INTEROPERABILITATEA DINTRE JDBC și HIBERNATE PRIN INTERMEDIUL VAADIN

Cristian CEMÎRTAN

Departamentul Informatică, I-2101, Facultatea de Matematică și Informatică,
Universitatea de Stat din Moldova, Chișinău, Moldova

Autorul corespondent: Cristian Cemîrtan, cemirtan.cristian@outlook.com

Rezumat. *În lucrarea de față se examinează nivelul de lucru al stilurilor JDBC și Hibernate prin interacțiune cu componentele cadrului Vaadin. Această aplicație web demonstrează nivelul de lucru configurabil în Vaadin. Hibernate permite tratarea înregistrărilor din baza de date ca instanțe de clase și ne eliberează de mediul cu restricții tehnice, unde se lucrează numai cu SQL. Prin utilizarea JDBC demonstrăm cât de dificil și voluminos este de lucrat numai cu SQL pentru că în SQL pot apărea multe dialecte ce conduc la apariția inconsistenței în mediul de dezvoltare.*

Cuvinte cheie*: sql, mapare entitate – relație, limbaj, aplicație web, bază de date, java.*

Introducere

În epoca contemporană a dezvoltării aplicațiilor web, există două modalități cunoscute pentru a gestiona o bază de date: scrierea și executarea interogărilor utilizând limbajul SQL, și persistarea obiectelor POO prin intermediul mapării obiect – relație. Maparea obiect – relație este o tehnică de programare ce permite administrarea relațiilor din baza de date utilizând limbajul de programare orientat pe obiect (în cazul nostru Java), fără a fi dependent de restricțiile sintactice a limbajului SQL.

Vaadin

Interfața grafica utilizată în dezvoltarea aplicației web implică aplicarea cadrului Vaadin. Vaadin este o platformă de dezvoltare a aplicațiilor web care ne permite de creat aplicații web cu limbajul de programare Java, eliminând necesitatea utilizării limbajului de machetare hipertext și a limbajului de programare ECMAscript. Originea fondării cadrului Vaadin este Finlanda. Spre deosebire de bibliotecile ECMAscript și plugin-urile specifice browser-ului, Vaadin oferă o arhitectură orientată spre server bazată pe Jakarta Enterprise Edition [1].

Jakarta Enterprise Edition

Utilizarea Jakarta Enterprise Edition permite ca partea principală a logicii aplicației să fie executată pe partea serverului. Vaadin folosește propriul set de componente web sau biblioteci ECMAscript pentru a reda elementele interfeței cu utilizatorul și a permite lui de a interacționa cu serverul. Vaadin adaugă validarea datelor suplimentare pe partea serverului: aceasta rezolvă problemele de securitate asociate cu posibilitatea falsificării datelor sau codului ECMAscript. În consecință, dacă datele primite de la browser se modifică sau devin corupte, serverul, după ce a stabilit acest lucru, nu transmite cereri.

JDBC

Java Database Connectivity, este o interfață de programare a aplicației ce permite conexiunea la o bază de date. Știind că există diverse implementări de a găzdui o bază de date, JDBC suportă încărcarea piloților pentru a putea crea conexiuni conforme cu specificațiile tehnice a unei bazei de date [2].

Hibernate

Dezvoltată de compania americană Red Hat în anul 2001, Hibernate este o bibliotecă gratuită pentru limbajul de programare Java. Este concepută pentru a realiza operații de mapare obiect – relație, prin implementarea specificației standarde Jakarta Persistence. Jakarta Persistence specifică

gestionarea datelor relaționale în aplicațiile Java, iar Hibernate ușurează realizarea conexiunii la baza de date, prin intermediul unui pilot JDBC. Pentru a fuziona datele modificate din programul Java în baza de date, Hibernate generează și execută un scenariu SQL [3–4].

Spring Boot

Fiind o extensie a cadrului Spring pentru aplicațiile Java, este utilizată pentru a găzdui, în mod accesibil, un server ce necesită un număr minim de configurații care prevăd utilizarea resurselor de calcul.

