

CURSO HIBERNATE Y JPA

EJERCICIO

LABORATORIO BÚSQUEDA AVANZADA



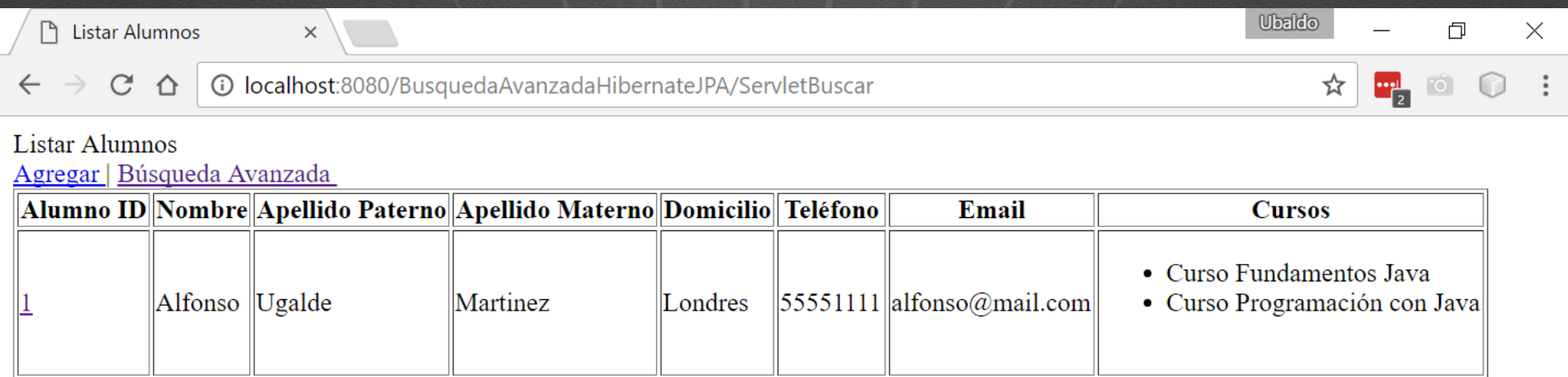
Experiencia y Conocimiento para tu vida

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

Utilizar el API de Criteria para crear el laboratorio de búsqueda avanzada. Al finalizar deberemos observar lo siguiente:



Ubaldo

localhost:8080/BusquedaAvanzadaHibernateJPA/ServletBuscar

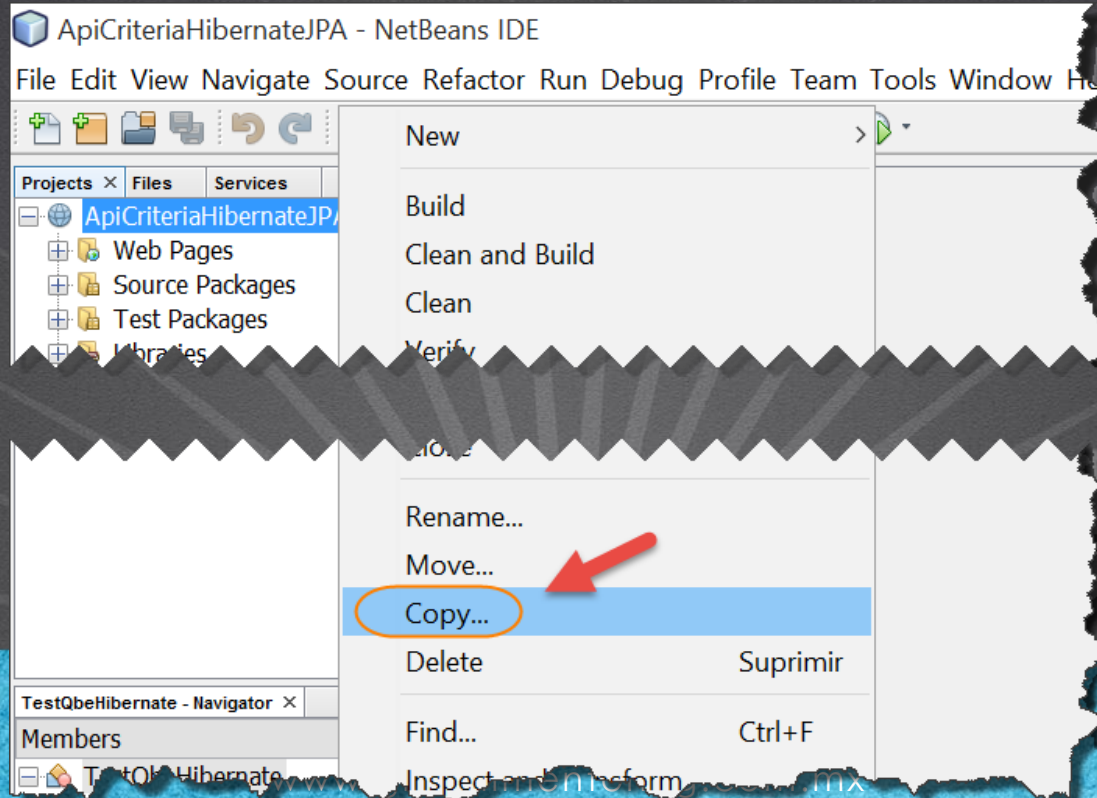
Listar Alumnos

[Agregar](#) | [Búsqueda Avanzada](#)

Alumno ID	Nombre	Apellido Paterno	Apellido Materno	Domicilio	Teléfono	Email	Cursos
1	Alfonso	Ugalde	Martinez	Londres	55551111	alfonso@mail.com	<ul style="list-style-type: none">• Curso Fundamentos Java• Curso Programación con Java

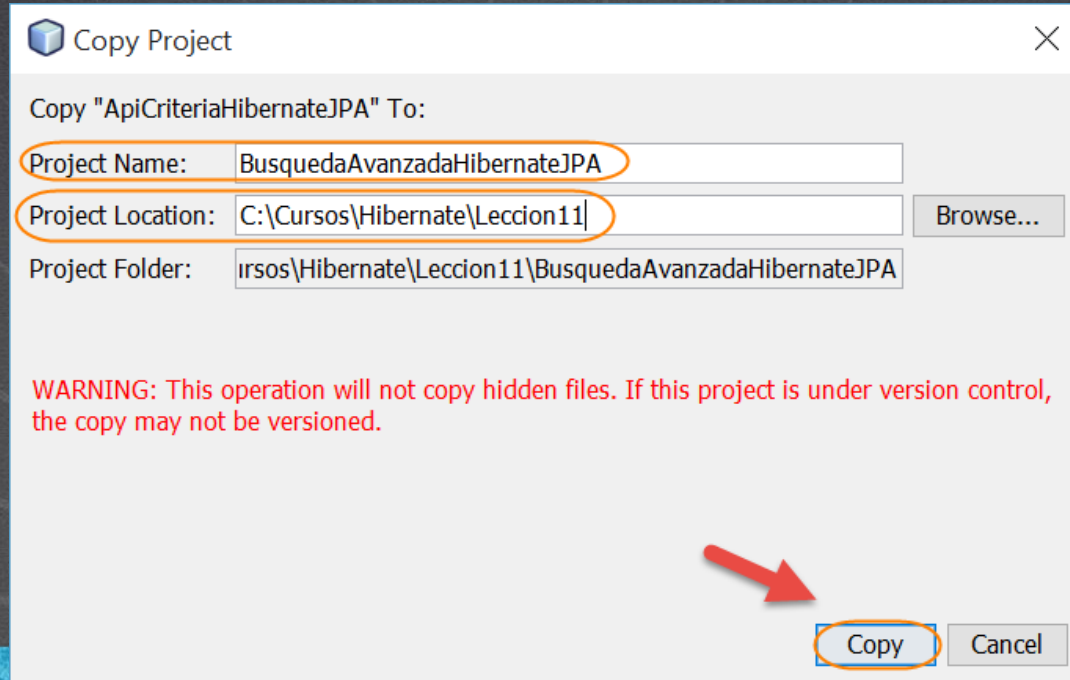
PASO 1. CREACIÓN DEL PROYECTO

Copiamos el proyecto partiendo de ApiCriteriaHibernateJPA :



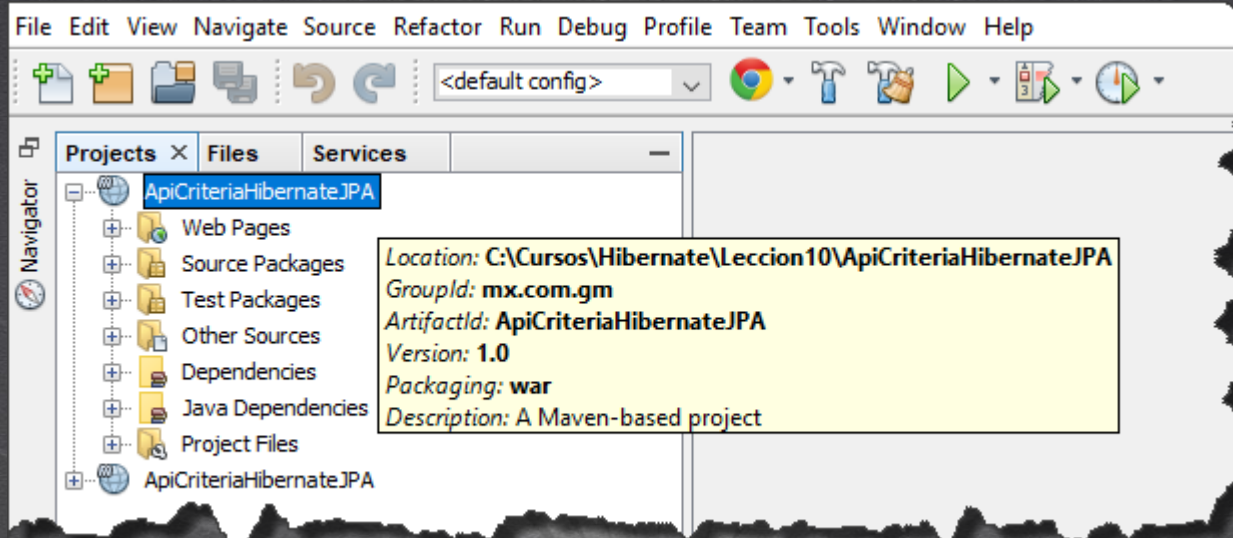
PASO 1. CREACIÓN DEL PROYECTO

Creamos el proyecto BusquedaAvanzadaHibernateJPA:



PASO 2. CERRAMOS EL PROYECTO NO USADO

Cerramos el proyecto que ya no usamos:

