A bit of history

- HTTP docs
  - Easy for humans, not so much for machines
- XML
  - Easy for both
  - Structures data with built-in types and tight, enforceable rules
- SOAP
  - Simple Object Access Protocol
  - Standardized the communication between servers
- XML-RPC
  - “SOAP light”
- REST
- https://thehistoryoftheweb.com/soap-rest-odds/

A bit of victory

SOAP and XML-RPC

- Set of technologies
- Lot of rules, pretty rigid
- Verb oriented
  - Think in actions
- Won in Enterprise space

REST

- Collection of design principles,
  taking advantage of built-in HTTP methods
- Noun oriented
  - Think in resources
  - Limited set of actions
- Won in Open API space

REST architecture

- Uniform interface (HTTP verbs)
- Client–server
- Stateless (scalable)
- Cacheable
- Layered system
- Code on demand (optional)

- https://restfulapi.net/rest-architectural-constraints/

Taking it further

- HATEOAS
  - Hypermedia As The Engine Of Application State
  - https://restfulapi.net/hateoas/
  - Resource contains the links (full URIs) that allow you to “move” from application state to application state
    - Includes navigational links! (including paging)
  - If every REST API server followed HATEOAS, we would be able to have a single client to consume all of them

Mindset shift

- From verbs to nouns
- REST resource and API design
  - Perhaps more akin to data modeling
- Pitfall: RPC style resources
  - One resource URI with a body that varies with one property, usually “action” or “type” that dictate
- Solution: when you think of an action find the nouns it is acting on

HTTP verbs

- GET
  - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
- POST
  - The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT
  - The PUT method replaces all current representations of the target resource with the request payload.
- PATCH
  - The PATCH method is used to apply partial modifications to a resource.
- DELETE
  - The DELETE method deletes the specified resource.
- HEAD
  - The HEAD method asks for a response identical to that of a GET request, but without the response body.
- CONNECT
  - The CONNECT method establishes a tunnel to the server identified by the target resource.
- OPTIONS
  - The OPTIONS method is used to describe the communication options for the target resource.
- TRACE
  - The TRACE method performs a message loop-back test along the path to the target resource.

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Creating: POST vs PUT

- POST
  - Creating a new item in a collection
  - Server decides identifier
  - NOT idempotent
- PUT
  - Creating or updating an identified item
  - Client decides identifier
  - Idempotent

- https://restfulapi.net/rest-put-vs-post/

Updating: PUT vs PATCH

- PUT
  - Always works on full resource
- PATCH
  - Allows partial updates on a resource

- Many REST servers allow partial updates in PUT
  - Server needs to use nullable types when deserializing the resource it receives

Resources

- REST API Tutorials
  - https://restfulapi.net
  - https://www.restapitutorial.com
- When to use POST or PUT to create a resource
- Be as specific as possible with status codes
  - https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
  - https://www.restapitutorial.com/httpstatuscodes.html
- “RESTful Web Services” book
- How to Design a Good API and Why it Matters – Research
- REST API Documentation best practices
  - Examples question on stackoverflow