HTTP Web Services GET

Module 2: 11

Week 7 Overview

Monday	Tuesday	Wednesday	Thursday	Friday
HTTP & Consuming APIs Part 1	Consuming APIs Part 2	Server Side APIs Part 1	Server Side APIs Part 2	Review

Today’s Objectives

1. **How the Internet Works**
    a. Clients and Servers
    b. Request/Response
    c. Stateless
    d. URL
    e. Domain Names
    f. Internet Protocol (IP)
    g. Domain Name System (DNS)
    h. Ports
    i. HyperText Transfer Protocol (HTTP)

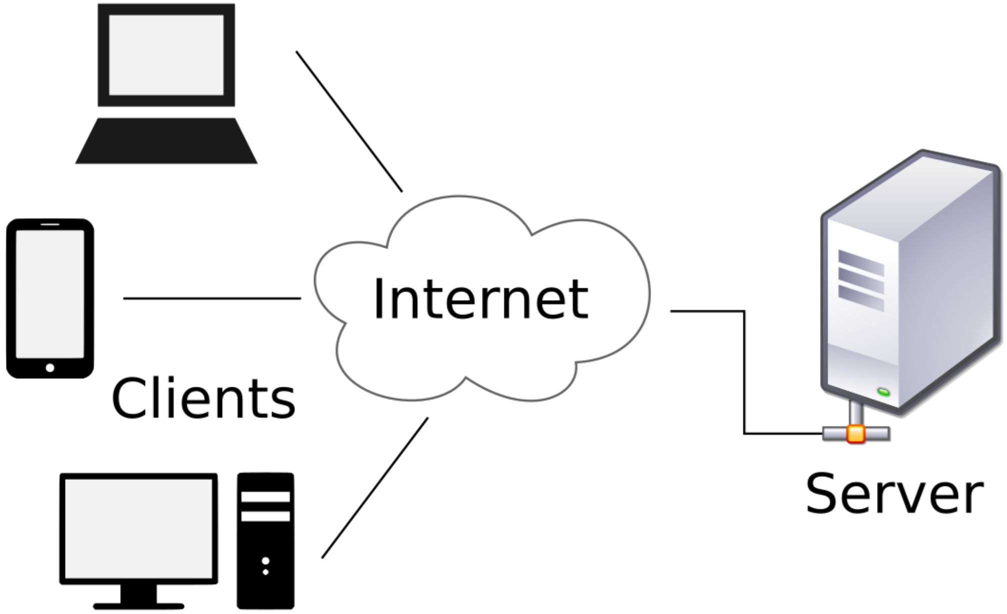
2. **APIs and Web Services**
    a. REST
    b. RESTful Web services
    c. JSON (JavaScript Object Notation)
3. **Consuming RESTful Web Services (API) with GET in Java**
    a. Endpoints
    b. RestTemplate
    c. Converting JSON to Java Objects
    d. Endpoint Parameters
        i. Path Parameters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

