

BÀI TẬP TUẦN 05_06_07_

Chương 5: Spring MVC

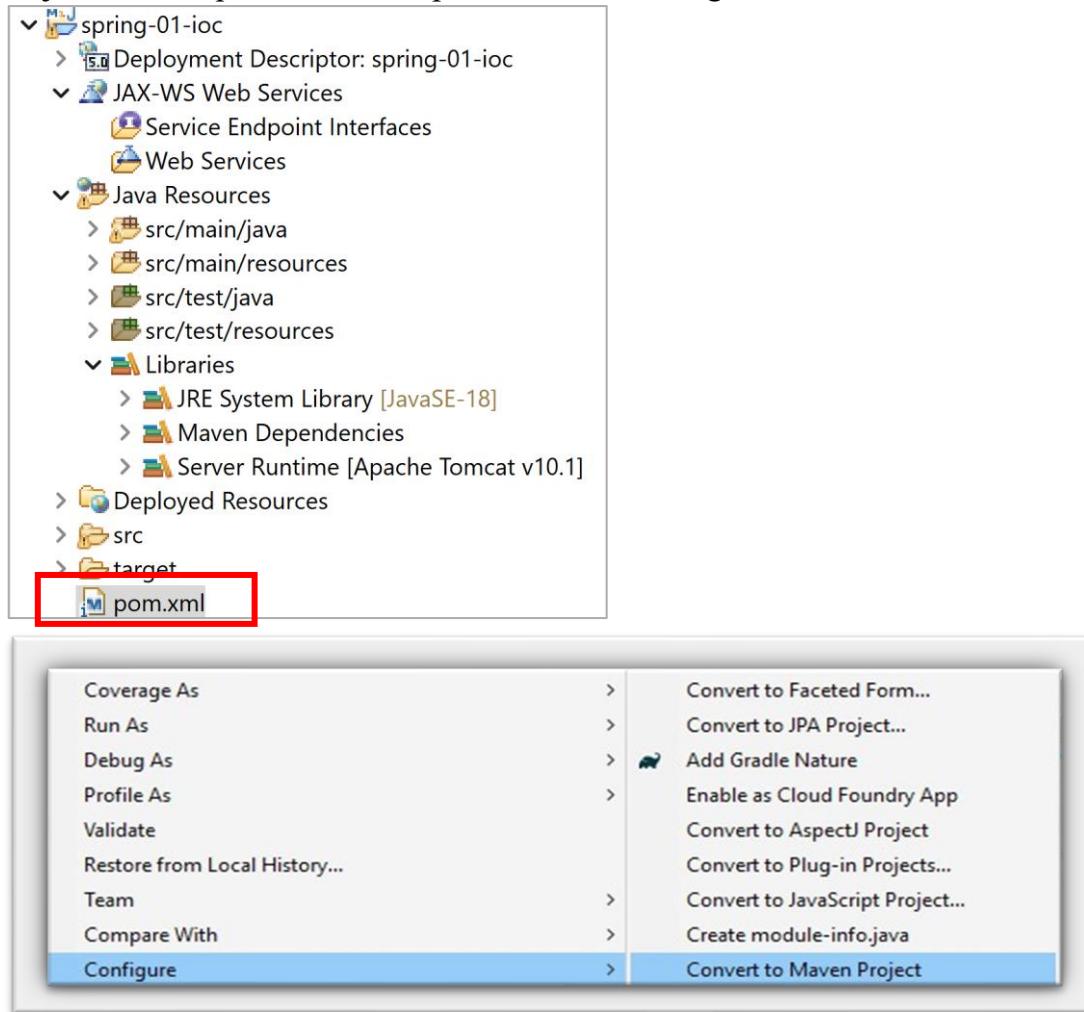
Phân 1. Spring Core

Lưu ý:

Sử dụng Maven Project

- Các dependencies tra cứu trong <https://mvnrepository.com/>

Project của Eclipse nhấn chuột phải → Chọn Configure → Chọn Convert to Maven Project



Các thay đổi về dependencies sẽ thay đổi trong file pom.xml

Các dependencies tra cứu trong <https://mvnrepository.com/>, khi tra cứu lưu ý các groupId, artifactId, version cần dùng.

```

<project xmlns="https://maven.apache.org/POM/4.0.0"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>iuh.fit.se</groupId>
  <artifactId>spring-demo-one</artifactId>
  <version>1.0</version>

  <dependencies>
    <!--
      https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-jar-plugin -->
    <dependency>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.3.0</version>
    </dependency>

    <!--
      https://mvnrepository.com/artifact/org.springframework/spring-context-support -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context-support</artifactId>
      <version>6.1.12</version>
    </dependency>

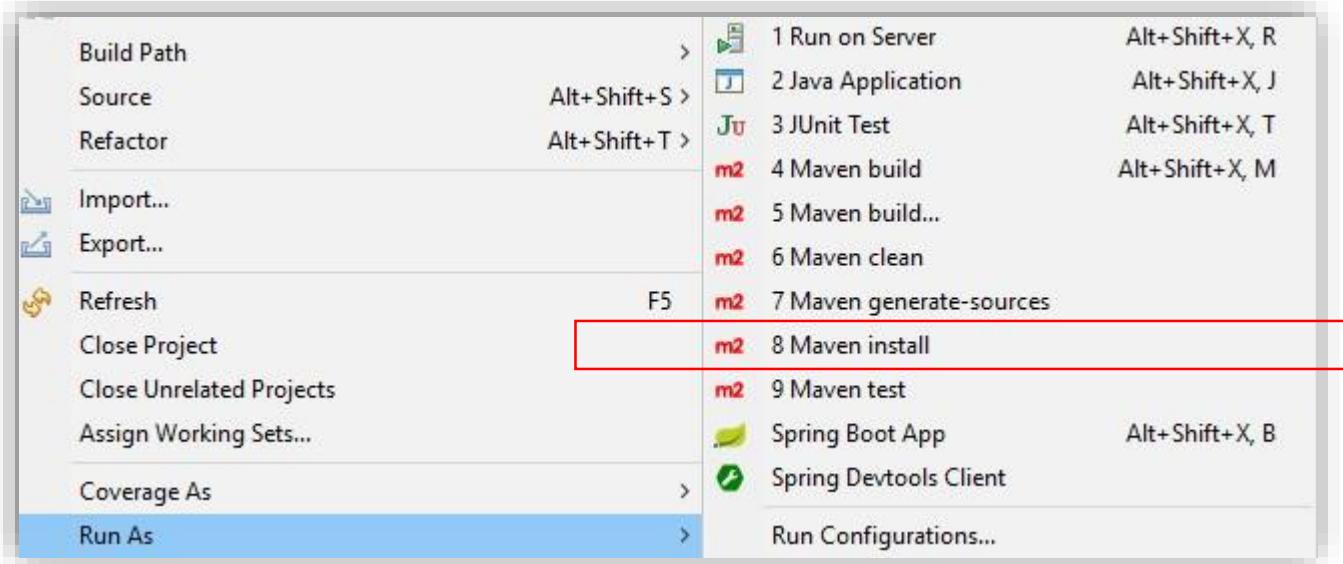
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring</artifactId>
      <version>5.2.18.RELEASE</version>
      <type>pom</type>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>6.1.12</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>6.1.12</version>
    </dependency>
  </dependencies>
</project>

```

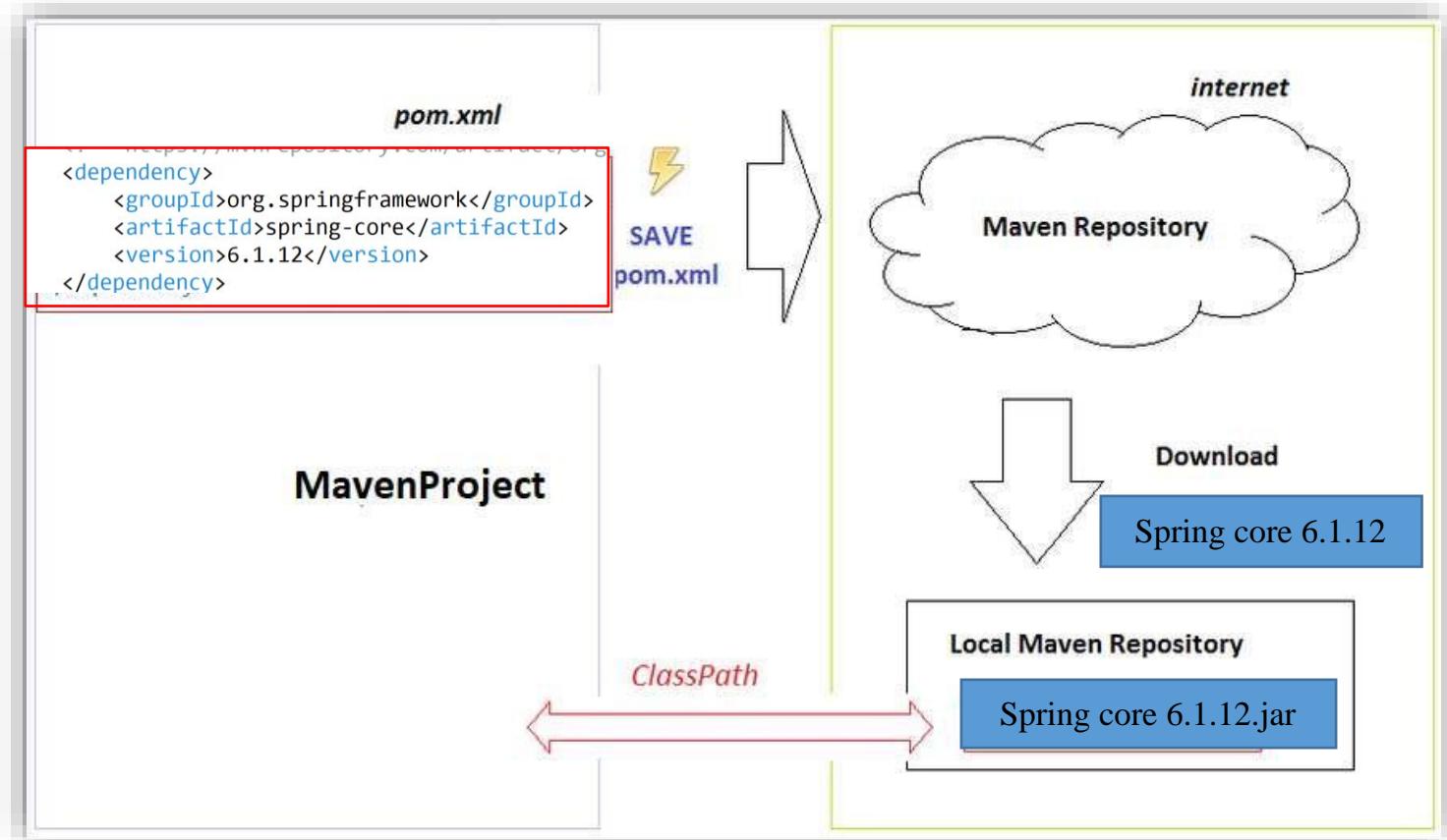
Các Maven dependencies sau khi cập nhật:



Tạo file jar từ Maven: **SpringMVCMultiLanguages0.0.1-SNAPSHOT.jar**



Nguyên tắc hoạt động của Maven



- Khai báo trên tập tin **pom.xml** các dependencies phụ thuộc vào các thư viện có **version** rõ ràng.
- Khi lưu file pom.xml, Maven kiểm tra các thư viện này đã tồn tại ở máy tính (Local Repository) chưa. Nếu chưa Maven sẽ download về từ kho dữ liệu từ Internet (Repository).
- Maven sẽ tự động khai báo CLASSPATH cho Project tới vị trí file jar được download.

IoC Container trong Spring là core của Spring Framework. IoC Container, IoC tạo ra các đối tượng, nối các đối tượng lại với nhau, cấu hình, và quản lý life cycle (từ khi tạo ra đến khi bị hủy). IoC Container sử dụng DI để quản lý các thành phần tạo nên một ứng dụng. Các đối tượng này được gọi là Spring Bean.

- IoC Container được cung cấp thông tin từ các tập tin XML.
- Hai loại IoC containers:

1. BeanFactory

BeanFactory cung cấp các chức năng cơ bản trong khi ApplicationContext cung cấp các tính năng mở rộng hơn cho các ứng dụng Spring nên ApplicationContext phù hợp với các ứng dụng J2EE hơn. Lưu ý ApplicationContext bao gồm tất cả các chức năng của BeanFactory.

```
Resource resource=new ClassPathResource("Beans.xml");
BeanFactory factory=new XmlBeanFactory(resource);
Student obj =(Student)factory.getBean("studentbean");
obj.displayInfo();
```

2. ApplicationContext

- ApplicationContext (*org.springframework.context.ApplicationContext* interface) cũng tương tự như BeanFactory (*org.springframework.beans.factory.BeanFactory*), sử dụng để mô tả Spring Container. ApplicationContext được xây dựng trên interface BeanFactory.
- Đối tượng của interface ApplicationContext như là một framework của Spring để gọi đến đối tượng cần lấy bằng cách sử dụng phương thức getBean().

```
ApplicationContext context = new
ClassPathXmlApplicationContext("Beans.xml");
Student obj = (Student) context.getBean("studentbean");
obj.displayInfo();
```

DI - Dependency Injection:

- Các module không giao tiếp trực tiếp với nhau, mà thông qua interface. Module cấp thấp sẽ implement interface, module cấp cao sẽ gọi module cấp thấp.

