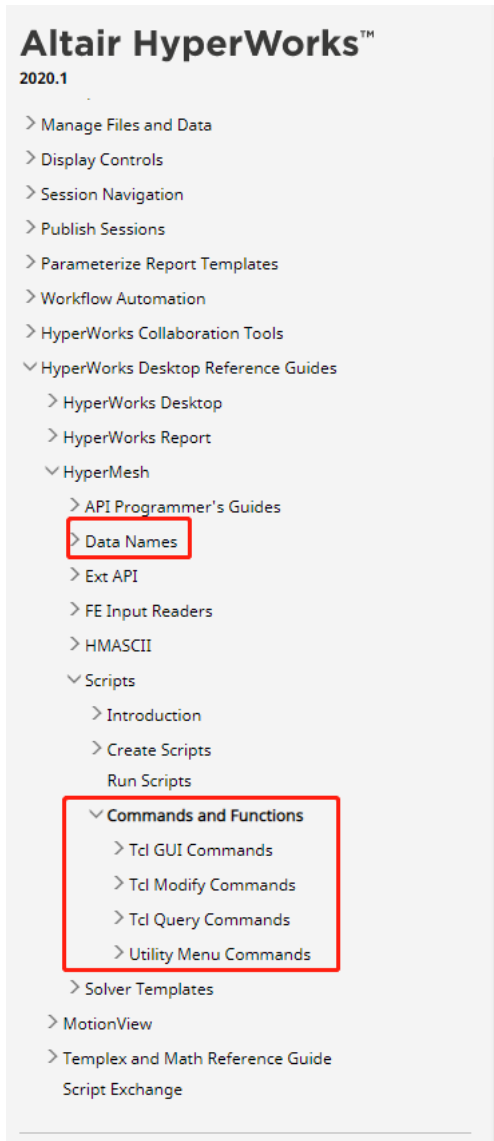
修改 HM内数据

- Hm Tcl Query Commands

获取HM内数据

- Utility Menu Commands

宏按钮等相关



# 一些常用的API及使用方法

## 内存容器命令

hm\_createmark

hm\_getmark

hm\_marklength

\*createlist

hm\_getlist

## HyperMesh实体交互命令

\*createmarkpanel

\*createlistpanel

## 核心命令

hm\_getvalue

\*setvalue

To delete components with names FRONT and SIDE:

```
hm_createmark comps 1 "FRONT SIDE"  
*deletemark comps 1
```

To delete components with names "Name with spaces" and SIDE:

```
hm_createmark comps 1 "{Name with spaces} SIDE"  
*deletemark comps 1
```

or

```
set names [list {Name with spaces} SIDE]  
hm_createmark comps 1 $names  
*deletemark comps 1
```

To mark the last 3 components that were created:

```
hm_createmark comps 1 "by id only" "-1 -2 -3"
```

To delete all elements in the database:

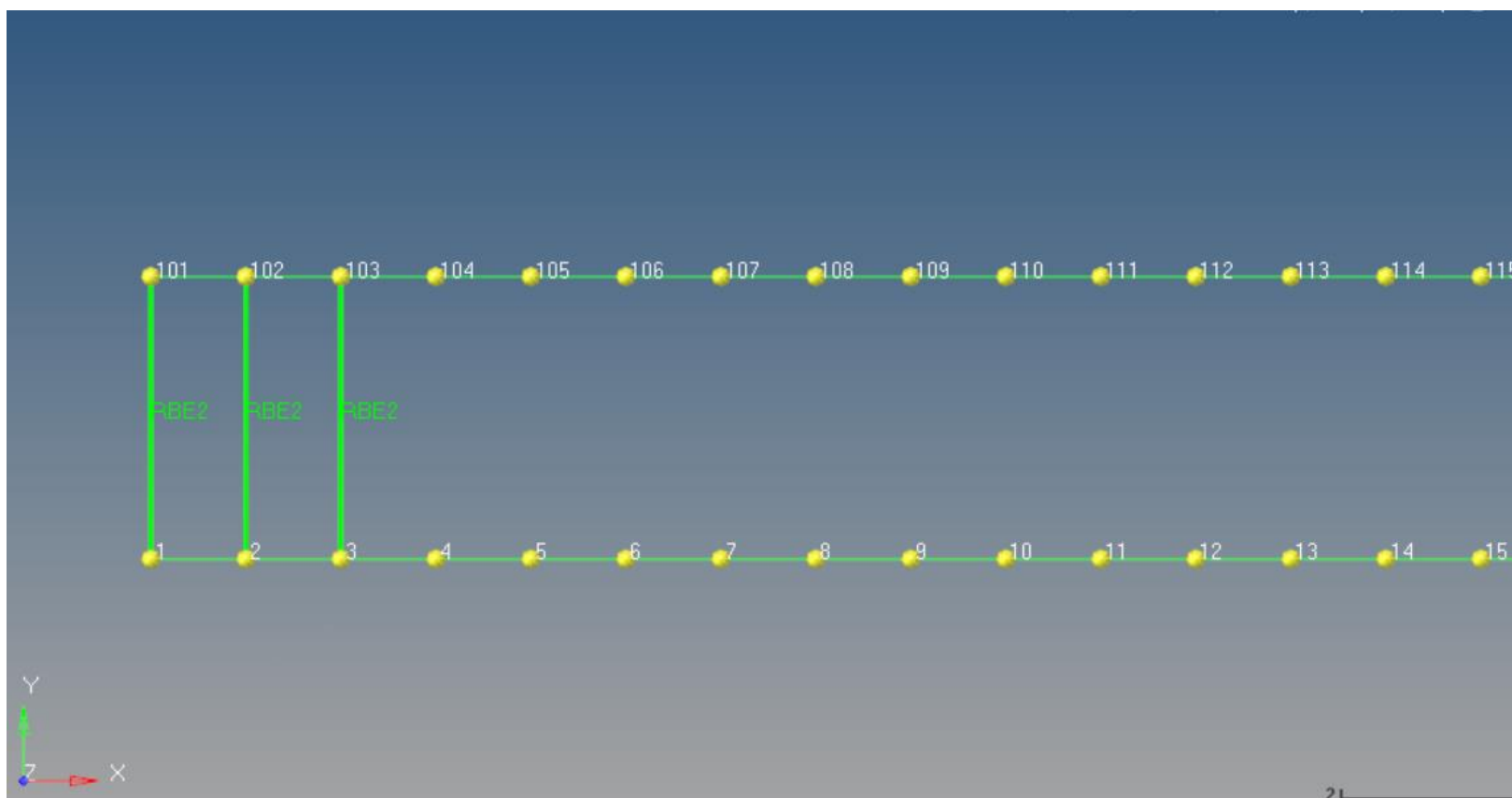
```
hm_createmark elems 1 "advanced" "all"  
*deletemark elems 1
```

