Express #4: Route Organization

Route Overload

- In our Express projects, we've started to end up with a *lot* of routes
- Many of them are related to a few routes (country app, portfolio stuff) but not to all of them
- But right now they're all in the same big file...
- Is there a way to split them out in to multiple files?

Method 1: Route "functions"

- All a route needs is the app variable to call get (or the others) on
- So we can export a **function** that takes in app as an **argument**
- This function can attach all of routes we need
- It can also do all of the route-specific stuff, such as loading the countries JSON for the country app

Method 1: Route "functions" (code)

```
// routes/portfolio.js
module.exports = function(app) {
    app.get("/", function(req, res) {
        res.render("home");
    });

    app.get("/gallery", function(req, res) {
        res.render("gallery");
    });
}

// app.js
const express = require("express");
const portfolioRoutes = require("./routes/portfolio");
const app = express();

portfolioRoutes(app); // Runs all of the app.get's from above

app.listen(3000);
```

Method 2: express.Router

- Since Express v4.0, Express has added a `Router` object
- Much like app, we create a `Router` object by calling `express.Router`
- A `Router` has the same method type functions as an app for setting up routes, as well as a `.use()` function that only sets configuration for the `Router`'s routes
- We can then tell app to `.use()` one of these routers, and put it under a specific root path

Method 2: express.Router (code)

```
// routers/portfolio.js
const express = require("express");
const router = express.Router();

router.get("/", function(req, res) {
    res.render("home");
});

router.get("/gallery", function(req, res) {
    res.render("gallery");
});

module.exports = router;
```

Method 2: express.Router (more code)

```
const express = require("express");
const portfolioRouter = require("./routers/portfolio");
const countryRouter = require("./routers/country");
const blogRouter = require("./routers/blog");

const app = express();

app.use("/", portfolioRouter);
app.use("/country", countryRouter);
app.use("/blog", blogRouter);

app.listen(3000);
```

Need To Share Things?

- Some people may have made helper functions for things like rendering the template, rendering errors, etc.
- Now that your routes are in different files, you can't just define those functions in app.js and use them everywhere
- **And you definitely don't want to copy/paste the function to the new files.** This is a *programming anti-pattern*, because it will be difficult to update the function in multiple places.
- Instead, turn that function in to a module!

Need To Share Things? (code)

```
// util/renderTemplate.js
module.exports = function(res, page, title, args) {
    res.render("template", {
        page: page,
        title: title,
        pageArgs: args,
    });
};

// router/portfolio.js
const renderTemplate = require("../util/renderTemplate");

router.get("/", function(req, res) {
    renderTemplate(res, "home", "Home", {
        name: "Will O.",
        skills: [
            "HTML & CSS",
            "jQuery",
            "Express"
        ],
    });
});
```

Exercise: Break Out Your Routes

- Take whatever express application you've most recently worked on (Country app, maybe?)
- Even if you only have one group of routes, make a routers/ folder and make a router for it
- Make modules for any shareable functions in a `util/` folder
- Have your app.js require the new router and `.use()` it

Additional Reading

- Express' guide to express.Router()