󠀠 Project B

Context

You keep on using the Express framework to build a prototype of Mr.Coffee's schedule management app.

You have completed the first phase of your action plan and got to know the framework. You already know how to create routes that return raw information.

You will now tackle the second step needed to build a web application: generating real web pages from your data.

For this step, you don't have any particular visual constraints. You simply need to build clean and readable HTML pages.

Guidelines

You will start from what you produced in the previous project. However, this time, instead of simply returning the data, you will render (i.e., transform into HTML pages) HTML templates.

Your data will be inserted into HTML templates, to let app users view the data through real web pages.

Step 1 : Set up the template engine

The very first step is the configuration of a template engine, as well as the folder where the engine will fetch the templates.

Read this brief tutorial to learn how to do it.

Step 2 : Create the layout template

You will first create the layout template, i.e. the page structure that remains the same no matter which page is displayed.

Your site is basic so you don't need to do anything complex. A bar at the top of the page with a title (e.g. "Schedule website") and a content area with margins is enough.

You will then define a content block. It will be replaced by the templates that inherit the layout.

Step 3 : Create templates for the GET routes

You will create templates for each GET route your application offers. These templates will extend the layout and will only redefine the content block.

Again, you are free to display the data as you like, as long as it’s user-friendly. For example, a page that displays user info will not display all the info in one line, but will display a "First Name:" field followed by the first name, on another line a "Last Name:" field followed by the last name, etc.

You will then modify the routes so that they generate HTML pages containing the data.

This means that the following routes will generate HTML pages:

- ‘/'
- ‘/users'
- ‘/schedules'
- route to display a given user
- route to display the schedules of a given user

🔲 Step 4 : Create forms for POST routes

Last step, we will make the creation of users and schedules possible via the browser!

For this you will need to add two routes:

- A '/users/new' route that displays a form to create a new user
- A '/schedules/new' route that displays a form to create a new schedule.

For this step you can simply collect the values in simple text fields.

Be careful to correctly configure the action and method attributes of your forms in order to send the data in POST to the right URL. Of course, you must also make sure that the names of your fields in the form correspond to the expected fields in your routes.

You will make sure that the routes '/users' and '/schedules' in POST redirect to their version in GET in order to display the list of users and schedules with their new element.

🔲 Bonus step : Limit the possible values in your form fields

Small finishing touch: you can limit the inputs of your users by pre-filling the form with possible values.

You can enforce that in the form for adding a schedule:

- There is a select field offering a choice among the days of the week.
- There is a select field offering a choice among existing users. Try to display the full names of the users!