Express Routes

Made by :

Teams

Ahmed Hamdy [Link]
Mina Sameh [Link]
Ahmed Abu Qahf [Link]

Overview:

I. **Route Model**

II. **Routes Primer**

III. **Define and using separate route modules**

IV. **Route Functions**

V. **HTTP Verbs**

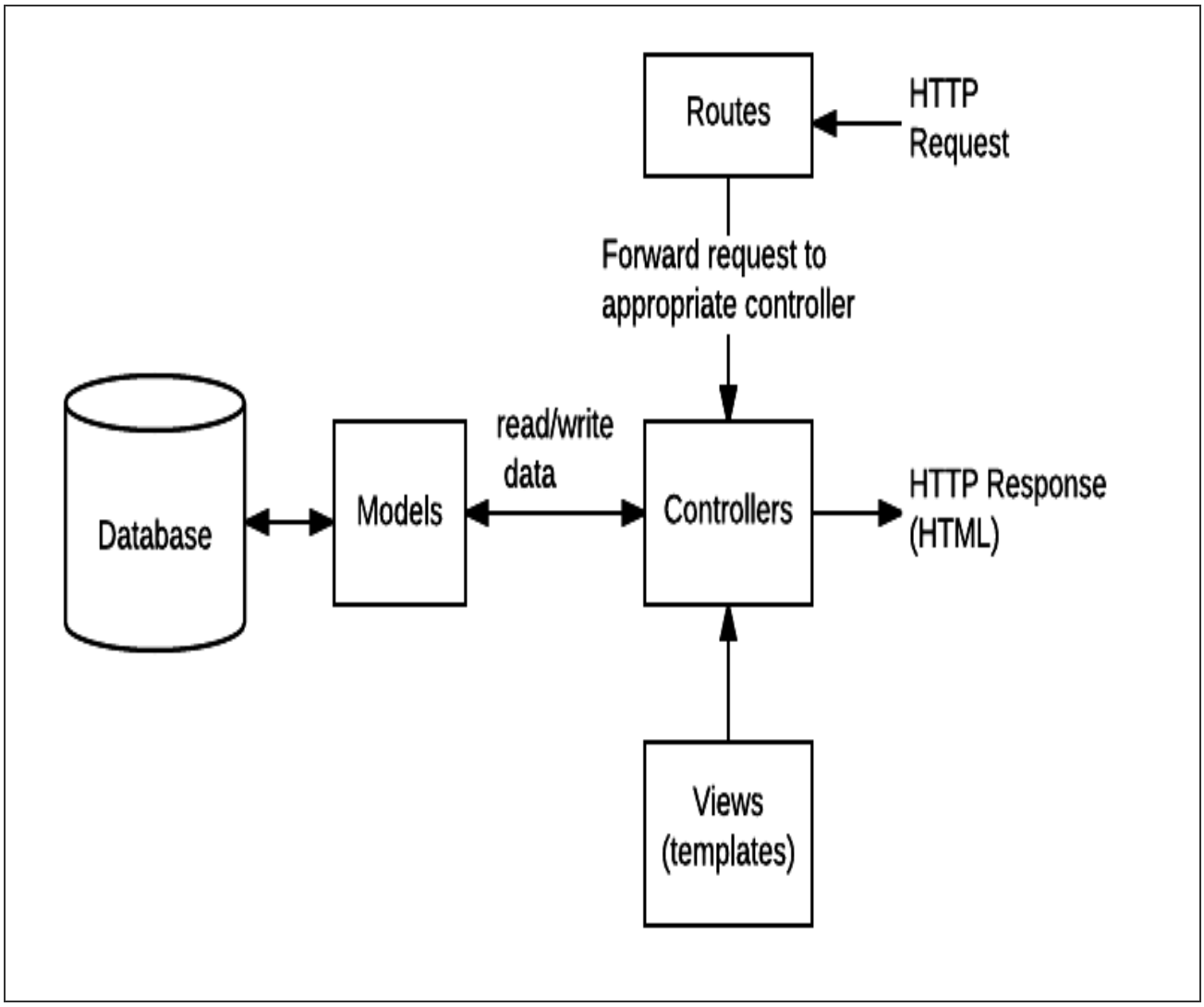
VI. **Route Path**

VII. **Route Parameters**

VIII. **Route Handlers**

IX. **Routes Naming Conventions**

Route Model



Route Primer

A web server typically divides up the website it serves into sections, called *routes*. For example, the server for `web.mit.edu` might have routes for:

- `/education` (accessible by the URL `http://web.mit.edu/education`)
- `/research`

- /campus-life

and so forth.

