JPA Interview Questions

A list of top frequently asked JPA interview questions and answers are given below:

1) What is the Java Persistence API?

The Java Persistence API (JPA) is the specification of Java that is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems. As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation. Therefore, ORM tools like Hibernate, TopLink, and iBatis implements JPA specifications for data persistence. The first version of the Java Persistence API, JPA 1.0 was released in 2006 as a part of EJB 3.0 specification.

---

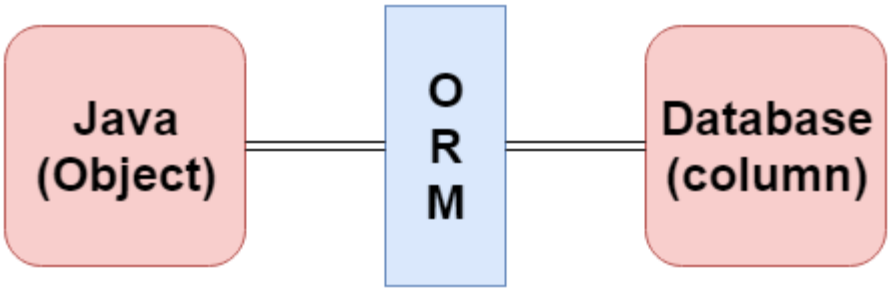
2) Does JPA performs the actual task like access, persist and manage data?

No, JPA is only a specification. The ORM tools like Hibernate, iBatis, and TopLink implements the JPA specification and perform these type of tasks.

---

3) What is the object-relational mapping?

The object-relational mapping is a mechanism which is used to develop and maintain a relationship between an object and the relational database by mapping an object state into the database column. It converts attributes of programming code into columns of the table. It is capable of handling various database operations easily such as insertion, updation, deletion, etc.



---

4) What are the advantages of JPA?

The advantages of JPA are given below.

- The burden of interacting with the database reduces significantly by using JPA.

- The user programming becomes easy by concealing the O/R mapping and database access processing.
- The cost of creating the definition file is reduced by using annotations.
- We can merge the applications used with other JPA providers
- Using different implementations can add the features to the standard Implementation which can later be the part of JPA specification.

---

5) What are the embeddable classes?

Embeddable classes represent the state of an entity but do not have a persistent identity of their own. The objects of such classes share the identity of the entity classes that owns it. An Entity may have single-valued or multivalued embeddable class attributes.

---

6) List some ORM frameworks.

Following are the various frameworks that function on ORM mechanism: -

- Hibernate
- TopLink
- ORMLite
- iBATIS
- JPOX

---

7) What is the JPQL?

JPQL is the Java Persistence query language defined in JPA specification. It is used to construct the queries.

8) What are the steps to persist an entity object?

The following steps are performed to persist an entity object.

- Create an entity manager factory object.
  The **EntityManagerFactory** interface present in **java.persistence** package is used to provide an entity manager.
  1. EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student_details");

- Obtain an entity manager from the factory.

```
1. EntityManager em=emf.createEntityManager();
```

- Initialize an entity manager.

```
1. em.getTransaction().begin();
```

- Persist the data into the relational database.

```
1. em.persist(s1);
```

- Closing the transaction

```
1. em.getTransaction().commit();
```

- Release the factory resources.

```
1. emf.close();
2.     em.close();
```

---

9) What are the steps to insert an entity?

We can easily insert the data into the database through the entity. The EntityManager provides persist() method to add records. The following steps are used to insert the record into the database.

