**Node with Templates**

There are several ways to create web applications with Node js. Web application in node are creted suing some of the templates like

* EJS (Embedded JavaScript)
* Jade
* Handlebars

**EJS (Embedded JavaScript)**

There are several ways to create web applications with Node js. One popular way is to use jade. Before understanding jade, the basics of EJS should be known.

"E" is for "effective." EJS is a simple template language that lets you generate HTML markup with plain JavaScript.

The beauty of EJS is that, you can create partial views using EJS. For example you can have a common header, footer, navigation for all pages and just change the internal content using EJS.  
Also you are able to pass data to views.

**EJS Partials**

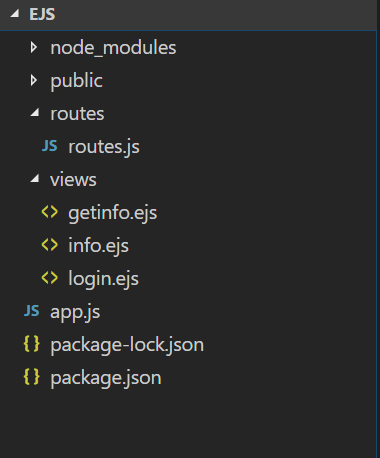
Partials come in handy when you want to reuse the same HTML across multiple views. Think of partials as functions, they make large websites easier to maintain as you don't have to go and change a piece of text in every page it appears in. Instead, you define that reusable bundle of code in a file and include it wherever you need it.

Let us start a project with EJS template.

Project details: login to web portal to know about some of the technologies.

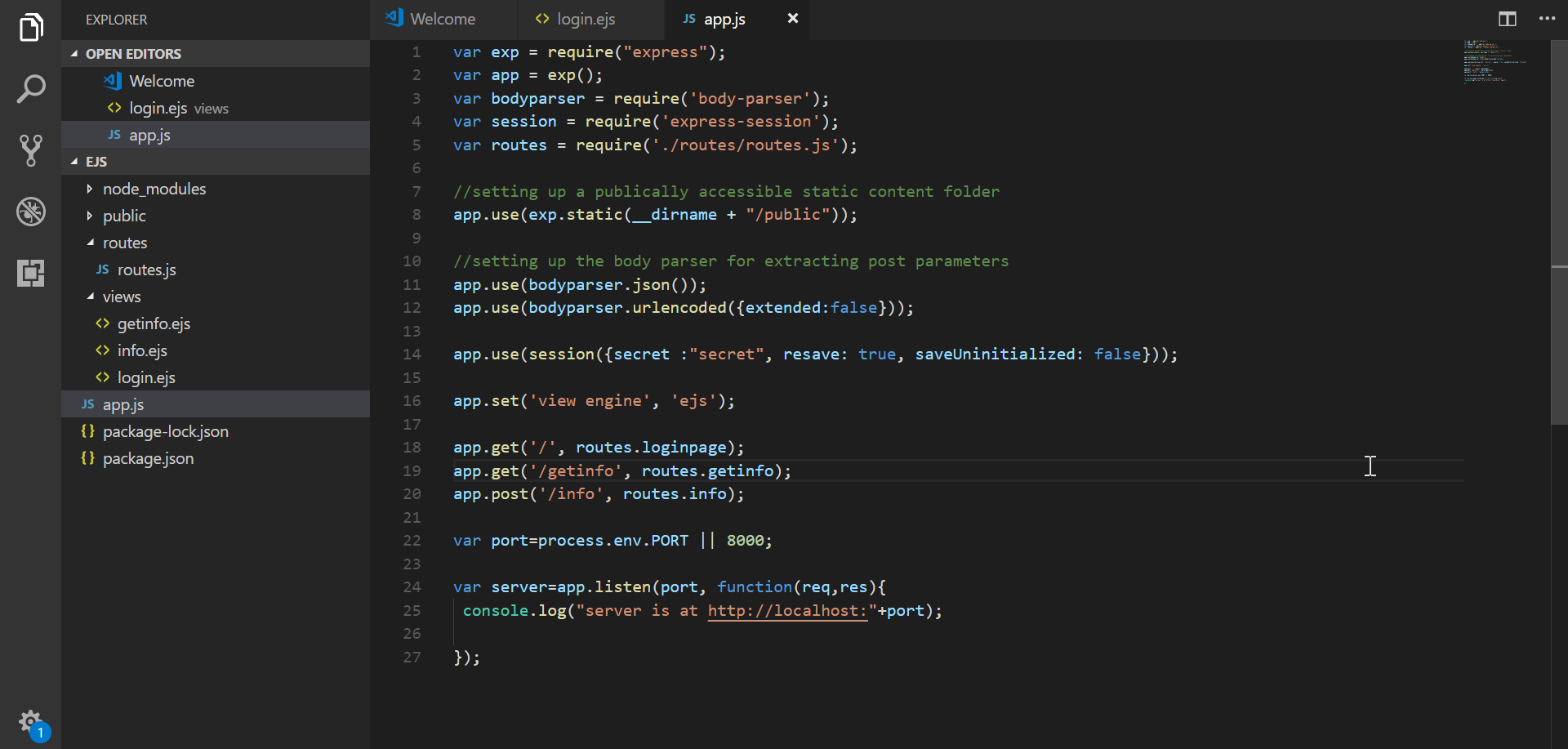
* we have login page
* Then, select the topic about you want know more about
* The information about selected topic would be displayed

Project structure:

****

* views/ will contain our EJS templates and the partials we will include
* app.js will have our Express configuration and routes
* package.json will maintain our dependencies (Express, EJS and express-session)

The project execution starts from app.js



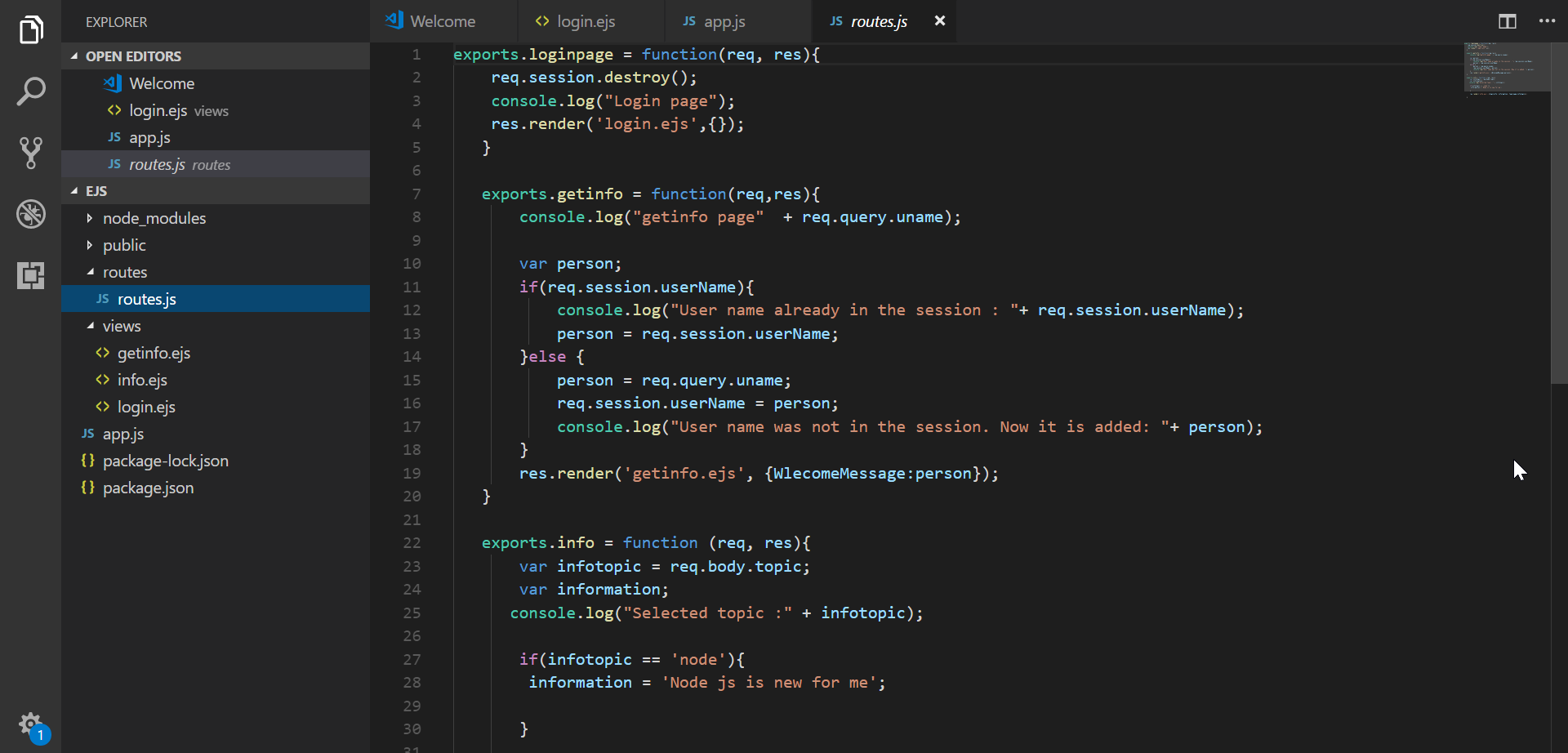
* Here from line 1 to line 4 is to get the third party module in our application.
* Line 8 is to setting up a publically accessible static content folder.
* Line 11 and 12 is for setting up the body parser for extracting post parameters.
* Line no 14 is for setting the session concept using express.
* Line no 16 is for third party module ejs template. The sentence says, we require ejs template with that we also adding ejs pages from views folder.
* The line no 18

