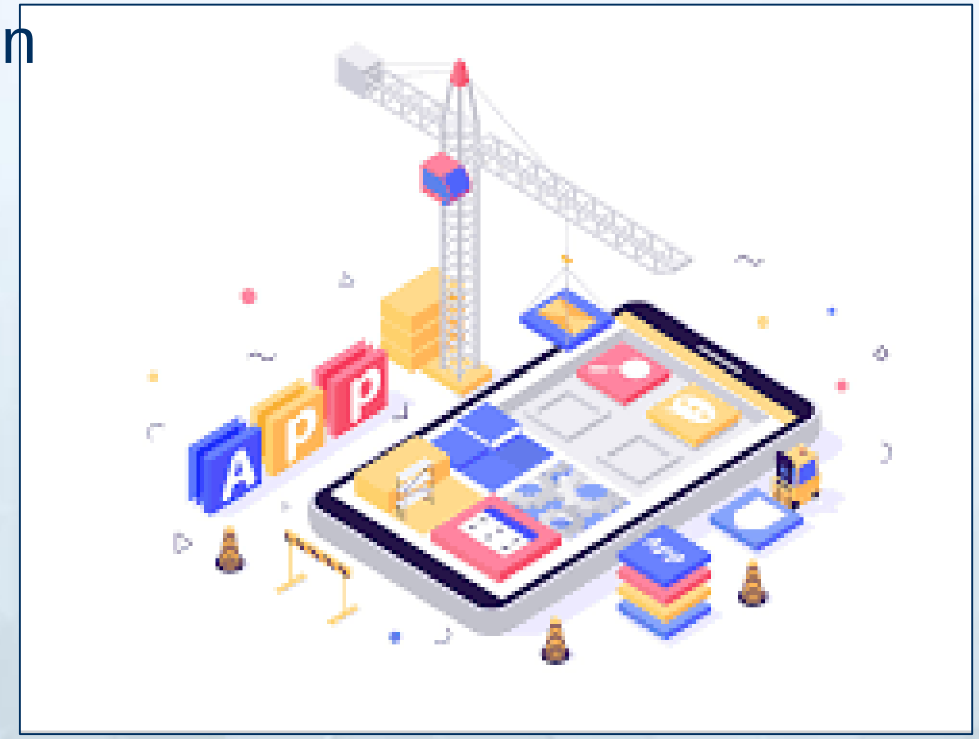
Protocols and webservices

Session Objective

- Multitier Architecture
- Http Protocol
- Webservices
- Restful webservices

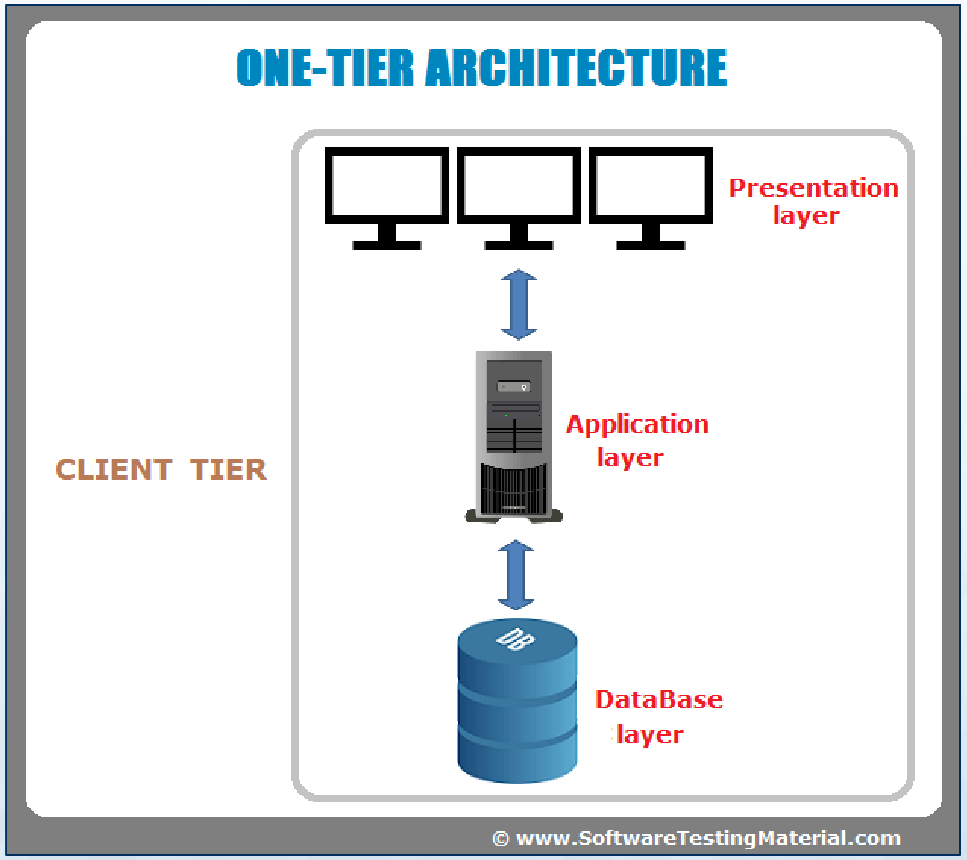
Application Development

- Application development is the process of designing, building, and implementing software applications.
- Application development defines the process of how the application is made, and generally follows a standard methodology.



One tier architecture

- One tier architecture contains the Presentation, Business and Data Access layers in a single software package.
- The data is stored in the local system or a shared drive.

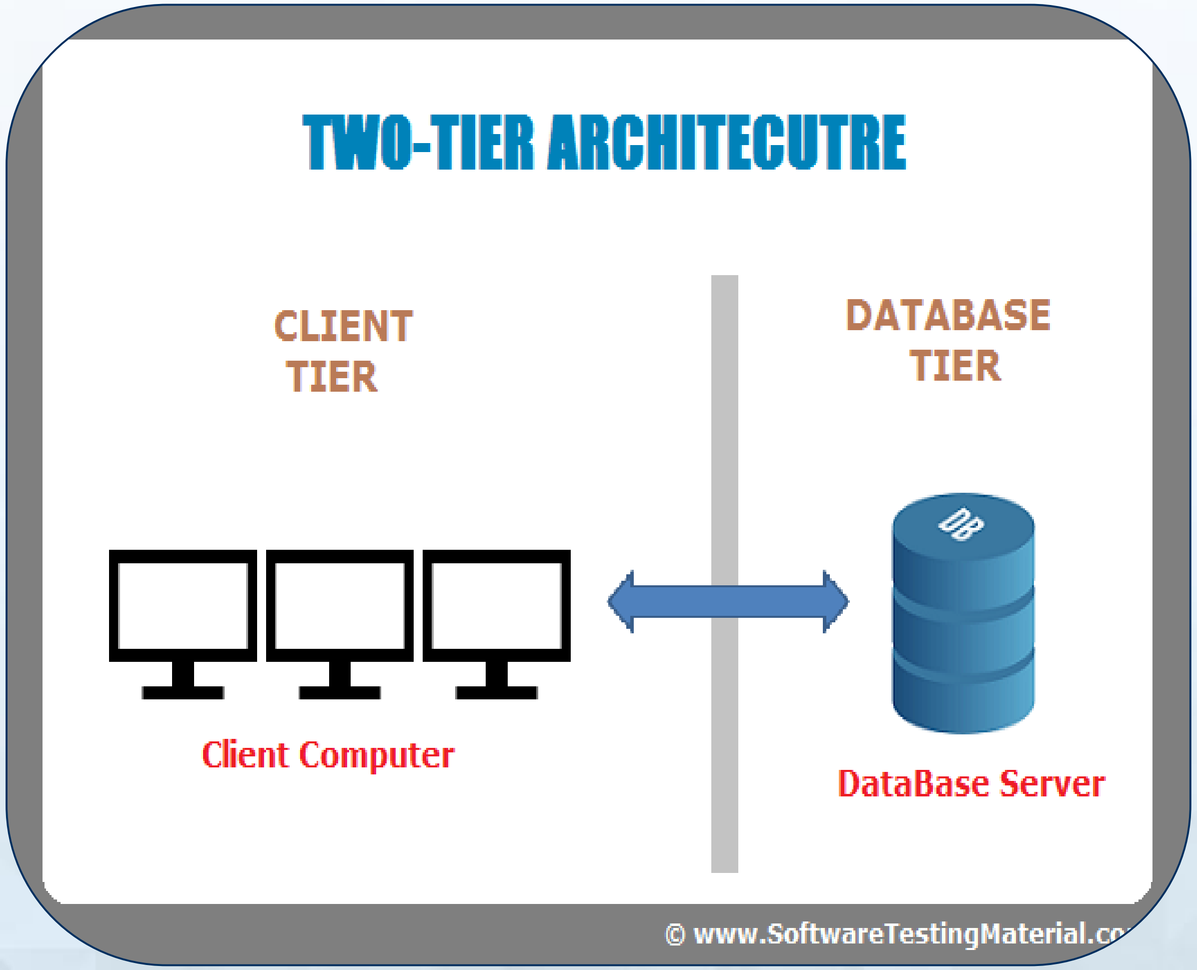


Two – Tier Architecture

The Two-tier architecture is divided into two parts:

1. Client Application (Client Tier)
2. Database (Data Tier)

Client system handles both Presentation and Application layers and Server system handles Database layer. It is also known as client server application.

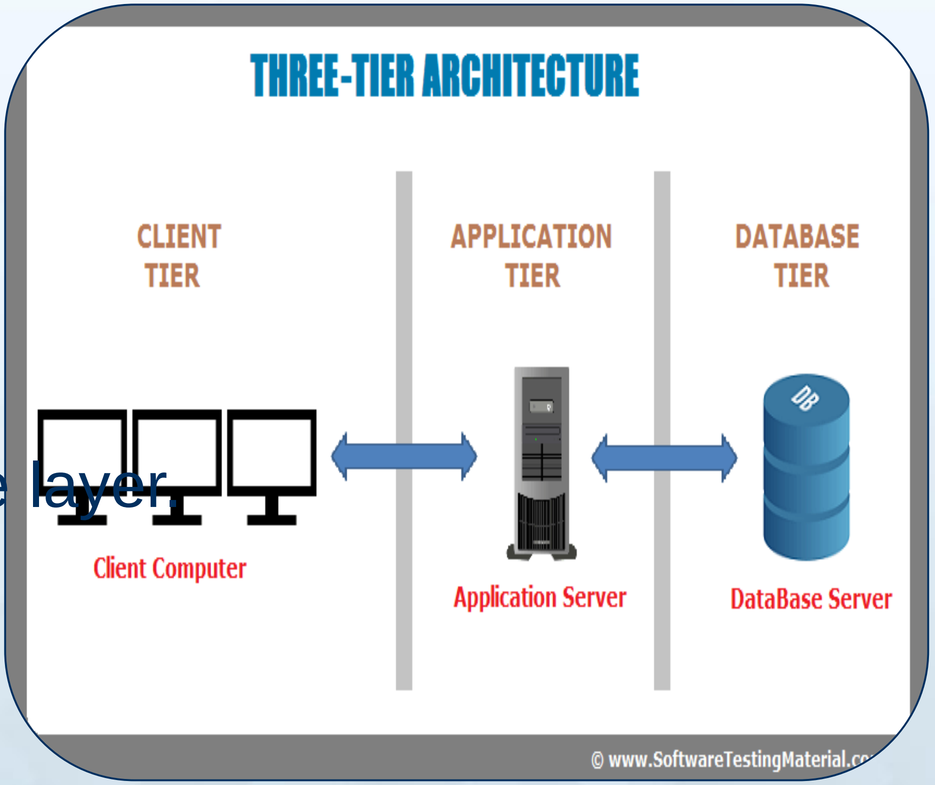


Three-Tier Architecture:

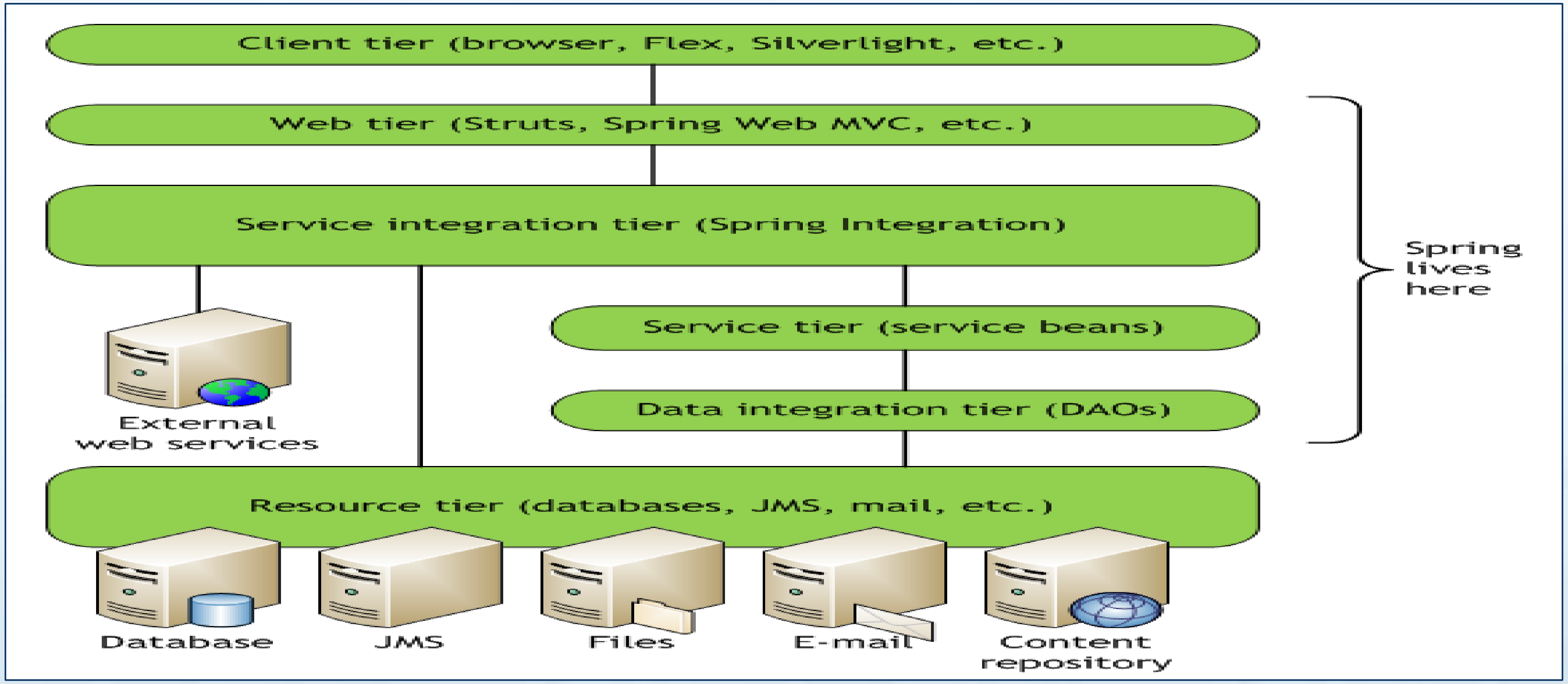
The Three-tier architecture is divided into three parts:

1. Presentation layer (Client Tier)
2. Application layer (Business Tier)
2. Database layer (Data Tier)

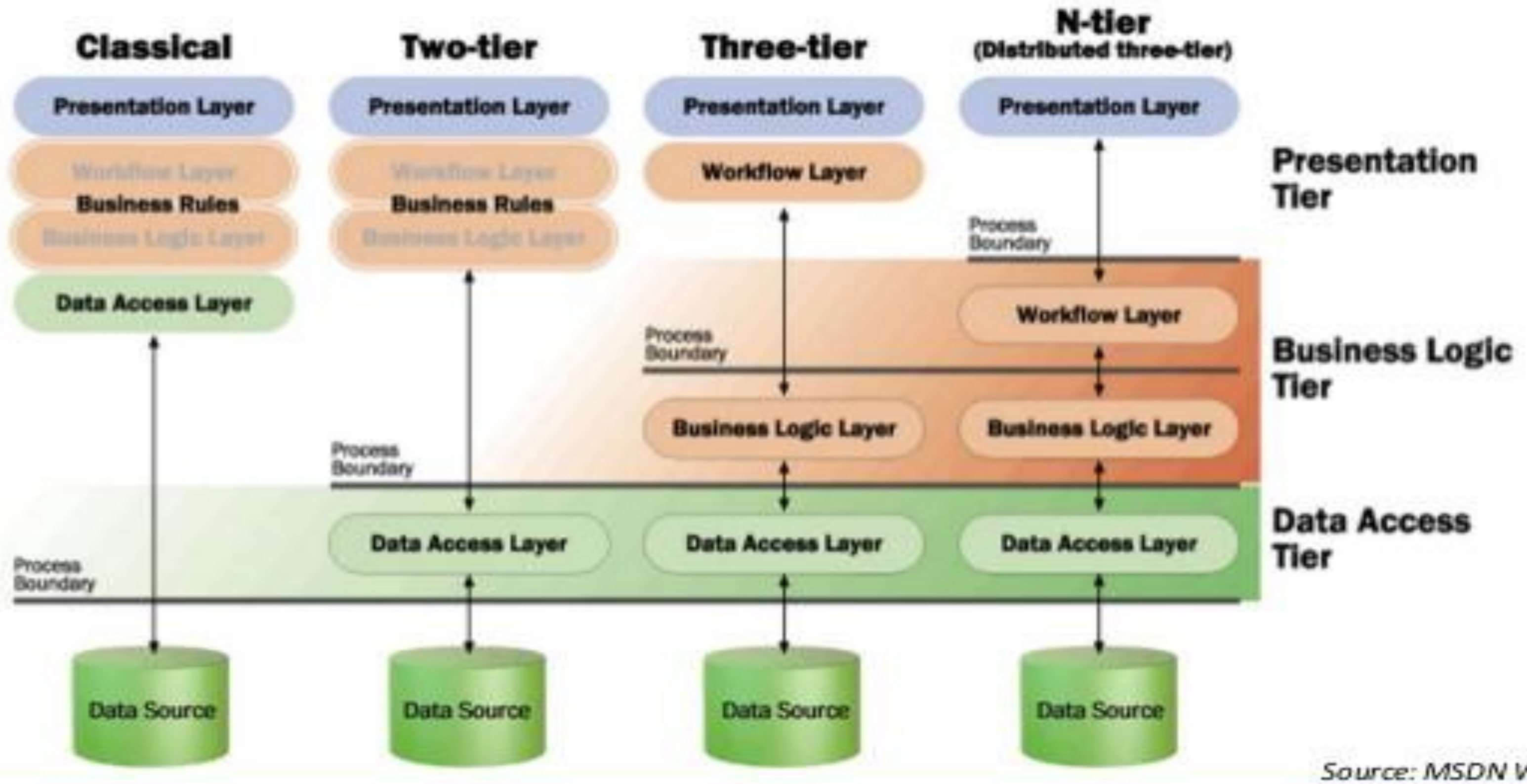
➔Client system handles Presentation layer
➔Application server handles Application layer
➔ Database Server system handles Database layer.



N-tier Architecture


Client tier (browser, Flex, Silverlight, etc.)
Web tier (Struts, Spring Web MVC, etc.)
Service integration tier (Spring Integration)
Service tier (service beans)
Data integration tier (DAOs)
Resource tier (databases, JMS, mail, etc.)
External web services
Database
JMS
Files
E-mail
Content repository
Spring lives here

Application Development - N-tier

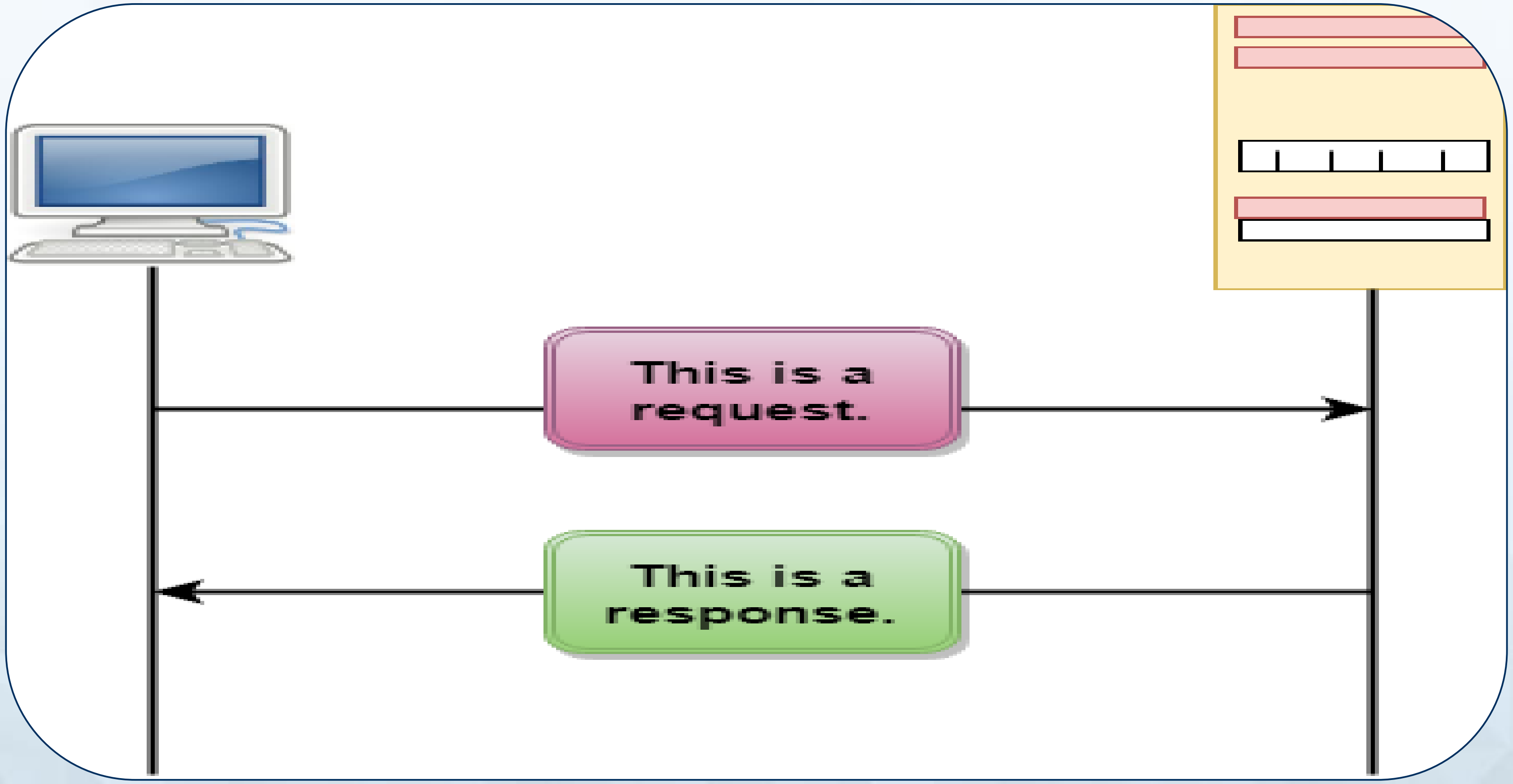


Source: MSDN Website, 2001

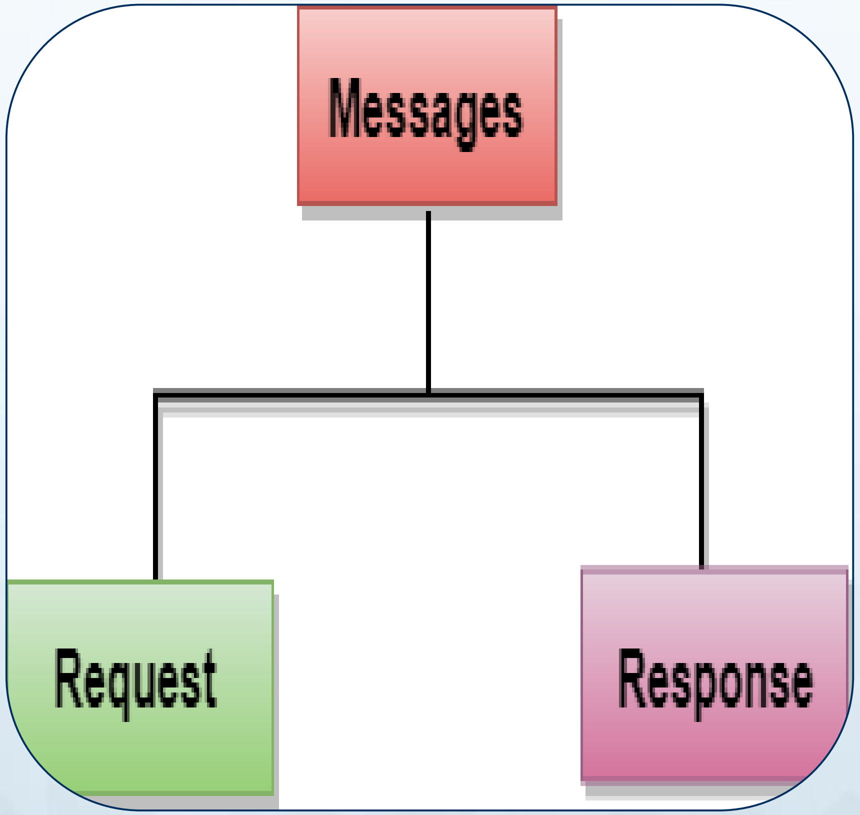
HTTP

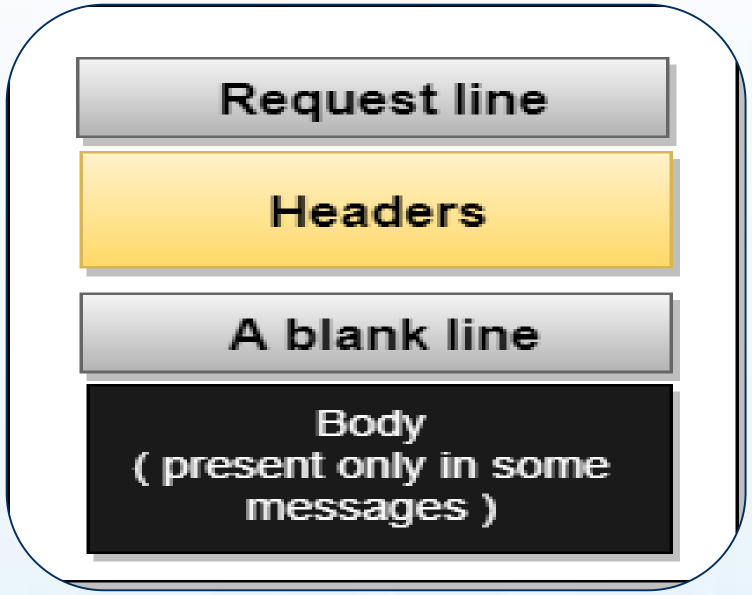
- HTTP stands for **Hypertext Transfer Protocol**.
- It is a protocol used to access the data on the World Wide Web.
- The HTTP protocol can be used to transfer the data in the form of plain text, hypertext, audio, video, etc..
- HTTP is used to carry the data in the form of MIME-like format.