Để giao tiếp với database, sử dụng interface *IDatabase*, các module cấp thấp là *XMLDatabase*, *SQLDatabase*. Module cấp cao là *CustomerBusiness* sẽ sử dụng interface *IDatabase*.

- Việc khởi tạo các module cấp thấp sẽ do DI Container thực hiện.

Trong module *CustomerBusiness*, không khởi tạo *IDatabase db = new XMLDatabase()*, việc này sẽ do DI Container thực hiện. Module *CustomerBusiness* sẽ không biết gì về module *XMLDatabase* hay *SQLDatabase*.

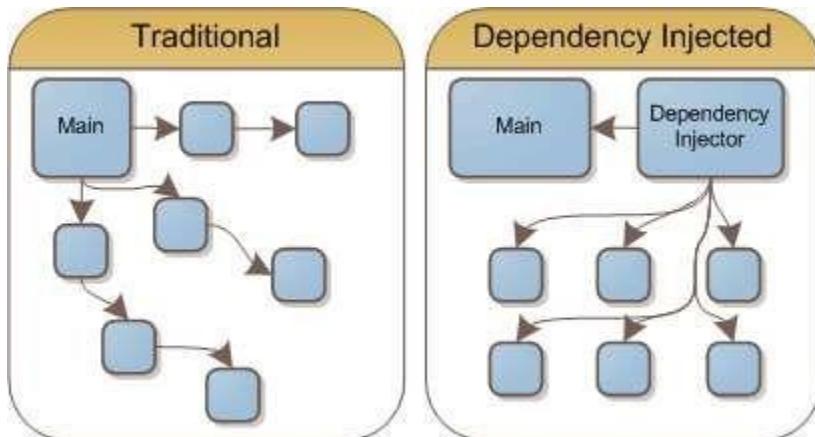
- Các Module kết hợp với các interface cụ thể sẽ được config trong code hoặc trong file XML.
- DI được dùng để làm giảm sự phụ thuộc giữa các module, dễ dàng trong việc thay đổi module, bảo trì code và testing.

Thực hiện Dependency Injection theo 3 cách:

1. **Constructor Injection:** Các dependency sẽ được container truyền/inject vào 1 class thông qua constructor của class.
2. **Setter Injection:** Các dependency sẽ được truyền vào 1 class thông qua các hàm setter.
3. **Interface Injection:** Class cần truyền/inject sẽ implement 1 interface. Interface này có chứa 1 hàm tên *Inject*. Container sẽ injection dependency vào 1 class thông qua việc gọi hàm *Inject* của interface.

Bài 1. Spring DI

IoC trong Spring và phương pháp tiếp cận truyền thống.



Một nhân viên (Employee) trong công ty sẽ có các thông tin như họ tên, địa chỉ Thiết kế đối tượng Employee và Address dùng Dependency Injection (DI) trong Spring để loại bỏ sự phụ thuộc giữa các mã chương trình.

B1. Tạo Project Java

B2. Configure Project → Convert to Maven Project

B3. Cấu hình các dependency của Spring trong pom.xml

```
<dependencies>
    <!--
        https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-jar-plugin -->
    <dependency>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.3.0</version>
    </dependency>

    <!--
        https://mvnrepository.com/artifact/org.springframework/spring-context-support -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>6.1.12</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring</artifactId>
        <version>5.2.18.RELEASE</version>
        <type>pom</type>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
        <version>6.1.12</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>6.1.12</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>6.1.12</version>
    </dependency>

</dependencies>
```

B4. Tạo lớp Address

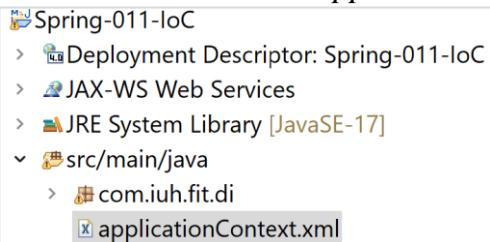
```
public class Address {  
    private String city;  
    private String state;  
    private String country;  
    public Address(String city, String state, String country) {  
        super();  
        this.city = city;  
        this.state = state;  
        this.country = country;  
    }  
    @Override  
    public String toString() {  
        return "Address [city=" + city + ", state=" + state + ", country=" +  
country + "]";  
    }  
}
```

B5. Tạo lớp Employee. Làm giảm sự phụ thuộc giữa Employee và Address, tạo lớp Address riêng và truyền/inject vào Employee thông qua Constructor (hoặc phương thức Setter).

Dùng Contructor

```
2  
3 public class Employee {  
4  
5     private int id;  
6     private String name;  
7     private Address address;// Aggregation [Employee, Address]  
8     public Employee() {  
9         super();  
10        // TODO Auto-generated constructor stub  
11    }  
12    public Employee(int id, String name, Address address) {  
13        super();  
14        this.id = id;  
15        this.name = name;  
16        this.address = address;  
17    }  
18    public Address getAddress() {  
19        return address;  
20    }  
21    public void setAddress(Address address) {  
22        this.address = address;  
23    }  
24    public void show() {  
25        System.out.println(id + " " + name);  
26        System.out.println(address.toString());  
27    }  
28}  
29 }
```

B6. Cấu hình file bean *applicationContext.xml*. File lưu trong thư mục **src/main/java**



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

    <bean id="s1" class="com.iuh.fit.di.Student">
        <constructor-arg value="1101" />
        <constructor-arg value="Thu" />
        <constructor-arg value="Ha" />
        <constructor-arg value="dtthuha@gmail.com" />
    </bean>

    <bean id="a1" class="com.iuh.fit.di.Address">
        <constructor-arg value="HCMCity"></constructor-arg>
        <constructor-arg value="HCM"></constructor-arg>
        <constructor-arg value="Viet Nam"></constructor-arg>
    </bean>
    <bean id="e" class="com.iuh.fit.di.Employee">
        <constructor-arg value="12" type="int"></constructor-arg>
        <constructor-arg value="Thu Ha"></constructor-arg>
        <constructor-arg>
            <ref bean="a1" />
        </constructor-arg>
    </bean>

</beans>
```

B7. Test chương trình : Tạo Class Java Demobean chứa void main()

```
public class Demobean {

    public static void main(String[] args) {

        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
        Helloworld hel = context.getBean("helloWorld", Helloworld.class);
        hel.getMessage();
        Student std = context.getBean("s1", Student.class);
        std.show();
        Employee e = context.getBean("e", Employee.class);
        // (Employee) factory.getBean("e");
        e.show();
    }
}
```

Run Project: R_Click project → Run As → Java Application

Bài 2. Spring Ioc_DI

Tình huống chương trình có đối tượng Coach(Huấn luyện viên), với mỗi môn thể thao sẽ có các huấn luyện viên khác nhau như BaseballCoach, TennisCoach, CricketCoach. Thiết kế các đối tượng Coach, BaseballCoach, CricketCoach,... để loại bỏ sự phụ thuộc lẫn nhau

B1. Tạo Project Java

B2. Configure Project → Convert to Maven Project

B3. Cấu hình các dependency của Spring trong pom.xml tương tự như bài 1

B4: Tạo package `iuh.fit.se.libs` ;

B4.1 : Tạo Class Interface: `Coach.java`, `FortuneService.java`

```
package iuh.fit.se.libs ;  
  
public interface Coach {  
    public String getDailyWorkout();  
  
    public String getDailyFortune();  
}
```

```
package iuh.fit.se.libs ;  
  
public interface FortuneService {  
    public String getFortune();  
}
```

B4.2 : Tạo Class `HappyFortuneService` implement tới class `Fortune` chứa các phương thức hoạt động cho các class `BaseballCoach`, `TennisCoach`, `CricketCoach` phía sau:

```
package iuh.fit.se.libs ;  
import org.springframework.stereotype.Component;  
public class HappyFortuneService implements FortuneService {  
  
    public String getFortune(){  
        return "Today is your lucky day! Happy Fortune";  
    }  
}
```

B4.3 : Tạo class `BaseballCoach.java` implement class Coach

Trong `BaseballCoach` muôc sử dụng các phương thức của `FortuneService` (interface) khác

Làm giảm sự phụ thuộc giữa `BaseballCoach` và `FortuneService` (interface), trong lớp `BaseballCoach` riêng truyền/inject vào `FortuneService` thông qua Constructor với thuộc tính của `FortuneService` (interface) – DI Dependence Injection Contructor

```

package iuh.fit.se.libs ;

public class BaseballCoach implements Coach {

    // define a private field for the dependency
    private FortuneService fortuneService;
    // define a constructor for dependency injection
    public BaseballCoach(FortuneService theFortuneService) {
        fortuneService = theFortuneService;
    }
    public String getDailyWorkout() {
        return "Spend 30 minutes on batting practice";
    }

    public String getDailyFortune() {
        // use my fortuneService to get a fortune
        return fortuneService.getFortune();
    }
}

```

Depenence Injection Contructor

Tạo Class BaseFortuneService

```

package iuh.fit.se.libs ;
public class BaseFortuneService implements FortuneService{

    public String getFortune()
    {
        return "Today is your lucky day!";
    }
}

```

B5: Tương tự tạo thêm class TennisCoach

```

package iuh.fit.se.libs ;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Component;
public class TennisCoach implements Coach{
    private FortuneService theFortune;

    public TennisCoach (FortuneService fortuneService)
    { this.theFortune = fortuneService; }

    public String getDailyWorkout() {
        return "TennisCoach: Run a hard 5k";
    }

    public String getDailyFortune() { //use my foruneSevice to get fortune
        return theFortune.getFortune();
    }
}

```

Depenence Injection Contructor

Tạo TennisFortuneService.java

```

package iuh.fit.se.libs;

public class TennisFortuneService implements FortuneService{

    public String getFortune()
    {
        return "Tennis fortune: Today is your lucky day! ";
    }
}

```

B6: Tạo Criket Coach.java Class này thực hiện DI bằng Setter

```

package iuh.fit.se.libs;

public class CricketCoach implements Coach {
    private String emailAddress;
    private String team;

    public void setEmailAddress(String mailAddress) {

        System.out.println("CriketCoach: inside setter method - setEmailAddress");
        this.emailAddress = mailAddress;
    }

    public void setTeam(String team) {
        System.out.println("CriketCoach: inside setter method - setteam");
        this.team = team;
    }

    public String getEmailAddress() {
        return emailAddress;
    }

    public String getTeam() {
        return team;
    }

    private FortuneService fortuneService;// lay method FortuneService
}

public void setFortuneService(FortuneService fortuneService) {
    this.fortuneService = fortuneService;
}

// create a no-arg constructor
public CricketCoach() {
    // System.out.println("CriketCoach: inside no-arg constructor");
}

// our setter method

public String getDailyWorkout() {

    return "CriketCoach: practice fast bowling for 15 minutes ";
}

public String getDailyFortune() {
    // TODO Auto-generated method stub
    return fortuneService.getFortune();
}

```

Depenence Injection Setter Properties

B7: Cấu hình file bean *applicationContext.xml*. File lưu trong thư mục **src/main/java**

```
<?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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context.xsd">
    <context:component-scan base-package="iuh.fit.se.Libs">
        </context:component-scan>
        <bean id="myFortune" class="iuh.fit.se.libs.HappyFortuneService">
            </bean>
    <bean id="myCricketCoach" class="iuh.fit.se.libs.CricketCoach">
        <property name="fortuneService" ref="myFortune" />
        <property name="emailAddress" value="thegoodcoach@gmail.com" />
        <property name="team" value="India National" />
    </bean>
    <bean id="myTennisCoach" class="iuh.fit.se.libs.TennisCoach">
        <constructor-arg ref="myFortune" />
    </bean>
    <bean id="myCoach" class="iuh.fit.se.libs.BaseballCoach">
        <constructor-arg ref = "myFortune" />
    </bean>
</beans>
```

B8: Tạo class *SpringHelloApp.java* có hàm void *main()* để chạy chương trình, test các phương thức và các bean đã tạo trong *applicationContext.xml*

```

package iuh.fit.se.apps;
import iuh.fit.se.libs.Coach;
import iuh.fit.se.libs.CricketCoach;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class SpringHelloApp {
    public static void main(String[] args) {
        // load the spring configuration file
        ClassPathXmlApplicationContext context=
            new ClassPathXmlApplicationContext
                ("applicationContext.xml");
        //retrieve bean from spring container

        Coach theCoachtennis= context.getBean("myTennisCoach",Coach.class);
        System.out.println(theCoachtennis.getDailyWorkout());
        System.out.println(theCoachtennis.getDailyFortune());

        /*
        * Coach theCoach= context.getBean("myCricketCoach",Coach.class);
        * System.out.println(theCoach.getDailyWorkout());
        * System.out.println(theCoach.getDailyFortune()); CricketCoach
mycricket=
        * (CricketCoach) context.getBean("myCricketCoach",Coach.class);
        * System.out.println(mycricket.getEmailAddress());
        * System.out.println(mycricket.getTeam());
        */

        /*
        * CricketCoach theCricketCoach=
        * context.getBean("myCricketCoach",CricketCoach.class);
        * System.out.println(theCricketCoach.getEmailAddress());
        * System.out.println(theCricketCoach.getTeam());
        */

