**Week 2: Servlets, HTTP & REST**

*The following is an overview of the topics to be covered in Week 2.*  
*You will be asked to answer the questions at the****bottom****in QC and your 1-on-1's with me.*

**HTTP Study Guide:**

* Hyper Text Transfer Protocol
* Primary networking protocol used for client-server communication
  + The internet (world-wide-web)
* This is also specifically used for RESTful web services
* For now, it's important to understand the differences between different HTTP verbs (methods), such as GET and POST as well as HTTP status codes
* All HTTP messages are composed of a **header** and a **body** which contains different pieces of information..
  + The header normally represents the metadata about the request.
  + The body represents the data in the message, or the message itself.
* **GET vs POST**
  + **GET** is utilized to retrieve data. We're retrieving data from some url, or endpoint, and normally clients and servers are not expecting these messages to contain content within the body. (Although, you definitely still can)
  + **POST** is generally used to send/update information on the url/endpoint, which means that the server os expecting the message to have information in the body.
* HTTP responses have a status code that represent the status of the request. We'll talk more about them in a moment, but we have seen examples such as 200 OK, and 201 CREATED *see the****HTTP Status Codes****listed below*\*\*
* HTTP responses can also have a body. If a GET request was performed, generally,the response is stored in the body.
* **HTTP request contents**
  + HTTP version
  + URL
  + HTTP verb / method
  + Request Headers
  + Request Body
* **HTTP response contents**
  + HTTP version
  + Status code
  + Response Headers
  + Response Body
* **HTTP verbs**
  + GET
  + POST
  + PUT
  + DELETE
  + PATCH
  + HEAD
  + OPTIONS
  + TRACE

**HTTP Response status codes**

* 100-199: informational
* 200-299: success
  + 201: Created
  + 204: No content
* 300-399: redirect
  + 301: Moved permanently
  + 304: Not modified
* 400-499: client error
  + 403: Forbidden
  + 404: Not Found
  + 405: Method not allowed
  + 415: Unsupported media type
  + 451: Unavailable for legal reasons
* 500-599: server error
  + 501: Not implemented
  + 502: Bad Gateway
  + 503: Service unavailable

**Client/Server Architecture**

* A Client and a Server establish a connection according to a set of rules called a protocol.
* [**HTTP**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview) is a protocol which allows the fetching of resources, such as HTML documents.
  + It is the foundation of any data exchange on the Web and it is a **client-server protocol**, which means requests are initiated by the recipient, usually the Web browser (client).
* *Describe the difference between****2 Tier****,****3 Tier****,****n-Tier****architecture*

2 Tier architecture: The Client is one Tier, and the database and web application ‘live’ on one machine (Tier 2) (An application on the client’s desktop that, when requested, pulls information from the database, which is on a different computer.

3 Tier Architecture: it is a web based application. The Client, on their browser, accesses the application on a server (business logic (tier 3)) and if the application needs information from a database (tier 2) retrieves that from a separate computer.

‘n’ Tier Architecture: has several layers for both logical and physical tiers. Each layer having its own responsibilities. (Seems like a manager delegating tasks to employees)

**Web Services (REST & SOAP)**

A **web service** is a collection of open protocols and standards used for exchanging data between applications or systems.  
A **web service** is any service that:

* Is available over the Internet
* Uses a standardized XML messaging system
* Is not tied to any one operating system or programming language
* Is self-describing via a common XML grammar
* Is discoverable via a simple find mechanism

**What is a RESTful Web Service?**

A RESTful Web Service is a lightweight, maintainable, and scalable service that is built on the REST architecture.

* A RESTful Web Service will expose an API from your application in a secure, uniform, stateless manner to the calling client.
* The calling client can perform predefined operations using the RESTful service.
* The underlying protocol for REST is HTTP.

**What is REST?**

* REST stands for **RE**presentational **S**tate **T**ransfer.
* REST is a way to access resources which lie in a particular environment.
* [Overview of REST](https://restfulapi.net/rest-architectural-constraints/)

For example, you could have a server that could be hosting important documents or pictures or videos. All of these are an example of resources. If a client, say a web browser needs any of these resources, it has to send a request to the server to access these resources. Now REST services defines a way on how these resources can be accessed.

**6 REST Constraints (*know these and google what they mean!*) ⭐**

There are six architectural constraints which makes any web service are listed below:

1️⃣ Uniform Interface  
2️⃣ Stateless  
3️⃣ Cacheable  
4️⃣ Client-Server  
5️⃣ Layered System  
6️⃣ Code on Demand (optional)

**What is SOAP?**

* Simple Object Access Protocol: used to "expose" and "consume" webservices
* XML-based web service protocol
* Platform and language independent
* Legacy protocol: most companies switching to RESTful web services
* Can be used in conjunction with any transport-layer protocol (HTTP, SMTP, FTP, etc)
  + When used with HTTP, POST requests are used
  + HTTP must set that content type to XML
* Uses a contract (WSDL)
  + WSDL = Web service description language
  + Contract first vs contract last development
* Built-in security
* SOAP message elements
  + envelope - (mandatory) defines the start and end of a message
  + header - (optional) specifies attributes)
  + body - (mandatory) this is the XML data
  + fault - (optional) describes error that may have occured

Resources for the differences:

* [UpWork Resource](https://www.upwork.com/resources/soap-vs-rest-a-look-at-two-different-api-styles)

<xml version=1.0>

<soap-env:Envelope xmlns=soap-env=”www.w3.org/2001/12/soap-envelope” soap-env:encodingStyle=”www.w3.org/2001/12/soap-encoding”>

<soap-env:Header>

</soap-env:Header>

<soap-env:Body>

<soap-env:Fault>

</soap-env:Fault>

</soap-env:Body>

</soap-env:Envelope>

**WSDL**

* Web Service Description Language
* XML file that describes the service
* "Contract" or "Endpoint"
* WSDL elements are placed within the namespaces like this:
  + xmlns:xsd, xmlns:soap
* WSDL specifies ports, messaes, operations, services

**⁉️ SOAP Questions**

1. What does the acronym SOAP stand for?

Simple

Object

Access

Protocol

1. What protocols and data format do SOAP services use?

XML only

1. What is the “contract” for a SOAP service?

The contract is a collection of metadata describing various aspects of the software.

1. What’s the structure of a SOAP message?

Envelope – describes the beginning and end of the message

Head – optional\* attributes

Body – XML data being sent

Fault – optional\* error codes

1. What are the SOAP messaging modes? Messaging Exchange Patterns?

RPC/Literal, RPC/Encoded

Document/Literal, Document/Encoded

Request/Response – the client sends SOAP request message to the service and the SOAP service responds with a SOAP response message

Response – The client connects to a SOAP service, but does not send a SOAP request. They send an HTTP request instead. The service, however, sends back a SOAP response.

1. Are SOAP messages delivered with GET or POST requests? POST only

**⁉️ REST Questions**

1. What does the acronym REST stand for? What makes a service “RESTful”?

**RE** - representational

**S**tate

**T**ransfer

1. What protocols and data format do REST services use?

Uses HTTP protocol, and JSON format

1. What are the architectural constraints of REST?
   1. Uniform Interface: defines the interface between the clients and servers
   2. Stateless: the server does not hold any information about the state of the client. Every request made must contain all the necessary information.
   3. Cacheable: With several clients accessing the same server, and requesting the same resources, the responses of the clients are cached to avoid unnecessary processing, increasing performance.
   4. Client – Server: a uniform interface that separates the client from the server. Neither really care about the other, which means either can be altered easily without causing issues for the other.
   5. Layered System: providing different layers from client to end server to improve scalability, and enforcing security policies.
   6. Code-on-Demand: Customers can run their own code to the server through scripts or applets.
2. What is a “resource” in a REST service? Any information that can be named can be a resource. (or pretty much anything digital)
3. What does the “uniform interface” constraint mean? It defines the interface between the client and the server. (or multiple of each)

Give an example of some RESTful endpoints you would create for an API.

Should the URLs contain nouns, verbs, or adjectives?

**Servlet Study Guide**

*You are expected to know the following topics*:

* **Servlet Inheritance Hierarchy**
  + Servlet interface
  + GenericServlet abstract class
  + HttpServlet abstract class
* Servlet container
* Servlet lifecycle
* **Deployment descriptor**
  + What folder is it in?

Web-INF

* + web.xml tags

welcome file, display-name, servlet, servlet name, servlet-class, servlet-mapping, url-pattern, error-page, code, location

* Creating custom servlets
* Eager vs lazy loading / instantiation of servlets

Lazy is initialized by default

Eager is initialized when it is directed to, numbering from 0 – 255 (lower number is loaded first)

* ServletConfig object

Used on one server. Should be stored in the web.xml, and the information retrieved from there.

* ServletContext object

Used on multiple servers, contains the information about the entire application

* RequestDispatcher
* Forwarding vs redirecting
* **Session tracking**
  + HttpSession API
  + Cookies
  + URL rewriting
  + HTML hidden fields
* Retrieving request parameters
* Retrieving data submitted from a form
* Sending a plain text response from a servlet
* Sending a JSON object via mapping with Jackson
* Front controller design pattern

**Session Management**

* The practice of storing information about the user in the server in order to improve user experience

**Why do sessions matter**?

* User Experience: Saving session information locally prevents needless hits to your database, which improves performance and allows you to personalize a user's experience.
* Security: Verify that a user actually has access to a resource that was provided by your website. Disallow access to sensitive information.

**Sending Users to different locations on the Web / Redirects**

* Servlets take requests from clients and forward them to requested resources. How?
  + There are 2 ways:
    - SendRedirect
      * Supplied by Response object
      * This is a method: response.sendRedirect(location);
      * Actually send a response back to the client and then sends a new request back.
      * As a result, the information in the previous request is lost
      * Returns a 300 Series status code
    - Forward
      * Supplied by Request object
      * This method is declared on the RequestDispatcher Interface
      * When you use forward, the request never leaves the server
      * You can't request dispatch into another location (stay within same domain)
      * You make only 1 request, not 2
  + Main Takeaway:
    - Use sendRedirect when your client asks for a resource that is in some other location (the extra request is visible in the client as a new request)
    - Use forward if the your client is asking for a resource from the same project (something you have access to immediately)
    - Forward is handled by the server
    - sendRedirect is handled by the browser/client

**Servlet Config**

* ServletConfig (interface) provides objects that are used by the Servlet containers (aka web container) to pass information to a servlet during initialization
  + getInitParameter()
  + getServletContext()
* Serializable interface is a marker interface (an empty interface) that can be implemented in order to allow an object to be "serialized" (to basically save to a file) and "deserialized".
  + In particular, if our Servlet classes implement Serializable, then we can serialize them to allow the servlet to "survive" restarting the web container.
* GenericServlet is an abstract class which supports any protocol, HTTP, UDP, TCP, etc
  + It is not necessarily designed for HTTP
* HttpServlet abstract class which allows you to create a servlet suitable for handling http requests.
  + doPOST, doGET, etc
  + Note, HttpServlet is an abstract class, so it is meant to be extended

**ServletConfig vs ServletContext**

* The ServletConfig is unique to each individual servlet
  + No other servlet can access another's config
* The ServletContext is shared across all servlets

**⁉️ Servlet QC Questions**

1. What is a servlet? What about a servlet container? Which servlet container have you worked with?

Servlet: layer between client and server

Servlet container works with java to dynamically load web pages

Used Tomcat!!!

1. Describe the servlet class inheritance hierarchy. What methods are declared in each class or interface?

Interfaces: Servlet, ServletConfig, Serialization

Built-In-Classes: GenericServlet, HttpServlet

\*User-Defined-Class Servlet

1. How would you create your own servlet?

Uses \*\*doPost, \*\*doGet

Protected void \*\*(HttpServletRequest, HttpServletResponse) throws IOException, ServletException

1. What is the deployment descriptor? What file configures your servlet container?

Aka: web.xml file

Describes the classes, resources and configuration how web servers will respond to requests and what to do with them

1. Explain the lifecycle of a servlet - what methods are called and when are they called?

Init() called only once to create the servlet

Serve(): called only once per request

Destroy(): called only once to destroy the servlet

1. Is eager or lazy loading of servlets the default? How would you change this?

Lazy is defaultedly loaded

Changed in the web.xml file under <load-on-startup> (pick a number from 0 to 255)(lowest number loaded first)

1. What are some tags you would find in the web.xml file?

Servlet, welcome first list, display name, servlet name, servlet class, url-pattern, servlet mapping

1. What is the difference between the ServletConfig and ServletContext objects? How do you retrieve these in your servlet?

Config is used for one servlet, used in initialization

Context is used for multiple servlets, used in the entirety of the application

1. What is the purpose of the RequestDispatcher?

Interface to dispatch to another resource

1. Explain the difference between RequestDispatcher.forward() and HttpServletResponse.sendRedirect()

Forward: server-side redirect

sendRedirect: client-side redirect (invisable to the user)

1. What are some different ways of tracking a session with servlets?

Cookies, hidden form fields, url rewriting, httpsession (request.getSession())

1. What is object mapping? Any Java libraries for this?

Mapping allows us to relate object from our applications to data in our databases. We’re using jackson

1. How would you send text or objects back in the body of the HTTP response from a servlet?
2. What is the difference between getParameter() and getAttribute() methods?
3. Explain the front controller design pattern