Limbaje de scriptare

Cuprins

limbaje server-side

asp - Luca Diana Sorina 431A

Scurt istoric
Introducere in ASP
Structuri de control
Functii si proceduri
Modelul obiectual
Bibliografie

jsp - Codita Bogdan Alexandru 431A

Introducere
De ce folosim JSP
Arhitectura JSP
Etapele necesare pentru o cerere JSP
Tag-uri JSP
Integrarea componentelor JavaBean
Comentarii JSP
Bibliografie

php - Busuioc Andrei 431A

Generalitati
Cum a aparut
Caracteristici
Tipuri de date si functii
Programarea Orientata pe Obiecte
Optimizari ale vitezei de executie
Compilatoare
Bibliografie

Generalitati

Un limbaj de scriptare este un limbaj de programare ce permite controlul uneia sau a mai multor aplicatii software. "scripturile" sunt diferite fata de codul de baza al aplicatiei, fiind de obicei scrise in in limbaje diferite fata de codul sursa al aplicatiei gazda. Scripturile sunt scrise de catre end-user, fiind adesea interpretate din cod sursa sau byte-code, unde aplicatia pe care o controleaza e in mod traditional compilata in cod masina.

Termenul *script* e derivat din forma scrisa a script-urilor artistice, unde dialogul este scris spre a fi vorbit de actori. Primele limbaje de scriptare erau numite *batch languages* sau *job control languages*. Limbajele de scriptare au fost create pentru a scurta procesul de editare-compilare-linkare-rulare al aplicatiilor.

Primele shell-uri interactive au fost create in anii 1960 pentru a permite controlul de la distanta al primelor sisteme bazate pe time-sharing, iar acestea, programe in interiorul unor programe.

GUI scripting sau macro-uri, au devenit utile odata cu aparitia interfetelor vizuale pentru aplicatii; aceste script-uri permit simularea interactiunilor umane cu aplicatia, lucru util in fazele de testare sau pentru automatizarea unor actiuni de altfel consumatoare de timp.

Limbaje specifice aplicatiilor gazda; in prezent exista multe aplicatii software ce permit realizarea de script-uri pentru programarea functionalitatilor puse la dispozitie de aplicatie, functionalitati care de altfel ar fi fost mult mai greu de exploatat daca s-ar fi mers numai pe interfete vizuale. Limbaje populare de scriptare sunt asociate engine-urilor grafice si permit realizarea de algoritmi de inteligenta artificiala in jocuri sau alte domenii unde programarea raspunsurilor aplicatiei e dorita si necesara de a fi facuta pentru fiecare caz particular in care aplicatia e folosita.

Browserele Web sunt cele mai raspandite forme de aplicatii ce folosesc in mod intens limbajele de scriptare, standarde riguloase fiind impuse pentru uniformizarea comportamentului fiecarui limbaj de scriptare.

Procesoarele text dispun de limbaje de scriptare pentru generarea fisierelor de istoric de activitati (log files) sau pentru fisierele de configurare.

Limbajele embeddable sunt limbajele care dispun de plug-in-uri create special pentru alte aplicatii gazda. Aceste plug-in-uri sunt apelate de fiecare data cand este intalnit un script asociat limbajului suportat, iar plug-in-ul e cel care interpreteaza script-ul si-l executa in cadrul aplicatiei gazda. Limbaje populare de scriptare de acest tip sunt: actionscript-ul Flash, Silverlight.

JavaScript

Introducere

JavaScript este un limbaj de programare din categoria limbajelor de scriptare, ce se foloseşte preponderent în paginile şi aplicaţiile web, pentru a îmbogăţi interfaţa cu utilizatorul. Script-urile sunt interpretate şi executate de către un engine (parser) care rulează într-un mediu „host” şi care acţionează ca o punte între nivelul scriptului şi nivelul aplicaţiei host, de obicei browserul web. Engine-ul permite scriptului accesul la API-ul host-ului prin intermediul unor metode sau obiecte. În cazul aplicaţiilor web, scriptul are acces la modelul de document implementat de către browser (DOM – Document Object Model). Deşi majoritatea scripturilor JavaScript rulează în cadrul browserului web, există şi alte host-uri ce permit execuţia lor. Cel mai întâlnit este „Windows Scripting Host”, dezvoltat de către Microsoft, care permite scripturilor să ruleze în cadrul sistemului de operare şi să acceseze API-uri legate de operaţii I/O, de managementul ferestrelor şi ale elementelor de interfaţă ş.a.m.d.

Caracteristici generale şi elemente de sintaxă

Limbajului JavaScript este puternic influenţat de limbajele C şi Java, de la care acesta împrumută atât sintaxa cât şi convenţiile de denumire. Principalele caracteristici de sintaxă sunt comune: structurile de control (if, while, for, switch), operatorii aritmetici, logici şi pe biţi, cuvintele cheie rezervate coincid cu cele din Java. O excepţie o reprezintă operatorul scope (::) care nu este suportat. În schimb scopingul se face prin folosirea unor obiecte ce îndeplinesc funcţia de namespace şi definirea variabilelor ca proprietăţi ale acestora. Se permite de asemenea omiterea simbolului „;” ca delimitator de statement-uri, fiind adăugat automat la sfârşitul liniei de către parser.
O diferenţă majoră faţă de limbajele Java şi C o reprezintă caracterul dinamic al variabilelor din punct de vedere a tipurilor de date. Dacă în C sau Java fiecare variabilă are un tip strict, în JavaScript o variabilă poate aparţine oricărui tip de date. Unei variabile i se poate asocia deci, spre exemplu, o valoare de tip int, iar apoi i se poate re-atribui o altă valoare de tip şir de caractere. Declararea variabilelor se face astfel printr-un cuvânt cheie general „var” şi este urmată opţional de o iniţializare:

```
var myVar = “myValue”
myVar = 25.31
myVar = 0;
```

Tipuri de date

Limbajul suportă tipuri de date primitive (boolean, numeric – float şi integer, string) şi tipuri de date complexe (obiecte). Obiectele sunt implementate ca tabele asociative de perechi proprietate – valoare. Numele proprietăţilor reprezintă indecşi în tabelă şi pot fi adăugate, şterse sau modificate în momentul execuţiei. Datorită acestei implementări, este posibilă enumerarea tuturor proprietăţilor unui obiect folosind sintaxa for...in. În JavaScript toate obiectele pot fi considerate ca fiind derivate din obiectul de bază Object, la fel ca în Java.

```
var myObj;
myObj.prop1 = ‘value’;
myObj[“prop2”] = “value2”;
```

Deşi este un limbaj obiect-orientat JavaScript nu suportă clase, în sensul clasic. Dacă în C sau Java, structura unui obiect se defineşte într-un mod strict, printr-o definiţie de clasă, în JavaScript efectul se obţine prin folosirea unor prototipuri, adică obiecte şablon, ce conţin structura clasei, şi care sunt clonate pentru a forma obiectul-instanţă. Clasa se defineşte astfel, definind constructorul ei, şi prototipul.

Datorită implementării ca tabele a obiectelor, structura lor poate fi modificată după instanţiere de către utilizator, fapt ce oferă un plus de flexibilitate. De asemenea, moştenirea claselor şi supraîncărcarea metodelor este obţinută prin această metodă. O altă posibilitate de creare a unor obiecte este prin enumerarea directă a proprietăţilor şi a valorilor prin sintaxa „ex nihilo” (din nimic), fără să se „cloneze” proprietăţile şi metodele predefinite într-un prototip, şi se pot folosi astfel ca tabele asociative:

```
var myClass = function(param1, param2,…) {
        //constructor body here
        this.prop1 = param1;
        this.prop2 = param2;
};
myClass.prototype = {
        prop1: “defaultVal1”,
                prop2: “defaultVal2”
};
```

```
var myEnumObj = {
        prop1: 'value1',
        prop2: 'value2',
        nestedObj: {
                prop: "value"
        },
        myArray: [1,2, 47, 256]
};
```

Funcţii, proceduri, metode

În JavaScript funcţiile sunt implementate ca obiecte de tip Function şi pot fi manipulate astfel ca oricare alt tip de date. Definirea lor se poate face fie prin sintaxa asemănătoare cu cea a C-ului

function function_name(param1, param2, ….) { …body….}

, fie prin sintaxa anonimă, în care definiţia funcţiei poate fi inclusă într-o expresie oarecare care returnează funcţia ca obiect de tip Function:

var function_name = function(param1,param2,…) { ….body….}.

De asemenea se pot construi expresii în care se definesc şi se apelează funcţii anonime „de unică folosinţă”:

(function(param1,param2,..){...body....})(val_param1, val_param2,...);

Limbajul nu face de asemenea diferenţa între funcţii şi metode, acestea putând fi interschimbate cu uşurinţă datorită implementării ca obiect, schimbând asocierea variabilei *this*. De asemenea funcţiile se confundă si cu constructorii. Prin folosirea cuvântului cheie *new* în faţa apelului de funcţie se provoacă apelarea funcţiei drept constructor, ce are ca efect clonarea prototipului şi returnarea obiectului rezultat. Un alt efect al implementării funcţiilor ca obiecte este posibilitatea includerii unor funcţii în alte funcţii, numit şi „nesting”, care permite şi încapsularea sau crearea unor spaţii private ce nu pot fi accesate din afară:

```
//Exemplu de încapsulare şi imbricare
/*Spaţiul unei aplicaţii ce conţine două proprietăţi interne şi o funcţie imbricată ce nu pot fi
accesate din exterior şi care returnează ca interfaţă o metodă şi o proprietate publică*/
var myApp = (function() {
        //private
                var private_var1, private_var2…;
        function internal_func() {return '…'};

        //public
        return {
                public_var1: value1,
                        public_func1: function() { return internal_func(); }
                        …
                }
})();
```

Alte facilităţi

Funcţiile folosesc parametri formali, adică tipul şi numărul parametrilor nu sunt statice. Se poate apela o funcţie cu un număr nedeterminat de parametri, accesul la aceştia putându-se face prin obiectul *arguments* existent în scope-ul funcţiei. Dacă declaraţia conţine mai mulţi parametri decât cei transmişi la apelare, atunci se completează cu NULL automat parametrii netransmişi. Vectorii (arrays) şi obiectele pot fi specificate inline, în cadrul oricărei expresii, prin sintaxa enumerată menţionată mai sus.

Printre altele, JavaScript suportă expresii regulate compatibile cu sintaxa din Perl, pentru manipularea complexă a şirurilor de caractere, şi obiecte integrate pentru formatul datei-orei (Date), a şirurilor de caractere (String) şi a valorilor numerice (Number).

Datorită faptului că scripturile sunt evaluate la rulare, fără o compilare prealabilă, este posibilă evaluarea prin intermediul unei funcţii eval(), a unui şir de caractere oarecare, ce permite construirea unui statement la rulare.

JavaScript în aplicaţiile web

Principala utilizare a limbajului JavaScript este în aplicaţiile şi paginile web, ca mijloc de îmbogăţire a interfeţei cu utilizatorul. Scriptul rulează în cadrul browserului web, într-un „sandbox” ce izolează scriptul de mediul „hostului”, şi are la dispoziţie obiecte ce permit manipularea documentului (paginii web) sau a ferestrei browserului. Un astfel de script poate îndeplini o plajă largă de funcţii, de la adăugarea unor elemente dinamice (meniuri drop-down, widgeturi etc), validarea unor formulare, deschiderea unor ferestre popup, la construirea de

aplicaţii cu interfeţe complexe, prin manipularea obiectului-document (DOM). Modalitatea de livrare către client este în cadrul documentului HTML prin imbricare directă între etichetele <script>…</script>, sau prin includerea în fişiere separate ce conţin numai cod JavaScript. Fiind un limbaj interpretat, clientul primeşte codul sursă, în plan, iar engine-ul din cadrul browserului îl evaluează la rulare, fără o compilare prealabilă.

```
<html>
<head>
	<title>Example</title>
</head>
<body>
	<script type=”text/javascript”>
		document.write(“Hello world!”);
	</script>
	<noscript>
			<p>Your browser doesn’t support JavaScript, or JavaScript is
			disabled</p>
		</noscript>
</body>
</html>
```

Compatibilitatea aplicaţiilor

Deşi facilităţile de bază ale limbajului JavaScript sunt comune, indiferent de engine-ul de interpretare, interfeţele DOM ce permit manipularea documentelor web sunt dependente de browserul care rulează codul şi apar astfel diferenţe între browsere ce duc la incompatibilităţi. Astfel un cod care funcţionează spre exemplu pe Mozilla Firefox, de multe ori este posibil să nu funcţioneze pe Internet Explorer şi nu numai. Pentru a se trece peste aceste dificultăţi se scriu secţiuni de cod alternative şi se testează existenţa proprietăţilor sau a metodelor care depinde de implementarea browserului înainte ca acestea să fie folosite.

