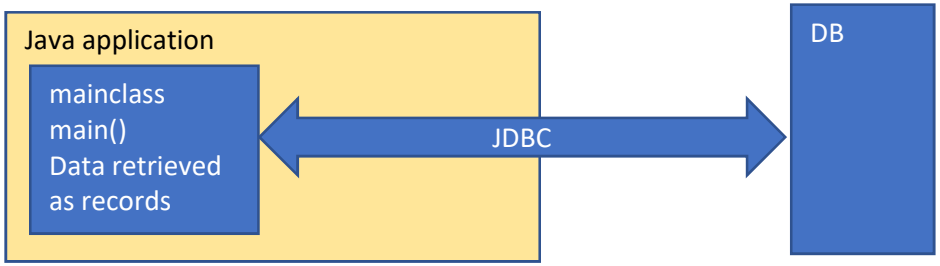
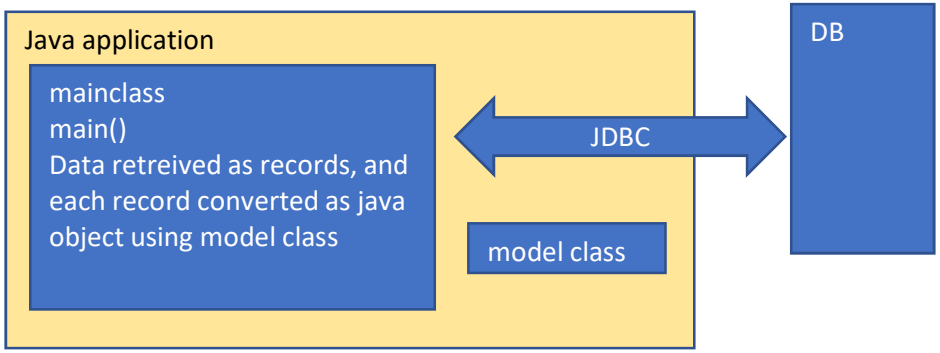
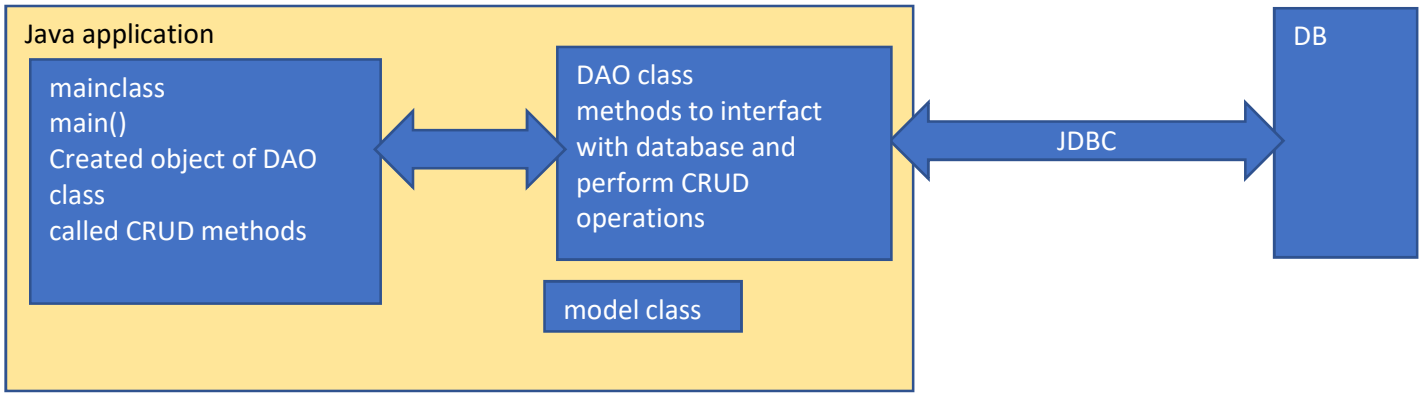
JDBC example
in corejava

Java Application <-> DB (MySQL)





