



# LẬP TRÌNH JAVA 4

## BÀI 7: HIBERNATE

### PHẦN 1

- ◎ Kết thúc bài học này bạn có khả năng
  - ❖ Giới thiệu về Hibernate
  - ❖ So sánh Hibernate và JDBC
  - ❖ Các bước tạo ứng dụng Hibernate
  - ❖ Ví dụ xây dựng ứng dụng MVC – Hibernate
  - ❖ Các config cơ bản file mapping.hbm.xml
  - ❖ Các cấu hình tùy chọn khác



- ❑ Cơ sở dữ liệu thường được thiết kế và lưu trữ theo hướng quan hệ.
  - ❑ Tuy nhiên phần mềm thường được xây dựng theo hướng đối tượng.
  - ❑ Đối với lập trình viên khi xây dựng phần mềm thường muốn làm việc với các đối tượng và không phải nhớ đến các dòng, các cột trong các bảng của cơ sở dữ liệu
  - ❑ JDBC xử lý các CSDL lớn rất chậm (từ 1000 bảng trở lên), đặc biệt nếu ứng dụng có khả năng kết nối với nhiều hệ quản trị khác nhau, câu truy vấn trong mỗi hệ quản trị có thể có cú pháp khác
- Khó vận hành, bảo trì và xử lý sự cố với các CSDL lớn

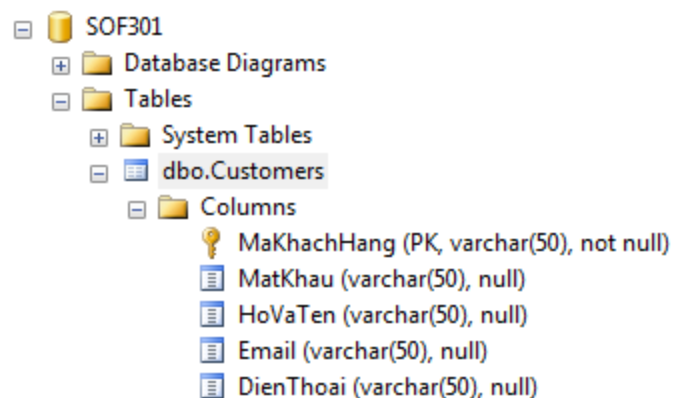
- ❑ Hibernate được phát triển bởi Gavin King từ năm 2001, là một ORM framework thuần Java. (Object-Relational Mapping)
- ❑ Hibernate giúp lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh, cho phép bạn truy vấn dữ liệu bằng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc bằng SQL thuần.
- ❑ Hibernate đóng vai trò là tầng trung gian giữa các đối tượng và CSDL
- ❑ Hibernate ánh xạ các lớp thực thể vào các bảng của CSDL quan hệ thông qua XML hoặc annotation.
  - ❖ Trong môn này chúng ta dùng XML
- ❑ Hibernate hỗ trợ hầu hết các CSDL hiện nay

## ❑ So sánh 2 giải pháp JDBC và Hibernate

JDBC	Hibernate
Với JDBC, lập trình viên phải viết mã để lập bản đồ dữ liệu của một mô hình đối tượng sang dữ liệu của một mô hình quan hệ và ngược lại để truy cập dữ liệu tương ứng của nó	Hibernate là giải pháp ORM linh hoạt và mạnh mẽ để lập bản đồ các lớp Java vào các bảng cơ sở dữ liệu. Hibernate tự quản lý bản đồ này bằng cách sử dụng các tập tin XML, vì vậy lập trình viên không cần phải viết mã cho điều này.
Với JDBC, các bản đồ của các đối tượng Java với các bảng cơ sở dữ liệu và ngược lại được quản lý bởi các lập trình viên bằng tay	Hibernate cung cấp một hệ thống transparent persistence.
JDBC chỉ hỗ trợ native Structured Query Language (SQL). Lập trình viên phải tự tìm ra cách phù hợp để truy cập cơ sở dữ liệu	Hibernate cung cấp HQL mở rộng ( không phụ thuộc vào kiểu cơ sở dữ liệu ) đồng thời vẫn hỗ trợ SQL
Ứng dụng sử dụng JDBC để xử lý các dữ liệu liên tục Nếu bảng cơ sở dữ liệu thay đổi thì lập trình viên phải viết lại mã	Hibernate cung cấp bản đồ tự động. Nếu có sự thay đổi trong cơ sở dữ liệu thì chỉ cần sửa file XML
Với JDBC, bộ đệm được quản lý bằng mã viết tay	Hibernate, với Transparent Persistence, bộ đệm được đặt vào không gian làm việc của ứng dụng. Điều này nâng cao hiệu năng của ứng dụng nếu ứng dụng đọc cùng một dữ liệu nhiều lần

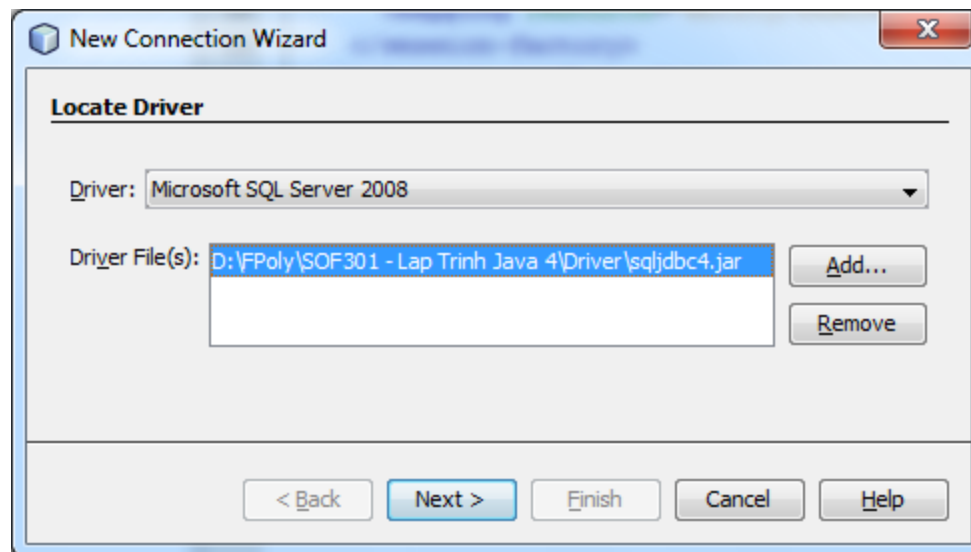
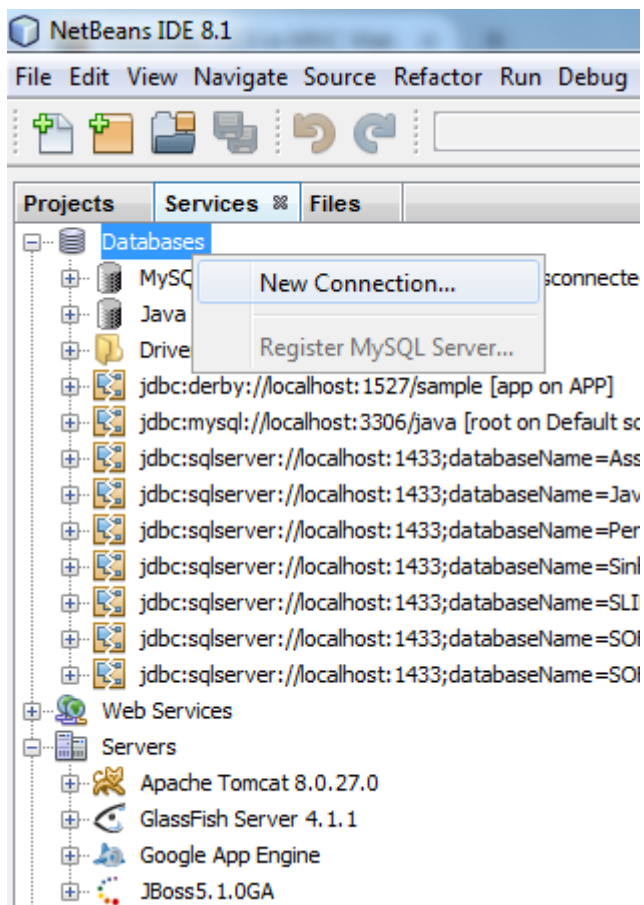
- ☐ Bước 1: Tạo cơ sở dữ liệu
- ☐ Bước 2: Tạo kết nối CSDL
- ☐ Bước 3: Tạo Web App
- ☐ Bước 4: Tạo file HiberntaeUtil.java
- ☐ Bước 5: Tạo Hibernate Mapping File và POJO
  - ❖ 5.1, Tạo Hibernate Reverse Engineering file
  - ❖ 5.2, Tạo Hibernate Mapping File và POJO
- ☐ Bước 6: Xây dựng các DAO & Sử dụng

# 1, Tạo cơ sở dữ liệu



SCD050718.SOF301 - dbo.Customers					
	MaKhachHang	MatKhau	HoVaTen	Email	DienThoai
	KH01	123	Nguyen Nghiem	nghiemn@fe.edu.vn	0913745789
	KH02	abc	Tran Duy Phong	phongtd@fe.edu.vn	0933922487
	KH03	123	Le Pham Tuan Kiet	kietlpt@fe.edu.vn	0903991033
	KH04	abc	Le Van Phung	phunglv@fe.edu.vn	0903414749
	KH05	bcd	Bui Minh Nhat	nhatbm@fe.edu.vn	0932030958
▶*	NULL	NULL	NULL	NULL	NULL