In a web API, the route often defines the *function name* of the request. For example, the US National Weather Service API defines routes for:

- https://api.weather.gov/points/... to get information related to a latitude/longitude point
- https://api.weather.gov/gridpoints/... to get the forecast for a particular area
- https://api.weather.gov/alerts/active... to get weather-warning messages for a particular area

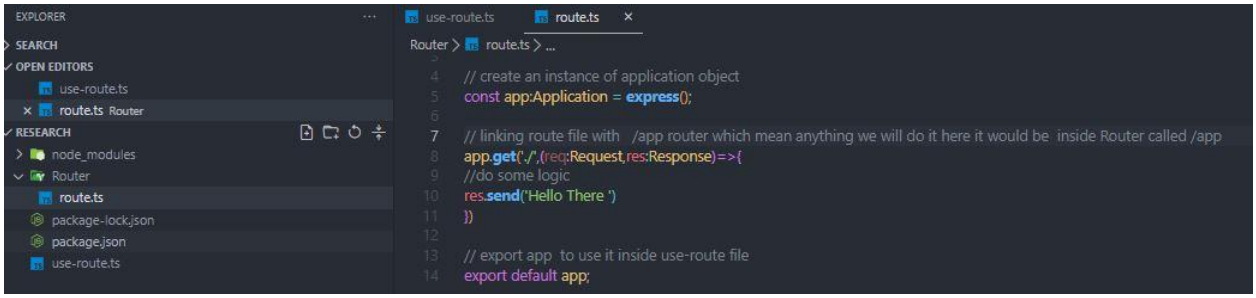
Also a route is a section of Express code that associates an HTTP verb (GET, POST, PUT, DELETE, etc.), a URL path/pattern, and a function that is called to handle that pattern, There are several ways to create routes example:

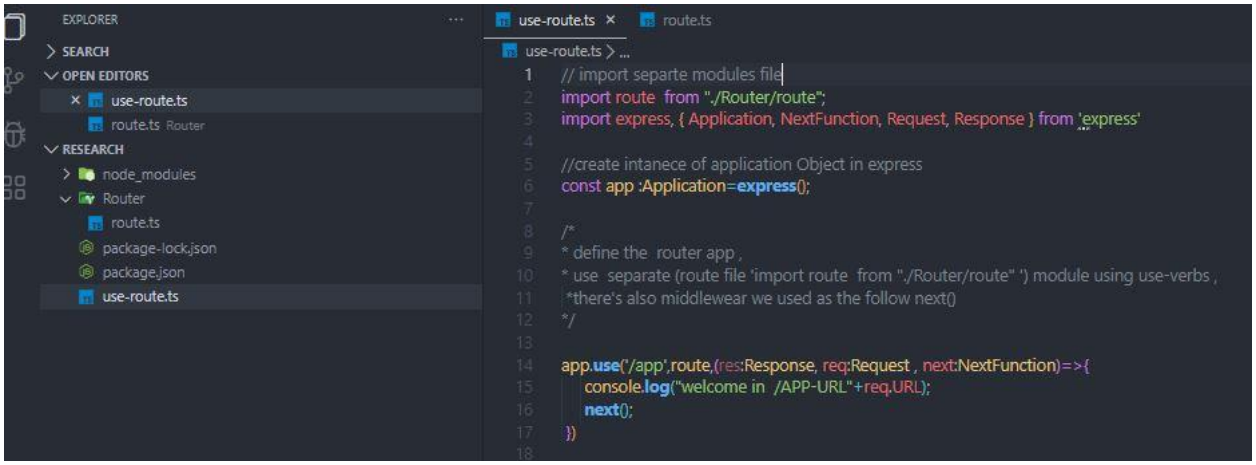
```
const express = require('express')
const app = express()

// respond with "hello world" when a GET request is made to the homepage
app.get('/', (req, res) => {
  res.send('hello world')
})
```

Define and using separate route modules

The code below provides a concrete example of how we can create a route module and then use it in an *Express* application file :





Route Functions

A route method is derived from one of the HTTP methods, and is attached to an instance of the express class like : (GET)

Router functions are also Express Middleware, which means that they must either complete (respond to) the request or call the next function in the chain. In the case above, we complete the request using send(), so the next argument is not used (and we choose not to specify it).

