


Open Opened 2 days ago by  [Rubén Montero](#)

Cuerpo de petición

Resumen

- Comprenderemos *qué* es el cuerpo de una petición HTTP
- Veremos cómo *enviar* JSON al servidor
- En nuestro *endpoint* de la tarea anterior, interpretaremos un valor recibido en el *cuerpo de la petición*

Descripción

El flujo de información hasta ahora ha sido relativamente sencillo:

1. Mandamos una petición a una URL (con o sin parámetros)
2. El servidor responde un JSON y un código HTTP

Pero...

El cliente HTTP **también** puede enviar JSON

¿Sí?

Sí

¿Y cómo?

En el **cuerpo de la petición HTTP**

Recordemos que las peticiones HTTP tienen **tres** partes:

- Línea de petición
- Cabeceras
- Cuerpo

Pues bien, el *cuerpo* puede tener un contenido *totalmente libre*.

Una restricción:

Las peticiones **GET** no **pueden** llevar cuerpo.

Pero las **POST** ... ¡sí!

Un momento, un momento...

Ya puedo enviar datos al servidor como *query params* ó *path params*. ¿Por qué necesito enviar datos en el **cuerpo HTTP?**

Cada parte de la petición HTTP tiene su cometido.

El cuerpo está destinado a albergar datos que son de mayor tamaño o cumplen un propósito que no *encaja* en la URL.

Por ejemplo: Si estás subiendo una imagen JPG a tu perfil, ¿cómo va a viajar en la URL?

Vale, entonces las peticiones POST pueden tener **cuerpo**

Sí.

Y nosotros trabajaremos enviando contenido JSON en ese **cuerpo**

Sí.

Y, ¿cómo?

Desde [cURL](#), podemos usar `-d` :

```
curl -X POST http://localhost:8000/resource/10 -d '{"mood": "Happy"}'
```

¡Ah! Si usamos `-d`, **cURL** entenderá que estamos enviando un **POST**, así que podemos *omitir* `-X POST`:

```
curl http://localhost:8000/resource/10 -d '{"mood": "Happy"}'
```

¡Ojo! Este ejemplo aún no está completo. Hay que [escapar las dobles comillas](#) (`" "`) dentro del JSON, o el terminal de Windows tendrá problemas:

```
curl http://localhost:8000/resource/10 -d '{"mood": \"Happy\"}'
```

¡Ejemplo **completo**!

Y esto a mi *endpoint*, ¿qué le importa?

Podemos **procesar** valores enviados por el cliente en el cuerpo HTTP.

Django nos los *pasa* en `request.body`.

Manos a la obra

Nuestro método era así:

```
@csrf_exempt
def resource_example(request, number):
    if request.method != 'POST':
        return JsonResponse({"error": "HTTP method not supported"}, status=405)
    # Precondición comprobada: EL método es POST
    return JsonResponse({"message": "You have sent a POST to the resource " + str(number)})
```

1) Comprobamos que `request.body` tiene contenido

Lo podemos hacer verificando el *tamaño* de `request.body`:

```
@csrf_exempt
def resource_example(request, number):
    if request.method != 'POST':
        return JsonResponse({"error": "HTTP method not supported"}, status=405)

    # Precondición comprobada: EL método es POST
    if len(request.body) == 0:
        # No hay cuerpo de petición
        # Nos comportamos como antes
        return JsonResponse({"message": "You have sent a POST to the resource " + str(number)})

    # Si hemos Llegado aquí, el cuerpo HTTP de la petición tiene algo
    # ...
```

2) Traducimos el JSON del cuerpo con `json.loads`

Para *interpretar* el cuerpo de la petición HTTP, usamos la librería `json`¹ sobre `request.body`.

```
import json

# ...

@csrf_exempt
def resource_example(request, number):

    # ...

    # Si hemos Llegado aquí, el cuerpo HTTP de la petición tiene algo
    http_body = json.loads(request.body)
```

3) Lo procesamos como un diccionario Python normal

Completaremos nuestro `resource_example` así:

```
# Si hemos Llegado aquí, el cuerpo HTTP de la petición tiene algo
http_body = json.loads(request.body)
client_mood = http_body.get("mood", "No mood") # "No mood" será un valor por defecto
return JsonResponse({"message": "You have sent a POST to the resource " + str(number) + " and you're in the following mood: " + client_mood})
```


¡Tarea completada!

Ya puedes usar el comando `curl` de arriba y verificar *qué* responde nuestro servidor REST.

Por último

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

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

-
1. Si alguna te encuentras con un error como `UnicodeDecodeError: 'utf-8' codec can't decode byte`, seguramente estás mandando caracteres especiales (e.g.: `á`, `é`, `í`, ...) que viajan codificados según su código Unicode. Prueba a usar `json.loads(request.body.decode('raw_unicode_escape'))` en lugar de `json.loads(request.body)` 
-



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



[Ania Blanco @ania.blanco](#) mentioned in commit [02d45bad](#) 1 hour ago