### Урок 7. Еще быстрее: кеширование в Django

1. Найти в проекте повторяющиеся вызовы методов для одного экземпляра модели и применить к ним декоратор «@cached\_property». Оценить, насколько уменьшилось число дублей при выполнении SQL-запросов и каков прирост производительности.

Переделал функции подсчета количества и сумм, в заказе, выигрыш микроскопический) так же как в лекции, один запрос экономится. Но может я что-то делаю не так.

2. Применить тег «with» в одном из шаблонов. Оценить, насколько уменьшилось число дублей при выполнении SQL-запросов и каков прирост производительности.

Добавил метод «get\_summary» в класс заказов, и использовал его в шаблоне «inc\_order\_summary.html» увы это не принесло ожидаемого результата, думаю потому, что значения используются один раз и прибавляется время на создание кэша.

3. Установить и настроить приложение «Memcached». Реализовать кеширование на низком уровне для функций, возвращающих редко изменяющиеся данные (продукты каталога, список категорий и т.д.). Оценить прирост производительности.

Sudo apt install memcached

Sudo apt install libmemcached-dev

source /home/django/venv/bin/activate

/etc/memcached.conf

-m 64 -> -m 256

systemctl restart memcached.service

3.1 кеширование шаблонов, не дало прироста, думаю из-за того, что основную проблему я устранил

3.2 кеширование контроллера, видимо с cbv не так все просто, но выигрыш все равно есть, 0.3 мс против 2.5 мс.

3.3 кеширование в урлах.

3.4 Низкоуровневое кеширование. Нарушило всю красоту в cbv. Так же попробовал кешировать получение корзины в контекстном процессоре, но похоже это не дает результата.

3.5 Кеширование всего сайта. Эффект есть) но нужно понимать, что кешируется абсюлютно все, не знаю даже как применить на практике.

7. Добавить замеры в файл с 6 урока с применением кэша

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Параметр** | **-r50 -c50** | **С кэшем** | **-r25 -c100** | **С кэшем** | **r17 -c150** | **С кэшем** | **-r12 -c200** | **С кэшем** |
| Число переходов | 34850 | 34850 | 32471 | 34800 | 35961 | 35961 | 34136 | 34172 |
| Доступность сервера | 100.00 | 100.00 | 99.36 | 100.00 | 100.00 | 100.00 | 99.64 | 99.65 |
| Время теста, с | 628.96 | 655.15 | 638.10 | 692.87 | 498.63 | 606.21 | 459.52 | 564.86 |
| Время отклика, с | 0.90 | 0.93 | 1.94 | 1.96 | 2.02 | 2.47 | 2.61 | 3.19 |
| Запросов в секунду | 55.41 | 53.19 | 50.89 | 50.23 | 72.12 | 59.32 | 74.29 | 60.50 |
| Пропускная способность, МБ/сек | 8.39 | 8.03 | 8.07 | 7.54 | 10.85 | 8.92 | 12.89 | 10.47 |
| Согласованность | 49.64 | 49.57 | 98.73 | 98.34 | 145.44 | 146.26 | 193.66 | 192.78 |
| Удачных транзакций | 34850 | 34850 | 32471 | 34800 | 36018 | 36018 | 34244 | 34280 |
| Неудачных транзакций | 0 | 0 | 210 | 0 | 0 | 0 | 122 | 119 |
| Самая долгая транзакция, с | 39.85 | 28.40 | 60.10 | 54.53 | 60.01 | 60.01 | 60.01 | 60.01 |

Довольно противоречивые данные на мой взгляд я получил. С одной стороны, я видел эффект, однако на замерах это не сильно отразилось. Время отклика везде выросло, самая долгая транзакция сначала уменьшилась, но с ростом числа пользователей сравнялась. Доступность сайта во втором тесте стала 100%. В общем фундаментальных изменений я не вижу, думаю, что при наличии каких либо вопросов по быстродействию системы нужен более глубокий анализ.