EX4 Sequelize Model Querying

ניצור את הקונטרולר אך הפעם - נתייחס להדגשים ונעבור שלב אחרי שלב - על מנת שנבין את הקוד
תחת תקיית הקונטרלר ניצור קובץ bookController.js

ונייבא אליו את אובייקט מסד הנתונים


```
const db = require('../models/index')
```

כפי שניתן לראות בקובץ הINDEX שאנו מייבאים אובייקט הDB מכיל את המודלים

```
models > index.js > <unknown>
const {Sequelize} = require('sequelize');
const {sequelize} = require('./sequelize');
const { applyExtraSetup } = require('./extra-setup');

const db = {}

db.Sequelize = Sequelize
db.sequelize = sequelize
//******************MODELS************************//
db.notes  = require('./note')
db.categories  = require('./category')
db.statuses  = require('./status')
db.areas  = require('./area')
db.authors  = require('./author')
db.books  = require('./book')
//******************END MODELS******************//
applyExtraSetup();

db.sequelize.sync({ alter: true })
.then(() => {
    console.log('yes re-sync done!')
})
module.exports = db
```

ולכן בשביל לקרוא לאובייקט

```
const Book = db.books;
```

היות שאין לנו כרגע ספרים במסד הנתונים, הפונקציה הראשונה שנבצע זה יצירת ספר/
ניצור את הפונקציה של create ונייצא אותה

```
const create = async (req, res) =>{

}

module.exports = {create}
```

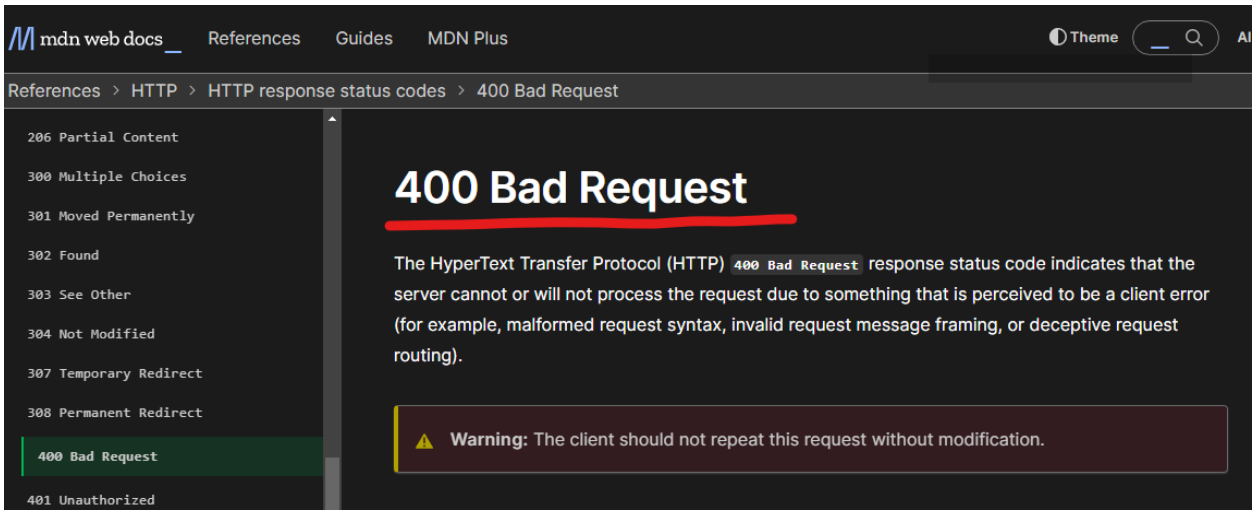
בתוך הפונקציה נקבל את כל השדות מתוך הBODY של הREQUEST

```
const { name , picture, cateogry_id, author_id } = req.body
```

