



**ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN**

---

## **ĐỒ ÁN TÌM HIỂU**

**Môn Công nghệ Java cho hệ thống phân tán**

**Báo cáo tìm hiểu [Spring MVC Framework](#)**

Version [3.0](#)

Sinh viên thực hiện:

[0612089 – Nguyễn Đức Linh Giang](#)

[0612119 – Nguyễn Đức Hoàng](#)

Spring MVC Framwork	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

## Mục lục

1	Giới thiệu về Spring MVC Framework :	2
1.1	Spring Framework :	2
1.2	Spring MVC Framework :	3
1.2.1	Các thành phần quan trọng nhất và chức năng của một Spring MVC framework:	3
1.2.2	Luồng xử lý request trong Spring MVC Framework :	4
2	Ứng dụng minh hoạ cơ bản (Hello world):	5
2.1	Môi trường triển khai ứng dụng :	5
2.2	Các bước thực hiện :	6
2.2.1	Tạo một Project Spring Web MVC bằng NetBeans :	6
2.2.2	Phân tích các thành phần trong Project :	9
2.2.3	Phát triển một ứng dụng đơn giản :	11
3	Ứng dụng minh hoạ nâng cao :	22
3.1	Phát biểu bài toán	22
3.2	Các bước xây dựng:	22
3.2.1	Tạo CSDL và liên kết khoá ngoại	22
3.2.2	Tạo Project áp dụng Spring Framework và Hibernate	28
3.2.3	Phát sinh các mapping file và pojoes tương ứng cho Hibernate	32
3.2.4	Tạo các lớp DAO và BUS	39
3.2.5	Tạo Controller cho chức năng thêm sản phẩm mới :	42
3.2.6	Tạo Views :	46
3.2.7	Config Controller :	49
3.2.8	Tạo Validation :	52
4	Các ưu điểm và khuyết điểm của Spring MVC Web Framework :	56
5	Tài liệu tham khảo :	56

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

# 1 Giới thiệu về Spring MVC Framework :

## 1.1 Spring Framework :

Spring là một Framework mã nguồn mở có phiên bản dùng cho Java Platform và cả .NET Platform.

Phiên bản đầu tiên được viết bởi Rod Johnson và đưa ra cùng cuốn sách Expert One-on-One J2EE Design and Development được xuất bản tháng 10 năm 2002.

Spring Framework có thể dùng cho tất cả các ứng dụng viết bằng Java, nhưng nó thành công nhất trên lĩnh vực ứng dụng web trên nền Java EE.

Spring Framework gồm các module chính sau :

- **Inversion of Control container:** hiệu chỉnh các components của chương trình và quản lý vòng đời (lifecycle) của các đối tượng Java.
- **Aspect-oriented programming:** kỹ thuật lập trình mới cho phép đóng gói những hành vi có liên quan đến nhiều lớp
- **Data access:** làm việc với *relational database management systems* (hệ thống quản lý cơ sở dữ liệu quan hệ) trên nền Java platform sử dụng JDBC và công cụ *object-relational mapping*.
- **Transaction management:** thống nhất các hàm APIs quản lý transaction và điều phối transactions cho đối tượng Java.
- **Model-View-Controller (MVC)** một framework dựa trên HTTP và Servlet cung cấp khả năng mở rộng và tùy biến nhiều hơn.
- **Remote Access framework:** hiệu chỉnh RPC-style trong việc import và export các đối tượng java thông qua mạng lưới hỗ trợ phương thức RMI, CORBA và HTTP bao gồm SOAP.
- **Convention-over-configuration:** một chương trình nhanh mạnh trong việc phát triển các hướng giải quyết cho cái chương trình sử dụng Spring enterprise.
- **Batch processing:** một framework tốt cho việc xử lý một lượng lớn thông tin và hàm như logging/tracing, transaction management, job processing statistics, job restart, skip, and resource management.
- **Authentication and authorization:** hiệu chỉnh chế độ an toàn bảo mật, cung cấp các phương thức, chương trình ở mức cơ bản cho chương trình sử dụng Spring.
- **Remote Management:** hiệu chỉnh cách hiển thị và quản lý các đối tượng java ở mức local hoặc remote qua JMX.
- **Messaging:** hiệu chỉnh việc xử lý các thông tin trao đổi giữa các đối tượng dựa trên chuẩn JMS APIs.
- **Testing:** cung cấp các lớp hỗ trợ việc viết các unit kiểm tra và phân tích lỗi.

Trong đề tài tìm hiểu này. Chỉ quan tâm và đề cập đến module **Model-View-Controller (MVC)** trong **Spring Framework**. Ta gọi nó là **Spring MVC Framework**.

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

## 1.2 Spring MVC Framework :

Tương tự như Struts, Spring MVC là một request-based framework. Họ định nghĩa các phương pháp chung cho tất cả các response phải được giải quyết bằng một request-based framework. Mục đích của họ đơn giản là dễ dàng hơn cho developer viết các bổ sung và các cải tiến của riêng họ.

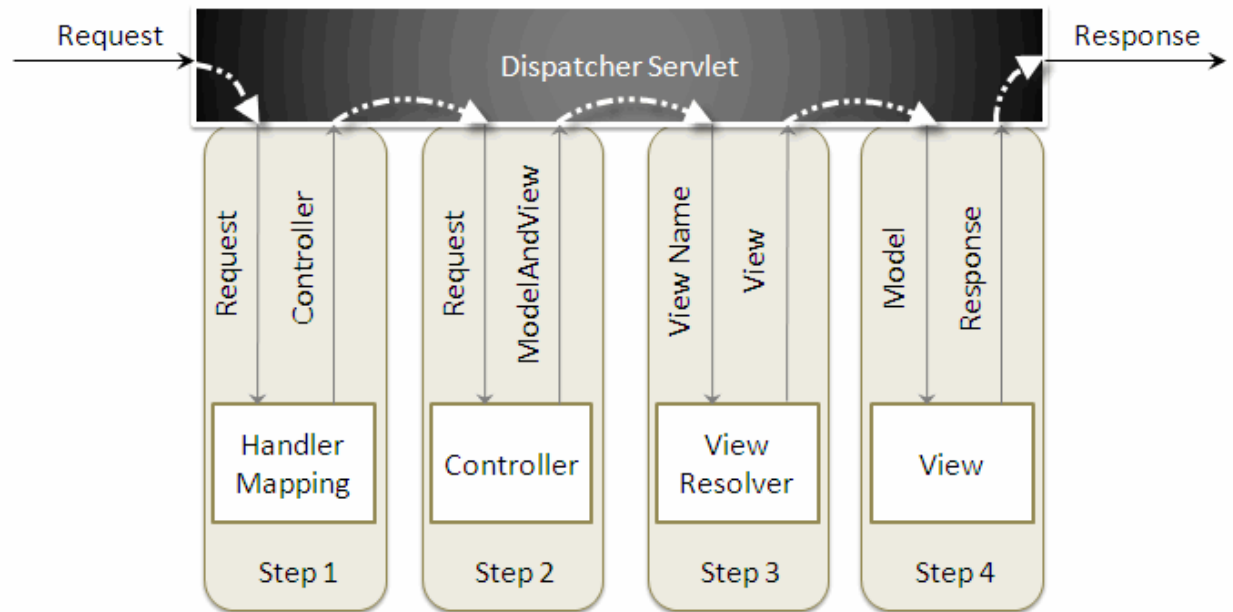
Spring MVC giúp cho việc xây dựng ứng dụng web một cách chặt chẽ và linh động. Mẫu thiết kế Model-View-Controller giúp phân chia rạch ròi 3 công việc business logic, presentation logic, và navigation logic. Models chịu trách nhiệm đóng gói dữ liệu của ứng dụng. Views có nhiệm vụ hiển thị thông tin cung cấp bởi đối tượng Model trả về cho người dùng. Controllers chịu trách nhiệm nhận request từ người dùng và gọi các dịch vụ bên dưới để xử lý.

### 1.2.1 Các thành phần quan trọng nhất và chức năng của một Spring MVC framework:

- **DispatcherServlet** là một lớp đứng ra quản lý toàn bộ các hành động của framework (front controller) trong suốt quá trình thực thi các lệnh thông qua HTTP request.
- **HandlerMapping**: chọn một đối tượng sẽ xử lý các request dựa trên các thuộc tính và điều kiện của các request đó.
- **HandlerAdapter**: thực thi các handler đã được chọn.
- **Controller**: đứng giữa Model và View để quản lý các request được gửi tới và chuyển các response chính xác.
- **View**: chịu trách nhiệm trả các response cho client.
- **ViewResolver**: chọn phương pháp view dựa trên các logical name có sẵn của **View**.
- **HandlerInterceptor**: ngăn chặn (lọc) các request từ user. Nó được coi như **Servlet filter** (không bắt buộc và không bị quản lý bởi **DispatcherServlet**).
- **LocaleResolver**: xử lý và lưu một phần các thông tin của user.
- **MultipartResolver**: làm cho việc upload file dễ dàng hơn bằng cách gói các request lại.

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

### 1.2.2 Luồng xử lý request trong Spring MVC Framework :



Trích từ <http://www.vaannila.com/spring/spring-mvc-tutorial-1.html>

Việc xử lý request và response trong **Spring MVC Framework** được mô tả như sau :

- **Bước 1 :**
  - o *DispatcherServlet* nhận Request.
  - o *DispatcherServlet* tra trong **HandlerMapping** và gọi *Controller* kết hợp với Request.
- **Bước 2 :**
  - o *Controller* xử lý Request bằng cách gọi những phương thức dịch vụ thích hợp và sau đó trả về một đối tượng **ModelAndView** cho *DispatcherServlet*. Đối tượng **ModelAndView** này chứa dữ liệu trong đối tượng Model và tên của View.
- **Bước 3 :**
  - o *DispatcherServlet* gửi tên của View đến cho một *ViewResolver*. *ViewResolver* sẽ tìm View thực sự cần dùng.
- **Bước 4 :**
  - o *DispatcherServlet* truyền đối tượng **Model** đến cho *View* đã xác định để hiển thị kết quả.
  - o *View* lấy dữ liệu trong đối tượng **Model** và hiển thị kết quả cho người dùng.

<a href="#">Spring MVC Framework</a>	Phiên bản: <a href="#">3.0</a>
Báo cáo tìm hiểu	Ngày: <a href="#">15/05/2010</a>

## 2 Ứng dụng minh họa cơ bản (Hello world):

Sau đây là một ví dụ HelloWorld đơn giản áp dụng Spring MVC Framework.

### 2.1 Môi trường triển khai ứng dụng :

- IDE : NetBeans IDE 6.8

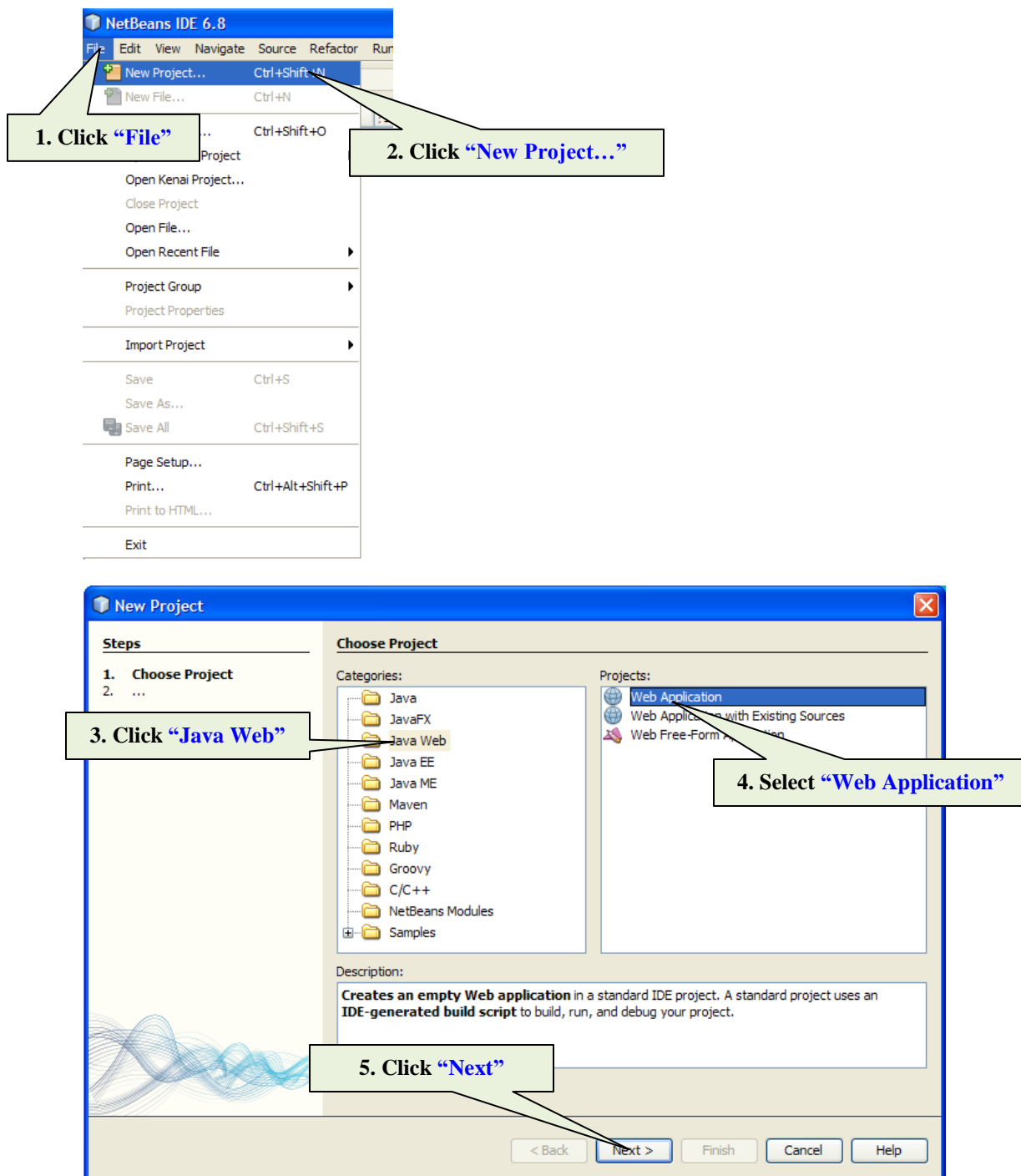


- Web Server : Apache Tomcat 6.0.20

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

## 2.2 Các bước thực hiện :

### 2.2.1 Tạo một Project Spring Web MVC bằng NetBeans :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

**New Web Application**

**Steps**

1. Choose Project
- 2. Name and Location**
3. Server and Settings
4. Frameworks

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Set as Main Project

**New Web Application**

**Steps**

1. Choose Project
2. Name and Location
- 3. Server and Settings**
4. Frameworks

**Server and Settings**

Add to Enterprise Application:

Server:

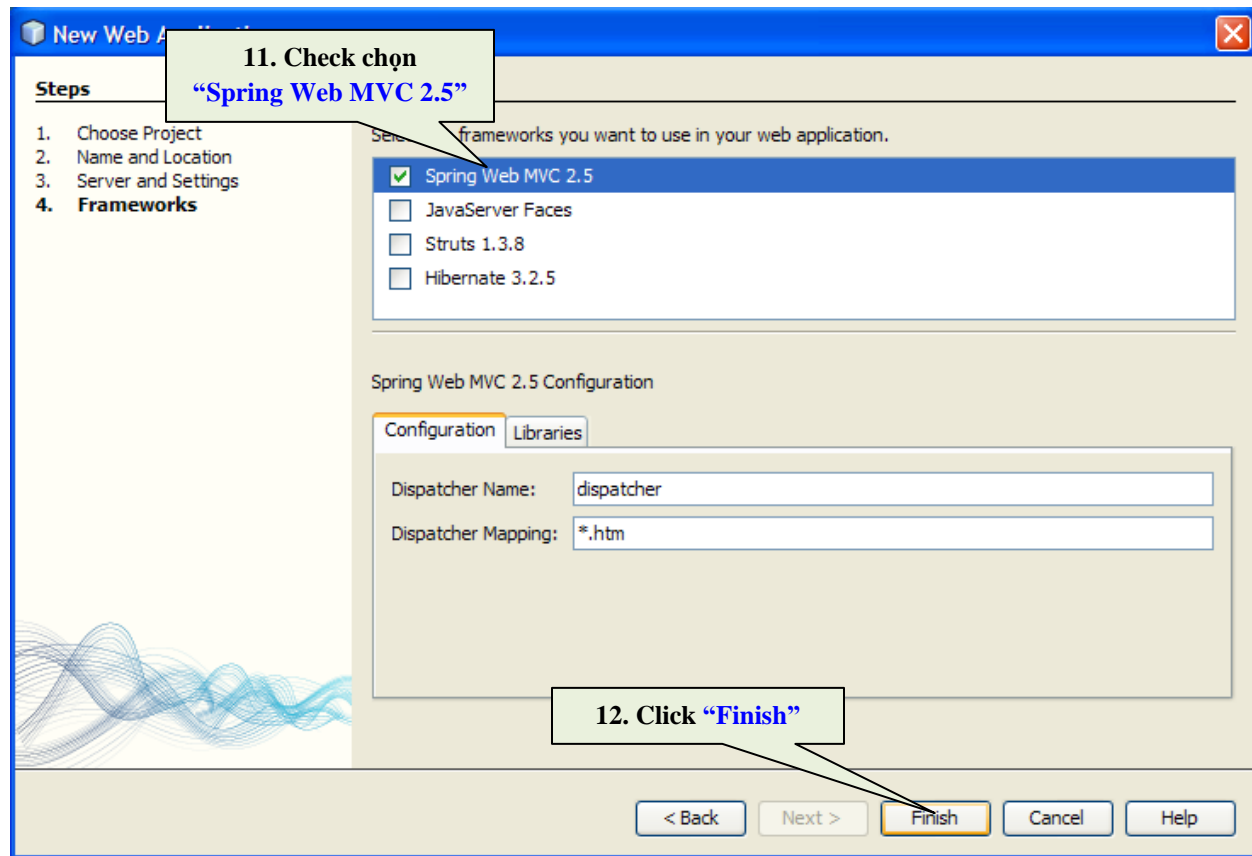
☐ Use dedicated library folder for server JAR files

Java EE Version:

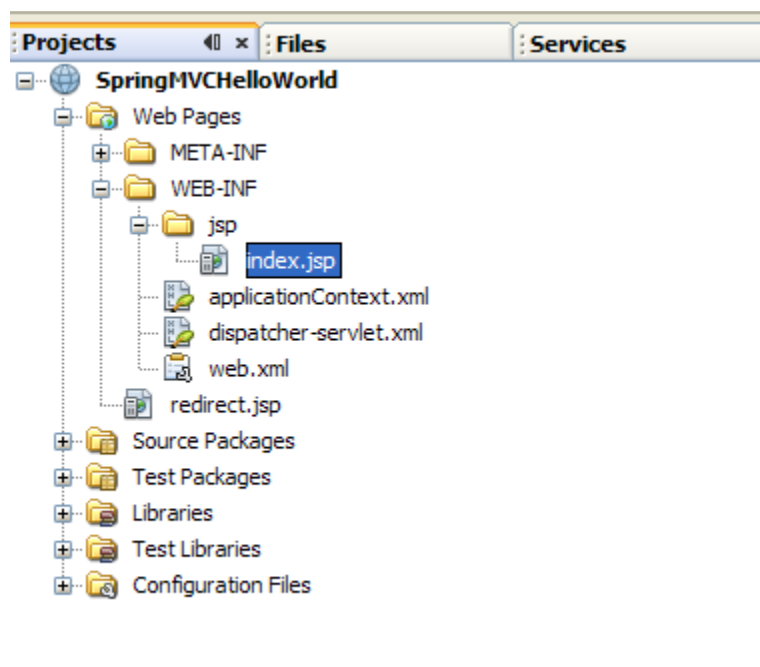
Context Path:



Spring MVC Framwork	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

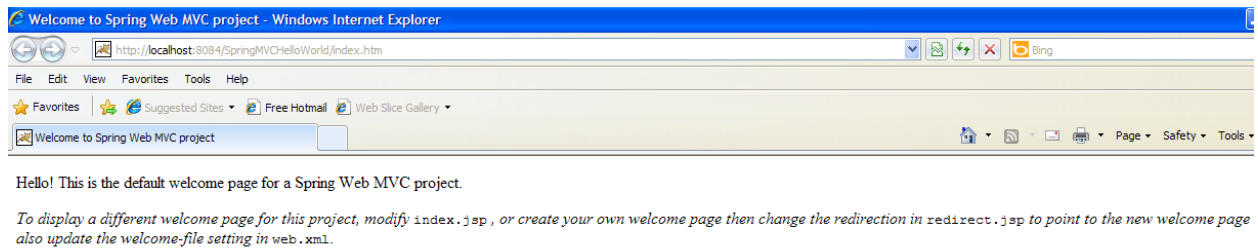


Kết quả ta có Project SpringMVCHelloWorld với cấu trúc thư mục như sau :

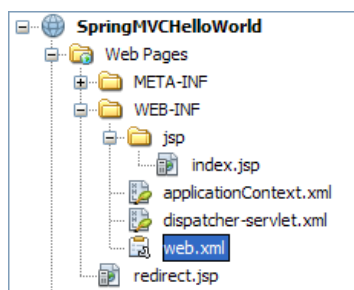


Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

Run ứng dụng ta có được kết quả như sau :



## 2.2.2 Phân tích các thành phần trong Project :



- Trong file **web.xml**, ta thấy có dòng thiết lập chỉ trang chủ của ứng dụng là **redirect.jsp** như sau ở cuối file :

```
<welcome-file-list>
  <welcome-file>redirect.jsp</welcome-file>
</welcome-file-list>
```

- Trong file **redirect.jsp** chỉ làm duy nhất một việc đó là redirect tất cả những request gửi đến tới **index.htm** :

```
<% response.sendRedirect("index.htm"); %>
```

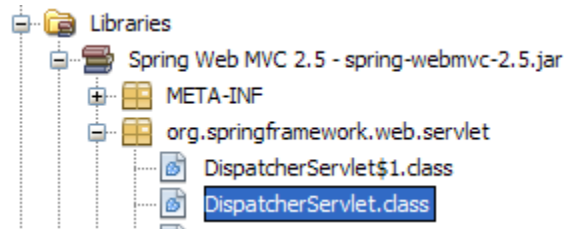
- Cũng trong file **web.xml** ta thấy dòng khai báo và ánh xạ sau :

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

Servlet **dispatcher** là lớp kế thừa từ  
"**org.springframework.web.servlet.DispatcherServlet**"

Ánh xạ tất cả requests có mẫu URL khớp \*.htm cho lớp  
**DispatcherServlet** của Spring này

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



- Class **DispatcherServlet** quản lý những request đến ứng dụng dựa trên những thiết lập cấu hình được tìm thấy trong file *dispatcher-servlet.xml*
- Giờ ta xét đến file *dispatcher-servlet.xml*, ta chú ý đến đoạn code sau :

```

<!--
Most controllers will use the
for the index controller we a
define an explicit mapping fo
-->
<bean id="urlMapping" class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping">
  <property name="mappings">
    <props>
      <prop key="index.htm">indexController</prop>
    </props>
  </property>
</bean>

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

<!--
The index controller.
-->
<bean name="indexController"
  class="org.springframework.web.servlet.mvc.ParameterizableViewController"
  p:viewName="index" />

```

Khi DispatcherServlet nhận một request khớp \*.htm như là index.htm nó sẽ tìm bên trong **urlMapping** này một controller cung cấp cho request đó.

Ánh xạ **index.htm** với **indexController**

**viewResolver** nhận vào **Logical view name** từ **indexController** và tìm trong thư mục **“/WEB-INF/jsp/”** trang jsp có tên tương ứng.

**indexController** là lớp kế thừa từ **“org.springframework.web.servlet.mvc.ParameterizableViewController”** đây là lớp được cung cấp bởi Spring và nó đơn giản sẽ trả về 1 view.

Logical view name : Dùng để **viewResolver** xử lý

⇒ Điều này cho phép định vị những file trong gói file WAR của ứng dụng lúc runtime và response với trang jsp tương ứng.

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

### 2.2.3 Phát triển một ứng dụng đơn giản :

#### 2.2.3.1 Tổng quan :

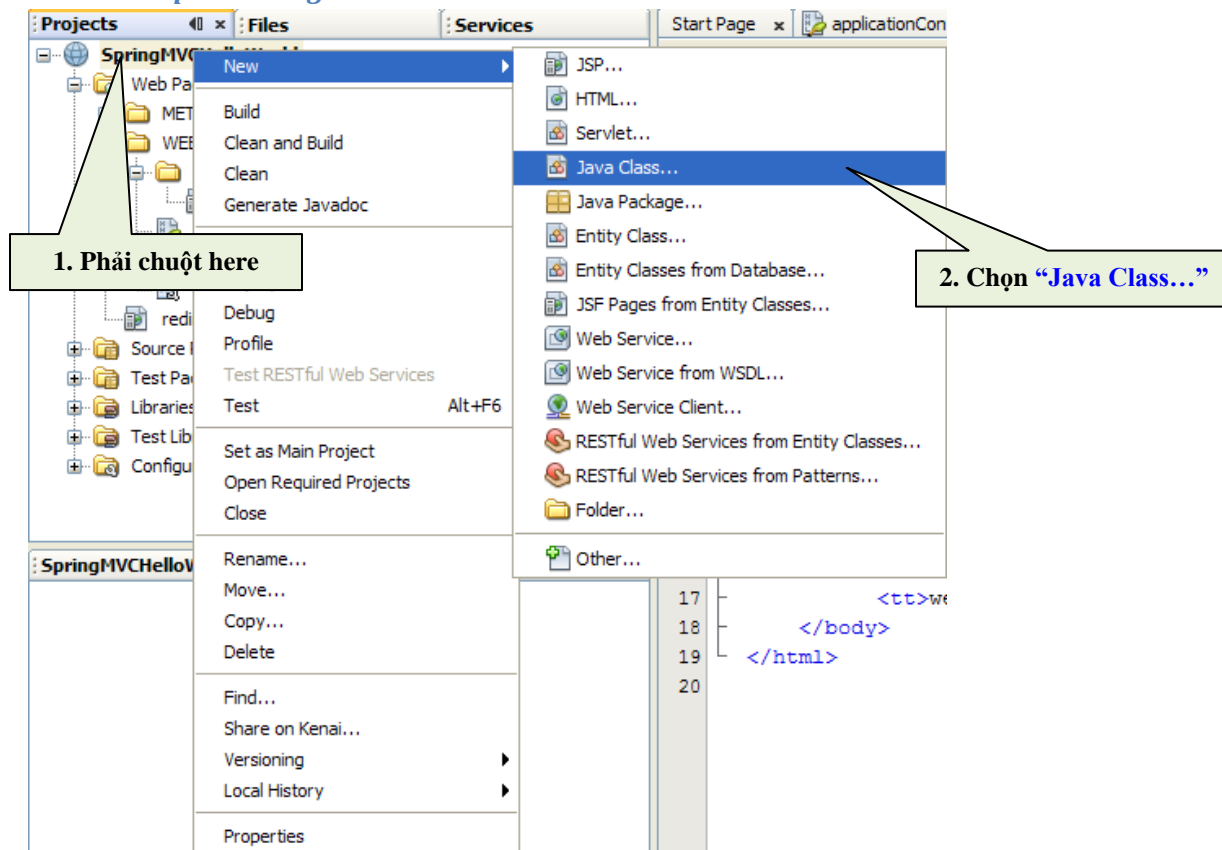
Ứng dụng này sẽ bao gồm 2 trang JSP hay còn gọi là 2 View trong công nghệ Spring Web MVC.

- View thứ nhất chứa 1 form HTML với một field yêu cầu nhập tên của người dùng.
- View thứ 2 là một trang đơn giản chỉ việc hiển thị : **Hello + [tên người dùng vừa nhập]**

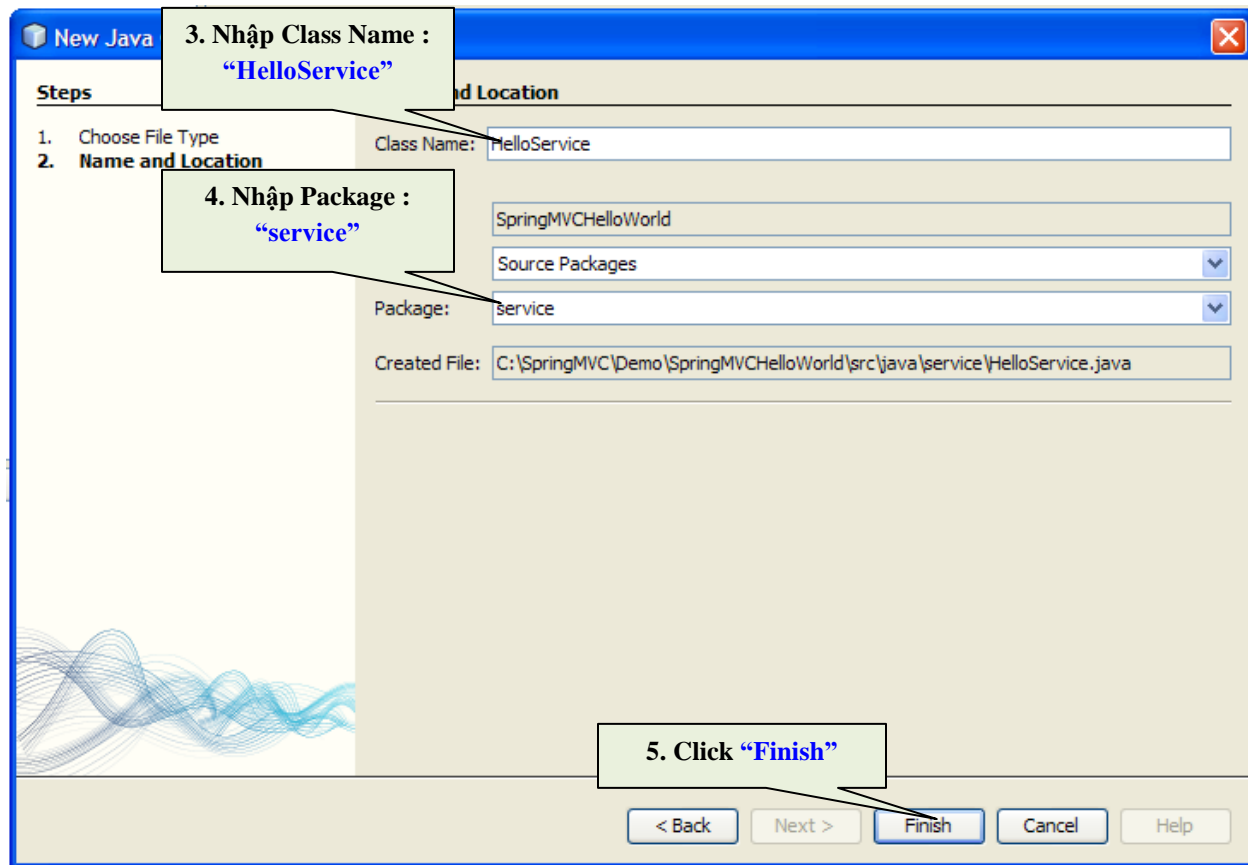
Những **views** này sẽ được quản lý bởi một **controller**, **controller** này sẽ nhận request và quyết định **view** nào được trả về. Nó cũng sẽ truyền cho **view** được chọn những thông tin cần thiết để view hiển thị, những thông tin này được gọi là **Model**. Ta sẽ đặt tên controller này là **HelloController**.

Trong những ứng dụng web phức tạp, những xử lý nghiệp vụ thuộc tầng Business Logic không được chứa trực tiếp trong **controller** mà thay vào đó những thực thể khác gọi là **service** được dùng bởi **controller** khi nó cần thực hiện những nghiệp vụ của tầng Business Logic. Trong ứng dụng này, nghiệp vụ cần xử lý là việc tính toán và tạo ra câu thông điệp hello, cho nên vì mục đích này ta tạo một lớp **HelloService**.

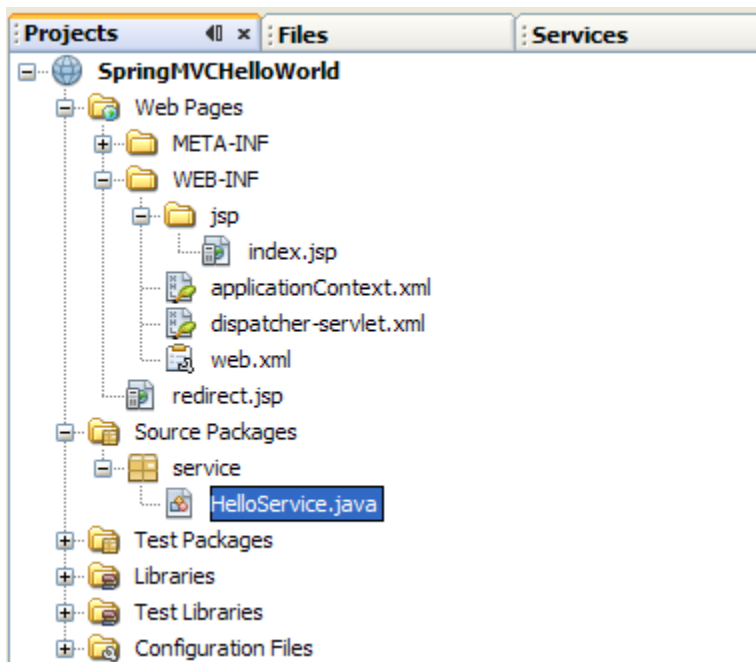
#### 2.2.3.2 Implementing the HelloService :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



Ta có lớp HelloService được tạo :



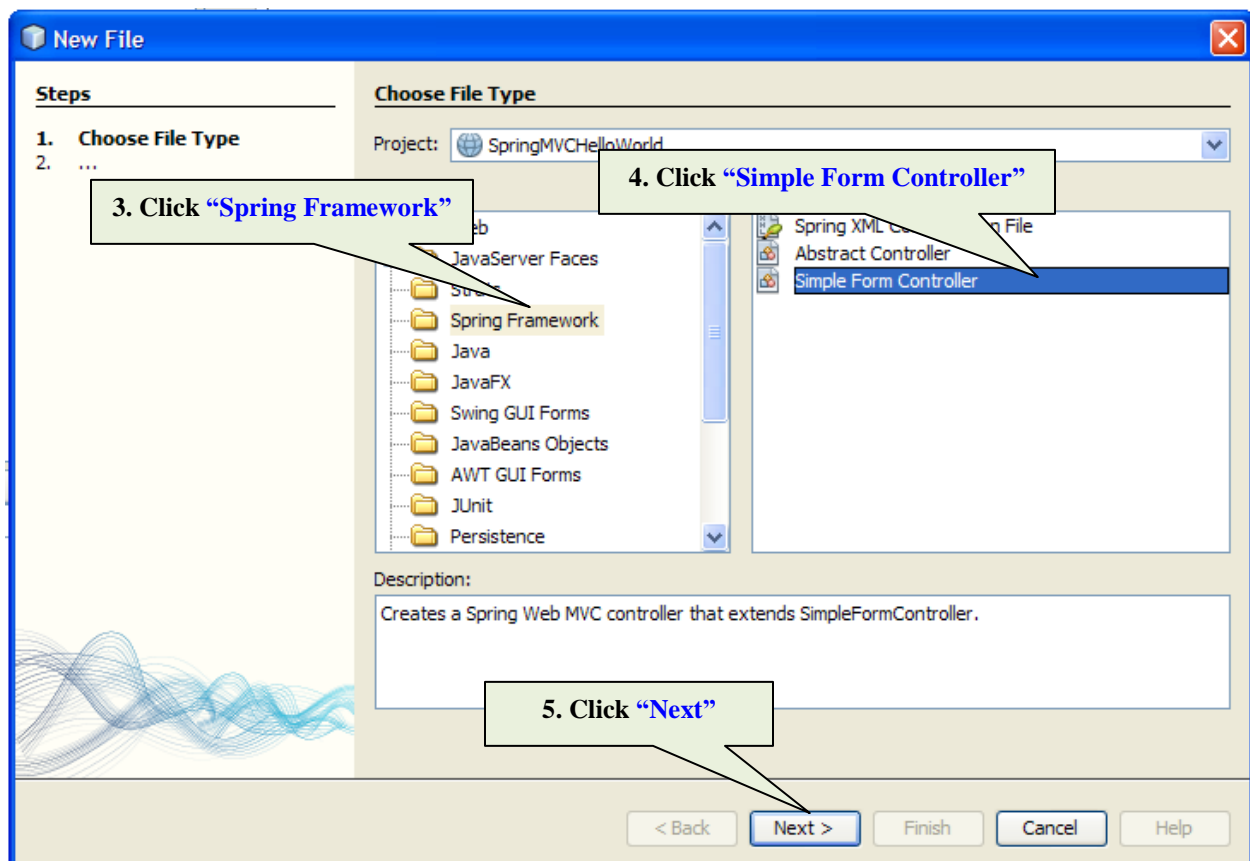
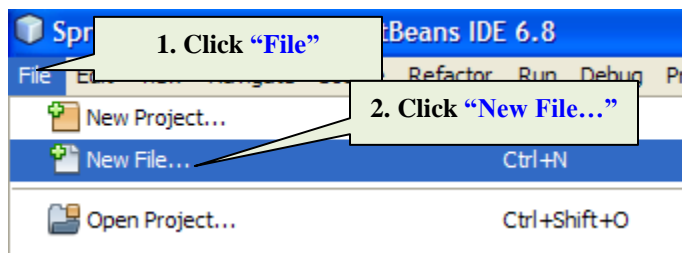
Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Thêm vào lớp này phương thức **sayHello** như sau :

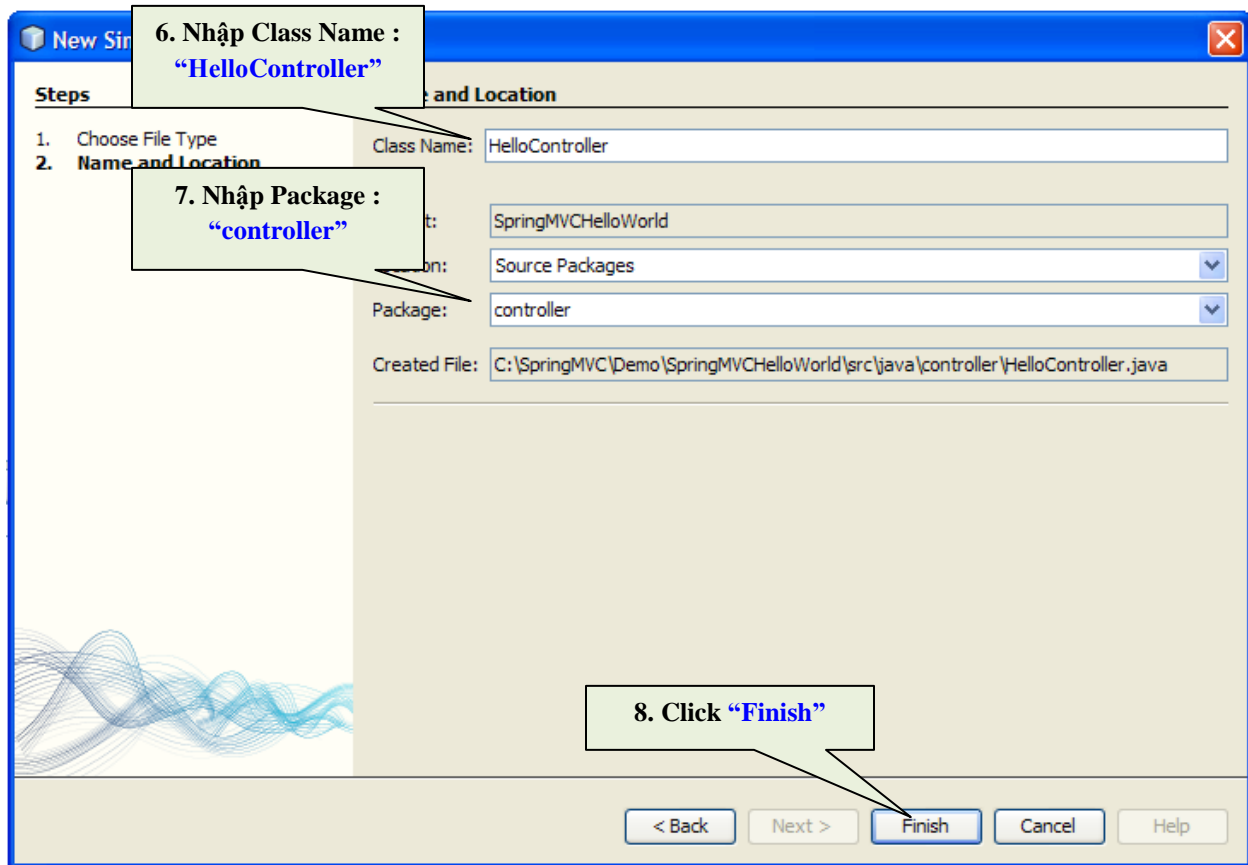
```
public class HelloService {
    public String sayHello(String name) {
        return "Hello " + name + "!";
    }
}
```

