Novinky v technologiích intranetu

Martin Molhanec

Katedra elektrotechnologie, ČVUT - Fakulta elektrotechnická, Technická 2, 166 21 PRAHA 6
molhanec@fel.cvut.cz, molhanec@technologist.com

Abstrakt

Příspěvek popisuje některé nové technologie v oblasti intranetu a internetu. Jedná se zejména o technologie firmy Microsoft (skriptlety, DHTML behavior, DSO, ASP), technologii firmy Netscape (Server Side JavaScript), technologie spojené s HTTP serverem Apache a technologie postavené na bázi jazyka Java firmy SUN.

1. Úvod

Tento příspěvek navazuje na můj loňský příspěvek, který se snažil přehlednou formou utřídit intranetové / internetové technologie, jak na straně klienta, tak na straně serveru. V tomto příspěvku se budu věnovat jednak technologiím, na které loni nezbyl čas a jednak technologiím, které během roku vznikly nebo se dále vyvíjely. Vzhledem k velkému rozsahu tématu a k relativní nedostupnosti některých informací nekladu před čtenáře pochopitelně úplný a definitivní přehled všech nových nebo známých technologií, ale pouze jejich určitý výběr.

Ve svém loňském příspěvku jsem naznačil objekt našeho zájmu, proto ho letos nebudu znovu opakovat. Zachovám však celkovou strukturu příspěvku a jeho koncepci.

2. Strana klienta

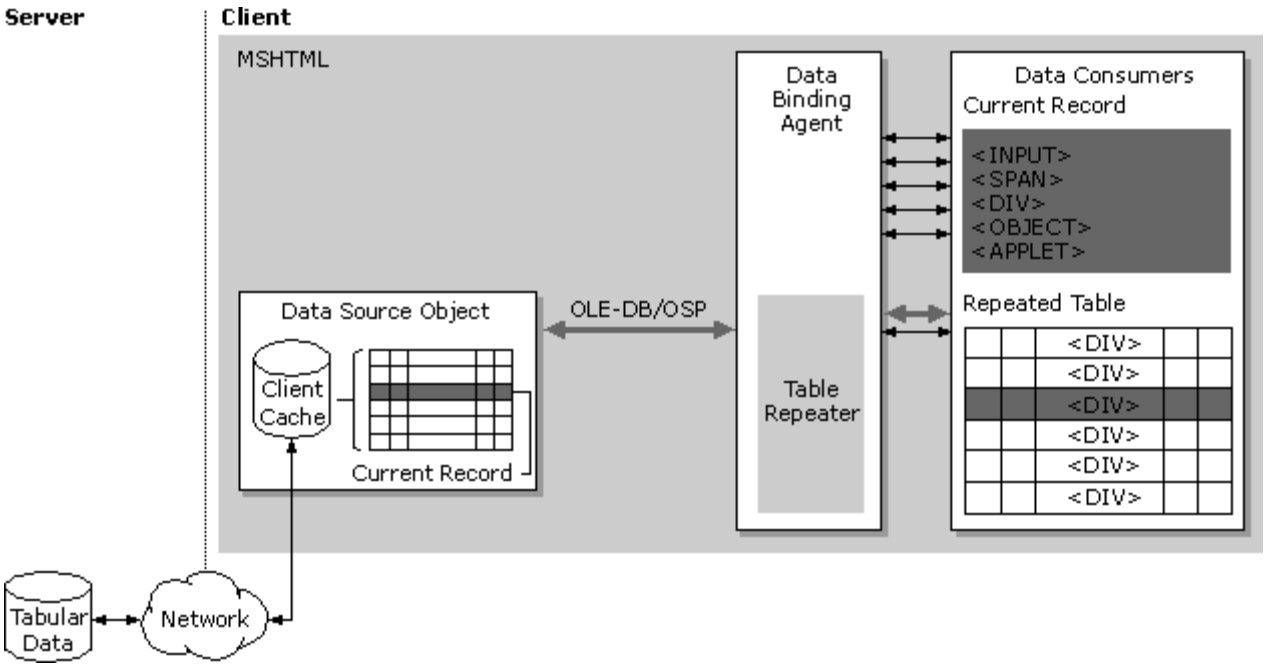
Strana klienta je především určená vlastnostmi internetového browseru. Zdá se, že v posledním roce získal nesporný náskok Internet Explorer (dále jen IE) firmy Microsoft před Netscape Navigatorem (dále jen NN) firmy Netscape. Nespornou výhodou IE 4.0 je jeho náskok před NN 4.0, který zejména v prvních verzích trpěl určitou nestabilitou, navíc IE 4.0 plně podporuje DHTML (Dynamic HTML) v plné šíři a o skutečnosti, že se IE 4.0 stal integrální součástí Windows 98 ani nemluvě!

Firma Netscape na podporu svého produktu podnikla neobvyklý krok, poskytla svůj produkt Netscape Navigator ve formě zdrojového kódu programátorům na celém světě, jedná se o verzi 5.0. Zároveň vydala svojí vlastní verzi NN 4.5. Zřejmě se v budoucnosti budou vyskytovat obě verze, jak firemní s určitou zárukou spolehlivosti, tak celosvětově uvolněná, v mnoha modifikacích, která bude v oblasti browserů prošlapávat cestu vpřed.

V současné době se na Internetu objevila finální verze IE 5.0 firmy Microsoft, která přináší uživateli některé nové vymoženosti. Microsoft se bude pochopitelně snažit, aby tato vylepšení byla přijata za platný standard a tím získat opět před svými soupeři nový náskok. Musím přiznat, že některá rozšíření firmy Microsoft jsou velice zajímavá, proto budou hlavním tématem popisu novinek v klientské části. Významnou vlastností posledních versí browserů je též podpora standardu XML.

2.1 Data Binding

Data Binding je velice užitečná technologie firmy Microsoft, která je již plně obsažena v IE 4.0. Je založena na následující úvaze. Technologie na straně serveru, například ASP, nám umožňuje spojení s databází a vygenerování tabulky v HTML, která je zaslána serverem klientu. Ale nebylo by lepší, kdyby přímo klient obsahoval elementy, které se dokáží s danou databází přímo spojit a pracovat s ní? Vždyť s takovýmito elementy by pak nebylo nutné programovat skripty na straně serveru, ale pouze navrhovat tyto elementy současně s návrhem stránky HTML, která by byla potom pouze statická? Princip *Data Binding* je naznačen na obr. 1.



