Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 23:15:46 GMT
View Forum Message <> Reply to Message

ServerWorker - a multithreaded high load server.
I'm of course aware of modern Skylark functionality created by Mirek. But there are cases when you need not a front-end server, but instead you need very effective intermediate processing server which effectively handles multiple connections simultaneously.
The base concept of this approach is a "worker". Worker is an instance executed in it's own thread which processes single user's request and finishes. In my implementation, if there's a free worker it is used for new request, or new worker/thread is added.
This approach leads to very efficient and extremely lightweight server which handles large amount of multiple connections.
Actually I think it is one of useful things among published, but it is up to you - if you consider it useful for your tasks (high load is not a kind of task you frequently meet in real life).

To demonstrate the class is really easy to use, I'll post example here.

How server with custom workers is created and used:

```
 TcpSocket server;
 server.Listen(port,connections);
 ServerWorker::ServerThread<CommentWorker>(server);
 ServerWorker::ClearWorkers_();
```

Making custom worker itself:

```
#ifndef _commentd_CommentWorker_h_
#define _commentd_CommentWorker_h_

#include <Core/Core.h>
#include "ServerWorker.h"
using namespace Upp;

class CommentWorker : public ServerWorker
{
public:
 CommentWorker()
 {
  ServerWorker::AddHandler<CommentWorker>("/login",
&CommentWorker::HandleAddLogin);
  ServerWorker::AddHandler<CommentWorker>("/comment",
&CommentWorker::HandleAddComment);
  //...
 }

 void HandleAddLogin(const String &request, HttpHeader &header, TcpSocket &socket)
 {
  VectorMap<String,String> values = ParseUrlQueryString(header.GetURI());
  String valueLogin = values.GetAdd("login");
```

```
  String valuePass = values.GetAdd("pass");
  //...
  HttpResponseKeepAlive(
   socket, false, 200, "OK", "text/html; charset=\"utf-8\"",
   HTML, NULL,
   VectorMap<String,String>()
  );
 }

 //...
};

#endif
```

That's it. You don't care about creating threads or workers. You just write handlers keeping in mind the multithreaded nature of your code. PersistentStorage classes fit here perfectly.
The current state of this class is almost beta. It is working but some non critical issues are found (i.e. longer answer timeout on Windows) as well as functionality itself is yet to be finalized.

I'd be very grateful on your suggestions and thought on this topic.

File Attachments

1) ServerWorker.h, downloaded 596 times