נבצע ואלידציה על מנת שנוכל לשלוח הודעה יותר ידידותית למשתמש,
לדוגמא שדה הNAME הוא חובה ואם הוא לא יהיה קיים תתקבל שגיאה במהלך הכנסת הנתונים כי הגדרנו את השדה שלא יאפשר NULL.
אך אז מתקבלת שגיאה לא ידידותית למשתמש לכן נבצע ואלידציה לשלוח הודעה למשתמש מה מקור השגיאה

```
if (!name || !cateogry_id || !author_id) {// Confirm data
  return res.status(400).json({ message: 'All fields are required' })
}
```

שגיאת 400 היא שגיאת נתונים שגויים



MODEL.CREATE

ניצור את הספר על ידי קריאת לפונקציה CREATE של אובייקט המודל

```
 const book = await Book.create({  name , picture, cateogry_id, author_id
                                                                      ({
```

הנכון להכניס לTRY CATCH אך אנו נפתור את הנושא באופן גלובלי לכל המערכת

אם חזר לנו האובייקט אז פעולת ההכנסה הצליחה ואם לא הייתה שגיאה במהלך הכנסת הנתונים

```
    if (book) { // Created
        return res.status(201).json({ message: 'New book created' })
    } else {
        return res.status(400).json({ message: 'Invalid book data
received' })
    }
```

אפשר אם רוצים להחזיר את האובייקט אם הצליח

```
return res.status(201).json({ message: 'New book created',data:book })
```

ניצור את הROTUE לפונקציה זו על מנת שנוכל להכניס נתונים דרך המערכת
ניצור קובץ bookRoutes.js תחת ROUTES

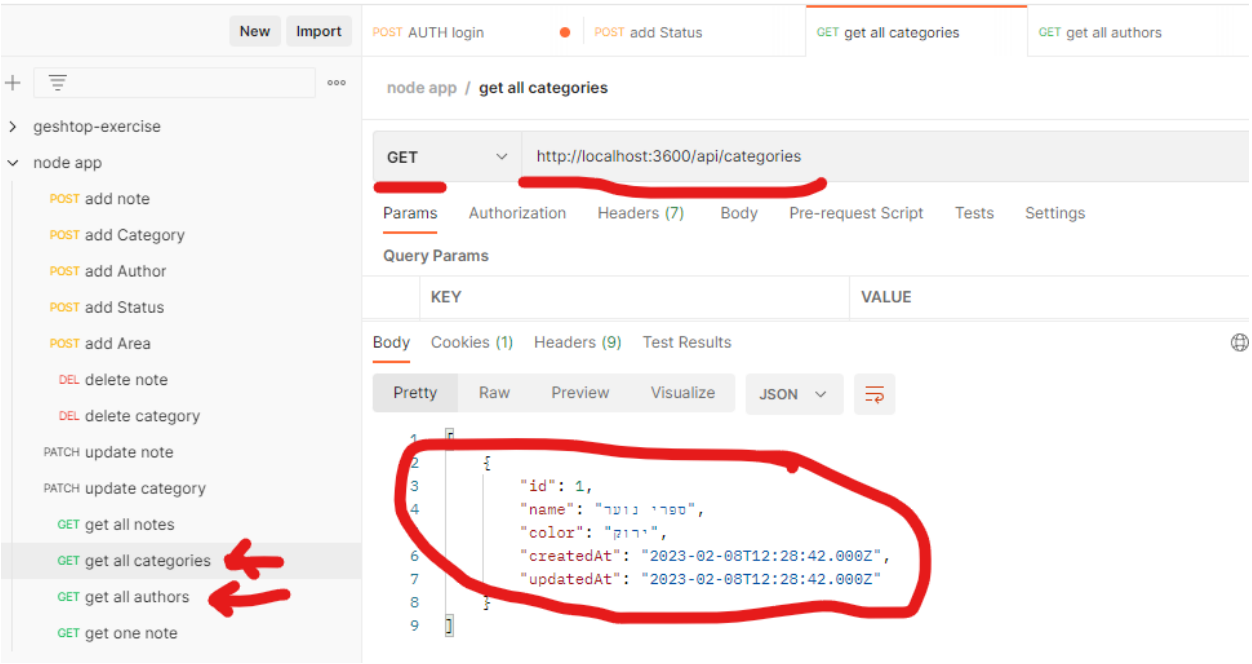
```
const express = require('express')
const router = express.Router()
const bookController = require("../controllers/bookController")

router.route('/')
        .post(bookController.create)
//router.post('/', bookController.create)//SAME
module.exports = router
```

