

## PHP5 e Web Services

por Pablo Dall'Oglio

### TAGS

PHP, Web Services, SOAP, WSDL, XML.

### Resumo

Este artigo visa discorrer de forma sucinta sobre o conceito de Web Services, as tecnologias e padrões necessários para sua implementação, bem como vislumbrar o novo cenário que se configura no mercado de software em relação à metodologias de desenvolvimento e sobre como as empresas podem utilizar essa tecnologia para tornarem-se mais competitivas, analisando os pontos de impacto em seus processos de construção de sistemas. Para ilustrar tudo, serão exibidos exemplos práticos, demonstrando como integrar aplicações usando web services através do PHP.

### 1. Introdução

Web Services são uma das maiores inovações advindas da utilização da internet. Web Services permitem aplicações se comunicarem umas com as outras, combinando funcionalidades de forma independente de plataforma ou linguagem [Chavda].

Web Services são serviços disponibilizados pela internet, através de um conjunto de tecnologias independentes de plataforma, que permite interoperabilidade através de aplicações, através da entrega de serviços [Vaughan] e a comunicação entre aplicações modularizadas, que podem ser descritas, publicadas e invocadas pela internet para utilização imediata, ou mesmo composição de novos serviços [Hansen] integrando tudo através de padrões abertos e conhecidos como XML, SOAP, WSDL e UDDI [Chung].

Ao contrário de tecnologias como Corba, que propõem um protocolo específico, WS provêm interoperabilidade através de XML [Vaughan] e da utilização de padrões abertos como SOAP, o que é uma grande vantagem em relação aos seus antecessores [Chung].

Web Services podem ser utilizados com muita vantagem sobre tecnologias EDI, CORBA, DCOM, RPC, RMI, dentre outras, devido ao fato destas serem de difícil implementação ou utilizarem padrões fechados, não ganhando aceitação dentre os fornecedores de software.

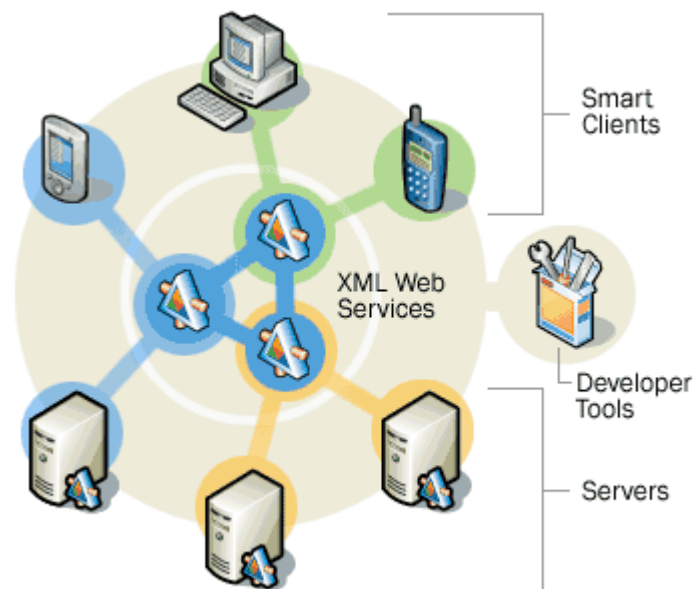


Figura 1. Um ambiente utilizando web services

Uma das características mais atraentes na adoção de WS é a possibilidade de criar novas aplicações baseadas em componentes de sistemas legados, preservando investimentos, possibilitando interoperabilidade, disponibilidade no ambiente web e uma oportunidade para reverem seus processos de negócio [Arsanjani].

Uma gama enorme de aplicações imediatas surge no contexto de WS. A interoperabilidade provida por WS, torna possível agregar e publicar dinamicamente aplicações como B2B e CMS [Chung]. Diversos serviços pessoais como catálogo de endereços e anotações de viagem poderiam ser desenvolvidos através de WS [Vaughan].

A internet, que começou como um esforço de pesquisa militar e educacional, hoje se tornou a base do comércio do século XXI. Neste contexto, os Web Services oferecem um novo modo de comunicação entre aplicações e novas formas de negócio pela Web [Chavda]. Web Services podem ser utilizados dentro das empresas (intranet), comunicando aplicações já existente, mas sua

maior vantagem fica visível quando estes são expostos na internet combinando serviços entre organizações.