Clients and Servers

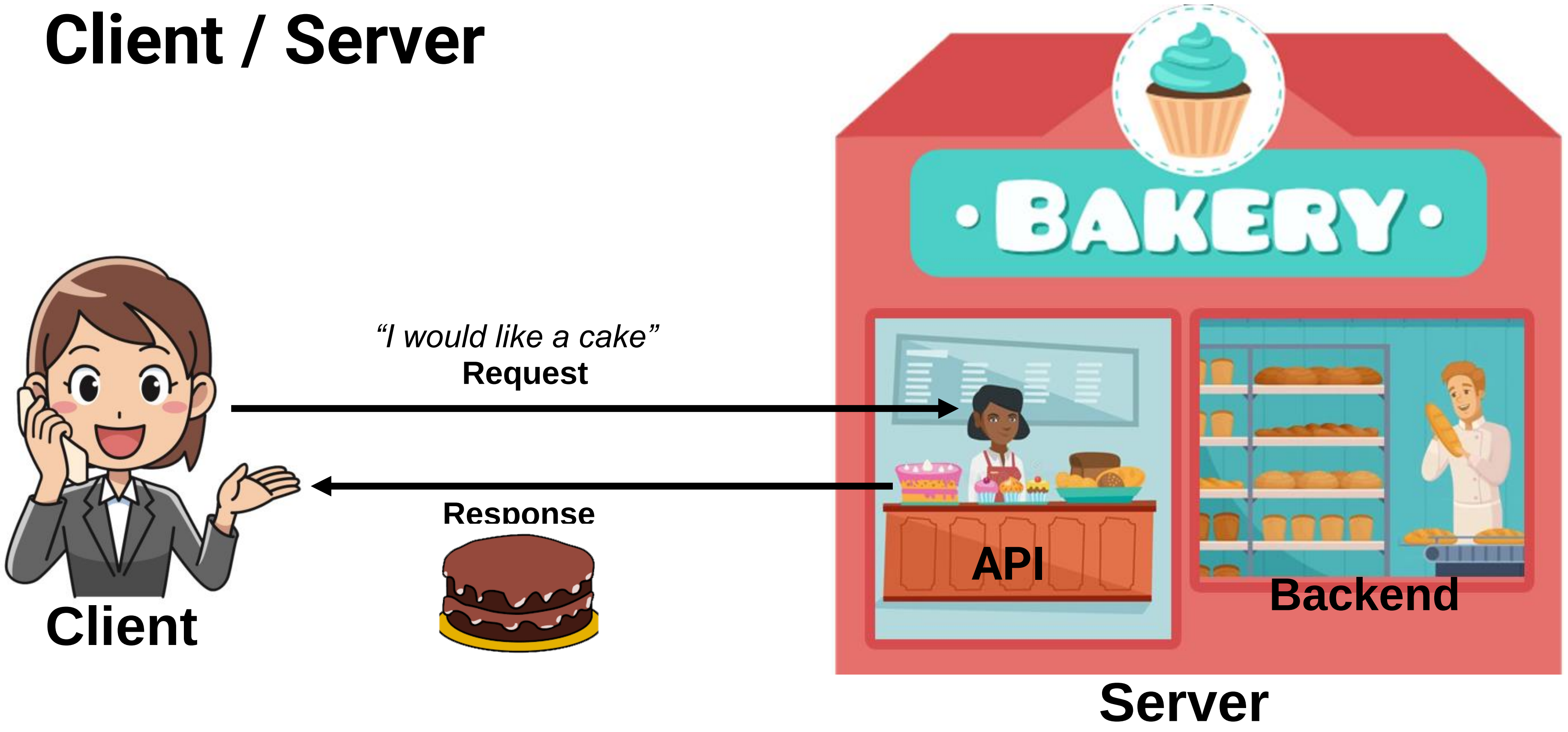
Server is a generic term referring to either software or hardware that processes request from **clients**. A server offers shared **resources** that can be requested for use by a **client**.

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

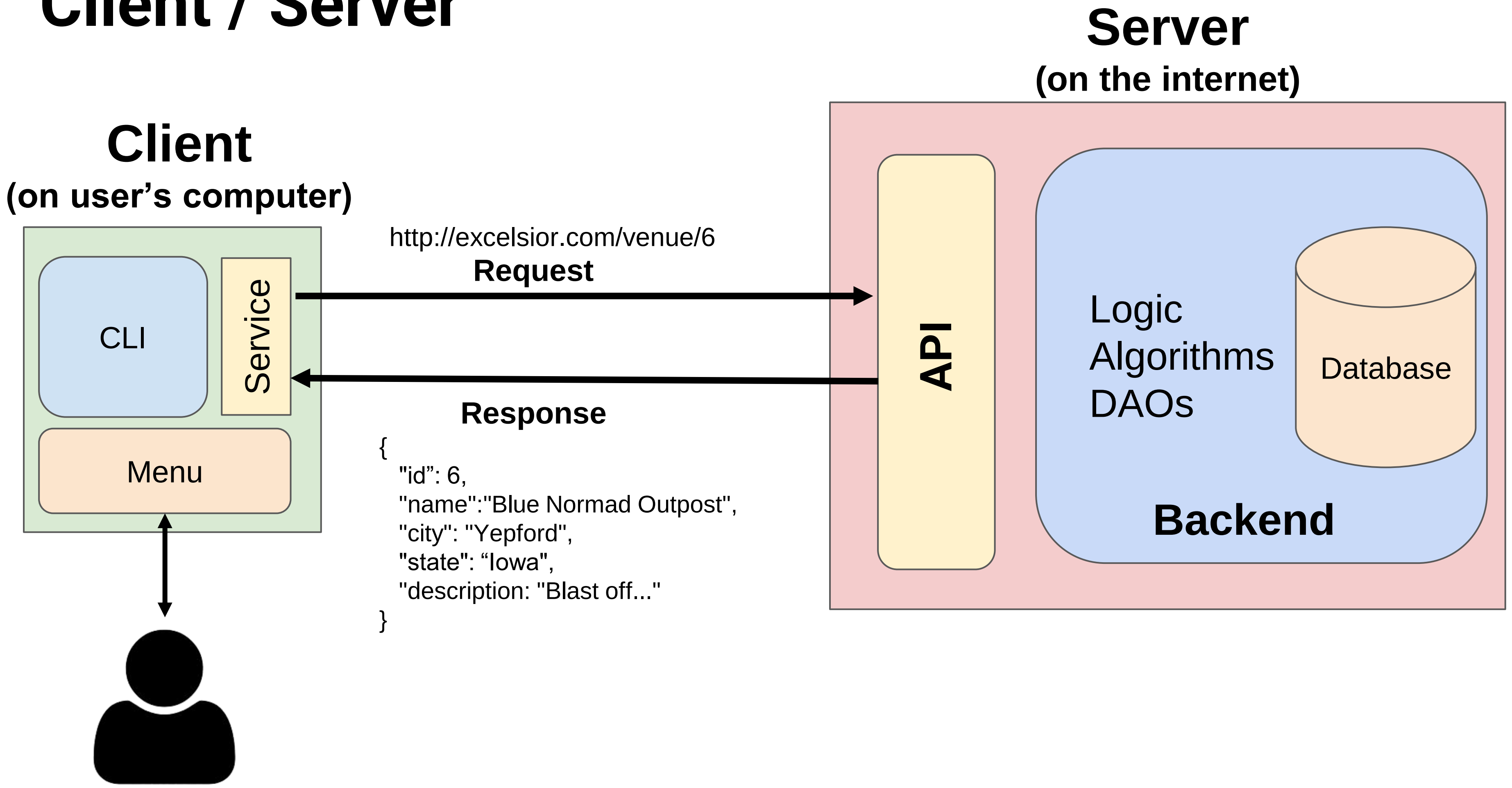
Client	Server
Web Browser	Web Server
Smart TV	Netflix's computers
Phone	Messaging Server
Email App	Email Server
DbVisualizer	PostgreSQL



