**Тема занятия № 41: Модуль 23. Cookie, сессии, всплывающие сообщения и подписывание данных**

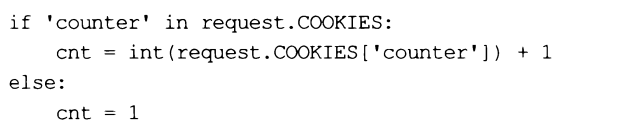
1. Cookie

Django поддерживает развитые средства для обработки cookie, хранения данных в сессиях, вывода всплывающих сообщений и защиты данных цифровой подписью.

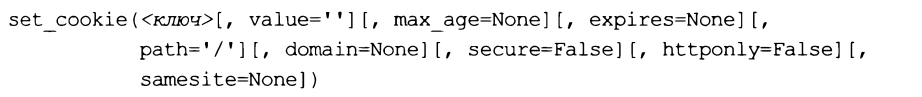
Cookie — небольшой, не более 4 Кбайт, фрагмент произвольных данных, сохраняемый на компьютере клиента. Обычно применяется для хранения служебных данных, настроек сайта и пр.

Все cookie, сохраненные на стороне клиента, относящиеся к текущему домену и еще не просроченные, доступны через атрибут cookies объекта запроса (экземпляра класса httprequest). Ключами элементов этого словаря выступают ключи всех доступных cookie, а значениями элементов— значения, сохраненные в этих cookie и представленные в виде строк. Значения cookie доступны только для чтения.

Вот пример извлечения из cookie текущего значения счетчика посещений страницы и увеличения его на единицу:

****

Для записи значения в cookie применяется метод set cookie () класса httpresponse, представляющего ответ. Вот формат вызова этого метода:

****

Ключ записываемого значения указывается в виде строки. Само значение задается в параметре value; если он не указан, то будет записана пустая строка.

Параметр max\_age указывает время хранения cookie на стороне клиента в секундах.

Параметр expires задает дату и время, после которых cookie станет недействительным и будет удален, в виде объекта типа datetime из модуля datetime Python. В вызове метода следует указать один из этих параметров, но не оба сразу. Если ни один из этих параметров не указан, то cookie будет храниться лишь до тех пор, пока посетитель не уйдет с сайта.

Параметр path указывает путь, к которому будет относиться cookie, — в таком случае при запросе с другого пути сохраненное в cookie значение получить не удастся.

Например, если задать значение "/testapp/”, cookie будет доступен только в контроллерах приложения testapp. Значение по умолчанию: "/" (путь к корневой папке), — в результате чего cookie станет доступным с любого пути.

Параметр domain указывает корневой домен, откуда должен быть доступен сохраняемый cookie, и применяется, если нужно создать cookie, доступный с другого домена. Так, если указать значение "site.ru”, то cookie будет доступен с доменов Www.site.ru,support.site.ru, shop.site.ru и др. Если параметр не указан, cookie будет доступен только в текущем домене.

Если параметру secure дать значение True, то cookie будет доступен только при обращении к сайту по защищенному протоколу. Если параметр не указан (или если ему дано значение False), cookie будет доступен при обращении по любому протоколу.

Если параметру httponiy дать значение True, cookie будет доступен только серверу.

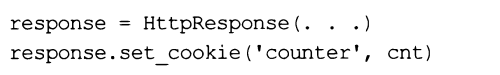
Если параметр не указан (или если ему дано значение False), cookie также будет доступен в клиентских веб-сценариях, написанных на javascript. параметр samesite разрешает или запрещает отправку сохраненного cookie при выполнении запросов на другие сайты. Доступны три значения:

□ None —разрешает отправку cookie (поведение по умолчанию);

□ "Lax" —разрешает отправку cookie только при переходе на другие сайты по гиперссылкам;

□ "strict" —полностью запрещает отправку cookie другим сайтам.

Вот пример записи в cookie значения счетчика посещений страницы, полученного ранее:



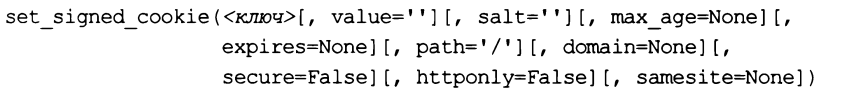
Удалить cookie МОЖНО ВЫЗОВОМ метода delete cookie () класса httpresponse:



Значения параметров path и domain должны быть теми же, что использовались в вызове метода set cookie (), создавшего удаляемый cookie. Если cookie с заданным ключом не найден, метод ничего не делает.

