

MỤC TIÊU

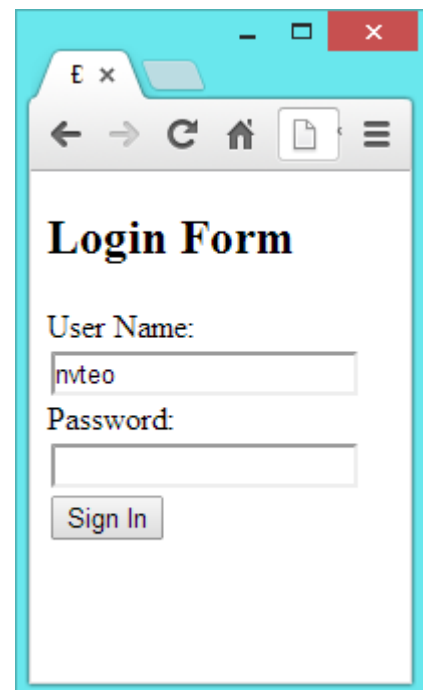
Kết thúc bài thực hành này, bạn có khả năng

- ✓ Xây dựng một trang web cho phép khách web đăng ký thành viên trực tuyến, sau đó sử dụng tài khoản đã đăng ký để đăng nhập sử dụng JdbcTemplate.
- ✓ Biết cách thông báo lỗi trùng mã khi đăng ký cũng như sai thông tin khi đăng nhập sai

MÔ TẢ



Đăng ký trùng mã



Đăng ký thành công



Đăng nhập sai



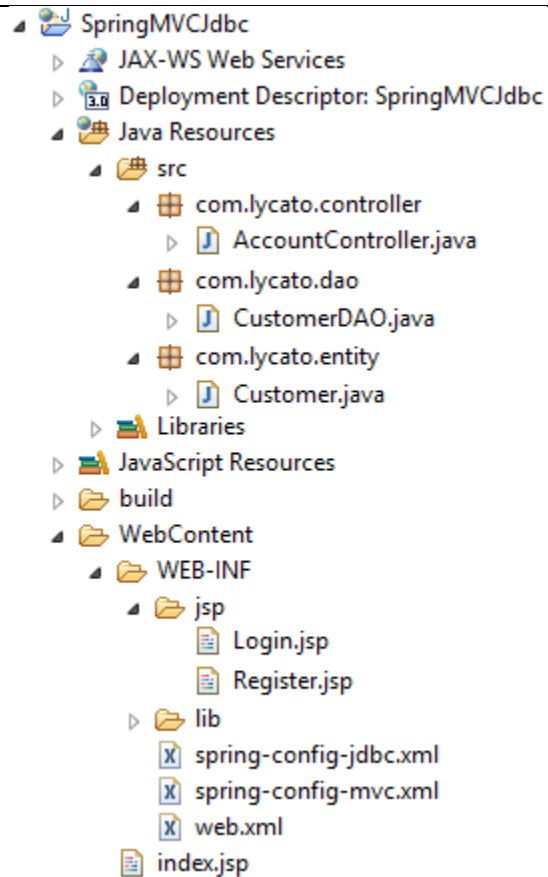
Đăng nhập đúng

Hoạt động:

- ✓ Chạy register.htm hiển thị form đăng ký
- ✓ Nhấp [Register] sẽ thực hiện thêm mới 1 thành viên vào CSDL. Và
 - Thông báo lỗi nếu đăng ký trùng mã
 - Chuyển sang trang đăng nhập login.htm nếu đăng nhập thành công
- ✓ Nhấp [Sign In] sẽ thực hiện xác thực thông tin tài khoản. Thông báo lỗi hoặc thành công nếu tùy vào kết quả xác thực.

THỰC HIỆN

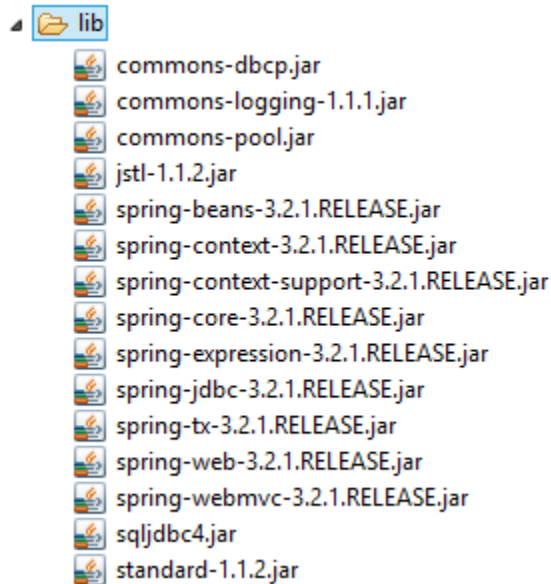
Trong bài này, bạn sẽ phải tạo một project có tổ chức như sau:



- ✓ Bước 1: Thư viện và cấu hình project
- ✓ Bước 2: CSDL
- ✓ Bước 4: Tạo lớp DAO và Entity
- ✓ Bước 5: Tạo giao diện
- ✓ Bước 6: Tạo Controller
- ✓ Bước 7: Chạy

Bước 1: Thư viện và cấu hình project

Thư viện



Bên cạnh các thư viện của Thư viện cần thiết cho ứng dụng

- ✓ SQLServerDriver
 - sqljdbc4.jar
- ✓ JdbcTemplate
 - commons-dbc.jar
 - spring-jdbc-3.2.1.RELEASE.jar
 - spring-tx-3.2.1.RELEASE.jar

Cấu hình

❖ Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
    <display-name>SpringMVCEmail</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring-config-*.xml</param-value>
```



LAB5: SECURITY VỚI JdbcTemplate

```
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>
</web-app>
```

Cấu hình để Spring MVC nạp nhiều file cấu hình: spring-config-*.xml. Dấu * sẽ đại diện cho nhóm ký tự bất kỳ. Cụ thể ở bài này là mvc, gmail và upload

❖ spring-config-mvc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

    <!-- Declare a view resolver -->
    <bean id="viewResolver" p:prefix="/WEB-INF/jsp/" p:suffix=".jsp"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"/>

    <!-- Spring MVC Annotation -->
    <mvc:annotation-driven />
    <context:annotation-config />

    <!-- Where to find component -->
    <context:component-scan base-package="com.lycato" />
</beans>
```

- ✓ Khai báo bean InternalResourceViewResolver để xử lý view
- ✓ Chỉ rõ package tìm kiếm các component là com.lycato
- ✓ Chỉ rõ ứng dụng Spring này được phép sử dụng annotation

❖ spring-config-jdbc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
```



LAB5: SECURITY VỚI JdbcTemplate

```
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

<bean id="dataSource" destroy-method="close"
      class="org.apache.commons.dbcp.BasicDataSource"
      p:driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
      p:url="jdbc:sqlserver://localhost:1433;DatabaseName=Seminar"
      p:username="sa"
      p:password="songlong"/>

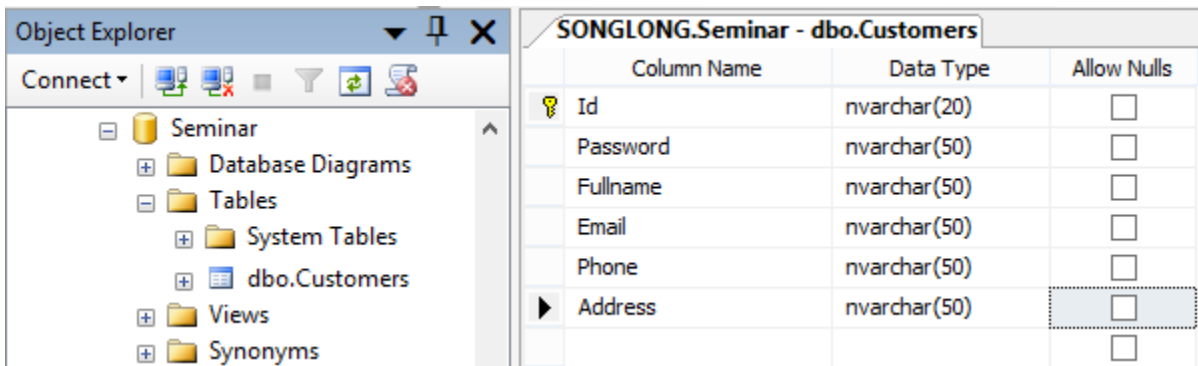
<bean id="jdbcTemplate"
      class="org.springframework.jdbc.core.JdbcTemplate">
  <constructor-arg ref="dataSource"/>
</bean>