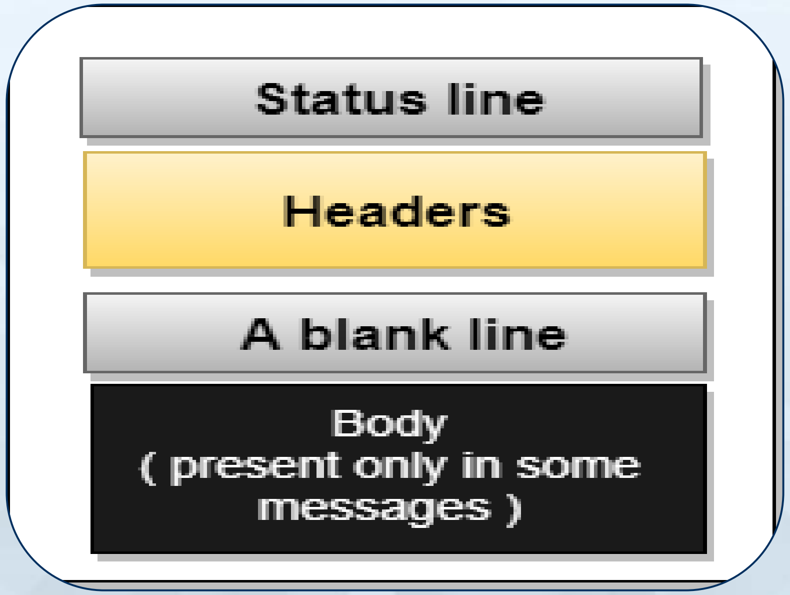
Http Transaction



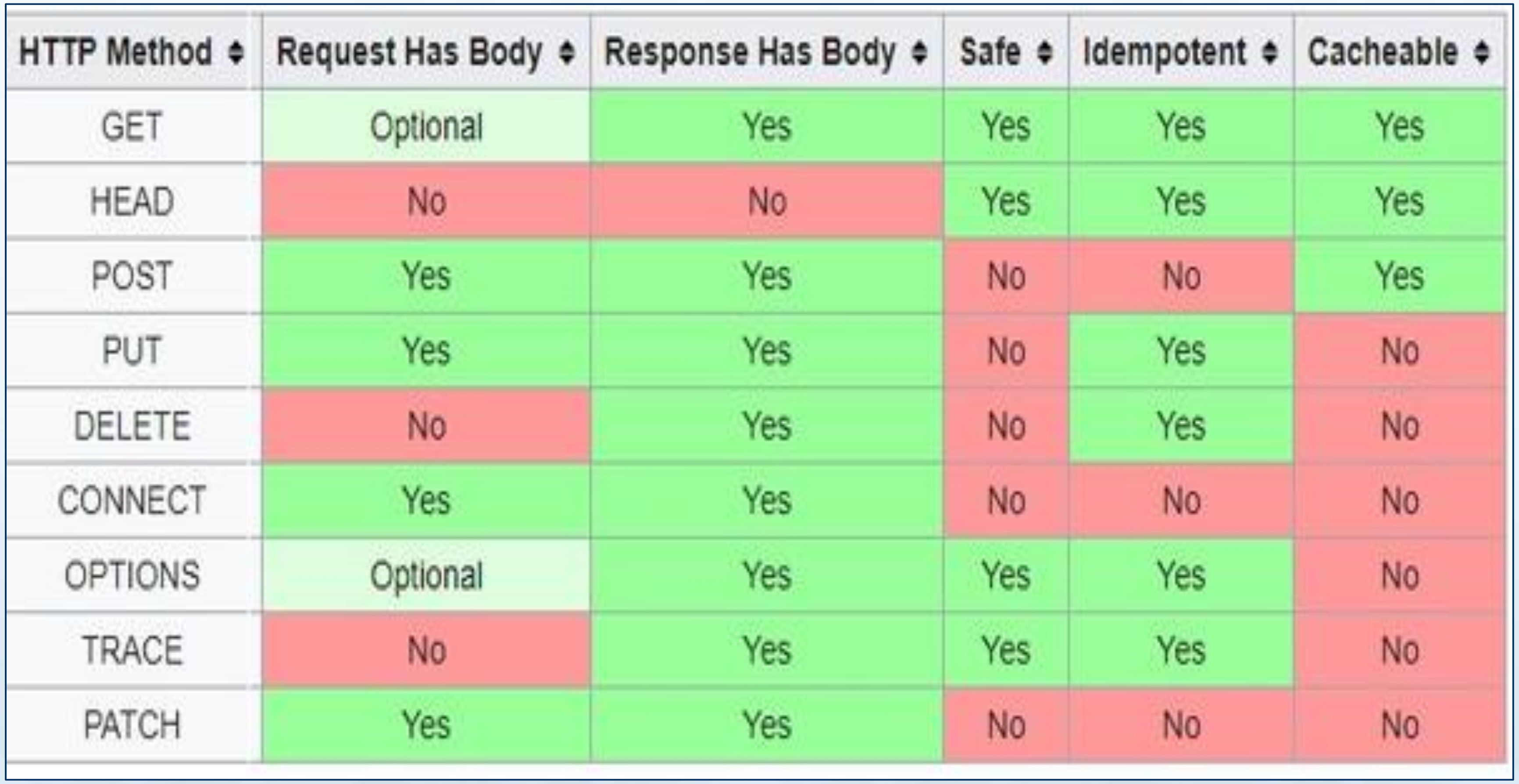
Http Messages





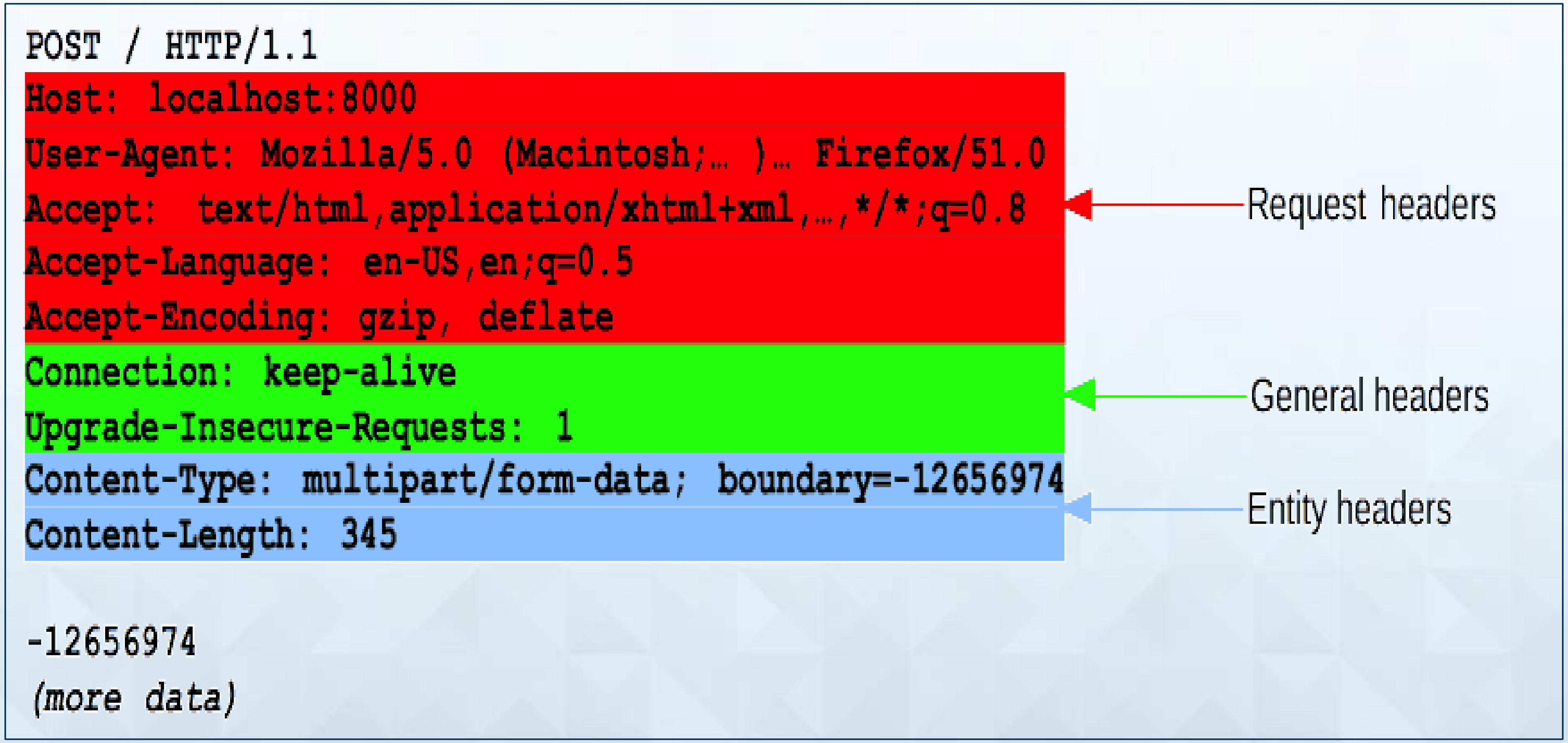


Http Protocol-method



HTTP Method	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET	Optional	Yes	Yes	Yes	Yes
HEAD	No	No	Yes	Yes	Yes
POST	Yes	Yes	No	No	Yes
PUT	Yes	Yes	No	Yes	No
DELETE	No	Yes	No	Yes	No
CONNECT	Yes	Yes	No	No	No
OPTIONS	Optional	Yes	Yes	Yes	No
TRACE	No	Yes	Yes	Yes	No
PATCH	Yes	Yes	No	No	No

Request Header

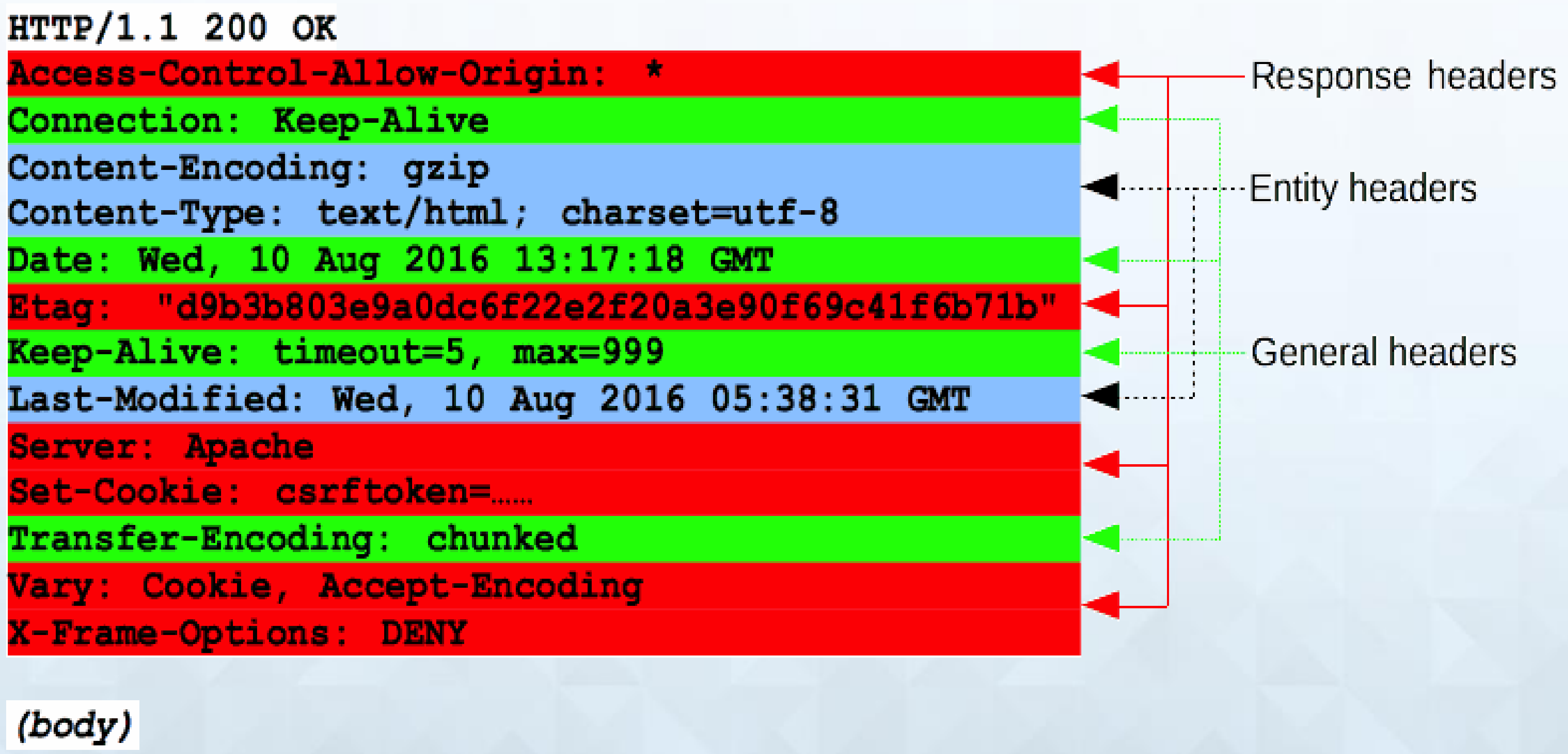


```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Request Body

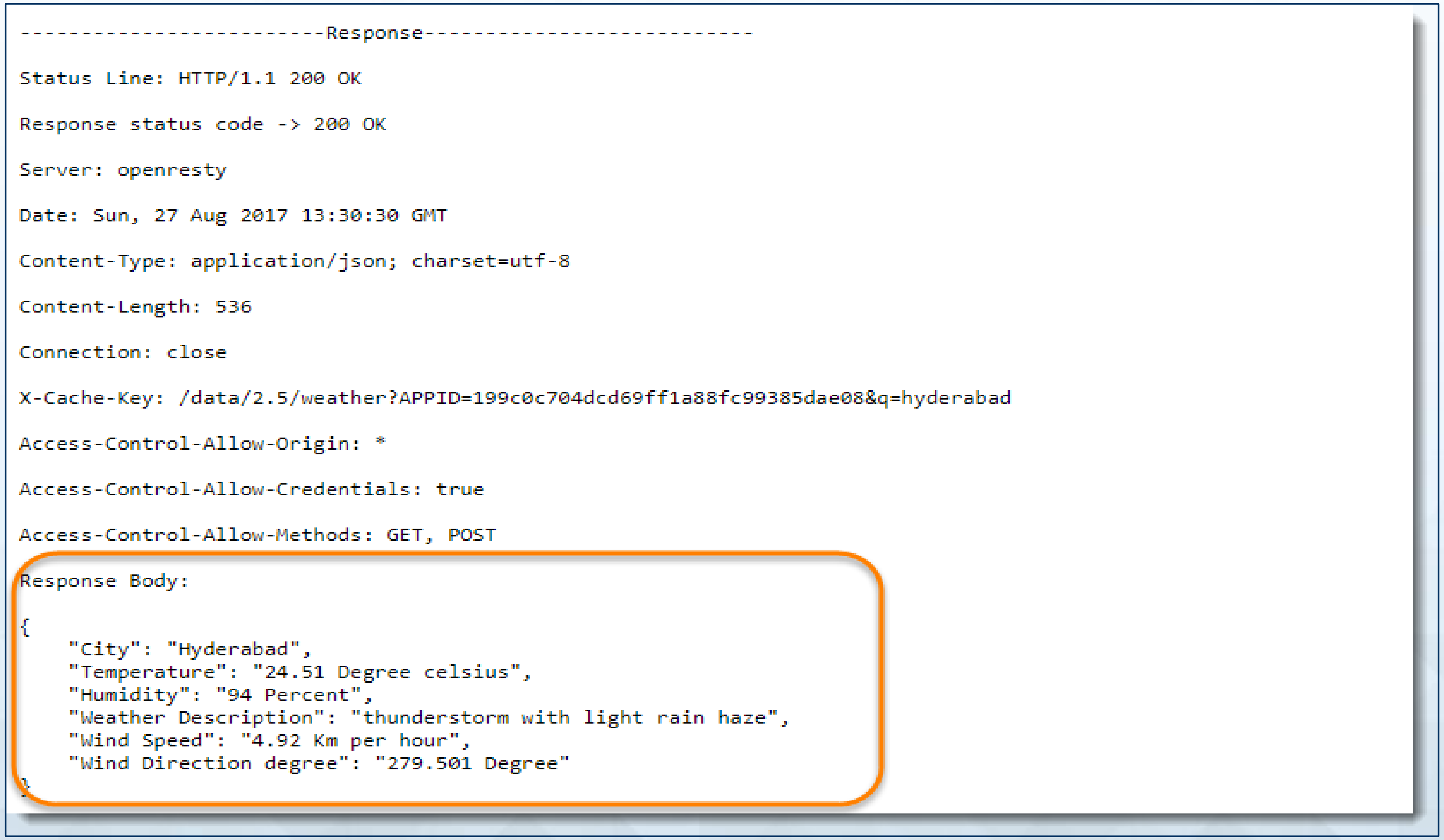
- The request fetching the resources will have the request body.
- Bodies can be broadly divided into two categories:
- Single-resource bodies, consisting of one single file, defined by the two headers: Content-Type and Content-Length.
- Multiple-resource bodies, consisting of a multipart body, each containing a different bit of information. This is typically associated with HTML Forms.

Response Header



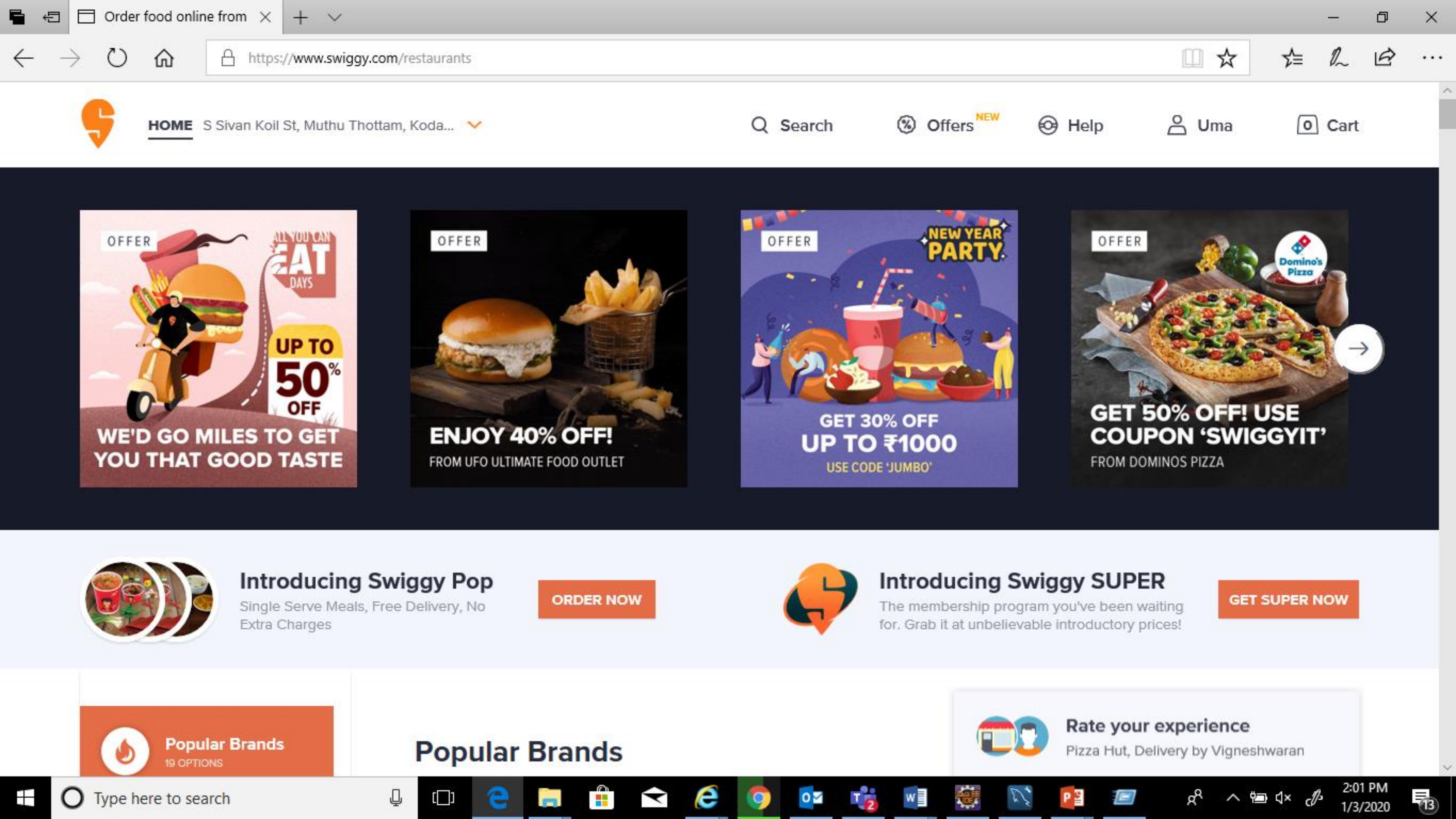
```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 10 Aug 2016 13:17:18 GMT
Etag: "d9b3b803e9a0dc6f22e2f20a3e90f69c41f6b71b"
Keep-Alive: timeout=5, max=999
Last-Modified: Wed, 10 Aug 2016 05:38:31 GMT
Server: Apache
Set-Cookie: csrftoken=......
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
X-Frame-Options: DENY