app.get('/', routes.loginpage);

The above statement uses the variable of routes.js file, which we have included in our application at the line no 5.

var routes = require('./routes/routes.js');

The roues.js file is in inside the routes folder. The routes.js file is having all the function of the application.



Whenever the call to **app.get('/', routes.loginpage);** is made, the execution will jump to routes.js file and the loginpage function would be executed.

exports.loginpage = function(req, res){

req.session.destroy();

console.log("Login page");

res.render('login.ejs',{});

}

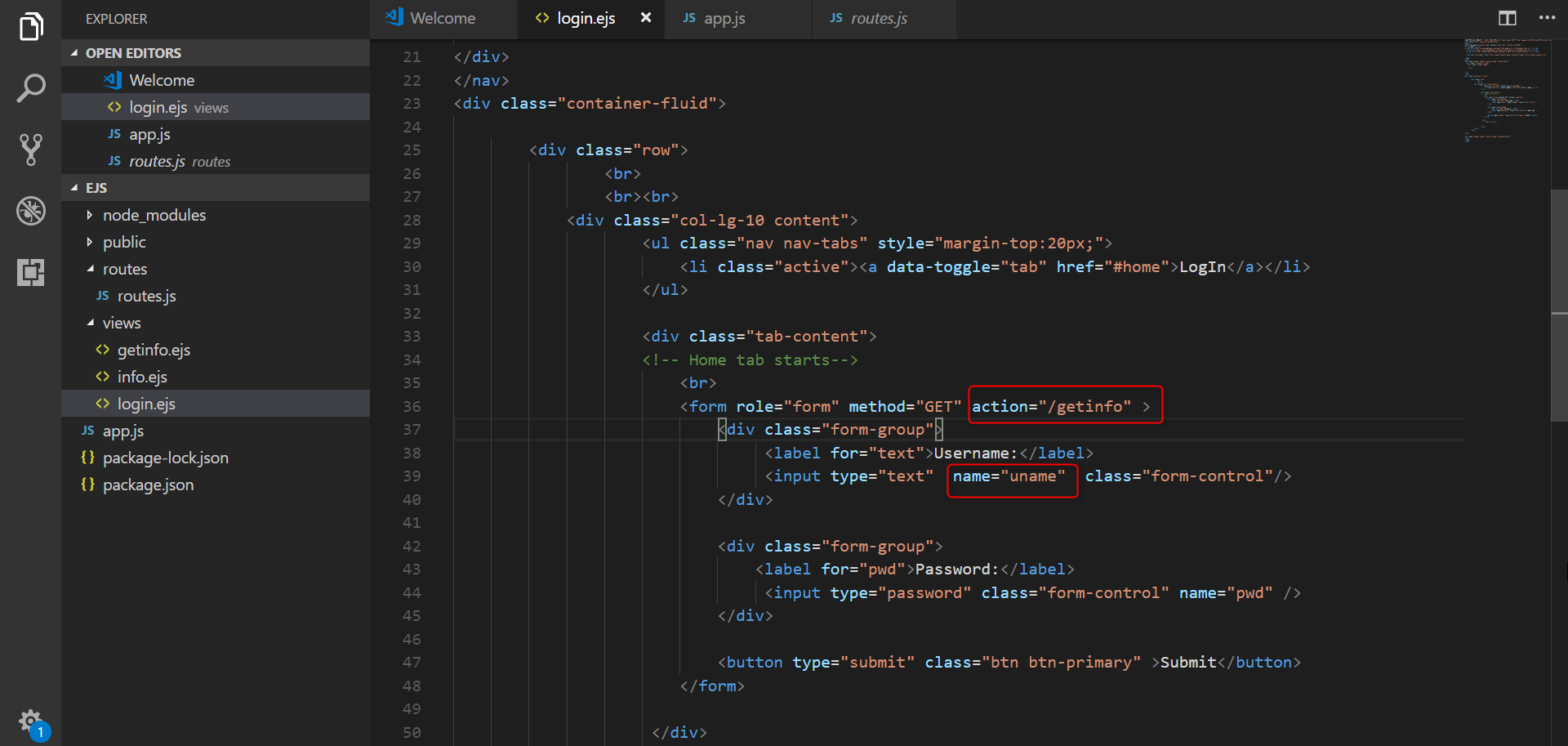
The control goes inside the loginpage function. If we have session which was already present, it will destroy by using the coder

