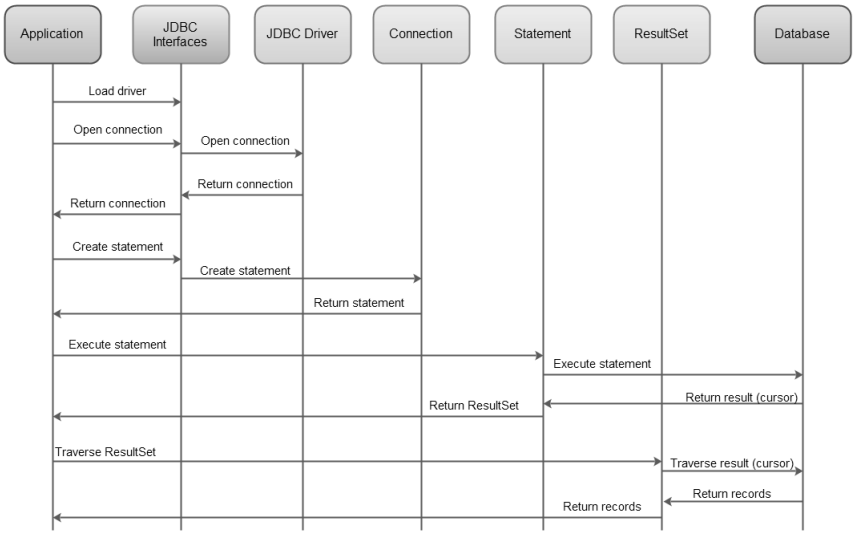
Programozás III

JPA

ISMÉTLÉS: JDBC



Forrás: http://tutorials.jenkov.com/jdbc/overview.html

ISMÉTLÉS: JDBC

JDBC: Java Database Connectivity

Gyakorlaton megoldott feladat: derby adatbázisból olvastunk.

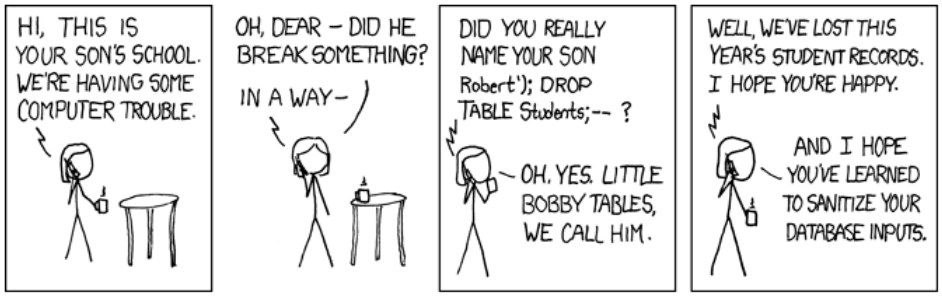
Létrehoztuk az adatbázist, kapcsolódtunk is hozzá, de ez nem elég, be kell tölteni a szükséges adatbázis-driver-t is.

Ez egy beépített Derby esetén egyszerű, de mi van, ha olyan adatbázis-drivert akarunk használni, amely nincs a felajánlottak között?

Hogy egyszerűbb legyen elérni az adatbázis driver-t, célszerű a feladatot Maven projektként megoldani.

ISMÉTLÉS: JDBC

Milyen veszélyei lehetnek az INSERT vagy UPDATE kezelésének? (A DELETE-ről nem is beszélve.)



https://xkcd.com/327/

SQL INJECTION (ÉRINTŐLEGESEN)

Ezt akarta a programozó:

INSERT INTO students (name) VALUES
('<*insertnamehere*>');

De ez lett belőle:

INSERT INTO students (name) VALUES
('Robert'); DROP TABLE students; -- ');

http://web.axelero.hu/lzsiga/ekezet.html#Q0037

Akkor lehet baj, ha a felhasználói input része lehet az SQL utasításnak.

SQL INJECTION (ÉRINTŐLEGESEN)

Ekkor a lekérdezés is veszélyes lehet, pl.:

SELECT * FROM users WHERE name = '' OR '1'='1';

Ez biztonságosabb:

sqlUtasitas = "SELECT * FROM users WHERE name = ?";

PreparedStatement utasitasObj = …

utasitasObj.setString(1, userName);

JDBC++

A JDBC az alapja a Java és valamilyen adatbázis közötti kapcsolat kialakításának és ezen keresztül SQL utasítások futtatásának.

Ugyanakkor néha nehézkes kapcsolatot teremteni egy-egy Java osztály és egy-egy relációs adattábla között.

Egy alaposabban végiggondolt megoldás olyan technológiát használ, amely szorosabb kapcsolatot teremt a Java és az adatbázis-szerkezet között.

Ez az Object Relational Mapping, melyet a JPA valósít meg.

JPA

Java Persistence API (**JPA):** Java keretrendszer relációs adatok kezelésére.

Perzisztencia: olyan adat, amely túléli az őt létrehozó folyamatot.

Perzisztálási módok pl.:
JDBC, szerializálhatóság, XML adatbázisok, stb.

Legtöbb esetben az adat nagy részét relációs adatbázisokban tároljuk. Az ehhez való hozzáférés egyik módja a JPA.

JPA

Perzisztencia entitás: olyan Java osztály, mely kapcsolódik egy relációs adatbázis táblájához, példányai az adattábla egyes sorainak felelnek meg.

A JPA definiál egy Entity Manager API-t is, amely futási időben dolgozza fel a lekérdezéseket és a tranzakciókat.

