Subject: RPC_METHOD how to "define"
Posted by NilaT on Tue, 31 Jan 2017 15:56:40 GMT
View Forum Message <> Reply to Message

Hello everybody,

I'm currently experimenting with RPC_METHOD and MtRpcClient/MtRpcServer.

I want to do a class and write some RPC_METHOD's my own, but as class Members.

How do i extract that weird define?
#define RPC_METHOD(name) void name(RpcData& rpc); INITBLOCK { Register(#name, name); } void name(RpcData& rpc)

So in particular, how can I use the example method getClientList as a member function?
And whats INITBLOCK and #name?

Thanks in advance.
Greets

Subject: Re: RPC_METHOD how to "define"
Posted by dolik.rce on Tue, 31 Jan 2017 21:10:11 GMT
View Forum Message <> Reply to Message

Hi NilaT,

NilaT wrote on Tue, 31 January 2017 16:56How do i extract that weird define?
#define RPC_METHOD(name) void name(RpcData& rpc); INITBLOCK { Register(#name, name); } void name(RpcData& rpc)

So in particular, how can I use the example method getClientList as a member function?
And whats INITBLOCK and #name?

First a little hint: In TheIDE open a file where RPC_METHOD is used and hit Build -> Preprocess (or Alt+F7). It will show you the file with all includes included and macros expanded.

Now, to decipher the macro:

The INITBLOCK is another U++ macro, which allows you to add a code block that will be executed when the program starts, before main() is executed. It is often used to register methods or plugins.

#name creates a string literal of the value passed to the name parameter. So the "RPC_METHOD(MyMethod){ some_code; }" expands to this:void MyMethod(RpcData& rpc);
```
INITBLOCK {
    Register("MyMethod", MyMethod);
}
```

```
void name(RpcData& rpc) {
    some_code;
}
```
Note that I added some newlines for better readability. So what the macro does, step by step, is:
1) The function is declared.
2) It is registered, so that the RPC server knows that the method exists.
3) The function is defined. Here the code block that follows macro handily becomes the function body.

Does this explanation make it clearer?

To create the methods as class members, you'll have to do the same steps as the macro does, but with a member functions. That's is declare and/or define the member function and register it before it can be used. It would be probably ok to call Register in the class constructor. So it could look somewhat like this:
```
class MyRpcClass {
public:
    typedef MyRpcClass CLASSNAME;
    void MyMethod(RpcData& data);
    MyRpcClass() {
        Register("MyMethod", THISBACK(MyMethod));
    }
}
```
Warning: I haven't tested the code (and I'm not even familiar with how the RpcServer works internally), so it will probably need a bit more work :) But in general, it should be possible to do this.

Best regards,
Honza

Subject: Re: RPC_METHOD how to "define"
Posted by NilaT on Wed, 01 Feb 2017 09:58:36 GMT
View Forum Message <> Reply to Message

Hi Honza, thanks for your reply.

Yes, your explanation makes it a bit clearer for me, thanks :)
Maybe you could add all those little, Upp specific things in the documentations some day? ;)

As for the member function... I've tried your code, but it won't work.
Tried some other things too, but still won't compile.
Maybe you know a solution?

Here's some code I tried and the resulting errors:
```
 Register("GetClients", THISBACK(GetClients));
```
test.cpp (70): error C2664: "void Upp::Register(const char *,void (__cdecl *)(Upp::RpcData &),const char *)" : Konvertierung von Argument 2 von "Upp::Event<Upp::RpcData &>" in "void (__cdecl *)(Upp::RpcData &)" nicht möglich

```
 Register("GetClients", GetClients);
test.cpp (70): error C3867: "Test::GetClients": Keine Standardsyntax; "&" zum Erstellen eines
Verweises auf das Member verwenden

Register("GetClients", &GetClients);
test.cpp (70): error C2276: "&": Ungültige Operation auf Ausdruck einer gebundenen
Memberfunktion
```

Any solutions?
Thanks :)

Subject: Re: RPC_METHOD how to "define"
Posted by dolik.rce on Thu, 02 Feb 2017 16:14:15 GMT
View Forum Message <> Reply to Message

Well, the problem is that THISBACK can't be converted to function pointer as expected by Register. This can be worked around, but I had to made an one assumption: Only one instance of the class exists at a time. This is quite reasonable in RPC server, since otherwise you wouldn't know which instance to use when client calls one of the methods.

Here is a working (tested :)) code:
```
#include <Core/Core.h>
#include <Core/Rpc/Rpc.h>

using namespace Upp;

class MyRpcClass {
    Time last;
public:
    typedef MyRpcClass CLASSNAME;
    void Status(RpcData& rpc) {
        rpc << last;
    }
    void Ping(RpcData& rpc) {
        last = GetSysTime();
        rpc << last;
    }
    static void RegisterMethods() {
        // we use lambda to get a singleton instance of the class and call its method
        Register("Status", [](RpcData& rpc){Single<MyRpcClass>().Status(rpc);});
        Register("Ping", [](RpcData& rpc){Single<MyRpcClass>().Ping(rpc);});
    }
};

INITBLOCK {
 // calls all the registrations before main executes
 MyRpcClass::RegisterMethods();
}

CONSOLE_APP_MAIN
{
 Cout() << "Server...\n";
 LogRpcRequests();
 RpcServerLoop(1234);
}
```

It could be made slightly prettier with macros, but I didn't want to obfuscate the logic too much...

