Introduction to API

- **Explain REST and RESTFUL?**

One of the most popular types of API is REST or, as they're sometimes known, RESTful APIs. REST or RESTful APIs were designed to take advantage of existing protocols. While REST - or Representational State Transfer - can be used over nearly any protocol, when used for web APIs it typically takes advantage of HTTP. This means that developers have no need to install additional software or libraries when creating a REST API.

One of the key advantages of REST APIs is that they provide a great deal of flexibility. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia. This flexibility allows developers to build an API that meets your needs while also meeting the needs of very diverse customers.

- **Mention what are the HTTP methods supported by REST?**

The following subset of HTTP methods are supported for the REST :

- **GET**

  The GET method retrieves specific information from the server as identified by the request URI.

- **PUT**

  The PUT method requests that the message body sent with the request be stored under the location provided in the HTTP message.

- **DELETE**

  The DELETE method deletes the specified resources.

- **POST**

  The POST method modifies data on the server from which a request was sent.

- **HEAD**

  The HEAD method is similar to the GET method except the message body is not returned in the response. The response only includes metainformation, such as a response code or corresponding headers.

Introduction to API

- **Explain the architectural style for creating web API?**

There are six architectural constraints which makes any web service are listed below:

Uniform Interface: It is a key constraint that differentiate between a REST API and Non-REST API. It suggests that there should be an uniform way of interacting with a given server irrespective of device or type of application (website, mobile app).
There are four guidelines principle of Uniform Interface are:

- **Resource-Based:** Individual resources are identified in requests. For example: API/users.
- **Manipulation of Resources Through Representations:** Client has representation of resource and it contains enough information to modify or delete the resource on the server, provided it has permission to do so. Example: Usually user get a user id when user request for a list of users and then use that id to delete or modify that particular user.
- **Self-descriptive Messages:** Each message includes enough information to describe how to process the message so that server can easily analyses the request.
- **Hypermedia as the Engine of Application State (HATEOAS):** It need to include links for each response so that client can discover other resources easily.
-

Stateless: It means that the necessary state to handle the request is contained within the request itself and server would not store anything related to the session. In REST, the client must include all information for the server to fulfill the request whether as a part of query params, headers or URI. Statelessness enables greater availability since the server does not have to maintain, update or communicate that session state. There is a drawback when the client need to send too much data to the server so it reduces the scope of network optimization and requires more bandwidth.

Cacheable: Every response should include whether the response is cacheable or not and for how much duration responses can be cached at the client side. Client will return the data from its cache for any subsequent request and there would be no need to send the request again to the server. A well-managed caching partially or completely eliminates some client–server interactions, further improving availability and performance. But sometime there are chances that user may receive stale data.

Client-Server: REST application should have a client-server architecture. A Client is someone who is requesting resources and are not concerned with data storage, which remains internal to each server, and server is someone who holds the resources and are not concerned with the user interface or user state. They can evolve independently. Client doesn’t need to know anything about business logic and server doesn’t need to know anything about frontend UI.

Layered system: An application architecture needs to be composed of multiple layers. Each layer doesn’t know any thing about any layer other than that of immediate layer and there can be lot of intermediate servers between client and the end server. Intermediary servers may improve system availability by enabling load-balancing and by providing shared caches.

Code on demand: It is an optional feature. According to this, servers can also provide executable code to the client. The examples of code on demand may include the compiled components such as Java applets and client-side scripts such as JavaScript.

<u>Introduction to API</u>

- **Explain the RESTFul Web Service?**

Restful Web Services is a lightweight, maintainable, and scalable service that is built on the REST architecture. Restful Web Service, expose API from your application in a secure, uniform, stateless manner to the calling client. The calling client can perform predefined operations using the Restful service. The underlying protocol for REST is HTTP. REST stands for REpresentational State Transfer.

- **Explain what is a "Resource" in REST?**

A resource in REST is a similar Object in Object Oriented Programming or is like an Entity in a Database. Once a resource is identified then its representation is to be decided using a standard format so that the server can send the resource in the above said format and client can understand the same format.

