Please Complete The Socrative Pulse Survey!

URL: gosocrative.com
Room Name: JAVAGOLD

For each of these unit tests:

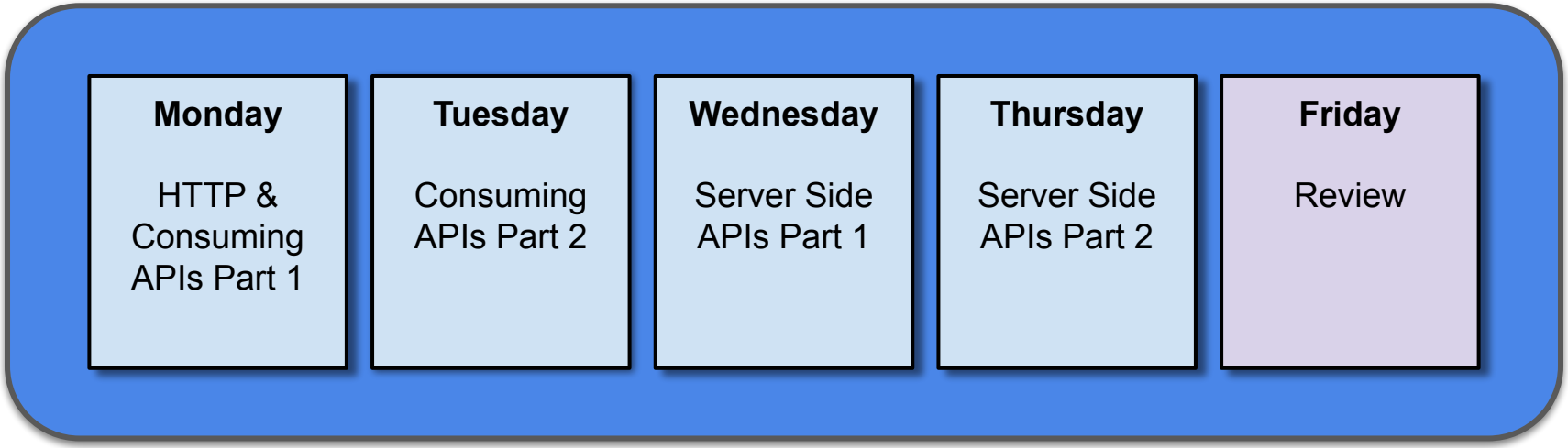
- be able to describe what each line is doing
- what the overall method is trying to achieve.
- determine an appropriate name for each test

```
@Test
public void what_am_I_testing_1() {
    dao.deleteAuthorById(2);

    Author retrievedAuthor = dao.getAuthorById(2);
    Assert.assertNull(retrievedAuthor);
}

@Test
public void what_am_I_testing_2() {
    Author newAuthor = new Author();
    newAuthor.setFirstName("TestFirst");
    newAuthor.setLastName("TestLast");

    Author createdAuthor = dao.createAuthor(newAuthor);

    int newId = createdAuthor.getAuthorId();
    Author retrievedAuthor = dao.getAuthorById(newId);

    Assert.assertEquals(newAuthor.getFirstName(), retrievedAuthor.getFirstName());
    Assert.assertEquals(newAuthor.getLastName(), retrievedAuthor.getLastName());
}

@Test
public void what_am_I_testing_3() {
    Author authorToUpdate = dao.getAuthorById(1);

    authorToUpdate.setFirstName("UpdatedFirst");
    authorToUpdate.setLastName("UpdatedLast");

    dao.updateAuthor(authorToUpdate);

    Author retrievedAuthor = dao.getAuthorById(1);
    Assert.assertEquals(authorToUpdate.getFirstName(), retrievedAuthor.getFirstName());
    Assert.assertEquals(authorToUpdate.getLastName(), retrievedAuthor.getLastName());
}
```

HTTP Web Services GET

Module 2: 11

Week 7 Overview



Today's Objectives

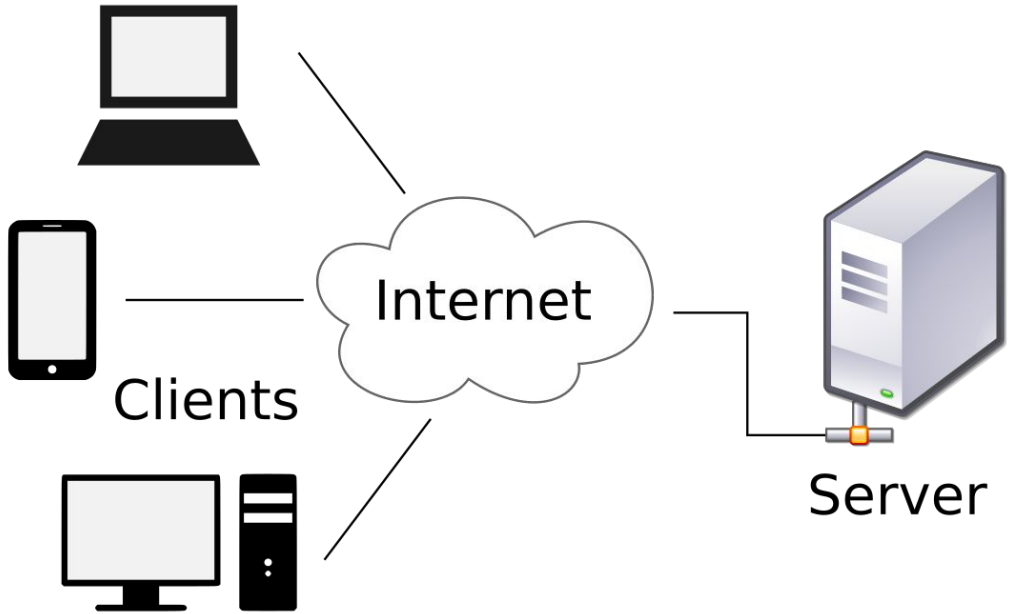
2. **APIs and Web Services**
    a. REST
    b. RESTful Web services
    c. JSON (JavaScript Object Notation)
3. **Consuming RESTful Web Services (API) with GET in Java**
    a. Endpoints
    b. RestTemplate
    c. Converting JSON to Java Objects
    d. Endpoint Parameters
        i. Path Parameters
        ii. Query String Parameters

Clients and Servers

Server is a generic term referring to either software or hardware that processes request from **clients**. A server offers shared **resources** that can be requested for use by a **client**.

