**Тема занятия № 25: Поддержка баз данных PostgreSQL. Кэширование**

1. Кэширование на стороне сервера

Когда веб-обозреватель получает от веб-сервера какой-либо файл, он сохраняет его на локальном диске — выполняет его кэширование. Впоследствии, если этот файл не изменился, веб-обозреватель использует его кэшированную копию вместо того, чтобы вновь загружать с сервера, — это заметно увеличивает производительность.

Так работает кэширование на стороне клиента.

Django предоставляет ряд инструментов для управления кэшированием на стороне клиента. Мы можем отправлять в заголовке запроса временную отметку последнего изменения страницы или какой-либо признак, указывающий, изменилась ли она с момента последнего обращения к ней. Также мы можем отключать кэширование каких-либо страниц на стороне клиента, если хотим, чтобы они всегда отображали актуальную информацию.

Помимо этого, Django может выполнять кэширование на стороне сервера, сохраняя какие-либо данные, фрагменты страниц или даже целые страницы в особом хранилище — кэше сервера. Благодаря этому можно увеличить быстродействие сайтов с высокой нагрузкой.

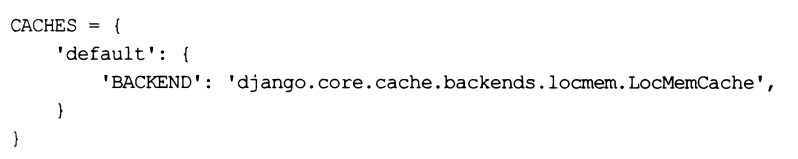
Кэш стороны сервера в Django организован по принципу словаря Python: каждая кэшируемая величина сохраняется в нем под уникальным ключом. Ключ этот может как генерироваться самим Django на основе каких-либо сведений (например, интернет-адреса страницы), так и задаваться произвольно.

Настройка подсистемы кэширования на стороне сервера. Все настройки этой подсистемы указываются в параметре caches модуля settings.py пакета конфигурации.

Значением этого параметра должен быть словарь. Ключи его элементов указывают псевдонимы созданных кэшей, которых может быть произвольное количество. По умолчанию будет использоваться кэш с псевдонимом default.

В качестве значений элементов этого словаря также указываются словари, хранящие собственно параметры соответствующего кэша. Каждый элемент вложенного словаря указывает отдельный параметр.

Вот значение параметра caches по умолчанию:

****

Оно задает единственный кэш, используемый по умолчанию и сохраняющий данные в оперативной памяти.

Доступные для указания параметры кэша:

□ backend— строка с именем класса, выполняющего сохранение кэшируемых данных. В составе Django поставляются следующие классы:

• django.core.cache.backends.db.databasecache — сохраняет данные В указанной таблице базы данных. Обеспечивает среднюю производительность и высокую надежность, но дополнительно нагружает базу данных;

• django.core.cache.backends.filebased.filebasedcache — сохраняет данные в файлах, находящихся в заданной папке. Немного медленнее предыдущего класса, но не нагружает базу данных;

• django.core.cache.backends.locmem.locmemcache — сохраняет данные В оперативной памяти. Дает наивысшую производительность, но при отключении компьютера содержимое кэша будет потеряно;

• dj ango. Core. Cache. Backends. Memcached. Memcachedcache — использует ДЛЯ хранения данных популярную программу Memcached (ее применение будет описано далее);

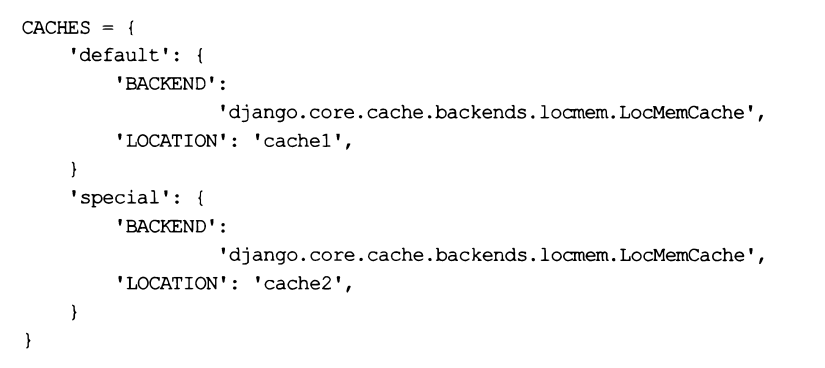
• django.core.cache.backends.dummy.dummycache — вообще не сохраняет кэшируемые данные. Служит исключительно для отладки;

□ location— назначение параметра зависит от класса, указанного в параметре Backend:

• имя таблицы — если выбран класс, хранящий кэш в таблице базы данных;

• полный путь к папке — если выбран класс, хранящий данные в файлах по указанному пути;

• псевдоним хранилища — если выбран класс, хранящий данные в оперативной памяти. Указывается только в том случае, если используются несколько кэшей такого типа. Пример:

****

• интернет-адрес и используемый TCP-порт— если выбран класс, сохраняющий данные в Memcached.

Значение по умолчанию — "пустая” строка;

□ timeout— время, в течение которого кэшированное значение будет считаться актуальным, в виде целого числа в секундах. Устаревшие значения впоследствии будут удалены. По умолчанию: зоо;

□ options — дополнительные параметры кэша. Значение указывается в виде словаря, каждый элемент которого задает отдельный параметр. Поддерживаются два универсальных параметра, обрабатываемые всеми классами-хранилищами данных:

• max entries — количество значений, которые могут храниться в кэше, в виде целого числа (по умолчанию: зоо);

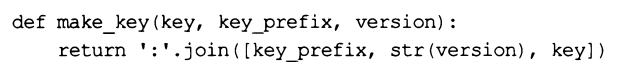
• cull frequency — часть кэша, которая будет очищена, если количество значений в кэше превысит величину из параметра max entries, указанная в виде целого числа. Так, если дать параметру cull frequency значение 2, то при заполнении кэша будет удалена половина хранящихся в нем значений. Если задать значение о, то при заполнении кэша из него будут удалены все значения. Значение по умолчанию: з.

Здесь же можно задать параметры, специфичные для классов-хранилищ, которые поставляются в составе сторонних библиотек;

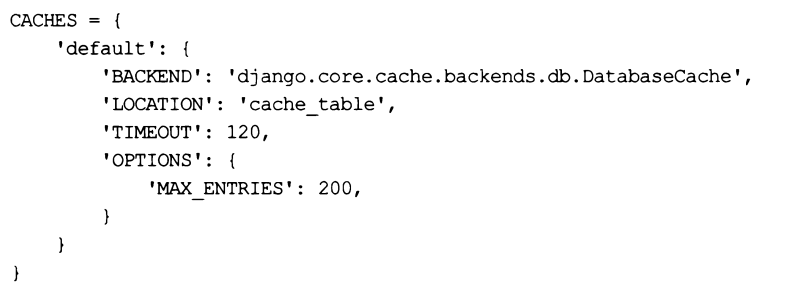
□ key prefix — заданный по умолчанию префикс, который участвует в формировании конечного ключа, записываемого в кэше (по умолчанию — ’’пустая” строка);

□ version — назначенная по умолчанию версия кэша, которая участвует в формировании конечного ключа, записываемого в кэше (по умолчанию: 1);

□ key function— строка с именем функции, которая формирует конечный ключ, записываемый в кэше, из префикса, номера версии и ключа кэшируемого значения. По умолчанию используется функция, которая составляет конечный ключ из префикса, номера версии и ключа, разделяя их символами двоеточия, и имеет следующий вид:

****

Пример указания параметров кэша:

****

**Создание таблицы для хранения кэша**

Если был выбран класс, сохраняющий кэшированные данные в таблице базы данных, то эту таблицу необходимо создать. В этом нам поможет команда Createcachetable утилиты manage.py:



Дополнительный ключ -dry-run выводит на экран сведения о создаваемой таблице, но не создает ее.

