**Entity**

- annotated with @Entity

- Non final class , enums and interfaces cannot be defined as Entity

- Must have no-arg constructor - public/protected if with arg constructors are defined.

- Persistence Instance variables cannot be final

**Access Type:**

- Field Access (private,protected or default can be used)

\* all non transient fields are peristable ie not annotated with @Transient

- Property Access (public ,protected can be used)

- runtime Exceptions like PersistenceException while accessing the property methods will mark current transaction for roll-back.

Default Accesstype can be declared at the Entity level

@Access(value = AccessType.FIELD)

Property Types Allowed for Entity

- Java Primitives /wrappers , int[],byte[],Byte[],char[],Character[]

- java.lang.String

- java.math.BigDecimal, BigInteger

-java.util.Date,Calendar,

- java.sql.Date,Time,Timestamp

- user defined types that implement Serializable

- enums

- entity types , collection of entity types

- embeddable class, collection of basic and embeddable types

Entity Identity:

Primary Key

- All entities should have a primary key

- @Id

Composite Primary Key

- @EmbeddedId , @IdClass

- should implement Serializable

- should support both field access and property access

- should override equals and hashcode

- If the composite primary key class is represented as an id class, the names of primary key fields or properties in the

primary key class and those of the entity class to which the id class is mapped must correspond and their types must be the same.

Primary Derived Identity:

When an identifier(dependant entity) in one entity includes a foreign key to another entity (i.e parent entity).

there exists a one-one or many-one relationship from dependant to parent.

- a dependent entity might have multiple parent entities

- a dependent entity should have all its relationships to parents entities set before it can be persisted

-The Id attribute in the entity class and the corresponding attribute in the id class must

have the same name.

- If an Id attribute in the entity class is of basic type, the corresponding attribute in the id class must have the same type.

- Go throught the example section 2.4.1.3 Examples of Derived Identitiesin JPA2 specification

Annotations

@Entity(value="")

@Table( name="" schema="schema\_name")

or catalog="schema\_name"

@Access(value=AccessType.FIELD)

@Column

name

nullable

insertable

updateable

@Basic(fetch=FetchType.LAZY)

FetchType.EAGER

@Lob

This handles all Lobs like CLOB, BLOB

@Enumerated(value=EnumType.ORDINAL) or @Enumerated(EnumType.ORDINAL)

EnumType.ORDINAL

EnumType.STRING

@GeneratedValue(generator="" strategy="")

GenerationType.AUTO , IDENTITY,SEQUENCE

GenerationType.TABLE

@TableGenerator(name="" , table="", pkColumnName="' , valueColumnName="" , initialValue= , allocationSize= , schema="", catalog="")

can be defined in any class file and its available for the entire persistence unit.

Eventhough its is available for entire persistence unit , it still needs to defined for each class.

@SequenceGenerator(name="" , sequenceName="" , initialValue= , schema="", catalog="")

Table RelationShip Annotations

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@ManyToOne

@JoinColumn(name="" , referencedColumnName ="")

name --> should refer to actual fk column name in the table

will denote the foreign key column in the relationship owner table.

@OneToOne ()

mappedBy=""placed on the inverse side of the relationship.

cascade={CascadeType.ALL, PERSIST,MERGE,DETACH,REFRESH,REMOVE,}

targetEntity=Sample.class

fetch=FetchType

@OneToMany

@ManyToMany

@JoinTable(name="',

joinColumns=@JoinColumn(name=""),inverseJoinColumns=@JoinColumn(name="")

)

@ElementCollection (targetClass=Sample.class)

JPA 2.0 defines an ElementCollection mapping. It is meant to handle several non-standard relationship mappings. An ElementCollection can be used to define a one-to-many relationship to an Embeddable object, or a Basic value (such as a collection of Strings). An ElementCollection can also be used in combination with a Map to define relationships where the key can be any type of object, and the value is an Embeddable object or a Basic value.

- @CollectionTable(name="", @JoinColumns=)

More Reference : http://en.wikibooks.org/wiki/Java\_Persistence/ElementCollection#ElementCollection

@OrderBy(" attributeName ASC") -

ASC is default

DESC for decending order.

changing the order in memory will not make changes in Db

the ordering is performed at the oracle query using "order by"

@OrderColumn(name="")

default value would be "attributename"\_ORDER .

Can be annotated in owner or the inverse side.

Details on Inheritance Type - TODO