REST API Introduction

Topics

1) How to request and send data to a server?
2) How to design a server’s API?

HTTP

Overview

- Front-end = client-side; browser
- Back-end = server side
- Why make web-based app?
  - server to allow interaction between users
  - server to store resources or do heavy processing
  - centrally managed deployment and admin

Server Interaction

- Browser getting data from webserver
  - browser does HTTP GET on URL
  - server sends back a web page (HTML, CSS, JS)
- Font-end/Back-end Interaction
  - client-side makes requests to server's RESTful API's endpoints (URLS) endpoint mapped with an HTTP method
  - data transmitted in JSON (or XML)

HTTP

- HTTP:.. hypertext transfer protocol
- URL:.. uniform resource locator
  - Ex: http://www.sfu.ca/~bfraser/answers

    <protocol>://<domain name>/<path>

    <protocol>://<domain name>:<port>/<path>
- Protocol ports
  - HTTP: 80 (or 8080 alt)
  - HTTPS: 443 (or 8443 alt)
    S = Secure

HTTP Methods

- HTTP methods:
  What does the client want to happen at a URL?
- These are the: "actions" that HTTP supports
  - GET : retrieve some information from the URL: does not change server state
  - POST : Submit a new entity (object) to the URL
  - DELETE : Delete some entity (object) at the URL
  - PUT : Replace an entity at the URL with new value
  - ... omitting HEAD, CONNECT, OPTIONS, TRACE, PATCH

HTTP Response Status Codes

- Each request message (a GET, POST, ...) returns a response code:
  - 200:.. OK
  - 201:.. created (usually from POST)
  - 400: Bad Request (client-side error)
  - 401: Unauthorized (who are you?)
  - 403: Forbidden (I know who you are, but still not allowed)
  - 404:.. file not found
  - 500: Server-side error
  - (... many omitted!)

Sending Data to the Server

- Front end can send data to the server via:
  - URL : Put data in path variables
    - Ex: GET http://my.com/api/person/5
  - query string : for GET only;
    no raw special characters (Ex: %20 = space)
    - Ex: https://www.google.com/search?q=hi+world
  - header : All HTTP messages have header
    - Ex: authentication or apiKey "ApiKey:abc123"
  - body : Block of data (often text such as JSON)
    - Ex: {"name":"Dr. Evil","age":95,"laugh":"Mwahah"}

URL Path Variables Details

- Path Variable Idea
  - URL encodes groups or categories as though they are “folders”, and items as “files”
- Example
  https://coursys.sfu.ca/2050sp-cmpt-276-d1/students/hiwld
  - It seems like we are browsing into folders for a specific file
  - server extracts “folder” and “file names to programmatically find the data ..

