

Validando DTO's de forma fácil con Symfony

Primero creamos la clase DTO
con las siguientes propiedades:

- **from**: Obligatorio
- **to**: Obligatorio

```
use Symfony\Component\Validator\Constraints as Assert;

readonly class QueryInputDto
{
    public function __construct(
        #[Assert\Datetime(message: 'From must be a valid datetime')]
        #[Assert\NotBlank(message: 'From date cannot be empty')]
        public string $from,
        #[Assert\Datetime(message: 'To must be a valid datetime')]
        #[Assert\NotBlank(message: 'To date cannot be empty')]
        public string $to
    ){}

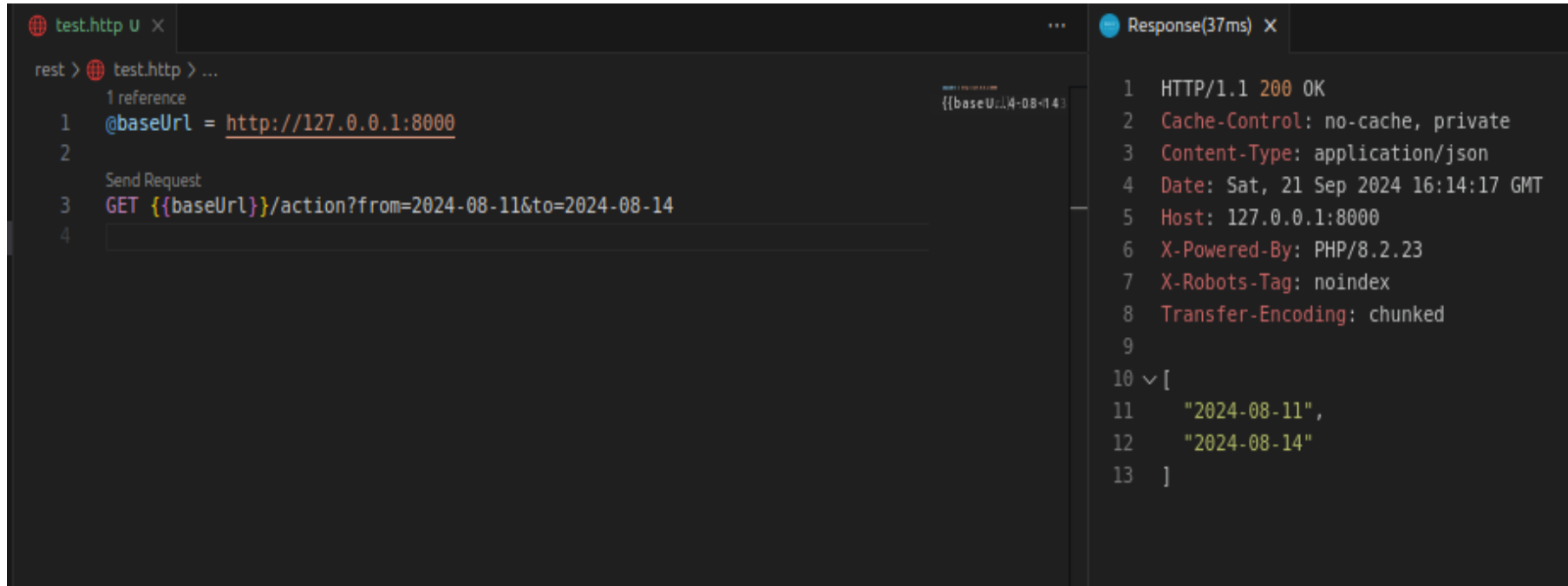
    public function getAll(): array
    {
        return [
            $this->from, $this->to
        ];
    }
}
```

Ahora usamos el atributo “MapQueryString” para que los datos que llegan en el query string se mapeen al DTO que hemos creado

```
use App\Dto\QueryInputDto;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpKernel\Attribute\MapQueryString;
use Symfony\Component\Routing\Attribute\Route;

class ApiController extends AbstractController
{
    #[Route("/action", name:"get_route", methods: ["GET"])]
    public function getAction(#[MapQueryString] QueryInputDto $queryInputDto): JsonResponse
    {
        return new JsonResponse($queryInputDto->getAll());
    }
}
```

