**REST Web services Framework interview questions answers**

**1) What is REST and RESTful web-services?**

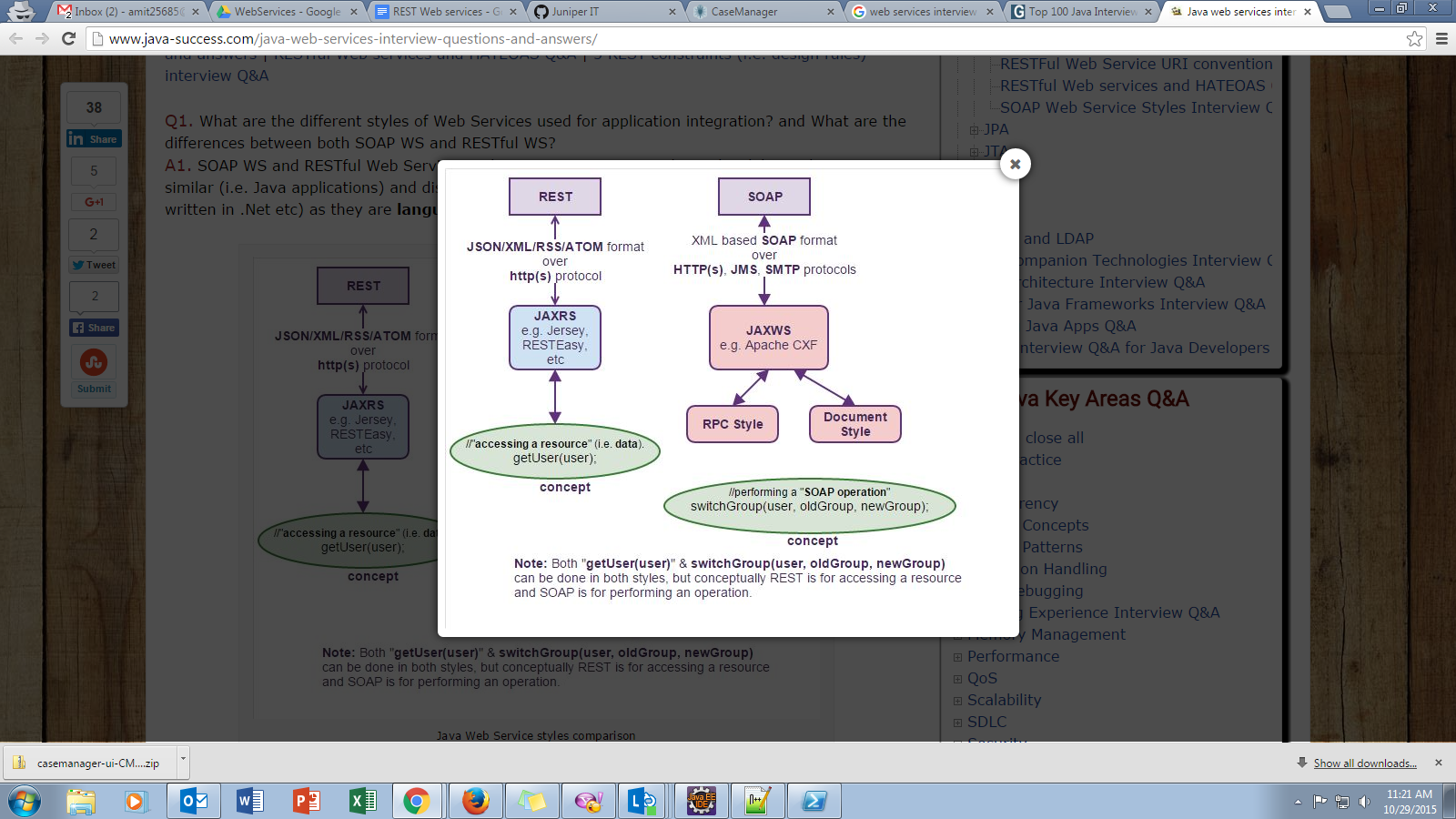
**Ans:** REST stands for REpresentational State Transfer (REST) it’s a relatively new concept of writing web services which enforces a stateless client server design where web services are treated as resource and can be accessed and identified by their URL unlike SOAP web services which were defined by WSDL.

Web services written by REST Architectural concept are called RESTful web services which focus on System resources and how state of Resource should be transferred over http protocol to a different clients written in different languages. In RESTful web services http methods like GET, PUT, POST and DELETE can be used to perform CRUD operations.

**2) What are differences between RESTful web services and SOAP web services?**

**Ans:** A SOAP client works like a custom desktop application, tightly coupled to the server. There's a rigid contract between client and server, and everything is expected to break if either side changes anything.

A REST client is more like a browser. It's a generic client that knows how to use a protocol and standardized methods, and an application has to fit inside that. You don't violate the protocol standards by creating extra methods, you leverage on the standard methods and create the actions with them on your media type. If done right, there's less coupling, and changes can be dealt with more gracefully. A client is supposed to enter a REST service with zero knowledge of the API, except for the entry point and the media type. In SOAP, the client needs previous knowledge on everything he will be using, or it won't even begin the interaction. Additionally, a REST client can be extended by code-on-demand supplied by the server itself, the classical example being javascript code used to drive the interaction with another service on the client-side.



Here are some of differences between REST and SOAP:

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| SOAP is a protocol. | REST is an architectural style. |
| SOAP stands for Simple Object Access Protocol. | REST stands for REpresentational State Transfer. |
| SOAP can't use REST because it is a protocol. | REST can use SOAP web services because it is a concept and can use any protocol like HTTP, SOAP. |
| SOAP uses services interfaces to expose the business logic. | REST uses URI to expose business logic. |
| JAX-WS is the java API for SOAP web services. | JAX-RS is the java API for RESTful web services. |
| SOAP defines standards to be strictly followed. | REST does not define too much standards like SOAP. |
| SOAP requires more bandwidth and resource than REST. | REST requires less bandwidth and resource than SOAP. |
| SOAP defines its own security. | RESTful web services inherits security measures from the underlying transport. |
| SOAP permits XML data format only. | REST permits different data format such as Plain text, HTML, XML, JSON etc. |
| SOAP is less preferred than REST. | REST more preferred than SOAP. |
| Since SOAP messages are wrapped inside a SOAP envelop it can be sent over to any transport mechanism e.g. TCP, FTP, SMTP or any other protocol. | On the other hand RESTful web services are heavily dependent upon HTTP protocol. They used HTTP commands their operation and depends upon on HTTP for transmitting content to server. Though in real world, SOAP is mostly over HTTP so this advantage of transport independence is not really utilized. |
| Works well in distributed enterprise environments | REST assumes direct point-to-point communication. |
| SOAP web services totally ignore web caching mechanism. | RESTful web service take full advantage of web caching mechanism because they are basically [URL based](http://java67.blogspot.sg/2013/01/difference-between-url-uri-and-urn.html). |
| SOAP (Simple Object Access Protocol) is a standard communication protocol on top of transport protocols such as HTTP, SMTP, Messaging, TCP, UDP, etc. | REST is an architectural style by which data can be transmitted over transport protocol such as HTTP(S). |
|  |  |
| Supports both SSL security and WS-security, which adds some enterprise security features. Supports identity through intermediaries, not just point to point SSL.  — WS-Security maintains its encryption right up to the point where the request is being processed.  — WS-Security allows you to secure parts (e.g. only credit card details) of the message that needs to be secured. Given that encryption/decryption is not a cheap operation, this can be a performance boost for larger messages.  — It is also possible with WS-Security to secure different parts of the message using different keys or encryption algorithms. This allows separate parts of the message to be read by different people without exposing other, unneeded information.  — SSL security can only be used with HTTP. WS-Security can be used with other protocols like UDP, SMTP, etc. | Supports only point-to-point SSL security.  — The basic mechanism behind SSL is that the client encrypts all of the requests based on a key retrieved from a third party. When the request is received at the destination, it is decrypted and presented to the service. This means the request is only encrypted while it is traveling between the client and the server. Once it hits the server (or a proxy which has a valid certificate), it is decrypted from that moment on.  — The SSL encrypts the whole message, whether all of it is sensitive or not. |
| Has comprehensive support for both ACIDbased transaction management for short-lived transactions and compensation based transaction management for long-running transactions. It also supports two-phase commit across distributed resources. | REST supports transactions, but it is neither ACID compliant nor can provide two phase commit across distributed transactional resources as it is limited by its HTTP protocol. |
| SOAP has success or retry logic built in and provides end-to-end reliability even through SOAP intermediaries. | REST does not have a standard messaging system, and expects clients invoking the service to deal with communication failures by retrying. |

**3) What is Restlet framework ?**

**Ans:** Restlet is leading RESTful web framework for Java applications is used to build RESTFul web services it has two part Restlet API and a Restlet implementation much like Servlet specification. There are many implementation of Restlet framework available you just need to add these jar in your classpath to use them. By using Restlet web framework you can write client and server.

**4) What is Resource in REST framework?**

**Ans:** It represent a "resource" in REST architecture. on RESTLET API it has life cycle methods like init(), handle() and release() and contains a Context, Request and Response corresponding to specific target resource. This is now deprecated over ServerResource class and you should use that. see Restlet documentation for more details.

**5) Can you use Restlet without any web-container ?**

**Ans:** Yes, Restlet framework provide default server which can be used to handle service request in web container is not available.

**6) What is difference between Restlets and Jersey ?**

**Ans:**

**Spring MVC:**

*Pros:*

This makes it much easier to manage dependency injection with JAX-RS and gives you all of the Spring MVC features like form binding and validation, but you are also able to use the Java standard approach for REST Spring provide easy configurations for security, transaction management & Caching etc

*Cons:*

If it is an EE 6 app, then you may want to think about ditching Spring as JAX-RS is part of the EE 6 specs and you can use EE CDI within your JAX-RS classes.

==========================================================================

**CXF:**

CXF is a merger between XFire and Celtix.

*Pros:*

Great to parse a WSDL and create Java POJOs to interact with, so CXF is pretty good for client-side WSDL services Jersey is mainly devoted to RESTful services and CXF deals mostly with SOAP CXF is devoted to RESTful services and SOAP as well as deeper Integration with spring.

*Cons:*

Inconsistencies in how I needed to handle SSL and HTTP Proxies

==========================================================================

**Restlet:**

probably the first REST framework, which existed prior to JAX-RS.

*Pros*

Restlet supports RESTful web services running as a standalone application, or inside a standard Servlet container. It's many extensions, and the ability to make your own, also enable it to integrate easily into an organization's existing software ecosystem.

Less abstraction An open, flexible, and effective development process; a solid community; and an emphasis on stability and strong support of current releases.

*Cons:*

When it comes to de-marshalling the response into java object you have to do it manually.

A Restlet application cannot only run inside a Servlet container, but can also be run as a standalone Java application using a single "org.restlet.jar" JAR.

==========================================================================

**Jersey :**

- The JAX-RS Reference Implementation from Sun.

*Pros:*

In Netbeans the integration is full

Reference implementation of the JAX-RS API, which the upcoming Restlet 2.0 also implements.

Jersey is usually deployed within a servlet container but does support an embedded mode of operation within Java programs.

Component integration

Jersey currently provides extension-based support for two dependency injection frameworks: the Spring Framework and the Google Guice framework:

Spring Framework. Spring support in Jersey requires a dependency on the jersey-spring module. Spring support is enabled by referencing the SpringServlet class in the web.xml file.

Google Guice framework. Guice support is provided when referencing the Guice filter GuiceFilter and a Guice-specific ServletContextListener in the web.xml file.

*Cons:*

Jersey is mainly devoted to RESTful services and CXF deals mostly (all?) with SOAP

Issues with the Jersey/Spring integration, mainly with AOP.

Note: RESTeasy is easier for EJB 3.0 and SEAM integration, whereas Jersey is easier for Spring and JSON integration.

=========================================================================

**NetKernel:**

*Pros:*

NetKernel can be run as its own platform or it can be instantiated as a POJO run embedded in any Java program or container.

It is a REST software platform that is written in Java, runs on the Java VM and supports languages that compile to or run on the JVM (Scala, Ruby, Python, JavaScript, XQuery, XSLT, etc.).

==========================================================================

**Resteasy:**

- JBoss's JAX-RS project.

RESTeasy's proprietary cache for the URL or query

*Pros:*

if component integration is of primary importance, the Restlet Framework or RESTEasy might be a good choice.Being a JBoss-driven implementation, it's not surprising that RESTEasy supports close integration with JBoss Seam. However, it also supports integration with other popular frameworks and standards like Enterprise JavaBean (EJB) technology (a Java Platform, Enterprise Edition [Java EE] standard), Spring, and Google Guice.

**Metro:**

Oracle provides Metro for Rest Implementaion.However latest develeopment have been stopped for last three years.Glassfish server gives much support for this implementation.

**AXIS2:**

*Pros:*

Axis2 is structured modularly, has many features and can be used as an application server for Web Services support of exchangeable binding frameworks, for example XMLBeans. Axis2 together with the XMLBeans framework is well suited for Web Services which are using very complex schema definitions.

*Cons:*

The disadvantages of Axis2 are its complexity as well as the insufficient JAX-WS support. Therefore anyone who wants to work with JAX-WS should choose Apache CXF or the reference implementation.

***Jersy Vs Spring***

spring vs jax-rs, mostly same concept utilizing annotation,

main difference is spring mvc default method to return view or view name (need body instead of using @RespondBody).

web.xml servlet-class: org.springframework.web.servlet.DispatcherServlet (spring) vscom.sun.jersey.spi.container.servlet.ServletContainer (jersey)

@RequestMapping (spring) vs @Path (jersey)

@RequestMapping(method=RequestMethod.GET) (spring) vs @Get (jersey), etc

@PathVariable (spring) vs @PathParam (jersey)

@RequestVariable (spring) vs @QueryParam (jersey)

@ResponseStatus (spring) vs get/set response via @Context (jersey)

@RequestBody (spring) vs method(String input) (jersey) ?

@ResponseBody (spring) vs method return type (jersey)

@ExceptionHandler (spring) vs @Provider and ExceptionMapper interface (jersey)

@RequestMapping(headers=…) (spring) vs @Produces, @Consumes (jersey)

Jersey @Consumers, if a resource is unable to consume the MIME type of a client request, the Jersey runtime sends back an HTTP 415 Unsupported Media Type error.

In Spring, we can also use **ContentNegotiatingViewResolver** to implement restful webservice instead of annotations

**7) What is RESTEasy ?**

**Ans:** RESTEasy is another REST framework introduced in JBoss Application Server. This was rather easy REST interview questions. you can answer in detail only if you have used this or working in JBoss.

**8) What are the tools used for creating RESTFul web services ?**

**Ans:** You can use AJAX(Asynchronous JavaScript with XAML) and Direct Web Removing to consume web services in web application. Both Eclipse and NetBeans also supported development of RESTFul services.

**9) How to display custom error pages using RestFul web services ?**

**Ans:** In order to customize error you need to extend StatusService and implement getRepresentation(Status, Request, Response) method with your custom code now assign instance of your CustomStatusService to appropriate "statusService property".

**10) Which HTTP methods are supported by RestFull web services ?**

**Ans:** Another common REST interview questioning RESTFul web service each Resource supports GET, POST, PUT and DELETE http methods.GET is mapped to represent(), POST - acceptRepresentation(), PUT- storeRepresentation and DELET for rmeoveRepresentation.

**11) What is difference between top-down and bottom-up approach of developing web services ?**

**Ans:** In top-down approach first WSDL document is created and than Java classes are developed based on WSDL contract, so if WSDL contract changes you got to change your Java classes while in case of bottom up approach of web service development you first create Java code and then use annotations like @WebService to specify contract or interface and WSDL field will be automatically generated from your build.

**12) What happens if Restful resources are accessed by multiple clients? Do you need to make it thread-safe?**

**Ans:** Since a new Resource instance is created for every incoming Request there is no need to make it thread-safe or add synchronization. Multiple clients can safely access Restful resources concurrently.

[**Comparing RESTful web services implementation with Jersey and Restlet**](http:///h)

|  |  |  |
| --- | --- | --- |
| Factors | Restlet | Jersey |
| Introduction | probably the first REST framework, which existed prior to JAX-RS. | the JAX-RS Reference Implementation from Sun. (http://jersey.java.net/) |
| Maturity of the product | Available in the market from 2005 | Available in the market from 2007 |
| Open Source | Licensed with CDDL 1.0, LGPL 2.1, LGPL 3.0, EPL 1.0. | Licensed with CDDL 1.1 and GPL 2 with CPE |
| Server-side integration strategies | A Standalone JSE type of code can be written to open a server port of any choice. Program can run as a standalone Java application. The restlet also supports JavaEE environment with the help of Jetty web container. | Update web.xml with reference to jersey Servlet Container and its available for server side jersey REST API. JAX-RS annotations can be used to represent a REST API. @Path, @GET, are easily available due to JAX-RS reference. |
| Java Client API | Another standalone application which requires org.restlet.ext.simple | Available and easy to use. However HttpClient API can also be used since jersey is for REST over http protocol only. |
| Configurability | Every configuration is part of POJO's. Class files are defined and implement methods of your choice to configure the server, security. | Easy configurations, just an update with web.xml and add jars through maven pom.xml and we are good to go with Jersey. |
| Security | SSL enabled, Security should be configured through programming only. Authentication and Authorization are set based on HTTP.Basic, HTTP.DIGEST, Amazon S3 and SMTP. The server runs with restlet is compared with Apache HTTP Server. | Avoids DoS, Brute Force, XSS type of vulnerabilities through Tomcat Security Configurations. Deploying unauthorized applications are prohibited, Access to directory listings is prohibited or password protected, SSL enabled. |
| Performance | 10 million records,throws Java Heap Space. 1 million records takes a long long time. 1 lakh records takes (15.95, 16.525, 15.889). 10,000 records takes (1.087, 0.649, 0.552) | 10 million records, throws Java Heap Space. 1 million records takes a long long time. 1 lakh records takes (16.081, 15.355, 15.289). 10,000 records takes (0.834, 0.752, 0.66) |
| Ease of use | Very easy to use, Documentation and tutorials available at wiki.restlet.org is very helpful. | Very easy to use, Documentation and tutorials available at jersey.java.net is very helpful. |
| Time to write code and deploy | Tutorials are available from restlet.org which can help write the code. Just for a sample hello world program, I have to write 1 FirstServerResource, 1 HelloWorldResource, 1 RestletServerTest and 1 RestletClientTest java class files. | Lots of tutorials are available in Web which can help write this code easily. 1 HelloWorldResource can deploy to a tomcat container to test it. |
| Asynchronous Support | Restlet client API does not support Asynchronous directly. Need to use a third party library calld AsynchronousHttpClient (From ning.com) | Jersey client API does not support Asynchronous directly. Need to use a third party library called AsynchronousHttpClient (From ning.com) |
| Runs in Tomcat along with other services like Java-PHP Bridge | Works because Restlet runs as a separate server. | Works nice and clean |
| URL Opens in a normal browser? | No. We need to write a Restlet client using Restlet API's. | Yes. |
| Maven Support | groupid: org.restlet.jse artifactid: org.restlet repository: http://maven.restlet.org | groupid: org.glassfish.jersey.core artifactid: jersey-server repository: maven-central |

**13: Difference between PUT and POST:**

**Ans:** After that discussion, a more realistic mapping would seem to be:

Create = PUT if you are sending the full content of the specified resource (URL).

Create = POST if you are sending a command to the server to create a subordinate of the specified resource, using some server-side algorithm.

Retrieve = GET.

Update = PUT if you are updating the full content of the specified resource.

Update = POST if you are requesting the server to update one or more subordinates of the specified resource.

Delete = DELETE.

**14:**

**9 Method Definitions**

**Ans:** The set of common methods for HTTP/1.1 is defined below. Although this set can be expanded, additional methods cannot be assumed to share the same semantics for separately extended clients and servers.

The Host request-header field (section [14.23](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#/h)) MUST accompany all HTTP/1.1 requests.

9.1 Safe and Idempotent Methods

**9.1.1 Safe Methods**

Implementers should be aware that the software represents the user in their interactions over the Internet, The convention has been established that the GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval. These methods ought to be considered "safe".

**9.1.2 Idempotent Methods**

Methods can also have the property of "idempotence" in that (aside from error or expiration issues) the side-effects of N > 0 identical requests is the same as for a single request. The methods GET, HEAD, PUT and DELETE share this property. Also, the methods OPTIONS and TRACE SHOULD NOT have side effects, and so are inherently idempotent.

However, it is possible that a sequence of several requests is non- idempotent, even if all of the methods executed in that sequence are idempotent. (A sequence is idempotent if a single execution of the entire sequence always yields a result that is not changed by a re execution of all, or part, of that sequence.) For example, a sequence is non-idempotent if its result depends on a value that is later modified in the same sequence.

A sequence that never has side effects is idempotent, by definition (provided that no concurrent operations are being executed on the same set of resources).

**9.2 OPTIONS**

The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI. This method allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval.

Responses to this method are not cacheable.

If the OPTIONS request includes an entity-body (as indicated by the presence of Content-Length or Transfer-Encoding), then the media type MUST be indicated by a Content-Type field. Although this specification does not define any use for such a body, future extensions to HTTP might use the OPTIONS body to make more detailed queries on the server. A server that does not support such an extension MAY discard the request body.

If the Request-URI is an asterisk ("\*"), the OPTIONS request is intended to apply to the server in general rather than to a specific resource. Since a server's communication options typically depend on the resource, the "\*" request is only useful as a "ping" or "no-op" type of method; it does nothing beyond allowing the client to test the capabilities of the server. For example, this can be used to test a proxy for HTTP/1.1 compliance (or lack thereof).

If the Request-URI is not an asterisk, the OPTIONS request applies only to the options that are available when communicating with that resource.

A 200 response SHOULD include any header fields that indicate optional features implemented by the server and applicable to that resource (e.g., Allow), possibly including extensions not defined by this specification. The response body, if any, SHOULD also include information about the communication options. The format for such a body is not defined by this specification, but might be defined by future extensions to HTTP. Content negotiation MAY be used to select the appropriate response format. If no response body is included, the response MUST include a Content-Length field with a field-value of "0".

The Max-Forwards request-header field MAY be used to target a specific proxy in the request chain. When a proxy receives an OPTIONS request on an absolute URI for which request forwarding is permitted, the proxy MUST check for a Max-Forwards field. If the Max-Forwards field-value is zero ("0"), the proxy MUST NOT forward the message; instead, the proxy SHOULD respond with its own communication options. If the Max-Forwards field-value is an integer greater than zero, the proxy MUST decrement the field-value when it forwards the request. If no Max-Forwards field is present in the request, then the forwarded request MUST NOT include a Max-Forwards field.

**9.3 GET**

The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

The semantics of the GET method change to a "conditional GET" if the request message includes an If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field. A conditional GET method requests that the entity be transferred only under the circumstances described by the conditional header field(s). The conditional GET method is intended to reduce unnecessary network usage by allowing cached entities to be refreshed without requiring multiple requests or transferring data already held by the client.

The semantics of the GET method change to a "partial GET" if the request message includes a Range header field. A partial GET requests that only part of the entity be transferred, as described in section [14.35](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#/h). The partial GET method is intended to reduce unnecessary network usage by allowing partially-retrieved entities to be completed without transferring data already held by the client.

The response to a GET request is cacheable if and only if it meets the requirements for HTTP caching described in section 13.

**9.4 HEAD**

The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response. The metainformation contained in the HTTP headers in response to a HEAD request SHOULD be identical to the information sent in response to a GET request. This method can be used for obtaining metainformation about the entity implied by the request without transferring the entity-body itself. This method is often used for testing hypertext links for validity, accessibility, and recent modification.

The response to a HEAD request MAY be cacheable in the sense that the information contained in the response MAY be used to update a previously cached entity from that resource. If the new field values indicate that the cached entity differs from the current entity (as would be indicated by a change in Content-Length, Content-MD5, ETag or Last-Modified), then the cache MUST treat the cache entry as stale.

**9.5 POST**

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions:

-Annotation of existing resources;

- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;  
- Providing a block of data, such as the result of submitting a form, to a data-handling process;  
- Extending a database through an append operation.

The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.

The action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either 200 (OK) or 204 (No Content) is the appropriate response status, depending on whether or not the response includes an entity that describes the result.

If a resource has been created on the origin server, the response SHOULD be 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a Location header

Responses to this method are not cacheable, unless the response includes appropriate Cache-Control or Expires header fields. However, the 303 (See Other) response can be used to direct the user agent to retrieve a cacheable resource.

**9.6 PUT**

The PUT method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, and that URI is capable of being defined as a new resource by the requesting user agent, the origin server can create the resource with that URI. If a new resource is created, the origin server MUST inform the user agent via the 201 (Created) response. If an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes SHOULD be sent to indicate successful completion of the request. If the resource could not be created or modified with the Request-URI, an appropriate error response SHOULD be given that reflects the nature of the problem. The recipient of the entity MUST NOT ignore any Content-\* (e.g. Content-Range) headers that it does not understand or implement and MUST return a 501 (Not Implemented) response in such cases.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are not cacheable.

The fundamental difference between the POST and PUT requests is reflected in the different meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request -- the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource. If the server desires that the request be applied to a different URI,

it MUST send a 301 (Moved Permanently) response; the user agent MAY then make its own decision regarding whether or not to redirect the request.

A single resource MAY be identified by many different URIs. For example, an article might have a URI for identifying "the current version" which is separate from the URI identifying each particular version. In this case, a PUT request on a general URI might result in several other URIs being defined by the origin server.

HTTP/1.1 does not define how a PUT method affects the state of an origin server.

PUT requests MUST obey the message transmission requirements set out in section 8.2.

Unless otherwise specified for a particular entity-header, the entity-headers in the PUT request SHOULD be applied to the resource created or modified by the PUT.

**9.7 DELETE**

The DELETE method requests that the origin server delete the resource identified by the Request-URI. This method MAY be overridden by human intervention (or other means) on the origin server. The client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully. However, the server SHOULD NOT indicate success unless, at the time the response is given, it intends to delete the resource or move it to an inaccessible location.

A successful response SHOULD be 200 (OK) if the response includes an entity describing the status, 202 (Accepted) if the action has not yet been enacted, or 204 (No Content) if the action has been enacted but the response does not include an entity.

If the request passes through a cache and the Request-URI identifies one or more currently cached entities, those entries SHOULD be treated as stale. Responses to this method are not cacheable.

**9.8 TRACE**

The TRACE method is used to invoke a remote, application-layer loop- back of the request message. The final recipient of the request SHOULD reflect the message received back to the client as the entity-body of a 200 (OK) response. The final recipient is either the

origin server or the first proxy or gateway to receive a Max-Forwards value of zero (0) in the request (see section 14.31). A TRACE request MUST NOT include an entity.

TRACE allows the client to see what is being received at the other end of the request chain and use that data for testing or diagnostic information. The value of the Via header field (section [14.45](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#/h)) is of particular interest, since it acts as a trace of the request chain. Use of the Max-Forwards header field allows the client to limit the length of the request chain, which is useful for testing a chain of proxies forwarding messages in an infinite loop.

If the request is valid, the response SHOULD contain the entire request message in the entity-body, with a Content-Type of "message/http". Responses to this method MUST NOT be cached.

**9.9 CONNECT**

This specification reserves the method name CONNECT for use with a proxy that can dynamically switch to being a tunnel (e.g. SSL tunneling [[44]](http://www.w3.org/Protocols/rfc2616/rfc2616-sec17.html#/h)).

[**POST vs. PUT**](http:///h)

Musicians will understand this analogy. Have you ever tried to learn a piece that goes the wrong way? That is, you’re playing along and it’s so obvious where the next notes are going to go and instead the piece goes off in a completely different direction. Half the time you find yourself playing the notes you think the piece should use rather than the notes it does use.

For me, understanding the difference between HTTP POST and PUT is very much like that. I’ve had a great deal of trouble understanding explicitly RESTful protocols like [APP](http:///h) because they follow the actual definition of POST and PUT instead of what is to me clearly the right definition. However, I think I’m finally starting to get it.

My mistake is in thinking that anything that creates a new page is a PUT and anything that changes an existing page is a POST. In SQL terms, POST is an UPDATE and PUT is an INSERT. However, that’s not the case. In fact, the mistake is in trying to model PUT and POST in terms of INSERT and UPDATE. They really aren’t even close.

What actually happens is this. PUT puts a page at a specific URL. If there’s already a page there, it’s replaced in toto. If there’s no page there, a new one is created. This means it’s like a DELETE followed by an insert of a new record with the same primary key.

POST, however, really has no equivalent in SQL. POST sends some data to a specified URL. The server on the other end of this URL can do whatever it wants with this data. It can store it somewhere private. (HTTP 204 NO CONTENT). It can store it in the page at the URL that was POSTed to (HTTP 205 RESET CONTENT). It can store it in a new page, in which case it returns the URL of that page in the Location field of the HTTP response header (HTTP 201 CREATED). It can use it as input for several different existing and new pages. It can throw the information away. It can insert, update, or delete records in a database (or all of the above). It can start brewing coffee (HTTP 202 ACCEPTED). It can start global thermonuclear war. POST is decidely non-side-effect free and non-idempotent.

PUT is a much more limited operation that never does anything more than PUT one page at a specified URL. It is idempotent, which is a fancy way of saying that doing it twice is the same as doing it once. Both PUT and POST can be used to create new pages. However PUT should be used when the client specifies the location for the page. PUT is normally the right protocol for a web editor like DreamWeaver or BBEdit. POST is used when the client gives sends the page to the the server, and the server then tells the client where it put it. POST is normally the right protocol for a blog editor like TypePad or anything that inputs into a content management system. In SQL analogy, POST is an INSERT with an automatically generated primary key, and PUT is an INSERT that specifies the primary key in the INSERT statement.

**15) Give me an example of real web service?**

**Ans:**One example of web services is IBM Web Services browser. You can get it from IBM Alphaworks site. This browser shows various demo's related to web services.Basically web services can be used with the help of SOAP, WSDL, and UDDI . All these, provide a plug-and-play interface for using web services such as stock-quote service, a traffic-report service, weather service etc.

**16) How you define web service protocol stack?**

**Ans:** It is basically set of various protocols that can be used to explore and execute web-services.

The entire stack has four layers i.e

**Service Transport, XML Messaging,Service Description and Service Discovery.**

**17) Can you define each of these layers of protocol stack?**

**Ans:**The Service Transport layer transfer messages between different applications, such as HTTP, SMTP, FTP, and Blocks Extensible Exchange Protocol (BEEP).

The XML Messaging layer encodes messages in XML format so that messages can be understood at each end, such as XML-RPC and SOAP.

The Service Description layer describes the user interface to a web service, such as WSDL. The Service Discovery layer centralizes services to a common registry and offer simple publish functionality, such as UDDI.

**18) Define XML – RPC?**

**Ans:** It is a protocol that makes use of XML messages to do Remote Procedure Calls.

**19) Define SOAP?**

**Ans:** SOAP is an XML based protocol to transfer between computers.

**20) Define WSDL?**

**Ans:** It means Web Services Description Language. It is basically the service description layer in the web service protocol stock. The Service Description layer describes the user interface to a web service.

**21) What kind of security is needed for web services?**

**Ans:Web Services**

The security level for web services should be more than that of what we say Secure Socket layer (SSL). This level of security can be only achieved from ***Entrust Secure Transaction Platform.*** Web services need this level of security to ensure reliable transactions and secure confidential information.

**22) Do you have any idea about foundation security services?**

**Ans:** As implies from its name, these services are the foundation or basics of integration,authentication, authorization, digital signatures and encryption processes.

**23) Define Entrust Identification Service?**

**Ans:** Entrust Identification Service comes from the Entrust Security Transaction Platform.This platform allows companies to control the identities that are trusted to perform transactions for Web services transactions.

**24) What UDDI means?**

**Ans:** UDDI stands for Universal, Description, Discovery, and Integration. It is the discovery layer in the web services protocol stack.

**25) Define Entrust Entitlements Service?**

**Ans:** This service verifies entities that attempt to access a web service. For Example, the authentication service, the Entitlements Service ensures security in business operations.

**26) Define Entrust Privacy Service?**

**Ans:**As its name implies, it deals with security and confidentiality. This service encrypts data to ensure that only concerned parties can access the data.

**27) What do you mean by PKI?**

**Ans:** It means Public-Key Infrastructure.

**28) Differentiate between a SOA and a Web service?**

**Ans:** SOA is a design and architecture to implement other services. SOA can be easily implemented using various protocols such as HTTP, HTTPS, JMS, SMTP, RMI, IIOP, RPC etc. While Web service, itself is an implemented technology. In fact one can implement SOA using the web service.

**29) Discuss various approaches to develop SOAP based web service?**

**Ans:** We can develop SOAP based web service with two different types of approaches such as contract-first and contract-last. In the first approach, the contract is defined first and then the classes are derived from the contract while in the later one, the classes are defined first and then the contract is derived from these classes.

**30) If you have to choose one approach, then what will be your choice?**

**Ans:** In my point of view, the first approach that is the contract-first approach is more feasible as compared to the second one but still it depends on other factors too.

**31) Can you name few free and commercial implementations for web services?**

**Ans:** The implementations I know are Apache SOAP, JAX-WS Reference Implementation, JAX-RS Reference Implementation, Metro, Apache CXF, MS.NET and Java 6.

**32) Name browser that allows access to web service?**

**Ans:** JavaScript XmlHttpRequest object is required to access web service via browsers.The browsers that support this object are Internet Explorer, Safari and Mozilla-based browsers like FireFox.

**33) What is REST?**

**Ans:** REST stands for Representational State Transfer. REST itself is not a standard, while it uses various standards such as HTTP, URL, XML/HTML/GIF/JPEG (Resource Representations) and text/xml, text/html, image/gif, image/jpeg, etc(MIME Types).

**34) How can you document web service?**

**Ans:** Web services are contemplated as self-documenting because they provide entire information regarding the available methods and parameters used for XML based standard, known as WSDL. One can also provide more information to explain web services via their own Web-Service and WebMethod attributes.

**35) What are the situations, when we need ASP.NET web services?**

**Ans:** ASP.NET web services are used when one need to implement three tier architecture in a web service. It allows handy ways to use middle tier components through internet. The main advantage of .NET Web services is that they are capable enough to communicate across firewalls because they use SOAP as transport protocol.

**36) Differentiate between web services, CORBA and DCOM?**

**Ans:** Web services transfer/receive messages to/from application respectively, via HTTP protocol. It uses XML to encode data. CORBA and DCOM transfer/receive messages to/from application respectively, via non-standard protocols such as IIOP and RPC.

**37) Can you name some standards used in web services?**

**Ans:** The standards used in web services are WSDL (used to create interface definition), SOAP (used to structure data), HTTP (communication channels), DISCO (used to create discovery documents) and UDDI (used to create business registries).

**38) Explain in brief, what DISCO is?**

**Ans:** DISCO means discovery. It groups the list of interrelated web services. The organization that provides web services, issues a DISCO file on its server and that file contains the links of all the provided web services. This standard is good when client knows the company already. Also it can be used within a local network as well.

**39) Explain in brief, what UDDI is?**

**Ans:** UDDI (Universal Description, Discovery, and Integration) provides consolidated directory for web services on the internet. Clients use UDDI to find web services as per their business needs. It basically hosts the web services from various companies. In order to share web services, you need to publish it in UDDI.

**40) How a SOAP message is structured?**

**Ans:** A SOAP message is consists of SOAP Envelope, SOAP Headers, and SOAP Body.

**41. How would you decide what style of Web Service to use? SOAP WS or REST?**

**Ans**. In general, a REST based Web service is preferred due to its simplicity, performance, scalability, and support for multiple data formats. SOAP is favored where service requires comprehensive support for security and transactional reliability.

The answer really depends on the functional and non-functional requirements. Asking the questions listed below will help you choose.

* **Does the service expose data or business logic?** (REST is a better choice for exposing data, SOAP WS might be a better choice for logic).Do the consumers and the service providers require a formal contract? (SOAP has a formal contract via WSDL)
* **Do we need to support multiple data formats?**
* **Do we need to make AJAX calls?** (REST can use the XMLHttpRequest)
* **Is the call synchronous or asynchronous?**
* **Is the call stateful or stateless?** (REST is suited for stateless CRUD operations)
* **What level of security is required?** (SOAP WS has better support for security)
* **What level of transaction support is required?** (SOAP WS has better support for transaction management)
* **Do we have limited bandwidth?** (SOAP is more verbose)
* **What’s best for the developers who will build clients for the service?** (REST is easier to implement, test, and maintain)

**42. Why not favor traditional style middle-ware such as RPC, CORBA, RMI and DCOM as opposed to Web services?**

**Ans.**

The **traditional middle-wares** tightly couple connections to the applications and it can break if you make any modification to your application. Tightly coupled applications are hard to maintain and less reusable. Generally do not support heterogeneity. Do not work across Internet. Can be more expensive and hard to use. **Web Services** support loosely coupled connections. The interface of the Web service provides a layer of abstraction between the client and the server. The loosely coupled applications reduce the cost of maintenance and increases re-usability. Web Services present a new form of middle-ware based on XML and Web. Web services are language and platform independent. You can develop a Web service using any language and deploy it on to any platform, from small device to the largest supercomputer. Web service uses language neutral protocols such as HTTP and communicates between disparate applications by passing XML messages to each other via a Web API. Do work across internet, less expensive and easier to use.

## **43: What types of operations are available in WSDL?**

**Ans:** There are four operations available:

1. One-way, where the operation can receive a message but will not return a response.

2. Request-response, where the operation can receive a request and will return a response.

3. Solicit-response, where the operation can send a request and will wait for a response.

4. Notification, where the operation can send a message but will not wait for a response.

**44:** Explain few disadvantages of Response Caching?

**Ans:** Response Caching is useless or incompetent when method accepts extensive amount of values because caching means to store lot of information. Also, if the method depends on external source of information, and that are not provided within the parameters then such methods are bypassed.

**45:** What is the alternate solution to Response Caching?

**Ans:** One can use Data Caching (System.Web.Caching.Cach) instead of Response Caching.

**46:** How can one access a class as a web service?

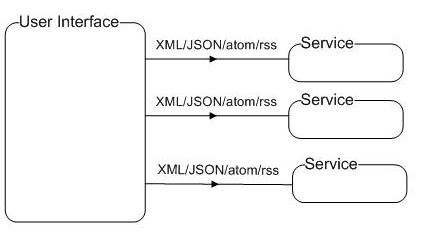
**Ans:** To access a class as a web service, one should inherit the class from the System.Web.Services.WebService class and qualify the class with the WebService attribute.

**47:** How can one access the web service class method via internet?

**Ans:** To access web service class method via internet, one should qualify a method with the WebMethod attribute.

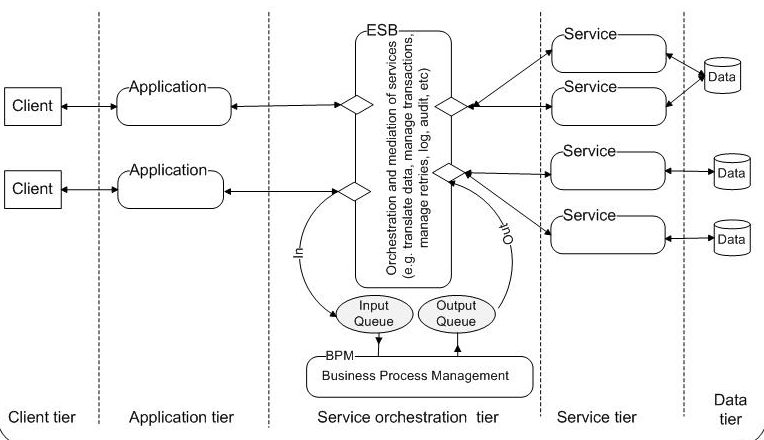
**48.** Differentiate between SOA (Service Oriented Architecture) versus WOA (Web Oriented Architecture)?

**Ans.** WOA extends SOA to be a light-weight architecture using technologies such as REST and POX (Plain Old XML). POX compliments REST. JSON is a variant for data returned by REST Web Services. It consumes less bandwidth and is easily handled by web developers mastering the Javascript language



WOA – RESTFul Service Calls via AJAX to populate different sections of a UI

SOA and WOA differ in terms of the layers of abstraction. SOA is a system-level architectural style that tries to expose business capabilities so that they can be consumed by many applications.WOA is an interface-level architectural style that focuses on the means by which these service capabilities are exposed to consumers. You can start out with a WOA and then grow into SOA.



SOA (Service Oriented Architecture)

**Q49.** How would you decide what style of Web Service to use? SOAP WS or REST?

**Ans.** In general, a REST based Web service is preferred due to its simplicity, performance, scalability, and support for multiple data formats. SOAP is favored where service requires comprehensive support for security and transactional reliability.

The answer really depends on the functional and non-functional requirements. Asking the questions listed below will help you choose.

1) Does the service expose data or business logic? (REST is a better choice for exposing data, SOAP WS might be a better choice for logic).

2) Do consumers and the service providers require a formal contract? (SOAP has a formal contract via WSDL)

3) Do we need to support multiple data formats?

4) Do we need to make AJAX calls? (REST can use the XMLHttpRequest)

5) Is the call synchronous or asynchronous?

6) Is the call stateful or stateless? (REST is suited for statless CRUD operations)

7) What level of security is required? (SOAP WS has better support for security)

8) What level of transaction support is required? (SOAP WS has better support for transaction management)

9) Do we have limited band width? (SOAP is more verbose)

10) What’s best for the developers who will build clients for the service? (REST is easier to implement, test, and maintain)

**Q50.** What is the difference between SOA and a Web service?

**Ans.** SOA is a software design principle and an architectural pattern for implementing loosely coupled, reusable and coarse grained services. You can implement SOA using any protocols such as HTTP, HTTPS, JMS, SMTP, RMI, IIOP (i.e. EJB uses IIOP), RPC etc. Messages can be in XML or Data Transfer Objects (DTOs).

Web service is an implementation technology and one of the ways to implement SOA. You can build SOA based applications without using Web services – for example by using other traditional technologies like Java RMI, EJB, JMS based messaging, etc. But what Web services offer is the standards based and platform-independent service via HTTP, XML, SOAP, WSDL and UDDI, thus allowing interoperability between heterogeneous technologies such as J2EE and .NET.