

Lab 05

Database access and event handling

Mục tiêu

- Thực hành với JPA
- Event handling với Servlet Listener

Phần I Bài tập step by step

Bài 1.1

Mục tiêu:

- Sử dụng JDBC để tương tác với database
- Tạo được ứng dụng web làm việc với database có các chức năng:
 - o Load all
 - o Detail
 - o Insert
 - o Update
 - o Delete

STEP 1: Tạo CSDL cho project

- Chạy file “DB_JspServlet_Lab5.sql” để tạo CSDL cho bài tập này
- Tên database: DBLab5
- Tên table: tblProduct

STEP 2: Tạo Project web đặt tên “JspServlet_Lab5_1”

- Add gói thư viện “sqljdbc4.jar” để tương tác với database
- Tạo class đóng/mở kết nối: Từ project click chuột phải chọn “New / Java Class”, đặt tên class là “DBUtility”, package là “utility”. Class này có nội dung như sau:

```
package utility;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author HAITHANH
 */
public class DBUtility {
    public static Connection openConnection() {
        Connection con = null;
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            con =
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=DBLab5","sa","12
34@");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(DBUtility.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(DBUtility.class.getName()).log(Level.SEVERE, null, ex);
        }
        return con;
    }

    public static void closeAll(Connection con, PreparedStatement pstmt, ResultSet rs) {
        if(con!=null)
            try {
                con.close();
            } catch (SQLException ex) {
                Logger.getLogger(DBUtility.class.getName()).log(Level.SEVERE, null, ex);
            }
        if(pstmt!=null)
            try {
                pstmt.close();
            } catch (SQLException ex) {
                Logger.getLogger(DBUtility.class.getName()).log(Level.SEVERE, null, ex);
            }
        if(rs!=null)
            try {
                rs.close();
            } catch (SQLException ex) {
                Logger.getLogger(DBUtility.class.getName()).log(Level.SEVERE, null, ex);
            }
    }
}
```

STEP 3: Tạo Servlet và class DAO để thực hiện lấy dữ liệu từ database và lưu vào một collection.

- Click chuột phải vào Project chọn “New / Servlet”, đặt tên class là “LoadAllProducts”, package là “controller”. Tiếp theo chọn “Next / Finish”
- Xóa khối lệnh try... trong hàm **processRequest()** đi để viết lệnh xử lý vào đó
- Tạo class Product có các thuộc tính tương ứng với các cột của bảng trong database: Click chuột phải vào project chọn “New / Java Class”, đặt tên lớp là “Product”, package là “entities”.

```
package entities;

/**
 *
 * @author HAITHANH
 */
public class Product {
    private int proId;
    private String proName;
    private String producer;
    private int yearMaking;
    private float price;

    public Product() {
    }

    public Product(int proId, String proName, String producer, int yearMaking, float price) {
        this.proId = proId;
        this.proName = proName;
        this.producer = producer;
        this.yearMaking = yearMaking;
        this.price = price;
    }

    public int getProId() {
        return proId;
    }

    public void setProId(int proId) {
        this.proId = proId;
    }

    public String getProName() {
        return proName;
    }
}
```

```
public void setProName(String proName) {
    this.proName = proName;
}

public String getProducer() {
    return producer;
}

public void setProducer(String producer) {
    this.producer = producer;
}

public int getYearMaking() {
    return yearMaking;
}

public void setYearMaking(int yearMaking) {
    this.yearMaking = yearMaking;
}

public float getPrice() {
    return price;
}

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

- Cài đặt hàm DAO để tương tác với database: Click chuột phải vào Project chọn “New / Java Class”, đặt tên file là “ProductDAO”, package là “dao”. Lớp này có các hàm để tương tác với database.

```
package dao;

import entities.Product;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import utility.DBUtility;

/**
 *
```

```
* @author HAITHANH
*/
public class ProductDAO {
    public List<Product> getAllProducts(){
        List<Product> list = new ArrayList<>();
        Connection con;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        con = DBUtility.openConnection();
        try {
            pstmt = con.prepareStatement("select * from tblProduct");
            rs = pstmt.executeQuery();
            while(rs.next()){
                Product p = new Product();
                p.setProId(rs.getInt("ProId"));
                p.setProName(rs.getString("ProName"));
                p.setProducer(rs.getString("Producer"));
                p.setYearMaking(rs.getInt("YearMaking"));
                p.setPrice(rs.getFloat("Price"));
                list.add(p);
            }
        } catch (SQLException ex) {
            Logger.getLogger(ProductDAO.class.getName()).log(Level.SEVERE, null, ex);
        } finally{
            DBUtility.closeAll(con, pstmt, rs);
        }
        return list;
    }
}
```

- Vào Servlet **"LoadAllProducts"** vừa tạo ở trên, thêm các lệnh sau vào hàm **"processRequest()"**

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    List<Product> allProducts = new ProductDAO().getAllProducts();
    request.setAttribute("listP", allProducts);
    request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

STEP 4: Tạo trang **"index.jsp"** và trình bày nội dung cần hiển thị

- Xóa trang **"index.xhtml"** đi và click chuột phải vào **"Web Pages"** chọn **"New / JSP..."**, đặt tên file là **"index"**.
- Thêm nội dung sau vào giữa thẻ **<body>**

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <center>
      <h1>ALL PRODUCTS</h1>
      <table border="1" cellspacing="10px">
        <tr>
          <th>Product Id</th>
          <th>Product Name</th>
          <th>Producer</th>
          <th>Year Making</th>
          <th>Price</th>
        </tr>
        <c:forEach items="${listP}" var="p">
          <tr>
            <td>${p.proId}</td>
            <td>${p.proName}</td>
            <td>${p.producer}</td>
            <td>${p.yearMaking}</td>
            <td><fmt:formatNumber currencySymbol=""
value="${p.price}"></fmt:formatNumber> đ</td>
          </tr>
        </c:forEach>
      </table>
    </center>
  </body>
</html>
```

STEP 5: Cài đặt để Servlet chạy trước và chạy thử ứng dụng

- Để lấy được dữ liệu và đẩy lên trang "index.jsp" thì Servlet cần phải được chạy trước
- Click chuột phải vào project chọn properties. Chọn mục "Run", trong mục "Relative URL" điền giá trị sau vào: **"/LoadAllProducts"** (đó chính là đường dẫn tương đối để gọi Servlet).
- Deploy project và chạy chương trình chúng ta sẽ được giao diện ở trang đầu tiên

Ta được giao diện trang "index.jsp" như sau:



ALL PRODUCTS

Product Id	Product Name	Producer	Year Making	Price
1	Xe máy	Honda	2015	40,000,000 đ
2	Ô tô	Audi	2016	1,200,000,000 đ
3	Máy tính	Sony	2017	14,000,000 đ
4	Tivi	Toshiba	2015	5,000,000 đ
5	Tủ lạnh	Daewoo	2015	8,000,000 đ
6	Điều hòa	Funiki	2016	10,000,000 đ



Ở đây chúng ta sử dụng thẻ

```
<fmt:formatNumber currencySymbol="" value="{p.price}"></fmt:formatNumber>
```

Để định dạng hiển thị dưới dạng tiền tệ, tuy nhiên chúng ta chưa sử dụng Locale để định nghĩa mã vùng hiển thị nên tạm thời thay thế kí tự tiền tệ của thẻ bằng rỗng và thêm vào kí tự 'đ' ở cuối để hiển thị là tiền tệ Việt Nam. Thẻ này cho phép chúng ta định dạng hiển thị các kiểu dữ liệu như: Số, tiền tệ, phần trăm.

Bài 1.2 Tạo chức năng chi tiết sản phẩm

STEP 1: Tạo liên kết để thực hiện chức năng detail

- Từ trang "index.jsp" thêm thẻ <a> để tạo một liên kết tới Servlet thực hiện detail

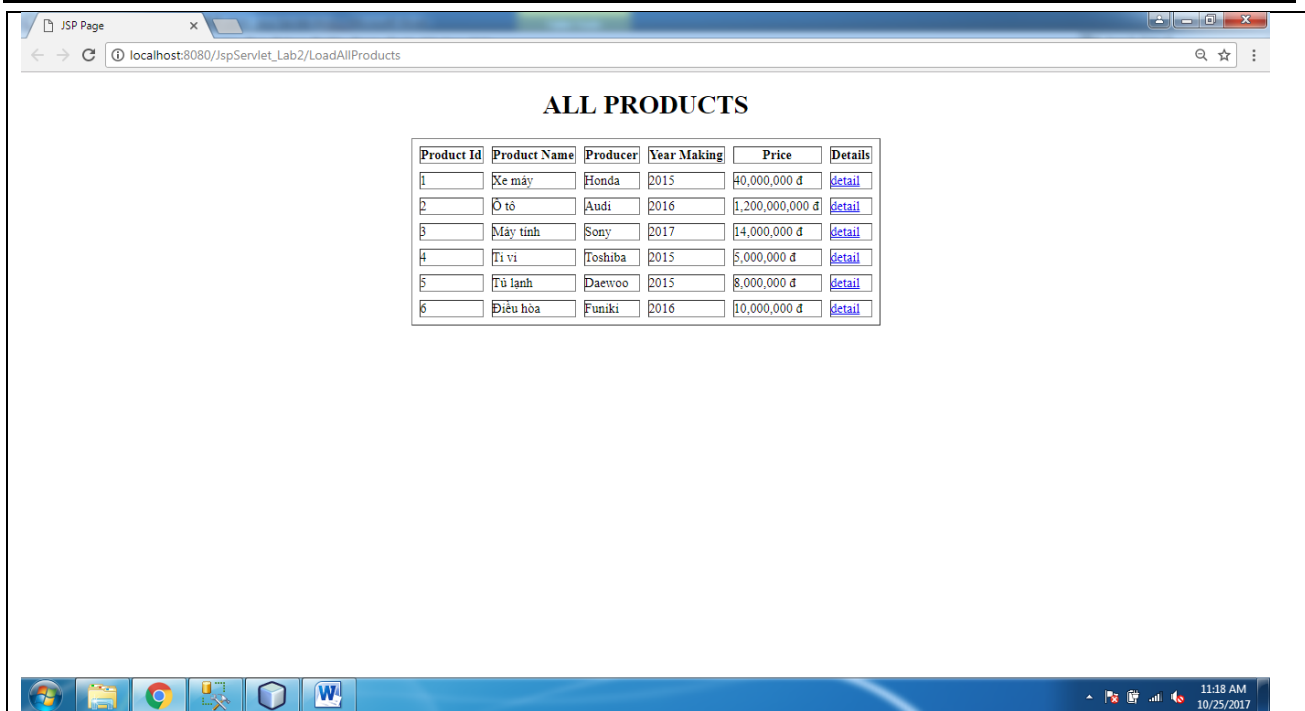
Trong thẻ tạo tiêu đề thêm thẻ tiêu đề sau:

```
<th>Details</th>
```

Trong thẻ <forEach> thêm thẻ sau:

```
<td><a href="DetailProduct?id={p.productId}">detail</a></td>
```

Khi chạy lên chúng ta sẽ có giao diện với liên kết như sau:



Product Id	Product Name	Producer	Year Making	Price	Details
1	Xe máy	Honda	2015	40,000,000 đ	Detail
2	Ô tô	Audi	2016	1,200,000,000 đ	Detail
3	Máy tính	Sony	2017	14,000,000 đ	Detail
4	Tivi	Toshiba	2015	5,000,000 đ	Detail
5	Tủ lạnh	Daewoo	2015	8,000,000 đ	Detail
6	Điều hòa	Funiki	2016	10,000,000 đ	Detail

STEP 2: Tạo Servlet và lấy chi tiết sản phẩm từ database

- Tạo một Servlet có tên **"DetailProduct"** trong cùng thư mục với servlet **"LoadAllProducts"**, và xóa khối lệnh try{...} trong hàm **"processRequest"** để thêm lệnh xử lý vào.
- Vào trong class **"ProductDAO"** thêm một hàm để lấy về **Product** theo id của nó

```
public Product getProductById(int proId){
    Product p = null;
    Connection con;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    con = DBUtility.openConnection();
    try {
        pstmt = con.prepareStatement("select * from tblProduct where ProId=?");
        pstmt.setInt(1, proId);
        rs = pstmt.executeQuery();
        if(rs.next()){
            p = new Product();
            p.setProId(rs.getInt("ProId"));
            p.setProName(rs.getString("ProName"));
            p.setProducer(rs.getString("Producer"));
            p.setYearMaking(rs.getInt("YearMaking"));
            p.setPrice(rs.getFloat("Price"));
        }
    } catch (SQLException ex) {
        Logger.getLogger(ProductDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```



```
}finally{  
    DBUtility.closeAll(con, pstmt, rs);  
}  
return p;  
}
```

- Tiếp theo vào trong hàm “**processRequest**” của servlet “**DetailProduct**” thêm lệnh xử lý sau:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException {  
    response.setContentType("text/html; charset=UTF-8");  
    int proId = Integer.parseInt(request.getParameter("id"));  
    Product productById = new ProductDAO().getProductById(proId);  
    request.setAttribute("pro", productById);  
    request.getRequestDispatcher("detailProduct.jsp").forward(request, response);  
}
```

STEP 3: Tạo trang detailProduct.jsp và trình bày các nội dung chi tiết của sản phẩm.