נוסיף את ההפניה לקובץ השרת שלנו server.js

```
app.use("/api/books", require("./routes/bookRoutes"));
```

ונבדוק באמצעות POSTMAN
לאחר שהוספנו מחבר וקטגוריה (גם דרך הPOSTMAN) נבדוק שאכן קיימים נתונים במערכת



עכשיו נוסיף את הספר - ונבדוק שאכן מתקבל פרטי הספר כמו שהגדרנו

POST http://localhost:3600/api/books

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "name" :"הכל למרות",
    "picture":"",
    "cateogry_id" : 1,
    "author_id":1
}
```

נשלח

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "message": "New book created",
    "data": {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "cateogry_id": 1,
        "author_id": 1,
        "updatedAt": "2023-02-08T12:38:32.713Z",
        "createdAt": "2023-02-08T12:38:32.713Z"
    }
}
```

התקבל

MODEL.FINDALL

נוסיף פונקציה לקבלת כל הספרים ונייצא גם אותה

```
const getAll = async (req, res) => {
    const books = await Book.findAll({})
    if (!books?.length) {
        return res.status(400).json({ message: 'No books found' })
    }
    res.json(books)
}
```

```
module.exports = {create, getAll}
```

אנו משתמשים בפונקצית findAll של האובייקט

נוסיף גם את הROUTE הרלוונטי

```
router.get('/', bookController.getAll)
```

מבדיקה בPOSTMAN מתקבלים כל הספרים

node app / **get all books**

GET http://localhost:3600/api/books

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
1  [
2      {
3          "id": 1,
4          "name": "למרות הכל",
5          "picture": "",
6          "cateogry_id": 1,
7          "author_id": 1,
8          "createdAt": "2023-02-08T12:38:32.000Z",
9          "updatedAt": "2023-02-08T12:38:32.000Z"
10     },
11     {
12         "id": 2,
13         "name": " אין בא מן העבר",
14         "picture": "",
15         "cateogry_id": 1,
16         "author_id": 2,
17         "createdAt": "2023-02-08T12:53:45.000Z",
18         "updatedAt": "2023-02-08T12:53:45.000Z"
19     }
20 ]
```


attributes

אם אנו לא מעוניינים בכל השדות אלא להגדיר שדות ספציפיים (אנו מעוניינים רק בID ובשם הספר והתמונה)

נשתמש בattributes

```
const books = await Book.findAll({
    attributes:['id','name','picture']
})
```

עכשיו אם נריץ נראה שמתקבלים רק השדות שציינו

GET http://localhost:3600/api/books

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": ""
    },
    {
        "id": 2,
        "name": " איש בא מן העבר",
        "picture": ""
    }
]
```


יש אפשרות גם להגדיר שלא מעוניינים בשדות מסויימים (לדוגמא לא להעביר את שדה הסיסמא של המשתמש)

```
        attributes: {
            exclude: ['createdAt', 'updatedAt']
        },
```

כמובן שנרצה את שם המחבר
אם נשים לב כאשר הגדרנו את הקשרים בין הטבלאות הגדרנו גם שם לקשר

JS bookController.js | JS extra-setup.js ● | JS bookRoutes.js | JS server.js ex3 | JS server.js e

ex4 > models > JS extra-setup.js > applyExtraSetup

```js
const { sequelize } = require("./sequelize");

const applyExtraSetup = () => {
  const { book, author, category } = sequelize.models;
  book.belongsTo(category, { foreignKey: "cateogry_id", as: "category" });
  book.belongsTo(author, { foreignKey: "author_id", as: "author" });
  author.hasMany(book, { foreignKey: "author_id", as: "books" });
  category.hasMany(book, { foreignKey: "cateogry_id", as: "books" });
};

module.exports = { applyExtraSetup };
```

include

נוסיף את המחבר

```js
const books = await Book.findAll({
    attributes:['id','name','picture'],
    include: 'author'
})
```

