UNIVERSITATEA POLITEHNICA din BUCUREŞTI

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Inginerie Software

NetBeans IDE

Profesor coordonator:

Prof. univ. Dr. ing. Ştefan Stăncescu

Studenţi: Ionescu - Niculescu Ana – Ioana

Sava Alexandru Nicolae

Grupa 443A

Cuprins

Introducere

NetBeans a apărut pentru prima dată în 1996 sub numele de *Xelfi* (un joc de cuvinte puntru Delphi), ca proiect Java IDE al unui student, sub îndrumarea Facultății de Matematică și Fizică de la Universitatea Charles din Praga. În 1997 Roman Stanek a format o companie în jurul proiectului și a produs versiuni comerciale ale NetBeans IDE. În 1999 a fost cumpărat de către Sun Microsystems. NetBeans IDE devine open souce în luna iunie a anului următor. De atunci, comunitatea NetBeans a continuat să crească. În 2010, Sun (și, astfel, NetBeans) a fost achiziționată de către Oracle.

NetBeans IDE reprezintă un mediu de dezvoltare integrat (Integrated Development Environment), open source, gratuit care vă permite să dezvoltați atât aplicații desktop, mobile, dar și aplicații web. Mediu de dezvoltare NetBeans oferă suport pentru dezvoltarea aplicațiilor în diferite limbaje de programare, inclusiv Java, HTML5, PHP și C ++. NetBeans oferă suport integrat pentru ciclul de dezvoltare completă, de la proiectare, creare, depanare, profilare până la implementare. IDE rulează pe Windows, Linux, Mac OS X și alte sisteme bazate pe UNIX.

Un mediu de dezvoltare integrat (IDE) reprezintă o aplicație software care oferă facilități complete pentru programatori în dezvoltarea de software. Un IDE constă în mod normal dintr-un cod editor sursă, instrumente de construcţie automate și un program de depanare. Cele mai multe IDE-uri moderne au şi un cod de completare a instrucţiunilor inteligent.

Unele IDE-uri conțin un compilator, interpretor sau ambele, cum ar fi NetBeans, în timp ce alte IDE-uri nu au. Granița dintre mediul de dezvoltare integrat (IDE) și alte părți generale ale mediului de dezvoltare nu este bine definită. Uneori, un sistem de control al versiunii (VSC* - Version Control Systems) sau diverse instrumente pentru a simplifica construirea unei interfeţe grafice pentru utilizator (GUI - Graphical User Interface), sunt integrate. Multe IDE moderne au, de asemenea, un browser de clasă, un browser obiect și o diagramă de clasă ierarhică, pentru a fi utilizate în dezvoltarea de software obiect orientat.

NetBeans oferă suport complet pentru tehnologiile JDK 7 și cele mai recente versiuni Java. Acesta este primul IDE care oferă suport pentru JDK 7, Java EE 7 și JavaFX 2. NetBeans sprijină pe deplin Java EE folosind cele mai noi standarde pentru Java, XML, Servicii web, SQL și sprijină GlassFish Server, implementarea de referinţă a Java EE.

**Controlul versiunii (VSC) este un sistem care înregistrează modificările aduse unui fişier sau unui set de fişiere în timp, astfel încât să puteţi reveni la versiuni anterioare mai târziu. Spre exemplu: dacă sunteţi un designer grafic sau web şi doriţi să păstraţi fiecare versiune a unei imagini sau aspect un sistem de control versiune (VCS) vă este foarte folositor. Acesta vă permite să reveniţi la o stare anterioară a fişierelor sau a întregului proiect, comparaţi modificările în timp, vedeţi cine a modificat ultima dată ceva şi când anume.*

2. Lucrul într-un mediu de colaborare

Dezvoltatorii care colaborează la un proiect necesită un set de instrumente şi au nevoie de o infrastructură care să îi poată ajuta să rămână conectati între ei și să lucreze împreună ca o echipă.

În plus pentru a partaja surse, membrii echipei trebuie să fie disponibili pentru a împărţii informaţii şi de a comunica unii cu alţii, iar felul în care sunt împărtăşite informaţiile depinde de tipul informaţiei de care au nevoie.