A **client** is software that sends a request to a server to access a shared resource and processes the response returned from the server.

Client	Server
Web Browser	Web Server
Smart TV	Netflix's computers
Phone	Messaging Server
Email App	Email Server
DbVisualizer	PostgreSQL

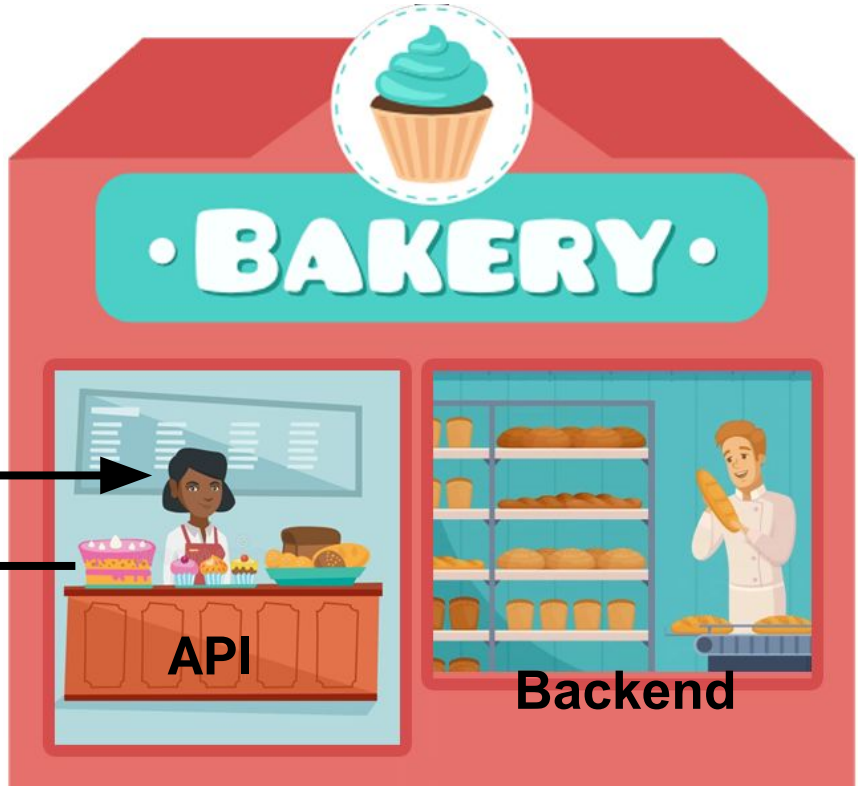


Client / Server

Client

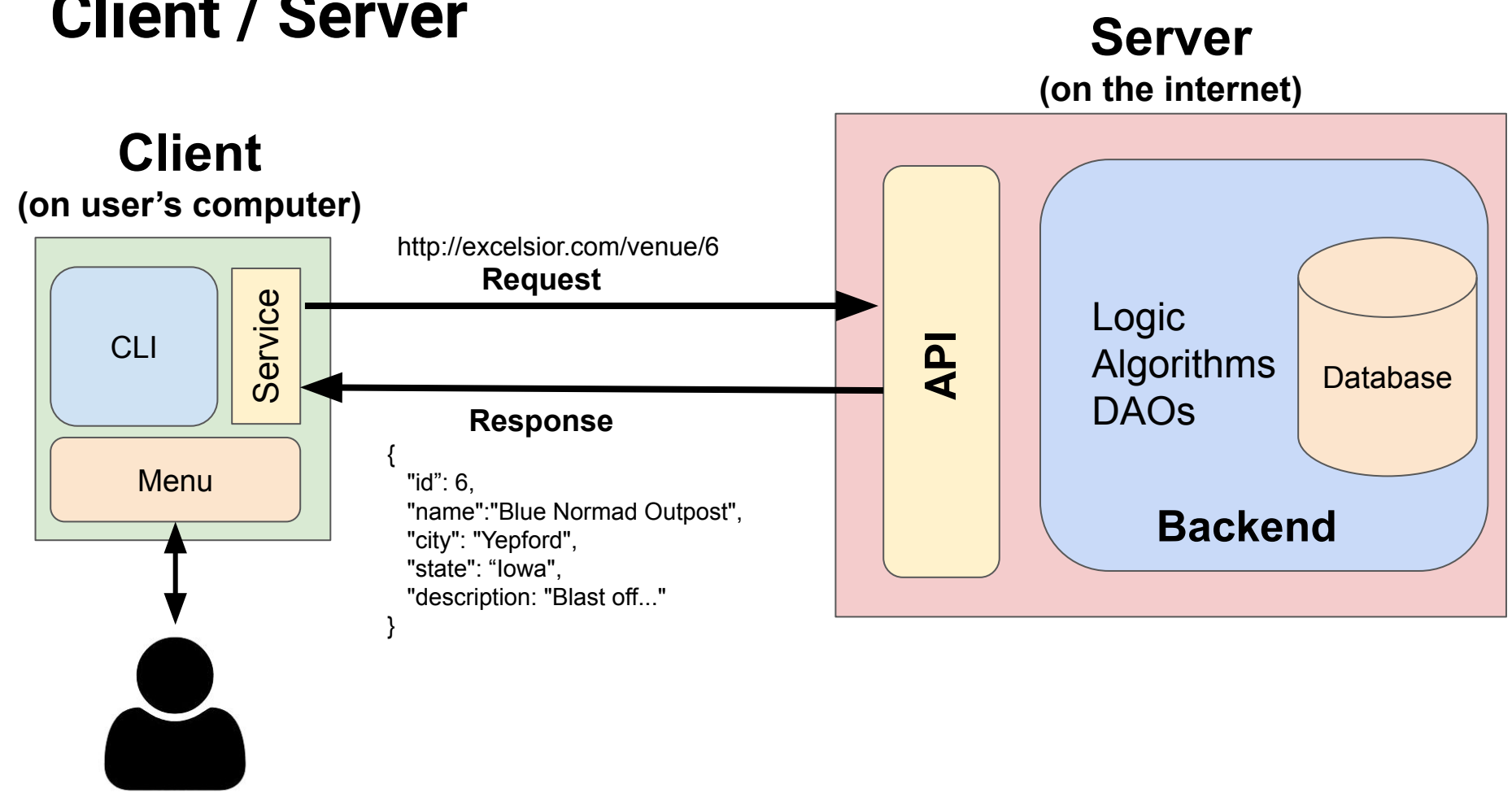
“I would like a cake”
Request

Response



Server

Client / Server

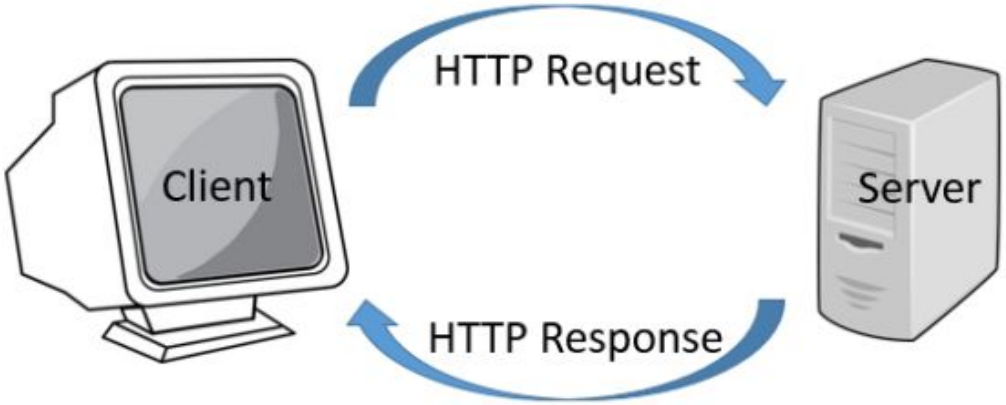


Request and Response

A client sends a request and the server replies with a response.

The HTTP request/response pair is **stateless**. Meaning that each request/response is independent and without context of previous request/response traded between the same client and server.

In HTTP, all communication between a client and server is via stateless request/response pair.



Stateless

Once a response is sent, the server does not retain any information about the request.

The server does not remember that the client sent a previous request, any details of that request, or anything about the client. Each request is independent and must include all information needed by the server to respond.

URL (Universal Resource Locator)

A URL (Universal Resource Locator) tells a client how to make a request to a server for a specific resource.

http://www.techelevator.com:80/events/current?month=march&day=27#00h02m30s

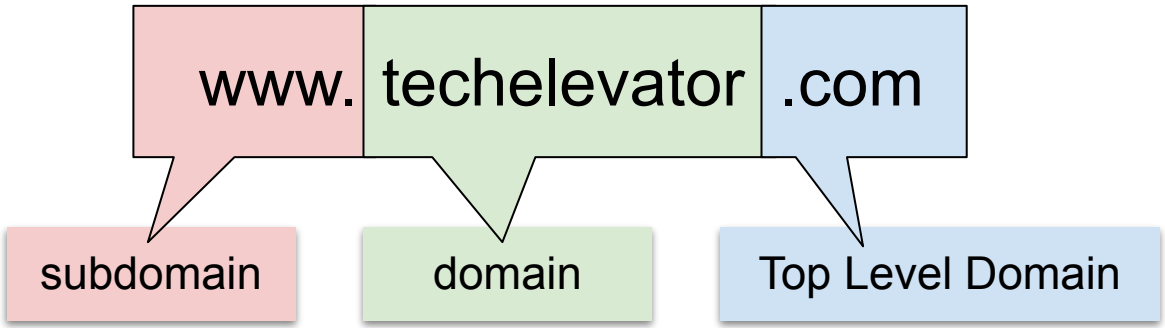
http://	**Protocol** (the language being spoken between client and server)
www.techelevator.com	**Domain** (the address of the computer the server is on)
:80	**Port** (identifies the server on the hosting computer)
/events/current	**Path to resource** (what is being requested)
?	**Query** (indicates parameters are being sent)
month=march&day=27	**Parameters** (keys with values for the server to use)
#00h02m30s	**Anchor/Fragment** (information that the server should return to the client)

Domain Names

Domain names are composed of parts, each separated by a period. Domains start a top level domain and then form a hierarchy of subdomains, each a child of the higher level domain.

Top Level Domains: .com, .net, .org, .gov, .io, …

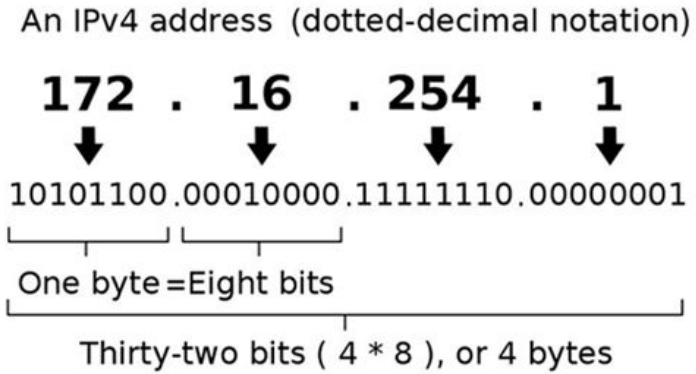
The hierarchy of a domain name is read right-to-left.

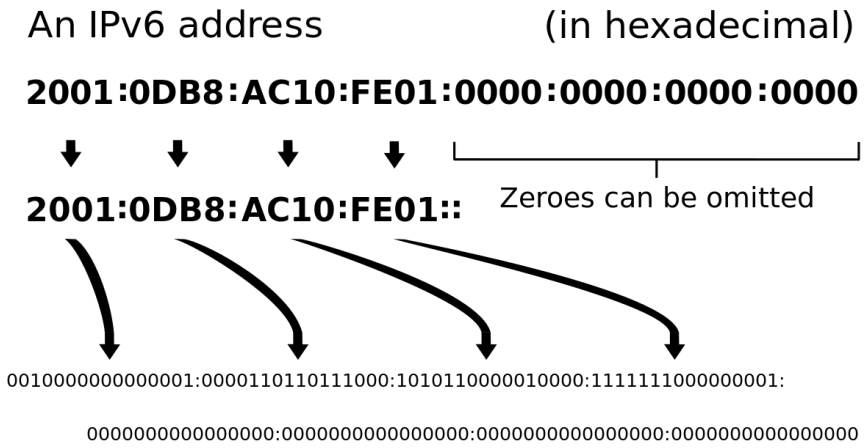


IP (Internet Protocol) Addresses

An **IP Address** identifies a computer or device on a network (including the internet).

All devices/computers on a network must have an IP address to communicate with other devices/computers on the network. It gives the location of the device like the street address of a house.



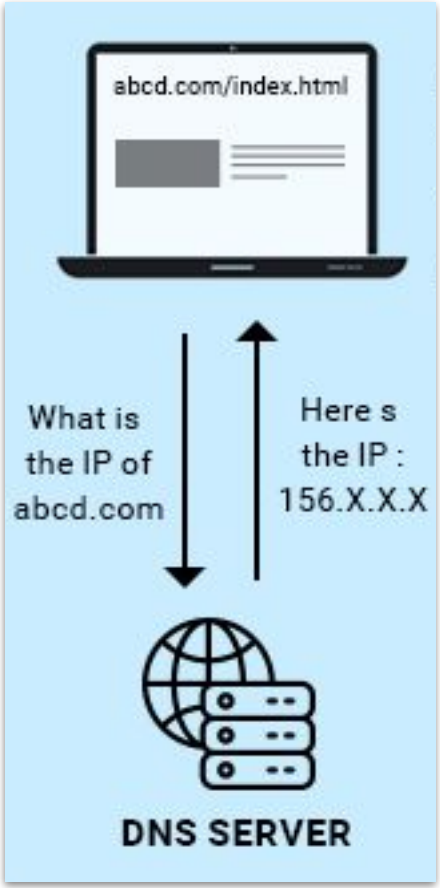


DNS (Domain Name System)

IP Addresses are understood by the computer, but are not easy to use for humans.

The **Domain Name System (DNS)** service allows for easy to remember names for humans to use. It works like a phone book, we give it the human relatable name and it returns the IP address for the computer to use.

Domain Name: techelevator.com
IP Address: 198.49.23.144



Port

There can be multiple software **servers** running on a single computer. Port numbers are used to identify which server application should handle the request.

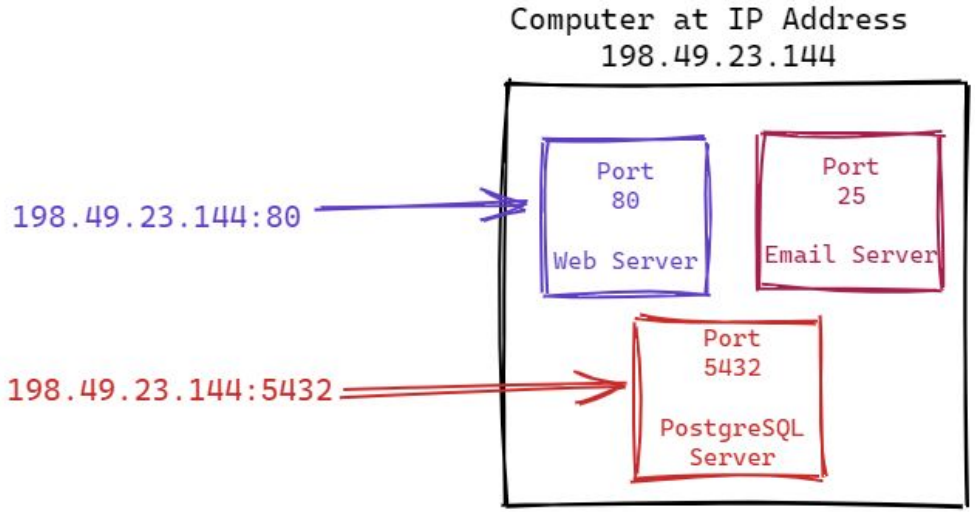
> *IP Address is like the street address of a building, and the port number would be equivalent to an apartment number in that building.*

The Port is added after the IP Address separated by a colon: 198.49.23.144:56

Port numbers range from 0-65535. Ports in the 0-1023 range are referred to as *well-known ports* and designated for specific uses.

Common well-known ports:

- Port 80 - HTTP
- Port 443 - HTTPS
- Port 25 - SMTP (Email Mail)
- Port 21 - FTP (File Transfer)
- Port 22 - SSH (Secure remote terminal)



HTTP (HyperText Transfer Protocol)

A **Protocol** defines the rules governing how clients and servers will communicate. It is a defined language and process that defines how a client should make a request and the server will return the response.

HTTP is the main **Application Protocol** used for the World Wide Web and is how browsers and web servers communicate.

HTTP communicates using a *stateless* **request and response**.

Parts of a HTTP Request

1. Method
2. Requested Resource
3. Header
4. Parameters

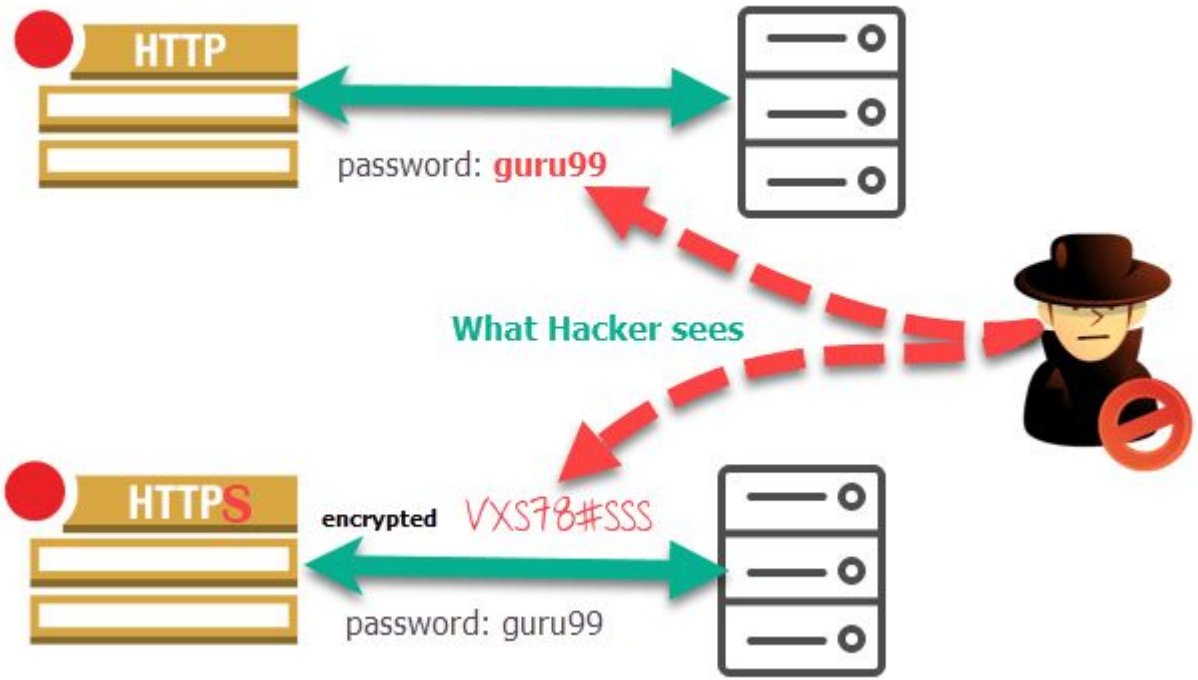
Parts of a HTTP Response

1. Status Code
2. Header
3. Content Type
4. Content (Body)

HTTP vs HTTPS

HTTP sends all data as plain text that is readable by anyone.

Like mailing a letter on a postcard.



HTTPS encrypts data using TLS (Transport Layer Security), so data is not readable.

Like mailing a letter in a sealed envelope.

-G

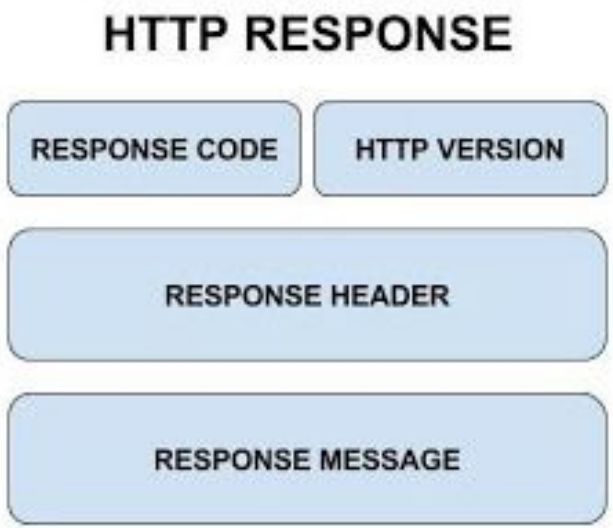
HTTP Request

1. **Method (VERB)**
   a. GET - Retrieves information
   b. POST - Adds information
   c. PUT - Updates information
   d. DELETE - Deletes information
2. **Requested Resource**
   A resource is something shared by the client. It can be an HTML page, CSS, a file like an image or pdf document, information from a database, access to a stream, or anything else the server can deliver
3. **Header**
   The header contains details about the request, such as the browser, IP address, prefered language, etc.
4. **Parameters**
   Parameters include data to be used to fulfill the request. Similar to arguments sent to a Java method.


```
Request Header:
GET / HTTP/1.1
Host: www.msaleh.co.cc
User-Agent:Mozilla/5.0 (Windows; U; Windows
NT 5.1; en-US; rv:1.9.0.10)
Gecko/2009042316 Firefox/3.0.10
Accept: */*
Connection:close
```

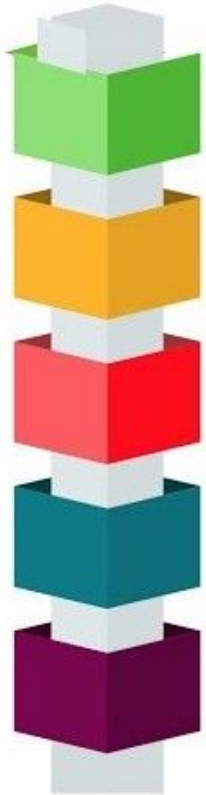
HTTP Response

1. **Status Code**
   The HTTP response status code indicates whether or not the request was successful.
2. **Header**
   Meta Information about the response.
3. **Content Type**
   The type of content being returned: text, json, image, stream, etc.
4. **Content (Body)**
   The data of the response. HTML, image, data, video stream, etc.




```
Response Header:
HTTP/1.1 200 OK
Date: Sat, 09 May 2009 12:27:54 GMT
Server: Apache/2.2.11 (Unix)
Last-Modified: Thu, 12 Feb 2009 15:29:42
GMT
Etag: "c3b-462ba63a46580"-gzip
Cache-Control: max-age=1200, private,
proxy-revalidate, must-revalidate
Expires: Sat, 09 May 2009 12:47:54 GMT
Accept-Ranges: bytes
Content-Length: 976
Content-Type:text/html
```

HTTP Response Status Code



Status Code	The Server is Saying...
1XX INFORMATIONAL	I need to tell you something
2XX SUCCESS	Everything is OK. 200-OK 201-Created
3XX REDIRECTION	You need to go somewhere else for that
4XX CLIENT ERROR	You messed up. 400-Bad Request 401-Unauthorized 403-Forbidden 404-Not Found
5XX SERVER ERROR	I messed up. 500 - Server Exception

418 I'm a teapot

The HTTP `418 I'm a teapot` client error response code indicates that the server refuses to brew coffee because it is, permanently, a teapot. A combined coffee/tea pot that is temporarily out of coffee should instead return 503. This error is a reference to Hyper Text Coffee Pot Control Protocol defined in April Fools' jokes in 1998 and 2014.

Some websites use this response for requests they do not wish to handle, such as automated queries.

Status

418 I'm a teapot

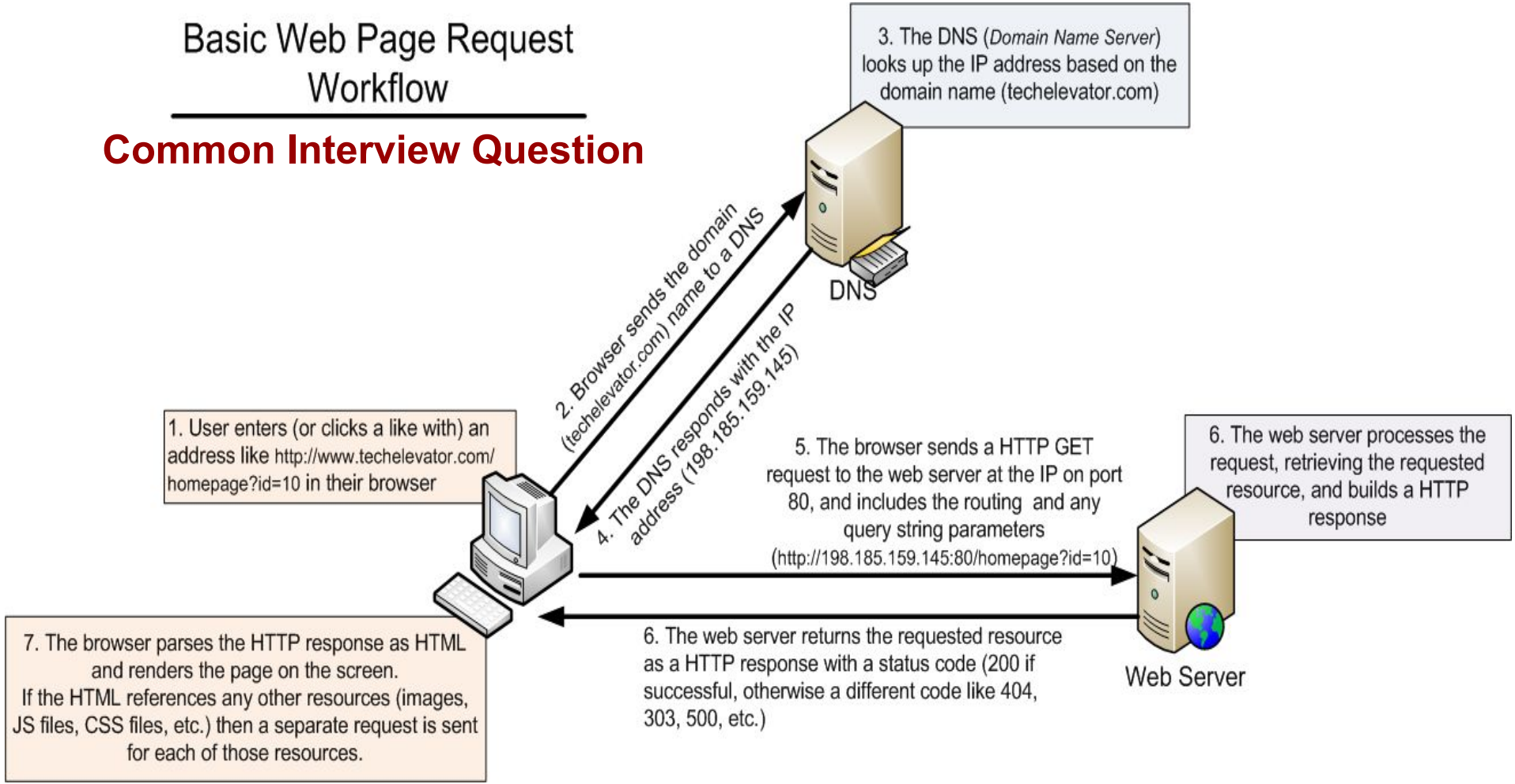
Copy to Clipboard

Specifications

Specification
RFC 2324 # section-2.3.2

Basic Web Page Request Workflow

Common Interview Question



API

Application **P**rogramming **I**nterface

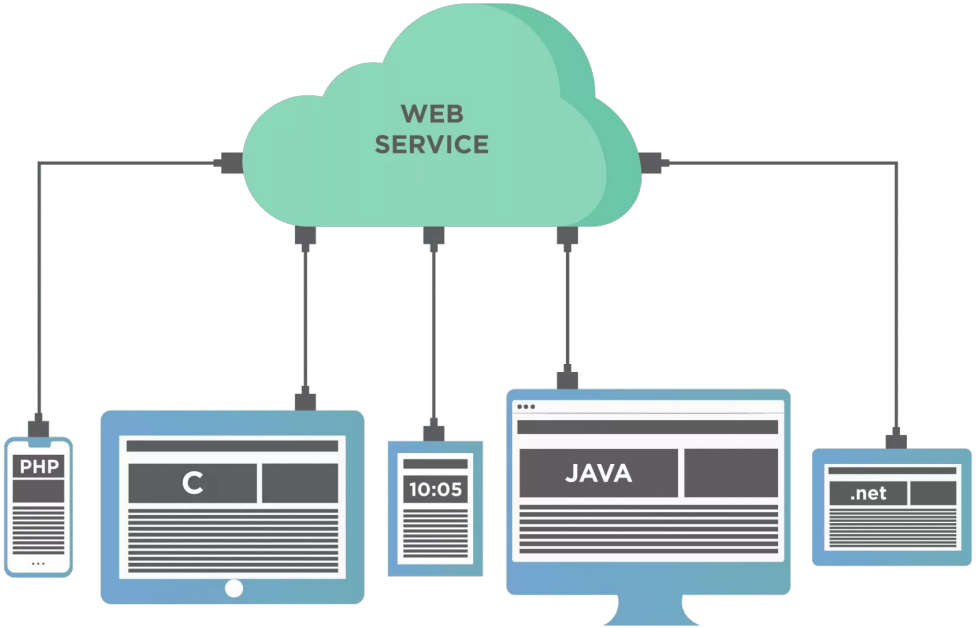
An API is a way for software components to talk to each other. The methods and properties in our classes that are exposed as public are the API of the object.

APIs allow communication between any software components, not just the ones in the same application. For example, browsers have an API that allows their functionality to be communicated with from JavaScript programs running inside them. When we use File.createNewFile() in Java it is calling the Operating System's API to create the file.

APIs commonly allow 2 applications to communicate.

Web Service (Web API)

A Web Service or Web API is an API that allows for applications to communicate with each other across the internet. *All Web Services are APIs, however, not all APIs are Web Services.*



REST
(REpresentational State Transfer)

A *software architectural pattern* that defines a set of constraints and standards for how 2 applications on a network should communicate.

Rules of REST

1. Uniform Interface (URI, URL)
2. Stateless
3. Cacheable
4. Client-Server based
5. Layered System

REST is not a technology.

REST is a set of rules that specifies the best practices for how communication on the web should work.

Do you want to know more?

RESTful Web Services (Web API)

- A Web Services that follow ALL of the REST rules, such as a uniform interface, uniform responses, and properly using HTTP Methods.
- Most commonly use HTTP(S) as the **application protocol**.
- Commonly uses **JSON** as a response.
- Also commonly referred to as *REST API, REST Web API,* or *REST Web Service*

Request Types to an API

We will see that a REST controller can be configured to handle various types of requests:

- **GET**: Intended to ***retrieve*** the predefined scope of data from a REST endpoint.
- GET (with path variable): path variables (i.e. puppy/1) allow client to retrieve a single record or a desired set of data.
- **POST**: Used for ***inserting*** new data into the data source.
- **PUT**: Used for ***updating*** an existing record within a data source.
- **DELETE**: Ideally suited for ***removing*** an existing record from the data source.