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<%@ page contentType="text/html" pageEncoding="UTF-8"%>  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>JSP Page</title>  
    </head>  
    <body>  
        <center>  
            <h1>DETAIL PRODUCTS</h1>  
            <table border="0" cellspacing="10px">  
                <tr>  
                    <td><b>Product Id:</b></td>  
                    <td>${pro.proId}</td>  
                </tr>  
                <tr>  
                    <td><b>Product Name:</b></td>  
                    <td>${pro.proName}</td>  
                </tr>  
                <tr>  
                    <td><b>Producer:</b></td>  
                    <td>${pro.producer}</td>  
                </tr>  
                <tr>  
                    <td><b>Year Making:</b></td>  
                    <td>${pro.yearMaking}</td>  
                </tr>  
            </table>  
        </center>  
    </body>  
</html>
```

```

</tr>
<tr>
<td><b>Price:</b></td>
<td><fmt:formatNumber currencySymbol=""
value="\${pro.price}"></fmt:formatNumber> đ</td>
</tr>
</table>
<a href="LoadAllProducts">Back</a>
</center>
</body>
</html>

```

Chúng ta được giao diện như sau:



DETAIL PRODUCTS

Product Id: 1
Product Name: Xe máy
Producer: Honda
Year Making: 2015
Price: 40,000,000 đ
[Back](#)



STEP 4: Tạo liên kết để quay về trang index.jsp

- Thêm thẻ <a> vào trang detailProduct.jsp để quay về trang index.jsp như sau:

```
<a href="LoadAllProducts">Back</a>
```

- Thẻ này được thêm vào sau thẻ <table> khi hiển thị chi tiết các thông tin của sản phẩm.

Bài 1.3 Tạo chức năng thêm mới sản phẩm

STEP 1: Tạo liên kết để thực hiện chức năng thêm mới

- Từ trang "index.jsp" sau thẻ <table> thêm thẻ <a> vào sau thẻ <table> như sau để vào trang thêm mới

[Add New Product](addProduct.jsp)

ALL PRODUCTS

Product Id	Product Name	Producer	Year Making	Price	Details
1	Xe máy	Honda	2015	40,000,000 đ	detail
2	Ô tô	Audi	2016	1,200,000,000 đ	detail
3	Máy tính	Sony	2017	14,000,000 đ	detail
4	Ti vi	Toshiba	2015	5,000,000 đ	detail
5	Tủ lạnh	Daewoo	2015	8,000,000 đ	detail
6	Điều hòa	Funiki	2016	10,000,000 đ	detail

[Add New Product](#)

STEP 2: Tạo trang addProduct.jsp và tạo form để thêm mới một Vegetable

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <center>
      <h1>ADD NEW PRODUCT</h1>
      <form action="AddProduct">
        <table border="1">
          <tr>
            <td>Product Name: </td>
            <td>
              <input type="text" name="proName" required/>
            </td>
          </tr>
          <tr>
            <td>Producer: </td>
            <td>
              <input type="text" name="producer" required/>
            </td>
          </tr>
          <tr>
            <td>Year Making: </td>
            <td>
              <input type="text" name="yearMaking" required/>
            </td>
          </tr>
        </table>
      </form>
    </center>
  </body>
</html>
```

```

</tr>
<tr>
  <td>Price: </td>
  <td>
    <input type="text" name="price" required/>
  </td>
</tr>
<tr>
  <td></td>
  <td>
    <input type="submit" value="AddNew"/>
    <input type="reset" value="Clear"/>
  </td>
</tr>
</table>
</form>
</center>
</body>
</html>

```

Giao diện trang thêm mới:

ADD NEW PRODUCT

Product Name:	<input type="text"/>
Producer:	<input type="text"/>
Year Making:	<input type="text"/>
Price:	<input type="text"/>
	<input type="button" value="AddNew"/> <input type="button" value="Clear"/>

STEP 3: Tạo Servlet **AddProduct** và lấy dữ liệu từ form xuống

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //Thiết lập kiểu kí tự để nhận được tiếng việt có dấu
    response.setContentType("text/html;charset=UTF-8");
    request.setCharacterEncoding("UTF-8");

    //Lấy dữ liệu từ form submit lên
    String proName = request.getParameter("proName");
    String producer = request.getParameter("producer");

```

```
int yearMaking = Integer.parseInt(request.getParameter("yearMaking"));
float price = Float.parseFloat(request.getParameter("Price"));

//Gọi hàm xử lý trong DAO
Product p = new Product();
p.setProName(proName);
p.setProducer(producer);
p.setYearMaking(yearMaking);
p.setPrice(price);

boolean blAdd = new ProductDAO().addProduct(p);
if(blAdd){
    //chuyển về trang index.jsp
    request.getRequestDispatcher("LoadAllProducts").forward(request, response);
}else{
    //Có lỗi, gán biến lỗi và quay về trang addProduct.jsp
    request.setAttribute("status", "Add product failed!");
    request.getRequestDispatcher("addProduct.jsp").forward(request, response);
}
}
```

- Thêm thẻ vào trang “addProduct.jsp” để hiển thị lỗi khi add new

```
<h3 style="color: red">${status}</h3>
```

STEP 4: Viết hàm trong DAO để thực hiện insert dữ liệu vào database

```
public boolean addProduct(Product pro){
    boolean bl = false;
    Connection con;
    PreparedStatement pstmt = null;

    con = DBUtility.openConnection();
    try {
        pstmt = con.prepareStatement("insert into tblProduct values (?, ?, ?, ?)");
        pstmt.setString(1, pro.getProName());
        pstmt.setString(2, pro.getProducer());
        pstmt.setInt(3, pro.getYearMaking());
        pstmt.setFloat(4, pro.getPrice());
        int i = pstmt.executeUpdate();
        if(i>0){
            bl = true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(ProductDAO.class.getName()).log(Level.SEVERE, null, ex);
    } finally{
        DBUtility.closeAll(con, pstmt, null);
    }
    return bl;
}
```

Bài 1.4 Tạo chức năng cập nhật sản phẩm

STEP 1: Tạo liên kết để thực hiện chức năng update

- Trong trang “detailProduct.jsp” thêm thẻ <a> gọi tới một Servlet để thực hiện hành động update

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <center>
      <h1>DETAIL PRODUCTS</h1>
      <table border="0" cellspacing="10px">
        <tr>
          <td><b>Product Id:</b></td>
          <td>${pro.proId}</td>
        </tr>
        <tr>
          <td><b>Product Name:</b></td>
          <td>${pro.proName}</td>
        </tr>
        <tr>
          <td><b>Producer:</b></td>
          <td>${pro.producer}</td>
        </tr>
        <tr>
          <td><b>Year Making:</b></td>
          <td>${pro.yearMaking}</td>
        </tr>
        <tr>
          <td><b>Price:</b></td>
          <td><fmt:formatNumber currencySymbol=""
value="${pro.price}"></fmt:formatNumber> đ</td>
        </tr>
        <tr>
          <td>
            <a href="PreUpdateProduct?proId=${pro.proId}">Update</a>
          </td>
        </tr>
      </table>
    </center>
  </body>
</html>
```

```
<a href="LoadAllProducts">Back</a>
</td>
</tr>
</table>
</center>
</body>
</html>
```

Giao diện sẽ như sau:

DETAIL PRODUCTS

Product Id: 1
Product Name: Xe máy
Producer: Honda
Year Making: 2015
Price: 40,000,000 đ
[Update](#) [Back](#)

STEP 2: Tạo Servlet và kết nối với database để lấy về chi tiết sản phẩm

- Tạo một Servlet có tên “**PreUpdateProduct**” trong package controller
- Thêm lệnh để xử lý trong hàm “**processRequest()**”:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    int proId = Integer.parseInt(request.getParameter("proId"));
    Product product = new ProductDAO().getProductById(proId);
    request.setAttribute("pro", product);
    request.getRequestDispatcher("updateProduct.jsp").forward(request, response);
}
```

STEP 3: Tạo trang “updateProduct.jsp” và đẩy dữ liệu từ servlet lên để cập nhật

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
  <center>
    <h1>UPDATE PRODUCT</h1>
    <h3 style="color: red">${status}</h3>
    <form action="UpdateProduct">
      <table border="1">
        <tr>
          <td>Product Id:</td>
          <td>
            <input type="text" name="proId" value="${pro.proId}" disabled="true"/>
            <input type="hidden" name="proId" value="${pro.proId}"/>
          </td>
        </tr>
        <tr>
          <td>Product Name: </td>
          <td>
            <input type="text" name="proName" value="${pro.proName}" required/>
          </td>
        </tr>
        <tr>
          <td>Producer: </td>
          <td>
            <input type="text" name="producer" value="${pro.producer}" required/>
          </td>
        </tr>
        <tr>
          <td>Year Making: </td>
          <td>
            <input type="text" name="yearMaking" value="${pro.yearMaking}" required/>
          </td>
        </tr>
        <tr>
          <td>Price: </td>
          <td>
            <input type="text" name="price" value="${pro.price}" required/>
          </td>
        </tr>
        <tr>
          <td></td>
          <td>
            <input type="submit" value="Update"/>
            <input type="reset" value="Clear"/>
          </td>
        </tr>
      </table>
    </form>
  </center>
</body>
```


</html>

Giao diện trang cập nhật:

UPDATE PRODUCT

Product Id:	1
Product Name:	Xe máy
Producer:	Honda
Year Making:	2015
Price:	4.0E7
	<input type="button" value="Update"/> <input type="button" value="Clear"/>

STEP 4: Tạo Servlet “UpdateProduct” và thực hiện lấy dữ liệu từ form đẩy lên (giống như phần insert):

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    request.setCharacterEncoding("UTF-8");

    //Lấy dữ liệu từ form submit lên
    int proId = Integer.parseInt(request.getParameter("proId"));
    String proName = request.getParameter("proName");
    String producer = request.getParameter("producer");
    int yearMaking = Integer.parseInt(request.getParameter("yearMaking"));
    float price = Float.parseFloat(request.getParameter("price"));

    //Gọi hàm xử lý trong DAO
    Product p = new Product();
    p.setProId(proId);
    p.setProName(proName);
    p.setProducer(producer);
    p.setYearMaking(yearMaking);
    p.setPrice(price);

    boolean blUpdate = new ProductDAO().updateProduct(p);
    if(blUpdate){
        //chuyển về trang index.jsp
        request.getRequestDispatcher("LoadAllProducts").forward(request, response);
    }else{
```

```
//Có lỗi, gán biến lỗi và quay về trang updateProduct.jsp
request.setAttribute("status", "Update product failed!");
request.getRequestDispatcher("updateProduct.jsp").forward(request, response);
}
}
```

STEP 5: Viết thêm hàm update trong class ProductDAO

```
public boolean updateProduct(Product pro){
    boolean bl = false;
    Connection con;
    PreparedStatement pstmt = null;

    con = DBUtility.openConnection();
    try {
        pstmt = con.prepareStatement("update tblProduct set ProName=?, Producer=?,
        YearMaking=?, Price=? where ProId=?");
        pstmt.setString(1, pro.getProName());
        pstmt.setString(2, pro.getProducer());
        pstmt.setInt(3, pro.getYearMaking());
        pstmt.setFloat(4, pro.getPrice());
        pstmt.setFloat(5, pro.getProId());
        int i = pstmt.executeUpdate();
        if(i>0){
            bl = true;
        }
    } catch (SQLException ex) {
        Logger.getLogger(ProductDAO.class.getName()).log(Level.SEVERE, null, ex);
    } finally{
        DBUtility.closeAll(con, pstmt, null);
    }
    return bl;
}
```

Bài 1.5 Tạo chức năng xóa sản phẩm

STEP 1: Tạo liên kết để thực hiện hành động xóa

- Từ trang "index.jsp" thêm thẻ <a> vào để tạo liên kết xóa sản phẩm

Trong thẻ tạo tiêu đề thêm thẻ tiêu đề sau:

```
<th>Details</th>
```

Trong thẻ <forEach> thêm thẻ sau:

```
<td><a href="DetailProduct?id=${p.proId}">detail</a></td>
```

Giao diện khi chạy lên:

ALL PRODUCTS

Product Id	Product Name	Producer	Year Making	Price	Details	Deletes
1	Xe máy SH	Honda	2015	40,000,000 đ	detail	delete
2	Ô tô	Audi	2016	1,200,000,000 đ	detail	delete
3	Máy tính	Sony	2017	14,000,000 đ	detail	delete
4	Ti vi	Toshiba	2015	5,000,000 đ	detail	delete
5	Tủ lạnh	Daewoo	2015	8,000,000 đ	detail	delete
6	Điều hòa	Funiki	2016	10,000,000 đ	detail	delete
7	Thêm mới Sản phẩm	Nhà sản xuất mới	2017	40,000 đ	detail	delete

[Add New Product](#)

STEP 2: Tạo Servlet “DeleteProduct” và thêm lệnh vào để xử lý

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    //lấy về product id truyền từ liên kết đến
    int proId = Integer.parseInt(request.getParameter("id"));
    //Gọi hàm trong ProductDAO để thực hiện xóa
    boolean blDelete = new ProductDAO().deleteProduct(proId);
    if(blDelete){
        request.setAttribute("status", "Delete successfully for product with id: "+proId);
    }else{
        request.setAttribute("status", "Delete failed for product with id: "+proId);
    }
    //Load lại dữ liệu và quay về trang index.jsp
    List<Product> allProducts = new ProductDAO().getAllProducts();
    request.setAttribute("listP", allProducts);
    request.getRequestDispatcher("index.jsp").forward(request, response);
}
```

STEP 3: Viết thêm hàm delete trong class ProductDAO

```
public boolean deleteProduct(int proId){
    boolean bl = false;
    Connection con;
    PreparedStatement pstmt = null;

    con = DBUtility.openConnection();
    try {
        pstmt = con.prepareStatement("delete from tblProduct where ProId=?");
        pstmt.setInt(1, proId);
        int i = pstmt.executeUpdate();
        if(i>0){

```

```

        bl = true;
    }
} catch (SQLException ex) {
    Logger.getLogger(ProductDAO.class.getName()).log(Level.SEVERE, null, ex);
} finally {
    DBUtility.closeAll(con, pstmt, null);
}
return bl;
}

```

ALL PRODUCTS

Delete successfully for product with id: 7

Product Id	Product Name	Producer	Year Making	Price	Details	Deletes
1	Xe máy SH	Honda	2015	40,000,000 đ	detail	delete
2	Ô tô	Audi	2016	1,200,000,000 đ	detail	delete
3	Máy tính	Sony	2017	14,000,000 đ	detail	delete
4	Ti vi	Toshiba	2015	5,000,000 đ	detail	delete
5	Tủ lạnh	Daewoo	2015	8,000,000 đ	detail	delete
6	Điều hòa	Funiki	2016	10,000,000 đ	detail	delete

[Add New Product](#)

Bài 2

Mục tiêu:

