Subject: Re: Make THISFN simpler and more powerful
Posted by Lance on Wed, 09 Nov 2022 23:32:44 GMT
View Forum Message <> Reply to Message

This is kind of C++20 related, I am going to reuse this thread for c++ language specific problems or what not.

Here I got a problem: how to detect if a class is Moveable?

A naive first thought would be

```
#include <Core/Core.h>
#include <concepts>

template <class T>
inline constexpr bool is_moveable = std::derived_from<T, Upp::Moveable<T>>;

using namespace Upp;

CONSOLE_APP_MAIN
{
 LOG(is_moveable<Upp::Rect>);
 LOG(is_moveable<Upp::String>);
}
```

Output

```
true
false
```

It turns out Moveable has an optional second template parameter which failed the test in [b]String[/]'s case.

What makes Moveable so special is that it introduced a friend AssertMoveable0 in <Core/Topt.h>

```
template <class T>
inline void AssertMoveable0(T *t) { AssertMoveablePtr(&**t, *t); }
// COMPILATION ERROR HERE MEANS TYPE T WAS NOT MARKED AS Moveable

template <class T, class B = EmptyClass>
struct Moveable : public B {
 friend void AssertMoveable0(T *) {}
};
```

Trying to use it directly, I come up with something like

```
#include <Core/Core.h>
#include <concepts>

template <class T>
inline constexpr bool is_moveable = /*std::derived_from<T, Upp::Moveable<T>>;*/
 requires (T t){ Upp::AssertMoveable0(&t); };

using namespace Upp;

CONSOLE_APP_MAIN
{
 LOG(is_moveable<Upp::Rect>);
 LOG(is_moveable<Upp::String>);
}
```

This time both give correct result, but unfortuately, any class T would give a true result regardless of whether it's derived from Upp::Moveable.

How would you check if a class type is Upp::Moveable ?

BTW, something like this

```
LOG( std::is_base_of<Ctrl, EditField>);
```

will fail to compile because of a missing guard in the definition of LOG in <Core/Diag.h>

```
#define LOG(a) UPP::VppLog() << a << UPP::EOL
```

I would think here change it to

```
#define LOG(a) UPP::VppLog() << (a) << UPP::EOL
```

might be a reasonable thing to do.

---