**Capítulo 9. O Futuro dos Web Services**

Ao longo deste livro, mantivemos um foco relativamente estreito sobre o que são web services e como criá-los. Para concluir, vamos dedicar algum tempo discutindo o futuro dos web services, tanto do ponto de vista das tecnologias quanto da arquitetura como um todo. Especificamente, discutiremos os futuros do SOAP, WSDL e UDDI, bem como a próxima geração de serviços ainda mais úteis e poderosos.

**9.1 O Futuro do Desenvolvimento Web**

Antes de entrarmos no destino das tecnologias no futuro, vamos dedicar um momento para destacar exatamente como os web services provavelmente irão impactar o desenvolvimento web.

Falamos bastante sobre interoperabilidade. Com web services, nossa preocupação é como encontrar informações e movê-las pela Web. Questões que costumavam ser importantes — como a linguagem de programação na qual o código foi escrito, o sistema operacional no qual o código está sendo executado, o modelo de objeto no qual o código se baseia ou o fornecedor do sistema de banco de dados subjacente — não importam tanto assim agora.

Essa é uma afirmação forte. Alguns podem até dizer que é exagerada. Eis o raciocínio.

Antes dos web services, a grande maioria das plataformas de desenvolvimento em escala corporativa era bastante “endogâmica”. Aplicações Java funcionavam melhor com aplicações Java; aplicações COM funcionavam melhor com aplicações COM; aplicações CORBA funcionavam melhor com aplicações CORBA, e assim por diante. Para tirar o máximo proveito de cada ambiente, era preciso padronizar e focar na própria plataforma tecnológica. Era possível fazer Java e COM funcionarem juntos, mas era doloroso.

Os web services, entretanto, abriram um canal de integração entre Java e COM, e COM e CORBA, etc., que não existia antes. Como esse canal é construído com padrões abertos que qualquer plataforma pode implementar, pela primeira vez temos uma situação na qual se pode facilmente invocar funcionalidades escritas em uma linguagem de programação em uma plataforma a partir de qualquer outra linguagem e plataforma. Isso nos permite olhar além das linguagens e focar nas próprias aplicações.

Esse ponto foi demonstrado no exemplo Hello World do Capítulo 3. Criamos o mesmo tipo de web service com três linguagens de programação diferentes. Excetuando pequenos bugs de interoperabilidade que existem nas ferramentas de web services, poderíamos facilmente alternar entre as três implementações sem prestar atenção em como foram realmente escritas. Se eu fosse procurar por um serviço Hello World em algum lugar da Internet, não teria importância para mim se ele foi escrito em Java, Perl, .NET, ou mesmo COBOL ou ADA. (Existe, aliás, uma implementação de SOAP para ADA. E o Visual Studio .NET da Microsoft suporta escrever assemblies com COBOL.)

**9.1.1 Web Services e Tecnologias Existentes**

Um insight crítico é que web services não substituem as infraestruturas tecnológicas existentes. Em vez disso, eles ajudam a integrar tecnologias existentes. Em outras palavras, se você precisa que uma aplicação J2EE converse com uma aplicação COM, web services tornam isso mais fácil. Web services não vão substituir completamente aquele sistema mainframe de 30 anos que fica no fundo do depósito e que ninguém mais lembra. Mas podem fornecer acesso automatizado e multiplataforma às aplicações do mainframe, abrindo assim novos canais de negócios.

**9.2 O Futuro do SOAP**

O protocolo SOAP já tem alguns anos. Uma das versões originais tornou-se o que hoje é chamado de XML-RPC, uma alternativa simples e popular ao SOAP defendida pela Userland Software. (O CEO da Userland, Dave Winer, é um dos coautores da especificação original do SOAP.) Para saber mais sobre XML-RPC, leia *Programming Web Services with XML-RPC* de Simon St. Laurent, Joe Johnston e Edd Dumbill (O’Reilly).

O XML-RPC se separou do SOAP em 1998. A primeira versão do protocolo SOAP foi anunciada em 1999 e, desde então, houve quatro revisões, com uma quinta sendo trabalhada pelo W3C. As duas versões discutidas neste livro (Versão 1.1 e Versão 1.2) são as atualmente usadas em ambientes de produção, mesmo sem serem padrões oficiais do W3C.

Em um futuro não muito distante, o rascunho da especificação do SOAP 1.2 evoluirá para a recomendação *W3C XML Protocol Version 1.0*, que será a primeira versão padronizada do protocolo.

Não devem haver muitas mudanças entre SOAP 1.2 e XML Protocol Version 1.0, pois o grupo de trabalho do W3C se comprometeu a usar o SOAP como base para seu trabalho e a garantir que a compatibilidade retroativa seja mantida, pelo menos em um nível fundamental. Infelizmente, ainda é muito cedo no processo para saber sobre quaisquer diferenças entre o XML Protocol e o SOAP 1.2. Se tiver curiosidade, acompanhe a discussão de desenvolvimento do XML Protocol através da lista *xml-dist-app* (consulte a página principal do W3C XML Protocol em <http://www.w3.org/2000/xp> para detalhes de assinatura).