Client / Server



Client / Server



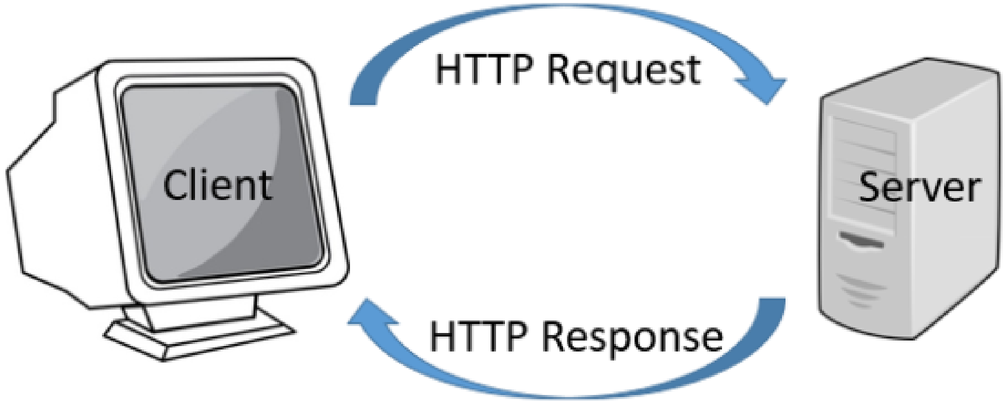
```
{
   "id": 6,
   "name":"Blue Normad Outpost",
   "city": "Yepford",
   "state": “Iowa",
   "description: "Blast off..."
}
```

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.

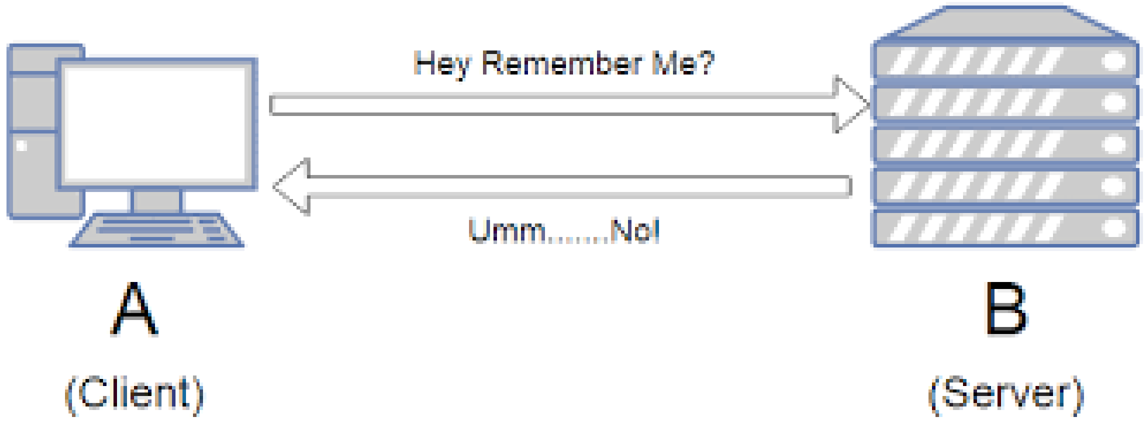
In HTTP, all communication between a client and server is via stateless request/response pair.



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

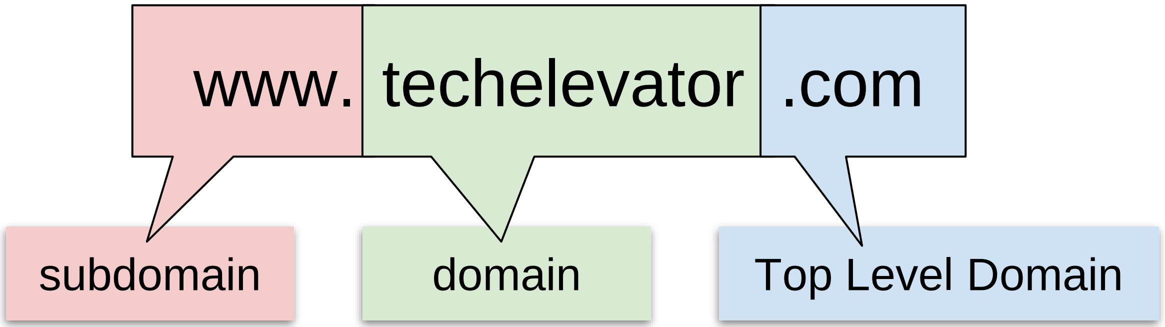
http://	**Protocol** (the language being spoken between client and server)
www.techelevator.com	**Domain** (the address of the computer the server is on)
:80	**Port** (identifies the server on the hosting computer)
/events/current	**Path to resource** (what is being requested)
?	**Query** (indicates 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, …

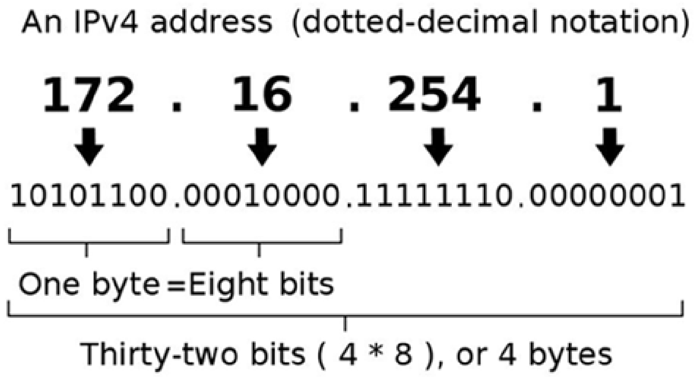
The hierarchy of a domain name is read right-to-left.



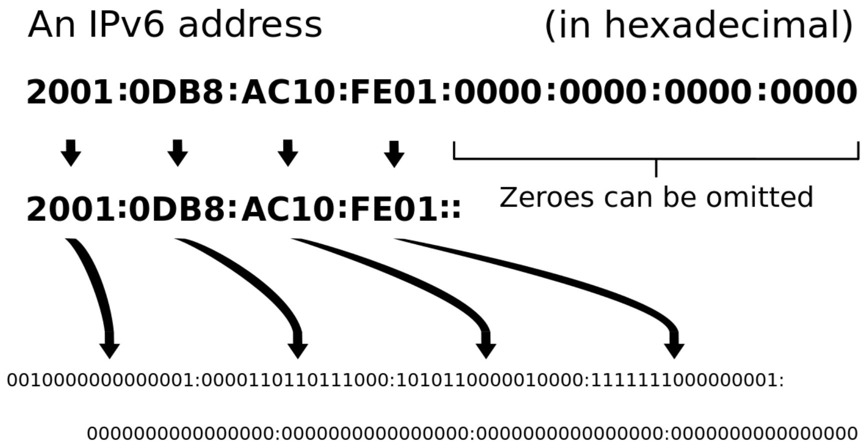
IP (Internet Protocol) Addresses

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.



An IPv4 address (dotted-decimal notation)
172 . 16 . 254 . 1
10101100.00010000.11111110.00000001
One byte=Eight bits
Thirty-two bits (4 * 8), or 4 bytes

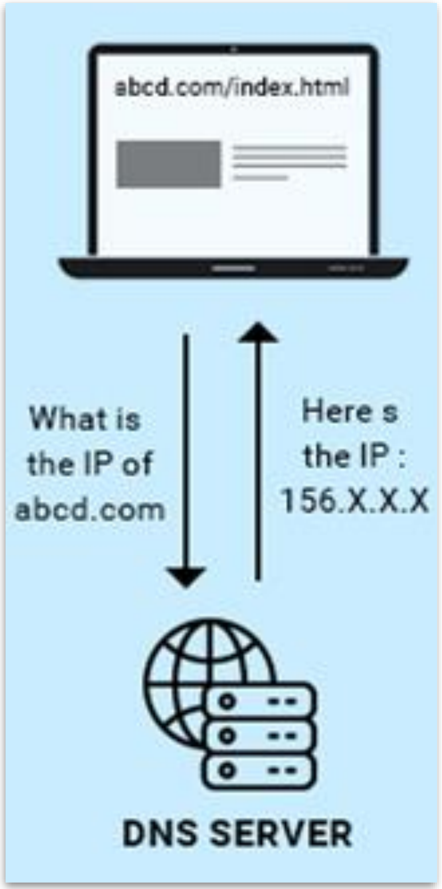


DNS (Domain Name System)

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.