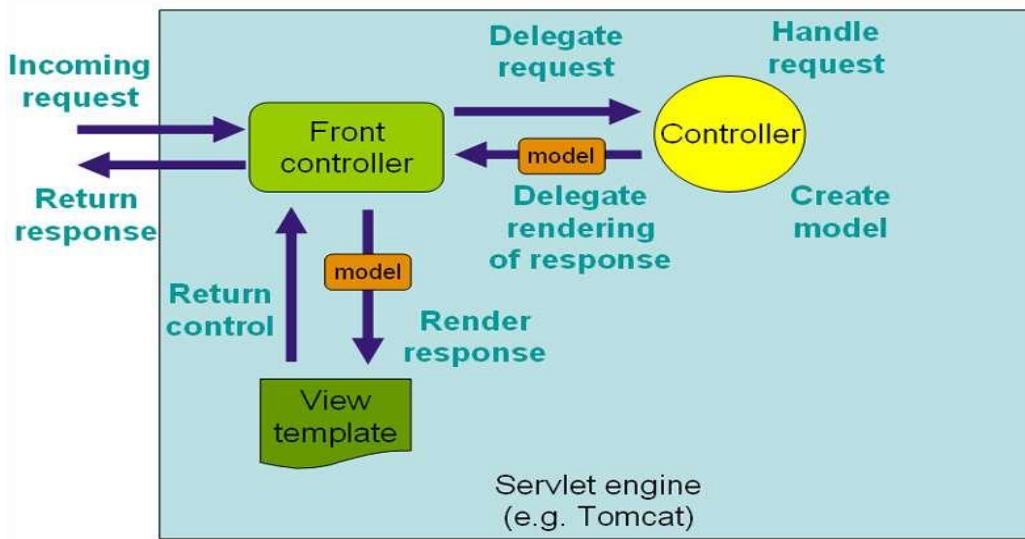
        //close the context
        context.close();
    }
}

```

Phần 2. Spring MVC

0. Mô hình Spring MVC trong môi trường Web

- Front Controller tiếp cận tất cả các request gửi tới, chuyển tới Controller cần xử lý.
- Controller trả về các Model cần thiết cho Front Controller và chọn View phù hợp.
- Front Controller cũng trả về kết quả cho người dùng.



Với Java, *Front Controller* thường là đối tượng implement interface **javax.servlet.Servlet**. Trong Spring MVC thì Front Controller là **org.springframework.web.servlet.DispatcherServlet**.

Các bước thực hiện:

- B0: Khai báo Dependencies trong pom.xml cho project Maven
- B1. Request tới ứng dụng Web, *DispatcherServlet* là đối tượng nhận tất cả các request.
- B2. Tìm và chuyển hướng request tới Request Handler phù hợp là các Controllers trong ứng dụng Web.
- B3. Các Controller xử lý request yêu cầu.
- B4. Chuẩn bị Model và chọn View hiển thị.
- B5. Trả về kết quả xử lý request cho *DispatcherServlet*.
- B6. *DispatcherServlet* sẽ gọi View Template phù hợp để xử lý việc hiển thị trên giao diện bằng cách sử dụng Model.
- B7. View Template trả kết quả về cho *DispatcherServlet*.
- B8. Trả Response.

Thực thi 1 project Application Web Spring MVC cần phải cấu hình một số các thành phần cho project:

Cấu hình 1. Cấu hình Front Controller **org.springframework.web.servlet.DispatcherServlet**: thường để trong web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
<display-name>spring-mvc-demo</display-name>
<absolute-ordering />

<!-- 1. Configure Spring dispatcher servlet -->

```

```

<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_mvc-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<!-- 2. setup url mapping for dispatcher servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<session-config>
    <session-timeout>30</session-timeout>
</session-config>
<jsp-config>
    <jsp-property-group>
        <url-pattern>*.jsp</url-pattern>
        <page-encoding>UTF-8</page-encoding>
    </jsp-property-group>
</jsp-config>

</web-app>

```

- Đường dẫn tới file xml
- Servlet-name: giống nhau
- Servlet-class: DispatcherServlet
- url-pattern: / mapping request từ client

Mặc định khi *DispatcherServlet* được khởi tạo, đối tượng này sẽ khởi tạo một đối tượng *org.springframework.web.context.WebApplicationContext* với hiện thực là đối tượng *org.springframework.web.context.support.XmlWebApplicationContext*.

Đối tượng *XmlWebApplicationContext* chứa cấu hình tất cả các beans mà ứng dụng định nghĩa trong khung chứa (container) của Spring. Đối tượng *XmlWebApplicationContext* này sẽ dùng một tập tin cấu hình của Spring với tên gọi mặc định là **[tên-servlet]-servlet.xml** nằm trong thư mục */src/webapp/WEB-INF* với ứng dụng SpringMVC tạo bằng Dynamic Web Project.

Cấu hình 2. Khai báo các beans trong root-servlet

Cấu hình root Servlet

Trường hợp định nghĩa nhiều Servlet trong một tập tin *web.xml*, mỗi servlet có thể được khởi tạo, ánh xạ đến các URL khác nhau. Dẫn đến trường hợp này sẽ có nhiều Spring container tương ứng với từng Servlet. Trong trường hợp đó, một số định nghĩa bean có thể lặp đi lặp lại trong nhiều Spring container.

Đối tượng *org.springframework.web.context.ContextLoaderListener* được tạo ra để giải quyết vấn đề lặp đi lặp lại này bằng cách **tạo ra một root *org.springframework.web.context.WebApplicationContext* sử dụng chung cho tất cả các Servlet**. *ContextLoaderListener* sẽ sử dụng tập tin được định nghĩa trong *contextConfigLocation* của tag *<context-param>*.

Thường trong tập tin *root-context.xml*, định nghĩa các bean, các thuộc tính dùng chung giữa các container của Spring, mỗi container Spring trong mỗi Servlet sẽ sử dụng các bean.

[tên-servlet]-servlet.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:mvc="http://www.springframework.org/schema/mvc"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!-- 3: Add support for component scanning : các package chứa các thành phần xử lý -->
    <context:component-scan base-package="iuh.fit.se.springmvc.demo, package
iuh.fit.se.springmvc.validation" >
    </context:component-scan>

    <!-- 4: Add support for conversion, formatting and validation support -->

    <!-- 5: Define Spring MVC view resolver: ViewResolver (xử lý View), các View là tập
tin JSP được ánh xạ từ thư mục /WEB-INF/view/ -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>

<!--Khai báo các bean sẽ dùng trong Controller như: database,... -->

<bean id="messageSource"
    class="org.springframework.context.support.ResourceBundleMessageSource">
    <property value="resources/messages" name="basenames" />
</bean>
</beans>

```

InternalResourceViewResolver sẽ map tên View mà trả về trong Controller với một tên file nằm trong một thư mục cụ thể.

Khai báo bean *InternalResourceViewResolver* sẽ map tất cả các tên View với các file có phần mở rộng .jsp nằm trong thư mục /src/main/WEB-INF/view để render giao diện.

- Định nghĩa request “/” với tên View trả về trong method home() là “home”, tương ứng với file /src/main/WEB-INF/view/home.jsp.
- Định nghĩa request “/login” với tên View trả về trong method loginPage () là “login”, tương ứng với file /src/main/WEB-INF/view/login/login.jsp.
- Định nghĩa request “/home” với tên View trả về trong method login() là “user”, tương ứng với file /src/main/WEB-INF/view/home/user.jsp.

```

@Controller
public class HomeController {

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        ...
    }
}

```

```

        return "home";
    }

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String loginPage(Locale locale, Model model) {
        return "login";
    }

    @RequestMapping(value = "/home", method = RequestMethod.POST)
    public String login(@Validated User user, Model model) {
        model.addAttribute("userName", user.getUserName());
        return "user";
    }
}

```

Cách ánh xạ giữa View và tập tin View:

- home → home.jsp
- login → login.jsp
- user → user.jsp

Cấu hình 3. Tạo Controller trong Spring MVC

- Annotation `@Controller` khai báo cùng với định nghĩa của lớp **XXXController**.
- Annotation `@RequestMapping` khai báo cùng với định nghĩa của phương thức xử lý nào đó `xulyData()`. Với annotation `@RequestMapping` có `@RequestMapping(value = "/", method = RequestMethod.GET)`
 - thuộc tính **value** để định nghĩa request URL
 - thuộc tính **method** để định nghĩa HTTP request method.

```

package iuh.fit.se.springmvc.demo;

import javax.servlet.http.HttpServletRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;
@Controller
@RequestMapping("/hello")
public class HelloWorldController {
    @RequestMapping("/showForm")
    public String showForm() {
        return "helloworld-form";
    }

    @RequestMapping("/processForm")
    public String processForm() {
        return "helloworld";
    } // action nhận
}

```

Client Request URL như sau:

<http://localhost:8080/.../hello/showForm>

→ *helloworld-form.jsp*

<http://localhost:8080/.../hello/processForm>

→ *helloworld.jsp*

- Tạo Controller để kiểm soát handle request từ client gửi tới dạng QueryString:

/login?username=xxx&password=xxx

```

import javax.servlet.http.HttpServletRequest;

```

```

import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class LoginController {
    @RequestMapping(value = "/login")

public void login(HttpServletRequest request){
    String username = request.getParameter("username");
    String username = request.getParameter("username");
}
}

```

Hoặc dùng @RequestParam annotation

```

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
@Controller
public class LoginController {

    @RequestMapping(value = "/login")
    public void login(@RequestParam String username, @RequestParam String password) {
        ...
    }
}

```

Nếu biến trong QueryString và biến phương thức khác tên thì thay đổi:

```

public void login(@RequestParam(value = "username") String user, @RequestParam
String password) {...}

```

Nếu bắt buộc phải có giá trị (khác null):

```

public void login(@RequestParam(value = "username", required = false) String user,
@RequestParam String password) {...}

```

Nếu muốn gán giá trị mặc định:

```

public void login(@RequestParam(value = "username", required = false, defaultValue = "xxxx")
String user, @RequestParam String password) {...}

```

Cấu hình 4. Đối tượng HttpServletRequest và HttpServletResponse trong Controller của Spring MVC

```

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

```

```

@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model, HttpServletRequest request,
HttpServletResponse response) {
        logger.info("Welcome home! The client locale is {}.", locale);
        logger.info(request.getContextPath());

        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG,
DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate);

        return "home";
    }
}

```

Cấu hình 5. Truyền dữ liệu vào Model trong Controller (Spring MVC)

Cách 1. Dùng đối tượng `org.springframework.ui.Model`, và phương thức `addAttribute("Tên thuộc tính", giá trị);`

```

@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Model model) {
    ...
    model.addAttribute("Tên thuộc tính", giá trị);
}

```

Cách 2. Dùng `java.util.Map`

```

@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Map<String, Object> model) {
    ...
    model.put("Tên thuộc tính", giá trị);
}

```

Cách 3. Dùng *org.springframework.ui.ModelMap*

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, ModelMap model) {
    ...
    model.addAttribute("Tên thuộc tính", giá trị)
}
```

Chuyển hướng (send redirect) request trong controller của Spring MVC

```
import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
public class HomeController {
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        return "redirect:/home";
    }
}
```

Câu hình 6. Đối tượng Enum: *org.springframework.web.bind.annotation.RequestMethod* gồm các request method

- RequestMethod.GET
- RequestMethod.HEAD
- RequestMethod.POST
- RequestMethod.PUT
- RequestMethod.PATCH
- RequestMethod.DELETE
- RequestMethod.OPTIONS
- RequestMethod TRACE

Bài 3: Spring MVC

Thực hiện kiểm soát dữ liệu truyền qua form dùng MVC

Thực hiện ứng dụng Spring MVC Sinhvien chứa form nhập thông tin sinh viên vào danh sách sinh viên



B1. Tạo *Dynamic Web Project*, khai báo các thông tin liên quan

B2. *pom.xml*:

Khai báo các dependencies, Maven → Update Project

Khai báo dependencies như bài 2 gồm các dependences cơ bản:

```
/springframework/spring-core  
/springframework/spring-beans  
/springframework/spring-context
```

Thêm dependency cần thiết cho spring mvc, spring web

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-webmvc</artifactId>  
    <version>5.2.20.RELEASE</version>  
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-web</artifactId>  
    <version>5.2.20.RELEASE</version>  
</dependency>
```

B3: *web.xml*: Khai báo cấu hình cần thiết của ứng dụng Web (*Cấu hình 1. DispatcherServlet*)

{1. Cấu hình Spring dispatcher servlet, 2. Mapping cho Spring dispatcher servlet }

B4: *WEB-INF/dispatcher-servlet.xml*:

Khai báo cấu hình tối thiểu về Spring MVC (*Cấu hình 2. servlet, servlet mapping của project*)

{3. Scan-context, 4. Annotation validation, 5. ViewResolver : như bài 2}

B5: Thực hiện Code, tạo Model com.demo.entities.Sinhvien: Model

Các thuộc tính của Model Sinhvien, tạo constructor, get, set, toString() cho Model Sinhvien

```
private Integer Id;  
  
private String fname;  
  
private String lname;  
  
private float diemtb;
```

B6: com.demo.controllers.SinhvienController: Controller bao gồm action hiển thị form, nhập thông tin sinh viên, hiển thị danh sách sinh viên

