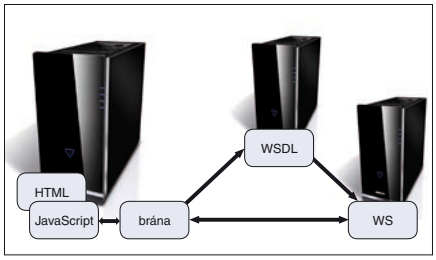
Ukážka SOA s webovou službou a využitím metódy Ajax

Pavel Horovčák

V čísle 7/2008 časopisu Automa na str. 53–55 bol publikovaný článok Architektúra SOA, Ajax a webové služby. V tomto referáte autor naväzuje na obecné princípy, ktoré boli opísané v tomto článku, a uvádza jednoduchý príklad použitia, blízky priemyselnej praxi.

1. Príklad monitorovania procesu

Pre jednoduchosť uvažujme proces s dvoma monitorovanými veličinami – teplotou a tlakom, ktoré sú v pravidelných časových intervaloch merané a spolu s hodnotou času ukladané do databázovej tabuľky, ktorá má teda tri stĺpce (čas, teplota, tlak). Čas je pre



Obr. 1. Štruktúra úlohy

zjednodušenie v podobe celého čísla (*auto_increment*). Na takto definovanom dátovom zdroji sa pokúsime jednoducho ilustrovať prístup SOA k vytváraniu služieb na báze webových služieb a ich kompozícii do výsledného riešenia v podobe klienta webovej služby, ilustrujeme využitie metódy Ajax a rovnako aj prístup SOA formou prepojenia metódy Ajax a webových služieb.

2. Štruktúra ukážkovej úlohy

Úloha pozostáva z niekoľkých spolupracujúcich častí, umiestnených na serveri klienta, na serveri webovej služby a na počítači s popisom webovej služby.

Popis webovej služby (WS) je uložený v súbore typu WSDL (*Web Service Description Language*), ktorý predstavuje spojovaciu časť medzi webovou službou (serverom) a jej klientom. Základným predpokladom úspešnej spolupráce klientskej a serverovej strany je správne zostavený súbor WSDL, ktorý je vždy umiestnený na inom serveri ako samotná webová služba (*obr. 1*). V našej ukážke je server služby umiestnený na serveri *ccdec* a súbor WSDL na serveri *lipko*. V súbore WSDL (*obr. 2*; súbor WSDL je dostupný na http://lipko.tuke.sk/~horovcak/php_ws/wsdl/temp.wsdl a obsahuje adresu URL webovej služby, čím umožní skriptu *brána* nadviazať komunikáciu so serverom webovej služby) sú štyri sekcie, ktoré špecifikujú parametre a návratové hodnoty jednotlivých funkcií servera (*<message>*), typ portu s názvami jednotlivých funkcií (*<portType>*, *<operation>*), väzby jednotlivých funkcií na typ portu (*<binding>*) a názov služby s uvedením URL serverovej strany služby (*<service>*).

Tab. 1. Význam a parametre webovej služby

Služba	rezim	od	do
Všetky hodnoty	1	*X*	*Y*
Priemer hodnôt	2	*X*	*Y*
Maximum hodnôt	3	*X*	*Y*
Všetky hodnoty tabuľky	4	–	–
Posledný riadok (aktuálne hodnoty)	5	–	–

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='SluzbaTemp'
 targetNamespace='urn:horovcak_php5_soap'
 xmlns:ph='urn:horovcak_php5_soap'
 xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
 xmlns='http://schemas.xmlsoap.org/wsdl/'>

<!-- Funkcia getTemp -->
<message name='getTemp_Ziadost'>
 <part name='Rezim' type='xsd:string'/>
 <part name='Param1' type='xsd:string'/>
 <part name='Param2' type='xsd:string'/>
</message>
<message name='getTemp_Odpoved'>
 <part name='Vysledok' type='xsd:string'/>
</message>

<portType name='TempTypPortu'>
 <operation name='getTemp'>
  <input message='ph:getTemp_Ziadost'/>
  <output message='ph:getTemp_Odpoved'/>
 </operation>
</portType>

