Java technológiák – 11. előadás Perzisztencia

ANTAL Margit

Sapientia - EMTE

2010

Az előadás célja

- JPA – Java Persistence API
- ORM – Object Relational Mapping
- Entitások közötti asszociációk megvalósítása
- Fontosabb fogalmak
  - perzisztencia szolgáltató (Persistence Provider)
  - perzisztencia egység (Persistence Unit)
  - entitás manager (Entity Manager)
  - entitás (Entity)
  - perzisztencia kontextus (Persistence Context)

Perzisztencia

- Állandóság.
- Mechanizmus, amely során az alkalmazás adatai megőrzésre kerülnek valamely perzisztens tárolóban.
- Perzisztens tároló: pl. adatbázis.

Java Persistence API

- Egy specifikáció, amelyet minden alkalmazásszervernek kötelező implementálni.
- Használható alkalmazásszerveren kívül is.
- Osztályozás:
  - CMP – Container Managed Persistence
  - AMP – Application Managed Persistence

ORM

- Probléma: az adatok szervezésének különbözősége
  - Alkalmazás: objektumok
  - Adatbázis: relációs táblák
- Megoldás: ORM (Object Relational Mapping) software (JPA implementáció, Persistence provider)
  - Oracle Toplink,
  - Hibernate,
  - Kodo,
  - Open JPA,
  - ...

Fejlesztési lehetőségek

- top-down: Java Domain Objects $\Rightarrow$ Database Schema; annotációk vagy XML konfigurációs fájl
- bottom-up: Database Schema + Data Model $\Rightarrow$ Java Domain Objects; Reverse Engineering

Annotációk

- Annotáció = Forráskódba illesztett metaadat, amely befolyásolja, hogy különböző eszközök és osztálykönyvtárak, hogyan kezeljék azt.
- Leggyakrabban osztályhoz, attribútumhoz vagy metódushoz rendeljük.

Szintaxis

@AnnotációNeve(paraméter1=érték1, paraméter2=érték2...)

Annotációk és a Java EE

- Az annotációkat az alkalmazásszerver értelmezi és figyelembe veszi a komponensek telepítésekor.
- Az annotációk átveszik a telepítésleírók (XML: web.xml) szerepét, amennyiben vegyesen használjuk, felülírják azokat.

Annotáció – példa

Erőforrás elérése - Nem kell kódot írni

```
public class SomeClass{


@Resource(name="jdbc/distedudb")

private javax.sql.DataSource datasource;
...
}
```

Entitásosztály

Entitás

Olyan osztály, amelynek példányai perzisztensek.

```
@Entity
public class Customer implements Serializable{
  @Id
  protected Long id;

  protected String name;
  protected Address address;
  protected PreferredStatus status;

  @Transient
  protected int orderCount;

  public Customer() {}
  ...
}
```

Entitásosztály – megkötések

- az osztály publikus,
- az attribútumok nem publikusak,
- ha paraméterként át akarom adni (távoli interfész) akkor szerializálható,
- elsődleges kulcsnak megfelelő attribútumot annotálni kell: @*Id*,
- public vagy protected argumentum nélküli konstruktor.

Elsődleges kulcs

- Minden entitásnak, amit relációs adatbázisra akarunk leképezni, rendelkeznie kell elsődleges kulccsal.
- Elsődleges kulcsként a következő típusok használhatók:
  - Primitív típusok: byte, int, short, long, char
  - Primitív típusok burkoló osztályai: Byte, Integer, Short, Long, Character
  - Primitív típusok vagy azok burkoló osztályainak tömbje: byte[], int[], short[], long[], char[], Byte[], Integer[], Short[], Long[], Character[]
  - Szöveges típusok: String
  - Speciális numerikus típusok: java.math.BigInteger
  - Dátumok: java.util.Date, java.sql.Date

Alapértelmezett leképzés

- Entitásosztály ⇒ tábla
- Entitásosztály attribútuma ⇒ oszlop (mező)
- Entitásosztály példánya ⇒ sor (rekord)

Annotációkkal felülírható

```
@Table(name="CUST")
public class Client{
   @Column(name="cname")
   private String clientName;
   ...
}
```

Elsődleges kulcsgenerálás

```
@Entity
@Table( name = "CUST")
public class Client{
   @Id
   @GeneratedValue(strategy=GenerationType.AUTO)
   private int clientReference;
   //...
}
```

Alapértelmezett leképzés felülírása

```
@Entity
@Table(name="CUST")
public class Customer{
    @Transient
    private int notPersistent;
    //..
    @Column(name=cname)
    private String clientName;
    //...

}
```