```
package com.demo.controllers;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.servlet.ModelAndView;  
  
import com.demo.entities.Sinhvien;  
  
@Controller  
@RequestMapping("/sinhvien")  
public class SinhvienController {  
    List<Sinhvien> list;  
    int count = 6;  
  
    public SinhvienController() {  
        list = new ArrayList<>();  
        list.add(new Sinhvien(1, "Hoang", "Minh", 7.8d));  
        list.add(new Sinhvien(2, "Lam", "Oanh", 8.5d));  
        list.add(new Sinhvien(3, "Minh", "Ngoc", 7.0d));  
        list.add(new Sinhvien(4, "Lan", "Phuong", 6.2d));  
        list.add(new Sinhvien(5, "Hoang", "Bao", 5.6d));  
    }  
  
    @RequestMapping("/sinhvienform")  
    public ModelAndView showForm() {  
        return new ModelAndView("sinhvienform", "command", new Sinhvien());  
    }  
  
    @RequestMapping(value = "/save", method = RequestMethod.POST)  
    public ModelAndView save(@ModelAttribute("sinhvien") Sinhvien sv) {  
        // thêm sinhvien moi vao list  
  
        int id = count++;  
        String fname = sv.getFname();  
        String lname = sv.getLname();  
        Double diemtb = sv.getDiemtb();  
        sv = new Sinhvien(id, fname, lname, diemtb);  
        list.add(sv);  
        return new ModelAndView("viewsinhvien", "list", list);  
    }  
  
    @RequestMapping("/viewsinhvien")  
    public ModelAndView viewstudent() {  
        return new ModelAndView("viewsinhvien", "list", list);  
    }  
}
```

B7. Tạo view sinhvienform.jsp

```
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <h2>Thông tin sinh viên</h2>
    <form:form action="save" method="post">
        <table>
            <tr>
                <td>First Name(*)</td>
                <td><form:input path="fname" /> </td>
            </tr>
            <tr>
                <td>Last Name(*)</td>
                <td><form:input path="lname" /> </td>
            </tr>
            <tr>
                <td>Diem TB(*)</td>
                <td><form:input path="diemtb" /> </td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit" value="Save"></td>
            </tr>
        </table>
    </form:form>
</body>
</html>
```

B8. Tạo view viewsinhvien.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <h3>Danh sách Sinh viên</h3>
    <table border="2" width="70%" cellspacing="0">
        <tr>
            <th>MSSV</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Diem Trung Bình</th>
        </tr>
        <c:forEach var="std" items="${list}">
            <tr>
                <td>${std.id}</td>
                <td>${std.fname}</td>
                <td>${std.lname}</td>
                <td>${std.diemtb}</td>
            </tr>
        </c:forEach>
    </table>
</body>
</html>
```

R-Click Project → Run On Server Project

Bài 4: Spring MVC - Truyền dữ liệu Model qua lại View, Controller

Thực hiện ứng dụng Spring MVC 2 với Model Student và Customer theo form với các view tương ứng sau:

Request mặc định : /

Spring MVC Demo - Home page

[Plain Hello World](#)

[Student form](#)

[Customer form](#)

Request mặc định : /

→ url: serletname/

Request hiển thị form hello

→ url: serletname/hello/showForm

Request để hiển thị form student

→ url: serletname/student/showForm

Request để hiển thị form customer

→ url: serletname/customer/showForm

Form student

First Name	<input type="text"/>
Last Name	<input type="text"/>
Country	<input type="text" value="Brazil"/> <input type="button" value="▼"/>
Favorite Language	<input type="radio"/> Java <input type="radio"/> C# <input type="radio"/> PHP <input type="radio"/> Ruby
Operating Systems	<input type="checkbox"/> Linux <input type="checkbox"/> Mac OS <input type="checkbox"/> MS Windows
<input type="button" value="submit"/>	

Form customer

Fill out the form. Asterisk (*) means required.

First name	<input type="text"/>
Last name (*)	<input type="text"/>
Free passes	<input type="text"/>
Postal Code	<input type="text"/>
Course Code	<input type="text"/>
<input type="button" value="Submit"/>	

Tạo [Dynamic Web Project](#), khai báo các thông tin liên quan

B1. *web.xml*: Khai báo cấu hình cần thiết của ứng dụng Web (*Cấu hình DispatcherServlet – 1.*)

{1. Cấu hình Spring dispatcher servlet, 2. Mapping cho Spring dispatcher servlet }

B2. *pom.xml*: Khai báo các dependencies Khai báo các dependencies, Maven → Update Project

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-web -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/javax.servlet.jsp/javax.servlet.jsp-api -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
    <dependency>
        </dependency>
    <!-- https://mvnrepository.com/artifact/org.hibernate.validator/hibernate-validator -->
    <dependency>
        </dependency>
</dependencies>
```

B3. WEB-INF/spring-mvc-content.xml:

Khai báo cấu hình tối thiểu về Spring MVC (Cấu hình servlet của project – 2.)

{3. Scan-context, 4. Annotation validation, 5. ViewResolver}

B4. Tạo Controller *iuh.fit.se.springmvc.demo.HomeController.java* hiển thị danh sách các liên kết ứng với các request /student hay /customer từ client gửi lên

```
package iuh.fit.se.springmvc.demo;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class HomeController {

    @RequestMapping("/")
    public String showMyPage(){
        return "main-menu";
    }
}
```

Request Mapping (): mặc định của Project chạy method showMyPage() → main-menu.jsp

B5. Trong thư mục WEB-INF/view/main-menu.jsp:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Spring MVC Demo - Home page</h1>
        <a href="hello/showForm">Plain Hello World</a>
        <br><br>
        <a href="student/showForm">Student form</a>
        <br><br>
        <a href="customer/showForm">Customer form</a>
    </body>
</html>
```

Request Mapping:

1.hello/showForm

→ Controller cho request mapping hello, method showForm

2.student/showForm

→ Controller cho request mapping student, method showForm

3.customer/showForm

→ Controller cho request mapping customer, method showForm

B6. Theo nhu Request trên, project cần có các Controller tương ứng cho các Request

Tạo Controller *iuh.fit.se.springmvc.demo.HelloWorldController.java*

```
package iuh.fit.se.springmvc.demo;

import javax.servlet.http.HttpServletRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
@RequestMapping("/hello")
public class HelloWorldController {
    @RequestMapping("/showForm")
    public String showForm() {
        return "helloworld-form";
    }

    @RequestMapping("/processForm")
    public String processForm() {
        return "helloworld";
    }
    // action nhận
}
```

Client Request URL như sau:

<http://localhost:8080/.../hello/showForm>

→ *helloworld-form.jsp*

<http://localhost:8080/.../hello/processForm>

→ *helloworld.jsp*

B7. Tạo view /WEB-INF/view/helloworld-form.jsp

helloworld-form.jsp chứa form nhập dữ liệu truyền qua trang *helloworld.jsp*

Truyền dữ liệu giữa các view jsp

helloworld-form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<html>
<head>
<title>Hello World - Input Form</title>
</head>
<body>

<form action="processForm" method="post">
    <input type="text" name="studentName"
    placeholder="What's your name? "/>
    <input type="submit" />
</form>

</body>
</html>
```

helloworld.jsp nhận parameter *name = "studentName"*

```
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<title>Hello World of Spring</title>
</head>
<body>
    Hello World of Spring!
    <br><br>
    Student name : ${param.studentName}
</body>
</html>
```

B8. Theo form Student thì cần tạo model iuh.fit.se.springmvc.demo.Student với các thuộc tính như sau:

```
package iuh.fit.se.springmvc.demo;

import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;

public class Student {
    private String firstName;
    private String lastName;
    private String country;
    private String favoriteLanguage;
    private String[] operatingSystems;

    private LinkedHashMap<String, String> countryOptions;
    private LinkedHashMap<String, String> favoriteLanguageOptions;
    private ArrayList<String> operatingSystemOptions;

    public Student() {
        // populate favorite language options
        favoriteLanguageOptions = new LinkedHashMap<String, String>();
        // parameter order: value, display label
        favoriteLanguageOptions.put("Java", "Java");
        favoriteLanguageOptions.put("#", "C#");
        favoriteLanguageOptions.put("PHP", "PHP");
        favoriteLanguageOptions.put("Ruby", "Ruby");

        operatingSystemOptions = new ArrayList<String>();
        operatingSystemOptions.add("Linux");
        operatingSystemOptions.add("Mac OS");
        operatingSystemOptions.add("MS Windows");

        //populate country options: use ISO country code
        countryOptions = new LinkedHashMap<String, String>();
        countryOptions.put("BR", "Brazil");
        countryOptions.put("FR", "France");
        countryOptions.put("DE", "Germany");
        countryOptions.put("IN", "India");
        countryOptions.put("US", "United State");
    }

    public String[] getOperatingSystems() {
        return operatingSystems;
    }
    public void setOperatingSystems(String[] operatingSystems) {
        this.operatingSystems = operatingSystems;
    }
    public LinkedHashMap<String, String> getFavoriteLanguageOptions() {
        return favoriteLanguageOptions;
    }
    public void setFavoriteLanguage(String favoriteLanguage) {
        this.favoriteLanguage = favoriteLanguage;
    }
    public String getFavoriteLanguage() {
        return favoriteLanguage;
    }
    public LinkedHashMap<String, String> getCountryOptions() {
        return countryOptions;
    }
    //favoriteOperating
    public ArrayList<String> getOperatingSystemOptions(){
        return operatingSystemOptions;
    }

    public String getFirstName() {}
    public void setFirstName(String firstName) {}
    public String getLastName() {}
    public void setLastName(String lastName) {}
    public String getCountry() {}
    public void setCountry(String country) {}
}
```

B9. Tạo Controller iuh.fit.se.springmvc.demo.StudentController

```
package iuh.fit.se.springmvc.demo;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotationModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;

//import java.util.Map;
@Controller
@RequestMapping("/student")
public class StudentController {
    @RequestMapping("/showForm")

    public String showForm(Model theModel) {
        Student theStudent = new Student();
        theModel.addAttribute("student", theStudent);
        return "student-form";
    }

    @RequestMapping("/processForm")
    public String processForm(@ModelAttribute("student") Student theStudent) {
        return "student-confirmation";
    }
}
```

Request Mapping

1. student/showForm
Truyền Model qua view
2. student/processForm
(@ModelAttribute("Student"))

B10. Trong view /WEB-INF/view/student-form.jsp trong đó sử dụng form MVC cho Listbox Country, Radio Favorite Language, Checkbox Operationg System được nhận items từ Model Student ở B3

```
<%@ page pageEncoding="UTF-8" language="java"
contentType="text/html charset=UTF-8"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome to Spring Web MVC project</title>
</head>
<body>
    <form:form action="processForm" modelAttribute="student">
        <table>
            <tr>
                <td>First Name</td>
                <td><form:input path="firstName" /> <form:errors
path="firstName"></form:errors></td>
            </tr>
            <tr>
                <td>Last Name</td>
                <td><form:input path="lastName" /></td>
            </tr>
            <tr>
                <td>Country</td>
                <td><form:select path="country">
                    <form:options items="${student.countryOptions}" />
                </form:select></td>
            </tr>
            <tr>
                <td>Favorite Language</td>
                <td><form:radioButtons path="favoriteLanguage"
items="${student.favoriteLanguageOptions}" /></td>
            </tr>
            <tr>
                <td>Operating Systems</td>
                <td><form:checkboxes path="operatingSystems"
items="${student.operatingSystemOptions}" /></td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit" value="submit" /></td>
            </tr>
        </table>
    </form:form>
</body>
</html>
```

First Name	<input type="text"/>
Last Name	<input type="text"/>
Country	<input type="text" value="Brazil"/> <input type="button" value="▼"/>
Favorite Language	<input type="radio"/> Java <input type="radio"/> C# <input type="radio"/> PHP <input type="radio"/> Ruby
Operating Systems	<input type="checkbox"/> Linux <input type="checkbox"/> Mac OS <input type="checkbox"/> MS Windows
<input type="button" value="submit"/>	

B11. Trong view /WEB-INF/view/student-confirmation.jsp nhận Model student truyền từ view sang view

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>

<!DOCTYPE html>
<html>
<head>
<title>Student Confirmation</title>
</head>
<body>
The student is confirmed: ${student.firstName} ${student.lastName}
<br>
<br> Country: ${student.country}
<br>
<br> Favorite Language: ${student.favoriteLanguage}
<br>
<br> Operating Systems:
<ul>
<c:forEach var="temp" items="${student.operatingSystems}">
<li>${temp}</li>
</c:forEach>
<a href="showForm">Back to Form </a>
</ul>
</body>
</html>
```

→Run Project →Run As→Run on Server

B12: Tương tự như Model Student, tạo Model iuh.fit.se.springmvc.demo.Customer theo form Customer
Trong phần này kết hợp kiểm tra Spring Validation cho các giá trị nhập vào trên Form:

Fill out the form. Asterisk (*) means required.

First name	<input type="text"/>
Last name (*)	<input type="text"/>
Free passes	<input type="text"/>
Postal Code	<input type="text"/>
Course Code	<input type="text"/>
<input type="button" value="Submit"/>	

Các Validation và Validation Annotations thường dùng do Spring cung cấp :

Validation Features
Required
Validate length
Validate numbers
Validate with regular expressions
Custom validation

Annotation	Description
@NotNull	The value can't be null.
@Min	Number must be equal or greater than the specified value.
@Max	Number must be equal or less than the specified value.
@Size	Size must be equal to the specified value.
@Pattern	Sequence follows the specified regular expression.
@Future/Past	Date must be in future or past of given date
Others...	

