Tematy: Oprogramowanie | Tworzymy usługę sieciową | Tworzymy aplikację kliencką | Podsumowanie

Tematem tego rozdziału są usługi sieciowe XML, a dokładniej ich implementacja w PHP. Ponieważ PHP jest językiem i środowiskiem do tworzenia aplikacji WWW przeznaczonych do realizacji różnych zadań, również do tworzenia poważnych aplikacji biznesowych, nie może w nim zabraknąć możliwości tworzenia usług sieciowych oraz oprogramowania klienckiego.

Czym są usługi sieciowe? Otóż usługi sieciowe XML to komponenty programowe świadczące określone usługi i wykonujące określone zadania przez sieć (najczęściej internet), do wymiany informacji wykorzystujące język XML. Korzystanie z usług sieciowych z założenia przypomina wywoływanie funkcji — musimy znać jej nazwę, przyjmowane parametry i typ (rodzaj) wartości zwracanej — tyle że odbywa się to za pośrednictwem sieci.

Usługi sieciowe z założenia są technologią ponadplatformową. Aby uniknąć problemów związanych z poznawaniem przez programistów nowych protokołów wymiany danych, usługi sieciowe wykorzystują istniejące już wysokopoziomowe protokoły — najczęściej jest to HTTP. Zaletą wykorzystania protokołu HTTP jest niewątpliwie jego szeroka znajomość przez twórców oprogramowania oraz to, że dane przesyłane przez HTTP zwykle nie są blokowane przez zapory sieciowe. Usługi sieciowe i korzystające z nich programy (ewentualnie inne usługi, które z nich korzystają) do wymiany danych używają tzw. kopert SOAP (ang. Simple Object Access Protocol — prosty protokół dostępu do obiektów) zapisanych w języku HTML. Do opisu usług sieciowych (dostępnego również przez sieć) służy język WSDL (ang. Web Services Description Language — język opisu usług sieciowych), również oparty na języku XML. Opis usługi jest dostępny przez sieć i służy programom klienckim do rozpoznania usługi i poprawnego jej wywołania — opis taki zawiera nazwy, listę parametrów oraz rodzaj wartości zwracanych przez usługi. Programy klienckie mogą taki opis analizować na bieżąco, ale najczęściej (szczególnie w rozbudowanych środowiskach programistycznych) na podstawie opisu usługi generowane są tzw. stuby, czyli klasy i funkcje umożliwiające bardzo łatwy dostęp do metod usługi sieciowej (wywołanie takiej metody niczym nie różni się od wywołania zwykłej funkcji czy metody danego obiektu). Na koniec, usługa sieciowa może być umieszczona w rejestrze usług — spowoduje to możliwość łatwego wyszukania usług realizujących jakieś konkretne zadanie przez program kliencki lub przez twórców oprogramowania.

Usługi sieciowe XML najczęściej realizowane są jako aktywne strony WWW, uruchamiane na popularnych serwerach WWW działających na różnych platformach systemowych. Najpopularniejszymi platformami językowymi i uruchomieniowymi używanymi do realizacji usług sieciowych XML są obecnie J2EE i .NET. PHP raczej nie słynie w świecie IT jako platforma dla usług sieciowych, ale nie oznacza to bynajmniej, że usług takich nie da się w PHP implementować.

Ponieważ usługi sieciowe XML używają uznanych standardów w wymianie informacji, są wygodne do tworzenia i używania w programach, stanowią duży krok w rozwoju systemów rozproszonych, gdzie określone zadanie może być realizowane przez różne, niemające ze sobą nic wspólnego (szczególnie jeżeli chodzi o stronę software'ową) hosty.

Rozdział ten przedstawia jeden ze sposobów implementacji usług sieciowych XML i aplikacji klienckich w PHP.

Oprogramowanie

Istnieje kilka mniej lub bardziej udanych implementacji obsługi protokołu SOAP w PHP. Na potrzeby tego kursu wybrany został NuSOAP — pakiet do tworzenia usług sieciowych w PHP i korzystania z nich. Oprogramowanie to jest rozwijane na licencji LGPL i dostępne na stronie http://sourceforge.net/projects/nusoap/.

Instalacja NuSOAP jest prosta — wystarczy pobrać plik (archiwum ZIP) zawierający aktualną wersję stabilną (w momencie redagowania tego kursu była to wersja 0.7.3) i wypakować jego zawartość do dowolnego folderu w folderze głównym aplikacji, która będzie świadczyć usługi sieciowe lub z nich korzystać, albo — jeszcze lepiej —

do dowolnego folderu zdefiniowanego za pomocą dyrektywy include_path pliku konfiguracyjnego php.ini. Jak zapewniają autorzy NuSOAP (i co jest zgodne z prawdą), nie są potrzebne żadne dodatki do PHP, aby móc w pełni korzystać z tego pakietu.

