Hibernate

- Configuration
- SessionFactory
- DataSet with annotations
- Session

Hibernate ORM

Hibernate is an object-relational mapping tool for the Java programming language.

It provides a framework for mapping an object-oriented domain model to a relational database.

Hibernate solves object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Hibernate's primary feature is mapping from Java classes to database tables, and mapping from Java data types to SQL data types.

Configuration

Maven

```
<dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.2.10.Final</version>
</dependency>
```

org.hibernate.cfg.Configuration

Configuration configuration = new org.hibernate.cfg.Configuration();

configuration.setProperty(propertyName, propertyValue);

propertyName	propertyValue
hibernate.dialect	org.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class	com.mysql.cj.jdbc.Driver
hibernate.connection.url	jdbc:mysql://localhost:3306/db_example
hibernate.connection.username	tully
hibernate.connection.password	tully
hibernate.connection.useSSL	false
hibernate.enable_lazy_load_no_trans	true
hibernate.show_sql	true
hibernate.hbm2ddl.auto	update

hibernate.show_sql

Hibernate create requests to DB for you. You can see them in log in show_sql == true

hibernate.hbm2ddl.auto

Automatically validates or exports schema to the database when the Hibernate initialized.

The list of possible options are:

- **validate**: validate the schema, makes no changes to the database.
- **update**: update the schema.
- **create**: creates the schema, destroying previous data.
- **create-drop**: drop the schema when the application is stopped.

addAnnotatedClass(...)

configuration.addAnnotatedClass(DataSet.class) reads metadata from the annotations associated with this class.

Hibernate needs to know about your DataSets (Entities).

They must be added to the configuration.

org.hibernate.SessionFactory

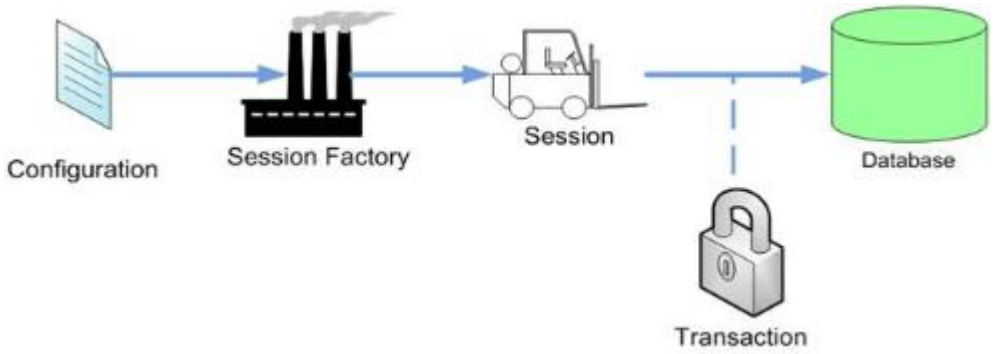
Factory of Sessions (Sessions are main runtime interface between a Java application and Hibernate).

Usually an application has a single SessionFactory instance and threads servicing client requests obtain Session instances from this factory.

The internal state of a SessionFactory is immutable.

Once it is created this internal state is set.

This internal state includes all of the metadata about Object/Relational Mapping.



How to create a SessionFactory

```
private static SessionFactory createSessionFactory(Configuration configuration) {
  StandardServiceRegistryBuilder builder = new StandardServiceRegistryBuilder();
  builder.applySettings(configuration.getProperties());
  ServiceRegistry serviceRegistry = builder.build();
  return configuration.buildSessionFactory(serviceRegistry);
}
```

org.hibernate.Session

Session is the main runtime interface between a Java application and Hibernate.

This is the central API class abstracting the notion of a persistence service.

The lifecycle of a Session is bounded by the beginning and end of a transaction.

The main function of the Session is to offer

- create,
- read,
- delete

operations for instances of mapped entity classes.

How to use Sessions

```
// like Executor for connection, but for SesstionFactory
private <R> R runInSession(Function<Session, R> function) {
  try (Session session = sessionFactory.openSession()) {
    Transaction transaction = session.beginTransaction();
    R result = // use session object to create query in JPQL
    transaction.commit();
    return result;
  }
}
```

Example

```
StandardServiceRegistryBuilder builder = new StandardServiceRegistryBuilder();
builder.applySettings(configuration.getProperties());
ServiceRegistry serviceRegistry = builder.buildServiceRegistry();

SessionFactory sessionFactory = configuration.buildSessionFactory(serviceRegistry);

Session session = sessionFactory.openSession();
Transaction transaction = session.beginTransaction();

System.out.append(transaction.getLocalStatus().toString());

session.close();
sessionFactory.close();
```