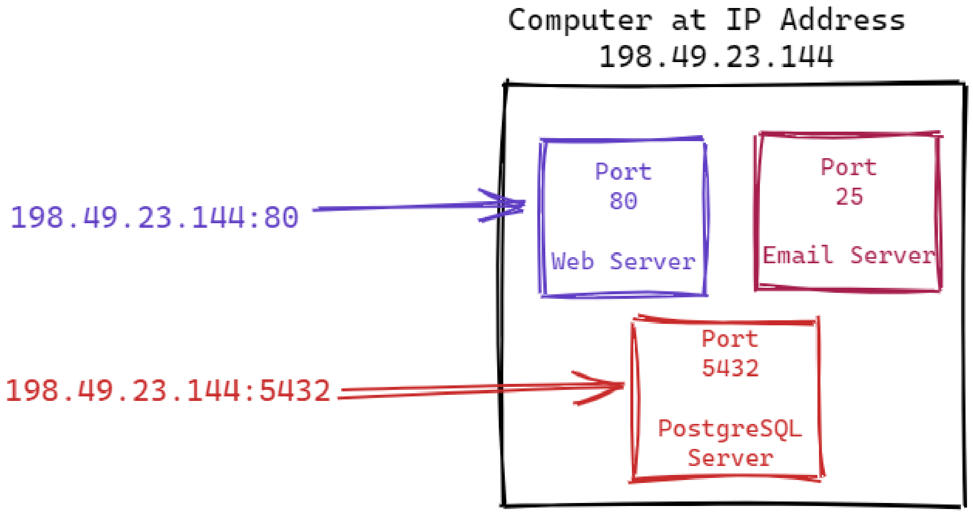
IP Address is like the street address of a building, and the port number would be equivalent to an apartment number in that building.

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)

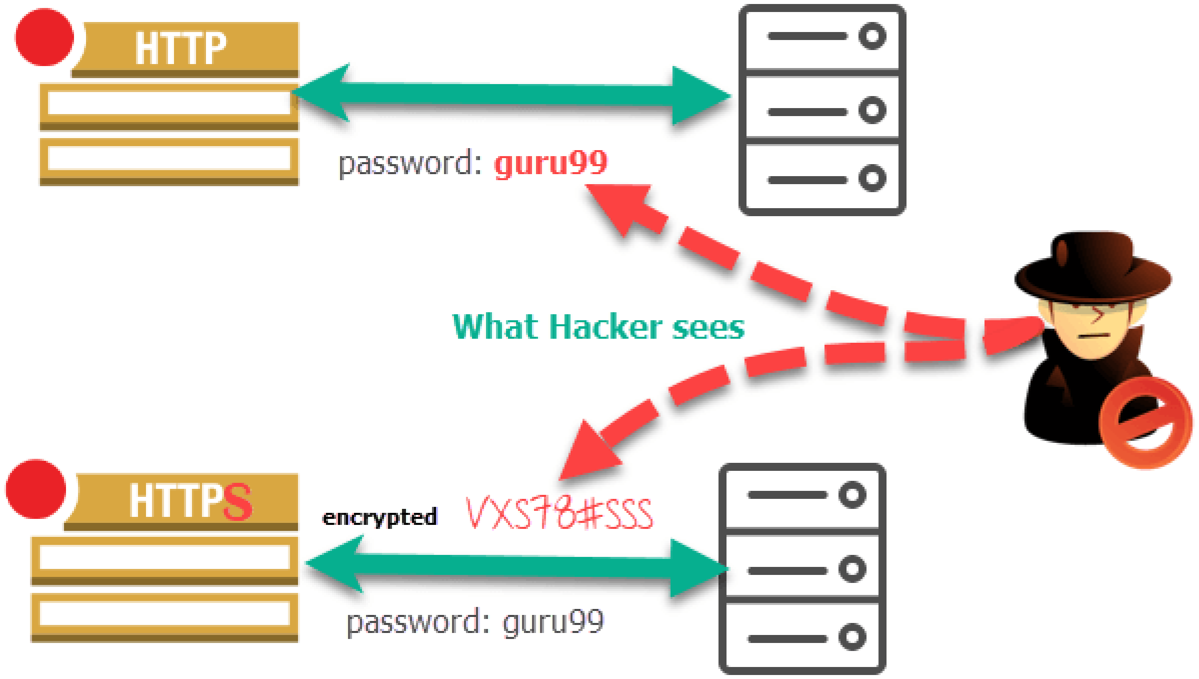
HTTP vs HTTPS

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

Like mailing a letter on a postcard.