Obrázek 1 Data Binding - architektura

2.1.1 Data Consumers

Data Consumers neboli spotřebitelé dat, jsou právě ty elementy HTML, které mohou být spojeny s lokální i vzdálenou databází a umí zobrazovat data v ní obsažená. V materiálech firmy Microsoft se také mluví o Data Bound Elements (DBE). Dají se rozdělit do dvou typů. Na elementy Single-valued consumers (SVC), které umějí současně zobrazit data pouze z jediného záznamu (recordu, řádky) a na elementy, které umějí zobrazit data z několika záznamů současně, Tabular data consumers (TDC), čili tabulkové elementy. Velice dobrý nápad byl ten, že do definice HTML nebyly přidány žádné nové elementy, ale stávající byly obohaceny o nové funkce! Je to možné pomocí nových atributů stávajících elementů, jak si dále ukážeme. Jedná se zejména o následující atributy.

Atribut	Význam
DATASRC	Identifikuje datový zdroj, čili tabulku
DATAFLD	Identifikuje sloupeček v tabulce
DATAFORMATAS	Určuje formát zobrazení dat
DATAPAGESIZE	Určuje kolik se zobrazí současně řádků (jen u tabelačních elementů)

Příkladem SVC může být například následující element, který zobrazí data z tabulky *dsoComposer* a položku *compsr_first* právě vybraného záznamu.

```
<SPAN DATASRC=#dsoComposer DATAFLD=compsr_first></SPAN>
```

Příkladem TDC je element *TABLE,* který zobrazí všechny položky z jednoho sloupečku jedné tabulky. Opět si uvedeme jednu ukázku.

```
<TABLE DATASRC=#dsoComposer>
        <TR><TD><DIV DATAFLD=compsr_first></DIV><TD><TR>
</TABLE>
```

Datové elementy se tedy spojují se zdrojem dat (tabulkou). Ale toto spojení není přímé! Datové elementy se spojují s *Data Source Objects* (DSO), které jsou umístěny na stránce, ve které se DBO vyskytují.

2.1.2 Data Source Objects (DSO)

DSO jsou normální ActiveX objekty nebo Applety, které jsou do stránky vloženy, ale nezobrazují se! Jejich účel je následující. DSO se umí spojit s lokální i vzdálenou databází libovolným protokolem a sám poskytuje přenášená data standardně, jako *OLE DB* nebo *OLE DB Simple Provider*. Poskytuje tedy data standardním způsobem a *Microsoft HTML* rozšiřuje objektový model DSO o standardní properties, methods a events, které jsou použitelné pro skriptovací jazyky v rámci HTML stránky.

Firma Microsoft poskytuje několik základních DSO a také poskytuje jejich úplnou specifikaci, aby si je vývojář intranetových aplikací mohl pro své potřeby sám

vyvinout.

2.1.2.1 Tabular Data Control

Tabular Data Control je jednoduchý DSO, který umí přistupovat k datové tabulce ve formátu jednoduchého textu s oddělovači. Uveďme si příklad použití tohoto elementu v rámci HTML tabulky.

Na výše uvedeném příkladu je patrné, že *Tabular Data Control* objekt má pouze

```
<OBJECT CLASSID="clsid:333C7BC4-4607-11D0-0080C7055A83"
        ID=dsoComposer WIDTH=0 HEIGHT=0>
        <PARAM NAME="DataURL" VALUE="composer.csv">
</OBJECT>
```

jeden parametr: *DataURL*, který je udává cestu k textovému souboru obsahujícímu data tabulky. Atributy objektu jsou standardní, všimněte si však, že výška i šířka objektu je rovna nule, protože samotný objekt se nikterak nezobrazuje.

2.1.2.2 Remote Data Service

Remote Data Service je DSO již poněkud složitější. Umí totiž data získat ze skutečné databáze prostřednictvím OLE-DB či ODBC. Ukažme si opět jeho jednoduché použití.

```
<OBJECT classid="clsid:BD96C5556-65A3-11D0-983A-00C04FC29E33"
        ID=dsoComposer HEIGHT=0 WIDTH=0>
        <PARAM NAME="Server" VALUE=http://musicserver>
        <PARAM NAME="Connect" VALUE="dsn=music;uid=guest;pwd=">
        <PARAM NAME="SQL" VALUE="select compsr_first from composer">
</OBJECT>
```

Jak je patrné, RDS objekt má již více parametrů. Parametr *Server* je název databázového serveru, se kterým se má spojit, parametr C*onnect* je řetězec použitý pro připojení na tento server a parametr *SQL* je řetězec obsahující SQL příkaz, který vrátí data.

2.1.2.3 JDBC DataSource Applet

JDBC DataSource Applet je na rozdíl od předchozích dvou objektů *applet*, ale jeho funkce je podobná. Pomocí datového rozhraní *JDBC* (Java Database Connection) se dokáže připojit ke vzdálené databázi a získat z ní potřebná data. Má však poněkud více parametrů.

```
<APPLET CODE=JDC.class NAME="DSA1" ID="DSA1" WIDTH=0 HEIGHT=0>
        <PARAM NAME=cabbase VALUE=
                "http://www.microsoft.com/gallery/files/datasrc/jdbcapplet/jdc.cab">
        <PARAM NAME=dbURL VALUE="jdbc:odbc:Northwind">
        <PARAM NAME=showUI VALUE=false>
        <PARAM NAME=sqlStatement VALUE="select ProductName, CompanyName,
              CategoryName from Products, Suppliers, Categories where
              Suppliers.SupplierID=Products.SupplierID and Categories.CategoryID =
              Products.CategoryID and Categories.CategoryID < 5">
        <PARAM NAME=allowInsert VALUE="true">
        <PARAM NAME=allowDelete VALUE="true">
        <PARAM NAME=allowUpdate VALUE="false">
        <PARAM NAME=user        VALUE="">
        <PARAM NAME=password    VALUE="">
</APPLET>
```

2.1.2.4 XML Data Source

Tento DSO umožňuje pracovat s daty ve formátu XML (Extensible Markup Language). Jedná se opět o applet a může sloužit k zobrazení dat, která mají hierarchickou strukturu. Způsob začlenění *XML Data Source* do stránky HTML je následující.

```
<APPLET
        CODE="com.ms.xml.dso.XMLDSO.class"
        ID="xmldso"
        WIDTH="0"
        HEIGHT="0"
        MAYSCRIPT="true">
        <PARAM NAME="URL" VALUE="composer.xml">
</APPLET>
```

2.1.2.5 MSHTML Data Source

Posledním příkladem DSO je datový zdroj přímo začleněný do stránky HTML. Ukázka takové databázové tabulky začleněné do HTML následuje.