Django также поддерживает создание и чтение подписанных cookie, в которых сохраненное значение дополнительно защищено цифровой подписью.

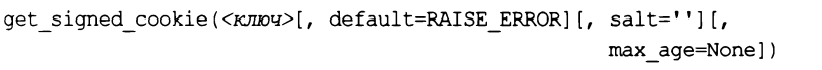
Сохранение значения в подписанном cookie выполняется вызовом метода Set\_signed\_cookie () класса httpresponse:

****

Здесь указываются те же самые параметры, что и у метода set cookie (). Дополнительный параметр salt задает соль — особое значение, участвующее в генерировании цифровой подписи и служащее для повышения ее стойкости.

Если в параметре max\_age или expires задано время существования подписанного Cookie, то сгенерированная цифровая подпись будет действительна в течение указанного времени.

Прочитать значение из подписанного cookie и удостовериться, что оно не скомпрометировано, позволяет метод get\_signed\_cookie () класса httprequest:

****

Значение соли, заданное в параметре salt, должно быть тем же, что использовалось в вызове метода set signed cookie (), создавшего этот cookie.

Если цифровая подпись у сохраненного значения не была скомпрометирована, то метод вернет сохраненное значение. В противном случае будет возвращено значение, заданное в необязательном параметре default. То же самое случится, если Cookie с заданным ключом не был найден.

Если в качестве значения параметра default указать значение переменной RAISE ERROR ИЗ МОДУЛЯ django.http.request, ТО будет Возбуждено ОДНО ИЗ Двух исключений: badsignature ИЗ модуля django.core.signing, если цифровая ПОДПИСЬ скомпрометирована, или keyerror, если cookie с заданным ключом не найден.

Если в необязательном параметре max age указано время существования подписанного cookie, то дополнительно будет выполнена проверка, не устарела ли цифровая подпись. Если цифровая подпись устарела, метод возбудит исключение Signatureexpired ИЗ модуля django.core. Signing.

Удалить подписанный cookie можно так же, как и cookie обычный — вызовом метода delete cookie () объекта ответа.

1. Сессии

Сессия — это промежуток времени, в течение которого посетитель пребывает на текущем сайте. Сессия начинается, как только посетитель заходит на сайт, и завершается после его ухода.

К сессии можно привязать произвольные данные и сохранить их в каком-либо хранилище (базе данных, файле и др.) На стороне. Эти данные будут храниться во время существования сессии и останутся в течение определенного времени после ее завершения, пока не истечет указанный промежуток времени и сессия не перестанет быть актуальной. Такие данные тоже называют сессией.

Для каждой сессии Django генерирует уникальный идентификатор, который затем сохраняется в подписанном cookie на стороне клиента (cookie сессии). Поскольку содержимое всех cookie, сохраненных для того или иного домена, автоматически отсылается серверу в составе заголовка каждого запроса, Django впоследствии без проблем получит сохраненный на стороне клиента идентификатор и по нему найдет данные, записанные в соответствующей сессии.

Мы можем сохранить в сессии любые данные, какие нам нужны. В частности, подсистема разграничения доступа хранит в таких сессиях ключ пользователя, который выполнил вход на сайт.

Поскольку данные сессии сохраняются на стороне сервера, в них можно хранить конфиденциальные сведения, которые не должны быть доступны никому.

**Настройка сессий**

Чтобы успешно работать с сессиями, предварительно следует:

□ проверить, присутствует ли приложение django.contrib.sessions в списке зарегистрированных в проекте (параметр installed\_apps);

□ проверить, присутствует ЛИ посредник dj ango. Contrib. Sessions .middleware.

Sessionmiddleware в списке зарегистрированных в проекте (параметр middleware).

Впрочем, и приложение, и посредник присутствуют в списках изначально, поскольку активно используются другими стандартными приложениями Django.

Параметры, влияющие на работу подсистемы сессий, как обычно, указываются в настройках проекта — в модуле settings.py пакета конфигурации:

□ session engine— имя класса, реализующего хранилище для сессий, в виде строки. Можно указать следующие классы:

• django.contrib.sessions.backends.db — хранит сессии в базе данных. Имеет среднюю производительность, но гарантирует максимальную надежность хранения данных;

• django. Contrib. Sessions .backends. File — хранит сессии В обычных файлах.