**Применение Memcached**

Memcached— популярная программа кэширования, сохраняющая данные исключительно в оперативной памяти.

Чтобы использовать ее для кэширования данных в Django, необходимо:

□ установить дополнительную библиотеку python-memcached подачей команды:

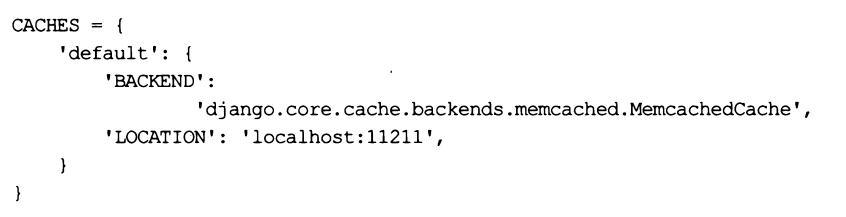


□ записать в параметре BACKEND настроек кэша ИМЯ класса django.core.cache, Backends. Memcached. Memcachedcache;

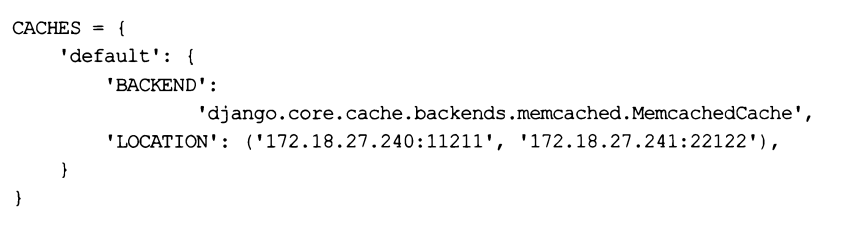
□ указать в параметре location настроек кэша строку формата:



Например, если Memcached установлена на локальном хосте и использует порт по умолчанию № 11211, следует записать:

****

Если кэшируемые данные одновременно сохраняются в нескольких экземплярах Memcached, то в параметре location следует указать список или кортеж из строк в приведенном ранее формате. Например, если для кэширования используются два экземпляра Memcached, доступные по интернет-адресам 172.18.27.240 и 172.18.27.241 и ТСР-портам 112211 и 22122, то следует указать такие настройки:

**** Высокоуровневые средства кэшируют либо все страницы сайта, либо только страницы, сгенерированные отдельными контроллерами.

Ключ, под которым сохраняется кэшированная страница, формируется самим Django на основе ее пути и набора GET-параметров.

**Кэширование всего веб-сайта**

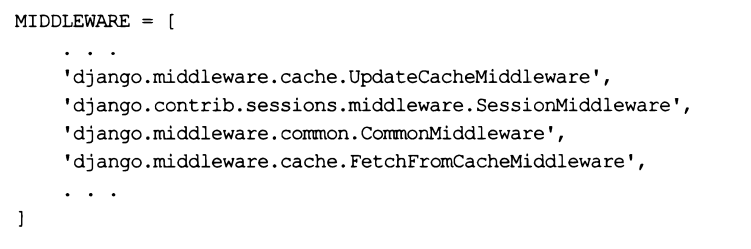
При этом подсистема кэширования Django работает согласно следующим принципам:

□ кэшируются все страницы, сгенерированные контроллерами в ответ на получение GET- и HEAD-запросов, с кодом статуса 200 (т. Е. Запрос был обработан успешно);

□ страницы с одинаковыми путями, но разным набором GET-параметров и разными cookie считаются разными, и для каждой из них в кэше создается отдельная копия.

Чтобы запустить кэширование всего сайта, необходимо добавить посредники django.middleware.cache.updatecachemiddleware И django.middleware.cache.

Fetchfromcachemiddieware в список зарегистрированных в проекте (параметр Middleware):

****

Еще можно указать дополнительные параметры:

□ cache middleware alias — псевдоним кэша, в котором будут сохраняться страницы (по умолчанию: "default");

□ cache middleware seconds — время, в течение которого кэшированная страница удет считаться актуальной, в виде целого числа в секундах (по умолчанию: боо);

□ cache middleware key prefix — префикс конечного ключа, применяемый только при кэшировании всего сайта (по умолчанию — ’’пустая” строка).

В ответы, отсылаемые клиентам и содержащие страницы, добавляются два следующих заголовка:

□ Expires — в качестве значения задается временная отметка устаревания кэшированной страницы, полученная сложением текущих даты и времени и значения из параметра cache\_middleware\_seconds;

□ Cache-Control — добавляется параметр max-age со значением, взятым из параметра CACHE\_MIDDLEWARE\_SECONDS.

Во многих случаях кэширование всего сайта — наилучший вариант повышения его производительности. Оно быстро реализуется, не требует ни сложного программирования, ни переписывания кода контроллеров, ответы которых нужно кэшировать.

При кэшировании всего сайта Django учитывает ситуации, при которых одна и та же страница может генерироваться в разных редакциях в зависимости от каких-либо ’’внутренних” условий (например, выполнил пользователь вход на сайт или нет). Дело в том, что идентификатор сессии сохраняется в cookie, а, как говорилось ранее, страницы с одинаковыми путями, но разными cookie (и GET-параметрами) считаются разными, и в кэше сохраняются их отдельные редакции — для каждого из наборов cookie. Таким образом, ситуации, при которых пользователь открывает страницу в качестве гостя, выполняет вход на сайт, заходит после этого на ту же страницу и получает ее устаревшую, ’’гостевую”, копию из кэша, полностью исключены.

**Кэширование на уровне отдельных контроллеров**

Если же требуется кэшировать не все страницы сайта, а лишь некоторые (например, наименее часто обновляемые и при этом наиболее часто посещаемые), то следует применить кэширование на уровне отдельных контроллеров.

Кэширование на уровне контроллера задействует декоратор cachejpage () из модуля Dj ango.views.decorators.cache:

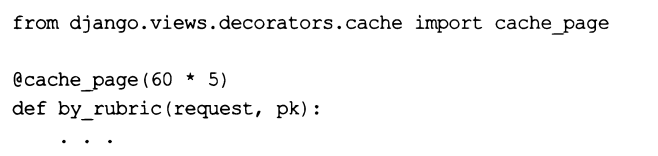
****

Время хранения сгенерированной страницы в кэше задается в виде целого числа в секундах.

