Servlet: java classes that can handle HTTP Requests and Responses from web clients.

Web server is a program that uses HTTP to serve the files from Web pages to users, in response to their requests.

Application servers is a software framework that provides facilities to create web applications and a server environment to run them.

Client-server interaction is a distributed application structure that partitions workloads between the servers, and clients.

Web container -- Tomcat:

* + Tomcat is “Thread-per-request”
  + Checks whether any thread is available to cater to an incoming request
  + Server can process however many concurrent requests as there are threads in the thread pool managed by Tomcat(~200)
  + Tomcat’s thread pool is represented by the executor.

Servlet hierarchy

* + Servlet Interface ---> Generic Servlet ---> HttpServlet ---> MyServlet
  + The servlet interface is the root of the hierarchy.
  + When developing HTTP we must extend HttpServlet class on that servlet.

Servlet lifecycle: init(), service(), destroy().

How is a request processed with servlets:

* Client sends http request
* Server receives request.
* Servlet container consults web.xml – Deployment descriptor – to map the request to appropriate servlet.
* Servlet container instantiates the servlet.
* Container calls the init() method of the servlet – the first step in the servlet life cycle – to set any parameters specified in the DD.
* Container calls the servlet’s service () method – this processes the request and invokes the request handler’s method.
* HTTP response is returned by servlet to the server then to the client.
* Each successive request to the server which maps to this servlet invokes service().
* Container calls the servlet’s destroy () method, releasing the instance to save memory. After the active period for the servlet has elapsed, container shut down.

Web.xml, deployment descriptor is the file with which we configure our web apps. We indicate which url-patterns (endpoints) will map requests to respective servlets and configure other details like parameters and when to initialize the servlets.

Important tags: <servlet> and <servlet mapping>

ServletConfig: is an interface and is used to initialize a single servlet in a web app by the servlet container. We put it inside web.xml.

ServletContext: allows objects to be shared to any servlet in the app.

PrintWriter allows you to write responses

Load on startup is defined within the .js file. The method that is called within window.onload is the first HTML that the app will show when the server is initialized.

GenericServlet is a super class of HttpServlet class and generic protocol independent servlet.

HttpServlet is a protocol dependent. Allows you to access a variety of methods specific to the HTTP protocol

doGet: used to make an HTTP GET request

Sends the encoded user information appended to the page request

doPost: used to make an HTTP POST request.

The servlet container must write the headers before committing the response, because in HTTP the headers must be sent before the response body.

doPut: for HTTP PUT request, allows a client to place a file on the server

doDelete: for HTTP Delete Requests, allows a client to remove a document or web page from the server

Init and destroy, to manage resources that are held for the life of the servlet

* + - Inti called by the servlet container to indicate to a servlet that the servlet is being placed into service
    - Destroy is called by the servlet container to indicate to a servlet is being taken out of service.
  + getServletInfo: which the servlet uses to provide information about itself.
* ObjectMapper provides functionality for reading and writing JSON. ObjectMappers are thread-safe.

HttpServletRequest receives data sent by client, like user name and password. Used many get() methods

HttpServletResponse sends data back to the client. Uses many set() methods

RequestDispatcher dines object that receives requests from the client and sends them to any resource.

Methods include: Forward: forwards to another source, Include: includes content in response

Forward:

* The request will be further processed on the server side.
* URL in a browser stays the same.
* Req & Resp objs will remain the same objects after forwarding.

Redirect:

* Request is redirected to a different resource
* Client will see URL change after redirect
* A new request is created
* Used within POST/REDIRECT/GET
* In general, a forward should be used if the operation can be safely repeated upon a browser reload of the resulting web page, otherwise, redirect must be used. Typically, if the operation performs an edit on the datastore, then a redirect is required. This is simply to avoid the possibility of inadvertently duplicating an edit to the database.

Session management

* Cookies: small piece of information persisted between multiple client request. Have names, values, optional attributes. STORED IN THE BROWSER CACHE.
* PROS: simplest form of session mgmt maintained client side.
* CONS: text-only and cookies can be disabled

URL rewriting: appendID to url.

