



sensedia

---

...

# Revista Mundo Java

Setembro/09

**Sensedia**

Rod. SP – Campinas/Mogi Mirim Km 118,5 - Prédio 9C

CEP: 13086-902 Campinas, SP

(19) 3705 – 5775

[contato@sensedia.com](mailto:contato@sensedia.com)

[www.sensedia.com/br](http://www.sensedia.com/br)

REVISTA  
**MUNDOJAVA**  
A Revista do Desenvolvedor Java!



**Marcelo Oliveira**

marcelo@coluna.com.br. Gerente de consultoria de Serviços - <http://www.colunainformatica.com.br>, engenheiro especializado em implantação corporativa de SOA. Participa de diversos projetos europeus de adoção SOA. Graduado pela Universidade Federal de Viçosa (UFV) e mestre em sistemas distribuídos pelo Universidade Estadual de Campinas (Unicamp), é professor de pós-graduação com ênfase em SOA e WebServices na Faculdade de Tecnologia (FATEC), Assisburgo. É um dos responsáveis pelo "Aquele Blog de SOA" ([www.aqueleblogsoa.com.br](http://www.aqueleblogsoa.com.br)) e realiza treinamentos de plano de capacitação de Serviços.

**Rafael Navarro**

rafael@coluna.com.br. Arquiteto de software da Servedia, onde é responsável por projetos de análise de qualidade e implantação de componentes e serviços e fornecimento de integrações e interoperação SOA. É graduado pela UFV mestrado em Engenharia Computação pelo Unicamp na área de Engenharia de Software, mestrando a SOA e atuando no "Aquele Blog de SOA".

Uma visão rápida dos desafios e principais padrões de interoperabilidade para implementação de projetos com Arquitetura Orientada a Serviço

*A interoperabilidade de serviços é uma característica que permite o acesso e consumo dos serviços de maneira transparente e independente de tecnologia. Como SOA é uma abordagem de adoção corporativa, frequentemente encontramos cenários tecnologicamente heterogêneos e é fundamental que este fato não impossibilite a utilização de serviços de maneira geral. Dessa forma, a busca por interoperabilidade é contínua e muitas vezes necessária. Neste artigo, são apresentados alguns conceitos relacionados, mas principalmente os padrões de interoperabilidade mais indicados na implementação de serviços em aplicações SOA.*

**D**entre várias questões a serem discutidas sobre Arquitetura Orientada a Serviços, interoperabilidade é palavra certa no momento de definições arquiteturais. As promessas são tantas que, às vezes, pensamos que interoperabilidade é um efeito direto e gratuito quando utilizamos serviços, e definitivamente isso não é verdade.

O objetivo deste artigo é levantar uma discussão baseada na importância da interoperabilidade em SOA. Quão importante é a existência de interoperabilidade em SOA? Web Services garantem a interoperabilidade? E, se não garantem, quem pode garantir? Além disso, apresentamos os principais padrões de interoperabilidade usados na implementação de serviços, disponibilizados pela WS-I (Web Services Interoperability Organization - [www.ws-i.org](http://www.ws-i.org)).



## Um pouco sobre arquitetura orientada a serviços

O objetivo deste artigo não é apresentar conceitos de SOA. Uma busca rápida pela internet nos retornará diferentes livros e referências e, consequentemente, diversas definições. Porém, como uma boa prática, antes de entrar em detalhes de interoperabilidade, é importante formalizar uma definição de SOA:

SOA é uma abordagem arquitetural corporativa que permite a criação de aplicações que disponibilizem serviços de negócio, que possam ser facilmente coordenados, reusados e compartilhados.

Esta é uma entre várias definições que podemos encontrar na literatura especializada e uma definição que pode ser evoluída, melhorada e até questionada. Sem levantar questões, vamos apenas entender, dividindo em três partes: SOA é...