The Focus of Mod 2 Day 11(Lecture 9) is on GET requests and how they are consumed using Java.

JSON

JSON (JavaScript Object Notation) is a messaging format that is commonly used by **RESTful** Web Services.

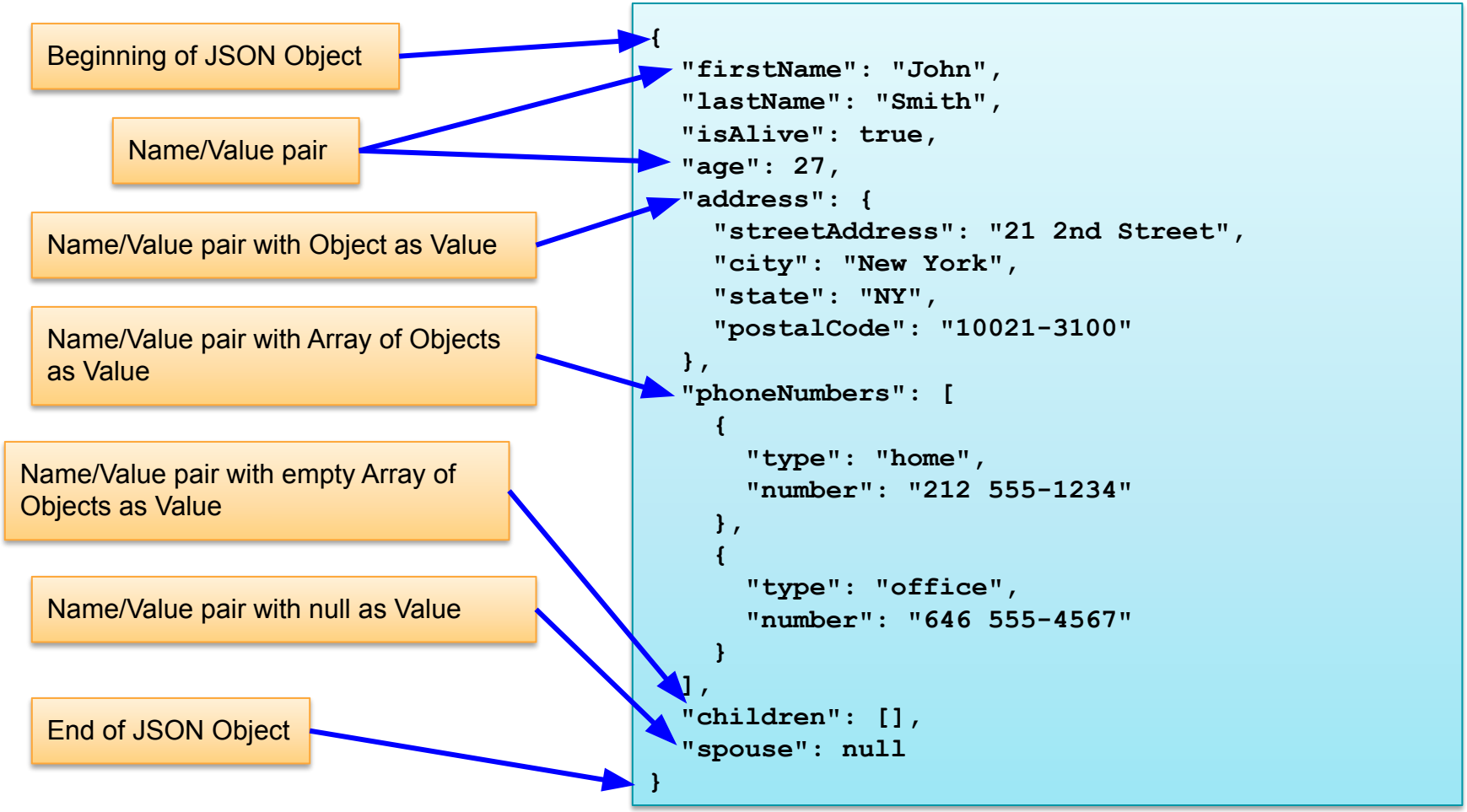
JSON uses key/value pairs to represent objects.

Objects are identified by **{ }**, Arrays by **[ ]**

Objects have **properties** and **values** separated by a **colon**. The property names are Strings in double quotes. The value can be a number, String (with double quotes), Array, or Object.

Each property/value pair, object, or array is separated by a **comma**.

```
{
  "reviews": [
    {
     "hotelID": 1,
     "title": "What a great hotel!",
     "review": "I thought this was a really great
hotel and would stay again!",
     "author": "John Smith",
     "stars": 4
    },
    {
     "hotelID": 1,
     "title": "Peaceful night sleep",
     "review": "I had a really good night sleep
and would stay again",
     "author": "Kerry Gold",
     "stars": 3
    }
  ]
}
```


Beginning of JSON Object
Name/Value pair
Name/Value pair with Object as Value
Name/Value pair with Array of Objects as Value
Name/Value pair with empty Array of Objects as Value
Name/Value pair with null as Value
End of JSON Object
{
"firstName": "John",
"lastName": "Smith",
"isAlive": true,
"age": 27,
"address": {
"streetAddress": "21 2nd Street",
"city": "New York",
"state": "NY",
"postalCode": "10021-3100"
},
"phoneNumbers": [
{
"type": "home",
"number": "212 555-1234"
},
{
"type": "office",
"number": "646 555-4567"
}
],
"children": [],
"spouse": null
}

Serialization & Deserialization

Java Objects can easily be converted to JSON data and vice versa, Mapping from an Object to JSON is known as **serialization** of an Object. Mapping from JSON to an Object is known as **deserialization**.

Java

```
Review[];

public class Review {
    private int hotelID;
    private String title;
    private String review;
    private String author;
    private int stars;
}
```

Serialization
Object to String

Deserialization
String to Object

JSON

```
[
    {
        "hotelID": 1,
        "title": "What a great hotel!",
        "review": "Great hotel,
        "author": "John Smith",
        "stars": 4
    },
    {
        "hotelID": 1,
        "title": "Peaceful night sleep",
        "review": "Would stay again",
        "author": "Kerry Gold",
        "stars": 3
    }
]
```

- A module of the Spring Framework that focuses on microservices and allows building stand-alone applications with minimal or no configuration.
- Widely used to consume or build RESTful services.
- Abstracts much of the repetitive and tedious work needed to create or use a Web Service.
- Contains an embedded application/web server (Tomcat).

Making a GET Request using Java

The **RestTemplate** class provides the means with which we can make a request to an API. Here is an example call:

```
RestTemplate restTemplate = new RestTemplate(); // Create a new client

ResponseEntity response = restTemplate.getForEntity(
                "https://api.exchangerate-api.com/v4/latest/USD",
                String.class); // Make GET request using Client

System.out.println(response.getBody()); // your return data returned from .getBody()
```

`response.getBody()` returns a `String` representation of the JSON, just like if you saw the API response in your browser. In the next slide, you'll see how RestTemplate can automatically convert the response data into a Java Object.

Making a GET Request with the Result Mapped to a Java Object

The RestTemplate class provides a way to make a request to an API and treat the result as a Java Object. The JSON return by the API call will be mapped to the specified Java class. Here is an example call:

```
private static final String API_BASE_URL = "http://helpful-site/v1/api/data";
private static RestTemplate restTemplate = new RestTemplate();
MyObj myobj = restTemplate.getForObject(API_BASE_URL, MyObj.class);
```

Note that we can specify the return type of the API call with the second parameter (MyObj.class). Alternatively, if you are getting an array of objects back, we can write the following:

```
MyObj [ ] myobj = restTemplate.getForObject(API_BASE_URL, MyObj[ ].class);
```

Calling a Web Service Using Java

Spring Boot provides the **RestTemplate**, which is a Java based client for calling RESTful Web Services.

Endpoint - a URL that points to a Web Service.

To access a web service with RestTemplate, we must provide it the API endpoint:
`http://localhost:3000/hotels`

```
String response = restTemplate.getForObject(endpointUrl, String.class);
```

getForObject() - method that retrieves a JSON response and converts it to an object.
String response - Variable to hold the response.
endpointUrl - the URL of the endpoint to make the request.
String.class - The Data Type of the Object to return. This should match the variable that will hold it.

JSON with Java Objects

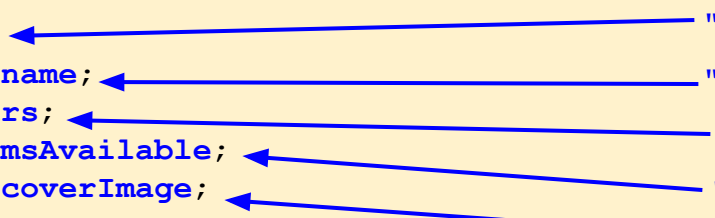
The RestTemplate getForObject() method can automatically populate a Java Object from the JSON. The Java Object must have member variables for the JSON properties we want mapped. **The data type and name must match!**

The Java Class must follow the **Java Bean** standard, so ***Getters and Setters must exist*** for each member variable the RestTemplate will populate, and there ***must be a no-argument constructor***.

Not all properties need to exist in the Java object.

```
public class Hotel {

    private int id;
    private String name;
    private int stars;
    private int roomsAvailable;
    private String coverImage;

    …
    getters/setters
    …
}
```

```
[
    {
        "id": 1,
        "name": "Greektown Detroit",
        "stars": 4,
        "roomsAvailable": 75,
        "coverImage": "greektown-detroit.webp"
    }
]
```



Parameters in Endpoints

A **Path Parameter** (Variable) is one that is passed as part of the resource path in the URL.

http://localhost:3000/hotels/4

http://localhost:3000/hotels/4/reviews

A **Query String Parameter** is one that is passed as a key/value pair as part of the Query portion (called the Query String) of the URL.

http://localhost:3000/hotels?stars=3

With API week coming up next week, here are some free public APIs you may enjoy sharing with your students:

HTTP://API.SHOUTCLOUD.IO/V1/SHOUT (good basic demo that switches anything you POST to it into ALL CAPS)
http://deckofcardsapi.com

Two "reflective" APIs that are good for testing HTTP requests of various types:
https://httpbin.org
https://httpstat.us/

Fun (Pokemon, Star Wars, Harry Potter)
https://pokeapi.co
https://swapi.dev/
https://wizard-world-api.herokuapp.com/swagger/index.html

Potentially useful for side projects or capstones
https://date.nager.at/Api (list public holidays for any country and year)
https://api.dictionaryapi.dev/api/v2/entries/en/<word>
https://quickchart.io/documentation/ (create charts and graphs from POSTed data)
https://goqr.me/api/doc/create-qr-code/ (create QR codes for any text or url)

All of those APIs are open and don't require any kind of key or authentication. A couple more free APIs with interesting data that require registering for a free key are:
https://www.nps.gov/subjects/developer/api-documentation.htm (National Park Service)
https://fdc.nal.usda.gov/api-guide.html (USDA food data)

Important Vocabulary

client: software that sends a request to a server to access a shared resource and processes the response returned from the server. (web browser, phone, pgadmin)

server: generic term referring to either software or hardware that processes request from clients (web server, email server, postgreSQL db)

Know the definition and structure of requests and responses

Know the difference between stateless and stateful

Know these request methods:

- GET - Retrieves information
- POST - Adds information
- PUT - Updates information
- DELETE - Deletes information

Know these response status code ranges:

- 100-199 - Informational responses
- 200-299 - Successful responses
- 300-399 - Redirection messages
- 400-499 - Client error responses
- 500-599 - Server error responses

URL: tells a client how to make a request to a server for a specific resource.

IP address- identifies a computer or device on a network (including the internet)

DNS: a service service allows for easy to remember names for humans to use (converts domain names to ip addresses)

port: a process-specific or an application-specific software construct serving as a communication endpoint

Important Vocabulary Continued

Know the difference between HTTP and HTTPS

Basic web page request work flow:

API (Application Programming Interface): a way for software components to talk to each other.

Web Service/Web API: an API that allows for applications to communicate with each other across the internet

REST(REpresentational State Transfer): A software architectural pattern that defines a set of constraints and standards for how 2 applications on a network should communicate.

Know the rules of REST:

1. Uniform Interface (URI, URL)
2. Stateless
3. Cacheable
4. Client-Server based
5. Layered System

JSON: (JavaScript Object Notation) is a messaging format that is commonly used by RESTful Web Services.

Time to write GET requests, try out the toolset, and code!