**9.3 O Futuro do WSDL**

Assim como o SOAP, a *Web Service Description Language* ainda não é um padrão oficial da Internet, mas está a caminho de se tornar um. Ela foi submetida ao W3C, e um grupo de trabalho está sendo formado para gerenciá-la. Ao contrário do SOAP, que tem uma direção relativamente estável dentro do W3C, o WSDL padronizado pode ser diferente da versão amplamente usada atualmente. É cedo demais para saber quão diferente será ou quando o padrão aprovado pelo W3C será lançado.

**9.3.1 Peças que Faltam**

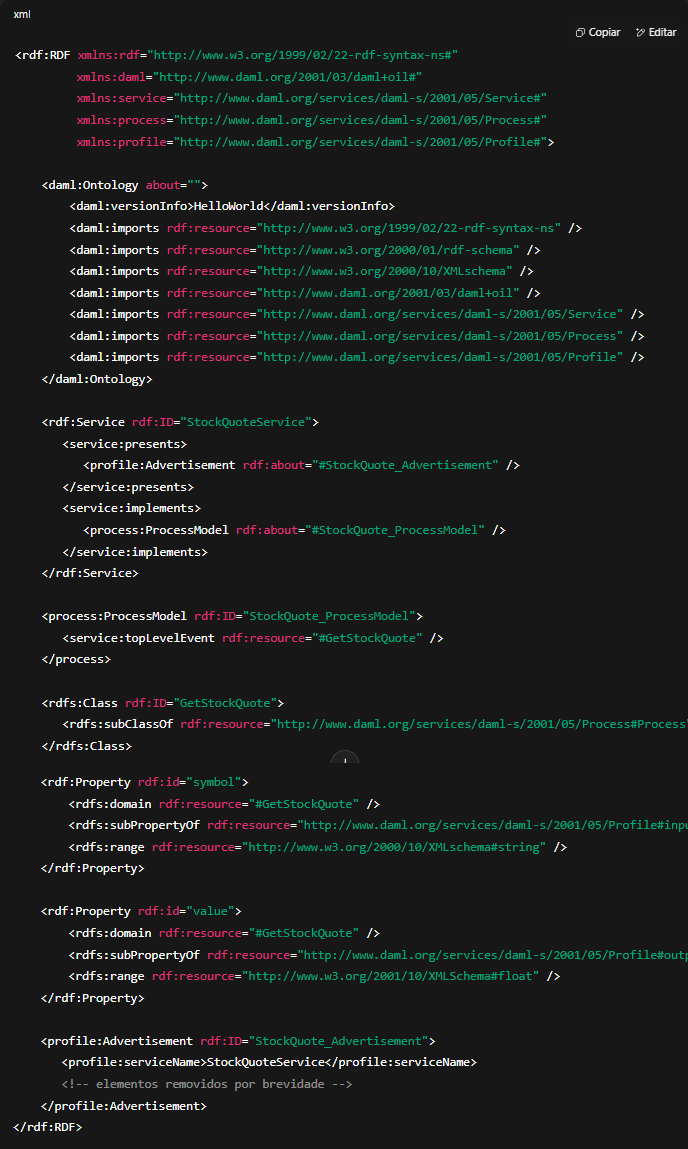
Há várias coisas importantes faltando no WSDL que precisarão ser abordadas pelo grupo de trabalho do W3C. Por exemplo, não há um mecanismo padronizado para estender a descrição WSDL para incluir informações sobre requisitos de segurança, atributos de qualidade de serviço, sequência de operações, etc. Embora isso não seja estritamente necessário quando o WSDL é usado para descrever serviços simples e básicos no estilo RPC, tais extensões padronizadas tornam-se críticas ao aplicar a tecnologia de web services a cenários corporativos de e-business.

**9.3.2 Uma Alternativa ao WSDL**

Outra questão importante que o grupo de trabalho do WSDL precisará abordar é a reconciliação do WSDL com outros mecanismos alternativos de descrição de serviços, como a *DARPA Agent Markup Language* (DAML) baseada no DAML-S (o S significa “services”). O DAML-S está focado na tarefa de construir um modelo formal de dados semânticos para web services. Em outras palavras, eles estão formalizando a linguagem que usamos para descrever web services. Embora o DAML-S não tenha um apoio corporativo sólido, grande parte do trabalho que está sendo feito terá impacto na direção futura da padronização do WSDL.

Os conceitos envolvidos no DAML-S não são muito diferentes do WSDL, mas a sintaxe é muito mais complexa. Por exemplo, o **Exemplo 9-1** mostra uma descrição parcial do serviço Hello World do Capítulo 3, em DAML-S em vez de WSDL.

**Exemplo 9-1. Descrição DAML-S de exemplo do serviço WSDL**



O DAML-S é baseado no padrão *Resource Description Framework* (RDF). Isso tende a torná-lo mais complexo, verboso e difícil de usar que o WSDL.