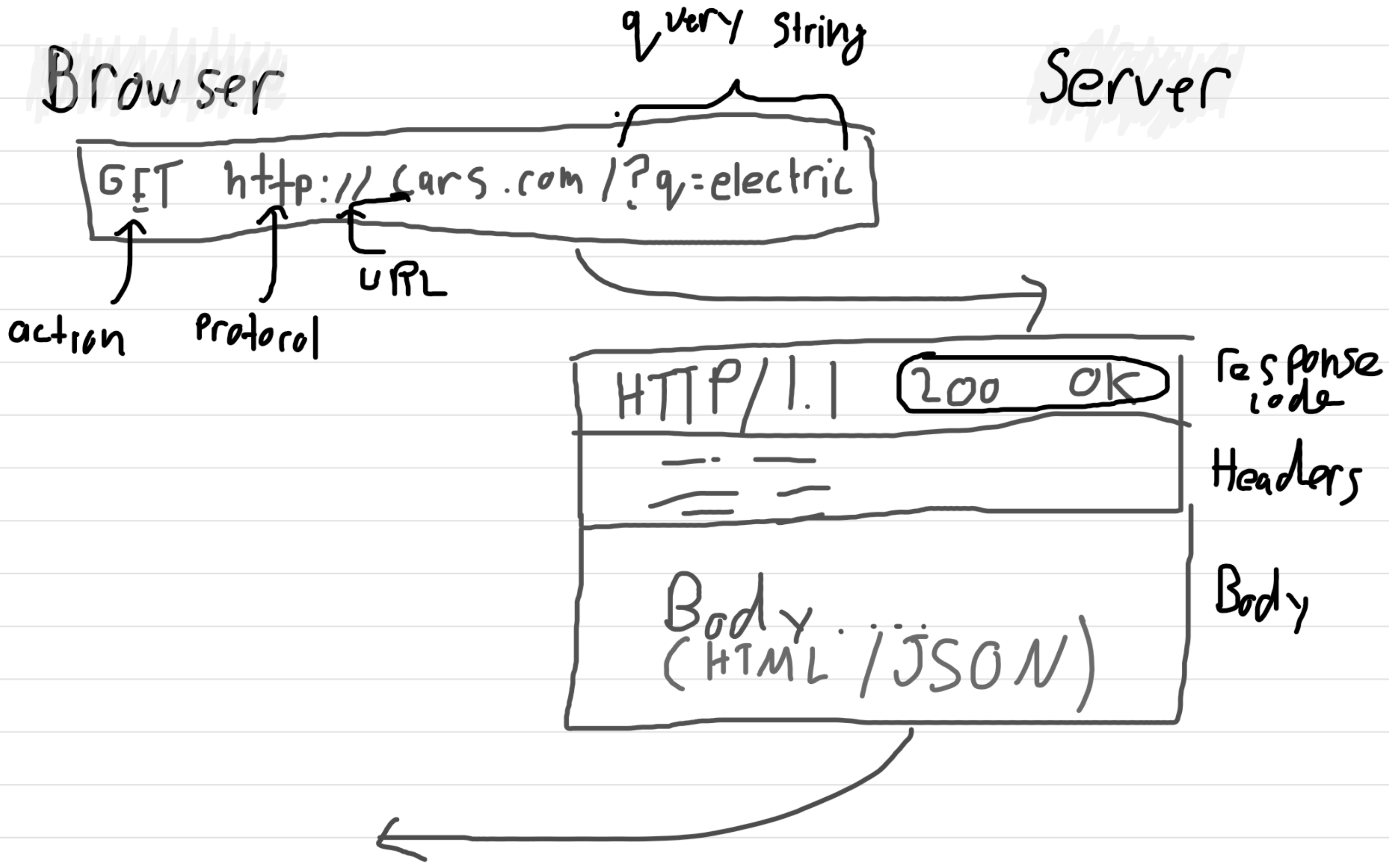
Query String Details

- Query String: the common way to send data for GET
  - Use to encode.. non-hierarchical parameters
    - Ex: search queries
- Common Format
  http://my.com/s?key=value&otherkey=othervalue
- Demo

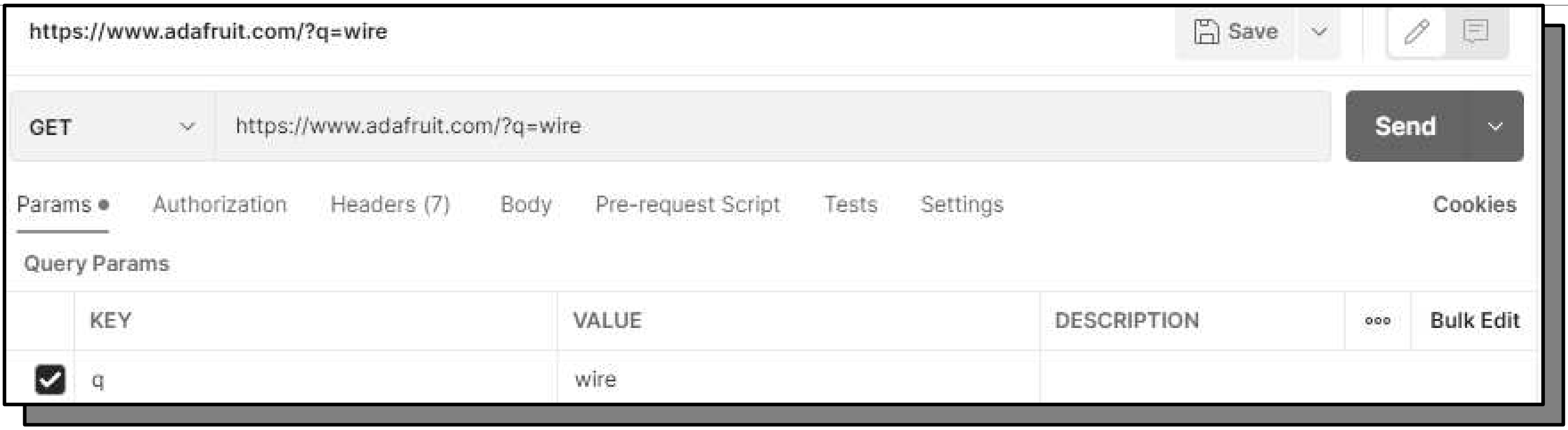
```
curl -k -i  -X GET https://www.adafruit.com/?q=wire
```