По сравнению с предыдущим хранилищем имеет пониженную производительность, но создает меньшую нагрузку на базу данных;

• django.contrib.sessions.backends.cache — хранит сессии В кэше стороны сервера. Обеспечивает высокую производительность, но требует наличия активной подсистемы кэширования;

• django. Contrib. Sessions.backends. Cached\_db — хранит сессии В кэше стороны сервера, одновременно дублируя их в базе данных для надежности. По сравнению с предыдущим хранилищем обеспечивает повышенную надежность, но увеличивает нагрузку на базу данных;

• django.contrib.sessions.backends.signed\_cookies — Хранит сессии непосредственно в cookie сессии. Обеспечивает максимальную производительность, но для каждой сессии позволяет сохранить не более 4 Кбайт данных.

Значение ПО умолчанию: "django. Contrib. Sessions.backends.db”;

□ session serializer— имя класса сериализатора, который будет использоваться для сериализации сохраняемых в сессиях данных, указанное в виде строки.

В составе Django поставляются два сериализатора:

• django.contrib.sessions.serializers.jsonserializer — сериализует данные в формат JSON. Может обрабатывать только элементарные типы Python;

• django.contrib.sessions.serializers.pickleserializer — сериализует средствами модуля pickle. Способен обработать значение любого типа.

Значение ПО умолчанию: "django.contrib.sessions.serializers. Jsonserializer";

□ session\_expire\_at\_browser\_close — если True, то сессии co всеми сохраненными в них данными будут автоматически удаляться, как только посетитель закроет веб-обозреватель, если False, то сессии будут сохраняться. По умолчанию — False;

□ SESSION\_SAVE\_EVERY\_REQUEST--- если True, ТО сессии будут сохраняться в хранилище при обработке каждого запроса, если False — только при изменении записанных в них данных. По умолчанию — False;

□ session cookie domain — домен, к которому будут относиться cookie сессий. По умолчанию — None (т. Е. Текущий домен);

□ session cookie path— путь, к которому будут относиться cookie сессий (по умолчанию: "/");

□ session cookie age — время существования cookie сессий, в виде целого числа в секундах. По умолчанию: 1209600 (2 недели);

□ session cookie name — ключ, под которым в cookie будет сохранен идентификатор сессии (по умолчанию: "sessionid");

□ session cookie httponly — если True, то cookie сессий будут доступны только серверу. Если False, то cookie сессий также будут доступны клиентским веб-сценариям. По умолчанию — True;

□ session cookie secure — если True, то cookie сессий будут доступны только при обращении к сайту по защищенному протоколу HTTPS, если False — при обращении ПО любому Протоколу. По умолчанию--- False;

□ session cookie samesite — признак, разрешающий или запрещающий отправку Cookie сессий при переходе на другие сайты. Доступны три значения:

• None — разрешает отправку cookie сессий (поведение по умолчанию);

• "Lax" — разрешает отправку cookie сессий только при переходе на другие сайты по гиперссылкам;

• "strict" — полностью запрещает отправку cookie сессий другим сайтам;

□ session file path— полный путь к папке, в которой будут храниться файлы с сессиями. Если указано значение None, то Django использует системную папку для хранения временных файлов. По умолчанию — None.

Этот параметр принимается во внимание только в том случае, если для хранения сессий были выбраны обычные файлы;

□ session cache alias— название кэша, в котором будут храниться сессии. По умолчанию: "default” (кэш по умолчанию).

Этот параметр принимается во внимание только в том случае, если для хранения сессий был выбран кэш стороны сервера без дублирования в базе данных или же с таковым.

Если в качестве хранилища сессий были выбраны база данных или кэш стороны сервера с дублированием в базе данных, то перед использованием сессий следует выполнить миграции.

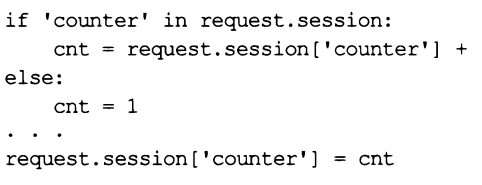
На заметку!

Если для хранения сессий были выбраны база данных или кэш стороны сервера с дублированием в базе данных, то в базе данных будет создана таблица django\_Session.

**Использование сессий**

Посредник django. Contrib.sessions.middleware. Sessionmiddleware добавляет объекту запроса атрибут sessions. Он хранит объект, поддерживающий функциональность словаря и содержащий все значения, которые были сохранены в текущей сессии.

