Java Web Services面试问题集锦

2012/12/15 | 分类： [基础技术](http://www.importnew.com/cat/basic), [职业生涯](http://www.importnew.com/cat/career) | [0 条评论](http://www.importnew.com/1851.html#respond) | 标签： [WEB SERVICE](http://www.importnew.com/tag/web-service), [面试](http://www.importnew.com/tag/%e9%9d%a2%e8%af%95)

分享到：**20**

本文作者： [ImportNew](http://www.importnew.com/) - [郑雯](http://www.importnew.com/author/zhengwen) 未经许可，禁止转载！

**Q.** 应用集成方式有哪些？

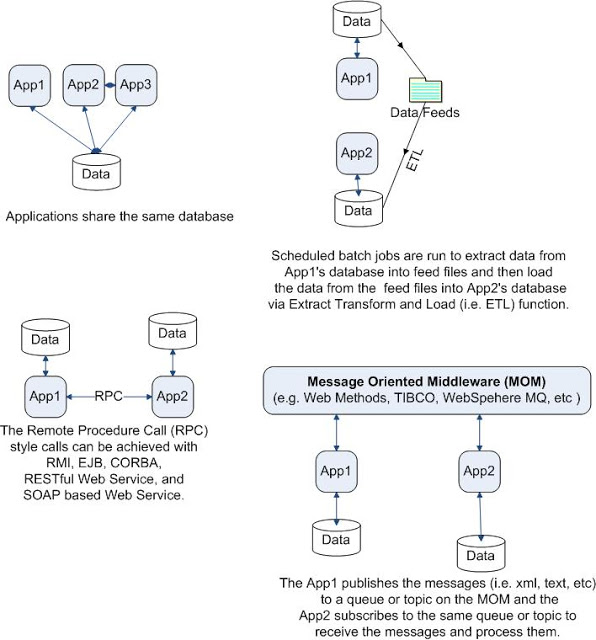
**A.** 应用可以采用以下方式集成：

1. 共享数据库

2. 批量文件传输

3. 远程过程调用（RPC）

4. 通过消息中间件来交换异步信息（MOM）

[](http://www.importnew.com/1851.html/integration-styles)

**Q.**应用集成可以采用的Web服务方式有什么？

**A.** SOAP WS（Simple Object Access Protocal） 和RESTful Web Service（REpresentational State Transfer）

**Q.** SOAP WS 和RESTful Web Service之间有什么不同呢？

**A.**

* SOAP WS支持既远程过程调用（例如，RPC）又支持消息中间件（MOM）方式进行应用集成。而Restful Web Service仅支持RPC集成方式。
* SOAP WS是传输协议无关的。它支持多种协议，比如，HTTP(S)、 Messaging、TCP、UDP SMTP等等。而REST是协议相关的，只支持HTTP或者HTTPS协议。
* SOAP WS仅允许使用XML数据格式。定义的操作通过POST请求发送。其重点是通过操作名来获取服务，并将应用逻辑封装为服务。而REST方式则允许多种数据格式，例如，XML、JSON、文本、HTML等等。而且由于REST方式采用标准GET、PUT、PSOT和DELETE 方法，因此所有的浏览器都可以支持。其重点是通过资源名来获取服务，并将数据封装为服务。AJAX支持REST方式，它可以使用XMLHttpRequest对象。无状态CRUD操作（创建、读、更新和删除）更加适合这种方式。

GET – represent()

POST – acceptRepresention()

PUT – storeRepresention()

DELETE – removeRepresention()

* 无法缓存SOAP方式读取的内容。而REST方式的则可以，而且性能和可扩展性都更好一些。
* SOAP WS支持SSL和WS-security，针对企业级应用可以有更多的安全保障，例如按需提升安全指数、通过第三方来保证身份认证信息的安全性、除了点到点SSL（point to point SSL）之外，更针对消息的不同部分来提供不同的保密算法等等。而REST只支持点到点SSL。而且无论是不是敏感消息，SSL都会加密整条消息。
* SOAP对于基于ACID的短寿命事务管理以及基于补偿事务管理的长寿命事务有深入的支持。同时，SOAP也支持分布式事务（译者：在一个分布式环境中涉及到多个资源管理器的事务）的两阶段提交（two-phase commit）方式。而REST由于基于HTTP协议，因此对于事务处理既不兼容ACID方式也不提供分布式事务的两阶段提交方式。
* 即便是要通过SOAP的第三方程序，SOAP通过内置的重试逻辑也可以提供端到端可靠性。REST没有一个标准的消息系统，因而寄希望于客户通过重连去解决通信失败问题。

**Q.**如何选择采用哪种Web service？SOAP WS还是REST？

**A.**一般而言，基于REST的Web service的优势在于其简单、性能不错、可扩展性好，并且也支持多种数据格式。而SOAP则适用于安全性和事务处理可靠性方面要求比较高的服务中。

对于这个问题的答案，更多的考虑依据是设计者对功能性和非功能性需求的要求。通过回答下列问题可以帮助你做出选择：

* **所提供的服务会暴露数据或者业务逻辑吗？**（如果会暴露数据的话可以选择REST方式，如果会暴露业务逻辑的话可以选择SOAP WS）。客户或者服务提供商需要一个正式的契约（contract）吗？（SOAP可以通过WSDL（Web Service Description Language）提供一个正式契约）
* **需要支持多种数据格式吗？**
* **需要进行AJAX调用吗？**（REST可以采用XMLHttpRequest来发送AJAX调用）
* **同步调用还是异步调用？**
* **有状态调用还是无状态调用？**（REST适合无状态CRUD操作）
* **对于安全性的要求？**（SOAP WS对于安全性的支持更好些）
* **对于事务处理的要求？**（SOAP WS这方面更有优势）
* **有带宽限制吗？**（SOAP消息比较冗长）
* **哪种方式更适合开发者呢呢？**（REST更好实现，也更好测试和维护）

