


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

Comentar, comentar...

Resumen

- Crearemos un *endpoint* para recuperar los comentarios de una entrada

Descripción

Ahora mismo tenemos funcionando un *endpoint* para recuperar **todas** las *Entradas*:

- <http://localhost:8000/entries>

Podríamos hacer un *endpoint* para **todos** los comentarios, pero eso **no tiene mucho sentido**.

Lo normal es que los clientes HTTP muestren al usuario una lista de *Entradas*, y cuando el usuario *abre* una de ellas, ve los **comentarios** asociados.

Por ello, vamos a crear un *endpoint* **GET** que devuelva todos los *Comentarios*...

...de una *Entrada*

Será:

```
GET /entries/<int:path_param_id>/comments
```

Como puedes ver, tiene un *path param* que será el **ID** de la *Entrada* que **tiene Comentarios**.

endpoints.py

Añadamos una nueva función para que sirva nuestro propósito dentro de `wallrest04app/endpoints.py` :

```
def entry_comments(request, path_param_id):
    if request.method != "GET":
        return JsonResponse({"error": "HTTP method not supported"}, status=405)
    # ...
```

Ya ves que hemos declarado el `path_param_id`.

Ahora queremos recuperar *todas las filas* de la tabla `Comment` cuyo `entry` sea el mismo que el recibido (`path_param_id`). En otras palabras, todos los `Comment` asociados a la *Entrada* con `id=path_param_id`.

Dos alternativas

1) La fácil

La forma más intuitiva sería lanzar una sentencia SQL como:

```
SELECT *
FROM wallrest04app_comment
WHERE entry=<path_param_id>
```

Para esto podemos usar `.filter()`, que nos permite especificar **condiciones**.

```
from .models import Comment # Hay que añadir también este import

comments = Comment.objects.filter(entry=path_param_id)
```

2) La fácil... Pero más larga

Django nos ofrece *ayudas* para acceder a los datos.

Vamos a ver una alternativa que puede facilitarnos recuperar datos de una relación **1:N**.

2.1) Recuperar la *Entrada*

Con `.get()` podemos lanzar un `SELECT` especificando condiciones igual que con `.filter()`. La diferencia es que `.get()` entrega siempre una (1) fila.

```
entry = Entry.objects.get(id=path_param_id)
```

Con dicha línea, hemos recuperado la *fila* de la *Entrada* asociada a los comentarios.

DoesNotExist

¡Ojo! Si una *Entry* con ese *id* no existe, `.get()` lanzará una excepción que debemos controlar así:

```
try:
    entry = Entry.objects.get(id=path_param_id)
except Entry.DoesNotExist:
    return JsonResponse({"error": "Entry not found"}, status=404)
```

2.2) Extraer los *Comentarios* de la *Entrada*

Una vez tenemos la *instancia* de *Entry*, podemos acceder a sus comentarios. Hay que invocar un atributo especial autogenerado:

- Comienza con el nombre del modelo asociado en minúsculas (`comment`)
- Termina con `_set`
- Debemos invocar `.all()` para que entregue un `QuerySet []` iterable

Es decir:

```
try:
    entry = Entry.objects.get(id=path_param_id)
except Entry.DoesNotExist:
    return JsonResponse({"error": "Entry not found"}, status=404)
# La Entrada sí existe. Vamos a acceder a los Comentarios relacionados
comments = entry.comment_set.all() # Así
```

Escoge tu alternativa

Cualquiera producirá un `comments` con los mismos valores de los comentarios de la *Entrada*:

Y a partir de aquí... Ya sabemos como seguir

La lógica de nuestra función será la misma que la usada para `/entries`.

¡No tiene pérdida!

```
def entry_comments(request, path_param_id):
    if request.method != "GET":
        return JsonResponse({"error": "HTTP method not supported"}, status=405)
    comments = # (?) Escoge tu alternativa preferida
    json_response = []
    for row in comments:
        json_response.append(row.to_json())
    return JsonResponse(json_response, safe=False)
```

Dos últimos pasos

Añadir `to_json` a la clase `Comment` en `models.py`:

```
class Comment(models.Model):
    content = # ...
    entry = # ...

    def to_json(self):
        return {
            "comment": self.content,
        }
```

...y añadir la línea necesaria a `urls.py`:

```
urlpatterns = [  
    # ...  
    path('entries/<int:path_param_id>/comments', endpoints.entry_comments),  
]
```

¡Listo!

Ya podemos obtener los comentarios de una *Entrada*.

De momento no hay ninguno en la base de datos, ¡pero puedes añadir alguno de prueba usando `sqlite3.exe` !

Así, visitando URLs como...

- <http://localhost:8000/entries/1/comments>
- <http://localhost:8000/entries/2/comments>

...verás los datos relevantes.

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