## ❑ 2, Tạo kết nối CSDL



Chỉ đường dẫn file sqljdbc4.jar



## ❑ 2, Tạo kết nối CSDL

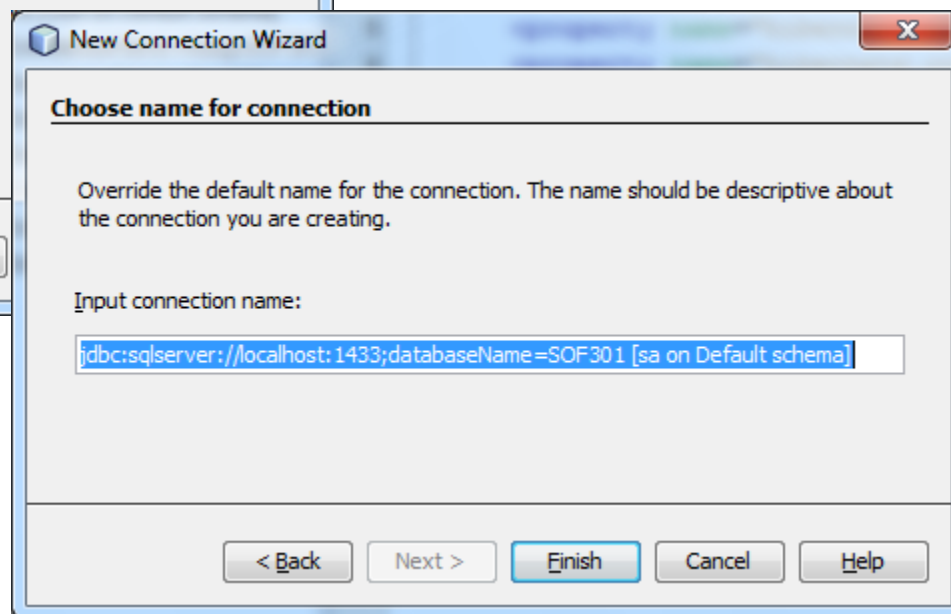
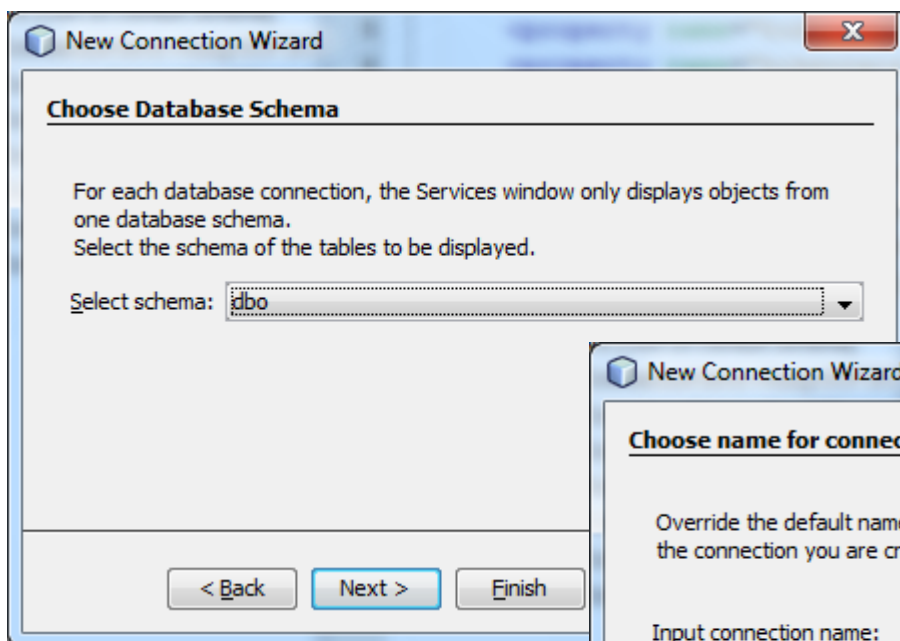
The screenshot shows the 'New Connection Wizard' dialog box, specifically the 'Customize Connection' step. The fields are filled as follows:

- Driver Name: Microsoft SQL Server 2005 on Microsoft SQL Server 2008
- Host: localhost
- Port: 1433
- Database: SOF301
- Instance Name: (empty)
- User Name: sa
- Password: (empty)
- ☒ Remember password
- Buttons: Connection Properties, Test Connection
- JDBC URL: jdbc:sqlserver://localhost:1433;databaseName=SOF301 (highlighted with an orange box)

At the bottom, there are navigation buttons: < Back, Next >, Finish, Cancel, and Help.

Khai báo các thông tin kết nối database

## ❑ 2, Tạo kết nối CSDL



## 3, Tạo Web App

The image displays two sequential screenshots of the 'New Web Application' wizard in an IDE.

**First Screenshot: 'Name and Location' Step**

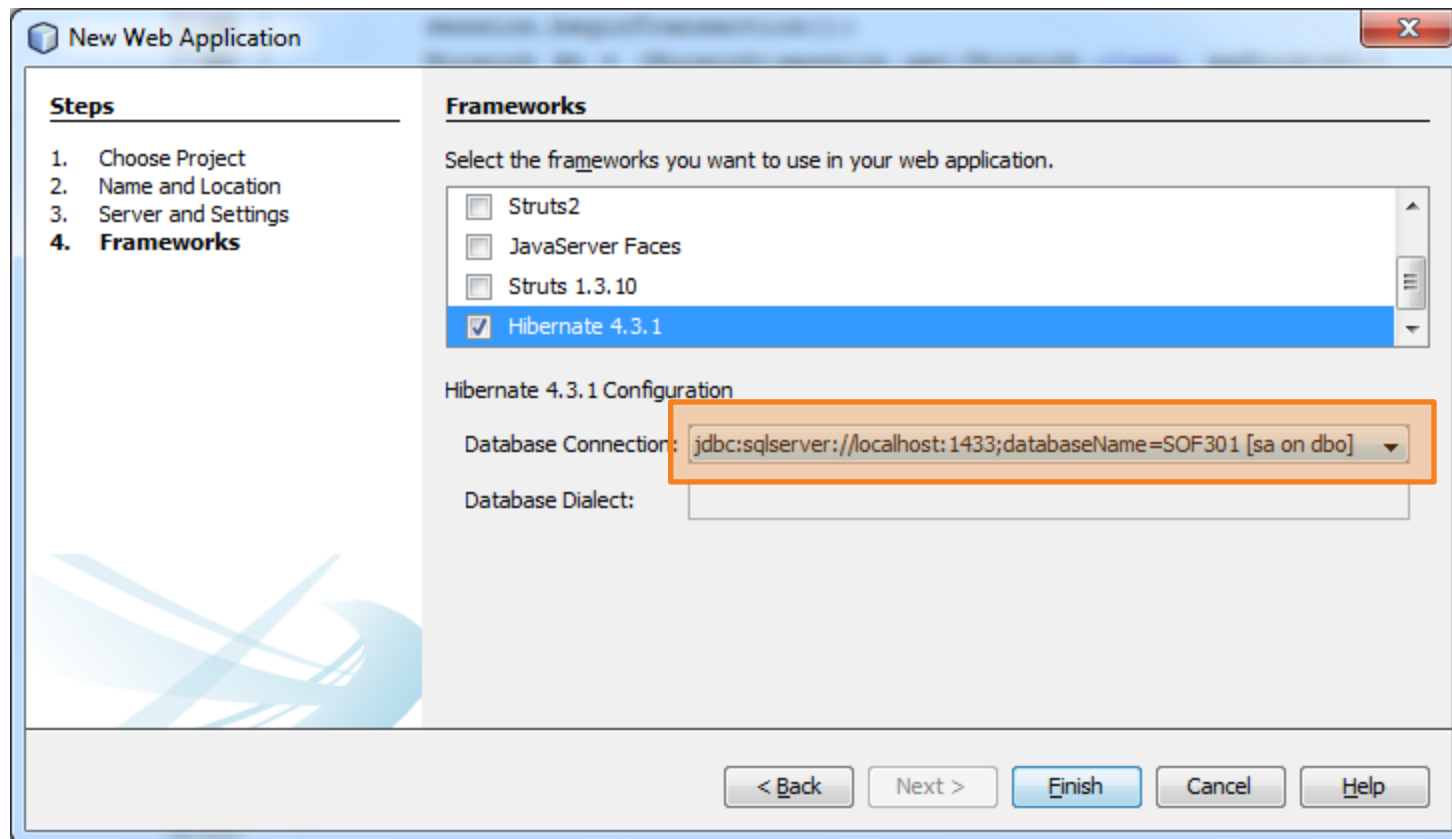
- Steps:**
  1. Choose Project
  - 2. Name and Location**
  3. Server and Settings
  4. Frameworks
- Name and Location:**
  - Project Name: SOF301Slide7Demo
  - Project Location: D:\DemoSOF301\Slide7 (with a 'Browse...' button)
  - Project Folder: D:\DemoSOF301\Slide7\SOF301Slide7Demo
  - ☐ Use Dedicated Folder for Storing Libraries
  - Libraries Folder: (empty)

**Second Screenshot: 'Server and Settings' Step**

- Steps:**
  1. Choose Project
  2. Name and Location
  - 3. Server and Settings**
  4. Frameworks
- Server and Settings:**
  - Add to Enterprise Application: <None>
  - Server: Apache Tomcat 8.0.27.0 (with an 'Add...' button)
  - Java EE Version: Java EE 7 Web
  - Note: Source Level 7 will be set for Java EE 7 project.
  - Context Path: /SOF301Slide7Demo