Dito isso, há várias lições que o WSDL pode aprender com o DAML-S:

1. O DAML-S naturalmente suporta a capacidade de estender descrições de serviços para incluir uma ampla variedade de informações semânticas e funcionais, como segurança, qualidade de serviço, etc.
2. O DAML-S naturalmente suporta herança em toda a descrição.
3. O DAML-S fornece um mecanismo rico para descrever processos de web service (sequências lógicas de operações). O WSDL não suporta operações de sequenciamento.
4. O DAML-S permite que um serviço implemente múltiplos processos (o equivalente do DAML-S a um *port type* do WSDL).
5. O DAML-S suporta uma descrição de anúncio de serviço rica, que fornece informações sobre quem está oferecendo o serviço, quais são as capacidades do provedor, etc. O WSDL não inclui qualquer informação de anúncio.

Esses recursos do DAML-S provavelmente farão parte da próxima geração do WSDL.

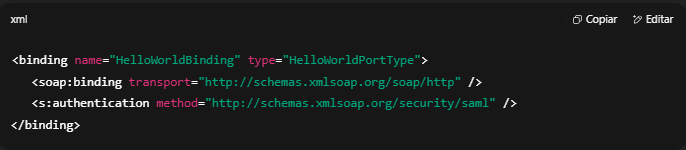
**9.3.3 Extensões Padrão**

Um componente-chave para o sucesso dos web services será a capacidade de descrever não apenas o serviço em si, mas também todas as capacidades, requisitos, premissas e processos do serviço de uma forma padronizada e consistente.

Por exemplo, considere como descrever um web service que utiliza um *single sign-on* baseado em SAML, como o que discutimos no Capítulo 5. Não há como declarar no WSDL o tipo de autenticação que um serviço suporta. Aliás, não há como declarar que um serviço suporta qualquer tipo de autenticação.

Como o WSDL do futuro poderia permitir expressar qual tipo de mecanismo de autenticação é usado? Uma forma seria definir uma extensão padrão para o elemento *binding* do WSDL, como no **Exemplo 9-2**.

**Exemplo 9-2. Extensão hipotética para *bindings* WSDL**



Ferramentas de web services habilitadas para WSDL que entendam a extensão de autenticação saberiam então que o SAML poderia ser usado para autenticação.

Atualmente, no entanto, não existem extensões padrão para WSDL, nem um esforço amplo da indústria para defini-las. Isso também pode se tornar parte do WSDL padronizado pelo W3C.

**9.4 O Futuro do UDDI**

O Capítulo 4 apresentou o UDDI como um web service para descobrir outros web services. Por meio da definição de um formato de registro padronizado e de uma interface *port type*, o UDDI permite que provedores e consumidores de serviços se descubram e integrem dinamicamente.

O UDDI foi originalmente desenvolvido pela Microsoft, IBM e Ariba, e agora é gerenciado por um amplo consórcio de empresas da indústria. O plano é submeter o UDDI para padronização assim que a Versão 3.0 da especificação estiver concluída (a versão atual é a 2.0).

Um dos requisitos-chave para versões futuras do UDDI é uma infraestrutura de segurança que permita aos consumidores de serviços validar a identidade dos provedores de serviços que publicam seus serviços — possibilitando o estabelecimento de uma relação de confiança muito mais robusta.

**9.4.1 Problemas com o UDDI**

Existem alguns problemas no UDDI que precisarão ser resolvidos em versões futuras da especificação. A segurança fraca é uma das questões mais significativas. Atualmente, é possível que qualquer pessoa crie uma entrada em um registro UDDI fingindo ser outra pessoa. Por exemplo, eu posso facilmente criar uma entrada em um registro UDDI fingindo ser a Microsoft. Nem é preciso dizer que isso não é bom.

Outro grande problema é a proliferação de “links quebrados” nos registros públicos do UDDI. Esses links apontam para empresas ou serviços que não existem ou que não estão mais disponíveis.

Há também uma falta de compreensão por parte das empresas sobre para que serve o UDDI e como ele pode ser útil. Isso pode prejudicar sua adoção no futuro. Mesmo entre as empresas que entendem, surgiram dúvidas sobre se os registros públicos UDDI serão úteis a longo prazo.

**9.5 Campos de Batalha dos Web Services**

Nas últimas décadas, vimos empresas travarem guerras para estabelecer seus sistemas operacionais, modelos de componentes, linguagens de programação, navegadores e assim por diante. Um aspecto refrescante do mundo dos web services é que a maioria dessas batalhas se torna irrelevante.

Considere os serviços e clientes SOAP que discutimos neste livro. Quando implantamos um serviço SOAP, definimos os métodos que queremos expor pela rede. No passado, teríamos definido esses métodos com CORBA IDL ou algo semelhante, gerado *bindings* para várias linguagens e plataformas, e depois torcido para que conseguíssemos conquistar uma parte suficiente do mercado para que utilizassem nosso serviço. Se sua plataforma ou suas ferramentas de desenvolvimento não fossem compatíveis com a sua infraestrutura (talvez não suportassem o nível correto de CORBA, por exemplo), você provavelmente ficaria sem sorte.

