

Hibernate Setup without jboss eclipse plugin – simple way to execute hibernate programs

Step 1:

We need to have latest eclipse to get all features. If you have old eclipse you need to download latest one so that we can utilize all latest features.

To install eclipse, follow this document, if you already have latest eclipse skip this step.

<https://javabykiran.in/core-java/JBKSETUP001-java-eclipse-setup.pdf>

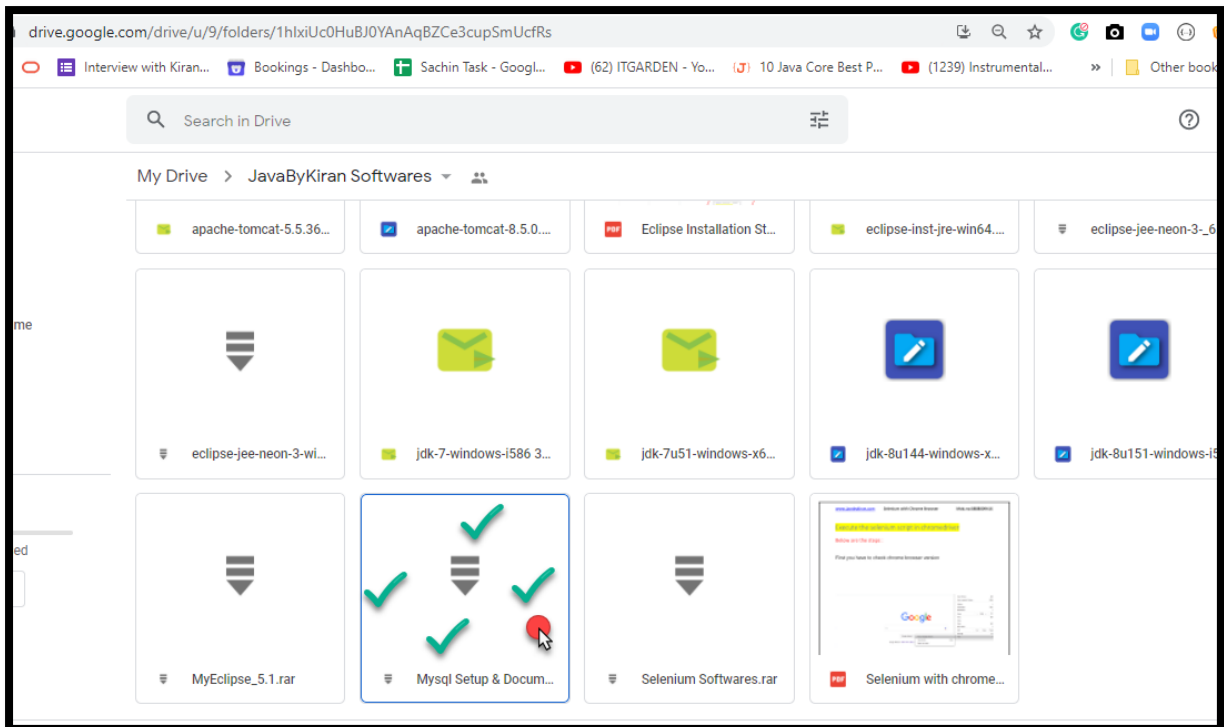
Step 2:

Make sure you have MySQL installed and MySQL connector jar as well.

If you already have MySQL skip this step.

To download MySQL database and MySQL connector follow below documents.

- Downloading of MySQL software can be done from below link.
<https://drive.google.com/drive/u/3/folders/1hIxiUc0HuBJ0YAnAqBZCe3cupSmUcfRs>



- Refer below document for MySQL installation and query browser installation.
<https://javabykiran.in/core-java/MySQL%20and%20Query%20Browser%20installtion.pdf>
- MySQL Connector will be needed for connecting java with database.
<https://drive.google.com/drive/folders/1qV3vR9WWrmkRb8Ush6RYuLEjr3hB51-q?usp=sharing>

Step 3:

Create database and one table for database operations.

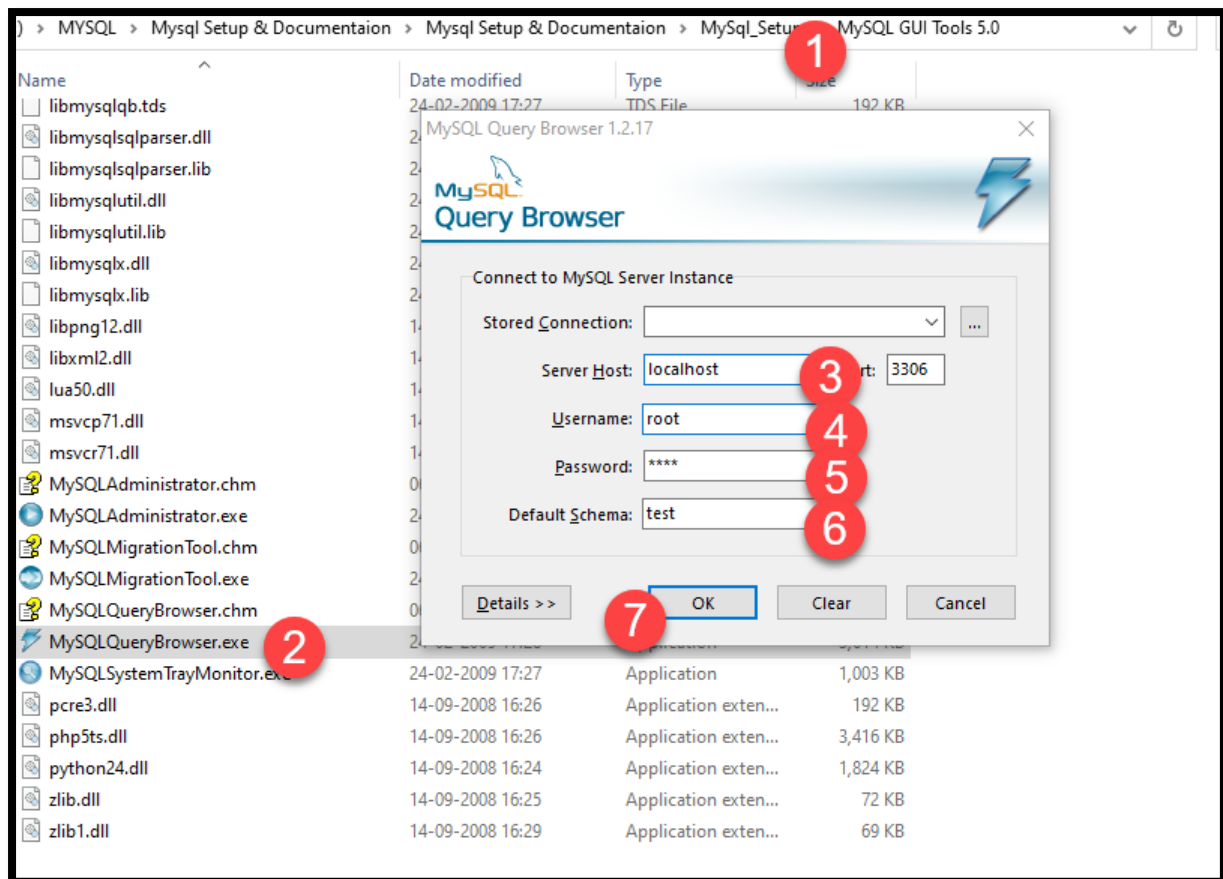
Schema: **studentmanagement**

Table: **student**

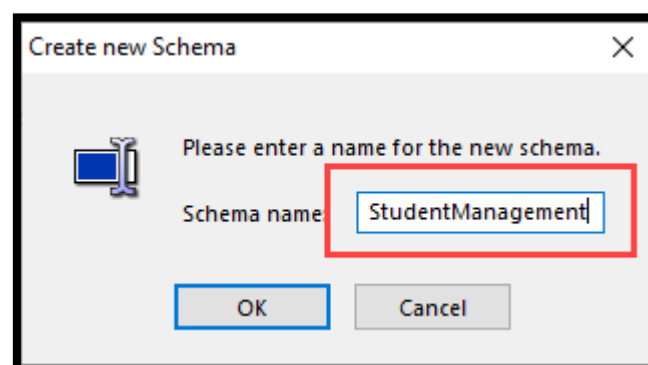
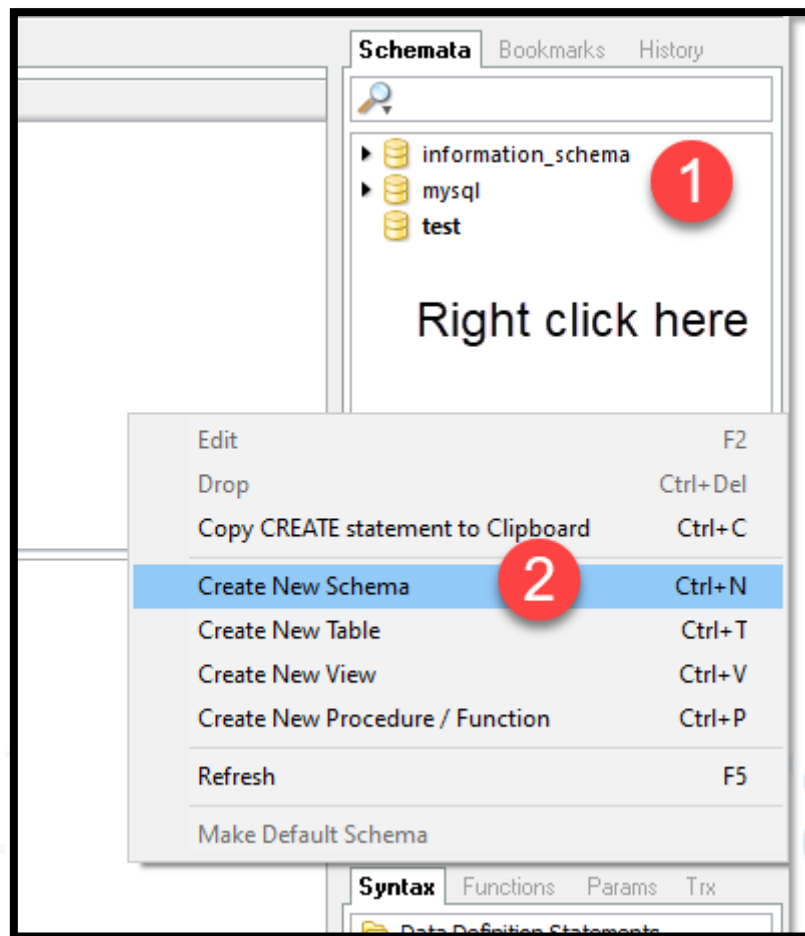
Columns: **sid, sname, sphone**

