

Re: VB6 Winsock action on Server

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2006-05/msg02489.html>

- *From:* jerry_ys <jerryys@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 30 May 2006 21:33:01 -0700
-

Thank you for reponse. I've been away.

Your point regarding placing a file in non exclusive mode rang the bell. For some illogical reason I was putting the dll defined variables in the same category as the file the dll must access. You can see my problem if a copy of the file was also created with each instance and all instances could change the file.

I wish it were as simple as one std exe that accumulates client requests and then processes in order. No one client request is related to any other.

Please comment if you have any stats on storage devices that have multiple read/write heads with their own controller so that a one main thread type server could not possibly utilize the capabilities of the additional read/write heads. This is why I believe the multiple thread multi user dll's will take advantage of this technology.

Is specifying multi thread, multi user dll limit the dll to only one instance? And if yes would single thread with mulitple instances be better for controlling the threads through what MS already provides, or not?

Thanks

Jerry Y

"Schmidt" wrote:

"jerry_ys" <jerryys@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> schrieb im Newsbeitrag <news:D3F6CA25-1399-45AF-8F65-57B8C1A14182@xxxxxxxxxxxxxxxxxxxx>

Just to verify your response at the very top level before I

Re: VB6 Winsock action on Server

get too deep into it, using an active X exe as the server program that defines the winsock cntl and receives multiple client requests to perform different, or the same tasks that are in a active x dll which this server will use will create a separate dll instance for each client request process it, and return to the active x exe the client data that the exe will then pass back to the client's own desktop provided multi-threading is specified for the active x dll?

Yes.

The right client will get the right data back?

Of course, if you place/send the (serialized) response on the appropri. "socket-channel", this client has established before after a successful connect.

If the dll above accesses a large file, when it creates an instance of itself for each client request, will it also make it's own copy of the file?

Not sure, what you mean here exactly.

If multiple DLL-Instances want to access the same file at the same time, then this file has to be opened in a nonexclusive mode.

Synchronization problems?

That depends – if the DLL-Instances only read from this file, then there are no Sync-Problems.

It's another story, if each DLL-Instances can change content inside the file.

Here you will have to use your own "Transaction-Mechanism" – if the file is an *.mdb, then you could use the Transaction-Mechanisms of the underlying DB-Engine.

Or, would the dll be better as the server with the winsock controls array, and it instead would call the active x exe to service the client, since

the exe

does not make instances of itself and can better control the data access

and

keep it synchronized.(one copy)

Seems you have a scenario, where the server has to ensure an "in-order-access"

Re: VB6 Winsock action on Server

to a single server–resource (large File) for requests from multiple clients. This is best solved, if you work singlethreaded with the Winsock–Array inside a normal Standard–Exe as the Server (AX–Exe not necessary). Inside the DataArrival of the SockArray you simply cumulate the incoming Data for each Client–Connect separately until the "JobData" is complete. Then you simply have to start the "JobWorkerRoutine" with the appropriate Data, related to the ClientConnect(–Index) and send the result back over the appropriate Winsock–Instance. Since you are working singlethreaded, no other ClientJob can be executed at the same time, as long as you don't execute DoEvents anywhere inside the JobWorkerRoutine.

If you want to use my RPCServer–Environment, you can achieve the same effect (serialized access to a single server–resource) by placing the appropri. Code inside a RPC–Singleton–Object, wich the RPCServer allows to access from inside the WorkerThreadPool.

Olaf