curl is utility to execute HTTP request

Request to Server & Reply



Postman Request & Response

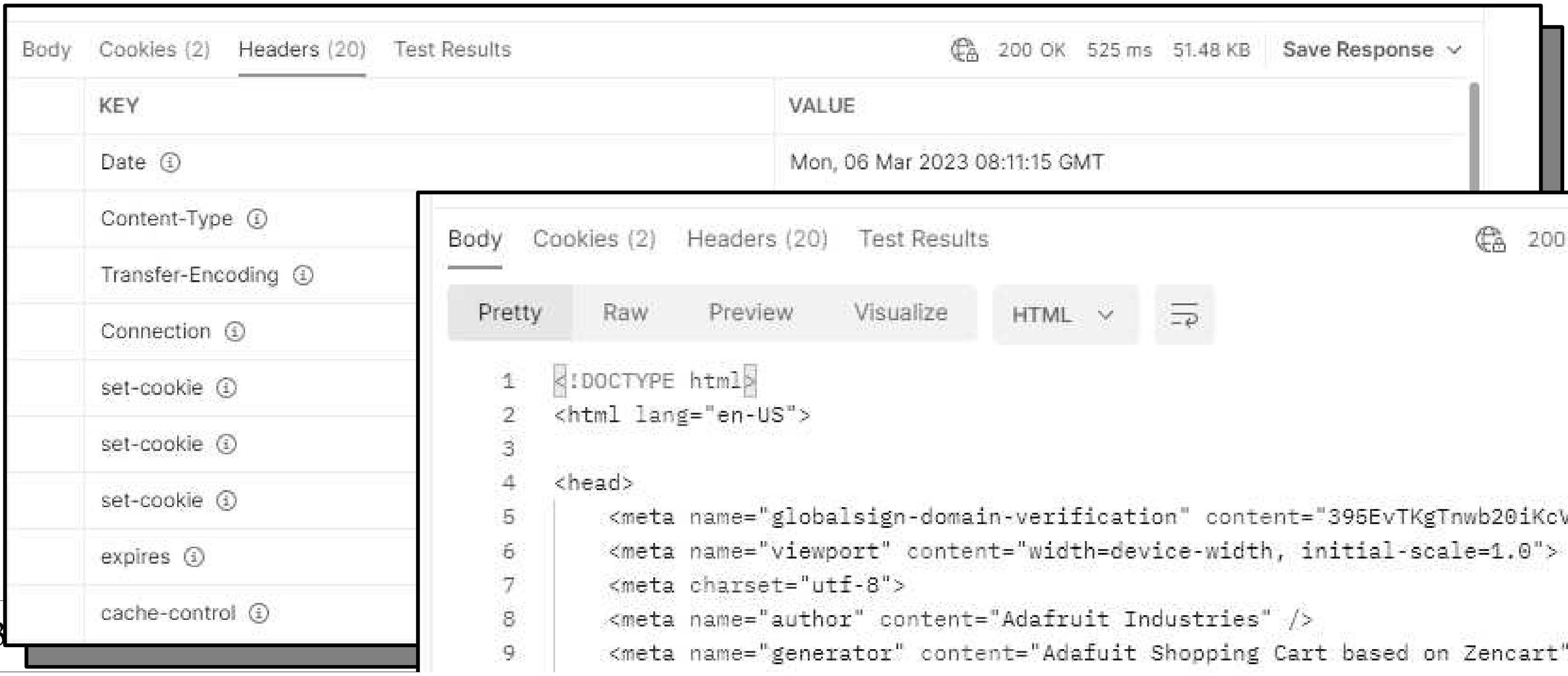


https://www.adafruit.com/?q=wire

GET https://www.adafruit.com/?q=wire

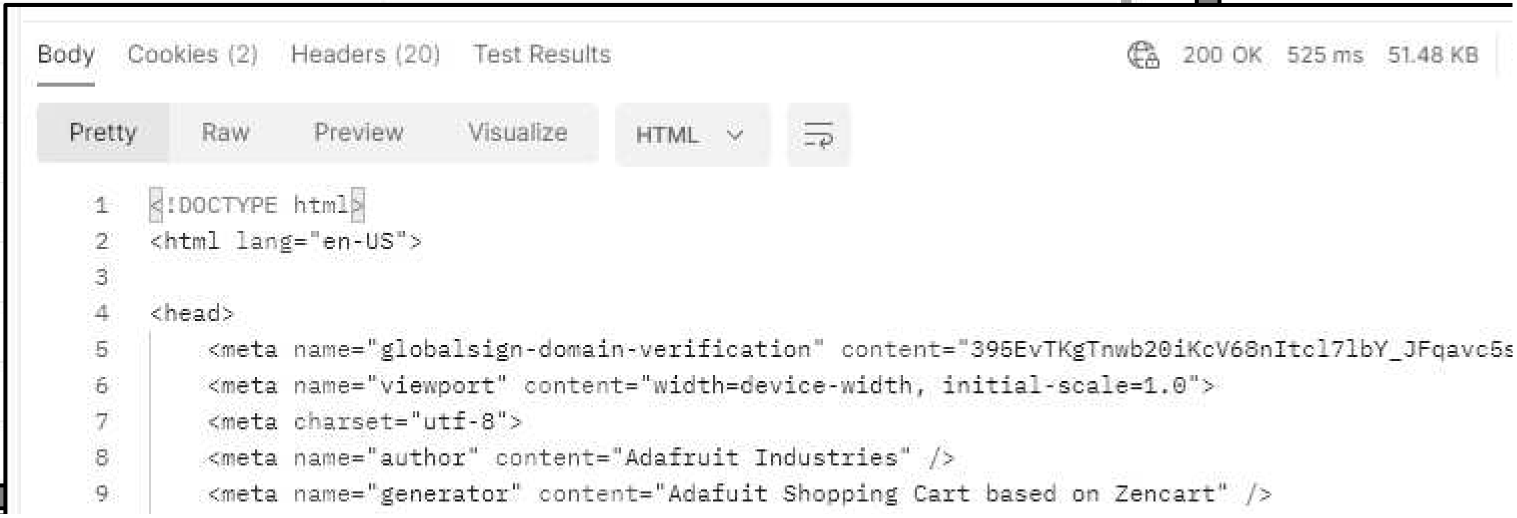
Send
Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Cookies
Query Params

	KEY	VALUE	DESCRIPTION
✓	q	wire	



Body Cookies (2) Headers (20) Test Results
200 OK 525 ms 51.48 KB Save Response

KEY	VALUE
Date	Mon, 06 Mar 2023 08:11:15 GMT
Content-Type	
Transfer-Encoding	
Connection	