Manipularea paginilor web folosind JavaScript

O pagină web (document) este implementată de către browsere ca un arbore ce conţine un noduri pentru fiecare tag HTML. Nodul rădăcină corespunde tagului HTML, iar celelalte noduri sunt adăugate ca ramuri ale sale. Această structură este denumită DOM (Document Object Model), iar accesul codului JavaScript la aceasta este garantat prin intermediul unei interfeţe (API).

Astfel se stabileşte o echivalenţă între funcţiile din codul sursă al browserului şi funcţiile oglindă din codul JavaScript.

Un script JavaScript poate să adauge, să modifice, sau să şteargă orice nod din document, putând astfel genera conţinut HTML dinamic (DHTML). Acest fapt face posibilă crearea unor aplicaţii cu intefeţe complexe bazate pe HTML şi CSS ca mijloc de randare / layout. În acest scop au apărut o serie de framework-uri ce permit crearea dinamică a unor elemente de interfaţă avansate (widget-uri, controale etc): ExtJS, Yahoo API, Adobe Spry etc.

Exemplu de manipulare a DOM-ului folosind JavaScript:

```
<html>
<head>
	<title>Modifying DOM Structure</title>
	<script type="text/javascript">
			//functie ce se executa atunci cand pagina s-a incarcat, folosind //eventul
			onLoad
		function onLoad() {
			var targetNode = document.getElementById('emptyDiv');
			var childNode = document.getElementById('childDiv');
			//Muta nodul childDiv din  parentDiv in emptyDiv
			targetNode.appendChild(childNode);
			}
		</script>
</head>
<body onLoad="onLoad()">
	<div id="emptyDiv" />
	<div id="parentDiv">
		<div id="childDiv" />
	</div>
</body>
</html>
```

Bibliografie

JavaScript > Caracteristici si elemente de sintaxa > http://en.wikipedia.org/wiki/Javascript
JavaScript > Clase si obiecte, prototipuri > http://en.wikibooks.org/wiki/JavaScript

VbScript

Notiunii introductive

VbScript, sau Visual Basic Scripting Editor, este un limbaj de scriptare realizat de catre firma Microsoft, bazat pe Visual Basic. Prima varianta a aparut in 1996, facand parte din Tehnologia de Scriptare a Microsoft Windows. Este un limbaj de scriptare activ, adica o tehnologie folosita de sistemele de operare Windows pentru a “implementa support-ul scripturilor bazate pe componente” (sursa http://en.wikipedia.org/wiki/Active_Scripting). Desi se bazeaza pe Visual Basic, este un limbaj mult mai simplu, mai apropiat de java script. Acest proces de simplificare a fost datorat necesitatii unui limbaj usor de interpretat de catre serverele web, principalul scop al acestui limbaj fiind de a fi recunoscut de catre browser-ul web Internet Explorer al lui Microsoft. Acest limbaj de scriptare ajuta realizatorilor de pagini web sa faca anumite operatii greu de realizat in limbaj html, si anume punerea de scroll-uri sau de butoane interactive in pagini, s.a.m.d.

Fiind un limbaj de scriptare activ, functionarea lui se bazeaza pe COM-uri. COM-urile au fost introduse tot de Microsoft in 1993 si au devenit un standard pentru dezvoltarea de softuri. COM-urile, defapt, sunt limbaje „neutre, ce pot fi folosite in medii diferite decat cel in care au fost im-plementate” (sursa http://en.wikipedia.org/wiki/Component_Object_Model), portabilitatea fiind marele avantaj al lor. Nu au nevoie sa stie exact cum functioneaza un program, acestea fortand softul sa isi creeze o interfata separata de implementare. Fiind versatile, stau la baza multor softuri. COM-urile sunt cel mai des folosite in Microsoft Windows. In prezent, COM-urile au inceput sa fie inlocuite cu Microsoft .NET framework, dar, in acelasi timp, obiectele COM pot fi folosite cu toate limbaje .NET.

Marele avantaj al vbScripturilor este faptul ca sunt foarte asemanatoare cu Visual Basic, deci orice cunoscator al acestui din urma limbaj nu va avea niciun fel de problema in a implementa o aplicatie folosind vbScript. Si chiar daca programatorul nu cunoaste vbScriptul, acest limbaj de scriptare ramane un limbaj simplu si usor de implementat in comparatie cu alte limbaje de scriptare.

De la Windows 98 pana in prezent, vbScript a fost inclus in pachetul limbajului de programare, astfel ca pentru utilizatorii de Windows vbScript e la indemana. De asemenea, Microsoft ofera un support pentru acest limbaj de scriptare. In plus, orice aplicatie realizata cu ajutorul vbScript poate fi licentiata pe gratis.

Ar mai trebui mentionat si Windows Script, care este un limbaj de scriptate special realizat de Microsoft pentru a ajuta browser-ele sa interpreteze limbaje de scriptare diferite si ofera posibilitatea serverelor sa compileze scripturi.

La ce este folosit vbScript

Orice vbScript, ca si alte scripturi, trebuie rulat intr-un anumit mediu: cel amintit mai sus, Windows Script, Internet Explorer si Internet Information Service. In acelasi timp, un vbScript poate fi incapsulat in alte programe cu ajutorul tehnologiilor Microsoft Script Control.

VbScript este folosit in multe domenii, dar de remarcat sunt 3 domenii:

1) “Administrarea sistemelor intr-un mediu Windows;
2) Limbaj de scriptare pentru Quick Test Professional;
3) Limbaj intern de scriptare pentru aplicatii cum ar fi interefetele operatorilor industriali” (sursa http://en.wikipedia.org/wiki/VBScript).

Desi inca se foloseste vbScript pentru administrarea sistemelor, in viitor e posibil sa se treaca integral la folosirea lui Windows PowerShell in acest scop. Acesta este un engine ce consta dintr-o linie de comanda si un limbaj de scriptare asociat. Avantajul mare al acestuia este posibilitatea sa de dezvoltare.

Quick Test Professional este un soft “user friendly” folosit cu scopul de a testa performantele unor aplicatii software sau a unor site-uri web. A fost realizat cu ajutorul vbScriptului, dar nu numai.

Evolutia vbScript

VbScript a aparut in 1996 si, in doar 2 ani, a ajuns la varianta 2.0. Se cauta o alternativa pentru limbajele batch care au fost realizate la sfarsitul anilor 1970. Limbajele batch sunt limbaje folosite de sistemul de operare in command prompt pentru a realiza anumite functii, ce au aparut odata cu DOS-ul.

Versiunea 5.0 a a adus ca inovatii precum: “regular expressions”, asociarea unui parametru a unui sir de caractere, includerea claselor, comanda “with”, functiile “eval”, “execute”, “executeglobal”, pointerul “getref”, si supportul DCOM.

Variantei 5.5 i-au fost adaugate la sintaxa claselor in vbScript “SubMatches”.

Viitorul vbScriptului este un pic incert. Se incearca trecerea la Windows Power Shell, dar cu toate acestea exista, in continuare, o echipa de support si o echipa de dezvoltare care aduc periodic inovatii la vbScript. In urmatoarele versiuni de Microsoft Windows e foarte posibil sa vedem vbScript in continuare.

Functionarea vbScript pe servere

Atunci cand implementam vbScript intr-un server, vbScript este asemanator cu JavaScript. Asemenea acestuia din urma, intr-un limbaj html, este necesara linia <SCRIPT LANGUAGE = “VBScript”> ... </SCRIPT> pentru a integra un cod scris in vbScript in interiorul unui html. Problema vbScripturilor este aceea ca browsere web precum Firefox nu au suport pentru vbScript, fiind necesara o implementare separata a unor aplicatii care sa recunoasca limbajul, dar browserul Internet Explorer recunoaste acest limbaj de scriptare.

VbScript este folosit si pentru Microsoft “Active Server Pages”, primul engine realizat de Microsoft pentru paginile cu continut dinamic. Pentru a marca un vbScript in interiorul unui ASP se folosesc, in mod asemanator cu html, marcajele <% ... %>.

Un exemplu de vbScript intr-un cod html:

```
‹HTML>
<HEAD>
<TITLE>Place Your Order</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
  Function CanDeliver(Dt)
    CanDeliver = (CDate(Dt) - Now()) > 2
  End Function
-->
</SCRIPT>
</HEAD>
<BODY>
```

(Sursa http://msdn.microsoft.com/en-us/library/3945y0f9(v=VS.85).aspx).

Marcajul <SCRIPT> spune codului html ca urmeaza un script si va sti cum sa il analizeze, insa foarte important este specificarea tipului de script deoarece, in functie de browserul web, va fi posibila sau nu aplicarea lui. Liniile de comentarii <!-- ... --> ajuta browserele care nu pot interpreta vbScriptul sa nu afiseze codul.

Scriptul poate fi pus oriunde in pagina, doar ca se prefera punerea lui in head, ca apoi in body sa fie apelat. De asemenea, este de preferat ca tot codul sa se afle in aceasi parte a codului html, pentru a nu aparea probleme la apelul lui.

Variant

Mai devreme am facut referire la clase si string-uri de caractere. VbScripturi, in esenta, au un singur de date numite variant. Acestia pot contine difirite tipuri de date in functie de modul in care vor fi folositi. Un variant poate fi, deci, fie o valoare, fie un sir de caractere, in functie de cerinta aplicatiei. Bineinteles ca un vbScript va returna un singur tip de date, un variant.

VbScriptul-ului nu i se explica exact cu ce tip de date lucreaza, ci el va interpreta variantul in functie de cerinta. De exemplu, daca aveam o informatie despre data sau ora, vbScript va interpreta mereu aceasta ca un numar. Dar, se poate si sa ii explicatam, in caz ca e nevoie, daca avem nevoie ca un sir de numere sub forma unui string de caractere si asta o facem folosind ghilimele (“ ...”).

Un alt lucru important despre varianti este ca si aceste 2 tipuri de varianti, numerici sau string-uri, se impart in mai multe tipuri de varianti, in functie de valorile pe care le iau. Asemanator cu limbajul C, avem:

- Empty: variantul ia valoarea 0 sau un string de caracter gol, (“ “);
- Null: variantul contine intentionat nu contine date valide;
- Boolean: contine fie “true” fie “false”, respectiv 0 sau 1;
- Byte: ia valori intre 0 si 255;
- Integer: ia valori intre -32,768 si 32,767;
- Currency: ia valori intre -922,337,203,685,477.5808 si 922,337,203,685,477.5807;
- Long: ia valori intre 2,147,483,648 si 2,147,483,647;
- Single: contine un singura numar in virgula mobila ce ia valori intre -1.79769313486232E308 si -4.94065645841247E-324 pentru numere negative si 4.94065645841247E-324 si 1.79769313486232E308 pentru numere pozitive;
- Date(Time): contine un numar ce reprezinta o data intre 1 Ianuarie 100 si 31 decembrie 9999;
- String: contine un sir de caracter de maxim, aproximativ, 2 miliarde de caractere;
- Object: contine un obiect;
- Error: contine un numar eroare.

De altfel, limbajul de scriptare ofera posibilitatea convertirii unui tip de date cu un alt tip. Mai exista si o functie VarType care returneaza tipul datei(important pentru a sti cum a fost interpretata acel tip de date, daca a fost interpretat de limbaj asa cum am vrut noi).
(Sursa: http://msdn.microsoft.com/en-us/library/9e7a57cf(v=VS.85).aspx).

