Subject: Re: moveable question
Posted by dolik.rce on Wed, 28 Nov 2012 07:04:09 GMT
View Forum Message <> Reply to Message

Hi,

First let me comment on you constructor, see the comments:SimpleVector::SimpleVector(){
 int n = 2;
 int * y = &n; // this is broken, you're taking addresss of temporary object,
                // the pointer will point to random data in memory when you leave the function
 v.Add(y);
 Cout()<<*v[0]; // you can use U++ Cout() instead of std::cout ;)
}

Now, the assertion, AssertMoveable merely checks if the object is moveable. It is users responsibility to assure that anything that inherits from Moveable<T> (or is marked with NTL_MOVEABLE() macro) is actually moveable. Moveable is just a hint from a programmer to the compiler, it doesn't really make it moveable, or prevent you from doing moveability-breaking operations.

To illustrate how the pointers in TestMove() break things, you have to think little further, lets see what happens when you try to copy the SimpleVector:CONSOLE_APP_MAIN
{
 SimpleVector s;
 s.TestMove();

 // To actually break it, try to perform a copy operation:
 SimpleVector v;
 v=s;

 // Now it is broken, because the pointers were just copied
 // The v.val points s.a, which is in most cases not what you want
 // Also v.sv == s.sv, same problem
 // Imagine what happens when s is destructed earlier than v
 //   -> you have pointers to non-existent objects, just asking for a crash ;)
}

Does that shed some light on the subject?

Best regards,
Honza