Kapcsolatok entitások között

Osztályozás:

- Kapcsolat számossága:
  - egy-az-egyhez
  - sok-az-egyhez
  - egy-a-sokhoz
  - sok-a-sokhoz
- Kapcsolat irányítottsága:
  - egyirányú
  - kétirányú

Egyirányú, egy az egyhez kapcsolat

Customer → Record

```
@Entity
public class Customer{
   @Id
   private int id;
   @OneToOne
   private Record custRecord;
   //...
}

@Entity
public class Record{
   @Id
   private int recId;
   //...
}
```

Kétirányú, egy-az-egyhez kapcsolat

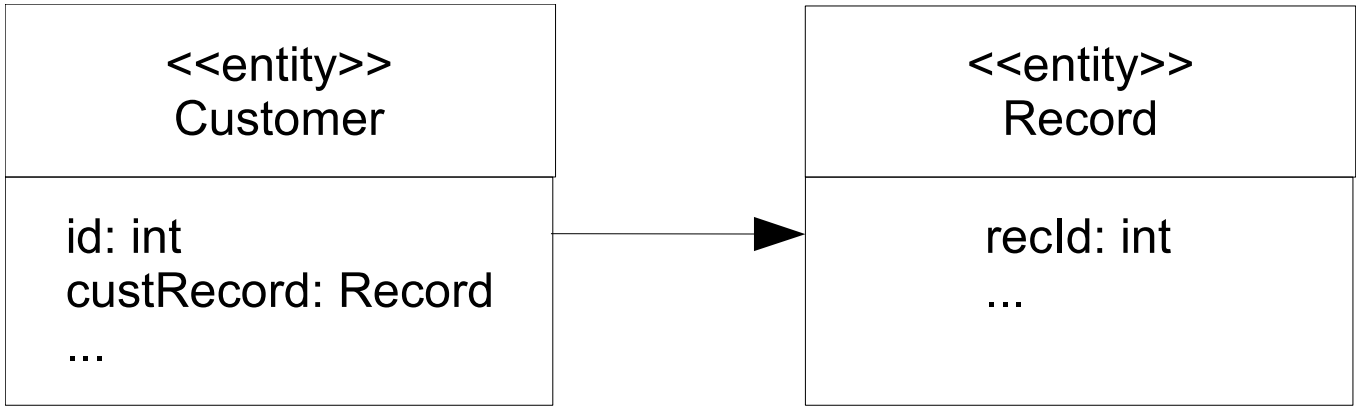
Customer ↔ Record

```
@Entity
public class Customer{
  @Id
  private int id;
  @OneToOne
  private Record custRecord;
  //...
}

@Entity
public class Record{
  @Id
  private int recId;

  @OneToOne(mappedBy="custRecord")
  private Customer customer;
  //...
}
```

Egy-az-egyhez kapcsolat (egy, illetve kétirányú)



CUSTOMER(id, ...,custrecord_recid)

RECORD(recid,...)

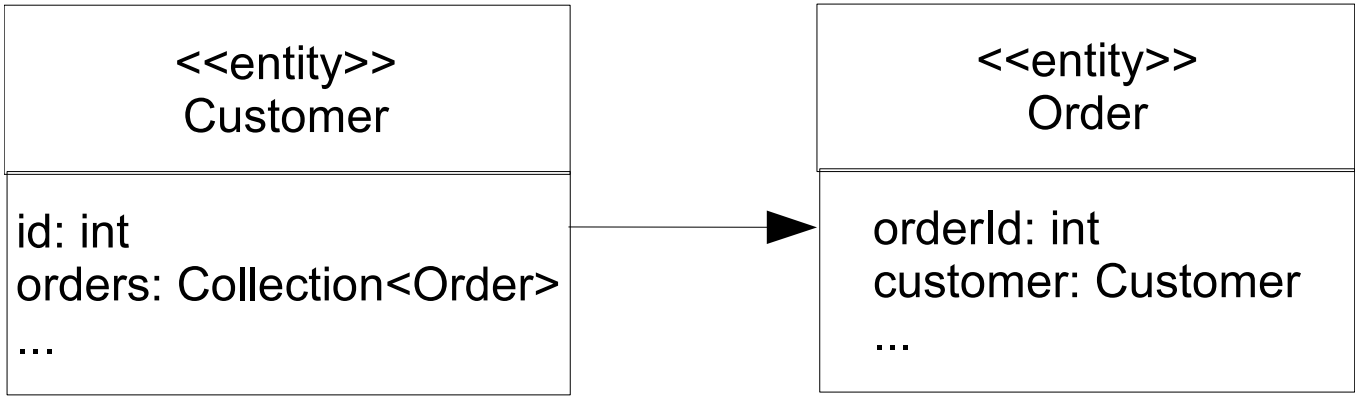
Kétirányú egy-a-sokhoz, illetve sok-az-egyhez kapcsolat