set-cookie	
set-cookie	
set-cookie	
expires	
cache-control	



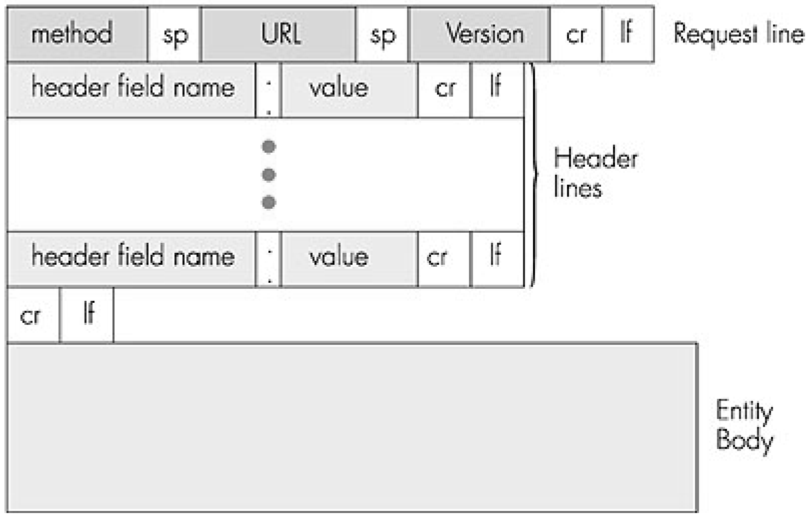
Body Cookies (2) Headers (20) Test Results
200 OK 525 ms 51.48 KB
Pretty Raw Preview Visualize HTML

```
<!DOCTYPE html>
<html lang="en-US">

<head>
    <meta name="globalsign-domain-verification" content="395EvTKgTnwb20iKcV68nItcl7lbY_JFqavc5s
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
    <meta name="author" content="Adafruit Industries" />
    <meta name="generator" content="Adafuit Shopping Cart based on Zencart" />
```

HTTP Body details

- HTTP messages can include a body
  - Used by POST and PUT to send data
  - Often a JSON structure or binary data



HTTP Request

```
GET   /~bfraser/    HTTP/1.1
Host: www.sfu.ca
Connection: keep-alive
Cache-Control: no-cache
User-Agent: Mozilla/5.0 ...
Accept: text/html,application/...
```

HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 02 Mar 2020 05:10:18 GMT
Server: Apache
box: b3 D=1361386 t=1583125818662494
Access-Control-Allow-Origin: *
Content-Length: 3795
Content-Type: text/html;charset=ISO-8859-1

<!DOCTYPE
<html>
 <head>
  <title>Index of /~bfraser</title>...
```

Image: http://netlab.ulusofona.pt/rc/book/2-application/2_02/index.htm

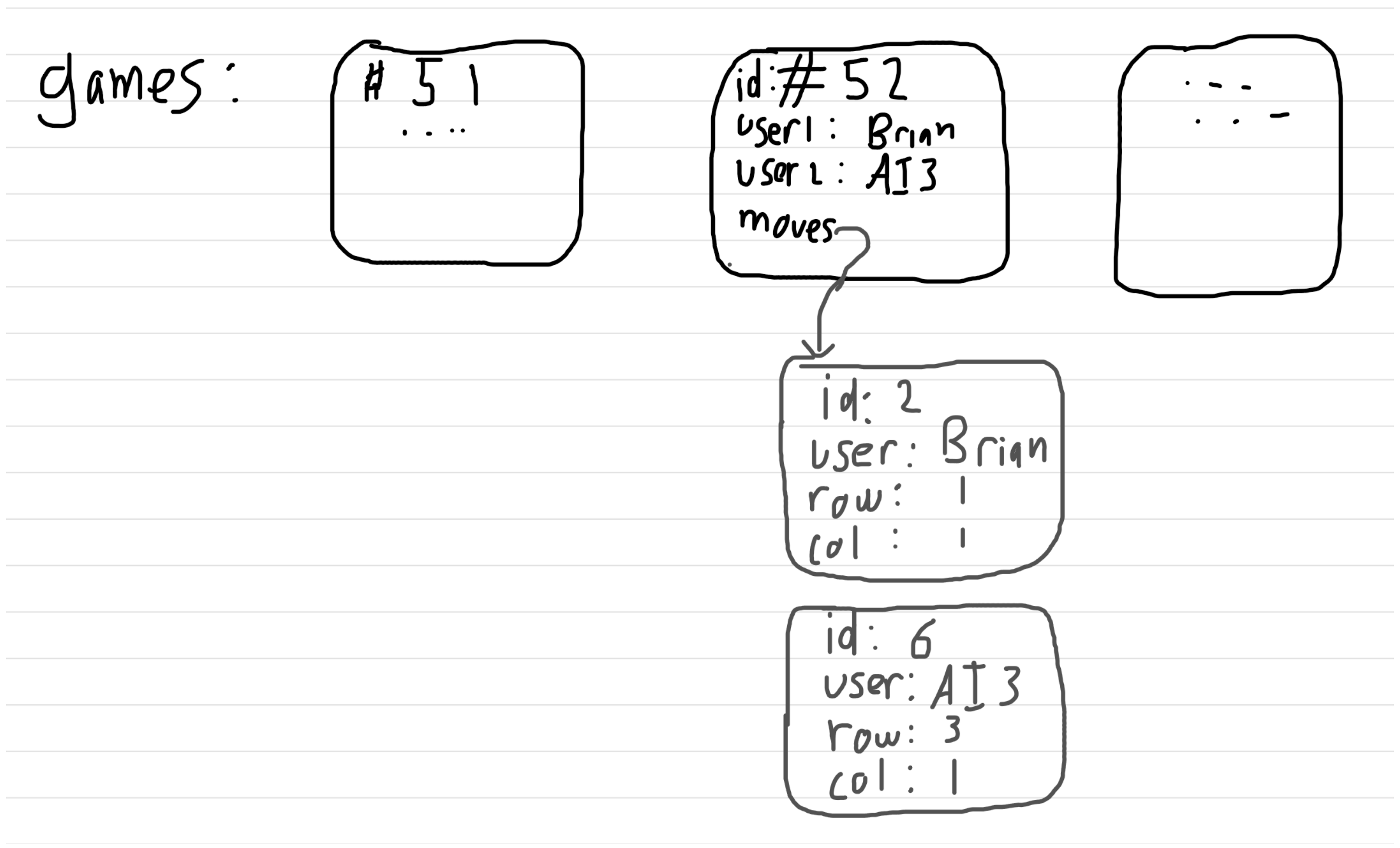
REST API

API & REST

- API:.. application program interface
  - How a program exposes its functionality for other programs to use.
- REST:... representational state transfer
  - .. architectural style using HTTP methods and URL to interact with a server
  - It works with HTTP caching and semantics to improve performance
  - REST is founded on some principles, not a strict prescription.
    So what is "RESTful" is up to interpretation
- TLA: Three Letter Acronym

note: rest is not a protocol but rather an idea to design protocols
REST: +exposing data one way, send data the other way with methods: PUT,…
+not set of functions (not change state of data)
+ex: different API style (remote execution API) will change the data (update student state to be graduate)
—> REST provides more flexibility

Example Tic Tac Toe Model



REST Example

- Example: Tic-tac-toe game
  - Base URL: my.com

folder supports methods

– /games	GET (list), POST (new)
– /games/52	GET (info), POST (change info)
– /games/52/moves	GET (list), POST (new)
– /games/52/moves/1	GET (info), POST (change info)

- Full Example
  GET my.com/games/52/moves/1
  - In games API, retrieve info on game #52’s move #1

REST Example (cont)

- Get Game Info

```
curl -X GET localhost/games/52

HTTP/1.1 200 OK
{
    "id":       52
    "user1":  "Brian",
    "user2":  "AI3",
    "href":    "/games/52"
}
```

- Data Structure

```
struct {
    int id;
    string user1;
    string user2;
    string href;
}
```

data transfer object (DTO)
:
Simple data structure to send data from back-end to front-end

REST Example (cont)

- Get Moves

```
curl -X GET localhost/games/52/moves

HTTP/1.1 200 OK
[
        {
                "id":       2,
                "user":     "Brian",
                "row":      1,
                "col":      1
        },
        {
                "id":       6,
                "user":     "AI3",
                "row":      3,
                "col":      1
        }
]
```

- Make a move

```
curl -X POST -d {
        "user":     "Brian",
        "row":      3,
        "col":      3
} localhost/games/52/moves
```

RESTful API Design

- Design API around things and actions
  - Structure URL for the hierarchical nature of the data
- Things (nouns)
  - Data you want to expose
  - make path names plural, no trailing slash
- Actions (verbs)
  - Create POST (or PUT)
  - Retrieve GET
  - Update POST (or PUT if you are updating the whole item at once, not just part).
  - Delete DELETE

RESTful API Design (cont)

- GET (and PUT) must be idempotent:
  - .. doing it twice does not have an extra effect
- POST is a catch all for doing anything.
- Properties of RESTful
  - uniform interface: Server returns self-descriptive resources
  - stateless: Server maintains nothing about state of the connection; everything comes from HTTP headers, etc why stateless is useful?
  - cacheable: Cache as much as possible to reduce server load
  - <...omitted more...>

Summary

- HTTP
  - Protocol for accessing resources via URL’s
- HTTP Methods
  - GET, POST, DELETE, PUT, etc.
- Data in URL, Query String, Header, Body
- REST
  - Design URLs for Hierarchical data
  - REST properties