Tworzymy usługę sieciową

Nasza przykładowa usługa sieciowa będzie nosiła nazwę „Usługa testowa" i będzie zawierała tylko jedną metodę (funkcję), zwracającą aktualną datę serwera. Data będzie zwracana w formacie długim (np. 20 lipca 2010 r.) lub krótkim (np. 2010-07-20) w zależności od tego, czy parametr przekazywany metodzie będzie miał wartość true, czy false. Na początek musimy napisać funkcję, która będzie zwracała datę jak opisano wyżej, ale która nie będzie miała nic wspólnego z usługą sieciową. Funkcja ta może wyglądać np. tak:

```
function podajDlugaDate($typ)
{
  //  $miesiace to tablica asocjacyjna zdefiniowana wcześniej,
  //tłumacząca  numery miesięcy na ich nazwy
  global $miesiace;

  if ($typ == false) {
    $data = date('Y-m-d');
  } else {
    $data = date('d').&amp;quot;  &amp;quot;.$miesiace[date('m')].&amp;quot; &amp;quot;.date('Y').&amp;quot; r.&amp;quot;;
  }
  return $data;
}
```

W funkcji tej umieszczone zostało odwołanie do zadeklarowanej wcześniej (w tym samym pliku co funkcja, tylko „ponad" nią — czyli globalnie) tablicy asocjacyjnej $miesiace. Tablica ta jest indeksowana numerami miesięcy (w postaci dwucyfrowych liczb traktowanych jako ciąg znaków) i zawiera odpowiadające im nazwy miesięcy.

Teraz tę funkcję przerobimy na metodę usługi sieciowej. W tym celu w udostępnianym przez serwer katalogu utworzymy plik o nazwie dateserv.php i wprowadzimy do niego zawartość listingu 11.1.

Listing 11.1. Definicja usługi sieciowej

```
<?php
 //  Dołączenie pakietu NuSOAP
 require_once(&amp;quot;soap/lib/nusoap.php&amp;quot;);
 // Tablica asocjacyjna do konwersji numerów miesięcy na nazwy
 $miesiace = array('01' => 'stycznia', '02' =>  'lutego', '03' => 'marca',
```

```
                '04' => 'kwietnia', '05' => 'maja',  '06' => 'czerwca',
                '07' => 'lipca', '08' => 'sierpnia',  '09' => 'września',
                '10' => 'października', '11' =>  'listopada', '12' => 'grudnia');

// Utworzenie usługi sieciowej (serwera SOAP)
$ns = &quot;http://127.0.0.1&quot;;
$serwer = new soap_server();
$serwer->configureWSDL('Usługa testowa',  $ns);
$serwer->wsdl->schemaTargetNamespace  = $ns;

// Rejestracja metod usługi sieciowej
$serwer->register('podajDlugaDate',
                array('typ' =>  'xsd:boolean'),
                array('return' =>  'xsd:string'),
                $ns,
                false,
                false,
                false,
                'Zwraca aktualną datę serwera.');

// Definicja metody usługi sieciowej
function podajDlugaDate($typ) {
  global $miesiace;

  if ($typ == false) {
    $data = date('Y-m-d');
  } else {
    $data = date('d').&quot;  &quot;.$miesiace[date('m')].&quot; &quot;.date('Y').&quot; r.&quot;;
```

```
  }

  return new soapval('return',  'xsd:string', $data);

 }



 $HTTP_RAW_POST_DATA =  isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';

 $serwer->service($HTTP_RAW_POST_DATA);

?>
```

Występująca na samym początku kodu z listingu 11.1 instrukcja require_once dołącza do naszego skryptu pakiet NuSOAP — jest to absolutna konieczność, żeby można było stworzyć usługę sieciową. Oczywiście ścieżka w instrukcji require_once jest zależna od miejsca, w którym umieściliśmy naszą kopię biblioteki NuSOAP.

Dalej mamy definicję tablicy asocjacyjnej miesięcy, a następnie linie kodu konieczne do utworzenia usługi sieciowej XML (m.in. podajemy nazwę usługi). Zauważmy, że zanim funkcja podajDlugaDate() została zdefiniowana, była rejestrowana jako metoda usługi. Widzimy, że podczas rejestracji metody podajemy jej nazwę, tablicę parametrów (parametry i ich typy podajemy jako pary klucz-wartość), tablicę definiującą typ wartości zwracanej, przestrzeń nazw usługi, szereg mniej ważnych parametrów (choć w niektórych przypadkach użytecznych) i na końcu opis usługi przeznaczony dla użytkownika.

Definicja metody podajDlugaDate() w zasadzie niczym się nie różni od przedstawionego wcześniej pierwowzoru, oprócz wartości zwracanej. W przypadku metody usługi sieciowej jako wartość zwracaną musimy przekazać obiekt klasy soapval, żeby usługa mogła prawidłowo działać. Na koniec skryptu dwie linie konieczne do uruchomienia usługi.

Mając tak sformułowany skrypt, możemy teraz odwołać się do niego z przeglądarki WWW — jeżeli nie popełnimy żadnego błędu, efekt powinien być podobny do tego z rysunku 11.1.

Rysunek 11.1. Ekran naszej usługi sieciowej

Aby uzyskać opis usługi w języku WSDL, musimy w polu adresu przeglądarki wpisać:

http://localhost/soap/dateserv.php?wsdl

Oczywiście adres URL usługi uzależniony jest od konfiguracji i nazwy serwera, ważne jest, żeby po nazwie pliku dateserv.php podać parametr wsdl (w przypadku NuSOAP nazwa tego parametru pisana jest małymi literami). Rysunek 11.2 przedstawia okno przeglądarki z opisem usługi, natomiast pełny wydruk opisu usługi w języku WSDL znajduje się na listingu 11.2.

Rysunek 11.2. Okno przeglądarki z załadowanym opisem usługi sieciowej

Listing 11.2 Opis usługi sieciowej w języku WSDL

```
<?xml version=&quot;1.0&quot;  encoding=&quot;ISO-8859-1&quot;?>

<definitions  xmlns:SOAP-ENV=&quot;http://schemas.xmlsoap.org/soap/envelope/&quot;
xmlns:xsd=&quot;http://www.w3.org/2001/XMLSchema&quot;
xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
xmlns:SOAP-ENC=&quot;http://schemas.xmlsoap.org/soap/encoding/&quot;
xmlns:tns=&quot;http://127.0.0.1&quot;  xmlns:soap=&quot;http://schemas.xmlsoap.org/wsdl/soap/&quot;
xmlns:wsdl=&quot;http://schemas.xmlsoap.org/wsdl/&quot;  xmlns=&quot;http://schemas.xmlsoap.org/wsdl/&quot;
targetNamespace=&quot;http://127.0.0.1&quot;>

 <types>

  <xsd:schema  targetNamespace=&quot;http://127.0.0.1&quot;>

   <xsd:import  namespace=&quot;http://schemas.xmlsoap.org/soap/encoding/&quot; />

   <xsd:import  namespace=&quot;http://schemas.xmlsoap.org/wsdl/&quot; />

  </xsd:schema>

 </types>

 <message  name=&quot;podajDlugaDateRequest&quot;>

  <part name=&quot;typ&quot;  type=&quot;xsd:boolean&quot; />

 </message>

 <message  name=&quot;podajDlugaDateResponse&quot;>

  <part name=&quot;return&quot;  type=&quot;xsd:string&quot; />

 </message>

 <portType name=&quot;Usługa  testowaPortType&quot;>

  <operation  name=&quot;podajDlugaDate&quot;>

   <documentation>Zwraca aktualną  datę serwera.</documentation>

   <input  message=&quot;tns:podajDlugaDateRequest&quot;/>

   <output  message=&quot;tns:podajDlugaDateResponse&quot;/>

  </operation>

 </portType>

 <binding name=&quot;Usługa  testowaBinding&quot; type=&quot;tns:Usługa testowaPortType&quot;>

  <soap:binding style=&quot;rpc&quot;  transport=&quot;http://schemas.xmlsoap.org/soap/http&quot;/>

  <operation name=&quot;podajDlugaDate&quot;>

   <soap:operation  soapAction=&quot;http://192.168.20.135/~rafal/dateserv.php/podajDlugaDate&quot;
```

style="rpc"/>

<input>

<soap:body use="encoded" namespace="http://127.0.0.1"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>

</input>

<output>

<soap:body use="encoded" namespace="http://127.0.0.1"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>

</output>

</operation>

</binding>

<service name="Usługa testowa">

<port name="Usługa testowaPort" binding="tns:Usługa testowaBinding">

<soap:address location="http://192.168.20.135/~rafal/dateserv.php"/>

</port>

</service>

</definitions>

Jak widać, opis usługi w języku WSDL jest dość trudny do zrozumienia, stąd zastosowanie znajdują różne biblioteki klas lub funkcji, które potrafią „rozszyfrować” WSDL i umożliwiają nam korzystanie z usług sieciowych.