(body)
```

Response Body



```
-------------------------Response---------------------------

Status Line: HTTP/1.1 200 OK

Response status code -> 200 OK

Server: openresty

Date: Sun, 27 Aug 2017 13:30:30 GMT

Content-Type: application/json; charset=utf-8

Content-Length: 536

Connection: close

X-Cache-Key: /data/2.5/weather?APPID=199c0c704dcd69ff1a88fc99385dae08&q=hyderabad

Access-Control-Allow-Origin: *

Access-Control-Allow-Credentials: true

Access-Control-Allow-Methods: GET, POST

Response Body:

{
    "City": "Hyderabad",
    "Temperature": "24.51 Degree celsius",
    "Humidity": "94 Percent",
    "Weather Description": "thunderstorm with light rain haze",
    "Wind Speed": "4.92 Km per hour",
    "Wind Direction degree": "279.501 Degree"
}
```

Webservice



Order food online from

https://www.swiggy.com/restaurants

HOME S Sivan Koil St, Muthu Thottam, Koda...

Search Offers NEW Help Uma 0 Cart

OFFER
ALL YOU CAN EAT DAYS
UP TO 50% OFF
WE'D GO MILES TO GET YOU THAT GOOD TASTE

OFFER
ENJOY 40% OFF!
FROM UFO ULTIMATE FOOD OUTLET

OFFER
NEW YEAR PARTY
GET 30% OFF UP TO ₹1000
USE CODE 'JUMBO'

OFFER
GET 50% OFF! USE COUPON 'SWIGGYIT'
FROM DOMINOS PIZZA

Introducing Swiggy Pop
Single Serve Meals, Free Delivery, No Extra Charges
ORDER NOW

Introducing Swiggy SUPER
The membership program you've been waiting for. Grab it at unbelievable introductory prices!
GET SUPER NOW

Popular Brands
19 OPTIONS

Popular Brands

Rate your experience
Pizza Hut, Delivery by Vigneshwaran

Type here to search

2:01 PM
1/3/2020

Web Service

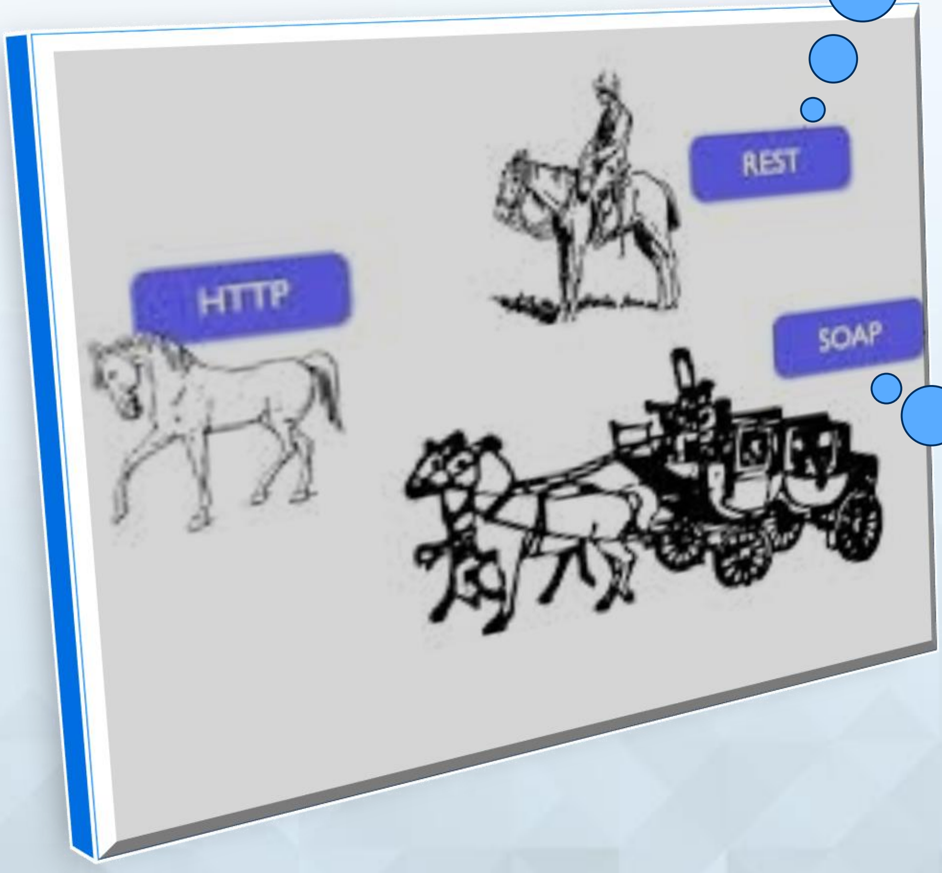
- Web service is a technology to communicate one programming language with another.
- Web service provides a way to achieve interoperability.
- For example:

Java programming language can interact with PHP and .Net by using web services.

Types of Web Services

- There are mainly two types of web services
  - SOAP web services.
  - RESTful web services.

Rides directly on HTTP. Plain and simple.



The coach is the soap. It wraps the data to be carried.

SOAP Vs REST

No.	SOAP	REST
1)	SOAP is a protocol.	REST is an architectural style.
2)	SOAP stands for Simple Object Access Protocol.	REST stands for REpresentational State Transfer.
3)	JAX-WS is the java API for SOAP web services.	JAX-RS is the java API for RESTful web services.
4)	SOAP permits XML data format only.	REST permits different data format such as Plain text, HTML, XML, JSON etc.

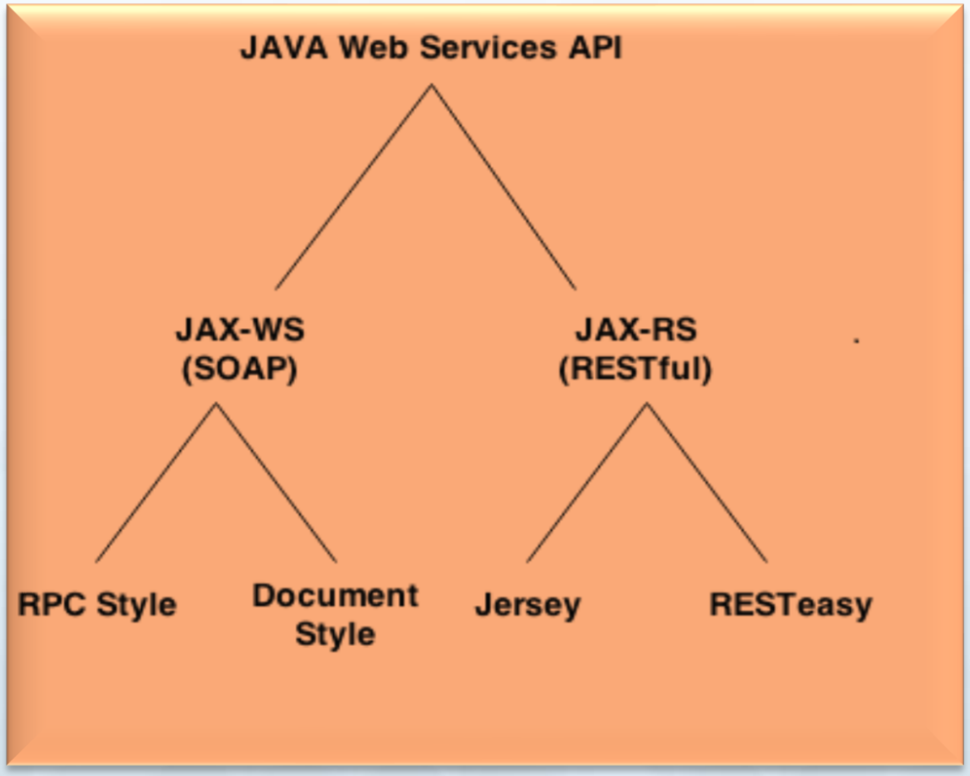
Web Service API

There are two main API's defined by Java for developing web service applications

1) **JAX-WS**: for SOAP web services. The are two ways to write JAX-WS application code: by RPC style and Document style.

2) **JAX-RS**: for RESTful web services. There are mainly 2 implementation currently in use for creating JAX-RS application: Jersey and RESTeasy.

Web Service API



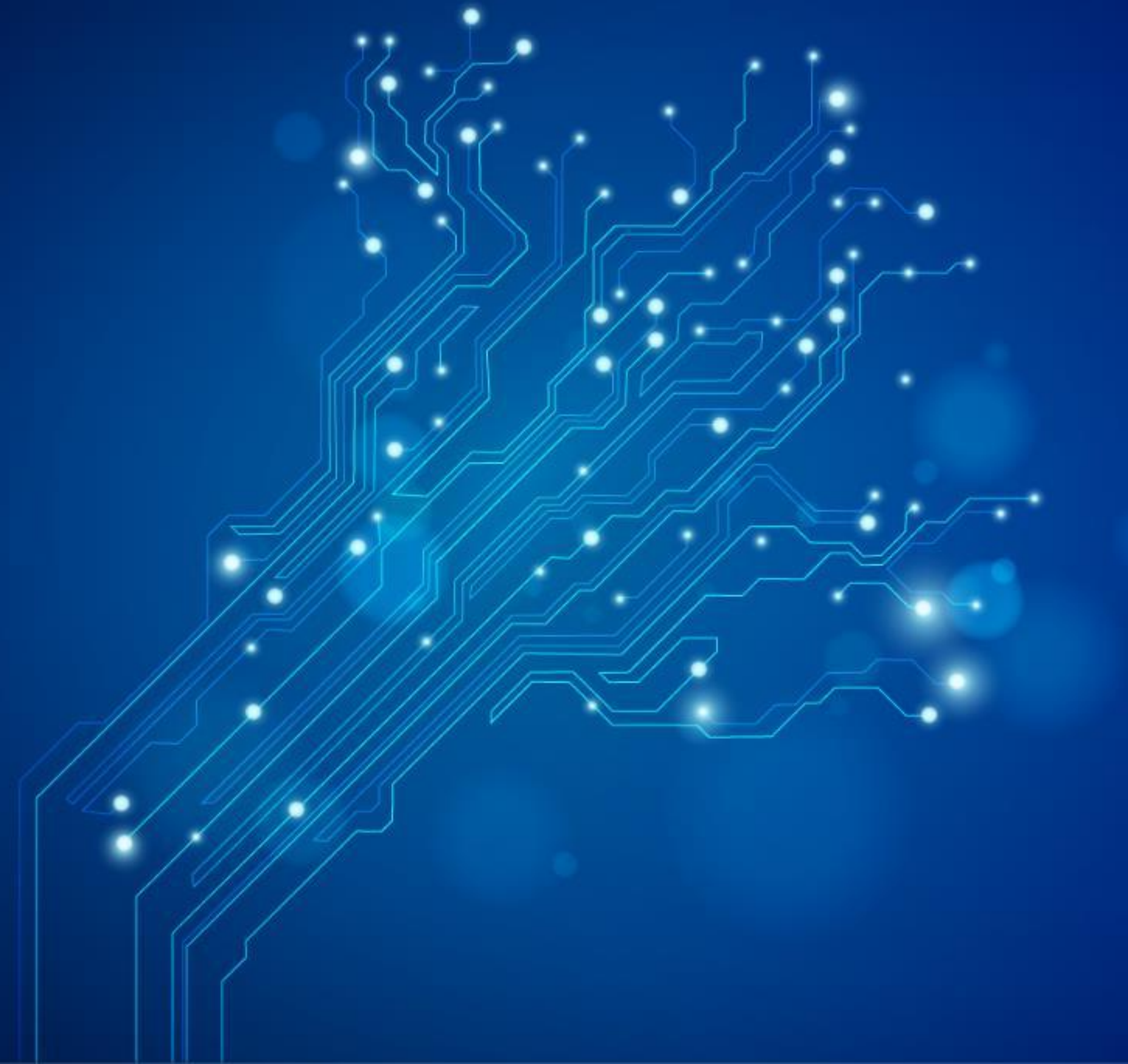
RESTful WEB SERVICE

RESTful Web Service

- REST is a web standards based architecture and uses HTTP Protocol for data communication.
- REST was first introduced by Roy Fielding in year 2000.
- REST uses various representations to represent a resource like Text, JSON and XML.

RESTful Resource Class

- Resource classes are POJOs
- The POJO class or method is annotated with @Path annotation
- They are also annotated with @GET, @PUT, @POST, or @DELETE



JAX-RS

REST with JAX-RS

