* Hibernate is a framework to persist java objects into database.
* It minimizes JDBC code.
* It is Object relational Mapping framework.

|  |  |
| --- | --- |
| Java | Database |
| Class | Table |
| Data members | Columns |
|  |  |

* In JDBC we need to write the SQL queries.
* In Hibernate there is HQL (Hibernate Query language) We write query with class instead of sql.
* Hibernate uses the JDBC to connect to database.

Hibernate 5.2 .It requires java 8

You will need

1)Database server

2)Java 8

3)JDBC jar. Hibernate Jar

4)IDE

Steps:

1)Hibernate Configuration

\* create bean of session factory. Set datasource and setHibernateProperties

\*In datasource

* JDBCURL 3306 port
* User
* Password

\*In setHibernateProperties

* Connection\_pool =How many connections are allowed

Min n default=1 ,,,, Max=100

* Dialect: SO hibernate can communicate with MySQL

For each database there is different dialect.

Session Factory :

The SessionFactory is a heavyweight object; it is usually created during application start up and kept for later use. You would need one SessionFactory object per database using a separate configuration file. So, if you are using multiple databases, then you would have to create multiple SessionFactory objects.

Session Object

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.

The session objects should not be kept open for a long time because they are not usually thread safe and they should be created and destroyed them as needed.

Primary Key

Uniquely identify each row in table

Can not be NULL

Foreign Key

It is used for defining relationship between two tables.

Foreign key of one table is primary key of another table.

Ex.

|  |  |  |
| --- | --- | --- |
| Student\_id(PK) | Marks | SubjectID(FK) |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| Subject\_id(pk) | Subject |
|  |  |
|  |  |

Cascade

you can apply same operation to related entities.

Eager and Lazy

Eager:Retrieve all the data

Lazy:retrieve data on request

How to integrate spring and hibernate

1)Spring Configuration

a)Configure dispatcher servlet

b)View Resolver

Annotations:

1)There are 3 ways to use component scan

 1)Annotation: @ComponentScan  provides beans auto-detection facility.

2)Web.xml:

<context:component-scan base-package="com.websystique.spring" />

3) In main method using Application Context

ApplicationContext context=**new** ClassPathXmlApplicationContext("SpringConfig.xml");

Person emp=(Person) context.getBean("person");;

2) @Configuration indicates that this class contains one or more bean methods annotated with @Bean producing beans manageable by spring container.

3) @PropertySource(value = { "classpath:application.properties" })

Annotation providing a convenient and declarative mechanism for adding a [PropertySource](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/core/env/PropertySource.html" \o "class in org.springframework.core.env) to Spring's [Environment](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/core/env/Environment.html). To be used in conjunction with @[Configuration](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Configuration.html) classes.

There are three ways of Mapping

1)OneToOne

2)OneToMany

3)ManyToMany

0. SessionFactory API

getCurrentSession vs openSession

public Session openSession() throws HibernateExc

opens new session from SF,which has to be explicitely closed by prog.

public Session getCurrentSession() throws HibernateExc

Opens new session , if one doesn't exist , otherwise continues with the exisitng one.

save --- if u give some non-null id while calling save(ref) --doesn't give any exc.

Ignores ur passed id & creates its own id & inserts a row.

public void persist(Object transientRef)

if u give some non-null id while calling persist(ref) --gives exc

org.hibernate.PersistentObjectException: detached entity passed to persist:

why its taken as detached ? ---non null id.

3. saveOrUpdate

null id -- fires insert (works as save)

non-null BUT existing id -- fires update (works as update)

non-null BUT non existing id -- throws StaleStateException --to indicate that we are trying to delete or update a row that does not exist.

5. update

Session API

public void update(Object object)

Update the persistent instance with the identifier of the given detached instance.

I/P --detached POJO containing updated state.

Same POJO becomes persistent.

Exception associated :

1. org.hibernate.TransientObjectException: The given object has a null identifier:

i.e while calling update if u give null id. (transient ----X ---persistent via update)

2. org.hibernate.StaleStateException --to indicate that we are trying to delete or update a row that does not exist.

3.

org.hibernate.NonUniqueObjectException: a different object with the same identifier value was already associated with the session

6. public Object merge(Object ref)

Can Transition from transient -->persistent & detached --->persistent.

Regarding Hibernate merge

1. The state of a transient or detached instance may also be made persistent as a new persistent instance by calling merge().

2. API of Session

Object merge(Object object)

3.

Copies the state of the given object(can be passed as transient or detached) onto the persistent object with the same identifier.

3.If there is no persistent instance currently associated with the session, it will be loaded.

4.Return the persistent instance. If the given instance is unsaved, save a copy of and return it as a newly persistent instance. The given instance does not become associated with the session.

5. will not throw NonUniqueObjectException --Even If there is already persistence instance with same id in session.

7.public void evict(Object ref)

It detaches a particular persistent object

detached or disassociates from the session.

(Remove this instance from the session cache. Changes to the instance will not be synchronized with the database. )

8.

void clear()

When clear() is called on session object all the objects associated

with the session object(L1 cache) become detached.

But Databse Connection is not closed.

(Completely clears the session. Evicts all loaded instances and cancel all pending saves, updates and deletions)

9. void close()

When close() is called on session object all

the objects associated with the session object become detached(l1 cache is cleared) and also closes the Database Connection.

. void flush()

When the object is in persistent state , whatever changes we made to the object

state will be reflected in the databse only at the end of transaction.

BUT If we want to reflect the changes before the end of transaction

(i.e before commiting the transaction )

call the flush method.

(Flushing is the process of synchronizing the underlying DB state with persistable state of session cache )

boolean contains(Object ref)

The method indicates whether the object is

associated with session or not.(i.e is it a part of l1 cache ?)

void refresh(Object ref) -- ref --persistent or detached

This method is used to get the latest data from database and make

corresponding modifications to the persistent object state.

(Re-reads the state of the given instance from the underlying database

1. get() loads the data as soon as it’s called whereas load() returns a proxy object and loads data only when it’s actually required, so load() is better because it support lazy loading.
2. Since load() throws exception when data is not found, we should use it only when we know data exists.
3. We should use get() when we want to make sure data exists in the database.