To delete displayed elements:

```
hm_createmark elems 1 "advanced" "displayed"  
*deletemark elems 1
```

## 实例一：巧用 for 循环 (Day1)

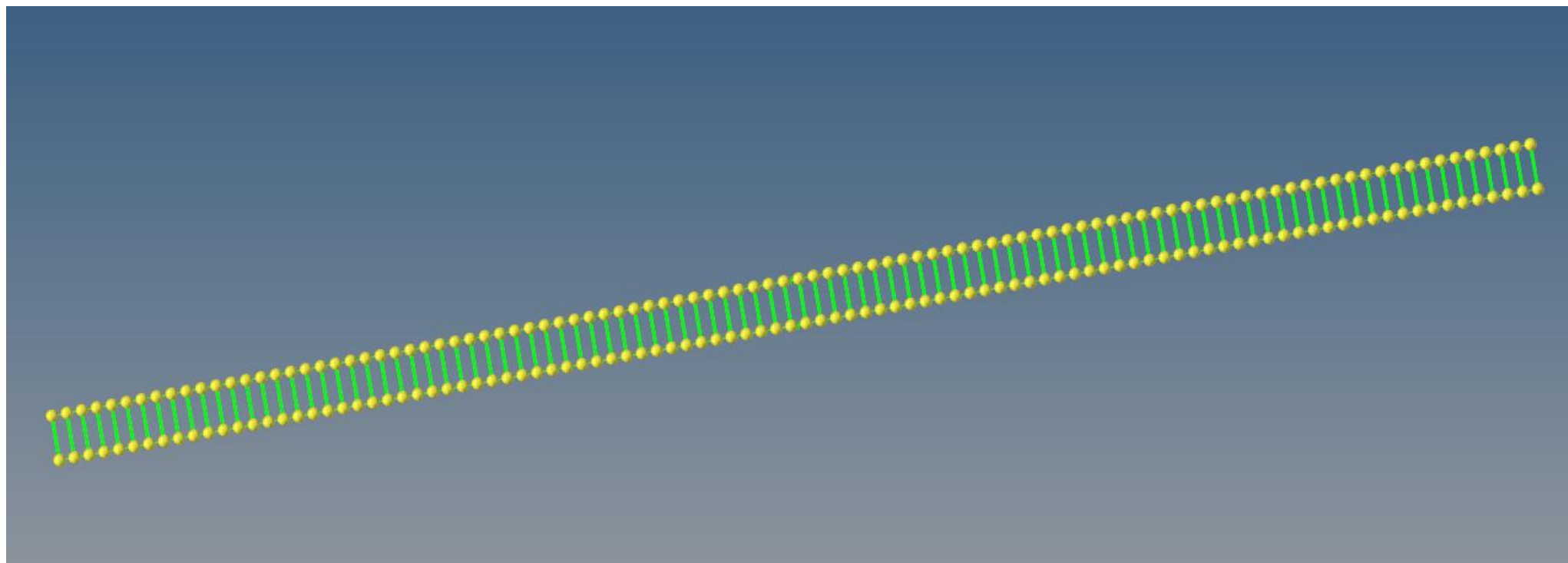
- 如果有两排分别100个节点，如何对这两排节点一一使用rigid进行快速焊接；  
(提示：手动焊接两次，在command.tcl中找出规律)



