Java Programming

Web Service

- **Definition:** A software system that allows different applications to communicate over the internet.
- **Example:** A weather app, it's actually talking to a weather web service to get the latest weather information!

1.Client: port.getTemperature(“CHENNAI")
2. JAX-WS: Creates SOAP XML
3. HTTP Client: POST to http://weather.service/temp
4. Server receives HTTP request
5. SOAP engine processes XML
6. Your service method runs
7. Response follows reverse path

What are Web services?

- **Web services**: Client and server applications communicating over the World Wide Web's HTTP.
- **Interoperability:** Allows software applications on different platforms and frameworks to work together.
- **Standardized communication**: Described by W3C for consistency and compatibility across systems.
- **Extensibility:** Can be easily extended to add new features or capabilities.
- **Machine-processable descriptions**: Use XML for machine-readable service definitions.
- **Loose coupling**: Web services can be combined to create complex operations without tightly linking systems.
- **Cross-language compatibility**: Applications in different programming languages can exchange data.
- **Data exchange over networks**: Similar to inter-process communication but over the Internet.

Web services

- A web service makes software application resources available over networks using standard technologies.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems
- Web services are based on standard interfaces and hence they can communicate even if they are running on different operating systems and are written in different languages.
- Therefore, Web services are an excellent approach for building distributed applications that must incorporate diverse systems over a network.

Why Do We Need Web Services?

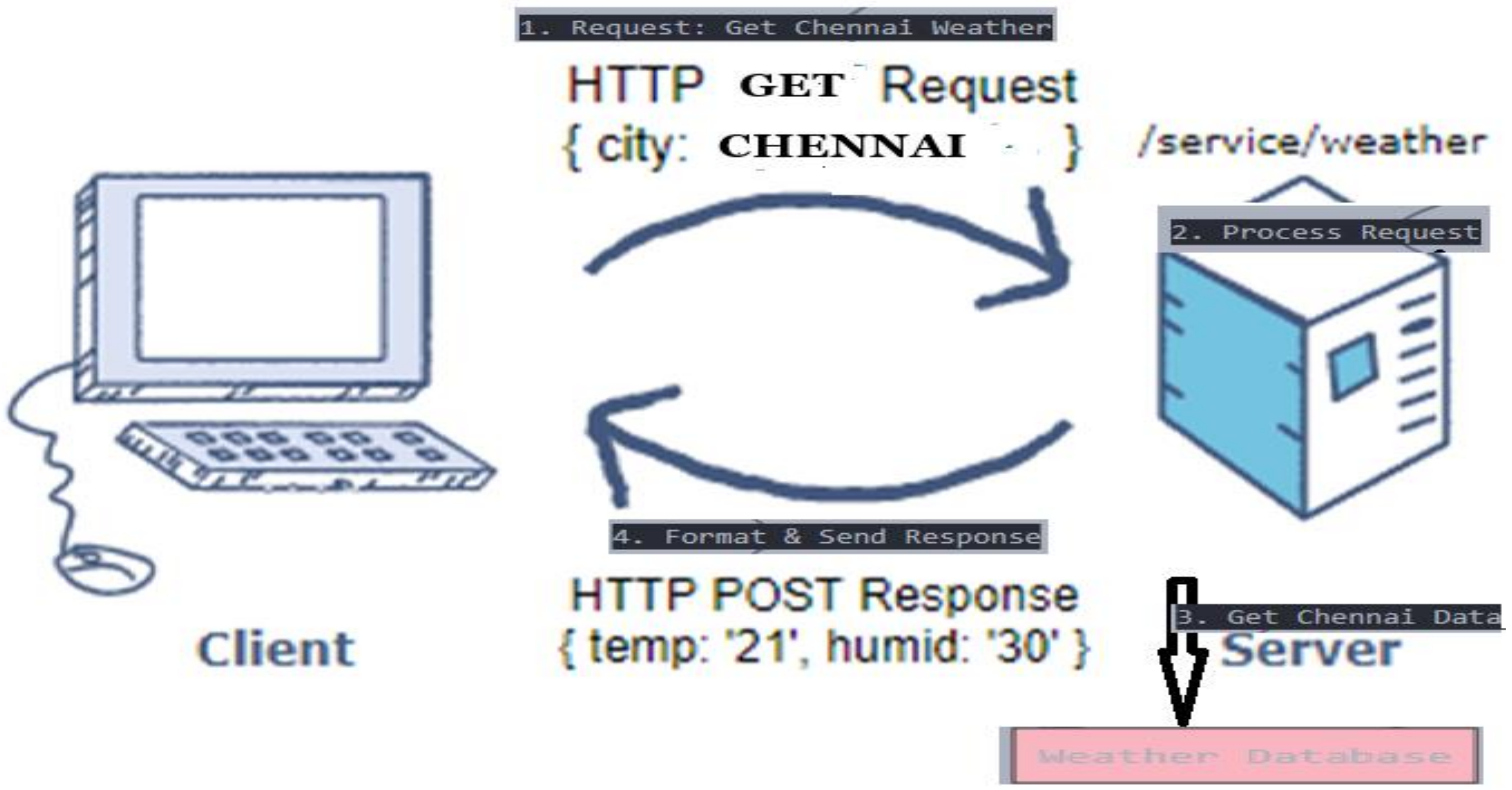
- **Share Information**
  - Like a library that many people can use
  - Example: Google Maps service used by many apps
- **Connect Different Applications**
  - Like a translator helping people who speak different languages
  - Example: A website and mobile app using the same data
- **Reuse Functionality**
  - Like using the same kitchen to serve many restaurants
  - Example: Payment service used by different online shops

How Does a Web Service Work? 🛠️

- **Basic Steps:**
- **Request** (Like ordering food)
  - App sends a request for information
  - Example: "What's the weather in Chennai?“
  - GET https://api.weather.com/ Chennai
- **Processing** (Like cooking food)
  - Web service receives request
  - Finds or calculates the answer
  - Example: Looking up Chennai's weather data
- **Response** (Like serving food)
  - Web service sends back the information
  - Example: “Chennai: 28°C, Sunny"

```
// This is like receiving your order
{ "city": "Chennai", "temperature": "28°C", "condition": "Sunny" }
```

Web Service Basic Flow



Interacting with a web service

Web Service

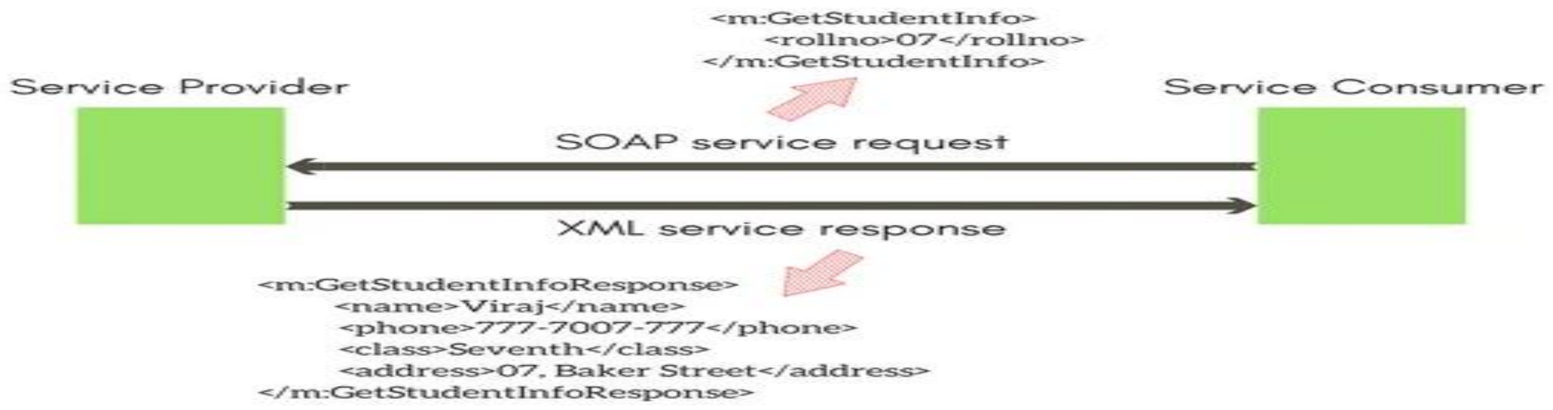
- A web service:
  - Publicly describes its own functionality through a WSDL file
    - WSDL is an XML, and it stands for Web Service Description Language. WSDL describes all the methods available in the web service, along with the request and response types. It describes the contract between service and client.
  - Communicates with other applications via XML messages, often formatted with SOAP
  - Employs a standard network protocol such as HTTP

