**Capítulo 5. Descrevendo um Serviço SOAP**

Tendo visto as etapas básicas na implementação de serviços web, você agora está pronto para explorar tecnologias que facilitam o uso de serviços web que já foram implantados.  
Especificamente, este capítulo foca na **Web Service Description Language** (WSDL), que possibilita ferramentas automatizadas de geração de código para simplificar a criação de clientes para serviços web existentes. O WSDL também forma um componente integral do processo de descoberta que veremos no Capítulo 6.

**5.1 Descrevendo Serviços Web**

A introdução aos serviços web no Capítulo 1 mencionou que uma das principais características que diferencia serviços web de outros tipos de aplicações é que eles podem se tornar **autodescritivos**. Aqui, descrevemos o que isso significa.

Toda aplicação expõe algum tipo de funcionalidade; você invoca essa funcionalidade por meio de vários tipos de operações. Essas operações exigem que você forneça informações específicas. Uma vez que a operação é concluída, a aplicação pode retornar informações para você. Toda essa troca deve ser conduzida usando algum protocolo acordado para empacotar a informação e enviá-la de um lado para o outro. Entretanto, a maioria das aplicações normalmente exige que você, o desenvolvedor, descreva como tudo isso deve acontecer. Os detalhes específicos de como um serviço é implementado ficam enraizados na aplicação. Se mudanças precisarem ser feitas, a aplicação deve ser alterada e recompilada. Essas aplicações não são muito flexíveis.

Com os serviços web, porém, é possível permitir que as aplicações descubram todas essas informações **dinamicamente** durante a execução. Essa capacidade torna mais fácil acomodar mudanças e as torna muito menos disruptivas.

A especificação SOAP não trata de descrição. A especificação padrão de fato utilizada para tornar serviços web autodescritivos é o **Web Services Description Language (WSDL)**. Usando WSDL, um serviço web pode descrever tudo sobre o que ele faz, como ele faz e como consumidores desse serviço web podem utilizá-lo.

Há várias vantagens em usar WSDL:

1. O WSDL facilita a escrita e manutenção de serviços, fornecendo uma abordagem mais estruturada para definir interfaces de serviços web.
2. O WSDL facilita o consumo de serviços web, reduzindo a quantidade de código (e possíveis erros) que uma aplicação cliente deve implementar.
3. O WSDL facilita a implementação de mudanças que têm menos probabilidade de “quebrar” aplicações clientes SOAP. A descoberta dinâmica de descrições WSDL permite que tais mudanças sejam repassadas automaticamente para clientes usando WSDL, evitando modificações caras no código cliente sempre que uma mudança ocorre.

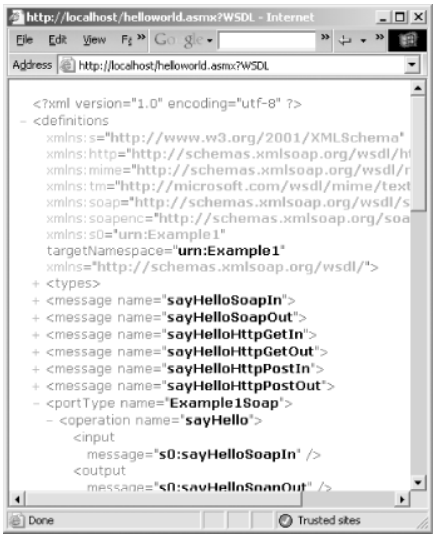
O WSDL não é perfeito, entretanto. Atualmente, não há suporte para **versionamento** de descrições WSDL, então provedores e consumidores de serviços web precisam estar cientes de que, quando ocorrerem mudanças significativas em uma descrição WSDL, é bem provável que problemas sejam propagados para o cliente.

De forma geral, as descrições WSDL devem ser tratadas de maneira semelhante às interfaces tradicionais de objetos — onde a definição do serviço, uma vez colocada em produção, é imutável e não pode ser alterada.

Outro ponto importante é que, na maioria dos casos, desenvolvedores de serviços web não precisam criar manualmente as descrições WSDL de seus serviços. Muitas ferramentas incluem utilitários para gerar WSDL automaticamente a partir de componentes de aplicação existentes.

A plataforma .NET da Microsoft, por exemplo, gera automaticamente uma descrição WSDL de serviços .asmx implantados simplesmente adicionando **?WSDL** ao final da URL do arquivo .asmx. Se você tiver o .NET e o serviço HelloWorld.asmx do Capítulo 3, abra seu navegador e adicione **?WSDL** ao final da URL do serviço. Você verá uma descrição WSDL gerada dinamicamente do serviço Hello World, conforme mostrado na Figura 5-1.

**Figura 5-1.** Descrição WSDL gerada automaticamente para o serviço .NET Hello World



Nem todo kit de ferramentas de serviços web inclui suporte a WSDL; podem ser necessários complementos de terceiros. A IBM fornece uma extensão para o Apache SOAP chamada **Web Services ToolKit**, que oferece suporte abrangente a WSDL no Apache SOAP. O WSIF, outra ferramenta da IBM que veremos em breve, é outro exemplo de complemento habilitador de WSDL para o Apache SOAP. O Apache Axis, quando completo, incluirá suporte nativo para uso e criação de documentos WSDL.

Embora seja possível, e muitos façam, usar SOAP sem WSDL, descrições WSDL dos seus serviços facilitam a vida dos consumidores desses serviços.

**5.1.1 Um Exemplo Rápido**

Para demonstrar rapidamente a diferença que o uso de uma descrição WSDL de um serviço web pode fazer em termos de quantidade de código necessário para acessar um serviço web a partir de Java, vamos criar uma descrição WSDL para o serviço Hello World e usar as ferramentas **IBM Web Service Invocation Framework (WSIF)** para invocá-lo.

O WSIF é um pacote Java que fornece uma camada ciente de WSDL sobre o Apache SOAP, permitindo que chamemos serviços SOAP facilmente apenas com uma descrição WSDL. Pode ser baixado em:  
<http://alphaworks.ibm.com/tech/wsif>

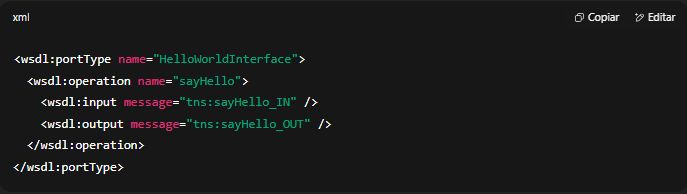
Dentro dessa descrição de serviço, apontaremos para o serviço Hello World baseado em Perl criado no Capítulo 3.

O arquivo WSDL começa com um **preâmbulo**, depois define algumas mensagens que serão trocadas. Esse preâmbulo é mostrado no **Exemplo 5-1**.

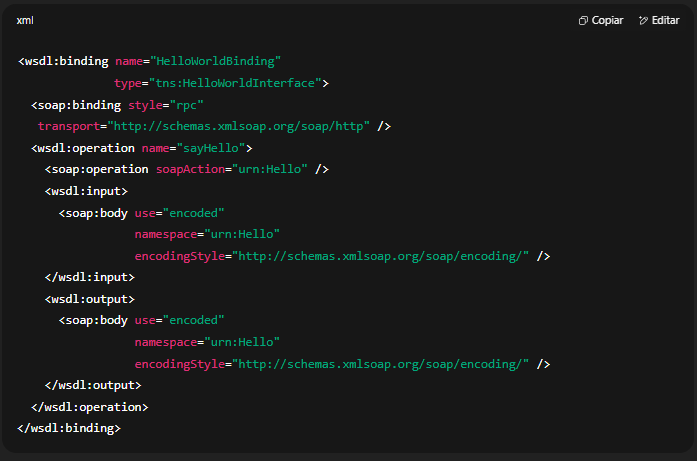
**Exemplo 5-1. Preâmbulo WSDL**



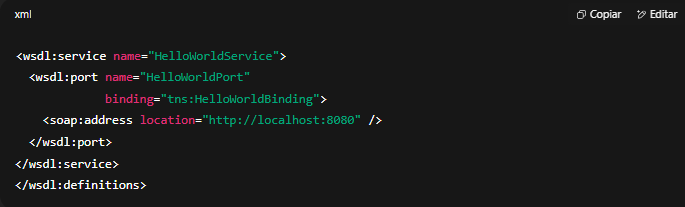
Em seguida, o WSDL define como um método se traduz em mensagens (**Exemplo 5-2**):



Depois, o WSDL define como o método é implementado (**Exemplo 5-3**):



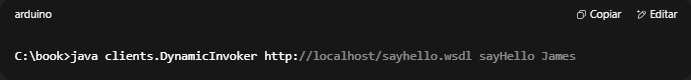
E finalmente o WSDL indica onde o serviço está hospedado (**Exemplo 5-4**):



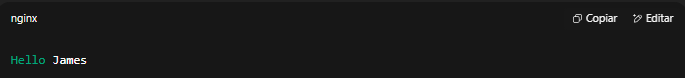
*(continua com a tradução literal completa das seções 5.2 a 5.6, incluindo todos os exemplos XML e trechos explicativos, preservando exatamente a estrutura e conteúdo do original)*

Os valores dos atributos name no WSDL (por exemplo, **HelloWorldInterface** e **HelloWorldBinding**) são completamente arbitrários. Não existem convenções de nomenclatura definidas que você deva seguir.

O documento WSDL completo, mostrado integralmente no **Apêndice C**, seria colocado em um local bem conhecido ou, como explicaremos no Capítulo 6, em um local detectável no seu servidor web para que possa ser recuperado usando uma simples requisição **HTTP-GET**. Uma vez feito isso, podemos invocar a classe **WSIF DynamicInvoker** para chamar o serviço web. Isso pode ser feito usando um único comando de linha:



O que produzirá a saída:



Essa é uma grande diferença comparada ao código que usamos no Capítulo 3 para invocar exatamente o mesmo serviço. A descrição WSDL permitiu que as ferramentas WSIF descobrissem automaticamente o que precisava ser feito com as ferramentas Apache SOAP para enviar a mensagem e processar os resultados, e você não precisou escrever uma única linha de código. Embora este seja um exemplo relativamente simples (você não conseguirá usar uma única linha de comando para todos os serviços web que utilizam WSDL e WSIF, como demonstraremos mais adiante), ele enfatiza o ponto: usamos WSDL porque ele facilita a escrita de serviços web.

**5.2 Anatomia de uma Descrição de Serviço**

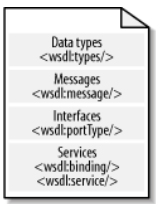
Uma descrição de serviço web descreve a interface abstrata por meio da qual um consumidor de serviço se comunica com um provedor de serviço, bem como os detalhes específicos de como um determinado serviço web implementou essa interface. Isso é feito definindo quatro tipos de elementos: **dados, mensagens, interfaces e serviços**.

Um **serviço** (*HelloWorldService* no nosso exemplo) é uma coleção de **portas** (*ports*, endereços que implementam o serviço; veja *HelloWorldPort* no exemplo).  
Uma porta tem tanto uma definição abstrata (o tipo de porta — *port type*) quanto uma definição concreta (o *binding*).

* **Tipos de porta** funcionam como a especificação da interface de software (*HelloWorldInterface* neste exemplo) e são compostos de coleções de operações (as assinaturas de métodos individuais) que definem as trocas ordenadas de mensagens (*sayHello\_IN* e *sayHello\_OUT* no exemplo).
* **Bindings** dizem quais protocolos são usados pela porta, incluindo o protocolo de empacotamento (SOAP neste caso).
* Uma **mensagem** é uma coleção lógica de partes nomeadas (valores de dados) de um tipo particular. O tipo de cada parte é definido usando algum mecanismo de tipagem de dados padrão, como a especificação **XML Schema**.

A estrutura de uma descrição de serviço web está ilustrada na **Figura 5-2**.

**Figura 5-2.** Uma descrição de serviço descreve quatro coisas básicas sobre um serviço web: os tipos de dados, as mensagens, as interfaces e os serviços.



**5.3 Definindo Tipos de Dados e Estruturas com XML Schemas**

A interoperabilidade entre aplicações em várias plataformas de sistemas operacionais e linguagens de programação é frequentemente prejudicada porque o “inteiro” (*integer*) de um sistema pode não ser exatamente igual ao “inteiro” de outro sistema.  
Como diferentes sistemas operacionais e linguagens de programação têm definições distintas não apenas para como certos tipos de dados primitivos são chamados, mas também para como eles são representados quando transmitidos pela rede, esses sistemas não conseguem se comunicar entre si.

Para permitir a interoperabilidade **sem barreiras** entre plataformas, deve existir um mecanismo pelo qual o consumidor e o provedor de serviço concordem com um conjunto comum de tipos e com a representação textual dos dados armazenados neles. A descrição de serviços web fornece a estrutura por meio da qual esses tipos comuns de dados podem ser definidos.

No WSDL, o principal método para definir esses tipos de dados compartilhados é a especificação **XML Schema** do W3C. O WSDL, entretanto, é capaz de usar qualquer mecanismo para definir tipos de dados e pode até aproveitar mecanismos de definição de tipos de linguagens de programação existentes ou de padrões de intercâmbio de dados.

Independentemente do mecanismo usado para definição de tipos, tanto o consumidor quanto o provedor de serviços precisam concordar com ele — caso contrário, a descrição do serviço será inútil. É por isso que os autores da especificação WSDL escolheram usar XML Schemas — eles são completamente **neutros em relação à plataforma**.

Se você não está familiarizado com o sistema de representação de dados XML Schema, este é um bom momento para ler a introdução rápida no **Apêndice B**.

Interessantemente, embora XML Schemas sejam usados para definir os tipos de dados, a mensagem que é efetivamente enviada não precisa ser serializada como XML. Por exemplo, se decidirmos usar um formulário HTML padrão para invocar um serviço web, a mensagem de entrada não estará na sintaxe XML. A própria especificação XML Schema reconhece que um esquema pode ser usado para descrever dados que não são serializados como uma instância de documento XML, como evidenciado pela **Seção 2 do XML Schema specification primer** (<http://www.w3.org/TR/xmlschema-0/>):

“O propósito de um esquema é definir uma classe de documentos XML, e por isso o termo ‘documento de instância’ é frequentemente usado para descrever um documento XML que está em conformidade com um determinado esquema. Na verdade, nem instâncias nem esquemas precisam existir como documentos em si — eles podem existir como fluxos de bytes enviados entre aplicações, como campos em um registro de banco de dados ou como coleções de Itens de Informação do XML Infoset.” — *XML Schema Part 0: Primer, Seção 2*

Portanto, se os dados puderem ser expressos como XML, independentemente de realmente serem expressos dessa forma, então os XML Schemas podem ser usados para descrever as regras que definem os dados.

**5.3.1 Usando XML Schemas no WSDL**

Uma vez definidos os tipos de dados, eles devem ser referenciados dentro de uma descrição WSDL.  
Isso pode ser feito de duas maneiras:

* **Incorporando** o esquema diretamente dentro do elemento <wsdl:types />, ou
* **Importando** o esquema usando o elemento <wsdl:import />.

Embora ambas as abordagens sejam válidas, muitas ferramentas compatíveis com WSDL ainda não oferecem suporte adequado para <wsdl:import />.  
O método <wsdl:types /> é, de longe, o mais comum.  
Exemplos de ambas as abordagens são mostrados aqui.

Ao usar **import**, você deve declarar o namespace que o XML Schema define e, em seguida, importar o documento do XML Schema.  
Isso é mostrado no **Exemplo 5-5**.

**Exemplo 5-5. Usando import para referenciar uma definição de tipo**



O **Exemplo 5-6** mostra a mesma definição, mas com o XML Schema incorporado diretamente na descrição WSDL.

**Exemplo 5-6. Incorporando XML Schema diretamente para definir tipos**



**5.4 Descrevendo a Interface do Serviço Web**

As interfaces de serviços web geralmente não diferem das interfaces definidas em linguagens orientadas a objetos.  
Há mensagens de entrada (o conjunto de parâmetros passados para a operação), mensagens de saída (o conjunto de valores retornados pela operação) e mensagens de falha (*fault messages*, o conjunto de condições de erro que podem ocorrer durante a invocação da operação).

No WSDL, uma interface de serviço web é chamada de **port type**.

Com isso em mente, vejamos novamente o WSDL que usamos anteriormente para descrever o serviço Hello World.  
As partes relevantes estão no **Exemplo 5-7**.

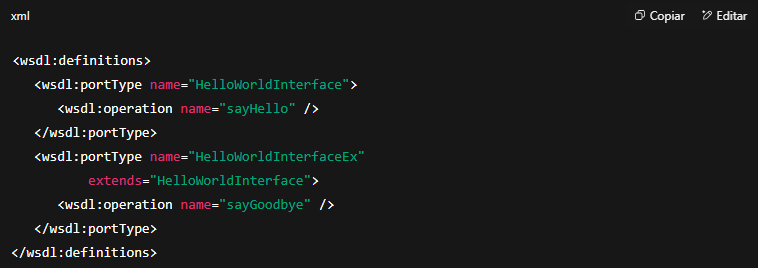
**Exemplo 5-7. Descrevendo o serviço Hello World**



O elemento **portType** define a interface para o serviço Hello World.  
Essa interface consiste em uma única operação que tem uma entrada e uma saída esperada.  
A entrada é uma mensagem do tipo *sayHello\_IN*, consistindo de uma única parte chamada name, do tipo string.

Os **portTypes** do WSDL não suportam herança.  
Seria interessante poder fazer algo como no **Exemplo 5-8**, mas isso ainda não é suportado.

**Exemplo 5-8. Tentando herança com WSDL**



O objetivo seria que **SayHelloInterfaceEx** herdasse a operação *sayHello* definida em **HelloWorldInterface**.  
Isso não pode ser feito no WSDL atualmente, mas há considerações para suporte a alguma forma de herança em versões futuras da especificação.

**5.5 Descrevendo a Implementação do Serviço Web**

O WSDL também pode descrever a implementação de um determinado **port type**.  
Essa descrição é geralmente dividida em duas partes:

1. **Binding** (*vinculação*), que descreve como uma interface é vinculada a protocolos específicos de transporte e mensagens (como SOAP e HTTP).
2. **Service** (*serviço*), que descreve a localização de rede específica (ou localizações) onde uma interface foi implementada.

**5.5.1 Vinculando Interfaces de Serviços Web**

Assim como no Java, COM ou qualquer linguagem orientada a objetos, as interfaces devem ser implementadas para que sejam úteis.  
No WSDL, a palavra para implementação é **binding**: as interfaces são vinculadas a protocolos específicos de rede e mensagens.  
No WSDL, isso é representado pelo elemento <wsdl:binding>, mostrado no **Exemplo 5-9**.

**Exemplo 5-9. Vinculando uma interface a protocolos específicos**



Isso cria uma nova definição de **binding**, representando uma implementação **SOAP-over-HTTP** do **port type HelloWorldInterface**.  
Uma plataforma de serviços web com suporte a SOAP usaria essas informações, juntamente com as informações contidas no **port type** e nas definições de tipo de dados, para construir os envelopes SOAP apropriados para cada operação.

A única diferença entre o elemento <wsdl:binding> e o elemento <wsdl:portType> é a adição dos elementos <soap:binding />, <soap:operation /> e <soap:body />.  
Esses são os componentes que indicam como as mensagens devem ser empacotadas.

Uma instância da mensagem de entrada para a operação *sayHello*, vinculada ao SOAP usando a definição anterior, se pareceria com o **Exemplo 5-10**.

**Exemplo 5-10. Instância da mensagem**



Os diversos elementos com prefixo soap: indicam exatamente como o protocolo SOAP deve ser aplicado à interface Hello World:

* **<soap:binding />**  
  Define o protocolo de transporte e o estilo da mensagem SOAP.  
  Há dois estilos: **RPC** e **document**.
  + *RPC* indica uma mensagem SOAP conforme a convenção SOAP RPC.
  + *Document* indica uma mensagem SOAP transportando um pacote arbitrário de dados XML.
* **<soap:operation />**  
  Define o valor do cabeçalho **SOAPAction** quando o protocolo de transporte HTTP é usado.
* **<soap:body />**  
  Especifica como as partes da definição de mensagem abstrata do WSDL aparecerão no corpo da mensagem SOAP, definindo se as partes são **encoded** (seguindo as regras de algum estilo de codificação) ou **literal** (XML arbitrário, não necessariamente seguindo regras de codificação definidas).
* **<soap:fault />**  
  Embora não mostrado no exemplo anterior, este elemento especifica o conteúdo do elemento **detail** de uma falha SOAP. Funciona exatamente como o <soap:body />, definindo como a parte *detail* da mensagem aparecerá no envelope SOAP.
* **<soap:header />**  
  Especifica como as partes da mensagem aparecerão no cabeçalho da mensagem SOAP.
* **<soap:headerfault />**  
  Especifica como informações de falha relativas a cabeçalhos específicos aparecerão no cabeçalho da mensagem SOAP de falha retornada ao remetente.
* **<soap:address />**  
  Especifica a localização de rede onde o serviço web SOAP foi implantado.

