**JPA Entity Relantionships**

**Unidirectional vs Bidirectional** -> prima inseamna ca o parte a relatiei nu va stii de cealalta, iar a doua ca ambele parti stiu de ambele. Este recomandat sa folosim Bidirectionala, asta ne lasa sa navigam din orice directie

**Inheritance**

* **MappedSuperclass** – entities inherit from a super class. A database table IS NOT created for the super class (the class annotated with this will be the superclass)

**Ex**:

**@MappedSuperclass**

**public class BaseEntity….-> this class would not have a table correlated**

* **Single Table (Hibernate Default)** – One Table is used for all subclasses
* **Joined Table** – Base class and subclasses have their own tables. Fetching sublass entities require a join to the parent table
* **Table Per Class** – each subclass has it own table

**Hibernates** provides **@CreationTimestamp** and **@UpdateTimestamp** **annotations** (for audit purposes) – you can use also JPA specific: **@PrePersit** or **@PreUpdate**

**!!!Daca nu ai o conexiune la o baza de date, poti folosi H2 in memory. Iti creezi relatiile tale cu JPA, si pornesti SpringBootApplication. Mergi pe localhost:8080/h2-console/ . Iar la JDBC URL punem : jdbc:h2:mem:testdb si dupa dai Connect**

**DDL =** data definition Langugage (used to define database structures such as tables and indexes)

**DML** = data manipulation language (used with data operations such as inserts and updates)

Hibernate property is set by the Spring property **spring.jpa.hibernate.ddl-auto**

**Entity relationships types :** @OneToOne, @OneToMany, @ManyToOne, @ManyToMany

**!Many to many** : require the use of a join table

**!Fetch types used by JPA:** lazy and eager

**JPA Cascade Type =** control how changes are cascaded from parent objects to child objects

**JPA 2.1 Cascade Type = None** (JPA does not have a default Cascade Type)

**Cascade Types available in JPA 2.1 : PERSIST, MERGE, REFRESH, REMOVE, DETACH,ALL**

**!Hibernaste’s DDL-AUTO property controls =** DDL operations Hibernate will perform on startup

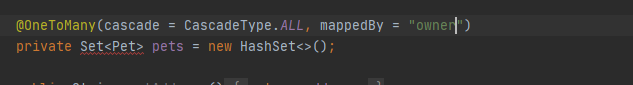
**Hibernate valid options for ddl-auto property:**  none, validate, update, create, create-drop

**!** When you are using **embedded database**, the default ddl-auto setting for Spring Boot is **CREATE-DROP**

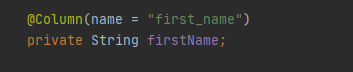
**!** When you are using **a NON** **embedded database**, the default ddl-auto setting for Spring Boot is **NONE**

**!** 2 files that Spring Boot will use to initialize the database : **schema.sql, data.sql** (these 2 will be looked on root classpath)

**CascadeType.ALL ->** this tells us that if in 2 tables we have 2 field that are correlated and you delete an input from the first table, all the others correlated fields will be deleted automatically. If an **owner** has more **pets**, and if you delete a **certain owner**, all the **pets** that belonged to that **specific owner** will be deleted.



**@Column ->** defines a column in JPA



**FetchType.EAGER ->** JPA it is trying to load everything from DB when the application runs, not when a query is performed (that would be “lazy” fetch)