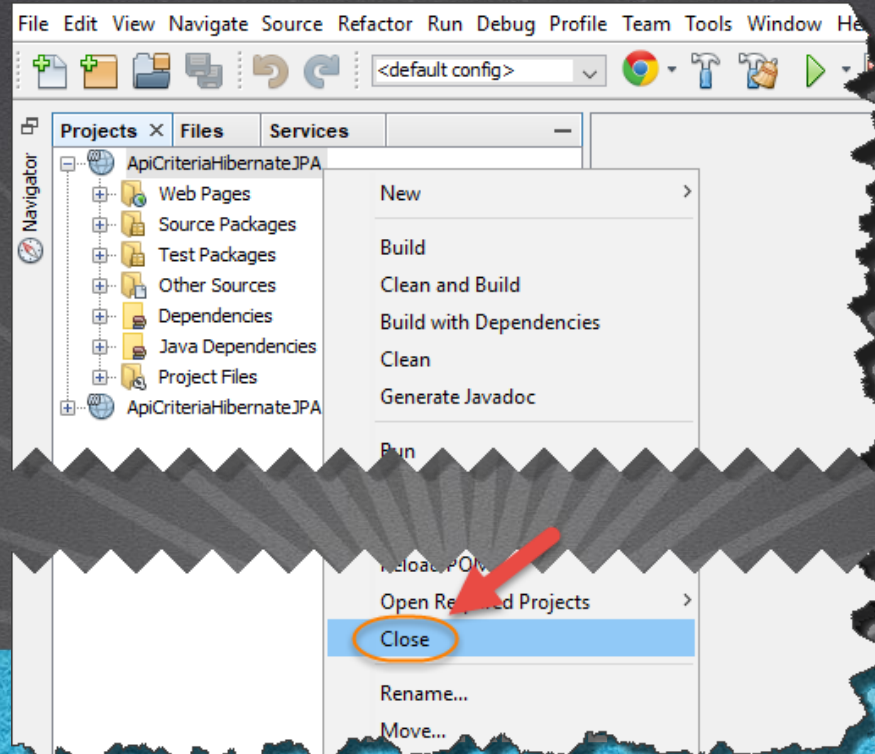


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

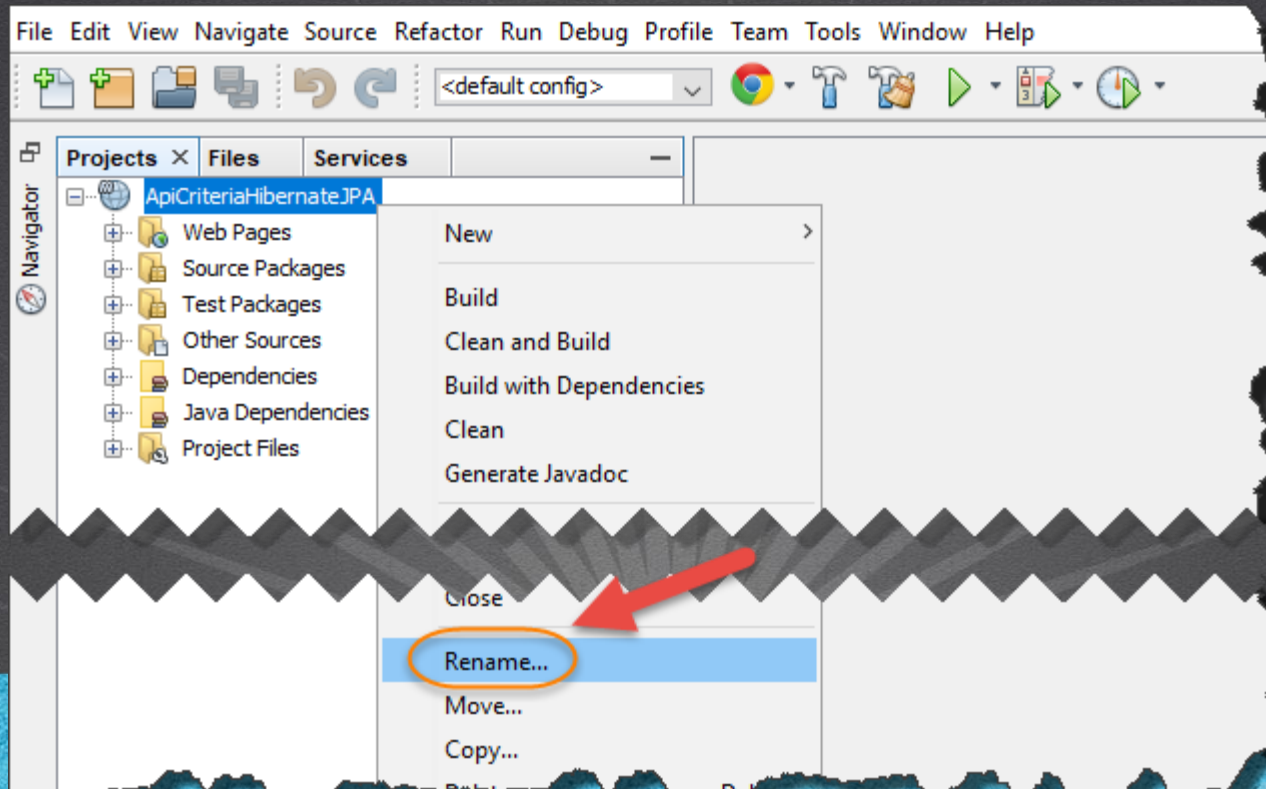
PASO 2. CERRAMOS EL PROYECTO NO USADO

Cerramos el proyecto que ya no usamos:



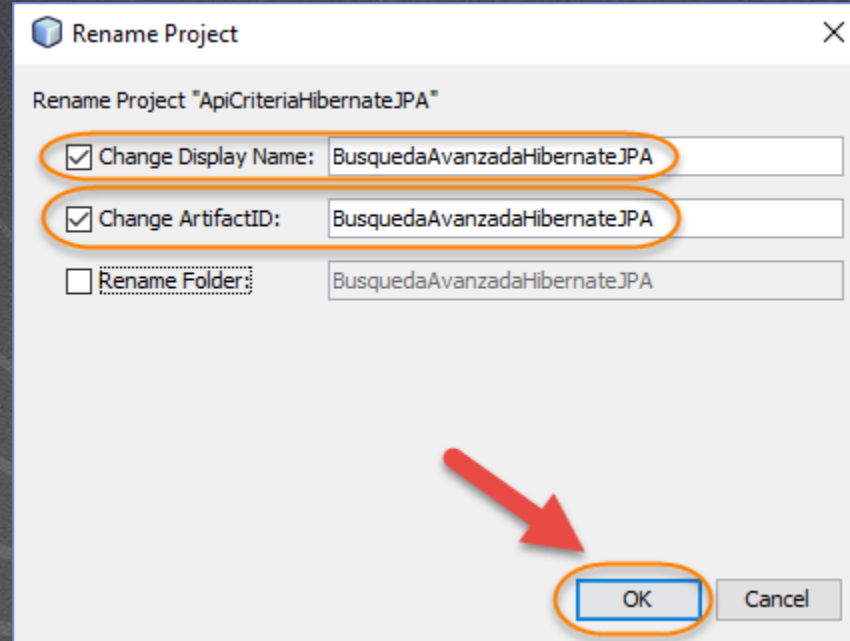
PASO 3. RENOMBRAMOS EL PROYECTO

Renombramos el proyecto:



PASO 3. RENOMBAMOS EL PROYECTO

Renombramos el proyecto:

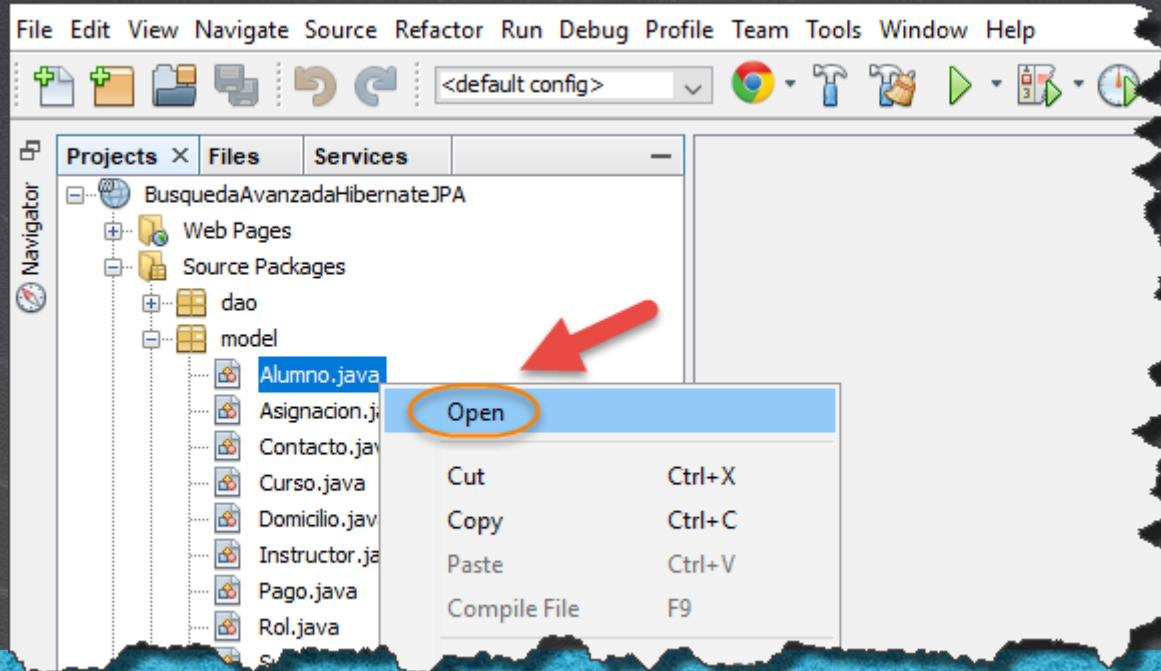


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 4. MODIFICAR UNA CLASE

Modificamos la clase de entidad Alumno.java:



CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
package model;

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

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

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_alumno")
    private Integer idAlumno;

    @Basic(optional = false)
    @Column(name = "nombre")
    private String nombre;

    @Basic(optional = false)
    @Column(name = "apellido_paterno")
    private String apellidoPaterno;
```

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
@Column(name = "apellido_materno")
private String apellidoMaterno;

@Column(name = "genero")
private String genero;

@Column(name = "fecha_nacimiento")
@Temporal(TemporalType.TIMESTAMP)
private Date fechaNacimiento;

@Basic(optional = false)
@Column(name = "version")
private int version;

@Basic(optional = false)
@Column(name = "deleted")
private int deleted;

@OneToMany(mappedBy = "alumno")
private List<Asignacion> asignaciones;

@JoinColumn(name = "id_contacto", referencedColumnName = "id_contacto")
@ManyToOne(cascade = CascadeType.ALL)
private Contacto contacto;

@JoinColumn(name = "id_domicilio", referencedColumnName = "id_domicilio")
@ManyToOne(cascade = CascadeType.ALL)
private Domicilio domicilio;
```

