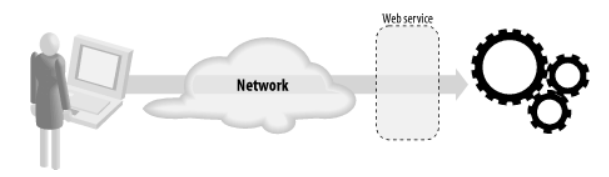
**Capítulo 1. Introduzindo Web Services**

Para fazer o melhor uso dos web services e do SOAP, você deve ter um entendimento sólido dos princípios e tecnologias sobre os quais eles se sustentam. Este capítulo é uma introdução a uma variedade de novas tecnologias, abordagens e ideias para escrever aplicações baseadas na web, de forma a tirar proveito da arquitetura de web services. O SOAP é uma parte de um quadro maior descrito neste capítulo, e você aprenderá como ele se relaciona com as outras tecnologias descritas neste livro: a **Web Service Description Language** (WSDL), a **Web Service Inspection Language** (WS-IL) e os serviços **Universal Description, Discovery, and Integration** (UDDI).

**1.1 O que é um Web Service?**

Antes de irmos mais longe, vamos definir o conceito básico de um "web service".  
Um web service é uma interface acessível pela rede para funcionalidades de uma aplicação, construída utilizando tecnologias padrão da Internet. Isso está ilustrado na Figura 1-1.

**Figura 1-1.** Um web service permite acesso ao código da aplicação utilizando tecnologias padrão da Internet.



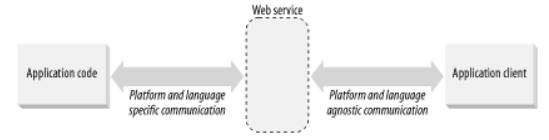
Em outras palavras, se uma aplicação puder ser acessada através de uma rede usando uma combinação de protocolos como HTTP, XML, SMTP ou Jabber, então ela é um web service. Apesar de todo o alvoroço da mídia em torno dos web services, é realmente simples assim.

Web services não são algo novo. Em vez disso, eles representam a evolução de princípios que têm guiado a Internet por anos.

**1.2 Fundamentos de Web Services**

Como as Figuras 1-1 e 1-2 ilustram, um web service é uma interface posicionada entre o código da aplicação e o usuário desse código. Ele atua como uma camada de abstração, separando os detalhes específicos de plataforma e linguagem de programação de como o código da aplicação é realmente invocado. Essa camada padronizada significa que qualquer linguagem que suporte o web service pode acessar a funcionalidade da aplicação.

**Figura 1-2.** Web services fornecem uma camada de abstração entre o cliente da aplicação e o código da aplicação.



Os web services que vemos implantados na Internet hoje são sites HTML. Neles, os serviços da aplicação — os mecanismos para publicar, gerenciar, pesquisar e recuperar conteúdo — são acessados por meio do uso de protocolos e formatos de dados padrão: HTTP e HTML. Aplicações clientes (navegadores) que entendem esses padrões podem interagir com os serviços da aplicação para realizar tarefas como encomendar livros, enviar cartões comemorativos ou ler notícias.

Graças à abstração fornecida pelas interfaces baseadas em padrões, não importa se os serviços da aplicação são escritos em Java e o navegador em C++, ou se os serviços da aplicação estão implantados em um servidor Unix enquanto o navegador está em Windows. Web services permitem interoperabilidade entre plataformas de forma que a plataforma se torna irrelevante.

Interoperabilidade é um dos principais benefícios obtidos com a implementação de web services. Soluções baseadas em Java e Microsoft Windows tipicamente têm sido difíceis de integrar, mas uma camada de web services entre aplicação e cliente pode reduzir bastante o atrito.

Atualmente, há um esforço contínuo dentro da comunidade Java para definir uma arquitetura exata para implementar web services dentro do framework da especificação Java 2 Enterprise Edition. Cada um dos principais fornecedores de tecnologia Java (Sun, IBM, BEA, etc.) está trabalhando para habilitar suas plataformas com suporte a web services.