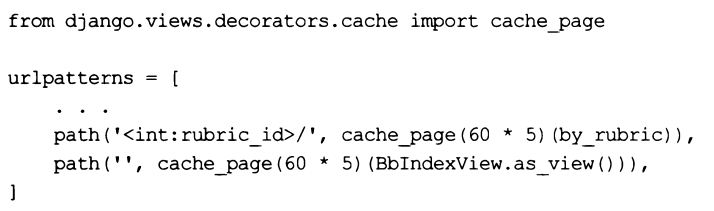
Необязательный параметр cache указывает псевдоним кэша, в котором будет сохранена страница. Если он не указан, будет использован кэш по умолчанию с псевдонимом default.

В необязательном параметре key prefix можно указать другой префикс конечного ключа. Если он не задан, то применяется префикс из параметра key prefix настроек текущего кэша.

Декоратор cache page () указывается непосредственно у контроллера-функции, результаты работы которого необходимо кэшировать:

****

Также этот декоратор можно указать в объявлении маршрута, ведущего на нужный контроллер. Это можно использовать для кэширования страниц, генерируемых контроллерами-классами. Пример:



Выбрав кэширование на уровне контроллеров, мы сможем указать, какие страницы следует кэшировать, а какие— нет. Понятно, что в первую очередь кэшировать стоит страницы, которые просматриваются большей частью посетителей и не изменяются в зависимости от каких-либо ’’внутренних” условий. Страницы же, меняющиеся в зависимости от таких условий, лучше не кэшировать (или, как вариант, применять средства управления кэшированием, которые будут рассмотрены прямо сейчас).

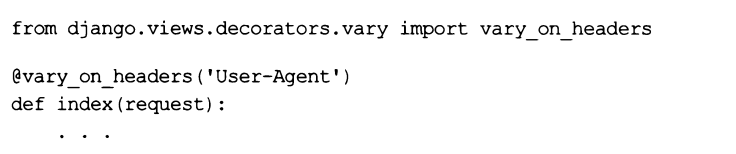
**Управление кэшированием**

Если в кэше требуется сохранять отдельную редакцию страницы для каждого из возможных значений заданного "внутреннего" признака, то следует как-то указать этот признак подсистеме кэширования, поместив его в заголовок vary ответа. Сделать ЭТО МОЖНО посредством декоратора vary\_on\_headers() ИЗ модуля django.views, Decorators.vary:



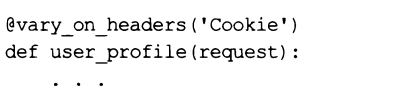
В качестве признаков, на основе значений которых в кэше будут создаваться отдельные редакции страницы, здесь указываются заголовки запроса, записанные в виде строк. Можно указать произвольное количество заголовков.

Вот как можно задать создание в кэше отдельных редакций одной и той же страницы для разных значений заголовка User-Agent (он обозначает программу веб-обозревателя, используемую клиентом):

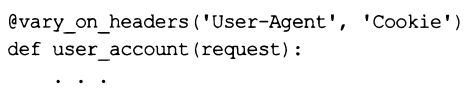
****

Теперь в кэше будет создаваться отдельная копия страницы на каждый веб-обозреватель, который ее запрашивал.

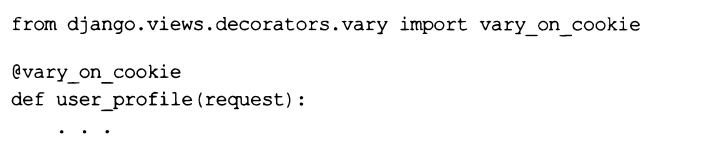
Чтобы создавать отдельные редакции страницы для гостя и пользователя, выполнившего вход, можно воспользоваться тем, что при создании сессии (в которой сохраняется ключ вошедшего на сайт пользователя) веб-обозревателю посылается Cookie с идентификатором этой сессии. А при отправке запроса на сервер все Cookie, сохраненные для домена, по которому отправляется запрос, посылаются серверу в заголовке cookie. В таком случае нужно написать код:

****

Пример указания в декораторе vary on headers () нескольких заголовков:



Поскольку создание в кэше разных редакций страницы для разных значений заголовка cookie является распространенной практикой, Django предоставляет декоратор vary\_on\_cookie () ИЗ ТОГО же модуля dj ango. Views. Decorators. Vary. Пример:

****

**Низкоуровневые средства кэширования**

Низкоуровневые средства кэширования применяются для сохранения в кэше сложных частей страниц и значений, получение которых сопряжено с интенсивными вычислениями.

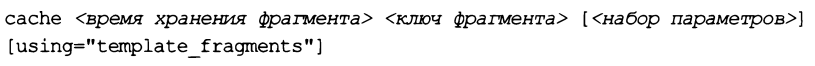
При кэшировании на низком уровне ключ, под которым сохраняется кэшируемая величина, указывается самим разработчиком. Этот ключ объединяется с префиксом и номером версии для получения конечного ключа., под которым значение и будет сохранено в кэше.

**Кэширование фрагментов веб-страниц**

Для кэширования фрагментов страниц понадобится библиотека тегов с псевдонимом cache:



Собственно кэширование выполняет парный тег cache . . . Endcache:



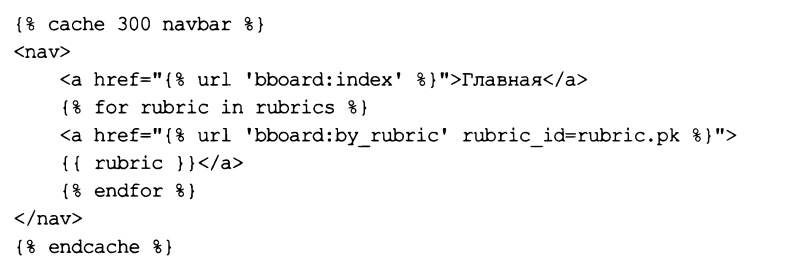
Время хранения фрагмента в кэше указывается в секундах. Если задать значение None, то фрагмент будет храниться вечно (пока не будет явно удален).

Ключ фрагмента необходим для формирования конечного ключа, под которым фрагмент будет сохранен в кэше, и должен быть уникальным в пределах всех шаблонов текущего проекта.

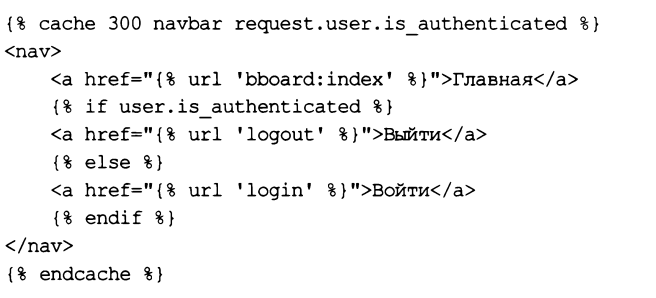
Набор параметров указывается в случаях, если требуется сохранять разные копии фрагмента для каждой комбинации значений заданных в наборе параметров.

Необязательный параметр using указывает псевдоним кэша. По умолчанию будет применяться кэш с псевдонимом template-fragments или, если таковой отсутствует, кэш по умолчанию.

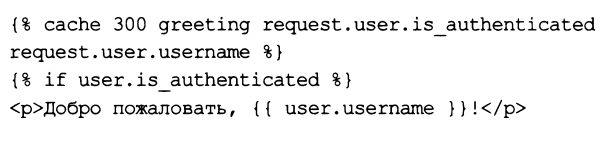
Вот так мы можем сохранить в кэше панель навигации нашего сайта на 300 секунд:

****

Если нам нужно хранить в кэше две копии фрагмента, одна из которых должна выдаваться гостям, а другая— пользователям, выполнившим вход, мы используем код такого вида:

****

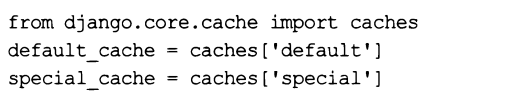
А если нужно хранить по отдельной копии фрагмента еще и для каждого из зарегистрированных пользователей сайта, мы напишем код:





**Кэширование произвольных значений**

Словарь со всеми кэшами, объявленными в настройках проекта, хранится в переменной caches из модуля django.core.cache. Ключи элементов этого словаря совпадают с псевдонимами кэшей, а значениями элементов являются сами кэши, представленные особыми объектами. Пример:

****

Если кэша с указанным псевдонимом не существует, то будет возбуждено исключение invalidcachebackenderror ИЗ модуля django.core.cache.backends.base.

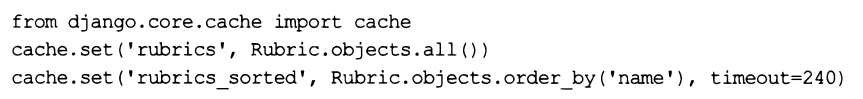
Переменная cache, также объявленная в модуле django.core.cache, хранит ссылку на объект кэша по умолчанию.

Любой объект кэша поддерживает следующие методы:

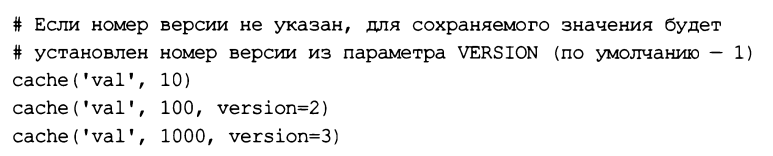
□ set(<jcn£)4>, <значение>[, timeout=<время хранения>][, version=None])—заносит в кэш значение под заданным ключом. Если заданный ключ уже существует в кэше, перезаписывает сохраненное под ним значение. Параметр timeout указывает время хранения значения в виде целого числа в секундах. Если он не задан, используется значение параметра timeout соответствующего кэша.

В необязательном параметре version можно задать версию, на основе которой будет формироваться конечный ключ для сохраняемого в кэше значения. Если версия не указана, будет использовано значение параметра version соответствующего кэша.

Примеры:

****

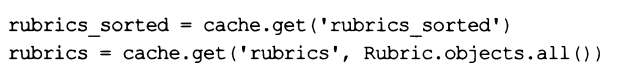
Имеется возможность сохранить произвольное количество значений под одним ключом, просто указав для этих значений разные номера версий:

****

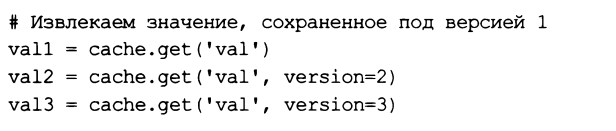
Л add(<ключ>, <значение>[, timeout=<время хранения>][, version=None]) —TO же самое, что set (), но сохраняет значение только в том случае, если указанный ключ не существует в кэше. В качестве результата возвращает True, если значение было сохранено в кэше, и False — в противном случае;

□ get (<ключ>[ f <значение по умолчанию>] [, version=None]) — извлекает ИЗ КЭШИ значение, ранее сохраненное под заданным ключом, и возвращает его в качестве результата. Если указанного ключа в кэше нет (например, значение еще не создано или уже удалено по причине устаревания), то возвращает значение по умолчанию или None, если оно не указано. Параметр version указывает номер версии.

Пример:



Сохранив ранее набор значений под одним ключом, мы можем извлекать эти значения, задавая для них номера версий, под которыми они были сохранены:

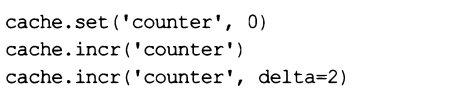
****

Л get\_or\_set (<ключ>, <значение>[, timeout=<время хранения>][, version=None]) — если указанный ключ существует в кэше, то извлекает хранящееся под ним значение и возвращает в качестве результата. В противном случае заносит в кэш значение под этим ключом и также возвращает это значение в качестве результата.

Параметр timeout задает время хранения, а параметр version — номер версии. Пример:

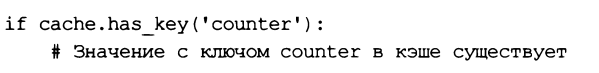


□ incr (<ключ>[, deita=i] [, version=None]) — увеличивает значение, хранящееся в кэше под заданным ключом, на величину, которая указана параметром delta (по умолчанию: 1). В качестве результата возвращает новое значение. Параметр Version задает номер версии. Пример:

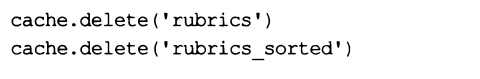


□ deer (<ключ>[, deita=i] [, version=None]) — уменьшает значение, хранящееся в кэше под заданным ключом, на величину, которая указана параметром delta (по умолчанию: 1). В качестве результата возвращает новое значение. Параметр Version задает номер версии;

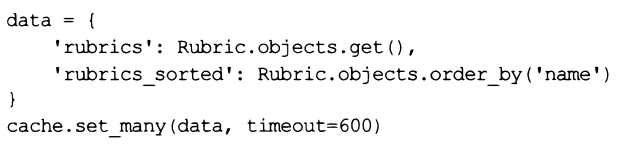
□ has\_key (<ключ> [, version=None] ) — возвращает True, если в кэше существует значение с указанным ключом, и False —в противном случае. Параметр version задает номер версии. Пример:

****

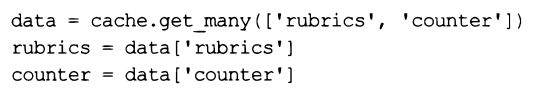
□ delete (<ключ>[, version=None]) — удаляет из кэша значение под указанным кКлючом. Параметр version задает номер версии. Пример:



О set\_many (<данные> [, timeout=<время хранения>] [, version=None]) —заносит В КЭШ заданные данные, представленные в виде словаря. Параметр timeout указывает время хранения данных, а параметр version —номер версии. Пример:

****

□ get\_many (<ключн> [, version=None]) — извлекает из кэша значения, ранее сохраненные под заданными ключами, и возвращает в виде словаря, ключи должны быть представлены в виде списка или кортежа. Параметр version указывает номер версии. Пример:

****

□ deiete\_many (<ключи>[, version=None]) — удаляет из кэша значения под указанными ключами, которые должны быть представлены в виде списка или кортежа.

Параметр version задает номер версии. Пример: Cache.delete\_many([’rubrics\_sorted’, ’counter’])

□ touch (<кпюч> [, timeout=<время хранения>]) — задает для значения с заданным ключом новое время хранения, указанное в параметре timeout. Если этот параметр опущен, то задается время кэширования из параметра timeout настроек кэша;

□ incr\_version(<ключ>[, delta=l] [, version=None]) — увеличивает версию значения, хранящегося в кэше под заданными ключом и версией, которая указана в параметре version, на величину, заданную в параметре delta. Новый номер версии возвращается в качестве результата;

О decr\_version (<ключ>[, delta=l] [, version=None]) — уменьшает версию значения, хранящегося в кэше под заданными ключом и версией, которая указана в параметре version, на величину, заданную в параметре delta. Новый номер версии возвращается в качестве результата;

□ clear () —полностью очищает кэш;

Внимание!

Вызов метода clear о выполняет полную очистку кэша, при которой из него удаляются абсолютно все данные.

□ close () — закрывает соединение с хранилищем, используемым в качестве кэша.

1. Использование Redis

Redis — другая популярная программа для кэширования, сохраняющая кэшируемые данные в файлах. Использовать ее для кэширования Django-сайтов позволяет дополнительная библиотека django-redis.

На заметку!

Документация по самой программе Redis находится по адресу https://redis.io/, а полная документация по библиотеке django-redis— по адресу https://niwinz.github.io/Django-redis/latest/.

**Установка django-redis и** **основные настройки кэша**

Библиотека устанавливается подачей команды:

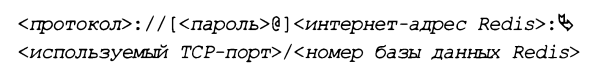


Помимо django-redis, будет установлена библиотека redis, необходимая для работы.

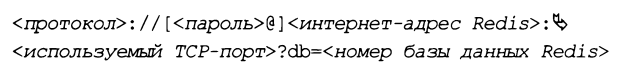
Далее в настройках кэша следует указать следующие параметры:

□ BACKEND — ИМЯ класса dj ango\_redis. Cache. Rediscache;

□ location —интернет-адрес для подключения к базе данных Redis формата:

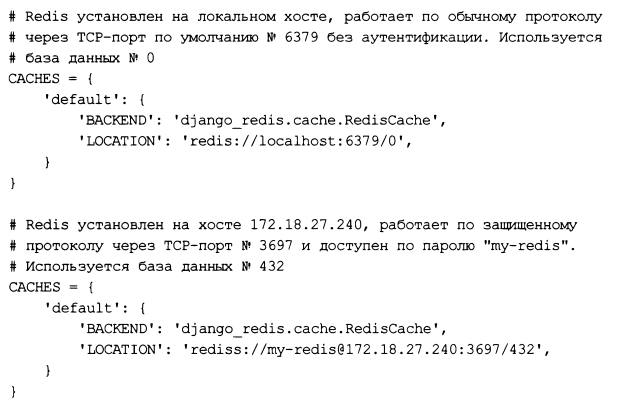


ИЛИ

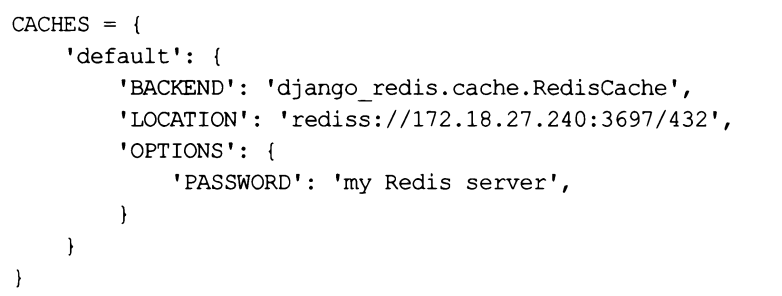


В качестве протокола МОЖНО указать redis (простой протокол) ИЛИ rediss (защищенный протокол), пароль задается только в случае, если для доступа к Redis используется аутентификация.

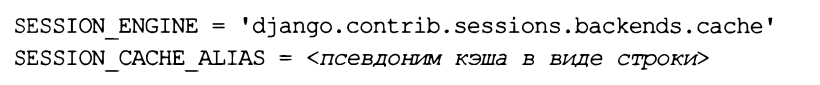
Примеры задания настроек:

****

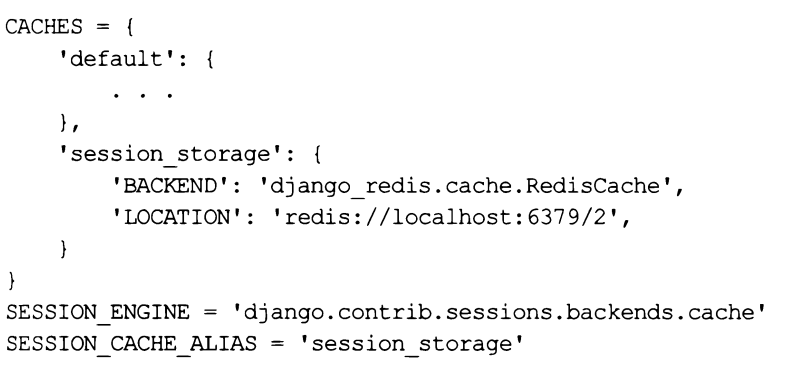
Если пароль не может быть указан непосредственно в интернет-адресе, поскольку содержит недопустимые символы, его можно записать в дополнительном параметре password:

****

Чтобы использовать кэш Redis для хранения сессий, необходимо указать непосредственно в настройках проекта (а не кэша!) Следующие параметры:

****

Пример:

****

Дополнительные инструменты кэширования, предоставляемые django-redis

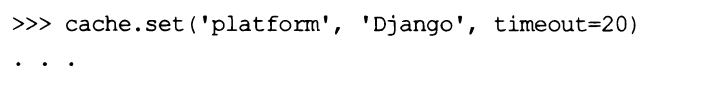
При занесении значения в кэш Redis вызовом метода set о можно сделать его "вечным", указав у параметра timeout значение None:

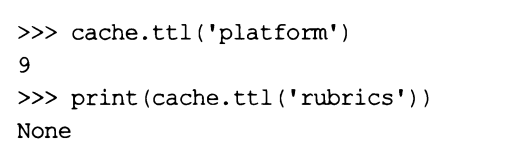


Такое значение никогда не устареет и может быть удалено только явно, вызовом метода delete (), delete\_many () ИЛИ clear ().

