Sabou Alexandru Mihai

Grupa 254

Tema: EJB-JPA

Descrierea temei:

Elaborarea unei aplicații care să folosească EJB, JPA, Servleturi și eventual JSP. Aplicația va trebui să conțină un server ce să gestioneze minimum două tabele în DB care să aibă relații între ele. Mai trebuie să conțină doi clienți, unul care să apeleze serverul prin JNDI, celălalt să folosească injectarea EJB. Aplicația va fi instalată atât pe WildFly (JBoss) cât și pe GlassFish.

Descrierea aplicației:

Aplicație de gestiune a cumpărăturilor:

- Un utilizator se poate loga la aplicație
- Dacă autentificarea eșuează, se va afișa un mesaj corespunzător
- Se poate creea un nou utilizator, iar după se poate loga la aplicație
- Utilizatorului logat îi va apărea lista de cumpărături
- Se va prezenta un form, unde utilizatorul logat poate introduce date pentru noile cumpărături pe care le va adăuga
- Dacă formul de adăgare noi cumpărături este completat partial, un mesaj de atenționare va fi prezentat
- Cumpărăturile au un buton de ștergere din listă
- Utilizatorul se poate deloga de la aplicație

Tehnologii:

- Java 17 (OpenJDK-17.0.2)
- Gradle 7.3.3
- Jakarta EE 9
- MySql 8.0.30

Servere:

- GlassFish: 6.2.5
- WildFly: wildfly-preview-27.0.0.Beta1
- MySQL în XAMPP-8.1.10

Dependințe și versiuni:

Dependințele și versiunile folosite pentru dezvoltarea aplicației enterprise sunt în fișierele build.gradle de la fiecare proiect în parte (server și client). Dacă la server nu există probleme de build sau probleme la rularea aplicației în GlassFish sau WildFly, la client se va adăuga o dependință în funcție de variabila ”server”. De asemenea, pentru evidențierea resurselor necesare

pentru diferitele servere de aplicații, se va exclude din war resursele folosite de către celălalt server de aplicație.

Acest lucru se poate observa în fotografiile următoare:

- Server:

```
dependencies {
    // https://mvnrepository.com/artifact/mysql/mysql-connector-java
    implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.30'
    // https://mvnrepository.com/artifact/org.glassfish.jaxb/jaxb-runtime
    implementation group: 'org.glassfish.jaxb', name: 'jaxb-runtime', version: '4.0.1'
    // https://mvnrepository.com/artifact/org.hibernate/hibernate-core-jakarta
    implementation group: 'org.hibernate', name: 'hibernate-core-jakarta', version: '5.6.14.Final'
    // https://mvnrepository.com/artifact/jakarta.persistence/jakarta.persistence-api
    implementation group: 'jakarta.persistence', name: 'jakarta.persistence-api', version: '3.1.0'
    // https://mvnrepository.com/artifact/jakarta.ejb/jakarta.ejb-api
    implementation group: 'jakarta.ejb', name: 'jakarta.ejb-api', version: '4.0.1'
    // https://mvnrepository.com/artifact/jakarta.annotation/jakarta.annotation-api
    implementation group: 'jakarta.annotation', name: 'jakarta.annotation-api', version: '2.1.1'
    // https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api
    compileOnly group: 'jakarta.servlet', name: 'jakarta.servlet-api', version: '6.0.0'
    // https://mvnrepository.com/artifact/org.slf4j/slf4j-api
    implementation group: 'org.slf4j', name: 'slf4j-api', version: '2.0.3'
    // https://mvnrepository.com/artifact/org.slf4j/slf4j-simple
    implementation group: 'org.slf4j', name: 'slf4j-simple', version: '2.0.3'
    // https://mvnrepository.com/artifact/org.projectlombok/lombok
    compileOnly group: 'org.projectlombok', name: 'lombok', version: '1.18.24'
    annotationProcessor group: 'org.projectlombok', name: 'lombok', version: '1.18.24'
}
```

