

JavaScript Basic Web: Phonebook

1. Problem

You have been tasked to create a simple **Phonebook** application. The application should hold **contacts**, which are the main app **entity**.

The functionality of the application should support:

- **Listing contacts**

Phonebook

All Contacts

Name	Number
John Smith	+359894102523

New Contact

Name

Number

Add

© Software University

- **Add Contact**

Phonebook

All Contacts

Name	Number
John Smith	+359894102523

New Contact

Name

Adam Stones

Number

+359894102129

Add

© Software University

Phonebook

All Contacts

Name	Number
John Smith	+359894102523
Adam Stones	+359894102129

New Contact

Name

Number

Add

© Software University

2. Overview

Requirements

- **Express** framework
- **Handlebars** view engine

Data Model

The **Contact** holds **2 properties**:

- **name** - non-empty text
- **number** - non-empty text

Project Skeletons

You will be given the application **skeletons**, which hold about **90%** of the logic. You'll be given some **files**. The files will have **partially implemented logic**, so you'll need to write some code for the application to **function properly**.

The application's views will be given to you fully implemented. You only need to include them in your business logic.

Everything that has been given to you inside the skeleton is **correctly implemented** and if you write your code **correctly**, the application should work just fine. You are free to change anything in the Skeleton on your account.

3. Preparation

Run **"npm install"** in the terminal in the folder of the project to install all of the dependencies you will need. (express and handlebars)

4. Database

It's time to create our database. Open the **"phonebook.js"** file and add the following functionality:

```

1  var phonebook = [];
2
3  function getPhonebook() {
4    return phonebook;
5  }
6
7  function addContact(contact) {
8    phonebook.push(contact);
9  }
10
11 module.exports = {
12   getPhonebook,
13   addContact
14 };
15

```

- We **create a phonebook array**, which will store all the contacts (**memory storage**). The array **will be empty** each time we **restart the server**
- We add the function that will **return the whole phonebook**
- We add the function that will **add a new contact**
- Finally, we **export those functions**, so we can **use them outside this file**

5. Model

We should create our **contact model** which has a **name** and **number** as properties. Inside **Contact.js** creates a **class** that has a constructor and accepts and sets both properties. After that export it:

```

class Contact {
  constructor(name, number) {
    this.name = name;
    this.number = number;
  }
}

module.exports = Contact;

```

6. View

Go to the **index.hbs** file and find the following code:

```

31  <tbody>
32    {{#each contacts}}
33    <tr>
34      <td>{{this.name}}</td>
35      <td>{{this.number}}</td>
36    </tr>
37    {{/each}}
38  </tbody>

```

What this does is, it should **receive variables called contacts**, it **iterates through** them, and for each, it adds a row in the table with this **person's name and number**.

7. Routing

The routes in our application are defined inside **routing.js**. There are a total of **two** routes inside **'/'** which is called when we load the application (HTTP GET) and **'/add'** which is called when we **add a new contact** inside the database array (HTTP POST). Every route leads to a function from the controller:

```
const phonebookController = require('./controllers/phonebook-controller');

module.exports = (app) => {
  app.get('/', phonebookController.index);
  app.post('/add', phonebookController.addPhonebookPost);
}
```

8. Phonebook Controller

To **list** and **add** contacts we need a controller so go to **controllers/phonebook-controller.js**:

```
module.exports = {
  index: (req, res) => {
    // TODO: load index page
  },
  addPhonebookPost: (req, res) => {
    // TODO: add a phonebook object to the array
  }
}
```

Listing Contacts

Then when we render the main page, we need to **pass the stored contacts to the view**, so we can see them. To do that, add the following code in the **index** method:

```
index: (req, res) => {
  res.render("index", {
    contacts: phonebook.getPhonebook()
  });
},
```

- We pass the contacts as an object that has a value for **"contacts"** in the phonebook we stored.
- We do that by using our **getPhonebook()** method we wrote earlier

Adding Contact

First, let us see what we need to do. Go to the **index.hbs** again and find the following lines of code:

```

40 <form class="form-horizontal" action="/add" method="POST">
41   <fieldset>
42     <legend>New Contact</legend>
43     <div class="form-group">
44       <label for="name" class="col-lg-2 control-label">Name</label>
45       <div class="col-lg-10">
46         <input type="text" autofocus="autofocus" name="name" title="Name"
47           class="form-control" id="name" />
48       </div>
49     </div>
50     <div class="form-group">
51       <label for="number" class="col-lg-2 control-label">Number</label>
52       <div class="col-lg-10">
53         <input type="text" autofocus="autofocus" name="number" title="Number"
54           class="form-control" id="number" />
55       </div>
56     </div>
57     <div class="form-group">
58       <div class="col-lg-10 col-lg-offset-2">
59         <button type="submit" class="btn btn-primary">Add</button>
60       </div>
61     </div>
62   </fieldset>
63 </form>

```

- Here we see, that when we **press the submit button**, we make a **POST** request on **route "/add"**.
- The information that we submitted in this form will come as an **object with properties name and number** (since those are the **names of the input fields**)
- This is only possible because we imported **"body-parser"**

So let us now go to **phonebook-controller.js** and add the final functionality, which goes in the **app.post()** method:

```

addPhonebookPost:(req, res) => {
  let name = req.body.name;
  let number = req.body.number;
  let contact = new Contact(name, number);
  phonebook.addContact(contact);
  res.redirect('/');
}

```

- Don't forget to require the **Contact** class from **models/contact.js**

```
const Contact = require('../models/Contact');
```

- By typing **req.body** we get that object that we mentioned earlier (you can log that to the console if you want to see it)
- Then we just **use the function** we wrote to **add that contact** and **redirect to the index** page again

9. Testing the Application

With that, we finished our **JavaScript Phonebook**. Type **"node index.js"** in the terminal to start the server. Then go to **localhost:3000**. Feel free to **build on your project even further**. 😊