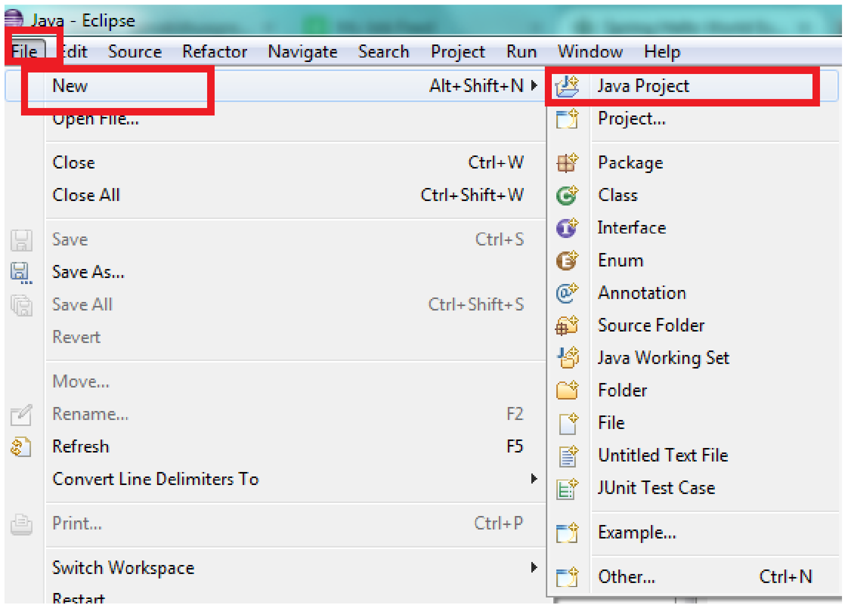
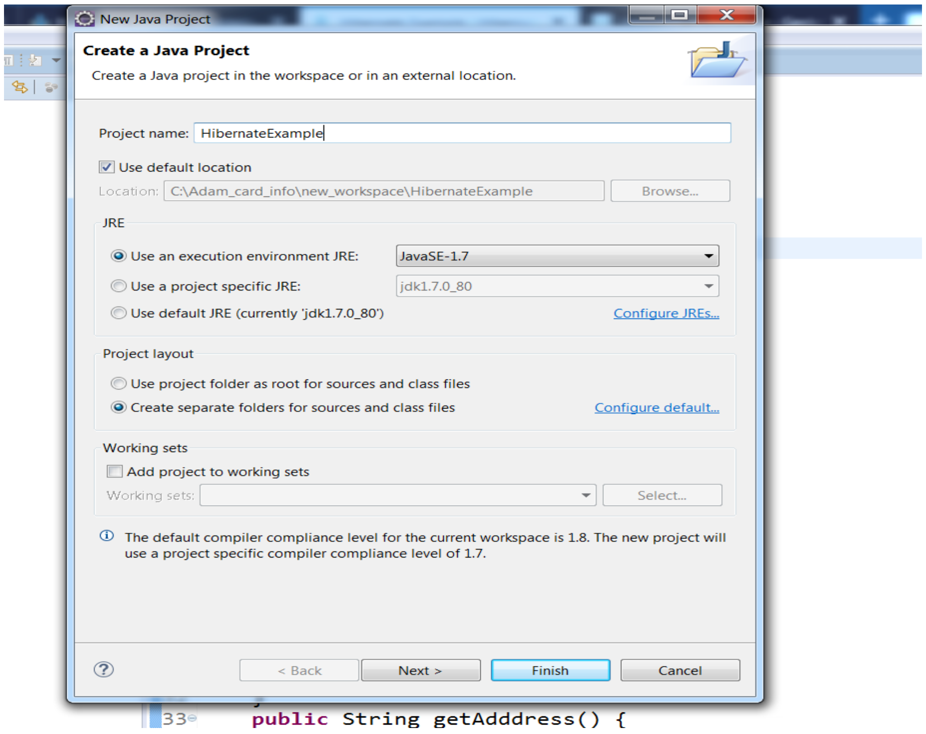
Create a Java project in Eclipse

To create a Java project in eclipse, Go to File (New Java Project (as shown below).



Enter the name of Java project and click Finish. In this chapter, we have created a project name as “HibernateExample” (see below figure).



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: HibernateExample

[✓] Use default location

Location: C:\Adam_card_info\new_workspace\HibernateExample Browse...

JRE

(●) Use an execution environment JRE: JavaSE-1.7

( ) Use a project specific JRE: jdk1.7.0_80

( ) Use default JRE (currently 'jdk1.7.0_80') Configure JREs...

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files Configure default...

Working sets

[ ] Add project to working sets

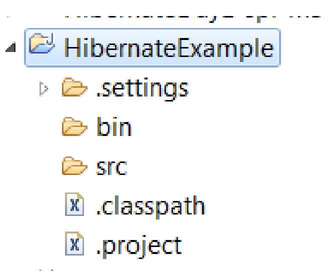
Working sets: Select...

The default compiler compliance level for the current workspace is 1.8. The new project will use a project specific compiler compliance level of 1.7.

< Back Next > Finish Cancel

```
public String getAdddress() {
```

Once done you will see the project created under Project explorer/Navigator.

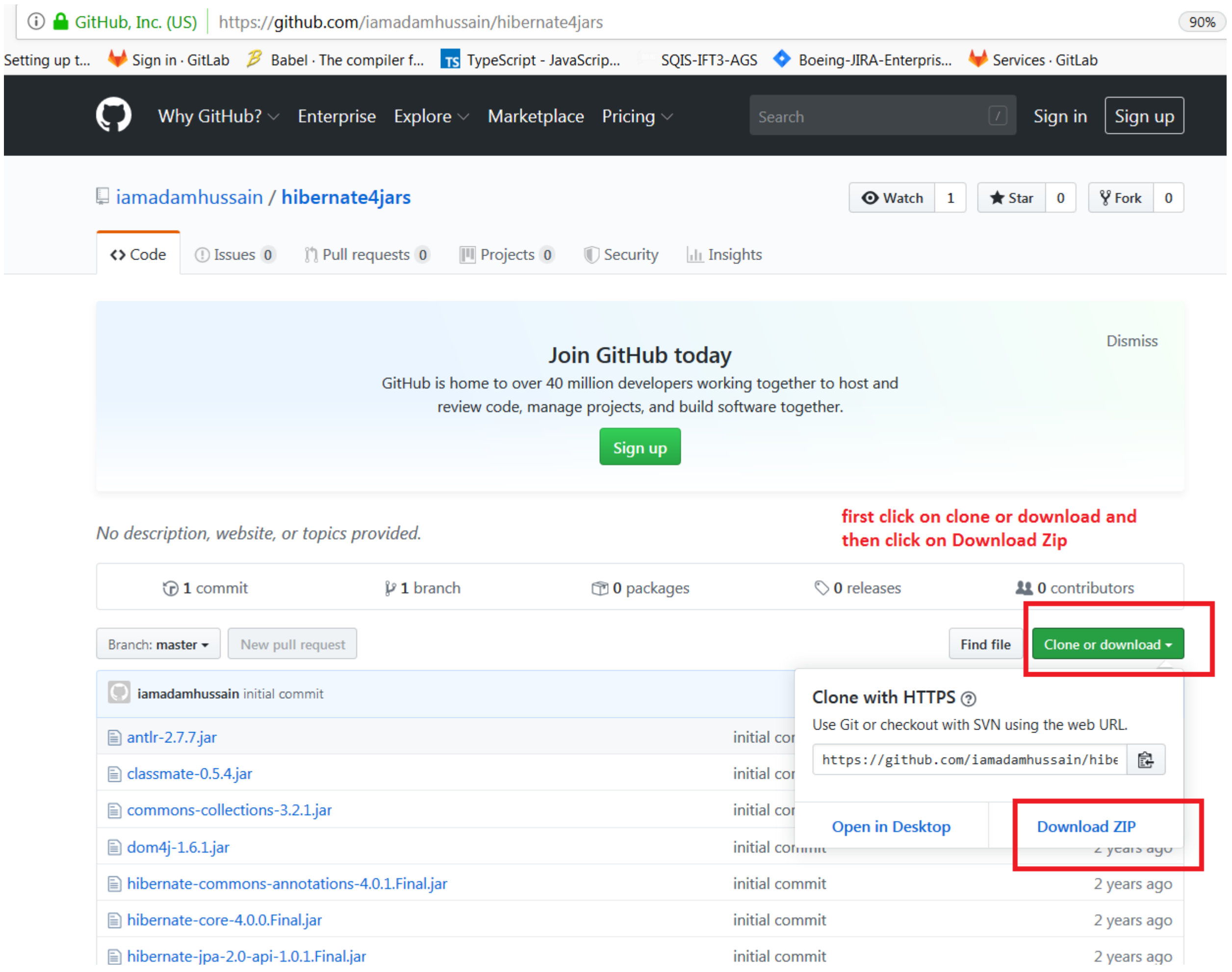


Then create lib folder under project, to add hibernate jars.

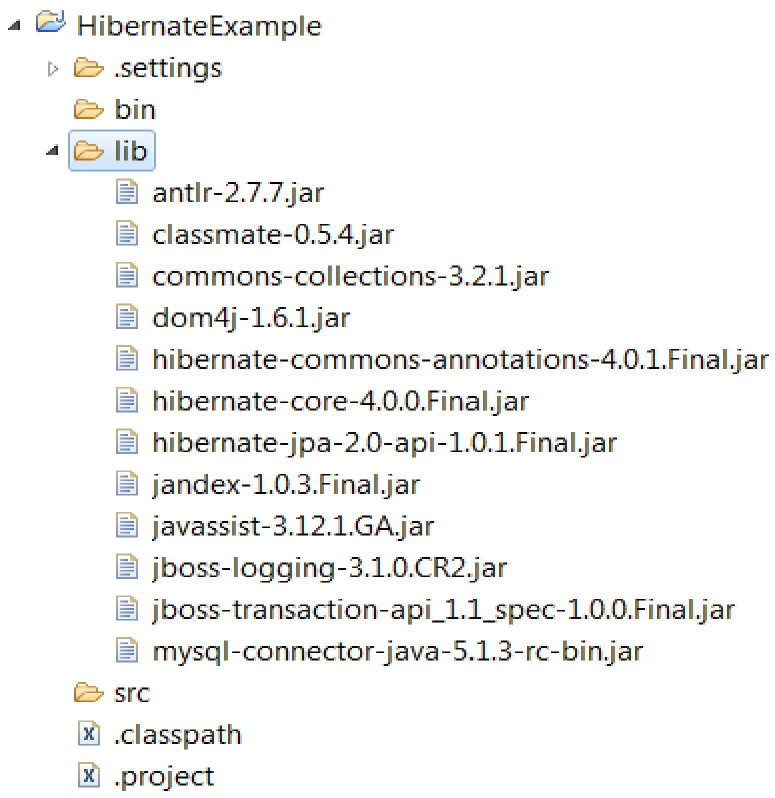
Right Click on Project -> New -> Folder and give the name “lib”.Once done, you can see “lib” folder created.

First download hibernate Hibernate jars from the below link

https://github.com/iamadamhussain/hibernate4jars as shown below.


GitHub, Inc. (US) https://github.com/iamadamhussain/hibernate4jars
Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up
iamadamhussain / hibernate4jars
Watch 1 Star 0 Fork 0
Code Issues 0 Pull requests 0 Projects 0 Security Insights
Join GitHub today
GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.
Sign up
Dismiss
No description, website, or topics provided.
first click on clone or download and then click on Download Zip
1 commit 1 branch 0 packages 0 releases 0 contributors
Branch: master New pull request Find file Clone or download
Clone with HTTPS
Use Git or checkout with SVN using the web URL.
https://github.com/iamadamhussain/hibe
Open in Desktop Download ZIP
iamadamhussain initial commit
antlr-2.7.7.jar
classmate-0.5.4.jar
commons-collections-3.2.1.jar
dom4j-1.6.1.jar
hibernate-commons-annotations-4.0.1.Final.jar initial commit 2 years ago
hibernate-core-4.0.0.Final.jar initial commit 2 years ago
hibernate-jpa-2.0-api-1.0.1.Final.jar initial commit 2 years ago

After downloading jars, extract and paste in side lib folder of the project.

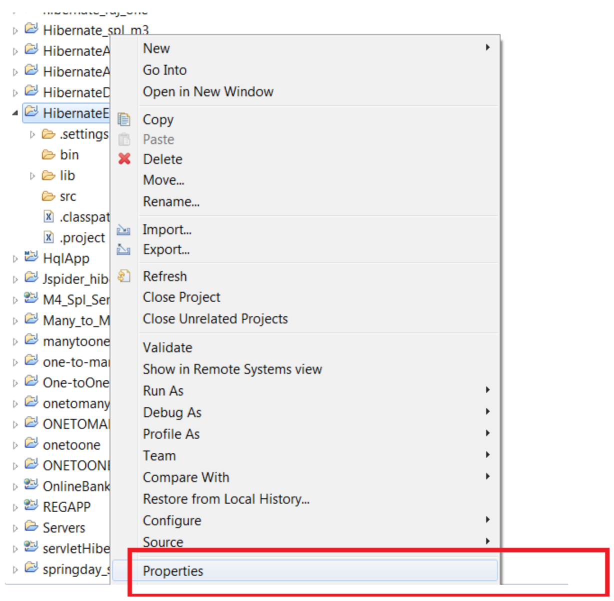


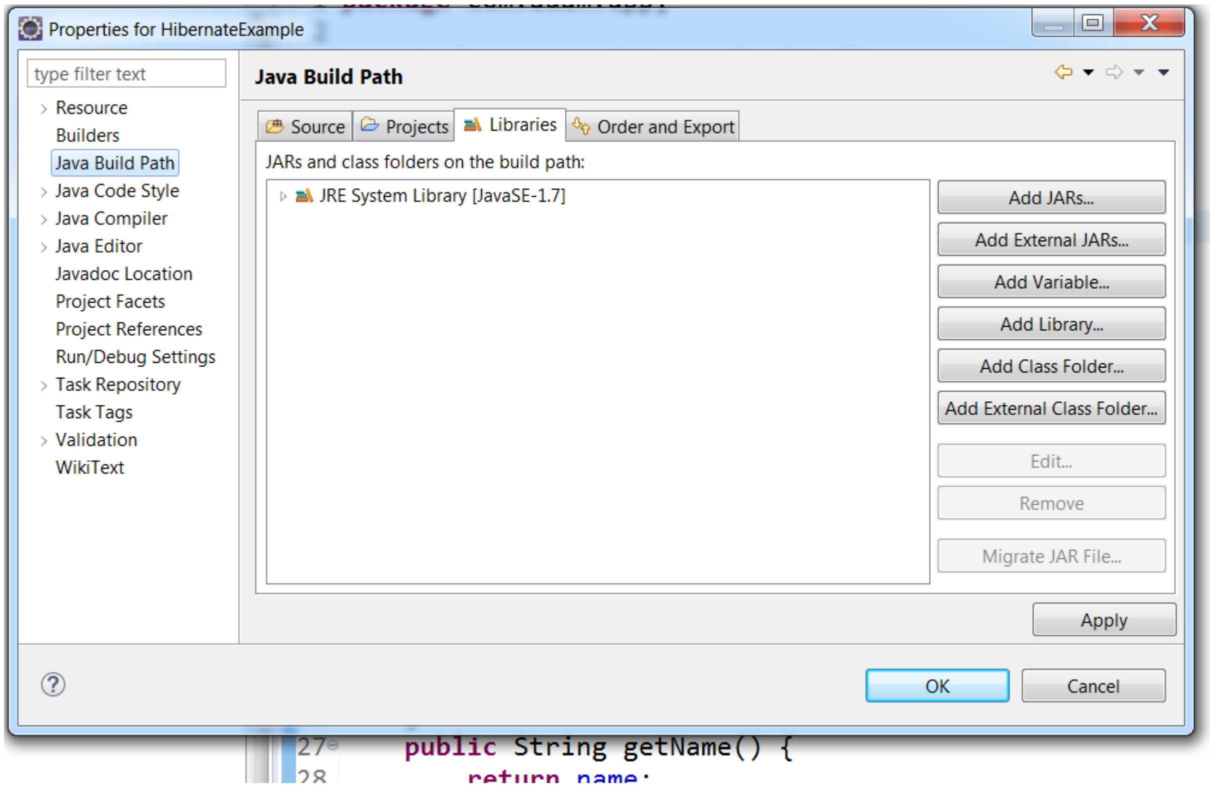
```
HibernateExample
  .settings
  bin
  lib
    antlr-2.7.7.jar
    classmate-0.5.4.jar
    commons-collections-3.2.1.jar
    dom4j-1.6.1.jar
    hibernate-commons-annotations-4.0.1.Final.jar
    hibernate-core-4.0.0.Final.jar
    hibernate-jpa-2.0-api-1.0.1.Final.jar
    jandex-1.0.3.Final.jar
    javassist-3.12.1.GA.jar
    jboss-logging-3.1.0.CR2.jar
    jboss-transaction-api_1.1_spec-1.0.0.Final.jar
    mysql-connector-java-5.1.3-rc-bin.jar
  src
  .classpath
  .project
```

lib/required from downloaded Hibernate) and paste in lib folder.

We need to have these jar files in java build path (class path). To do so select all the jar files and

Right Click project -> Properties ->Build Path -> go to java Build Path (see below figure)




Properties for HibernateExample

type filter text
- Resource
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project Facets
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

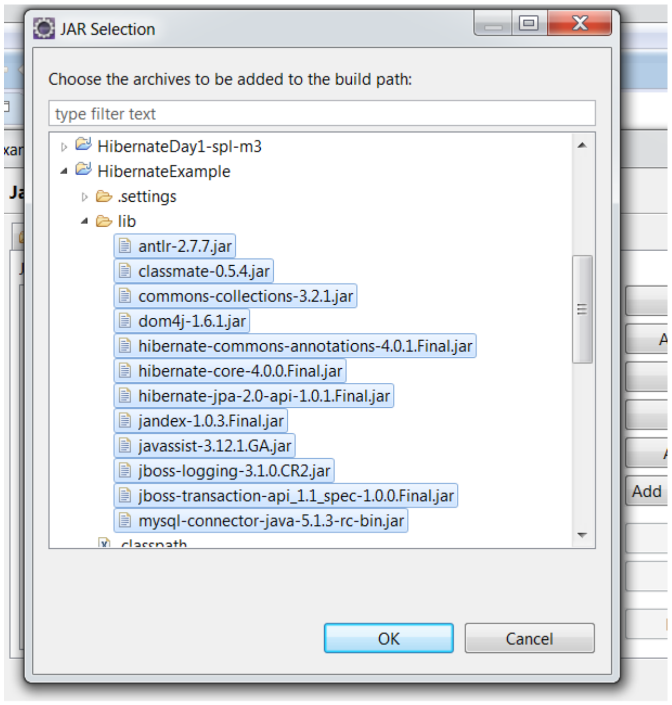
JRE System Library [JavaSE-1.7]

Add JARs... | Add External JARs... | Add Variable... | Add Library... | Add Class Folder... | Add External Class Folder... | Edit... | Remove | Migrate JAR File...

Apply

OK | Cancel

Click on Add JARs button, select jar from the project and select all added jars, show below


JAR Selection

Choose the archives to be added to the build path:

type filter text

```
HibernateDay1-spl-m3
HibernateExample
  .settings
  lib
    antlr-2.7.7.jar
    classmate-0.5.4.jar
    commons-collections-3.2.1.jar
    dom4j-1.6.1.jar
    hibernate-commons-annotations-4.0.1.Final.jar
    hibernate-core-4.0.0.Final.jar
    hibernate-jpa-2.0-api-1.0.1.Final.jar
    jandex-1.0.3.Final.jar
    javassist-3.12.1.GA.jar
    jboss-logging-3.1.0.CR2.jar
    jboss-transaction-api_1.1_spec-1.0.0.Final.jar
    mysql-connector-java-5.1.3-rc-bin.jar
  .classpath
```

OK | Cancel

To select all jars atonce,select first jar then press shift button and select last jar

Then click ok and ok.

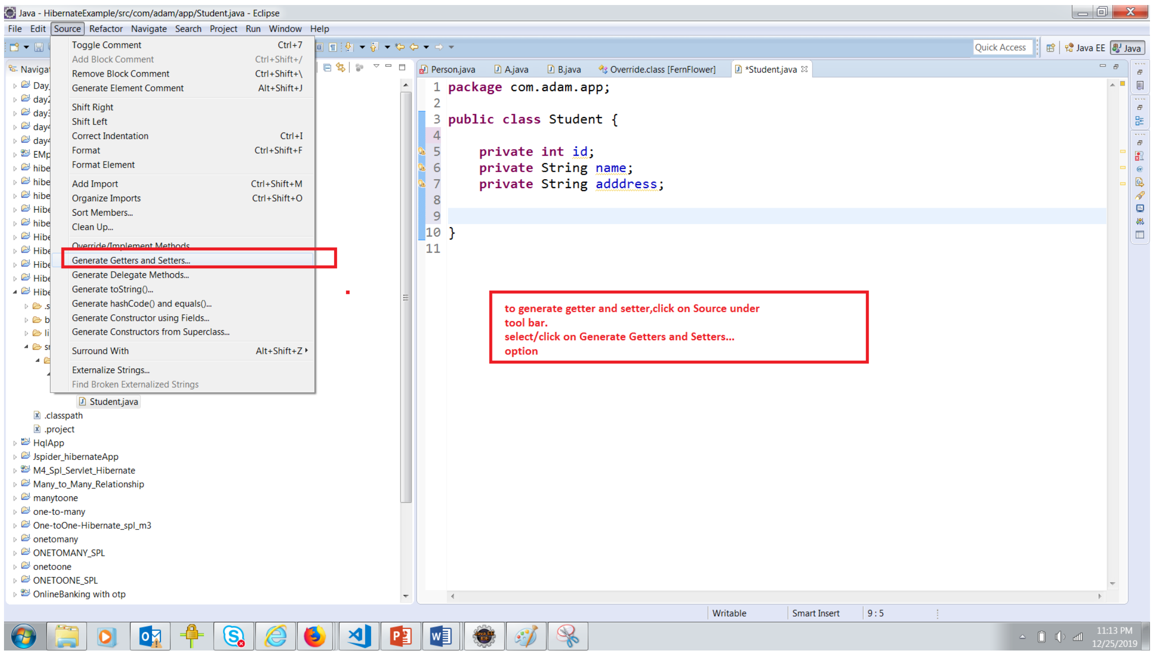
Create package under src folder.

Right Click Src -> New ->Folder/Package ->

Enter package name “com/adam/app” Or Enter package name “com.adam.app”

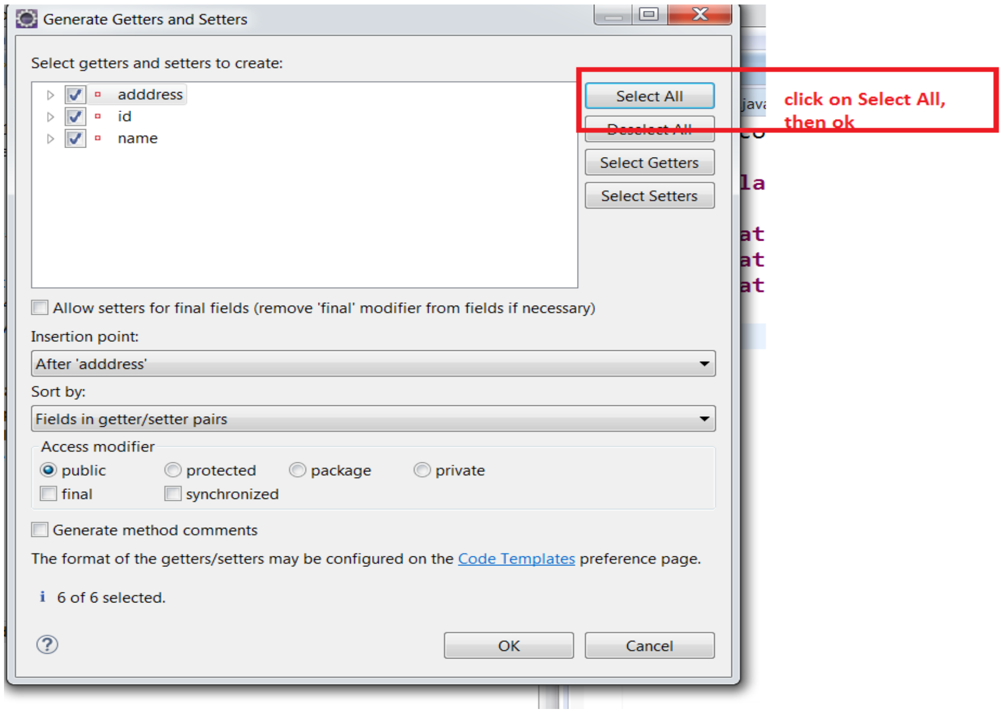
Create JavaBean/Pojo class.

Creating Student.java under package and generate getter n setter show below


Java - HibernateExample/src/com/adam/app/Student.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Toggle Comment Ctrl+7
Add Block Comment Ctrl+Shift+/
Remove Block Comment Ctrl+Shift+\
Generate Element Comment Alt+Shift+J
Shift Right
Shift Left
Correct Indentation Ctrl+I
Format Ctrl+Shift+F
Format Element
Add Import Ctrl+Shift+M
Organize Imports Ctrl+Shift+O
Sort Members...
Clean Up...
Override/Implement Methods...
Generate Getters and Setters...
Generate Delegate Methods...
Generate toString()...
Generate hashCode() and equals()...
Generate Constructor using Fields...
Generate Constructors from Superclass...
Surround With Alt+Shift+Z
Externalize Strings...
Find Broken Externalized Strings
Person.java A.java B.java Override.class [FernFlower] *Student.java
Student.java
.classpath
.project
HqlApp
Jspider_hibernateApp
M4_Spl_Servlet_Hibernate
Many_to_Many_Relationship
manytoone
one-to-many
One-toOne-Hibernate_spl_m3
onetomany
ONETOMANY_SPL
onetoone
ONETOONE_SPL
OnlineBanking with otp
Quick Access
Java EE
Java
Writable
Smart Insert
9 : 5

```java
package com.adam.app;

public class Student {

    private int id;
    private String name;
    private String adddress;


}
```

to generate getter and setter,click on Source under tool bar.
select/click on Generate Getters and Setters...
option



Select getters and setters to create:

- [✓] adddress
- [✓] id
- [✓] name

click on Select All, then ok

[ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: After 'adddress'

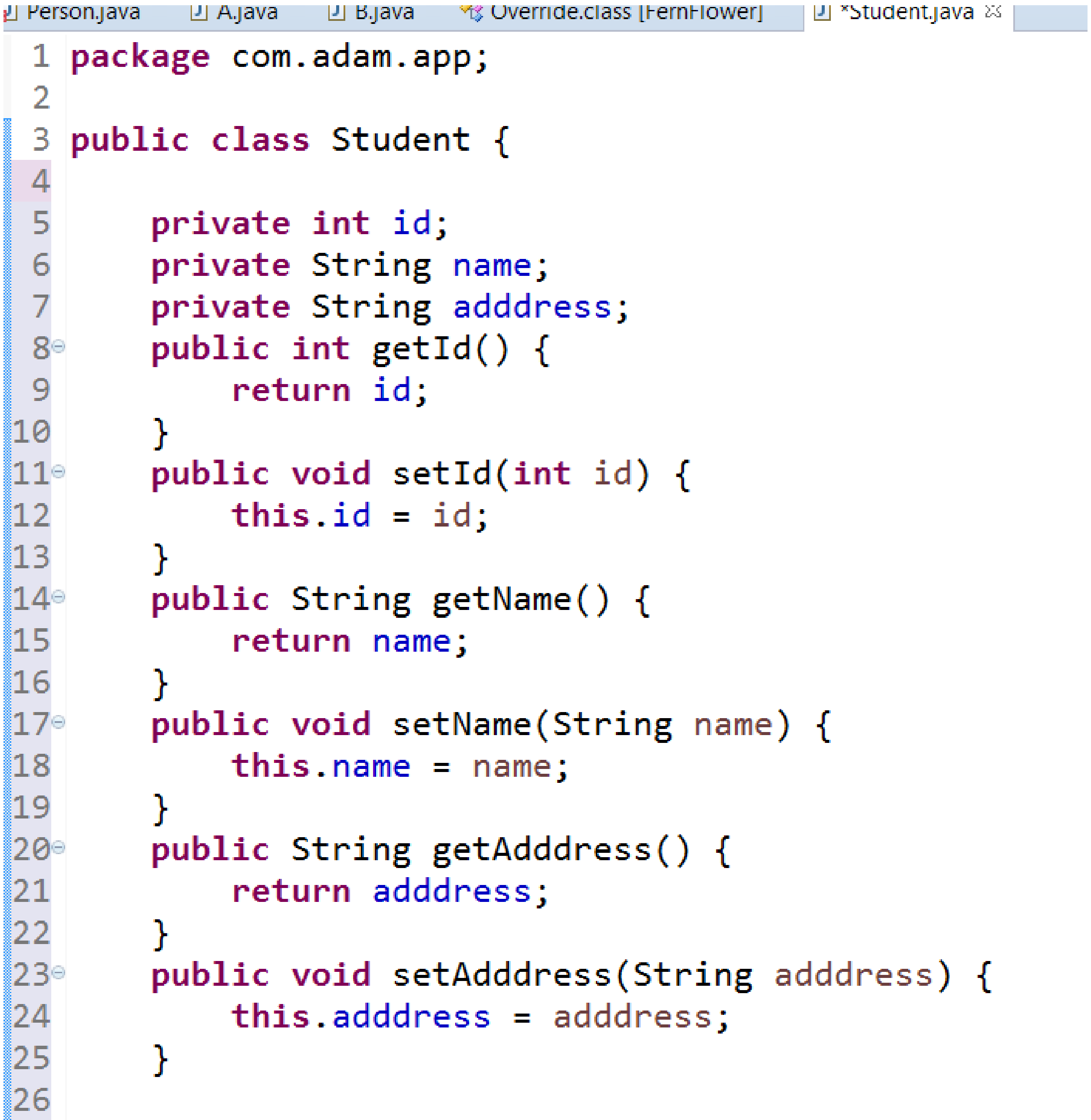
Sort by: Fields in getter/setter pairs

Access modifier: [✓ public] protected package private
[ ] final [ ] synchronized

[ ] Generate method comments

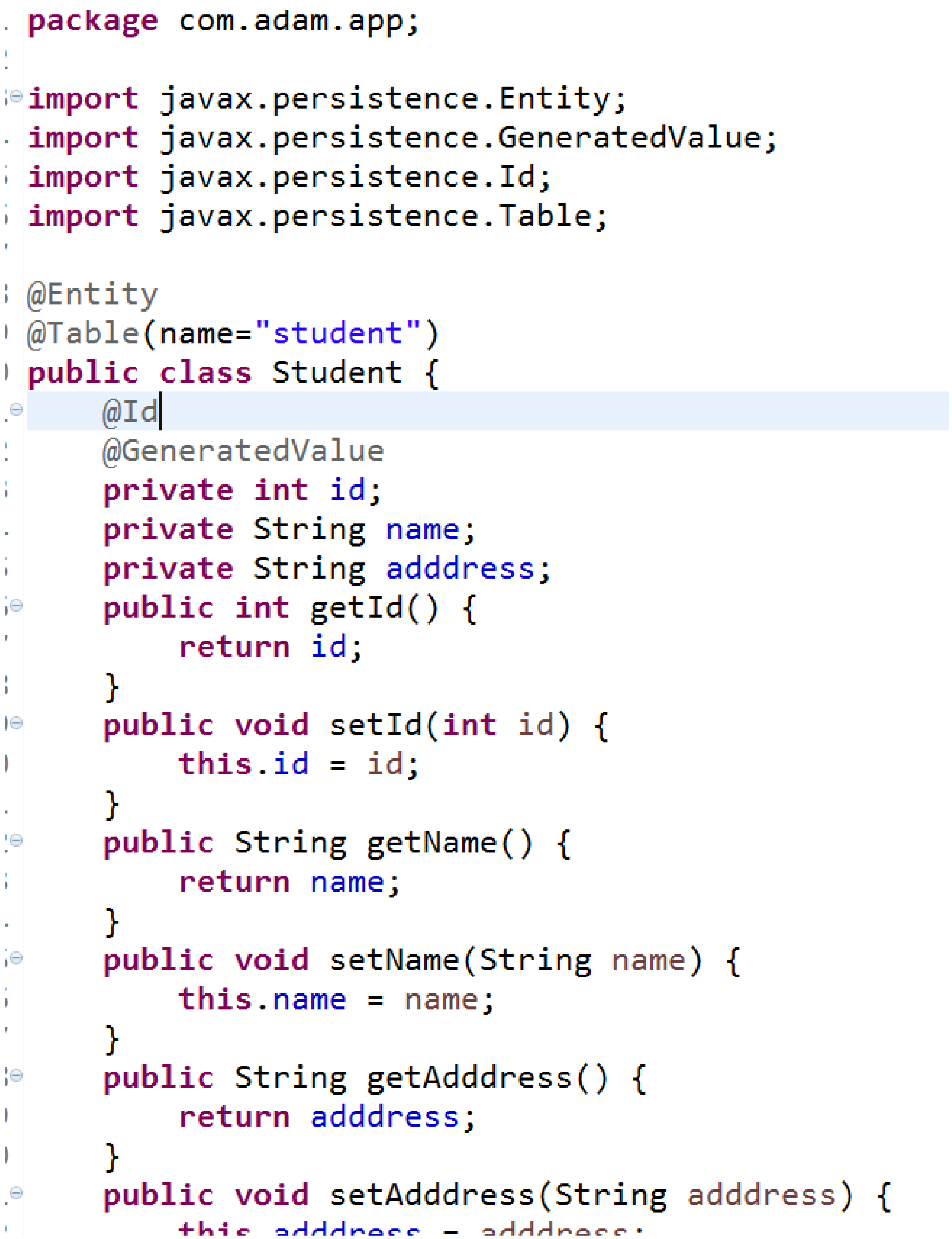
The format of the getters/setters may be configured on the Code Templates preference page.

6 of 6 selected.



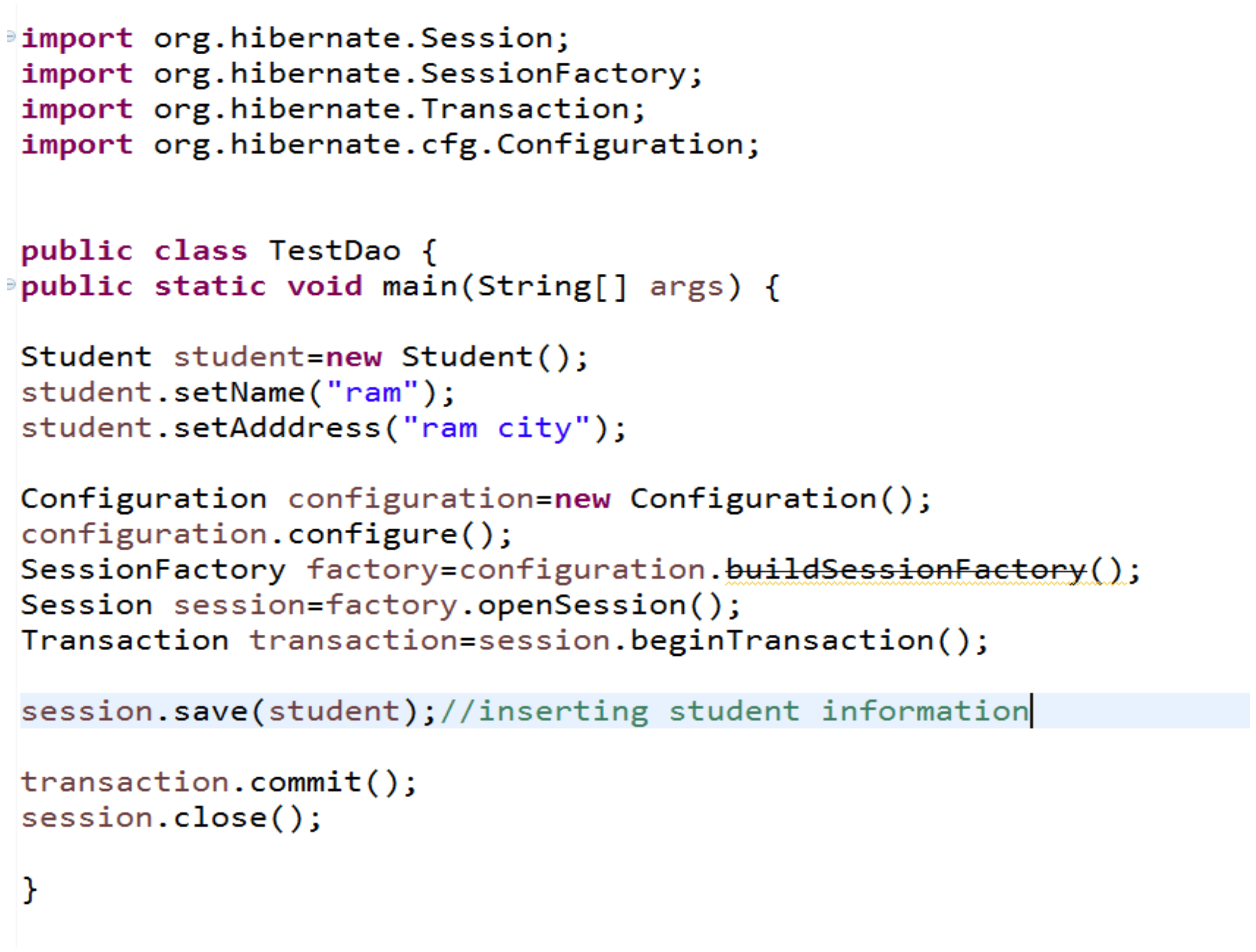
```java
package com.adam.app;

public class Student {

    private int id;
    private String name;
    private String adddress;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAdddress() {
        return adddress;
    }
    public void setAdddress(String adddress) {
        this.adddress = adddress;
    }

```

Add Hibernate JPA annoation in JavaBean class as shown in below



```java
package com.adam.app;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="student")
public class Student {
    @Id
    @GeneratedValue
    private int id;
    private String name;
    private String adddress;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAdddress() {
        return adddress;
    }
    public void setAdddress(String adddress) {
        this.adddress = adddress;
```

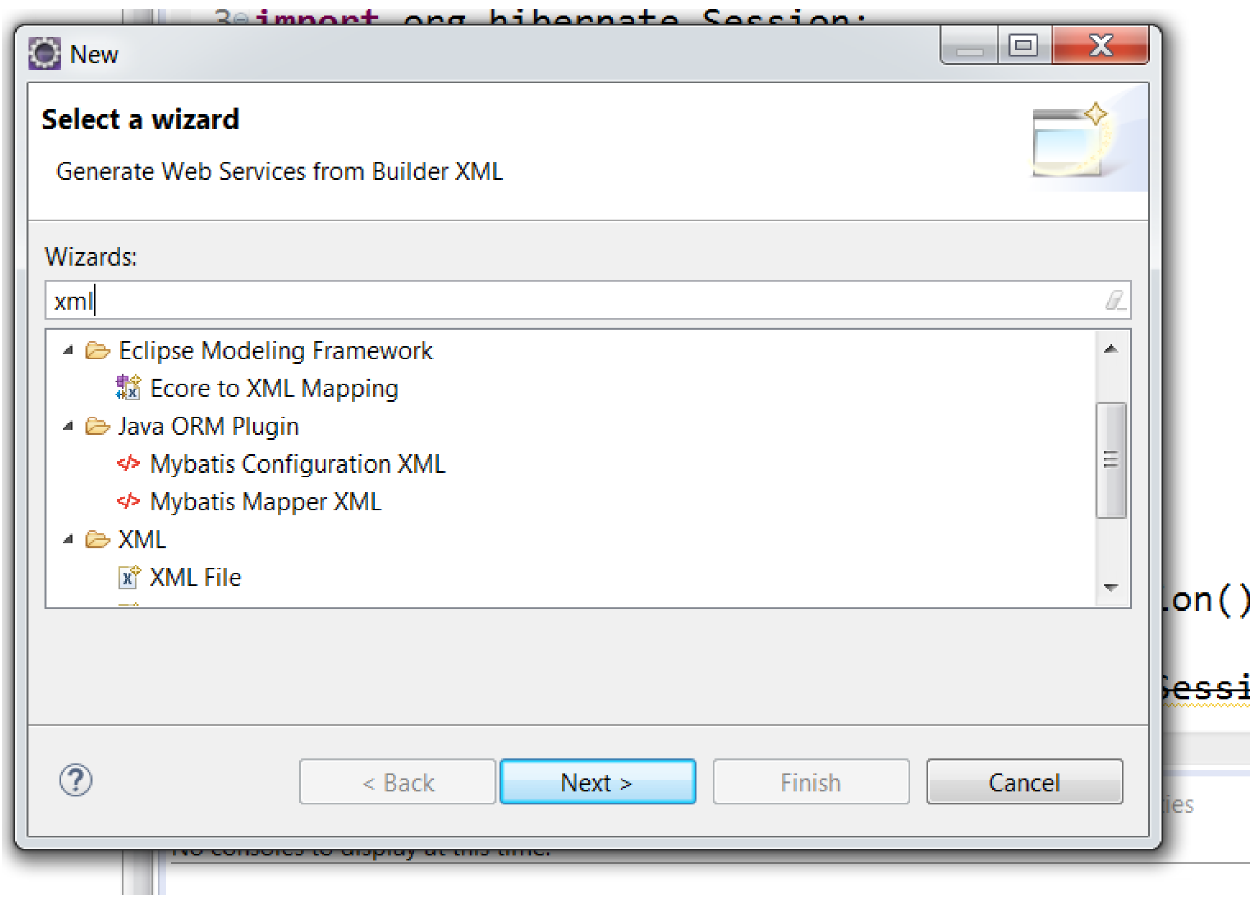
Create TestDao.java in package for running the hibernate application



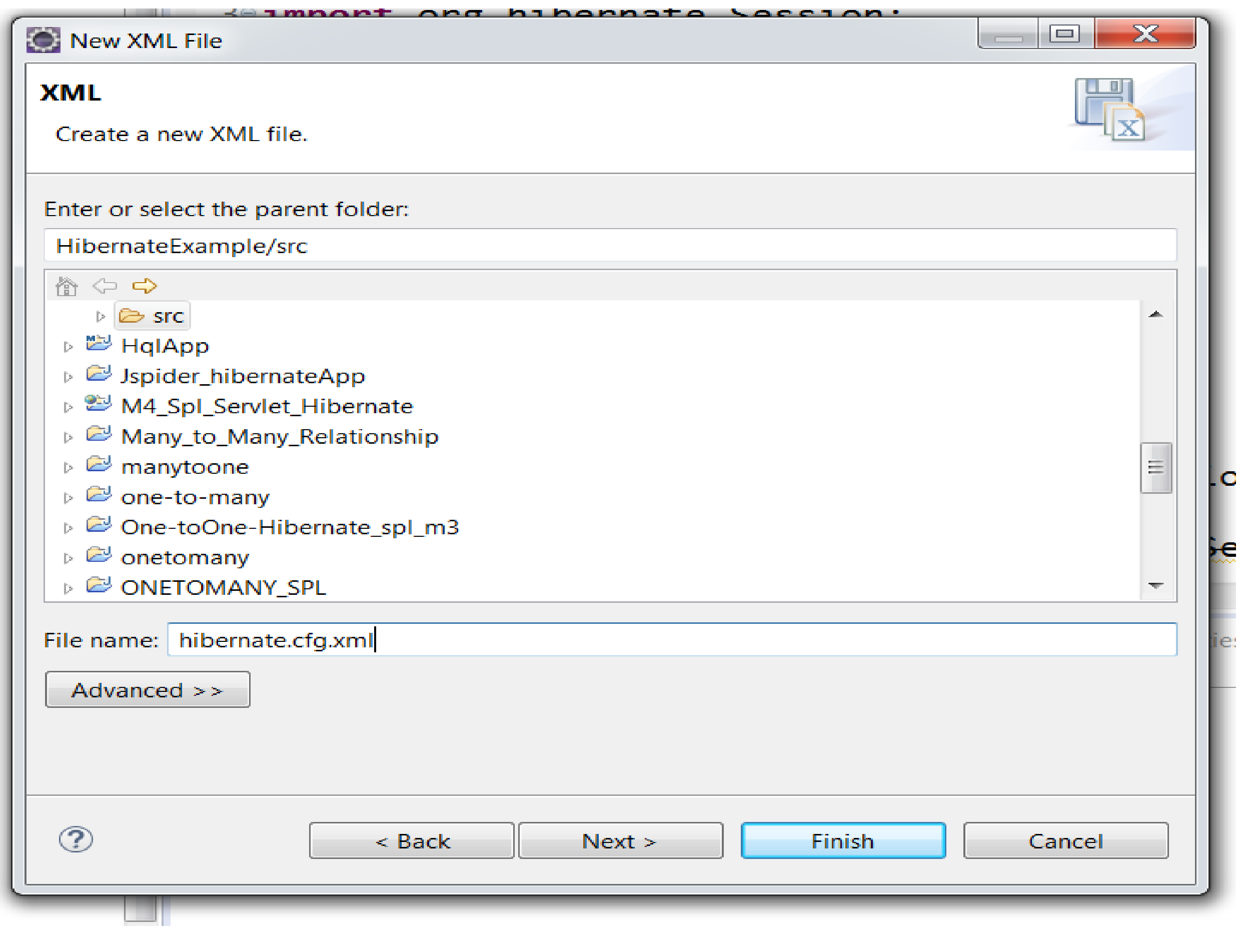
```java
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;


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

Student student=new Student();
student.setName("ram");
student.setAdddress("ram city");

Configuration configuration=new Configuration();
configuration.configure();
SessionFactory factory=configuration.~~buildSessionFactory~~();
Session session=factory.openSession();
Transaction transaction=session.beginTransaction();

session.save(student);//inserting student information

transaction.commit();
session.close();

}
```

Create Xml file Right Click Src -> New -> other ->

Then type xml in wizard and select XML File -→ click on next Button as shown in below



Then give file name as **hibernate.cfg.xml** and click on finish as shown below


New XML File

XML

Create a new XML file.

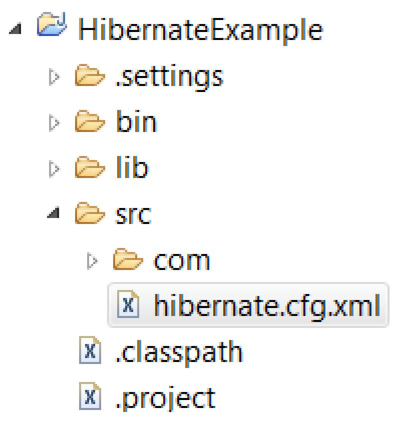
Enter or select the parent folder:

HibernateExample/src

- src
- HqlApp
- Jspider_hibernateApp
- M4_Spl_Servlet_Hibernate
- Many_to_Many_Relationship
- manytoone
- one-to-many
- One-toOne-Hibernate_spl_m3
- onetomany
- ONETOMANY_SPL

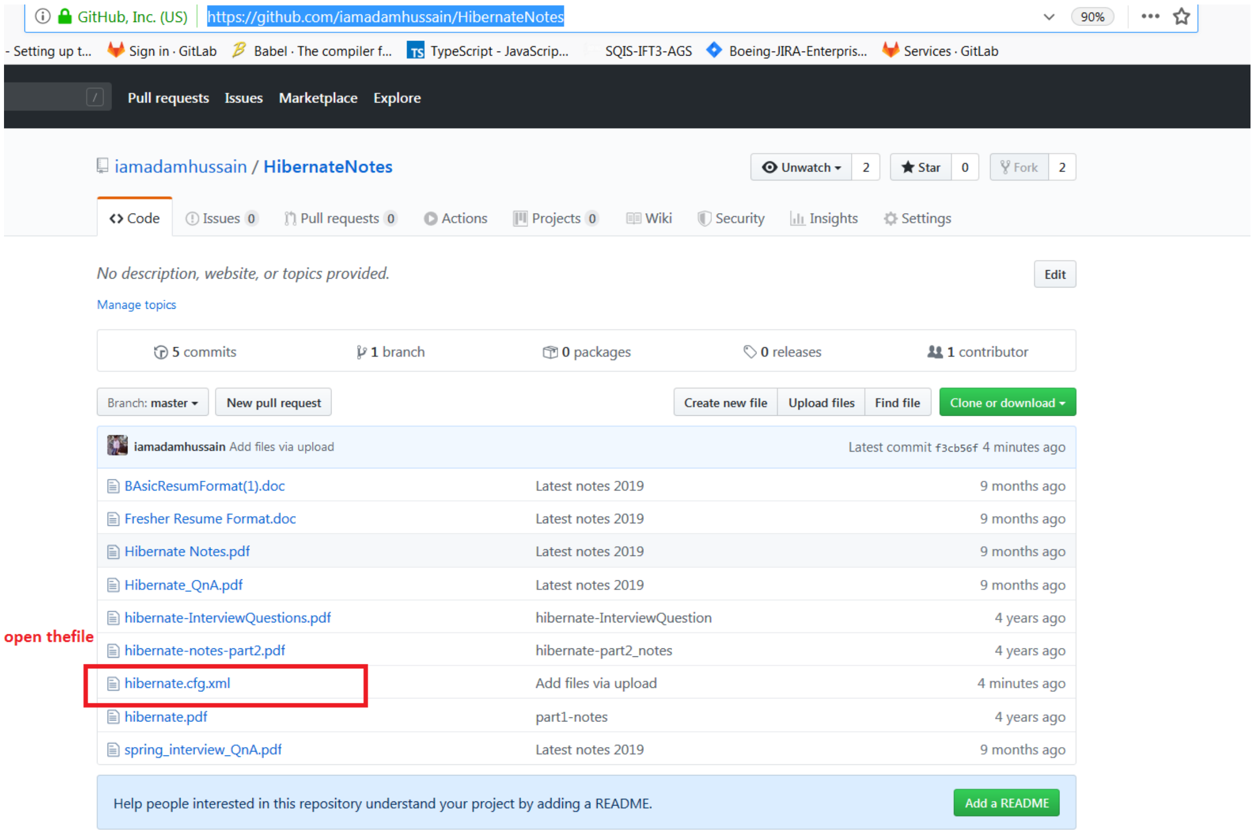
File name: hibernate.cfg.xml

Once click on finish ,then project would been seen as shown below


```
HibernateExample
    .settings
    bin
    lib
    src
        com
        hibernate.cfg.xml
    .classpath
    .project
```

Copy the content of hibernate.cfg.xml

From the link https://github.com/iamadamhussain/HibernateNotes


GitHub, Inc. (US) https://github.com/iamadamhussain/HibernateNotes
90%
- Setting up t... Sign in · GitLab Babel · The compiler f... TypeScript - JavaScrip... SQIS-IFT3-AGS Boeing-JIRA-Enterpris... Services · GitLab
Pull requests Issues Marketplace Explore
iamadamhussain / HibernateNotes
Unwatch 2 Star 0 Fork 2
Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings
Edit
Manage topics
Branch: master New pull request Create new file Upload files Find file Clone or download
Add a README

No description, website, or topics provided.

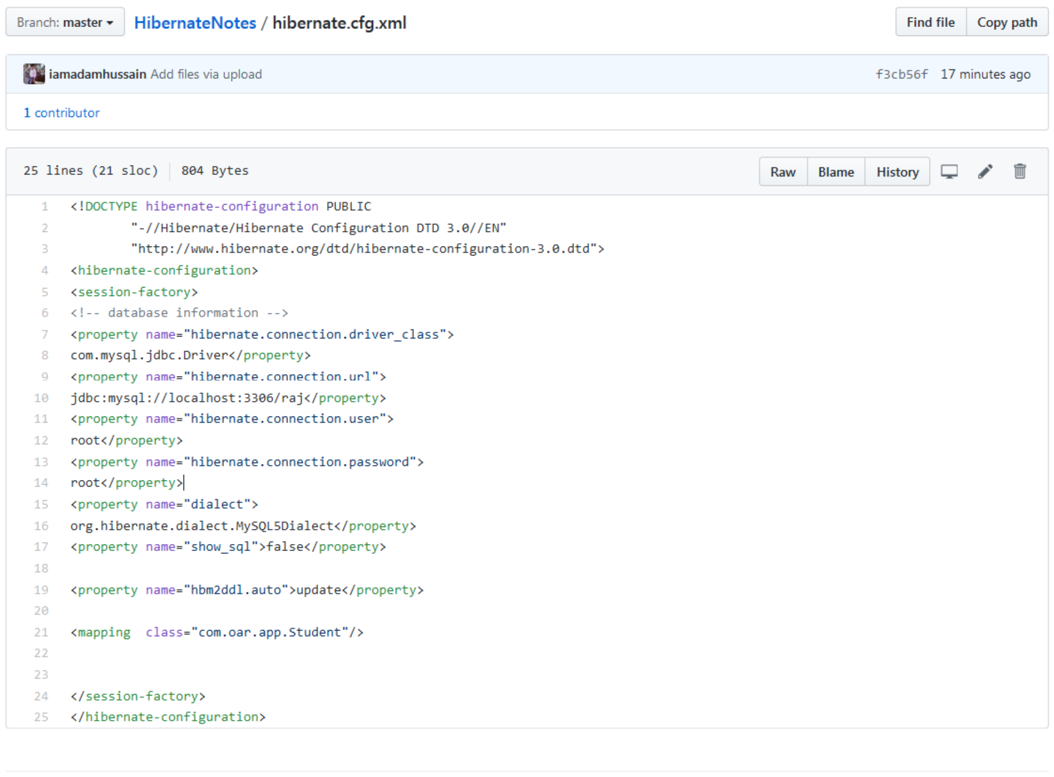
5 commits 1 branch 0 packages 0 releases 1 contributor

iamadamhussain Add files via upload — Latest commit f3cb56f 4 minutes ago

BAsicResumFormat(1).doc	Latest notes 2019	9 months ago
Fresher Resume Format.doc	Latest notes 2019	9 months ago
Hibernate Notes.pdf	Latest notes 2019	9 months ago
Hibernate_QnA.pdf	Latest notes 2019	9 months ago
hibernate-InterviewQuestions.pdf	hibernate-InterviewQuestion	4 years ago
hibernate-notes-part2.pdf	hibernate-part2_notes	4 years ago
hibernate.cfg.xml	Add files via upload	4 minutes ago
hibernate.pdf	part1-notes	4 years ago
spring_interview_QnA.pdf	Latest notes 2019	9 months ago

open thefile

Help people interested in this repository understand your project by adding a README.



HibernateNotes / hibernate.cfg.xml

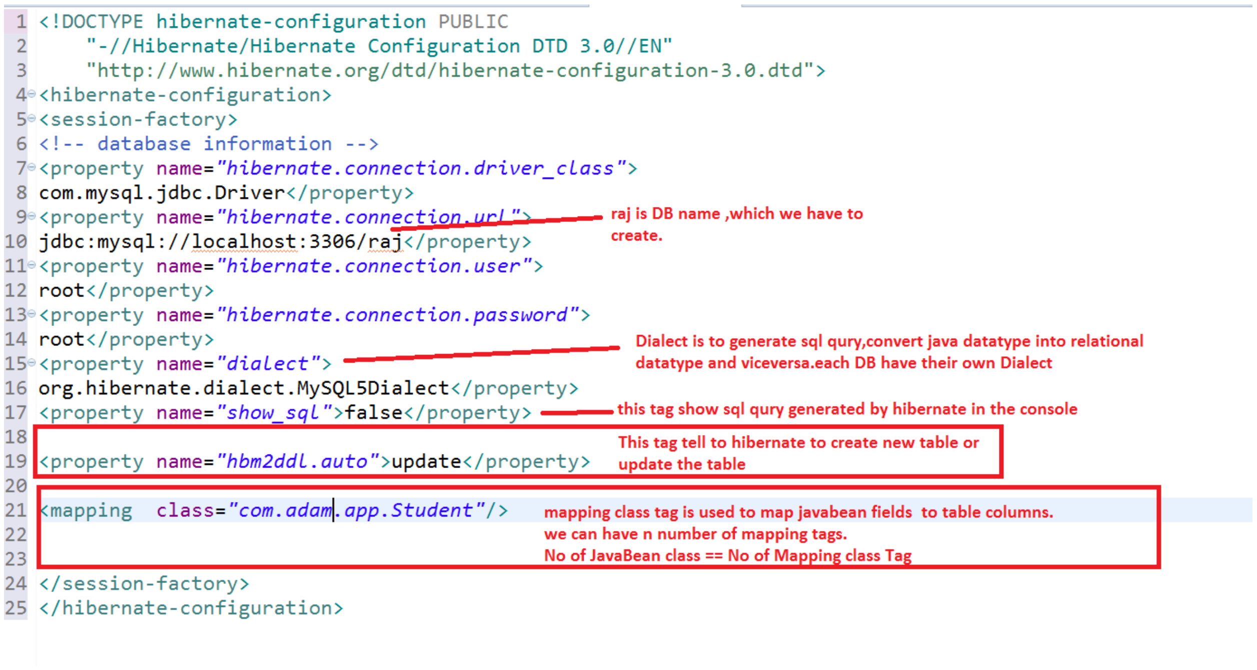
iamadamhussain Add files via upload — f3cb56f 17 minutes ago

1 contributor

25 lines (21 sloc) 804 Bytes

```
<!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>
<!-- database information -->
<property name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">
jdbc:mysql://localhost:3306/raj</property>
<property name="hibernate.connection.user">
root</property>
<property name="hibernate.connection.password">
root</property>
<property name="dialect">
org.hibernate.dialect.MySQL5Dialect</property>
<property name="show_sql">false</property>

<property name="hbm2ddl.auto">update</property>

<mapping  class="com.oar.app.Student"/>


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

Then paste into src --→ hibernate.cfg.xml as shown below



```
<!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>
<!-- database information -->
<property name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">
jdbc:mysql://localhost:3306/raj</property>
<property name="hibernate.connection.user">
root</property>
<property name="hibernate.connection.password">
root</property>
<property name="dialect">
org.hibernate.dialect.MySQL5Dialect</property>
<property name="show_sql">false</property>

<property name="hbm2ddl.auto">update</property>

<mapping  class="com.adam.app.Student"/>


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

raj is DB name ,which we have to create.

Dialect is to generate sql qury,convert java datatype into relational datatype and viceversa.each DB have their own Dialect

this tag show sql qury generated by hibernate in the console

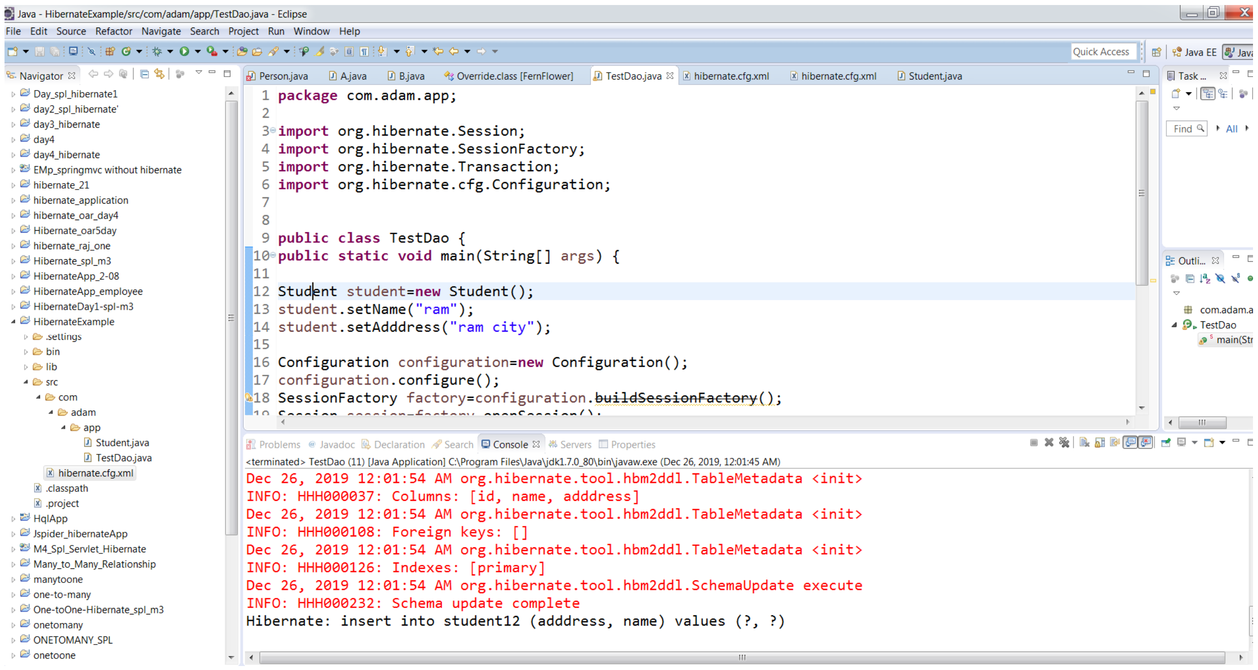
This tag tell to hibernate to create new table or update the table

mapping class tag is used to map javabean fields to table columns.
we can have n number of mapping tags.
No of JavaBean class == No of Mapping class Tag

Last RUN TestDao.java

Before running TestDao.java make show you create DB Name in your local DB.

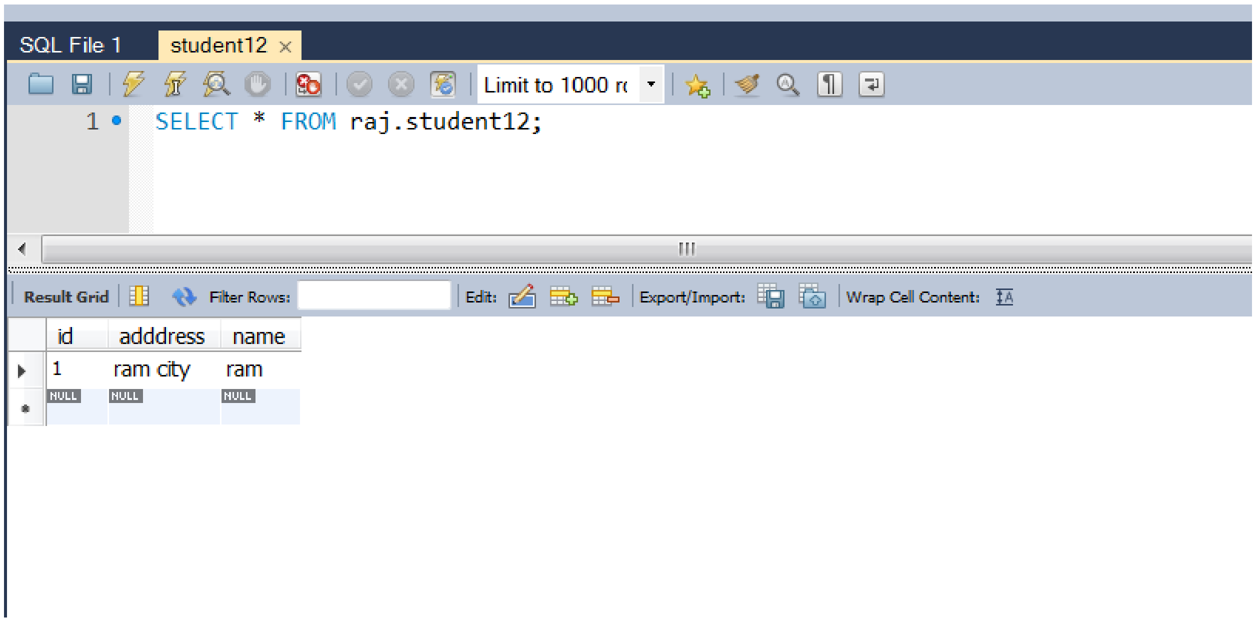
Output shown below



```
package com.adam.app;

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


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

Student student=new Student();
student.setName("ram");
student.setAdddress("ram city");

Configuration configuration=new Configuration();
configuration.configure();
SessionFactory factory=configuration.buildSessionFactory();
```

```
<terminated> TestDao (11) [Java Application] C:\Program Files\Java\jdk1.7.0_80\bin\javaw.exe (Dec 26, 2019, 12:01:45 AM)
Dec 26, 2019 12:01:54 AM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000037: Columns: [id, name, adddress]
Dec 26, 2019 12:01:54 AM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000108: Foreign keys: []
Dec 26, 2019 12:01:54 AM org.hibernate.tool.hbm2ddl.TableMetadata <init>
INFO: HHH000126: Indexes: [primary]
Dec 26, 2019 12:01:54 AM org.hibernate.tool.hbm2ddl.SchemaUpdate execute
INFO: HHH000232: Schema update complete
Hibernate: insert into student12 (adddress, name) values (?, ?)
```

Running the program will end up in below state of Student12 Table



```
SELECT * FROM raj.student12;
```

id	adddress	name
1	ram city	ram
NULL	NULL	NULL

Annotations Details

a. **@Entity** – this annotation is used at a class level to define that this class is an entity so the class must follow all the entity rules.

b. **@Table**- this annotation is also used at a class level and is used to provide the table name to which the entity will be mapped. We need to use name attribute of @Table annotation to supply the table name.

d. **@Id** – This annotation is applied on a column and it has a special meaning. This informs hibernate that corresponding property will be used as an entity identifier.

e. **@Generated Value** – Hibernate automatically uses the id generation strategy when we add @Id annotation. However to override the ID generation strategy we can use @GeneratedValue annotation with @Id. This annotation accepts two attributes strategy and generator.

e. **@Column**- this annotation is applied on a property and is used to map the property with table column. @Column annotation exposes following commonly used attributes-

- **name**- to specify the name of column
- **length-** to specify the length of column

- **nullable**- to apply not-null constraint
- **unique**- to apply unique constraint
- **insertable**- is equivalent to insert=false/true which decide to use this column in insert or not when the value is null
- **updatable -** is equivalent to update=false/true which decide to use this column in update or not when the value is null

f. **@Version** – this annotation is applied on column and specifies that this column will be used for versioning.

g. **@Transient**- By default all non-static properties are considered as persistent so to ignore any particular property from persistent, add @Transient annotation. If we do not mark field with @Column then also property will be considered as persistent and it will be mapped with the column of same name as property name. Hibernate assumes those fields to be annotated as @Basic.

Below is the detail of Associations Annotations

h. **@OneToMany** – this annotation is applied on a collection property to manage one to many relationship. Use mappedBy attribute for bidirectional associations.

For a unidirectional association with join column approach , we need to use @JoinColumn annotation in conjunction with @OneToMany annotation

i. **@ManyToOne**- this annotation is used to manage one to many relationship with @JoinColumn to specify the join column in entities.

x. **@ManyToMany** – this annotation is used to manage many to many relationship with @JoinTable annotation to specify the join table.

xi. **@OneToOne**- this annotation is used to manage one to one relationship with @JoinColumn annotation to specify foreign key- primary key relationship