- Client:

```
14
15  war {
16      if (server == "wildfly") {
17          rootSpec.exclude('**/glassfish/**')
18      }
19      if (server == "glassfish") {
20          rootSpec.exclude('**/wildfly/**')
21      }
22  }
23
24  dependencies {
25      if (server == "wildfly") {
26          // https://mvnrepository.com/artifact/org.wildfly/wildfly-ejb-client-bom
27          implementation group: 'org.wildfly', name: 'wildfly-ejb-client-bom', version: '27.0.0.Final', ext: 'pom'
28      }
29      // https://mvnrepository.com/artifact/jakarta.annotation/jakarta.annotation-api
30      implementation group: 'jakarta.annotation', name: 'jakarta.annotation-api', version: '2.1.1'
31      // https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api
32      compileOnly group: 'jakarta.servlet', name: 'jakarta.servlet-api', version: '6.0.0'
33      // https://mvnrepository.com/artifact/org.slf4j/slf4j-api
34      implementation group: 'org.slf4j', name: 'slf4j-api', version: '2.0.3'
35      // https://mvnrepository.com/artifact/org.slf4j/slf4j-simple
36      implementation group: 'org.slf4j', name: 'slf4j-simple', version: '2.0.3'
37      // https://mvnrepository.com/artifact/org.projectlombok/lombok
38      compileOnly group: 'org.projectlombok', name: 'lombok', version: '1.18.24'
39      annotationProcessor group: 'org.projectlombok', name: 'lombok', version: '1.18.24'
40  }
41
```

Structura:

1. **com.tpjad.ejbjpa.server**: acesta are 3 seturi de directory/fișiere principale (persistence, src, build.gradle). În directorul persistence, putem observa alte două sudirectoare glassfish și wildfly care conțin câte un fișier persistence.xml unde este definit unitatea de persistență pentru fiecare AS în parte. În src/main/java se află pachetul de bază com.tpjad.ejbjpa.groceries care are la rândul lui alte pachete (beans, client, domain, interfaces, utils). În pachetul domain sunt definite entity-urile folosite la mapările pe tabel, interfaces conține interfețele folosite de modulul EJB, beans implementările interfețelor, client conține injectarea EJB, iar în utils diferite utilități. În resources\META-INF se află persistence.xml pentru unitatea de persistență.
2. **com.tpjad.ejbjpa.client**: src și build.gradle. Pachetul de bază com.tpjad.ejbjpa.groceries are 4 pachete: glassfish unde sunt prezentați clienții GlassFish Servlet care apelează prin JNDI server-ul, interfaces interfețele folosite de modulul EJB, utils diferite clase utilitare, wildfly clientii WildFly Servlet
3. **run.bat**: script utilitar folosit la build și deploy pe diferitele AS-uri

Descrierea servleturilor:

- **LoginServlet**:
  - responsabil cu afișarea formului de autentifcare sau creare cont nou
  - afișare mesaj de atenționare dacă logarea eșuează
  - pentru clientul local, acesta conține referința EJB la ILoginService
  - pentru clientul remote, se face un lookup prin JNDI la interfața ILoginService
  - protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
  - protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
- **LogoutServlet**:
  - responsabil cu delogarea utilizatorului (ștergerea acestuia din HttpSession) și redirectarea către pagina de login (LoginServlet)
  - protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
- **GroceriesServlet**:
  - responsabil cu afișarea listei de cumpărături pentru utilizatorul logat
  - pentru clientul local, acesta conține referința EJB la IGroceryServie
  - pentru clientul remote, se face un lookup prin JNDI la interfața IGroceryService
  - protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
- **GroceryAddServlet**:
  - responsabil cu adăugarea de cumpărături noi în listă
  - validare cumpărături (fielduri umplute) și afișare mesaj corespunzător
  - pentru clientul local, acesta conține referința EJB la IGroceryServie
  - pentru clientul remote, se face un lookup prin JNDI la interfața IGroceryService

    - protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
- **GroceryDeleteServlet**:
    - responsabil cu ștergerea cumpărăturilor
    - pentru clientul local, acesta conține referința EJB la IGroceryServie
    - pentru clientul remote, se face un lookup prin JNDI la interfața IGroceryService
    - protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException

Descrierea beanurilor:

- **LoginBean** este implementarea interfeței locale ILoginService și remote ILoginServiceR
    - void login(String username, String password);
    - boolean isLoggedIn();
    - void create(UserDTO userDTO);
    - UserDTO findByUsername(String username);
    - UserDTO getLoggedUser();
- **GroceryBean** este implementarea interfeței locale IGroceryService și remote IGroceryServiceR
    - void create(GroceryDTO groceryDTO);
    - void delete(final Long id);
    - List<GroceryDTO> findAll(final String user);

Descrierea claselor din domeniul aplicației:

- **UserEntity**:

```
@Entity
@Table(name = "users")
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(unique = true)
    private String username;
    private String password;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "userEntity")
    @ToString.Exclude
    private Collection<GroceryEntity> groceries;
}
```

    - Adnotarea @GeneratedValue indică faptul că la fiecare user nou adăugat, id-ul se va incrementa automat
    - Adnotarea @Column(unique = true) indică faptul că această coloană conține valori unice
    - Adnotarea @OneToMany indică faptul că userEntity de la instanțele UserEntity va fi folosit pentru obținerea colecției

- **GroceryEntity**:

```
@Entity
@Table(name = "groceries")
public class GroceryEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne
    private UserEntity userEntity;

    private String description;
    private String nrItems;
    private String tillDate;
}
```

  - Adnotarea @ManyToOne indică faptul că mai multe instanțe ale lui GroceryEntity referă un UserEntity

Descrierea JNDI-urilor:

- **Pentru serverul WildFly**:

```
public class ContextLookupUtils {
    public static ILoginServiceR lookupLoginServiceJNDIEJBWF() throws NamingException {
        final Hashtable jndiProperties = new Hashtable();
        jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY, "org.wildfly.naming.client.WildFlyInitialContextFactory");
        jndiProperties.put(Context.PROVIDER_URL, "remote+http://localhost:8080");
        final Context context = new InitialContext(jndiProperties);
        return (ILoginServiceR)
                context.lookup("ejb:/com.tpjad.ejbjpa.groceries.server/LoginBean!com.tpjad.ejbjpa.groceries.interfaces.ILoginServiceR?stateful");
    }

    public static ILoginServiceR lookupLoginServiceJNDIWF() throws NamingException {
        final Hashtable jndiProperties = new Hashtable();
        jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY, "org.wildfly.naming.client.WildFlyInitialContextFactory");
        jndiProperties.put(Context.PROVIDER_URL, "remote+http://localhost:8080");
        final Context context = new InitialContext(jndiProperties);
        return (ILoginServiceR)
                context.lookup("com.tpjad.ejbjpa.groceries.server/LoginBean!com.tpjad.ejbjpa.groceries.interfaces.ILoginServiceR");
    }

    public static IGroceryServiceR lookupGroceryServiceJNDIWF() throws NamingException {
        final Hashtable jndiProperties = new Hashtable();
        jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY, "org.wildfly.naming.client.WildFlyInitialContextFactory");
        jndiProperties.put(Context.PROVIDER_URL, "remote+http://localhost:8080");
        final Context context = new InitialContext(jndiProperties);
        return (IGroceryServiceR) context.lookup("com.tpjad.ejbjpa.groceries.server/GroceryBean!com.tpjad.ejbjpa.groceries.interfaces" +
                ".IGroceryServiceR");
    }

    public static IGroceryServiceR lookupGroceryServiceJNDIEJBWF() throws NamingException {
        final Hashtable jndiProperties = new Hashtable();
        jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY, "org.wildfly.naming.client.WildFlyInitialContextFactory");
        jndiProperties.put(Context.PROVIDER_URL, "remote+http://localhost:8080");
        final Context context = new InitialContext(jndiProperties);
        return (IGroceryServiceR) context.lookup("ejb:/com.tpjad.ejbjpa.groceries.server/GroceryBean!com.tpjad.ejbjpa.groceries.interfaces" +
                ".IGroceryServiceR");
    }
```

- **Pentru serverul GlassFish**:

```
public static ILoginServiceR lookupLoginServiceJNDIGF() throws NamingException {
    final Hashtable jndiProperties = new Hashtable();
    jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.enterprise.naming.impl.SerialInitContextFactory");
    final Context context = new InitialContext(jndiProperties);
    return (ILoginServiceR)
            context.lookup("java:global/com.tpjad.ejbjpa.groceries.server/LoginBean!com.tpjad.ejbjpa.groceries.interfaces.ILoginServiceR");
}

public static IGroceryServiceR lookupGroceryServiceJNDIGF() throws NamingException {
    final Hashtable jndiProperties = new Hashtable();
    jndiProperties.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.enterprise.naming.impl.SerialInitContextFactory");
    final Context context = new InitialContext(jndiProperties);
    return (IGroceryServiceR) context.lookup("java:global/com.tpjad.ejbjpa.groceries.server/GroceryBean!com.tpjad.ejbjpa.groceries.interfaces" +
            ".IGroceryServiceR");
}
```

Deploy-ul aplicației:

- **Serverul WildFly**:
  - aici am ales ca să folosesc direct Hibernate, fără un datasource configurat pe serverul de aplicații. Acest lucru s-a făcut prin simpla adăgare a lui mysql-connector-java la dependințele aplicației
  - fișierul unde este marcată unitatea de persistență:

om.tpjad.ejbjpa.groceries.server > persistence > wildfly > persistence.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="https://jakarta.ee/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence https://jakarta.ee/xml/ns/persistence/persistence_3_0.xsd"
             version="3.0">
    <persistence-unit name="ejbjpa">
        <class>com.tpjad.ejbjpa.groceries.domain.UserEntity</class>
        <class>com.tpjad.ejbjpa.groceries.domain.GroceryEntity</class>
        <properties>
            <property name="jakarta.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>
            <property name="jakarta.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/ejbjpa"/>
            <property name="jakarta.persistence.jdbc.user" value="ejbjpa"/>
            <property name="jakarta.persistence.jdbc.password" value="ejbjpa"/>

            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
            <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
            <property name="hibernate.show_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>
```

  - pentru deploy-ul aplicației, am copiat arhiva war generată în %JBOSS_HOME%\standalone\deployments
  - start server: %JBOSS_HOME%\bin\standalone.bat
- **Serverul GlassFish**:
  - aici am ales să configurez datasource-ul pe serverul de aplicații
  - descărcare mysql-connector-java-8.0.30.jar, redenumire în mysql-connector-java.jar și mutare în %GF_HOME%\glassfish\lib
  - executare comanda: asadmin> create-jdbc-connection-pool --restype javax.sql.DataSource --datasourceclassname com.mysql.cj.jdbc.MysqlDataSource --property "url=jdbc\\:mysql\\://localhost\\:3306/ejbjpa" mySqlPool
  - executare commandă: asadmin> create-jdbc-resource --connectionpoolid mySqlPool jdbc/mysql
  - fișierul unde este marcată unitatea de persistență:

om.tpjad.ejbjpa.groceries.server > persistence > glassfish > persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="https://jakarta.ee/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence https://jakarta.ee/xml/ns/persistence/persistence_3_0.xsd"
             version="3.0">
    <persistence-unit name="ejbjpa">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <jta-data-source>jdbc/mysql</jta-data-source>
        <class>com.tpjad.ejbjpa.groceries.domain.UserEntity</class>
        <class>com.tpjad.ejbjpa.groceries.domain.GroceryEntity</class>
        <properties>
            <property name="jakarta.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>
            <property name="jakarta.persistence.jdbc.user" value="ejbjpa"/>
            <property name="jakarta.persistence.jdbc.password" value="ejbjpa"/>

            <property name="hibernate.dialect" value="org.hibernate.dialect.MariaDBDialect"/>
            <property name="hibernate.hbm2ddl.auto" value="create-drop"/>
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.transaction.jta.platform" value="org.hibernate.service.jta.platform.internal.SunOneJtaPlatform"/>
        </properties>
    </persistence-unit>
</persistence>

```

- pentru deploy-ul aplicației am copiat arhiva war în %GF_HOME%\glassfish\domains\domain1\autodeploy
- start server: %GF_HOME%\bin\asadmin.bat start-domain

Descrierea utilitarului run.bat

- acesta se va executa dându-se un parametru în care se specifică server-ul pe care să se facă deploy