### 2.2.3.3 Implementing the Controller and Model

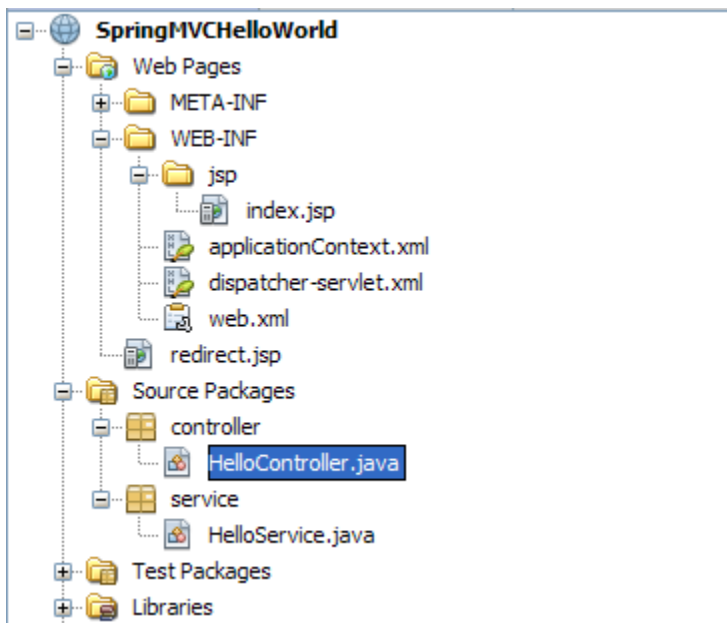
Ở đây ta có thể dùng một **SimpleFormController** để quản lý dữ liệu người dùng và xác định view nào được trả về.



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



Kết quả ta tạo được lớp **HelloController** :



Ta edit lại constructor của lớp **HelloController** như sau :

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

```

public HelloController() {
    //Initialize controller properties here or
    //in the Web Application Context

    setCommandClass(Name.class);
    setCommandName("name");
    setSuccessView("helloView");
    setFormView("nameView");
}

```

Một instance của lớp này sẽ được tạo với mỗi request.

Định nghĩa tên của command trong model

Đặt tên view được dùng để hiển thị thông điệp Hello khi submit thành công

Đặt tên view được dùng để hiển thị form nhập tên

Bây giờ ta cần tạo class Name như một bean đơn giản để giữ thông tin của mỗi request.

Tương tự như cách tạo lớp ở trên.

- Phải chuột vào project chọn **New => Java Class...**
- Tạo một lớp tên là **Name** trong package **pojo**.

Trong lớp Name này ta tạo một biến private **value** để lưu tên người dùng nhập vào. Sau đó là các phương thức get set. Như sau :

```

public class Name {
    private String value;

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }
}

```

Quay trở lại class **HelloController**, ta xóa phương thức **doSubmitAction()** đi và bỏ comment phương thức **onSubmit()** bên dưới. Phương thức **onSubmit** này cho phép ta tự tạo **ModelAndView**.

- Import **org.springframework.web.servlet.ModelAndView**. Lớp **ModelAndView** được trả về bởi một controller và được xử lý bởi một **DispatcherServlet**. View có thể được lấy dựa vào chuỗi View Name mà sẽ được xử lý bởi một đối tượng **ViewResolver**, hoặc đối tượng **View** có thể được chỉ định trực tiếp.
- Thêm một biến mức độ lớp kiểu **HelloService** cho **HelloController**. Import **service>HelloService** và thêm những dòng sau vào lớp **HelloController** :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

```
private HelloService helloService;

public void setHelloService(HelloService helloService) {
    this.helloService = helloService;
}
```

- Edit lại phương thức **onSubmit()** như sau :

```
//Use onSubmit instead of doSubmitAction
//when you need access to the Request, Response, or BindException objects

@Override
protected ModelAndView onSubmit(Object command) throws Exception
{
    Name name = (Name) command;
    ModelAndView mv = new ModelAndView(getSuccessView());
    //Do something...
    mv.addObject("helloMessage", helloService.sayHello(name.getValue()));
    return mv;
}
```

Vào file **applicationContext.xml** để đăng ký **HelloService**. Để đăng ký ta thêm dòng code sau :

```
<?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:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
http://www.springframework.org/schema/aop http://www.springframework.org/
http://www.springframework.org/schema/tx http://www.springframework.org/

<!--bean id="propertyConfigurer"
      class="org.springframework.beans.factory.config.PropertyPlaceholder
      p:location="/WEB-INF/jdbc.properties" />

<bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource"
      p:driverClassName="${jdbc.driverClassName}"
      p:url="${jdbc.url}"
      p:username="${jdbc.username}"
      p:password="${jdbc.password}" /-->

<bean name="helloService" class="service.HelloService" />
<!-- ADD PERSISTENCE SUPPORT HERE (jpa, hibernate, etc) -->

</beans>
```

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Sau đó thực hiện đăng ký **HelloController** trong file **dispatcher-servlet.xml** :

```
<bean name="myController"
      class="controller.HelloController"
      p:helloService-ref="helloService" />
```

Và đăng ký mapping controller của chúng ta với url mong muốn được xử lý :

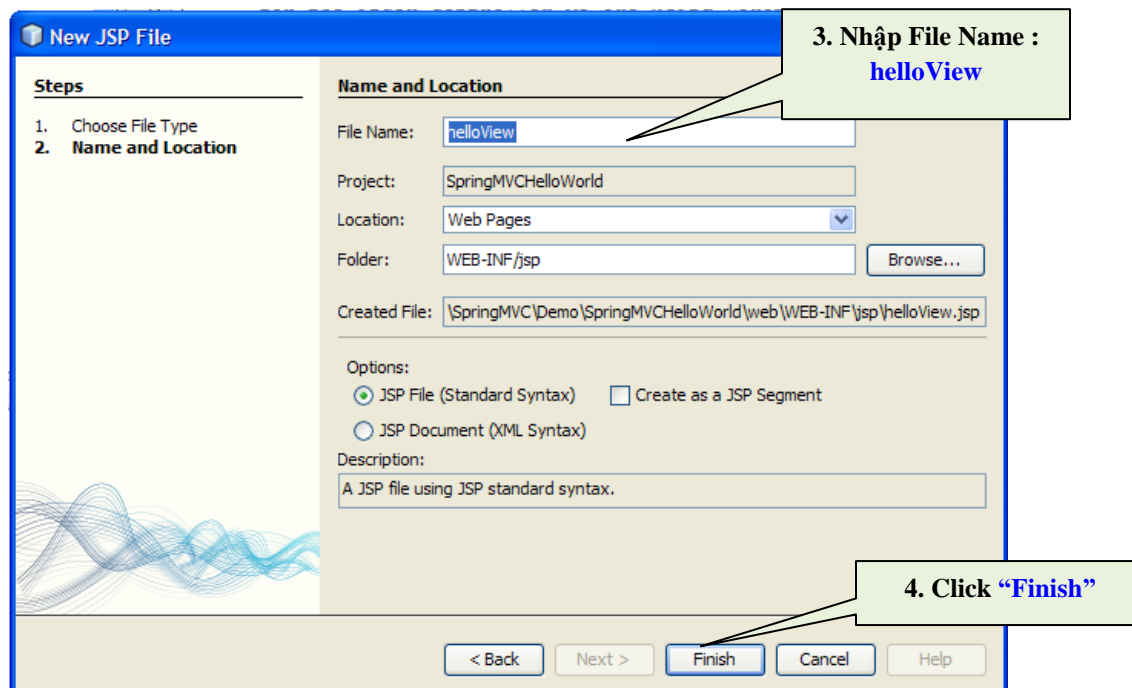
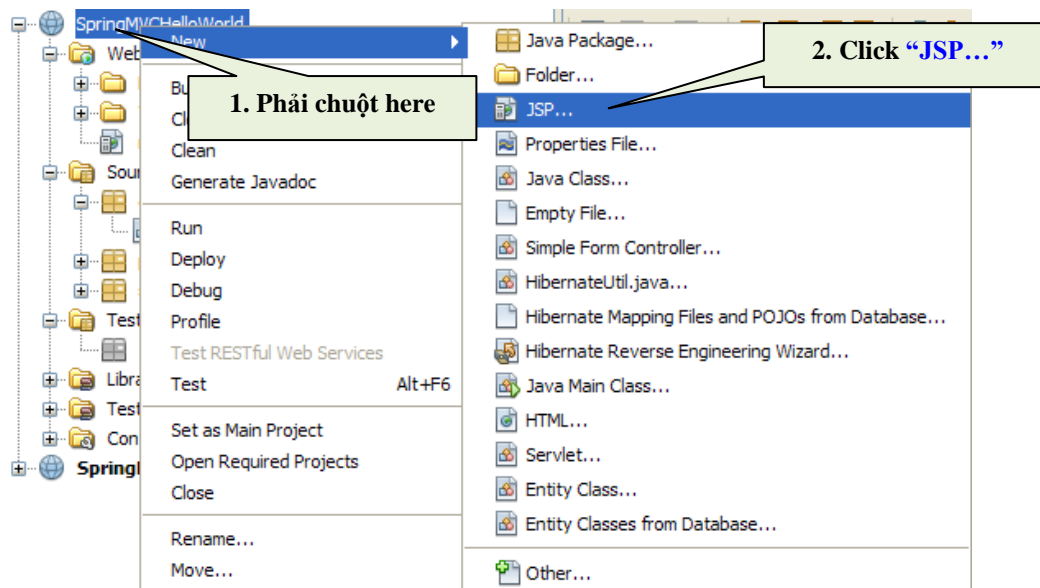
```
<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="index.htm">indexController</prop>
      <prop key="hello.htm">myController</prop>
    </props>
  </property>
</bean>
```

Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

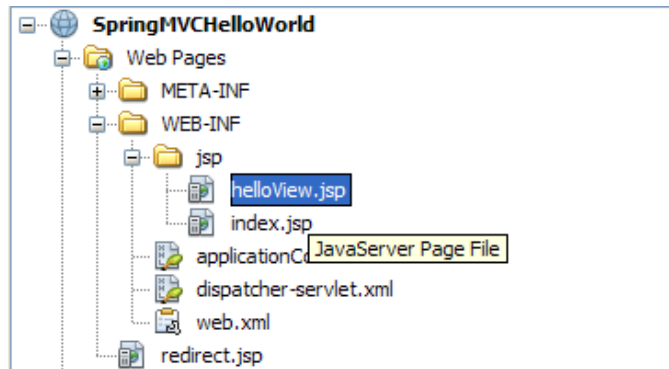
#### 2.2.3.4 Implementing the Views :

Ta cần tạo 2 lớp JSP :

- Cái thứ nhất gọi là **nameView.jsp** hiển thị trang welcome và cho phép người dùng nhập tên.
- Cái thứ hai gọi là **helloView.jsp** hiển thị lời chào cùng với tên của người dùng vừa nhập.



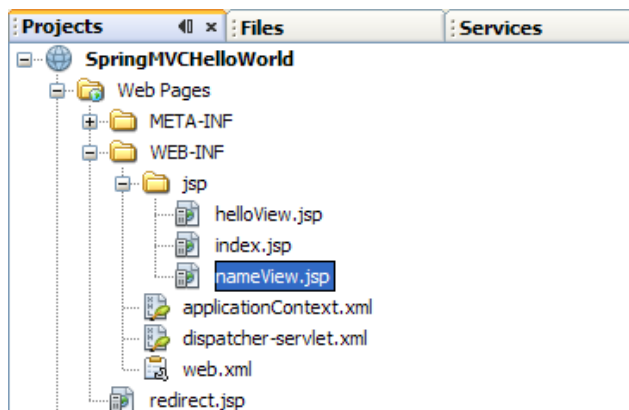
Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



Edit lại file **helloView.jsp** như sau :

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Hello</title>
  </head>
  <body>
    <h2>${helloMessage}</h2>
  </body>
</html>
```

Tương tự ta tạo một trang JSP khác tên **nameView.jsp** tương tự như trên.



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Thay đổi nội dung file này như sau :

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
"http://www.w3.org/TR/html4/loose.d

<%@taglib uri="http://www.springframework.org/tags" prefix="spring" %>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Enter Your Name</ti
  </head>
  <body>
    <spring:nestedPath path="name">
      <form action="" method="post">
        Name:
        <spring:bind path="value">
          <input type="text" name="${status.expression}" value="${status.value}">
        </spring:bind>
        <input type="submit" value="OK">
      </form>
    </spring:nestedPath>
  </body>
</html>
```

Thư viện chứa những tags hữu dụng khi implement views

Cho phép chỉ định một bean

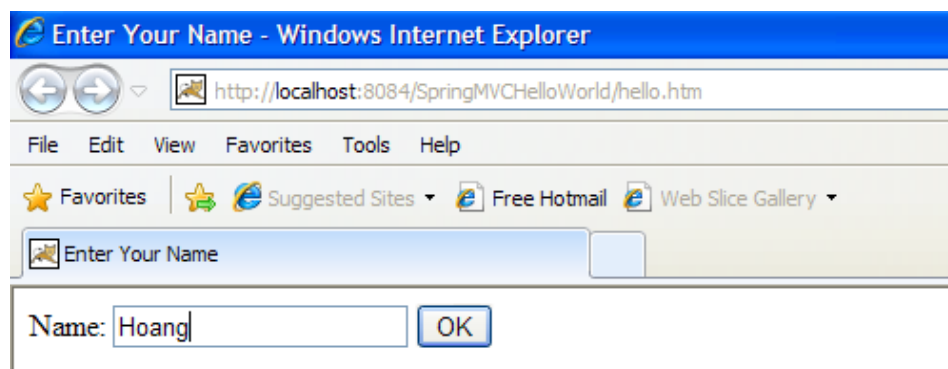
Cho phép kết buộc với một thuộc tính của một bean

Tag **spring:nestedPath** chỉ định một bean và tag **spring:bind** kết buộc tham số với một thuộc tính. Đường dẫn đến bean này trở thành **name.value**. Như cài đặt từ trước, tên command của **HelloController** là **name**. Vì vậy đường dẫn này tham chiếu đến thuộc tính **value** của bean tên là **name** trong phạm vi trang.

Cuối cùng mở file **redirect.jsp** và đổi **index.htm** thành **hello.htm**.

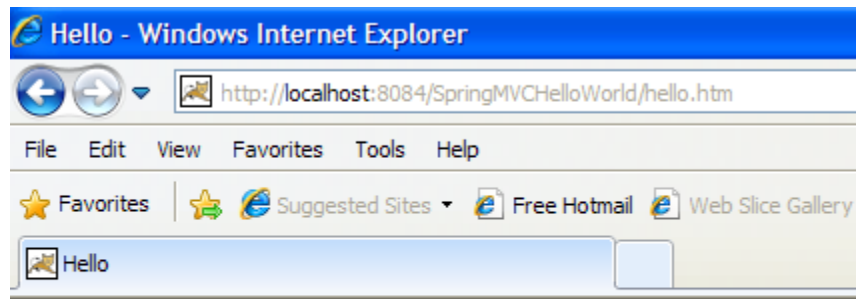
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<% response.sendRedirect("hello.htm"); %>
```

Sau đó Run ứng dụng :



Spring MVC Framwork	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Click OK :



**Hello Hoang!**

Spring MVC Framwork	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

### 3 Ứng dụng minh hoạ nâng cao :

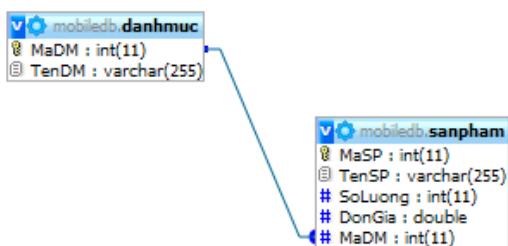
Sau khi trải qua ứng dụng HelloWorld và hiểu khái quát về các bước cần thiết để cấu hình cho ứng dụng sử dụng Spring MVC Framework. Sau đây là một ví dụ nâng cao hơn và gần với thực tế hơn.

Ứng dụng sẽ có đầy đủ kiến trúc của một phần mềm nên có. Dữ liệu của ứng dụng được lưu trữ trong cơ sở dữ liệu.

#### 3.1 Phát biểu bài toán

Xây dựng chức năng thêm thêm sản phẩm mới cho một ứng dụng web thương mại điện tử.

Để lưu trữ dữ liệu cho chức năng này ta cần 2 bảng DanhMuc và SanPham với quan hệ khoá ngoại như hình dưới



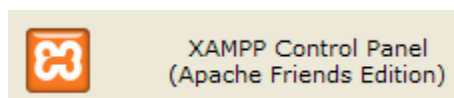
Ứng dụng được xây dựng theo kiến trúc 3 tầng dùng Spring MVC ở tầng giao diện (presentation) và Hibernate ở tầng truy xuất dữ liệu.

#### 3.2 Các bước xây dựng:

##### 3.2.1 Tạo CSDL và liên kết khoá ngoại

###### 3.2.1.1 Môi trường tạo lập :

Trong ứng dụng này ta sử dụng Hệ quản trị CSDL MySQL cung cấp bởi **XAMPP Control Panel** :



Có thể download tại url sau :

<http://www.apachefriends.org/en/xampp-windows.html>

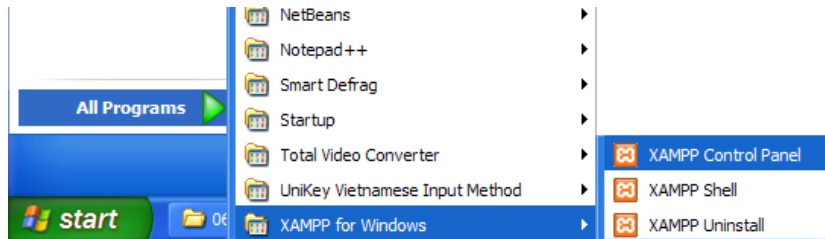
Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

### 3.2.1.2 Các bước thực hiện

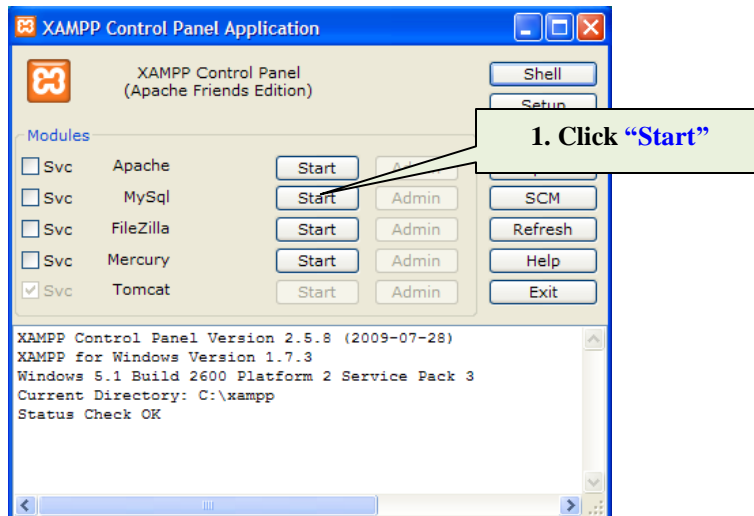
Sau khi cài đặt xong XAMPP và Navicat bây giờ ta bắt đầu công việc chính

- Khởi động XAMPP Control Panel :

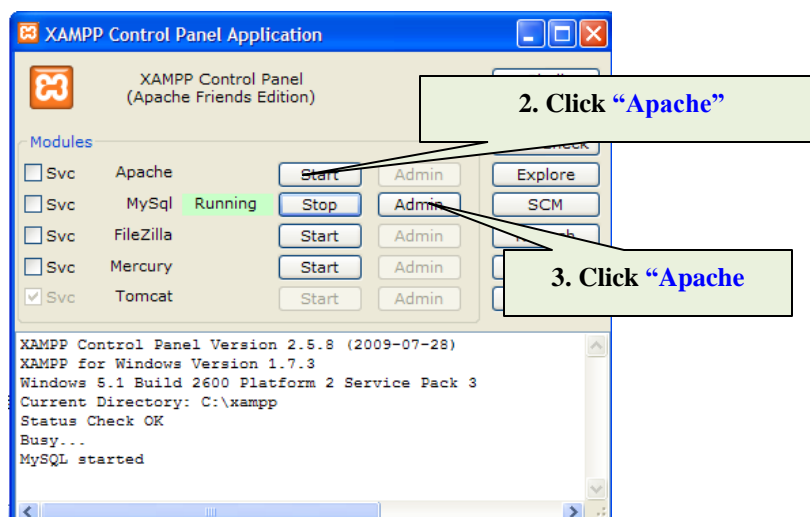
***Start >> All Programs >> XAMPP for Windows >> XAMPP Control Panel***



- Cửa sổ sau hiển thị :



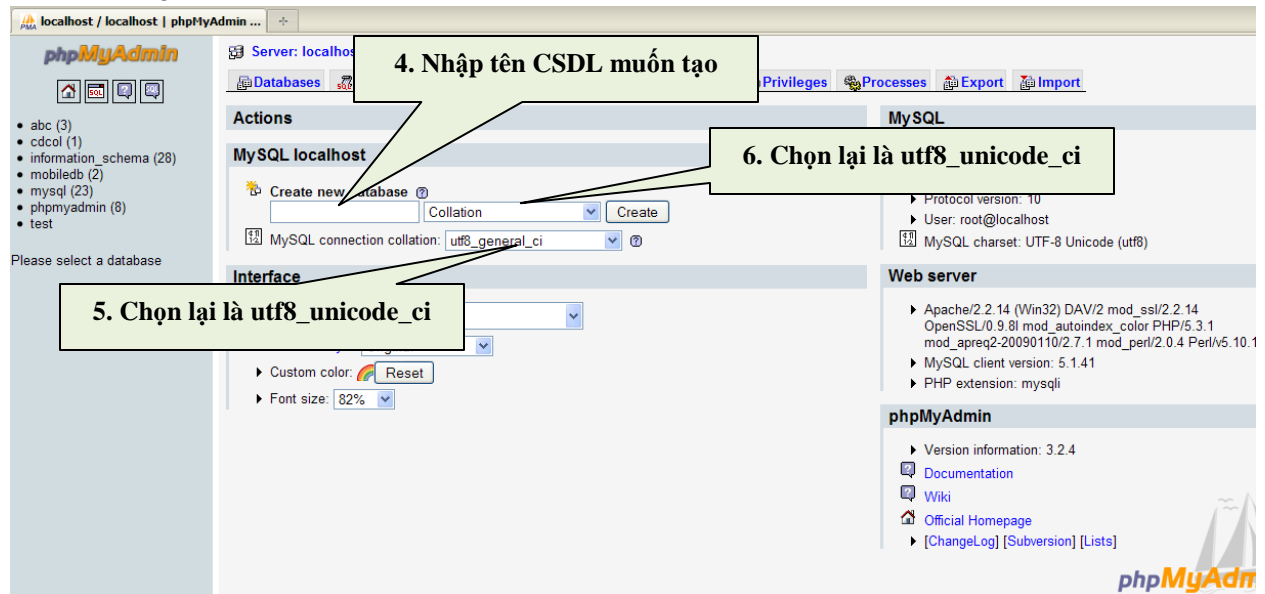
- Kết quả như sau có nghĩa là Hệ quản trị MySQL đã được khởi động và sẵn sàng để sử dụng. Sau đây ta sẽ mở trang cấu hình MySQL để tạo CSDL.



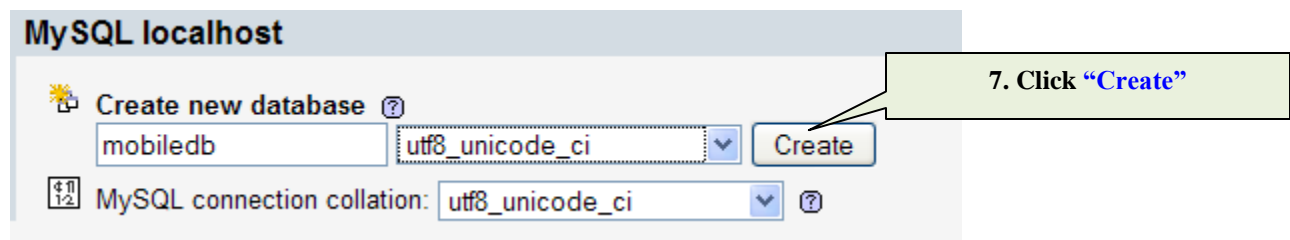


Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

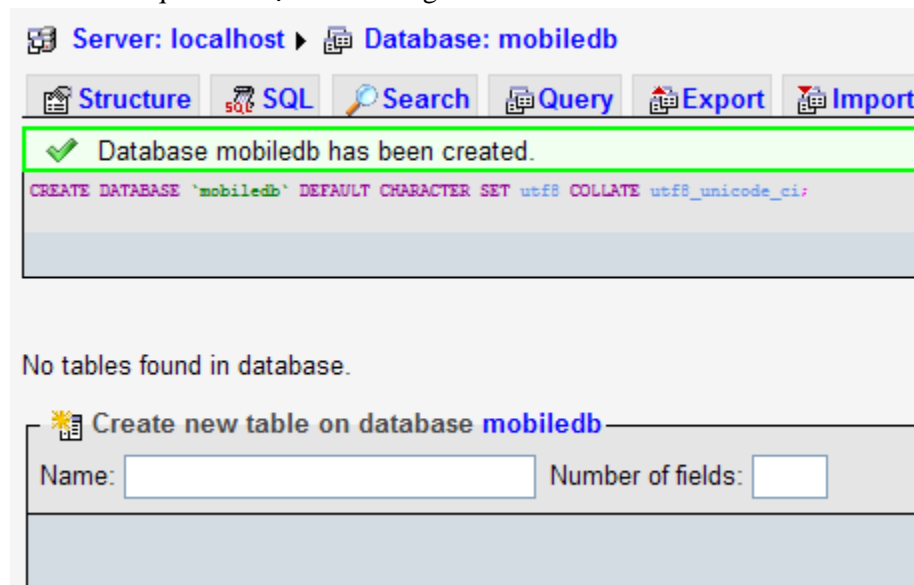
- Trang sau được hiển thị :



Như sau :



- Kết quả ta đã tạo thành công CSDL mới với tên là mobiledb :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

- Giờ ta tạo 2 bảng  
*DanhMuc (MaDM, TenDM)* và *SanPham(MaSP, TenSP, SoLuong, DonGia, MaDM)* :

Create new table on database **mobiledb**

Name:  Number of fields:

8. Click "Go" to create

Kế tiếp ta điền các thông tin thuộc tính của bảng như sau :

Field	<input type="text" value="MaDM"/>	<input type="text" value="TenDM"/>
Type	<input type="text" value="INT"/>	<input type="text" value="VARCHAR"/>
Length/Values <sup>1</sup>	<input type="text"/>	<input type="text" value="255"/>
Default <sup>2</sup>	<input type="text" value="None"/>	<input type="text" value="None"/>
Collation	<input type="text"/>	<input type="text" value="utf8_unicode_ci"/>
Attributes	<input type="text"/>	<input type="text"/>
Null	<input type="checkbox"/>	<input type="checkbox"/>
Index	<input type="text" value="PRIMARY"/>	<input type="text" value="---"/>
AUTO INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comments	<input type="text"/>	<input type="text"/>
MIME type	<input type="text"/>	<input type="text"/>
Browser transformation	<input type="text"/>	<input type="text"/>
Transformation options <sup>3</sup>	<input type="text"/>	<input type="text"/>

9. Fill it

Table comments:

PARTITION definition:

Storage Engine:

Collation:

10. Chọn là "InnoDB" mới có thể thực hiện tạo khoá ngoại

11. Click "Save"

Or Add  field(s)

Kết quả như sau nếu tạo thành công :

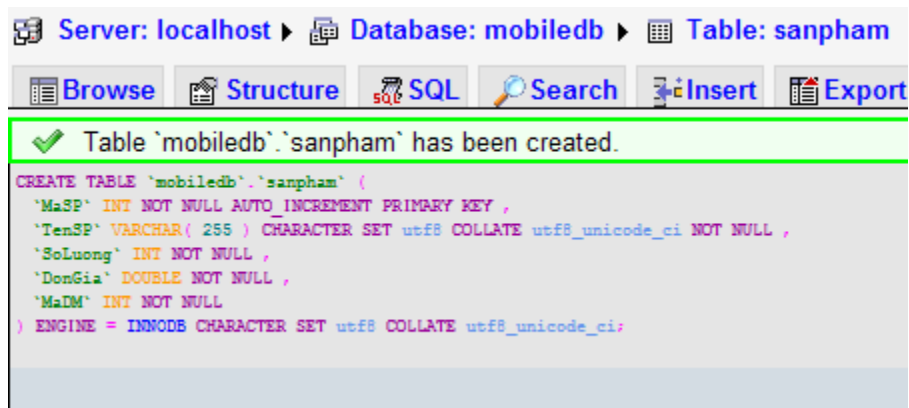
Server: localhost ▶ Database: mobiledb ▶ Table: DanhMuc

✓ Table `mobiledb`.`DanhMuc` has been created.

```
CREATE TABLE `mobiledb`.`DanhMuc` (
  `MaDM` INT NOT NULL AUTO INCREMENT PRIMARY KEY ,
  `TenDM` VARCHAR( 255 ) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL
) ENGINE = INNODB CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

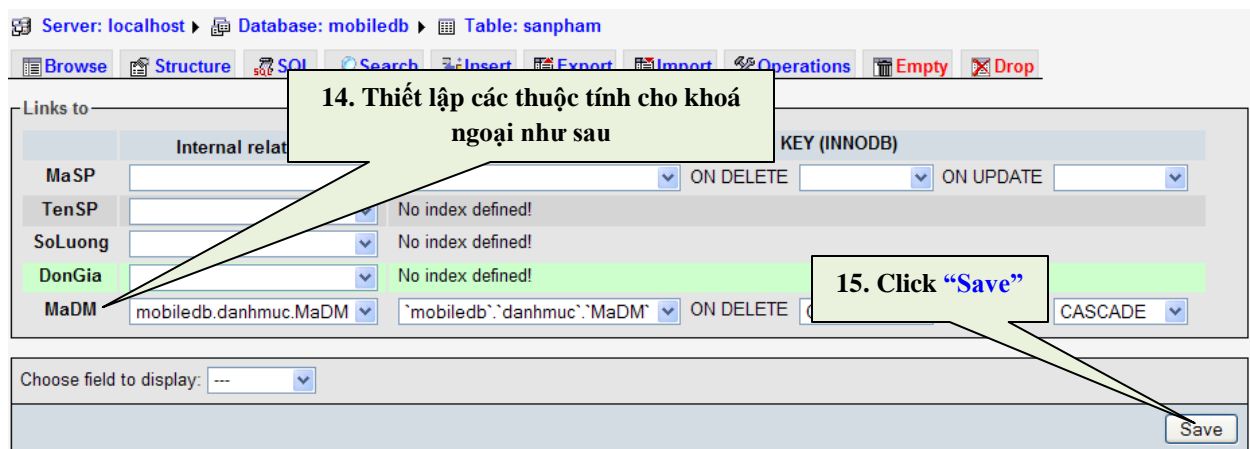
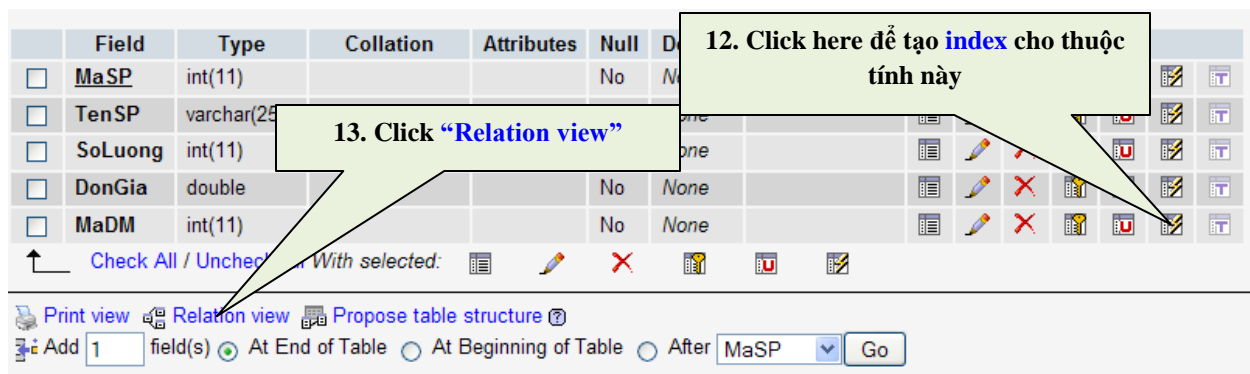
Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Tương tự cho bảng SanPham :



Giờ ta đã có 2 bảng ta cần liên kết khoá ngoại từ thuộc tính SanPham.MaDM tham chiếu đến DanhMuc.MaDM

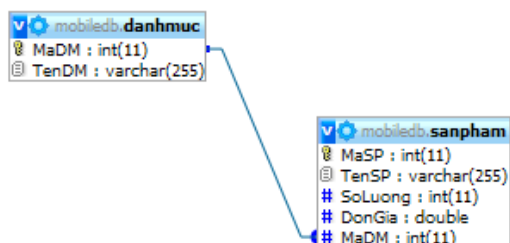
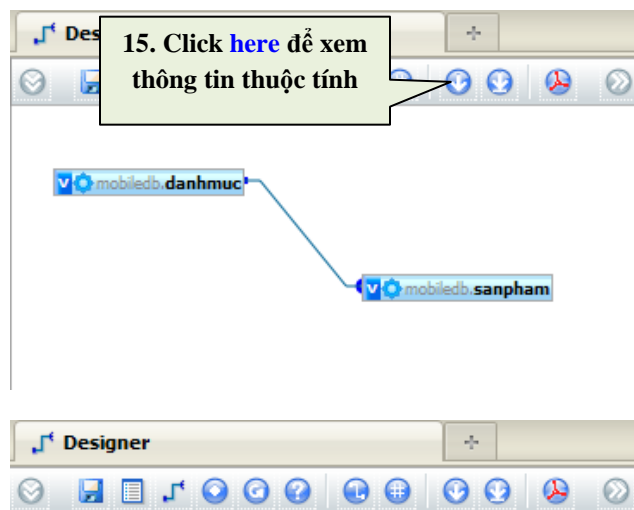
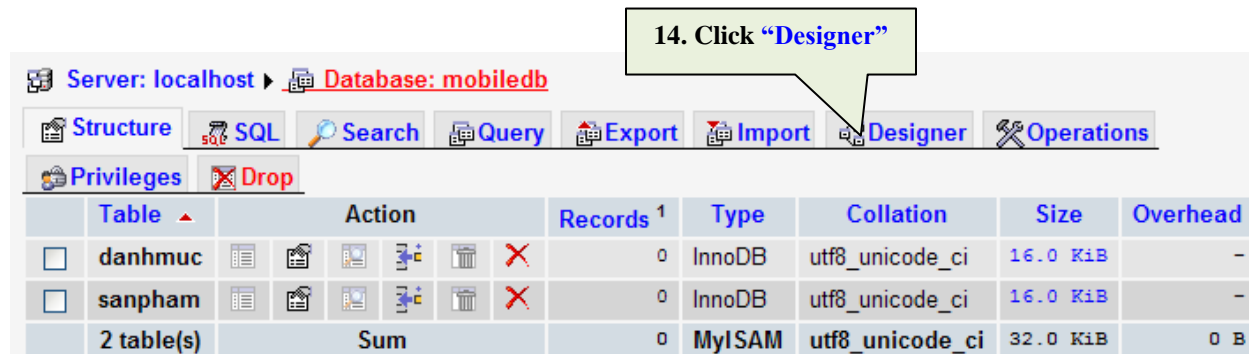
Tại trang của bảng SanPham :



Vậy là ta đã tạo khoá ngoại thành công từ SanPham.MaDM tham chiếu đến DanhMuc.MaDM.

