


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

Las tablas del 0 al $+\infty$

Resumen

- Veremos que en Django se pueden *mapear path params* de tipo `int` específicamente
- Añadiremos una nueva función a `endpoints.py` que recibirá como parámetro un `int` directamente y calculará la tabla de multiplicar del número recibido, sea cual sea

Descripción

En nuestra tarea anterior hemos implementado *a mano* las respuestas para las tablas de multiplicar del 1 al 10. Esto es un poco tedioso, ¿no?

La [unidad aritmético-lógica \(ALU\)](#) de nuestro ordenador puede procesar un montón de operaciones en cuestión de milisegundos

Calcular una sencilla tabla de multiplicar

No le costará mucho.

En Python, bastaría el código:

```
def multiply_number_improved(number):
    result = []
    for i in range(1, 11):
        result.append(number * i)
    return result
```

Ojalá ese `number` pudiera equivaler al **número** cuando se visita una URL como:

- <http://localhost:8000/v2/multiplication/5>
- <http://localhost:8000/v2/multiplication/8>
- <http://localhost:8000/v2/multiplication/142>

En ese caso, podríamos añadir una segunda *versión* mejorada del *endpoint* de la tarea anterior...

¡Sí que se puede!

¿De verdad? ¿Cómo?

En Django, se puede especificar que un *path param* es numérico

¡Interesante!

¿Y cómo?

`urls.py`

Vamos a añadir un nuevo *mapeo* a este fichero, pero esta vez **tendrá el prefijo `int:` en el nombre**. Así:

```
urlpatterns = [
    # ...
    path('v2/multiplication/<int:number>', ... ),
]
```

Si **añadimos** la función de arriba a `endpoints.py`, podemos **conectarla** al *endpoint* directamente:

```
urlpatterns = [
    # ...
    path('v2/multiplication/<int:number>', endpoints.multiply_number_improved),
]
```

Fíjate en los siguientes detalles:

- En la URL, `number` lleva el prefijo `int`:
- **Sólo** valores numéricos **positivos** en la URL son válidos. Por ejemplo, si el usuario visita http://localhost:8000/v2/multiplication/pepe_depura, **no** se ejecutará la función.
 - Por ello, dentro de `multiply_number_improved` tenemos **garantizado** que `number` es de tipo entero (`int`) y **positivo**

La tarea

Si has seguido los pasos expuestos ya está casi todo el trabajo hecho.

Tan sólo:

1) Modifica la firma de `multiply_number_improved` para que declare el **necesario** parámetro `request` :

```
-def multiply_number_improved(number):  
+def multiply_number_improved(request, number):
```

2) Modifica el valor devuelto (`return`) para que sea una `JsonResponse`

- Recuerda que necesitarás `safe=False`

Cuando completes estos pasos, lanza el servidor con `python manage.py runserver` y visita tu *endpoint* para comprobar que funciona

Por último

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

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



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



[Ania Blanco @ania.blanco](#) mentioned in commit [ca8d5488](#) 7 hours ago