
Subject: Re: [c++11] Problem with executing callback with Vector

Posted by [mirek](#) on Tue, 02 Jun 2015 18:35:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

Zbych wrote on Sat, 30 May 2015 19:20mirek wrote on Fri, 29 May 2015 08:32
Passing Vector to callback...

Well, the trouble is you need copy and containers do not have "direct" copy.

Quick fix is IMO to use `Callback1< WithDeepCopy< Vector<int> > >`.

Another quick fix is to use `Value[Array]` instead.

If you insist on using Vector directly (both above options come at certain performance penalty), you need to reorganize the code so that you can pass reference or pointer.

Anyway, as long as you want to use `PostCallback`, I think your example is not correct, as `PostCallback` accepts simple `Callback` (no parameters). Perhaps you meant using something like `THISBACK1` ?

If so, you could perhaps use lambda to overcome the problem...

Mirek

If you have time, please take a look at `ClientHandler::OnProcessMessageReceived` from ChromiumBrowser package:

<https://code.google.com/p/upp-mirror/source/browse/trunk/bazaar/ChromiumBrowser/ClientHandler.cpp>

This function receives message from V8 rendering process - message name and a vector of values.

I want to convert `CefValue` to `Upp::Value` and pass it to GUI thread. The problem is that I don't want to store this vector somewhere and pass a reference since I don't know how long I should keep it. The easiest way is to pass a copy of vector to callback.

But maybe there is some other way. To be honest, I don't know what do you mean by `Value[Array]`.

Regards,
Zbych

I mean `ValueArray` par, instead of `Vector<Value>` par.

If you want to keep parameter as `const Vector<Value>&` (it is certainly nicer), you can perhaps solve it by creating `Vector<Value>` on the heap, pass pointer to some auxiliary callback in `PostCallback` (grrr, naked heap pointer... but here it might be acceptable), in this callback call `WhenMessage`, then delete pointer. Something like

```
void HelperCallback(Callback1<const Vector<Value>&> target, Vector<Value> *vec) {  
    target(*vec);  
    delete vec;  
}
```

```
....  
Vector<Value> *vec = new Vector<Value>();  
vec->Add(...);  
....  
PostCallback(callback(HelperCallback, vec));
```
