# Routing REST functions

##########################################################################

**Purpose**: Its a REST Routing function which can be called as [http://localhost/movies/<year](http://localhost/movies/%3cyear)>. The purpose of the function is to retrieve and display a list of movies released in a specified year.

**Parameters**:

year (str): The release year of the movies to fetch, passed as a URL parameter.

**Process**:

Calls `fetch\_all\_movies` function with the `year` parameter to retrieve the list of movies. Renders the 'movies.html' template with the fetched movies list.

**Returns**:

A rendered HTML page displaying the list of movies for the specified year.

###########################################################################

@app.route('/movies/<year>', methods=['GET'])

def get\_all\_movies(year):

######################################################################

**Purpose -** Its a REST Routing function which can be called as [http://localhost/moviesforyears/<years](http://localhost/moviesforyears/%3cyears)>. The purpose of the function is to retrieve and display a list of movies released in last number of specified years.

**Parameters:**

years (str): A number of years for which movies to be extracted, passed as a URL parameter.

**Process:**

Calls `fetch\_movies\_by\_years` function with the `years` parameter to retrieve the list of movies released in last number of specified years. Renders the 'movies.html' template with the fetched movies list.

**Returns:**

A rendered HTML page displaying the list of movies for the specified years.

######################################################################

@app.route('/moviesforyears/<years>', methods=['GET'])

def get\_movies\_by\_years(years):

###########################################################################

**Purpose:** Its a REST Routing function which can be called as [*http://localhost/performancebyyears/<years*](http://localhost/performancebyyears/%3cyears)*>.* The purpose of the function is to generate and display a performance plot for movies released over specified number of years.

**Parameters:**

years (str): A number of years for which to fetch and analyze movie performance data, passed as a URL parameter.

**Process:**

Calls `plot\_movies\_performance` function with the `years` parameter to generate a plot showing the performance of movies over the specified number of years. Receives the file path of the generated plot.Renders the 'performance.html' template with the plot file path as `plot\_url`.

**Returns:**

A rendered HTML page displaying the performance plot for the specified years.

###########################################################################

@app.route('/performancebyyears/<years>')

def movies\_performance(years):

# Controller Functions for REST

###########################################################################

**Purpose:**

Fetch movie data from the OMDb API based on the given movie title.

**Parameters:**

movie\_title (str): The title of the movie to search for in the OMDb database.

**Process:**

Constructs a request with the movie title and API key as parameters.

Sends a GET request to the OMDb API and receives movie data in JSON format.

**Returns:**

dict: A dictionary containing the movie data retrieved from the OMDb API.

###########################################################################

def fetch\_movie\_data(movie\_title):

###########################################################################

**Purpose:**

Fetch a list of movies released in a specified year from the OMDb API.This function handles paginated results, processes the movie data, and tracks the time taken for the fetch operation.

**Parameters:**

year (str): The year of movie releases to search for, passed as a query parameter.

**Process:**

- Initializes an empty list `movie\_items` to store the fetched movie data.

- Iterates over multiple pages of the OMDb API response using a `while` loop:

- Sends a GET request to the OMDb API with the specified search term (`movie`),

year (`y`), and API key.

- For each page, retrieves the list of movie data (up to 10 movies per page).

- For each movie, extracts key details like title, year, genre, director, actors,

plot, language, country, awards, ratings, and poster URL.

- Appends each movie's data to the `movie\_items` list.

- Tracks the time taken to fetch all the pages.

- Breaks out of the loop if the API response indicates no more data.

**Returns:**

tuple:

- list: A list of dictionaries, each containing movie data for a specific movie.

- float: The total time (in seconds) taken to fetch all the movies.

##########################################################################

def fetch\_all\_movies(year):

###########################################################################

**Purpose:**

Fetch a list of movies released in a specified year from the OMDb API.

**Parameters:**

year (str): The release year of the movies to fetch, passed as an argument.

**Process:**

- Initializes an empty list to store movie data (`movie\_items`) and sets the starting page to 1.

- Defines a search term ("movie") and starts a timer to track request duration.

- In a loop:

- Sends a GET request to the OMDb API with the year, search term, API key, and page number.

- If the response is successful, checks for movies in the "Search" field:

- Extracts movie details (e.g., title, year, genre, director, actors, etc.) and adds them to `movie\_items`.

- Increments the page number to fetch the next set of movies, until no more results are returned.

- Calculates the total time taken for the API requests.

**Returns:**

tuple: A tuple containing:

- movie\_items (list of dict): A list of dictionaries, each with detailed movie information.

- time\_taken (float): The time taken to fetch all movie data, in seconds.