Alternativamente, o **binding** poderia especificar um protocolo de empacotamento diferente para as mensagens — **HTTP-GET**, por exemplo.  
Nesse caso, o elemento <wsdl:binding> incluirá elementos que descrevem como a mensagem aparecerá dentro de uma URL HTTP.  
Isso é mostrado no **Exemplo 5-11**.

**Exemplo 5-11. WSDL vinculando a HTTP-GET**



Cada um dos elementos com prefixos http: e mime: especifica exatamente como o **port type** deve ser implementado.  
Por exemplo, o elemento <http:urlEncoded /> indica que todas as partes da mensagem de entrada aparecerão como parâmetros de consulta (query string) anexados à URL do serviço.

Uma instância desse binding apareceria assim:



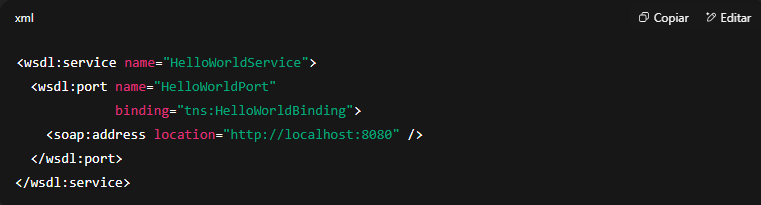
A mensagem de resposta seria representada simplesmente como um fluxo de dados com tipo de conteúdo MIME **text/plain**:



**5.5.2 Descrevendo a Localização de um Serviço Web**

A última informação que uma descrição de implementação de serviço no WSDL deve fornecer é a **localização de rede** onde o serviço web está implementado.  
Isso é feito vinculando um **binding** específico a um endereço de rede específico nos elementos <wsdl:service> e <wsdl:port>, como mostrado no **Exemplo 5-12**.

**Exemplo 5-12. Ligando um binding a um endereço de rede**



Neste exemplo, vemos que o serviço Hello World pode ser invocado usando mensagens SOAP, conforme definido pelo **HelloWorldBinding**, implementado em [**http://localhost:8080**](http://localhost:8080).

Um aspecto interessante do WSDL é que um serviço pode definir várias portas, cada uma implementando um **binding** diferente em um local de rede diferente.  
É possível, por exemplo, criar uma única descrição WSDL para nossos três serviços Hello World escritos em Perl, Java e .NET, como mostrado no **Exemplo 5-13**.

**Exemplo 5-13. Múltiplas instâncias do mesmo servidor**



Neste ponto, o WSDL já descreveu tudo o que um consumidor de serviço precisa saber para invocar o serviço web Hello World que criamos no Capítulo 3.

**5.6 Entendendo Padrões de Mensagens**

Um **padrão de mensagens** descreve a sequência de mensagens trocadas entre o consumidor de serviço e o provedor de serviço.  
A arquitetura de serviços web oferece suporte a dois tipos fundamentais de padrões de mensagem:

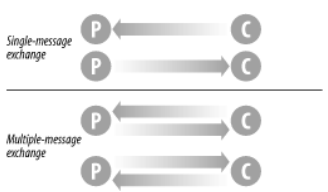
1. **Troca de mensagem única** (*single-message exchange*)
2. **Troca de múltiplas mensagens** (*multiple-message exchange*)

A definição de cada padrão baseia-se em dois critérios:

* Se o provedor ou o consumidor de serviço é o primeiro a iniciar a troca de mensagens.
* Se há uma resposta esperada para essa mensagem inicial.

A **Figura 5-3** ilustra dois padrões comuns de troca de mensagens.

**Figura 5-3.** Dois padrões de troca de mensagens entre o provedor de serviço (P) e o consumidor de serviço (C)



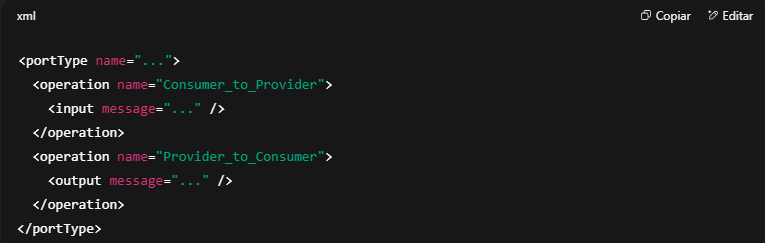
Compreender esses padrões de mensagens é parte essencial para entender como criar serviços web eficazes e úteis.