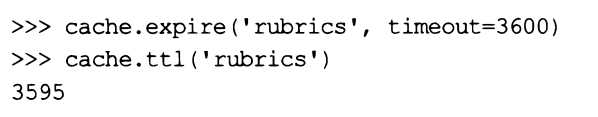
Объект кэша Redis поддерживает следующие дополнительные методы:

□ tti (<ключ> [, version=None]) — возвращает оставшееся время хранения (в секундах) кэшированного значения с заданным ключом. Если значение ’’вечное”, возвращается None. Параметр version задает номер версии. Примеры:

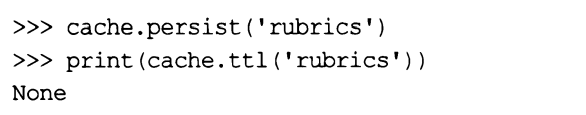




□ expire (<ключ>, timeout=<время хранения>[, version=None]) —указывает ДЛЯ значения с заданным ключом другое время хранения (в секундах). Параметр version задает номер версии. Пример:

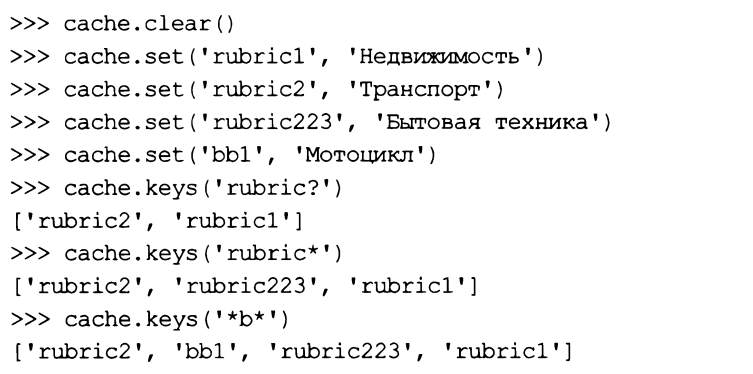
****

□ persist(<ключ>[, version=None]) — делает значение С заданным ключом ’’вечным”. Параметр version задает номер версии. Пример:

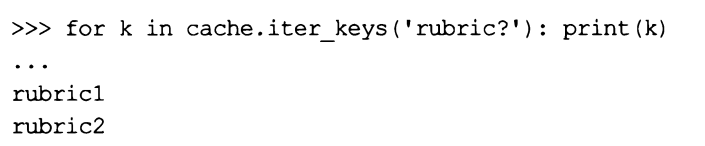
****

□ keys (<шаблон> [, version=None]) — ищет в кэше ключи, совпадающие с заданным шаблоном, и возвращает их в виде списка. В шаблоне можно использовать специальные символы \* (обозначает произвольное количество любых символов) и ?

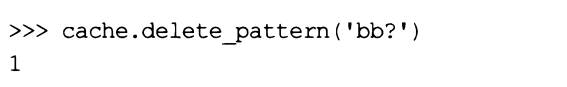
(обозначает один произвольный символ). Параметр version задает номер версии. Примеры:

****

□ iter keys (<шаблон> [, version=None]) — TO же самое, ЧТО И keys (), НО возвращает итератор, последовательно выдающий совпадающие с шаблоном ключи. Параметр Version задает номер версии. Пример:

****

□ delete\_pattem (<шаблон> [, prefix=None] [, version=None]) —удаляет значения с ключами, совпадающими с заданным шаблоном. Параметр prefix указывает префикс, который должен присутствовать в начале каждого ключа (если не задан, префикс не учитывается). Параметр version указывает номер версии. Пример:

****

□ lock () — блокирует значение с указанным ключом:

Lock(<ключ> [, version=None][, timeout=None][, sleep=0.1][,Blocking\_timeout=None])

Необязательные параметры:

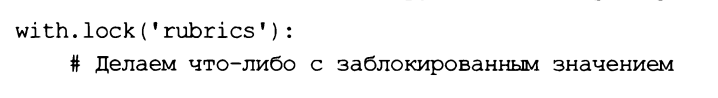
• version —номер версии;

• timeout — время существования блокировки (в секундах). Если None, то блокировка будет существовать до тех пор, пока не будет явно снята;

• sleep— время ожидания между последовательными попытками установить блокировку, если значение с указанным ключом уже заблокировано другим процессом (в секундах);

• blocking timeout— время, в течение которого библиотека будет пытаться заблокировать значение.

В качестве результата возвращается экземпляр класса Lock ИЗ МОДУЛЯ redis, представляющий наложенную на значение блокировку. Класс Lock поддерживает протокол обработчиков контента, вследствие чего его экземпляр может использоваться в языковой конструкции with. Пример:



Расширенные настройки django-redis следующие расширенные настройки записываются в дополнительных параметрах кэша (в словаре, присваиваемом параметру options настроек кэша):

□ compressor— имя модуля, выполняющего сжатие кэшируемых данных, в виде строки. Библиотека включает следующие модули сжатия:

• django\_redis. Compressors. Identity. Identitycompressor — вообще не сжимает данные (используется по умолчанию);

• django\_redis.compressors.zlib.zlibcompressor — Сжимает данные ПО алгоритму zlib;

• django\_redis. Compressors. Lzma. Lzmacompressor —сжимает ПО алгоритму lzma;

• django\_redis. Compressors. Iz4. Lz4Compressor —сжимает ПО алгоритму LZ4;

□ socket connect timeout— время, в течение которого Django будет пытаться установить соединение с сервером Redis, в виде целого числа в секундах. Если соединиться за указанное время с Redis не получится, будет возбуждено исключение connectionerгог ИЗ модуля redis. Exceptions;

□ socket timeout — время, в течение которого Django будет ожидать завершения операции чтения или записи значения из кэша, в виде целого числа в секундах.

Если прочитать или записать значение за заданное время не удастся, будет возбуждено исключение timeouterror ИЗ модуля redis.exceptions;

□ ignore exceptions— если True, то никакие исключения в случае неполадок с Redis возбуждаться не будут, если False —будут (по умолчанию — False).

Внимание!

Подавлять исключения указанием параметра ignore exceptions со значением True следует, только если Redis используется исключительно для кэширования. Если же он служит и для хранения сессий, то следует разрешить исключения — в противном случае сессии сохраняться не будут, и сайт не будет работать как положено.

Следующие параметры записываются непосредственно в настройках проекта:

□ django\_redis\_ignore\_exceptions— значение True включает подавление исключений, возникающих в случае неполадок, для всех Redis-кэшей, описанных в параметре caches. По умолчанию: False (подавлять исключения в кэше или не подавлять, определяет дополнительный параметр ignore exceptions);

□ django\_redis\_log\_ignored\_exceptions — если True, то исключения, возбужденные в случае неполадок с Redis, в любом случае будут записываться в журнал, даже если в настройках кэша указано подавлять их. Если False, то подавленные исключения записываться в журнал не будут. Принимается во внимание, только если включено подавление исключений. По умолчанию — False;

□ django redis logger— имя класса регистратора, выполняющего журналирование всех исключений в кэшах Redis, в виде строки. Если не указан, то будет использоваться регистратор, заданный в настройках журналирования.

1. Кэширование на стороне клиента

Для управления кэшированием на стороне клиента (в частности, для уменьшения объема пересылаемых по сети данных) Django предоставляет один посредник и несколько декораторов.

