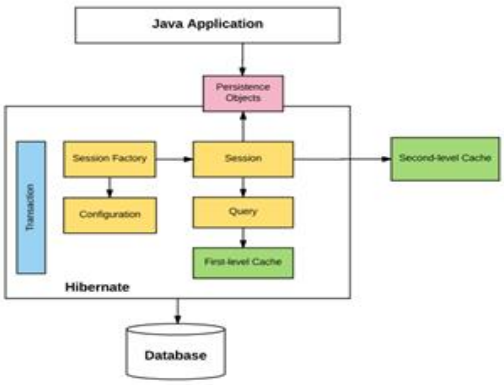
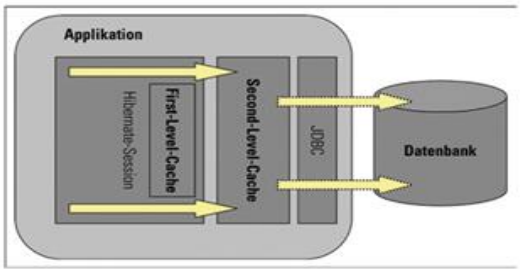
HIBERNATE - open persistence and ORM framework for rational databases supporting JPA!





FEATURES - ORM, JPA, Idiomatic, performance (lazy initialization)

CONFIGURATION - done in hibernate.properties or hibernate.cfg.xml or with @Configuration annotation.

SESSION FACTORY - one factory per connection, it is instantiated from configuration and provides us session objects

SESSION - It represents the interaction between application and database (org.Hibernate.Session)

QUERY - hbm representation of a query, framework provide support for different query types NamedQuery and CriteriaAPI

CACHE - session cache (created with every session) & global cache

load()

- refers first level cache before loading from database and throws org.hibernate.ObjectNotFoundException if not found in database

- returns a proxy object for lazy loading

- has three variants; Object load(String entityName, Serializable id), Object load(Class theEntityClass, Serializable id), void load(Object entity, Serializable id)

get()

- fetches the object directly from database and returns null if not found in database

- returns a fully initialized object

- has two variants; Object get(Class theEntityClass, Serializable id), Object get(String "entityName", Serializable id)

update()

- saves a detached object if there is no persistent object in the current session

- makes the detached object persistent

merge()

- merges the changes of a detached object with existing object in session/loads it from DB

- returns the merged persistent object and detached object remains as it is

refresh()

- method to re-populate the entity with latest data available in database

- has two variants; public void refresh(Object object) throws HibernateException, public void refresh(Object object, LockMode lockMode) throws HibernateException

save()

- calling save() on the same entity in one session works while in different sessions violates primary key constraint

- one should not call save() method on a persistent entity, any changes done to persistent entity is automatically saved.

HIBERNATE EQUALITY

- Requesting a persistent object again from the same Hibernate session returns the “same java instance” of a class.

- Requesting a persistent object from the different Hibernate session returns “different java instance” of a class.

- Hibernate wraps the actual object in a proxy so always use the getter methods inside instead of actual properties to compare.

ASSOCIATIONS

- OneToOne : Both the entities can be owner but only one should be made owner using the mappedBy property otherwise there will be cyclic dependency. This element is only specified on the inverse (non-owning) side of the association.

- OneToMany : The many end must be marked with this annotation and both the entities are responsible for creation and maintaining the association.

- ManyToOne : The is just the view from other end.

- ManyToMany : Enther end of the association can be made owner using the mappedBy property.

@OneToMany(cascade = CascadeType.[PERSIST, REFRESH, REMOVE, MERGE, DELETE, ALL], orphanRemoval = true, fetch = FetchType.[LAZY, EAGER])

Apart from JPA provided cascade types, there is one more cascading operation in hibernate which is not part of the normal set above discussed, called “orphan removal“.

LAZY LOADING

- The default behaviour is to load ‘property values eagerly’ and to load ‘collections lazily’.

- @OneToMany and @ManyToMany associations are defaulted to LAZY loading

- @OneToOne and @ManyToOne are defaulted to EAGER loading

N+1 SELECTS PROBLEM

- In a lazy association, O/R implementation would SELECT psrent entities and then do N additional SELECTs for getting the information of children.

- One solution is to use HQL's fetch join

- Another is to use criteria query with FetchMode.EAGER for the association