U++ - Feature #523
Periodic background jobs in skylark

09/18/2013 07:49 PM - Jan Dolinár

Status:	Patch ready	**Start date:**	09/18/2013
Priority:	Normal	**Due date:**	
Assignee:	Jan Dolinár	**% Done:**	50%
Category:	Skylark	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

Sometimes it is handy to be able to execute some periodic job inside the server that runs in background, independently from request handling threads. Probably most common example of this would be expiration of temporary data (either on disk or in database). Currently, it is not easy to do such thing in Skylark, as you can only react when a request comes and there is no guarantee when or if a request occurs. Also doing this correctly in thread is troublesome as SkylarkApp::quit is private, so it is not simple to quit the thread when the application is terminated.

So I wrote a patch that adds AddJob(Callback cb, int period=60) function to the public interface of SkylarkApp. This function can be used to add multiple jobs that will be executed with given periodicity (in seconds) in a background thread. The timing is not guaranteed to be precise and is only approximate, as longer job can postpone other jobs. If there are no jobs added, the extra thread is not created.

Example of usage:

```
struct MyApp : SkylarkApp {
    MyApp() {
        root = "myapp";
    }
};

void job1() {
    LOG("JOB1 "<<GetSysTime());
}

void job2() {
    LOG("JOB2 "<<GetSysTime());
}

CONSOLE_APP_MAIN
{
    MyApp()
    .AddJob(callback(job1),3)
    .AddJob(callback(job2),5)
    .Run();
}
```

What do you think Mirek, do you mind if I commit it? I'll of course add proper documentation if you say it is ok.

I think this might be even used internally to periodically expire the sessions. I think the way it is done now (on every 1000th request per thread, if I remember correctly), can lead to piling up way too many sessions in database, which in turn caused deadlocks in database on delete statements for me.

History

#1 - 09/22/2013 09:59 AM - Miroslav Fidler

- Assignee changed from Miroslav Fidler to Jan Dolinár

Generally, I like the idea, but I would like to be able to schedule one time job too and the ability to kill the job. That of course would require some for of identification, I guess String would do... For periodic job, I think negative number convention similiar to Ctrl time callbacks would be fine..

```
MyApp
  .AddJob("job1", callback(job1), 2)
  .AddJob(callback(job2), -1)

.KillJob("job1")
```

Also, while suggested implementation works fine with small number of jobs and it is likely that this number will be ever low, perhaps it would not be much harder to use InVector for optimized code...

#2 - 09/22/2013 11:05 AM - Jan Dolinár

- % Done changed from 100 to 50

I was thinking about similar interface to what is in CtrlCore (or using bazaar/Timer). I decided to drop support for that because in preforked mode there would be no simple way to communicate the commands back to the timer thread. (Or is there some shared memory or other IPC mechanism in U++?).

Right now, one simple solution I can imagine is to represent jobs as files (or optionally store them in database, as with sessions). But you can't simply store a callback, so all "schedulable" callbacks would have to be registered at the application start. At best, you could pass some simple arguments through the file (strings, numbers etc.). Would that be ok?

Having one time jobs would be nice too, so they can be scheduled from web interface, I'll think about that. Also, using InVector shouldn't be a problem.

#3 - 10/07/2013 09:24 AM - Miroslav Fidler

http://www.ultimatepp.org/forum/index.php?t=msg&#38;goto=40912&#38;#msg_40912

Files

jobs.patch	2.64 KB	09/18/2013	Jan Dolinár