Вот пример реализации счетчика посещений страницы, хранящего текущее значение в сессии:



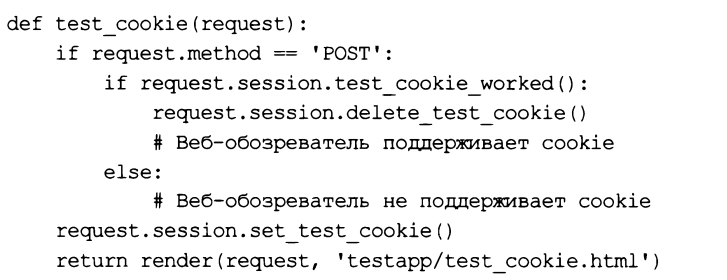
Помимо этого, объект, хранящийся в атрибуте sessions объекта запроса, поддерживает следующие методы:

□ flush () — удаляет все данные, сохраненные в текущей сессии, наряду с cookie сессии (метод clear (), поддерживаемый тем же объектом, равно как и словарями Python, не удаляет cookie);

□ set test cookie () — создает тестовый cookie, позволяющий удостовериться, что веб-обозреватель клиента поддерживает cookie;

□ test cookie worked () — возвращает True, если веб-обозреватель клиента поддерживает cookie, и False — в противном случае. Проверка, поддерживает ли веб-обозреватель cookie, запускается вызовом метода set test cookie ();

□ delete test cookie () — удаляет созданный ранее тестовый cookie. вот пример использования трех из описанных выше методов:

****

□ set\_expiry( <время>) — задает время устаревания текущей сессии, по достижении которого сессия будет удалена. В качестве значения времени можно указать:

• целое число — задаст количество секунд, в течение которых сессия будет актуальна;

• объект типа datetime ИЛИ timedelta ИЗ модуля datetime — укажет временную отметку устаревания сессии. Поддерживается только при использовании сериализатора django.contrib.sessions.serializers. Jsonserializer;

• o — сессия перестанет быть актуальной и будет удалена, как только посетитель закроет веб-обозреватель;

• None— будет использовано значение из параметра session cookie age настроек проекта;

О get\_expiry\_age ([modification=datetime.datetime. Today () ] [, ] [expiry=None]) — возвращает время, в течение которого текущая сессия еще будет актуальной, в секундах. Необязательный параметр modification указывает временную отметку последнего изменения сессии (по умолчанию — текущие дата и время), а параметр expiry— время ее устаревания в виде временной отметки, количества секунд или None (в этом случае будет использовано время устаревания, заданное вызовом метода set expiryo или, если этот метод не вызывался, взятое из параметра session\_cookie\_age);

□ get\_expiry\_date ([modification=datetime.datetime, today () ] [, ] [expiry=None]) — возвращает временную отметку устаревания текущей сессии. Необязательный параметр modification указывает временную отметку последнего изменения сессии (по умолчанию — текущие дата и время), а параметр expiry— время ее устаревания в виде временной отметки, количества секунд или None (в этом случае будет использовано время устаревания, заданное вызовом метода Set expiryo или, если этот метод не вызывался, взятое из параметра session\_Cookie\_age);

□ get expire at browser ciose () — возвращает True, если текущая сессия устареет и будет удалена, как только посетитель закроет веб-обозреватель, и False — в противном случае;

□ ciear expired () — удаляет устаревшие сессии;

□ cycie keyo — создает новый идентификатор для текущей сессии без потери сохраненных в ней данных.

Для удаления всех устаревших сессий, которые по какой-то причине не были удалены автоматически, достаточно применить команду clearsessions утилиты Manage.py. Формат ее вызова очень прост: Manage.py clearsessions

1. Всплывающие сообщения

Всплывающие сообщения существуют только во время выполнения текущего запроса. Они применяются для вывода на страницу какого-либо сообщения (например, об успешном добавлении новой записи), актуального только в данный момент.

**Настройка всплывающих сообщений**

Перед использованием подсистемы всплывающих сообщений необходимо:

□ проверить, присутствует ли приложение django.contrib.messages в списке зарегистрированных в проекте (параметр installed apps);

□ проверить, присутствуют ЛИ посредники django.contrib.sessions.middleware.

Sessionmiddleware И dj ango. Contrib.messages,middleware. Messagemiddleware В списке зарегистрированных в проекте (параметр middleware);

□ проверить, присутствует ЛИ обработчик контекста django.contrib.messages.

Context\_processors.messages в списке зарегистрированных для используемого шаблонизатора.

Впрочем, и приложение, и посредники, и обработчик контекста присутствуют в списках изначально.