Aplicația Vaadin și gestionarea bazei de date

În rularea aplicației web, vom gestiona baza de date „Magazin Hardware”, diagrama căreia este reprezentată în figura 1. Din baza de date menționată, aplicația se va baza pe relațiile `RAM` și `Firma`.

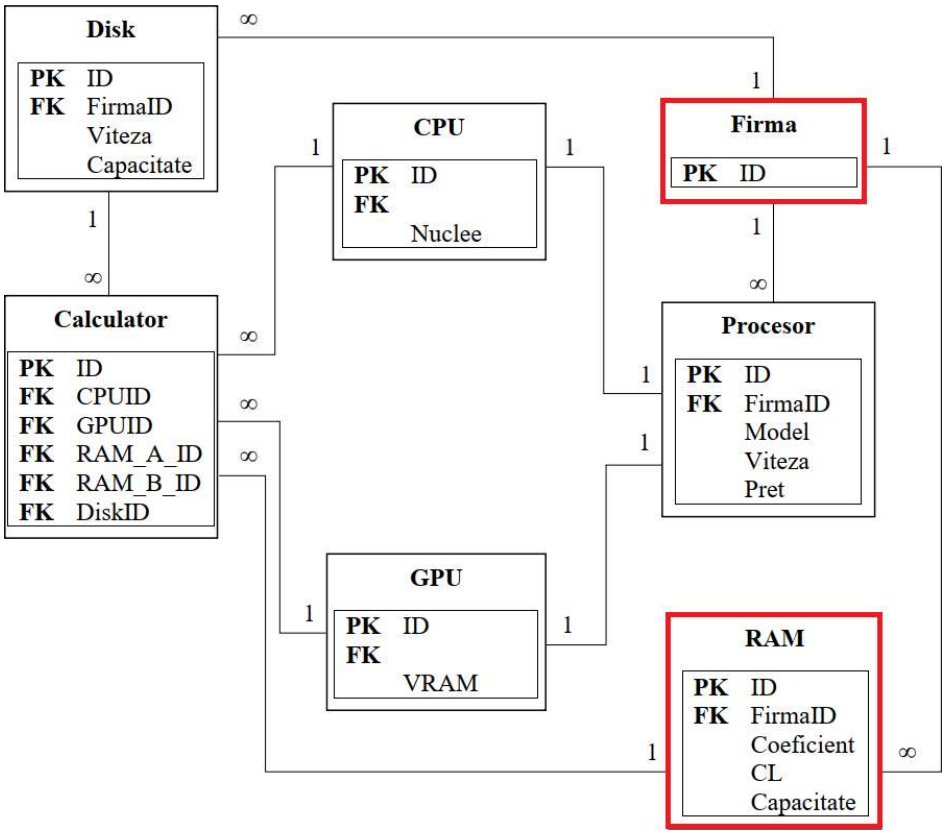

Figura 1. Diagrama bazei de date

Pagina aplicației web este divizată în următoarele secțiuni: sistemul de autentificare (dispare la o autentificare cu succes), butonul de actualizare a conținutului paginii, formularul de manipulare a unei înregistrări de date din relația `RAM`, grupul de butoane radio pentru a permite modul de lucru a bazei de date, între JDBC și Hibernate, cât și două componente ce afișează conținutul relației `RAM`: grila specializată (vizibilă în modul Hibernate) și zona de text (vizibilă în modul JDBC) [5].

În formularul de manipulare, câmpul de selecție Firma conține valorile culese din tabela omonimă, utilizând tehnologia Hibernate sau JDBC, la discreția clientului. De asemenea, manipularea relației `RAM` poate fi realizată utilizând tehnologia aleasă la discreție.

În aplicația web, sunt apelate următoarele două metode statice, în modul SQL: `executeQuery` din clasa `Core` și `executeBatches` din clasa `Helper`. Prima metodă transformă setul de rezultate `ResultSet` în forma textuală [6]. În a doua metodă, se execută o interogare pentru orice rând din tabloul bidimensional `Object[][]`.