1. **... abordagem arquitetural corporativa...** SOA não é ferramenta, não é tecnologia, não é processo (apenas). SOA na verdade é a abordagem sob a qual se constrói a arquitetura das aplicações, objetivando que a arquitetura seja orientada a serviços identificados a partir do negócio. Ou seja, os serviços que a aplicação disponibiliza não são resultados que a TI pode oferecer, e sim que o negócio de fato precisa.
2. **... criação de serviços de negócio...** É fundamental que estes serviços sejam serviços de negócio. Tudo bem que serviços técnicos existem, mas uma das grandes vantagens é poder apresentar serviços que possuam valor de negócio. Daí a importância de se envolver pessoas de negócio nos grupos de decisão relacionados a SOA (na referência existem muitos nomes para este grupo: CoE – Center of Excellence, Núcleo SOA, ICC-Innovation Competence Center, e por aí vai). O fato é que os serviços para SOA são de negócio. Pessoas que nunca se falaram deverão trabalhar em conjunto (analista de negócio e arquiteto).
3. **... possam ser reusados e compartilhados...** Dentre os milhares de benefícios vislumbrados para SOA, temos desde agilidade até maturidade de integração, mas o reuso é um efeito colateral que deve ser fortemente considerado. Mais uma vez, a interoperabilidade é muito importante, para que os serviços de fato "criem uma camada de abstração tecnológica", que possam ser reusados entre diferentes aplicações e processos de negócio.

Agora que nos alinhamos sobre o que é SOA (pelo menos neste artigo), podemos ir em frente. O objetivo é focarmos em interoperabilidade e deixar para os próximos artigos outras características, como baixo acoplamento, identificação e deploy de serviços, governança runtime, dentre outras.

## Definindo interoperabilidade

Começando por uma definição interessante para o termo na Wikipedia: interoperabilidade é a capacidade de um sistema (informatizado ou não) de se comunicar de forma transparente (ou o mais próximo disso) com outro sistema (semelhante ou não). Para um sistema ser considerado interoperável, é muito importante que ele trabalhe com padrões abertos. Seja um sistema de portal, seja um sistema educacional ou ainda um sistema de comércio eletrônico, ou e-commerce, hoje em dia se caminha cada vez mais para a criação de padrões para sistemas (<http://pt.wikipedia.org/wiki/Interoperabilidade>).

Tecnologicamente, existe interoperabilidade quando a utilização dos serviços segue padrões agnósticos, independentes de tecnologia, deixando sua utilização transparente para diferentes clientes em diferentes tecnologias. A utilização de Web Services, com certeza, é uma grande aposta para se atingir este objetivo. Porém, como veremos adiante, a adoção simples de Web Services não garante sempre a interoperabilidade do serviço.

## Por que a interoperabilidade é tão importante, mas não suficiente?

Dentre os diversos benefícios de utilização de SOA na construção ou integração de aplicações, podemos destacar: agilidade, flexibilidade de processo, economia, time-to-market etc. A interoperabilidade garante a visão transparente para acesso aos serviços disponibilizados por TI, fornecendo uma camada de abstração acima das plataformas tecnológicas, influenciando inclusive no baixo acoplamento. Visando interoperabilidade, devem ser definidos os padrões de interface, mas, sobretudo, padrões de implementação, sobre os quais vamos falar mais adiante.

Porém, é importante deixar claro que a interoperabilidade não é suficiente para uma adoção de SOA com sucesso. Por exemplo, para que os serviços sejam compostos e replicados em novos contextos é importante que exista baixo acoplamento. Porém, apenas a independência de tecnologia não garante a existência de baixo acoplamento nos serviços (que vai muito além da tecnologia). Para entender melhor sobre baixo acoplamento e seus diversos tipos, leiam artigo de José Ventura, em: <http://www.aqueleblogdesoa.com.br/2008/07/baixo-acoplamento>.

