NJC LABS ACCELERATOR PROGRAM

SESSION 3 NOTES

HTTP Fundamentals

1. **What are the basic Features of HTTP?**
   The basic features of HTTP are:
   - Media independent - Any kind of media content can be sent through HTTP, as long as both client and server are able to handle the data.
   - Connectionless – An HTTP client i.e. browser initiates a connection with a server to send an HTTP request and disconnects after sending the request.
   - Stateless – The client and server know each only in the current state. Neither the client nor the server can retain the information about different requests across web pages.

2. **What are request methods in HTTP?**
   The HTTP Request method indicates the method that is to be performed on the resource identified by the Requested URI (Uniform Resource Identifier). HTTP Request methods are case sensitive and should be used in uppercase.
   The HTTP request methods are:

HTTP method	Description
GET	Requests a representation of the specified resource. Used only to retrieve data.
HEAD	Asks for a response identical to GET method, but without a response body.
POST	Used to submit an entity to the specified resource, often causing a change in state of the server.
PUT	Replaces all current representations of the target resource with the request payload or creates the resource if it does not exist.
DELETE	Deletes the specified resource
PATCH	Used to partially update a resource with the resources submitted

3. **What are the differences between GET and POST methods?**

GET	POST
Only limited amount of data can be sent because data is sent in header.	Any amount of data can be sent as it is sent in body.
Not secured, because data is exposed in URL bar.	Secured because data is not exposed in URL and encrypted in the body.
GET Request can be bookmarked.	POST Request can not be bookmarked.
Idempotent – Second request will be ignored until response of first request is served.	Not idempotent.
More efficient and used more than POST.	Less efficient and used less than GET.

4. **What is status code in HTTP?**
   Status codes are set of digits, generally three digits, issued by the server in response to the client's HTTP request. The first digit of the code specifies one of the five standard classes of response. The last two digits have no categorization role. The following are the five values of the first digit of the status code:

- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

5. **What are the header fields in HTTP?**

Header fields in HTTP are the components of the header sections of request and response messages in HTTP. They provide additional information about the request or response, or any object/data that is sent in the message body.

There are a few header fields which have general applicability for both request and response messages, but which do not apply to the entity being transferred. These header fields apply only to the message being transmitted.

```
general-header = Cache-Control
               | Connection
               | Date
               | Pragma
               | Trailer
               | Transfer-Encoding
               | Upgrade
               | Via
               | Warning
```

6. **What is URI?**

A Uniform Resource Identifier (URI) is a string of characters that unambiguously identifies a particular logical or physical resource. Examples of resources include electronic documents, elevator door sensors, XML namespaces, web pages and ID microchips for pets. To guarantee uniformity, all URIs follow a predefined set of syntax rules. Within the URI of a resource(object), first element is the name of the scheme, separated by a colon with the rest of the URI. The colon is followed by a path in a format depending upon the scheme. The path is interpreted in a manner dependent on the protocol being used.
The generic form of any URI is
scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]

An authority component is made up of multiple parts: an optional authentication section, a host (either a registered name or an IP address) and an optional port number. If a URL refers to a server in port 80, it doesn't explicitly be mentioned in the URI.
A URL for a server on a different port to 80 looks like:
http://www.w3.org:8000/imaginary/test

The path in the URI has a significance defined by the particular scheme. Typically, it is used to encode a name in a given name space, or an algorithm for accessing an object. There are few reserved characters that have some special use in the path.

Reserved Characters

Percent sign ("%") is used as the escape character in the encoding scheme and is never allowed to be used for anything else.

Slash ("/") character is reserved for delimiting of substrings whose relationship is hierarchical. Substring consisting of single (".") or double dots ("..") are similarly reserved.

The significance of slash between two segments means that the segment of the path to the left of slash is more significant to the segment of the path to the right of slash, where "significant" is perceived as the closeness to the root of the path.

The **hash** ("#") character is reserved as a delimiter to separate the URI of an object from a fragment identifier.

The **question mark** ("?") is used to delimit the boundary between the URI of a queryable object, and a set of words used to express a query on that object.

Within the query string, the **plus sign**("+") is reserved as shorthand notation for a space. Therefore, real plus signs must be encoded.

Systems generally use local addressing schemes to access a resource/object which accepts the reserved characters. However, to allow a resource/object to be accessed globally through global gateways or servers, it is necessary to provide a mapping from local addresses into URIs by encode the reserved characters that are otherwise used in local addressing schemes.

According to the conventional URI encoding scheme, where the local naming scheme uses ASCII characters which are not allowed in URI, they are replaced with "%" sign immediately followed by two hexadecimal digits giving the ISO Latin 1 code for the character. Character codes other than those allowed by the syntax are not used unencoded in a URI.

Examples

The URIs

http://www.w3.org/albert/bertram/marie-claude

and

http://www.w3.org/albert/bertram/marie**%2D**claude

are identical, as the %2D encodes a hyphen character.

The URIs

http://www.w3.org/albert/bertram**/**marie-claude

and

http://www.w3.org/albert/bertram**%2F**marie-claude

are NOT identical, as in the second case the encoded slash does not have hierarchical significance.

The **query** contains a string of nonhierarchical data. Although the syntax is not well-defined, it is most often a sequence of attribute value pairs separated by a delimiter, such as an ampersand("&") or a semicolon(";"). The query is separated from the preceding part by a question mark.

For example, a search on text database might look like,

http://info.my.org/AboutUs/Index/Phonebook?dobbins

or

http://www.w3.org/RDB/EMP?*%20where%20name%%3Ddobbins

The **fragment** contains a fragment identifier that provides direction to a secondary resource. For example, if the primary resource is an HTML document, the fragment is often an ID attribute of a specific element of that document. If the fragment identifies a certain section of an article identified by the rest of the URI, a Web browser will scroll this particular element into view. The fragment is separated from the preceding part by a hash (#). For example, a a reference to a particular part of a document, including the fragment identifier, may look like:
http://www.myu.edu/org/admin/people#andy

7. **What are Idempotent methods and why do we call them?**
   An idempotent method can be called several times to return the same outcome. When a client needs a response for sure, it can resend an idempotent request method to the server to receive the same response all the time. Idempotent methods can be used to build fault-tolerant APIs.
   The methods GET,HEAD,PUT and DELETE are idempotent methods.

8. **Explain HTTP Request & Response Messages.**
   The client and server communicate using text messages.
   When a client wants to send a request to the server, it does so by sending a message, known as 'HTTP request message'. The client sends a request message to the server.
   A server responds to the request sent by a client with a message, known as the 'HTTP response message'.
   An HTTP message consists of a message header and an optional message body, separated with a blank line.
   HTTP Request Message
   The HTTP request message conforms to the format as in figure 1.

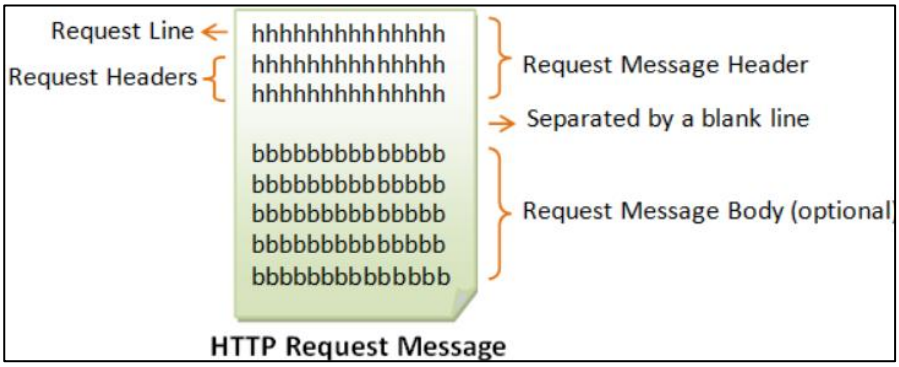


Figure 1

The first line of a HTTP Request Message is called a **Request line**, which is of the following format:
Request-method SP Request-URI SP HTTP-Version CRLF

A **Request method** indicates the HTTP request method that needs to be performed on the target resource identified by 'Requested-URI'. The request method is case sensitive as is the HTTP request methods.

Each element of the request line is separated by space characters indicated with '**SP**' in the above format.

'**Request-URI**' is a Uniform Resource Identifier which identifies the target resource on which the client wants the request method to be performed on. The most common forms of Request URIs is as follows:

Request URI = * | absoluteURI |abs_path | authority
The "*" is used when an HTTP request method does not apply on a particular resource but the whole server. This form is restricted to be used along with only a certain HTTP request method. For example:
OPTION * HTTP/1.1
'absoluteURI' is used when a HTTP request is being made to a proxy. The proxy is requested to forward a request or service from a valid cache and return the response. For example:
GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1

'abs_path' is most commonly used form of Request-URI, used to identify a resource on an origin server or gateway. For example, when a client wishes to retrieve a resource directly from the origin server, it would first create a TCP connection on port 80 of the host and send the following request line:
GET /pub/WWW/TheProject.html HTTP/1.1

Host: www.w3.org
The 'abs_path' can not be empty, so if there is no particular resource that we are indicating in URI, it gives "/" in the request-URI indicating the server root.

HTTP-version indicates the HTTP protocol version in use.
CRLF stands for Carriage Return and Line Feed. It is a set of special characters ('\r\n') that are used to signify End of Line.
No CR or LF characters are allowed in the request line except for the final CRLF sequence.

A Request Message can have zero or more **Request Header(s)**. The request header provides additional information about the request and the client, to the server. There are few identified request header fields that are generally used in request message header. The client may use custom request fields given all parties in the communication (the client and server) agree and recognize them to be request header-fields in prior. Request header is generally a set of header fields and values of the following format:
request-header-field1: value1
request-header-field2: value2….

A Request Message header is separated from the Request Body with a blank line. A **Request Body** is optional in a request message. It generally holds any resource or information that needs to be sent to the target resource.

HTTP Response Message
A HTTP Response Message conforms to the following format as shown in figure 2:

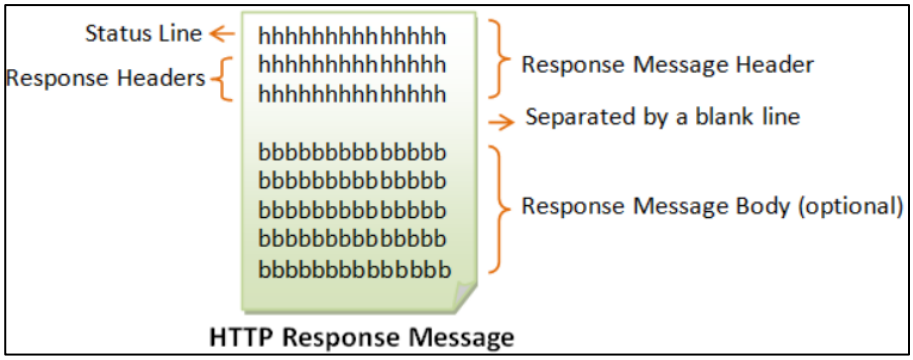


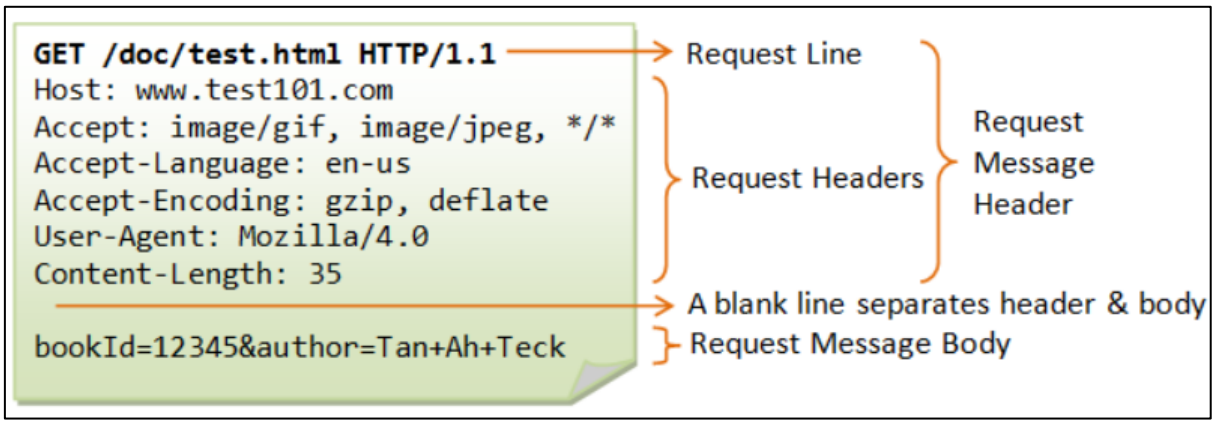
Figure 2

The first line of a HTTP Response Message is called **Status Line**, which is of the following format:
HTTP-version SP status-code SP reason-phrase CRLF
HTTP-version indicates the HTTP version used in that particular session.
Status-code is the three-digit status code that the server sends as a response to the client request.
Reason-phrase gives a short explanation of the status code.
Common status code and reason phrase are "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".

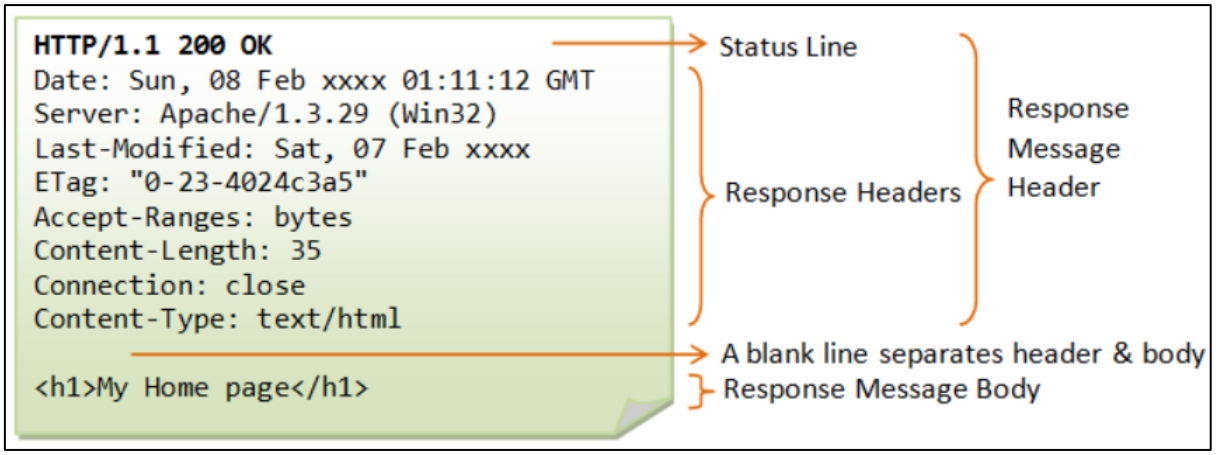
Response Header is optional and allows the server to pass additional information about the response which can not be placed in the status-line. The response header provides additional information about the server and about further access to the resource identified by the Request-URI. The response header is generally a set of response header fields and values of the following format:
response-header-field1: value1
response-header-field2: value2….

Response Message Header is separated from the Response Message Body with a blank line. **Response Message Body** is optional and holds the any resources that is to be sent to client as a response.

Following is an example of a HTTP Request Message and Response Message:
HTTP Request Message



HTTP Response Message



9. **What is Session State in HTTP?**
   HTTP is stateless, which means that it has no built-in way to keep track of a user as they navigate from one webpage to another. Session state is a method to maintain state, i.e., maintain user identity and to store user-specific data during multiple request/response interactions between the client application and web application. HTTP session preserves:
   - Information about the session (session identifier, creation time, time last accessed)
   - Contextual information about the user (client login state)
10. **What is HTTPS?**
    HTTPS is a secure version of HTTP, used to transfer data from web browser to website. HTTPS is encrypted in order to increase the security of data transfer. HTTPS uses encryption protocol to encrypt the communication.

Introduction to API

1. **Explain REST and RESTFUL?**
   REST stands for Representational State Transfer. Its an architectural pattern for providing standards between computer systems on web, making it easier for systems to communicate. Also, it is an architectural pattern for creating web services. Systems or Web services that conform to REST architecture style are called RESTful systems or services respectively.
   Like any other architectural pattern, REST also has guiding principles:
   1. Client-Server
   2. Stateless
   3. Cacheable
   4. Uniform Interface
   5. Layered System
   6. Code On-Demand
2. **Mention what are the HTTP methods supported by REST.**
   REST allows to build any kind of web application with CRUD (create, retrieve, update, delete) operations. REST guidelines suggest using specific HTTP method for specific type of call made to the server. REST supports the following HTTP methods:
   HTTP GET
   GET requests are used to retrieve resource representation or information only and can not be used to update/modify any resource.
   If the requested resource is found, the server responds with a response status code 200 (OK) along with response body, which maybe XML or JSON or text/HTML content.

If the resource is not found, it responds with the status code 404 (NOT FOUND) or if the format of the request is wrong/incomplete it responds with 400 (BAD REQUEST).
HTTP GET method can be used for entire collection of data or a specific item as well.

HTTP POST

POST method is used to create a new resource. When a resource is created, the server responds with status code 201 (CREATED) with the entity describing the status of the resource and location header in the response body. Many times the action performed by POST method may not result in a resource that can be identified by a URI. In that case, a status code with 200 (OK) or 204 (No Content) is an appropriate response code.
HTTP POST method is ideal to be used for a collection of resources rather than a single item.

HTTP PUT

PUT method is used to update existing resources. If a resource is not found, then it can also create the resource. If a new resource is created then the origin server informs the client with a response status code 201 (Created) and if the existing resource is modified, then it responds with the status code 200 (OK) or 204 (No content) to indicate successful completion of request.
If a specific resource is requested and it is not found, then the server responds with the status code 404 (Not Found).
PUT is not allowed to be used for collection of resources unless it is intended to update every resource of the entire collection of resources.

HTTP DELETE

DELETE method is used to delete a specific resource or entire collection of resources. The possible responses of a DELETE method request are:
200 (OK) – if resource is deleted successfully and includes an entity describing the status in response body
202 (Accepted) - if the action has been queued
204 (No content) - if the action has been performed but there is no entity describing the status
404 (Not Found) - if the resource does not exist
405 (Method Not Allowed) – if the request tries to delete an entire collection of resources which maybe restricted.

HTTP PATCH

PATCH is used to make partial update on a resource. A PUT method modifies the entire resource unlike a PATCH method which only partially updates a resource.
PATCH receives similar responses to PUT method.

3. Explain the architectural style for creating web API?

An application programming interface (API) is a set of defined subroutine definitions, protocols, specifications and tools for building software and applications. A Web API is an API over web which can be accessed using HTTP protocol.
The most commonly used architectural style for creating web API is REST. A REST Web API can be created by adhering to following five constraints:

- Client-Server: A separation between client and server which defines different responsibilities and allows to develop APIs on both sides separately as long as they agree upon request formats.
- Statelessness: Communication between client and server is stateless. Every client request holds all the information necessary for the server to process the request which means any server can perform the request of the client. The client has no idea which server is processing its request. The client or server do not need to remember the other party once the request is served.

- Caching: Stateless client-server communication may increase load on the server since some information may have to be transferred several times, so requests that only retrieve data should be cached.
- Layered System: A client can not necessarily know if it directly communicating with the server or the proxy.
- Uniform Interface: REST defines set of operations that can be executed on a resource. A Uniform Interface is a contract between clients and servers which allows each part to evolve regardless of each other.

4. Explain the RESTFul Web Service?

RESTful web services are loosely coupled, light-weight web services that are well suited for creating APIs for clients spread across the internet. Web services that conform to the REST architectural pattern are called RESTful web services. A web service is a collection of open protocols and standards used for exchanging data between applications or systems. RESTful web services use HTTP methods to implement the concept of REST architecture.

In REST, data or functionality is considered resource and are accessed using Uniform Resource Identifier (URI), typically links on Web. The resources are represented by documents and acted upon by pre-defined set of protocols and operations.

Resources are decoupled from their representation so that its content can be accessed in any formats such as HTML, JSON, XML, PDF, plain-text, JPEG and other document formats.

Every interaction with a resource is stateless, that is, request messages are self-contained. Several techniques are explicitly followed for explicit state transfer which allows stateful interactions.

5. Explain what is a "Resource" in REST?

REST considers every content to be a "resource". A resource can be text, image, document, audio file, video, html pages or dynamic business data. In REST style, a resource is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it. It is similar to the object in OOP with a difference that it has definite set operations (HTTP methods like GET, PUT, POST, DELETE) that can be performed on it.

Each resource is identified with a unique URI or Global ID. REST uses various representations to represent resources such as Text, JSON, XML. The most popular representations of resources are XML and JSON.

6. Which protocol is used by RESTful web services?

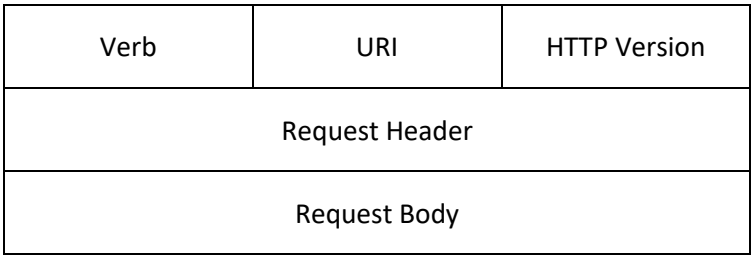
RESTful Web services use HTTP web protocol. It serves as a medium for data communication between client and server. HTTP standard methods are used to access or modify resources in RESTful web services architecture.

HTTP is a stateless protocol which serves as a good communication medium between client and server. This adheres to the constraints of REST architecture style.

7. What is messaging in RESTful web services?

RESTful web services use HTTP protocol as a medium of communication between client and server. A client sends a message in the form of HTTP Request and the server replies with a HTTP Response. This technique of communication is called "Messaging". These messages contain message data and metadata, that is, information about message itself.

8. State the core components of an HTTP Request?

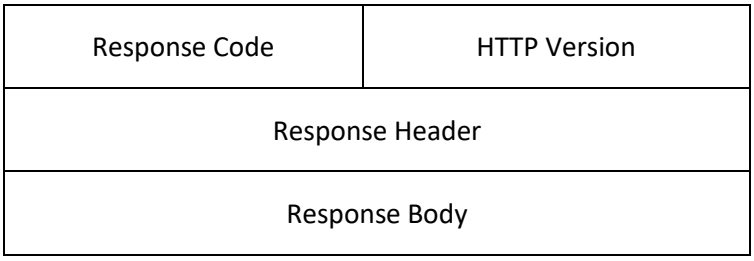


Request Message

A HTTP Request Message has five major parts:

- **Verb** – that indicates one of the HTTP Methods, GET, PUT, DELETE, POST which represents the action to be performed on the resource
- **URI** – Uniform Resource Identifier (URI) to identify the resource on the server
- **HTTP Version** – Indicates HTTP Version used by client
- **Request Header** – Holds metadata of request message as key-value pairs. For example, client or browser type, format supported by client, format of message body, cache settings, etc.
- **Request Body** – Holds message content or resource representation

9. State the core components of an HTTP response?



Response Message

A HTTP Request Message has five major parts:

- **Status/Response Code** – Indicates server response to the action to be performed on resource. For example, 200 means response is OK while 404 means resource is not found.
- **HTTP Version** – Indicates HTTP Version used by server
- **Response Header** – Holds metadata of response message as key-value pairs. For example, content-type, content-length, response date, server type, etc.
- **Response Body** – Holds response message content or resource representation

10. What do you understand about payload in RESTFul web service?

The term "payload" differentiates between the important information in a chunk of data and the overhead to support it. In REST, the most common usage of the term is in the context of message protocol to differentiate between protocol overhead and the actual data.

In a RESTful Web service, HTTP protocol is used as a medium of communication between clients and servers. A client sends a request suing HTTP Request message and the server responds with HTTP Response Message. In these messages, the Message headers are the overheads and the message body in both type of messages hold the actual important data, thus called the payload. The message header gives important information about the client, or the resource access authentication information. However, the message body holds the resource information or data which is the most important data.

A payload in RESTful web service is generally of the format JSON or XML.

11. Explain the caching mechanism.

Cache is a computational storage framework focused on keeping copies of data that are accessed frequently. The purpose of caching is to speed up the process of accessing data.

In REST architecture, caching refers to the process of storing the server responses at the client's end for a time span, which saves the client from making repetitive requests to the server for the same data as it affects the use of server resources.

The benefits of caching can be:

- Reduced network latency
- Reduced server processing load
- Optimization of client response time

There are two categories of cache:

- Shared Cache – Stores server responses to be used by multiple clients
- Private Cache – Stores server responses to be used by a single client

These caches can be implemented through the browser, proxy server, gateway, CDN, reverse proxy, or load balancer of web servers.

In REST architecture, the server sends the information about caching settings along with the response to the client in the header. The cache is stored in the client for a certain time span or not stored at all.

To configure client's caching, a server response can have the following header:

- Date: Date and time of the resource when it was created
- Last Modified: Date and time of the resource when it was last modified
- Cache-Control: Primary header to control caching
- Expires: Expiration date and time of caching
- Age: Duration in seconds from when resources was fetched from the server

The following fields are used in cache-control headers:

- Public: indicates the resource is cacheable by any component
- Private: indicates the resource can be cacheable only by client and server only, and no other intermediary component can cache the resource
- no-cache/no-store: indicates the resource is not cacheable
- max-age: indicates that the cache is valid upto max-age in seconds. After that the client has to make a new request
- must-revalidate: indicates that the cache can be revalidated with the server after the cache expired

12. List the main differences between SOAP and REST?

SOAP (Simple Object Access Protocol)	REST (Representational State Transfer)
Protocol designed with specification. Includes WSDL file which has required information on what web service does and its location.	Architectural style in which web services can be treated as RESTful service only if it follows loose guideline constraints.
Can not make use of REST as SOAP is a protocol and REST is architectural pattern.	REST can make use of SOAP as underlying protocol for web services.
Function-driven; data available as service.	Data-driven; data available as resource.

Uses service interfaces to expose its functionality to client applications.	Uses Uniform Resource locators to access components on the hardware device. (e.g. URI)
Requires more bandwidth. SOAP messages contain lot of information.	Does not need lot of bandwidth. REST messages are mostly JSON.
SOAP can only work with XML format.	Permits data formats such as plain-text, JSON, XML, HTML, etc.
Stateless by default	Stateless
API calls cannot be cached.	API calls can be cached.
Uses HTTP, SMTP, UDP and other transfer protocols.	Uses HTTP.

13. Enlist advantages and disadvantages of 'Statelessness'.

Advantages

- It allows to treat each request independently.
- Scalable. The server does not need to manage any session, enabling deployment of services to any number of servers possible.
- Simplifies application design as client's previous interactions need not be maintained.
- The statelessness constraint brings down the server's response time, as the server requests can be cached by underlying application. This improves the performance with regard to response time.

Disadvantages

- Extra information need to be sent with each request.
- Cookies are used to handle user session information on the internet. Cookies are less efficient.