Restful API

Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, termed RESTful Web services (RWS), provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations.

1. Features/Constraints

Client-server architecture: separating the user interface (front) concerns from the data storage (back) concerns improves the portability of the user interface across multiple platforms.
Statelessness: no client context being stored on the server between requests. Each request from any client contains all the information necessary to service the request. In other word, each request/response is independent.
Cache ability: clients and intermediaries can cache responses.
Layered system: A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers can improve system scalability by enabling load balancing and by providing shared caches. They can also enforce security policies.
Uniform interface: fundamental to the design of any RESTful system.[3] It simplifies and decouples the architecture, which enables each part to evolve independently.

2. Relationship between URI and HTTP methods

URI	Collection Resource: api.example.com/collection	Member Resource: api.example.com/collection/item3
GET	*Retrieve* the URIs of the member resources of the collection resource in the response body.	*Retrieve* a representation of the member resource in the response body.
POST	*Create* a member resource in the collection resource using the instructions in the request body. The URI of the created member resource is *automatically assigned* and returned in the response *Location* header field.	*Create* a member resource in the member resource using the instructions in the request body. The URI of the created member resource is *automatically assigned* and returned in the response *Location* header field.
PUT	*Replace* all the representations of the member resources of the collection resource with the representation in the request body, or *create* the collection resource if it does not exist.	*Replace* all the representations of the member resource, or *create*the member resource if it does not exist, with the representation in the request body.
DELETE	*Delete* all the representations of the member resources of the collection resource.	*Delete* all the representations of the member resource.
PATCH	*Update* all the representations of the member resources of the collection resource using the instructions in the request body, or *may create* the collection resource if it does not exist.	*Update* all the representations of the member resource, or *may create* the member resource if it does not exist, using the instructions in the request body.

The GET method is safe, meaning that applying it to a resource does not result in a state change of the resource (read-only semantics). The GET, PUT and DELETE methods are idempotent, meaning that applying them multiple times to a resource result in the same state change of the

resource as applying them once, though the response might differ. The GET and POST methods are cacheable, meaning that responses to them are allowed to be stored for future reuse.

Unlike SOAP-based Web services, there is no "official" standard for RESTful Web APIs. This is because REST is an architectural style, while SOAP is a protocol. REST is not a standard in itself, but RESTful implementations make use of standards, such as HTTP, URI, JSON, and XML. Many developers also describe their APIs as being RESTful, even though these APIs actually don't fulfill all of the architectural constraints described above (especially the uniform interface constraint).

3. Q&A

List the advantages and disadvantages of statelessness of Restful.

Advantages:

- Every method required for communication is identified as an independent method i.e. there are no dependencies to other methods.
- Any previous communication with the client and server is not maintained and thus the whole process is very much simplified.
- If any information or metadata used earlier in required in another method, then the client sends again that information with HTTP request.
- HTTP protocol and REST web service, both shares the feature of statelessness.

Disadvantages:

- In every HTTP request from the client, the availability of some information regarding the client state is required by the web service.

Understanding of Cache

Caching is the process in which server response is stored so that a cached copy can be used when required and there is no need of generating the same response again. This process not only reduces the server load but in turn increase the scalability and performance of the server. Only the client is able to cache the response and that too for a limited period of time.

Mentioned below are the header of the resources and their brief description so that they can be identified for the caching process:

- Time and Date of resource creation
- Time and date of resource modification that usually stores the last detail.
- Cache control header
- Time and date at which the cached resource will expire.
- The age which determines the time from when the resource has been fetched.

Do you hear about JAX-RX?

JAX-RS is defined as the Java API for RESTful web service. Among multiple libraries and framework, this is considered as the most suitable Java programming language based API which supports RESTful web service.

Some of the implementations of JAX-RS are:

Jersey(mot popular), RESTEasy, Apache CFX, Play

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

SOAP vs RESTful

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

Reference:

1. https://en.wikipedia.org/wiki/Representational_state_transfer
2. https://www.softwaretestinghelp.com/restful-web-services-interview-question/