Adding the search capabilities on products and middleware for validation

Relevel
by Unacademy

Class Agenda

- We will be going to establish the relationship between Category and Product resource.
- We will then add custom middlewares for the request validations.
- And we will also implement the filter query parameters in the REST APIs.

Educator Introduction

Understanding the relationships in the world of RDBMS

Different types of relationships :

- One to One - For ex : Citizen <-> Aadhar
- One to Many - For ex : Father <-> Children
- Many to One - For ex : Children <-> Father
- Many to Many - For ex : Teacher <-> Student

Understanding the relationships in the world of RDBMS

Similarly, Categories —> Products (One to Many) One category can have many products associated with it

And this is how we establish this relationship using Sequelize:

```
Category.hasMany(Product); // This will create a foreign key column( categoryId) in Products table
```

Above is the code snippet from the server.js file :
https://github.com/Vishwa07dev/eCommerce/blob/session4/server.js

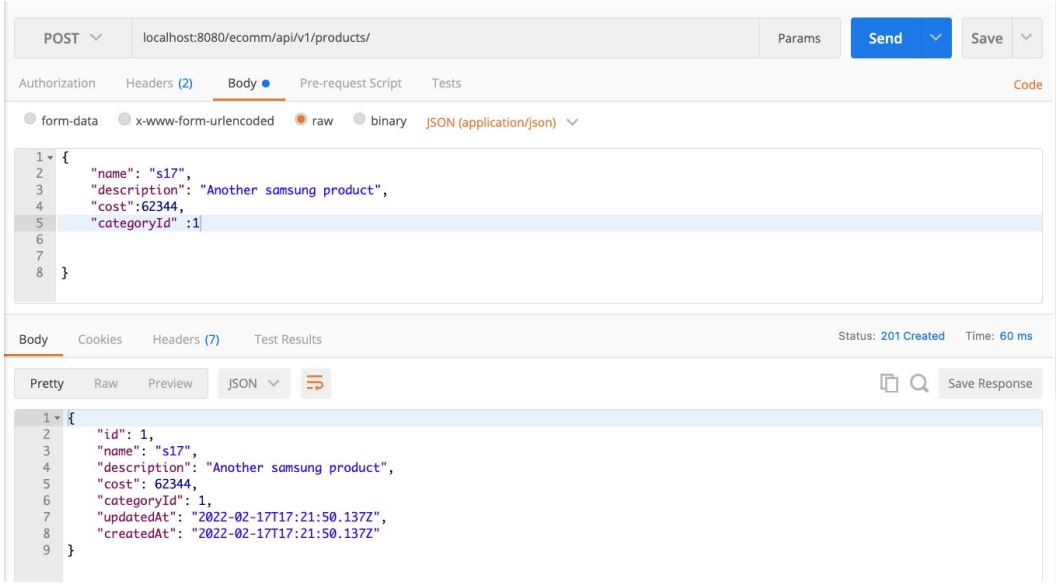
Impact of this ?

- Before creating any Product, we need to specify the Category id to it.
- Newly created Product schema will look like below :

Field	Type	Null	Key	Default
name	varchar(255)	NO		NULL
description	varchar(255)	YES		NULL
cost	int	NO		NULL
createdAt	datetime	NO		NULL
updatedAt	datetime	NO		NULL
categoryId	int	YES	MUL	NULL

Impact of this ?

This will also have the impact on the request body for REST APIs, where, we need to pass an extra field ‘categoryId’ every time.



Validations

Why do we need validations ?