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
@JoinColumn(name = "id_usuario", referencedColumnName = "id_usuario")
@ManyToOne
private Usuario usuario;

public Alumno() {
}

public Alumno(Integer idAlumno) {
    this.idAlumno = idAlumno;
}

public Alumno(Integer idAlumno, String nombre, String apellidoPaterno, int version, int deleted) {
    this.idAlumno = idAlumno;
    this.nombre = nombre;
    this.apellidoPaterno = apellidoPaterno;
    this.version = version;
    this.deleted = deleted;
}

public Integer getIdAlumno() {
    return idAlumno;
}

public void setIdAlumno(Integer idAlumno) {
    this.idAlumno = idAlumno;
}
```

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
public String getNombre() {  
    return nombre;  
}  
  
public void setNombre(String nombre) {  
    this.nombre = nombre;  
}  
  
public String getApellidoPaterno() {  
    return apellidoPaterno;  
}  
  
public void setApellidoPaterno(String apellidoPaterno) {  
    this.apellidoPaterno = apellidoPaterno;  
}  
  
public String getApellidoMaterno() {  
    return apellidoMaterno;  
}  
  
public void setApellidoMaterno(String apellidoMaterno) {  
    this.apellidoMaterno = apellidoMaterno;  
}  
  
public String getGenero() {  
    return genero;  
}
```

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
public void setGenero(String genero) {  
    this.genero = genero;  
}  
  
public Date getFechaNacimiento() {  
    return fechaNacimiento;  
}  
  
public void setFechaNacimiento(Date fechaNacimiento) {  
    this.fechaNacimiento = fechaNacimiento;  
}  
  
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;  
}
```

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
public List<Asignacion> getAsignaciones() {  
    return asignaciones;  
}  
  
public void setAsignaciones(List<Asignacion> asignaciones) {  
    this.asignaciones = asignaciones;  
}  
  
public Contacto getContacto() {  
    return contacto;  
}  
  
public void setContacto(Contacto contacto) {  
    this.contacto = contacto;  
}  
  
public Domicilio getDomicilio() {  
    return domicilio;  
}  
  
public void setDomicilio(Domicilio domicilio) {  
    this.domicilio = domicilio;  
}  
  
public Usuario getUsuario() {  
    return usuario;  
}
```

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

```
public void setUsuario(Usuario usuario) {
    this.usuario = usuario;
}

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

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

Clic para descargar
el código

PASO 4. MODIFICAMOS EL CÓDIGO

Archivo Alumno.java:

Clic para descargar el código

```
@Override
public String toString() {
    return "Alumno{" + "idAlumno=" + idAlumno + ", nombre=" + nombre + ", apellidoPaterno=" + apellidoPaterno + ",
apellidoMaterno=" + apellidoMaterno + ", genero=" + genero + ", fechaNacimiento=" + fechaNacimiento + ", version=" + version
+ ", deleted=" + deleted + ", contacto=" + contacto + ", domicilio=" + domicilio + ", usuario=" + usuario + '}';
}

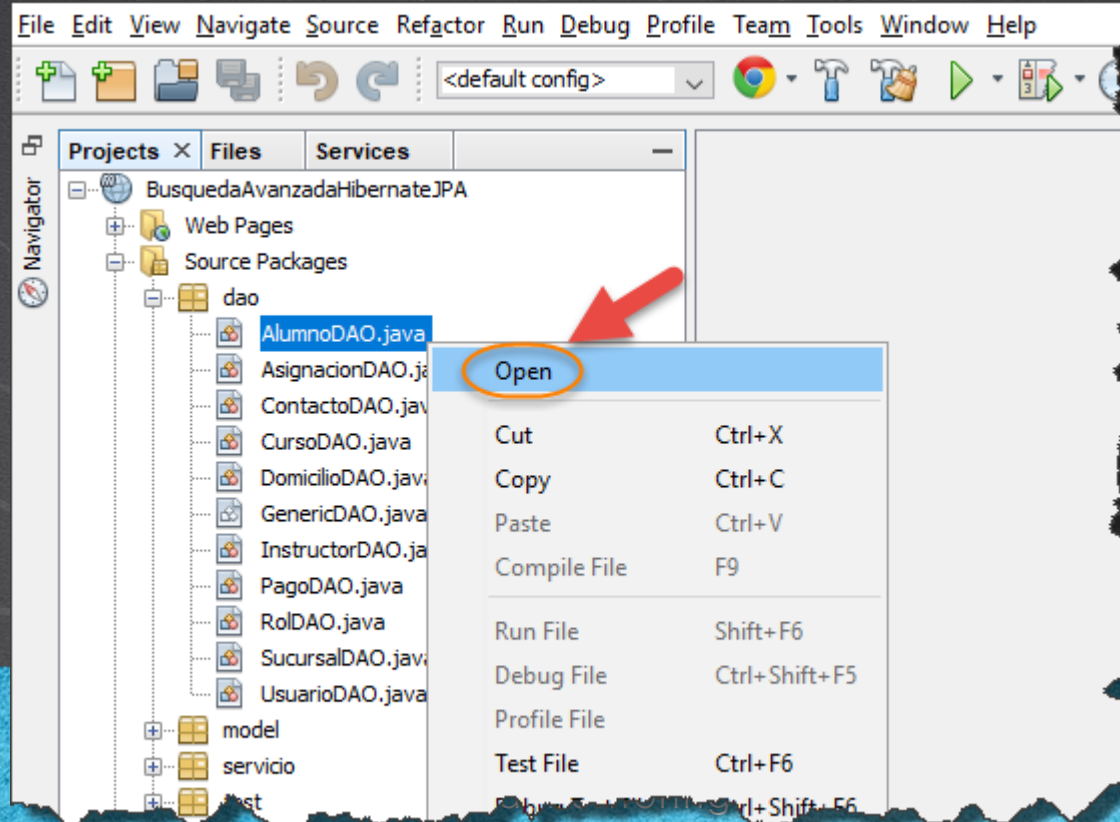
}
```

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 5. MODIFICAR UNA CLASE

Modificamos la clase de entidad AlumnoDAO.java:



PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

```
package dao;

import java.util.*;
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 AlumnoDAO extends GenericDAO {

    Logger log = LogManager.getRootLogger();

    public void listar() {
        // Consulta a ejecutar
        // No necesitamos crear una nueva transaccion
        String hql = "SELECT a FROM Alumno a";
        em = getEntityManager();
        Query query = em.createQuery(hql);
        List<Alumno> list = query.getResultList();
        for (Alumno a : list) {
            System.out.println(a);
        }
    }
}
```

Clic para descargar
el código

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

```
public List<Alumno> listarAlumnos() {
    String hql = "SELECT a FROM Alumno a";
    em = getEntityManager();
    Query query = em.createQuery(hql);
    return query.getResultList();
}

public void insertar(Alumno alumno) {
    try {
        em = getEntityManager();
        // Iniciamos una transaccion
        em.getTransaction().begin();
        // Insertamos nuevo objeto
        em.persist(alumno);
        // Terminamos la transaccion
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error al insertar objeto:" + ex.getMessage());
        // ex.printStackTrace();
    }
}
```

Clic para descargar
el código

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

```
public void actualizar(Alumno alumno) {
    try {
        em = getEntityManager();
        // Iniciamos una transaccion
        em.getTransaction().begin();
        // Actualizamos al objeto
        em.merge(alumno);
        // Terminamos la transaccion
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error al actualizar objeto:" + ex.getMessage());
        // ex.printStackTrace();
    }
}

public void eliminar(Alumno alumno) {
    try {
        em = getEntityManager();
        // Iniciamos una transaccion
        em.getTransaction().begin();
        // Sincronizamos y eliminamos
        em.remove(em.merge(alumno));
        // Terminamos la transaccion
        em.getTransaction().commit();
    } catch (Exception ex) {
        System.out.println("Error al eliminar objeto:" + ex.getMessage());
        // ex.printStackTrace();
    }
}
```

Clic para descargar
el código

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

Clic para descargar
el código

```
public Alumno buscarPorId(Alumno alumno) {
    em = getEntityManager();
    return em.find(Alumno.class, alumno.getIdAlumno());
}

public List buscarAlumnosPorCriterios(Map criterios) {
    log.debug("finding Alumno instances by different criteria");
    List lista = null;
    //1. Obtenemos los criterios uno por uno según la búsqueda que programamos
    Alumno alumnoDTO = null;
    Contacto contactoDTO = null;
    Domicilio domicilioDTO = null;
    Curso cursoDTO = null;

    if (criterios != null) {
        alumnoDTO = (Alumno) criterios.get("alumno");
        contactoDTO = (Contacto) criterios.get("contacto");
        domicilioDTO = (Domicilio) criterios.get("domicilio");
        cursoDTO = (Curso) criterios.get("curso");
    }
    try {
        //2. Creamos el objeto criteria para ejecutar el query
        //El método API de Criteria de Hibernate ha sido depreciado en la version 5.2
        //Sin embargo JPA no ofrece de momento una opción a QueryByExample
        //Así que hasta que JPA no ofrezca una opción para este tipo de queries pueden usar
        //esta versión, y en cuanto exista una opción viable para estos queries
        //se actualizará este código
        Criteria criteria = em.unwrap(Session.class).getSession().createCriteria(Alumno.class);
```

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

```
//3. Agregamos los criterios recibidos
if (alumnoDTO != null) {

    //Utilizamos un DTO de Alumno y un QBE (Query By Example)
    //Lo envolvemos en un Example
    Example exampleAlumno = Example.create(alumnoDTO).enableLike(MatchMode.ANYWHERE);

    //agregamos el example al query de criteria
    criteria.add(exampleAlumno);
}
if (contactoDTO != null) {

    //Utilizamos un DTO de Contacto y un QBE
    //Lo envolvemos en un Example
    Example exampleContacto = Example.create(contactoDTO).enableLike(MatchMode.ANYWHERE);

    //Agregamos la restriccion del contacto
    criteria.createCriteria("contacto", JoinType.LEFT_OUTER_JOIN)
        .setFetchMode("contacto", FetchMode.JOIN)
        .add(exampleContacto);
}
```

Clic para descargar
el código

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

Clic para descargar
el código

```
if (domicilioDTO != null) {  
  
    //Utilizamos un DTO de Domicilio y un QBE  
    //Lo envolvemos en un Example  
    Example exampleDomicilio = Example.create(domicilioDTO).enableLike(MatchMode.ANYWHERE);  
  
