

Prerequisites



1. Setup Local DB:

- Please use the SQL Server project named, **MovieInfoDB** to publish and setup the database named, **MovieInfoDB** with some sample data.

You may use the publish file, **MovieInfoDB.publish.xml** that is available in the MovieInfoDB project, to preload the connection string and other related information as shown below

Publish Database MovieInfoDB.publish.xml

Target Database Settings

Target database connection:
Data Source=(localdb)\mssqllocaldb;Integrated Security=True;Persist Security Info=False Edit... Clear

Database name:
MovieInfoDB

Publish script name:
MovieInfo.sql

☐ Register as a Data-tier Application
☐ Block publish when database has drifted from registered version Advanced...

Load Profile...

Save Profile Save Profile As... Generate Script Publish Cancel

- Please change the connection string, **MovieInfoDB_Connection** at appsettings.json file, if needed

3. Build and Run the API

You may choose *MovieInfoApi* instead of IIS Express option at the VS top toolbar, if you wish to continue with the preset port of 5000

API Endpoints

Please note the following endpoints

Base URL:

<http://localhost:5000/api/v1>

API A : To query movie data based on provided filter criteria: title, year of release, genre(s)

1. Movies by the filter criteria – with all 3 params

[{{URL}}](#)/movie?title=The&year=2009&genre=Comedy

2. Movies by the filter criteria - With 2 params

[{{URL}}](#)/movie?year=2009&genre=Comedy

3. Movies by the filter criteria - Partial search string

[{{URL}}](#)/movie?year=2009

API B : To query top n (default 5) movies based on total user rating

1. Top 'n' Movies - default ie top 5

[{{URL}}](#)/movie/top

2. Top 20 (or 'n') Movies

[{{URL}}](#)/movie/top/20

API C : To query top n (default 5) movies based on a certain user's rating

1. Top 10 (or 'n') Movies by User Ratings

[{{URL}}](#)/movie/top/10/1

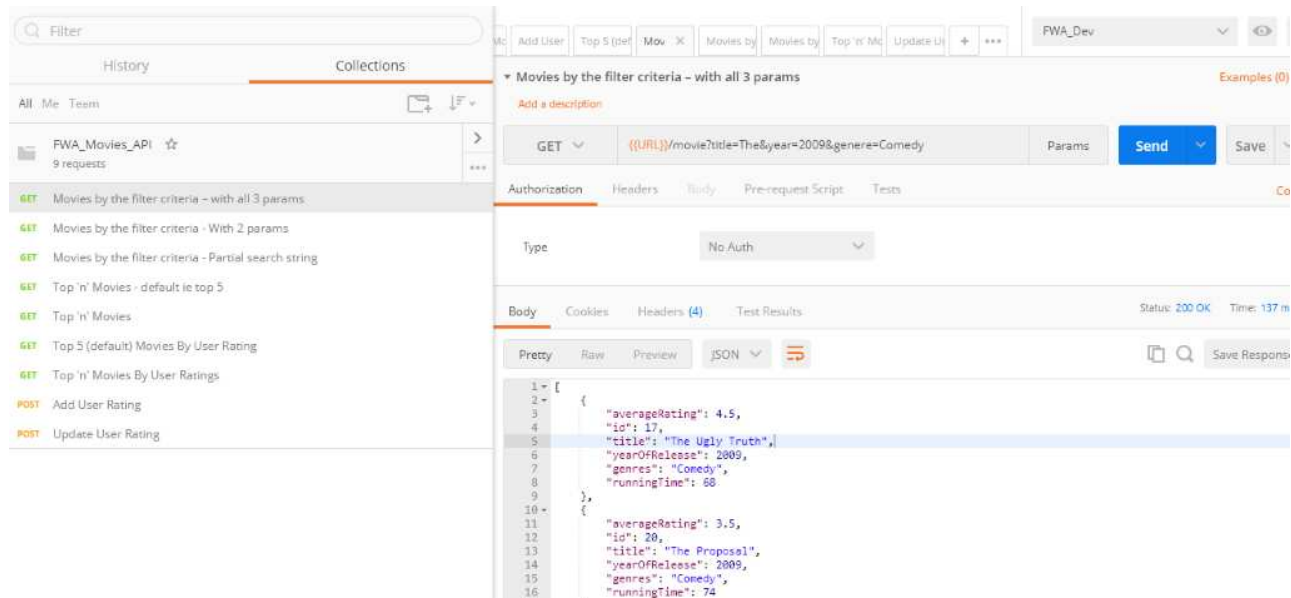
API D : To add or update user rating for a movie

[{{URL}}](#)/movieratings

Postman Data dump

Please use the postman data dump, **FWA_Dev_postman_dump**, that is attached to the project for the environments and some test APIs.

Please make sure to use the environment, **FWA_Dev** to be able to run the test APIs, as shown below



Scope for improvement(s)

API A - To query movie data based on provided filter criteria: title, year of release, genre(s)

Since this API deals with multiple optional query parameters, which has a data limitation, we could send the search criteria in a POST, which would return the Id after save criteria, which in turn would be used to Get the Movies for the given search criteria ID.

Some of the **benefits** in using above method are as follows:

1. Scalable for future needs ie Search Criteria can be extended to more attributes
2. Data would be secured, if any of the search criteria has sensitive information

However, the **disadvantage** is that it involves multiple API calls.

API D - To add or update user rating for a movie

Its highly recommended to avoid POST for the sake of updating a resource as POST is not idempotent but PUT is however for the purpose of this coding assignment, I have used the POST for Add and update.

Exception Handling

No exception handling is included

Logging

Not much logging is included

LINQ vs Lambda expressions

Used mostly LINQ but included one method with Lambda expressions

Unit testing

Due to limited time, I could not include or cover edge cases.

Best practices

I tried to follow the best practices but had to take some exemptions to be able to finish faster.