- To ensure the request sent by the client is correct
- To ensure the request sent by the client is complete
- To ensure, client is provided enough information when a request fails
- For the better handling of the boundary and error conditions

Validations that we need to use in our REST API

1. Ensure Client do not pass the request body with empty name for Creating a new Category

Solution :

- Create a middleware which will validate the request body, before it’s being handled by the controller layer. In the case of the validation error, should return response to the user about the effective cause
- Code snippet is from the file : https://github.com/Vishwa07dev/eCommerce/blob/session4/middlewares/requestValidator.js

1. Ensure Client do not pass the request body with empty name for Creating a new Category

```
const validateCategoryRequest = (req, res, next) =>{

    if (!req.body.name) {
        res.status(400).send({
            message: "Name of the category can't be empty !"
        })
        return;
    }
    next();
}
```

2. Ensure for the product request :

- Product name passed is not null
- Product have a category Id passed to it all the time

Solution :

- Create a middleware which will validate the request body, before it’s being handled by the controller layer. In the case of the validation error, should return response to the user about the effective cause

Code snippet is from the file :
https://github.com/Vishwa07dev/eCommerce/blob/session4/middlewares/requestValidator.js

2. Ensure for the product request :

```
const validateProductRequest = (req, res, next) => {

   /**
    * Validation of the request body
    */
   if (!req.body.name) {
       res.status(400).send({
           message: "Name of the product can't be empty !"
       })
       return;
   }

   if (req.body.categoryId) {
       //Check if the category exists, if not return the proper error
message
       Category.findByPk(req.body.categoryId).then(category => {
           if (!category) {
               res.status(400).send({
                   message: `category id passed is not available :
${req.body.categoryId}`
               });
               return;
           }
           next();
       }).catch(err => {
           res.status(500).send({
               message: "Some Internal error while storing the product!"
           });
       });
   } else {
       res.status(400).send({
           message: `category id passed is not available `
       })

       return;
   }
```

Making use of these Validator middleware

Making use of these Validator middleware

Add these validators in the route files

Category.route.js :
https://github.com/Vishwa07dev/eCommerce/blob/session4/routes/category.routes.js

Making use of these Validator middleware

```
/**
* This file will contain the routes logic for the Category resource
* and will export it.
*/
const { requestValidator } = require("../middlewares");

const categoryController = require("../controllers/category.controller")

module.exports = function(app){

   //Route for the POST request to create the category

app.post("/ecomm/api/v1/categories",[requestValidator.validateCategoryReque
st], categoryController.create);

   //Route for the GET request to fetch all the categories
   app.get("/ecomm/api/v1/categories", categoryController.findAll);

   //Route for the GET request to fetch a category based on the id
   app.get("/ecomm/api/v1/categories/:id", categoryController.findOne);

   //Route for the PUT request to update a category based on the id

app.put("/ecomm/api/v1/categories/:id",[requestValidator.validateCategoryRe
quest] ,categoryController.update);

   //Route for the DELETE request to delete a category based on the id
   app.delete("/ecomm/api/v1/categories/:id", categoryController.delete);
}
```

Making use of these Validator middleware

Similarly we need to add the middleware in the Product routes

https://github.com/Vishwa07dev/eCommerce/blob/session4/routes/product.routes.js

```
//Route for the POST request to create the product

app.post("/ecomm/api/v1/products",[requestValidator.validateProductRequest]
, productController.create);
```

Enhancing the capabilities of existing Product REST APIs

1. Support to filter the products based on the cost. Ability to set the minimum and maximum cost filters

Changes in Product controller :

https://github.com/Vishwa07dev/eCommerce/blob/session4/controllers/product.controller.js

1. Support to filter the products based on the cost. Ability to set the minimum and maximum cost filters

```
exports.findAll = (req, res) => {

  //Supporting the query param
  let productName = req.query.name;
  let minCost = req.query.minCost;
  let maxCost = req.query.maxCost;
  let promise;
  if (productName) {
      promise = Product.findAll({
          where: {
              name: productName
          }
      });
  } else if (minCost && maxCost) {

      promise = Product.findAll({
          where: {
              cost: {
                  [Op.gte]: minCost,
                  [Op.lte]: maxCost
              }
          }
      });
```

```
    }else if(minCost){
        promise = Product.findAll({
            where: {
                cost: {
                    [Op.gte]: minCost
                }
            }
        });
    }else if(maxCost){
        promise = Product.findAll({
            where: {
                cost: {
                    [Op.lte]: maxCost
                }
            }
        });
    }
    else {
        promise = Product.findAll();
    }
    promise.then(products => {
        res.status(200).send(products);
    }).catch(err => {
        res.status(500).send({
            message: "Some Internal error while fetching all the
products"
        })
    })
}
```

