Subject: Converters and Value problem
Posted by mubeta on Thu, 22 Jul 2010 20:44:04 GMT
View Forum Message <> Reply to Message

Hi all,
I am using my own class for converting an integer number in a binary mode:

```
...
Value ConvertBinShort::Scan(const Value& text) const
{
 Value v = UPP::ScanInt((const char*)String(text), NULL, 2);
 if(IsError(v)) return v;
 if(IsNull(v)) return notnull ? NotNullError() : v;
 int m = v;
>>>>>> if(m > 0x7FFF) m = 0xFFFF - m;
 if(m >= minval && m <= maxval) return v;
 return ErrorValue(UPP::Format(t_("Number must be between %X and %X."), minval, maxval));
}
...
```

For every type of number: char, short, int and int64 I have a different class. The same problem for Octal ed Hexadecimal converters.

In fact, I don't found any solution for get the type of the number from Value. Value allow only 32 bit integres, or 64 bit integers. This result in an error when the Scan() method verfy the Min and Max limits.

Now, with my 4 different class: ConvertBinChar; ConvertBinShort, ConvertBinInt and ConvertBinInt64, (and other 4 more for Hex ... for Oct, etc), I am able to "rewind" the resulting number.
This all because Value do not allow char and short numbers, so I am not able to write an unique converter that automatically recognize the numbers limit in byte. (The main problem is when the ScanInt() result in a positive value, instead of negative, and I get an error of Min Max).

Question, (the same from two days ago, but maybe not so clear): can add to the Value also char and short numbers, in the user application?? (All the code will be more light and much more clear than now).

Thanks.

Subject: Re: Converters and Value problem
Posted by mirek on Fri, 23 Jul 2010 08:51:19 GMT
View Forum Message <> Reply to Message

mubeta wrote on Thu, 22 July 2010 16:44Hi all,
I am using my own class for converting an integer number in a binary mode:

```
...
Value ConvertBinShort::Scan(const Value& text) const
{
 Value v = UPP::ScanInt((const char*)String(text), NULL, 2);
 if(IsError(v)) return v;
 if(IsNull(v)) return notnull ? NotNullError() : v;
 int m = v;
>>>>>> if(m > 0x7FFF) m = 0xFFFF - m;
 if(m >= minval && m <= maxval) return v;
 return ErrorValue(UPP::Format(t_("Number must be between %X and %X."), minval, maxval));
}
...
```

For every type of number: char, short, int and int64 I have a different class. The same problem for Octal ed Hexadecimal converters.

In fact, I don't found any solution for get the type of the number from Value.

```
v.GetType() == INT_V
v.GetType() == INT64_V
```

or

```
v.Is<int>
v.Is<int64>
```

Not that I would recommend testing in most cases...

Mirek

Subject: Re: Converters and Value problem
Posted by mubeta on Fri, 23 Jul 2010 10:11:03 GMT
View Forum Message <> Reply to Message

Mirek,
yes, I know what you answer, but the problem is: Value don't allow char and short type, so, in fact, I am not able to recognize then from Value:

```
Value.GetType() == CHAR_V // does not exist
Value.GetType() == SHORT_V // does not exist
```

I work around PLC and small microcontrollers, where CHAR and SHORT are daily normal work.

If I develop an U++ application for comunicate with this kind of device, char and short are basically need.

Example:
char i;

i limits are: -128, +127 in decimal
i limits are: 0x80, 0x7f in exadecimal
etc...

If I use the converter, with minval and maxval, for show and edit one byte in exadecimal, (note that upp only allow integers), typing: FE result in a positive value, the editor complain about the max limit.

Second: Formatting -1 in exadecimal, for U++ result in "FFFFFFFFFFFF...", where, for char, must be only "FF".

So, again, for now I have, for every format, 4 different converter of type: char, short, int, and int64 (last one I really don't need), only for adjust the format result string, and for verify as well the typed value limits.

Hope this explain help you to undertand the my point of view.

Subject: Re: Converters and Value problem
Posted by mirek on Sun, 25 Jul 2010 07:18:54 GMT
View Forum Message <> Reply to Message

mubeta wrote on Fri, 23 July 2010 06:11
Example:
char i;

i limits are: -128, +127 in decimal
i limits are: 0x80, 0x7f in exadecimal
etc...

Both ranges are subset of that one of int.

What you need here is special converter (one that enforces only -128, +127 range), not another Value type.

If you do not like having 4 Convert classes, create one parametrized..

Mirek
