1️⃣

Add cover Add comment

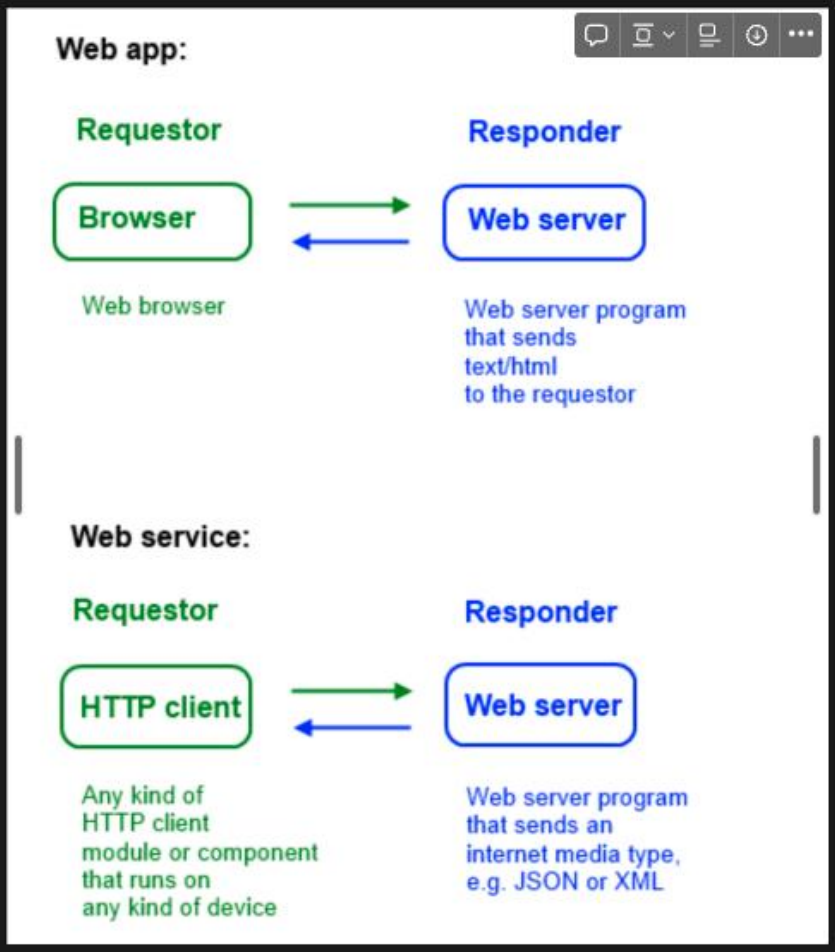
Introduction

What is a Web Service?

- A Web Service is an application that runs on a web server, and it accessed programatically.
- It is a software system designed to facilitate communication and interaction between different applications over the internet.
- It enables applications running on different platforms and written in different programming languages to exchange data and invoke functionalities provided by the web service.
- Web services typically follow a client-server model, where the client is an application or system that requests information or performs a specific task, and the server is the web service itself that provides the requested information or performs the task.\
- Web services use standard protocols such as **HTTP (Hypertext Transfer Protocol)** for communication and **XML (Extensible Markup Language)** for data representation.
- The most common web service architectures are **SOAP (Simple Object Access Protocol)** and **REST (Representational State Transfer)**.
  - SOAP-based web services use XML to structure messages and often rely on the Web Services Description Language (WSDL) to define the interface and operations of the service. They provide a more rigid and standardized approach to communication.
  - On the other hand, RESTful web services leverage the existing HTTP protocols and use lightweight data formats such as JSON (JavaScript Object Notation) for data exchange. They follow a stateless, resource-based approach where different HTTP methods (GET, POST, PUT, DELETE, etc.) are used to perform operations on resources identified by unique URLs.
- Web services can offer a wide range of functionalities, including data retrieval, data manipulation, business logic execution, and integration with other systems. They are commonly used in distributed systems, where different components or applications need to communicate and share information reliably and securely over the internet.

- Here are a few examples of web services:
  1. Weather Service
  2. Payment Gateway
  3. Social Media API:
  4. Geolocation Servic
  5. Stock Market Data
  6. Mapping and Routing Services
  7. Email Service

WEB APP vs WEB SERVICES



TERMINOLOGY

- **Resource:** A Resource is a Digital Asset. Familiar examples include a document, or an image. You identify a resource by using its URI (uniform resource identifier).
- **Representation:** A representation is a digital asset that is formatted as a specific internet media type.
    - Think about a scenario where a web service was used to manage students in a course. Each student is a resource - a digital asset - that can be identified by a URI.
    - If a user requested a specific resource through a web browser, you would expect that the resource would be *represented* by some HTML that included the student's name, student ID, and so on.
    - Alternatively, it's also possible to request the same specific resource - using the same URI - but also specify that it be returned in a data format (like JSON or XML, discussed later). The server will return a data *representation* of the resource.
- **Internet Media Type:** An internet media type is defined as a data format for a representation of a resource on the internet. For web service programmers, two important internet media types are used as data formats, **JSON** and **XML.** Both are plain-text data formats. They are somewhat human-readable.
- **HTTP Client:** An HTTP client is a software application or a program that establishes a connection to a remote server using the Hypertext Transfer Protocol (HTTP) and sends requests to retrieve information or perform actions on the server. It acts as a client-side component in the client-server architecture of the World Wide Web.

JSON (JavaScript Object Notation)

It is a lightweight data-interchange format. It is language-independent, however it uses conventions that were first suggested by the JavaScript object literal or initializer. JSON has become the *de facto* data-interchange format standard.

- In JSON, each property name must be surrounded by quotes (typically double-quotes). In pure JavaScript, this is optional for single-word property names.
- In JSON, there is no Date type. Dates are expressed as strings, almost always in ISO 8601 format. In contrast, JavaScript does have a Date object (not a *type*, but an *object*).
- **Data Types:**
    1. string
    2. number (integer or decimal)
    3. object (i.e. { })
    4. array (i.e. [ ])
    5. null

State - Data and Interaction

> **HTTP is a stateless protocol**

In a web app or web service context, the meaning of the word *state* is the current value of a resource.

- At the server, a resource could be in the form of an item in a persistent store (i.e. a database system), or it could be in the form of an item that is generated upon request. That difference doesn't matter; what matters is that when a resource is requested, its current state is sent, or *transferred*, as the response.
- A resource's state can be *changed* (or modified), of course. The stimulus for the change can originate from anywhere in the distributed system, ranging from server-based batch or automatic processes that modify the persistent store (apart from or separately from the web app or web service), through to client-sent requests that are received by a web app or web service.
- One of the characteristics of a web app or web service is that it can be used by a single client app, or by theoretically unlimited numbers of client apps.
- The main point is that the server effectively treats every request as separate/discrete/atomic, and *does not*actively maintain any notion of a logical session over time. In other words, no interaction state maintenance or management is done at the server.
- Therefore, if the interaction state of a (message exchange) session is important to maintain, it's done by the client.

REST (Representational State Transfer)

It is an architectural style for designing networked applications that communicate over the Hypertext Transfer Protocol (HTTP). It is a widely adopted approach for building web services and APIs due to its simplicity, scalability, and ease of integration.

Here are some key characteristics of a REST API:

1. **Resource-Based:** REST APIs model resources as the primary entities exposed by the API. Resources can be anything that can be uniquely identified and manipulated, such as users, products, orders, or any other data entity. Each resource is typically represented by a unique URL (Uniform Resource Locator) or endpoint.
2. **Stateless:** A REST API is stateless, meaning that each request from a client to the server contains all the necessary information for the server to understand and process that request. The server does not retain any client-specific session state between requests. Any required session information is usually sent in the request headers.
3. **CRUD Operations:** REST APIs typically support CRUD (Create, Read, Update, Delete) operations for manipulating resources. These operations are mapped to the standard HTTP methods: POST (Create), GET (Read), PUT/PATCH (Update), and DELETE (Delete). For example, to create a new user, you would send a POST request to the appropriate user endpoint.
4. **Uniform Interface:** REST APIs adhere to a uniform set of constraints and principles. They use standard HTTP methods and status codes to indicate the desired action and provide meaningful responses. Additionally, they often utilize common data formats such as JSON (JavaScript Object Notation) or XML for representing and exchanging data.
5. **Hypermedia as the Engine of Application State (HATEOAS):** HATEOAS is a principle in REST APIs that allows clients to navigate and discover resources dynamically by including hyperlinks in the responses. These hyperlinks provide the necessary information and actions that can be taken by the client based on the current state of the application.

HTTP Request

HTTP is a message-passing protocol, between two endpoints. One end will send a request, and the other will send a response. HTTP defines several kinds of requests. At a minimum, a simple "get me some data" request must include the *HTTP method* (e.g. GET, POST, etc.), and a *URL*. Other kinds of requests must include other metadata.