1. **Access Types**:

You can use both the types, either with field or with property types. But don't use both as the same property will be validated twice. For a property level, you have to keep the annotation on getters. Also, if you keep for the id, then the remaining all will be understood as property..and shimilar to the field level also.

2. **Mapping to a Table:**

By default, if you give a pojo with @entity, then it will search for the same classname in the database for the table name. Otherwise, you have to give the annotation as @Table after @Entity. Example: @Table(name="EMPLOYEE")

If you want to specify the schema as well, you give it as @Table(name="EMP", schema="HR")

**3. @Basic mappings:**

An optional @Basic annotation can be placed on a field or property to explicitly mark it as being

persistent. This annotation is mostly for documentation purposes and is not required for the field or

property to be persistent. Because of the annotation, we call mappings of simple types basic mappings.

@Column:

This is used, when the property name in the pojo does not match with that of the database. Here the mandatory parameter what we give is the name.

**@Basic mappings** annotation is also used in order to specify the fetch type. The fetch types will be of the form:

@Basic(fetch=FetchType.LAZY) (OR)

@Basic(fetch=FetchType.EAGER)

The directive to lazily fetch an attribute is meant only to be a hint to the persistence provider to help the application achieve better performance. The provider is not required to respect the request because the behavior of the entity is not compromised if the provider goes ahead and loads the attribute.

I see that jpa, hibernate annotations dosen't apply for normal fields. **Maybe they will be applicable to collections only..! Need to check further.**

4. BLOB and CLOB:

LOBs come in two flavors in the database: character large objects, called CLOBs, and binary large

objects, or BLOBs. As their names imply, a CLOB column holds a large character sequence, and a BLOB

column can store a large byte sequence. The Java types mapped to BLOB columns are byte[], Byte[],

and Serializable types, while char[], Character[], and String objects are mapped to CLOB columns.

The provider is responsible for making this distinction based on the type of the attribute being

mapped.

Example of BLOB column mapping:

@Entity

public class Employee {

@Id

private int id;

@Basic(fetch=FetchType.LAZY)

@Lob @Column(name="PIC")

private byte[] picture;

// ...

}