**Explain what is REST and RESTFUL?**

REST represents REpresentational State Transfer; it is a relatively new aspect of writing web API.

RESTFUL is referred for web services written by applying REST architectural concept are called RESTful services, it focuses on system resources and how state of resource should be transported over HTTP protocol to different clients written in different language. In RESTFUL web service HTTP methods like GET, POST, PUT and DELETE can be used to perform CRUD operations.

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

HTTP methods supported by REST are:

* **GET:**It requests a resource at the request URL. It should not contain a request body as it will be discarded. Maybe it can be cached locally or on the server.
* **POST:**It submits information to the service for processing; it should typically return the modified or new resource
* **PUT:**At the request URL it update the resource
* **DELETE:**At the request URL it removes the resource
* **OPTIONS:**It indicates which techniques are supported
* **HEAD:**About the request URL it returns meta information

**Mention whether you can use GET request instead of PUT to create a resource?**

* No, you are not supposed to use PUT for GET. GET operations should only have view rights, while PUT resource is used for updating a data.

**Mention what are resources in a REST architecture?**

* Resources are identified by logical URLs; it is the key element of a RESTful design. Unlike, SOAP web services in REST, you view the product data as a resource and this resource should contain all the required information.

**Mention some key characteristics of REST?**

Some key characteristics of REST includes

* REST is stateless, therefore the SERVER has no state (or session data)
* With a well-applied REST API, the server could be restarted between two calls as every data is passed to the server
* Web service mostly uses POST method to make operations, whereas REST uses GET to access resources

**Mention what is the difference between PUT and POST?**

* “PUT” puts a file or resource at a particular URI and exactly at that URI. If there is already a file or resource at that URI, PUT changes that file or resource. If there is no resource or file there, PUT makes one
* POST sends data to a particular URI and expects the resource at that URI to deal with the request. The web server at this point can decide what to do with the data in the context of specified resource
* PUT is idempotent meaning, invoking it any number of times will not have an impact on resources.
* However, POST is not idempotent, meaning if you invoke POST multiple times it keeps creating more resources

**Mention what is the difference between SOAP and REST?**

| **SOAP** | **REST** |
| --- | --- |
| * SOAP is a protocol through which two computer communicates by sharing XML document. | * Rest is a service architecture and design for network-based software architectures. |
| * SOAP permits only XML | * REST supports many different data formats |
| * SOAP based reads cannot be cached | * REST reads can be cached |
| * SOAP is like custom desktop application, closely connected to the server | * A REST client is more like a browser; it knows how to standardized methods and an application has to fit inside it |
| * SOAP is slower than REST | * REST is faster than SOAP |
| * It runs on HTTP but envelopes the message | * It uses the HTTP headers to hold meta information |
| SOAP - Simple Object Access Protocol | REST - Representational State Transfer |
| SOAP is a protocol used to implement web services. | REST is an architectural design pattern for developing web services |
| SOAP cannot use REST as it is a protocol. | REST architecture can have SOAP protocol as part of the implementation. |
| SOAP specifies standards that are meant to be followed strictly. | REST defines standards but they need not be strictly followed. |
| SOAP client is more tightly coupled to the server which is similar to desktop applications having strict contracts. | The REST client is more flexible like a browser and does not depend on how the server is developed unless it follows the protocols required for establishing communication. |
| SOAP supports only XML transmission between the client and the server. | REST supports data of multiple formats like XML, JSON, MIME, Text, etc. |
| SOAP reads are not cacheable. | REST read requests can be cached. |
| SOAP uses service interfaces for exposing the resource logic. | REST uses URI to expose the resource logic. |
| SOAP is slower. | REST is faster. |
| Since SOAP is a protocol, it defines its own security measures. | REST only inherits the security measures based on what protocol it uses for the implementation. |
| SOAP is not commonly preferred, but they are used in cases which require stateful data transfer and more reliability. | REST is commonly preferred by developers these days as it provides more scalability and maintainability. |

**Mention which markup language can be used in restful web api?**

* JSON and XML are the two markup language that can be used in restful web api
* **What is URI?**
* **Uniform Resource Identifier** is the full form of URI which is used for identifying each resource of the REST architecture. URI is of the format:

<protocol>://<service-name>/<ResourceType>/<ResourceID>

There are 2 types of URI:

* **URN:**Uniform Resource Name identifies the resource by means of a name that is both unique and persistent.
  + URN doesn’t always specify where to locate the resource on the internet. They are used as templates that are used by other parsers to identify the resource.
  + These follow the urn scheme and usually prefixed with urn:. Examples include
    - urn:isbn:1234567890 is used for identification of book based on the ISBN number in a library application.
    - urn:mpeg:mpeg7:schema:2001 is the default namespace rules for metadata of MPEG-7 video.
  + Whenever a URN identifies a document, they are easily translated into a URL by using “resolver” after which the document can be downloaded.
* **URL:**Uniform Resource Locator has the information regarding fetching of a resource from its location.
  + Examples include:
    - http://abc.com/samplePage.html
    - ftp://sampleServer.com/sampleFile.zip
    - file:///home/interviewbit/sampleFile.txt
  + URLs start with a protocol (like ftp, http etc) and they have the information of the network hostname (sampleServer.com) and the path to the document(/samplePage.html). It can also have query parameters.

### What are the features of RESTful Web Services?

Every RESTful web service has the following features:

* The service is based on the Client-Server model.
* The service uses HTTP Protocol for fetching data/resources, query execution, or any other functions.
* The medium of communication between the client and server is called “Messaging”.
* Resources are accessible to the service by means of URIs.
* It follows the statelessness concept where the client request and response are not dependent on others and thereby provides total assurance of getting the required data.
* These services also use the concept of caching to minimize the server calls for the same type of repeated requests.
* These services can also use SOAP services as implementation protocol to REST architectural pattern

### What is the concept of statelessness in REST?

The REST architecture is designed in such a way that the client state is not maintained on the server. This is known as statelessness. The context is provided by the client to the server using which the server processes the client’s request. The session on the server is identified by the session identifier sent by the client.

### What are HTTP Status codes?

These are the standard codes that refer to the predefined status of the task at the server. Following are the status codes formats available:

* 1xx - represents informational responses
* 2xx - represents successful responses
* 3xx - represents redirects
* 4xx - represents client errors
* 5xx - represents server errors

Most commonly used status codes are:

* 200 - success/OK
* 201 - CREATED - used in POST or PUT methods.
* 304 - NOT MODIFIED - used in conditional GET requests to reduce the bandwidth use of the network. Here, the body of the response sent should be empty.
* 400 - BAD REQUEST - This can be due to validation errors or missing input data.
* 401- UNAUTHORIZED - This is returned when there is no valid authentication credentials sent along with the request.
* 403 - FORBIDDEN - sent when the user does not have access (or is forbidden) to the resource.
* 404 - NOT FOUND - Resource method is not available.
* 500 - INTERNAL SERVER ERROR - server threw some exceptions while running the method.
* 502 - BAD GATEWAY - Server was not able to get the response from another upstream server

### What are the HTTP Methods?

HTTP Methods are also known as HTTP Verbs. They form a major portion of uniform interface restriction followed by the REST that specifies what action has to be followed to get the requested resource. Below are some examples of HTTP Methods:

* GET: This is used for fetching details from the server and is basically a read-only operation.
* POST: This method is used for the creation of new resources on the server.
* PUT: This method is used to update the old/existing resource on the server or to replace the resource.
* DELETE: This method is used to delete the resource on the server.
* PATCH: This is used for modifying the resource on the server.
* OPTIONS: This fetches the list of supported options of resources present on the server.

The POST, GET, PUT, DELETE corresponds to the create, read, update, delete operations which are most commonly called **CRUD**

GET, HEAD, OPTIONS are safe and idempotent methods whereas PUT and DELETE methods are only idempotent. POST and PATCH methods are neither safe nor idempotent.

### Can you tell the disadvantages of RESTful web services?

The disadvantages are:

* As the services follow the idea of statelessness, it is not possible to maintain sessions. (Session simulation responsibility lies on the client-side to pass the session id)
* REST does not impose security restrictions inherently. It inherits the security measures of the protocols implementing it. Hence, care must be chosen to implement security measures like integrating SSL/TLS based authentications, etc.

