CMSC335

Web Application Development with JavaScript

Express

Department of Computer Science

University of MD, College Park

Slides material developed by Ilchul Yoon, Nelson Padua-Perez

Important (run npm i in folder with examples)

- We are removing the **node_modules** folder from the lecture examples we are posting
- Before you run any code examples, execute **npm i** (not npm init)
- npm i (or npm install) will install in the **node_modules** folder any necessary modules (based on the file package.json)

Express

- **Express** is an abstraction layer on top of Node's http-server
- **Express** simplifies the implementation of tasks that otherwise will require significant effort using the **http** module
- What **Express** provides:
  - **Extensions** - The basic **request** and **response** objects have extra functionality
  - **Middleware**
    - » Functions **Express** executes in the middle after the incoming request and before the output
    - » Might make changes to the **request** and **response** objects
    - » The use() function is used to register middleware
  - **Routing** - Routing allows us to associate an URL and an HTTP method with some functionality
  - **Views** - Dynamic generation of HTML

Installing Express Module

- Let's install **Express** and save it as a dependency to **package.json by** executing the following command in the previous folder (example) we created
  - **npm install express --save**
    - » Note: As of Node 5.0.0 installed modules are added as a dependency by default, and you don't need the --save
- After installing, you will see a directory called **node_modules** (let's take a look)

Express Example

- **Example:** expressExample.js
  - To run, execute **node expressExample.js**
  - In the browser, type the URL you see in the node console

Middleware

- Middleware is a function
- In node a single function processes the request; using middleware the request can be processed by several functions
- **For example:**
  - One function can do authentication
  - One function can do logging
- A request does not need to be processed by every middleware function (any of them could provide a response). If none provides a response the server will hang
- A middleware function can modify the **request** or **response** objects
- In **app = express()**, app is a function that goes through the set of functions that are part of middleware stack
- **app.use** allow us to add middleware functions to the middleware stack
- **Example:** middleware.js
  - To run, execute **node middleware.js**
  - In the browser, type the URL you see in the node console

Logger example

- We can log requests using a third-party logger
- Installing morgan
  - **npm install morgan**
- writeHead is used with text/html
- **Example:** loggingHTML.js
  - To run, execute **node loggingHTML.js**
  - In the browser, type the URL you see in the node console

Serving Static Files

- **express.static** - part of **Express**
  - Allow us to serve files
- **path**
  - built-in module we use to generate a cross-platform (Windows, Mac, Linux) path
- **Example:** servingFiles.js
  - To run, execute **node servingFiles.js**

Additional Functionality to request/response

- **Express** expands the **request** and **response** objects
- **request.ip** - ip address
- **request.get** - to obtain HTTP headers
- **request.status** - to set status code
- **request.send**
- **response.redirect**
  - Redirects to a particular site
- **response.sendFile**
  - To send a file
- **response.json** - sending JSON response
- **Example:** additionalFunc.js (redirect)
  - To execute, type node additionalFunc.js

HTTP Verbs/Methods

- An HTTP request has a method/verb associated with it
- HTTP Methods
  - **GET**
    - » Gets a resource
    - » Most common method used
    - » Idempotent (executing many times does not cause server change)
  - **POST**
    - » Generates a change of server state (e.g., you bought an item)
    - » Non-idempotent
  - **PUT**
    - » To update or change
    - » Idempotent
  - **DELETE**
    - » To remove a resource
    - » Idempotent
  - **PATCH**
    - » Can be used to update

HTTP Verbs/Methods

- You can use **Express** to handle different HTTP verbs
- **curl** application enables you to generate http requests with different methods/verbs. You will find it in most systems (no need to install it). Just in case (https://curl.haxx.se/download.html)
- **Example:** httpMethods.js
  - To execute, type **node httpMethods.js**
  - In the browser, type the URL shown in the node console
- You can issue requests using curl. For example, using PC's cmd and assuming port 8001
  - GET → curl http://localhost:8001
  - POST → curl –X POST http://locahost:8001
  - PUT → curl –X PUT http://locahost:8001
  - DELETE → curl –X DELETE http://locahost:8001
  - **In PowerShell** use **curl –Method Get** or **curl –Method Post** **or** **curl –Method Put** or **curl –Method Delete**
- API Client/Design tools (allow you to issue http requests, among other things)
  - Insomnia - https://insomnia.rest/products/insomnia
  - Postman - https://www.postman.com/downloads/

Routing

- **Routing** - Mapping a URI and HTTP verb to a request handler
- In **Express,** you specify routes using strings and can specify them as regular expressions
- **Example:** routing.js
  - To execute, type **node routing.js**

Dynamic Generation of HTML

- **View/templating engines** - Allows you to generate dynamic HTML
- **EJS (Embedded JavaScript) engine** - templating engine that compiles/generates HTML for you
- **EJS is a superset of HTML**
- Files with the **.ejs** extension are placed in a folder where **Express** can locate them
- To install ejs
  - **npm install ejs**
- Interpolate variables in a template file by using:

  <%= variableName %>
- Inclusion of ejs file in another by using:

  <% fileNameWithoutEJSExtension %> // Notice no = in <%
- **Example:** dynamicHTML.js, templates/welcome.ejs
  - To run, execute **node dynamicHTML.js**

Retrieving Query Arguments

- We can use **request.query.<ARGUMENT_NAME>** to retrieve arguments provided in the URL (e.g., GET)
- **Example:** formGet.html, queryArguments.js, templates/courseInfo.js
  - To execute, type **node queryArguments.js**

Retrieving values associated with POST

- The **body-parser** module allows you to retrieve parameters submitted using post
- To access a parameter: request.body.<PARAMETER_NAME>
- To install the **body-parser** module
  - **npm install body-parser**
- **Example:** formPost.html, postParameters.js, templates/courseInfo.js
  - To execute, type **node postParameters.js**
  - Open **formPost.html** in the browser and provide some data
- Aside: You can clear form data (and other data) using Chrome's Clear Cache Extension. See https://www.cs.umd.edu/~nelson/classes/resources/web/ and look for "Clear Cache Extension")
  - After running the extension, you can clear previous entries typed in a text field
  - Can also be used for clearing cached CSS

Retrieving Form Data

- **Example:** Retrieving data sent via get
  - To execute, type **node formsSummaryGet.js**
  - In the browser, open **formsSummaryGet.html** and provide data
- **Example:** Retrieving data sent via post
  - To execute, type **node formsSummaryPost.js**
  - In the browser, open **formsSummaryPost.html** and provide data

Nodemon

- **nodemon** utility restarts the server after a modification has taken place
- To run: nodemon <application>
  - **nodemon .\queryArguments.js**
  - Modify queryArguments.js to see server restarted
  - Installation: npm i nodemon
    - » Can use --save-dev to save as a development dependency in package.json
    - » npm i --save-dev nodemon

References

- **Express in Action**

  Writing, building, and testing Nodes.js applications

  Evan M. Hahn

  April 2016 , ISBN 9781617292422