Week 8: MVC and REST

Exercise Instructions:

During this exercise, document your progress using screenshots of the entire screen to demonstrate that you have completed all the required tasks. Save all screenshots in a folder named “proof” within your GitHub project, and organize them chronologically (e.g., 1.png, 2.png, etc.). Screenshots can be in any standard image format (e.g., png, jpeg).

You don’t need hundreds of screenshots, but please include enough to clearly show completion of the required tasks. Your README file must also include a link to the repository.

For Submission:

Download the entire repository from GitHub and upload it to Moodle. Do not provide links to external locations (e.g., Google Drive or GitHub links). Note: Moodle has a 5MB upload limit. If necessary, remove screenshots from the submission to fit the size limit, but ensure that all screenshots remain in the GitHub repository.

MVC:

1. Implement the code from the lecture (8 - MVC example.pdf) until slide 20 (including).
2. You have to implement the code gradually. That is, you first execute only the code from the first slide. Then, implement the changes described in slides 2-3 and execute. Then, the changes in slides 5-6 and execute. Continue like this, understanding the gradual changes that each slide introduces towards MVC.
3. You have to document (using print screens) that you have indeed implemented the code gradually as instructed.
4. Use debugging to see the MVC structure “in action”.

REST:

We will now transform the MVC project into RESTful API.

1. In:

```
const getArticle = (req,res) => {
    const id = 1
    res.render('article.ejs', { foo : articles.getArticle(id) })
}
```

Replace:

res.render('article.ejs', { foo: articles.getArticle(id) })

With:

res.json(articles.getArticle(id))

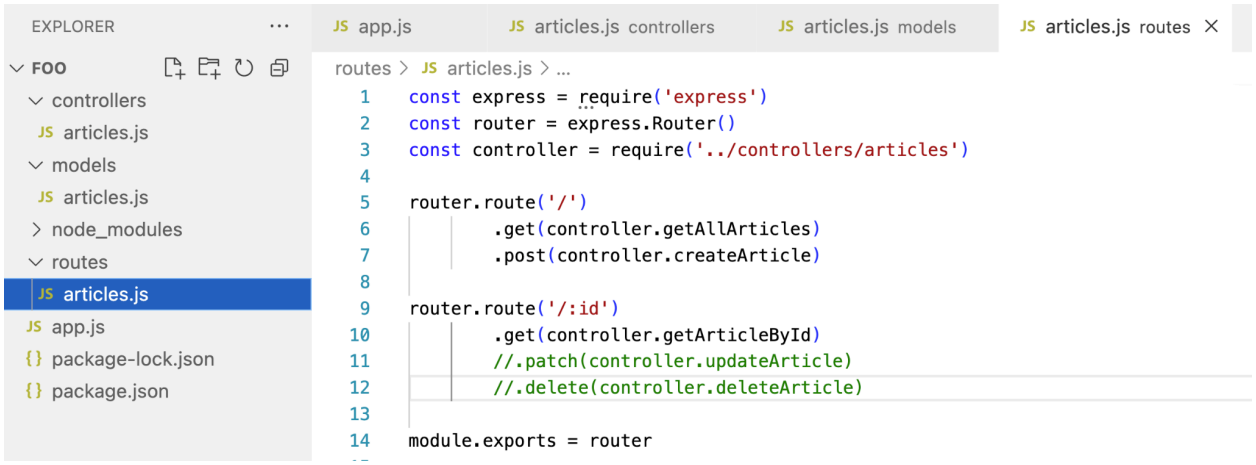
2. Use curl (or an equivalent tool, but not a browser) to invoke the getArticle function and ensure that the response contains the relevant json object and **not** an HTML.
3. Add the necessary code to the server so that the server will support CRUD operations (create, read, update, delete) in a RESTful manner. That is:
    a. Get (to get all articles)
        i. Simple - very similar to what we did above for getArticle
    b. Post (to add a new article)
        i. What needs to be added?
            1. Relevant code in routes (for post)
            2. Relevant code in controller (to extract new article's fields from the request's body, call model function, return 201)
            3. Relevant code in model (to add the new article object to the array)
    c. PUT or PATCH (to update an article)
    d. DELETE (to delete an article)

e. אני מצרף בסוף התרגיל את הפתרון לסעיפים a,b פלוס דוגמת הרצה. את הפתרון לסעיפים c,d תצטרכו להשלים **בעצמכם** על בסיס הקוד שנתתי.

4. Demonstrate using curl (or equivalent) that each of these CRUD operation works.

End of Week 8 Mini-Exercise


EXPLORER
FOO
controllers
articles.js
models
articles.js
node_modules
routes
articles.js
app.js
package-lock.json
package.json
app.js
articles.js controllers
articles.js models
articles.js routes
app.js > ...
1 const express = require('express')
2 const app = express()
3 const articleRoutes = require('./routes/articles')
4
5 app.use(express.json())
6 app.use('/api/articles', articleRoutes)
7
8 app.listen(3000)
9


EXPLORER
FOO
controllers
articles.js
models
articles.js
node_modules
routes
articles.js
app.js
package-lock.json
package.json
app.js
articles.js controllers
articles.js models
articles.js routes
routes > articles.js > ...
1 const express = require('express')
2 const router = express.Router()
3 const controller = require('../controllers/articles')
4
5 router.route('/')
6 .get(controller.getAllArticles)
7 .post(controller.createArticle)
8
9 router.route('/:id')
10 .get(controller.getArticleById)
11 //.patch(controller.updateArticle)
12 //.delete(controller.deleteArticle)
13
14 module.exports = router


EXPLORER
app.js
articles.js controllers
articles.js models
articles.js routes
FOO
controllers
articles.js
models
articles.js
node_modules
routes
articles.js
app.js
package-lock.json
package.json
controllers > articles.js > getArticleById > getArticleById > article
const Article = require('../models/articles')
exports.getAllArticles = (req, res) => {
res.json(Article.getAllArticles())
}
exports.getArticleById = (req, res) => {
const article = Article.getArticle(parseInt(req.params.id))
if (!article)
return res.status(404).json({ error: 'Article not found' })
res.json(article)
}
exports.createArticle = (req, res) => {
const { title, content } = req.body
if (!title || !content)
return res.status(400).json({ error: 'Title and content required' })
const newArticle = Article.createArticle(title, content)
res.status(201).location(`/api/articles/${newArticle.id}`).end()
}


EXPLORER
app.js
articles.js controllers
articles.js models
articles.js routes
FOO
controllers
articles.js
models
articles.js
node_modules
routes
articles.js
app.js
package-lock.json
package.json
models > articles.js > createArticle
let idCounter = 0
const articles = []
const getAllArticles = () => articles
const getArticle = (id) => articles.find(a => a.id === id)
const createArticle = (title, content) => {
const newArticle = { id: ++idCounter, title, content }
articles.push(newArticle)
return newArticle
}
module.exports = {
getAllArticles,
getArticle,
createArticle
}

```
% curl -i http://localhost:3000/api/articles
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 2
ETag: W/"2-l9Fw4VUO7kr8CvBlt4zaMCqXZ0w"
Date: Sun, 18 May 2025 21:57:56 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[]%
```

Get

```
% curl -i -X POST http://localhost:3000/api/articles \
-H "Content-Type: application/json" \
-d '{"title":"hello", "content": "world"}'
HTTP/1.1 201 Created
X-Powered-By: Express
Location: /api/articles/1
Date: Sun, 18 May 2025 21:58:09 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 0
```

Post

```
% curl -i http://localhost:3000/api/articles
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 44
ETag: W/"2c-wLz1NM9KqUvpeNSe3YU1o1iWyDE"
Date: Sun, 18 May 2025 21:58:15 GMT
Connection: keep-alive
Keep-Alive: timeout=5

[{"id":1,"title":"hello","content":"world"}]%
```

Get

```
% curl -i http://localhost:3000/api/articles/1
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 42
ETag: W/"2a-gpTkubZhIRwUp+9yPF0tf1tzdL4"
Date: Sun, 18 May 2025 21:58:38 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"id":1,"title":"hello","content":"world"}%
```

Get

```
% curl -i http://localhost:3000/api/articles/2
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 29
ETag: W/"1d-Ub44aXKiijEe3tqbqvC+PRJfqto"
Date: Sun, 18 May 2025 21:58:50 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{"error":"Article not found"}%
```

Get