As primeiras iniciativas de integração e utilização de WS partirão primeiro dentro empresas e depois entre as mesmas. Companias que já exercem frequente troca de dados entre aplicações são fortes candidatas à essa tecnologia [Chung], utilizando WS para integrar aplicações já existentes (sistemas internos e externos) [Vaughan].

## 2. Arquitetura

Através da figura a seguir, temos um panorama geral do funcionamento de Web Services, através de um conjunto de tecnologias de padrão aberto, interagindo sob uma plataforma de internet. Temos o papel do Requerente, que é a aplicação que interage com o Serviço [Hansen], podendo ser desde um Desktop ou Servidor, até um Celular ou Palmtop. Temos o servidor de aplicação, que provê o serviço e se comunica com o Requerente através da camada e pacotes SOAP. E temos o Registro do WS codificado em WSDL, publicado e descoberto através da especificação UDDI [Hansen].

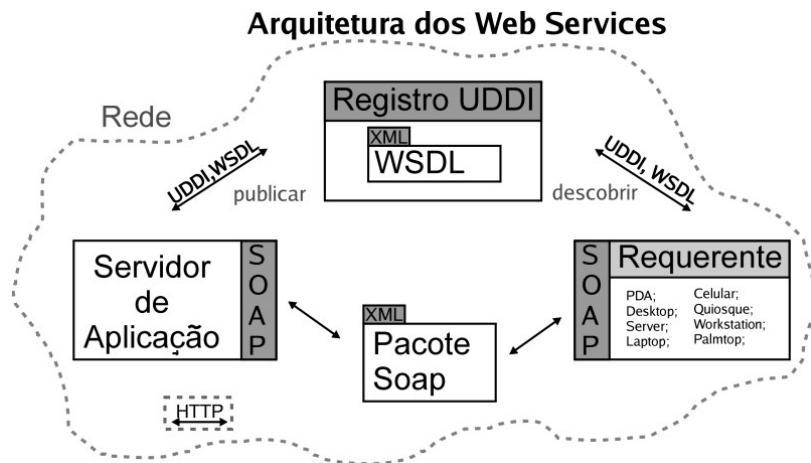


Figure 2. Web Services Architecture

A aplicação cliente envia para aplicação servidora um pacote XML através do protocolo SOAP. A aplicação servidora processa esta requisição, de acordo com suas regras internas de negócio e retorna ao cliente a resposta também através de um pacote XML pelo protocolo SOAP. [Chavda]

Tendo em vista este panorama, fica claro que a utilização de padrões abertos (SOAP é mantido pelo W3C) e vastamente conhecidos para implementação/distribuição de WebServices são fatores preponderantes que acabam por levar à sua adoção.

HTTP (Hypertext Transfer Protocol) é o protocolo padrão usado sobre a porta 80 que é responsável pela requisição e transmissão de dados sobre a internet. XML Extensible Markup Language é uma linguagem de marcação utilizada para descrever a informação nela contida. Ambos são padrões utilizados mundialmente.

No meio de todo o processo, provendo a comunicação entre as aplicações, está o protocolo SOAP (*Simple Object Access Protocol*). SOAP é um protocolo herdeiro do padrão XML que encapsula um conjunto de regras para descrição de dados e processos, através de um mecanismo simples para definir a semântica de uma aplicação através de um modelo de empacotamento e um mecanismo de codificação [Chavda]. É projetado para a troca de informações em um ambiente descentralizado [Hansen] através do protocolo de comunicação HTTP e do formato XML [Vaughan]. Dessa forma, para uma aplicação trabalhar com WS, basta a compatibilidade com SOAP, tanto no lado do cliente (criando o documento XML com a informação necessária para invocar o serviço) quanto no lado do servidor (responsável por executar a mensagem como um interpretador) [Hansen].

SOAP é peça central de um Web Service porque prove um mecanismo leve para troca de informação estruturada entre pares, em um ambiente descentralizado e distribuído, usando XML. SOAP em si, não define o modelo de programação. Define sim, um mecanismo simples para "expressar" a semântica da aplicação através de um modelo de pacotes [Chavda].