GET http://localhost:3600/api/books

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "author": {
            "id": 1,
            "name": "שמואל ארגמן",
            "createdAt": "2023-02-08T12:32:18.000Z",
            "updatedAt": "2023-02-08T12:32:18.000Z"
        }
    },
    {
        "id": 2,
        "name": " איש בא מן העבר",
        "picture": "",
        "author": {
            "id": 2,
            "name": " דבורה עוף",
            "createdAt": "2023-02-08T12:52:39.000Z",
            "updatedAt": "2023-02-08T12:52:39.000Z"
        }
    }
]
```


אפשר גם להביא שדות ספציפים מטבלת הקשר

בשביל זה נוסיף גם את האובייקט אותו אנו מעוניינים לקשר

```
const Author = db.authors;
```

ובקריאה נבחר רק את שדות הID והNAME מטבלת המחברים

```
const books = await Book.findAll({
  attributes:['id','name','picture'],
  include  : [{ model: Author, as: 'author', attributes:['id','name']}]
```

```
})
```

והתוצאה

GET http://localhost:3600/api/books

Params Authorization Headers (7) Body Pre-request Script T

Query Params

KEY	VALUE

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "author": {
            "id": 1,
            "name": "שמואל ארגמן"
        }
    },
    {
        "id": 2,
        "name": " איש בא מן העבר",
        "picture": "",
        "author": {
            "id": 2,
            "name": " דבורה עוף"
        }
    }
]
```


נבצע את אותו הדבר גם לקטגוריות

```
const books = await Book.findAll({
    attributes:['id','name','picture'],
    include     : [
        { model: Author, as: 'author', attributes:['id','name']},
        { model: Category, as: 'category', attributes:['id','name']}
    ]
})
```

והתוצאה

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "author": {
            "id": 1,
            "name": "שמואל ארגמן"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    },
    {
        "id": 2,
        "name": " איש בא מן העבר",
        "picture": "",
        "author": {
            "id": 2,
            "name": " דבורה עוף"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    }
]
```


where

אם קיים לנו דף לדוגמא סינון הספרים לפי קטגוריה מחבר או שם ספר

קטגוריה ▽ | מחבר ▽ | שם ספר

ניצור בקונטרולר פונקציה נוספת של חיפוש, שמקבלת את המשתנים QUERY ונייצא את הפונקציה

```
const search = async (req, res) => {
    const { cateogry_id, author_id, q } = req.query
    res.json({})
}
```

```
module.exports = {create, getAll, search}
```

יש להבדיל בין req.body, req.params, req.query

request.params

Suppose you have defined your route name like this:

```
https://localhost:3000/user/:userId
```

which will become:

```
https://localhost:3000/user/5896544
```

Here, if you will print: **request.params**

```
{
userId : 5896544
}
```

so

```
request.params.userId = 5896544
```

so **request.params** is an object containing properties to the named route

---

request.query

The **request.query** comes from query parameters in the URL eg:

```
https://localhost:3000/user?userId=5896544
```

request.query

```
{

userId: 5896544

}
```

בגלל שהנתונים שלנו הם **אופציונאליים** ייתכן לחפש לפי שם ייתכן לפי המחבר וייתכן לפי שלושת האפשרויות לכן נכון יותר להשתמש בquery בשונה מparams

בשלב ראשון נצא מנקודת הנחה שכל הנתונים קיימים ואנו רוצים לחפש לפי שלושתם

```
const search = async (req, res) => {
    const { cateogry_id, author_id, q } = req.query

    const books = await Book.findAll({
        attributes:['id','name','picture'],
        include    : [
            { model: Author, as: 'author', attributes:['id','name']},
            { model: Category, as: 'category', attributes:['id','name']}
        ],
        where:{cateogry_id:cateogry_id, author_id:author_id, name:q}
    })
    if (!books?.length) {
        return res.status(400).json({ message: 'No books found' })
    }
    res.json(books)
}
```

נוסיף ROUTE

