How well does ATL CComPtr support class template argument deduction (CTAD)?

devblogs.microsoft.com/oldnewthing/20240314-00

March 14, 2024

Raymond Chen

Continuing our investigation of which C++ COM wrappers support class template argument deduction (CTAD), next up is ATL's `CComPtr`.

ATL's `CComPtr` works with CTAD right out of the box because the constructor that takes a pointer provides all the information needed to specialize the template class.

```
template <class T>
class CComPtr : public CComPtrBase<T>
{
public:
    CComPtr(_Inout_opt_ T* lp); // this constructor

    ⟦ other stuff ⟧
};

void sample(Test* p)
{
    // Works right out of the box!
    auto smart = CComPtr(p);
}
```

The compiler sees that the constructor parameter is a `T*`, so it can figure out what `T` needs to be.

Interestingly, but perhaps not surprisingly, it is the smart pointer class with the most straightforward constructor that works best with CTAD. CTAD tries to enable magic template arguments, and if you make your constructor too fancy, CTAD won't be able to figure out what it needs to do.

Next up is WRL, and things get more complicated.