Модуль urllib.parse в Python, операции с URL-адресом

Парсинг и манипулирование URL-адресом

Модуль urllib.parse определяет стандартный интерфейс для разбора URL-адреса на компоненты: протокол, порт, домен, путь и т. д., чтобы можно было объединять компоненты обратно в строку URL-адреса и преобразовать относительный URL-адрес в абсолютный URL-адрес с учетом базового URL-адреса.

Модуль был разработан в соответствии с "Internet RFC on Relative Uniform Resource Locators". Он поддерживает следующие схемы URL: file, ftp, http, https, imap, mailto, mms, news, nntp, rsync, sftp, shttp, sip, sips, svn, svn+ssh, telnet и т. д.

Разбор URL-адреса в байтовой кодировке.

Функции синтаксического анализа URL-адресов изначально были разработаны для работы только с символьными строками. На практике полезно иметь возможность манипулировать правильно цитируемыми и закодированными URL-адресами как последовательностями байтов ASCII. Соответственно, все функции синтаксического анализа URL в этом модуле работают с bytes и объектами bytearray в дополнение к объектам str.

Если переданы данные str, то результат также будет содержать только строковые данные str. Если переданы bytes или данные байтового массива bytearray, то соответственно результат будет содержать только байтовые данные.

Попытка смешать строковые данные с байтами или байтовым массивом в одном вызове функции приведет к возникновению ошибки TypeError, а попытка передать байтовые значения, отличные от ASCII, вызовет исключение UnicodeDecodeError.

Чтобы упростить преобразование результата между строками и байтами, все возвращаемые значения из функций синтаксического анализа URL предоставляют либо метод str.encode() (когда результат содержит данные str), либо метод bytes.decode() (когда результат содержит данные bytes).

Сигнатуры этих методов соответствуют сигнатурам соответствующих методов str и bytes, за исключением того, что кодировка по умолчанию - ascii, а не utf-8. Каждый из методов создает значение соответствующего типа, которое содержит либо данные bytes (для методов .encode()), либо данные str (для методов .decode()).

Вверх

Приложениям, которым необходимо работать с URL-адресами с потенциально неправильно процитированными кавычками, которые могут содержать данные, отличные от ASCII, необходимо будет выполнить собственное декодирование с байтов на символы перед вызовом методов синтаксического анализа URL.

Поведение, описанное выше, применимо только к функциям синтаксического анализа URL. Функции цитирования URL используют свои собственные правила при создании или использовании байтовых последовательностей.

Примеры использования модуля urllib.parse:

Пример разбора URL-адреса на компоненты:

```
>>> from urllib.parse import urlparse
>>> url = urlparse('http://www.cwi.nl:80/%7Eguido/Python.html')
>>> url
# ParseResult(scheme='http', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
#             params='', query='', fragment='')
>>> url.scheme
# 'http'
>>> url.port
# 80
>>> url.geturl()
'http://www.cwi.nl:80/%7Eguido/Python.html'
```

Пример функции для создания URL-адреса из компонентов:

```
>>> import urllib.parse
>>> def build_url(base_url, path, args_dict):
...     # Возвращает список в структуре urlparse.ParseResult
...     url_parts = list(urllib.parse.urlparse(base_url))
...     url_parts[2] = path
...     url_parts[4] = urllib.parse.urlencode(args_dict)
...     return urllib.parse.urlunparse(url_parts)
...
>>> args = {'arg1': 'value1', 'arg2': 'value2'}
# работает со сценарием двойной косой черты
>>> build_url('http://www.example.com/', '/somepage/index.html', args)
# http://www.example.com/somepage/index.html?arg1=value1&arg2=value2

# работает без косой черты
>>> build_url('http://www.example.com', 'somepage/index.html', args)
# http://www.example.com/somepage/index.html?arg1=value1&arg2=value2
```

Добавление/изменение протокола URL-адреса:

```
>>> from urllib.parse import urlparse
>>> url = urlparse('//www.cwi.nl:80/%7Eguido/Python.html')
>>> url
# ParseResult(scheme='', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
#             params='', query='', fragment='')

# Добавление протокола к URL-адресу
>>> url._replace(scheme='http')
ParseResult(scheme='http', netloc='www.cwi.nl:80', path='/%7Eguido/Python.html',
            params='', query='', fragment='')
```

Содержание раздела: