Day 1

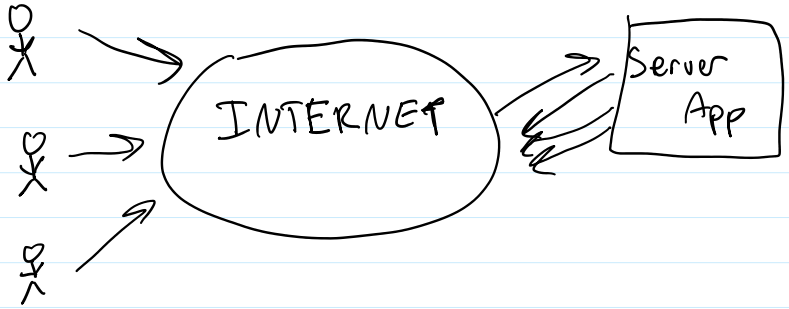
Monday, August 1, 2022 9:22 AM

Client Server Architecture

Networking model in which a server (your app) provides services to a front end / client

Server: a server software designed to process requests and deliver responses

Client: is some user interface a person interacts with
- CLI, website, postman



A client establishes a connection to the server through the internet
- Communicate with a set of rules called a protocol

We will be using HTTP as our protocol
- Client will send HTTP Requests
- Server will respond with an HTTP Response

Types of Client Server Arch.
- 2 tier: simple client to server interface
- 3 tier: client connects to a middleware
  - the middle connects to the server
- n-tier: multiple layers between the client and server

How will we achieve this? Servlets

Servlets are a java class that take incoming requests from a server and allow java to send a response back

Web/Application Server:

A web server is a server used to handle HTTP requests

A web server is a server used to handle HTTP requests

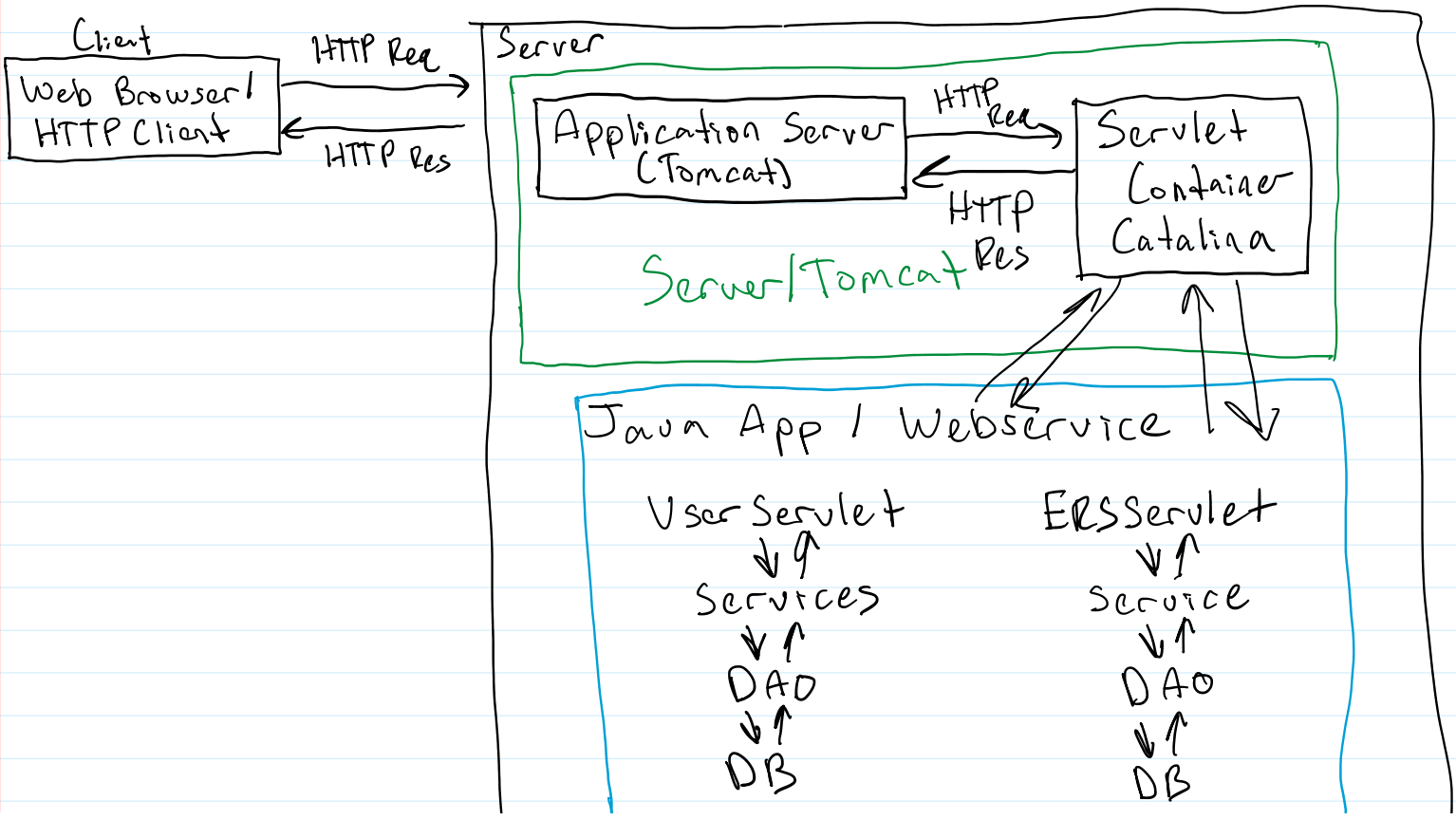
Application Server is handled to handle any type of request

