Subject: Core: Null handling incoherent?
Posted by kohait00 on Wed, 04 May 2011 08:46:33 GMT
View Forum Message <> Reply to Message

hi all,

i'm wondering why

template <class T> bool IsNull(const T& x) { return x.IsNullInstance(); }

and not

template <class T> bool IsNull(const T& x) { return x.IsNull(); }

is there a reason for it? name clashes??
analogue to upp philosophy it should be latter case, (Xmlize() { x.Xmlize() } etc.)

some classes define IsNullInstance(), some do IsNull(), it's kinda 'not clean'

(background: i'm tackling Null handling in terms of extension of Value with other types on user side (i.e. float), where the Null handlig is the major problem.)

EDIT:
it's also the point of what Null actually is.. is it only to be seen in context with Value handling? because all classes interacting with Value the classes need to know about Value, but Value.h defines some of the interface handling with the types as well. so it's mixed. (i.e. String.h has template definition of IsNull())..

maybe to think of null handling as kind a independant from Value and define it in Defs.h

template <class T> void SetNull(T& x) { x.SetNull(); }
template <class T> bool IsNull(const T& x) { return x.IsNullInstance(); }

so both, Value.h and all the others are aware of that concept..
this probably would also make sense to move Nuller concept to Defs.h

i'm currently making some changes, just trying to get trhough it properly..