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

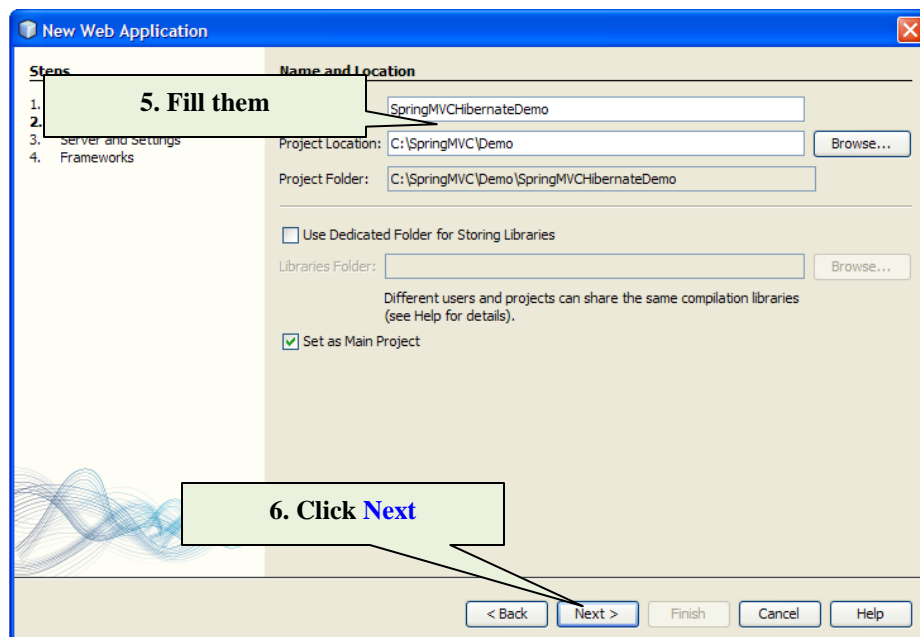
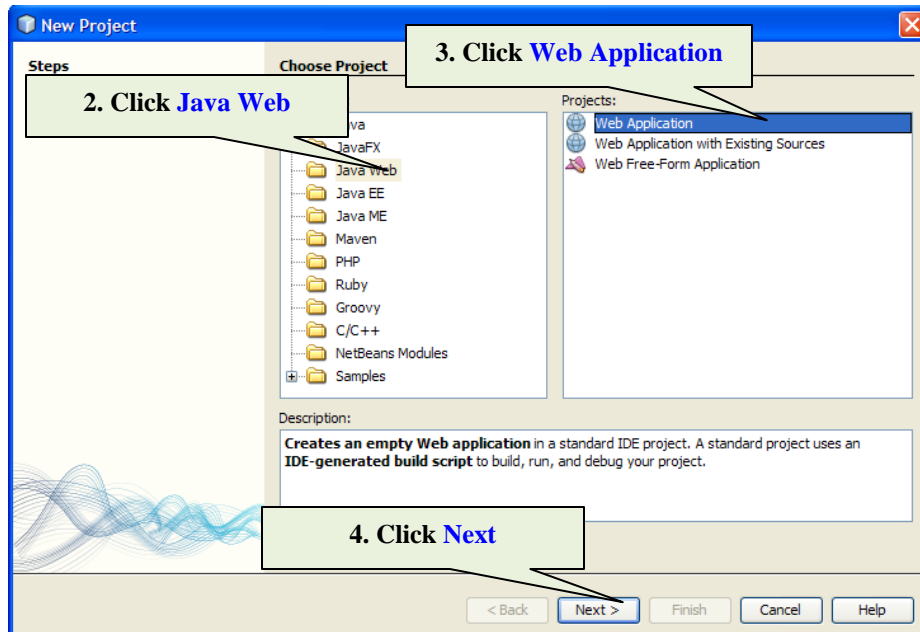
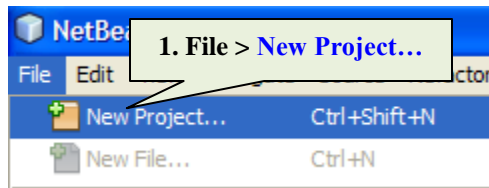
Để xem lược đồ CSDL để thấy quan hệ này ta click vào link **Database : mobiledb** để về trang quản lý Database :



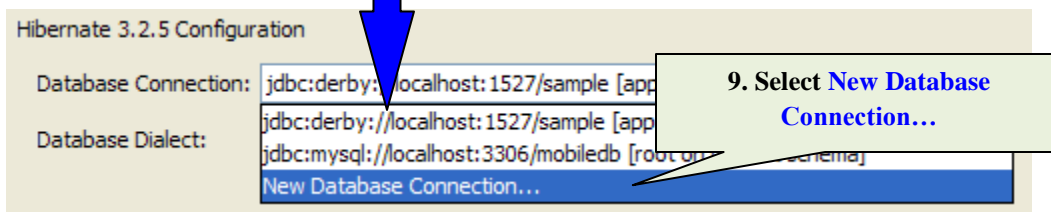
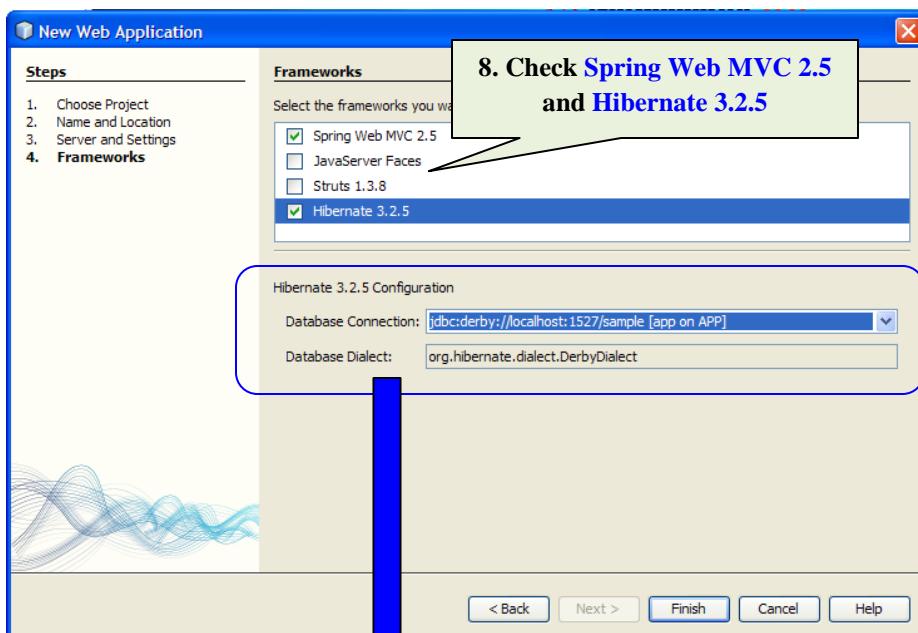
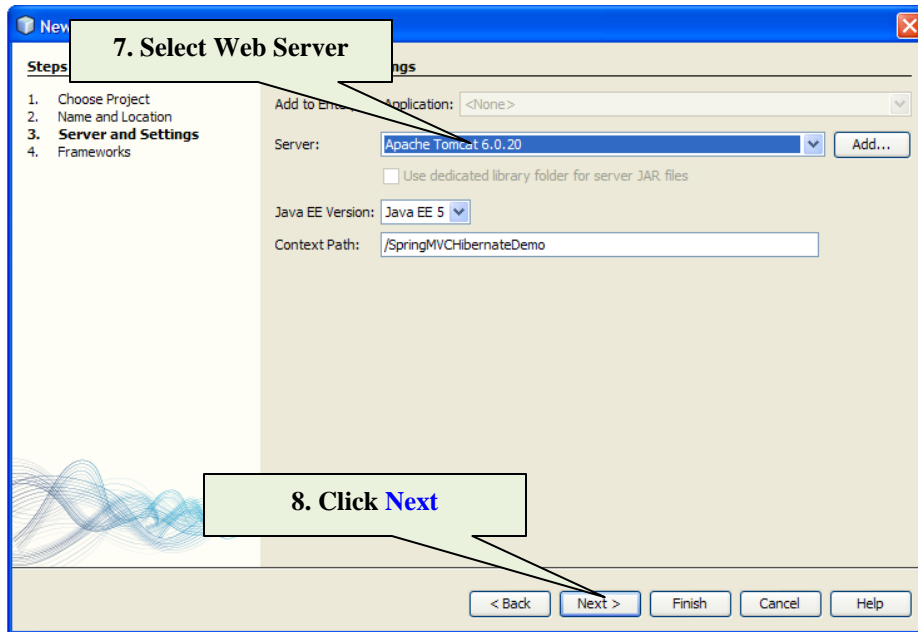
Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

### 3.2.2 Tạo Project áp dụng Spring Framework và Hibernate

- Mở NetBean IDE.
- Tạo Project



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

**New Database Connection**

Basic setting | Advanced

Data Input Mode: ☒ Field Entry ☐ Direct URL Entry

Driver Name: MySQL (Connector/J driver)

Host: localhost

Port: 3306

Database: mobiledb

User Name: root

Password:

Display Name (Optional):

☐ Remember password  
(see help for information on security risks)

Additional Props:

☒ Show JDBC URL

JDBC URL: jdbc:mysql://localhost:3306/mobiledb

10. Nhập các thông tin về connection như sau

11. Click OK

OK Cancel Help

**New Web Application**

Steps

1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks

Frameworks

Select the frameworks you want to use in your web application.

☒ Spring Web MVC 2.5

☐ JavaServer Faces

☐ Struts 1.3.8

☒ Hibernate 3.2.5

Hibernate 3.2.5 Configuration

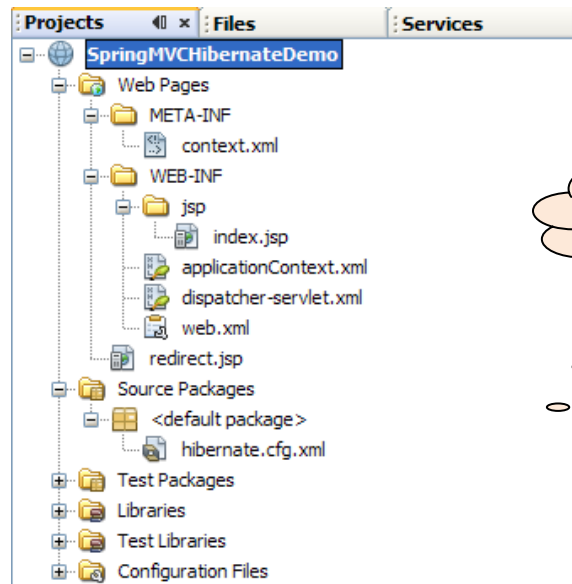
Database Connection: jdbc:mysql://localhost:3306/mobiledb [root on Default schema]

Database Dialect: org.hibernate.dialect.MySQLDialect

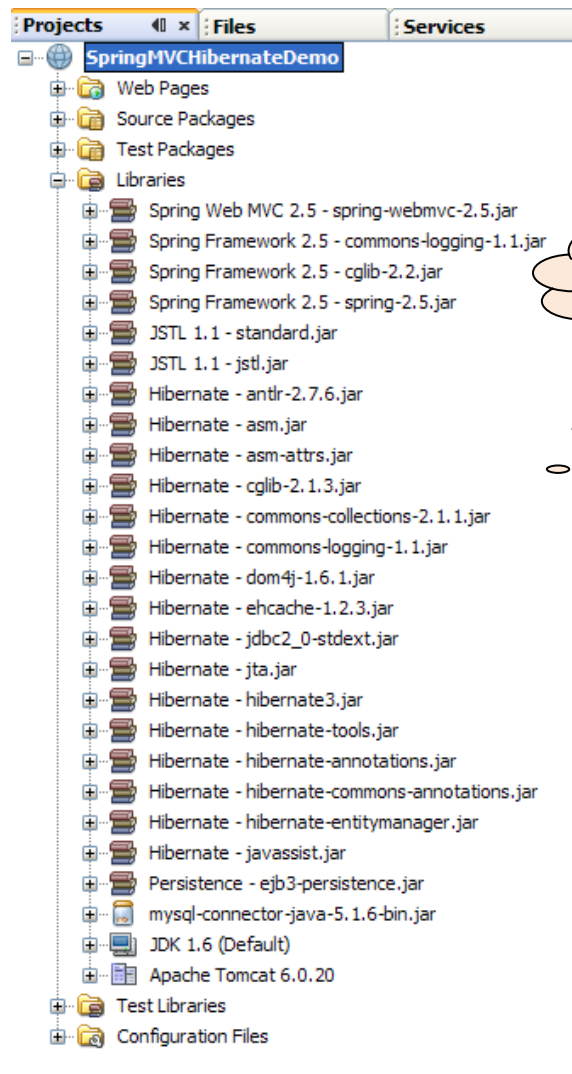
12. Click Finish

< Back Next > Finish Cancel Help

Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010



Project được tạo



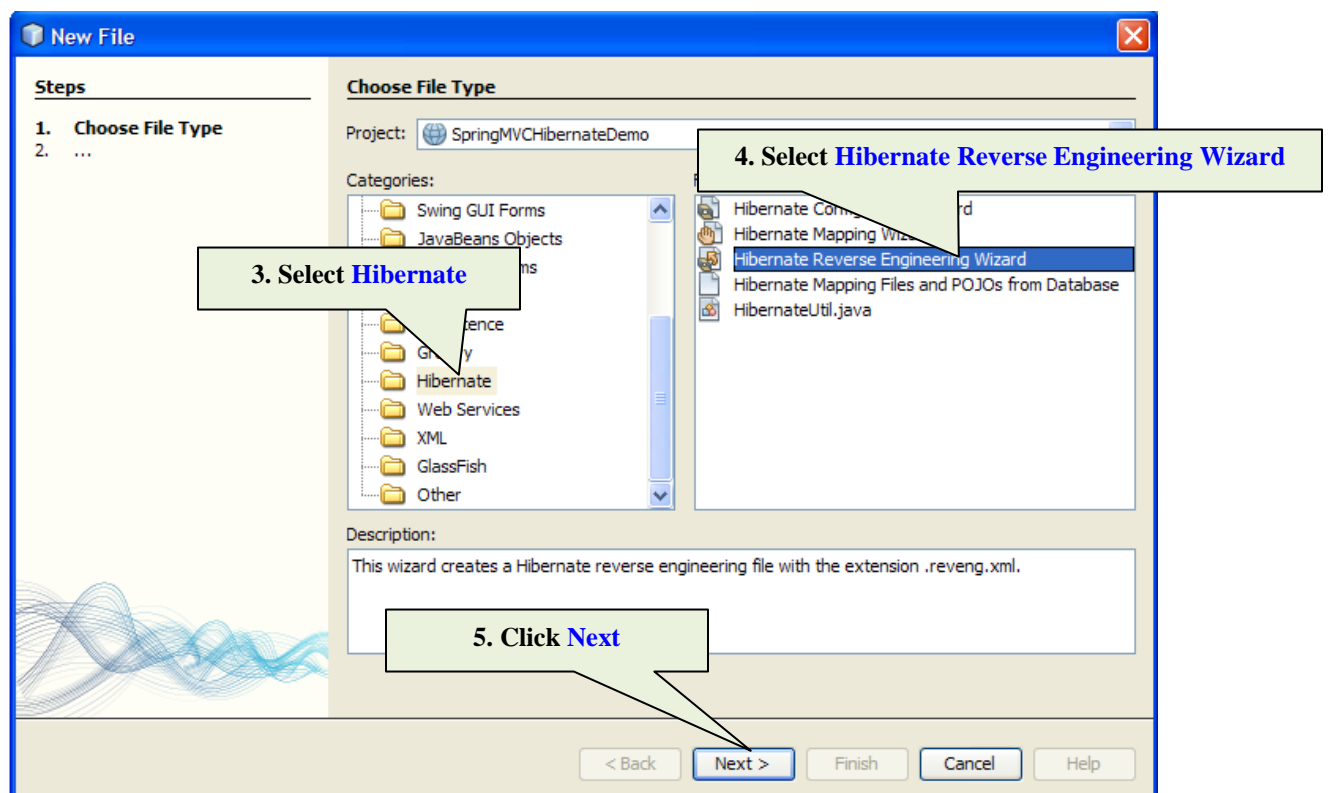
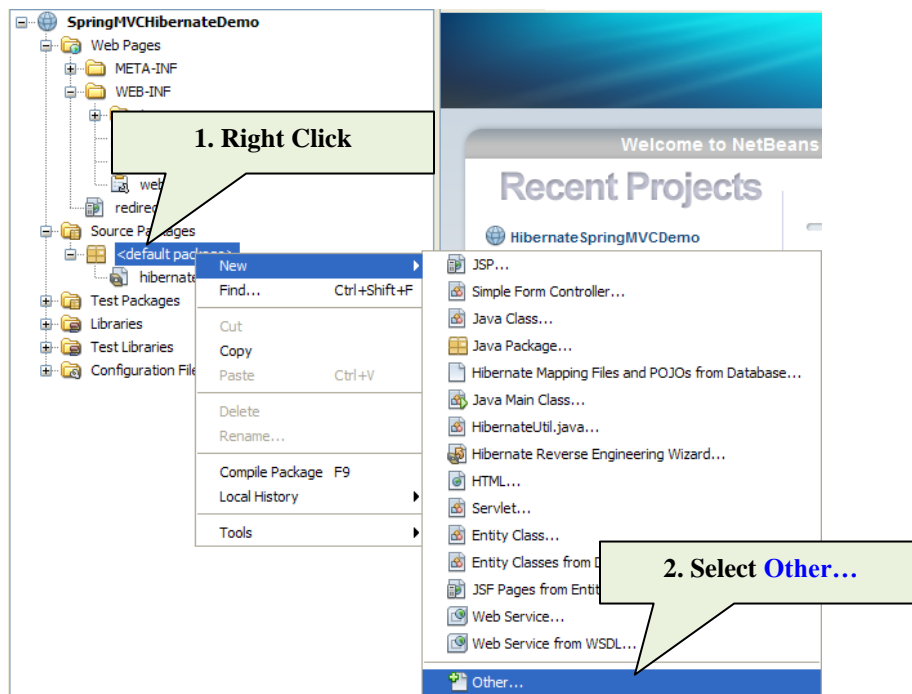
Các thư viện được include sử dụng

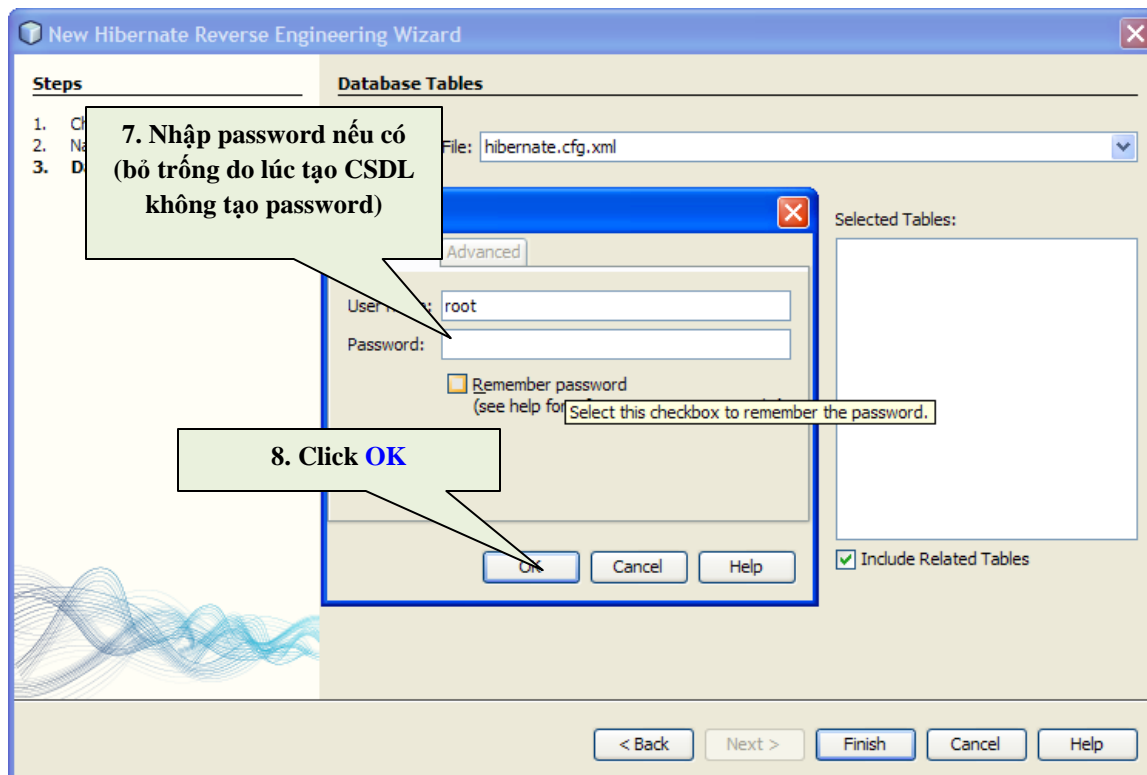
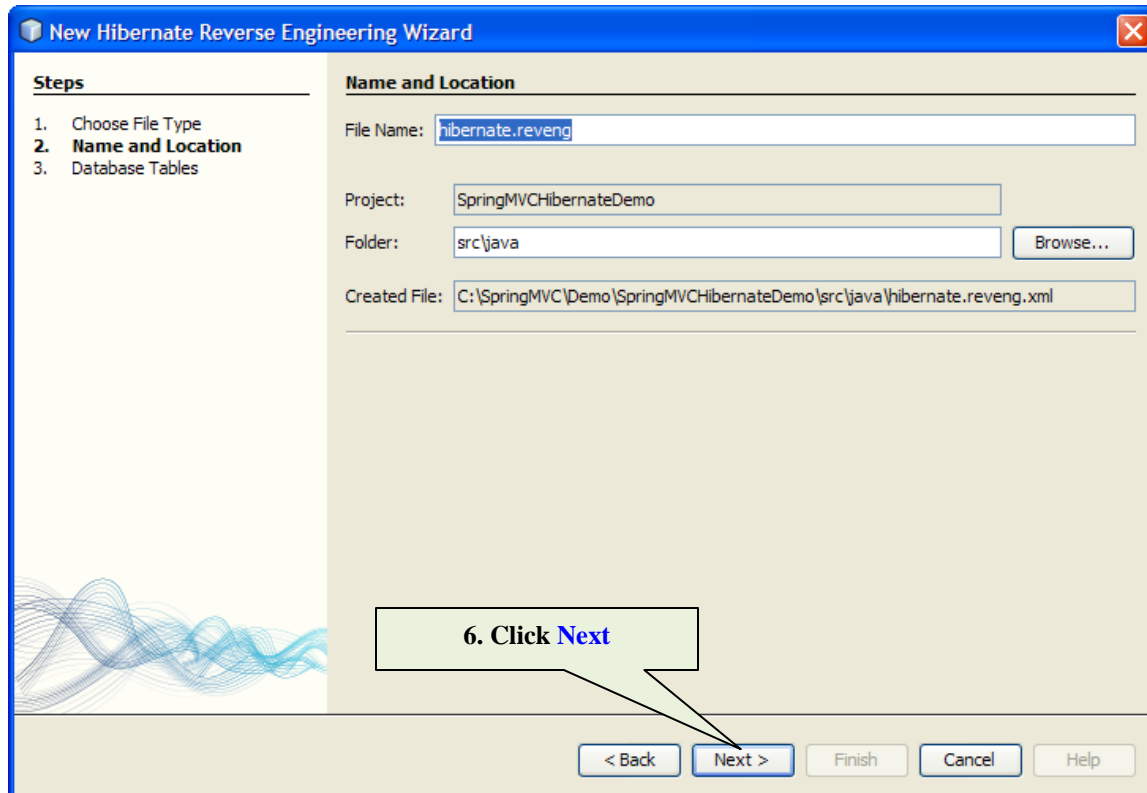


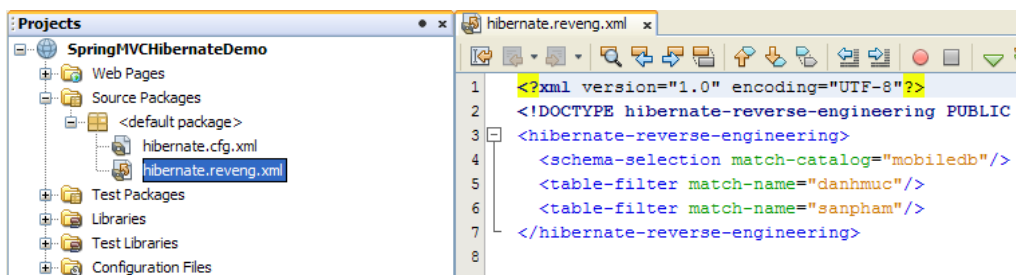
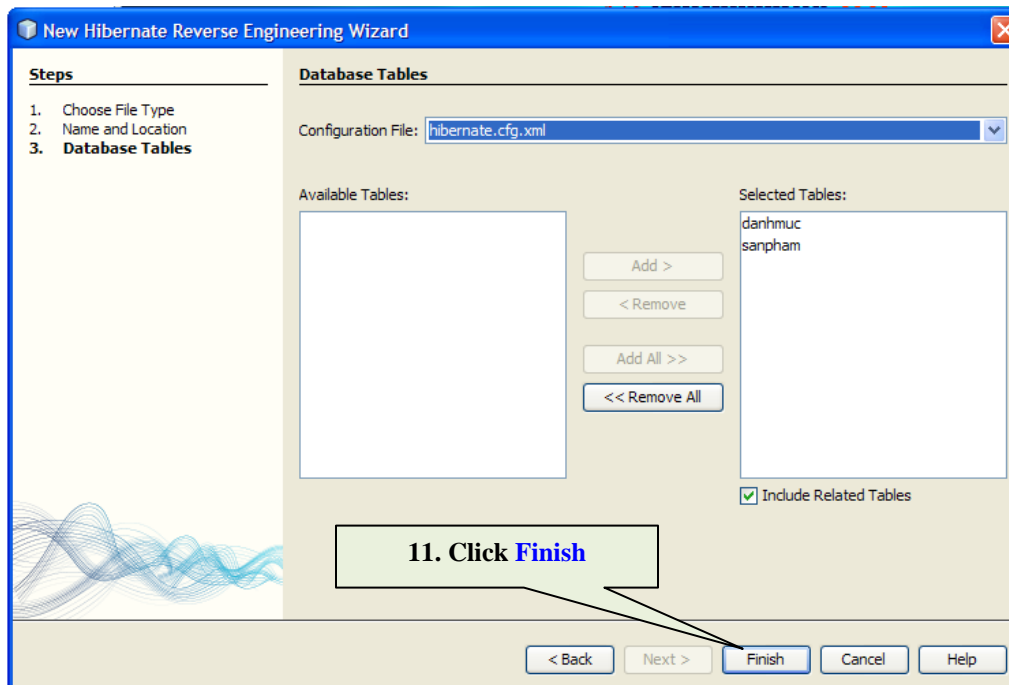
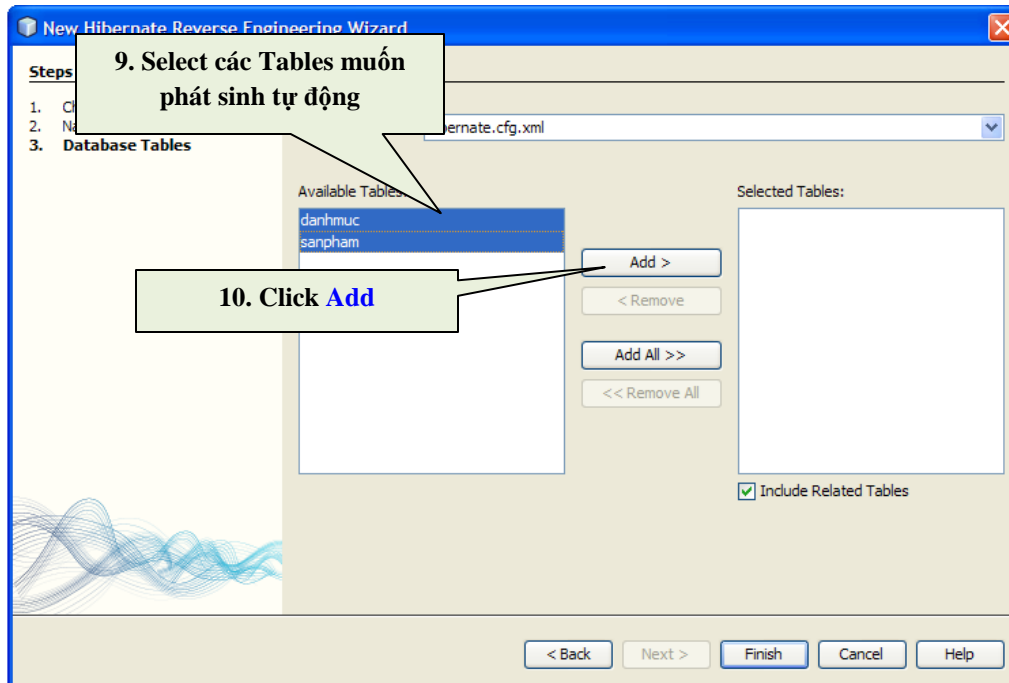
Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

### 3.2.3 Phát sinh các mapping file và pojos tương ứng cho Hibernate

Tạo file **Hibernate Reverse Engineering Wizard** :

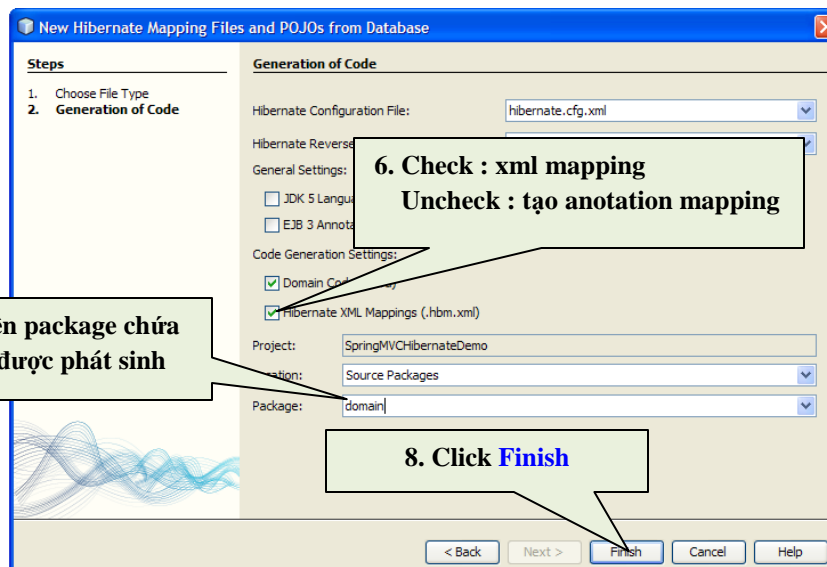
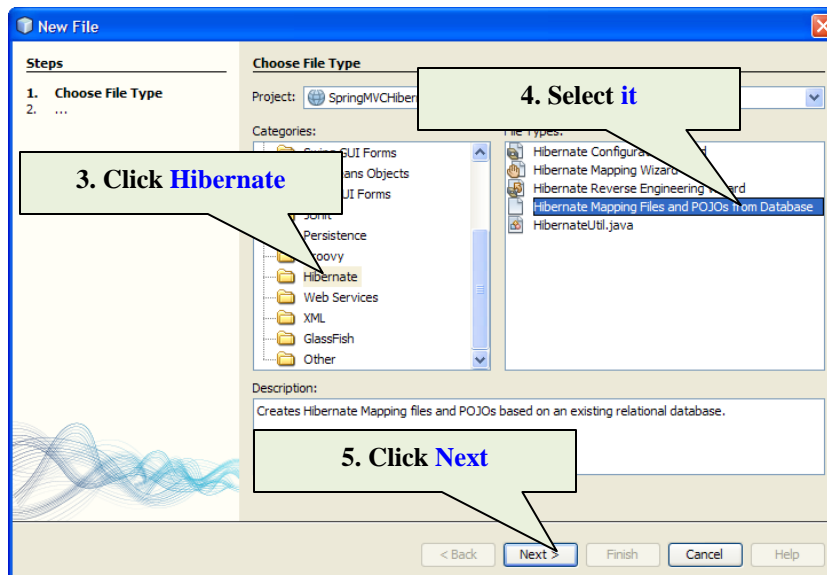
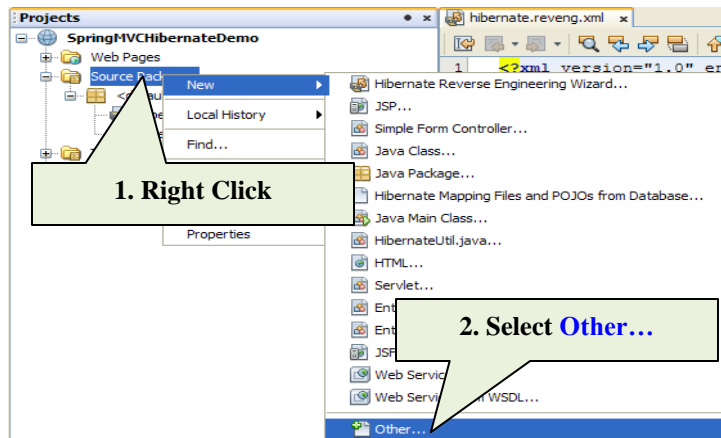




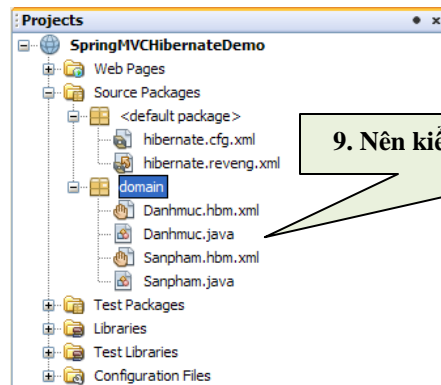


Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

Phát sinh các file pojo và hibernate mapping.



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



9. Nên kiểm tra lại để đảm bảo mapping đúng

File mapping cho DanhMuc

```

5 <hibernate-mapping>
6   <class name="domain.Danhmuc" table="danhmuc" catalog="mobiledb">
7     <id name="maDm" type="java.lang.Integer">
8       <column name="MaDM" />
9       <generator class="identity" />
10    </id>
11    <property name="tenDm" type="string">
12      <column name="TenDM" not-null="true" />
13    </property>
14    <set name="sanphams" inverse="true" cascade="all">
15      <key>
16        <column name="MaDM" not-null="true" />
17      </key>
18      <one-to-many class="domain.Sanpham" />
19    </set>
20  </class>
21 </hibernate-mapping>

```

10. Thêm attributes cần thiết

File mapping cho SanPham

```

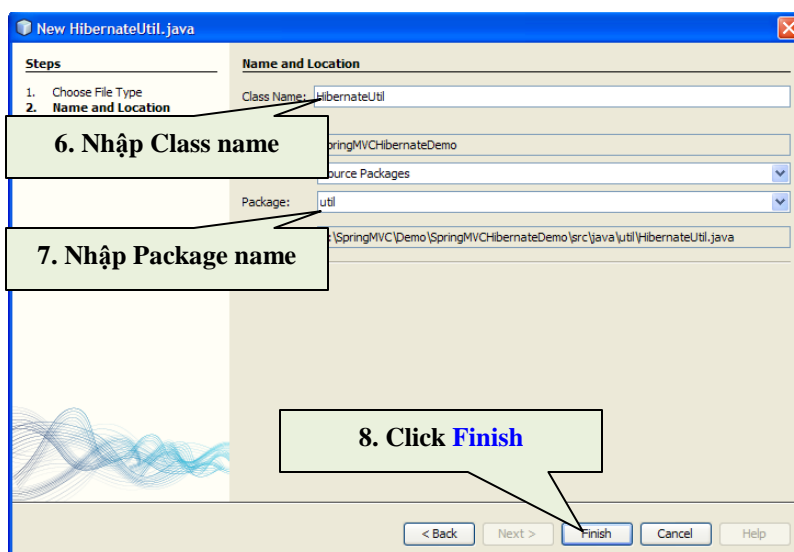
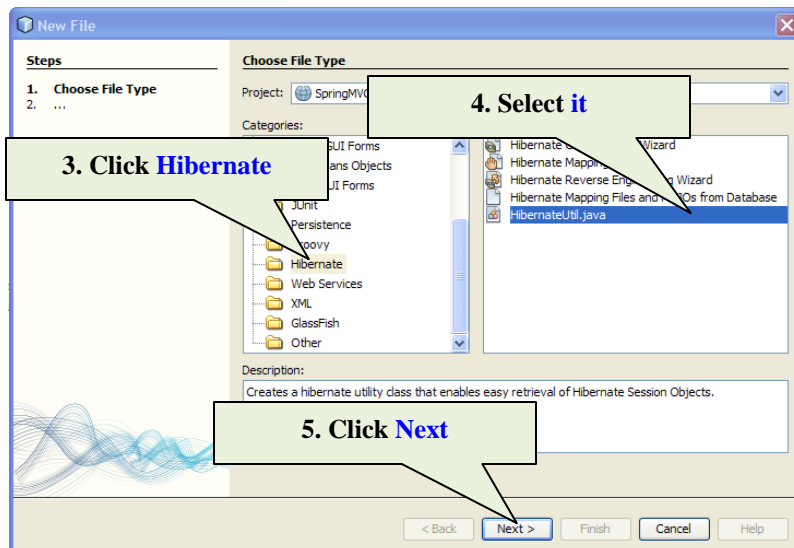
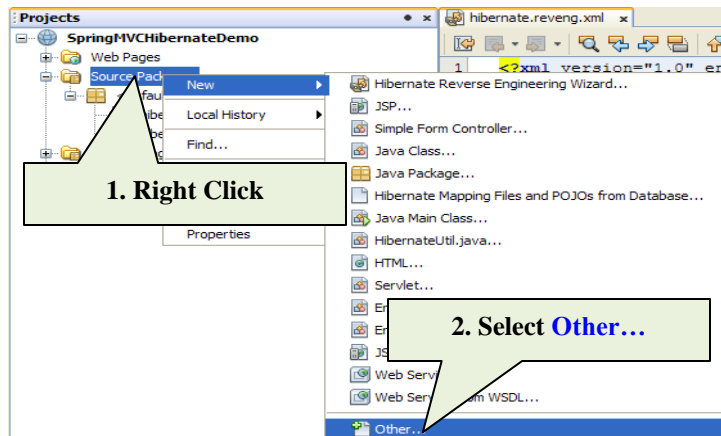
5 <hibernate-mapping>
6   <class name="domain.Sanpham" table="sanpham" catalog="mobiledb">
7     <id name="maSp" type="java.lang.Integer">
8       <column name="MaSP" />
9       <generator class="identity" />
10    </id>
11    <many-to-one name="danhmuc" class="domain.Danhmuc" fetch="select" cascade="all">
12      <column name="MaDM" not-null="true" />
13    </many-to-one>
14    <property name="tenSp" type="string">
15      <column name="TenSP" not-null="true" />
16    </property>
17    <property name="soLuong" type="int">
18      <column name="SoLuong" not-null="true" />
19    </property>
20    <property name="donGia" type="double">
21      <column name="DonGia" precision="22" scale="0" not-null="true" />
22    </property>
23  </class>
24 </hibernate-mapping>

```

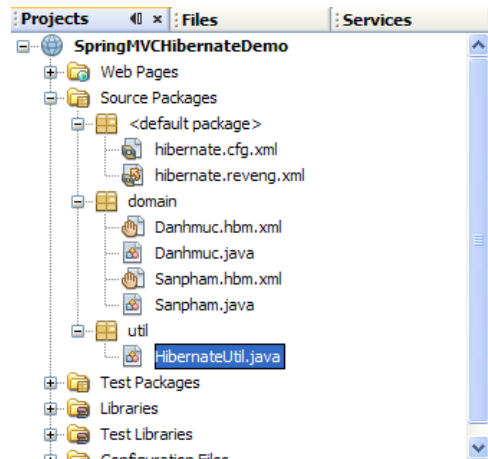
11. Thêm attributes cần thiết

Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

Tạo lớp HibernateUtil để quản lý session cho Hibernate.