Open Query Browser from above installer as per steps mentioned in

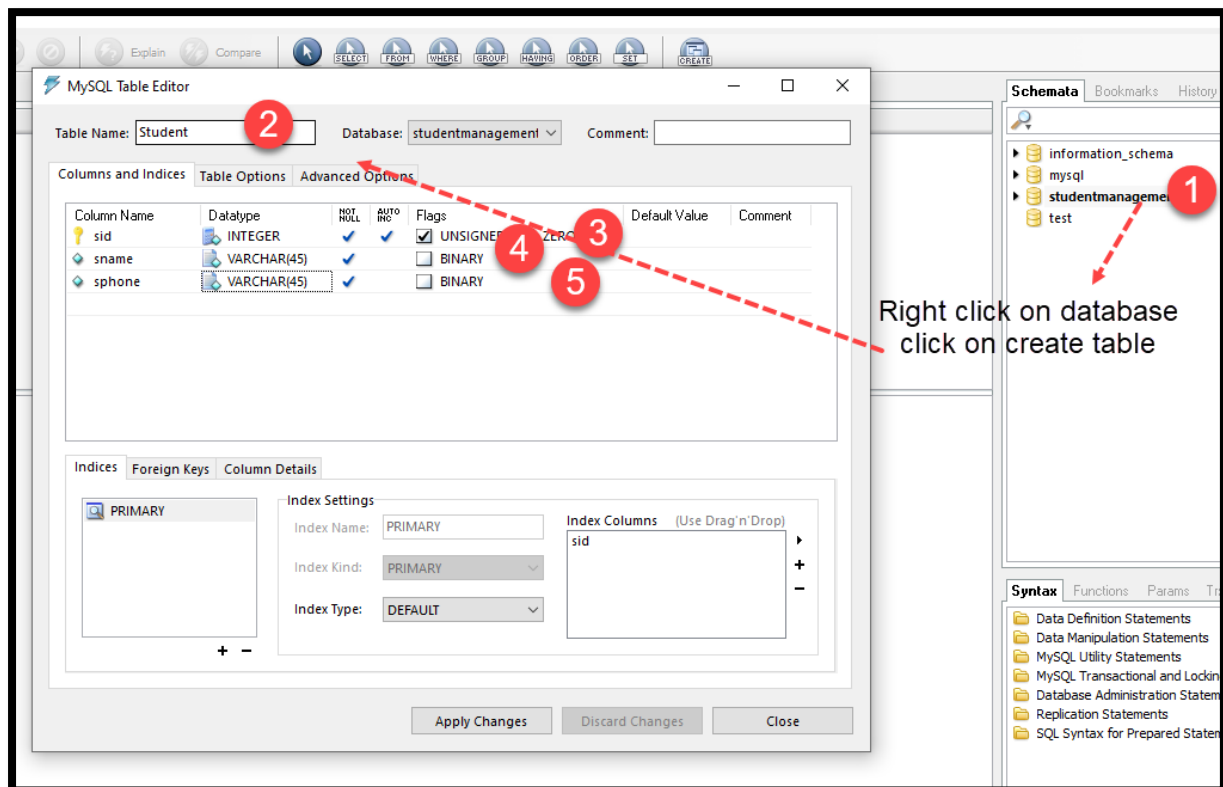
<https://javabykiran.in/core-java/MySQL%20and%20Query%20Browser%20installtion.pdf>.



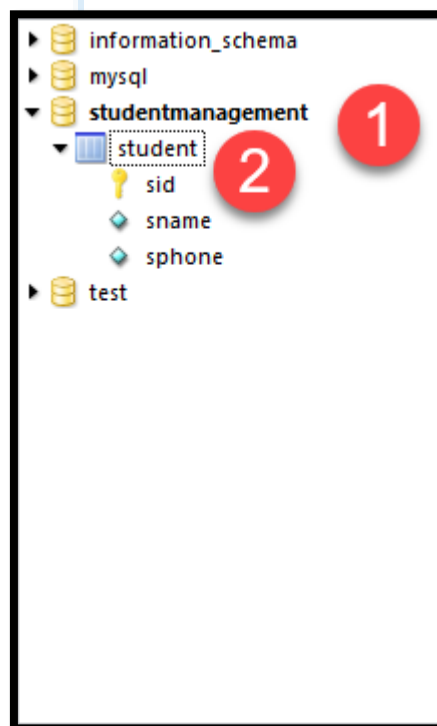
Create database **StudentManagement**



Create table **student**.



Schema will look like below.

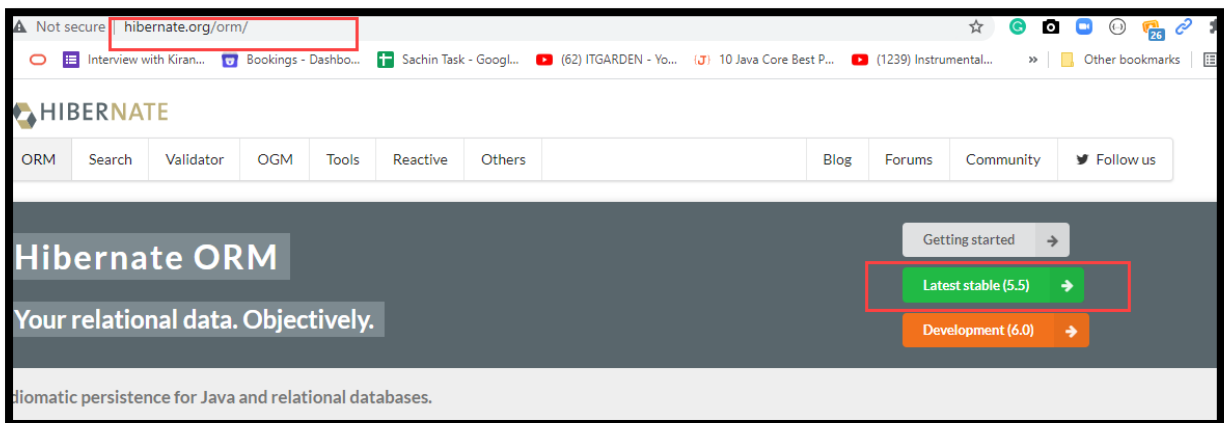


Step 4:

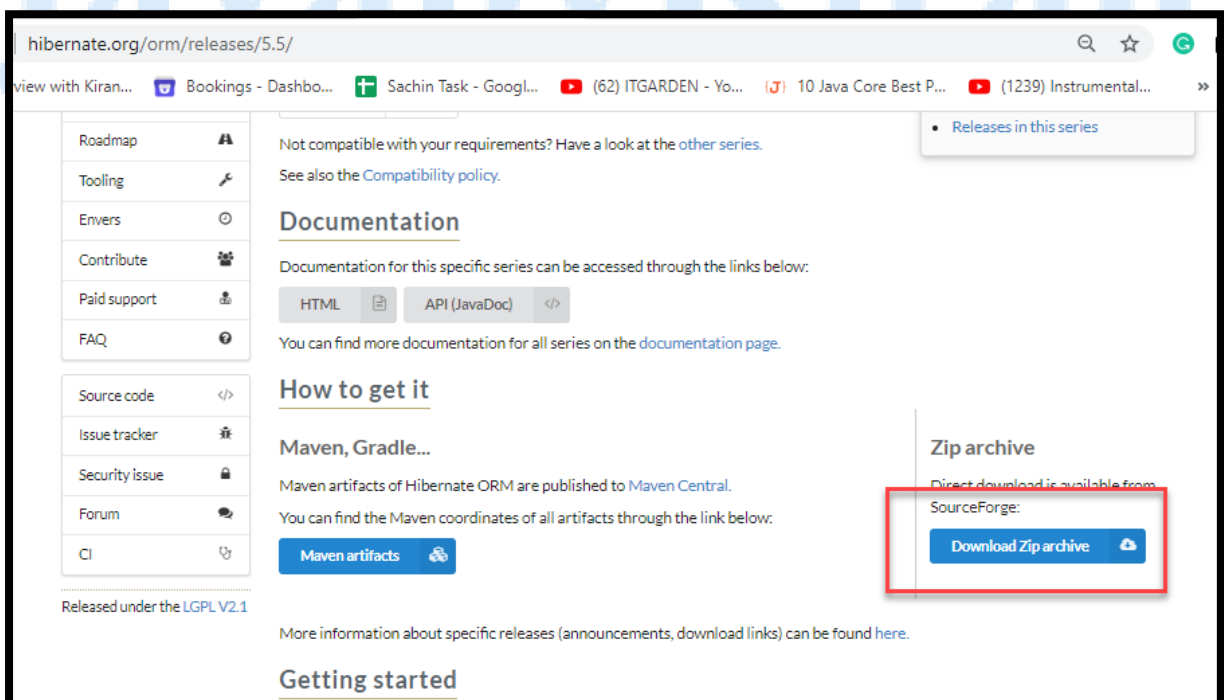
We need to download latest hibernate jars in our system.

Below are steps.