```
router.get('/search', bookController.search)
```

נקרא לפונקציה דרך postman

http://localhost:3600/api/books/search?cateogry_id=1&author_id=1&q=למרות הכל

כאשר מעבירים פרמטרים דרך QUERY STRING, הפרמטר הראשון הוא באמצעות סימן שאלה ? כל שאר הפרמטרים הם באמצעות &

http://localhost:3600/api/books/search?cateogry_id=1&author_id=1&q=למרות הכל

ניתן לראות שרק ספר אחד עונה על הדרישה ולכן מגיע רק ספר אחד

GET http://localhost:3600/api/books/search?cateogry_id=1&author_id=1&q=למרות הכל

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (9) Test Results Stat

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "author": {
            "id": 1,
            "name": "שמואל ארגמן"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    }
]
```


כפי שניתן לראות מתוצאות השאילתא וגם לראות בטרמינל ברירת המחדל של החיפוש זה AND

מצא את הספרים שהם בשם הזה **וגם** בקטגוריה הזו **וגם** המחבר הזה

```
Executing (default): SELECT `book`.`id`, `book`.`name`, `book`.`picture`, `author`.`id` AS `author.id`, `author`.`name` AS `author.name`, `categ
ory`.`id` AS `category.id`, `category`.`name` AS `category.name` FROM `books` AS `book` LEFT OUTER JOIN `authors` AS `author` ON `book`.`author_
id` = `author`.`id` LEFT OUTER JOIN `categories` AS `category` ON `book`.`cateogry_id` = `category`.`id` WHERE `book`.`cateogry_id` = '1' AND `b
ook`.`author_id` = '1' AND `book`.`name` = 'לכה תורמל';
[]
```

אם נרצה למצוא לפי **או**

או המחבר הזה או הקטגוריה הזו או המחבר הזה

Op

נוסיף את האפשרות של האופרטור

```
const { Op } = require('sequelize');
```

ובשאילתא

```
        where:{
        [Op.or]:{cateogry_id:cateogry_id, author_id:author_id, name:q}
        }
```

עכשיו שנריץ נקבל 2 ספרים היות שגם הספר השני הוא של אותה הקטגוריה

GET http://localhost:3600/api/books/search?cateogry_id=1&author_id=1&q=למרות הכל

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "author": {
            "id": 1,
            "name": "שמואל ארגמן"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    },
    {
        "id": 2,
        "name": " איש בא מן העבר",
        "picture": "",
        "author": {
            "id": 2,
            "name": " דבורה צוף"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    }
]
```


חיפוש באמצעות LIKE

```
        where:{
            [Op.or]:{
                cateogry_id:cateogry_id,
                author_id:author_id,
                name:{
                    [Op.like]:`%${q}%`
```

```
                }
            }
        }
```

http://localhost:3600/api/books/search?cateogry_id=3&author_id=3&q=מ

אם נכתוב עכשיו בפרמטרים קוד קטגוריה ומחבר לא קיים אך נכתוב רק חלק משם הספר זה יחזיר את הספר

שיפור בשאילתא לבדוק אילו פרמטרים התקבלו ולבנות חיפוש דינמי

```
    let where ={}
    if(cateogry_id) where.cateogry_id= cateogry_id
    if(author_id) where.author_id= author_id
    if(q) where.name= { [Op.like]:`%${q}%` }
```

זה הקוד המלא של החיפוש (החזרנו לחיפוש לפי AND כי זה יותר מתאים)

