6.148

Backend III: Node.js with Databases

HELLO AND WELCOME!

Your Feels

- Lecture too fast!

Your Feels

- Lecture too fast!
- Too many languages

Your Feels

- Lecture too fast!
- Too many languages
- Code more in class

Your Feels

- Lecture too fast!
- Too many languages
- Code more in class
- Stop coding so much in class

Your Feels

- Lecture too fast!
- Too many languages
- Code more in class
- Stop coding so much in class
- How do I put it all together?

Your Feels

- Lecture too fast!
- Too many languages
- Code more in class
- Stop coding so much in class
- How do I put it all together?
- More workshops / interactive help!

We’re here to help

office hours
hackathon
piazza
starter kits
annotated sample code

A (very) Brief Review

- Ideas!
- Design
  - Usability
  - Graphic design

A (very) Brief Review

client (you)

HTTP request: GET xkcd.com

HTTP response: web content
(HTML, CSS, JavaScript)

server (xkcd)

A (very) Brief Review

- Client technologies run on the browser
  - HTML, CSS, (client) JS
  - In Node.js, these are your *views* and your *public directory*

A (very) Brief Review

- Server technologies run on the server
  - Node/Express (server JS), Ruby on Rails
  - Model-view-controller/router
  - Handle a request, send back HTML, etc.

A (very) Brief Review

- Databases store data persistently
- We'll use MongoDB with Node
  - Don't worry about installation for now; instructions to come

Where we left off

- MongoDB is a NoSQL database
- Convenient -- reads/writes JS objects
- No structure to data: collections can contain (basically) anything
- No built-in way to express relations between objects

Our Photos App

- User inputs photo URL and caption; page that shows photos
- Last week: stored URLs in a “fakedb” object
- Today: store URLs in a real database

Enter Mongoose

- Library/wrapper *on top of* MongoDB
- Lots of nice things:
    - Schemas
    - Validators
    - Methods
    - Relations

package.json

```
"dependencies": {
  "express": "~4.8.6",
  "body-parser": "~1.6.6",
  "cookie-parser": "~1.3.2",
  "morgan": "~1.2.3",
  "serve-favicon": "~2.0.1",
  "debug": "~1.0.4",
  "ejs": "~0.8.5",
  "mongodb": "*",
  "mongoose": "*"
}
```

Schema

- Structure your data! Can't store anything that doesn't conform to schema

```
var photoSchema = mongoose.Schema({
  caption: String,
  url: String
});
```

Schema

- Structure your data! Can't store anything that doesn't conform to schema

```
var photoSchema = mongoose.Schema({
  caption: String,
  url: String
});
```

Schema

- Structure your data! Can't store anything that doesn't conform to schema

```
var photoSchema = mongoose.Schema({
  caption: String,
  url: String
});
```

Models

- Think of a schema as a declaration for what your data will look like
- Models are the actual classes that are created

in Javascript

Models

```
var photoSchema = mongoose.Schema({
  caption: String,
  url: String
});
```

```
var Photo = mongoose.model('Photo',
photoSchema);
```

Validators

- Perform additional checks on the data before storing

```
var checkLength = function(s) {
  return s.length > 0;
};

Photo.schema.path('caption').validate(
  checkLength, "Caption cannot be empty");

Photo.schema.path('url').validate(
  checkLength, "URL cannot be empty");
```

Making a DB Query

```
models.Photo.findOne(
  {_id: photoId},
  function(err, result) {
      console.log(result);
      res.render('photo',
          { photo: result });
});
```

Making a DB Query

```
models.Photo.findOne(
  {_id: photoId},
  function(err, result) {
      console.log(result);
      res.render('photo',
          { photo: result });
});
```

an import from a “models” file

Making a DB Query

```
models.Photo.findOne(
  {_id: photoId},
  function(err, result) {
      console.log(result);
      res.render('photo',
          { photo: result });
});
```

a MongoDB query

a query selector

Making a DB Query

```
models.Photo.findOne(
  {_id: photoId},
  function(err, result) {
      console.log(result);
      res.render('photo',
          { photo: result });
});
```

a MongoDB query

a query callback

Why Callback?

- DB queries are expensive!
- If do nothing and wait, our server will be very slow!

- Send off DB query, do other stuff, when DB query returns then execute the callback

MongoDB Callbacks

- Always of the form:
  function(error, result) {...}

- Result object generally intuitive (what you were looking for, what you just saved, etc.)

MongoDB Callbacks

- Always of the form:
  function(error, result) {...}
- Result object generally intuitive (what you were looking for, what you just saved, etc.)
- Remember Neander Lin? Check your errors!

Creating New Objects

```
var newPhoto = new models.Photo({
    caption: req.body['submitted-url'],
    url: req.body['caption']
  });
```

a Mongoose model, imported from models file

a simple JS object; note that it satisfies the schema

Not Done Yet...

```
var newPhoto = new models.Photo({
    caption: req.body['submitted-url'],
    url: req.body['caption']
});

// at this point newPhoto is only in memory

newPhoto.save(function(err, result) {
    res.redirect('/photos/' + result._id);
});
```

callback looks the same as the find(...) call

result is what we just saved

Mongoose does a lot more

- Helper functions/instance functions
- References to other objects
- Not going to demo right now, but will be in annotated code
- DEMO TIME