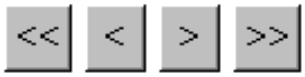
```
<H1ID=COMPSR_FIRST>Hector</H1>
<MARQUEE ID=COMPSR_LAST>Berlioz</MARQUEE>
<DIV ID=COMPSR_BIRTH>1803</DIV>
<H2 ID=COMPSR_FIRST>Modest</H2>
<H3 ID=COMPSR_LAST>Moussorgsky</H3>
<BUTTON ID=COMPSR_BIRTH>1839</BUTTON>
<TEXTAREA ID=COMPSR_FIRST>Franz</TEXTAREA>
<XMP ID=COMPSR_LAST>Liszt</XMP>
<SPAN ID=COMPSR_BIRTH>1811</SPAN>
```

Vypadá to šíleně? Ano. Význam je následující. Pokud náš DBO přistupuje k tabulce obsažené v této HTML stránce, jsou za hodnoty ve sloupečku *compsr_first* považovány všechny hodnoty elementů označených *ID=compsr_first* v pořadí od počátku HTML stránky. První hodnota (na první řádce tabulky) je tedy v našem případě *Hector*, druhá hodnota téhož sloupečku je potom *Modest*. Jak je vidět, je podstatné označení HTML elementu pomocí atributu *ID* a nikoliv typ elementu. Ukažme si nyní začlenění jednoduchého *MSHTML Data Source do stránky HTML*.

```
<OBJECT ID=htmlComposer DATA="compdata.htm" HEIGHT=0 WIDTH=0>
</OBJECT>
```

Jak je patrné, jediným významným parametrem je atribut *DATA*, který odkazuje na HTML stránku, ve které je vložená tabulka s daty.

Pochopitelně by pouhé zobrazení tabulky nebo položek z jednoho řádku DSO nebylo pro opravdovou aplikaci dostačující. Nejčastější požadavek uživatele bude, aby se mohl posouvat po jednotlivých řádkách tabulky pomocí tlačítek *na začátek, zpět, vpřed a na konec* s možným grafickým zobrazením.



Výše uvedený požadavek je možné splnit díky tomu, že DSO poskytuje standardní object *recordset* a jeho obvyklé metody a property. Ukázka kódu, který realizuje pohyb v datové tabulce pomocí čtyř tlačítek, následuje.

```
<INPUT ID=cmdNavFirst TYPE=BUTTON VALUE="<<"
        onclick="tdcComposers.recordset.MoveFirst()">
<INPUT ID=cmdNavPrev TYPE=BUTTON VALUE=" < "
        onclick="tdcComposers.recordset.MovePrevious();
        if(tdcComposers.recordset.BOF)
                tdcComposers.recordset.MoveFirst();">
<INPUT ID=cmdNavNext TYPE=BUTTON VALUE=" > "
        onclick="tdcComposers.recordset.MoveNext();
        if (tdcComposers.recordset.EOF)
                tdcComposers.recordset.MoveLast();">
<INPUT ID=cmdNavLast TYPE=BUTTON VALUE=">>"
        onclick="tdcComposers.recordset.MoveLast()">
```

2.2 Remote scripting

Jedná se opět o technologii firmy Microsoft. Jaký je její význam? Dosavadní skriptování na straně klienta neumožňovalo jednoduchým způsobem, aby skript na straně klienta mohl zavolat funkci, která se vykoná na straně serveru. Pochopitelně existuje několik možností, jak toto omezení obejít. Například prostřednictví ActiveX objektů nebo Appletů, které se již se serverem spojit mohou. Koneckonců, výše

popisovaná technologie *Data Binding* je jedním ze speciálních příkladů této možnosti.

Nicméně *Remote Scripting* umožňuje volání vzdálených procedur a funkcí přímo a jednoduše ze skriptovacího jazyka na straně klienta a vývojář se nemusí učit, jak psát ActiveX objekty nebo Applety. Komunikace mezi klientem a serverem může být následující.

- *Synchronní*, kdy klient počká na výsledek operace na serveru.
- *Asynchronní*, kdy klient na výsledek nečeká, ale může se o ukončení operace na serveru dovědět prostřednictvím události.

Pro realizaci spojení mezi stranou klienta a serveru je nutné mít nainstalovány následující soubory.

- *Rs.Htm* Obsahující metody nutné pro inicializaci spojení na straně klienta.
- *Rs.Asp* Obsahují metody potřebné na straně serveru.
- *Rsproxy.Class* Applet uskutečňující komunikaci mezi klientem a serverem.

Nyní si ukážeme stránku s inicializací vlastnosti *remote scripting* na straně klienta.

```
<HTML><HEAD><TITLE>Remote ScriptingTest</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript"
        src="../_ScriptLibrary/RS.HTM"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript">
        RSEnableRemoteScripting("../_ScriptLibrary')</SCRIPT>
</BODY>
</HTML>
```

Výše uvedená ukázka obsahuje dva klíčové řádky, které realizují *Remote Scripting* na straně klienta.

- *<SCRIPT LANGUAGE="JavaScript" src="../_ScriptLibrary/RS.HTM"></SCRIPT>*
  Tato řádka vloží do stránky klienta knihovnu, která umožní remote scripting

- *RSEnableRemoteScripting("../_ScriptLibrary')*
  Volání této funkce zajistí inicializaci Remote Scripting knihovny.

