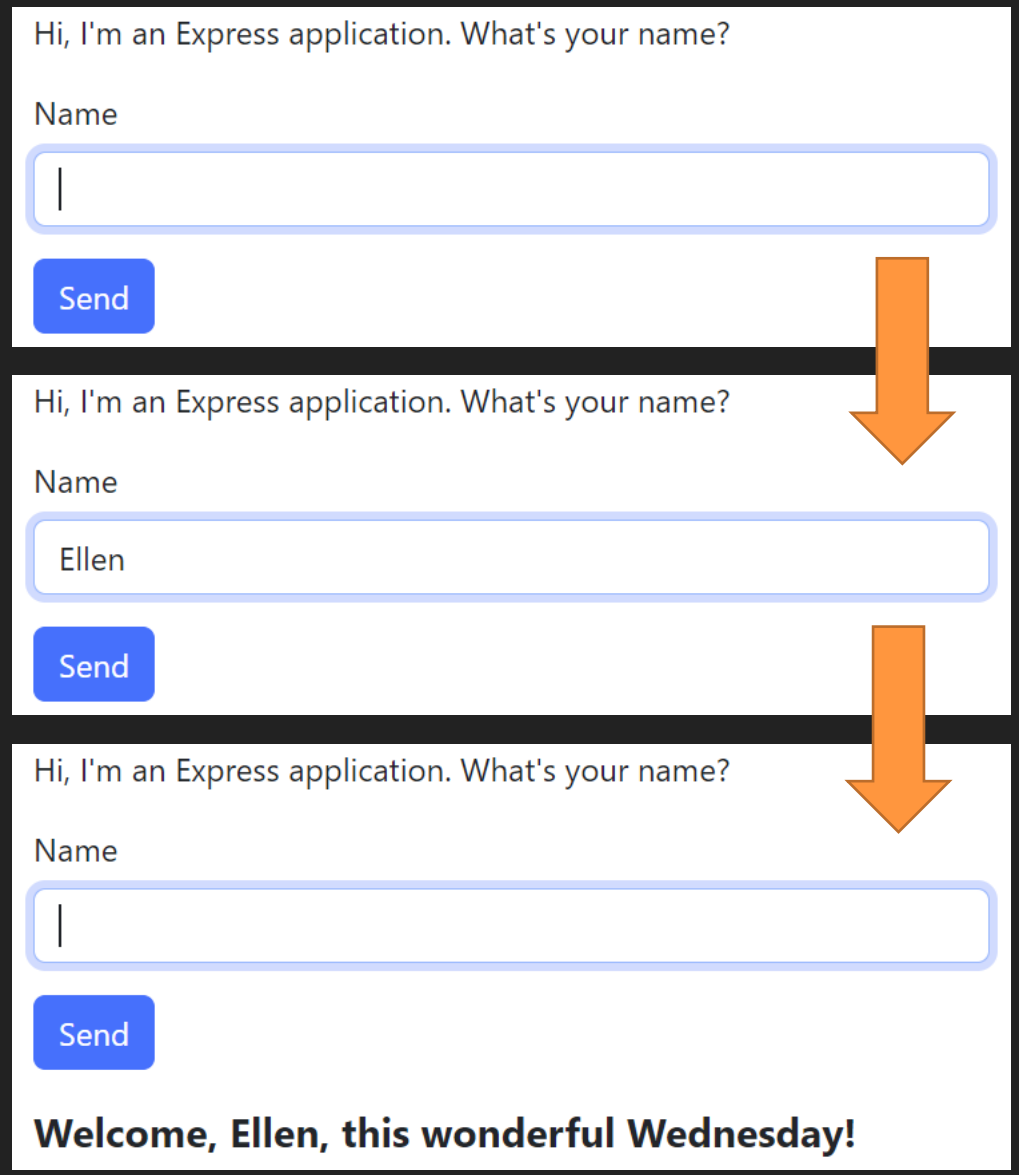
This work is licensed under CC BY 4.0

Your first Express application

Express – A Node.js Web Application Framework

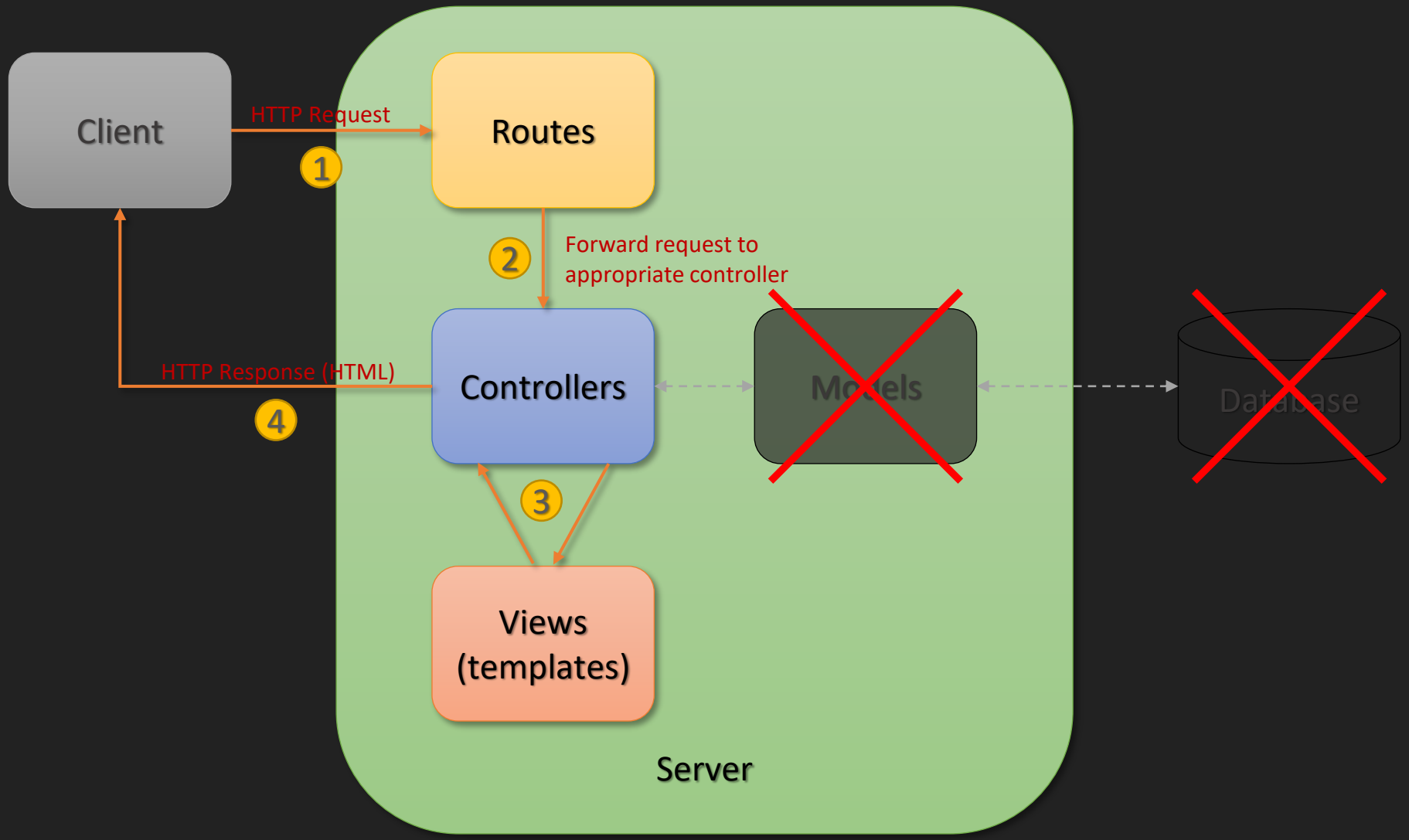


“Mockup” of Your First Express Application

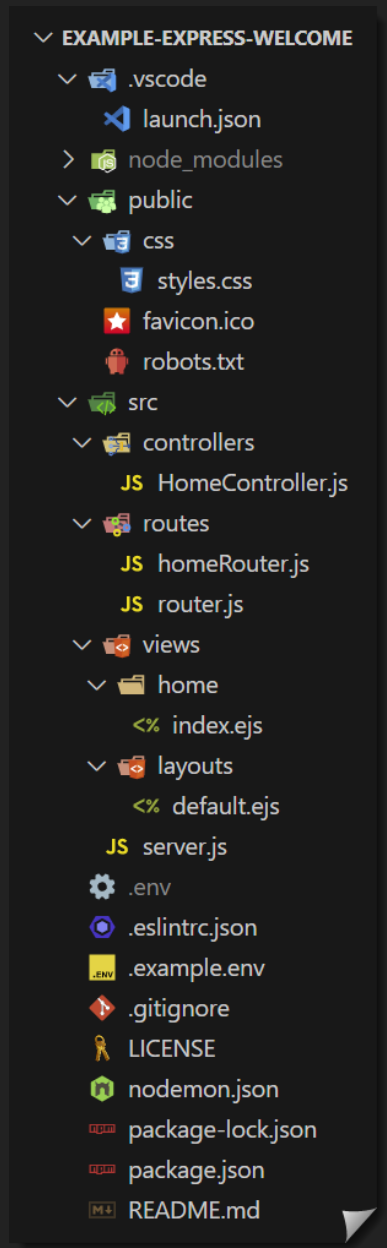


1. The user requests a suitable URL...
2. ...enters a name and posts the form.
3. The server responds with a welcome message.

Overview of Your First Express Application



The Final Directory Structure



- ✓ `server.js` is the application's entry point.
- ✓ Routes are stored in separate modules under the `routes/` directory.
- ✓ Controllers are stored under the `controllers/` directory.
- ✓ View templates are stored under the `views/` directory.
- ✓ Static files are stored under the `public/` directory.
- ✓ ...and there are several other opinions.

The dependencies and devDependencies – `package.json`

```
{
  "name": "example-express-welcome",
  "version": ”3.0.0",
  "description": "1DV026 - Example - Express Welcome",
  "type": "module",
  "main": "src/server.js",
  "engines": {
    "node": ">=20.6.0"
  },
  "scripts": {
    "start": "node src/server.js",
    "dev": "nodemon --inspect --env-file=.env src/server.js",