## Web Services e a promessa da interoperabilidade

Em vários artigos já foi discutida a relação entre SOA e Web Services. Pergunta comum: com Web Services, existe SOA? A resposta é: não! Os Web Services são, muitas vezes, a melhor forma de implementar serviços para SOA. Em novos projetos, é bastante comum a utilização de Web Services para viabilizar as comunicações distribuídas (salvo quando temos alguns requisitos fortes, como performance).

Uma vez que a decisão por utilizar Web Services é feita, vem a segunda pergunta: com Web Services, a interoperabilidade está garantida? E a resposta é: não! Os Web Services utilizam padrões baseados em XML (SOAP, WSDL, UDDI), mas infelizmente, é possível (e até comum) encontrar Web Services que não são interoperáveis. Geralmente, por dois motivos:

1. Foram implementados sem considerar boas práticas de desenvolvimento, usando objetos não-serializáveis nas interfaces (ou específicos de plataforma). É como usar um Dataset em um Web Service construído em .NET. Este é um cenário cada vez mais raro.
2. Quando precisamos de features mais avançadas, como segurança, gerência de políticas, transação distribuída, dentre outras, as equipes de implementação podem optar por soluções proprietárias e não baseadas em padrões abertos. Este é um cenário cada vez mais comum.

Com a forte adoção de Web Services, há alguns anos ainda não havia padrões interoperáveis para algumas necessidades, como segurança na mensagem e transação distribuída. Isso fez com que os fornecedores implementassem seus próprios mecanismos, comprometendo a interoperabilidade dos serviços disponibilizados. Ai surgiram os Ws-Security, Ws-Transaction/Coordination, dentre outros. Mas isso não quer dizer que a coisa ficou fácil.



Frete à necessidade de garantir segurança em um serviço na mensagem, podemos usar os seguintes recursos:

1. criptografar usando uma biblioteca Java. O que pode romper a interoperabilidade do serviço;
2. criar um mecanismo baseado no WS-Security, que utilize os padrões propostos, sem dependência de tecnologia.

Na primeira, tenho grandes chances de fazer uma implementação que empacote a criptografia no SOAP de uma forma que não consegue ser lida por outra tecnologia.

Vale lembrar que a utilização de HTTPS/SSL garante segurança no canal, porém o foco é garantir segurança dentro da mensagem, conforme veremos adiante, em WS-Security.

O padrão WS-Security é independente de tecnologia e será apresentado mais adiante.

### ↳ Padrões de interoperabilidade (Ws-\*).

Felizmente existem iniciativas para definir padrões interoperáveis para as principais necessidades relacionadas aos Web Services. Os principais são trabalhados pela WS-I, organização que reúne mais de 100 empresas com o objetivo de assegurar a interoperabilidade de Web Services. Algumas das metas da WS-I são:

1. integrar especificações;
2. promover implementações consistentes;
3. oferecer guias e boas práticas de implementação;
4. fornecer ferramentas e aplicações de exemplo;
5. encorajar a adoção através de consenso.

Dentre os principais padrões de interoperabilidade apoiados pela WS-I, vamos citar rapidamente Ws-Addressing, WS-Policy, WS-Transaction e WS-Security, que são os padrões mais maduros e utilizados da WS-I. Nosso objetivo não é detalhar cada um, é apenas apresentar o conceito, abrangência e principais funcionalidades.

### ↳ Ws-Addressing

Como o protocolo SOAP não fornece nenhuma informação sobre os endereçamentos de origem e destino da mensagem, diversos problemas no transporte podem acontecer. Surgiu assim a necessidade de se utilizar uma especificação para garantir um meio de transporte neutro na própria mensagem SOAP que fosse capaz de endereçar a origem e destino da mensagem de forma independente da camada de transporte. Para solucionar este problema, foi criada a especificação WS-Addressing.

Esta especificação define duas construções de informações que são normalmente encontradas nos protocolos de transporte, são elas:

- **endpoint-reference:** é um mecanismo que descreve o lugar na rede que pode receber mensagem SOAP. Os endpoints-reference definem um nível mais detalhado de informações que uma URL convencional;
- **message information header:** adicionam propriedades à mensagem, permitindo assim o endereçamento e interação entre serviços.