Před volání procedury nebo funkce na serveru je nutné zajistit, aby se vzdálená procedura obsažená v .ASP stránce na serveru jevila klientu jako metoda lokálního objektu. Toto zajistíme voláním knihovní funkce *RSGetASPObject* a pak již můžeme volat naší vzdálenou proceduru prostřednictvím lokálního objektu, jak je vidět na naší další ukázce.

```
RsMath = RSGetASPObject("RSMath.asp")
Result = rsMath.Add(number1, number2)
```

Variantou může být použití funkce RSExecute, aniž by bylo nutné vytvářet na straně klienta zástupný objekt.

```
Result = RSExecute("RSMath.asp","add",number1,number2)
```

A nyní obraťme ještě pozornost ke straně serveru. Stránka .ASP, která má být přístupná pro remote scripting musí obsahovat.

- *<!--#INCLUDE FILE="../_ScriptLibrary/RS.ASP"-->*
  Tato řádka přidá knihovnu pro stranu serveru do vaší .ASP stránky.

- *<% RSDispatch %>*
  Tento příkaz způsobí inicializaci knihovny na straně serveru

Nyní je ještě nutné zpřístupnit procedury a funkce ve vašem .ASP souboru pro vzdálené volání. Děje se tak vytvoření objektu *public_description,* do které se zveřejňované procedury a funkce přidají prostřednictvím jeho konstruktoru.

```
<%  RSDispatch%>
<!--#INCLUDE FILE="../ScriptLibrary/RS.ASP"-->

<SCRIPT RUNAT=SERVER LANGUAGE="JavaScript">
        var public_description = new MyServerMethods();

        function MyServerMethods(){
        this.add = AddNumbers;
        }
        function AddNumbers(num1,num2){
        return num1 + num2;
        }
</SCRIPT>
```

Jak je patrné, každý kdo umí trochu skriptovat, může vytvářet složité aplikace, aniž by se musel učit jazyk C++ a způsob tvorby DCOM. Podotýkám, že při použití VBScriptu je nutné použít ve výše uvedeném příkladu trochu odlišnou syntaxi. O využití asynchronní komunikace a registrace chybových hlášení se ve svém příspěvku zmiňovat nebudu.

2.3 Extensible Markup Language (XML)

S tímto pojmem jsme se již v tomto článku setkali. Nechci se v tomto příspěvku XML zabývat, protože je to téma na samostatný článek. Pouze si připomeňme, že XML je *markup language* podobně jako třeba HTML. Podobně jako HTML se jedná o podmnožinu *SGML (Standard Generalized Markup Language)* a stejně jako HTML je definován organizací *World Wide Web Consortium*. Na rozdíl od HTML je však možné pomocí XML specifikovat vlastní jazyky podobné například právě HTML. Firma Microsoft podporuje XML již od verze IE 4.0, ale plná podpora XML je obsažena teprve v IE 5.0. Tato verze podporuje přímo XML pomocí tzv. *XML Data*

Islands. Jedná se, jak je z anglického názvu patrné, o jakési *ostrůvky XML* přímo v HTML stránkách. Ukažme si opět jeden příklad.

```
<!--[if gte IE 5]>
<XML ID="xml1">
<topic-info>
   <page-type>reference</page-type>
   <member-type>property</member-type>
   <persistent-name>ACCESSKEY</persistent-name>
   <runtime-name readable="1" writeable="1">accessKey</runtime-name>
   <abstract>Sets or retrieves the accelerator key for the object.</abstract>
</topic-info>
</XML>
<![endif]-->
```

Ve výše uvedeném příkladu je pomocí XML definována property *accessKey*. Všimněte si použití podmínky v komentáři, která zabraňuje browserům, které XML nerozumějí, aby XML definici interpretovaly jako prostý text.

Důležitá je možnost přistupovat k takto definovaným ostrůvkům XML ze skriptovacích jazyků, například takto.

```
Function return XMLData(){
    Return document.all("xml1").XMLDocument.nodeValue;
}
```

Ostrůvky v XML je také možné vytvářet prostřednictvím SCRIPT elementu, ukážeme si ještě jeden jednoduchý příklad.

```
<SCRIPT ID="xml1" LANGUAGE="XML">
        <XMLDATA>
                <DATA>Text</DATA>
        </XMLDATA>
</SCRIPT>
```

3. Strana serveru

Jaké jsou novinky na straně serveru? Pochopitelně vedoucí roli tady hrají tři nejrozšířenější WWW servery.

- *Apache*
- *Internet Information Server* a *Personal Web Server* firmy Microsoft
- *Fast Track Server* firmy Netscape

V oblasti serverů je patrný boj mezi proprietními řešeními, které jsou typické zejména pro firmu Microsoft a řešeními obecnými nezávislými na platformě.

3.1 Obecná rozhraní

Mezi obecná rozhraní nezávislá na platformě patří především rozhraní CGI a FastCGI, o kterých jsem podrobněji pojednal ve svém příspěvku v loňském roce, proto o nich letos již hovořit nebudu. Pokud mohu usuzovat, rozhraní CGI a FastCGI jsou vesměs na ústupu. Jejich podstatnou výhodou je značná obecnost a stupeň standardizace. Nevýhodou je pak nízká efektivita při obsluze velkého počtu přístupů.

3.1.1. Servlety

Toto rozhraní, stručně zmíněné již v mém loňském příspěvku, umožňuje rozšiřovat WWW server o moduly psané v jazyce Java. V současné době je k dispozici definice *Java Servlet API 2.1*. O tom, že se nejedná o žádnou nevýznamnou hříčku programátorů, svědčí následující seznam WWW serverů, které mají toto rozhraní implementované.

Úplná implementace Java Servlet API od firmy SUN

Java Web Server *Sun WebServer*

Úplná implementace Java Servlet API od dalších firem

Acme Acme.Serve
Apache Web Server
ATG Dynamo Application Server
IBM Internet Connection Server
IBM VisualAge WebRunner Toolkit
jo!
KonaSoft Enterprise Server
Live Softeware JRun
Lotus Domino Go Webserver
Mort Bay Jetty
Novocode NetForge
Paralogic WebCore ServletFactory
Tandem iTP WebServer
vqServer
W3C Jigsaw
WebEasy WEASEL
WebLogic Tengah Application Server
Zeus Web Server

Server Add-On Engines
Tyto produkty umožňují používat servlety s libovolným WWW serverem

Gefion Software WAICoolRunner
IBM WebSphere Application Server
Live Software JRun
New Atlanta Communications ServletExec
Unicom Servlet CGI Development Kit

Nicméně i tento seznam, umístěný na stránkách firmy SUN není zřejmě úplný, protože zde chybí například *mod_jserv* pro server Apache a některé další dostupné implementace dle jiných pramenů. A ještě jednu poznámku. Modul *mod_jserv* je modul, který umožňuje, aby na serveru Apache pracovaly *Java Servlety*, které jsou na platformě nezávislé. Je nutné ho odlišit od modulu *mod_java*, který umožňuje psát moduly pro server Apache v jazyce Java a nikoliv v tradičním jazyce C.

