Philo - Feature #164

Optimization work

06/22/2011 01:51 PM - Stephen Burrows

Status:	Closed	**Start date:**	06/22/2011
Priority:	High	**Due date:**	
Assignee:	Stephen Burrows	**% Done:**	0%
Category:	Core	**Estimated time:**	0.00 hour
Target version:			

Description

Working on optimizing philo - see also: http://www.codinghorror.com/blog/2011/06/performance-is-a-feature.html

Will use this issue to track progress.

History

#1 - 06/22/2011 01:53 PM - Stephen Burrows

- Category set to Core

Commit 5da36a2 test times in seconds (max, min, avg):

```
Before: 11.373934 05.021761 05.505069
After:  07.170535 03.784184 04.082582
```

#2 - 06/22/2011 01:55 PM - Stephen Burrows

Incidentally, tests are run in python with urllib. The test page is accessed 100 times and is (probably) not representative of every feature that philo offers.

#3 - 06/22/2011 02:08 PM - Stephen Burrows

Commit 6164c1b test times in seconds (max, min, avg), reusing the 'after' statistics from the previous commit.

```
Before: 07.170535 03.784184 04.082582
After:  05.391495 03.684451 03.785000
```

#4 - 06/22/2011 04:02 PM - Stephen Burrows

Commit 7ebd19f test times in seconds (max, min, avg), reusing the 'after' statistics from the previous commit.

```
Before: 07.170535 03.784184 04.082582
After:  07.660383 03.617115 03.862377
```

No significant overall difference, though the first page load (when caches are warmed) seems slightly faster.

#5 - 06/22/2011 04:03 PM - Stephen Burrows

That last one should've read:

```
Before: 05.391495 03.684451 03.785000
After:  07.660383 03.617115 03.862377
```

#6 - 06/22/2011 04:38 PM - Stephen Burrows

[optimization adbb2ab] Added memoization (optional but enabled by default) to TreeEntity.get_path.

This optimization mostly helps pages that make heavy use of shipherd, node_url, and (more generally) node.get_absolute_url. The principal is that in the course of a Node (or other TreeEntity subclass) instance's life, its path is not likely to change. In cases where change is important, developers can pass in memoize=False to force evaluation.

develop: 11.373934 05.021761 05.505069
5da36a2: 07.170535 03.784184 04.082582
6164c1b: 05.391495 03.684451 03.785000
7ebd19f: 07.660383 03.617115 03.862377
adbb2ab: 04.043702 03.158195 03.265863

#7 - 06/22/2011 09:02 PM - Stephen Burrows

```
e829cc9: 04.983335 03.052666 03.135104
```

#8 - 06/24/2011 01:39 PM - Stephen Burrows

Cacheless shipherd (fa49fa4) vs. optimization otherwise (c679970):

```
c679970: 06.403735 03.060383 03.146596
fa49fa4: 04.328504 02.848535 02.933984
```

#9 - 06/24/2011 03:15 PM - Stephen Burrows

Unfortunately, those results were bogus; the cacheless shipherd code was just not running, because of template filter errors. Once those were corrected, cacheless lost by a wide margin. Currently trying to build a better cache.

#10 - 06/24/2011 06:33 PM - Stephen Burrows

Optimization 13aa220 - this particular commit made single-level navigation structures (as are used on the test site) much faster.
Better cache a05c936 - this is a version that actually works. ;-)

```
13aa220: 04.884354 02.901736 02.995597
a05c936: 04.300317 02.871088 02.943538
```

#11 - 06/24/2011 07:31 PM - Stephen Burrows

Turns out that a05c936 actually still had some slight issues, but they did not affect performance. In cases where a navigation item didn't have a target node, the navigation was evaluated, but not displayed. Correcting the issue actually seems to have improved performance slightly, though it could be slight a glitch in the results.

```
7b3b752: 04.403769 02.803217 02.883353
```

#12 - 06/26/2011 09:19 PM - Stephen Burrows

I think that it might be possible to do an optimization for shipherd and sobol similar to the one described in slides 41-44 of the first presentation at http://justcramer.com/2011/06/24/europython/. We're already mostly there. I'm not sure, though, whether the technique really applies to the problem at hand.

#13 - 06/27/2011 06:41 PM - Stephen Burrows

```
12c9cc7: 04.337620 02.821180 02.913605 - fast containers with reference prefetching
829ebbd: 04.480525 02.766281 02.836390 - fast containers without reference prefetching
```

It should be noted that reference prefetching would probably be more efficient given a template that referenced many instances of the same model; however, it has (in uses of philo I have seen) been more common to reference various models once each. In this case, prefetching is slower.

#14 - 06/27/2011 08:47 PM - Stephen Burrows

```
eeb8a55: 04.446627 02.752377 02.837770 - now caches philo-root reversal in Node.construct_url
```

At this point it should be mentioned that these tests are being run using the locmem cache backend, which is much slower than, say, memcached, and which consequently results in comparable performance rather than improved performance for this particular commit.

#15 - 06/28/2011 06:44 PM - Stephen Burrows

```
b7bb8ec: 04.439187 02.615551 02.690186
183b0a2: 04.157114 02.611704 02.675582
```

The difference in 183b0a2 (which introduced caching of target urls on TargetURLModel instances) is not significant enough to be considered a clear improvement; however, I think that may have something to do with the site that was being used to test this. I'm marking this topic as resolved for now. It's possible that there are some more optimizations that can be found, but they don't need to be found right now.

#16 - 07/06/2011 01:20 PM - Stephen Burrows

- Status changed from In Progress to Closed

Marking as closed for now.