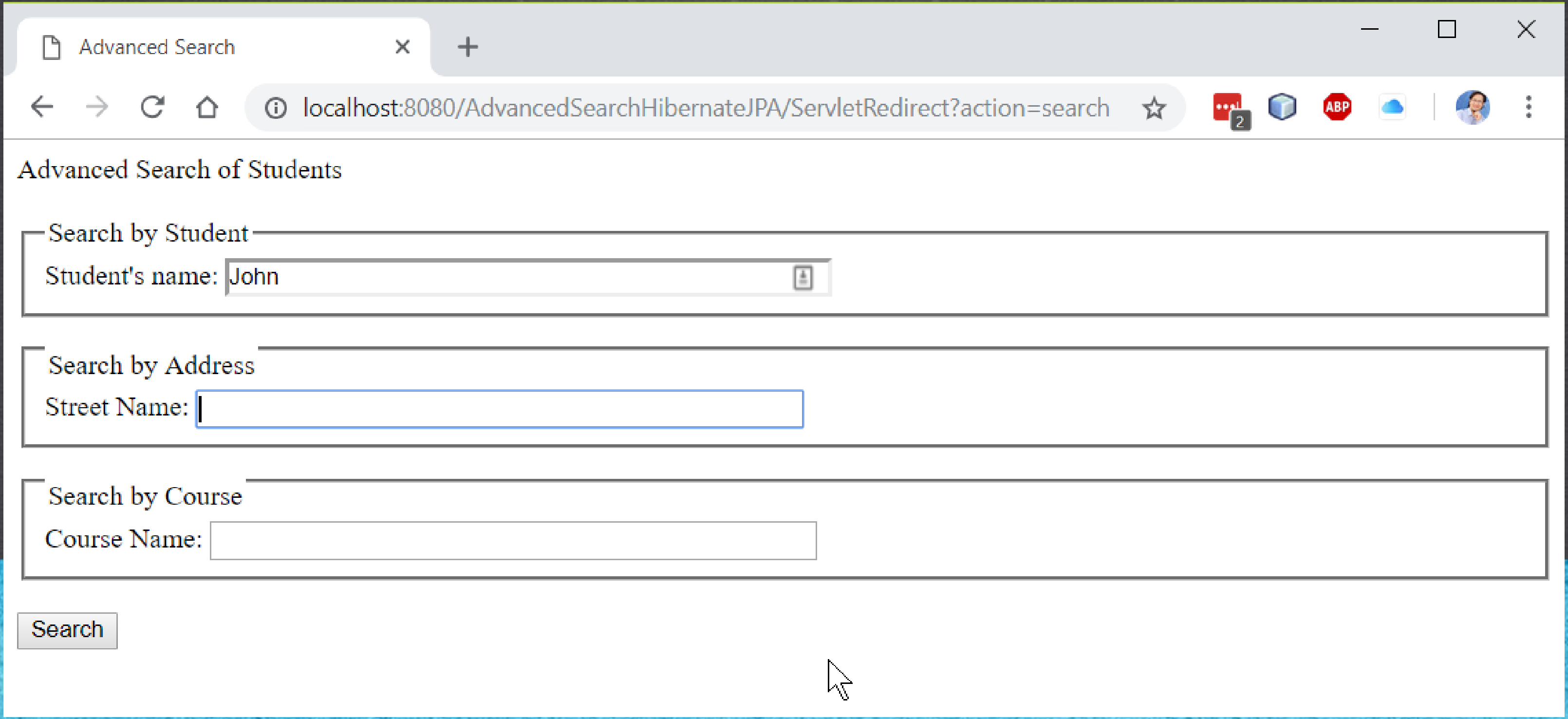
HIBERNATE & JPA COURSE

EXERCISE

LAB
ADVANCED SEARCH

EXERCISE OBJECTIVE

Use the Criteria API to create the advanced search laboratory. At the end we should observe the following:



Advanced Search
localhost:8080/AdvancedSearchHibernateJPA/ServletRedirect?action=search

Advanced Search of Students

Search by Student

Student's name: John

Search by Address

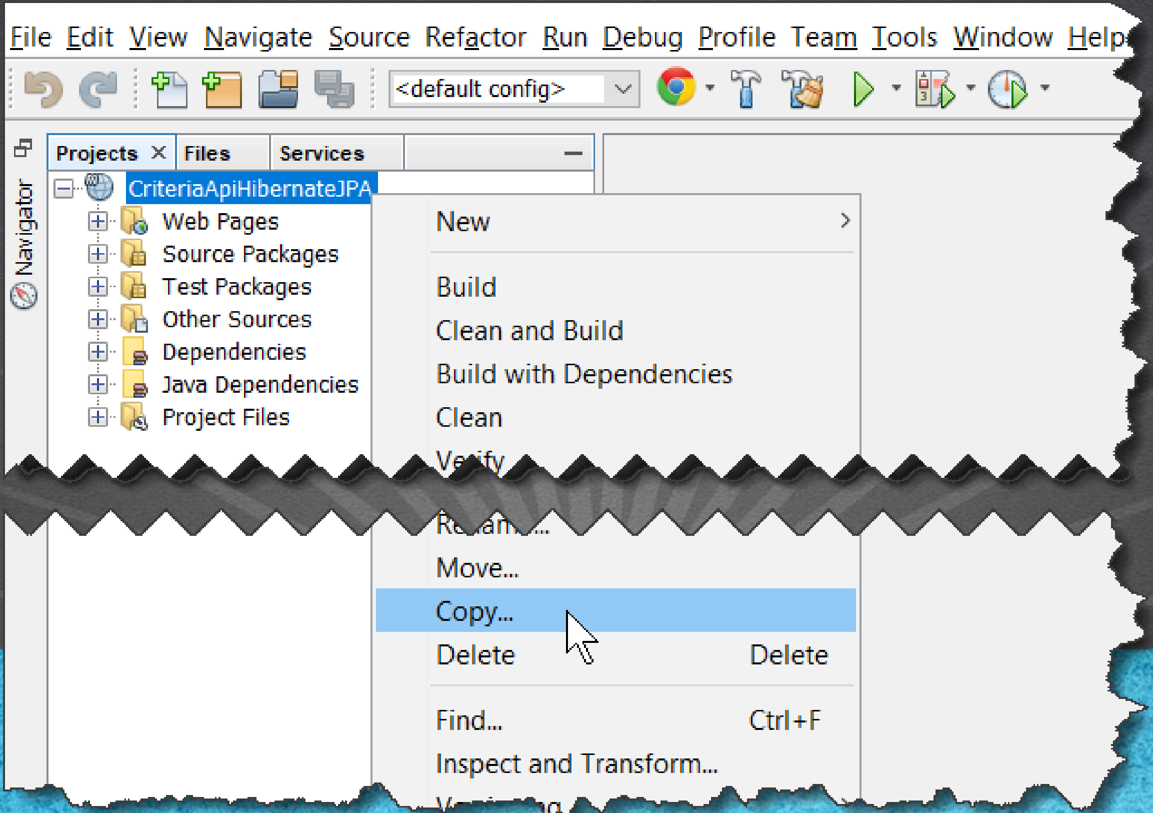
Street Name:

Search by Course

Course Name:

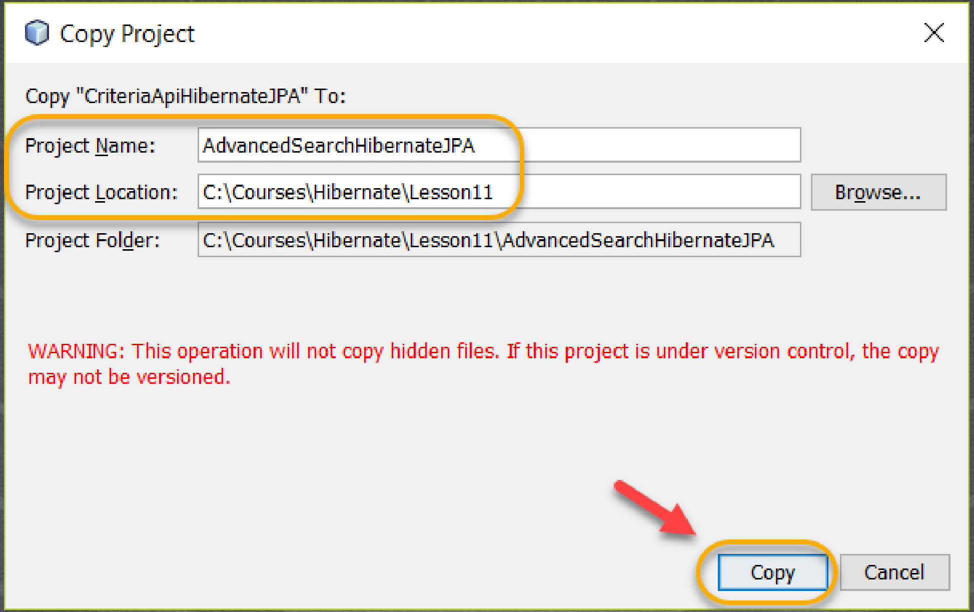
1. COPY THE PROJECT

We copy the project starting from CriteriaApiHibernateJPA:



1. COPY THE PROJECT

Create the Project AdvancedSearchHibernateJPA:



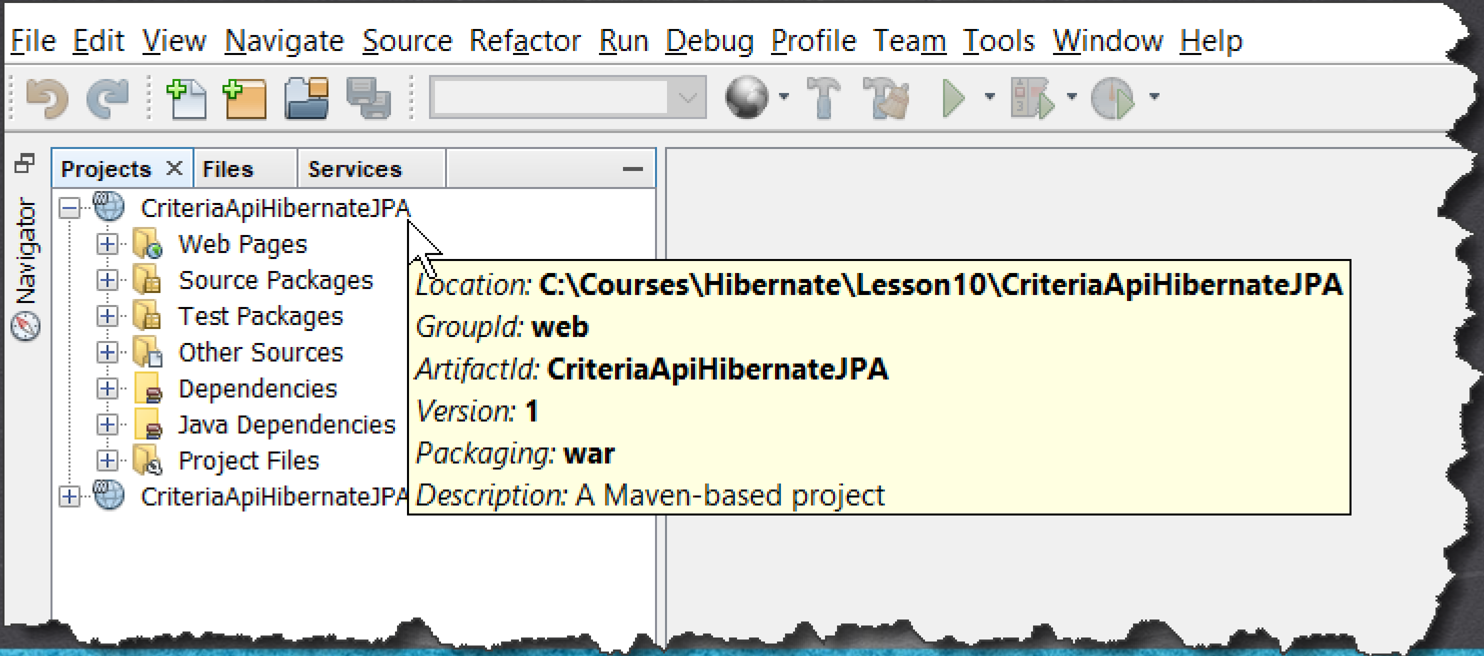
Copy "CriteriaApiHibernateJPA" To:

Project Name:	AdvancedSearchHibernateJPA
Project Location:	C:\Courses\Hibernate\Lesson11
Project Folder:	C:\Courses\Hibernate\Lesson11\AdvancedSearchHibernateJPA

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

2. CLOSED THE UNUSED PROJECT

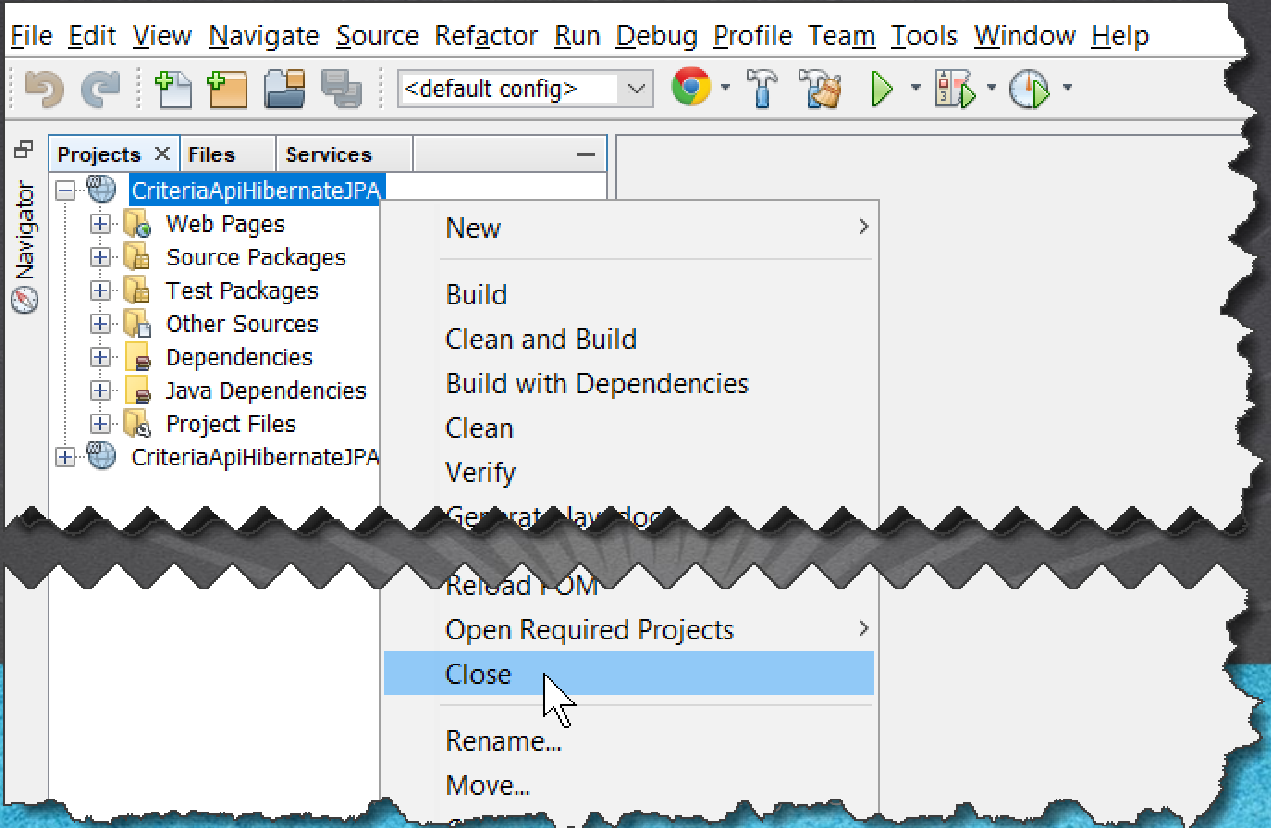
We close the project that we no longer use:



Location: C:\Courses\Hibernate\Lesson10\CriteriaApiHibernateJPA
GroupId: web
ArtifactId: CriteriaApiHibernateJPA
Version: 1
Packaging: war
Description: A Maven-based project

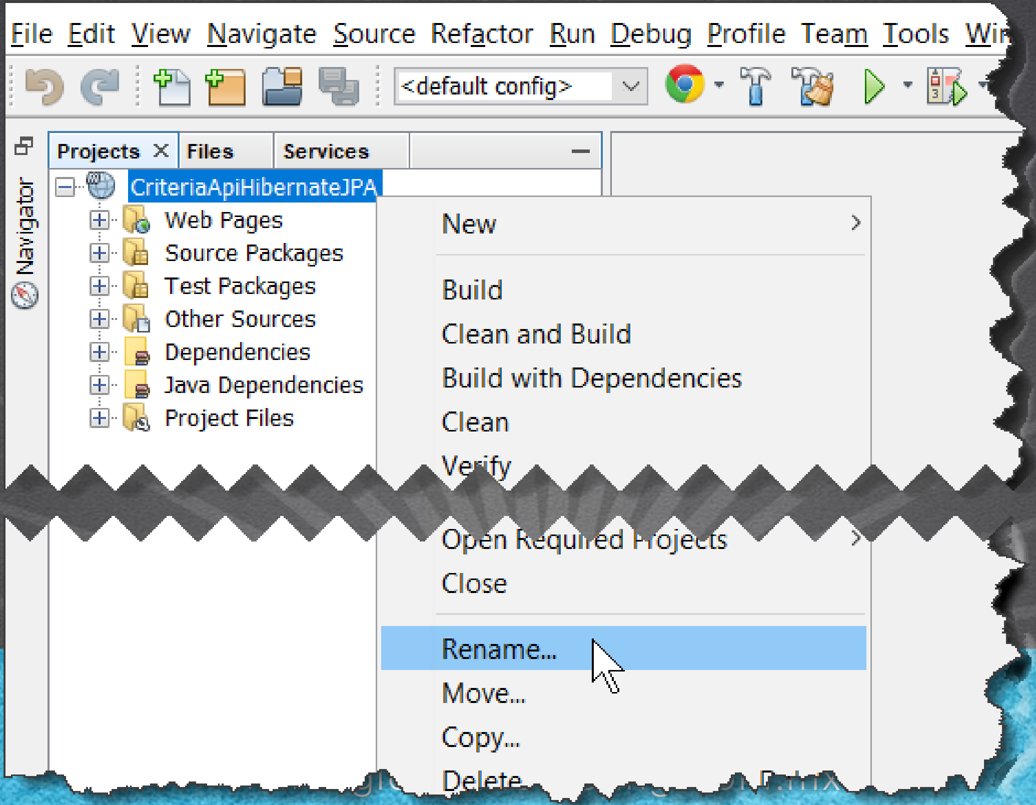
2. CLOSED THE UNUSED PROJECT

We close the project that we no longer use:



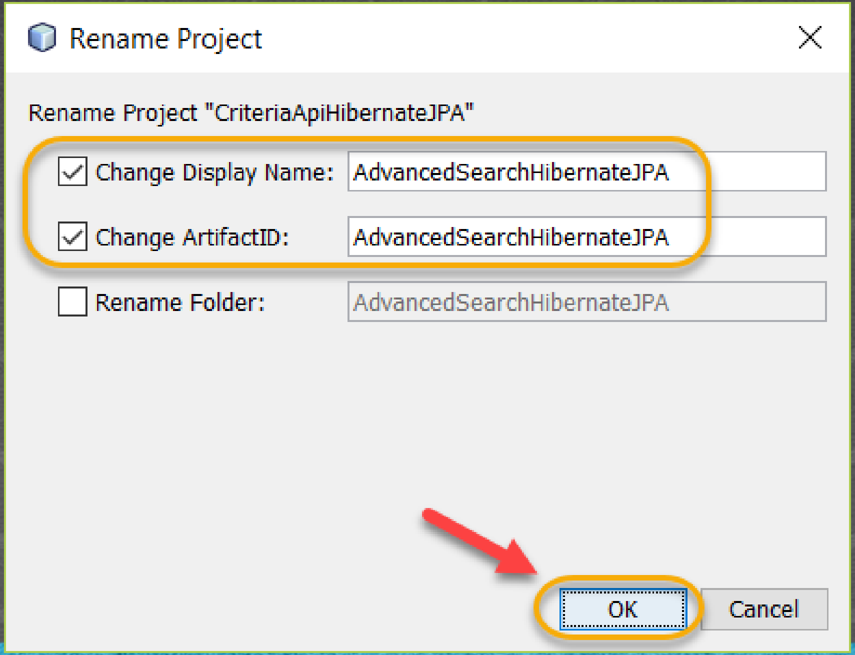
3. RENAME THE PROJECT

Rename the project:



3. RENAME THE PROJECT

Rename the Project:



Rename Project

Rename Project "CriteriaApiHibernateJPA"