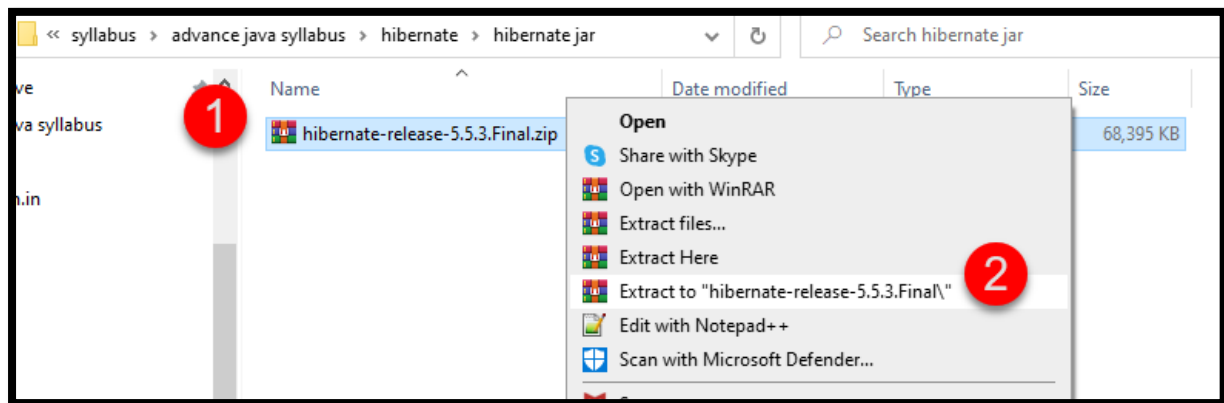
1. Go to URL <http://hibernate.org/orm/> Click on Latest Stable 5.5



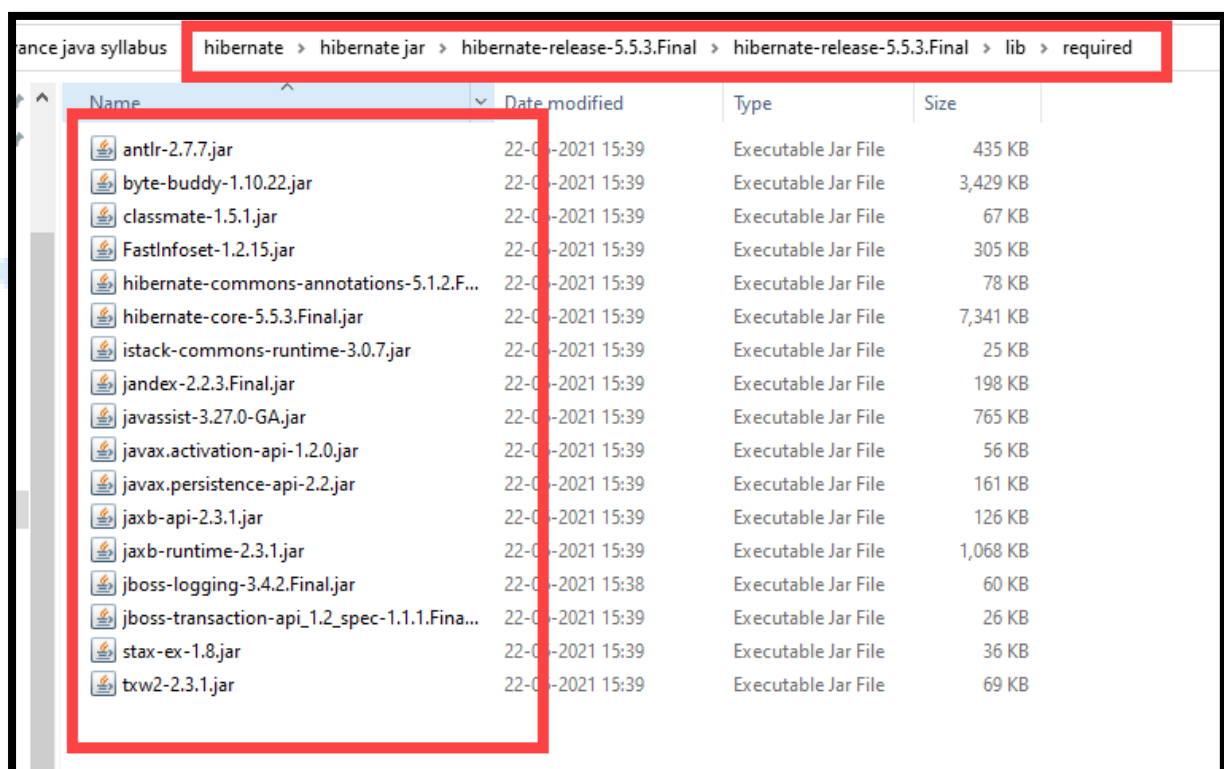
2. Download as below.



3. Extract downloaded folder.



4. Go to required folder as below and check if you have all libraries required for hibernate. We need to add these jars in future steps.



Now we have almost all prerequisite

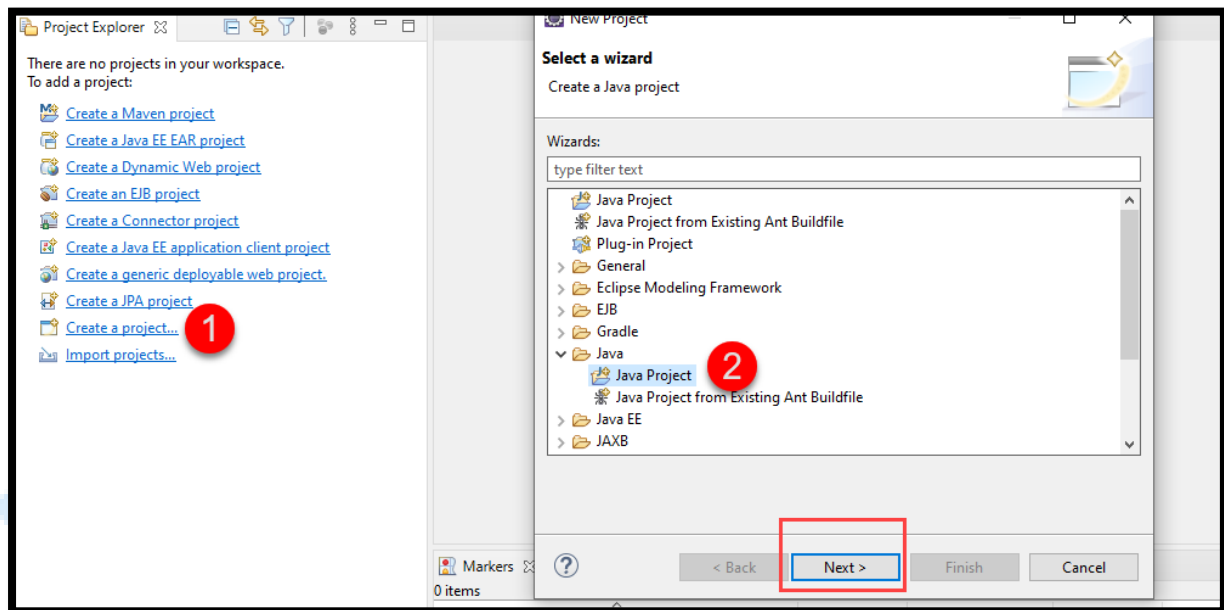
- Latest eclipse installed. -- **OPTIONAL**
- MySQL installed.
- MySQL connector jar file
- MySQL schema and table is ready.
- Hibernate required jars.
- Hibernate plugin in eclipse for configuration and reverse engineering. – **NOT NEEDED**

javabyKiran
java | selenium | python

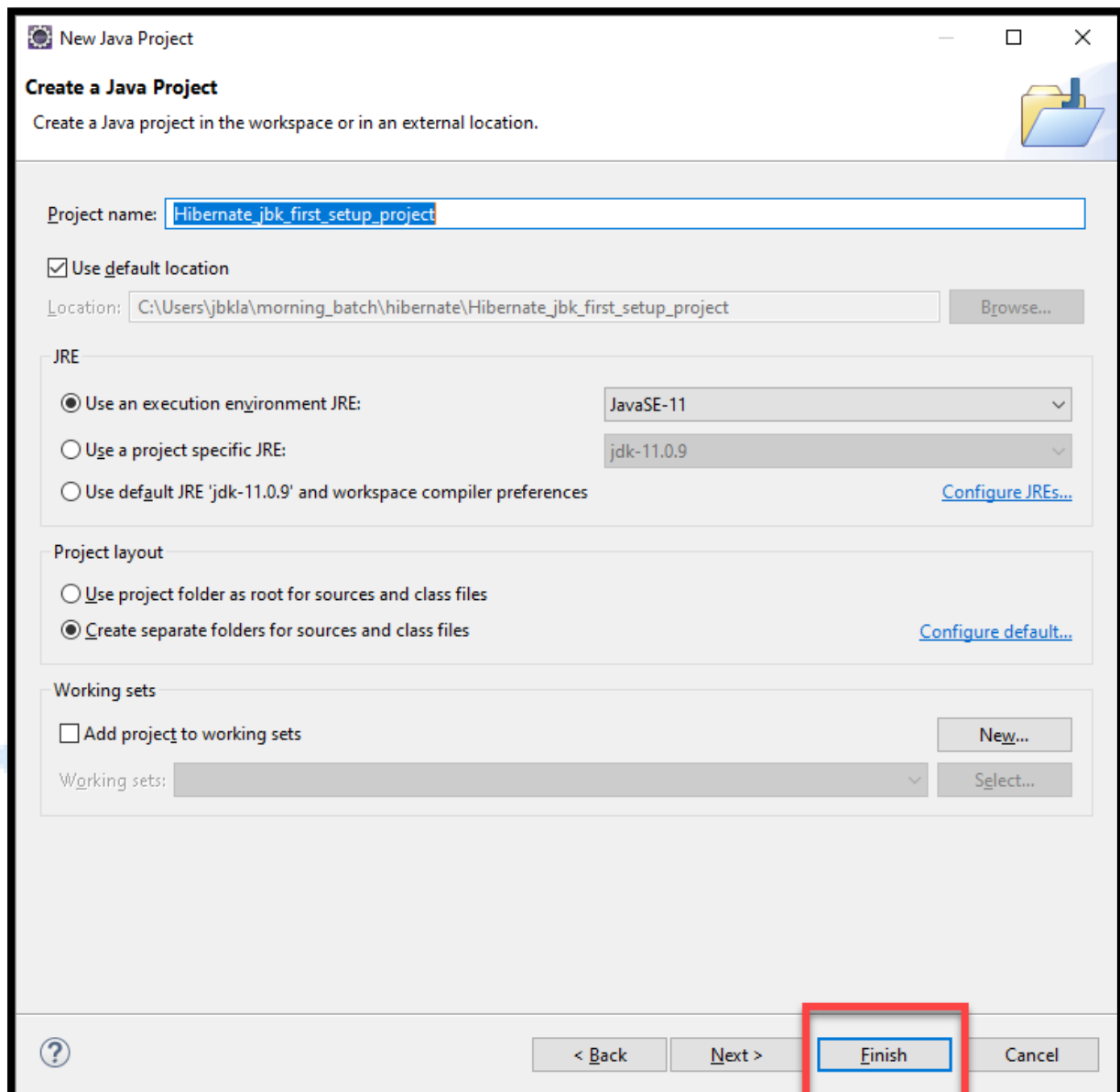
It is time to start eclipse and start project.

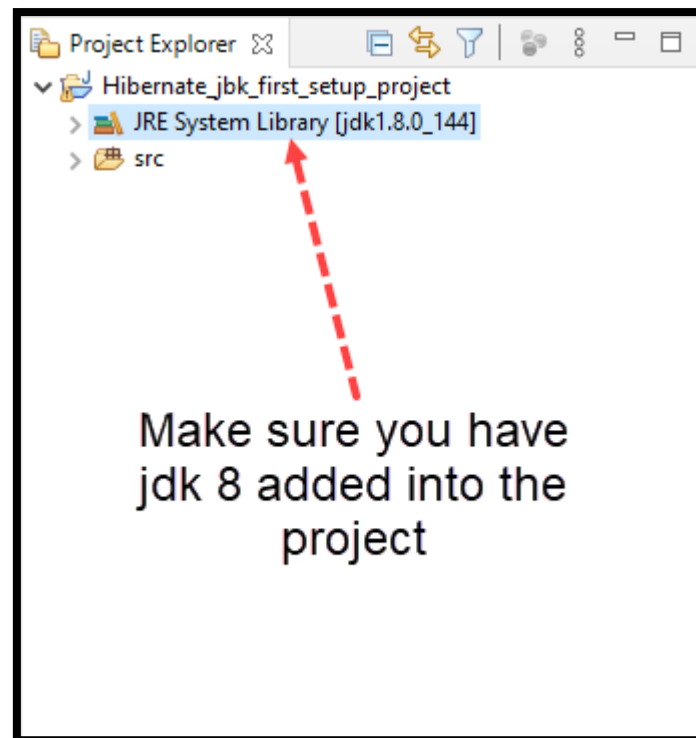
Step 5:

1. Create simple java project.
2. Add all jars in build path.



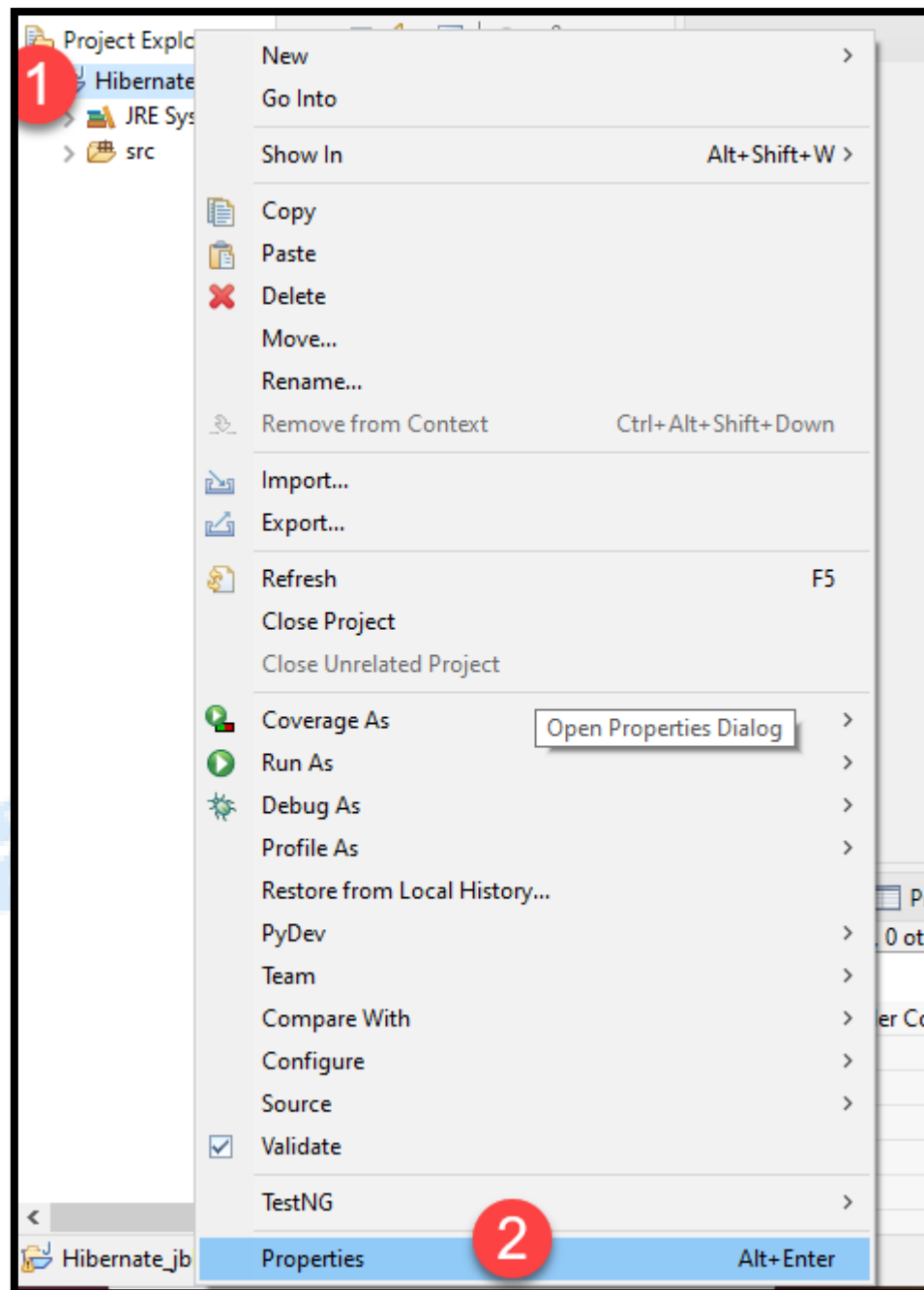
Create a project with name Hibernate_jbk_first_setup_project



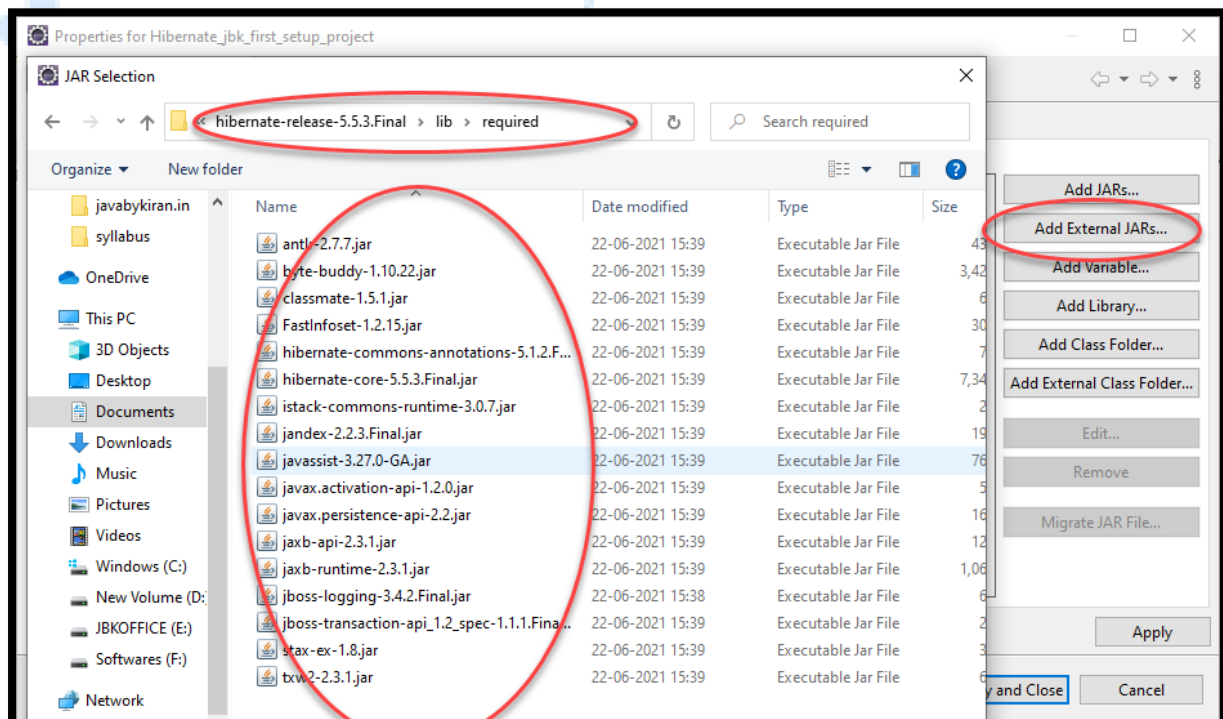
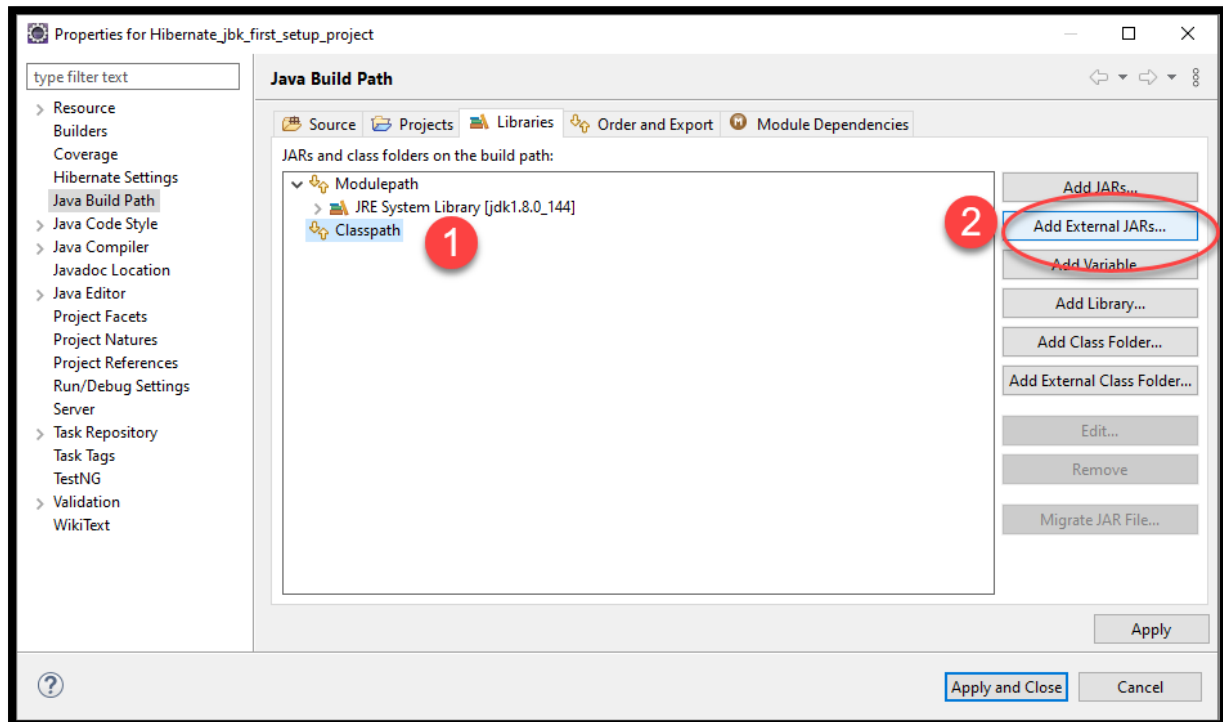


javabyKiran

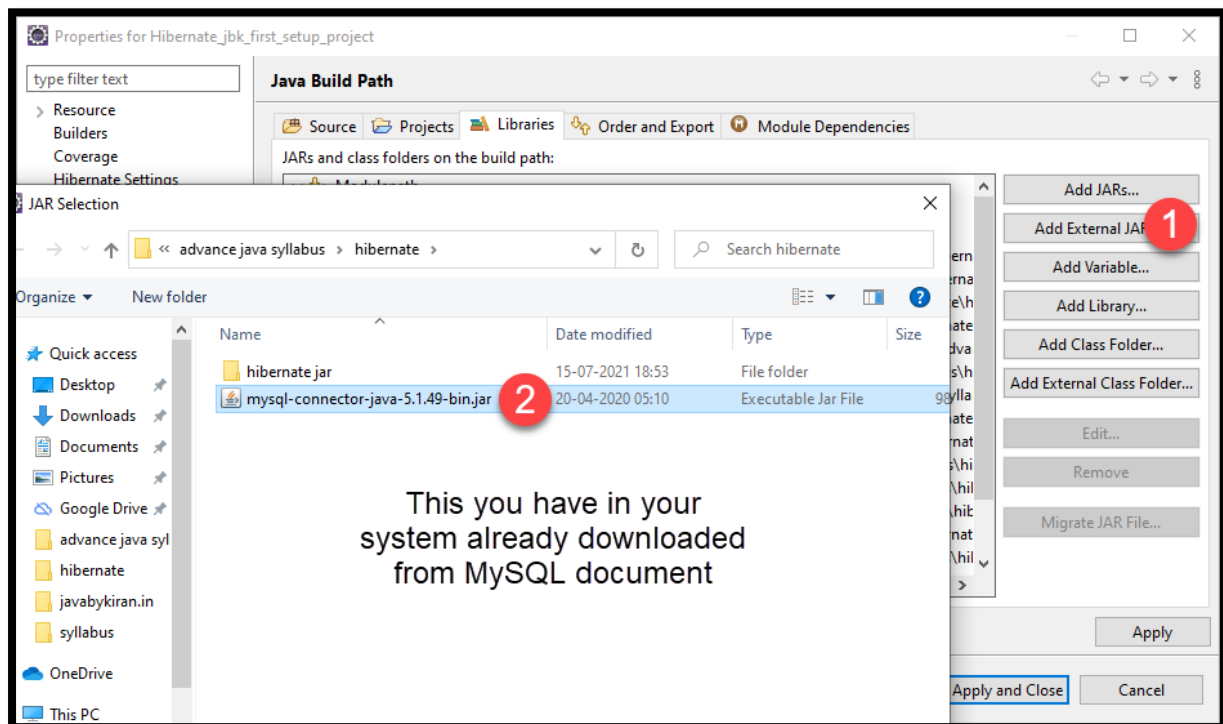
java | selenium | python



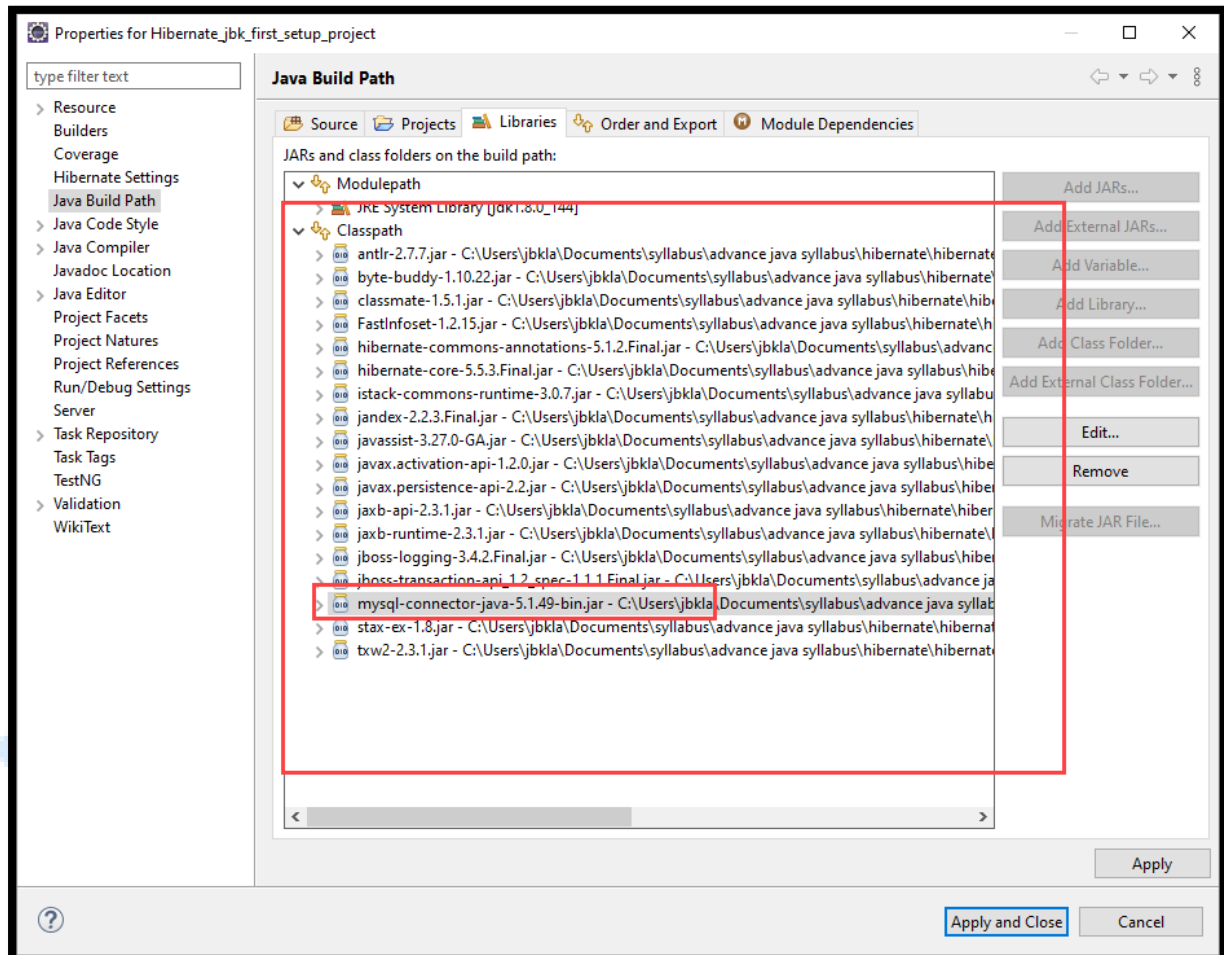
Add hibernate jars and MySQL connector jar in build path as below



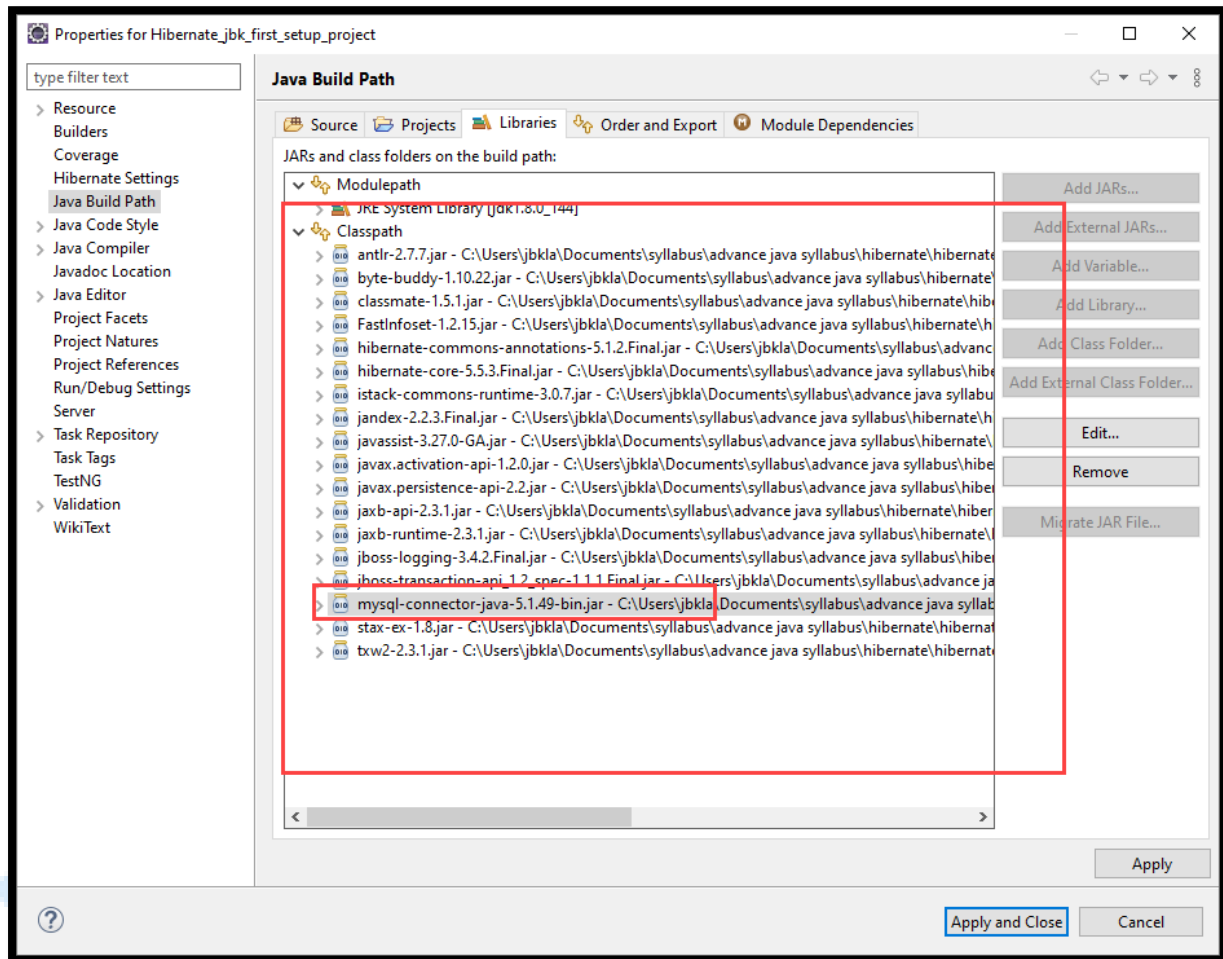
Now MySQL connector jar we need to add in build path.



This is what you see in libraries after above steps. click apply and close.



Now our project looks like below.



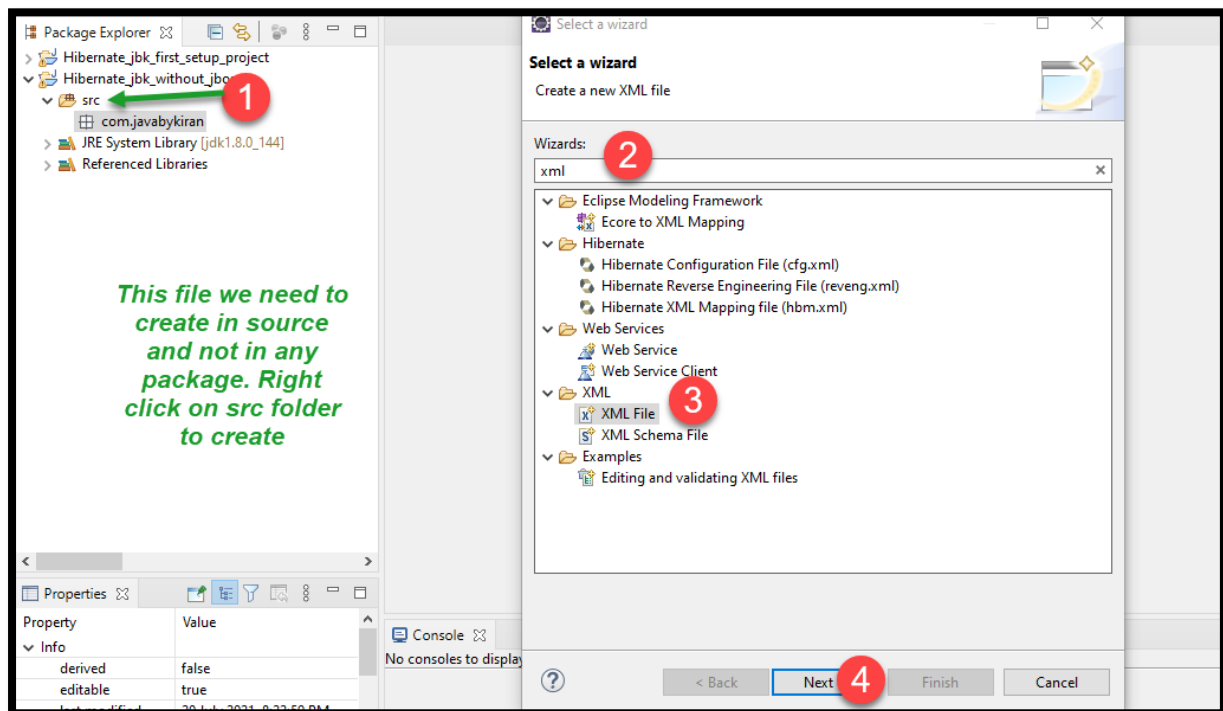
Step 6:

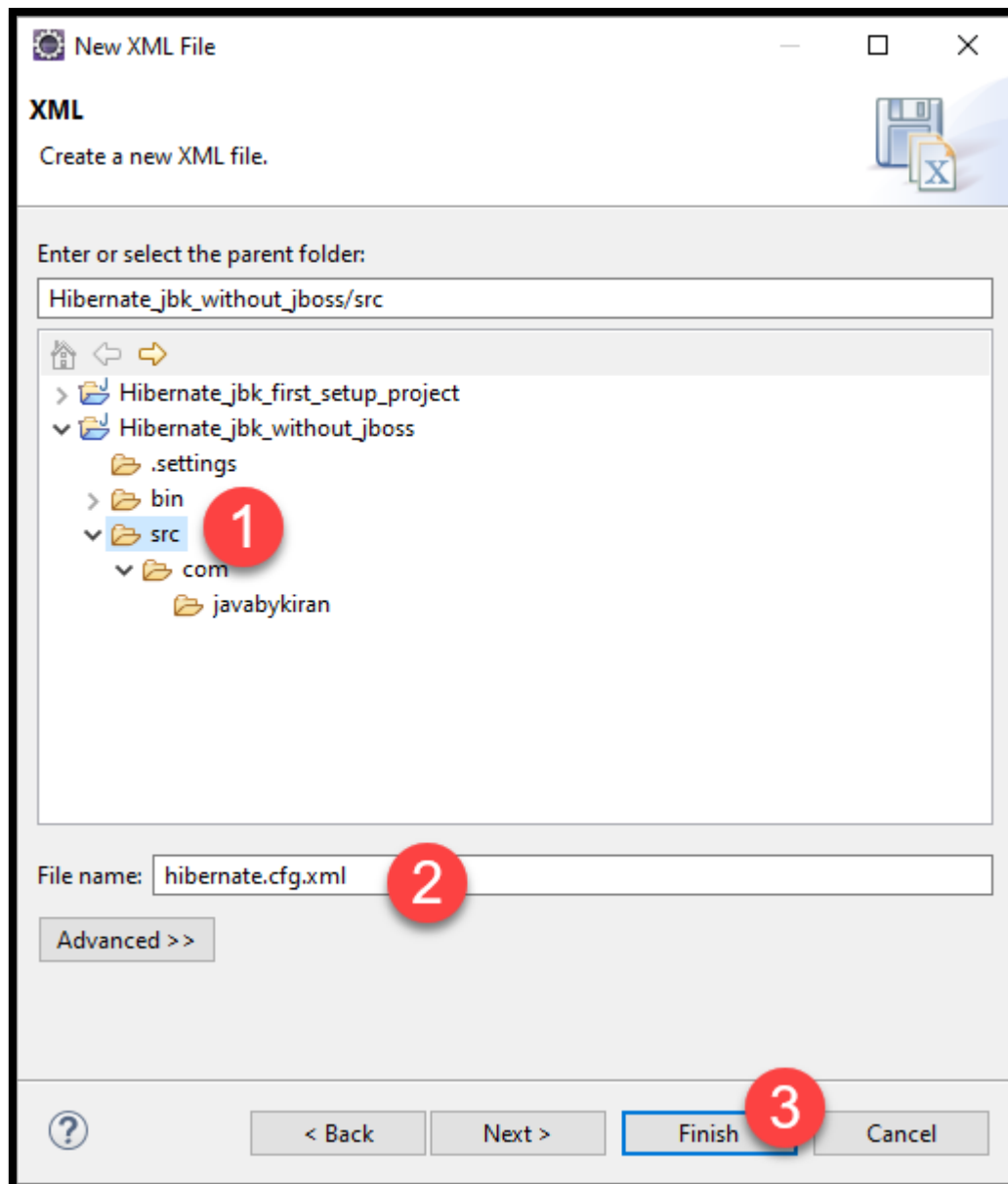
Just remember to run any hibernate program we need following.

- Hibernate.cfg.xml
 - In this file we will be having all hibernate configurations.
 - driver class – same as jdbc
 - connection URL – same as jdbc
 - username – same as jdbc
 - password – same as jdbc
 - Dialect – MySQLDialect
 - This is used for generating queries on the basis of selection.
- Entity Class
 - This class will use annotations
 - This class is mapped with database.
 - Every variable will be mapped with some column name by using @Column annotation
 - Class is mapped with table by using @Entity annotation
 - Primary key is mapped with variable in java
 - @Id
 - It represent it is a primary key
 - @GeneratedValue
 - This is used for auto generation algorithm
- Client code
 - Main method and we need to create following objects.
 - Sessionfactory object
 - Session object

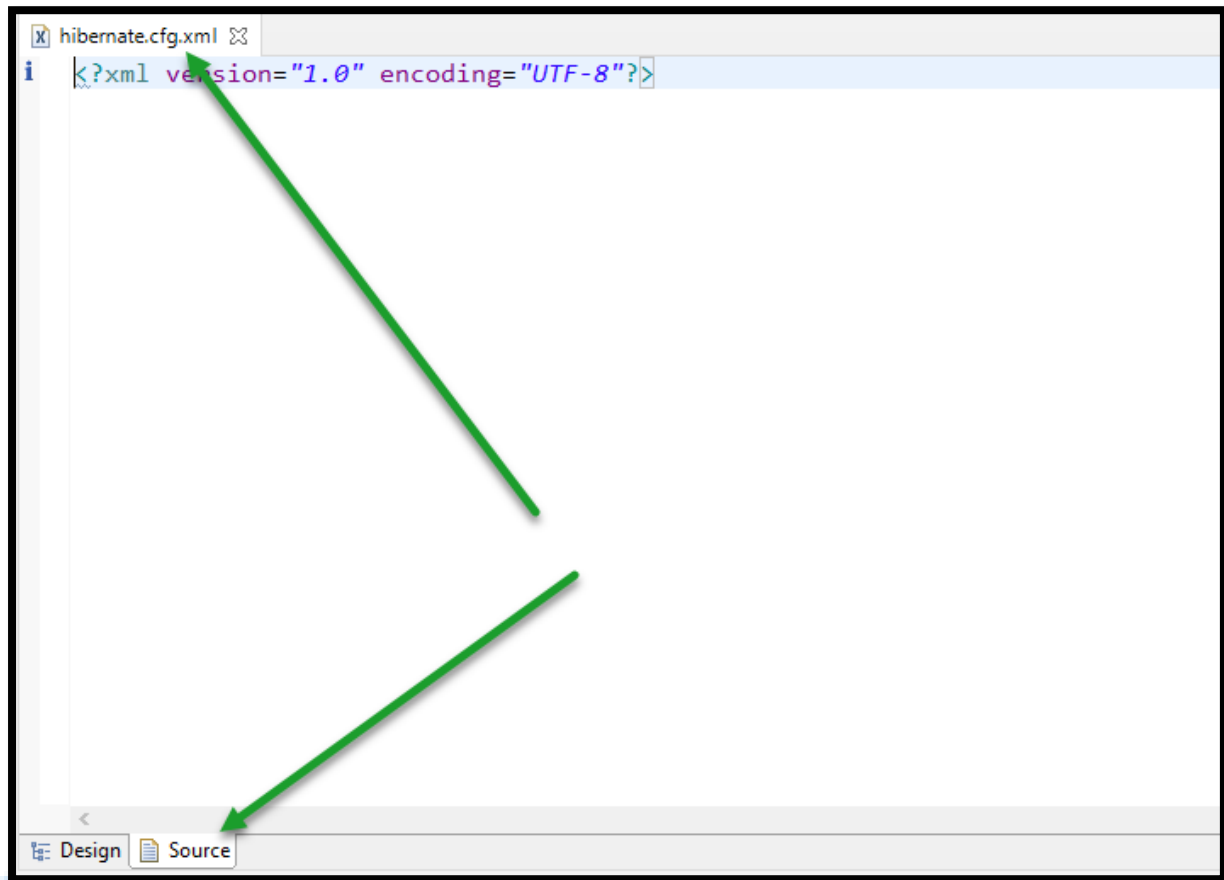
Step#6.1

Write hibernate.cfg.xml





Open Source

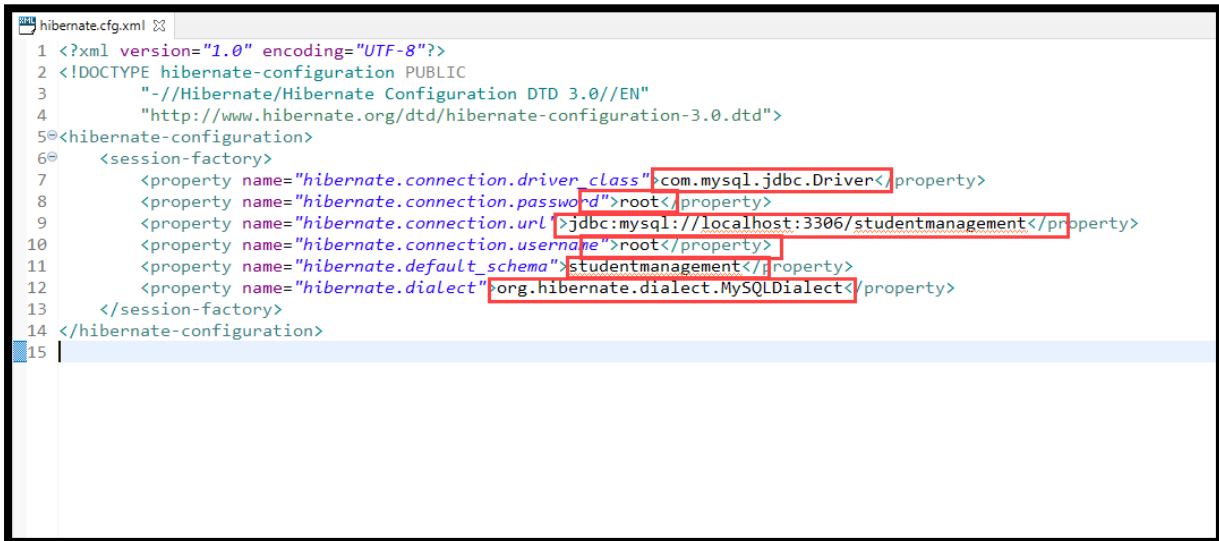


java | selenium | python

Now add below code in this file. It never changes only db connections you need to change in industry. Always copy and paste from existing projects and change on db. details. This is only one file for complete project.

```
<?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.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.password">root</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/studentmanagement</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.default_schema">studentmanagement</property>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    </session-factory>
</hibernate-configuration>
```

Observe below.



```
hibernate.cfg.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3     "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4     "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5 <hibernate-configuration>
6     <session-factory>
7         <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
8         <property name="hibernate.connection.password">root</property>
9         <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/studentmanagement</property>
10        <property name="hibernate.connection.username">root</property>
11        <property name="hibernate.default_schema">studentmanagement</property>
12        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
13    </session-factory>
14 </hibernate-configuration>
15
```

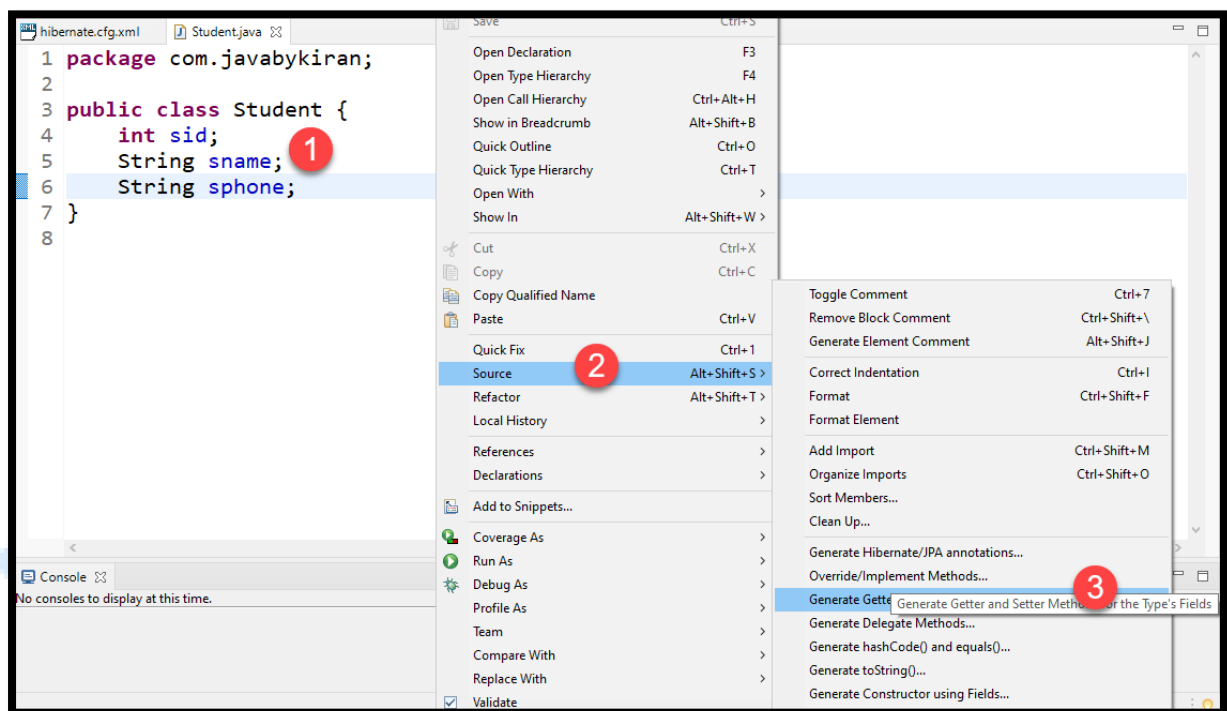
javabyKiran

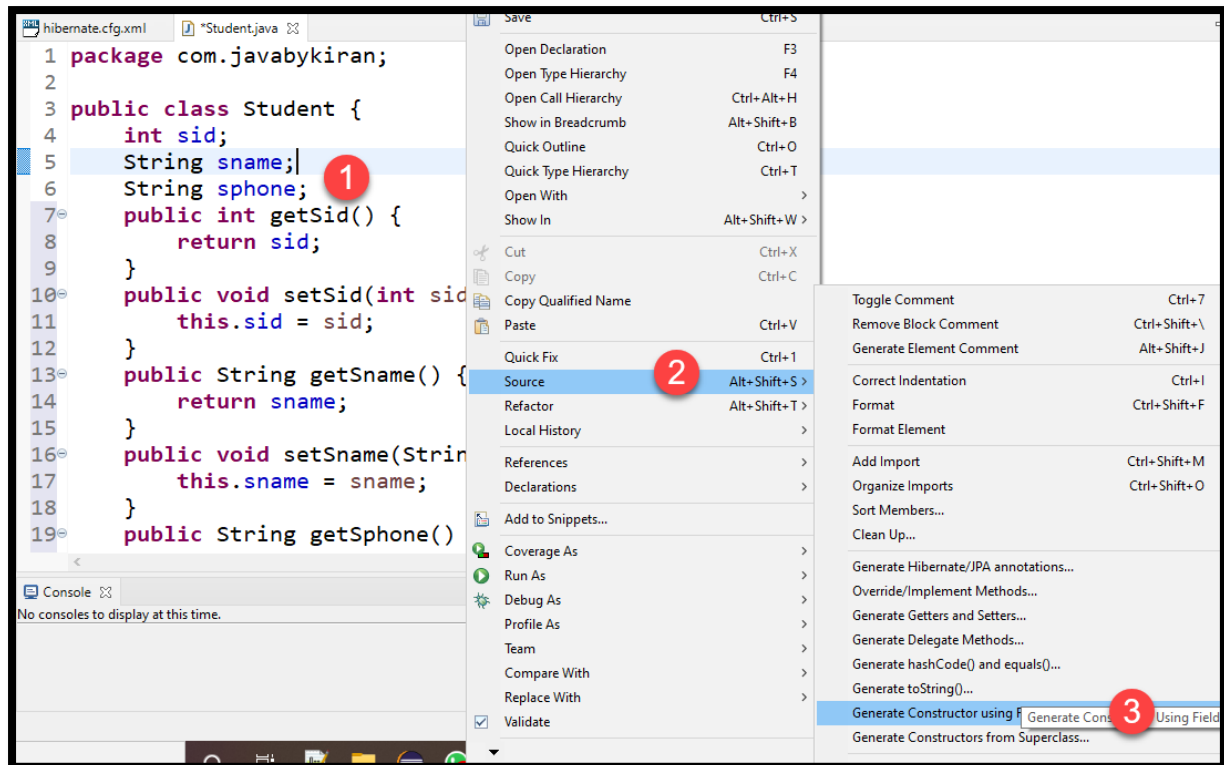
java | selenium | python

Step#6.2

Now create Entity class.

- Declare variables
- Generate getters and setters
- Generate constructors with fields
- Generate constructor without fields





Student.java

```

package com.javabykiran;

public class Student {
    int sid;
    String sname;

    public Student(int sid, String sname, String sphone) {
        super();
        this.sid = sid;
        this.sname = sname;
        this.sphone = sphone;
    }

    public Student() {
        super();
    }

    String sphone;

    public int getSid() {
        return sid;
    }

```

```
public void setSid(int sid) {
    this.sid = sid;
}

public String getSname() {
    return sname;
}

public void setSname(String sname) {
    this.sname = sname;
}

public String getSphone() {
    return sphone;
}

public void setSphone(String sphone) {
    this.sphone = sphone;
}
}
```

Step#6.3

Now start adding annotations as below

```
package com.javabykiran;

@Entity
@Table(name = "student")
public class Student implements java.io.Serializable {

    private Integer sid;
    private String sname;
    private String sphone;

    public Student() {
    }

    public Student(String sname, String sphone) {
        this.sname = sname;
    }
}
```



```
        this.sphone = sphone;
    }

    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "sid", unique = true, nullable = false)
    public Integer getSid() {
        return this.sid;
    }
    public void setSid(Integer sid) {
        this.sid = sid;
    }
    @Column(name = "sname", nullable = false, length = 45)
    public String getSname() {
        return this.sname;
    }
    public void setSname(String sname) {
        this.sname = sname;
    }
    @Column(name = "sphone", nullable = false, length = 45)
    public String getSphone() {
        return this.sphone;
    }
    public void setSphone(String sphone) {
        this.sphone = sphone;
    }
}
```

Everything over now with respect to configuration, it is time to write client code.

Till now whatever we done it is one time job, we do not do it again and again. All tables entity classes will be written at once. Only we need to play with client class in future.

Step 7:

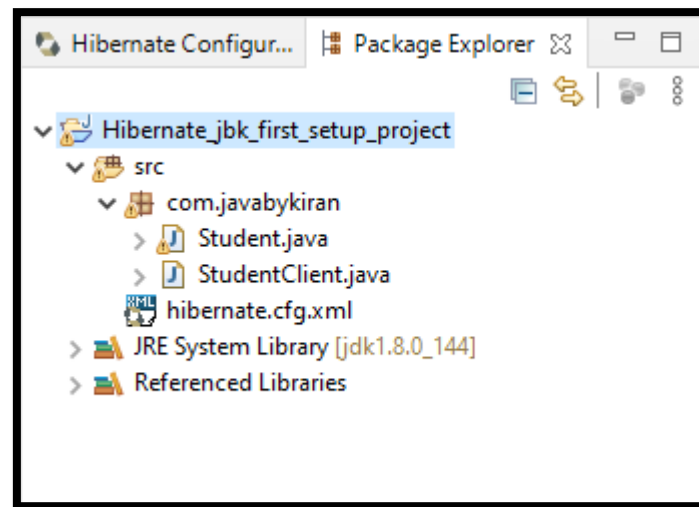
Write studentClient.java.

```
package com.javabykiran;

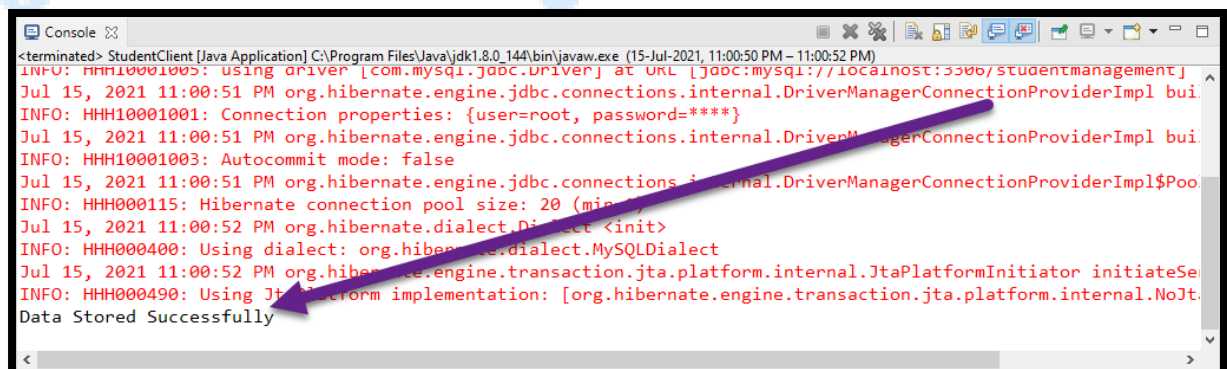
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

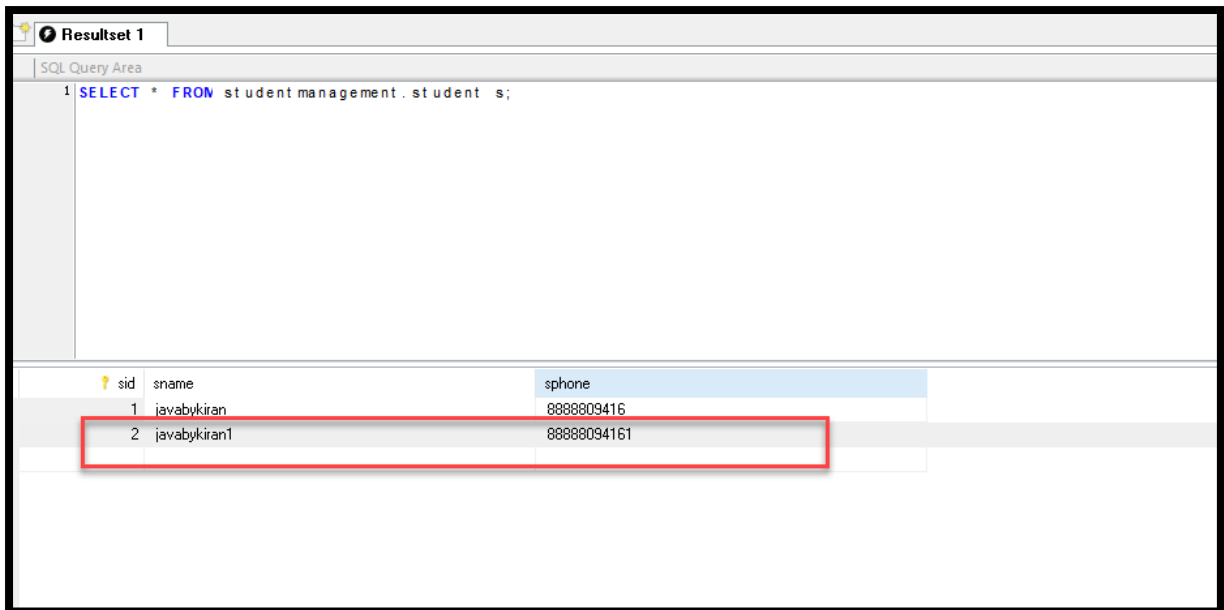
public class StudentClient {
    public static void main(String[] args) {
        Configuration cfg = new Configuration();
        cfg.configure().addAnnotatedClass(Student.class);
        SessionFactory sf = cfg.buildSessionFactory();
        Session session = sf.openSession();
        Transaction tt = session.beginTransaction();
        Student stu = new Student("javabykiran", "8888809416");
        session.save(stu);
        tt.commit();
        session.close();
        System.out.println("Data Stored Successfully");
    }
}
```

Project structure will look like



It is time to check if data is inserted into DB.





The screenshot shows a database query result in a tool. At the top, it says "ResultSet 1". Below that, the "SQL Query Area" contains the query: `1 SELECT * FROM studentmanagement.student s;`. The result is displayed as a table with three columns: `sid`, `sname`, and `sphone`. The first row has values 1, javabykiran, and 8888809416. The second row has values 2, javabykiran1, and 88888094161, and this row is highlighted with a red border.

sid	sname	sphone
1	javabykiran	8888809416
2	javabykiran1	88888094161

javabyKiran

java | selenium | python

Troubleshooting if needed.

- Make sure you have proper compiler in place that is 1.8.