Developing RESTful Web Services with JAX-RS

- The JAX-RS API uses Java programming language annotations to simplify the development of RESTful web services.
- Developers decorate Java programming language class files with JAX-RS annotations to define resources
- JAX-RS annotations are runtime annotations, runtime reflection will generate the helper classes and artifacts for the resource.

JAX-RS Annotations

Annotation	Description
@Path	The @Path annotation's value is a relative URI path indicating where the Java class will be hosted: for example, /helloworld.
@GET	The @GET annotation is a request method designator and corresponds to the similarly named HTTP method.
@POST	The @POST annotation is a request method designator and corresponds to the similarly named HTTP method
@PUT	The @PUT annotation is a request method designator and corresponds to the similarly named HTTP method.
@DELETE	The @DELETE annotation is a request method designator and corresponds to the similarly named HTTP method.
@HEAD	The @HEAD annotation is a request method designator and corresponds to the similarly named HTTP method.

JAX-RS Annotation cont…

@PathParam	**The @PathParam annotation is a type of parameter that can be extracted for use in the resource class. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation.**
@QueryParam	The @QueryParam annotation is a type of parameter that can be extracted for use in your resource class. Query parameters are extracted from the request URI query parameters.
@Consumes	The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client.
@Produces	The @Produces annotation is used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain".
@Provider	The @Provider annotation is used for anything that is of interest to the JAX-RS runtime, such as MessageBodyReader and MessageBodyWriter.

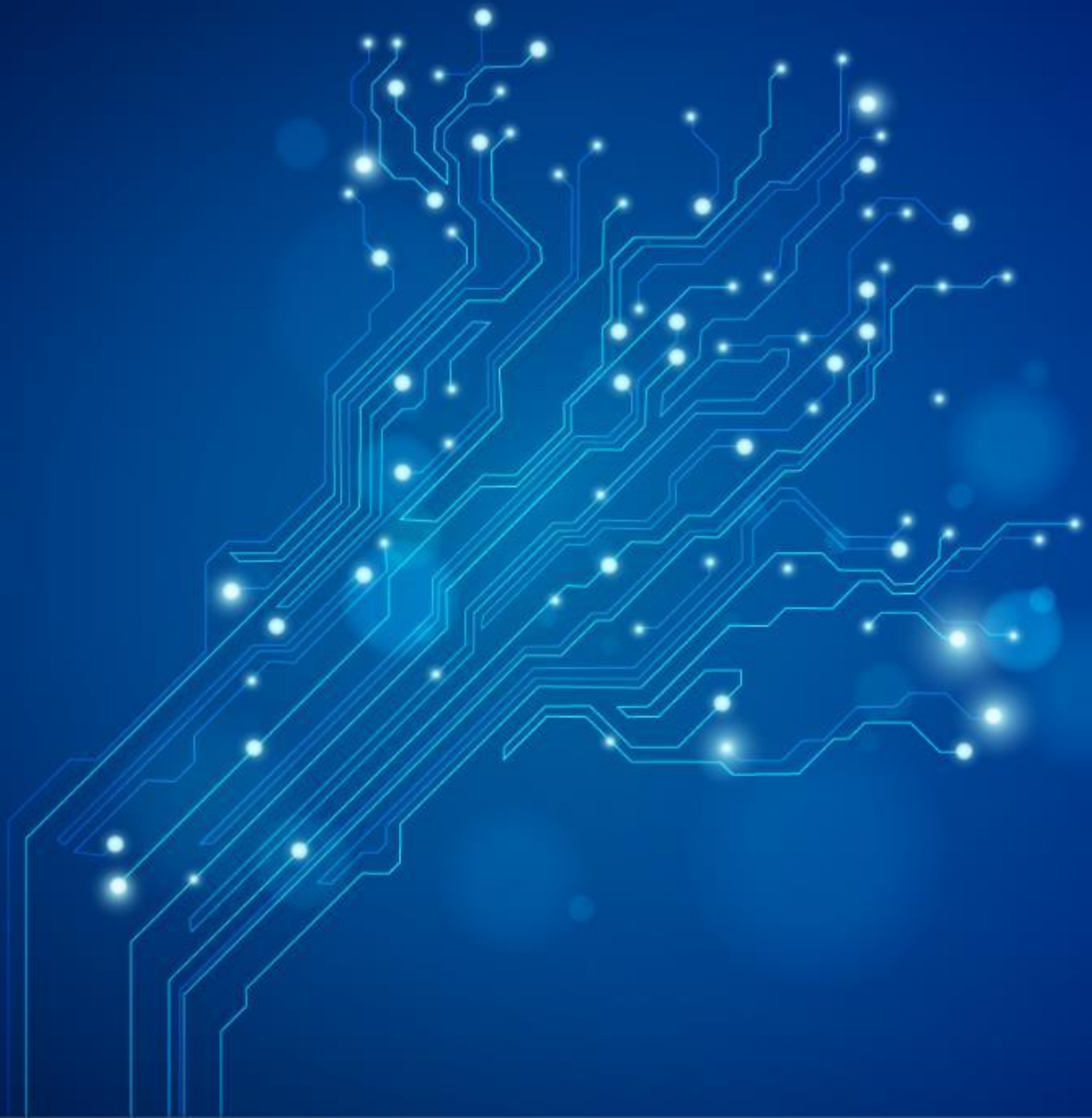
JERSEY

Jersey

- Jersey RESTful Web Services framework is open source, production quality, framework for developing RESTful Web Services
- In order to simplify development of RESTful Web services and their clients in Java, a standard and portable **JAX-RS API** has been designed
- Jersey provides it's own API that extend the JAX-RS toolkit with additional features and utilities to further simplify RESTful service and client development

Jersey - Goals

- It track the JAX-RS API and provide regular releases of production quality Reference Implementations that ships with GlassFish
- Provide APIs to extend Jersey & Build a community of users and developers
- Make it easy to build RESTful Web services utilizing Java and the Java Virtual Machine.



JSON

JSON - Introduction

- JSON: **J**ava**S**cript **O**bject **N**otation.

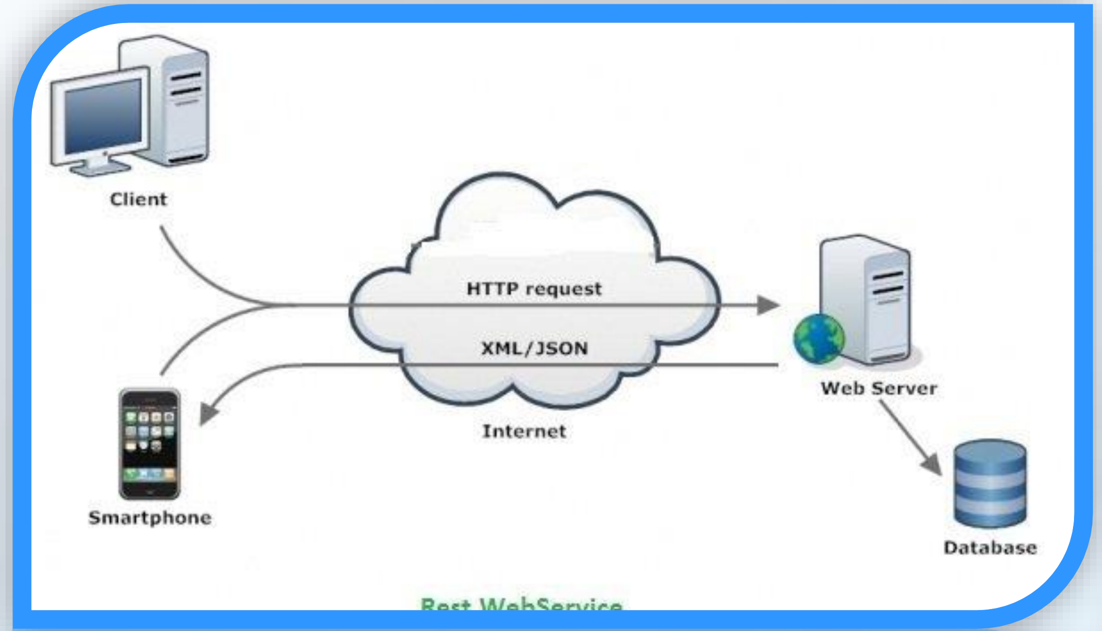
- JSON is lightweight data-interchange format.
- JSON is easy to read and write than XML.
- JSON is language independent

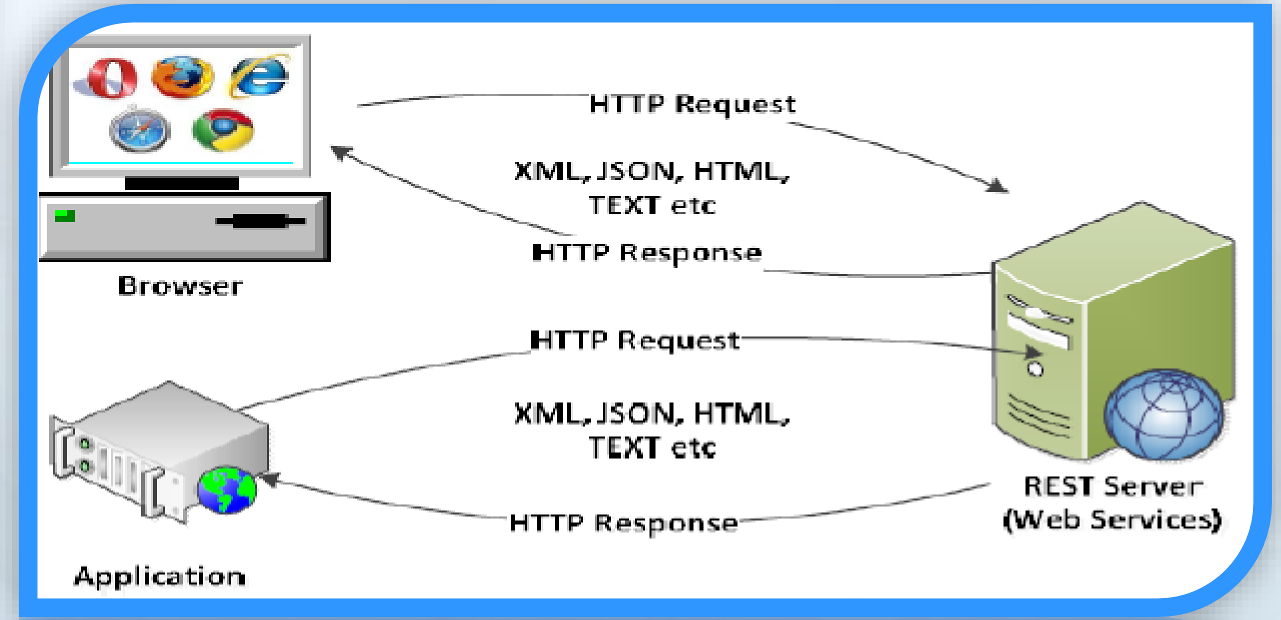
Uses of JSON

Why to Use JSON?

- JSON is a syntax for storing and exchanging data.
- JSON uses JavaScript syntax, but the JSON format is text only.
- Text can be read and used as a data format by any programming language.

JSON - Architecture




HTTP Request
XML, JSON, HTML, TEXT etc
HTTP Response
Browser
HTTP Request
XML, JSON, HTML, TEXT etc
HTTP Response
Application
REST Server (Web Services)

JSON – Data Types

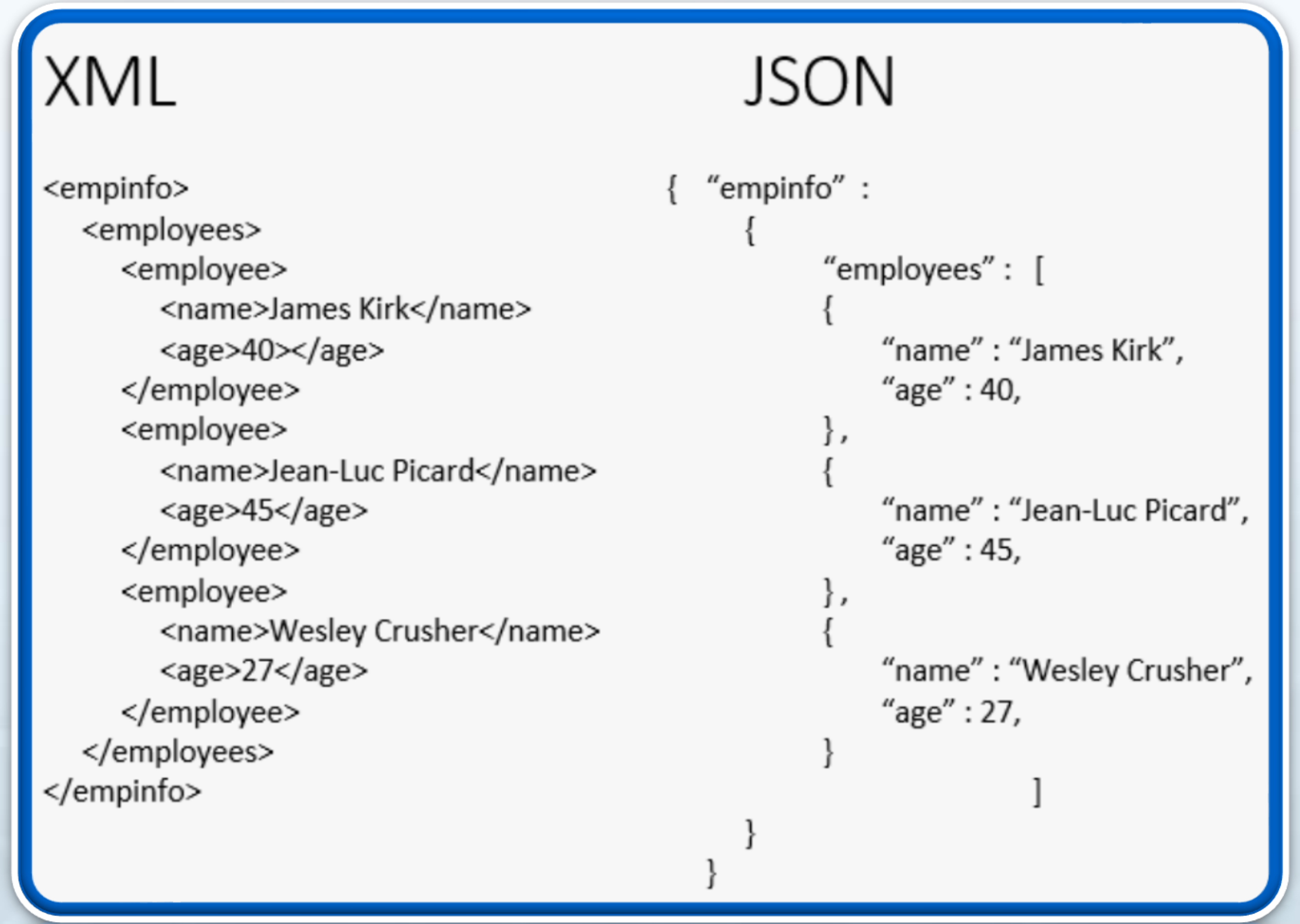
- Number
- String
- Boolean
- Array
- Object
- Null

JSON - Syntax