- Create an entity class, for example, **Student.java** with the attribute student_name.

```
1. package com.javatpoint.jpa.student;
2. import javax.persistence.*;
3.
4. @Entity
5. @Table(name="student")
6. public class Student {
7.
8.     @Id
9.     private String s_name;
10.
11.    public StudentEntity(String s_name) {
12.        super();
```

```
        this.s_name = s_name;
    }

    public StudentEntity() {
        super();
    }

    public String getS_name() {
        return s_name;
    }

    public void setS_name(String s_name) {
        this.s_name = s_name;
    }
}
```

- Now, map the entity class and other databases configuration in **Persistence.xml** file.

```
<persistence>
<persistence-unit name="Student_details">

    <class>com.javatpoint.jpa.student.StudentEntity</class>

<properties>
<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/studentdata"/>
<property name="javax.persistence.jdbc.user" value="root"/>
<property name="javax.persistence.jdbc.password" value=""/>
<property name="eclipselink.logging.level" value="SEVERE"/>
<property name="eclipselink.ddl-generation" value="create-or-extend-tables"/>
</properties>

    </persistence-unit>
</persistence>
```

- Create a persistence class named as PersistStudent.java under com.javatpoint.jpa.persist package to persist the entity object with data

```
1. package com.javatpoint.jpa.persist;
2.
3. import com.javatpoint.jpa.student.*;
4. import javax.persistence.*;
5. public class PersistStudent {
6.
7.     public static void main(String args[])
8.     {
9.
10.         EntityManagerFactory emf=Persistence.createEntityManagerFacto
    ry("Student_details");
11.         EntityManager em=emf.createEntityManager();
12.
13.em.getTransaction().begin();
14.
15.         StudentEntity s1=new StudentEntity();
16.         s1.setS_name("Gaurav");
17.         em.persist(s1);
18.         em.getTransaction().commit();
19.         emf.close();
20.         em.close();
21.     }
22.}
```

---

10) What are the steps to find an entity?

To find an entity, EntityManger interface provides find() method that searches an element by the primary key. The following steps are used to find an entity in the record.

- Create an entity class named as **StudentEntity.java** under com.javatpoint.jpa.student package that contains attributes s_name.

```
1. package com.javatpoint.jpa.student;
2. import javax.persistence.*;
3.
4. @Entity
```

```
@Table(name="student")
public class StudentEntity {

    @Id
    private String s_name;
    private int s_id;
    public StudentEntity(String s_name, int s_id) {
        super();
        this.s_name = s_name;
        this.s_id = s_id;
    }

    public StudentEntity() {
        super();
    }


    public String getS_id() {
        return s_id;
    }

    public void setS_id(int s_id) {
        this.s_name = s_id;
    }
    public String getS_name() {
        return s_name;
    }

    public void setS_name(String s_name) {
        this.s_name = s_name;
    }
}
```

- Now, map the entity class and other databases configuration in **Persistence.xml** file.

```
<persistence>
<persistence-unit name="Student_details">
```

```
3.
4.      <class>com.javatpoint.jpa.student.StudentEntity</class>
5.
6.  <properties>
7.  <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
8.  <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/studentdata"/>
9.  <property name="javax.persistence.jdbc.user" value="root"/>
10. <property name="javax.persistence.jdbc.password" value=""/>
11. <property name="eclipselink.logging.level" value="SEVERE"/>
12. <property name="eclipselink.ddl-generation" value="create-or-extend-tables"/>
13. </properties>
14.
15.     </persistence-unit>
16. </persistence>
```

- Create a persistence class named as **FindStudent.java** under com.javatpoint.jpa. Find the package to persist the entity object with data.

```
1.  package com.javatpoint.jpa.find;
2.  import javax.persistence.*;
3.
4.  import com.javatpoint.jpa.student.*;
5.
6.  public class FindStudent {
7.      public static void main(String args[])
8.      {
9.          EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student_details");
10.         EntityManager em=emf.createEntityManager();
11.
12.
13.
14.         StudentEntity s=em.find(StudentEntity.class,101);
15.
16.         System.out.println("Student id = "+s.getS_id());
```

```
17.         System.out.println("Student Name = "+s.getS_name());
18.         System.out.println("Student Age = "+s.getS_age());
19.
20.     }
21.}
```

---

11) What are the steps to update an entity?

JPA allows us to change the records in the database by updating an entity. The following steps are to be performed to update the entity.

- Create an entity class named as StudentEntity.java under com.javatpoint.jpa.student package, that contains attribute s_id and s_name.

  StudentEntity.java

```
1.  package com.javatpoint.jpa.student;
2.  import javax.persistence.*;
3.
4.  @Entity
5.  @Table(name="student")
6.  public class StudentEntity {
7.
8.      @Id
9.      private String s_name;
10.     private int s_id;
11.     public StudentEntity(String s_name, int s_id) {
12.         super();
13.         this.s_name = s_name;
14.         this.s_id = s_id;
15.     }
16.
17.     public StudentEntity() {
18.         super();
19.     }
20.
21.
```

```
22.    public String getS_id() {
23.        return s_id;
24.    }
25.
26.    public void setS_id(int s_id) {
27.        this.s_name = s_id;
28.    }
29.    public String getS_name() {
30.        return s_name;
31.    }
32.
33.    public void setS_name(String s_name) {
34.        this.s_name = s_name;
35.    }
36.}
```

- Now, map the entity class and other databases configuration in Persistence.xml file.

  Persistence.xml

```
1.  <persistence>
2.  <persistence-unit name="Student_details">
3.
4.      <class>com.javatpoint.jpa.student.StudentEntity</class>
5.
6.  <properties>
7.  <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
8.  <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/studentdata"/>
9.  <property name="javax.persistence.jdbc.user" value="root"/>
10. <property name="javax.persistence.jdbc.password" value=""/>
11. <property name="eclipselink.logging.level" value="SEVERE"/>
12. <property name="eclipselink.ddl-generation" value="create-or-extend-tables"/>
13. </properties>
```

```
14.
15.     </persistence-unit>
16. </persistence>
```

- Create a persistence class named as UpdateStudent.java under com.javatpoint.jpa.update package to persist the entity object with data.

  UpdateStudent.java

```
1.  package com.javatpoint.jpa.update;
2.  import javax.persistence.*;
3.
4.  import com.javatpoint.jpa.student.*;
5.  public class UpdateStudent {
6.
7.      public static void main(String args[])
8.      {
9.          EntityManagerFactory emf=Persistence.createEntityManagerFacto
    ry("Student_details");
10.         EntityManager em=emf.createEntityManager();
11.         StudentEntity s=em.find(StudentEntity.class,102);
12.         System.out.println("Before Updation");
13.         System.out.println("Student Name = "+s.getS_name());
14.         s.setName("Ayush");
15.         System.out.println("After Updation");
16.         System.out.println("Student Name = "+s.getS_name());
17.     }
18. }
```

---

12) What are the steps to delete an entity?

To delete a record from the database, EntityManager interface provides remove() method. The remove() method uses the primary key to delete the particular record. The following examples are to be performed to delete an entity.

- Create an entity class named as **StudentEntity.java** under com.javatpoint.jpa.student package that contains attribute s_id and s_name.

```
1.  package com.javatpoint.jpa.student;
2.  import javax.persistence.*;
3.
```

```
@Entity
@Table(name="student")
public class StudentEntity {

    @Id
    private int s_id;
    private String s_name;

    public StudentEntity(int s_id, String s_name) {
        super();
        this.s_id = s_id;
        this.s_name = s_name;
    }

    public StudentEntity() {
        super();
    }

    public int getS_id() {
        return s_id;
    }

    public void setS_id(int s_id) {
        this.s_id = s_id;
    }

    public String getS_name() {
        return s_name;
    }

    public void setS_name(String s_name) {
        this.s_name = s_name;
    }
}
```

- Now, map the entity class and other databases configuration in Persistence.xml file.

