Building your own Data Service:

A Whirlwind Tour using Postgres and Javascript

Goal:
Build a program that returns data that meet your filters/searches on demand.

Motivation:
- Reduce dataset size
- Do filtering on a (powerful) server, instead of a client (e.g., phone)
- Tailor search results to different clients so different people can get different results
- Track usage statistics

Example:
National Transportation Safety Board ***Airline Delay*** Dataset

Fields:
- Airline and Flight Number
- Date
- Arrival and Departure Airports
- Delay (in minutes)
- Delay attribution

What if you wanted to map...

...what airports have the biggest delays?
..is weather directly correlated to flight delays?
...what states have the worst performance?
...flight route frequency?

January 2016
>500,000 records!

Components of a Data Service

Database | *Bridge* | *Client*

Database:
- Hold onto the data
- Enable querying and data insertion
- Indexing, optimization, etc
- User management & authentication

Bridge:
- Process user-specified parameters
- Execute queries
- Get/put data with code
- Control what clients can do (i.e., they can't delete everything)
- aka API, application backend

Node.js

Client:
- Faciliate interaction with user
- Get query parameters from UI
- Return results to UI (and map...)

JavaScript

Database Steps:
1. Create Database
2. Outline and implement user roles and privileges
3. (optional) Enable PostGIS
4. Define schema and table structure
5. Upload data (may require a script)

Node-what?
Node.js is JavaScript for the backend. It's not totally the same as front-end JS programming, but if you know the front-end, it can be a more continuous process than programming in other backend languages, like PHP, ASP, .net, etc...

Use a module
Node is modular, so you can include libraries that other people write. The Express.js framework is a module designed for web request processing. Use it.

Bridge Steps:
1. Install node on your computer or server
2. Create a package.json file that lists module dependencies.
3. Install dependencies
4. Create a file called app.js
5. Set up express request routing
6. Write your API endpoints

Sending and Receiving Data

Sending
1. Collect query parameters from UI controls (e.g., text inputs, radio buttons, map)
2. Write an AJAX function
3. Send the query parameters in the data object of the AJAX call

Receiving
1. Check for success
2. (optional) Parse JSON
3. Do stuff with your data

Endpoints
An endpoint should be like a function (in fact, it is,when you use node/express), in that it serves a single, repeatable process that can be slightly changed by supplying it with parameters or arguments.

For example:

`https://api.mywebsite.com/flights?`
Should serve a list of **flights**

`https://api.mywebsite.com/airports?`
Should serve a list of **airports**

Adding Arguments
Arguments are given to an endpoint using key=value pairs. They are (usually) case senstive. You can add as many as you want by separating paris with an '&'.

For example:

`https://api.mywebsite.com/flights?origin=PHX`
Should serve a list of the **flights** from Phoenix

`https://api.mywebsite.com/airports?state=CA&airline=AA`
Should serve a list of the **airports** in California that serve American

Tips, Tricks, & Suggestions
1. Think about what operation(s) you're trying to support:
   - what data do you need?
   - what fields are required?
   - what parameters/arguments do need?
2. Make as many endpoints as you feel you need -- don't cram.
3. Go spatial: make full use of spatial queries where appropriate. You can even customize your backend to return your choice of data format (e.g., GeoJSON or vanilla JSON).
4. Use modules. Node has modules for everything, from password hashing to array manipulation.
5. Return JSON from your backend. It's easy to produce and easy to consume. Flat files (CSV) are difficult to produce and harder to consume in code.
6. Play around! Explore!

Tutorial:
1. Exploration database (must be on SH workstation):
   Username: classtutorialuser
   Password: Tutorial!
   Database: class-tutorial
2. Annotated Scripts