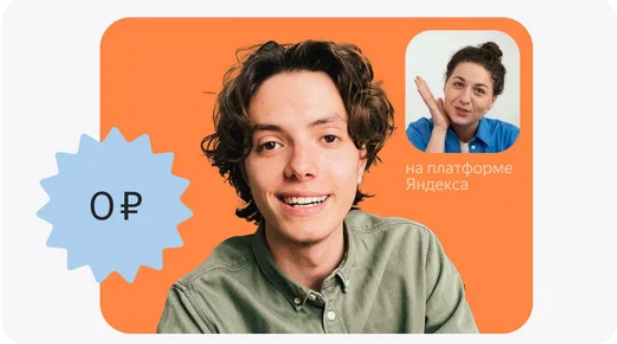


ХОЧУ ПОМОЧЬ
ПРОЕКТУ



practicum.yandex.ru

РЕКЛАМА · 18+ Я

Бесплатное занятие английским в Яндекс Практикуме

Тест на уровень языка • Разбор грамматики • Разговорная практика • За 30 минут

Узнать больше


Статья

Модуль http.cookies в Python

из файлов cookies

Модуль http.cookies предоставляет классы для абстрагирования концепции файлов cookie, механизма управления состоянием строк cookie, так и предоставляет абстракцию для использования любого cookie в качестве значения cookie.

Модуль http.cookies полезен для кода на стороне сервера, для генерации заголовков cookie, которые в последствии будут установлены клиенту.



Ранее модуль строго применял правила синтаксического анализа, описанные в спецификациях RFC 2109 и RFC 2068. С тех пор было обнаружено, что MSIE 3.0x не следует правилам символов, изложенным в этих спецификациях, а также многие современные браузеры и серверы имеют упрощенные правила синтаксического анализа, когда дело доходит до обработки файлов cookie. В результате используемые правила синтаксического анализа немного менее строги.

Набор символов, [string.ascii_letters](#), [string.digits](#) и `!#$%&'*+-.^_~:` обозначают набор допустимых символов, разрешенных этим модулем в имени файла cookie (как ключ).

Примечание. При обнаружении недопустимого файла cookie возникает [ошибка CookieError](#), поэтому, если данные cookie поступают из браузера, то при синтаксическом анализе, всегда надо готовиться к недопустимым данным и [перехватывать исключение](#) `http.cookies.CookieError`.

Содержание:

- `http.cookies.BaseCookie()` [объект подобный словарю](#),
- `http.cookies.SimpleCookie()` [является производным BaseCookie](#),
- `http.cookies.CookieError` [исключение модуля](#).
- [Объект cookie](#):
 - `BaseCookie.value_decode()` [ничего не декодирует, существует для переопределения](#),
 - `BaseCookie.value_encode()` [ничего не кодирует, существует для переопределения](#),
 - `BaseCookie.output()` [вернет строку заголовков cookie](#),
 - `BaseCookie.js_output()` [вернет фрагмент кода JavaScript](#),
 - `BaseCookie.load()` [анализирует строку как HTTP COOKIE](#).
- [Объект Morsel](#):
 - `Morsel.value` [значение cookie](#),
 - `Morsel.coded_value` [закодированное значение cookie](#),
 - `Morsel.key` [имя cookie](#),
 - `Morsel.set()` [устанавливает атрибуты key, value и coded_value](#),
 - `Morsel.isReservedKey()` [проверка членства ключа](#),
 - `Morsel.output()` [строковое представление Morsel](#),
 - `Morsel.js_output()` [фрагмент кода JavaScript](#),
 - `Morsel.OutputString()` [объект Morsel в строку](#),
 - `Morsel.update()` [обновляет словарь объекта Morsel](#),
 - `Morsel.copy()` [неглубокая копия Morsel](#),
 - `Morsel.setdefault()` [ведет себя так же, как dict.setdefault\(\)](#),
- [Примеры составления и генерации заголовков cookie модулем 'http.cookies'](#).

Вверх

https://docs-python.ru/standart-library/modul-http-cookies-python/

1/5

http.cookies.BaseCookie([input]):

Класс `http.cookies.BaseCookie()` представляет собой объект подобный [словарю](#), ключи которого являются строками, а значения - экземпляры класса [http.cookies.Morsel](#).

Объект `BaseCookie` устанавливает значение ключа, то он сначала преобразуется в [объект Morsel](#), содержащий ключ и значение. Если значение не задано, то он передается методу [BaseCookie.load\(\)](#).

http.cookies.BaseCookie([input]):

Класс `SimpleCookie` является производным от [BaseCookie](#) и переопределяет методы `value_decode()` и `value_encode()`.

Класс `SimpleCookie` берет строки как значения файлов cookie. При установке значения `SimpleCookie` вызывает преобразования значения в строку. Значения, полученные от HTTP, хранятся в виде строк.

`http.cookies:`

http.cookies.SimpleCookie:

Исключение `CookieError` может возникнуть из-за недействительности данных RFC 2109: неверные атрибуты, неправильный заголовок `Set-Cookie` и т. д.

Смотрите также модуль `http.cookiejar`, который представляет собой обработчик HTTP-файлов cookie для веб-клиентов. Модули `http.cookiejar` и `http.cookies` не зависят друг от друга.

Объект cookie.

- `BaseCookie.value_decode()` [ничего не декодирует, существует для переопределения](#),
- `BaseCookie.value_encode()` [ничего не кодирует, существует для переопределения](#),
- `BaseCookie.output()` [вернет строку заголовков cookie](#),
- `BaseCookie.js_output()` [вернет фрагмент кода JavaScript](#),
- `BaseCookie.load()` [анализирует строку как HTTP_COOKIE](#).