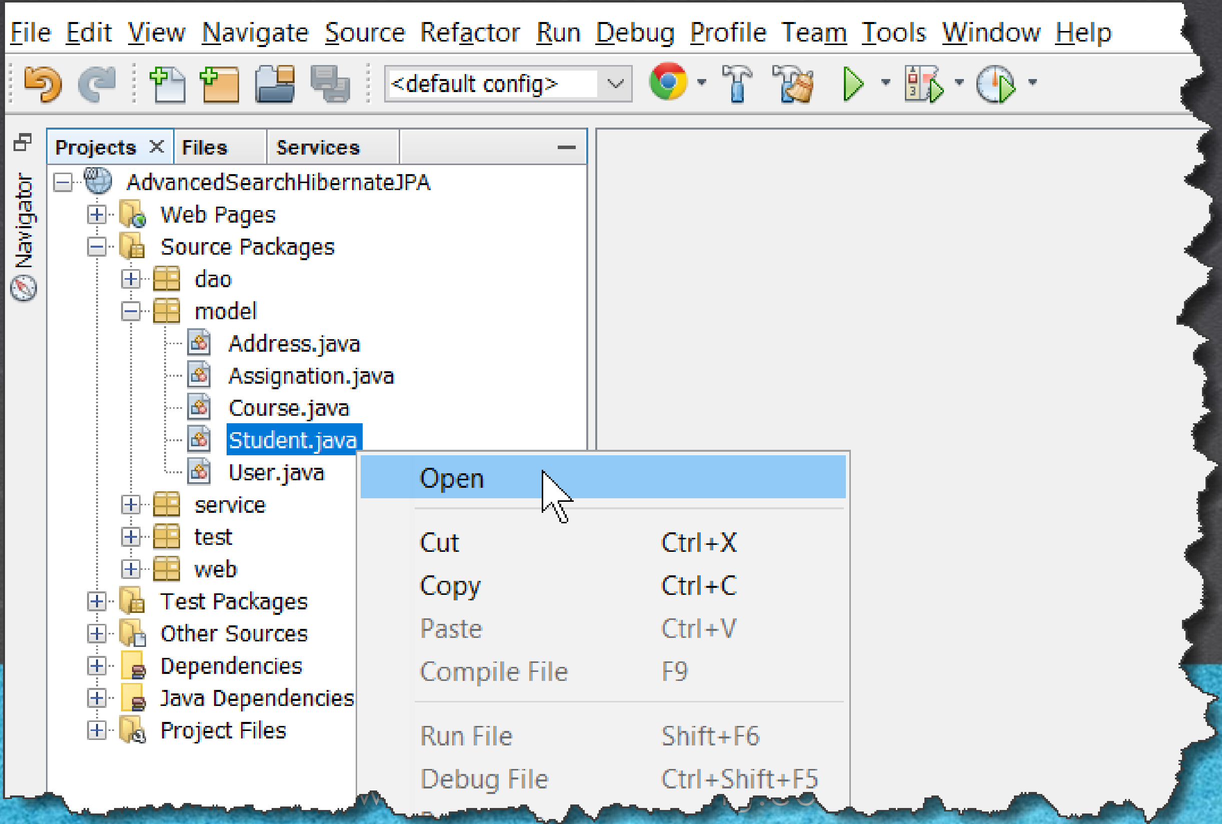
[✓] Change Display Name: AdvancedSearchHibernateJPA

[✓] Change ArtifactID: AdvancedSearchHibernateJPA

[ ] Rename Folder: AdvancedSearchHibernateJPA

4. MODIFY THE CLASS

Modify the Student.java. We change the name of the assignation's property:



4. 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;

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

4. MODIFY THE CODE

Student.java:

Click to download

```
@JoinColumn(name = "id_address", referencedColumnName = "id_address")
@ManyToOne(cascade = CascadeType.ALL)
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;
}

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

4. MODIFY THE CODE

Student.java:

Click to download

```
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;
}

public int getDeleted() {
    return deleted;
}

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

4. MODIFY THE CODE

Student.java:

Click to download

```
    public List<Assignation> getAssignations() {
        return assignations;
    }

    public void setAssignations(List<Assignation> assignations) {
        this.assignations = assignations;
    }

    public Address getAddress() {
        return address;
    }

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

    public User getUser() {
        return user;
    }

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

4. MODIFY THE CODE

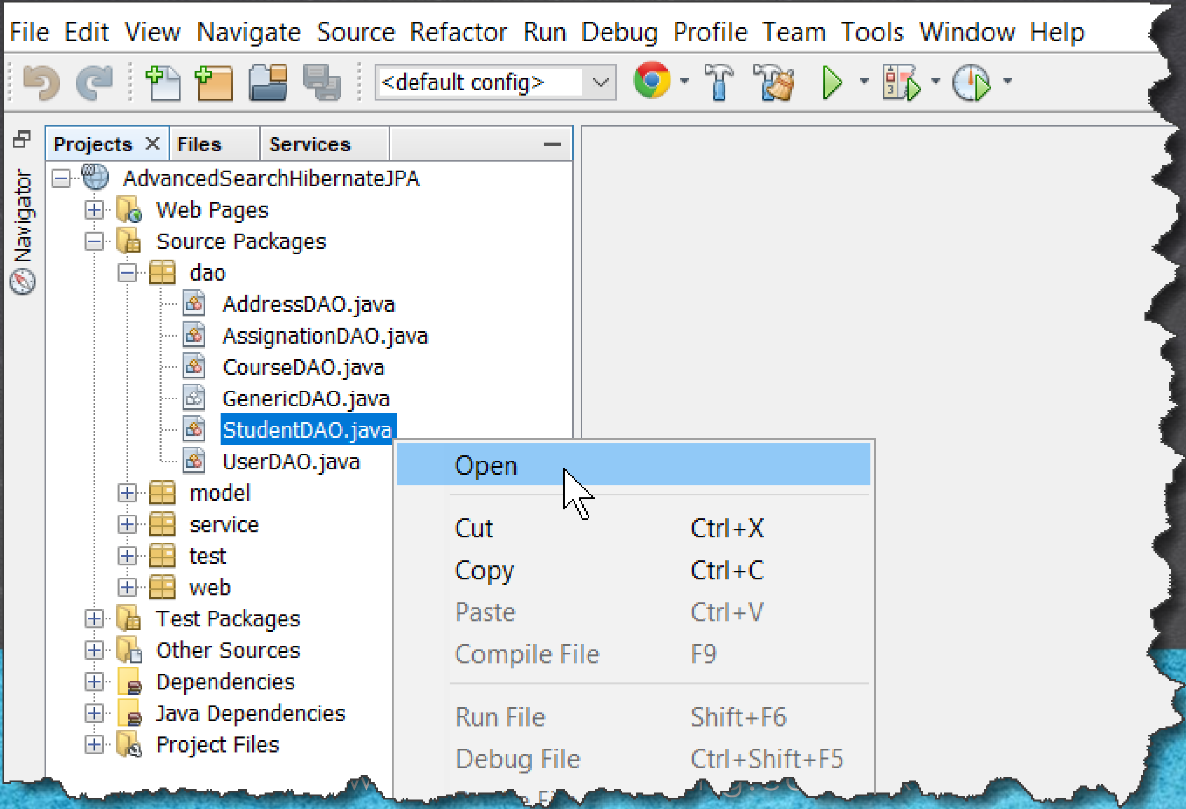
Student.java:

Click to download

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

    @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 + '}';
    }

}
```

5. MODIFY A CLASS

Modify the StudentDAO.java class:



5. MODIFY THE CODE

StudentDAO.java:

Click to download

```
package dao;

import static dao.GenericDAO.em;
import java.util.List;
import java.util.Map;
import javax.persistence.Query;
import model.*;
import org.apache.logging.log4j.*;
import org.hibernate.*;
import org.hibernate.criterion.*;
import org.hibernate.sql.JoinType;

public class StudentDAO extends GenericDAO {

    Logger log = LogManager.getRootLogger();

    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;
    }
```

5. MODIFY THE CODE

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;
        }
    }
}
```

5. MODIFY THE CODE

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;
        }
    }
}
```

5. MODIFY THE CODE

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());
    }

    public List searchStudentsByCriteria(Map criteriaMap) {
        log.debug("finding Student instances by different criteria");
        List list = null;
        //1. We obtain the criteria one by one according to the search that we program
        Student studentDTO = null;
        Address addressDTO = null;
        Course courseDTO = null;
```

5. MODIFY THE CODE

StudentDAO.java:

Click to download

```
if (criteriaMap != null) {
    studentDTO = (Student) criteriaMap.get("student");
    addressDTO = (Address) criteriaMap.get("address");
    courseDTO = (Course) criteriaMap.get("course");
}
try {
    //2. We create the criteria object to execute the query.
    //The Criteria API method of Hibernate has been depreciated
    //in version 5.2, however JPA does not offer at the moment an
    //option to QueryByExample. So until JPA offers an option for
    //this type of queries you can use this version
    Criteria criteria = em.unwrap(Session.class).getSession().createCriteria(Student.class);

    //3. We add the criteria received
    if (studentDTO != null) {

        //We use a Student's DTO and a QBE (Query By Example). We wrap it in an Example
        Example exampleStudent = Example.create(studentDTO).enableLike(MatchMode.ANYWHERE);

        //we add the example to the criteria query
        criteria.add(exampleStudent);
    }
    if (addressDTO != null) {

        //We use an Address DTO and a QBE. We wrap it in an Example
        Example exampleAddress = Example.create(addressDTO).enableLike(MatchMode.ANYWHERE);
```

5. MODIFY THE CODE

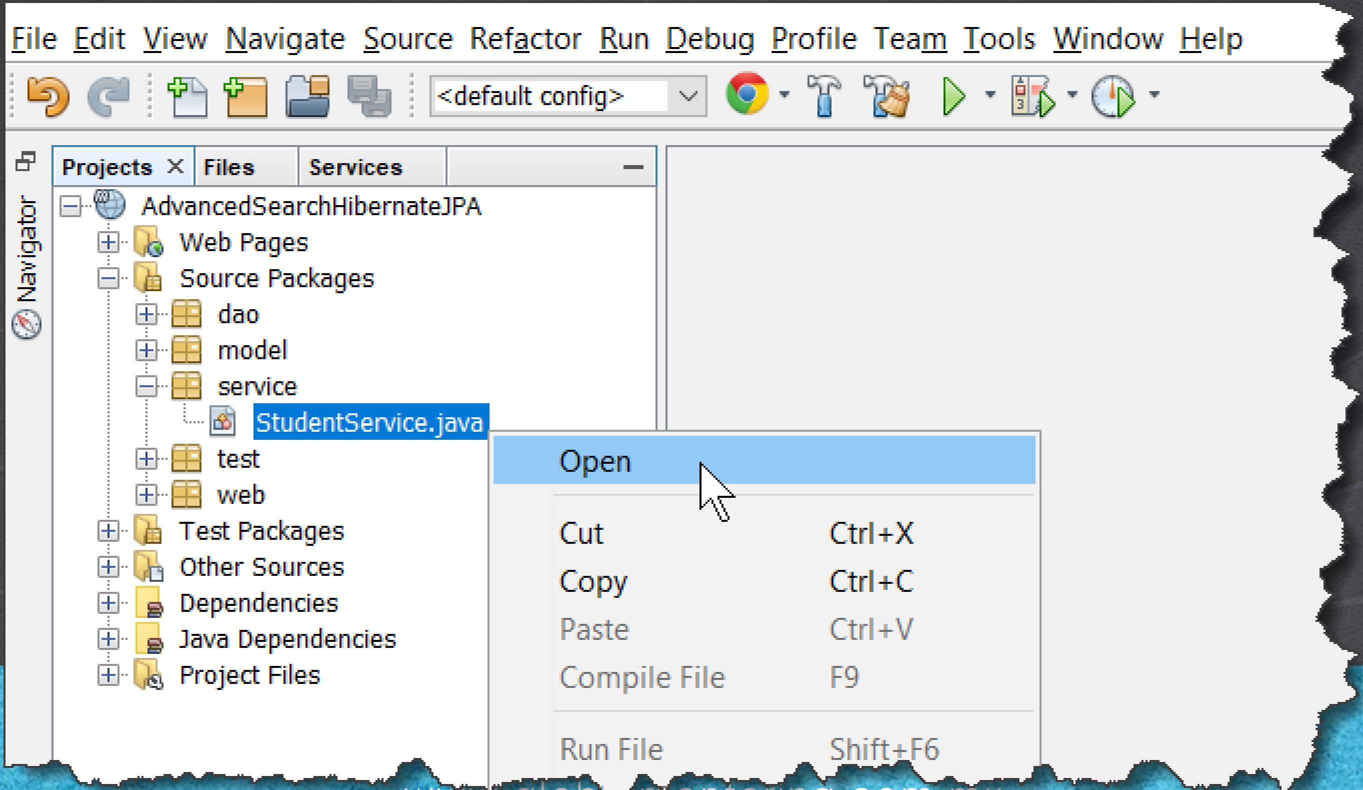
StudentDAO.java:

Click to download