A mensagem SOAP consiste de quatro elementos (envelope, cabeçalho, corpo e exceção). O envelope é o elemento raiz da mensagem que descreve o que está na mensagem; Ele identifica a mensagem SOAP. O envelope também contém um cabeçalho opcional que contém a informações específicas da mensagem. O corpo do envelope contém os dados direcionados para o recipiente (destino) da mensagem. O corpo também pode conter a exceção que é usada para carregar mensagens de erro. É uma forma padrão de comunicar mensagens de erro de volta para o cliente.

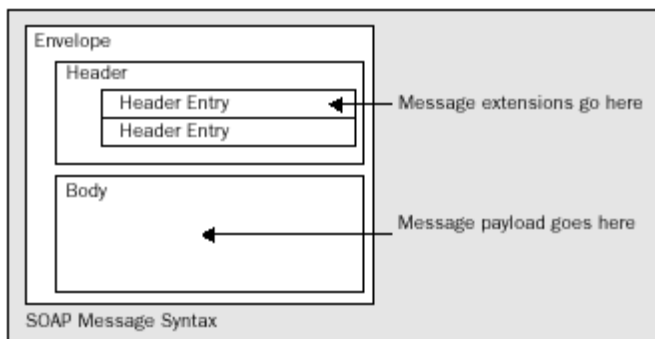


Figure 3. Envelope SOAP

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/
  encoding/">
  <soap:Header>
    <h:from xmlns:h="http://www.wrox.com/Header">SoapGuy@wrox.com</h:from>
  </soap:Header>
  <soap:Body>
    <w:GetSecretIdentity xmlns:w="http://www.wrox.com/heroes/">
      <w:codename>XSLT-Man</w:codename>
    </w:GetSecretIdentity>
  </soap:Body>
</soap:Envelope>
```

Para descrever os serviços, é utilizada a linguagem WSDL (*Web Services Description Language*), baseada no formato XML que descreve um WS [Vaughan] através da definição das interfaces e os mecanismos de interação, contendo informações como o protocolo, formato de dados, segurança, dentre outros [Hansen]. O WSDL descreve um conjunto de mensagens SOAP e fornece informações necessária para os clientes interagirem com ele [Chavda]. WSDL especifica a localização do WebService, as operações que estão disponíveis, os tipos de dados intercambiados e os protocolos de comunicação que serão utilizados.

Um Web Service se torna disponível quando colocado em um diretório central chamado UDDI (*Universal Description, Discovery and Integration*). O UDDI é um diretório central onde os serviços podem ser publicados, registrados e pesquisados por Web Services. Os dados armazenados no diretório UDDI estão no formato XML. Os dados capturados dentro do UDDI estão divididos em três categorias: páginas brancas, amarelas e verdes. As páginas brancas contêm as

informações gerais como nome, descrição, endereço sobre a companhia que oferece o serviço. As páginas amarelas contêm a classificação geral dos dados em categorias da indústria, baseadas em padrões taxonômicos para cada tipo de serviço oferecido. As páginas verdes contêm informações técnicas detalhadas sobre o Web Service, permitindo outra pessoa escrever uma aplicação utilizando este Web Service [Chavda]. Os Web Services registrados em um UDDI que podem estar inclusive em servidores diferentes [Fox].

Utilizando estes padrões abertos, os desenvolvedores podem criar componentes abertos, que podem ser acessados de qualquer plataforma ou linguagem de programação capaz de se comunicar com os conhecidos protocolos da internet [Chavda].

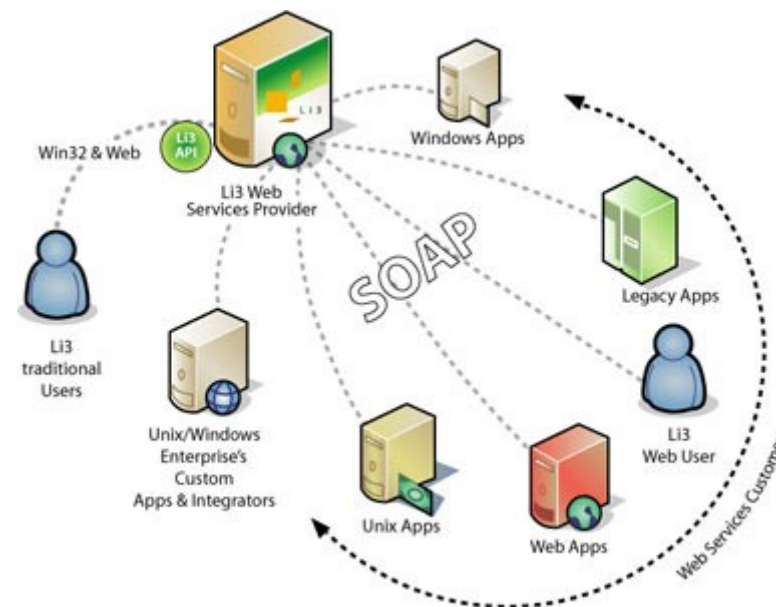


Figure 4. Diferentes plataformas utilizando Web Services

### 3. Vantagens

Ao longo dos anos, a promessa de interoperabilidade tem sido ilusória. Presenciamos o surgimento de padrões como DCOM e CORBA [Chung] e mais recentemente o de Web Services. A promessa, é de que Web Services (WS) integrarão PC's, dispositivos, bancos de dados, redes, em uma única plataforma virtual, a internet, movendo funcionalidades desde aplicações e dados do Desktop para servidores Web, que passarão a controlar segurança, privacidade e acessibilidade, lidando com aplicações complexas e transações críticas [Vaughan]. Mas desenvolver WS para automatizar as aplicações ainda representa um desafio [Chung].

É importante analisar tecnologias antecessoras aos WebServices (EDI, CORBA, DCOM, RPC e RMI), para concluir se WebServices são realmente menos complexos e de fácil manutenção em relação a estes.

De acordo com [Chavda], EDI (*Electronic Data Interchange*) é de difícil implementação, além de complexo e caro; CORBA (*Common Object Request Broker Architecture*) e DCOM (*Distributed Component Object Model*), além de competirem entre si, eram difíceis de entender, tendo pouco suporte das companhias de software; RPC (*Remote Procedure Call*) ficou muito restrito ao uso dentro do universo Unix e Java RMI (*Remote Procedure Call*) possui suporte de poucos fornecedores.

Como a implementação de WS utiliza padrões abertos como XML, para percorrer os resultados, pode-se utilizar técnicas consagradas como SAX (*Simple API for XML*), realizando o parser do XML por partes, ou mesmo DOM (*Document Object Model*), percorrendo o XML de forma hierárquica, possibilitando o controle do XML e mesmo a conversão para outros formatos [Hansen]. Também advindo do fato da utilização de padrões abertos, o fato de não existir descrição explícita em WS de uma forma de prover segurança na transmissão dos pacotes, poderia ser considerada uma desvantagem de utilização, mas a mesma pode ser provida por padrões abertos e consagrados como chaves públicas e privadas (SSL) [Vaughan].

Pode-se construir novas aplicações, provendo interoperabilidade, utilizando-se de componentes de aplicações legadas entregues via WS [Kleijnen]. O futuro está direcionado em tornar públicos os serviços de componentes na organização e de como estes serviços irão se intercomunicar (tecnologicamente e politicamente) [Arsanjani].

### 4. Mudanças

Para a adoção de WS nas organizações, são necessárias estruturas que suportem este novo paradigma, uma vez que WS proporcionam sistemas de diferentes plataformas interagindo via XML. Também é necessário criar uma cultura organizacional orientada à construção de aplicações fortemente baseadas em componentes em grande escala e os disponibilizar como WS, além de primar constantemente pela separação dos aspectos, externalizando as partes variáveis dos problemas [Arsanjani].

No tocante à logística da engenharia de software, a utilização de WS, permite que o processo de concepção do software seja geograficamente distribuído e composto por partes de diferentes fornecedores. Além disso, transforma vários aspectos de implementação, como processos em lote em elegantes WS [Arsanjani].

Mais do que tudo isso, surge o conceito da empresa provedora de WS pela Web, através da venda para seus assinantes [Vaughan]. Assim, seria possível desenvolver a aplicação através de WS que são colados somente no momento da execução [Chung]. Dessa forma, a utilização de WS quebra a tradicional separação entre local e global no desenvolvimento de aplicações [Chung], trazendo à tona aspectos gerenciais como a necessidade de confiança entre as organizações [Arsanjani]. Também é necessário se preocupar com várias esferas para o sucesso da adoção, desde a organizacional (gerência de projeto), a metodologia (suportar o desenvolvimento baseado em componentes), a arquitetura (escalabilidade), tecnologia (mapear projeto para tecnologia) e infraestrutura (ferramentas de desenvolvimento e distribuição) [Arsanjani].

### 5. Ferramentas

Em relação à engenharia de software, para o sucesso na adoção em WS, deve-se trazer para etapas iniciais do projeto, preocupações como coordenação e semântica [Chung]. Para que tudo funcione é necessário simular o mais próximo do real desde o início, preocupando-se com aspectos de rede e de segurança [Arsanjani]. Em relação aos aspectos gerenciais, a idéia de posse de dados e processos ficará mais acentuada (quem é responsável pelas partes e pelo todo) em um ambiente distribuído e de reuso. Também se dará maior ênfase na ligação dos componentes neste processo de cadeia e em como provêr maior velocidade e menor custo, através de um processo dinâmico [Arsanjani]. WS proporcionarão uma gama de composições de novos WS, muitas vezes de fornecedores diferentes, o que faz com que os projetistas gastem mais tempo

integrando os WS do que construindo novos. Uma vez que os WS podem estar sob o domínio de terceiros, não há como utilizar os métodos de engenharia de software tradicionais para compor novos WS, pois estas metodologias são baseadas em controles centralizados e estáticos [Arsanjani].

A utilização de frameworks como .NET, J2EE, dotGNU, Mono, dentre outros, traz vantagens, principalmente em relação à funcionalidades de privacidade, segurança, conexão, localização, sincronização e o conceito de passaporte para utilização dos WS, providas através de camadas e de tecnologias como LDAP, RDF, UDDI, DHCP, IMAP, POP, JDBC, ODBC, FTP, SMS, NFS, Bind, PPP, dentre outros [Kleijnen]. Sua utilização também ajuda a abstrair o processo de criação, integração e descoberta dos WS.

Comparado com as alternativas anteriores, Web Services representam um método mais simples de se atingir os mesmos resultados. Através de ambientes de desenvolvimento integrados (plataformas como Java, .NET, PHP), a criação de um Web Service pode se tornar ainda mais simples pela geração automática de código e arquivos necessários. Ainda mais, desenvolvendo padrões para descoberta, descrição e comunicação de mensagens.

Em um dos frameworks mais utilizados que provêm suporte a Web Services, o .NET, a utilização de XML, proporciona que mesmo utilizando linguagens diferentes como Cobol ou C#, o WS seja independente de plataforma [Vaughan]. Outro framework muito utilizado, a plataforma SunOne também utiliza os padrões XML, SOAP, WSDL e UDDI onde os programas são desenvolvidos em Java e entregues por servidores J2EE [Vaughan].

Para demonstrar uma pequena aplicação utilizando Web Services, utilizaremos a linguagem PHP. O PHP é usado principalmente como uma linguagem de script server-side embutida em código HTML, sendo uma das linguagens para o desenvolvimento de aplicações Web mais utilizada no mundo. A popularidade do PHP se deve à sua facilidade para criar aplicações dinâmicas para a Web, com suporte à maioria dos gerenciadores de bancos de dados existentes, além de interoperabilidade provida por um grande conjunto de funções e extensões [Dall'Oglio].

Uma das vantagens em se utilizar PHP no ambiente Web são suas capacidades de Orientação a Objetos, classes para parser (SAX e DomXML) e geração de XML, extensões para XML-RPC e SOAP, além de bibliotecas para comunicação utilizando protocolos diversos como HTTPS, LDAP, dentre outros. Além de sua sintaxe ser muito similar a de linguagens como Java e C e o fato de se conseguir implementar Web Services em poucas linhas de código [Alaya].

Para demonstrar o funcionamento de WS em PHP, utilizaremos as classes nativas para manipulação do protocolo SOAP, desenvolvido por Dmitry Stogov. Através da utilização de SOAP em PHP, poderíamos escrever um código em PHP que envia uma pesquisa (encapsulada em XML pelo SOAP) para uma aplicação de Banco de Dados em C++ localizada em outro continente para obter o preço de um livro, por exemplo [Apple].

O Funcionamento é simples, a transação inicia quando a aplicação cliente realiza a chamada remota de uma função enviando um pacote SOAP contendo a descrição do método a ser invocado via HTTP. O Servidor recebe o pacote SOAP, interpreta-o, executa a função correspondente e retorna a resposta da execução também encapsulada via SOAP [Apple]. Para demonstração, construiremos um exemplo de WebServices em PHP para simular a requisição de dados de um cliente. Sendo que os dados estão armazenados em um Banco de dados PostgreSQL ao lado do servidor.

Para construir este WebService em PHP, o primeiro passo é criar o arquivo WSDL (Web Services Description Language), que irá descrever o funcionamento do Web Service. A seção "message" descreve processos de requisição (request) e resposta (response) de uma determinada funcionalidade do WebService, que neste caso é a função getNome. Cada processo possui seus parâmetros, bem como os tipos de dados. A seção "portType" define uma funcionalidade do WebService e indica quais processos de requisição e resposta serão utilizados para tal. Neste caso, getNomeRequest para input e getNomeResponse para output. A seção binding define como as mensagens devem ser transmitidas e codificadas. Neste caso, via RPC usando SOAP sobre HTTP. E por fim, a seção "service" define a URL na qual o serviço está rodando. Neste caso, no servidor Apache local.

#### Listagem: exemplo.wsdl