Segue um exemplo da utilização da especificação WS-Addressing:



Figura 1. Exemplo de WS-Addressing.

A figura 1 apresenta alguma das tags definidas pelo padrão WS-Addressing.

**<MessageID>** Representa o identificador da mensagem, responsável pela identificação da mensagem em um ambiente ou aplicação.

**<ReplyTo>** Especifica o endpoint reference para onde deverá ser enviada a resposta para esta mensagem.

**<To>** Especifica o endpoint reference destino desta mensagem.

**<Action>** Identifica a semântica da mensagem. Em outras palavras, "amarra" a mensagem com o portType do WSDL identificando se a mensagem é um <input>, <output> ou <fault>.

### ↳ WS-Policy

É uma especificação recomendada pelo World Wide Web Consortium (W3C) que permite ao Web Services definir políticas adicionais que devem ser cumpridas pelo cliente e o prestador de serviço para que a interação entre eles possa acontecer. Estas validações de políticas são feitas de forma automática e em tempo de execução.

Algumas das principais características do WS-Policy são:

- fornecer um modelo de propósito geral para descrever políticas;
- prover uma gramática flexível e extensível que permite descrever uma ampla variedade de requisitos e habilidades para o ambiente dos Web Services;
- em conjunto com o WSDL fornece uma descrição de requisitos que o serviço possui que devem ser cumpridos pelos requisitantes.

O WS-Policy é um conjunto de políticas representadas através de documentos XML, no qual o elemento principal é o Policy. Dentro deste elemento, são definidas as coleções de asserções, que quando combinadas definem uma política. Existem dois tipos principais de combinações de políticas:

- **ExactlyOne** — indica que somente uma das asserções será apresentada;
- **All** — permite que várias asserções sejam representadas como alternativas políticas.

Em um ambiente heterogêneo, com diferentes tecnologias implementando serviços disponibilizados em um barramento corporativo,

é importante centralizar a implementação de políticas de acesso aos serviços. Se as políticas forem implementadas dentro dos serviços, a manutenção e evolução de cada política serão custosas e dependentes da tecnologia. Por outro lado, se as políticas forem centralizadas, devem ser implementadas usando um padrão interoperável, solução fornecida pelo WS-Policy.

Atualmente existem middlewares que podem cuidar da implementação das políticas de maneira centralizada, alguns deles usam WS-Policy, ou formatos proprietários.

### WS-Transaction

Com o aumento de interconexões entre sistemas e distribuição de serviços pela web, podemos facilmente atingir um grau de integração que garante a agilidade desejada pelas empresas. Porém, algumas facilidades que não nos preocupavam há tempos, agora voltaram, como é o caso do controle de transação. Suponha que uma empresa de viagens resolveu integrar o sistema de venda de passagens aéreas e reserva de hotel, onde cada uma seja disponibilizada por Serviços Web. Tendo esse cenário, suponha que você compre uma passagem aérea, porém, quando tenta reservar um quarto de hotel ocorre um problema. Assim temos uma falha no controle de transação, pois teremos que garantir o cancelamento na reserva da passagem efetuada pelo serviço anterior.

Essas são preocupações que não tínhamos mais em desenvolvimento de aplicativos tradicionais com banco de dados, ou seja, requisitos que antes poderiam ser resolvidos pelo SGBD (como garantia de atomicidade nas transações), com o contexto de aplicação distribuída e atividades sendo realizadas através de serviços remotos, devem ser repensados. Pensando em um contexto no qual os serviços que compõem a aplicação estão implementados em diferentes tecnologias, mais uma vez é fundamental a adoção de um padrão interoperável, que atue como uma camada superior de abstração e seja capaz de garantir características transacionais na "chamada" de Web Services remotos.

A especificação WS-Transaction define mecanismos para interoperabilidade entre domínios de Web Services fornecendo um meio para compor a qualidade de serviços transacionais entre aplicações Web Services.

