Subject: BUG? EditInt / PromptOK not functioning as thought?
Posted by ptkacz on Sun, 14 May 2017 18:29:26 GMT
View Forum Message <> Reply to Message

When an EditInt field is populated with a valid value (i.e. 0..999; 4 allowable digits), the PromptOK box displays as expected upon the clicking of a submit button.

When an EditInt field is populated with a number greater than the max allowable value, the field turns red, and the incorrect PromptOK dialog box displays. In order to get the PromptOK window to display with the correct message, the close window button needs to be selected.

What's going on here? Code listed below.

Peter

Bug1.lay

```
LAYOUT(Bug1Layout, 84, 60)
 ITEM(EditInt, editField, Max(999).Min(0).MaxChars(4).LeftPosZ(8, 64).TopPosZ(8, 19))
 ITEM(Button, submitButton, SetLabel(t_("submit")).LeftPosZ(8, 64).TopPosZ(36, 15))
END_LAYOUT
```

Bug1.h

```
#ifndef _Bug1_Bug1_h
#define _Bug1_Bug1_h

#include <CtrlLib/CtrlLib.h>

using namespace Upp;

#define LAYOUTFILE <Bug1/Bug1.lay>
#include <CtrlCore/lay.h>

class Bug1 : public WithBug1Layout<TopWindow> {
public:
 typedef Bug1 CLASSNAME;
 Bug1();
 void buttonClicked(void);
};
#endif
```

main.cpp

```
#include "Bug1.h"

Bug1::Bug1()
{
 CtrlLayout(*this, "Bug1 Window");

    submitButton << THISBACK(buttonClicked);
}


void Bug1::buttonClicked(void) {
 int value = editField;

 if( value > 999 ) {
  PromptOK("Number too big!");
  return;
 }
 else
  PromptOK("Number accepted!");
}


GUI_APP_MAIN
{
 Bug1().Run();
}
```

Subject: Re: BUG? EditInt / PromptOK not functioning as thought?
Posted by Klugier on Sun, 14 May 2017 19:17:23 GMT
View Forum Message <> Reply to Message

Hello,

It is classical example of "Working as design bug". When you use Min and Max attribute of EditField you don't need to control the validation of the field value. It will happen automatically and if the number is wrong it will mark the field with red co and hook at the end of window life time. I am not 100% certain when the prompt will be displayed, but i based on my previous experience.

So if you want to disable this validation - simply remove Min and Max from the EditField layout property.

Sincerely,
Klugier

Subject: Re: BUG? EditInt / PromptOK not functioning as thought?
Posted by dolik.rce on Sun, 14 May 2017 19:33:35 GMT
View Forum Message <> Reply to Message

Hi Peter,

I haven't check the code, but if I remember it correctly the check you perform should actually look like this:

```
if (IsNull(value)) {
  PromptOK("Invalid number!");
  return;
 }
 else
  PromptOK("Number accepted!");
```

The edit field returns Null value when it is empty, invalid or outside of the allowed range.

Best regards,
Honza

Subject: Re: BUG? EditInt / PromptOK not functioning as thought?
Posted by ptkacz on Mon, 15 May 2017 03:21:17 GMT
View Forum Message <> Reply to Message

Thanks guys.

The IsNull(editField) works.

There's still a problem though. As long as the value that's populated within the edit field is outside the specified range on the edit field, one has to correct the problem, or one can't exit out of the application when selecting the close application button (i.e. 'x'). Selection of the close application button results in a pop-up window displaying the message, "Number must be between 0 and 999".

The message is kind of vague since it doesn't indicate which field has the issue.

Wonder what happens where there are multiple issues?

Peter

Subject: Re: BUG? EditInt / PromptOK not functioning as thought?
Posted by Zbych on Mon, 15 May 2017 18:37:14 GMT
View Forum Message <> Reply to Message

ptkacz wrote on Mon, 15 May 2017 05:21As long as the value that's populated within the edit field is outside the specified range on the edit field, one has to correct the problem, or one can't exit out of the application when selecting the close application button (i.e. 'x')

I must admit that this behaviour is really annoying, you can set Rejector to IDCANCEL in window constructor to disable it:

```
Bug1::Bug1()
{
 Rejector(*this, IDCANCEL);//<----------------------
 CtrlLayout(*this, "Bug1 Window");
 submitButton << THISBACK(buttonClicked);
}
```

Subject: Re: BUG? EditInt / PromptOK not functioning as thought?
Posted by ptkacz on Tue, 16 May 2017 03:40:22 GMT
View Forum Message <> Reply to Message

That's interesting, the Rejector.

