Implementari Java pentru aplicatiile de comert electronic

Asist. Carmen STANCIU
Catedra de Informatica Economica, A.S.E. Bucuresti

Desi a aparut doar de câtiva ani, limbajul Java este prezent deja în peste 200.000 de pagini de Web, are peste 400.000 de programatori care-l utilizeaza si peste 40% dintre aplicatiile dezvoltate, în special cele dedicate comertului electronic. Pentru a întelege ce înseamna Java, trebuie avuta în vedere tendinta Internet-ului de a deveni un bun de folosinta comun, o piata imensa, un teren electronic de desfasurare a afacerilor, ceea ce a focalizat interesul cercetatorilor si firmelor de specialitate, precum si utilizarea tot mai frecventa a solutiilor de tip Intranet în cadrul întreprinderilor de toate marimile. Coincidenta aparitiei Java cu fenomenul legat de Internet, Extranet, Intranet, a facut din el un instrument fundamental al dezvoltarii tehnologiei informatie.
Cuvinte cheie: *Java, applet, RMI, JVM, servlet, aglet, J-AAPI, ATP.*

Limbajul Java a aparut datorita necesitatii rezolvarii problemelor actuale ale programarii, fiind initial creat pentru a dezvolta software pentru aparatele electrocasnice, acestea fiind mici, portabile, distribuite, în timp real, usor de exploatat si fiabile.

Caracteristicile limbajului Java sunt multe, si de acea în continuare vor fi prezentate cele mai semnificative:

- **usor de utilizat** – atât pentru avansati cât si pentru începatori;
- **simplu** – fiind îndepartate aspectele derutante din C++, cum ar fi încarcarea operatorilor si mostenirea multipla; a fost introdus un colector automat de gunoaie care rezolva problema dealocarii memoriei în mod uniform, fara interventia programatorului; colectarea gunoaielor se realizeaza în timp ce un alt fir de asteptare asteapta o operatie de intrare/iesire sau un semafor;
- **independent de platforma** – compilatorul Java furnizeaza o secventa de instructiuni ale unei masini virtuale Java, iar executia aplicatiilor este interpretata; singura parte din mediul de executie Java care tine de masina este mediul de executie, care contine interpretorul si o parte din bibliotecile standard; îndeplineste conditia: "***Write once, run everywhere***";
- **viteza scazuta de executie** – este o caracteristica a limbajelor interpretate fata de cele compilate; pentru cresterea acestei viteze se poate cere mediului de executie Java sa genereze automat, plecând de la codul masinii virtuale, un executabil nativ platformei respective; de obicei, în Java se compileaza doar partile mari consumatoare de timp, celelalte fiind interpretate;
- **interpretorul Java** – este gândit pentru a lucra pe masini mici, iar împreuna cu bibliotecile standard sa nu depaseasca 300KB;
- **orientat obiect** – deoarece se pot crea clase de obiecte si instante ale acestora, se pot încapsula informatii, se pot mosteni variabile si metode de la o clasa la alta; suplineste mostenirea multipla (care lipseste), cu interfata, care permite definirea unui anumit comportament pentru a crea o clasa de obiecte, altul decât cel standard; în Java orice element este **obiect**, cu exceptia datelor primare si lipsesc functiile si variabilele globale;
- **distribuit** – are implementate biblioteci pentru lucrul în reteaua TCP/IP, suporta URL si încarcarea resurselor din retea; aplicatiile Java pot accesa foarte usor reteaua, prin apelurile catre un set standard de clase;
- **robust** – legarea functiilor se face în timpul executiei, iar informatiile de compilare sunt disponibile pâna în momentul rularii aplicatiei; astfel sistemul poate determina în orice moment neconcordanta dintre tipul referit la compilare si cel referit în timpul executiei, evitându-se intruziunile în sistem prin intermediul referintelor falsificate; pointerii lipsesc împreuna cu aritmetica

lor, eliminând înca o sursa principala de surse de erori, iar eliberarea memoriei ocupate de obiecte si tablouri se face automat, evitându-se încercarile de eliberare multipla a zonei de memorie;