Esta especificação descreve um framework de coordenação extensível (WS-Coordination) e tipos de coordenação específicos para:

- transações de curta duração, ACID (WS-AtomicTransaction);
- transações de negócio de longa duração (WS-BusinessActivity).

Devido à importância destes três conceitos (WS-Coordination, WS-AtomicTransaction e WS-BusinessActivity), vamos citar rapidamente cada um.

#### WS-Coordination

É comum, em aplicações distribuídas, que a execução das atividades exija interações complexas entre os componentes de um sistema. No exemplo citado, a ordem em que as requisições são efetuadas é muito importante. Por exemplo, não faz sentido realizar a reserva do hotel se não foi possível comprar a passagem aérea. Neste caso, um mecanismo

de coordenação é necessário para assegurar que as requisições sejam executadas numa ordem correta. Trata-se, portanto, de um exemplo de atividade que precisa ser coordenada.

Atividades coordenadas envolvem três papéis: o coordenador, o participante e o protocolo de coordenação. O coordenador atua como um mediador entre os participantes. O protocolo define as mensagens que podem ser trocadas entre o coordenador e os participantes, as regras que as mensagens devem obedecer e a semântica das mensagens trocadas entre os envolvidos. A figura 2 ilustra esse conceito.



Figura 2. Protocolo de troca de mensagens entre coordenador e os participantes.

O WS-Coordination é uma especificação para Web Services que define um coordenador de atividades genérico e extensível. Esta especificação descreve um framework extensível pelo protocolo de fornecimento que coordena as ações de aplicações distribuídas. Tal protocolo de coordenação é utilizado para suportar um número de aplicações, incluindo aquelas que necessitam alcançar consistentes acordos sobre o resultado das atividades distribuídas e uma fundação para outras especificações, tais como WS-AtomicTransaction e WS-BusinessActivity. O seu coordenador genérico possui duas funções principais:

- permite a criação de contextos de coordenação;
- fornece uma infraestrutura para o registro dos participantes da transação.

#### WS-AtomicTransaction

Esta especificação fornece a definição do tipo de coordenação de transação atômica, que é para ser usada com o framework de coordenação extensível descrito na especificação WS-Coordination. WS-AtomicTransaction define:

- um tipo de coordenação, representado pelo URI <http://schemas.xmlsoap.org/ws/2004/10/wsat>;
- dois protocolos de coordenação: o completion, utilizado para encerrar uma transação, e o Two Phase Commit. Este último possibilita que múltiplos participantes cheguem a um consenso sobre o desfecho de uma transação. Na realidade, WS-AtomicTransaction define duas variações do Two Phase Commit, denominadas Two Phase Commit durável e Two Phase Commit



volatile;

- um conjunto de port types, que devem ser implementados pelos participantes e coordenadores dos protocolos completion e Two Phase Commit.

O WebSphere Application Server implementa a especificação Web Services Atomic Transaction. Esta especificação ativa os Web Services para participarem de transações globais através de um ambiente Web heterogêneo e distribuído.

### WS-BusinessActivity

Para determinadas situações, o modelo de transação atômica não é adequado. Por exemplo, transações de longa duração, cuja execução pode durar diversas horas. Para esse tipo de transação, é quase sempre inviável manter locks sobre os recursos transacionais.

Outro cenário no qual o modelo transação atômica não é adequado é o de transações "negócio-a-negócio". Nesse tipo de interação, os recursos transacionais estão espalhados por várias empresas e, portanto, a obtenção de locks sobre recursos de terceiros pode ser (geralmente é) inviável.

Esta especificação fornece a definição do tipo de coordenação de atividades de negócio que é para ser utilizada com o framework de coordenação extensível, descrito na especificação WS-Coordination. A especificação define dois protocolos de coordenação específicos para o tipo de coordenação de atividades de negócio:

- BusinessAgreementWithParticipantCompletion;
- BusinessAgreementWithCoordinatorCompletion.