```
const search = async (req, res) => {
    const { cateogry_id, author_id, q } = req.query
    //console.log(cateogry_id, author_id, q)
    let where ={}
    if(cateogry_id) where.cateogry_id= cateogry_id
    if(author_id) where.author_id= author_id
    if(q) where.name= { [Op.like]:`%${q}%` }

    const books = await Book.findAll({
        attributes:['id','name','picture'],
        include    : [
            { model: Author, as: 'author', attributes:['id','name']},
            { model: Category, as: 'category', attributes:['id','name']}
        ],
        where:{
            [Op.and]:where
        }
    })
    if (!books?.length) {
        return res.status(400).json({ message: 'No books found' })
    }
    res.json(books)
                                                                    {
```

GET http://localhost:3600/api/books/search?q=מ

Params Authorization Headers (7) Body Pre-request Script Tests

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "למרות הכל",
        "picture": "",
        "author": {
            "id": 1,
            "name": "שמואל ארגמן"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    },
    {
        "id": 2,
        "name": " איש בא מן העבר",
        "picture": "",
        "author": {
            "id": 2,
            "name": " דבורה צוף"
        },
        "category": {
            "id": 1,
            "name": "ספרי נוער"
        }
    }
]
```


limit

במידה ורוצים רק מספר רשומות

limit - מספר הרשומות

offset - תדלג על הרשומות

```
// Fetch 10 instances/rows
Project.findAll({ limit: 10 });
```

```
// Skip 8 instances/rows
Project.findAll({ offset: 8 });

// Skip 5 instances and fetch the 5 after that
Project.findAll({ offset: 5, limit: 5 });
```

Order

למיין את הרשומות ASC סדר עולה- DESC סדר יורד

```
order:[['createdAt','DESC']]
```

Raw Queries

למרות העבודה עם ORM יש כמובן אפשרות להריץ שאילתות SQL "רגילות"
נייבא

```
const { QueryTypes } = require('sequelize')
```

```
 const books = await db.sequelize.query("SELECT * FROM `books`", { type:
QueryTypes.SELECT });
```

כאשר עובדים עם שרשור **SQL יש להיזהר לעבוד עם פרמטרים** - לעבודה נכונה ומאובטחת
דוגמא לעדכון כולל פרמטרים

```
const { QueryTypes } = require('sequelize');

await sequelize.query(
 'SELECT * FROM users WHERE name LIKE :search_name',
 {
   replacements: { search_name: 'ben%' },
   type: QueryTypes.SELECT
 }
);
```

Update

נכתוב פונקצית עדכון לספרים - דומה מאוד לפונקציה של CREATE אך משתמשת בWHERE ונייצא אותה

```
const update = async (req, res) => {
    const {id, name , picture, cateogry_id, author_id } = req.body

    if (!id || !name || !cateogry_id || !author_id) {// Confirm data
        return res.status(400).json({ message: 'All fields are required'
})
    }
    const book = await Book.update({  name , picture, cateogry_id,
author_id },{where:{id:id}})
    if (!book) {
        return res.status(400).json({ message: 'book not found' })
    }
    res.json(book)
}

module.exports = {create, getAll, search, update}
```

נוסיף ROUTE

```
router.patch('/', bookController.update)
```

המערכת מחזירה את מספר הרשומות שעודכנו

PATCH http://localhost:3600/api/books

Params Authorization Headers (10) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQ

```
1 {
2     "id":1,
3     "name" :"חדש הכל למרות",
4     "picture":"",
5     "cateogry_id" : 1,
6     "author_id":1
7 }
```

Body Cookies (1) Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2     1
3 ]
```

/ findByPk / findOne

על מנת לסגור את המודל נוסיף את השאילתות GETONE

```
const getOne = async (req, res) => {
    const id = req.params.id
    //const book = await Book.findOne({where:{id:id}})
    //const book = await Book.findByPk(id)
    const book = await Book.findByPk(id,{attributes:['name']})
    res.json(book)
}
module.exports = {create, getAll, search, update, getOne}
```

נוסיף ROUTE

```
router.get('/:id', bookController.getOne)
```

destroy

ואחרון חביב (?) למחיקת הרשומה

```
const deleteOne = async (req, res) => {
    const { id } = req.body
    if (!id) {// Confirm data
        return res.status(400).json({ message: 'book ID required' })
    }
    await Book.destroy({ where: {id: id}});
    res.json( `Book  with ID ${id} deleted`)
}


module.exports = {create, getAll, search, update, getOne, deleteOne}
```

ונוסיף את הROUTE

```
router.delete('/', bookController.deleteOne)
```