


Open Opened 2 days ago by  **Rubén Montero**

POST

Resumen

- Veremos qué es un verbo ó método HTTP
- Entenderemos que tienen diferente *semántica*
- Crearemos un *endpoint* que procesa una petición **POST**
 - Especificaremos el decorador `@csrf_exempt`
- Como ya no servirá el navegador web, usaremos `curl` para probarlo

Descripción

Cuando vimos las [peticiones HTTP](#) dijimos que constaban de **tres** partes, de las cuales sólo es *obligatoria* la primera:

- Línea de petición
- Cabeceras
- Cuerpo

Pues bien, la línea de petición está compuesta *principalmente* de **2** cosas:

- Método (o verbo) HTTP
- Recurso solicitado

¿Método HTTP?

Es un valor de los definidos en el [RFC 7231](#). `GET` es uno de ellos.

Así, una petición HTTP típica de un navegador web podría tener esta línea de petición:

```
GET /index.php
```

¿Y existen más?

Sí: `HEAD` , `POST` , `PUT` , `DELETE` , `CONNECT` , `OPTIONS` , `TRACE` , `PATCH` .

¿Por qué hay varios?

Cada uno tiene una *semántica* diferente. Esto es parte de la arquitectura HTTP y sirve para que se pueda *distinguir* el objetivo de una petición.

Por ejemplo:

- Una petición `GET` se considera *segura*. Sólo **lee** datos del servidor
- Una petición `DELETE` se considera *insegura*. Puede eliminar datos del servidor

Pero... ¿para qué?

Como **primer ejemplo**, imaginemos un [webcrawler](#) de Google ejecutándose.

Un *webcrawler* es un proceso que manda peticiones automáticas y *navega* por la web, para *descubrirla*. ¡Así es como Google sabe qué resultados de búsqueda ofrecerte! Sus *webcrawlers* navegan por la web con frecuencia.

Pues bien, si no hubiera diferentes métodos HTTP, un *webcrawler* podría mandar peticiones que, sin querer, eliminasen datos de un servidor. ¡Que error!

- Sin embargo, un *webcrawler* sólo manda peticiones `GET` , y así, *investiga* la web sin causar daño

Como **segundo ejemplo**, expondremos brevemente que `POST` , además de ser inseguro, tiene **permitido** producir efectos *colaterales*. Esto tiene que ver con el concepto de [idempotencia](#).

El método HTTP `POST` **no** es idempotente.

Por eso, a veces en el navegador nos salta una alerta *¿Desea confirmar el reenvío del formulario?* Esto sucede al refrescar una página que lanzaba un `POST` porque... ¡Podríamos producir efectos *colaterales* al recargarla!

- Gracias a que `POST` está definido como *no idempotente*, el navegador sabe que debe *avisarnos* de que vamos a repetir una petición peligrosa. Por ejemplo, podríamos pagar **2** veces un pedido en una tienda on-line.

Hhmm... Ya veo...

Quedémonos con un resumen en forma de tabla de los **4** métodos HTTP que más nos importan:

-	GET	PUT	DELETE	POST
¿Seguro?	Sí	No	No	No
¿Idempotente?	Sí	Sí	Sí	No

Vale

Bien.

¿Y ahora?

Vamos a crear un nuevo *endpoint* en nuestro proyecto Django que **procese una petición POST**.

Empecemos como siempre:

urls.py

Añadiremos una nueva línea:

```
urlpatterns = [
    # ...
    path('resource/<int:number>', endpoints.resource_example),
]
```

endpoints.py

Aquí, **añadiremos** una función con el nombre apropiado y que declara recibir `number` como parámetro:

```
def resource_example(request, number):
    # ...
```

Como vamos a procesar un POST, debemos añadir el *decorador* `@csrf_exempt`¹. Así:

```
from django.views.decorators.csrf import csrf_exempt

# ...

@csrf_exempt
def resource_example(request, number):
    # ...
```

request.method

Para *comprobar* **qué** método HTTP fue usado en la petición, sólo hay que investigar `request.method`.

```
@csrf_exempt
def resource_example(request, number):
    if request.method == 'POST':
        return JsonResponse({"message": "You have sent a POST to the resource " + str(number)})
```

Si es otro método distinto de **POST**, vamos a devolver una respuesta distinta:

```
@csrf_exempt
def resource_example(request, number):
    if request.method == 'POST':
        return JsonResponse({"message": "You have sent a POST to the resource " + str(number)})
    else:
        return JsonResponse({"error": "HTTP method not supported"})
```

Ahorremos líneas de código...

La función anterior se puede escribir ahorrándonos el `else`.

Es **equivalente** a:

```
@csrf_exempt
def resource_example(request, number):
    if request.method == 'POST':
        return JsonResponse({"message": "You have sent a POST to the resource " + str(number)})
    return JsonResponse({"error": "HTTP method not supported"})
```

Bien, probemos nuestro *endpoint*

Si lanzas el servidor con `python manage.py runserver` y visitas la siguiente URL:

- <http://localhost:8000/resource/15>

...¿qué respuesta ves?

Mi navegador **no** ha mandado un POST

Correcto.

¿Qué ha mandado?

Un GET.

Siempre manda GET.

¿Y cómo envío un HTTP **POST** ?

Debes usar **algo distinto**.

Puedes usar alternativas como [Postman](#), pero nosotros veremos cómo usar una utilidad que viene con todos los sistemas operativos.

cURL

Abre un terminal (cmd.exe) de Windows y escribe `curl`.

Al darle a ENTER verás:

```
curl: try 'curl --help' for more information
```

Esta utilidad ya está instalada. Basta con escribir `curl <url>` para enviar una **petición HTTP**. Puedes probar con `curl www.google.es`. ¿Qué ves?

Usando **cURL** para nuestro *endpoint*

Si tecleas `curl localhost:8000/resource/10` verás lo mismo que desde el navegador:

```
> curl localhost:8000/resource/10
{"error": "HTTP method not supported"}
```

Por defecto, `curl` también envía un `GET`.

Pero **podemos especificar el método HTTP usando el argumento `-X` ó `--request`**.

Así:

```
> curl -X POST localhost:8000/resource/10
```

Si ves:


```
{"message": "You have sent a POST to the resource 10"}
```

¡Enhorabuena! Has implementado y verificado tu primer ***endpoint POST*** con Django.

Por último

Comprueba que tu código pasa el *test* asociado a la tarea.

Haz **commit** y **push** para subir los cambios al repositorio.

1. En este paso deshabilitamos un mecanismo de Django para proteger los formularios web de ataques [Cross Site Request Forgery](#). Como no estamos desarrollando páginas web, sino APIs REST, nos daría problemas tenerlo habilitado. 
-



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 4](#) 2 days ago



[Ania Blanco @ania.blanco](#) mentioned in commit [c9058277](#) 1 hour ago