- **securitate ridicata** – prin verificarea codului la fiecare încarcare, prin mecanisme de CRC si prin verificarea operatiilor disponibile pentru fiecare set de obiecte, robustete, încorporarea protectiei obiectelor la citire/scriere; (verificarea se face în timpul executiei, configurarea mediului de executie Java, care poate fi configurat pentru a proteja reteaua locala, fisierele si celelalte resurse ale calculatorului pe care ruleaza aplicatia Java;
- **multithreading** – suport nativ pentru aplicatii care lucreaza cu mai multe fire de executie, inclusiv primitive de sincronizare între firele de executie, independent de platforma;
- **dinamic** – bibliotecile de clase în Java putând fi reutilizate cu mare usurinta; variabilele sunt legate doar în faza de executie, rezolvând problema fragilitatii superclasei din C++; regasirea variabilelor se face prin nume si nu prin deplasament fix; se elimina necesitatea actualizarii aplicatiilor, datorata aparitiei unei noi versiuni de biblioteca, asa cum se întâmpla cu MFC-ul Microsoft si celelalte ierarhii C++;

Java furnizeaza un cadrul de lucru curat, format din mai multe straturi diferite care creeaza mediu de executie Java. Aceste straturi sunt: limbajul de programare; biblioteca standard API; compilatorul Java ; codul specific (bytecode); mecanism pentru încarcarea dinamica si verificarea bibliotecilor; automat de colectare a deseurilor (garbage collection) pentru eliberarea deseurilor; masina virtuala universala pentru executarea bytecod-ului, si anume JVM (Java Virtual Machine).

Ca limbaj, Java este înca în plina evolutie, din punctul de vedere al bibliotecilor standard de metode utilizate. O clasificare a **familiilor de interfetelor** în functie de domeniul de aplicatie a fost facuta, o parte din acestea apartinând lui Java Soft, iar restul dezvoltatorilor de aplicatii:

- **Core** – biblioteca care contine nucleul de clase Java, apartin lui Java Soft si sunt livrate în JDK (Java Development Kit); contine operatii legate de interfata grafica, implementarea structurilor de date, operatii de intrare/iesire, utilizare a retelei;
- **Enterprise** – grup de interfete pentru realizarea aplicatiilor de informatizare a întreprinderilor;
- **JDBC (Java Database Connectivity)** – este un set de interfete pentru accesul la bazele de date relationale;
- **RMI** – biblioteci de programare distribuita;
- **Transaction Services** – servicii în timp real pentru procesarea tranzactiilor;
- **Object Serialization, ODBMS** – interfata de comunicare cu Object Management System;
- **Media** – grup de biblioteci pentru publicare, animatie, multimedia;
- **Commerce** – creeaza un mediu standard pentru comertul electronic;
- **Embedded** – biblioteca de clase pentru aparatura inteligenta;
- **Management** – biblioteca de clase pentru supravegherea si gestiunea retelelor;
- **Security** – grup de biblioteci pentru transmiterea sigura a aplicatiilor în Internet;
- **Server** – biblioteca de clase pentru scrierea de aplicatii de tip server.

Legatura care s-a creat între Internet si Java a condus la situatia în care majoritatea companiilor care dezvolta aplicatii pentru Internet sa considere Java ca limbaj de implementare, în acest moment Java nefiind un limbaj de programare obisnuit, ci desemând un ansamblu de tehnolgii care permit trecerea de la o abordare a aplicatiilor *desk-centric*, la *network-centric*, realizându-se platforme de calcul deschise si universale accesibile din Internet.

Câteva dintre utilizarile si extinderile implementarilor limbajului Java vor fi prezentate în continuare, si anume applet-uri, servlet-uri, aglet-uri, precum si legatura lui Java cu bazele de date si cu securitatea pe Internet.

1. Applet-uri

Programele Java pot rula fie independent, fie în interiorul unei pagini HTML încarcate de un browser. Aplicatiile Java care ruleaza într-o pagina HTML se numesc applet-uri.

În contextul navigatoarelor de pe Internet, applet-urile Java sunt foarte des utilizate, datorita caracterului dinamic pe care-l confera paginilor Web.

Pentru vizualizarea rezultatelor rularii unui applet, trebuie creat un document HTML minimal, care sa contina tag-ul:

<APPLET CODE="nume_appplet.class" WIDTH=150 HEIGHT=25> </APPLET>

Spre deosebire de o aplicatie normala Java, applet-urile nu pot primi parametri pe linia de comanda, deoarece acesta nu exista. Parametrii trebuie introdusi în fisierul HTML, imediat dupa linia de declaratie a applet-ului, prin tag-ul PARAM, ca în exemplul urmator:

<PARAM NAME=mesaj VALUE="continut mesaj.....">

Executia unui applet este marcata de câteva evenimente importante, generate de catre navigator, dupa cum urmeaza:

- porneste încarcarea codului necesar rularii applet-ului, atunci când browser-ul întâlneste o eticheta APPLET;
- se initializeaza - în acest moment applet-ul îsi pregateste parametrii si obtine de la sistem resursele necesare rularii;
- browser-ul trimite catre applet o comanda de pornire, care va pune efectiv în functiune applet-ul, deschizând interactiunea cu utilizatorul.