## 实例一：巧用 for 循环 (Day1)

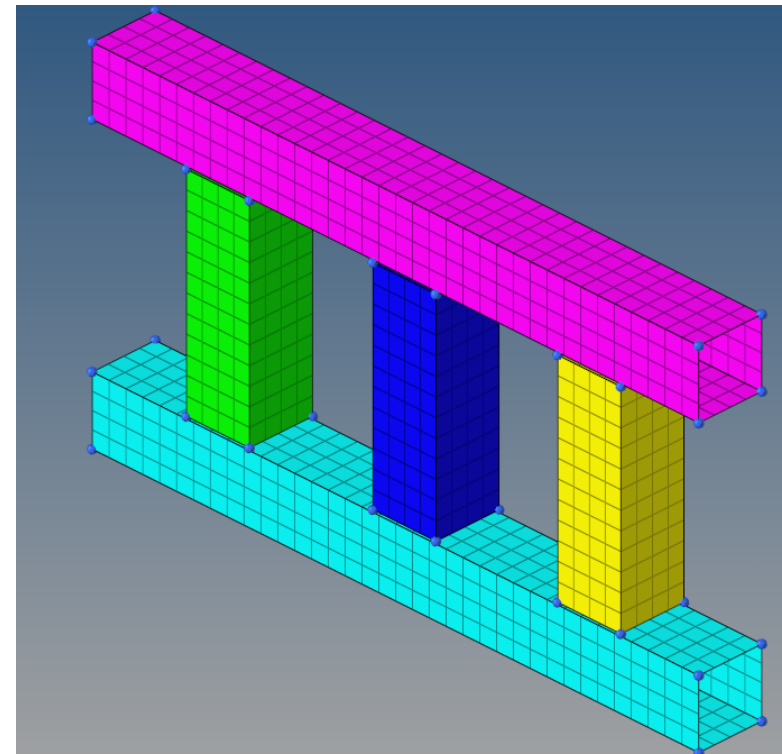
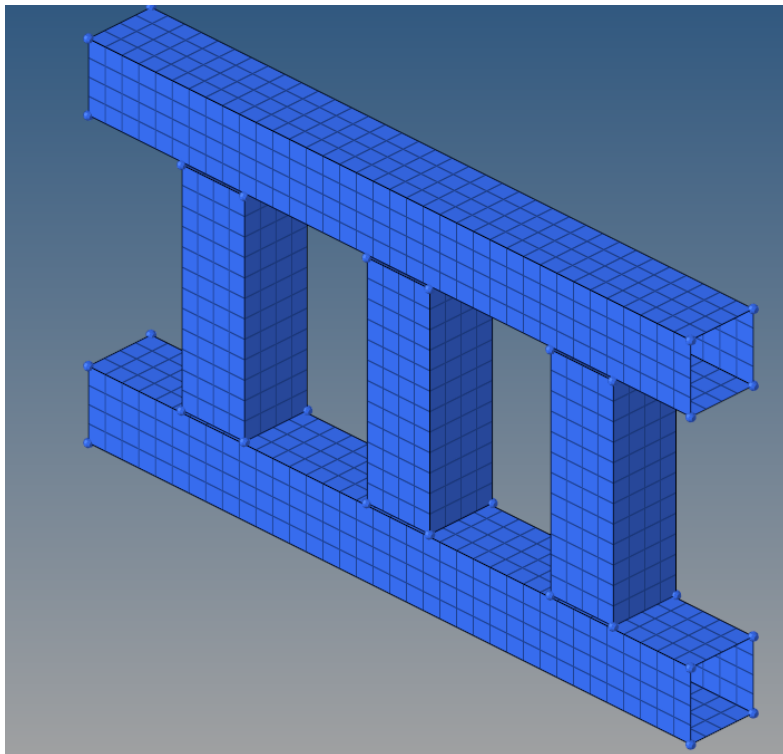
解题思路：

- 1、在command.tcl中记录两条rigid的生成命令；
- 2、结合第一步的记录，使用for循环将这100个rigid单元生成



## 实例二：在大模型中如何将离散的Comp划分成独立的Comp（Day2）

- 一个离散的Comp, 如何将每个离散的部分, 都分解成一个独立的Comp。  
(提示: 使用while循环, 和 `hm_createmark` , `*appendmark` , `hm_getmark` )





