**Questions/Answers for PM3**

1. **When should we use REST and when should we use SOAP?**

**REST** is used in scenarios when one wanted to use only HTTP as a communication protocol. REST is usually preferred for services that are exposed as public APIs. If you are writing code in a scripting language such as JavaScript, Python, Ruby or an older language such as VBA, or you are concerned about the size/format of data being sent, then REST is for you. REST is particularly useful for restricted-profile devices such as mobile and PDAs for which the overhead of additional parameters like headers and other SOAP elements are less.

**SOAP** can be used when you wanted to use transport protocol not just HTTP. When there is a major security concern SOAP can be used. SOAP is preferred in enterprise environment where APIs are used locally and does not contain any public calls. If you are writing code in a compiled language (Java, C#) that has a strong SOAP library in place and you don't want to worry about how the data gets sent to/from, then SOAP may be the easiest option since all the SOAP functions become standard functions in the library being used. If you are having an application that involves transaction management, SOAP is always a better option, providing support for Atomic Transactions. We can also use SOAP when publishing a complex API to the public.

**REFERENCE**

<http://stackoverflow.com/questions/1368014/why-do-we-need-restful-web-services>

<http://www.infoq.com/articles/rest-soap-when-to-use-each>

<http://stackoverflow.com/questions/90451/why-would-one-use-rest-instead-of-soap-based-services>

<http://stackoverflow.com/questions/6166746/why-use-soap-for-webservices>

<http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service>

1. **REST vs SOAP**

|  |  |
| --- | --- |
| **REST** | **SOAP** |
| REST is an architectural style. | SOAP is a protocol. |
| In REST client has a specific list of functions exposed to them. | The biggest benefit of SOAP is that it provides a complete end-to-end RPC style interface with functions and types that are fully discoverable. |
| REST has better performance and scalability. REST reads can be cached. | SOAP based reads cannot be cached. |
| REST is not a protocol but an architectural style. | SOAP is a XML based messaging protocol. |
| REST uses URI to expose business logic. | SOAP uses services interfaces to expose the business logic. |
| REST does not define too much standards like SOAP. | SOAP defines standards to be strictly followed. |
| RESTful web services inherits security measures from the underlying transport. | SOAP defines its own security. |

­

**REFERENCE**

<http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/>

<http://www.javatpoint.com/soap-vs-rest-web-services>

1. **Overview of SOA**

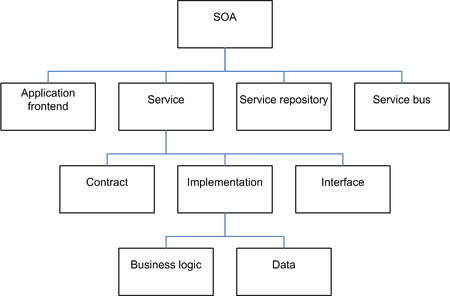
Service-oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature; i.e., the service interface is independent of the implementation. Application developers or system integrators can build applications by composing one or more services without knowing the services' underlying implementations

**Some of the key features of SOA are**

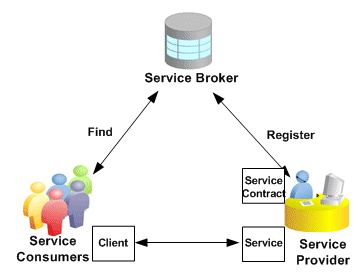
1. SOA use standard WSDL to describe the service.

2. Messages are defined in XML schema. Communication typically happened in heterogeneous environments, with little or no knowledge about the provider.

3. Each SOA service has a quality of service (Qos) associated with it. Some of the key Qos elements are security requirements, such as authentication and authorization reliable messaging, and policies regarding who can invoke services.



**Above is hierarchy of SOA architecture**



**Above diagram shows communication in SOA environment**

**Web Service Approach**

Web service can implement a SOA architecture. They make functional building blocks accessible over standard internet protocols independent of platforms and programming languages. These services can represent either new applications or just wrappers around existing legacy systems to make them network-enabled. On web SOA can be implemented using web services standards like SOAP.

Other SOA concepts includes RPC, REST, DCOM, CORBA, Java RMI, WCF etc.

**REFERENCE**

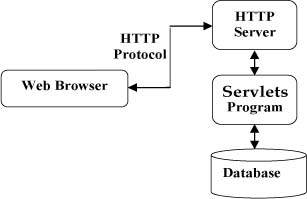
<http://www.javaworld.com/article/2071889/soa/what-is-service-oriented-architecture.html>

<https://en.wikipedia.org/wiki/Service-oriented_architecture>

1. **How to implement web services using servlets?**

**Servlets:** Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP Server. Servlets are used to create dynamic web pages. Java Servlets are Java classes run by a web server that has an interpreter that supports the Java Servlet specification. Servlets can be created using the javax.servlet and javax.servlet.http packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java class library that supports large-scale development projects.

**Servlets Architecture:**



**Implemeting a Servlet as WebService:**

A web service is a service that provides service methods to its clients using either the REST programming paradigm or the SOAP protocol for communication. There are several ways to implement a web service such as using SOAP, Jersey. The most simple way to write a web service would be to write a class and annotate it with the **@WebService** and **@WebMethod** annotations from **javax.jws**, and then launch it from a main-method. The result is that you can view the **WSDL** at the registered URL and if you have SoapUI or any other SOAP client you can also test and use your web service.

A servlet on the other hand is used to transport **HTTP requests** and **responses**. It can be used to write a web application with JSPs and HTML, or to serve XML and JSON responses (as in a RESTful service) and of course also to receive and return SOAP messages. You can think of it as one layer below web services. A more comprehensive and practical approach is to write a web service with a framework and to publish it on an application server or servlet container such as **Tomcat or JBoss**. In this case you would use a Servlet to handle the transport of the HTTP requests which transmit your SOAP or REST messages.

The way to implement web service endpoints (the places that serve responses to your requests) in a Java web server is with servlets. So servlets are simply a technology for implementing web service communication (in the JVM). When you look at JAX-WS and JAX-RS, the frameworks that implement these standards are simply creating abstractions on top of Java's HttpServlet (typically) to make SOAP and REST web services easier to build by taking care of a lot of the low-level details

**REFERENCE**

<http://stackoverflow.com/questions/19254593/servlet-vs-webservices>

<http://www.tutorialspoint.com/servlets/>