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



Nội dung file này :

```
package util;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/**
 * Hibernate Utility class with a convenient static method that initializes and
 * returns the SessionFactory object.
 *
 * @author hoangnd
 */
public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            // config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

9. Do ta sử dụng XML mapping nên đổi  
AnnotationConfiguration thành Configuration

9. Do ta sử dụng XML mapping nên đổi  
AnnotationConfiguration thành Configuration

Vậy là ta đã phát sinh xong các lớp cần thiết để thao tác với Hibernate.

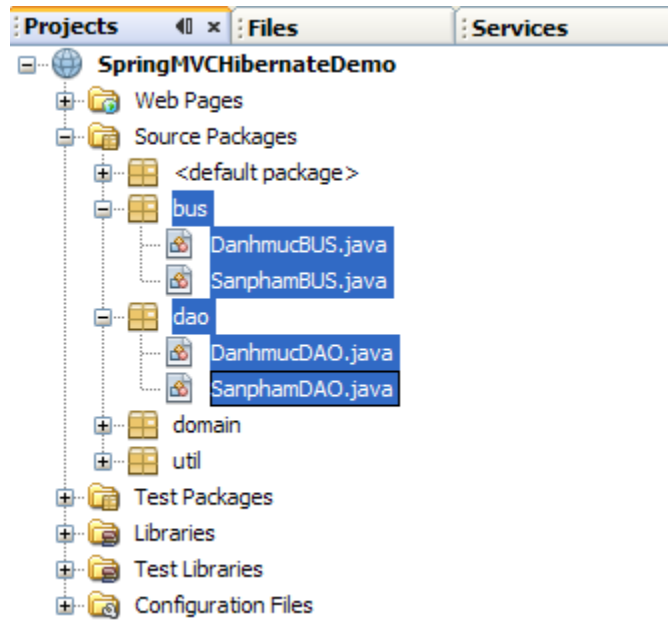
Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

### 3.2.4 Tạo các lớp DAO và BUS

Các lớp DAO : DanhmucDAO, SanphamDAO sử dụng Hibernate để truy xuất dữ liệu từ CSDL.

Các lớp BUS : DanhmucBUS, SanphamBUS chính là các Service cung cấp để xử lý nghiệp vụ sẽ được sử dụng bởi các Control trên tầng presentation theo SpringMVC framework.

Tạo các packages và files sau :



Với các chức năng cần cài đặt, trong DanhmucDAO ta chỉ cần cài đặt 2 phương thức là saveDanhMuc và listDanhMuc như sau :

```
public static Integer saveDanhMuc(Danhmuc danhMuc) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = null;
    Integer danhMucId = null;
    try {
        transaction = session.beginTransaction();

        danhMucId = (Integer) session.save(danhMuc);
        transaction.commit();
    } catch (HibernateException e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
    return danhMucId;
}
```

Mở session

Bắt đầu Transaction

Save đối tượng và lấy về ID của đối tượng mới thêm

Hoàn tất Transaction

Nếu có lỗi thì rollback

Đóng session



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

```

public static List<Danhmuc> listDanhMuc() {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = null;
    ArrayList<Danhmuc> lst = new ArrayList<Danhmuc>();
    try {
        transaction = session.beginTransaction();
        List lstDMs = session.createQuery("from Danhmuc").list();
        for (Iterator iterator = lstDMs.iterator(); iterator.hasNext();) {
            Danhmuc danhMuc = (Danhmuc) iterator.next();

            lst.add(danhMuc);
        }
        transaction.commit();
    } catch (HibernateException e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }

    return lst;
}

```

Mở session

Bắt đầu Transaction

Tạo và thực thi truy vấn

Lấy danh sách đối tượng

Hoàn tất Transaction

Nếu có lỗi thì rollback

Đóng session

Tương tự ta cài đặt 2 phương thức saveSanPham và listSanPham như sau :

```

public static Integer saveSanPham(Sanpham sanPham) {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = null;
    Integer sanPhamId = null;
    try {
        transaction = session.beginTransaction();

        sanPhamId = (Integer) session.save(sanPham);
        transaction.commit();
    } catch (HibernateException e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
    return sanPhamId;
}

```

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

```

public static List<Sanpham> listSanPham() {
    Session session = HibernateUtil.getSessionFactory().openSession();
    Transaction transaction = null;
    ArrayList<Sanpham> lst = new ArrayList<Sanpham>();
    try {
        transaction = session.beginTransaction();
        List lstSPs = session.createQuery("from SanPham").list();
        for (Iterator iterator = lstSPs.iterator(); iterator.hasNext();) {
            Sanpham sanPham = (Sanpham) iterator.next();

            lst.add(sanPham);
        }
        transaction.commit();
    } catch (HibernateException e) {
        transaction.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
    return lst;
}

```

Lớp SanphamBUS như sau :

```

public class SanphamBUS {
    public static Integer saveSanPham(Sanpham sanPham) {
        return SanphamDAO.saveSanPham(sanPham);
    }

    public static Integer saveSanPham(Danhmuc danhMuc, String tenSp,
        int soLuong, double donGia) {
        return SanphamDAO.saveSanPham(danhMuc, tenSp, soLuong, donGia);
    }

    public static List<Sanpham> listSanPham() {
        return SanphamDAO.listSanPham();
    }

    public static void updateSanPham(Integer sanPhamId, DanhMuc danhMuc,
        String tenSp, int soLuong, double donGia) {
        SanphamDAO.updateSanPham(sanPhamId, danhMuc, tenSp, soLuong, donGia);
    }

    public static void deleteSanPham(Integer sanPhamId) {
        SanphamDAO.deleteSanPham(sanPhamId);
    }
}

```

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Lớp DanhMucBUS như sau :

```
public class DanhMucBUS {
    public static Integer saveDanhMuc(DanhMuc danhMuc) {
        return DanhMucDAO.saveDanhMuc(danhMuc);
    }

    public static Integer saveDanhMuc(String tenDm) {
        return DanhMucDAO.saveDanhMuc(tenDm);
    }

    public static Integer saveDanhMuc(String tenDm, Set sanphams) {
        return DanhMucDAO.saveDanhMuc(tenDm, sanphams);
    }

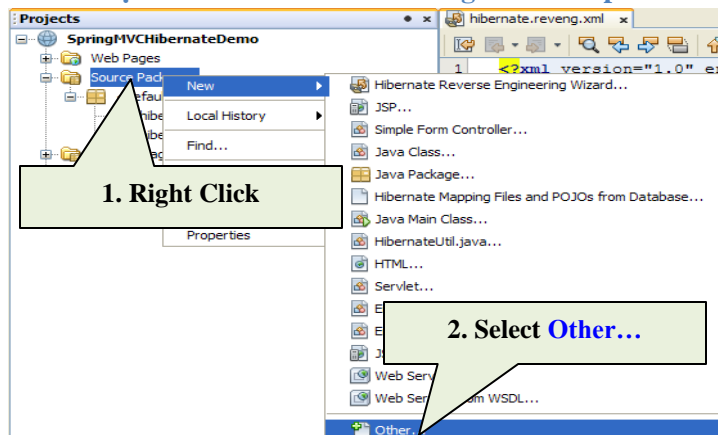
    public static List<DanhMuc> listDanhMuc() {
        return DanhMucDAO.listDanhMuc();
    }

    public static void updateDanhMuc(Integer sanPhamId, String tenDm) {
        DanhMucDAO.updateDanhMuc(sanPhamId, tenDm);
    }

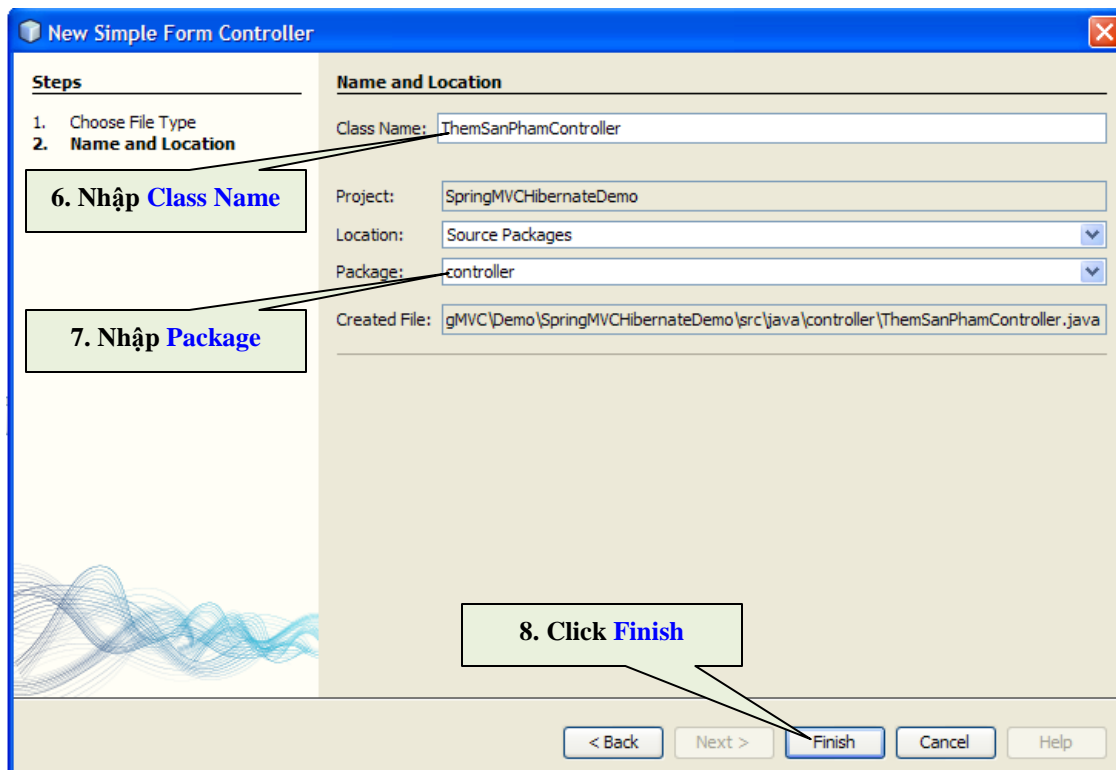
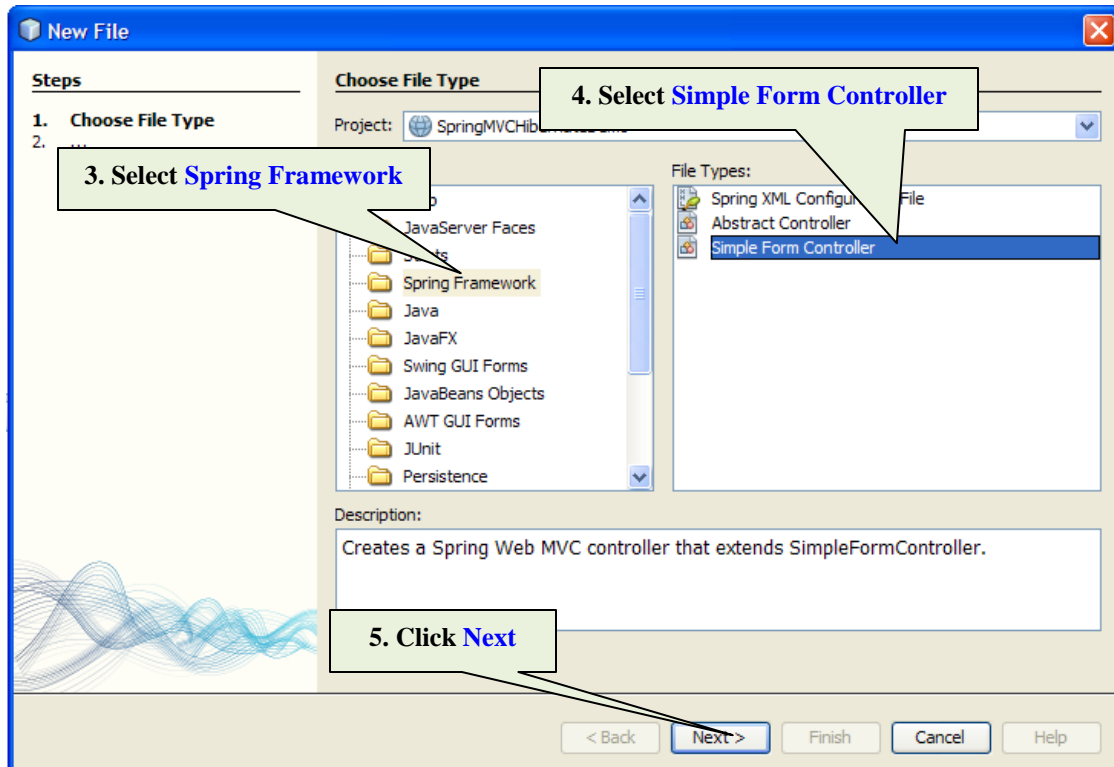
    public static void updateDanhMuc(Integer sanPhamId,
        String tenDm, Set sanphams) {
        DanhMucDAO.updateDanhMuc(sanPhamId, tenDm, sanphams);
    }

    public static void deleteDanhMuc(Integer sanPhamId) {
        DanhMucDAO.deleteDanhMuc(sanPhamId);
    }
}
```

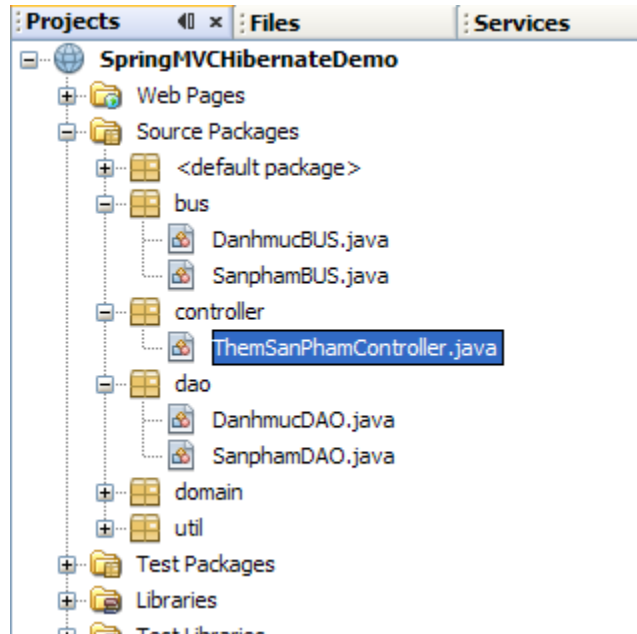
### 3.2.5 Tạo Controller cho chức năng thêm sản phẩm mới :



Spring MVC Framwork	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010



Edit lại constructor của class như sau :

```
public ThemSanPhamController() {
    //Initialize controller properties here
    //in the Web Application Context

    setCommandClass(Sanpham.class);
    setCommandName("sanpham");
    setSuccessView("ThemSanPham_SuccessView");
    setFormView("ThemSanPham_InputView");
}
```

Khai báo lớp được dùng để lấy command về từ FormView để xử lý khi submit

Tên command, tên này sẽ được dùng để kết buộc các thuộc tính của CommandClass với các control input

Đặt tên của View sẽ hiển thị trước khi submit

Đặt tên của View sẽ hiển thị khi submit thành công

Thêm 2 thuộc tính kiểu DanhmucBUS và SanphamBUS đóng vai trò là 2 services để lớp Controller này sử dụng để xử lý các nghiệp vụ trong quá trình xử lý :

```
private DanhmucBUS danhmucBUS;
private SanphamBUS sanphamBUS;

public void setDanhmucBUS(DanhmucBUS danhmucBUS) {
    this.danhmucBUS = danhmucBUS;
}

public void setSanphamBUS(SanphamBUS sanphamBUS) {
    this.sanphamBUS = sanphamBUS;
}
```

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Override lại phương thức showForm để add một danh sách các Danh mục cho người dùng chọn khi nhập loại sản phẩm. Phương thức này được gọi khi form được hiển thị ban đầu.

```

@Override
protected ModelAndView showForm(HttpServletRequest request,
    HttpServletResponse response,
    BindException errors) throws Exception {
    ModelAndView mv = super.showForm(request, response, errors);

    List<Danhmuc> lst = danhmucBUS.listDanhMuc();

    mv.addObject("lstDM", lst);

    return mv;
}

```

Lưu lại đối tượng ModelAndView trả về từ các lớp cha

Lấy danh sách danh mục

Add thêm đối tượng danh sách danh mục vào ModelAndView kết quả

Trả về

Xoá phương thức doSubmitAction được cài đặt sẵn. Bỏ comment và edit lại phương thức onSubmit như sau :

```

@Override
protected ModelAndView onSubmit(
    HttpServletRequest request,
    HttpServletResponse response,
    Object command,
    BindException errors) throws Exception {
    ModelAndView mv = new ModelAndView(getSuccessView());
    //Do something...
    Sanpham sp = (Sanpham)command;

    int id = Integer.parseInt(request.getParameter("MaDM"));
    sp.setDanhmuc(danhmucBUS.getDanhMucById(id));

    sanphamBUS.saveSanPham(sp);
    mv.addObject("sanpham", sp);

    return mv;
}

```

Tạo ModelAndView sẽ trả về khi submit thành công

Ép kiểu command thành Sanpham để lấy kết quả được commit

Lưu sản phẩm mới vào CSDL

Trả về

Lấy mã Danh mục được chọn, sau đó tạo và set thuộc tính danh mục cho sản phẩm được nhập. Tại sao lại phải lấy riêng như vậy sẽ được giải thích ở phần sau

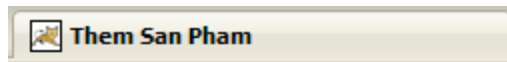
Đưa đối tượng sản phẩm mới nhập thành công cho viewSuccess hiển thị

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010


### 3.2.6 Tạo Views :

Ta sẽ tạo 2 Views sau :

- View để nhập thông tin của sản phẩm mới :

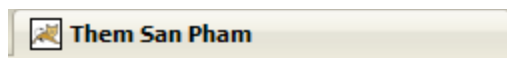


## Nhap san pham :

Ten SP:   
 So luong:   
 Don gia:   
 Danh muc:  

**Danh sách danh mục hiện có trong CSDL**

- View để hiển thị thông tin sản phẩm mới nhập nếu nhập thành công :

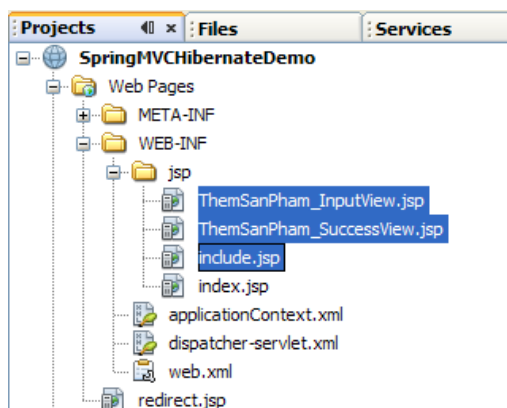


## San pham Details

MaSP : 3  
 TenSP : N72  
 So Luong : 3  
 Don Gia : 5000000.0  
 Danh muc : Nokia

Nhận vào một đối tượng sanpham và hiển thị giá trị các thuộc tính

Ta tạo 3 file sau :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

- File **include.jsp** để load các thư viện tag cần dùng

```
<%@ page session="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
```

- File **ThemSanPham\_SuccessView.jsp** để hiển thị kết quả nếu commit thành công

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Them San Pham</title>
</head>
<body>
<h2>San pham Details</h2>
<hr>
MaSP      : ${sanpham.maSp} <br/>
TenSP     : ${sanpham.tenSp} <br/>
So Luong  : ${sanpham.soLuong} <br/>
Don Gia   : ${sanpham.donGia} <br/>
Danh muc  : ${sanpham.danhmuc.tenDm} <br/>
</body>
</html>
```

**sanpham** là object được truyền vào trong câu lệnh  
*mv.addObject("sanpham", sp);*  
trong phương thức **onSubmit** của **ThemSanPhamController**

- File **ThemSanPham\_InputView.jsp**

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Them San Pham</title>
<script language="javascript" type="text/javascript">
function isNumeric(evt) {...}
</script>
</head>
```

Hàm javascript xử lý các textbox chỉ nhập số



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

