**Пример добавления CSRF токена**

<div style="display:none">

<input type="hidden" name="csrfmiddlewaretoken" value="$csrf\_token"/>

</div>

**Заменить пример в разделе "Всплывающие сообщения"**

Для добавления сообщения в очередь флэш сообщений, используется метод flash():

request.session.flash('mymessage')

Метод flash () добавляет сообщение в очередь, если это необходимо.

flash(message, queue='', allow\_duplicate=True)

Аргумент сообщение обязательно требуется. Он представляет собой сообщение, которое вы хотите отобразить пользователю. Это, как правило, строка, но сообщение, которое вы предоставляете, не изменяется каким-либо образом.

Аргумент очереди позволяет выбрать очередь, в которую добавить сообщение, которое вы предоставляете. Это может быть использовано, чтобы толкать различные виды сообщений, в флэш-памяти для последующего отображения в разных местах на странице. Вы можете передать любое имя для вашей очереди, но оно должно быть строкой. Каждая очередь не зависима, и может быть очищен с помощью метода pop\_flash () или с помощью peek\_flash () отдельно. По умолчанию очередь — это пустая строка. Пустая строка представляет очередь мгновенное сообщение по умолчанию.

request.session.flash(msg, 'myappsqueue')

request.session.pop\_flash()

request.session.peek\_flash()

**Резюме**

|  |
| --- |
| Скрипт для обновления резюме |
|  | import requests |
|  |  |
|  | def fun(): |
|  | access\_token = '\*\*\*\*\*' #your acces token |
|  | url = 'https://api.hh.ru/resumes/\*\*\*\*\*/publish' #your resume id |
|  | response = requests.post( |
|  | url , |
|  | headers={ |
|  | 'User-Agent': 'Test/1.0 (\*\*\*\*\*@gmail.com)', #your @mail |
|  | 'Content-Type': 'application/json', |
|  | 'Authorization': 'Bearer %s' % access\_token |
|  | } |
|  | ) |
|  | print(response.status\_code) |
|  | x = fun() |
|  | print('finish') |

**Добавить пример использования сессий пирамиды в других фреймворках**

Сессии в django

Django полностью поддерживает сессии для анонимных пользователей, позволяет сохранять и получать данные для каждой посетителя сайта. Механизм сессии сохраняет данные на сервере и самостоятельно управляет сессионными куками. Куки содержат ID сессии, а не сами данные.

Активация

Сессии реализованы с помощью отдельного модуля и моделей Django. Для активации сессий выполните следующие шаги:

Измените параметр MIDDLEWARE\_CLASSES, добавив в него значениеdjango.contrib.sessions.middleware.SessionMiddleware.

Внесите в параметр INSTALLED\_APPS значение django.contrib.sessions. Запустите manage.py syncdb после этого.

Стандартный шаблон приложения, созданный с помощью startproject уже имеет все необходимые настройки и если вы ничего не удаляли, то ничего добавлять не понадобится для работы сессий.

Если вам не требуется использовать сессии, вы можете удалить строку SessionMiddleware из параметраMIDDLEWARE\_CLASSES и строку django.contrib.sessions из параметра INSTALLED\_APPS. Это позволит вам освободить крайне незначительную часть ресурсов, но из маленьких частей складываются большие.

Использование в функциях представления

После активации SessionMiddleware каждый объект HttpRequest — первый аргумент любой функции представления Django — будет иметь атрибут session по функциональности аналогичный словарю. Пример:

# Установим значение ключа в сессии:

request.session["fav\_color"] = "blue"

# Получим значение ключа из сессии -- можно это сделать в

# другом представлении или через несколько запросов

# (или и то и другое):

fav\_color = request.session["fav\_color"]

# Чистим значение ключа в сессии:

del request.session["fav\_color"]

# Проверка наличия ключа в сессии:

if "fav\_color" in request.session:

...

Вы также можете использовать другие методы, такие как keys() и items() объекта request.session.

Есть ряд простых правил для эффективного использования сессий в Django:

Используйте для объекта request.session обычные строки Python в качестве ключей словаря. Это правило служит больше для удобства, чем для ускорения работы, но оно стоит того, чтобы ему следовать.

Словарные ключи сессии, которые начинаются с символа «подчёркивание», зарезервированы для внутреннего использования Django. На практике среда использует небольшой набор зарезервированных переменных в сессии, но пока вы не знаете об их назначении, не стоит их трогать.

Не заменяйте request.session новым объектом и не работайте с его атрибутами. Используйте его как обычный словарь Python.

Рассмотрим несколько примеров. Нижеприведённая простейшая функция представления устанавливает переменной has\_commented значение True после того, как пользователь отправляет комментарий. Это простой (но небезопасный) способ предотвращения повторной отправки пользовательского комментария:

def post\_comment(request, new\_comment):

if request.session.get('has\_commented', False):

return HttpResponse("Вы уже отправили комментарий")

c = comments.Comment(comment=new\_comment)

c.save()

request.session['has\_commented'] = True

return HttpResponse('Спасибо за ваш комментарий!')

Это простейшее представление аутентифицирует пользователя на сайте:

def login(request):

try:

m = Member.objects.get(username\_\_exact=request.POST['username'])

if m.password == request.POST['password']:

request.session['member\_id'] = m.id

return HttpResponse("Вы авторизованы.")

except Member.DoesNotExist:

return HttpResponse("Ваши логин и пароль не соответствуют.")

Это представление отключает ранее авторизованного пользователя:

def logout(request):

try:

del request.session['member\_id']

except KeyError:

pass

return HttpResponse("Вы вышли.")