Injected class names: The C++ feature you didn't even realize that you were using

devblogs.microsoft.com/oldnewthing/20220321-00

March 21, 2022

Raymond Chen

C++ has a feature called injected class names, which lets you refer to the class being defined by its plain name without needing to fully qualify it with namespaces and template parameters.

You have probably been taking advantage of this feature without even realizing it:

```
template<typename T>
struct Wrapper
{
    Wrapper() { /* constructor */ }
};
```

That `Wrapper()` is using the injected class name. The full name of the constructor is

```
    Wrapper<T>() { /* constructor */ }
```

but C++ secretly injects the simple name of the class into the class itself, as if you had written

```
    using Wrapper = Wrapper<T>;
```

Injected class names are public and therefore can be inherited:

```
namespace details {
    struct Base
    {
        Base(int) { /* constructor */ }
    };
}

struct Derived : details::Base
{
    Derived() : Base(42) { /* constructor */ }
};
```

The construction of the base class from the derived class can be done by writing just `Base(42)` instead of the full name `details::Base(42)`. The injection goes like this:

```
namespace details {
    struct Base
    {
        using Base = ::details::Base; // injected
        Base(int) { /* constructor */ }
    };
}

struct Derived : details::Base
{
    Derived() : Base(42) { /* constructor */ }
}
```

When you write `Base` inside the definition of `Derived`, you are using the name `Base` that was defined way over there in the `Base` class.

Note that making base classes private prevents their types from being inherited:

```
namespace details {
    struct Base { };
    struct Middle : private Base { };
}

struct Derived : details::Middle
{
    void blah()
    {
        Middle m; // "Middle" accessible from details::Middle
        Base b; // "Base" not accessible because it was privately inherited
        details::Base b; // Need to use the full name
    }
};
```

Raymond Chen

Follow