###########################################################################

def fetch\_movies\_by\_years(num\_years):

##########################################################################

**Purpose:**

Generate and save a plot of movie counts and API request times over a range of years.

**Parameters:**

num\_year (int): The number of past years to include in the plot, ending in the current year.

**Process:**

- Converts `num\_year` to an integer and calculates a range of years ending with the current year.

- For each year in the range:

- Calls `fetch\_all\_movies` to retrieve movies released in that year.

- Counts the movies and tracks the time taken to fetch them.

- Appends each year's data (movie count and time taken) to `years\_data`.

- Extracts the years, movie counts, and time taken from `years\_data` for plotting.

- Creates a plot with:

- Movie counts on the left y-axis.

- Time taken (seconds) on the right y-axis.

- Saves the plot as 'rest\_movies\_performance.png' in the static folder.

**Returns:**

str: The file path to the saved plot image.

###########################################################################

def plot\_movies\_performance(num\_year):

###########################################################################

**Purpose:**

Generate and save a plot of movie counts and API request times over a range of years.

**Parameters:**

num\_year (int): The number of past years to include in the plot, ending in the current year.

**Process:**

- Converts `num\_year` to an integer and calculates a range of years ending with the current year.

- For each year in the range:

- Calls `fetch\_all\_movies` to retrieve movies released in that year.

- Counts the movies and tracks the time taken to fetch them.

- Appends each year's data (movie count and time taken) to `years\_data`.

- Extracts the years, movie counts, and time taken from `years\_data` for plotting.

- Creates a plot with:

- Movie counts on the left y-axis.

- Time taken (seconds) on the right y-axis.

- Saves the plot as 'rest\_movies\_performance.png' in the static folder.

**Returns:**

str: The file path to the saved plot image.

###########################################################################

def fetch\_movies\_by\_year(year, max\_pages=10):

# Routing GraphQL Functions

###########################################################################

**Purpose:**

This is a GraphQL endpoint that accepts POST requests containing

  GraphQL queries. It processes the queries using the GraphQL schema and

  returns the results as a JSON response.

**Parameters:**

  None (the input comes from the request body as JSON data).

**Process:**

  - Extracts the JSON data from the incoming POST request.

  - Passes the JSON data to the `graphql\_sync` function, along with:

      - The GraphQL schema to validate and resolve the query.

      - The request context (`request`) to be passed to the resolvers.

      - The `debug` flag to help with debugging during development.

  - Based on the success of the GraphQL query execution, determines the

    appropriate HTTP status code (200 for success, 400 for failure).

  - Returns the result of the GraphQL query execution as a JSON response

    with the corresponding HTTP status code.

**Returns:**

  tuple:

    - JSON response: The result of the executed GraphQL query.

    - int: HTTP status code (200 for success, 400 for failure).

###########################################################################

@app.route("/graphql", methods=["POST"])

###########################################################################

**Purpose:**

  This is a GraphQL Playground endpoint that returns a simple HTML page

  allowing developers to interactively explore and test GraphQL queries.

  It serves the GraphQL Playground UI on a GET request to the "/graphql"

  URL.

**Parameters:**

  None (the function responds to GET requests to the "/graphql" endpoint).

**Process:**

  - Responds with the HTML content of the GraphQL Playground UI, allowing

    users to interact with the GraphQL schema and make queries.

  - Sets the response's content type to 'text/html' to indicate that the

    response is an HTML document.

**Returns:**

  tuple:

    - HTML content: The GraphQL Playground interface as an HTML string.

    - int: HTTP status code (200 for successful response).

    - dict: Response header indicating the content type is 'text/html'.

###########################################################################

@app.route("/graphql", methods=["GET"])

###########################################################################

**Purpose:**

  This function handles the `/showmovies` route, which executes a

  GraphQL query to fetch a list of movies and renders the data in an HTML

  page. It uses the GraphQL query 'allMovies' to fetch a list of movies

  from the GraphQL schema and displays the results on a webpage.

**Parameters:**

  None (the function handles GET requests to the `/showmovies` route).

**Process:**

  - Constructs a GraphQL query to fetch movie data from the server.

  - Executes the GraphQL query using `graphql\_sync` to fetch the movie data.

  - If the query is successful:

      - Extracts the movie data from the query result.

      - Converts the movie data into a pandas DataFrame for easier handling.

      - Renders the `movies.html` template with the fetched movie data.

  - If the query fails, returns an error message in JSON format with a

    400 status code.

**Returns:**

  - If successful: A rendered HTML page displaying the fetched movie data

    using the `movies.html` template.

  - If the query fails: A JSON response with an error message and a 400

    status code.

###########################################################################

@app.route('/showmovies')

def showmovies():

1. Util Common Functions
2. GraphQL Schema
3. Resources