Un applet ruleaza atâta timp cât browser-ul este activ. La schimbarea paginii curente, applet-urile din vechea pagina primesc o comanda de oprire temporara. La reîncarcarea paginii applet-ul este reluat printr-o comanda de pornire. La oprirea browser-ului, applet-ul primeste o comanda de oprire definitiva, moment în care trebuie sa elibereze toate resursele pe care le blocheaza.

Între doua evenimente de oprire si de pornire temporara a applet-ului, browserul îi transmite evenimente specifice oricarei interfete grafice (cum ar fi apasarea unor taste, butoane ale mouse-ului etc), iar applet-ul îi raspunde corespunzator. Figura 1 prezinta starile prin care trece un applet Java înglobat într-o pagina HTML.

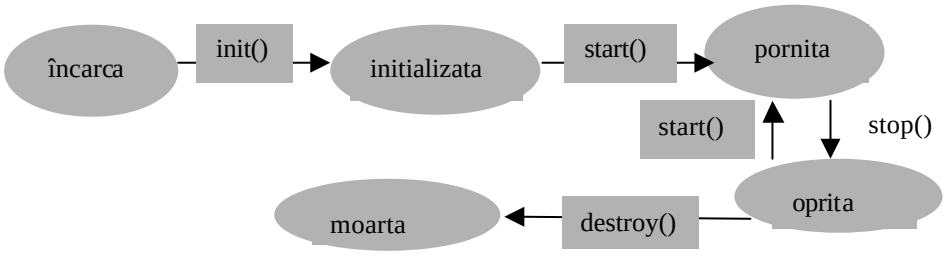

Figura 1

Orice applet Java reprezinta din punctul de vedere al limbajului un nou obiect, derivat din obiectul standard Applet. Când browserul lanseaza un nou obiect, el practic instantiaza un nou obiect din clasa respectiva.

Pentru crearea applet-urilor exista în prezent o multitudine de instrumente, dintre care amintim: *Activator Pro V2 Pack, Astound Webmotion, DimensionX Liquid Motion, Sausage Egor v3.1, IBM Netrexx 1.00, Iavadraw 3.0 de la SFS, JDK de la JavaSoft, Visual J++ de la Microsoft s.a.*

2. Servlet-uri

Servlet-urile, componente independente de platforma si de protocol ale aplicatiilor server, extind dinamic server-ele care au integrat suport Java si asigura un cadrul general pentru servicii pe baza modelului cerere-raspuns.

Sunt folosite la programarea sistemelor distribuite, începând cu apeluri de proceduri la distanta si terminând cu protocolul HTTP pentru cereri catre server-ele Web.

Servlet-urile pot fi considerate echivalentul applet-urilor pe partea de server, dar lipsite

de interfata grafica, în rest fiind din multe puncte de vedere asemanatoare. Sunt un hibrid între CGI si interfetele de programare la nivel inferior, cum ar fi NAPI sau ISAPI, se comporta ca o aplicatie CGI cu o durata de viata mai mare si care se încarca dinamic, eliminând astfel gâtuirile specifice CGI-ului. Dupa încarcare, la o noua cerere nu mai este necesar sa fie creat un nou servlet, ci sa se apeleze o metoda care deserveste cererea, fiind necesara doar o schimbare usoara de context pentru un alt fir de executie.

Pentru ca un servlet sa poata rula pe un server Web este necesar ca:

- serverul sa aiba integrata o masina virtuala Java (JVM –Java Virtual Machine);
- sa dispuna de o interfata de comunicare cu servlet-uri;
- sa gestioneze servlet-uri aflate în executie.

Servlet-urile au fost initial suportate de Java Web Server de la JavaSoft, dupa care au fost introduse si în alte servere Java, cum ar fi cele de la Netscape. Interfata de programare Java Servlet API, dezvoltata de JavaSoft nu face parte din nucleul de programare Java, fiind o extensie Java.

În API-ul servlet exista suport integrat pentru securitate. Acest lucru se datoreaza mediului Java standard care are un Security Manager care poate controla accesul la toate resursele, cum ar fi fisiere locale, alte fire de executie care ruleaza pe server, informatii administrative ale serverului, accesul la alte servere din retea etc. Astfel se poate asigura un mediu de executie controlat pentru servlet-uri, asemanator cu cel folosit de browsere pentru a controla applet-uri. Administratorul serverului poate specifica carui servlet i se acorda drepturi de accesare a retelei sau fisierelor locale prin intermediul componentei Security Manager.

