Prior to the year 2000, Application Program Interface, which is now commonly called **APIs**, were designed to be secure which were also very complex to develop, harder to develop, and even harder to maintain. They were not meant to be accessible. In 2000, a group of researchers headed by Roy Fielding came up with the idea of REST (REpresentational State Transfer) which brought out the true power and potential of APIs. The purpose of REST was to create standards for communication between two servers that are located anywhere in the world. They came up with various principles, constraints, and properties that constituted a resource-oriented architecture, client-server-based architecture, and interface uniformity that required no state preservation. These were easily cacheable, simple to develop, and could be implemented by means of Hypertext Transfer Protocol(HTTP). The rules seemed many, but the purpose was universal. The onset of REST became a true game-changer for the API landscape as the APIs developed under REST used less bandwidth, were simple to develop and the communication was supported via the internet thereby not requiring the servers to be connected physically.

RESTful APIs have also given birth to various trends like cloud computing and microservices-based architecture. They have made communication and computing over the internet seem easy. Hence, it is important for any developer to know what REST is, how it works, what are its features and how can you develop services in a secure manner to go with the trend. Many companies prefer developers with REST knowledge as they can help them develop products that are scalable, easy to maintain and make their products reach out to the world due to the power of the internet.

In the below section, we will see what are the most commonly asked questions on RESTful web services during an interview and some questions on the JAX-RS library, and some on RESTful web services implemented using the Spring MVC framework.

REST API Basic Interview Questions

1. What do you understand by RESTful Web Services?

RESTful web services are services that follow REST architecture. REST stands for Representational State Transfer and uses HTTP protocol (web protocol) for implementation. These services are lightweight, provide maintainability, scalability, support communication among multiple applications that are developed using different programming languages. They provide means of accessing resources present at server required for the client via the web browser by means of request headers, request body, response body, status codes, etc.

2. What is a REST Resource?

Every content in the REST architecture is considered a resource. The resource is analogous to the object in the object-oriented programming world. They can either be represented as text files, HTML pages, images, or any other dynamic data.

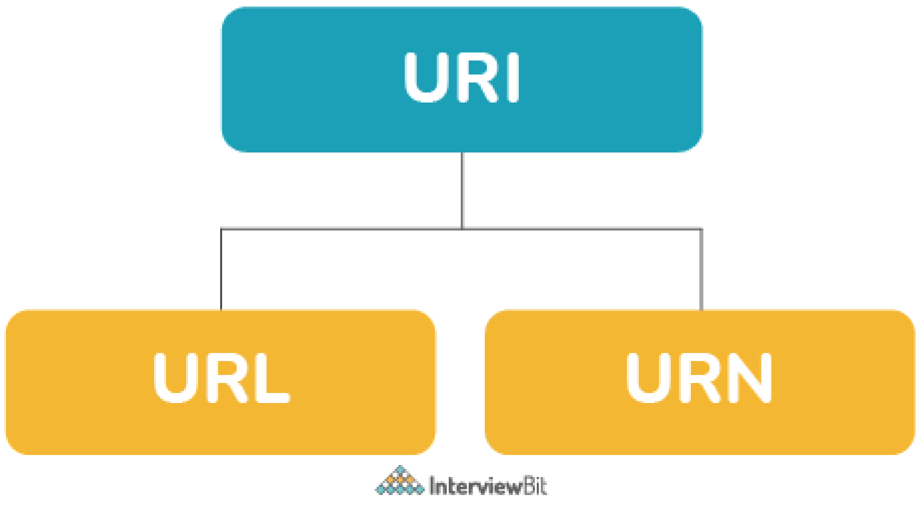
- The REST Server provides access to these resources whereas the REST client consumes (accesses and modifies) these resources. Every resource is identified globally by means of a URI.

3. 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:

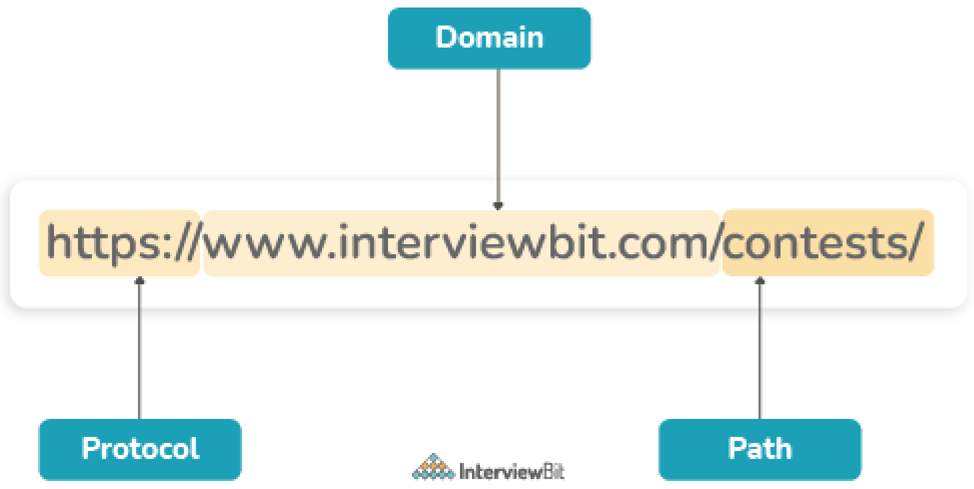
```
<protocol>://<service-name>/<ResourceType>/<ResourceID>
```

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.