<binding name='TempBinding' type='ph:TempTypPortu'>
 <soap:binding style='rpc'  transport='http://schemas.xmlsoap.org/soap/http'/>
 <operation name='getTemp'>
  <soap:operation soapAction='urn:horovcak_php5_soap#getTemp'/>
  <input>
   <soap:body use='encoded' namespace='urn:horovcak_php5_soap'
    encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
  </input>
  <output>
   <soap:body use='encoded' namespace='urn:horovcak_php5_soap'
    encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
  </output>
 </operation>
</binding>

<service name='SluzbaTemp'>
 <port name='TempPort' binding='TempBinding'>
  <soap:address
location='http://ccdec.tuke.sk/~horovcak/php_ws/phccdec/tempserver.php'/>
 </port>
</service>
</definitions>
```

Obr. 2. Kód súboru temp.wsdl

Serverová časť úlohy (WS) zabezpečuje poskytovanie webovej služby jednotlivým klientom.

Klientska časť je tvorená troma súbormi – skriptami. Skript *brána* zabezpečuje obojsmerné prepojenie Ajax-klienta so serverom WS a takisto obojsmernú komunikáciu so súborom JavaScript. Súbor HTML je nositeľom formulára úlohy, kde prijíma vstup používateľa a zobrazuje výsledky. Súbor JavaScript zabezpečuje prenos vstupov z HTML na skript *brána* (to sú parametre WS) a spätný prenos výsledkov WS do formulára HTML.

Webová služba ukážky (súbor *tempserver.php*) je založená na databázovej tabuľke s troma stĺpcami. Nad touto tabuľkou definujeme päť rôznych funkcií podľa *tab. 1*. Volanie služby má teda tri parametre – *rezim*, *od* a *do*. Parameter *rezim* určuje konkrétnu funkciu, parametre *od* a *do* začiatok a koniec časového intervalu, hodnoty ktorého služba vracia (všetky, maximum alebo priemer). Pre hodnoty *rezim* 4 a 5 na hodnotách *od*, *do* nezáleží, ale pri volaní musia byť zadané (napr. 0, 0).

Každá funkcia má dve zložky. Prvá zložka (metóda *getTemp()*) realizuje príslušný vý-

```
<?php     //tempserver.php
require_once ('db_ajax_config.php');     // load configuration file
// class packages functionality for the MySql application
class SluzbaTemp {
  function __construct($tb) {
  ...
  public function createName($i,$cas,$temp,$press) {     //create the XML response
  ...
  function getTemp($rezim,$zac,$kon) {  // retrieve the values from the database
  ...
  public function __destruct()  {
  ...
}
$server = new SoapServer("http://lipko.tuke.sk/~horovcak/php_ws/wsdl/temp.wsdl");
$server->setClass("SluzbaTemp",$tb);
$server->handle();
?>
```

Obr. 3. Ukážka kódu servera webovej služby

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>AJAX temp klienttempajax.php</title>
    <script type="text/javascript" src="klienttempajax.js"></script>
    <link href="styles.css" type="text/css" rel="stylesheet"/>
  </head>
  <body >
<h3>Klient AJAX temp db (Select)</h3>
        <span style="font-weight:bold;">Rezim:   </span>
        <select id="rezim" onchange="process();">
          <option value="">Zadajte pros&iacute;m</option>
          <option value="1">od,do (vsetko)</option>
          <option value="2">od,do (priemer)</option>
          <option value="3">od,do (max)</option>
          <option value="4"> (vsetko)</option>
          <option value="5">posledne (aktualne) hodnoty</option>
        </select>
    <br/>
        <span style="font-weight:bold;">Zadaj:   </span>
        <input id="param1" type="text" value="" />
        <input id="param2" type="text" value="" />
    <br/>
    <div id="myDivElement" />
  </body>
</html>
```

Obr. 4. Klient úlohy s Ajaxom – kód súboru v HTML

3. Klientska časť úlohy

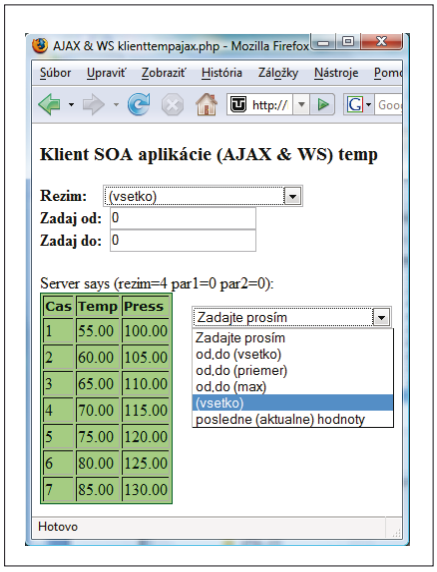
3.1 Skript v HTML

Samotná klientska časť úlohy (formulár) je vytvorená v HTML (popr. v PHP). Prijíma vstupné hodnoty od používateľa, odovzdáva ich príslušnej metóde JavaScriptu a po získaní odpovede webovej služby zobrazí správnym spôsobom odpoveď, vytvorenú v JavaScripte. Táto odpoveď sa zobrazuje vždy v konkrétnom (pomenovanom) vstupnom prvku alebo v prvku typu * * alebo **. Zdrojový kód ilustruje *obr. 4*. Formulár umožňuje používateľovi vybrať jednu z pripravených funkcií webovej služby a odoslať tri parametre (*rezim*, *param1*, *param2*) súboru *klienttempajax.js* (*obr. 6*), konkrétne funkciu *process()*. Po získaní odpovede od webovej služby a jej spracovaní je príslušný výsledok zobrazený v elemente * *, kde atribút *id* udáva jeho označenie (identifikáciu). Ukážka konkrétneho tvaru výstupu úlohy je na *obr. 5*, kde sú v jeho pravej časti znázornené aj jednotlivé voľby (element *<select>*) formulára.

3.2 JavaScript

Kód spojovacej časti v JavaScripte (súbor *klienttempajax.js*) predstavuje *obr. 6*. Funkcia *process()* je volaná z HTML pri každej zmene výberového poľa – parameter *rezim*. Prevezme zadané vstupné polia (tri parametre), vytvorí z nich dotazovací reťazec a prostredníc-

ber hodnôt z tabuľky, druhá zložka (metóda *createName()*) z týchto hodnôt vytvára zodpovedajúcu štruktúru v XML, ktorá je následne odosielaná na klienta. V tejto ukážke vracia server webovej služby pre každú službu rovnakú štruktúru v XML (t. j. so všetkými stĺpcami tabuľky). Ak to situácia vyžaduje, je možné vytvárať vhodnú štruktúru XML individuálne pre každú konkrétnu službu. Pri



Obr. 5. Výstup úlohy SOA (Ajax plus WS)