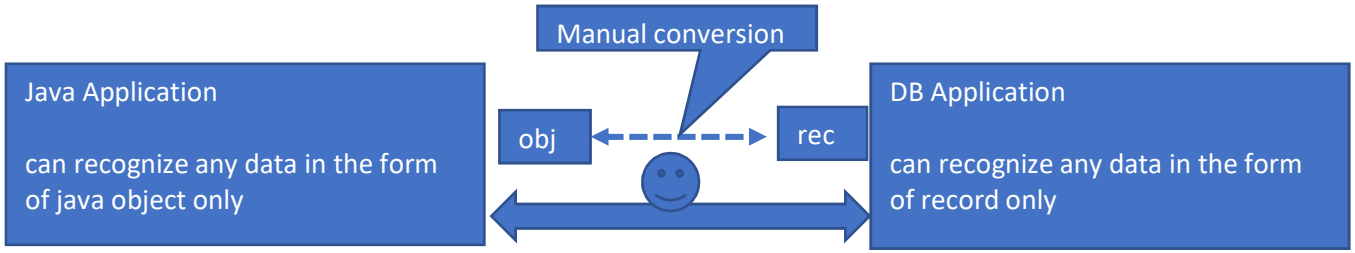


Methods in DAO

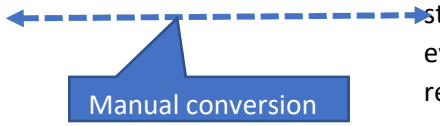
```
public boolean addBook(Book book){  }
public List<Book> getBooks() { }
public boolean updateBook(Book book){ }
public boolean deleteBook(int bookid){ }
public Book getBookById(int bookid) { }
```

load driver
statement (query)
fill params in prepared st (query)
execute respective method (ddl/dml/dql)
convert rec type data into object (viceversa)

In JDBC program
Table in DB and class in Java are mapped manually



Structure of data decided by model class
every data stored as object
object has member variables inside



structure of data decided by table
every data stored as record
record has fields

ORM tool
Hibernate

Advantages

1 java model class(es) are maped with table(s) in database automatically
2 rec type data <-> object type
3 comes with collection of methods to perform different operations on db

ex
select * from ....
list()
insert into ....
save()
delete from ...
update table set ....
delete()
update()
select * from __ where bid =__
get() /load()

SessinFactory

Used to provide sessions for programmers

Session

Temporary object for programmer to perfrom any CRUD opertation with database

Hibernate configuration

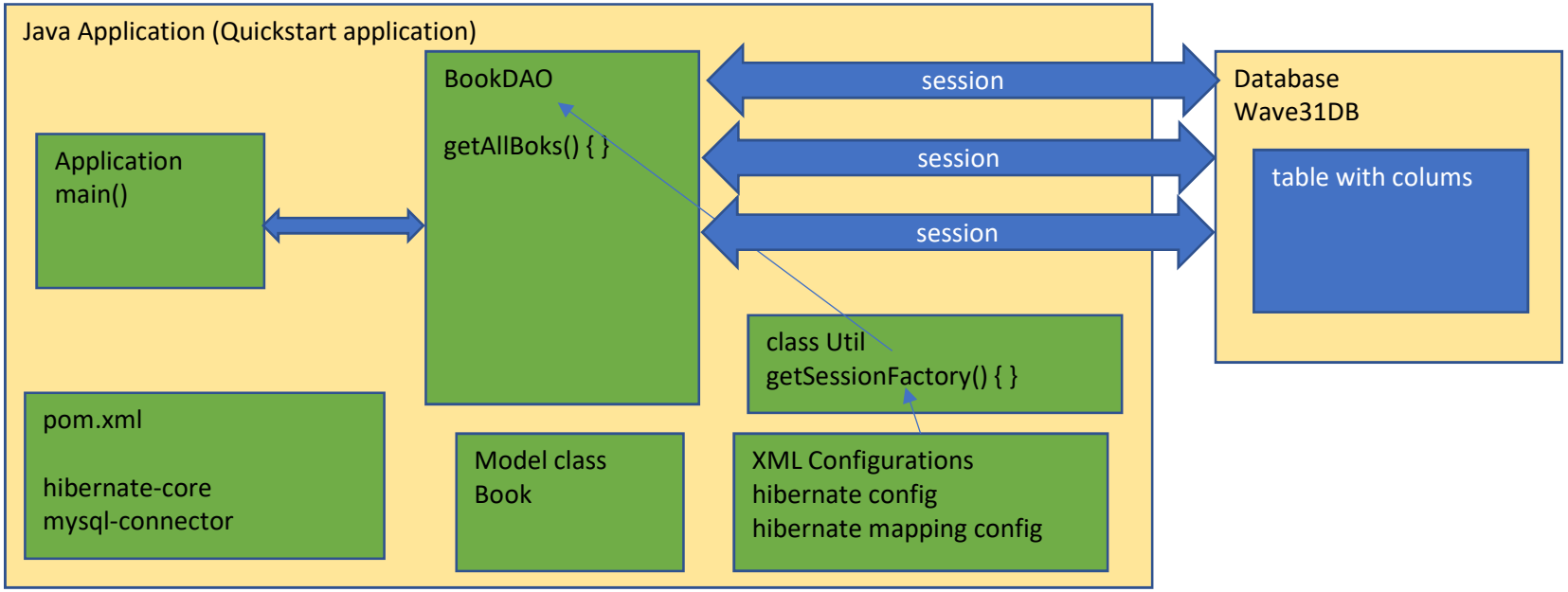
driver

db url, username, password

show queries / dialect

Hibernate mapping configuration

model <-> table

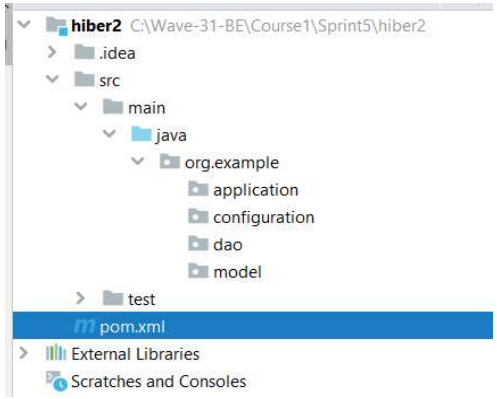


Steps to create new maven application, add hibernate

1 Create new Maven quickstart application

create required packages

application/model/configuration/repository(dao)



2 add required dependencies

hibernate-core

mysql-connector 8.0.30 C:\Program Files (x86)\MySQL\Connector J 8.0

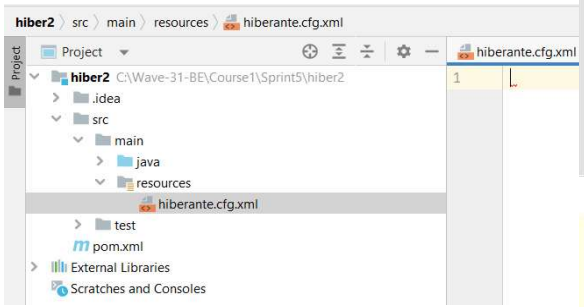
```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.6.12.Final</version>
</dependency>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.30</version>
</dependency>
```

3 Create model class

```java
public class Book {
    4 usages
    private int bkId;
    4 usages
    private String bkName,bkSubject, bkAuthor;
    4 usages
    private int bkPrice, bkStock;
    public Book() {}
    public Book(int bkId, String bkName, String bkSubject, String bkAuthor, int bkPrice, int bkStock) {...}
    public int getBkId() { return bkId; }
    public void setBkId(int bkId) { this.bkId = bkId; }
    public String getBkName() { return bkName; }
    public void setBkName(String bkName) { this.bkName = bkName; }
    public String getBkSubject() { return bkSubject; }
    public void setBkSubject(String bkSubject) { this.bkSubject = bkSubject; }
    public String getBkAuthor() { return bkAuthor; }
    public void setBkAuthor(String bkAuthor) { this.bkAuthor = bkAuthor; }
    public int getBkPrice() { return bkPrice; }
    public void setBkPrice(int bkPrice) { this.bkPrice = bkPrice; }
    public int getBkStock() { return bkStock; }
    public void setBkStock(int bkStock) { this.bkStock = bkStock; }
    @Override
    public String toString() {...}
}
```

4 Create hibernate.cfg.xml under resources folder, define properties for session factory



hiberante.cfg.xml

Who can see what you share here? Recording On

```xml
<?xml version="1.0" ?>

<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/wave31db</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">password</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>
    </session-factory>
</hibernate-configuration>
```

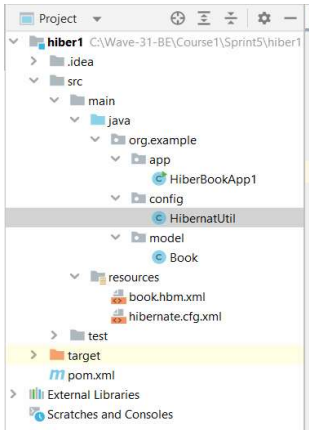
5 create hbm.xml file under resources folder
define model class maping with table

```xml
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="org.example.model.Book" table="Book">
        <id name="bkId">
            <generator class="identity"></generator>
        </id>
        <property name="bkName"></property>
        <property name="bkSubject"></property>
        <property name="bkAuthor"></property>
        <property name="bkPrice"></property>
        <property name="bkStock"></property>
    </class>

</hibernate-mapping>
```

6 Define util class under configuration packege
define method to gernerate and return SessionFactory object
by using xml file configurations



HibernatUtil.java

```java
package org.example.config;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;
2 usages
public class HibernatUtil {
    1 usage
    public static SessionFactory getSessionFactory() {
        SessionFactory sf = null;
        // returns SF object by using xml configuration
        try {
            // load hibernate configuration
            Configuration cfg = new Configuration();
            cfg.configure( resource: "hibernate.cfg.xml");
            cfg.addResource( resourceName: "book.hbm.xml");

            StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder()
            ssrb.applySettings(cfg.getProperties());
            ServiceRegistry registry = ssrb.build();
            sf = cfg.buildSessionFactory(registry);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        return sf;
    }
}
```

7 Make DAO/Repository layer

inject SessionFactory dependency

define getAllBooks() using sessionFactory object

BookDAO.java

```java
package org.example.dao;

import org.example.configuration.HibernateUtil;
import org.example.model.Book;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;

import java.util.List;

public class BookDAO {
    // needs sessionFactory object
    2 usages
    SessionFactory sf=null;
    public BookDAO(){
        sf= HibernateUtil.getSessinFactory();
    }
    // method to get all books
    public List<Book> getAllBooks(){
        // need sessionfactory object to get a session
        Session ses = sf.openSession();
        Query q=ses.createQuery( s: "from Book"); // makes query as select * from Book
        List<Book> books=q.list(); // executes 'select * from Book' in DB, returns List<MODEL>
        ses.close();
        return books;
    }
```

8 Define application class

define main()

create DAO object

Make sure hiberate is creating table in db as per model class

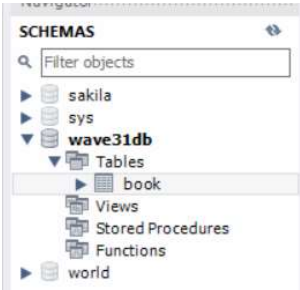
HibernateBookMain1.java

```java
package org.example.application;

import org.example.dao.BookDAO;

public class HibernateBookMain1 {
    public static void main(String[] args) {
        BookDAO bookDao = new BookDAO();
        // bookDao -> sessionFactory -> loads configuration from xml files -> creates table in db
    }
}
```

```
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentIn
Hibernate: create table Book (bkId integer not null auto_increment, bkName varchar(255), bkSubject varchar(255), bkAuthor
Oct 06, 2022 5:38:44 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
```



insert few records

```
insert into book (bkname, bksubject,bkauthor,bkprice, bkstock) values
('Let us C','C','BGS',123,34),
('Tags in HTML','HTML','BGS',223,14),
('OSI layers','Networking','McG',324,51);

select * from book;
```

9 in main()
call dao.getAllBooks()

```
public class HibernateBookMain1 {
    public static void main(String[] args) {
        BookDAO bookDao = new BookDAO();
        // bookDao -> sessionFactory -> loads configuration from xml files -> creates table in db
        List<Book> data = bookDao.getAllBooks();
        //System.out.println(data);
        for(Book b:data){
            System.out.println(b);
        }
    }
}
```

```
Hibernate: select book0_.bkId as bkid1_0_, book0_.bkName as bkname2_0_, book0_.bkSubject as bksubjec3_0_,
Book{bkId=1, bkName='Let us C', bkSubject='C', bkAuthor='BGS', bkPrice=123, bkStock=34}
Book{bkId=2, bkName='Tags in HTML', bkSubject='HTML', bkAuthor='BGS', bkPrice=223, bkStock=14}
Book{bkId=3, bkName='OSI layers', bkSubject='Networking', bkAuthor='McG', bkPrice=324, bkStock=51}
```