
Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar


Eredetit készítette: Hegedűs Bettina

# Object-relational impedance mismatch

- Identity
  - Primary key attribute(s)
- Data type differences
  - Strings
- Manipulative differences
- Transactional differences
  - DBMS: transactions, OO: individual operations
- …

# ORM

ORM (Object Relational Mapping)- Objektum relációs leképzés

Az ORM egy programozási technika, ami objektum orientált rendszert  hivatott leképezni (nem kompatibilis) rendszerbe (pl: relációs adatbázis).

A cél az, hogy az objektumok tulajdonságait és kapcsolatait megőrizzük, így helyesen vissza tudjuk tölteni azokat az adatbázisba.

# Object-Relational Mapping

- Kétirányú leképezés az objektumorientált világ és a relációs modell között
  - One class – one table (a legegyszerűbb esetben)
  - One instance – one row
- Egyszerű példa:

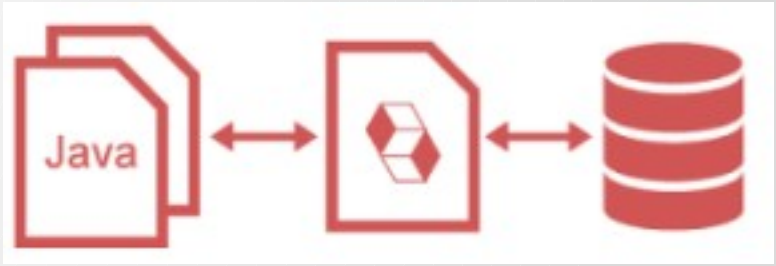
```
Statement cmd = conn.createStatement(
   "SELECT id, first_name, last_name, phone FROM
   persons WHERE id = 10";);
ResultSet res = cmd.executeQuery();
res.next();
Person p = new Person(res.getString("
   first_name"));
```

JDBC helyett Hibernate-tel:

```
Person p = session.get(Person.class, 10);
```

# Mi a Hibernate?

- Hibernate egy teljesen Java alapú framework és egy ORM , amely lehetővé teszi az objektumrelációs leképezést:
  POJO-k relációs adatbázishoz való kapcsolását.
- Nagyon hatékony leképzést hoz létre a java objektumok és az adatbázis táblák között
- Hibernate-et használva az alkalmazásfejlesztést nagyon le lehet egyszerűsíteni, fel lehet gyorsítani



# Hibernate Architekturális felépítése


# Kommunikációs egységek

- *SessionFactory*: Szálbiztos objektum Session-ök létrehozására. Az alkalmazások az ezt az interfészt implementáló osztály egyetlen példányát használják.
- *Session*: Ennek az interfésznek az implementációi biztosítják az entitásokon való műveletek végzését.A kliens és az adatbázis között egy párbeszédet reprezentál. Mikor adatbázis módosításokat akarunk végezni, a Session objektumok létrehoznak egy Transaction típusú objektumot. Ezek az objektumok reprezentálják az adatbázison elvégzett munkát.  A Session objektum nem szálbiztos, csak egy kliens használhatja egyszerre.
- *Perzisztens objektumok*: rövid életű objektum, egy Session-höz tartozhatnak csak. Ha a Session bezárul, az objektum elérhetıvé válik az alkalmazás többi rétege számára.

# Hibernate felépítése

- 3 rész:
  - Java osztály
  - Relációs adatbázisbeli táblák
  - Leíró (descriptor)
    - Definiálja a konverziós szabályokat
    - 2 fajtája van:
      - XML fájlok (valaminév.hbm.xml kiterjesztéssel)
      - Annotációk használata

# Hibernate előnyei

- Objektumokat használ táblák helyett, objektum perzisztencia
- Rövidebb fejlesztési idő
- Könnyen karbantartható
- Cache-elés (lazy vagy eager)
- Version property (optimistic locking)
- Adatbáziskezelőtől független fejlesztés
- Connection pooling

- Persze futásnál valamennyi overhead-et jelent

# (Main)

```
SessionFactory sessionFactory;
ServiceRegistry serviceRegistry;

Configuration configuration=new Configuration();
configuration.configure();
serviceRegistry = new StandardServiceRegistryBuilder().applySettings(
configuration.getProperties()).build();
sessionFactory = configuration.buildSessionFactory(serviceRegistry);

User user1=new User();
user1.setUserName(”Lajos");

Session ss= sessionFactory.openSession();
ss.beginTransaction();
ss.save(user1);
ss.save(user2);
ss.getTransaction().commit();
ss.close();
sessionFactory.close();
```

# (Konfigurációs fájl)

- **hibernate.cfg.xml**
- **Létre kell hozni a projectben új fájlként**

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
        <property name="connection.url">jdbc:oracle:thin:@oracle.itk.ppke.hu:1521:gyak</property>
        <property name="connection.username">nevem</property>
        <property name="connection.password">jelszvam</property>
        <!-- JDBC connection pool (use the built-in) -->
        <property name="connection.pool_size">1</property>
        <!-- SQL dialect -->
        <property name="dialect">org.hibernate.dialect.Oracle10gDialect</property>
        <!-- Echo all executed SQL to stdout -->
        <property name="show_sql">true</property>
        <!-- Drop and re-create the database schema on startup -->
        <property name="hbm2ddl.auto">create</property>
        <!-- Magic -->
    <property name="hibernate.temp.use_jdbc_metadata_defaults">false</property>
<mapping class="package.UserClass"></mapping>
 </session-factory>

</hibernate-configuration>
```

JDBC URL kicserélendő a JDBC-nél használtra, ami az Oracle Pluggable Database-hez való

# (XML)

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="hibernate02XML.DBUser" table="DBUSER">
        <id name="userId" type="int">
            <column name="USER_ID" precision="5" scale="0" />
            <generator class="assigned" />
        </id>
        <property name="username" type="string">
            <column name="USERNAME" length="20" not-null="true" />
        </property>
        <property name="createdBy" type="string">
            <column name="CREATED_BY" length="20" not-null="true" />
        </property>
        <property name="createdDate" type="date">
            <column name="CREATED_DATE" length="7" not-null="true" />
        </property>
    </class>
</hibernate-mapping>
```

# (Annotációk)

Az @ anotációkat a Java 5.0-ás verziójánál vezették be. Az annotációk a programkód elemeihez rendelhetők (csomagokhoz, típusokhoz, metódusokhoz, attribútumokhoz, konstruktorokhoz, lokális változókhoz), plusz információt hordoznak a Java fordító ill. speciális eszközök számára.

```
@Entity(name="User_table")
public class User {

  @Id @GeneratedValue  // primary key és generáljuk a keyt
  int userId;

  @Column(name="User_Name") // adatbázisban oszlop neve
  String userName;

  ....stb
}
```

**További:**

**https://docs.jboss.org/hibernate/stable/annotations/reference/en/html/entity.html**

# (HQL)

## Egy egyszerű példa

```
Session ss = sessionFactory.openSession();
        Transaction tx = null;
        try{
          tx = ss.beginTransaction();
          List choclate = ss.createQuery("FROM Chocolates").list();
          for (Iterator iterator1 =
                        choclate.iterator(); iterator1.hasNext();){
            Chocolates csoki = (Chocolates) iterator1.next();
            System.out.print("Neve:  " + csoki.getFlavour());
          }

          tx.commit();
        }catch (HibernateException e) {
          if (tx!=null) tx.rollback();
          e.printStackTrace();
        }finally {
          ss.close();
        }
```

# Kapcsolatok

- @OneToOne: 1-1 kapcsolat megvalósítása. Alapértelmezetten kapcsoló mezővel (join column) kerül leképezésre.
- @ManyToOne: N-1 kapcsolat megvalósítása, az N oldali entitásban. Alapértelmezetten kapcsoló mezővel (join column) kerül leképezésre.
- @OneToMany: 1-N kapcsolat megvalósítása, az 1 oldali entitásban. Az adattag típusa a kapcsolódó entitás típusából készített kollekció (List, Set). Alapértelmezetten kapcsoló táblával (join table) kerül leképezésre, de legtöbbször érdemesebb az N oldali entitás táblájába illesztett kapcsoló mezővel megvalósítani.
- @ManyToMany: N-M kapcsolat megvalósítása. Az adattag típusa a kapcsolódó entitás típusából készített kollekció (List, Set). Alapértelmezetten kapcsoló táblával (join table) kerül leképezésre, de sok esetben az N-M kapcsolatoknak saját tulajdonságai vannak, ami új köztes entitás bevezetését igénylik

```
<property name="studentName" type="string" not-null="true" length="100" column="STUDENT_NAME" />
<set name="studentPhoneNumbers" table="STUDENT_PHONE" cascade="all">
    <key column="STUDENT_ID" />
    <many-to-many column="PHONE_ID" unique="true" class="hibernate.Phone" />
</set>
```

```
@OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
@JoinTable(name = "beerFac_beer", joinColumns = {
        @JoinColumn(name = "beerFactory_id", nullable = false, updatable = false) }, inverseJoinColumns = {
                @JoinColumn(name = "beer_id", nullable = false, updatable = false) })
```

- Kapcsolat betöltésének két féle módja:
  - Mohó
  - Lusta

# (Példa)

```java
@Id @GeneratedValue
@Column(name = "USER_ID", nullable = false)
public int getUserId() {
    return userId;
}

public void setUserId(int userId) {
    this.userId = userId;
}
```

Tábla név

```java
@Entity
@Table(name = "DBUSER")
public class DBUser {

    private int userId;
    private String userName;
    private String createdBy;
    private Date createdDate;
```

Id : Elsődleges kulcs az adatbázisban

Getter-Setter : minden adattaghoz

```java
SessionFactory sessionFactory;
StandardServiceRegistry serviceRegistry;

Configuration configuration=new Configuration();
configuration.configure();
serviceRegistry = new StandardServiceRegistryBuilder().applySettings(
configuration.getProperties()).build();
sessionFactory = configuration.buildSessionFactory(serviceRegistry);

DBUser user = new DBUser();
DBUser user2 = new DBUser();
Session ss = sessionFactory.openSession();
ss.beginTransaction();
user.setUserName("testName");
user.setCreatedBy("system");
user.setCreatedDate(new java.util.Date(2010, 10, 10));
user2.setUserName("testName2");
user2.setCreatedBy("system2");
user2.setCreatedDate(new java.util.Date(2016, 10, 10));
ss.save(user);
ss.save(user2);
ss.getTransaction().commit();
ss.close();
sessionFactory.close();
```

Adattal való feltöltés

# Segítség a feladatokhoz

- **http://www.tutorialspoint.com/hibernate/index.htm**
- **http://www.mkyong.com/hibernate/hibernate-many-to-many-relationship-example-annotation/**
- https://hu.wikibooks.org/wiki/Hibernate_annot%C3%A1ci%C3%B3k_-_entit%C3%A1s_asszoci%C3%A1ci%C3%B3k/kapcsolatok

Töltsd le a Hibernate csomagot a következő linkről:
https://sourceforge.net/projects/hibernate/files/hibernate-orm/5.4.8.Final/
Készíts egy új java projectet tetszőlegesen választott néven.

- Add hozzá a hibernate csomagban található required .jar fájlokat a projeckthez
- Add hozzá a korábbi órán letöltött jdbc fájlt a projekthez

- Hozz létre egy **hibernate.cfg.xml** fájlt és állítsátok be az előadáson hozott példa szerint.
- Hozz létre egy **Categories** nevű osztályt aminek legyen egy **categoryId**  egy **name** és egy **longname** objektuma
- A **categoryId** legyen elsődleges kulcs és az adatbázis generálja nekünk ne nekünk kelljen megadni.
- A táblában legyen a longname nevű oszlop ami a kódban a description néven szerepeljen
- A táblában legyen a két kedvenc csokid!

- Hozz létre egy egy-sok kapcsolatban álló blogentry és blogentrycomment táblát között Hibernate segítségével
- A blogentry-nek legyenek a következő attribútumai:
  - Id (primary key), name és longname és a blogentrydate
- blogentrycomment -nak legyen:
  - Id-ja, commenttext és commentdate attribútuma

- Töltsd fel néhány próbabejegyzést
- Ellenőrizd sqldeveloperen keresztül hogy minden rendben működött!!
- Az egyetlen kikötés ezen kívül hogy az adatbázisban lévő elemek ne törlődjenek minden futtatás után.