Com o SOAP, podemos descrever tudo em termos de tipos de dados independentes de plataforma, definidos pelo XML Schema. Se sua plataforma de desenvolvimento possui ferramentas de análise XML (e hoje em dia é difícil encontrar uma plataforma que não possua, desde celulares até mainframes), você pode começar a desenvolver aplicações que usem o serviço.

Não pense, nem por um momento, que os ferozes concorrentes do mercado atual de repente passarão a se dar maravilhosamente bem. À medida que as empresas descobrirem que as velhas batalhas já não importam, todas tentarão ganhar vantagem sobre seus concorrentes de alguma outra forma. Vamos examinar alguns dos campos de batalha do futuro.

**9.5.1 Ferramentas de Desenvolvimento**

Uma das razões para a dominância das plataformas Windows é o sucesso da Microsoft em conquistar desenvolvedores. Independentemente dos benefícios da sua tecnologia, se você conseguir convencer centenas de milhares de pessoas inteligentes a começar a criar produtos com ela, você ganha uma vantagem esmagadora no mercado. Você não precisa criar o *killer app* sozinho; os desenvolvedores terceirizados podem fazer isso por você.

Com isso em mente, veremos os principais fornecedores de software trabalhando de forma muito agressiva para diferenciar suas ferramentas de desenvolvimento para web services. Se eu conseguir convencê-lo de que minhas ferramentas o tornarão infinitamente mais produtivo e bem-sucedido, a tarefa de prendê-lo ao meu ecossistema de desenvolvimento se torna muito mais fácil. E, uma vez que você esteja confortável com minhas ferramentas, posso integrar minhas iniciativas proprietárias a elas, removendo lentamente sua capacidade de usar outras ferramentas.

Os fornecedores precisam parecer compatíveis com os padrões, mas também aparentar ser de alguma forma superiores. Muitos dos recursos diferenciadores serão complementos não padronizados — uma forma de “abraçar e estender” (*embracing and extending*) — que tem o potencial de enfraquecer a interoperabilidade dos web services, ao mesmo tempo em que prende os desenvolvedores aos produtos de um único fornecedor.

**9.5.2 Serviços Matadores (*Killer Services*)**

Se milhões de desenvolvedores puderem acessar web services com ferramentas gratuitas, um modelo de negócios óbvio é fornecer web services tão atraentes que os desenvolvedores estarão dispostos a se amarrar a eles. Isso é semelhante à Web, na qual milhões de clientes podem acessar sites com um navegador gratuito.

Um concorrente inicial nessa corrida por serviços matadores é a carteira online (*online wallet*). Uma carteira online de próxima geração é um web service que permite aos clientes armazenar senhas, números de cartão de crédito e outras informações sensíveis. O provedor da carteira online se torna um centro de compensação (*clearinghouse*) para o comércio eletrônico. Se quisermos montar uma loja virtual, podemos usar o serviço para processar transações com cartão de crédito. Um cliente nos fornece algumas informações (nome de usuário e senha, por exemplo), e nós acessamos o centro de compensação online para obter um código de aprovação para a transação.

Talvez tenhamos que pagar ao centro de compensação uma taxa (uma porcentagem do total, talvez) por cada transação, mas, se esse serviço for fácil de usar e acessar, oferecer um alto nível de serviço, for seguro e amplamente aceito pelos consumidores, poderemos economizar muito tempo e dor de cabeça na operação e gestão da nossa loja online. Se as ferramentas de desenvolvimento facilitarem muito o uso de uma carteira online específica, o fornecedor por trás dessas ferramentas e da carteira online estará em uma posição muito favorável. Isso é amplamente visto como parte das estratégias do .NET e do .NET My Services (anteriormente conhecido como Hailstorm) da Microsoft.

À medida que os web services se consolidarem no mercado, veremos muitos provedores tentando criar outros serviços matadores para atrair o mundo para suas portas virtuais.

**9.5.3 Mercados Lucrativos (*Lucrative Marketplaces*)**

A indústria de EDI (*Electronic Data Interchange*) trabalhou por décadas para automatizar a troca de pedidos de compra, faturas e documentos semelhantes. Infelizmente, esses sistemas tradicionalmente foram muito caros para criar e manter. Com os custos iniciais mais baixos dos web services (você pode criar, implantar e acessar web services com as tecnologias que já possui), muitas empresas menores agora podem participar dessas comunidades de negócios online, assim como o advento da Web introduziu muitas novas empresas que desafiaram comerciantes estabelecidos.

À medida que a revolução dos web services avança, veremos a indústria tentar novamente estabelecer mercados de negócios entre empresas (*B2B marketplaces*). No passado, eles fracassaram por dois motivos:

* Os compradores queriam mais controle sobre suas decisões de compra; eles não queriam que uma máquina decidisse com base em qual empresa aparecesse primeiro em uma lista de resultados ordenada alfabeticamente.
* Os fornecedores queriam mais controle sobre os preços. Um mercado onde os preços de um vendedor são comparados online pode ser bom para um agente tentando encontrar o preço mais baixo, mas não é bom para os fornecedores, especialmente quando um agente pode não levar em conta fatores como a capacidade de um fornecedor atender grandes pedidos, ou como outros compradores avaliaram determinado fornecedor.

À medida que os web services amadurecerem, essas preocupações serão abordadas. Por meio de chamadas de método SOAP para um registro UDDI, um comprador online poderá encontrar todos os fornecedores que afirmam atender às suas necessidades. Novos web services construídos sobre o UDDI permitirão que agentes obtenham mais informações sobre os fornecedores, incluindo suas classificações de crédito, a rapidez com que entregaram pedidos no passado, etc. Outros serviços poderão assegurar aos fornecedores que os compradores terão condições de comparar diferentes fornecedores de forma justa. Por exemplo, minha empresa pode ter preços um pouco mais altos, mas não afirmamos ter produtos em estoque quando nossos depósitos estão vazios.

Os web services prometem criar um ambiente no qual agentes possam avaliar diversos fatores como um ser humano faria, permitindo que os usuários humanos foquem em aspectos mais importantes para seus negócios.

**9.5.4 A Empresa**

Talvez uma das batalhas mais significativas ainda por surgir será a pela dominância no mercado de web services corporativos (*enterprise web services*). Estes são os serviços de infraestrutura que fornecerão a base para concretizar a promessa dos agentes e de formas mais dinâmicas de e-business. Esses serviços incluem elementos como: gerenciamento e negociação de confiança distribuída; medição, contabilização e cobrança; gerenciamento de conteúdo e informações; aplicação e auditoria de privacidade; fornecimento e aquisição inteligente e dinâmica de materiais; e inúmeros outros serviços que formam o alicerce do desenvolvimento de negócios corporativos.

Ainda não está claro qual será o efeito de basear tais componentes essenciais da infraestrutura em tecnologia de web services sobre o mercado e, neste momento, é cedo demais para oferecer qualquer percepção real. Seja qual for o impacto, espere ver muito mais atividade nessa área em um futuro próximo, à medida que empresas de tecnologia da Internet (tanto antigas quanto novas) disputam posições em um novo mercado em crescimento.

Os web services são uma abordagem jovem para escrever aplicações distribuídas. Como tal, estão longe de ser tão maduros e ricos em recursos quanto mecanismos como J2EE, CORBA e .NET. É particularmente necessária a funcionalidade que permita aos web services operar no ambiente corporativo: segurança, transações, integração com banco de dados, etc. Isso é semelhante aos primeiros dias do Java — foi preciso chegar ao Java 2 Enterprise Edition para que os programadores tivessem um conjunto de extensões padrão para segurança, transações, mensagens, suporte a servidores, bancos de dados, etc.

Com os web services, vemos uma evolução paralela. Atualmente, temos as tecnologias (por exemplo, SOAP, WSDL e UDDI) que permitem que os web services funcionem. Sozinhas, essas tecnologias têm grande potencial, mas ainda não são suficientes para o ambiente corporativo.

**9.6 Tecnologias**

Embora muitos padrões de web services já estejam definidos, há também muitas tecnologias que ainda não chegaram lá. Vamos discutir essas peças que faltam e especular sobre como e quando elas serão completadas.

**9.6.1 Agentes**

Um agente é um programa que pode agir em seu nome. Por exemplo, eu gostaria de ter um agente que fizesse reservas de voo, carro alugado e hotel para uma viagem de negócios futura. Meu agente ideal saberia quais companhias aéreas e hotéis eu prefiro, possivelmente com base em viagens anteriores para a mesma região. Se assumirmos que todos os dados relevantes que nosso agente possa usar estão em um documento XML ricamente estruturado, um agente poderia ser programado para aproveitar diversos tipos de informações ao planejar uma viagem.

Por exemplo, ao voar de costa a costa, Chicago tem mais probabilidade de atrasos devido ao clima no inverno, enquanto Dallas tem mais probabilidade de atrasos no verão. Um agente poderia descobrir que há uma promoção de milhagem que me daria 10.000 milhas extras se eu passasse por Toronto. Talvez um agente pudesse verificar automaticamente minha agenda para ver a que horas estou livre para partir no dia do voo.

Agentes têm sido um sonho da IA por anos. O XML e os web services têm o potencial de torná-los reais. Eis o que é necessário:

* Todos os dados envolvidos devem ser codificados em XML, usando vocabulários bem compreendidos. Isso significa que precisamos de conjuntos de tags padrão para agendas, voos, aeroportos, previsões do tempo, etc. Alguns desses vocabulários existem, mas a maioria precisará ser criada.
* Todas as diversas companhias aéreas, hotéis, empresas de aluguel de carros e outros fornecedores devem fornecer web services que facilitem para meu agente criar, alterar e cancelar reservas.
* Mais importante ainda, a tecnologia de agentes deve ser poderosa, confiável, segura e fácil de usar. Isso não é exatamente a tarefa mais simples no mundo do desenvolvimento de software. As pessoas não usarão agentes se eles não forem confiáveis, não puderem fazer muito ou forem complicados demais para alguém sem doutorado em Ciência da Computação.