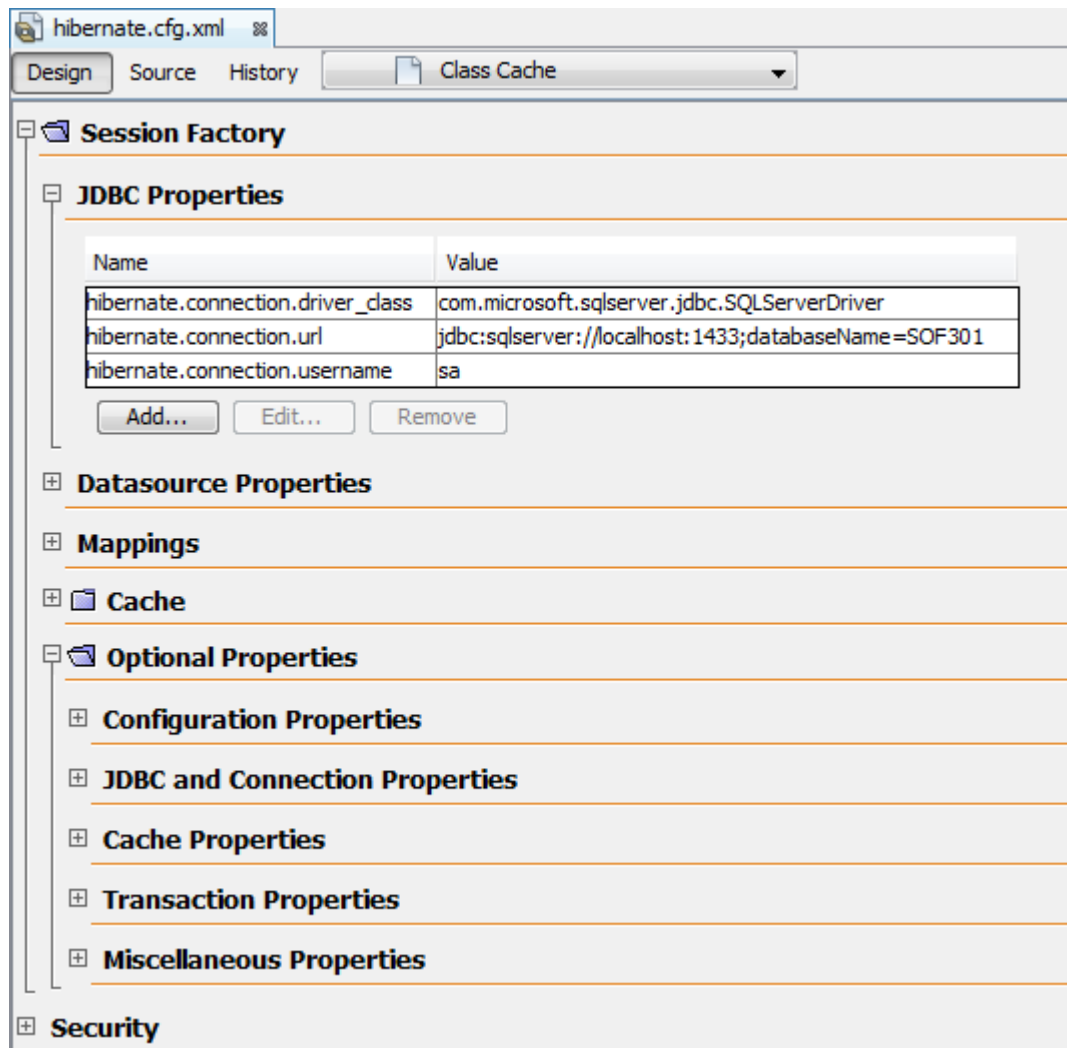
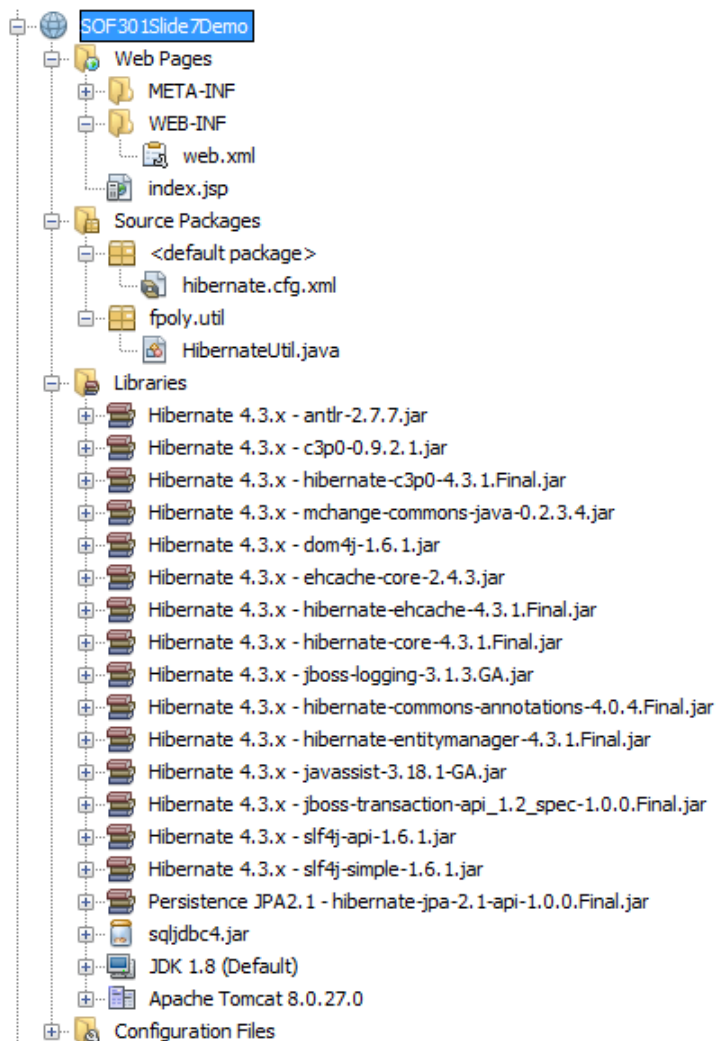
Navigation buttons at the bottom of the second window include: < Back, Next >, Finish, Cancel, and Help.

## 3, Tạo Web App



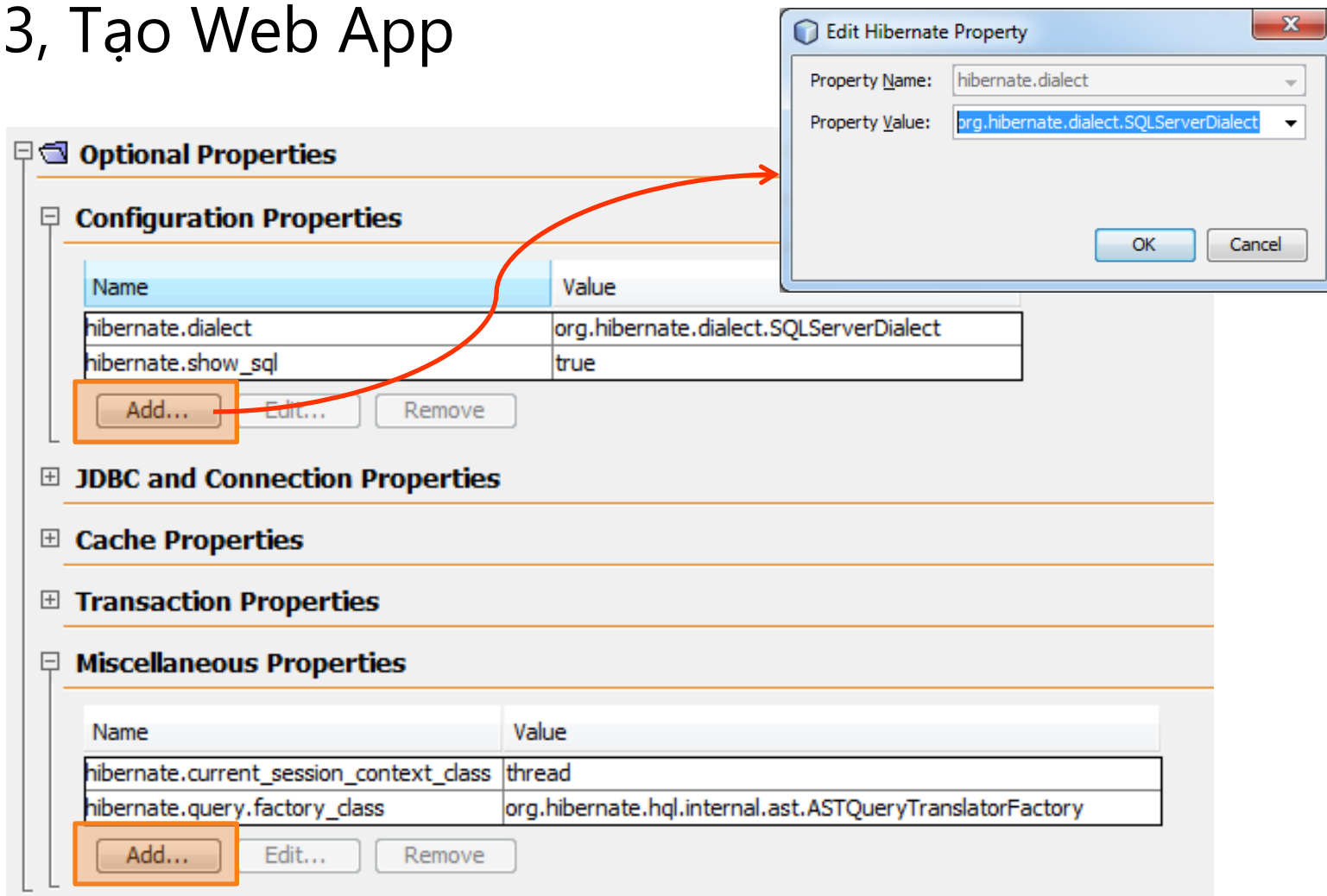
Click chọn “Hibernate” và chuỗi kết nối đã tạo ở bước 1