Desenvolvedores podem utilizar um ou ambos os protocolos ao construir aplicações que requerem acordos consistentes sobre o resultado de atividades de longa duração.

### Ws-Security

Existem diversas formas de implementar segurança em serviços web, podendo ser adotado duas abordagens principais: segurança na camada de rede ou transporte e segurança na aplicação ou mensagem.

#### Segurança na camada de rede ou de transporte

Nesta abordagem, todo o canal de comunicação é protegido usando SSL/TSL ([http://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security)), o que garante que toda mensagem trafegue de maneira segura ponto a ponto.

É fortemente recomendável o uso da segurança na camada de aplicação, ou também no chamado nível da mensagem. A segurança no nível da mensagem garante a segurança da mensagem da sua origem até o seu destino independentemente dos nós intermediários, chamada de segurança "fim-a-fim". Veja a figura 3.

Neste artigo, vamos focar apenas na segurança no nível de mensagem, que é a abordagem implementada pelo WS-Security.

#### Segurança no nível de mensagem

O contexto de serviços Web, chamado de segurança no nível de mensa-



Figura 3. Exemplos de aplicações de segurança

gem XML, envolve a encriptação e decriptação de documentos XML. O W3C, o qual mantém o padrão XML, tem criado grupos de trabalhos para definir padrões para segurança em XML, incluindo assinaturas digitais, encriptação e gerenciamento de chaves para XML.

As implementações de segurança no nível de mensagem podem ser usadas para garantir a confidencialidade e a integridade das mensagens conforme passam por um número arbitrário de sistemas intermediários. As mensagens podem ser assinadas para fornecer integridade. Para a confidencialidade, pode-se escolher entre criptografar toda a mensagem ou parte dela.

Há várias vantagens em proteger a mensagem em vez de usar o protocolo de transporte. Em primeiro lugar, esse método é mais flexível, pois partes da mensagem, ao invés da mensagem inteira, podem ser assinadas ou criptografadas. Isso significa que os intermediários conseguirão ver as partes da mensagem destinadas a eles. Um exemplo é o ESB (Enterprise Service Bus), que roteia mensagens SOAP e é capaz de inspecionar partes não-criptografadas de uma mensagem para determinar o local para onde a mensagem deverá ser enviada, enquanto outras partes da mensagem permanecem criptografadas. Depois, os intermediários poderão adicionar seus próprios cabeçalhos à mensagem e assiná-la para fins de registro de auditoria. Finalmente, a mensagem protegida poderá ser enviada através de vários protocolos diferentes, como HTML, SMTP, FTP e TCP, sem que seja preciso contar com o protocolo para garantir a segurança, uma vez que as informações confidenciais continuam seguras na mensagem, independentemente do canal.

O WS-Security é um protocolo de comunicações que fornece meios para aplicarmos segurança aos Web Services. O protocolo contém especificações de como reforçar integridade e confidencialidade em mensagens Web Services. O protocolo WSS inclui detalhes no uso de SAML (Security Assertion Markup Language) e Kerberos (é um protocolo desenvolvido para fornecer poderosa autenticação em aplicações usuário/servidor), e formatos de certificação como X509. WS-Security descreve como integrar assinaturas e cabeçalhos criptografados em mensagens SOAP. Além disso, descreve como incluir tokens de segurança, inclusive tokens de segurança binários, como X509 e licenças Kerberos, nas mensagens.

A integridade da mensagem é obtida com assinaturas digitais XML (XML Signature). Isso garante que partes da mensagem não tenham sido adulteradas após a assinatura do originador. A confidencialidade da mensagem é baseada na especificação de criptografia XML (XML Encryption) e garante que partes correspondentes da mensagem só possam ser compreendidas pelo(s) destinatário(s) desejado(s). A segurança é oferecida no próprio protocolo de troca de mensagens – SOAP. Com este recurso,



é possível uma mensagem passar por vários Web Services ao longo do seu processamento.

