HTTP Methods

http methods: {

1. Get
2. Put
3. Patch
4. Delete
5. Post
6. Head
7. Options
8. Connect
9. Trace

}

|  |  |  |
| --- | --- | --- |
| methods | CRUD | e.g. |
| Post | Create | عمدتا یک رکورد در لیستی که درمنبعی مشخص است ایجاد میکند. |
| Get | Read | معمولا برای دریافت و خواندن اطلاعات استفاده میشه . |
| Put | Update/Replace | این متد برای آپدیت کردن دیتا استفاده میشود. |
| Patch | Update/Modify | این متد نیز برای آپدیت استفاده میشود که تفاوتش با متد قبلی توضیح داده شده . |
| Delete | Delete | ساده ترین متد برای حذف رکوردی خاص یا همه رکورد ها میباشد. |

حالا این CRUD که اون بالا نوشتیم چی هست ؟!

به مجموعه عملیات ساختن یا ایجاد (Create) . خواندن (Read) . آپدیت یا بروزرسانی یک رکورد (Update) و حذف رکوردی یا رکورد هایی (Delete) CRUD میگیم.

نکته : متد های Idempotent متد هایی هستن که اجرای درخواست های از این نوع هر چند مرتبه که انجام بشه نتیجه یکسانی داره. اما متد های غیر Idempotent متد هایی هستند که اجرا اونها به دفعات باعث بوجود آمدن آنومالی و side effect های متعددی خواهد شد.

متد POST :

این متد برای ساخت یا ایجاد رکوردی جدید در منبع (میتواند DB باشد) استفاده میشود

پارامتر های ارسالی از طریق نوار url قابل خواندن نیست .

این متد از نوع غیر Idempotent است و اجرای چندباره این نوع درخواست باعث بوجود آمدن رکورد های متعددی از پارامتر های تکراری خواهد بود

امنیت دیتای ارسالی در درخواست نسبتا بالاست (به قول بچه های تست نفوذ و امنیت no system is safe ) و بازه انواع درخواست های ارسالی شامل :

* رشته ها (داده های متنی)
* داده های باینری (آپلود فایل)

تمامی دیتاهای ارسالی در بدنه درخواست قرار میگیره .

مزایا و معایب متد Post :

* + امنیت نسبتا بالاتری نسبت به متد Get داره و اطلاعات ارسالی در Address bar مرورگر قال مشاهده نیستن.
  + سرعت درخواست های post (به مقدار خیلی خیلی کمی) کمتر از درخواست های متد Get هست.
  + محدودیت کمتری (هم به لحاظ نوع مقادیر ارسالی و هم به لحاظ طول url) نسبت به Get داره.(توی برخی منابع گفته شده که طول محدودی برای url ندارد.)
  + هرگز در history مرورگر ذخیره نمیشه.
  + به هیچ وجه قابل cache شدن هم نیست.

متد GET :

این متد از نوع Idempotent است. و اجرای چند باره این درخواست نتیجه یکسانی خواهد داشت.

این متد عمدتا برای خواندن اطلاعات از یک سورس مشخص استفاده میشن و فرمت response به صورت json/xml هست .

درخواست های از نوع get بدنه یا body ندارند چرا که عمدتا برای درخواست اطلاعات استفاده میشن.

گاهی برای فرمت کردن دیتایی که از قبل json یا xml نیست هم از این متد استفاده میشه.

این متد پارامتر های ورودی را با فرمت زیر معمولا ارسال میکند

کد زیر query Params پیش فرض سرچ در cms وردپرس میباشد.

Front-course.ir/?s=Search-Term

رشته هایی که space در میان واژگان دارند از طریق کاراکتر ‘-’ از هم جدا میشوند.

پارامتر ها با ? شروع میشوند و هر property میتواند value داشت باشد یا نداشته باشد.(بسته به برنامه نویسی سمت back-end)

مقادیر با & از هم جدا میشوند. به عنوان مثال:

front-course.ir/?name=Amin&user-age=&user-id=1

در کد بالا name و user-id هرکدام دارای مقداری هستند ولی user-age مقداری ندارد.

مزایا و معایب :

* برای ارسال اطلاعات مهم نظیر پسورد ها یا اطلاعات کارت های بانکی نمیتوان(نباید) از این متد استفاده کرد چون تمام مقادیر ارسالی در هر درخواست در address bar مرورگر قابل مشاهده هستن.
* داده ها میتوانند bookmark بشوند (امکان به اشتراک گذاری لینک حاوی این query params وجود دارد )
* متد get دارای محدودیت طول میباشد یعنی مقادیر بسیاری را نمیتوان در هر request ارسال کرد (max length = 2048 chars in url).
* عموما درخواست های از نوع Get قابل کش شدن هستن و توی لاگ سرور(عمدتا یک سری اسکریپت وجود داره که هزاران رکورد از تمام اتفاقاتی که میوفته (سمت کاربر یا سمت سرور) لاگ میگیرن ) هم ذخیره میشن.
* در درخواست های این نوع تنها کاراکتر های ASCII قابل ارسال هستن.

متد PUT :

این متد از نوع Idempotent است. و اجرای چند باره این درخواست نتیجه یکسانی خواهد داشت.