## 3, Tạo Web App



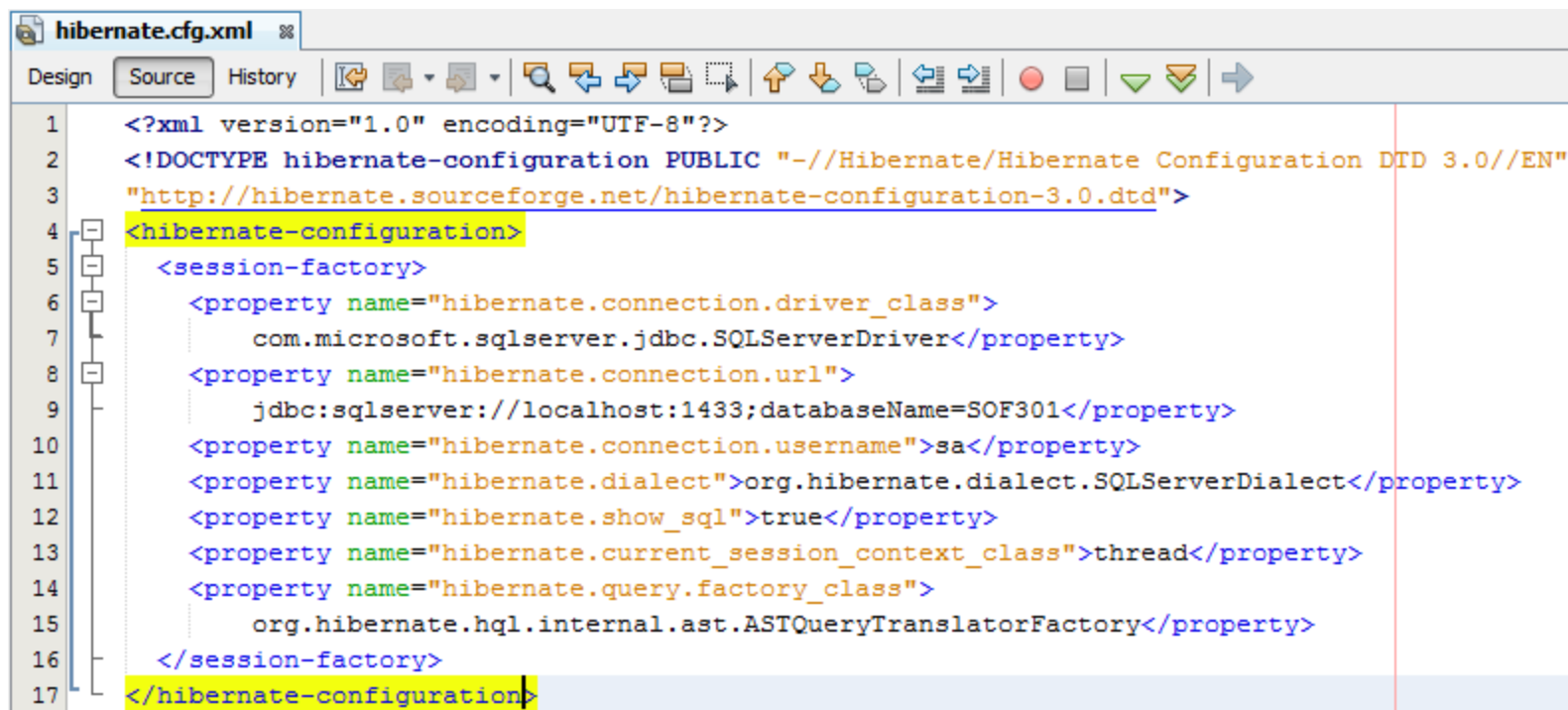
Kết quả Web App vừa tạo

## 3, Tạo Web App



Config file hibernate.cfg.xml như hình

## 3, Tạo Web App



```
hibernate.cfg.xml
Design Source History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
3 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
4 <hibernate-configuration>
5   <session-factory>
6     <property name="hibernate.connection.driver_class">
7       com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
8     <property name="hibernate.connection.url">
9       jdbc:sqlserver://localhost:1433;databaseName=SOF301</property>
10    <property name="hibernate.connection.username">sa</property>
11    <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>
12    <property name="hibernate.show_sql">true</property>
13    <property name="hibernate.current_session_context_class">thread</property>
14    <property name="hibernate.query.factory_class">
15      org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory</property>
16    </session-factory>
17  </hibernate-configuration>
```

file hibernate.cfg.xml

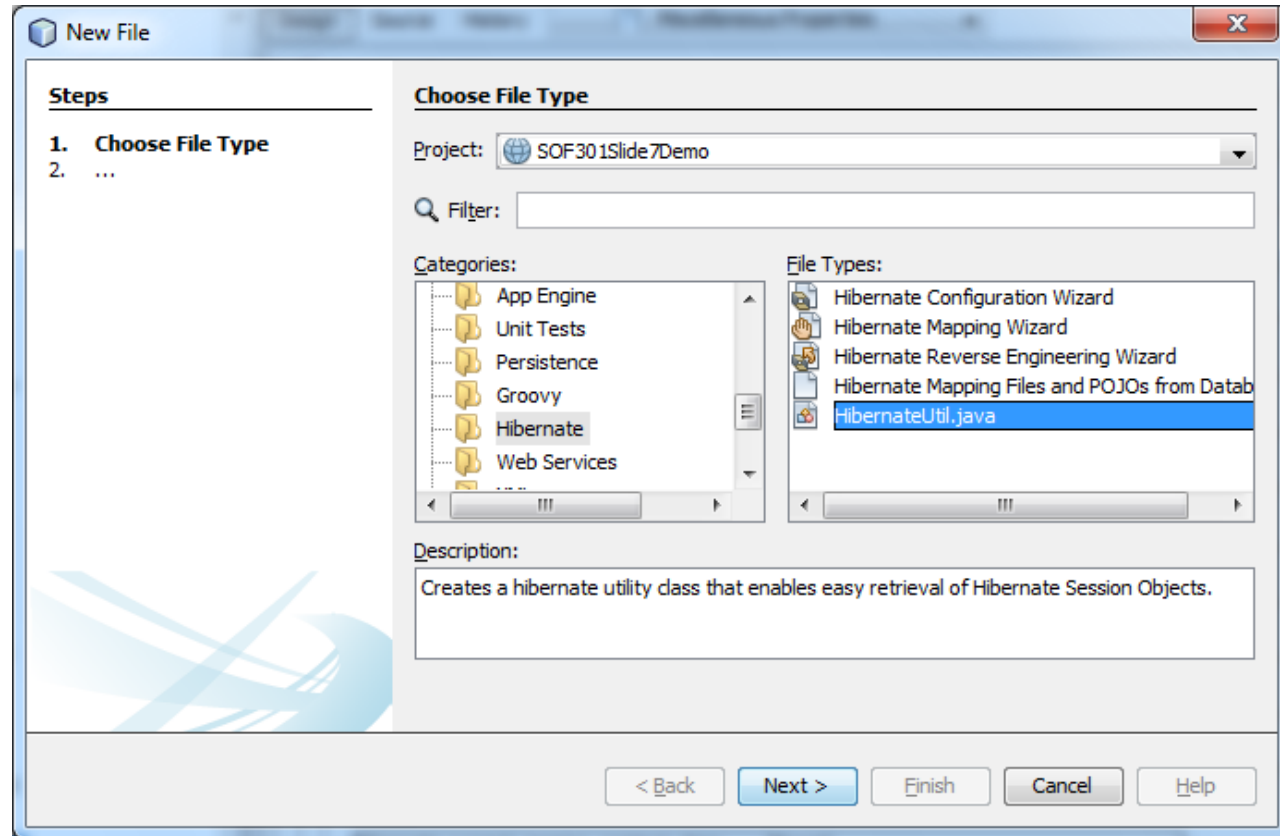
### Lưu ý: Không lưu được tiếng việt

→ Thêm đoạn sau vào phần cấu hình CSDL trong file config Hibernate

**?useUnicode=true&characterEncoding=UTF-8**

`jdbc:sqlserver://localhost:1433;databaseName=SOF301?useUnicode=true&characterEncoding=UTF-8`

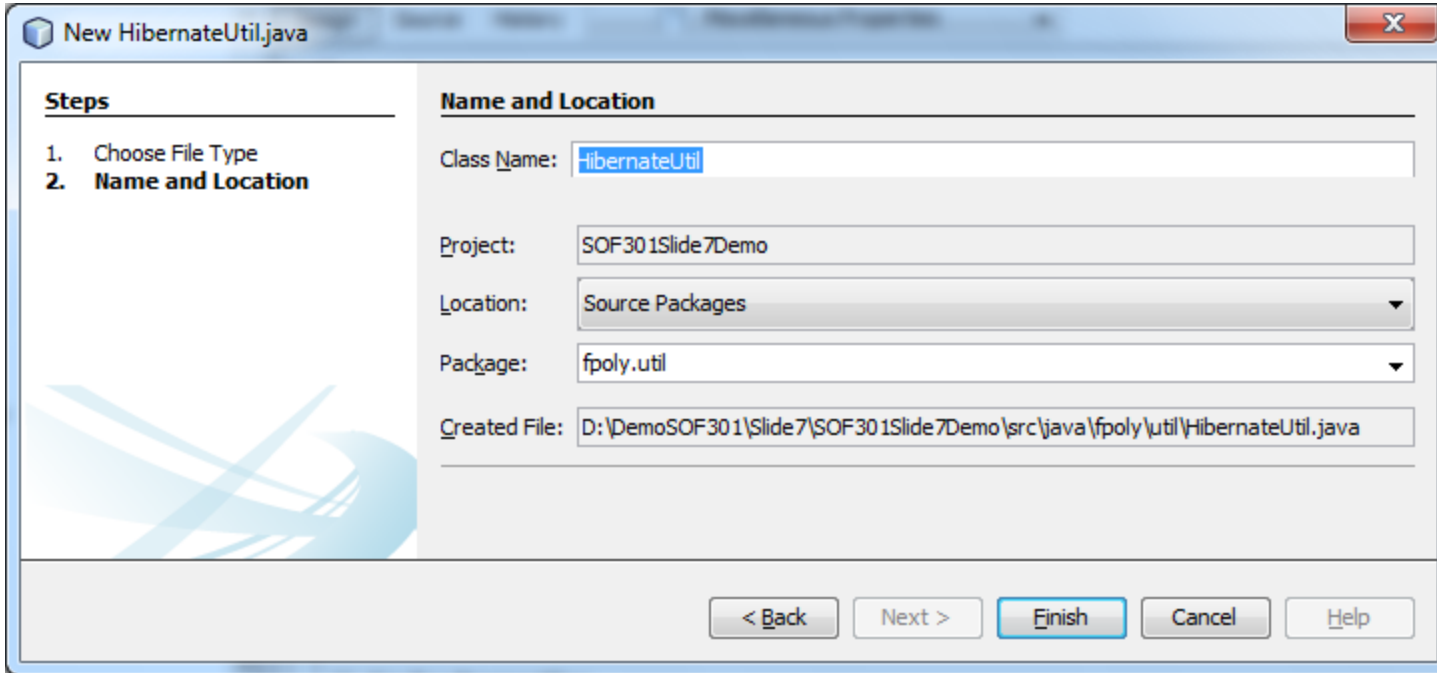
## ❑ 4, Tạo file HibernateUtil.java



Để sử dụng Hibernate, chúng ta cần tạo một helper class để access Session Factory cho việc lấy đối tượng Session. Class này gọi hàm configure() để load file hibernate.cfg.xml và build đối tượng SessionFactory.