```
                //We add the address restriction
                criteria.createCriteria("address", JoinType.LEFT_OUTER_JOIN)
                        .setFetchMode("address", FetchMode.JOIN)
                        .add(exampleAddress);
            }
            if (courseDTO != null) {

                //We wrap it in an Example
                Example exampleCourse = Example.create(courseDTO).enableLike(MatchMode.ANYWHERE);

                //We add the restriction of the course, first accessing to assignments
                //and then to course adding one criterion inside another (adding a criterion
                //means that it becomes the pivot table or root at that moment of the query)
                criteria.createCriteria("assignations", JoinType.LEFT_OUTER_JOIN)
                        .setFetchMode("assignations", FetchMode.JOIN)
                        .createCriteria("course", JoinType.LEFT_OUTER_JOIN)
                        .setFetchMode("course", FetchMode.JOIN)
                        .add(exampleCourse);
            }
            //Restrict to obtain only the different elements
            criteria.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);
            //4. We get the list
            list = criteria.list();
        } catch (RuntimeException re) {
            log.error("find students by criteria failed", re);
            throw re;
        }
        return list;
    }
}
```

6. MODIFY A CLASS

Modify the StudentService.java class:



6. MODIFY THE CODE

StudentService.java:

Click to download

```
package service;

import dao.StudentDAO;
import model.Student;
import java.util.List;
import java.util.Map;

public class StudentService {

    StudentDAO studentDao = new StudentDAO();

    public List<Student> listStudents() {
        return studentDao.list();
    }

    public boolean saveStudent(Student student) {
        if (student != null && student.getIdStudent() == null) {
            studentDao.insert(student);
        } else {
            studentDao.update(student);
        }

        return true;// If nothing fails, we return true
    }
```

6. MODIFY THE CODE

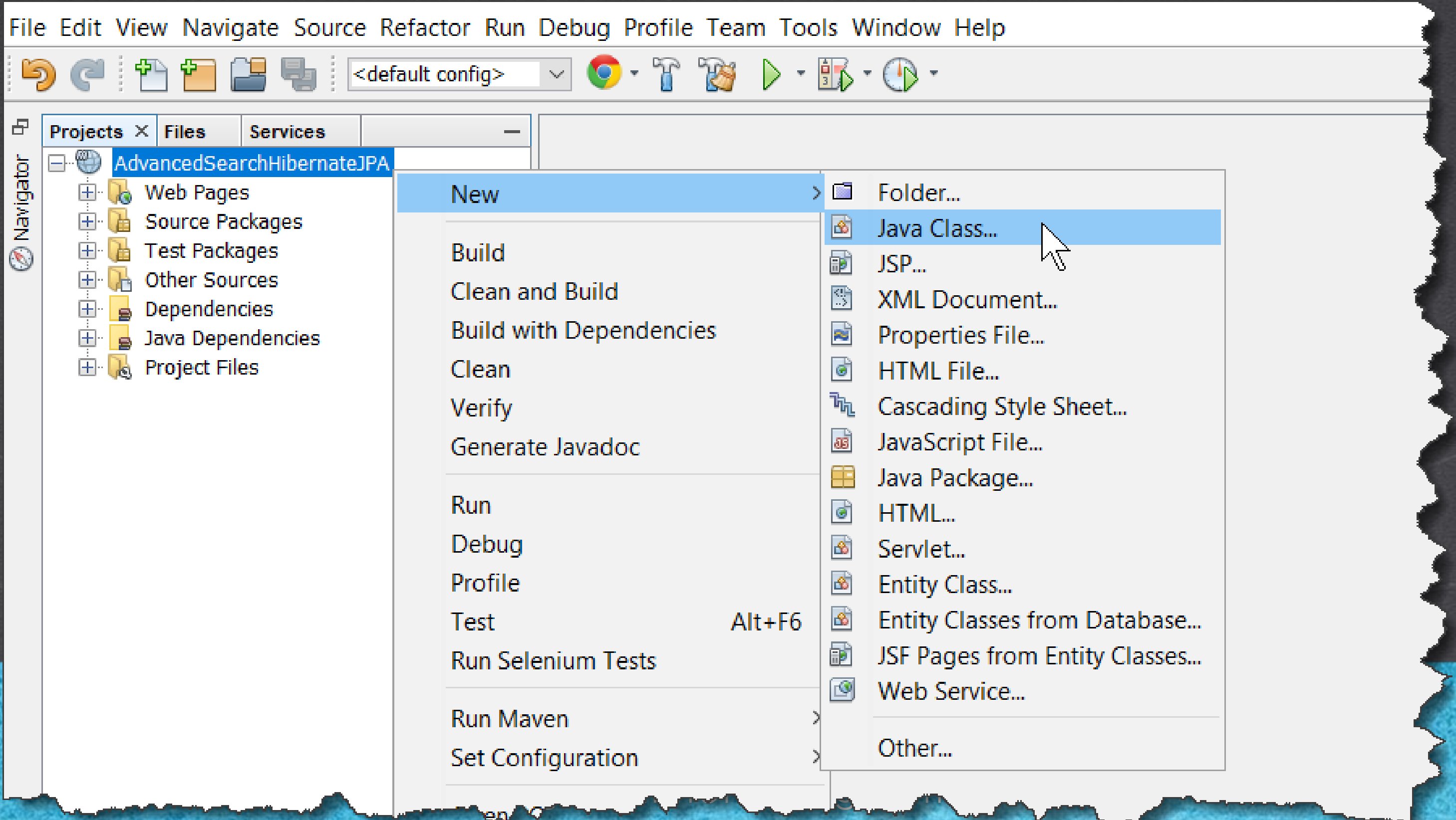
StudentService.java:

Click to download

```
    public boolean deleteStudent(Integer idStudent) {
        studentDao.delete(new Student(idStudent));
        return true;// If nothing fails, we return true
    }

    public Student findStudent(Integer idStudent) {
        return studentDao.findById(new Student(idStudent));
    }

    public List<Student> searchStudentsByCriteria(Map criterios) {
        return studentDao.searchStudentsByCriteria(criterios);
    }

}
```

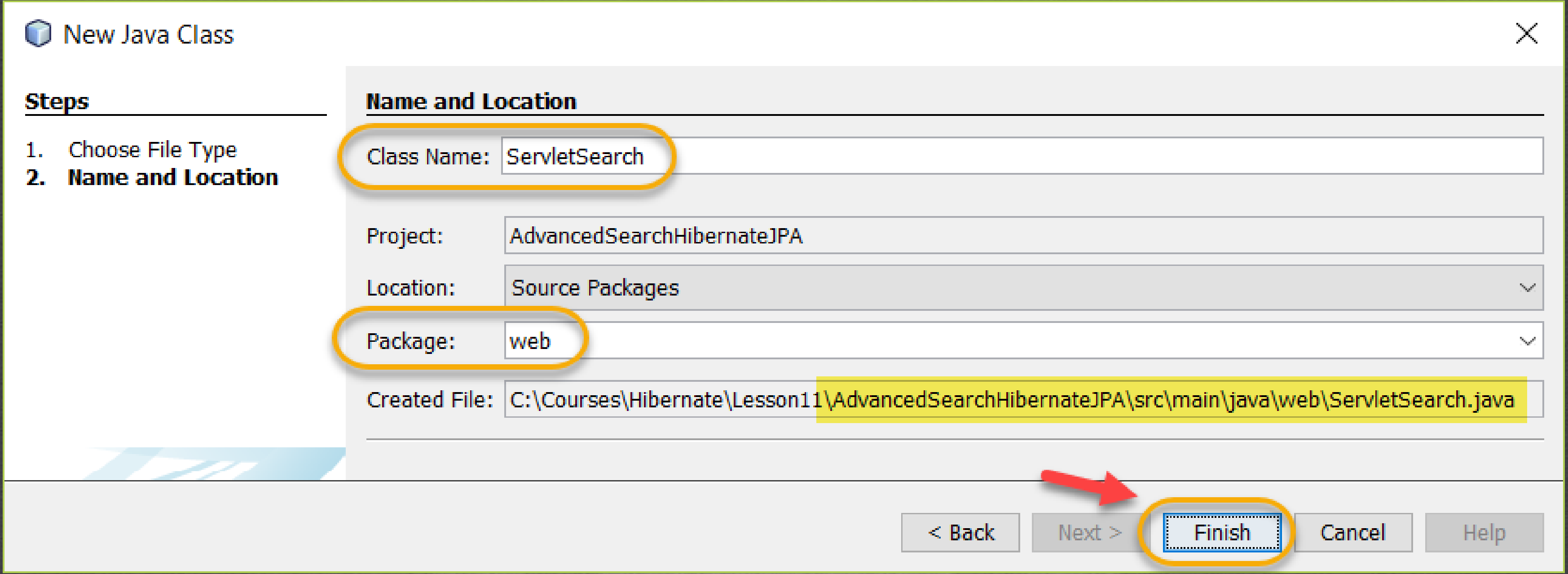
7. CREATE A CLASS

Create the ServletSearch.java class:



7. CREATE A CLASS

Create the ServletSearch.java class:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: ServletSearch

Project: AdvancedSearchHibernateJPA

Location: Source Packages

Package: web

Created File: C:\Courses\Hibernate\Lesson11\AdvancedSearchHibernateJPA\src\main\java\web\ServletSearch.java

8. MODIFY THE CODE

ServletSearch.java:

Click to download

```java
package web;

import java.io.IOException;
import java.util.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import model.*;
import service.StudentService;

@WebServlet("/ServletSearch")
public class ServletSearch extends HttpServlet {

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        //1. We retrieve the search parameters
        //We retrieve the student's data
        String studentName = request.getParameter("studentName");
        studentName = "".equals(studentName) ? null : studentName.trim();
        Student studentDTO = null;
        if (studentName != null) {
            studentDTO = new Student();
            studentDTO.setName(studentName);
        }
```

8. MODIFY THE CODE

ServletSearch.java:

Click to download

```
//We retrieve the Address data
String streetName = request.getParameter("streetName");
streetName = "".equals(streetName) ? null : streetName.trim();
Address addressDTO = null;
if (streetName != null) {
    addressDTO = new Address();
    addressDTO.setStreetName(streetName);
}

