

POST: collecting forms data

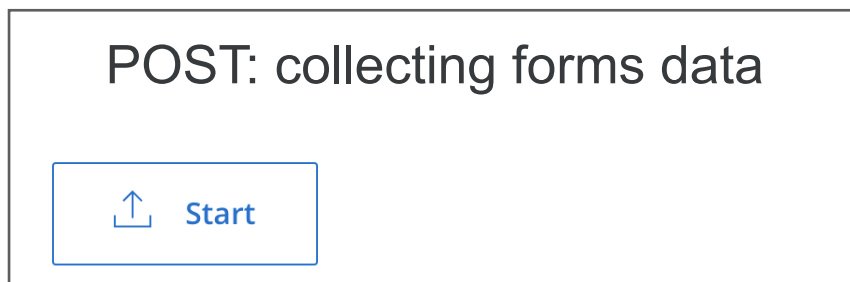
Welcome to this Lab activity

In this lab activity you will be exploring how to collect data from you existing form using the POST request method. For the purpose of this lab activity you will be working on the same structure as your previous lab: *topic4/htmlExpress*.

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: Adding another form and collecting form data, when the http request method is POST

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 *topic4* folder structure. You will be working in your previous lab application folder *topic4/htmlExpress*. Let’s get started!

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

For handing post request we need to take four steps:

- Installing ‘body-parser’ module: `npm install body-parser --save`
- Updating the ‘index.js’ file
- Updating the ‘main.js’
- Creating and editing a new html file named register.html in the ‘views’ directory

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

- **cd topic4/htmlExpress**: type this command and press *Enter*. This command will change your working directory to be the *htmlExpress* folder.
- **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.

Body-parser is a module that will allow you to access the data sent using the POST request method easily. By definition, it extracts the entire body portion of an incoming request stream and exposes it on **req.body**. Do not worry if this sounds confusing, we will explore this together.

Once you have installed *body-parser*, you need to edit your ‘index.js’ file inside the *topic4/htmlExpress* folder to be able to use it.

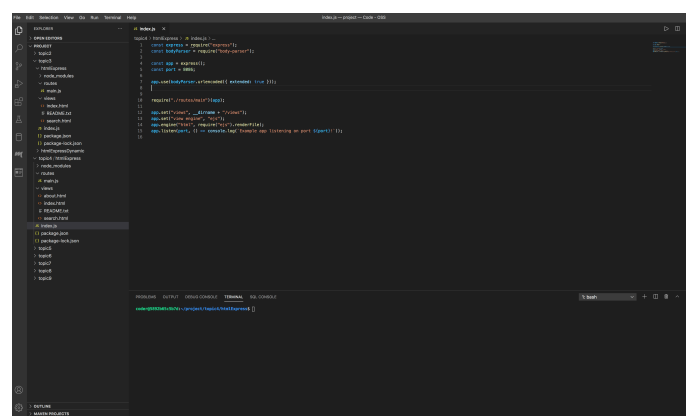
Add the following code to your ‘index.js’ file:

```
const bodyParser= require ("body-parser");
```

In addition, add the following line of code to your ‘index.js’ after the definition of *app* and *port* variables:

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

If you have successfully followed all the above steps you should see the following result:



Now that you have *body-parser* correctly installed, you need to create a new html page inside the *topic4/htmlExpress/views* folder called “register.html”. Add the same content as the “search.html” page to the newly created “register.html” page. Change the header accordingly.

Replace the form element inside the “register.html” with the following form:

```
<form method="POST" action="/registered">
  <input id="first" type="text" name="first" value="first name" />
  <input id="last" type="text" name="last" value="last name" />
  <input type="submit" value="OK" />
</form>
```

At this point you need to add two new routes inside your “main.js” file located in the *topic4/htmlExpress/routes* folder. You will need one route called “/register” to display the “register.html” page and one called “/registered” to display the form data collected from the registration form.

Add the following code to your “main.js” file:

```
app.get("/register", function (req,res) {
  res.render("register.html");
});

app.post("/registered", function (req,res) {
  // saving data in database
  res.send(req.body)
});
```

As you can see we have added a comment to search in the database. You will explore how to integrate databases with your application later in the course. For the moment you are simply collecting the data passed along with the form. These information are collected inside **req.body**.

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 8086.

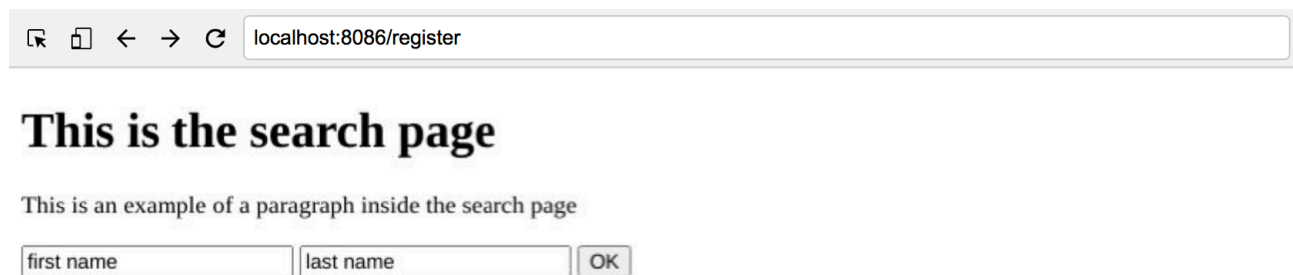
Task 2: Access your server via HTTP

Now that your code is running, serving your application on port 8086, 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:8086/register` on the “Browser Preview” tab and press *Enter* on your keyboard to visualise your web application “register” page:



The screenshot shows a web browser window with the address bar containing `localhost:8086/register`. The page content includes a heading **This is the search page**, a paragraph *This is an example of a paragraph inside the search page*, and a form with two input fields labeled 'first name' and 'last name', followed by an 'OK' button.

Now enter your first and last name and press the “OK” button to submit your form data. If you have successfully followed all the above steps you should get the following result:



The screenshot shows a web browser window with the address bar containing `localhost:8086/registered`. The page content displays a JSON object: `{"first": "first name", "last": "last name"}`.

Note: your output will depend from the data submitted with your form.

Finally replace the “res.send” line in your “main.js” file with the following:

```
app.post("/registered", function (req,res) {  
  // saving data in database  
  res.send("Hello "+ req.body.first + " "+ req.body.last +", you are now  
  registered!");  
});
```

If you run your application now, you should see the following result:



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 serves html files containing forms. Furthermore, you are now able to collect data using the POST request method.