## ❑ 4, Tạo file HiberntaeUtil.java



**New HibernateUtil.java**

**Steps**

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

**Name and Location**

Class Name: HibernateUtil

Project: SOF301Slide7Demo

Location: Source Packages

Package: fpoly.util

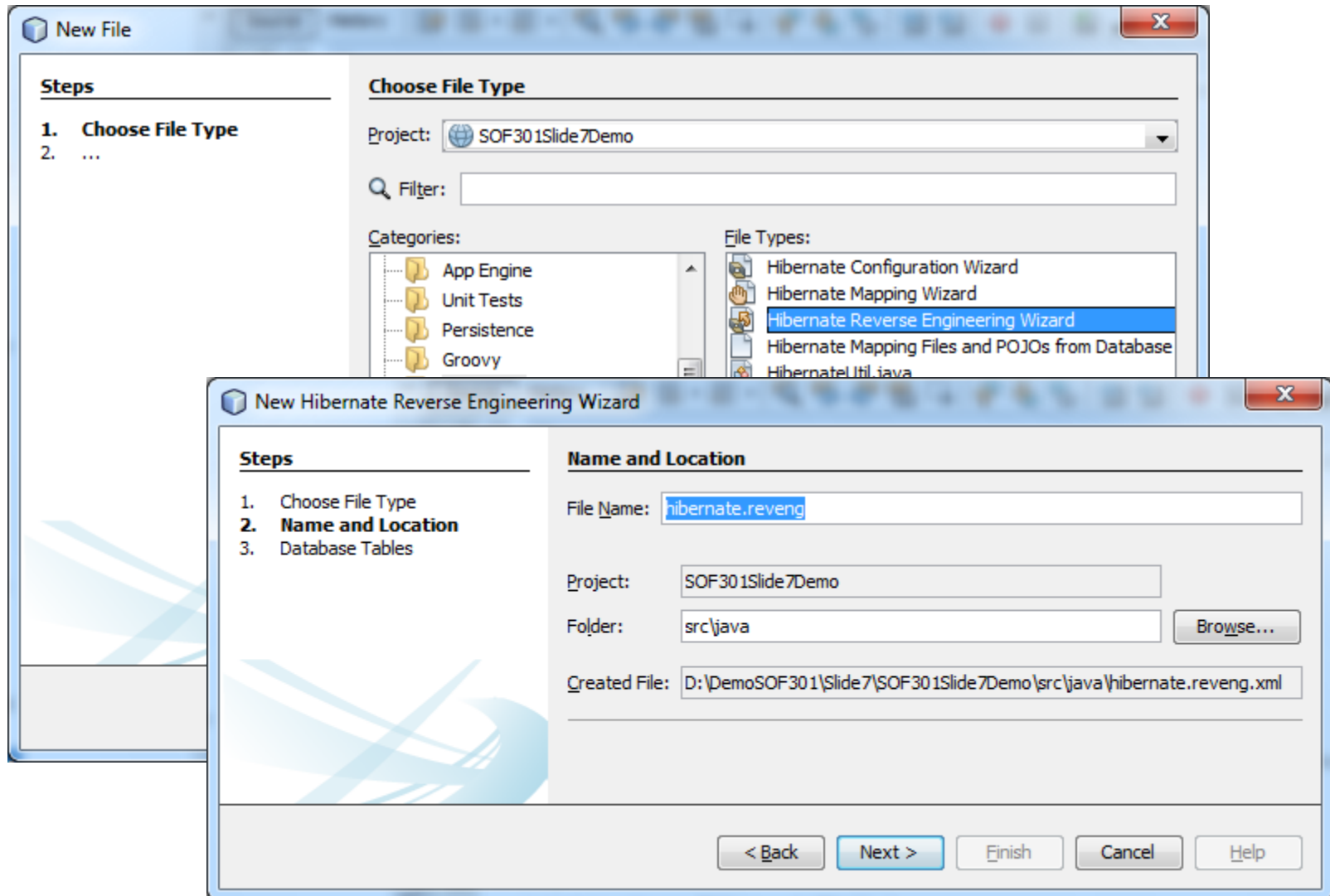
Created File: D:\DemoSOF301\Slide7\SOF301Slide7Demo\src\java\fpoly\util\HibernateUtil.java

< Back   Next >   **Finish**   Cancel   Help

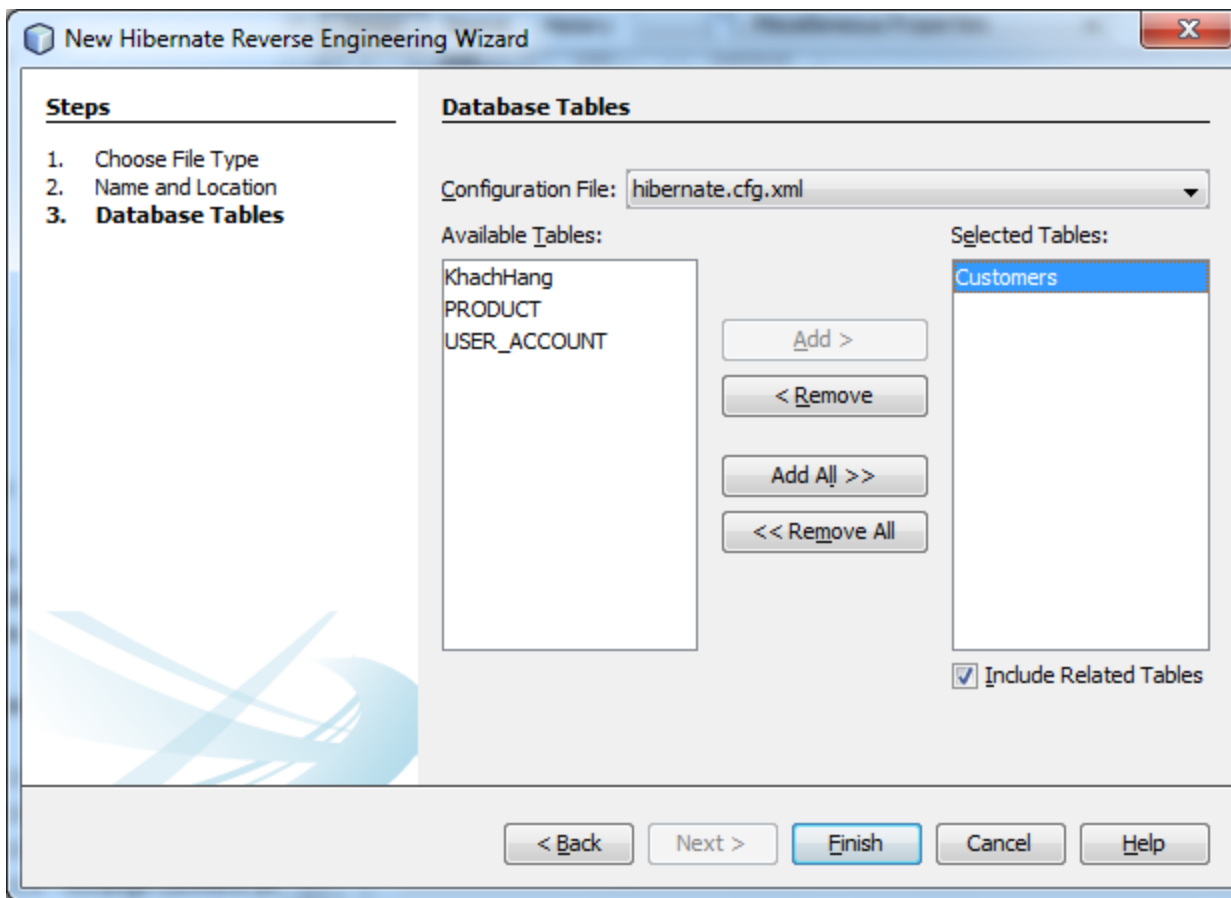
## ❑ 5, Tạo Hibernate Mapping File và POJO:

- ❖ Hibernate dùng POJO (Plain Old Java Object) để mô tả dữ liệu của table trong database. Mỗi thuộc tính của object tương ứng với một field trong table và chúng được mapping thông qua hibernate mapping file xml hoặc dùng annotation trong class. Trong bài này chúng ta sử dụng mapping file xml. POJO cũng tương tự như JavaBean, chứa các getter và setter.
- ❖ Để sử dụng Hibernate Mapping file và POJO, trước hết cần tạo file hibernate.reveng.xml (reverse engineering file ) để hỗ trợ chúng ta lựa chọn table cần dùng trong database dễ dàng hơn.

