How to keep a C++ realtime server application with a modern web client interface?

Asked 16 years, 1 month ago Modified 15 years, 4 months ago

Viewed 2k times

1

I develop industrial client/server application (C++) with strong real time requirements. I feel it is time to change the look of the client interface - which is developed in MFC - but I am wondering which would be the right choice. If I go for a web client is there any way to exchange data between C++ and javascript other than AJAX <-> Web service <-> COM ? Requirements for the web client are: Quick statuses refresh, user commands, tables

c++ ajax com

Share

Improve this question

Follow

asked Nov 18, 2008 at 21:06

Alessio

4 Answers Sorted by: Highest score (default)

2

My team had to make that same decision a few months ago...

The cool thing about making it a web application would be that it would be *very* easy to modify later on. Even the user of the interface (with a little know-how) could modify it to suit his/her needs. Custom software becomes just that much easier.

We went with a web interface and ajax seems the way to go, it was quite responsive.

On the other hand, depending on how strong your real time requirements are, it might prove difficult. We had the challenge of plotting real time data through a browser, we ended up going with a firefox plugin to draw the plot. If you're simply trying to display real time text data, it shouldn't be as big an issue.

Run some tests for your specific application and see what it looks like.

Something else to consider, if you are having a web page be an interface to your server, keep in mind you will need to figure a way to update one client when another changes the state of the server if you plan on allowing multiple interfaces to your server.

Share Improve this answer

Follow

answered Nov 18, 2008 at 21:17

Jeffrey Martinez

4,594 ● 3 ● 33 ● 36

2

I usually build my applications 2-folded :

1. Have the real heavy-duty application CLI-only. The protocol used is usually text-only based, composed of requests and answers.
2. Wrap a GUI around as another process that *talks* to the CLI back-end.

The web interface is then just another GUI to wrap around. It is also much easier to wrap a REST/JSON based API on the CLI interface (just automatically translate the messages).

The debugging is also quite easy to do, since you can just dump the requests between the 2 elements and reproduce the bugs much more easily.

Share Improve this answer

Follow

answered Jul 31, 2009 at 15:31

Steve Schnepp

4,680 ● 6 ● 41 ● 56

1

Write an HTTP server in your server to handle the AJAX feedback. If you don't want to serve files, create your server on a non-standard port (eg. 8081) and use a regular web server for the actual web page delivery. Now have your AJAX engine communicate with the server on the Bizarro port instead of port 80.

But it's not that hard to write the file server part, also. If you do that, you also get to generate web pages on-the-fly with your data pre-filled, if you want.

Google Desktop Search does this now. When I search my desktop for 'foobar', the URL that opens is this:

> http://127.0.0.1:4664/search?q=foobar&flags=68&num=10

In this case, the 4664 is the Bizarro port. (GoogleDesktop serves all the data here; it only uses the Bizarro port to avoid conflicts with any web server I might be running.)

Share Improve this answer

Follow

answered Nov 18, 2008 at 21:32

Phil Hord

13.1k ● 1 ● 29 ● 30

0

You may want to consider where your data lives. If your application feeds a back-end database, you could write a web app leaving your c++ code in tact -- the web application would be independent and offer up pages to web users and talk directly to the database -- In this case you have as many options, and more, as you have indicated.

Share Improve this answer

Follow

answered Nov 18, 2008 at 21:10

Borzio

845 ● 1 ● 6 ● 12