**Top-rated Movie Microservice**

In this exercise, we'll create a Movie-details microservice. We'll add an endpoint/API to this microservice which returns the top rated movie details from the list of given movies across multiple OTT platforms.

At the end of this Hands on exercise, you'll have a running microservice and you'll be able to hit the endpoint and fetch top-rated movie.

The client app hits the API endpoint /movie/top-rated with a GET request when it has to fetch the top-rated movie.

**Step 1.** First up, let's create the required entities. We'll work with 3 entities: Movie, MovieComparator, MovieList.

* Movie entity with movie details - movie name, movie id, OTT Platform, rating
* MovieList entity storing sample movie details. (Mocks DB entries)
* MovieComparator entity used to sort a list of movies.

**MovieEntity:** (Add all getters, setters, all-args-constructor and no-args constructor as well)

**public class** Movie {}

**private** String **movieId**;

**private** String **movieName**;

**private** String **oTTPlatform**;

**private int rating**;

… }

**MovieList:**

**public class** MovieList {

**public** List<Movie> getMovieList() {

Movie mv1 = **new** Movie(**"MV0001"**, **"Pushpa"**, **"Amazon Prime"**, 5);

Movie mv2 = **new** Movie(**"MV0002"**, **"Sholay"**, **"Hotstar"**, 3);

Movie mv3 = **new** Movie(**"MV0005"**, **"Inception"**, **"Amazon Prime"**, 4);

List<Movie> movieList = **new** ArrayList<>(3);

movieList.add(mv1);

movieList.add(mv2);

movieList.add(mv3);

**return** movieList;

}

}

**MovieComparator:** Contains logic to sort movies by rating, highest rating first.

**public class** MovieComparator **implements** Comparator<Movie>{

@Override

**public int** compare(Movie m1, Movie m2) {

**return** m2.getRating() - m1.getRating();

}

}

**Step 2:** Next we’ll define the MovieService. Service uses movieList and movieComparator instances to return top-rated movie.

@Service

**public class** MovieService {

**public** Movie getTopRatedMovie() {

MovieList movieList = **new** MovieList();

MovieComparator mc = **new** MovieComparator();

List<Movie> list = movieList.getMovieList();

Collections.*sort*(list, mc);

**return** list.get(0);

}

}

**Step 3:** Add controller logic in *getTopRatedMovie()*. Controller delegates the call to service and sends the generated response back to the client. Controller has the API endpoint and method type mappings as well.

@RestController

**public class** MovieController {

@Autowired

**private** MovieService **movieService**;

@RequestMapping(value = **"/movie/top-rated"**, method = RequestMethod.***GET***)

**public** Movie getTopRatedMovie() {

**return movieService**.getTopRatedMovie();

}

}

**Step 4:** Update application.yml file under ‘resources’ package to add port number for service

**server**:

**port**: 8088

**Step 5:** Build and run the project to start the service. Hit the endpoint http://<host-url>:8088/movie/top-rated using Postman/browser to view Response

Response:

{

"movieId": "MV0001",

"movieName": "Pushpa",

"oTTPlatform": "Amazon Prime",

"rating": 5

}

That’s it! You got the top rated movie.