

Mongoose crud

Εισαγωγή

- Αφού είδαμε την αρχιτεκτονική MVC, και πως μπορεί να διαμορφωθεί μια εφαρμογή με βάση την εν λόγω αρχιτεκτονική, θα δούμε τώρα πως μπορούμε να εφαρμόσουμε τις λεγόμενες CRUD λειτουργίες με το mongoose
- Create
- Read
- Update
- Delete

Create

Create

- Στο προηγούμενο παράδειγμα, είχαμε δει τον παρακάτω τρόπο για να δημιουργούμε έγγραφα στη βάση

```
//lets create a new document out of the model
const testLandmark= new Landmark({
  type: "landmark",
  name: "Acropolis",
  description: "The Acropolis is ...",
  ratingsAverage: 4.9,
  ratingsQuantity: 97
});
//save it to Landmarks collection and we have also access to it
testLandmark.save().catch(err=>{
  console.log("ERROR: "+ err);
});
```

1^ο-> δημιουργούμε νέο έγγραφο

2^ο -> χρησιμοποιούμε τη μέθοδο save για να το σώσουμε

Άρα καλούμε τη μέθοδο στο έγγραφο...

Το έγγραφο έχει πρόσβαση στη μέθοδο αυτή όπως και σε πολλές άλλες

Create

- Τώρα θα δημιουργήσουμε ένα έγγραφο κατευθείαν χρησιμοποιώντας το μοντέλο
- Επιπλέον, θα χρησιμοποιήσουμε την συνάρτηση `async`
- Μια `async` συνάρτηση δηλώνεται με τις λέξεις κλειδιά `async/await`

Create

- Με το `async/await` επιτυγχάνουμε ασύγχρονη συμπεριφορά βασισμένη σε «promises»

“The `async` and `await` keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.”

Source:

https://developer.mozilla.org/enUS/docs/Web/JavaScript/Reference/Statements/async_function

async/await

Οπότε δηλώνουμε μια συνάρτηση ως `async` και με τη λέξη-κλειδί `await` αναβάλλουμε την εκτέλεση του κώδικα.

Προσέξτε ότι μέσα σε μια `async` συνάρτηση μπορούμε να έχουμε ένα ή περισσότερα `await`

Αντί για `then` έχουμε `async...` κάνουμε έτσι τον κώδικα πιο όμορφο

Async functions:

- are accessible natively in Node: **async keyword** for declaration
- they always return a promise

await keyword is currently restricted to async functions (cannot be used in the global scope)

Create

```
landmarkController.js X JS landmarkRoutes.js
controllers > JS landmarkController.js > createLandmark > createLandmark > data > tour
const Landmark = require('../models/landmarkModel');

//landmarks routehandler
exports.createLandmark = async (req, res) => {
  try {
    const newLandmark = await Landmark.create(req.body);

    res.status(201).json({
      status: 'success',
      data: {
        tour: newLandmark
      }
    });
  } catch (err) {
    res.status(400).json({
      status: 'fail',
      message: err
    });
  }
};
```

- Μπορούμε να χρησιμοποιήσουμε το **μοντέλο** απευθείας με την **μέθοδο create**
- Τα μοντέλα Mongoose έχουν μια συνάρτηση `create()` που χρησιμοποιείται για τη δημιουργία νέων εγγράφων.
- Στη μέθοδο περνάμε τα δεδομένα που θέλουμε να αποθηκεύσουμε :post body (αποθηκεύεται στο `req.body`)
- Αποθηκεύουμε το νέο έγγραφο στη μεταβλητή `newLandmark`
- Queries are *thenables* but they are not promises, for more check <https://mongoosejs.com/docs/promises.html>

Create

```
landmarkController.js X JS landmarkRoutes.js
controllers > JS landmarkController.js > createLandmark > createLandmark > data > tour
const Landmark = require('../models/landmarkModel');

//landmarks routehandler
exports.createLandmark = async (req, res) => {
  try {
    const newLandmark = await Landmark.create(req.body);

    res.status(201).json({
      status: 'success',
      data: {
        tour: newLandmark
      }
    });
  } catch (err) {
    res.status(400).json({
      status: 'fail',
      message: err
    });
  }
};
```