B13. Tại Model: Các validation được cung cấp qua các Annotation trực tiếp trên các thuộc tính.

```
private String courseCode;
@Pattern(regexp="^[a-zA-Z0-9]{5}", message="only 5 chars/digits")
private String postalCode;
@NotNull(message="is required")
@Size(min=4, message="is required")
private String lastName;
@NotNull(message="is required")
@Min(value=0, message="must be greater than or equal to zero")
@Max(value=10, message="must be less than or equal to 10")
private Integer freePasses;
```

B14. Tại Controller: sử dụng @Valid trên các Method có các Model đã thiết lập Validation, sử dụng Model **BindingResult** để nhận kết quả của Validation.

```
@RequestMapping("/processForm")
public String processForm(@Valid @ModelAttribute("customer") Customer theCustomer, BindingResult theBindingResult) {
    if (theBindingResult.hasErrors()) {
        return "customer-form";
    } else {
        return "customer-confirmation";
    }
}
```

View cho customer

B15. Tại View: Hiển thị thông báo lỗi trên mỗi thành phần của form Validation thông qua `<form:errors>`

```
<tr>
    <td>Last name (*)</td>
    <td><form:input path="lastName" />
        <form:errors path="lastName" cssClass="error" /></td>
</tr>
<tr>
    <td>Free passes</td>
    <td><form:input path="freePasses" />
        <form:errors path="freePasses" cssClass="error" /></td>
</tr>
```

Lưu ý:

- Trong file pom.xml thêm dependency của Hibernate-Validator cho project

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.2.0.Final</version>
</dependency>
```

- Trong file WEB-INF/spring-mvc-content.xml khai báo bean validation

```
<!-- 4: Add support for conversion, formatting
and validation support -->
<mvc:annotation-driven/>
```

B16: Tạo view /WEB-INF/view/customer-form.jsp, customer-confirmation

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
4
5 <!DOCTYPE html >
6 <html>
7
8 <head>
9     <title>Customer Registration Form</title>
0
1<style>
2     .error {color:red}
3 </style>
4 </head>
5 <body>
6 <i>Fill out the form. Asterisk (*) means required.</i>
7 <br><br>
8 <form:form action="processForm" modelAttribute="customer" >
9     First name: <form:input path="firstName" />
10    <br><br>
11    Last name (*): <form:input path="lastName" />
12    <form:errors path="lastName" cssClass="error" />
13    <br><br>
14    Free passes: <form:input path="freePasses" />
15    <form:errors path="freePasses" cssClass="error" />
16    <br><br>
17    Postal Code: <form:input path="postalCode" />
18    <form:errors path="postalCode" cssClass="error" />
19    <br><br>
20    Course Code: <form:input path="courseCode" />
21
22    <br><br>
23    <input type="submit" value="Submit" />
24 </form:form>
25 </body>
26
27 </html>

```

```

1 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
2 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>
3 <!DOCTYPE html>
4
5 <html>
6 <head>
7 <title>Customer Confirmation</title>
8 </head>
9 <body>
0     The customer is confirmed: ${customer.firstName} ${customer.lastName}
1     <br>
2     <br> Free passes: ${customer.freePasses}
3     <br>
4     <br> Postal Code: ${customer.postalCode}
5     <br>
6     <br> Course code: ${customer.courseCode}
7 </body>
8 </html>

```

Bài 5: Spring MVC – CRUD – EMPLOYEE

Spring MVC với các thao tác CRUD thêm, xóa, sửa với cơ sở dữ liệu.

Thực hiện thao tác thêm xóa sửa với CSDL quản lý thông tin nhân viên. CSDL quản lý bao gồm bảng Employee (ID, NAME, SALARY, DESIGNATION), trong đó mã nhân viên ID tự động tăng.

Employees List

Id	Name	Salary	Designation	Edit	Delete
1	Thanh Van	2400000.0	Tester	Edit	Delete
3	Hoàng Giang	2300000.0	Developer	Edit	Delete
4	Hồng Minh	2400000.0	QA	Edit	Delete
5	Hoàng Khánh	2500000.0	Developer	Edit	Delete
6	Ngọc Dung	2200000.0	QA	Edit	Delete

[Add New Employee](#)

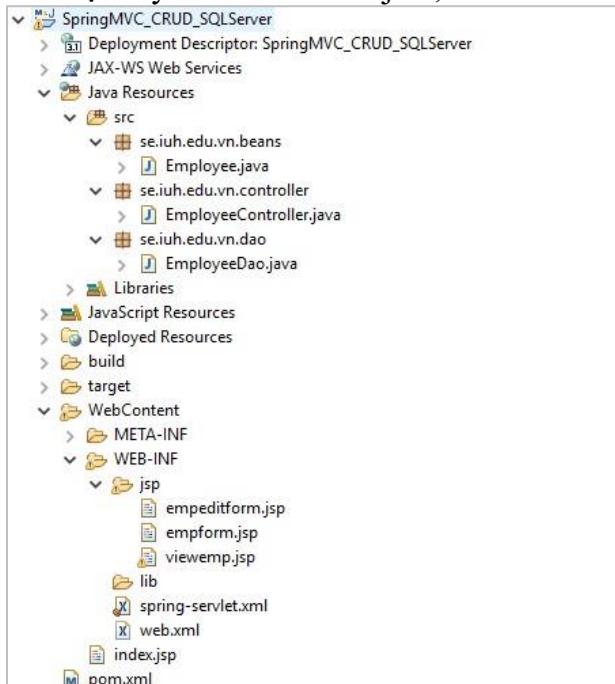
Thực hiện các chức năng:

- Khi chương trình chạy cho phép hiển thị danh sách nhân viên.
- Chọn “Edit” cho phép sửa thông tin của từng nhân viên, chọn “Delete” cho phép xóa thông tin của nhân viên và cập nhật CSDL..

Chọn “Add New Employee” hiển thị form cho phép thêm nhân viên mới vào CSDL.

Tham khảo hướng dẫn:

B1. Tạo Dynamic Web Project, khai báo các thông tin liên quan



B2. pom.xml:

Khai báo các dependencies, Maven → Update Project

Khai báo dependencies như bài 2 gồm các dependences cơ bản:

```
/springframework/spring-core  
/springframework/spring-beans  
/springframework/spring-context  
/springframework/spring-webmvc  
/springframework/spring-web  
/springframework/spring-jstl
```

Khai báo thêm dependencies cho việc thực hiện kết nối database:

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-jdbc</artifactId>  
    <version>5.2.22.RELEASE</version>  
</dependency>  
<!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->  
<dependency>  
    <groupId>com.microsoft.sqlserver</groupId>  
    <artifactId>mssql-jdbc</artifactId>  
    <version>11.2.3.jre17</version>  
</dependency>
```

SSL

B3: web.xml: Khai báo cấu hình cần thiết của ứng dụng Web (*Cấu hình 1. DispatcherServlet*)

{1. Cấu hình Spring dispatcher servlet, 2. Mapping cho Spring dispatcher servlet }

B4: WEB-INF/spring-servlet.xml:

Khai báo cấu hình tối thiểu về Spring MVC (*Cấu hình 2. servlet, servlet mapping của project*)

{3. Scan-context, 4. Annotation validation, 5. ViewResolver : như bài 2}

Thêm cấu hình cho kết nối SQL

{6. khai báo bean ds cấu hình các thông số kết nối CSDL }

{7. Khai báo bean jt đến class JdbcTemplate làm việc với CSDL được inject vào và sử dụng sau này trong ứng dụng }

Tham khảo code dưới:

```

<?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:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <!--3. Khai báo package chứa các thành phần xử lý Controller -->
    <context:component-scan base-
package="se.iuh.edu.vn.controller"></context:component-scan>
    <!--4. Validation -->

    <!--5. Khai báo ViewResolver (xử lý View), các View là tập tin JSP được ánh
    xạ từ thư mục /WEB-INF/jsp/ -->
    <bean
        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
    <!--6. Khai báo Data Source -->

    <bean id="ds"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver"></property>
        <property name="url"
value="jdbc:sqlserver://localhost:1433;databaseName=EmployeeDB"></property>
        <property name="username" value="sa"></property>
        <property name="password" value="1234567890"></property>
    </bean>
    <!--7. Khai báo jdbc Template -->
    <bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="ds"></property>
    </bean>

    <bean id="dao" class="se.iuh.edu.vn.dao.EmployeeDao">
        <property name="template" ref="jt"></property>
    </bean>
</beans>

```

B5. Lớp mô tả dữ liệu (Entity)

```
package se.iuh.edu.vn.beans;

public class Employee {
    private int id;
    private String name;
    private float salary;
    private String designation;

    + public int getId() {..}

    + public void setId(int id) {..}

    + public String getName() {..}

    + public void setName(String name) {..}

    + public float getSalary() {..}

    + public void setSalary(float salary) {..}

    + public String getDesignation() {..}

    + public void setDesignation(String designation) {..}

}
```

B6. 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 inject vào Controller trong ứng dụng bởi `@Autowired` để sử dụng sau này. Khai báo này có thể để trong lớp EmployeeDao hoặc khai báo bên Controller.

```
package se.iuh.edu.vn.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import se.iuh.edu.vn.beans.Employee;

public class EmployeeDao {
    JdbcTemplate template;

    public void setTemplate(JdbcTemplate template) {
        this.template = template;
    }

    public int save(Employee p) {
        String sql = "insert into Employee(name,salary,designation) values('"
            + p.getName() + "','" + p.getSalary()
            + "','" + p.getDesignation() + "')";
        return template.update(sql);
    }

    public int update(Employee p) {
        String sql = "update Employee set name='" + p.getName() + "', salary="
            + p.getSalary(), designation="
            + p.getDesignation() + ' where id=' + p.getId() + "";
        return template.update(sql);
    }

    public int delete(int id) {
        String sql = "delete from Employee where id=" + id + "";
        return template.update(sql);
    }

    public Employee getEmpById(int id) {
        String sql = "select * from Employee where id=?";
        return template.queryForObject(sql, new Object[] { id },
            new BeanPropertyRowMapper<Employee>(Employee.class));
    }

    public List<Employee> getEmployees() {
        return template.query("select * from Employee", new RowMapper<Employee>() {
            public Employee mapRow(ResultSet rs, int row) throws SQLException {
                Employee e = new Employee();
                e.setId(rs.getInt(1));
                e.setName(rs.getString(2));
                e.setSalary(rs.getFloat(3));
                e.setDesignation(rs.getString(4));
                return e;
            }
        });
    }
}
```

B7. Lớp xử lý (Controller)

```
package se.iuh.edu.vn.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import se.iuh.edu.vn.beans.Employee;
import se.iuh.edu.vn.dao.EmployeeDao;

@Controller
public class EmployeeController {
    @Autowired
    EmployeeDao dao; // sẽ inject dao khai báo trong file spring-servlet.xml

    /* Hiển thị form để nhập dữ liệu, lệnh "command" là reserved request attribute
     * dùng để hiển thị đối tượng dữ liệu vào form */
    @RequestMapping("/empform")
    public ModelAndView showform() {
        return new ModelAndView("empform", "command", new Employee());
    }

    /* Lưu đối tượng vào CSDL. @ModelAttribute dùng để đưa dữ liệu vào đối
     * tượng model. Lưu ý cần chỉ rõ phương thức truyền là POST: RequestMethod.POST,
     * mặc định GET request */
    @RequestMapping(value = "/save", method = RequestMethod.POST)
    public ModelAndView save(@ModelAttribute("emp") Employee emp) {
        dao.save(emp);
        // sẽ chuyển hướng sang viewemp request mapping
        return new ModelAndView("redirect:/viewemp");
    }

    /* Trả về danh sách nhân viên */
    @RequestMapping("/viewemp")
    public ModelAndView viewemp() {
        List<Employee> list = dao.getEmployees();
        return new ModelAndView("viewemp", "list", list);
    }

    /* Hiển thị đối tượng dữ liệu vào form với id cụ thể. @PathVariable đưa
     * dữ liệu URL vào biến. */
    @RequestMapping(value = "/editemp/{id}")
    public ModelAndView edit(@PathVariable int id) {
        Employee emp = dao.getEmpById(id);
        return new ModelAndView("empeditform", "command", emp);
    }

    /* Cập nhật đối tượng model. */
    @RequestMapping(value = "/editsave", method = RequestMethod.POST)
    public ModelAndView editsave(@ModelAttribute("emp") Employee emp) {
        dao.update(emp);
        return new ModelAndView("redirect:/viewemp");
    }