- Tạo các JPA POJO Object
- Sử dụng DAO Pattern

Sử dụng các câu lệnh sql sau để tạo database cho bài tập này

```

create database JspServlet_Lab5
go
use JspServlet_Lab5
go
create table tblStudent(RollNumber varchar(20) not null primary key,
FullName nvarchar(50), Gender bit, Birthday date, Address nvarchar(200),
ClassName varchar(30))

insert into TblStudent values ('B1234',N'Nguyễn Hải Anh',1,'1995-11-21',N'Ba Vì
- Hà Nội','C1605L')
insert into TblStudent values ('B1233',N'Trần Tiên Nam',1,'1994-06-28',N'Ý Yên

```

```
- Nam Định', 'C1605L')
insert into TblStudent values ('B1244', N'Nguyễn Diệu Huyền', 0, '1995-02-
11', N'Sóc Sơn - Hà Nội', 'C1605L')

select * from TblStudent
```

STEP 1: Tạo project web và add các gói thư viện tương tác database

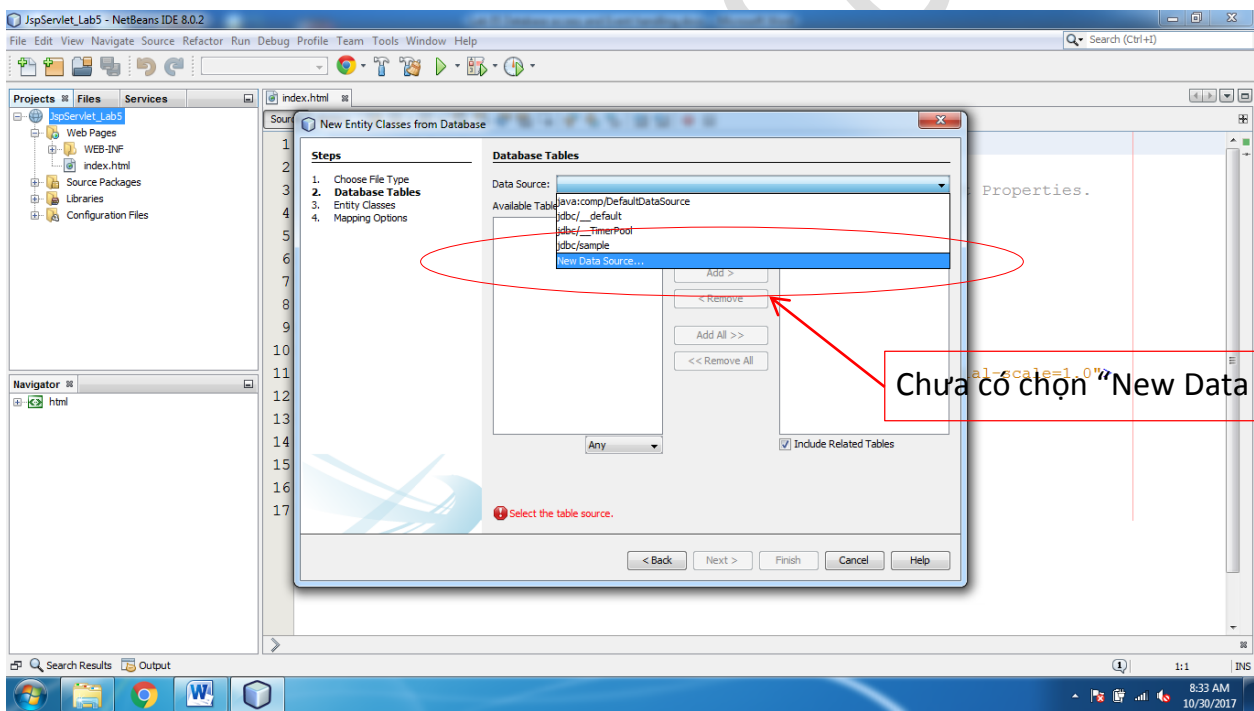
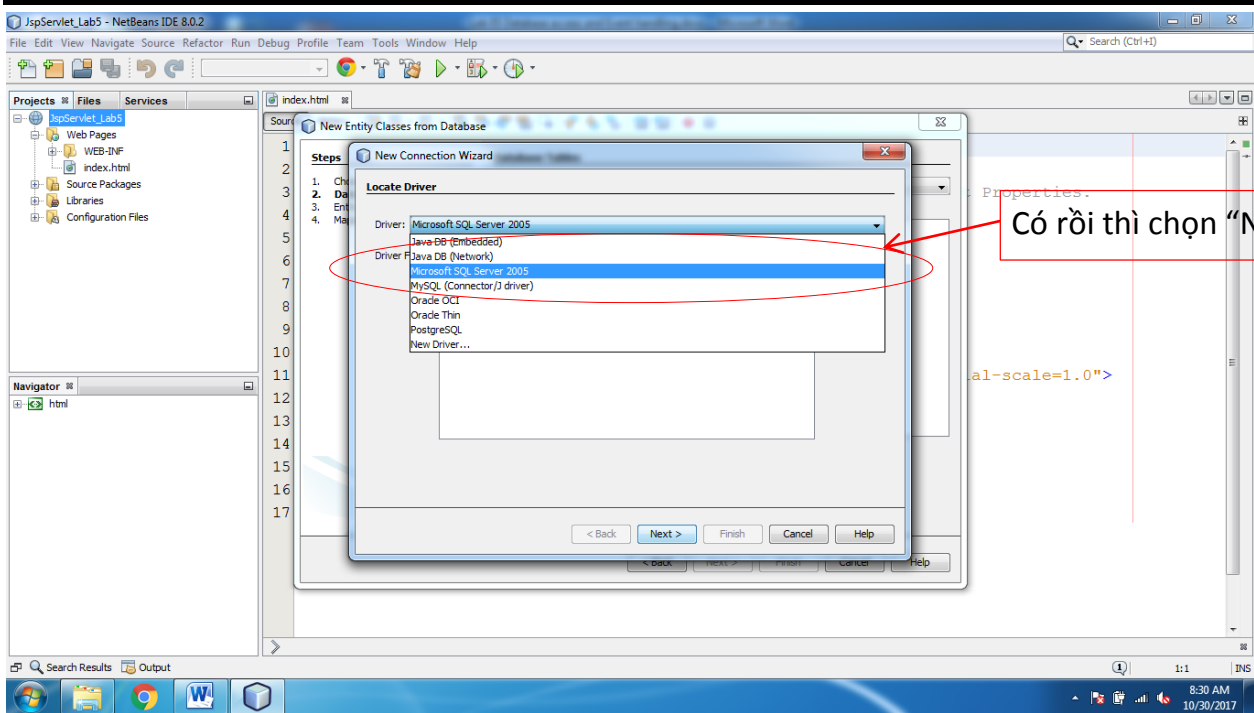
- Tạo project web, đặt tên “JspServlet_Lab5”.
- Add gói thư viện “sqljdbc4.jar” vào project.

STEP 2: Tạo các class POJO cho các bảng của CSDL tblStudent bằng Persistence.

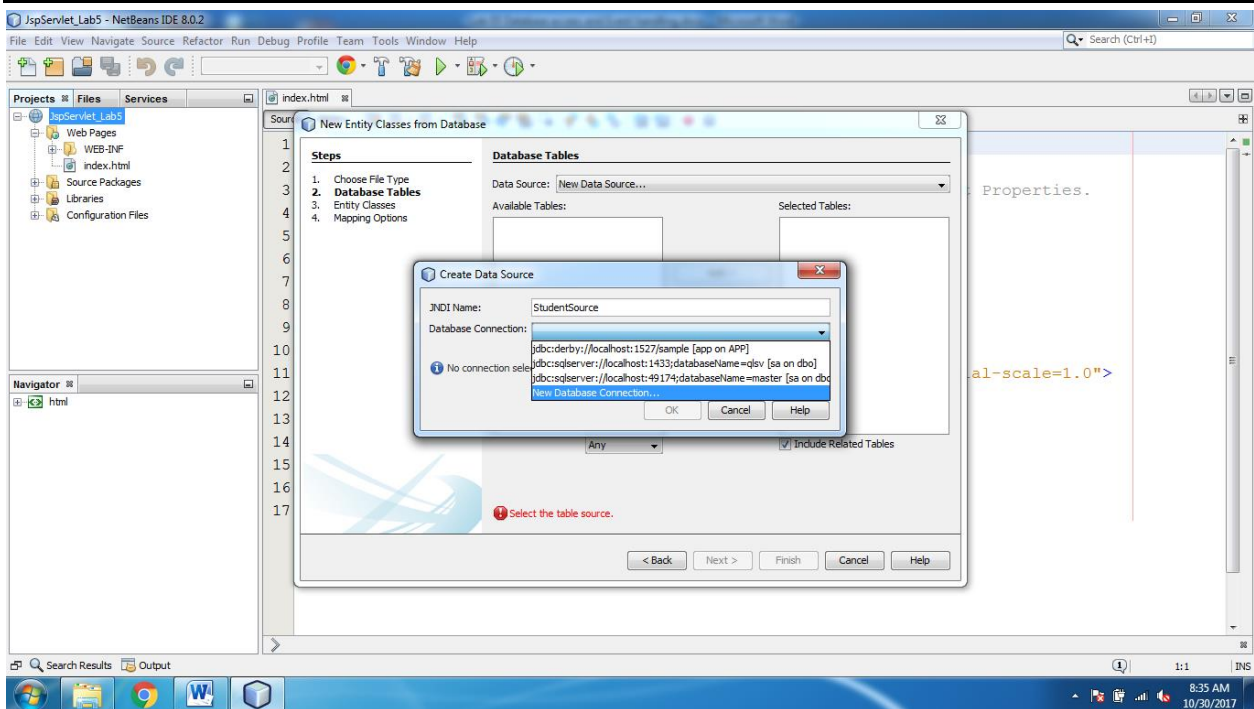
New -> Other -> Persistence -> Entity Classes from Database

Phần Data Source:

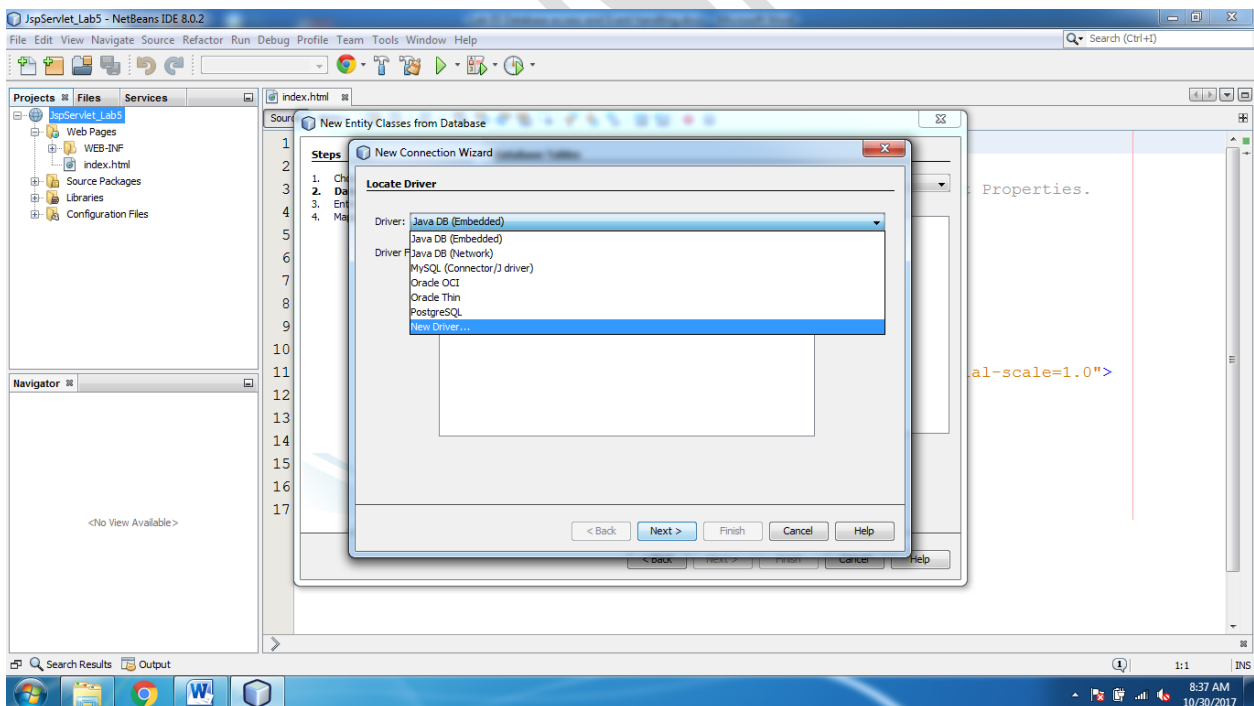
- Chọn “New Data Source..”
- Đặt tên cho JNDI Name: “StudentSource”
- Database Connection: Chọn “New Database Connection..”
- Phần Driver: Click drop down list, nếu có “Microsoft SQL Server1 2005” thì chọn “Next. Nếu chưa có thì chọn nút “Add”, tìm đến thư viện “sqljdbc4.jar” và chọn “Open”.