    //Agregamos la restriccion del domicilio  
    criteria.createCriteria("domicilio", JoinType.LEFT_OUTER_JOIN)  
        .setFetchMode("domicilio", FetchMode.JOIN)  
        .add(exampleDomicilio);  
}  
if (cursoDTO != null) {  
  
    //Lo envolvemos en un Example  
    Example exampleCurso = Example.create(cursoDTO).enableLike(MatchMode.ANYWHERE);  
  
    //Agregamos la restriccion del curso, primero accediendo a asignaciones y luego a curso  
    //agregando un criterio dentro de otro (agregar un criterio significa que se vuelve  
    //la tabla pivote o root en ese momento del query)  
    criteria.createCriteria("asignaciones", JoinType.LEFT_OUTER_JOIN)  
        .setFetchMode("asignaciones", FetchMode.JOIN)  
        .createCriteria("curso", JoinType.LEFT_OUTER_JOIN)  
        .setFetchMode("curso", FetchMode.JOIN)  
        .add(exampleCurso);  
}
```

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo AlumnoDAO.java:

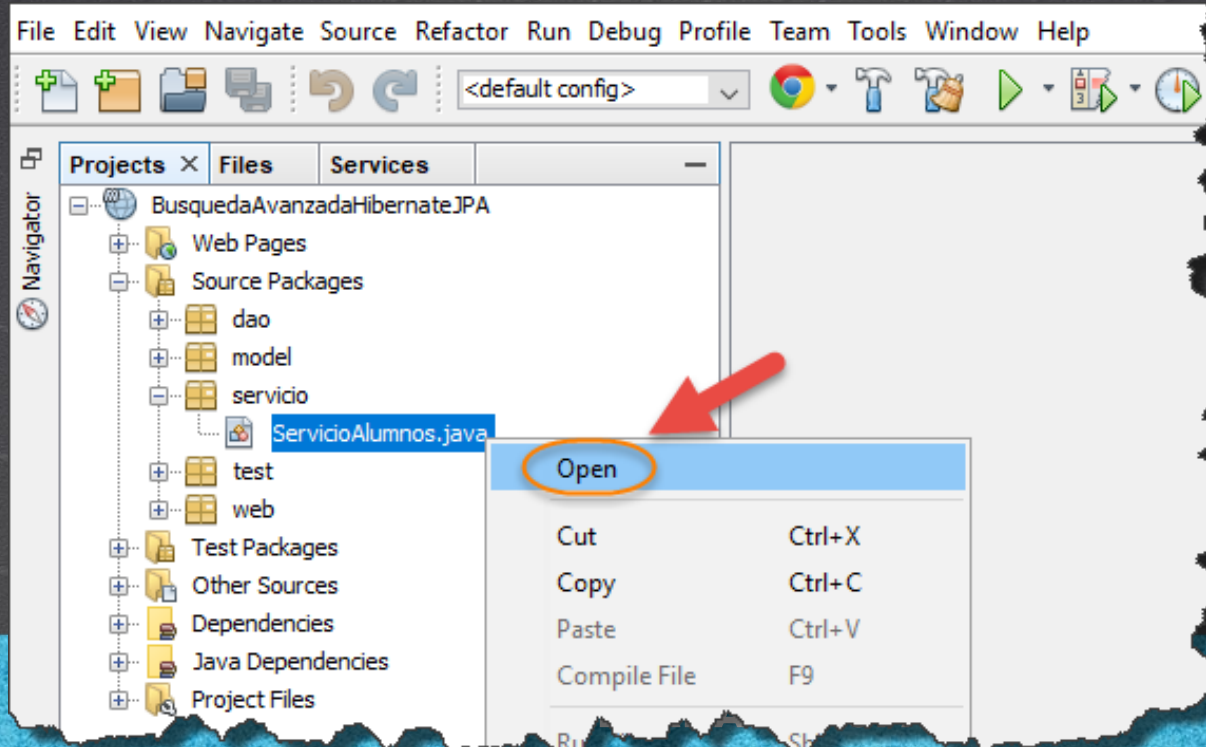
```
//Restringe a obtener solo los elementos distintos
criteria.setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY);

//4. Obtenemos la lista
lista = criteria.list();
} catch (RuntimeException re) {
    log.error("find by example failed", re);
    throw re;
}
return lista;
}
```

Clic para descargar
el código

PASO 6. MODIFICAR UNA CLASE

Modificamos la clase de entidad ServicioAlumnos.java:



PASO 6. MODIFICAMOS EL CÓDIGO

Archivo ServicioAlumnos.java:

```
package servicio;

import dao.AlumnoDAO;
import model.Alumno;
import java.util.List;
import java.util.Map;

public class ServicioAlumnos {

    // El Dao se asocia con la clase
    AlumnoDAO alumnoDao = new AlumnoDAO();

    // Esta clase se comunica con los DAO's que necesite
    public List<Alumno> listarAlumnos() {
        return alumnoDao.listarAlumnos();
    }

    public boolean guardarAlumno(Alumno alumno) {
        if (alumno != null && alumno.getIdAlumno() == null) {
            //insert
            alumnoDao.insertar(alumno);
        } else {
            //update
            alumnoDao.actualizar(alumno);
        }

        return true; // si nada fallo, regresamos verdadero
    }
}
```

Clic para descargar
el código

PASO 6. MODIFICAMOS EL CÓDIGO

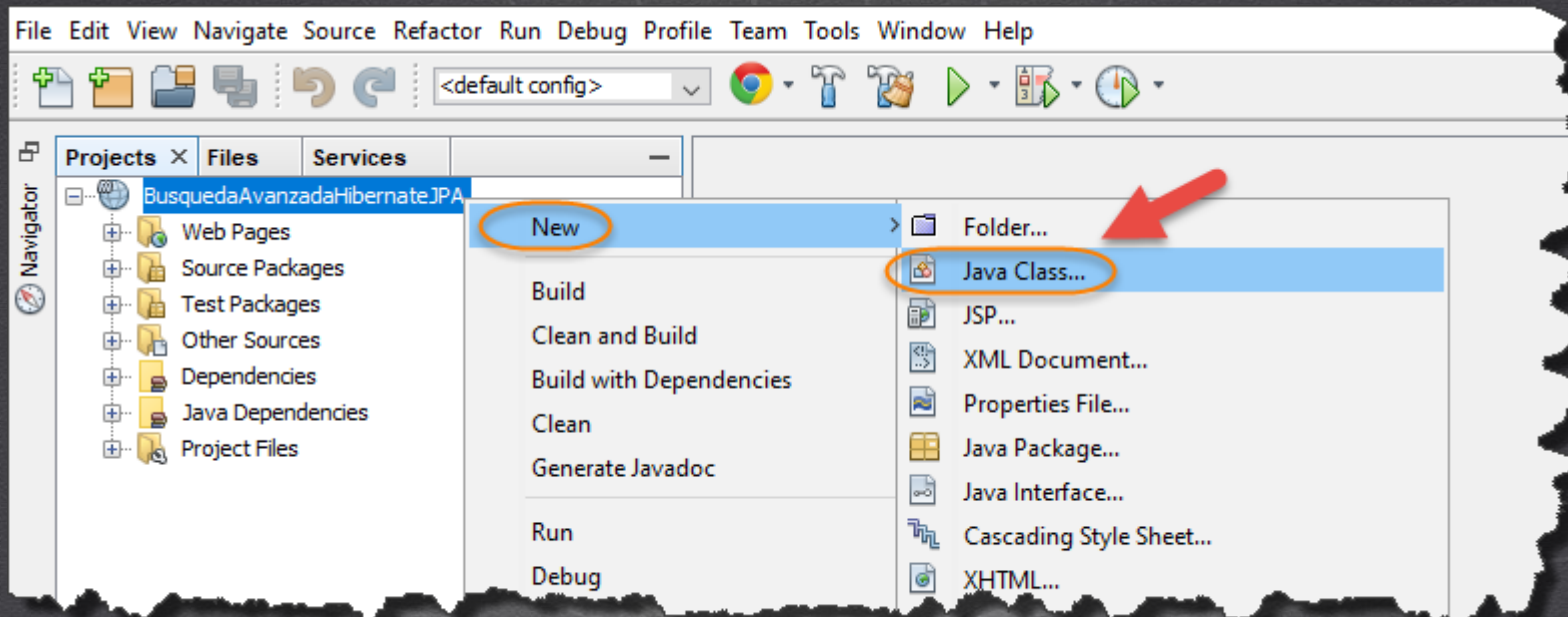
Archivo ServicioAlumnos.java:

Clic para descargar
el código

```
public boolean eliminarAlumno(Integer idAlumno) {  
    //delete  
    alumnoDao.eliminar(new Alumno(idAlumno));  
    return true; // si nada falla regresamos verdadero  
}  
  
public Alumno encontrarAlumno(Integer idAlumno) {  
    return alumnoDao.buscarPorId(new Alumno(idAlumno));  
}  
  
public List<Alumno> encontrarAlumnosPorCriterios(Map criterios) {  
    alumnoDao = new AlumnoDAO();  
    return alumnoDao.buscarAlumnosPorCriterios(criterios);  
}  
}
```

PASO 7. CREAR UNA CLASE

Creamos la clase ServletBuscar.java:



CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 7. CREAR UNA CLASE

Creamos la clase ServletBuscar.java:

New Java Class

Steps

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

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back Next > **Finish** Cancel Help

PASO 8. MODIFICAMOS EL CÓDIGO

Archivo ServletBuscar.java:

Clic para descargar
el código

```
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 servicio.ServicioAlumnos;

@WebServlet(name = "ServletBuscar", urlPatterns = {"/ServletBuscar"})
public class ServletBuscar extends HttpServlet {

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

        //1. Recuperamos los parametros de buqueda
        //Recuperamos los datos del alumno
        String nombreAlumno = request.getParameter("nombreAlumno");
        nombreAlumno = "".equals(nombreAlumno) ? null : nombreAlumno.trim();

        String apellidoPaterno = request.getParameter("apellidoPaterno");
        apellidoPaterno = "".equals(apellidoPaterno) ? null : apellidoPaterno.trim();
```

PASO 8. MODIFICAMOS EL CÓDIGO

Archivo ServletBuscar.java:

Clic para descargar
el código

```
String apellidoMaterno = request.getParameter("apellidoMaterno");
apellidoMaterno = "".equals(apellidoMaterno) ? null : apellidoMaterno.t