You can download a PDF version of Rest Api Interview Questions.

Download PDF

---

4. What are the features of RESTful Web Services?

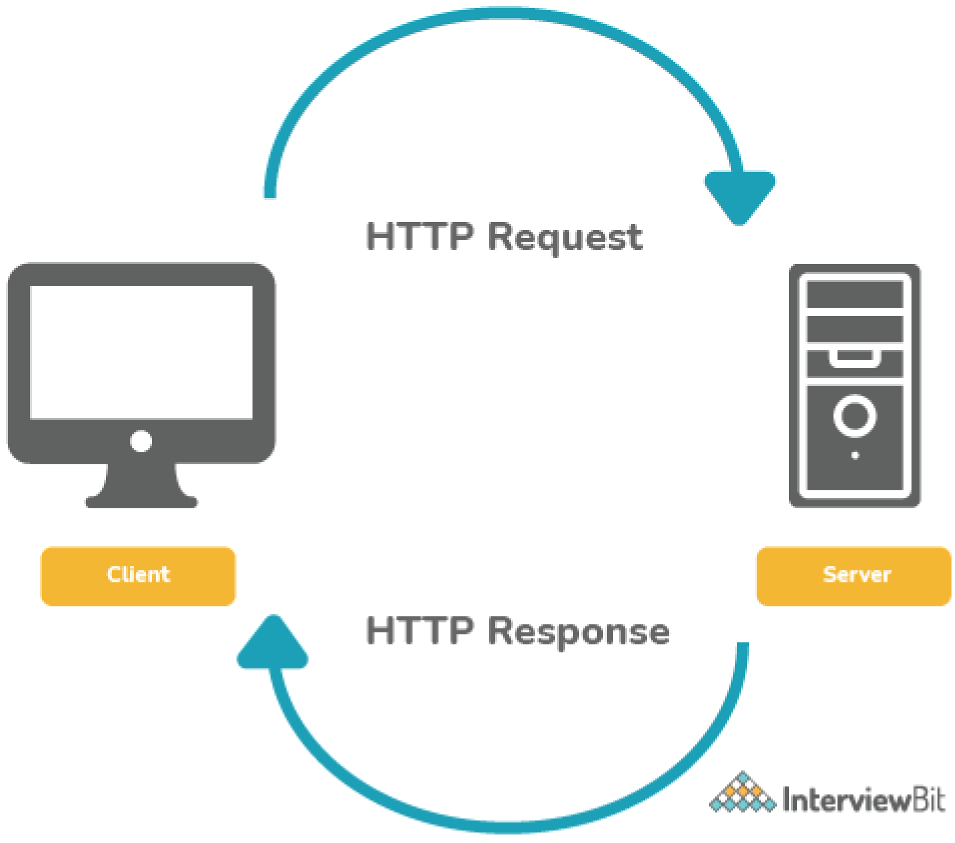
Every RESTful web service has the following features:

- 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.

5. 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.



6. What do you understand by JAX-RS?

As the name itself stands (JAX-RS= Java API for RESTful Web Services) is a Java-based specification defined by JEE for the implementation of RESTful services. The JAX-RS library makes usage of annotations from Java 5 onwards to simplify the process of web services development. The latest version is 3.0 which was released in June 2020. This specification also provides necessary support to create REST clients.

7. 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.

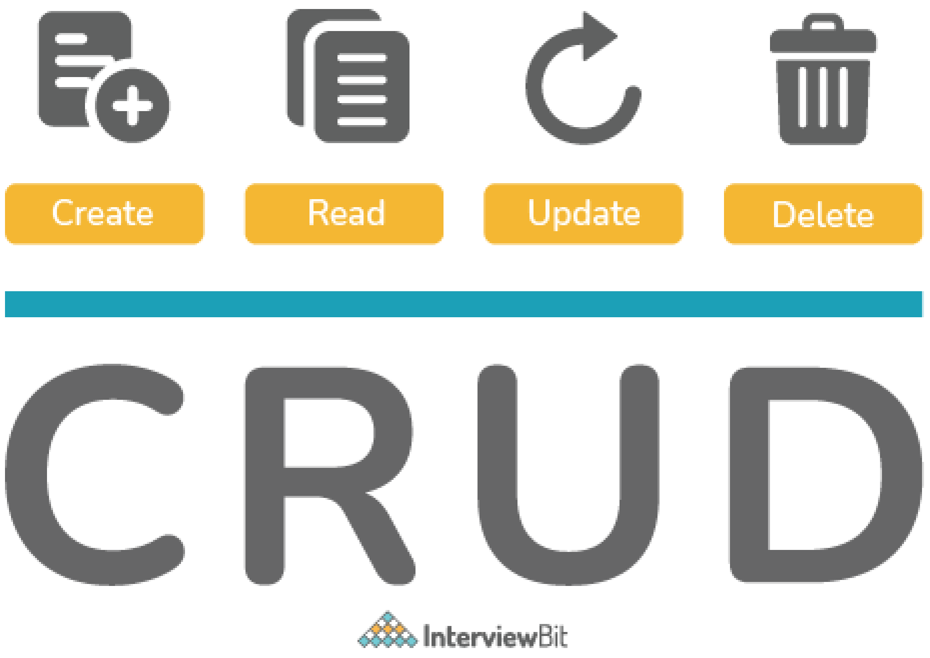
8. What are the HTTP Methods?