Common Types of Web Services

- **REST** (Most Common)
  - Simple and popular
  - Uses standard web methods (GET, POST)
  - Like ordering from a menu with clear options
- **SOAP**
  - More formal and structured
  - Like a fancy restaurant with strict procedures
  - Used in business applications

SOAP

- SOAP is an XML-based protocol.
- SOAP stands for **Simple Object Access Protocol**.
- SOAP was intended to be a way to do remote procedure calls to remote objects by sending XML over HTTP.
- POST https://weatherservice.com/WeatherService Content-Type: text/xml



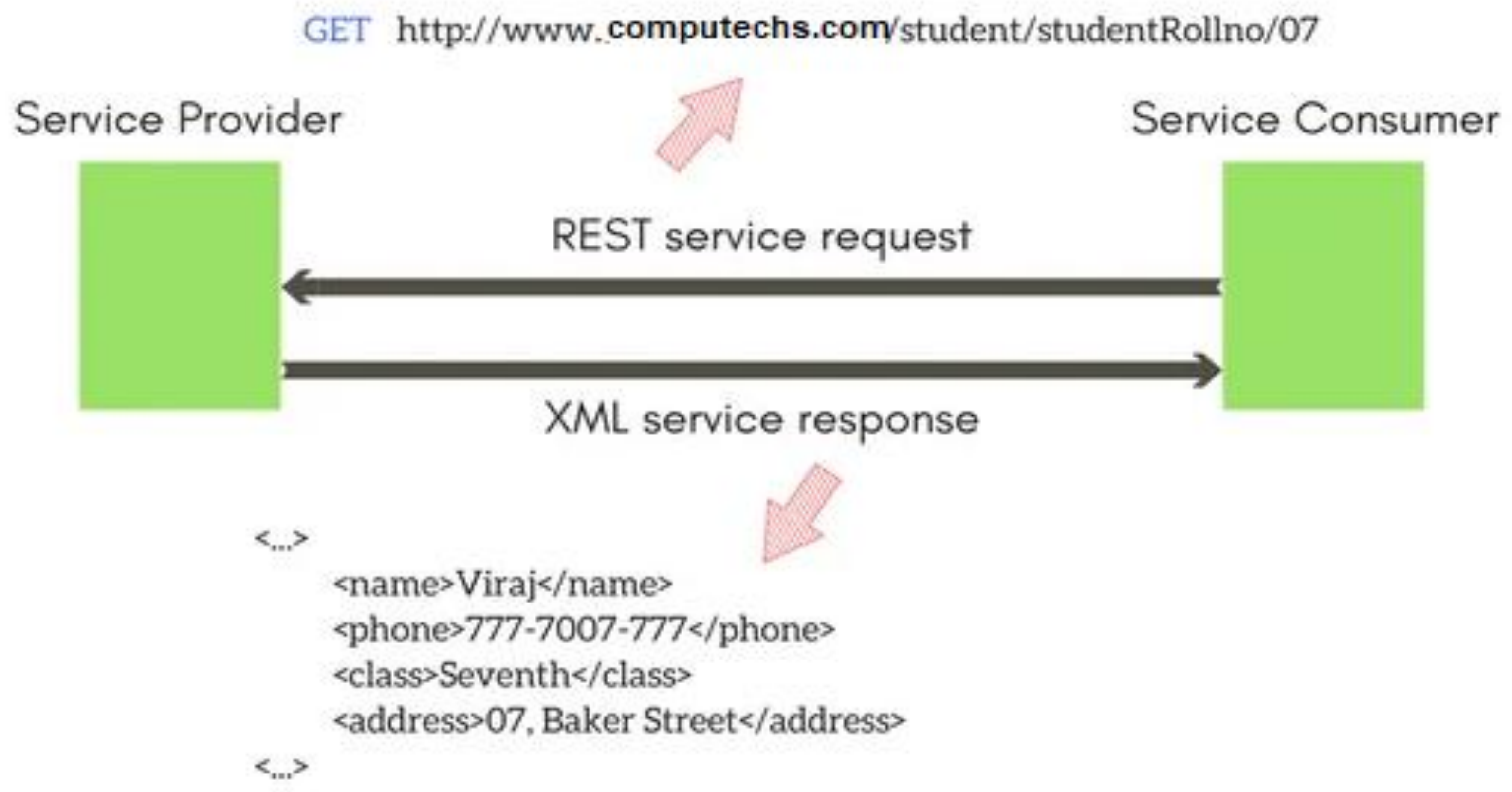
```
<m:GetStudentInfo>
    <rollno>07</rollno>
</m:GetStudentInfo>
```

```
<m:GetStudentInfoResponse>
    <name>Viraj</name>
    <phone>777-7007-777</phone>
    <class>Seventh</class>
    <address>07, Baker Street</address>
</m:GetStudentInfoResponse>
```