Honza

Subject: Re: RPC_METHOD how to "define"
Posted by mirek on Fri, 03 Feb 2017 17:00:34 GMT
View Forum Message <> Reply to Message

NilaT wrote on Tue, 31 January 2017 16:56
So in particular, how can I use the example method getClientList as a member function?

Well, in addition to what was said, I guess trouble is that "member function" does not make much sense here. RPC on this basic level has no notion of object instances, the whole server is treated as, more or less, single object.

You can add object layer over this, but usually it makes little sense.

Mirek

Subject: Re: RPC_METHOD how to "define"
Posted by NilaT on Sun, 05 Feb 2017 10:02:26 GMT
View Forum Message <> Reply to Message

Thanks guys for your help.
Mirek, my intention why I want it as member functions is, because my RPC Methods all need a database connection, so in one class function I open the connection and keep it open. And because I don't want to pass this PostgreSqlSession to all RPC Functions, I want to use a member.
Furthermore I need an Instance of a very big class and don't want to pass this either, so a member seems to be the best solution in my opinion.

PS: Maybe you can explain this code? I mean... it compiles, but I really don't know what this does:
[](RpcData& rpc){Single<MyRpcClass>().Status(rpc);});

Subject: Re: RPC_METHOD how to "define"
Posted by mirek on Sun, 05 Feb 2017 13:03:14 GMT
View Forum Message <> Reply to Message

NilaT wrote on Sun, 05 February 2017 11:02Thanks guys for your help.
Mirek, my intention why I want it as member functions is, because my RPC Methods all need a database connection, so in one class function I open the connection and keep it open. And because I don't want to pass this PostgreSqlSession to all RPC Functions, I want to use a member.
Furthermore I need an Instance of a very big class and don't want to pass this either, so a member seems to be the best solution in my opinion.

The basically you need single global connection, that is all. Situation would not change with introduction of class.

There are some further consideration (like will be your application multithreaded?), but you will either end with one connection per server thread or one connection per server call...

Quote:
PS: Maybe you can explain this code? I mean... it compiles, but I really don't know what this does:
[](RpcData& rpc){Single<MyRpcClass>().Status(rpc);});

Single<MyRpcClass> actually pretty much does the same thing - there will be single global instance of MyRpcClass.

Mirek

Subject: Re: RPC_METHOD how to "define"
Posted by dolik.rce on Sun, 05 Feb 2017 19:48:32 GMT
View Forum Message <> Reply to Message

NilaT wrote on Sun, 05 February 2017 11:02Mirek, my intention why I want it as member functions is, because my RPC Methods all need a database connection, so in one class function I open the connection and keep it open. And because I don't want to pass this PostgreSqlSession to all RPC Functions, I want to use a member. As Mirek already said, global instance is just fine for this purpose. If you don't like global variables, than you can use the template Single<T>, just as I did in the example, to create single instance of some type T, which is shared by all the code that calls Single<T>(). Furthermore, for Sql connection, there are already application-wide variables SQL and SQLR for this (the second one can be used for readonly connections). These variables even allow to write some sql commands in simpler way.

NilaT wrote on Sun, 05 February 2017 11:02Furthermore I need an Instance of a very big class and don't want to pass this either, so a member seems to be the best solution in my opinion.No U++ solution for this, but again, global variable or any kind of singleton should do.

NilaT wrote on Sun, 05 February 2017 11:02PS: Maybe you can explain this code? I mean... it

compiles, but I really don't know what this does:
[](RpcData& rpc){Single<MyRpcClass>().Status(rpc);});Single<> creates singleton instance, while "[](RpcData& rpc) {}" is a definition of lambda function from C++11. It is basically an anonymous function that takes RpcData& as parameter and calls your member function on the singleton as mentioned above.

Honza

Subject: Re: RPC_METHOD how to "define"
Posted by NilaT on Sun, 05 Feb 2017 21:00:28 GMT
View Forum Message <> Reply to Message

Wow, thanks for your input.
Still very new terrain for me.
I understand the part with global variables...
Not sure about that Single-part though. Is this some kind of static?
And Lambda is completly new to me... Didn't understand a thing here but as long as it work's im kinda happy ;)

And mirek yes, it's going to be MT. Like the MtRpcServer.

Subject: Re: RPC_METHOD how to "define"
Posted by mirek on Mon, 06 Feb 2017 15:17:44 GMT
View Forum Message <> Reply to Message

NilaT wrote on Sun, 05 February 2017 22:00Wow, thanks for your input.
Still very new terrain for me.
I understand the part with global variables...
Not sure about that Single-part though. Is this some kind of static?

Single represents singleton pattern, single global instance of calss.

Quote:
And Lambda is completly new to me... Didn't understand a thing here but as long as it work's im kinda happy ;)

And mirek yes, it's going to be MT. Like the MtRpcServer.

Well, then you have sort of different issue to resolve, Single is not good enough there... You will rather need one DB connection per thread.

Subject: Re: RPC_METHOD how to "define"
Posted by NilaT on Mon, 06 Feb 2017 15:33:08 GMT
View Forum Message <> Reply to Message

I'll give it a shot and come back to you if any errors occur ;)

In the meanwhile, thanks for your help