```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<definitions name = 'Exemplo'
  targetNamespace = 'http://example.org/Exemplo'
  xmlns:tns = 'http://example.org/Exemplo'
  xmlns:soap = 'http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:xsd = 'http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc = 'http://schemas.xmlsoap.org/soap/encoding/'
  xmlns:wSDL = 'http://schemas.xmlsoap.org/wsdl/'
  xmlns = 'http://schemas.xmlsoap.org/wsdl/'>
<message name = 'getNomeRequest'>
```

```

    <part name='codigo' type='xsd:string' />
</message>
<message name='getNomeResponse'>
    <part name='resultado' type='xsd:string[]' />
</message>

<portType name='ExemploPortType'>
    <operation name='getNome'>
        <input message='tns:getNomeRequest' />
        <output message='tns:getNomeResponse' />
    </operation>
</portType>

<binding name='ExemploBinding' type='tns:ExemploPortType'>
    <soap:binding style='rpc'
        transport='http://schemas.xmlsoap.org/soap/http' />
    <operation name='getNome'>
        <soap:operation soapAction='exemplo#getNome' />
        <input>
            <soap:body use='encoded' namespace='exemplo'
                encodingStyle='http://schemas.xmlsoap.org/soap/encoding' />
        >
            </input>
        <output>
            <soap:body use='encoded' namespace='exemplo'
                encodingStyle='http://schemas.xmlsoap.org/soap/encoding' />
        >
            </output>
        </operation>
    </binding>

<service name='ExemploService'>
    <port name='ExemploPort' binding='ExemploBinding'>
        <soap:address location='http://127.0.0.1/servidor.php' />
    </port>
</service>
</definitions>

```

O Programa servidor, recebe o código de um cliente, consulta seus dados no Banco de dados e retorna todas as informações em forma de vetor. Caso contrário, retorna os devidos erros, comentados mais adiante.

#### Listagem: servidor.php

```

<?php
function getNome($codigo)
{
    // verifica a passagem do parâmetro
    if (!$codigo)
    {
        throw new SoapFault('Client', 'Parametro nao preenchido');
    }

    // conecta ao Banco de Dados
    $id = @pg_connect("dbname=samples user=postgres");

    if (!$id)
        throw new SoapFault("Server", "Conexao nao
estabelecida");

    // realiza consulta ao Banco de Dados
    $result = pg_query($id, "select * from clientes " .
        "where codigo=$codigo");

    $matriz = pg_fetch_all($result);
    if ($matriz == null)
        throw new SoapFault("Server", "Cliente nao encontrado");

    // retorna os dados.
    return $matriz[0];
}

// instancia servidor SOAP
$server = new SoapServer("exemplo.wsdl",
array('encoding'=>'ISO-8859-1'));
$server->addFunction("getNome");
$server->handle();
?>

```

A seguir, temos o código da aplicação cliente, que estabelece conexão com o servidor da aplicação (servidor.php, listado acima), informa os parâmetros da função remota e exibe os dados de retorno em formato HTML (no Browser).

#### Listagem: cliente.php