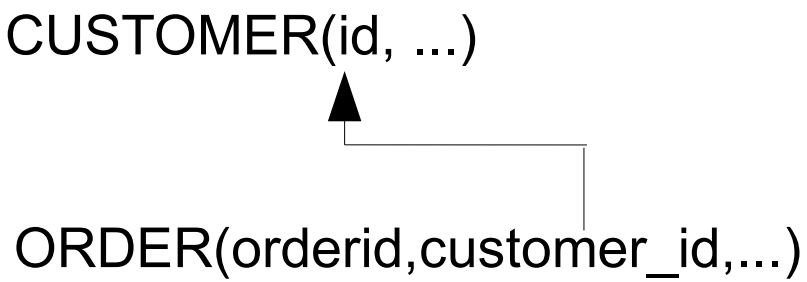
Customer ↔ Order

```
@Entity
public class Customer{
  @Id
  private int id;
  @OneToMany(mappedBy="customer")
  private Collection<Order> orders;
  //...
}

@Entity
public class Order{
  @Id
  private int orderId;
  @ManyToOne
  private Customer customer;
  //...
}
```

Egy-a-sokhoz kapcsolat





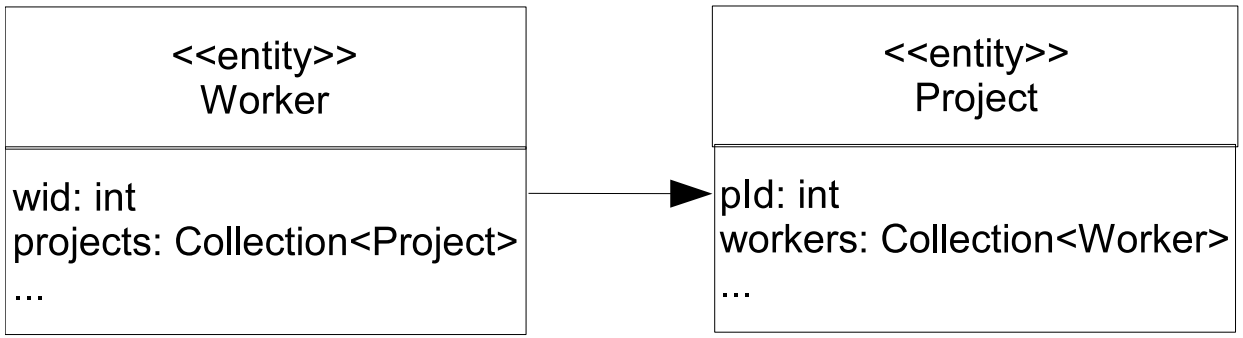
Kétirányú sok-a-sokhoz kapcsolat

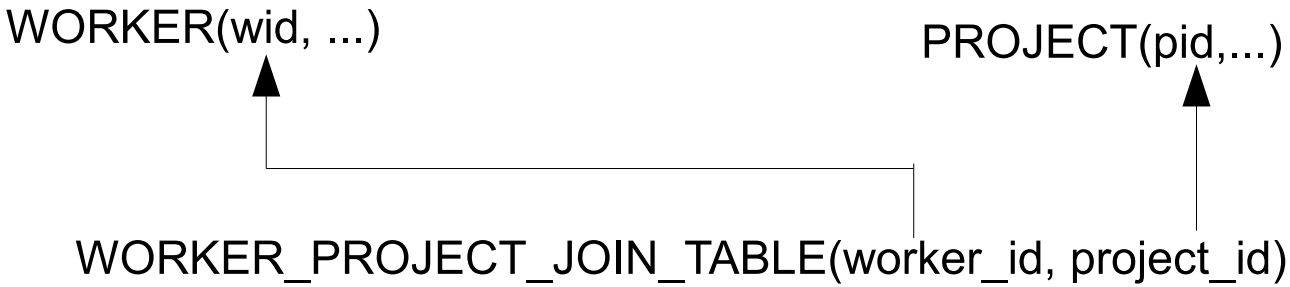
Worker ↔ Project

```
@Entity
public class Worker{
   @Id
   private int id;
   @ManyToMany
   private Collection<Project> projects;
   //...
}

@Entity
public class Project{
   @Id
   private int pId;
   @ManyToMany(mappedBy="projects")
   private Collection<Worker> workers;
   //...
}
```

