

## Re: equality and errors

**Source:** <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2003-12/0612.html>

---

**From:** Aric Bills ([aricb\\_at\\_u.washington.edu](mailto:aricb_at_u.washington.edu))

**Date:** 12/11/03

Date: Thu, 11 Dec 2003 10:53:55 -0800

Hi Marvin,

This seems to me like a non-standard use of the catch mechanism. Normally, I use catch to warn me if something is going to break so I can keep it from breaking, like if I need to do division where the denominator might be zero, for example.

If I understand right, you have a proc which validates the contents of two entry widgets. If there is a problem, it throws an error, which the calling code catches. That code then throws another error. That structure is reasonable, although there's a problem with one of your catch statements. If I were writing this program, though, I would use a different approach besides catch and error.

```
> set fail [catch { $first < 1 } range1]
> if {$fail} {
> error "range1 error"
> }
```

The catch above is not doing what you think it's doing... It's going to try to execute the command [`$first < 1`]. Unless `$first` contains a Tcl command that takes two arguments, catch is going to catch this error: "invalid command name <whatever was in \$first>". What you want here, instead of [catch], is [if] :

```
if { $first < 1 } {
    error "value ($first) is less than 1"
}
```

If I were writing the program, I would have the errorProc return a value telling 1) whether there was an error and 2) what the error was. Then I'd have the calling code respond intelligently to that value, either with an error message, a popup window, or some code to read the user's mind and fix his/her mistakes automagically. Here's a quick implementation that might give you some ideas:

```
proc errorProc { val1 val2 } {
    if { $val1 < 0 && $val2 < 0 } {
```

comp.lang.tcl: Re: equality and errors

```
    return "val1 and val2 are too small"
  } elseif {$val1 < 0} {
    return "val1 too small"
  } elseif {$val2 < 0} {
    return "val2 too small"
  } else {
    return ""
  }
}

proc button_code {} {
  set a [Toplevel1.Entry1 get]
  set b [Toplevel1.Entry2 get]

  set errorMsg [errorProc $a $b]
  if {$errorMsg == ""} {
    # do whatever the button does
  } else {
    tk_messageBox -message $errorMsg
  }
}
```

To use this code with your button, you would configure the button to call `button_code` when pressed (using the `-command` option).

Regards,  
Aric