Alumno alumnoDTO = null;
if (nombreAlumno != null || apellidoPaterno != null || apellidoMaterno != null) {
    alumnoDTO = new Alumno();
    alumnoDTO.setNombre(nombreAlumno);
    alumnoDTO.setApellidoPaterno(apellidoPaterno);
    alumnoDTO.setApellidoMaterno(apellidoMaterno);
}

//Recuperamos los datos de Contacto
String telefono = request.getParameter("telefono");
telefono = "".equals(telefono) ? null : telefono.trim();

String email = request.getParameter("email");
email = "".equals(email) ? null : email.trim();

Contacto contactoDTO = null;
if (telefono != null || email != null) {
    contactoDTO = new Contacto();
    contactoDTO.setTelefono(telefono);
    contactoDTO.setEmail(email);
}
```

PASO 8. MODIFICAMOS EL CÓDIGO

Archivo ServletBuscar.java:

Clic para descargar
el código

```
//Recuperamos los datos del Domicilio
String calle = request.getParameter("calle");
calle = "".equals(calle) ? null : calle.trim();

Domicilio domicilioDTO = null;
if (calle != null) {
    domicilioDTO = new Domicilio();
    domicilioDTO.setCalle(calle);
}

String nombreCurso = request.getParameter("nombreCurso");
nombreCurso = "".equals(nombreCurso) ? null : nombreCurso.trim();
Curso cursoDTO = null;
if (nombreCurso != null) {
    cursoDTO = new Curso();
    cursoDTO.setNombre(nombreCurso);
}

//2. Agregamos los parametros a un mapa
//esto permite ir agregando mas filtros si se necesitaran
Map criterios = new HashMap();
criterios.put("alumno", alumnoDTO);
criterios.put("contacto", contactoDTO);
criterios.put("domicilio", domicilioDTO);
criterios.put("curso", cursoDTO);
```


PASO 8. MODIFICAMOS EL CÓDIGO

Archivo ServletBuscar.java:

Clic para descargar
el código

```
//3. Nos comunicamos con la capa de servicio
ServicioAlumnos servicioAlumnos = new ServicioAlumnos();

//4. Enviamos el mapa de parametros para crear el filtro
List<Alumno> alumnos = servicioAlumnos.encontrarAlumnosPorCriterios(criterios);

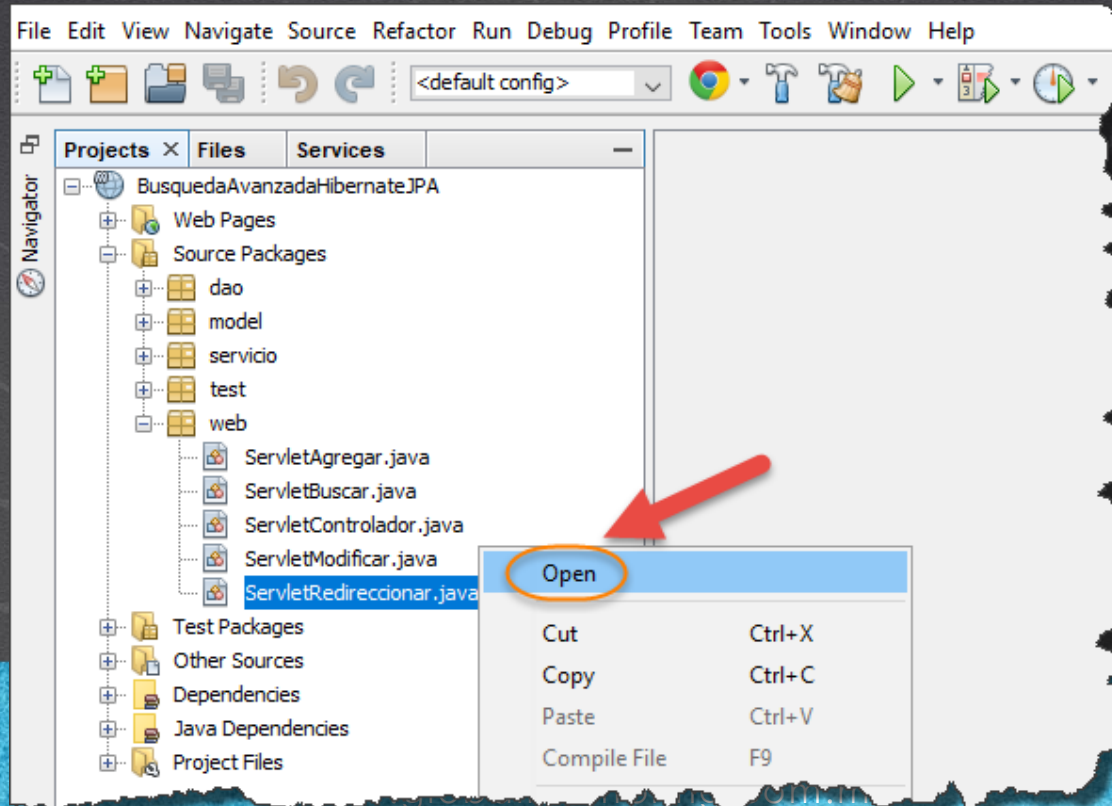
//3. Compartimos la informacion (Modelo) con la vista
request.setAttribute("alumnos", alumnos);

//4. Seleccionamos la vista a mostrar la info de alumnos
request.getRequestDispatcher("/WEB-INF/listarAlumnos.jsp").forward(request, response);
}

}
```

PASO 9. MODIFICAR UN SERVLET

Modificamos el ServletRedireccionar.java:



PASO 9. MODIFICAMOS EL CÓDIGO

Archivo ServletRedireccionar.java:

Clic para descargar
el código

```
package web;

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

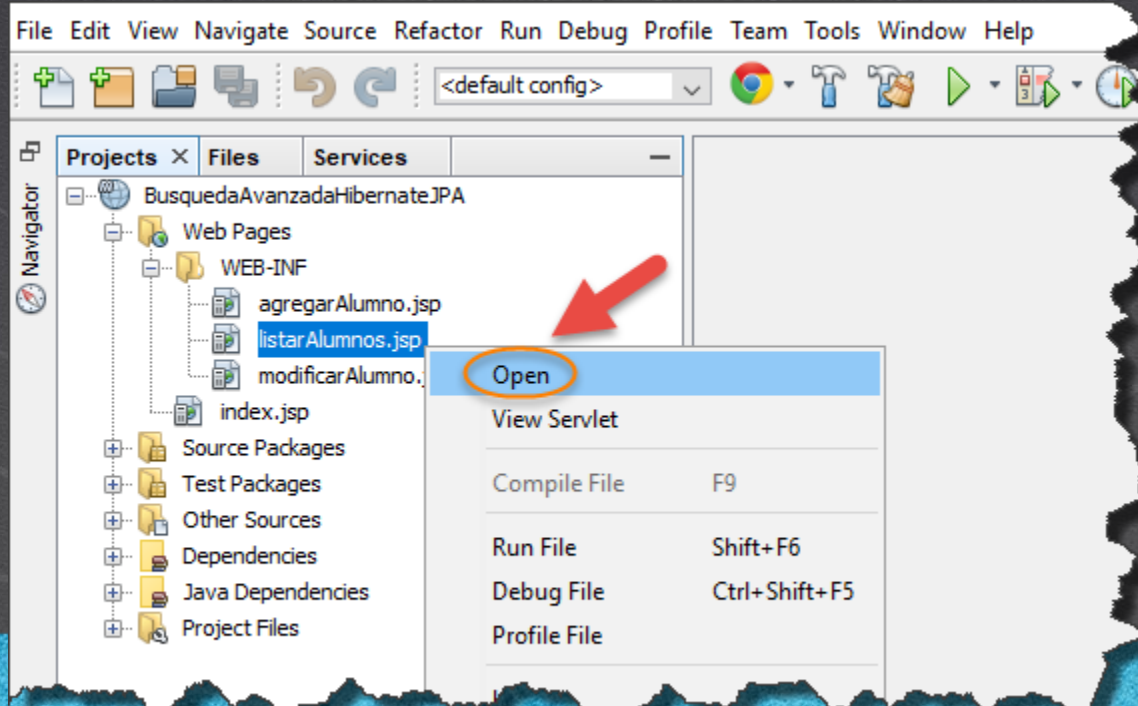
@WebServlet(name = "ServletRedireccionar", urlPatterns = {"/ServletRedireccionar"})
public class ServletRedireccionar extends HttpServlet {

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

        String accion = request.getParameter("accion");
        if ("agregar".equals(accion)) {
            //Redireccionamos a la pagina de Agregar Alumno
            request.getRequestDispatcher("WEB-INF/agregarAlumno.jsp").forward(request, response);
        } else if ("buscar".equals(accion)) {
            //Redireccionamos a la pagina de Búsqueda avanzada para usar criterios
            request.getRequestDispatcher("WEB-INF/busquedaAvanzada.jsp").forward(request, response);
        }
    }
}
```


PASO 10. MODIFICAR UN JSP

Modificar el archivo listarAlumnos.jsp:



PASO 10. MODIFICAMOS EL CÓDIGO

Archivo listarAlumnos.jsp:

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Listar Alumnos</title>
  </head>
  <body>
    Listar Alumnos
    <br />
    <a
      href="${pageContext.request.contextPath}/ServletControlador">
      Listar </a> |
    <a
      href="${pageContext.request.contextPath}/ServletRedireccionar?accion=agregar">
      Agregar </a> |
    <a
      href="${pageContext.request.contextPath}/ServletRedireccionar?accion=buscar">
      Búsqueda Avanzada </a>
    <br />
  </body>
</html>
```

Clic para descargar
el código

PASO 10. MODIFICAMOS EL CÓDIGO

Archivo listarAlumnos.jsp:

Clic para descargar
el código

```
<c:if test="${not empty alumnos}">
  <table border="1">
    <tr>
      <th>Alumno ID</th>
      <th>Nombre</th>
      <th>Apellido Paterno</th>
      <th>Apellido Materno</th>
      <th>Domicilio</th>
      <th>Teléfono</th>
      <th>Email</th>
      <th>Cursos</th>
    </tr>
    <c:forEach var="alumno" items="${alumnos}">
      <tr>
        <td>
          <a
            href="${pageContext.request.contextPath}/ServletModificar?idAlumno=${alumno.idAlumno}">
              ${alumno.idAlumno}
            </a>
        </td>
        <td>${alumno.nombre}</td>
        <td>${alumno.apellidoPaterno }</td>
        <td>${alumno.apellidoMaterno }</td>
        <td>${alumno.domicilio.calle }</td>
        <td>${alumno.contacto.telefono }</td>
        <td>${alumno.contacto.email1 }</td>
      </tr>
    </c:forEach>
  </table>
</c:if>
```

PASO 10. MODIFICAMOS EL CÓDIGO

Archivo listarAlumnos.jsp:

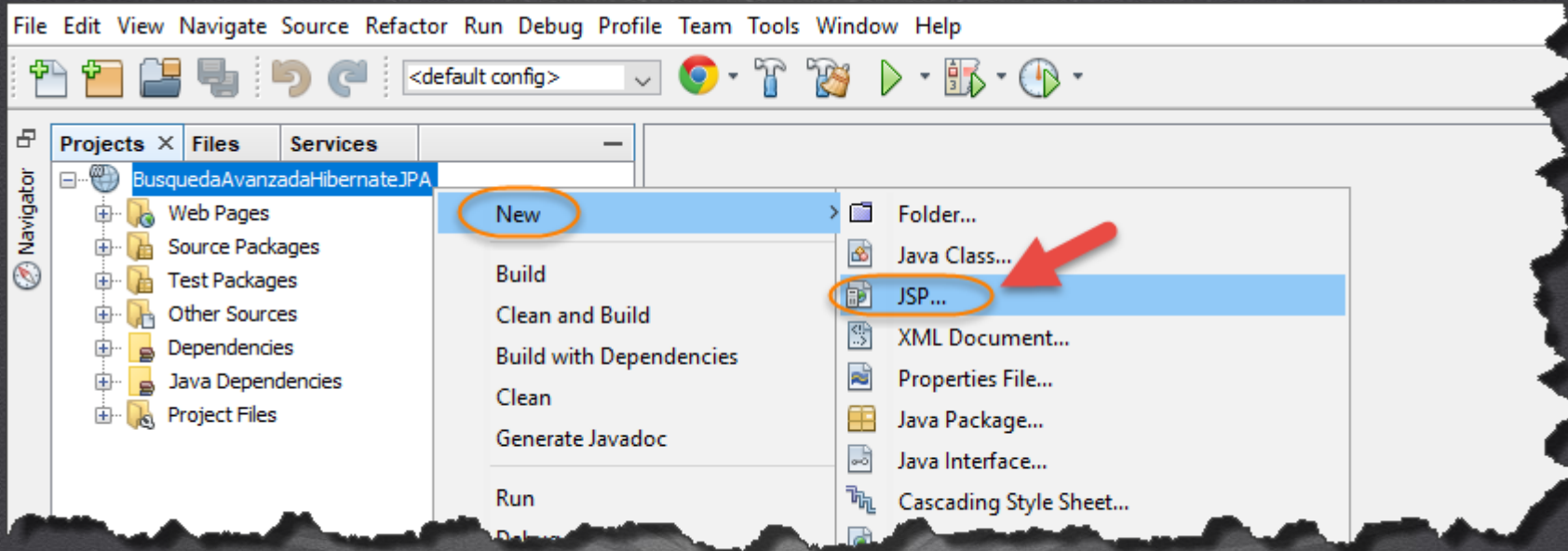
Clic para descargar
el código

```
        <td>
            <ul>
                <c:forEach var="asig" items="${alumno.asignaciones}">
                    <li>
                        ${asig.curso.nombre}
                    </li>
                </c:forEach>
                &nbsp;
            </ul>
        </td>

    </tr>
</c:forEach>
</table>
</c:if>
<c:if test="${empty alumnos}">
    <br/>
    Sin registros encontrados
</c:if>
</body>
</html>
```

PASO 11. CREAR UN JSP

Creamos el archivo busquedaAvanzada.jsp:



CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 11. CREAR UN JSP

Creamos el archivo busquedaAvanzada.jsp:

New JSP

Steps

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

Name and Location

File Name:

Project:

Location:

Folder:

Created File:

Options:

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

Description:

< Back Next > **Finish** Cancel Help

PASO 12. MODIFICAMOS EL CÓDIGO

Archivo busquedaAvanzada.jsp:

Clic para descargar
el código

```
<!DOCTYPE html>
<html>
  <head>
    <title>Busqueda Avanzada</title>
  </head>
  <body>
    <form action="${pageContext.request.contextPath}/ServletBuscar"
          method="post">
      Búsqueda Avanzada de Alumnos
      <br />
      <br />
      <fieldset>
        <legend>
          Búsqueda por Alumno
        </legend>
        Nombre Alumno:
        <input type="text" name="nombreAlumno" size="50">
        <br>
        Apellido Paterno:
        <input type="text" name="apellidoPaterno" size="50">
        <br>
        Apellido Materno:
        <input type="text" name="apellidoMaterno" size="50">
      </fieldset>
    </form>
  </body>
</html>
```

PASO 12. MODIFICAMOS EL CÓDIGO

Archivo busquedaAvanzada.jsp:

Clic para descargar
el código

```
<fieldset>
  <legend>
    <br>
    Búsqueda por datos de Contacto
  </legend>
  Teléfono:
  <input type="text" name="telefono" size="50">
  <br>
  Email:
  <input type="text" name="email" size="50">
</fieldset>
<fieldset>
  <legend>
    <br>
    Búsqueda por datos de Domicilio
  </legend>
  Calle:
  <input type="text" name="calle" size="50">
</fieldset>
```

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 12. MODIFICAMOS EL CÓDIGO

Archivo busquedaAvanzada.jsp:

Clic para descargar
el código

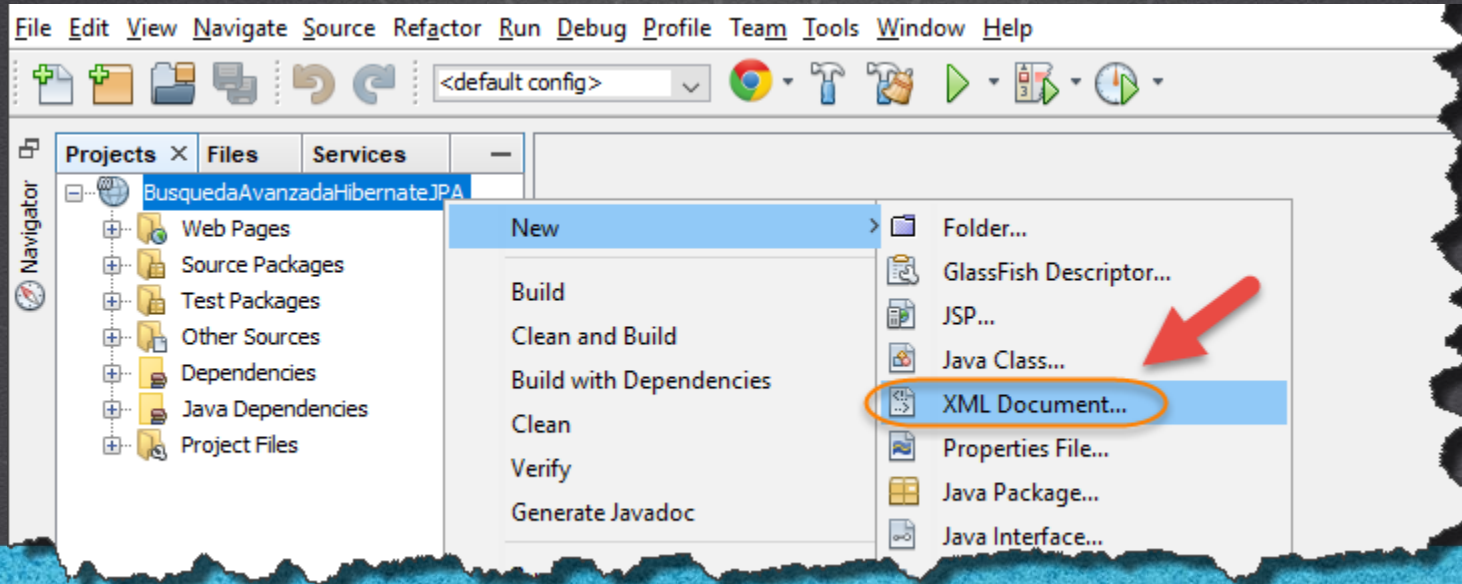
```
<fieldset>
  <legend>
    <br>
    Búsqueda por Curso
  </legend>
  Nombre Curso:
  <input type="text" name="nombreCurso" size="50">
</fieldset>
<br/>
<input type="submit" value="Buscar">
</form>
</body>
</html>
```

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 13. CREAR UN ARCHIVO XML

Creamos un archivo xml de glassfish-web.xml. Este archivo evita que glassfish use sus librerías, y utilice exclusivamente las librerías que hemos agregado a nuestro proyecto, por ejemplo log4j2:



CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 13. CREAR UN ARCHIVO XML

Creamos un archivo xml de glassfish-web.xml:

New XML Document

Steps

1. Choose File Type
- 2. Name and Location**
3. Select Document Type
4. ...

Name and Location

File Name:

Project:

Folder:

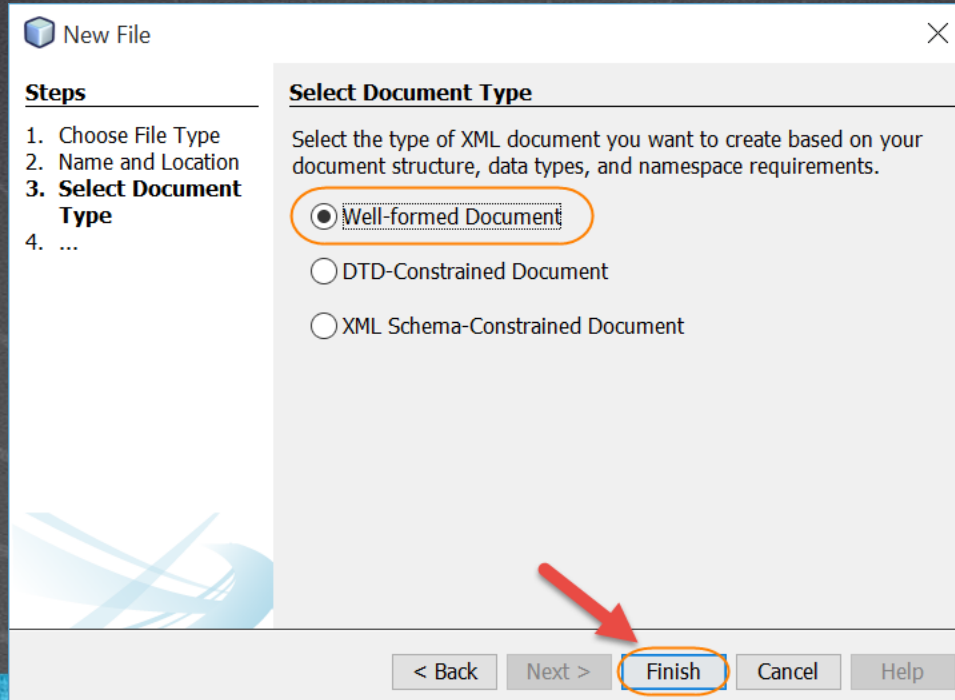
Created File:

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 13. CREAR UN ARCHIVO XML

Creamos un archivo xml de glassfish-web.xml:



PASO 14. MODIFICAMOS EL CÓDIGO

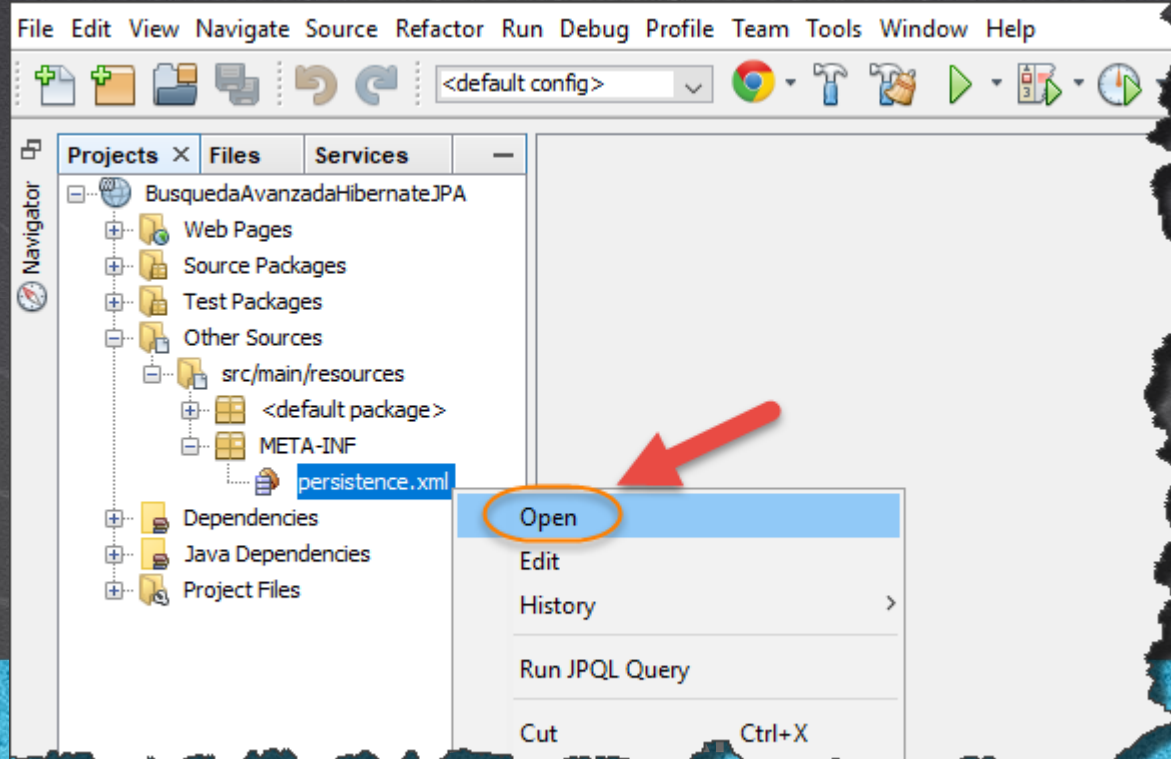
Archivo glassfish-web.xml:

Clic para descargar el código

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Servlet 3.0//EN"
"http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app error-url="">
  <class-loader delegate="false"/>
</glassfish-web-app>
```


PASO 15. MODIFICAR UN ARCHIVO XML

Modificamos el archivo persistence.xml:



PASO 15. MODIFICAMOS EL CÓDIGO

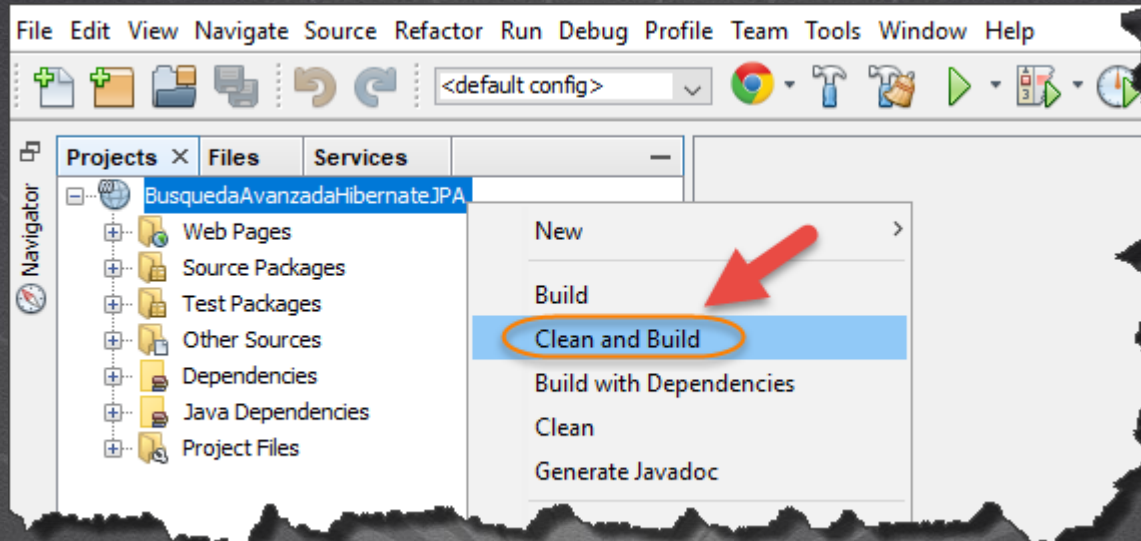
Archivo persistence.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" 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_1.xsd">
  <persistence-unit name="HibernateJpaPU" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>model.Rol</class>
    <class>model.Sucursal</class>
    <class>model.Curso</class>
    <class>model.Alumno</class>
    <class>model.Usuario</class>
    <class>model.Contacto</class>
    <class>model.Pago</class>
    <class>model.Domicilio</class>
    <class>model.Asignacion</class>
    <class>model.Instructor</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/sga_db?useSSL=true"/>
      <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.format_sql" value="true" />
    </properties>
  </persistence-unit>
</persistence>
```

Clic para descargar
el código

PASO 16. HACEMOS UN CLEAN & BUILD

Hacemos Clean & Build para obtener la última versión de cada archivo:

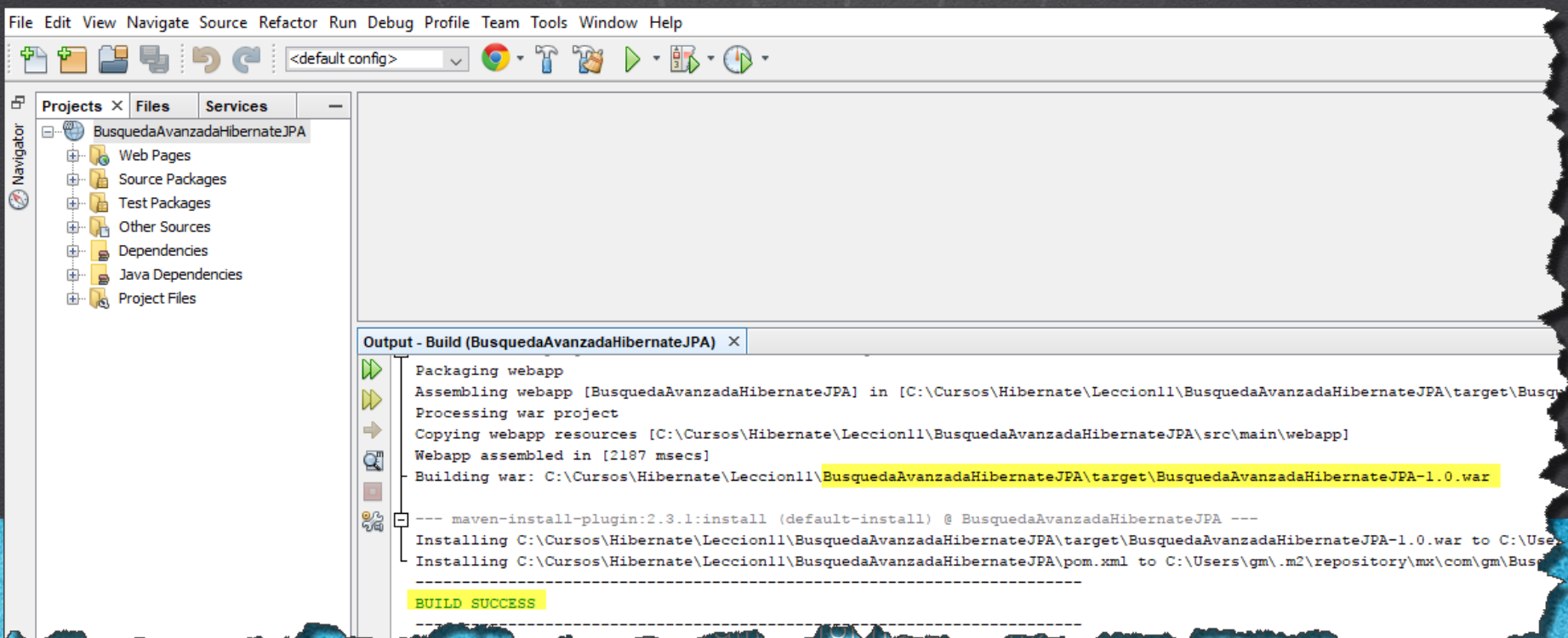


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

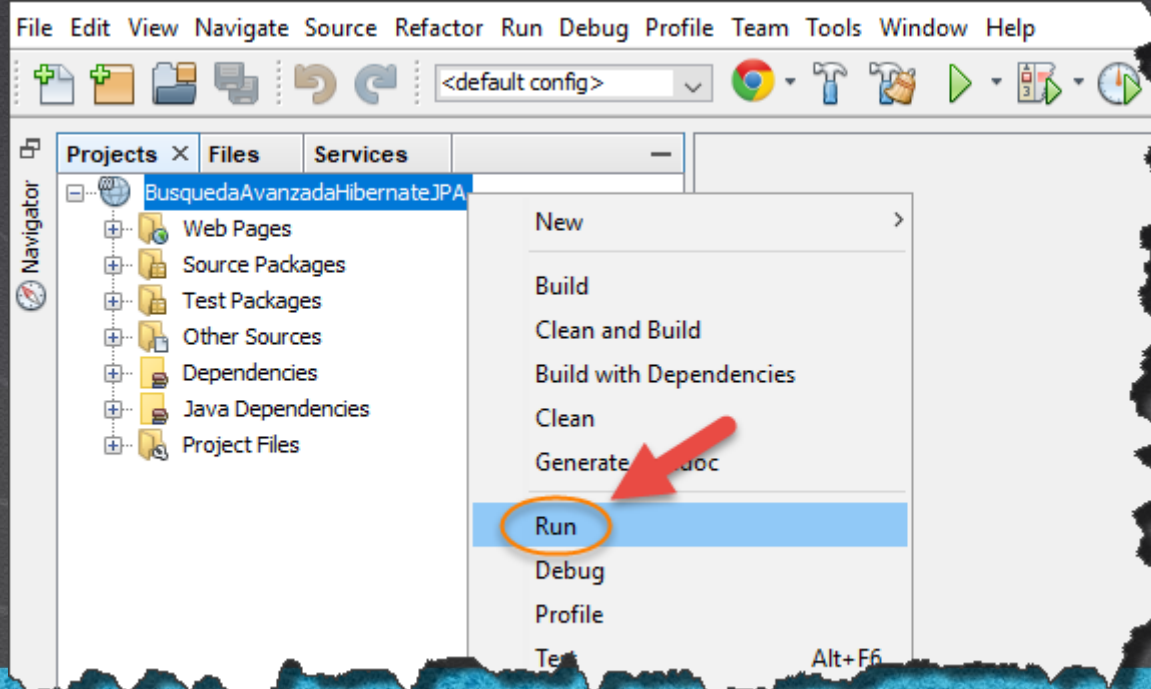
PASO 16. HACEMOS UN CLEAN & BUILD

Hacemos Clean & Build para obtener la última versión de cada archivo:



PASO 17. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:

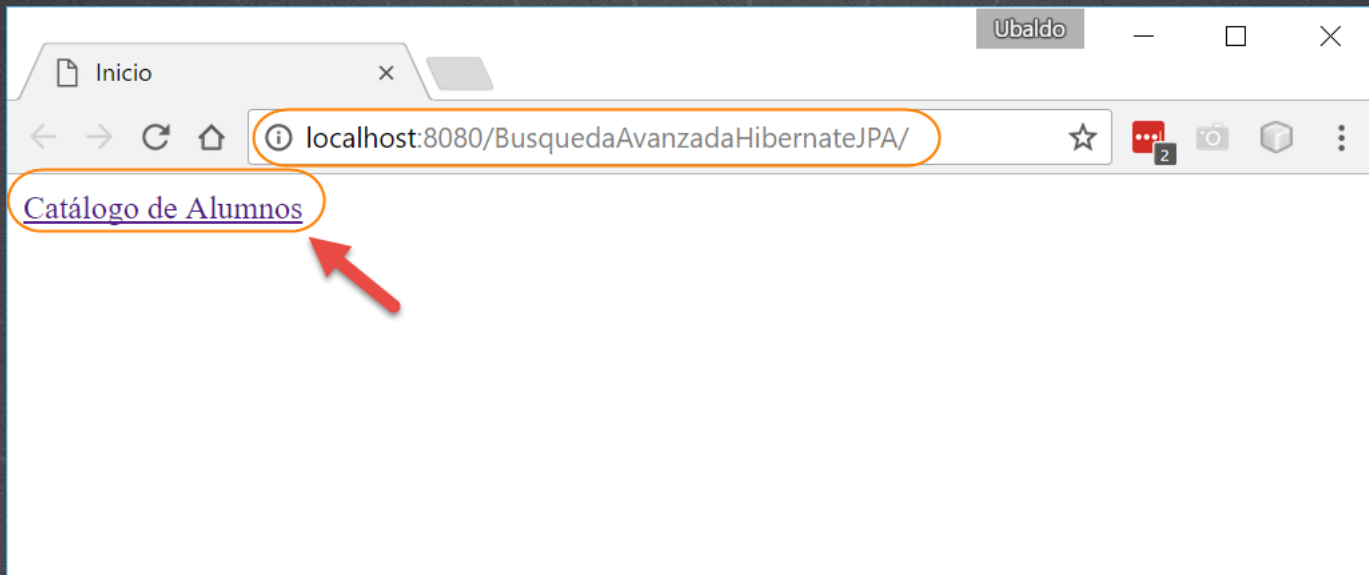


CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx

PASO 17. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:



PASO 17. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto (agregar algunos datos de prueba como se muestran):



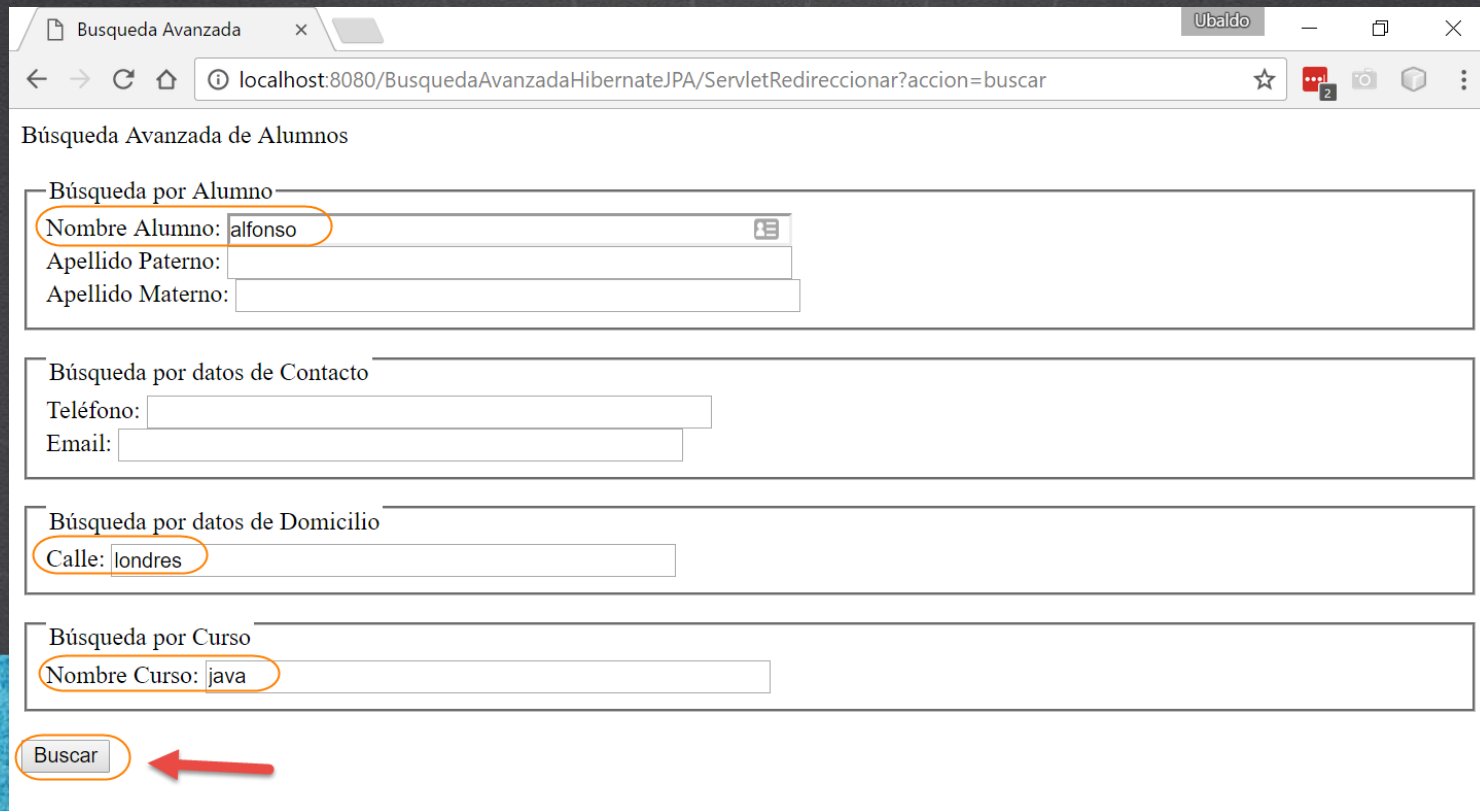
Listar Alumnos

[Agregar](#) [Búsqueda Avanzada](#)

Alumno ID	Nombre	Apellido Paterno	Apellido Materno	Domicilio	Teléfono	Email	Cursos
1	Alfonso	Ugalde	Martinez	Londres	55551111	alfonso@mail.com	<ul style="list-style-type: none">• Curso Fundamentos Java• Curso Programación con Java
2	Martha	Martinez	Garcia	Allende	55717189	contacto@mail.com	
7	Carlos	Mendez	Velez	Jupiter	44441111	carlos@mail.com	
8	Maria	Perez	Lara	Darwin		maria@mail.com	

PASO 17. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:



The screenshot shows a web browser window with the title 'Busqueda Avanzada'. The address bar displays the URL: `localhost:8080/BusquedaAvanzadaHibernateJPA/ServletRedireccionar?accion=buscar`. The page content is titled 'Búsqueda Avanzada de Alumnos' and contains four search sections, each with a title and input fields:

- Búsqueda por Alumno:** Includes input fields for 'Nombre Alumno: alfonso', 'Apellido Paterno:', and 'Apellido Materno:'. The 'Nombre Alumno' field is highlighted with an orange circle.
- Búsqueda por datos de Contacto:** Includes input fields for 'Teléfono:' and 'Email:'.
- Búsqueda por datos de Domicilio:** Includes an input field for 'Calle: londres', which is highlighted with an orange circle.
- Búsqueda por Curso:** Includes an input field for 'Nombre Curso: java', which is highlighted with an orange circle.

At the bottom left, there is a 'Buscar' button, also highlighted with an orange circle. A red arrow points to this button from the right.

PASO 17. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:

Listar Alumnos

[Agregar](#) | [Búsqueda Avanzada](#)

Alumno ID	Nombre	Apellido Paterno	Apellido Materno	Domicilio	Teléfono	Email	Cursos
<u>1</u>	Alfonso	Ugalde	Martinez	Londres	55551111	alfonso@mail.com	<ul style="list-style-type: none">• Curso Fundamentos Java• Curso Programación con Java

CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos ejecutado la consulta avanzada utilizando el framework de Hibernate.
- De momento estamos utilizando el API de Criteria de Hibernate. Posterior a la versión 5.2 de Hibernate se ha depreciado esta API, sin embargo no existe una opción mejor de momento que podamos recomendar.
- Es posible utilizar el API de Criteria de JPA, pero no incluye el API de QBE (Query by Example) por lo que en caso de querer utilizarlo se tendrá que validar campo por campo e ir concatenando cada uno de los criterios que deseemos utilizar.

CURSO ONLINE

HIBERNATE & JPA

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO HIBERNATE Y JPA

www.globalmentoring.com.mx