**Автоматическая обработка заголовков**

Посредник django.middleware.http.conditionalgetmiddleware применяется в случае кэширования на стороне сервера и выполняет автоматическую обработку заголовков, управляющих кэшированием на стороне клиента.

Как только клиенту отправляется запрошенная им страница, в состав ответа добавляется заголовок Е-Tag, хранящий хэш-сумму содержимого страницы. Эти сведения веб-обозреватель сохраняет в своем кэше.

Как только посетитель снова запрашивает загруженную ранее и сохраненную в локальном кэше страницу, веб-обозреватель посылает в составе запроса такие заголовки:

□ if-Match или if-None-Match со значением полученного с ответом заголовка Е-тад — если ранее с ответом пришел заголовок Е-тад;

□ If-Modified-Since ИЛИ If-Unmodified-Since С временной отметкой последнего изменения страницы, полученной с заголовком Last-Modified— если ранее в составе ответа был получен заголовок Last-Modified ИЛИ Last-Unmodified.

Получив запрос, посредник сравнивает значения:

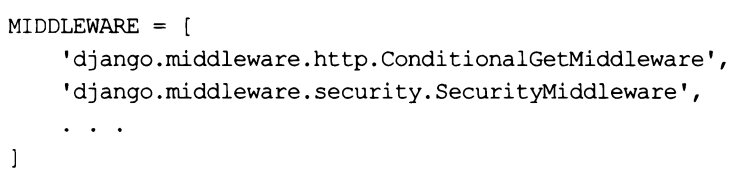
□ конечного ключа— полученное от клиента в заголовке if-Match или if-nonematch и находящееся в кэше;

□ временной отметки последнего изменения страницы — полученное от клиента в заголовке if-Modified-since или if-unmodified-since и хранящееся в кэше вместе с самой страницей.

Если эти значения равны, то Django предполагает, что кэшированная страница еще актуальна, и отправляет клиенту ответ с кодом статуса 304 (запрошенная страница не изменилась). Веб-обозреватель вместо того, чтобы повторно загружать страницу по сети, извлекает ее из локального кэша, что выполняется гораздо быстрее.

Если же эти значения не равны, значит, страница либо устарела и была удалена из кэша, либо еще не кэшировалась. Тогда веб-обозреватель получит полноценный ответ с кодом статуса 200, содержащий запрошенную страницу.

Посредник django.middleware.http.conditionalgetmiddleware В списке зарегистрированных в проекте (параметр middleware настроек сайта) должен помещаться перед посредником django.middleware.common.commonmiddleware. Обычно помещают его в самом начале списка зарегистрированных посредников:

****

Если кэширование на стороне сервера не задействовано, то посредник django.

Middleware.http.conditionalgetmiddleware не дает никакого эффекта. Хотя описанные ранее заголовки добавляются в ответ, получаются из запроса и обрабатываются посредником, но, поскольку страница не сохранена в кэше, она при каждом запросе генерируется заново.

**Условная обработка запросов**

Если кэширование на стороне сервера не используется, то для управления кэшированием на стороне клиента следует применять три декоратора из модуля Dj ango.views.decorators.http.

□ condition ([etag\_func=None] [,] [last\_modified\_func=None]) — выполняет обработку заголовков E-Tag И Last-Modified.

В параметре etag func указывается ссылка на функцию, которая будет вычислять значение заголовка Е-Tag. Эта функция должна принимать те же параметры, что и контроллер-функция, у которого указывается этот декоратор, и возвращать в качестве результата значение для упомянутого ранее заголовка, представленное в виде строки.

В параметре last modififed func указывается ссылка на аналогичную функцию, вычисляющую значение для заголовка Last-Modified. Эта функция должна возвращать значение заголовка в виде временной отметки (объекта типа datetime из модуля datetime).

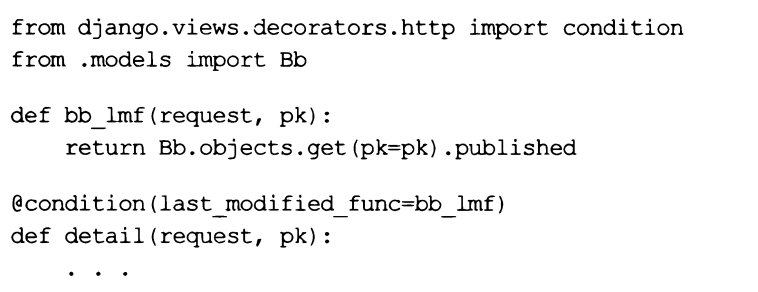
Можно указать либо оба параметра сразу, либо, как поступают наиболее часто, лишь один из них.

Декоратор указывается у контроллера-функции и в дальнейшем управляет кэшированием генерируемой им страницы.

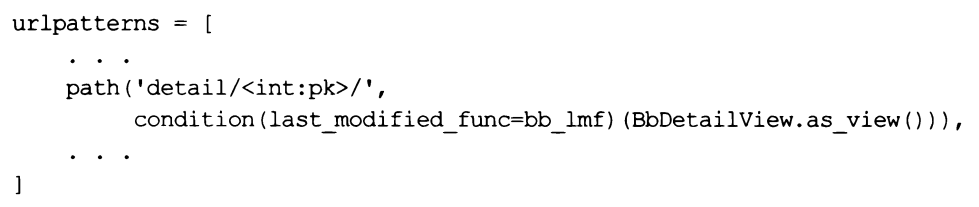
При формировании страницы функции, записанные в вызове декоратора, вычисляют значения заголовков Е-Tag и (или) Last-Modified. Эти заголовки отсылаются клиенту в составе ответа вместе с готовой страницей.

Как только от того же клиента будет получен запрос на ту же страницу, декоратор извлечет из запроса значения заголовков Е-Tag и (или) Last-Modified и сравнит их с величинами, возвращенными функциями, что указаны в его вызове.

Если значения Е-тад не совпадают или если значение Last-Modified, вычисленное функцией, больше значения, полученного в заголовке, то будет выполнен контроллер, который сгенерирует страницу. В противном случае клиенту отправится ответ с кодом статуса 304 (запрошенная страница не изменилась). Пример:

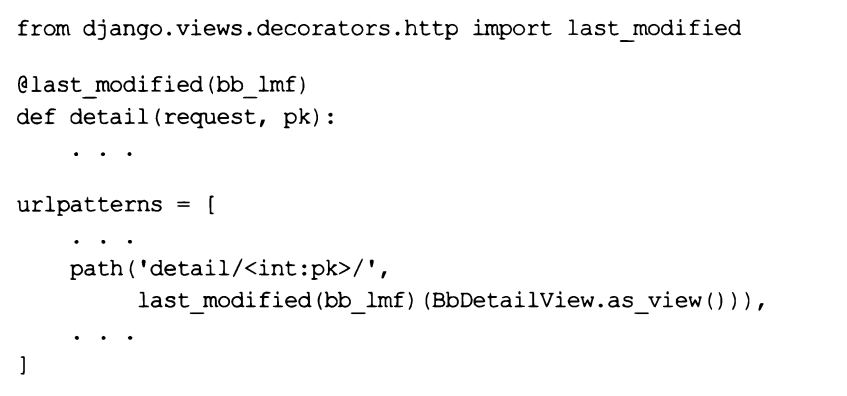
****

Декоратор condition () можно указать и у контроллера-класса:

****

□ etag (<функция E-Tag>) —обрабатывает ТОЛЬКО Заголовок E-Tag;

□ last\_modified (<функция Last-Modified>} —обрабатывает ТОЛЬКО Заголовок Last\_Modified. Пример:

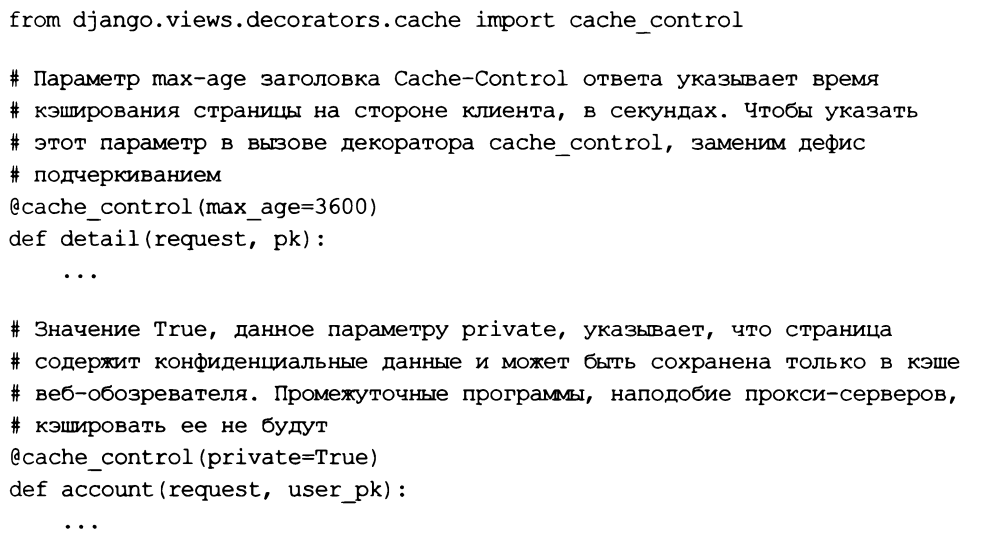
****

**Прямое указание параметров кэширования**

Параметры кэширования страницы на стороне клиента записываются в составе значения заголовка Cache-Control, который отсылается клиенту в составе ответа, включающего эту страницу. Указать эти параметры напрямую можно, воспользовавшись декоратором cache\_control (<параметры>) ИЗ модуля django.views.

Decorators. Cache. В его вызове указываются именованные параметры, которые и зададут настройки кэширования.

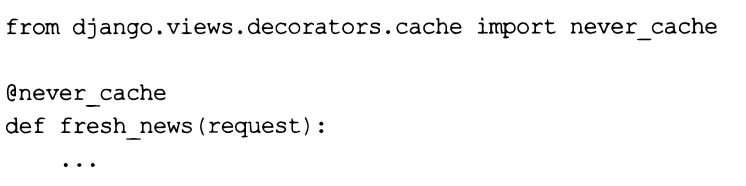
Чтобы вставить в заголовок Cache-Control ответа параметр, не имеющий значения, следует присвоить соответствующему именованному параметру декоратора значение True. Если параметр заголовка включает дефис, то в именованном параметре его следует заменить подчеркиванием. Примеры:

****

**Запрет кэширования**

Если необходимо запретить кэширование какой-либо страницы на уровне клиента (например, если страница содержит часто обновляющиеся или конфиденциальные данные), достаточно использовать декоратор never\_cache() из модуля django.

Views. Decorators. Cache. Пример:

****

Этот декоратор добавляет в отсылаемый клиенту ответ следующий заголовок:

****

Параметр private добавляется, начиная c Django 3.0.

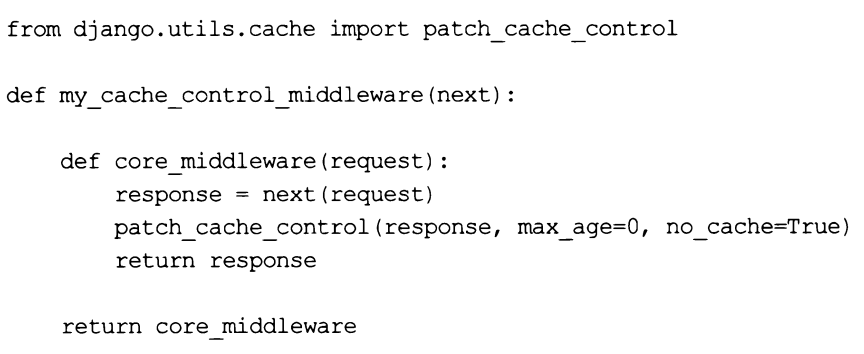
**Управление кэшированием в посредниках**

Для указания параметров кэширования на стороне клиента в посредниках (о них рассказывалось ранее) применяются следующие функции, объявленные в модуле dj ango.utils.cache:

О patch\_cache\_control (<ответ>, <параметры кэширования^ — непосредственно указывает параметры кэширования в заголовке Cache-Control заданного ответа, параметры кэширования указываются в соответствующих именованных параметрах функции.

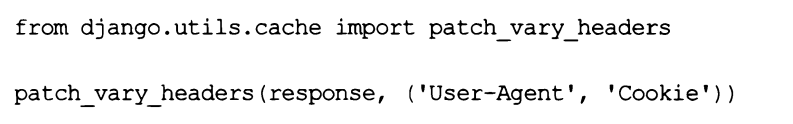
Чтобы вставить в заголовок Cache-Control ответа параметр, не имеющий значения, следует присвоить соответствующему именованному параметру функции значение True. Если параметр кэширования включает дефис, то в именованном параметре его следует заменить подчеркиванием.

Пример посредника, добавляющего в ответ заголовок Cache-Control: max-age=o, No-cache:

****

□ patch\_response\_headers (<ответ> [, cache\_timeout=None]) — задает время кэширования страницы клиентом в заголовке Expires и параметре max-age заголовка Cache-Control заданного ответа. Время кэширования указывается в параметре Cache timeout; если оно не задано, то устанавливается время кэширования из параметра cache\_middleware\_seconds настроек проекта;

□ patch\_vary\_headers (<ответ>, <заголовки>) — добавляет в заголовок Vary ответа заданные во втором параметре заголовки запроса. Последние указываются в виде списка или кортежа из строк. Пример:

****

□ add never cache headers (<ответ>) — добавляет В ответ заголовок, запрещающий кэширование страниц на стороне клиента:



Параметр private добавляется, начиная c Django 3.0;

□ get\_max\_age (<ответ>) — возвращает время кэширования страницы клиентом (в секундах), взятое из параметра max-age заголовка Cache-Control заданного ответа. Если такой параметр отсутствует или его значение не является целым числом, ТО возвращается None.