## 5.1, Tạo Hibernate Reverse Engineering file

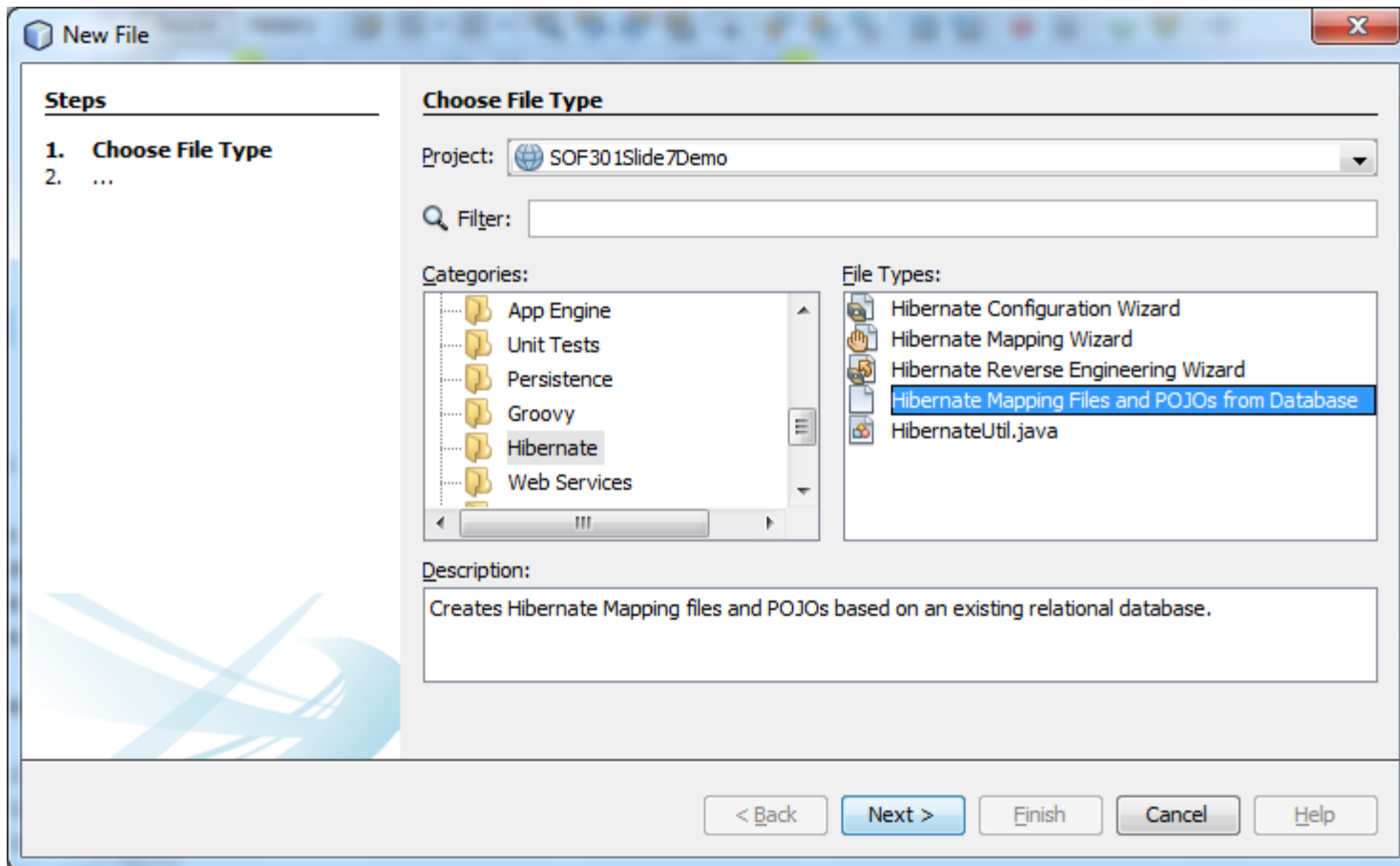


## 5.1, Tạo Hibernate Reverse Engineering file



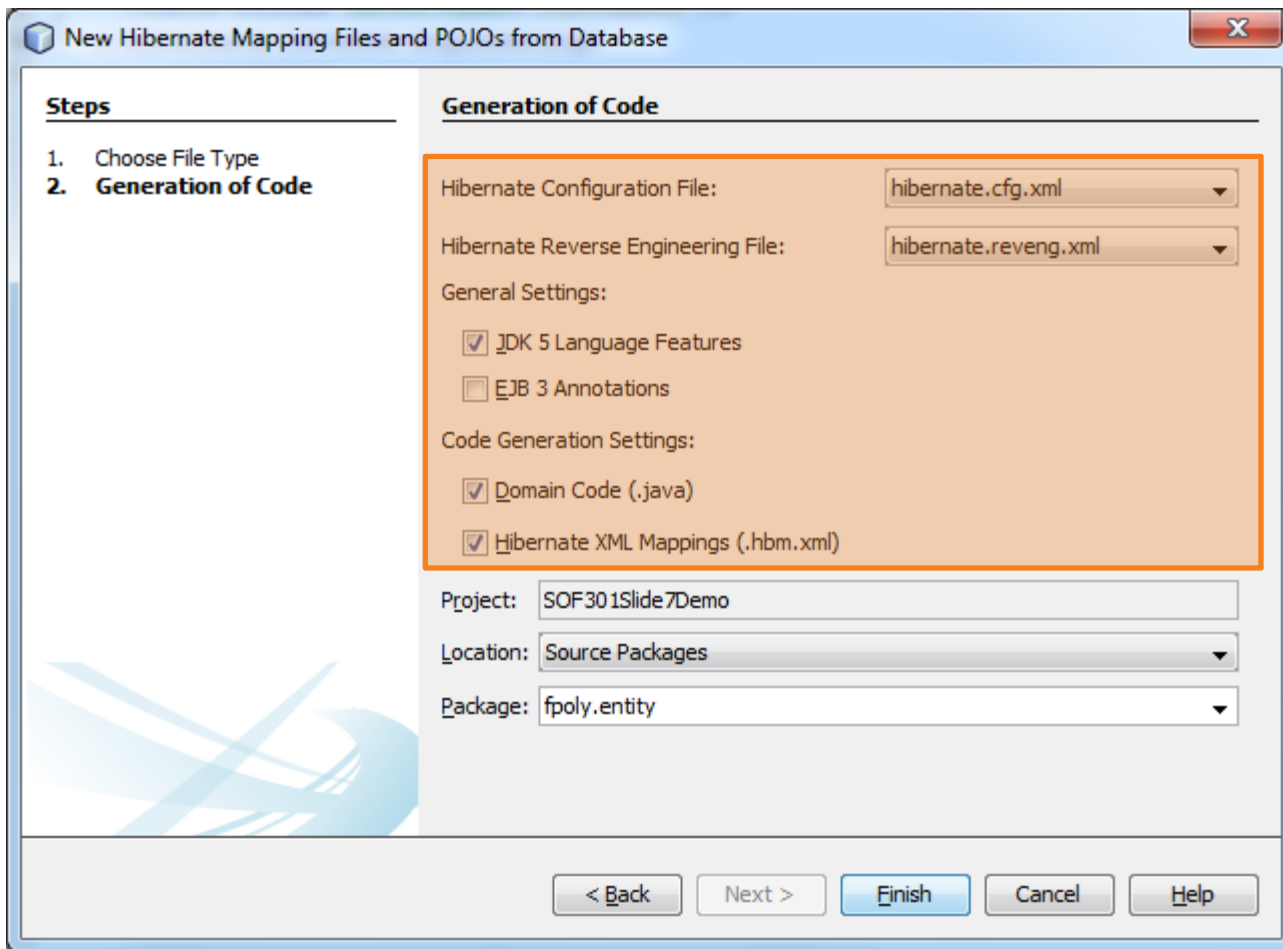
lựa chọn table cần dùng trong database

## 5.2, Tạo Hibernate Mapping File và POJO



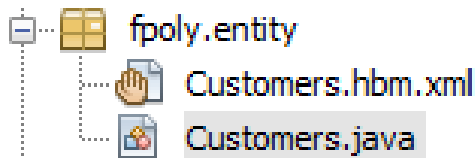
Chọn như hình

## 5.2, Tạo Hibernate Mapping File và POJO



Chọn như hình

## ❑ 5.2, Tạo Hibernate Mapping File và POJO



```

1  package fpoly.entity;
2  public class Customers implements java.io.Serializable {
3      private String maKhachHang;
4      private String matKhai;
5      private String hoVaTen;
6      private String email;
7      private String dienThoai;
8      public Customers() {...2 lines }
9
10     public Customers(String maKhachHang) {...3 lines }
11
12     public Customers(String maKhachHang, String matKhai, String hoVaTen, String email,
13
14     public String getMaKhachHang() {...3 lines }
15
16     public void setMaKhachHang(String maKhachHang) {...3 lines }
17
18     public String getMatKhai() {...3 lines }
19
20     public void setMatKhai(String matKhai) {...3 lines }
21
22     public String getHoVaTen() {...3 lines }
23
24     public void setHoVaTen(String hoVaTen) {...3 lines }
25
26     public String getEmail() {...3 lines }
27
28     public void setEmail(String email) {...3 lines }
29
30     public String getDienThoai() {...3 lines }
31
32     public void setDienThoai(String dienThoai) {...3 lines }
33 }
  
```

Kết quả hệ thống tạo ra class Customers.java và file mapping Customers.hbm.xml

## 5.2, Tạo Hibernate Mapping File và POJO



```
1 <?xml version="1.0"?>
2 <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
3 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
4 <!-- Generated Jul 4, 2017 12:08:42 AM by Hibernate Tools 4.3.1 -->
5 <hibernate-mapping>
6   <class name="fpoly.entity.Customers" table="Customers" schema="dbo" catalog="SOF301">
7     <id name="maKhachHang" type="string">
8       <column name="MaKhachHang" length="50" />
9       <generator class="assigned" />
10    </id>
11    <property name="matKhau" type="string">
12      <column name="MatKhau" length="50" />
13    </property>
14    <property name="hoVaTen" type="string">
15      <column name="HoVaTen" length="50" />
16    </property>
17    <property name="email" type="string">
18      <column name="Email" length="50" />
19    </property>
20    <property name="dienThoai" type="string">
21      <column name="DienThoai" length="50" />
22    </property>
23  </class>
24 </hibernate-mapping>
25
```

### Chú ý:

- Tên class, thuộc tính: phải đúng theo tên class, thuộc tính trong POJO, có phân biệt hoa thường.
- Tên table, column: đúng theo tên table, cột trong cơ sở dữ liệu, không phân biệt hoa thường.

Kết quả hệ thống tạo ra class Customers.java và file mapping Customers.hbm.xml



## ❑ Insert một record vào bảng Customer

```

index.jsp  ClassMain.java
Source  History
16  * @author Fpoly
17  */
18  public class ClassMain {
19      public static void main(String[] args) {
20          Customers kh = new Customers("KH06", "123", "Nguyen Van Teo",
21              "teonv@gmail.com", "0909999999");
22          SessionFactory factory = HibernateUtil.getSessionFactory();
23          Session session = factory.getCurrentSession();
24          try{
25              session.beginTransaction();
26              session.save(kh);
27              session.getTransaction().commit();
28          }catch(Exception e){
29              if(session.getTransaction().isActive()){
30                  session.getTransaction().rollback();
31              }
32              e.printStackTrace();
33          }
34      }
35  }
36

```

Trước khi tạo web application với mô hình MVC và xây dựng các DAO để sử dụng chúng ta thử insert 1 record vào table

	MaKhachHang	MatKhau	HoVaTen	Email	DienThoai
1	KH01	123	Nguyen Nghiem	nghiemn@fe.edu.vn	0913745789
2	KH02	abc	Tran Duy Phong	phongtd@fe.edu.vn	0933922487
3	KH03	123	Le Pham Tuan Kiet	kietlpt@fe.edu.vn	0903991033
4	KH04	abc	Le Van Phung	phunglv@fe.edu.vn	0903414749
5	KH05	bcd	Bui Minh Nhat	nhatbm@fe.edu.vn	0932030958
6	KH06	123	Nguyen Van Teo	teonv@gmail.com	0909999999