### Define Messaging in terms of RESTful web services.

The technique of sending a message from the REST client to the REST server in the form of an HTTP request and the server responding back with the response as HTTP Response is called Messaging. The messages contained constitute the data and the metadata about the message.

### While creating URI for web services, what are the best practices that needs to be followed?

Below is the list of best practices that need to be considered with designing URI for web services:

* While defining resources, use plural nouns. Example: To identify user resource, use the name “users” for that resource.
* While using the long name for resources, use underscore or hyphen. Avoid using spaces between words. For example, to define authorized users resource, the name can be “authorized\_users” or “authorized-users”.
* The URI is case-insensitive, but as part of best practice, it is recommended to use lower case only.
* While developing URI, the backward compatibility must be maintained once it gets published. When the URI is updated, the older URI must be redirected to the new one using the HTTP status code 300.
* Use appropriate HTTP methods like GET, PUT, DELETE, PATCH, etc. It is not needed or recommended to use these method names in the URI. Example: To get user details of a particular ID, use /users/{id} instead of /getUser
* Use the technique of forward slashing to indicate the hierarchy between the resources and the collections. Example: To get the address of the user of a particular id, we can use: /users/{id}/address

### What are the best practices to develop RESTful web services?

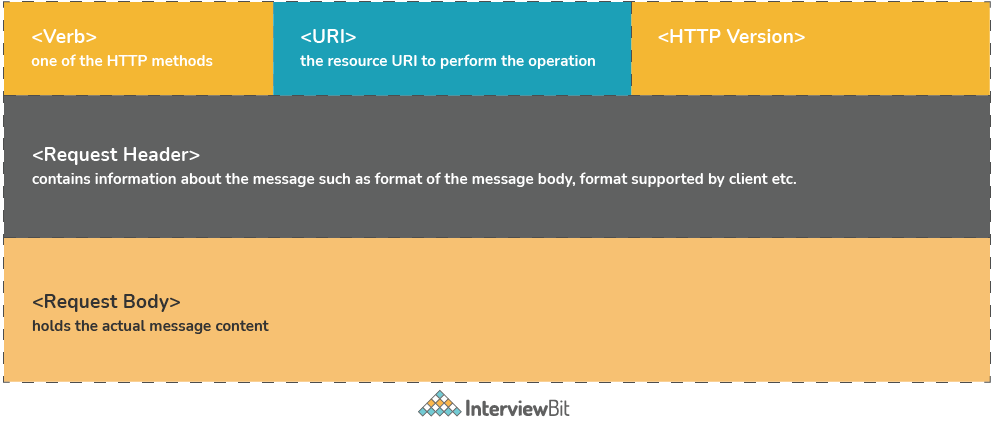
RESTful web services use REST API as means of implementation using the HTTP protocol. REST API is nothing but an application programming interface that follows REST architectural constraints such as statelessness, cacheability, maintainability, and scalability. It has become very popular among the developer community due to its simplicity. Hence, it is very important to develop safe and secure REST APIs that follow good conventions. Below are some best practices for developing REST APIs:

* Since REST supports multiple data formats, it is however good practice to develop REST APIs that accept and responds with JSON data format whenever possible. This is because a majority of the client and server technologies have inbuilt support to read and parse JSON objects with ease, thereby making JSON the standard object notation.
  + To ensure that the application responds using JSON data format, the response header should have Content-Type set to as application/JSON, this is because certain HTTP clients look at the value of this response header to parse the objects appropriately.
  + To ensure that the request sends the data in JSON format, again the Content-Type must be set to application/JSON on the request header.
* While naming the resource endpoints, ensure to use plural nouns and not verbs. The API endpoints should be clear, brief, easy to understand, and informative. Using verbs in the resource name doesn’t contribute much information because an HTTP request already has what the request is doing in its HTTP method/verb. An appropriate HTTP verb should be used to represent the task of the API endpoint.
  + Below are the most commonly used HTTP methods to define the verb:
    - GET - indicates get/retrieve the resource data
    - POST - indicates create new resource data
    - PUT - indicates update the existing resource data
    - DELETE - indicates remove the resource data
* To represent the hierarchy of resources, use the nesting in the naming convention of the endpoints. In case, you want to retrieve data of one object residing in another object, the endpoint should reflect this to communicate what is happening. For example, to get the address of an author, we can use the GET method for the URI /authors/:id/address'
  + Please ensure there are no more than 2 or 3 levels of nesting as the name of the URI can become too long and unwieldy.
* Error Handling should be done gracefully by returning appropriate error codes the application has encountered. REST has defined standard HTTP Status codes that can be sent along with the response based on the scenario.
  + Error codes should also be accompanied by appropriate error messages that can help the developers to take corrective actions. However, the message should not be too elaborate as well which can help the hacker to hack your application.
  + Common status codes are:
    - 400 - Bad Request – client-side error - failed input validation.
    - 401 - Unauthorized – The user is not authenticated and hence does not have authority to access the resource.
    - 403 - Forbidden – User is authenticated but is not authorized to access the resource.
    - 404 - Not Found – The resource is not found.
    - 500 - Internal server error – This is a very generic server-side error that is thrown when the server goes down. This shouldn’t be returned by the programmer explicitly.
    - 502 - Bad Gateway – Server did not receive a valid response from the upstream server.
    - 503 - Service Unavailable – Some unexpected things happened on the server such as system failure, overload, etc.
* While retrieving huge resource data, it is advisable to include filtering and pagination of the resources. This is because returning huge data all at once can slow down the system and reduce the application performance. Hence, filter some items reduces the data to some extent. Pagination of data is done to ensure only some results are sent at a time. Doing this can increase the server performance and reduce the burden of the server resources.
* Good security practices are a must while developing REST APIs. The client-server communication must be private due to the nature of data sensitivity. Hence, incorporating SSL/TLS becomes the most important step while developing APIs as they facilitate establishing secure communication. SSL certificates are easier to get and load on the server.
  + Apart from the secure channels, we need to ensure that not everyone should be able to access the resource. For example, normal users should not access the data of admins or another user. Hence, role-based access controls should be in place to make sure only the right set of users can access the right set of data.
* Since REST supports the feature of caching, we can use this feature to cache the data in order to improve the application performance. Caching is done to avoid querying the database for a request repeated times. Caching makes data retrieval fast. However, care must be taken to ensure that the cache has updated data and not outdated ones. Frequent cache update measures need to be incorporated. There are many cache providers like Redis that can assist in caching.
* API Versioning: Versioning needs to be done in case we are planning to make any changes with the existing endpoints. We do not want to break communication between our application and the apps that consume our application while we are working on the API release. The transition has to be seamless. Semantic versioning can be followed. For example, 3.0.1 represents 3rd major version with the first patch. Usually, in the API endpoints, we define /v1,/v2, etc at the beginning of the API path.

### Can you tell what constitutes the core components of HTTP Request?

In REST, any HTTP Request has 5 main components, they are:

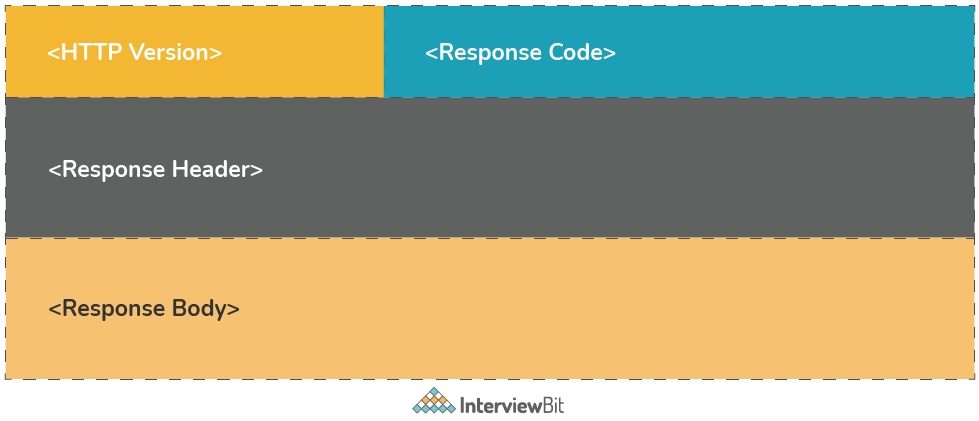
* Method/Verb − This part tells what methods the request operation represents. Methods like GET, PUT, POST, DELETE, etc are some examples.
* URI − This part is used for uniquely identifying the resources on the server.
* HTTP Version − This part indicates what version of HTTP protocol you are using. An example can be HTTP v1.1.
* Request Header − This part has the details of the request metadata such as client type, the content format supported, message format, cache settings, etc.
* Request Body − This part represents the actual message content to be sent to the server.