ORM

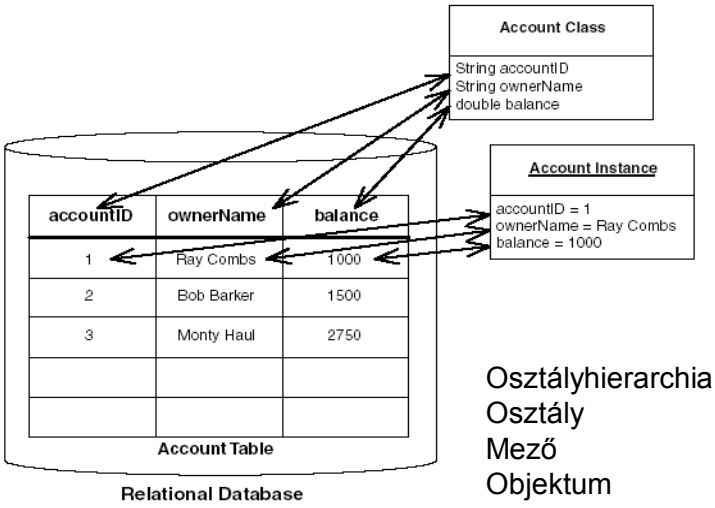
Objektum relációs leképezés (Object-Relational Mapping**):**

Cél: objektumok tárolása adatbázisban

Probléma: az objektum-orientált és a relációs adatmodell közötti különbségek

Megoldás: automatizált leképezés

ORM



Osztályhierarchia – Séma
Osztály – Tábla
Mező – Oszlop
Objektum – Sor

JPA HASZNÁLATA

A JPA egy interfész, melyet implementálni lehet – különböző provider-ek (szolgáltatók) implementálják, pl. EclipseLink, Hibernate, stb.

Entitások: Java osztályok (POJO) annotációkkal

Alapelem: Entity = perzisztens osztály

Perzisztens modul minden olyan jar, amelynek META-INF könyvtárában van persistence.xml fájl, amelyben definiáljuk pl. a providert, az adatbázis nevét, stb.

A javax.persistence csomag tartalmazza.

ENTITÁS MEGADÁSA

Java osztály `@Entity` annotációval indítva, default (azaz argumentum nélküli) konstruktorral.
Általában szerializálható (implements Serializable)

Kötelező elsődleges kulcs attribútum: `@Id`

Többféle automatikus ID-generáló stratégia adható meg a `@GeneratedValue` strategy paraméterben.

A perzisztens attribútumokat a kliensek getterek/setterek-en keresztül érhetik el.

ENTITÁS MEGADÁSA

Minden további annotáció opcionális, mert az entitás, illetve az attribútumok nevével azonos a default tábla- és oszlopnév.

Saját tábla-, ill. oszlopnév megadása:

```
@Table(name="MyTable")
@Column(name="MyColumn")
```

Az oszlop egyéb paraméterei:

`nullable`, `unique`, `length`,...

RELÁCIÓK

Számosság szerint:

```
@OneToOne
@OneToMany
@ManyToOne
@ManyToMany
```

Irány szerint kétféle:
- Egyirányú
- Kétirányú (a kapcsolat mindkét végén levő entitásnak lesz kapcsolatmenedzselő getter/setter metódusa): mappedBy paraméter.

RELÁCIÓK

Tulajdonos oldalon (Employee):

```
@ManyToOne
@JoinColumn(name="company_id")
private Company company;
```

Másik oldalon (Company):

```
@OneToMany(mappedBy="company_id")
private Collection<Employee> employees;
```

A `@JoinColumn` helyett `@JoinTable` használandó, ha külön tábla tartja nyilván a kapcsolatot (pl. ManyToMany esetén mindenképpen)
A `@ManyToOne` kötelezően tulajdonos oldal, mert nincs mappedBy paramétere, többi esetben szabadon választható a tulajdonos oldal

JPQL

A Java Persistence Query Language (JPQL) egy objektum szintű lekérdező nyelv, segítségével a relációs adatbázisban tárolt entitások kérdezhetők le.

A lekérdezések szintaxisa hasonló az SQL lekérdezésekhez, de ezekben a lekérdezésekben entitás objektumokat kezelünk, nem közvetlenül adattáblákat.

Pl: `select s from Student s`

ECLIPSELINK

A provider neve
`org.eclipse.persistence.jpa.PersistenceProvider`

Automatikus sémalétrehozás

```
<property name="eclipselink.ddl-generation"
                value="dropand-create-tables"/>
```

vagy `create-tables`

```
<property name="eclipselink.ddl-generation.
                                    output-mode"
                value="database"/>
```

ECLIPSELINK HASZNÁLATA

EntityManager lekérése

```
EntityManagerFactory factory =
Persistence.createEntityManagerFactory(PERSISTEN
CE_UNIT_NAME);
EntityManager em =factory.createEntityManager();
```

Tranzakció kezdése:

```
entityManager.getTransaction().begin();
```

Tranzakció befejezése:

```
entityManager.getTransaction().commit();
```

EntityManager lezárása:

```
entityManager.close();
```

PÉLDÁK

Előadáspéldák (neptun):

- studentJPA (többször kell futtatni)
- studentJPAFromMySQL (nem él az AB szerver)
- studentsPhonesJPA

Mindegyik Maven projekt.

A pom.xml helye: Project Files
A persistence.xml helye:
Other Sources / src / main / resources / META-INF

PÉLDÁK

További megjegyzések:

A studentsPhonesJPA induló sql scriptjének helye:
META-INF/sql
kódolása: UTF-8

Az 1. és 3. példa esetében futtatáskor létrejön az adatbázist tároló …DB mappa – ez akár ki is törölhető, akkor visszaáll az eredeti kezdőállapot.