spring

HIBERNATE

SUNBEAM

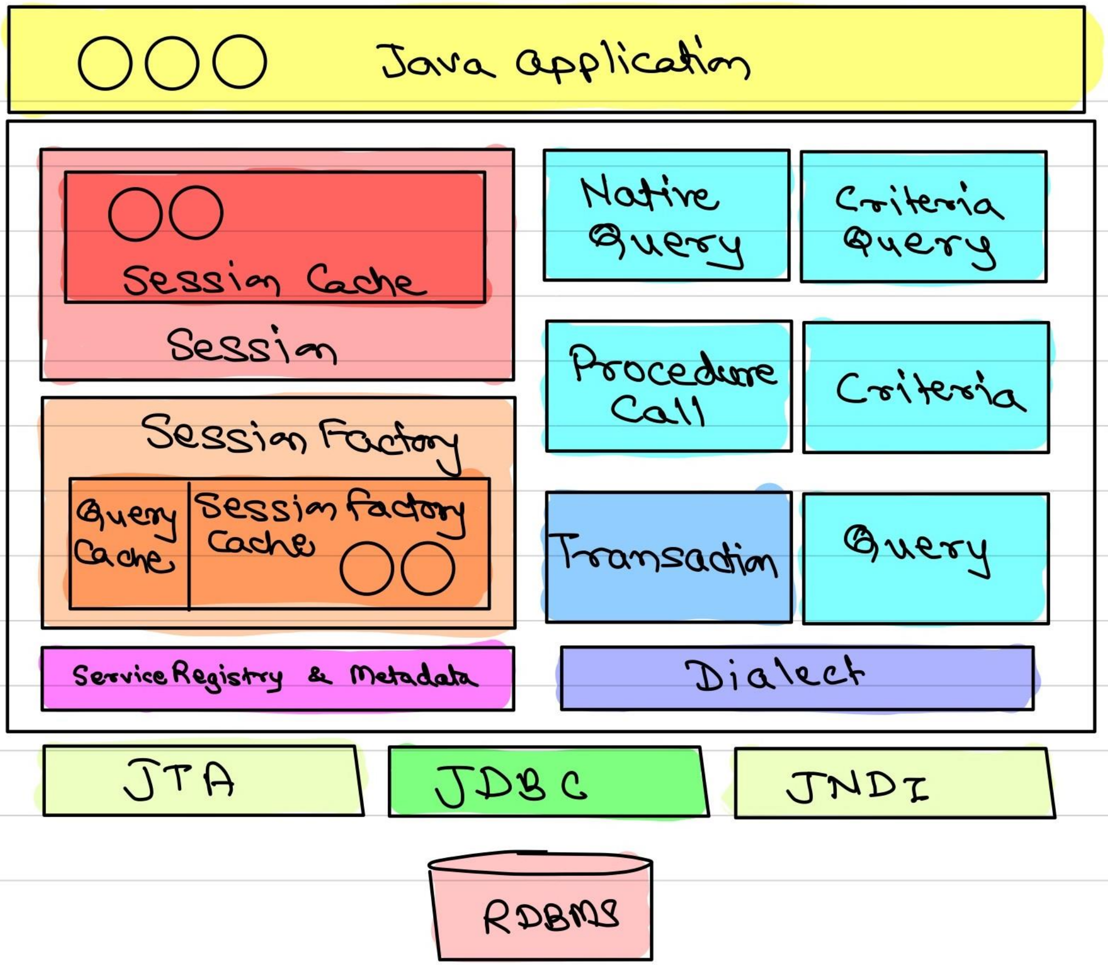
Trainer: Nilesh Ghule

Wake up from Hibernate, Spring up!!!

Agenda

- Mini-Project Idea ✓
- Hibernate Architecture ✓
- Hibernate Configuration ✓
- Hibernate5 Bootstrapping ✓
- ORM using Annotation ✓
- ORM using XML file ✓
- CRUD operations & Transactions ✓
- openSession() vs getCurrentSession() ✓
- Hibernate entity life cycle ✓
- Criteria and DetachedCriteria

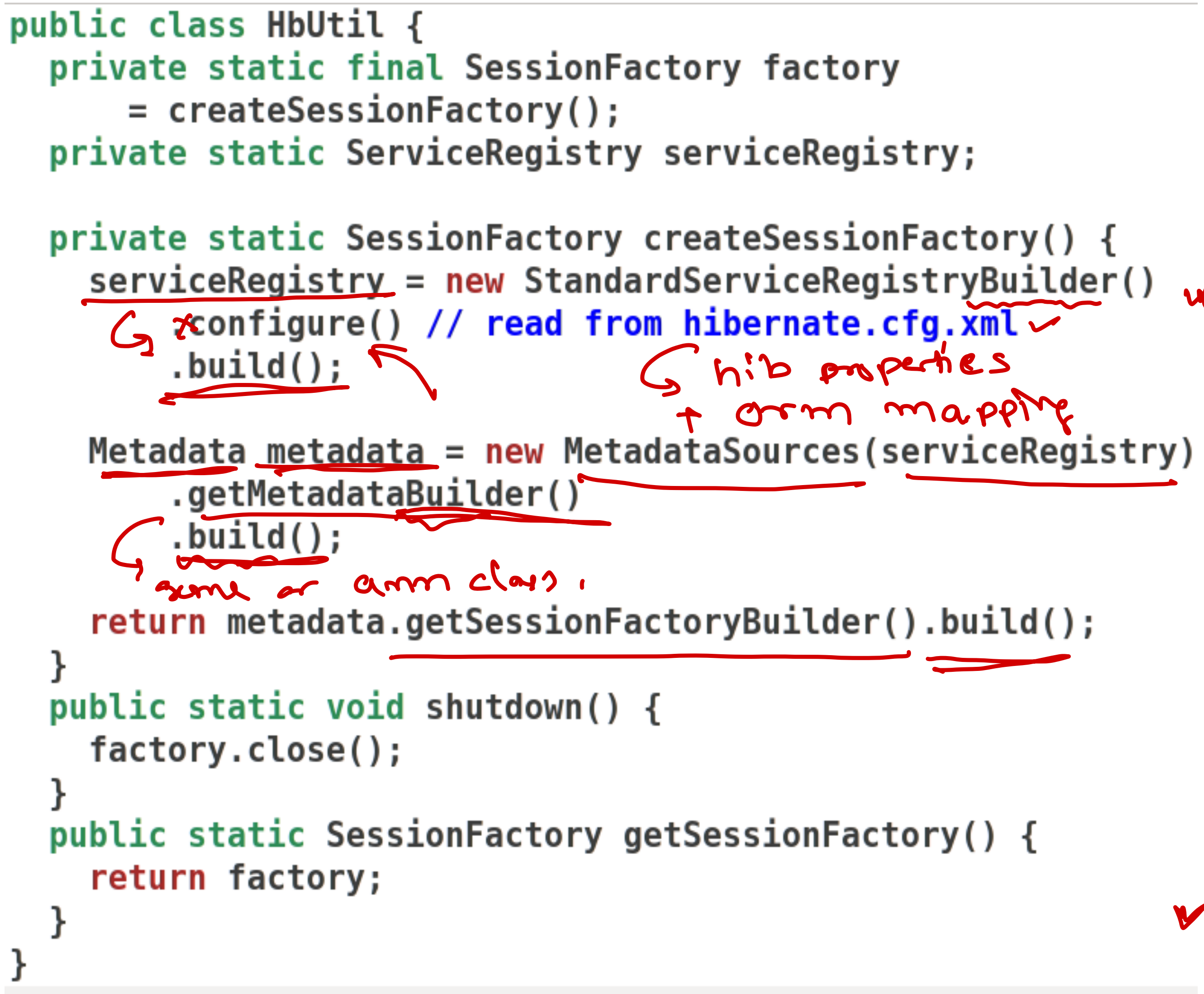
Hibernate



- SessionFactory
  - One SessionFactory per application (per db).
  - Heavy-weight object. Not recommended to create multiple instances.
  - Thread-safe. Can be accessed from multiple threads (synchronization is built-in).
  - Typical practice is to create singleton utility class for that.
- Session
  - Created by SessionFactory & it encapsulate JDBC connection.
  - All hibernate operations are done on hibernate sessions.
  - Is not thread-safe. Should not access same session from multiple threads.
  - Light-weight. Can be created and destroyed as per need.

Hibernate5 Bootstrapping

mapping class = ? - - - > ≡



```
public class HbUtil {
  private static final SessionFactory factory
      = createSessionFactory();
  private static ServiceRegistry serviceRegistry;

  private static SessionFactory createSessionFactory() {
    serviceRegistry = new StandardServiceRegistryBuilder()
        .configure() // read from hibernate.cfg.xml
        .build();

    Metadata metadata = new MetadataSources(serviceRegistry)
        .getMetadataBuilder()
        .build();

    return metadata.getSessionFactoryBuilder().build();
  }
  public static void shutdown() {
    factory.close();
  }
  public static SessionFactory getSessionFactory() {
    return factory;
  }
}
```

hib properties + orm mapping

xml or ann class .

Hibernate 5 Bootstrapping

- Create ServiceRegistry. → hib props
- Create Metadata. → orm mapping
- Create SessionFactory.

ServiceRegistry → Java SPI

- ServiceRegistry is interface. Connector Service, Tx service
- Some implementations are StandardServiceRegistry, BootstrapServiceRegistry, — minimal EventListenerRegistry, ...
- Add, manage hibernate services.

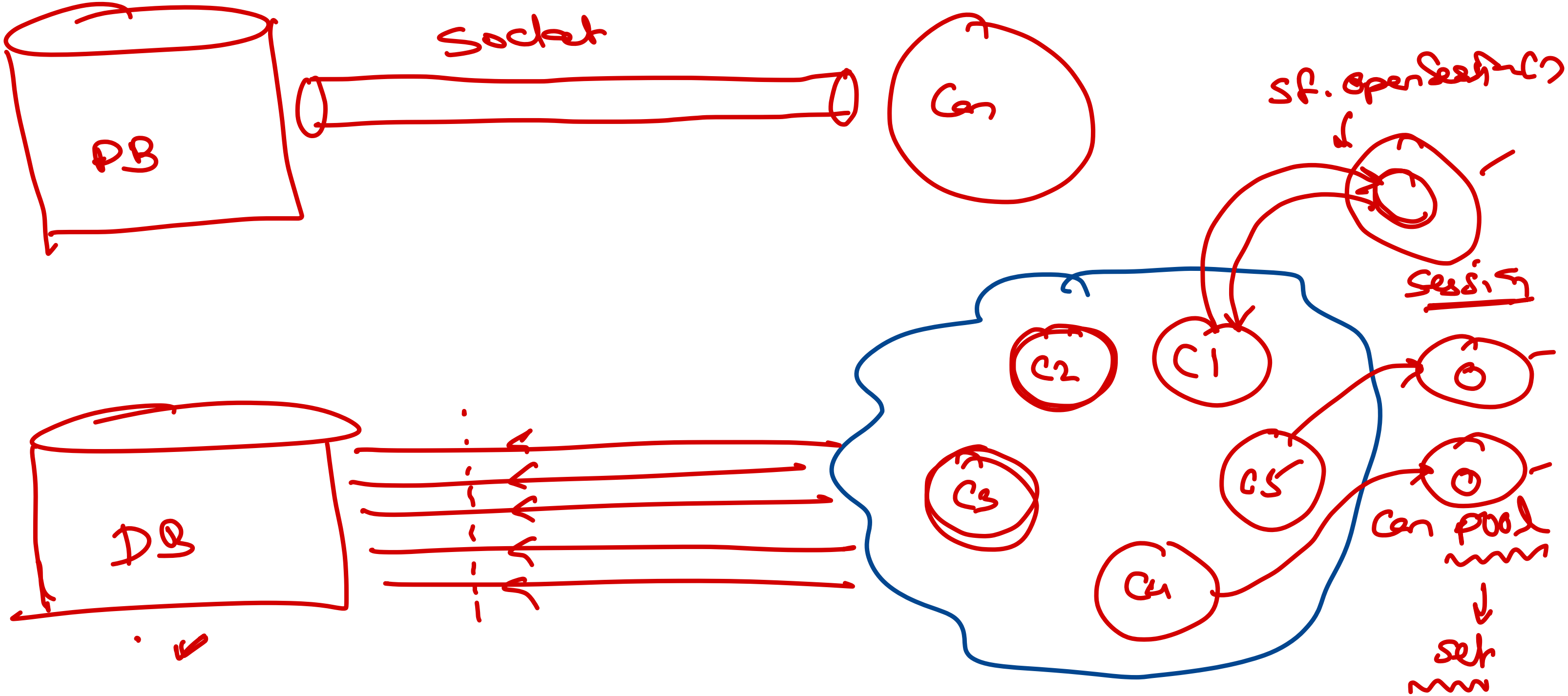
Metadata

- Represents application's domain model & its database mapping.

https://docs.jboss.org/hibernate/orm/4.3/topical/html/registries/ServiceRegistries.html

Jdbc
Connection

DM.getCon()



Hibernate – ORM using Annotations

- Hibernate3 added annotations for ORM.
- ORM using annotations
  - @Entity
  - @Table class ←→ table
  - @Column field ←→ column
  - @Id → PK field. (must)
  - @Temporal ?
  - @Transient → skip mapping field → column.
- @Column can be used on field level or on getter methods. (recommended) ✓

@Column → field level

- fields are directly initialized by hibernate when data is read from db. No setter code is executed.

@Column → getter level

- fields are initilized using setter method by hib when data fetched from db.

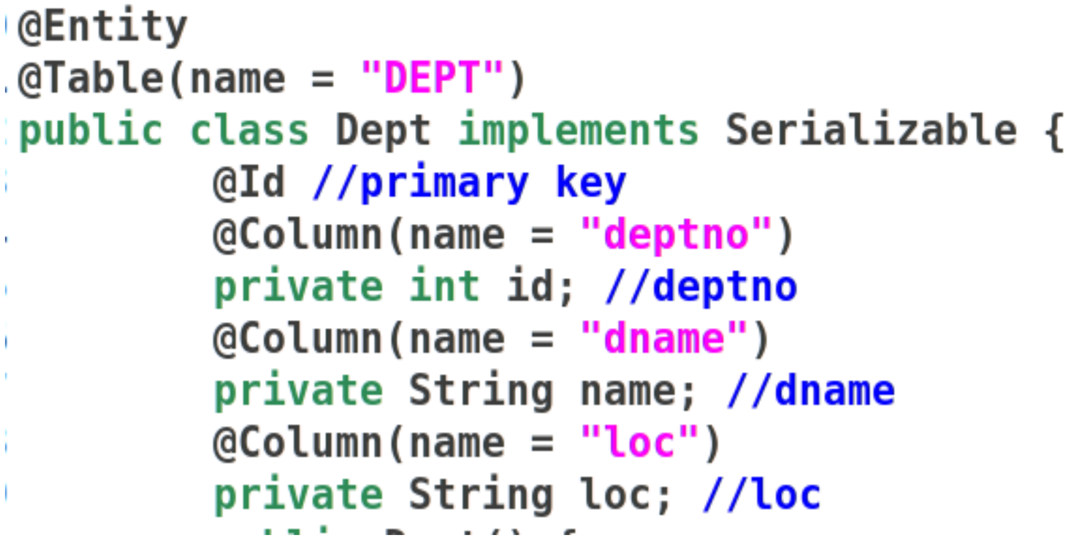
Hibernate – ORM using XML

- Earlier versions does ORM using .hbm.xml files.

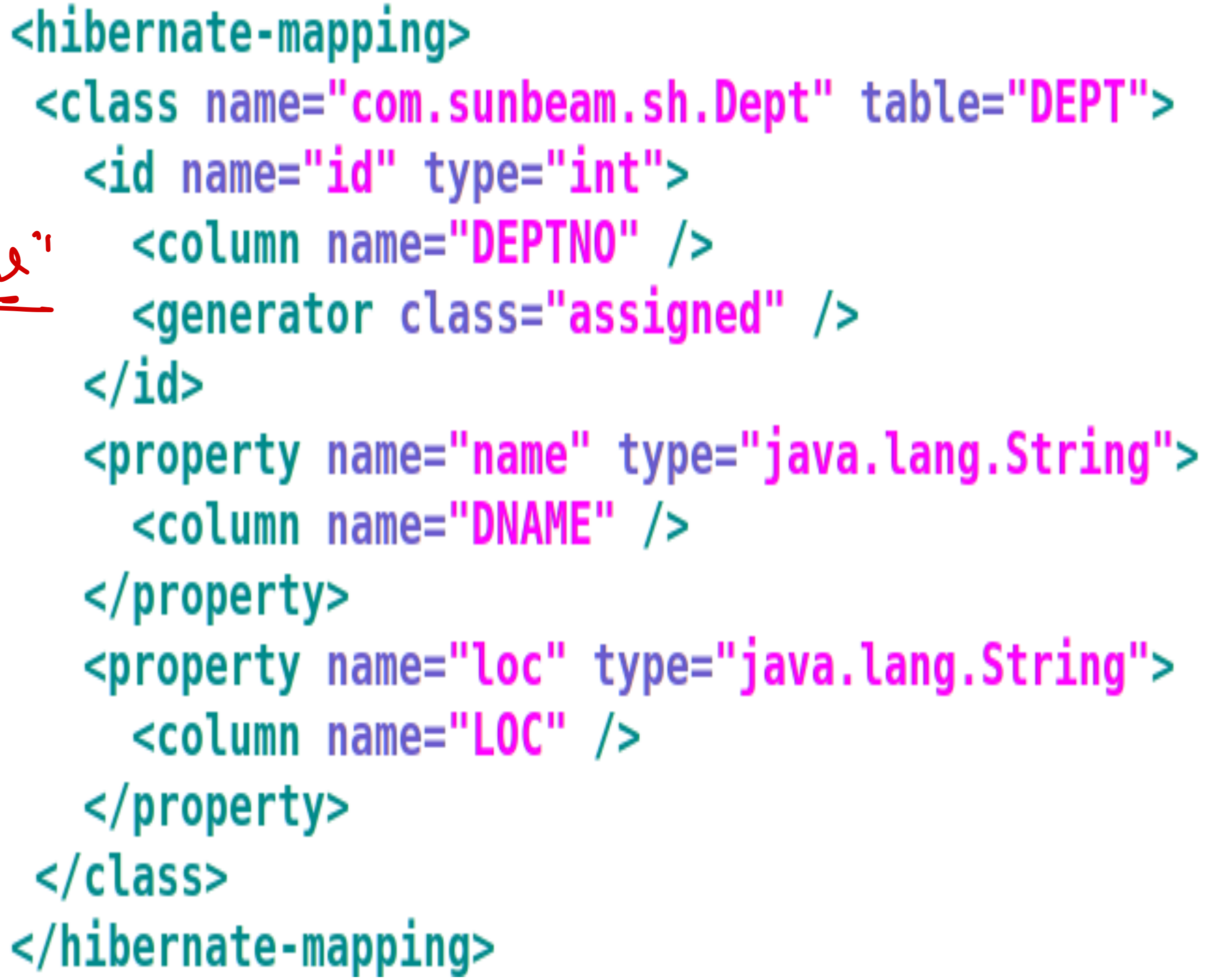
↳ inside package → entity class

hib.cf's xml

<mapping resource=" pkg/Dept.hbm.xml"



```
@Entity
@Table(name = "DEPT")
public class Dept implements Serializable {
        @Id //primary key
        @Column(name = "deptno")
        private int id; //deptno
        @Column(name = "dname")
        private String name; //dname
        @Column(name = "loc")
        private String loc; //loc
```



```
<hibernate-mapping>
 <class name="com.sunbeam.sh.Dept" table="DEPT">
   <id name="id" type="int">
     <column name="DEPTNO" />
     <generator class="assigned" />
   </id>
   <property name="name" type="java.lang.String">
     <column name="DNAME" />
   </property>
   <property name="loc" type="java.lang.String">
     <column name="LOC" />
   </property>
 </class>
</hibernate-mapping>
```

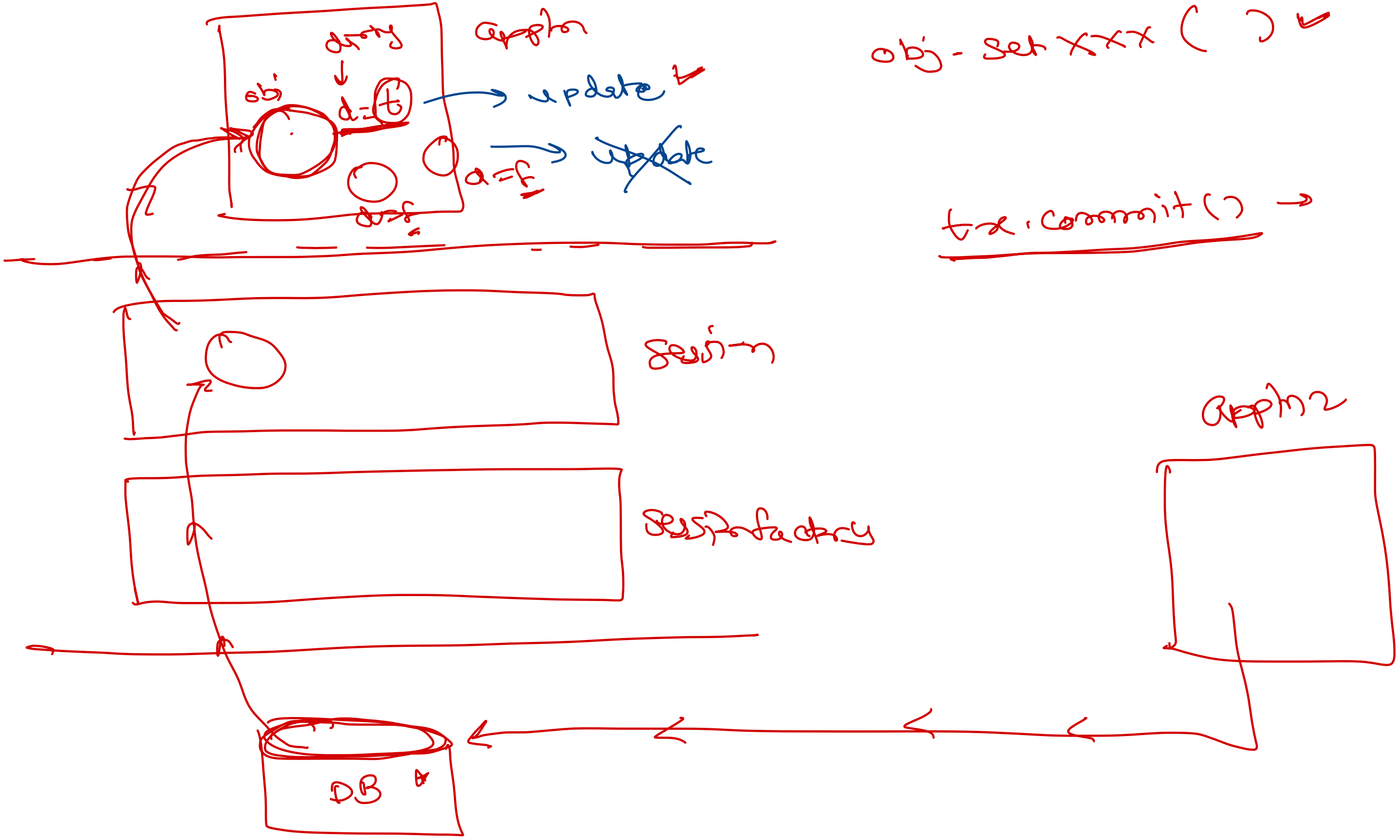
CRUD operations

JPA - ORM spec

Hibernate - ORM impl

- Hibernate Session methods
  - get() b = session.get(Book.class, pk); return null if record not found.
  - load() b = session.load(Book.class, pk); - return proxy with pk. actual data is fetched from db when entity obj fields are acessed. if record not found, throws exception.
  - find() b = session.find(Book.class, pk);
  - save() - insert record into db. & return pk. - hibernate specific (not jpa compliant).
  - persist() - add obj into session. - inserted while commiting tx - jpa compliant.
  - update() - add obj into session & set dirty flag to true. - so that at tx commit, records are updated.
  - saveOrUpdate() → check if record is present in db - SELECT → if not found, do insert op → INSERT → if found, do update op → UPDATE
  - merge() → similar to saveOrUpdate()
  - delete() → delete the record from the database.
  - remove() - same as delete
  - evict() - remove the object from session. - after this changes in object will not be updated in db. → detach() → Same as evict() - JPA compliant
  - clear() → removes all objects from session.
  - refresh() - session.refresh(obj); → load data of obj again from db.

- Hibernate transactions
  - tx = session.beginTransaction()
  - tx.commit()
  - tx.rollback()


appln
dirty
obj
d=t
update
d=f
~~update~~
d=f
session
SessFactory
DB
appln2
obj.setXXX()
tx.commit()

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>