**9.6.2 Qualidade de Serviço**

Os web services tornam possível criar aplicações a partir de múltiplos componentes distribuídos pela Web. Essa é uma ideia muito poderosa, mas, para algumas aplicações, os desenvolvedores precisam ter a garantia de que esses componentes estarão disponíveis constantemente e com velocidades aceitáveis. Isso significa que contratos de Qualidade de Serviço (*Quality of Service* – QoS) se tornarão ainda mais importantes, simplesmente porque a Web passará a ser uma parte vital de um número cada vez maior de aplicações.

**9.6.3 Privacidade**

Se os dispositivos e agentes da minha vida receberam dados pessoais sensíveis, é crucial que eles compreendam meus desejos em relação à privacidade. Também é crucial que esses dispositivos e agentes entendam como as várias entidades na rede lidarão com esses dados.

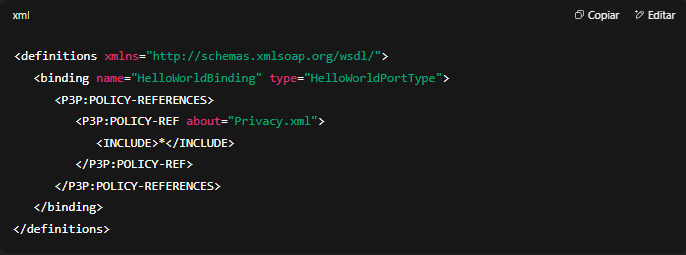
O trabalho da *Platform for Privacy Preferences* (P3P), feito pelo W3C, se tornará cada vez mais importante. Documentos P3P são legíveis por máquina, o que significa que agentes e outros pedaços de código podem examinar a política de privacidade de um site e determinar se ela é aceitável.

À medida que a importância da privacidade cresce (assim como a consciência pública sobre o quão pouco a Web realmente possui), outras tecnologias de privacidade podem ser necessárias. Por exemplo, um agente poderia obter de um provedor um documento P3P assinado digitalmente e criptografado, obtendo um contrato juridicamente vinculante de que os dados fornecidos ao provedor pelo agente serão protegidos e tratados de uma maneira específica.

O primeiro passo é relativamente simples: criar uma política P3P e associá-la ao seu web service por meio de links fornecidos na descrição WSDL desse serviço. No entanto, isso é apenas parte da solução. O que é necessário é uma infraestrutura mais abrangente e padronizada para proteger as informações à medida que viajam pela Web. Até que tal estrutura esteja implementada, o impacto e a utilidade dos web services voltados para lidar com informações pessoais serão limitados, no mínimo. Atualmente, não há propostas em discussão para isso.

O **Exemplo 9-3** mostra como uma referência a uma política P3P poderia aparecer dentro de um documento WSDL. Aqui, estamos afirmando que a política P3P Privacy.xml se aplica a todas as operações definidas pelo HelloWorldBinding.

**Exemplo 9-3. P3P dentro de WSDL**



**9.6.4 Segurança**

Acima de tudo, a segurança é primordial. Não importa o que um determinado web service possa fazer — se ele tiver probabilidade de expor meu número de cartão de crédito, não quero usá-lo. Embora a especificação base do SOAP em si não tenha sido projetada com a segurança em mente, isso não significa que a segurança seja impossível.

Um dos exemplos que discutimos neste livro usa o *IBM's XML Security Suite* para criptografar o conteúdo de envelopes SOAP enquanto eles trafegam pela rede. À medida que os web services se popularizarem, veremos mais tecnologias como essa, com o resultado final sendo que envelopes SOAP seguros se tornarão tão comuns quanto documentos HTML transmitidos pela *Secure Socket Layer*.

A questão da segurança exige uma resposta complexa — uma que sempre retorna ao ponto de não focar na tecnologia em si, mas em como essa tecnologia é implementada, implantada e utilizada. As empresas de tecnologia só podem fazer até certo ponto no fornecimento de métodos para expressar confiança ou afirmar fatos. A segurança acontece apenas quando as empresas dedicam tempo para torná-la uma prioridade.

**9.6.5 Gerenciamento de Confiança**

A confiança é o requisito primordial para conduzir negócios pela Internet e será um componente-chave para o sucesso da arquitetura de web services. Já estão surgindo tecnologias que ajudam as empresas a expressar e estabelecer relações de confiança dentro do contexto dos web services. Um exemplo de tal tecnologia é o *XML Key Management Service*, um mecanismo padrão para gerenciar chaves públicas e privadas.

**9.6.6 Contratos Online**

Falamos sobre contratos e outros documentos juridicamente vinculantes ao longo desta seção, enfatizando que, se os web services se tornarem comuns, o impacto de um serviço específico ficar indisponível ou fornecer dados incorretos poderá ser catastrófico. Como esses contratos serão negociados ou aplicados?

Claramente, ter os advogados do provedor do serviço reunindo-se com os advogados do solicitante do serviço não funcionará em um mundo de aplicações construídas a partir de conglomerados de serviços.

Várias tentativas já foram feitas para criar linguagens baseadas em XML capazes de descrever acordos e contratos. O *Collaboration Profile Protocol* e o *Collaboration Profile Agreement* (CPP-CPA) do ebXML são um exemplo dessa tecnologia. Infelizmente, nenhuma dessas tentativas foi amplamente adotada, e o vencedor final ainda está por surgir.

