Subject: Extending EditString to filter keys
Posted by Jeremi on Fri, 11 Feb 2011 10:54:38 GMT
View Forum Message <> Reply to Message

Hi all
I am a beginner in Upp. Very cool lib!!
I am trying to make an edit control to acept only letters (a-z and A-Z) so I extended a new class from EditString and i am trying to implement it, like this:

```
#ifndef _MYEDIT_h
#define _MYEDIT_h
#include <CtrlLib/CtrlLib.h>
using namespace UPP;
class MyEditString : public EditString
{
public:
 typedef MyEditString CLASSNAME;
       MyEditString();
 ~MyEditString();
 virtual bool  Key(dword key, int repcnt);

private:

};
#endif
```

```
#include "MyEditString.h"

MyEditString::MyEditString()
{
}

MyEditString::~MyEditString()
{
}

bool MyEditString::Key(dword key, int repcnt)
{
 if(key == K_F)
 {
              Exclamation("f was pressed")
  return true;
 }
 return EditString::Key(key, repcnt);
}
```

Then I added a EditString to a dialog and change its type from EditString to MyEditString (now just to detect the F and prevent it from appearing in edit box!)

It works. It detects f and it prevents it from appearing in the edit box but if I remove the Exclamation the f appears in the edit box.

I thought that when the return true we are saying that the event is already handled, but returning true the f appears and I want to prevent it from appear.

Any help?

Thank to all of you

Jeremi

Subject: Re: Extending EditString to filter keys
Posted by dolik.rce on Fri, 11 Feb 2011 13:08:32 GMT
View Forum Message <> Reply to Message

Hi Jeremi,

There is a litle bit simpler way. All the U++ edit fields based on EditField have a method SetConvert(const Convert&), which allows to handle how seting and retrieving values work and, most importantly for you, it allows to filter which characters are alowed.

All you need to do is to create a class inherited from Convert and override its Filter method. The code could be something simple such as:class MyConvert:public Convert{
public:
```
    virtual int Filter(int chr){
        return IsAlpha(chr)?chr:0; //return 0 if we want to discard the character
    }
};
```
Then you can use it as myeditstring.SetConvert(MyConvert()) and it will filter out all characters except letters. For full description of Convert have a look in manual. Nice examples can be found Core/Convert.{h,cpp}.

Best regards,
Honza

PS: I wrote it from top of my head, hopefully there are no big mistakes...

Subject: Re: Extending EditString to filter keys

Posted by Jeremi on Fri, 11 Feb 2011 15:52:41 GMT
View Forum Message <> Reply to Message

Very nice!

It works when I use (from example)

edName.SetConvert(Single<ConvertAlpha>());

Why Single? What does it do? Singleton?

Thank you

Jeremi

Subject: Re: Extending EditString to filter keys
Posted by dolik.rce on Fri, 11 Feb 2011 16:31:30 GMT
View Forum Message <> Reply to Message

Jeremi wrote on Fri, 11 February 2011 16:52It works when I use (from example)

edName.SetConvert(Single<ConvertAlpha>());

Why Single? What does it do? Singleton?
I don't know proper definition of Singleton, but I guess that Single matches it Single<T>() creates static instance of T and any subsequent call to Single<T> returns reference to this variable. It saves memory, because you have only one instance for all the places in code where you use it.

Honza

Subject: Re: Extending EditString to filter keys
Posted by Jeremi on Fri, 11 Feb 2011 17:29:58 GMT
View Forum Message <> Reply to Message

Ok!
Singleton is the same definition for me

Done
Thank you very much

Jeremi