## 实例三：创建一个界面，通过界面中的参数自动创建材料(Day2)

---

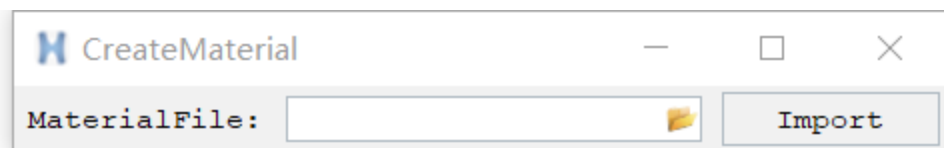
- 如题，创建一个界面，通过用户在界面上交互式输入参数，点击按钮，即可创建材料。

## 实例三：创建一个界面，通过界面中的参数自动创建材料(Day2)

- 如题，创建一个界面，通过读取材料文件（csv格式），点击按钮，即可创建材料。

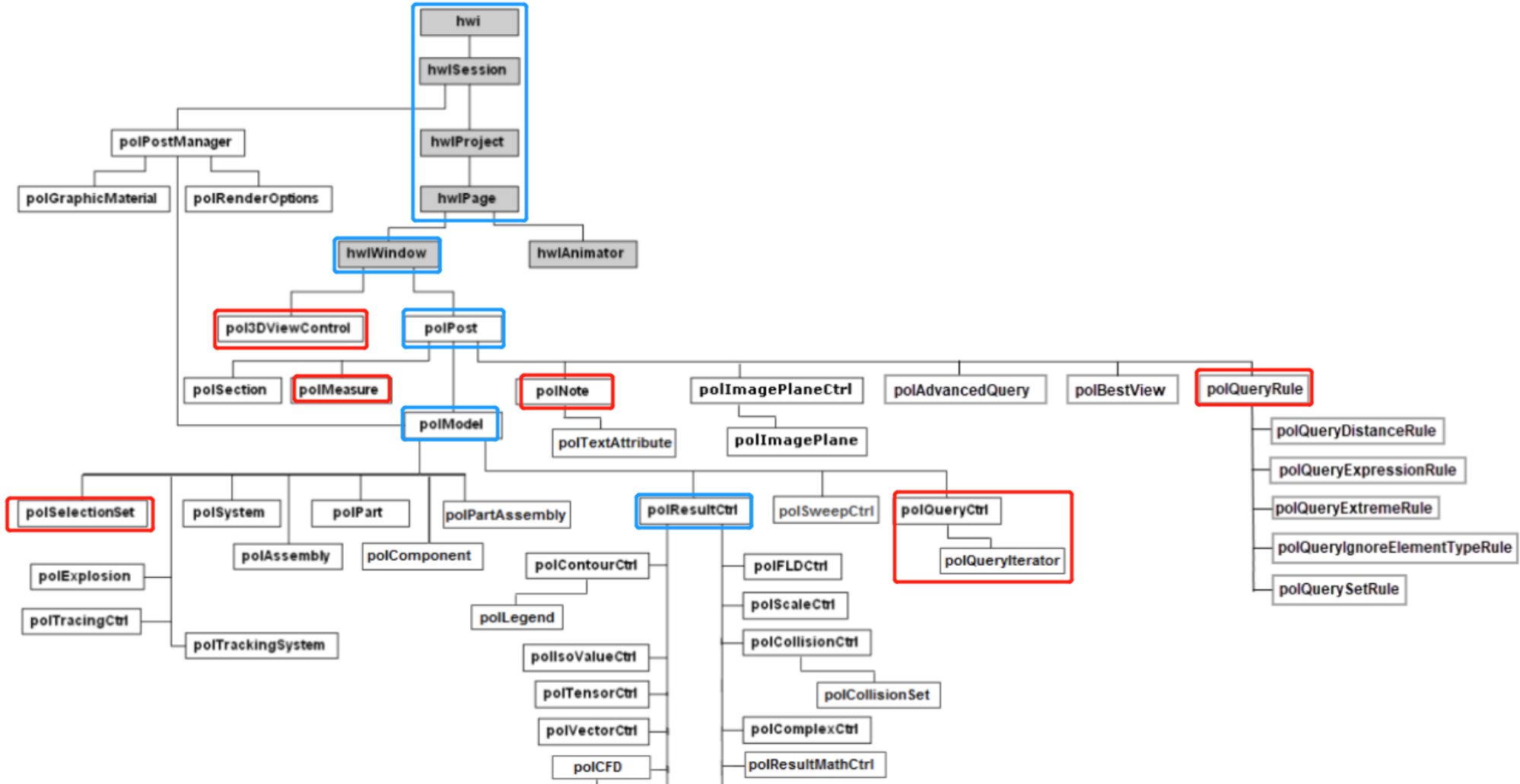
参考代码见附件中的 `sample_3_CreateMaterial.tcl`  
调用界面的子程序为 `::importmat::TestImportMat`