Persistence.xml

```
<persistence>
<persistence-unit name="Student_details">

    <class>com.javatpoint.jpa.student.StudentEntity</class>

<properties>
<property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
<property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/studentdata"/>
<property name="javax.persistence.jdbc.user" value="root"/>
<property name="javax.persistence.jdbc.password" value=""/>
<property name="eclipselink.logging.level" value="SEVERE"/>
<property name="eclipselink.ddl-generation" value="create-or-extend-tables"/>
</properties>

    </persistence-unit>
</persistence>
```

Deletion.java

```
package com.javatpoint.jpa.delete;
import javax.persistence.*;
import com.javatpoint.jpa.student.*;

public class DeleteStudent {

    public static void main(String args[])
    {
    EntityManagerFactory emf=Persistence.createEntityManagerFactory("Student_details");
    EntityManager em=emf.createEntityManager();
em.getTransaction().begin();

    StudentEntity s=em.find(StudentEntity.class,102);
```

```
30.em.remove(s);
31.em.getTransaction().commit();
32.emf.close();
33.em.close();
34.
35.    }
36.}
```

---

13) Insert a record mechanism using JPA?

```
1.
2.  @Override
3.  @Transactional
4.  public void create(Category entity) throws MeetingAppDAOException {
5.  try {
6.  logger.info("Enter - create()");
7.  super.create(entity);
8.  logger.info("Exit - create()");
9.  } catch (PersistenceException exception) {
10.logger.error("create()::REASON OF EXCEPTION=" + exception.getMessage(),
    e);
11.}
12.}
13.
```

---

14) What are the different directions of entity mapping?

The direction of a mapping can be either unidirectional or bidirectional. In unidirectional mapping, only one entity can be mapped to another entity, whereas in bidirectional mapping each entity can be mapped or referred to another entity.

---

15) What are the different types of entity mapping?

Following are the types of object-relational mapping: -

- **One-to-one mapping:** The one-to-one mapping represents a single-valued association where an instance of one entity is associated with an instance of another entity. In this type of association, one instance of source entity can be mapped with at most one instance of the target entity.

- **One-To-Many mapping:** The One-To-Many mapping comes into the category of collection-valued association where an entity is associated with a collection of other entities. In this type of association, the instance of one entity can be mapped with any number of instances of another entity.
- **Many-to-one mapping** The Many-To-One mapping represents a single-valued association where a collection of entities can be associated with the similar entity. In the relational database, more than one row of an entity can refer to the same row of another entity.
- **Many-to-many mapping** The Many-To-Many mapping represents a collection-valued association where any number of entities can be associated with a collection of other entities. In the relational database, more than one row of one entity can refer to more than one row of another entity.

---

16) What is an orphan removal in mappings?

If a target entity in one-to-one or one-to-many mapping is removed from the mapping, then remove operation can be cascaded to the target entity. Such target entities are known as orphans, and the orphanRemoval attribute can be used to specify that orphaned entities should be removed.

---

17) Explain persistence life cycle of an object?

In persistence life cycle, the object lies in the following states: -

- Transient - The object is called to be in the transient state when it is just declared by using the new keyword. When an object remains in the transient state, it doesn't contain any identifier(primary key) in the database.
- Persistence - In this state, an object is associated with the session and either saved to a database or retrieved from the database. When an object remains in the persistence state, It contains a row of the database and consists of an identifier value. We can make an object persistent by associating it with the hibernate session.
- Detached - The object enters into a detached state when the hibernate session is closed. The changes made to the detached objects are not saved to the database.

---

18) What are the different types of identifier generation?

Following are the types of id generation strategy required to specify with @GeneratedValue annotation: -

- Automatic Id generation - In this case, the application doesn't care about the kind of id generation and hand over this task to the provider. If any value is not specified explicitly, the generation type defaults to auto.
- Id generation using a table - The identifiers can also be generated using a database table.
- Id generation using a database sequence - Databases support an internal mechanism for id generation called sequences. To customize the database sequence name, we can use the JPA @SequenceGenerator annotation.
- Id generation using a database identity - In this approach, whenever a row is inserted into the table, a unique identifier is assigned to the identity column that can be used to generate the identifiers for the objects.

