REST API

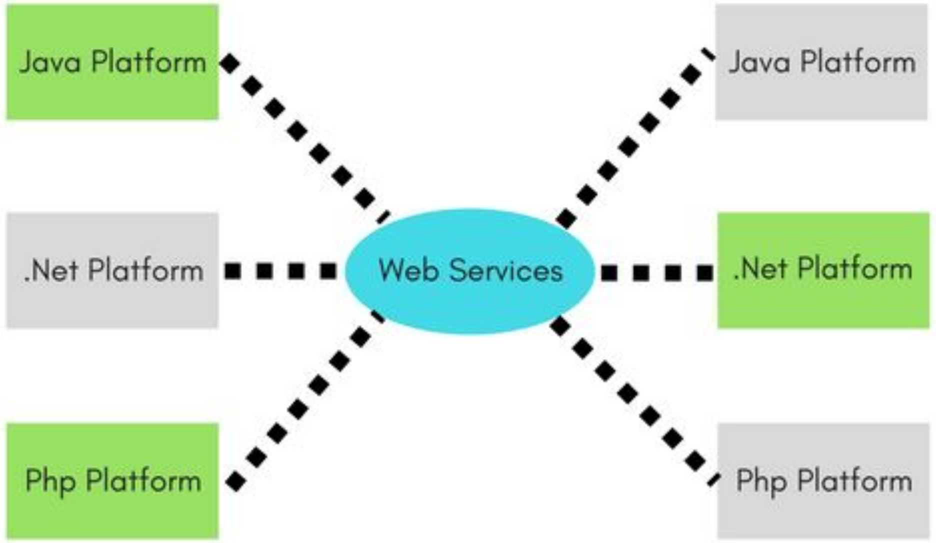
REpresentational State Transfer

What is API?

- Application Programming Interface
- In practice, an API is “a set of functions and procedures” that allow you to access and build upon the data and functionality of an existing application.

Web services

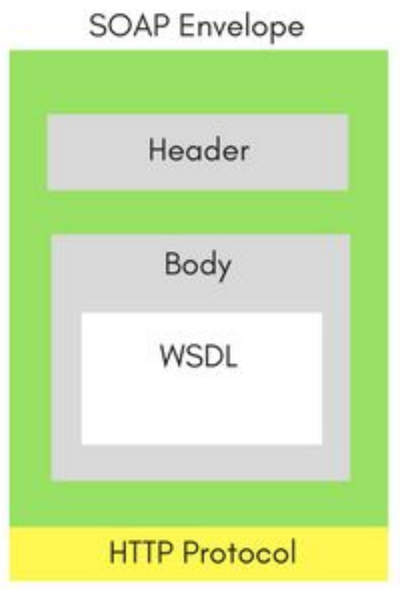
- Web Services are client and server applications that communicate over the World Wide Web's (WWW) Hypertext Transfer Protocol (HTTP).
- Provide a standard means of interoperating between software applications running on a variety of platforms and frameworks
- A web service is a function or method which we can call by sending an HTTP request to a URL, with arguments and the service returns the result back as response.
- Platform independent


Java Platform
.Net Platform
Php Platform
Web Services
Java Platform
.Net Platform
Php Platform

Types of web services

- SOAP (Simple Object Access Protocol) web services
- REST (REpresentational State Transfer)web services

SOAP Web Services



WSDL (Web Services Description Language)

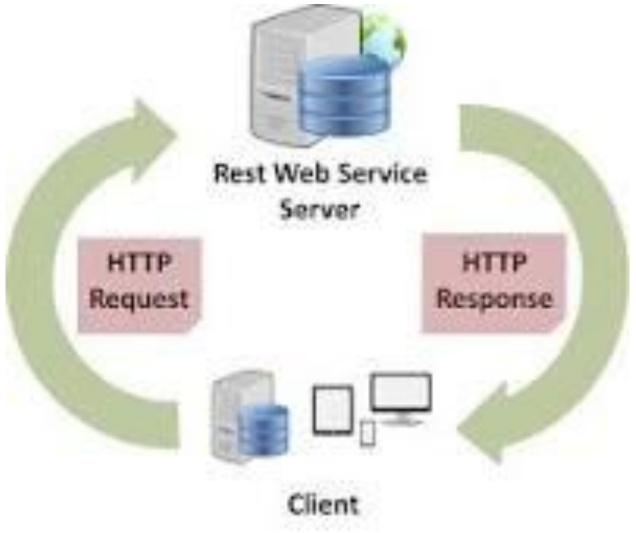
```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>
```

```
<binding name = "Hello_Binding" type = "tns:Hello_PortType">
    <soap:binding style = "rpc"
      transport = "http://schemas.xmlsoap.org/soap/http"/>
    <operation name = "sayHello">
      <soap:operation soapAction = "sayHello"/>
      <input>
        <soap:body
          encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
          namespace = "urn:examples:helloservice"
          use = "encoded"/>
      </input>

      <output>
        <soap:body
          encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
          namespace = "urn:examples:helloservice"
          use = "encoded"/>
      </output>
    </operation>
  </binding>

  <service name = "Hello_Service">
    <documentation>WSDL File for HelloService</documentation>
    <port binding = "tns:Hello_Binding" name = "Hello_Port">
      <soap:address
        location = "http://www.examples.com/SayHello/" />
    </port>   </service></definitions>
```

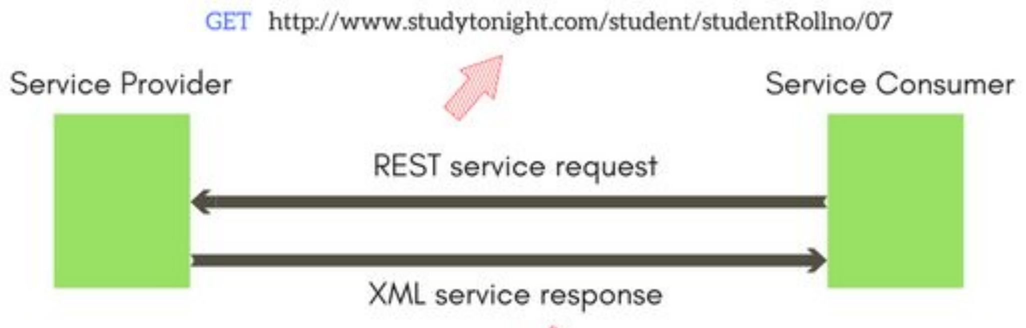
REST Web Services

- REpresentational State Transfer
- REST is not a set of standards or rules.
- REST is a style of software architecture.
- The applications which follow this architecture are referred to as RESTful
- Exposes as an Object
- Object is a noun not verb
- Apply a verb to noun to perform action

REST Architecture



Example of REST



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

What is CRUD

Create, Read, Update, Delete

Create

Creates a new resource

```
“book”: {
  "id": 1,
  “title”: “Web Security”,
  “author”: “Williams”,
  “isbn”: “12612”
}
```

Read

Retrieves resource details

```
“book”: {
  "id": 1,
  “title”: “Web Security”,
  “author”: “Williams”,
  “isbn”: “12612”
}
```

Update

Update resource details

```
“book”: {
  "id": “1”
  “title”: “Web Application”,
  “author”: “Peter kim”,
  “isbn”: “234”
}
```

Delete

Delete a resource

```
“book”: {
  "id": “1”
  “title”: “Web Application”,
  “author”: “Peter kim”,
  “isbn”: “234”
}
```

CRUD and REST

CREATE	-	POST
READ	-	GET
UPDATE	-	PUT
DELETE	-	DELETE

Response from REST Web service

It can be simple XML or JSON or any other media-type

Response status codes

- 1xxx- Informational codes
- 2xxx - Success codes
  - 200 - Ok
  - 201 - Created
  - 202 - Accepted
  - 204 - No content
- 3xxx - Used in case of redirections
- 4xxx - Client request error
- 5xxx - Server errors

General Response codes

- GET — return 200 (OK)
- POST — return 201 (CREATED)
- PUT — return 200 (OK)
- DELETE — return 204 (NO CONTENT)

REST API Design Principles

1. **Unique Identifier**

REST APIs are designed around resources, which are any kind of object, data, or service that can be accessed by the client. A resource has an identifier, which is a URI that uniquely identifies that resource.

Example: The URI for an employee can be: /employees/1234

REST API Design Principles

2. Resource base URLs

There should be only 2 base URLs per resource. The first URL is for a collection and the second is for a specific element in the collection.

Example (Collections): /employees

Example (Specific Elements): /employees/1234

REST API Design Principles

3. Nouns are good and verbs are bad

Avoid using verbs and use only nouns.

GET /getAllEmployees

GET /getEmployees/1234

POST /addEmployee

REST API Design Principles

4. Use HTTP Verbs

Resource	POST (Create)	GET (Read)	PUT (Update)
/employees (Collection)	Create New Employee	List All Employees	Bulk Updates of Employees
/employees/{id} (Element)	Error	List Employee Based on ID	Update A Specific Employee

REST API Design Principles

5. Associations

Collection	/employees
Specific Element	/employees/1234

REST API Design Principles

6. Asynchronous operations

HTTP status code 202 (Accepted) to indicate the request was accepted for processing but is not completed.

Steps of designing a REST APIs

1. Which objects to expose and their respective representations

   Ex: Employee as a object, Representation of Employee Object

   ```
   {
       "Id":"1206",
       "Name":"Siva",
       "Address":"Hyderabad"
   }
   ```

Steps of designing a REST APIs

2. Make the URI easy for the client

- http://employeeinfo/v1.4/employee/1206 - this gives 1206 employee information
- http://employeeinfo/v1.4/employees?name=Tom - this give all employees information whose name is Tom.
- employee - represents one employee and employees - represents collection of employees are two resources

Steps of designing a REST APIs

3. Represent resource with noun not verb

- http://employeeinfo/v1.4/employees?name=siva - correct
- http://employeeinfo/v1.4/getemployees?name=siva - not correct

Request Form

- an HTTP verb, which defines what kind of operation to perform
- a header, which allows the client to pass along information about the request
- a path to a resource
- an optional message body containing data

HTTP Verbs

GET - used to retrieve the data of a resource identified on the URI

POST - Used to create new resource

PUT - used to update/replace a resource

DELETE - used to delete a resource on the server

This we call CRUD (Create, Read,Update, Delete) operations on resources

Sample application Demo

- Nodejs
- Express
- REST APIs
- mongodb

Thank you