1. 创建窗口，tk相关
2. 读取材料文件
3. 创建材料属性

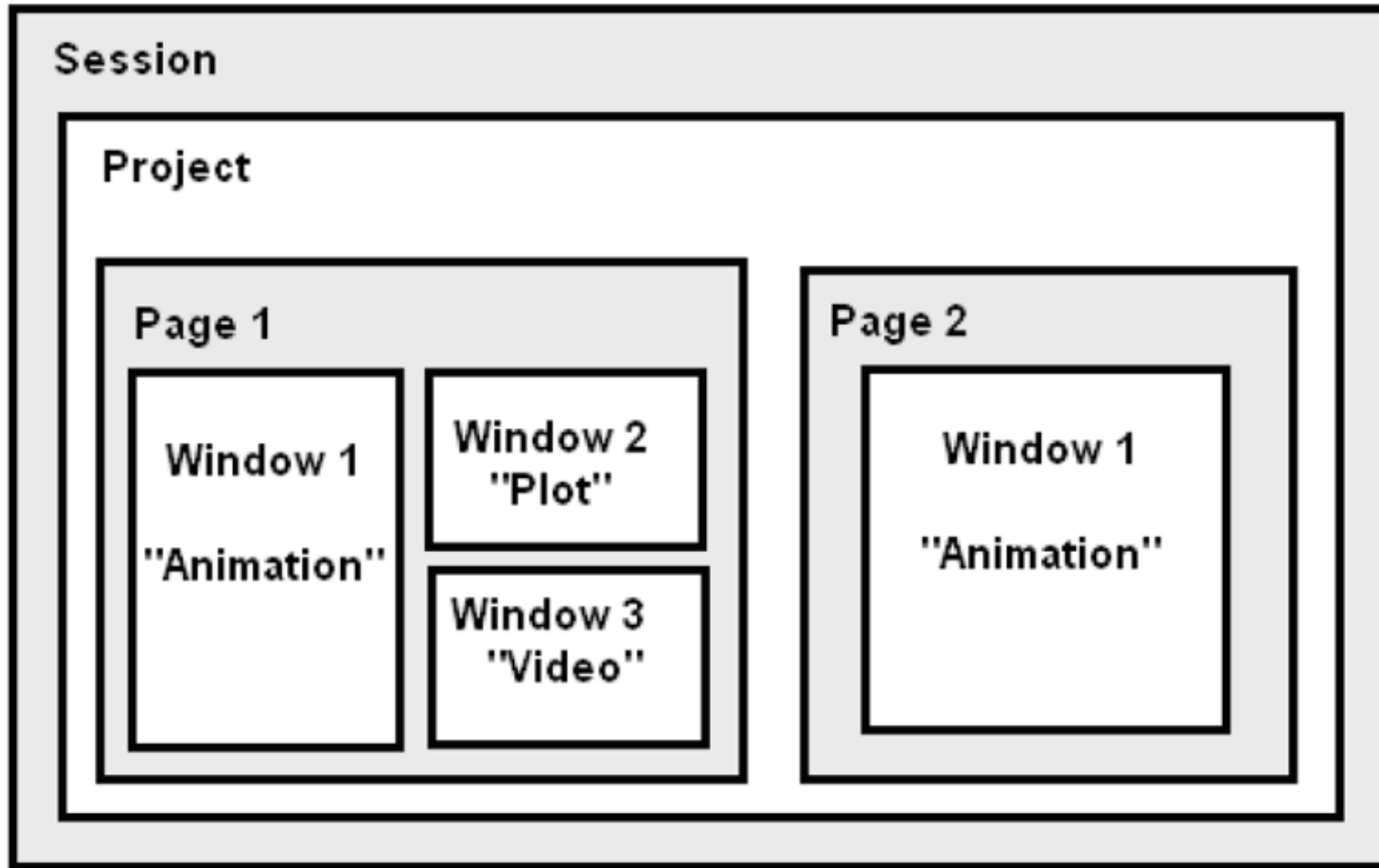


# 后处理

# 这是一个层级的世界



# HyperWorks框架与通用层级的作用范围



## 后处理二次开发的基本思路：

- 1、2019及之前版本没有历史操作记录文件参考；
- 2、查找你的操作或者查询所在的对象层级；
- 3、从hwi 开始逐级任命你的各个层级的对象；
- 4、当任命到你需要操作或查询的命令所在的对象时，调用操作或者查询命令，完成动作；
- 5、动作都完成后，一定要**关闭所有任命的对象**；

## 后处理各层级对象的通用规则：

后处理命令通用格式：

{object\_name} {command} {command parameter(s)}

后处理层级对象支持的命令：

- Handle Commands ; 每一个对象都支持对象相关命令；
- Property Commands Query Options ; 每一个对象都有属性查询命令；
- List Commands ; 每一个对象都有列表查询命令