    /* Xóa mẫu tin với id trên URL và chuyển sang RequestMapping /viewemp */
    @RequestMapping(value = "/deleteemp/{id}", method = RequestMethod.GET)
    public ModelAndView delete(@PathVariable int id) {
        dao.delete(id);
        return new ModelAndView("redirect:/viewemp");
    }
}
```

B8. Các giao diện xử lý xem danh sách, thêm nhân viên mới, cập nhật nhân viên đã tồn tại trong CSDL.

viewemp.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<h1>Employees List</h1>
<table border="2" width="70%" cellpadding="2">
    <tr>
        <th>Id</th>
        <th>Name</th>
        <th>Salary</th>
        <th>Designation</th>
        <th>Edit</th>
        <th>Delete</th>
    </tr>
    <c:forEach var="emp" items="${list}">
        <tr>
            <td>${emp.id}</td>
            <td>${emp.name}</td>
            <td>${emp.salary}</td>
            <td>${emp.designation}</td>
            <td><a href="editemp/${emp.id}">Edit</a></td>
            <td><a href="deleteemp/${emp.id}">Delete</a></td>
        </tr>
    </c:forEach>
</table>
<br />
<a href="empform">Add New Employee</a>
```

empform.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<form:form method="post" action="save">
    <table>
        <tr>
            <td>Name :</td>
            <td><form:input path="name" /></td>
        </tr>
        <tr>
            <td>Salary :</td>
            <td><form:input path="salary" /></td>
        </tr>
        <tr>
            <td>Designation :</td>
            <td><form:input path="designation" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="Save" /></td>
        </tr>
    </table>
</form:form>
```

empeditform.jsp

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<h1>Edit Employee</h1>
<form:form method="POST" action="/SpringMVC_CRUD_SQLServer/editsave">
    <table>
        <tr>
            <td></td>
            <td><form:hidden path="id" /></td>
        </tr>
        <tr>
            <td>Name :</td>
            <td><form:input path="name" /></td>
        </tr>
        <tr>
            <td>Salary :</td>
            <td><form:input path="salary" /></td>
        </tr>
        <tr>
            <td>Designation :</td>
            <td><form:input path="designation" /></td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Edit Save" /></td>
        </tr>
    </table>
</form:form>
```

Bài 6. Spring MVC - Upload file

Thực hiện ứng dụng Spring MVC Upload File

Tham khảo hướng dẫn:

B1. Khai báo Maven pom.xml:

Ngoài những thư viện của Spring, khai báo thêm các dependencies:

```
<!-- https://mvnrepository.com/artifact/jstl/jstl -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
```

```
    <version>1.3.3</version>
</dependency>
```

B2. Cấu hình Spring *spring-servlet.xml*

Ngoài các khai báo Khai báo package chứa các thành phần xử lý Controller, khai báo bean ViewResolver, khai báo thêm bean
org.springframework.web.multipart.commons.CommonsMultipartResolver:

```
<bean id="multipartResolver"  
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver" />
```

B3. Cấu hình Project web.xml

Khai báo FrontController: DispatcherServlet,

B4. Tao UploadController

B5. Tạo View

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

<!DOCTYPE html>
<html>
<head>
<title>Image File Upload</title>
<style>
    #dvPreview1 {
        height: 100px;
        width: 100px;
        color:blue;
    }
</style>
<script src="jquery-1.10.2.js"></script>

<script>
    //đoạn script hiển thị hình khi người dùng chọn file hình
    $(function () {
        $("#fileToUpload").change(function () {
            //kiểm tra xem có phải hình không? dùng RegularExpression
            var regex = /^[a-zA-Z0-9\s_\\.\-:]+(\.jpg|.jpeg|.gif|.png|.bmp)$/;
            if (regex.test($(this).val().toLowerCase())) {
                if (typeof (FileReader) != "undefined") {
                    var reader = new FileReader();
                    reader.onload = function (e) {
                        $("#dvPreview1").attr("src", e.target.result);
                    }
                    reader.readAsDataURL($(this)[0].files[0]);
                } else {
                    alert("This browser does not support FileReader.");
                }
            } else {
                alert("Please upload a valid image file.");
            }
        });
    });
</script>

</head>
<body>
    <h1>File Upload Example</h1>

    <h3 style="color: red">${filesuccess}</h3>
    <form:form method="post" action="savefile"
        enctype="multipart/form-data">
        <p>
            <label for="image">Choose Image</label>
        </p>
        <p>
            <input name="file" id="fileToUpload" type="file" />
        </p>
        <p>
            <input type="submit" value="Upload">
        </p>
    </form:form>
    <div id="#dvPreview1"> </div>
</body>
</html>
```

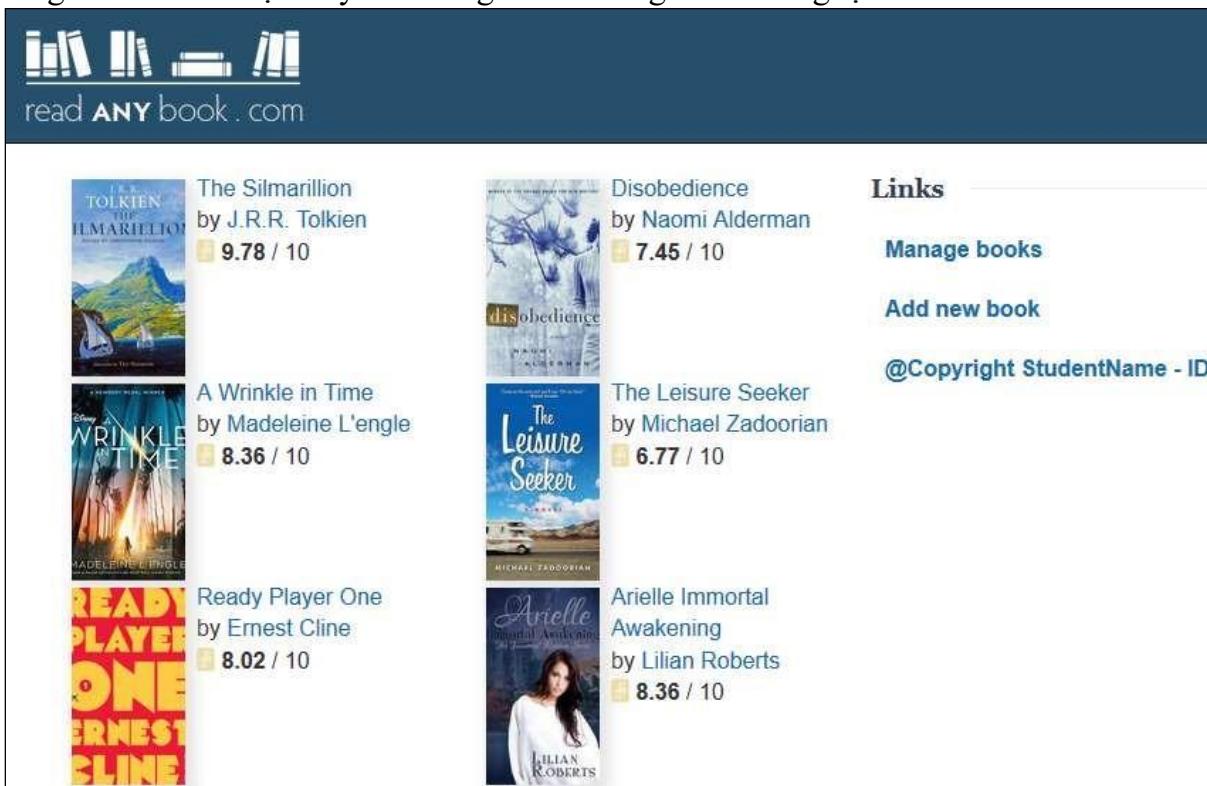
Bài 7. Spring MVC – CRUD - BOOKONLINE

Xây dựng ứng dụng quản lý thông tin sách, các trang Web cần thực hiện các chức năng: *xem danh sách các quyền sách, thêm mới sách và xóa sách không cần thiết*. Một phần của cơ sở dữ liệu được mô tả như sau:

BookOnline(BookID, BookName, Author, PublishYear, Rating, ImageURL)

Xây dựng ứng dụng dùng Spring MVC Framework kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình.
- Dùng HTML/CSS tạo Layout chung cho chương trình tương tự:



Hình 1. Layout của trang Web quản lý thông tin sách cho website readanybook.com

- Viết lớp kết nối CSDL và lớp thực hiện thao tác với việc truy xuất cơ sở dữ liệu (bao gồm các phương thức: liệt kê danh sách sách, thêm mới, xóa thông tin sách).
- Tạo Controller cho phép hiển thị giao diện Web với danh sách các quyền sách online lấy từ cơ sở dữ liệu.
- Tạo Controller cho phép hiển thị giao diện thao tác với việc thêm dữ liệu (*nhấn vào link "Add new book"*). Dữ liệu phải được kiểm tra trước khi lưu vào cơ sở dữ liệu:
 - Tất cả dữ liệu *BookID*, *BookName*, *Author*, *PublishYear*, *Rating*, *ImageURL* đều bắt buộc.
 - Năm xuất bản (*PublishYear*): từ 1900 – 2018

- Tên sách (*BookName*): Các ký tự đầu của mỗi từ là ký tự hoa, không có số hay ký tự đặc biệt trong tên.

Sau khi thực hiện thêm thành công, hiển thị danh sách mới.

- Tạo Controller cho phép hiển thị giao diện Web cho phép thực hiện chức năng quản lý (*nhấn vào link “Manage books”*): thao tác hủy dữ liệu. Trước khi hủy hỏi người dùng có muốn hủy không.

Bài 8: Spring MVC – HIBERNATE – WEB APPLICATION

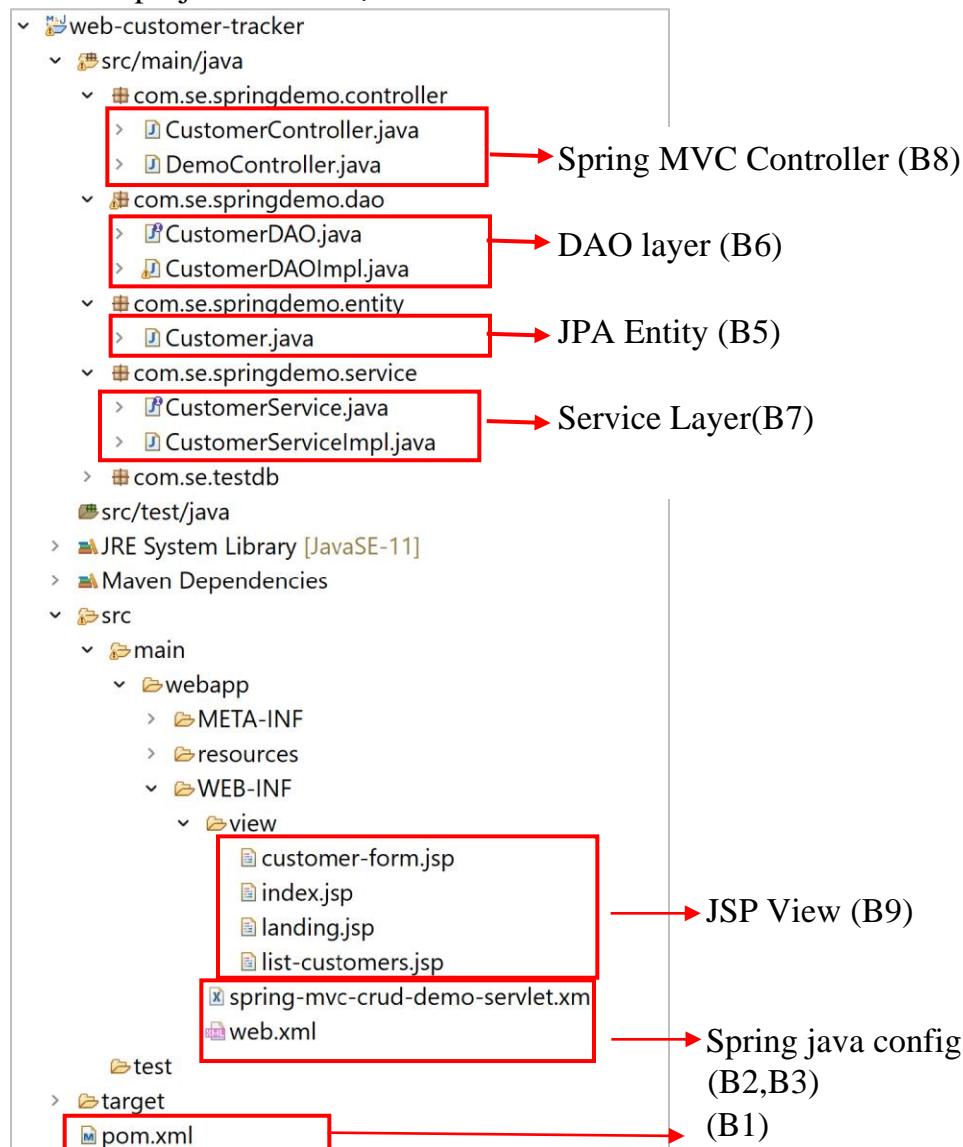
{*Bài tập trên Slide Lý thuyết 5.14. Spring MVC – Web App – Hibernate*}

Spring MVC với các thao tác CRUD thêm, xóa, sửa với cơ sở dữ liệu.

Database web-tracker-customer

B1: Tạo Project Maven Web Application

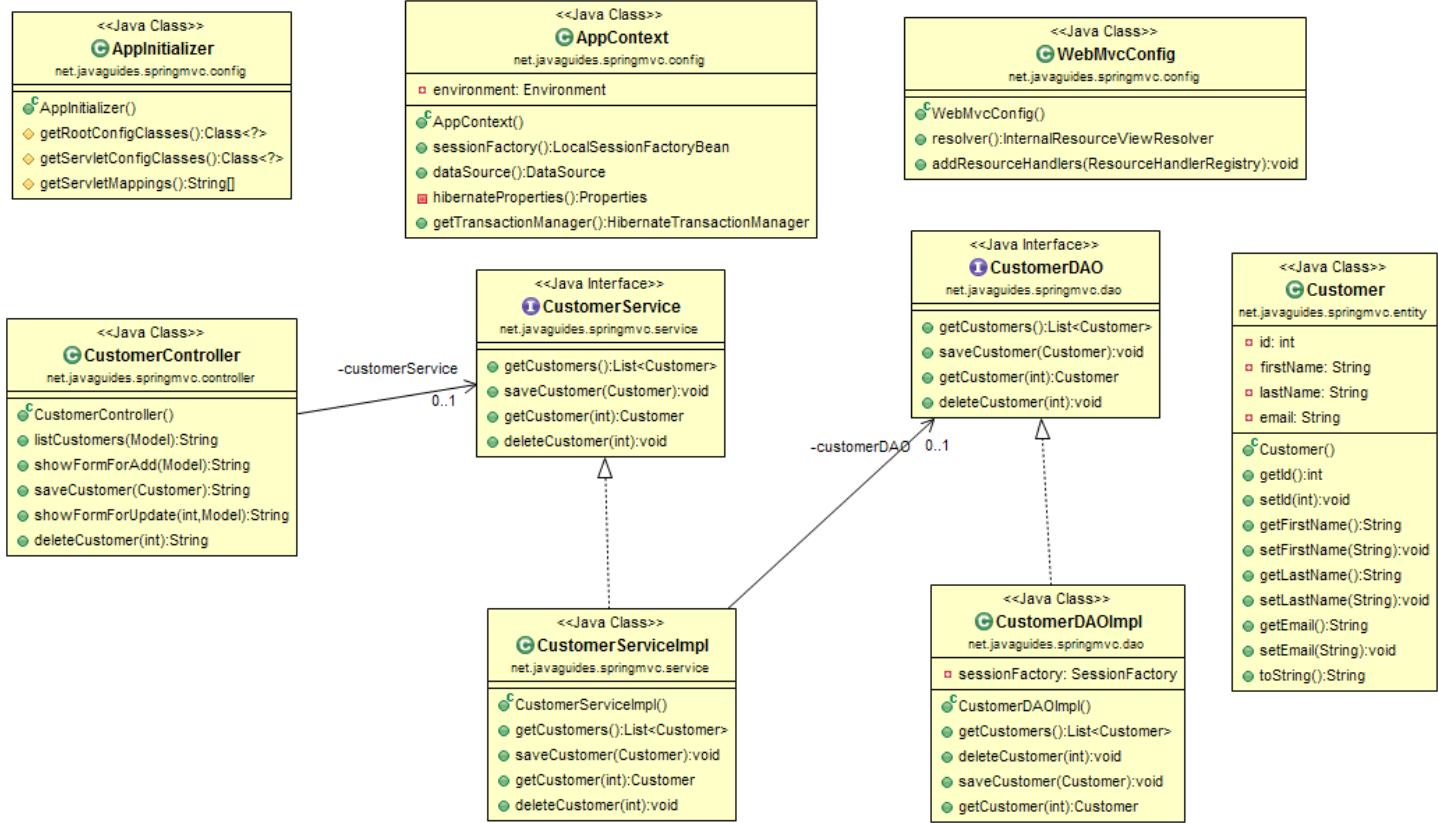
Cấu trúc project sau khi tạo:



B2: khai báo dependencies

