REST support implemented with Jersey

**1. What is differences between RESTful web services and SOAP web services ?**  
**Ans:**Though both RESTful web series and SOAP web service can operate cross platform they are architecturally different to each other, here is some of differences between REST and SOAP:  
1) REST is more simple and easy to use than SOAP. REST language is based on use of nouns and verbs (better readability)  
2) REST uses HTTP protocol for producing or consuming web services while SOAP uses XML.

* The SOAP WS is transport protocol neutral. Supports multiple protocols like HTTP(S),  Messaging, TCP, UDP SMTP, etc.
* The REST is transport protocol specific. Supports only HTTP or HTTPS protocols.

3) REST is lightweight as compared to SOAP and preferred choice in mobile devices and PDA’s. REST does not need XML parsing, no message header (to and from), hence less bandwidth  
4) REST supports different format like text, JSON and XML while SOAP only support XML.

* The SOAP WS permits only XML data format. You define operations, which tunnels through the POST. The focus is on accessing the named operations and exposing the application logic as a service.
* The REST permits multiple data formats like XML, JSON data, text, HTML, etc. Any browser can be used because the REST approach uses the standard GET, PUT, POST, and DELETE Web operations. The focus is on accessing the named resources and exposing the data as a service. REST has AJAX support. It can use the XMLHttpRequest object. Good for stateless CRUD (Create, Read, Update, and Delete) operations.

GET – represent()  
POST – acceptRepresention()  
PUT – storeRepresention()  
DELETE – removeRepresention()  
5) SOAP based reads cannot be cached. REST based reads can be cached. Performs and scales better.  
6) Different error handling:  
REST: requires HTTP error handling  
SOAP: can have user defined error  
7) REST only supports synchronous  message because of its reliance of HTTP and HTTPS  
8) SOAP WS supports both SSL security and WS-security, which adds some enterprise security features like maintaining security right up to the point where it is needed, maintaining identities through intermediaries and not just point to point SSL only, securing different parts of the message with different security algorithms, etc.  
The REST supports only point-to-point SSL security. The SSL encrypts the whole message, whether all of it is sensitive or not.  
9) The SOAP has comprehensive support for both ACID based  transaction management  for short-lived transactions and compensation based transaction management for long-running transactions. It also supports two-phase commit across distributed resources.  
The 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.  
10) The 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.

**2. What is REST and RESTful web services?**  
Ans: REST stands for REpresentational State Transfer (REST) its 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 there URL unlike SOAP web services which were defined by WSDL.  
Web services written by apply 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 can be used to perform CRUD operations.

**3.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.

**4.What happens if RestFull 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 RestFull resources concurrently.

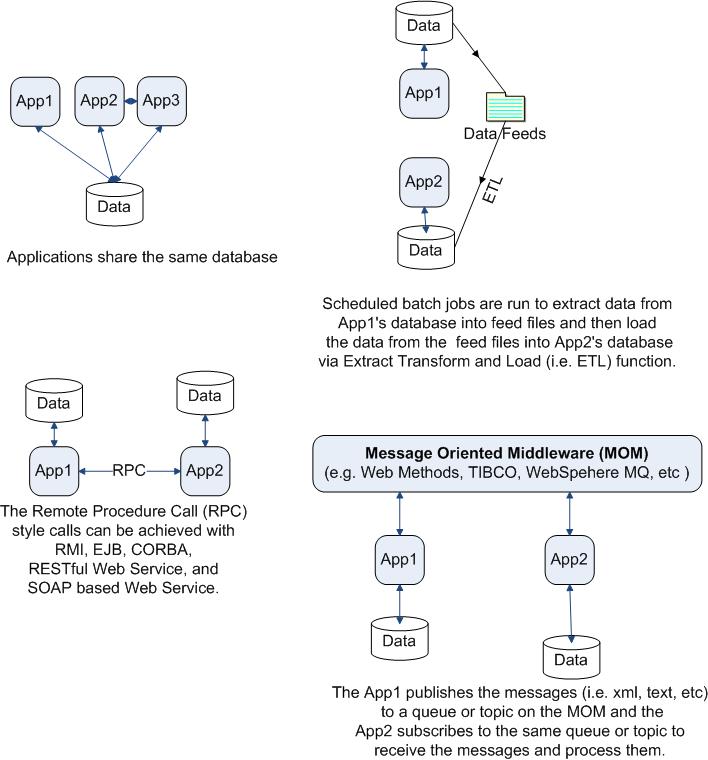
**5.What will you do when an error code has to be returned to the client  
OR  
How will you handle application error scenarios in RESTful web service?**

**7.Can I use GET request instead of PUT to create resources?**  
**Ans:**No, you are supposed to use PUT or POST. GET operations should only have view rights.

**9.What all tools have you used to write RESTful web service?**

**10.What is meant by JAX-WS and JAX-RS?**  
**Ans:**Java API for XML Web Services(**JAX-WS**)  
Java API for RESTful Web Services (**JAX-RS**)

**11.What are the different application integration styles?**  
**Ans.**There are a number of different integration styles like  
1. Shared database  
2. batch file transfer  
3. Invoking remote procedures (RPC)  
4. Exchanging asynchronous messages over a message oriented middle-ware (MOM).



* **Do we need to make AJAX calls?**(REST can use the XMLHttpRequest)
* **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 band width?** (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)

**14. What tools do you use to test your Web Services?**  
**Ans:SoapUI** tool for SOAP WS and the Firefox “**poster**” plugin for RESTFul services.

**15.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.

**17.What are the different approaches to developing a SOAP based Web service?**  
**Ans. Following are the two approaches.**

* The **contract-first** approach, where you define the contract first with XSD and WSDL and the generate the Java classes from the contract.
* The **contract-last**approach where you  define the Java classes first and then generate the contract, which is the  WSDL file from the Java classes.

**Note:** The WSDL describes all operations that the service provides, locations of the endpoints (i.e. where the services can be invoked), and simple and complex elements that can be passed in requests and responses.

**18.What are the pros and cons of each approach, and which approach would you prefer?**  
**Ans:**  
**Contract-first Web service**  
**PROS:**

* Clients are decoupled from the server, hence the implementation logic can be revised on the server without affecting the clients.
* Developers can work simultaneously on client and server side based on the contract both agreed on.
* You have full control over how the request and response messages are constructed — for example, should “status” go as an element or as an attribute? The contract clearly defines it. You can change OXM (i.e. Object to XML Mapping) libraries without having to worry if the “status” would be generated as “attribute” instead of an element. Potentially, even Web service frameworks and tool kits can be changed as well from say Apache Axis to Apache CXF, etc

**CONS:**

* More upfront work is involved in setting up the XSDs and WSDLs. There are tools like XML Spy, Oxygen XML, etc to make things easier. The object models need to be written as well.
* Developers need to learn XSDs and WSDLs in addition to just knowing Java.

**Contract-last Web service**  
**PROS:**

* Developers don’t have to learn anything related to XSDs, WSDLs, and SOAP. The services are created quickly by exposing the existing service logic with frameworks/tool sets. For example, via IDE based wizards, etc.
* The learning curve and development time can be smaller compared to the Contract-first Web service.

**CONS:**

* The development time can be shorter to initially develop it, but what about the on going maintenance and extension time if the contract changes or new elements need to be added? In this approach, since the clients and servers are more tightly coupled, the future changes may break the client contract and affect all clients or require the services to be properly versioned and managed.
* In this approach, The XML payloads cannot be controlled. This means changing your OXM libraries could cause something that used to be an element to become an attribute with the change of the OXM.

**19.So, which approach will you choose?**  
**Ans:**The best practice is to use “**contract-first**“, and here is the link that explains this much better with examples –>  contract-first versus contract-last web services In a nutshell, the contract-last is more fragile than the “contract-first”.  You will have to decide what is most appropriate based on your requirements, tool sets you use, etc.

- See more at:

http://www.javaface.com/19-restful-web-service-interview-questions/#sthash.3M21we0i.dpuf