Variabile si constante VbScript

O variabila vbScript se refera la o locatie de memorie unde este memorat, unde se stocheaza o parte din program. Aceasta poate fi modificata de mai multe ori pe parcursul programului, dar la final programul retine adresa si stie de unde sa ia valoarea respectiva. Toate tipurile de variabile in vbScript sunt varianti.

Tipurile de variabile pot fi, similar cu limbajul C, publice sau private. In functie de tipul lor vor putea fi accesate doar din clasa din care fac parte sau si dinafara clasei. De asemenea, variabile pot fi declarate explicit, prin a declara tipul si numele variabilei, similar cu limbajul C, sau pot fi declarate implicit, atunci cand este specificat undeva in script numele variabilei. Aceasta metoda nu este indicata deoarece, in cazul in care scriem gresit vreo litera, pot aparea rezultate ciudate in urma rularii aplicatiei.

Ca si in C, numele variabilelor are anumite conditii. O prima conditie ar fi sa nu se suprapuna cu anumite cuvinte cheie ale limbajului, cum ar fi tipurile de varianti declarati mai sus. Alte conditii ar mai fi sa inceapa mereu cu o litera, sa nu depaseasca 255 de caractere si sa nu existe 2 variabile cu acelasi nume pe parcursul scriptului.

Asignarea unei valori la o variabila se face in mod traditional prin comanda: variabila = 5. In cazul in care avem o data sau o ora, putem pune inainte variabilei un #, ca in exemplul urmator: data = #17/04/2010#.

In vbScript se folosesc variabile de tip scalare sau de tip array. Variabile de tip scalar sunt variabilele clasice cu o singura valoare, in timp ce variabilele array sunt un sir de variabile. Declararea unei variabile de tip array se face folosind parantezele, ca in exemplul urmator: Dim vector(5). V(5) este echivalentul unui vector in limbajul C, cu 6 valori posibile, de la 0 la 5. Dupa declararea unei variabile de tip array putem sa declaram fiecare valoare particulara acestuia, lafel ca in C, astfel: v(0) = 12, v(1)= 5, etc. Pe de alta parte un array poate fi si o matrice cu un numar maxim de 60 de coloane(unde o coloana este echivalenta cu vector). Declararea unei astfel de matrici se face astfel: Dim matrice(5, 3), unde primul numar este numarul de linii si al 2 lea de coloane. De asemnea, pot fi declarate si array-uri de dimeninsiune dinamica atunci cand dorim ca dimensiunea array sa varieze pe parcursul aplicatiei. Declararea acestora se face omitand orice parametru de dimensiune, astfel: Dim matrice().

Despre constante, ce ar fi de zis este ca acestea se declara folosind cuvantul cheie “const”. Punerea acestuia in fata unei variabile face ca aceasta sa nu mai poate fi modificata pe parcursul programului. Exemplu: Const name_author = “Costanzo Riccardo”.

Operatori VbScript

Exista in vbScript o biblioteca de operatori speciali care sunt interpretati de program. Pentru altfel de operatii ar trebuie create functii de utilizator, asemanator cu limbajul C. Pe langa operatori exista parantezele, care forteaza o anumite ordine a executiei operatilor. In cazut in care nu avem paranteze ordinea fireasca a operatilor este: operatori aritmetici, operatori de comparare si operatori logici, si acestuia fiind efectuati de la stanga la dreapta.

In continuare voi prezenta o lista de operatori (sursa pentru acestuia este http://msdn.microsoft.com/en-us/library/9da4s2eh(v=VS.85).aspx):

1) Operatori aritmetici:
- Exponential: ^;
- Adunare: +;
- Scarede: -;
- Inmultire: *;
- Impartire: /;
- Modul: Mod;
- Partea intreaga a unei impartiri: \;
- Concatenarea sirurilor de caractere: &.

2) Operatori de comparatie:
    - Egalitate: =;
    - Inegalitate: <>;
    - Mai mic: <;
    - Mai mare: >;
    - Mai mic sau egal: <=;
    - Mai mare sau egal: >=;
    - Egalitate intre obiecte: Is.

3) Operatori logici:
    - Negare logica: Not;
    - Si logic: And;
    - Sau logic: Or;
    - Sau exclusiv logic: Xor;
    - Echivalenta logica: Eqv;
    - Implicare logica: Imp.

Instructiuni folosite in vbScript

Ca si in alte limbaje, exista 2 tipuri de instructiuni: conditionale si repetitive. Instructiunile conditionale sunt: “if ... then ... else” si “select case”. Instructiuniile repetitive sunt: “Do ... Loop”, “While ... Wend”, “For ... Next” si “For each ... Next”.

“If ... Then ... Else” realizeaza o operatie in functie de conditiile stabilite. In cazul in care rezultatul operatiei din if este true, realizeaza o operatie ce urmeaza dupa if, altfel realizeaza o operatie ce urmeaza dupa else. Exemplu:

```
If numar = 5 Then
    MsgBox numar
Else
    MsgBox numar+1
Endif
```

In urma executarii exemplului de mai sus va aparea un mesaj cu 5, in cazul in care numar are valoarea 5 sau cu numar+1, in cazut in care numar are alta valoare.

“Select Case” functioneaza similar cu “if ... then ... else”, doar ca este mult mai bine structurat si presupune mai putine linii de cod. Instructiunea incepe cu evaluarea unui expresii, iar apoi va fi comparata cu un numar dorit de cazuri. Exemplu:

```
Select Case numar
    Case numar = 5
        MsgBox numar
    Case numar!= 5
        MsgBox numar+1
End Select
```

In acest exemplu nu se observa avantajul lui “Select Case”, insa, daca erau mai multe conditii, atunci era mai avantajos decat un numar mare de “If ... Then ... Else If ... Then .... Else ...”.

Pentru instructiunea “Do ... Loop”, se executa un set de instructiuni atata timp cat o conditie este adevarata sau pana cand o conditie devine adevarata, in functie de ce se doreste. Exemplu:

```
Do
    Numar = numar -1;
Loop While numar > 0;
```

In acest exemplu, numar se va decrementa pana va ajunge sa fie egal cu 0.

“While ... Wend” este o alternativa la “Do ... Loop”, insa nu este lafel de flexibila si nu este recomandata folosirea ei.

Instructiunea “For ... Next” are avantajul ca poate fi ales un numarul de repetari al buclei, ca in exemplul urmator:

```
For i = 0 to 9
    Numar = numar-1;
Next
```

Instructiunea “Each ... Next” nu repeta doar o instructiune de un numar dorit de ori, ci o grupare de instructiuni pentru o colectie de obiecte sau pentru o matrice.

Compilatoare si dezasambloare vbScript

Nu exista un compilator propriu-zis de vbScript, insa exista debuggere de vbScript. Un astfel de debugger este vbsedit, care poate fi downloadat in versiunea pentru 32 de biti sau 64 de biti. Altfel, orice aplicatie de vbScript poate fi realizata in orice editor text, cu singura conditie ca acest fisier sa fie salvat cu extensia “.vbs”.

Concluzie

Limbajul de scriptare vbScript nu este lafel de popular precum javaScript, insa probabil ca avem de-a face cu el zilnic cand navigam pe internet. Deriva din Visual Basic, ceea ce ii confera o usurinta in implementare, este un limbaj intuitiv, operatori si instructiunile sunt foarte asemanatoare cu cele din limbajul C, acesta fiind un mare avantaj pentru orice programator de nivel mediu. Dezavantajele limbajul sunt lipsa lui de popularitate. Deoarece tot ce se face in vbScript poate fi facut in javaScript, care este un limbaj mai elegant in opinia mea, vbScript isi pierde din popularitate. Totusi vbScript a supravietuit 12 ani si e posibil sa mai reziste sau sa ii mai fie aduse inovatii de catre Microsoft.

Active Server Pages

Scurt istoric

Tehnologia ASP a aparut in noiembrie 1996 cand Microsoft si-a anuntat proiectul pentru Active Platform. Active Platform reflecta ideile Microsoft despre modul in care ar trebui sa comunice un desktop computer si un server. Ea presupune doua parti: Active Desktop si Active Server. Active Desktop-ul se refera la partea de client, sau de utilizator, unde fisierele HTML sunt afisate intr-un browser. Active Server-ul se refera la componentele la nivel de server. Acestea consta din pagini care pot fi interpretate de catre server, de unde si termenul de Active Server Pages.

Tehnologia ASP a devenit cu timpul o parte integranta a numeroaselor tehnici Windows orientate spre Web, devenind pentru cei care realizeaza aplicatii Web unul dintre cele mai naturale si normale moduri de-a construi pagini Internet dinamice, chiar aplicatii Internet pe aceste platforme.

Documentatia Microsoft descrie ASP ca fiind “un mediu de scripting la nivel de server care poate fi folosit pentru crearea si rularea unor aplicatii Web dinamice, interactive si de performanta ridicata”. ASP extinde standardul HTML adaugand scripturi “server-side”, acces la baze de date si componente ActiveX. Când browser-ul cere o pagina ASP, server-ul IIS o citeste, executa script-urile server si trimite rezultatul clientului în format HTML.

Versiuni:

- ASP 1.0 (distribuit cu IIS 3.0) in Decembrie 1996,
- ASP 2.0 (distribuit cu IIS 4.0) in Martie 1998,
- ASP 3.0 (distribuit cu IIS 5.0),
- ASP.NET (parte componenta a platformei Microsoft .NET). Versiunile pre-.NET sunt referite in prezent ca si ASP “clasic”.

Avantajele tehnologiei ASP:

- combina HTML si scripturi in acelasi fisier pentru a construi aplicatii al caror cod sa fie usor de inteles si de intretinut;
- pentru ca o mare parte din scripturi sunt rulate pe server, nu vor aparea probleme legate de capacitatea browserelor de a rula aceste scripturi;
- suporta atat VBScript cat si JavaScript;

- lucreaza cu modele obiectuale bine definite;
- ofera posibilitatea pastrarii de informatii intre paginile unei aplicatii sau chiar intre diverse accesari ale unei aplicatii Web;
- permite programatorilor care cunosc limbajul Visual Basic sa adauge unei aplicatii Web si acele functionalitati care inainte cereau programarea folosind CGI sau ISAPI.

Introducere in ASP

ASP vine ca parte a IIS in orice pachet Windows Professional. Aceasta strategie de piata a facut ca tehnologia ASP sa fie destul de raspandita. ASP a trecut prin 3 versiuni, ASP 3.0 fiind ultima. Limbajul cel mai des utilizat cu ASP Clasic este VBScript. Multi dezvoltatori de aplicatii web folosesc insa implementarea Microsoft pentru JavaScript, numita Jscript.

ASP 3.0 vine cu 6 obiecte integrate pe care programatorul le poate folosi pentru a construi pagini web dinamice. Aceste obiecte sunt: Application, Response, Request, Server, Session si ASPError.

In continuare vom incerca sa cream pagini web dinamice folosind ASP. In primul rand toate paginile ASP 3.0 sunt prevazute cu extensia .asp. Aceasta ii spune serverului sa acceseze biblioteca asp.dll pentru a procesa pagina. In al doilea rand intregul cod ASP trebuie sa fie inclus intre urmatorii delimitatori:

<%

%>

Acestia ii indica serverului sa proceseze informatia ca si cod ASP. Linia care trebuie sa urmeze dupa delimitatorul initial <% este @Language=VBScript pentru a indica faptul ca limbajul folosit in aceste pagini este VBScript. In continuare putem sa introducem cod Visual Basic Scripting intre aceste paranteze.

Pentru inceput, un element folositor si simplu este metoda *write* a obiectului *response*. *Response.write* va afisa in pagina un string sau valoarea unei variabile. De exemplu:

response.write("Hello!")

va afisa:

Hello!

