Backend 2.6_CW Exercises

ex01: creating a movie API

challenge

Create a POST route at `/movies` that accepts JSON data with movie details. Use the `createMovie` function to save the movie data. Handle errors too.

1. Create a POST Route: Set up a POST route at `/movies`.
2. Handle the Request: In the route handler, use the `createMovie` function to save the movie data received in the request body.
3. Success Response: If the movie is successfully added, respond with a success message and the saved movie details.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
POST /movies

Request Body:
},
  {
    "title": "Bahubali 2",
    "releaseYear": 2018,
    "genre": ["Action", "Fantasy"],
    "director": "S. S. Rajamouli",
    "actors": ["Prabhas", "Anushka Shetty"],
    "language": "Telugu",
    "country": "India",
    "rating": 8.1,
    "plot": "A man embarks on a journey to rescue his mother from a tyrant.",
    "awards": "National Film Award",
    "posterUrl": "https://example.com/poster2.jpg",
    "trailerUrl": "https://example.com/trailer2.mp4"
  },
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex01

```
app.post('/movies', async (req, res) => {
  try {
    const savedMovie = await createMovie(req.body)
    res.status(201).json({ message: 'Movie added', movie: savedMovie })
  } catch (error) {
    res.status(500).json({ error: 'Failed to add movie' })
  }
})
```

COPY

ex02: reading a movie API

challenge

Set up a GET route at `/movies/:title` that allows users to retrieve movie details by providing the movie title as a route parameter. Utilize the `readMovie` function to fetch the movie data. Don't forget error handling!

1. Create a GET Route: Define a GET route at `/movies/:title` to handle requests for specific movies.
2. Retrieve Movie Data: In the route handler, use the `readMovie` function to find the movie with the provided title.
3. Success Response: If the movie is found, respond with the movie details.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
GET /movies/My%20Awesome%20Movie
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex02

```
app.get('/movies/:title', async (req, res) => {
  try {
    const movie = await readMovie(req.params.title)
    if (movie) {
      res.json(movie)
    } else {
      res.status(404).json({ error: 'Movie not found' })
    }
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movie' })
  }
})
```

COPY

ex03: reading all movies API

challenge

Create a GET route at `/movies` to fetch all movie details. Utilize the `readAllMovies` function to retrieve the movie data. Handle any potential errors gracefully.

1. Create a GET Route: Define a GET route at `/movies` to handle requests for all movies.
2. Retrieve All Movies: In the route handler, use the `readAllMovies` function to retrieve all movie data.
3. Success Response: If movies are found, respond with an array containing all movie details.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
GET /movies
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex03

```
app.get('/movies', async (req, res) => {
  try {
    const allMovies = await readAllMovies()
    res.json(allMovies)
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movies' })
  }
})
```

COPY

ex04: reading movies by Actor API

challenge

Set up a GET route at `/movies/actor/:actorName` that allows users to retrieve movies featuring a particular actor. Utilize the `readMoviesByActor` function to fetch the relevant movie data. Handle errors effectively.

1. Create a GET Route: Define a GET route at `/movies/actor/:actorName` to handle requests for movies featuring a specific actor.
2. Retrieve Movies: In the route handler, use the `readMoviesByActor` function to find movies featuring the provided actor's name.
3. Success Response: If movies are found, respond with an array containing the details of those movies.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
GET /movies/actor/John%20Doe
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex04

```
app.get('/movies/actor/:actorName', async (req, res) => {
  try {
    const movies = await readMoviesByActor(req.params.actorName)
    res.json(movies)
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movies' })
  }
})
```

COPY

ex05: reading movies by director API

In this exercise, you'll create a GET request API to retrieve movies directed by a specific director using the `readMoviesByDirector` function. Let's dive into the task!

challenge

Create a GET route at `/movies/director/:directorName` that allows users to retrieve movies directed by a particular director. Utilize the `readMoviesByDirector` function to fetch the relevant movie data. Handle errors gracefully.

1. Create a GET Route: Define a GET route at `/movies/director/:directorName` to handle requests for movies directed by a specific director.
2. Retrieve Movies: In the route handler, use the `readMoviesByDirector` function to find movies directed by the provided director's name.
3. Success Response: If movies are found, respond with an array containing the details of those movies.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
GET /movies/director/Director%20Name
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex05

```
app.get('/movies/director/:directorName', async (req, res) => {
  try {
    const movies = await readMoviesByDirector(req.params.directorName)
    res.json(movies)
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movies' })
  }
})
```

COPY

ex06: reading movies by genre API

challenge

Develop a GET route at `/movies/genre/:genreName` that enables users to access movies belonging to a particular genre. Utilize the `readMoviesByGenre` function to retrieve the relevant movie data. Handle errors efficiently.