The router function above takes a single callback, but you can specify as many callback arguments as you want, or an array of callback functions. Each function is part of the middleware chain, and will be called in the order it is added to the chain (unless a preceding function completes the request)

HTTP Verbs

HTTP verbs it's HTTP method is supplied in the request and specifies the operation that the client has requested.The following examples lists the most HTTP methods : post(), put(), delete(), options(), trace(), copy(), lock(), mkcol(), move(), purge(), propfind(), proppatch(), unlock(), report(), mkactivity(), checkout(), merge(), m-search(), notify(), subscribe(), unsubscribe(), patch(), search(), and connect().

Route Path

The Route path along with the Route method, is the endpoint which requests are made to, however *query* strings are **not** part of the path. They can be strings, string patterns or regular expressions.

For example, a "/api/checkhealth" is a Path Example of regular expressions as paths in Express: app.get(/.*fly$/, handler) will match to *butterfly* and *dragonfly* but not *flyswatter*. **Note: the $ means end of sentence, and * means match anything in regular expressions.*

Route Parameters

They are named segments in the URL, that capture values at their position in the url, Then the values are populated in the req.params , with the name of the parameter as key.

Ex:

Route path: “/api/book/:id”
Request URL: http://localhost:8000/api/book/30
Req.params: { id: 30 }
Express code: app.get(“/api/book/:id”, handler)

**Note: The route parameter must be of word characters (A-Z, a-z, 0-9, _)*
Since the `-` and `.` are ignored from params, you can use them like this:

Route path: “/api/flights/:from-:to”
Request URL: http://localhost:8000/api/flights/EGY-USA
Req.params: { “from”: “EGY”, “to”: “USA” }
Express code: app.get(“/api/flights/:from-:to”, handler)

Route Handlers

They are callback functions that behave like middlewares, we can use them to impose conditions on route, before the business logic and response is sent, like verifying login.

They can be in the form of function or array of functions.
Ex:

```
firstHandler = (req: Request, res: Response, next: NextFunction ) => {
    console.log('This will run first!')
    next();
}
secondHandler = (req: Request, res: Response) => {
  return res.send('This is the response.')
}

app.get('/api/path', firstHandler, secondHandler)
```

Routes naming conventions

1. Use plural nouns to represent a route (resource) for example /api/books
2. To get a single document, use singular nouns like this /api/books/:book_id
3. Use verbs to denote a controller resource of a route like cart checkout for example /api/cart-mgmt/users/:user_id/checkout
4. Use forward slash to indicate hierarchy for example /api/users/:id/games/:game_id and so on.
5. Use hyphens (-) to improve readability for example /api/user-devices.
6. Do not use underscores because depending on the font they can either be obscured or hidden by the browser, use hyphens instead.
7. Use lower case letters.
8. Do not use file extensions like this /api/users.json, use this instead /api/users as adding the file extension do not add any advantage.
9. Do not use CRUD functions names like /api/get-all-users as URIs should only be used to uniquely identify the resources and not any action upon them.
10. Use *request.query* for filtering collections for example to set a limit for the returned data, for setting the format of the return type (JSON, XML…etc.) and so on.
    For example:
    /api/users?limit=50 // will return only 50 users
    /api/countries?region=mena // will return countries in the middle east
    /api/devices?format=json // will return results in a json format

References

- Massachusetts/MIT.edu
- MDN WEB DOCS
- Express Official Docs
- REST Resource Naming Guide

Other good resources For Software Engineer and students:

- Type racer for fast typing code
- CS-50
- Software Engineer Interviews over (GOOGLE-APPLE-Microsoft…etc) with TechLead
  Get the first episode by email