# 后处理命令示例:

## 例1: 新增页面

hwi OpenStack

```
hwi GetSessionHandle sess
sess GetProjectHandle proj
proj AddPage
```

hwi CloseStack

```
proc AddPage {args} \
{
```

```
set t [clock clicks]
```

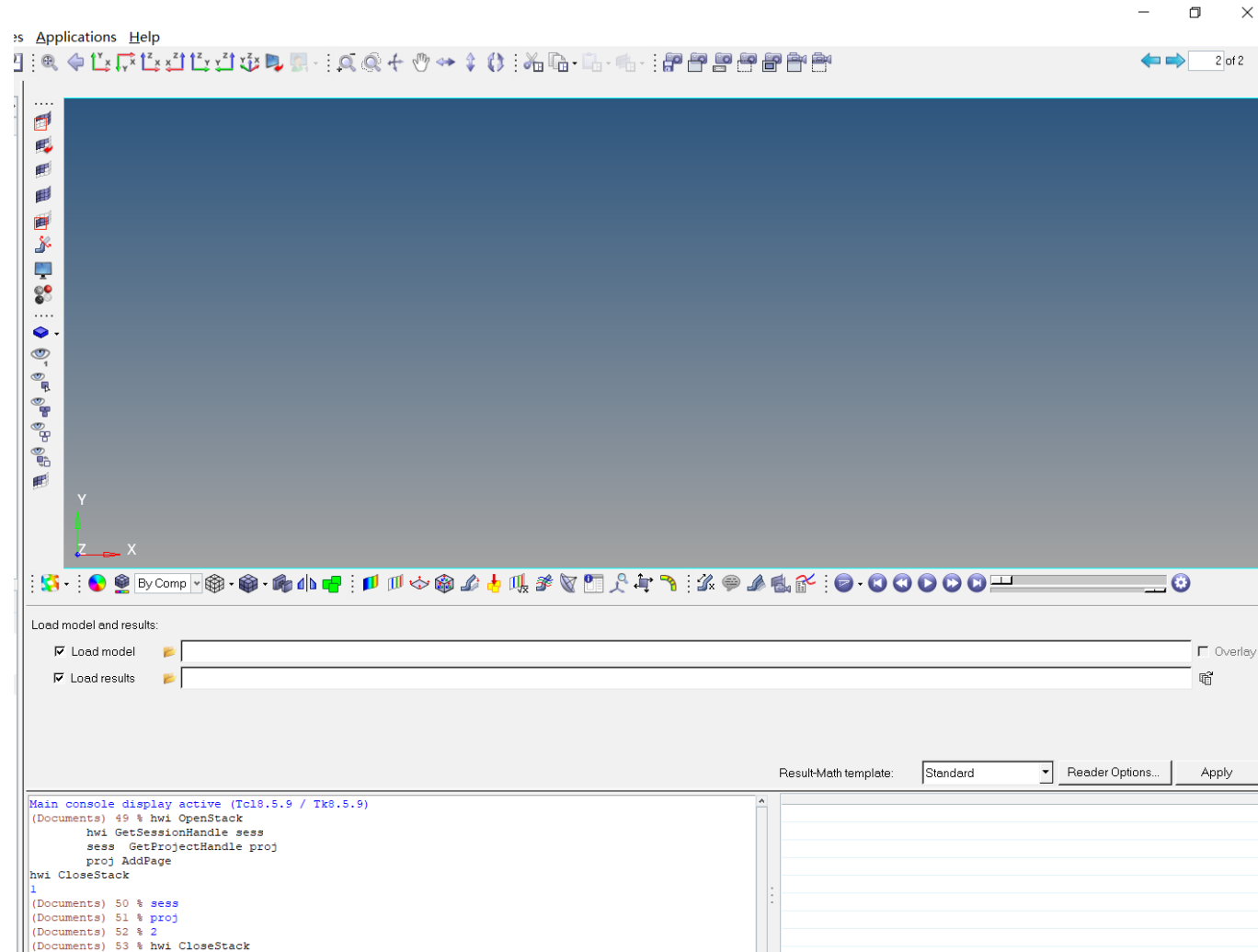
hwi OpenStack

```
hwi GetSessionHandle sess$t
sess$t GetProjectHandle proj$t
set pageid [proj$t AddPage]
proj$t SetActivePage $pageid
```

