Subject: Re: SCGI Class
Posted by mirek on Mon, 28 Mar 2011 06:54:41 GMT
View Forum Message <> Reply to Message

zsolt wrote on Sun, 27 March 2011 16:51mirek wrote on Sun, 27 March 2011 10:02Mindtraveller wrote on Sat, 26 March 2011 18:00zsolt wrote on Sat, 26 March 2011 00:55BTW it will not solve the problem of long running SQL queries or doing some slow communication in the scgi process.

In such situations you will have to start a lot of scgi processes to be able to handle the traffic.

Increasing that number, just allows the scgi process to have a large backlog.
What if in the main cycle the the newly created client socket is processed in another thread while server socket is free to accept more connections?

```
while (run)
{ if  (serverSocket.Accept(&clientSocket)
  {
    GetThreadFromPoolAndProcess(clientSocket);
  }
}
```

Actually, this can be quite nicely with something like

```
while(run)
{ if  (serverSocket.Accept(&clientSocket) {
     DoWork();
  }
}
```

and then simply starting several threads to run this loop (with single clientSocket). As accept is reentrant and MT safe, it would return only for single thread running, thus managing the thread pool.

This is a very elegant way to solve the problem using threads.

My only problem with MT in situations like this is, that a few things in Upp are global, such as lenguage-dependant setups (e.g. date or number formatting).

It is a common requirement now from a web based app to show content in the user's language, so I think multi process arrangement would be better.

Or is it somehow possible to make these global Upp settings thread local?

I guess that while there is no direct support, it is in fact quite easy to do in app, as there is:

String	GetLngString(int lang, const char *id);

-> #undef t_, replace it with your own version which is thread local... Or perhaps just use 'lang' parameter.