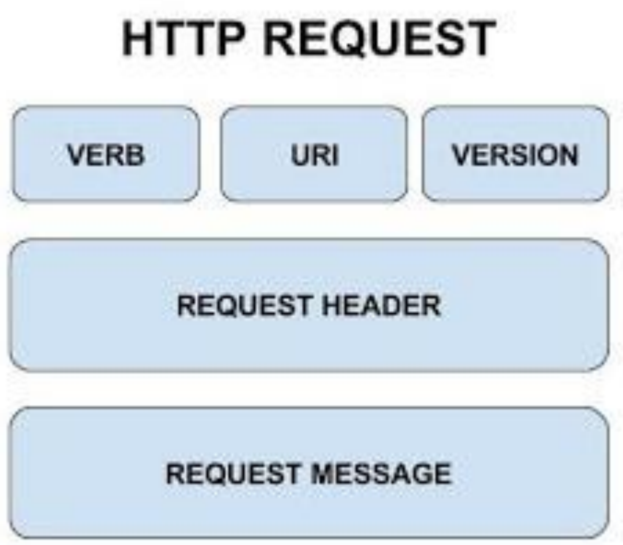
HTTPS encrypts data using TLS (Transport Layer Security), so data is not readable.

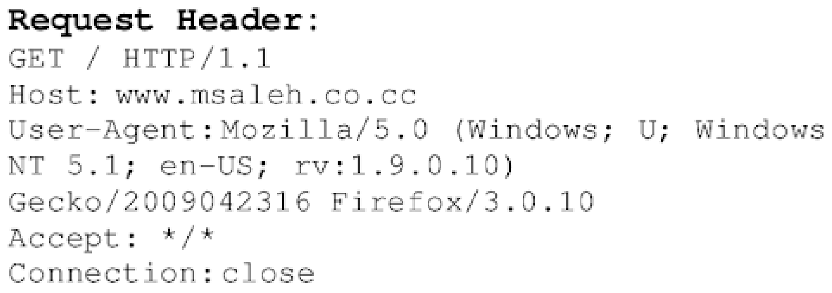
Like mailing a letter in a sealed envelope.



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.



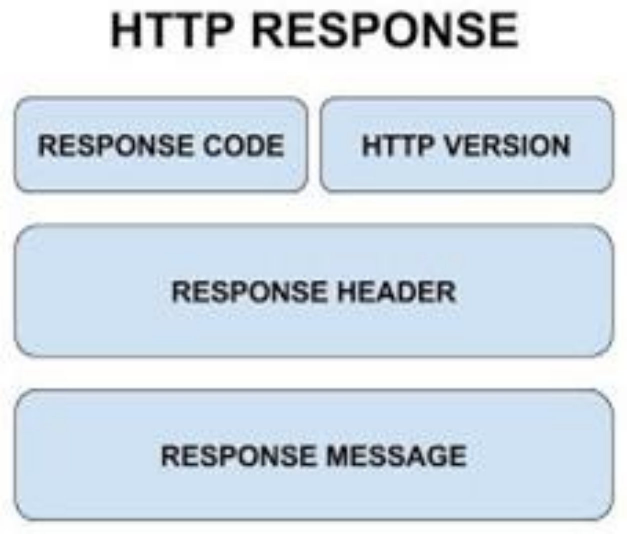


Request Header:

```
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
```

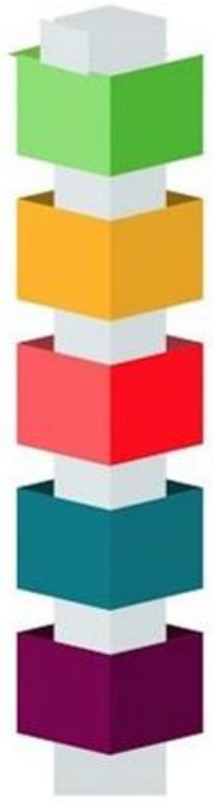
HTTP Response

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.