---

19) What is an entity?

The entity is a group of states associated together in a single unit. An entity behaves as an object and becomes a major constituent of the object-oriented paradigm. In other words, we can say that an entity is an application-defined object in the Java Persistence Library. Each entity is associated with the metadata which represents its information in the form of XML or annotation.

---

20) What are the properties of an entity?

Following are the properties of an entity that an object must have: -

- **Persistability:** An object is called persistent if it is stored in the database and can be accessed anytime.
- **Persistent Identity:** In Java, each entity is unique and represents an object identity. Similarly, when the object identity is stored in a database, then it is represented as persistence identity. This object identity is equivalent to the primary key in the database.
- **Transactionality:** In Java, each entity is unique and represents an object identity. Similarly, when the object identity is stored in a database, then it is

represented as persistence identity. This object identity is equivalent to the primary key in the database.

- **Granularity:** Entities should not be primitives, primitive wrappers or built-in objects with single dimensional state.

---

21) What is the role of Entity Manager in JPA?

An entity manager is responsible for the following operations.

- The entity manager implements the API and encapsulates all of them within a single interface.
- The entity manager is used to read, delete and write an entity.
- An object referenced by an entity is managed by entity manager.

---

22) What are the constraints on an entity class?

An entity class must fulfill the following requirements:

- The class must have a no-argument constructor.
- The class can't be final.
- The class must be annotated with @Entity annotation.
- The class must implement a Serializable interface if value passes an empty instance as a detached object.

---

23) What is the purpose of Java collections in JPA?

In JPA, Java collections are used to persist the object of wrapper classes and String.

---

24) What type of objects can be stored in the JPA collections mapping?

Following are the type of objects that JPA allows to store: -

- Basic Types
- Entities
- Embeddable

25) What type of collections can be used in JPA?

To store multivalued entity associations and a collection of objects, following types of Java collections is used: -

- List
- Set
- Map

26) What is the purpose of cascading operations in JPA?

If we apply any task to one entity then using cascading operations, we make it applicable to its related entities also.

27) What are the types of cascade supported by JPA?

Following is the list of cascade type: -

- **PERSIST:** In this cascade operation, if the parent entity is persisted then all its related entity will also be persisted.
- **MERGE:** In this cascade operation, if the parent entity is merged, then all its related entity will also be merged.
- **DETACH:** In this cascade operation, if the parent entity is detached, then all its related entity will also be detached.
- **REFRESH:** In this cascade operation, if the parent entity is refreshed, then all its related entity will also be refreshed.
- **REMOVE:** In this cascade operation, if the parent entity is removed, then all its related entity will also be removed.
- **ALL** In this case, all the above cascade operations can be applied to the entities related to the parent entity.

28) What is JPQL?

The Java Persistence Query language (JPQL) is a part of JPA specification that defines searches against persistence entities. It is an object-oriented query language which is used to perform database operations on persistent entities. Instead of the database table, JPQL uses entity object model to operate the SQL queries. Here, the

role of JPA is to transform JPQL into SQL. Thus, it provides an easy platform for developers to handle SQL tasks. JPQL is an extension of Entity JavaBeans Query Language (EJBQL).

---

29) What are the features of JPQL?

Some of the essential features of JPQL are: -

- It is simple and robust.
- It is a platform-independent query language.
- JPQL queries can be declared statically into metadata or can also be dynamically built in code.
- It can be used with any database such as MySQL, Oracle.

---

30) What is the Criteria API?

The Criteria API is a specification that provides type-safe and portable criteria queries written using Java programming language APIs. It is one of the most common ways of constructing queries for entities and their persistent state. It is just an alternative method for defining JPA queries. Criteria API defines a platform-independent criteria queries, written in Java programming language. It was introduced in JPA 2.0. The main purpose behind this is to provide a type-safe way to express a query.