Немногочисленные настройки, управляющие работой подсистемы всплывающих сообщений, записываются в модуле settings.py пакета конфигурации:

□ message storage— имя класса, реализующего хранилище всплывающих сообщений, представленное в виде строки. В составе Django поставляются три хранилища всплывающих сообщений:

• django.contrib.messages.storage.cookie.cookiestorage — использует cookie;

• django.contrib.messages.storage.session.Sessionstorage — использует сессии;

• django.contrib.messages.storage.fallback.Fallbackstorage — использует Cookie для хранения сообщений, чей объем не превышает 4 Кбайт, а более объемные сохраняет в сессии.

Значение по умолчанию:



□ message level— минимальный уровень всплывающих сообщений, которые будут выводиться подсистемой. Указывается в виде целого числа. По умолчанию: 20;

□ message tags — соответствия между уровнями сообщений и стилевыми классами. Более подробно это будет рассмотрено позже.

**Уровни всплывающих сообщений**

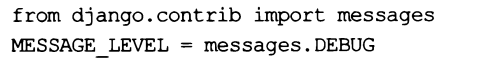
Каждое всплывающее сообщение Django, помимо текстового содержимого, имеет так называемый уровень, указывающий его ранг и выражаемый целым числом.

Каждому такому уровню соответствует свой стилевой класс, привязываемый к HTML-тегу, в котором выводится текст сообщения.

Изначально в Django объявлено пять уровней всплывающих сообщений, каждый из которых имеет строго определенную область применения. Значение каждого из этих уровней занесено в отдельную переменную, и эти переменные, объявленные в модуле django. Contrib. Messages, можно использовать для указания уровней сообщений при их выводе.

Параметр message level настроек проекта указывает минимальный уровень сообщений, которые будут выводиться на страницы. Если уровень создаваемого сообщения меньше указанной в нем величины, то сообщение не будет выведено.

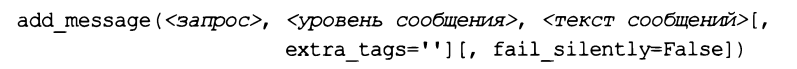
По умолчанию этот параметр имеет значение 2 о (переменная info), следовательно, сообщения с меньшим уровнем, в частности отладочные (debug), обрабатываться не будут. Если нужно сделать так, чтобы отладочные сообщения также выводились на экран, необходимо задать для этого параметра подходящее значение:



**Создание всплывающих сообщений**

Создать всплывающее сообщение для его последующего вывода можно вызовом описанных далее функций ИЗ модуля django. Contrib.messages.

В первую очередь это ’’универсальная” функция add message о, создающая сообщение произвольного уровня. Вот формат ее вызова:

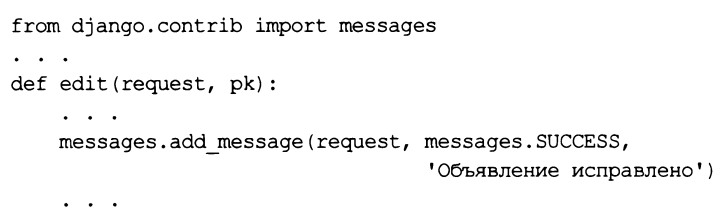


Запрос представляется экземпляром класса httprequest, уровень сообщения—целым числом, а его текст —строкой.

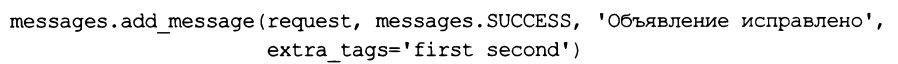
Необязательный параметр extra tags указывает перечень дополнительных стилевых классов, привязываемых к HTML-тегу, в котором будет выведен текст сообщения. Перечень стилевых классов должен представлять собой строку, а стилевые классы в нем должны отделяться друг от друга пробелами.

Если задать необязательному параметру faii silentiy значение True, то в случае невозможности создания нового всплывающего сообщения (например, если соответствующая подсистема отключена) ничего не произойдет. Используемое по умолчанию значение False указывает в таком случае возбудить исключение Messagefailure ИЗ ТОГО же модуля django.contrib.messages.

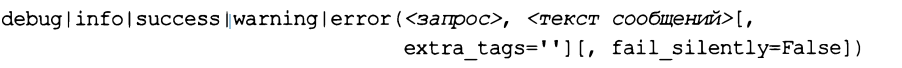
Пример создания нового всплывающего сообщения:

****

Пример создания всплывающего сообщения с добавлением дополнительных стилевых классов first И second:



Функции debug (), info (), success (), warning () и error () создают сообщение соответствующего уровня. Все они имеют одинаковый формат вызова:

****

Значения параметров здесь задаются в том же формате, что и у функции Add\_message().