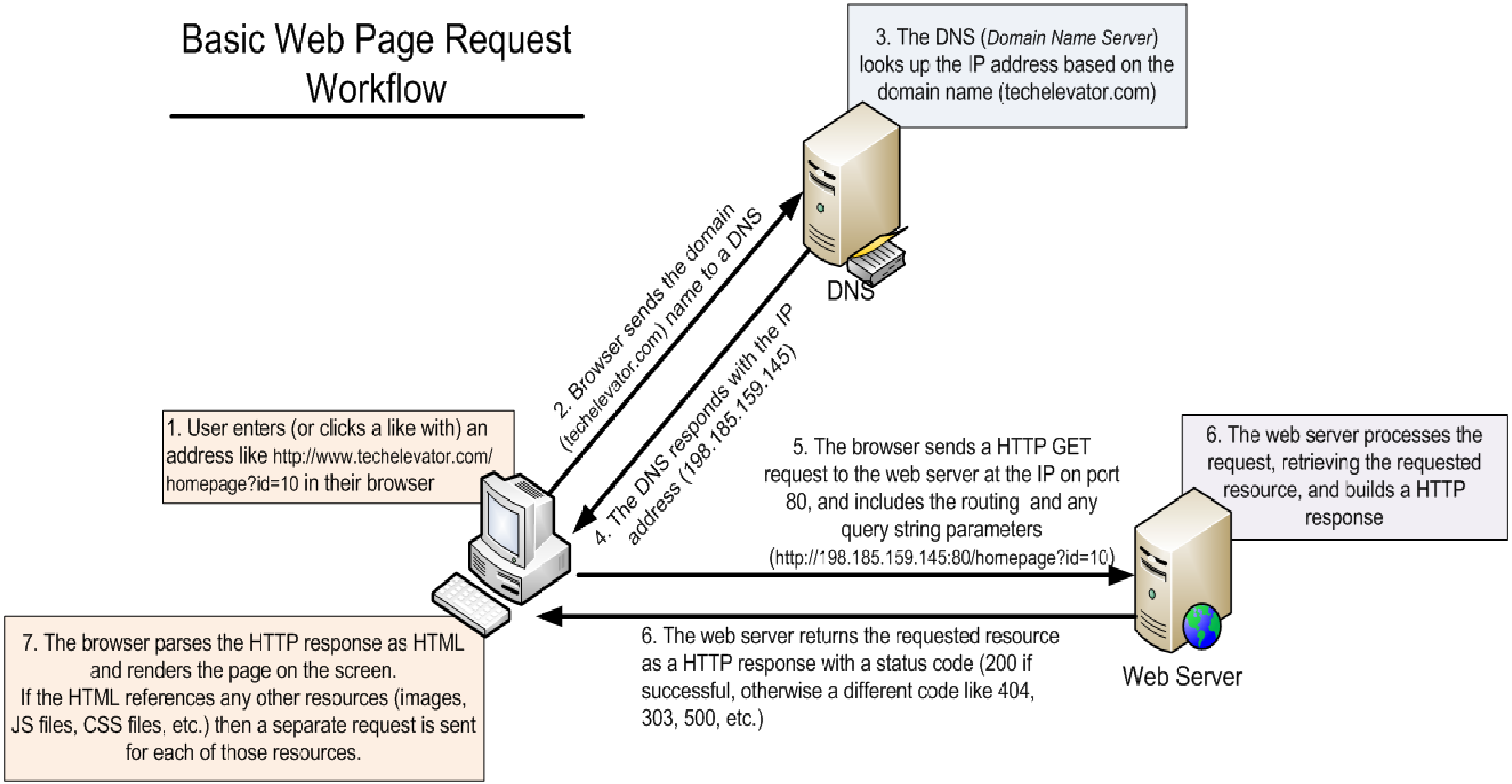
```
Response Header:
HTTP/1.1 200 OK
Date: Sat, 09 May 2009 12:27:54 GMT
Server: Apache/2.2.11 (Unix)
Last-Modified: Thu, 12 Feb 2009 15:29:42
GMT
Etag: "c3b-462ba63a46580"-gzip
Cache-Control: max-age=1200, private,
proxy-revalidate, must-revalidate
Expires: Sat, 09 May 2009 12:47:54 GMT
Accept-Ranges: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

Basic Web Page Request Workflow



1. User enters (or clicks a like with) an address like http://www.techelevator.com/homepage?id=10 in their browser

2. Browser sends the domain (techelevator.com) name to a DNS

3. The DNS (*Domain Name Server*) looks up the IP address based on the domain name (techelevator.com)

4. The DNS responds with the IP address (198.185.159.145)

5. The browser sends a HTTP GET request to the web server at the IP on port 80, and includes the routing and any query string parameters (http://198.185.159.145:80/homepage?id=10)

6. The web server processes the request, retrieving the requested resource, and builds a HTTP response

6. The web server returns the requested resource as a HTTP response with a status code (200 if successful, otherwise a different code like 404, 303, 500, etc.)

7. The browser parses the HTTP response as HTML and renders the page on the screen. If the HTML references any other resources (images, JS files, CSS files, etc.) then a separate request is sent for each of those resources.

API

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

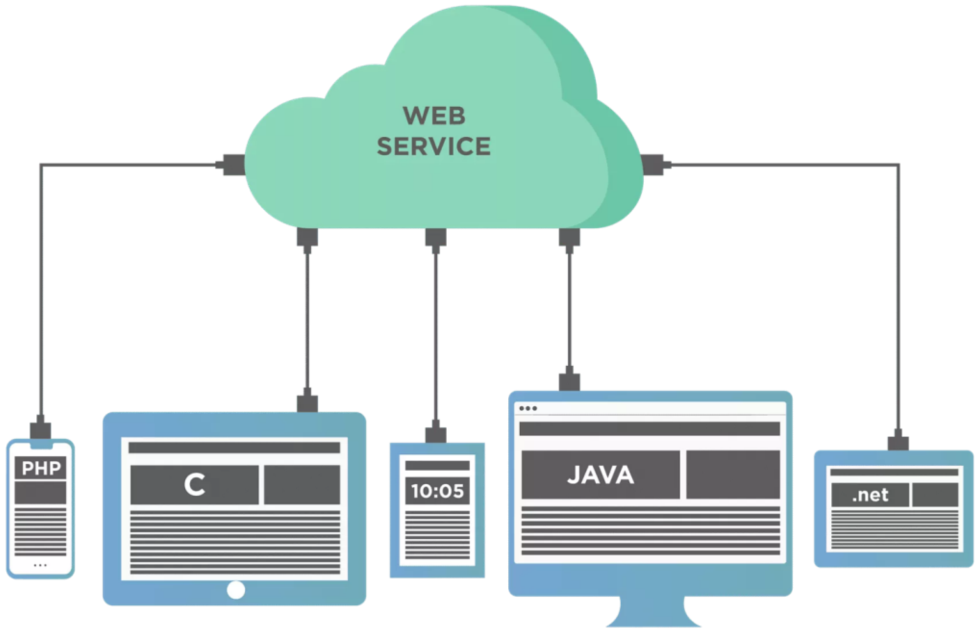
An API is a way for software components to talk to each other. The methods and properties in our classes that are exposed as public are the API of the object.

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*

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
    }
  ]
}
```