Pro připomenutí, jak se servlety programují, uvedeme si zde jednu malou ukázku. Jedná se známy servlet, který vygeneruje HTML stránku se slovy *"Hello world!"*. Jak je patrné, nejedná se o nic komplikovaného.

```
public class HelloHttpServlet extends HttpServlet{
   public void doGet(HttpServletRequest req, HttpServletResponse res)
         throws IOException, ServletException    {
                res.setContentType("text/plain");
                ServletOutputStream out = res.getOutputStream();
                out.println("Hello, world!!!");
   }
}
```

3.1.1.1 Java Server Pages (JSP)

Programování v jazyku Java je pochopitelně poněkud obtížné. Programování v ASP nebo PHP je snazší, ale Java je velice kvalitní jazyk. Existuje nějaké řešení? Odpovědí na tuto otázku jsou *Java Server Pages (JSP)*, pro které existuje zatím verze specifikace 0.92. JSP jsou opravdu velice podobné ASP. Uveďme si malou ukázku vložení kódu Javy do HTML stránky.

```
<SCRIPT RUNAT=server>
int i = 0;
String foo = "Hello";
private void foo() {
// some code
}
</SCRIPT>
```

Druhá možnost využívá podobně jako ASP oddělovačů <% a %>.

```
<%
foo = request.getParameter("Name");
out.println(foo);
%>
```

A pochopitelně existuje i třetí možnost, opět podobná jako v ASP.

```
<p>I live at: <%= myaddress %> </p>
```

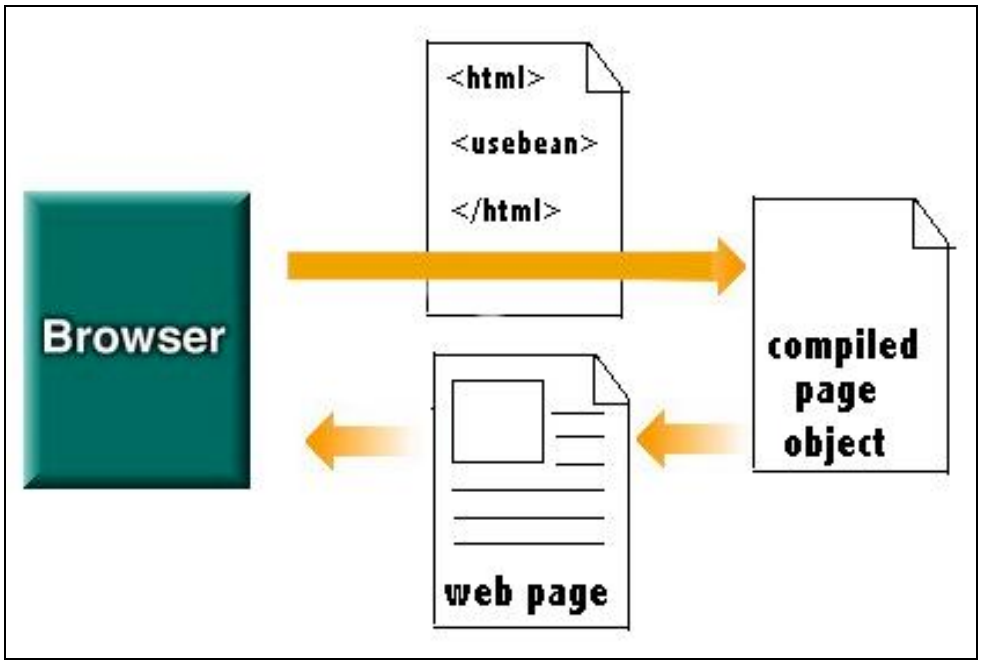
Na rozdíl od ASP existuje v JSP speciální způsob na generování tabulek. Je umožněn existencí speciálních HTML značek LOOP a DISPLAY s následující syntaxí.

```
<LOOP        PROPERTY="beanname:indexedproperty"
              PROPERTYELEMENT="x" >
<DISPLAY PROPERTY= "x"  PLACEHOLDER="nullvalue">
</LOOP>
```

Jak je patrné, JSP jsou přímo napojeny na JavaBeans, které zjednodušeně vysvětleno odpovídají ActiveX/COM objektům firmy Microsoft v ASP prostředí.

JSP technika je zatím poměrně mladá, nicméně možnost používat v HTML stránkách kvalitní programovací jazyk Java je lákavá. Navíc je možné takové stránky přímo propojovat s JavaBeans objekty na serverové straně. Takové spojení může být velice vhodným prostředím pro bezpečné průmyslové aplikace.

Zdrojový kód JSP má koncovku .JSP. Koneckonců, jak jinak ☺. Důležitá je však následující skutečnost. Stránky JSP jsou při prvním přístupu on line zkompilovány do Javovského servletu, který je nadále přítomen v paměti serveru. To znamená, že odezva při dalších přístupech na JSP stránky je již velice rychlá. Zjednodušený princip JSP je na obr. 2. Přestože se jedná o velice mladé rozhraní, existuje již i GNU implementace!



Obrázek 2 Princip JSP

3.2 Proprietní rozhraní

Jak již bylo napsáno výše, pro každý z vedoucích WWW serverů existují proprietní rozhraní, která jsou pro daný server určena, jako hlavní vývojové prostředí pro serverové aplikace. Protože o prostředí ASP bylo pohovořeno v mém loňském

příspěvku, budu se letos věnovat platformám WWW serveru Apache a WWW serveru firmy Netscape.

3.2.1 Apache modules

Proprietní prostředí serveru Apache se nazývá *Apache Modules*. Jedná se o API orientované na jazyk C. Odpovídá tomu, co je u *Internet Information Serveru* nazýváno ISAPI a u serverů firmy Netscape NSAPI. Toto rozhraní nízké úrovně není pochopitelně vhodné pro vývoj aplikací, ale pro vytvoření prostředí vyšší úrovně (podobně jako u Internet Information Serveru je ASP vytvořeno nad rozhraním ISAPI). Pro vytváření WWW aplikací jsou pro Apache k dispozici zejména následující rozhraní vyšší úrovně.

3.2.1.1 Perl

Toto rozhraní je realizováno modulem *mod_perl* a umožňuje psát pro Apache moduly v jazyce Perl. Jedná se tedy stále o nižší vrstvu rozhraní, nicméně nad touto vrstvou již existují vysokoúrovňové nadstavby.

3.2.1.1.1 Embperl

Princip tohoto rozhraní je založen na podobném principu jako například princip rozhraní ASP. Vývojář vytváří stránky, jako kombinaci HTML textu a příkazů jazyka Perl. Příkazy jazyka Perl se vkládají do HTML stránek uzavřené mezi hranaté závorky [a]. K vývoji je možné použít libovolný HTML editor a vnořené příkazy vkládat jako prostý text. Protože se jedná o modul v jazyce Perlu, je možné z *Embperlu* volat všechny další knihovny, které jsou pro Perl k dispozici, například knihovny pro práci s databázemi. *Embperl* má jednoduchou automatickou podporu pro generování tabulek. Jednoduchou ukázku HTML stránky generované z databáze si nyní ukážeme. Poznámka: $raw a $col jsou speciální proměnné, které mají význam pro generování databázové tabulky.

```
[-
# connect to database
$dbh = DBI -> connect($DSN);
# prepare the SQL statement
$sth = $dbh -> prepare("SELECT * from $table");
# execute the query
$sth -> execute;
# get the fieldnames for the heading in $head
$head = $sth -> (NAME);
# get the result in $dat
$dat = $sth -> fetchall_arrayref;
-]
<TABLE>
        <TR><TH>[+ $head -> [$col] +]</TH></TR>
        <TR><TD>[+ $dat -> [$row][$col] +]</TD></TR>
</TABLE>
```

3.2.1.1.2 Apache::ASP

Jeho účelem je emulovat prostředí ASP pro server Apache. Ale pozor, pouze pro případ, kdy je skriptovacím jazykem Perl, čili nikoliv VBScript nebo JScript. Podle popisu vypadá tento produkt dostatečně kompatibilní s ASP pro Internet Information Server. Modul *Apache::ASP* emuluje i všechny hlavní ASP objekty, jako například *Application a Session* objekt a také jejich události, například *Session_OnStart*. Nicméně pochopitelně neemuluje obecné ActiveX/COM objekty, ani objekt ADO určený na ASP platformě pro přístup k databázím. Pro přístup k databázím je nutné použít obecné perlovské databázové moduly. Toto je ovšem značná nevýhoda z hlediska přenositelnosti aplikací. Pro představu čtenáře si ukážeme malou ukázku stránky ASP napsané v Perlu. Následující ukázka ve smyčce vypisuje text ve zvětšujícím se fontu.

```
<!-- sample here -->
 <html>
 <body>
 For loop incrementing font size: <p>
 <% for(1..5) { %>
      <!-- iterated html text -->
      <font size="<%=$_%>" > Size = <%=$_%> </font> <br>
 <% } %>
 </body>
 </html>
 <!-- end sample here -->
```

Je užitečné podotknout, že přestože firma Microsoft jako skriptovací jazyky pro ASP na platformě Internet Information Server nabízí pouze VBScript a Jscript, je pro tuto platformu k dispozici i skriptovací jazyk Perl od dvou dodavatelů.

- PerlScript od firmy ActiveState
- PScript od firmy MKS

3.2.1.2 PHP

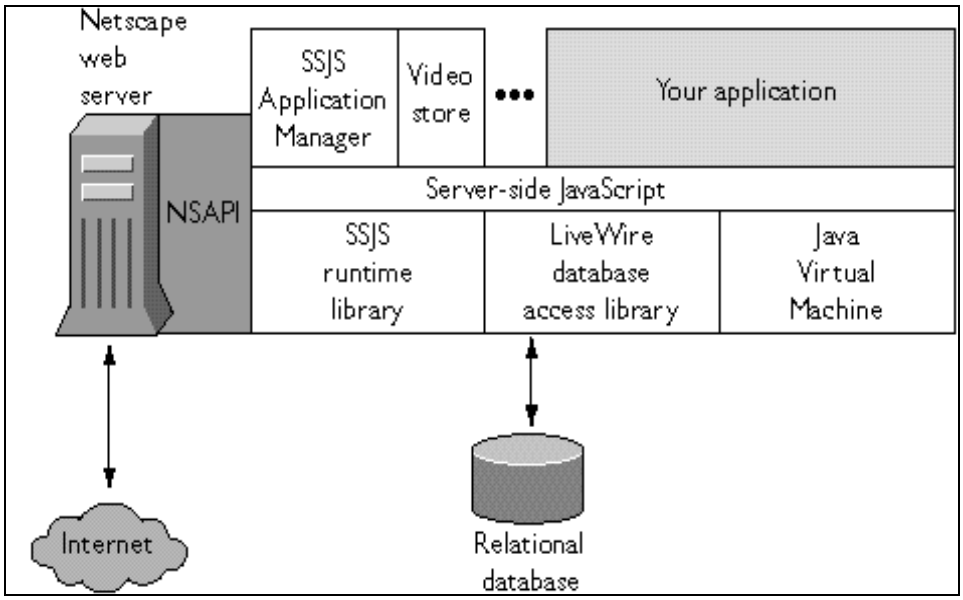
Hlavní vývojovým prostředím pro Apache je systém *PHP*, který je samostatným modulem. V současné době existuje ve verzi 3.0 a je na silném vzestupu. Malá ukázka, jak pomocí PHP komunikovat s databází je na výpisu č. 3. PHP disponuje velice širokým rozsahem funkcí, zejména pro téměř každý dostupný databázový systém. V PHP jsou také k dispozici ucelené aplikace, například konferenční systémy, groupware, nadstavba nad IMAP poštovní servery a další zajímavé projekty, jak komerční, tak volně dostupné.

