

## Web Services

Por que os Web Services são atrativos para a integração de sistemas?

Pois os Web services são componentes que possibilitam que as aplicações se comuniquem utilizando protocolos padrão da internet (XML, por exemplo). Os recursos de uma aplicação ficam disponíveis de forma normalizada mesmo que as aplicações estejam em ambientes diferentes descritas em diferentes linguagens.

O que se espera para o futuro dos Web Services?

Acredita-se que no futuro as empresas poderão publicar seus Web services em diretórios públicos utilizando protocolos padrão, de onde poderão ser vendidos como serviços para outras empresas, instituições ou usuários comuns que poderão combinar esses serviços de forma a prover outros serviços. Contudo, algumas modificações no modelo atual devem ser realizadas para que isso aconteça.

Quais são as tecnologias mais usadas atualmente no lugar de XML/SOAP em Web Services? Ainda faz sentido utilizar a WSDL quando se substitui XML/SOAP por essas alternativas?

As tecnologias são respectivamente JSON e REST.

O uso da WSDL ainda faz sentido, uma vez que essa descreve as funcionalidades e interface do serviço, e a maneira como são requisitados os serviços e passados os dados é independente da interface, podendo ser especificada (a partir da versão 2.0 a definição da WSDL passa a permitir o uso dos quatro verbos do HTTP e fica mais adequada para a descrição de serviços REST). O uso de XML é, no entanto, ainda o mais comum quando se utiliza WSDL.

Com o descritor em WSDL de um Web Service é possível gerar automaticamente código que fornece ou que utiliza os serviços descritos. O caminho inverso, isto é, a criação automática de um descritor em WSDL também é possível a partir do código de um servidor e da marcação das funções que deverão ser disponibilizadas (como com a marcação remote em Java).

Cite vantagens do uso de JSON e REST no lugar da pilha XML/SOAP/WSDL/UDDI.

Ambos são mais leves, permitindo maior eficiência.

JSON é código JavaScript válido, e está frequentemente mais próximo de objetos de linguagens de programação do que XML (que oferece mais funcionalidades, como namespaces, que por sua vez exigem parsers mais complexos, o que torna o processamento mais lento). Isso torna o uso de JSON favorável à eficiência do Web Service quando este não fizer proveito das capacidades a mais providas pelo XML.

REST se baseia no uso dos verbos de HTTP (GET, POST, PUT e DELETE) para comunicação entre cliente e servidor, sendo os Web Services acessados através de URIs, o que utiliza automaticamente facilidades providas pelo protocolo HTTP, como caching (o que é mais útil quando o serviço não depende de consultas anteriores, i.e. é stateless). É mais simples, sendo vantajoso quando funcionalidades específicas do SOAP não são necessárias.

O que são Web Services?

É uma solução que procura resolver o problema de comunicação entre aplicações que empregam diferentes tecnologias. Temos um web service quando disponibilizamos um serviço de aplicação por meio de uma rede de forma normalizada, onde a troca de dados é feita no formato XML. A ideia é oferecer uma funcionalidade no estilo “black box” que segue o padrão para recebimento de dados, executando assim a operação requisitada e então enviando dados de retorno, caso seja necessário, no mesmo padrão. A criação desse padrão dá a ideia de uma “linguagem universal” para a comunicação entre serviços e aplicações, onde todos os dados transmitidos usando essa tecnologia são convertidos para o mesmo formato.

Quem define os padrões de Web Services?

Os dois maiores responsáveis são o W3C e a OASIS. Entretanto, grandes empresas como Microsoft e IBM apoiam e tem certa influência sobre o destino da tecnologia.

Como funciona a segurança em Web Services?

Contrariando a premissa inicial de ser um padrão universal de interoperabilidade, não existe um consenso entre as empresas e cada uma acaba optando por uma

implementação. No entanto, existem dois tipos básicos de mecanismos de segurança:

O primeiro deles é no nível da mensagem, onde extensões para o padrão SOAP são usadas para prover autenticação, integridade e privacidade para as mensagens, adicionando tags extras e portanto um overhead de comunicação e processamento para cada mensagem.

Outra solução é utilizar uma camada extra de rede que provê segurança para os principais protocolos utilizados como HTTP, SMTP etc. Essa camada, chamada de TSL - Transport Layer Security, baseia-se em criptografia e chaves públicas para troca de mensagens.

Comparação de Desempenho:

Devido às suas características de interoperabilidade, web services geralmente são mais lentos que o JAVA RMI. Além disso, padrões de extensão de segurança SOAP como o WS-Security deixam a troca de mensagens ainda mais lenta. A tabela abaixo mostra uma comparação entre as tecnologias:

Corba

O que é necessário implementar em uma linguagem para que ela suporte a arquitetura CORBA?