//We retrieve the course data
String courseName = request.getParameter("courseName");
courseName = "".equals(courseName) ? null : courseName.trim();
Course courseDTO = null;
if (courseName != null) {
    courseDTO = new Course();
    courseDTO.setName(courseName);
}

//2. We add the parameters to a map, this allows adding more filters if needed
Map criteriaMap = new HashMap();
criteriaMap.put("student", studentDTO);
criteriaMap.put("address", addressDTO);
criteriaMap.put("course", courseDTO);
```

8. MODIFY THE CODE

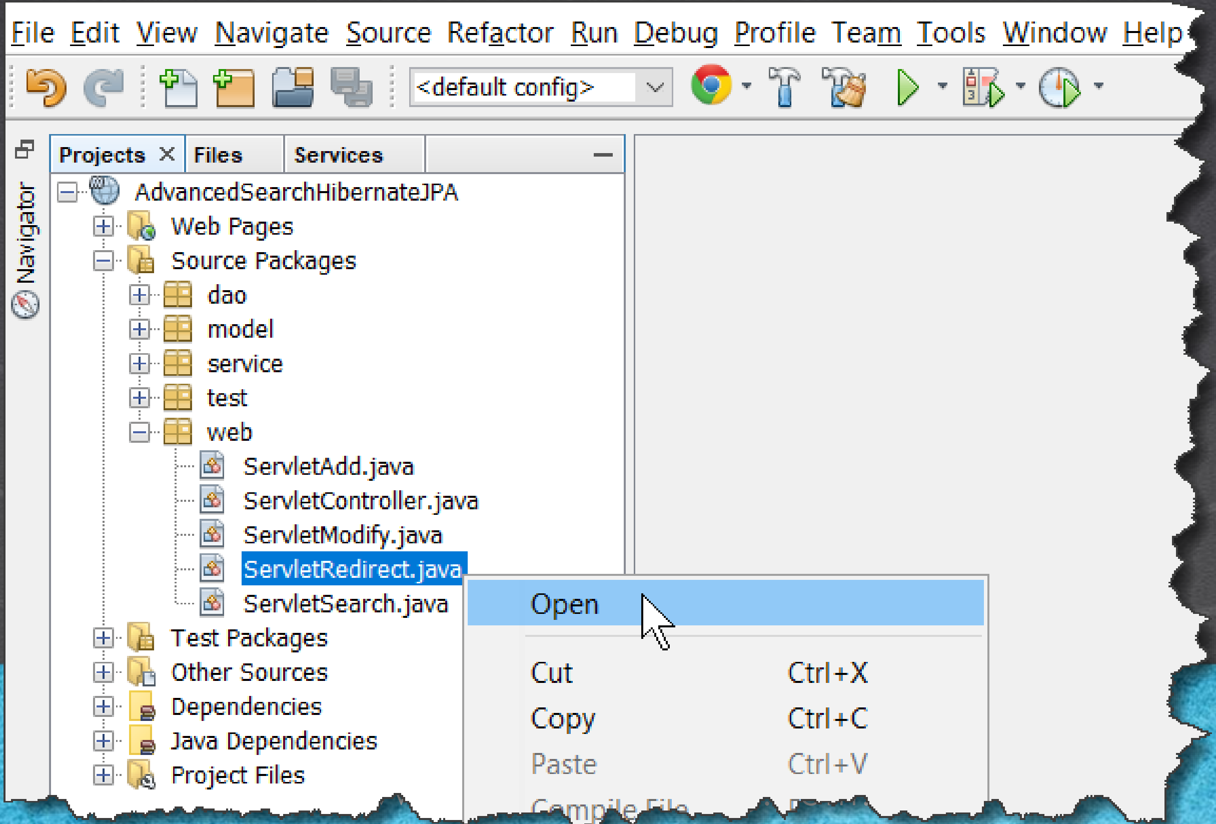
ServletSearch.java:

Click to download

```
        //3. We communicate with the service layer
        StudentService studentService = new StudentService();

        //4. We send the map of parameters to create the filter
        List<Student> students = studentService.searchStudentsByCriteria(criteriaMap);

        //5. We share the information (Model) with the view
        request.setAttribute("students", students);

        //6. We select the view to show the info of students
        request.getRequestDispatcher("/WEB-INF/listStudents.jsp").forward(request, response);
    }

}
```

9. MODIFY A SERVLET

We modify the ServletRedirect.java:



9. MODIFY THE CODE

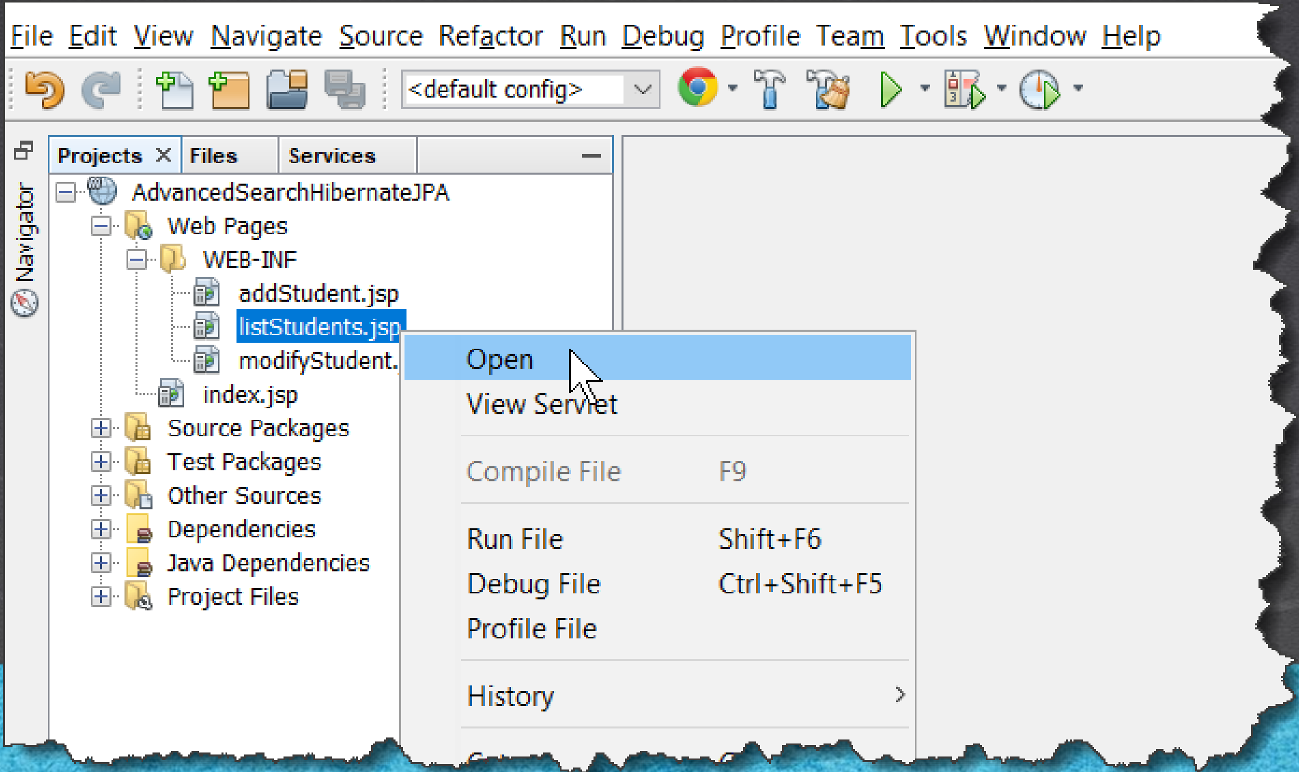
ServletRedirect.java:

Click to download

```
package web;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/ServletRedirect")
public class ServletRedirect extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String accion = request.getParameter("action");
        if ("add".equals(accion)) {
            //Redirect to the Add Student page
            request.getRequestDispatcher("WEB-INF/addStudent.jsp").forward(request, response);
        } else if ("search".equals(accion)) {
            //We redirected to the Advanced Search page to use criteria
            request.getRequestDispatcher("WEB-INF/advancedSearch.jsp").forward(request, response);

        }
    }
}
```

10. MODIFY A JSP

Modify the listStudents.jsp:



10. MODIFY THE CODE

listStudents.jsp:

Click to download

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"  %>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>List of Students</title>
    </head>
    <body>
        List of Students
        <br/>
       <a
            href="${pageContext.request.contextPath}/ServletController">
            List </a> |
        <a
            href="${pageContext.request.contextPath}/ServletRedirect?action=add">
            Add </a> |
        <a
            href="${pageContext.request.contextPath}/ServletRedirect?action=search">
            Advanced Search</a>
        <br/>
```

10. MODIFY THE CODE

listStudents.jsp:

Click to download

```
        <c:if test="${not empty students}">
             <table border="1">
                 <tr>
                     <th>Student ID</th>
                     <th>Name</th>
                     <th>Street name</th>
                     <th>Street number</th>
                     <th>Country</th>
                     <th>Courses</th>
                 </tr>
                 <c:forEach var="student" items="${students}">
                     <tr>
                         <td>
                             <a href="${pageContext.request.contextPath}/ServletModify?idStudent=${student.idStudent}">
                                 ${student.idStudent}
                             </a>
                         </td>
                         <td>${student.name}</td>
                         <td>${student.address.streetName }</td>
                         <td>${student.address.streetNumber }</td>
                         <td>${student.address.country }</td>
```

10. MODIFY THE CODE

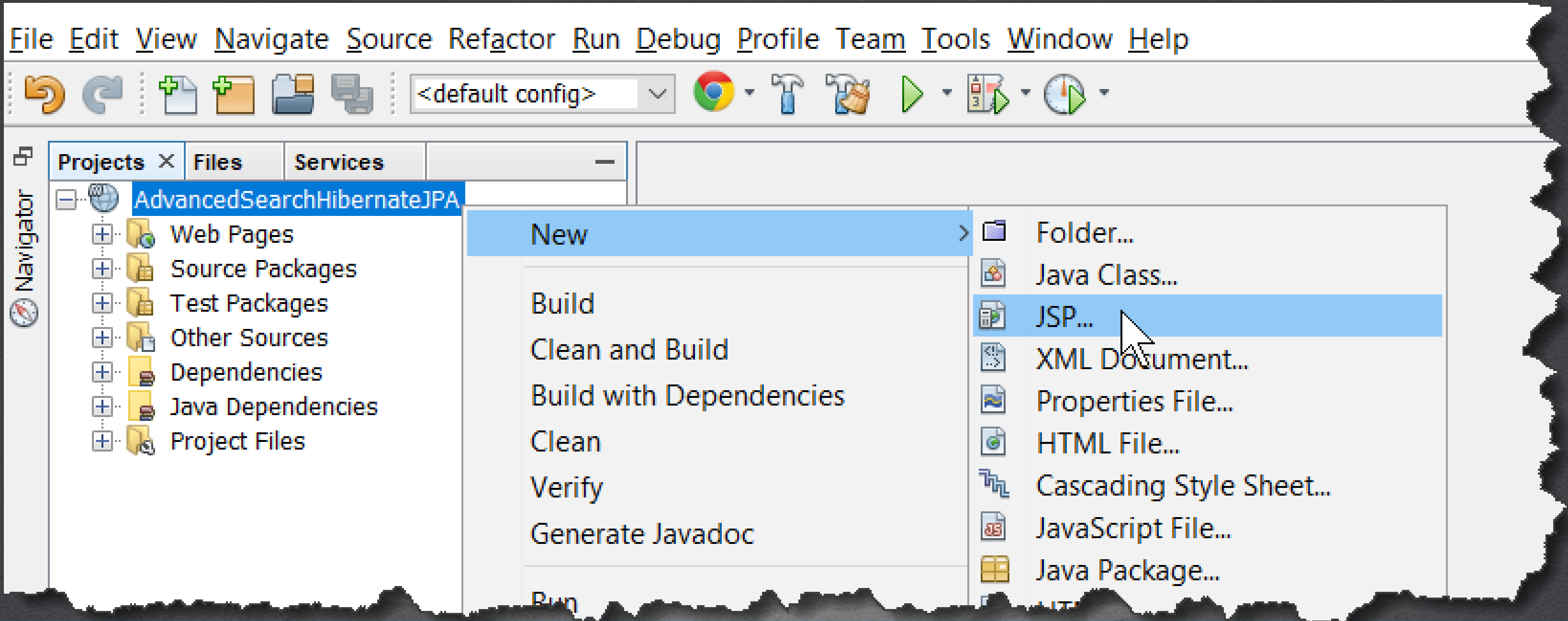
listStudents.jsp:

Click to download

```
                        <td>
                                 <ul>
                                     <c:forEach var="assignation" items="${student.assignations}">
                                         <li>
                                             ${assignation.course.name}
                                         </li>
                                     </c:forEach>
                                     &nbsp;
                                 </ul>
                         </td>
                     </tr>
                 </c:forEach>
             </table>
          </c:if>
         <c:if test="${empty students}">
             <br/>
             No students found
         </c:if>
     </body>
</html>
```

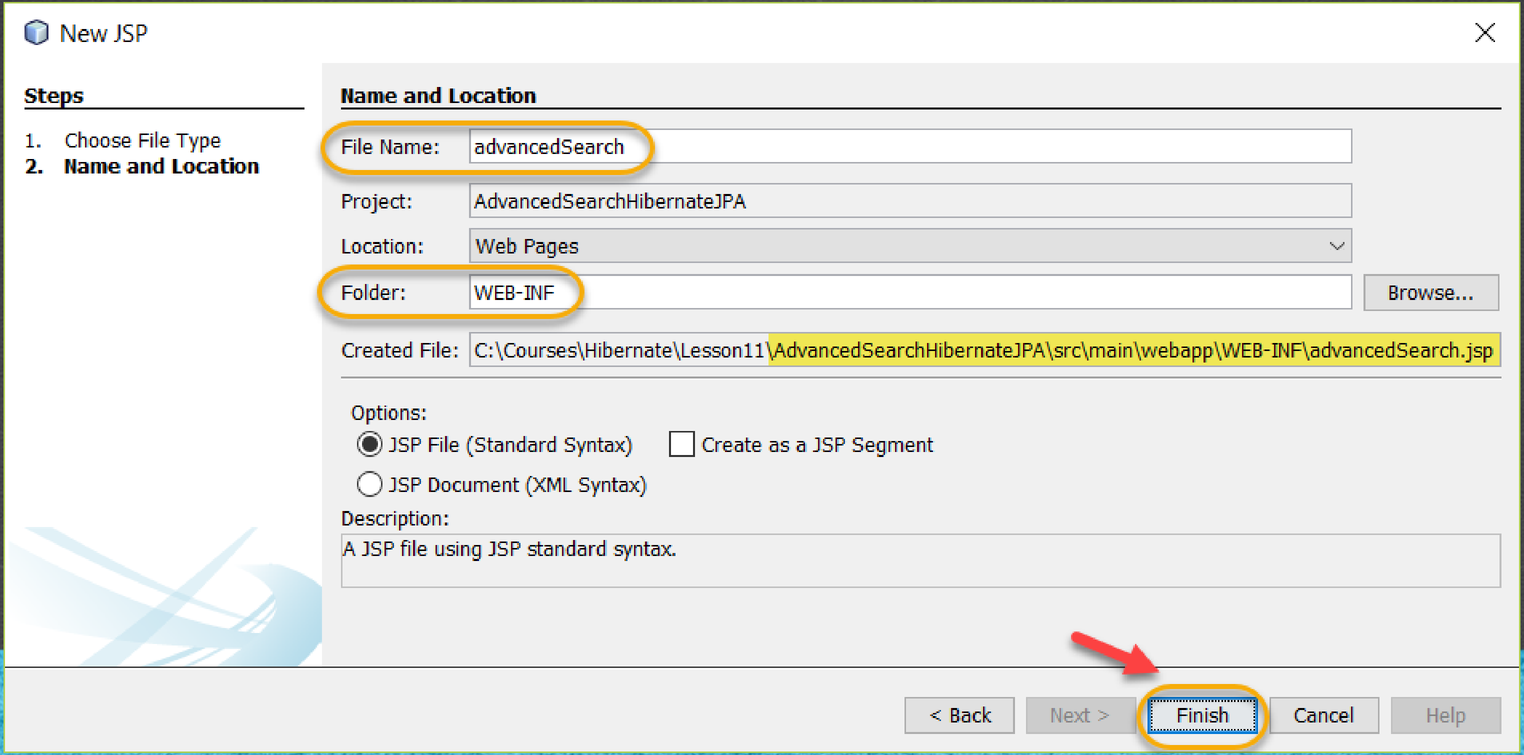
11. CREATE A JSP

Create the advancedSearch.jsp:



11. CREATE A JSP

Create the advancedSearch.jsp:



New JSP

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name: advancedSearch

Project: AdvancedSearchHibernateJPA

Location: Web Pages

Folder: WEB-INF

Created File: C:\Courses\Hibernate\Lesson11\AdvancedSearchHibernateJPA\src\main\webapp\WEB-INF\advancedSearch.jsp

Options:

[● JSP File (Standard Syntax)] ☐ Create as a JSP Segment

○ JSP Document (XML Syntax)

Description:

A JSP file using JSP standard syntax.

12. MODIFY THE CODE

advancedSearch.jsp:

Click to download

```
<!DOCTYPE html>
<html>
    <head>
        <title>Advanced Search</title>
    </head>
    <body>
        <form action="${pageContext.request.contextPath}/ServletSearch" method="post">
           Advanced Search of Students
            <br />
            <br />
            <fieldset>
                <legend>
                    Search by Student
                </legend>
                Student's name:
                <input type="text" name="studentName" size="50">
            </fieldset>
            <fieldset>
                <legend>
                    <br>
                    Search by Address
                </legend>
                Street Name:
                <input type="text" name="streetName" size="50">
            </fieldset>
```

12. MODIFY THE CODE

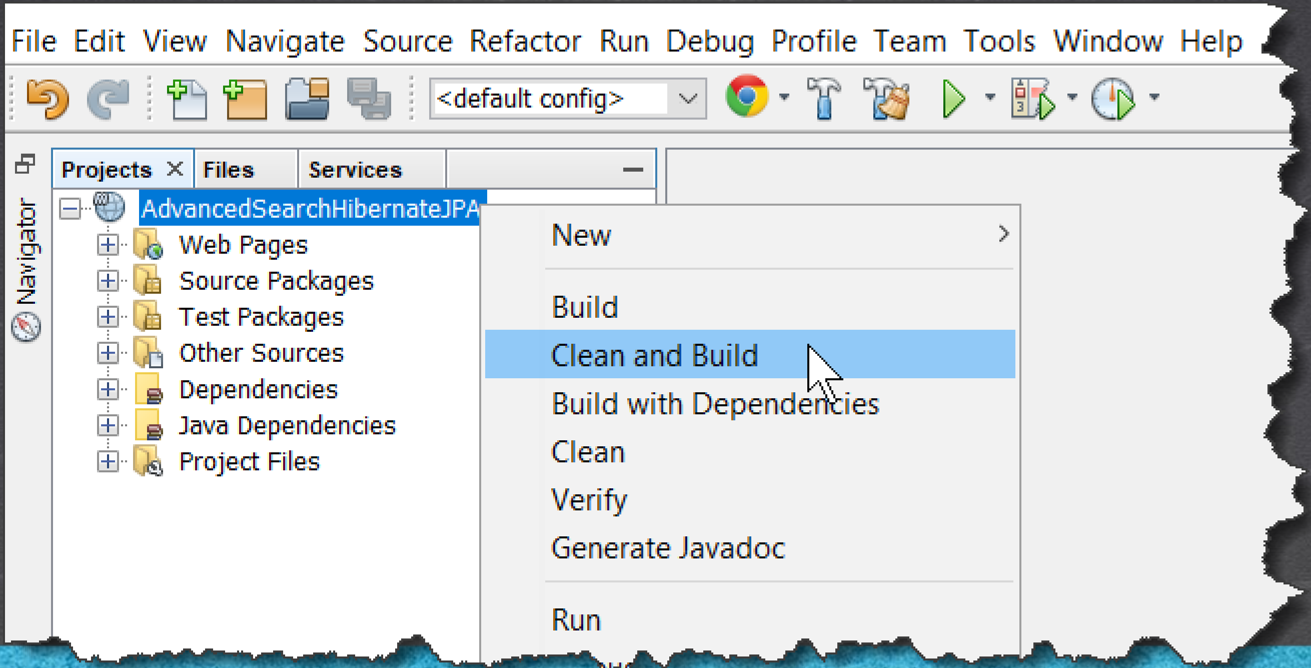
advancedSearch.jsp:

Click to download

```
            <fieldset>
                <legend>
                    <br>
                    Search by Course
                </legend>
                Course Name:
                <input type="text" name="courseName" size="50">
            </fieldset>
            <br/>
            <input type="submit" value="Search">
        </form>
    </body>
</html>
```