Servlet-urile au acces si la informatii despre clientii lor prin intermediul cererii receptionate, si anume pot afla adresa IP si numele calculatorului gazda de unde a venit cererea, tipul protocolului cererii respective etc. Daca se foloseste un protocol de comunicatie sigur cum ar fi SSL (Secure Sockets Layer), atunci se poate face o identificare sigura a clientului.

Servlet-urile pot fi identificate prin URL, iar cele locale, prin numele clasei respective. Dupa încarcarea si initializarea servlet-ului, acesta poate fi invocat printr-un URL, astfel: ***http://server_host/servlets/<nume_servlet>?argumente.***

Unele servere Web, care au suport complet pentru servlet-uri, pot asigura si functionalitatea SSI (Server Side Include). Aceste servere pot apela servlet-uri pentru a preprocesa pagini Web înainte de a fi trimise clientilor, printr-o sintaxa HTML speciala, si anume tag-ul **SERVLET**. Sintaxa tag-ului în fisierele .**shtml** este urmatoarea:

<SERVLET NAME=nume_servlet
CODE=nume_clasa_servlet.class
CODEBASE=locatie_servlet
initparam1=initvalue1 ...>
<PARAM NAME=param1 VALUE=valoare1> ....
</SERVLET>

Avantajele utilizarii servlet-ului sunt:

- este mai **rapid** si mai **sigur** decât script-ul CGI;
- este **portabil** – poate rula pe diverse platforme fara a necesita o rescriere;
- permite **încarcarea dinamica** – poate fi încarcat si invocat de pe discul local sau de la distanta;
- este usor de **configurat** – instalarea, pornirea/oprirea, specificarea parametrilor si caracteristicilor unui servlet se poate face prin intermediul lui GUI based Administration Tool;
- ofera **siguranta în functionare** – modelul de securitate si de încapsulare (sandbox) protejeaza servlet-ul de atacuri;
- ofera **posibilitate de înlantuire** – posibilitatea de a invoca unul sau mai multe servlet-uri într-o secventa;
- **executie dinamica** – posibilitatea de a fi apelat direct din paginile de Web, prin tag-uri recunoscute de server;
- **existenta unui API standard** – utilizeaza Servlet API, o extensie a API-ului standard Java, fiind astfel independent de protocol si de platforma;
- **usurinta în dezvoltare** – prin mediul de dezvoltate JSDK (Java Servlet Development

Kit), care pune la dispozitie instrumente care faciliteaza scrierea servlet-urilor;
- **persistenta si configurarea "on-the-run"** – permite utilizarea tehnologiei Java Beans pentru crearea servlet-urilor.

Arhitectura unui servlet este ilustrata în figura 2.

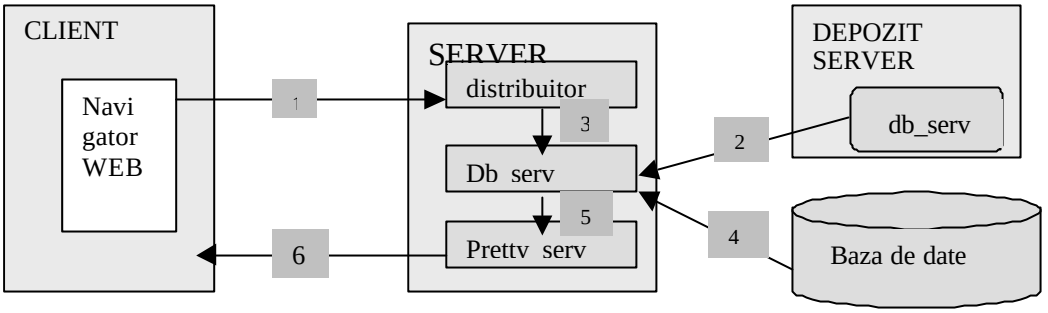

Figura 2

unde:
1- clientul Intranet lanseaza o cerere HTTP;
2- distribuitorul încarca servlet-ul din depozit (daca este nevoie);
3- servlet-ul primeste cererea;
4- servlet-ul acceseaza baza de date;
5- rezultatele sunt oferite urmatorului servlet;
6- servlet-ul traduce rezultatele în HTML si le trimite clientului.

În concluzie, servlet-urile reprezinta o solutie mult mai rapida si mai sigura decât CGI-urile, iar prin implementarile lor în Java Web Server primesc si încarcare si administrare dinamica.

Java Web Server este o platforma fiabila, flexibila, peste care pot rula usor aplicatii, oferind suport si pentru componente Java Beans, administrare facila, securitate, precum si alte servicii.