</beans>
```

Trong file cấu hình này bạn phải khai báo 2 bean.

- ✓ BasicDataSource: bean này cấu hình các thông số kết nối CSDL
- ✓ JdbcTemplate: bean này được khai báo để làm việc với CSDL được tiêm vào và sử dụng sau này trong ứng dụng

Bước 2: CSDL



Column Name	Data Type	Allow Nulls
Id	nvarchar(20)	<input type="checkbox"/>
Password	nvarchar(50)	<input type="checkbox"/>
Fullname	nvarchar(50)	<input type="checkbox"/>
Email	nvarchar(50)	<input type="checkbox"/>
Phone	nvarchar(50)	<input type="checkbox"/>
Address	nvarchar(50)	<input type="checkbox"/>

Hình: CSDL Seminar chứa bảng Customers

Bước 4: Tạo lớp mô tả và truy xuất dữ liệu

Lớp mô tả dữ liệu (Entity)

Lớp này mô tả cấu trúc bảng. Mục đích là để chứa một bản ghi dữ liệu thao tác với CSDL. Nó cũng được kết nối với các trường form để hiển thị dữ liệu đọc được cho người dùng xem.

```
package com.lycato.entity;

public class Customer {
    String id, fullname, password, email, phone, address;
```

```
public String getId() {  
    return id;  
}  
public void setId(String id) {  
    this.id = id;  
}  
  
public String getFullname() {  
    return fullname;  
}  
public void setFullname(String fullname) {  
    this.fullname = fullname;  
}  
  
public String getPassword() {  
    return password;  
}  
public void setPassword(String password) {  
    this.password = password;  
}  
  
public String getEmail() {  
    return email;  
}  
public void setEmail(String email) {  
    this.email = email;  
}  
  
public String getPhone() {  
    return phone;  
}  
public void setPhone(String phone) {  
    this.phone = phone;  
}  
  
public String getAddress() {  
    return address;  
}  
public void setAddress(String address) {  
    this.address = address;  
}  
}
```

Lớp truy xuất dữ liệu (DAO)

Lớp này chứa các phương thức thao tác dữ liệu (thêm, sửa, xóa) và truy vấn dữ liệu.

- ✓ Insert(): thêm
- ✓ Update(): sửa
- ✓ Delete(): xóa
- ✓ getXyz(): truy vấn

Lớp nào được chú thích bởi @Repository để có thể tiêm vào AccountController trong ứng dụng bởi @Autowire để sử dụng sau này.

```
package com.lycato.dao;
```



LAB5: SECURITY VỚI JdbcTemplate

```
import java.io.Serializable;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;

import com.lycato.entity.Customer;

@Repository
public class CustomerDAO{
    /**
     * Inject từ <bean class="...JdbcTemplate>
     */
    @Autowired
    protected JdbcTemplate jdbc;

    /**
     * Thêm mới 1 thực thể
     * @param entity là thực thể mới
     */
    public void insert(Customer entity) {
        String sql = "INSERT INTO Customers (Id, Fullname, Password, Email, Phone, Address) VALUES (?, ?, ?, ?, ?, ?)";
        jdbc.update(sql, entity.getId(), entity.getFullname(),
            entity.getPassword(), entity.getEmail(), entity.getPhone(), entity.getAddress());
    }

    /**
     * Cập nhật thực thể
     * @param entity là thực thể cần cập nhật
     */
    public void update(Customer entity) {
        String sql = "UPDATE Customers SET Fullname=?, Password=?, Email=?, Phone=?, Address=? WHERE Id=?";
        jdbc.update(sql, entity.getFullname(), entity.getPassword(),
            entity.getEmail(), entity.getPhone(), entity.getAddress(), entity.getId());
    }

    /**
     * Xóa thực thể theo mã
     * @param id mã thực thể cần xóa
     */
    public void delete(Serializable id) {
        String sql = "DELETE FROM Customers WHERE Id=?";
        jdbc.update(sql, id);
    }

    /**
     * Truy vấn 1 thực thể theo mã
     * @param id mã thực thể cần truy vấn
     * @return thực thể truy vấn được
     */
    public Customer getById(Serializable id) {
        String sql = "SELECT * FROM Customers WHERE Id=?";
        return jdbc.queryForObject(sql, getRowMapper(), id);
    }
}
```



```
/**
 * Truy vấn tất cả các thực thể
 * @return danh sách thực thể truy vấn được
 */
public List<Customer> getAll() {
    String sql = "SELECT * FROM Customers";
    return getBySql(sql);
}