Velice zajímavým produktem, který má souvislost s PHP je *ZEND*. Jedná se o kompilátor pro PHP, vyvíjený nezávislou firmou s cílem zachovat plnou kompatibilitu s PHP.

3.2.2 Netscape

Firma Netscape je tvůrcem nejen známého a dlouhou dobu dominantního prohlížeče Netscape Navigator, ale také celé řady vysoce výkonných WWW serverů vhodných pro průmyslové aplikace. Není bez zajímavosti, že například firma Novell ukončila vývoj svého WWW serveru a licencovala právě WWW server firmy Netscape.

Rozhraní nízké úrovně serverů firmy Netscape se nazývá NSAPI a je určeno pro jazyk C. My se však budeme věnovat rozhraní vyšší úrovně. Podobně jako firma Microsoft nabízí uživatelům Internet Information Serveru systém ASP, je odpovídající řešení firmy Netscape nazýváno *Server Side JavaScript (SSJS)*. Princip SSJS a jeho začlenění do WWW serveru je na obr. 3.



Obrázek 3 Server Side JavaScript – princip

Ukázka jednoduché stránky využívající SSJS následuje. Aplikace vypíše obvyklé *Hello* a informaci o IP adrese přistupujícího a počet přístupů.

```
<html><head> <title> Hello World </title></head>
<body><h1> Hello World </h1>
<p>Your IP address is
<server>write(request.ip);
        if (client.number == null)   client.number = 0
        else   client.number = 1 + parseInt (client.number, 10); </server>
<p>You have been here <server>write(client.number);</server> times.
</body></html>
```

Jak je patrné, je kód jazyka JavaScript uzavřen mezi HTML značku SERVER. SSJS má přístup k objektům WWW serveru a disponuje značnou řadou funkcí. Po stránce funkční je SSJS zcela rovnocenný ASP.

Existuje však několik rozdílů mezi ASP a SSJS. Na rozdíl od ASP se SSJS nejprve kompiluje a teprve kompilovaná forma stránek se instaluje na WWW server. Rozšíření názvu souboru HTML stránek se SSJS zůstává stále .HTML. Stránky obsahující jen kód SSJS mají rozšíření názvu .JS. Po překladu mají stránky s kódem SSJS rozšíření názvu .WEB. Výhoda kompilace stránek se SSJS je zejména v rychlosti vykonávání výsledného kódu.

3.2.3 LiveWire

V terminologie firmy Netscape je název LiveWire nyní rezervován pro označení spojení SSJS na nejrůznější databázové systémy. Spojení s databází je realizováno prostřednictvím objektově orientovaných funkcí SSJS, které nejsou obsaženy v Client Side JavaScriptu.

3.2.4 LiveConnect

V terminologie firmy Netscape je název LiveConnect určen pro technologii vzájemné komunikace mezi SSJS a moduly v jazyce Java. Komunikace směrem od SSJS k modulům v jazyce Java probíhá prostřednictvím tzv. *Packages*, které jsou WWW serverem zaregistrovány. LiveConnect je důležitým prostředkem pro komunikaci s tzv. *Netscape Application Serverem*, který je prostředím pro aplikační logiku WWW aplikace.

3.2.5 JavaScript Beans, Netscape Visual JavaScript

JavaScript Beans firmy Netscape jsou obdobou *JavaBeans* firmy SUN. Důvod pro vznik *JavaScript Beans* je snaha o vývoj standardních komponent v jazyce JavaScript. *Netscape Visual JavaScript* je visuální nástroj pro navrhování HTML stránek prostřednictvím Java, JavaScript a HTML komponent.

4. Technologie pro stranu klienta i serveru

4.1 Scriptlets

Další významnou technologií, kterou firma Microsoft obohatila tvorbu aplikací v prostředí Windows, je koncepce tzv. Scriptlets. Jedná se o COM objekty vytvořené skriptovými jazyky. Scriptlety je možné použít jak na straně klienta, tak na straně serveru a nebo dokonce jen tak! Jedná se pravděpodobně o ten nejjednodušší způsob vytváření COM objektů, který existuje!

Díky koncepci Microsoftu v oblasti skriptovacích technologií je možné jako skriptový jazyk použít nejen VBScript a JScript, ale i další skriptové jazyky slučitelné s technologií *Microsoft Script Engine*. Samotná firma uvádí jako další možné jazyky PERLScript, PScript a Python.

Samotné Scriptlets jsou uloženy v samostatném .sct souboru ve formátu XML. Skriptlet se skládá z několika základních částí.

- *<scriptlet> a <package> elementy.* <scriptlet> element uzavírá celou definici skriptletu. Pokud je v jednom .sct souboru obsaženo několik skriptletu, jsou uzavřeny v elementu <package>
- *<registration> element* obsahuje informace nutné pro registraci skriptletu jako COM objektu.
- *<public> element* zveřejňuje jaké property, metody a eventy jsou viditelné z vnějšku skriptletu.
- *<implements> element* specifikuje jaký COM interface bude pro daný skriptlet použit.
- *<script> element* obsahuje vlastní kód skriptletu, jeho property, metody a eventy a logiku chování.
- *<object> element* obsahuje informaci o objektech použitých ve vašem scriptletu.
- *<resource> element* obsahuje hodnoty, které je vhodné vyčlenit z kódu vašeho scriptletu, například z důvodu internacionalizace.
- *<reference> element* odkazuje na *type library* použitou ve vašem scriptletu.
- *<comment> element* vymezuje komentář.

Před vlastním použitím je nutné skriptlet zaregistrovat podobně jako jiné ActiveX/COM objekty. Je to možné použitím programu *regsvr32.exe*, například takto.

```
Regsvr32 file://my_server/my_scriptlet.sct
```

Pokud máme starší verzi programu regsvr32.exe, je možná následující syntaxe.

```
Regsvr32 scrobj.dll /n /i:http://my_server/my_scriplet.sct
```

kde *scrobj.dll* je knihovna umožňující práci se skriptlety. Skriptlety je možné využívat nejen jako COM objekty, ale pochopitelně i jako DCOM objekty.