Supported APIs because of the above change :

- GET localhost:8080/ecomm/api/v1/products?minCost=80000
- GET localhost:8080/ecomm/api/v1/products?maxCost=80000
- GET localhost:8080/ecomm/api/v1/products?minCost=60000&maxCost=80000

2. Support to find the products under a given category

Changes in Products controller :

https://github.com/Vishwa07dev/eCommerce/blob/session4/controllers/product.controller.js

```
/**
    * Get the list of all the products under a category
    */
exports.getProductsUnderCategory = (req, res) => {
    const categoryId = parseInt(req.params.categoryId);

    Product.findAll({
        where: {
            categoryId: categoryId
        }
    }).then(products => {
        res.status(200).send(products);
    }).catch(err => {
        res.status(500).send({
            message: "Some Internal error while fetching  products based on
the category id "
        })
    })

}
```

2. Support to find the products under a given category

Changes in the products router file

https://github.com/Vishwa07dev/eCommerce/blob/session4/routes/product.routes.js

```
    //Route for getting the list of products with cost greater than the

app.get("/ecomm/api/v1/categories/:categoryId/products",[requestValidator.v
alidateCategoryPassedInReqParam],
productController.getProductsUnderCategory);

    //Route for getting the list of products under a category

app.get("/ecomm/api/v1/categories/:categoryId/products",[requestValidator.v
alidateCategoryPassedInReqParam],
productController.getProductsUnderCategory);
```

MCQs

1. Which of the following class is used to create and consume custom events in Node.js?

A. EventEmitter
B. Events
C. NodeEvent
D. None of the Above

MCQs

1. Which of the following class is used to create and consume custom events in Node.js?

A. EventEmitter
B. Events
C. NodeEvent
D. None of the Above

Answer: A

MCQs

2. Which of the following is true about console global object?

A. There are built-in methods to be used for printing informational, warning and error messages in console.
B. console is used in synchronous way when destination is file or a terminal.
C. console is used in asynchronous way when destination is a pipe.
D. All of the above.

MCQs

2. Which of the following is true about console global object?

A. There are built-in methods to be used for printing informational, warning and error messages in console.
B. console is used in synchronous way when destination is file or a terminal.
C. console is used in asynchronous way when destination is a pipe.
D. All of the above.

Answer: D

MCQs

3. Which of the following core module is used to create a web server in Node.js?

A. fs
B. url
C. connect
D. http

MCQs

3. Which of the following core module is used to create a web server in Node.js?

A. fs
B. url
C. connect
D. http

Answer: D

MCQs

4. Which of the following types of application can be built using Node.js?

A. Web Application
B. Chat Application
C. RESTful Service
D. All of the Above

MCQs

4. Which of the following types of application can be built using Node.js?

A. Web Application
B. Chat Application
C. RESTful Service
D. All of the Above

Answer: D

MCQs

5. Which module is used to serve static resources in Node.js?

A. node-static
B. http
C. node-resource
D. static

MCQs

5. Which module is used to serve static resources in Node.js?

A. node-static
B. http
C. node-resource
D. static

Answer: A

Practice Problem:

1. In our mini project built till session3 let’s add custom filters as mentioned below
    - Get books with given title
    - Get books after given published date
2. Also create a middleware to validate request instead and check for empty title here

Next session

- What is Authentication
- What is Authorization
- Basic Auth
- Token Based Authentication
- Create a signup API
- Create a login API which return the access token

Thank you!