Passing variables to templates

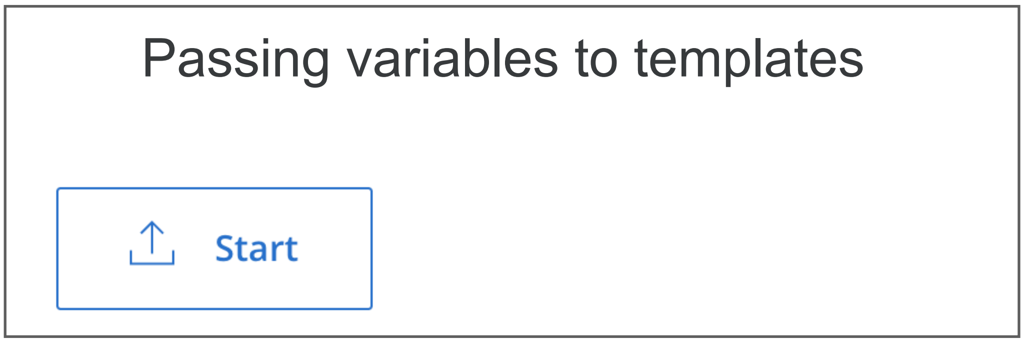
Welcome to this Lab activity

In this lab activity you will be exploring how to pass different variables to templates; remember EJS? For the purpose of this lab activity you will be working on the same structure as the “Accessing the database from Node” lab but under a new folder inside *topic6* called *expressDynamicApp*.

Start the Lab environment application

It is simple to launch a lab exercise. You only need to click on the button “Start” below the activity title to enter a lab environment.

Let’s explore this lab activity. Go ahead and click on the “Start” button!



Task 1: Query data in your Node.js file using templates

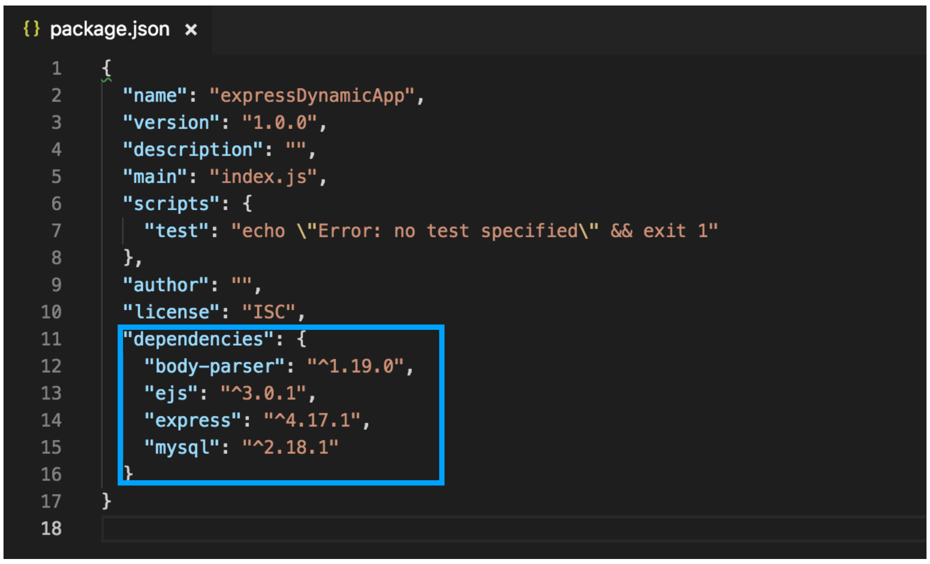
The folder structure has already been partially constructed for you and organised into different topics. For the purpose of this lab, you will be making changes inside the *topic6* folder structure. Let’s get started!

Please do not delete or move any existing folders/files inside the lab environment.

Use the Visual Studio Code Terminal and run the following commands:

- **cd topic6/expressDynamicApp**: type this command and press *Enter*. This command will change your working directory to be the express*DynamicApp* folder.
- **npm init**: type this command and press *Enter*. This command will set up a new or existing npm package. You can skip all the npm initialisation questions by pressing *Enter.* Once the npm package file has been created, you can view it inside the express*DynamicApp* directory (*package.json*). The *package.json* file contains a set of information regarding your current node project.
- **npm install express --save**: type this command and press *Enter*. This command will install the *Express* framework within your working directory so that it will be available for you to use.
- **npm install ejs --save**: type this command and press *Enter*. This command will install the *EJS* framework within your working directory so that it will be available for you to use.
- **npm install body-parser --save**: type this command and press *Enter*. This command will install the *body-parser* module within your working directory so that it will be available for you to use.
- **npm install mysql --save**: type this command and press *Enter*. This command will install the mysql module within your working directory so that it will be available for you to use. It will allow you to run mysql commands directly in your javascript files.

Once you run the above Terminal commands, your *package.json* file located inside the *topic6/expressDynamicApp* folder should contain **Express, EJS, Body-parser** and **Mysql** modules:


package.json ×

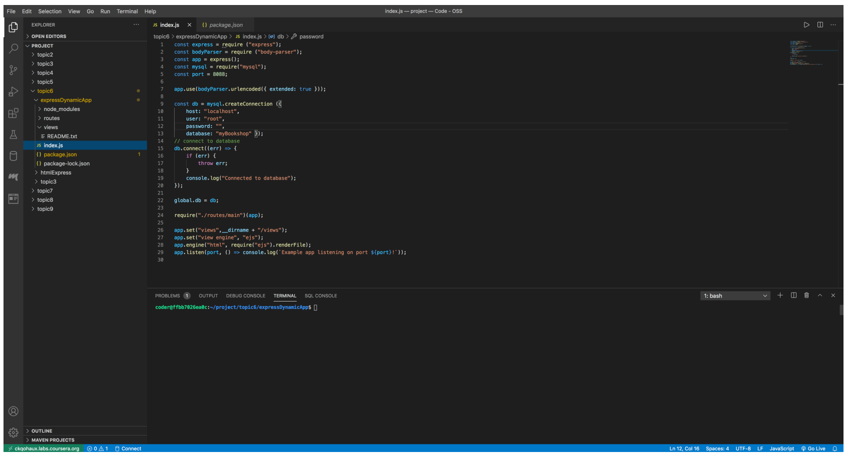
```
{
  "name": "expressDynamicApp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "ejs": "^3.0.1",
    "express": "^4.17.1",
    "mysql": "^2.18.1"
  }
}
```

The next step is to add some code to the *index.js* file located inside the *topic6/expressDynamicApp* folder in order to create your web server.

Add the following code to the *index.js* file and remember to save it:

```
const express = require ("express");
const bodyParser = require ("body-parser");
const app = express();
const mysql = require("mysql");
const port = 8088;

app.use(bodyParser.urlencoded({ extended: true }));

const db = mysql.createConnection ({
    host: "localhost",
    user: "root",
    password: "",
    database: "myBookshop" });
// connect to database
db.connect((err) => {
    if (err) {
        throw err;
    }
    console.log("Connected to database");
});

global.db = db;

require("./routes/main")(app);

app.set("views",__dirname + "/views");
app.set("view engine", "ejs");
app.engine("html", require("ejs").renderFile);
app.listen(port, () => console.log(`Example app listening on port ${port}!`));
```

If you have correctly followed all the above steps, your environment should be similar to the one below:



```
const express = require ("express");
const bodyParser = require ("body-parser");
const app = express();
const mysql = require("mysql");
const port = 8088;

app.use(bodyParser.urlencoded({ extended: true }));

const db = mysql.createConnection ({
    host: "localhost",
    user: "root",
    password: "",
    database: "myBookshop" });
// connect to database
db.connect((err) => {
    if (err) {
        throw err;
    }
    console.log("Connected to database");
});

global.db = db;

require("./routes/main")(app);

app.set("views",__dirname + "/views");
app.set("view engine", "ejs");
app.engine("html", require("ejs").renderFile);
app.listen(port, () => console.log(`Example app listening on port ${port}!`));
```

```
coder@ffbb7026ea0c:~/project/topic6/expressDynamicApp$
```

Now you have to copy your web application created inside the *topic6/htmlExpress* folder to the *topic6/expressDynamicApp* folder by following the steps below.

Create four html files inside the *topic6/expressDynamicApp/views* folder called *index.html about.html, search.html* and *register.html* as you did inside the *topic6/htmlExpress/views* folder. Copy the content of each html page from the *topic6/htmlExpress/views* folder to the ones inside the *topic6/expressDynamicApp/views* folder.

Finally copy the code from the *topic6/htmlExpress/routes/main.js* file inside the *topic6/expressDynamicApp/routes/main.js* file.

At this point you should have a copy of your Express application created in *topic6/htmlExpress* inside the *topic6/expressDynamicApp* folder.

Running the index.js file

Run the *index.js* file with the following Terminal command:

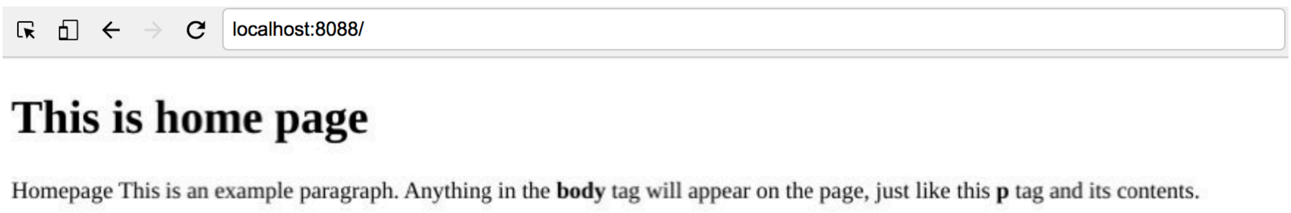
- **node index.js**: type this command and press Enter. The node command, followed by the file name, tells Node.js to execute the content of the file.

The above command will start a web server running on port 8088.

Use the "Browser Preview" plugin to check that the application is running correctly.

If you do not remember how to use the "Browser Preview" plugin, please refer to the "Creating my first Node.js web server" lab instructions.

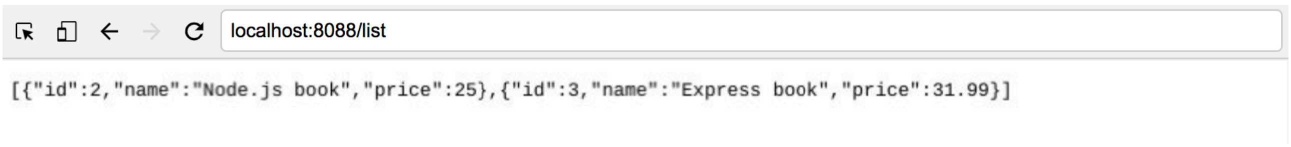
The 'root' route of your application should render the following:



This is home page

Homepage This is an example paragraph. Anything in the **body** tag will appear on the page, just like this **p** tag and its contents.

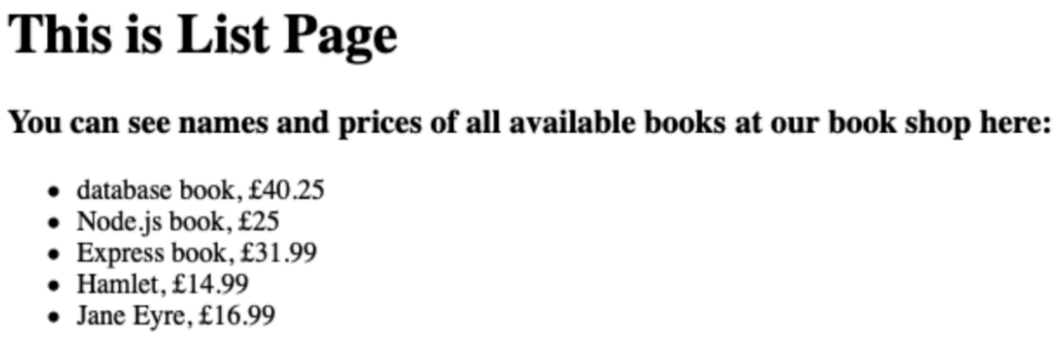
Now take a look at the "/list" route in your web application; it should look similar but with different data to the picture below:



localhost:8088/list

```
[{"id":2,"name":"Node.js book","price":25},{"id":3,"name":"Express book","price":31.99}]
```

Wouldn't it be great if you could display your data in a more clean and organised way? What about the following picture:



This is List Page

You can see names and prices of all available books at our book shop here:

- database book, £40.25
- Node.js book, £25
- Express book, £31.99
- Hamlet, £14.99
- Jane Eyre, £16.99

In order to create a similar output to the picture above, you need to change few things in your application:

- Update the "main.js" file to render the "list.html" template file and pass the result as a parameter to the latter
- Create a new template file, list.html in the "views" folder.

Let's update the "main.js" file located inside *topic6/expressDynamicApp/routes* folder. Update the "/list" route by replacing "res.send(result)" with the following code:

res.render("list.html", {availableBooks: result});

Next create a new template, "list.html", inside the *topic6/expressDynamicApp/views* folder and add the following code to it:

```
<!doctype html>
<html>
  <head>
    <title>List webpage!</title>
  </head>
 <body>
   <h1> This is List Page </h1>
   <h3>You can see names and prices of all available books at our book shop
   here:</h3>
   <ul>
     <% availableBooks.forEach(function(book){ %>
     <li><%= book.name %>, £<%= book.price %></li>
     <% }) %>
    </ul>
 </body>
 </html>
```

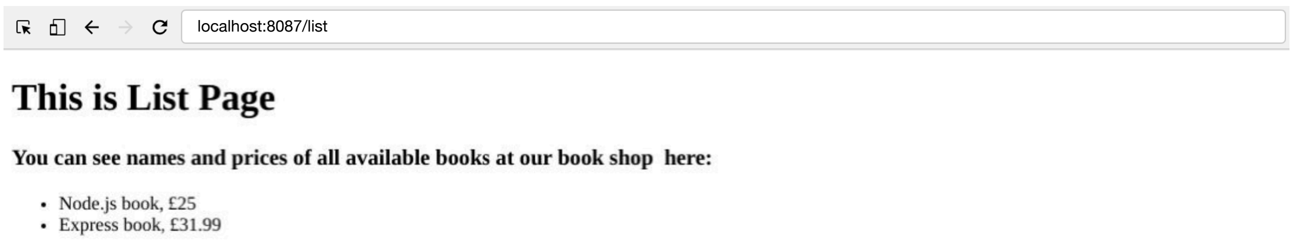
Task 2: Access your server via HTTP

Now that your code is running, serving your application on port 8087, you can access your web page from a browser!

Use the “Browser Preview” plugin to visualise your web application.

If you do not remember how to use the “Browser Preview” plugin, please refer to the “Creating my first Node.js web server” lab instructions.

Type *localhost:8087/list* on the “Browser Preview” tab and press *Enter* on your keyboard to visualise your web application:



localhost:8087/list

This is List Page

You can see names and prices of all available books at our book shop here:

- Node.js book, £25
- Express book, £31.99

Note that you will have different books in your list route.

End of Section

Congratulations for completing this section. As long as you have saved your work, your files will remain when you close this lab activity so do not worry about losing your data. You have successfully created a web application that connects to a database and uses dynamic templates to construct web pages. In the next lab activity you will learn how to pass variables to the back-end and add books to your database