


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

GET /movies/{movieID}

Resumen

- Crearemos un nuevo método en `MoviesConnector.java`
- Crearemos una nueva clase `GetMovieByID.java`
- Entenderemos qué es un *path param*
- Serviremos en un nuevo *endpoint* `/movies/{movieID}` los datos de la película con `id = {movieID}` de `movies.db`

Descripción

Todos nuestros *endpoints*, hasta ahora, eran **fijos**.

Vamos a empezar a trabajar con *endpoints* que tiene **partes variables** ó *path params*.

En esta tarea conseguiremos que el usuario pueda visitar:

- `http://localhost:8104/movies/2`
- `http://localhost:8104/movies/6`
- etc.

...y obtenga en JSON la información de la película con `id=2` , `id=6` , etc.

Primer paso: Otro método en `MoviesConnector`

Es similar al de la tarea anterior, pero en esa ocasión recibe un **parámetro** tipo `int` . Se prevee que sea el `id` de la película deseada:

```
public Movie getMovieUsingID(int movieID) {
    Movie aMovie = null;
    try {
        Statement statement = this.connection.createStatement();
        String sql = "SELECT * FROM TMovies WHERE id = " + movieID;
        ResultSet result = statement.executeQuery(sql);
        while(result.next()) {
            int id = result.getInt(1);
            String title = result.getString(2);
            int year = result.getInt(3);
            int duration = result.getInt(4);
            String countryIso3166 = result.getString(5);
            String genre = result.getString(6);
            String synopsis = result.getString(7);
            aMovie = new Movie(id, title, year, duration, countryIso3166, genre, synopsis);
        }
        statement.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return aMovie;
}
```

¡Ojo! La `Movie` devuelta será `null` si **no** existe una película con el `int movieID` recibido.

Segundo paso: Un `ServerResource`

Añadamos un nuevo `GetMovieByID.java` .

Para poder interpretar correctamente el *path param* enviado por el cliente HTTP que mencionamos anteriormente:

- `http://localhost:8104/movies/{pathParam}`

...la clave será usar `getAttribute("nombrePathParam")` .

Podemos invocarlo *alegremente* sobre la instancia actual porque **heredamos** de `ServerResource` . Así:

```
public class GetMovieByID extends ServerResource {
    @Get
    public StringRepresentation getEndpointResponse() {
        String movieID = getAttribute("movieID");
        System.out.println("El ID de película visitado es: " + movieID);
        // ...
    }
}
```

Este código ayudará a que cuando visitemos `http://localhost:8104/movies/6`, veamos que IntelliJ IDEA imprime por consola:

```
El ID de película visitado es 6
```

¡Ojo! Si visitásemos `http://localhost:8104/movies/pepeDepura`, también veríamos:

```
El ID de película visitado es pepeDepura
```

Como queremos **ceñirnos a valores numéricos en esta ocasión**, usaremos `Integer.parseInt(...)`:

```
public class GetMovieByID extends ServerResource {
    @Get
    public StringRepresentation getEndpointResponse() {
        String movieID = getAttribute("movieID");
        Integer movieInt = Integer.parseInt(movieID);
        System.out.println("El ID de película visitado es: " + movieInt);
        // ...
    }
}
```

Si nos olvidamos del `print` y escribimos el código completo, sería:

```
@Get
public StringRepresentation getEndpointResponse() {
    String movieID = getAttribute("movieID");
    Integer movieInt = Integer.parseInt(movieID);
    MoviesConnector connector = new MoviesConnector();
    Movie movie = connector.getMovieUsingID(movieInt);
    // Aquí podríamos controlar que el valor no sea nulo
    // De momento, lo dejamos así
    connector.closeConnection();
    String jsonString = movie.toJSONString().toString();
    StringRepresentation representation = new StringRepresentation(jsonString);
    representation.setMediaType(MediaType.APPLICATION_JSON);
    return representation;
}
```

Tercer paso: *Mapear el endpoint*

Lo haremos en `MoviesREST.java` igual que en la tarea anterior.

Pero, **muy importante**, debemos **especificar entre llaves { }** la parte correspondiente al `path param` esperado.

Es decir:

```
host.attach("/movies/{movieID}", GetMovieByID.class);
```

Nótese que `{movieID}` debe ser equivalente al `getAttribute("movieID")`; para que funcione correctamente.

La tarea

Si has seguido los pasos vistos hasta ahora y tu `MoviesREST` sirve el `endpoint /movies/{movieID}`, ¡**enhorabuena!** La tarea ha sido completada.

Por último

Verifica que el `test` funciona correctamente.

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



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