```
<!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-tx -->
<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
<!-- https://mvnrepository.com/artifact/javax.servlet/servlet-api -->
<!-- https://mvnrepository.com/artifact/javax.servlet.jsp/javax.servlet.jsp-api -->
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<!-- https://mvnrepository.com/artifact/com.mchange/c3p0 -->
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-c3p0 -->
<!-- https://mvnrepository.com/artifact/com.mchange/mchange-commons-java -->
```

Class diagram cách thức hoạt động của project



B3: web.xml: : Khai báo cấu hình cần thiết của ứng dụng Web

(Câu hình 1. DispatcherServlet)

{1. Cấu hình Spring dispatcher servlet, 2. Mapping cho Spring dispatcher servlet }

B4: WEB-INF/spring-mvc-crud-demo-servlet.xml:

Khai báo cấu hình tối thiểu về Spring MVC (Câu hình 2. servlet, servlet mapping của project)

{3. Scan-context, 4. Annotation validation, 5. ViewResolver : như bài 2}

Khai báo thêm các bean cho các class để sử dụng Hibernate và các câu truy vấn HQL như câu hình bên dưới

Step 1: Xác định DataSource/chuỗi kết nối đến connection pool thông qua
class="com.mchange.v2.c3p0.ComboPooledDataSource"

Step 2: Thiết lập Section Factory để sử dụng các câu truy vấn HQL
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean"

Step 3: Quản lý kết nối Hibernate
class="org.springframework.orm.hibernate5.HibernateTransactionManager"

Step 4: Cấu hình các file resource sử dụng trong project như: hình ảnh, css,...

```

<?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: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.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd">
    <!-- 3. Add support for component scanning -->
    <context:component-scan
        base-package="com.se.springdemo" />
    <!--4. Add support for conversion, formatting and validation support -->
    <mvc:annotation-driven />

    <!-- 5. Define Spring MVC view resolver -->
    <bean

        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/view/" />
        <property name="suffix" value=".jsp" />
    </bean>
<!-- Step 1: Define Database DataSource / connection pool -->

    <bean id="myDataSource"
        class="com.mchange.v2.c3p0.ComboPooledDataSource"
        destroy-method="close">
        <property name="driverClass"
            value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
        <property name="jdbcUrl"
            value="jdbc:sqlserver://localhost:1433;databaseName=web_customer_tracker" />
        <property name="user" value="sa" />
        <property name="password" value="123" />
        <!-- these are connection pool properties for C3P0 -->
        <property name="minPoolSize" value="5" />
        <property name="maxPoolSize" value="20" />
        <property name="maxIdleTime" value="30000" />
    </bean>

```

```
<!-- Step 2: Setup Hibernate session factory -->
```

```
<bean id="sessionFactory"
      class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
    <property name="dataSource" ref="myDataSource" />
    <property name="packagesToScan"
              value="com.se.springdemo.entity" />
    <property name="hibernateProperties">
      <props>
        <prop
          key="hibernate.dialect">org.hibernate.dialect.SQLServer2012Dialect</prop>
          <prop key="hibernate.show_sql">true</prop>
        </props>
      </property>
    </bean>
```

```
<!-- Step 3: Setup Hibernate transaction manager -->
```

```
<bean id="myTransactionManager"
      class="org.springframework.orm.hibernate5.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
  </bean>
```

```
<!-- Step 4: Enable configuration of transactional behavior based on
annotations -->
```

```
<tx:annotation-driven proxy-target-class="true"
  transaction-manager="myTransactionManager" />
<!-- Add support for reading web resources: css, images, js, etc ... -->
<mvc:resources location="/resources/"
  mapping="/resources/**"></mvc:resources>
```

```
</beans>
```

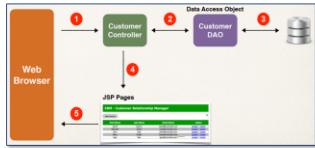
List Customers Development Process



1. Create Customer.java

2. Create:

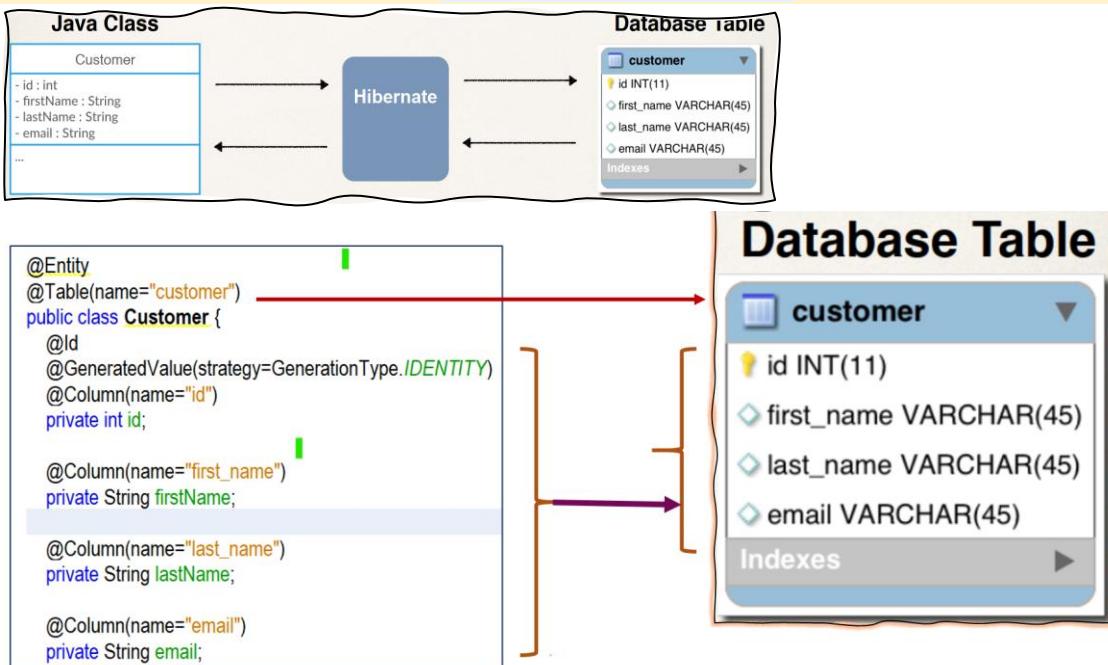
- Interface **CustomerDAO.java**
- Class **CustomerDAOImpl.java**



3. Create controller CustomerController.java

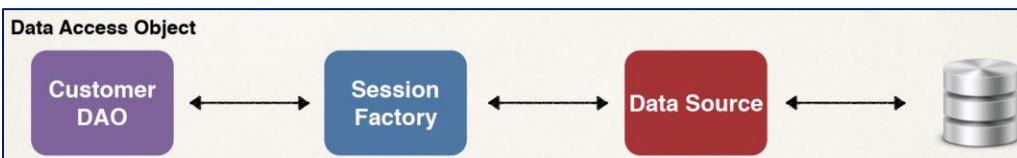
4. Create view list-customer.jsp

B5: Tạo Entities Customer trên com.se.springdemo.entity (1.)



B6: Tạo DAO Layer (2.)

DAO Layer chứa các methods thực hiện các chức năng CRUD trên Entity Customer, sử dụng Hibernate SessionFactoryBean để thực hiện các câu truy vấn HQL (Hibernate Query Language). Hibernate SessionFactoryBean cần DataSource để thực hiện kết nối (đã định nghĩa bean trong file xml Step 1.).



- Tạo Interface CustomerDAO trong com.se.springdemo.dao

```
package com.se.springdemo.dao;
import java.util.List;

import com.se.springdemo.entity.Customer;
public interface CustomerDAO {
    public List<Customer> getCustomers();
    public void saveCustomer(Customer theCustomer);

    public Customer getCustomer(int theId);
    public void deleteCustomer(int theId);
}
```

- Tạo class CustomerDAOImpl implements CustomerDAO trong com.se.springdemo.dao

```

package com.se.springdemo.dao;

import java.util.List;□
@Repository
public class CustomerDAOImpl implements CustomerDAO {
    // need to inject the session factory
    @Autowired
    private SessionFactory sessionFactory;
    @Override
    @Transactional
    public List<Customer> getCustomers() {
        // get the current hibernate session
        Session currentSession = sessionFactory.getCurrentSession();
        // create a query ... sort by last name
        Query<Customer> theQuery =
            currentSession.createQuery("from Customer order by lastName",
                                      Customer.class);
        // execute query and get result list
        List<Customer> customers = theQuery.getResultList();
        // return the results
        return customers;
    }

    @Override
    public void saveCustomer(Customer theCustomer) {
        // get current hibernate session
        Session currentSession = sessionFactory.getCurrentSession();
        // save/update the customer ... finally
        currentSession.saveOrUpdate(theCustomer);
    }

    @Override
    public Customer getCustomer(int theId) {
        // get the current hibernate session
        Session currentSession = sessionFactory.getCurrentSession();
        // now retrieve/read from database using the primary key
        Customer theCustomer = currentSession.get(Customer.class, theId);
        return theCustomer;
    }