vytvorení servera služby je uprednostňovaný objektový prístup, je však možné pracovať aj v štruktúrovanom modeli – na úrovni funkcií. Pri objektovom prístupe jedinou metódou (*SoapServer->setClass()*) sú zaradené všetky metódy danej triedy do súboru funkcií servera. Pri funkčnom prístupe je potrebné zaradiť každú funkciu do súboru funkcií servera príslušnou metódou (*SoapServer->addFunction()*) zvlášť. Situáciu ilustruje úryvok zdrojového kódu servera služby na *obr. 3*. Trieda *Sluzba-Temp* okrem konštruktora a deštruktora má dve spomínané metódy pre výber údajov z databázy a pre vytvorenie štruktúry XML.

```
// holds an instance of XMLHttpRequest
var xmlHttp = createXmlHttpRequestObject();

// creates an XMLHttpRequest instance
function createXmlHttpRequestObject() {
  // will store the reference to the XMLHttpRequest
object
  var xmlHttp;
  // this should work for all browsers except IE6 and
older
  try  {
    // try to create XMLHttpRequest object
    xmlHttp = new XMLHttpRequest();
  }
  catch(e)  {
    // assume IE6 or older
    var XmlHttpVersions = new
Array('MSXML2.XMLHTTP.6.0',
                                'MSXML2.XMLHTTP.5.0',
                                'MSXML2.XMLHTTP.4.0',
                                'MSXML2.XMLHTTP.3.0',
                                'MSXML2.XMLHTTP',
                                'Microsoft.XMLHTTP');
    // try every prog id until one works
    for (var i=0; i<XmlHttpVersions.length && !xmlHttp;
i++) {
      try {
        // try to create XMLHttpRequest object
        xmlHttp = new
ActiveXObject(XmlHttpVersions[i]);
      }
      catch (e) {}
    }
  }
  // return the created object or display an error
message
  if (!xmlHttp)
    alert("Error creating the XMLHttpRequest object.");
  else
    return xmlHttp;
}
// creates the request string
function createRequestString(rezim,par1,par2) {
  var requestString = "rezim="+rezim+"&param1=" +
par1+"&param2=" + par2;
  return requestString;
}
// read answer from the server
function process() {
  var rez = document.getElementById("rezim").value;
  var par1 =
document.getElementById("param1").value;
  var par2 =
document.getElementById("param2").value;
  var url = "tempbrana.php?" +
createRequestString(rez,par1,par2);
  if(rez == "" || par1 == "" || par2 == "") {
    removeTable();
    return;
  }
  // only continue if xmlHttp isn't void
  if (xmlHttp) {
    // try to connect to the server
    try {
      // initiate reading a file from the server
      xmlHttp.open("GET", url, true);
      xmlHttp.onreadystatechange =
handleRequestStateChange;
      xmlHttp.send(null);
    }
    // display the error in case of failure
    catch (e) {
      alert("Can't connect to server:\n" + e.toString());
    }
  }
}
```

```
// function called when the state of the HTTP request
changes
function handleRequestStateChange() {
  // when readyState is 4, we are ready to read the
server response
  if (xmlHttp.readyState == 4) {
    // continue only if HTTP status is "OK"
    if (xmlHttp.status == 200) {
      try {
        // do something with the response from the
server
        handleServerResponse();
      }
      catch(e) {
        // display error message
        alert("Error reading the response: " +
e.toString());
      }
    }
    else {
      // display status message
      alert("There was a problem retrieving the data:\n"
+
            xmlHttp.statusText);
    }
  }
}


// handles the response received from the server
function handleServerResponse() {
  // read the message from the server
  var xmlResponse = xmlHttp.responseXML;
  // obtain the XML's document element
  xmlRoot = xmlResponse.documentElement;
  // obtain arrays with book parameters
  casAr = xmlRoot.getElementsByTagName("cas");
  tempAr = xmlRoot.getElementsByTagName("temp");
  pressAr =
xmlRoot.getElementsByTagName("press");
  // generate HTML output
  var html = "<table border=1  class=Table1> <tr
class=TableHead1>" +
"<td>Cas</td><td>Temp</td><td>Press</td></tr>";
  // iterate through the arrays and create an HTML
structure
  for (var i=0; i<casAr.length; i++)
    html += "<tr><td>"+casAr.item(i).firstChild.data +
"</td>"
            +"<td>"+ tempAr.item(i).firstChild.data +
"</td>"
          +"<td>" + pressAr.item(i).firstChild.data +
"</td></tr>";
  html += "</table>";
// obtain a reference to the <div> element on the page
  var rez = document.getElementById("rezim").value;
  var par1 =
document.getElementById("param1").value;
  var par2 =
document.getElementById("param2").value;
  myDiv =
document.getElementById("myDivElement");

  // display the HTML output
  myDiv.innerHTML = "<br />Server says
(rezim="+rez+" par1="+par1+" par2="+par2+"): <br />"
+ html;
}
// removes old results
function removeTable() {
  var temperature =
document.getElementById("myDivElement");
  while(temperature.childNodes.length > 0) {

temperature.removeChild(temperature.childNodes[0]);
}
}
```

Obr. 6. Kód spojovacej časti v JavaScripte – súbor klienttempajax.js

tvom vytvoreného objektu *XMLHttpRequest* nadviaže spojenie so serverovou časťou označenou ako skript *brána* (*obr. 1*). Po úspešnom získaní odpovede vytvára z prijatej štruktúry XML zodpovedajúce elementy HTML s ich obsahmi, ktoré potom vhodne publikuje do príslušného pomenovaného elementu (funkciou *innerHTML*).

Tento prenos údajov zo servera môže byť orientovaný do jedného alebo viacerých elementov HTML. Podobne jeden objekt *XMLHttpRequest* môže obstarávať rôzne režimy komunikácie, alebo každej komunikácii môže byť priradený samostatný objekt *XMLHttpRequest*, popr. môže byť využívané celé pole týchto objektov. To všetko závisí na rozložení vstupných prvkov stránky, na koncepcii ich funkcií a na koncepcii asynchrónnej komunikácie medzi klientom a serverom. V našom prípade komunikáciu zabezpečuje jeden objekt *XMLHttpRequest*, ktorý odosiela na server vždy jednu trojicu vstupných parametrov a prijíma jednu alebo viac trojíc údajov z tabuľky.

Súbor *klienttempajax.js* je tvorený skupinou funkcií, ktoré budú popísané v poradí, v akom sa nachádzajú v súbore. Funkcia *createXmlHttpRequestObject()* vytvára inštanciu objektu *XmlHttpRequest* podľa typu prehliadača, ktorá zabezpečuje komunikáciu so skriptom *brána*. Funkcia *createRequestString()* zabezpečuje jednoduché spojenie troch reťazcov – parametrov volania skriptu brána. Funkcia *process()* vykonáva volanie skriptu *brána* (súbor *tempbrana.php*) a pri zmene jeho stavu (príchode odpovede webovej služby) volá funkciu *handleRequestStateChange()*, ktorá prekontroluje úspešnosť komunikácie a v prípade jej správnosti volá funkciu *handleServerResponse()*. Táto funkcia „rozoberá“ prijatú správu vo formáte XML s využitím DOM (*Document Object Model*) a vytvára z jej častí (v našom prípade <*cas*>, <*temp*>, <*press*>) príslušnú štruktúru v HTML (v našom prípade <*table*>). Následne je vytvorený prvok HTML zapísaný do príslušnej položky formulára pomocou funkcie *innerHTML* (*myDiv.innerHTML*). Posledná funkcia *removeTable()* má za úlohu „zmazať“ zapísané hodnoty z formulára.

Je potrebné poznamenať, že klientsky skript dostáva výsledok webovej služby vždy v tvare XML štruktúry odoslanej serverom (napr. pre funkciu *posledné (aktuálne) hodnoty* má tvar <*response*><*data*><*cas*>7</*cas*><*temp*>85.00</*temp*><*press*>130.00</*press*></*data*></*response*>), pre ktorú je potrebné vhodným spôsobom zabezpečiť zodpovedajúce zobrazenie. V našej ukážke je zobrazenie riešené s využitím kaskádového štýlu (súbor *styles.css*).

```
<?php   //klienttempbrana.php
// load configuration file
require_once('error_handler.php');
header('Content-Type: text/xml');
// make sure the user's browser doesn't cache the result
header('Expires: Wed, 23 Dec 1980 00:30:00 GMT'); // time in the past
header('Last-Modified: ' . gmdate('D, d M Y H:i:s') . ' GMT');
header('Cache-Control: no-cache, must-revalidate');
header('Pragma: no-cache');
// read the action parameter
$rezim = $_GET['rezim'];
$param1 = $_GET['param1'];
$param2 = $_GET['param2'];
// we call the ws
   $my = new SoapClient
("http://lipko.tuke.sk/~horovcak/php_ws/wsdl/temp.wsdl");
   $xm=$my->getTemp($rezim,$param1,$param2);
   echo $xm;
?>
```

Obr. 7. Spojovací skript brána (klienttempbrana.php)

3.3 Spojovací skript *brána*

Spojovací skript *brána* zabezpečuje vlastné volanie webovej služby (konzumáciu) na základe znalosti jej rozhrania, ktoré poskytuje súbor WSDL služby (*obr. 7*), takže je vlastne klientom webovej služby. Výsledok volania webovej služby – funkcia *getTemp()* s príslušnými parametrami, získava skript v premennej *$xm* (vo formáte XML), ktorá je následne zapisovaná na obrazovku klienta úlohy, to znamená je vstupom súboru v JavaScripte. Pritom je dôležité správne nastavenie prehliadača (typ súboru *text/xml*, nedovoliť „kešovanie“ vzhľadom na kontinuálnu komunikáciu medzi formulárom a WS). Klientska časť webovej služby sa tak po doplnení o jednotlivé nastavenia prehliadača (pomocou hlavičiek *header*) stáva serverom JavaScriptu, ktorý zabezpečuje obojsmernú komunikáciu s využitím objektu *XmlHttpRequest*. Skript ďalej zabezpečuje preberanie vstupných parametrov (*$_GET[]*), vytvorenie komunikácie so serverom WS (premenná *$my*), volanie príslušnej funkcie WS s danými parametrami (premenná *$xm*) a jej odovzdanie na klienta Ajax (echo *$xm*).

4. Záver

V rámci zostavenej ukážky sme sa pokúsili naznačiť možnosti a výhody servisne orientovanej architektúry. Použitie Ajaxu zjednodušuje a zrýchľuje úlohu z používateľského hľadiska, pripojenie jednej alebo viacerých webových služieb do aplikácie je veľmi jednoduché. Naopak najmä prácnosť manuálneho zostavenia a ladenia súboru v JavaScripte narastá. Webové služby je možné kombinovať na úrovni úlohy, ako aj priamo na úrovni webovej služby. Ukážka zostavenej úlohy v prostredí PHP5 (technológia Ajax v spojení s WS) je dostupná na adrese: http://omega.tuke.sk/pavel.horovcak/php_ws/ajaxwsom/temp/klienttempajax.php

Úloha bola vypracovaná v rámci riešenia projektov VEGA 1/4194 /07 (L), VEGA 1/0194/08 (S) a VEGA 1/0365/08 (T).

doc. Ing. Pavel Horovčák, CSc.,
Ústav riadenia a informatizácie výrobných procesov, fakulta BERG, Technická univerzita v Košiciach
(pavel.horovcak@tuke.sk)

Tento článek, jako i předchozí s názvem Architektúra SOA, Ajax a webové služby, lektoroval Petr Klán, Ústav informatiky AV ČR, v. v. i., a Fakulta strojní, České vysoké učení technické v Praze.

krátké zprávy

▶ Těžaři ropy a plynu rozšiřují možnosti využití RFID

V podmínkách mimořádně vysokých cen surové ropy a prudkého růstu poptávky po ní po celém světě hledají těžařské firmy způsoby, jak racionalizovat svůj vnitřní chod. Jednou z vhodných technik se jeví RFID, umožňující automatizovat či zdokonalit vše – od nákupu různých položek po zvýšení bezpečnosti a spolehlivosti provozního zařízení.

V čele snah o takovou podporu průmyslu nyní stojí *Oil & Gas RFID Solution Group*, sdružení expertů v dané oblasti, akademických výzkumníků, dodavatelů techniky, představitelů předních ropných společností, organizací a firem jako Texas A&M University, University of Houston, Avery Dennison, Merlin Concepts & Technology, Shipcom Wireless, Motorola Inc atd. Tato širokospektrální skupina odborníků pomáhá prospektorům, vrtařům i producentům ropy a plynu nacházet způsoby efektivního použití techniky RFID k řešení problémů jejich denní praxe, definovat potřebné specifikace, datové modely a aplikační struktury a zavádět vyzkoušené pracovní postupy. Velká pozornost je přitom věnována krajně nepříznivým provozním podmínkám, jimž je technika RFID vystavena zejména ve vrtech a jejich blízkosti (přítomnost např. kyseliny chlorovodíkové, sirovodíku, barytu atd.). S uvedenou skupinou aktivně spolupracuje také EPC Global, mezinárodní standardizační orgán pro oblast RFID.

Cílem všech zúčastněných je nabídnout ropnému průmyslu nové systémy RFID schopné uchovávat v krajně nepříznivých podmínkách větší objemy dat, komunikují na větší vzdálenosti a přiměřeně nákladné. Ty pak otevřou nové možnosti využití techniky RFID v oblastech evidence zařízení a řízení výrobních i logistických procesů.

[ARCwire, 25. července 2008.] *(sk)*