```
<?php
// instancia cliente SOAP
$client = new SoapClient("exemplo.wsdl",
array('encoding'=>'ISO-8859-1'));
try
{
    // realiza chamada remota de método
    $retorno = $client->getNome(3);

    // Imprime os dados de retorno
    echo '<table border=1>';
    echo '<tr bgcolor=gray><td>Coluna </td> <td> conteúdo
</td></tr>';
    echo '<tr><td>Código </td> <td>' . $retorno['codigo'] .
'</td></tr>';
    echo '<tr><td>Nome </td> <td>' . $retorno['nome'] .
'</td></tr>';
    echo '<tr><td>Telefone </td> <td>' . $retorno['telefone'] .
'</td></tr>';
    echo '<tr><td>Rua </td> <td>' . $retorno['rua'] .
'</td></tr>';
    echo '<tr><td>Idade </td> <td>' . $retorno['idade'] .
'</td></tr>';
    echo '</table>';
}
catch (SoapFault $excecao) // ocorrência de erro
{
    echo "Erro: ";
    echo "<b> {$excecao->faultstring} </b>";
}
?>
```

Caso não ocorram erros, será exibido no Browser, os dados requisitados:

Coluna	Conteúdo
Código	3
Nome	Pablo Dall'Oglio
Telefone	55 1234-5678
Rua	Rua Conceicao
Idade	24

Caso o parâmetro do método remoto não seja informato, a mensagem de erro retornada será a seguinte.

**Erro:** Parametro nao preenchido

Caso o Banco de Dados não seja conectado devidamente, a mensagem de erro retornada será a seguinte:

**Erro:** Conexao nao estabelecida

## 6. Conclusão

Percebemos novos perfis de empresas e profissionais no mercado. Vislumbramos o surgimento de organizações especializadas em desenvolver WS para um determinado segmento de mercado, podendo desenvolver um serviço de qualidade e específico. As empresas não mais precisarão se preocupar com todo o processo vertical de concepção de aplicações, e sim em estabelecer parcerias a fim de encontrar os melhores WS para comporem sua aplicação. Mudam as relações, mudam as culturas organizacionais e muda o processo de concepção de software. Novas oportunidades surgem, lógicas de negócio de aplicações legadas são preservadas e integradas à novos ambientes, à novas plataformas de desenvolvimento. Os negócios das organizações passam a ser melhor compreendidos através da separação lógica dos problemas, e ainda mais, quando da utilização de técnicas como a de ontologias, combinadas com orientação a objetos para descrever os conceitos e as regras lógicas advindas dos seus relacionamentos.

## References

Vaughan-Nichols, S. "Web Services: Beyond the Hype".

Chung, J., Lin, K., Mathieu, R. "Web Services Computing: Advancing Software Interoperability".

Kleijnen, S., Raju, S. "An Open Web Services Architecture".

Hansen, R., Santos, C., Crespo, S., Lanius, G., Massen, F.. "Web Services: An Architecture Overview".

Arsanjani, A., Hailpern, B., Martin, J., Tarr, P. "Web Services Promises and Compromises".

Fox, G., Pierce, M., Youn, C., Mueller, K., Mock, S., Balsoy, O. "Interoperable Web Services for Computational Portals"

Alaya, D., Browne, C., Chopra, V., Sarang, P., Apshankar, K., McAllister, T. "Professional Open Source Web Services". Wrox.

Dall'Oglio, P. "PHP-GTK, Criando Aplicações Gráficas com PHP". Editora Novatec, 2004.

Apple Corporation. "Using SOAP with PHP".  
<http://developer.apple.com/internet/webservices/soapphp.html>

Campbel, S. "Web Services with NuSOAP",  
<http://www.zend.com/zend/tut/tutorial-campbell.php>, September, 2002.

### Sobre o Autor

**Nome** Pablo Dall'Oglio

**E-mail** pablo@daloglio.net

Pablo Dall'Oglio é bacharel em Informática pela UNISINOS. Autor dos projetos em software livre Agata Report e Tulip, além dos livros "PHP Programando com Orientação a Objetos" e "PHP-GTK Criando Aplicações Gráficas com PHP", pela editora Novatec. Mantenedor da comunidade brasileira de PHP-GTK. Atualmente, é diretor de tecnologia e proprietário da Adianti Solutions ([www.adianti.com.br](http://www.adianti.com.br)).