Subject: partial parametrizing of Callback<>
Posted by kohait00 on Thu, 15 Jul 2010 19:33:41 GMT
View Forum Message <> Reply to Message

hi folks,

imagine i want to do the following

```
void MyCallback::Action(int i, float f)
{
 RLOG("" << i << " " << f);
}

void MyCallback::Test()
{
 Callback1<int> cb;
 cb = callback2(this, &CLASSNAME::Action, 1.0f); //latter param already specified
 cb(123); //first parameter specified now
}
```

partially specify the arguments to the callback, the rest specify later.

would it be possible with callbacks (i imagine one needs to extend the template generators for that)?

Subject: Re: partial parametrizing of Callback<>
Posted by dolik.rce on Thu, 15 Jul 2010 20:36:34 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 15 July 2010 21:33hi folks,

imagine i want to do the following

```
void MyCallback::Action(int i, float f)
{
 RLOG("" << i << " " << f);
}

void MyCallback::Test()
{
 Callback1<int> cb;
 cb = callback2(this, &CLASSNAME::Action, 1.0f); //latter param already specified
 cb(123); //first parameter specified now
```

}

partially specify the arguments to the callback, the rest specify later.

would it be possible with callbacks (i imagine one needs to extend the template generators for that)?

Hi kohait,

If I understand right what you want to achieve, than it is possible and actually quite easy :

```
void MyCallback::Action(int i, float f)
{
 RLOG("" << i << " " << f);
}

void MyCallback::Test()
{
 Callback1<int> cb;
 cb = callback1(this, &CLASSNAME::Action, 1.0f);
 cb(123);
}
```

As you can see, you were very close, just one letter is different

Hope I got you question right, it looks suspiciously simple.

Best regards,
Honza

Subject: Re: partial parametrizing of Callback<>
Posted by kohait00 on Fri, 16 Jul 2010 05:06:33 GMT
View Forum Message <> Reply to Message

yeaaaaaa, and it works..thanks a lot.

i didnt know something like that was possible, found callback was strict enough concerning type checks.. well perfect. has this support for such semmi parametrized Callback arrived recently or already long time ago.

but what does not work is with 3 params..

```
void MyCallback::Action(int i, float f)
{
 RLOG("" << i << " " << f);
```

```
}

void MyCallback::Action2(int i, float f, int k)
{
 RLOG("" << i << " " << f);
}

void MyCallback::Test()
{
 //works
 Callback1<int> cb;
 cb = callback1(this, &CLASSNAME::Action, 12.0f);
 cb(123);

 //DONT works
 Callback2<int, float> cb2;
 cb2 = callback2(this, &CLASSNAME::Action2, 1234);
 cb2(234, 10.f);

 //DONT works
 Callback1<int> cb3;
 cb3 = callback1(this, &CLASSNAME::Action2, 1234, 13.0f);
 cb3(234);
}
```

Subject: Re: partial parametrizing of Callback<>
Posted by dolik.rce on Fri, 16 Jul 2010 13:16:47 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 16 July 2010 07:06i didnt know something like that was possible, found callback was strict enough concerning type checks.. well perfect. has this support for such semmi parametrized Callback arrived recently or already long time ago.
It is there as long as I remember

The case with three parameters is not used that often, therefore there is no template for it. Also with increasing number of parameters, the complexity rises. For 3 parameter function, we should supply 2 functions (returning 1 and 2 parameter callback), for 4 parameter function 3 generating functions etc.

But you can "easily" (not to be taken too seriously) extend the templates yourself. I know you already looked into the internals of Callback, so it should not be that hard for you. Sorry for not supplying the actual code, but my head goes dizzy whenever I start thinking about templates with so many parameters - BTW: that is possibly one of the reasons why such complicated callbacks are not present in U++ by default

Regards,
Honza

Subject: Re: partial parametrizing of Callback<>
Posted by kohait00 on Fri, 16 Jul 2010 17:17:46 GMT
View Forum Message <> Reply to Message

thanks for the infos,

its no problem not having them, i was just wondering one can accomodate the code to fit it as well.but good to know