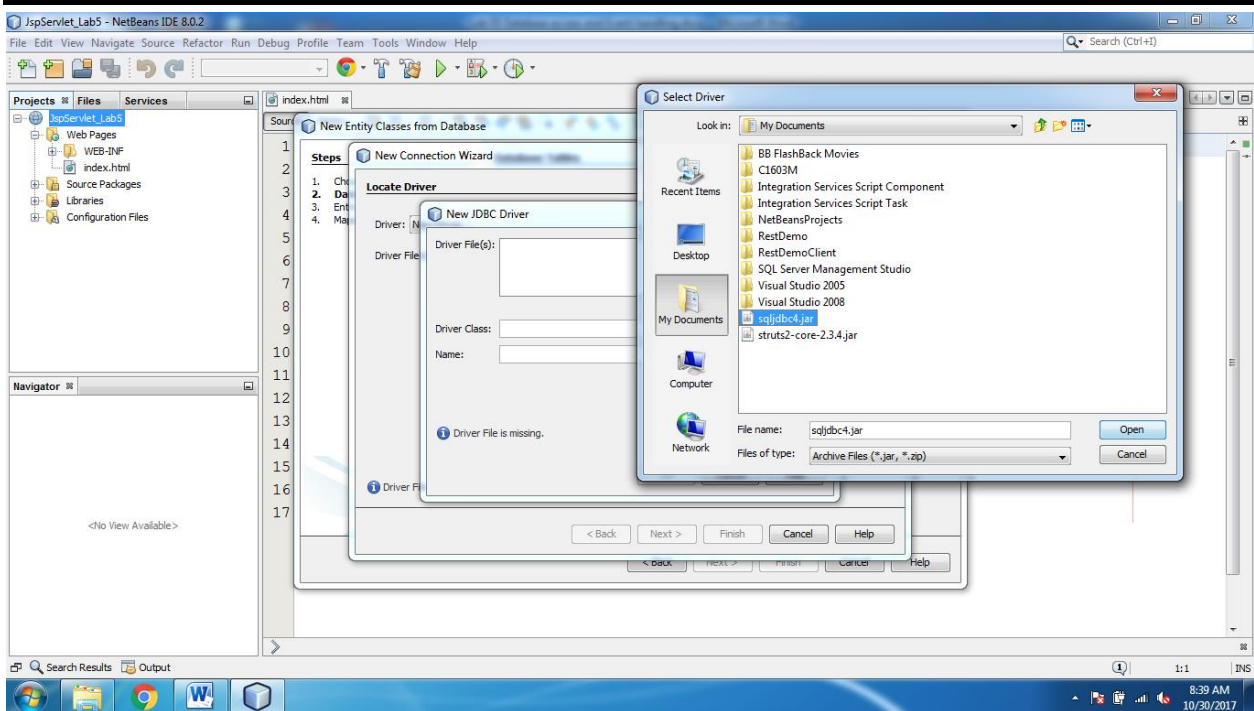
Đặt tên cho JNDI Name và chọn Database Connection:



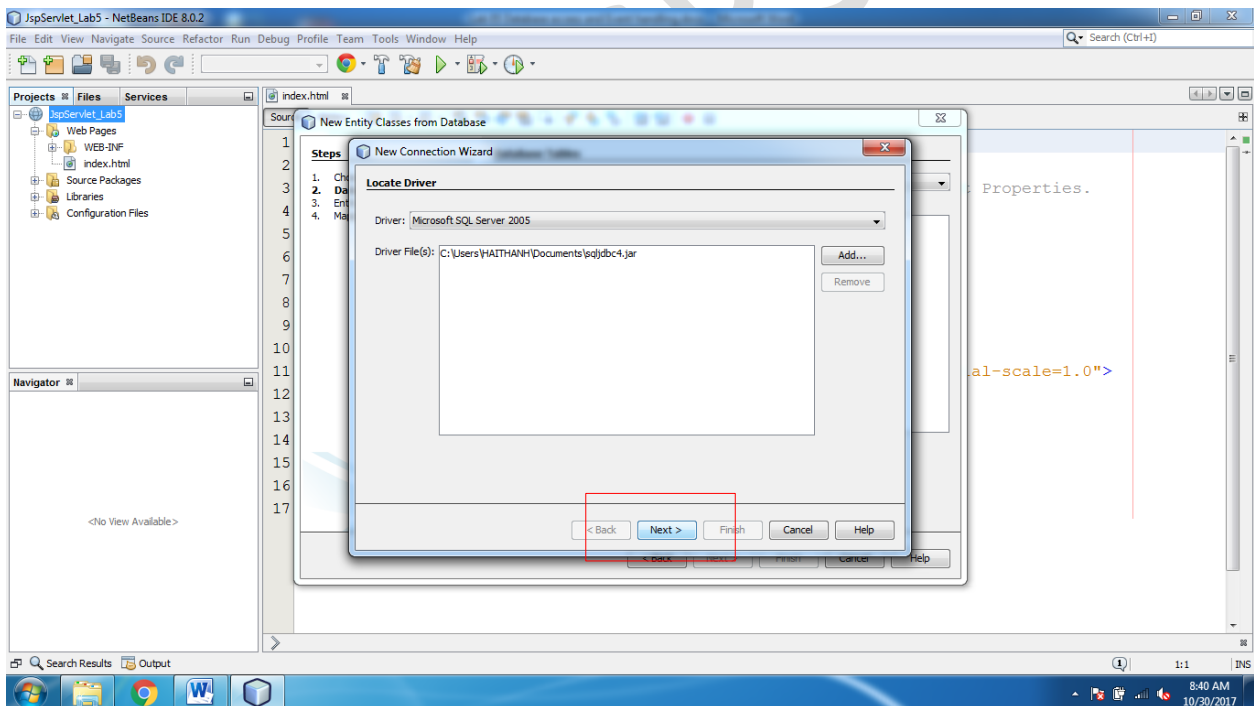
Chưa có “Microsoft SQL Server 2005” chọn “New Driver” (có rồi thì chọn luôn không phải chọn New Driver nữa)



Chọn “Add” rồi tìm đến file “sqljdbc4.jar” + “Ok”.



Chọn “Next” để tiếp tục:



Trong cửa sổ “New Connection Wizard” điền các thông số để kết nối vào database:

New Connection Wizard

Customize Connection

Driver Name: Microsoft SQL Server 2005

Host: localhost Port: 1433

Database: JspServlet_Lab5

Instance Name:

User Name: sa

Password: •••••

☒ Remember password

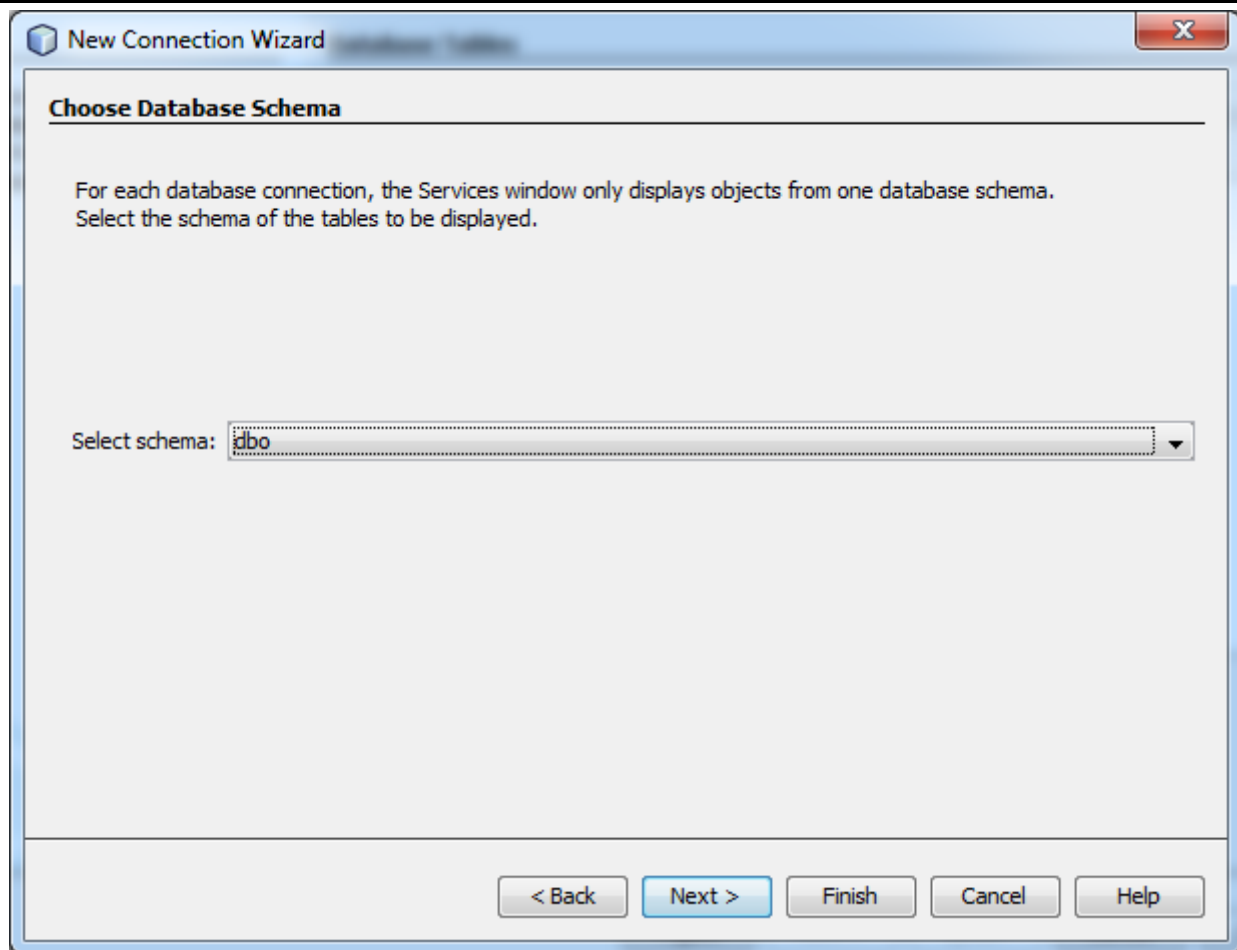
Connection Properties Test Connection

JDBC URL: jdbc:sqlserver://localhost:1433;databaseName=JspServlet_Lab5

< Back Next > Finish Cancel Help

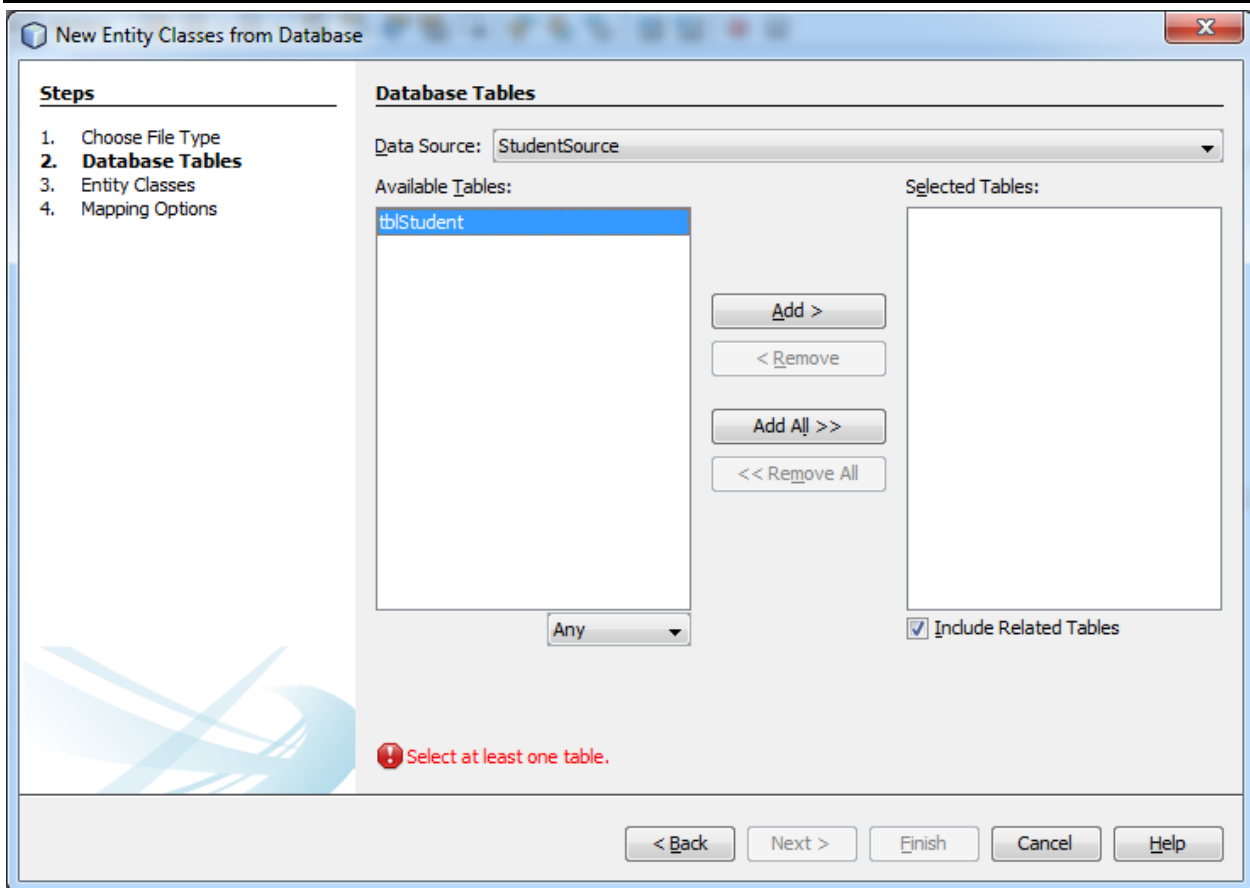
- Chọn “Test Connection” để đảm bảo kết nối thành công
- Chọn “Next” để đi đến bước tiếp sau

Cửa sổ tiếp theo, trong mục “Select Schema” chọn “dbo”:

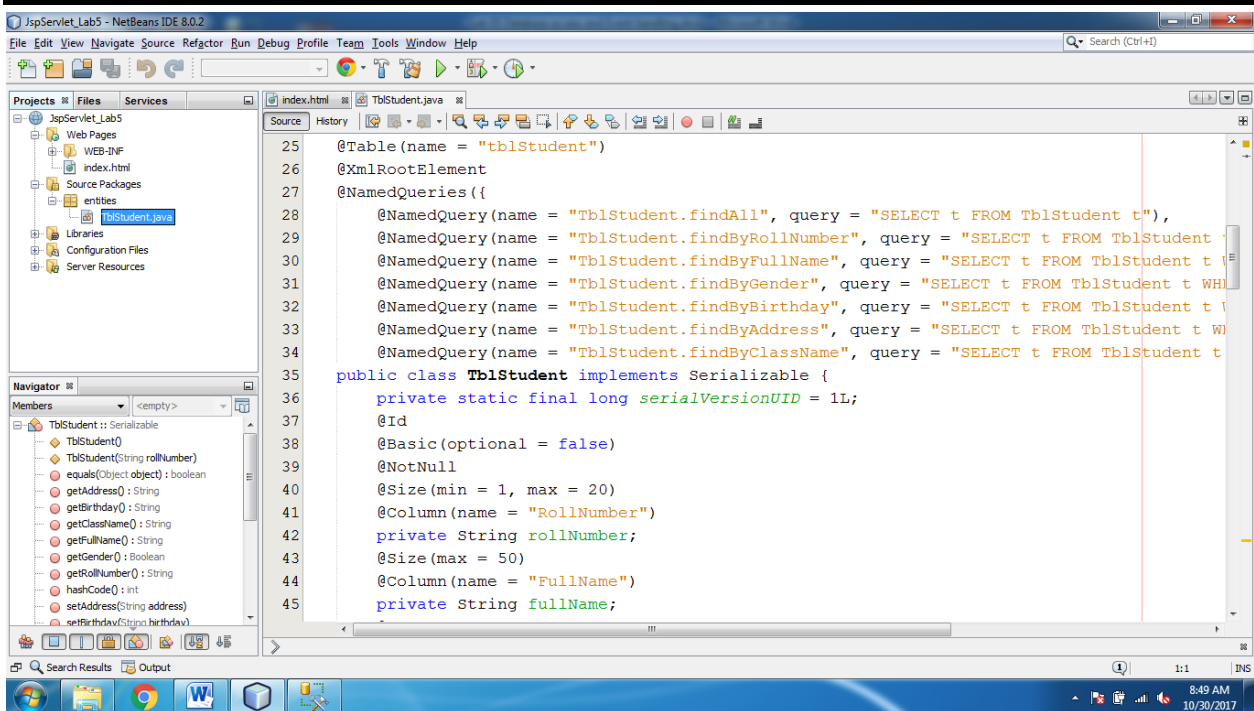


➔ **Next và Finish**

Chọn các bảng và chọn Add, sau đó chọn Next

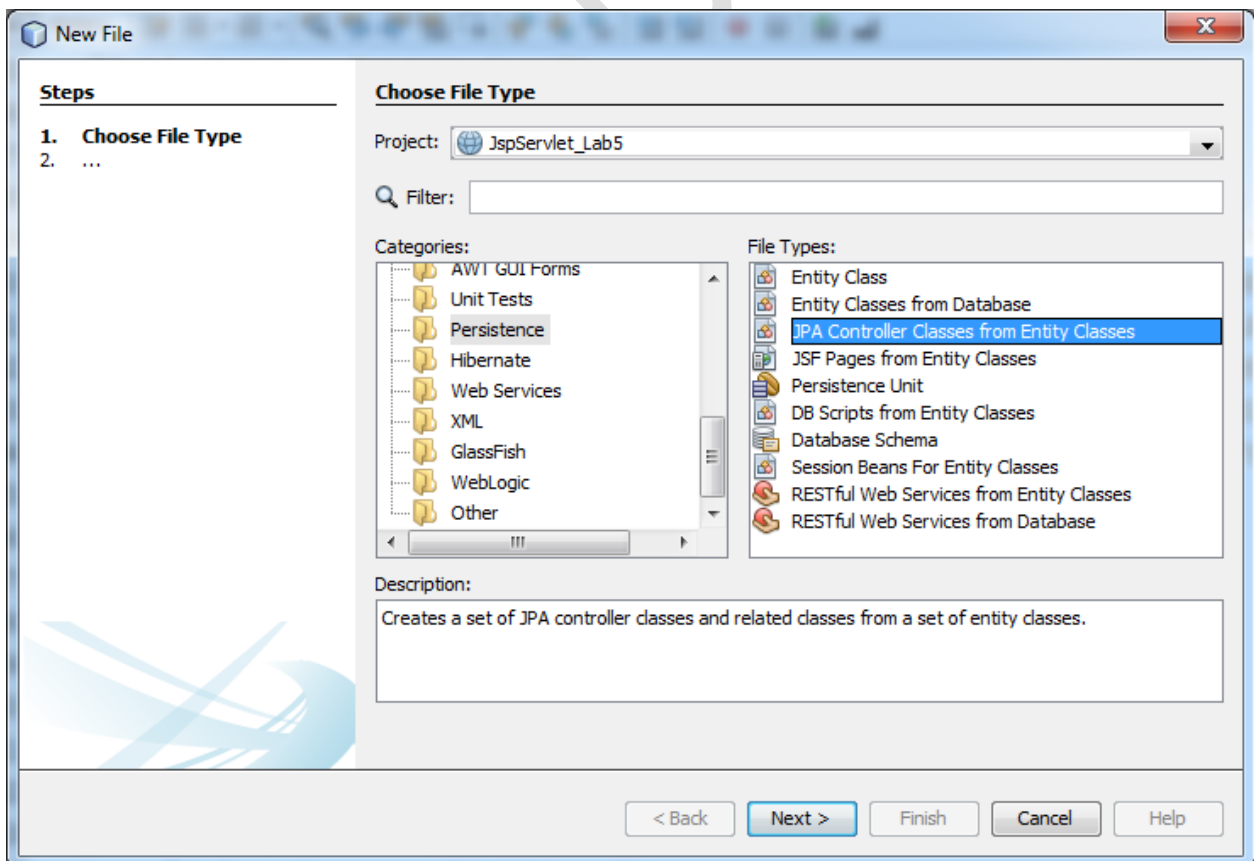


- Đặt tên cho package là “entities” và “Next -> Finish”.
- Khi đó JPA (viết tắt của Java Persistence API, chính là phần “Persistence” mà chúng ta đã chọn khi tạo Entity Class From Database), sẽ tạo ra các class tương ứng với các bảng và các mối quan hệ giữa các bảng đó (nếu có).



STEP 3: Tạo JPA controller cho các bảng vừa được sinh ra:

- Click phải vào project “New -> Other -> Persistence -> JPA Controller Classes from Entity Class”



- Chọn các bảng cần tạo controller và chọn “Add>”

- **Next** -> Đặt package là “dao” + **Finish**
- Chúng ta không sử dụng **UserTransaction** vì vậy trong class “TblStudentJpaController” được sinh ra bỏ đi thuộc tính “UserTransaction” và sửa lại constructor:
- Trong các biến “utx” ở các hàm, chúng ta thay bằng lệnh “em.getTransaction()”.
- Nội dung file này sẽ như sau:

```
package dao;

import dao.exceptions.NonexistentEntityException;
import dao.exceptions.PreexistingEntityException;
import dao.exceptions.RollbackFailureException;
import entities.TblStudent;
import java.io.Serializable;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import javax.transaction.UserTransaction;

/**
 *
 * @author HAITHANH
 */
public class TblStudentJpaController implements Serializable {

    public TblStudentJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }

    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(TblStudent tblStudent) throws PreexistingEntityException,
RollbackFailureException, Exception {
        EntityManager em = null;
        try {
            em.getTransaction().begin();
            em = getEntityManager();
            em.persist(tblStudent);
            em.getTransaction().commit();
        } catch (Exception ex) {
            try {
                em.getTransaction().rollback();
            }
        }
    }
}
```

```
    } catch (Exception re) {
        throw new RollbackFailureException("An error occurred attempting to roll back the transaction.",
re);
    }
    if (findTblStudent(tblStudent.getRollNumber()) != null) {
        throw new PreexistingEntityException("TblStudent " + tblStudent + " already exists.", ex);
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}

public void edit(TblStudent tblStudent) throws NonexistentEntityException, RollbackFailureException,
Exception {
    EntityManager em = null;
    try {
        em.getTransaction().begin();
        em = getEntityManager();
        tblStudent = em.merge(tblStudent);
        em.getTransaction().commit();
    } catch (Exception ex) {
        try {
            em.getTransaction().rollback();
        } catch (Exception re) {
            throw new RollbackFailureException("An error occurred attempting to roll back the transaction.",
re);
        }
        String msg = ex.getLocalizedMessage();
        if (msg == null || msg.length() == 0) {
            String id = tblStudent.getRollNumber();
            if (findTblStudent(id) == null) {
                throw new NonexistentEntityException("The tblStudent with id " + id + " no longer exists.");
            }
        }
        throw ex;
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public void destroy(String id) throws NonexistentEntityException, RollbackFailureException, Exception
{
    EntityManager em = null;
    try {
        em.getTransaction().begin();
        em = getEntityManager();
        TblStudent tblStudent;
        try {
            tblStudent = em.getReference(TblStudent.class, id);
            tblStudent.getRollNumber();
        }
```

```
    } catch (EntityNotFoundException enfe) {
        throw new NonexistentEntityException("The tblStudent with id " + id + " no longer exists.",
enfe);
    }
    em.remove(tblStudent);
    em.getTransaction().commit();
} catch (Exception ex) {
    try {
        em.getTransaction().rollback();
    } catch (Exception re) {
        throw new RollbackFailureException("An error occurred attempting to roll back the transaction.",
re);
    }
    throw ex;
} finally {
    if (em != null) {
        em.close();
    }
}
}

public List<TblStudent> findTblStudentEntities() {
    return findTblStudentEntities(true, -1, -1);
}

public List<TblStudent> findTblStudentEntities(int maxResults, int firstResult) {
    return findTblStudentEntities(false, maxResults, firstResult);
}

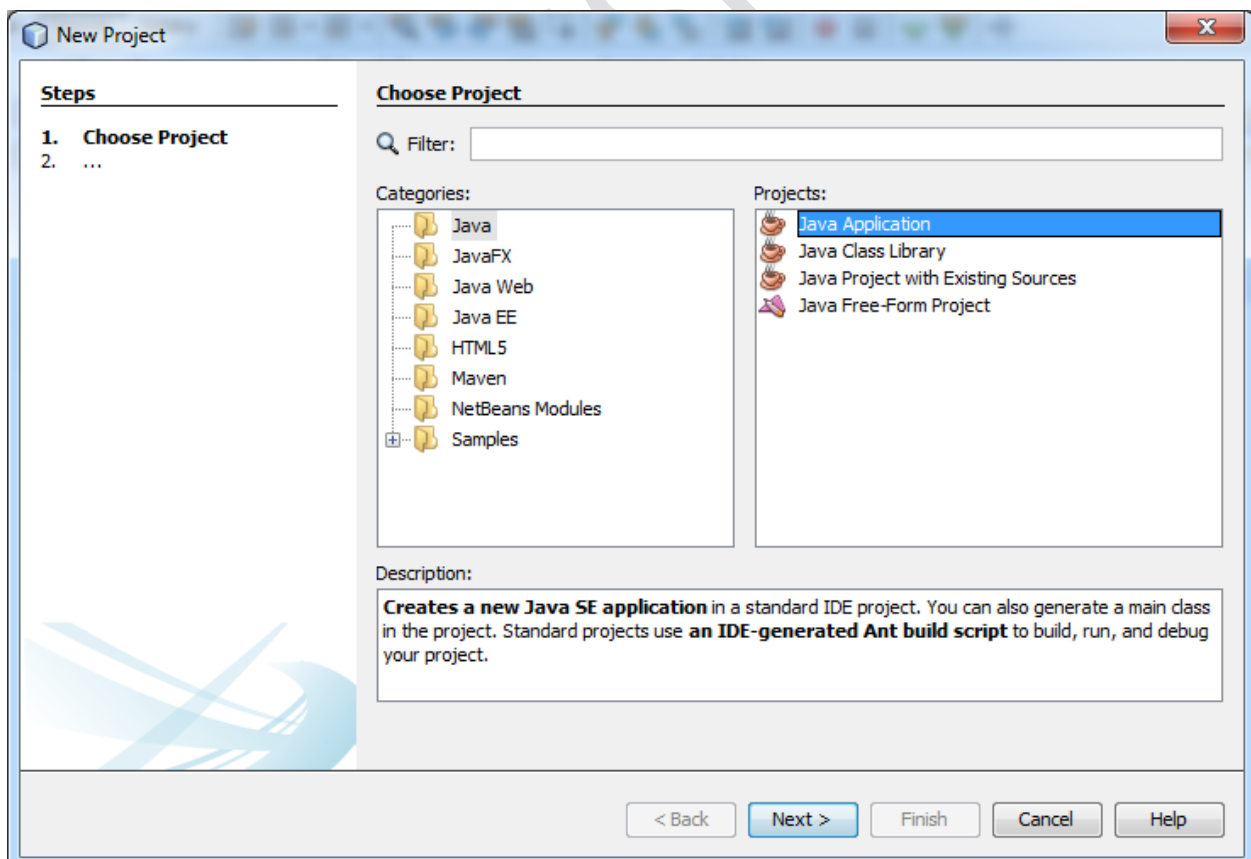
private List<TblStudent> findTblStudentEntities(boolean all, int maxResults, int firstResult) {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        cq.select(cq.from(TblStudent.class));
        Query q = em.createQuery(cq);
        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}

public TblStudent findTblStudent(String id) {
    EntityManager em = getEntityManager();
    try {
        return em.find(TblStudent.class, id);
    } finally {
        em.close();
    }
}
}
```

```
public int getTblStudentCount() {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        Root<TblStudent> rt = cq.from(TblStudent.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    } finally {
        em.close();
    }
}
```

STEP 4: Sửa file cấu hình “persistence.xml”

- Nếu là project swing thì JPA sinh ra có thể dùng được ngay, còn nếu là ứng dụng web thì file này cần phải sửa lại thì JPA mới có thể sử dụng được.
- Cách sửa rất đơn giản: Chúng ta sẽ tạo file này ở project swing (cho database và các bảng tương tự) rồi copy file này sang project web là xong.
- Tạo project swing: New Project -> Java -> Java Application -> Để mặc định tên và Finish

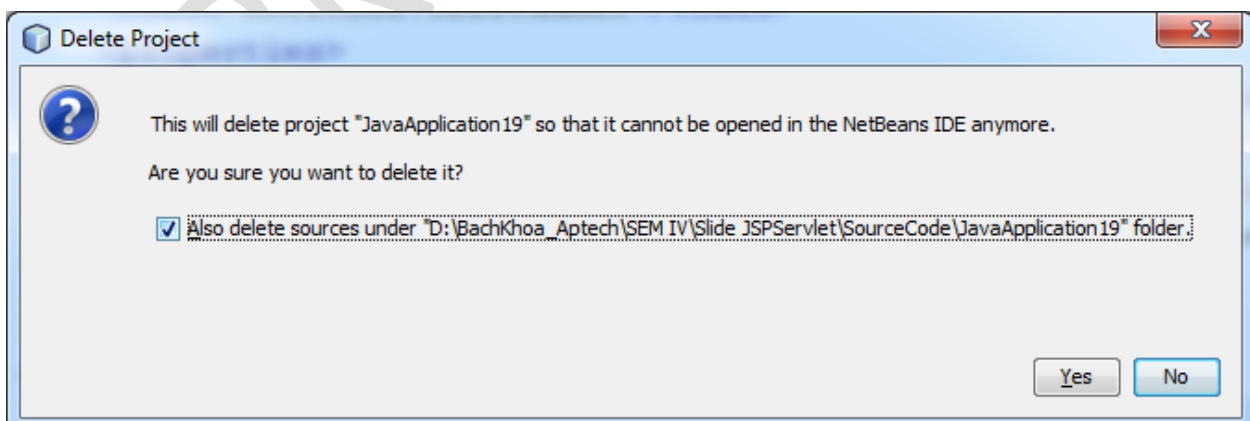


- Click phải vào project chọn: New -> Entity Classes from Database

- Database Connection: Chọn chuỗi kết nối với database ở bước trên -> Chọn các bảng và Add -> Next -> Đặt tên cho package là "entities" -> Next -> Finish
- Click phải vào project: New -> JPA Controller Classes from Entity Classes.. -> Chọn các bản + Add -> Next -> Finish. Khi đó file "persistence.xml" được sinh ra
- Vào "META-INF" copy file "persistence.xml" thay cho file "persistence.xml" ở project web. Vì không copy được luôn file nên chúng ta mở file ra: Select All -> Copy -> Mở file persistence.xml ở project web -> Select All -> Paste
- Nội dung của file "persistence.xml" như sau:

```
<?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="JavaApplication19PU" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>entities.TblStudent</class>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:sqlserver://localhost:1433;databaseName=JspServlet_Lab5"/>
      <property name="javax.persistence.jdbc.user" value="sa"/>
      <property name="javax.persistence.jdbc.driver"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
      <property name="javax.persistence.jdbc.password" value="1234$"/>
    </properties>
  </persistence-unit>
</persistence>
```

- Delete project swing đã tạo (nhớ tích chọn delete cả ổ đĩa đi)



STEP 5: Tạo class StudentDAO và gọi các hàm trong JPA Controller để tương tác với database.