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 Operations**.



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.

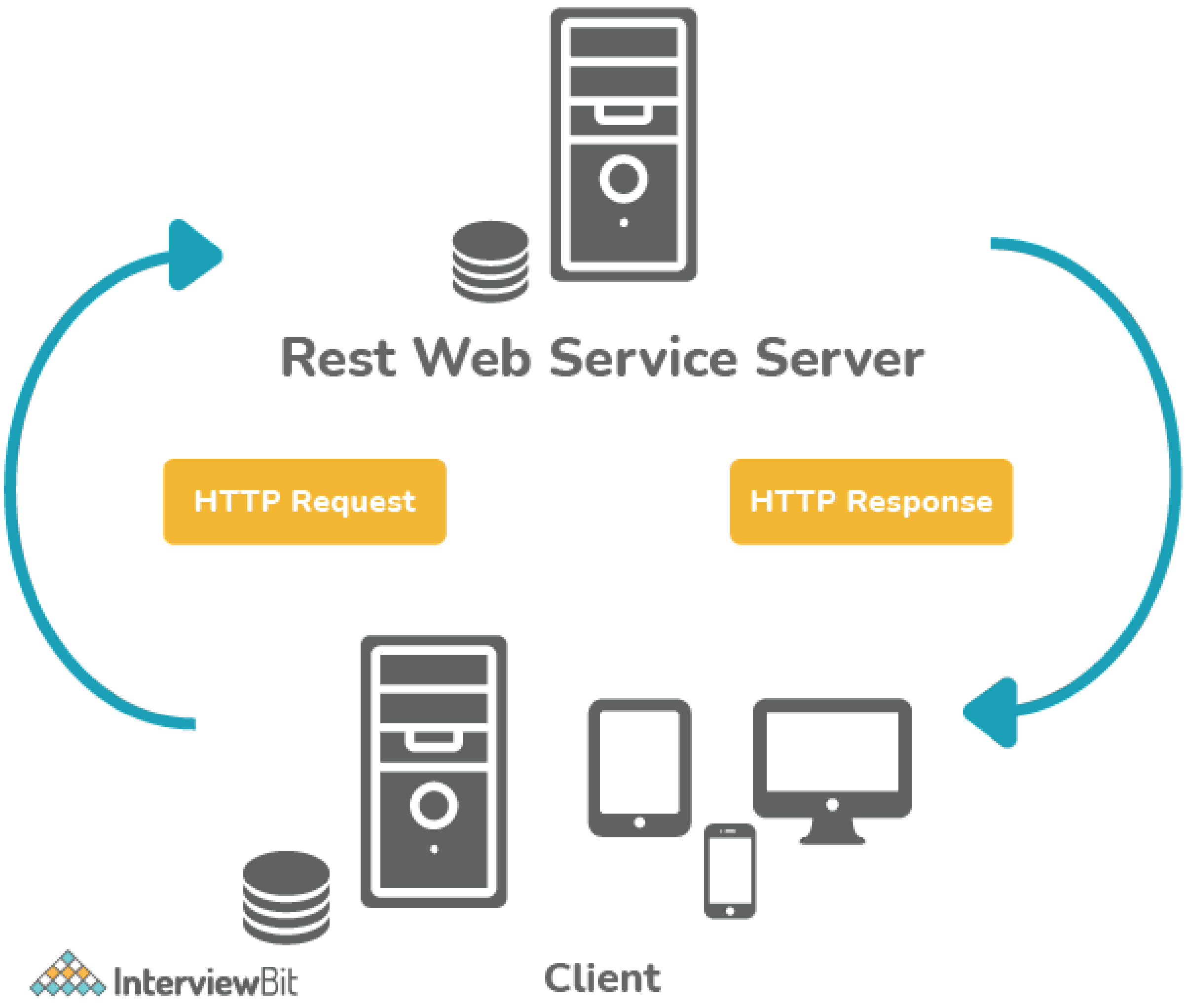
9. Can you tell the disadvantages of RESTful web services?

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.

10. 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.



REST API Experienced Interview Questions

11. Differentiate between SOAP and REST?

SOAP	REST
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	REST
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.

12. 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`

13. 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.

14. What are Idempotent methods? How is it relevant in RESTful web services domain?

The meaning of idempotent is that even after calling a single request multiple times, the outcome of the request should be the same. While designing REST APIs, we need to keep in mind to develop idempotent APIs. This is because the consumers can write client-side code which can result in duplicate requests intentionally or not. Hence, fault-tolerant APIs need to be designed so that they do not result in erroneous responses.

- Idempotent methods ensure that the responses to a request if called once or ten times or more than that remain the same. This is equivalent to adding any number with 0.
- REST provides idempotent methods automatically. GET, PUT, DELETE, HEAD, OPTIONS, and TRACE are the idempotent HTTP methods. POST is not idempotent.
    - POST is not idempotent because POST APIs are usually used for creating a new resource on the server. While calling POST methods N times, there will be N new resources. This does not result in the same outcome at a time.
    - Methods like GET, OPTIONS, TRACE, and HEAD are idempotent because they do not change the state of resources on the server. They are meant for resource retrieval whenever called. They do not result in write operations on the server thereby making it idempotent.
    - PUT methods are generally used for updating the state of resources. If you call PUT methods N times, the first request updates the resource and the subsequent requests will be overwriting the same resource again and again without changing anything. Hence, PUT methods are idempotent.
    - DELETE methods are said to be idempotent because when calling them for N times, the first request results in successful deletion (Status Code 200), and the next subsequent requests result in nothing - Status Code 204. The response is different, but there is no change of resources on the server-side.
        - However, if you are attempting to delete the resource present, at last, every time you hit the API, such as the request `DELETE /user/last` which deletes the last user record, then calling the request N times would delete N resources on the server. This does not make DELETE idempotent. In such cases, as part of good practices, it is advisable to use POST requests.

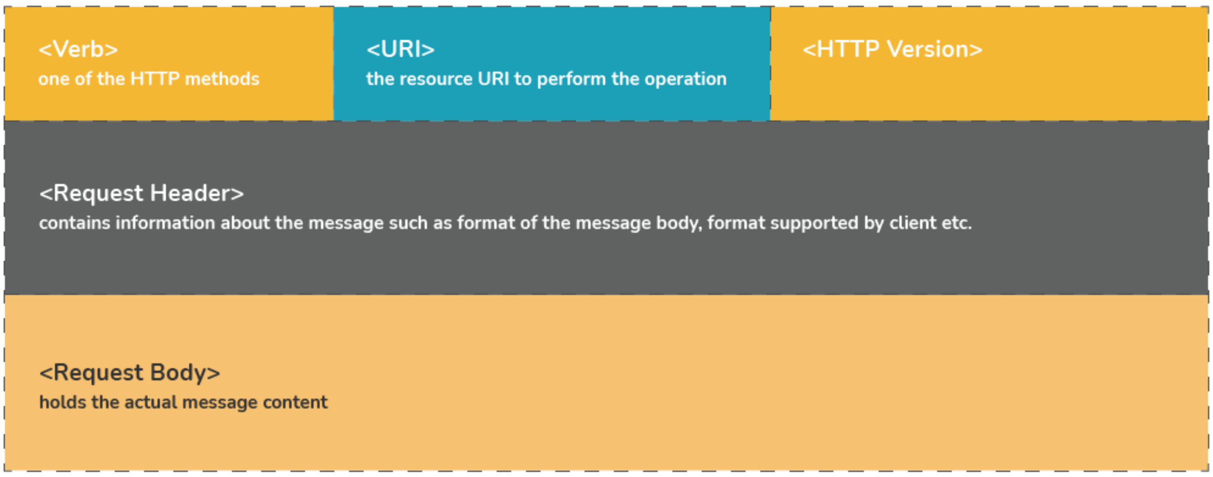
15. What are the differences between REST and AJAX?

REST	AJAX
REST- Representational State Transfer	AJAX - Asynchronous javascript and XML
REST has a URI for accessing resources by means of a request-response pattern.	AJAX uses XMLHttpRequest object to send requests to the server and the response is interpreted by the Javascript code dynamically.
REST is an architectural pattern for developing client-server communication systems.	AJAX is used for dynamic updation of UI without the need to reload the page.
REST requires the interaction between client and server.	AJAX supports asynchronous requests thereby eliminating the necessity of constant client-server interaction.

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

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

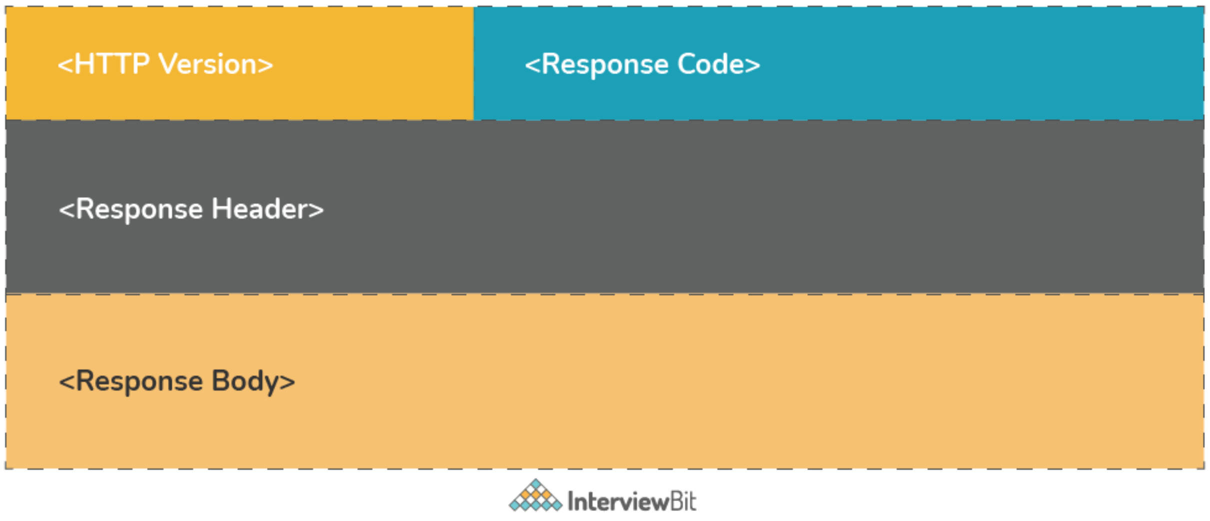
- Method/Verb − This part tells what methods the request operation represents. Methods like GET, PUT, POST, DELETE, etc are some examples.
- URI − This part is used for uniquely identifying the resources on the server.
- HTTP Version − This part indicates what version of HTTP protocol you are using. 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.


<Verb>
one of the HTTP methods
<URI>
the resource URI to perform the operation
<HTTP Version>
<Request Header>
contains information about the message such as format of the message body, format supported by client etc.
<Request Body>
holds the actual message content

17. 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.



18. 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>
```

19. 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.

PUT	POST
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.

20. What makes REST services to be easily scalable?

REST services follow the concept of statelessness which essentially means no storing of any data across the requests on the server. This makes it easier to scale horizontally because the servers need not communicate much with each other while serving requests.

21. Based on what factors, you can decide which type of web services you need to use - SOAP or REST?

REST services have gained popularity due to the nature of simplicity, scalability, faster speed, improved performance, and multiple data format support. But, SOAP has its own advantages too. Developers use SOAP where the services require advanced security and reliability.

Following are the questions you need to ask to help you decide which service can be used:

- Do you want to expose resource data or business logic?
    - SOAP is commonly used for exposing business logic and REST for exposing data.
- Does the client require a formal strict contract?
    - If yes, SOAP provides strict contracts by using WSDL. Hence, SOAP is preferred here.
- Does your service require support for multiple formats of data?

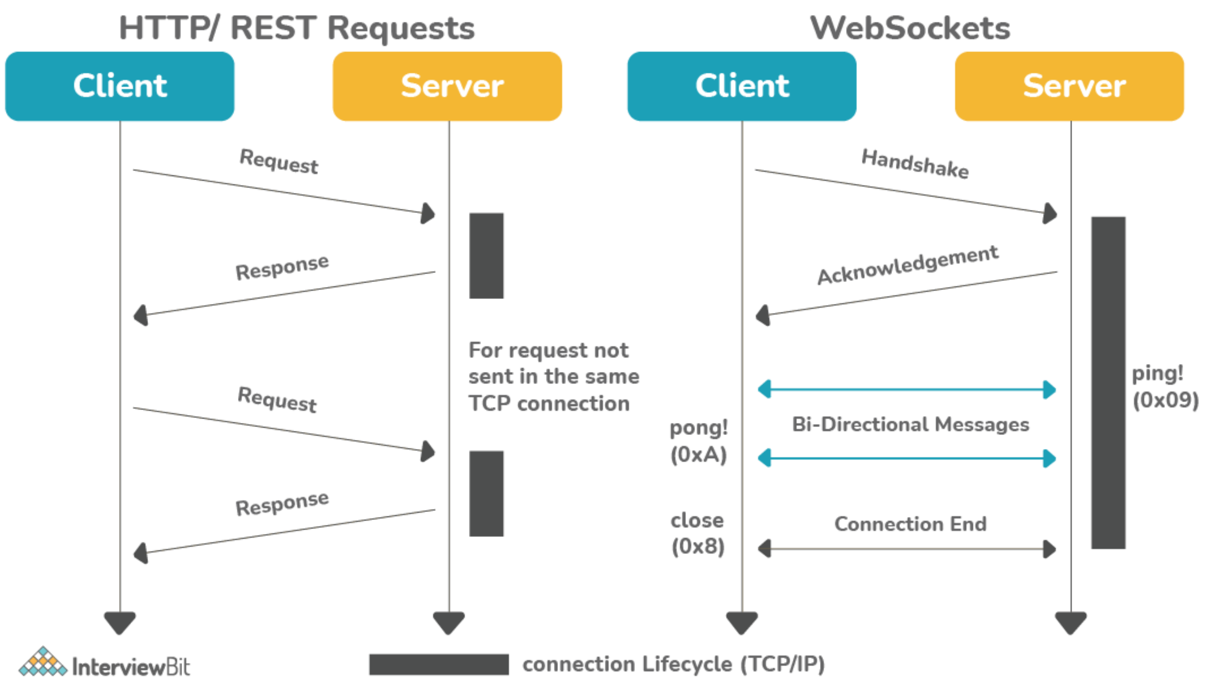
    - If yes, REST supports multiple data formats which is why it is preferred in this case.
- Does your service require AJAX call support?
    - If yes, REST can be used as it provides the XMLHttpRequest.
- Does your service require both synchronous and asynchronous requests?
    - SOAP has support for both sync/async operations.
    - REST only supports synchronous calls.
- Does your service require statelessness?
    - If yes, REST is suitable. If no, SOAP is preferred.
- Does your service require a high-security level?
    - If yes, SOAP is preferred. REST inherits the security property based on the underlying implementation of the protocol. Hence, it can't be preferred at all times.
- Does your service require support for transactions?
    - If yes, SOAP is preferred as it is good in providing advanced support for transaction management.
- What is the bandwidth/resource required?
    - SOAP involves a lot of overhead while sending and receiving XML data, hence it consumes a lot of bandwidth.
    - REST makes use of less bandwidth for data transmission.
- Do you want services that are easy to develop, test, and maintain frequently?
    - REST is known for simplicity, hence it is preferred.

22. 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 Socket
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.

The request flow difference between the REST and Web Socket is shown below:



23. 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.

24. Should we make the resources thread safe explicitly if they are made to share across multiple clients?

There is no need to explicitly making the resources thread-safe because, upon every request, new resource instances are created which makes them thread-safe by default.

25. 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.

26. 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 methods.

27. How can you test RESTful Web Services?

RESTful web services can be tested using various tools like Postman, Swagger, etc. Postman provides a lot of features like sending requests to endpoints and show the response which can be converted to JSON or XML and also provides features to inspect request parameters like headers, query parameters, and also the response headers. Swagger also provides similar features like Postman and it provides the facility of documentation of the endpoints too. We can also use tools like Jmeter for performance and load testing of APIs.

28. What is the maximum payload size that can be sent in POST methods?

Theoretically, there is no restriction on the size of the payload that can be sent. But one must remember that the greater the size of the payload, the larger would be the bandwidth consumption and time taken to process the request that can impact the server performance.

29. 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.

30. 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, 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

JAX-RS Interview Questions

31. What are the key features provided by JAX-RS API in Java EE?

JAX-RS stands for Java API for RESTful Web services. They are nothing but a set of Java-based APIs that are provided in the Java EE which is useful in the implementation and development of RESTful web services.

Features of JAX-RS are:

- **POJO-based**: The APIs in the JAX-RS is based on a certain set of annotations, classes, and interfaces that are used with POJO (Plain Old Java Object) to expose the services as web services.
- **HTTP-based:** The JAX-RS APIs are designed using HTTP as their base protocol. They support the HTTP usage patterns and they provide the corresponding mapping between the HTTP actions and the API classes.
- **Format Independent**: They can be used to work with a wide range of data types that are supported by the HTTP body content.

- **Container Independent**: The APIs can be deployed in the Java EE container or a servlet container such as Tomcat or they can also be plugged into JAX-WS (Java API for XML-based web services) providers.

32. Define RESTful Root Resource Classes in the JAX-RS API?

- A resource class is nothing but a Java class that uses JAX-RS provided annotations for implementing web resources.
- They are the POJOs that are annotated either with @Path or have at least one method annotated with @Path, @GET, @POST, @DELETE, @PUT, etc.

Example:

```
import javax.ws.rs.Path;
/**
* InterviewBitService is a root resource class that is exposed at
'resource_service' path
*/
@Path('resource_service')
public class InterviewBitService {
   // Defined methods
}
```

33. What do you understand by request method designator annotations?

They are the runtime annotations in the JAX-RS library that are applied to Java methods. They correspond to the HTTP request methods that the clients want to make. They are @GET, @POST, @PUT, @DELETE, @HEAD.

Usage Example:

```
import javax.ws.rs.Path;
/**
* InterviewBitService is a root resource class that is exposed at
'resource_service' path
*/
@Path('resource_service')
public class InterviewBitService {
    @GET
    public String getRESTQuestions() {
        // some operations
    }
}
```

34. How can the JAX-RS applications be configured?

JAX-RS applications have the root resource classes packaged in a war file. There are 2 means of configuring JAX-RS applications.

1. Use @ApplicationPath annotation in a subclass of `javax.ws.rs.core.Application` that is packaged in the WAR file.
2. Use the <servlet-mapping> tag inside the web.xml of the WAR. web.xml is the deployment descriptor of the application where the mappings to the servlets can be defined.

35. Is it possible to make asynchronous requests in JAX-RS?

Yes. the JAX-RS Client API provides a method called `Invocation.Builder.async()` that is used for constructing client requests that need to be executed asynchronously. Invoking a request asynchronously does the task of returning the control to the caller by returning with datatype `java.util.concurrent.Future` whose type is set to return the service call type. Future objects are used because they have the required methods to check whether the asynchronous calls have been completed and if yes, then retrieve the responses. They also provide the flexibility to cancel the request invocations and also check if the cancellation has been successful.

Let us understand this with the help of a random example. We know that the `Future` interface from the `java.util.concurrent` has the below functions available:

```
package java.util.concurrent;
public interface Future<V> {
   // informs the executor to stop the thread execution
   boolean cancel(boolean mayInterruptIfRunning);

   // indicates whether the Future was cancelled or not
   boolean isCancelled();

   // indicates if the executor has completed the task
   boolean isDone();

   // gets the actual result from the process.
   // This blocks the program execution until the result is ready.
   V get() throws InterruptedException, ExecutionException;

   // also gets actual result from the process but it throws
   // the TimeoutException in case the result is not obtained before
specified timeout
   V get(long timeout, TimeUnit unit)
       throws InterruptedException, ExecutionException, TimeoutException;
}
```

Let us consider we have this function below which is used for processing 2 Ids parallelly.

```
public void processIds(String userId1, String questionId){
   Client client = ClientBuilder.newClient();
   Future<Response> futureResponse1 =
client.target("http://interviewbitserver.com/users/"+userId).request().asy
nc().get();
```

```
   Future<Order> futureResponse2 =
client.target("http://interviewbitserver.com/questions/"+questionId).reque
st().async().get(Question.class);

   // block the process until complete
   Response response1 = futureResponse1.get();
   User userObject = response1.readEntity(User.class);
   //Do processing of userObject

   // Wait for 2 seconds before fetching record
   try {
      Question question = futureResponse2.get(2, TimeUnit.SECONDS);
      //Do Processing of question
   } catch (TimeoutException timeoutException ) {
       //handle exceptions
   }
   return;
}
```

In the above example, we see that there are 2 separate requests getting executed parallelly. For the first future object, we await the `javax.ws.rs.core.Response` indefinitely using the get() method until we get the response. For the second future object, we wait for the response only for 2 seconds and if we do not get within 2 seconds, then the get() method throws TimeoutException. We can also use the isDone() method or isCancelled() method to find out whether the executors have completed or cancelled.

36. List the key annotations that are present in the JAX-RS API?

- @Path - This specifies the relative URI path to the REST resource.
- @GET - This is a request method designator which is corresponding to the HTTP GET requests. They process GET requests.
- @POST - This is a request method designator which is corresponding to the HTTP POST requests. They process POST requests.
- @PUT - This is a request method designator which is corresponding to the HTTP PUT requests. They process PUT requests.
- @DELETE - This is a request method designator which is corresponding to the HTTP DELETE requests. They process DELETE requests.
- @HEAD - This is a request method designator which is corresponding to the HTTP HEAD requests. They process HEAD requests.
- @PathParam - This is the URI path parameter that helps developers to extract the parameters from the URI and use them in the resource class/methods.
- @QueryParam - This is the URI query parameter that helps developers extract the query parameters from the URI and use them in the resource class/methods.
- @Produces - This specifies what MIME media types of the resource representations are produced and sent to the client as a response.

- @Consumes - This specifies which MIME media types of the resource representations are accepted or consumed by the server from the client.

Spring RESTful Web Services Interview Questions

37. Define RestTemplate in Spring.

The RestTemplate is the main class meant for the client-side access for Spring-based RESTful services. The communication to the server is accomplished using the REST constraints. This is similar to other template classes such as JdbcTemplate, HibernateTemplate, etc provided by Spring. The RestTemplate provides high-level implementation details for the HTTP Methods like GET, POST, PUT, etc, and gives the methods to communicate using the URI template, URI path params, request/response types, request object, etc as part of arguments.

- Commonly used annotations like `@GetMapping`, `@PostMapping`, `@PutMapping`, etc are provided by this class from Spring 4.3. Prior to that, Spring provided (and still provides) `@RequestMapping` annotation to indicate what methods were being used.

38. 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	The default behavior of the @RestController needs to be written on the response body because it is

@Controller	**@RestController**
@ResponseBody needs to be used on the class or the method handlers.	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.

41. Is it necessary to keep Spring MVC in the classpath for developing RESTful web services?

Yes. **Spring MVC** needs to be on the classpath of the application while developing RESTful web services using Spring. This is because, the Spring MVC provides the necessary annotations like @RestController, @RequestBody, @PathVariable, etc. Hence the spring-mvc.jar needs to be on the classpath or the corresponding Maven entry in the pom.xml.

42. Define HttpMessageConverter in terms of Spring REST?

HttpMessageConverter is a strategic interface that specified a converter for conversion between HTTP Requests and responses. Spring REST uses the HttpMessageConverter for converting responses to various data formats like JSON, XML, etc. Spring makes use of the “Accept” header for determining the type of content the client expects. Based on this, Spring would find the registered message converter interface that is capable of this conversion.

Conclusion

43. Conclusion

We have seen what are the most commonly asked questions on RESTful web services during an interview. REST APIs have become a very important tool in the software industry. Developing RESTful web services that are scalable and easily maintainable is considered an art. As the industry trends increase, the REST architecture would become more concrete and the demand for developers who know the development of RESTful web services would increase steadily.

References:

To learn more about REST, you can refer to the below 2 links:
https://restcookbook.com/
https://www.restapitutorial.com/

REST API MCQ

1.

What protocol does REST follow?

○ FTP
○ HTTP
○ SFTP
○ FTPS

2.

HTTP Code 200 represents which among the following?

○ Success
○ Completed
○ Failure
○ Warning

3.

Which of the following options are true for REST Services?

○ Each resource can be identified by multiple URIs.
○ REST supports only JSON representation.
○ In REST Services, the client gets access to resources and the server provides access to them.
○ All of the above.

4.

What method should be used to obtain a list of supported operations in REST services?

○ GET
○ DELETE
○ HEAD
○ OPTION

5.

Which of the below does the task of binding the parameters passed to the HTTP method to HTTP Header while using JAX RS API?

- @PathParam
- @QueryParam
- @HeaderParam
- @PathVariable

6.

What category does the 5xx HTTP code belong to?

- Redirection
- Warning
- Client Error
- Server Error

7.

Is it possible to maintain sessions in REST on the server-side?

- No
- Yes
- Depends on situation
- I don’t know

8.

Which among the below directives belonging to the Cache-Control header of HTTP response provide information to the server that the resources have to be revalidated if max-age has crossed?

- no-store
- must-revalidate
- no-cache
- None of the above

9.

What category do 1xx HTTP status codes belong to?

- Server Error
- Redirection
- Client Error
- Informational

10.

What does the status code 302 represent?

- User can select among multiple links and go to different page.
- Not Modified
- The resource requested has been found and moved temporarily to new URL location.
- The page requested is available only by means of proxy address given in the response.

11.

What constraint is not a strict requirement for a service to be called a RESTful web service?

- Uniform Interface
- Code on Demand
- Stateless
- Client-Server
- All of the above

12.

Which component of HTTP response has the metadata for stating the type of response message is in the form of key-value pairs?

- Status Code
- Response Body
- HTTP Version
- Response Header

13.

Which directive of the Cache-control header in the HTTP Response tells that the resource cannot be cached?

- public
- max-age
- no-cache/no-store
- private

- Blog
- Community
- About Us
- FAQ
- Contact Us
- Terms
- Privacy Policy
-
-
-

-
  - **Practice Questions**
  - Programming
  - Scripting
  - System Design
  - Databases
  - Puzzle
-
  - **Fast Track Courses**
  - Python
  - Java
  - C++
  - Javascript

- **Online Interviewbit Compilers**
- Online C Compiler
- Online C++ Compiler
- Online Java Compiler
- Online Javascript Compiler
- Online Python Compiler

- **Interview Preparation**
- java interview questions
- javascript interview questions
- sql interview questions

- python cheat sheet
- java mcq
- python interview questions
- angular interview questions
- spring boot interview questions
- dbms interview questions
- oops interview questions
- dbms mcq
- selenium interview questions
- javascript cheat sheet
- c++ cheat sheet
- hr interview questions
- html interview questions
- c interview questions
- c++ mcq
- data structure mcq
- c++ interview questions
- machine learning mcq
- c# interview questions
- c programming mcq
- .net interview questions
- react interview questions
- node js interview questions
- php interview questions
- css interview questions
- java 8 interview questions
- python mcq
- mvc interview questions
- git cheat sheet
- power bi interview questions
- javascript mcq
- microservices interview questions
- mysql interview questions
- aws interview questions
- html cheat sheet
- api testing interview questions
- networking interview questions
- data structure interview questions
- hibernate interview questions
- asp.net interview questions
- spring interview questions
- linux interview questions
- rest api interview questions
- web api interview questions
- software testing interview questions
- machine learning interview questions
- jenkins interview questions
- devops interview questions
- salesforce interview questions
- docker interview questions

- kubernetes interview questions
- tableau interview questions
- os interview questions
- scrum master interview questions
- django interview questions
- laravel interview questions
- jquery interview questions
- data science interview questions
- android interview questions
- java interview questions for 5 years experience
- asp.net mvc interview questions
- automation testing interview questions
- mongodb interview questions
- informatica interview questions
- system design interview questions
- oracle interview questions
- data analyst interview questions
- design patterns interview questions
- big data interview questions
- sap interview questions
- tcs interview questions
- accenture interview questions
- infosys interview questions
- capgemini interview questions
- sql cheat sheet
- operating system mcq
- wipro interview questions
- amazon interview questions
- cognizant interview questions
- google interview questions
- deloitte interview questions
- zoho interview questions
- hcl interview questions
- servicenow interview questions
- sap abap interview questions
- mindtree interview questions
- tech mahindra interview questions
- goldman sachs interview questions
- apisero interview questions
- dxc interview questions
- genpact interview questions
- flipkart interview questions
- adobe interview questions
- accolite interview questions
- cisco interview questions
- facebook interview questions
- siemens interview questions
- linkedin interview questions
- directi interview questions
- paytm interview questions

- yahoo interview questions
- markdown cheat sheet
- linux commands cheat sheet
- java cheat sheet
- bootstrap cheat sheet
- software engineering mcq
- computer network mcq
- html mcq
- control system mcq
- web technology mcq
- artificial intelligence mcq
- software testing mcq
- cyber security mcq
- iot mcq
- sql mcq
- compiler design mcq
- View All