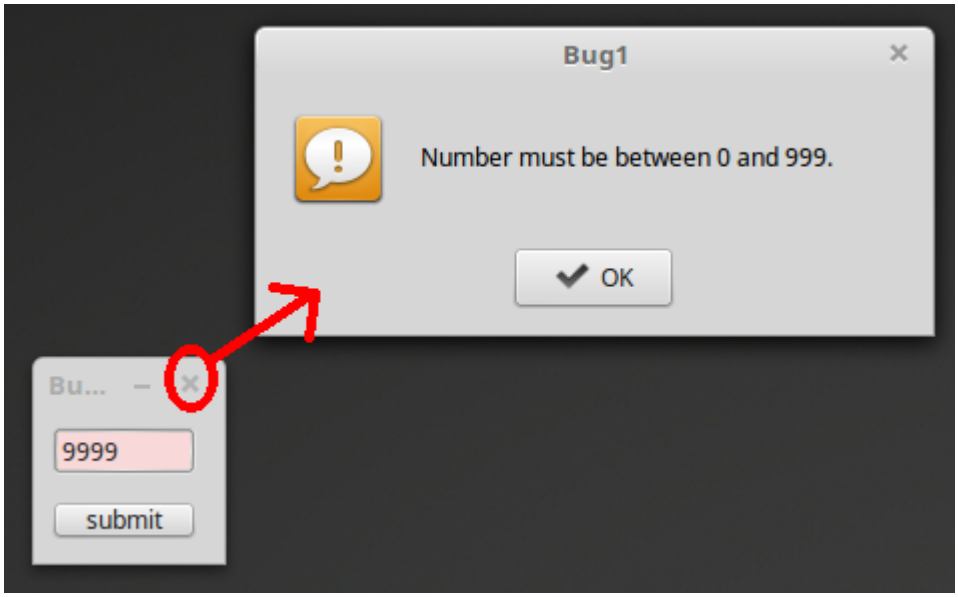
I'm guessing that the functionality should really work by popping up the message when the field loses focus? Similar to how the field turns red when validation is applied against it.

How about instead, make it such that if a field entry is in error, a tool tip becomes enabled on the field. So that when the mouse hovers over the highlighted edit field, a tool tip displays indicating what the issue is? A simpler solution?

Here's a screen shot of the issue:

File Attachments

1) BugScreenShot.png, downloaded 1050 times



Subject: Re: BUG? EditInt / PromptOK not functioning as thought?
Posted by mirek on Sat, 24 Jun 2017 14:00:06 GMT
View Forum Message <> Reply to Message

ptkacz wrote on Tue, 16 May 2017 05:40That's interesting, the Rejector.

I'm guessing that the functionality should really work by popping up the message when the field loses focus? Similar to how the field turns red when validation is applied against it.

I do not think that would be good - as general solution. Sometimes the error is a result of combination of values, so associating it with value of single field is probably not a good idea.

In general, the 'annoying behaviour' is mostly the result default settings.

In U++, you dialog can be 'breaked' (no errors displayed, nothing else is done), 'accepted' (errors are checked and dialog does not exit if there are errors found) or 'rejected' (no errors displayed AND dialog is restored to values before Run). See http://www.ultimatepp.org/srcdoc$CtrlLib$Tutorial$en-us.html ch. 18, 20

Now the problem is with the default settings of WhenClose callback that gets invoked by clicking "window close" [X] button. Default is to call TopWindow::Close virtual method, which is defined this way:

```
void TopWindow::DefaultBreak()
{
 if(FindAction(IDCANCEL) || close_rejects)
  RejectBreak(IDCANCEL);
```

```
 else
 if(FindAction(IDNO))
  RejectBreak(IDNO);
 else
 if(FindAction(IDEXIT))
  AcceptBreak(IDEXIT);
 else
 if(FindAction(IDYES))
  AcceptBreak(IDYES);
 else
  AcceptBreak(IDOK);
}
```

That means it reacts on presence of different types of 'breakers' and if there is none defined, defaults to 'accept'. However, fix is simple, all you need is to override Close or set WhenClose to something else, e.g.

```
app.WhenClose = [=] { app.Break(); };
```