In acest moment pare mult mai simplu sa cream o pagina HTML care sa faca acelasi lucru. Sa presupunem insa ca am dori sa afisam “Hello!” de 5 ori. Utilizand o instructiune for-next-loop codul ASP ar avea 3 randuri:

```
    For i = 1 to 5
       response.write("Hello!<br>")
Next i
```

In timp ce codul HTML ar avea 5 randuri:

```
Hello!<br>
Hello!<br>
Hello!<br>
Hello!<br>
Hello!<br>
```

Acesta este un exemplu simplu si nestructurat, dar arata modul in care ASP ne poate usura munca. In exemplul anterior este utilizata o variabila “i” pentru a realiza iteratiile in for-next-loop. In ASP toate variabilele trebuie declarate utilizand “Dim”:

```
Dim i
```

Aceasta declara variabila “i” ca fiind de tip variabil (“i” poate contine un intreg, float, string, etc.) Toate constantele trebuie declarate utilizand cuvantul cheie “Const”:

```
Const t=5
```

Pentru a ne asigura ca toate variabilele sunt declarate, una din primele linii de cod ar trebui sa fie “Option Explicit”. Exemplul anterior for-next-loop ar functiona si fara declararea initiala a lui “i”, dar este considerat un stil de programare gresit, nestructurat. In restul exemplelor vom considera ca “Option Explicit” precede exemplul, precum si declararea variabilelor.

Structuri de control

- If-Then-Else: este folosita pentru a rula diferite blocuri de cod in functie de anumite conditii.

```
If name="Joe" then
        Response.write("Hello Joe!")
else
        Response.write("I don't think we have met.")
End If
```

- For-Next: executa un bloc de cod de un numar specificat de ori.

```
for i=1 to 20
        response.write(i)
next
```

Acest exemplu afiseaza valoarea lui “i” de 20 de ori, de fiecare data incrementandu-l. Aceasta va produce: 12345678910111213141516171819 20. In orice moment se poate iesi din instructiune folosind “Exit For”.

```
y=7
for i=1 to 20
        response.write(i)
        if i=y then exit for
next
```

In acest exemplu cand i=y se iese din instructiune si se proceseaza restul paginii.

- For-Each-Next: repeat un bloc de cod pentru fiecare element dintr-o colectie sau dintr-un vector.

```
Dim x(20)
For Each g in x
        response.write(g)
Next
```

Instructiunea “Exit For” functioneaza in acelasi fel ca si in cazul anterior.

- Do-While/Until: executa un bloc de cod cat timp o conditie este adevarata sau pana cand o conditie devine adevarata.

```
x=1
Do While x<10
        Response.write("Hi There!<BR>")
        x=x+1
Loop
```

Acest exemplu afiseaza de fiecare data Hi There!
, incrementandu-l pe “x”. Cand x ajunge la valoarea 10 loop-ul se incheie si pagina continua sa fie procesata. Loop-ul Do-Until functioneaza in acelasi fel, dar in loc sa parcurgem loop-ul cat timp o valoare ramane adevarata, il parcurgem pana cand o valoare devine adevarata.

In orice moment se poate iesi dintr-o astfel de instructiune prin “Exit Do”.

```
x=10
y=7
Do Until x<5
        response.write("hello")
        x=x-1
        if x=y then Exit Do
Loop
```

In acest exemplu cand x il egaleaza pe y loop-ul nu mai continua si se va procesa restul paginii.

- While-Wend: executa un bloc de cod cat timp o conditie e adevarata.

- **While-Wend**: executa un bloc de cod cat timp o conditie e adevarata.

```
x=1
While x<10
        Response.write("Hi There!<BR>")
        x=x+1
Wend
```

Se afiseaza **Hi There!
** si se incrementeaza “x”. Instructiunea se incheie cand “x” ajunge la valoarea 10.

- Select-Case: este folosita pentru a inlocui instructiunile *if-then-else* in cazurile in care exista mai multe conditii. Se foloseste pentru conditii precise sin u pentru un interval de conditii.

```
JobType="Programmer"

Select Case JobType
        Select "GraphicsDesigner"
                Response.write("I am a Graphics Designer")

        Select "NetworkEngineer"
                Response.write("I am a Network Engineer")

        Select "Programmer"
                Response.write("I am a Programmer")
End Select
```

In acest exemplu in pagina se va afisa "I am a Programmer". Se poate adauga ramura “Case Else” care va fi executata daca nu este gasita conditia pe celelalte ramuri.

- With: permite efectuarea de operatii asupra unui obiect fara ca acesta sa fie referit de fiecare data

```
With ColorScheme
    .Trim="Green"
    .MainColor="Red"
    .PinStrips="Yellow"
End With
```

Functii si proceduri

Declararea functiilor:

```
Function nume_functie(lista_parametrii)
…corp functie…
nume_functie=…
…
End Function
```

Declararea procedurilor:

```
Sub Nume_procedura(lista_parametrii)
...corp procedura...
...
End Sub
```

Apelul functiilor:

```
variabila=nume_functie(lista_variabile)
sau
variabila=nume_functie(lista_valori)
```

Apelul procedurilor:

```
Call nume_procedura(lista_variabile)
sau
nume_procedura lista_variabile
```

Exemple:

```
g=totalamount(1,5)
Function TotalAmount(x,y)
    TotalAmount=x+y
End Function
```

In aceasta functie valorile x si y sunt date ca parametri, adunate, iar rezultatul final este returnat. Acesta este un exemplu in care functia returneaza o valoare. Functiile pot sa execute cateva linii de cod si sa revina fara sa returneze nici un rezultat

```
Call WriteThis
Function WriteThis
        Response.write("Hello!")
        Response.write("This is a function!")
End Function
```

Functia “WriteThis” este apelata utilizand instructiunea Call, executa cele doua instructiuni *response.write* dupa care revine. Nu are nici un parametru deoarece nu necesita nici o valoare din afara functiei.

```
Call WhatColorisThis(8)
Sub WhatColorisThis(x)
        Select Case x
                Case 2
                        Response.write("Yellow")
                Case 4
                        Response.write("Blue")
                Case 6
                        Response.write("Red")
                Case 8
                        Response.write("Green")
End Sub
```

Prin apelarea acestei proceduri se va afisa pe pagina “Green”.

Primul script

Sa incercam in continuare sa scriem un script.

```
<%
@Language=VBScript
Option Explicit
Dim FinalResult
Dim Number1
Dim Number2
FinalResult=NumberstoMultiply(5,6)
Function NumberstoMultiply(Number1,Number2)
        NumberstoMultiply=5*6
End Function
Response.write ("Your Final Result is " & FinalResult & ".")
%>
```

FinalResult va fi 30. Dupa cum s-a vazut din exemplul anterior, metoda Response.write afiseaza un string urmat de variabila FinalResult. Acest lucru se face cu ajutorul operatorului “&” plasat intre string si variabila.

ASP-Modelul obiectual

Modelul Obiectual, conceput pentru tratarea cererilor si raspunsurilor in cadrul tehnologiei ASP, cuprinde 5 obiecte (in versiunea ASP 3.0 e introdus si ASPError).

Obiect	Folosit pentru
Request	A obtine informatii de la utilizator
Response	A trimite informatii catre utilizator
Session	A pastra informatii despre sesiunea unui utilizator
Application	A transmite informatii intre diferite pagini ale aplicatiei
Server	A accesa resursele serverului

Obiectul Request furnizeaza toate informatiile pe care clientul le ofera in momentul in care face o cerere pentru o pagina (este cazul folosirii controalelor dintr-un form).
Sintaxa :
Request[**.***colecte*|*proprietate*|*metoda*]**(***variabila***)**
Contine 5 colectii:

- ClientCertificate
Valorile tuturor campurilor si intrarilor certificarii clientului, pe care trebuie sa o prezinte in momentul accesarii unei resurse
- **Cookies**
Colectie a tuturor valorilor “cookie” trimise de sistemul client in aceasta cerere
- **Form**
Valorile tuturor elementelor de control continute in sectriunea <FORM> din care s-a facut cererea catre server.
- **QueryString**
Colectia tuturor perechilor de nume si valoare care apar in cererea clientului.
- **ServerVariables**
Valorile variabilelor de mediu

Proprietati:

TotalBytes	Read-only. Returneaza numarul total de bytes continuti in corpul cererii trimise de client.

Metode:

BinaryRead(“count”)	Intoarce “count” bytes din cererea clientului, cand datele sunt trimise la server, ca parte a unei cereri POST

Obiectul Response este folosit pentru a accesa raspunsul pe care il creeaza serverul pentru cererea clientului.

Sintaxa:
Response.*collection*|*property*|*method*

Colectii:

Cookies	:Specifica valorile cookie. Folosind aceasta colectie valorile cookie pot fi setate.

Proprietati:

Buffer	Precizeaza daca pagina create de server va fi tinuta in byffer-ul IIS pana cand toate script-urile server vor fi procesate sau pana cand va fi apelata una din metodele *Flush* sau *End*
CacheControl	Va fi setata ca *Public* daca se doreste ca serverul proxy sa intercepteze aceasta pagina, sau *Private* in caz contrar
Charset	Seteaza numele setului de caractere precizat in header-ul http
ContentType	Specifica tipul continutului http pentru raspuns (implicit este “text/xml”).
Expires	Specifica lungimea in minute a timpului de valabilitate a paginii.
ExpiresAbsolute	Specifica data si ora la care pagina va expira si nu va mai fi valida.
IsClientConnected	Indica daca clientul s-a deconectat de la server.
Pics	Creeaza un header PICS care va fi adaugat la header-ul http din raspuns pentru client.
Status	Specifica valoarea status-ului si mesajul care va fi

	trimis la client in header-ul http din raspuns pentru a indica aparitia unei erori sau terminarea cu success a procesarii paginii.

Metode:

AddHeader(nume,valoare)	Seteaza header-ul HTML *nume* la *valoare*.
AppendToLog(string)	Adauga un string la sfarsitul intrarii in log-ul server-ului Web, pentru cerinta clientului.
BinaryWrite(SafeArray)	Scrie continutul parametrului de tipul Variant, *SafeArray* in http-ul care va fi scris in pagina trimisa la client, in format binar.
Clear	Sterge continutul paginilor buffer-ate.
End	Opreste procesarea paginii ASP si intoarce continutul current.
Flush	Trimite continutul buffer-at al paginii.
Redirect(url)	Instruieste browser-ul sa incarce pagina definite de parametrul “url”.
Write(String)	Scrie continutul parametrului String in stream-ul de raspuns http si in buffer-ul http, care va fi continut in pagina de raspuns trimisa la client

Obiectul Application este creat in momentul in care este incarcat pentru prima data fisierul asp.dll pentru a crea raspuns la o cerere a unui client si va contine date accesibile tuturor clientilor site-ului.

Sintaxa:
Application.*method*

Colectii:

Contents	Colectia tuturor varabilelor care sunt definite in obiectul *Application*, fara a se folosi elemental HTML <OBJECT>.
StaticObjects	Colectia tuturor variabilelor create in obiectul *Application* folosind elemental HTML <OBJECT>.

Metode:

Lock	Previne modificarea proprietatilor

obiectului *Application* de catre alti clienti.

Unlock
Permite altor clienti sa modifice proprietatile obiectului *Application*.

Evenimente:

Application onEnd	Se produce cand aplicatia e oprita
Application onStart	Se produce cand aplicatia ASP porneste, inainte ca pagina ceruta de primul client sa se creeze.

Obiectul Session este creat de fiecare data cand un client cere prima sa pagina si ramane disponibil pentru acesta pana in momentul in care va parasi aplicatia respectiva, deci pe timpul executiei sesiunii create intre el si server.

Sintaxa:
Session.*collection*|*property*|*method*

Colectii:

Contents	Colectia tuturor varabilelor care sunt definite in obiectul *Session*, fara a se folosi elemental HTML <OBJECT>.
StaticObjects	Colectia tuturor variabilelor create in obiectul *Session* folosind elemental HTML <OBJECT>.