Într-un mediu de colaborare, membrii unei echipe au diferite roluri şi cerinte. De exemplu, în plus, faţă de dezvoltatorii de software, o echipa ar putea include oameni în urmatoarele atribuţii:

- Asigurarea calităţii(QA)
- Management-ul proiectelor
- Documentaţie
- Marketing

Nu toţi membrii unei echipe folosesc aceleaşi unelte (instrumente), dar comunicarea între membrii poate fi simplificată atunci când infrastructura şi uneltele sunt integrate. IDE(Integrated Development Enviroment-Mediu de dezvoltare) oferă suport integrat pentru următoarele instrumente şi servicii:

Problema de urmărire (Issue Tracking)-(Bugzilla, JIRA) – o problemă de urmărire permite dezvolatatorilor şi utilizatorilor să raporteze şi să urmărească sarcinile(tasks-urile) care sunt asociate cu un proiect şi oferă un mecanism valoros de feedback pentru persoanele implicate în proiect. Integrarea unui sistem de urmărire în IDE permite programatorilor(developers) să găsească,să vadă şi să rezolve sarcinile proiectului din cadrul IDE.

Sistem de control al versiunii (Subversion, Mercurial, Gât)-IDE oferă suport integrat pentru sisteme de control al versiunii pentru a ajuta programatorii să gestioneze revizia istoria fişierelor.

Echipa de server (Team Server) – O echipă de server poate da o infrastructură de servicii pentru fiecare proiect în parte ţinut pe server-ul desemnat. După ce te conectezi la o echipă înregistrată de server, poţi să deschizi şi să creezi proiecte şi să accesezi multe servicii din IDE.

2.1. Lucrul cu task-urile (sarcini)

Se pot organiza problemele ca task-uri în IDE, acestea fiind înregistrate într-un registru de urmărire a problemelor (registered issue tracker). Pentru a lucra cu task-uri în IDE este nevoie să se specifice sistemului pentru urmărire a problemelor că este utilizat ca un depozit pentru proiectul în desfăşurare.

2.1.1Depozite de sarcini (task-uri)

Un depozit de sarcini este un sistem pentru urmărirea problemelor, folosit pentru un proiect. Dacă proiectul este asociat cu un depozit de sarcini, se pot trimite sarcini şi să se stabilească obligaţia pentru rezolvarea task-urilor membrilor unei echipe care colaborează la un proiect.

După ce se înregistrează un depozit de sarcini cu IDE, se pot efectua următoarele task-uri din cardul IDE:

-Găsirea, actualizarea şi rezolvarea sarcinilor (task-urilor)
-Crearea unei noi sarcini
-Organizarea sarcinilor în funcţie de categorie
-Crearea şi salvarea interogărilor

2.1.2 Cum să lucrăm cu sarcinile

După ce se înregistrează un depozit de sarcini cu IDE, se poate utiliza fereastra Tasks pentru a ajuta la găsirea, actualizarea şi crearea unei sarcini.

2.1.3 Cum să adăugăm un depozit de sarcini

Pentru a utiliza un sistem pentru urmărirea problemelor cu IDE, este nevoie să se înregistreze sistemul ca depozit de sarcini. După ce depozitul de sarcini este înregistrat se pot utiliza uneltele în IDE pentru a căuta, raporta şi soluţiona sarcinile care sunt înregistrate în depozit.

2.1.4 Cum să adăugăm sprijin pentru JIRA

Pentru a activa suportul integrat pentru sistemul de urmărirea problemelor JIRA în IDE este nevoie să se instaleze JIRA plugin.

Dacă JIRA plugin nu este instalat şi se încearcă vizualizarea sau interacţiunea cu sarcinile care sunt urmărite folosind JIRA, IDE va indica utilizarea manager-ului Plugin să instaleze JIRA plugin.

2.2. Lucrul cu Tasks Window (Fereastra Tasks)

Fereastra Tasks oferă o viziune organizată a task-urilor care sunt înregistrate într-un depozit de sarcini. Pentru a utiliza fereastra Tasks trebuie să fie înregistrat un depozit de sarcini in IDE. Se poate adăuga un depozit de acest gen în fereastra Services.

