Subject: C++ and HTML+Javascript
Posted by forlano on Tue, 02 Apr 2024 15:30:33 GMT
View Forum Message <> Reply to Message

Hello,

I found this software https://saucer.github.io/
that looks very interesting. It combine C++ with a GUI created in HTMLS, CSS and Javascript.
Unfortunately I was not able to setup it and give it a try.

I wonder if some U++ user has already tried with success a similar approach to combine the backend in C++ with the frontend in HTML5?

Thanks,
Luigi

Subject: Re: C++ and HTML+Javascript
Posted by zsolt on Wed, 03 Apr 2024 02:43:15 GMT
View Forum Message <> Reply to Message

Hi Luigi,

our invoicing app's website works in a similar way.

The backend is U++ (HTTP, JSON), the frontend is vanilla Javascript + Bootstrap CSS + HTML.

My next project will be the JS frontend for the invoicing app.
The U++ backend is in production already. The desktop app uses it currently as an HTTPS client.
Much faster and more scalable than using an open PostgreSQL port.

U++ is absolute usable for this type of development.

Subject: Re: C++ and HTML+Javascript
Posted by forlano on Wed, 03 Apr 2024 06:16:57 GMT
View Forum Message <> Reply to Message

Hi Zsolt,

very interesting!
Is it based on skylark? The gui and U++ comunicate via HTTP?

Luigi

Subject: Re: C++ and HTML+Javascript
Posted by zsolt on Wed, 03 Apr 2024 20:22:54 GMT
View Forum Message <> Reply to Message

Hi Luigi,

it is not skylark.

It is a very simple single process multi threaded server.

It binds to a port, starts a fixed number of threads to serve the incoming requests and parses headers with HttpHeader class.
After that it finds out the endpoint from the URL.

There is a macro generated map with the registered endpoints and based on them, it calls a macro generated function, that parses the HTTP body into the struct, that is the argument of that specific endpoint method.
After that it calls that method within a server class with the deserialized argument.
I use Jsonize here.
Then the method does whatever it wants and returns an other struct.
Then the macro generated code serializes it to JSON and sends it back to the client.

So yes, the U++ client communicates with this remote U++ (HTTP + JSON) server.

On client side, the code is just calling the same methods with the jsonizeable structs and a macro generated code does the rest.

The invoicing software can also run with a local sqlite or postgresql database. For this case there are other macros, generating the same methods in a class, that calls the endpoint code directly as methods.

Implementing the server and client side code is very easy this way
You just have to write normal methods on server side and call the methods with the same name on client side. And the new Assist is helping you also.

There are some more complex things in it also. For example the exceptions thrown in the server code are propagated to the client side, so the error handling is quite simple as well.

There is a macro, generating the callable javascript functions also.

This "framework" is similar to the database framework of U++.

I run the server side U++ daemon behind Nginx on Linux.

Subject: Re: C++ and HTML+Javascript
Posted by forlano on Thu, 04 Apr 2024 12:09:32 GMT
View Forum Message <> Reply to Message

Hi Zsolt,

your approach seems very promising.

I am not expert in this field and I am not able to replicate what you have done.
But if it can work on a remote server than should work with a local server too. If I have understood, then a desktop app can be built using U++, HTML, CSS and Javascript using a browser as renderer.

I wish one day to see a "hello world" test case built with such tecnology.

Best regards,
Luigi

Subject: Re: C++ and HTML+Javascript
Posted by zsolt on Tue, 09 Apr 2024 21:49:04 GMT
View Forum Message <> Reply to Message

Hi Luigi,

Quote:If I have understood, then a desktop app can be built using U++, HTML, CSS and Javascript using a browser as renderer.
Yes, but my purpose is to run the JS GUI in a web browser.

Quote:I wish one day to see a "hello world" test case built with such tecnology.
I plan releasing the core of it. Currently the code is messy with a lot of different older experimental APIs and it is not separated from the app enough. After finishing the web based GUI, I think it will be evolved to a more general and cleaner state.
