HIBERNATE & JPA COURSE

EXERCISE

ASSOCIATIONS IN HIBERNATE/JPA

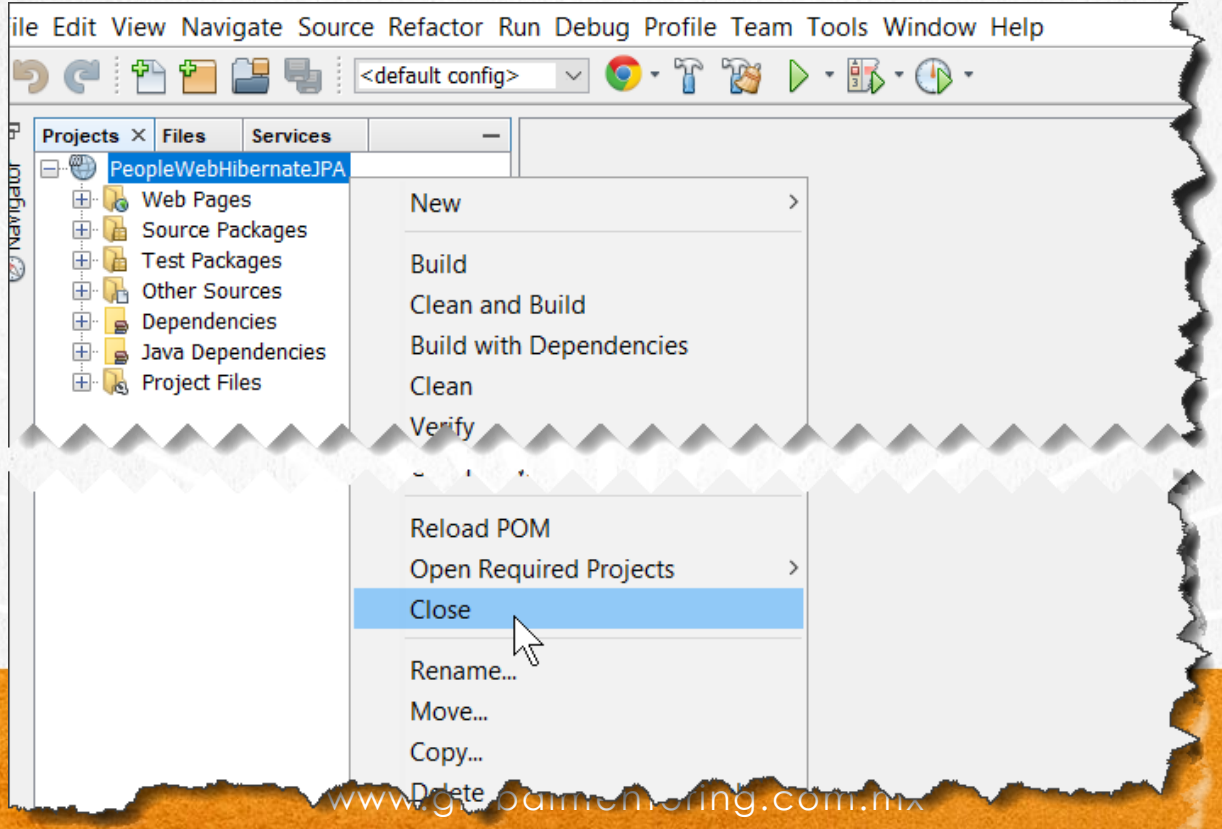
EXERCISE OBJECTIVE

Create a program to practice mapping relationships with Hibernate and / or JPA. At the end we should observe the following:

```
Output
Retriever Output   Run (TestDAOs)
20:31:25 [main] INFO  org.hibernate.hql.internal.QueryTranslatorFactoryInitiator - HHH000397: Using ASTQueryTranslatorFactory
20:31:25 [main] DEBUG org.hibernate.SQL - insert into address (country, deleted, street_name, street_number, version) values (?, ?, ?, ?, ?)
Hibernate: insert into address (country, deleted, street_name, street_number, version) values (?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [VARCHAR] - [Mexico]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [VARCHAR] - [Estrella]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [VARCHAR] - [109]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into user (deleted, password, username, version) values (?, ?, ?, ?)
Hibernate: insert into user (deleted, password, username, version) values (?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [VARCHAR] - [1234]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [VARCHAR] - [john]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into student (id_address, deleted, name, id_user, version) values (?, ?, ?, ?, ?)
Hibernate: insert into student (id_address, deleted, name, id_user, version) values (?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [VARCHAR] - [Jhon Smith]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into course (deleted, name, price, schedule, version) values (?, ?, ?, ?, ?)
Hibernate: insert into course (deleted, name, price, schedule, version) values (?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [VARCHAR] - [Java Fundamentals]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [FLOAT] - [1000.0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [VARCHAR] - [Morning]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into assignation (id_course, deleted, finish_date, start_date, id_student, version) values (?, ?, ?, ?, ?, ?)
Hibernate: insert into assignation (id_course, deleted, finish_date, start_date, id_student, version) values (?, ?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [TIMESTAMP] - [Wed Sep 12 20:31:25
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [TIMESTAMP] - [Wed Sep 12 20:31:25
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [6] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - select address0_.id_address as id_addre1_0_, address0_.country as country2_0_, address0_.deleted as deleted3_0_, addre
Hibernate: select address0_.id_address as id_addre1_0_, address0_.country as country2_0_, address0_.deleted as deleted3_0_, address0_.street_name as street_n4_
Address{idAddress=1, streetName=Estrella, streetNumber=109, country=Mexico, version=0, deleted=0}
20:31:25 [main] DEBUG org.hibernate.SQL - select user0_.id_user as id_user1_4_, user0_.deleted as deleted2_4_, user0_.password as password3_4_, user0_.username
Hibernate: select user0_.id_user as id_user1_4_, user0_.deleted as deleted2_4_, user0_.password as password3_4_, user0_.username as username4_4_, user0_.version
User{idUser=1, username=john, password=1234, version=0, deleted=0}
20:31:25 [main] DEBUG org.hibernate.SQL - select student0_.id_student as id_studel_3_, student0_.id_address as id_addre5_3_, student0_.deleted as deleted2_3_, s
Hibernate: select student0_.id_student as id_studel_3_, student0_.id_address as id_addre5_3_, student0_.deleted as deleted2_3_, student0_.name as name3_3_, st
```

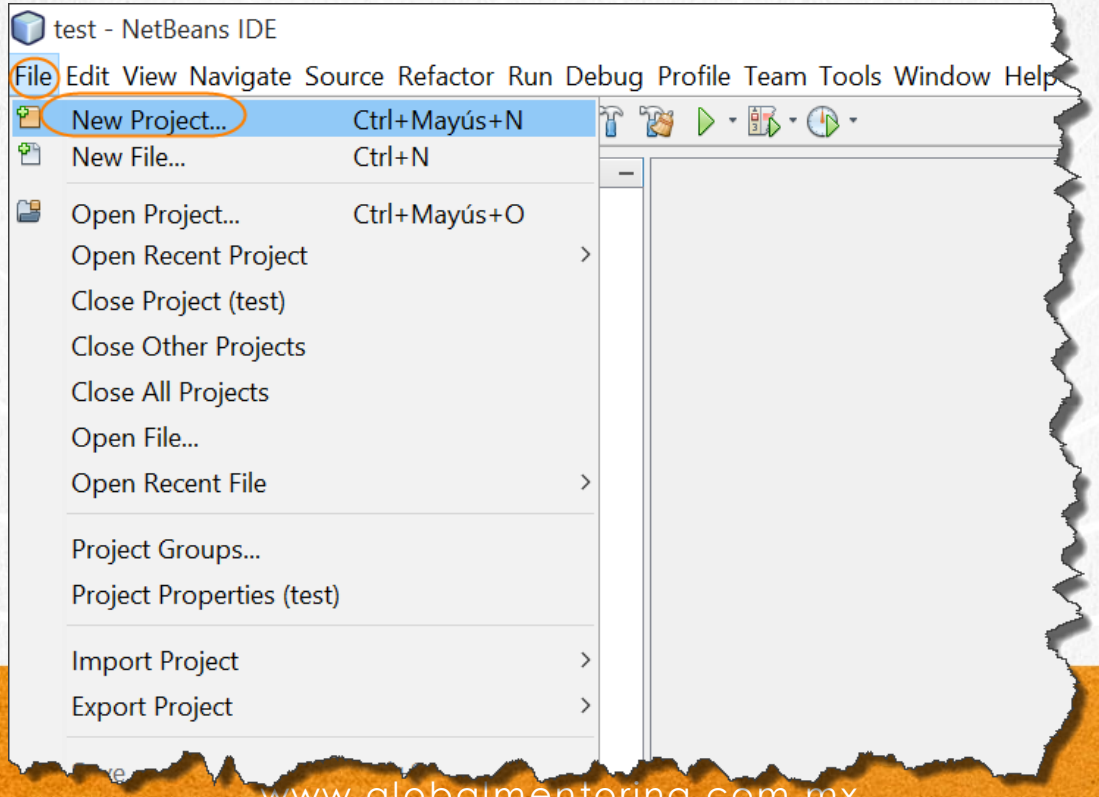
CLOSE ANY OPEN PROJECT

We close any other open project:



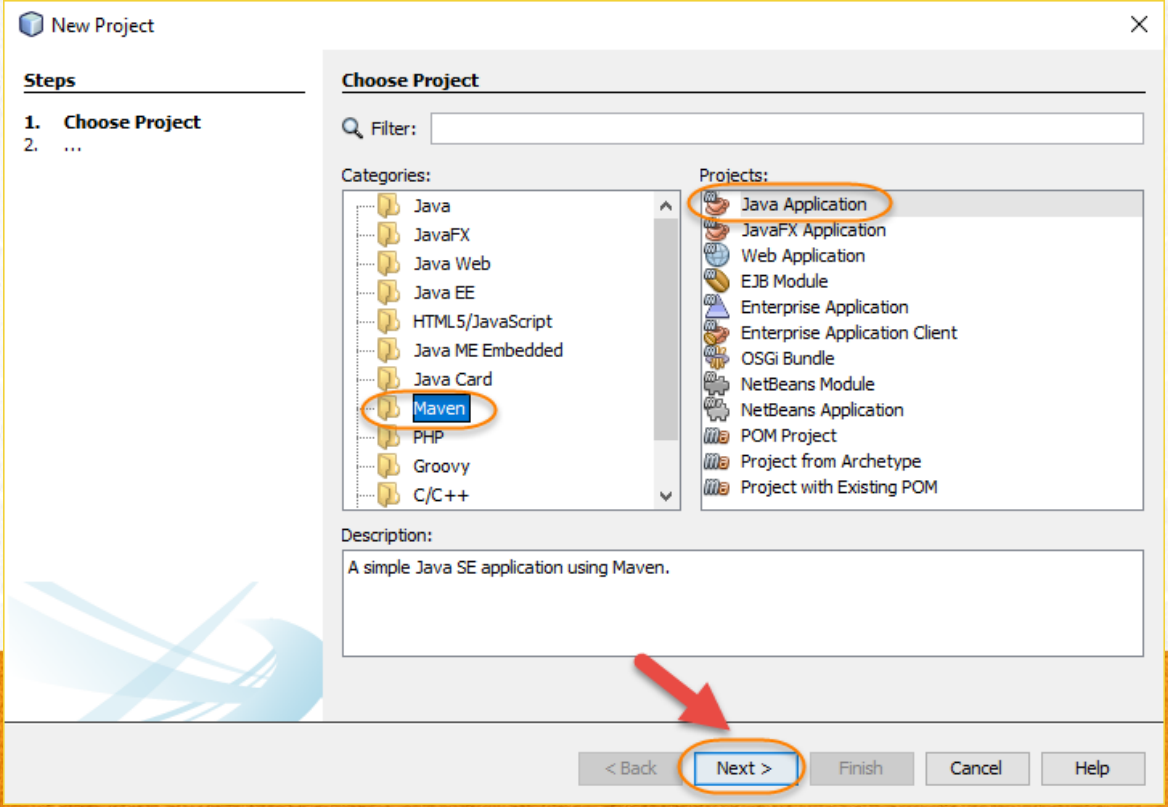
1. CREATE A NEW PROJECT

Create a new Project:



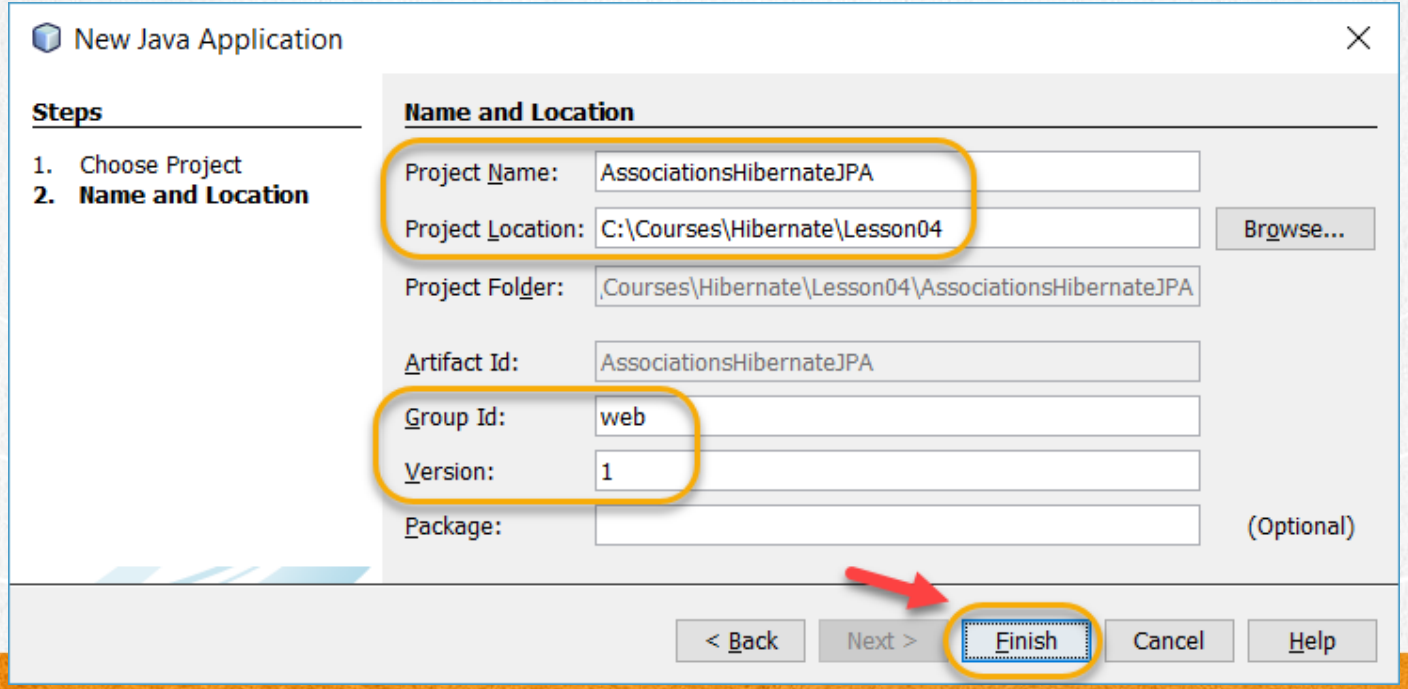
1. CREATE A NEW PROJECT

Create a new Project using Maven:



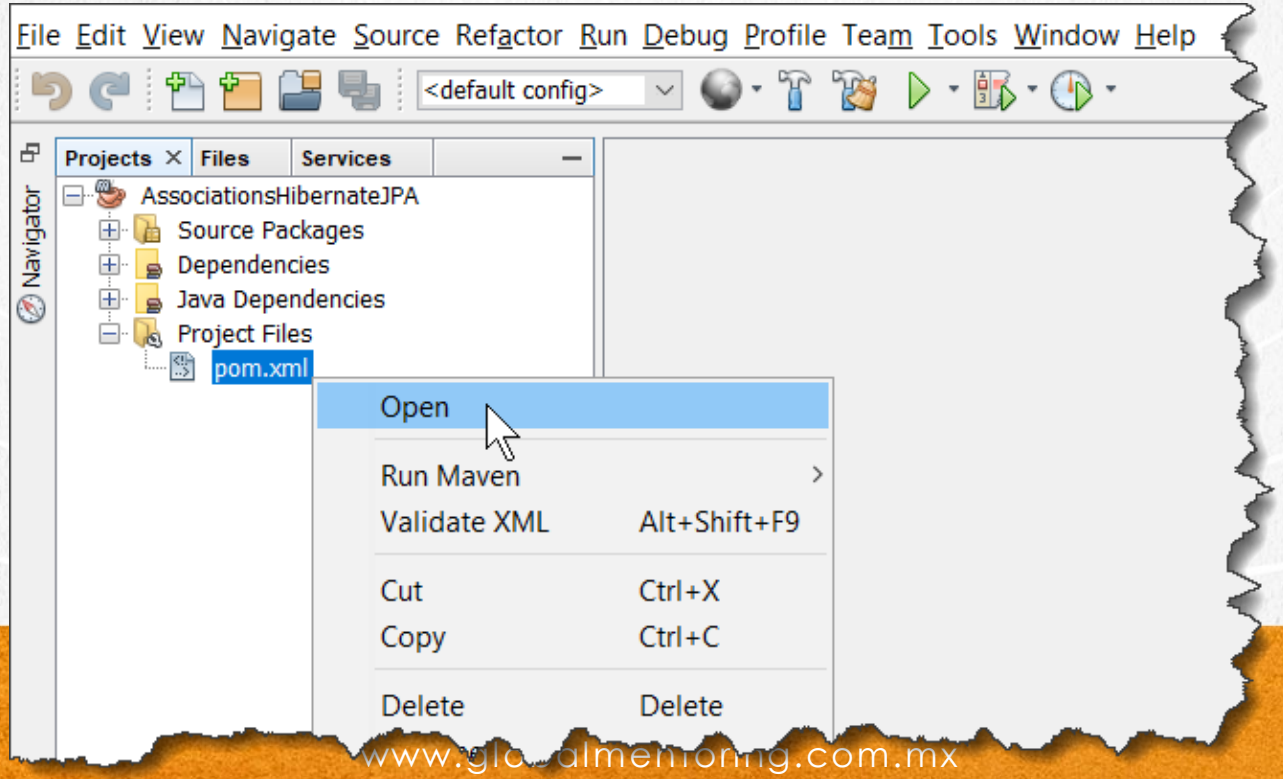
1. CREATE A NEW PROJECT

Create a new Project:



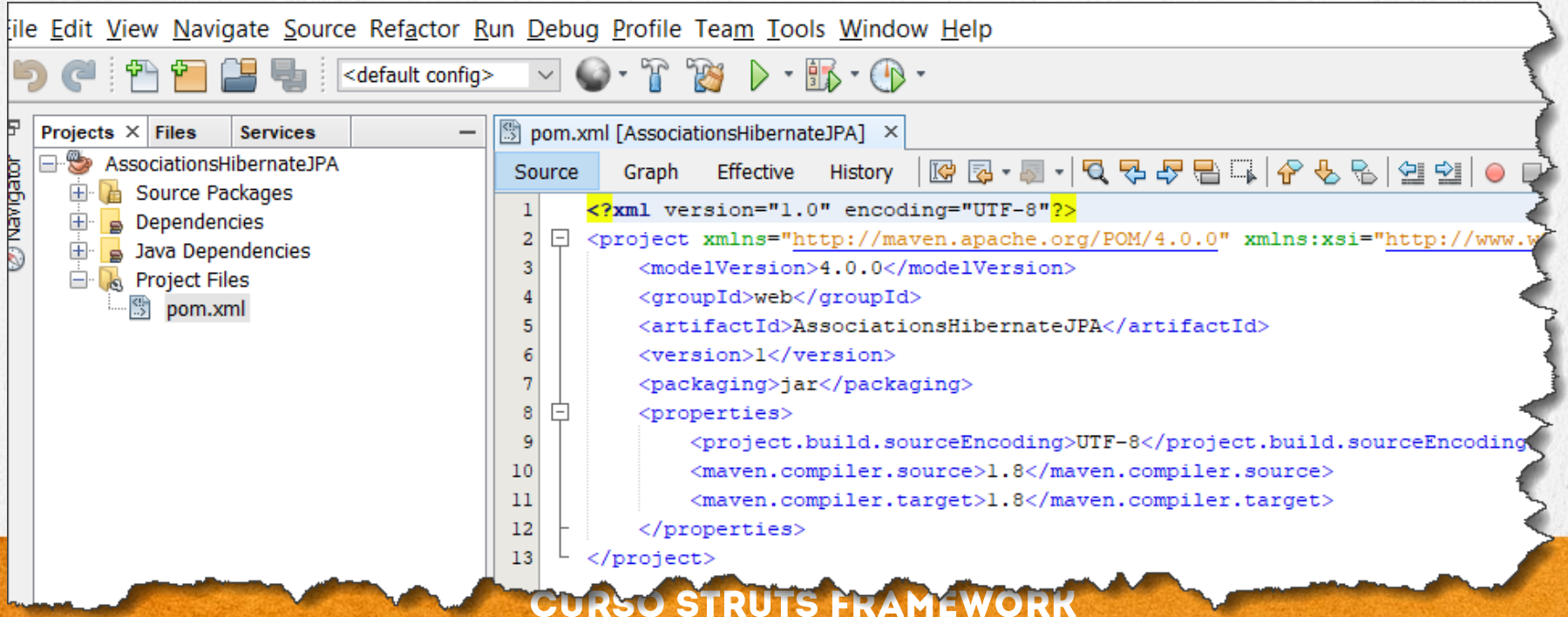
2. OPEN MAVEN'S POM.XML FILE

- The maven pom.xml file manages the Java libraries we are going to use. We open the pom.xml file to modify it with the following code:



2. OPEN MAVEN'S POM.XML FILE

- Once opened, we will modify the information completely of this file, with the information provided below:



3. MODIFY THE CODE

pom.xml:

Click to download

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>web</groupId>
    <artifactId>AssociationsHibernateJPA</artifactId>
    <version>1</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.3.6.Final</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.46</version>
        </dependency>
```

3. MODIFY THE CODE

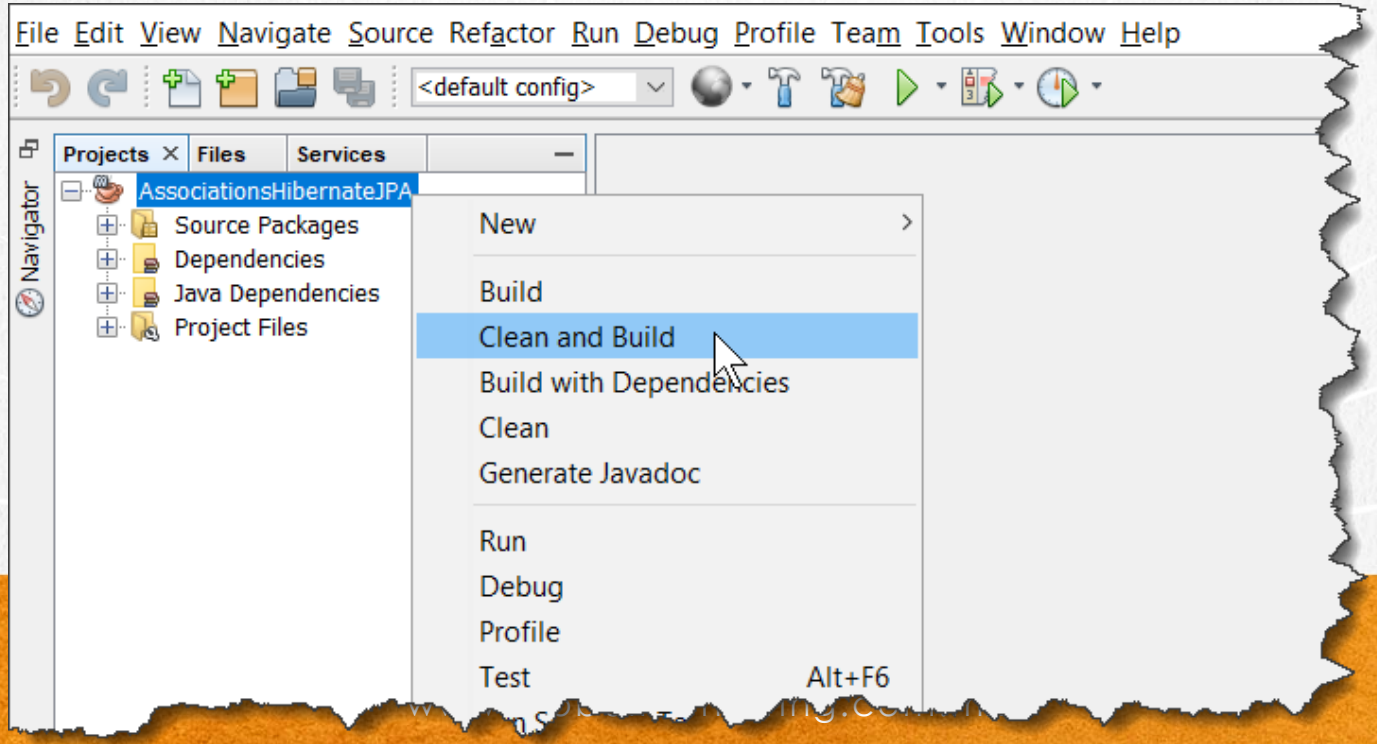
pom.xml:

Click to download

```
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-api</artifactId>
            <version>2.11.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.11.1</version>
        </dependency>
    </dependencies>
</project>
```

4. EXECUTE CLEAN & BUILD

- To download the libraries, we make Clean & Build the project. If for some reason this process fails, you must disable any software such as antivirus, Windows defender or firewall during this process so that the download of Java .jar files is not prevented. Once finished, these services can be activated again. This process may take several minutes depending on your internet speed:



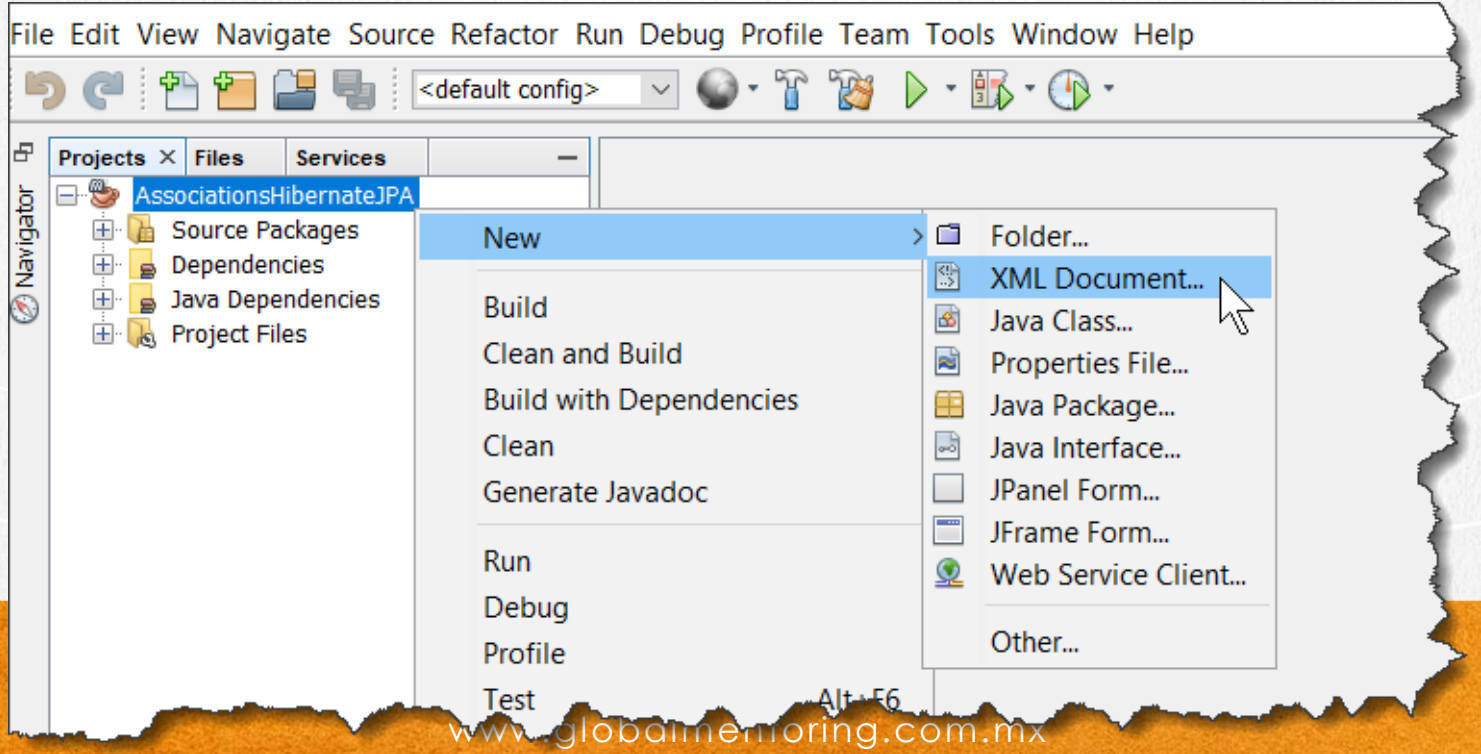
4. EXECUTE CLEAN & BUILD

- Once the process is finished, an output similar to the following should be shown:

```
Output - Build (AssociationsHibernateJPA)
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

--- maven-jar-plugin:2.3.2:jar (default-jar) @ AssociationsHibernateJPA ---
JAR will be empty - no content was marked for inclusion!
Building jar: C:\Courses\Hibernate\Lesson04\AssociationsHibernateJPA\target\AssociationsHibernateJPA-1.jar

--- maven-install-plugin:2.3.1:install (default-install) @ AssociationsHibernateJPA ---
Installing C:\Courses\Hibernate\Lesson04\AssociationsHibernateJPA\target\AssociationsHibernateJPA-1.jar to C:\Us
Installing C:\Courses\Hibernate\Lesson04\AssociationsHibernateJPA\pom.xml to C:\Users\user\.m2\repository\web\As
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 3.539s
Finished at: Tue Sep 11 20:50:24
Final Memory: 10M/489M
------------------------------------------------------------------------
```

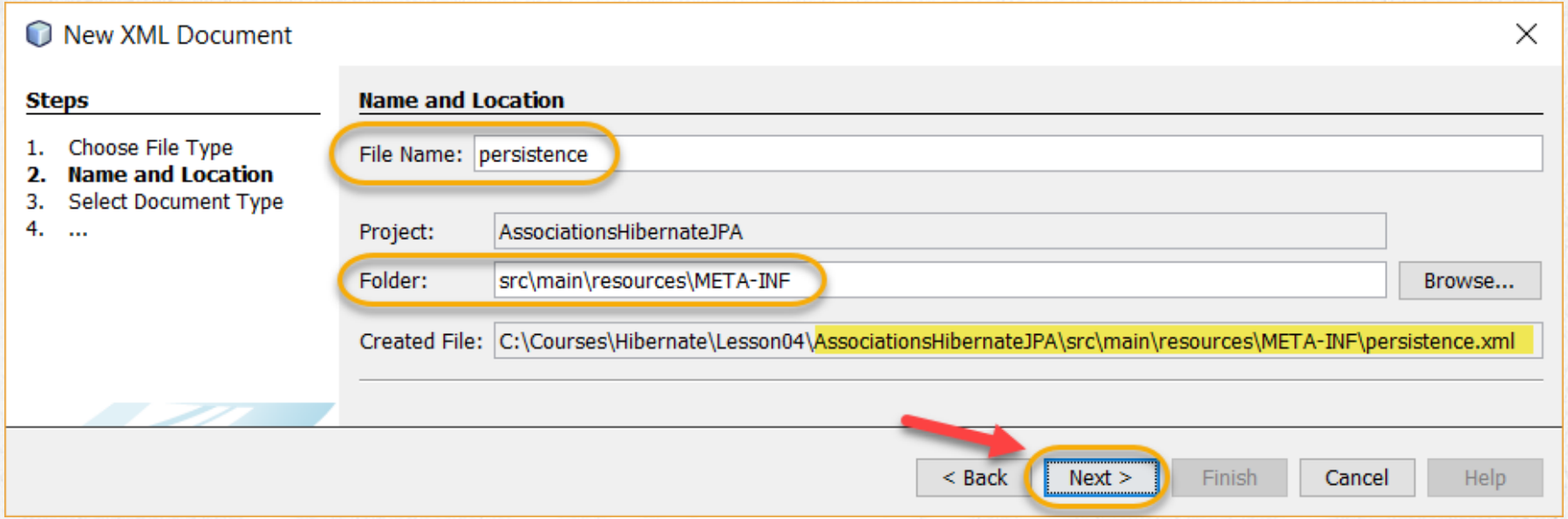
5. CREATE AN XML FILE

- We create the persistence.xml file. The persistence.xml file has the configuration of connection to the database, in this case MySql, among other values, such as the entity classes that we are going to use in the project, and the provider of the Entity Manager:



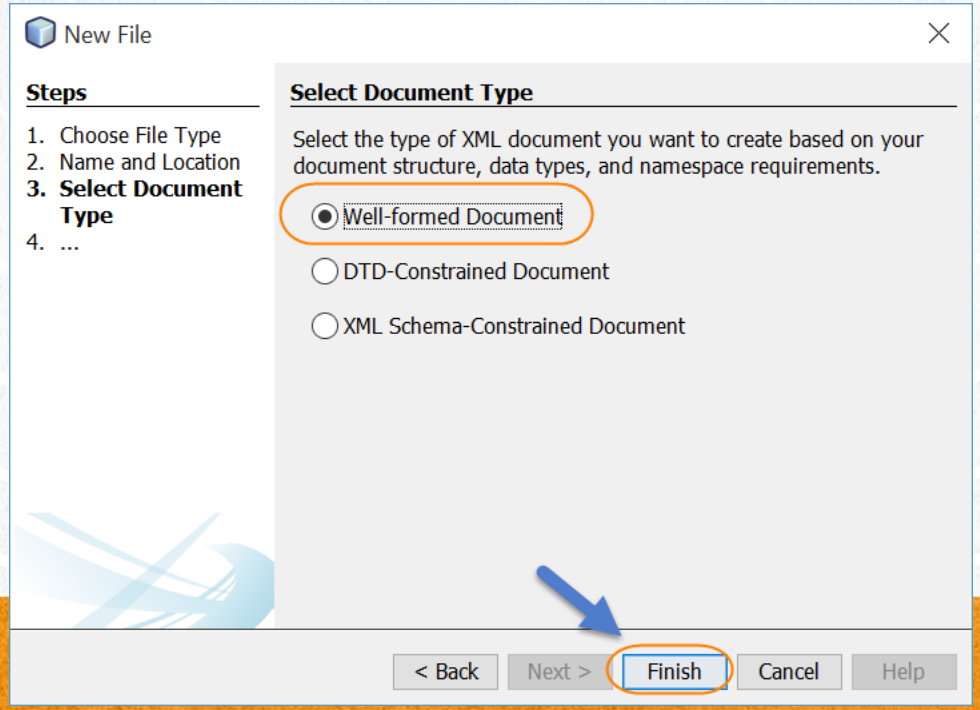
5. CREATE AN XML FILE

- We create the persistence.xml file. We deposit it in the indicated path:



5. CREATE AN XML FILE

- In this step we select any option, it is not important since we are going to overwrite the file:



6. MODIFY THE FILE

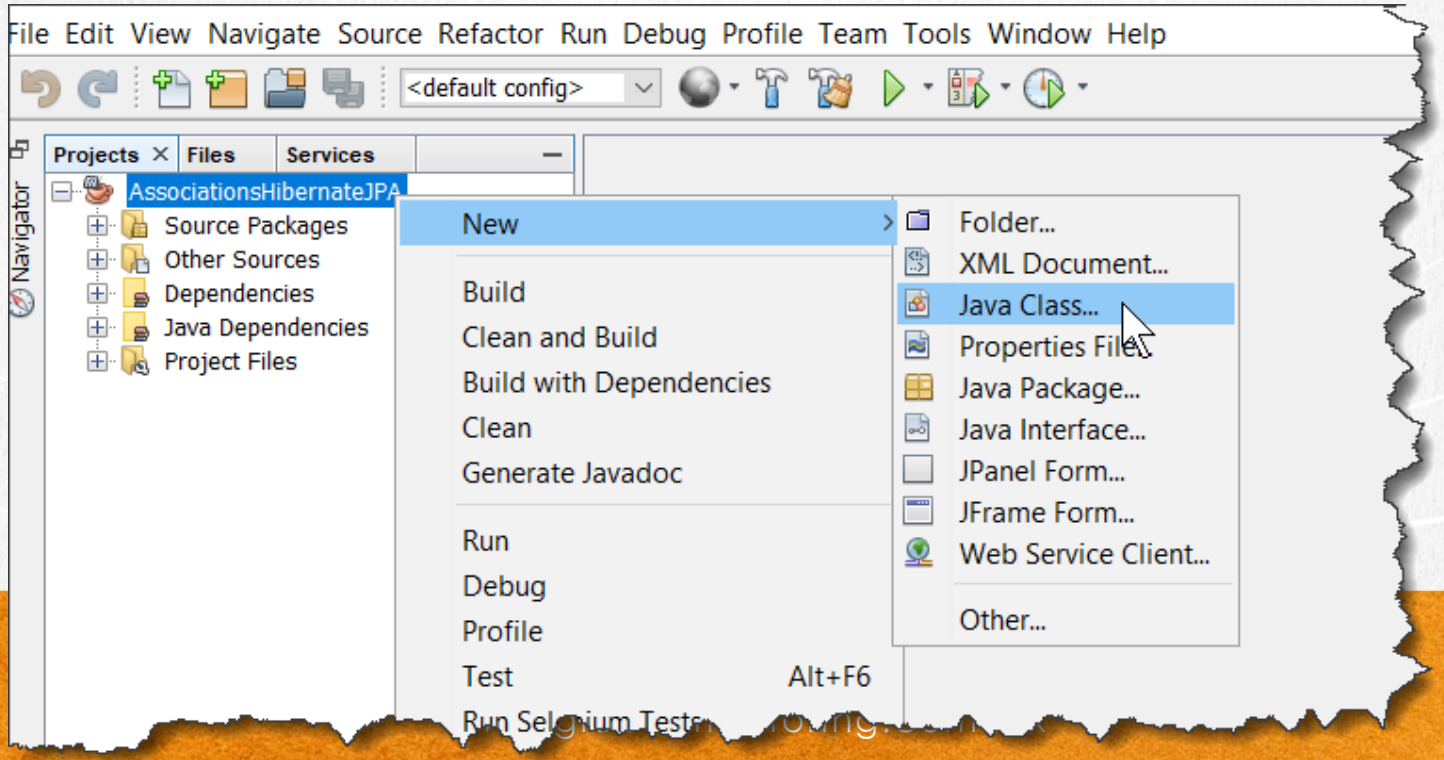
persistence.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd"
             version="2.2">
    <persistence-unit name="HibernateJpaPU" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <class>model.Course</class>
        <class>model.Student</class>
        <class>model.User</class>
        <class>model.Address</class>
        <class>model.Assignation</class>
        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/sms_db?useSSL=false"/>
            <property name="javax.persistence.jdbc.user" value="root"/>
            <property name="javax.persistence.jdbc.password" value="admin"/>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
             <property name="hibernate.show_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>
```

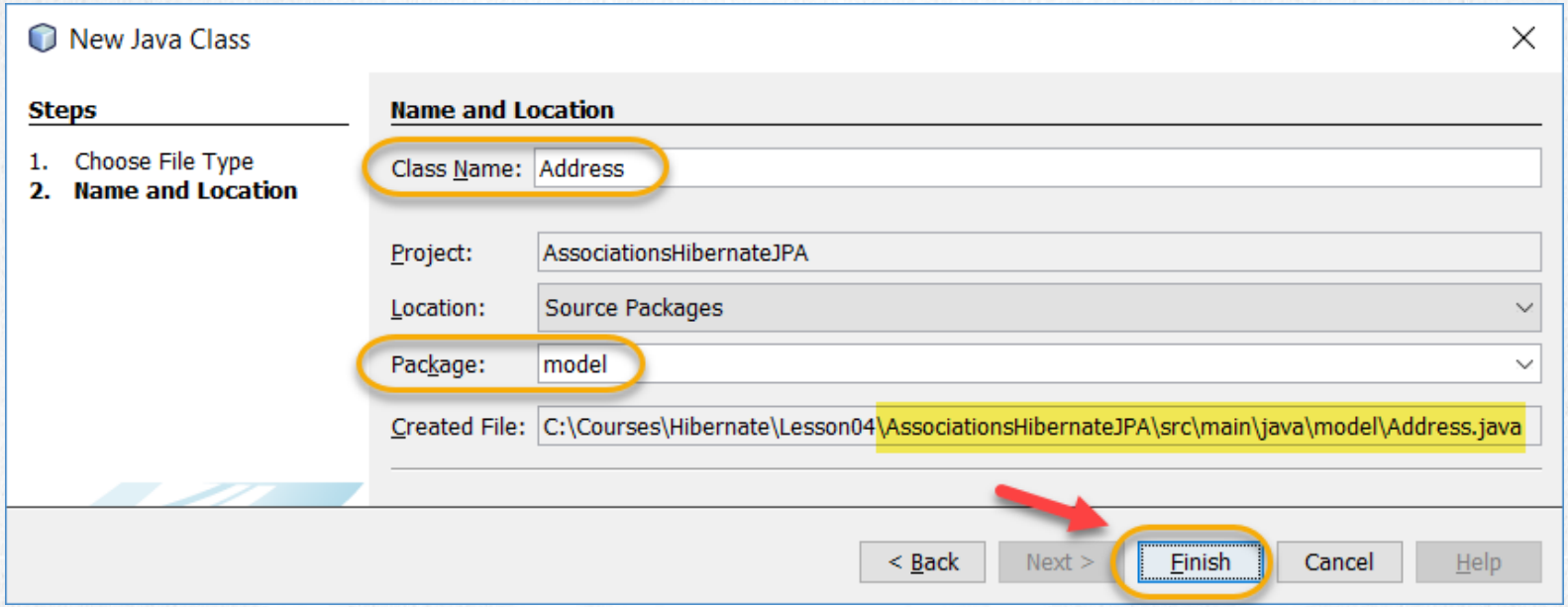
7. CREATION OF ENTITY CLASSES

Create each of the following Entity classes. The procedure to create a class is the same for all of them:



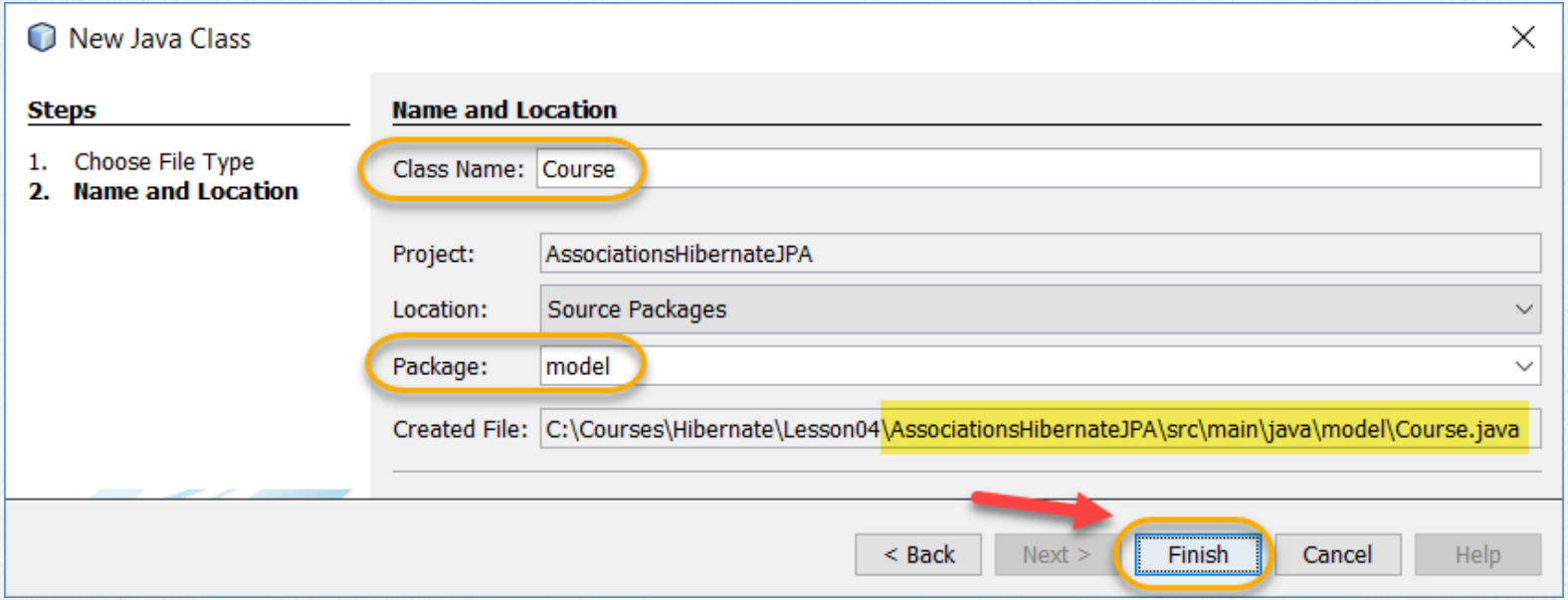
8. CREATE A NEW CLASS

We add a class to the project:



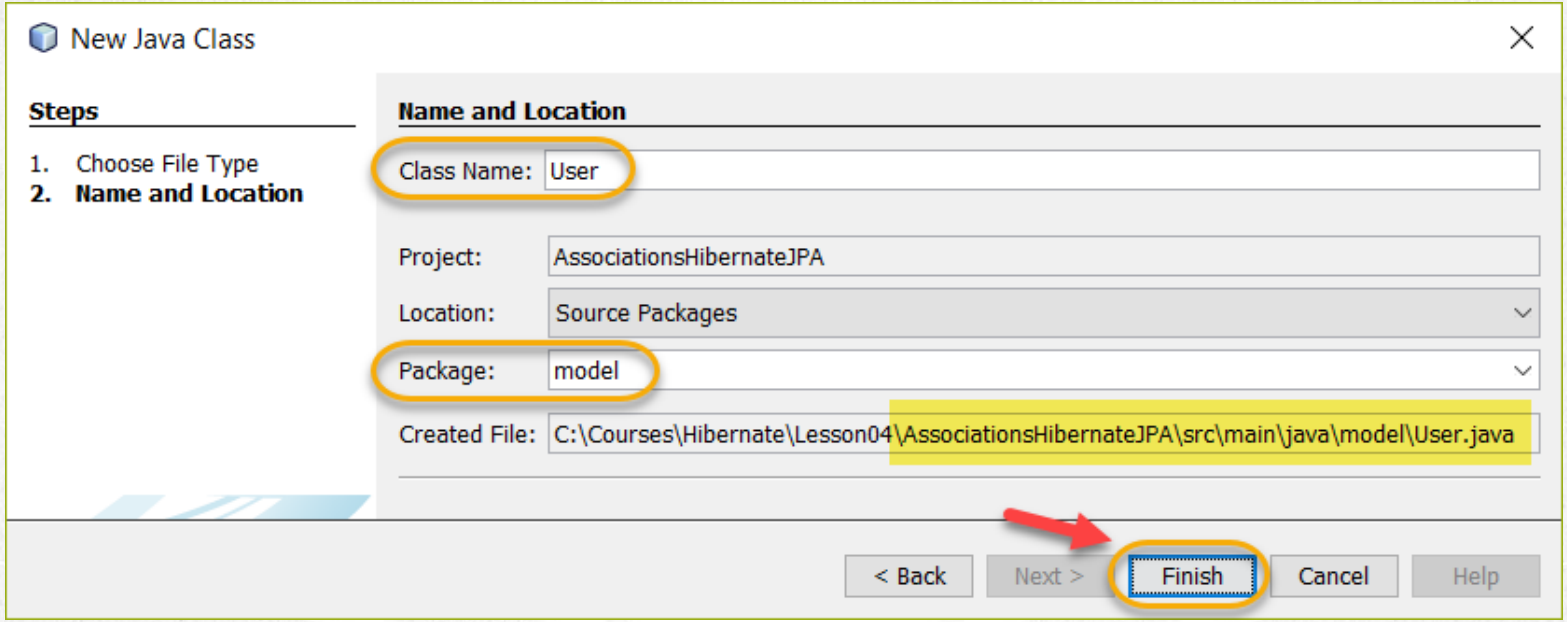
9. CREATE A NEW CLASS

We add a class to the project:



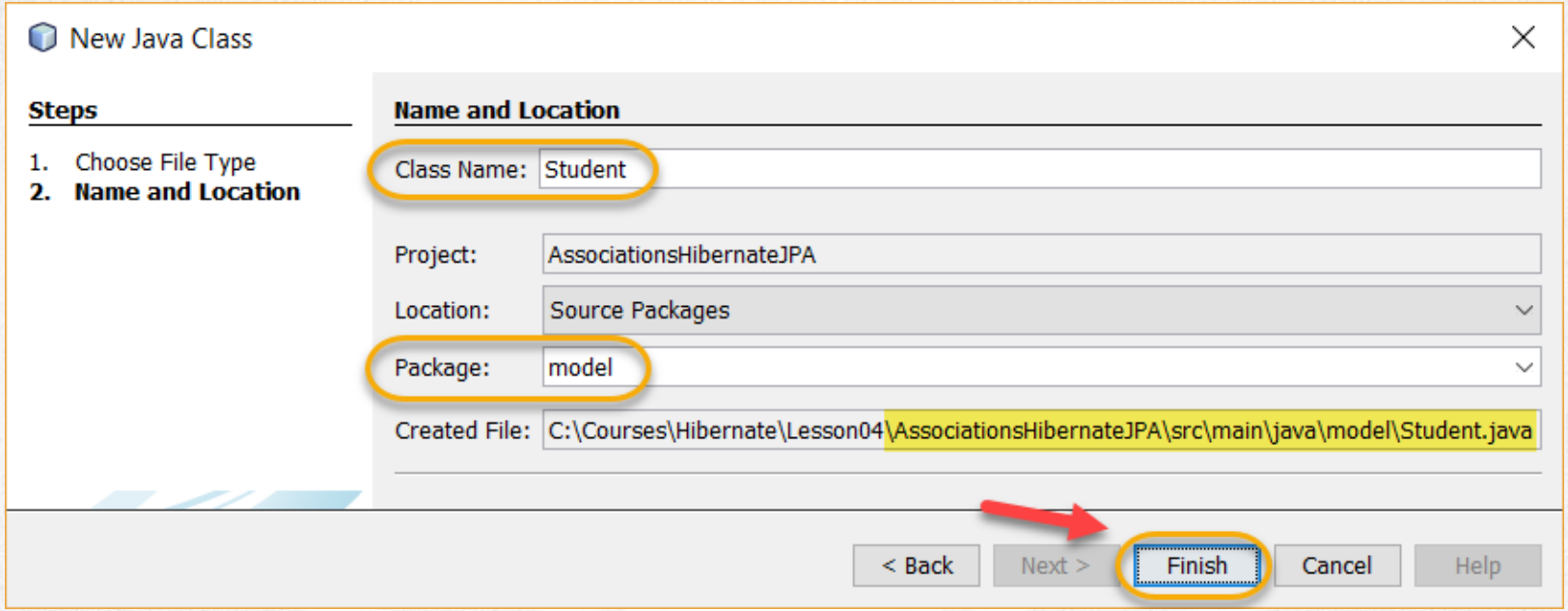
10. CREATE A NEW CLASS

We add a class to the project:



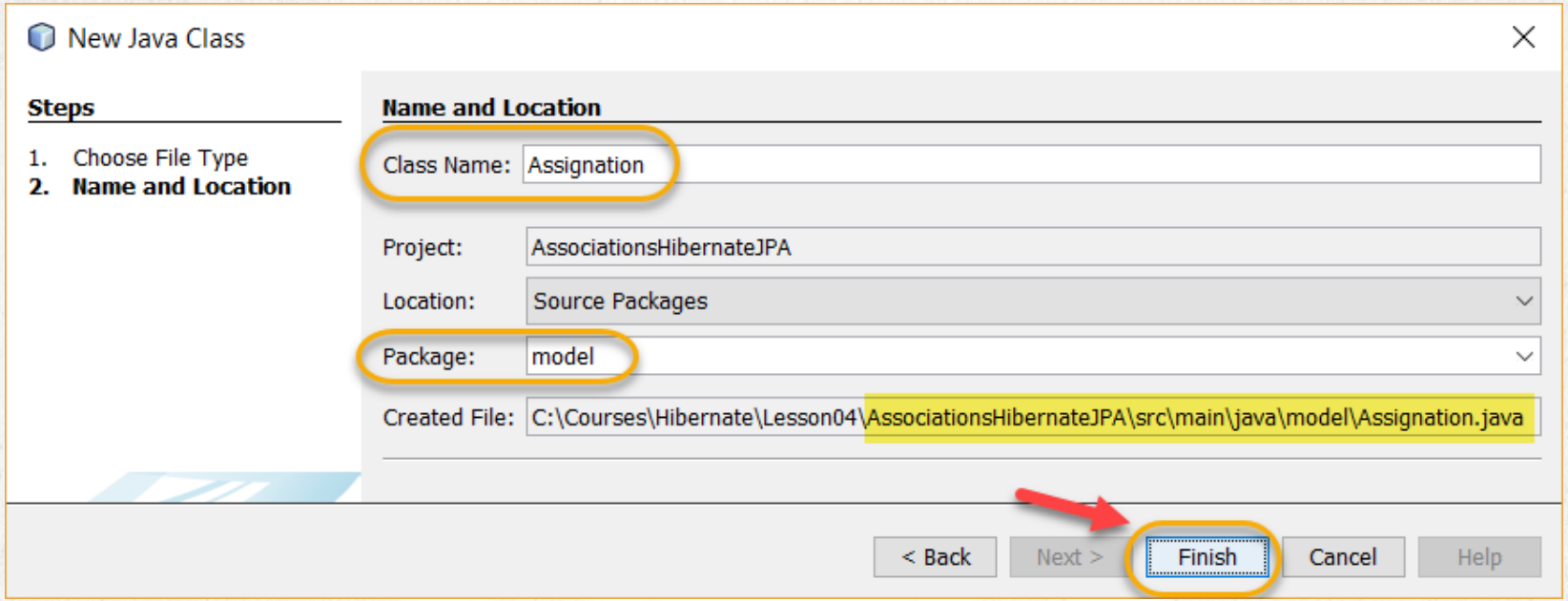
11. CREATE A NEW CLASS

We add a class to the project:



12. CREATE A NEW CLASS

We add a class to the project:



13. MODIFY THE CODE

Address.java:

Click to download

```
package model;

import java.io.Serializable;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name = "address")
@NamedQueries({
    @NamedQuery(name = "Address.findAll", query = "SELECT a FROM Address a")})
public class Address implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_address")
    private Integer idAddress;

    @Column(name = "street_name")
    private String streetName;

    @Column(name = "street_number")
    private String streetNumber;
```

13. MODIFY THE CODE

Address.java:

Click to download

```
    private String country;

    private Integer version = 0;

    private Integer deleted = 0;

    @OneToMany(mappedBy = "address")
    private List<Student> studentList;

    public Address() {
    }

    public Address(Integer idAddress) {
        this.idAddress = idAddress;
    }

    public Address(Integer idAddress, String streetName) {
        this.idAddress = idAddress;
        this.streetName = streetName;
    }

    public Integer getIdAddress() {
        return idAddress;
    }
```

13. MODIFY THE CODE

Address.java:

Click to download

```
    public void setIdAddress(Integer idAddress) {
        this.idAddress = idAddress;
    }

    public String getStreetName() {
        return streetName;
    }

    public void setStreetName(String streetName) {
        this.streetName = streetName;
    }

    public String getStreetNumber() {
        return streetNumber;
    }

    public void setStreetNumber(String streetNumber) {
        this.streetNumber = streetNumber;
    }

    public String getCountry() {
        return country;
    }
```

13. MODIFY THE CODE

Address.java:

Click to download

```
    public void setCountry(String country) {
        this.country = country;
    }

    public Integer getVersion() {
        return version;
    }

    public void setVersion(Integer version) {
        this.version = version;
    }

    public Integer getDeleted() {
        return deleted;
    }

    public void setDeleted(Integer deleted) {
        this.deleted = deleted;
    }

    public List<Student> getStudentList() {
        return studentList;
    }
```

13. MODIFY THE CODE

Address.java:

Click to download

```
    public void setStudentList(List<Student> studentList) {
        this.studentList = studentList;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idAddress != null ? idAddress.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Address)) {
            return false;
        }
        Address other = (Address) object;
        if ((this.idAddress == null && other.idAddress != null) || (this.idAddress != null &&
!this.idAddress.equals(other.idAddress))) {
            return false;
        }
        return true;
    }
```

13. MODIFY THE CODE

Address.java:

Click to download

```
    @Override
    public String toString() {
        return "Address{" + "idAddress=" + idAddress + ", streetName=" + streetName + ", streetNumber=" +
streetNumber + ", country=" + country + ", version=" + version + ", deleted=" + deleted + '}';
    }



}
```

14. MODIFY THE FILE

Course.java:

Click to download

```java
package model;

import java.io.Serializable;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name = "course")
@NamedQueries({
    @NamedQuery(name = "Course.findAll", query = "SELECT c FROM Course c")})
public class Course implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_course")
    private Integer idCourse;

    private String name;

    private String schedule;

    private Float price;
```

14. MODIFY THE FILE

Course.java:

Click to download

```
    private int version = 0;

    private int deleted = 0;

    @OneToMany(mappedBy = "course")
    private List<Assignation> asignacionList;

    public Course() {
    }

    public Course(Integer idCourse) {
        this.idCourse = idCourse;
    }

    public Course(Integer idCourse, String name, int version, int deleted) {
        this.idCourse = idCourse;
        this.name = name;
        this.version = version;
        this.deleted = deleted;
    }

    public Integer getIdCourse() {
        return idCourse;
    }
```

14. MODIFY THE FILE

Course.java:

Click to download

```
    public void setIdCourse(Integer idCourse) {
        this.idCourse = idCourse;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSchedule() {
        return schedule;
    }

    public void setSchedule(String schedule) {
        this.schedule = schedule;
    }

    public Float getPrice() {
        return price;
    }
```

14. MODIFY THE FILE

Course.java:

Click to download

```
    public void setPrice(Float price) {
        this.price = price;
    }

    public int getVersion() {
        return version;
    }

    public void setVersion(int version) {
        this.version = version;
    }

    public int getDeleted() {
        return deleted;
    }

    public void setDeleted(int deleted) {
        this.deleted = deleted;
    }

    public List<Assignation> getAsignacionList() {
        return asignacionList;
    }
```

14. MODIFY THE FILE

Course.java:

Click to download

```
    public void setAsignacionList(List<Assignation> asignacionList) {
        this.asignacionList = asignacionList;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idCourse != null ? idCourse.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Course)) {
            return false;
        }
        Course other = (Course) object;
        if ((this.idCourse == null && other.idCourse != null) || (this.idCourse != null &&
!this.idCourse.equals(other.idCourse))) {
            return false;
        }
        return true;
    }
```

14. MODIFY THE FILE

Course.java:

Click to download

```
    @Override
    public String toString() {
        return "Course{" + "idCourse=" + idCourse + ", name=" + name + ", schedule=" + schedule + ",
price=" + price + ", version=" + version + ", deleted=" + deleted + '}';
    }

}
```

15. MODIFY THE FILE

User.java:

Click to download

```
package model;

import java.io.Serializable;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name = "user")
@NamedQueries({
    @NamedQuery(name = "User.findAll", query = "SELECT u FROM User u")})
public class User implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_user")
    private Integer idUser;

    private String username;

    private String password;

    private int version = 0;
```

15. MODIFY THE FILE

User.java:

Click to download

```
    private int deleted = 0;

    @OneToMany(mappedBy = "user")
    private List<Student> studentList;

    public User() {
    }

    public User(Integer idUser) {
        this.idUser = idUser;
    }

    public User(Integer idUser, String username, String password, int version, int deleted) {
        this.idUser = idUser;
        this.username = username;
        this.password = password;
        this.version = version;
        this.deleted = deleted;
    }

    public Integer getIdUser() {
        return idUser;
    }
```

15. MODIFY THE FILE

User.java:

Click to download

```
    public void setIdUser(Integer idUser) {
        this.idUser = idUser;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public int getVersion() {
        return version;
    }
```

15. MODIFY THE FILE

User.java:

Click to download

```
    public void setVersion(int version) {
        this.version = version;
    }

    public int getDeleted() {
        return deleted;
    }

    public void setDeleted(int deleted) {
        this.deleted = deleted;
    }

    public List<Student> getStudentList() {
        return studentList;
    }

    public void setStudentList(List<Student> studentList) {
        this.studentList = studentList;
    }
```

15. MODIFY THE FILE

User.java:

Click to download

```
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idUser != null ? idUser.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof User)) {
            return false;
        }
        User other = (User) object;
        if ((this.idUser == null && other.idUser != null) || (this.idUser != null &&
!this.idUser.equals(other.idUser))) {
            return false;
        }
        return true;
    }
```

15. MODIFY THE FILE

User.java:

Click to download

```
    @Override
    public String toString() {
        return "User{" + "idUser=" + idUser + ", username=" + username + ", password=" + password + ",
version=" + version + ", deleted=" + deleted + '}';
    }


}
```

16. MODIFY THE CODE

Student.java:

Click to download

```
package model;

import java.io.Serializable;
import java.util.*;
import javax.persistence.*;

@Entity
@Table(name = "student")
@NamedQueries({
    @NamedQuery(name = "Student.findAll", query = "SELECT s FROM Student s")})
public class Student implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_student")
    private Integer idStudent;

    private String name;

    private int version = 0;

    private int deleted = 0;
```

16. MODIFY THE CODE

Student.java:

Click to download

```
@OneToMany(mappedBy = "student")
private List<Assignation> assignationList;

@JoinColumn(name = "id_address", referencedColumnName = "id_address")
@ManyToOne
private Address address;

@JoinColumn(name = "id_user", referencedColumnName = "id_user")
@ManyToOne
private User user;

public Student() {
}

public Student(Integer idStudent) {
    this.idStudent = idStudent;
}

public Student(Integer idStudent, String name, int version, int deleted) {
    this.idStudent = idStudent;
    this.name = name;
    this.version = version;
    this.deleted = deleted;
}
```

16. MODIFY THE CODE

Student.java:

Click to download

```
    public Integer getIdStudent() {
        return idStudent;
    }

    public void setIdStudent(Integer idStudent) {
        this.idStudent = idStudent;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public int getVersion() {
        return version;
    }

    public void setVersion(int version) {
        this.version = version;
    }
```

16. MODIFY THE CODE

Student.java:

Click to download

```
    public int getDeleted() {
        return deleted;
    }

    public void setDeleted(int deleted) {
        this.deleted = deleted;
    }

    public List<Assignation> getAssignationList() {
        return assignationList;
    }

    public void setAssignationList(List<Assignation> assignationList) {
        this.assignationList = assignationList;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }
```

16. MODIFY THE CODE

Student.java:

Click to download

```
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idStudent != null ? idStudent.hashCode() : 0);
        return hash;
    }
```

16. MODIFY THE CODE

Student.java:

Click to download

```
    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Student)) {
            return false;
        }
        Student other = (Student) object;
        if ((this.idStudent == null && other.idStudent != null) || (this.idStudent != null &&
!this.idStudent.equals(other.idStudent))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Student{" + "idStudent=" + idStudent + ", name=" + name + ", version=" + version + ",
deleted=" + deleted + ", address=" + address + ", user=" + user + '}';
    }

}
```

17. MODIFY THE CODE

Assignation.java:

Click to download

```
package model;

import java.io.Serializable;
import java.util.*;
import javax.persistence.*;

@Entity
@Table(name = "student")
@NamedQueries({
    @NamedQuery(name = "Student.findAll", query = "SELECT s FROM Student s")})
public class Student implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_student")
    private Integer idStudent;

    private String name;

    private int version = 0;

    private int deleted = 0;
```

17. MODIFY THE CODE

Assignation.java:

Click to download

```
@OneToMany(mappedBy = "student")
private List<Assignation> assignationList;

@JoinColumn(name = "id_address", referencedColumnName = "id_address")
@ManyToOne
private Address address;

@JoinColumn(name = "id_user", referencedColumnName = "id_user")
@ManyToOne
private User user;

public Student() {
}

public Student(Integer idStudent) {
    this.idStudent = idStudent;
}

public Student(Integer idStudent, String name, int version, int deleted) {
    this.idStudent = idStudent;
    this.name = name;
    this.version = version;
    this.deleted = deleted;
}
```

17. MODIFY THE CODE

Assignation.java:

Click to download

```
    public Integer getIdStudent() {
        return idStudent;
    }

    public void setIdStudent(Integer idStudent) {
        this.idStudent = idStudent;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public int getVersion() {
        return version;
    }

    public void setVersion(int version) {
        this.version = version;
    }
```

17. MODIFY THE CODE

Assignation.java:

Click to download

```
    public int getDeleted() {
        return deleted;
    }

    public void setDeleted(int deleted) {
        this.deleted = deleted;
    }

    public List<Assignation> getAssignationList() {
        return assignationList;
    }

    public void setAssignationList(List<Assignation> assignationList) {
        this.assignationList = assignationList;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }
```

17. MODIFY THE CODE

Assignation.java:

Click to download

```
public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (idStudent != null ? idStudent.hashCode() : 0);
    return hash;
}
```

17. MODIFY THE CODE

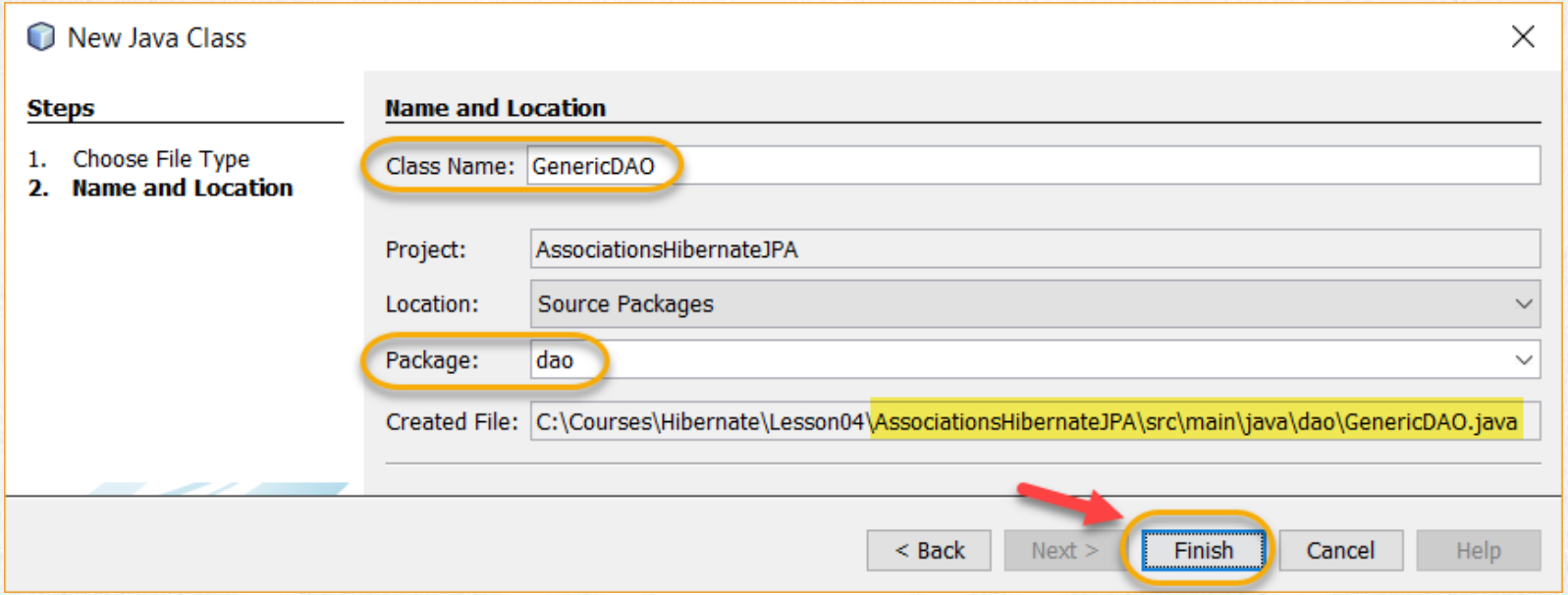
Assignation.java:

Click to download

```
    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Student)) {
            return false;
        }
        Student other = (Student) object;
        if ((this.idStudent == null && other.idStudent != null) || (this.idStudent != null &&
!this.idStudent.equals(other.idStudent))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Student{" + "idStudent=" + idStudent + ", name=" + name + ", version=" + version + ",
deleted=" + deleted + ", address=" + address + ", user=" + user + '}';
    }

}
```

18. CREATE A NEW CLASS

We add a class to the project:



19. MODIFY THE CODE

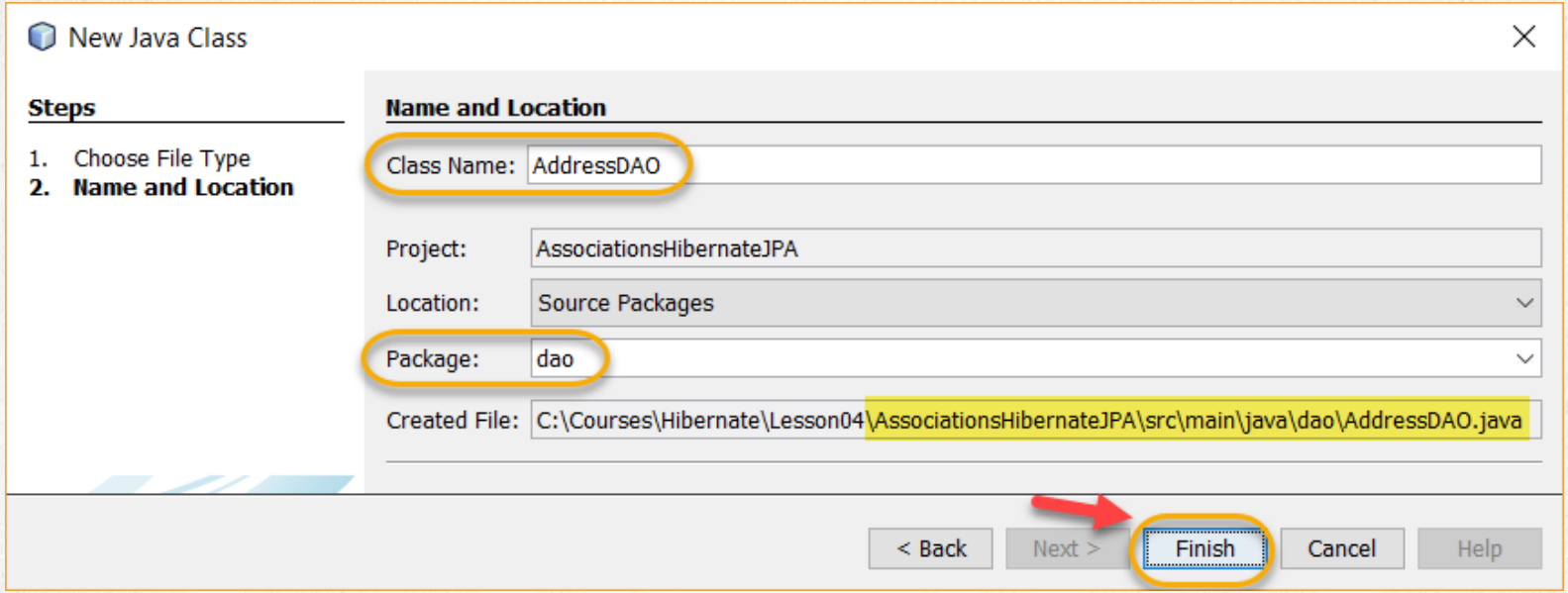
GenericDAO.java:

Click to download

```java
package dao;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public abstract class GenericDAO {

    protected static EntityManager em;
    private static EntityManagerFactory emf;
    private static final String PERSISTENCE_UNIT_NAME = "HibernateJpaPU";

    public GenericDAO() {
        if (emf == null) {
            emf = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
        }
    }

    protected EntityManager getEntityManager() {
        if( em == null){
            em = emf.createEntityManager();
        }
        return em;
    }
}
```

20. CREATE A NEW CLASS

We add a class to the project:



21. MODIFY THE FILE

AddressDAO.java:

Click to download

```
package dao;

import static dao.GenericDAO.em;
import java.util.List;
import javax.persistence.Query;
import model.Address;

public class AddressDAO extends GenericDAO{

      public List<Address> list() {
        String hql = "SELECT a FROM Address a";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        List<Address> list = query.getResultList();
        for(Address a : list){
            System.out.println(a);
        }
        return list;
    }
```

21. MODIFY THE FILE

AddressDAO.java:

Click to download

```
public void insert(Address address) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.persist(address);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error inserting object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

21. MODIFY THE FILE

AddressDAO.java:

Click to download

```
public void update(Address address) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.merge(address);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error updating object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

21. MODIFY THE FILE

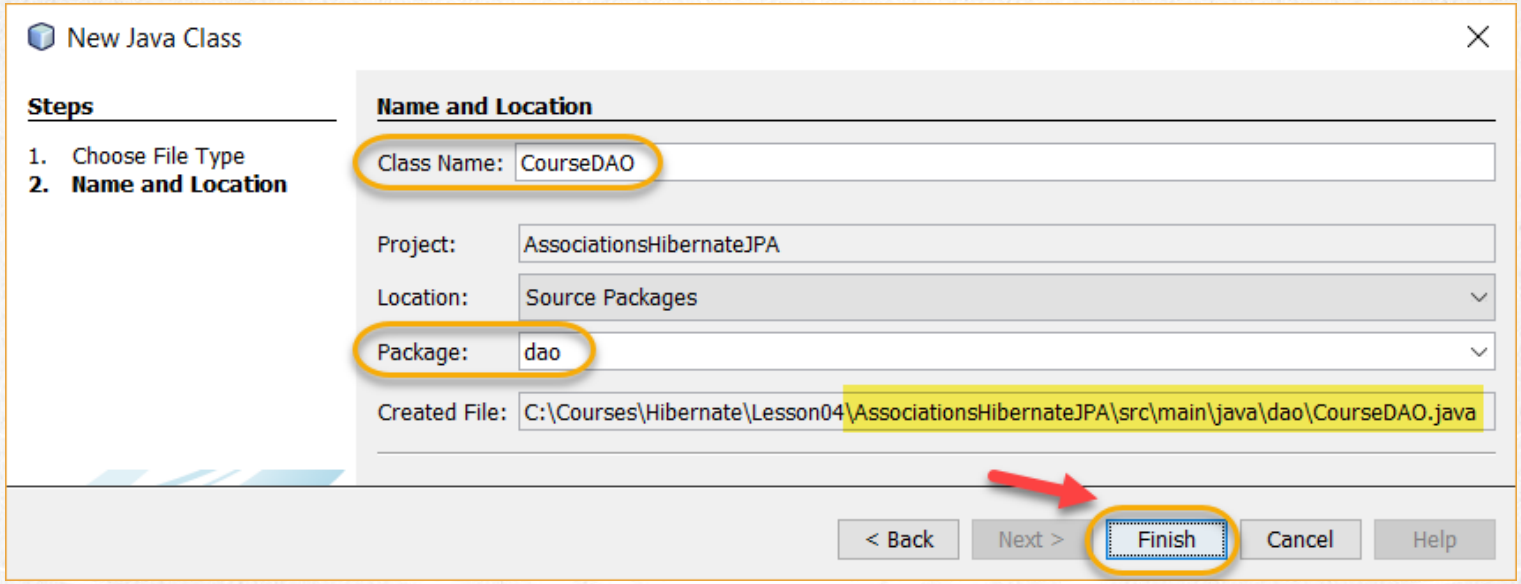
AddressDAO.java:

Click to download

```
    public void delete(Address address) {
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.remove(em.merge(address));
            em.getTransaction().commit();
        } catch (Exception ex) {
            System.out.println("Error removing object:" + ex.getMessage());
        } finally {
            if (em != null) {
                em.close();
                em = null;
            }
        }
    }

    public Address findById(Address address) {
        em = getEntityManager();
        return em.find(Address.class, address.getIdAddress());
    }
}
```

22. CREATE A NEW CLASS

We add a class to the project:



23. MODIFY THE FILE

CourseDAO.java:

Click to download

```
package dao;

import static dao.GenericDAO.em;
import java.util.List;
import javax.persistence.Query;
import model.Course;

public class CourseDAO extends GenericDAO{

      public List<Course> list() {
        String hql = "SELECT c FROM Course c";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        List<Course> list = query.getResultList();
        for(Course c : list){
            System.out.println(c);
        }
        return list;
    }
```

23. MODIFY THE FILE

CourseDAO.java:

Click to download

```
public void insert(Course course) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.persist(course);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error inserting object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

23. MODIFY THE FILE

CourseDAO.java:

Click to download

```
public void update(Course course) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.merge(course);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error updating object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

23. MODIFY THE FILE

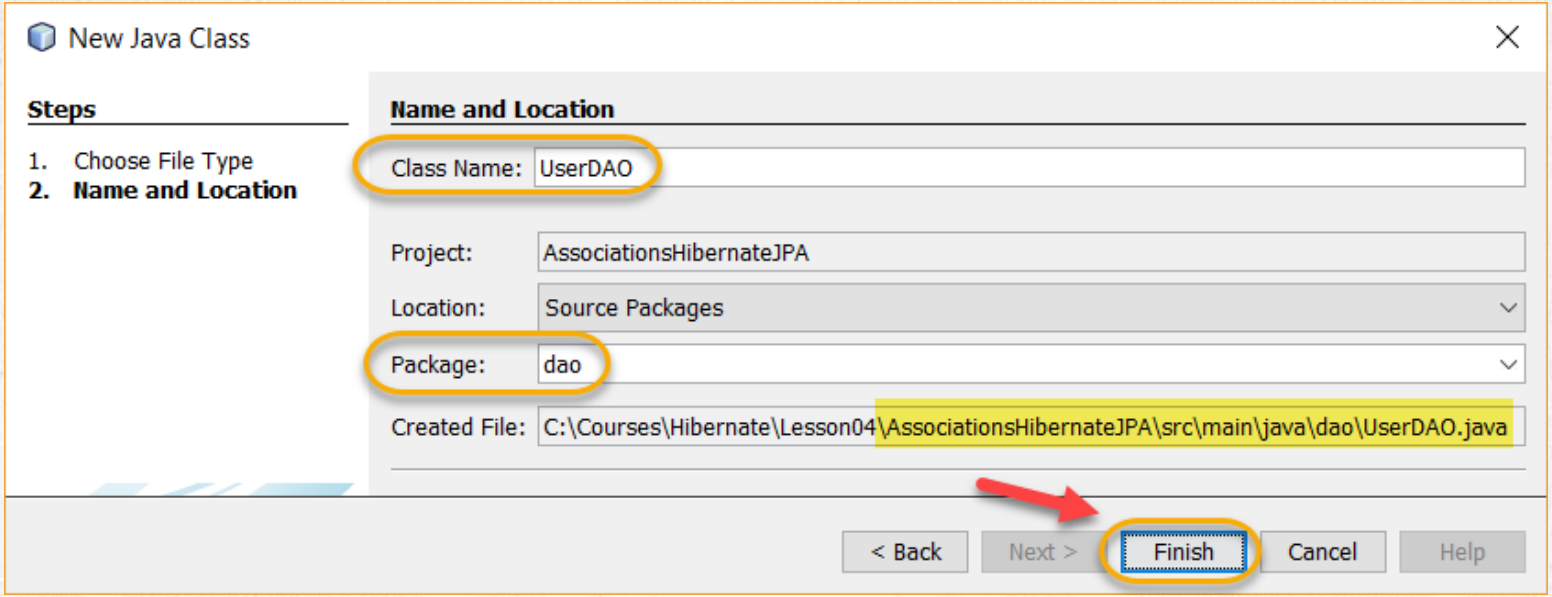
CourseDAO.java:

Click to download

```
    public void delete(Course course) {
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.remove(em.merge(course));
            em.getTransaction().commit();
        } catch (Exception ex) {
            System.out.println("Error removing object:" + ex.getMessage());
        } finally {
            if (em != null) {
                em.close();
                em = null;
            }
        }
    }

    public Course findById(Course course) {
        em = getEntityManager();
        return em.find(Course.class, course.getIdCourse());
    }
}
```

24. CREATE A NEW CLASS

We add a class to the project:



25. MODIFY THE CODE

UserDAO.java:

Click to download

```
package dao;

import static dao.GenericDAO.em;
import java.util.List;
import javax.persistence.Query;
import model.User;

public class UserDAO extends GenericDAO{

      public List<User> list() {
        String hql = "SELECT u FROM User u";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        List<User> list = query.getResultList();
        for(User u : list){
            System.out.println(u);
        }
        return list;
    }
```

25. MODIFY THE CODE

UserDAO.java:

Click to download

```
public void insert(User user) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.persist(user);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error inserting object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

25. MODIFY THE CODE

UserDAO.java:

Click to download

```
public void update(User user) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.merge(user);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error updating object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

25. MODIFY THE CODE

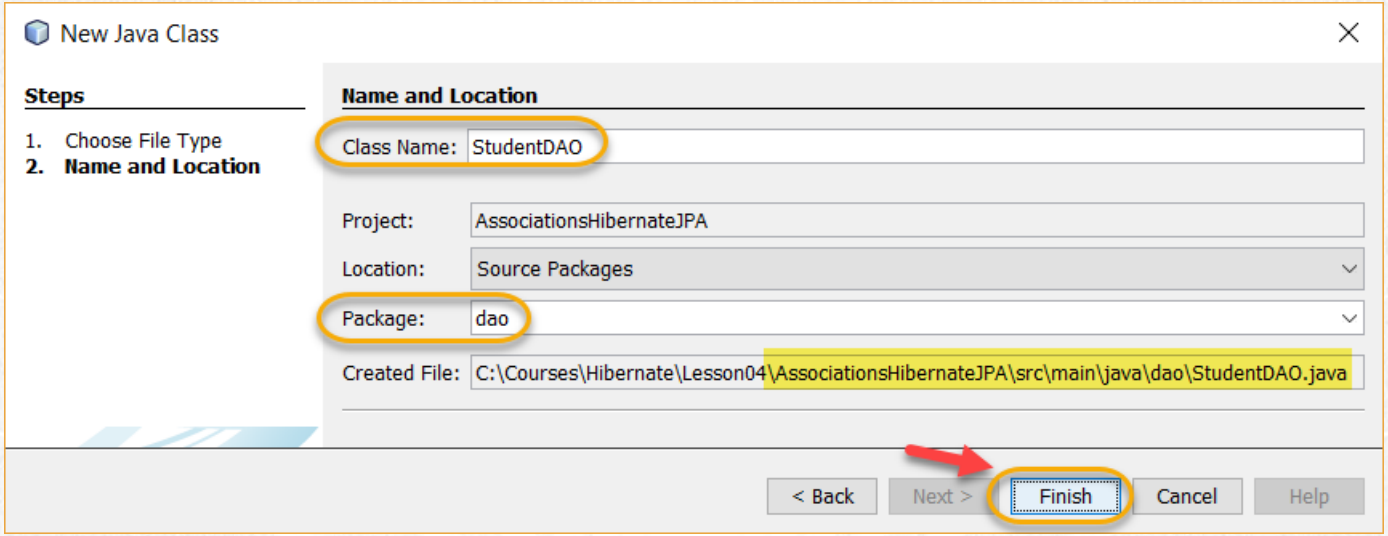
UserDAO.java:

Click to download

```
    public void delete(User user) {
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.remove(em.merge(user));
            em.getTransaction().commit();
        } catch (Exception ex) {
            System.out.println("Error removing object:" + ex.getMessage());
        } finally {
            if (em != null) {
                em.close();
                em = null;
            }
        }
    }

    public User findById(User user) {
        em = getEntityManager();
        return em.find(User.class, user.getIdUser());
    }
}
```

26. CREATE A NEW CLASS

We add a class to the project:



27. MODIFY THE FILE

StudentDAO.java:

Click to download

```
package dao;

import static dao.GenericDAO.em;
import java.util.List;
import javax.persistence.Query;
import model.Student;

public class StudentDAO extends GenericDAO{

      public List<Student> list() {
        String hql = "SELECT s FROM Student s";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        List<Student> list = query.getResultList();
        for(Student s : list){
            System.out.println(s);
        }
        return list;
    }
```

27. MODIFY THE FILE

StudentDAO.java:

Click to download

```
public void insert(Student student) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.persist(student);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error inserting object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

27. MODIFY THE FILE

StudentDAO.java:

Click to download

```
public void update(Student student) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.merge(student);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error updating object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

27. MODIFY THE FILE

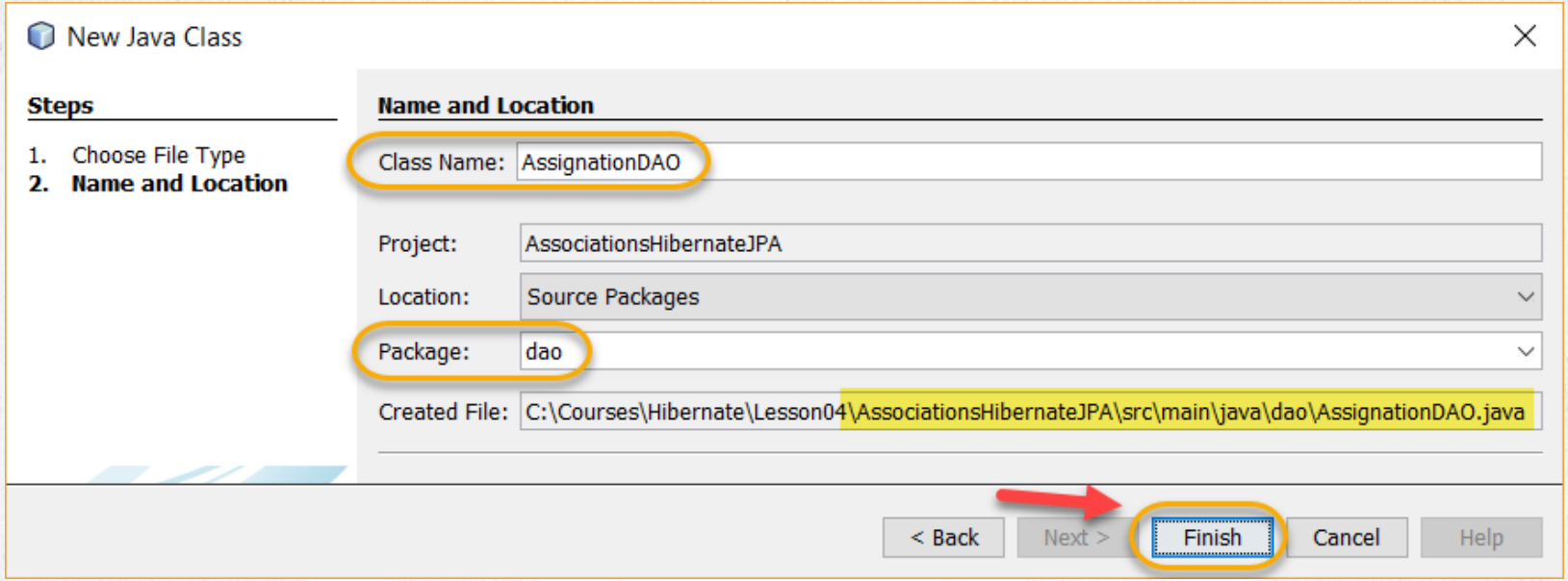
StudentDAO.java:

Click to download

```
    public void delete(Student student) {
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.remove(em.merge(student));
            em.getTransaction().commit();
        } catch (Exception ex) {
            System.out.println("Error removing object:" + ex.getMessage());
        } finally {
            if (em != null) {
                em.close();
                em = null;
            }
        }
    }

    public Student findById(Student student) {
        em = getEntityManager();
        return em.find(Student.class, student.getIdStudent());
    }
}
```

28. CREATE A NEW CLASS

We add a class to the project:



29. MODIFY THE FILE

AssignationDAO.java:

Click to download

```
package dao;

import static dao.GenericDAO.em;
import java.util.List;
import javax.persistence.Query;
import model.Assignation;

public class AssignationDAO extends GenericDAO{

      public List<Assignation> list() {
        String hql = "SELECT a FROM Assignation a";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        List<Assignation> list = query.getResultList();
        for(Assignation a : list){
            System.out.println(a);
        }
        return list;
    }
```

29. MODIFY THE FILE

AssignationDAO.java:

Click to download

```
public void insert(Assignation assignation) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.persist(assignation);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error inserting object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

29. MODIFY THE FILE

AssignationDAO.java:

Click to download

```
public void update(Assignation assignation) {
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        em.merge(assignation);
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error updating object:" + ex.getMessage());
    } finally {
        if (em != null) {
            em.close();
            em = null;
        }
    }
}
```

29. MODIFY THE FILE

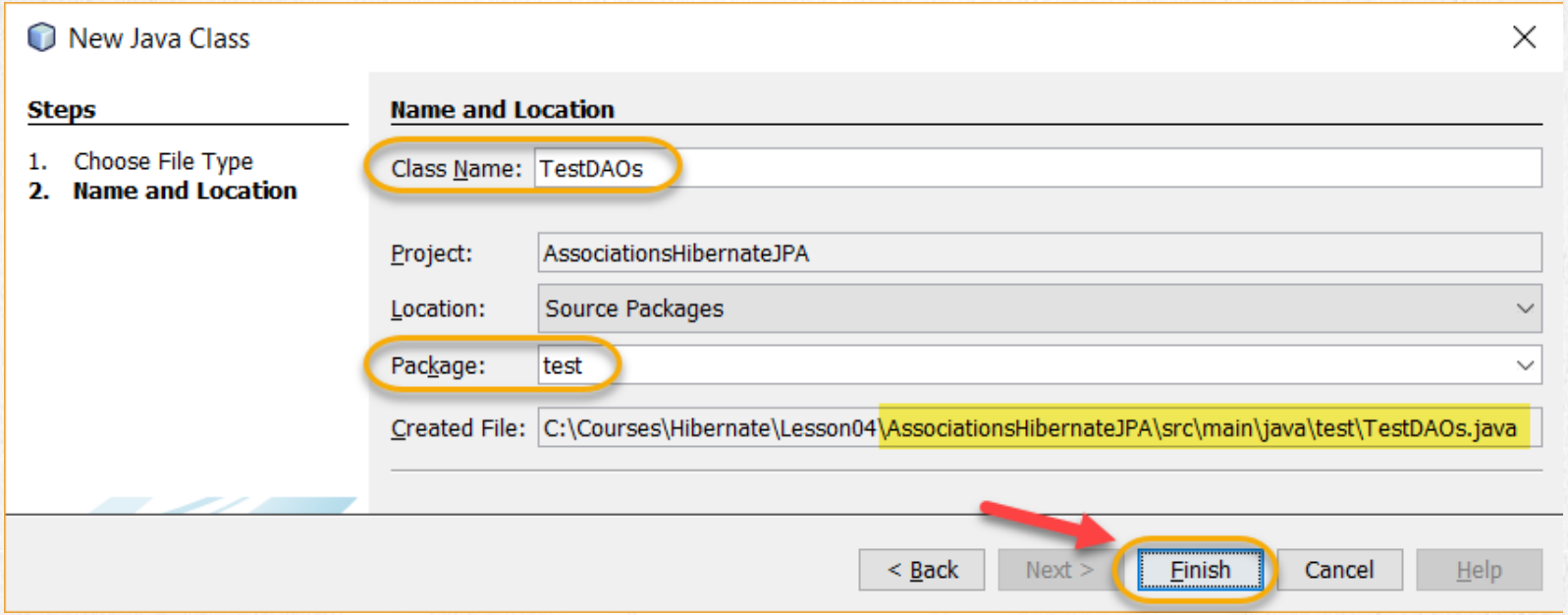
AssignationDAO.java:

Click to download

```
    public void delete(Assignation assignation) {
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.remove(em.merge(assignation));
            em.getTransaction().commit();
        } catch (Exception ex) {
            System.out.println("Error removing object:" + ex.getMessage());
        } finally {
            if (em != null) {
                em.close();
                em = null;
            }
        }
    }

    public Assignation findById(Assignation assignation) {
        em = getEntityManager();
        return em.find(Assignation.class, assignation.getIdAssignation());
    }
}
```

30. CREATE A NEW CLASS

We add a class to the project:



31. MODIFY THE FILE

TestDAOs.java:

Click to download

```
package test;

import dao.*;
import java.util.Date;
import model.*;

public class TestDAOs {

    public static void main(String[] args) {

        AddressDAO addressDao = new AddressDAO();
        StudentDAO studentDao = new StudentDAO();
        UserDAO userDao = new UserDAO();
        CourseDAO courseDao = new CourseDAO();
        AssignationDAO assignationDao = new AssignationDAO();

        //Insert Address object
        Address address = new Address();
        address.setStreetName("Estrella");
        address.setStreetNumber("109");
        address.setCountry("Mexico");
        addressDao.insert(address);
```

31. MODIFY THE FILE

TestDAOs.java:

Click to download

```
//Insert User object
User user = new User();
user.setUsername("john");
user.setPassword("1234");
userDao.insert(user);

//Insert Student Object
Student student = new Student();
student.setName("Jhon Smith");
student.setUser(user);
student.setAddress(address);
studentDao.insert(student);

//Insert Course Object
Course course = new Course();
course.setName("Java Fundamentals");
course.setPrice(1000F);
course.setSchedule("Morning");
courseDao.insert(course);
```

31. MODIFY THE FILE

TestDAOs.java:

Click to download

```
        //Insert Assignation Object
        Assignation assignation = new Assignation();
        assignation.setStudent(student);
        assignation.setCourse(course);
        assignation.setStartDate(new Date());
        assignation.setFinishDate(new Date());
        assignationDao.insert(assignation);

        //List all objects
        addressDao.list();
        userDao.list();
        studentDao.list();
        courseDao.list();
        assignationDao.list();
    }
}
```

32. CREATE AN XML FILE

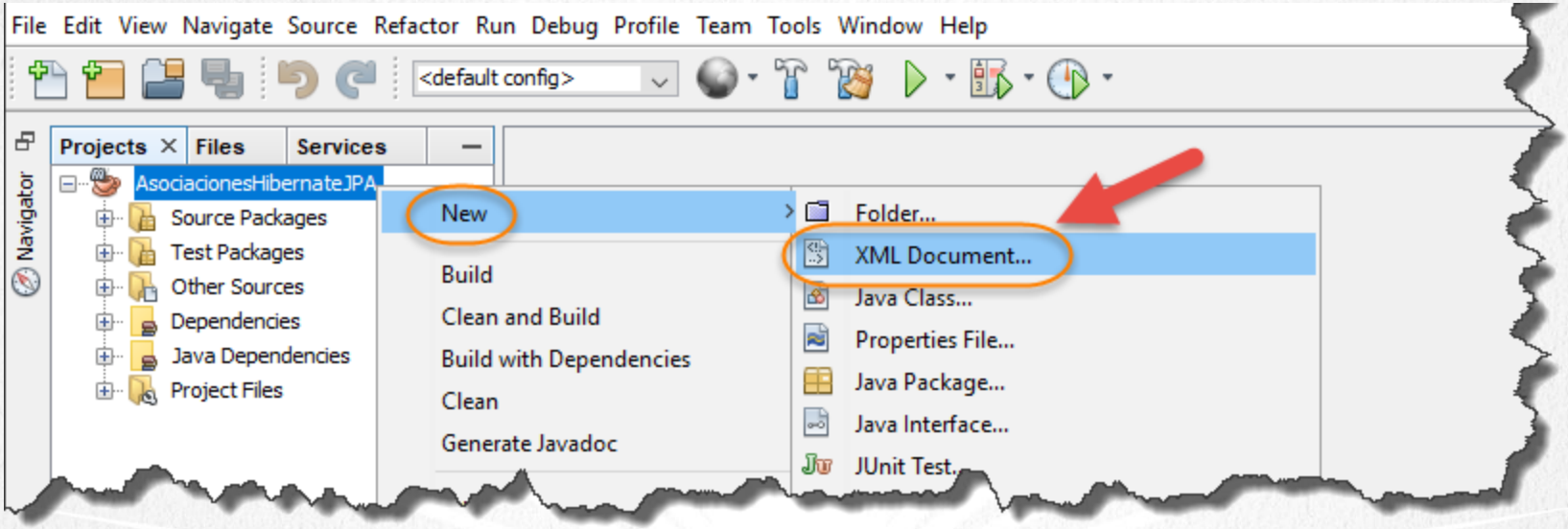
We create a log4j2.xml file. The log4j API allows us to manage the log or log of a Java application in a simpler way.

In order to use this API it is only necessary to add the log4j libraries which have already been added to the maven pom.xml file, and the log4j2.xml file somewhere that recognizes the classpath, for example in the project's resources folder .

With this we will be ready to specify what information we want to be sent to the console or other places, such as a file. For more information about this API consult:
https://logging.apache.org/log4j/2.x/

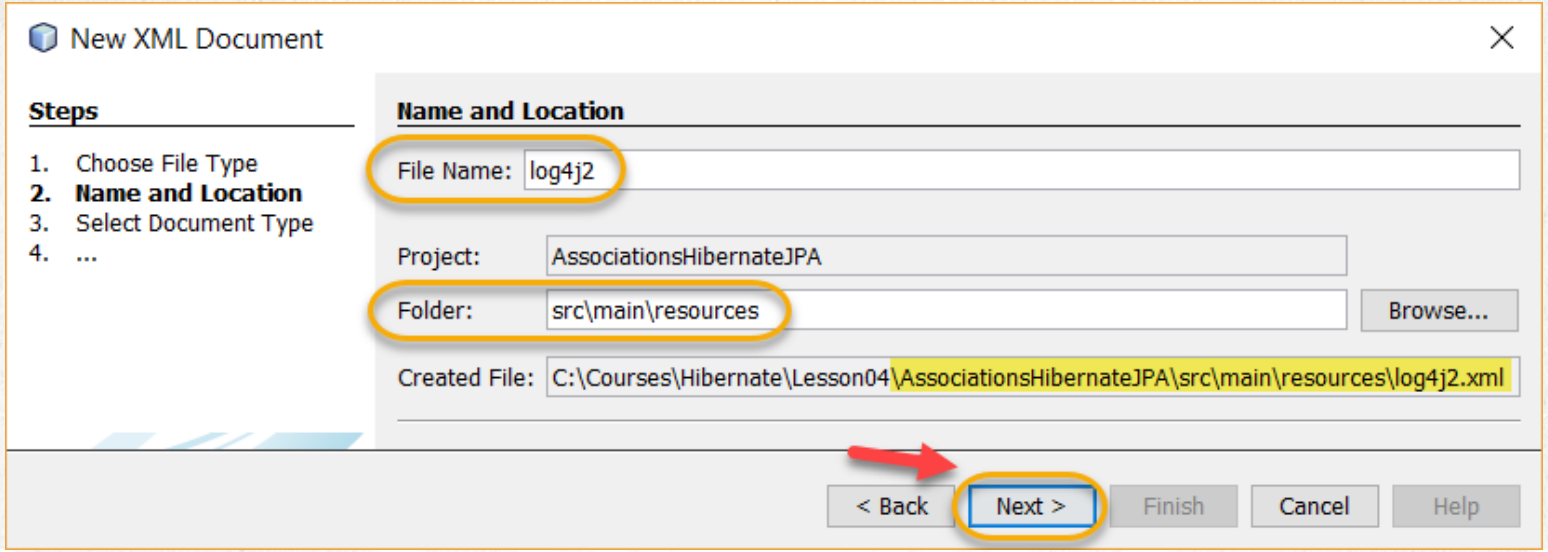
32. CREATE AN XML FILE

- We create the log4j2.xml file:



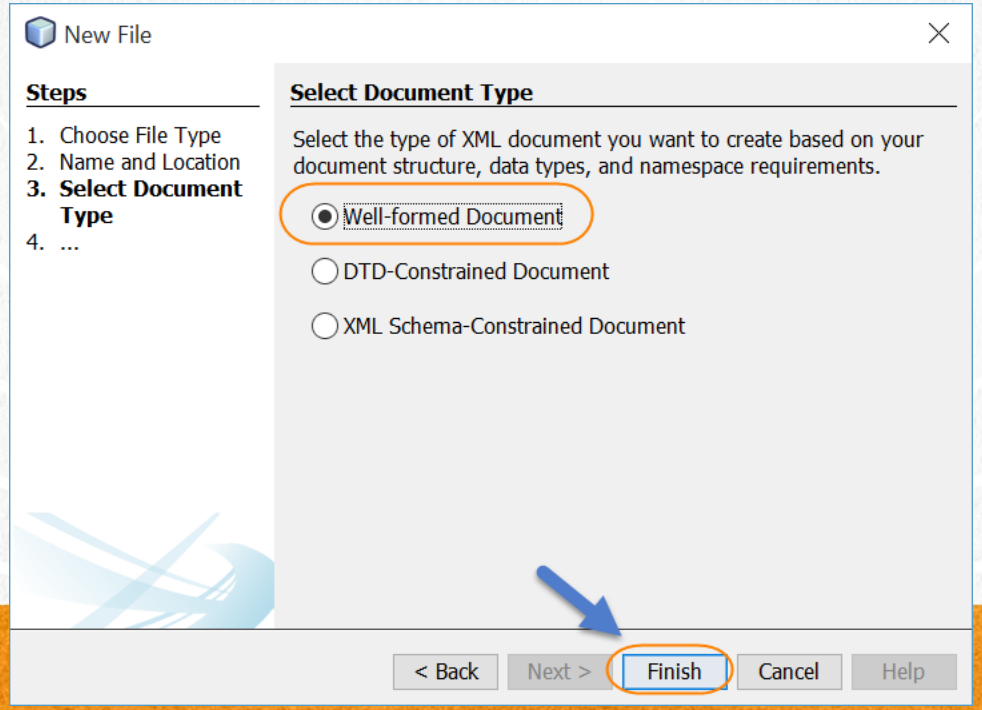
32. CREATE AN XML FILE

- We create the log4j2.xml file:



32. CREATE AN XML FILE

- We create the log4j2.xml file. In this step we select any option, it is not important since we are going to overwrite the file:



33. MODIFY THE FILE

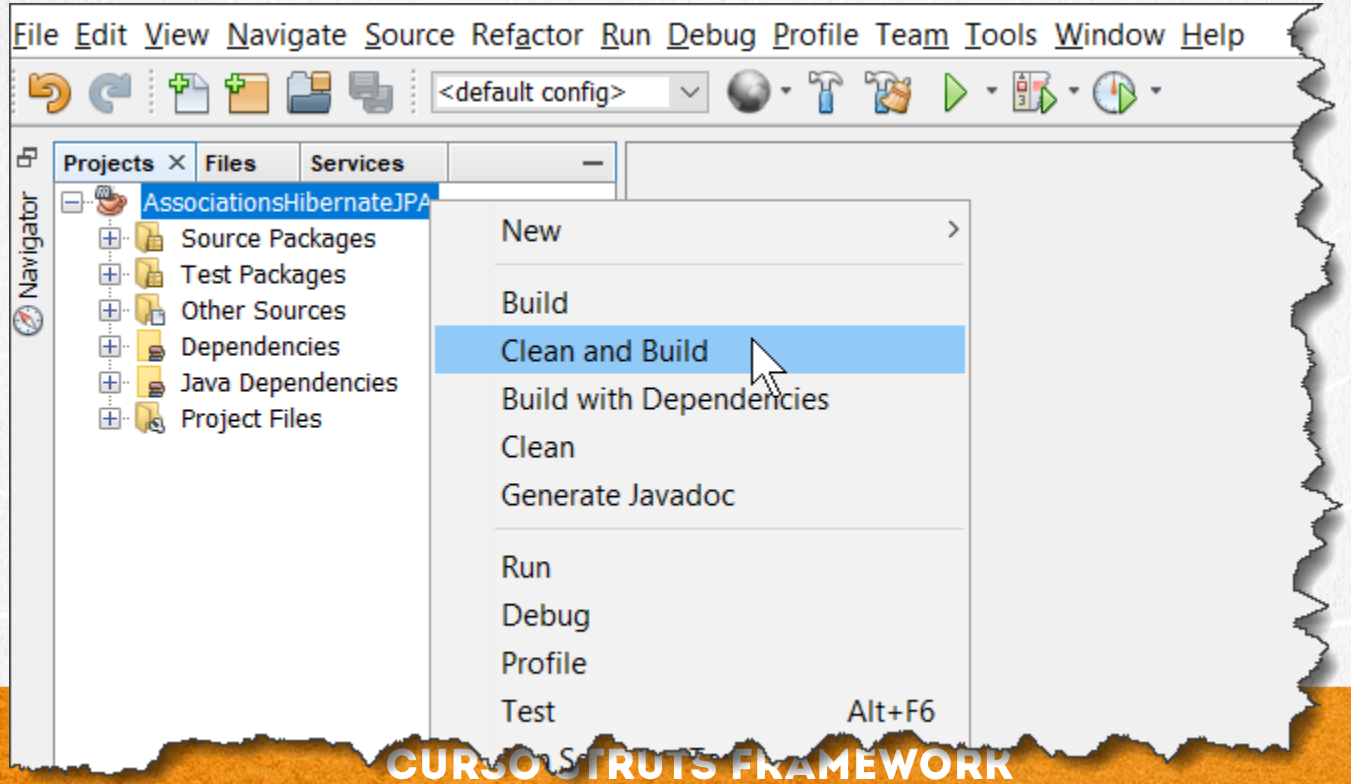
log4j2.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
    <Appenders>
        <Console name="Console" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss} [%t] %-5level %logger{36} - %msg%n" />
        </Console>
    </Appenders>
    <Loggers>
        <Logger name="org.hibernate.SQL" level="debug" additivity="false">
            <AppenderRef ref="Console"/>
        </Logger>
         <logger name="org.hibernate.type.descriptor.sql.BasicBinder" level="trace" additivity="false">
            <AppenderRef ref="Console"/>
        </logger>
        <Root level="info">
            <AppenderRef ref="Console" />
        </Root>
    </Loggers>
</Configuration>
```

34. EXECUTE CLEAN & BUILD

- To have the latest version of each file, we run a Clean & Build:



34. EXECUTE CLEAN & BUILD

- Once the process is finished, an output similar to the following should be shown:

```
Output
Retriever Output   Build (AssociationsHibernateJPA)

--- maven-jar-plugin:2.3.2:jar (default-jar) @ AssociationsHibernateJPA ---
Building jar: C:\Courses\Hibernate\Lesson04\AssociationsHibernateJPA\target\AssociationsHibernateJPA-1.jar

--- maven-install-plugin:2.3.1:install (default-install) @ AssociationsHibernateJPA ---
Installing C:\Courses\Hibernate\Lesson04\AssociationsHibernateJPA\target\AssociationsHibernateJPA-1.jar to C:\U
Installing C:\Courses\Hibernate\Lesson04\AssociationsHibernateJPA\pom.xml to C:\Users\user\.m2\repository\web\Ass
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.525s
Finished at: Wed Sep 12 19:42:41
Final Memory: 16M/385M
```

35. EXECUTE THE PROJECT

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Navigator

Projects | Files | Services

- AssociationsHibernateJPA
  - Source Packages
    - dao
    - model
    - test
      - TestDAOs.java
  - Test Packages
  - Other Sources
  - Dependencies
  - Java Dependencies
  - Project Files

Open	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Compile File	F9
Run File	Shift+F6
Debug File	Ctrl+Shift+F5
Profile File	
Test File	Ctrl+F6
Debug Test File	Ctrl+Shift+F6

35. EXECUTE THE PROJECT

- We observe the following result of execution of each operation using the DAO's created for each table. The result may vary depending on the data that is in the database:

```
Output
Retriever Output   Run (TestDAOs)
20:31:25 [main] INFO  org.hibernate.hql.internal.QueryTranslatorFactoryInitiator - HHH000397: Using ASTQueryTranslatorFactory
20:31:25 [main] DEBUG org.hibernate.SQL - insert into address (country, deleted, street_name, street_number, version) values (?, ?, ?, ?, ?)
Hibernate: insert into address (country, deleted, street_name, street_number, version) values (?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [VARCHAR] - [Mexico]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [VARCHAR] - [Estrella]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [VARCHAR] - [109]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into user (deleted, password, username, version) values (?, ?, ?, ?)
Hibernate: insert into user (deleted, password, username, version) values (?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [VARCHAR] - [1234]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [VARCHAR] - [john]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into student (id_address, deleted, name, id_user, version) values (?, ?, ?, ?, ?)
Hibernate: insert into student (id_address, deleted, name, id_user, version) values (?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [VARCHAR] - [Jhon Smith]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into course (deleted, name, price, schedule, version) values (?, ?, ?, ?, ?)
Hibernate: insert into course (deleted, name, price, schedule, version) values (?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [VARCHAR] - [Java Fundamentals]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [FLOAT] - [1000.0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [VARCHAR] - [Morning]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - insert into assignation (id_course, deleted, finish_date, start_date, id_student, version) values (?, ?, ?, ?, ?, ?)
Hibernate: insert into assignation (id_course, deleted, finish_date, start_date, id_student, version) values (?, ?, ?, ?, ?, ?)
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [1] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [2] as [INTEGER] - [0]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [3] as [TIMESTAMP] - [Wed Sep 12 20:31:25
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [4] as [TIMESTAMP] - [Wed Sep 12 20:31:25
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [5] as [INTEGER] - [1]
20:31:25 [main] TRACE org.hibernate.type.descriptor.sql.BasicBinder - binding parameter [6] as [INTEGER] - [0]
20:31:25 [main] DEBUG org.hibernate.SQL - select address0_.id_address as id_addre1_0_, address0_.country as country2_0_, address0_.deleted as deleted3_0_, addre
Hibernate: select address0_.id_address as id_addre1_0_, address0_.country as country2_0_, address0_.deleted as deleted3_0_, address0_.street_name as street_n4_0
Address{idAddress=1, streetName=Estrella, streetNumber=109, country=Mexico, version=0, deleted=0}
20:31:25 [main] DEBUG org.hibernate.SQL - select user0_.id_user as id_user1_4_, user0_.deleted as deleted2_4_, user0_.password as password3_4_, user0_.username
Hibernate: select user0_.id_user as id_user1_4_, user0_.deleted as deleted2_4_, user0_.password as password3_4_, user0_.username as username4_4_, user0_.version
User{idUser=1, username=john, password=1234, version=0, deleted=0}
20:31:25 [main] DEBUG org.hibernate.SQL - select student0_.id_student as id_studel_3_, student0_.id_address as id_addre5_3_, student0_.deleted as deleted2_3_, s
```

EXERCISE CONCLUSION

As this exercise we have created the basis of our project that we will be using throughout the course.

We create each of the Entity classes, with their relationships included according to the Entity - Relationship scheme described above.

In addition, the base DAO classes that will allow us to interact with each of the Entity classes using Hibernate / JPA.

ONLINE COURSE

HIBERNATE & JPA

By: Eng. Ubaldo Acosta