

Industrial strength Java/JEE Career Companion to open more doors

[Home](#)
[Java FAQs](#)
[600+ Java Q&As](#)
[Career](#)
[Tutorials](#)
[Member](#)
[Why?](#)
[Can u Debug?](#)
[Java 8 ready?](#)
[Top X](#)
[Productivity Tools](#)
[Judging Experience?](#)

[Home](#) › [member-paid](#) › 10: SOAP Web Service Styles Interview Q&A

10: SOAP Web Service Styles Interview Q&A

Posted on [August 26, 2015](#) by [Arulkumaran Kumaraswamipillai](#)



Q1. What is the difference between the “RPC” and “Document” styles in SOAP?

A1. There are two communication style models that are used to translate a WSDL binding to a SOAP message body.

In **RPC style**, the body of the SOAP request body must contain both the **1)** “operation name and **2)** “method parameters”. The RPC style model assumes a specific structure to the XML instance contained in the message body.

RPC Style: Example 1

```

1 <soap:envelope>
2   <soap:body>
3     <calcInterest>    <!-- web method name -->
4       <a>250.00</a>    <!-- first parameter: amount
```

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

[open all](#) | [close all](#)

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[JEE Interview Q&A \(3](#)

[JEE Overview \(2\)](#)

[Web basics \(8\)](#)

[WebService \(11\)](#)

[01: ♥♦ 40+ Java](#)

[02: ♦ 6 Java RE](#)

[03: ♥ JAX-RS hc](#)

[04: 5 JAXB inter](#)

[05: RESTful We](#)

[06: RESTful Wel](#)

[07: HATEOAS R](#)

[08: REST constr](#)

[09: 11 SOAP We](#)

[10: SOAP Web](#)

[11: ♥ JAX-WS hc](#)

[JPA \(2\)](#)

[JTA \(1\)](#)

[JDBC \(4\)](#)

[JMS \(5\)](#)

```

5         <b>7.0</b>      <!-- second parameter: inter
6     </calcInterest>
7 </soap:body>
8 </soap:envelope>

```

RPC Style: Example 2 (XML tree as a parameter)

In the below RPC example, “CalcInterest” is the name of the method being invoked and “cust” is a parameter of that procedure. Note that “cust” is not namespace-qualified.

```

1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
2   <soap:Body>
3     <x:CalcInterest xmlns:x="http://mycompany.com/x" >
4       <cust>
5         <t:Customer xmlns:t="http://mycompany.com/t">
6           <t:Name>John SMith</t:Name>
7           <t:Id>1234</t:Id>
8           <t:Amount>250.00</t:Amount>
9           <t:Interest>7.0</t:Interest>
10        </t:Customer>
11      </cust>
12    </x:CalcInterest>
13  </soap:Body>
14 </soap:Envelope>

```

The main downside of the RPC style is that it is tightly coupled to the application code. For example, If you want to change the order of the parameters or change the types of those parameters, this change will affect the definition of the web service itself.

Document Style: Example

The advantage of using a **Document style** is that you can structure as you wish as long as the content is XML. The Document style is also referred to as **Message-Oriented** style. In the following XML document, the contract is using XML Schema. The “CalcInterest” may or may not be the name of a remote method being invoked by this message. The “cust” may or may not be the name of a parameter. You only know the structure of the XML document, but not how the service processes it.

```

1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

```

- [JMX \(3\)](#)
- [JNDI and LDAP \(1\)](#)
- [Pressed for time? Java Interview Questions](#)
- [SQL, XML, UML, JSC](#)
- [Hadoop & BigData Interview Questions](#)
- [Java Architecture Interview Questions](#)
- [Scala Interview Q&As](#)
- [Spring, Hibernate, & JPA Interview Questions](#)
- [Testing & Profiling/SaaS Interview Questions](#)
- [Other Interview Q&As](#)
- [Free Java Interview](#)

16 Technical Key Areas

[open all](#) | [close all](#)

- [Best Practice \(6\)](#)
- [Coding \(26\)](#)
- [Concurrency \(6\)](#)
- [Design Concepts \(7\)](#)
- [Design Patterns \(11\)](#)
- [Exception Handling \(3\)](#)
- [Java Debugging \(21\)](#)
- [Judging Experience Interview Questions](#)
- [Low Latency \(7\)](#)
- [Memory Management Interview Questions](#)
- [Performance \(13\)](#)
- [QoS \(8\)](#)
- [Scalability \(4\)](#)
- [SDLC \(6\)](#)
- [Security \(13\)](#)
- [Transaction Management Interview Questions](#)

80+ step by step Java Tutorials

[open all](#) | [close all](#)

- [Setting up Tutorial \(6\)](#)

```

2      <soap:Body>
3          <CalcInterest xmlns="http://mycompany.com/so
4              <cust>
5                  <Customer>
6                      <Name>John SMith</Name>
7                      <Id>1234</t:Id>
8                      <Amount>250.00</Amount>
9                      <Interest>7.0</Interest>
10                 </Customer>
11             </cust>
12         </CalcInterest>
13     </soap:Body>
14 </soap:Envelope>

```

The wsdl will have a binding like

```

1  <wsdl:binding name="CalcIntSvc_Binder" type="tns
2      <soap12:binding style="document" transpo
3
4      <wsdl:operation name="calcInterest">
5          <soap12:operation soapAction="calcIn
6              <wsdl:input>
7                  <soap12:body parts="parameters"
8              </wsdl:input>
9              <wsdl:output>
10                 <soap12:body parts="parameters"
11             </wsdl:output>
12         </wsdl:operation>
13 </wsdl>

```

RPC/literal is a subset of document/literal. This indicates that for any given “RPC/literal” WSDL, you can create a completely equivalent “document/literal” WSDL that would describe the same wire messages.

Q2. What is the difference between the “Encoded” and “Literal” styles in SOAP?

A2. Literal means that the SOAP body follows an XML schema, which is included in the web service’s WSDL document. This means the body contents should conform to a user-defined XSD. The advantages are

1) You can validate the message body with the user-defined XSD

2) You can also transform the message using XSLT.

Encoded message has to use XSD datatypes, but the structure of the message need not conform to any user-

- [Tutorial - Diagnosis \(2\)](#)
- [Akka Tutorial \(9\)](#)
- [Core Java Tutorials \(2\)](#)
- [Hadoop & Spark Tuto](#)
- [JEE Tutorials \(19\)](#)
- [Scala Tutorials \(1\)](#)
- [Spring & Hibernate Ti](#)
- [Tools Tutorials \(19\)](#)
- [Other Tutorials \(45\)](#)

100+ Java pre-interview coding tests

[open all](#) | [close all](#)

- [Can you write code? \(1\)](#)
- [♦ Complete the given](#)
- [Converting from A to I](#)
- [Designing your classe](#)
- [Java Data Structures](#)
- [Passing the unit tests](#)
- [What is wrong with th](#)
- [Writing Code Home A](#)
- [Written Test Core Jav](#)
- [Written Test JEE \(1\)](#)

How good are your?

[open all](#) | [close all](#)

- [Career Making Know-](#)
- [Job Hunting & Resum](#)

defined XSD. This makes it harder to validate the message body or use XSLT for transformations of the message contents. This style is NOT endorsed by the “WS-I” as SOAP body does not follow a schema, but still follows a specific format which may lead to slight differences in the way different programming languages and web service frameworks interpret these formatting rules, resulting in incompatibilities.

Q3. Which style will you use, and why?

A3. The SOAP **RPC-encoded** offers the most simplicity at the cost of tight coupling. It can also suffer from small incompatibilities as the “Encoded” style does not follow the XSD.

The SOAP **RPC-literal** sends XML data to the web service as a single field that is serialized. This is very useful if you already have some data in XML format. The “RPC-literal” will only have one parameter, which is an “XML tree”. The SOAP stack will deal with the transport issues to get the request to the web service, bind the request to the remote object and finally handles the response. However, in RPC-literal, you are more involved with XML parsing compared to “RPC-encoded”

The SOAP **Document-literal** message can contain any sort of XML data that is appropriate to the remote Web service. Unlike the RPC-literal, in SOAP “document-literal”, the developer needs to handle what transport (e.g., HTTP, MQ, SMTP) protocol to use?, marshaling and unmarshaling the SOAP envelope, and parsing the XML in the request and response. The SOAP “Document-literal” is the most difficult for the developers, but it requires lesser SOAP overhead.

Popular Posts

♦ [11 Spring boot interview questions & answers](#)

825 views

♦ [Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers](#)

766 views

[18 Java scenarios based interview Questions and Answers](#)

400 views

001A: ♦ 7+ Java integration styles & patterns
interview questions & answers

388 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions
& answers

295 views

♦ 7 Java debugging interview questions & answers

293 views

01: ♦ 15 Ice breaker questions asked 90% of the time
in Java job interviews with hints

285 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview
Questions and Answers

279 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces
and generics interview questions & answers

239 views

001B: ♦ Java architecture & design concepts
interview questions & answers

201 views

Bio

Latest Posts



Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



About [Arulkumaran Kumaraswamipillai](#)



Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ ♦ Q31 – Q37 JavaScript Interview Q&A on Closure

20+ Maven plugins listed for an enterprise Java (i.e. a JEE) project ▶

Posted in member-paid, Webservice

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)
☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.