Subject: MakeOne
Posted by mirek on Tue, 21 Feb 2017 10:39:32 GMT
View Forum Message <> Reply to Message

Little useful helper class...

```
void DumpFile(One<Stream>& s)
{
 LOG("=========");
 int ii = 0;
 while(!s->IsEof())
  LOG(++ii << ": " << s->GetLine());
}

CONSOLE_APP_MAIN
{
 String fn = GetDataFile("One.cpp");
 {
  One<Stream> s;
  s.Create<FileIn>(fn);
  DumpFile(s);
 }
 {
  MakeOne<FileIn> in(fn);
  One<Stream> s = pick(in);
  DumpFile(s);
 }
 {
  One<Stream> s = MakeOne<FileIn>(fn);
  DumpFile(s);
 }
}
```

Subject: Re: MakeOne
Posted by Klugier on Tue, 21 Feb 2017 11:14:25 GMT
View Forum Message <> Reply to Message

Hello,

I have got one question. Will it work with auto?

```
auto stream = MakeOne<FileIn>(fn); // Is auot MakeOne or One?
```

Why not make MakeOne function rather than class?

Sincerely,
Klugier

Subject: Re: MakeOne
Posted by mirek on Tue, 21 Feb 2017 15:05:30 GMT
View Forum Message <> Reply to Message

Klugier wrote on Tue, 21 February 2017 12:14Hello,

I have got one question. Will it work with auto?

auto stream = MakeOne<FileIn>(fn); // Is auot MakeOne or One?

Sure. It will be MakeOne, but that hardly matters.

```
{
 auto in = MakeOne<FileIn>(GetDataFile("Console.cpp"));
 One<Stream> s = pick(in);
 while(!s->IsEof())
  DDUMP(s->GetLine());
}
```

Quote:
Why not make MakeOne function rather than class?

Class seems to be more versatile here - you can use it just like function, but you can declare the variable too.

E.g. you can do this:

```
One<Stream> OpenFile()
{
   MakeOne<FileIn> in;
   in->Open("asd");
   return in;
}
```

With "function only" you would have to type a bit more...

Mirek

Subject: Re: MakeOne
Posted by Didier on Wed, 22 Feb 2017 18:24:39 GMT
View Forum Message <> Reply to Message

It may be obvious, but I don't see the point to the 'MakeOne' helper ==> you only gain one line of writing.
I suppose there is more to it ??

Subject: Re: MakeOne
Posted by mirek on Wed, 22 Feb 2017 19:43:19 GMT
View Forum Message <> Reply to Message

Didier wrote on Wed, 22 February 2017 19:24It may be obvious, but I don't see the point to the 'MakeOne' helper ==> you only gain one line of writing.
I suppose there is more to it ??

Yes, you are right. Still, it was requested by users. And personally I longed for it in certain situations too (perhaps being lazy to write that one line... :)

Mirek

Subject: Re: MakeOne
Posted by Klugier on Sat, 30 Jun 2018 12:45:41 GMT
View Forum Message <> Reply to Message

Hello,

I have a problem with current MakeOne implementation. It doesn't work greate with auto:

```
One<Foo> CreateFoo() {
 auto pFoo = MakeOne<Foo>();

 int i = 0;
 if (i == 0) {
  return nullptr;
 }

 return pFoo; // Compilation error - /home/klugier/MyApps/MakeOneTest/MakeOneTest.cpp (19): error: use of deleted function 'Upp::One<T>::One(const Upp::One<T>&) [with T = Foo]'
```

```
}
```

The below example works as expected:

```
One<Foo> CreateFoo() {
 One<Foo> pFoo = MakeOne<Foo>();

 int i = 0;
 if (i == 0) {
  return nullptr;
 }

 return pFoo;
}
```

I am not sure it is the good design to limit it to only explicit types. std::make_shared and std::make_unique works in above case.

Sincerely,
Klugier

Subject: Re: MakeOne
Posted by mirek on Sat, 30 Jun 2018 18:30:45 GMT
View Forum Message <> Reply to Message

Hopefully fixed.

Subject: Re: MakeOne
Posted by Klugier on Sat, 30 Jun 2018 19:55:28 GMT
View Forum Message <> Reply to Message

Hello,

You are absolutely amazing - thanks for immediate fix :)

Sincerely,
Klugier