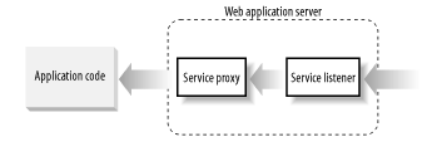
Muitos fornecedores de aplicações importantes, como IBM e Microsoft, abraçaram completamente os web services. A IBM, por exemplo, está integrando suporte a web services em todos os seus produtos WebSphere, Tivoli, Lotus e DB2. E a nova plataforma de desenvolvimento **.NET** da Microsoft é construída em torno de web services.

**1.2.1 Como são os Web Services**

Web services são uma estrutura de mensagens. O único requisito imposto a um web service é que ele deve ser capaz de enviar e receber mensagens usando alguma combinação de protocolos padrão da Internet. A forma mais comum de web services é chamar procedimentos que estão sendo executados em um servidor, caso em que as mensagens codificam "Chame esta sub-rotina com estes argumentos" e "Aqui estão os resultados da chamada da sub-rotina".

A **Figura 1-3** mostra as partes de um web service. O código da aplicação contém toda a lógica de negócios e o código para realmente realizar tarefas (listar livros, adicionar um livro ao carrinho de compras, pagar por livros, etc.). O **Service Listener** fala o protocolo de transporte (HTTP, SOAP, Jabber, etc.) e recebe solicitações de entrada. O **Service Proxy** decodifica essas solicitações em chamadas para o código da aplicação. O Service Proxy pode então codificar uma resposta para o Service Listener responder, mas é possível omitir essa etapa.

**Figura 1-3.** Um web service consiste em vários componentes-chave.



O Service Proxy e o Service Listener podem ser aplicações independentes (um daemon servidor TCP ou HTTP, por exemplo) ou podem ser executados dentro do contexto de algum outro tipo de servidor de aplicações. Como exemplo, o IBM WebSphere Application Server inclui suporte embutido para receber uma mensagem SOAP via HTTP e usá-la para invocar aplicações Java implantadas no WebSphere. Em comparação, o popular servidor web Apache possui um módulo que implementa SOAP. De fato, existem implementações de SOAP tanto para os sistemas operacionais Palm quanto PocketPC para PDAs.

Tenha em mente, no entanto, que web services não exigem um ambiente de servidor para serem executados. Web services podem ser implantados em qualquer lugar onde as tecnologias padrão da Internet possam ser usadas. Isso significa que web services podem ser hospedados ou utilizados por qualquer coisa, desde uma enorme fazenda de servidores de um Provedor de Serviços de Aplicação até um PDA.

Web services não exigem que as aplicações sigam um modelo tradicional cliente-servidor (onde o servidor mantém os dados e faz o processamento) ou de desenvolvimento em múltiplas camadas (onde o armazenamento de dados é separado da lógica de negócios, que é separada da interface do usuário), embora estejam sendo fortemente implantados nesses ambientes. Web services podem assumir qualquer forma, ser usados em qualquer lugar e servir a qualquer propósito. Por exemplo, há fortes intersecções entre sistemas peer-to-peer (com dados ou processamento descentralizado) e web services, onde pares usam protocolos padrão da Internet para fornecer serviços uns aos outros.

**1.2.2 Interseção entre Negócios e Programação**

Como um web service expõe a funcionalidade de uma aplicação para qualquer cliente em qualquer linguagem de programação, ele levanta questões interessantes tanto no mundo da programação quanto no mundo dos negócios.

Os programadores tendem a fazer perguntas como:

* "Como fazemos transações com commit em duas fases?"
* "Como faço herança de objetos?"
* "Como faço para que essa coisa rode mais rápido?"

São perguntas normalmente associadas ao ato de escrever código.

Já o pessoal de negócios tende a perguntar:

* "Como posso garantir que a pessoa usando o serviço é realmente quem diz ser?"
* "Como podemos conectar múltiplos web services em um fluxo de trabalho?"
* "Como posso garantir a confiabilidade das transações de web services?"

Suas perguntas normalmente abordam preocupações de negócio.