# DEMO

Chạy và giải thích





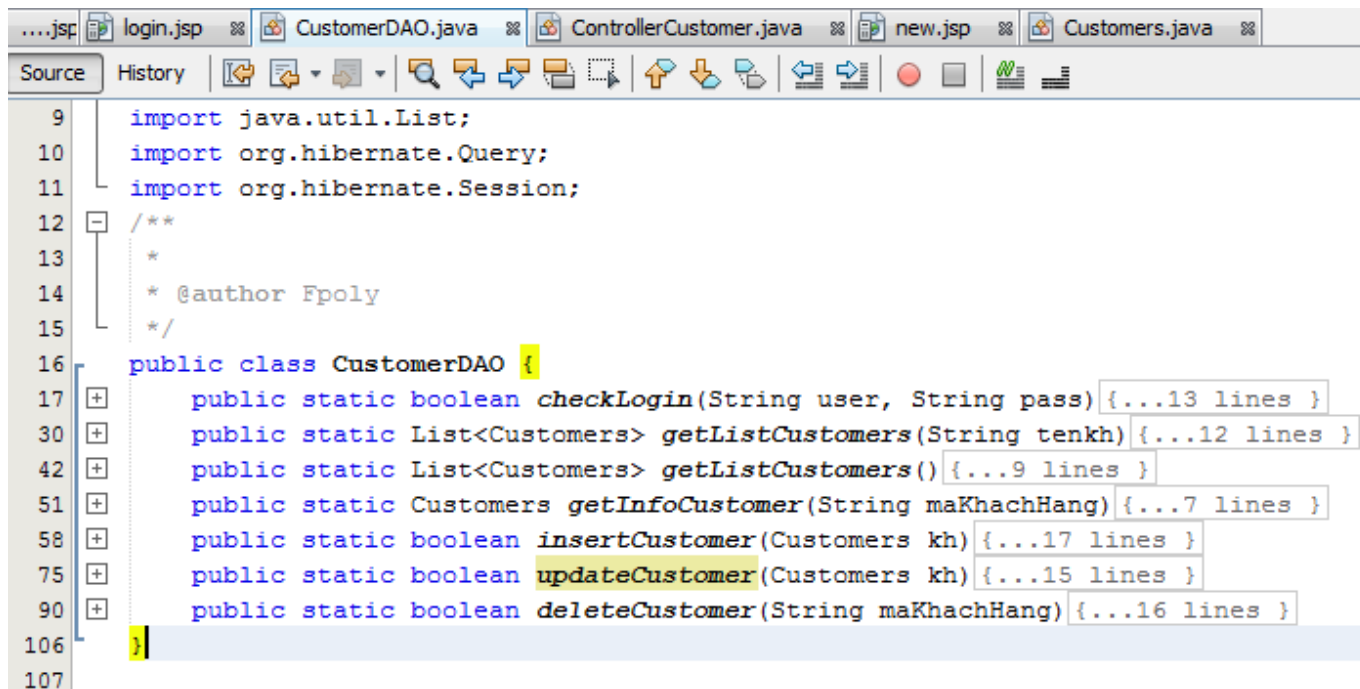
# LẬP TRÌNH JAVA 4

## BÀI 7: HIBERNATE

### PHẦN 2

## ❑ Model

- ❑ Tạo class DAO(Data Access Object) gồm các chức năng: checkLogin(), getListDustomers(), getInfoCustomer(), insertCustomer(), updateCustomer(), deleteCustomer().



```
9      import java.util.List;
10     import org.hibernate.Query;
11     import org.hibernate.Session;
12
13     /**
14      * @author Fpoly
15      */
16     public class CustomerDAO {
17         public static boolean checkLogin(String user, String pass) {...13 lines }
30         public static List<Customers> getListCustomers(String tenkh) {...12 lines }
42         public static List<Customers> getListCustomers() {...9 lines }
51         public static Customers getInfoCustomer(String maKhachHang) {...7 lines }
58         public static boolean insertCustomer(Customers kh) {...17 lines }
75         public static boolean updateCustomer(Customers kh) {...15 lines }
90         public static boolean deleteCustomer(String maKhachHang) {...16 lines }
106     }
107
```

## ❑ Model

- ❑ checkLogin(): dùng để kiểm tra thông tin đăng nhập, nếu user và pass tồn tại hàm trả về true, ngược lại trả về false

```
public static boolean checkLogin(String user, String pass){
    List<Customers> list = null;
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    String sql="from Customers where maKhachHang = '"+user+"' and "
        + "matKhau = '"+pass+"'";
    Query query = session.createQuery(sql);
    list = query.list();
    if(list.size()>0){
        return true;
    }
    return false;
}
```

### Chú ý:

- Sử dụng `getCurrentSession()` → không cần phải `open session` và `close session` sau khi sử dụng xong. Tuy nhiên phải khai báo trong file config:

`<property name="hibernate.current_session_context_class">thread</property>`

- Sử dụng `HibernateUtil.getSessionFactory().openSession();`  
→ phải close session sau khi sử dụng xong. `Session.Close();`

## ❑ Model

- ❑ getListCustomer(String tenkh): hàm trả về danh sách Customers dựa vào tham số truyền vào.

```
public static List<Customers> getListCustomers(String tenkh){  
    List<Customers> list = null;  
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();  
    session.beginTransaction();  
    String sql="from Customers";  
    if(tenkh.length()>0){  
        sql += " where hoVaTen like '%" +tenkh+"%'";  
    }  
    Query query = session.createQuery(sql);  
    list = query.list();  
    return list;  
}
```

## ❑ Model

- ❑ getInfoCustomer(String maKhachHang): hàm trả về thông tin Customers theo maKhachHang

```
public static Customers getInfoCustomer(String maKhachHang){  
    Session session = HibernateUtil.getSessionFactory().openSession();  
    session.beginTransaction();  
    Customers kh = (Customers)session.get(Customers.class, maKhachHang);  
    session.close();  
    return kh;  
}
```

Sử dụng `HibernateUtil.getSessionFactory().openSession();`  
→ close session sau khi sử dụng xong. `Session.Close();`

- ❑ insertCustomer(Customers kh): thêm mới Customers

```
public static boolean insertCustomer(Customers kh){  
    if(CustomerDAO.getInfoCustomer(kh.getMaKhachHang()) != null)  
        return false;  
    Session session = HibernateUtil.getSessionFactory().openSession();  
    try{  
        session.beginTransaction();  
        session.save(kh);  
        session.getTransaction().commit();  
        return true;  
    }catch(Exception e){  
        session.getTransaction().rollback();  
        System.out.println(e);  
        return false;  
    }finally{  
        session.close();  
    }  
}
```

## ❑ Model

- ❑ updateCustomer(Customers kh): update thông tin Customers

```
public static boolean updateCustomer(Customers kh){  
    if(CustomerDAO.getInfoCustomer(kh.getMaKhachHang())==null)  
        return false;  
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();  
    try{  
        session.getTransaction().begin();  
        session.update(kh);  
        session.getTransaction().commit();  
        return true;  
    }catch(Exception e){  
        session.getTransaction().rollback();  
        System.out.println(e);  
        return false;  
    }  
}
```



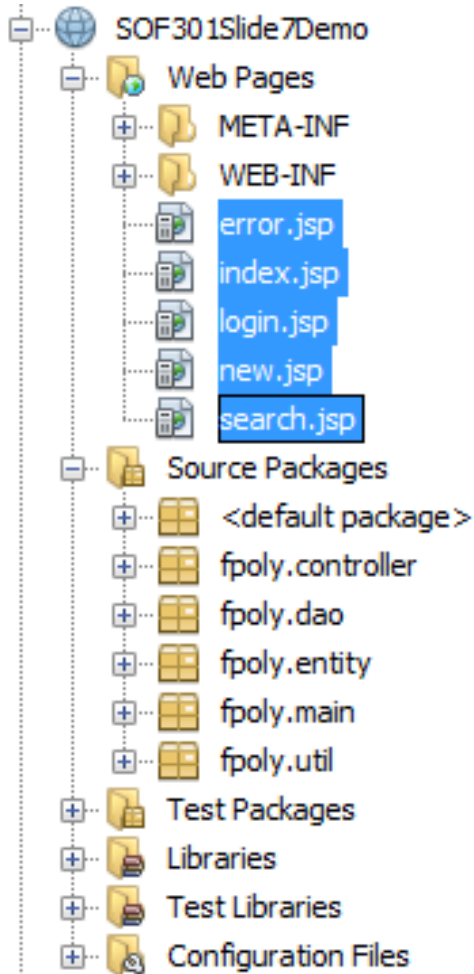
## ❑ Model

- ❑ deleteCustomer(String maKhachHang): xóa Customer theo maKhachHang

```
public static boolean deleteCustomer(String maKhachHang){
    Customers kh = CustomerDAO.getInfoCustomer(maKhachHang);
    if(kh==null)
        return false;
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    try{
        session.getTransaction().begin();
        session.delete(kh);
        session.getTransaction().commit();
        return true;
    }catch(Exception e){
        session.getTransaction().rollback();
        System.out.println(e);
        return false;
    }
}
```

## □ View

□ Tạo các view sau: login, new, error, search



## View

### Login

```
<body>
  <h1>Login</h1>
  <form action="ControllerCustomer">
    Username: <input type="text" name="txtUser" value=""/><br/>
    Password: <input type="password" name="txtPass" value=""/><br/><br/>
    <input type="submit" name="action" value="Login"/>
    <input type="reset" name="action" value="Reset"/>
    <input type="submit" name="action" value="New"/>
  </form>
</body>
```

### New

```
<h1>New Customer</h1>
<form action="ControllerCustomer">
  Ma KH: <input type="text" name="txtMaKH" value=""/><br/>
  Mat khau: <input type="password" name="txtMatKhau" value=""/><br/>
  Ho ten: <input type="text" name="txtHoten" value=""/><br/>
  Email: <input type="text" name="txtEmail" value=""/><br/>
  So DT: <input type="text" name="txtSoDT" value=""/><br/>
  <input type="submit" name="action" value="Insert"/>
</form>
```

### Error

```
<body>
  <h1>Username or password invalid!!!</h1>
</body>
```

## ❑ View

## ❑ search

```
Welcome ${sessionScope.USER}
<h1>Search</h1>
<form action="ControllerCustomer">
    Ten KH: <input type="text" name="txtTenKH" value=""/>
    <input type="submit" name="action" value="Search"/>
</form><br/>
<table border="1">
    <tr>
        <td>MaKH</td><td>HoTen</td><td>MatKhou</td><td>Email</td>
        <td>SoDT</td>
        <td>Delete</td>
    </tr>
    <c:forEach var="rows" items="${listKH}">
        <form action="ControllerCustomer">
            <tr>
                <td>${rows.maKhachHang}</td>
                <td>${rows.hoVaTen}</td>
                <td>${rows.matKhou}</td>
                <td>${rows.email}</td>
                <td>${rows.dienThoai}</td>
                <td>
                    <input type="hidden" name="txtMaKH" value="${rows.maKhachHang}"/>
                    <input type="submit" name="action" value="Delete"/>
                </td>
            </tr>
        </form>
    </c:forEach>
</table>
```

## ❑ Controller

```

public class ControllerCustomer extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        try{
            String action = request.getParameter("action");
            if(action.equals("Login")){
                //.....
            }else if(action.equals("New")){
                //.....
            }else if(action.equals("Insert")){
                //.....
            }else if(action.equals("Search")){
                //.....
            }else if(action.equals("Delete")){
                //.....
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}

```

## ❑ Controller

```
38     PrintWriter out = response.getWriter();
39     try{
40         String action = request.getParameter("action");
41         if(action.equals("Login")){
42             String user = request.getParameter("txtUser");
43             String pass = request.getParameter("txtPass");
44             CustomerDAO cus = new CustomerDAO();
45             boolean check = cus.checkLogin(user, pass);
46             String url = "error.jsp";
47             if(check==true){
48                 HttpSession session = request.getSession(true);
49                 session.setAttribute("USER", user);
50                 url="search.jsp";
51             }
52             RequestDispatcher rd = request.getRequestDispatcher(url);
53             rd.forward(request, response);
54         }else if(action.equals("New")){
55             RequestDispatcher rd = request.getRequestDispatcher("new.jsp");
56             rd.forward(request, response);
```