**9.6.7 Mensagens Confiáveis (*Reliable Messaging*)**

Mensagens confiáveis envolvem garantir que tanto o remetente quanto o destinatário de uma mensagem saibam se ela foi realmente enviada e recebida, além de garantir que a mensagem tenha sido enviada uma única vez ao destinatário pretendido. Este é um problema que assola o desenvolvimento de aplicações na Internet desde o seu início.

A Internet, por sua própria natureza, é não confiável. Servidores que estavam ativos e funcionando em um momento podem estar fora do ar no próximo. Os protocolos usados para conectar remetentes e destinatários não foram projetados para suportar construções de mensagens confiáveis, como identificadores de mensagens e confirmações (*acknowledgments*). Os destinatários das mensagens devem poder confirmar que de fato receberam uma mensagem. Os remetentes devem poder armazenar em cache essas mensagens caso não recebam uma confirmação e precisem reenviá-las. A tecnologia fundamental que move a Internet hoje não oferece suporte a tais mecanismos. Portanto, somos obrigados a implementar novos protocolos e tecnologias para atender a essas necessidades.

A importância das mensagens confiáveis dentro da empresa não pode ser subestimada, especialmente quando discutimos a implementação de web services que podem atravessar *firewalls* para integrar clientes, fornecedores e parceiros.

No contexto corporativo, as mensagens confiáveis geralmente foram fornecidas por soluções proprietárias como o *IBM’s MQ Series* ou o *Microsoft Message Queue*, nenhuma das quais é capaz de se integrar facilmente à outra (existem maneiras de fazê-las funcionar juntas, mas são, no mínimo, dolorosas).

Do ponto de vista dos web services, existem duas formas de abordar a implementação de mensagens confiáveis:

1. Implementar mensagens confiáveis na camada de aplicação, significando que os princípios de mensagens confiáveis devem ser incorporados diretamente à implementação do web service.
2. Implementar mensagens confiáveis na camada de transporte, significando que os web services não precisam fazer nada para oferecer suporte ao uso de mensagens confiáveis.

A primeira abordagem é implementada por produtos como o *Microsoft’s BizTalk*, que usa tecnologias de web services como SOAP para trocar documentos de negócios (por exemplo, pedidos de compra e solicitações de cotação) de forma confiável.

A segunda abordagem é implementada por protocolos como o *IBM’s Reliable HTTP (HTTP-R)*. O HTTP-R é uma implementação do HTTP padrão com a adição de “gerenciadores de ponto final” (*endpoint managers*) que garantem a confiabilidade da conexão entre o solicitante HTTP e o servidor HTTP.

Uma discussão completa sobre HTTP-R e BizTalk está fora do escopo desta discussão. Para mais informações sobre eles, consulte as referências online no Apêndice A.

**9.6.8 Transações**

Um dos requisitos essenciais para aplicações implantadas dentro de uma empresa é o suporte a transações. Múltiplas operações que precisam ser executadas em lote devem ter sucesso todas juntas ou falhar todas juntas para que qualquer uma delas seja válida. Atualmente, não existe um método padrão (ou mesmo proposto) para implementar e gerenciar transações no ambiente de web services.

Há um debate de longa data sobre se os web services exigem um método para realizar transações no estilo de *two-phase commit* (confirmação em duas fases). Uma transação de duas fases é aquela na qual todas as operações em um lote devem ser invocadas, mas não finalizadas. Depois que todas as operações relatam invocação bem-sucedida, elas podem voltar e finalizar suas operações. O exemplo clássico de uma transação de duas fases é quando uma aplicação precisa gravar dados em duas tabelas diferentes de um banco de dados. Ambas as tabelas devem ser atualizadas ou nenhuma delas pode ser. Se a gravação em uma tabela for bem-sucedida, mas falhar na segunda, a primeira operação deve ser desfeita e um erro relatado ao usuário.

O principal problema com a confirmação em duas fases na Web é que, enquanto cada participante na transação (por exemplo, as duas tabelas de banco de dados no exemplo anterior) aguarda a confirmação final de que todas as operações foram concluídas com sucesso, eles devem manter um bloqueio sobre o recurso que está sendo modificado dentro da transação. Esse bloqueio impede que qualquer outra pessoa faça alterações no recurso que, de outra forma, poderiam fazer a transação falhar. Esses bloqueios são aceitáveis quando todos os recursos estão sendo gerenciados pelo mesmo computador, mas causam problemas de desempenho, escalabilidade e confiabilidade em um ambiente de computação distribuída.

Esse problema remete à discussão sobre mensagens confiáveis. Com web services, a maior parte do tráfego será sobre HTTP. Sem a promessa de confiabilidade absoluta, se a conexão entre dois participantes de uma transação for interrompida enquanto a transação está sendo executada, nenhum deles poderá finalizar suas operações, porque nenhum saberá se a operação do outro foi concluída com sucesso. Os bloqueios colocados nos recursos em questão poderiam ser mantidos indefinidamente, e o processamento pararia.