- Click phải vào package “dao” -> New -> Java Class -> Đặt tên file “StudentDAO” -> Finish.
- Khai báo một thuộc tính EntityManager và khởi tạo trong constructor như sau:

```
package dao;

import entities.TblStudent;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author HAITHANH
 */
public class StudentDAO {
    private EntityManager em;

    public StudentDAO() {
        em = Persistence.createEntityManagerFactory("JavaApplication19PU").createEntityManager();
    }
}
```

- Trong đó “JavaApplication19PU” được lấy từ file “persistence.xml”
- Tạo phương thức để lấy về tất cả dữ liệu của bảng “tblStudent”

```
package dao;

import entities.TblStudent;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

/**
 *
 * @author HAITHANH
 */
public class StudentDAO {
    private EntityManager em;

    public StudentDAO() {
        em = Persistence.createEntityManagerFactory("JavaApplication19PU").createEntityManager();
    }

    //Hàm để lấy tất cả dữ liệu của bảng tblStudent
    public List<TblStudent> getAllStudents(){
        Query query = em.createNativeQuery("select * from tblStudent", TblStudent.class);
        return query.getResultList();
    }
}
```

```
}
```

STEP 6: Tạo các hàm cần tương tác với database còn lại như: Insert, Update, Delete, Find By Id,...

```
package dao;

import entities.TblStudent;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import javax.persistence.Query;

/**
 *
 * @author HAITHANH
 */
public class StudentDAO {

    private EntityManager em;

    public StudentDAO() {
        em = Persistence.createEntityManagerFactory("JavaApplication19PU").createEntityManager();
    }

    //Hàm để lấy tất cả dữ liệu của bảng tblStudent
    public List<TblStudent> getAllStudents() {
        Query query = em.createNativeQuery("select s from tblStudent s", TblStudent.class);
        return query.getResultList();
    }

    //Hàm để lấy về thông tin của 1 Student theo roll
    public TblStudent getStudentByRoll(String roll) {
        Query query = em.createNativeQuery("select s from tblStudent s where s.RollNumber = :rollNumber",
        TblStudent.class);
        query.setParameter("rollNumber", roll);
        return (TblStudent) query.getSingleResult();
    }

    //Hàm insert:
    public boolean addNewStudent(TblStudent stu) {
        em.getTransaction().begin();
        em.persist(stu);
        em.getTransaction().commit();
        return true;
    }

    //Hàm update:
    public boolean updateStudent(TblStudent stu) {
        em.getTransaction().begin();
        em.merge(stu);
        em.getTransaction().commit();
    }
}
```

```
        return true;
    }

    //Hàm delete:
    public boolean deleteStudent(String roll) {
        TblStudent stu = em.find(TblStudent.class, roll);
        em.getTransaction().begin();
        em.remove(stu);
        em.getTransaction().commit();
        return true;
    }
}
```

- ***Từ các hàm trong DAO này bạn hãy tạo tiếp các Servlet và các trang .jsp để thực hiện các chức năng của một ứng dụng web: Load All, Detail, Insert, Update, Delete.***

Bài 1.2

Mục tiêu:

- Thực hành với Event Handling trong JSP và Servlet
- ServletContextListener

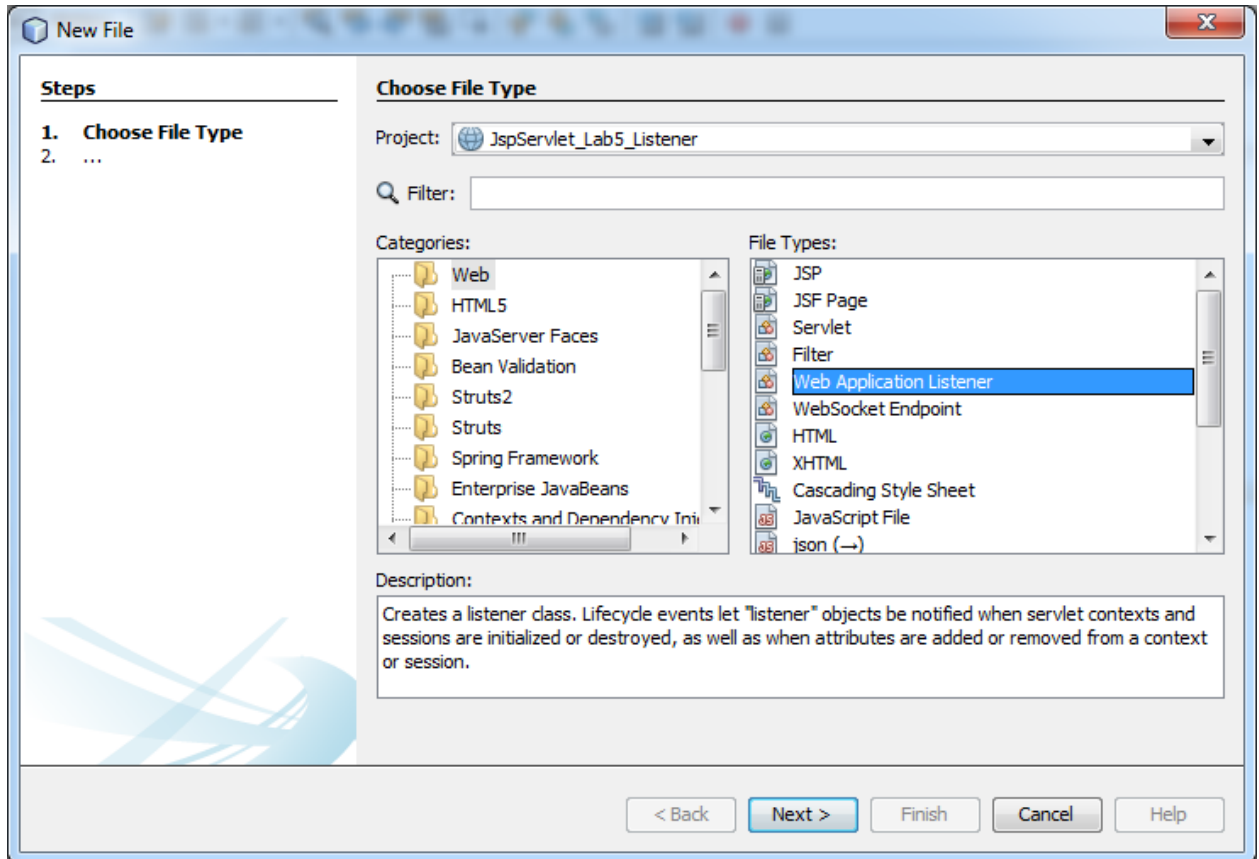
ServletContextListener được sử dụng khi chúng ta muốn một đoạn code được thực thi trước khi web application khởi động.

STEP 1: Tạo Project.

- New Project -> Java Web -> Web Application -> Đặt tên project "JspServlet_Lab5_Listener" -> Next -> Next -> Finish.

STEP 2: Tạo ServletListener

- Right click Source Packages -> New -> Other -> Web -> Web Application Listener -> Đặt tên file "MyServletListener", package "listener" -> Finis



- Có 2 phương thức được sinh ra, một phương thức được gọi khi ứng dụng được deploy và một phương thức được gọi khi tắt Web server đi.
- Chúng ta sẽ viết lệnh và chạy ứng dụng để thấy sự hoạt động của hai phương thức này.

```
package listener;

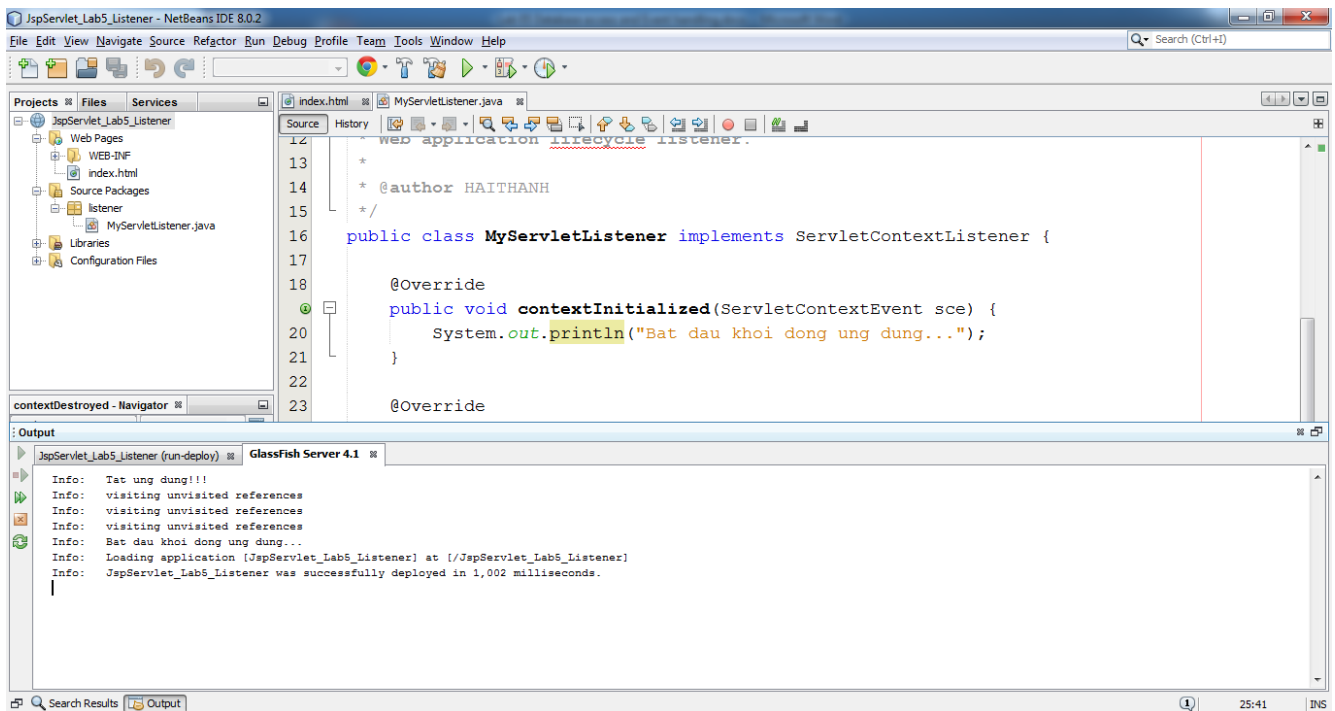
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

/**
 * Web application lifecycle listener.
 *
 * @author HAITHANH
 */
public class MyServletListener implements ServletContextListener {

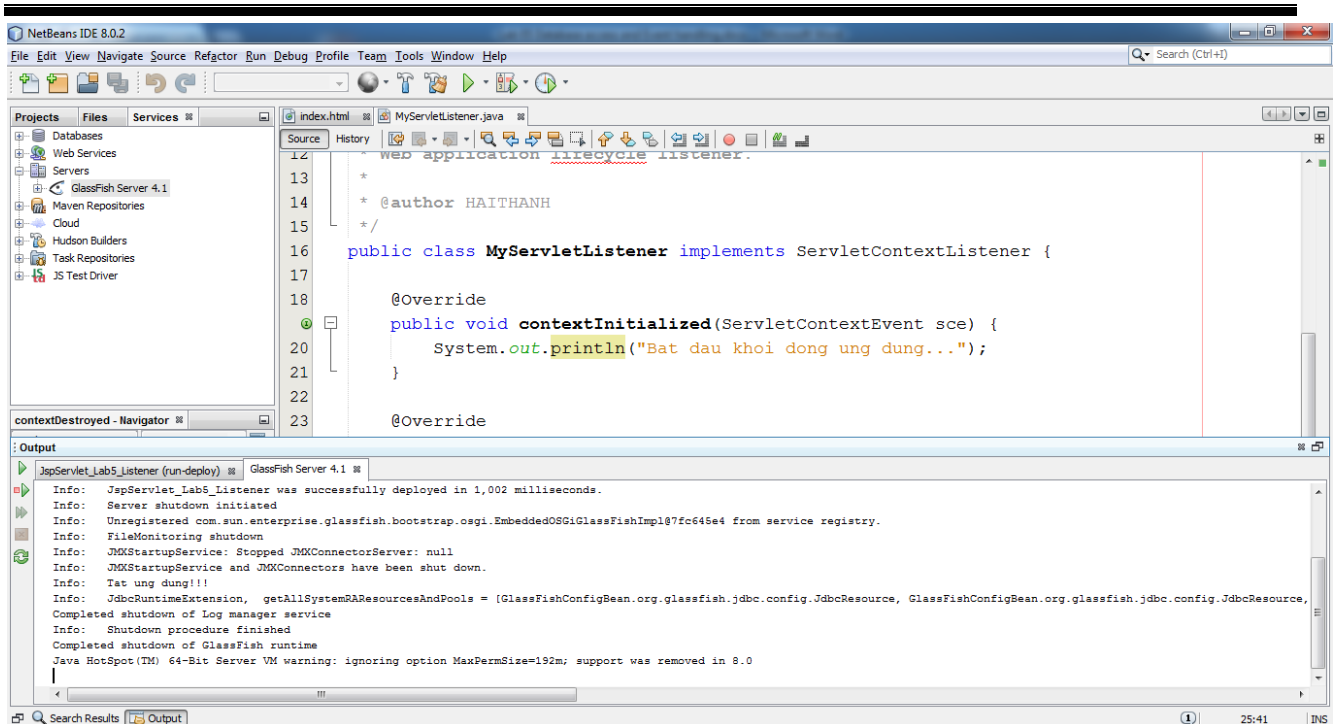
    @Override
    public void contextInitialized(ServletContextEvent sce) {
        System.out.println("Bắt đầu khởi động ứng dụng...");
    }
}
```

```
@Override
public void contextDestroyed(ServletContextEvent sce) {
    System.out.println("Tắt ứng dụng!!!");
}
}
```