Nosso objetivo não é detalhar tecnicamente a especificação WS-Security nem qualquer outra. Especificamente sobre o WS-Security já foi publicado um artigo na Edição 28 da MundoJava.

Por fim, a utilização de um padrão interoperável de segurança visa padronizar a aplicação da segurança na troca de mensagens entre o fornecedor e o consumidor de serviços. É importante ressaltar que nem todos os serviços devem fornecer suas mensagens sobre o formato de segurança, ou seja, apenas as mensagens que realmente necessitam é que devem aplicar este mecanismo, pois a inserção de um mecanismo de segurança aumenta o consumo de processamento dos servidores e pode afetar o tempo de resposta de um serviço.

### O barramento de serviços

O ESB (Enterprise Service Bus) é um dos principais conceitos relacionados a SOA. Isso mesmo, o ESB é um conceito. A razão para incluímos o ESB nesta discussão sobre interoperabilidade é para mostrar que um barramento pode ajudar a prover interoperabilidade na infraestrutura. Se as aplicações legadas estão implementadas em tecnologias que não oferecem interoperabilidade, o barramento de serviços pode ajudar, através de adapters.

Obviamente este conceito é implementado através de ferramentas e existem opções Open Source e proprietárias (e nada impede que você construa seu próprio ESB). Como referências para ESBs Open Source, podemos citar algumas (extraída do blog do Fernando Ribeiro — <http://fernandoribeiro.eti.br/>):

- Apache ServiceMix
- Apache Synapse
- Celtix
- ChainBuilder
- Eclipse Swordfish – baseado no SÓPERA
- Glassfish – baseado no OpenESB
- JBoss ESB
- Mule
- OpenESB
- PÉtALS
- Progress FUSE – baseado no Apache ServiceMix
- SÓPERA
- WSO2 – baseado no Apache Synapse

### SOA na Prática • Interoperabilidade em SOA – Desafios e padrões

O foco para garantir a interoperabilidade nos barramentos são os adapters. Como citamos anteriormente, um serviço pode ser implementado usando EJB, por exemplo, e pode não ser interessante mudar isso na aplicação. Neste contexto, é possível que o ESB exponha este EJB como Web Service e implemente internamente a conversão das interfaces. Além disso, o ESB pode ser utilizado para validar as políticas de acesso aos serviços, usando o padrão de WS-Policy, apresentado acima.

Saber mais

Aquele Blog de SOA: [www.aqueleblogdesoa.com.br](http://www.aqueleblogdesoa.com.br)

### Considerações finais

Dentre todas as discussões, existe uma que gostaríamos que ficasse na memória: Embora algumas tecnologias facilitem nossa vida, a garantia de sucesso (neste caso, interoperabilidade) depende muito mais de boas práticas e padrões do que de uma linguagem de programação ou tecnologia específica.

Web services não garantem, mas ajudam muito. ESB também não garante, mas também pode ajudar. No entanto, o que é fundamental é a adoção dos padrões abertos na publicação dos serviços (seja diretamente na aplicação, seja no barramento).

Neste artigo, apresentamos os padrões mais maduros, porém ainda é possível encontrar novos, no WS-I. Veja de quais você precisa e mãos à obra! Para os próximos artigos, apresentaremos separadamente exemplos práticos de implementação para cada padrão apresentado. **NU**

#### Referências

- Web Services Interoperability Organization (WS-I) — <http://www.ws.org>
- Baixo Acoplamento (João Ventura) — <http://www.aqueleblogdesoa.com.br/2008/07/baixo-acoplamento>
- Interoperabilidade com Web Services (Meier Back) — <http://www.aqueleblogdesoa.com.br/2008/06/interoperabilidade-com-web-services>
- Open Source ESBs — <http://fernandoribeiro.eti.br/2008/04/11/open-source-esbs/>
- SOABooks — <http://www.soabooks.com>

