Introduction to NodeJS
Week 1 - Day 5

Topics Covered

1. A Simple Server using Express
2. Serving Static Files
3. Setting up a REST API
4. Using Express Router

Introduction to Express

What is Express?

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. It provides mechanisms to:

- ➔ Write handlers for requests with different HTTP verbs at different URL paths (routes).
- ➔ Integrate with "view" rendering engines in order to generate responses by inserting data into templates.
- ➔ Set common web application settings like the port to use for connecting, and the location of templates that are used for rendering the response.
- ➔ Add additional request processing "middleware" at any point within the request handling pipeline.

While Express itself is fairly minimalist, developers have created compatible middleware packages to address almost any web development problem. There are libraries to work with cookies, sessions, user logins, URL parameters, POST data, security headers, and many more.

Installing Express

1. Navigate to the directory `*node-examples*` where you initialized the `*package.json*` through the command `$ npm init`

   ```
   $ npm install express --save
   ```

(Source)

A Simple Server using Express

Resources

- Video mode
- Reading mode
    - Resource 1
    - Resource 2

Serving Static Files

Resources

- Reading mode
    - Resource 1
    - Resource 2
    - Resource 3

REpresentational State Transfer (Optional)

Guiding Principles of REST

- Client-server – By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
- Stateless – Each request from the client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client.
- Uniform interface – In order to obtain a uniform interface, multiple architectural constraints are needed to guide the behaviour of components. REST is defined by four interface constraints: identification of resources; manipulation of resources through representations; self-descriptive messages; and, hypermedia as the engine of application state.

- Layered system – The layered system style allows an architecture to be composed of hierarchical layers by constraining component behaviour such that each component cannot “see” beyond the immediate layer with which they are interacting.

Other interesting reads

- https://www.codecademy.com/articles/what-is-rest

Setting up a REST API using Express

body-parser module

body-parser is the Node.js body parsing middleware. It is responsible for parsing the incoming request bodies in a middleware before you handle it.

Some interesting reads

Resources

- Video mode
- Reading mode
  - Resource 1
  - Resource 2

Using Express Router

Resources

- Video mode (Same as above)
- Reading mode
  - Resource 1
  - Resource 2
  - Resource 3