

# WEB SERVICES

Practical Definition:

- Web Service - web call that returns data for programmatic use
- Endpoint - the URL for that call

# **SERVICES AREN'T PAGES**

The difference is purely in how it is used.

Not a code difference.

Pages return HTML intended for the browser.

Other endpoints return CSS, JS, images, media.

Service endpoints can return HTML fragments, text, JSON, XML, YAML, etc

Services used by frontend and/or backend

# SERVICE CONVENTIONS

- What do you send?
  - Url
  - Parameters
  - Headers
- How do you send it?
  - HTTP Method
  - Body/URL Parameters
- What do you get back?
  - Body
  - Status
  - Headers

# ONE WAY: ANYTHING GOES

- Have an endpoint
- Send params to define a command and details
- Return a 200 status code
- Return a message body that contains results or error message

This is common...but not very good

Why?

# **ANOTHER WAY: SOAP**

Anything goes, but with a lot of XML

- benefits of XML (schemas)
- drawbacks of XML
- control at data assembly/parsing, not HTTP layer

# **ANOTHER WAY: GRAPHQL**

- Send a "query" using a query language
- describe what data you want
- describe what format (data structure) you want

# OUR WAY: REST

Nuanced system, often done partially

Three Basic rules:

- URL represents a "resource" (to interact with)
- HTTP Methods are the interaction
  - GET: Read
  - POST: Create
  - PUT: Replace
  - DELETE: Delete
  - PATCH: Update
- HTTP Status code is result of resource interaction

# MODERN DAY

- SOAP - rare
- GraphQL - new and rapidly growing
- REST - most common
- JSON most common data format

For this clase:

- use REST services
- use cookies for service authentication
- send services data as JSON in the body
- services return JSON data as the body