```

<body>
  <h1>Nhap san pham : </h1>

  <form:form method="POST" commandName="sanpham">
    <table>
      <tr>
        <td><b>Ten SP:</b></td>
        <td><form:input path="tenSp" id="TenSP" maxlength="100"/></td>
      </tr>
      <tr>
        <td><b>So luong:</b></td>
        <td><form:input path="soLuong" id="SoLuong" maxlength="5"
          onkeypress="return isNumeric(event)"/></td>
      </tr>
      <tr>
        <td><b>Don gia:</b></td>
        <td><form:input path="donGia" id="DonGia" maxlength="10"
          onkeypress="return isNumeric(event)"/></td>
      </tr>
      <tr>
        <td><b>Danh muc:</b></td>
        <td>
          <select name="MaDM">
            <c:forEach var="danhmuc" items="${lstDM}" >
              <option value="<c:out value="{danhmuc.maDm}"></c:out>">
                <c:out value="${danhmuc.maDm}"></c:out>
              </option>
            </c:forEach>
          </select>
        </td>
      </tr>
      <tr>
        <td colspan="2" align="center"><input type="submit" value="OK"></td>
      </tr>
    </table>
  </form:form>
</body>

```

Tag hỗ trợ bởi Spring Form

Chỉ định commandName của object lưu dữ liệu sẽ được commit

Bind thuộc tính của lớp Command với giá trị sẽ được nhập vào input

ID của control select trong form

Vòng lặp add các phần tử của danh sách danh mục có trong CSDL vào select control

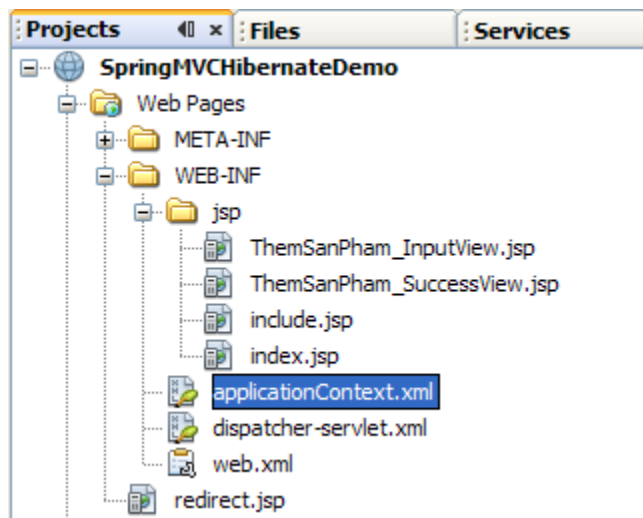
lstDM là object được truyền vào trong câu lệnh `mv.addObject("lstDM", lst);` trong phương thức `showForm` của `ThemSanPhamController`

**Ghi chú :** Trả lời câu hỏi tại sao phải gọi phương thức `request.getParameter` trong phương thức `onSubmit` để lấy giá trị mã danh mục được chọn bởi người dùng để tạo và gán thuộc tính `Sanpham.danhmuc` chứ không lấy sẵn trong `Object command`. Do tag `select` cung cấp bởi **Spring Form** khi trả về sẽ trả về giá trị kiểu `String` của mục được chọn, nên ta không thể kết nối giá trị của control này với thuộc tính `Sanpham.danhmuc` kiểu class `Danhmuc` được. Do đó ta phải submit riêng lẻ giá trị mã danh mục được chọn bởi người dùng và sẽ xử lý request để lấy, sau đó tạo và set thuộc tính `Sanpham.danhmuc` kiểu `Danhmuc` trong hàm `onSubmit`.

Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

### 3.2.7 Config Controller :

Đầu tiên ta khai báo các service sẽ được sử dụng trong file **applicationContext.xml**



Thêm 2 dòng sau vào trong file này :

```
<?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:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx" >

    <!--bean id="propertyConfigurer"
         class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
         p:location="/WEB-INF/jdbc.properties" />

    <bean id="dataSource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource"
          p:driverClassName="${jdbc.driverClassName}"
            p:url="${jdbc.url}"
            p:username="${jdbc.username}"
            p:password="${jdbc.password}" />

    <!-- ADDPersistenceSupport HERE (jpa, hibernate, etc) -->

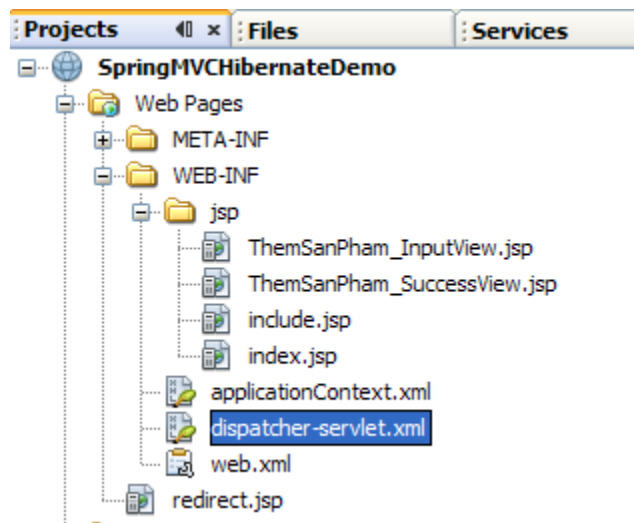
    <bean id="sanphamService" class="bus.SanphamBUS" />
    <bean id="danhmucService" class="bus.DanhmucBUS" />
</beans>
```

Khai báo 1 Bean có id là **sanphamService**

Khai báo 1 Bean thuộc class **bus.SanphamBUS**

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Trong file **dispatcher-servlet.xml**



Thêm **bean** này vào trong thân **bean** gốc. Đây là bean khai báo cho **ThemSanPhamController** của chúng ta.

```
<bean name="ThemSanPham"
      class="controller.ThemSanPhamController"
      p:sanphamBUS-ref="sanphamService"
      p:danhmucBUS-ref="danhmucService" />
```

Gán 2 **services** khai báo ở trên cho 2 thuộc tính của class **ThemSanPhamController** là **sanphamBUS** và **danhmucBUS**

Ta thêm dòng mapping sau để map ThemSanPhamController của chúng ta với url "ThemSanPham.htm" :

```
<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="index.htm">indexController</prop>
      <prop key="ThemSanPham.htm">ThemSanPham</prop>
    </props>
  </property>
</bean>
```

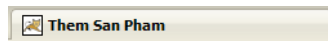
Map với **name** của Bean khai báo cho Controller của chúng ta ở trên.

Cuối cùng ta vào file **redirect.jsp** và sửa **index.htm** thành **ThemSanPham.htm**

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<% response.sendRedirect("ThemSanPham.htm"); %>
```

<a href="#">Spring MVC Framework</a>	Phiên bản: <a href="#">3.0</a>
Báo cáo tìm hiểu	Ngày: <a href="#">15/05/2010</a>

Vậy là ta đã cấu hình xong cho phần Spring MVC Framework để xử lý phần giao diện. Bây giờ ta có thể build và run chương trình. Nếu thành công kết quả sẽ như sau :



## Nhap san pham :

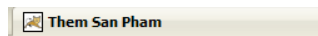
Ten SP:

So luong:

Don gia:

Danh muc:  ▼

Nhap thông tin :



## Nhap san pham :

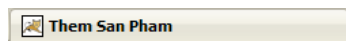
Ten SP:

So luong:

Don gia:

Danh muc:  ▼

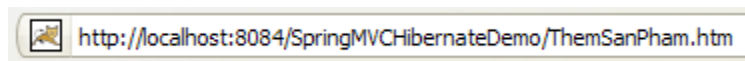
Click OK ta được kết quả như sau :



## San pham Details

MaSP : 5  
 TenSP : N70  
 So Luong : 10  
 Don Gia : 4500000.0  
 Danh muc : Nokia

Để ý có một điểm đặc biệt khi sử dụng Spring MVC Framework đó là ở chỗ url. Cho dù thật sự trong ứng dụng ta có 2 file jsp. Nhưng url luôn là url mà được map với controller tương ứng :



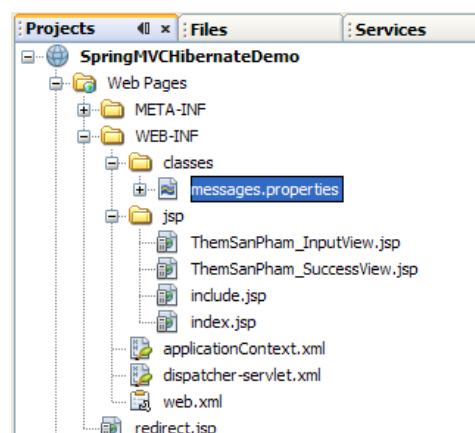
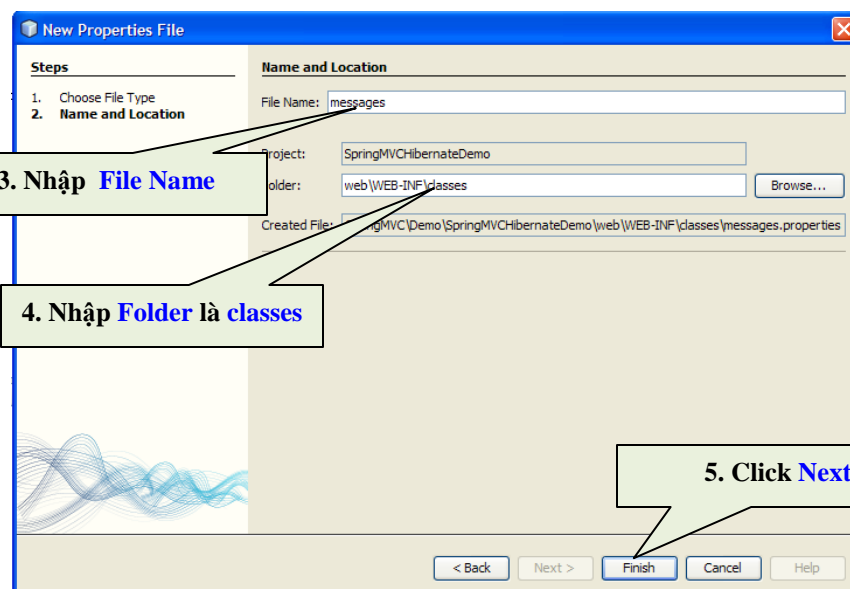
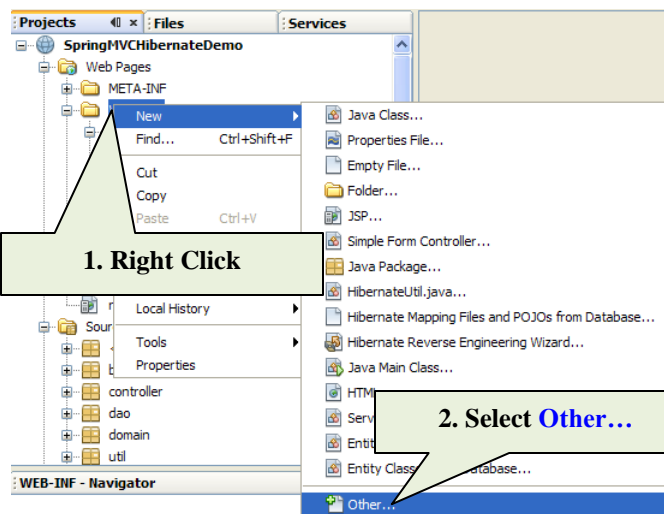
Điều này giúp che dấu được nên công nghệ bên dưới và tính an ninh hơn vì framework đã hỗ trợ quản lý trang thành việc truy cập trực tiếp vào url của trang trái phép. Đây cũng là một ưu điểm của Spring MVC Framework.

Phần kế tiếp ta sẽ bàn về việc Validation dữ liệu nhập của người dùng sử dụng Spring MVC Framework.

Spring MVC Framework	Phiên bản: 3.0
Bảo cáo tìm hiểu	Ngày: 15/05/2010

### 3.2.8 Tạo Validation :

Validator dùng để kiểm tra tính hợp lệ của input từ người dùng. Để cài đặt các input này ta làm như sau :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

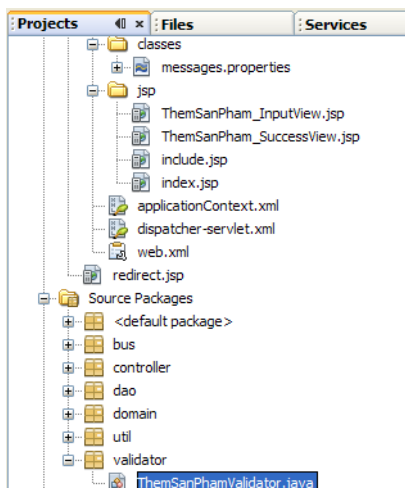
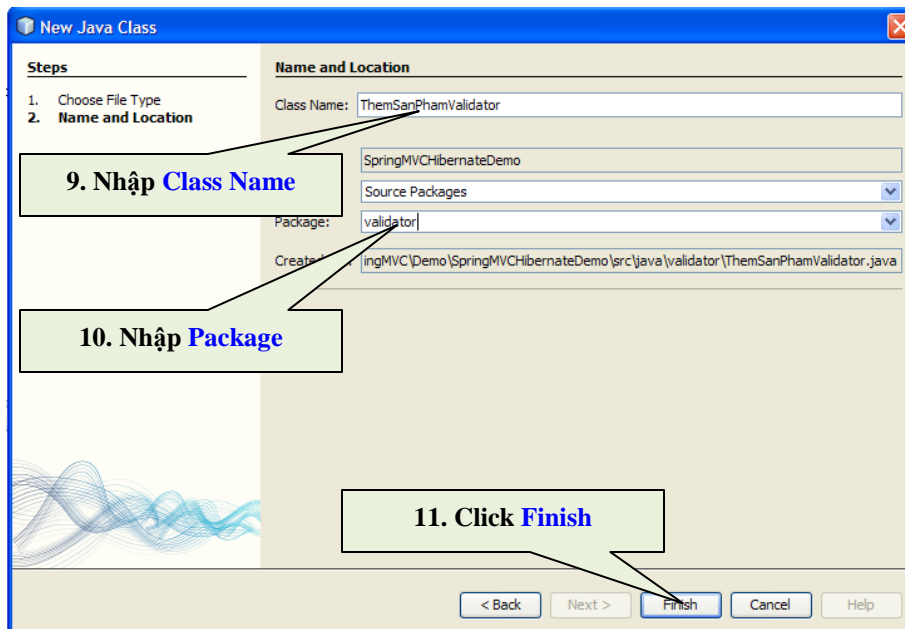
Trong file này tạo các properties sau :

```
TenSP.required=TenSP is invalid
SoLuong.required=SoLuong is invalid
DonGia.required=DonGia is invalid
```

Trong file **dispatcher-servlet.xml** khai báo thêm **bean** sau :

```
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basename">
        <value>messages</value>
    </property>
</bean>
```

Tạo một class mới :



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

Nội dung class **ThemSanPhamValidator** như sau :

```
package validator;

import domain.*;

import org.springframework.validation.Errors;
import org.springframework.validation.ValidationUtils;
import org.springframework.validation.Validator;

public class ThemSanPhamValidator implements Validator {

    public boolean supports(Class type) {
        return Sanpham.class.isAssignableFrom(type);
    }

    public void validate(Object o, Errors errors) {
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "tenSp", "TenSP.required");

        Sanpham sp = (Sanpham) o;
        if (sp.getSoLuong() <= 0) {
            errors.rejectValue("soLuong", "SoLuong.required");
        }
        if (sp.getDonGia() <= 0) {
            errors.rejectValue("donGia", "DonGia.required");
        }
    }
}
```

**Implement Validator**

**Override lại phương thức supports**

**Nếu type là class Sanpham thì support**

**Override lại phương thức validate để xử lý việc input validate**

**Nếu là chỉ có khoảng trống hoặc không nhập gì thì báo lỗi**

**Xuất lỗi tại tag form:errors với path = "tenSp" và xuất chuỗi thông báo lưu trong thuộc tính TenSP.required trong messages.properties**

**Đây là phương thức yêu cầu loại bỏ kết quả và thông báo lỗi. Các tham số như mô tả ở trên**

Trong file ThemSanPham\_InputView.jsp :

- Thêm định nghĩa css sau :

```
<title>Them San Pham</title>
<style type="text/css">
    .even {
        background-color: silver;
    }
    .error {
        color: #ff0000;
        font-style: italic;
    }
</style>
```

Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

- Add các tag **form:errors** : Tag này dùng để hiển thị thông báo lỗi nếu có.

```

<tr>
    <td><b>Ten SP:</b></td>
    <td><form:input path="tenSp" id="TenSP" maxlength="100"/></td>
    <td><form:errors path="tenSp" cssClass="error"/></td>
</tr>
<tr>
    <td><b>So luong:</b></td>
    <td><form:input path="soLuong" id="SoLuong" maxlength="5"
        onkeypress="return isNumeric(event)"/></td>
    <td><form:errors path="soLuong" cssClass="error"/></td>
</tr>
<tr>
    <td><b>Don gia:</b></td>
    <td><form:input path="donGia" id="DonGia" maxlength="10"
        onkeypress="return isNumeric(event)"/></td>
    <td><form:errors path="donGia" cssClass="error"/></td>
</tr>

```

Trong file **applicationContext.xml** thêm dòng khai báo cho **validator** mới cài đặt.

```

<bean id="sanphamService" class="bus.SanphamBUS" />
<bean id="danhmucService" class="bus.DanhmucBUS" />
<bean id="themsanphamValidator" class="validator.ThemSanPhamValidator" />

```

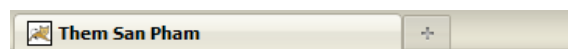
Trong file **dispatcher-servlet.xml** gán thêm thuộc tính **validator** cho **controller** :

```

<bean name="ThemSanPham"
    class="controller.ThemSanPhamController"
    p:sanphamBUS-ref="sanphamService"
    p:danhmucBUS-ref="danhmucService"
    p:validator-ref="themsanphamValidator"/>

```

Vậy là xong. Giờ build và run ứng dụng và để kết quả không hợp lệ và nhấn OK. Ta được kết quả như sau



## Nhap san pham :

Ten SP:  *TenSP is invalid*

So luong:

Don gia:  *DonGia is invalid*

Danh muc:



Spring MVC Framework	Phiên bản: 3.0
Báo cáo tìm hiểu	Ngày: 15/05/2010

## 4 Các ưu điểm và khuyết điểm của Spring MVC Web Framework :

### 4.1 Ưu điểm :

Spring là một framework Java mạnh mẽ được sử dụng trong những ứng dụng Java có phạm vi lớn. Nó cung cấp những dịch vụ Enterprise đến những Plain Old Java Objects (POJOs). Cơ chế IoC giúp ứng dụng đạt được sự đơn giản hoá và tăng khả năng kiểm tra lỗi.

Spring MVC cung cấp một sự phân chia rất rõ ràng, rành mạch giữa những Controller, Java Bean models và Views.

Spring MVC rất linh hoạt, toàn bộ Spring MVC được xây dựng dựa trên những interfaces. Mọi phần của Spring MVC framework được cấu hình thông qua việc lắp ghép những interface, class tiện ích sẵn có, và thậm chí được tạo bởi người dùng.

Spring không chỉ sử dụng công nghệ JSP mà còn có thể dễ dàng tích hợp các công nghệ view khác như Velocity, XSLT, FreeMarker, XL, ...

Cung cấp cơ chế che dấu nền công nghệ bên dưới, trang web khi hiển thị chỉ có extension là .htm, không thể biết được bên dưới ta sử dụng công nghệ, kỹ thuật gì, JSP hay Velocity, XLST, ... thậm chí là là những công nghệ view được custom bởi người dùng.

Spring Controller được cấu hình thông qua IoC như mọi đối tượng khác. Điều này làm chúng dễ dàng được test, và được tích hợp dễ dàng với những đối tượng khác được quản lý bởi Spring.

Kết buộc trực tiếp các input từ view với những đối tượng domains.

### 4.2 Khuyết điểm :

Cấu hình phức tạp và cồng kềnh => không phát huy được sức mạnh khi áp dụng cho các ứng dụng quy mô nhỏ mà có thể ngược lại còn làm cho ứng dụng phức tạp.

## 5 Tài liệu tham khảo :

<http://www.vaannila.com/spring/spring-tutorial/spring-tutorial.html>

[http://en.wikipedia.org/wiki/Spring\\_Framework](http://en.wikipedia.org/wiki/Spring_Framework)

<http://netbeans.org/kb/docs/web/quickstart-webapps-spring.html>

<http://sites.google.com/site/springmvcnetbeans/step-by-step/>

<http://forum.springsource.org/showthread.php?t=16553>