Elements of J2EE Architecture Used in Design of Information Systems

Victor BĂRCAN
Facultatea de Ştiinţe Economice, Universitatea Babeş-Bolyai din Cluj-Napoca

The Java 2 Enterprise Edition (J2EE) Platform is defined by four key pieces: the specification, the reference implementation, the compatibility test suite, and the BluePrints program. BluePrints describes the best practices and design guidelines for a distributed component architecture. This article describes the principals characteristics of the J2EE architecture, the evolution of the norms of J2EE and the notion of the J2EE security.
***Keywords**: J2EE platform, security, architecture.*

Arhitectura J2EE -Scurt istoric

Norma J2EE[1] este o normă iniţiată de către SUN, la care au participat un anumit număr de parteneri importanţi, dintre care cel care a investit cel mai mult este IBM. Unul dintre motivele creării limbajului de programare JAVA este independenţa de platformă, deziderat realizat prin crearea de JVM-uri (Java Virtual Machine) specifice sistemelor de operare. O dată cu dezvoltarea tehnologiei internet a apărut necesitatea redării de conţinut dinamic în pagini. Răspunsul tehnologiei SUN, orientată java, la această nouă provocare a fost apariţia tehnologiilor SERVLETE si JSP (java server pages). În paralel a apărut şi tehnologia EJB (Entrepise Java Beans), iar mai târziu, începând cu 2004, JSF (java server faces).

Din ce în ce mai mulţi creatori de soft au început să producă servere de aplicaţie care să suporte aceste tehnologii. S-a constatat însă că aplicaţiile scrise în acea perioadă (până la sfârşitul lui 2000), erau dependente de serverul de aplicaţie şi în consecinţă nu erau portabile pe alt server.

Norma J2EE a apărut ca răspuns la nevoia de a extinde ideea de portabilitate de la nivelul limbajului (java) la nivelul serverelor de aplicaţie. De exemplu, un server care este certificat conform normei J2EE 1.3 garantează că implementează specificaţiile de servlete 2.3, JSP 1.2 şi EJB[2] 2.0, etc.

Acest fapt permite portarea unei aplicaţii scrisă conform normei J2EE 1.3 de pe un server de aplicaţie pe altul.

În afară de portabilitate, norma J2EE prezintă o serie de alte avantaje, cum ar fi dezvoltarea orientată pe componente, separarea containerelor etc., care depăşesc obiectul de studiu al lucrării de faţă.

Gestiunea securităţii într-o aplicaţie J2EE

Securizarea unei aplicaţii de întreprindere conform normei J2EE se face pe baza unor standarde, în cadrul cărora se pune accent pe separarea ***autentificării*** de ***autorizare***. Securizarea se poate face pe două nivele: declarativ şi aplicativ.

În cadrul implementării securităţii declarative a unei aplicaţii J2EE, prima etapă este autentificarea în care utilizatorul furnizează un nume de utilizator şi o parolă pentru a dovedi că este cine pretinde. %n principiu, această etapă nu trebuie gestionată de către aplicaţie aşa cum se întâmplă din păcate in majoritatea proiectelor, ci trebuie externalizată. O arhitectură fiabilă, şi evolutivă în acest sens o constituie folosirea unui server LDAP[3] ca şi referenţial unic al întreprinderii. Toţi utilizatorii întreprinderii sunt descrişi în acest referenţial. Serverul LDAP este folosit doar pentru autentificare. Când un utilizator încearcă să acceseze o aplicaţie, el se autentifică pe serverul LDAP, după care controlul este reluat de către aplicaţia apelată, care autorizează accesul utilizatorului pe diferite module

[1] Amănunte despre norma J2EE pot fi găsite la adresa: http://java.sun.com/j2ee/

2 Amănunte despre tehnologie EJB pot fi găsite la adresa: http://java.sun.com/products/ejb/

[3] Amănunte despre protocolul LDAP pot fi găsite la adresa: http://www.stanford.edu/~hodges/talks/mactivity.ldap.97/index2.html

