Ceicoschi Gabriel

EJB/JPA
SDI, Grupa 244-1

Tema EJB/JPA: Aplicatie de salvat notite pentru un utilizator

PROBLEMA PROPUSA

Tema **EjbJpa** presupune realizarea unei aplicatii relevante care sa foloseasca **EJB**, **JPA**, **Servleturi** (si eventual JSP). Aplicatia va trebui să conțină un server ce să gestioneze minimum două tabele în **DB** care să aibă relaţii între ele. Mai trebuie să conţină doi clienţi, unul care să apeleze serverul prin **JNDI**, celălalt să folosească injectarea **EJB**. Aplicatia trebuie să fie instalabilă atat pe **AS WindFly** (JBoss) cat si pe **AS GlassFish**. Realizarea presupune inclusiv download-ul distributiilor corespunzătoare şi instalarea acestora. Arhiva cu trebuie uploadată prin portalul AMS, conform cerintelor de la: prezentare lucrari.

SOLUTIA IMPLEMENTATA

Solutia implementata isi propune sa:

- Adaugarea de notite text pentru un utilizator
- Afisarea tuturor notitlelor pentru acel utilozator
- Inserarea de notite folosind clienti cu injectie
- Afisarea notitelor in consola acestor clienti
- Deployment pe instante de Glassfih si Wildfly

ARHITECTURA

- Aplicatia server detine persistenta entitatilor **UserEntity** si **NoteEntity** care sunt in relatie de one-to-many(UserEntity detine o lista de NoteEntity). Logica apicatiei este pastrata in clasa **LogicBean** fiind de tip stateless bean care implementeaza interfetele **Logic** si **LogicR**(remote). Pentru functiile

implementate din Logic se folosesc clasele de entitati **UserEntity** si **NoteEntity** iar pentru functiile implementate din **LogicR** care va fi folosita la invocarea remote prin JNDI se folosesc clasele de tip Data Tansfer Object **NoteDTO** si **UserDTO**.

- Aplicatia client care este de tip servlet foloseste doua fisiere statice pentru randare: **note.jsp** pentru afisarea aplicatiei propriuzise si **error.jsp** pentru afisarea erorilor.
- Aplicatia client care se foloseste de invocarea **JNDI** este facuta pentru a fi rulata pe **Wildfly**. Aceasta contine interfata folosita in aplicatia server **LogicR** si obiectele de tip **DTO**, **NoteDTO** si **UserDTO**. Aceasta este o aplicatie simpla de consola care initializeaza parametrii necesari pentru JNDI si apoi invoca printr-un obiect proxy, obiectul de tip stateless bean din aplicatia server, iar apoi executa cateva operatii cu acesta.

INSTALARE SI DEPLOYMENT

- Pentru a face deploy cu .war sau context extern se compileaza continutul folderului /app cu **gradle clean build.**
- Pentru ca aplicatiile de server si client sa ruleze corespunzator, trebuie ca sa se defineasca DataSource in Glassfish si Wildfly. Configurarile pentru a obtine acest lucru in ambele AS sunt urmatoarele:

Glassfish v5

1. Se muta mysql-connector-java.jar (ultima versiune) in **$GLASSFISH_HOME/glassfish/lib**
2. *asadmin>create-jdbc-connection-pool --restype javax.sql.DataSource --datasourceclassname com.mysql.jdbc.jdbc2.optional.MysqlDataSource --property "url=jdbc\\:mysql\\://localhost\\:3306/mydb" mySqlPool*
3. *asadmin>create-jdbc-resource --connectionpoolid mySqlPool jdbc/mysqlDS*
4. *asadmin> deploy --createtables=true <parentDir>/ejb7cs/build/libs/ejb7cs.war*

Wildfly v11

1. Se muta mysql-connector-java.jar (ultima versiune) in

$JBOSS_HOME/bin/mysql-connector-java.jar

2. Dupa pornirea WildFly (cu **$JBOSS_HOME/bin/standalone**) si a serverului **MySQL**, se lanseaza utilitarul **$JBOSS_HOME/bin/jboss-cli.sh -c** (sau dupa caz **jboss-cli.bat)**. Prompterul lui este **[standalone@localhost:9990 /]** si la el se vor da urmatoarele trei comenzi:
3. Instalarea modulului connector, cu numele **com.mysql**, (comanda pe o singura linie): **module add --name=com.mysql --resources=mysql-connector-java.jar --dependencies=javax.api,javax.transaction.api**
4. Instalarea unui driver, numit **mysql**, (comanda pe o singura linie, la terminare apare {**"outcome" => "success"**}): **/subsystem=datasources/jdbc-driver=mysql: add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)**
5. Definirea DataSource, numita **mysqlDS**, mydb este numele bazei de date asociate(noi am folosit numele test), optional (noi nu le-am folosit) userul si parola de acces la DB (comanda pe o singura linie): **data-source add --name=mysqlDS --driver-name=mysql --jndi-name=java:jboss/datasources/mysqlDS --connection-url=jdbc:mysql://localhost:3306/mydb --user-name=gabriel --password=gabriel --enabled=true**

Instalare client cu injectie Wildfly

- Se lanseaza utiliatrul cu **$JBOSS_HOME/bin\add-user.bat**
- Se completeaza datele clientului
- Se lanseaza **$JBOSS_HOME/bin/standalone**
- Gradle clean build
- Se executa arhiva .jar cu **java -jar client7WF.jar**

BIBLIOGRAFIE

Exemplele din arhiva **6JPA2** de pe
http://www.cs.ubbcluj.ro/~florin/TPJAD/
Documentatia Wildfly: https://docs.wildfly.org/
Documentatia Glassfish:
https://javaee.github.io/glassfish/documentation