```
{
“Name”:”Thomas”,
“Age”:22,
“address”:{
“street”:”5th street”,
“city”:”Chennai”};
}
```

- Data is represented in name/value pairs.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).

JSON Vs XML

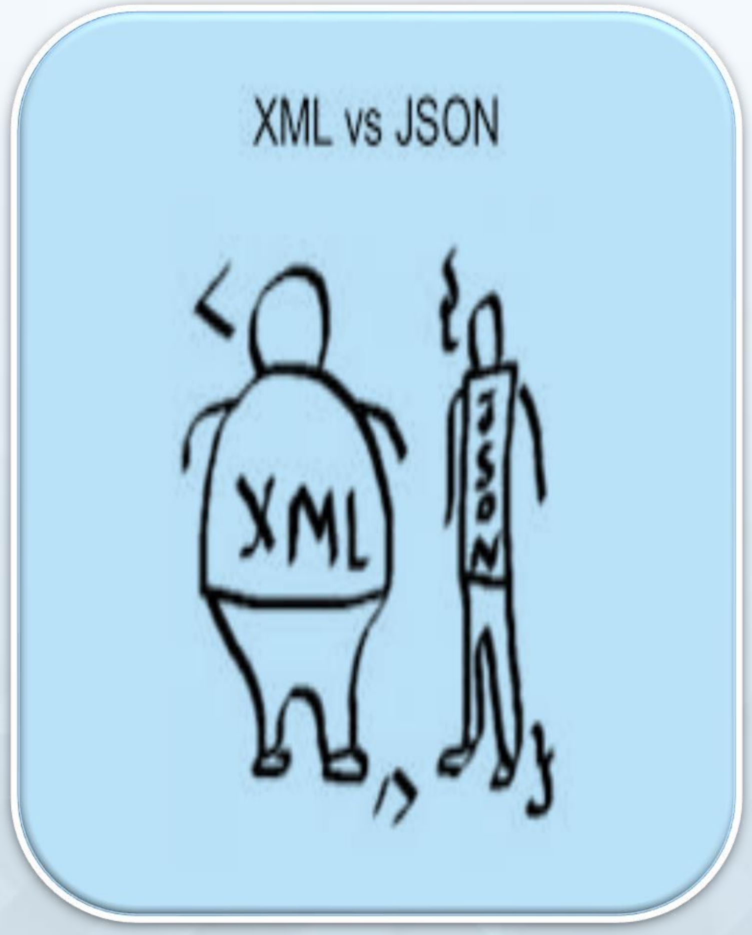


XML