Um projeto de pesquisa promissor da IBM na área de transações é chamado de *Dependency Sphere*. Uma *Dependency Sphere*, ou D-Sphere, é uma nova forma de encarar transações sob a perspectiva de computação distribuída baseada em mensagens. Em um *two-phase commit*, uma transação é bem-sucedida se todas as operações executadas dentro do contexto dessa transação forem realizadas sem gerar erros. Na abordagem D-Sphere, a transação é bem-sucedida se todas as mensagens enviadas forem recebidas e confirmadas de forma confiável pelo destinatário pretendido dessas mensagens.

Aplicadas aos web services, as D-Spheres introduzem um novo tipo de web service para gerenciar o contexto de transação da D-Sphere. É função desse serviço de gerenciamento garantir que a transação tenha sucesso ou falhe. Se falhar, um aviso será enviado aos participantes para que possam executar as ações compensatórias apropriadas. A vantagem dessa abordagem é que ela assume mensagens confiáveis (assim, desconexões temporárias entre participantes deixam de ser um fator) e elimina a necessidade de bloqueios de recursos, evitando os tipos de *deadlock* que poderiam ocorrer com a abordagem de confirmação em duas fases.

Um exemplo de como as D-Spheres poderiam ser aplicadas em um ambiente corporativo de web services é quando um solicitante de serviço precisa executar múltiplas operações em múltiplos serviços — por exemplo, criar um novo usuário nos serviços CRM e ERP ao mesmo tempo. A D-Sphere poderia garantir que ambos os serviços recebessem e confirmassem com sucesso a solicitação para adicionar o novo usuário. O Apêndice A contém links para mais informações sobre D-Spheres.

**9.6.9 Licenciamento e Serviços de Contabilização**

Parte da visão dos web services é a ideia de que o software possa ser vendido como um serviço. Ou seja, as empresas pagarão para alugar o acesso a aplicações em vez de assumir o custo de comprar e manter as aplicações por conta própria. Esse conceito pode reduzir os custos de manutenção, mas exige web services padronizados para gerenciar licenças e monitorar o uso dos serviços.

No ambiente corporativo, esses serviços terão que se integrar com as soluções existentes de contabilidade e faturamento, autenticação e autorização, e serviços de eventos e notificações para serem realmente significativos e úteis.

**9.7 Implantação dos Web Services**

Como os web services provavelmente serão implantados no mercado? Achamos que o cenário mais provável é que os clientes construam web services internamente e, em seguida, avancem para aplicações construídas com web services mais amplamente distribuídos.

Já discutimos as tecnologias que precisam ser construídas sobre o SOAP e tecnologias relacionadas para que os web services assumam mais responsabilidades nos negócios. Dado que questões como segurança, autenticação e não repúdio são difíceis de abordar na Web de hoje, acreditamos que muitos dos primeiros adotantes começarão implementando web services internamente. Como administrador de rede, posso controlar o acesso aos servidores internos com muito mais facilidade do que posso controlar o acesso a um site público.

Como exemplo, digamos que eu crie uma aplicação baseada em SOAP para processar relatórios de despesas. Sempre que um usuário retorna de uma viagem de negócios, ele usa o aplicativo cliente SOAP para preencher seu relatório de despesas. O cliente SOAP envia uma consulta ao registro UDDI local, que aponta o cliente para um documento WSDL, o qual fornece as informações necessárias para que o cliente acesse a aplicação de despesas. O chefe do departamento de contabilidade pode mudar o local da aplicação de despesas a qualquer momento, e o cliente ainda será capaz de encontrá-la e acessá-la.

Como a aplicação é construída em SOAP, é possível (talvez até fácil) escrever aplicações clientes que funcionem em praticamente qualquer plataforma que eu suporte. Como todos os clientes estão dentro da minha rede, estou menos preocupado com segurança e privacidade do que estaria de outra forma. Como os metadados sobre a aplicação são descritos com WSDL e armazenados em um registro UDDI, posso mudar a localização, a plataforma hospedeira, a linguagem hospedeira, etc., da aplicação sem afetar os clientes. Isso dá aos administradores de sistemas uma quantidade tremenda de flexibilidade.

À medida que mais e mais aplicações internas forem construídas com web services, veremos os primeiros adotantes começarem a envolver seus fornecedores e parceiros de negócios. É ótimo que eu possa fazer uma requisição interempresarial para suprimentos; o próximo passo óbvio é fazer requisições de fornecedores externos. Esse próximo passo exige que meus fornecedores também usem SOAP (e WSDL e UDDI e ...).

Aplicações baseadas em web services se tornarão comuns, e uma arquitetura de componentes baseada em SOAP se tornará o paradigma dominante de desenvolvimento.

**Sumário do Capítulo 9:**

* 9. O Futuro dos Web Services
* 9.2 O Futuro do SOAP
* 9.3 O Futuro do WSDL
* 9.4 O Futuro do UDDI
* 9.5 Campos de Batalha dos Web Services
* 9.6 Tecnologias
* 9.7 Implantação dos Web Services