    "lint": "npx eslint ./src || exit 0",
    "lint:fix": "npx eslint ./src --fix || exit 0"
  },
  "license": "MIT",
  "private": true,
  "dependencies": {
    "date-fns": "^2.30.0",
    "ejs": "^3.1.9",
    "express": "^4.18.2",
    "express-ejs-layouts": "^2.5.1",
    "morgan": "^1.10.0"
  },
  "devDependencies": {
    "@lnu/eslint-config": "^1.1.10",
    "nodemon": "^3.0.2"
  }
}
```

- ✓ `date-fns`
  - Manage dates.
- ✓ `ejs`
  - A simple template engine.
- ✓ `express`
  - Web framework for Node.js.
- ✓ `express-ejs-layouts`
  - Layout support for EJS in Express.
- ✓ `morgan`
  - HTTP request logger.
- ✓ `nodemon`
  - Automatically restarts the application when file changes.

The Entry Point – `src/server.js`

```
import express from 'express'
import expressLayouts from 'express-ejs-layouts'
import logger from 'morgan'
import { dirname, join } from 'node:path'
import { fileURLToPath } from 'node:url'
import { router } from './routes/router.js'

// Get the path of the current module's directory.
const directoryFullName = dirname(fileURLToPath(import.meta.url))

/ Set the base URL to use for all relative URLs in a document.
const baseURL = process.env.BASE_URL || '/'

// Create Express application.
const app = express()

// Set up a morgan logger using the dev format for log entries.
app.use(logger('dev'))

// View engine setup.
app.set('view engine', 'ejs')
app.set('views', join(directoryFullName, 'views'))
app.use(expressLayouts)
app.set('layout', join(directoryFullName, 'views', 'layouts', 'default'))
app.set('layout extractScripts', true)
app.set('layout extractStyles', true)

// Parse requests of the content type application/x-www-form-urlencoded.
// Populates the request object with a body object (req.body).
app.use(express.urlencoded({ extended: false }))

// Serve static files.
app.use(express.static(join(directoryFullName, '..', 'public')))

// Middleware to be executed before the routes.
app.use((req, res, next) => {
  // Pass the base URL to the views.
  res.locals.baseURL = baseURL

  next()
})

// Register routes.
app.use('/', router)

// Error handler.
app.use(function (err, req, res, next) {
  console.error(err)

  res
    .status(err.status || 500)
    .send(err.message || 'Internal Server Error')
})

// Starts the HTTP server listening for connections.
const server = app.listen(process.env.PORT, () => {
  console.log(`Server running at http://localhost:${server.address().port}`)
  console.log('Press Ctrl-C to terminate...')
})
```

