Subject: make U++ more elegant
Posted by Ulti on Sun, 22 Apr 2007 00:28:39 GMT
View Forum Message <> Reply to Message

four things will make U++ more elegant
1.use XML to store resource
2.signal slot mode event system
3.boost based core(this will make people use U++ with other lib easier)
4.focused on GUI

and maybe another good point:
RMI for C++
http://rmi.sourceforge.net/pmwiki/pmwiki.php

Subject: Re: make U++ more elegant
Posted by mirek on Sun, 22 Apr 2007 06:27:05 GMT
View Forum Message <> Reply to Message

Ulti wrote on Sat, 21 April 2007 20:28
RMI for C++
http://rmi.sourceforge.net/pmwiki/pmwiki.php

Funny, this looks like a good example how not to present your project

I do not know whether this is good or bad, but after 2 minutues trying to find any info about C++RMI, I gave up...

Mirek

Subject: Re: make U++ more elegant
Posted by zsolt on Sun, 22 Apr 2007 06:47:59 GMT
View Forum Message <> Reply to Message

Ulti wrote on Sun, 22 April 2007 02:28
1.use XML to store resource

You can use it. XML classes are very simple and useful. For theIde project files, plain text is much better, because it can be read, edit, merge very easliy.
Quote:
2.signal slot mode event system

Current callback system does the same with different terminology and simpler usage (Callback, THISBACK).
Quote:
3.boost based core(this will make people use U++ with other lib easier)

I don't think so. Boost is a very slowly growing, old fashioned thing.
Quote:
4.focused on GUI

I think Upp is focused on GUI.

Quote:
and maybe another good point:
RMI for C++
http://rmi.sourceforge.net/pmwiki/pmwiki.php
The pages are password protected.
BTW I was thinking about something similar in UPP, because current callback system could be easily extended to work transparently on network.

Subject: Re: make U++ more elegant
Posted by Ulti on Sun, 22 Apr 2007 07:02:49 GMT
View Forum Message <> Reply to Message

http://www.ddj.com/dept/cpp/184403949#2

and

http://java.sun.com/developer/onlineTraining/rmi/RMI.html

Subject: Re: make U++ more elegant
Posted by Ulti on Sun, 22 Apr 2007 07:06:02 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 22 April 2007 02:47
I don't think so. Boost is a very slowly growing, old fashioned thing.

but a lot of things were written using boost and stl.if you want use
these things,always need translation between two.

Subject: Re: make U++ more elegant
Posted by Ulti on Sun, 22 Apr 2007 07:13:54 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 22 April 2007 02:27
Funny, this looks like a good example how not to present your project

I do not know whether this is good or bad, but after 2 minutues trying to find any info about C++RMI, I gave up...

Mirek
sorry,it's not my project,it was googled.the idea sounds good.

Subject: Re: make U++ more elegant
Posted by mirek on Sun, 22 Apr 2007 07:19:26 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 22 April 2007 02:47
Quote:
and maybe another good point:
RMI for C++
http://rmi.sourceforge.net/pmwiki/pmwiki.php
The pages are password protected.
BTW I was thinking about something similar in UPP, because current callback system could be easily extended to work transparently on network.

So do I But my thinking is rather based on Serialize...

The real question(s) is:

- can we do RMI without IDL?
- is it worth the trouble?
- does it need to look like method invokation or we can can sustain a little bit less transparent solution (like passing structures with Serialize member)?

Mirek

P.S.: As this is an interesting topic, I am moving a copy to technology lab, if you have any ideas, let us continue there...

Subject: Re: make U++ more elegant
Posted by mirek on Sun, 22 Apr 2007 07:20:53 GMT
View Forum Message <> Reply to Message

Ulti wrote on Sun, 22 April 2007 03:06zsolt wrote on Sun, 22 April 2007 02:47
I don't think so. Boost is a very slowly growing, old fashioned thing.

but a lot of things were written using boost and stl.if you want use
these things,always need translation between two.

Yes, that is unfortunately true. There is always some kind of tradeoff, you can use STL or you can have our applications run 4 times faster with U++ Core... (latest benchmark with new Core).

Mirek

Subject: Re: make U++ more elegant
Posted by Ulti on Sun, 22 Apr 2007 08:47:37 GMT
View Forum Message <> Reply to Message

Two more links:
http://www.codeproject.com/threads/Rcf_Ipc_For_Cpp.asp
http://www.codeproject.com/threads/Rcf_Ipc_For_Cpp.asp

using boost::serlization
and some Macros

replace them with U++::serlization and template?

Subject: Re: make U++ more elegant
Posted by ebojd on Mon, 23 Apr 2007 15:50:45 GMT
View Forum Message <> Reply to Message

One of the things I like about Boost is that the official list has been vetted, and many of the things are being included into the ISO runtime library definitions (at least if I recall correctly). That formal review is one of the of the things slowing updates down... As for speed, execution time can often be cleaned up by simply tracking what all is being called by value instead of reference (and has to be created/instantiated on the function call)...