REST Web Services

- The **REST** stands for **Representational State Transfer**.
- REST is not a set of standards or rules, rather it is a style of software architecture.
- The applications which follow this architecture are referred to as **RESTful**
- REST locates the resources by using URL and it depends on the type of **transport** protocol(with HTTP - GET, POST, PUT, DELETE,...) for the actions to be performed on the resources.
- while requesting the data from a website, the data should be in a browser readable format, which is **HTML**, while in case of the REST API, response can be anything like **XML/JSON** or any other media type.

REST Web Services



```
<...>
    <name>Viraj</name>
    <phone>777-7007-777</phone>
    <class>Seventh</class>
    <address>07, Baker Street</address>
<...>
```

Difference between REST and SOAP

REST

- REST is a style of software architecture.
- REST can use SOAP because it is a concept and can use any protocol like HTTP, SOAP etc.
- REST uses URI to expose business logic.
- REST inherits security measures from the underlying transport protocols.
- REST accepts different data formats like, Plain Text, HTML, JSON, XML etc.

SOAP

- SOAP is a protocol or a set of standards.
- SOAP cannot use REST because it itself is a protocol.
- SOAP uses the service interface to expose business logic.
- SOAP defines its own security layer.
- SOAP only works with XML format.

HTTP

- HTTP is a **TCP/IP** based communication protocol, which is used to deliver data (HTML files, image files, query results, etc.) accross the World Wide Web.
- The basic features of HTTP are:
  - HTTP is **connectionless**.
  - HTTP is **media independent**, which means any type of data can be sent through the http.
  - HTTP is **stateless**, neither the server nor the client keeps a track of the last request.
- HTTP makes use of the **Uniform Resource Identifier (URI)** to identify any given resource and establish a connection
- Therefore, while requesting the data from a website, the data should be in a browser readable format, which is **HTML**, while in case of the REST API, response can be anything like **XML/JSON** or any other media type.

REST WEB SERVICE

- Server(exposing web service)
- Client (consumes web service)

