Introduction to Java EE

Java Persistence API (JPA)

Entities

Entities

- Entity is simple (POJO) Java class satisfying requirements of *JavaBeans specification*
  - Setters and getters must conform to strict form
- Every entity must have *primary key*
  - Annotation `@Id` marks the property that will be used as the entitie's primary key
- Entities may have simple business logic methods
  - These methods should operate only with entities internal fields
- Complex business logic should be implemented in dedicated *business components*
  - Introduction to business component technologies will be given later in this course

Example of an entity

```
@Entity
public class Account {
  @Id // The primary key
  @GeneratedValue(strategy=GenerationType.IDENTITY)
  private int accountNumber;

  @Column(name = "OWNER_NAME", nullable="false")
  private String ownerName; // Persistent property
  private int balance;      // Persistent too

  private transient int tmpVariable; // Not persistent!

  // Entities must have a public no-arg constructor
  public Account() {}

  // setters and getters
  ...
  // Deposit a given amount (simple business method)
  public void deposit(int amount) { balance += amount; }
}
```

JPA requirements for entity classes

- JPA requires only two pieces of metadata when you are creating a persistent class:
  - the `@javax.persistence.Entity` annotation denotes that the class should be mapped to your database,
  - the `@javax.persistence.Id` annotation marks which property in your class will be used as the primary key.
- The persistence provider will assume that all other properties in your class will map to a column of the same name and of the same type.
- The table name will default to the unqualified name of the bean class

Additional Requirements (Best Practices)

- Every *independent* entity additionally to JPA requirements should define:
  1. business key
  2. methods `equals()` and `hashCode()` that operate with business key
  3. field annotated with `@Version` – used for optimistic locking
- What is an indepenent entity?
  - Life cycle of independent entity doesn't depend upon other entities
    - "`Student`" and "`Univertity`" are independent entities
    - Entity "`Study journal`" is dependent upon entity "`Student`" – study journal cannot exist without a student

Business Key

- Primary key is used by RDBMS to guarantee referential integrity of data
  - its value must not be meaningful (for example: 32213523)
  - it shouldn't be shown in user interface
- Conversely, *business key* is a **meaningful** property (maybe composite property) of the entity that *uniquely identifies the entity among other entities in some domain*
- Entity may have multiple business keys, for example, entity "Student" might have these business keys:
  - student's number
  - student's social security number
  - student's personal code

Business Key

- What properties of an entity comprise business key decide domain experts and/or logical data model designers
- In relational databases business keys will be:
  - marked as NOT NULL
  - covered by *unique index*
- Examples:
  - Entity “University” – university’s title
  - Entity “Student” – student’s number
  - Entity “Course” – course code

Methods `equals()` and `hashCode()`

- These methods are used by Java Collections API
  - Set, HashSet, List, ArrayList, Map, etc.
- There are three choices for each entity:
  1. these methods are absent;
  2. we leave the methods generated by IDE (NetBeans / Eclipse)
     - such methods use entity's primary key (property annotated with `@Id`)
  3. we override these methods and make them use only entity's business key

Comparison of three choices

https://community.jboss.org/wiki/EqualsAndHashCode

	Without methods equals() and hashCode()	Methods use only @Id property	Methods use business key
May be used in composite key?	No	Yes	Yes
Multiple new instances in set?	Yes	No	Yes
Equal to same entity from other EntityManager?	No	Yes	Yes
Collections intact after saving to DB?	Yes	No	Yes

Requirements Summary

- Every entity must have:
  - Primary Key (@Id)
  - Business Key
    - This property will have unique index (NOT NULL, UNIQUE) in database
  - equals() and hashCode() methods that use only business key
  - Field annotated with `@Version` – used for optimistic locking

Example

```
@Entity
public class University
        implements Serializable {
  @Id
  @GeneratedValue(strategy=
      GenerationType.IDENTITY)
  private Integer id;

  private String title;

  @Version
  private int optLockVersion;

  ... get and set methods ...

  @Override
  public int hashCode() {
    int hash = 3;
    hash = 59 * hash +
      (this.title != null
         ? this.title.hashCode()
         : 0);
      return hash;
  }
```

```
@Override
public boolean equals(Object obj) {
  if (obj == null) return false;
  if (getClass() != obj.getClass())
      return false;

  final University other =
      (University) obj;

  if ((this.title == null)
    ? (other.title != null)
    : !this.title.equals(
         other.title)) {
    return false;
  }

  return true;
}
}
```