- Try/catch syntax: έχει πρόσβαση στο err object
- Από τη μέθοδο create δημιουργείται ένα νέο έγγραφο στη βάση
- Αν κάτι πάει στραβά -> θα τρέξει το catch block
- Στο catch block στέλνουμε πίσω ένα response ότι υπήρξε σφάλμα.
- res.status : 400 -> σημαίνει bad request

```
POST http://localhost:8080/api/v1/landmarks
Body
[{"type": "landmark",
  "name": "Temple of Olympian Zeus",
  "description": "Behind Hadrian's Arch, stands the captivating temple of Olympian Zeus. The building process started in the 6th century by Peisistratos and was finally finished 100 years later in 131 AD by Emperor Hadrian. Originally it consisted of 104 Corinthian columns of which only 15 remain standing today. Inside the temple, Hadrian built an enormous gold and ivory statue of Zeus and an equal one of himself. To this day we do not know when the temple was destroyed but like many other large buildings in Greece, it is possible that it was brought down by an earthquake during the mediaeval period and the ruins sold for other building materials.",
  "ratingsAverage": 4.4,
  "ratingsQuantity": 27}
```

ARISTEA'S ORG - 2021-09-27 > LANDMARKS-APP > DATABASES

Cluster0 VERSION 4.4.9 REGION AWS N. Virginia (us-east-1)

Overview Real Time Metrics Collections Search Profiler Performance Advisor Online Archive

DATABASES: 1 COLLECTIONS: 1 VISUALIZE YOUR DATA REFRESH

+ Create Database

NAMESPACES

- landmarks-app
 - landmarks

landmarks-app.landmarks

COLLECTION SIZE: 2.02KB TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER { field: 'value' } OPTIONS Apply Reset

QUERY RESULTS 1-3 OF 3

```
{
  "_id": "ObjectId('6154669a0ee2870f5844c820')",
  "type": "landmark",
  "name": "Acropolis",
  "description": "The Acropolis is the strongest and most important monument of Ancient ...",
  "ratingsAverage": 4.9,
  "ratingsQuantity": 97,
  "__v": 0
}
```

```
{
  "_id": "ObjectId('615ed49ef544292fd8d20ba1')",
  "type": "landmark",
  "name": "Hadrian's Arch",
  "description": "Before starting the climb to get the Parthenon, it is impossible to mi...",
  "ratingsAverage": 4.2,
  "ratingsQuantity": 34,
  "__v": 0
}
```

```
{
  "_id": "ObjectId('615ed724f544292fd8d20ba3')",
  "type": "landmark",
  "name": "Temple of Olympian Zeus",
  "description": "Behind Hadrian's Arch, stands the captivating temple of Olympian Zeus....",
  "ratingsAverage": 4.4,
  "ratingsQuantity": 27,
  "__v": 0
}
```

Read

Model. find

- find all documents
- `MyModel.find({});`

- `// find all documents named aristeia and age >= 32`
- `MyModel.find({ name: 'aristeia', age: { $gte: 32 } })`

- `// find name LIKE aristeia and only selecting the "name" and "friends" fields`
- `MyModel.find({ name: /aristeia/i }, 'name friends');`

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Read

```
// landmarks.js
> exports.createLandmark = async (req, res) => {
  };

exports.getAllLandmarks = async (req, res) => {
  try {
    // find method returns an array of all documents and converts them into JavaScript objects
    const landmarks = await Landmark.find();
    console.log(landmarks);
    // SEND RESPONSE
    res.status(200).json({
      status: 'success',
      results: landmarks.length,
      data: {
        landmarks
      }
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

Ola ca Landmarks ↗

Ερώτηση

- Θα μπορούσα να κάνω το παραπάνω με then?

- Select all data
 - `Landmark.find({})`
 - `SELECT * FROM landmarks`
-

- get all data from table without `_id`
 - `Landmark.find({}, {_id:0})`
-

- Get all data name and id field
- `Landmark.find({}, {roll:1})`
 - `SELECT id, roll FROM landmarks`

- find specified data fields using where clause

```
Landmark.find({type:req.params.type},{name:1,_id:0});
```

```
SELECT name FROM landmarks WHERE type = req.params.type
```

- find data using greater than condition

```
Landmark.find({ratingsAverage:{ $gt: req.params.type}},{name:1,_id:0});  
SELECT name FROM landmarks WHERE ratingsAverage >  
req.params.type
```


Read by id

```
routes.route('/:id')  
  .get(landmarkController.getLandmarkById)
```

έχουμε παράμετρο ID στο route μας

Find the landmark with the given 'id'

```
const landmarks = await Landmark.findById(req  
.params.id);
```

```
const landmarks = await Landmark.findOne({_id:  
req.params.id })
```

// ίδια, διαφορετικός τρόπος!

αποκτήμα πρόσβαση στο id routehandler -> req.params.id.
req.params object -> used to access Route parameters

```
const Landmark = require('../models/landmarkModel');  
  
//landmarks routehandler  
exports.createLandmark = async (req, res) => { ...  
};  
  
exports.getAllLandmarks = async (req, res) => { ...  
};  
  
exports.getLandmarkById = async (req, res) => {  
  try {  
    const landmarks = await Landmark.findById(req.params.id);  
    // Tour.findOne({ _id: req.params.id })  
  
    res.status(200).json({  
      status: 'success',  
      data: {  
        landmarks  
      }  
    });  
  } catch (err) {  
    res.status(404).json({  
      status: 'fail',  
      message: err  
    });  
  }  
};
```

Select one field

- Θα δούμε στη συνέχεια πως γίνεται να επιλέξουμε συγκεκριμένα πεδία από ένα document

Router →

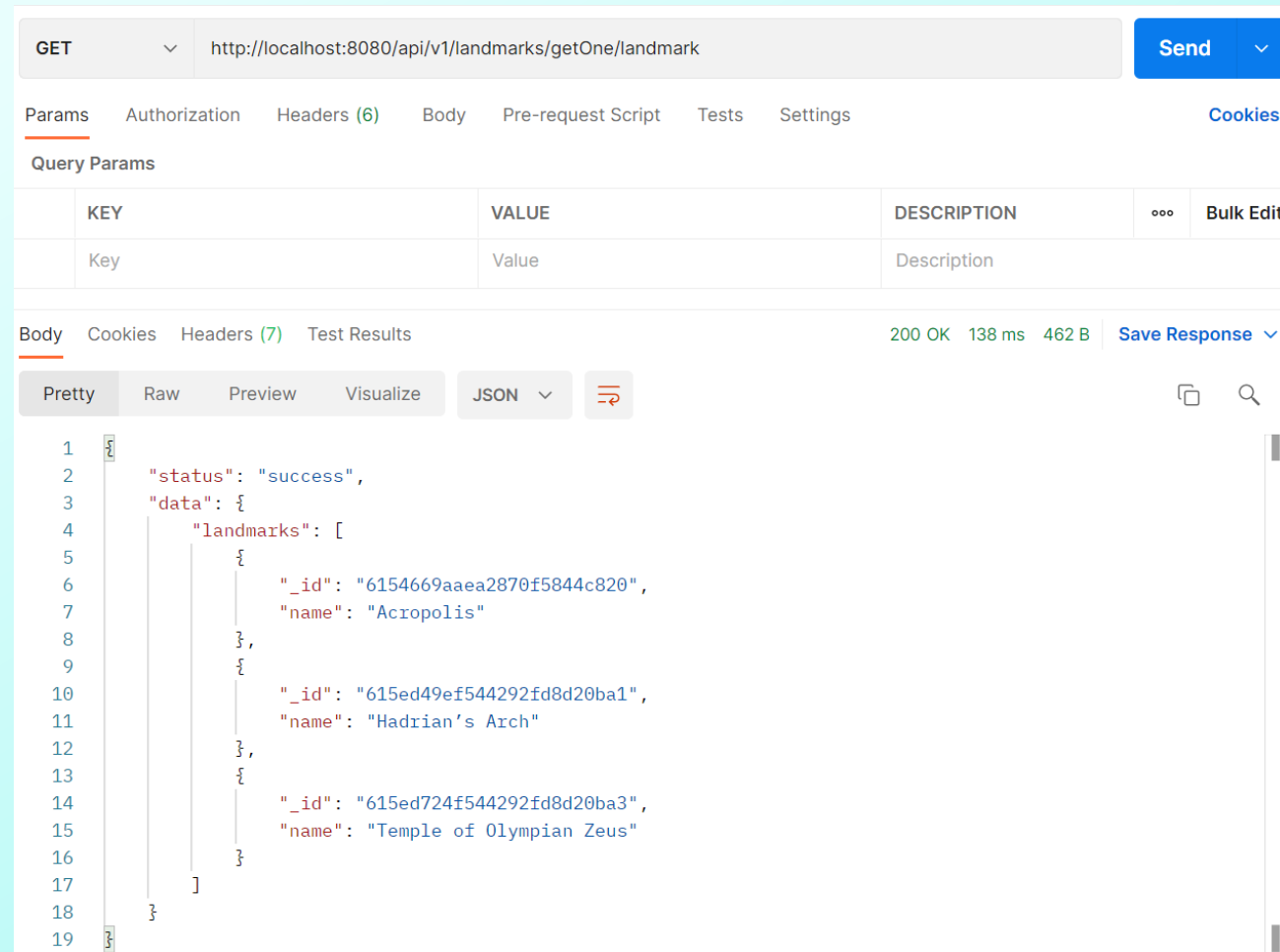
```
routes.route('/getOne/:type')  
  .get(landmarkController.getLandmarkOneById)
```

```
// routerhandler to get one  
exports.getLandmarkOneById = async (req, res) => {  
  try {  
    const landmarks = await Landmark.find({type:req.params.type},{name:1});
```

↑
η θα δείξει
+ id

Select one field

- Θα δούμε στη συνέχεια πως γίνεται να επιλέξουμε συγκεκριμένα πεδία από ένα document



The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/v1/landmarks/getOne/landmark`. The response is a JSON object with a status of "success" and a data object containing an array of landmarks. The response status is 200 OK, with a response time of 138 ms and a size of 462 B.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

```
1  {
2    "status": "success",
3    "data": {
4      "landmarks": [
5        {
6          "_id": "6154669aaaa2870f5844c820",
7          "name": "Acropolis"
8        },
9        {
10         "_id": "615ed49ef544292fd8d20ba1",
11         "name": "Hadrian's Arch"
12       },
13       {
14         "_id": "615ed724f544292fd8d20ba3",
15         "name": "Temple of Olympian Zeus"
16       }
17     ]
18   }
19 }
```

- `// Find one document where gender: female ` otherwise `null``
- `MyModel.findOne({ gender: female ' });`

- `// select only the document age and eyes`
- `MyModel.findOne({gender: female ' }, 'age eyes');`

Ερώτηση

- Πως θα κάναμε read by name στο παράδειγμα δλδ?

Ερώτηση

- Πως θα κάναμε read by name?

```
routes.route('/:name')  
.get(landmarkController.getLandmarkByName)
```

```
exports.getLandmarkByName = async (req, res) => {  
  try {  
    const landmarks = await Landmark.findOne({name: req.params.name })  
  
    res.status(200).json({  
      status: 'success',  
      data: {  
        landmarks  
      }  
    });  
  } catch (err) {  
    res.status(404).json({  
      status: 'fail',  
      message: err  
    });  
  }  
};
```

Update

Update

- `findByIdAndUpdate()` function
 - finds a matching document
 - updates it according to the update arg
 - returns the found document (if any) to the callback.

Update

- **new: true** -> *return the document after update was applied.*
- `findByIdAndUpdate(id, ...)` \Leftrightarrow `findOneAndUpdate({ _id: id }, ...)`.
-

Update

The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/api/v1/landmarks/6154669aaaa2870f5844c820`. The request body is a JSON object with `"ratingsAverage": 5` and `"ratingsQuantity": 300`. The response is a 200 OK status with a JSON body containing a success message and the updated landmark details, including its ID, type, name, description, and the updated ratings.

```
exports.updateLandmarkById = async (req, res) => {
  try {
    // req.body contains data we want to change
    const landmarks = await Landmark.findByIdAndUpdate(req.params.id, req.body, {
      new: true, // we want this method to return new updated document to the client
    });

    res.status(200).json({
      status: 'success',
      data: {
        landmarks
      }
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
}
```

- PUT -> is used to modify an existing entity
 - it replaces an entity.-> If we don't include a property that an entity contains, it should be removed
- PATCH -> is used to apply a **partial** modification to a resource.
 - Used to update only the properties we wish

Delete

Delete

```
exports.deleteLandmarkById = async (req, res) => {
  try {
    await Landmark.findByIdAndDelete(req.params.id);

    res.status(204).json({
      status: 'success',
      data: null
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err
    });
  }
};
```

- <https://mongoosejs.com/docs/queries.html>
- <https://mongoosejs.com/docs/models.html#constructing-documents>

To be continued...