**Hibernate**

* Persistent layer: to achieve 100% data persistency (representing data permanently in my databases in backend systems).
* CRUD operations are called persistent operations.
* To achieve data persistency, we use
  + Serialization and deserialization
  + JDBC
  + ORM tools
    - EJBs- Entity beans
    - Hibernate
    - JPA
    - iBatis
* Serialization and Deserialization:
  + The process of separating data from object is called serialization.
  + The process of regenerate an object based on data is called deserialization.
  + objectOutputStream - serialization
  + objectInputstream – Deserialization
  + not suited for enterprise level persistent mechanism.
  + Not much compatible with char stream
  + Can perform limited operations.
  + No query language support.
  + So, we go for next level persistent mechanism (JDBC).
* JDBC:
  + To interact with database from java applications.
  + Connection is having Auto-Commit nature. (so, called persistent technology)
  + Why hibernate?
    - Developer need to check driver management, connection management, statement management.
    - To get solution without having database knowledge.
    - We need to write boiler plate code in JDBC.
    - So, we go for ORM
* **ORM (Object relational mapping)**:
  + 3 data copies of same data in JDBC. (three data models)
  + Developer is responsible for converting data from object-oriented data model to relational data model. (may be chances for mismatches).
  + To resolve mismatches between object-oriented data model and relational data model we need some mapping which is provided ORM tools.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of text

Description generated with very high confidence

Mismatches between Object Oriented Data Model and Relational Data Model:

* Granularity Mismatch:
  + Count mismatch between ODM and RDM.
  + ORM provides solution to this by using mapping between ODM and RDM with Component mapping (one component can represent multiple columns of database).
  + ORM stores data in respective columns mentioned in the mapping.

A close up of a piece of paper

Description generated with high confidence

* Subtypes Mismatches:
  + During Inherited ORD how tables should be created in database side or how it need store.
  + To determine correct mapping to RDM we need ORM.
  + In hibernate we have 3 hierarchies
    - Table per class hierarchy
    - Table per concreate class
    - Table per class

A close up of text on a white surface

Description generated with high confidence

* Association Mismatches:
  + To improve communication between entity classes and to improve data navigation we go for associations.
  + At database side we need decide the mechanism for many to many association.
  + How data is stored in database side?
  + How to do mapping between tables?
  + This kind is very difficult to implement using JDBC.
  + ORM provides mapping for this
  + Hibernate gives Associations mapping
    - 1-1
    - 1-M
    - M-1
    - M-M
* Identities Mismatches:
  + Can check duplicates objects and manages to a single record using ORM tool.
* Cost Mismatch
* ---

**ORM is between Java Application and database.**

**ORM:**

* Is a set of rules and regulations to provide mapping between object-oriented programming elements (classes, id properties) and relations data model elements (tables, columns) either by using XML files or by using Annotations?
* This is implemented by
  + EJBs- Entity Beans
  + Hibernate
  + JPA
  + iBatis and many

**A screenshot of a cell phone

Description generated with very high confidence**

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with high confidence

A screenshot of a cell phone

Description generated with very high confidence

**HIBERNATE:**

Product by: Red hat System

Author: Gavin King

Type: ORM Implemented Product

Objective: To simplify Persistency layer in Enterprise Application.

Latest Version: Hibernate5.X[2017]

**Features**:

1. Hibernate is database Independent. Based on the database type hibernate will create queries and execute them and return results.
2. It can be used for both Standalone applications and Enterprise Applications
3. It will run with or without application servers.
4. It has more portable persistency mechanism.
5. Light weight persistency mechanism.
6. It is having very good collection framework support to perform database operations (JDBC result set doesn’t implement serializable, so it is difficult for developer to transfer in to collection).
7. It is having HQL [Hibernate query language] (database Independent, Object oriented and powerful query language)
8. It is having very good Annotations support to reduce XML dependency.
9. It is having very good implementation for all the primary key generation algorithms to generate primary key values internally.
10. Good transaction support.
11. Very good cache mechanisms to improve object reusability and performance.
12. It is having tools support to create and alter tables at databases automatically.
13. It is having its own connection pooling mechanism to reuse the connection objects.
14. It has very good implementations for Associations and Joins.

**Architecture:**

* Hibernate software will interact with database after interacting with configuration file.
* Hibernate mapping file provides the details mapping between POJO class and DB tables to software in order to prepare DB native SQL queries to execute.
* Client Application perform the following operations:
  + Activating Hibernate Software:
    - Hibernate software is loaded and then it checks the configuration file loads and parses it. Then it creates configuration object and provides an environment to interact with the database. Also, it loads mapping files and parses and loads it internally as objects.

A screenshot of a cell phone

Description generated with very high confidence

* + Creating Bean Objects:
  + Perform Persistence operation.:
    - Based on the method used it will create the respective queries internally
    - Based on POJO class object it identifies POJO class name and then it identifies properties and provides mapping POJO class and corresponding table and then hibernate software gets all table details. Then it prepares SQL query and executes it and returns results to Client Application.
  + Get Results.
  + Display Results.

**Steps to Prepare Hibernate Application:**

* **Hibernate persistence class:** The main intension of persistence object is to manage data which we want to store in database, on which data we want to perform retrieve operations, by using we are able to perform database operations like update, delete.
  + Persistence class must be POJO class (it must not extend or implement any predefined library except java.io.Serializable)
  + Persistence class must be public, non-abstract, non-final.
    1. public because to make available to hibernate software to create object.
    2. Non-abstract is to create objects.
    3. Non-final is to extend one persistent class to another persistence class to improve reusability.
  + In hibernate applications, we must prepare persistence classes w.r.t database tables, that is persistence classes properties names are not required to be matched with database column names, but their data types must be compatible.
  + In persistence classes, all properties must be declared as **private** and all methods are declared as **public**.
  + If we want to provide our own comparison mechanisms between persistence objects than it is suggestable to override object class provided equals method.
  + If we want to provide our own hash code values to the persistence objects in order to store in particular order in collection objects then it is suggestible to override object class provided hashcode() method.
  + If we want to provide constructor in persistence class then it is suggestible to provide “public” and 0-arg constructor, because hibernate software will search and execute that 0-arg constructor while creating persistence object.
* **Hibernate Mapping File:**
  + The main intension of mapping file is to provide mapping details between
    1. Class --- table
    2. Object --- Record
    3. Id prop --- primary key column
    4. Property ---column
    5. ---- ---- -----
  + In hibernate mapping file is responsible for following things
    1. Basic O-R mapping
    2. Component mapping
    3. Inheritance mapping
    4. Associations mapping
    5. Collections mapping
    6. -----

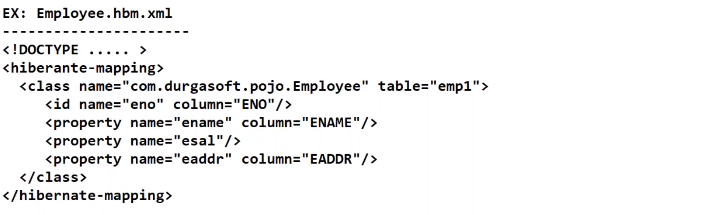
Ex:

A close up of a logo

Description generated with high confidence

**A close up of text on a white background

Description generated with very high confidence**

****

* + **We can use Annotations instead of Mapping file.**
  + If name property and column attribute are same, then there is no need of column tag
* **Hibernate Configuration File:**

The main intension of configuration file is to provide

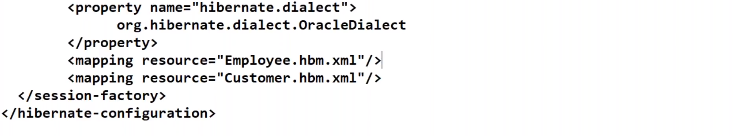
* JDBC parameters configuration
* Dialects configuration: on the basis of this property hibernate will prepare database dependent sql queries. Predefined dialect classes are provided in software.
* Mapping files configuration: by giving names and locations of mapping files
* Cache mechanisms configuration:
* Connection pooling mechanisms.
* Transaction configurations: Isolation levels
* ----
* More than one cfg file can be used depends upon the no of databases used.

A screenshot of a cell phone

Description generated with very high confidence

A screenshot of a cell phone

Description generated with very high confidence



* **Configuration file is not mandatory we set the values by properties file or XML file or from java code.**
* **Client/Test Application:**
  + Create Configuration object and hold Configuration Details
  + Create SessionFactory object
  + Create session object.
  + Create transaction object for performing non-select operations.
  + Perform persistence operation and commit the results.
  + Close SessionFactory and session objects.

1. **Configuration Object:** The main intension of configuration object is to manage all details which we provided in the configuration file.

Predefined class by hibernate software : “org.hibernate.cfg.Configuration”

Configuration cfg = new Configuration(); // only object will be created in heap memory with no details

cfg.configure(); //when we call this method details will be copied from configuration file to cfg object. (hibernate software will search for configuration file with name hibernate.cfg.xml and loads that file parses it and stores the content inside the object.

If you change the configuration file name then use **cfg.configure(“myconfig.xml”);**

**A screenshot of a cell phone

Description generated with very high confidence**

* It is thread safe object up to a particular database and heavy weight object.

1. **SessionFactory: (database setup)**

The main intension is to create hibernate session objects and it is able to load the driver, create connections, statements……

To represent session factory in hibernate, hibernate software has provided a separate predefined interface, that is **org.hibernate.SessionFactory**

Use following method to get session factory object.

Public SessionFactory buildSessionFactory();

SessionFactory sf = cfg.buildSessionFactory();

Note: It is heavy weight object and it is threadSafe up to particular database.

1. **Session Object:**

Session in hibernate applications has provided a **set of predefined methods** inorder to perform database operations and also provide transaction object.

To represent session object, hibernate software has provided a predefined interface **org.hibernate.session.**

To get this we need to use the following method public Session openSession().

**Ex: Session session = sf.openSession();**

Note: It is light weight object and it is not thread safe.

1. **Transaction Object:**

To perform only non-select operations.

Interface is in **org.hibernate.Transaction**

* By using beginTransaction()
  + Public Transaction beginTransaction()
  + It will generate transaction and it will start transaction
  + **Transaction tx = session.beginTransaction();**
* By using getTransaction()
  + It is able to return transaction object
  + To begin we use **public Tranaction begin();**
  + **Transaction tx = session.getTransaction();**
  + **tx.begin();**
* Use commit() or rollback () after success or failure of operations.

1. **Perform Persistence Operation; methods provided by session Interface**
   1. For insertion:
      1. Persist(---)
      2. Save(----)
   2. For retrieval
      1. get(---)
      2. load(---)
   3. For Updations
      1. Update(----)
      2. SaveOrUpdate(-----)
   4. For delete:
      1. delete(----)
2. **Close SessionFactory and Session Objects:**
   1. **Session.close()**
   2. **SessionFactory.close()**

**A screenshot of a cell phone

Description generated with very high confidence**

**A screenshot of a cell phone

Description generated with very high confidence**

**A screenshot of a cell phone

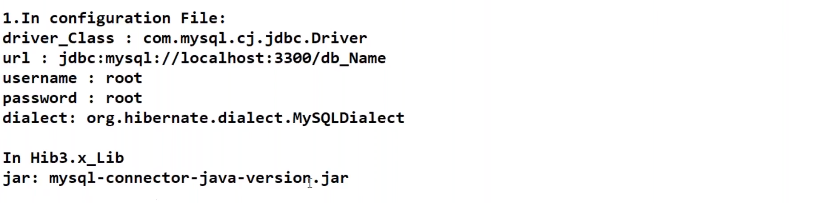
Description generated with very high confidence**

**A picture containing indoor

Description generated with very high confidence**

Before interacting with the database software will check for the respective object in the cache memory if not available then it interacts with the database and creates new bean object.

**For MYSQL database:**



**Struts Hibernate Integration:**

**Frameworks: collection of tools and API which is semi implemented application.**

* Web Frameworks:
* Provide environment to create only web applications.
* Struts, JSF, XWORK
* Application Framework
  + Provide environment to create all types of applications.
  + Spring

**Struts: MVC based Framework**

* **Elements:**
  1. View: structs has its own tag library or we can use html or jsp
  2. Web.xml:
  3. Controller: predefined Action servlet
  4. ActionForm/FormBean
  5. Action/Controller Component
  6. Struts Configuration File

**Outline Flow:**

**A close up of a map

Description generated with high confidence**

Name some of the properties you would require to configure for a databases in a standalone situation.

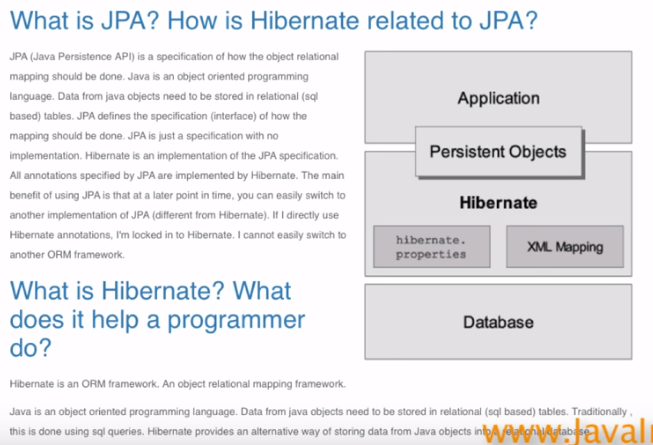
|  |  |
| --- | --- |
| Sr.No. | Properties & Description |
| 1 | hibernate.dialect  This property makes Hibernate generate the appropriate SQL for the chosen database. |
| 2 | hibernate.connection.driver\_class  The JDBC driver class. |
| 3 | hibernate.connection.url  The JDBC URL to the database instance. |
| 4 | hibernate.connection.username  The database username. |
| 5 | hibernate.connection.password  The database password. |
| 6 | hibernate.connection.pool\_size  Limits the number of connections waiting in the Hibernate database connection pool. |
| 7 | hibernate.connection.autocommit  Allows autocommit mode to be used for the JDBC connection. |

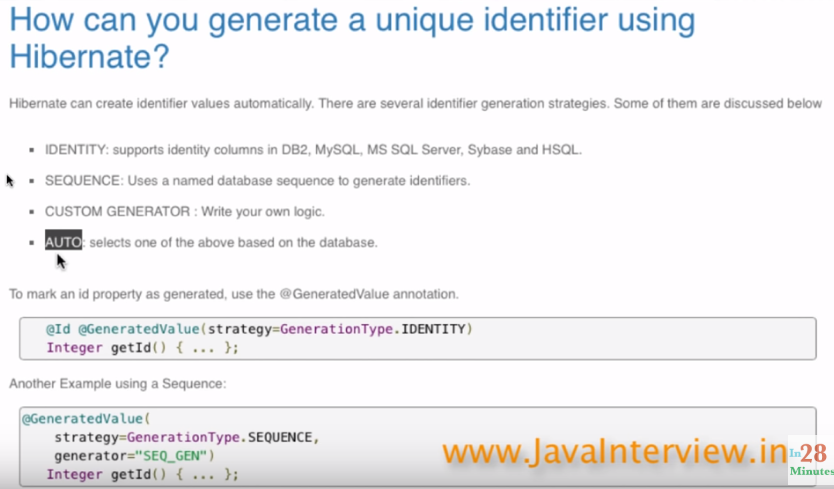
What are the three states of a persistent entity at a given point in time?

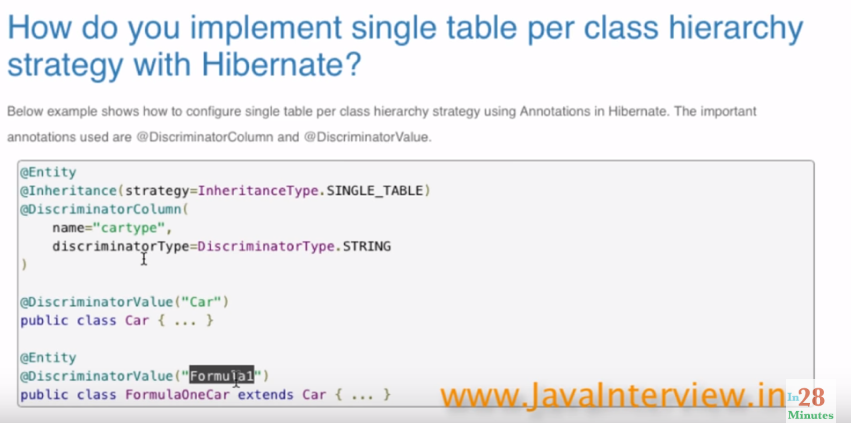
Instances may exist in one of the following three states at a given point in time −

* **transient** − A new instance of a persistent class which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.
* **persistent** − You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.
* **detached** − Once we close the Hibernate Session, the persistent instance will become a detached instance.