## ❑ Controller

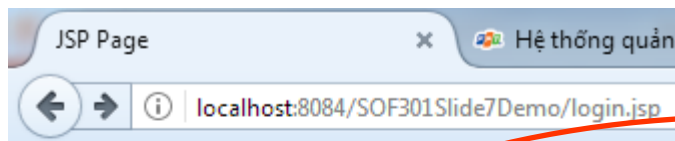
```
57         }else if(action.equals("Insert")){
58             String makh = request.getParameter("txtMaKH");
59             String matkhou = request.getParameter("txtMatKhau");
60             String hoten = request.getParameter("txtHoten");
61             String email = request.getParameter("txtEmail");
62             String sodt = request.getParameter("txtSoDT");
63             Customers newkh = new Customers(matkhou, matkhou, hoten, email, email);
64             CustomerDAO.insertCustomer(newkh);
65             RequestDispatcher rd = request.getRequestDispatcher("new.jsp");
66             rd.forward(request, response);
67         }
68         else if(action.equals("Search")){
69             String tenkh = request.getParameter("txtTenKH");
70             List<Customers> list = CustomerDAO.getListCustomers(tenkh);
71             request.setAttribute("listKH", list);
72             String url="search.jsp";
73             RequestDispatcher rd = request.getRequestDispatcher(url);
74             rd.forward(request, response);
```

## ❑ Controller

```
75         }else if(action.equals("Delete")){
76             String makh = request.getParameter("txtMaKH");
77             boolean daxoa = CustomerDAO.deleteCustomer(makh);
78             if(daxoa){
79                 String url="ControllerCustomer?txtTenKH=&action=Search";
80                 RequestDispatcher rd = request.getRequestDispatcher(url);
81                 rd.forward(request, response);
82             }
83         }
84     }catch(Exception e){
85         e.printStackTrace();
86     }
87 }
```



## Demo

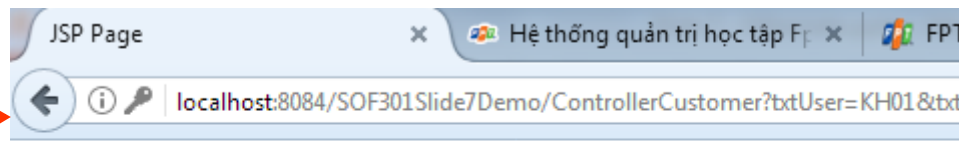


### Login

Username: KH01  
Password: ●●●

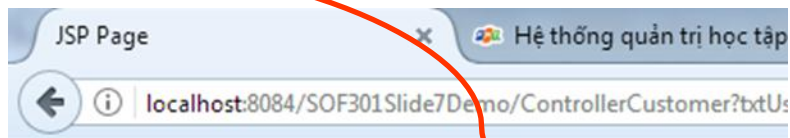
Login Reset New

Login fail



Username or password invalid!!!

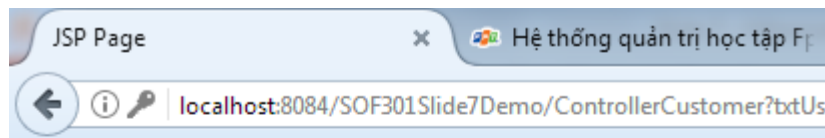
New Customer



### New Customer

Ma KH: KH07  
Mat khau: ●●●  
Ho ten: Tony Teo  
Email: teotn@gmail.com  
So DT: 0123456789

Insert



Welcome KH01

### Search

Ten KH:  Search

MaKH	HoTen	MatKhau	Email	SoDT	Delete
------	-------	---------	-------	------	--------

Login okay

## ❑ Kết quả “Search” không tham số

JSP Page x Hệ thống quản trị học tập F x FPT-Polytechnic

localhost:8084/SOF301Slide7Demo/ControllerCustomer?txtTenKH=&action=Search

Welcome KH01

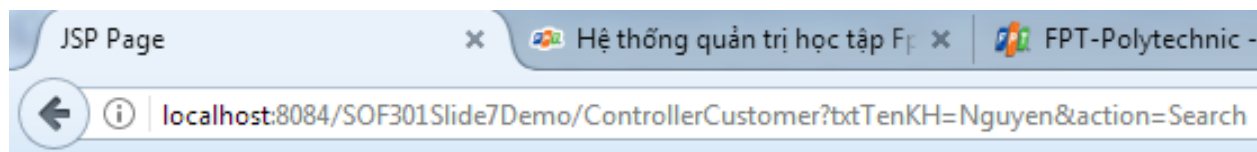
### Search

Ten KH:

MaKH	HoTen	MatKhau	Email	SoDT	Delete
KH01	Nguyen Nghiem	123	nghiemn@fe.edu.vn	0913745789	<input type="button" value="Delete"/>
KH02	Tran Duy Phong	abc	phongtd@fe.edu.vn	0933922487	<input type="button" value="Delete"/>
KH03	Le Pham Tuan Kiet	123	kietlpt@fe.edu.vn	0903991033	<input type="button" value="Delete"/>
KH04	Le Van Phung	abc	phunglv@fe.edu.vn	0903414749	<input type="button" value="Delete"/>
KH05	Bui Minh Nhat	bcd	nhatbm@fe.edu.vn	0932030958	<input type="button" value="Delete"/>
KH06	Nguyen Van Teo	123	teonv@gmail.com	0909999999	<input type="button" value="Delete"/>

Kết quả Search không tham số, có thể Delete các Customer trên kết quả tìm được

## ❑ Search theo tên



Welcome KH01

## Search

Ten KH:

MaKH	HoTen	MatKhau	Email	SoDT	Delete
KH01	Nguyen Nghiem	123	nghiemn@fe.edu.vn	0913745789	<input type="button" value="Delete"/>
KH06	Nguyen Van Teo	123	teonv@gmail.com	0909999999	<input type="button" value="Delete"/>

Kết quả Search theo tên, có thể Delete các Customer trên kết quả tìm được



# DEMO

Chạy và giải thích



- ❑ Hibernate được thiết kế để hoạt động với nhiều môi trường khác nhau → vì thế nó cần được cấu hình khi sử dụng.

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.microsoft.sqlserver.jdbc.SQLServerDriver
    </property>
    <property name="hibernate.connection.url">jdbc:sqlserver://localhost:1433;databaseName=SOF301
    </property>
    <property name="hibernate.connection.username">sa</property>
    <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>
    <property name="hibernate.show_sql">>true</property>
    <property name="hibernate.current_session_context_class">thread</property>
    <property name="hibernate.query.factory_class">
      org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory</property>
    <mapping resource="fpoly/entity/Customers.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

- ❑ **hibernate.dialect** : loại cơ sở dữ liệu được sử dụng.
- ❑ **hibernate.connection.driver\_class**: driver được sử dụng.
- ❑ **hibernate.connection.url**: url cơ sở dữ liệu.
- ❑ Sử dụng  
    ?useUnicode=true&characterEncoding=UTF-8 để có thể lưu dữ liệu unicode xuống cơ sở dữ liệu.
- ❑ **hibernate.connection.username**: username.
- ❑ **hibernate.connection.password**: password.
- ❑ **hibernate.connection.pool\_size**: số lượng kết nối tối đa tới CSDL tại một thời điểm.

❑ **hibernate.dialect** : loại cơ sở dữ liệu được sử dụng.

Ví dụ: `org.hibernate.dialect.MySQLDialect` ⇔ MySQL

Tên cơ sở dữ liệu	Tên property
DB2	<code>org.hibernate.dialect.DB2Dialect</code>
HypersonicSQL	<code>org.hibernate.dialect.HSQLDialect</code>
Infomix	<code>org.hibernate.dialect.InformixDialect</code>
Ingres	<code>org.hibernate.dialect.IngresDialect</code>
Interbase	<code>org.hibernate.dialect.InterbaseDialect</code>
MySQL	<code>org.hibernate.dialect.MySQLDialect</code>
Oracle (any version)	<code>org.hibernate.dialect.OracleDialect</code>
Oracle 9	<code>org.hibernate.dialect.Oracle9Dialect</code>

❑ Ngoài các cấu hình cơ bản, việc cấu hình hibernate còn rất nhiều tùy chọn khác để điều khiển cách thức hoạt động của hibernate. Thông thường các thuộc tính này nếu không được khai báo sẽ có giá trị mặc định.

- Các thuộc tính về Configuration
- JDBC và các thuộc tính về Connection.
- Các thuộc tính về Transaction
- Các thuộc tính khác.



# CÁC CẤU HÌNH TÙY CHỌN KHÁC

## ☐ Các thuộc tính về Configuration.

Tên thuộc tính	Ý nghĩa	Giá trị
hibernate.show_sql	In tất cả các câu truy vấn SQL đã dùng ra console (phục vụ debug)	True, False
hibernate.format_sql	In các câu SQL đã dùng ra file log và console (phục vụ debug)	True, False
hibernate.default_schema	Tên table mặc định, nếu trong câu SQL không chỉ rõ table đang dùng.	Tên CSDL. VD: table HocSinh
hibernate.default_catalog	Tên cơ sở dữ liệu mặc định, nếu trong câu SQL không chỉ rõ cơ sở dữ liệu đang dùng.	Tên CSDL. VD: quanlyhocsinh
hibernate.max_fetch_depth	Độ sâu join fetch (tự động join các bảng trong quá trình mapping)	Chỉ nên từ 0 - 3
hibernate.order_updates	Thay đổi mặc định hibernate sẽ update khóa chính của một đối tượng trước, sau đó mới tới các cột còn lại.	True, false
hibernate.generate_statistics		True, false

## ☐ Các thuộc tính về Transaction.

Tên thuộc tính	Ý nghĩa	Giá trị
hibernate.transaction.flush _before_completion	Nếu là true thì session sẽ tự động flush trước khi hoàn thành.	True, False
hibernate.transaction.auto _close_session	Tự động đóng session sau khi hoàn thành.	True, False

# CÁC CẤU HÌNH TÙY CHỌN KHÁC

## ☐ Các thuộc tính khác.

Tên thuộc tính	Ý nghĩa	Giá trị
hibernate.current_session_context_class	Các tùy chỉnh cho session hiện tại	Jta, thread, managed, custom.Class
hibernate.query.factory_class	Chọn phương pháp phân tích cú pháp HQL, mặc định là true.	ASTQueryTranslatorFactory ClassicQueryTranslatorFactory

- ❖ Giới thiệu về Hibernate
- ❖ So sánh Hibernate và JDBC
- ❖ Các bước tạo ứng dụng Hibernate
- ❖ Ví dụ xây dựng ứng dụng MVC – Hibernate
- ❖ Các config cơ bản file mapping.hbm.xml
- ❖ Các cấu hình tùy chọn khác





**Cảm ơn**