JPA INTERVIEW QUESTIONS

What is Java Persistence API?

The mapping between relational database tables and java objects are called object relational mapping (ORM). Basically JPA helps to manage ORM in java based applicaions. It is very important that JPA is just a specification and some implementations are Hibernate or EclipseLink.

What is the difference between persistence.xml and hibernate.cfg.xml?

When we are dealing with JPA, we use persistence.xml (by the way all the configurations can be done in java code). Hibernate API needs persistence.cfg.xml. It is a good programming practise to use JPA configurations because we may change the vendors at runtime (for example from Hibernate to EclipseLink or vice versa)

What is an EntityManager?

It provides basically all the database related operations such as persist objects, find objects or remove objects. EntityManager is created by EntityManagerFactory. All the entities are managed by EntityManager. These are the so called persistent objects. If we close EntityManager, all the entities will be in a detached state: these are the detached objects. If we want to synchronize them with the database again, we have to merge() them. Thus, they will become persistent objects again.

What is an entity?

Basically every class that is annotated by @Entity annotation. These classes can be managed by the EntityManager. An instance of these annotated classes will be rows in the relational database tables. In the JPQL queries we can reference the given entity as we have specified in the @Entity(name="entityName") name parameter.

Why prefer JPA to Hibernate or any other vendors?

Basically JPA is the specification or the standard. Using JPA does not ties us to any of the vendors such as Hibernate or EclipseLink. We can decide what to use according to what problems we want to solve. Sometimes Hibernate is better, sometimes EclipseLink is the better choice. IMPORTANT: sometimes we have to use the vendors directly because there are some features that JPA does not support!

What is JPQL?

It stands for Java Persistent Query Language. It is like SQL but instead of manipulating relational database tables, JPQL manipulates java objects (classes annotated by the @Entity annotation to be precise). JPQL is interpreted at runtime, so the compiler does not warn us if the syntax is incorrect at compile time. Thats why Criteria API is came to be, it is a bit more convenient in this sense.

What are the most important JPA annotations?

@Entity we define the entity like this

@Table(name="TABLE_NAME") JPA will create a database table with name TABLE_NAME

// @Entity also creates a database table, so @Table is optional

@Transient	any field within an entity will not be persisted so no database table column will be created
@Version	for concurrent modification and optimistic locking
@Id	we can define the primary key
@GeneratedValue	we can specify that the database is going to generate the value for the given field