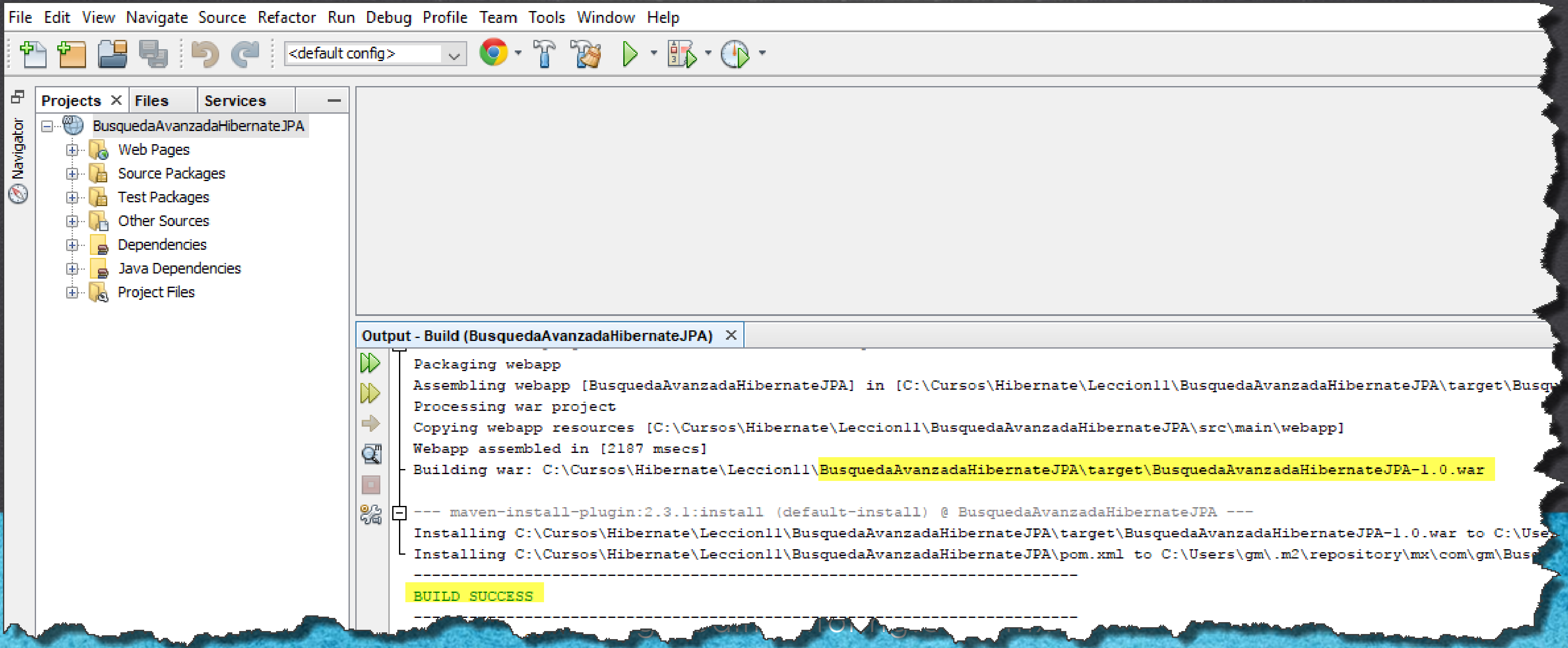
13. EXECUTE CLEAN & BUILD

We do Clean & Build to get the latest version of each file:



13. EXECUTE CLEAN & BUILD

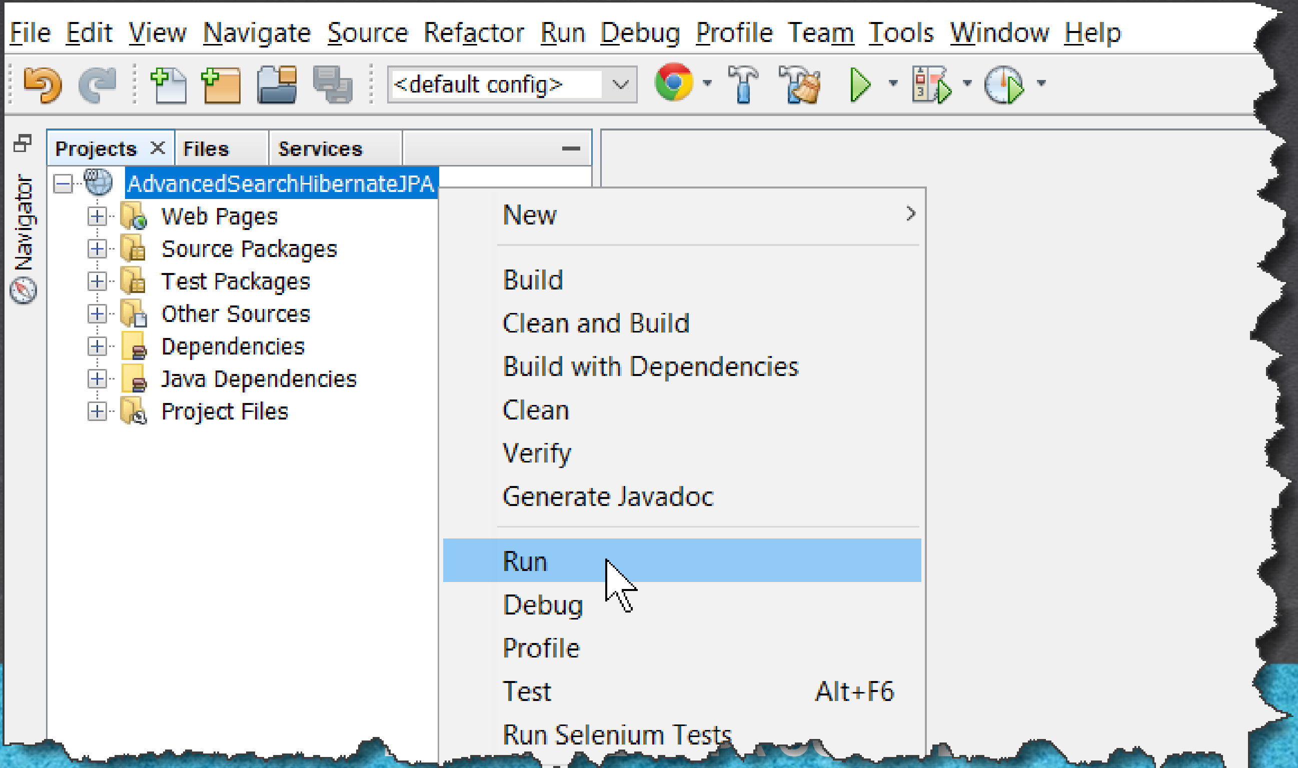
We do Clean & Build to get the latest version of each file:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
Files
Services
Navigator
BusquedaAvanzadaHibernateJPA
Web Pages
Source Packages
Test Packages
Other Sources
Dependencies
Java Dependencies
Project Files
Output - Build (BusquedaAvanzadaHibernateJPA)

```
Packaging webapp
Assembling webapp [BusquedaAvanzadaHibernateJPA] in [C:\Cursos\Hibernate\Leccion11\BusquedaAvanzadaHibernateJPA\target\Busq
Processing war project
Copying webapp resources [C:\Cursos\Hibernate\Leccion11\BusquedaAvanzadaHibernateJPA\src\main\webapp]
Webapp assembled in [2187 msecs]
Building war: C:\Cursos\Hibernate\Leccion11\BusquedaAvanzadaHibernateJPA\target\BusquedaAvanzadaHibernateJPA-1.0.war

--- maven-install-plugin:2.3.1:install (default-install) @ BusquedaAvanzadaHibernateJPA ---
Installing C:\Cursos\Hibernate\Leccion11\BusquedaAvanzadaHibernateJPA\target\BusquedaAvanzadaHibernateJPA-1.0.war to C:\Use
Installing C:\Cursos\Hibernate\Leccion11\BusquedaAvanzadaHibernateJPA\pom.xml to C:\Users\gm\.m2\repository\mx\com\gm\Bus
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

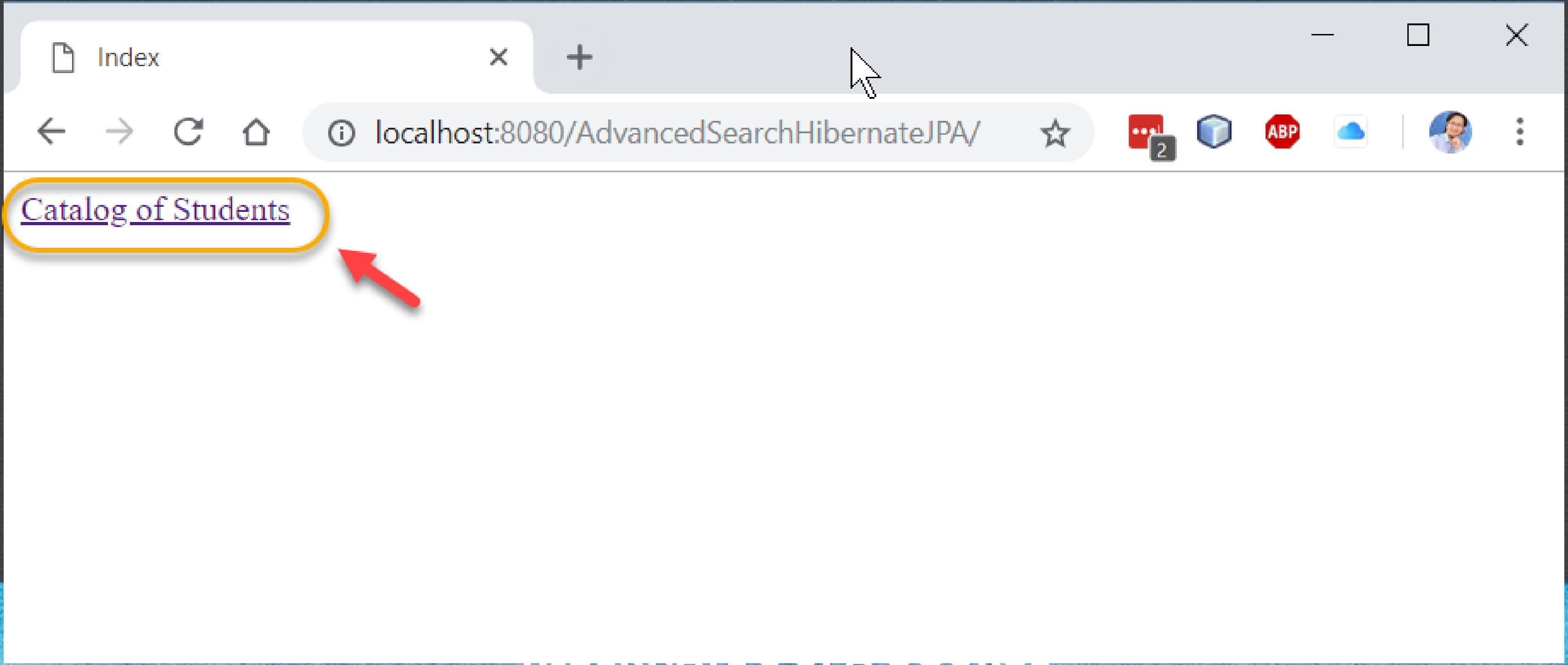
14. EXECUTE THE PROJECT

Execute the project:



14. EXECUTE THE PROJECT

Execute the project:

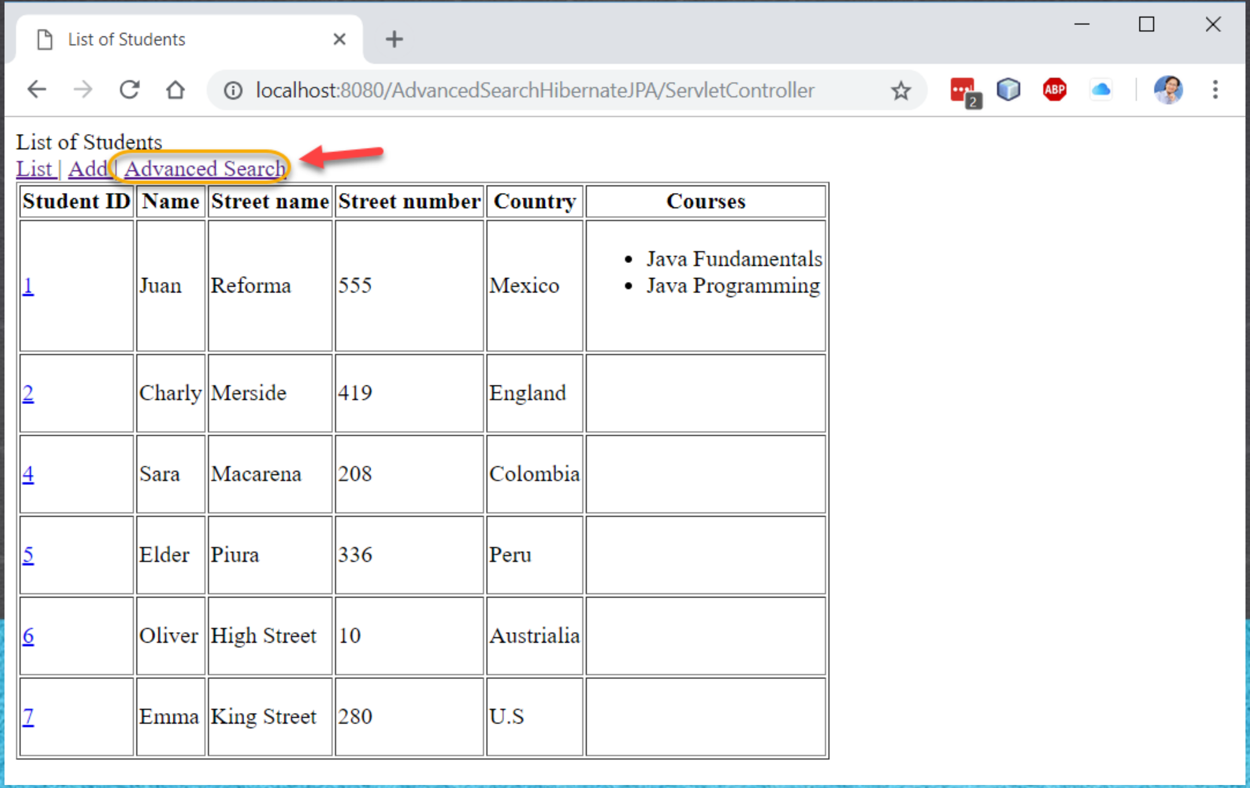

Index

localhost:8080/AdvancedSearchHibernateJPA/

Catalog of Students

14. EXECUTE THE PROJECT

We execute the project (add some test data in the database similar to how they are shown):



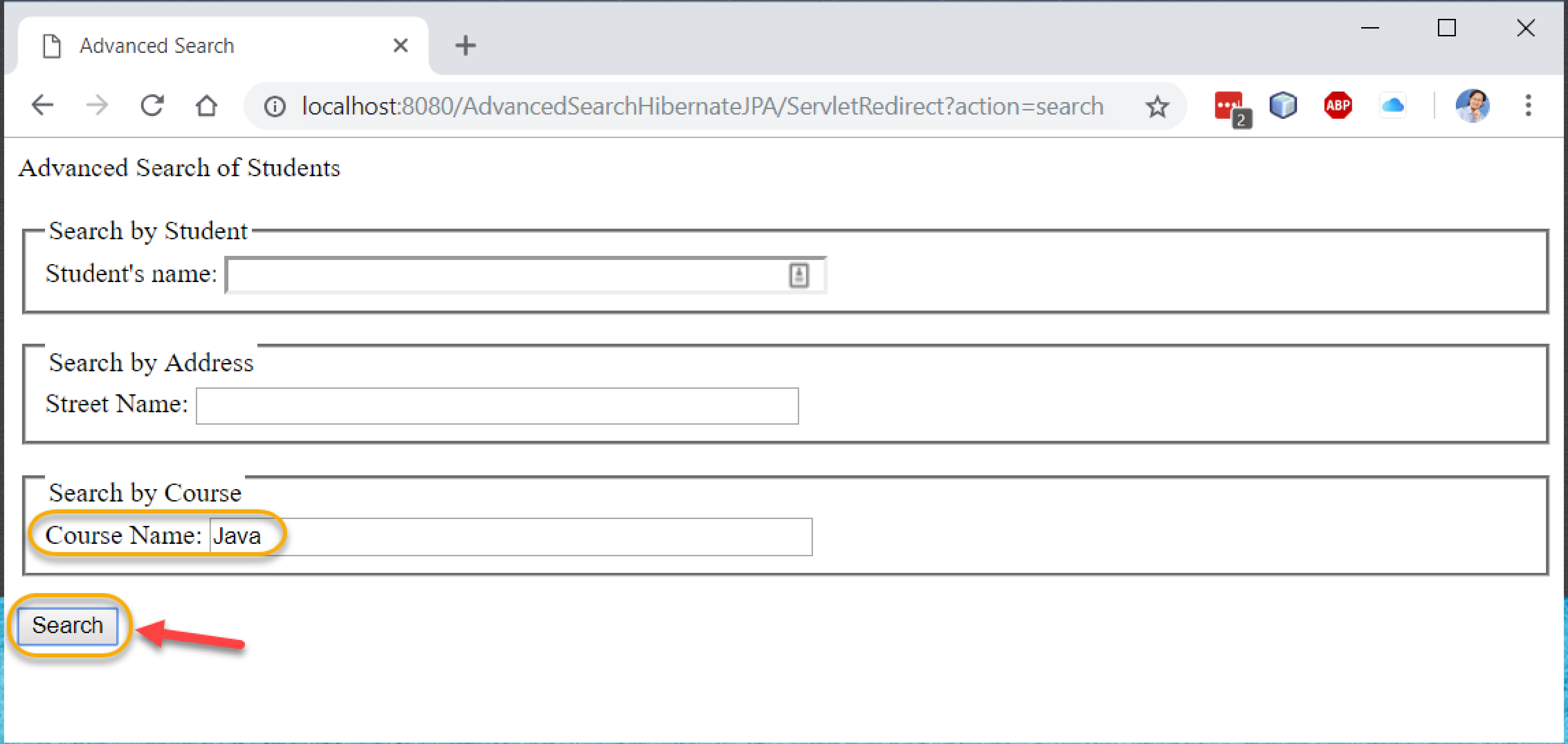
List of Students

List | Add | Advanced Search

Student ID	Name	Street name	Street number	Country	Courses
1	Juan	Reforma	555	Mexico	• Java Fundamentals • Java Programming
2	Charly	Merside	419	England	
4	Sara	Macarena	208	Colombia	
5	Elder	Piura	336	Peru	
6	Oliver	High Street	10	Austrialia	
7	Emma	King Street	280	U.S	

14. EXECUTE THE PROJECT

Execute the project:



Advanced Search

localhost:8080/AdvancedSearchHibernateJPA/ServletRedirect?action=search

Advanced Search of Students

Search by Student

Student's name:

Search by Address

Street Name:

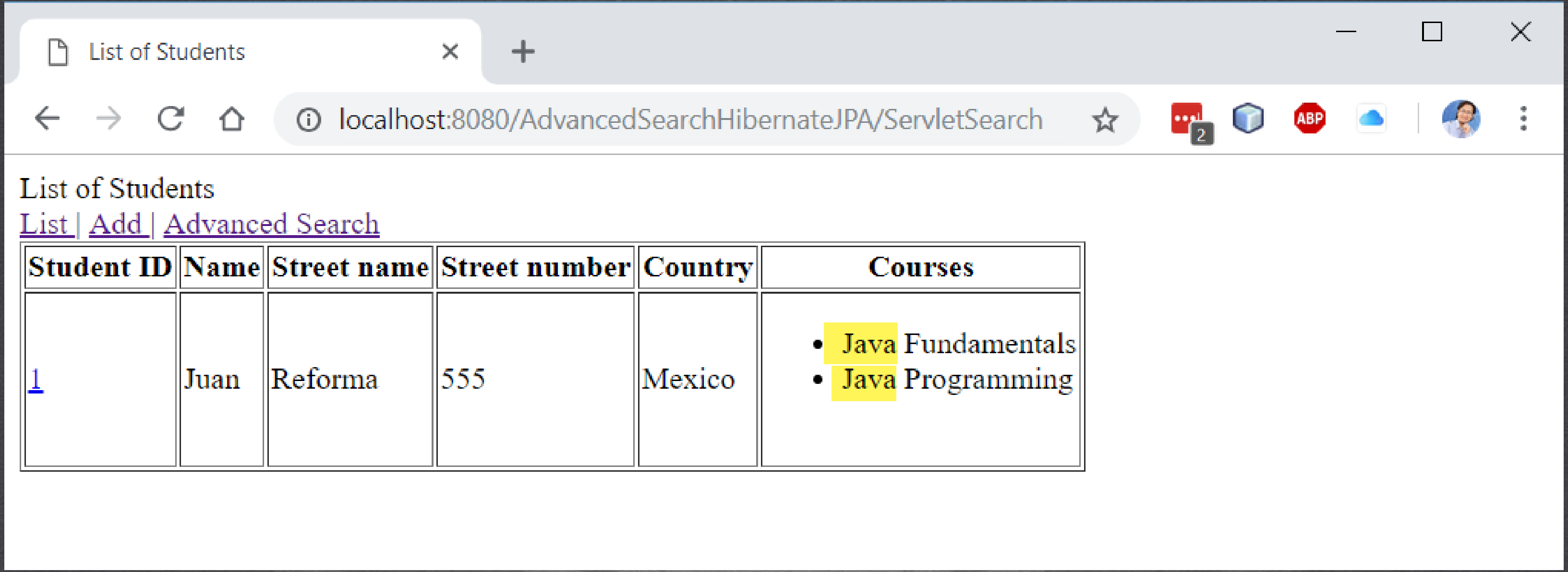
Search by Course

Course Name: Java

Search

14. EXECUTE THE PROJECT

Execute the project:



List of Students

List | Add | Advanced Search

Student ID	Name	Street name	Street number	Country	Courses
1	Juan	Reforma	555	Mexico	• Java Fundamentals • Java Programming

EXERCISE CONCLUSION

- With this exercise we have executed the advanced query using the Hibernate framework.
- At the moment we are using the Criteria API of Hibernate. After the version 5.2 of Hibernate this API has been depreciated, however there is no better option at the moment that we can recommend.
- It is possible to use the JPA Criteria API, but it does not include the QBE (Query by Example) API, so if you want to use it, you will have to validate field by field and concatenate each one of the criteria that you want to use.
- With this we conclude this course too. We hope you enjoy this course as much as us. We wait for you in more Global mentoring courses. See you soon ☺

ONLINE COURSE

HIBERNATE & JPA

By: Eng. Ubaldo Acosta

Global Mentoring

Experiencia y Conocimiento para tu vida

HIBERNATE & JPA COURSE

www.globalmentoring.com.mx