* PROS: works regardless of cookies being disabled, doesn’t require extra form submission.
* CONS: text-only and only works with links
  + Hidden Form Fields- use invisible fields to track user state
  + Session Object(HTTPSession)- use the req.getSession() method to obtain the current session.
* Asynchronous requests with Servlets
* Sending html as response

The front controller design pattern means that all requests that come from a response in an application will be handled by a single handler and then dispatched to the appropriate handler for that type of request.

TYPESCRIPT is a superset of JS, meaning that any valid JS code is also valid TS code. TS code must be TRANSPILED into JS code.

TS Data Types: strong typing

* + - Number, boolean, string, void, null, undefined, never, any.
    - A variable initialized with undefined means that the variable has no value or object assigned to it whereas null means that it has been set to an object whose value is undefined

Decorators: Special kind of declaration that can be attached to a class declaration, method, accessor, property, or parameter.

Access modifiers: Public and Private, by default the members are public

Classes:

Similar to classes in most OOP languages

Properties are made public by default but can be made private

In classes, functions are called methods

Abstract Classes, interfaces same as java.

ANGULAR

* Npm: package manager for javascript
* Angular: is a TypeScript-based open-source front-end web application platform.
* CLI: Command-line interface, is a command line tool for creating angular apps

Components

* + Ui building block of an angular app
  + Decorator that marks a class as an angular component and provides configuration metadata that determines how the component should be processed, instantiated, and used at runtime.
* Modules
  + Refers to a place where you can group the components, directives, pipes, and services, which are related to the application.
  + To define module, we use the NgModule
* Templates: HTML
* Data binding: a mechanism for coordinating parts of a template (HTML) with parts of a component (TS). Allows you to define communication between a component and the DOM.
* Directives: changes the appearance or behavior of a DOM element
  + There are three kinds:
    - 1. Components: directives with a template
    - 2. Structural directives: change the DOM layout by adding and removing DOM elements: ngIf, ngFor, ngSwitch
    - 3. Attribute directives: change the appearance or behavior of an element component, or another directive: ngClass, ngStyle.
* Services:
  + Great way to share information among classes that don’t know each other.
* Dependency Injection
  + Increase efficiency and modularity
  + Dependency injection (DI) is a coding pattern in which a class asks for dependencies from external sources rather than creating them itself.
  + DI framework provides declared dependencies to a class when that class is instantiated.
* Pipes: takes in data as input and transforms it to a desired output
* HttpClient offers a simplified client HTTP API for angular applications that rests on the XMLHTTPRequest interface. Additional benefits of HttpClient include testability features, typed request and response object, request and response interception, observable apis, and streamlined error handling.
* RxJS
  + Reactive Extensions for JavaScript is a library for reactive programming using observables that makes it easier to compose asynchronous or callback-based code
  + Reactive programming is an asynchronous programming paradigm concerned with data streams and the propagation of change.

Routing enables navigation from one view to the next as users perform application tasks.

Design Patterns

() = “”: event binding

[] = “” property binding

[(ngModel)]: two way binding

Guard: strict on route

Hibernate: JPA (Java Persistence API).

An ORM tool: object relational mapping tool. Maps Java classes to database tables.

HQL (hibernate Query Language): Use hibernate query language, like SQL, but database agnostic dialect.

Can use Hibernate with an existing DB or use hibernate mappings to create tables & other DB entities.

Important interfaces in the Hibernate API

* Session: Represents your session with a database. Methods: save, get, update, delete, beginTransaction, createQuery, createCriteria.

Session session = sessionFactory.openSession();

session.close();

* SessionFactory: Used to create session objects, Only need one sessionFactory
* Configuration: used to create a sessionFactory, configured via XML -> hibernate.cfg.xml
* Criteria can only retrieve from database, cannot modify anything
* Transaction: manages ACID – compliant interactions with DB. i.e. tx.commit();
* Query: Can do more complicated CRUD than the session method., Use hibernate query language (HQL)

Hibernate object states: describe the state of an object in Java with relation to a row in DB.