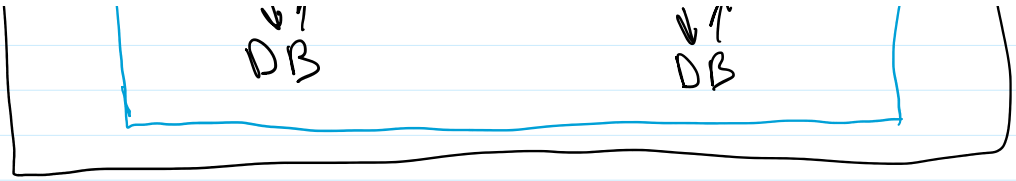
Most application servers double as a webserver

We will be using an application server known as tomcat
- Plays nicely with servlets and the servlet container

The Servlet Container:
- the component of some application servers which interacts with java servlets
- responsible for the life cycle of the servlets
- responsible for mapping URL's to servlets
- responsible for the URL requester to have the correct access

The built in Tomcat Servlet container is known as Catalina
- modified through web.xml





Web Services:

Is a software that allows machines to exchange information across a network
- Client Server Architecture is the idea of separating clients from servers
- Webservices are the actual applications running the server for client server arch.

These services must adhere to some set of standards and they should be exposed by some API

Two types:
- REST
- SOAP

Advantages of Webservices
- Use the web as transactional tool
- Expose functionality of a business service both internally and externally
- Capitalize on existing standards and create new ones
- Hardware and OS independent
- Loose coupling

Introduction REST

Representational State Transfer
- Style that outlines communication between a client and server

Rest Constraints

To restful you must follow these guidlines

1. Uniform interface:
   - the service should adhere to a commonly decided API standard
   - Your URL's to be unified & follow the same pattern

The service should adhere to a commonly decided API standard
- Your URL's to be unified a follow the same pattern
- the results/messages to be self descriptive

2. Client-Server: separation of concerns

3. Stateless: the server should not keep track of app/user state

4. Cacheable: responses should be labeled as cacheable or not

5. Layered System: the app should be organized in a way such that only components on the same "layer" communicate

6. Optional: Code on demand
- the ability to send entire scripts/code through requests

Resources in REST
- anything/any information which can be named
- models
- JSON data or XML sent to the client

We will identify which resource we want by the URL
- Unified Resource Locator
- Also known as resource identifier

Some rules for uniform resource indentifiers
- use nouns to name resources
- begin with plural to get a collection /users
- to specify a specific resource use path variables/parameters /user?id=2
- capitalize on path structure to represent hierarchy /user/login
- indentify stores/subcollections of resources

The biggest thing is start a pattern and only use that pattern

Rest Content Negotiation

Rest content negotiation

Process of selecting the data representation for a given request

This is done on the serverside based on the content type in the request header
- Content-Type in the header followed by the type of content

Or you could do it through the URL
- add the file extension onto what you are looking for
  /person.json → json

  /person.html → html

Richardson Maturity Model

The process of taking an app and making it Restful

0. Start with HTTP
   - Introduce HTTP interaction with our App
1. Introduce the resources
   - Start declaring endpoints for your resources
2. Verbs of HTTP
   - Take it further by allowing more actions tied to HTTP verbs
3. Hypermedia Controls
   - tell us what we can do next, and a why of recieving info back from the server

Intro to HTTP

Hypertext Transfer Protocol

Allows clients to send information to servers and servers to send information back

HTTP Request

Sent from the client to the server

The request is composed of:
- Verb: (method)
- URI: the endpoint to the resource
- HTTP version
- Request Headers: info / metadata
- Request body: information sent to server

HTTP Response

Sent from the server to the client

The response is composed of:
- Status Code: a number which tells some info about the success or failure of a request
- HTTP Version
- Respons Headers: metadata
- Response Body: information sent back to the client

Verbs and their Characteristics

Two different characteristics

Idempotent:
- the request can made multiple times in a row and return the same result

Safe:
- the request does not alter the state of the server / database

GET:
Used to retrieve data / get data
No request body
Idempotent
Safe
Cacheable
allowed in html forms

POST
Send data to the server
often used to create or update data
Not Safe
Not idempotent
Not cacheable

Not safe
Not idempotent
Not cacheable
allowed in html form
Has request body

PUT
- used to update and replace a resource
- is idempotent
- not safe
- not cacheable
- not allowed in Html forms
- request body

DELETE
- used to delete a resource
- may or maynot have a request body
- not safe
- idempotent
- not cacheable
- not allowed in Html forms

HTTP Status Codes

Give info about our requests

Informational codes: 100 - 199

Successful codes: 200 - 299
- 200 OK
- 201 CREATED
- 202 ACCEPTED
- ... and more

Redirects: 300 - 399
- 300: Multiple Choices
- 301: Moved Permanently
- 302: found

Client Errors: 400 - 499
- 401: Unauthorized
- 403: forbidden
- 404: Not found

Server Errors: 500-519
- 500 Internal Server Error
- 503 Service Unvailable