### What constitutes the core components of HTTP Response?

HTTP Response has 4 components:

* Response Status Code − This represents the server response status code for the requested resource. Example- 400 represents a client-side error, 200 represents a successful response.
* HTTP Version − Indicates the HTTP protocol version.
* Response Header − This part has the metadata of the response message. Data can describe what is the content length, content type, response date, what is server type, etc.
* Response Body − This part contains what is the actual resource/message returned from the server.



**Define Addressing in terms of RESTful Web Services.**

Addressing is the process of locating a single/multiple resources that are present on the server. This task is accomplished by making use of URI (Uniform Resource Identifier). The general format of URI is

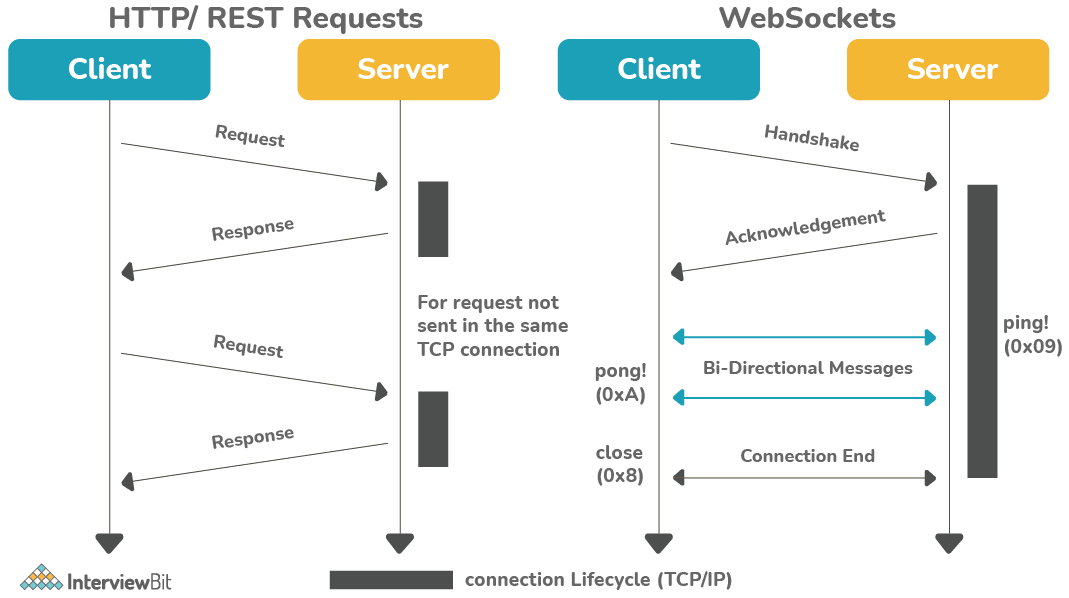
<protocol>://<application-name>/<type-of-resource>/<id-of-resource>

**What are the differences between PUT and POST in REST?**

| **PUT** | **POST** |
| --- | --- |
| PUT methods are used to request the server to store the enclosed entity in request. In case, the request does not exist, then new resource has to be created. If the resource exists, then the resource should get updated. | POST method is used to request the server to store the enclosed entity in the request as a new resource. |
| The URI should have a resource identifier. Example: PUT /users/{user-id} | The POST URI should indicate the collection of the resource. Example: POST /users |
| PUT methods are idempotent. | POST methods are not idempotent. |
| PUT is used when the client wants to modify a single resource that is part of the collection. If a part of the resource has to be updated, then PATCH needs to be used. | POST methods are used to add a new resource to the collection. |
| The responses are not cached here despite the idempotency. | Responses are not cacheable unless the response explicitly specifies Cache-Control fields in the header. |
| In general, PUT is used for UPDATE operations. | POST is used for CREATE operations. |