Пример:



"Научить" создавать всплывающие сообщения высокоуровневые контроллеры-классы можно, унаследовав их от класса-примеси successmessagemixin из модуля Django.contrib.messages.views. Этот класс поддерживает следующие атрибут и метод:

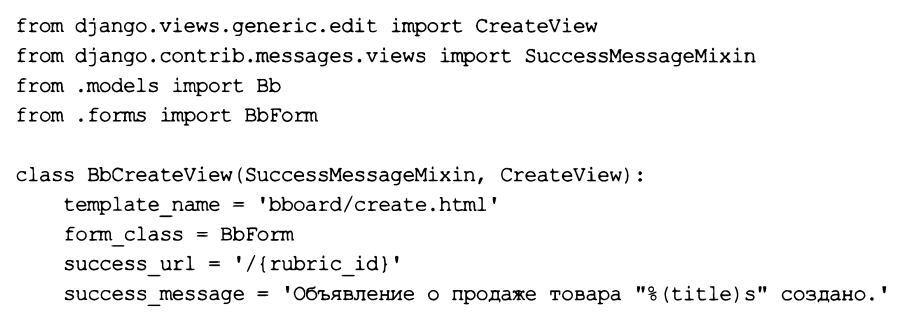
□ success message — текст сообщения об успешном выполнении операции в виде строки. В строке допускается применять специальные символы вида % (<имя поля формы>) s, вместо которых будут подставлены значения соответствующих полей;

□ get\_success\_message(self, cleaded data) — должен возвращать ПОЛНОСТЬЮ сформированный текст всплывающего сообщения об успешном выполнении

Операции. Словарь с данными формы из параметра cleaned\_data содержит полностью готовые к использованию значения полей формы.

В изначальной реализации возвращает результат форматирования строки из атрибута success message с применением полученного словаря с данными формы.

Приведен код контроллера, создающего новое объявление, который в случае успешного его создания отправляет посетителю всплывающее сообщение.

****

**Вывод всплывающих сообщений**

Вывести всплывающие сообщения в шаблоне удобнее всего посредством обработчика контекста django.contrib.messages.context\_processors.messages. Он добавляет в контекст шаблона переменную messages, которая хранит последовательность всех всплывающих сообщений, созданных к настоящему времени в текущем запросе.

Каждый элемент этой последовательности представляет собой экземпляр класса Message. Все необходимые сведения о сообщении хранятся в его атрибутах:

□ message — текст всплывающего сообщения;

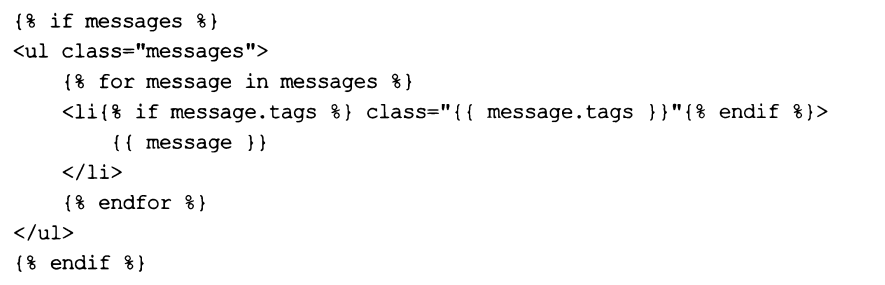
□ level —уровень всплывающего сообщения в виде целого числа;

□ level tag — имя основного стилевого класса, соответствующего уровню сообщения;

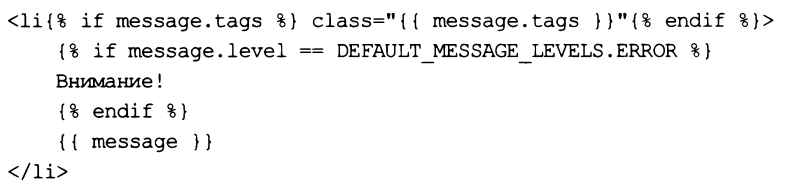
□ extra tags— строка с дополнительными стилевыми классами, указанными в параметре extra tags при создании сообщения;

□ tags —строка со всеми стилевыми классами (и основным, и дополнительными), записанными через пробелы.

