Project Scaffolding, Step by Step

Contents

Express Scaffolding

1. In your terminal, run the following to scaffold a new Express application called "my-express-app": `npx express-generator --no-view my-express-app`. (It may ask the first time if it can install `express-generator`. Say "yes," of course.)
2. `cd` into the new folder and install dependencies: `npm install`
3. Install packages that you'll almost certainly need, such as MySQL, Nodemon, Dotenv and CORS: `npm install mysql nodemon dotenv cors`
4. Add the following two lines to the server `app.js`:

```
const cors = require('cors');  // add at the top

app.use(cors());  // add after 'app' is created
```

5. Comment out the following line in `app.js` (around line 17):

```
app.use(express.static(path.join(__dirname, 'public')));
```

6. Change the homepage route in `routes/index.js` to this: `res.send({ title: 'Express' });`
7. Copy the `model` folder from a previous DB activity. This contains the `helper.js` file, which contains a nice wrapper around DB connections, so you can use the `db()` function from within your code. It also contains the `database.js` file, which is the *migration file* for your project that you use to (re)create your DB tables and sample data.
8. Modify the start script in `package.json` so it uses nodemon instead of node: `"start": "nodemon ./bin/www"`
9. Add a new script to your `package.json` file that you will use to run your migrations: `"migrate": "node model/database.js"`. When you want to (re)create your DB tables, run `npm run migrate`
10. In the file `./bin/www`, change the default port from 3000 to 5000 (around line 15)
11. If you need to store private data and passwords (such as your DB connect info), create a `.env` file in the Express project directory.
12. From the project directory, initialize Git for your app: `git init`
13. Add a `.gitignore` file to your project. It should contain at least these: `node_modules/`, `.env` and `.DS_Store`.

14. Do `git add .` and `git commit -m "Initial Express commit"` to commit your initial Express files.
15. Happy (back-end) coding!

React Scaffolding

1. In your terminal, run the following to scaffold a new React application called "my-react-app" using vite: `npm create vite@latest my-react-app`. Follow the prompts and choose React. NOTE: If you are creating a full-stack app, call the app "client" and create it in the Express folder.

Set Up Proxy for Full-Stack Development

If you are creating a full-stack app, do these steps so the client can "find" the server.

1. Open the configuration file `vite.config.js`, and update the code to the following:

```
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      "/api": {
        target: "http://localhost:5000",
        changeOrigin: true,
        secure: false
      },
    },
  },
})
```

This means that your server is listening to port 5000 and all of your back-end routes must begin with `/api`; that's a good thing.

Everyone Do This

Do this final step, regardless if you're building a full-stack or front-end app.

1. Do `git add .` and `git commit -m 'Initial React commit'` to commit your changes.
2. Happy (front-end) coding!

Push Your Repository to GitHub

Once you have created the scaffolding for front end and/or back end, you'll want to connect it to a repo on GitHub.

1. On your GitHub page, select the `+` sign in the top right corner, and select `New repository`.
2. Choose your project name.
3. Do *not* select Add a README file.
4. Click on `Create repository`.

5. Follow the instructions for "...or push an existing repository from the command line" by doing a copy/paste of those three commands into a terminal in your project folder on your computer.
6. Invite your instructor as a collaborator: Go to the Settings tab of the repo on GitHub, then choose Collaborators in the left column, and press the green Add People button. Give her/him admin access.
7. Happy coding!

Updated: 21 Feb 2023