Routes, Templates, AJAX

Preparation

- npm install
- MySQL in myapp database:
  - "CREATE TABLE items (id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY, item VARCHAR(30) NOT NULL)"
  - "INSERT INTO items (item) VALUES (car')"

Template rendering

1) Server-side
    - Controller/route retrieves data
    - Render view with template
    - View (with data inserted) is sent to browser

2) Client-side
    - AJAX
    - Controller/route retrieves and returns data as JSON
    - Client-side JavaScript receives data, updates HTML

myapp/routes/index.js

- Server-side template rendering
- return res.render('index', {'title': "my list of items"});

EJS

- Templating library: combines data and a template to produce HTML.
- We will use EJS for both server-side and client-side template rendering
- http://www.embeddedjs.com/

- Other templating libraries: http://www.creativebloq.com/web-design/templating-engines-9134396

Bootstrap

- CSS, JavaScript style library
- For fast styling, scaffolding: getbootstrap.
com

AJAX

- AJAX = Asynchronous JavaScript and XML
- Not a language; a way of programming
- Send requests to server / receive data from server, update HTML, without reloading entire page
- You can implement AJAX with JavaScript or jQuery

GET

- **myapp/public/js/index.js**
  - AJAX request to server
- **myapp/routes/items.js**
  - route matches URL
  - call controller method
- **myapp/controllers/ItemsController.js**
  - getItems sends a SQL query to the database
  - on successful query, return JSON object with data
- **myapp/public/js/index.js**
  - success and failure callbacks
  - modify HTML using client-side templates
- **myapp/public/partials/index.ejs**
  - client-side template

POST

- **myapp/public/js/index.js**
  - AJAX request with data to server
  - req.body: http://expressjs.com/api.html#req.body
- **myapp/routes/items.js**
  - route matches URL
  - call controller method
- **myapp/controllers/ItemsController.js**
  - addItem sends a SQL query to the database
  - on successful query, return JSON object with data
- **myapp/public/js/index.js**
  - success and failure callbacks
  - modify HTML using client-side templates

DELETE

- **myapp/public/js/index.js**
    - AJAX request to server
- **myapp/routes/items.js**
    - route matches URL
    - URL contains a “route parameter” (http://expressjs.com/api.html#req.params)
    - call controller method
- **myapp/controllers/ItemsController.js**
    - removeItem sends a SQL query to the database
    - on successful query, return JSON object with data
- **myapp/public/js/index.js**
    - success and failure callbacks
    - modify HTML using client-side templates

So far...

- We have:
  - MVC on server side
  - HTML templates updated by AJAX
- Separate steps:
  - Sending/receiving data to server, receiving data from server
  - Updating view

MVC for frontend JavaScript

- http://www.infoq.com/research/top-javascript-mvc-frameworks
  - Backbone
  - Ember
  - Angular