Proprietati:
CodePage
Defineste un cod de pagina folosit pentru a afisa continutul paginii in browser-ul clientului
LCID
Defineste un identificator local unic pentru pagina trimisa la client. **SessionID**
Returneaza identificatorul pentru sesiunea respective, care a fost generat de server. **Timeout**
Defineste perioada in minute, la care va expira obiectul *Session* .

Metode:

Abandon Distruge obiectul *Session* si ii elibereaza resursele.

Evenimente:

Session onEnd Se produce cand sesiunea e oprita, la safarsitul comunicarii cu clientul respectiv.

Session onStart Se produce cand sesiunea porneste, inainte ca serverul sa inceapa sa trateze cererea clientului care genereaza aceasta sesiune

Obiectul Server este conceput pentru a realiza anumite operatii pe server, mai ales relativ la mediul de lucru al aplicatiei respective si la procesele sale.

Sintaxa:

Server.*property*|*method*

Proprietati:

ScriptTimeout Permite setarea sau returnarea intervalului maxim de executie pe care il acorda serverul unui script.

Metode:

CreateObject(identificator) Creeaza o instanta a obiectului recunoscut prin “identificator” si returneaza o referinta la el.

HTMLEncode(string) Converteste in format HTML stringul specificat ca parametru.

MapPath(url) Reda calea fizica a unui fisier sau resursa, specificat in “url”.

URLEncode(string) Converteste stringul specificat ca parametu (cu caractere care nu sunt intr-o adresa valida: ‘?’,’&’) in coduri care sa fie echivalente pentru un URL(“%3F”,”%26”).

ASP .NET

ASP .NET este urmatorul pas evolutiv al platformei ASP, fiind insa mult mai mult decat un upgrade al unei versiuni deja mature (3.0). Este cea mai avansata platforma de dezvoltare Web existenta la ora actuala, fiind rescrisa de la zero in concordanta cu platforma .Net Framework pe care ruleaza. Schimbarile aduse de ASP.NET fata de predecesorul sau, ASP 3.0, sunt semnificative. In acelasi timp, mergand pe ideea „compatibilitatii inapoi" (backword compatibility), ASP.NET contine toate obiectele de programare existente in versiunile anterioare de ASP, cum ar fi (foarte des folositele) *Request* şi *Response*, *Application, Session* şi *Server*. Programatorul, care s-a obişnuit cu maniera de lucru din versiunile anterioare, poate folosi în continuare tag-urile „<% %>" sau „<script runat=server>", pentru a delimita codul care trebuie să fie rulat pe server de pagina statică. De altfel, ASP.NET a fost proiectat astfel incat migrarea spre noua tehnologie sa se faca cat mai usor posibil, cu posibilitatea rularii in paralel a vechilor script-uri ASP cu cele noi (ASP.NET).

ASP.NET suporta programarea orientata pe obiecte in adevaratul sens al cuvantului, incluzand mostenirea, polimorfismul si incapsularea. Se introduce modelul de dezvoltare 3-tier (nivelul baza de date, nivelul business logic si nivelul prezentare). In ASP Clasic deseori cele 3 nivele se amesteca si acest lucru conduce la dezvoltarea unor solutii ineficiente cu arhitectura voluminoasa. Spre deosebire de ASP Clasic, in .NET nivelele logic, prezentare si baza de date sunt separate. Nivelul prezentare cuprinde paginile aspx si partea logica cuprinde „codul din spatele paginilor". Este permisa chiar folosirea stilului de codificare ASP Clasic, desi nu e indicat.

Printre facilitatile introduse de noua platforma se numara urmatoarele:

- **un suport mai bun pentru diferite limbaje de programare**: ASP .NET suporta Visual Basic (nu VBScript), C#, C++, JScript si foloseste noul ADO .NET; spre deosebire de ASP Clasic, nu putem utiliza mai multe limbaje in aceeasi pagina. Putem folosi insa limbaje diferite in pagini diferite ale aplicatiei, deoarece codul este compilat in IL (Intermidiate Language) indiferent de alegerea facuta.
- **formulare web** (Web Forms): combina facilitatile puternice ale ASP-ului cu usurinta in dezvoltare de interfete grafice si productivitatea Visual Basic-ului. Puteti sa faceti drag-and-drop de componente pe o pagina, iar apoi sa introduceti cod care sa asigure interactivitatea dintre componente;
- **controale de tip server** (server controls): sunt componente care pot fi utlizate în formulare web si care mapeaza intern tag-uri şi elemente de tip HTML, cum ar fi de exemplu componenta TextBox care are ca si corespondent HTML un tag

de tipul <input type=textbox>, sau componenta Button cu corespondentul <input type=submit>. Aceste controale, dupa cum le spune si numele, ruleaza pe server si genereaza cod HTML, care apoi poate fi vizualizat pe orice browser Internet compatibil cu HTML versiunea 3.2 (Internet Explorer 5.x sau mai nou, Netscape, Opera etc.);

- **programare orientata pe evenimente**: toate obiectele ASP .NET de pe o pagina web pot avea atasate evenimente tratate prin cod. Sa luam exemplul controalelor web. In ASP clasic, un text box se defineste in HTML in modul urmator:

<input type=”text” value=”Submit” name=”txtBox” >

In .NET, cu ajutorul controalelor web, va fi definti astfel:

<asp:textbox id=”txtBox” runat=”server” text=”Submit”></asp:textbox>

Diferenta consta in faptul ca aceste controale sunt orientate pe evenimente. Ele pot sa refere evenimente care vor fi procesate de codul .NET. Prin evenimente precum Load, Change, Click, IndexChange se obtine un cod mult mai simplu si mai bine organizat, la fel cum s-ar intampla intr-un Form Windows sau o aplicatie Visual Basic sau Delphi. In plus, controalele web au o sintaxa mult mai simpla fata de traditionalul HTML.

- **Servicii Web** (web services): acestea reprezinta o componenta cheie a platformei ASP.NET, si ii dau posibilitatea programatorului sa creeze niste librarii de functii (servicii), accesibile altor utilizatori/programatori prin intermediul retelelor (local intranet, internet). Serviciile web se bazeaza pe un protocol de comunicatie in curs de standardizare, independent de platforma, numit SOAP (Simple Object Access Protocol), care asigura un nivel ridicat de interoperabilitate cross-platform;
- **caching**: ASP.NET conţine un motor de caching imbunatatit, care ii da posibilitatea programatorului sa decida la nivel de componenta ce informatii vor fi pastrate in cache, cat timp vor fi stocate etc;
- **performante mai bune la executie:** codul de nivel înalt este verificat la compilare (paginile ASP se scriu de obicei în VBScript sau JScript, verificat la rulare); prima cerere a unei pagini ASP .NET va compila codul corespunzator pe server si il va memora in cache determinand astfel o crestere a performantei;
- i**mbunatatiri legate de configurarea paginilor ASP**: spre deosebire de versiunile anterioare, in care informatiile de configurare legate de paginile ASP erau tinute de catre serverul web IIS, intr-o baza de date interna greu accesibila (in format binar proprietar), la aceasta versiune, toate informatiile sunt mentinute in fisiere de configurare XML (format text), foarte usor de citit si de modificat (din orice editor de text, chiar şi din Notepad); fisierele de configurare pot fi incarcate sau schimbate in timp ce aplicatia se executa, nefiind necesara repornirea serverului;

- s**ecuritate imbunatatita**: diverse tipuri de autentificare (windows, forms, passport), autorizare la nivel de sistem de fisiere NTFS prin intermediul ACL-urilor (Access Control Lists) sau la nivel de URL;
- suport pentru aplicatii mobile: începând cu versiunea 1.1 a ASP.NET a fost adaugat suport pentru crearea de aplicatii care sa ruleze pe sisteme de calcul mobile ca PDA-uri (Personal Digital Assistants) si telefoane mobile.
- **compatibilitate cu ASP Clasic:** ASP.NET a fost proiectat sa lucreze alaturi de ASP; ASP.NET nu e perfect compatibil cu versiunile mai vechi de ASP, deci cea mai mare parte a codului ASP va necesita o serie de schimbari pentru a fi executat in ASP.NET. Pentru a solutiona aceasta problema, ASP.NET foloseste o noua extensie pentru fisiere „.aspx". Aceasta va face ca aplicatiile ASP.NET sa functioneze impreuna cu aplicatii ASP standard pe acelasi server.

Diferente de implementare

Una din cele mai mari diferente intre ASP si ASP.NET in ceea ce priveste modul de implementare consta in faptul ca functiile ASP nu vor mai fi intercalate cu cod HTML. Unul dintre punctele tari ale ASP este posibilitatea de a introduce cu usurinta functii in pagini HTML existente fara a modifica prea mult codul. ASP.NET cere ca orice HTML ce apare printre functii sa fie afisat utilizand functia *response.write*.

ASP:

```
<%
Option Explicit
Function PrintHello
Dim i
For i= 1 to 5
%> <font size=<%=i%>>Hello</font>> <%
Next
End Function
%>
```

ASP.NET:

```
<%
Option Explicit
Function PrintHello()
Dim i
For i= 1 to 5
response.write("<font size=" & i & "> Hello")
<%
Next
End Function
```

```
%>
```

Cand declaram o functie in ASP.NET trebuie inclusa intr-un bloc *<script runat=server>* cu specificarea limbajului folosit. Nu mai sunt necesare tag-urile <% %>. Trebuie sa avem grija ca functiile noastre sa existe in aceste blocuri, altfel ele vor genera eroare.

ASP:

```
<%
Sub HelloWrite()
Response.Write "Hello World"
End Sub
HelloWrite
%>
```

ASP.NET:

```
<script language="VB" runat=server>
Sub HelloWrite()
Response.Write "Hello World"
End Sub
</script>
<%
HelloWrite()
%>
```

O alta diferenta care se poate observa ar fi necesitatea parantezelor () la apelul unei functii.

In ASP de fiecare data cand este creat un obiect sau trebuie facuta o conexiune la o baza de date, se aloca o variabila folosind “Set”. In ASP.NET nu mai este necesar ca variabilele sa fie setate, fiind suficienta operatia de atribuire. Aceasta regula se aplica si pentru “Let”.

ASP:

```
<%
Dim Connection
Set Connection = Server.CreateObject("ADODB.Connection")
%>
```

ASP.NET:

```
<%
Dim Connection
Connection = Server.CreateObject("ADODB.Connection")
%>
```

La anumite componente, cum ar fi ADO, exista o proprietate “default” care putea sa fie referita doar prin numele obiectului respectiv. ASP.NET este un

mediu mai structurat in care acest lucru nu este permis.

ASP:

```
<%
Connection = Server.CreateObject("ADODB.Connection")
Connection .Open("MyDatabase")
RS = Connection .Execute("Select * from Operations")
Response.Write(RS("ID"))
%>
```

ASP.NET:

```
<%
Connection = Server.CreateObject("ADODB.Connection")
Connection .Open("MyDatabase")
RS = Connection .Execute("Select * from Operations")
Response.Write(RS("ID").Value)
%>
```

In exemplul de mai sus proprietatea “Value” trebuie sa fie indicata in mod explicit, in caz contrar nu va fi recunoscuta valoarea ce trebuie afisata si se va genera o eroare.