1. Create a GET Route: Set up a GET route at `/movies/genre/:genreName` to handle requests for movies of a specific genre.
2. Retrieve Movies: In the route handler, employ the `readMoviesByGenre` function to find movies in the provided genre.
3. Success Response: If movies are found, respond with an array containing the details of those movies.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
GET /movies/genre/Action
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex06

```
app.get('/movies/genre/:genreName', async (req, res) => {
  try {
    const movies = await readMoviesByGenre(req.params.genreName)
    res.json(movies)
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movies' })
  }
})
```

COPY

ex07: updating a movie API

challenge

Develop a POST route at `/movies/:movieId` that allows users to update a movie's details. Utilize the `updateMovie` function to make the necessary changes. Handle errors effectively.

1. Create a POST Route: Design a POST route at `/movies/:movieId` to handle requests for updating movie details.
2. Update Movie: In the route handler, use the `updateMovie` function to modify the specified movie using its ID and the updated data.
3. Success Response: If the movie is successfully updated, respond with the updated movie details.
4. Error Handling: If an error arises during the process, respond with an error message.

example

```
POST /movies/617a2ed9466e8c00160192e0

Request Body:
{
  "title": "Updated Movie Title",
  "releaseYear": 2023
}
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex07

```
app.post('/movies/:movieId', async (req, res) => {
  try {
    const updatedMovie = await updateMovie(req.params.movieId, req.body)
    if (updatedMovie) {
      res.json(updatedMovie)
    } else {
      res.status(404).json({ error: 'Movie not found' })
    }
  } catch (error) {
    res.status(500).json({ error: 'Failed to update movie' })
  }
})
```

COPY

ex08: deleting a movie API

challenge

Craft a DELETE route at `/movies/:movieId` that enables users to delete a movie. Utilize the `deleteMovie` function to perform the deletion. Handle errors gracefully.

1. Create a DELETE Route: Establish a DELETE route at `/movies/:movieId` to manage requests for deleting a movie.
2. Delete Movie: In the route handler, use the `deleteMovie` function to remove the specified movie using its ID.
3. Success Response: If the movie is successfully deleted, respond with a success message.
4. Error Handling: If an error arises during the process, respond with an error message.

example

```
DELETE /movies/617a2ed9466e8c00160192e0
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex08

```
app.delete('/movies/:movieId', async (req, res) => {
  try {
    const deletedMovie = await deleteMovie(req.params.movieId)
    if (deletedMovie) {
      res.json({ message: 'Movie deleted', movie: deletedMovie })
    } else {
      res.status(404).json({ error: 'Movie not found' })
    }
  } catch (error) {
    res.status(500).json({ error: 'Failed to delete movie' })
  }
})
```

COPY

ex09: reading movies by rating API

challenge

Develop a GET route at `/movies/ratings` that allows users to access movies sorted by their ratings in descending order. Utilize the `readMoviesByRating` function to retrieve the movie data. Handle errors effectively.

1. Create a GET Route: Set up a GET route at `/movies/ratings` to handle requests for movies sorted by ratings.
2. Retrieve Movies: In the route handler, use the `readMoviesByRating` function to find and sort movies by their ratings in descending order.
3. Success Response: If movies are found, respond with an array containing the details of those movies.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
GET /movies/ratings
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex09

```
app.get('/movies/ratings', async (req, res) => {
  try {
    const movies = await readMoviesByRating()
    res.json(movies)
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movies' })
  }
})
```

COPY

ex10: reading movies by release year API

challenge

Design a GET route at `/movies/release-years` that allows users to access movies sorted by their release years in ascending order. Utilize the `readMoviesByReleaseYear` function to retrieve the movie data. Handle errors proficiently.

1. Create a GET Route: Set up a GET route at `/movies/release-years` to handle requests for movies sorted by release years.
2. Retrieve Movies: In the route handler, use the `readMoviesByReleaseYear` function to find and sort movies by their release years in ascending order.
3. Success Response: If movies are found, respond with an array containing the details of those movies.
4. Error Handling: If an error arises during the process, respond with an error message.

example

```
GET /movies/release-years
```

COPY

solution

https://replit.com/@tanaypratap/BE26CW-ex10

```
app.get('/movies/release-years', async (req, res) => {
  try {
    const movies = await readMoviesByReleaseYear()
    res.json(movies)
  } catch (error) {
    res.status(500).json({ error: 'Failed to fetch movies' })
  }
})
```