Sok-a-sokhoz kapcsolat



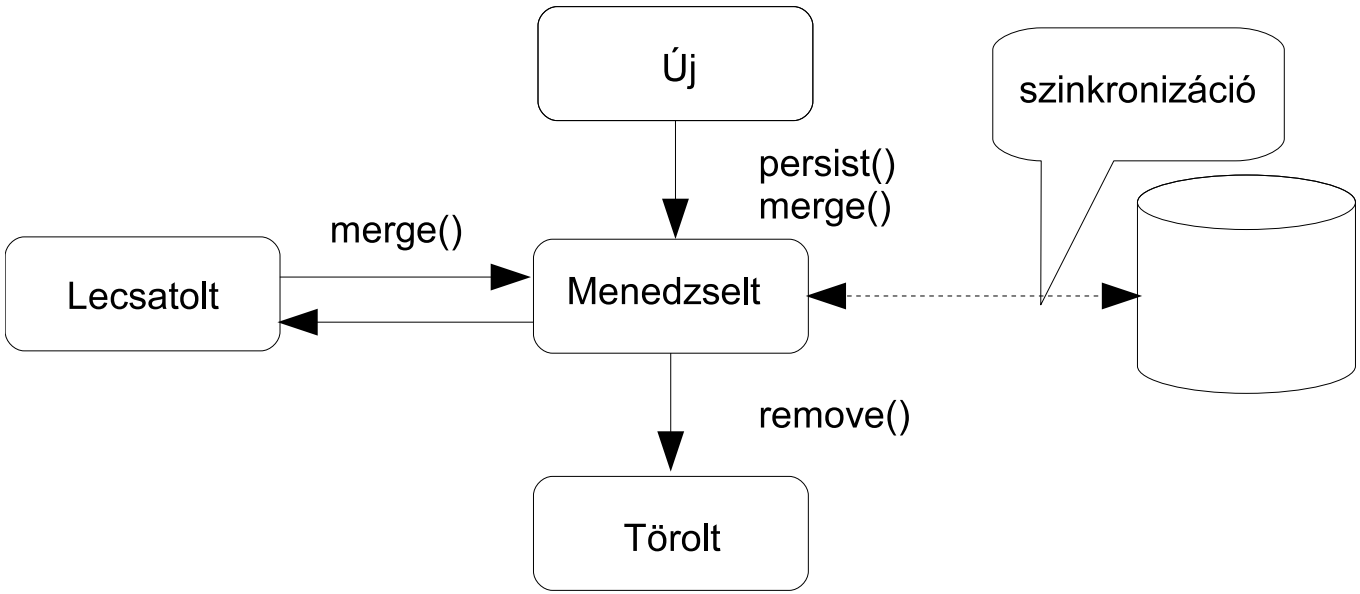


Alapfogalmak

- Entity life-cycle states – életciklus-állapotok:
  - új – new – nincs hozzárendelt rekord
  - menedzselt – managed – szinkronban van az entitás az adatbázis rekorddal
  - lecsatolt - detached – nincs szinkronban az entitás az adatbázis rekorddal
  - törölt - removed - az adattábla megfelelő rekordja ki fog törlődni
- Entity manager: az az objektum, amely kezeli az entitásobjektumokat, vezérli ezek életciklusát
- Persistence context: egyedi entitások halmaza
- Persistence identity: egy egyedi érték, amely az entitás azonosítására szolgál. Egy entitás objektum $\Rightarrow$ egy rekord

EntityManager műveletek

- persist()
- merge()
- remove()
- find()



```
SELECT [<result>]
    [FROM <candidate-class(es)>]
    [WHERE <filter>]
    [GROUP BY <grouping>]
    [HAVING <having>]
    [ORDER BY <ordering>]
```

1. példa

```
SELECT p FROM Person WHERE p.city.id IN ("SF","NY", "FS")
```

2. példa

```
Named Parameters :
Query q = em.createQuery("SELECT p FROM Person p
  WHERE p.lastName = :surname AND o.firstName = :forename");
q.setParameter("surname", theSurname);
q.setParameter("forename", theForename");

Numbered Parameters :
Query q = em.createQuery("SELECT p FROM Person p
  WHERE p.lastName = ?1 AND p.firstName = ?2");
q.setParameter(1, theSurname);
q.setParameter(2, theForename);
```