```
<empinfo>
    <employees>
        <employee>
            <name>James Kirk</name>
            <age>40></age>
        </employee>
        <employee>
            <name>Jean-Luc Picard</name>
            <age>45</age>
        </employee>
        <employee>
            <name>Wesley Crusher</name>
            <age>27</age>
        </employee>
    </employees>
</empinfo>
```

JSON

```
{   "empinfo" :
        {
                "employees" :  [
                {
                    "name" : "James Kirk",
                    "age" : 40,
                },
                {
                    "name" : "Jean-Luc Picard",
                    "age" : 45,
                },
                {
                    "name" : "Wesley Crusher",
                    "age" : 27,
                }
                                ]
        }
}
```

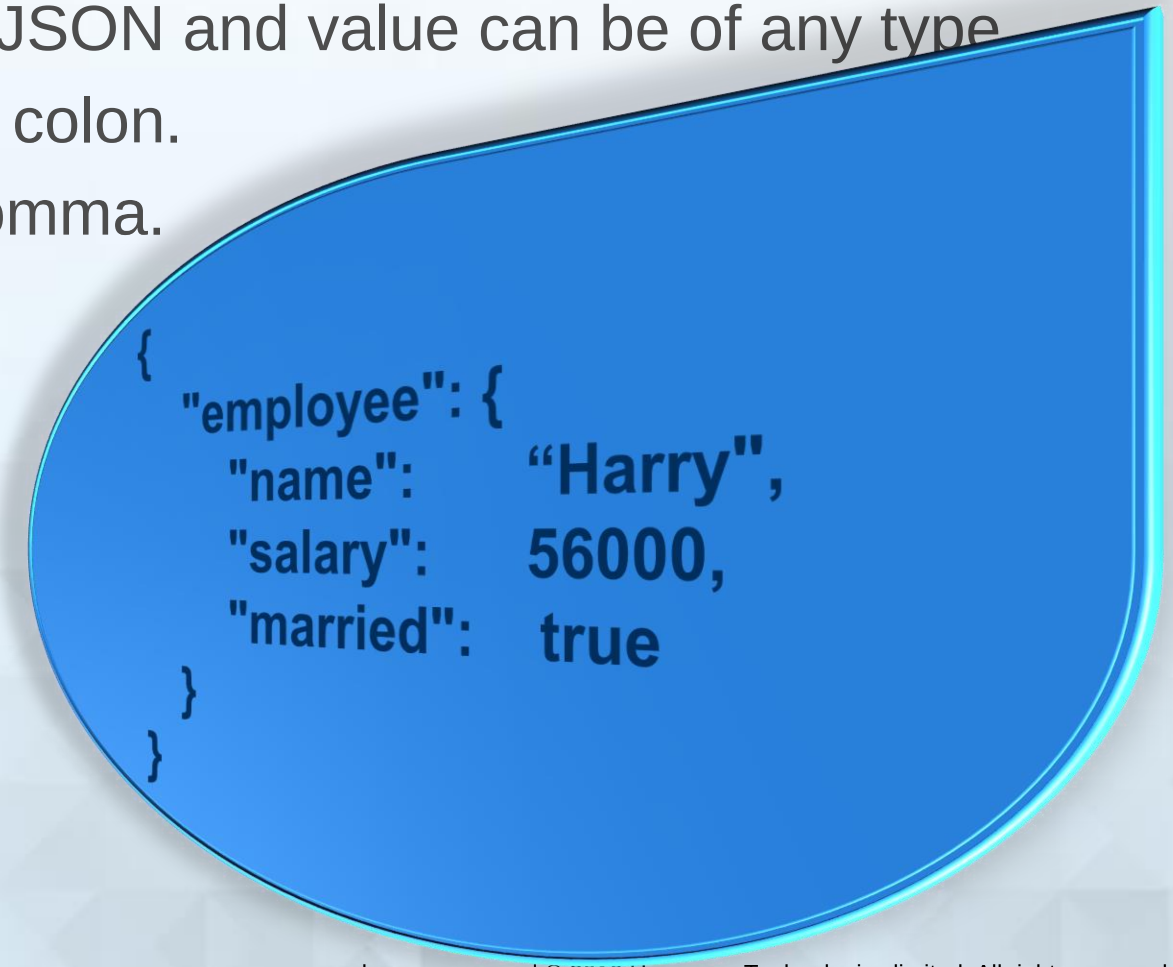


JSON Vs XML

1)	JSON is data-oriented.	XML is document-oriented.
2)	JSON doesn't provide display capabilities.	XML provides the capability to display data because it is a markup language.
3)	JSON supports array.	XML doesn't support array.
4)	JSON files are more human readable than XML.	XML files are less human readable.
5)	JSON supports only text and number data type.	XML support many data types such as text, number, images, charts, graphs etc. Moreover, XML offeres options for transferring the format or structure of the data with actual data.

JSON - Object

- JSON object holds key/value pair.
- Each key is represented as a string in JSON and value can be of any type
- The keys and values are separated by colon.
- Each key/value pair is separated by comma.



```
{
  "employee": {
    "name":      "Harry",
    "salary":    56000,
    "married":   true
  }
}
```

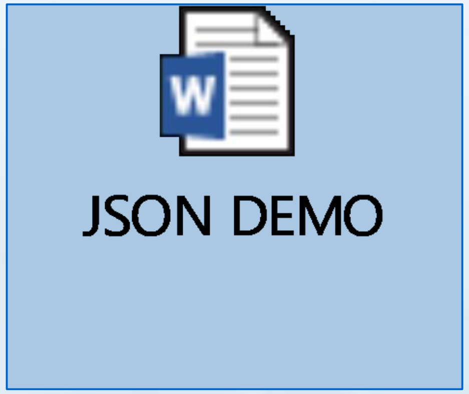
JSON - Array

- JSON array represents ordered list of values.
- JSON array can store multiple values.
- It can store string, number, boolean or object in JSON array.

```
{"employees":[
    {"name":"Ram", "email":"ram@gmail.com", "age":23},
    {"name":"Shyam", "email":"shyam23@gmail.com", "age":28},
    {"name":"John", "email":"john@gmail.com", "age":33},
    {"name":"Bob", "email":"bob32@gmail.com", "age":41}
]}
```

JSON - Demo

- Create a file **JsonDemo**
- Save the file as **JsonDemo.htm**



JSON
and
REST

Rest with Jersey, JAX-RS and Json

STEP 1:

- Create a Web project

STEP 2:

- Convert the Web project into Maven project
  - Right click the Web project-> Configure ->Convert to Maven Project

Rest with Jersey, JAX-RS and Json Cont…

STEP 3:

Add the following Servlet data in the Web.xml file

```
 <servlet>
<servlet-name>jersey-serlvet</servlet-name>
<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-
class>
<init-param>
<param-name>javax.ws.rs.Application</param-name>
<param-value>com.Hexa.RestService.FTPApplication</param-value>
</init-param>
<init-param>
<param-name>jersey.config.server.tracing</param-name>
<param-value>ALL</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>jersey-serlvet</servlet-name>
<url-pattern>/api/*</url-pattern>
</servlet-mapping>
```

Rest with Jersey, JAX-RS and Json Cont…

STEP 4:

Add the following dependencies in Pom.xml

```
<dependency>
<groupId>com.sun.jersey</groupId>
<artifactId>jersey-json</artifactId>
<version>1.8</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.
media/jersey-media-json-jackson -->
<dependency>
<groupId>org.glassfish.jersey.media</groupId>
<artifactId>jersey-media-json-jackson</artifactId>
<version>2.3.1</version>
</dependency>
```

Rest with Jersey, JAX-RS and Json Cont…

```xml
<dependency>
<groupId>org.glassfish.jersey.core</groupId>
<artifactId>jersey-server</artifactId>
<version>2.26</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.glassfish.jersey.cont
ainers/jersey-container-servlet -->
<dependency>
<groupId>org.glassfish.jersey.containers</groupId>
<artifactId>jersey-container-servlet</artifactId>
<version>2.26</version>
</dependency>
```

Rest with Jersey, JAX-RS and Json Cont…

STEP 5:

1. Create a java class "DataCenter.java"
2. Annotate the class with @Path() annotation

   @Path("data")

   public class DataCenter {}
3. Create a method and annotate it with

   @Produce():annotation is used to specify the MIME media types or representations a resource can produce and send back to the client

   @GET:The HTTP GET method is used to read (or retrieve) a representation of a resource.

Rest with Jersey, JAX-RS and Json Cont…

4. @PathParam: It is used to inject the value of URI parameter that defined in @Path expression, into Java method.

```
@GET
 //@Produces(MediaType.TEXT_HTML)
@Produces(MediaType.APPLICATION_JSON)
@Path("{name}")
public String getMsg(@PathParam("name") String name) {
String output = "Welcome to Web world: " + name;
System.out.println(output);
return output;
}
```

Rest with Jersey, JAX-RS and Json Cont…

STEP 6:

CORSFilter: **Cross Origin Resource Sharing**

- CORS is a mechanism supported by W3C to enable cross origin requests in web-browsers
- It requires support from both browser and server to work
- This is a Java Jersey Web Server filter implementation of server-side CORS for web containers such as Apache Tomcat and other Embedded Web Servers

Rest with Jersey, JAX-RS and Json Cont…

CORSFilter Code:

```
public void filter(ContainerRequestContext request,
            ContainerResponseContext response) throws IOException {
        response.getHeaders().add("Access-Control-Allow-Origin",
"*");
        response.getHeaders().add("Access-Control-Allow-Headers",
                "origin, content-type, accept, authorization");
        response.getHeaders().add("Access-Control-Allow-
Credentials", "true");
        response.getHeaders().add("Access-Control-Allow-Methods",
                "GET, POST, PUT, DELETE, OPTIONS, HEAD");
    }
}
```

Rest with Jersey, JAX-RS and Json Cont…

STEP 7:

Start the Tomcat Server in GIT bash

1. Open GIT bash
2. Specify the location of Tomcat in GIT bash
3. Execute the following command,
   - A. To shutdown the Tomcat Server

     ```
     $./bin/shutdown.sh
     ```
   - B. To Start the Tomcat Server

     ```
     $./bin/startup.sh
     ```
   - C. To view the logs

     ```
     $ tail –f logs/*
     ```

Rest with Jersey, JAX-RS and Json

STEP 8:

Open another GIT bash to Execute the rest service

1. mvn package
2. cp target/RestWeb-0.0.1-SNAPSHOT.war C:\Users\31410\Desktop\apache-tomcat-8.5.16-windows-x64\apache-tomcat-8.5.16
3. curl -vvv "http://localhost:8886/RestWeb-0.0.1-SNAPSHOT/api/data/Thomas"

HEXAWARE

Innovative Services

Passionate Employees

Delighted Customers

Thank you

www.hexaware.com