Spring Boot: Github Events API

In this challenge, your task is to implement a simple REST API to manage a collection of GitHub events.

Each event is a JSON entry with the following keys:

- `id` : the unique ID of the event (Integer)
- `type` : the type of the event, written in PascalCase (String)
- `public` : whether the event is public, either true or false (Boolean)
- `repoId` : the ID of the repository the event belongs to (Integer)
- `actorId` : the ID of the user who created the event (Integer)

Here is an example of a trade JSON object:

```
{
      "type": "PushEvent",
      "public": true,
      "repoId": 1,
      "actorId": 1,
}
```

You are provided with the implementation of the Event model. The task is to implement a REST service that exposes the `/events` endpoint, which allows for managing the collection of events records in the following way:

`POST` request to `/events` :

- creates a new event
- expects a JSON event object without an id property as the body payload. You can assume that the given object is always valid.
- adds the given event object to the collection of events and assigns a unique integer id to it. The first created event must have id `1`, the second one `2`, and so on.
- you can assume that payload given to create the object is always valid.
- the response code is `201` and the response body is the created event object, including its id

`GET` request to `/events` :

- returns a collection of all events

- the response code is 200, and the response body is an array of all events ordered by their ids in increasing order

`GET` request to `/repos/{repoId}/events`:

- Returns a collection of events related to the given repository, ordered by their ids in increasing order.
- The response code is 200 if the repository exists, even if there are no events for that repository.
- The response code is 404 if the repository doesn't exist.

`GET` request to `/events/{eventId}`:

- returns an event with the given id
- if the matching event exists, the response code is 200 and the response body is the matching event object
- if there is no event in the collection with the given id, the response code is 404

You should complete the given project so that it passes all the test cases when running the provided *unit* tests. The project by default supports the use of the H2 database. Implement the POST request to `/events` first because testing the other methods requires POST to work correctly.

▼ Example requests and responses

`POST` request to `/events`

Request body:

```
{
          "type": "PushEvent",
          "public": True,
          "repoId": 1,
          "actorId": 1,
    }
```

The response code is 201, and when converted to JSON, the response body is:

```
{
        "id" : 1,
        "type": "PushEvent",
        "public": True,
        "repoId": 1,
        "actorId": 1,
  }
```

This adds a new object to the collection with the given properties and id 1.

`GET` request to `/events`

The response code is 200, and when converted to JSON, the response body (assuming that the below objects are all objects in the collection) is as follows:

```
[
        {
          "id": 1,
          "type": "PushEvent",
          "public": true,
          "repoId": 1,
          "actorId": 1
        },
        {
          "id": 2,
          "type": "ReleaseEvent",
          "public": true,
          "repoId": 1,
          "actorId": 1
        },
        {
          "id": 3,
          "type": "PushEvent",
          "public": true,
          "repoId": 2,
          "actorId": 1
        }
```

```
    ]
```

GET `/repos/1/events`

The response code is 200, and when converted to JSON, the response body (assuming that the below objects are all objects with *repoId* 1) is as follows:

```
[
        {
          "id": 1,
          "type": "PushEvent",
          "public": true,
          "repoId": 1,
          "actorId": 1
        },
        {
          "id": 2,
          "type": "ReleaseEvent",
          "public": true,
          "repoId": 1,
          "actorId": 1
        }
  ]
```

GET `/events/1`**est to**

Assuming that the object with id 1 exists, then the response code is 200 and the response body, when converted to JSON, is as follows:

```
{
    "id": 1,
    "type": "PushEvent",
    "public": true,
    "repoId": 1,
    "actorId": 1
  }
```

If an object with id 1 doesn't exist, then the response code is 404 and there are no particular requirements for the response body.