Hibernate -ORM

What is ORM?

* ORM
* The **O**bject-**R**elational **M**apping (ORM) is the solution to handle all the mismatches.
* A programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc.

Advantages of ORM

* Lets business code access objects rather than DB tables.
* Hides details of SQL queries from OO logic.
* Based on JDBC 'under the hood‘
* No need to deal with the database implementation.
* Entities based on business concepts rather than database structure.
* Transaction management and automatic key generation.
* Fast development of application.

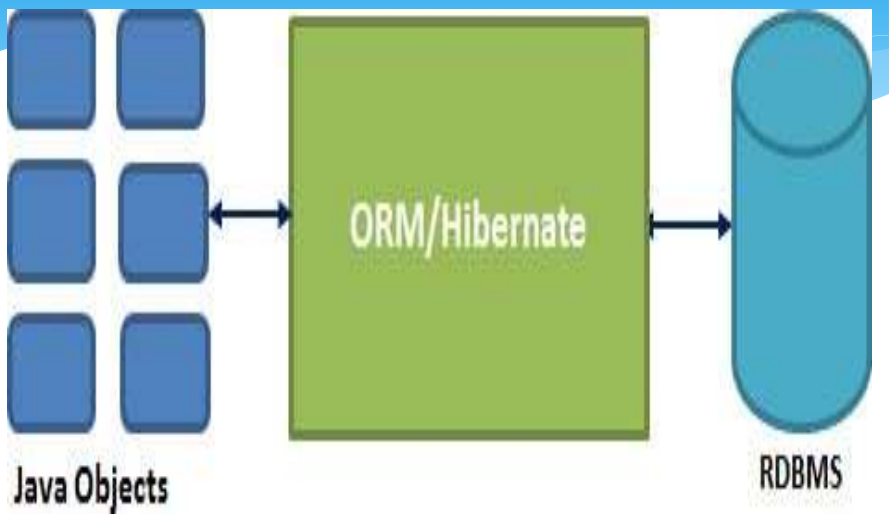
Java ORM Frameworks

* A persistent framework is an ORM service that stores and retrieves objects into a relational database.
  * Enterprise JavaBeans Entity Beans
  * Java Data Objects
  * Castor
  * TopLink
  * Spring DAO
  * Hibernate

Hibernate

* Hibernate is an Object-Relational Mapping(ORM) solution for JAVA and it raised as an open source persistent framework.
* It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.
* Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

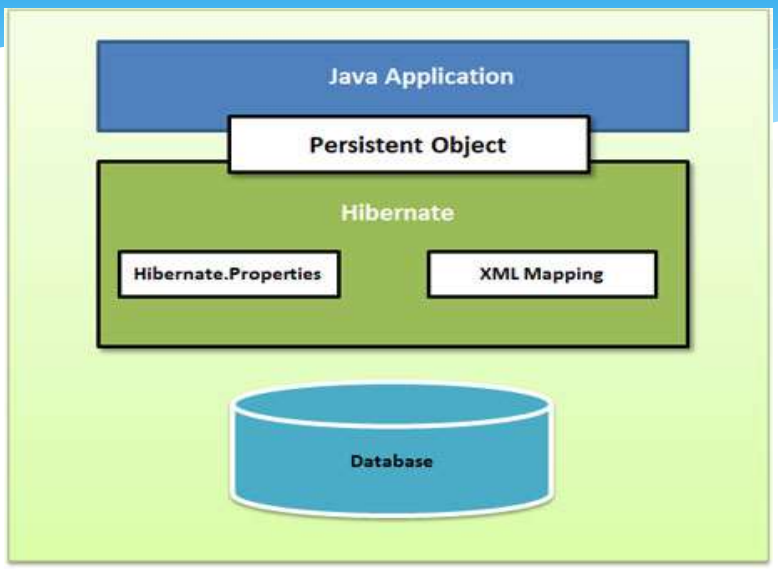
What Hibernate does?



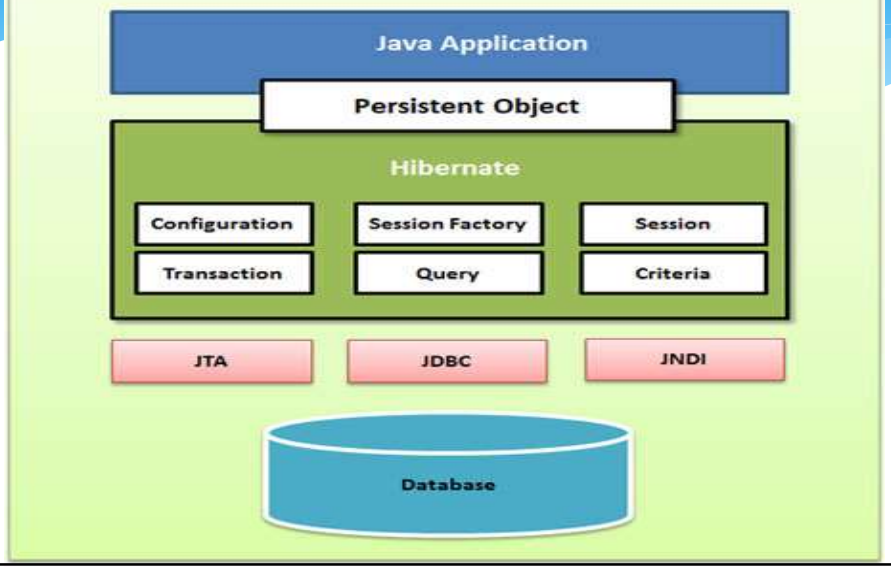
Hibernate Advantages

* Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.
* Provides simple APIs for storing and retrieving Java objects directly to and from the database.
* If there is change in Database or in any table then the only need to change XML file properties.
* Abstract away the unfamiliar SQL types and provide us to work around familiar Java Objects.
* Hibernate does not require an application server to operate.
* Manipulates Complex associations of objects of your database.
* Minimize database access with smart fetching strategies.
* Provides Simple querying of data.

Hibernate Architecture (high level)



Hibernate architecture (details)



Configuration Object

* The Configuration object is the first Hibernate object created in any Hibernate application.
* Created only once during application initialization.
* It represents a configuration or properties file required by the Hibernate.

Configuration Object

* The Configuration object provides two keys components:
  * **Database Connection:** This is handled through one or more configuration files supported by Hibernate. These files are **hibernate.properties** and **hibernate.cfg.xml**.
  * **Class Mapping Setup**
    This component creates the connection between the Java classes and database tables.

SessionFactory Object

* Configuration object is used to create a SessionFactory object.
* It configures Hibernate for the application using the supplied configuration file.
* A Session object is instantiated using SessionFactory.

SessionFactory Object

* It is a thread safe object and used by all the threads of an application.
* The SessionFactory is heavyweight object so usually created during application start up and kept for later use.
* One SessionFactory object id required per database using a separate configuration file.

Session Object

* A Session is used to get a physical connection with a database.
* This is lightweight object
* Needs to be instantiated each time an interaction is needed with the database.
* Persistent objects are saved and retrieved through a Session object.
* These are not thread safe.

Transaction Object

* A Transaction represents a unit of work with the database and most of the RDBMS supports transaction functionality.
* Transactions in Hibernate are handled by an underlying transaction manager and transaction (from JDBC or JTA).

Query Object

* Query objects use SQL or Hibernate Query Language (HQL) string to retrieve data from the database and create objects.
* A Query instance is used to bind query parameters, limit the number of results returned by the query, and finally to execute the query.

Hibernate sessions

* A Session is used to get a physical connection with a database.
* The Session object is lightweight and designed to be instantiated each time an interaction is needed with the database.
* Persistent objects are saved and retrieved through a Session object.

Hibernate sessions

* 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.

Hibernate persistence classes

* The entire concept of Hibernate is to take the values from Java class attributes and persist them to a database table.
* A mapping document helps Hibernate in determining how to pull the values from the classes and map them with table and associated fields.
* Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate.

Example

```
package com.ameya;
public class Employee {
   private int id;
   private String firstName;
   private String lastName;
   private int salary;
   public Employee() {}
   public Employee(String fname, String lname, int salary)    {
         this.firstName = fname;
         this.lastName = lname;
         this.salary = salary;                          }
   public int getId() { return id; }
   public void setId( int id ) { this.id = id; }
   public String getFirstName() { return firstName; }
   public void setFirstName( String first_name ) { this.firstName = first_name; }
   public String getLastName() { return lastName; }
   public void setLastName( String last_name ) { this.lastName = last_name; }
   public int getSalary() { return salary; }
   public void setSalary( int salary ) { this.salary = salary; } }
```

JPA Annotations

- @Entity
- @Table
- @Id
- @Column
- @GeneratedValue
- @GenericGenerator
- @OneToOne
- @PrimaryKeyJoinColumn
- @OneToMany
- @JoinColumn
- @OrderColumn
- @ManyToOne
- @ManyToMany
- @JoinTable joinColumns

81

Generator

- Types of generator
  - Assigned
  - Increment
  - Sequence
  - Identity
  - Hilo
  - Native
  - Foregin

assigned

* Supported by all DBs
* Default generator type
* Developer has to specify which PK value is assigned to object i.e. which column will be used as PK

increment

* Supported by all DBs
* Generates the id for new record by using formula as max of id value in database plus 1.

sequence

* Not supported by MySQL
* First ensures if DB supports this.
* If yes then while inserting a record, hibernate gets next value from the sequence.
* If programmer has created sequence that can be used

identity

* Not supported by Oracle
* Id value is generated by database & not hibernate.

hilo

* Supported by all DBS
* For 1st record id is 1
* For next 32768
* For next previous+32768

native

* Decides whether to use identity or sequence or hilo.
* It checks if database supports identity.
* If not checks for sequence.
* If not hilo will be used.

foreign

* Used in one-to-one mapping

Auto

* is the default generation type and lets the persistence provider choose the generation strategy
* If you use Hibernate as your persistence provider, it selects a generation strategy based on the database specific dialect.