HTTP

Hyper Text Transfer Protocol
A set of rules that establish standards for how data is transmitted between a client and a server.
Clients and Servers use this protocol over the internet (on the web) to communicate with each other.

- HTTP supports one request per connection.
- Clients connect to server, send one request, and then disconnect.

HTTP Request (sent by the client)

- just a packet of information
- Includes:
    - HTTP Version
    - URL
    - HTTP Method
    - Request Headers
    - Request Body

HTTP Response (sent by the server back to client)

- Includes:
    - HTTP Version
    - HTTP Status Code
    - Response Headers
    - Response Body

HTTP Methods: tell the server what type of request is being sent, how the request is being made, and how the rest of the information in the request will be formatted.

- Most popular echo our CRUD methods
    - POST -> create
    - GET -> read
    - PUT -> update
    - DELETE -> delete
- others: PATCH, TRACE, OPTIONS, HEAD

HTTP Status Codes: tell the client (us) how the request was handled.

- Attached to the Response - so that the client knows how everything went.
- Grouped in increments of 100s (100 - 500)

100s -> Informational

200s -> Success

- 200 OK the request has succeed
- 201 Created success, and a resource was created
-

300s -> Redirects

- 301 Resource Moved Permanently
- 307 Temporary Redirect

400s -> Client-Side Errors

- 400 Bad Request (syntax - server doesn't get it)
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 418 I'm a teapot

500s -> Server-Side Errors

- 500 Internal Server Error
- 502 Bad Gateway
- 503 Service Unavailable

URL: Uniform Resource Locator - location for where a resource (on the server) is located.

protocol + domain name (hostname:port) + URI + parameters

URI: Uniform Resource Identifier

- determines specifically which resource is needed from the server

Parameters

- there are 2 different kinds
- Path Parameters - `/path_parameters`
- Query Parameters - `/?param1=value&param2=value2`

Servlets

Client and Server Architecture

A networking model where the server provides services to clients in order to perform user-based tasks.

Server: software designed to process requests and deliver responses to another computer over the internet.

Client: program that runs on a local machine requesting service from a server

Client and Server might be on the same computer or two different computers connected by the internet

Servlets can have many definitions depending on the context

Servlet: a Java class designed to handle, process, and respond to incoming requests from a client (extends the capability of a server)

```
- we will be dealing specifically with HttpServlet
```

- can respond to any type of request
- a web component that is deployed on the server to create a dynamic web page.

Concepts

Website: a collection of **static** web pages (html)

Web Application: a website that has **dynamic** functionality on a server (i.e. google, facebook, twitter, etc.)

Web Server: a computer or machine that is designed to handle incoming HTTP requests

Servlet Container: contains one to many Servlets and is primarily responsible for mapping the Servlets to different addresses.

- It configures our servlets.

Tomcat: a server designed by Apache

- it will host our application
- this means our application will live on our tomcat server

web.xml:

- xml is used as a media type like json
- it is used to transport information
- web.xml is known as the **Deployment Descriptor**
    - this is what gives the servlet container the mapping and configuration details of our servlets
- It decides what requests are handled by which servlets.
    - Kind of like a manual
    - The Servlet Container uses this manual to interpret instructions and enforce them
    - If we want to change how our servlets work, we go through the deployment descriptor

Servlet Lifecycle

The Servlet Container manages the lifecycyle of servlets

1. When the application server starts (i.e. Tomcat) the servlet container deploys and loads all servlet classes.
2. The container creates one instance of each servlet class.
3. The `init()` method is used to intialize the servlet. Its called **once** for each servlet.
4. The Servlet Container calls the `service()` method **each** time a request for the servlet is received. The service method determines the type of HttpRequest and then calls another method: `doGet()`, `doPost()`, `doDelete()`, etc.
5. The `destroy()` method is called **once** at the end of a servlet's life, when the Servlet Container is ready to remove the instance of the servlet.

Library vs Framework

Library: a set of code that we add into our applications to bring additional functionality (we did not write this code ourselves) - Math Class, java.util,

Framework: IS an application, which works in its own way - we simply add our code to this application - then the application will run, and use our code as needed - JUnit - Servlet - Spring - Angular

In these frameworks - a LOT of the code is abstracted away and handled for us

JSON

JavaScript Object Notation

- text-based data storage format that is designed to be easy to read for bothe humans and machines.
- it's a way of formatting data so that it can be transported.

JSON is platform independent

Front Controller Design pattern

There is a single handler that routes incoming HTTP Requests.

- The Front Controller is a single entry point for all requests, and routes incoming user requests to the appropriate servlet.