Subject: Re: The problem with 'Null'
Posted by mirek on Thu, 19 Mar 2009 08:16:34 GMT
View Forum Message <> Reply to Message

gridem wrote on Thu, 19 March 2009 03:04I found that Upp uses the following practice: instead of creating already prepared object it creates the object with default constructor and than fill the necessary members in later calls. But using such approach the programmer should distinguish between init and non init state. One of the possible solution: apply 'Null' to the fields and than check by using IsNull.

The simple types (int, long etc) and Value already have such posibility. String also can use this but using another functionality:
String::GetVoid() return 'super' empty string that may be treated as 'Null'. The problem is that IsNull(String()) and IsNull(String::GetVoid()) return both true. This may be workarounded but it's not a good solution. But for the Vector<T> the workaround is more complex: the programmer should use One<Vector<T> >. The problem may occur in situation when function should return result or error. In the following example:
String LoadFile(...)
the solution exist: return String::GetVoid() on error. But what I can do when I must return Vector:
Vector<Templates> GetTemplateList()
Empty list denotes the there are no templates. But how can I return error without using terrible One<Vector<T> >?

Null is for "full values".

Note that String::GetVoid is deliberately defined "Null" because e.g. for LoadFile, most code can safely ignore the error as long as they get "Null" file.

As for your vector example, I would simply use

bool GetTemplateList(Vector<Templates>& r);

Mirek