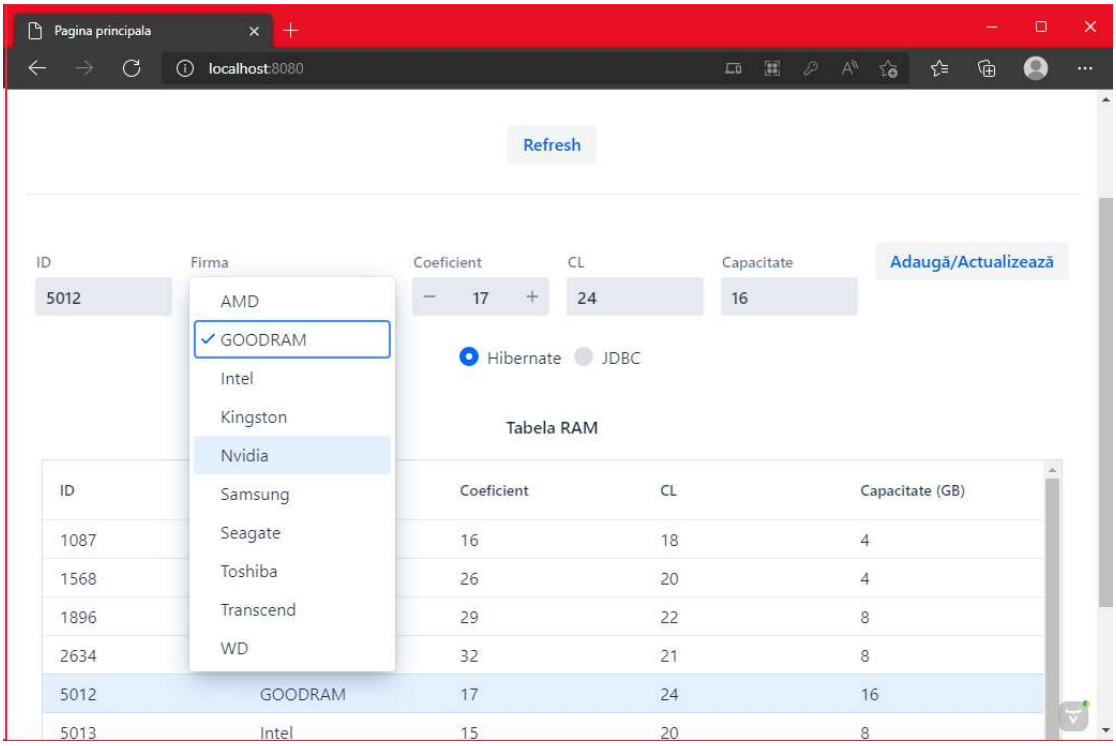

Figura 2. Pagina principală a aplicației Vaadin

În lucrare, când are loc procesul de actualizare a conținutului paginii, fie automat sau manual, se execută următorul fragment de cod dacă este setată tehnologia Hibernate:

```
grid.setItems(session.createNativeQuery("SELECT * FROM RAM", RAM.class).list());

if (b)  // adevarat, daca manual
	firmaSelect.setItems(session.createNativeQuery("SELECT * FROM Firma",
Firma.class).list());
```
(1)

În modul realizat cu conectorul JDBC:

```
textAreaList.setValue(Helper.formatQuery(c, "SELECT * FROM RAM"));
var rs = Core.executeQuery(c, "SELECT * FROM FIRMA");

if (b)
{
	var list = new ArrayList<Firma>();

	while (rs.next())
		list.add(new Firma(rs.getString("ID")));

	firmaSelect.setItems(list);
}
```
(2)

În primele două secvențe de cod sursă a fost menționată reîncărcarea conținutului paginii web, unde prima secvență reprezintă partea realizată în Hibernate, pe când a doua este o conexiune cu conectorul JDBC.

În prima secvență de cod, pentru actualizarea ambelor grile `RAM` și a câmpului de selecție `Firma`, se execută o singură interogare de selecție pentru relațiile `RAM` și `Firma`, iar rezultatele le vom obține în Hibernate sub formă de listă și vor fi anexate în ambele componente Vaadin.

În ce privește a doua secvență, rezultatele le vom obține sub forma textuală. Pentru interogarea cu `RAM`, rezultatul se anexează în zona de text, iar pentru rezultatul interogării cu `Firma` vom obține

un set de rezultate prin `ResultSet`, și în continuare iterăm tuplurile din `ResultSet` că pentru orice tuplu într-o instanță nouă de `ArrayList` [7], cu parametrul generic `Firma`, să adăugăm o nouă instanță `Firma` cu denumirea tuplului. După iterare se anexează în câmpul de selecție `Firma` instanța `ArrayList` recent creată.

Pentru secvența de cod:

```
session.beginTransaction();

session.merge(new RAM(
       idField.getValue(),
       firmaSelect.getValue(),
       coeficientField.getValue(),
       clField.getValue(),
       capacitateField.getValue()));

session.getTransaction().commit();
```

(3)

si secvența:

```
if (
       Helper.executeBatches(c, "UPDATE RAM SET FirmaID = ?, Coeficient = ?, CL =
?, Capacitate = ? WHERE ID = ?",
       new Object[][]
       {
              {
                     firmaSelect.getValue(),
                     coeficientField.getValue(), clField.getValue(),
                     capacitateField.getValue(), idField.getValue()
              }
       },
       new Integer[] { Types.VARCHAR, Types.INTEGER, Types.INTEGER,
Types.INTEGER, Types.INTEGER, Types.INTEGER }
)[0] == 0)
{
       Helper.executeBatches(c, "INSERT INTO RAM VALUES (?, ?, ?, ?, ?)",
              new Object[][]
              {
                     {
                            idField.getValue(), firmaSelect.getValue(),
                            coeficientField.getValue(), clField.getValue(),
                            capacitateField.getValue()
                     }
              },
              new Integer[] { Types.INTEGER, Types.VARCHAR, Types.INTEGER,
Types.INTEGER, Types.INTEGER, Types.INTEGER }
       );

       ...
}
```

(4)

In secvența a treia de cod se execută o tranzacție cu Hibernate prin metoda `beginTransaction()` [8]. Vom fuziona o nouă instanță `RAM` cu valorile obținute din formularul de adăugare în sesiune Hibernate și prin comiterea tranzacției redirecționăm conținutul instanței în baza de date. În secvența a patra ilustrăm o interogare de actualizare SQL, prin conectorul JDBC, a relației `RAM`.

Spre deosebire de metoda realizată în Hibernate specificăm valorile tipurilor de date a coloanelor tuplului și dacă numărul de tupluri modificate este zero atunci se va executa o interogare de adăugare.

Concluzii

Realizarea acestei aplicații se bazează pe principiile programării orientate pe obiect, astfel putem concluziona că utilizarea cadrului Hibernate ameliorează mult productivitatea creării codului sursă, dar și simplificarea interogărilor pentru administrarea bazei de date „Magazin Hardware”.

Referințe

1. Vaadin Team. *Book of Vaadin*. Turku: Vaadin Ltd., 2019.
2. Oracle. *Java JDBC API* [online]. [accesat 12.03.2022]. Disponibil: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/.
3. The Hibernate Team, The JBoss Visual Design Team. *HIBERNATE – Relational Persistence of Idiomatic Java* [online]. 2013. [accesat 11.03.2022]. Disponibil: https://docs.jboss.org/hibernate/core/4.3/manual/en-US/html_single/.
4. Arjam Tijms. *Transistion from Java EE to Jakarta EE* [online]. 2020. [accesat 12.03.2022]. Disponibil: https://blogs.oracle.com/javamagazine/post/transition-from-java-ee-to-jakarta-ee.
5. Vaadin Ltd., *Vaadin UI Components*. [online]. [accesat 09.03.2022]. Disponibil: https://vaadin.com/components.
6. Oracle. *Interface ResultSet* [online]. [accesat 10.03.2022]. Disponibil: https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html.
7. Oracle. *Class ArrayList<E>* [online]. [accesat 13.03.2022]. Disponibil: https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html.
8. JBoss. *Interface Transaction* [online]. [accesat 13.03.2022]. Disponibil: https://docs.jboss.org/hibernate/orm/3.5/api/org/hibernate/Transaction.html.