Secţiunea Categories (Categorii) din fereastra Tasks afişează o listă de sarcini care sunt organizate după y categorii. Secţiunea Repositories (Depozite) din fereastra Tasks afişează o listă

cu toate sarcinile care reprezintă rezultatul unei interogări salvate. Se pot crea noi categorii şi interogări din fereastra Tasks.

2.2.1 Cum să vizualizăm task-urile

Fereastra Tasks oferă posibilitatea de a vizualiza liste cu sarcini organizate. Se pot organiza sarcini prin încadrarea acestuia într-o categorie. De asemenea, se pot salva interogarile şi se pot vizualiza rezultatele interogării in fereastra Tasks.

2.3.2 Cum se pot organiza sarcinile

Se pot utiliza categoriile pentru a grupa sarcinile, în fereastra Tasks. Dupa ce se crează o categorie, se poate atribui orice sarcină acelei categorii, iar aceasta sarcină va rămâne acolo până când este scoasă sau adăugată unei alte categorii. O sarcina poate să fie într-o singură categorie. Secţiunea Categories afişează toate sarcinile care sunt atribuite unei categorii.

3. Crearea proiectelor Java

Un proiect este un grup de fişiere sursă şi setări cu care se construieşte, se rulează şi se depanează aceste fişiere sursă. In IDE , toate dezvoltările Java trebuie să se desfăşoare în cadrul unui proiect. Pentru aplicaţiile care au un cod destul de mare, este deseori avantajos să se împartă codul sursă al aplicaţiei în câteva proiecte.

IDE include câteva modele de proiecte destinate să sprijine diferite tipuri de dezvoltări, inclusiv aplicaţii web, aplicaţii generale Java. Setul IDE de modele standard de proiecte generează automat un script Ant si propietăţi. IDE mai conţine şi modele de proiecte free-form care pot fi utilizate pentru a pune bazele unui proiect pe un script Ant deja existent. În plus faţă de Ant, IDE sprijina Maven, un sistem de build si management al proiectelor(open source build management tool). Maven utilizează un proiect model obiect (POM-project object model) care descrie un set de standarde pe care toate proiectele care folosesc Maven le urmăresc pentru a permite coerenţa între proiecte. Se poate crea cu uşurinţă un proiect Maven, prin alegerea unui model (şablon) de proiect şi oferii câteva detalii despre proiect.

3.1. Folosirea modelelor de proiecte Java

Există facilitatea de a crea o aplicaţie Java utilizând unul din şabloanele disponibile date de către IDE. Pentru fiecare tip de aplicaţie Java, IDE oferă două tipuri de modele de proiect:
-Standard templates (Modele Standard) – Modele în care IDE controlează toate sursele şi setările classpath, compilarea, rularea şi depanarea.
-Free-form templates (Modele free-form) – Modelele în care scriptul tău Ant controlează toate setările classpath, compilarea, rularea şi depanarea.

3.1.1 Standard Project Templates

Cu modelele de proiecte standard, IDE controlează toate aspectele legate de cum este construită, rulată şi depanată aplicaţia. Se poate seta un folder sursa al proiectului, classpath şi alte setări ale proiectului când se creează proiectul şi in Project Properties dialog box. IDE generează un Ant build sript în care toate setările sunt stocate.
IDE conţine următoarele modele standard:

- Standard Java Applications
- JavaFX Applications
- Web Applications
- EJB Modules
- Enterprise Applications
- Enterprise Applications Clients
- NetBeans Modules

3.1.2. Free-Form Templates

Cu modelele de proiecte free-form, IDE se bazează pe scriptul Ant existent pentru instrucțiuni privind modul de a compila, de a rula, și depana aplicații.
Setările configurate de utilizator în expertul New Project, atunci când se creează un proiect, precum şi în caseta de dialog Project Properties, sunt utilizate pentru a spune IDE-ului cum gestionează scriptul Ant codul sursă și cum trebuie să fie în concordanță cu setările din script-ul Ant.