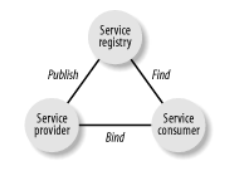
Essas duas perspectivas andam lado a lado. Cada questão de negócios terá uma solução baseada em software. Mas as duas perspectivas também podem estar em desacordo: os processos de negócios exigem completude, confiança, segurança e confiabilidade, o que pode ser incompatível com os objetivos dos programadores de simplicidade, desempenho e robustez.

**1.2.3 Integração Just-In-Time**

Depois que você entende os conceitos básicos de web services apresentados anteriormente, o próximo passo é adicionar a **Integração Just-In-Time**. Isto é, a integração dinâmica de serviços de aplicação baseada não na plataforma tecnológica em que os serviços foram implementados, mas nos requisitos de negócio do que precisa ser feito.

A Integração Just-In-Time reformula o modelo de desenvolvimento de aplicações para Internet em torno de um novo framework chamado **arquitetura de web services** (Figura 1-4).

**Figura 1-4.** A arquitetura de web services.



Na arquitetura de web services, o **provedor de serviço** publica uma descrição do(s) serviço(s) que oferece através do **registro de serviços**. O **consumidor de serviço** pesquisa no registro de serviços para encontrar um serviço que atenda às suas necessidades. O consumidor de serviço pode ser uma pessoa ou um programa.

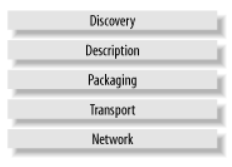
O termo *binding* refere-se a um consumidor de serviço realmente utilizar o serviço oferecido por um provedor de serviço. A chave para a integração Just-In-Time é que isso pode acontecer a qualquer momento, particularmente em tempo de execução (*runtime*). Ou seja, um cliente pode não saber quais procedimentos irá chamar até estar em execução, pesquisar no registro e identificar um candidato adequado. Isso é análogo ao *late binding* na programação orientada a objetos.

Imagine um web service de compras, onde consumidores solicitam produtos de um provedor de serviços. Se o programa cliente tiver codificado de forma fixa o servidor com o qual conversa, então o serviço está vinculado em tempo de compilação. Se o programa cliente pesquisa por um servidor adequado e se conecta a ele, então o serviço está vinculado em tempo de execução. Este último é um exemplo de integração Just-In-Time entre serviços.

**1.3 A Pilha de Tecnologias de Web Services**

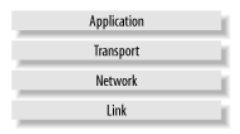
A arquitetura de web services é implementada por meio do empilhamento de cinco tipos de tecnologias, organizadas em camadas que se constroem umas sobre as outras (Figura 1-5).

**Figura 1-5.** A pilha de tecnologias de web services.



Não deve ser surpresa que essa pilha seja muito semelhante ao modelo de rede TCP/IP usado para descrever a arquitetura de aplicações baseadas na Internet (Figura 1-6).

**Figura 1-6.** O modelo de rede TCP/IP.



As camadas adicionais de empacotamento, descrição e descoberta na pilha de web services são essenciais para fornecer a capacidade de integração Just-In-Time e o modelo de programação independente de plataforma.

Como cada parte da pilha de web services aborda um problema de negócio diferente, você só precisa implementar aquelas partes que fazem mais sentido em um determinado momento. Quando uma nova camada da pilha for necessária, você não precisa reescrever grandes partes de sua infraestrutura apenas para suportar uma nova forma de trocar informações ou uma nova maneira de autenticar usuários.

O objetivo é a **modularização total** do ambiente de computação distribuída, em oposição à recriação das grandes soluções monolíticas das plataformas distribuídas tradicionais como Java, CORBA e COM. A modularidade é particularmente necessária em web services devido à natureza de rápida evolução dos padrões. Isso é mostrado no exemplo da aplicação **CodeShare** do Capítulo 7, onde não usamos a camada de descoberta, mas incluímos outro padrão XML para lidar com segurança.

**1.3.1 Além da Pilha**

As camadas da pilha de web services não fornecem uma solução completa para muitos problemas de negócios. Por exemplo, elas não tratam de segurança, confiança, fluxo de trabalho, identidade ou muitas outras preocupações de negócio. Aqui estão algumas das iniciativas de padronização mais importantes que estão atualmente sendo perseguidas nessas áreas:

* **XML Protocol** – O grupo de trabalho W3C XML Protocol tem como objetivo padronizar o protocolo SOAP. Seu trabalho eventualmente substituirá completamente o protocolo SOAP como o padrão *de facto* para implementar web services.
* **XKMS** – *XML Key Management Services*, um conjunto de serviços relacionados a segurança e confiança que adicionam capacidades de PKI (*Private Key Infrastructure*) a web services.
* **SAML** – *Security Assertions Markup Language*, uma gramática XML para expressar a ocorrência de eventos de segurança, como um evento de autenticação. Usada na arquitetura de web services, fornece um sistema de autenticação padrão e flexível.
* **XML-Dsig** – *XML Digital Signatures*, permite que qualquer documento XML seja assinado digitalmente.
* **XML-Enc** – *XML Encryption*, especificação que permite que dados XML sejam criptografados e que a expressão de dados criptografados seja feita como XML.
* **XSD** – *XML Schemas*, aplicação de XML usada para expressar a estrutura de documentos XML.
* **P3P** – *Platform for Privacy Preferences* do W3C, gramática XML para a expressão de políticas de privacidade de dados.
* **WSFL** – *Web Services Flow Language*, extensão ao WSDL que permite a expressão de fluxos de trabalho dentro da arquitetura de web services.
* **Jabber** – Novo protocolo de transporte assíncrono e leve usado em aplicações peer-to-peer.
* **ebXML** – Conjunto de especificações baseadas em XML para conduzir negócios eletrônicos. Construído para usar SOAP, o ebXML oferece uma abordagem para implementar serviços de integração *business-to-business*.

**1.3.2 Descoberta**

A camada de descoberta fornece o mecanismo para consumidores obterem as descrições dos provedores. Um dos mecanismos de descoberta mais amplamente reconhecidos é o projeto **UDDI** (*Universal Description, Discovery, and Integration*). A IBM e a Microsoft propuseram conjuntamente uma alternativa ao UDDI, a **Web Services Inspection Language** (WS-Inspection). Discutiremos tanto o UDDI quanto o WS-Inspection em detalhes (incluindo argumentos a favor e contra seu uso) no Capítulo 6.

**1.3.3 Descrição**

Quando um web service é implementado, ele deve tomar decisões em todos os níveis sobre quais protocolos de rede, transporte e empacotamento irá suportar. Uma descrição desse serviço representa essas decisões de forma que o consumidor de serviço possa contatar e usar o serviço.

O **WSDL** (*Web Service Description Language*) é o padrão *de facto* para fornecer essas descrições. Outras abordagens menos populares incluem o uso do **Resource Description Framework** (RDF) do W3C e a **DARPA Agent Markup Language** (DAML), ambas oferecendo uma capacidade muito mais rica (mas também mais complexa) de descrever web services do que o WSDL.

Nós cobriremos o WSDL no Capítulo 5. Você pode encontrar mais informações sobre DAML e RDF em:

* <http://daml.semanticweb.org/>
* <http://www.w3.org/rdf>

**1.3.4 Empacotamento**

Para que dados de aplicação sejam movidos pela rede pela camada de transporte, eles precisam ser “empacotados” em um formato que todas as partes possam entender (outros termos para este processo são “serialização” e “marshalling”). Isso inclui a escolha de tipos de dados compreendidos, a codificação de valores, e assim por diante.

O HTML é um tipo de formato de empacotamento, mas pode ser inconveniente de trabalhar porque está fortemente ligado à apresentação da informação em vez de seu significado. O XML é a base para a maioria dos formatos de empacotamento atuais de web services porque pode ser usado para representar o significado dos dados transferidos e porque os analisadores XML são agora onipresentes.

O **SOAP** é um formato de empacotamento muito comum, construído sobre XML. No Capítulo 2, veremos como o SOAP codifica mensagens e valores de dados, e no Capítulo 3 veremos como escrever web services reais com SOAP. Há vários protocolos de empacotamento baseados em XML disponíveis para desenvolvedores (como o XML-RPC, por exemplo), mas, como você pode ter adivinhado pelo título deste livro, o SOAP é o único formato que cobriremos.

**1.3.5 Transporte**

A camada de transporte inclui as várias tecnologias que permitem comunicação direta de aplicação para aplicação sobre a camada de rede. Tais tecnologias incluem protocolos como **TCP**, **HTTP**, **SMTP** e **Jabber**. O papel principal da camada de transporte é mover dados entre dois ou mais locais na rede. Web services podem ser construídos sobre praticamente qualquer protocolo de transporte.

A escolha do protocolo de transporte é baseada, em grande parte, nas necessidades de comunicação do web service que está sendo implementado. O HTTP, por exemplo, fornece o suporte de firewall mais onipresente, mas não oferece suporte para comunicação assíncrona. O Jabber, por outro lado, embora não seja um padrão, fornece um bom canal de comunicação assíncrona.

**1.3.6 Rede**

A camada de rede na pilha de tecnologias de web services é exatamente a mesma que a camada de rede no modelo de rede TCP/IP. Ela fornece as capacidades críticas básicas de comunicação, endereçamento e roteamento.

**1.4 Aplicação**

A camada de aplicação é o código que implementa a funcionalidade do web service, a qual é localizada e acessada através das camadas inferiores da pilha.

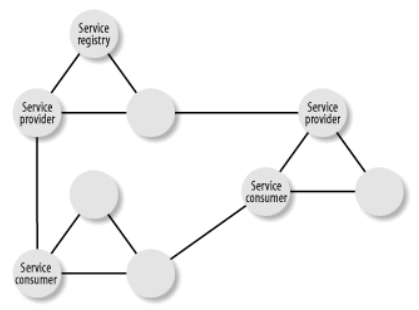
**1.5 O Modelo de Serviços Peer**

O modelo de serviços peer é uma visão complementar, mas alternativa, da arquitetura de web services. Baseado na arquitetura *peer-to-peer* (P2P), cada membro de um grupo de *peers* compartilha uma coleção comum de serviços e recursos. Um *peer* pode ser uma pessoa, uma aplicação, um dispositivo ou outro grupo de *peers* operando como uma única entidade.

Embora possa não ser imediatamente aparente, os mesmos componentes fundamentais de web services estão presentes na arquitetura de serviços peer. Existem tanto provedores de serviço quanto consumidores de serviço, e existem registros de serviço. A distinção entre provedores e consumidores, no entanto, não é tão clara como no caso dos web services. Dependendo do tipo de serviço ou recurso que os *peers* estão compartilhando, qualquer *peer* individual pode desempenhar o papel de provedor de serviço e consumidor de serviço. Isso torna o modelo de serviços peer mais dinâmico e flexível.

A **Mensageria Instantânea** (*Instant Messaging*) é a implementação mais amplamente utilizada do modelo de serviços peer. Cada pessoa que usa mensageria instantânea é um *peer*. Quando você recebe um convite para conversar com alguém, você está desempenhando o papel de provedor de serviço. Quando você envia um convite para conversar com outra pessoa, você está desempenhando o papel de consumidor de serviço. Quando você se conecta ao servidor de mensageria instantânea, o servidor está desempenhando o papel de registro de serviço — ou seja, o servidor de mensageria instantânea mantém o controle de onde você está atualmente e quais são suas capacidades de mensageria instantânea. A Figura 1-7 ilustra isso.

**Figura 1-7.** O modelo de web services *peer* simplesmente aplica os conceitos da arquitetura de web services em uma rede *peer-to-peer*.



Serviços *peer* e web services surgiram e evoluíram separadamente, e consequentemente fazem uso de protocolos e tecnologias diferentes para implementar seus respectivos modelos. **Web services peer** unem os dois, unificando as tecnologias, os protocolos e os modelos em um único grande quadro abrangente. A implementação de um web service *peer* será o foco central do Capítulo 7.

📄 **Sumário deste capítulo no arquivo**

1. Introducing Web Services  
    1.1 What Is a Web Service?  
    1.2 Web Service Fundamentals  
    1.3 The Web Service Technology Stack  
    1.4 Application  
    1.5 The Peer Services Model