**Q.**有什么可以用来测试Web Service的工具吗？

**A.**测试SOAP WS可以使用SoapUI，测试RESTFul service可以采用Firefox的“poster”插件。

**Q.**SOA和Web service的区别是什么？

**A.**SOA是一种软件设计准则，一种实现松耦合，高可复用性和粗粒度的web服务的[设计模式](http://www.amazon.cn/gp/product/B001130JN8/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&tag=importnew-23&linkCode=as2&camp=536&creative=3200&creativeASIN=B001130JN8)。开发者可以选择任意协议实现SOA，例如，HTTP、HTTPS、JMS、SMTP、RMI、IIOP（例如，采用IIOP的EJB）、RPC等。消息可以采用XML或者数据传输对象（Data Transfer Objects，DTOs）。

Web Service是实现SOA的技术之一。也可以不用Web service来实现SOA应用：例如，用一些传统的技术，像Java RMI，EJB，JMS消息等。但是Web service提供的是标准的平台无关的服务，这些服务采用HTTP、XML、SOAP、WSDL和UDDI技术，因此可以带来J2EE和.NET这些异构技术（heterogeneous technologies）之间的互操作性。

**Q.**如果可以使用传统的中间件方式，例如，RPC、CORBA、RMI和DCOM，为什么还要选择Web service？

**A.**传统的中间件跟应用关系紧密，对应用的任何修改都可能会引起对应中间件的修改。因此这种方式下应用不好维护，复用性也比较差。一般情况下也无法支持异质系统（heterogeneity）。另外，采用这种方式应尽量避免互联网访问应用。还有，这种方式代价更高并且可用性差。

Web service采用松耦合连接，即在客户端和服务器端之间提供了一个抽象层。这种松耦合应用可以减少维护成本并增加可复用性。Web service提供了一种基于XML和Web的新的中间件形式。Web service是语言和平台无关的，任何语言都可以用来开发一个Web service，并且部署到任何平台上，包括小型设备到大型超级计算机。Web service采用语言无关协议，例如HTTP，并且通过Web API来在不同的应用程序之间传输XML消息。Web service可以通过互联网访问，开销少并且可用性好。

**Q.**开发基于SOAP的Web service有哪些方式呢？

**A.**有两种方式;

* 契约先行（也称为自顶向下，contract-first）方式：通过XSD和WSDL来定义contract，然后根据contract生成Java类；
* 契约后行（也称为自底向上，contract-last）方式：先定义Java类，然后生成约定，也就是从Java类得到WSDL文件。

注意：WSDL描述这样一些信息：服务所提供的所有用户操作、终端位置信息（例如，调用服务的URL），请求和响应中的简单或者复杂元素等。

**Q.**上面两种方式各有什么优缺点吗？你更推荐哪种？

**A.**

**契约先行方式的Web service**

优点：

* 客户端程序和服务器端程序分离，因此[重构](http://www.amazon.cn/gp/product/B003BY6PLK/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&tag=importnew-23&linkCode=as2&camp=536&creative=3200&creativeASIN=B003BY6PLK)服务器端代码不会影响到客户端。
* 由于遵守相同的规范，因此客户端和服务器端的开发可以并行进行。
* 开发者可以控制请求\响应消息的结构：例如，“status”应该是作为消息的一个元素还是一个属性？契约规定的非常明确。因此开发者可以大胆的去修改OXM（Object to XML Mapping）库，而不用担心是否会导致“status”从元素变成“属性”。不仅如此，甚至Web service框架和工具箱都可以更换，比如从Apache Axis变成Apache CXF等等。

缺点：

* 开发前期需要额外的一些搭建XSD和WSDL的工作。使用XML Spy、OxygenXM等工具可以简化这些工作。另外，还需要开发对象模型。
* 开发者需要除了Java之外，还需要去学习XSD和WSDL。

**契约后行方式的Web service**

优点：

* 开发者不用去学习XSD、WSDL和SOAP相关知识。可以通过框架或者工具集利用已有服务来快速构建新的服务。例如，通过基于IDE向导快速构建应用。
* 学习曲线和开发时间会比契约先行方式小。

缺点：

* 项目初始的开发时间会缩短，但是一旦contract发生变化或者需要加入新的元素，那么随之而来的维护和扩展应用所带来的开发时间会是怎样呢？采用这种方式，由于客户端和服务器端之间紧密耦合，因此未来潜在的变化可能破坏客户端的contract，导致所有的客户都会受到影响，为了避免这中情况的发生，项目开发时需要很小心的开发和管理未来要发布的服务。
* XML的有效负载（payload）无法控制。这也就是说修改应用的OXM库会导致某个元素错误的变成了属性。

**那么，你会选择哪一种方式呢？**

最好的方式是“契约先行”（contract-first），这里有篇[文章](http://static.springsource.org/spring-ws/site/reference/pdf/spring-ws-reference.pdf)（[ontract-first versus contract-last web services](http://static.springsource.org/spring-ws/site/reference/pdf/spring-ws-reference.pdf) ）用一些例子来解释为什么。总的来说，契约先行方式比契约后行方式开发出来的应用更加强壮。当然，选择哪种方式是跟需求和工具等等有关的，需要综合这些因素考虑决定。