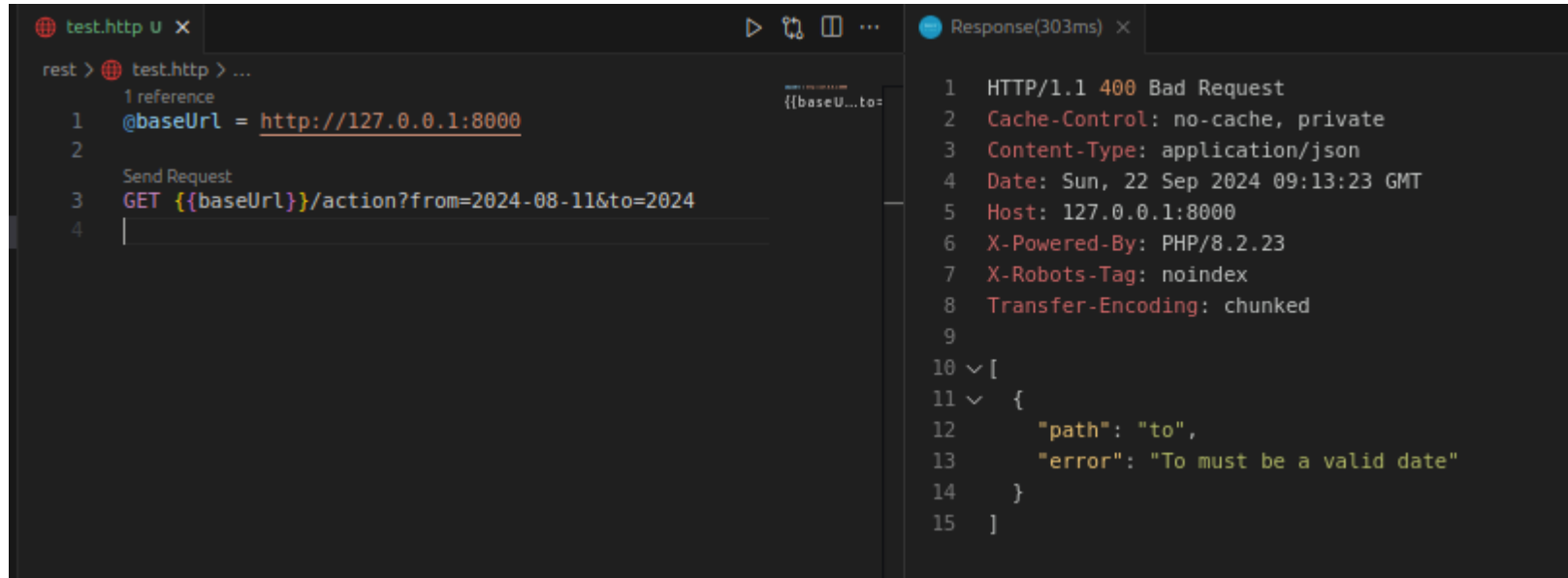
Si realizamos una petición, vemos como los datos se han mapeado correctamente



The screenshot shows a REST client interface with a request tab on the left and a response tab on the right. The request is a GET to a URL with a base URL variable and date parameters. The response is a 200 OK status with various headers and a JSON array of dates.

```
rest > test.http > ...  
1 reference  
1 @baseUrl = http://127.0.0.1:8000  
2  
Send Request  
3 GET {{baseUrl}}/action?from=2024-08-11&to=2024-08-14  
4  
  
Response(37ms) X  
1 HTTP/1.1 200 OK  
2 Cache-Control: no-cache, private  
3 Content-Type: application/json  
4 Date: Sat, 21 Sep 2024 16:14:17 GMT  
5 Host: 127.0.0.1:8000  
6 X-Powered-By: PHP/8.2.23  
7 X-Robots-Tag: noindex  
8 Transfer-Encoding: chunked  
9  
10 √ [  
11   "2024-08-11",  
12   "2024-08-14"  
13 ]
```

Si realizamos una petición en la cual se envía algún dato incorrecto, vemos como se devuelve el error de validación



The screenshot shows a REST client interface with two panels. The left panel displays the request configuration, and the right panel shows the response details.

Request Panel:

- Tab: test.http U x
- rest > test.http > ...
- 1 reference
- 1 @baseUrl = http://127.0.0.1:8000
- 2
- Send Request
- 3 GET {{baseUrl}}/action?from=2024-08-11&to=2024
- 4

Response Panel:

- Tab: Response(303ms) x
- 1 HTTP/1.1 400 Bad Request
- 2 Cache-Control: no-cache, private
- 3 Content-Type: application/json
- 4 Date: Sun, 22 Sep 2024 09:13:23 GMT
- 5 Host: 127.0.0.1:8000
- 6 X-Powered-By: PHP/8.2.23
- 7 X-Robots-Tag: noindex
- 8 Transfer-Encoding: chunked
- 9
- 10 ∨ [
- 11 ∨ {
- 12 "path": "to",
- 13 "error": "To must be a valid date"
- 14 }
- 15]