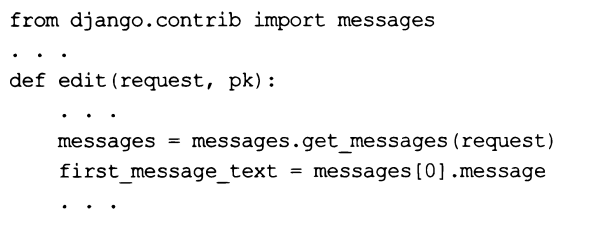
Вот пример кода шаблона, выполняющего вывод всплывающих сообщений:

****

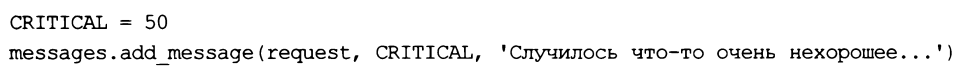
Еще обработчик контекста django.contrib.messages.context\_processors.messages добавляет в контекст шаблона переменную default message levels. Она хранит словарь, в качестве ключей элементов которого выступают строковые названия уровней сообщений, а значений элементов— соответствующие им числа. Этот словарь можно использовать в операциях сравнения, подобных этой:

****

Если же нужно получить доступ к сообщениям в контроллере, то можно воспользоваться функцией get\_messages (<запрос>) ИЗ модуля django.contrib.messages. Запрос представляется экземпляром класса httprequest, а результатом будет список сообщений в виде экземпляров класса Message. Пример:

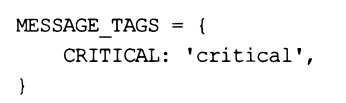
** Объявление своих уровней всплывающих сообщений**

Никто и ничто не мешает нам при создании всплывающего сообщения указать произвольный уровень:

****

Нужно только проследить за тем, чтобы выбранное значение уровня не совпало с каким-либо из объявленных в самом Django.

Если мы хотим, чтобы при выводе всплывающих сообщений на экран для объявленного нами уровня устанавливался какой-либо стилевой класс, то должны выполнить дополнительные действия. Мы объявим словарь, добавим в него элемент, соответствующий объявленному нами уровню сообщений, установим в качестве ключа элемента значение уровня, а в качестве значения элемента — строку с именем стилевого класса, после чего присвоим этот словарь параметру message tags настроек проекта. Вот пример:

****

1. Подписывание данных

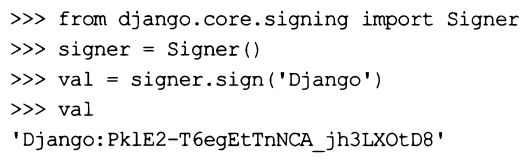
Для подписывания строковых значений обычной цифровой подписью применяется класс signer из модуля django.core.signing. Конструктор этого класса вызывается в формате:



Параметр key указывает секретный ключ, на основе которого будет генерироваться цифровая подпись (по умолчанию используется секретный ключ из параметра Secret key настроек проекта). Параметр sep задает символ, которым будут отделяться друг от друга подписанное значение и сама подпись (по умолчанию — двоеточие). Наконец, параметр salt указывает соль (если он опущен, соль задействована не будет).

Класс signer поддерживает два метода:

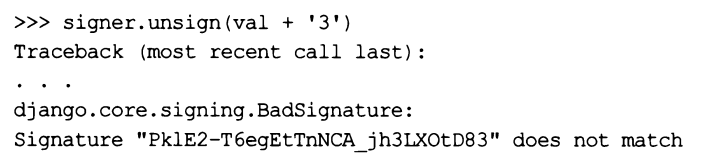
□ sign (Подписываемое значение^ — подписывает полученное значение И возвращает результат:

****

О unsign (Подписанное значение^ —ИЗ полученного подписанного значения извлекает оригинальную величину, которую и возвращает в качестве результата:

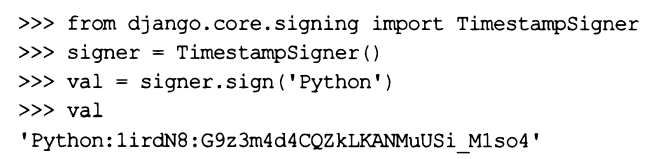
****

Если подписанное значение скомпрометировано (не соответствует цифровой подписи), ТО возбуждается исключение badsignature ИЗ модуля django.core, Signing:

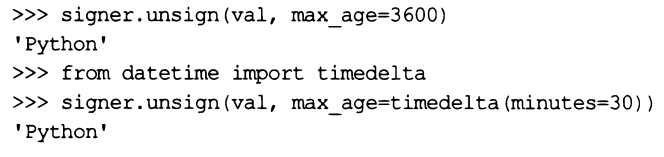
****

Класс timestampsigner ИЗ ТОГО же модуля django. Core, signing подписывает значение цифровой подписью с ограниченным сроком действия. Формат вызова его конструктора такой же, как у конструктора класса signer. класс Timestampsigner поддерживает два метода:

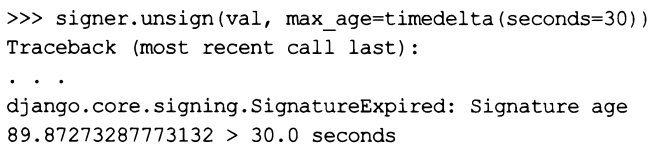
□ sign (^подписываемое значение^ — подписывает полученное значение И возвращает результат:

****

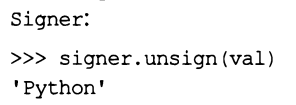
□ unsign ^подписанное значение>[, max\_age=None])—ИЗ полученного подписанного значения извлекает оригинальную величину, которую и возвращает в качестве результата. Параметр max age задает промежуток времени, в течение которого актуальна цифровая подпись, в виде целого числа, в секундах, или в виде объекта типа timedelta из модуля datetime. Если подписанное значение скомпрометировано, ТО возбуждается исключение badsignature ИЗ модуля django.core, Signing. Примеры:

****

Если цифровая подпись уже не актуальна, возбуждается исключение Signatureexpired ИЗ модуля django.core. Signing:

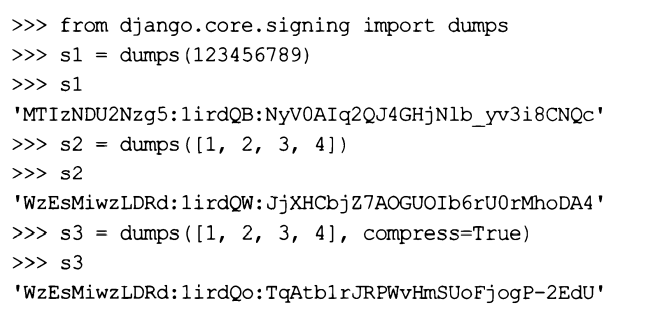
****

Если параметр max age не указан, то проверка на актуальность цифровой подписи не проводится, и метод unsign о работает так же, как его ’’тезка” у класса Signer:

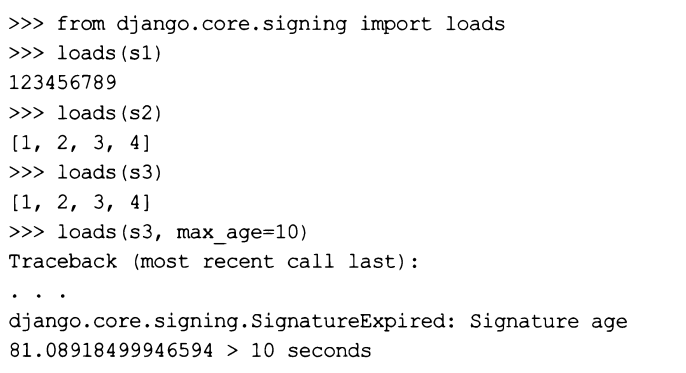
****

Если нужно подписать значение, отличающееся от строки, то следует воспользоваться двумя функциями ИЗ модуля django.core.Signing:

□ dumps (Оначение>[, key=None] [, salt='django.core.signing’ ] [, compress=False]) — подписывает указанное значение С применением класса Timestampsigner и возвращает результат в виде строки. Параметр key задает секретный ключ (по умолчанию — значение из параметра secret key настроек проекта), а параметр Salt— соль (по умолчанию— строка "django.core.signing”). Если параметру Compress передать значение True, то результат будет сформирован в сжатом виде (значение по умолчанию — False). Сжатие будет давать более заметный результат при подписывании данных большого объема. Примеры:

****

□ loads(Подписанное значение>[, key=None][, salt=’django.core.signing’][, max\_age=None]) — ИЗ полученного подписанного значения извлекает оригинальную величину и возвращает в качестве результата. Параметры key и salt указывают соответственно секретный ключ и соль — эти значения должны быть теми же, что использовались при подписывании значения вызовом функции Dumps (). Параметр max\_age задает промежуток времени, в течение которого цифровая подпись актуальна. Если он опущен, то проверка на актуальность подписи не проводится. Примеры:

****

Если цифровая подпись скомпрометирована или потеряла актуальность, то будут возбуждены исключения badsignature ИЛИ signatureexpired соответственно.