Um compilador de IDL que gere código nativo para a linguagem.

Um ORB para ser o “servo”, falar com outros ORBs e repassar as chamadas de métodos para os objetos.

Uma implementação do protocolo IIOP nos ORBs.

Qual o problema que ambos RMI e CORBA enfrentam na hora de fazer a comunicação? Qual a solução?

Ambos usam protocolos construídos sobre o TCP e que geram tráfego em portas arbitrárias, que geralmente estão bloqueadas por firewalls em sistemas empresariais (onde estas tecnologias são geralmente usadas). Soluções possíveis

são a configuração do firewall, o tunelamento do tráfego por outro protocolo (provavelmente HTTP usando a porta 80) ou a troca de tecnologias. Para falar a verdade este é um dos aspectos mais citados para exemplificar as vantagens dos Web Services, que usam tráfego HTTP por padrão.

Qual é a função do POA (Portable Object Adapter) em CORBA?

Ele é um módulo entre um ORB e um servo, que disponibiliza uma interface consistente para o ORB interagir com o código do servo. Dentre suas possíveis funções estão gerar referências de objetos, demultiplexar chamadas a servos específicos, ativar e desativar servos e invocar operações em servos.

Porque é necessário a utilização de IR (Interface Repository) em CORBA?

O IR é uma fonte de dados das interfaces de cada objeto que um ORB reconhece. Essas interfaces são obtidas dinamicamente pelo ORB para que ele possa manipular os objetos registrados nele.

Qual a principal desvantagem que RMI apresenta em relação a CORBA?

Remote method invocation é uma poderosa inovação da linguagem java pois permite que os programadores invoquem métodos de objetos remotos, isto é, ao invés de usar chamadas de procedimentos onde parâmetros de tipos primitivos (ou estruturas destes tipos) devem ser passados, objetos inteiros podem ser passados como parâmetro. Desta maneira é possível que uma JVM remota obtenha objetos sem conhecer a classe da qual são instâncias, possibilitando assim, que código novo seja passado remotamente. Porém, como RMI é dependente de máquinas virtuais, ao interagir com sistemas legados como C, C++, FORTRAN, etc., problemas podem ocorrer.

Qual a principal desvantagem que CORBA apresenta em relação a RMI?

O problema visto no caso do RMI pode ser contornado através da Common Object Request Broker Architecture. Esta tecnologia de sistema distribuido permite a

portabilidade entre diferentes linguagens, desde que para cada uma das linguagens exista uma interface escrita em IDL (Interface Definition Language). Porém, CORBA é menos poderosa que RMI; somente tipos primitivos e estruturas destes tipos podem ser passadas remotamente, e isto implica no fato de que código novo não pode ser passado.

Como funcionam transações no contexto de Web Services?

Transações em web services não necessariamente aderem a todas propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

Para seguir essas propriedades os seguintes cenários seriam encontrados:

Recursos utilizados pelos Web Services seriam bloqueados por um período indefinido de tempo, até que todos os participantes da transação concluíssem;

Um gerenciador de transações centralizado controlaria os serviços e coordenaria suas ações. Com isso, o serviço perderia o controle sobre seus recursos.

Estes cenários deixam os Web Services expostos a ataques DoS.

Para superar esses problemas, existem várias propostas de controle de transação.

O OASIS propõe 3 especificações inter-relacionadas:

WS-Coordination - especifica o protocolo para serviços que criam um contexto de coordenação, identificando unicamente as atividades e registrando seus participantes;

WS-AtomicTransaction - especifica protocolos concretos para transações atômicas distribuídas, usando two-phase commit (ACID);

WS-BusinessActivity - especifica um protocolo para atividades de longa duração, usando um protocolo de compensação.

Para transações do tipo Business Activity, as ações são realizadas pro cada web service participante.

Em caso de rollback, operações de compensação são disparadas para desfazer as atividades.

A decisão de commit ou rollback fica associada à lógica do negócio, tornando desnecessário que todos os participantes completem suas atividades com sucesso.

Qual a pilha de protocolos utilizadas em Web Services?

A pilha de protocolos permite variações e está em constante evolução.

Existem 4 camadas principais:

Transporte: responsável por transportar mensagens entre aplicações. Protocolos: HTTP, SMTP, FTP e outros.

Mensagem: responsável por codificar as mensagens em um formato XML comum. Protocolos: SOAP e XML-RPC

Descrição: responsável por descrever a interface do Web Service. Protocolo: WSDL.

Descoberta: responsável por centralizar a informação de web services, possibilitando a publicação e descoberta de serviços disponíveis na rede. Protocolo: UDDI.