4.1.1 Aplikační skriptlet

Aplikační skriptlet je skriptlet určený k nejširšímu použití jako COM nebo i DCOM objekt. Může se použít zcela samostatně nebo z HTML stránky na straně klienta. Ukažme si příklad jednoho velice jednoduchého skriptletu (výpis č. 1). Po zaregistrování tohoto skriptletu ho můžeme použít v naší aplikaci, například takto.

```
dim oSCT, x1, x2
Set oSCT = CreateObject("Pokus.Scriptlet")

x1 = oSCT.Add(1,2)
x2 = oSCT.Result
wscript.echo "x1: " & x1
wscript.echo "x2: " & x2
```

Zajímavější je však využití skriptletu v HTML stránce na straně klienta. Například takto.

```
<HTML><HEAD><TITLE>Pouziti skriptletu</TITLE></HEAD><BODY>
<OBJECT
   ID="oSCT"
   CLASSID="clsid:cb5bdc20-e513-11d2-9eb3-ceb8d8cde573">
</OBJECT>
Skriptlet na strane klienta!<BR>
<script language="JavaScript">
<!--
document.write("x1: ")
document.write(oSCT.Add(1,2))
document.write("<BR>x2: ")
document.write(oSCT.Result)
// -->
</script>
</BODY></HTML>
```

4.1.2 ASP skriptlet

Na straně serveru v .ASP stránkách by sice bylo možné používat skriptlety stejně, jako straně klienta, ale nevýhodou tohoto přístupu by byla skutečnost, že kód ve skriptletu by neměl přístup k objektům ASP. Tento nedostatek je odstraněn pomocí elementu *Implements,* jak je uvedeno na výpisu č. 2. ASP stránka, využívající tento scriptlet může být následující.

```
<HTML><HEAD><TITLE>Pouziti skriptletu</TITLE></HEAD><BODY>
Na strane serveru<BR>
<% Set oSCT = CreateObject("Pokus2.Scriptlet") %>
Je to <%= oSCT.OutPut  %>
</BODY></HTML>
```

4.1.3 DHTML behavior skriptlet

Nejnáročnější je implementace tzv. *behavior scriptlets*. Jedná se o speciální skriptlety spojené s chováním HTML elementů. Tyto skriptlety jsou pomocí událostí (*event*) připojeny k dokumentu a jeho elementům a umožňují tak definovat chování jednotlivých HTML elementů. Celá záležitost je poměrně komplikovaná, spojují se zde vlastnosti *kaskádních stylů, XML a behaviour scriptlets* v jeden celek, který umožňuje definovat prakticky libovolné chovaní elementů HTML stránky. Poskytují tak tvůrci WEBu veliké možnosti při definování WEBové aplikace. Vzhledem ke složitosti této problematiky a rozsahu tohoto článku nebudu čtenáři předkládat žádnou ukázku těchto skriptletů. Možná na ní dojde někdy v budoucnosti.

5. Závěr

Ve svém příspěvku jsem se snažil seznámit čtenáře s několika důležitými nebo zajímavými technologie vhodnými pro tvorbu Internetových a intranetových aplikací. Jejich výběr byl mimo jiné ovlivněn dostupností informací a osobními zkušenostmi. Proto skutečnost, že některé aplikace či nástroje jsou popsány snad až příliš stručně neznamená, že jsou nevýznamné.

Celkově je patrný, dle mého názoru, především veliký souboj firmy Microsoft proti celému světu. Je také patrná snaha o vznik visuálních komponent pro stranu serveru. Nicméně definitivní řešení je stále před námi. Uvidíme, jaké novinky nám přinese další rok.

Odkazy

Microsoft (data binding) ..... http://www.microsoft.com/workshop
Microsoft(scripting, Vbscript, Jscript) ..... http://msdn.microsoft.com
Microsoft a XML ..... http://www.microsoft.com/xml
Netscape (SSJS) ..... http://www.netscape.com
SUN (servlets, java) ..... http://www.sun.com
Java ..... http://java.sun.com
Apache ..... http://www.apache.org
Perl a Apache ..... http://perl.apache.org
Java a Apache ..... http://java.apache.org
PHP ..... http://www.php.net
W3C (HTML, XML) http://www.w3.org

Výpis 1 Jednoduchý scriptlet (pokus.sct)

```
<?XML version="1.0"?>
<scriptlet>
<registration
        description="Pokus"          progid="Pokus.Scriptlet"
        version="1.00" classid="{cb5bdc20-e513-11d2-9eb3-ceb8d8cde573}" />
<public>
        <property name="Result"/>
        <method name="Add">
                <PARAMETER name="num1"/>
                <PARAMETER name="num2"/>
        </method>
        <event name="overflow"/>
</public>
<script language="VBScript">
<![CDATA[
dim Result
function Add(num1, num2)
        Result = num1 + num2
        If Result > 1 Then fireEvent("overflow")
        Add = Result
end function
]]>
</script>
</scriptlet>
```

Výpis 2 ASP scriptlet (pokus2.sct)

```
<?XML version="1.0"?>
<scriptlet>
<registration
        description="Pokus2"       progid="Pokus2.Scriptlet"
        version="1.00"       classid="{60d19220-e52a-11d2-9eb3-f40fb6f7f373}"/>
<public>
   <property name="value"/>
   <method name="OutPut"/>
</public>
        <implements type="ASP" id="ASP"/>
<script language="VBScript">
<![CDATA[
dim value
value = "OK."
function OutPut()
   Response.Write(value)
end function
]]>
</script>
</scriptlet>
```

Výpis 3 Ukázka PHP

```
<?
$hostname = "myhost";
$username = "username";                    $password = "password";
$usertable = "mytable";                    $dbName = "mydb";
/* make connection to database */
MYSQL_CONNECT($hostname,$username,$password);
@mysql_select_db("$dbName");               /* select all users */
$query = "SELECT * FROM $usertable";
$result = MYSQL_QUERY($query);             /* number of rows */
$number = MYSQL_NUMROWS($result);          /*  make a table */
WHILE ($i < $number) :
        $name = mysql_result($result, $i, "name")
        PRINT "Name: $name"
ENDWHILE
?>
```