req.session.destroy();

then next is to render the control to UI, that is calling login.ejs from views folder.

res.render('login.ejs',{});

here we are not passing any dynamic value to UI, so the empty flower brackets have been used. Instead here we are just displaying the login page. The code would be

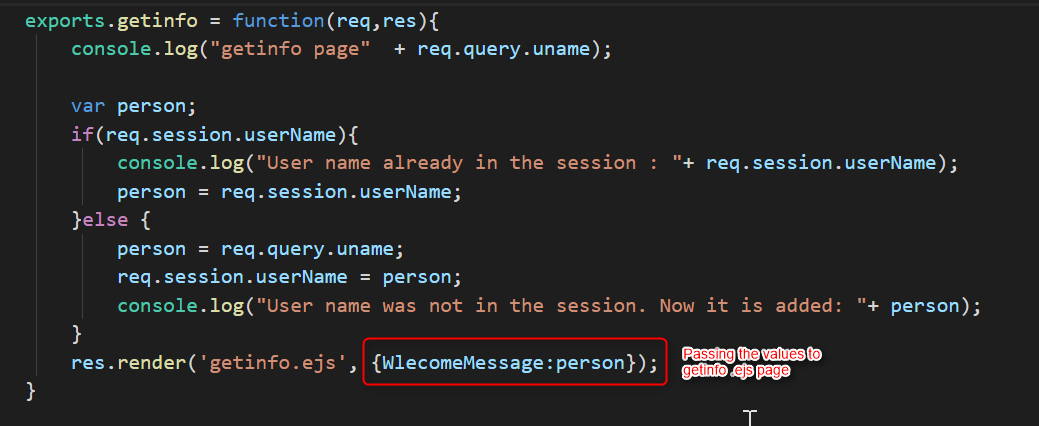


You should give the value for action attribute inside form tag. (shown in above fig). Based on the method and action attribute, the next line of code to be executed will be decided in app.js.

Once you give input to your Login page, the next line of code from app/js would be

app.get('/getinfo', routes.getinfo);

the execution will jump to routes.js file and the getinfo function would be executed.

