11
REST and Express

Dependencies

- A dependency is any additional software that your code “depends upon” in order to get its work done.

- Example: NodeJS is a dependency when we write a Node app.

- If you want to use a database to store your data you would likely use something like MySQL or MongoDB. These would then be dependencies.

- In NodeJS we can list our dependencies in the package.json file and let NPM handle setting up and configuring our dependencies for us.

What is an API?

- Application Programming Interface
- APIs exist to make programming applications easier
- Instead of a user interface with buttons, switches, knobs, etc we have a programming interface with methods that we can call, parameters to pass to those methods, and returned responses with which to make decisions
- APIs exist for all types of applications - desktop, mobile, web. Operating systems expose API's for writing low level application code - Win32, Cocoa, Linux Standard Base, POSIX, etc.
- We can build our own web APIs in NodeJS and Express to provide client code (running in the browser) a mechanism to interact with our application.

Representational State Transfer (REST)

- Roy Fielding wrote his doctoral dissertation about the underlying principles.
- We can focus on the basic concepts - the so-called RESTful aspects.
- For our discussion REST is an architectural style for building stateless web APIs.
- The web is already stateless by design so instead of taking the traditional approach and trying to build a stateful machine on top of the stateless web, REST just goes with it and embraces the statelessness of the web.

Representational State Transfer (REST)

- Central to REST is the concept of a resource
- REST attempts to separate the resource (an abstract thing) from its representation (a concrete, tangible thing)
- A resource can have any number of different representations - think of a music piece can be represented on various physical mediums - tapes, CD's, MP3, etc but the piece is still the same musical work - the same resource.
- We build REST API's around the HTTP verbs GET, POST, PATCH, DELETE.

Express

- Express.js is a Node.js library that helps us build web server applications.
- Provides a routing + sugar layer on top of Node.js http server.
- Uses the Model View Controller design pattern
- Declarative routing

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

Model View Controller (MVC)

- Model - Define domain objects, load data from data stores, write data to data stores
- View - Format display of the data to the user interface
- Controller - Route requests, load data into views

Routing

- The idea of routing is to map http requests for specific resources to some function that handles the request and returns a response. Similar in concept to how we map events to their event handlers.

- The concept of routing is not specific to Node/Express, it is a common denominator across all web application frameworks from Rails, to Django, Java/Spring, PHP, ASP.NET etc. If it's a web app, you will need routing!

- The two components of a route are the HTTP verb (GET, POST, PATCH, DELETE) and the URI of the requested resource.

Routing examples using Express

```
app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.get('/about', function (req, res) {
  res.render('about.html')
})

app.post('/users/new', function (req, res) {
  res.render('user.html')
})
```

MIME Types

- Multipurpose Internet Mail Extensions
- Originally designed for email servers - SMTP
- Used to describe the format of data so the client knows which application should be used to read/view the data.
- Content types are placed in the header of messages - Ex: text/plain, text/html, text/css, application/json, multipart/form-data, audio/mpeg
- Most of the time we don't have to worry about MIME types if we are using a framework like Express since it handles them for us. But it is important to be aware of the types of responses we send to the client.
- MIME gives context to data being sent over HTTP

Additional Resources

- What is an API? - https://www.programmableweb.com/api-university/what-are-apis-and-how-do-they-work
- REST https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- MIME types - https://en.wikipedia.org/wiki/MIME
- Examples of using REST APIs and feeding the output of one API into the input of another API creating a mashup - https://www.youtube.com/watch?v=7YcW25PHnAA
- Express - https://expressjs.com/en/starter/hello-world.html
- Express sources - https://github.com/expressjs/express

Time to build an app using Express!

- Express comes with generators - scripts that set up a scaffolding of code
- We will build things from scratch without using generators
- $ npm init - creates a package.json file that describes our application.
- $ npm install express --save - installs express as a dependency
- npm install nodemon -g - installs a utility called nodemon that automagically restarts our node server whenever it detects any changes to the file structure - saves a whole lot of dev time

View Templating

- Engines - EJS, Handlebars, Jade
- EJS - Embedded JavaScript - html with some extra hooks to embed our data
- Use <% and %> when the value will not be retrieved from a variable
- Use <%= and %> when the value will be retrieved from a variable
- Handlebars (or mustache) uses the {{ }} style of syntax
- Jade uses a Haml-like syntax where the code is indented and no <>, </>

Client side code

- Html forms
- <form> tag encloses the whole form
- <label> tags place a text label before an input element
- <input type=”text”> tags define input elements (text boxes)
- <input type=”submit”> tags define a submit button - clicking on this button is how the user triggers the post action to the server
- We can also use a little bit of bootstrap CSS classes to freshen up the appearance of our pages so they don’t look as boring.