Bibliografie

www.minwar.com/Learn/ASP/

http://msdn.microsoft.com/library/enus/dnanchor/html/activeservpages.asp

www.tutorial-web.com/asp.net/

http://ms.utt.ro/microsoft/elearning/courses.aspx

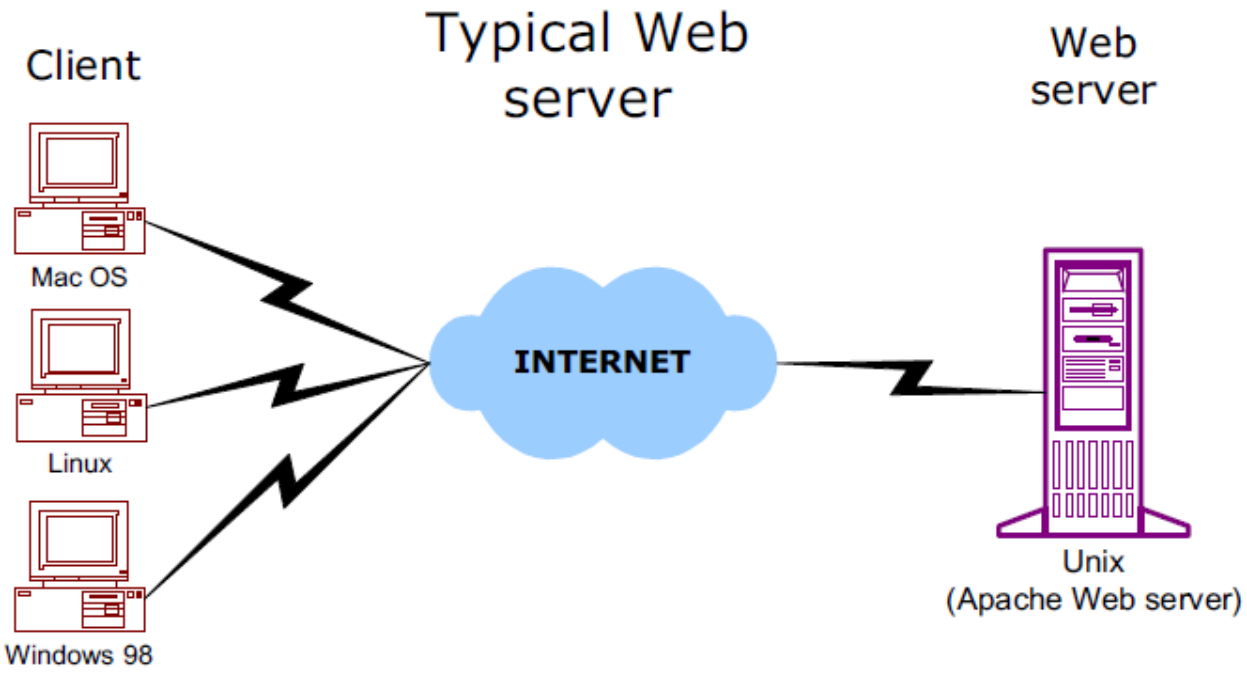
www.w3schools.com/aspnet/aspnet_vsasp.asp

Java Server Pages

Introducere

JavaServer Pages (JSP) este o tehnologie bazata pe limbajul Java si permite implementarea site-urilor web dinamice. JSP a fost creat de Sun Microsystems pentru dezvoltarea aplicatiilor si serviciilor web pe server. Fisierele JSP sunt fisiere HTML cu tag-uri speciale care contin coduri sursa Java ce genereaza continut dinamic.

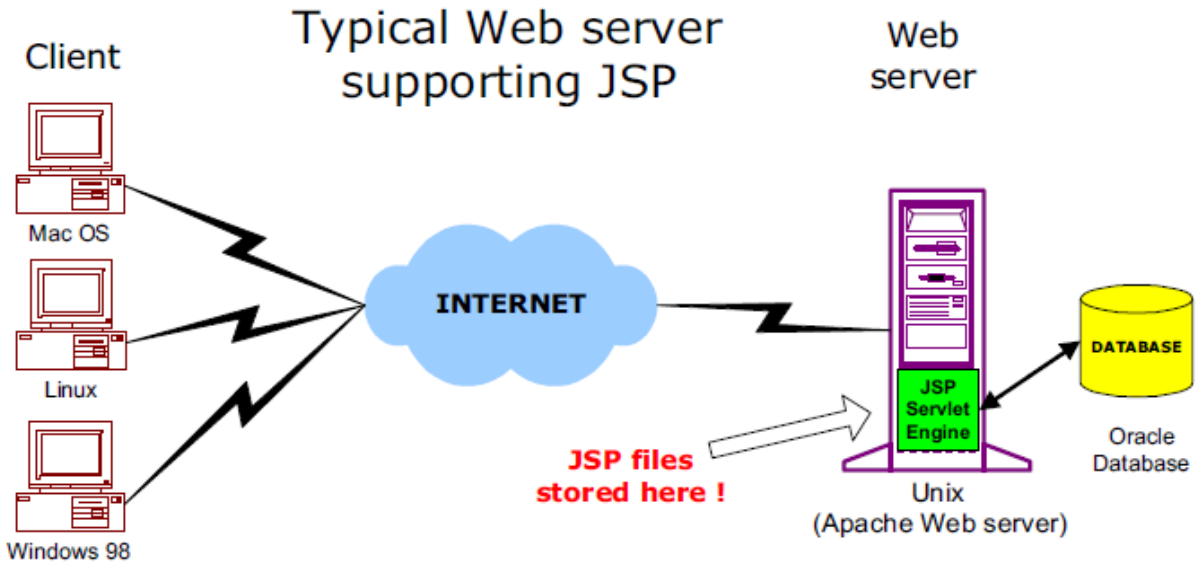
Urmatoarea figura prezinta web server-ul “tipic”, mai multi clienti conectandu-se prin Internet la server. In acest exemplu, web server-ul ruleaza pe Unix si este popularul web server Apache.



Au aparut primele pagini web statice. Aceasta a fost prima experienta a oamenilor in a face pagini web ce constau in site-uri My Home Page. Apoi au fost folosite limbajele Perl si C pentru a genera continut dinamic. La scurt timp, cele mai multe limbaje (inclusiv Visualbasic, Delphi, C++, Java) au putut fi folosite pentru a scrie aplicatii care dau continut dinamic folosind date din fisiere text sau cereri din baze de date. Acestea erau cunoscute ca aplicatii pe partea de server CGI. ASP a fost creat de Microsoft pentru a permite dezvoltatorilor HTML generarea usoara de continut dinamic avand ca suport standard server-ul

web gratuit al Microsoft, IIS (Internet Information Server). JSP este echivalentul de la Sun Microsystems.

Urmatoarea diagrama ne arata un web server care accepta fisierele JSP. Observam ca server-ul este conectat la o baza de date.



Codul sursa JSP ruleaza pe web server in JSP Servlet Engine. Motorul JSP Servlet genereaza dinamic HTML-ul si trimite semnalul HTML pentru browser-ul clientului.

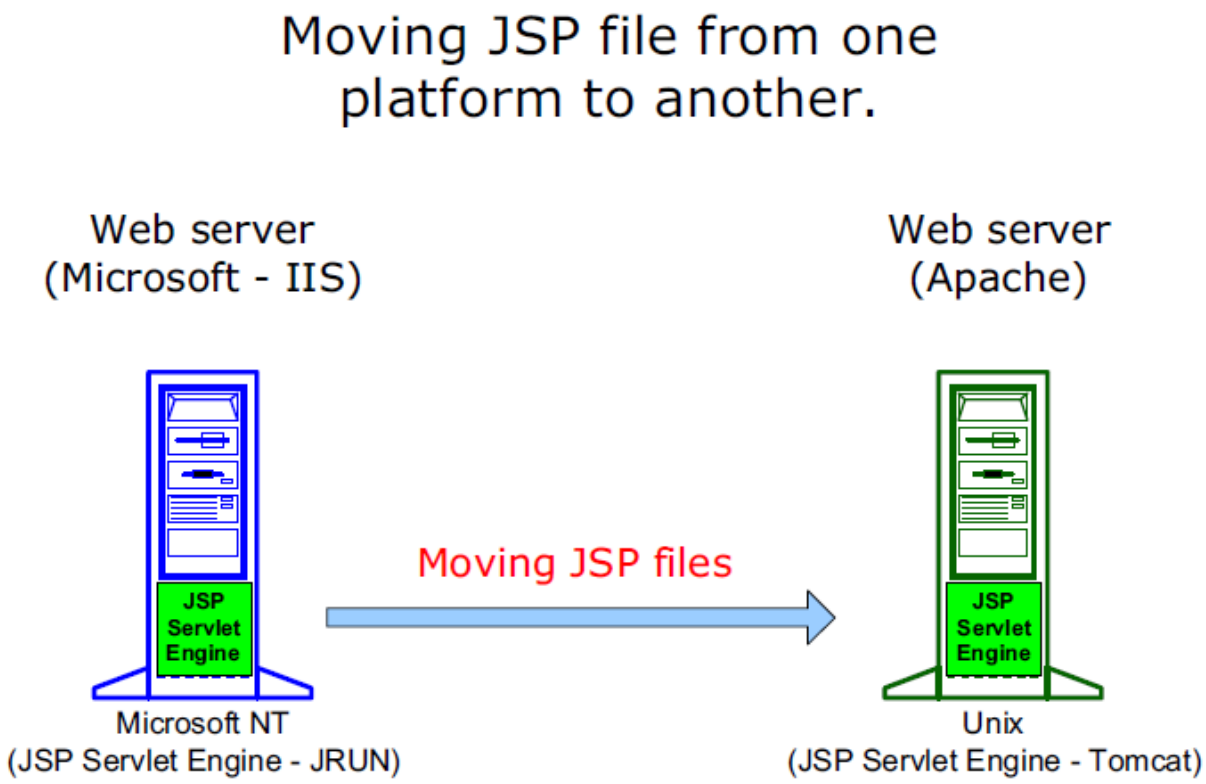
De ce folosim JSP

JSP este usor de invatat si permite dezvoltatorilor producerea rapida a site-urilor web si a aplicatiilor intr-un mod deschis. JSP este bazat pe Java, un limbaj orientat pe obiecte. JSP ofera o platforma “stufoasa” pentru dezvoltarea web.

Principalele motive pentru a utilize JSP:

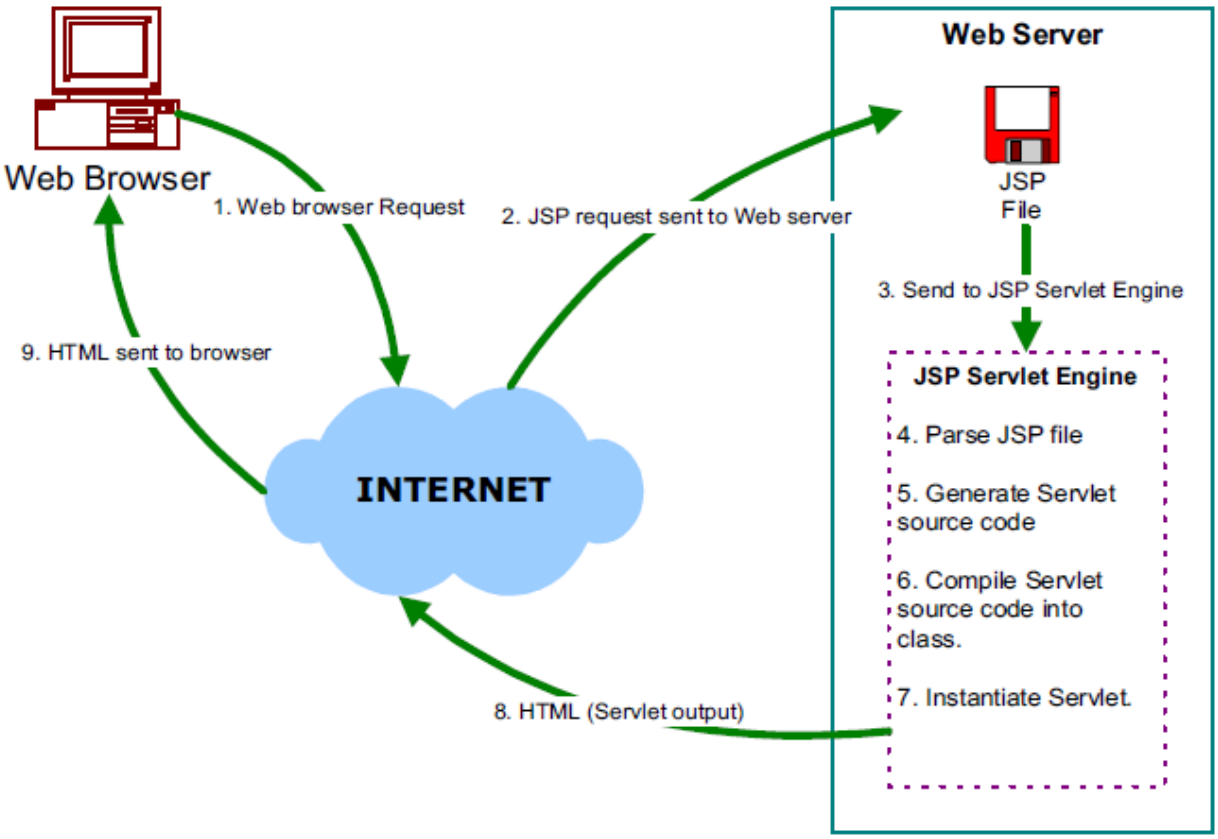
- platforma multipla
- reutilizarea componentei folosind Javabeans si EJB
- avantajele Java

Putem muta un fisier JSP pe o alta platforma, web server sau pe un motor JSP Servlet.



Arhitectura JSP

JSPs sunt construite cu tehnologia servlet de varf a SUN. Au la baza pagini HTML cu tag-uri speciale JSP incorporate. Tag-urile JSP pot contine cod Java. Extensia fisierelor JSP este ".jsp". Motorul JSP creaza fisierul sursa pe care il compileaza in fisierul clasa; acesta este mai lent la prima rulare si devine mai rapid.



Etapele necesare pentru o cerere JSP

1. User-ul intra pe un web site realizat cu JSP. Intra pe pagina JSP (cea cu extensia .jsp). Browser-ul trimite cererea prin Internet.
2. Cererea JSP este trimisa catre web server.
3. Web server-ul recunoaste fisierul special (.jsp) trimis, prin urmare trimite fisierul JSP catre JSP Servlet Engine.
4. Daca fisierul JSP a fost apelat prima data, acum este analizat, in caz contrar se trece la pasul 7.
5. Urmatorul pas este generarea unui Servlet special din fisierul JSP. Toate documentele HTML solicitate sunt convertite in comenzi println.
6. Codul sursa Servlet este compilat intr-o clasa.
7. Servlet este instantiate, apeland metodele “init” si “service”.
8. Codul HTML de la iesire este trimis prin Internet.
9. Rezultatele sunt afisate de catre browser-ul user-ului.

Crearea primei pagini JSP

Urmatorul cod are ca rezultat afisarea mesajului “Hello World” in browser:

```
<html>
<head>
<title> My first JSP page
</title>
</head>
<body>
<%@ page language=”java” %>
<% System.out.println(“Hello World”); %>
</body>
</html>
```

Tag-uri JSP

Exista 5 tipuri de tag-uri: pentru declaratii, pentru expresii, directive, scriptlet-uri si actiuni.

1.Tag-urile pentru declaratii permit declararea variabilelor si a metodelor. Declaratiile sunt scrise intre “<%!” si “%>”. Fiecare linie de cod se termina cu “;”.

Exemplu:

```
<%!
        private int counter=0;
        private String get Account (int  accountNo);
%>
```

2.Tag-urile pentru expresii permit inglobarea expresiilor Java si reprezinta o prescurtare pentru “out.println()”. Liniile de cod din interiorul tag-ului nu se termina cu “;”.

Exemplu: afisarea datei si a orei:

```
Date:<%=new java.util.Date()%>
```

3.Directivele ofera posibilitatea de adaugare de informatii aditionale si pentru descrierea atributelor paginii. De exemplu, directivele sunt utilizate pentru importarea pachetelor Java, includerea fisierelor si pentru accesarea librariilor de tag-uri definite de utilizator. Sintaxa generala pentru directive este:

<%@ directive […]%>

Cea mai utilizata directiva este directiva “page”. Aceasta poate fi utilizata pentru a importa clase si pachete Java. Directiva “page” include atributele din urmatorul tabel:

Atribut	Valoare	Valoare implicită	Descriere
Language	java	java	specifică limbajul de programare
Extends	superclasa	depinde de platformă	indică superclasa pentru clasa servletului care se va genera
Import	listă de pachete separate prin virgulă	java.lang.*, javax.servlet.http.*, javax.servlet.*, javax.servlet.jsp.*	importă lista de pachete şi/sau clase specificate
Session	true | false	true	indică dacă se stabileşte o sesiune
Buffer	dimensiunea în kilo-octeţi	8 sau mai mult	stabileşte dimensiunea buffer-ului
autoFlush	true | false	true	indică dacă se goleşte automat buffer-ul
isThreadSafe	true | false	true	indică dacă poate fi accesată pagina de către mai multe fire de execuţie simultan
Info	text	şirul vid	specifică informaţii despre pagina JSP
errorPage	un URL	nimic	indică URL-ul

			paginii care va fi transmisă utilizatorului în caz de eroare
isErrorPage	true | false	false	indică dacă este pagina pentru erori
contentType	tipul MIME	text/html	specifică tipul documentului returnat

Alte directive JSP sunt directiva “include” si directiva “taglib”.

Directiva “include” permite includerea continutului unui fisier in alt fisier.
Directiva “taglib” este o colectie de tag-uri ce poate fi utilizata de pagina web. Exemplu:

<%@taglib uri=”tag library URI” prefix=”tag Prefix”%>

4.Scriptlet-uri: pana acum am vazut cum putem declara date si functii member, precum si inserarea expresiilor. Scriptlet-urile permit adaugarea de cod Java imbinat cu secvente HTML. Sintaxa generala este urmatoarea:

<% cod_Java %>

respectiv

<jsp:scriptlet> cod_Java </jsp:scriptlet>

5.Actiunile sunt tag-uri particulare predefinite. Acestea nu au corespondent un tag specific JSP (cele care incep cu <%), ci sunt numai tag-uri XML cu spatiul de nume jsp. Actiunile confera un nivel inalt de functionalitate fata de declaratii, expresii si scriptlet-uri.
Exista trei categorii de actiuni standard:
-cele utilizate pentru component Bean
-cele pentru controlul din momentul executiei, cum ar fi redirectarea sau includerea
-cele care ofera suport pentru plug-in-un Java.

Integrarea componentelor JavaBean

Tehnologia JSP ofera o integrare foarte buna a formularelor HTML cu componentele JavaBean. Pentru utilizarea unei componente JavaBean avem la dispozitie actiuni.
Actiunea jsp:useBean se poate scrie ca un tag de sine statator (fara continut):

<jsp:useBean id=”nume” scope=”scop” Specificare/>

sau ca tag cu continut:

<jsp:useBean id=”nume” scope=”scop” Specificare/>
corp_pentru_creare
</jsp:useBean>

Acesta duce la crearea unui obiect corespunzator componentei; poate avea valorile din tabelul urmator:

scope	Durata existenţei componentei
page	Componenta este disponibilă doar în pagina JSP curentă şi va fi recreată la fiecare cerere.
request	Componenta va fi disponibilă pe tot parcursul cererii, inclusiv în paginile incluse sau redirectate
session	Componenta este disponibilă pe tot parcursul sesiunii stabilite
application	Componenta va fi disponibilă pentru toate sesiunile şi îşi va înceta existenţa o dată cu aplicaţia Web

Elementul Specificare este destul de flexibil si ofera o varietate de optiuni care sunt prezentate in tabelul de mai jos:

Opţiuni	Semnificaţie
class=”numeClasa”	Se specifică clasa corespunzătoare componentei(numeClasa)
type=”numeTip”	Se indică tipul care se va utiliza în pagină pentru componentă(numeTip),care trebuie să fie compatibil cu

	tipul clasei. Se indică şi numele clasei.
type=”numeTip” beanName=”numeComp”	Se indică tipul care se va utiliza în pagină pentru componentă(numeTip), precum şi numele componentei. Acesta trebuie să fie furnizat prin intermediul unei expresii JSP dat în forma JSP(cu simbolurile < %).
type=”numeTip”	Se stabileşte tipul componentei care va fi utilizat în cadrul paginii JSP.

Sintaxa actiunii jsp:setProperty este urmatoarea:
<jsp:setProperty name=”numeComp” extensie/>
Numele componentei este cel stabilit de atributul id al actiunii jsp:useBean.

Comentarii JSP

Comentariile JSP sunt similare cu cele HTML, dar nu sunt trimise catre browser-ul utilizatorului. Comentariile HTML sunt vizibile in pagina sursa.

```
<html>
<head>
<title>
    Comentarii JSP si HTML
</title>
</head>
<body>
<h2>
    Comentarii
</h2>
<!--comentariu HTML-vizibil in pagina sursa-- >
<%--comentariu JSP-invizibil in pagina sursa--%>
</body>
</html>
```

Bibliografie

pentru: -introducere in JSP; -de ce folosim JSP; -arhitectura JSP; -etapele necesare pt o cerere JSP; -crearea primei pagini JSP;

http://www.visualbuilder.com/

pentru: -tag-uri JSP; -integrarea componentelor JavaBean;

"Java de la 0 la expert", Editura Polirom

pentru: -comentarii JSP;

http://www.visualbuilder.com/

Limbajul PHP

Generalitati

PHP este un limbaj de scriptare server-side. Numele PHP provine din limba engleză şi este un acronim recursiv : Hypertext Preprocessor. Folosit iniţial pentru a produce pagini web dinamice, este folosit pe scară largă în dezvoltarea paginilor şi aplicaţiilor web. Se foloseşte în principal înglobat în codul HTML, dar începând de la versiunea 4.3.0 se poate folosi şi în mod „linie de comandă” (CLI), permiţând crearea de aplicaţii independente. Este unul din cele mai importante limbaje de programare web open-source şi server-side, existând versiuni disponibile pentru majoritatea web serverelor şi pentru toate sistemele de operare. Conform statisticilor este instalat pe 20 de milioane de situri web şi pe 1 milion de servere web. Este disponibil sub Licența PHP și Free Software Foundation îl considerã a fi un software liber.

Iniţial, limbajul a fost dezvoltat de inventatorul său, Rasmus Lerdorf. Odată cu creşterea numărului de utilizatori, dezvoltarea a fost preluată de o nouă entitate, numită The PHP Group (Grupul PHP).

Cum a aparut

PHP a însemnat iniţial Personal Home Page. PHP a fost început în 1994 ca o extensie a limbajului server-side Perl, şi apoi ca o serie de CGI-uri compilate de către Rasmus Lerdorf, pentru a genera un curriculum vitae şi pentru a urmări numărul de vizitatori ai unui site. Apoi a evoluat în PHP/FI 2.0, dar proiectul open-source a început să ia amploare după ce Zeev Suraski şi Andi Gutmans, de la Technion au lansat o nouă versiune a interpretorului PHP în vara anului 1998, această versiune primind numele de PHP 3.0. Tot ei au schimbat şi numele în acronimul recursiv de acum, până atunci PHP fiind cunoscut ca Personal Home Page Tools. Apoi Suraski şi Gutmans au rescris baza limbajului, producând astfel şi Zend Engine în 1999. În mai 2000 a fost lansat PHP 4.0, având la bază Zend Engine 1.0.

PHP 5

Pe 13 iulie 2004 a fost lansat PHP 5, cu Zend Engine II, ce a adus şi o orientare obiect mai pronunţată şi suportând mai multe caracteristici ale acestui tip de programare.