ماهیت این متد در ظاهر برای آپدیت یک رکورد از اطلاعات استفاده میشه ولی در عمل به این صورت هست که اول رکورد مورد نظر را پاک میکنه و (در همان مکان) یک رکورد جدید با استفاده از پارامتر های ارسالی جدید ایجاد میکنه که عملا دیتای ما Replace میشه .

حالا اینجا سوال این هست که اگر فیلد هایی که در سرور موجود هست جایگزینی در ارسال درخواست put ما نداشته باشن چه اتفاقی براشون میوفته ؟؟!

چون رکورد جاری حذف میشه و بعدش دیتای ما replace میشه فیلد هایی ک جایگزین جدیدی براشون تعریف نشده مقدار null خواهند گرفت. و عملا در صورت عدم درنظر گرفتن همه فیلد ها در درون درخواست مقداری از اطلاعات قبلی ما که نیازی به ویرایش یا آپدیت شدن نداشتن از بین میرن.

متد PATCH :

این متد هم از نوع Idempotent است و و اجرای چند باره این درخواست نتیجه یکسانی خواهد داشت.

این متد هم مکانیزمی مشابه به متد put داره ولی مزیتش نسبت به put این هست که فیلد هایی که مقادیر جایگزین در درخواست ارسالی براشون درنظر گرفته نمیشه . از بین نمیرن.

در اصل فقط مقادیری که باید نغییر کنند . تغییر میکنند.

مثلا اگر نیاز باشه پسورد یک کاربر عوض بشه اگر از متد put استفاده میکردیم همه اطلاعات کاربری اون شخص اعم از ایمیل و نام کاربری از بین میرفت.

ولی در متد Patch فقط پسورد عوض میشه و باقی فیلد ها مقادیرشون دست نخورده باقی میمونن (از بین نمیرن).

متد DELETE :

این متد برخلاف متد Post از نوع Idempotent است و اجرای چند باره این درخواست نتیجه یکسانی خواهد داشت.

ساده ترین روش حذف فایل یا رکوردی خاص از یک منبع مشخص است.

ممکن است این request ها body داشته باشند.

Response های موفقیت آمیز هم ممکن است body داشته باشد.

این نوع درخواست ها در مرورگر cache نمیشوند.

عموما وقتی درخواستی موفقیت آمیز ارسال میشه . Response ها با http message های متفاوتی ممکن هست دریافت بشه.

برای مثال درخواست زیر میگه فایلی به نام badfile.html رو از سرور حذف کن.

DELETE /badFile.htm HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.TorfehNegar.com

Accept-Language: en-us

Connection: Keep-Alive

و در پاسخ چنین چیزی دریافت میشه:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2019 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Content-type: text/html

Content-length: 30

Connection: Closed

که با توجه به message code 200 که به منزله ok هست میگه که این کار انجام شد

و اگر این درخواست body هم داشته باشه به شکل زیر میشه :

<html>

<body>

<h1>‌url deleted.</h1>

</body>

</html>

متد HEAD :

همانند متد get عمل میکنه با این تفاوت که فقط head درخواست return میشه که این head شامل یک سری اطلاعات از پارامتر های ارسالی و... است. Body درخواست return نمیشه.

در عمل بیشتر برای چک کردن مقدار برگشتی هر Get Request استفاده میشه .

متد OPTIONS :

این متد عمدتا زمانی استفاده میشه که میخوایم بدونیم چه متد هایی رو وب سرور مد نظرما پشتیبانی میکنه

که میتونیم url خاصی رو بهش بدیم یا اینکه از کاراکتر Astrisk (\*) جلوی OPTIONS استفاده کنیم تا ببینیم در کلیت سرور چه متد هایی ساپورت میشه.

برای مثال اینجا داریم که :

OPTIONS \* HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

و درصورت موفقیت آمیز بودن درخواست نتیجه به شکل زیر خواهد بود :

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2019 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Allow: GET,HEAD,POST,OPTIONS,TRACE

Content-Type: httpd/unix-directory

اگر به سطر سوم دقت کنید تمام متد های پشتیبانی شده رو برای ما مشخص کرده .

متد TRACE :

این متد معمولا برای دیباگ کردن و در پروسه Development استفاده میشه.

حالا به چه صورت؟؟!

به این شکل که وقتی درخواستی از نوعTRACE برای سرور از سمت کلاینت ارسال میشه . سرور برای کلاینت درخواستی که کلاینت فرستاده بود رو مجدد میفرسته تا کلاینت بتونه Content درخواستی که ارسال کرده بوده رو مشاهده کنه.

مثال:

TRACE / HTTP/1.1

Host: www.TorfehNegar.com

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

و نتیجه برگشتی درصورت موفقیت آمیز بودن درخواست به شکل زیر خواهد بود.

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2019 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Connection: close

Content-Type: message/http

Content-Length: 39

TRACE / HTTP/1.1

Host: www.TorfehNegar.com.com

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

متد CONNECT

با استفاده از این متد میشه یک اتصال یا connection بین کلاینت و وب سرورمون که توی مثال ما Apache هست برقرار کنیم :

برای مثال توی درخواست زیر قرار هست که به سرور فرضا طرفه نگار درخواست کانکشن بدیم

CONNECT www.TorfehNegar.com HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

در صوررتی که این درخواست با موفقیت انجام بشه نتیجه Head به شکل زیر خواهد بود.

HTTP/1.1 200 Connection established

Date: Mon, 27 Jul 2019 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

منابع:

MDN

Tutorials Point

W3