Subject: Value: BOOLEAN_V, USERVALUE_V [REQUEST]
Posted by fallingdutch on Thu, 14 Dec 2006 11:00:55 GMT
View Forum Message <> Reply to Message

HI,

i would like two new const int for Values and one new type:
the new const ints are
- USERVALUE_V which should be the first free to use Value-Type ID
- BOOLEAN_V the id of the new Type BOOLEAN

I need Boolean for XML-Rpc:

```
//Value.h
const int USERVALUE_V = 1024;
const int BOOLEAN_V = [booleanid];
inline dword ValueTypeNo(const bool) {return BOOLEAN_V;}

//Value.cpp
Value::Value(bool b)          { ptr = new RichValueRep<bool>(b); }
Value::operator bool() const
{
 if(IsNull()) return false;
 return (GetType()==BOOLEAN_V)?bool(RichValue<bool>::Extract(*this)): operator int();
}
```

i guess that should do it.

Bas

Subject: Re: Value: BOOLEAN_V, USERVALUE_V [REQUEST]
Posted by mirek on Thu, 14 Dec 2006 19:28:02 GMT
View Forum Message <> Reply to Message

I am not quite sure BOOL_V is a good idea, but be it... But IMHO be aware that such solution is extremely fragile because of conversions between numerical values (note that IsNumber is the recommended method of type inquiry there).

Maybe the right solution is to provide special type, RpcBool, or something like that.

USERVALUE_V does not have sense. I must admit that the idea of numeric ids is somewhat fragile as well, but USERVALUE_V would not solve that. At least Value system checks for duplicate definitions.

Mirek

Subject: Re: Value: BOOLEAN_V, USERVALUE_V [REQUEST]
Posted by fallingdutch on Fri, 15 Dec 2006 08:41:04 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 14 December 2006 20:28I am not quite sure BOOL_V is a good idea, but be it...
Why isn't it a good idea in your oppinon?
it is needed in database, too

luzr wrote on Thu, 14 December 2006 20:28
But IMHO be aware that such solution is extremely fragile because of conversions between numerical values (note that IsNumber is the recommended method of type inquiry there).
What about just adding a new ID BOOL_V and a function IsBool that returns true if the Value was constructed with a bool as param.
all other conversions will stay the same, so bool is returned as integer, means IsNumber tests on BOOL_V, too and BOOL_V is casted to int every time. so the whole is not broken, but one can distinguish, wether the value was created with a boolean or with any other number.
That would result in:
//Value.h
const int BOOLEAN_V = [booleanid];
inline dword ValueTypeNo(const bool) {return BOOLEAN_V;}
inline bool IsNumber(const Value& v) { return v.GetType() == DOUBLE_V || v.GetType() == INT_V || \
v.GetType() == BOOL_V || v.GetType() == INT64_V; }

//Value.cpp
Value::Value(bool b) { ptr = new RichValueRep<bool>(b);}
Value::operator int() const //same for int64, double
{
 if(IsNull()) return Null;
 return GetType() == INT_V   ? RichValue<int>::Extract(*this)
 : GetType() == INT64_V ? int(RichValue<int64>::Extract(*this))
 : GetType() == BOOL_V ? int(RichValue<bool>::Extract(*this))
 : int(RichValue<double>::Extract(*this));
}

luzr wrote on Thu, 14 December 2006 20:28
Maybe the right solution is to provide special type, RpcBool, or something like that.
That would mean i have to create a RpcValue:Value with just the changes mentioned above.
luzr wrote on Thu, 14 December 2006 20:28
USERVALUE_V does not have sense. I must admit that the idea of numeric ids is somewhat fragile as well, but USERVALUE_V would not solve that. At least Value system checks for duplicate definitions.
Why not? so anyone using u++ and wants to define his own id knows where to start: USERVALUE_V+k (k in {0,1,2, ...} is known to be unused.

Bas

Subject: Re: Value: BOOLEAN_V, USERVALUE_V [REQUEST]
Posted by mirek on Fri, 15 Dec 2006 09:40:34 GMT
View Forum Message <> Reply to Message

fallingdutch wrote on Fri, 15 December 2006 03:41
luzr wrote on Thu, 14 December 2006 20:28
But IMHO be aware that such solution is extremely fragile because of conversions between numerical values (note that IsNumber is the recommended method of type inquiry there).
What about just adding a new ID BOOL_V and a function IsBool that

Well, that is what I have done yesterday.... (no IsBool there, but you can use GetType).

But still seems fragile to me.

Quote:
Why not? so anyone using u++ and wants to define his own id knows where to start: USERVALUE_V+k (k in {0,1,2, ...} is known to be unused.

The real question is where ends U++ and starts "user". In any case, Value.h is not the only place where these numbers are defined. Plus what if user has more than single package (so that 0,1,2,3 will start to clash).

Mirek

Subject: Re: Value: BOOLEAN_V, USERVALUE_V [REQUEST]
Posted by fallingdutch on Fri, 15 Dec 2006 09:48:43 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 15 December 2006 10:40
Well, that is what I have done yesterday.... (no IsBool there, but you can use GetType).

thank you, Mirek

luzr wrote on Fri, 15 December 2006 10:40
The real question is where ends U++ and starts "user". In any case, Value.h is not the only place where these numbers are defined. Plus what if user has more than single package (so that 0,1,2,3 will start to clash).
i now see the problem you mean ...

Bas

Subject: Re: Value: BOOLEAN_V, USERVALUE_V [REQUEST]
Posted by mirek on Fri, 15 Dec 2006 10:54:52 GMT
View Forum Message <> Reply to Message

Quote:
luzr wrote on Fri, 15 December 2006 10:40
The real question is where ends U++ and starts "user". In any case, Value.h is not the only place where these numbers are defined. Plus what if user has more than single package (so that 0,1,2,3 will start to clash).
i now see the problem you mean ...

Bas

Well, note that all the trouble is result of "numeric" ids. Maybe we should rather use texts (and numerics just for a couple of "internal" ids).

Mirek