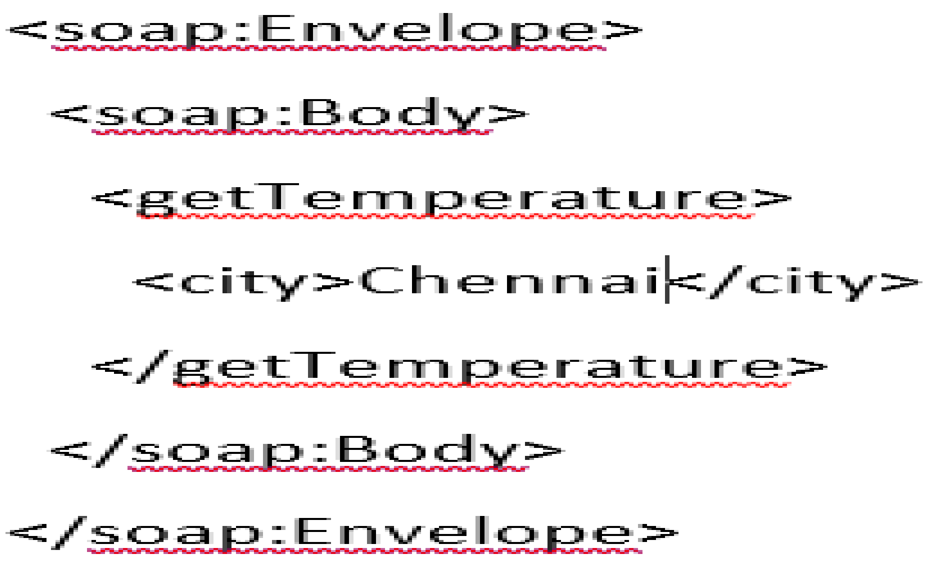
JSON VS XML

```
{
 "rollno":"10",
 "firstName":"Amit",
"lastName":"Agarwal",
"age":"25"
"contactNumber":"98877271127"
}
```

```
<Student>
 <rollno>10</rollno>
 <firstName>Amit</firstName >
<lastName>Agarwal</lastName>
 <age>25</age>
<contactNumber> 98877271127 </contactNumber>
 </Student>
```

Example: SOAP

- POST https://weatherservice.com/WeatherService Content-Type: text/xml



```
<soap:Envelope>
  <soap:Body>
    <getTemperature>
      <city>Chennai</city>
    </getTemperature>
  </soap:Body>
</soap:Envelope>
```

SOAP:*SOAP Response in XML*

```
<?xml version="1.0" encoding="UTF-8"?>
 <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://example.com/weather"> <soapenv:Header/>
<soapenv:Body>
 <web:GetTemperatureResponse>
 <web:temperature>
 <web:city>Chennai</web:city>
 <web:value>28.5</web:value>
 <web:unit>Celsius</web:unit>
 <web:timestamp>2024-03-22T10:00:00Z</web:timestamp>
<web:details>
 <web:humidity>65</web:humidity>
<web:pressure>1013</web:pressure>
 <web:conditions>Partly Cloudy</web:conditions> </web:details>
</web:temperature>
 </web:GetTemperatureResponse>
</soapenv:Body>
 </soapenv:Envelope>
```

REST

- GET https://api.weather.com/v1/temperature/Chennai
  Accept: application/json
- REST JSON Response Format

```
{
  "temperature": {
    "city": "Chennai",
    "value": 28.5,
    "unit": "Celsius",
    "timestamp": "2024-03-22T10:00:00Z",
    "details": {
      "humidity": 65,
      "pressure": 1013,
      "conditions": "Partly Cloudy"
    }
  }
}
```

Real-World Examples

- **Weather Services**
  - Apps get weather data
  - Updates every few minutes
  - Used by weather apps
- **Payment Services**
  - Process credit cards
  - Handle money transfers
  - Used by online shops
- **Social Media Services**
  - Share posts
  - Get updates
  - Used by social apps

Benefits of Web Services

- **Easy to Use**
  – Put in request, get response
- **Works Everywhere**
  – Different apps can use same service
- **Always Available**
  – Can use anytime

Definitions : API

- API (Application Programming Interface)
  - **Definition:** A set of rules that lets one application talk to another.
  - **Simple Example:** Like a menu at a restaurant - it tells you what you can order and how to order it.

Definitions : **HTTP**

- **HTTP (Hypertext Transfer Protocol)**
- **Definition:** The language that web services use to communicate.
- **Simple Example:** Like the common language between you and the waiter to order food.

Definitions :JSON

- **JSON (JavaScript Object Notation)**
- **Definition:** A common format for sending and receiving data.

Definitions :REST

- **REST (Representational State Transfer)**
- **Definition:** A popular style of building web services that's simple and standard.