

HTTP-методы PUT и PATCH

Мы рассмотрели GET и POST запросы. В будущем мы также напишем DELETE запрос. К сожалению, в курсе не будут рассмотрены методы PUT и PATCH, поэтому давайте рассмотрим их в этом тексте.

Рассмотрим в виде таблицы, за что отвечает каждый метод и какие из этих методов принимают набор параметров, а какие — тело запроса (request body).

Метод

GET

POST

PUT

PATCH

DELETE

Назначение

Получение ресурса

Создание ресурса

Обновление ресурса

Обновление ресурса

Удаление ресурса

Запрос может содержать тело запроса

*Не обязательно

Нет

Да

Да

Да

Да*

И если с методами GET, POST, DELETE более-менее все понятно, то методы PUT и PATCH, на первый взгляд, занимаются одним и тем же.

Зачем же нужно разделять обновление ресурса на PUT и PATCH?

КУПЛЕНО НА
SKLADCHIK.COM

Довольно часто мы предоставляем пользователям возможность изменять данные, например, свой никнейм, название репозитория, свой комментарий и т.п.

Предположим, что в базе данных есть таблица с пользователями, в которой есть столбцы `username` и `email`. В случае с `PATCH`, если пользователь хочет сменить только `email`, не меняя `username`, ему достаточно послать параметр `email` с новым значением:

```
GET /users/1
{
  "username": "ramirez",
  "email": "ramirez@abc.com"
}
PATCH /users/1
{
  "email": "sebastian@ya.ru" <-- при использовании PATCH достаточно передать
  только изменяемый параметр
}
GET /users/1
{
  "username": "ramirez",
  "email": "sebastian@ya.ru"
}
```

В случае с `PUT`, если пользователь хочет сменить только `email`, не меняя `username`, он обязан послать и `email` и `username`:

```
GET /users/1
{
  "username": "ramirez",
  "email": "ramirez@abc.com"
}
PUT /users/1
{
  "username": "ramirez",
  "email": "sebastian@ya.ru" <-- при использовании PUT клиент обязан передать
  все параметры, даже если он их не изменяет.
}
GET /users/1
{
  "username": "ramirez",
  "email": "sebastian@ya.ru"
}
```

На этом отличия `PUT` и `PATCH` заканчиваются.