**We can develop webservices using web sockets as well as REST. What are the differences between these two?**

| **REST** | **Web Socket** |
| --- | --- |
| REST follows stateless architecture, meaning it won’t store any session-based data. | Web Socket APIs follow the stateful protocol as it necessitates session-based data storage. |
| The mode of communication is uni-directional. At a time, only the server or the client will communicate. | The communication is bi-directional, communication can be done by both client or server at a time. |
| REST is based on the Request-Response Model. | Web Socket follows the full-duplex model. |
| Every request will have sections like header, title, body, URL, etc. | Web sockets do not have any overhead and hence suited for real-time communication. |
| For every HTTP request, a new TCP connection is set up. | There will be only one TCP connection and then the client and server can start communicating. |
| REST web services support both vertical and horizontal scaling. | Web socket-based services only support vertical scaling. |
| REST depends on HTTP methods to get the response. | Web Sockets depend on the IP address and port number of the system to get a response. |
| Communication is slower here. | Message transmission happens very faster than REST API. |
| Memory/Buffers are not needed to store data here. | Memory is required to store data. |



### Can we implement transport layer security (TLS) in REST?

Yes, we can. TLS does the task of encrypting the communication between the REST client and the server and provides the means to authenticate the server to the client. It is used for secure communication as it is the successor of the Secure Socket Layer (SSL). HTTPS works well with both TLS and SSL thereby making it effective while implementing RESTful web services. One point to mention here is, the REST inherits the property of the protocol it implements. So security measures are dependent on the protocol REST implements

### What is Payload in terms of RESTful web services?

Payload refers to the data passes in the request body. It is not the same as the request parameters. The payload can be sent only in POST methods as part of the request body.

### Is it possible to send payload in the GET and DELETE methods?

No, the payload is not the same as the request parameters. Hence, it is not possible to send payload data in these method

### How does HTTP Basic Authentication work?

While implementing Basic Authentication as part of APIs, the user must provide the username and password which is then concatenated by the browser in the form of “username: password” and then perform base64 encoding on it. The encoded value is then sent as the value for the “Authorization” header on every HTTP request from the browser. Since the credentials are only encoded, it is advised to use this form when requests are sent over HTTPS as they are not secure and can be intercepted by anyone if secure protocols are not used.

### What is the difference between idempotent and safe HTTP methods?

* Safe methods are those that do not change any resources internally. These methods can be cached and can be retrieved without any effects on the resource.
* Idempotent methods are those methods that do not change the responses to the resources externally. They can be called multiple times without any change in the responses.

According to [restcookbook.com](https://restcookbook.com/), the following is the table that describes what methods are idempotent and what is safe.

| **HTTP Methods** | **Idempotent** | **Safe** |
| --- | --- | --- |
| OPTIONS | yes | yes |
| GET | yes | yes |
| HEAD | yes | yes |
| PUT | yes | no |
| POST | no | no |
| DELETE | yes | no |
| PATCH | no | no |

### What is the use of @RequestMapping?

* The annotation is used for mapping requests to specific handler classes or methods.
* In spring, all the incoming web request routing is handled by Dispatcher Servlet. When it gets the request, it determines which controller is meant for processing the request by means of request handlers. The Dispatcher Servlet scans all the classes annotated with @Controller. The process of routing requests depends on @RequestMapping annotations that are declared inside the controller classes and their methods.

### 39. What are the differences between the annotations @Controller and @RestController?

| **@Controller** | **@RestController** |
| --- | --- |
| Mostly used traditional Spring MVC service. | Represents RESTful web service in Spring. |
| It is mostly used in Spring MVC service where model data needs to rendered using view. | It is used in case of RESTful web service that returns object values bound to response body. |
| If response values need to be converted through HttpMessageConverters and sent via response object, extra annotation @ResponseBody needs to be used on the class or the method handlers. | The default behavior of the @RestController needs to be written on the response body because it is the combination of @Controller and @ResponseBody. |
| @Controller provides control and flexibility over how the response needs to be sent. | @RestController annotation has no such flexibility and writes all the results to the response body. |

### 40. What does the annotation @PathVariable do?

@PathVariable annotation is used for passing the parameter with the URL that is required to get the data. Spring MVC provides support for URL customization for data retrieval using @PathVariable annotation.