PHP 5 aduce mai multe noutăţi faţă de versiunea 4:

* Suport îmbunătăţit pentru OOP
* Introduce extensia PDO - PHP Data Objects, care defineşte o modalitate facilă şi consistentă de accesare a diferitelor baze de date
* Imbunătăţiri de performanţă
* Suport îmbunătăţit pentru MySQL şi MSSQL
* Suport nativ pentru SQLite
* Suport SOAP integrat
* Iteratori pentru date
* Controlul erorilor prin tratarea de excepţii

La sfârşitul lui 2007 doar versiunea 5.x mai era întreţinută, deoarece în data de 13 iulie 2007 (exact la 3 ani după lansarea PHP5), PHP Group a anunţat că PHP4 va fi scos din uz pe 31 decembrie 2007, deşi prognozează că anumite upgrade-uri de securitate se vor oferi până pe 8 august 2008.[4]. Dezvoltarea la PHP 6 începuse deja în decembrie 2007 şi urmează să fie oferit odată cu scoaterea din uz a PHP4.
[modifică] PHP 6

PHP 6 are următoarea agendă de îmbunătăţiri şi modificări:

* îmbunătăţirea suportului pentru Unicode
* retragerea definitivă a unor funcţii ca register_globals şi magic_quotes, şi a variabilelor tip $HTTP_*_VARS
* var va fi un alias pentru public, şi folosirea lui va ridica o atenţionare E_STRICT.
* suport pentru int pe 64 biti.
* taguri tip ASP sunt retrase definitiv.
* XMLReader, XMLWriter, Fileinfo vor face parte din distribuţia principală
* urmatoarele pachete au fost scoase din distribuţia principală: Freetype1, GD1, mime_magic
* funcţia ereg() nu mai este disponibilă

* instanţierea obiectelor prin referină (& new Obiect()) generează o eroare E_STRICT.
* erorile tip E_STRICT sunt incluse în E_ALL.
* adaugarea instrucţinuii goto permite salturi la un alt bloc de comenzi.
* namespace, import, şi goto devin cuvinte rezervate.
* accesarea caracterelor într-un şir (string) se face prin operatorul []. {} se scoate din uz (ex: $str[42] funcţionează, $str{42} nu funcţionează)
* constantele FILE_BINARY şi FILE_TEXT devin disponibile pentru folosirea in funcţii de citire/scriere fişiere
* foreach va suporta array multi dimensional: foreach($a as $b => list($c, $d))
* pentru operatorul ternar expresia pentru valoarea true nu mai este obligatoriu ($a = $s ?: 'b'; // returns $a = $s;)
* opţiunea safe_mode a fost înlăturată.
* operatorul and a fost înlăturat.
* funcţia microtime() returnează un float.
* zend.ze1_compatibility_mode a fost înlăturat.

Caracteristici

PHP-ul este unul din cele mai folosite limbaje de programare server-side, conform unui studiu efectuat de Netcraft în aprilie 2002, apărând pe 9 din cele 37 milioane de domenii cercetate în studiu. De asemenea, există un grafic al creşterii folosirii PHP-ului pe site-ul oficial. Popularitatea de care se bucură acest limbaj de programare se datorează următoarelor caracteristici :

* Familiaritatea : sintaxa limbajului este foarte uşoară combinând sintaxele unora din cele mai populare limbaje Perl sau C;

* Simplitatea : sintaxa limbajului este destul de liberă. Nu este nevoie de includere de biblioteci sau de directive de compilare, codul PHP inclus într-un document executându-se între marcajele speciale;

* Eficienţa : PHP-ul se foloseşte de mecanisme de alocare a resurselor, foarte necesare unui mediu multiutilizator, aşa cum este web-ul;

* Securitate : PHP-ul pune la dispoziţia programatorului un set flexibil şi eficient de măsuri de siguranţă;

* Flexibilitate : fiind apărut din necesitatea dezvoltării web-ului, PHP a fost modularizat pentru a ţine pasul cu dezvoltarea diferitelor tehnologii. Nefiind legat de un anumit server web, PHP-ul a fost integrat pentru numeroasele servere web existente: Apache, IIS, Zeus, server, etc.;

* Gratuitate : este probabil cea mai importantă caracteristică a PHP-ului. Dezvoltarea PHP-ului sub licenţa open-source a determinat adaptarea rapidă a PHP-ului la nevoile web-ului, eficientizarea şi securizarea codului.

PHP este simplu de utilizat, fiind un limbaj de programare structurat, ca şi C-ul, Perl-ul sau începând de la versiunea 5 chiar Java, sintaxa limbajului fiind o combinaţie a celor trei. Datorită modularităţii sale poate fi folosit şi pentru a dezvolta aplicaţii de sine stătătorare, de exemplu în combinaţie cu PHP-GTK sau poate fi folosit ca Perl sau Python în linia de comandă. Probabil una din cele mai importante facilităţi ale limbajului este conlucrarea cu majoritatea bazelor de date relaţionale, de la MySQL şi până la Oracle, trecând prin MS Sql Server, PostgreSQL, sau DB2.

PHP poate rula pe majoritatea sistemelor de operare, de la UNIX, Linux, Windows, sau Mac OS X şi poate interacţiona cu majoritatea serverelor web. Codul dumneavoastră PHP este interpretat de serverul WEB şi generează un cod HTML care va fi văzut de utilizator (clientului -browserului- fiindu-i transmis numai cod HTML).

Arhitectura tip LAMP a devenit populară în industria web ca modalitate rapidă, gratuită şi integrată de dezvoltare a aplicaţiilor. Alături de Linux, Apache şi Mysql, PHP reprezintă litera P, deşi unori aceasta se referă la Python sau Perl. Linux ocupă rolul de sistem de operare pentru toate celelalte aplicaţii, Mysql gestionază bazele de date, Apache are rol de server web, iar PHP are rol de interpretator şi comunicator între acestea.

PHP foloseste extensii specifice pentru fişierele sale: .php, .php3, .ph3, .php4, .inc, .phtml. Aceste fişiere sunt interpretate de catre serverul web iar rezultatul este trimis în formă de text sau cod HTML către browser-ul clientului.

Tipuri de date si functii

1. *Boolean* - valori logice tip adevărat sau false, similare cu cele din C++ sau Perl.
2. *Integer* - numere întregi (în baza 10, 2 sau 16). Valoarea maximă depinde de sistem şi de tipul de integer. Tipul poate fi "signed" sau "unsigned", adica dependent de semnul + sau - sau independent de acestea. Valorile pentru integer unsigned sunt mai mari decât cele pentru signed. Sistemele pe 32 bits pot crea numere întregi între -2147483648 şi 2147483647(2^32-1). Maximul pentru sisteme pe 64 bits este 9223372036854775807.
3. *Float* - cunoscute ca numere reale. Valorile maxime sunt de asemenea dependete de platform, in general cu un maxim de ~1.8e308 cu o precizie de 14 zecimale dupa virgula (formatul 64 bits IEEE).
4. *String* - şiruri de caractere. Înainte de PHP6, un caracter era echivalent cu un byte. Nu există limitări pentru lungimea unui şir, în afara memoriei alocate PHP.
5. *Array* - în PHP un array este un tip de data care conţine un grup de elemente. Fiecare element are un indice intern în group, iar fiecărui indice îi corespunde o valoare - elementul în sine. Un astfel de grup poate fi folosit ca o simulare pentru diverse situaţii matematice precum vectori, serii, dicţionare de elemente, liste ordonate, matrici sau matrici de matrici. Indicii şi valorile unui grup pot fi orice tip de data interna PHP (cu excepţii: obiectele, resursele şi null nu pot fi indici).
6. *Obiecte* - O clasa este o colecţie de proprietăţi şi funcţii având o logică comună. Obiectele sunt instanţe ale unei clase, în care proprietăţile obiectului primesc valori specifice. Vezi POO - programarea orientată pe obiecte.
7. *Resurse* - aceastea sunt variabile speciale care conţin legături cu resurse externe PHP. De exemplu, conexiunea cu o bază de date este o resursa deschisă şi menţinută cu ajutorul unor funcţii special definite pentru aceasta muncă.
8. *NULL* - este un tip special de dată, care semnifică că variabila respectivă nu a fost definită şi că nu are valoare.

PHP are sute de funcţii incorporate şi alte câteva mii disponibile prin intermediul extensiilor. Clasicul program *hello-world* în PHP:

```
<?php
 echo "Hello world!";
?>
```

Versiunile 5.2 şi anterioare

În aceste versini funcţiile nu sunt obiecte de prima clasă. Aceasta înseamnă că funcţiile nu pot fi create dinamic în timpul executării programului şi ca pot fi chemate doar prin numele dat cand au fost definite. Utilizatorul poate crea funcţii în orice moment în program. În acest exemplu cuvântul cheie function defineşte funcţia cu numele adauga care primeşte un număr de 2 parametrii de intrare şi returnează suma acestora.

```
function suma($x, $y)
{
    return $x + $y;
}
```

echo adauga(2, 4); // returnează 6

Versiunile 5.3 şi mai noi

PHP are suport pentru funcţii de rangul întâi şi pentru funcţii anonime, precum cele folosite în Javascript.

```
function getAdder($x)
{
    return function ($y) use ($x) {
        // or: lexical $x;
        return $x + $y;
    };
}
```

Programarea Orientata pe Obiecte

Funcţionalităţi bazice de programare orientată pe obiecte au fost adăugate în PHP 3. În PHP 3 şi 4 obiectele erau tratate ca un tip de dată bazic, însemnand că de fiecare dată când o variabilă era asignată sau folosită într-o funcţie tot obiectul era copiat. Felul în care obiectele sunt tratate a fost complet rescris în PHP 5 iar acum obiectele sunt referenţiate printr-un vector intern şi nu după valoarea pe care o au. PHP 5 a introdus metode private şi protejate, clase abstracte, constructori şi destructori, functionalităţi similare cu cele din alte limbaje de programare care folosesc paradigma OOP, precum C++.

```
Class Person
{

 public $first;
 public $last;

 public function __construct($f,$l)
 {

 $this->first = $f;
 $this->last = $l;

 }

 public function greeting()
 {

 return "Hello, my name is {$this->first} {$this->last}.";

 }

 public function staticGreeting($first, $last)
 {

 return "Hello, my name is {$first} {$last}.";

 }

}

$him = new Person('John','Smith');
$her = new Person('Sally','Davis');

echo $him->greeting();
// afiseaza "Hello, my name is John Smith."

echo '<br />';
echo $her->greeting();
// afiseaza "Hello, my name is Sally Davis."

echo '<br />';
echo Person::staticGreeting('John','Smith');
// afiseaza "Hello, my name is John Smith."
```

Optimizari ale vitezei de executie

Codul sursa PHP e compilat de catre server la fata locului intr-un format intern ce poate fi executat de catre engine-ul PHP. Pentru a mari viteza de executie si pentru a scuti serverul de la a recompila codul sursa de fiecare data cand o pagina este accesata, scripturi PHP pot fi folosite in formatul executabil folosind un compilator PHP.

Optimizatoarele de cod au ca tinta reducerea timpului de compilare prin reducerea dimensiunii codului si facand si alte schimbari ce pot micsora timpul de executie marind in acelasi timp si performanta. Natura compilatorului PHP este in asa fel incat adesea exista opurtunitati de optimizare, un astfel de optimizator de cod fiind extensia eAccelerator.

O alta metoda de a reduce timpul de overhead pentru serverele cu incarcare mare de trafic este folosirea unui cache opcode. Ele functioneaza prin stocarea formei compilate a unui script PHP (opcode) in memoria shared pentru a evita overheadul parsarii si pe cel al compilarii de fiecare data cand scriptul e rulat.

Compilatoare

Limbajul PHP a fost original implementat folosind un interpretor de cod. Momentan exista numeroase compilatoare, ce decupleaza limbajul PHP de interpretorul sau.

phc - un compilator bazat pe C++ ce foloseste run-time-ul Zend pentru compatibilitate maxima;
Roadsend - atinge compilarea nativa prin compilarea in scheme bigloo, care apoi sunt compilate in C, apoi in limbaj masina;
Raven - e o rescriere a lui Roadsend PHP (rphp), bazat pe LLVM si un nou run-time C++;
Phalanger - compileaza codul sursa PHP in byte-code CIL;
Caucho Resin/Quercus - compileaza codul sursa PHP in byte-code Java;
HipHop - creat la Facebook si acum disponibil open-source, transforma scriptul PHP in cod C++ pe care apoi il compileaza;
php-to-scala - converteste codul PHP in cod Scala care apoi poate fi compilat in byte-code Java;

Avantajele compilarii includ nu doar timpi de executie mai buni, ci si obfuscare, analiza statica si interoperabilitate imbunatatita cu cod scris in alte limbaje.

Bibliografie:

istoric php si logo php: php.net
functionalitati versiuni: http://en.wikipedia.org/wiki/Php
sintaxa: Dezvoltarea aplicatiilor WEB cu PHP si MySQL, Teora, 2005