ale aplicației în funcție de grupul la care aparține pe serverul LDAP.
%n cazul unei aplicații J2EE interacțiunea reală a diferiților actori este modelizată prin intermediul noțiunii de rol J2EE. Un rol J2EE corespunde de fapt unui tip de utilizator al aplicației. De exemplu, va exista un rol J2EE supervizor, un rol responsabil departament, un rol agent comercial etc.
Norma J2EE nu impune folosirea unui server LDAP, ci doar gestionarea drepturilor de acces cu ajutorul rolurilor J2EE. Utilizatorii reali pot fi de pildă, definiți ca şi grupuri de utilizatori pe serverul pe care este instalat serverul de aplicație.
Pentru containerul WEB, aceste roluri sunt definite în cadrul unui fişier XML de configurare, cu numele web.xml.
Pe urmă, se atribuie drepturi de acces rolurilor J2EE asupra diferitelor resurse WEB. O resursă WEB este unic identificată în cadrul aplicației prin URL-ul de acces.
Foarte important de reținut este faptul că drepturile de acces se atribuie nu utilizatorilor reali, ci rolurilor J2EE.
%n momentul instalării aplicației pe serverul J2EE se defineşte o corespondență între rolurile J2EE şi grupurile definite pe serverul LDAP, sau grupurile de utilizatori ai sistemului de operare local.
Dacă o nouă persoană se angajează în întreprindere, într-o astfel de arhitectură este suficient ca ea să fie atribuită grupului LDAP corespunzător, fără nici o intervenție în administrarea aplicațiilor care se sprijină pe serverul LDAP pentru autentificare.
Securizarea declarativă împiedică sau acordă accesul utilizatorilor asupra resurselor WEB ale aplicației. Pentru a avea o aplicație corectă din punct de vedere funcțional, acest lucru nu este suficient. Astfel, dacă un utilizator nu are acces la o pagină, butonul care accesează pagina respectivă trebuie să nu fie vizibil pentru utilizatorul în cauză. Acest lucru este realizat prin implementarea securității aplicative, prin API-urile suportate de către norma J2EE şi care permit testarea rolului unui utilizator autentificat. Important este de reținut faptul că odată ce utilizatorul s-a autentificat pe aplicație nu se mai testează asupra ID-ului său ci asupra apartenenței sale la un rol sau altul. Securitatea aplicativă întregeşte astfel ansamblul de unelte puse la dispoziția programatorului de către implementările normei J2EE pentru o gestiune completă a securității unei aplicații.

a. Motorul de servlete[4].

Motorul de servlete denumit şi "Web Container" este cel care garantează execuția unei servlete într-un context bine definit.
O servletă mai este denumită şi extensie a serverului sau „faceless object". Acestea sunt programe care se execută pe server şi nu pe client, şi au apărut ca o alternativă performantă față de programele CGI. Diferența principală dintre o servletă şi un program CGI este că pentru „n" cereri există o instanță şi „n" Thread-uri în cazul servletei şi „n" instanțe în cazul programului CGI.
O definiție[5] a conceptului de servletă s-ar enunța în felul următor: „ Tehnologia servletelor constă intr-un ansamblu de API java [6] care permit programatorului să adauge conținut dinamic unui server Web bazat pe tehnologia java. Conținutul generat este în general HTML[7], dar poate fi şi de altă natură ca de pildă XML[8]".
Există numeroase alte avantaje ale acestei tehnologii care depăşesc cu mult scopul acestei lucrări.
Motorul de servlete se ocupă de execuția servletei în sistem de Thread-uri, de gestiune a variabilelor de instanță, de colectare a instanțelor nefolosite etc.

b. Procesorul JSP[9].

Tehnologia JSP a apărut pentru a acorda dinamicitate paginilor statice HTML, şi pentru a evita scrierea de cod HTML în servlete. Există la ora actuală o multitudine de medii

4 Detalii despre Tehnologia servletelor pot fi găsite la adresa: http://java.sun.com/products/servlet/index.jsp
5 Definiție adaptată după: http://encyclopedia.laborlawtalk.com/Java_Servlet
6 Acronimul de **API** simbolizează « **Application Programming Interface**", iar o definiție a acestui concept poate fi găsită la adresa: http://encyclopedia.laborlawtalk.com/API
7 Acronimul de **HTML** simbolizează « HyperText Markup Language", iar o definiție a acestui concept poate fi găsită la adresa: http://encyclopedia.laborlawtalk.com/HTML
8 Acronimul de **XML** simbolizează « Extensible Markup Language", iar o definiție a acestui concept poate fi găsită la adresa: http://encyclopedia.laborlawtalk.com/XML
9 Detalii despre tehnologia JSP pot fi găsite la adresa: http://java.sun.com/products/jsp/

de programare care pot gestiona vizual compoziția unei pagini JSP, obținând câştiguri foarte importante în productivitate față de generarea dinamică a codului HTML în cadrul unei servlete. Practic, tehnologia a apărut pentru a concepe partea vizuală a aplicației prin editoare vizuale specifice.
Următoarea fază de scriere a unei JSP este translatarea acesteia, adică crearea unei servlete aferente şi a fişierului cu extensia java. Urmează apoi faza clasică de compilare în care fişierul cu extensia class este generat. De aici încolo, pagina se comportă exact ca o servletă.
Este tehnologia utilizată împreună cu tehnologia servletelor în cadrul unor J2EE Design Patterns, aşa cum vor fi prezentate în cadrul lucrării.

c. Tehnologia JSP Custom Tags[10]

Constă în extensii ale limbajului JSP denumite „Custom Tags Libs", pe care orice organizație sau individ le poate scrie şi folosi în cadrul proiectelor proprii precum le poate şi pune la dispoziția comunităților libere cum ar fi comunitatea „open source". Ideea de bază a „custom tag libs" este de a reutiliza la maxim codul java din cadrul JSP-urilor, precum şi de a reduce la maxim codul java în cadrul acestora. Întrucât moştenirea de clase nu este aplicabilă tehnologiei JSP, (este însă aplicabilă tehnologiei bazate pe servlete), această tehnologie apărut pentru a permite un grad de reutilizare ridicat a codului de prezentare din cadrul paginilor de tip JSP.

d. Accesul la o bază de date cu ajutorul unei DataSource.

Primele aplicații scrise în java care accesau baze de date foloseau un driver manager, care presupunea crearea unei conexiuni la baza de date şi distrugerea acesteia după executarea interogării. Dacă un alt utilizator dorea executarea unei interogări, procesul era reluat de la capăt. %n cazul unei interogări SQL, partea care consumă mai mult timp şi mai multe resurse sistem este crearea conexiunii şi nu execuția interogării. Astfel, a apărut necesitatea folosirii unui mecanism care să permită refolosirea conexiunilor JDBC[11] denumit DataSource. O DataSource foloseşte un pool de conexiuni JDBC pentru a optimiza accesul la baza de date.
Crearea unei Data Source este un proces administrativ, folosindu-se în general o consolă de administrație pentru a crea şi parametriza un astfel de obiect.
Printre parametrii de configurare folosiți se află şi numărul minim şi numărul maxim de conexiuni JDBC. În momentul lansării serverului de aplicație sunt create şi inițializate atâtea conexiuni câte au fost specificate ca şi număr minim de conexiuni. Spre exemplu, să presupunem că valoarea acestui parametru este 100. Dacă 100 de utilizatori execută simultan o interogare SQL, cele 100 de conexiuni sunt folosite în acest scop. %n momentul în care un utilizator eliberează o conexiune terminându-şi tranzacția, această conexiune nu este distrusă ci este înapoiată pool-ului de conexiuni. Dacă un nou utilizator încearcă să execute o interogare, DataSource-ul verifică dacă printre cele 100 de conexiuni preinițializate există una disponibilă, caz în care aceasta este folosită pentru execuția interogării. Doar dacă toate cele 100 de conexiuni sunt folosite la momentul respectiv, o nouă conexiune este creată, inițializată, folosită, iar după folosire este înapoiată pool-ului de conexiuni.
Acest mecanism permite refolosirea conexiunilor JDBC şi obținerea unor performanțe sporite ale aplicației.
Pentru a accesa o DataSource din cadrul unui program java în vederea obținerii unei conexiuni, se foloseşte un protocol denumit JNDI[12]. Acest protocol permite localizarea diferitelor obiecte şi date definite în prealabil pe un server de nume. Toate serverele comerciale sau open source care implementează norma J2EE, încorporează un « server de nume ». De exemplu, la instalarea unei aplicații de tip EJB, pentru fiecare componentă de tip EJB trbuie specificat un nume logic

10 Detalii despre tehnologia JSP Custom Tags pot fi găsite la adresa:
http://java.sun.com/products/jsp/taglibraries/index.jsp

11 Acronimul de JDBC simbolizează «Java Database Connectivity", iar o definiție a acestui concept poate fi găsită la adresa : http://encyclopedia.laborlawtalk.com/JDBC

12 Acronimul de JNDI simbolizează «Java Naming and Directory Interface", iar o definiție a acestui concept poate fi găsită la adresa : http://encyclopedia.laborlawtalk.com/JNDI

unic JNDI unic[13], care va fi înregistrat de către serverul de nume, şi folosit de către clienţi pentru a accesa componenta respectivă.
În acelaşi mod, în procesul definirii unei DataSource i se atribuie un nume logic JNDI unic, folosit apoi de către programatori pentru a accesa la această componentă în vederea obţinerii unei conexiuni la baza de date.

Bibliografie

1. Hanumant Deshmukh, Jignesh Malavia, *SCWCD Exam Study Kit: Java Web Component Developer Certification,* Manning Publications 2002 ISBN: 1930110596
2. Lazăr, I.n Frenţiu, M., Niculescu, V.,"Programare orientată în JAVA" Ed. Univ. Petru Maior, Târgu Mureş, 1999
3. Phillip Heller, Simon Roberts, *Complete Java 2 Certification Study Guide, 3th Edition,* Sybex Inc 2002 ISBN: 0782140777
4. http://java.sun.com/j2ee/
5. http://www.javaranch.com

13 Este general acceptată şi folosită o convenţie de nume pentru prefixurile diferitelor tipuri de componente:

- ejb/ pentru componente de tip EJB
- jdbc/ pentru compenente de tip DataSource
- jms/ pentreu compenentede tip Messaging (Topic sau Queue)