

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

Express

- **Express** is an abstraction layer on top of Node's http-server
 - Similar to JQuery is to JavaScript
- **Express** simplifies 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** - Instead of a single function handling the requests, a stack of functions (middleware stack) is available. This allows organizing the processing into separate functions
 - **Routing** - Routing allow us to associate an URL and a HTTP method with some functionality
 - **Views** - Dynamic generation of HTML

Creating a Project in Node

- A Node project has a file called **package.json** providing information such as project's name, author, version, and dependencies (which modules your project relies on)
- You can create this file yourself or you can rely on **npm** init
- **Example:** Let's create a project
- To install **Express** and save it as dependency to **package.json**
 - **npm install express –save**
 - » Note: As of Node 5.0.0 installed modules are added as a dependency by default
- After installing you will see a directory called **node_modules**
- **Example:** example1.js
 - To run type **node example1.js**
 - In the browser type <http://localhost:8000/>

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**
- 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 type **node middleware.js**
 - In the browser type <http://localhost:5000/>

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 type **node loggingHTML.js**
 - In the browser type <http://localhost:7000/>

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 type **`node servingFiles.js`**
 - In the browser type <http://localhost:7001/Testudo.jpg>

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
 - To run type node additionalFunc.js
 - Try in the browser
 - » <http://localhost:7002/Testudo.jpg>
 - » <http://localhost:7002/>

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**
 - » Relatively new
 - » Can be use 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 run type **node httpMethods.js**
 - In the browser type <http://localhost:8001/>
- You can issue requests using curl. For example, using PC's cmd:
 - GET → curl <http://localhost:8001>
 - POST → curl -X POST <http://localhost:8001>
 - PUT → curl -X PUT <http://localhost:8001>
 - DELETE → curl -X DELETE <http://localhost:8001>
 - **In PowerShell** use **curl -Method Get** **or** **curl -Method Post** **or** **curl -Method Put** **or** **curl -Method Delete**

Routing

- **Routing** - Mapping an 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 run type **node routing.js**
 - In the browser type
 - » <http://localhost:8000/class/summer>
 - » <http://localhost:8000/syllabus>

Dynamic Generation of HTML

- View/templating engines - Allows you to generate dynamic HTML
- EJS (Embedded JavaScript) engine is a 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 type **node dynamicHTML.js**
 - In the browser type <http://localhost:7001/>

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 run type `node queryArguments.js`
 - In the browser type <http://localhost:7002/?semester=fall>
 - » Or use **`formGet.html`**

Retrieving values associated with POST

- The body-parser module allows you to retrieve parameters submitted using post by using `request.body.<PARAMETER_NAME>`
- To install
 - **npm install body-parser**
- **Example:** `formPost.html`, `postParameters.js`, `templates/courseInfo.js`
 - To run type **node postParameters.js**
 - Open **formPost.html** in the browser and provide some data

Retrieving Form Data

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

References

- **Express in Action**

Writing, building, and testing Nodes.js applications

Evan M. Hahn

April 2016 , ISBN 9781617292422