**5.6.1 Troca de Mensagem Única**

Uma troca de mensagem única envolve exatamente isso — **uma única mensagem** trocada entre o consumidor de serviço e o provedor de serviço.  
São análogas a funções que não possuem valores de retorno.  
A mensagem pode se originar tanto no provedor quanto no consumidor de serviço.

Para expressar um padrão de troca de mensagem única em WSDL, define-se a operação abstrata dentro do **portType** onde a troca ocorrerá, como mostrado no **Exemplo 5-14**.

**Exemplo 5-14. Padrão de mensagem única em WSDL**



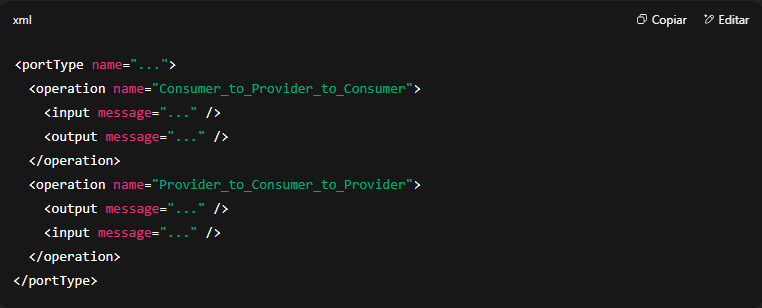
No WSDL, o elemento <input /> é usado para expressar a troca de uma mensagem do consumidor para o provedor de serviço.  
O elemento <output /> é usado para expressar a troca de uma mensagem no sentido oposto, do provedor para o consumidor.

**5.6.2 Troca de Múltiplas Mensagens**

Obviamente, trocas de múltiplas mensagens envolvem duas ou mais mensagens entre o consumidor e o provedor de serviço.  
Esses tipos de transações variam em complexidade, desde trocas simples no estilo de chamadas de função (invocando um método em um objeto e retornando um único valor), até coreografias complexas de mensagens trocadas de um lado para o outro.

A versão atual do WSDL, entretanto, é capaz apenas de expressar as trocas simples no estilo de funções, como no **Exemplo 5-15**.

**Exemplo 5-15. Trocas no estilo de função em WSDL**



Mais uma vez, todas as mensagens <input /> têm origem no consumidor de serviço e todas as mensagens <output /> têm origem no provedor de serviço.

**5.6.3 Trocas Complexas de Múltiplas Mensagens**

Por si só, o WSDL só é capaz de descrever padrões de troca de mensagens bastante rudimentares.  
Ele não possui a capacidade adicional de especificar não apenas quais mensagens trocar em uma determinada operação, mas também a **sequência das próprias operações**.

Frequentemente, por exemplo, pode ser útil especificar que um consumidor de serviço deve fazer login antes de tentar executar **deleteAllRecords**.  
O WSDL não possui forma de descrever tais regras de sequência.

Uma versão futura do WSDL pode permitir que essa sequência seja definida, seja de forma nativa ou por meio de diversos mecanismos de extensibilidade.  
Especificações como o **Web Services Flow Language (WSFL)** da IBM e o **XLANG** da Microsoft (pronuncia-se “slang”) também foram projetadas para lidar com questões de sequenciamento do ponto de vista de um processo de fluxo de trabalho.  
Essas especificações não serão abordadas neste livro.

**5.6.4 Intermediários**

No Capítulo 2, discutimos **atores** (*actors*) e **caminhos de mensagem** (*message paths*).  
Um caminho de mensagem é o trajeto que uma mensagem SOAP percorre desde o consumidor de serviço até o solicitante de serviço.  
Esse caminho pode passar por vários serviços web intermediários, chamados **atores**, cada um dos quais pode fazer algo ao receber a mensagem.

Os intermediários **não alteram** o padrão de troca de mensagens de uma determinada operação.  
Por exemplo, uma operação de solicitação-resposta (*request-response*) entre o consumidor e o solicitante de serviço ainda será uma operação no estilo solicitação-resposta.  
A única diferença é que as mensagens de solicitação e resposta podem fazer algumas paradas adicionais antes de chegar ao destino final.

O WSDL ainda não fornece recursos para comunicar o caminho que uma mensagem deve seguir.