/**
 * Truy vấn các thực thể theo câu lệnh sql
 * @param sql câu lệnh truy vấn
 * @return danh sách thực thể truy vấn được
 */
protected List<Customer> getBySql(String sql) {
    return jdbc.query(sql, getRowMapper());
}

/**
 * Ánh xạ cấu trúc bản ghi theo thuộc tính của bean
 * @return ánh xạ bản ghi theo thuộc tính bean
 */
private RowMapper<Customer> getRowMapper() {
    return new BeanPropertyRowMapper<Customer>(Customer.class);
}
}
```

Bước 5: Tạo giao diện

Giao diện cho bài này gồm 1 form đăng ký và một form đăng nhập.

register.htm

```
<%@page pageEncoding="utf-8" contentType="text/html; charset=utf-8" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Đăng ký</title>
</head>
<body>
    <h2>Registration Form</h2>
    <h4 style="color:red">${errors}</h4>
    <form:form action="register.htm"
        method="post" modelAttribute="user">

        <div>User Name:</div>
        <form:input path="id"/>

        <div>Password:</div>
        <form:password path="password"/>

        <div>Full Name:</div>
        <form:input path="fullname"/>

        <div>Email:</div>
        <form:input path="email"/>
    </form>
</body>
</html>
```



LAB5: SECURITY VỚI JdbcTemplate

```
<div>Phone Number:</div>
<form:input path="phone"/>

<div>Address:</div>
<form:textarea path="address" rows="5" cols="40"/>

<br>
<input type="submit" value="Sign Up">
</form:form>
</body>
</html>
```

login.htm

```
<%@page pageEncoding="utf-8" contentType="text/html; charset=utf-8" %>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Đăng nhập</title>
</head>
<body>
    <h2>Login Form</h2>
    <h4 style="color:red">${errors}</h4>
    <form:form action="login.htm"
        method="post" modelAttribute="user">
        <div>User Name:</div>
        <form:input path="id"/>

        <div>Password:</div>
        <form:password path="password"/>

        <br>
        <input type="submit" value="Sign In">
    </form:form>

</body>
</html>
```

Bước 6: Tạo Controller

```
package com.lycato.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.lycato.dao.CustomerDAO;
import com.lycato.entity.Customer;

@Controller
public class AccountController {
    /**
     * Inject từ @Repository CustomerDAO
     */
}
```

```

    */
    @Autowired
    private CustomerDAO dao;

    /**
     * GET: register.htm
     */
    @RequestMapping(value="register", method=RequestMethod.GET)
    public String register(ModelMap model) {
        model.addAttribute("user", new Customer());
        return "Register";
    }

    /**
     * POST: register.htm
     */
    @RequestMapping(value="register", method=RequestMethod.POST)
    public String register(ModelMap model,
        @ModelAttribute("user") Customer user) {
        try{
            dao.insert(user);
            return "Login";
        }
        catch (Exception ex){
            model.addAttribute("errors", "User Name đã được sử dụng !");
            return "Register";
        }
    }

    /**
     * GET: login.htm
     */
    @RequestMapping(value="login", method=RequestMethod.GET)
    public String login(ModelMap model) {
        model.addAttribute("user", new Customer());
        return "Login";
    }

    /**
     * GET: login.htm
     */
    @RequestMapping(value="login", method=RequestMethod.POST)
    public String login(ModelMap model,
        @ModelAttribute("user") Customer user) {
        try{
            Customer cust = dao.getById(user.getId());
            if(cust.getPassword().equals(user.getPassword())){
                model.addAttribute("errors", "Đăng nhập thành công !");
            }
            else{
                model.addAttribute("errors", "Sai mật khẩu !");
            }
        }
        catch (Exception ex){
            model.addAttribute("errors", "Sai mã đăng nhập !");
        }
        return "Login";
    }
}

```

Khi bạn tương tác vào chương trình tương:



LAB5: SECURITY VỚI JdbcTemplate

STT	Hành động	Phương thức	Mô tả
1	Chạy register.htm	register(), GET	Hiển thị form đăng ký
2	Nhấp [Register]	register(), POST	Đăng ký – thêm mới thành viên
3	Chạy login.htm	Login(), GET	Hiển thị form đăng nhập
4	Nhấp [Sign In]	Login(), POST	Đăng nhập – xác thực

Bước 7: Chạy

<http://localhost:8080/SpringMVCJdbc/register.htm>