Tworzymy aplikację kliencką

Program kliencki dla usługi sieciowej tworzy się podobnie do samej usługi sieciowej. Oprogramowanie NuSOAP przetwarza opis usługi w języku WSDL na bieżąco, ale mimo to programy klienckie działają szybko i wydajnie. Listing 11.3 zawiera kod skryptu klienta usługi sieciowej zdefiniowanej w poprzednim punkcie.

Listing 11.3. Klient usługi sieciowej

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

<html lang="pl">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Usługa sieciowa</title>

```
</head>

<body>

<div>

<?php

  // Dołączenie pakietu NuSOAP

  require_once('soap/lib/nusoap.php');

  // Adres opisu usługi, z której  będziemy korzystać

  $wsdl_url = 'http://localhost/soap/dateserv.php?wsdl';

  // Utworzenie  obiektu klienta SOAP

  $klient = new nusoap_client($wsdl_url, 'wsdl');



  // W  zależności od wartości parametru &amp;quot;typ&amp;quot; przekazanego przez URL

  // wyświetlana  jest data długa, data krótka lub komunikat o błędzie

  if (isset($_GET['typ'])) {

    // Wywołanie  metody &amp;quot;podajDlugaDate&amp;quot; usługi sieciowej

    print($klient->call('podajDlugaDate',  array('typ' => $_GET['typ'])));

  } else {

    print(&amp;quot;Podaj parametr: true lub  false&amp;quot;);

  }

?>

</div>

</body>

</html>
```

Na początku skryptu z listingu 11.3 dołączany jest pakiet NuSOAP, podobnie jak to miało miejsce w skrypcie definiującym usługę sieciową. Dalej tworzony jest obiekt $klient klasy soapclient (w przypadku usługi sieciowej tworzyliśmy obiekt klasy soap_server) w oparciu o opis usługi WSDL. Skrypt z listingu 11.3 działa w taki sposób, że pobiera parametr typ przekazany przez URL i przekazuje jego wartość metodzie usługi sieciowej. Zwróćmy uwagę na wywołanie metody podajDlugaDate():

```
$klient->call('podajDlugaDate', array('typ' =>  $_GET['typ']))
```

Metoda podajDlugaDate() nie jest niestety dostępna wprost — trzeba ją wywołać, wykonując metodę call na

obiekcie klasy soapclient. Pierwszym parametrem tej metody jest nazwa metody usługi sieciowej, drugim — tablica parametrów wywołania metody usługi sieciowej. Tablica ta gromadzi parametry i ich wartości jako pary klucz-wartość (jest to tablica asocjacyjna). Wynik zwracany przez metodę podajDlugaDate() jest typu string, dlatego też wyświetlenie tego wyniku da efekt podobny do tego z rysunku 11.3 (pod warunkiem że wszystko poszło dobrze).

Rysunek 11.3. Efekt wykonania kodu klienta usługi sieciowej

Stosując ogólny schemat korzystania z usług sieciowych, będziemy w stanie połączyć się i pobrać dane z większości dostępnych w Internecie usług sieciowych XML. Jedynie w przypadku usług uruchamianych na platformie .NET musimy zwracać szczególną uwagę na tablicę parametrów wywołania metody usługi sieciowej — w tym przypadku należy zastosować tablicę zagnieżdżoną.

Podsumowanie

W tym rozdziale zaledwie zaczęliśmy tematykę usług sieciowych. Tak potężne narzędzie, jakim są właśnie usługi sieciowe XML, wszystkie swoje zalety objawia dopiero w sytuacji, gdy dla osiągnięcia założonego celu gromadzone są w oprogramowaniu klienckim dane pochodzące z wielu usług sieciowych, uruchomionych na różnych hostach w sieci, na różnych platformach systemowych i sprzętowych. Usługi sieciowe XML zwalniają programistę z tworzenia oprogramowania sieciowego z wykorzystaniem niskopoziomowego API operującego niejednokrotnie na gniazdkach i pakietach.

W Internecie można znaleźć wiele rejestrów usług sieciowych. Jednym z ciekawszych jest WebserviceX.NET dostępny pod adresem http://www.webservicex.net/. Można w nim znaleźć wiele ciekawych usług.

Jeśli chodzi o sam pakiet NuSOAP, bardzo ciekawy podręcznik, którego autorem jest Scott Nichol, dostępny jest pod adresem http://www.scottnichol.com/soap.htm.

Warto też zainteresować się ciągle rozwijanym zestawem funkcji SOAP zintegrowanym z PHP (od wersji 5.0): http://www.php.net/manual/pl/ref.soap.php. Dodatkowo obsługę SOAP przygotowano również w ramach projektu PEAR (http://pear.php.net/).