De exemplu, toate elementele classpath-urilor sunt manipulate de Ant script. Când se declară classpath-ul pentru un proiect free-form, utilizatorul este cel care îi spune IDE-ului ce clase vor deveni disponibile pentru finalizarea codului şi pentru refactorizare. Aceste setări nu afectează classpath-ul curent, folosit pentru compilarea sau rularea codului sursă.

Proiectele free-form pot conţine tot atâtea foldere sursă precum script-ul Ant, în funcţie de cât este acesta configurat să gestioneze. Dacă script-ul Ant nu conține obiective pentru toate acțiunile IDE, cum ar fi depanarea și rularea proiectului dumneavoastră, puteți scrie cu ușurință obiective Ant pentru aceste acțiuni.

Distribuţia standard a IDE-ULUI conţine următoarele modele de proiecte free-form Java:

- Java Free-Form Project – Un proiect free-form conţine unul sau mai multe rădacini sursa (source roots)
- Web Free-Form Project – Un proiect free-form conţine o aplicaţie web şi optional alte rădăcini sursa Java (Java source roots)

3.2. Importul unui proiect Eclipse sau JBuilder

Se pot importa proiecte create în alte medii IDE cum ar fi Eclipse sau JBuilder, în NetBeans IDE. Funcţionalitatea de a importa un proiect Eclipse este de a include în NetBeans-ul standard distribuţia IDE. Pentru proiectele Eclipse, se poate importa un set de proiecte dependente dintr-un spaţiu de lucru Eclipse sau să se importe un singur proiect ignorând dependenţele.

Pentru proiectele JBuilder, trebuie sa fie instalat plugin-ul JBuilder Project Importer din NetBeans Update Center.

4. Implementarea interfeţelor grafice Java

În aplicațiile Java, componentele care conțin interfață grafică GUI (Graphical User Interface) sunt depozitate în containere numite modele. Limbajul Java oferă un set de componente cu ajutorul cărora pot fi construite interfețele grafice pentru utilizatori (GUI).

Mediul de dezvoltare integrat *IDE GUI Builder* vă ajută în proiectarea și construirea modelelor Java prin furnizarea unor serii de instrumente care simplifică procesul.

În figura 1 este reprezentată o interfață grafică Java realizată pentru o aplicaţie mobilă în mediul de dezvoltare NetBeans IDE:

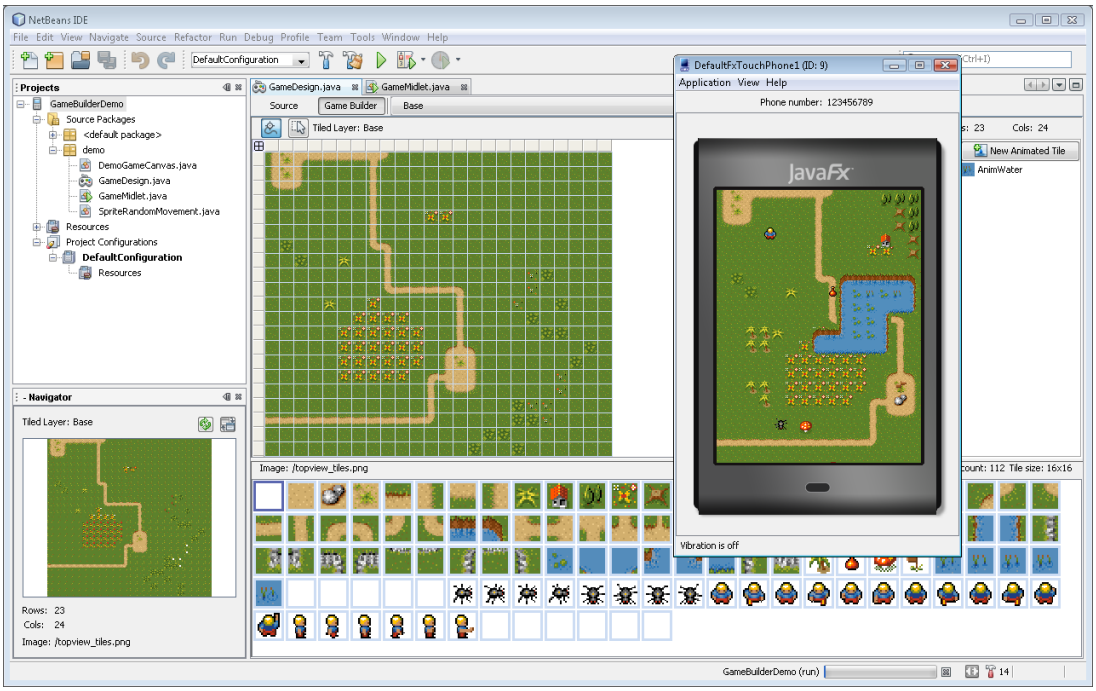

Figura 1. Interfaţa grafică Java

4.1. Instrumentele mediul de dezvoltare integrat Java GUI

IDE-ul oferă mai multe instrumente pentru a simplifica procesul de construire GUI:

- *GUI Builder*. Spațiul de lucru primar în care proiectarea GUI are loc în IDE. GUI Builder vă permite să stabiliţi modele prin plasarea componentelor, în cazul în care le doriți, și prin furnizarea de feedback-ul vizual sub formă de orientări.
- *Fereastra Navigator*. Afișează o ierarhie de tip arbore a tuturor componentelor deschise în modelul curent. Elementele afișate includ componente vizuale și containere, cum ar fi butoane, etichete, meniuri și panouri, precum şi componente non-vizuale cum ar fi cronometre și surse de date.
- *Fereastra Palette*. O listă care conține toate componentele care pot fi adăugate în modele. Puteți personaliza fereastra pentru a afișa conținutul său numai ca pictograme, sau ca pictograme cu nume de componente.
- *Fereastra Properties*. Afișează setările editabile pentru componenta selectată.

- *Expert Conexiune*. Asistă în stabilirea evenimentelor dintre componentele dintr-un model fără a fi nevoie de a scrie manual codul.
- Manager. Vă permite să adăugați, eliminați și să organizaţi componente de ferestre, cum ar fi componente Swing, componente AWT, aşezarea componentelor în pagină (Layouts) și *Beans*.

În plus, IDE-ul oferă suport pentru *Beans Binding* ce oferă o modalitate de a sincroniza valorile diferitelor proprietăți ale claselor *Beans*. Acest suport simplifică, de asemenea, crearea de aplicaţii de baze de date desktop.

4.1.1. Utilizarea *GUI Builder*

GUI Builder este un instrument pentru proiectarea interfeţelor grafice pentru utilizatori. Pe măsură ce creați și modificați interfața grafică, IDE-ul generează în mod automat codul de Java pentru punerea în aplicare a interfaţei.

Când deschideți un model GUI, IDE-ul afișează într-o filă Editor, cu butoane de comutare care vă permit comutarea între puncte de vedere fereastra cu codul sursă - *Source* şi fereastra ce vă afisează componentele grafice - *Design*. Fereastra *Design* vă permite să lucraţi cu modele ale GUI, în timp ce fereastra *Source* vă permite să editaţi direct codul sursă al modelului care urmează să fie editat.

Componentele sunt de obicei adăugate într-un model şi se foloseşte fereastra *GUI Builder* pentru aranjarea acestora în spațiu de lucru. În timp ce lucrați, *Builder GUI* afișează automat linii directoare sugerând o aliniere și ancorare pentru componentele pe care le adăugați.

4.1.2. Utilizarea *Layout Managers*

Managerul de aspect (Layout Managers) vă permit să controlaţi modul în care componentele vizuale sunt aranjate în modelele GUI prin determinarea mărimii şi a poziției elementelor în cadrul containerelor. Aceasta se realizează prin punerea în aplicare a interfeței *Layout Manager*.

În mod implicit, modelel noi sunt create cu constructorul GUI folosind aspectul *GroupLayout* manager. IDE-ul are suport special pentru *GroupLayout* numit *Free Design*. Acesta vă permite să stabiți, folosind ca mod automat liniile directoare vizuale, aranjarea componentelor în model.

Lucrul cu managerul de aranjare sugerează alinierea optimă și distanța dintre componente. În timp ce lucrați, GUI Builder traduce deciziile de proiectare într-un GUI funcțional, fără a necesita să specificați un manager de aspect. Deoarece *Free Design* folosește un model de aspect dinamic, ori de câte ori redimensionam modelul, GUI ajustează modelul pentru a potrivi modificările fără a schimba relațiile dintre componente.

Puteți combina containerele *FreeDesign* și containere cu ajutorul altor manageri de aspect împreună în aceeași model.

4.2. Dezvoltarea de aplicaţii GUI

Pentru ca aplicațiile pe care le creați sa funcţioneze şi în afara mediului de dezvoltare IDE, ar putea fi necesară includerea unor fișiere JAR suplimentare, în momentul în care implementați aplicația.

Pentru a vă asigura că aplicaţia GUI poate utilizează aceste biblioteci în timpul rulării, IDE-ul copiază automat fișierele bibliotecă JAR (și orice alte fișiere JAR din proiect) în folderul dist/lib ori de câte ori va rula proiectul. IDE-ul adaugă, de asemenea, fiecare dintre fișierele JAR la elementul Class-Path în fișierul MANIFEST.MF JAR al aplicației.

După ce ați distribuit o arhivă a apicaţiei dumneavoastră GUI, aplicaţia dumneavoastră poate fi rulată în afara IDE-ului, din linia de comandă.

5. Dezvoltarea aplicaţiilor enterprise

O aplicaţie enterprise reprezintă o colecție de aplicații web și module EJB (Enterprise Java Beans), care sunt configurate pentru a lucra împreună, atunci când sunt implementate la un server de aplicații Java EE. Aplicaţiile enterprise conțin informații despre modul în care modulele lucrează împreună.

Acestea cuprind, de asemenea, informații despre cum lucrează modulele cu serverul de aplicații la care aplicaţia este implementată. Spre exemplu, în cazul în care orice entitate Bean foloseşte un manager de tranzacţii, aplicația enterprise spune serverului de aplicație ce tranzacții sunt necesare.

O aplicaţie enterprise nu are fișierele sursă proprii. Acesta conține doar implementarea descriptorilor și alte fișiere de configurare. La compilare, fișierele arhivă (fișiere JAR și fișierele WAR), pentru fiecare dintre modulele aplicaţiei enterprise sunt construite și asamblate într-un fișier AER (Enterprise Archive). Acest fișier este implementat la server-ul de aplicații.

5.1. Adăugarea de module proiectului

Module EJB sau aplicații web pot fi implementate fie independent, fie ca o parte a unei aplicații enterprise. De asemenea, clienții de aplicaţii Java pot fi clinţi Java normali sau pot fi client Java EE. Adăugarea unui modul EJB, modul web sau modul aplicaţie client la o aplicaţie enterprise vă permite să configuraţi modul în care aceste module interacționează unele cu altele.

Când adăugați un modul la o aplicație enterprise, IDE-ul face urmatoarele:

- Listează modulul într-un nod Java EE Module, în aplicația enterprise.
- Adaugă modulul proiectului în lista proiectului aplicaţiei enterprise, necesară proiectului. Acest lucru înseamnă că modulul proiectului este construit de la zero de fiecare dată când creaţi un proiect de aplicaţie enterprise.
- Adaugă un modul referinţă aplicaţiei enterprise cu descriptori generali de implementare.
- Adaugă în fisierul EAR (Enterprise Archive) al aplicaţiei enterprise modulul de export (fişier JAR sau fişier WAR) când proiectul este finalizat.

5.2. Adăugarea de resurse proiectului

Când contruiţi o aplicaţie enterprise ca un fișier EAR, pachetul implicit include doar acele fișiere care sunt parte a aplicaţiei enterprise. Pentru a adăuga un fișier, care este în afara aplicaţiei enterprise, în pachetul EAR, utilizați caseta de dialog *Proprietăți Proiect*. Resursele externe ar putea include fișiere JAR, biblioteci sau alte proiecte care îşi au locația în afara aplicaţiei enterprise.

5.3 Editarea descriptorilor de implementare

O aplicaţie enterprise include în general două tipuri de descriptori de implementare. În aplicaţiile entrreprise Java EE, fișierele descriptor sunt opționale și puteți folosi adnotări pentru a specifica majoritatea setărilor și a resurselor de implementare. Aplicaţiile enterprise J2EE 1.4 necesită, în general, următoarele fișiere de descriptori:

- Un descriptor general de implementare J2EE care configurează setările de implementare cu privire la orice implementare J2EE. Acesta descrie aplicația enterprise, ale cărei

componente sunt folosite, cum sunt legate componentele între ele și care resurse sunt folosite. În general, descriptor de implementare al aplicatiilor enterprise se numește application.xml.

- Un descriptor de implementare-server care configurează setările de implementare pentru serverul de aplicații la care sunteți conectați. Dacă implementați aplicația la Sun Java System Application Server sau GlassFish, IDE-ul oferă un editor grafic pentru descriptorul de implementare și actualizează automat descriptorul de implementare, pe măsură ce editați fișierele. Pentru toate celelalte servere de aplicații, trebuie să scriți descriptorii de implementare-server specifici.

Intrările în aceste fișiere sunt generate automat atunci când adăugați un modul la aplicația enterprise. Ambii descriptori de implementare sunt fișiere XML care sunt stocate în folder META-INF al aplicației enterprise. Aveți posibilitatea să editați ambele fișiere în *Editor Sursă*. Fișierul GlassFish-application.xml poate fi, de asemenea, editat într-un editor visual.

5.4. Verificarea aplicațiilor enterprise

Înainte de a implementa o aplicație entreprise sau orice modul web sau EJB independente pentru serverul de aplicații, trebuie să-l verificați, pentru a vă asigura că este pus în aplicare în mod corespunzător în Java EE. Când serverul țintă pentru proiect este sistemul Sun Java Application Server sau server-ul de aplicatii GlassFish, puteți utiliza instrumentul de verificare pentru a valida atât descriptorii Java EE, cât și implementarea aplicație împotriva fișierelor corespunzătoare DTD și erori sau avertismente în cazul în care un modul sau aplicație nu este compatibilă (Figura 2). Puteți verifica descriptori de implementare în EAR, WAR, RAR și fișiere JAR.

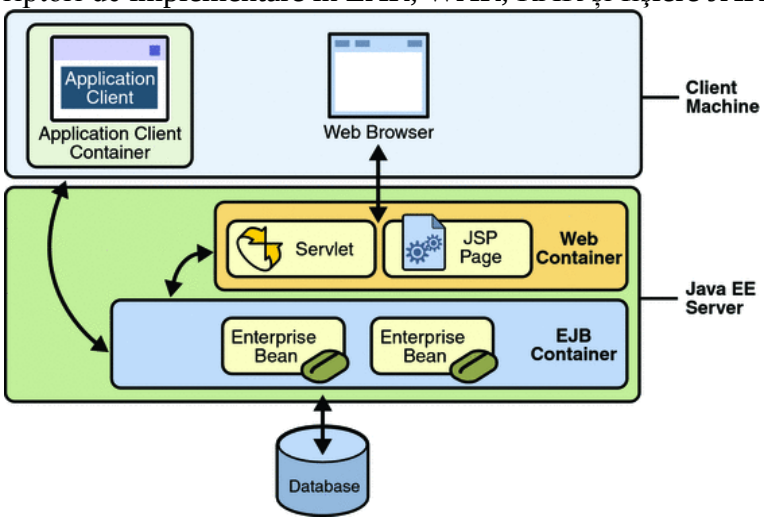

Figura 2. Exemplu Aplicatie Enterprise

5.5. Implementarea aplicațiilor enterprise

Cu excepția cazului în care modulele EJB și modulele de aplicații web sunt concepute pentru a fi utilizate ca module independente, acestea ar trebui să fie implementate printr-o aplicație enterprise care sa le conțină. Dacă executați comenzile *Run* sau *Deploy* pe un modul individual, care este parte a unei aplicații enterprise, IDE-ul implementează numai acel modulul. Orice modificări pe care le-ați făcut în celelalte module din cadrul proiectului nu sunt propagate la server.

Bibliografie:

[1] Developing Applications with NetBeans, IDE Release 7.4
[2] ***, https://netbeans.org/about/history.html
[3] ***, https://netbeans.org/community/releases/roadmap.html