- ✓ The purpose of `server.js` is to create an instance of an Express application, configure it, and start it as Express server.
- ✓ Middleware functions are functions that have access to the request object (`req`), the response object (`res`), and the next middleware function in the application's request-response cycle.
  - Application-level: `app.use()` and `app.METHOD()`, `(req, res, next)`).
  - Router-level: same as application-level (use `express.Router()`).
  - Error-handling: an error-handling middleware function has four parameters, `(err, req, res, next)`).
  - Built-in: `express.static`, `express.urlencoded`
  - Third-party: Add functionality.

The Routing

```
// routes/homeRouter.js
import express from 'express'
import { HomeController } from '../controllers/HomeController.js'

export const router = express.Router()

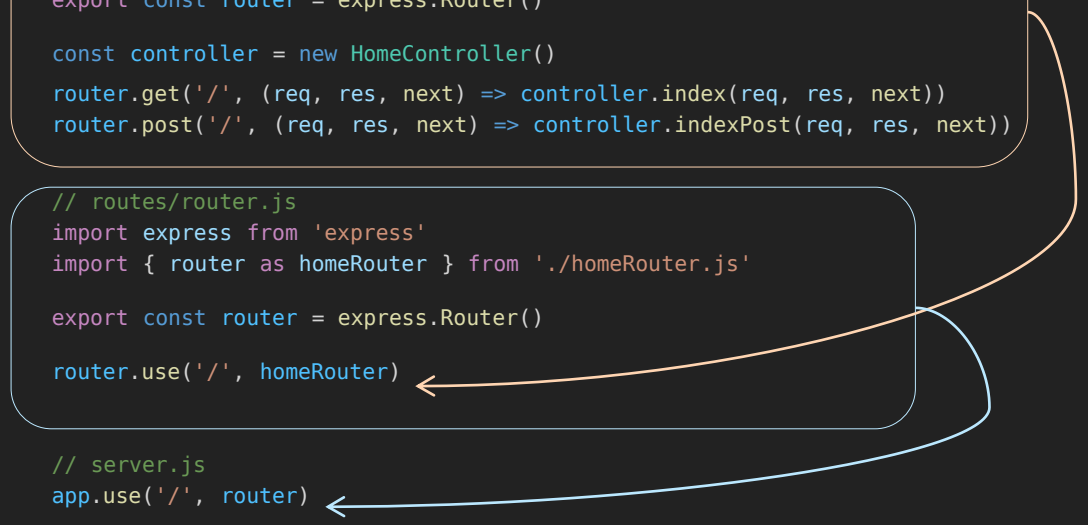
const controller = new HomeController()
router.get('/', (req, res, next) => controller.index(req, res, next))
router.post('/', (req, res, next) => controller.indexPost(req, res, next))
```

```
// routes/router.js
import express from 'express'
import { router as homeRouter } from './homeRouter.js'

export const router = express.Router()

router.use('/', homeRouter)
```

```
// server.js
app.use('/', router)
```



- ✓ The `router.get()` and `router.post()` methods are two routing functions in Express.
- ✓ You provide a function to be called for a request matching the method and specified path.
- ✓ You use the router object as an argument to `router.use([path], callback [,callback…])` and `app.use()`.
  - `router.use('/', router)`