hwi CloseStack

```
return $pageid
```

```
}
```





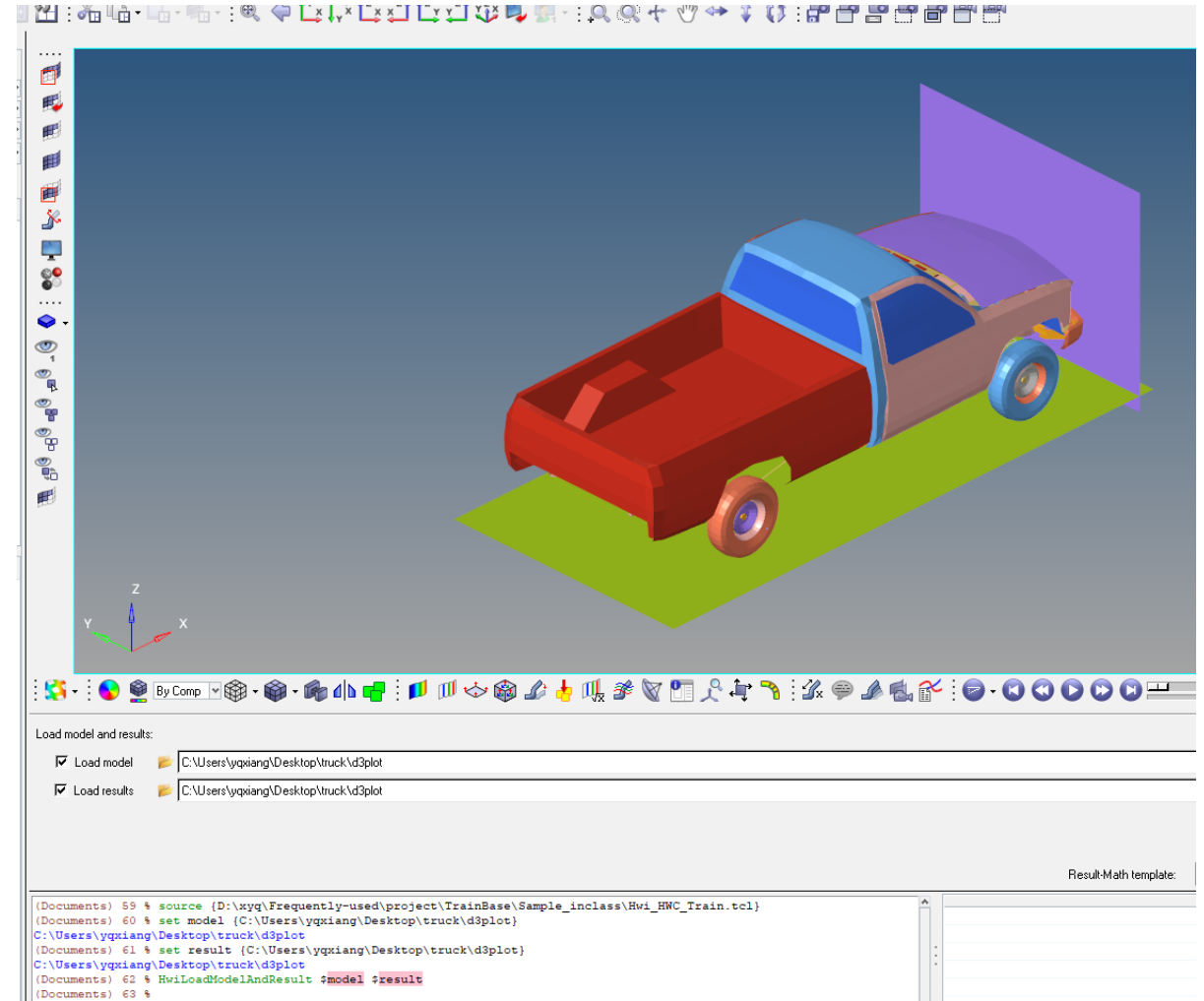
## 后处理命令示例:

### 例2: 载入模型和结果

```

proc HwiLoadModelAndResult {model result} \
{
    set t [clock clicks]
    hwi OpenStack
        hwi GetActiveClientHandle cln$t
        # 设置模型文件
        cln$t AddModel $model
        cln$t GetModelHandle mdl$t 1
        # 设置结果文件
        mdl$t SetResult $result
        # 刷新界面
        cln$t Draw
    hwi CloseStack
}