- 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.

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);
```

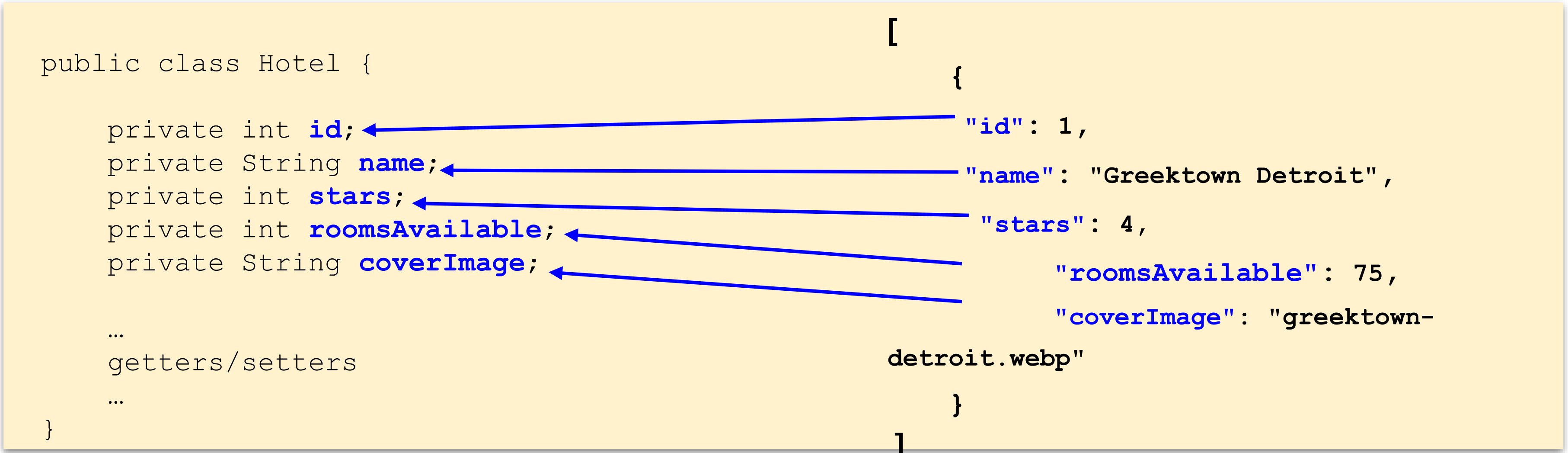
getForObject() - method that retrieves a JSON response and converts it to an object.
String response - Variable to hold the response.
endpointUrl - the URL of the endpoint to make the request.
String.class - The Data Type of the Object to return. This should match the variable that will hold it.

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

Tools:

- npm, Jason Server, Postman

NPM

NPM is a package manager for JavaScript code. It's one of the most widely used package managers in software development and is often touted as having the largest repository.
Its name is short for Node.js Package Manager. Node.js is a runtime environment for running complex applications in JavaScript without a browser.

NPM stores the information of the packages you use for a project in a file named package.json. This file also contains other information about the project like its name and version.

To install all dependencies:

```
>npm install
```

NPM will begin the process to make a development server available for your application:

```
>npm start
```

Get the version of npm

```
>npm –version
```

JASON Server

JSON Server is a Node Module that you can use to create demo REST JASON webservice. All you need is a JSON file for sample data.

If you wish to have a mockup Rest Web service in place to get the demo data for you, then JASON Server is the tool you are looking for.

Postman

Postman is a software tool that helps developers test Web APIs.

Postman is an effective testing tool because it allows you to quickly create API requests, input request data, and read response data.