În scurt timp, una dintre cele mai frecvente utilizari ale servlet-urilor va fi în cadrul middle-tier-urilor, în retelele companiilor pentru conectare la baze de date SQL prin JDBC, în cadrul EJB, Enterprise Java Beans.

3. Aglet-i

Una dintre cele mai interesante domenii în care Java ar putea fi utilizat în viitor ar fi **agentii**, produse software destinate efectuarii unor activitati de rutina.

Aceste programe se mai numesc si **agenti inteligenti**, deoarece sunt capabile sa ia decizii în cele mai dificile situatii, precum si **agenti mobili**, atunci când se pot deplasa pe Internet pentru a opera pe alte calculatoare.

Agentii mobili pot fi definiti ca fiind obiecte care au încapsulate functionalitati, stari si adrese de rezidenta. Pentru fiecare agent mobil exista un model de la care mosteneste o mare parte din functionalitati, cum ar fi migrarea de pe un calculator pe altul; metode pentru schimbul de mesaje cu alti agenti; seturi de evenimente care sunt importante în timpul vietii agentului, cum ar fi *crearea, lichidarea, expedierea, sosirea si comunicarea*.

Aglet-ii sunt agenti mobili programabili cu ajutorul limbajului Java .

În opinia lui NWANA, agletii sunt procese software computationale capabile sa calatoreasca prin retele WAN, sa interactioneze cu host-uri straine în scopul obtinerii unor informatii pentru utilizatorul care i-a creat si apoi sa se întoarca de unde au plecat si sa prezinte rezultatul calatoriei lor.

Un mediu de agenti mobili este **AWB - Aglets Workbench** -, creat de o echipa de cercetatori de la IBM Japan Laboratories.

Un agent creat cu AWB se numeste aglet si are urmatoarele caracteristici:
- **mobilitate** – se refera la posibilitatea de a realiza un task la un moment dat pe un anumit host dupa care se pot opri, deplasa pe un alt host si continua task-ul întrerupt; astfel aglet-ul îsi transporta prin retea *codul* si *starea* ;
- **autonomie** – poate contine suficient de multa informatie pentru a decide ce task sa execute, unde si când;

- **executie asincrona si paralela** – a firului de executie (thread) care implementeaza aglet-ul;
- **interactiune cu alti aglet-i** – se realizeaza cu ajutorul mesajelor, aglet-ul putând trimite un mesaje sincrone sau asincrone altui aglet sau mediului agent.

Echipa de la IBM pune la dispozitia implementatorilor un pachet Java care contine API-ul pentru dezvoltarea aglet-ilor proprii, numit **J-AAPI** (Java Aglet API) si care contine o seama de interfete si clase cum ar fi: interfetele *AgletContext, AgletProxy*, clasele *Aglet, AgletID, FutureReply, Message, ReplySet* etc.

Tehnologia de realizare a aglet-ilor cuprinde si un protocol de transfer agent numit **ATP (Agent Transfer Protocol)**, necesar transferurilor agentilor prin retea. ATP, un protocol de retea uniform si independent de platforma la nivel aplicatie pentru sisteme distribuite bazate pe agenti, foloseste URL-ul (Unified Resource Locator) ca adresa pentru host-urilor unde se gasesc agenti . Este scris în întregime în Java, având astfel avantajele portabilitatii si formarii unui API standard, conectarea la host-uri aTP, generarea de cereri si raspunsuri ATP specifice.

Uneltele puse de IBM la dispozitia programatorilor pentru dezvoltarea aglet-ilor sunt: ***pattern-uri de aglet-i, server de aglet si lansator de aglet-i***.

În concluzie, domeniul agentilor mobili este în continua dezvoltare, iar aparitia limbajului Java alaturi de tehnologiile aferente au ca scop cresterea interesului fata de acest domeniu precum si extinderea utilizarii lui în cât mai multe sfere ale aplicatiilor distribuite.

În scurt timp reteaua Internet, cadrul propice pentru dezvoltarea si actiunea agentilor, va fi împânzita de agenti inteligenti si mobili care vor face activitati de rutina pentru utilizatori.

Bibliografie

*** java.sun.com

*** java.sun.com/products/jdbc, 1999

*** Java Development Kit 1.2 beta 4, documentatie

/ARPA97/ ARPAD, Z., Servleturi – un alt fel de aplicatii Java, BYTE România, mai, 1997

/CLIP98/ CLIP, P., Servleturi: alternativa Java la CGI, BYTE România, mai,1998

/RUSU97/ RUSU, I., Agletii cei vioi, PC Report, sept., 1997

/RUSU97/ RUSU, I., Agenti mobili, BYTE România, aug., 1997