I am involved with a project that is looking to submit to Boost. If they do not want it I might post it here if people find it useful

EBo --

Subject: Re: make U++ more elegant
Posted by Novo on Wed, 25 Apr 2007 14:39:06 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 22 April 2007 03:20Ulti wrote on Sun, 22 April 2007 03:06zsolt wrote on Sun,

22 April 2007 02:47
I don't think so. Boost is a very slowly growing, old fashioned thing.

but a lot of things were written using boost and stl.if you want use
these things,always need translation between two.

Yes, that is unfortunately true. There is always some kind of tradeoff, you can use STL or you can have our applications run 4 times faster with U++ Core... (latest benchmark with new Core).

Mirek

Intrusive containers have been accepted to Boost lately, if I recall correctly. So, "4 times faster" won't last too long

---

Subject: Re: make U++ more elegant
Posted by mirek on Wed, 25 Apr 2007 19:20:59 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 25 April 2007 10:39
Intrusive containers have been accepted to Boost lately, if I recall correctly. So, "4 times faster" won't last too long

What makes you think intrusive containers are going to change anything?

Mirek

Subject: Re: make U++ more elegant
Posted by Novo on Wed, 25 Apr 2007 19:58:20 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 25 April 2007 15:20Novo wrote on Wed, 25 April 2007 10:39
Intrusive containers have been accepted to Boost lately, if I recall correctly. So, "4 times faster" won't last too long

What makes you think intrusive containers are going to change anything?

Mirek

UPP Core is based on NTL, which provides more efficient containers relatively to STL. "4 times faster" is related to containers, if I understood your idea.

New intrusive containers in Boost are faster than containers in STL.

Am I wrong?

Subject: Re: make U++ more elegant
Posted by mirek on Wed, 25 Apr 2007 20:07:37 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 25 April 2007 15:58
UPP Core is based on NTL, which provides more efficient containers relatively to STL. "4 times faster" is related to containers, if I understood your idea.

New intrusive containers in Boost are faster than containers in STL.

Am I wrong?

AFAIK, intrusive containers are faster / more effective in certain specific scenarios than STL. But that does not make it match U++ Core. In fact, from what I have read, intrusive containers mostly deal with node based elements, however the sole idea of node based containers is faulty. Continuous storage wins.

BTW, U++ Core and NTL relation: In fact, U++ Core was first. NTL was just failed attempt to take a part of U++ Core and make it a library of it own. It was our first attempt to make some code public...

Mirek

Subject: Re: make U++ more elegant
Posted by Novo on Wed, 25 Apr 2007 20:53:08 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 25 April 2007 16:07
AFAIK, intrusive containers are faster / more effective in certain specific scenarios than STL. But that does not make it match U++ Core. In fact, from what I have read, intrusive containers mostly deal with node based elements, however the sole idea of node based containers is faulty. Continuous storage wins.

OK. You convinced me Actually, I've already read a book, which explains a similar technique. Unfortunately, I haven't had a chance to apply it yet.

Quote:
BTW, U++ Core and NTL relation: In fact, U++ Core was first. NTL was just failed attempt to take a part of U++ Core and make it a library of it own. It was our first attempt to make some code public...

Mirek

As far as NTL is that much better, it worth submitting to Boost. In this case you'll get unlimited advertising for free

I think Boost worth using it. For example you could use Boost.Spirit to parse upp config files instead of manual processing them. Formal parsers let you discover interesting things like one below.

library
,
,
,
,
;

I found it in GridCtrl.upp

Subject: Re: make U++ more elegant
Posted by mirek on Wed, 25 Apr 2007 21:16:53 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 25 April 2007 16:53
As far as NTL is that much better, it worth submitting to Boost. In this case you'll get unlimited advertising for free

Not as simple. boost:: is extension of std::. U++ Core is negation of std::

Quote:
I think Boost worth using it. For example you could use Boost.Spirit to parse upp config files instead of manual processing them.

Yes, that is interesting concept. But I yet have to convinced that it will let me do more with less

code.

Quote:
Formal parsers let you discover interesting things like one below.

library
,
,
,
,
;

I found it in GridCtrl.upp

Yes, if I remember well, this is what .upp parser is supposed to accept

---

Subject: Re: make U++ more elegant
Posted by Novo on Wed, 25 Apr 2007 22:14:44 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 25 April 2007 17:16Novo wrote on Wed, 25 April 2007 16:53
As far as NTL is that much better, it worth submitting to Boost. In this case you'll get unlimited advertising for free

Not as simple. boost:: is extension of std::. U++ Core is negation of std::

I'd say boost:: is a bunch of quite useful stuff, which extends and often replaces std::. For example Boost.Bind. There is no MPL in std::. There is no serialization in std::. There is no multi-indexed containers in std::. There is no lazy function invocation in std::. E.t.c.

STL is just about containers, iterators, and algorithms.