BaseCookie.value_decode(val):

Метод `BaseCookie.value_decode()` возвращает кортеж `(real_value, coded_value)` из строкового представления.

Аргумент `val` может быть любого типа. Этот метод в `BaseCookie` не выполняет декодирование, он просто существует, поэтому его можно переопределить.

BaseCookie.value_encode(val):

Метод `BaseCookie.value_encode()` возвращает кортеж `(real_value, coded_value)`.

Аргумент `val` может быть любого типа, но значение `coded_value` всегда будет преобразовано в строку. Этот метод в `BaseCookie` ничего не кодирует, он просто существует, поэтому его можно переопределить.

В общем, это должно быть так, что методы [.value_encode\(\)](#) и [.value_decode\(\)](#) инвертируются в диапазоне `value_decode`.

BaseCookie.output(attrs=None, header='Set-Cookie:', sep='\r\n'):

Метод `BaseCookie.output()` возвращает строковое представление, подходящее для отправки в виде заголовков HTTP.

Аргументы `attrs` и `header` отправляются каждому методу объекта [Morsel.output\(\)](#).

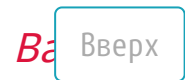
Аргумент `sep` используется для объединения заголовков и по умолчанию и представляет собой комбинацию `'\r\n'` (CRLF).

BaseCookie.js_output(attrs=None):

Метод `BaseCookie.js_output()` возвращает встраиваемый фрагмент кода JavaScript, который при запуске в браузере будет действовать так же, как если бы были отправлены заголовки HTTP.

Значение аргумента `attrs` такое же, как и в [BaseCookie.output\(\)](#).

BaseCookie.load(rawdata):



Morsel.output(attrs=None, header='Set-Cookie:'):

Метод Morsel.output() возвращает строковое представление объекта Morsel, подходящее для отправки в качестве заголовка HTTP.



В качестве заголовка HTTP включены все атрибуты. Если не указан аргумент attrs, то в этом атрибутах для передачи в качестве заголовка HTTP.

По умолчанию - Set-Cookie:.

Morsel.output(attrs=None):

Метод Morsel.output() возвращает встраиваемый фрагмент кода JavaScript, который при запуске в браузере, поведет действовать так же, как если бы был отправлен заголовок HTTP.

Значение такое же, как и в [Morsel.output\(\)](#).

Morsel.output(attrs=None):

Метод Morsel.output() возвращает строку, представляющую объект Morsel, без каких-либо окружающих HTTP или JavaScript кода.

Значение такое же, как и в [Morsel.output\(\)](#).

Morsel.update(values):

Метод Morsel.update() обновляет значения в словаре объекта Morsel значениями из словарных значений. Вызывает ошибку, если какой-либо из ключей values не является допустимым атрибутом RFC 2109 (имена допустимых атрибутов смотрите выше).

Morsel.copy(value):

Метод Morsel.copy() возвращает неглубокую копию объекта Morsel.

Morsel.setdefault(key, value=None):

Метод Morsel.setdefault() вызывает ошибку, если ключ не является допустимым атрибутом RFC 2109, в противном случае ведет себя так же, как [dict.setdefault\(\)](#).

Примеры составления и генерации заголовков cookie модулем http.cookies:

В примере показано как собирать cookie HTTP заголовки из ключей и значений.

```
>>> from http import cookies
>>> cook = cookies.SimpleCookie()
>>> cook["fig"] = "newton"
>>> cook["sugar"] = "wafer"

# генерация HTTP заголовков
>>> print(cook)
# Set-Cookie: fig=newton
# Set-Cookie: sugar=wafer

# То же самое
>>> print(cook.output())
# Set-Cookie: fig=newton
# Set-Cookie: sugar=wafer

# генерация HTTP заголовков с
# дополнительными параметрами
>>> cook = cookies.SimpleCookie()
>>> cook["rocky"] = "road"
>>> cook["rocky"]["path"] = "/cookie"
>>> print(cook.output(header="Cookie:"))
# Cookie: rocky=road; Path=/cookie
>>> cook.output(attrs=[], header="Cookie:")
# Cookie: rocky=road
```

В следующем примере показано как генерировать cookie HTTP заголовки из строки.

```
>>> from http import cookies
>>> cook = cookies.SimpleCookie()
#  такие куки
# =finger"
#
>>>
>>>
#
#
#
#
>>> cook["rybody; L=\\\"Loves\\\""; fudge=\\012;";"]
>>> cook()
>>>
>>>
#
rybody; L=\\\"Loves\\\""; fudge=\\012;";"
```

Пр...рые особенности модуля.

```
>>>
>>> cook = cookies.SimpleCookie()
>>> cook["oreo"] = "doublestuff"
>>> cook["oreo"]["path"] = "/"
>>> print(cook)
# Set-Cookie: oreo=doublestuff; Path=/

>>> cook = cookies.SimpleCookie()
>>> cook["twix"] = "none for you"
>>> cook["twix"].value
# 'none for you'

>>> cook = cookies.SimpleCookie()
# Следующая операция эквивалентна C["number"] = str(7)
>>> cook["number"] = 7
>>> cook["number"].value
# '7'
>>> cook["string"] = "seven"
>>> cook["string"].value
# 'seven'
>>> print(cook)
# Set-Cookie: number=7
# Set-Cookie: string=seven
```