```

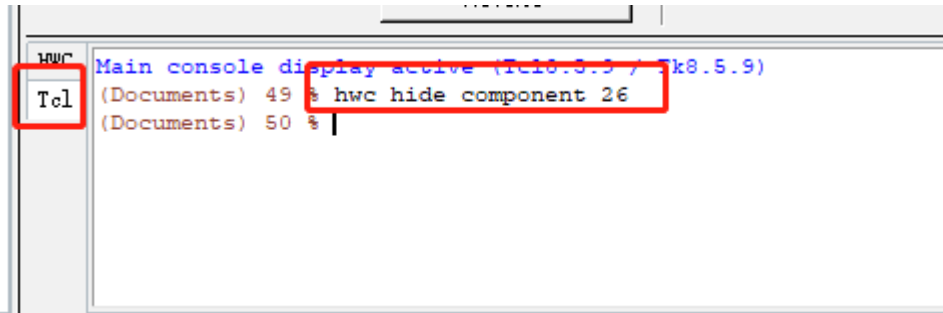
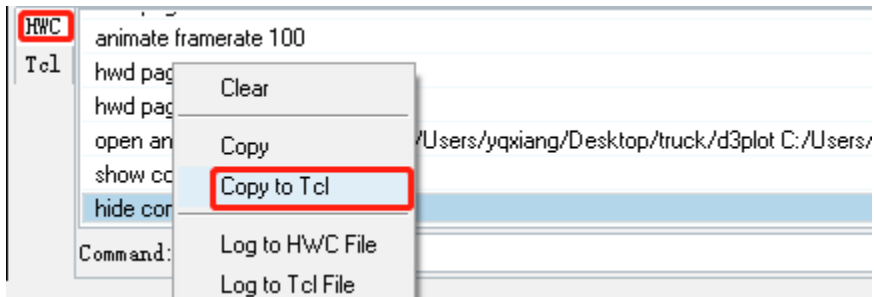
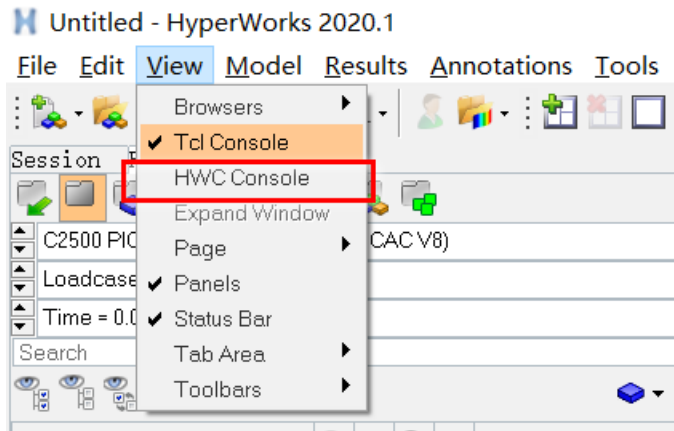


## HWC介绍:

- 1、2019之后版本可以通过HWC命令实现图形界面的快速调整;
- 2、用户操作图形界面, 例: 载入结果,加载云图, 创建note或者measure;
- 3、在HWC面板中将命令选中右键复制为tcl命令;
- 4、在tcl console中运行HWC命令;
- 5、主要支持的功能: 载入结果,加载云图, 创建注释, 视角调整, 截图, 组件/单元显示或隐藏;

# HWC介绍:

- 1、在view标签中勾选HWC Console，底部命令框出现HWC Console；
- 2、在HWC面板中将命令选中，右键复制为tcl命令；
- 3、在tcl console中运行HWC命令；



# HWC介绍:

**Altair HyperWorks™**  
2020.1

- HyperWorks Desktop
  - External Readers and Altair Binary Format
  - Generic ASCII Reader
  - Batch Mode
  - External Resources
  - Tcl/Tk Commands
  - Translators
  - Result Math
- HyperWorks Record and Playback
  - Introduction
  - Work with the HWC Console
  - Interactive Help/Intellisense
  - HWC Console Command Syntax
  - Record Views
  - Command Files
  - Execute HWC Files
  - HWC and Tcl**
  - Error Messages
  - HWC Commands**
    - animate
    - annotation
    - attribute
    - color
    - config
    - delete
    - hide
    - hwd
    - mark

**HWC and Tcl**  
Use HWC commands in the Tcl Console.

**HWC Wrapper**  
The `hwc` wrapper is on the same root level as the `hwi` parent handle. The wrapper allows you to execute the HV prefix for this command:

```
hwc <HWC Command>
```

**Example**  
The HWC command to hide all components which contain the string "door" in their label:

```
hide component *door*
```

can be executed in Tcl as follows:

```
hwc hide component *door*
```

Multiple commands can also be executed in one line:

```
hwc hide all | show component *door*
```

The advantage is that no handle needs to be created; the wrapper class always exists. The command can be used as follows:

**Important:** Recorded commands which contain name=value pairs are recorded with a blank space between these variables are of a type which can contain backslashes. If multiple entities are selected typing a command with multiple entities and the " " This blank space is required for the Tcl parser to work.

**Important:** Please do not remove the blank spaces that HWC automatically inserts.

**Example**  
The blank space between `text=` and `"Value` is required for the command to work:

```
hwc annotation note "Note 4" display text= "Value \\"Force\\" = {entity.id}"
```

All HWC commands from the command history can be copied in the Tcl syntax to the clipboard using the **Copy to console:**

## 后处理二次开发的流程思路：

HWI和HWC相结合的方式，HWC没有返回值，利用HWI查询结果文件数据，通过HWC处理图形内容。

1. 新建页面，设置页面布局，调整窗口类型；
2. 加载结果文件；
3. Subcase,Simulation以及Animation设置；
4. 云图设置；
5. 添加note/measure；
6. 读取数据
7. 截图
8. 创建报告

# THANK YOU

更多资讯，欢迎关注Altair微信公众号



官网 : [www.altair.com.cn](http://www.altair.com.cn)

技术博客 : [blog.altair.com.cn](http://blog.altair.com.cn)

邮箱 : [info@altair.com.cn](mailto:info@altair.com.cn)

技术服务热线: 400-619-6186