µWeb - Feature #526

Improve performance of the router by pre-compiling all routes

2011-11-29 13:27 - Elmer de Looff

Status:	Closed	**Start date:**	2011-11-29
Priority:	Normal	**Due date:**	
Assignee:	Elmer de Looff	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	4.00 hours

Description

The performance of the router could be improved upon by compiling all paths before using them.

The current use of the router is that the existing routes and the requested path are passed in, and the result will be returned by this function. Compiling in this situation would not result in any benefit, because the compile result would not be stored.

However, the Router function could be turned into a closure that returns a function that takes the request-path and returns the correct handler from a list of routes.

Ideally, the methods to return could be pre-resolved as well, removing the attribute lookup entirely.

Associated revisions

Revision 3107:e40dd669c2cf - 2011-11-29 16:20 - Elmer de Looff

Refactor of the Router in uWeb. This not pre-compiles all routes, and prefetches all handler methods from the page_class. Updated relevant code and documentation and fixed a few style nits. This resolves #526.

Revision 107:fa14894b444d - 2011-11-29 16:20 - Elmer de Looff

Refactor of the Router in uWeb. This not pre-compiles all routes, and prefetches all handler methods from the page_class. Updated relevant code and documentation and fixed a few style nits. This resolves #526.

History

#1 - 2011-11-29 16:25 - Elmer de Looff

- Status changed from New to Resolved

- % Done changed from 0 to 70

Updated the Router and Handler code to use precomputed routes and methods. The router now returns unbound methods from the appropriate PageMaker class, which only needs an instance thereof (created from the request, so not precomputable) and the arguments from the request.

Following are benchmark results using ApacheBench. The scores are based on best out of four runs with the following command: ab -n20000 -c4 http://localhost/login

We're requesting the Login page for Lintme.com 20.000 times, using 4 concurrent connections. This is done on my laptop, running all cpu governors for 'performance' (i5 M560 @ 2.67GHz).

Old code with login as second (out of 30) routes, **2227 req/s**:

```
Concurrency Level:      4
Time taken for tests:   8.977 seconds
Complete requests:      20000
Failed requests:        0
Write errors:           0
Total transferred:      94120000 bytes
HTML transferred:       90720000 bytes
Requests per second:    2227.95 [#/sec] (mean)
Time per request:       1.795 [ms] (mean)
Time per request:       0.449 [ms] (mean, across all concurrent requests)
Transfer rate:          10239.00 [Kbytes/sec] received
```

Old code with login as second-to-last route, **2083 req/s**:

```
Concurrency Level:      4
Time taken for tests:   9.601 seconds
Complete requests:      20000
Failed requests:        0
Write errors:           0
Total transferred:      94120000 bytes
HTML transferred:       90720000 bytes
Requests per second:    2083.14 [#/sec] (mean)
Time per request:       1.920 [ms] (mean)
Time per request:       0.480 [ms] (mean, across all concurrent requests)
Transfer rate:          9573.48 [Kbytes/sec] received
```

New code with login as second (out of 30) routes, **2270 req/s**:

```
Concurrency Level:      4
Time taken for tests:   8.811 seconds
Complete requests:      20000
Failed requests:        0
Write errors:           0
Total transferred:      94120000 bytes
HTML transferred:       90720000 bytes
Requests per second:    2270.00 [#/sec] (mean)
Time per request:       1.762 [ms] (mean)
Time per request:       0.441 [ms] (mean, across all concurrent requests)
Transfer rate:          10432.25 [Kbytes/sec] received
```

New code with login as second-to-last route, **2220 req/s**:

```
Concurrency Level:      4
Time taken for tests:   9.008 seconds
Complete requests:      20000
Failed requests:        0
Write errors:           0
Total transferred:      94120000 bytes
HTML transferred:       90720000 bytes
Requests per second:    2220.22 [#/sec] (mean)
Time per request:       1.802 [ms] (mean)
Time per request:       0.450 [ms] (mean, across all concurrent requests)
Transfer rate:          10203.49 [Kbytes/sec] received
```

Despite the best of four runs, there is a considerable amount of noise. The slight performance boost in the best case scenario using the new code is likely only minimal. The drop in performance from having to go through 30 regexen is significantly reduced with the new code, which is in line with expectations.

#2 - 2011-11-29 17:55 - Elmer de Looff

- Status changed from Resolved to Closed

- % Done changed from 70 to 100

All fixed, last bugs removed.