Deploy và run ứng dụng chúng ta sẽ có kết quả:



Khi chúng ta tắt Glash Fish thì sẽ được kết quả:



Bài 1.3

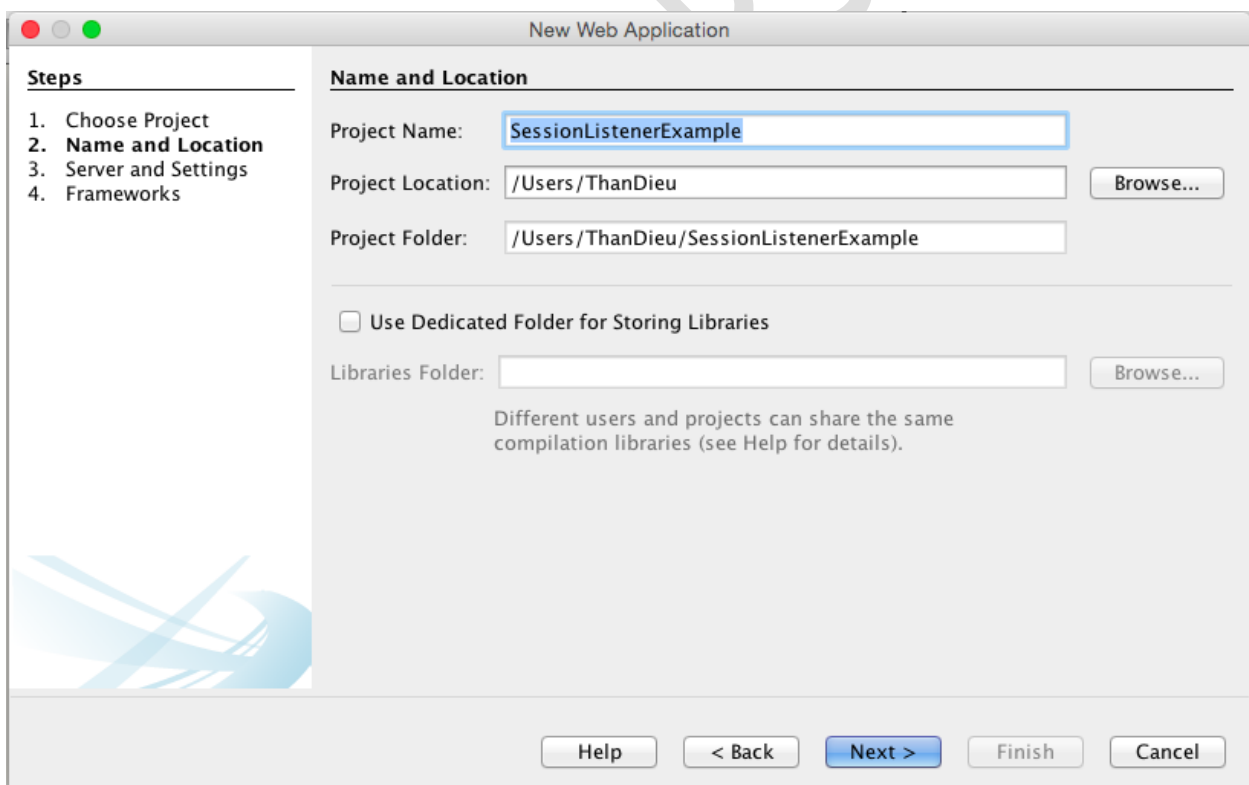
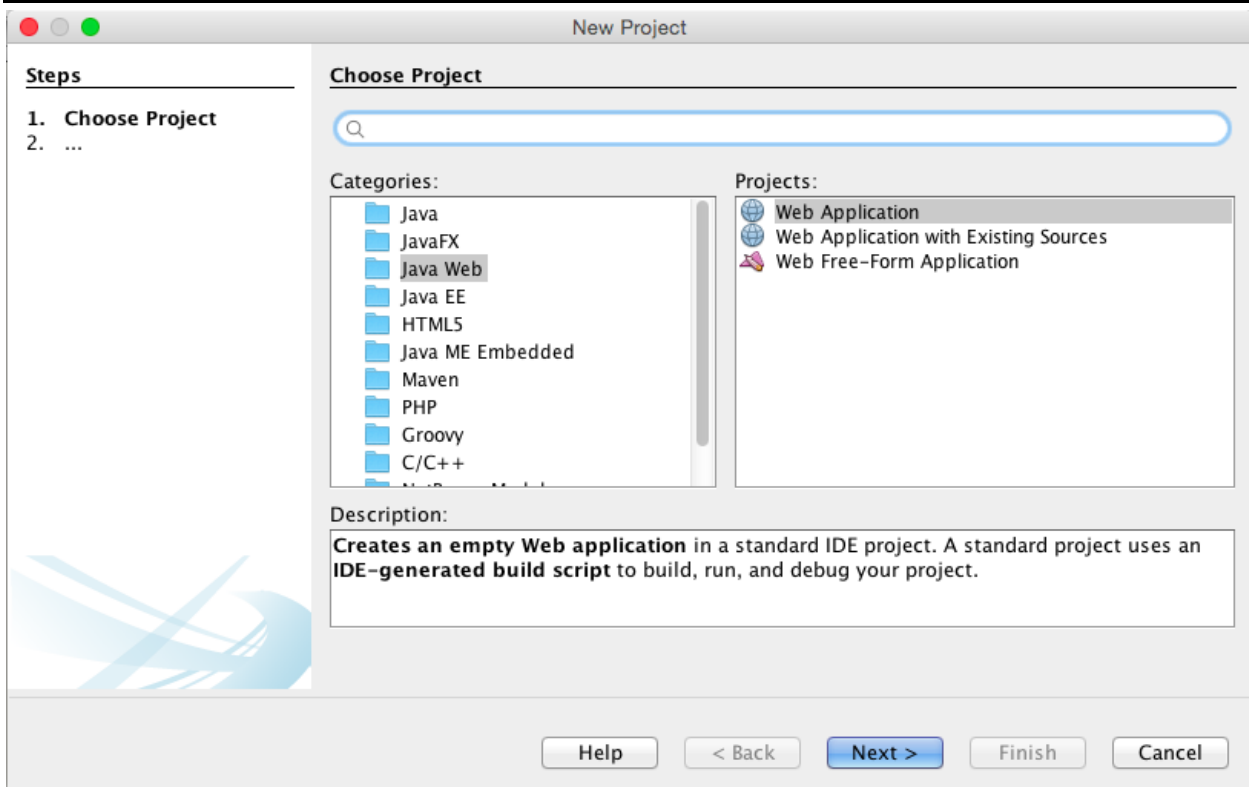
Mục tiêu:

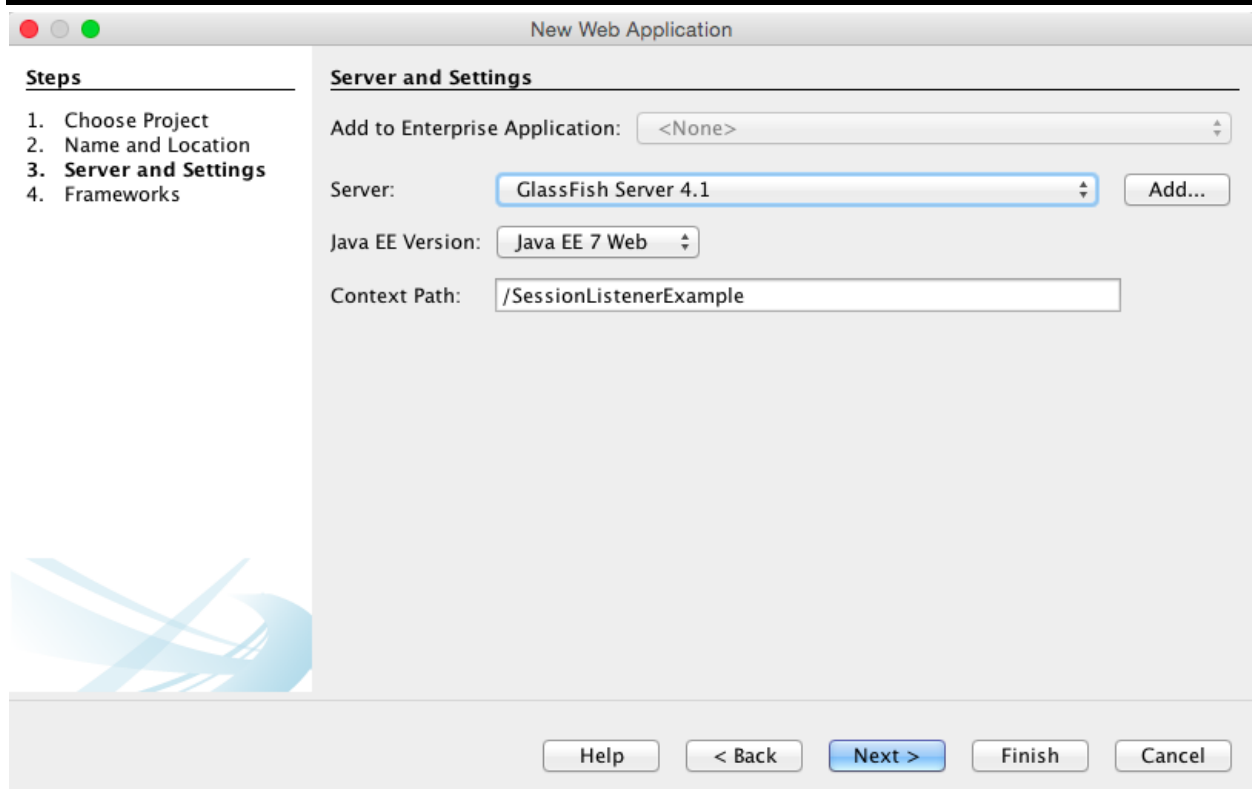
- Thực hành với HttpSessionListener

HttpSessionListener được sử dụng khi chúng ta muốn một đoạn code nào đó được thực hiện khi có session được khởi tạo hoặc đếm số session được khởi tạo trên toàn hệ thống.

Project SessionListenerExample sẽ đếm số session được tạo ra trên toàn hệ thống.

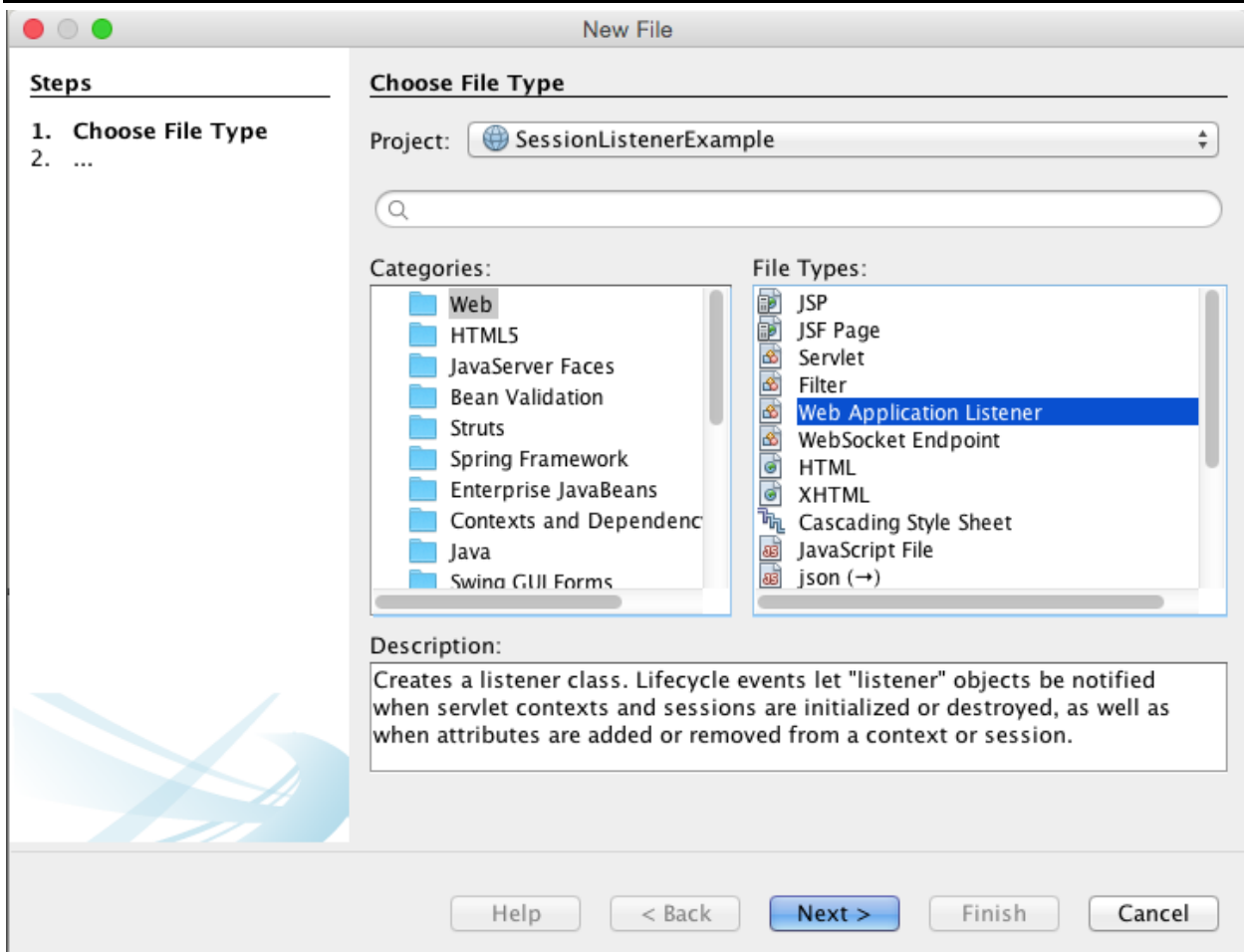
STEP 1: Tạo web project SessionListenerExample





STEP 2: Tạo ControllerSessionListener

Tạo package listener -> Tạo ControllerSessionListener



Steps

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

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

Interfaces to implement:

- ☐ Context Listener
- ☐ Context Attribute Listener
- ☒ HTTP Session Listener
- ☐ HTTP Session Attribute Listener
- ☐ Request Listener (J2EE 1.4)
- ☐ Request Attribute Listener (J2EE 1.4)

Description:

Context listener observes when servlet context is created or is about to be shut down.

Help < Back Next > **Finish** Cancel

Chọn HTTP Session Listener -> Finish

Hai method cần được override: sessionCreated và sessionDestroyed

```
package listener;

import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;

/**
 * Web application lifecycle listener.
 *
 * @author ThanDieu
 */
public class ControllerSessionListener implements HttpSessionListener {

    private int sessionCounter = 0;

    @Override
    public void sessionCreated(HttpSessionEvent se) {
        synchronized (this){
            sessionCounter++;
        }
    }
}
```

```
        System.out.println("Session Created: " + se.getSession().getId());
        System.out.println("Total sessions: " + sessionCounter);
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        synchronized (this) {
            sessionCounter --;
        }
        System.out.println("Session Destroyed: " + se.getSession().getId());
        System.out.println("Total sessions: " + sessionCounter);
    }
}
```

Mỗi lần đếm session cần synchronized với các request khác.

Mở file web.xml, đảm bảo listener config đã được thêm vào:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <listener>
        <description>HttpSessionListener</description>
        <listener-class>listener.ControllerSessionListener</listener-class>
    </listener>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

STEP 3: Tạo 3 trang jsp: index.jsp; addUser.jsp và destroySession.jsp

Tạo trang index.jsp để list các user.

```
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Servlet Session Listener Example</title>
    </head>
    <body>
        <h1>Add users:</h1>
        <span style="float: right">
            <a href="destroySession.jsp">Destroy this session</a>
        </span>
    </body>
</html>
```

```
<form method="post" action="addUser.jsp">
    <h3>Enter Username to Add in List</h3>
    <input type="text" name="user"/>
    <input type="submit" value="Add User"/>
</form>
<%
    List<String> users = (List<String>) session.getAttribute("users");
    for (int i = 0; null != users && i < users.size(); i++) {
        out.println("<br/>" + users.get(i));
    }
%>
</body>
</html>
```

Trang addUser.jsp sẽ add user vào session.

```
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.List"%>
<%
    String username = request.getParameter("user");
    List<String> users = (List<String>)session.getAttribute("users");

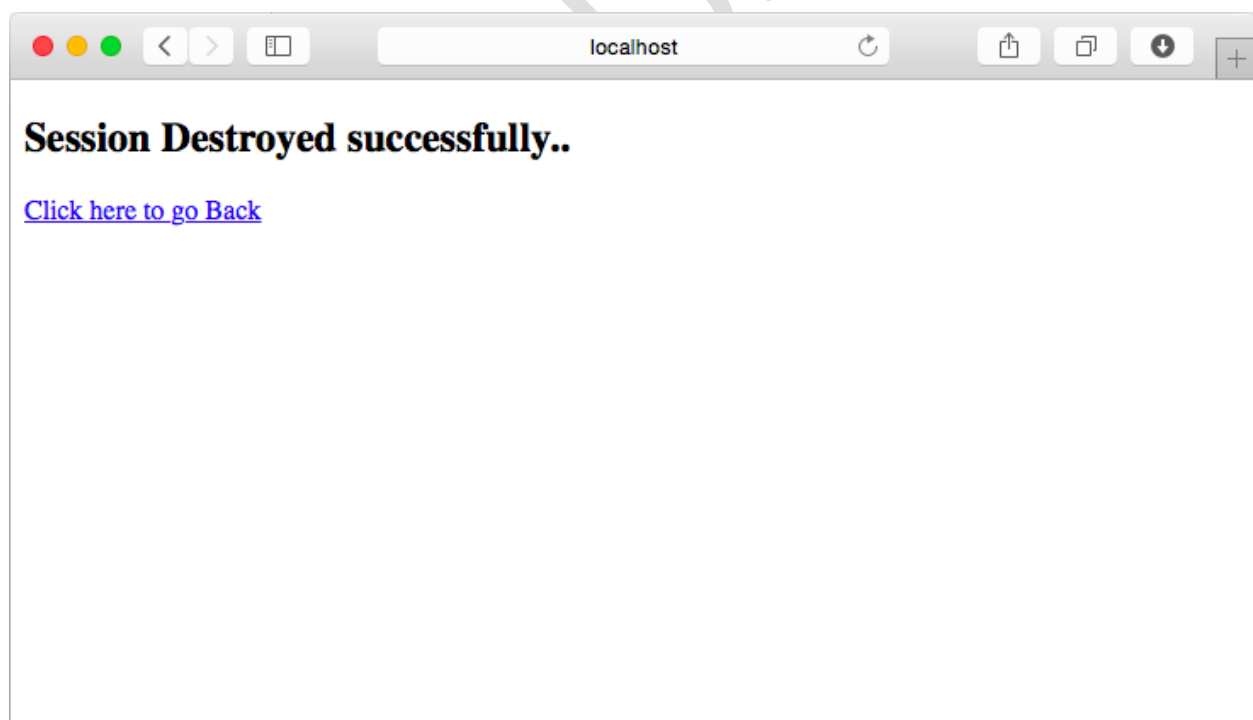
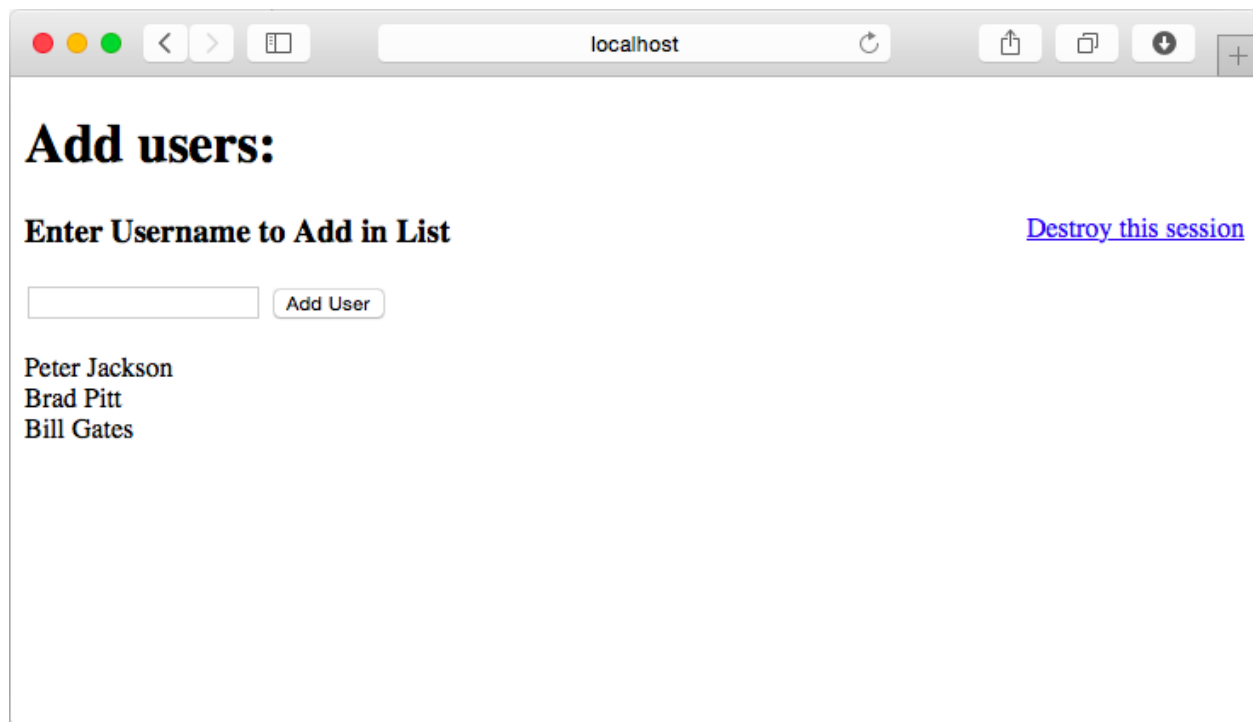
    if(null == users) {
        users = new ArrayList<String>();
    }
    users.add(username);
    session.setAttribute("users", users);
    response.sendRedirect("index.jsp");
%>
```

Trang destroySession.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            session.invalidate();
        %>
        <h2>Session Destroyed successfully.. </h2>
        <a href="javascript:history.back()">Click here to go Back</a>
    </body>
</html>
```

Trong bài này chúng ta sử dụng 3 implicit object của JSP: request, session và response.

Chạy chương trình và view kết quả.





Phần 2: Bài tập tự làm

Cho database sau:

```
create database Lab5_Exam
go
use Lab5_Exam
go
create table Categories(
    CategoryID varchar(10) not null primary key,
    CategoryName varchar(20),
    Description varchar(100));

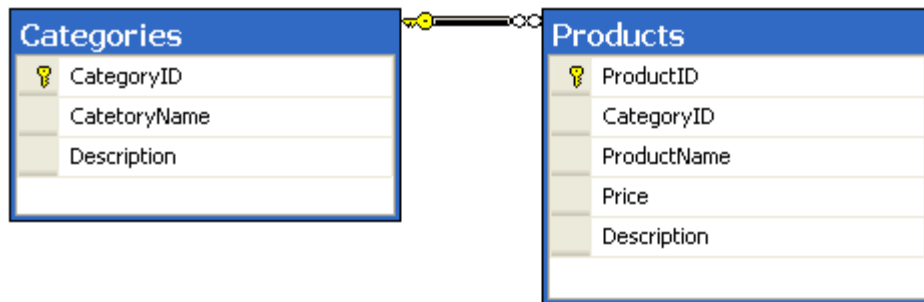
create table Products(
    ProductID varchar(10) not null primary key,
    CategoryID varchar(10),
    ProductName varchar(30),
    Price int,
    Description varchar(100)
    constraint fk_cat_pro foreign key (CategoryID) references Categories(CategoryID));

insert into Categories values('C0101', 'Beverages', 'Soft drinks, coffees, teas, beers, and ales');
insert into Categories values('C0102', 'Condiments', 'Sweet and savory sauces, relishes, spreads,
    and seasonings');
insert into Categories values('C0103', 'Confections', 'Desserts, candies, and sweet breads');
insert into Categories values('C0104', 'Dairy Products', 'Cheeses');
insert into Categories values('C0105', 'Grains_Cereals', 'Breads, crackers, pasta, and cereal');
insert into Categories values('C0106', 'Meat_Poultry', 'Prepared meats');
insert into Categories values('C0107', 'Produce', 'Dried fruit and bean curd');
insert into Categories values('C0108', 'Seafood', 'Seaweed and fish');

insert into Products values('P0101', 'C0101', 'Chai', 18, 'Good');
insert into Products values('P0102', 'C0101', 'Chang', 19, 'Good');
insert into Products values('P0103', 'C0101', 'Aniseed Syrup', 10, 'Good');
insert into Products values('P0104', 'C0102', 'Chef Anton's Cajun Seasoning', 22, 'Good');
insert into Products values('P0105', 'C0102', 'Chef Anton's Gumbo Mix', 21.35, 'Good');
insert into Products values('P0106', 'C0103', 'Grandma's Boysenberry Spread', 25, 'Good');
insert into Products values('P0107', 'C0103', 'Uncle Bob's Organic Dried Pears', 30, 'Good');
insert into Products values('P0108', 'C0103', 'Northwoods Cranberry Sauce', 40, 'Good');
```

```
insert into Products values('P0109', 'C0104', 'Mishi Kobe Niku', 97, 'Good');
insert into Products values('P01010', 'C0104', 'Ikura', 31, 'Good');
insert into Products values('P01011', 'C0105', 'Queso Cabrales', 21, 'Good');
insert into Products values('P01012', 'C0105', 'Queso Manchego La Pastora', 38, 'Good');
```

```
select * from Categories;
select * from Products;
```



Sử dụng JPA để tạo ra các hàm tương tác với database theo các chức năng dưới đây của ứng dụng web:

Trang “index.jsp”

Category Information

CateogyrID	CateogoryName	Product Details
C0101	Beverages	C0101
C0102	Condiments	C0102
C0103	Confections	C0103
C0104	Dairy Products	C0104
C0105	Grains_Cereals	C0105
C0106	Meat_Poultry	C0106
C0107	Produce	C0107
C0108	Seafood	C0108

Khi người dùng chọn Product Details thì hiển thị trang “productDetail.jsp” với nội dung như sau:

Category Information

ProductID	ProductName	Price	CatetoryName	Description
P0102	Chang	19	Beverages	Good
P0103	Aniseed Syrup	10	Beverages	Good

HẾT