Подключение зависимостей:
Сам -hibernate
Сама - база данных, с которой hibernate будет работать(PostgreSQL)

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.3.Final</version>
</dependency>

<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.2.5</version>
</dependency>
```

Примитивный mapping класса для работы с hibernate

```
@Entity
@Table(name = "person")
public class Person
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "person_id")
    private Long personId;
    @Column(name = "first_name")
    private String firstName;
    @Column(name = "last_name")
    private String lastName;

    public Long getPersonId() {
        return personId;
    }

    public void setPersonId(Long personId) {
        this.personId = personId;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return "Person{" +
```

```
                "personId=" + personId +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                '}';
    }
}
```

В папке resources необходимо добавить конфигурационный файл hibernate.cfg.xml

```
     <?xml version='1.0' encoding='utf-8'?>
        <!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.dialect">org.hibernate.dialect.PostgreSQL10Dialect</property>
<!--   можно вроде как и не указывать диалект и сам хибернэйт найдет нужный
диалект по jdbs драйвер, но лучше указывать-->
   <property
name="hibernate.connection.driver_class">org.postgresql.Driver</property>
   <!--jdbs драйвер-->
   <property
name="hibernate.connection.url">jdbc:postgresql://localhost/register_office</
property>
   <property name="hibernate.connection.username">postgres</property>
   <property name="hibernate.connection.password">postgres</property>
   <property name="hibernate.show_sql">true</property>

   <mapping class="edu.javacourse.register.domain.Person" />
<!--с каким классом работать-->
</session-factory>

</hibernate-configuration>
```

Теперь о том, как воспользоваться - конфигурационным файлом
- возможностями хибернэйт

Сериализация- преставление объекта в поток байт для передачи
Десериализация- восстановление байт в объект

Транзакции стоит использовать только если ты что-то меняешь в бд(INSERT, UPDATE), если ты хочешь читать , то ненадо

```
public class PersonManager
{
    public static void main(String[] args) {

        SessionFactory sf = buildSessionFactory();

        System.out.println();
        System.out.println();
        System.out.println();

        Session session = sf.openSession();

        session.getTransaction().begin();

        Person p = new Person();
        p.setFirstName("Василий");
        p.setLastName("Сидоров");

        Long id = (Long) session.save(p); // hibernate будет считать что БД
сама первичный ключ сгенерирует метод save(возвращает Serializable id)
выполняет INSERT, то есть если уже в БД поле с таким же id есть, то будет
ошибка, если хотите перезаписать UPDATE, то используй метод saveOrUpdate(не
возвращает id) он либо инсерт сделает, если нету id, либо перезапись
        System.out.println(id);

        session.getTransaction().commit();
        session.close();

        session = sf.openSession();
        Person person = session.get(Person.class, id);//транзакция ненужна ,
так как режим чтения и выполняется SELECT запрос
        System.out.println(person);
        session.close();


        session = sf.openSession();
                            //создание объекта Query на HQL + указание
подсказки типа возвращаемых объектов из БД(Person.class)
        List<Person> list = session.createQuery("FROM Person",
Person.class).list();
        list.forEach(p1 -> System.out.println(p1));

        session.close();
    }
```

```
 //SessionFactory - это типо аналог pool connections , дает то, что позволяет
с БД общаться
    private static SessionFactory buildSessionFactory() {
        try {
            //   StandardServiceRegistry- специальный сервис, который
регистрируется в системе hibernate и через который, hibernate всем этим
управляет, в старых версиях использовался класс configure
            StandardServiceRegistry serviceRegistry = new
StandardServiceRegistryBuilder()
                    .configure("hibernate.cfg.xml").build();

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

            return metadata.getSessionFactoryBuilder().build();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory creation failed." +
ex);
            throw new ExceptionInInitializerError(ex);
        }
    }


}
```

Обрати внимание на метод buildSessionFactory он возвращает SessionFactory для работы с сессиями и сессии выполняют методы, которые генерируют запросы к БД от hibernate