**Web Server Basics: How to Submit Form Data to Express Server?**

In this blog post, let’s take a look at how we can submit (or send) data to our server using a html form. So basically we’ll be sending data from frontend to the backend. And our server will then acknowledge the reception of data by responding with a simple message.

Let’s get started.

**Prerequisites:**

You should know how to initialize an express project and setup a simple express server. If not, refer to <this> article.

1. **Create a Simple Express Server:** You already know this. Refer the video clip below to refresh your memory and setup a simple express server.

  <insert simple server video here>

2. **Rendering HTML Form Using Express Server:** First you have to configure express server such that starts serving html pages. To do this, you have to include the following line of code in your index.js file.

 <app.use(express.static….) video clip goes here….

**What does app.use(express.static('public')); do?**

It makes the app variable (which is our server) use a middleware called static that is responsible for serving html pages from the server.

What is public then? Public is the location (folder) from where the html page needs to be served.

So we now have to create a folder called public and place the html page that is going to contain our form in it.

**Note:** Whenever you come across app.use() statement, it means you are using an express middleware. Express middleware is simply a function that does something for us. We use the static middleware because it’s responsible for rendering of static html files.

To know more about express.static middleware read <this>. To know more about express middleware in general, read <this>.

For now though, this much knowledge is sufficient. So let’s move on.

<creation of public folder & index.html goes here>

**Testing Whether index.html File Renders**

**Note:** Before doing this, ensure you add some html content like <h1>hi from index.html</h1> in your index.html.

After restarting your server using node index.js command, try accessing the server @ http://localhost:3000/.

You’ll notice that our html page gets rendered, even without us configuring a specific get route for its rendering. What’s more is that if you try accessing http://localhost:3000/index.html URL, you’ll realize that our html page gets rendered again!

This is exactly what static middleware is supposed to do. Create http://localhost:3000/index.html link to serve our html page (just like we do in static websites) and serve it at the ‘/’ (root route) in our server for us by default.

And that’s the reason why the link http://localhost:3000/index.html still works. Because express.static middlware created it for us behind the scenes.

**3. Creating The HTML Form**

It’s time to create our html form. Here’s the code for it.

<selectable html code snippet goes here…>

If you observe our html code, you’ll realize that it is linked to a style.css file. This .css file too has to go into the same public folder. Here’s the code for it.

<selectable css code snippet goes here…>

Save both the files and restart your server using node index.js command and you’ll see a decent looking html form getting served at http://localhost:3000/.

**4. How HTML Form Works?**

2 form attributes (method and action) play a crucial role in sending data to our server.

method defines what our form is going to do. In our case, it’s going to post (send) data to server. Therefore we define it as POST (note the all caps rule).

action defines where our form data is going to be posted to. In our case, it’s the home or root route(’/’). Therefore we define it as ‘/’.

So our form is now configured and ready to post data to our server.

<video clip indicating form ‘method’ and ‘post’ attributes>

**5. How Express Server Handles Submitted Data?**

2 things need to be done to allow our server to receive the form data.

enable our server to read the data being submitted and

set up a route for receiving the data being submitted

**i) Enabling Our Server to Read the Data Being Submitted**

The request object (req of app.get(’/’) route discussed in <this blog>) has many properties associated with it. One of them is req.body. req.body is responsible for carrying the data to be submitted to the server.

So whenever we submit data to server using html forms, it’s this req.body that carries the submitted details and hands over them to the server for further processing.

But to read what’s in the req.body, our express server needs the help of another middleware: express.urlencoded({ extended: true })

So include the following line of code to enable express to read the form data.

<video clip showing adding of the middleware>

**ii) Setup a Route for Receiving the Data Being Submitted**

Now with express, everything happens through routes. If we are to serve some data (like a webpage) to the users, we need a separate route for it (known as get route). If we are to receive some data from the users (like through a form), we need a separate route for it (known as post route).

In our case, we need a post route that allows the html form to submit (or post) its data to our server. So let’s create one.

<video clip showing creation>

**Note:** Express routes are simply paths that enable data transfer between the server and the browser.

Now restart your server and try submitting the form. You’ll notice that our server is trying to process something (indicated by the loader icon) but is unable to.

This is because our app.post() route is empty at the moment. So let’s make it do something. Let’s add some code.

We know req.body contains the form details. Let’s try console logging this information that the app.post() route gets hold of.

<video snippet here…>

You’ll notice that the form details get console logged successfully. In other words, the contents of req.body object gets console logged successfully.

<video snippet here…>

We finally got our form to submit data to the express server! Congrats!