Inner Bean

→ Keeping a bean limited to a specific bean

eg:- <bean id: "alien" class: "- - - >

<property name: "lap" >

<bean id: "lap" class: . . . > </bean>

</property>

</bean>

So, here the bean lap can only be accessed by Alien class.

(#) Spring allows 3 types of configuration → 1) xml Configuration

→ 2) Java Based Config

→ 3) Annotations.

Hibernate Continued

Steps to create a hibernate connection :-

1.) Create a class with @Entity annotation
It will be mapped to the table in database

2.) In src/main/resources :-
create :- hibernate.cfg.xml

→ This file defines :- database connection details
→ the sql dialect
→ the mapping to entity class.

eg:- configur hibernate.cfg.xml

```
<hibernate-configuration>
    <session-factory>
        <property name="org.hibernate.connection.driver_class">
            com.mysql.cj.jdbc.Driver </property>

        <property name="hibernate... url"> </property>

        <property ... username>
        <... ... password>

        <... ... dialect>

    </session-factory>
</hibernate-configuration>
```

(or) Same as jdbc → define driver, url, username, password

3.) In the main class.

- Create an object

```
Configuration cfg = new Configuration();
cfg.configure("hibernate.cfg.xml");
cfg.addAnnotatedClass(Student.class);
```
} loads the setting

```
SessionFactory sessf = cfg.buildSessionFactory();
Session session = sf.openSession();
```
sessf → Similar to JDBC Connection
session → Only once per use

```
Transaction tr = session.beginTransaction();
session.persist(s);
transaction.commit();
```
} to save any changes in db.