    `http://localhost:8080/`
  - `router.use('/home', router)`

    `http://localhost:8080/home`

The Controller – `controllers/HomeController.js`

```
import { format } from 'date-fns'

/**
 * Encapsulates a controller.
 */
export class HomeController {
  /**
   * Renders a view and sends the rendered HTML string as an HTTP response.
   * index GET.
   *
   * @param {object} req - Express request object.
   * @param {object} res - Express response object.
   * @param {Function} next - Express next middleware function.
   */
  index (req, res, next) {
    res.render('home/index')
  }

  /**
   * Renders a view, based on posted data, and sends
   * the rendered HTML string as an HTTP response.
   *
   * @param {object} req - Express request object.
   * @param {object} res - Express response object.
   * @param {Function} next - Express next middleware function.
   */
  indexPost (req, res, next) {
    const viewData = {
      name: req.body.name,
      dayName: format(new Date(), 'iiii')
    }

    res.render('home/index', { viewData })
  }
}
```

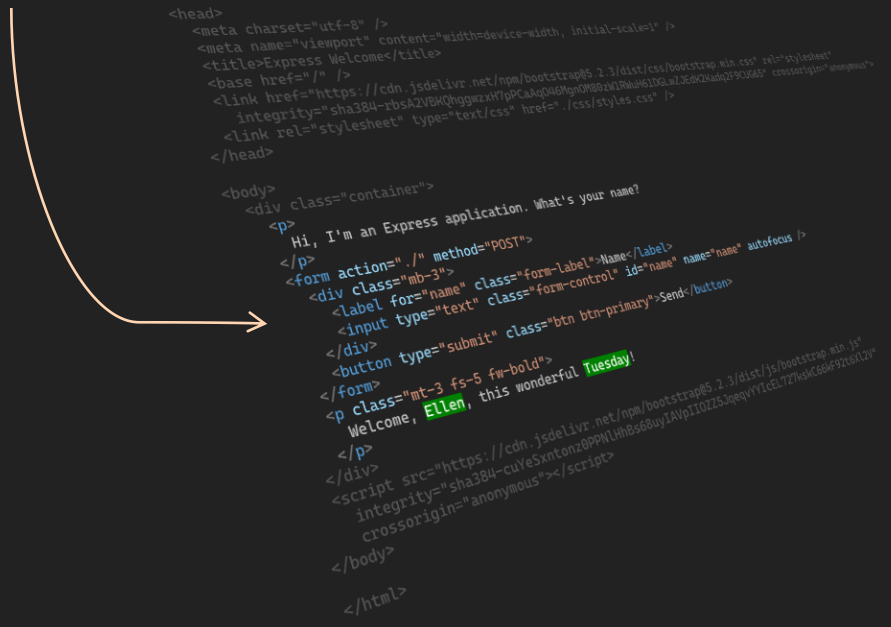
GET /

POST /

- ✓ Controller functions...
  - ...get the requested data (if any), ...
  - ...render an HTML page using a template engine, and...
  - ...return HTML to the client.
- ✓ The method `res.render(view [, locals])` renders a view and sends the HTML string to the client.
  - `view`, the path of the view file to render.
  - `locals` (optional), an object that contains data to pass from the controller to the view.

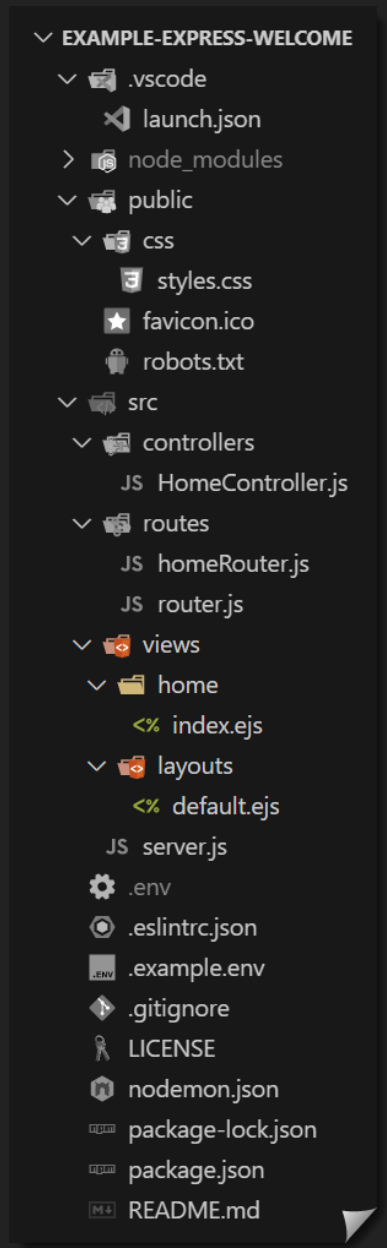
The View – `views/home/index.ejs`

```ejs
<p>
  Hi, I'm an Express application. What's your name?
</p>
<form action="./" method="POST">
  <div class="mb-3">
    <label for="name" class="form-label">Name</label>
    <input type="text" class="form-control" id="name" name="name" autofocus />
  </div>
  <button type="submit" class="btn btn-primary">Send</button>
</form>
<% if (locals.viewData) { %>
<p class="mt-3 fs-5 fw-bold">
  Welcome, <%= viewData.name %>, this wonderful <%= viewData.dayName %>!
</p>
<% } %>
```



- ✓ A template engine enables you to transform a template to HTML sent to the client.
- ✓ Express supports several template engines, such as Pug, Mustache, and EJS (Embedded JavaScript templating).
- ✓ You will take a closer look at EJS template engine.
  - Find out more about EJS at https://ejs.co/.
  - To get layout support you will use express-ejs-layouts.
  - Some tags:
    - <% “Scriptlet” tag, for control-flow, no output.
    - <%= Outputs the value into the template (HTML escaped).
    - <%- Outputs the unescaped value into the template.
    - <%# Comment tag, no execution, no output.
    - %> Plain ending tag.

Installation and Configuration



- ✓ Install [ejs](ejs) and express-ejs-layouts.
  - `npm install ejs express-ejs-layouts`
- ✓ Set the view engine as ejs and views directory for views. Configure the template engine to use `default.ejs` as the default layout.

```
import expressLayouts from 'express-ejs-layouts'

const directoryFullName = dirname(fileURLToPath(import.meta.url))

// View engine setup.
app.set('view engine', 'ejs')
app.set('views', join(directoryFullName, 'views'))
app.set('layout', join(directoryFullName, 'views', 'layouts', 'default'))
app.set('layout extractScripts', true)
app.set('layout extractStyles', true)
app.use(expressLayouts)
```

The Template Engine

controllers/home-controller.js

```
const viewData = {
  name: req.body.name,
  dayName: format(new Date(), 'iiii')
}

res.render('home/index', { viewData })
}
```

```
locals.viewData = {
  name: 'Ellen',
  dayName: 'Wednesday'
}
```

views/home/index.ejs

```
<p>
  Hi, I'm an Express application. What's your name?
</p>
<form action="./" method="POST">
  <div class="mb-3">
    <label for="name" class="form-label">Name</label>
    <input type="text" class="form-control" id="name" name="name" autofocus />
  </div>
  <button type="submit" class="btn btn-primary">Send</button>
</form>
<% if (locals.viewData) { %>
<p class="mt-3 fs-5 fw-bold">
  Welcome, <%= viewData.name %>, this wonderful <%= viewData.dayName %>!
</p>
<% } %>
```

https://ejs.co/

Template Engine

views/layouts/default.ejs

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Express Welcome</title>
  <base href="<%= baseURL %>" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
  <link rel="stylesheet" type="text/css" href="./css/styles.css" />
  <%- defineContent('style') %>
  <%- defineContent('script') %>
</head>

<body>
  <div class="container">
    <%- body %>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz"
    crossorigin="anonymous"></script>
</body>

</html>
```



```
// View engine setup.
app.set('view engine', 'ejs')
app.set('views', join(directoryFullName, 'views'))
app.use(expressLayouts)
app.set('layout', join(directoryFullName, 'views', 'layouts', 'default'))
```

