


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

# 1 + 1 + 1 = 3 pequeñas mejoras

## Resumen

- Añadiremos soporte para el *query param* `?older_than=<fecha>` en el *endpoint* `/entries`
- Si viene, devolveremos sólo las *Entradas* con **fecha de publicación** anterior a `<fecha>`

## Descripción

Nuestro *endpoint* `/entries` ahora permite devolver **hasta N resultados**.

Pero, ¿y si un usuario quiere ver resultados *más antiguos*?

No puede.

Debemos **permitirlo**, y la respuesta que debemos dar es **paginación**.

### Paginación

Mostrando *página 1 de 523* en Google, ó los Tweets de tu *timeline* que aparecen instantáneamente cuando haces *scroll* hasta el final, son dos buenos ejemplos de paginación.

Consiste en **presentar los datos al usuario poco a poco**.

### Hablando con ejemplos

Usemos un ejemplo ficticio, donde un *endpoint*:

- <http://algunservidor:8000/mensajes>

...devuelve los siguientes datos:

Todos los datos
Mensaje 1: ¡Bonito domingo para terminar la semana! <i>(Escrito el domingo a las 22.00)</i>
Mensaje 2: ¡Un sábado de deporte! <i>(Escrito el sábado a las 21.00)</i>
Mensaje 3: Hoy empieza el fin de semana 🥳 <i>(Escrito el viernes a las 20.30)</i>
Mensaje 4: El día de Júpiter <i>(Escrito el jueves a las 18.30)</i>
Mensaje 5: Media semana, ¡qué bien! <i>(Escrito el miércoles a las 11.05)</i>
Mensaje 6: Ni te cases ni te embarques <i>(Escrito el martes a las 11.00)</i>
Mensaje 7: ¡Así empezó todo! <i>(Escrito el lunes a las 9.00)</i>

### Entonces

Si soportase parámetros como `size` y `older_than`, el cliente HTTP podría efectuar las siguientes peticiones para consumir los datos **poco a poco** (en páginas de 3 elementos):

- <http://algunservidor:8080/mensajes?size=3>

Primera página
Mensaje 1: ¡Bonito domingo para terminar la semana! <i>(Escrito el domingo a las 22.00)</i>
Mensaje 2: ¡Un sábado de deporte! <i>(Escrito el sábado a las 21.00)</i>
Mensaje 3: Hoy empieza el fin de semana 🥳 <i>(Escrito el viernes a las 20.30)</i>

- [http://algunservidor:8080/mensajes?older\\_than=viernes20.30&size=3](http://algunservidor:8080/mensajes?older_than=viernes20.30&size=3)

Segunda página
Mensaje 4: El día de Júpiter ( <i>Escrito el jueves a las 18.30</i> )
Mensaje 5: Media semana, ¡qué bien! ( <i>Escrito el miércoles a las 11.05</i> )
Mensaje 6: Ni te cases ni te embarques ( <i>Escrito el martes a las 11.00</i> )

- [http://algunservidor:8080/mensajes?older\\_than=martes11.00&size=3](http://algunservidor:8080/mensajes?older_than=martes11.00&size=3)

Tercera página
Mensaje 7: ¡Así empezó todo! ( <i>Escrito el lunes a las 9.00</i> )

## older\_than

Para soportarlo, querremos usar la `Entry.objects.filter(publication_date__lt=<fecha>)`

### Un momento, ¿qué?

En `filter` podemos *filtrar* las filas de una tabla por **igualdad** de un atributo y un valor:

```
values = Entry.objects.filter(publication_date=<fecha>)
```

...pero también podemos usar los *sufijos*:

- `__lt` : *less than*
- `__lte` : *less than or equal*
- `__gt` : *greater than*
- `__gte` : *greater than or equal*

...para comparar valores *numéricos*.

Por ejemplo:

```
values = Partida.objects.filter(puntuacion__gte=200)
```

### En resumen

Como las fechas se puede comparar (`>`, `<`, `=`) igual que los números, podemos usar:

```
all_rows = Entry.objects.filter(publication_date__lt=<fecha>) # Entradas más antiguas que <fecha>
```

### Manos a la obra

Vayamos a la parte que procesa el `"GET"` de `all_entries` y asignemos el *query param* `?older_than` a una **variable**:

```
if request.method == "GET":
+   older_than = request.GET.get("older_than", None)
    size = request.GET.get("size", None)
    if size is not None:
        try:
            size = int(size)
        except ValueError:
            return JsonResponse({"error": "Wrong size parameter"}, status=400)
    # Y ahora, haremos la consulta
    # ...
```

**No** hace falta que efectuemos una validación. Aunque sería una buena práctica, vamos a evitar complicar demasiado el código.

Ya tenemos **4** casos que contemplar ante la pregunta:

- ¿Viene el *query param* en la petición?

size	older_than
No	No
No	Sí
Sí	No
Sí	Sí

A continuación se da una versión de código que contempla claramente estos **4** casos y asigna el `all_rows` adecuado en cada uno:

(Nótese cómo podemos concatenar `.filter()` y `.order_by()` )

```
# Y ahora, haremos la consulta
if size is None:
    if older_than is None:
        all_rows = Entry.objects.order_by("-publication_date")
    else:
        all_rows = Entry.objects.filter(publication_date__lt=older_than).order_by("-publication_date")

else:
    if older_than is None:
        all_rows = Entry.objects.order_by("-publication_date")[:size]
    else:
        all_rows = Entry.objects.filter(publication_date__lt=older_than).order_by("-publication_date")[:size]
json_response = []
for row in all_rows:
    json_response.append(row.to_json())
return JsonResponse(json_response, safe=False)
```

Sin embargo, existen **muchas formas** y **más compactas** de implementar esta funcionalidad.

¿Te atreves a mejorar este código?

## Por último

Comprueba 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