- **Which protocol is used by RESTful web services?**

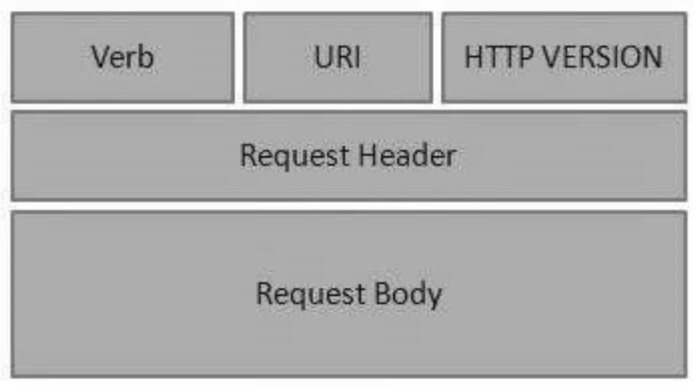
The REST architectural style constrains an architecture to a client/server architecture and is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol.

- **What is messaging in RESTful web services?**

RESTful Web Services make use of HTTP protocols as a medium of communication between client and server. A client sends a message in form of a HTTP Request and the server responds in the form of an HTTP Response. This technique is termed as Messaging. These messages contain message data and metadata i.e. information about message itself. Let us have a look on the HTTP Request and HTTP Response messages for HTTP 1.1.

Introduction to API

➢ **State the core components of an HTTP Request?**

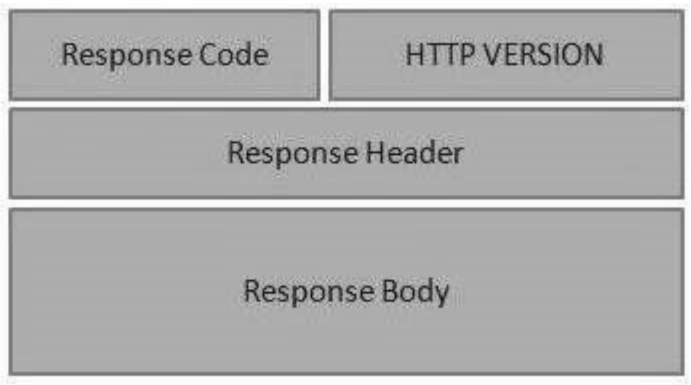


HTTP Request

An HTTP Request has five major parts −

- **Verb** − Indicates the HTTP methods such as GET, POST, DELETE, PUT, etc.
- **URI** − Uniform Resource Identifier (URI) to identify the resource on the server.
- **HTTP Version** − Indicates the HTTP version. For example, HTTP v1.1.
- **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client (or browser) type, format supported by the client, format of the message body, cache settings, etc.
- **Request Body** − Message content or Resource representation.

➢ **State the core components of an HTTP response?**



HTTP Response

Introduction to API

An HTTP Response has four major parts −

- **Status/Response Code** − Indicates the Server status for the requested resource. For example, 404 means resource not found and 200 means response is ok.
- **HTTP Version** − Indicates the HTTP version. For example HTTP v1.1.
- **Response Header** − Contains metadata for the HTTP Response message as keyvalue pairs. For example, content length, content type, response date, server type, etc.
- **Response Body** − Response message content or Resource representation.

➢ **What do you understand about payload in RESTFul web service?**

The request data which is present in the body part of every HTTP message is referred to as ‘Payload’. In Restful web service, the payload can only be passed to the recipient through the POST method.There is no limit of sending data as payload through the POST method but the only concern is that more data will consume more time and bandwidth. This may consume much of the user’s time also.

➢ **Explain the caching mechanism?**

Caching is a technique to speed up data lookups (data reading). Instead of reading the data directly from it source, which could be a database or another remote system, the data is read directly from a cache on the computer that needs the data. Here is an illustration of the caching principle:

➢ **List the main differences between SOAP and 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)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.

5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

➢ **Enlist advantages and disadvantages of 'Statelessness'.**

Advantages of Statelessness

Following are the benefits of statelessness in RESTful Web Services −

- Web services can treat each method request independently.
- Web services need not maintain the client's previous interactions. It simplifies the application design.
- As HTTP is itself a statelessness protocol, RESTful Web Services work seamlessly with the HTTP protocols.

Disadvantages of Statelessness

Following are the disadvantages of statelessness in RESTful Web Services −

- Web services need to get extra information in each request and then interpret to get the client's state in case the client interactions are to be taken care of.