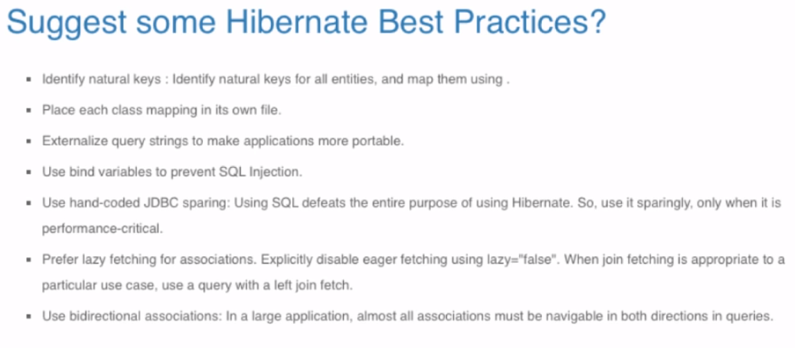
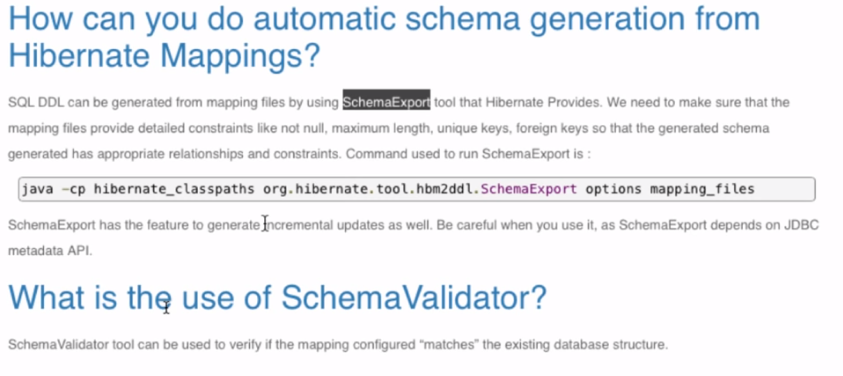
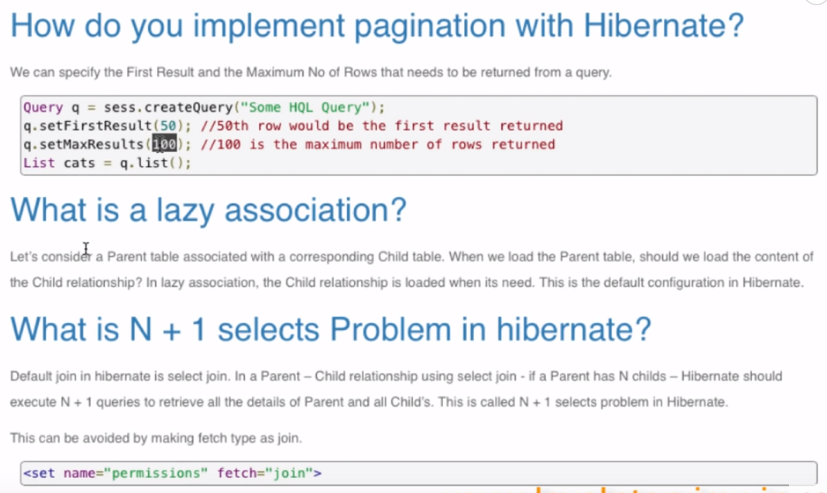
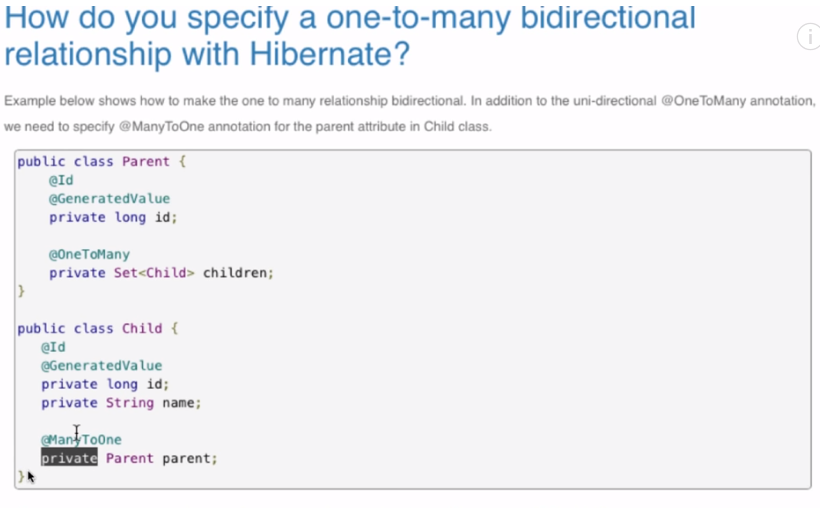
**Interview Questions:**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated**