Subject: Re: Nested template question
Posted by Novo on Mon, 10 Jun 2019 21:48:30 GMT
View Forum Message <> Reply to Message

I don't really get what you need to do.
If it is just setting any class to zero, as in your example, then you do not need method GetAZero(). Class AZero will work out of the box.
If you are trying to specialize your template method for all possible variants of std::complex<T>, then this is called partial template specialization, and it works only for classes. You need to create a dummy class and partially specialize it.

```
template <typename T>
struct dummy {
 T DoIt() const { return T(); }
};

template <typename T>
struct dummy<std::complex<T>> {
 using PT = std::complex<T>;
 PT DoIt() const { return PT(0, 0); }
};

struct Boo {
 Boo() {
  double val    = GetAZero<double>();
  std::complex<float>  valc1 = GetAZero<std::complex<float>>();
  std::complex<double>  valc2 = GetAZero<std::complex<double>>();
 }

 template <class T>
 T GetAZero() { return dummy<T>().DoIt(); }
};
```