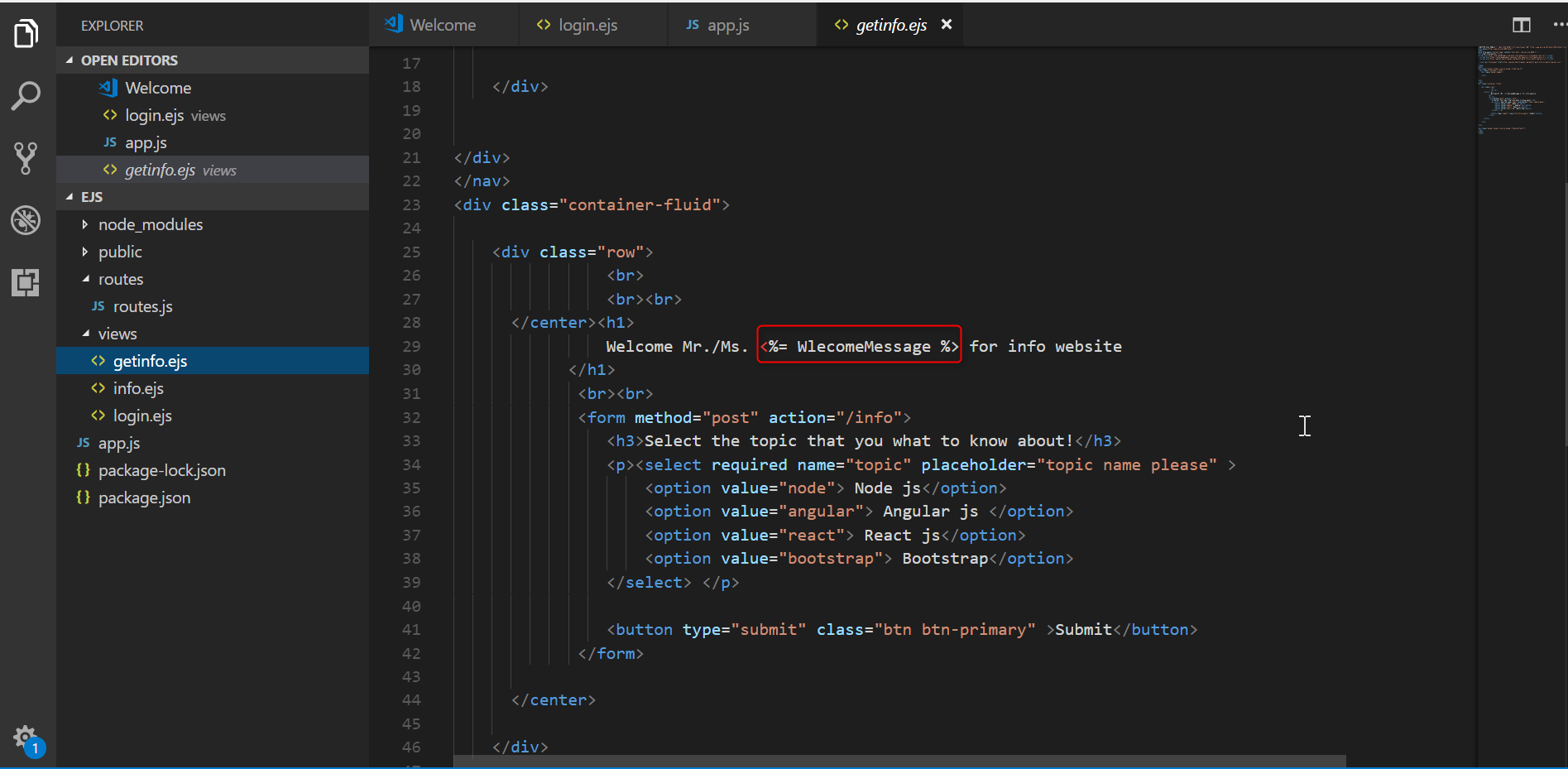


Here the necessary action will be taken, and at the end we are rendering to getinfo.ejs page by passing values.

res.render('getinfo.ejs', {WlecomeMessage:person});

here the line of code says, render to getinfo.ejs should be done by passing the variable WlecomeMessage, which will hold the value of person.

Use the variable WlecomeMessage in getinfo.ejs, to display it value.



Here the value of variable should be displayed suing the tag <%= variable\_name>.

Based on the method and action attribute, the next line of code to be executed will be decided in app.js.

Once you give input to your getino page, the next line of code from app/js would be

app.post('/info', routes.info);

the execution will jump to routes.js file and the info function would be executed.

exports.info = function (req, res){

var infotopic = req.body.topic;

var information;

console.log("Selected topic :" + infotopic);

if(infotopic == 'node'){

information = 'Node js is new for me';

}

res.render('info.ejs', {topicinfo: information, topicname:infotopic});

}

res.render('info.ejs', {topicinfo: information, topicname:infotopic});

here the line of code says, rendering to info.ejs should be done by passing the variable topicinfo and topicname which will hold the value of information, and infotopic variables.

Use the variable variable topicinfo and topicname in info.ejs, to display its value.