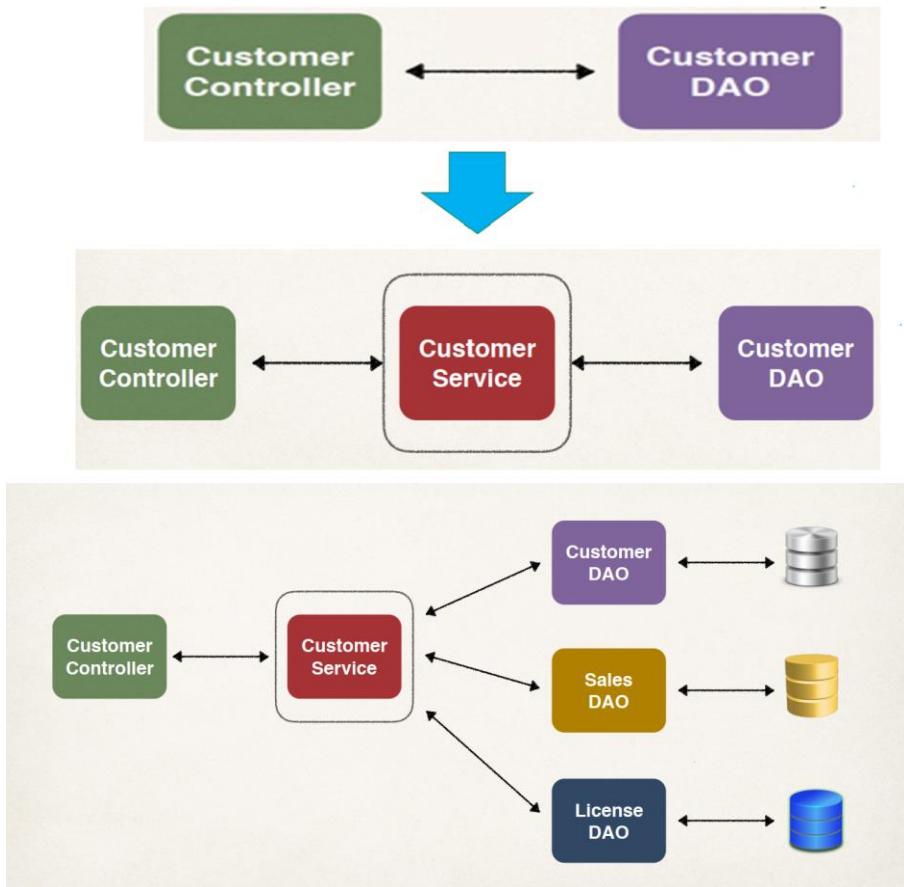
    @Override
    public void deleteCustomer(int theId) {
        // get the current hibernate session
        Session currentSession = sessionFactory.getCurrentSession();
        // delete object with primary key
        Query theQuery =
            currentSession.createQuery("delete from Customer where id=:customerId");
        theQuery.setParameter("customerId", theId);

        theQuery.executeUpdate();
    }
}

```

B7. Tầng Service Layer

- Thành phần trung gian giữa DAO và Controller
- Tích hợp các DataSource khác nhau.



- Tạo Interface `CustomerService` trong `com.se.springdemo.service`

```

package com.se.springdemo.service;

import java.util.List;

public interface CustomerService {
    public List<Customer> getCustomers();
    public void saveCustomer(Customer theCustomer);

    public Customer getCustomer(int theId);
    public void deleteCustomer(int theId);
}

```

- Tạo class `CustomerServiceImpl` implements `CustomerService` trong `com.se.springdemo.service`, gọi các phương thức từ DAO Layer

```

package com.se.springdemo.service;

import java.util.List;

@Service
public class CustomerServiceImpl implements CustomerService {
    // need to inject customer dao
    @Autowired
    private CustomerDAO customerDAO;
    @Override
    @Transactional
    public void saveCustomer(Customer theCustomer) {
        customerDAO.saveCustomer(theCustomer);
    }
    @Override
    @Transactional
    public Customer getCustomer(int theId) {
        return customerDAO.getCustomer(theId);
    }
    @Override
    @Transactional
    public void deleteCustomer(int theId) {
        customerDAO.deleteCustomer(theId);
    }
    @Override
    @Transactional
    public List<Customer> getCustomers() {
        return customerDAO.getCustomers();
    }
}

```

B8: Tạo Controller

Tạo Class CustomerController trong com.se.springdemo.controller

```

package com.se.springdemo.controller;

import java.util.List;
@Controller
@RequestMapping("/customer")
public class CustomerController {

    @Autowired
    // need to inject our customer service
    private CustomerService customerService;
    @PostMapping("/saveCustomer")
    public String saveCustomer(@ModelAttribute("customer") Customer theCustomer) {
        // save the customer using our service
        customerService.saveCustomer(theCustomer);
        return "redirect:/customer/list";
    }
}

```

```

@GetMapping("/showFormForUpdate")
public String showFormForUpdate(@RequestParam("customerId") int theId, Model theModel) {
    // get the customer from our service
    Customer theCustomer = customerService.getCustomer(theId);
    // set customer as a model attribute to pre-populate the form
    theModel.addAttribute("customer", theCustomer);
    // send over to our form
    return "customer-form";
}

@GetMapping("/showFormForAdd")
public String showFormForAdd(Model theModel) {
    // create model attribute to bind form data
    Customer theCustomer = new Customer();
    theModel.addAttribute("customer", theCustomer);
    return "customer-form";
}

@GetMapping("/delete")
public String deleteCustomer(@RequestParam("customerId") int theId) {

    // delete the customer
    customerService.deleteCustomer(theId);

    return "redirect:/customer/list";
}

@GetMapping("/list")
public String listCustomers(Model theModel) {
    // get customers from the service
    List<Customer> theCustomers = customerService.getCustomers();
    // add the customers to the model
    theModel.addAttribute("customers", theCustomers);
    return "list-customers";
}
}

```

Bg: Tạo View:

list-customer.jsp

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<title>List Customers</title>
<!-- reference our style sheet --&gt;
&lt;link type="text/css" rel="stylesheet"
      href="${pageContext.request.contextPath}/resources/css/style.css" /&gt;
&lt;/head&gt;
&lt;body&gt;
    &lt;div id="wrapper"&gt;
        &lt;div id="header"&gt;
            &lt;h2&gt;CRM - Customer Relationship Manager&lt;/h2&gt;
        &lt;/div&gt;
    &lt;/div&gt;
    &lt;div id="container"&gt;
        &lt;div id="content"&gt;
            <!-- put new button: Add Customer --&gt;
            &lt;input type="button" value="Add Customer"
                  onclick="window.location.href='showFormForAdd'; return false;" 
                  class="add-button" /&gt;
            &lt;%-- window.location.href returns the href (URL) of the current page
                Changing the value of the property will redirect the page.--%&gt;
        &lt;/div&gt;
    &lt;/div&gt;
&lt;/body&gt;
</pre>

```

```

<!-- add our html table here -->
<table>
    <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Email</th>
        <th>Action</th>
    </tr>
    <!-- loop over and print our customers -->
    <c:forEach var="tempCustomer" items="${customers}">
        <!-- construct an "update" link with customer id -->
        <c:url var="updateLink" value="/customer/showFormForUpdate">
            <c:param name="customerId" value="${tempCustomer.id}" />
        </c:url>
        <!-- construct an "delete" link with customer id -->
        <c:url var="deleteLink" value="/customer/delete">
            <c:param name="customerId" value="${tempCustomer.id}" />
        </c:url>
        <tr>
            <td>${tempCustomer.firstName}</td>
            <td>${tempCustomer.lastName}</td>
            <td>${tempCustomer.email}</td>
            <td>
                <!-- display the update link --> <a href="${updateLink}">Update</a>
                | <a href="${deleteLink}">
                    onclick="if (!confirm('Are you sure you want to delete this customer?')) return false">Delete</a>
                </td>
            </tr>
        </c:forEach>
    </table>

</div>
</div>

</body>
</html>

```

customer-form.jsp

```

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<!DOCTYPE html >
<html>
<head>
    <title>Save Customer</title>
    <link type="text/css"
          rel="stylesheet"
          href="${pageContext.request.contextPath}/resources/css/style.css">
    <link type="text/css"
          rel="stylesheet"
          href="${pageContext.request.contextPath}/resources/css/add-customer-style.css">
</head>
<body>
    <div id="wrapper">
        <div id="header">
            <h2>CRM - Customer Relationship Manager</h2>
        </div>
    </div>
    <div id="container">
        <h3>Save Customer</h3>
        <form:form action="saveCustomer" modelAttribute="customer" method="POST">
            <!-- need to associate this data with customer id -->
            <form:hidden path="id" />

```

```

<table>
    <tbody>
        <tr>
            <td><label>First name:</label></td>
            <td><form:input path="firstName" /></td>
        </tr>
        <tr>
            <td><label>Last name:</label></td>
            <td><form:input path="lastName" /></td>
        </tr>
        <tr>
            <td><label>Email:</label></td>
            <td><form:input path="email" /></td>
        </tr>
        <tr>
            <td><label></label></td>
            <td><input type="submit" value="Save" class="save" /></td>
        </tr>
    </tbody>
</table>
</form:form>
<div></div>
<p>
    <a href="${pageContext.request.contextPath}/customer/list">Back
        to List</a>
</p>
</div>
</body>
</html>

```

Bài 9. Spring MVC - Bài tập tổng hợp (2 bảng quan hệ 1-N)

Bài tập ôn lập trình Web nâng cao (Java) – Thời gian 120 phút

Quản lý thông tin về đê tài và giảng viên ra đê tài tốt nghiệp cho sinh viên ngành công nghệ thông tin được mô tả như sau:

Giảng viên bao gồm các thông tin như *mã giảng viên, tên giảng viên, lĩnh vực nghiên cứu, số điện thoại* và *email của giảng viên*. Một giảng viên sẽ thực hiện nhiều đê tài khác nhau, mỗi một đê tài có các thông tin như *mã đê tài, tên đê tài, năm đăng ký thực hiện, hình ảnh minh họa, và mô tả của đê tài*.

GIANGVIEN(MAGV, TENGV, LINHVUCNGHIENCUU, DIENTHOAI, EMAIL)
DETAI(MADETAI, TENDETAI, NAM, MOTADETAI, URLHINH, MAGV)

Xây dựng ứng dụng dùng mô hình MVC với công nghệ Java (*Spring Web MVC* kết hợp với SQL Server và Web server Tomcat thực hiện các công việc sau:

- Tạo cơ sở dữ liệu với các quan hệ như mô tả trên và nhập dữ liệu cần thiết để kiểm tra chương trình.
- Dùng HTML/CSS tạo Layout chung cho chương trình. Layout tối thiểu:

Hình ảnh Logo + Menu chức năng (Danh sách | Thêm | Chức năng quản lý)

Nội dung

@Copywrite 2023

- Xây dựng các lớp Entity bao gồm các get/set cho các thuộc tính, constructors, và các annotation kiểm tra dữ liệu (dùng Hibernate và Java Validator)
- Lớp GiangVien: Mã, tên giảng viên là bắt buộc, số điện thoại từ 9-11 số và email phải đúng chuẩn email yourname@yourcompany.com/org/net.vn.
- Lớp DeTai: Tên đề tài, mã đề tài, năm bắt đầu là NOT NULL. Tên đề tài mỗi từ bắt đầu bởi ký tự hoa, có số nhưng không có ký tự đặc biệt và không quá 100 ký tự (dùng RegularExpression).
- Thực hiện các cấu hình cho Project, cấu hình cho Spring/JavaServer Faces.
- Tạo lớp (trong Controller cần @Autowired đến lớp DAO này) cho phép thao tác với CSDL, lớp này bao gồm các phương thức: *lấy danh sách giảng viên, lấy danh sách đề tài theo mã giảng viên, lấy thông tin chi tiết đề tài theo mã đề tài, thêm đề tài mới, thêm giảng viên mới, xóa đề tài theo mã đề tài*.
- Tạo giao diện Web cho phép thực hiện thao tác liệt kê dữ liệu: Liệt kê dữ liệu danh sách các đề tài theo các giảng viên. Có thể thực hiện theo 1 trong 2 cách sau:
 - Ban đầu hiển thị ComboBox danh sách giảng viên khi chọn giảng viên cụ thể, hiển thị danh sách tên các đề tài.
 - Hiển thị danh sách GV, mỗi giảng viên có từng danh sách con bao gồm tên các đề tài.
- Nhấn vào link của tên đề tài, cho phép xem chi tiết thông tin của từng đề tài, thông tin này lấy từ CSDL.
- Tạo giao diện Web cho phép thực hiện giao diện thao tác với việc thêm dữ liệu
 - a. Tạo Form thêm thông tin đề tài mới với đầy đủ thông tin phù hợp với CSDL. Nếu người dùng không chọn hình ảnh, lấy hình ảnh mặc định, danh sách GV phải được chọn từ ComboBox. Kiểm tra dữ liệu nhập phía Server dùng Hibernate và Java Validation trên Form: Tên đề tài mỗi từ bắt đầu bởi ký tự hoa, có số nhưng không có ký tự đặc biệt và không quá 100 ký tự (dùng RegularExpression). Thông tin nhập vào hợp lệ sau khi nhấn nút “Lưu” sẽ được thêm vào cơ sở dữ liệu và hiển thị trang danh sách với kết quả mới.
 - b. **Upload hình khi thêm đề tài mới (optional).**
 - c. Form thêm giảng viên mới. Mã, tên giảng viên là bắt buộc, số điện thoại từ 9-11 số và email phải đúng chuẩn email yourname@yourcompany.com/org/net.vn. Thông tin nhập vào hợp lệ sau khi nhấn nút “Lưu” sẽ được thêm vào cơ sở dữ liệu và hiển thị trang danh sách với kết quả mới.

Tạo giao diện Web dùng Layout chung cho phép thực hiện chức năng quản lý: thao tác hủy dữ liệu liên quan đến đề tài. Khi chọn chức năng này, hiển thị danh sách và kèm theo chức năng hủy thông tin đề tài khỏi cơ sở dữ liệu.