Transient: no session associated with this object. Just instantiate using “new” keyword. No persisted representation in DB no PK value. Dereferenced transient objects are eligible for garbage collection.

Persistent: object has representation in DB (and has a unique identifier). Hibernate will depict any changes made to an object in this state and will reflect changes at the end of the transaction.

Detached: object was once persisted, but session has been closed, still has representation in DB, but changes to this object will not be reflected in DB, unless it is reattached to a session.

Hbm2ddl

* + Update: updates schema
  + Create: creates schema, destroys previous data
  + Validate: makes no changes, validates data
  + Create-drop: Drop the schema when session factory is closed

ATOMICITY – “all or nothing” – either all operations of the tx execute successfully, or no commit is made.

Consistency – DB is in a valid state according to existing structure & constraints after a commit.

Isolation – The system state during concurrent transactions is the same as if the tx’s were sequential.

Durability – All commits are final & cannot be rolled back – even in case of system failure.

Session Methods:

Retrieve Data:

Get: immediately hits DB, eager fetching, returns null if obj not found.

Load: “lazily fetches” object from DB returns a proxy until a method is called on the object (in session).

Proxy – hibernate term for a sort of placeholder objects. Has correct ID. Only useful within session. Throws exception if attempting to retrieve object that does not exist.

Inserting Data:

Save – immediately return ID of object. Can execute insert statement outside of transaction.

Persist – make a transient object persistent. Void return type.

Insert executes within transaction. Good when you don’t immediately need the ID.

Changing Data:

Update: update object in DB, bring detached object into persistent state.

Can throw NotUniqueObjectException if you provide a transient / detached obj with the same ID as the persistent obj.

MERGE – checks whether persistent obj with given IO exists.

Creates new persistent obj if none is present, copies over data from object provided to persistent object.

Save or Update() – generally brings object into persistent state, regardless of initial state.

Automatic dirty checking: when session is closed, hibernate checks for changes in any persistent objects and puts them into the detached state.

Transactional write behind in the context of a transaction, changes made to an object in the persistent state are not immediately propagated to the DB.

Transitive persistent (cascades): No default cascading of state between related entities.

Available cascade styles: create, merge, save – update, delete, lock, refresh, evict, replicate

Typically cascade one to many, many to one, not one to one or many to many.

Hibernate Caching: improves the performance of the application by pooling objects in the cache. 2 levels:

Level 1: default, bysession

Level 2: must configure with vendor, session factory.

Caching is a mechanism to enhance the performance of a system. It is a buffer memory that lies between the application and the database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.

Any third - party cache can be used with Hibernate

Concurrency strategies: transactional, read-write, non-strict read-write, read-only.

Cache provider:

EHCache: can cache in memory or on disk and clustered caching and it supports the optional Hibernate query result cache.

**OSCache**

Supports caching to memory and disk in a single JVM with a rich set of expiration policies and query cache support.

**warmCache**

A cluster cache based on JGroups. It uses clustered invalidation, but doesn't support the Hibernate query cache.

**JBoss Cache**

Devops

DevOps is a set of practices that automates the processes between software development and IT teams, in order that they can build, test, and release software faster and more reliably.

**IAS:** infrastructure as service

PAS: platform as service

SAS: software as service

Continuous integration is a testing automation to check that the application is not broken whenever new commits are integrated into the main branch.

Continuous delivery: every code change is built, tested, and then pushed to a non-production testing or staging environment.

Continuous deployment goes one step further than continuous delivery. Every change that passes all stages of your production pipeline is released to the customers.

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of the software development process, with continuous integration and facilitating technical aspects of continuous delivery.

Cassandra

A NoSQL database is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

Apache Cassandra is an open source, distributed and decentralized/distributed database, for managing very large amounts of structured data. It provides highly available service with no single point of failure.