run.bat

```bat
set JAVA_HOME=D:\tools\jdk-17.0.2
set GRADLE_HOME=D:\tools\gradle-7.3.3
set JBOSS_HOME=D:\tools\wildfly-preview-27.0.0.Beta1
set GF_HOME=D:\tools\glassfish6

set PATH=%JAVA_HOME%\bin;%PATH%
set PATH=%GRADLE_HOME%\bin;%PATH%
set PATH=%JBOSS_HOME%\bin;%PATH%
set PATH=%GF_HOME%\bin;%PATH%

set SERVER_PROJECT=D:\Master\An1\Sem1\TPJAD\EJB-JPA\com.tpjad.ejbjpa.groceries.server
set CLIENT_PROJECT=D:\Master\An1\Sem1\TPJAD\EJB-JPA\com.tpjad.ejbjpa.groceries.client

for /F "tokens=5 delims= " %%P in ('netstat -ano ^| findstr :8080') do taskkill /PID %%P /F

set SERVER=%1

if not "%SERVER%" == "wildfly" if not "%SERVER%" == "glassfish" (
    echo "No server was specified!"
    exit 1
)

del %SERVER_PROJECT%\src\main\resources\META-INF\persistence.xml

copy %SERVER_PROJECT%\persistence\%SERVER%\persistence.xml %SERVER_PROJECT%\src\main\resources\META-INF

```

- liniile 1-12 fac referire la setarea unor path-uri și adaugarea acestora la variabila de mediu PATH

- linia 14: kill la procesul care rulează pe portul 8080 (pentru o mai ușoară tranziție de la un server de aplicație la altul)
- liniile 16-21: se validează argumentul dat ca parametru la run.bat
- liniile 23-25: în funcție de variabila SERVER se va folosi fisierul persistence.xml

```
27  if "%SERVER%" == "wildfly" (
28      cd %SERVER_PROJECT%  & call gradle clean build
29      del %JBOSS_HOME%\standalone\deployments\com.tpjad.ejbjpa.groceries.server.war
30      copy %SERVER_PROJECT%\build\libs\com.tpjad.ejbjpa.groceries.server.war %JBOSS_HOME%\standalone\deployments
31      cd %CLIENT_PROJECT%  & call gradle clean build -Dserver=%SERVER%
32      del %JBOSS_HOME%\standalone\deployments\com.tpjad.ejbjpa.groceries.client.war
33      copy %CLIENT_PROJECT%\build\libs\com.tpjad.ejbjpa.groceries.client.war %JBOSS_HOME%\standalone\deployments
34      start %JBOSS_HOME%\bin\standalone.bat
35      start "" http://localhost:8080/com.tpjad.ejbjpa.groceries.server/LoginLocalServlet
36      start "" http://localhost:8080/com.tpjad.ejbjpa.groceries.client/LoginWFServlet
37  )
38
39  if "%SERVER%" == "glassfish" (
40      cd %SERVER_PROJECT%  & call gradle clean build
41      del %GF_HOME%\glassfish\domains\domain1\logs\server.log
42      del %GF_HOME%\glassfish\domains\domain1\autodeploy\com.tpjad.ejbjpa.groceries.server.war
43      copy %SERVER_PROJECT%\build\libs\com.tpjad.ejbjpa.groceries.server.war %GF_HOME%\glassfish\domains\domain1\autodeploy
44      cd %CLIENT_PROJECT%  & call gradle clean build -Dserver=%SERVER%
45      del %GF_HOME%\glassfish\domains\domain1\autodeploy\com.tpjad.ejbjpa.groceries.client.war
46      copy %CLIENT_PROJECT%\build\libs\com.tpjad.ejbjpa.groceries.client.war %GF_HOME%\glassfish\domains\domain1\autodeploy
47      start call %GF_HOME%\bin\asadmin.bat start-domain
48      start "" http://localhost:8080/com.tpjad.ejbjpa.groceries.server/LoginLocalServlet
49      start "" http://localhost:8080/com.tpjad.ejbjpa.groceries.client/LoginGFServlet
50  )
```

- linia 28 (respective linia 40): build-ul proiectelor server
- linia 29 (respective liniile 41-42): undeploy-ul (stergerea vechilor arhive war și a logurilor a aplicațiilor server)
- linia 30 (linia 43): copierea arhivei war în serverele de aplicații pentru deploy
- linia 31 (linia 44): build-ul proiectelor client
- linia 32 (linia 45): undeploy-ul (ștergera vechilor arhive war a aplicațiilor cient)
- linia 33 (linia 46): copierea arhivei war în serverele de aplicații pentru deploy
- linia 34 (linia 47): pornirea serverelor de aplicații
- liniile 35-36 (liniile 48-49): deschiderea in browserul implicit aplicația dezvoltată, atât clientul local, cât și clientul remote